Karl-Erik Årzén

Title: Simulation of Cyber-Physical Control Systems

Abstract: Whether a control application should be considered CPS or not depends on (at least) three different things. A control system is a CPS when the temporal effects of the implementation platform caused by computing and communication, needs to modeled and included in the design at a more detailed levels than what is traditionally done in computer-based control (periodic sampling and constant latencies). Second, control applications of a CPS nature are typically more distributed and decentralized in nature than the classical, more centralized, control approaches. Third, a control application can be considered to be CPS when the system under control itself is a computing and/or communication system, e.g., a data center on an embedded MPSoC.

Simulation is a technique that is well-established in the control community as a complement to more formal verification techniques. Correct simulation of cyber-physical control systems requires support for co-simulation of the temporal effects of real-time kernels and networks together with the continuous-time dynamics of the physical system under control. This is supported by the TrueTime simulator. TrueTime has been available since around 2000 in a Matlab/Simulink version. Currently TrueTime is being ported to Modelica. The equation-based and object-oriented nature of Modelica makes it widely superior to Simulink for representing the Physical part of a CPS system. The new synchronous support for hybrid and discrete-time systems currently being developed within the Modelica community, also makes it ideally suited for representing discrete-time controllers. With the additional support provided by TrueTime for simulating real-time kernels and networks Modelica is also able to handle the Cyber part of a CPS system. The talk will focus on the new features in TrueTime, e.g., support for multicore kernels and bandwidth reservations and the Modelica version of TrueTime currently being developed jointly by Lund University and Vanderbilt University. The version is based on the open model exchange interface FMI (Flexible Mock-Up Interface), an open source alternative to Simulink's S-functions.