POLITECNICO DI TORINO
Repository ISTITUZIONALE

A portable open-source controller for safe Dynamic Partial Reconfiguration on Xilinx FPGAs

Original
A portable open-source controller for safe Dynamic Partial Reconfiguration on Xilinx FPGAs / DI CARLO, Stefano; Prinetto, Paolo Ernesto; Trotta, Pascal; Andersson, Jan. - ELETTRONICO. - (2015), pp. 1-4. (Intervento presentato al convegno 25th International Conference on Field Programmable Logic and Applications (FPL) tenutosi a London, UK nel 2-4 Sept. 2015) [10.1109/FPL.2015.7294002].

Availability:
This version is available at: 11583/2622325 since: 2016-10-07T17:59:15Z

Publisher:
IEEE

Published
DOI:10.1109/FPL.2015.7294002

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2015 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

25 April 2024

A portable open-source controller for safe Dynamic Partial Reconfiguration on Xilinx FPGAs

Stefano Di Carlo, Paolo Prinetto, Pascal Trotta
Dip. di Automatica e Informatica, Politecnico di Torino
Torino, Italy
email: name.familyname@polito.it

Jan Andersson
Cobham Gaisler AB
Goteborg, Sweden
email: jan.andersson@gaisler.com

***Abstract*—Thanks to their flexibility, increasing performances and low Non-Recurrent Engineering costs, SRAM-based Field Programmable Gate Array (FPGA) devices often represent the preferred platforms for the final deployment of highly reliable systems. In this context, Dynamic Partial Reconfiguration (DPR) is far from being widely adopted due to the additional complexity introduced during the hardware design phase, and the dependability issues related to the FPGA reconfiguration process itself. This paper presents a portable open-source controller for safely enabling self dynamic and partial reconfiguration of systems implemented on Xilinx FPGAs. The controller embeds configurable error detection and correction circuitry that enables a safe DPR by monitoring for partial bitstreams data errors. Experiments highlight the high performances achieved and the limited hardware resources needed to implement it on different devices. The HDL source code has been made available through the popular open-source Cobham Gaisler GRLIB IP-cores library.[1]**

I. Introduction

Nowadays, FPGAs represent a feasible and popular alternative solution to Application Specific Integrated Circuits (ASICs), providing flexibility, limited development costs and continuously growing computing capabilities. These characteristics push designers to adopt them also for the implementation of final release products with low time-to-market in a broad range of applications, spanning from high-performance computing data centers to digital signal and image processing in real-time embedded systems. FPGAs are also widely employed in mission-critical applications, or more in general, in those applications demanding high reliability [1], such as in space where, as example, the development time and costs for producing new ASICs are not affordable due to the very narrow market.

Modern SRAM-based FPGAs offer Dynamic Partial Reconfiguration (DPR) features [2], i.e., the ability to run-time change the functionality implemented by selected portions of a circuit while maintaining the rest of the design in a fully operating state. Although DPR can be used to increase reliability figures of a system [3], its adoption in applications demanding high reliability is actually very limited for two main reasons. The first one concerns the additional complexity introduced during the system design phase. To efficiently enable and manage run-time DPR, designers are often required to develop ad-hoc external or embedded hardware controllers. The latter, instead, is related to the dependability of the reconfiguration process itself. DPR exposes the system to errors affecting both the hardware controller and configuration data (i.e., *partial bitstreams*) that are used to overwrite portions of the FPGA configuration memory content at run-time. These errors are very critical since a mis-reconfiguration can lead, in the worst case, to a permanent disruption of the entire system functionality. Recovering from such errors could require a full device reconfiguration and/or reset of system operations [4].

This paper tackles the aforementioned issues by proposing a portable open-source embedded controller for safe DPR of systems implemented on Xilinx FPGAs. The main novelties w.r.t. state-of-the-art solutions mainly concern its high configurability and the possibility to introduce embedded error detection and correction circuitry, which monitors for bitstreams data errors during reconfigurations. The HDL source code has been made available through the open source Cobham Gaisler GRLIB GPL IP-cores library [5].

The paper is organized as follows: Section II briefly overviews related works, Section III details the proposed controller architecture, while Section IV shows the achieved results. Eventually, Section V summarizes the contributions.

II. Related Works

The Internal Configuration Access Port (ICAP) usually represents the best choice for enabling DPR, thanks to its higher bandwidth (i.e., 3.2 Gbps [6]). It provides a 32-bit interface to the device configuration engine accessible only through logic implemented in the FPGA.

Xilinx provides several IP-cores for enabling DPR and interfacing user designs with the ICAP. *XPS HWICAP* [7] and *AXI HWICAP* [8] represent two DPR controllers equipped with PLB and AXI4-Lite slave bus interfaces, respectively. Their main limitations concern the restricted applicability to processor-based systems and the inefficiency of data transfers due to the slave interface, that leads to reconfiguration throughputs far below the ICAP theoretical limit. A slightly more flexible solution is represented by the PRC/EPRC controller [9], which provides a FIFO user interface that allows its usage with custom user logic. However, its source code is not provided and netlists are available only for Virtex-5 and Virtex-6 devices.

Several works have been proposed in literature to overcome the limitations of the IP-cores provided by Xilinx. Recently, generic DPR controllers have been proposed in [10] and [11]. However, none of the aforementioned solutions takes into account DPR dependability issues. Few works can be found in literature targeting the development of reliable DPR controllers aimed at increasing overall reconfiguration process

[1] GRLIB source code (GPL version) can be downloaded at www.gaisler.com.

reliability. As example, in [12] and [13] authors propose and discuss several alternatives for increasing reliability of embedded DPR controllers (e.g., by applying Triple or Dual Modular Redundancy). Although the proposed solutions provide DPR controllers that are robust w.r.t. faults affecting the FPGA device, they do not tackle bitstream integrity issues, that are the focus of the controller proposed in this paper.

III. Proposed architecture

The proposed DPR controller has been designed in order to provide portability and configurability on different FPGA families. Depending on the target system and application requirements it can be configured at design-time, through VHDL generics, to operate in three different modes: (i) Synchronous/Asynchronous DPR, (ii) Dependable DPR with Cyclic Redundancy Check (*D^2PR-CRC*) or (iii) Dependable DPR with Error Detection and Correction (*D^2PR-EDAC*).

A. Synchronous/Asynchronous DPR

Figure 1 shows the architecture of the proposed controller in its basic configuration, i.e., *Synchronous/Asynchronous* mode.

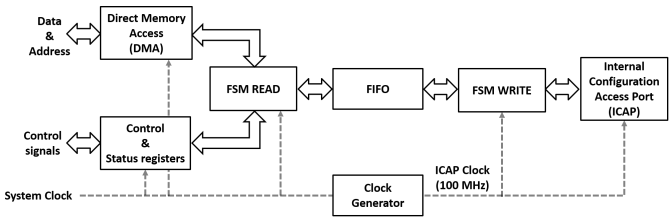

Fig. 1: DPR controller architecture for Synchronous/Asynchronous DPR mode.

It mainly consists of two control units (i.e., *FSM READ* and *FSM WRITE*), a First-In-First-Out (FIFO) buffer and clock generation circuitry. The *FSM READ* is supported by a Direct Memory Access (DMA) engine and a control and status registers block. *FSM WRITE* drives the ICAP interface and monitors its status. The control and status registers block includes several registers that can be read and written to setup, trigger, and monitor the reconfiguration process at run-time. The DMA is in charge of retrieving 32-bit bitstream data words from an embedded or external memory, starting from the address specified by the user through a 32-bit register in the Control and Status registers block. Obviously, depending on the actual system implementation, a custom wrapper may be needed to adapt the interface of the DMA engine to the user requirements (e.g., to connect the controller to a bus infrastructure or directly to a memory controller. If the system clock frequency is greater than the one sustainable by the ICAP (i.e., 100 Mhz), the proposed controller is configured in *Asynchronous DPR* mode. The *FIFO* buffer is used to transfer data across two different clock domains, i.e., *System clock* and *ICAP clock*. This buffer is implemented using one or more FPGA embedded Block-RAMs, and its depth can be configured through VHDL generics. *ICAP clock* is generated exploiting a clock manager hard macro sourced by the *System clock*. The *FSM READ* module is in charge of managing the bitstream data retrieval through the DMA. As soon as data are available in the buffer, the *FSM WRITE* controller reads-out and deliver them to the ICAP.

On the other hand, if the system clock frequency is lower than 100 MHz, the FIFO buffer and ICAP clock generation circuitry are not instantiated, since all modules, including the ICAP, can operate synchronously w.r.t. the input system clock (*Synchronous* DPR).

The registers block and DMA engine make the proposed architecture suitable to be employed in systems with custom user interfaces and also in processor-based bus infrastructures, allowing the processor to control and manage the reconfiguration process through software drivers. In this last case, the advantage deriving from the adoption of DMA is twofold: on one hand it frees the processor from directly managing the data transfer and repeatedly polling the status of the reconfiguration, while, on the other hand it accelerates bitstream data retrieval operations [14].

The maximum reconfiguration throughput in both operating modes can be estimated by the following equation:

$$Max_TP = \frac{BS}{max\{T_{sys}, T_{ICAP}\} \cdot \frac{BS}{32}} \leq 400MBytes/s \quad (1)$$

where BS represents the bitstream size in bits, while T_{sys} and T_{ICAP} are the system and ICAP clock periods, respectively. In Equation 1, the denominator represents the time required to write BS bits of data to the 32-bit interface of the ICAP.

The *Synchronous/Asynchronous* configuration do not natively provide any bitstream error detection and/or correction functionality. It represents a solution to enable DPR with very low hardware overheads. This configuration can be used when reliability is not a major concern, or if the user logic or memory controller used to retrieve bitstream data embed error detection and correction capabilities.

B. Dependable DPR (D^2PR)

Monitoring for partial bitstream data errors is essential to avoid FPGA mis-reconfigurations due to a corrupted partial bitstream. Two alternative methods and DPR controller architectures are presented. Both methodologies are based on introducing information redundancy at design-time in the partial bitstream data files generated by the Xilinx EDA tools. The first approach aims at detecting partial bitstream data errors by performing periodic on-line cyclic redundancy checks. At design-time the partial bitstream generated by Xilinx tools is parsed and processed by a software routine in order to compute and embed CRC signatures. In particular, the partial bitstream is split in blocks of 32-bit words. A signature is then computed for each data block and appended at the end of it to compose a new *protected* version of the partial bitstream. Figure 2 shows the architecture of the DPR controller when configured in *D^2PR-CRC* mode. With respect to the *Asynchronous* mode, the *FSM READ* is assisted by an on-line *CRC Generator* and a *CRC checker*. Basically, whenever a *Protected Bitstream* word is read from the memory through the DMA, it is directly written in the *FIFO* buffer. In addition, it is sent to the *CRC generator*, implemented as a parallel 32-bit *Linear Feedback Shift Register* (LFSR) [15], that computes at run-time a 32-bit signature in a single clock cycle. Whenever a data block is completely received, the associated pre-computed

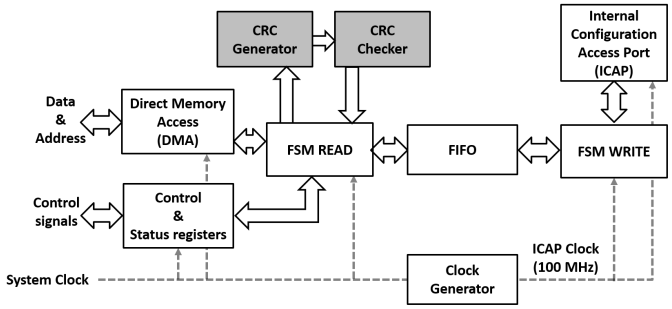

Fig. 2: DPR controller architecture for D^{2}PR-CRC mode.

signature is read trough the DMA and compared with the one extracted at run-time by the *CRC generator*. If no errors are detected, the currently received data block is validated and *FSM WRITE* can start reading it from the *FIFO* in order to deliver bitstream data to the ICAP. While the buffer is being emptied, *FSM READ* instructs the DMA engine to retrieve the following data block. Obviously, the *FIFO* buffer must be sized in order to store at least one full data block. The data block size can be configured by the user at design-time. This parameter has a direct impact on (i) bitstream storage memory requirements, (ii) actual reconfiguration throughput, and (iii) error detection capabilities.

Equation 2 shows the relationship between the total Protected Bitstream Size (PBS) and the user-defined data block size (Blk).

$$PBS = BS + 32 \cdot \lceil \frac{BS}{BlkS} \rceil \tag{2}$$

The second term of Equation 2 represents the additional contribution given by the 32-bit signatures embedded in the partial bitstream. The maximum reconfiguration throughput that can be achieved by the proposed architecture can be estimated by the following equation:

$$Max_TP = \frac{BS}{max\{T_{sys}, T_{ICAP}\} \cdot \frac{PBS}{32} + T_{ICAP} \cdot \frac{BlkS}{32}} \tag{3}$$

In Equation 3, the first term of the denominator represents the overall time needed to read the *protected bitstream* through the DMA, while the second term considers the additional latency, introduced by the adopted buffering approach, for delivering the last data block to the ICAP after its validation. The 32-bit CRC error detection approach provides detection of all burst errors, up to 32 bits in a single data block, and a tunable coverage on random errors depending on the chosen CRC polynomial and data block size [16] (see Section IV).

Finally, it is worth mentioning that, with respect to the Xilinx *PerFrameCRC* functionality offered *only* by 7Series FPGAs [6], the proposed approach represents a more flexible solution, allowing designer to tune memory requirements, error detection capabilities and timing overhead depending on the constraints imposed by the target application.

In *D^{2}PR-EDAC* mode, the proposed controller provides partial bitstream error detection and correction capabilities through an Error Correcting Code (ECC). In particular, a Single Error Correction Double Error Detection (SECDED) ECC has been chosen due to its representative target error model and its limited code and hardware overheads [17]. Partial bitstreams generated by the Xilinx EDA tools must be first processed at design-time by a software routine in order to produce a *Protected bitstream* composed of SECDED encoded words packets, that will be subsequently decoded by the DPR controller at run-time. The partial bitstream is split in blocks of 4 32-bit words. Each 32-bit word is encoded. The SECDED encoding process results in 7 bits overhead on a 32 bits input word [17]. The 28 overhead bits resulting from the encoding process are then grouped in a single 32-bit word and appended at the end of the 4-words data block (similarly to the aforementioned CRC approach) to compose a SECDED packet.

The architecture of the DPR controller configured in *D^{2}PR-EDAC* mode is depicted in Figure 3. In this case, *FSM WRITE* is assisted by an on-line SECDED decoder, that checks and decode the received words and, if a single error is found, correct them. SECDED packets are continuously read through the DMA and 32-bit encoded data words are written in the *FIFO*. *FSM WRITE* continuously reads out data from the buffer until an entire 5-words SECDED packet is reconstructed. Whenever the fifth word of a packet is read out, the decoding phase starts. It lasts four clock cycles, needed to decode four 32-bit bitstream data words. At each clock cycle a data word is decoded and delivered to the ICAP. If present, any single error is corrected before writing the data word in the configuration port. On the other hand, the reconfiguration process is terminated if uncorrectable errors are found during the decoding process. At the end of this phase, *FSM WRITE* restarts the process by reading the following 5-words packet from the *FIFO*.

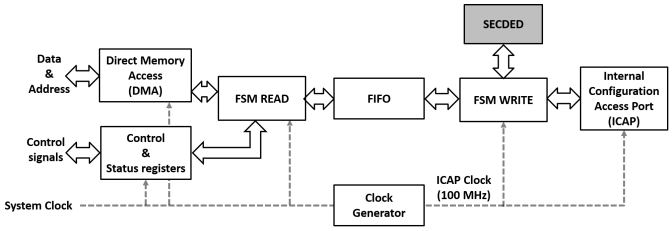

Fig. 3: DPR controller architecture for D^{2}PR-EDAC mode.

Similarly to the DPR controller operating in *D^{2}PR-CRC* mode, the SECDED *Protected Bitstream Size* and the maximum throughput achievable by the proposed controller can be estimated using Equations 2-3. However, in this operating mode, data block size ($BlkS$) is fixed and equal to 128 bits.

Integrating error correction in the DPR controller allows to avoid interruptions during reconfiguration processes due to single bitstream data errors, therefore preserving overall system performances.

IV. Experimental Results

The proposed DPR controller has been implemented on a *Virtex6-vlx240t* FPGA in order to be compared with other portable state-of-the-art and vendor solutions. The test-case is represented by a system in which the DPR controller is directly interfaced to an on-chip memory, implemented using Block-RAMs, storing the partial bitstream that is used to reconfigure an FPGA portion. User logic must trigger and monitor the

TABLE I: Comparison of the proposed DPR controller with state of the art and vendor solutions. The target device is a *Virtex6-vlx240t* FPGA.

Implementation	LUTs	FFs	BRAMs	Throughput [MB/s]
[10]	586	672	8	399.8
[11]	673	254	5	253
XPS-HWICAP [7]	799	746	1	8.5
AXI-HWICAP [8]	502	477	1	9.1
Synchronous	249	112	-	400
Asynchronous	443	226	1	400
D^2PR-CRC	588	278	1	395.4
D^2PR-EDAC	592	310	1	319.9

reconfiguration process by reading/writing the values in the *Control and Status registers* block. Reconfiguration time and throughput have been measured using an hardware timer embedded in the DPR controller. The system frequency is set to 200 MHz, ICAP frequency is fixed to 100 MHz, and the size of the bitstream needed to change the functionality of the selected reconfigurable FPGA portion is equal to 119.4 KiBytes.

When configured in *D^2PR-CRC* mode, the CRC polynomial implemented by the *CRC generator* is $x^{32}+x^{18}+x^{14}+x^3+1$. This particular polynomial provides detection of all burst errors up to 32 bits in a data block and detection of all 5 random errors for data blocks smaller than 31 Kib [16]. When operating in *D^2PR-CRC* mode, the theoretical throughput can be maximized by finding the data block size value that maximize Equation 3. In the considered test case, the data block size has been set to $BlkS = 5.5Kib$ to maximize the reconfiguration throughput.

Results shown in Table I highlight that it is possible to monitor bitstream data errors incurring in limited reconfiguraton timing overheads if the optimal bitstream data block size is selected at design-time when the proposed DPR controller is configured in *D^2PR-CRC* mode. The measured throughput values match the ones estimated using Equations 1 and 3, since the DPR controller is directly interfaced to a fast on-chip memory, and the throughput bottleneck is represented by the maximum ICAP clock frequency (i.e., 100 MHz).

With respect to state-of-the-art and vendor solutions, the proposed controller provides similar hardware overheads while being able to monitor and eventually correct bitstream data errors. At the same time it is able to sustain high reconfiguration throughputs. It is worth noting that both *XPS_HWICAP* and *AXI_HWICAP* can be used only in conjunction with processor-based systems, since they show an OPB or AXI bus slave interface. The processor or an external master in the bus is directly in charge of managing bitstream data trasfer, thus leading to increased timing overheads.

The HDL source code of *Synchronous/Asynchronous* and *D^2PR-CRC* modes has been already released and is part of the open-source Cobham Gaisler AB GRLIB IP-cores Library (v.1.4.1-b4156) [5]. GRLIB also includes several examples of LEON3-processor based systems including the proposed controller for Virtex-4, Artix-7 and Virtex-7 devices. The source code for *D^2PR-EDAC* mode will be made available in the next GRLIB release.

V. Conclusions

This paper proposed an open-source controller for safe dynamic and partial reconfiguration of Xilinx FPGA-based systems. The proposed controller is highly configurable and portable on different FPGA device families. Its source code has been made available through a popular open-source IP cores library.

Experimental results demonstrate the hardware overheads and performances of the proposed controller, highlighting its ability to monitor for partial bitstream data errors without incurring in notable hardware and timing overheads w.r.t. state-of-the-art solutions.

References

[1] N. Battezzati, L. Sterpone, and M. Violante, *Reconfigurable field programmable gate arrays for mission-critical applications*. Springer Science & Business Media, 2010.

[2] *Partial Reconfiguration of Xilinx FPGAs Using ISE Design Suite*, Xilinx Corporation, July 2012.

[3] L. Miculka and Z. Kotasek, "Generic partial dynamic reconfiguration controller for transient and permanent fault mitigation in fault tolerant systems implemented into fpga," in *Design and Diagnostics of Electronic Circuits Systems, 17th International Symposium on*, April 2014.

[4] S. Di Carlo, G. Gambardella, M. Indaco, P. Prinetto, D. Rolfo, and P. Trotta, "Dependable dynamic partial reconfiguration with minimal area and time overheads on xilinx fpgas," in *Field Programmable Logic and Applications (FPL), 2013 23rd International Conference on*, Sept 2013, pp. 1–4.

[5] Cobham Gaisler AB, *GRLIB IP Library User's Manual*, v1.4.1 - b4156 ed., 2015. [Online]. Available: www.gaisler.com/index.php/downloads/leongrlib.

[6] *Vivado Design Suite User Guide - Partial Reconfiguration v2014.4 (UG909)*, Xilinx Corporation, November 2014.

[7] *LogiCORE IP XPS HWICAP v5.01a (DS586)*, Xilinx Corporation, June 2011.

[8] *LogiCORE IP AXI HWICAP v3.0 (PG134)*, Xilinx Corporation, December 2013.

[9] *PRC/EPRC: Data Integrity and Security Controller for Partial Reconfiguration (XAPP887) v1.1*, Xilinx Corporation, June 2012.

[10] K. Vipin and S. Fahmy, "A high speed open source controller for fpga partial reconfiguration," in *Field-Programmable Technology (FPT), 2012 International Conference on*, Dec 2012, pp. 61–66.

[11] J. Tarrillo, F. Escobar, F. Lima Kastensmidt, and C. Valderrama, "Dynamic partial reconfiguration manager," in *Circuits and Systems (LASCAS), 2014 IEEE 5th Latin American Symposium on*, Feb 2014, pp. 1–4.

[12] A. Ebrahim, K. Benkrid, X. Iturbe, and C. Hong, "A novel high-performance fault-tolerant icap controller," in *Adaptive Hardware and Systems (AHS), 2012 NASA/ESA Conference on*, June 2012.

[13] A. Ebrahim, T. Arslan, and X. Iturbe, "On enhancing the reliability of internal configuration controllers in fpgas," in *Adaptive Hardware and Systems (AHS), 2014 NASA/ESA Conference on*, July 2014.

[14] M. Liu, W. Kuehn, Z. Lu, and A. Jantsch, "Run-time partial reconfiguration speed investigation and architectural design space exploration," in *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, Aug 2009, pp. 498–502.

[15] M. Ayinala and K. Parhi, "High-speed parallel architectures for linear feedback shift registers," *Signal Processing, IEEE Transactions on*, vol. 59, no. 9, pp. 4459–4469, Sept 2011.

[16] P. Koopman, "32-bit cyclic redundancy codes for internet applications," in *Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on*, 2002, pp. 459–468.

[17] E. Fujiwara, *Code design for dependable systems: theory and practical applications*. John Wiley & Sons, 2006.