Accellera SystemC Standards Update October 2021

Martin Barnasconi

Accellera Technical Committee Chair

accellera.org

SYSTEMS INITIATIVE™

SYSTEMC™ EVOLUTION DAY

OCT 28, 2021 | VIRTUAL WORKSHOP

Outline

- Accellera Systems Initiative
- SystemC ecosystem
- Accellera SystemC Working Groups
  - SystemC Language Working Group
  - SystemC Analog/Mixed-Signal Working Group
  - SystemC Configuration, Control & Inspection Working Group
  - SystemC Synthesis Working Group
  - SystemC Verification Working Group
- Contribute to systemc.org
- IEEE related Working Groups
- Advancing SystemC Standards Together

EVOLUTION DAY
OCT 28, 2021 | VIRTUAL WORKSHOP

Accellera Systems Initiative

Our Mission

To provide a platform in which the electronics industry can collaborate to innovate and deliver global standards that improve design and verification productivity for electronics products.

Accellera Membership - Broad Industry Support

Corporate Members

AMD arm cādence
ERICSSON intel
QUALCOMM SIEMENS NXP
ST SYNOPSYS

Start-Up and University

BATTELLE KÄRNTEN University of Applied Sciences tu technische universität dortmund
UNIVERSITY OF MARYLAND SCIENCES SORBONNE UNIVERSITÉ 芯华章 X-EPIC

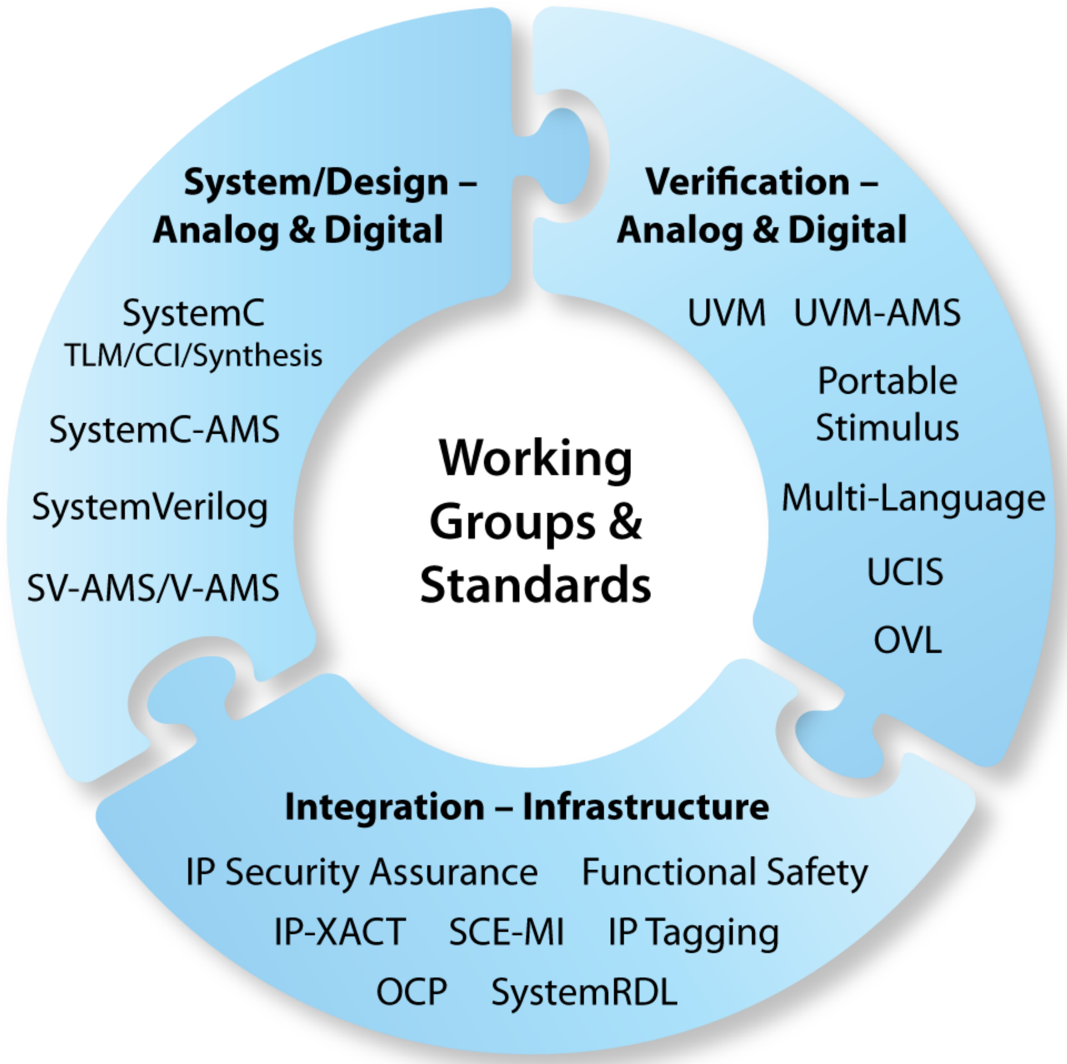
Associate Members

ADVANTEST aedvices CONSULTING AGNISYS ALDEC AMIQ amu
ANALOG DEVICES ARTERIS IP BOSCH Invented for life BREKER COSEDA technologies DEFACTO TECHNOLOGIES
dialog SEMICONDUCTOR DOULOS Fraunhofer IIS HUAWEI IBM
infineon MARVELL MICROCHIP Micron NVIDIA onespin
PERFORCE Semifore Sigasi; TEXAS INSTRUMENTS Tortuga Logic Vayavya Labs Pvt. Ltd.
Verific Design Automation verilab XILINX

accellera SYSTEMS INITIATIVE

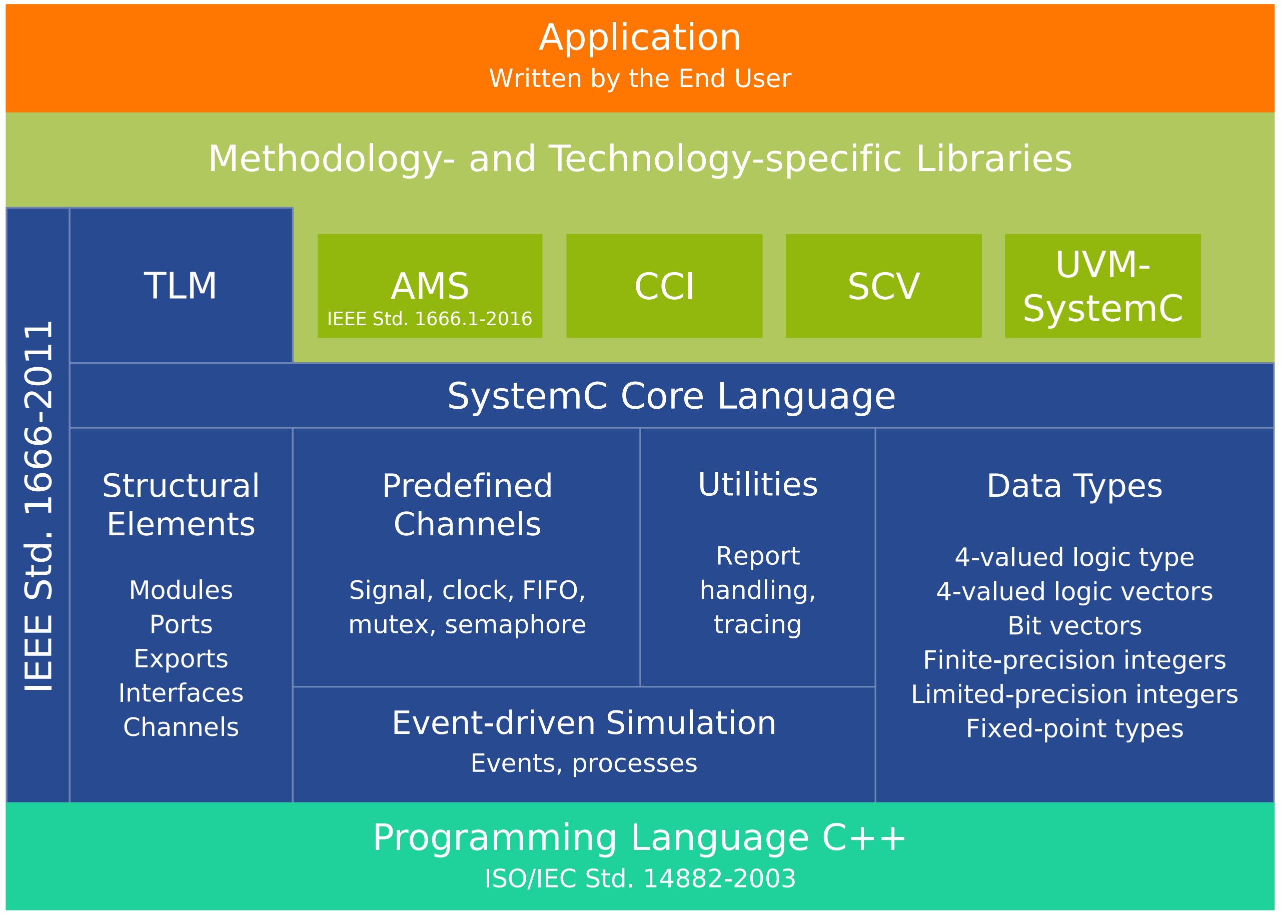
SYSTEM C EVOLUTION DAY OCT 28, 2021 | VIRTUAL WORKSHOP

Accellera Standards Developments



SystemC ecosystem

- SystemC is a C++-based language standard, widely used for
  - System-level modeling, design and verification
  - Architectural exploration, performance modeling
  - Analog/mixed signal modeling
  - High-level Synthesis
  - Software development
- Defined by Accellera, ratified as IEEE Std. 1666-2011 (SystemC) and IEEE Std. 1666.1-2016 (SystemC AMS)



Accellera SystemC Working Groups

- SystemC Language Working Group (LWG)
  - Chair: Laurent Maillet-Contoz (ST)
  - Subgroups
    - Transaction-Level Modeling (TLMWG), Chair: Bart Vanthournout (Synopsys)
    - SystemC Datatypes (SDTWG), Chair: Frederic Doucet (Qualcomm)
    - Common Practices (CPSWG): Chair: Mark Burton (GreenSocs)
- SystemC Analog/Mixed-Signal Working Group (AMSWG)
  - Chair: Martin Barnasconi (NXP)
- SystemC Configuration, Control & Inspection Working Group (CCIWG)
  - Chair: Ola Dahl (Ericsson)
- SystemC Synthesis Working Group (SWG)
  - Chair: Andres Takach (Mentor)
- SystemC Verification Working Group (VWG)
  - Chair: Stephan Gerth (Bosch)

SystemC Language Working Group

- The SystemC Language Working Group is responsible for the definition and development of the SystemC core language, the foundation on which all other SystemC libraries and functionality are built
- **Current status**
  - Finalizing SystemC language updates and delivering these to IEEE P1666 Working Group
  - SystemC reference implementation available at GitHub
  - IEEE Std. 1666-2011 is available at no cost, sponsored by Accellera, under the IEEE GET Program
- **Developments & future plans**
  - Studying performance of updated SystemC Datatypes implementations, integration into SystemC reference implementation to enable regression testing and validation
  - SystemC Common practices group currently evaluating various register implementations, following the Call for Contributions announced in July 2021
  - Updating SystemC reference implementation, as part of upcoming release SystemC 2.3.4

Common Practices Working Group

- Open to contributions from everybody
- Published PySysC
  - SystemC bindings for Python, available in the accellera-official public repository
- Evaluating community contributions offering register implementations
  - SystemC Components (SCC) contributed by MINRES Technologies GmbH
  - Virtual Components Modeling Library (VCML) contributed by Jan Weinstock and Lukas Jünger
- References to these libraries and results will be published on systemc.org

Data types Working Group

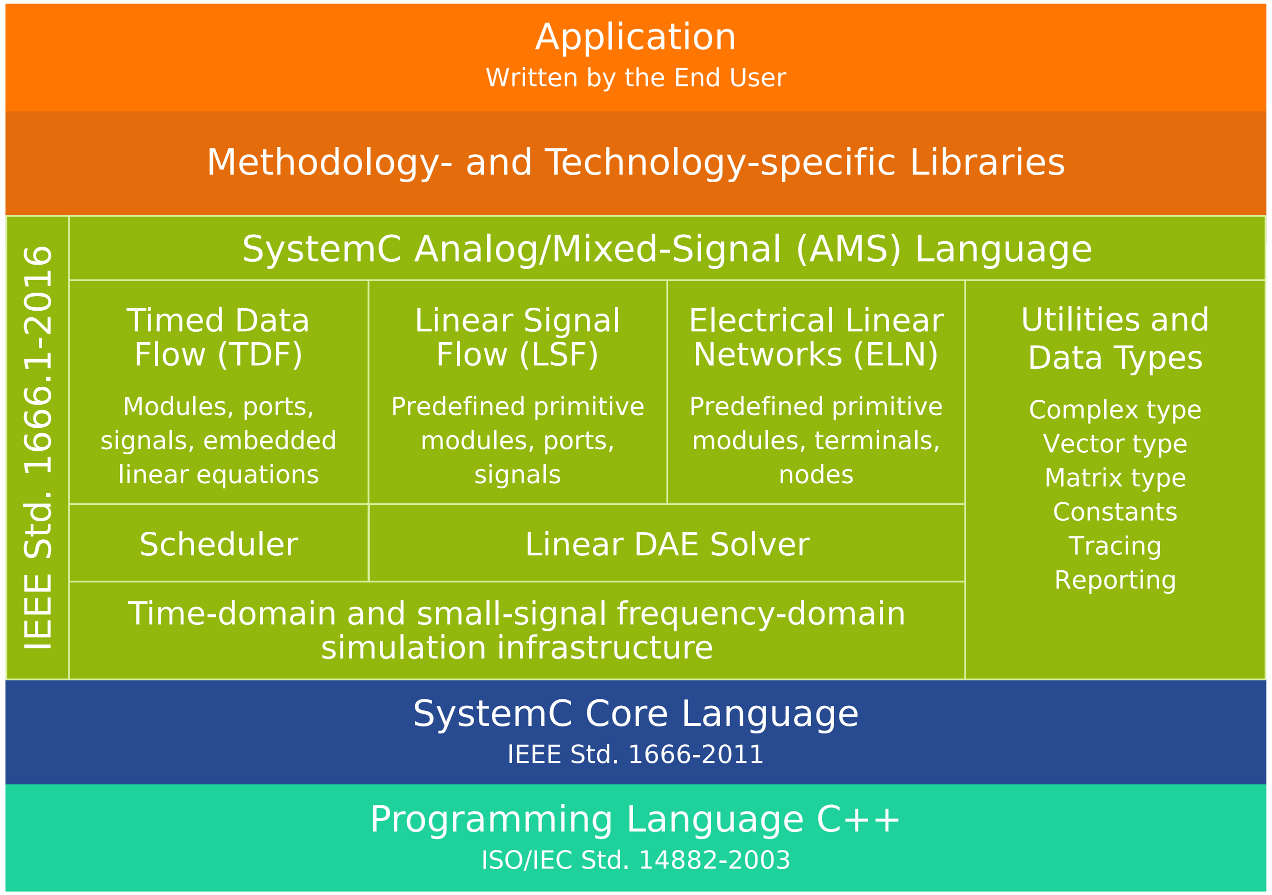
- Simulation performance improvements under development for types sc_bigint and sc_biguint
- Current focus is on
  - Benchmarking and optimization
  - Functional correctness
- Aim is to have limited or no changes to the SystemC language standard
  - Different implementation configurations are being explored
- New data types being considered: sc_complex and sc_float

SystemC Analog/Mixed-Signal WG

- The SystemC AMS Working Group is responsible for the standardization of the SystemC AMS extensions, defining and developing the language, methodology and class libraries for analog, mixed-signal and RF modeling in SystemC
- **Current status**
  - IEEE Std. 1666.1-2016 is available at no cost, sponsored by Accellera, under the IEEE GET Program
  - SystemC AMS Proof-of-concept version 2.3 made available via Accellera member company
  - SystemC AMS getting started material available: User's Guide and application examples
- **Developments & future plans**
  - Finalizing updates of the SystemC AMS regression suite, released by Accellera in coming period
  - First proposals created for new SystemC AMS language enhancements and features for next IEEE P1666.1 revision

SystemC Analog/Mixed-Signal WG

- SystemC AMS defines 3 additional models of computation focusing on efficient AMS / RF system-level modeling concepts
  - Timed Data Flow (TDF)
  - Linear Signal Flow (LSF)
  - Electrical Linear Networks (ELN)
- Practical SystemC AMS User's Guide and application examples explaining the language constructs and execution semantics in detail

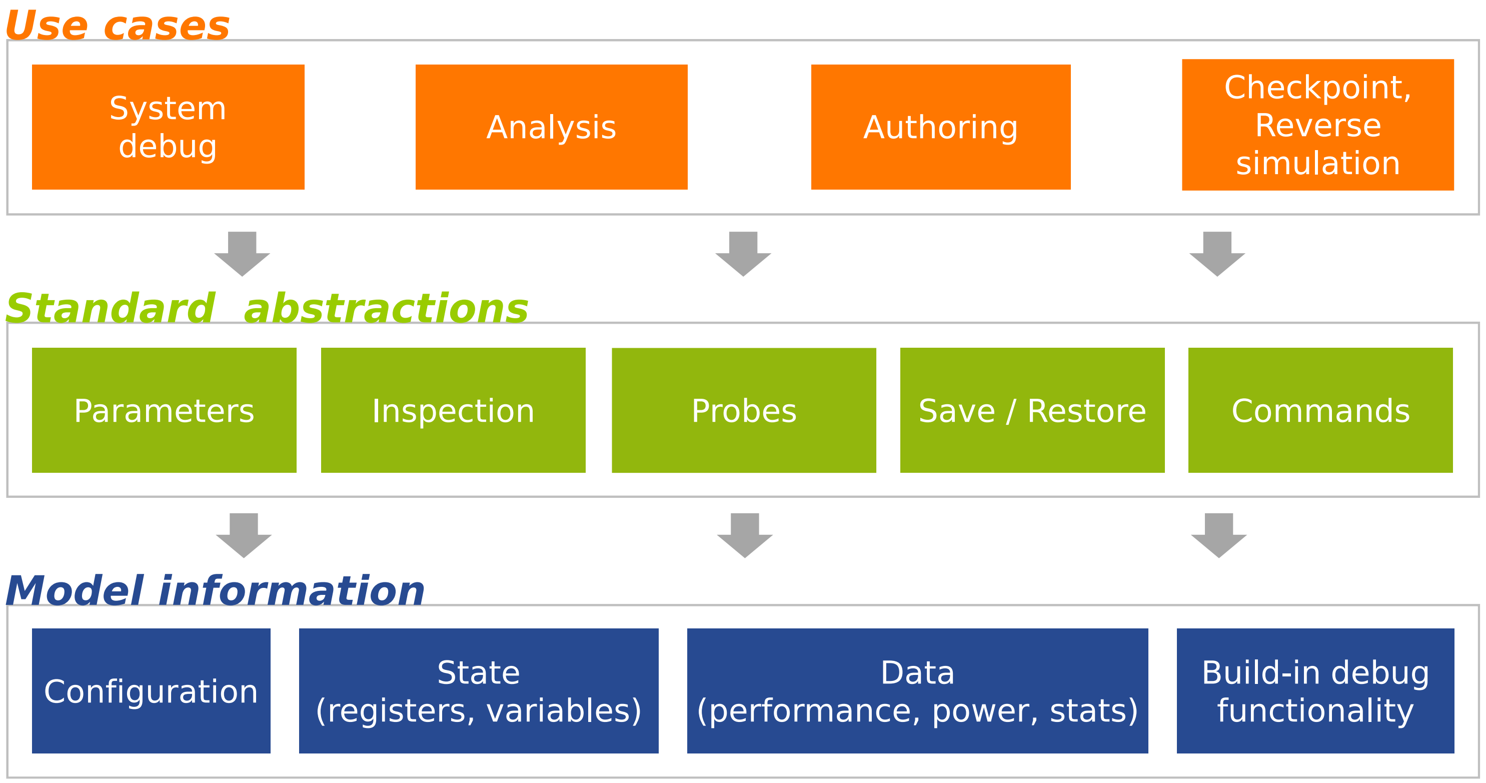

Application
Written by the End User
Methodology- and Technology-specific Libraries
IEEE Std. 1666.1-2016
SystemC Analog/Mixed-Signal (AMS) Language
Timed Data Flow (TDF)
Modules, ports, signals, embedded linear equations
Linear Signal Flow (LSF)
Predefined primitive modules, ports, signals
Electrical Linear Networks (ELN)
Predefined primitive modules, terminals, nodes
Utilities and Data Types
Complex type
Vector type
Matrix type
Constants
Tracing
Reporting
Scheduler
Linear DAE Solver
Time-domain and small-signal frequency-domain simulation infrastructure
SystemC Core Language
IEEE Std. 1666-2011
Programming Language C++
ISO/IEC Std. 14882-2003

SystemC Configuration, Control & Inspection WG

- The SystemC Configuration, Control and Inspection WG is responsible for developing standards that allow tools to interact with models in order to perform activities such as debug, analysis, authoring and checkpointing
- **Current status**
  - SystemC CCI 1.0 Language Reference Manual released as Accellera standard in 2018
  - CCI 1.0 standard supported by Proof-of-Concept Kit including reference implementation and examples
- **Developments and future plans**
  - Updates and clean-up of the CCI Proof-of-Concept Kit, compatible with other SystemC reference implementations, preparing release CCI 1.0.1
  - Definition of Inspection API ongoing, in close collaboration SystemC Common Practices WG which is reviewing register contributions

SystemC Configuration, Control & Inspection WG

- CCI 1.0 covers standardized interfaces for parameters
- Current focus on Inspection API



SystemC Synthesis WG

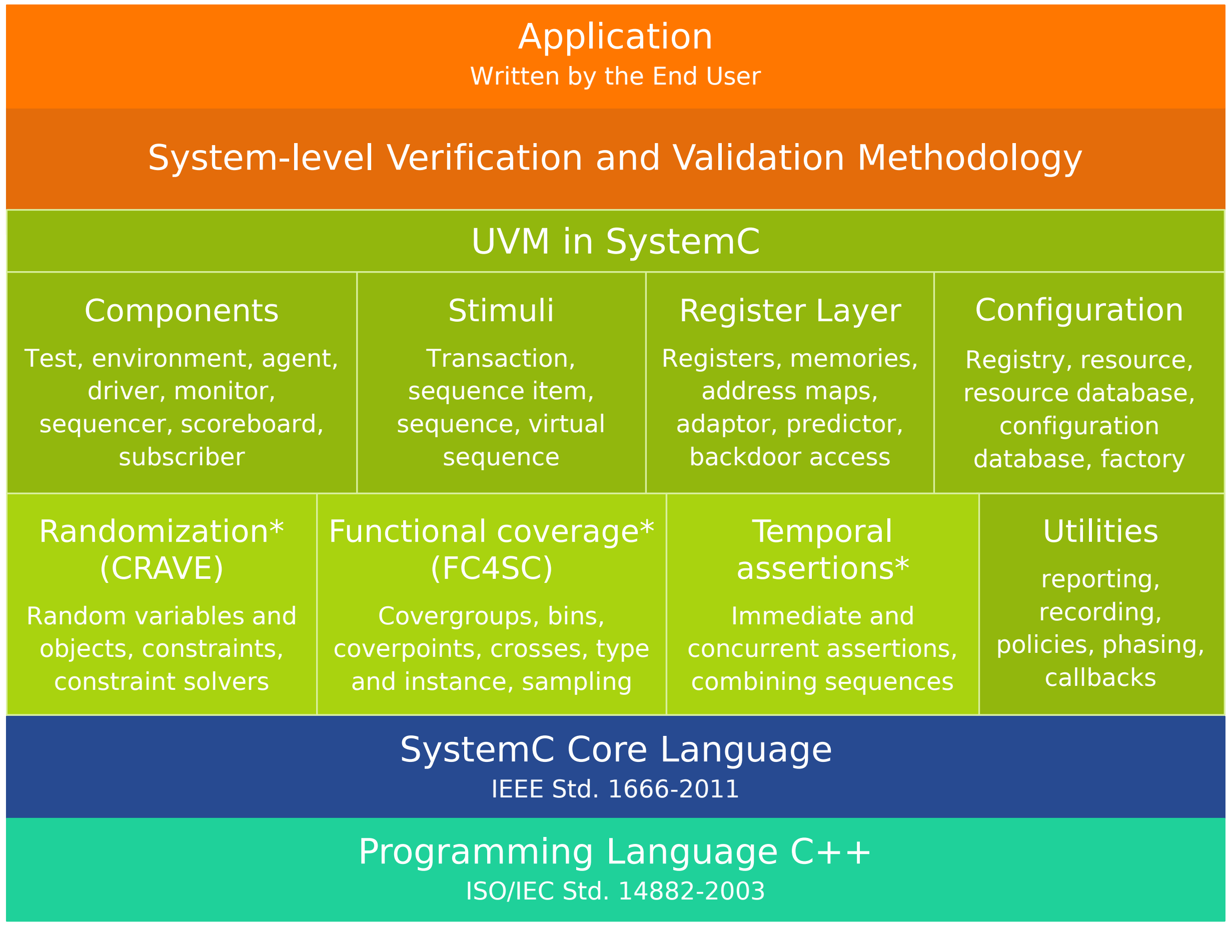
- The SystemC Synthesis Working Group is responsible for the SystemC synthesizable subset, to enable synthesis of digital hardware from high-level specifications
- **Current status**
  - Released the SystemC Synthesis Subset Language Reference Manual version 1.4.7 in 2017
- **Developments and future plans**
  - Working Group defining next revision of the SystemC Synthesizable Subset, including:
  - Alignment and consolidation on SystemC Datatypes to enhance HLS flows
  - Update and finalize support of modern C++ language features defined in C++11/14/17

SystemC Verification Working Group

- The SystemC Verification WG is responsible for defining verification extensions to the SystemC standard and reference implementation by offering an add-on libraries to ease the deployment of a verification methodology based on SystemC
- **Current Status**
  - UVM in SystemC (UVM-SystemC) standard and reference implementation 1.0beta3 released in 2020
  - SystemC Verification Library version 2.0.1 in maintenance mode
  - DVCon U.S. 2021 Video available: UVM-SystemC Randomization - Updates From The SystemC Verification WG
- **Developments of future plans**
  - Finalizing UVM-SystemC library 1.0beta4, planned for later this year
  - Standardization of API for Constrained Randomization(CRAVE) and Functional Coverage (FC4SC)
  - Initial discussions on Temporal Assertions in the SystemC language started

SystemC Verification Working Group

- The UVM-SystemC library enables the creation of a modular, scalable, configurable and reusable testbenches
  - Following the principles of the Universal Verification Methodology (UVM)
  - Implemented in C++/SystemC, offering flexibility and reuse across verification and validation domains
- Additional verification-specific features such as constrained randomization and functional coverage will be addressed by supporting add-on libraries such as CRAVE and FC4SC

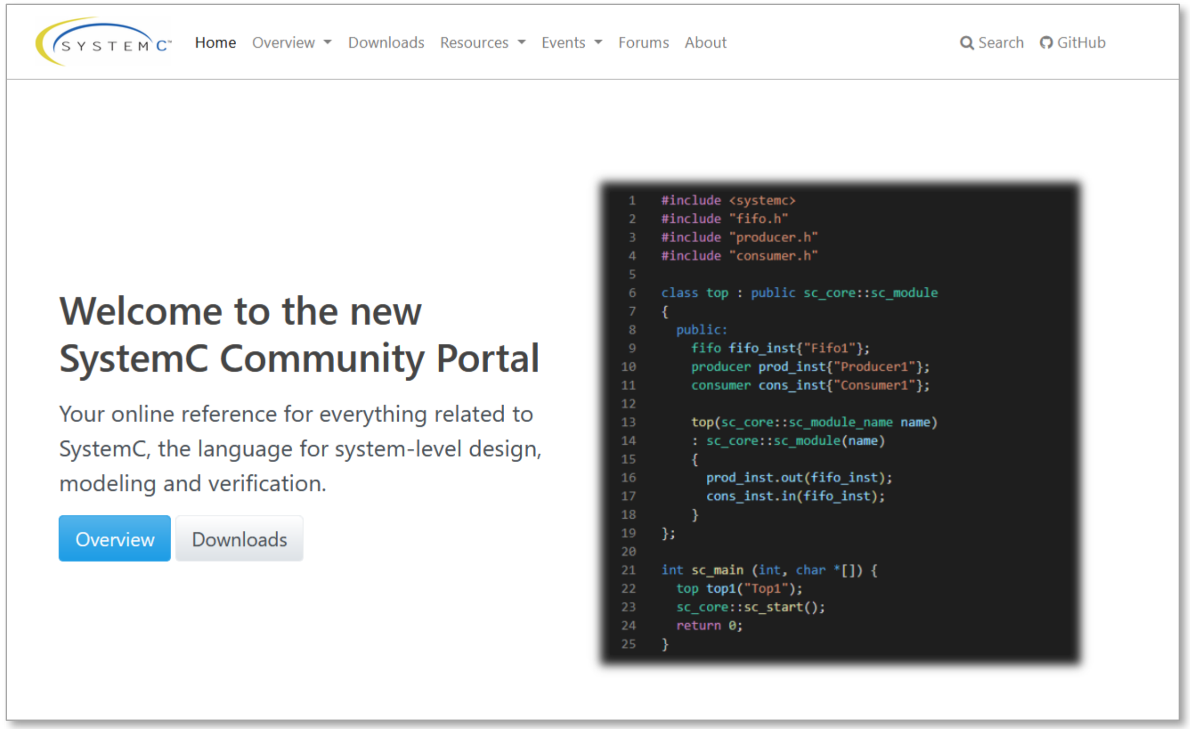


Application
Written by the End User
System-level Verification and Validation Methodology
UVM in SystemC
Components
Test, environment, agent, driver, monitor, sequencer, scoreboard, subscriber
Stimuli
Transaction, sequence item, sequence, virtual sequence
Register Layer
Registers, memories, address maps, adaptor, predictor, backdoor access
Configuration
Registry, resource, resource database, configuration database, factory
Randomization* (CRAVE)
Random variables and objects, constraints, constraint solvers
Functional coverage* (FC4SC)
Covergroups, bins, coverpoints, crosses, type and instance, sampling
Temporal assertions*
Immediate and concurrent assertions, combining sequences
Utilities
reporting, recording, policies, phasing, callbacks
SystemC Core Language
IEEE Std. 1666-2011
Programming Language C++
ISO/IEC Std. 14882-2003

* Integration on Roadmap

Contribute to systemc.org

- New SystemC Community Portal
  - Download standards and reference implementations
  - Resources (Books, Project, Tutorials, Videos, …)
  - Upcoming and past events
  - Link to Discussion Forum (link)
  - Link to Accellera public GitHub repository
  - Common practices and community libraries
- **YOU** can help in adding content!
  - Submit your pull request to github.com/accellera-official/systemc.org



Welcome to the new SystemC Community Portal

Your online reference for everything related to SystemC, the language for system-level design, modeling and verification.

```cpp
#include <systemc>
#include "fifo.h"
#include "producer.h"
#include "consumer.h"

class top : public sc_core::sc_module
{
  public:
    fifo fifo_inst{"Fifo1"};
    producer prod_inst{"Producer1"};
    consumer cons_inst{"Consumer1"};

    top(sc_core::sc_module_name name)
    : sc_core::sc_module(name)
    {
      prod_inst.out(fifo_inst);
      cons_inst.in(fifo_inst);
    }
};

int sc_main (int, char *[]) {
  top top1("Top1");
  sc_core::sc_start();
  return 0;
}
```

IEEE related Working Groups

- P1666
  - IEEE Standard for Standard SystemC Language Reference Manual Working Group (LWG)
  - Latest version: IEEE 1666-2011, published 2012-01-09
  - Chair: Jerome Cornet (ST Microelectronics)
  - P1666 WG currently **active**, standardization of next 1666 revision ongoing
- P1666.1
  - IEEE Standard for Standard SystemC(R) Analog/Mixed-Signal Extensions Language Reference Manual
  - Latest version: IEEE 1666.1-2016, Published 2016-04-06
  - Chair: Martin Barnasconi (NXP)
  - P1666.1 WG not active, will restart in ~2023

Advancing SystemC Standards Together

- Become an Accellera Working Group member
  - Join Accellera and participate in the Accellera working groups
  - Direct access to the latest standardization proposals and development tree
- Become a member of the IEEE Standards Association
  - Join IEEE-SA to participate in the *ongoing* standardization in the P1666 (SystemC) working group
- Share your experiences
  - Visit www.accellera.org and join the community forums at forums.accellera.org
  - Report your issues and/or create pull requests on the public SystemC GitHub repository
- Help us to grow the SystemC ecosystem and community
  - Participate in community events such as the SystemC Evolution Day
  - Contribute to the SystemC Community Portal systemc.org
  - Promote the use of the SystemC standard in complex system simulation tasks

Thank You

Q&A