The Evolution of SystemC AMS

Martin Barnasconi

Accellera SystemC AMS Working Group Chair

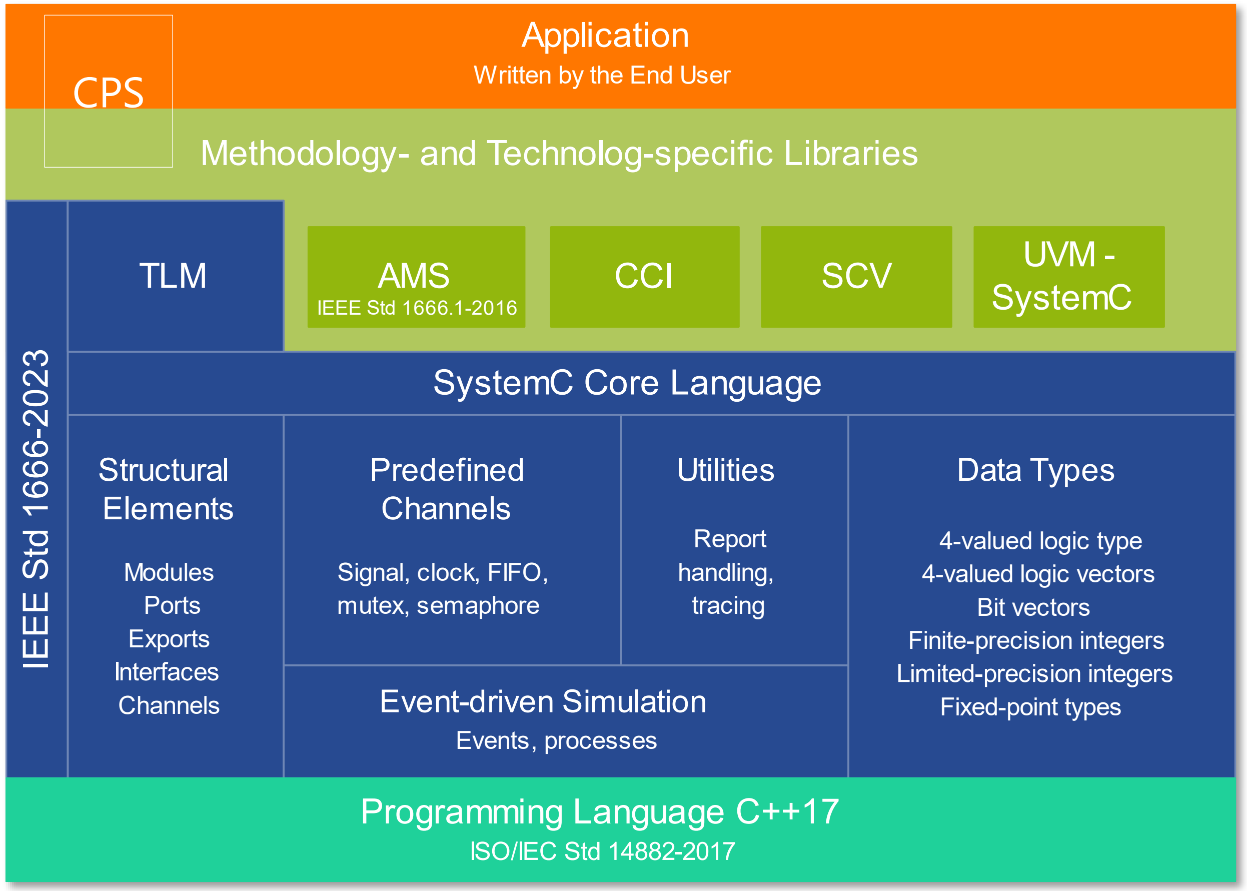
accellera.org

Outline

- The SystemC ecosystem
- Historical perspective on the SystemC AMS extensions
- *Intermezzo: (the truth about) Real Number Modeling*
- Industry trends
- SystemC AMS applications - an overview
- Future perspective on SystemC AMS
- Summary

The SystemC ecosystem

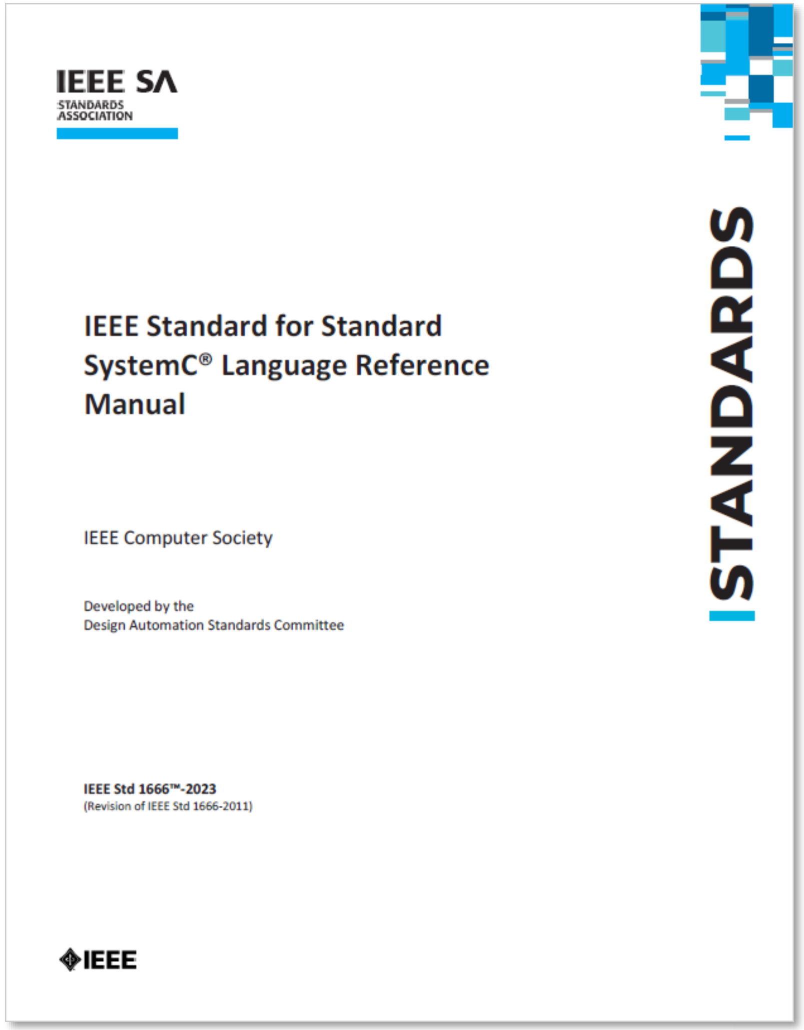
- SystemC is a C++-based language standard, widely used for
  - System-level modeling, design and verification
  - Architectural exploration, performance modeling
  - Analog/mixed signal modeling
  - High-level Synthesis (HLS)
  - HW/SW co-design
- Released as IEEE standards
  - IEEE Std. 1666-2023 (SystemC)
  - IEEE Std. 1666.1-2016 (SystemC AMS)



More information: https://systemc.org/

SystemC - IEEE Std 1666-2023

- Latest revision of the SystemC Language Reference Manual released in Sept 2023
- Standard can be downloaded at no cost under the IEEE Get Program thanks to Accellera sponsorship
- Highlights
  - Based on C++17 (ISO/IEC 14882:2017)
  - Simulation stage callbacks
  - Suspend/unsuspend kernel to enable interaction with external OS threads
  - Time resolution down to yoctoseconds (10^{-24})
  - More info: systemc.org



IEEE SA
STANDARDS ASSOCIATION

IEEE Standard for Standard SystemC® Language Reference Manual

IEEE Computer Society

Developed by the
Design Automation Standards Committee

IEEE Std 1666™-2023
(Revision of IEEE Std 1666-2011)

IEEE

STANDARDS

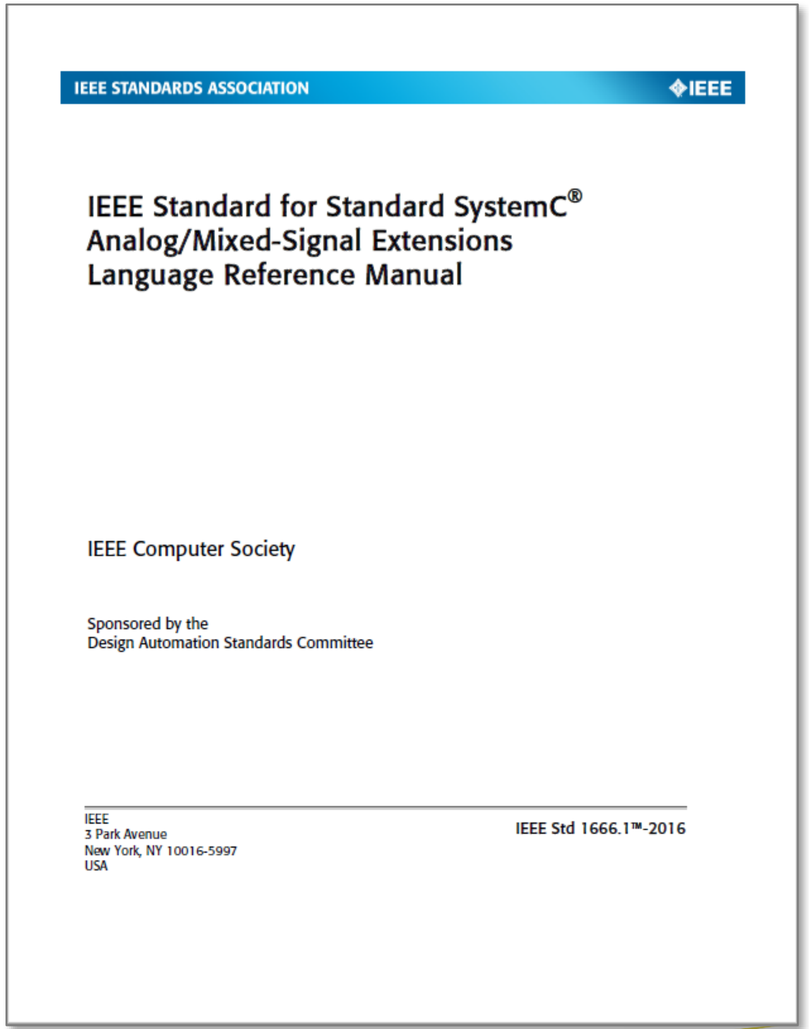
Latest news: SystemC 3.0.0 public review!

Fork me on GitHub

- SystemC 3.0.0 public review version has been released by Accellera in Dec 2023
  - Open source reference implementation licensed under Apache 2.0
  - Fully compliant with IEEE Std 1666-2023
  - Now available for download via Accellera public repository on GitHub: https://github.com/accellera-official/systemc/releases/tag/3.0.0_pub_rev_20231129
  - SystemC Community is invited to review and test SystemC 3.0.0 and provide feedback
  - Final version of SystemC 3.0.0 expected later this year
- Next steps of the Accellera SystemC Working Group
  - Integrate SystemC tests into main SystemC repository.
  - Establishing CI/CD flow in the Accellera public repository on GitHub
  - Start collecting inputs and requirements for next standardization cycle

SystemC AMS - IEEE Std 1666.1-2016

- SystemC Analog/Mixed-Signal (AMS) extensions released as IEEE standard in 2016
- Standard can be downloaded at no cost under the IEEE Get Program thanks to Accellera sponsorship
- Highlights
  - Based on C++03 (ISO/IEC 14882:2003)
  - Introducing models of computation: Timed Data Flow, Linear signal Flow, Electrical Linear Networks
  - Time domain and small signal frequency domain analysis
  - Tracing of AMS signals and traceable objects
- Next revision planned for 2026


IEEE STANDARDS ASSOCIATION ◆IEEE

IEEE Standard for Standard SystemC®
Analog/Mixed-Signal Extensions
Language Reference Manual

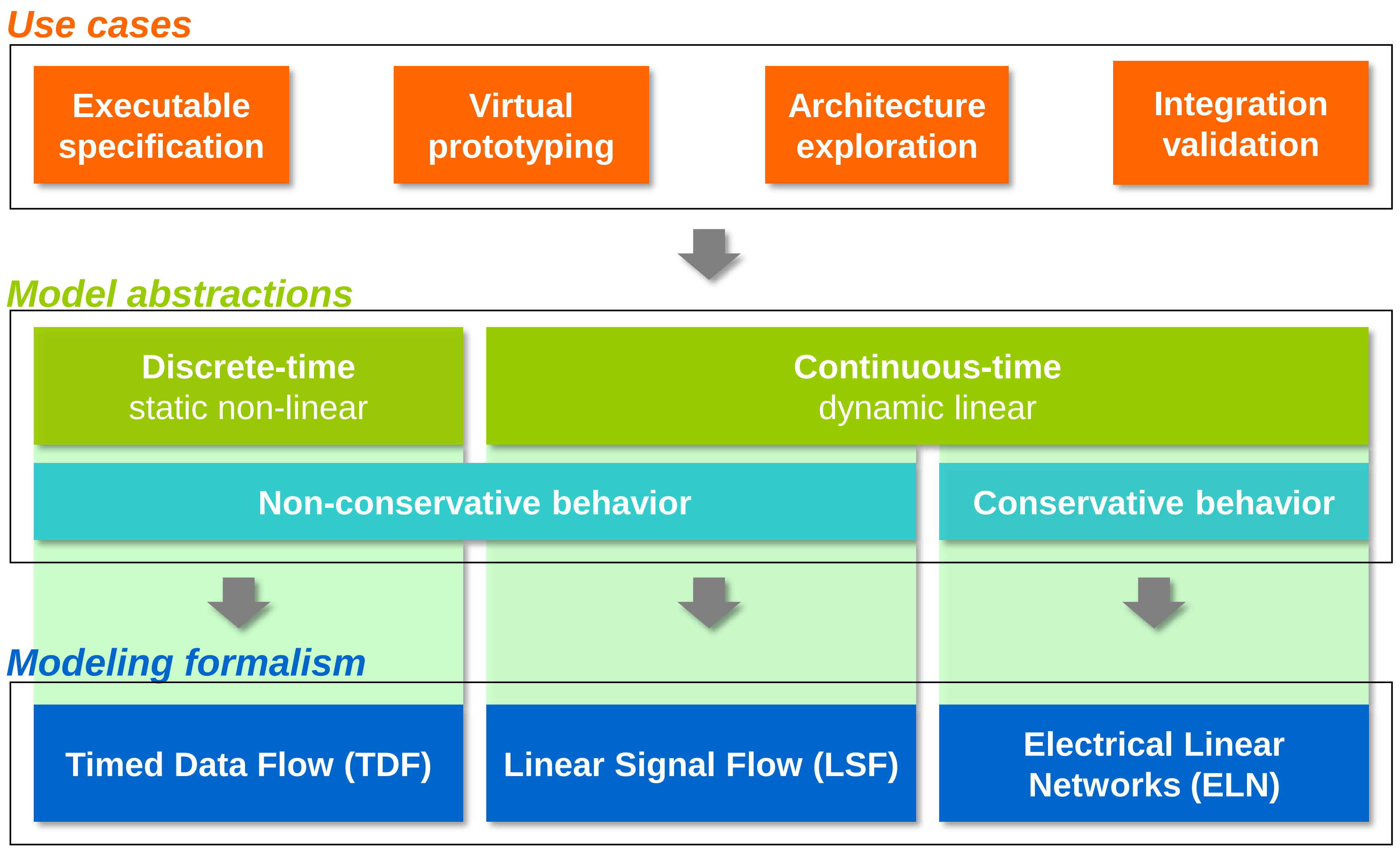
IEEE Computer Society

Sponsored by the
Design Automation Standards Committee

IEEE
3 Park Avenue
New York, NY 10016-5997
USA

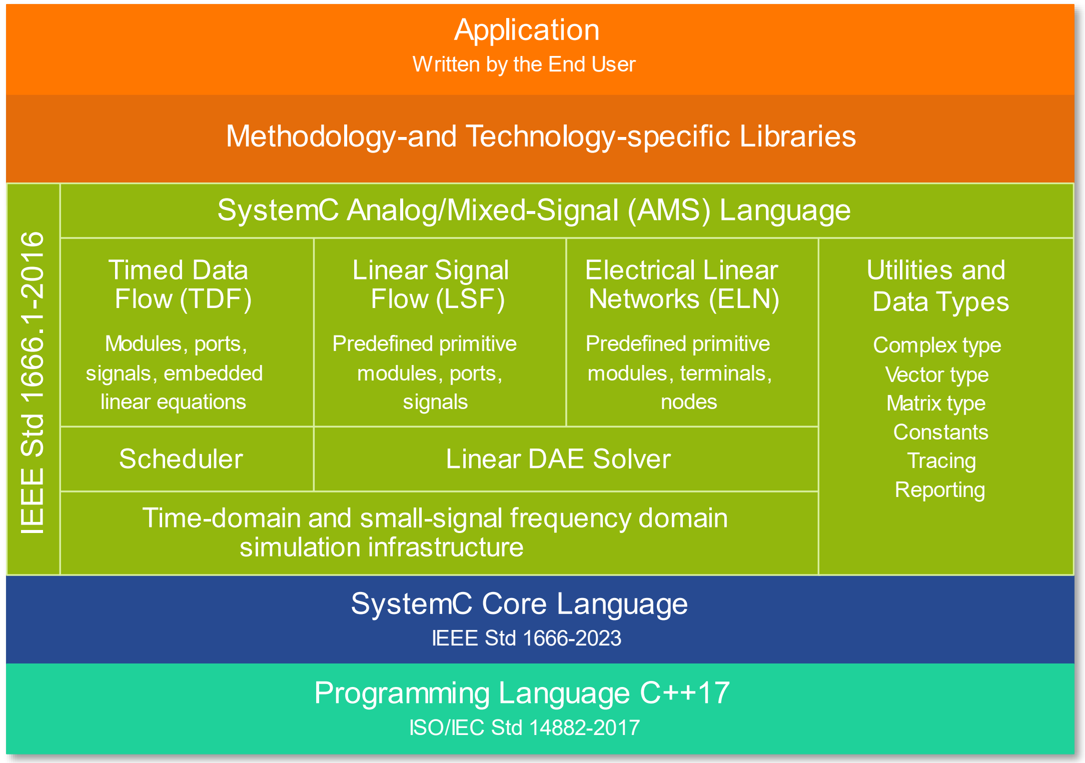
IEEE Std 1666.1™-2016

SystemC AMS model abstraction and formalisms



SystemC AMS technology developments

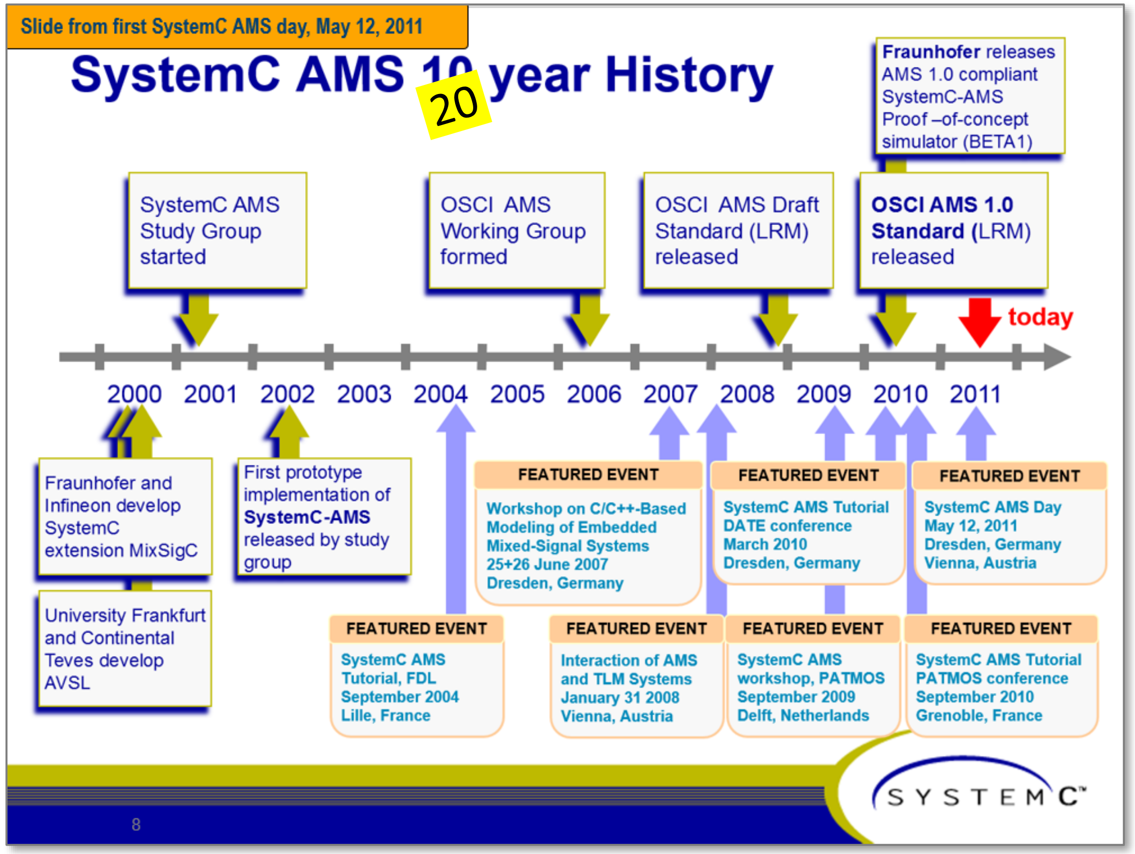
- SystemC AMS proof-of-concept implementation is offered by COSEDA, available under Apache 2.0 license
  - Implementation can be compiled against any SystemC IEEE Std 1666 compatible simulator (e.g., commercially available EDA tools)
- Accellera SystemC AMS Working Group released supplemental material
  - SystemC AMS Users Guide update in 2020
  - SystemC AMS regression suite released in July 2023, covering more than 700 unit-level and application-level tests and examples


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 1666-2023
Programming Language C++17
ISO/IEC Std 14882-2017

More information
https://systemc.org/overview/systemc-ams/

Historical perspective on SystemC AMS

- 20+ years of development and knowhow
- Main drivers (in those days)
  - Unified and standardized modeling language for system/architecture design
  - System modeling and simulation technologies to create mixed-signal virtual prototypes
  - Foundation for development of AMS system-level design tools
- Key concepts
  - Modeling and simulation at higher level of abstraction (hence use of data flow MoC)
  - Avoid unnecessary synchronization and events in and between analog and digital domain


Slide from first SystemC AMS day, May 12, 2011

SystemC AMS 10 year History

20

Fraunhofer releases AMS 1.0 compliant SystemC-AMS Proof –of-concept simulator (BETA1)

SystemC AMS Study Group started

OSCI AMS Working Group formed

OSCI AMS Draft Standard (LRM) released

OSCI AMS 1.0 Standard (LRM) released

today

2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011

Fraunhofer and Infineon develop SystemC extension MixSigC

University Frankfurt and Continental Teves develop AVSL

First prototype implementation of **SystemC-AMS** released by study group

FEATURED EVENT
SystemC AMS Tutorial, FDL
September 2004
Lille, France

FEATURED EVENT
Workshop on C/C++-Based Modeling of Embedded Mixed-Signal Systems
25+26 June 2007
Dresden, Germany

FEATURED EVENT
Interaction of AMS and TLM Systems
January 31 2008
Vienna, Austria

FEATURED EVENT
SystemC AMS Tutorial
DATE conference
March 2010
Dresden, Germany

FEATURED EVENT
SystemC AMS workshop, PATMOS
September 2009
Delft, Netherlands

FEATURED EVENT
SystemC AMS Day
May 12, 2011
Dresden, Germany
Vienna, Austria

FEATURED EVENT
SystemC AMS Tutorial
PATMOS conference
September 2010
Grenoble, France

8

Slide from COSEDA User Group meeting October 2014

Historical perspective – Recognized features

Simulation Speed	• Simulation speed improvements between 10 – 10^6 x (depending on selected modeling concept and abstraction level) • Faster than traditional discrete-event real-number-modeling approaches • Faster than (most) commercial signal flow and/or data flow solutions
Scalability	• Modeling and simulation of complex heterogeneous designs blocks (e.g. SDPLL) • Enabling modeling and simulation of complex (RF) ASIC or SoC at sufficient level of detail (e.g. end-to-end simulation of RF system) • Scenario modeling and simulation of systems-of-systems (e.g. power plant/grid)
Extendibility	• Leveraging C++ functions / libraries to enhance modeling capabilities • Open source ecosystem facilitates flow automation and extensions to algorithm design (e.g. use of Python) • C++/SystemC enables seamless integration in (most) commercial EDA solutions

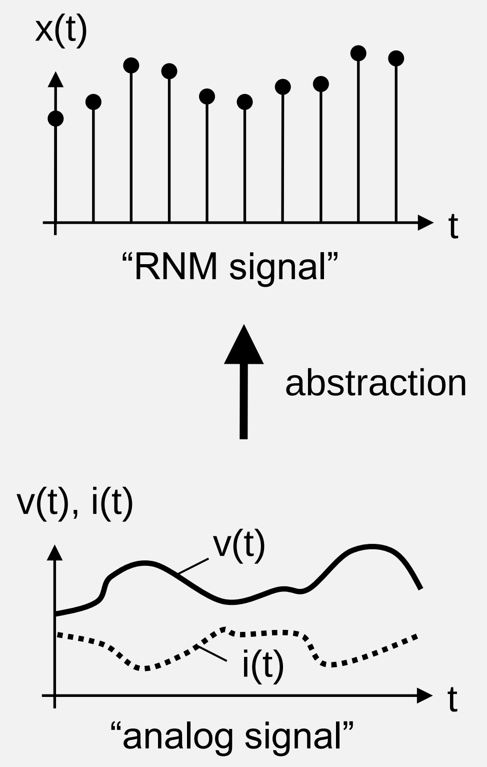
Historical perspective – Perceived constraints

Fixed Time step	• Not all continuous-time behavior can be mapped on a static, fixed time step • SystemC AMS 2.0 and IEEE Std 1666.1 introduce dynamic time step ("Dynamic TDF"), but this is often perceived as complex and difficult to setup • Tendency to use SystemC discrete-event models to address dynamic time step
Linear Solver	• Limitations when modeling dynamic non-linear systems • Partitioning over dynamic linear and static non-linear in combination with piecewise linear approximation techniques not always feasible • SystemC AMS standard does not offer an API to integrate a 3rd party solver
Threshold detection	• No modeling primitive for threshold detection of continuous-time signals • Alternative approach to delay the continuous-time signal to calculate the exact threshold crossing is not always feasible • Dynamic time step could be used for this, but often seen as complex to model

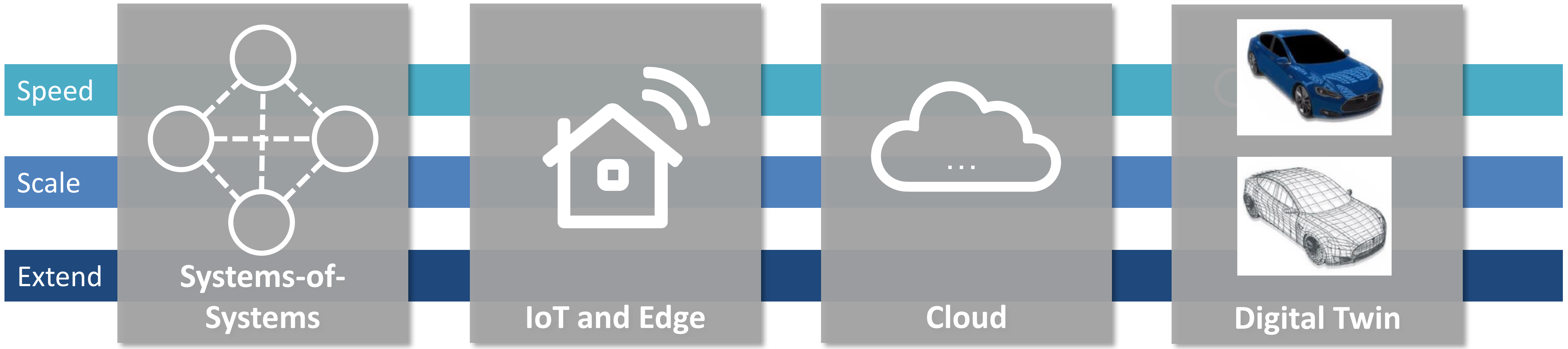
NOTE: Constraints are there for a reason:
Avoid unnecessary synchronization and events

Intermezzo: (The truth about) Real Number Modeling

- Real Number Modeling is a concept where a continuous-time signal is abstracted to a time-discrete signal with a real-value
- Most ‘digital’ modeling languages (except Verilog) support RNM
  - SystemC (double), SystemVerilog (real), Verilog-AMS (wreal), VHDL (REAL)
  - RNM models often rely on a discrete-event (‘digital’) simulator. This means each ‘sample’ of the continuous-time signal will generate an event
- Although RNM could be significantly faster than traditional analog simulation, it has drawbacks
  - The bigger the system, the slower it gets (more models, more events)
  - For RF system modeling, oversampling will kill simulation speed
- Conclusion: using discrete-event for RNM is not scalable
  - Instead, SystemC AMS recommends using Timed Data Flow for RNM



Industry trends – Overall industry landscape



Examples:
- Cyber Physical Electrical Energy Systems (CPEES)
- Advanced Driver-Assistance System (ADAS)

Examples:
- IoT sensor nodes and Edge devices (e.g. smart thermostat, smart doorbell, etc.)

Examples:
- Intensive compute in the cloud
- App stores
- OTA SW updates

Examples:
- Virtual ECU/Car/… to enable early testing, validation, SW development
- Hardware/Model/Software in the Loop

SystemC AMS applications - an overview

Automotive Radar Applications

- System modeling of radar front-end incl. relevant RF impairments
- Using specialized C++ data types for RF signals (e.g. envelope, polar,…)

Power Management Applications

- Automated system simulations of hundreds of mixed-signal use cases and scenario's
- Detailed analog behaviour models of the IC and its application

Sensor Applications

- High-performance, high-precision sensors for automotive applications
- Perform x-sigma Monte Carlo to prove the architecture and design

Future perspective on SystemC AMS

- The Accellera SystemC AMS Working Group is currently developing extensions and enhancements as preparation for the next IEEE 1666.1 update (~2026)
  - Incorporating feedback from the user base
  - Assess the needs for the coming decade in terms of emerging industrial applications and use cases
- SystemC AMS features under development / consideration
  - Analog solver API
  - Introduction of converter primitives between LSF and ELN model of computation
  - Interactive tracing and debug interface, tracing customization
  - Threshold detection
- Collaboration with the Accellera SystemC Working Group to define a common infrastructure and foundation as part of the SystemC core language and standard

Future perspective – Standardization plan

A joined Accellera and IEEE-SA standardization journey *

2023	2024	2025	2026
In Accellera: • Define SystemC AMS feature set • Propose language and API	In IEEE-SA: • Initiate P1666 standardization project • P1666 Working Group formation In Accellera: • Finalize SystemC AMS feature set • Handover of draft standard updates to IEEE	In IEEE-SA: • Technical editing Language Reference Manual • Document review & ballot In Accellera: • Validate new language with regressions and examples	In IEEE-SA: • Release IEEE 1666.1-2026 In Accellera: • Release regression suite and/or examples

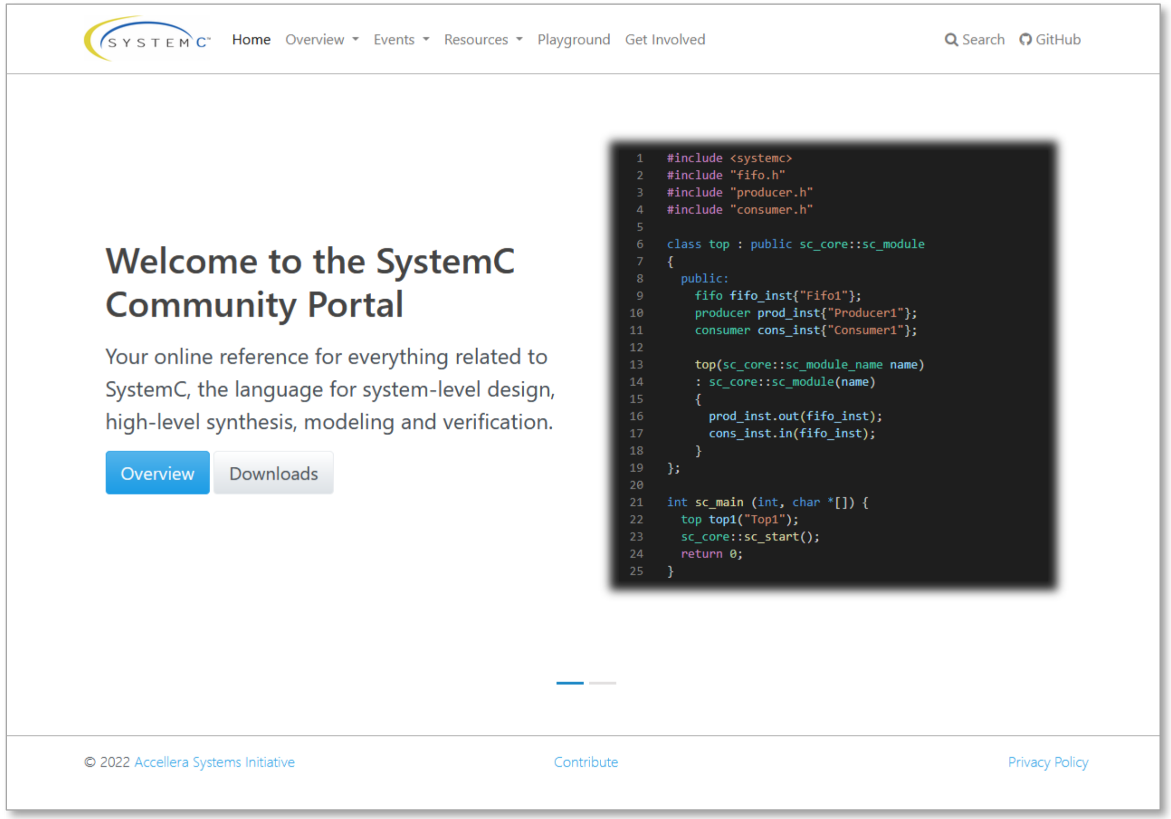
* Tentative planning, Subject to change

Summary

- SystemC ecosystem continues to evolve
  - Revision of IEEE 1666 and Accellera reference implementation v3.0.0 released in 2023
  - SystemC AMS one of the many methodology and technology libraries built on top of SystemC
- The SystemC AMS extensions offers a mature, stable and proven solution
  - Standardized in IEEE 1666.1-2016, available as proof-of-concept simulator and supported by an EDA tool provider
  - Continued effort to balance simulation speed, scalability and extendibility against modeling fidelity and accuracy
- Industry moves to complex multi-domain systems-of-systems containing more SW
  - Additional SystemC AMS extensions and enhancements defined to address systems engineering, modeling and simulation needs for the coming decade
  - Leverage SystemC core language to enable full systems-of-systems and HW/SW co-design

SystemC Community Portal - systemc.org

- Highlights
  - SystemC overview pages covering all Working Groups
  - SystemC Evolution Day Events and Fikas: all presentations and videos
  - Open Access Publications
  - Libraries and Projects
- **YOU** can help in adding content!
  - Submit your pull request to github.com/accellera-official/systemc.org



Welcome to the SystemC Community Portal

Your online reference for everything related to SystemC, the language for system-level design, high-level synthesis, 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;
}
```

© 2022 Accellera Systems Initiative   Contribute   Privacy Policy

Thank You

Q&A