SystemC CCI
What's new? What's next?

SystemC CCI WG

accellera SYSTEMS INITIATIVE™

SYSTEMC™ EVOLUTION DAY
NOV 16, 2023 | MUNICH | GERMANY

Copyright Permission

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

Introduction

- Took over CCI chair position from Ola Dahl in 2022
- Working at MachineWare on Virtual Platforms
  - Previously at Synopsys, GreenSocs and RWTH Aachen
- Also active in
  - SystemC Language Working Group
  - SystemC Common Practices Subgroup
  - Federated Simulation Standard PWG

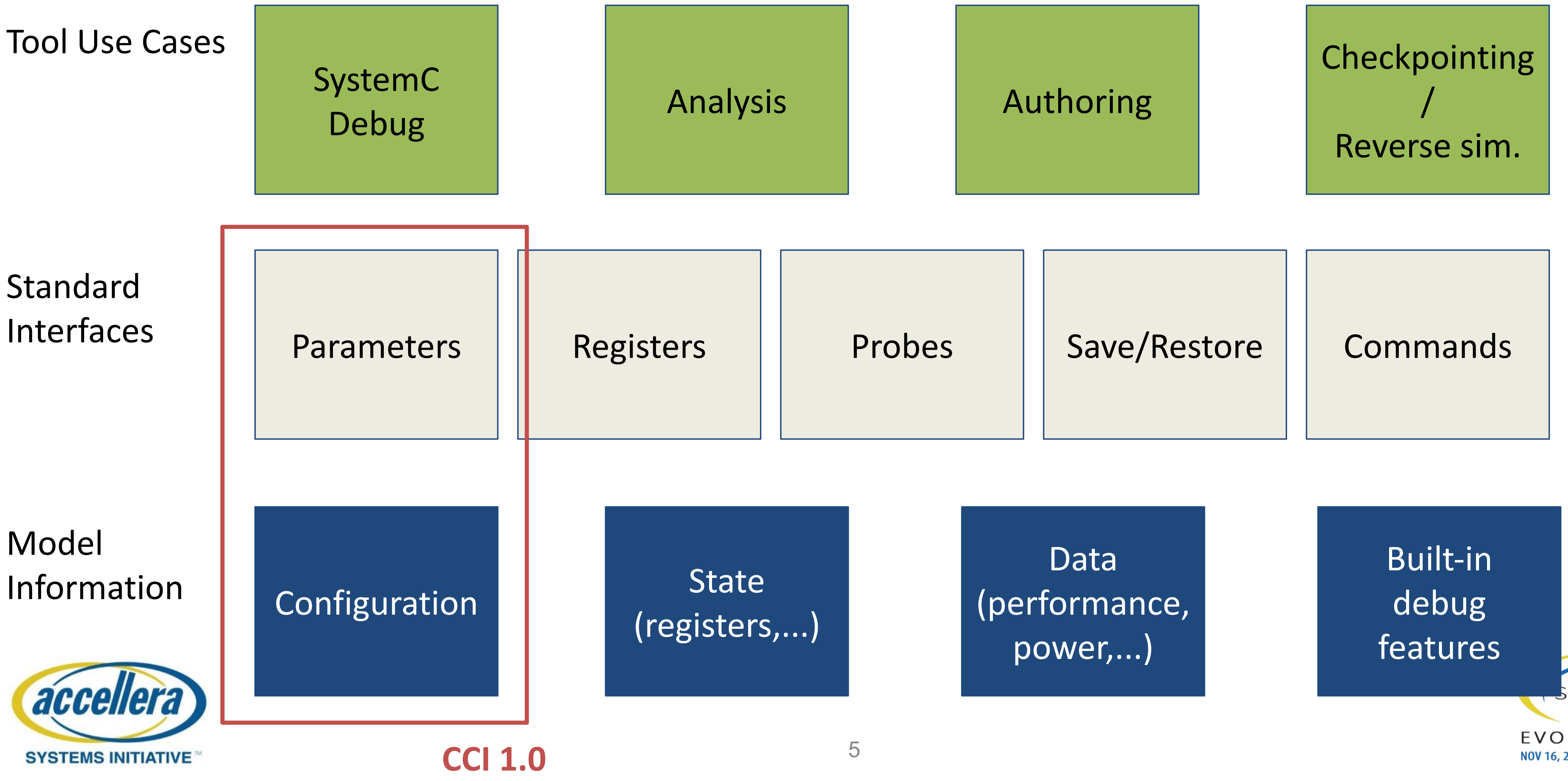
accellera SYSTEMS INITIATIVE™

SYSTEMC™ EVOLUTION DAY NOV 16, 2023 | MUNICH | GERMANY

Agenda

- Introduction to CCI
- CCI Basics
  - CCI Memory Inspection API
  - State of CCI
- Open discussion
- Wrap-Up

CCI Roadmap


Tool Use Cases
SystemC Debug
Analysis
Authoring
Checkpointing / Reverse sim.
Standard Interfaces
Parameters
Registers
Probes
Save/Restore
Commands
Model Information
Configuration
State (registers,...)
Data (performance, power,...)
Built-in debug features
CCI 1.0

SystemC Configuration, Control, Inspection

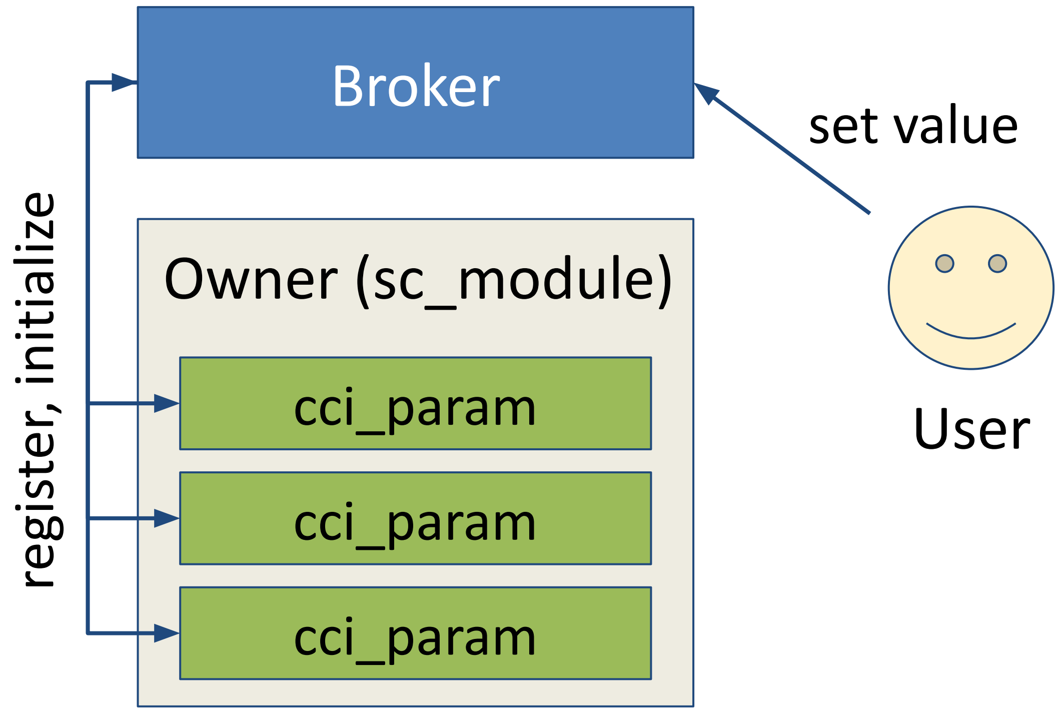
- Motivation
  - VPs are combination of models (black boxes)
  - Users want to configure, control and inspect the VP and its models
- Goal
  - Provide standardized means for VP/model configuration, control and inspection
- Addressed in CCI 1.0
  - Configuration: introduce parameters and means to use them
- Not addressed so far
  - Control, Inspection
  - Proposal for memory inspection from NXP under review

1.0.1 Release

- 1.0.1 mainly bug-fix and infrastructure release
- Added CMake build
  - Contributed by Mark Burton. Thank you!
- Added CI infrastructure for testing different OSs and SystemC versions
  - Implemented using GitHub Actions
  - Testing every commit, MR, …
  - Contributed by Nils Bosbach. Thank you!
- Added release automation on GitHub
  - Contributed by Nils Bosbach. Thank you again!

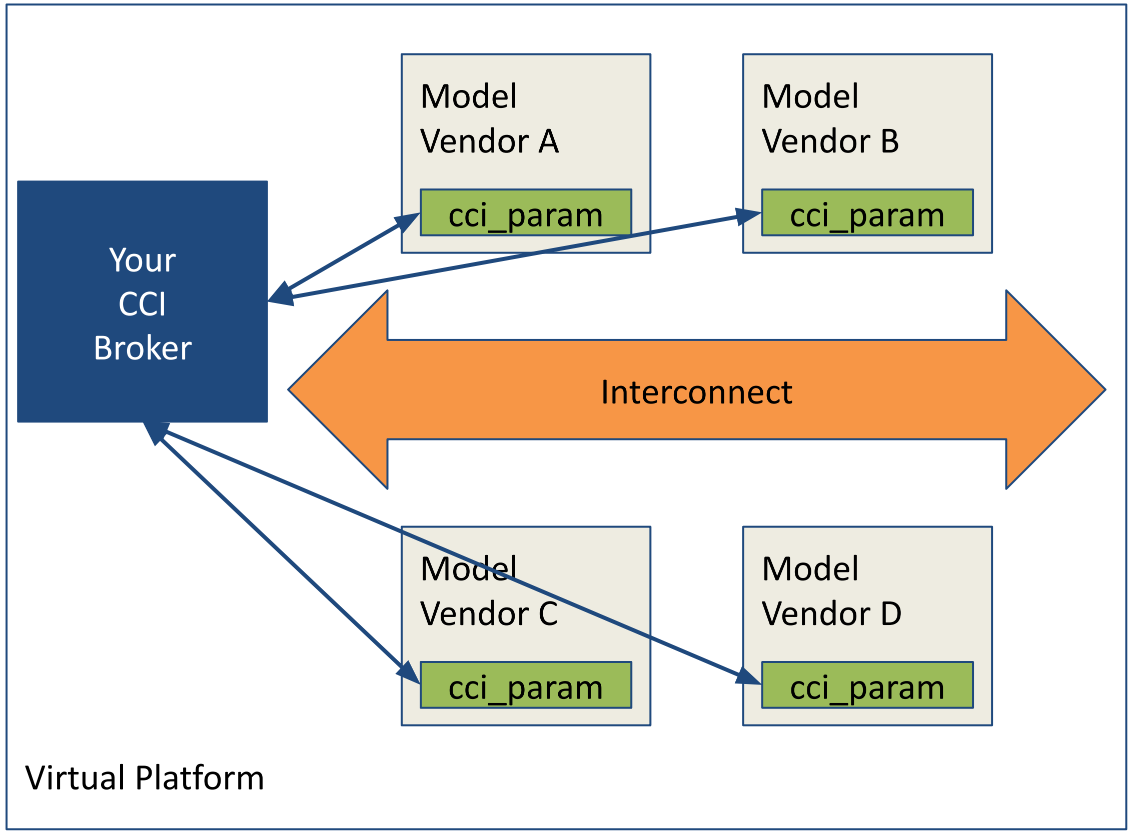
Configuration: Idea (simplified)

- Parameters get value from Broker
- Parameter owner specifies default
- Broker overrides default
- User sets value in Broker
- Broker is singleton
- Parameter identified by name



Configuration: Use Case

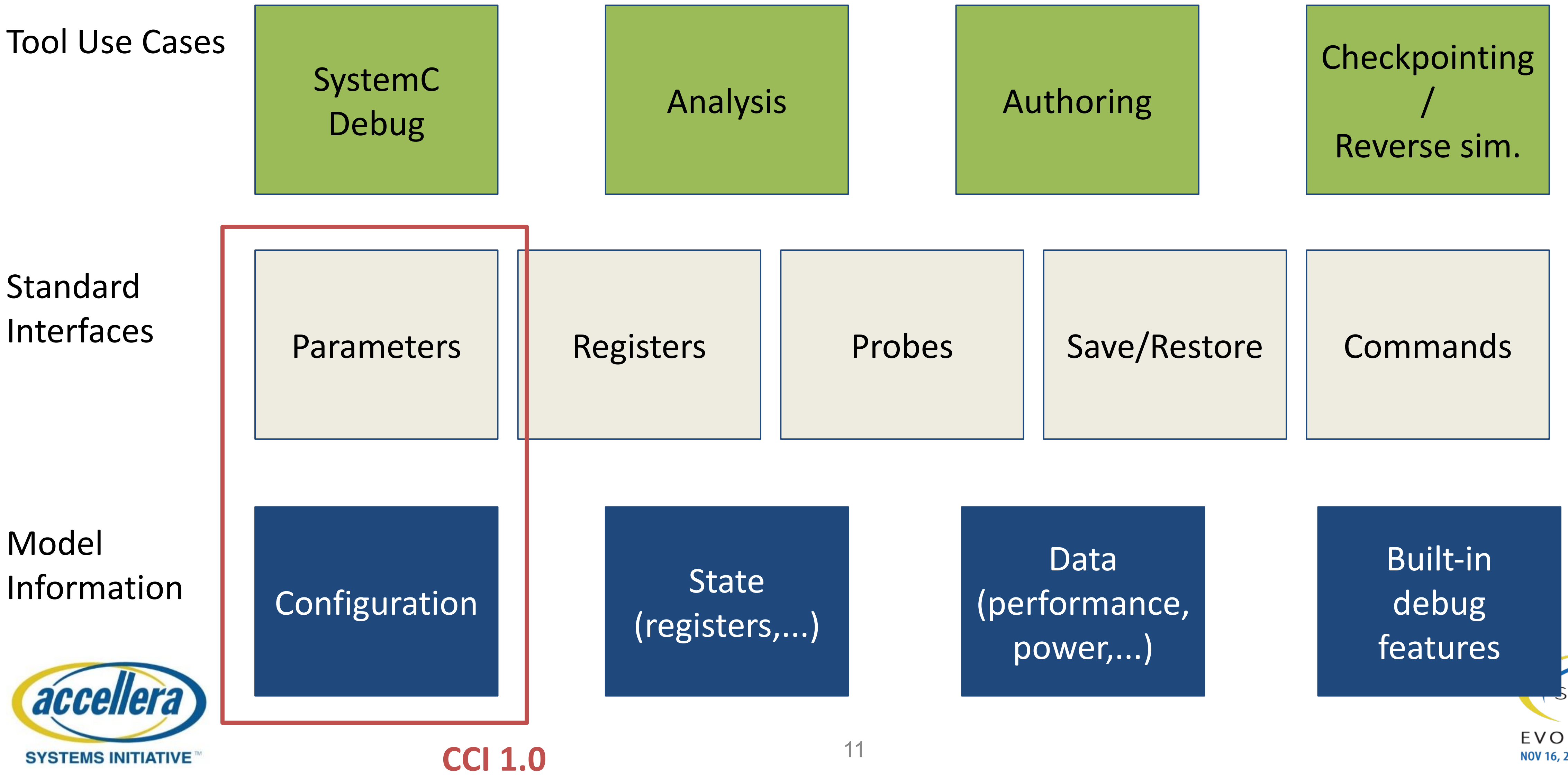
- Parametrization of sc_modules
- Set parameters of any (black box) SystemC model from any vendor
- Combine models into user configurable VP



Configuration Inventory Continued

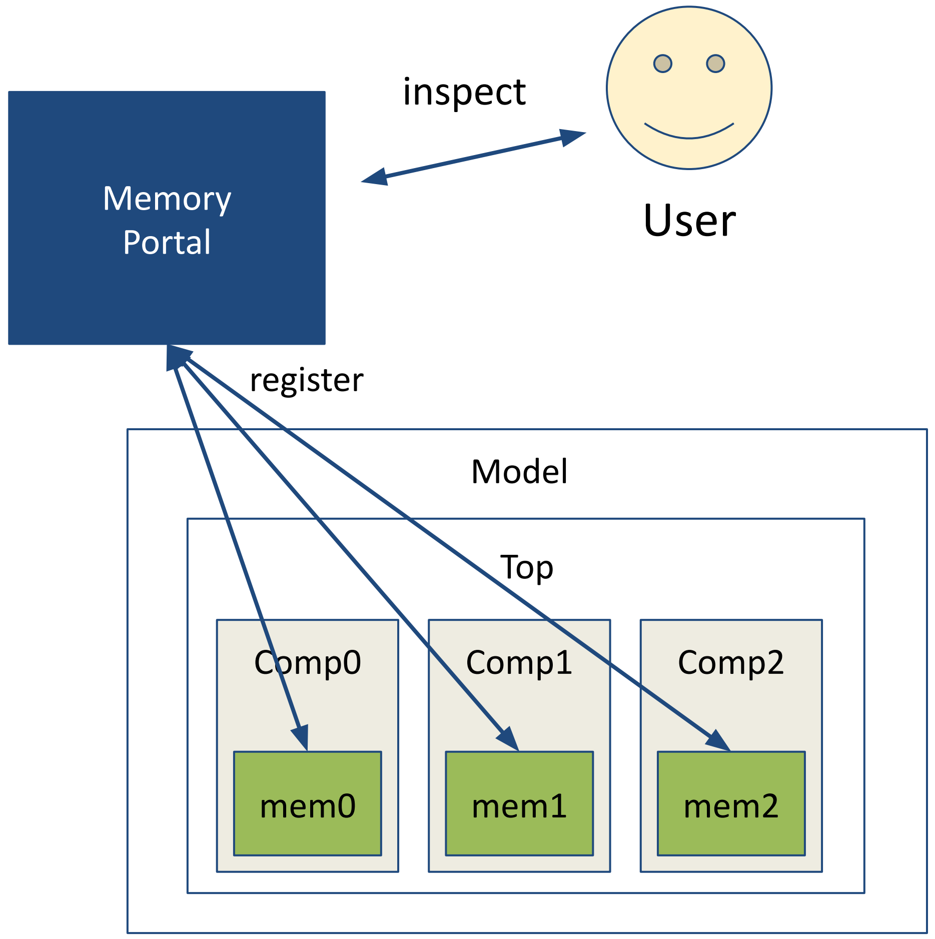
- Inventory implemented in POC for CCI 1.0
  - Parameter: Carries a default value, may be overwritten by user through broker, may have read/write callbacks
  - Broker, broker manager: manage parameter values, callbacks for parameter creation/destruction, broker hierarchies possible
  - Originator: track origin of parameter values
  - cci_value: variant type for storing parameter values, may be provided in JSON, some utilities exist like list, map

CCI Roadmap


Tool Use Cases
SystemC Debug
Analysis
Authoring
Checkpointing / Reverse sim.
Standard Interfaces
Parameters
Registers
Probes
Save/Restore
Commands
Model Information
Configuration
State (registers,...)
Data (performance, power,...)
Built-in debug features
CCI 1.0

CCI Memory Inspection: Idea

- Proposal by NXP, under review
- Allow user to peek/poke/register callbacks any memory from (black-box) models
- Model memories register to portal
- User utilizes portal for memory inspection



CCI Memory Inspection: Status

- Ongoing review process, feedback encouraged
- Discussion about
  - Should the API support memory hierarchies
  - What (if any) callbacks should be supported
  - Can we reuse existing CCI components (broker, …)
  - Exposed memory endianness

The State of CCI

What is the vision?

- Configure, control and inspect any VP/model the same way
- Combine any (black-box) SystemC models into a configurable, controllable, inspectable VP
- Reuse configuration files, scripting interfaces, inspection tools, …
- Standardized, open interfaces
  - POCs exist in the wild: GS libraries, SCC, VCML, …
  - Can’t be that hard ;-)

What do we have?

- CCI 1.0 standard: Parameters, Brokers, etc.
- A CCI 1.0.1 Proof-Of-Concept implementation
  - POC C++ library and tests
  - Continuous integration testing with major SystemC releases and OSs
- An agreement that the need exists (or does it?)
- An completely open collaboration platform: Public GitHub repository
- Motivated WG members

SYSTEMS INITIATIVE™

EVOLUTION DAY
NOV 16, 2023 | MUNICH | GERMANY

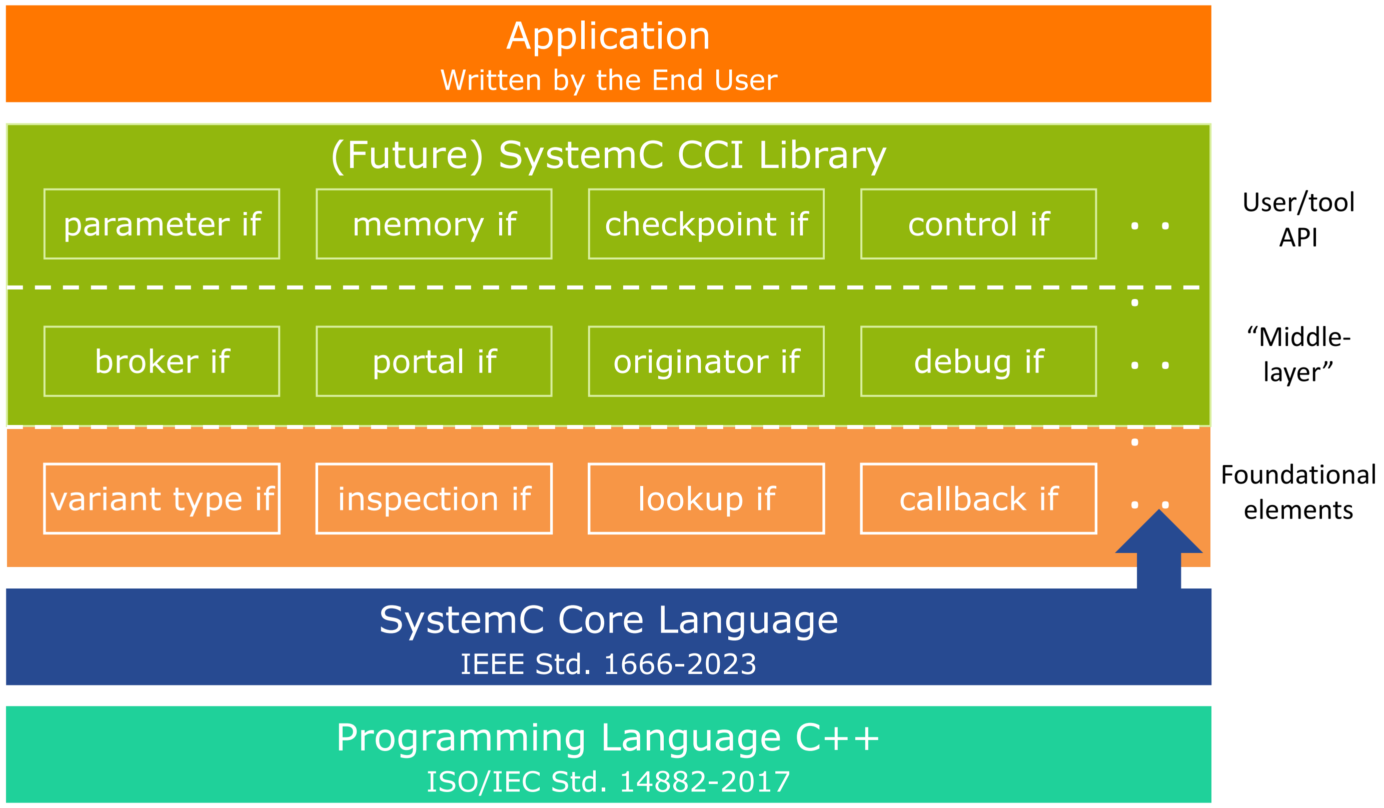
What do we not have?

- A widely adopted standard for Configuration
- A standard for Control and Inspection
  - Control: Session, breakpoints
  - Inspection: Memory, register inspection, watchpoints
- Enough human resources

Revisiting / refactoring the CCI library architecture?

- Current CCI library geared towards configuration (parameters, values, …)
- Standardization of register/memory inspection API revealed that we need a similar (but slightly different) API
- Challenge: the more CCI extensions we introduce, the higher the risk of duplication of functionality and APIs
- We need to make a **strategic decision:**
  - Each CCI domain gets its own unique (user/tool) API and corresponding (base) class libraries and features
  - Each CCI domain gets its own unique (user/tool) API but shall leverage the same underlying foundational elements

  … and which of these foundational elements should end-up in a future SystemC standard?



Discussion