A Highly Flexible Reconfigurable System on a Xilinx FPGA

Tomáš Drahoňovský, Martin Rozkovec, Ondřej Novák
Institute of information technologies and electronics
Faculty of Informatics, Mechatronics and Interdisciplinary studies, Technical University of Liberec
Liberec, Czech Republic
{tomas.drahonovsky,martin.rozkovec,ondrej.novak}@tul.cz

***Abstract* —Runtime reconfigurable systems become more prevalent in numerous practical applications because these systems have a great flexibility. This paper presents a reconfigurable system implemented on Xilinx Field Programmable Gate Array (FPGA) where partial bitstream relocation (PBR), configuration memory readback and internal registers restoration techniques are supported. It can reduce a number of partial bitstreams stored in memory, save the implementation time and generally increase the flexibility of the reconfigurable system. The article describes a relocatable system creation where the relocation procedure is based on the bitstream major address modifications and design where the relocation of individual modules including their internal states is supported.**

Keywords—FPGA reconfigurable systems; partial bitstreams relocation, configuration memory readback and restoration

I. INTRODUCTION

FPGAs are semiconductor devices used for processing of the digital information. It is based on a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. The possibility of re-programming and high performance of nowadays FPGAs allows developing and implementing complex systems with low budget in short time.

Currently, the most of commercial FPGAs are using SRAM based configuration memory. By changing the content of the memory (i.e. by performing full or partial reconfiguration) it is possible to adapt functions of the system to new computational requirements at run time. Run time reconfiguration can improve performance, increase functional density and reduce power dissipation [19].

The dynamic partial reconfiguration (DPR) allows designers to use part of the FPGA for more reconfigurable modules in time-multiplexed way. By this way one can reduce the area, obtain higher design security and improve FPGA fault tolerance. However, since a unique partial bitstream is needed for each reconfigurable module and these modules can be mapped to multiple reconfigurable regions, nearly identical partial bitstreams must be stored in memory. Another unpleasant characteristic of the DPR is reconfigurable module's initial states setting after every reconfiguration. These issues can be eliminated by using PBR technique with configuration memory readback and restoration.

In this paper we present a reconfigurable system where the relocation of reconfigurable modules including their internal states among FPGA device is supported. The Xilinx Virtex-6 device and Integrated Software Environment (ISE) 14.7 are used for the suggested design verification.

The paper structure is following: After an overview of the related works, we introduce the essential tools. Description of the used techniques is in the section four. Section five includes experimental results and in section six we conclude the paper and mention possibilities of the future work.

II. RELATED WORKS

There have been many significant research projects focused on related topics that have explored a new areas in reconfigurable systems design. In [14] the authors present a partially reconfigurable system on an FPGA and methodology for increasing wireless sensor network agility. REPLICA [11] is a dedicated hardware relocation filter, which is capable of performing the necessary bitstream modifications during the regular download process. When the relocation (reconfiguration) is required the partial bitstream stored in a memory is used as an input of the filter. The bitstream must be modified in order to place it to required position. This filter targets FPGAs Virtex-E. The next version of this filter REPLICA2Pro [10] is very similar to previous but it targets Virtex II family. BiRF [5] is another hardware based relocation filter. In addition to Virtex II Pro it targets the newer FPGAs devices where the partial reconfiguration is performed via bus macro interface (Virtex 4 and some Virtex 5). In [4] the authors present a reconfigurable system which is based on swapping of relocatable partial bitstreams on FPGA Xilinx Virtex 4. Authors use the multiple frame write feature to significantly reduce configuration time. In [13] the hardware based solution for partial reconfiguration and bitstream relocation is described. This hardware component allows read back a partial bitstream modify it and write (relocate) it to another place in an FPGA.

There are some software solutions where the DPR and PBR are supported. PARBIT [7] is a C based software usable for partial bitstreams generation. PARBIT reads information from the original bitstream and the modified data are written to the target configuration bitstream. The relocation method for heterogeneous modules is discussed in [1]. This bitstream manipulation technique is based on precondition that there are identical routing resources in all equally sized reconfigurable partitions. These resources (i.e. interconnections) are used to connect non-identical elements when the modules are being

relocated. A different approach is proposed in [8], where authors describe design flow based on proxy logic position and routing constraining. It allows creating reconfigurable partitions with the identical pin positions. This enables relocating individual reconfigurable modules to various locations in the device. GoAhead [2] is a tool enabling implementation of run-time reconfigurable systems and reconfigurable modules relocation. It is based on a combination of its own GUI (Graphical User Interface) with the XDL (Xilinx Design Language) text format and standard Xilinx synthesis and implementation tools. COMMA [16] is a project where the authors solve the problem of interconnection between reconfigurable modules for the Virtex 4 devices. In [9] authors present a reconfigurable system with partial bitstreams relocation and states restoration techniques support for the Virtex 4 devices. In [12] a software based technique for hardware tasks switching on the FPGA is presented.

Design methodology presented in this work uses the current Xilinx Partial Reconfiguration (PR) tool kit with purchased PR license.

III. TECHNIQUES AND TOOLS

A. Dynamic Partial Reconfiguration

DPR is an advanced method that allows changing a single or multiple parts of an FPGA while the remaining portion of the circuit is running uninterrupted. This capability allows use smaller FPGAs for implementing the same design (compared with the bigger devices without DPR), which results in the reduction of cost and power consumption.

With DPR the function of the Reconfigurable partition (RP) can be changed by downloading one of the partial bitstreams (i.e. part of the FPGA configuration memory content) from an external memory. When the DPR technique is used the internal logic of the FPGA is split into static part and reconfigurable (dynamic) part/parts. Since these two parts are implemented separately there have to be some interface (bridge) between them (i.e. there have to be identical signal connection for all reconfigurable modules). There are two basic approaches used in Xilinx devices for bridging static and dynamic parts. The first one is designer-defined signal interface called *bus macros* (Virtex-4 and older) and the second one is automatically inserted interface called *proxy logic* (the newest devices).

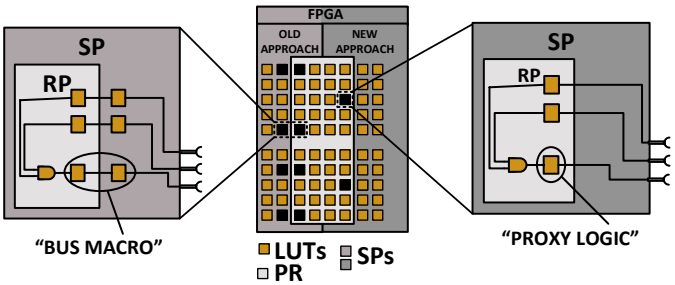

Fig. 1. DPR interaface basic principles

A bus macro is a slice-based interface (extra logic) required for the communication between static and reconfigurable parts of the design. Its cost is two LUTs (Look-Up Tables) per a bit signal crossing the boundary of the reconfigurable partition.

Proxy logic is a single LUT element automatically inserted by the implementation tools for each signal (partition pin) passing from static to reconfigurable region. It takes one LUT per a bit signal crossing the boundary between both parts of the device. Basic principle of these two approaches is illustrated in the Fig.1. More details about DPR can be found in [19].

Partial bitstream created by DPR for each Reconfigurable Module (RM) is bound to that specific Reconfigurable Region (RR) with respect to physical region boundaries (i.e. the partial bitstream of the RM created for the particular RR cannot be used directly for configuring of another RR at a different location in the device). For example, if there are two identical RM with the same logic configured in each of the RR, the partial bitstream of the one RM cannot be used in another RR. It means that the system memory is wasted by storing two different partial bitstreams with same logic and the implementation of the individual RM takes more time. An alternative approach is to use a technique called Partial Bitstream Relocation (PBR) instead of the common DPR.

B. Partial Bitstream Relocation

Partial bitstream (reconfigurable module) relocation is a DPR based technique that allows moving partially reconfigurable modules originally located in one area of an FPGA into different location of the same size and properties. PBR can improve the design flexibility, reduce memory usage and reduce time necessary for design implementation as only one bitstream needs to be created for identical RMs.

Depending on the direction where RM is relocated, we distinguish two relocation types. If we can relocate in columns only then we talk about 1-D relocation, if we can relocate in both directions (columns and rows), then we talk about 2-D relocation [3]. The possibility of 1-D or 2-D relocation is dependent on the region size and on the FPGA resources distribution. Since nowadays FPGA devices are heterogeneous (i.e. they are composed of many diverse resources) 2-D relocation is usually available for small and medium-sized reconfigurable regions where it is possible to find different positions with the same resources. Unfortunately, in spite of benefits of the PBR nowadays Xilinx FPGAs design tools do not support this technique.

Generally, the individual reconfigurable regions have to meet the condition of the same resource configuration availability. There are several requirements that must be fulfilled if one wants to create a design where the PBR is supported:

- The same region size
- The same number, type and composition of reconfigurable resources
- No static signal in the reconfigurable parts (except of the long lines)
- The same routing between static and reconfigurable regions
- No reconfigurable signal in the static part

C. *Configuration Memory Readback*

FPGAs Virtex 6 allow user to read the content of the internal configuration memory. This process is called Readback and we can say that it is essentially the opposite of the configuration process. There are two styles of the readback technique. The first one is called readback verify when the all configuration memory cells are read, including the current values on all user memory elements (LUT RAM, BRAM, etc.). The second one is called readback capture, it is a technique when in addition to reading all configuration memory cells the current state of the all internal registers (CLB, IOB) is stored to the configuration memory and read back from the device. Principle of the readback technique is illustrated in the Fig. 2 [20].

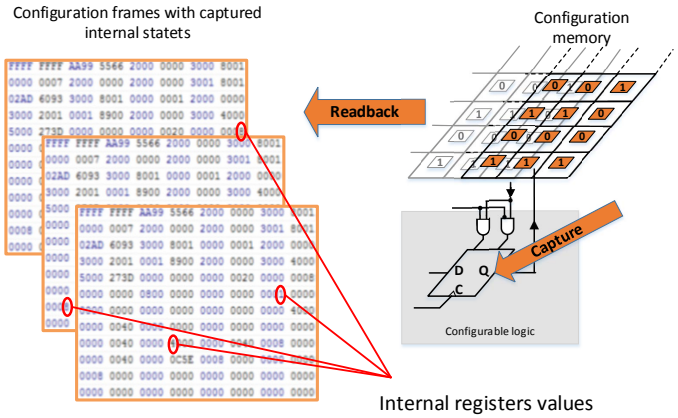

Fig. 2. Readback technique basic principle

Readback verify is most often used for verification of the read back configuration frames by comparing it to the original configuration frames that were written to the device (bitstream scrubbing technique).

In a readback capture, the internal states of LUTs, flip-flops, and certain IOB signals are extracted from the readback data to check proper design functionality (debugging). The *.ll file (created by BitGen tool) is used to correlate bits in the readback stream to actual logic bits in the device [20].

D. *Internal Register's States Restoration*

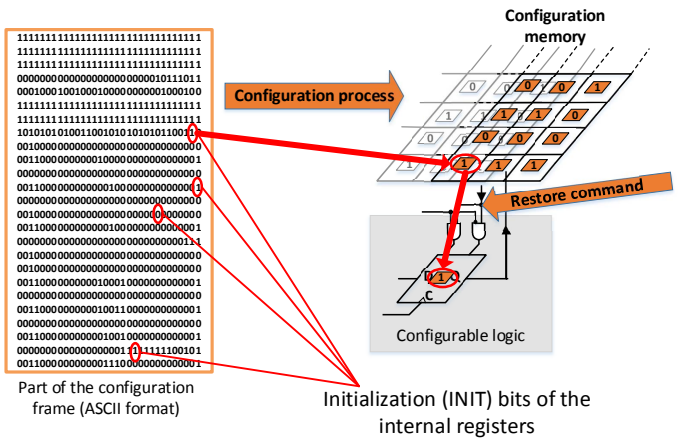

Fig. 3. Restoration process basic principle

For each flip-flop (register) on the FPGA device, there is one corresponding bit (INT0/INT1) in the configuration memory. To save state, user design can write a CAPTURE command to copy the value of the flip-flop to the INT0/INT1 bit of the configuration memory.

The user design then reads back the configuration bitstream and extracts the INT0/INT1 bit of interest. The restoration process is the reverse of state saving, when the bitstream is written to the configuration memory and a RESTORE command is used to copy INT0/INT1 bits to the flip-flops. Restore process principle is illustrated in the Fig. 3.

IV. IMPLEMENTATION

A. *Relocation Methodology*

Since the current Xilinx design tools (i.e. ISE Design Suite 14.7) with Xilinx license for DPR are used for our reconfigurable system design, there is the proxy logic interface utilized for bridging signals between SP and RP. Proxy logic approach itself is not suitable for PBR performing because of the routing tool uses its own routing scheme for every reconfigurable partition. It means that output or input signals of individual reconfigurable modules are routed from and to partition by different wires (failure to comply with condition of the same routing between SR and RP).

The only way, how to create a relocatable design without XDL or large interventions in the device bitstream composition is using the bus macros. The partial bitstreams relocation methodology used in this work is based on proxy logic to bus macro transformation.

Xilinx implementation constraints are used for meeting the particular conditions necessary for the relocatable design creation (Section III, Subsection B).

To meet the size, number and type of resources requirements the *AREA_GROUP* constraints are used. *AREA_GROUP* is an implementation constraint enabling partitioning of the design into physical region for mapping, packing, routing and placement [18].

Preventing placing of any static logic in the reconfigurable part and routing wires from static part through reconfigurable regions can be achieved by *PRIVATE* user constraints which completely forbid using of these static components inside the reconfigurable partitions.

There is another condition necessary for relocatable design preparation that is no reconfigurable module's signals should be routed through the static part of the design. This phenomenon occurs with the large number of the signal passing the border between SP and RP.

Reconfigurable signals routing in the static area can be limited by the other *AREA_GROUP* constraint *CONTAINED*. The example of the *CONTAINED* constraint is:

```
AREA_GROUP "name" CONTAINED=ROUTE;
```

Routing condition (same routing between static and reconfigurable regions) is satisfied by bus macros use. These bus macros are created by proxy logic LUT elements and LUT elements witch are added to each signal that crosses the border between reconfigurable and static part. Proxy logic LUTs are

placed at the border of the reconfigurable region and the new (added) LUTs are placed at the border of the static region. These LUTs are placed to specific site in the design by using *LOC* and *BEL* constraints.

Routing between these two bus macro's LUTs is controlled by DIRT (Directed Routing) constraints. Unfortunately the Xilinx hard macro (i.e. *.mnc file containing a detailed description of the design's parts) cannot be used for to ensure the same routing inside our bus macro because the one part of the bus macro is created by proxy logic LUTs and these are not present in the design at the beginning of the implementation.

The DIRT constraint may be restricted by the availability of the sets wires. It means that some wires in the device have been already used by other signals at the moment when the DIRT constraints are accepted by routing tools. A disregard of some DIRT constraints can be caused by this occurrence. This problem occurs with the large number of the reconfigurable modules pins (as well as the previous condition).

This issue can be prevented by setting the routing priority. The routing priority setting can be done by *PRIORITIZE* constraints written to Physical Constraints File (PCF). The example of this constraint is:

```
NET net_name PRIORITIZE = 100;
```

The procedures described above allow reconfigurable system preparation where the relocation of the partial bitstreams is possible by changing the frame address of these bitstreams.

Every bus macro based reconfigurable design causes an overhead of two LUTs per signal (wire) crossing regions (RP and SP) borders compared with system without partial reconfiguration. There is a possibility how to reduce this overhead when the both outputs of the added LUTs are used. Using both the LUT outputs is principally illustrated in Fig. 4. This approach allows reduction of the overhead to three LUTs per two signals.

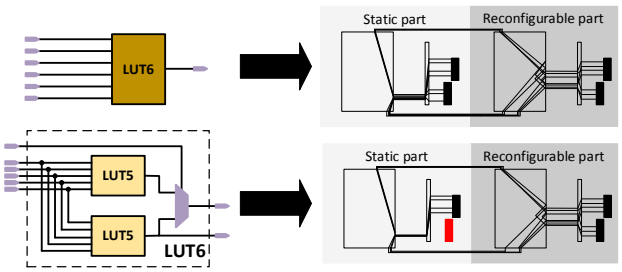

Fig. 4. Use of both LUT outputs

B. *Readback and Restoration Methodology*

To perform the readback and restore techniques it is necessary to set some parameters in the implementation tools and some attributes of the reconfigurable partitions before the implementation process is started.

For readback performing there are some BitGen parameters:

```
-g ReadBack: yes
-g Persist: yes
```

Readback capture is performed by entering the GCAPTURE command to the FPGA's configuration (CMD) register or instantiating the CAPTURE_VIRTEX primitive into the design and asserting the CAP input of this primitive.

After capture command asserting all the internal register's states are stored into the configuration memory.

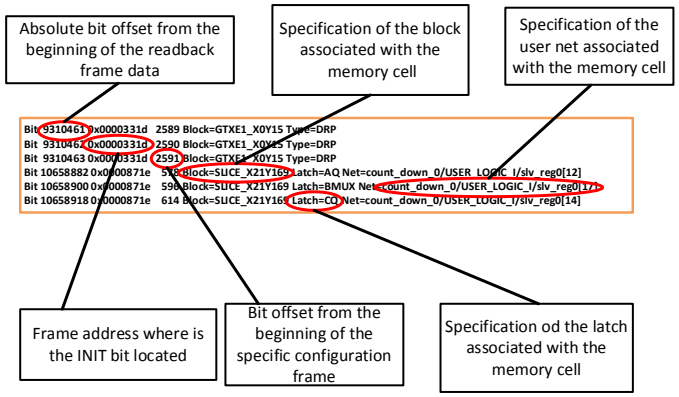

Fig. 5. Logic allocation file with description of the individual parts

The smallest addressable part of the configuration memory is called a configuration frame. To perform the state saving and restoration designer need to locate the desired INT0/INT1 bits in the configuration memory and in the configuration bitstream. For that the designer need to know the frame address of the INT0/INT1 bit and the offset of the bits within the frame. These information can be found in the logic allocation file (*.ll file) generated by the BitGen tool. A screen of the allocation file is illustrated in the Fig. 5.

Restore is performed either by entering the GRESTORE command to the FPGA's configuration (CMD) register or instantiating the STARTUP_VIRTEX primitive into the design. Both solutions assert the internal Global Set/Reset (GSR) signal of the FPGA and copy all INT0/INT1 bits to the internal flip-flops.

In order to restore some particular flip-flops instead of restoring whole FPGA it is necessary to setup a special mask for the part of the circuit where the restoration is undesirable before using the restore command.

Each column of the configurable logic within the FPGA has a mask bit. This bit is located in the special ('hidden') configuration frame and can be accessed through the configuration bitstream. Frame address of this frame is the same as frame address of the column of logic that to be masked, except the *Block type* part of the frame address.

Since Xilinx ISE Design Tools version 14.3 it is possible use the *Reset_After_Rerconfiguration* implementation attribute for special mask generation. This attribute can be written to the UCF file:

```
AREA_GROUP "name" RESET_AFTER_RECONFIG=TRUE;
```

V. EXERIMENTAL RESULTS

For testing of relocation methodology and internal states readback and restoration we created some reconfigurable systems.

The relocation technique for reconfigurable modules with large number of the inputs and outputs pins (144) was tested with reconfigurable system where 32 bit adders and 32 bit multiplier connected to the system via Advanced eXtensible Interface (AXI) were relocated. Fig. 6 illustrates the block schemes of the individual types of design for this testing application (i.e. system without DPR Fig.6/a, system with DPR support only Fig. 6/b and system where the PBR is supported Fig. 6/c). There are three reconfigurable partitions with size two reconfigurable frames (i.e. 640 LUT elements for Virtex 6 device).

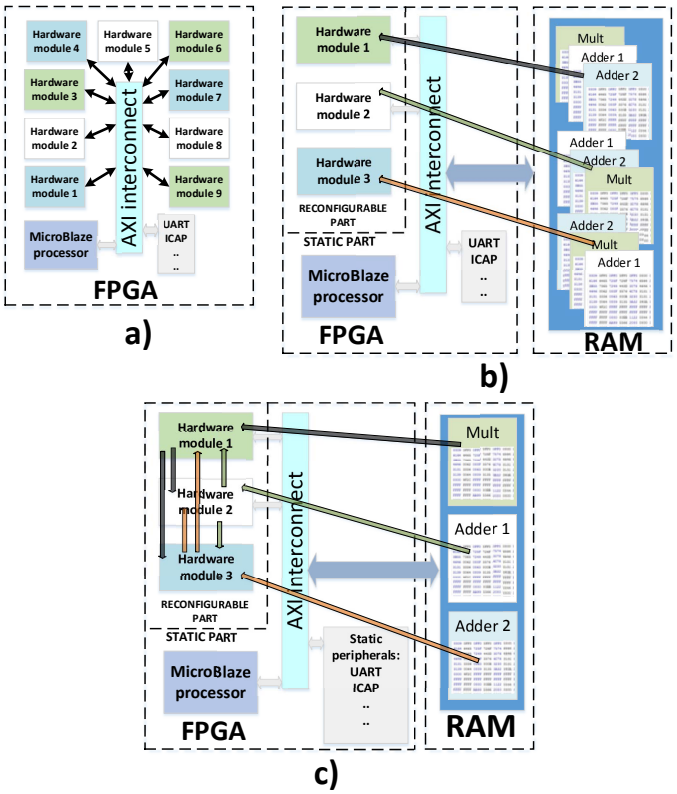

Fig. 6. Block schemes of the systems for relocatable design testing

The implementation details such as amount of the logic, implementation time, memory requirements for partial bitstreams storing and logic overhead caused by reconfiguration for four design types are compared in graphs in the Fig. 7. The first type is a design without DPR, the second one is a design where the standard proxy logic approach is used (this design supports only the partial reconfiguration but module relocation isn't supported). The third type is a design where one output of the added LUTs is used (classical bus macro approach) and the last one is a design where both outputs of the added LUTs are used. The two last designs allow the PBR technique.

The amount of all logic required is proportional on the number of system component and size of the whole design. The logic overhead caused by DPR is directly proportional to the number of the modules pins and used reconfiguration approach. The memory overhead (i.e. memory space required for the partial bitstreams storing) is directly proportional to the number and size of the reconfigurable modules. The time required for design implementation (time overhead) - it means time necessary for configuration bitstreams preparing is directly proportional to the number of the reconfigurable modules and to the size of the device (the implementation was performed on the computer with Intel i7 processor and 16 GB RAM).

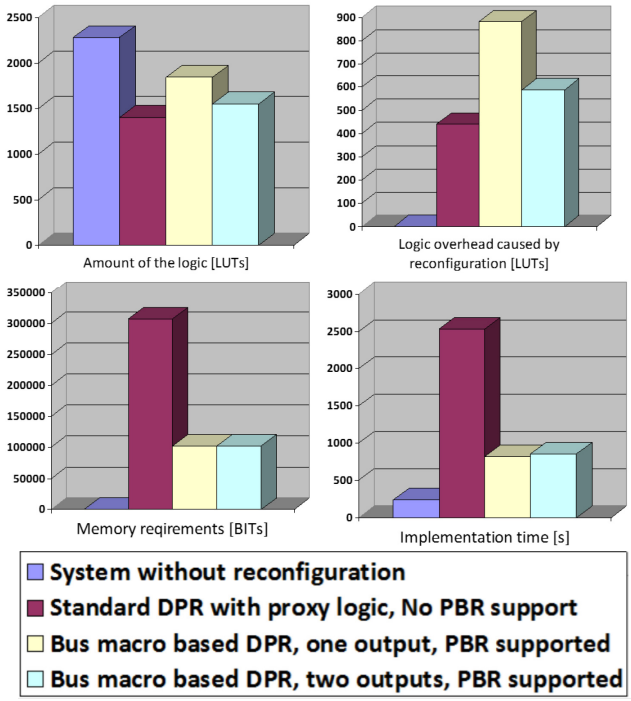

Fig. 7. Graphical comparison of the system with diferent reconfiguration approaches

Possibility of the relocation of the reconfiguration modules including their internal states was tested on the simple reconfigurable system where the actual states (internal values) of two counters were mutually relocated.

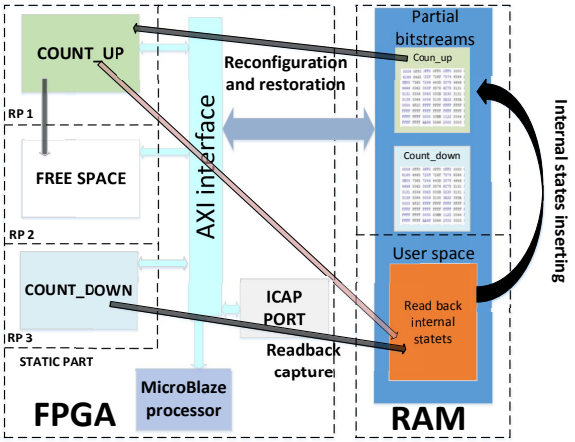

Fig. 8. Block schemes of the systems where the restore technique is used

In the Fig. 8 we can see the block scheme of the system where the internal states of the counters can be read back from the configuration memory by readback capture technique. These states are written to the partial bitstream (positions of the individual bits in the bitstream are stored in the allocation

file) and this bitstream is written back to the FPGA and the internal states are restored. For restoration via CMD register the sequence of the commands has to be written to the CMD register.

VI. CONCLUSION

Although, the systems designed on an FPGA show a great flexibility (configuration of the circuit at any time during its live cycle) the number of logic resources available for individual modules implemented inside the FPGA is restricted by the size of the circuit. DPR allows use of smaller devices, reduces power consumption, improves system upgradability and dependability and improves system fault tolerance. On the other side partial reconfigurable systems bring more design complexity, higher memory requirements, longer implementation time and individual reconfigurable modules are reset after every reconfiguration.

In this paper we describe a reconfigurable system where the partial reconfiguration is supplemented with partial bitstream relocation, configuration readback and internal states restoration techniques. It decreases memory space requirements, reduces implementation time and it increases a flexibility of the reconfigurable system. Proposed methodology can be used for FPGAs from Xilinx Virtex family (Virtex 5, Virtex 6 and Virtex 7).

Combination of relocation, restoration and readback capture technique allows performing hardware modules relocation including their internal states and synchronization of hardware modules after reconfiguration.

In the following work we will focus on application of the proposed methodology to the more complex reconfigurable systems (synchronization of a TMR (Triple Modular Redundancy) system after reconfiguration, relocatable multiprocessor system with the MicroBlaze processors).

ACKNOWLEDGMENT

The research is supported by the Student Grant Scheme (SGS) at the Technical University of Liberec, and co-financed by the Czech Ministry of Education, Youth and Sport. The work is also supported by the COST LD-13019 program and the COST Action IC1103-Median program.

REFERENCES

[1] Becker, T.; Luk, W.; Cheung, P.Y.K.; , "Enhancing Relocatability of Partial Bitstreams for Run-Time Reconfiguration," Field-Programmable Custom Computing Machines, 2007. FCCM 2007. 15th Annual IEEE Symposium on , vol., no., pp.35-44, 23-25 April 2007

[2] Beckhoff, C.; Koch, D.; Torresen, J.; , "Go Ahead: A Partial Reconfiguration Framework," Field-Programmable Custom Computing Machines (FCCM), 2012 IEEE 20th Annual International Symposium on , vol., no., pp.37-44, April 29 2012-May 1 2012

[3] Corbetta, S.; Morandi, M.; Novati, M.; Santambrogio, M.D.; Sciuto, D.; Spoletini, P.; , "Internal and External Bitstream Relocation for Partial Dynamic Reconfiguration," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* , vol.17, no.11, pp.1650-1654, Nov. 2009

[4] Ebrahim, A.; Benkrid, K.; Iturbe, X.; Chuan Hong, "Multiple-clone configuration of relocatable partial bitstreams in Xilinx Virtex FPGAs," *Adaptive Hardware and Systems (AHS), 2013 NASA/ESA Conference on* , vol., no., pp.178,183, 24-27 June 2013

[5] Ferrandi, F.; Morandi, M.; Novati, M.; Santambrogio, M.D.; Sciuto, D., "Dynamic Reconfiguration: Core Relocation via Partial Bitstreams Filtering with Minimal Overhead," *System-on-Chip, 2006. International Symposium on* , vol., no., pp.1,4, 13-16 Nov. 2006

[6] Gohringer, D.; Becker, J., "High performance reconfigurable multi-processor-based computing on FPGAs," *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on* , vol., no., pp.1,4, 19-23 April 2010

[7] Horta, E.; Lockwood, J. W., "Parbit: A tool to transform bitfiles to implement partial reconfiguration of field programmable gate arrays (FPGAS)," Dept. Comput. Sci., Washington Univ., Saint Louis, MO, Tech. Rep. WUCS-01-13, 2001,.

[8] Ichinomiya, Y.; Usagawa, S.; Amagasaki, M.; Iida, M.; Kuga, M.; Sueyoshi, T.; , "Designing Flexible Reconfigurable Regions to Relocate Partial Bitstreams," Field-Programmable Custom Computing Machines (FCCM), 2012 IEEE 20th Annual International Symposium on , vol., no., pp.241, April 29 2012-May 1 2012

[9] Jozwik, K.; Tomiyama, H.; Honda, S.; Takada, H., "A Novel Mechanism for Effective Hardware Task Preemption in Dynamically Reconfigurable Systems," *Field Programmable Logic and Applications (FPL), 2010 International Conference on* , vol., no., pp.352,355, Aug. 31 2010-Sept. 2 2010

[10] Kalte H, Porrmann M. REPLICA2Pro: Task Relocation by Bitstream Manipulation in VIRTEX-II/Pro FPGAs. In:*Proceedings of the 3rd Conference on Computing Frontiers*.; 2006: 403–412

[11] Kalte, H.; Lee, G.; Porrmann, M.; Ruckert, U.;, "REPLICA: A Bitstream Manipulation Filter for Module Relocation in Partial Reconfigurable Systems," Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International , vol., no., pp. 151b, 04-08 April 2005

[12] Morales-Villanueva, A.; Gordon-Ross, A., "On-chip Context Save and Restore of Hardware Tasks on Partially Reconfigurable FPGAs," *Field-Programmable Custom Computing Machines (FCCM), 2013 IEEE 21st Annual International Symposium on* , vol., no., pp.61,64, 28-30 April 2013

[13] Otero, A.; Morales-Cas, A.; Portilla, J.; de la Torre, E.; Riesgo, T., "A Modular Peripheral to Support Self-Reconfiguration in SoCs," *Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on*, vol., no., pp.88,95, 1-3 Sept. 2010

[14] R. Garcia, A. Gordon-Ross, and A. D. George, "Exploiting Partially Reconfigurable FPGAs for Situation-Based Reconfiguration in Wireless Sensor Networks," in 2009 17th IEEE Symposium on Field Programmable Custom Computing Machines. IEEE, 2009, pp. 243-246

[15] Rana, V.; Santambrogio, M.; Sciuto, D.; Kettelhoit, B.; Koester, M.; Porrmann, M.; Ruckert, U., "Partial Dynamic Reconfiguration in a Multi-FPGA Clustered Architecture Based on Linux," *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International* , vol., no., pp.1,8, 26-30 March 2007

[16] Shannon Koh, Oliver Diessel: COMMA: A Communications Methodology for Dynamic Module-based Reconfiguration of FPGAs. ARCS Workshops 2006: 173-182

[17] Xilinx Inc.:Command Line Tools User Guide (ug628), [online], [2011-10-19]

[18] Xilinx Inc.:Constraints Guide (ug625), [online], [2010-4-19]

[19] Xilinx Inc.:Partial Reconfiguration User Guide (ug702), [online], [2010-10-5]

[20] Xilinx Inc.:Virtex-6 FPGA Configuration User Guide (UG360), [online], [2012-9-11]