An Evolvable Image Filter: Experimental Evaluation of a Complete Hardware Implementation in FPGA

Tomáš Martínek and Lukáš Sekanina

Faculty of Information Technology, Brno University of Technology,
Božetěchova 2, 612 66 Brno, Czech Republic
{martinto, sekanina}@fit.vutbr.cz

Abstract. This paper describes an evolvable image filter which is completely implemented in a field programmable gate array. The proposed system is able to evolve an image filter in a few seconds if corrupted and original images are supplied by user. The architecture is generic and can easily be modified to realize other evolvable systems. COMBO6 card with Xilinx Virtex xc2v3000 FPGA is used as a target platform.

1 Introduction

As image processing deals with large data sets, a hardware implementation of image processing algorithms becomes unavoidable in many applications to ensure reasonable processing time. Furthermore, efficient image processing algorithms require a certain level of intelligence to correctly interpret and present the input data. An adaptation is required in many cases. Hence image processing in general, and image filtering in particular, belong to most popular applications of evolvable hardware [4]. Evolvable hardware can be utilized either to find the required solution at the design time (i.e. to assist the designer during the design process) or to ensure (perhaps real-time) adaptation of hardware at the runtime. The both approaches have been reported in literature (see [1, 9, 13, 17]).

The objective of this paper is to explore the performance of an evolvable image filter that is *completely* implemented in a Field Programmable Gate Array (FPGA). In order to perform these investigations, a smoothing filter (whose function can be evolved) was implemented in FPGA. The main feature of the proposed implementation is that everything is implemented in a cutting-edge reconfigurable hardware platform available today. For the presented experiments we utilized the PCI COMBO6 card developed in the Liberouter project [7]. Therefore, our results should indicate what is possible to do with such the FPGA-based evolvable systems nowadays. Evolutionary algorithm, implemented in hardware, is used to find the filter which minimizes the difference between the corrupted image and training image. These images are stored in RAM memories available on COMBO6. A personal computer is used only for communication with the COMBO6 card, i.e. for writing/reading the images to/from RAMs etc. The performed experiments should allow us to exactly determine the adaptation time

J.M. Moreno, J. Madrenas, and J. Cosp (Eds.): ICES 2005, LNCS 3637, pp. 76–85, 2005.

and the quality of the evolved filters and, consequently, to specify the class of applications in which the filters could be evolved in real-time. We evaluated various variants of the evolvable filter, including the size of the virtual reconfigurable circuit and the parameters of the evolutionary algorithm. A crucial feature of the proposed architecture is that all EA operations as well as reconfiguration are completely overlapped by evaluation of candidate circuits, i.e. they are for free.

A lot of work has been done in the area of image filter evolution. Section 2 briefly introduces the field and emphasizes the differences between our approach and the existing approaches. In Section 3 the proposed complete hardware implementation is described. Section 4 summarizes the obtained results. Advantages and disadvantages of the evolvable system and potential applications are discussed in Section 5. Conclusions are given in Section 6.

2 Evolution of Image Filters in FPGA: A Survey

Various approaches have been proposed to the evolutionary design of image operators (filters). The authors of paper [5] evolved circuits for edge detection using elementary binary operations supported in FPGAs. Other edge detectors (also evolved in an FPGA) were represented as 2D arrays of integers that defined the convolution kernel [2]. Ebner evolved an edge detector using genetic programming which approximates the Canny edge detector [3]. An automatic feature identification algorithm that utilizes functional level operators like mean, standard deviation, convolution and linear scale was developed for multi-spectral images [9]. The evolutionary approach usually works in the time domain and produces non-linear filters. In many cases the evolved filters have exhibited better properties (the cost, quality of filtering) than conventional filters (such as median and mean filters, Sobel operators etc.) in tasks such as Gaussian or salt-and-pepper noise removal or edge detection [13, 11].

This work extends the approach initially developed by Sekanina [10, 11] who has evolved novel image filters (for 3×3 neighbourhood) using Cartesian genetic programming (CGP) applied at the functional level. Furthermore, the hardware implementation of CGP was proposed for FPGAs [13, 12]. However, in his approach image filters were evolved only by using a virtual reconfigurable circuit simulated in software that could eventually be implemented of the top of a conventional FPGA (no evolution in hardware was performed). Recently, Zhang et al. have implemented a very similar virtual reconfigurable circuit in FPGA and evolved some image operators in FPGA [17, 16]. Smith et al. used the same approach [15]. However, they have not evaluated the complete hardware implementation (including the evolutionary algorithm) in FPGA. Their papers do not indicate the time of evolution and speeding up against the software approach. It seems that some parts of evolutionary algorithm (such as circuit evaluation) have been carried out in software. As their system was proposed as asynchronous utilizing local handshaking protocols [16] they were not able to make the system completely pipelined. For similar purposes Sekanina and Friedl have proposed a completely pipelined implementation of an evolvable combinational unit for

FPGAs that can be considered as an evolvable IP core [14]. They obtained a significant speedup against the software approach. In this paper we will derive a complete hardware implementation of the evolvable filter according to the approach [14].

3 The Proposed Architecture

Architecture of the evolvable image filter is based on the component approach to evolvable hardware [13, 12, 11]. As Fig. 1 shows, it is composed of three main components—Fitness Unit, Genetic Unit and Virtual Reconfigurable Circuit.

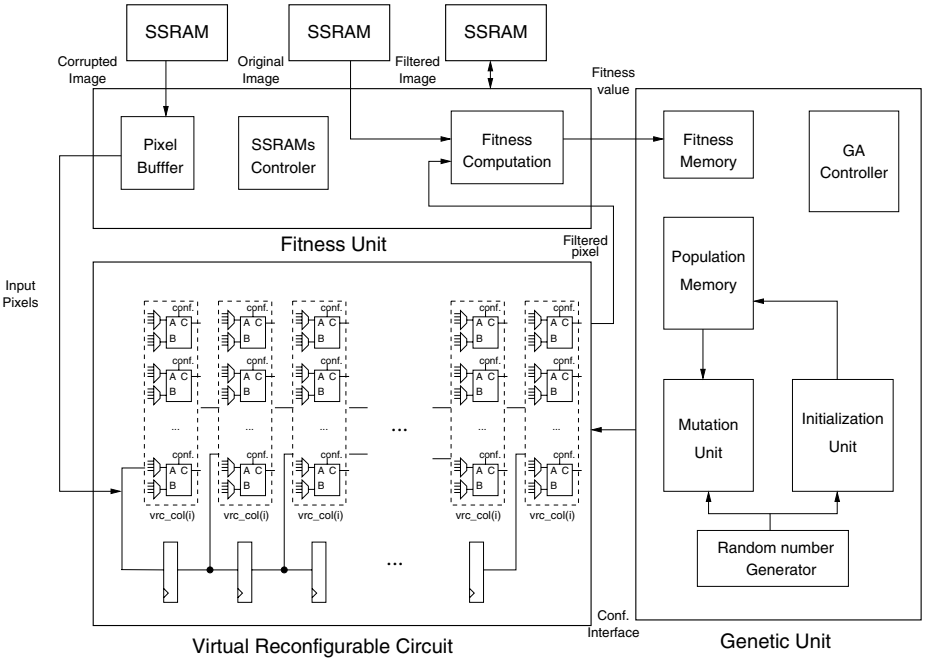

Fig. 1. Architecture of the evolvable image filter in FPGA

3.1 Virtual Reconfigurable Circuit

Every image operator will be considered as a digital circuit of nine 8bit inputs and a single 8bit output, which processes gray-scaled (8bits/pixel) images. As Fig. 2 shows every pixel value of the filtered image is calculated using a corresponding pixel and its eight neighbours in the processed image.

We approached the problem using Cartesian Genetic Programming (CGP) operating at the functional level. In contrast to the conventional CGP [8]—where gates and 1 bit connection wires are utilized—Configurable Functional Blocks

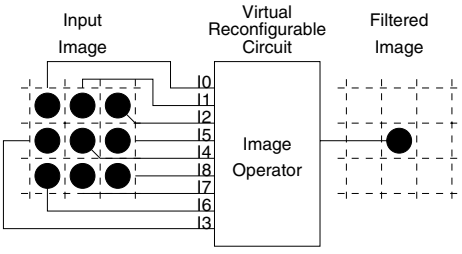

Fig. 2. Image Operator

(CFBs) and 8bit datapaths are employed [6]. Our model of the reconfigurable circuit consists of 2-input CFBs placed in a grid of n_c columns and n_r rows.

Any input of each CFB may be connected to the primary circuit inputs. Any input of each CFB may be connected to the output of a CFB, which is placed anywhere in the preceding column. The interconnection is implemented using multiplexers. Any CFB can be programmed to realize one of functions given in Table 1. These functions were recognized as useful for this task in [11]. Configuration bits of VRC are directly connected to the multiplexers that control the selection of CFB inputs and CFB functions. The reconfiguration is performed column by column. The computation is pipelined; a column represents a stage of the pipeline. Registers are inserted between columns in order to synchronize the input pixels with CFB outputs.

Table 1. Functions in CFBs

Number	Function	Description
0	$x \vee y$	binary or
1	$x \wedge y$	binary and
2	$x \oplus y$	binary xor
3	$x + y$	addition
4	$x + y^s$	addition with saturation
5	$(x + y) >> 1$	average
6	$Max(x, y)$	maximum
7	$Min(x, y)$	minimum

3.2 Fitness Unit

The design objective is to minimize the difference between the filtered image and the original image. Let u denote a corrupted image and let v denote a filtered image. The original (uncorrupted) version of u will be denoted as w. The image size is $K \times K$ (K=256) pixels but only the area of 254×254 pixels is considered because the pixel values at the borders are ignored and thus remain unfiltered. The fitness value of a candidate filter is obtained as follows: (1) the

VRC is configured using a candidate chromosome, (2) the circuit created is used to produce pixel values in the image v, and (3) the fitness value is calculated as

$$fitness = 255.(K-2)^2 - \sum_{i=1}^{K-2}\sum_{j=1}^{K-2} |v(i,j) - w(i,j)|.$$

Fitness computation is realized in Fitness Unit. The pixels of corrupted image u are loaded from external SSRAM memory and forwarded to inputs of VRC. Pixels of filtered image v are sent back to the Fitness Unit, where they are compared with the pixels of original image w. Filtered image is simultaneously stored into the additional SSRAM memory. Note that all image data are stored in external SSRAM memories due to the limited resources available in the FPGA chip.

3.3 Genetic Unit

Genetic algorithm is based only on the mutation operator (bit inversion); similarly to experiments reported in [13] a crossover is not taken into account. Population size is configurable. The new population is always generated from the best member of the previous one. Genetic algorithm operates in following steps: (1) Initialization Unit generates the first population at random (LFSR seeded from software is utilized). (2) Mutation Unit changes a given number of genes (bits) of a population member (this number is configurable) and the modified member is loaded into the VRC; it represents an image operators. (3) Genetic Unit is waiting for the evaluation performed by Fitness Unit and if the fitness value is better that the parent's fitness then the chromosome replaces its parent. (4) This is repeated until an appropriate number of generations are produced.

4 Experimental Results

4.1 Target Platform

COMBO6 developed in the Liberouter project is a PCI card primarily dedicated for a dual-stack (IPv4 and IPv6) router hardware accelerator. This board offers a very high computational power (FPGA Virtex XC2V3000 by Xilinx, Inc. with more than 3 mil. equivalent gates, up to 2GB DDR SDRAM, up to 9Mbit context addressable memory, and the three 2MB SSRAM memories) and so it is well suited for development and the use in various application domains, including evolvable hardware. We decided to use this card for our experiments because it offers us a sufficient performance and capacity of FPGA. Furthermore, the design software is available for free.

4.2 Synthesis for COMBO6

In order to compare different implementations we have decided to synthesize the whole system with VRC of size 8×4 CFBs and 8×7 CFBs. The evolutionary algorithm operates in the same way for both implementations; however, the size of

chromosome depends on the number of CFBs. The results of synthesis obtained using Leonardo Spectrum and Xilinx ISE tools are shown in the following table.

Table 2. Results of synthesis for Virtex II xc2v3000 FPGA

Resource	Avail	Used 8x4	Utilization	Used 8x7	Utilization
Function Generators	28672	10638	37.1%	18432	64.2%
Slices	14336	6175	43.0%	10042	70.0%
Dffs or Latches	30724	3172	10.3%	3668	11.9%
IOBs 256	684	236	34.0%	236	34.0%
Block RAMs	96	2	2.0%	3	3.0%

4.3 Time of Evolution

The evaluation of candidate filters consists of three basic activities: (1) preparation of a new candidate chromosome (filter), (2) reconfiguration of VRC circuit according to the prepared chromosome, and (3) evaluation of the filter. As most time is spent in filter evaluation, the architecture of evolvable image filter is designed in order to overlap the evaluation by other activities (1, 2). Therefore, because there is no overhead for reconfiguration of VRC (VRC is reconfigured at the beginning of filter evaluation) and the preparation of a new candidate circuit configuration is performed during the evaluation, it is possible to express the time of evaluation of a single filter as:

$$t_{eval} = (K-2)^2.\frac{1}{f} = (256-2)^2.\frac{1}{50.10^6} = 1.29ms$$

if the size of images is 256 x 256 pixels and the system operates at 50 MHz. Time of evolution can be expressed as follows:

$$t_e = t_{init} + g.n.t_{eval},$$

where g is the number of generations, n is population size and t_{init} is time needed to generate the initial population (t_{init} is negligible).

4.4 Functional Evaluation

The proposed evolvable image filter has been used to remove two types of noise– Gaussian noise ($\sigma = 16$) and Salt-and-Pepper noise (5% pixels with white or black shots)—that are popular in evolvable hardware literature [13, 11, 16, 15]. Original as well as filtered versions of Lena image were utilized in the fitness function. As the image is relatively large (256 x 256 pixels) we can assume that the evolved filter is general, i.e. the filter is able to remove the *same* type of noise

also from other images. Examples of filtered images and evolved filters are given in Table 5 and Figure 3.

We performed 100 runs for each problem and measured *mdpp* (mean difference per pixel) and checked the final generation. The average number of generations was calculated for the 100 runs. The evolution was stopped when no improvement in the best fitness value was detected over the last 5000 generations. Table 3 and 4 summarize the experiments. We performed the experiments with population size of four individuals and with the ratio of mutations 3 bits per chromosome; then we repeated all the experiments with mutation of 6 bits in chromosome. All the experiments were also performed for two different sizes of VRC: 8×4 and 8×7 CFBs.

Fig. 3. Evolved Salt and Pepper filter

Table 3. Results for Salt-and-Pepper noise

VRC size	Number of mutations	The best mdpp	In generation	Average mdpp	Average num. of gen.
8x4	3	0.64	4548	3.75	10475.00
8x4	6	0.51	36765	3.06	12189.00
8x7	3	0.77	8130	4.11	8617.00
8x7	6	0.48	22346	3.69	10340.00

Table 4. Results for Gaussian noise

VRC size	Number of mutations	The best MDPP	In generation	Average MDPP	Average num. of gen.
8x4	3	6.47	13767	7.98	10496.00
8x4	6	6.49	16058	7.77	10698.00
8x7	3	7.33	8435	10.10	7602.00
8x7	6	6.43	26647	8.27	8553.00

Table 5. The filter was evolved using Lena image and tested on other images

5 Discussion

We can compare the best-obtained results with the results reported in literature. In reference [11], the three best Salt-and-Pepper noise filters have *mdpp* 0.379, 0.507 and 0.656. The three best values of *mdpp* for Gaussian noise are 6.243, 6.312 and 6.326. Tables 3 and 4 show that the filters evolved here and in [11] exhibit a very similar quality. A small improvement visible in [11] is probably due the fact that some other properties (such as testability) were required for the filters in [11]; these properties were not considered in our hardware implementation. The influence of the mutation ratio and size of VRC is unclear from these experiments. Some other experiments will be arranged to clarify it.

From Tables 3 and 4 it can be derived that 9871 generations (i.e. 51 seconds at 50MHz) are needed in average to finish the evolution. The obtained time is

very reasonable if the proposed system should operate "instead" of a designer in the image filter design task. For some application, our solution could also operate as real-time evolving filter. However, if we consider that 100MHz operation frequency is easily reachable at COMBO6 and the training image could consist of 128 x 128 pixels only then the time of evolution is 6.3 second. Note that the speedup we obtained against the software approach (Pentium III/800MHz) is 50 if the FPGA operates at 100 MHz.

Our VHDL design benefits from a generic approach. All the implemented units are parameterized using various constants (such as the size of chromosome, the number of mutations, the size and topology of VRC, the size of input images etc.). Therefore, a novel FPGA-based implementation for some other evolvable systems can be obtained in a very short time. The FPGA communicates with PC via a special software allowing the designer to prepare scripts describing experiments that have to be performed. Typically, designer specifies the VRC, EA and fitness function, performs synthesis, uploads the evolvable system into FPGA and executes all experiments described in scripts. This approach can be considered as user-friendly interface to evolvable hardware.

6 Conclusions

A complete FPGA-based implementation of an evolvable image filter was realized and experimentally evaluated in an FPGA. The proposed system is able to evolve image filters in a few seconds. The architecture is generic and can easily be modified to realize other evolvable systems. Future research will be devoted to integrating the proposed solution to a real-world industrial application.

Acknowledgements

The research was performed with the financial support of the FRVS 2987/2005/-G1 project *The utilization of evolutionary algorithms for implementations of image filters in FPGAs*. Lukas Sekanina was supported from the research project of the Grant Agency of the Czech Republic under No. 102/03/P004 *Evolvable hardware based applications design methods*.

References

[1] Burian, A., Takala, J.: Evolved Gate Arrays for Image Restoration. Proc. of 2004 Congress on Evolutionary Computing, IEEE Publ. Press, 2004, p. 1185-1192

[2] Dumoulin, J. et al.: Special Purpose Image Convolution with Evolvable Hardware. In: Real-World Applications of Evolutionary Computing – Proc. of the 2nd Workshop on Evolutionary Computation in Image Analysis and Signal Processing. LNCS 1803, Springer, 2000, p. 1–11

[3] Ebner, M.: On the Evolution of Edge Detectors for Robot Vision Using Genetic Programming. In: SOAVE 97: Selbstorganisation von Adaptivem Verhalten, VDI Verlag 1997, p. 127–134

[4] Higuchi, T. et al.: Real-World Applications of Analog and Digital Evolvable Hardware. IEEE Trans. on Evolutionary Computation, Vol. 3, No. 3, 1999, p. 220–235
[5] Hollingworth, G., Tyrrell, A., Smith, S.: Simulation of Evolvable Hardware to Solve Low Level Image Processing Tasks. In: Proc. of the Evolutionary Image Analysis, Signal Processing and Telecommunications Workshop. LNCS 1596, Springer, 1999, p. 46–58
[6] Murakawa, M. et al.: Evolvable Hardware at Function Level. In: Proc. of the Parallel Problem Solving from Nature PPSN IV, LNCS 1141, Springer-Verlag Berlin, 1996, p. 62–72
[7] Liberouter project. www.liberouter.org
[8] Miller, J., Job, D., Vassilev, V.: Principles in the Evolutionary Design of Digital Circuits – Part I. Genetic Programming and Evolvable Machines, Vol. 1, No. 1, 2000, p. 8–35
[9] Porter, R.: Evolution on FPGAs for Feature Extraction. PhD thesis, Queensland University of Technology, Brisbane, Australia, 2001, p. 229
[10] Sekanina, L.: Image Filter Design with Evolvable Hardware. In: Applications of Evolutionary Computing – Proc. of the 4th Workshop on Evolutionary Computation in Image Analysis and Signal Processing EvoIASP'02, LNCS 2279, Springer-Verlag, Berlin, 2002, p. 255–266
[11] Sekanina, L., Ruzicka, R.: Easily Testable Image Operators: The Class of Circuits Where Evolution Beats Engineers. In: Proc. of the 2003 NASA/DoD Conference on Evolvable Hardware, USA, IEEE Computer Society Press, 2003, p. 135-144
[12] Sekanina, L.: Virtual Reconfigurable Circuits for Real-World Applications of Evolvable Hardware. In: Proc. of the 5th International Conference Evolvable Systems: From Biology to Hardware, ICES 2003, Trondheim, Norway, LNCS 2606, Springer-Verlag, 2003, p. 186-197
[13] Sekanina, L.: Evolvable components: From Theory to Hardware Implementations, Springer-Verlag, Natural Computing Series, 2003
[14] Sekanina, L., Friedl, S.: On Routine Implementation of Virtual Evolvable Devices Using COMBO6. In: Proc. of the 2004 NASA/DoD Conference on Evolvable Hardware, Seattle, USA, IEEE Computer Society Press, 2004, p. 63-70
[15] Smith, S., Leggett, S., Tyrrell, A.: An Implicit Context Representation for Evolving Image Processing Filters. In: Applications of Evolutionary Computing, LNCS 3449, Berlin, Springer Verlag, 2005, p. 407-416
[16] Zhang, Y., Smith, S., Tyrrell, A.: Intrinsic Evolvable Hardware in Digital Filter Design. In: Applications of Evolutionary Computing, Berlin, DE, Springer, LNCS 3005, 2004, p. 389-398
[17] Zhang, Y., Smith, S., Tyrrell, A.: Digital Circuit Design Using Intrinsic Evolvable Hardware. In Proc of the 2004 NASA/DoD Conference on Evolvable Hardware. Seattle, USA, IEEE CS Press, 2004, p. 55-62