Hardware-in-the-Loop Simulation based Evolutionary Design of Image Filter

Kaifeng Zhang, Huanzhang Lu, Shanzhu Xiao, Weidong Hu
ATR key lab, National University of Defense Technology, Changsha, China
zkf0100007@163.com

***Abstract*—An evolutionary design method of image filter based on hardware-in-the-loop (HIL) simulation is proposed. An image filter evolvable system based on System Generator was designed. Experimental results and analyses demonstrate that the optimal evolved filter outperforms traditional filters both on performance and implementation costs. In comparison with the existing evolutionary design methods of image filter, the proposed strategy brings higher computational ability and more flexibility. Evolutionary algorithm of the evolvable system can be synthesized into hardware which provided a basis for the follow-up research to achieve online adaptive filtering.**

***Index Terms*—Evolutionary algorithm, evolvable hardware, hardware-in-the-loop simulation, image filtering.**

I. INTRODUCTION

Image filters widely used in the pre-processing phase, are an important part of the image processing systems. Traditional design process of image filter is mostly based on experimental work with the images and it leads to a very time consuming job [1]. Evolvable hardware (EHW) [2]–[7] was introduced to the design of image filter, due to the limitations of traditional image filter design methods. EHW includes evolutionary design and online adaptive hardware. Evolutionary design refers to self-reconfiguration hardware design, where the configuration is under the control of an evolutionary algorithm (EA) [4]. The evolutionary design method can explore the regions in design space that are beyond the scope of conventional methods [2]. In recent years, several studies are being focused on the evolutionary design of image filter [1], [8]–[12]. These methods can be classified into different classes based on the location of the fitness evaluation module and the evolutionary algorithm:

1. Extrinsic hardware evolution (EHE)—The image filters were evolved only by using a virtual reconfigurable circuit (VRC) simulated in software, no evolution in hardware was performed [1]. The main drawback of this approach is that the discrepancy between hardware and simulation models. Another disadvantage is the slow evolution speed due to the serial execution feature of software.
2. PC-based intrinsic hardware evolution (PCIHE)—The candidate circuits were evaluated in a reconfigurable device while EA was executed on a PC or a cluster of workstations [9], [10]. This approach suffers from slow speed because of communication delays between PC and reconfigurable device.
3. Complete hardware evolution (CHE)—The evolving filter and the EA are both implemented on a single chip. The primary advantages of this approach are high speed, low cost and potentially low power consumption in comparison with a solution which utilises a common PC. But due to the finite logic resources available on a single chip, the complexity and number of fitness functions supported are limited.

Although the PCIHE suffers from high communication delays, it is useful in applications where the fitness evaluation time dominates the communication time. Especially, in the design phase, evolution time may not be a main concern. In this paper, we proposed a hardware-in-the-loop (HIL) simulation based evolutionary design method of image filter. The EA runs in MATLAB/Simulink, while the fitness evaluation which is the most time consuming part, is implemented on hardware. Unlike the approach suggested in [1], the EA runs in MATLAB/Simulink can be synthesized into hardware directly.

The rest of this paper is organized as follows. Section II gives a brief introduction to HIL simulation, and then depicts the architecture, operational mechanism of the proposed evolvable system in detail. Section III reports the experiments conducted to verify the validity of our proposed approach and demonstrate its effectiveness. Section IV concludes with a brief summary of the features and advantages of the proposed approach and discusses the future directions.

II. HIL SIMULATION AND EVOLVABLE SYSTEM DESIGN

A. HIL Simulation

Software simulation is a time-consuming work, and due to the difference between the model and real hardware, the simulation results depend on the accuracy of the model.

HIL simulation means that the hardware is incorporated in a simulation loop, which allows a portion to be tested in actual hardware while the rest part runs in software. HIL simulation is achieved by System Generator and FPGA platform. System Generator is a DSP design tool from Xilinx that enables the use of MATLAB/Simulink for FPGA design. System Generator provides accelerated simulation through hardware co-simulation. This hardware will co-simulate with the rest of the Simulink system to provide up to a 1000 × simulation speed-up.

Manuscript received April 1, 2013; accepted January 21, 2013.
This research was funded by a National 863 High-Tech Research and Development Plan of China (No. 2009AA8050701).

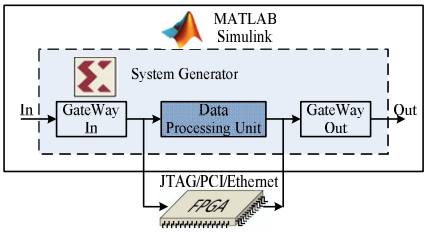

Fig. 1. Schematic of soft-hardware co-simulation.

As can be seen from Fig. 1, the *GateWay In* and *GateWay Out* module define the boundary for FPGA. The function blocks between the two modules can be synthesized into working hardware i.e. FPGA platform. Depending on the hardware configuration, the communication interface between FPGA platform and PC can be JTAG (Joint Test Action Group), PCI or Ethernet interface.

B. Proposed Evolvable System

The proposed evolvable system consists of a MATLAB/Simulink program running on the host PC and the virtual reconfigurable circuits implementing on a FPGA platform.

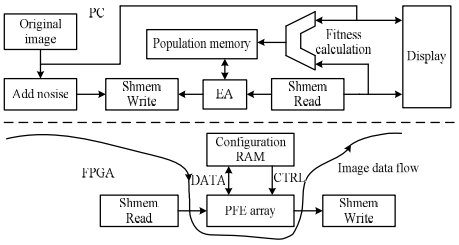

Fig. 2. Block diagram of evolvable system.

As depicted in Fig. 2, the VRC on FPGA is composed of three modules: PFE array unit, From/To FIFO unit and Configuration RAM. The VRC was implemented on Xilinx ML506 board. The 1 Gbps Ethernet interface is used for data transmission between PC and FPGA board. The configuration RAM is implemented by the Unprotected Shared Memory (USM) and auxiliary control logic. By altering the content of the USM, the VRC could be reconfigured by the PC at runtime. When the evolution begins, the image data are transferred to the FPGA and those data are fed into the input of the programmable function element (PFE) array unit. The PC reads the filtered image data from the FPGA, and calculates the fitness value. The image data transmission can be realized by Shmem Write/Read API functions provided by MATLAB. The original image and filtered image are displayed for comparing the performance of evolving filter. The evolution process is under the control of the EA unit, when the stop criteria of the EA satisfied, the expected filter is obtained.

In this work, the VRC employed is similar to the earlier model of Wang *et al*. [11], which is a two-dimensional PFE array. The VRC is the core component of the evolvable system, similar to the two-dimensional geometry of Cartesian Genetic Programming (CGP). The advantage of the two-dimensional geometry is that the genotype representation used is independent of the data type of the phenotype, which brings more flexibility.

As shown in Fig. 3, the PFE array is composed of eight PFE columns, and each column includes four PFEs except the last column. The levels back parameter L determines the connectivity between PFEs. We chose L = 1 here, which means that the input of each PFE can only be connected to the output of its previous column. The 2-input, 1-output PFE can be configured to realize one of eight functions shown in Table I. The connections between PFEs are implemented by means of multiplexers that are controlled by configuration information. In order to achieve pipeline processing, registers are inserted in the in/output of each PFE.

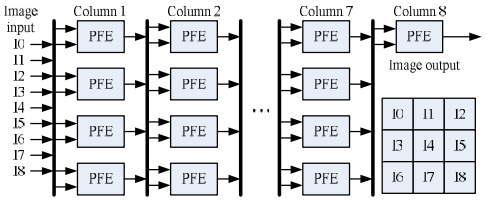

Fig. 3. PFE array for the evolution of image filter.

TABLE I. FUNCTIONS OF PFES.

Index	Function	Index	Function
0	A	4	MIN(A, B)
1	(A+B)>>1	5	A<<1
2	(A+B+1)>>1	6	A XOR B
3	MAX(A, B)	7	B

Fitness evaluation includes two phase: filter output response acquisition and fitness calculation. The image filtering is implemented in hardware, resulting in a significant speedup over the software implementation. The fitness calculation can be easily achieved in MATLAB by using matrix operations. The mean difference per pixel (MDPP) is employed to measure the quality of the filtered image, which is defined as follows

$$MDPP = \frac{\sum_{i=1}^{R-2}\sum_{j=1}^{C-2}\left|v(i,j)-w(i,j)\right|}{(R-2)\cdot(C-2)}, \quad (1)$$

where R and C signify the row and column of the image respectively, $v(i, j)$ is the pixel value of the original uncorrupted image , and $w(i, j)$ is the pixel value of the filtered image.

C. Chromosome Encoding

The chromosome encoding has a significant impact on the efficiency and the convergence rate of evolutionary algorithms. In this paper, the VRC configuration information is relatively simple; a direct coding method was employed. The structure of chromosome was shown in Fig. 4.

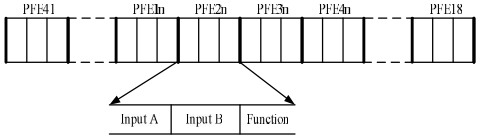

Fig. 4. Structure diagram of chromosome.

The parameters of the VRC are defined as follows: n_c = 8, n_r = 4, n_i = 9, n_o = 1, which signify the column number, the row number, the input number and the output number of the VRC respectively.

As the levels back parameter L = 1, the chromosome length is described as follows

$$L_{chrom} = (\lfloor n_i \rfloor \times n_n + \lfloor n_f \rfloor) \times n_r + (\lfloor n_r \rfloor \times n_n + \lfloor n_f \rfloor) \times \times n_r \times (n_c - 2) + (\lfloor n_r \rfloor \times n_n + \lfloor n_f \rfloor) \times n_o. \quad (2)$$

In (2), $\lfloor x \rfloor$ is the minimum length of the binary string of x, n_f is the length of select bit string of PFE function, n_n is the number of inputs of the PFE. The chromosome length is $(4 \times 2 + 3) \times 4 + (2 \times 2 + 3) \times 4 \times 6 + (2 \times 2 + 3) = 219$ bits. The data structure of the individual *Ind* can be defined as follows:

```
Ind.Chrom[Row][Col]
Ind.Fitness
```

Here *Chrom* is the chromosome, *Fitness* is the individual fitness value, *Row* and *Col* indicate the row and column of the corresponding PFE respectively.

D. Evolutionary Strategy

In this paper, the EA used in the evolutionary design process is evolutionary strategy (ES). The genetic operator of the EA includes selection and mutation, no crossover is applied.

The hardware evolution basic process is as follows:

Step 1: A set of initial population were randomly generated;
Step 2: Configure the circuit. The configuration information is downloaded to the configuration RAM;
Step 3: Feed the noisy image data to the FPGA platform, and calculate the difference between filtered pixel value and the original image to obtain the fitness value. If the circuit meets the requirements or the specified iteration is exhausted, then go to step 5, otherwise go to step 4;
Step 4: Evolution operation (including selection, mutation) to create offspring, go to step 2;
Step 5: Evolution terminated.

III. EXPERIMENTS AND RESULTS

The EA was written in M language provided by MATLAB. The VRC module is a Black Box in System Generator, which implemented by hardware description language. The other module can be realized by predefined function module provided by System Generator.

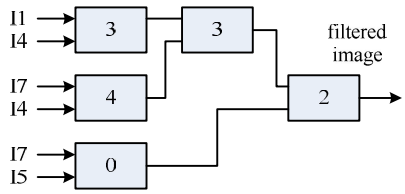

Fig. 5. Block diagram of EF3. The functions of PFEs are defined in Table I.

A 256 × 256 Lena image corrupted by Salt & Pepper noise (5 % pixels with white or black shots) was used for training our evolved image filter. The employed EA is (1 + λ) evolutionary strategy, where λ = 4. The stop condition of the EA is defined as follows: the predefined maximum generation number is exhausted or the expected solution is obtained (MDPP below 5 when dealing with Salt & Pepper noise). The mutation rate is 0.03, and the maximum generation number is 20000.

50 independent runs were performed and one of the evolved filters is shown in Fig. 5.

The filtering results of evolved filters and median filter (MF) are shown in Fig. 6.

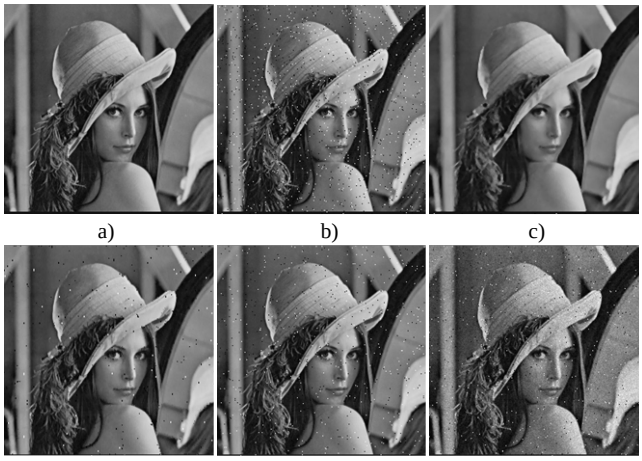

Fig. 6. Filtering results of Lena. (a) Original image. (b) Corrupted image (Salt & Pepper noise). (c) Restored by MF. (d) Restored by EF1. (e) Restored by EF2. (f) Restored by EF3.

As seen from Fig. 6, the Salt & Pepper noise was suppressed by MF, but suffering significant loss in the image detail. The evolved filters suppressed the Salt & Pepper noise without obvious detail loss. Besides the filtering performance, the logic resource utilization is also an important measure for the hardware implementation of image filter. The filters were synthesized using Xilinx ISE 10.1 and mapped to a XC5VSX50T FPGA. Table II shows the logic utilization, MDPP and convergence generations of filters.

TABLE II. PERFORMANCE COMPARISON OF FILTERS.

Filters	MDPP	Logic utilization (Slice)	Convergence generations
MF	3.496	256	—
EF1	2.050	51	15732
EF2	3.693	96	10301
EF3	1.903	58	15667

As seen from Table II, EF1 and EF3 outperform the MF in terms of MDPP and logic utilization.

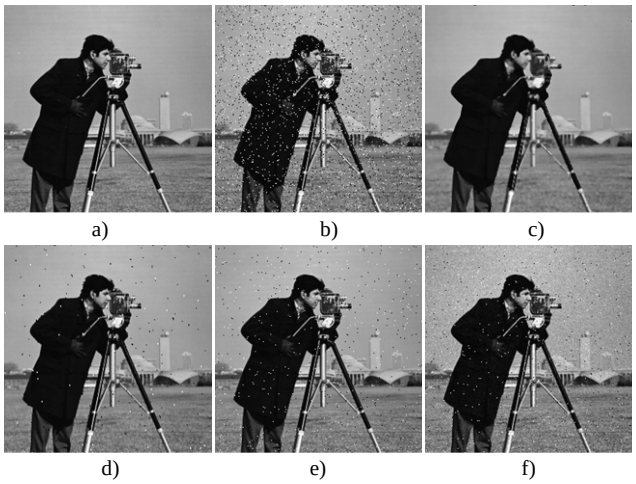

Fig. 7. Filtering results of Cameraman. (a) Original image. (b) Corrupted image (Salt & Pepper noise). (c) Restored by MF (MDPP=4.383). (d) Restored by EF1 (MDPP=2.449). (e) Restored by EF2 (MDPP=4.512). (f) Restored by EF3 (MDPP=2.351).

In order to verify the generalization of the evolved filters, the evolved filters trained by Lena image were used to restore Cameraman image corrupted by Salt & Pepper noise.

The evolved filters are usually general, which are

demonstrated in Fig. 7. The evolved filters also outperform the MF in terms of MDPP.

To make an equal comparison with existing methods, a Lena image corrupted by Gaussian (mean 0 and variance 0.008) is also employed to train our evolved filters. In the case of Gaussian noise, we chose MDPP below 10 as the stop criteria. Table III summarized the performance comparison of the existing evolutionary design method of image filter.

TABLE III. PERFORMANCE COMPARISON OF EVOLUTIONARY DESIGN METHOD OF IMAGE FILTER.

Evolutionary design method	Noise type/ FPGA clock frequency (MHz)	MDPP	Time cost (s)	Logic utilization (slice)
CHE [11]	Salt & Pepper/33	1.86	128	71
CHE [11]	Gaussian/33	8.39	128	85
PCIHE [10]	Gaussian/ Asynchronous	17.5	5100	97
Proposed	Salt & Pepper/100	1.903	247	58
Proposed	Gaussian/100	9.035	282	82

The PCIHE suffers from the lowest running speed, due to the asynchronous feature that no pipeline processing could be employed. As seen from Table III, the CHE outperforms our proposed method in terms of filtering performance, but occupying more logic area. By comparison with the evolution speed, the proposed method is slower than the CHE, but outperforms the PCIHE significantly.

It takes about 8.3 ms to make a median filtering operation for a 256 × 256 image using MATLAB (R2007b) on an Intel DualCore E6600 (2.4 GHz) with 2 GB DDR II RAM. For the HIL simulation, FPGA operating clock is 100 MHz, and the theoretical time for processing a 256 × 256 image is 256 × 256 × 10 ns ≈ 0.66 ms, a speedup of approximately 12.5 × can be achieved. The total evolution time cost t can be defined as follows

$$t = t_{init} + N \times (M \times (t_{cfg} + t_{filter} + t_{tran}) + t_{EA}), \qquad (3)$$

where t_{init} is the system initialization time, N is the number of generations, M is the population size, t_{cfg} is the configuration time of VRC, t_{filter} is the filtering time, t_{tran} is the data transmission time between PC and FPGA, and t_{EA} is the time required to run EA. Table IV summarized the time cost for evolution in detail.

TABLE IV. TIME COST FOR EVOLUTION.

t_{init}	t_{cfg}	t_{filter}	t_{tran}	t_{EA}
32 s	0.024 ms	0.66 ms	2.16 ms	1.32 ms

As shown in Table IV, t_{init} takes the most time, while the system initialization only runs once at the beginning. The data transmission time t_{tran} also takes much time. It is worth pointing out that the filtering operation in our work is relatively simple, and the data processing time is much less than the data transmission time. In the case of more complex computing algorithm employed, much more advantage of hardware acceleration could be obtained.

TABLE V. SYNTHESIS RESULTS IN XC5VSX50T FPGA.

Resource	*Available*	*Used for evolvable system*	*Used for EA*
slices	8160	3019 (37 %)	1061 (13 %)
flip flops	32640	7507 (23 %)	2285 (7 %)
LUTs	32640	5875 (18 %)	1632 (5 %)

The EA is implemented using MCode which is a limited subset of the M language. As a result, the EA can be synthesized into hardware directly together with the VRC and other auxiliary logic modules. The synthesis results are as shown in Table V. According to the synthesis reports, the maximum FPGA clock frequency attained is 135.41 MHz.

IV. CONCLUSIONS

A HIL simulation based evolutionary design method of image filter is proposed. The experimental results show that it can effectively improve the evolution speed of traditional PCIHE. And the optimal evolved filters have better filtering performance and lower implementation cost than existing methods. Furthermore, in contrast to CHE, by taking advantage of MATLAB, our method has more powerful computing capability and flexibility.

In addition, this paper only studied the evolutionary design of static fitness function. However, the proposed method also can be used for dynamic adaptive fitness function, such as hardware online adaptive filtering. The online adaptation is a challenging issue in evolvable hardware research areas, which is also the focus of future research work.

REFERENCES

[1] L. Sekanina, V. Drabek, "Automatic design of image operators using evolvable hardware", in *Proc. 5th IEEE Design and Diagnostics of Electronic Circuits and Systems*, Czech Republic, 2002, pp. 132–139.

[2] A. Thompson, P. Layzell, R. S. Zebulum, "Explorations in design space: unconventional electronics design through artificial evolution", *IEEE Trans. Evolutionary Computation*, vol. 3, pp. 167–196, 1999. [Online]. Available: http://dx.doi.org/10.1109/4235.788489

[3] J. Q. Xu, Y. Dou, Q. Lv, "A bio-inspired fault-tolerant hardware system supporting hierarchical self-healing", *Elektronika Ir Elektrotechnika*, vol. 4, pp. 103–106, 2012.

[4] E. Stomeo, T. Kalganova, C. Lambert, "Generalized disjunction decomposition for evolvable hardware", *IEEE Trans. Systems, Man, and Cybernetics*, vol. 36, pp. 1024–1043, 2006. [Online]. Available: http://dx.doi.org/10.1109/TSMCB.2006.872259

[5] P. Kaufmann, K. Glette, T. Gruber, M. Platzner, J. Torresen, B. Sick, "Classification of electromyographic signals: comparing evolvable hardware to conventional classifiers", *IEEE Trans. Evolutionary Computation*, vol. 17, pp. 46–63, 2013. [Online]. Available: http://dx.doi.org/10.1109/TEVC.2012.2185845

[6] F. Canare, S. Bhadari, D. B. Bartolini, M. Carminati, M. D. Santambrogio, "A bird's eye view of FPGA-based evolvable hardware", *2011 NASA/ESA Conf. on Adaptive Hardware and Systems*, USA, 2011, pp. 169–175. [Online]. Available: http://dx.doi.org/10.1109/AHS.2011.5963932

[7] R. A. Ashraf, R. F. DeMara, "Scalable FPGA refurbishment using netlist-driven evolutionary algorithms", *IEEE Trans. Computers*, vol. 62, pp. 1526–1541, 2013. [Online]. Available: http://dx.doi.org/10.1109/TC.2013.58

[8] Z. Vasicek, M. Bidlo, L. Sekanina, K. Glette, "Evolutionary design of efficient and robust switching image filters", *2011 NASA/ESA Conf. on Adaptive Hardware and Systems*, USA, 2011, pp. 192–199. [Online]. Available: http://dx.doi.org/10.1109/AHS.2011.5963935

[9] Z. Vasicek, L. Sekanina, "An evolvable hardware system in Xilinx Virtex II Pro FPGA", *Int. J. Innovative Computing and Applications*, vol. 1, pp. 63–73, 2007. [Online]. Available: http://dx.doi.org/10.1504/IJICA.2007.013402

[10] Y. Zhang, S. Smith, A. M. Tyrrell, "Digital circuit design using intrinsic evolvable hardware", *2004 NASA/DoD Conf. on Evolvable Hardware*, USA, 2004, pp. 24–26.

[11] J. Wang, Q. S. Chen, C. H. Lee, "Design and implementation of a virtual reconfigurable architecture for different applications of intrinsic evolvable hardware", *IET Computers & Digital Techniques*, vol. 2, pp. 386–400, 2008. [Online]. Available: http://dx.doi.org/10.1049/iet-cdt:20070124

[12] P. C. Haddow, A. M. Tyrrell, "Challenges of evolvable hardware: past, present and the path to a promising future", *Genetic Programming Evolvable Machines*, vol. 12, pp. 183–215, 2011. [Online]. Available: http://dx.doi.org/10.1007/s10710-011-9141-6