Parallel Hardware- and Software Threads in a Dynamically Reconfigurable System on a Programmable Chip

Marko Rößler
Chair Circuit and System Design
Chemnitz University of Technology
09126 Chemnitz, Germany
email: marr@infotech.tu-chemnitz.de

Today's embedded systems depend on the availability of hybrid platforms, that contain heterogeneous computing resources such as programmable processors units (CPU's or DSP's) and highly specialized hardware cores. These platforms have been scaled down to integrated embedded system-on-chip. Modern platform FPGAs enhance such systems by the flexibility of runtime configurable silicon. One of the major advantages that arises is the ability to use hardware (HW) and software (SW) resources in a time-shared manner. Though the ability to dynamically assign computing resources based on decisions taken at runtime is given.

The traditional way to implement efficient computational tasks on FPGA based hybrid platforms requires the development along two paths. Application development starts with the decision about HW/SW partitioning of parallel component computations (subtasks). It is followed by embedded software implementation, most likely in C, and custom hardware core implementation by the use of hardware description languages like VHDL or Verilog. Synthesis, mapping and extensive testing finalize the process. It would be advantageous to abstract away the distinction between the two sides of the traditional low-level HW/SW development into a system-level perspective and provide a continuous programming scheme throughout the development process.

Previous work in the field of unified programming models for platform FPGA's has been dominated by Andrews et al. [1] and their HThreads project. They were first in proposing a thread based model and utilize a C-to-HDL transformation to generate simple hardware threads. Remarkable is that they implement major parts of the operating system in hardware with focus on hard real time constraints. Thus dropping the flexibility of executing computation in software and the portability of their applications. Another approach is the RoConOS project by Lübbers et al. [2]. Off-the-shelf embedded OS (eCos and Linux) are extended to provide multi threading primitives to native hardware cores. Focus is to integrate existing IP into a software multi threaded environment. They embed the cores into a VHDL wrapper that consists of a state machine. It executes all interactions with the OS and enables a blockable control flow. Another OS for dynamically reconfigurable FPGAs has been developed by Steiger et al. [3]. All works manage task queues and placement but do not allow migration between computing resources and preemption of tasks.

Respective programming models are still immature, as they generally treat FPGAs as independent accelerators and therefore miss to utilize the full potential. Further more distributed and parallel computation on hybrid platforms requires:

- Independence from execution domain for component computations
- Execution synchronization between component computations
- Preemption of component computations for dynamic partitioning

From the programmers point of view the following features are critical:

- Uniform programming language
- Access to an HW/SW runtime environment that provides common operating system primitives
- Automated tool and work flow with simulation capabilities during the design process.

The major novelty of this work is to deploy modern High-Level-Synthesis technology as key to create a uniform programming model [4] to address these points. Thus achieving major improvements in system efficiency while increasing productivity and portability. To the best of my knowledge this is the first work that combines an integrative programming view and an unitary runtime environment across the HW/SW boundary. The software concept of concurrently running threads is utilized to implement component computations in line with the well known POSIX API standard in augmented C language. The synthesis tool flow is based on a modified commercial High-Level C to VHDL transformation. The tool is enhanced to generate functional identically instances of a thread for HW and SW domain, communication interfaces, management code injection to support preemption, and runtime primitives i.e. semaphores. A modified Linux kernel is used to provide the uniform runtime environment including the scheduler that is aware of an applications real time constraints. Management of the configurable logic area is carried out by the reconfiguration controller in a fragmentation preventing manner.

The current state of my work supports shared memory, creation and destruction of threads and protection of shared resources by mutexes due to the enhancement of the High-Level synthesis tool CoDeveloper. Presently we are capable to cooperatively preempt threads and migrate them including the traversal of the HW/SW boundary at runtime [5]. We further introduce a simple scheduler that preferably schedules threads to hardware computation and assigns mutexes priority based. The integration of more sophisticated scheduling algorithms as in [6] is subject to further work.

REFERENCES

[1] D. Andrews, R. Sass, E. Anderson, J. Agron, W. Peck, J. Stevens, F. Baijot, and E. Komp, "Achieving programming model abstractions for reconfigurable computing," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 16, no. 1, pp. 34–44, 2008.

[2] E. Lubbers and M. Planner, "ReconOS: An RTOS Supporting Hard-and Software Threads," Aug. 2007, pp. 441–446.

[3] C. Steiger, H. Walder, and M. Platzner, "Operating systems for reconfigurable embedded platforms: online scheduling of real-time tasks," *Computers, IEEE Transactions on*, vol. 53, no. 11, pp. 1393–1407, Nov. 2004.

[4] M. Rössler and U. Heinkel, "Preemptive HW/SW-Threading by combining ESL methodology and coarse grained reconfiguration," in *ReCoSoC'08: Proceedings of the 4th International Workshop on Reconfigurable Communication Centric System-on-Chips*, Barcelona, Spain, July 2008.

[5] E. Billich, M. Rössler, and U. Heinkel, "Flexible Implementation of a MJPEG Coding Chain using Preemptive HW/SW-Threading," in *submitted*, December 2008.

[6] N. Guan, Q. Deng, Z. Gu, W. Xu, and G. Yu, "Schedulability analysis of preemp-