UVM-SystemC and CCI Configuration

Martin Barnasconi

SystemC Evolution Day
May 3, 2016

accellera SYSTEMS INITIATIVE

SYSTEMC™ EVOLUTION DAY
MAY 3, 2016 | MUNICH | GERMANY

Presentation Copyright Permission

– A non-exclusive, irrevocable, royalty-free copyright permission is granted by **NXP Semiconductors** to use this material in developing all future revisions and editions of the resulting draft and approved Accellera Systems Initiative **SystemC** standard, and in derivative works based on this standard.

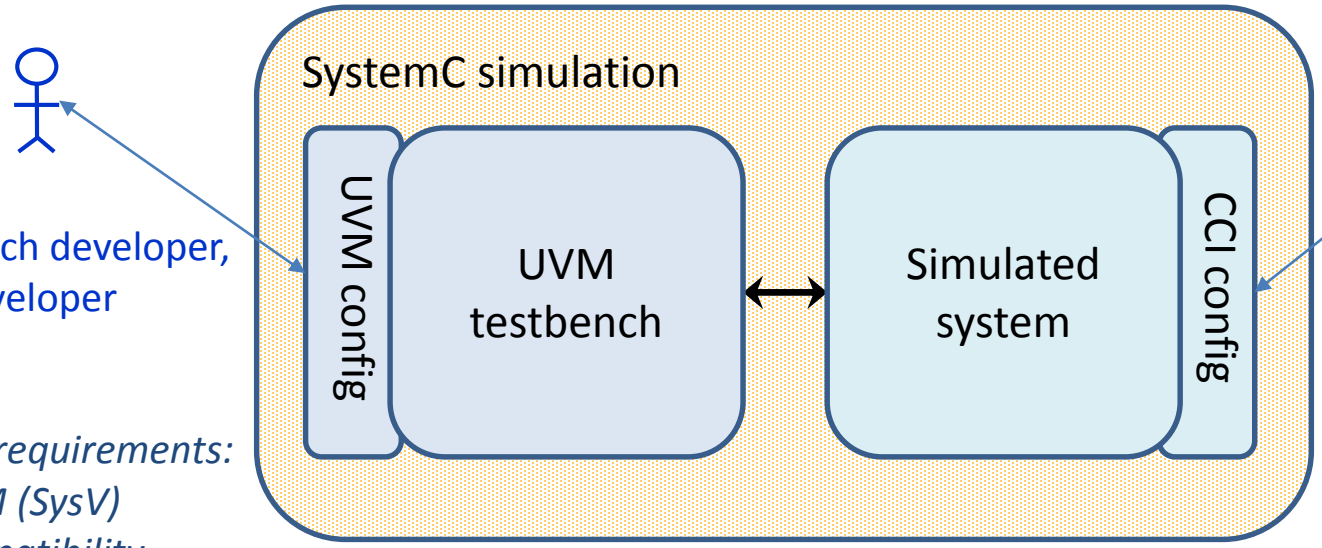
Presentation outline

- Some logistics
- SystemC Configuration Use Models
- Discussion topics
- Summary (TBD)

Some logistics…

- We need a **volunteer to take minutes**
- This session is all about having an **interactive discussion**
- We need to prepare a **summary** for the wrap-up session
  - Main consensual/controversial points
  - Next steps in Accellera (including owners and destination WGs)
  - Present summary in 1-2 slides

SystemC Configuration Use Models



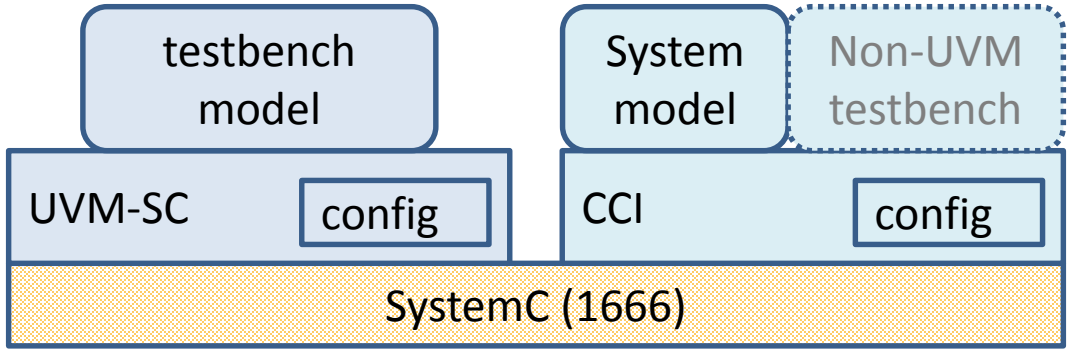
Testbench developer,
Test developer

Unique requirements:
- *UVM (SysV) compatibility*
- *Non-intrusive to simulated system*

IP developer,
System integrator,
System SW developer

Unique requirements:
- *Tool introspection*
- *Configure how system constructed*
- *Authoring support*
- *IP protection*

Software Architecture:



accellera SYSTEMS INITIATIVE

SYSTEMC EVOLUTION DAY
MAY 3, 2016 | MUNICH | GERMANY

Discussion topics

- Configuration
  - UVM-SystemC versus CCI: combine or separate mechnisms?
  - Use model: Tool/model interface and/or tool/testbench interface?
- Register interface
  - uvm_reg versus sci_reg: combine or separate mechnisms?
  - Use model: Connect UVM-SystemC backdoor to a HDL or SystemC object via the HDL path

Summary

- TBD

UVM-SystemC Layered Architecture

- The top-level (e.g. **`sc_main`**) contains the test(s), the DUT and its interfaces
- The DUT interfaces are stored in a configuration database, so it can be used by the UVCs to connect to the DUT
- The test bench contains the UVCs, register model, adapter, scoreboard and (virtual) sequencer to execute the stimuli and check the result
- The test to be executed is either defined by the test class instantiation or by the member function **`run_test`**