Towards Utilizing Reconfigurable Shared Resources in Multi-Core Hard Real-Time Systems

Luca Pezzarossa
lpez@dtu.dk

Martin Schoeberl
masca@dtu.dk

Jens Sparsø
jspa@dtu.dk

Department of Applied Mathematics and Computer Science
Technical University of Denmark
Kongens Lyngby, Denmark

ABSTRACT

Dynamic partial reconfiguration (DPR) of FPGAs allows the reconfiguration of selected areas of an FPGA after its initial configuration, while the remaining part of the system continues to operate without interruption. Hard real-time systems are a class of systems whose temporal behavior has to be completely predictable. Our research explores the use of DPR of FPGAs in the context of hard real-time multi-processor systems for embedded applications, targeting the T-CREST multi-core platform. This paper provides an overview and discusses the challenges related to the use of DPR to share reconfigurable resource in a time-predictable manner. It presents an approach to the application of DPR in multi-core hard real-time systems and proposes two models to describe the effect of DPR on the software tasks execution and scheduling. Related works, plans for future evaluation and a preliminary test are also presented.

1. INTRODUCTION

Dynamic partial reconfiguration (DPR) is an emerging concept in the FPGA industry that allows the reconfiguration of portions of the FPGA, while the rest of the device continues to operate without interruption [1, 2, 3].

Hard real-time embedded systems are a class of systems characterized by strict timing constraints on the execution time of the tasks. These systems are used for safety-critical applications where a failure to respond in time may lead to catastrophic consequences (e.g., flight electronics, wind turbine control systems, medical devices, factory automation systems, etc.). The design process of such systems, addressing multi-core architectures instead of single processor architectures, is more complicated and challenging due to the fact that the temporal behavior has to be completely predictable and analyzable.

For multi-core hard real-time embedded systems, which are mainly used in professional or high-end applications, the development and the fabrication cost of an ASIC is typically prohibitive, since it is not possible to amortize the development costs over the production volume. Therefore, FPGAs usage is preferable for this class of applications. FPGAs are less efficient than ASICs, but this can be compensated for by shorter development time and increased flexibility. DPR brings this flexibility even further by enabling run-time changes in the hardware.

Our research investigates the usage of FPGAs' DPR in the context of multi-core hard real-time systems. It targets the existing platform T-CREST [4] and aims to supplement it with the DPR feature. T-CREST is a homogeneous multi-core platform for embedded hard real-time applications especially optimized to simplify static worst-case execution time (WCET) analysis. Our hypothesis is that DPR can provide substantial benefits by allowing dynamic hardware modifications. More specifically, we hypothesize that a system which uses a DPR approach can be more efficient in terms of size, power consumption and cost than an equivalent static version, while maintaining comparable computational performance.

This paper provides an overview and discusses the challenges related to the use of DPR to share reconfigurable resources between software tasks in a time-predictable manner. More specifically, it contributes (i) by presenting an approach to the application of DPR in multi-core hard real-time systems and (ii) by proposing two models to describe the effect of DPR on the tasks execution and scheduling. Related works, plans for future evaluation and a preliminary test are also presented.

This paper is organized as follows: Section 2 provides general background about DPR by presenting the state-of-the-art, the potential benefits and the limitations of the current FPGA technology. Section 3 presents a classification of DPR with the relative configuration latencies and proposes two formulated models for DPR feature in hard real-time systems. Section 4 describes our plans for evaluation and a preliminary test utilizing the T-CREST platform. Section 5 briefly presents related works and finally Section 6 concludes the paper.

2. DYNAMIC PARTIAL RECONFIGURATION OF FPGAS

Dynamic partial reconfiguration (DPR) allows the modification of an operating FPGA design by loading a partial configuration file (bit-file), while the remaining part of the system continues to operate without interruption. After the initial configuration of the FPGA, partial bit-files can be loaded into the FPGA to modify selected regions, without compromising the integrity and the functionality of those parts of the device that are not being affected by the reconfiguration.

Therefore, a system that uses DPR can be conceptually considered as divided in two main parts: a static part and a dynamic part. The static part is configured only once at boot-time with a full bit-file. The dynamic part, which may consist of several independent reconfigurable regions, can be reconfigured multiple times during run-time with different

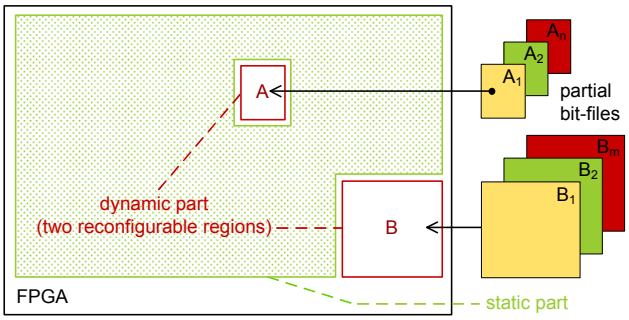


Figure 1: A FPGA divided into a static and a dynamic part (two reconfigurable regions).

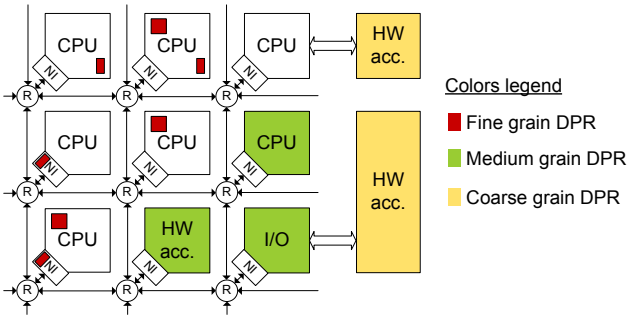


Figure 2: Example of the three DPR classes in a network-on-chip-based multi-core platform.

partial bit-files. Figure 1 shows a FPGA divided into a static part and a dynamic part, where the dynamic part consists of two non-topologically-connected reconfigurable regions (A and B). For every region, a partial bit-file can be loaded from a set (e.g., A_1, A_2, ..., A_n) without interfering with the functionality of the static part.

The use of DPR allows the creation of systems with a very high level of flexibility since reconfigurable regions can be dynamically reused by realizing the functionality that is needed at any point in time. This makes more efficient usage of the FPGA hardware resources, leading to a reduction of the FPGA size, with consequent reduction of power consumption and cost. Moreover, a blank (empty) partial bit-file can be loaded into a temporarily not used reconfigurable region in order to reduce the power consumption to the minimum.

Using DPR to obtain these benefits increases the complexity of the design process and of the hardware design itself. The design flow performed with the commercial tools is more articulated and it requires additional steps to generate the partial bit-files. For instance, the definition of the physical areas on the FPGA chip where the reconfigurable regions must be placed and the specification of dedicated timing constrains for the reconfigurable regions. Something that is not required for standard non-reconfigurable design.

From a hardware point of view, additional logic is needed in the borders between the static part and the dynamic part in order to insulate the reconfigurable regions during reconfiguration. Moreover, a hardware controller is also required to manage the reconfiguration process and to move the partial bit-file from the memory where they are stored (on-chip block-RAM or off-chip memory) to the FPGA's configuration memory.

The next section addresses this design challenges keeping in mind that, since we target multi-core platform for hard real-time systems, the time behavior has to be completely predictable. Therefore, also the DPR approach must be performed in a time-predictable manner and be completely analyzable.

3. APPROACH AND MODELS

The main idea is to share the dynamic part of the FPGA between the resources used by one or more processors of the platform for a limited period of time (e.g., hardware accelerators, co-processors, I/O units, etc.) or to reconfigure part of the platform (e.g., processors or sections of them, networks-on-chip, etc.) to dynamically adapt the hardware to the actual needs of the software tasks running on it.

In this section we address how to perform this. First we identify 3 classes of DPR, then we present the interfaces available for invoking DPR, we discuss latency aspects of these and finally we propose two models for the DPR feature in hard real-time system.

3.1 Classes of DPR

In a multi-core platform, such as T-CREST, we have identified three possible classes of DPR depending on the size of the reconfigured area: fine grain, medium grain, and coarse grain DPR. An example of the DPR classes is shown in Figure 2, where the three granularities are represented in different colors.

We define the DPR as fine grain when the area to be reconfigured is very small, in the order of hundreds of FPGA logic cells (LCs). An example of fine grain reconfiguration is the application of minor changes to a CPU architecture in order to modify, during run-time, its instruction set (shown in red in Figure 2).

A medium grain DPR involves an area in the order of thousands of LCs. An example is a reconfiguration of an entire intellectual property of the platform, such as CPUs, small stateless hardware accelerators, etc. (shown in green in Figure 2).

Finally, a coarse grain DPR involves a large area of the FPGA, in the order of tenths of thousands of LCs. An example of this class is a set of stateful hardware accelerators for compute-intensive operations (fast Fourier transform, encryption/decryption, etc.) to be swapped into large reconfigurable regions and connected to a subset of CPUs with a dedicated network-on-chip or a shared bus (shown in yellow in Figure 2).

Since the reconfiguration time depends on the amount of LCs to be reconfigured, it is immediately clear that a fine grain DPR is very fast. On the contrary, a coarse grain DPR is a relatively slow process.

3.2 Interfaces and Reconfiguration Latencies

For XILINX FPGAs, DPR can be performed through different interfaces. In this paper we mention only the two interfaces that we have considered to use: the internal configuration access port (ICAP) and the SelectMap interface.

The ICAP is a primitive found in XILINX FPGAs. It is an internal interface on the FPGA fabric that can be accessed by the hardware implemented on the FPGA itself and it provides direct access to the configuration memory.

Table 1: Calculated reconfiguration latencies for the three classes of DPR for a XILINX Virtex-6 FPGA.

Class of DPR	# of CLBs	Bit-file size	Reconfig. latency
Fine grain	150	45 kB	110 μs
Medium grain	2 000	680 kB	1.5 ms
Coarse grain	10 000	2.9 MB	7.3 ms

It requires the instantiation of an ICAP controller and the logic to drive the interface itself.

The SelectMap interface is a fast external reconfiguration interface that also provides direct access to the configuration memory. It dedicates I/O pins for a bi-directional data bus and control signals.

For a XILINX Virtex-6 FPGA, both interfaces have a maximum data width of 32 bits and a maximum frequency of 100 MHz, hence the maximum transfer speed that can be reached to load a partial bit-file is 3.2 Gb/s. Table 1 shows some calculated values of reconfiguration latencies for the three classes of DPR, based on the aforementioned transfer speed. CLB stands for configurable logic block of XILINX FPGAs. The reconfiguration latency is the time interval needed to load a partial bit-file in a reconfigurable region.

3.3 DPR Models

Considering the granularity of the DPR and the associated configuration latencies we have formulated two different models to describe the effect of DPR on the tasks execution and schedule: a task-level DPR model and a mode-level DPR model.

Task-Level DPR Model

Observing the reconfiguration latencies shown in Table 1 we can safely assume that the latency for fine grain DPR is smaller than (or comparable to) the WCET of a software task (maximum possible duration of a task). Therefore, the effect of DPR can be modelled by associating the reconfiguration latency to the tasks to which the reconfiguration is related. In other words, the tasks that uses reconfigurable resources need to include the reconfiguration latency in their WCET before the task set is scheduled.

As an example, Figure 4(a) shows a static schedule period with three tasks T_1, T_2 and T_3 sharing a reconfigurable region. Every time a task is started or resumed, it needs to reconfigure the dynamic part in order to have available the hardware resources to perform its operation. The solid color at the beginning of each task represents the reconfiguration delay added to each task WCET. The last row of the diagram shows how the reconfigurable region is shared between the tasks.

Mode-Level DPR Model

Assuming the reconfiguration latency for coarse grain DPR larger than the WCET of a software task, the reconfiguration of this DPR class must be associated to an operation mode change, where the system, during normal operation, changes a subset of the executing tasks to adapt its behavior to new environment conditions.

The graph in Figure 3 shows the operational mode changes between three modes: M_1, M_2, and M_3. Every mode consists of a set of tasks and a set of resources (e.g., hardware accelerators) implemented on the dynamic part of the design that corresponds to a different operational scenario.

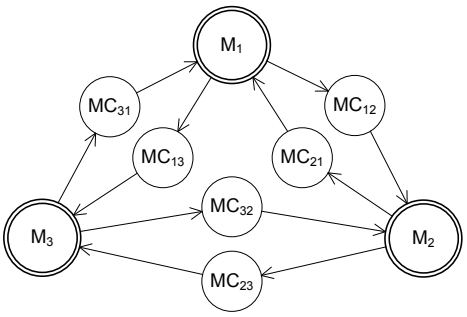


Figure 3: Graph showing the operational mode changes for three modes.

A reconfiguration associated with a mode change can be modelled as a task that belongs to a mode change scenario (e.g., MC_{12}, MC_{21}, etc. in Figure 3). A mode change scenario consists of a set of task that need to be maintained active in the transition between the old and the new mode and a set of tasks that models the reconfiguration process.

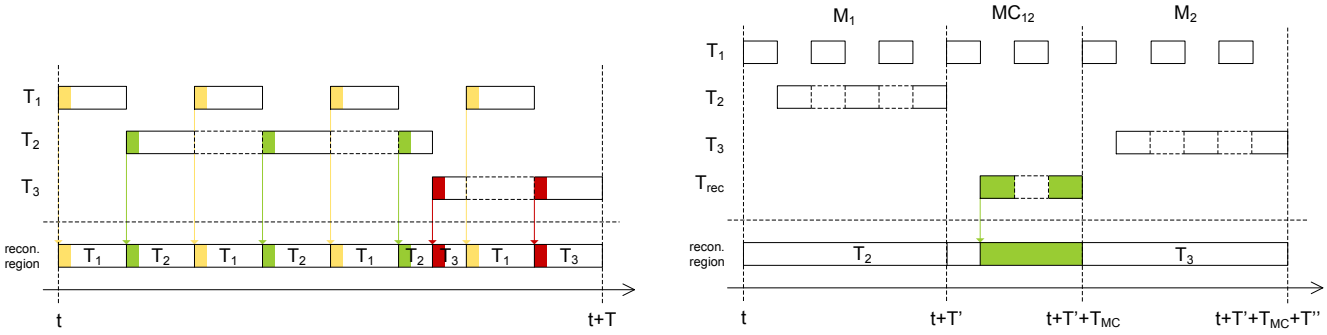
As an example, Figure 4(b) shows a change between two statically scheduled modes M_1 and M_2. M_1 consists of the tasks T_1 and T_2, while M_2 consists of the tasks T_1 and T_3. We assume that T_2 and T_3 need different resources to be implemented on the shared reconfigurable region and that the periodic execution of T_1 cannot be suspended during the mode change. We also assume that a task that continues its execution through the mode change cannot reconfigure its own resources. We can observe that during the mode change scenario MC_{12} the task T_1 continues to run and the task T_{rec}, which models the reconfiguration process, is executed. The last row of the diagram shows how the reconfigurable region usage changes between different modes. Also in this case, the solid color represents the reconfiguration delay.

3.4 Final Remarks

The two models presented above, which are applicable to different classes of DPR, are not exclusive. Fine, medium, and coarse grain DPR can be present in the same architecture and be modelled independently using the proper model. However, since the current FPGA technology allows to reconfigure only one region at a time, interference between tasks can occur and this must be taken into account during the task scheduling. Utilizing the presented models to describe the effects of the DPR makes still possible to apply the traditional shared resources scheduling protocol (e.g., PIP, PCP, SRP, etc.) to properly share the reconfigurable regions between the tasks. Finally, we mention that medium grain DPR can be modelled with both methods depending on the actual relation between the task WCET and the reconfiguration latency. If the reconfiguration latency is smaller or comparable than the task WCET, the task-level model can be applied. Otherwise, the mode-level DPR model needs to be used.

4. EVALUATION

We plan to evaluate our approach, models and design utilizing the T-CREST platform. T-CREST is a homogeneous multi-core platform for embedded hard real-time applica-



(a) Task-level DPR model.

(b) Mode-level DPR model.

Figure 4: Example of time diagrams for the task-level and the mode-level DPR models.

tions [4]. All features are optimized to simplify static WCET analysis. The T-CREST platform contains several processing cores, called Patmos [5], and it is supported by the WCET analysis tool aiT [6] from AbsInt that allows to statically derive tight WCET bounds. T-CREST contains two time-predictable networks-on-chip (NOCs): a time-division multiplexing message passing NOC between the cores, called Argo [7], and a memory tree NOC towards the shared external main memory.

We are currently developing an ICAP/SelectMap controller that allows to load partial bit-files in a time-predictable manner. Preliminary experiments have been carried out targeting the XILINX Virtex-6 FPGA on the ML605 development board. The tested architecture consists of a single Patmos processor that manage a fine grain reconfigurable region, using the ICAP/SelectMap controller to swap a simple I/O led driver between different configurations. The test allowed to prove the functionality of the controller and to collect some preliminary measurements regarding the reconfiguration latency. With a size of 40 configurable logic blocks, and a bit-file size of 12.7 kB stored in block-RAM, the measured reconfiguration latency is 35 μs.

Further evaluation is expected in the near future when the ICAP/SelectMap controller and the software tools currently under development will be completed and stable.

5. RELATED WORKS

The use of DPR in the context of hard real-time systems and of multi-processor platforms is largely unexplored. However, we briefly mention two works addressing general purpose non-real-time platforms that have been taken into account in our research.

The ReCoBus-builder [8] is a FPGA design oriented framework for component-based reconfigurable systems. It uses DPR to generate systems with one or more reconfigurable areas to be used by different hardware modules.

The LogiCORE IP XPS HWICAP [9] is an IP from XILINX that enables an embedded microprocessor to read and write the FPGA configuration memory through the ICAP interface. This controller, although being widely used, is not designed to be time-predictable. Hence, it is not suitable to be directly used in our design.

6. CONCLUSION

This paper presented a preliminary exploration of the use of FPGAs' DPR in the context of multi-core hard real-time systems. It discussed the challenges related to the use of DPR and it presented an approach on how to use the DPR feature to share reconfigurable resource in a time predictable manner. The paper also proposed two models to describe the effect of DPR on the tasks execution and scheduling.

7. REFERENCES

[1] L. Wang and F. Y. Wu. Dynamic partial reconfiguration in FPGAs. In *Proc. of IEEE Third International Symposium on Intelligent Information Technology Application*, volume 2, pages 445–448, 2009.

[2] XILINX. UG702: Partial reconfiguration user guide. Technical report, 2012. Online.

[3] ALTERA Corporation. QII51026: Design planning for partial reconfiguration. Technical report, 2013. Online.

[4] M. Schoeberl et al. T-CREST: Time-predictable multi-core architecture for embedded systems. *Journal of Systems Architecture*, 2015. Accepted for pubblication. Online at *http://www.jopdesign.com/doc/t-crest-jnl.pdf*.

[5] M. Schoeberl, P. Schleuniger, W. Puffitsch, F. Brandner, C. W. Probst, S.Karlsson, and T. Thorn. Towards a time-predictable dual-issue microprocessor: the Patmos approach. In *Proc. of First Workshop on Bringing Theory to Practice: Predictability and Performance in Embedded Systems*, pages 11–20, 2011.

[6] R. Heckmann and C. Ferdinand. Worst-case execution time prediction by static program analysis. Technical report. Online at *http://www.absint.de/aiT_WCET.pdf*.

[7] E. Kasapaki et al. Argo: A real-time network-on-chip architecture with an efficient GALS implementation. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 2015. Accepted for pubblication. Online at *http://www.jopdesign.com/doc/argo-jnl.pdf*.

[8] D. Koch, C. Beckhoff, and J. Teich. ReCoBus-Builder - a novel tool and technique to build statically and dynamically reconfigurable systems for FPGAs. In *Proc. of International Conference on Field Programmable Logic and Applications*, pages 4629918, 119–124. Inst. of Elec. and Elec. Eng. Computer Society, 2008.

[9] XILINX. DS586: LogiCORE IP XPS HWICAP (v5.00a) product specifications. Technical report, 2010. Online.