Towards Evolvable Systems Based 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@fit.vutbr.cz, sekanina@fit.vutbr.cz

***Abstract*—Field programmable gate arrays (FPGAs) are considered as a good platform for digital evolvable hardware systems. Researchers introduced virtual reconfigurable circuits as the response to the insufficient support of partial reconfiguration in early FPGAs. Later, the features of FPGAs allowed the designers to develop evolvable systems fully exploiting native reconfiguration infrastructures. Xilinx recently introduced a new platform called Zynq-7000 all programmable (AP) system-on-chip (SoC) which has the potential to become the next revolutionary step in evolvable hardware design. The paper analyzes Zynq-7000 AP SoC from the perspective of an evolvable hardware designer. Several scenarios are described of how to implement evolvable systems on a developmental board equipped with this programmable SoC. These scenarios are evaluated in terms of area overhead, execution time, reconfiguration time and throughput. The resulting observations should be useful for those who are going to develop real-world evolvable systems on the Zynq-7000 AP SoC platform.**

I. Introduction

By *evolvable hardware* (EHW) we usually mean either evolutionary hardware design or adaptive hardware exploiting some of bio-inspired computing methods [1], [2]. While *evolutionary hardware design* is the use of bio-inspired algorithms for creating innovative physical designs, the goal of *adaptive hardware* is to endow physical systems with a capability of adaptation in order to allow them to operate successfully in a changing environment or under presence of faults.

Field Programmable Gate Arrays (FPGAs) have always been considered as an "almost perfect" platform for digital evolvable hardware systems. The first experiments with evolutionary circuit design carried out by Adrian Thompson directly on the XC6216 FPGA in 1996 [3] motivated many researchers to work in the evolvable hardware field. After some disillusion from taking the XC6200 family back from the market in late '90s and insufficient support of the partial reconfiguration in early Virtex chips, researchers have introduced *virtual reconfigurable circuits* (VRCs) for evolvable hardware and proposed new FPGA-based evolvable systems in several application domains [4]–[6]. Recent Virtex FPGAs (Virtex-4, -5 and -6) have provided a reasonable support for dynamic partial reconfiguration which included a suitable granularity of reconfiguration, a fast internal configuration access port (ICAP) and either hard or soft on-chip processors that can be utilized to control the reconfiguration. These features allowed the designers to develop evolvable systems fully exploiting the native reconfiguration infrastructure of FPGAs [7]–[9]. It has to be noticed that Xilinx's FPGAs have predominantly been used for evolvable hardware and hence we do not deal with FPGAs of other vendors in this paper.

In 2011, Xilinx introduced a new reconfigurable system-on-chip (SoC) called *Zynq-7000 all programmable (AP) SoC* which integrates programmable logic, ARM based processing system and numerous subsystems [10]. It seems to be an ideal platform for evolvable hardware as it contains reconfigurable logic which can easily and quickly be reconfigured by the on-chip ARM processor.

The goal of this paper is to describe and analyze the Zynq-7000 AP SoC from the perspective of an evolvable hardware designer. We propose and investigate several scenarios of how to implement evolvable systems on a developmental board equipped with the Zynq-7000 AP SoC. These scenarios are evaluated in terms of area overhead, time of execution, reconfiguration time and throughput. The resulting observations should be useful for those who are going to develop real-world evolvable systems on the Zynq-7000 AP SoC platform.

The rest of the paper is organized as follows. Section II contains the related work and Section III the introduction of the Zynq-7000 AP SoC platform. This new platform is evaluated in Section IV from the point of evolvable systems. Section V summarizes the achieved results and Section VI concludes the paper.

II. FPGA-based Evolvable Hardware

In FPGA-based evolvable hardware the evolutionary algorithm (EA) generates candidate chromosomes (configurations) that are used to configure chosen reconfigurable blocks of the FPGA. Once a new candidate circuit is established on the basis of the configuration, it is evaluated by means of a fitness function. The evaluation is performed for all candidate circuits in the population either sequentially or in parallel. New populations are created using bio-inspired operators such as crossover, mutation and selection. The process is repeated until a required solution is obtained or a predefined number of generations is met.

The reconfiguration can directly be performed at the level of the configuration bit stream for some FPGA families. In case of the XC6200 FPGAs there are no constraints. In case of Virtex FPGAs, one has to ensure that the reconfiguration is safe. These implementations typically utilize the concept of *dynamic partial reconfiguration* (DPR) allowing designers

to modify only a part of the FPGA while other parts of the FPGA can perform computing unaffected. There is the so-called ICAP in the Virtex FPGAs which enables one to accomplish the partial reconfiguration from a device (e.g. the MicroBlaze processor) located inside the FPGA.

A different approach is to reconfigure a VRC which is built on the top of the FPGA using multiplexers and application-specific processing elements. Here, the reconfiguration means just writing a set of registers. Virtual reconfigurable circuits have been developed in order to avoid slow and not-well-supported reconfiguration mechanisms existing in former FPGAs. A compromise between the DPR and VRC is the approach proposed by Glette et al. which exploits the shift behavior of look up tables (LUTs) to change its logic [11]. In this case, the reconfiguration process is carried out by directly shifting the configuration bits into the LUTs.

The EA is implemented either outside the FPGA (e.g. in a personal computer) or inside the FPGA. The second option is currently a preferred solution as the EA is, in fact, a software which can be executed in on-chip processors such as MicroBlaze and PowerPC. Another approach is to implement the EA as a specialized circuit using resources available in the FPGA.

The development of FPGA-based evolvable hardware systems was surveyed in [2]. The survey has identified the following application domains: image filtering, recognition and classification, infinite/finite impulse response filtering, oscillators/discriminators, adaptive logic circuits (e.g. hash functions) and cellular automata. The following list gives main classes of FPGA-based evolvable hardware systems according to the implementation strategy employed:

- Externally reconfigurable EHW systems [3], [4], [12].
- Internally reconfigurable EHW systems based on VRC and hardwired EA [13]–[15].
- Internally reconfigurable EHW systems based on VRC and EA in an on-chip processor [16]–[18].
- Internally reconfigurable EHW systems based on DPR and EA in an on-chip processor [7]–[9].

The last class can nowadays be considered as the state-of-the-art in this domain. It benefits from a relatively fast DPR of modern Virtex chips and EAs implemented as software which can easily be tuned for a particular application. In contrast to the VRC-based approach, the reconfiguration time is still slower. However, the circuits instantiated by means of DPR do not exhibit additional delay during normal operation because, unlike in VRCs, there are no multiplexers implementing the reconfiguration network [19].

III. Zynq-7000 AP SoC Platform

In 2011, Xilinx introduced the new reconfigurable SoC platform Zynq-7000 AP SoC. The platform consists of the powerful ARM processor based processing system (PS) and the 28 nm Xilinx programmable logic (PL). The dual-core ARM Cortex-A9 processor together with caches, on-chip memory, external memory interfaces, direct memory access (DMA) controller and input-output peripherals form the PS. The PL is equivalent to Artix-7 or Kintex-7 FPGAs consisting of configurable logic blocks (CLBs), block random-access memories, digital signal processing blocks, programmable input-output blocks, serial transceivers and analog-to-digital converters (ADCs). The PS frequency and the PL size is given by the selected Zynq-7000 AP SoC device. The maximum operational frequency of the PS is 667 MHz – 1 GHz; the PL contains 17 600 – 218 600 LUTs, 35 200 – 437 200 flip-flops, 240 – 2 180 kB block random-access memories [10]. The PS and the PL are on independent power supplies, with 1.0 V supply for the logic, 1.8 – 3.3 V for the input-output buffer bank and 1.2 – 1.8 V for the external dynamic memory interface [20]. The estimated power consumption for the smaller and larger devices is below 3 W and 15 W, respectively [10].

Previous FPGA families are in fact PLs with some optional on-chip processor extension (PL-centric architecture). On the other hand, Zynq-7000 AP SoC is an FPGA platform built around the processor (PS-centric architecture). The PS boots first and the PL part is configured only afterward. The PL cannot be powered on before the PS [10]. The dual-core PS can work in several operating configurations:

1) One core is operational and the second one is turned off using clock gating.
2) Both cores are operating. This multiprocessing cooperation can be
   - symmetric, when both cores are running the same operating system (OS) and participate in the same operations (e.g. multithread and multiprocess execution on a higher-level OS like Linux), or
   - asymmetric, when the cores are independent with different OSs (e.g. full featured OS and non-OS standalone bare-metal application).

Previous FPGA architectures allowed the on-chip processor to reconfigure the programmable part. This was facilitated by ICAP which needed the instantiation of hardware intellectual property core in the programmable part (the programmable part needed to be configured before the processor could perform further reconfiguration). Zynq-7000 AP SoC has a new feature called processor configuration access port (PCAP) which is part of the PS, and in contrary to ICAP, does not need any instantiation in the PL part. The PS and PL occupy different power planes, therefore, the PS can run with the PL powered off. The PS can boot up without the configuration of the PL, and can configure the PL through PCAP later only when, and only if it is required. The PCAP supports 400 megabytes per second download throughput for non-secure PL configuration [20].

The configuration bit stream which contains among others the configuration instructions and frame configuration data is downloaded from a memory location into the PL. The download is performed by DMA transfer, therefore the PS is free during the download. Partial reconfiguration is possible after a full configuration. This means that configuration data is downloaded only for some of the frames and the remaining

part of the FPGA not belonging to configured frames remains unchanged (and operational without any interruption) [21], [22].

IV. Evolvable Systems on the Zynq Platform

The PS-centric architecture and the features of Zynq-7000 AP SoC are conceptually very beneficial from the perspective of evolvable hardware design. Firstly, the evolvable system can use the ADC in order to get feedback from the environment, and consequently to adapt itself to it. Secondly, it can execute the user application natively without PL support, but still can use the PL in case it is necessary to accelerate the evolution of a new program/application. The user application can run on a full featured higher-level OS with support to all of the peripherals and can initiate the evolution of new functionalities (e.g. required by the user or as an adaptation request to the changed environmental conditions). The hardware-accelerated evolution can be controlled by an EA running either

1) on the other core as a bare-metal application (in order to eliminate the overhead caused by an OS), or
2) in symmetric operational mode on the same OS (in order to gain access to the higher-level OS functionalities, e.g. dual-core multiprocessing/multithreading or filesystem access).

A. Partial Reconfiguration

Virtex, Artix-7 and Kintex-7 FPGAs have the configuration memory arranged in configuration frames. These frames are the smallest addressable parts of the device configuration memory space. All configuration operations must work with whole frames, therefore the partial reconfiguration can influence one or more (but always whole) frames. The frames of Zynq-7000 AP SoC are 50 CLB high and 1 CLB wide [22]. Similar frames exist for other PL resources (e.g. block memory, digital processing blocks). For comparison, the height of frames in Virtex-6 was 40 CLBs and in Virtex-5 only 20 CLBs [22]. This increased frame size can be actually disadvantageous for evolvable hardware design if during the evolution only small changes are applied, but the EA is forced to reconfigure a full frame (i.e. 50 CLBs are reconfigured instead of a single CLB). Furthermore, a mutation will probably take more than twice as long as in the case of a Virtex-5 since the frames are more than twice larger (under the assumption that the reconfiguration speed is the same).

Figure 1 shows some example reconfigurable blocks (RBs) in a Zynq-7000 AP SoC device (XC7Z020) where the RBs are black rectangles numbered from 1 to 6, and the clock regions are white rectangles with designations from X0Y0 to X1Y2. The widths of RBs in Figure 1 are 1 CLB while the heights are various. RB1 occupies the height of the whole clock region X1Y0 which corresponds to exactly one configuration frame (i.e. 50 CLBs). RB2 is smaller than RB1 but still during the reconfiguration 50 CLBs will be reconfigured (similarly as if it would occupy the whole height of the clock region). However, RB2 can share the column with the static part of the implementation, while RB1 cannot (the static part is the part which is never reconfigured and its configuration remains the same). An RB cannot share the column with another RB in the same clock region (e.g. RB3 and RB4 in Figure 1) but can share in different clock regions (e.g. RB5 and RB6).

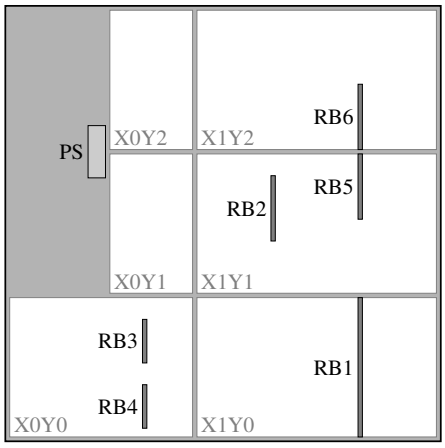

Figure 1. Example RBs in a Zynq-7000 AP SoC device

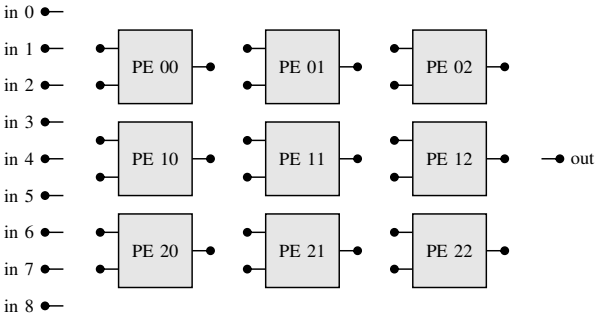

Figure 2. Array of PEs for image filtering

B. Case Study: Symbolic Regression

The feasibility of Zynq-7000 AP SoC for evolvable hardware design will be demonstrated by a typical case study scenario: symbolic regression by Cartesian genetic programming (CGP) for image filtering [14]. In this scenario, an image filter is designed which is a digital circuit able to suppress some errors in images. The structure of the circuit is not known in advance. Instead, a fixed-size two-dimensional array of processing elements (PEs) is used, and the "program" (solution) is constructed by interconnecting these PEs and implementing simple operations inside of these PEs. The operations for PEs are usually specified in advance and are a limited, small set of operations.

An example array of PEs is shown in Figure 2 where the PEs are indexed from 00 to 22; "in 0", ..., "in 8" are input image pixels; and "out" is the output (filtered) image pixel. There are nine inputs considered in Figure 2; one represents the pixel which should be filtered and the other eight are the immediate eight neighbor pixels (since image filters usually restore the pixel by considering also the neighbor pixels). The inputs, the output and the operands of PEs are 8-bit natural

numbers, and hence the resulting filter will be able to filter 8-bit grayscale images. The image is repaired by processing the pixels sequentially (together with the neighbor pixels). The fitness function measures the difference between the filtered image and original (uncorrupted) image, and will guide the EA toward better solutions. The fitness is computed as follows:

$$fitness = \sum_{i=0}^{c-1} \sum_{j=0}^{r-1} \left| p(i,j) - p_{orig}(i,j) \right|$$

where c is the number of columns, r the number of rows, $p(i,j)$ the filtered image pixel and $p_{orig}(i,j)$ the original image pixel. A PE input can be connected to a filter input or to a PE output in the direction to the filter inputs. The levels-back parameter of CGP sets the distance in columns these PEs allowed to be interconnected. For example, if the levels-back parameter is one then the PE inputs can be connected to PE outputs in the neighbor column only (and of course, to the filter inputs). The interconnection and the numerical identifiers of PE operations are encoded into the chromosome which unequivocally describes the image filter. A PE is encoded by three numbers (one for each input-connection and one for operation selection). The output is encoded by a number specifying the output connection. This gives a chromosome length $3 \times 3 \times 3 + 1 = 28$ for the array of 3×3 PEs in Figure 2.

Now several scenarios will be described in order to demonstrate various CGP-based implementations of image filter design in Zynq-7000 AP SoC. It is assumed in all of the scenarios that the candidate filters are evaluated sequentially.

1) Software-Based Symbolic Regression: The evolution consists of the spawn and the evaluation of the individual (image filter). The individual is spawned by copying and mutating the chromosome of the parent. This can be performed in a negligible short time since the chromosome is short and the mutation is just a matter of generating a small number of pseudo-random numbers. The prevalent part of the execution time is taken by the evaluation, i.e. the determination of the fitness, because the response of the circuits is measured/evaluated sequentially for all of the pixels. This means $(128-2)^2 = 15\ 876$ evaluations for an image of resolution 128×128 and would mean more than 2 million evaluations for an image with full high-definition resolution (note that there are not 128^2 evaluations because the pixels at the border of the image do not have neighbors in all directions, and therefore they are not evaluated).

From the point of execution time the mutations have negligible influence, and the level-back parameter no influence at all on the performance of the software-based implementation of symbolic regression (because the PEs are in random-access memory where the access is in constant time to all of the PEs).

Zynq-7000 AP SoC is a PS-centric FPGA with the on-chip processor always running, therefore there is no reason to consider to move out the implementation of EA from the PS. However, it is reasonable to consider PL-based hardware acceleration for evaluation since that is the most time consuming part of the software-based approach.

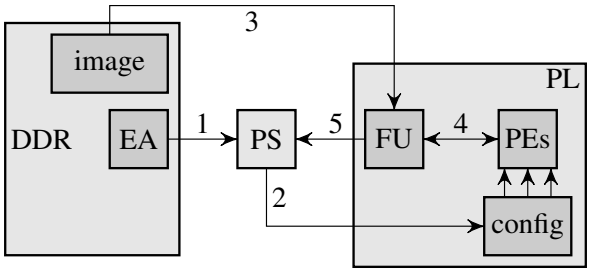

Figure 3. Acceleration of image filtering by VRC

2) Acceleration by VRC: The principle of the PL-based hardware acceleration by means of VRC is depicted in Figure 3 where DDR is external dynamic random access memory, FU is the fitness unit responsible for the computation of the fitness, and "config" is the configuration register of the VRC. As a matter of fact, the configuration register stores the chromosome representing the given image filter. The content of this register is decoded and used to configure the PEs: the interconnections are implemented by multiplexers and the PE operations are also selected by multiplexers. The evolution is performed as follows.

1) The EA is executed by the PS which controls the evolution. New individual (chromosome representing an image filter) is created by copying the parent chromosome and performing some mutations. The execution time of this step is negligible similarly to the pure software-based approach.
2) The chromosome is downloaded into the configuration register of the VRC. This sets the PE array.
3) The corrupted and the original image is transferred by the FU to the filter for evaluation.
4) The responses of the image filter are measured and evaluated by the FU. The fitness is computed.
5) The fitness is transferred back to the PS and the evaluation of another candidate image filter can continue from step 2.

The program of the EA and the images are stored in dynamic memory, but could be also in static memory, or even in internal on-chip memory. Moreover, the images could be stored also in block memories inside the PL.

The VRC-based approach has very significant area overhead. The configuration register, the multiplexers for the PE input-connections and operation selection are considerable. Moreover, the PEs implement all of the operations (but only one operation is selected at the same time for a given PE). The level-back parameter further influences the area overhead. The higher this parameter the higher is the area required to implement the input-connections for the PEs. This is the reason why in hardware implementations the level-back parameter is usually set to one.

This PL-based filter evaluation significantly accelerates the evolution but also limits it in some matter. The PE multiplexers

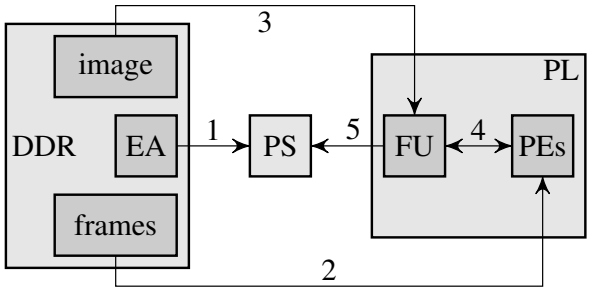

Figure 4. Acceleration of image filtering by DPR

prolong the propagation paths in the FPGA and cause longer evaluations than it would be possible without multiplexers.

3) Acceleration by DPR: The principle of the PL-based hardware acceleration by means of DPR is depicted in Figure 4 which is very similar to the VRC-based approach. The difference is in step 2, because the array of PEs is configured by the configuration frames instead of the configuration register. This configuration replaces the PE with another one which implements another operation. Therefore, there is no area overhead inside the PEs and the PE outputs are not multiplexed either. The input-interconnections can be similarly changed by reconfiguration.

As the final result, the DPR-based approach has no area overhead like the VRC-based approach. However, mutation by means of DPR takes much longer than the update of the configuration register of VRC. The higher the number of mutations is the longer will take the reconfiguration (because more frames are needed to be downloaded). However, the evaluation can be performed faster since this approach does not require multiplexers and therefore, the propagation paths are not prolonged (which leads to higher operational frequency).

V. Experimental Results

The considered scenarios of evolvable hardware design were evaluated by experiments performed on a developmental board equipped with an XC7Z020-1CLG484CES device. The evolution of an image filter was considered in the experiments. The (1+4) evolutionary strategy was used with a PE array consisting of 8 columns and 4 rows and implementing 16 operations shown in Table I. Kernel size 3×3 was considered (9 filter inputs) with Lena benchmark image of size 256×256 and 5% "salt and pepper" noise. The level-back parameter was selected to be 1 in order to not put the hardware implementations into disadvantage. All experiments were focused on the time needed to evaluate a given number of generations and the efficiency of a particular search method was not considered at all. Similarly, the area overhead is not taken into account because it is safe to assume that even the most area consuming VRC-based approach can be synthesized into a Zynq-7000 AP SoC device. The device used in the experiments has 106 400 flip-flops and 53 200 LUTs (6 650 slices) [10] while an implementation of a VRC of the same size requires approximately 1 290 slices and 1 084 flip-flops in a Virtex-5 FPGA [17] which has similarly 6-input LUTs.

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

Table II
Experimental results

	Mutations (1)	Individ. (µs)	Generation (µs)	Generations (s^{-1})	Accel. (1)
PS		225 285.3	901 141.1	1.1	1
i5		42 372.9	169 491.5	5.9	5
VRC		469.3	1877.2	532.7	484
DPR	1	206.2	824.8	1212.4	1102
DPR	2	247.2	988.8	1011.3	919
DPR	3	288.2	1152.8	867.5	789
DPR	4	329.2	1316.8	759.4	690
DPR	5	370.2	1480.8	675.3	614
DPR	6	411.2	1644.8	608	553
DPR	7	452.2	1808.8	552.9	503
DPR	8	493.2	1972.8	506.9	461

The achieved results are summarized in Table II where the columns from left to right contain the type of the approach, the number of mutations used to create new individuals, the required time to assemble and evaluate an individual (image filter), the required time to assemble and evaluate one generation, the number of assembled and evaluated generations per second, and the achieved relative acceleration in comparison with the pure PS-based approach. In the first three approaches 7 mutations are considered. This information is not present in Table II because for those approaches the mutations do not influence the execution time.

A. Software-Based Symbolic Regression

The first approach considered was the pure-software symbolic regression executed on the on-chip processor (PS-based approach). The evaluation was limited to 100 generations and 10 runs were considered. The average number of cycles was 30 038 035 659 which is equivalent to approximately 90

seconds (the cycles are counted by the global counter which is incremented in each two cycles, and the processor frequency is 667 MHz).

The implemented software-based approach was pre-ported to desktop computers by changing only the standard output interface and the time measurement (the parameters of EA and the input image remained the same). The implementation was tested on a powerful Intel Core i5 661 3.33 GHz processor (the approach is designated as i5 in Table II). The achieved average number of generations per second was 5.9 (based on 10 measurements and 50 000 generations). According to these measurements this processor was five times faster than the PS of Zynq-7000 AP SoC for solving the given symbolic regression problem. It should be noted that the highest power consumption of such a processor is approximately 80 W (not counting the other parts of the computer) while the estimated consumption of the used Zynq-7000 AP SoC device is below 3 W [10].

B. Acceleration by VRC

A set of PE operations (8-bit logic and arithmetic functions) was synthesized in order to measure the increased propagation delay caused by the additional multiplexers. The maximum propagation took 2.560 ns without the multiplexers which mean that the evaluation without them could be performed with 391 MHz operational frequency. After inserting the multiplexers into the PE array (under the assumption that the level-back parameter is 1) the maximum propagation path became 7.274 ns which corresponds to 138 MHz operational frequency and means 65% decrease.

A PE can be configured by 12 bits (4 bits for encoding the connection to one of the 9 inputs or 4 PE outputs from the neighboring column, 4 bits for the second PE input, and another 4 bits for selecting one of the 16 functions). For the array of 8 (columns) $\times$ 4 (rows) PEs this means 384 configuration bits. However, these bits can be written in parallel into the configuration register, i.e. the overhead in mutation time caused by VRC is only 7.274 ns (1 clock period).

During the evaluation $254^2 = 64\ 516$ inputs are applied to the image filter with 138 MHz operational frequency. This means 64 516 $\times$ 7.274 ns, i.e. 469.3 µs evaluation time for an image filter (under the assumption that the array is pipelined and the initial fill-up of 8 columns is negligible in comparison with the overall evaluation time).

The required time to assemble and evaluate an image filter is 7.274 ns + 469.3 µs $\doteq$ 469.3 µs which yields 532.7 generations per second. Therefore, the relative acceleration achieved by the VRC-based approach is 484. This is very impressive in comparison with the 5-times acceleration achieved by the Intel Core i5 processor.

C. Acceleration by DPR

The reconfiguration time of single frames was measured and 13 677 cycles were observed (average of 10 measurements) which is equivalent to 41 µs. The bit stream length was 3 891 $\times$ 32-bit words (15 564 B) with 3 737 words of actual frame data (the rest of the words were synchronization and configuration operations). This gives 380 MB per second configuration speed (400 MB is the official download speed in the non-secure PCAP-mode [20]). The reconfiguration with a 100 MHz ICAP would take 3 891/100 $\doteq$ 39 µs which roughly equals to the download speed achieved in our experiments. However, there is published evidence that ICAP can be successfully overclocked [9] and achieve significantly higher download speed. Furthermore, the frame relocation necessary to reduce the number of stored bit streams requires the computation of cyclic redundancy check sums. This computation can be executed concurrently with the download and does not need to be pre-computed like before the DMA transfer of PCAP. If these considerations are taken into account then the use of PCAP looks not so advantageous for evolvable hardware design (but ICAP is available also in Zynq-7000 AP SoC [20]).

The achieved reconfiguration time implies that a single mutation performed by DPR will take at least 41 µs which is very significant in comparison with the 7.274 ns of the VRC-based approach (almost 6 000 times longer for a single mutation). Furthermore, if several mutations are performed then several frames are needed to be reconfigured. It would be possible to include more than one PEs into the configuration frame in order to decrease the overall configuration time of the PE array. This could be achieved at the expense of the numbers of pre-generated bit streams (the number of bit streams increases exponentially with the number of included PEs).

On the other hand, the evaluation can be performed at 391 MHz instead of 138 MHz (because there are no additional multiplexers in the case of DPR-based approach). The increased operational frequency by factor of 3 may seem negligible in comparison with 6 000 times longer mutations, but can be still very advantageous during evaluations. The evaluation of the pipelined image filter will take only 64 516 $\times$ 2.56 ns $\doteq$ 165.2 µs if the operational frequency is 391 MHz which is very impressive in comparison with the 469.3 µs evaluation time of the VRC-based approach. The total amount of time necessary to assemble and evaluate an image filter is (41 µs $\times$ m) + 165.2 µs, where m is the number of mutations. The results for various numbers of mutations are shown in Table II. It can be observed that the DPR-based approach achieves better acceleration than VRC (if the number of mutations is at most 7 which is reasonable since usually small number of mutations is applied).

It should be noted that the results for the DPR-based approach are only estimates computed based on the reconfiguration time of one frame. There are some other computational overheads which were not considered yet (e.g. frame relocation, image filter copy). However, the possible accelerations are very promising which gives a good ground for the DPR-based evolutionary hardware design in the Zynq-7000 AP SoC platform.

VI. Conclusions

The paper introduced Zynq-7000 AP SoC from the perspective of an evolvable hardware designer. The platform provides new features such as PCAP or asymmetric multiprocessing which can be exploited to the benefit of evolvable hardware design. Other characteristics like on-chip ADC are also attractive for adaptive hardware.

Evolvable hardware design by VRC and DPR was considered in the paper. The advantages and disadvantages were compared in context of area overhead, execution time, reconfiguration time and throughput. Experiments with the symbolic regression problem were performed in order to demonstrate the possibilities of the platform. The performance of the on-chip processor and a relatively powerful desktop computer were compared with the performance achieved in the assistance of programmable logic. The experiments confirmed the superiority of the platform for evolvable hardware design which was at least 500 times faster using reconfigurable logic and DPR, and clearly outperformed even a powerful desktop computer.

The preliminary analysis and the experiments confirmed that Zynq-7000 AP SoC has the potential to become the next revolutionary step in evolvable hardware design. Further work will be conducted to exploit the possibilities of this platform.

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), the IT4Innovations Centre of Excellence (CZ.1.05/1.1.00/02.0070) and the Czech science foundation under the project Natural Computing on Unconventional Platforms (GAP103/10/1517).

References

[1] 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.

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

[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] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] K. Glette, J. Torresen, and M. Hovin, "Intermediate level FPGA reconfiguration for an online EHW pattern recognition system," in *NASA/ESA Conference on Adaptive Hardware and Systems, AHS 2009*. IEEE, 2009, pp. 19–26.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.

[19] 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.

[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.