A HYBRID MULTI-CORE ARCHITECTURE FOR REAL-TIME VIDEO TRACKING

Markus Happe *

International Graduate School
University of Paderborn
33098 Paderborn - Germany
email: markus.happe@uni-paderborn.de

Enno Lübbers *

EADS Innovation Works
Technical Capability Center 5
81663 Munich - Germany
email: Enno.Luebbers@eads.net

ABSTRACT

In this paper, we present an implementation of real-time video tracking on a novel reconfigurable multi-core architecture capable of reacting to changing workload while minimizing the number of active cores. The system is comprised of multiple processor cores executing sequential software threads, and hardware cores implemented in an FPGA executing dynamically reconfigurable hardware threads. SW and HW threads interact using a unified multithreaded programming model, which allows on-the-fly reconfiguration to shift workload between hardware and software components. Our self-adaptation technique effectively re-partitions threads across hardware and software cores to keep the performance of a video object tracking application within a predefined budget while minimizing the number of used processing elements and, thus, saving power consumption.

1. INTRODUCTION

Modeling system components as threads that interact using strictly defined system services is a popular approach for software development, and is increasingly gaining popularity in the domain of hybrid HW/SW systems [1]. Here, software functions and hardware modules can be commonly thought as threads using identical system services such as semaphores or message queues for communication. As a result, managing the complex problem of adaptively changing the HW/SW partitioning of a system at run-time becomes a matter of instantiating or terminating HW and SW threads.

Thread re-mapping can improve the power efficiency of multi-core systems where only as many cores are used as are required for a defined performance. Inactive cores are assumed to consume less power than active cores. Our novel self-adaptation technique re-partitions threads onto different cores and even across the HW/SW boundary to keep the performance of a video object tracking application [2] despite changing input data within an user-defined performance budget while minimizing the number of active cores.

*The research leading to these results has received funding from the European Union Seventh Framework Programme under grant agreement n° 257906.

2. HYBRID MULTI-CORE ARCHITECTURE

Our architecture and algorithm depend on a unified programming model for both SW and HW components. The operating system ReconOS [1] extends the multithreaded programming model to the reconfigurable hardware domain. ReconOS promotes HW coprocessors to independent *hardware threads* and treats them equally to SW threads running on the system. In particular, ReconOS allows HW threads to use the same operating system services for communication and synchronization as SW threads, providing a transparent programming model across the HW/SW boundary.

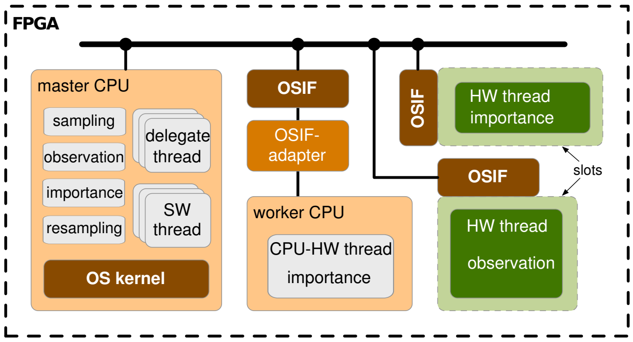

Fig. 1. Hardware architecture of the video object tracking system based on ReconOS

ReconOS takes advantage of the dynamic partial reconfiguration capabilities of Xilinx FPGAs to reconfigure hardware threads during run-time. This allows multiple hardware threads to transparently share the reconfigurable resources. Figure 1 shows the hardware architecture of a typical ReconOS system. The reconfigurable area is divided into multiple slots holding the individual hardware threads. A dedicated hardware OS interface (OSIF) handles the hardware thread's OS requests and forwards them to its corresponding delegate thread running on the CPU. ReconOS treats a worker CPU executing a software thread (which we call a CPU-HW thread) in the same way as a hardware slot executing a hardware thread. CPU-HW threads are thus also represented by delegate threads in ReconOS.

3. REAL-TIME VIDEO OBJECT TRACKING

For video object tracking, a histogram-based particle filter is used that can be divided into the stages sampling, observation, importance, and resampling. The observation stage calculates the histograms of the particles and the importance stage compares them to the object's histogram. Each of the filter stages can have an arbitrary number of software and hardware threads. In histogram-based video object tracking systems the object size strongly influences the computational complexity. Thus, many real-time video tracking systems track fixed-sized objects. When considering self-adaptive hybrid multi-core systems, however, we can allow changing object sizes by activating or deactivating cores.

We have implemented the video object tracker prototype on a Virtex-4 XC4VFX100 FPGA. The system is designed following the system architecture shown in Figure 1 and includes one master processor (running the OS kernel and housekeeping tasks of the particle filter framework [2]) and one worker processor and two hardware slots, each of which can execute one thread at a time. Both processors (PowerPC 405 CPUs) run at 300 MHz, while the hardware slots execute at 100 MHz. In our experiments, we track 100 particles, and measure the raw particle processing time.

For self-adaptation, we apply an add/remove strategy. Initially, the application executes entirely on the master processor. The master processor also measures the total application performance at user-defined time intervals. In case the performance drops below a lower threshold, the master creates an additional instance for the thread on the core that promises either meeting the desired performance budget, if possible, or the largest increase in performance, else. When the performance exceeds an upper threshold, the master terminates the thread instance that will lead to the reduction which is as close to the desired performance budget as possible, effectively reducing the dynamic power consumption of the system by suspending execution on the respective core.

Figure 2 shows an exemplary run of our self-adaptive video object tracking system for an exemplary video. The application's performance is measured in frames per second (FPS) and the desired average performance range is set to 8 FPS, where the budget is set to be 33% faster or slower than the defined average performance. In this example, we execute the self-adaptation algorithm every 20 frames with an initial offset of 8 frames. The time interval for running the self-adaptation algorithm is set to keep the overhead incurred by partial reconfiguration reasonably low. Using our proposed self-adaptation algorithm, the power consumption can be reduced by deactivating up to 3 of 4 cores.

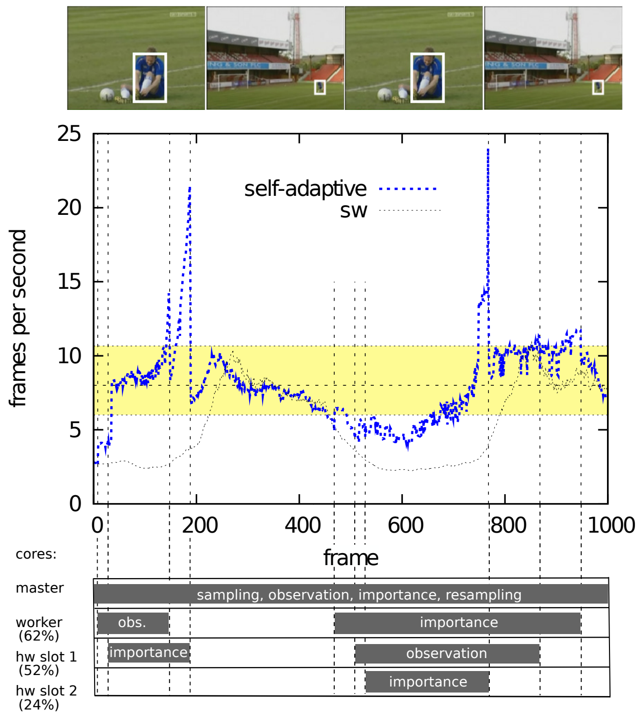

Fig. 2. Self-adaptation exemplary run: Resulting performance in FPS (upper part) and thread assignment (lower part). Re-partitioning points are represented by vertical dashed lines. The performance target is highlighted by a horizontal bar. [3]

4. CONCLUSION

In this paper, we present a novel thread-based self-adaptive task partitioning technique based on a reconfigurable hybrid multi-core architecture. By adaptively changing the HW/SW partitioning in reaction to data-dependent variations in application performance, our video object tracking system is able to maintain a predefined performance envelope while minimizing the number of required processing resources and, thus, lowering power consumption.

5. REFERENCES

[1] E. Lübbers and M. Platzner, "ReconOS: Multithreaded Programming for Reconfigurable Computers," *ACM TECS Special Issue (CAPA)*, 2009.

[2] M. Happe, E. Lübbers, and M. Platzner, "An Adaptive Sequential Monte Carlo Framework with Runtime HW/SW Partitioning," *IEEE International Conference on Field Programmable Technology (FPT)*, 2009.

[3] 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, to appear.