Modeling Application-Specific Processors for Embedded Systems

Florian Brandner•, Viktor Pavlu°, and Andreas Krall°

•COMPSYS, LIP, ENS de Lyon
UMR 5668 CNRS – INRIA – UCB Lyon
florian.brandner@ens-lyon.fr

° Institute of Computer Languages
Vienna University of Technology
vpavlu,andi@complang.tuwien.ac.at

Abstract: Embedded systems often have to operate under rigid power and performance constraints. Off-the-shelf processors often cannot meet those requirements, instead *Application-Specific Instruction Processors* (ASIP) are used that are tuned for the particular system at hand.

A popular and powerful way of modeling ASIPs is the use of a *Processor Description Language* (PDL). These languages capture the internal hardware organization as well as the processor's instruction set using a formal specification. Given a processor description, generator tools can (semi-)automatically derive software development tools, instruction set simulators, and even hardware reference models.

An integral part of the software, running on the ASIP, is the interaction with devices outside of the computing platform. However, these external devices are neglected by many PDLs. This is, in part, due to their diverse nature and complex behavior. Explicitly including such devices in processor models, is thus unlikely to give a practical solution.

We propose a basic set of communication patterns for the xADL processor exploration system that allow to interact with external devices, while otherwise treating them as black boxes. The xADL system allows to model three kinds of communication: (1) data exchange using dedicated instructions or memory mapped I/O, (2) asynchronous delivery of data directly into processor registers or memory, and (3) asynchronous signaling using interrupts. A major advantage of our approach is that all side-effects of these interactions are visible to the xADL tool suite. For example, our compiler generator accounts for side-effects during code generation, while the generated simulators reduce simulation time by refactoring the expensive emulation of interrupts.