The Role of Flexibility in Hardware/Software Co-Design

Rolf Ernst

The term “flexibility” is often used as an argument to support the use of programmable versus hardcoded system functionality. Despite the common use of this term, and despite it’s obvious importance, there is no clear and easy definition or even a quantification of flexibility. How does a designer know that a programmable system does allow later modifications of the system to be implemented and what kind of modifications will be supported? How can we optimize for flexibility? How can we define an interval of required system functionality to optimize a system for maximum reusability in a certain area of application (e.g. optimize for market segment)? In this context, time constraints are obviously a part of functionality.

The computer architecture and designer communities have developed a good knowledge of how to systematically design either general purpose architectures or systems which are targeted to a single or a well-defined set of individual applications. However, the more general flexibility and reuse issue as described here becomes crucial with systematic co-design, because the decision to implement functions in hardware or software has an influence on flexibility (independent of how we want to define flexibility as long as it is somehow intuitive). When it finally comes to optimization, such as in co-synthesis, then a qualitative, “hand-waving” flexibility perception is not acceptable any more. So, we should either drop this term from system evaluation and optimization or try to work towards a reasonable definition of flexibility.

We should try to collect and discuss some ideas and possible alternatives on how to define flexibility and how to treat it systematically in co-design.