A Complete SystemC Process Instrumentation Interface and Its Application to Simulation Performance Analysis

Bishnupriya Bhattacharya, Chandra Sekhar Katuri, Vincent Motel
Cadence Design Systems

Abstract
In the Electronic System Level domain, the SystemC language uses non-preemptive processes to simulate hardware parallelism in C++. Processes are a fundamental construct of the language, which makes modeling of hardware easier than in plain C++. Since processes are a key design element of SystemC, there are various application domains (e.g., debugging, performance analysis) where a generic mechanism to associate user-defined callbacks with process state changes will prove to be beneficial.

We propose such a generic interface for registering user-defined callbacks triggered at SystemC process state changes. We also present an application case study for such an interface for simulation performance profiling in a virtual platform and in an approximately-timed memory controller model.

SystemC processes
• Non-preemptive processes simulate hardware parallelism
• Process execution follows a co-routine semantics: processes voluntarily yield control, the SystemC scheduler never preempts a process forcefully
• Two kinds of processes
  SC_METHOD: executes completely from beginning to end in each execution, and does not maintain any local state
  SC_THREAD: maintains its own local state, and on execution can voluntarily suspend itself using the wait() construct
• Static processes are created in the initial phases
• Dynamic processes are created during the simulation

SystemC Process Instrumentation Interface
We propose a SystemC language extension to register user-defined callbacks triggered at process state changes.

Class sc_process_callback

Implementation of the process instrumentation interface
• A subset of the process callback interface has been implemented in the Cadence Intrinsics simulator
• In the 15.10 release
• Some constructs for advanced use modes are not implemented yet: process�� (process level) and node level dynamic sc_process_callback processes(s), callback registration for individual process handles. No performance degradation when process callbacks are not registered
• The natural next step is to propose the specification of this interface to Accellera (SystemC kernel)
• This models the parallel nature of electronic systems
• From the start of the simulation to the end of the simulation, all executable code is initiated from one or more processes
• But the simulation execution is sequential

Application to Simulation Performance Analysis
The process instrumentation interface can be used to make accurate simulation performance analysis, by measuring the exact CPU time spent in each process execution.

Profile the virtual platform of Xilinx Zynq-7000 using SystemC. "All Programmable SoC", running MRI image processing in hardware and software

Comparison to Sample-based Profiling
Comparison to sample-based profiling, on memory controller model: top process

Comparison to sample-based profiling, on memory controller model: top 8 processes

Conclusion
We have proposed a complete interface that allows user’s application code to be notified of all the state changes that occur in a SystemC process’s lifetime, and execute a user-defined callback action, without a need for the user to modify the SystemC kernel. This interface is very generic and enables diverse and useful application areas, which we have demonstrated on a profiling case study.

In the simulation performance analysis domain, profiling solutions based on this process interface can produce very accurate results, which are useful for incremental optimization of model performance.