Complete FPGA Implemented Evolvable Image Filters

Jin Wang and Chong Ho Lee

Department of Information Technology & Telecommunication,
Inha University, Incheon, Korea
wangjin_liips@yahoo.com.cn

Abstract. This paper describes a complete FPGA implemented intrinsic evolvable system which is employed as a novel approach to automatic design of spatial image filters for two given types of noise. The genotype-phenotype representation of the proposed evolvable system is inspired by the Cartesian Genetic Programming and the function level evolution. The innovative feature of the proposed system is that the whole evolvable system which consists of evolutionary algorithm unit, fitness value calculation unit and reconfigurable function elements array is realized in a same FPGA. A commercial and current FPGA card: Celoxica RC1000 PCI board with a Xilinx Virtex xcv2000E FPGA is employed as our hardware platform. The main motive of our research is to design a general, simple and fast virtual reconfigurable hardware platform with powerful computation ability to achieve intrinsic evolution. The experiment results show that a spatial image filter can be evolved in less than 71 seconds.

1 Introduction

Inspired by the natural evolution, evolvable hardware (EHW) is a kind of electronic device which can change its inner architecture and function dynamically and autonomously to adapt its environment by using evolutionary algorithm (EA). According to the difference in the process of fitness evaluation, EHW can be classified into two main categories: extrinsic and intrinsic EHW. Extrinsic EHW uses software simulation to evaluate the fitness value of each individual and only the elite individual is finally implemented by hardware. In recent years, with the advent of programming logic devices, it is possible to realize intrinsic evolution by evaluating a real hardware circuit within the evolutionary computation framework: the fitness evaluation of each individual is directly implemented in hardware and the hardware device will be reconfigured the same number of times as the population size in each generation. Field Programmable Gate Array (FPGA) is the most commonly used commercial reconfigurable device for digital intrinsic EHW. Due to pipelining and parallelism, the very time consuming fitness evaluation process can be accessed more quickly in FPGA based intrinsic evolution than in software simulation. A lot of work has been done in the area of FPGA implemented intrinsic evolution. Generally, these approaches can be divided into two groups:

(1) FPGA is employed for the evaluation of the candidate individuals produced by evolutionary algorithm, which is executed in a host PC or an individual embedded processor. This idea has been proposed by Zhang [1] in the image filter evolution.

The author of [2] built an on-chip evolution platform implemented on a xc2VP7 Virtex-II Pro FPGA which equipped a PowerPC 405 hard-core processor to implement evolutionary algorithm. Other type of interesting approach has been reported in [3] in which the hardware reconfiguration was based on JBits API.

(2) Implementing complete FPGA based evolutions, this type of implementation integrates the evolutionary algorithm and the evaluations of the candidate circuits into a single FPGA. As an example, we can mention Tufte's and Haddow's research in which they introduced complete hardware evolution approach where the evolutionary algorithm implemented on the same FPGA as the evolving design [4]. Running complete evolution in FPGA has also been reported by Sekanina for different applications [5, 6, 7]. In his works, the full evolvable system was implemented on the top of FPGA which was named as virtual reconfigurable architecture. However, his systems were based on a specialized hardware platform-COMBO6 card which was custom-made for a special project and not achievable for most of researchers.

In this paper, we describe how to design a complete FPGA implemented evolvable system using a commercial and general FPGA card-Celoxica RC1000 PCI board [8]. Our evolvable system was designed for evolving spatial image operators. A lot of works have been done in the area of evolving spatial image filter by extrinsic evolution approach [9] and intrinsic evolution approaches [1] [6] because of their potential merit in research and industry. The objective of this paper is to exhibit the potential ability of our proposed system in the complicated application of evolving spatial image operators.

The rest of this paper is organized as follows: in the next section, we discuss the problem domain and the idea of the proposed approach. The hardware realization of the evolvable image filter is described in Section 3. The experimental results are presented in Section 4. Finally, Section 5 concludes the paper.

2 Intrinsic Evolution on FPGA

Fitness evaluation is generally computationally expensive and the most time consuming part in the evolutionary process of evolvable hardware. To speed up fitness evaluation, with the rapid development of reconfigurable devices, such as FPGA, the idea of hardware based intrinsic evolution has been proposed in the recent years. The system clock frequency of FPGA is normally slower than personal computer (PC), but it has an inherently parallel structure with flexible communication. This feature of FPGA which is similar to those in the biological neural networks offers it powerful computational ability to superior to the conventional personal computer. FPGA are widely accessible reconfigurable devices. However, in the most of applications of intrinsic evolution, there have some common problems when FPGA is considered as a hardware platform: (1) a popular idea of EHW is to regard the configuration bit string of FPGA as a chromosome for EA. This makes the chromosome length be on the order of tens of thousands of bits, rendering evolution practically impossible using current technology. (2) In today's FPGA, under the demand for 1000s or even more than 100,000s of reconfigurations to evolve a result circuit, the configuration speed of current reconfigurable device becomes an obvious bottleneck. Partial reconfiguration technology seems to be a solution to this problem for the reconfiguration process is faster than the complete reconfiguration. With the exit of Xilinx XC6200 family, JBits was introduced by Xilinx to make this work easier [3]. However, JBits is too complicated and still too slow to achieving more than 100,000s of reconfigurations in a reasonable time.

To conquer the mentioned problems, the virtual reconfigurable circuit technology [10] which is inspired by the Cartesian Genetic Programming (CGP) [11] is employed in our application as a kind of rapidly (virtual) reconfigurable platform utilizing conventional FPGA for digital EHW. For making FPGA be evolvable at a higher level and limit the size of chromosome, in virtual reconfigurable circuit, the FPGA cells are arranged into a grid of sub-blocks of cells (it is named as function elements array in the following sections). Compare with directly encoding the configuration bit string of FPGA as chromosome, the functions of each sub-block and the networks connections of the sub-blocks grid are encoded as chromosome of EA. For achieving complete hardware evolution, both of the evolutionary algorithm and the virtual reconfigurable circuit are implemented on the same FPGA to construct an intact evolvable system. In this scheme, the FPGA configuration bit string will not be changed during evolution, but rather the content of the internal registers which store the configuration information of the virtual reconfigurable circuit (which is generated by EA as chromosomes). The most obvious advantage of complete hardware evolution is that the evolvable system is complete pipelined which conquers the bottleneck introduced by slow communication between the FPGA and a personal computer and a very fast reconfiguration can be achieved (in several system clocks).

3 Implementing Evolvable System on RC1000 PCI Board

Celoxica RC1000 PCI board (as shown in Fig. 1) which has been successfully applied as a high performance, flexible and low cost FPGA based hardware platform for different computationally intensive applications [12, 13, 14] is employed as our experimental platform. Celoxica RC1000 PCI board supports traditional hardware design language: e.g. VHDL and Verilog. On the other hand, a new C like system description language called Handel-C introduced by Celoxica [8] is also supported by this board, which allows users who is not familiar with hardware design to focus more on the specification of an algorithm rather than adopting a structural approach to coding. In this paper, VHDL is employed as our design language.

Fig. 1. The Celoxica RC1000 PCI board with a Virtex xcv2000E FPGA

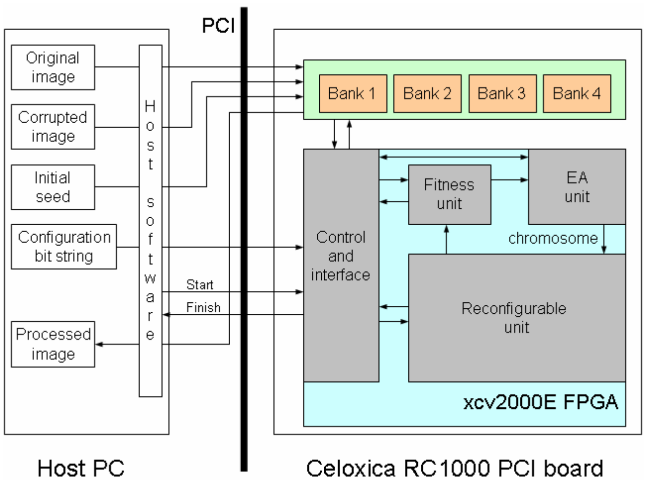


Fig. 2. Organization of the proposed evolvable system

The organization of the architecture on Celoxica RC1000 PCI board is given in Fig. 2. This architecture makes use of the on board Virtex xcv2000E FPGA chip and has 8Mbytes of SRAM which is directly connected to the FPGA in four 32-bit wide memory banks. All memory banks are accessible by the FPGA and host PC. The host software (it is a C program) that controls the flow of the data runs on the host PC. The corrupted image and original image are rearranged by the host software to support neighborhood window operations which will be detailed in section 3.1. The rearranged corrupted image, original image and the initial seed (for generating random numbers in the evolutionary process) are read and stored into the memory banks 1-3 in advance. Then the design configuration bit string is read and downloaded to the FPGA. Once this process is accomplished, the host software signals the complete FPGA implemented evolvable system to start the evolutionary operations. In Fig. 2, the proposed evolvable system is composed of four components: Reconfigurable unit, Fitness unit, EA unit and Control and interface. In the evolvable system, all operations are controlled by the control and interface which communicates with the on board SRAM and the host software. The EA unit implements the evolutionary operations and generates configuration bit string (chromosomes) to configure the reconfigurable unit. The reconfigurable unit processes the input 9×8 bits gray scale image pixels and its function is reconfigurable. Fitness unit calculates individual fitness by comparing the output image from the reconfigurable unit with the original image. Once the system evolution is finished, the evolvable system signals the host software the completion of the operation. The result image which is stored in memory bank 4 is then transmitted to the host software and saved as the result image file.

3.1 Reconfigurable Unit

The reconfigurable unit processes nine 8 bits input image pixels (labeled in Fig. 3, 4 as I0-I8) and has one 8 bits pixel output. It means any one pixel of the filtered image is generated by using a 3×3 neighborhood window (see Fig. 3 for an example, an output pixel is generated by processing its corresponding input pixel and 8 neighbors). By sliding the 3×3 neighborhood window one pixel by one pixel in the pixels array of the input corrupted image, the image can be processed.

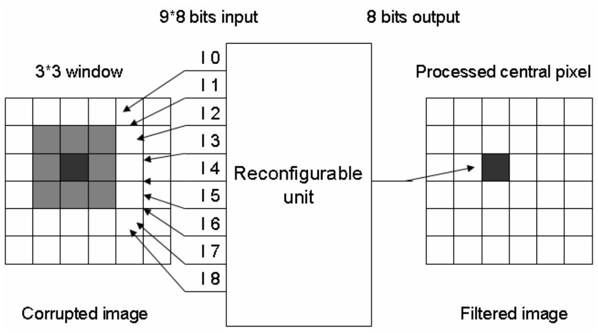


Fig. 3. Neighborhood window operation for image processing

The genotype-phenotype representation scheme in our proposed system is very similar as the Cartesian Genetic Programming [11]. The system genotype is a linear binary bit string which defines the connections and functions of a function elements (FE) array. The genotype and the mapping process of the genotype to phenotype are illustrated in Fig. 4. The main advantage of this representation of a program is that the genotype representation used is independent of the data type used in the phenotype. This feature brings us a generic and flexible platform as for a different application one would just need to change the data type, leaving the genotype unchanged. In hardware implementation, a $N\times M$ array of 2-inputs FEs has been implemented in the reconfigurable unit as system phenotype representation. In order to allow the design of evolvable image filter, the traditional CGP is expended to function level [15] and the FEs with 8 bits datapaths are employed.

In our experiment, the FEs array consists of 7 FEs layers from system input to output. Except for the last layer, 8 uniform FEs are placed in each layer. The last layer only includes one FE. The input connections of each FE are selected using 8:1 multiplexers. FE's two inputs in layer *l* (*l*=2, 3, 4, 5, 6, 7) can be connected to anyone output of FEs in layer *l*-1. In layer 1, the first input 8:1 multiplexer of a FE is constrained to be connected with system inputs I0-I7 and the second 8:1 multiplexer links system input I1-I8. Each FE can execute one of 16 functions which are evident from Fig. 4. Flip-flop is equipped by each FE to support the pipeline process and the reconfiguration of the FEs array is performed layer by layer, which was detailed in [10]. The active function of each FE and its two input connections are configurable and determined by the configuration bit string which is uploaded from EA unit as chromosome. By continuously altering the configuration bit string, the system can be evolved. The encoding of each FE in the chromosome is 3+3+4 bits. For the FEs array consisting of 49 units, the intact chromosome length is $(6\times 8\times 10)+10=490$ bits.

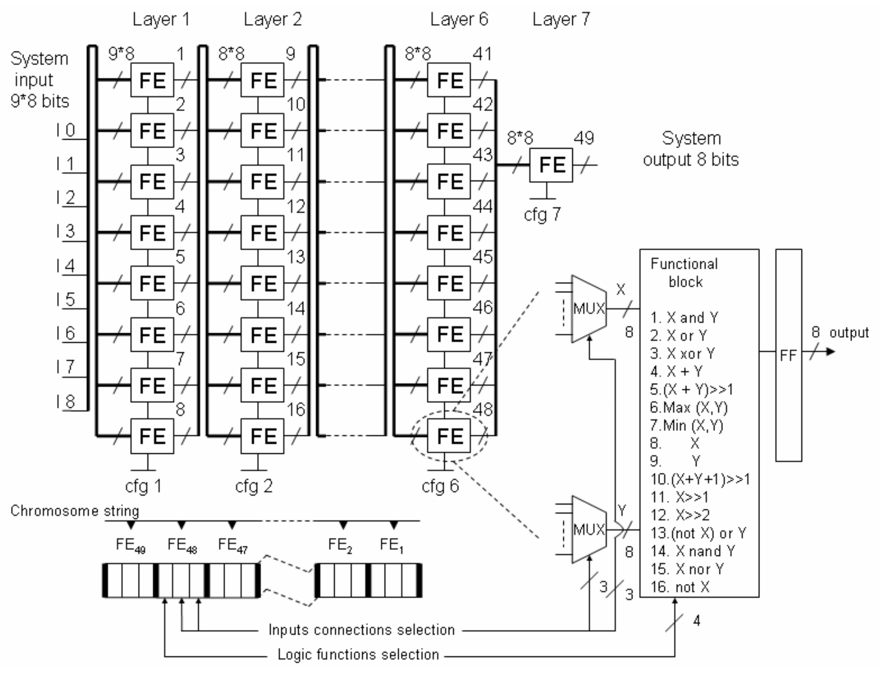
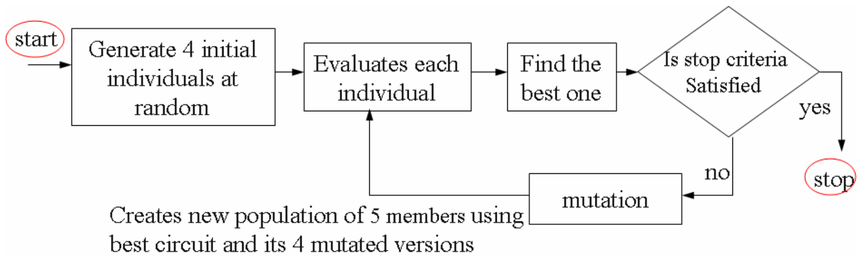


Fig. 4. Mapping chromosome string into the function elements array

3.2 Evolutionary Algorithm Unit

The evolutionary algorithm employed in EA unit is according to the $1+\lambda$ evolutionary strategy, where $\lambda=4$. In our experiment, evolution is only based on the mutation and selection operators, crossover is not taken into account. The Flow diagram of EA operations is presented in Fig. 5.

The initial population which includes 4 individuals is generated randomly with the initial seed according to the rules 90 and 150 as described by Wolfram [16]. To get the fitness value of each individual, each of them is downloaded to the reconfigurable unit to generate a candidate circuit, respectively. By comparing the candidate circuit output image with the original image, the fitness value of each individual can be



start
Generate 4 initial individuals at random
Evaluates each individual
Find the best one
Is stop criteria Satisfied
yes
stop
no
mutation
Creates new population of 5 members using best circuit and its 4 mutated versions

Fig. 5. Flow diagram of evolutionary algorithm

obtained. The individual with the best fitness in the initial population is selected out and the new population is generated by using the fittest individual and its 4 mutants. This process is repeated until the predefined generation number (8192) is exhausted.

3.3 Fitness Unit

The single evolutionary objective in our experiment is to minimize the difference between the filtered image and the original image which is uncorrupted by the noise. To measure the quality of the filtered image, the MDPP (mean different per pixel) based fitness function is implemented in the fitness unit. The original image size is $k \times k$ (k=256), but only a sub-area image of (k-2)$\times$ (k-2) pixels is considered, because the pixels at the image borders are untouchable for the 3×3 window and remain unfiltered. The MDPP based fitness function is described as follow:

$$fitness_{MDPP} = 255 \times (k-2)^2 - \sum_{i=1}^{k-2} \sum_{j=1}^{k-2} \left| v(i,j) - w(i,j) \right| \qquad (1)$$

In the Eq. (1), v denotes the filtered image and w denotes the original image. MDPP based fitness function is calculated by comparing the diversity of each pixel in the filtered image with its corresponding pixels in the original image.

4 Results

4.1 Synthesis Report

The evolvable system was designed by using VHDL and synthesized into Virtex xcv2000E FPGA using Xilinx ISE 6.3. The synthesized result was optimized for speed and shown in Table 1. According to our synthesis report, the proposed system can be operated at 71.169MHz. However, the actual hardware experiment was run at 30MHz because of easier synchronization with PCI interface.

Table 1. Synthesis results in Virtex xcv2000E

Resource	Used	Available	Percent
Slices	8596	19200	44%
Slice Flip Flops	5479	38400	14%
4 input LUTs	14038	38400	36%
bonded IOBs:	267	408	65%

4.2 Time of Evolution

As the pipeline process is supported by the proposed evolvable system, all EA operations time as well as reconfiguration time of FEs array could be overlapped by the fitness evaluation. The total system evolution time can be determined as:

$$t = t_{init} + \frac{(k-2)^2 \times p \times ngen}{f} \tag{2}$$

Where (k-2)^2 is the number of processed image pixels, p is population size, $ngen$ is the number of generations and t_{init} is the time needed to generate the first output pixel of the FEs array in pipeline process (several FPGA system clocks only).

4.3 Hardware Evolution Results

Only gray scale (8 bits each pixel) image of size 256× 256 pixels was consider in this paper. Two types of additive noise which were independent of the image itself were processed: Gaussian noise (mean 0 and variance 0.008) and salt & pepper noise (the image contains 5% corrupted pixels with white or block shots). Both of the mentioned noises were generated by using Matlab functions. Two types of image filters were evolved to process the proposed noises individually. The evolved image operators were trained by using original Lena and its filtered version.

Table 2. Results for Gaussian noise (mean 0 and variance 0.008)

Mutation rate	The best MDPP	Average MDPP	Evolution time
0.8%	7.32	13.35	70.5 s
1.6%	7.13	8.95	
3.2%	6.98	8.23	
6.4%	8.39	10.00	

Table 3. Results for salt & pepper noise (5% corrupted pixels with white or block shots)

Mutation rate	The best MDPP	Average MDPP	Evolution time
0.8%	1.08	7.64	70.5 s
1.6%	1.27	3.71	
3.2%	1.13	3.23	
6.4%	2.81	4.21	

In our experiment, each evolved result was achieved after 8192 generations of EA run. To measure the quality of the filtered image, the MDPP (mean different per pixel) function was employed. Different mutation rates were tested to explore the effect of the diversity of the EA parameter. The mutation rate is defined as the percentage of the genes of the entire chromosome string, which will undergo mutation. We performed 20 runs for each experimental setting and measured the best and average MDPP by comparing the filtered images with the original image. Table 2 and Table 3 summarize the evolved filters for Gaussian and salt & pepper noises, respectively.

An example evolved circuit for processing salt & pepper noise is shown in Fig. 6. After the image filter was evolved, to test the generality of the result filter which was trained by Lena, various test images set: baboon and cameraman were employed. Fig.7 presents the processed images set by the mentioned filter in Fig.6. As a comparison, the original corrupted images are also included in Fig.7.

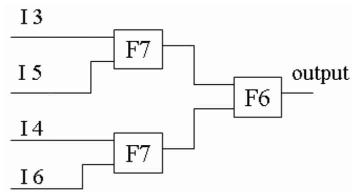


Fig. 6. An evolved circuit for processing salt & pepper noise

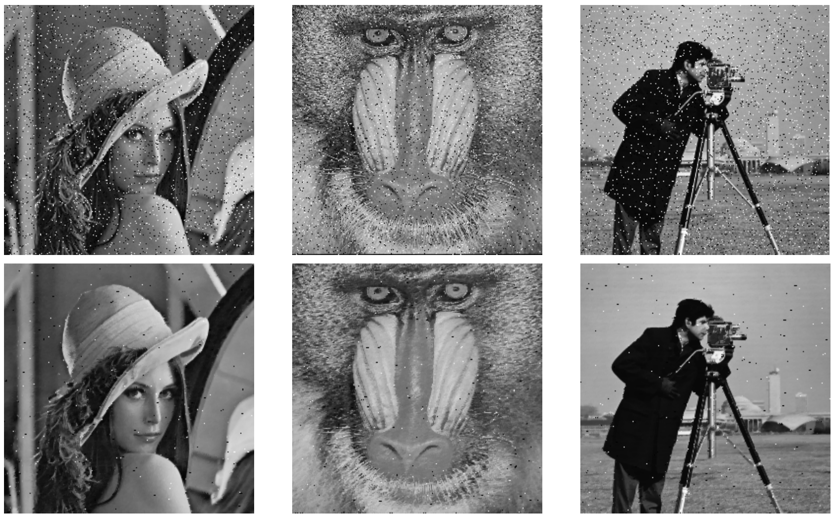


Fig. 7. The EHW filter trained by Lena with salt & pepper noise and tested on baboon and cameraman

Our hardware platform can run steadily under 30MHz and used only 70.5 seconds to finish 8192 generations of evolution. In reference [1], Zhang et al. have implemented a similar reconfigurable architecture in FPGA to evolve image operators which process Gaussian and salt & pepper noises. However, in their evolvable system, some evolutionary operations have been processed by host PC. Slow communication between the FPGA and the host PC becomes the bottleneck of speedup the evolution in their experiment. Considering the factors of the different population size and the generation number in their experiment, we still can announce we achieved a speedup of 20 times when compared to the report in [1]. Sekanina's group reported a very similar hardware implemented evolvable image operators in literature [6]. As shown in table 2 and 3, the image operators evolved here and in [6] present a very close performance on the quality of the processed images. A significant difference between Sekanina and our design is the hardware cost. Our design seems more compact: in the design of the evolvable image filter which employs the similar FEs array with the same size of 8×7 and includes the same number of individuals in EA unit, Sekanina used 10042 slices of Virtex II xc2v3000 FPGA and only 8596 slices in

Virtex xcv2000E FPGA was consumed in our design. According to our results reports, the mutation rate of 3.2% seems the most optimal parameter under the proposed experimental frame.

5 Conclusions

This paper has presented an approach for implementing a complete FPGA based intrinsic evolvable system in Celoxica RC1000 PCI board with a Xilinx Virtex xcv2000E FPGA. A prototype of the evolvable system was fully tested by evolving different spatial image filters to process two kinds of additional noise. This work demonstrates the generality and feasibility of the proposed hardware architecture. The powerful computation ability of the proposed evolvable system is presented in the experiments-an image operator which is expected for processing 256×256 gray scale image can be evolved in less than 71 seconds. Further work will be concentrated on developing the reported evolvable system for the automatic evolution of more complex image operators.

Acknowledgment

This work was supported by INHA University Research Grant.

References

1. Zhang, Y. et al.: Digital Circuit Design Using Intrinsic Evolvable Hardware. In: Proc. of the 2004 NASA/DoD Conference on the Evolvable Hardware, IEEE Computer Society (2004) 55-63
2. Glette, K., Torresen, J.: A Flexible On-Chip Evolution System Implemented on a Xilinx Virtex-II Pro Device. In: Proc. of the 6th Int. Conference on Evolvable Systems: From Biology to Hardware ICES 2005, LNCS 3637, Springer-Verlag (2005) 66-75
3. Hollingworth, G. et al.: The Intrinsic Evolution of Virtex Devices Through Internet Reconfigurable Logic. In: Proc. of the 3rd International Conference on Evolvable Systems: From Biology to Hardware ICES'00, LNCS 1801, Springer-Verlag (2000) 72-79
4. Tufte, G., Haddow, P.C.: Prototyping a GA Pipeline for Complete Hardware Evolution. In: Proc. of the first NASA/DoD Workshop on Evolvable Hardware, IEEE Computer Society (1999)18–25
5. Sekanina, L., Friedl, S.: On Routine Implementation of Virtual Evolvable Devices Using COMBO6. In: Proc. of the 2004 NASA/DoD Conference on Evolvable Hardware, IEEE Computer Society (2004) 63-70
6. Martínek, T., Sekanina, L.: An Evolvable Image Filter: Experimental Evaluation of a Complete Hardware Implementation in FPGA. In: Proc. of the 6th Int. Conference on Evolvable Systems: From Biology to Hardware ICES 2005, LNCS 3637, Springer-Verlag (2005) 76-85
7. Korenek, J., Sekanina, L.: Intrinsic Evolution of Sorting Networks: A Novel Complete Hardware Implementation for FPGAs. In: Proc. of the 6th Int. Conference on Evolvable Systems: From Biology to Hardware ICES 2005, LNCS 3637, Springer-Verlag (2005) 46-55
8. Celoxica Ltd. www.celoxica.com