

Application of Task Concurrency Management on Dynamically Reconfigurable Hardware Platforms

Javier Resano¹, Diederik Verkest^{2,3,4}, Daniel Mozos¹, Serge Vernalde², Francky Catthoor^{2,4}
javier1@fdi.ucm.es*

Abstract

Dynamically Reconfigurable Hardware (DRHW) can take advantage of its reconfiguration capability to adapt at run-time its performance and its power consumption. However, due to the lack of programming support for dynamic task placement on these platforms, no previous work has been presented studying the performance/power trade-offs. To cope with the task placement problem in a straight way that allows us to go one step further, we have adopted an interconnection-network-based DRHW mode, which includes Operating System support to reallocate tasks at run-time. On top of this model we have applied an emerging task concurrency management (TCM) methodology initially developed for multiprocessor platforms with promising results. Moreover, we have identified the next step needed to create a specific TCM support for DRHW platforms.

1. Introduction

Dynamically Reconfigurable Hardware (DRHW), that allows partial reconfiguration at run-time, represents a powerful and flexible way to deal with the dynamism of current multimedia applications. However, compared with application specific integrated circuits (ASICs), DRHW systems are less power efficient. Since power consumption is one of the most important design concerns, this problem must be addressed at every possible level; thus, we propose to use a task concurrency management (TCM) approach, especially designed to deal with current dynamic multimedia applications, which attempts to reduce the energy consumption at task-level.

Previously, other research groups have addressed the power consumption of FPGAs, proposing a technique to allocate configurations [1], presenting an

energy-conscious architectural exploration [2], introducing a methodology to decrease the voltage requirements [3], or carrying out a static scheduling [4]. However, all these approaches are applied at design-time, so they cannot tackle efficiently concurrent dynamic applications, whereas our approach can select between different power/performance trade-offs at run-time.

2. Mapping methodology

The Interconnection-Network (ICN) DRHW model [5] partitions an FPGA platform into an array of identical tiles. At run-time tasks are assigned to these tiles using partial dynamic reconfiguration. However, typically the task interfaces differ, so the system has to perform a new Place&Route (P&R) of the whole FPGA after each assignment, which represents an unaffordable overhead. Hence, in order to allow partial reconfiguration at run-time, our ICN model proposes to use a fixed communication interface for all the tasks. Thus, when a new task is assigned to a tile, this task shares the same interface as the previous one, avoiding a new P&R process. Also, the communication interfaces contains some Operating System (OS) support like storage space and routing tables, which allow run-time task migration. Communications between different tiles are carried out over a soft packet-switched ICN implemented using the FPGA fabric.

Applying the ICN model to a DRHW platform greatly simplifies the dynamic task allocation problem, providing a software-like approach, where tasks can be assigned to HW resources in the same way that threads are assigned to processors. Thus, this model enables the use of the emerging Matador TCM methodology [6]. TCM proposes a task scheduling technique for heterogeneous multiprocessor embedded systems. It

* This work has been partially supported by TIC-2002-00160

1. Universidad Complutense de Madrid

3. also professor at Katholieke Universiteit Leuven., Belgium

2. IMEC vzw, Kapeldreef 75, 3001, Leuven, Belgium

4. also professor Vrije Universiteit Brussel, Belgium

starts from an application specification, composed of one or several tasks, which are called Thread Frames (TF). The behavior between these TFs can be dynamic and even non-deterministic. Each TF is represented as a Control-Data Flow Graph with support for data-dependent conditions and while-loops. Each node of the graph is called Thread Node (TN). TCM accomplishes the scheduling in two phases. The first phase generates at design-time a set of optimal scheduling solutions for each TF called a Pareto curve. Each solution represents a scheduling and an assignment of the TNs over the available processors. Thus, each point of the Pareto curve represents a different performance/energy tradeoff [Fig. 1]. Whereas this first step accomplishes the design-space exploration for each TF standing alone, the second phase tackles their run-time behavior, selecting at run-time for each executing TF its most suitable Pareto point. The goal of the methodology is to minimize the cost (in this case the energy consumption of the system) while meeting the timing constraints of the running applications (typically, highly-dynamic multimedia applications).

3. Practical Demonstration and Conclusions

As a first approach towards the integration on a DRHW platform of the two aforementioned ideas, we have developed a practical example, using a motion JPEG decoder. This application does not have enough dynamism to take full advantage of all the TCM features. However, we believe that it is interesting enough to illustrate this approach.

We use a platform composed of a Strong ARM SA-1110 processor, coupled with a Virtex2 v6000 FPGA. The FPGA contains four tiles and a soft ICN that includes the interface between the processor and the FPGA. Further details about this ICN are described in [5]. Once the ICN and the communication interfaces are implemented, the FPGA can be considered as a multiprocessor system, where HW tasks are assigned to the FPGA tiles, in the same way that threads are assigned to processors. Thus, the TCM methodology can be easily applied.

At design time, TCM must build a Pareto curve for each task. To this end, all the TN inside the task must be labelled with the execution time and energy consumption for each possible platform. In order to obtain the execution time, we model the application using OCAPI XL [7] and we simulate its performance both for the SA processor and for the FPGA tiles. For the energy consumption we use the Virtex2 power estimator, and the datasheet of the SA processor. With

these estimations the existing design-time scheduler generates the Pareto curve [Fig. 1]. With this curve, the run-time scheduler can select at run-time different energy/performance trade-offs. Thus, if there is not a tight timing constraint, it will select the least energy consuming solution, whereas, if the timing constraint changes (e.g. the system must share the resources with other applications), the run-time scheduler will look for a faster solution which meets the new constraint (with the consequent energy penalty).

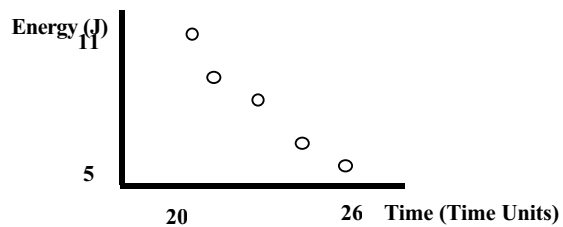


Figure 1. Pareto curve of the motion JPEG task.

Ideally, in this example TCM can achieve up to 55% energy savings, when it is compared to a static approach that implements in the same platform the solution with the highest performance. Hence, we believe that applying TCM to DRHW platforms can drastically reduce the overall energy consumption. However, the current TCM scheduler does not take into account the HW reconfiguration overhead. In this experiment we have observed that when this overhead is added, the shape of the Pareto curves can greatly change. Thus, if it is neglected, a decision that attempts to reduce the energy can actually increase it. Hence, we need to extend current TCM scheduling tools to include the reconfiguration overhead. We intend to carefully study this topic in the near future.

4. References

- [1] R. Maestre et al., "Configuration Management in Multi-Context Reconfigurable Systems for Simultaneous Performance and Power Optimizations", Proc. of ISSS'00, pp 107-113, 2000.
- [2] M. Wan et al. "Design Methodology of a Low-Energy Reconfigurable Single-Chip DSP System", Journal of VLSI Signal Processing 28, pp. 47-61, 2001.
- [3] A. D. Garcia et al., "Reducing the power Consumption in FPGAs with keeping a high Performance Level", WVLSI00, pp 47-52, 2002
- [4] Li Shang and N. K. Jha, "Hardware-Software Co-synthesis of Low Power Real-Time Distributed Embedded Systems with Dynamically Reconfigurable FPGAs", Proc. of ASP-DAC/VLSI Design'02, pp. 345-360, 2002.
- [5] T. Marescaux et al., "Interconnection Network enable Fine-Grain Dynamic Multi-Tasking on FPGAs", FPL'02, pp. 795-805, 2002.
- [6] P. Yang et al., "Energy-Aware Runtime Scheduling for Embedded-Multiprocessors SOCs", IEEE Journal on Design&Test of Computers, pp. 46-58, 2001.
- [7] G. Vanmeerbeeck, et al., "Hardware/Software Partitioning for Embedded Systems in OCAPI-XL", Proc. of CODES'01, 2001.