This course introduces into the system-level language SystemC. Special emphasis will be given on modelling across different levels of abstraction from untimed via timed transaction level models down to register transfer models including the needed refinement steps. Further topics are integration of hardware-dependent software and hardware synthesis from SystemC.

<table>
<thead>
<tr>
<th>Course Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Course Title</strong></td>
</tr>
<tr>
<td><strong>Category</strong></td>
</tr>
</tbody>
</table>
| **Objectives**     | • Introduction into the SystemC 2.1 language  
|                    | • Transaction-level modelling using SystemC |
| **Structure/Agenda** | **SystemC**  
|                    | • Evolution and philosophy of SystemC  
|                    | • Brief excursus to C++  
|                    | • Introduction into SystemC  
|                    | - modules, processes (method, cthread, thread), primitive communication, data types  
|                    | • Advanced SystemC modelling  
|                    | - complex communication (primitive and hierarchical channels), temporal behaviour (wait, event, notify, sensitivity lists), simulation semantics  
|                    | **System level modelling in SystemC**  
|                    | • Modelling at different levels of abstraction in SystemC  
|                    | - computational models, computation-centric abstraction: RT level (explicit vs. implicit FSM), algorithmic level (superstate, timed, untimed), system level (dataflow vs. discrete event models)  
|                    | • Transaction-level modelling in SystemC  
|                    | - Models of abstraction: cycle accurate, a cycle approximated, timed functional and untimed functional vs. cycle callable, programmers view (timed and untimed), communicating processes (timed and untimed)  
|                    | - Communication refinement using transactors and inheritance  
|                    | • SystemC Synthesis  
|                    | - Synthesis subsets (RTL and Behavioural Level), synthesis phases, optimization steps, synthesis tools  
|                    | • Integration of hardware-dependent software tasks in SystemC  
| **Level**          | Introductory till intermediate level |
| **Target Audience**| Designers, engineers and technical managers with knowledge in system level design or hardware design using VHDL or Verilog |
| **Course Provider**| Dr. Oliver Bringmann  
|                    | FZI Forschungszentrum Informatik  
|                    | Karlsruhe, Germany |