Image Filter Evolution on the Xilinx Zynq Platform

Roland Dobai and Lukas Sekanina
Brno University of Technology, Faculty of Information Technology, IT4Innovations Centre of Excellence
Brno, Czech Republic
Email: {dobai, sekanina}@fit.vutbr.cz

***Abstract*—The limitations of reconfigurable chips have always raised barriers for evolvable hardware. Zynq-7000 all programmable system-on-chip, the recent innovation in the reconfigurable field offers new possibilities for bypassing once again these barriers. In this paper, evolvable hardware implementations are considered on this new Zynq platform. The possibilities of the platform are demonstrated by evolutionary design of switching image filters. The investigated implementations include virtual reconfigurable circuits and the use of dynamic partial reconfiguration. The achieved results demonstrate the advantages and disadvantages of the Zynq platform. The observations are intended to be useful for designers who are going to develop evolvable hardware on this new platform.**

I. Introduction

Evolvable hardware (EHW) is a collective term which usually refers to either evolutionary hardware design or adaptive hardware. Evolutionary hardware design is based on the development of hardware by the use of evolutionary algorithms (EAs) or other bio-inspired methods. The goal of adaptive hardware is self-adaptation to changing environmental conditions or self-repair in the presence of faults [1], [2].

EHW utilizes many types of reconfigurable devices including field programmable analog arrays, reconfigurable antennas, mirrors, special reconfigurable materials but field programmable gate arrays (FPGAs) are the most popular ones [1]. The creativity of EHW designers has always been limited by the possibilities offered by the available FPGAs. The XC6200 family of FPGAs was an almost ideal platform for EHW since the device configuration could be directly generated by an EA. The first experiments with this device family [3] motivated many researchers. Later, newer FPGAs had insufficient support for reconfiguration which gave inspiration to introduce virtual reconfigurable circuits (VRCs) for EHW. VRCs subsequently proved to be very successful in many application domains [4]–[6].

Recent Virtex FPGAs have better support for reconfiguration. Their features include internal configuration access port (ICAP) which allows the FPGA to reconfigure itself by an on-chip hard or soft processor. This progress made available to develop EHW using native reconfiguration of FPGAs [7]–[9]. Mainly Xilinx FPGAs are used for EHW [1] therefore, FPGAs from other manufacturers are not studied in this paper.

Zynq-7000 all programmable (AP) system-on-chip (SoC) introduced in 2011 by Xilinx is a reconfigurable SoC. It integrates programmable logic (PL) compatible with other 7-series FPGAs and a processing system (PS) with a dual-core ARM processor [10]. It is an interesting and potentially beneficial platform for EHW because among others, the processor is very powerful with various operational modes and can exploit the attached PL.

In this paper, EHW implementations are considered on the new Zynq platform. The possibilities of the platform are demonstrated by evolutionary design of switching image filters. The investigated implementations include VRCs and the use of dynamic partial reconfiguration. The achieved results demonstrate the advantages and disadvantages of the Zynq platform. The observations are intended to be useful for designers who are going to develop EHW on this new platform.

The rest of the paper is organized as follows. Section II contains the related work. Section III introduces the Zynq-7000 AP SoC platform and evaluates its feasibility for EHW. FPGA-based acceleration of switching image filter evolution is discussed in Section IV. Section V discusses the achieved results and Section VI provides final conclusions.

II. FPGA-Based Evolvable Hardware

FPGA-based EHW uses an EA to generate candidate chromosomes. The chromosome represents the candidate circuit by specifying the function and interconnections of configurable resources in the FPGA. (1) The candidate circuit is configured/assembled and (2) evaluated based on a fitness function. (3) This evaluation is repeated for all candidate circuits in the given population. The evaluation can be executed sequentially or in parallel. (4) New populations are assembled by the use of bio-inspired operators (e.g. crossover, mutation) until an acceptable solution is found or some limit is exceeded (e.g. the number of generations, the time of evolution).

The format of the configuration bit stream of the given FPGA determines how the candidate circuit can be downloaded into the FPGA. If the format is open (i.e. the format is well documented and the setup of the configuration resources of the FPGA is known exactly) then the chromosome can be directly used as the configuration bit stream. The XC6200 family is an example of this class of FPGAs.

If the format of the bit stream is not documented well enough in detail then the EA needs to use an additional software from the FPGA manufacturer to transform the chromosome into the configuration bit stream. The use of this software is necessary in order to ensure that the configuration bit stream represents a valid FPGA configuration which will not damage the FPGA. The chromosome can define in the case of Virtex FPGAs the set of pre-generated partial bit streams.

These selected bit streams can be downloaded into the FPGA by ICAP which establishes the candidate circuit inside the FPGA. This process is called dynamic partial reconfiguration (DPR).

The approach based on VRC is platform independent because VRC can be implemented in an FPGA of any kind. It uses multiplexers to select the specified functionality of the candidate circuit. Here, the reconfiguration means just writing the configuration bit stream (i.e. the chromosome) into a set of registers (the configuration sets the multiplexers which reconfigures the circuit to the desired function).

The EA can be implemented either outside the FPGA in a personal computer [3], [6], [11] or inside the FPGA. The second option is currently the preferred solution [1] because the EA is a software and the on-chip processor can execute it very effectively. Here the reconfiguration is performed by either VRC [12]–[14] or DPR [7]–[9]. Another approach is to implement the EA as a specialized circuit by programmable resources of the FPGA [15]–[17].

EHW systems based on DPR, internal reconfiguration and EA executed by on-chip processors can nowadays be considered as the state-of-the-art [1]. They prosper from fast DPR and EA optimized for the given application. The main disadvantage of this DPR-based approach in comparison with VRC is the slower reconfiguration time. On the other hand, DPR does not need the additional multiplexers required by VRC and therefore the operational frequency is higher and the candidate evaluation faster [18].

The Zynq-7000 AP SoC was recently analyzed as a possible platform for EHW [19]. Image filter evolution was considered as the case study. VRC- and DPR-based hardware accelerations were evaluated and the possible speed-up of evolution was estimated. In this paper, VRC is also considered but furthermore, a new hybrid VRC-DPR approach is proposed. Previous DPR-based EHWs use hard-wired interconnections which is a certain limitation in comparison with VRC. The proposed hybrid approach has computationally an equal power to the pure VRC (because there is no limitation in comparison with VRC) what makes it a good choice for accurate comparison. Previously only estimations were published concerning the possible magnitude of hardware acceleration on the Zynq platform. Here, the results of a successful hardware implementation are evaluated. The achieved results demonstrate that the previous estimations were too optimistic and the DPR is not so advantageous in comparison with VRC. Furthermore, the control unit represents a serious limitation for both approaches. One can assume that this is a Zynq-specific (or 7-series Xilinx PL) problem since previous publications did not indicate that the propagation paths inside the control unit limit the operational frequency of the EHW system (previously the propagation paths inside the functional part were the only concern). The contribution of the paper can be summarized as follows:

1) Evolutionary design of switching image filters is implemented on the Zynq-7000 AP SoC platform.
2) The hybrid VRC-DPR approach is proposed which has an equal computational power to the pure VRC-based approach.
3) The limitation represented by the control unit is identified and the possible solutions are discussed in detail.

III. Xilinx Zynq-7000 AP SoC Platform for Evolvable Hardware

Zynq-7000 AP SoC was introduced recently by Xilinx. It has an ARM-based PS and a 7-series Xilinx PL fabricated with 28 nm technology. The hard on-chip processor used in the PS is a dual-core ARM Cortex-A9. The processor is equipped with instruction and data caches, on-chip read-only memory (ROM) and random-access memory (RAM), external memory interfaces, direct memory access (DMA) controller and input-output peripherals. The PL is based on an FPGA system similar to Artix-7 and Kintex-7 with the usual programmable resources: configurable logic blocks (CLBs), block RAMs, digital signal processing blocks. Among the most interesting features is the analog-to-digital converter (ADC) which gives the opportunity to get feedback from the real-life “analog world” [10].

Zynq-7000 AP SoC is conceptually different from the members of previous FPGA families. It has a PS-centric architecture with the platform built around the processor. The architecture of previous FPGAs was PL-centric with the on-chip processor used only as an extension. The PS of Zynq boots always first and the PL configuration is performed only later. As a matter of fact, the PL cannot be powered on until the PS is configured and operational [10].

The dual-core ARM processor can function in several operating modes. (1) It is possible that only one of the cores is functional and the other one is turned off. (2) Or otherwise, both cores can be turned on. In this case they can work in a symmetric or asymmetric way. Symmetric operational mode exploits both cores in the service of a single operating system (OS), or each core has its own OS in the asymmetric mode.

Configuration by ICAP requires the implementation of a special core in the PL. This means that the PL has to be configured before the processor can perform reconfiguration. ICAP is supported also by the Zynq platform but a new feature called processor configuration access port (PCAP) has been introduced. PCAP in comparison with ICAP does not need the implementation of any special core what makes the reconfiguration be more consistent with the new unique architecture. The PS can work with the PL turned off because they have different power supplies. Later when it is required, the PS can configure the PL. The supported download speed of reconfiguration is 400 megabytes per second (in non-secure configuration mode) [20].

The configuration bit stream contains instructions for synchronization and configuration, and also data for configuration. The bit stream is downloaded into the PL by DMA transfer what allows the PS to be free and perform some other task during the download [21], [22].

The ARM processor in the center of the architecture and other unique features make Zynq an interesting and potentially beneficial platform for EHW. The ADC allows measuring environmental conditions or getting feedback from the system-under-control, and subsequently Zynq can adapt itself to the changed environment. It can also host the user application natively in the PS and can use the PL only if it is necessary. The user application can be executed by a high-level OS with full support for the peripherals and the evolution can be performed in the PL. The EA can run either on the same OS as the user application or on a different OS (and the other processor core).

The PL of Zynq consists of several configuration frames similarly to previous FPGA families. The configuration operation must work with whole frames because they are the smallest addressable parts of the device configuration memory space [21]. DPR can rewrite the content of these frames but the reconfiguration of partial frames is not possible. These frames compose larger frames which will be the only frames considered in the rest of the paper because the current recommended partial reconfiguration flow generates bit streams for these frames only. This kind of frames contains 50 CLB in the case of Zynq-7000 AP SoC and is organized in 50 CLB high and 1 CLB wide rows [22]. Similar frames are defined for the other programmable resources. These frames are much larger than the frames of previous FPGA families (Virtex-6 had 40 CLBs and Virtex-5 only 20 CLBs) [22]. This can be considered as a disadvantage for EHW because the EA usually changes only small portion of the chromosome but always 50 CLBs are needed to be reconfigured no matter how small the change in the chromosome is. This means that Zynq will require more than twice the time for reconfiguration in comparison with a Virtex-5 FPGA because the frames are more than twice larger (the speed of reconfiguration has not been improved in the new platform).

IV. Case Study: Evolution of Switching Image Filters

The possibilities offered by EHW on the Zynq-7000 AP SoC platform is demonstrated by the case study of switching image filter evolution [23]. An example of switching filter is shown in Figure 1 where $i0, \dots, i8$ are filter inputs, y is the filter output, PE00, …, PE22 are processing elements (PEs), f is the filtered pixel, s is the "switch" between the filtered pixel and the current pixel $i4$. The filter shown in Figure 1 is unconnected. One of the tasks during filter evolution is to interconnect these unconnected PEs. Another task is to select the appropriate functions for these PEs (these functions are selected usually from a pre-defined set of arithmetic and logic operations). The main goal of the evolution is to accomplish these two tasks in such a way that the resulting candidate filter will be good for image filtering (the "goodness" is measured by the fitness and is described later). The image filter in the case study is able to filter 8-bit grayscale images, therefore, all the wires in Figure 1 have 8 bits. The filter tries to correct the potential damage in the current pixel denoted as i4. This is performed by exploiting the neighbors. The filter considered here uses a kernel of 3 $\times$ 3 which means that there are 9 filter inputs: the current pixel and the 8 neighbors. The PE array computes the filtered pixel f and the filter switch s. The filter switch will choose (by means of a multiplexer shown as MUX in Figure 1) between the filtered pixel and the current pixel. This will allow detecting if the current pixel has not been corrupted. Consequently, the current pixel will be passed to the filter output and the filtered image will have better quality [23]. Otherwise, the filtered pixel will become the filter output. It should be noticed that the filter processes only one pixel at the time; the pixels of the image are filtered sequentially.

The PE inputs can be connected to a filter input or to a PE output from the previous l columns, where l is the level-back parameter. If $l = 1$ then only neighbor PE columns can be interconnected.

The chromosome encodes the interconnections and the selected PE functions. This means that each PE is represented in the chromosome by 3 numbers: 2 identifiers for the 2 input connections and 1 identifier for the function. The filtered pixel and the filter switch each have 1 connection-identifier in the chromosome. This gives 3 columns $\times$ 3 rows $\times$ 3 + 1 + 1 = 29 numbers in the chromosome for the example in Figure 1.

The filter evolution is performed by the EA and the search is directed toward better filters. The search is guided by the fitness function which determines how good the current filter candidate is. The fitness function used in the case study is the following one:

$$\sum_{i=1}^{c}\sum_{j=1}^{r}\left|y(i,j) - r(i,j)\right|$$

where c is the number of image columns, r the number of image rows, $y(i,j)$ the filter output and $r(i,j)$ the correct (reference) image pixel. This function measures the difference between the filtered and correct images. A smaller value indicates a better filter than a greater value. The value 0 represents an ideal filter which is able to remove all the noises and restore the image to its original (correct) form.

In the rest of the paper, several approaches are considered of how to implement the evolution of image filters on the Zynq-7000 AP SoC platform. Sequential candidate evaluation will be considered in all of the approaches.

A. Software-Based Filter Evolution

The evolution consists of the establishing and evaluation of the candidates. Establishing a candidate means copying the chromosome of the parent and changing (mutate) a part of it. The mutation requires the generation of a few pseudo-random numbers. The time spent for establishing the candidates is negligible since the chromosome is relatively short and only a small number of mutations are performed.

Most of the time during the evolution is spent by evaluating the candidates. Here, several arithmetic and logic operations are executed for all the image pixels. For example, all these operations are executed 126 $\times$ 126 = 15 876 times for a training image of size 128 $\times$ 128 (the border pixels are not considered because they do not have 8 neighbors).

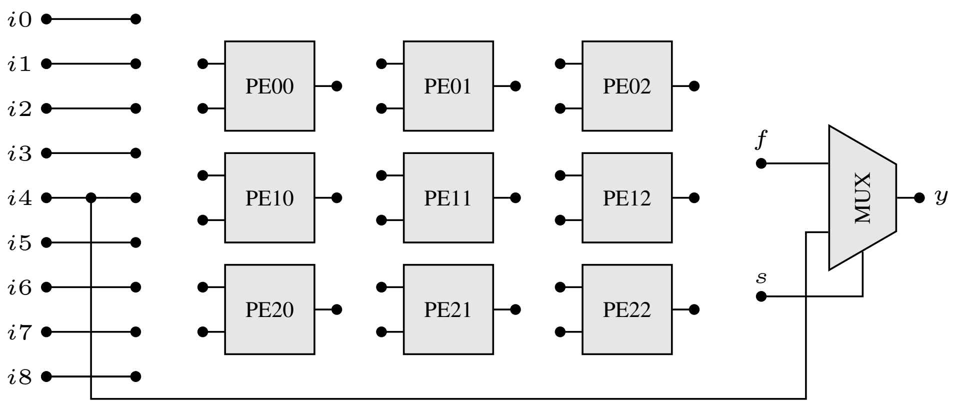


Figure 1. Switching image filter with an array of PEs

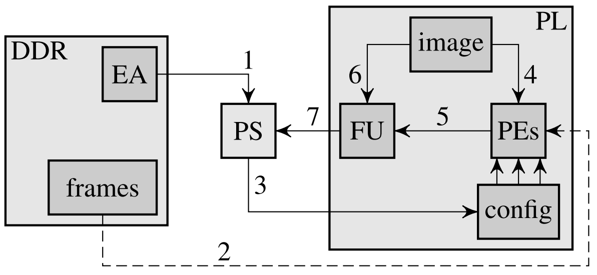


Figure 2. Hardware acceleration of image filtering

The following approaches will consider the evaluation of candidates by FPGA since the evaluation is the most time consuming part of the filter evolution. The ARM processor of the Zynq platform will be employed for establishing the candidates because this can be executed by software very effectively.

B. FPGA-Based Acceleration

The hybrid VRC-DPR approach is proposed to accelerate the evaluation of candidate circuits. The EHW system is shown in Figure 2 and the filter evolution is performed according to the following steps.

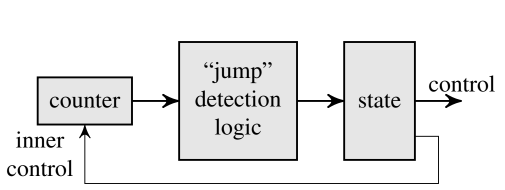
1) The EA stored in external memory (shown as DDR) is executed by the PS. The EA generates a chromosome (which represents a candidate filter).
2) The PEs are reconfigured in a way which corresponds to the chromosome. The reconfiguration is performed by DPR which uses configuration bit streams for frames. These frames are stored in external memory (DDR).
3) The interconnections of PEs are set by the config register (corresponding part of the chromosome).
4) The training (corrupted) image is transferred kernel-by-kernel to the PE array.
5) The filter output is transferred to the fitness unit (FU) in order to include it into the fitness.
6) The FU computes the absolute difference of the current (correct) pixel and the filter output. The result is accumulated.
7) The accumulated result becomes the fitness value after the whole image has been processed. The fitness is transferred to the EA. The evaluation of the next candidate follows by repeating these steps.

The PE array is pipelined in order to achieve higher operational frequency. Each PE column corresponds to one stage in the pipeline. Furthermore, each PE input is connected to a multiplexer which selects the desired interconnection based on the chromosome (config register). This means that the propagation paths of these pipeline stages are determined by the implementation of PE functions prolonged by a multiplexer. The longest among these paths will be the critical path which will specify the maximal operational frequency of the PE pipeline.

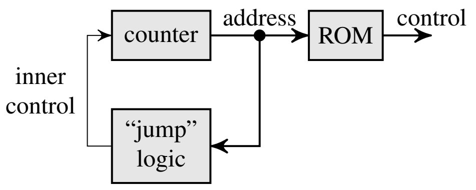
The principle of the pure VRC-based approach is very similar: the only difference is that the PE functions are set also by the config register and not by DPR. This means that this register contains the whole chromosome. One of the main implications is that the reconfiguration is much faster because writing the chromosome into the config register is very fast in contrast to the replacement of frames by DPR. However, all functions need to be implemented in all PEs. Firstly, this will cause higher implementation area (more programmable resources). Secondly, an additional multiplexer is placed at each PE output (the multiplexer selects the PE function based on the chromosome stored in the config register). This implies that the propagation paths of the pipeline stages will be determined by the implementation of PE functions prolonged by two multiplexers. As the result, the operational frequency of the PE array will be lower in comparison with the hybrid VRC-DPR approach since the propagation paths are longer by the implementation of one additional multiplexer.

C. Implementation of the Control Unit

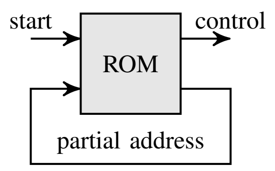
Previous research indicates that the operational frequency of FPGA-based EHW systems depends exclusively on the propagation paths of the PE array. It was revealed after our implementation on the Zynq-7000 AP SoC platform that the control unit has also serious influence.



(a) Control words based on states



(b) Control words in ROM



(c) Whole "program" in ROM

Figure 3. Control unit implementations

The first control unit implementation shown in Figure 3a is a standard, finite state machine (FSM) style. The FSM contains only a few states. Additional counters are used to determine that during how many clock cycles will the FSM remain in the given state. The counter and the additional "jump" detection logic create longer propagation paths than those in the PE array and therefore, the control unit will influence the operational frequency of the filter evolution. This implies that the hybrid VRC-DPR approach will not be able to work on higher frequency than the pure VRC-based approach.

The second considered implementation is shown in Figure 3b. The goal was to reduce the paths by implementing the control unit partially by ROM. In this case the "program", i.e. the control words containing the control signals, is stored in the ROM and the address is generated by a counter. There is some additional "jump" logic which implements jumps in the program, i.e. if a given address is detected then it changes the value of the counter. There were several program alignment considered with this implementation style (it has influence on the jump logic). However, even the best one resulted in an even lower frequency than the previous low-area implementation style.

The third implementation style was a full ROM control unit shown in Figure 3c. The ROM in this implementation stores the whole program, i.e. the control words and the next address which easily implements the jump functionality and additional logic is not necessary. The stored address is only a partial address which is joined with the start input to form the actual, full address. The start input signalizes that the candidate is ready for evaluation. This implementation style proved to be the most successful one. It allows high enough operational frequency that the critical path moves into the PE array and both VRC and VRC-DPR gain in the speed of evolution. The disadvantage of the full ROM implementation style is high area overhead which is challenging for some smaller Zynq devices as the ROM is implemented by block RAMs.

V. Experimental Results

The described approaches were implemented in order to demonstrate the evolutionary hardware design on the new Zynq-7000 AP SoC platform. The device used in the experiments was an XC7Z020-1CLG484CES. The parameters used for evolutionary design of switching image filters were as follows: (1+4) evolutionary strategy, pipelined 4 rows and 8 columns of PEs, 16 operations of PEs shown in Table I, kernel size 3 $\times$ 3, Lena benchmark image of size 128 $\times$ 128 and 256 $\times$ 256, 5% salt and pepper noise, level-back 1. These parameters were selected as the most common ones in the field [1] and the experiments were not aimed at their optimization.

Table I
8-BIT OPERATIONS OVER OPERANDS x, y IMPLEMENTED BY PEs

Code	Operation	Description
0	255	constant
1	x	identity
2	$255 - x$	inversion
3	$x \vee y$	bitwise OR
4	$\overline{x} \vee y$	bitwise $\overline{x}$ OR y
5	$x \wedge y$	bitwise AND
6	$\overline{x \wedge y}$	bitwise NAND
7	$x \oplus y$	bitwise XOR
8	$x \gg 1$	right shift by 1
9	$x \gg 2$	right shift by 2
10	$swap(x, y)$	swap nibbles
11	$x + y$	addition
12	$x +^s y$	addition with saturation
13	$(x + y) \gg 1$	average
14	$max(x, y)$	maximum
15	$min(x, y)$	minimum

A. Time Required for Filter Evolution

VRC, the most popular approach and the proposed hybrid VRC-DPR approach were implemented with the main goal to evaluate the evolution time on the Zynq platform. The achieved results are summarized in Table II where "PS (ARM)" denotes the results achieved by the on-chip ARM processor (667 MHz) without the PL, "desk" the results of a desktop computer with an Intel Core i5 661 3.33 GHz processor, VRC the results of the ARM processor with the VRC-based approach, and VRC-DPR the results of the ARM processor with the hybrid VRC-DPR approach. The results of the hybrid approach are shown for various number of PE mutations because this parameter has an influence on the reconfiguration time. The mutation of interconnections has similarly negligible influence on the evaluation time just like for the other approaches. The total number of mutations considered for the other approaches was 7 (approximately 2 PE mutations are equivalent to 7 mutations considering the same mutation ratio). Table II shows the time required to evaluate one filter candidate (individual), the time for a generation of filter candidates (4 individuals), the

Table II
TIME OF FILTER EVOLUTION

	PE mutations	Image 256 × 256				Image 128 × 128			
		Individual	Generation	Generations	Acceleration	Individual	Generation	Generations	Acceleration
	(1)	(µs)	(µs)	(s^{-1})	(1)	(µs)	(µs)	(s^{-1})	(1)
PS (ARM)		225 285.3	901 141.1	1.1	1	55 055.7	220 222.9	4.5	1
desk		42 372.9	169 491.5	5.9	5	10 081.7	40 326.7	24.8	6
VRC		316.8	1267.4	789	717	78	312	3206.5	713
VRC-DPR	1	284.2	1136.6	879.8	800	100.8	403.3	2479.3	551
VRC-DPR	2	325.2	1300.6	768.9	699	141.8	567.3	1762.6	392
VRC-DPR	3	366.2	1464.6	682.8	621	182.8	731.3	1367.3	304
VRC-DPR	4	407.2	1628.6	614	558	223.8	895.3	1116.9	248
VRC-DPR	5	448.2	1792.6	557.8	507	264.8	1059.3	944	210
VRC-DPR	6	489.2	1956.6	511.1	465	305.8	1223.3	817.4	182
VRC-DPR	7	530.2	2120.6	471.6	429	346.8	1387.3	720.8	160
VRC-DPR	8	571.2	2284.6	437.7	398	387.8	1551.3	644.6	143

number of generation per second and the relative acceleration in comparison with the ARM processor (without the PL).

First, the pure software-based approach was evaluated on the ARM processor of the available Zynq device. 10 runs and 100 generations were considered and the number of clock cycles was measured. Table II contains the average results after the cycles were converted to seconds. The observed 1.1 and 4.5 generations per second were used later to determine the relative acceleration for images 256 × 256 and 128 × 128, respectively.

The second experiment was aimed to compare the processor of Zynq with a desktop processor (Intel i5). The code of the EA developed in language C was pre-ported to that processor (standard output interface and time measurement were changed). Here, 10 measurements and 50 000 generations were considered and the averages of the achieved results are denoted as "desk" in Table II. According to this experiment, the Intel i5 processor was 5 and 6 times faster than the ARM processor of the Zynq device.

It can be noticed that the filter evolution takes almost exactly 4 times longer for the larger training image (the larger image has 4 times more pixels than the smaller one). This can be considered as a proof that the evaluation of candidate filters represents almost the full time of filter evolution (the other parameters of the experiments were constants).

The goal of the third experiment was to determine the magnitude of the FPGA-based acceleration of the filter evolution. The implementation revealed that the operational frequency of the pure VRC and hybrid VRC-DPR approach is 203.6 MHz and 265.3 MHz, respectively. This means that the hybrid approach is able to evaluate the candidate filters approximately by 30% faster. On the other hand, the VRC approach mutates the circuit in negligible time and the hybrid approach requires more time. The hybrid approach changes the interconnections similarly to the pure VRC approach but the replacement of the PEs by DPR takes longer. According to the experiments, the replacement of a PE requires 41 µs through PCAP (for a 100 MHz ICAP this would take 39 µs).

Table III
FPGA UTILIZATION FOR IMAGE 128 × 128

	Available	VRC		VRC-DPR without PEs	
Flip-flops	106 400	3492	3.3 %	644	0.6 %
LUTs	53 200	6340	11.9 %	2325	4.4 %
Logic	53 200	6276	11.8 %	2261	4.3 %
Shift	17 400	64	0.4 %	64	0.4 %
Block RAM	140	58	41.4 %	58	41.4 %

As the results demonstrate in Table II, the FPGA-based approaches achieve several hundred times faster filter evolution than the pure ARM-based approach. Even the improvement by the Intel i5 processor is negligible compared to the FPGA-based approaches. The hybrid VRC-DPR was faster with comparison to the VRC only for the larger image and 1 PE mutation. In this case the speed of candidate evaluation is more dominant than the reconfiguration time. However, with higher number of PE mutations and/or smaller image the reconfiguration time becomes significant and the hybrid method cannot be faster than the pure VRC approach. It is possible to address this issue by using a pure DPR-based approach where all of the multiplexers are removed from the PE array. This would result in a higher operational frequency and faster candidate evaluation. However, the removal of the rest of the multiplexers would introduce a certain limitation and the comparison with VRC would be less objective.

B. Required FPGA Resources

The FPGA resources required to implement the VRC and VRC-DPR approaches are shown in Table III. The utilization of shift-register look up tables (LUTs) and block RAMs are the same for these approaches. The difference is in the number of LUTs used as logic and the number of flip-flops. The VRC-based approach requires more programmable resources because each function from Table I is implemented in each PE all the time. On the other hand, DPR replaces these PEs according to the chromosome content. It should be noted that

Table IV
OPERATIONAL FREQUENCIES

Control unit	VRC	VRC-DPR
Low overhead	198.2 MHz	198.2 MHz
Partial ROM	128.2 MHz	128.2 MHz
Full ROM	203.6 MHz	265.3 MHz

the results for the VRC-DPR approach do not contain the PEs (the implementation of PEs is implied by the chromosome content; the PEs are downloaded into the FPGA after the establishment of the chromosome). For these PEs there are frames reserved in the PL: 8 columns $\times$ 4 rows $\times$ 50 CLBs = 1 600 CLBs = 12 800 LUTs + 25 600 flip-flops. The reserved space is relatively large but the PEs occupy only a small fragment of it. Usually it would be allowed for reconfigurable and non-reconfigurable parts to share these reserved frames. However, this is not possible here because the PEs need to be relocated in order to limit the number of configuration bit streams. Relocation means that only one bit stream is prepared for a given PE function which is used to reconfigure all PEs to this function (for 16 functions only 16 bit streams are required). Otherwise, a bit stream should be prepared for each function in each PE location which would require a considerable memory space to store (8 columns $\times$ 4 rows $\times$ 16 functions = 512 bit streams).

C. *Influence of the Control Unit on the Operational Frequency*

So far the results considered the full ROM implementation style of the control unit (shown in Figure 3c). 50 block RAMs out of 58 shown in Table III are occupied by this unit. The initial part of the "program" takes 259 clock cycles (initial fill-up of the pipeline) and after that the evaluation of a candidate filter takes 16 388 cycles. The full program has a length of 33 794 cycles. Each word of this program has 25 bits (10 control bits and 15 address bits). One additional address bit (start bit in Figure 3c) together with the 15 bits stored in ROM defines the address space of length 65 536. The synthesis tool from the manufacturer was not able to generate the required control unit. A special software developed in C was used to produce the content of this ROM.

Image 256 $\times$ 256 implies approximately 4 times more states in the control unit and longer program with wider program words. Such program requires more than the available amount of block RAM resources. A possible solution is to choose another Zynq device with more block RAM resources [10].

Operational frequencies for various control unit implementations are shown in Table IV in order to demonstrate the possible effects and implications. If the full ROM implementation of the control unit does not fit into the device then the low-overhead style is used (because the operational frequency will be still better than that of the partial ROM implementation). In this case the critical path influencing the maximum operational frequency moves from the PE array into the control unit. Therefore, the control unit will influence the "speed" of the circuit and both implementation will be able to operate on 198.2 MHz. This means that the main advantage of DPR (faster candidate evaluation) is eliminated and the VRC approach becomes definitely the preferable one (because the reconfiguration by VRC is faster).

It can be observed that the pure VRC-based approach does not gain much by the full ROM implementation style. Probably this is the reason why the published methods concerning VRC does not deal with the implementation style of the control unit. However, as it was shown by the experiments, the approach using DPR requires the effective control unit implementation in order to be able to operate on higher operational frequency.

VI. CONCLUSIONS

Zynq-7000 AP SoC is a recent innovation in the reconfigurable field which offers new possibilities for EHW designers. It has useful features (e.g. asymmetric multiprocessing, ADC or PCAP) which make it an interesting and potentially beneficial platform for EHW.

In this paper, the Zynq platform was evaluated based on the usual requirements for evolutionary hardware design. The possibilities of the platform were demonstrated by evolutionary design of switching image filters. The experimental results and the observations are intended to be useful for designers who are going to develop EHW on the new Zynq platform.

VRC, the most popular approach and a hybrid VRC-DPR approach were considered. The hybrid approach represent computationally equal power to the pure VRC what makes it a good choice for accurate comparison.

DPR offers higher operational frequencies by shortening the propagation paths in the circuit. This is achieved by the elimination of some of the multiplexers and also results in lower area overhead if the reserved large reconfigurable frames are not taken into account. The high reconfiguration time of DPR caused by the coarse-grained reconfigurable frames remains an open problem for EHW.

As the experiments demonstrated, DPR can outperform the pure VRC-based approach only in special cases when the time required for candidate evaluation dominates the reconfiguration time. However, there is a catch. The candidate evaluation dominates the reconfiguration time only if there is a large amount of training data which consequently imply a control unit with many states. But such a control unit requires low-area implementation style which reduces the operational frequency of the circuit and eliminates the main advantage of the DPR-based approach. Obviously, a golden mean can be found.

EHW on the Zynq platform is promising in many aspects but raises many questions some of which were answered by this paper but others require further investigation. Future work will be conducted to exploit all the features and advantages of this platform for EHW design. The results provided in this paper are only preliminary. Some further changes in the proposed method are expected in order to overcome the limitations. Power consumption will be investigated and the quality of the evolved filters will be addressed.

Acknowledgments

This work was supported by The European Social Fund (ESF) under the project Excellent Young Researchers at BUT (CZ.1.07/2.3.00/30.0039) and the IT4Innovations Centre of Excellence (CZ.1.05/1.1.00/02.0070).

References

[1] L. Sekanina, "Evolvable hardware," in *Handbook of Natural Computing*. Springer Verlag, 2012, pp. 1657–1705.

[2] P. C. Haddow and A. M. Tyrrell, "Challenges of evolvable hardware: past, present and the path to a promising future," *Genetic Programming and Evolvable Machines*, vol. 12, no. 3, pp. 183–215, 2011.

[3] A. Thompson, "Silicon evolution," in *Genetic Programming 1996: Proc. 1st Annual Conf. (GP96)*, J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, Eds. Cambridge, MA: MIT Press, 1996, pp. 444–452.

[4] L. Sekanina, "Virtual reconfigurable circuits for real-world applications of evolvable hardware," in *Evolvable Systems: From Biology to Hardware*, ser. LNCS, no. 2606. Springer Verlag, 2003, pp. 186–197.

[5] D. Gwaltney and K. Dutton, "A VHDL Core for Intrinsic Evolution of Discrete Time Filters with Signal Feedback," in *Proc. of the 2005 NASA/DoD Conference on Evolvable Hardware*. Washington D.C., USA: IEEE Computer Society, 2005, pp. 43–50.

[6] G. Hollingworth, S. L. Smith, and A. M. Tyrrell, "Safe intrinsic evolution of virtex devices," in *The Second NASA/DoD Workshop on Evolvable Hardware*. IEEE Computer Society, 2000, pp. 195–202.

[7] R. Salvador, A. Otero, J. Mora, E. de la Torre, T. Riesgo, and L. Sekanina, "Evolvable 2D computing matrix model for intrinsic evolution in commercial FPGAs with native reconfiguration support," in *Proceedings of the 2011 NASA/ESA Conference on Adaptive Hardware and Systems*. IEEE Computer Society, 2011, pp. 184–191.

[8] A. Upegui and E. Sanchez, "Evolving hardware with self-reconfigurable connectivity in Xilinx FPGAs," in *The 1st NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2006)*. Los Alamitos, CA, USA: IEEE Computer Society, 2006, pp. 153–160.

[9] F. Cancare, M. D. Santambrogio, and D. Sciuto, "A direct bitstream manipulation approach for virtex4-based evolvable systems," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2010, pp. 853–856.

[10] "Zynq-7000 All Programmable SoC Overview DS190 (v1.2)," Xilinx, 2012.

[11] L. Huelsbergen, E. Rietman, and R. Slous, "Evolving oscillators in silico," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 3, pp. 197–204, 1999.

[12] Z. Vasicek and L. Sekanina, "An evolvable hardware system in Xilinx Virtex II Pro FPGA," *International Journal of Innovative Computing and Applications*, vol. 1, no. 1, pp. 63–73, 2007.

[13] ——, "Hardware accelerator of cartesian genetic programming with multiple fitness units," *Computing and Informatics*, vol. 29, no. 6, pp. 1359–1371, 2010.

[14] K. Glette, J. Torresen, M. Yasunaga, and Y. Yamaguchi, "On-Chip Evolution Using a Soft Processor Core Applied to Image Recognition," in *The 1st NASA/ESA Conference on Adaptive Hardware and Systems*. Los Alamitos, CA, USA: IEEE Computer Society, 2006, pp. 373–380.

[15] G. Tufte and P. C. Haddow, "Evolving an adaptive digital filter," in *2nd NASA/DoD Workshop on Evolvable Hardware (EH 2000)*. IEEE Computer Society, 2000, pp. 143–150.

[16] T. Martinek and L. Sekanina, "An evolvable image filter: Experimental evaluation of a complete hardware implementation in FPGA," in *Evolvable Systems: From Biology to Hardware*, ser. LNCS, vol. 3637. Springer Verlag, 2005, pp. 76–85.

[17] J. Wang, Q. S. Chen, and C. Lee, "Design and implementation of a virtual reconfigurable architecture for different applications of intrinsic evolvable hardware," *IET Computers and Digital Techniques*, vol. 2, no. 5, pp. 386–400, 2008.

[18] R. Salvador, A. Otero, J. Mora, E. de la Torre, T. Riesgo, and L. Sekanina, "Implementation techniques for evolvable HW systems: Virtual vs. dynamic reconfiguration," in *Proc. of the 22nd International Conference on Field Programmable Logic and Applications (FPL)*. IEEE Computer Society, 2012, pp. 547–550.

[19] R. Dobai and L. Sekanina, "Towards Evolvable Systems Based on the Xilinx Zynq Platform," in *2013 IEEE International Conference on Evolvable Systems (ICES)*, 2013, pp. 89–95.

[20] "Zynq-7000 All Programmable SoC technical reference manual UG585 (v1.3)," Xilinx, 2012.

[21] "7 Series FPGAs Configuration User Guide UG470 (v1.5)," Xilinx, 2012.

[22] "Partial Reconfiguration User Guide UG702 (v14.3)," Xilinx, 2012.

[23] Z. Vasicek, M. Bidlo, L. Sekanina, and K. Glette, "Evolutionary design of efficient and robust switching image filters," in *Proc. 2011 NASA/ESA Conference on Adaptive Hardware and Systems*, 2011, pp. 192–199.