Image Filter Design with Evolvable Hardware

Lukáš Sekanina

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

Abstract. The paper introduces a new approach to automatic design of image filters for a given type of noise. The approach employs evolvable hardware at simplified functional level and produces circuits that outperform conventional designs. If an image is available both with and without noise, the whole process of filter design can be done automatically, without influence of a designer.

1 Introduction

Image recognition is a problem that has to be solved successfully in various industrial applications, namely in automatic traffic sign recognition, car registration number recognition or in the automatic control of the producing line in a factory where correct and damaged products have to be detected.

The whole recognition system has to be extremely accurate. For example, only one wrong decision of the million is acceptable in case of recognition of the correct/damaged component on the production line. The quality of recognition algorithm strongly depends on quality of the images coming from a camera since these algorithms are commonly designed for idealized images.

Images usually acquired through modern cameras may be contaminated by a variety of noise sources (e.g. photon or on-chip electronic noise) and also by distortions such as shading or improper illumination. Therefore, a preprocessing unit (image filter) has to be incorporated before recognition to improve image quality.

This paper deals with filters for smoothing images. Industry calls for automatic design of such filters since (1) the system should adapt to changing environment autonomously (e.g. to the changes of illumination or after replacement of a damaged camera) and (2) it is expensive to pay designers when no standard solution can be easily adopted.

A new approach to automatic design of image filters for a given type of noise is introduced. The approach employs *evolvable hardware* at functional level and produces circuits that outperform conventional designs in terms of the resulting image quality and implementation cost in most cases. We do not know any work that is related to evolutionary design of image filters at hardware level at the moment. Available evolutionary designs reported in past years are oriented towards filters for one-dimensional signals or to specific image operators. Our solution is based on simple functions (binary operations or 8bit adders) that may be effectively implemented in low-cost, commercial off-the-shelf hardware devices like FPGA (Field Programmable Gate Array).

The next section briefly summarizes conventional image filters while the basic principles of evolvable hardware are described in Section 3. Some of the already published evolutionary approaches to filter and image operator design are mentioned in Section 4. Section 5 introduces experimental framework of our approach. Section 6 reports evolved designs that are discussed in Section 7. And finally, conclusions and problems for future work are given in Section 8.

2 Conventional Design of Image Filters

The conventional approach to image filter design is explained in many textbooks, e.g. in [1,2]. An image can be filtered either in the frequency or in the spatial domain. We are interested in the spatial domain where the input image x convolves with the filter function h. In discrete convolution, the kernel is shifted over the image and multiplies its values with the corresponding pixel values of the image. A kernel is a small matrix of numbers whose members define weights of accounted pixels. Let $y(i,\ j)$ denotes a pixel value of the resulting image at position $(i,\ j)$. For a square kernel of size $M \times M$, we can calculate the output image with the following formula:

$$y(i,j) = \sum_{m=-\frac{M}{2}}^{\frac{M}{2}} \sum_{n=-\frac{M}{2}}^{\frac{M}{2}} h(m,n)x(i-m,j-n)$$

Various standard kernels exist for specific noise, where the size and the form of the kernel determine the characteristics of the operation. The filter can be applied on an already filtered image repeatedly. In contrast to the frequency domain, it is possible to implement non-linear filters in the spatial domain. In this case, the summations in the convolution function are replaced with some kind of non-linear operator (e.g. Median or Kuwahara filter). Another advanced filters like non-linear mean filter or averaging using a rotating mask are given in [1,2].

Let us briefly describe mean (denoted as $FA1$ in the paper), mean-2 ($FA2$), mean-4 ($FA4$) and median (FME) filters since reported results will be compared with them in the next sections. Consider M=3 for the 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 produce better results than mean filter for Gaussian noise since their coefficients are derived from the curve of Gaussian distribution. The kernels are defined as:

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

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

3 Evolvable Hardware

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 [3]. A circuit connection of the fast reconfigurable circuit (whose configuration bits are encoded in a chromosome) is autonomously synthesized by an evolutionary algorithm. In the case of a single fitness function, the approach is usually called evolutionary circuit design. Evolution is free to explore many unconventional solutions beyond the scope of conventional engineering design and thus should introduce a new quality to solution. Real-world applications of EHW are summarized in [4].

Miller and Thomson have introduced *Cartesian Genetic Programming* (CGP) [5] that was recently applied by several researchers especially for evolutionary design of combinational circuits [6,14]. Reconfigurable circuit is modeled as an array of u (columns) $\times$ v (rows) programmable elements (gates). The number of circuit inputs and outputs is fixed. Feedback is not allowed. A gate input can be connected to the output of some gate 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 performed by programmable elements (typically binary operations over two or three inputs). In other words, these parameters define a configuration of the programmable circuit (see Figure 1).

The idea of EHW at *functional level*, where the programmable elements include functions like adders, multipliers, dividers, sine or cosine generators over floating point numbers, was initially introduced in [7].

4 Evolutionary Filter and Image Operator Design

In the case of the spatial domain, image filters and image operators are designed similarly. The designer usually determines M and the values of the kernel. This is a very time consuming job, especially when the noise type is unknown. Thus evolutionary design of either the kernel or the whole function (i.e. a circuit at hardware level) offers an alternative approach. The resulting structure evolves from primitives instead of calculating coefficients for a general-purpose model. Evolved solutions (circuits) should be more efficient than conventional design in terms of performance and implementation cost.

Authors in [8] evolved circuits for edge detection using elementary binary operations supported in FPGAs while another edge detectors (also evolved in

FPGA) were represented as 2D arrays of integers that defined the convolution kernel [9]. Evolutionary optimization of soft morphological filters for archive film restoration was extended to temporal domain in [10].

At least one paper at every conference on EHW was devoted to filter design in history: Evolvable System: From biology to hardware conference ICES96 (1 paper), ICES98 (1), ICES00 (1), ICES01 (1); NASA/DoD Workshops on Evolvable hardware 1999 (3), EH00 (1), EH01 (3). Proposed approaches however deal with one-dimensional signals only. Miller used pure gate array and CGP to filter simple signals [11]. In [12] the authors implemented a simple filter as well as whole evolutionary algorithm in the FPGA. Genetic programming approach to analog filter design is explained in detail in [13]. In [14] the authors used CGP for the design of finite impulse response digital filters with reduced power consumption. Resulting design is automatically transformed to VHDL and synthesized. The filter evolves from primitives like adder, subtractor or shifters.

5 Image Filter Evolution: Experimental Framework

The goal is to evolve a digital circuit operating as an image filter for a given type of noise. Gray-scale (8bits/pixel) images of size $N \times N$ (N=256) pixels are considered in the paper. The pixel value is filtered using 3×3 neighborhood. The new pixel value is available on the 8bit output of the circuit. The circuit input consists of nine pixel values. Based on initial experiments, the following parameters were set up as default:

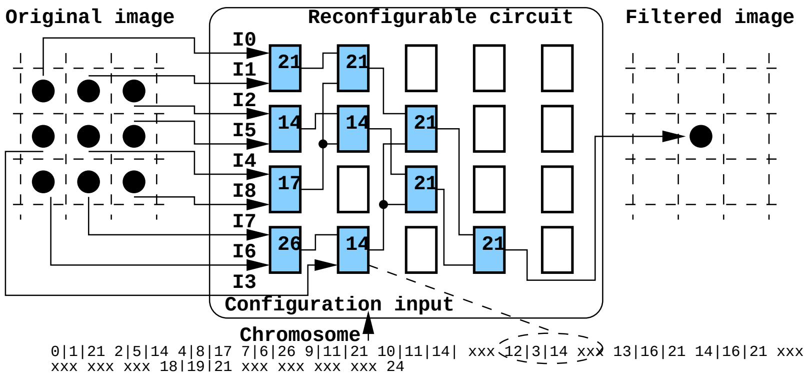
Original image
Reconfigurable circuit
Filtered image
I0
I1
I2
I5
I4
I8
I7
I6
I3
21
21
14
14
21
17
21
26
14
21
Configuration input
Chromosome
0|1|21 2|5|14 4|8|17 7|6|26 9|11|21 10|11|14| xxx 12|3|14 xxx 13|16|21 14|16|21 xxx
xxx xxx xxx 18|19|21 xxx xxx xxx xxx 24

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

5.1 Reconfigurable Circuit

Parameters of the reconfigurable circuit according to CGP are: 9 inputs (8bits), 1 output (8bits), $u = 10$ (columns), $v = 4$ (row), L-back = 2. A programmable element has two inputs and operates over 8 bits. Table 1 lists functions supported in the programmable element. Circuit inputs correspond with the pixels of the kernel according to the Figure 1. The proposed architecture operates rather like parallel gate level evolution than functional level evolution. In the case of adders (functions 14, 20, 21 and 30), only lower 8bits are considered as output. Except the adders, the elements have trivial hardware implementation.

Table 1. A list of functions implemented in a programmable element. The inputs a and b and the outputs operate over 8bits. Symbols used: $>>$ right shifter, $<<$ left shifter, $\wedge$ binary AND, $\vee$ binary OR, $\oplus$ binary exclusive-OR, $+$ 8bit adder, $\bar{a}$ is a binary negation of a. Constants are given in a hexadecimal system.

0	$a >> 1$	1	$a >> 2$	2	$a >> 4$
3	$\bar{a}$	4	$a << 1$	5	$a << 2$
6	$a << 4$	7	$(a << 4) \vee (a >> 4)$	8	0
10	FF	11	AA	12	55
13	33	14	$(a + b + 1) >> 1$	15	$\bar{a} \vee b$
16	$\overline{a \wedge b}$	17	$(a \wedge 0F) \vee (b \wedge F0)$	18	$(a \wedge CC) \vee (b \wedge 33)$
19	$(a \wedge AA) \vee (b \wedge 55)$	20	$a + b$	21	$(a + b) >> 1$
22	$a \vee b$	23	$a \wedge b$	24	$a \wedge \bar{b}$
25	$\bar{a} \wedge b$	26	$a \oplus b$	27	$\overline{a \vee b}$
28	$\overline{a \oplus b}$	29	$a \vee \bar{b}$	30	$((a + b) >> 1) + 1$

5.2 An Evolutionary Algorithm

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 Figure 1). A gene is described by three values: the position of the first input, the position of the second input and a number 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.
Population: Population size is 16. Initial population is generated randomly, but only the function 21 was used in some runs (see Section 7). 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 gates is applied per circuit. A 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 up the new population (deterministic selection with elitism).

Fitness function: Various approaches exist to measure image visual quality. The *signal-to-noise ratio* or the pure *average difference per pixel* (*dpp*) are the commonest ones. We chose the second approach. Let *orig* denote an original image without any noise (e.g. Lena), *noise* denotes the original image corrupted with noise of type Xxx (e.g. LenaXxx), and $filtered$ denotes an image filtered using some filter Fyy (e.g. lenaXxxFyy). The filter is trained using Lena256 and Lena256Xxx images for Xxx noise. Only the area of 254 x 254 pixels is filtered because the pixel values at the borders are ignored. To obtain the fitness value, the differences between pixels of the filtered and original image are added and the sum is subtracted from a maximum value (representing the worst possible difference: #grey_levels × #pixels):

$$FitnessValue = 255.(N-2)^2 - \sum_{i=1}^{N-2} \sum_{j=1}^{N-2} |orig(i,j) - filtered(i,j)|$$

6 Results

For our experiments, we consider three types of noise denoted as: $G16$ (Gaussian with a mean of zero and a standard deviation of 16), $R32$ (uniform random with parameter 32) and $N1$ (block uniform random). Images with $G16$ and $R32$ noise were generated from originals using Adobe Photoshop program. We designed the $N1$ noise for testing purposes to model random defects in the image. $N1$ noise is generated as $R32$ noise but applied only for randomly selected blocks of the image. The rest of the image preserves.

We have evolved more than one hundred image filters and 20 of them are presented. The test set contains the following images: Lena (popular for testing), Man (a man), Bld (a building), Cpt (a capacitor), and Rel (a relay). Cpt and Rel images were acquired through a camera and the system for automatic recognition of damaged/correct product on the production line. The best designs as well as results of traditional filters (typed as bold) are sorted according to their ability to remove general noise in Table 2. The ranks for a given type of noise are presented in the last three columns. As an example, the complete results for $G16$ noise are given in Table 3.

It was detected after analysis of the evolved designs that some filters do not employ all the functional elements effectively. For instance, the filter F20 uses two functional elements with the same inputs (marked elements in the Figure 3). These functional elements can be omitted (i.e. replaced by a direct connection) since they do not influence the output of the filter. The number of functions used in the evolved designs after manual optimization is listed in the column #O of the Table 2.

Table 2. A list of evolved and conventional filters sorted according to their ability to filter general noise. Columns have these purposes: TNF – trained for noise (or tested for noise for conventional filters); *dpp* – *dpp* for trained image; $u \times v$ – circuit topology; L – L-back parameter; $\#E$ – the number of functional elements used in the evolved design; $functions\,used$ in the evolved designs; $\#O$ – the number of functional elements used after manual optimization; *gener* – the generation where the solution occured; $G16, R32, N1$ – the final rank for a given noise type and all the test images.

Filter	TFN	dpp	u × v	L	#E	functions used	#O	gener	G16	R32	N1
F24	G16	6.362	10x4	2	21	8, 17, 22, 21(12), 14(6)	14	185168	1	2	3
F20	G16	6.358	10x4	2	17	17, 21(14), 30(2)	15	79369	2	3	7
F26	G16	6.358	10x4	2	19	17, 18(2), 21(12), 14(4)	14	24853	4	4	8
F25	G16	6.356	10x4	2	24	21(16), 14(8)	22	13415	6	6	5
F21	G16	6.354	20x2	4	19	21(17), 30(2)	18	133224	7	1	12
F14	G16	6.401	20x2	4	14	17, 18, 21(9), 23, 30(2)	12	13678	3	16	4
F24A	G16	6.388	10x4	2	14	21(14)	14		5	12	6
F11	G16	6.354	10x4	2	26	21(19), 22, 30(6)	24	81532	8	5	10
FA4	G16	6.437							9	14	2
F23	N1	6.060	40x1	40	10	18, 21(9)	9	42772	13	18	1
F15	G16	6.363	40x1	40	18	21(18)	18	51356	10	11	13
F21A	G16	6.384	20x2	4	19	21(18)	18		11	8	17
F13	G16	6.360	20x2	4	22	21(16), 22, 30(5)	21	43749	14	7	15
F16	G16	6.367	40x1	40	16	21(16)	16	71518	12	9	16
F6	G16	6.400	50x1	50	17	17,21(10),22,24,29,30(3)	16	24693	15	17	11
F18	N1	6.077	40x1	40	9	21(9)	9	141744	16	19	9
F27	R32	6.926	10x4	2	25	18, 21(10), 14(14)	23	128025	18	10	19
F17	R32	6.977	40x1	40	14	2, 20, 21(12)	14	43549	17	13	18
F19	R32	6.981	40x1	40	16	10, 17(3), 21(12)	16	38007	19	15	20
F8	G16	6.640	50x1	50	12	7, 17, 21(9), 30	12	70304	20	21	14
FA2	G16	6.469							21	20	22
F22	N1	6.283	40x1	40	7	21(7)	7	35872	22	23	21
FA1	G16	6.655							23	22	24
FME	G16	7.157							24	24	23

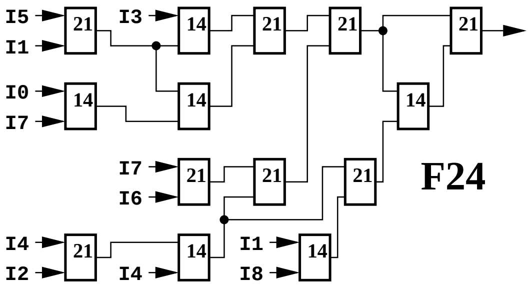

Fig. 2. The best filter evolved for the G16 noise. F24 (with topology 10×4, L-back=2) employs after optimization only functions 21 and 14.

Table 3. Columns 2-6 report *dpp* for a given filter and image with $G16$ noise. $Total-dpp$ is an average of *dpp* for all the test images. The last column shows the standard deviation calculated using the best known *dpp* value (typed as bold) for a given image.

Filter	LenaG16	CptG16	RelG16	ManG16	BldG16	Total-dpp	Std. deviation
no filter	12.857	12.754	11.994	12.860	12.526	12.598	6.081
F24	6.362	6.046	4.930	8.243	7.763	6.669	0.082
F20	6.358	6.038	4.879	8.324	7.919	6.704	0.141
F14	6.401	6.109	4.998	8.256	7.762	6.705	0.115
F26	6.358	6.045	4.879	8.327	7.921	6.706	0.143
F24A	6.388	6.125	4.951	8.302	7.771	6.707	0.111
F25	6.356	6.065	4.913	8.310	7.893	6.707	0.135
F21	**6.354**	6.039	4.848	8.351	7.950	6.708	0.156
F11	6.354	6.058	4.877	8.356	7.914	6.712	0.146
FA4	6.437	6.094	4.981	**8.196**	7.875	6.717	0.140
F15	6.363	6.090	4.889	8.357	7.885	6.717	0.140
F21A	6.384	6.103	4.872	8.408	7.833	6.720	0.137
F16	6.367	6.084	4.876	8.372	7.915	6.723	0.152
F23	6.446	6.166	5.050	8.243	7.763	6.734	0.146
F13	6.360	6.073	4.879	8.376	7.985	6.735	0.179
F6	6.400	6.148	4.983	8.260	7.937	6.746	0.169
F18	6.438	6.171	5.023	8.277	7.958	6.773	0.192
F17	6.410	6.197	**4.804**	8.460	8.021	6.778	0.223
F27	6.411	**6.023**	4.882	8.475	8.143	6.787	0.261
F19	6.388	6.052	4.859	8.461	8.215	6.795	0.285
F8	6.640	6.328	5.158	8.511	**7.640**	6.855	0.283
FA2	6.469	6.105	4.856	8.516	8.414	6.872	0.381
F22	6.711	6.477	5.396	8.556	7.748	6.978	0.406
FA1	6.655	6.270	4.858	9.054	9.022	7.172	0.748
FME	7.157	6.837	5.820	9.160	8.042	7.403	0.828

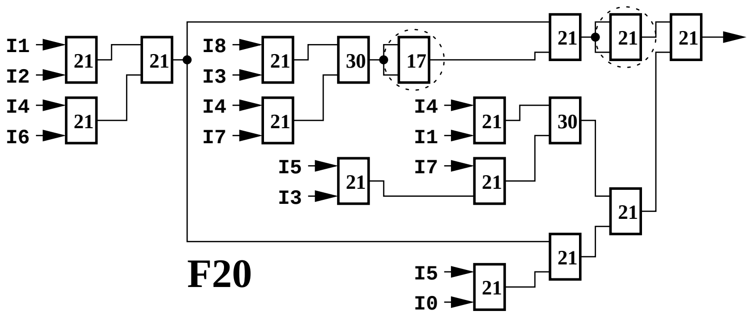

Fig. 3. Evolved filter F20 (with topology 10×4, L-back=2) employs only functions 17, 21 and 30. Marked functions may be omitted.

7 Discussion

7.1 Performance

Evolved filters are compared with conventional approaches in this section. However, we know the type of noise a priori and thus efficient conventional solutions may be prepared for comparison. It is not a case of real world situation, where we may not know anything about the noise and suitable conventional solution may not exist at all. Experiments performed with EHW at simplified functional level allowed us to predict that it is possible to evolve:

1. a filter that exhibits less *dpp* than conventional filters (like mean filters or median) for *each* given training image (without exceptions!), and
2. a filter that exhibits in average less *dpp* than conventional filters for a given noise and test images (e.g. $F24$ for $G16$ noise, $F21$ for $R32$ noise or $F23$ for $N1$ noise).

The result (1) is important especially for a production line where an image recognition system operates with very similar images corrupted by the same noise. Then the evolution leads to very efficient filters since they are not trained only for a given noise, but also for a given class of images (e.g. only capacitors). On the other hand, the result (2) proves generality of the evolved filters.

The filter $F24$ (Figure 2) ranked among the best known filters independently of the noise type in the image. It seems to be very general image filter and may probably be considered as a first solution pattern when the type of noise is unknown a priori. It consists only of 14 functions after manual optimization but evolution needed 21 functions to ensure the same behavior.

7.2 Hardware Requirements

Successful evolution requires adders, i.e. the function 21 at least. The approach does not work without adders as well. The function 21 is an 8bit adder equipped with a right shifter to simulate "average of two" operator. Another improvement is due the functions 30 ("average of two plus one") and 14 ("average of two with a carry"). We have manually replaced all the "average of two plus one" in the filter $F21$ and all the "average of two with a carry" functions in the $F24$ filter by "average of two" function to establish the filters $F21A$ and $F24A$ with uniform (and so cheaper) implementations than $F21$ and $F24$. The *dpp* of the filters $F21A$ and $F24A$ increased about 0.5-2% according to type of noise. But the resulting filter $F24A$ still exhibits better results than any of the conventional filters. If the initial population consists of the gates with the function 21 only, faster convergence occurs. The reason is evident: evolved filters are based on this function.

Evolved filters are compared with conventional $FA4$ filter that ranked as the best of conventional filters in the Table 2. The $FA4$ filter with tree architecture requires four 8bit adders, two 9bit adders, one 11b adder, one 12b adder and four shifters. A cost of hardware implementation of some evolved filters (e.g. $F23$,

$F14$, $F22$, $F18$) is evidently cheaper than for $FA4$ filter. As far the $F23$ filter produces the best *dpp* for the $N1$ noise and, furthermore, its implementation (nine 8bit adders and one logical operation) is cheaper than implementation of $FA4$ filter, the $F23$ outperforms conventional design totally. A cost of the cheapest circuit implementations of the filters for $G16$ and $R32$ noise are comparable with a cost of $FA4$ filter. Optimized $F14$ filter for G16 noise requires at least eleven 8bit adders and a simple logical function 17 while the $F24A$ filter for $R32$ noise consists of 14 adders. These preliminary considerations about hardware cost will be followed by detailed analysis in future work.

As claimed in some papers (e.g. [6]), a sufficient number of the gates is important for efficient evolution. We have applied 40 gates for different topologies (40x1, 20x2 and 10x4) and L-back parameters with similar results, but the *dpp* was significantly worse for 20x1 gates.

7.3 Experiments with the Simulation Model

The fitness calculation is very time consuming since a circuit simulator has to calculate $(N-2)^2$ pixel values. Two approaches were applied to speed up the fitness evaluation: (1) if deterministic selection is used, about 42% of fitness evaluations need not be finished because the fitness value reached after calculation of some number of pixels is worse than the worst already known solution needed for selection. (2) As far the functional element operates over 8 bits and *unsigned int* type occupies 32 bits, four independent pixels can be simulated in the circuit simulator concurrently.

Some experiments did not lead to efficient designs: (1) When we use four training images in the fitness function, only an average filter $F8$ has been evolved. (2) If programmable elements of the reconfigurable circuit operate on four or two bits, the *dpp* is more than 20% higher than for the filters operating on 8bits. (3) If a small training image (32 x 32 and 62 x 62 pixels were tested) is considered for fitness calculation, very efficient filter for a given image is evolved. However the filter fails for another images since it is not general. The optimal size of the training image is a question for future research.

It is also interesting that the best filters for $R32$ noise were evolved using training image with $G16$ noise while the best solutions for $G16$ (and $N1$) noise were evolved using training images with $G16$ ($N1$) noise.

7.4 The Evolvable Component for Image Pre-processing

It seems that the proposed approach can be useful also for other problems of image pre-processing like removal of other types of noise, edge detection, illumination enhancement or image restoration. That is also the reason why all the functions are still supported in the programmable element although many of them have not been used. These "useless" functions for image filters may be important e.g. for edge detection. The goal is to develop the *evolvable component* [15] for image pre-processing (with fixed architecture of the reconfigurable circuit and genetic operators) that should be reused in future designs only by redefinition of the fitness function.

8 Conclusions

In this paper, we experimentally proved that efficient circuit implementations of image filters can be evolved. It was shown for three different types of noise. Evolved filters outperform conventional designs in terms of average difference per pixel. Implementation cost is similar or better than in conventional approaches. It is important that evolved filters are built only from simple primitives like logical functions or 8bit adders that are suitable for hardware implementation. If an image is available both with and without a noise, the whole process of filter design can be done automatically and remotely, without influence of a designer. We plan these possible applications of evolved digital filters at the moment:

– The best-evolved filters will be translated to VHDL, synthesized and stored in a library to be reused instead of conventional filters for a given noise type.
– A new filter will be evolved for a given situation in an industrial application (e.g. for a given camera placed on a production line, which defines certain type of noise that should be removed). In a case of the changes of working conditions, the evolution will be restarted manually to find a better filter. Suitable reconfigurable system for an evolutionary design of image filters will be based on the Virtex Xilinx platform or implemented using the technique proposed in [16].

Acknowledgment

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*. The author would like to acknowledge Dr. Jim Tørresen for continuous support during his work with Department of Informatics, University of Oslo, funded by The Research Council of Norway. The author would like to acknowledge Dr. Otto Fučík for inspiration too.

References

1. Šonka, M., Hlaváč, V., Boyle R.: Image Processing, Analysis and Machine Vision. Chapman & Hall, University Press, Cambridge (1993)
2. Russ, J., C.: The Image Processing Handbook (third edition). CRC Press LLC (1999)
3. Sanchez, E., Tomassini, M. (Eds.): Towards Evolvable Hardware: The Evolutionary Engineering Approach. LNCS 1062, Springer-Verlag, Berlin (1996)
4. Tørresen, 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

5. Miller, J., Thomson, P.: Cartesian Genetic Programming. In: Proc. of the Genetic Programming European Conference EuroGP 2000, LNCS 1802, Springer-Verlag, Berlin (2000) 121–132
6. Miller, J., Job, D., Vassilev, V.: Principles in the Evolutionary Design of Digital Circuits – Part I. In: Genetic Programming and Evolvable Machines, Vol. 1(1), Kluwer Academic Publisher (2000) 8–35
7. 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) 62–72
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 Workshop EvoIASP'99, LNCS 1596 Springer-Verlag, Berlin (1999) 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) 1–11
10. Harvey. N, Marshall, S.: GA Optimization of Spatio-Temporal Gray-Scale Soft Morphological Filters with Applications in Archive Film Restoration. In: Proc. of the Evolutionary Image Analysis, Signal Processing and Telecommunications Workshop EvoIASP'99, LNCS 1596 Springer-Verlag, Berlin (1999) 31–45
11. Miller, J.: Evolution of Digital Filters Using a Gate Array Model. In: Proc. of the Evolutionary Image Analysis, Signal Processing and Telecommunications Workshop EvoIASP'99, LNCS 1596 Springer-Verlag, Berlin (1999) 17–30
12. Tufte, G., Haddow, P.: Evolving an Adaptive Digital Filter. In: Proc of the Second NASA/DoD Workshop on Evolvable Hardware, IEEE Computer Society, Los Alamitos (2000) 143–150
13. Koza, J. et al.: Genetic Programming III : Darwinian Invention and Problem Solving. Morgan Kaufmann Publishers (1999)
14. Erba, M. et al.: An Evolutionary Approach to Automatic Generation of VHDL Code for Low-Power Digital Filters. In: Proc. of the Genetic Programming European Conference EuroGP 2001, LNCS 2038, Springer-Verlag, Berlin (2001) 36–50
15. Sekanina, L., Sllame, A.: Toward Uniform Approach to Design of Evolvable Hardware Based Systems. In: Proc. of the Field Programmable Logic And Applications FPL 2000, LNCS 1896, Springer-Verlag, Berlin (2000) 814–817
16. 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) 161–168