A Novel Approach for Accelerating Bitstream Relocation in Many-core Partially Reconfigurable Applications

Gilberto Ochoa-Ruiz, Touiza Maamar, El-Bay Bourennane, Abderrezak Guessoum, Kamel Messaoudi, Mohamed Ali Hajjaji

A NOVEL APPROACH FOR ACCELERATING BITSTREAM RELOCATION IN MANY-CORE PARTIALLY RECONFIGURABLE APPLICATIONS

G. Ochoa-Ruiz[1], M. Touiza[12], E. Bourennane[1], A. Guessoum[2], K. Messaoudi[1], M.A.Hajjaji[1]
[1]LE2I Laboratory - Burgundy University - 21000 Dijon, France
[2]LATSI Laboratory, Blida University, BP270 Route de Soumaa, Blida, Algeria
Corresponding author: gilberto_ochoa-ruiz@etu.u-bourgogne.fr

Abstract –**Partial Bitstream Relocation (PBR) has been introduced in recent years, as a means to overcome the limitations of the traditional Xilinx Partial Reconfiguration flow, particularly in terms of the limited module placement, a fact that can greatly reduce the memory footprint of applications which require multiple implementations of the same module... However, PBR consumes scarce resources in hardware implementations, and introduces a prohibitive time overhead when done in software. This is particularly true in applications such as large scalable systems, which typically require multiple copies of the same module to accelerate a task, but in which the relocation time overhead might proof prohibitive. In order to find the best compromise between these approaches, we make use of the OORBIT tool (Offline /Online Relocation of Bitstreams) which helps us to accelerate the PBR considerably. In this paper, we compare the developed tool to others in previous works, specifically in the context of many-core applications; we give a particular importance to the reduction in the relocation time, which must to increase the time overhead already incurred by using partial reconfiguration. In this paper, we show how the tool has been used in this context, and a comparative analysis is detailed to highlight the significant relocation speedups that might help in making the relocation process more amenable.**

Index Terms—Partial Reconfiguration, Bitstream Relocation, FPGA, Embedded Systems

1. INTRODUCTION

In recent years, a great deal of research has been carried out in extending the capability of reconfigurable SoC via partial reconfiguration [1, 2]. The basic idea is that specific areas of the FPGA can be used to map hardware tasks on-demand, effectively promoting hardware tasks virtualization. These hardware tasks are stored in the form of "pre-compiled" bitstreams, usually in external memory, where can be loaded by the processor. They can be updated remotely in applications where it's impossible to access the platform, in many cases to carry out a change of the intended mission of the module. Other applications might require multiple instances of the same module, for instance, systems supporting Hot Spot Migration [3] and Fault Tolerant Systems [4]. In the case of the first two scenarios, multiples copies of the same function could potentially be necessary at any given time; in the latter cases, the location of the same module would be required to change depending on the physical conditions of the system.

However, module relocation requires being able to map a bitstream in any available reconfigurable area, which is not possible with the standard DPR design flow. Previous DPR design flows described by Xilinx [5] limit the location of the PR modules (PRMs) to predefined reconfigurable regions (PRRs). The research community has found this methodology too limiting to implement more complex systems and functionalities. Limitations in the PDR Flow are manifold, most of them arising from the fact that PRMs are restricted to specific PRRs, which increases the memory footprint necessary when multiple copies of the same module are required (e.g. highly scalable DPR systems).

In recent years, an interest on overcoming this problem has lead to the development of a series of techniques and approaches, which can be globally denominated as Partial Bitstream Relocation (PBR). By manipulating the bitstream location related information, a PRM can be mapped to different PRRs. This process can be carried out by using only hardware implementations, or by running the relocation process in a processor. However, PBR can most important aspect is to keep the relocation time overhead as low as possible, given that DPR systems suffer already of a time penalty introduced by the reconfiguration process. In this paper we introduce a comparative study carried out by using the OORBIT methodology [6], and comparing the relocation times with previous approaches, to demonstrate its advantages in the context of many-core applications.

The rest of this paper is organized as follows: in Section II we discuss the motivation for using PBR, specifically in the context of multiple cores used to build highly scalable DPR applications. Section III presents the previous works in PBR, while section IV we introduces the deployed PBR framework. In section V we discuss the used case study, and in Section VI we present a benchmark against previous approaches, focusing in the relocation time needed, and how this impacts the DPR application they are intended for. Finally, Section VII concludes the paper.

2. MOTIVATION

PBR techniques have been used in the past to be able to map the same PRM in different PRRs. The basic motivation is to reduce the on-off chip or off-chip memory storage

requirements, which is one of the downsides of using DPR techniques in general. This is achieved through the use of a module, the Bitstream Relocator (BR), as depicted in Figure 1; this module is in charge of parsing the bitstream during the reconfiguration process and modifying the information related to the bitstream placement, effectively re-mapping its functionalities to a different PRR. However, the BR must be designed in such a way that the relocation time overhead is reduced as much as possible. This typically implies the use of a hardware implementation, which can led to an increased FPGA resource utilization. In this paper, we make use of a software-based approach, we exploit the processor already required to manage the DPR process, to perform the PBR-related tasks. This is achieved through the use of a combination of an off-line partial bitstream modification, followed by the utilization of an optimized PBR approach. We look at minimizing the time overhead necessary for relocation the partial bitstreams, while keeping the resource utilization to a minimum.

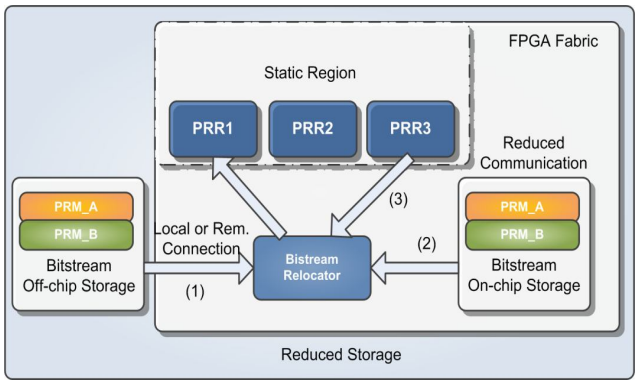

Fig 1. Bitstream Relocation Capable System

PBR techniques are better exploited in scenarios where several implementations of the same core are required. The discussion that follows aims at emphasizing the importance of reducing the relocation time as much as possible, which is the goal of this work.

Let us consider for instance applications in which functional scalability is a necessity. Some examples in the state of the art of functional scalable cores are Discrete Wavelet Transform (DWT) and the variable-size Discrete Cosine Transform (DCT). A direct approach to create variable- size scaling cores would be to implement the same task in several cores, with different performance and area requirements, and load the most suitable one in the system depending on the available hardware resources.

Other approaches overcome this problem by using highly parallel, modular and regular architectures as alternatives to reduce the overhead of the adapting process. These architectures can be scaled, in advance of the execution of a task, by means of the addition and removal of parallel blocks, resulting in lower adaptation times.

Among the suitable architectures to implement scalable solutions based on dynamic and partial reconfiguration, systolic arrays [7] are the most widely used. Systolic architectures can solve full computing-intensive tasks in a broad range of fields. For instance, authors in [8] present a scalable FPGA-based architecture for DCT computation. It achieves quality scalability by performing 2D-DCT operations for different zones, i.e., from 1×1 to 8×8. Their scalable architecture is adjusted through DPR to perform different types of DCT zonal coding. Therefore, using the traditional DPR techniques would incur in the aforementioned issues: large memory storage requirements, increased memory accesses, among others.

3. RELATED WORKS

In this section we briefly introduce the approaches proposed in the literature, discussing its advantages and drawbacks. We divide the approaches in two camps: HW and SW implementations. For a most complete discussion in the technological aspects of the PBR techniques, the reader is directed elsewhere.

In the hardware camp, the authors in [9] propose a hardware relocation module, REPLICA that modifies the bitstream while it is being downloaded from off-chip memory. BiRF [10, 11] is yet another hardware-based relocation filter that communicates to the ICAP via a custom wrapper. These hardware approaches are efficient in terms of time overhead but suffer from the fact of using additional logic resources needed for the BR, especially for calculating the Cyclic Redundancy Check (CRC) value.

On the other hand, software approaches typically use of some kind of soft-processor, taking care of the DPR management, and in addition, performing the relocation-related tasks. The approach presented in [12] transforms the relocatable bitstream on an embedded MicroBlaze processor. The same applies for the work in [13] which uses a software driver for the HWICAP core that parses the stored bitstreams, identifies and modifies the frame addresses, and relocates it to a destination PRR.

Authors in [14] have described two options of realizing this architecture on Xilinx Virtex 4 FPGAs: (a) hardware based accelerated relocation circuit (ARC) [15] and (b) a software solution executed on MicroBlaze.

All previous methodologies offer advantages or drawbacks regarding the FPGA resource utilization or the reconfiguration time overhead. In this work we perform a series on benchmarks to compare our PBR methodology with other PBR in the literature to demonstrate the feasibility of our approach; we perform this analysis in the context of large-scale systems, which require a very low PBR time overhead to produce systems that can effectively exploit DPR. We show how by using our approach, PBR can potentially be used in a broader set of scenarios.

4. DEPLOYED RELOCATION METHODOLOGY.

In order to make the relocation process less costly in terms of modification time, we make use of the OORBIT methodology (for Off-line/On-line Relocation of Bitstreams), based on a combination of off-line and on-line PBR approaches [6]. For the off-line stage, we have developed a tool that enables us, in the first place, in the analysis of the bitstreams obtained through the PR Design Flow. The configuration FAR addresses and PRRs are obtained in order to specify all the possible relocatable areas in the design. Secondly, to calculate the new FAR addresses and the CRC values for each of the obtained possible allocations. This information is subsequently added to the original bitstream in the form of an addendum. The modified bitstream supporting PBR will be utilized during the application execution by an on-line relocation module, considered as a new service of the OS and executed in an embedded processor. Therefore, the bitstream relocation consists in modifying the old FAR addresses and CRC values of the original bitstream (stored in the reconfiguration memory) with the values calculated offline which correspond to the new desired allocation.

The Bitstream Analysis is the first phase of the tool (see top part of Figure 2). It allows performing a series of functions upon the bitstream file created by the Xilinx DPR design flow. The first function accesses the configuration file and extracts from its header the several configuration parameters. The two main parameters of interest are the FPGA part to which this partial bitstream is targeted and the size of the data included in the file.

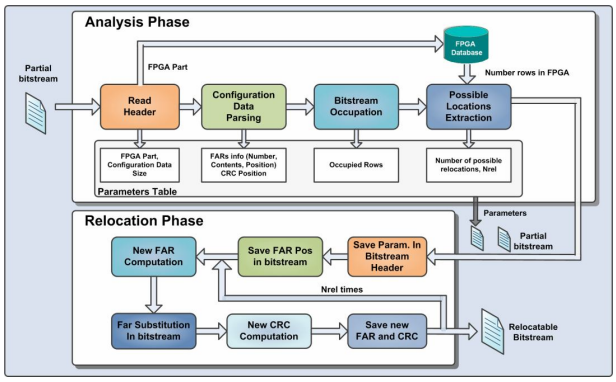

Fig 2. Bitstream Off-line bitstream relocation tool flow

The second function is bitstream Parser, which decodes all bitstream words used for configuration. It is capable of distinguishing between commands and data, and reading and writing commands; it searches for the FAR and CRC write commands in the bitstream, because the following data words have to be changed in the relocation process.

The relocation phase is in charge of creating the relocatable partial bitstream through a series of modifications. The first step is to store the parameters N_{FAR} and N_{REL} as new configuration parameters in the header of bitstream; the time parameter can be ignored without affecting the integrity of the bitstream. These two parameters are to be used in the online bitstream relocation tool to perform the PBR at-run time.

Once all new FARs registers have been computed for the first relocation, they will replace the old ones in the original bitstream at the previously identified positions. This substitution is done in order to compute the new Cyclic Redundancy Check, CRC. As relocation changes a part of the bitstream (FARs), the CRC has to be recalculated to prevent error recognition by the FPGA and allows the reconfiguration process to be successfully completed.

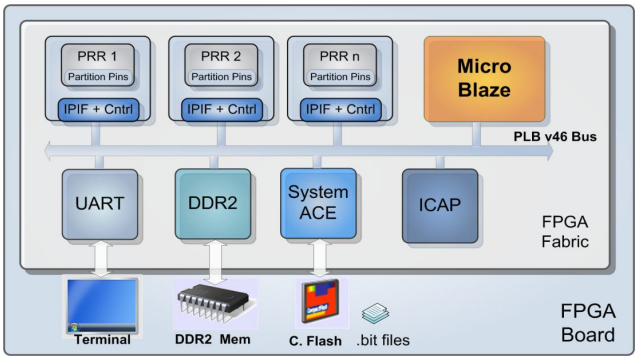

Fig 3. Framework for demonstrating the proposed PBR approach

The relocation on-line module (relocator) is a software program executed on an embedded processor (MicroBlaze), which manages the modified partial bitstream files generated by OORBIT, as depicted in Figure 3. In order to accelerate the relocation operation, an initialization stage is necessary at the beginning of the application. This operation consists in transferring all the necessary bitstreams from the non-volatile memory (i.e. FLASH or Compact Flash) to the SDRAM memory of the system. By doing so, the access to the configuration data and its manipulation becomes not only easier, but faster. During the transfer operation, the main configuration parameters are extracted from the bitstream header, specifically the number of possible relocations N_{REL}, the number of FAR addresses and the size of the configuration data chunk.

The relocation module is executed following a demand of placement; it receives as inputs the desired location for the bitstream and the relocation parameters mentioned before. The relocator reads the parameters stored at the end of the modified bitstream, which describe the positions and the information contained in the FARs and CRC registers related to the desired relocation partition. This process is less costly in terms of relocation time, which is a necessity in real applications of PBR, in which the placement of several modules can seriously increase the total reconfiguration time, and impact the system performance.

5. CASE STUDY

In this section, we present a framework for demonstrating how our PBR methodology can decrease the relocation time compared to previous approaches.. For this, we have created an on-demand DPR system. The utilized architecture (as depicted in Figure 3) is based on a MicroBlaze embedded soft-processor and it has been implemented in a series of FPGAs, Virtex 5 and 6. The processor facilitates the creation of systems in which the dynamic reconfiguration process is performed.

The static part consists of a MicroBlaze processor and the associated data and program memories, connected to a series of IP peripherals via the PLB bus. These peripherals give support to several of the tasks of the system. The System ACE module charges the configuration bitstreams and the modified partial bitstreams, which are stored in a Compact Flash memory. The DDR2 controller manages the read/write operations of an external DDR2 memory; it is in this memory where the modified partial bitstreams are recorded to speed up the modification of the FARs and CRC values, and the configuration process itself.

The partial dynamic reconfiguration component of the system is carried out by the modules in grey, namely the ICAP module and the Partial Reconfigurable Regions (PRR1 to PRRn). The ICAP module receives the partial bitstreams and uses this configuration data to modify the behaviour of the PRR by means of PR. The partial modules are separated from the static logic through the use of partition pins, as it was explained in Section 2.6; these partition pins are located in the same relative positions to perform the PBR. They don't have to be controlled by the processor as with the previously utilized bus macros, but the logic has to be reset after configuration to avoid any unexpected behaviours. This is done directly by the processor, which uses the Bus2IP reset signal after charging the partial bitstream in the defined PRR.

The architecture in Figure 3 is synthesized in the EDK. Afterwards, the design files are imported to PlanAhead, where the areas are defined for the PRRs and where the initial information of the partition pins is obtained. This information is used to relocate the partition pins and to insert the blocker macros before moving to the final MAP and PAR phases. Figure 4 depicts the organization of the PRRs modules in the left side of the FPGA. We have chosen this area of the FPFA since it contains both BRAM and DSP modules and due to the fact that the underlying resources are more homogenous. Each PRR occupies a whole FPGA row, and it has to be noted that this particular physical implementation is only for validation purposes. We have made use of placement constrains in the rest of the modules for the MAP tool to locate these functionalities in particular areas of the FPGA and to facilitate the methodology.

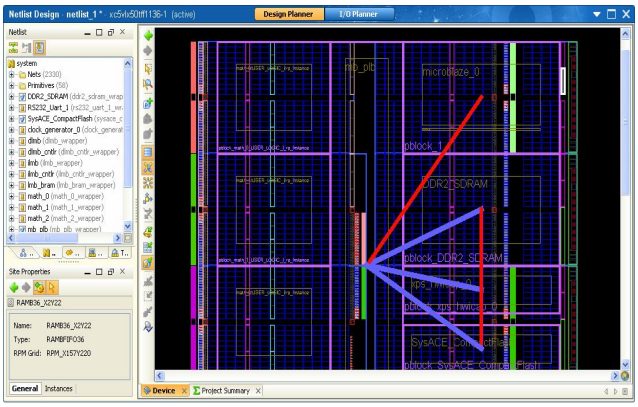

Figure 4. Physical block location for PRRs and modules in the system

6. RESULTS AND DISCUSSION.

In this section we embark in a thorough discussion of a set of experiments performed to validate the performance of our approach. This section is also intended to give more insight into the importance of certain parameters used to compare our approach to the existing solutions, in particular the relocation time. We start by providing an analytical calculation of the relocation time for multiple instances of a module, providing the minimum and maximum relocation time. Using this information, we compared our approach with previous approaches, using three modules as benchmarks. We make use of these modules to discuss how the relocation time might impact he performance of very large scalable systems, in which a module has to be mapped to multiple PRRs; this is an special case of relocation, in which minimizing the relocation time is critical.

6.1. Relocation time of our approach.

Each partial bitstream is used to configure a portion of FPGA composed by a defined number of rows. Only up to two FAR addresses are assigned to each row in the bitstream. The first address is used to access normal configuration frames while the second FAR address is specific to the BRAM contents and it is used only if BRAM are configured in the row. The time consumed by the relocation module to change the original bitstream stored in memory with another possible location in the FPGA is relatively simple to calculate. This time corresponds to the number of cycles used to modify all the FARs and CRC registers in the bitstream by those corresponding to the new location. The new values of these registers for each possible location are stored at the bottom of bitstream. In our case, we use an external DDR2 memory to store all relocatable bitstreams. The relocation time needed to process all register modifications is estimated as:

$$T_{relocation} = T_{read\_pos} + T_{red\_regs} + N_{regs} \times T_{write\_reg} \quad (1)$$

Where T_{read_pos} is the time necessary to read all the position values of the FARs and CRC registers from the bitstream, T_{read_regs} is the time needed to read the values of FARs and CRC registers for the new location, N_{regs} is the number of registers to be modified, and T_{write_reg} is the time needed to write each register in the bitstream. The transfer time needed to read/write from/to DDR2 memory in burst mode is equal to the latency time of the memory (estimated as 19 clock cycles), plus one clock cycle per word.

6.2. Benchmarks against competing approaches.

In order to emphasize the performance of our approach in terms of relocation time, we make use of the reconfigurable modules studied in [29] for the same platform. The three modules are: Discrete Cosine Transform (DCT), Discrete Wavelet Transform (DWT), and Color Space Conversion (CSC). The processor performs the relocation and writes the configuration in burst mode to HWICAP module, which transfers it simultaneously to the ICAP for reconfiguration at the speed of 100MB/s. We provide in Table 1 the resource utilization, the bitstream size, and the number of rows occupied by each of these modules in the FPGA. We compare also the times needed for each bitstream relocation between the ARC [29] method, in both two versions (SW and HW) of the BIRF approach [22] and our approach (OORBIT). We give finally in the last column the reconfiguration time of each bitstream.

Table 1. Resource usage and partial bitstream size per module.

Case	FPGA Resources (Virtex 4VLX25)				Bitstream	
	LUT	FF	DSP	BRAM	KB #	Rows Used
DCT	1419	1636	8	8	44	2
DWT	940	389	0	4	47	1
CSC	318	438	1	12	17	3

In terms of relocation time, we can observe from Table 2 an average speed-up of more than 3000x and 8000x over HW-BiRF and SW-BiRF, respectively. This great speed-up with OORBIT is due to the fact that for both HW-BiRF and SW-BiRF, an excessive time is required during relocation to recalculate CRC value. Similarly, the obtained results show an average speed-up more than 300x over ARC, which can be attributed mainly to additional time taken by this approach to read back all configuration frames before relocation.

Table 2. Relocation times benchmark for different approaches.

Case	Relocation time (ms)				Config. time (ms)
	HW BIRF [10]	SW BIRF [11]	ARC [26]	OORBIT	HWICAP
DCT	6.02	16.73	0.72	0.0015	0.44
DWT	6.43	17.87	0.40	0.0010	0.47
CSC	2.32	06.46	0.40	0.0019	0.17

Furthermore, the relocation times of bitstreams through OORBIT are almost negligible compared with their reconfiguration times; they are more than 100 times less on average. This fast relocation allows reaching the ICAP reconfiguration speed limit of 100MB/s unlike the other approaches, where their relocation process degrade the reconfiguration speed to 7.3 MB/s and 2.6 MB/s respectively for HW-BiRF and SW-BiRF and to less than 30 MB/s in average for ARC. This factor is considerably considered in many cases where performance-critical applications require fast switching of IP cores through partial reconfiguration.

Figure 5 summarizes previous discussion; it shows comparison of the different relocation approaches regarding relocation and reconfiguration times for one DCT module. The reconfiguration times remain relatively constant, whilst the relocation time varies depending on the approach, as it can be observed OORBIT surpasses the other approaches performance.

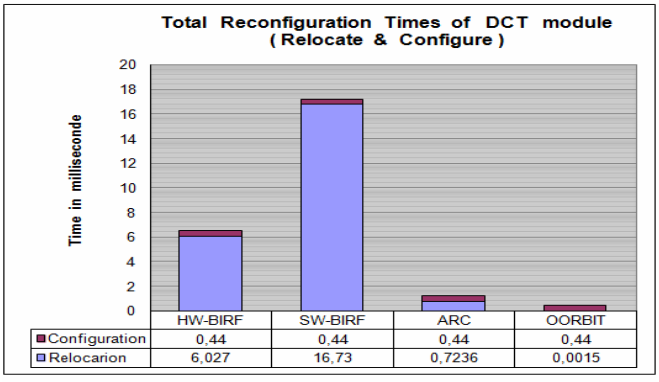

Figure5. Relocation plus configuration time for one DCT module

6.3. Discussion on very large scalable systems.

If we take the implementations for the DCT provided in Table 3, we will have memory requirements, only for DCT bitstreams storage, of 44KB x 8 PRRs. The total reconfiguration time for the 8 PRR would be 3.52ms.

Using PBR the total storage would represent only 44KB plus some words added by our approach; however, the relocation time for each of the bitstreams has to be keep as low as possible to minimize the degradation of the QoS of the application, as explained before.

The objective of all the PRB relocation methodologies is therefore reducing the time overhead incurred during the relocation process for each of the partial bitstreams to be mapped in the FPGA. If we take the comparative data in Table 2 and use in the context of the previous discussion, we can compute the total time needed to map the PRM to the 8 PRRs. The results for one DCT are depicted in Figure 6, where it can be observed that the proposed approach clearly surpasses the previous approaches.

Furthermore, we can observe that both versions of BIRF surpass by a large amount the configuration time for the 8 modules (48.24ms and 133.84ms for HW and SW versions, respectively). ARC performs slightly better in this regard, but introduces a time overhead the doubles of the configuration time for each module and deals only with configuration through ICAP read-back capture. In many dynamic partial reconfiguration applications, the configuration time itself might be considered as a penalty; therefore, introducing additional time is prohibitive. We can see that OORBIT introduces a very low time overhead, almost negligible. One of the novelties of the approach is that this is achieved without introducing any extra hardware resources, which is an important aspect in reconfigurable systems using DPR.

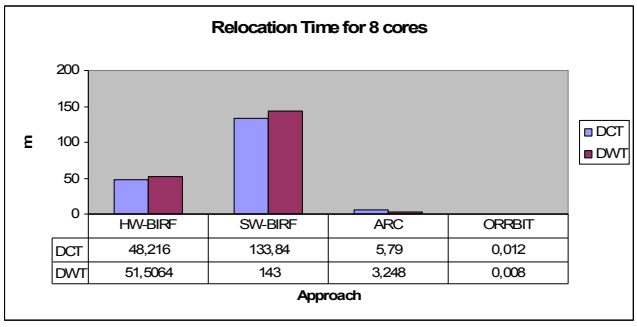

Figure 6. Relocation time for the mapping into 8 PRRs of the same module

7. CONCLUSIONS.

In this paper, we have presented a comparative approach between several implementations of the Partial Bitstream Relocation approach. The aim of the paper was to introduce the technique in the context of large-scalable systems that might take advantage of Dynamic Partial Reconfiguration. In such a scenario, multiple copies of the same module implementation are usually, and their mapping to different reconfigurable areas directly leads to the use of PBR techniques. However, to make any practical implementation in this context, the relocation time overhead has to been greatly minimized. This is especially true considering that a major factor in avoiding the adoption of DPR in many applications is due to the penalty incurred by the reconfiguration time overhead.

Using OORBIT, the relocation process is limited to a few substitution operations in the bitstream, up to 20 words in total. The relocation time is equivalent to the time necessary to read these words from a location in memory and to write them back in predefined locations in the bitstream. We have shown how our method significantly decreases the relocation time when compared to previous approaches. Furthermore, we have performed several benchmarks using highly scalable image processing IPs (DCT and DWT) to make clear how an improved relocator can impact in the overall reconfiguration time and its applicability in real-time embedded systems.

8. ACKNOWLEDGMENTS

This work has been supported by the ANR FAMOUS Project (ANR-09-SEGI-003) by the Agence Nationale de la Recherche.

8. REFERENCES.

[1] J. Hagemeyer, B. Kettelhoit, M. Koester, and M. Porrmann, Design of homogeneous communication infrastructures for partially reconfigurable FPGAs, in Proc of the International Conference on Engineering of Reconfigurable Systems and Algorithms ERSA '07, (CSREA Press, 2007).

[2] A. Oetken, S. Wildermann, J. Teich, D. Koch, A Bus-based SoC Architecture for Flexible Module Placement on FPGAs, (FPL 2010).

[3] A. Gupte, P. Jones, Hotspot Mitigation Using Dynamic Partial Reconfiguration for Improved Performance, ReConFig 2009, 89-94.

[4] D. Montminy, R. Baldwin, P. Williams, and B. Mullins, Using relocatable bitstreams for fault tolerance, Adaptive Hardware and Systems, Second NASA/ESA Conference (Aug. 2007) 701–708.

[5] Xilinx, Partial Reconfiguration User Guide, (Xilinx UG208, 2011).

[6] M. Touiza et al., A novel methodology for accelerating bitstream relocation in partially reconfigurable systems, Micpro, Springer, 2012.

[71] J.Huang, J.Lee,Y imin Ge, "An array-based scalable architecture for DCT computations in video coding,", June 2008

[8] J. Huang, M. Parris, J. Lee, and R. F. DeMara, "Scalable FPGA Architecture for DCT Computation using Dynamic Partial Reconfiguration", ERSA, 2008.

[9] H. Kalte, G. Lee, M. Porrmann, and U. Ruckert, REPLICA : A bitstream manipulation filter for module relocation in partial reconfigurable systems, in Proc of Parallel and Distributed Processing, 2005.

[10] S. Corbetta, F. Ferrandi, M. Morandi, M. Novati, M. D. Santambrogio, and D. Sciuto, Two novel approaches to online partial bitstream relocation in a dynamically reconfigurable system, VLSI, 2007.

[11] S. Corbetta, M. Morandi, M. Novati, M. Santambrogio, D. Sciuto, and P. Spoletini, Internal and external bitstream relocation for partial dynamic reconfiguration, VLSI, 2009

[12] J. Carver, R. Pittman, and A. Forin, Relocation and Automatic Floorplanning of FPGA Partial Reconfiguration Bitstreams, Microsoft Research, WA, Technical Report no.MSR-TR-2008-111 (Aug 2008).

[13] T. Becker, W. Luk, and P. Cheung, Enhancing Relocatability of Partial Bitstreams for Run-time Reconfiguration, Field Programmable Custom Computing Machines, (15th Annual IEEE Symposium, Apr 2007) 35–44.

[14] A. Sudarsanam, R. Kallam, and A. Dasu, PRR-PRR Dynamic Rellocation, IEEE Computer Architecture Letters, Jul-Dec 2009.

[15] R. Kallam, A. Sudarsanam, and A. Dasu, Accelerated Relocation Circuit, (IET Electronics Letters, 2009)