Easily Testable Image Operators: The Class of Circuits Where Evolution Beats Engineers

Lukáš Sekanina and Richard Růžička
Faculty of Information Technology, Brno University of Technology
Božetěchova 2, 612 66 Brno, Czech Republic
sekanina@fit.vutbr.cz ruzicka@fit.vutbr.cz

Abstract

The paper deals with a class of image filters in which the evolutionary approach consistently produces excellent and innovative results. Furthermore, a method is proposed that leads to the automatic design of easily testable circuits. In particular we evolved "salt and pepper" noise filters, random shot noise filters, Gaussian noise filters, uniform random noise filters, and edge detectors.

1. Introduction

The evolutionary circuit design and evolvable hardware represent alternative approaches to the classical engineering design [4, 11]. The classical design is based on detailed analysis of a given problem, mathematical models, top-down decomposition, and abiding by the rules. In case of the evolutionary circuit design a designer is to define a set of programmable elements, specify the number of circuit inputs and outputs, design representation and genetic operators and express the desired behavior in terms of fitness function. Then an *evolutionary algorithm* [5] is responsible for the rest of the work.

It has been shown experimentally that the evolved circuits can compete with the conventional circuits in terms of quality and the implementation cost. If a target application is chosen carefully then the evolutionary design can produce excellent circuits that are quite beyond the scope of conventional engineering approaches [8, 17].

Scalability of representation and circuit verification are primary problems of the evolutionary circuit design. The circuits evolved so far are relatively small and simple devices in comparison with that circuits developed by engineers routinely. Hence it is inescapable to insert a lot of domain knowledge to the evolutionary algorithm in order to evolve more complex circuits and to outperform a random search. In evolvable hardware the knowledge usually takes a form of functional level representation [9], incremental evolution [18] or a developmental process [3].

Similarly to evolution of 1D signal filters, the evolutionary design of image operators and filters belongs to the category where evolvable hardware is quite successful as seen in a number of research reports [2, 6, 10, 13, 14]. Note that the image operators are not trivial circuits. For instance, the circuits—utilizing eight neighboring pixels to filter the pixel values in gray-scaled images (eight bits per pixel)—have 9 $\times$ 8 = 72 inputs and 8 outputs. We have recognized these operators (i.e. so-called 3 $\times$ 3 image operators) as a class of circuits where evolution beats engineering approaches in terms of quality as well as implementation cost very often.

The objective of this paper is twofold. First, we will show that the evolutionary approach consistently produces excellent and innovative circuits for that class of filters. Second, we will also show that some additional requirements can be added into the evolutionary process in order to obtain the circuits with specific features. In our case we will evolve such the circuits that are not only innovative but that are also *easily testable* inherently. From our point of view a circuit is easily testable if each of its elements can be tested separately without any specialized datapath using its primary inputs and outputs only. Note that testability is an important feature of circuits especially when the circuits have to be produced in large series or when we need to diagnose the circuits at a given working place (e.g. in space).

The paper is organized as follows. Second section describes the experimental framework for the evolutionary circuit design. The elementary principles of the design for testability by means of *i path* concepts are introduced in Section 3. In Section 4 the evolved easily testable filters are presented. They are also compared with conventional filters and with those evolved filters where no requirements on testability have been specified. Features of the obtained filters are discussed in Section 5. In particular, an example of application of the test is proposed. Finally, conclusions are given in last section.

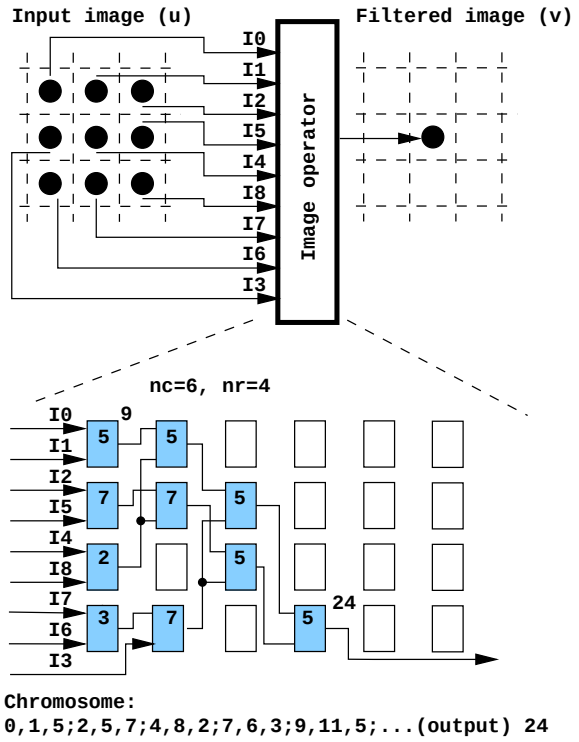

Figure 1. An array of CFBs is configured to operate as an image filter

Table 1. A list of functions that were tested in CFBs. The inputs x and y and the outputs operate over 8 bits. Symbols used: $>>$ right shifter, $<<$ left shifter, $\wedge$ binary AND, $\vee$ binary OR, $\oplus$ binary exclusive-OR, $+$ 8bit adder, $+^s$ 8bit adder with saturation, $\bar{x}$ is a binary negation of x

0	$x >> 1$	1	$x >> 2$
2	$x >> 4$	3	$\bar{x}$
4	$x << 1$	5	$x << 2$
6	$x << 4$	7	$(x << 4) \vee (x >> 4)$
8	0	9	33
10	FF	11	$x \oplus \bar{y}$
12	CC	13	$Max(x, y)$
14	$(x + y + 1) >> 1$	15	$\bar{x} \vee y$
16	$\overline{x \wedge y}$	17	$(x \wedge 0F) \vee (y \wedge F0)$
18	$(x \wedge CC) \vee (y \wedge 33)$	19	$(x \wedge AA) \vee (y \wedge 55)$
20	$x + y$	21	$(x + y) >> 1$
22	$x \vee y$	23	$x \wedge y$
24	$x \wedge \bar{y}$	25	$\bar{x} \wedge y$
26	$x \oplus y$	27	$\overline{x \vee y}$
28	$\overline{x \oplus y}$	29	$x \vee \bar{y}$
30	$((x + y) >> 1) + 1$	31	x
32	$x +^s y$	33	$Min(x, y)$

2. Experimental Framework

In order to evolve a *single* filter (digital circuit), which suppresses a *given* type of noise, we need an original image to measure the fitness values of candidate filters. The generality of the evolved filters (i.e. whether the filters operate sufficiently also for other images of the same type of noise) is tested by means of a test set.

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. 1 shows every pixel value of the filtered image is calculated using a corresponding pixel and its eight neighbors 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 (CFBs) and 8bit datapaths are employed [13]. 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 placed on the leftmost two columns) 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 (usually two) columns. Any CFB can be programmed to realize one of functions taken from Table 1. We have tested various combinations of the functions in CFBs during the experiments; a particular combination will be denoted F.

Similarly to conventional CGP [8], only a very simple variant of the evolutionary algorithm has been developed. Population size is 16. The initial population is generated randomly, however, only function "21" (see Table 1) is preferred. 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. Only mutation of two randomly selected active CFBs is applied per circuit. Four individuals with the highest fitness values are utilized as parents and their mutated versions build up the new population.

The design objective is to minimize the difference between the filtered image and the original image. We chose to measure *mean difference per pixel* (*mdpp*) since it is easy for hardware implementation. 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 circuit simulator 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

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

3. Evolution of Easily Testable Filters

3.1. Fault-Tolerant Circuits and Diagnostics

Evolvable hardware has already been considered as a tool for the design of *inherently fault-tolerant* circuits. Let us suppose that the evolutionary algorithm is responsible for the adaptation of a target system. Then in case of a faulty event (some circuit's elements are damaged), the evolution could find a satisfactory circuit using the remaining elements of a given reconfigurable device. Furthermore, a number of requirements—such as the environmental conditions under which the circuit must operate for some minimal lifetime at a given minimal failure—can be tested in the fitness function [17].

As far as a producer manufactures large series of circuits, the testability of the circuits becomes crucial issue from a business viewpoint. It is a well-known paradigm of the modern circuits design that the testability issues must be reflected already in the design time of a circuit and that the structure of the circuit is modified in order to make the circuit testable. A number of strategies for the "design for testability" have been developed, for instance, BIST, scan techniques, test point insertion and so on [16, 7].

In our case the requirement on testability will not be included into the fitness function; rather, only such a representation of the problem that ensures testability will be employed. We identified that the image operators introduced in the previous sections are suitable for the proposed approach. The method is based on three observations. An operator is easily testable if (1) an output register is connected to every CFB, (2) all the functions supported in CFBs have got so-called *i-mode*, and (3) CFB's inputs are not connected to the same data source. Note that all filters use the CFBs with output registers even if testability is not supported at all. The registers are utilized in order to allow pipelining and their clock signals must be controllable from primary circuit inputs. Then no additional circuits and datapaths are needed to ensure testability and any test might be performed using only primary inputs and outputs of the circuit.

3.2. *i path* concept

When a circuit element is tested, the test patterns must be applied to all its inputs. Then the responses to these patterns are picked-up at the outputs of the tested element. If each element is tested separately, a problem of diagnostic data transport arises. It must be defined beforehand which path will be used to transfer the test pattern from outside the circuit to the tested element input (and similarly for responses). In order to ensure the testability of a circuit, all inputs of all circuit elements must be controllable apart and all output ports of all output elements must be observable apart. The concepts of observability and controllability belong to the traditional approaches in diagnostics.

In this paper, the *i path* (identity path) concept is applied [1]. It means that *i paths* from some primary inputs to all inputs of all tested elements must be identified and also *i paths* from all outputs of all tested elements to some primary outputs must be identified. The concept is as follows:

Element $e1$ with input port x and output port y is said to have an *identity mode* (*i mode*) if $e1$ has a mode of operation in which the data on port x is transferred to port y without being modified. Similarly, there is an *identity transfer path* (*i path*) from output port y of element $e1$ to input port z of element $e3$, if the data at port y can be transferred to port z without being modified. Thus two ports of some elements are in *i path relation* when this *i path* exists.

In our case a CFB can be described by the formula f such that $q = f(a, b)$ (see Table 1). A specific situation appears when b can be found for which $q = a$. Then we say that the CFB is *transparent* for input data in at least one of its operational modes. A simpler situation from an *i path* setting point of view appears if b is a control input. The control inputs are generated by a test controller, therefore it can be stated that element *transparency* can be guaranteed by the test controller. If b is a data input then we denote such a situation as the *data dependent transparency*. It is evident that this mode can be utilized to transfer data (without being changed) from inputs to outputs of the circuit (however a "proper" value must be loaded to b input). As an example a two input adder can be considered. If b input is set to 0 then q output is equal to a.

On the basis of these requirements, some characteristics of testable circuits can be formulated:

- Each element must have one or more *i modes* of operation such that *i paths* from all inputs to the output must exist.
- Each input of each element must be controllable separately, i.e. no inputs of an element can be connected together.

To assure that the diagnostic data can flow through a CFB, both its inputs must be controllable. From our point of view, each input of each element must be controllable from primary inputs. Because wires are always transparent, the main problem of controllability lies in *i modes* of the CFBs. Hence we suppose that all CFBs have got *i modes* which enable us to transfer diagnostic data from any input

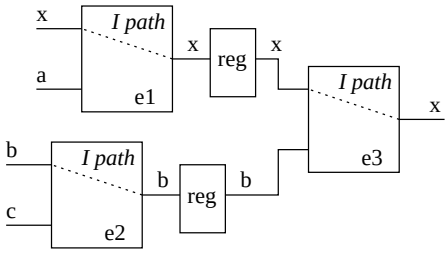

Figure 2. Diagnostic data transfer utilizing *i modes* **of elements**

Table 2. A list of functions implemented in CFBs for the design of easily testable circuits

No	ID	function	*i mode* if
0	22	$x \vee y$	y = 0
1	23	$x \wedge y$	y = FF
2	26	$x \oplus y$	y = 0
3	20	$x + y$	y = 0
4	32	$x +^s y$	y = 0
5	21	$(x + y) >> 1$	y = x
6	13	$Max(x, y)$	y = 0
7	33	$Min(x, y)$	y = FF

to any output. We also assume that the result of the evolution is a pipelined structure of n_c stages where every stage contains a register. No feedback is allowed. In that case, all nodes (and all registers) in the circuit will be controllable and all nodes (all registers) in the circuit will be observable.

An example of diagnostic data transfer in a very simple circuit—which our models in fact operate with—is depicted in Fig. 2. A request to transfer value x from the input of element $e1$ to the output of element $e3$ occurs. To fulfill the request, an *i path* through both elements must be established. As it was mentioned above, the specific values must be loaded to the second input of these elements. Suppose that it is value a for element $e1$ and b for element $e3$. Because the second input of element $e3$ is controlled from another element $e2$, establishing of *i mode* of operation of the element $e2$ takes importance too. Note that in order to control the output of element $e3$, four inputs of other elements must be controllable.

The functions we chose as the suitable building blocks for the evolutionary design are given in Table 2. We can see that each of them has got *i mode* of operation.

4. The Evolved Filters

This section reports the best image filters and operators we have evolved using the setting defined in Section 2 (it is referred to as Phase I in this paper) and the best easily testable image filters and operators we have evolved so far (Phase II). Every subsection contains a table summarizing the results. Its first part is devoted to the conventional filters; the second part reports the filters evolved in Phase I; and the third part shows the easily testable circuits evolved in Phase II (these filters are denoted as FET). The filters were evolved using various setting of CGP parameters. In Phase II the CFBs have supported only functions listed in Table 2. Values "n_c" determine the number of columns of CFBs utilized in CGP (n_r is always 4).

Some of the filters from Phase I have already been reported in [13, 14, 12]—here they are included only for the comparison. The evolved filters are also compared with typical conventional filters and operators such as the median filter (denoted as FME) and the averaging using various weights of coefficients. See, for instance, 3×3 kernel of FA4 filter that employs only the multiples of 2.

$$FA4 = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

Note that FME is an easily testable filter since it consists only of Max and Min elements. FA4 filter is not easily testable because of shifters that are needed to perform multiplication. Note that the filters contain the registers allowing pipelined execution.

Most of the filters were described by means of VHDL and synthesized into Xilinx FPGA XC4028XLA to obtain their implementation cost (the number of equivalent gates denoted as "EqG" in the tables). Some of the evolved filters contain introns. Hence the initial number of CFBs and the number of CFBs after manual optimization (introns removal) are also included in the tables (in columns "CFB" and "Opt"). The "*nsy*" stands for *not synthesized yet* and it denotes the filters we have not considered as interesting for the synthesis at the moment. However, their "EqG" can be estimated easily. If a CFB is not a simple logical function (i.e. it is Max, Min, addition, average etc. equipped with a register) then the CFB costs about 150 equivalent gates.

We have utilized Lena image (256×256 pixels) in the fitness calculation. The best evolved filters were tested on various images and they seem to be general enough. Note that Lena image offers 64516 "training pixels" in our case. Some images are depicted in Fig. 10. The values "*mdpp*" that are included in the tables are valid only for Lena image. The generation in which a given filter has been detected is put in column "gnr". The filter that produces lowest *mdpp* is typed in bold in a given table. If a figure shows a filter

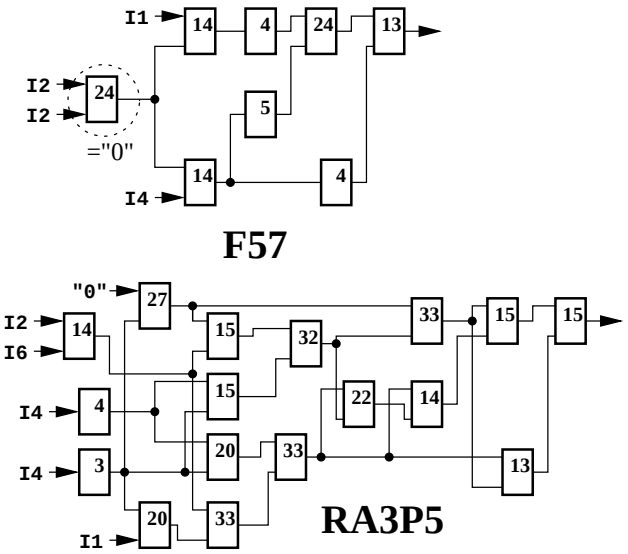

Figure 3. F57 and RA3P5 evolved in Phase I

evolved in Phase I then the functions in CFBs are numbered according to Table 1, otherwise according to Table 2.

4.1. Salt and Pepper Noise Filters

"Salt and pepper" noise (see Fig. 10A) in which 5% of pixels are randomly set up to 0 (black) or 255 (white) values is traditionally suppressed by means of median filter FME. Although visual quality of the images produced by FME is relatively good, FME modifies all the pixels independently of whether they are corrupted or not. As the result, the images are a little bit smudged (see Lena's hair in Fig. 10B). It is also the reason why FME's *mdpp* is worse than *mddp* of the evolved filters. A computationally cheaper ad hoc solution for "salt and pepper" noise—which, unfortunately, does not remove all the "salt and pepper"—utilizes only a simple `if-then-else` function checking an occurrence of 0 or 255. The filter replaces any corrupted pixel by one of its neighbors. The filter (denoted as FIF) could operate, for instance, according to the formula:

$$v(i,j) = \begin{cases} u(i-1,j) & when \quad u(i,j) = 0 \quad or \quad 255, \\ u(i,j) & otherwise. \end{cases}$$

Hence the objective is to evolve a filter that modifies the corrupted pixels only.

A number of interesting filters have been evolved. The F57 filter employed the following functions in CFBs: $F = \{0, 1, \ldots, 29\}$; the other filters of Phase I employed $F = \{0, 1, 3, 4, 5, 7, 8, 10, \ldots, 24, 26, 27, 28, 32, 33\}$ (according to Table 1). F57 filter produces the images very similar to the FIF's output. Fig. 10C shows that some pixels remain unfiltered. RA3P5 is the best filter we evolved in Phase I.

Table 3. "Salt and pepper" noise filters.

Filter	mdpp	CFB	Opt	EqG	gnr	n_c
FME	2.954	–	–	4740	–	–
FA4	9.044	–	–	1397	–	–
FIF	0.782	–	–	129	–	–
F57	1.703	8	7	441	63763	10
RA3P5	0.656	23	17	1702	17539	10
HF3P5	0.746	22	16	1681	17539	10
RF1P5	0.726	27	21	1926	18845	10
HA1P5	1.159	14	11	1492	9314	10
HA5P5	0.940	13	12	1271	16836	10
FET0	0.507	26	26	3656	35968	10
FET7	0.682	24	24	nsy	11313	10
FET9	1.312	8	8	nsy	6757	10
FET12	0.707	22	22	nsy	8601	10
FETX0	**0.379**	18	17	2075	122205	7
FETX3	0.908	15	15	nsy	5392	7
FETX15	0.954	11	11	nsy	33334	5

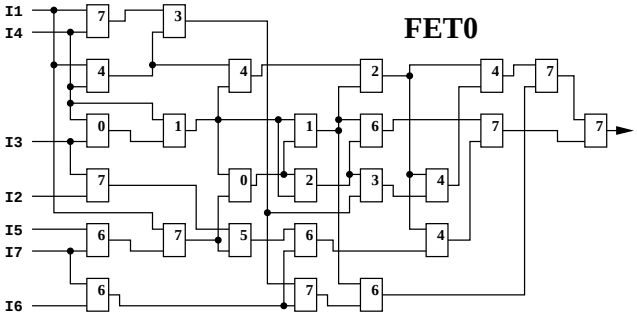

Figure 4. FET0 filter

It produces the output images similar to FME, however, its implementation cost is lower than in case of FME (see Fig. 10D and Table 3). As seen in the same table and in Fig. 10E FETX0 is the best filter we have ever evolved for this type of noise. Furthermore, its implementation cost is less than a half of FME's cost.

4.2. Random Shot Noise Filters

In case of the random shot noise the shots are randomly generated values instead of 0 or 255. The median filter works well independently of whether an image contains random shot noise or "salt and pepper" noise. Similarly to the previous subsection, we have tried to evolve random shot noise filters that could compete with the median filter. Note that FIF filter does not work here at all. The CFBs utilized $F = \{0, 1, 3, 4, 5, 7, 8, 10, \ldots, 24, 26, 27, 28, 32, 33\}$ (for Phase I).

We have not been able to evolve a filter which removes the noise completely; some pixels still remain unfiltered.

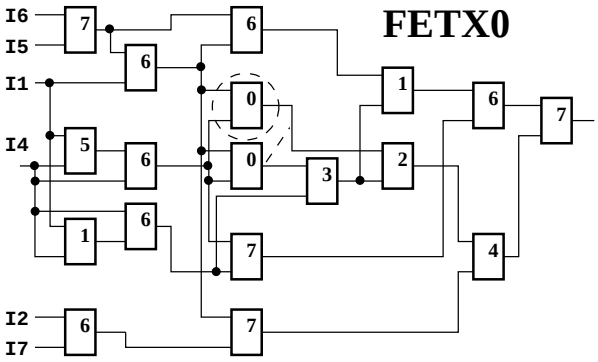

Figure 5. FETX0 filter contains a redundant element whose function can be replaced by its neighbor

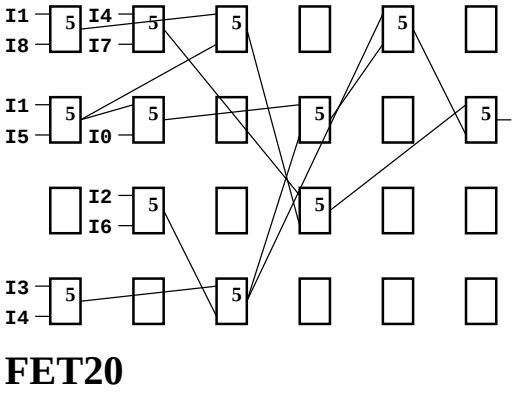

Figure 6. FET20 filter and its placement in a grid of CFBs

Table 4. Random shot noise filters.

Filter	mdpp	CFB	Opt	EqG	gnr	n_c
FME	2.986	–	–	4740	–	–
FIF	4.366	–	–	129	–	–
FA4	6.614	–	–	1397	–	–
FRS2	1.542	7	6	948	16709	10
FRS3	1.612	5	5	790	9097	10
FRS5	1.305	27	22	2486	37435	10
FRS6	1.199	13	10	1422	19170	10
FRS12	1.297	12	11	1360	29028	10
FRS13	1.176	26	25	3288	28925	10
FET30	**1.081**	29	29	3537	51742	10
FET32	1.191	21	21	nsy	28781	10
FET35	1.512	9	9	nsy	623	10
FET37	1.152	28	28	nsy	71538	10
FET38	1.175	25	25	nsy	24381	10
FET39	1.231	20	20	nsy	19215	10

Table 5. Gaussian noise filters.

Filter	mdpp	CFB	Opt	EqG	gnr	n_c
FA4	6.437	–	–	1397	–	–
FME	7.157	–	–	4740	–	–
F24	6.362	21	14	2128	185168	10
F20	6.358	17	15	2626	79369	10
F21	6.354	19	18	3028	133224	20
F23	6.446	10	9	1368	42772	40
HA5G16	6.342	25	25	nsy	14457	10
FET21	6.335	25	25	nsy	79811	10
FET23	**6.243**	31	31	nsy	146539	10
FET24	6.358	24	24	nsy	49561	10
FET28	6.312	31	31	nsy	124206	10
FETX20	6.367	12	12	1824	144498	8
FETX22	6.326	25	25	nsy	153762	8

Nevertheless, all the evolved filters are very interesting if their implementation costs are compared with the cost of FME filter. FET30 filter is depicted in Fig. 7.

4.3. Gaussian Noise Filters

We have tried to suppress Gaussian noise with a mean zero and standard deviation of 16. In case of this type of noise the conventional FA4 filter works well in terms of quality as well as implementation cost.

In Phase I, filter F24 (see Fig. 8) ranked among the best filters we have ever evolved and tested using the test set [13]. F24 consists only of 14 CFBs after manual optimization but CGP needed 21 CFBs to ensure the same behavior. The filters evolved in Phase II have shown in average lower *mdpp* than all the previous ones, however, their implementation costs are relatively high. Filters F20, F21, F23, and F24 were evolved using $F = \{0, 1, \ldots, 29\}$ while we utilized $F = \{0, 1, 3, 4, 5, 7, 8, 10, \ldots, 24, 26, 27, 28, 32, 33\}$ for HA5G16 filter.

4.4. Other Filters

Very similar results have also been obtained for the uniform random noise and the block-uniform random noise [13]. It is interesting that F23 filter (see Fig. 8) outperforms FA4 and FME in terms of quality as well as the implementation cost for the block-uniform random noise. Furthermore, F23 is an easily testable filter since it consists of elements "average" ("21" in Table 1) only. A number of extraordinary filters have also been evolved for Gaussian noise with standard deviation 12, 32, and 40 and for "salt and pepper" noise with 1%, 3%, and 8% of corrupted pixels [12].

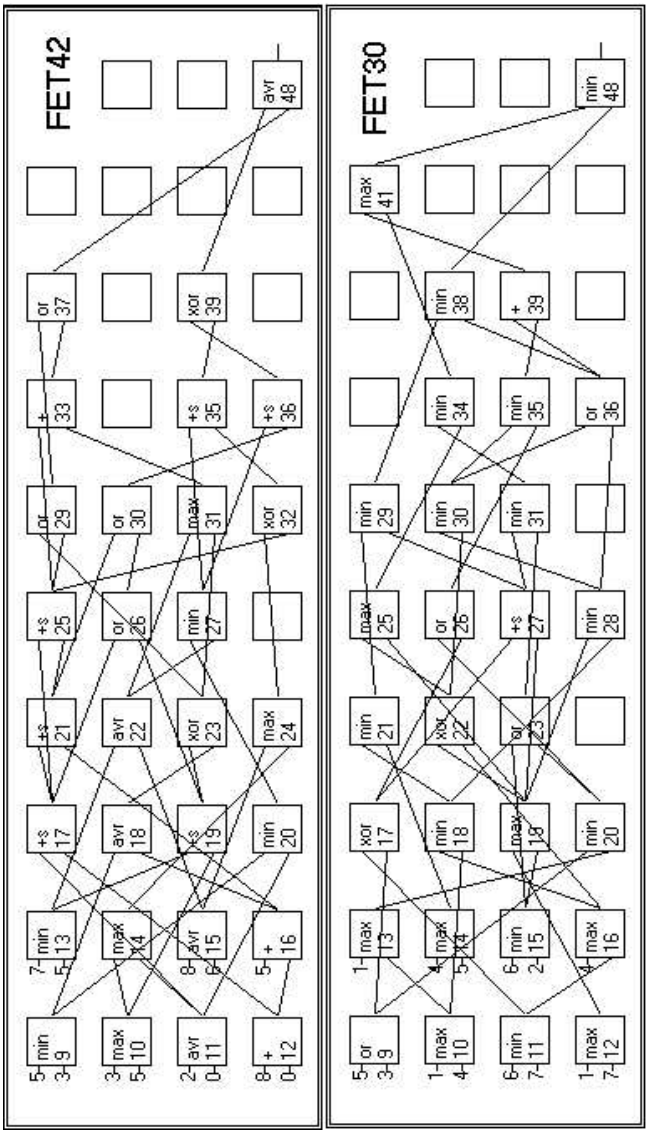

Figure 7. The placement of FET42 (edge detector) a FET30 (random shot noise filter) taken from the filter design tool

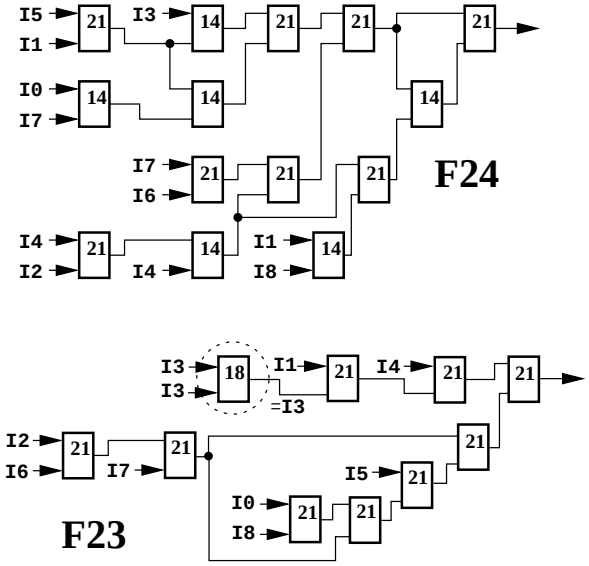

Figure 8. F24 and F23 filters

Table 6. Edge detectors.

Filter	mdpp	CFB	Opt	EqG	gnr	n_c
Sobel	–	–	–	1988	–	–
FS3	–	21	17	1350	29741	10
FS7	1.536	22	16	2079	140557	10
FET42	1.203	29	29	3871	72835	10
FET44	1.349	28	28	nsy	97858	10
FET46	3.545	25	25	nsy	67486	10

4.5. Edge Detectors

In order to evolve edge detectors we have in fact tried to evolve implementations of Sobel operator [15]. In our case the Sobel operator has been defined as an image filter with two convolution kernels that are specified as

$$k_1 = \frac{1}{8}\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \qquad k_2 = \frac{1}{8}\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

Then a new pixel value is calculated according to the following formula

$$NewPixel = 128 + |k_1| + |k_2| \qquad (1)$$

Note that Lena image—filtered by means of the Sobel operator—has been utilized as a target design (Fig. 10F). Hence *mdpp* denotes in Table 6 the differences of the images processed by the evolved operators and the Sobel operator.

The outputs produced by two interesting operators, FS3 and FS7, are depicted in Fig. 10G and 10H. The FS3 operator has been designed with the same CGP parameters as

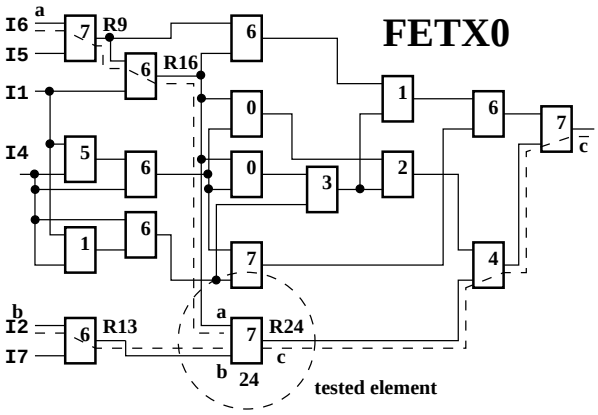

Figure 9. *i path* **created to test CFB$_{24}$ in FETX0 filter**

F57; the other filters have employed the same parameters as RA3P5 filter. Note that FS3 has been evolved using another training image, the "Signs", in the fitness function.

While the implementation cost of FS7 operator is very close to the cost of the conventional solution, the implementation of FS3 is about 30% cheaper than in case of the conventional Sobel operator (see Table 6). As seen in Fig. 10I and Table 6 the evolved easily testable edge detectors are very good, however, relatively expensive.

5. Discussion

5.1. Properties of the Evolved Circuits

The circuits evolved both in Phase I and Phase II exhibit better quality than the conventional circuits (such as FME and FA4) if *mdpp* is measured. It holds not only for Lena image but also for the other images we tested. It is evident that it can not hold for an arbitrary image. However, if a class of images is specified for a given application then the evolved filters are general enough. While the shot noise filters are based on the elements like Max and Min (i.e. if-then-else suppression of corrupted pixels), Gaussian and uniform noise filters perform a sort of "averaging". The shot noise filters clearly show that *mdpp* is not an ideal measure of visual quality. However, *mdpp* represents a uniform approach that might be applied immediately without knowledge of type of noise. Furthermore, its hardware implementation is relatively inexpensive.

It was surprising for us that quality of filters—in terms of *mdpp*—is higher in case of the easily testable circuits. It can be clarified in this way: We restricted in fact the search space to those circuits whose CFBs do not have the inputs connected to the same data source and whose CFBs support only eight functions. And the chosen functions seem to be the right ones for our application domain. Especially, Max, Min and Average are important for successful evolution.

In some cases the evolved circuits require less of equivalent gates than the conventional circuits. It is mainly evident if an evolved circuit is compared with FME filter. The circuits evolved in Phase II contain more CFBs than those from Phase I. Unlike Phase I, some of these CFBs can not usually be removed.

We can conclude that the representation applied in order to evolve easily testable circuits has led to the occurrence of higher quality of operators, however, the evolution utilized all available resources.

Table 7. A test data sequence applied to test CFB$_{24}$ in FETX0 filter

clk	I1	I2	I5	I6	I7	R9	R13	R16	R24
1		b	FF	a	0				
2	0					a	b		
3							a	b	
4							a	b	c

5.2. Testability Analysis

As far as all the CFBs employ registers, provide *i mode* of operation and their inputs must not be connected to the same data source, all the circuits that have been evolved in Phase II are easily testable. As an example, Fig. 9 shows how the CFB with identification 24 can be tested in FETX0 filter.

Remind that every CFB contains a register—registers R9, R13, R16, and R24 are emphasized in Fig. 9. The objective is to transport the values a and b from the primary inputs to the inputs of CFB$_{24}$ and then the output value c to the primary circuit output. Table 7 contains a sequence that has to be performed to obtain c in register R24. Because CFB$_9$ operates as Min(I6, I5), input I5 is set up to FF in order to transfer a to R9. Similarly, zero is loaded into I7 in order to obtain b in R13. Then I1 is utilized to transport a to R16. Finally, CFB$_{24}$ can be tested and its output is available in R24. The same approach is applied to "open a path" from R24 to the primary circuit output and thus to read c and compare its value with a desired vector. Because each CFB of FETX0 can be tested in the proposed way, FETX0 is an easily testable filter.

6. Conclusions

We presented a class of digital circuits in which the evolutionary approach is a really successful design tool. In particular we evolved "salt and pepper" noise filters, random shot noise filters, Gaussian noise filters, uniform random noise filters, and edge detectors.

An open question is whether the idea of the evolutionary design of easily testable circuits could be interesting for some companies right now. It works well for a relatively small class of circuits. However, we could observe that the requirement on easy-testability was useful for the evolution—the fitness values have been increasing remarkably. Because of this requirement we have learned how to reduce the design space. The highest quality circuits that we have ever evolved are recognized as easily testable! The proposed approach could also be presented as a way in which fault-tolerant systems could be realized more effectively.

7. Acknowledgments

The research was performed with the Grant Agency of the Czech Republic under No. 102/03/P176 *Formal approach to digital circuits test scheduling* and No. 102/01/1531 *Formal approach in digital circuit diagnostics – testable design verification*; and the Research intention CEZ MSM 262200012 – *Research in information and control systems*.

References

[1] M. S. Abadir and M. A. Breuer. A Knowledge Based System for Designing Testable VLSI Chips. *IEEE Design & Test of Computers*, pages 56–68, August 1985.

[2] J. Dumoulin, J. Foster, J. Frenzel, and S. McGrew. 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 EvoIASP'00*, volume 1803 of *Lecture Notes in Computer Science*, pages 1–11. Springer-Verlag, 2000.

[3] T. Gordon and P. Bentley. Towards Development in Evolvable Hardware. In *Proc. of the 4th NASA/DoD Conference on Evolvable Hardware EH'02*, pages 241–250, Alexandria, Virginia, USA, 2002. IEEE Computer Society.

[4] T. Higuchi, T. Niwa, T. Tanaka, H. Iba, H. de Garis, and T. Furuya. Evolving Hardware with Genetic Learning: A First Step Towards Building a Darwin Machine. In *Proc. of the 2nd International Conference on Simulated Adaptive Behaviour*, pages 417–424. MIT Press, 1993.

[5] J. Holland. *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press, 1975.

[6] G. Hollingworth, A. Tyrrell, and S. Smith. Simulation of Evolvable Hardware to Solve Low Level Image Processing Tasks. In *Proc. of the Evolutionary Image Analysis, Signal Processing and Telecommunications Workshop*, volume 1596 of *Lecture Notes in Computer Science*, pages 46–58. Springer-Verlag, 1999.

[7] K. Kiefer and H. Wunderlich. Deterministic BIST with Multiple Scan Chains. In *Proc. of IEEE European Test Workshop (ETW)*, pages 39–43, 1998.

[8] J. Miller, D. Job, and V. Vassilev. Principles in the Evolutionary Design of Digital Circuits – Part I. *Genetic Programming and Evolvable Machines*, 1(1):8–35, 2000.

[9] M. Murakawa, S. Yoshizawa, I. Kajitani, T. Furuya, M. Iwata, and T. Higuchi. Evolvable Hardware at Function Level. In *Parallel Problem Solving from Nature PPSN IV*, volume 1141 of *Lecture Notes in Computer Science*, pages 62–71. Springer-Verlag, 1996.

[10] R. Porter, K. McCabe, and N. Bergmann. An Application Approach to Evolvable Hardware. In A. Stoica, D. Keymeulen, and J. Lohn, editors, *Proc. of the 1st NASA/DoD Workshop on Evolvable Hardware*, Pasadena, CA, USA, 1999. IEEE Computer Society.

[11] E. Sanchez and M. Tomassini. *Towards Evolvable Hardware: The Evolutionary Engineering Approach*, volume 1062 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.

[12] L. Sekanina. Evolution of Digital Circuits Operating as Image Filters in Dynamically Changing Environment. In *Proc. of the 8th International Conference on Soft Computing Mendel*, pages 33–38, Brno, Czech Republic, 2002. Brno University of Technology.

[13] L. Sekanina. 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*, volume 2279 of *Lecture Notes in Computer Science*, pages 255–266, Kinsale, Ireland, 2002. Springer-Verlag.

[14] L. Sekanina and V. Drábek. Automatic Design of Image Operators Using Evolvable Hardware. In *Proc. of the 5th IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop DDECS2002*, pages 132–139, Brno, Czech Republic, 2002. Brno University of Technology.

[15] M. Sonka, V. Hlaváč, and R. Boyle. *Image Processing, Analysis and Machine Vision*. Chapman & Hall, University Press, Cambridge, 1993.

[16] N. N. Tendolkar and R. L. Swan. Automatic Diagnostic Methodology for the IBM 3081 Processor Complex. *IBM Journal on Research and Development*, pages 78–89, January 1982.

[17] A. Thompson, P. Layzell, and S. Zebulum. Explorations in Design Space: Unconventional Electronics Design Through Artificial Evolution. *IEEE Transactions on Evolutionary Computation*, 3(3):167–196, 1999.

[18] J. Torresen. A Divide-and-Conquer Approach to Evolvable Hardware. In M. Sipper, D. Mange, and A. Perez-Uribe, editors, *Proc. of the 2nd International Conference on Evolvable Systems: From Biology to Hardware ICES'98*, volume 1478 of *Lecture Notes in Computer Science*, pages 57–65, Lausanne, Switzerland, 1998. Springer-Verlag.

(A) salt and pepper 5% (B) FME (C) F57

(D) RA3P5 (E) FETX0 (F) Sobel

(G) FS3 (H) FS7 (I) FET42

Figure 10. The images produced by some of conventional and evolved operators