The Sixteenth International Symposium on Artificial Life and Robotics 2011 (AROB 16th '11),
B-Con Plaza, Beppu,Oita, Japan, January 27-29, 2011

Fault-tolerant Image Filter Design using Particle Swarm Optimization

Zhiguo Bao, Fangfang Wang, Xiaoming Zhao and Takahiro Watanabe
Graduate School of Information Production and Systems
Waseda University
Kitakyushu-shi, Japan
Email: baozhiguo@moegi.waseda.jp

Abstract

This paper describes mixed constrained image filter design with fault tolerant using Particle Swarm Optimization (PSO) on a reconfigurable processing array. There may be some faulty Configurable Logic Blocks (CLBs) in a reconfigurable processing array. The proposed method with PSO autonomously synthesizes a filter fitted to the reconfigurable device with some faults, to optimize the complexity and power of a circuit, and signal delay in both CLBs and wires. An image filter for noise reduction is experimentally synthesized to verify the validity of our method. By evolution, the quality of the optimized image filter on a reconfigurable device with a few faults is almost same as that with no fault.

Keywords

Particle Swarm Optimization, Mixed Constrained Image Filter Design, Fault Tolerant

1 Introduction

The image filter design problem is often approached by means of evolutionary design techniques. In addition to an optimization of filter coefficients (for example, [1]), evolutionary approaches are applied to find a complete structure of image filters. In [2], Gaussian noise filters were evolved using a variant of Cartesian Genetic Programming in which target filters were composed of simple digital components, such as logic gates, adders and comparators. A few years later, image filters for other types of noise and edge detectors were evolved using the same technique [3, 4, 5]. But there were few discussions about fault-tolerance for an image filter design.

Recently chip integration is higher and higher, so that it increases the probability of faulty components and the complexity of designs increases the probability of human errors. The tolerance for faults is diminishing as the systems are demanded for high reliability. Therefore, the needs for fault-tolerant designs are stated as the long-term grand challenges in [6]. To solve this problem, we proposed a fault-tolerant image filter design using GA (Genetic Algorithm) [8], and the experimental results showed that the resultant image filter was surely fault-tolerant. But the problems of quality and processing time still remain, and which evolutionary method (such as GA, GP (Genetic Programming), PSO (Particle Swarm Optimization) and ACO (Ant Colony Optimization)) is most suitable has not been investigated.

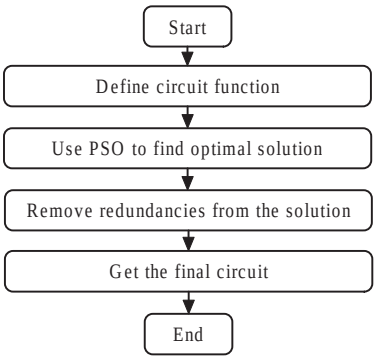

Figure 1: The overview of our method.

This paper describes an efficient image filter design for noise reduction using PSO on a reconfigurable processing array, where some faulty Configurable Logic Blocks (CLBs) may exist at random. The mixed constrained on circuit complexity, power and signal delay in both logic blocks and wires are optimized. In this design, first, the evaluating value about correctness, complexity, power and signal delay are introduced to the fitness function. Then PSO autonomously synthesizes an image filter which is simple and has better performance and fits to the reconfigurable processing array with some faults. To verify the validity of our method, an image filter for noise reduction is experimentally synthesized.

The organization of this paper is as follows: a brief overview of PSO is described in the next section. Section 3 describes fault-tolerant design optimization for an image filter using PSO. Section 4 shows the experimental results. Finally, Sect. 5 concludes this paper.

2 Particle swarm optimization

PSO is an algorithm model on swarm intelligence that finds a solution to an optimization problem in a search space.

In PSO, a particle represents a candidate solution to the problem. Each particle is treated as a point in the D-dimensional problem space. The i-th particle is represented as $X_i = (x_{i1}, x_{i2}, \cdots, x_{iD})$. The best previous position (the position giving the best fitness value) of the i-th particle is recorded and represented as $P_i = (p_{i1}, p_{i2}, \cdots, p_{iD})$. The index of the best particle among all the particles in the population is represented by the symbol g. The rate of the position change (velocity) for particle i is represented as $V_i = (v_{i1}, v_{i2}, \cdots, v_{iD})$. The particle is updated according to the following equations:

$$v_{id}^{(t+1)} = w * v_{id}^{(t)} + c_1 * rand() * (p_{id} - x_{id}^{(t)}) + c_2 * Rand() * (p_{gd} - x_{id}^{(t)}), \quad (1)$$

$$x_{id}^{(t+1)} = x_{id}^{(t)} + v_{id}^{(t+1)}. \quad (2)$$

where,
$0 \le i \le (n-1), 1 \le d \le D.$

n: number of particles in a group.

D: number of members in a particle.

t: pointer of iterations (generations).

w: inertia weight factor.

c_1, c_2: acceleration constant.

$rand(), Rand()$: uniform random value in the range [0,1].

$v_{id}^{(t)}$: velocity of particle i at iteration t, $V_{id}^{min} \le v_{id}^{(t)} \le V_{id}^{max}$.

$x_i^{(t)}$: current position of particle i at iteration t.

The inertia weight factor w is employed to control the impact of the previous history of velocities on the current velocity, thereby influencing the trade-off between global (wide-ranging) and local (fine-grained) exploration abilities of the "flying points". A larger w facilitates global exploration (searching new areas) while a smaller w tends to facilitate local exploration to free-tune the current search area. Good values of w are usually slightly less than 1 [7].

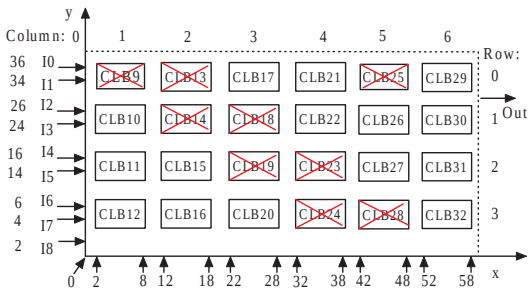

Figure 2: A reconfigurable processing array with faults.

3 Image filter design using PSO

PSO is applied to search good solutions to optimize the image filter design on a reconfigurable processing array with some faults.

The resultant image filter has an identical functional behavior with less circuit complexity, less power and less signal delay.

3.1 Image Filter

Every image operator is considered as a digital circuit with nine 8-bit inputs and a single 8-bit output, which processes gray-scaled (8-bit/pixel) images.

Every pixel value of the filtered image is calculated using a corresponding pixel and its eight neighbors in the processed image [4, 5].

3.2 Reconfigurable Processing Array for Image Filter

The reconfigurable image filter is implemented as a Virtual Reconfigurable Circuits (VRC) (Fig. 2) originally proposed in [4]. As a new pixel value is calculated using nine pixels, the VRC has got nine 8-bit inputs and a single 8-bit output. The VRC consists of two-input Configurable Logic Blocks (CLBs) placed in an array. In our proposed, a $6 * 4$ array is need because its size is enough for our image filter from the results in paper [8]. Any input of each CLB may be connected to either a primary circuit input or the output of a CLB in the preceding column. Any CLB can be programmed to implement one of the functions given in Table 1 [8, 9], all these functions operate with 8-bit operands and produce 8-bit results.

The CLB with different function, has different complexity, power and signal delay. We newly define the values of complexity (FC), power (FP) and signal delay (SD) for each function in a CLB, as in Table 1.

As shown in Fig. 2, the coordinates of inputs and output of each logic block are defined based on VRC, so that we

Table 1: Functions implements in a CLB.

ID	Function	Description	FC	FP	SD
0	255	Constant	8	5	1
1	x	Identity	16	10	2
2	$255 - x$	Inversion	24	15	3
3	$x \vee y$	Bitwise OR	32	20	3
4	$\overline{x} \vee y$	Bitwise $\overline{x}$ OR y	40	25	4
5	$x \wedge y$	Bitwise AND	32	20	3
6	$not(x \wedge y)$	Bitwise NAND	40	25	4
7	$x \oplus y$	Bitwise XOR	64	38	4
8	$x \gg 1$	Right shift by 1	15	9	2
9	$x \gg 2$	Right shift by 2	14	8	2
10	$(x \ll 4) \vee (y \gg 4)$	Swap	16	10	2
11	$x + y$	+ (addition)	358	215	18
12	$x +^{s} y$	+ with saturation	367	220	19
13	$(x + y) \gg 1$	Average	350	210	18
14	$max(x, y)$	Maximum	240	145	16
15	$min(x, y)$	Minimum	240	145	16
-	(wire)	(wire)	16	10	2

FC: function complexity.
FP: function power.
SD: signal delay.

can calculate the critical length of connection wires.

There may be some faulty CLBs in a reconfigurable processing array at random. The output of a faulty CLB is a value in the range [0,255] at random.

3.3 Genetic encoding

The chromosome (particle) is a string of integers where each three continuous integers constitute a logic block. Each triplet in the chromosome encodes the two inputs and the function type of a logic block, respectively, such as:

$$(Input\_1, Input\_2, Function\_type).$$

A typical chromosome then can be a sequence of triplets, such as:

$$X_i = ((IN_1^1, IN_2^1, F_{type}^1) \cdots (IN_1^i, IN_2^i, F_{type}^i) \cdots)$$

Here, IN_1^i and IN_2^i mean positions of the corresponding input signal. F_{type}^i means function type of logic block. For primary input, $0 \leq IN^i \leq 8$. For input from output of a logic block $CLBm$ shown in Fig. 2, $IN^i = m$. Function in a CLB is defined as shown in Table 1.

3.4 Fitness function

The pixels of corrupted image ci are used as inputs of VRC. Pixels of filtered image fi are generated, which are compared to the pixels of original image oi.

The design objective is to minimize the difference between the filtered image and the original image. The image size is $nc*nr$ pixels, but only the area of $(nc-2)*(nr-2)$ pixels is considered, because the pixel values at the borders are ignored, and thus remain unfiltered. The fitness value of a candidate filter is obtained as follows:

(1) the VRC is configured using a candidate chromosome,

(2) the created circuit is used to produce pixel values in the image fi, and

(3) the fitness value is calculated as

$$Fitness = (-1) * (F_1 * \beta + F_2). \tag{3}$$

where,

F_1 and F_2 are defined as follows and β is the weight on F_1.

$$F_1 = \sum_{i=1}^{nc-2} \sum_{j=1}^{nr-2} (|fi(i,j) - oi(i,j)|). \tag{4}$$

where,

nc**:** the number of columns of the pixels in the image.

nr**:** the number of rows of the pixels in the image.

$fi(i,j)$**:** the pixel (i, j) in filtered image fi, the value range is [0,255].

$oi(i,j)$**:** the pixel (i, j) in original image oi, the value range is [0,255].

$$F_2 = SD * \alpha_{sd} + Pg * \alpha_{pg} + Cg * \alpha_{cg} + Pw * \alpha_{pw} + Cw * \alpha_{cw}. \tag{5}$$

where,

SD**:** signal delay of a circuit individual, determined by a critical path.

Pg**:** power of logic blocks in a circuit, calculated by summation of all logic block's power.

Cg**:** complexity of logic blocks in a circuit, calculated by summation of all logic block's complexity.

Pw**:** power of all wires in a circuit, calculated by summation of all wire's power.

Cw**:** complexity of wires in a circuit, calculated by summation of all wire's complexity.

$\alpha_{sd}, \alpha_{pg}, \alpha_{cg}, \alpha_{pw}, \alpha_{cw}$**:** the weights on SD, Pg, Cg, Pw, Cw, respectively.

Table 2: Conditions for evolution.

Number of Generation : 300
Population Size : 1210
Inertia weight factor w : 0.9.
Limit of change in velocity of each member in an individual: $V_{id}^{max} = 0.5 * p_{id}^{max}, V_{id}^{min} = -0.5 * p_{id}^{max}$.
Acceleration constant : c1 = 2, c2 = 2.

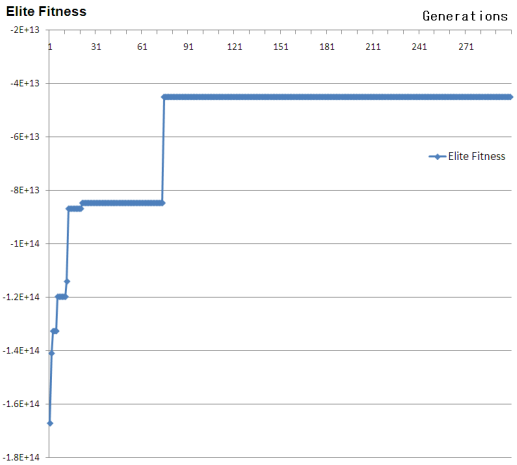

Figure 3: Elite fitness of PSO (Y-axis) vs. the number of generations (X-axis).

The priority of evaluating values in Eq. (3) is: $F_1 > F_2$. In this experiment, β is set to $0.1 * 10^9$. The priority of evaluating values in Eq. (5) is: $SD > Pg > Cg > Pw > Cw$. In this experiment, α_{sd} is set to $0.1 * 10^9$, α_{pg} is set to $0.1 * 10^6$, α_{cg} is set to $1 * 10^3$, α_{pw} is set to 100, and α_{cw} is set to 1. All α's and β are empirically assigned in our experiment.

4 Experimental results

Table 2 shows the parameters of the evolution of PSO used in this experiment. Some preliminary experiments were performed in advance to decide parameters suitable for our experiment.

The proposed method was implemented in Eclipse SDK 3.5.1 with Java Runtime Environment(JRE) 1.6.0; and tested on a PC with Inter(R) Core(TM) i7 CPU at 3.33 GHz and 9.0 GB RAM.

The image filter is evolved for a $512 * 512$ Lena image corrupted by 5% salt-and-pepper noise, shown in Fig. 5 (a).

Fig. 3 shows the elite fitness of PSO vs. the number of generations during the image filter evolution. The elite fitness is increasing during evaluation time.

Table 3 shows the results on a reconfigurable processing array with different faults. For each case, we execute over 10 independent trials. "Available" means the number of available CLBs. "Best one" means the best optimized image filter in terms of Mean Difference Per Pixel (MDPP) [10] among 10 trials, "Average" the average values of 10 individuals, and "Worst one" the worst optimized image filter among 10 trials. The number of used CLBs, the value of the MDPP and running time are listed. "Ratio" is the relative value of MDPP of each case compared to that of no fault $Faults(0)$. The larger the fitness is, the better the quality of image filter is. The less the MDPP value is, the better the quality is. The less the ratio is, the better the quality is.

The quality of the optimized image filter on a reconfigurable processing array with a few faults (2-4 faults) is almost same as that on a reconfigurable processing array with no fault, that is less than 12.6% in the item of different value of MDPP of the best one.

The quality of the optimized image filter of $Fault(2)$ is only 1.2% less than that of $Fault(0)$ in the item of the MDPP value of the best one.

An example of chromosome of best one of $fault(2)$ is as follows:

$(0,0,0)(0,0,0)(1,4,15)(8,0,0)(0,0,0)(0,0,0)$
$(11,12,11)(11,0,0)(0,0,0)(0,0,0)(0,0,0)(15,7,5)$
$(0,0,0)(0,0,0)(0,0,0)(4,20,14)(0,0,0)(0,0,0)$
$(0,0,0)(0,0,0)(16,24,11)(0,0,0)(0,0,0)(0,0,0)$

The graphical representation of this chromosome is shown in Fig. 4.

As the used Lena image size is relatively large, we can say that the resultant evolved filter is general purpose for the same type of noise, that is the filter is able to remove the same type of noise also from other images. Therefor, the image filter was first evolved using Lena image and then tested on other images.

Figures in Fig. 5 show the input image with 5% salt-and-pepper noise, the MDPP value of these images are 6.33, 6.39 and 6.28, respectively. Figures in Fig. 6 show the output image by the image filter of Fig. 4, the MDPP value of these images are 1.74, 1.42 and 1.76, respectively. Obviously, this image filter could reduce noise for all cases, even if there are two faulty CLBs.

5 Conclusions

This paper described mixed constrained image filter design with fault tolerance for noise reduction using PSO on a reconfigurable processing array. By evolution, the quality of the optimized image filter on a reconfigurable processing array with a few faults is almost same as that on a reconfigurable processing array with no fault. Consequently

(a) Lena (MDPP: 6.33) (b) Cameraman (MDPP: 6.39) (c) Airplane (MDPP: 6.28)

Figure 5: The input images with noise.

(a) Lena (MDPP: 1.74) (b) Cameraman (MDPP: 1.42) (c) Airplane (MDPP: 1.76)

Figure 6: The output images by the evolved filter of Fig. 4.

Table 3: Results on a reconfigurable processing array with different faults.

Item	CLBs	Best one								Average				Worst one		
	Available	Used	MDPP	Ratio	SD	Pb	Cb	Pw	Cw	Used	MDPP	Ratio	Time	Used	MDPP	Ratio
Faults(0)	21	8	1.722	1.000	71	645	1064	372	593	6.6	2.365	1.000	405.2	8	2.442	1.000
Faults(2)	19	7	1.742	1.012	88	695	1150	263	419	5.7	2.390	1.011	411.5	7	2.537	1.039
Faults(4)	17	12	1.939	1.126	85	1070	1774	393	625	5.6	2.955	1.249	416.7	5	3.453	1.414
Faults(6)	15	6	2.008	1.166	50	550	910	227	362	5.4	2.965	1.254	425.2	4	3.525	1.443
Faults(8)	13	3	3.117	1.810	48	435	720	198	316	4.8	3.624	1.533	432.0	2	3.889	1.593

$Available$: In a 6*4 CLBs array, only one CLB is used in the last column, so the max number of available CLBs is 21.
$Faults(x)$: A reconfigurable processing array with x faulty CLBs at random position, $x = 0, 2, 4, 6, 8$.
$Used$: the number of the used CLBs.

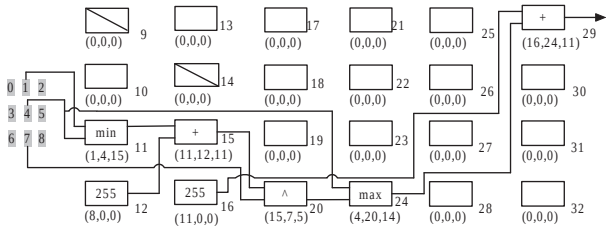

Figure 4: The optimized image filter of $Fault(2)$.

our proposed design method is effective for fault-tolerant optimization.

We will also apply PSO to autonomous design circuits for more complex functional requirements, and enhance more practical information about circuit to fitness function. Future works are to find better genetic encoding method to apply PSO to large sized circuits, and to improve PSO to reduce the processing time.

Acknowledgements

This research was supported by Waseda University Global COE Program 'International Research and Education Center for Ambient SoC' sponsored by the Ministry of Education, Culture, Sports, Science and Technology (MEXT), Japan, and partially by the Regional Innovation Cluster Program 2nd stage of MEXT, Japan and the Core Research for Evolutional Science and Technology (CREST) of Japan Science and Technology Agency (JST).

References

[1] J. Dumoulin, J. Foster, J. Frenzel, et al. (2000), "Special Purpose Image Convolution with Evolvable Hardware," Real-World Applications of Evolutionary Computing, vol. 1803 of LNCS, Springer, 2000, pp. 1-11.

[2] L. Sekanina (2002), "Image Filter Design with Evolvable Hardware," Applications of Evolutionary Computing, (the 4th Workshop on Evolutionary Computation in Image Analysis and Signal Processing, EvoIASP 2002), vol. 2279 of LNCS, Springer, 2002, pp. 255-266.

[3] L. Sekanina (2004), Evolvable components: From Theory to Hardware Implementations, Springer.

[4] T. Martinek and L. Sekanina (2005), "An Evolvable Image Filter: Experimental Evaluation of a Complete Hardware Implementation in FPGA," Evolvable Systems: From Biology to Hardware, vol. 3637 of LNCS, Springer, 2005, pp. 76-85.

[5] Z. Vasicek and L. Sekanina (2007), "Evaluation of a New Platform For Image Filter Evolution," Proc. the Second NASA/ESA Conference on Adaptive Hardware and Systems, 2007 (AHS 2007), Aug. 2007, pp. 577-586.

[6] International Roadmap Committee (2009), "Executive Summary, International Technology Roadmap for Semiconductors," http://www.itrs.net/Links/2009ITRS/Home2009.htm.

[7] Jacob Robinson and Yahya Rahmat-Samii (2004), "Particle swarm optimization in electromagnetics", IEEE Transactions on Antennas and Propagation, Volume 52, Issue 2, Feb. 2004, pp. 397 - 407.

[8] Z. Bao and T. Watanabe (2009), "Evolutionary Design for Image Filter using Genetic Algorithm," Proc. IEEE TENCON 2009, Singapore, Nov. 2009, pp. 1-6.

[9] Z. Bao and T. Watanabe (2010), "Mixed Constrained Image Filter Design Using Particle Swarm Optimization," Proc. The Fifteenth International Symposium on Artificial Life and Robotics 2010 (AROB 15th 2010), Beppu, Oita, Japan, pp. 230-235.

[10] B. Rajan and S.Ravi (2006), "FPGA Based Hardware Implementation of Image Filter With Dynamic Reconfiguration Architecture," IJCSNS International Journal of Computer Science and Network Security, Vol. 6, No. 12, pp. 121-127.