EVOLUTION OF DIGITAL CIRCUITS OPERATING AS IMAGE FILTERS IN DYNAMICALLY CHANGING ENVIRONMENT

Lukáš Sekanina

Faculty of Information Technology, Brno University of Technology
Božetěchova 2, 612 66 Brno, Czech Republic
Tel: +42-05-41141215, Fax: +42-05-41141270, e-mail: sekanina@fit.vutbr.cz

Abstract. This paper deals with an automatic design of image filters at hardware level in dynamic environment. The environment is modeled using eight different fitness functions representing eight types of noise. Various properties of adaptation process have been measured when these fitness functions were applied for a constant time. The resulting filters operate in average much better than the conventional solution.

Keywords: evolutionary design, evolvable hardware, image operator, dynamic environment, adaptation, digital circuit.

1 Introduction

Evolutionary algorithms (EA) were successfully applied to design of digital circuits in the field of evolvable hardware in recent years. *Evolvable hardware* (EHW) may be considered as a technology, which enables to establish an evolvable system with the ability of hardware on-line adaptation to dynamically changing environments [1]. A circuit connection of the reconfigurable circuit (whose configuration bits are encoded in a chromosome) is autonomously synthesized by an EA. The evolution is free to explore many unconventional solutions beyond the scope of conventional engineering design and thus it should introduce a new quality to solution. Real-world applications of EHW are summarized in [2].

The fitness function reflects the circuit specification. In case of a *single* fitness function, the approach is usually called *evolutionary circuit design*. The best resulting circuit can be stored in a library and reused instead of conventional ones in future designs. Then the circuit quality, generality, robustness and an implementation cost are very important while the time needed for evolution is not crucial. The EA is only employed in a design phase and is not a part of the final system. It is possible to adjust the parameters of the EA to the problem.

When various fitness functions which represent changing environment (circuit specifications) are considered then the time needed to find a sufficient solution is critical. On the other hand, an implementation cost (e.g. the number of gates needed for implementation) can be of less importance since the reconfigurable logic where the evolution operates is always physically devoted for evolutionary purposes. Then it is irrelevant whether all gates or a single gate is employed in the final design. The EA is a part of a final system and it is responsible for adaptation. It is difficult to adjust the parameters of the EA to the problem because the fitness function is changed. Robot controllers belong to typical applications that require dynamic fitness function [3]. Theoretical aspects of the problem are summarized in [4].

This paper deals with an automatic design of image filters at hardware level in dynamic environment. The approach extends the previous paper [5], where it was shown that if an image is available both with and without noise, the whole process of the image filter design could be done automatically without influence of a designer. Furthermore, the paper [6] proved that an implementation cost of circuits evolved in FPGA (Field Programmable Gate Array) is quite competitive with conventional designs. The architecture initially proposed in [5] is not only employed for an evolutionary design, but it is mainly utilized for dynamic adaptation in this paper. The environment is modeled using eight different fitness functions representing eight types of noise. Various properties of adaptation process have been measured when these fitness functions were applied for a constant time. As a result, the best parameters of the EA (especially an approach to generating of the initial population) are reported.

The next section briefly summarizes conventional image operators. An evolutionary approach to image filter design is introduced in Section 3. The purpose of the section is to present the results we have obtained in case of a single fitness function. Experimental framework, results and discussion of the proposed method in dynamic environment are subject of Section 4. Conclusions are given in Section 5.

2 Conventional image filters

The paper deals with the gray-scale (8bits/pixel) images. Basic information on the theory of image operators is available e.g. in [7]. Images may be contaminated by a variety of noise sources. If a type of noise is known a priori, an efficient operator can be usually designed to suppress the noise. We are interested in a *Gaussian* and *shot* noise.

Gaussian noise is characterized by random changes in values of original pixels that are distributed according to the Gaussian curve. For our purposes, G12, G16, G32 and G40 will denote the images contaminated by Gaussian noise with a mean of zero and the standard deviation of 12, 16, 32 and 40, respectively.

Hardware implementation of a filter mostly operates in the spatial domain where the input image convolves with the filter function. In discrete convolution, the kernel (a small weight matrix) shifts over the image and multiplies its values with the corresponding pixel values of the image. Let us briefly describe *mean* (*FA1* in the paper), *mean-2* (*FA2*), *mean-*

4 (*FA4*) and *median* (*FME*) filters since these filters are usually used as the first attempts at a solution, and in addition, our results will be compared with them in the next sections. Consider the kernel 3 × 3 in this paper. The idea of mean filtering is simply to replace each pixel value in an image with the mean (average) value of its neighbors, including itself. Mean-2 and mean-4 filters take some pixels in account several times and should produce better results than mean filter for Gaussian noise since their coefficients are derived from Gaussian curve. The kernels are defined as:

$$FA1 = \frac{1}{9}\begin{pmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{pmatrix} \qquad FA2 = \frac{1}{10}\begin{pmatrix}1 & 1 & 1\\ 1 & 2 & 1\\ 1 & 1 & 1\end{pmatrix} \qquad FA4 = \frac{1}{16}\begin{pmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{pmatrix}$$

A pixel value is replaced with the median of neighboring values in the median filter. Such filter is much better at preserving sharp edges than the mean filter since it does not create the new (potentially unrealistic) pixel values.

In case of shot noise (also called "salt and pepper" noise, see Fig. 4A), some pixels are randomly set up to maximum (255) or minimum (0) value. For our purposes, P1, P3, P5 and P8 will denote the images contaminated by shot noise with 1%, 3%, 5% and 8% of corrupted pixels. Conventional robust solution, which suppresses shot noise, requires the median filter. However a cheaper solution, which does not remove all the "salt and pepper", needs only a simple *if-then-else* function (denoted as the *FIF* filter) checking an occurrence of 0 or 255 (see Fig. 4C).

3 Evolutionary design of image filters: A single fitness function

Evolutionary approach to image filtering

In case of evolutionary image operator design, either the kernel or the whole function (i.e. a circuit) is automatically evolved. The resulting structure evolves from primitives instead of calculating coefficients for a general-purpose model. Typical approaches to evolutionary image operator design published in past years include [8, 9]. However, these approaches deal with a single fitness function.

This paper tries to attack the problem of evolutionary design of various filters using a uniform approach at hardware level. The evolution is performed at simplified functional level, which requires only simple logical functions and adders as building blocks (see Table 1A). The method is based on a combination of *Cartesian Genetic Programming* (CGP) [10] and the evolution at *functional* level [11].

The reconfigurable circuit is modeled as an array of u (columns) × v (rows) programmable elements in CGP. The number of circuit inputs and outputs is fixed. Feedback is not allowed. An element's input can be connected to the output of some element in the previous columns or to some of circuit inputs. *L-back* parameter defines the level of connectivity and thus reduces/extends the search space. For example, if L = 1, only neighboring columns may be connected; if $L = u$, the full connectivity is enabled. For a given application, designer has to define: the number of inputs and outputs, L, u, v and a set of functions provided in the programmable elements.

Experimental framework

Similarly to [5], we have employed evolvable hardware for the design of image operators according to Fig. 1. The parameters of the system architecture are summarized in the following paragraphs.

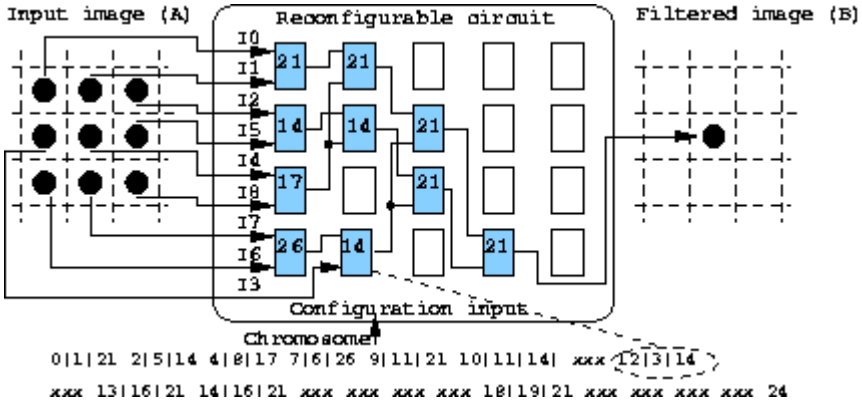

Fig. 1. An example of the reconfigurable circuit and its configuration for an image operator. Nine inputs (pixel values) are used to calculate a new pixel value. Parameters: 9 inputs, 1 output, circuit topology 5 × 4, *L-back* = 1. Only utilized elements are marked.

Reconfigurable circuit: Parameters of the reconfigurable circuit according to CGP are: 9 inputs (8bits), 1 output (8bits), u = 10 (columns), v = 4 (rows), *L-back* = 2. A programmable element has two inputs and operates (inputs as well as outputs) over 8 bits. Tab. 1A lists functions supported in the programmable element.

Chromosome encoding: A chromosome is a fixed-size string of integers, containing $u \times v$ genes (corresponding to the programmable elements in the reconfigurable circuit) and one place devoted to the index of the element representing the circuit output (see a chromosome in Fig. 1). A gene is described by three values: the position of the first input, the position of the second input and an index of the function applied on inputs. Thus genotype is of fixed length while phenotype is variable length since all the programmable elements need not be used.

Tab. 1. A list of functions implemented in a programmable element. The inputs X and Y and the outputs operate over 8bits. Symbols used: >> right shifter, << left shifter, ∧ binary AND, ∨ binary OR, ⊕ binary exclusive-OR, + 8bit adder, $+^s$ 8bit adder with saturation, ¬X is a binary negation of X. Constants are given in a hexadecimal system.

Set A – evolutionary design				Set B – dynamic environment			
0	X >> 1	16	¬(X ∧ Y)	0	X >> 1	16	X ⊕ ¬Y
1	X >> 2	17	(X∧0F) ∨ (Y∧F0)	1	X >> 2	17	¬(X ⊕ Y)
2	X >> 4	18	(X∧CC) ∨ (Y∧33)	2	¬X	18	(X∧0F) ∨ (Y∧F0)
3	¬X	19	(X∧AA) ∨ (Y∧55)	3	X << 1	19	(X∧CC) ∨ (Y∧33)
4	X << 1	20	X + Y	4	X << 2	20	(X∧AA) ∨ (Y∧55)
5	X << 2	21	(X + Y) >> 1	5	(X<<4) ∨ (X>>4)	21	X + Y
6	X << 4	22	X ∨ Y	6	0	22	$X +^S Y$
7	(X<<4) ∨ (X>>4)	23	X ∧ Y	7	FF	23	(X + Y) >> 1
8	0	24	X ∧ ¬Y	8	CC	24	(X + Y + 1) >> 1
9	33	25	¬X ∧ Y	9	X ∨ Y	25	Max (X, Y)
10	FF	26	X ⊕ Y	10	¬X ∨ Y	26	Min (X, Y)
11	X ⊕ ¬Y	27	¬(X ∨ Y)	11	¬(X ∨ Y)		
12	CC	28	¬(X ⊕ Y)	12	X ∧ Y		
13	Max (X, Y)	29	X ∨ ¬Y	13	X ∧ ¬Y		
14	(X+Y+1) >> 1	30	((X+Y) >> 1) + 1	14	¬(X ∧ Y)		
15	¬X ∨ Y			15	X ⊕ Y		

Population: Population size is 16. The circuits of initial population were created either randomly or using the function (X + Y) >> 1 only. This function (i.e. a simple average) is beneficial especially for design of Gaussian noise filters [5]. The evolution was typically stopped (1) when no improvement of the best fitness value occurs in the last 50000 generations, or (2) after 500000 generations.

Genetic operators: Mutation of two randomly selected functional elements is applied per circuit. Either function or a connection of one of the element's input is modified. The mutation always produces a correct circuit configuration. Crossover is not used. Four of the best individuals are utilized as parents and their mutated versions build the new population up (deterministic selection with elitism).

Fitness function: The design goal is to minimize the average difference per pixel between the noise and original images. Let A denote an input image (i.e. an image corrupted by a noise), let B denote an image after application of the image operator (i.e. a filtered image) and let C denote an ideal version of the image A. The image C must be available in the training phase. The image size is $N \times N$ ($N = 256$) pixels but only the area of 254×254 pixels is considered because the pixel values at the borders are ignored. The fitness value of a candidate chromosome is obtained as follows: (1) the circuit simulator is configured using a candidate chromosome, (2) so created circuit is used to calculate pixel values in the image B, and (3) the differences between pixels of the images C and B are added and the sum (denoted as *DIFF*) is subtracted from a maximum value (representing the worst possible difference: #grey_levels × #pixels):

$$FitnessValue = 255.(N-2)^2 - DIFF, \quad \text{where} \quad DIFF = \sum_{i=1}^{N-2}\sum_{j=1}^{N-2}\left|B(i,j) - C(i,j)\right|$$

Examples of evolved filters

More than one hundred image filters were evolved. An exhaustive report is available in [5]. Here, as an example, three interesting filters that were evolved (using Lena as a training image) are depicted in Fig. 2. They were also successfully applied on the test set to prove their generality. Note that RA3P5 was evolved in dynamic environment (see Section 4).

The *F24* filter (for G16 noise) appeared in the generation 185168. It consists of 21 functional elements (14 after manual optimization in which redundant functional elements were removed) and exhibits the lowest *DIFF* for the test set that we have ever reached (including the best conventional design *FA4*!). However, its implementation cost (2128 equivalent gates in XC4028XLA FPGA) is higher than for the *FA4* (1379 equivalent gates).

The *F57* (for P5 noise) filter appeared in the generation 63763, it consists of 8 functional elements (7 after manual optimization) and exhibits *DIFF* = 109875 for the Lena image. The cost of the *F57* is 441 equivalent gates and its quality is similar to the *FME*, which requires 4740 equivalent gates. Very cheap *FIF* filter costs 129 equivalent gates.

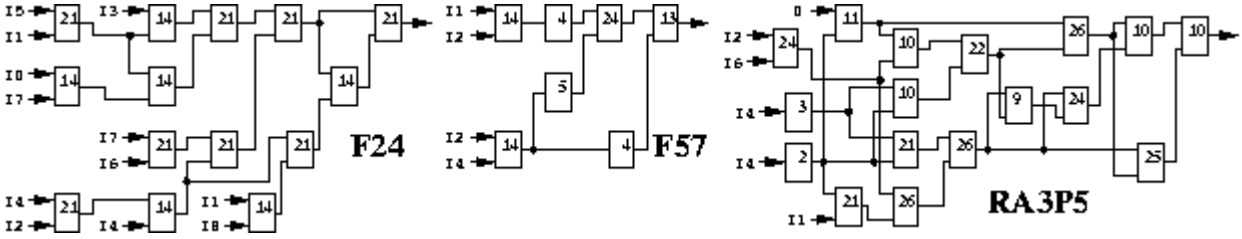

Fig. 2. The examples of filters evolved to suppress Gaussian noise (F24) and shot noise (F57, RA3P5). The functions of programmable elements are according to Tab. 1A (F24, F57) and Tab. 1B (RA3P5).

4 Dynamic environment

The EA is always implemented in the final system in case of dynamic environment. In this paper, the dynamic environment is defined by the sequence of eight types of noise: P5 → G16 → P1 → P3 → G12 → G32 → P8 → G40. The Gaussian and shot noise were selected because they represent different types of noise from viewpoint of the filter design. While elimination of Gaussian noise is based on averaging, the elimination of shot noise requires a procedure that is able to detect corrupted pixels. The ordering of the sequence should enable to obtain some information about the influence of the "neighboring noises" (i.e. to answer the questions like: Is it easier to evolve Gaussian filter in case that its predecessor was evolved for Gaussian or shot noise?)

The evolution is executed only for 20000 generations for a given noise. This number actually determines the frequency of change of the fitness function. The main goal of the experiment is to investigate: (1) overall quality of circuits produced in so strongly restricted time of evolution, (2) how the initial population influences the result, and (3) statistical relevance of the produced circuits.

Experimental framework

We have intuitively updated the table of functional elements (see Tab 1B) according to experiences from the previous runs (some functions were omitted because they were not utilized by evolution). Architecture of the reconfigurable circuit and parameters of the EA were set up according to Section 3 and left unchanged during experiments. Eight noised images (referred as *A* in Section 3) were prepared using the Lena image and the shot and Gaussian noise generators to obtain source data for eight fitness functions. Then the following runs (up to 20000 generations only), taking the fitness functions P5→ G16 → P1 → P3 → G12 → G32 → P8 → G40 sequentially, were repeated five times:
(a) A new initial population was generated randomly for every fitness function (RA – *Randomly Always* scheme).
(b) A new initial population was generated using only the functional element (X + Y) >> 1 for every fitness function (HA –*Heuristically Always* scheme).
(c) The first initial population was generated using only the functional element (X + Y) >> 1. Other initial populations were created as a copy of the last population in the previous run (HF – *Heuristically First* scheme).
(d) The first initial population was generated randomly. Other initial populations were created as a copy of the last population in the previous run (RF – *Randomly First* scheme).

Results

Tab. 2 shows the differences of the original and filtered images for conventional and all 160 evolved filters. Also the averages and the standard deviations are included for every set of runs. First, the conventional filters *FME*, *FA1*, *FA2*, *FA4* and *FIF* were employed to solve the problem. *FA4* is the best one (i.e. with the lowest *DIFF*) for a Gaussian noise. Although the *FIF* filter exhibits the lowest *DIFF* for shot noise, the visual effect of the filter is not good since some "salt and pepper" is unfiltered. Thus the *FME* filter is the best alternative in average. The following points summarize the experiments in dynamic environment.
(1) At least one evolved filter shows lower *DIFF* than the best conventional filter for every type of noise (33 filters of 160 typed in bold in Tab. 2, i.e. 20.6% in total). For instance, look at RA3P5 on Fig. 2 and Fig. 4E.
(2) HF-3 is the best sequence of filters evolved since it produces the lowest *DIFF* (291050) in average and 5 better filters than conventional ones. The filters in HF-3 consist of 21 functional elements in average and appeared in the generation 18023 in average. It shows that some improvements of the circuits will probably appear if more time is available for evolution. Note that the average generation where all the filters from Tab. 2 appeared is 12251.
(3) Success of the HF-3 is probably due to a good seed of the first initial population. If we look at the other runs of the HF scheme, large differences in the average *DIFF* are apparent (its standard deviation 33054 is high). HF schemes also produced only 9 filters better than conventional ones. The HA scheme can be considered as the best approach how to generate initial populations. It shows 19 (i.e. 47.5% of all in the scheme) better circuits than conventional ones and the lowest average difference (317733) as well as standard deviation (11123) at all. It produces excellent Gaussian filters (90% of them are superior to conventional ones). Some shot noise filters are very close to the best of conventional ones.
(4) Initial population should not be generated randomly, especially for Gaussian noise. As we learned in [5], the function (X + Y) >> 1 is crucial for Gaussian filters. Shot noise filters are evolved easier when the initial population is taken as the last population of the previous run (5 filters superior to conventional ones in the HF scheme). Efficiency of the HA and HF against the RA and RF schemes is also demonstrated by the average number of the generations (about 14049 against 10453) where the fittest circuits occurred and by the average number of functional elements (16 against 6) needed for design. It means that evolution is able to exploit hardware resources and to improve results continuously.
(5) While the HA scheme is considered as the best one, then the ordering of the sequence of fitness functions is not important since a new population does not depend on a previous run. It seems that the population size is too small to keep some useful genetic material from the previous runs in HF or RF scheme.
(6) HA-3 is the worst one of HA runs. It is mainly due to problems with P3 noise, other filters are close to the best ones.
(7) Fig. 3 shows different characteristics of evolution of the average run in HF and HA scheme. Both averages are better in comparison to the median filter. It is also possible to observe decreasing quality of *DIFF* if less than 2000 generations are produced in a run.
(8) The functions in Tab 1B were selected suitably since it was much easier to obtain a good circuit than in case of functions listed in Tab 1A.

Tab. 2. The list of differences (*DIFF*) for conventional and evolved filters operating in eight environments. The best conventional designs are typed in bold in the part "Conventional filters". Every scheme (RA, HA, HF and RF) was run 5 times and the averages ("Average" rows) as well as standard deviations ("Stddev" rows) are given bellow. The designs that outperform the best conventional ones are typed in bold in the part "Evolved filters". The best evolved design for a given environment is marked in a gray box. The "avr/std" denotes an average and standard deviation of a scheme.

Conventional filters									
Filter/noise	**P5**	**G16**	**P1**	**P3**	**G12**	**G32**	**P8**	**G40**	**avr/std**
no filter	501229	829514	158358	329235	626434	1647786	732891	2027948	856674
FME	190629	461765	173887	182178	382512	769791	204488	918634	410486
FA1	613602	429354	390196	505975	376207	**651435**	744078	**759620**	558808
FA2	596667	417411	366487	485742	360447	653977	731850	769055	547705
FA4	583513	**415273**	341849	466804	**351339**	678807	728003	806999	546573
FIF	**50470**	828716	**10085**	**25861**	626389	1620186	**95362**	1971301	653546
Evolved filters									
RA-1	234021	452793	111078	81547	**349783**	671050	167597	843089	363870
RA-2	234021	437000	46999	215164	409330	656444	**68751**	854425	365267
RA-3	**42485**	495891	44059	53299	364896	1191178	156830	769001	389705
RA-4	501229	423026	79762	**25844**	359536	702871	296670	969996	419867
RA-5	114319	495891	111078	54157	374886	865341	296670	897686	401254
Average	225215	460920	78595	86002	371686	817377	197304	866839	**387992**
Stddev	174662	33616	32797	74842	22917	224972	98481	73965	**23943**
HA-1	74770	**410254**	83667	61333	**346283**	**640656**	**84957**	**756532**	307307
HA-2	118520	**414018**	111078	58230	**349645**	**646299**	138892	763634	325040
HA-3	71842	**410469**	67665	207307	**348976**	**648835**	116978	759972	329006
HA-4	133592	**412512**	12000	89487	**350418**	**646009**	181657	**757866**	322943
HA-5	60647	**409182**	12001	70259	**346637**	**640674**	147175	**748390**	304371
Average	91874	411287	57282	97323	348392	644495	133932	757279	**317733**
Stddev	32091	1945	44155	62678	1840	3665	35929	5647	**11123**
HF-1	147934	453991	111078	204002	367264	698531	206534	845416	379344
HF-2	84532	**412528**	**7907**	**19947**	359793	696744	115629	813514	313824
HF-3	**48152**	**411107**	10974	**20131**	364026	**644608**	**64869**	764536	291050
HF-4	102038	443729	56898	70259	353546	742690	137112	762348	333578
HF-5	90254	433956	21309	42063	373376	667773	143731	**755340**	315975
Average	94582	431062	41633	71280	363601	690069	133575	788231	**326754**
Stddev	35977	18949	43434	77008	7496	36908	51189	39442	**33054**
RF-1	**46854**	428328	111078	215164	367527	654811	161399	786584	346468
RF-2	234021	495859	111078	89487	409330	800735	201011	909636	406395
RF-3	107051	438908	39607	65291	365140	702795	148476	833656	337616
RF-4	235270	442804	17001	72834	369896	741742	147183	849225	359494
RF-5	236636	495891	111078	204002	409470	757040	184085	843144	405168
Average	171966	460358	77968	129356	384273	731425	168431	844449	**371028**
Stddev	89313	32852	46036	73864	23000	55353	23482	43985	**32669**

Discussion

All the results in the Tab. 2 are based on the Lena image. Natural question is whether the evolved filters are general. We have learned in [5] that if *DIFF* for Gaussian noise (and the Lena image 256 × 256 pixels) is lower than for the *FA4* filter, than the evolved filter is general enough. In case of shot noise, we know that *DIFF* should be lower that for the *FME* filter, but not necessary lower than for the *FIF* filter. For instance, the *DIFF* of the *F57* filter is about two times worse than for the *FIF* filter, but the *F57* operates slightly better than *FIF*. We have tested the filters *RA3P5* (see Fig. 2) and *HA5G16* filters on the test set and they really operate generally.

The problem lies in a uniform metric (i.e. *DIFF*) that has to be applied in the fitness calculation because the type of noise is unknown a priori in open environment. From this viewpoint, the approach of the paper is unrealistic since we know the type of noise in all cases. On the other hand, the approach can be easily compared to a conventional solution.

Experiments are very time consuming since images of 256 × 256 pixels have to be considered to ensure generality of evolved filters. It takes one day of evolution (Pentium III/800MHz) to get the values for a single line of Tab. 2 (i.e. for eight runs). The number of pixels filtered during a single run is 254 × 254 (image size) × 16 (population size) × 20000 (generations) = 2.1×10^{10} pixels. If a reconfigurable circuit is able to operate at 50MHz (and it is possible, see [12]), than the evolution will be finished in 6.9 minutes. Therefore, hardware implementation will be very beneficial for future experiments.

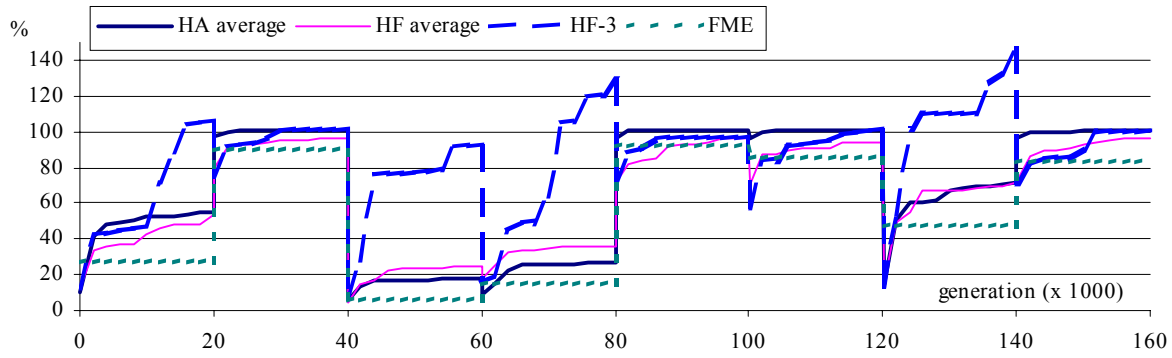

Fig. 3. Waveforms of the fitness values in eight environments (8 x 20000 generations). *DIFF* is related to the best conventional filter for a given environment (i.e. 100.best_conventional/*DIFF* [%]). The best run (HF-3) and averages of the HF and HA schemes are compared to a solution that employs the median filter only (FME is also related to the best conventional filters).

A) P5 noise B) FME C) FIF D) F57 E) RA3P5

Fig. 4. A part (120 × 120 pixels) of the Lena image with the P5 noise (A) and after application of various filters (B, C, D, E).

5 Conclusions

We claim that efficient image filters can be also evolved in the dynamic environment. In other words, if the reconfigurable circuit and the EA are used according to the proposed method, various circuits operating as image filters can be successfully evolved in reasonable number of generations. Furthermore, we have also evolved another operators, e.g. excellent edge detectors. However, as we learned from No Free Lunch theorems, the approach can not be any panacea solving a design problem of every digital circuit of nine (8 bit) inputs and a single output (8 bit).

Acknowledgments

The research was performed with the Grant Agency of the Czech Republic under No. 102/01/1531 *Formal approach in digital circuit diagnostic – testable design verification.*

References

[1] Sanchez, E., Tomassini, M. (Eds.): Towards Evolvable Hardware: The Evolutionary Engineering Approach. LNCS 1062, Springer-Verlag, Berlin 1996.

[2] Torresen, J.: Possibilities and Limitations of Applying Evolvable Hardware to Real-World Applications. In: Proc. of the Field Programmable Logic and Applications FPL2000, LNCS 1896, Springer-Verlag, Berlin 2000, 230–239.

[3] Islam M., Terao, S., Murase, K.: Effect of Fitness for the Evolution of Autonomous Robots in an Open-Environment. In: Proc. of Evolvable Systems: From Biology to Hardware ICES2001, LNCS 2210, Springer-Verlag, Berlin 2001, 171–181.

[4] Branke, J.: Evolutionary Approaches to Dynamic Optimization Problems – Update Survey. In: Proc. of the GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, 2001, 27–30.

[5] Sekanina, L.: Image Filter Design with Evolvable Hardware. To appear In: Proc. of the 4th Evolutionary Image Analysis and Signal Processing Workshop EvoIASP'2002, LNCS 2279, Springer-Verlag 2002, pp. 12 .

[6] Sekanina, L., Drábek, V.: Automatic Design of Image Operators Using Evolvable Hardware. To appear In: Proc of the 5th IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop DDECS 2002 Brno, Czech Rep., 2002, pp. 8.

[7] Russ, J., C.: The Image Processing Handbook (third edition). CRC Press LLC 1999.

[8] 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 EvoIASP'99, Springer, 1999, pp. 46–58.

[9] Dumoulin, J. et al.: Special Purpose Image Convolution with Evolvable Hardware. In: Proc. of the EvoIASP 2000 Workshop, Real-World Applications of Evolutionary Computing, LNCS 1803, Springer-Verlag, Berlin 2000, pp. 1–11.

[10] Miller, J., Thomson, P.: Cartesian Genetic Programming. In: Proc. of the Genetic Programming European Conference EuroGP 2000, LNCS 1802, Springer-Verlag, Berlin 2000, pp. 121–132.

[11] 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, pp. 62–72.

[12] Sekanina, L., Růžička, R.: Design of the Special Fast Reconfigurable Chip Using Common FPGA. In: Proc. of the Design and Diagnostic of Electronic Circuits and Systems IEEE DDECS'2000, Polygrafia SAF Bratislava, Slovakia 2000, pp. 161–168.