Multicore Systems: Impact of the Programming Language

F. Siebert
AICAS

The move towards multicores is so strong that real-time and safety-critical applications will make use of multicores and have to adapt to the rules dictated by multicore hardware, while off-the-shelf multicore hardware is optimized for average-case throughput. This talk focusses on the impact this move to multicores has on real-time and safety-critical code. The software developer has to be aware of the effects of cache structures and memory models to understand the consequences on performance and correctness of his code.

Multicore systems have become the norm for desktop computer systems. The percentage of multicore systems in the embedded domain is still marginal, but growing at an incredible pace such that multicore will become the norm in the embedded area as well. However, embedded systems have additional requirements with respect to safety, reliability, real-time behaviour, etc. The use of parallel multicore systems introduces new challenges to the embedded systems developers who has to fulfil these requirements when developing new software or porting existing code to multicore systems.

This paper gives an overview over typical problems that arise with the use of multicore systems. Unfortunately, a multicore system in many cases does not behave the same as a single core system with
multithreading. These differences may result in severe program errors. A good understanding of the sources of these errors is required.

It will be explained how problems on multicore systems can be avoided. Mechanisms present in different programming languages such as C/C++ and Java will be presented. An important role is played by the memory model provided by the language implementation and how this may manifest in false program behaviour. The memory model permits different compiler and CPU optimisation on atomic and non-atomic operations, and allow different semantics of the volatile modifier.