Towards New Applications of Multi-Function Logic: Image Multi-Filtering

Lukas Sekanina and Vojtech Salajka
Brno University of Technology, Faculty of Information Technology, IT4Innovations Centre of Excellence
Brno, Czech Republic
Email: sekanina@fit.vutbr.cz

***Abstract*—Multifunctional (or polymorphic) gates are capable of performing two or more logic functions according to the setting of control signals. They can be considered as building blocks for new and cheap reconfigurable chips. In this paper, we utilized multifunctional components that can be implemented using multifunctional gates as building blocks of image filters. We applied genetic programming to evolve image filters performing different filtering tasks under different settings of control signals. Evolved solutions exhibit a significant reduction in utilized operations and interconnects w.r.t. the multiplexing of conventional solutions.**

I. Introduction

A low cost reconfiguration, which does not require an expensive reconfiguration infrastructure (such as switches and configuration memory) could be achieved by means of recently developed *multifunctional gates*. For example, a reconfigurable graphene logic gate design based on graphene pn junctions was proposed which can provide multifunctions just by adjusting some control gate voltages [1]. Another technology is called polymorphic electronics, where new components – CMOS-based *polymorphic gates* – are capable of performing several logic functions as response to various control voltages, V_{dd} levels or temperature [2], [3], [4], [5]. For instance, a 6-transistor NAND/NOR gate controlled by V_{dd} was fabricated in a 0.5-micron HP process [3]. Multifunctional logic could lead to a new class of reconfigurable devices with significantly reduced interconnecting networks.

Designing a single compact circuit, which contains multifunctional gates to implement several required functions is a challenging task. Let us suppose that only two functions (F1 and F2) have to be provided. Fig. 1 (left) shows a conventional multiplexer-based implementation. However, when a suitable multifunctional gate is available then a very elegant solution can be obtained (Fig. 1 (right)). While the first solution is universal and area inefficient, the second solution assumes that one is able to effectively 'merge' F1 and F2 using multifunctional gate(s). If this is possible (and that is our case in Fig. 1 (right)), a significant reduction in the area and routing can be obtained.

Formally, a single network containing multifunctional and ordinary gates has to be constructed such that F1 is implemented for one setting of the global control signal and F2 is implemented for another setting of the control signal. This concept can be generalized for n settings of the control signal. Then we say that a multifunctional circuit operates in n *modes*.

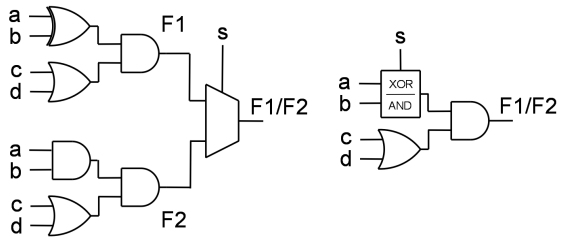

Fig. 1. Implementations of a multifunctional circuit based on multiplexer (left) and multifunctional gate (right)

Several methods have been proposed to design multifunctional circuits [6]. The basic method, called *polymorphic multiplexing*, utilizes a two-input polymorphic multiplexer which propagates signal A in the first mode and signal B in the second mode. Consider that a target polymorphic circuit has to implement functions F and G. A conventional approach is used to synthesize a circuit implementing F and another circuit implementing G independently. The outputs of the circuits are then multiplexed using polymorphic multiplexers. In order to reduce the number of gates, the goal of synthesis can be to maximize the number of gates that are shared by both circuits and minimize the outputs that have to be equipped with polymorphic multiplexers.

The most area-efficient multifunctional circuits have been obtained using evolutionary design, namely Cartesian genetic programming (CGP) [6]. Unfortunately, direct evolutionary design is applicable for very small problem instances only. Theoretical foundations of multifunctional logic such as the completeness theory were presented in [7].

In this paper, we propose to extend the concept of multifunctional gates to multifunctional components at the register-transfer level (RTL). We devise a method for designing of digital circuits that operate as multifunctional image filters. These circuits enable to suppress a given type of noise in the first mode and another type of noise in the second mode of operation of multifunctional components. The method is based on genetic programming which is employed to create a circuit description at RTL. We will show that resulting circuits are functional and area-efficient in comparison with a solution based on multiplexing.

978-3-9810801-8-6/DATE12/©2012 EDAA

II. Proposed Method

Every image filter will be considered as a circuit operating with nine 8-bit inputs (the 3×3-pixel kernel) and a single 8-bit output, which processes grayscale (8-bits/pixel) images. The objective is to propose a filter composed of ordinary and multifunctional components capable of suppressing Gaussian noise in the first mode and shot noise in the second mode.

The shot noise (salt-and-pepper noise) is usually suppressed by a (nonlinear) median filter which calculates the median value from the nine input pixels. The area-optimal implementation consists of a network of 30 two-input components calculating minimum or maximum [8]. The Gaussian noise elimination is based on a simple averaging filter.

A straightforward implementation (without multifunctional gates) of the multifunctional filter for the shot noise and Gaussian noise should perform multiplexing of the median circuit and the averaging circuit. As the structures of both filters are totally different, the final cost (area) is expected to be roughly a sum of the areas required for both filters.

In order to design inherently multifunctional circuits, we will apply CGP [9]. The proposed approach extends the work on (single-purpose) image filter evolution using CGP which has been conducted for several years and led to well-performing and area-efficient filtering circuits [10], [11], [12]. In next subsections, we will describe the proposed circuit encoding in the chromosome, genetic operators, search strategy and fitness function.

A. Problem Representation

In order to model a generic image filter, a candidate circuit is represented as an array of w (columns) $\times$ h (rows) of programmable elements. All candidate circuits have nine 8-bit primary inputs and one 8-bit primary output. Every 2-input programmable element can be connected either to the output of an element placed in previous L columns or to one of the primary inputs. The feedback is not allowed. Programmable elements accept two 8-bit inputs and provide a single 8-bit output. Examples of supported functions are given in Table I. A programmable element can perform either a single function (then it is not a multifunctional element) or two functions (both selected from Table I). In the second case, the programmable element is considered as multifunctional providing that the first function is activated in the first mode and the second function is activated in the second mode. The set of available functions will be denoted as R.

The chromosome is a list of integers starting with the value of c utilized by the first function of Table I. Then, it contains $w \times h$ triplets, each of them encoding a single programmable element, where the first two integers address the connection points for its two inputs and the third integer is the function code. Note that the primary inputs are addressed by $0, 1 \ldots 8$ and programmable elements by $9 \ldots wh + 8$. The last integer defines the primary output of the circuit. Every circuit is thus encoded using $3wh + 2$ integers. An example of chromosome and a corresponding circuit is given in Fig. 2. We assume that the circuit function is controlled by a single Boolean value (not shown in Fig. 2, but denoted as s). In the first mode ($s = 0$), the functions shown in the upper part of boxes are active. In the second mode ($s = 1$), the functions of the bottom part are active. Notice that some elements (13 and 16) are not utilized.

Code	Function	Description
`const`	`c`	constant
`ident`	`x`	identity
`or`	`x | y`	bitwise OR
`nor`	`~(x | y)`	bitwise OR inverted
`and`	`x & y`	bitwise AND
`nand`	`~(x & y)`	bitwise AND inverted
`xor`	`x ^ y`	bitwise XOR
`nxor`	`~(x ^ y)`	bitwise XOR inverted
`_or`	`(~x) | y`	not and bitwise OR
`inv`	`~x`	inversion
`div2`	`x >> 1`	division by 2
`div4`	`x >> 2`	division by 4
`add`	`x + y`	add
`adds`	`min(x + y, 255)`	add with saturation
`mean`	`(x + y) >> 1`	average
`max`	`max(x, y)`	maximum
`min`	`min(x, y)`	minimum

TABLE I
List of functions supported by programmable elements

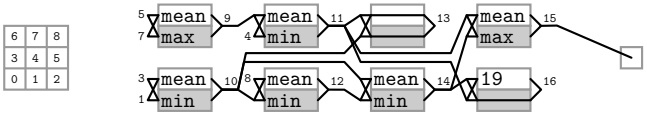

Fig. 2. Example of a candidate bi-functional image filter with CGP parameters $w = 4$, $h = 2$, $L = 2$, $R = \{const/ident, ident, add, max, mean/min, mean/max\}$. Chromosome: 19, 5,7,5, 3,1,4, 9,4,4, 8,10,4, 11,10,1, 10,12,4, 11,14,5, 14,11,0, 15.

B. Search Algorithm

CGP employs a simple $(1 + \lambda)$ evolution strategy to search in the search space [9]. The initial population is randomly generated. Then, it is evaluated and the best-scored individual is considered as the parent for a new population. CGP uses a mutation operator to create λ offspring of the parent to fill the new population. The mutation randomly picks k integers and replaces them by randomly generated (but legal) values. The evolution is terminated after producing g generations.

C. Fitness function

In order to evolve an image filter capable of suppressing a given type of noise, the original uncorrupted (training) image is needed to determine the fitness value. The goal of CGP is to find a circuit minimizing the difference between the original image and the output of the filter. The quality of filtering has to be numerically expressed. For this purpose, the mean absolute error per pixel (*mdpp*) is calculated [10]. In case of multifunctional filters, we have to evaluate both modes of any candidate filter. The fitness value is then calculated as:

$$f = Q - (D_1 + D_2), \tag{1}$$

Fig. 3. Training image, 256×256 pixels

CGP array size	7×4		8×5	
L	4		5	
Max. poly functions	8	unlimited	8	unlimited
Mutations	2	2	2	2
Average $mdpp$	**8.01**	12.14	8.72	13.35
Best $mdpp$	**6.40**	7.32	6.63	7.39

TABLE II
THE BEST AND AVERAGE $mdpp$ FOR VARIOUS CGP PARAMETERS

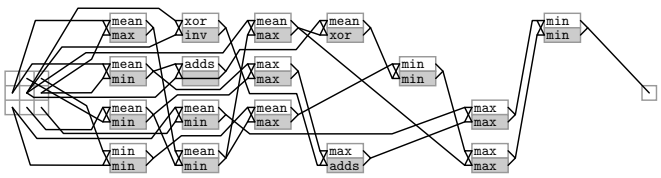

Fig. 4. One of filters evolved for Gaussian/Shot noise

Filter	min	max	mux	div2 logic shift	add mean sub	Sum
Conventional Average					8	8
Conventional Median	15	15	1			31
Average-Median MUX						**39**
Evolved Gaussian (mode 1)	3	4		1	9	17
Evolved Shot (mode 2)	7	6		3	1	17-6*
Gaussian/Shot (Fig. 4)						**28**

TABLE III
THE NUMBER OF OPERATIONS FOR THE MULTIPLEXED CONVENTIONAL FILTER AND EVOLVED MULTIFUNCTIONAL FILTER. *THE OPERATIONS SHARED BY BOTH MODES ARE SUBTRACTED IN THE SUM.

where

$$D_1 = \sum_{i=1}^{N-2} \sum_{j=1}^{N-2} |B_1(i,j) - C_1(i,j)| \quad (2)$$

and

$$D_2 = \sum_{i=1}^{N-2} \sum_{j=1}^{N-2} |B_2(i,j) - C_2(i,j)|. \quad (3)$$

The ideal image which we are attempting to reach in the first mode (second mode) is denoted by C_1 (C_2). The filtered image which was obtained in the first mode (second mode) is denoted by B_1 (B_2). D_1 and D_2 are auxiliary differences. Finally, $N \times N$ denotes the size of image and $Q = 2.2^8.(N-2)^2$ is a constant representing the worst possible score. Figure 3 shows the ideal version of the training image.

III. EXPERIMENTAL RESULTS

The target filter should be capable of suppressing the Gaussian noise ($\sigma = 0.1$ for normalized inputs $\langle 0{,}0; 1{,}0 \rangle$) in the first mode and the shot noise (the 5% salt and pepper noise) in the second mode.

A. Evolved Filter

CGP was utilized with parameters $\lambda = 4$ and $k = 2$. Several combinations of the CGP array size and L-back parameter were tested where each test consisted of 40,000 generations in 20 independent runs. In one experiment, all possible functions and their combinations from Table I were allowed in R. In another experiment, max. 8 polymorphic functions were allowed. The results reported in Table II indicate that the restricted function set, $L = 4$, $w = 7$ and $h = 4$ give the best minimum and average $mdpp$.

The best-evolved filter is shown in Fig. 4. While the mean function is the most frequent one used in the first mode, the functionality of the second mode is based on computing the minimum and maximum. We expected this type of function utilization. Examples of input images and filtered images are given in Fig. 5.

B. Comparisons and Design Time

1) Filtering Quality: The evolved filter was compared in both modes with conventional filters using 16 test images (taken from [12]). We can observe that the average $mdpp$ of the evolved filter for Gaussian noise (first mode) is slightly higher than the $mddp$ obtained for the conventional averaging filter (10.367 vs. 10.019). The evolved salt-and-pepper noise filter (second mode) exhibits lower average $mdpp$ with respect to the median filter (2.267 vs. 4.216).

2) Implementation Cost: Because of the early stage of development of the multifunctional gate technology it is difficult to calculate the area on a chip with a high level of confidence. Hence we have performed our comparison at the RTL and adopted a relatively pessimistic assumption that the cost of a multifunctional element is twofold w.r.t. ordinary elements. Table III shows that we counted the number of operations utilized in both modes, including the output multiplexer in conventional filters. A significant area reduction is observable as 39 operations of the conventional Averaging/Median filter are reduced to 28 operations of the evolved filter. Finally, the same reduction can be achieved in the interconnects assuming that the control signal of multifunctional gates can be distributed inexpensively.

3) Time of Evolution: We measured the running time of CGP ($\lambda = 4$, $g = 500$, $k = 1$, $L = 2$, $w = h = 4$) using a laptop equipped with the Pentium M 1.8 GHz processor. For a small training image (64×64 pixels), CGP finished in 3.06 s. The running time for a large training image (1024×1024 pixels) was 1215.82 s. A typical experiment utilizing a 256×256-pixel image and running for 20,000 generations then took 48

(a) Gaussian noise (b) First mode output (c) Averaging filter

(d) 5% shot noise (e) Second mode output (f) Median filter

Fig. 5. Test images with noise (a, d); images filtered by the evolved filter (b, e); images filtered by conventional filters (c, f)

min. To reduce the design time, multiple runs were carried out on a server with two four-core Intel Xeon 2.66 GHz chips.

IV. Conclusions

We presented multifunctional image filters as a new application of multi-function logic. We utilized multifunctional components that can be implemented using multifunctional or polymorphic gates as building blocks of image filters. In the case study, CGP evolved a solution which exhibits a significant reduction in utilized operations and interconnects w.r.t. multiplexing of conventional solutions. Future work will be devoted to developing new applications of this technology as well as improving the CGP-based design method.

V. Acknowledgments

This work was supported by the Czech science foundation project P103/10/1517, the research programme MSM 0021630528 and the IT4Innovations Centre of Excellence CZ.1.05/1.1.00/02.0070.

References

[1] S. Tanachutiwat, J. U. Lee, W. Wang, and C. Y. Sung, "Reconfigurable multi-function logic based on graphene p-n junctions," in *Design Automation Conference, DAC*. ACM, 2010, pp. 883–888.

[2] A. Stoica, R. S. Zebulum, and D. Keymeulen, "Polymorphic electronics," in *Proc. of Evolvable Systems: From Biology to Hardware Conference*, ser. LNCS, vol. 2210. Springer, 2001, pp. 291–302.

[3] A. Stoica, R. Zebulum, X. Guo, D. Keymeulen, I. Ferguson, and V. Duong, "Taking Evolutionary Circuit Design From Experimentation to Implementation: Some Useful Techniques and a Silicon Demonstration," *IEE Proc.-Comp. Digit. Tech.*, vol. 151, no. 4, pp. 295–300, 2004.

[4] L. Sekanina, R. Ruzicka, Z. Vasicek, R. Prokop, and L. Fujcik, "Repomo32 – new reconfigurable polymorphic integrated circuit for adaptive hardware," in *2009 IEEE Workshop on Evolvable and Adaptive Hardware*. IEEE Computational Intelligence Society, 2009, pp. 39–46.

[5] R. Ruzicka, V. Simek, and L. Sekanina, "Behavior of cmos polymorphic circuits in high temperature environment," in *Proc. of the 2011 IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems*. IEEE CS, 2011, pp. 447–452.

[6] Z. Gajda and L. Sekanina, "On evolutionary synthesis of compact polymorphic combinational circuits," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 17, no. 6, pp. 607–631, 2011.

[7] Z. Li, W. Luo, L. Yue, and X. Wang, "On the completeness of the polymorphic gate set," *ACM Transactions on Design Automation of Electronic Systems*, vol. 15, no. 4, p. 25, 2011.

[8] J. I. Smith, "Implementing median filters in xc4000e fpgas," *Xilinx Xcell*, vol. 23, p. 16, 1996.

[9] J. F. Miller, D. Job, and V. K. Vassilev, "Principles in the Evolutionary Design of Digital Circuits – Part I," *Genetic Programming and Evolvable Machines*, vol. 1, no. 1, pp. 8–35, 2000.

[10] Z. Vasicek and L. Sekanina, "An area-efficient alternative to adaptive median filtering in fpgas," in *Proc. of the 17th Conf. on Field Programmable Logic and Applications*. IEEE CS, 2007, pp. 216–221.

[11] Z. Vasicek, L. Sekanina, and M. Bidlo, "A method for design of impulse bursts noise filters optimized for fpga implementations," in *DATE 2010: Design, Automation and Test in Europe*. European Design and Automation Association, 2010, pp. 1731–1736.

[12] S. Harding and W. Banzhaf, "Genetic programming on gpus for image processing," in *Proc. of the First Int. Workshop on Parallel and Bioinspired Algorithms*. Complutense University of Madrid Press, 2008, pp. 65 – 72.