HARDWARE/SOFTWARE PLATFORM FOR SELF-AWARE COMPUTE NODES

Markus Happe, Andreas Agne, Christian Plessl, Marco Platzner

University of Paderborn, Germany
email: {markus.happe, agne, christian.plessl, platzner}@uni-paderborn.de

1. INTRODUCTION

Today's design and operation principles and methods do not scale well with future reconfigurable computing systems due to an increased complexity in system architectures and applications, run-time dynamics and corresponding requirements. Hence, novel design and operation principles and methods are needed that possibly break drastically with the static ones we have built into our systems and the fixed abstraction layers we have cherished over the last decades. Thus, we propose a HW/SW platform that collects and maintains information about its state and progress which enables the system to reason about its behavior (self-awareness) and utilizes its knowledge to effectively and autonomously adapt its behavior to changing requirements (self-expression).

To enable self-awareness, our compute nodes collect information using a variety of sensors, i.e. performance counters and thermal diodes, and use internal self-awareness models that process these information. For self-awareness, online learning is crucial such that the node learns and continuously updates its models at run-time to react to changing conditions. To enable self-expression, we break with the classic design-time abstraction layers of hardware, operating system and software. In contrast, our system is able to vertically migrate functionalities between the layers at run-time to exploit trade-offs between abstraction and optimization.

This paper presents a heterogeneous multi-core architecture, that enables self-awareness and self-expression, an operating system for our proposed hardware/software platform and a novel self-expression method.

2. SELF-AWARE MULTI-CORE ARCHITECTURE

As architecture we propose a heterogeneous multi-core that consists of processors (that execute software threads) and reconfigurable hardware cores (that execute hardware threads). As prototyping platform, we use the programming model and execution environment ReconOS [1]. ReconOS extends the well-known multithreading approach to reconfigurable hardware. Here, hardware threads can access the same shared resources (i.e. shared memory, synchronization and communication primitives) like the software threads.

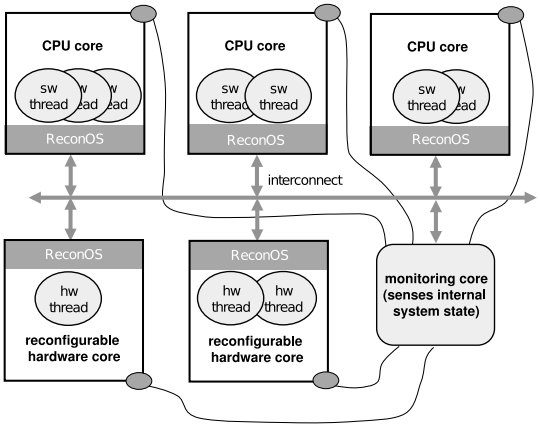

Fig. 1. Proposed heterogeneous multi-core architecture [2].

Figure 1 depicts an example architecture that consists of three processors and two (reconfigurable) hardware cores. The system contains a monitoring core that captures core-specific information. We currently support ring-oscillator based thermal sensors to capture the on-chip temperature distribution and performance counters to measure the system's performance.

3. OPERATING SYSTEM

ReconOS extends current operating systems (OS), i.e., Linux or eCos, to support reconfigurable hardware. In ReconOS, hardware threads are represented by delegate threads in software. Whenever a hardware thread makes an OS call, an interrupt is generated and the function name and its parameters are forwarded to delegate software thread. The delegate thread makes the OS call on behalf of the hardware thread and returns the results to the calling hardware thread.

Figure 2 gives an overview of the ReconOS system where the software threads interact directly with the OS kernel, while the hardware threads in the FPGA's logic are connected through operating system interfaces (OSIFs) and delegate threads. The operating system runs on one (main) processor while the other processors (workers) only execute software threads. The software on the worker processors are

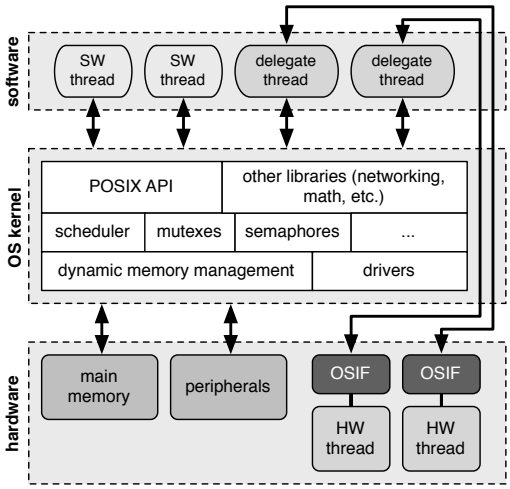

Fig. 2. Conceptual overview of the ReconOS system [1].

also represented by delegate threads on the main processor.

In extension to our previous work on ReconOS, we introduce vertical function migration as a novel self-expression method where the system can migrate threads between the software, OS and hardware layers at run-time to affect various aspects of the system such as performance, power consumption, temperature etc.

4. SELF-EXPRESSION BY THREAD MIGRATION

Migrating a thread from the main processor to a hardware core affects the performance, overall power consumption and the thermal profile of the chip. Considering heterogeneous processors similar effects can be observed for the migration between processors. Migrating a thread into the OS kernel is a special case which can lead to an improved performance if the operating system distinguishes between the a kernel space and a user space (such as Linux). In contrast to a user thread, a kernel thread can avoid internal context switches because it can directly access OS objects which results in an improved performance. For thread migration between hardware cores and between the hardware/software boundary we propose cooperative multitasking [3] where the threads have well-defined migration points and inform the OS every time they reach these points. Resuming execution form these migration points should be possible for both, the hardware and the software thread.

In [4] we demonstrated that a heterogeneous multi-core can regulate an application's performance by adding and removing software and/or hardware threads at run-time for a particle filter-based video object tracker. The performance of the application varies when a tracked object moves into the foreground or the background because this influences the computational complexity of the histogram calculation. The system measured the execution time of the individual threads in order to assess the application's performance. According to a static internal performance model, the system adapted the thread partitioning to either meet a user-defined lower performance bound or to stay within a performance budget. We currently work on a heterogeneous multi-core that learns a thermal model at run-time and performs thermal management autonomously using vertical thread migration.

5. CONCLUSION

To meet the rising dynamics, complexities and requirements of future computing systems, we propose that future systems collect and maintain information about their system state and their progress autonomously. Therefore, we have proposed a HW/SW platform that consists of heterogeneous HW/SW cores and a monitoring core which senses the current state of each core. To enable self-expression, we break with the classic design-time abstraction layers of hardware, operating system and software. In contrast, our proposed system vertically migrates threads between these layers to affect various system characteristics, such as application performance, temperature distribution and power consumption.

Acknowledgment

The research leading to these results has received funding from the European Union 7th Framework Programme under grant agreement n° 257906, from the German Research Foundation (DFG) as part of the priority program "Dependable Embedded Systems" (SPP 1500) and the Collaborative Research Centre "On-The-Fly Computing" (SFB 901).

6. REFERENCES

[1] E. Lübbers and M. Platzner, "Communication and Synchronization in Multithreaded Reconfigurable Computing Systems," in *Int. Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA'08)*. CSREA Press, 2008.

[2] C. Plessl, M. Platzner, A. Agne, M. Happe, and E. Lübbers, "Programming Models for Reconfigurable Heterogeneous Multi-Cores," *Awareness Magazine, http://www.awareness-mag.eu*, 2012.

[3] E. Lübbers and M. Platzner, "Cooperative Multithreading in Dynamically Reconfigurable Systems," in *IEEE International Conference on Field Programmable Logic and Applications (FPL'09)*. IEEE, 2009.

[4] M. Happe, E. Lübbers, and M. Platzner, "A Self-adaptive Heterogeneous Multi-core Architecture for Embedded Real-time Video Object Tracking," *Journal on Real-Time Image Processing (JRTIP)*, 2011.