Dependable Dynamic Partial Reconfiguration with minimal area & time overheads on Xilinx FPGAS

DEPENDABLE DYNAMIC PARTIAL RECONFIGURATION WITH MINIMAL AREA & TIME OVERHEADS ON XILINX FPGAS

Stefano Di Carlo, Giulio Gambardella, Marco Indaco, Paolo Prinetto, Daniele Rolfo, Pascal Trotta

Politecnico di Torino, Dipartimento di Automatica e Informatica
Corso Duca degli Abruzzi 24, I-10129, Torino, Italy
Email: {name.familyname}@polito.it

ABSTRACT

Thanks to their flexibility, FPGAs are nowadays widely used to implement digital systems' prototypes and, more frequently, their final releases. Reconfiguration traditionally required an external controller to upload contents in the FPGA. Dynamic Partial Reconfiguration (DPR) opens new horizons in FPGAs' applications, providing many new utilization paradigms, as it enables an FPGA to reconfigure itself: no external controller is required since it can be included in the FPGA. However, DPR also introduces reliability issues related to errors in the partial reconfiguration bitstreams. FPGA manufacturers currently provide solutions that are not efficient. In this paper new DfD (*Design for Dependability*) techniques are proposed. Exploiting information density of configuration data, they improve the performance while providing the same reliability characteristics as the previous ones.

1. INTRODUCTION

Field Programmable Gate Arrays (FPGAs) are semiconductor devices made up of a cluster of interconnected functional blocks, that can be programmed to perform user-defined logic functions. Modern Xilinx FPGAs (e.g., Virtex family) offer an interesting feature to increase their on-the-fly adaptability. *Dynamic Partial Reconfiguration* (DPR) extends the native flexibility of FPGAs making possible to dynamically change selected portions of a circuit while the rest of the design is left unchanged and fully functional. The ability to time-multiplex hardware modules at run-time enables designing complex Systems-on-Chip (SoCs), thus reducing cost, board space, and power consumption [1].

DPR is performed by mean of automatically generated *partial bitstreams* that can be stored either outside the FPGA (e.g., in a RAM memory) or inside the FPGA (e.g., in Block-RAMs). To validate the integrity of a partial bitstream file, Xilinx Virtex devices implement a built-in Cyclic Redundancy Check (CRC) mechanism. Unfortunately, the configuration circuitry supports error detection only after a partial bitstream file has been fully loaded. As a consequence of this "late" check, a loaded faulty partial bitstream file may fatally compromise the entire design (both static and reconfigurable portions).

Two kinds of partial bitstream errors may occur: data errors and address errors, respectively. According to Xilinx specifications, when errors occur in the data portion of the bitstream, the recovery procedure is not critical, because just the reconfigurable area has been corrupted. Loading a new partial bitstream file is thus sufficient to fix the error [2]. However, an intensive fault injection campaign, proved that this is true only under specific conditions. In fact, if links between static modules are routed through the reconfigurable area, a full reconfiguration of the FPGA could be required. When errors occur in the address section of the bitstream, the corruption may modify the static portion of the design and, therefore, a safe recovery requires to reconfigure the whole FPGA.

Highly reliable or real-time applications must adopt additional error detection mechanisms to overcome this issue. In the new 7 series family, Xilinx introduced *PerFrameCRC* option [2]. This new feature allow the insertion of a CRC value after every frame in a partial reconfiguration bitstream, checked before shifting the frame into memory. Nonetheless, this solution can only be applied in the 7 series devices, introducing a significant delay in the reconfiguration process; for older technologies, Xilinx proposes the methodology explained in Sec. 2.

This paper proposes an innovative methodology aimed at providing a dependable DPR flow, minimizing both area occupation and reconfiguration time. Exploiting data provided by Xilinx manuals, a tool has been developed to isolate and protect the critical parts of a *partial bitstream* (i.e., address part). In addition, *Design for Dependability* (DfD) rules are provided to protect critical modules in the static part of the FPGA.

The paper is organized as follows: Sec. 2 describes in detail the Xilinx solution. The proposed methodology and the related experimental results are presented in Sec. 3 and 4, respectively. Sec. 5, eventually, concludes the paper.

2. XILINX APPROACH

The problem of dependable partial reconfiguration has been addressed by Xilinx in the *Partial Reconfiguration User Guide* [2], where a Partial Bitstream CRC Checking is suggested.
The main idea is to split the original partial bit file into *blocks*, and for each block to calculate a single word *CRC* signature. The hardware implementation requires a *CRC* evaluator, a Finite State Machine (FSM) to control the reconfiguration process (i.e., a Reconfiguration Manager) and a set of BRAM(s) to buffer the data. Depending on the available FPGA resources, the designer need to choose the best block size, that impacts on both reconfiguration time and internal memory occupation.
By increasing the number *N* of blocks, more *CRC* signatures must be stored, increasing the bitstream size. At the same time, the memory occupation becomes smaller, since the required buffering capability equals the block dimension.
By decreasing the number of blocks, fewer *CRC* signatures must be stored in the bitstream. However, during the *CRC* checking process, more words must be stored in the BRAM inside the FPGA, thus increasing the area occupation.
The total reconfiguration time is due to two contributions:

$$T_{X-CRC} = T_{read} + T_{block} \tag{1}$$

T_{read} is the time required to load the bit file from the external memory:

$$T_{read} = \frac{K+N}{\min(f_{ICAP}, f_{Mem})} \tag{2}$$

where K is the bitstream dimension in terms of 32 bit words; N is the number of *CRC* signatures in terms of 32 bit words; f_{ICAP} is the working frequency of the ICAP; f_{Mem} is the memory working frequency.
T_{block} is the time spent loading the buffered block from BRAM to the ICAP:

$$T_{block} = \frac{K/N}{f_{ICAP}} \tag{3}$$

where K/N is the block dimension in terms of 32 bit words.

3. PROPOSED METHODOLOGY

Despite the solution proposed by Xilinx is fairly comprehensive, it may implies significant time and area overheads. In the sequel we propose a methodology that reduces these overheads, by protecting just the critical part of the partial bitstream. Some ad-hoc *DfD* rules are also introduced.

3.1. Partial bitstream file splitting

As aforementioned, the partial bit file is composed of three main parts. The first part contains the address and control information. The second includes the data for the reconfiguration in the selected frame(s). The last part is the built-in ICAP CRC checksum.
It is straightforward that, in terms of dependability, the most critical part is the first one, since it defines the portion of the FPGA to reconfigure. In fact, if an error occurs in an address or control information, a static portion of the FPGA could be unintentionally reconfigured and the system could become inoperative.
The proposed approach deeply protects the most critical words of the partial bitstream. At reconfiguration time, the critical words are checked, while the non-critical words are loaded from the external memory and directly sent to the ICAP, without any time overhead or buffering. The proposed solution requires a *CRC* evaluator and a Reconfiguration Manager to control the reconfiguration process, while no BRAMs are required.
The reconfiguration time with the proposed *CRC* checking is:

$$T_{OUR-CRC} = \frac{K+C}{\min(f_{ICAP}, f_{Mem})} \tag{4}$$

where C is the number of critical words.
Fig. 1 plots the ratio $T_{X-CRC}/T_{OUR-CRC}$, which proved to be always >1. Therefore, the proposed solution is always faster then the Xilinx one.

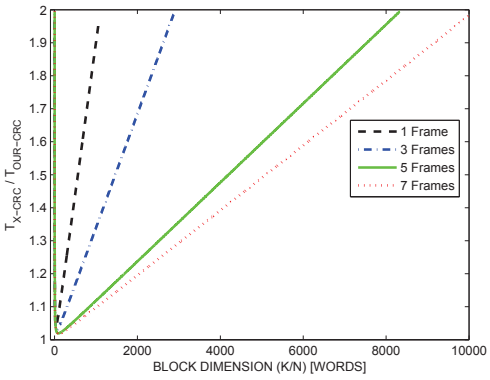

Fig. 1. Comparison between proposed solution and Xilinx solution

Despite the proved advantages of the proposed solution, to assure a high dependability of the reconfiguration process, we have to guarantee that an error in the non-checked part of the bit file will not lead to a fault in the system. The jeopardy is that some static connections are routed in the reconfigurable area, and, due to a faulty reconfiguration process, the link between two points could be broken. This goal is achieved by fulfilling the following *DfD* rules: (1) potential critical links must not cross any reconfigurable area; (2) connection inside critical modules must not cross (i.e., be routed through) reconfigurable areas.

3.2. DfD#1: Critical links protection

To ensure a dependable reconfiguration process, critical connections must be protected. These include links between External-Memory to Memory Controller, Memory Controller to Reconfiguration Manager and Reconfiguration Manager to ICAP. In order to guarantee that these links do not cross the reconfigurable area, after the automatic routing performed by the synthesis tool, the layout must be checked and some links manually re-routed, if required.

3.3. DfD#2: Critical modules protection

The second DfD rule imposes that all critical modules must be protected. In systems which use partial reconfiguration, all modules involved in DPR must be considered critical. In addition, also design specific modules could be considered critical. Their integrity can be preserved by constraining critical modules in predefined physical region called *partitions*. Xilinx PlanAhead tool enables the user to manually place a module in a specific area, guaranteeing that all the specified hardware and the related connections are inside the physical regions.

4. EXPERIMENTAL RESULTS

This section reports a set of experiments performed to validate the proposed methodology and to compare its performance with the Xilinx solution.
The experimental setup includes a *Leon3* [3] based SoC, implemented on a Xilinx ML403 demo board, equipped with a Xilinx Virtex 4 FPGA device and 64 MB of DDR SDRAM [4]. The SoC contains a reconfigurable area in which it is possible to dynamically load two modules, namely the *APBUART* and the *GPTIMER* from *GRIP Library* [5]. Both modules require a reconfigurable area composed of 2 frames. The SoC also includes an ad-hoc reconfiguration manager connected to the AMBA bus. The manager addresses the external memory, loading bitstream files, and manages the whole reconfiguration process, performing the *CRC* check and eventually the DPR through ICAP. The synthesis of the overall design has been performed using Xilinx ISE Design Suite 14.4. The *Leon3* processor works at 66 MHz, the ICAP controller and the DDR SDRAM at 100 MHz.

4.1. Xilinx approach implementation

The Xilinx protection methodology has been implemented using a parallel CRC-32 to minimize the CRC latency. The selected polynomial is *0x90022004* which guarantees Hamming distance equal to 6 [6].
Fig. 2 shows the relation between the reconfiguration time and the block size. The solid line plots the reconfiguration time calculated using Eq. 1 while the dots report the measured reconfiguration time, confirming the model relevancy. The synthesized reconfiguration manager requires 290 slices and 1 BRAM.

Table 1. Area occupation and reconfiguration time of different implementations

Solution	Block Size [bit]	CRC [#]	BRAM [#]	RM [# slices]	Rec. Time [us]
w/o CRC	0	0	0	197 (3.60%)	61.04
Proposed	1x16	42	0	295 (5.39%)	61.78
Xilinx	64 x 32	31	1	290 (5.30%)	63.72
Xilinx	1 x 32	1,969	0	290 (5.30%)	77.88
Xilinx	1,969 x 32	1	8	290 (5.30%)	73.49

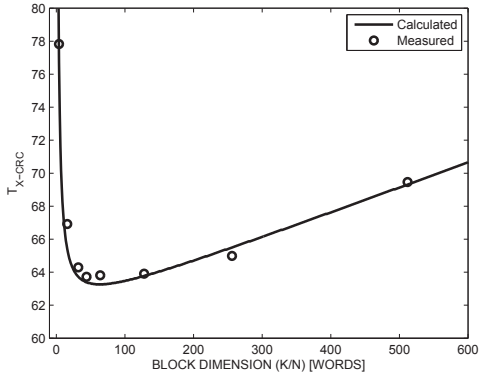

Fig. 2. Reconfiguration time with 2 Frames

4.2. Proposed approach implementation

Differently from the Xilinx solution, the proposed approach is designed to protect 16-bit critical words. A smaller CRC can therefore be adopted. We implemented a parallel CRC-16 with polynomial equal to *0x968B* which guarantees Hamming distance equal to 7.
The two presented DfD rules have been applied to the proposed design. Partitions have been created using Xilinx PlanAhead to protect the SoC critical modules (i.e., Reconfiguration Manager (RM) and Memory Controller (MC)). To ensure that the processor keeps running also after a faulty reconfiguration, the *Leon3* has been constrained in a specific region, introducing a minimal degradation (1.2%) in the working frequency. All critical connections have been checked, in order to assure that they do not cross the reconfigurable area, and only 2 links were manually re-routed using the Xilinx *FPGA Editor* Tool (see Sec. 3.2).
The synthesized reconfiguration manager requires 295 slices and no BRAMs. A fault free reconfiguration takes 61.78 μs.

4.3. Comparison

Table 1 compares the two analyzed solutions, in terms of area occupation and fault free reconfiguration time. The assets of the proposed solution are no BRAM occupation and a shorter reconfiguration time compared to the Xilinx solution, with a very small area overhead in the configuration manager (increase of 1.7% of slices) due to a more complex FSM. The reported reconfiguration times, computed with the model presented in Sec. 3, are related to a 2 frames reconfigurable area (1,969 words), with a 1506 Mbit/s throughput of the DDR SDRAM. For sake of completeness, Table 1 also provides information about the worst cases of the Xilinx solution, and a CRC free DPR system.

So far the performance comparison have not considered the presence of faults. The rest of this section will compare DPR performance in case of faults in the bitstream, which have been injected in the words of the data portion, only. This condition represents the worst-case condition for the proposed solution, since the fault is detected just at the end of the process by the ICAP CRC check (requiring a new DPR), while in Xilinx' solution is detected as soon as the CRC of the block is checked (requiring the reload of the block only). The time overhead introduced by faults in the loaded blocks for the two considered solutions is therefore influenced by the DPR rate, and by the word error probability observed when loading bitstream blocks. Fig. 3 and Fig. 4 analyze the difference in system activity time spent for DPR in the two solutions over a one day observation period for different DPR rates and word error probabilities, thus enabling an easy comparison of the two solutions.

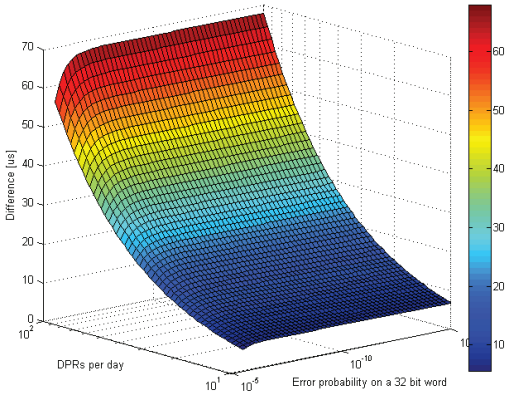

Fig. 3. Difference of DPRs time in 1 day - 2 Frames

Fig. 3 analyzes the case of a 2 frames reconfigurable area, i.e., a partial bit file of 1,969 32-bit words. The experiments show that the proposed technology outperforms the Xilinx solution in all working conditions enabling a significant improvement in the overall reconfiguration time.

Fig. 4 performs a similar analysis, but considering a larger reconfigurable area composed of 8 frames, i.e., a partial bit file of 11,040 32-bit words. In this case, when the word error probability increases over 10^{-6}, the Xilinx solution should be preferred. This is due to the additional reconfiguration process required in our solution whenever data words are corrupted. Nevertheless, decreasing the error probability or the number of reconfigurations per day, the proposed methodology is the best one.

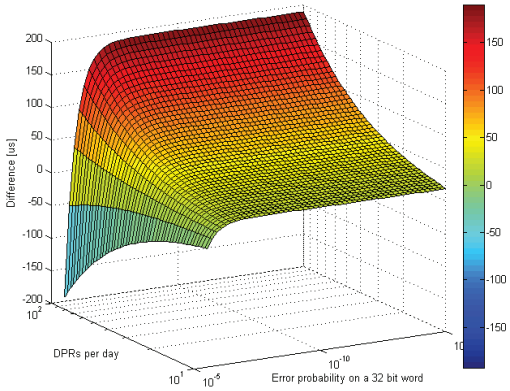

Fig. 4. Difference of DPRs time in 1 day - 8 Frames

5. CONCLUSIONS

This paper presented an innovative methodology for a dependable partial reconfiguration process, assuring a good level of dependability with marginal impact on performances. The proposed methodology has a low impact on reconfiguration time, and does not affect the static portion of the FPGA, even in presence of faulty bitstream files. Experimental results demonstrated that the proposed solution is able to overcome Xilinx solution, thus representing an interesting methodology to increase FPGA dependability while reducing the DPR time and area overhead.

6. REFERENCES

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

[2] *Partial Reconfiguration User Guide*, Ug702 (v12.3) ed., Xilinx Corporation, October 2012.

[3] J. Gaisler, "A portable and fault-tolerant microprocessor based on the SPARC v8 architecture," in *Dependable Systems and Networks (DSN), 2002. International Conference on*, Jun. 23–26, 2002, pp. 409–415.

[4] Xilinx, *ML403 Evaluation Platform*, v2.5 ed., May 2006.

[5] *GRLIB IP Core Users Manual*, 1st ed., Aeroflex Gaisler, January 2012.

[6] P. Koopman, "32-bit Cyclic Redundancy Codes for internet applications," in *Dependable Systems and Networks (DSN), 2002. International Conference on*, December 2002, pp. 459–468.