DESIGN OF THE SPECIAL FAST RECONFIGURABLE CHIP USING COMMON FPGA

Lukáš Sekanina, Richard R□□i□ka
Department of Computer Science and Engineering, Brno University of Technology, Czech Republic
sekanina@dcse.fee.vutbr.cz
ruzicka@dcse.fee.vutbr.cz

***Abstract**. Some applications require chips with fast partial reconfiguration. These requirements are traditionally satisfied by a special chip design, but it is usually a very expensive solution. This paper describes a new approach. Special fast partially reconfigurable chip is implemented with a common FPGA. The format of the configuration bit stream is suggested and optimized according to the given task. Result chip offers many good properties, but some problems with scalability can appear.*

1 Introduction

New bio-inspired hardware requires some new features to be covered by digital circuits. Evolvable hardware [1] is a new technology, where circuit connection is subject to evolution. It means that hundreds of connections must be downloaded and evaluated in the physical chip per second. Connections are encoded into the bit strings because evolution looks for the bit string, which represents a final content of the circuit configuration memory. Following points present the main requirements (and reasons for them) for the potential chips:

- The chip consists of programmable elements and their interconnection is also programmable. Configuration memory is implemented as RAM. **Reason:** Evolution looks for the best functions of elements and their interconnection. Solutions are encoded into bit strings, which are used as configuration bits of programmable devices. RAM technology doesn't restrict the number of reconfigurations.
- These elements are suitable for solution of the given task. **Reason:** Different tasks need different basic blocks (offered functions, data width) to constitute application, e.g. gates, register transfer level (RTL), cellular arrays.
- Format of the configuration information is known and simple. **Reason:** Configuration information is created on-line and a relatively simple machine (e.g. microcontroller) should be able to configure entire chip. An unknown format causes that only several predefined configurations can be used (which are created before in some design tool).
- Random bits, downloaded as the configuration bits, never destroy the chip. **Reason:** It is the main condition in case that a connection is produced by evolutionary approach (i.e. it is not produced by traditional engineering techniques).
- Configuration process is quick (parallel). **Reason:** The chip has downloaded many connections during evolutionary process, thus the configuration time is the most important chip feature.
- Partial reconfiguration is supported. **Reason:** If not, entire connection must be downloaded again — it takes a lot of time in the case of complex circuits. During partial reconfiguration, contrary to the full reconfiguration, execution can be kept.

Several solutions are considered:

- Special chip (e.g. [2,8]), which is implemented by some technology. It is easy to fulfill all requirements, but this solution is too expensive from our point of view (chip for experiments only). The resulting design can not be modified (in contrary to the Field Programmable Gate Array, FPGA) — it is other disadvantage.
- Common FPGA (e.g. the 4000, Spartan or Virtex families of Xilinx) which is used in usual way does not satisfy most of the requirements. These chips usually exhibit these undesirable properties: The connection design must be created in some developmental tool, the format of the configuration bit stream is unknown, partial reconfiguration is not supported, and a configuration process is slow.

The new variant, GeneticFPGA — a Java based tool for evolving stable circuit on XC4000EX and XC4000XL devices [3] — is not yet available.

- Xilinx's XC6200 family appears to be the best platform for our experiments. Sophistic configuration format could bring a small difficulty sometimes. Generally, the CLB (Configurable Logic Block) structure is not the best basic element for each task. The main problem is out of technical parameters — these chips are not produced nowadays.

It is necessary to accept a new concept to satisfy our requirements; Layzell [4] investigates several ways. According to the previous list of requirements, special reconfigurable chip is implemented on top of the normal FPGA. This concept has been firstly described for evolvable hardware [5], but it is the general approach and more details should be explained.

The paper is organized as follows. Section 2 gives an overview of the chip building according to proposal concept, while Section 3 emphasizes a case study (the chip for pixel prediction), results of high level simulations and preliminary analysis. Section 4 introduces VHDL model and section 5 gives an example of connection. Results of the routing onto several widely used FPGAs are subject of the Section 6. Section 7 summarizes main conclusions.

2 Building the chip

The chip architecture depends on the given task, but this section brings some general aspects. Our final goal is the virtual partially quickly reconfigurable chip, implemented, let us say, with Xilinx XC4000 family. The structure of such chip is designed in some developmental tool and resulting configuration stream is downloaded into configuration memory through Xilinx's configuration inputs. Thus, a new reconfigurable machine is established. Machine design supposes that some pins are considered as the configuration inputs and another as data or control inputs and outputs. From the user (of our machine) point of view, there are no other pins of the chip. Of course, physical Xilinx chip has even more pins (e.g. configuration pins), but they are not "visible" for the user. A new chip must inside determine the set of the function blocks (FB), which are used to construct the circuit, rules of their interconnections and ways of the input/output connections. Further it defines structure and writing mechanisms of the configuration memory. The most important parts are the logic circuits, which configure function blocks according to data in the configuration memory. Figure 1 shows architecture of a new reconfigurable chip. Configuration memory is implemented as an array *m* x *n* of the static memory elements. Each of *m* rows represents a configuration of the circuit element (functional block). The *WR* signal causes that the FB at position given by address bits is reconfigured — thus the partial reconfiguration is achieved. The *n*-bit configuration information (gene) determines function of the FB and the connection of its inputs. In this strongly constrained structure, any random configuration information can never destroy the chip.

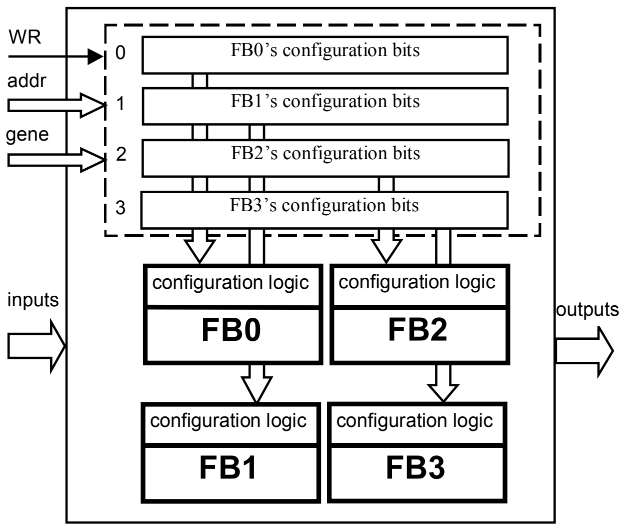

Fig. 1. Architecture of the reconfigurable chip, which consists of 4 programmable function blocks. A row of the configuration memory defines function and input connections for a block.

Chip under design should be implemented in the Xilinx XC4000 family, thus the CLB is the basic element of the physical implementation. The CLB can be configured (in the simplest way) as a boolean function of five variables or two independent boolean functions of four variables or a small (several bits) memory. It contains two flip-flops. Two main problems must be solved for each application. First, how many CLBs are needed for solution, i.e. do common chips support such a number of CLBs? This number determines the complexity of possible applications. Note that most of CLBs will be used for implementation of the configuration logic. The second problem is associated with timing. The number of combinational degrees between two flip-flops determines maximal frequency and again, configuration logic has the main influence to this number of degrees.

3 Case study — evolvable circuit

This case study presents simple evolvable circuit, which implements evolvable combinational function. Note that this paper doesn't explain why the chip is constructed in this way. The number of the functional blocks, selection of their functions and interconnection are the results of the higher-level application design. Special simulators for this chip and the task have been developed and many simulations have been done.

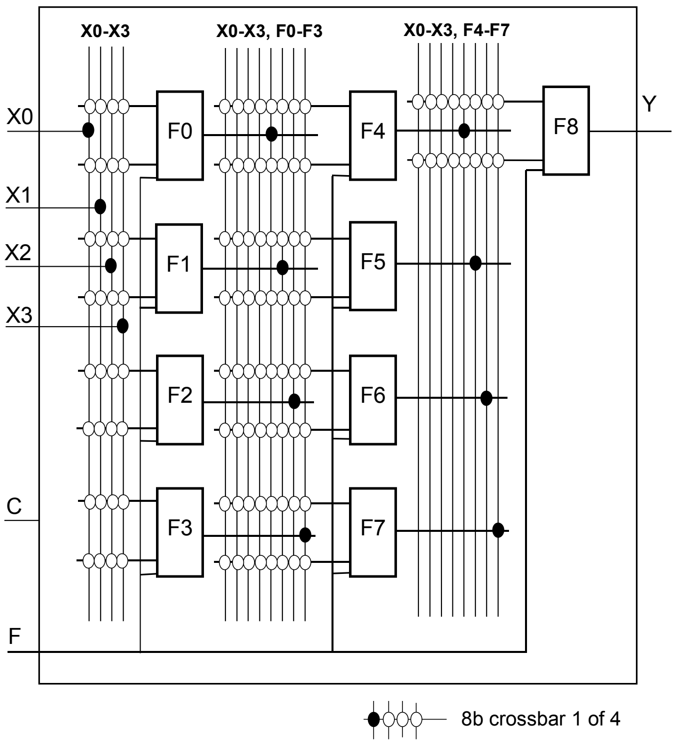

Fig. 2. The proposal model of reconfigurable circuit

The circuit in Figure 2 operates as a nonlinear predictor in the task of image compression [6]. The eight-bit pixel value (circuit's output *Y*) is predicted from pixel's four neighbors (four inputs *X0–X3* per eight bits). The evolution looks for an optimal function, which ensures the best source image reproduction in the data block of 16x16 pixels (an image is divided onto data blocks of 256 pixels). Content of the used part of the configuration memory with the best connection and some other information then represent 256 pixels in the compressed file. Note that in some cases, a connection can consist of a single functional block (i.e. a connection is described by ten bits) and in this way, the best compression ratio is achieved. Every function block can implement one of sixteen functions (all possible combinational functions over two bits) over its two 8-bits inputs. The FB input may be connected to one of the circuit inputs or to one of the FB outputs in the previous column. Gene (ten bits) given to the function block has the following structure: FFFFBBBAAA, where FFFF is the function selector (16 combinations), AAA encodes the place of the first operand connection (the value 0–3 is one of the circuit inputs, 4–7 is one of the outputs of the FBs in the previous column) and BBB is used for the second operand in the same way. Symbol F represents the connection of the configuration memory.

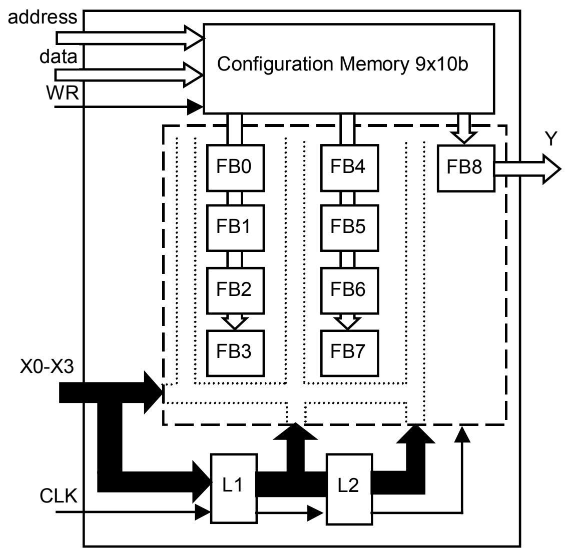

Fig. 3. The architecture of the reconfigurable chip for the pixel prediction

Figure 3 shows the entire virtual chip diagram. Nine function blocks are used, thus configuration memory contains 90 bits and can be configured in nine cycles. In the case of pipelined execution, columns of the function blocks are considered as stages. It is necessary to separate them by input data latches L, because the first stage operates with actual inputs at time *t*, the second stage with inputs at time *t-1* and the third stage with inputs at time *t-2*. The first result Y is gained after three clocks, next results are available per a clock. Now, preliminary time analyses can be discussed: It is possible to obtain a prediction of the block of data (16x16 pixels = 256 values) in the 9+3+255=267clocks (full reconfiguration is included).

Software simulation (i486/100MHz) takes about 5.5ms for this prediction. Simulation library, developed in C++, contains well parameterized simulator of the reconfigurable chip under design and genetic algorithm. This library may be used for different application. An example, model application — lossy image compression — is implemented in C++/Builder under MS Windows and uses proposal library. Application enables to modify many parameters of compression (e.g. data block size, a number of FB in connection, parameters of evolution) and presents obtained results and statistics. Thus, it is possible to investigate performance of evolvable hardware in given application and design optimal architecture to evolution.

Now, one can estimate the number of CLBs, which will be needed for the implementation. This estimation is based on analysis of figures 3, 4 and 5 (architectural details are explained in next sections). The new chip should take about 864 CLBs to cover the functional blocks (16x5 CLBs for the multiplexers and 16 CLBs for the FB's function per FB), 50 CLBs for the configuration memory, and 32 CLBs for two latches. The total number is about 946 CLBs, 76% is used for the reconfiguration logic, 9% for the memory and the latches, and only 15% is the useful logic. It is important that this number and type of CLBs are supported in the XC4000 family and design can go on by the next step, the VHDL description.

4 VHDL modelling

The whole circuit is described in the VHDL, using structural domain of modelling, because the main structure of the chip is already known. Only few parts are described behaviorally, such as elementary function generators, FFs, multiplexers and decoders. Their implementation was left to the synthesis tool.

The implementation has the same hierarchy as the RT level structure, described in section 3. The top-level entity, named *ReconfCirc*, represents the whole reconfigurable chip. It has four eight-bit data inputs (*X0–X3*) and one eight-bit data output (*Y*). For the reconfiguring, there is the input *GEN*, ten-bit wide. Through this input, the Configuration Memory content is loaded. To provide random access, the four-bit wide address input also exists. The writing of the configuration word (which is already present on the *GEN* input, of course) is synchronized by the rising edge of the *WR* signal. The clock signal for all FFs in the circuit inputs to the *clk* pin. All these ports are primary ports of the reconfigurable chip and they will be connected to the pins of the FPGA.

The reconfigurable chip consists of two entities: the Configuration Memory and the Evolvable Circuit.

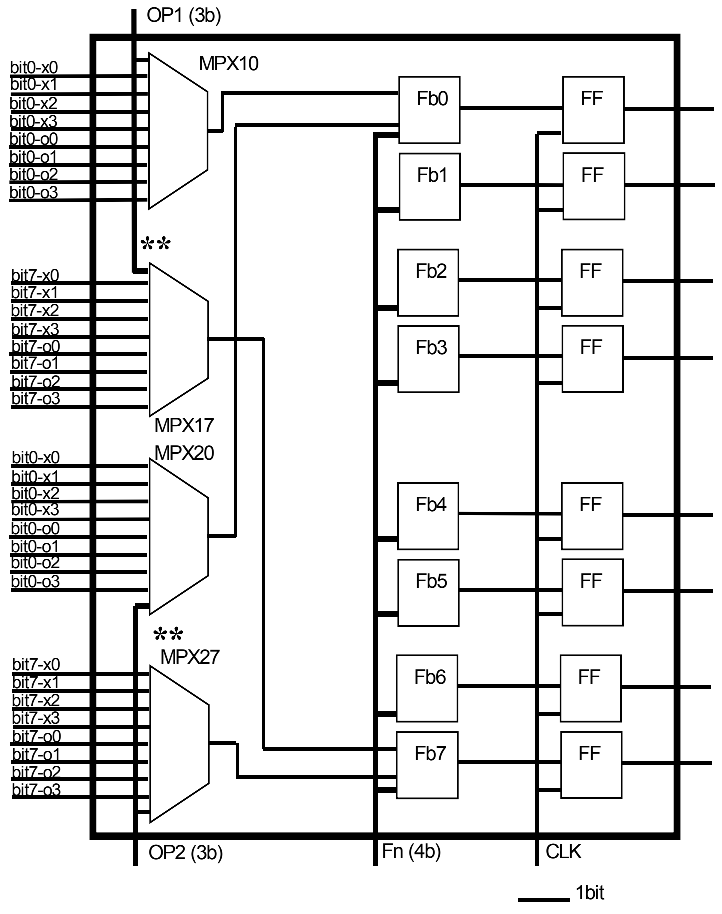

Fig. 4. Single functional block

The Configuration Memory (see Fig. 5) consists of the address decoder (AD) and the array of 9x10 memory cells. Each cell is implemented as a common D-type FF (described behaviorally) and a group of ten cells makes one word. As can be seen in the Fig. 5, two memory cells are in one CLB. Address decoder is the simple decoder BCD to 1-of-9. It is implemented as simple look-up table. The active output of the decoder enables write (strobe with the *WR* signal) to one group of cells. Outputs of all cells are taken out of the Configuration Memory and connected directly to the configuration inputs of all Functional blocks in the circuit. The outputs of the cells are permanently active and thus it is guaranteed that the selected structure and function of the evolvable circuit is kept unchanged. The change of one cell content modifies the structure and/or function immediately. The whole circuit can be reconfigured in 9 clock cycles (one clock cycle per FB).

Evolvable Circuit is a pipelined structure of nine functional blocks. The pipeline has three stages separated by two 32-bit wide latches. They can be, as described in section 3, bypassed by selecting another data input of the Functional Block — e.g. by connecting inputs of Functional Blocks in the second stage directly to primary data inputs of the circuit (with the suitable “gene” pattern). First two stages aim both four Functional Blocks, but the last stage has one Functional Block only. This Functional block is primarily designated to combine data from Functional Blocks in previous stages to single resulting value. Therefore, the output of this block is connected directly to the primary output of the circuit.

Each 32-bit wide latch is composed of 32 D-type FFs. Latches are controlled by main clock signal.

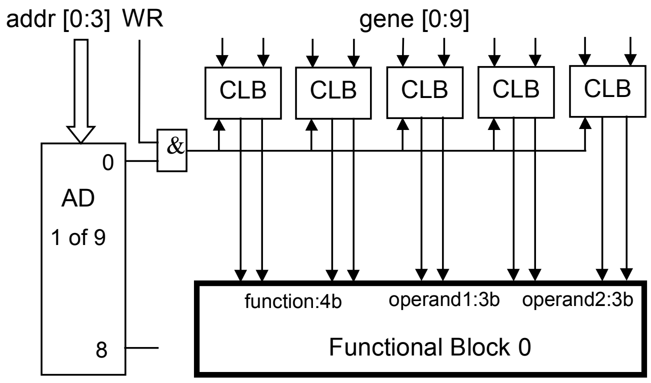

Fig. 5. Configuration memory – single row implementation: Outputs of the CLBs (configuration information) are directly connected to the Functional Block and determine its function.

Each Functional Block has eight eight-bit wide data inputs (four of them come from latches and four directly from the previous stage). Functional Block performs selected operation on two of data inputs. The lower part of "gene" pattern selects which of them are used — there are sixteen 8-to-1 multiplexers on the inputs (eight for the first and eight for the second argument of the operation). The upper part of "gene" pattern describes the type of operation, which is performed by the eight parallel Functional Units. They are implemented as simple look-up tables. For all combinations of both input bits, the look-up table contains all possible values of output bit. Finally, there are latching D-type FFs on the outputs of all Functional units. Structure of the Function Block is depicted in Figure 4.

5 Simulation of the model

There is an example of the circuit in Figure 6, realized with the Reconfigurable Chip. The circuit has three data inputs and one data output (all eight-bit wide). For the realization, only three Function Blocks were used (FBs 4 and 5, and the only FB in the last stage — FB8). First two FBs performs logical AND and OR operations, FB8 combines their results using the logical "Exclusive-OR" function.

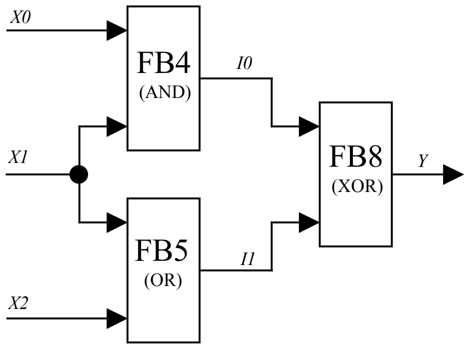

Fig. 6. Example of the circuit

The function of the circuit can be seen in Figure 7. First, there are three configuration cycles. During each cycle, suitable configuration word is prepared at the input *GEN* and loaded into configuration memory. With three rising edges of the *WR* signal, connections and function of FB4, FB5 and FB8 are defined. The format of the configuration word was described in section 3. Configuration word 208_{hex} means, that FB will perform operation "logical AND" with two arguments taken from inputs *X0* and *X1*. Word 391_{hex} means "logical OR" with *X1* and *X2*. Word $1AC_{\text{hex}}$ means "logical Exclusive-OR" with *I0* and *I1*.
After the configuration phase, the chip is ready to work. If there are right values on the data inputs, the result will be taken in three clock cycles, because there are three stages in the pipeline of the chip.

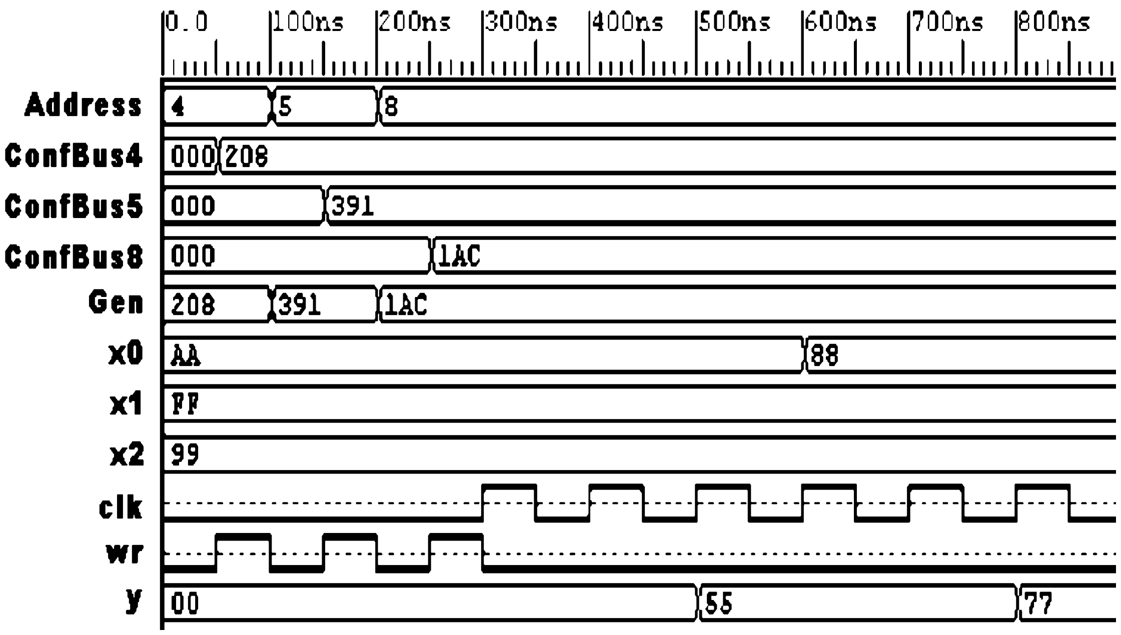

Fig. 7. Simulation timing diagram

6 FPGA routing results

Results of the routing onto several widely used FPGAs are show in Table 1. The absolutely minimal number of used CLBs is about 400. This result is indeed penalified by speed. About 40ns (23MHz) is minimal period of main clock signal only. When the better performance is required, the best result is less than 30ns using famous XC4000 FPGA family, but the size grows up to 750 used CLBs. When the new Xilinx's Virtex family is used, very nice results are achieved. There are about 500 Virtex slices[7] occupied only, and speed is up to 60MHz.

Table 1. Routing results

Chip	optimized for	No of CLBs	Clk period
XC4020E	area	405	42.8 ns
XC4028XLA	speed	757	29.8 ns
Virtex XCV300	area	479	16.5 ns

7 Conclusion

This paper presents the design of a very cheap fast partially reconfigurable chip, which is implemented using the non-expensive commercial chip (e.g. from the XC4000 family). Selection of the best hardware architecture for each task, definition of an ideal bit string format, and the possibility of changing the reconfigurable virtual machine are another advantages of the proposal concept. Generally, given task, which should be solved in this way, requires detail preliminary analyses (how many CLBs, timing) to find an acceptable ratio between space of reconfiguration logic and useful logic. Furthermore, commercial chips must support necessary number of CLBs. In our example, only 15% of the chip has been used for useful logic, the rest is an implementation overhead. It demonstrates the main disadvantage of this approach — problems with scalability. Further research should lead to more complex reconfigurable chips design, which consist of more sophisticated elementary function blocks. The improvement of the reconfiguration logic implementation is also the key point of our next research.

This research was performed with the research intent CEZ: J22/98:262200012 "Research of information and control systems" and the Grant Agency of the Czech Republic under No. 102/98/1463.

References

[1] SANCHEZ, E. - TOMASSINI, M. (Eds.): Towards Evolvable Hardware: The Evolutionary Engineering Approach. Springer-Verlag, Berlin, 1996, pp. 265.

[2] HIGUCHI, T. - KAJIHARA, N.: Evolvable Hardware Chips for Industrial Applications. Communication of the ACM, Vol. 42, 1999, No. 4, pp. 60–66.

[3] LEVI, D. - GUCCIONE, S.: GeneticFPGA: Evolving stable circuits on mainstream FPGA devices. In: Stoica, A., Keymeulen, D., Lohn, J. (Eds.): Proc. of The First NASA/DoD Workshop on Evolvable Hardware (EH'99), Published by IEEE Computer Society, Pasadena, California, USA, 1999.

[4] LAYZELL, P.: Reducing Hardware Evolution's Dependency on FPGAs. In Proceedings of MicroNeuro'99, 7th International Conference on Microelectronics for Neural, Fuzzy and Bio-inspired Systems, IEEE Computer Society, CA, 1999, pp. 171-178.

[5] SEKANINA, L. - DRÁBEK, V.: The Concept of Pseudo Evolvable Hardware. In: IFAC Workshop on Programmable Devices and Systems 2000, Ostrava, Czech Rep., 2000.

[6] SEKANINA, L: Evolvable Hardware as Non-Linear Predictor for Image Compression. In: Zelinka, I. (Ed.): Proc. of the 2nd Prediction Conference Nostradamus'99, Knihovna F. Bartoše, Zlín, Czech Rep., 1999, pp. 87–92.

[7] Xilinx Data Book 1999.

[8] KAJITANI, I. et al: A gate-level EHW chip: Implementing GA operations and reconfigurable hardware on a single LSI. In: Sipper, M., Mange, D., Perez-Uribe, A. (Eds.): Evolvable Systems: From Biology to Hardware, Lecture Notes in Computer Science 1478, Springer Verlag, 1998, pp. 1-12.