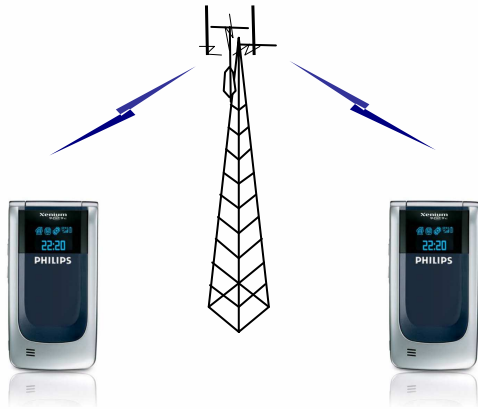




SystemC Analog & Mixed Signal Extensions: What's It All About?

**Martin Barnasconi, AMS WG Chairman
NASCUG IX – June 9, 2008**

Embedded Analog/Mixed-Signal Applications



Telecommunications

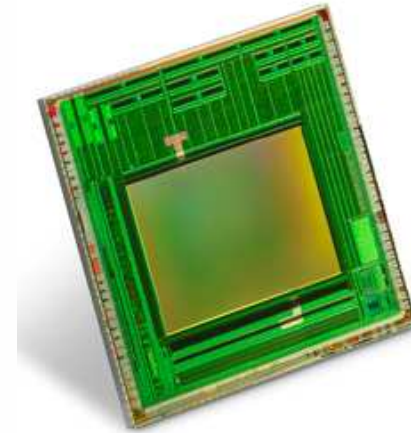
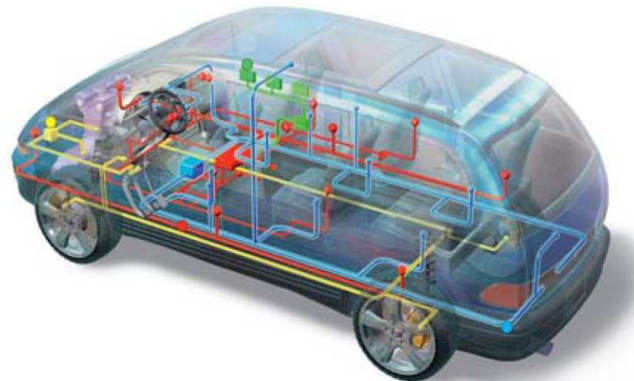


Image courtesy of STMicroelectronics

Imaging sensors

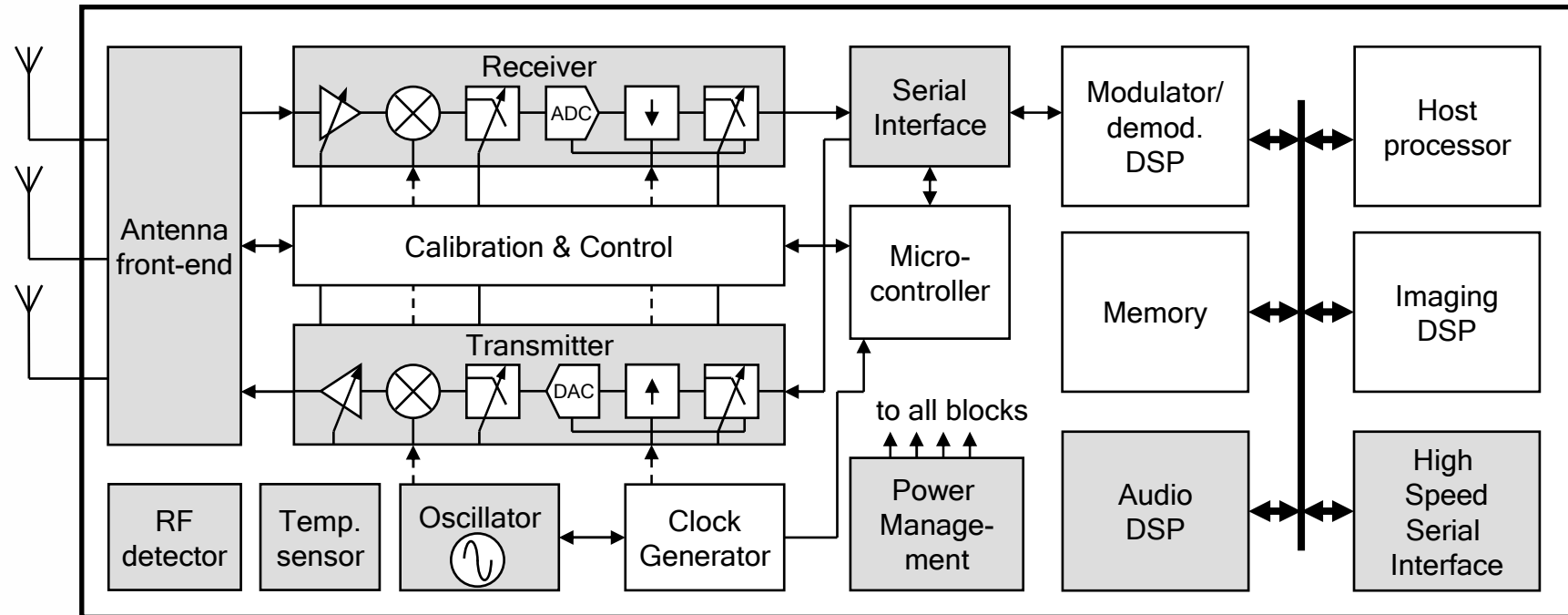


Automotive

Overview

- Embedded AMS systems...a closer look
- OSCI AMS Working Group
- *Intermezzo*
- Requirements for the SystemC AMS Extensions
- The SystemC AMS Extensions - explained
- *Bonus*: Code examples
- Conclusions
- Acknowledgements
- What's next...

Embedded AMS systems – a closer look...



- **Tight interaction between digital HW/SW and AMS sub-systems**
 - Control path: more and more HW/SW calibration and control of analog blocks
 - Signal path: ISO OSI protocol stack – modeling including PHY layer

Why having AMS extensions for SystemC?

- **Missing is**

- An agreed system modeling language and methodology to design Embedded AMS systems
- An architecture design tool for AMS system-level design and verification
- A platform that facilitates AMS model exchange and reuse of intellectual property (IP)
- An open modeling and programming interface between AMS and digital HW/SW system descriptions

- **It's time to standardize AMS extensions for SystemC !**

- Open SystemC Initiative will drive standardization, deployment and support of the SystemC AMS extensions
- Targeting an open source standard for system-level design for Embedded AMS systems

AMS WG applications and use cases

- Embedded analog/
mixed-signal systems
 - Heterogeneous systems including analog, mixed-signal, RF and digital HW/SW

- Application domains
 - Wireless
 - Wired
 - Automotive
 - Imaging sensors

- Use cases

- Virtual prototyping for SW development
- Creating reference models for functional verification
- Architecture exploration, definition and algorithm validation

End Product Markets	2003	2004	2005	2006	2007
Microprocessor/DSP	18.9%	16.0%	13.1%	10.5%	14.7%
Computer, Peripheral	22.9%	21.6%	18.5%	24.2%	19.0%
Wired Network	11.2%	5.2%	5.8%	4.8%	5.2%
Wireless Network	13.1%	10.4%	13.1%	7.3%	6.9%
Multimedia	25.6%	34.2%	33.8%	37.9%	31.9%
Automotive	1.9%	3.0%	3.8%	4.0%	4.3%
Others	6.4%	9.7%	11.9%	11.3%	18.1%

source: SystemC Trends report, April 2007

focus of AMS WG



Current OSCI AMS WG Roster



- **37 individuals from 17 organizations**
 - Strong drive from semiconductor industry
 - Full support of universities and research institutes
 - Growing interest and participation of EDA/ESL vendors
- **Chair: Martin Barnasconi, NXP Semiconductors**
Vice chair: Christoph Grimm, Vienna University of Technology

OSCI AMS WG charter & objectives

■ Charter

- The Analog/Mixed-Signal (AMS) Working Group develops and recommends techniques and provides *AMS extensions* to the SystemC language standard
- Promoting the modeling of heterogeneous systems including both continuous-time and discrete-event behaviors at architectural level

■ Objectives

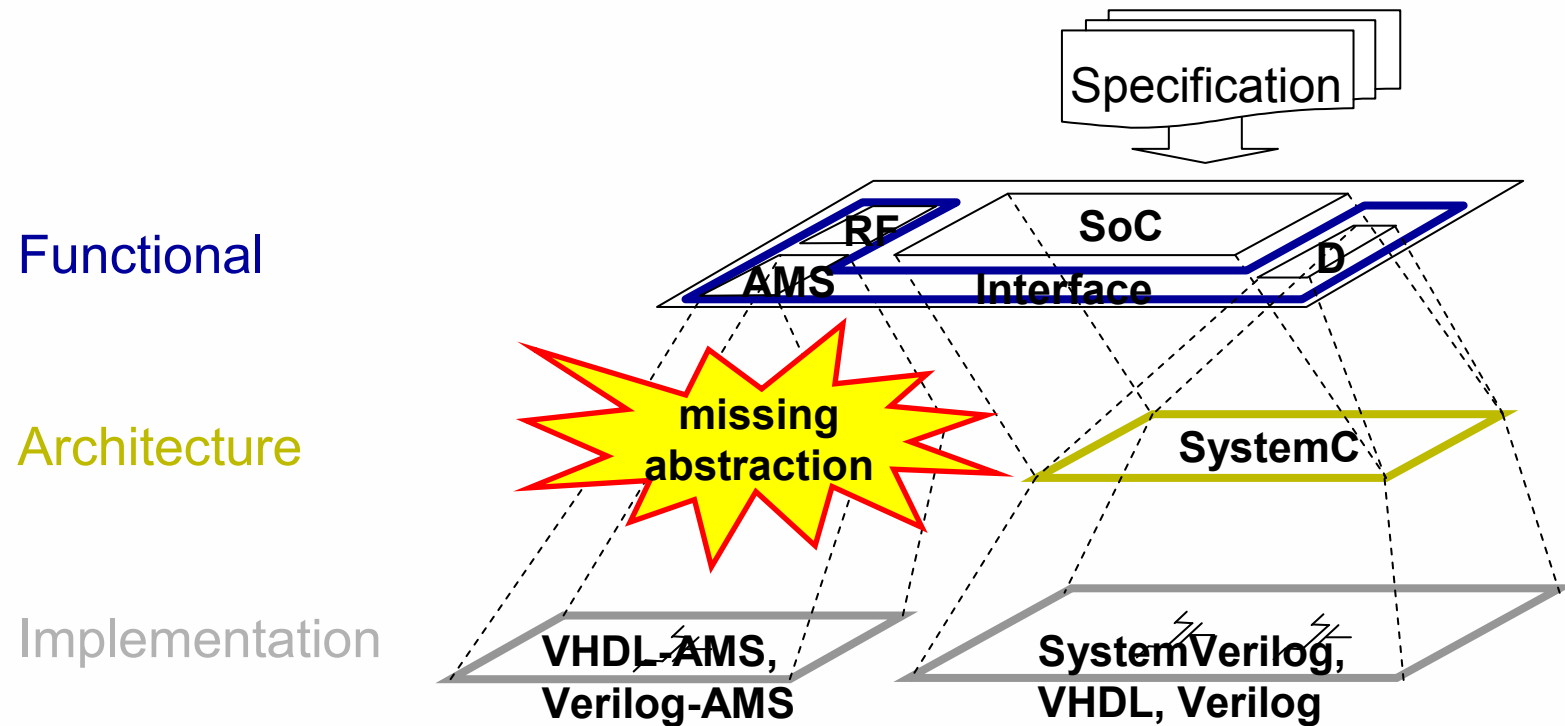
- Analyze and standardize extensions of SystemC with a semantic for describing non-conservative and conservative systems with continuous-time descriptions for electrical or non-electrical domains

Planning and timing

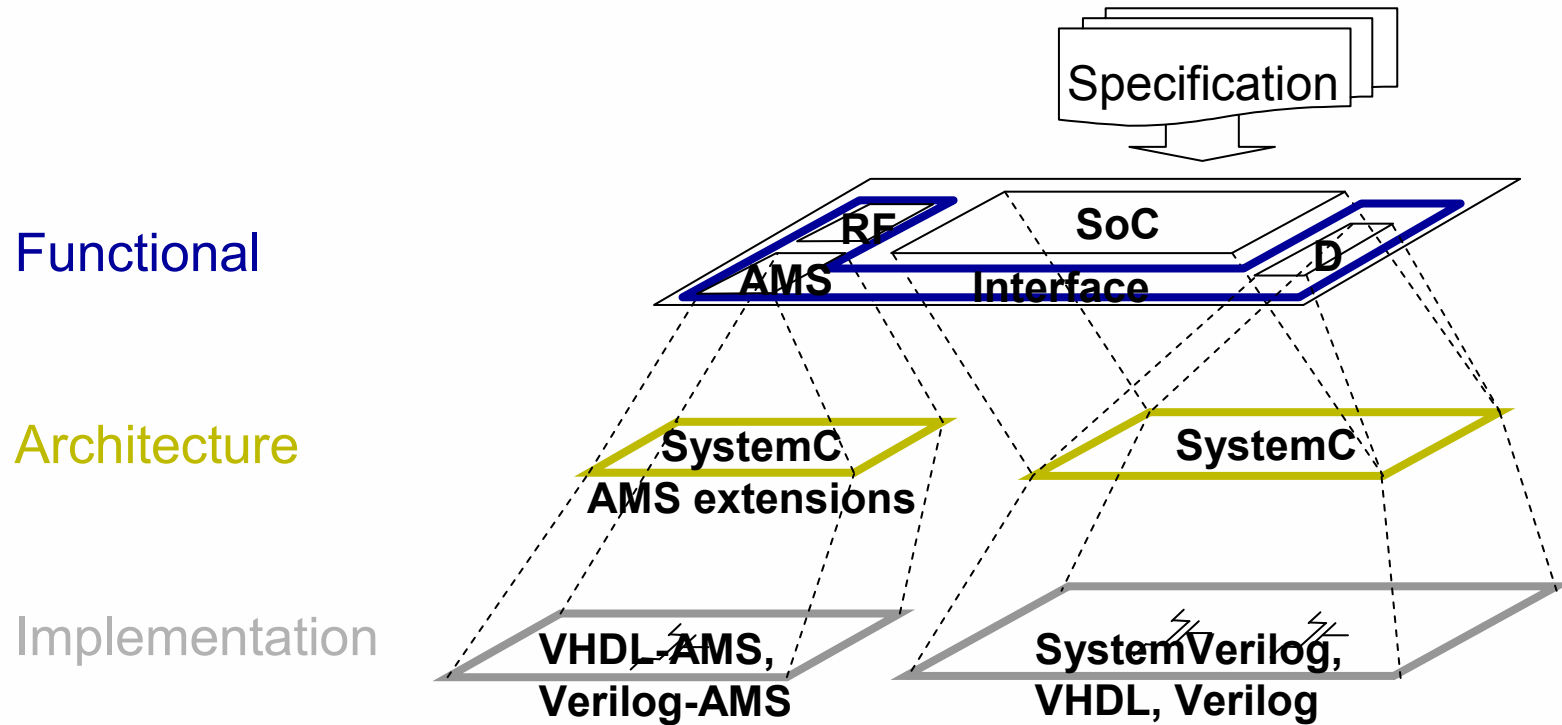
- **Phase 1: Requirements study (2006-2007)**
 - Agreement of functional requirement specification
 - Architecture and code review existing solutions
- **Phase 2: Definition and Proposal (2007-2008)**
 - Whitepaper introducing SystemC AMS Extensions
 - Draft proposal for SystemC AMS Language Reference Manual
- **Phase 3: Feedback and Standardization (2008-2009)**
 - Public review of SystemC AMS Language Reference Manual
 - Promote SystemC AMS Language Reference Manual as OSCI standard
- **AMS WG status and drafts will be announced via www.systemc.org**



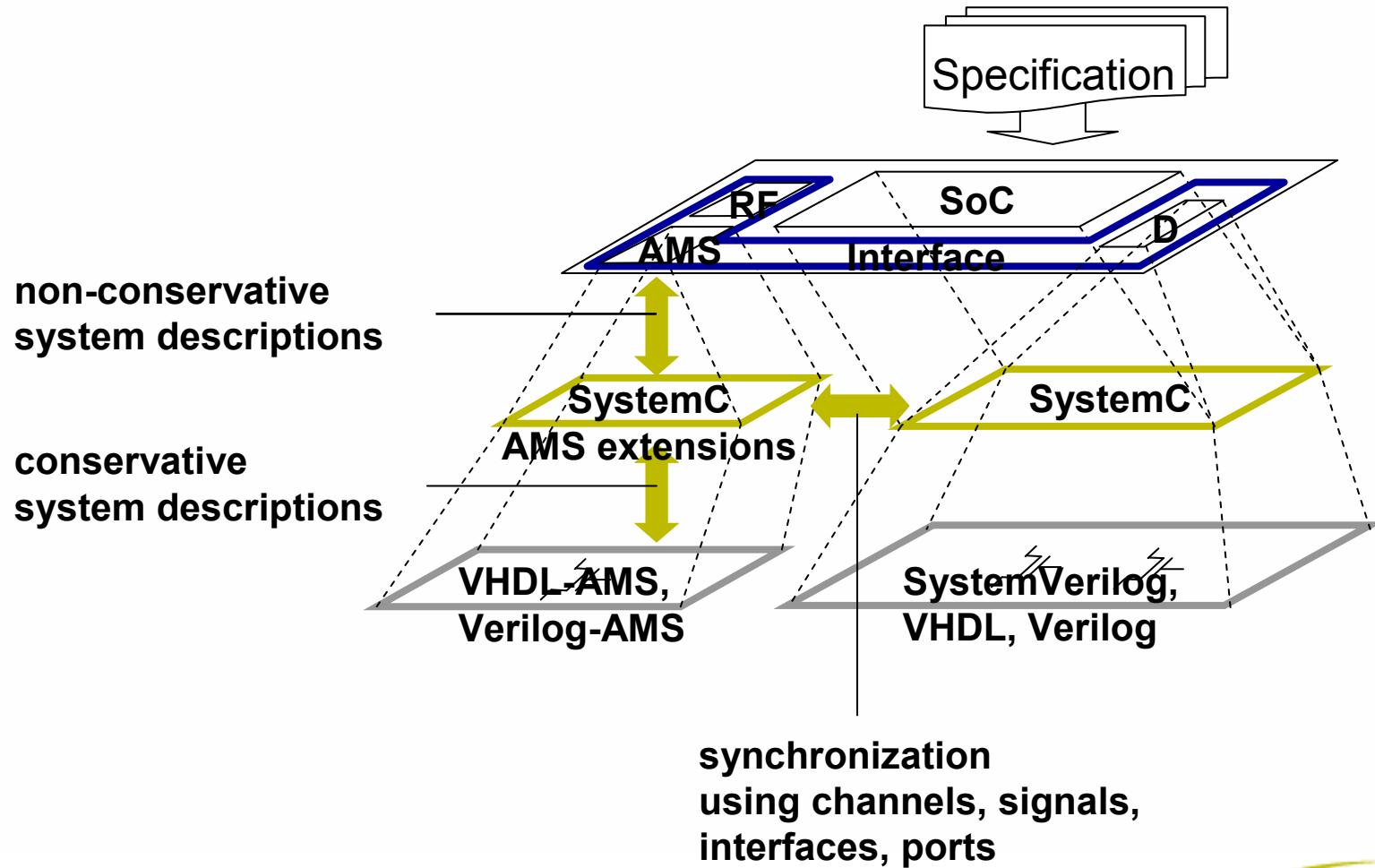
Positioning SystemC AMS Extensions



Positioning SystemC AMS Extensions



Positioning SystemC AMS Extensions



The SystemC AMS extensions

■ Objectives

- Unified and standardized modeling approach to design Embedded AMS systems
- AMS model descriptions supporting a design refinement methodology, from functional specification to implementation
- AMS constructs and semantics in a SystemC compatible class library implemented in C++
- Providing an interoperable modeling platform for development and exchange of AMS intellectual property
- Creating a robust foundation for development of system-level tools

Intermezzo:

Quiz_{Zz}

AMS system modeling – What is it?

True or False?

- ***“AMS system modeling is about including transistor-level implementation and solving Kirchhoff's laws all the time, which significantly slow-down system simulation...”***

False...

- ***“AMS system modeling is about including transistor-level implementation and solving Kirchhoff's laws all the time, which significantly slow-down system simulation...”***

...because...

- **we can abstract conservative behaviors into non-conservative ones**
- **this means we will abstract voltages and currents into directed real-value signals**
- **so, we use the *signal flow* modeling formalism for efficient simulations**

True or False?

- ***“AMS system modeling is about analysis of continuous-time waveforms, using very small time steps which significantly slow-down system simulation...”***

False...

- ***“AMS system modeling is about analysis of continuous-time waveforms, using very small time steps which significantly slow-down system simulation...”***

...because...

- **we can abstract continuous-time signals into discrete-time signals**
 - assuming we can use a sampling frequency \gg eigen-frequency of the design
- **we use an algorithmic or procedural description instead**
- **we can schedule and process the samples using a *data flow* modeling formalism for efficient simulations**

True or false?

- *“AMS system modeling is about solving complex non-linear differential equations which take ages to converge...”*

False...

- ***“AMS system modeling is about solving complex non-linear differential equations which take ages to converge...”***

...because...

- **we can abstract non-linear behavior into linear behavior for a particular operating point**
- **we use a limited number of *electrical linear network primitives***
- **with this, we can simplify the equation system, which can be calculated efficiently**

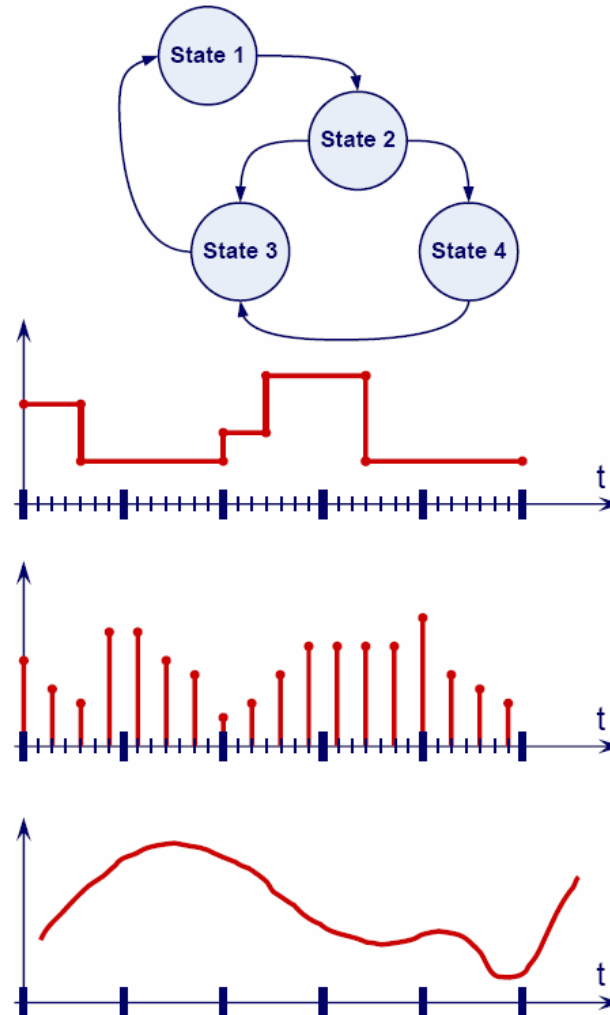
Intermezzo:

Quiz_{Zz}

AMS system modeling – What is it?

Abstraction !

Time
—
Clock ticks (synchronous syst.)
Discrete (integer value $f(\text{MRT})$)
Continuous (real value)



Behavior
Causal
Synchronous
Discrete
Continuous/Signal flow
Continuous/Conservative

Data
Tokens ((un)interpreted)
Enumerated (symbols, alphabet)
Logic values
Integer values
Real values

Primitives
Processor, memory, bus, RF emitter/receiver, PLL, sensor, actuator
ALU, register, control, converter, filter, VCO
Logical gates, Op-Amp
Transistor, R, C, source

source: C. Grimm, K. Einwich, A. Vachoux

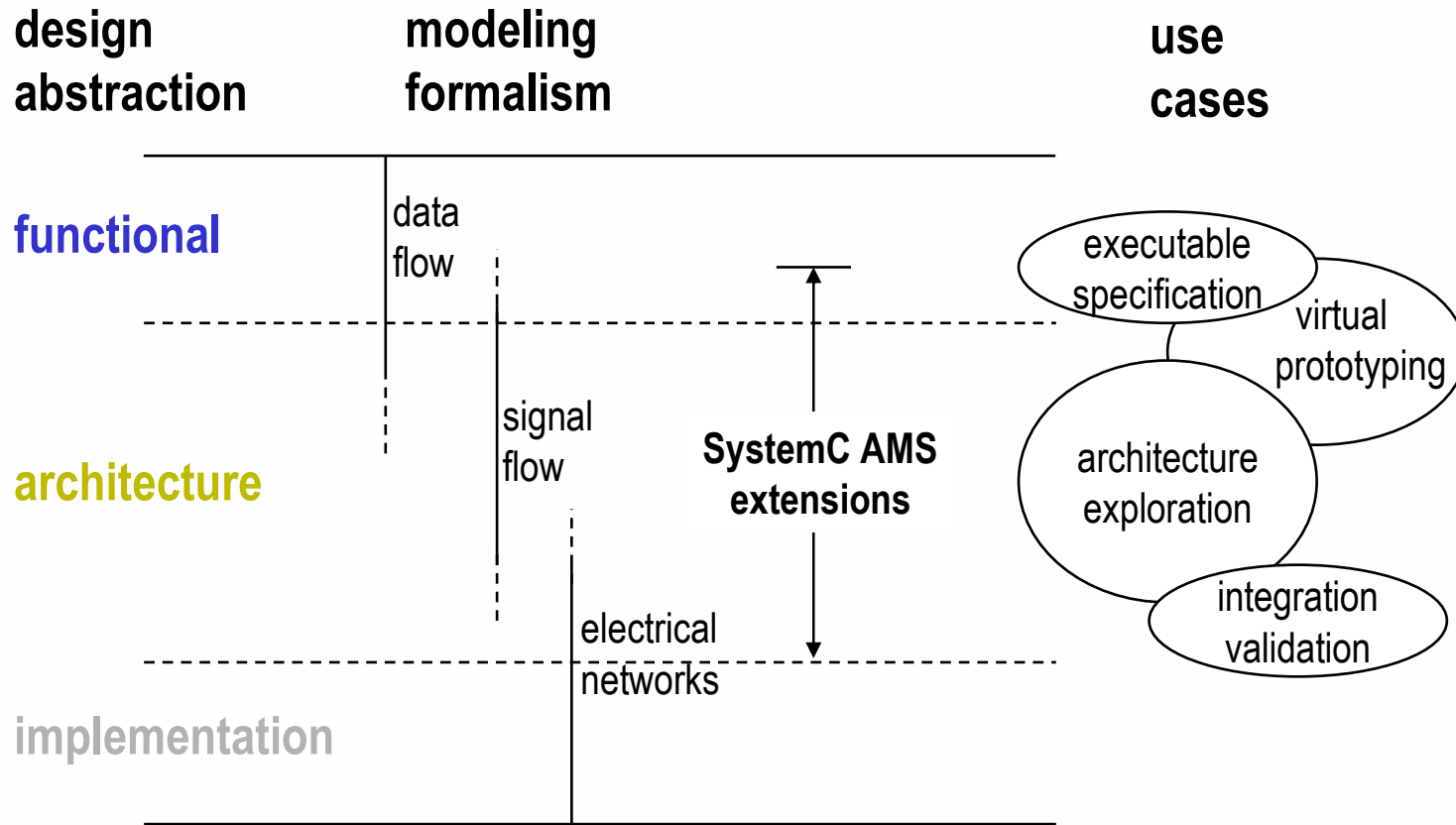
Requirements for the AMS extensions

■ Requirements

- Standardized modeling formalism and semantics for the modeling of AMS behavior at different levels of design abstraction
- Supporting multiple use cases: functional modeling, architecture exploration, integration validation and virtual prototyping
- Acceptable *simulation performance* while modeling the architecture's behavior with sufficient accuracy
- Simulation framework for the modeling AMS components and their interactions with digital HW/SW systems
- *Extensibility* of the framework to integrate 3rd party simulators, solvers and/or tools

■ Support of multiple *models of computation*

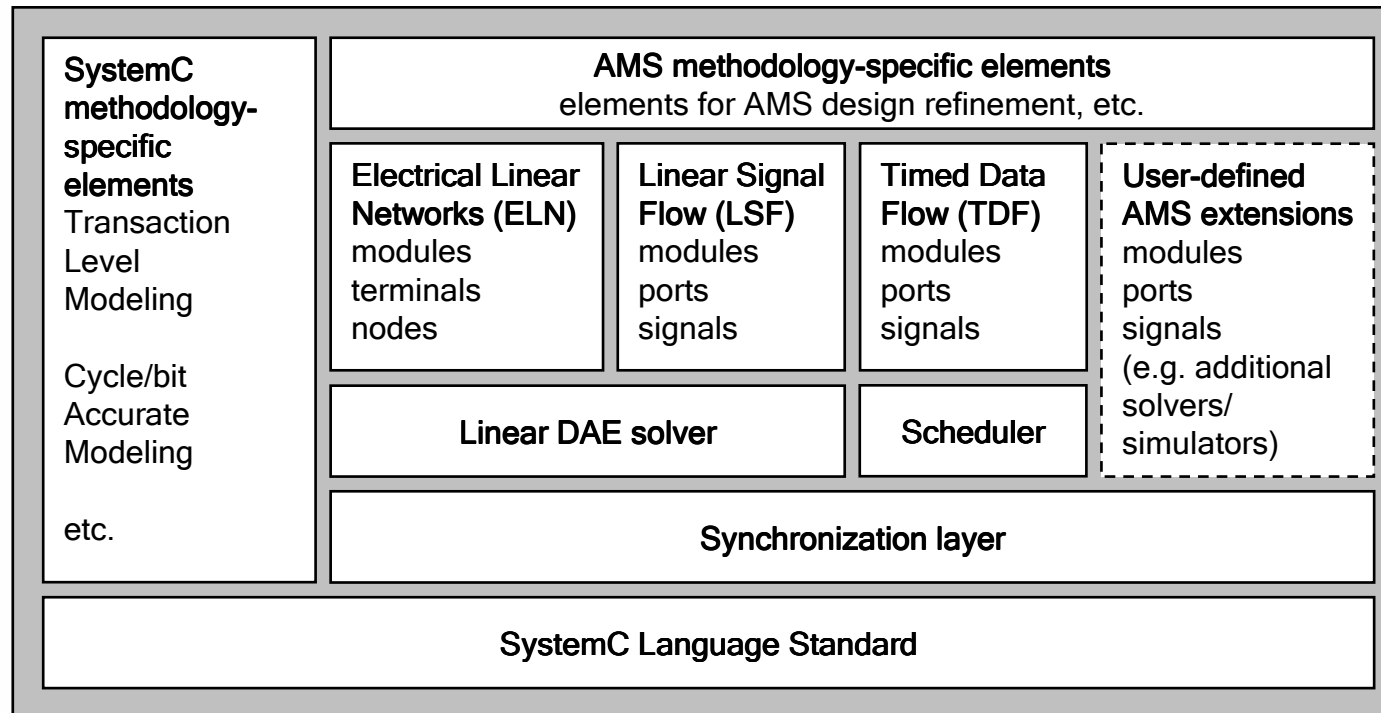
Modeling formalisms and use cases



SystemC AMS extensions – what it is?

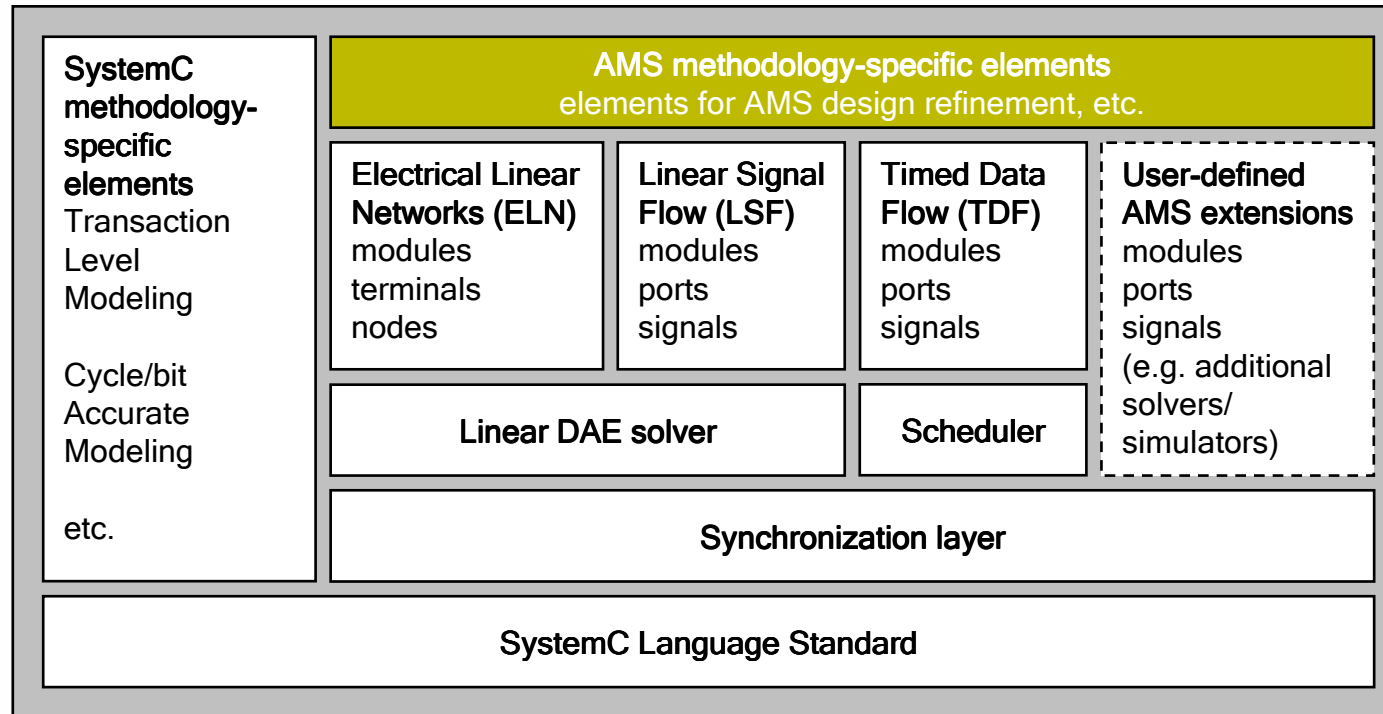
- Modeling of discrete-time and continuous-time systems
- Introduce a *design refinement methodology*, having different levels of design abstraction, to support multiple use cases
- Features
 - Timed Data Flow (TDF) for efficient simulation of discrete-time behavior (including static non-linear behavior)
 - Linear Signal Flow (LSF) and Electrical Linear Networks (ELN) primitives for efficient simulation of continuous-time behavior and electrical networks
 - Time-domain analysis and Small-signal frequency-domain AC and noise analysis
 - Synchronization to the SystemC kernel using specific converter ports and modules
- Fully compatible with SystemC Language Reference Manual IEEE Std. 1666-2005

SystemC AMS extensions – structure



SystemC AMS methodology-specific elements

1/2



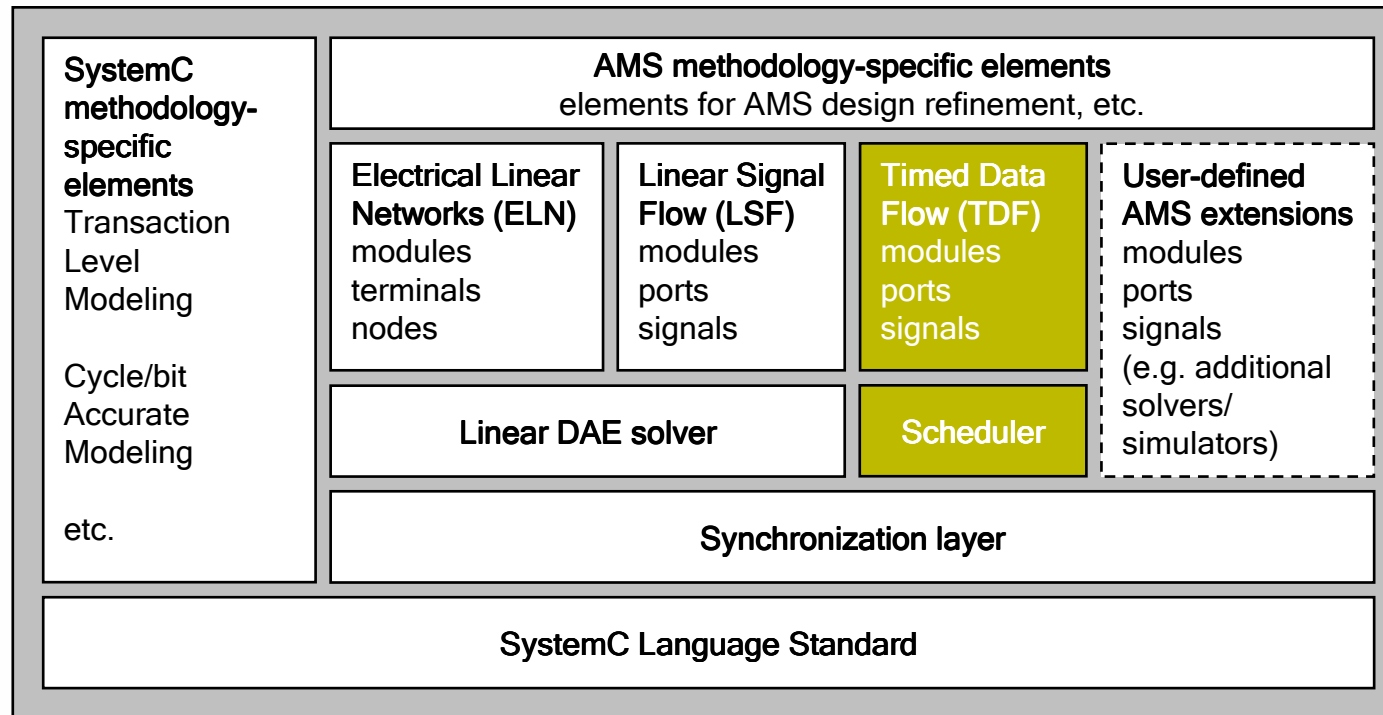
■ AMS methodology-specific elements

- Unified design refinement methodology to support different use cases
- Time domain simulation and Small-signal frequency-domain AC and noise analysis

SystemC AMS methodology-specific elements 2/2

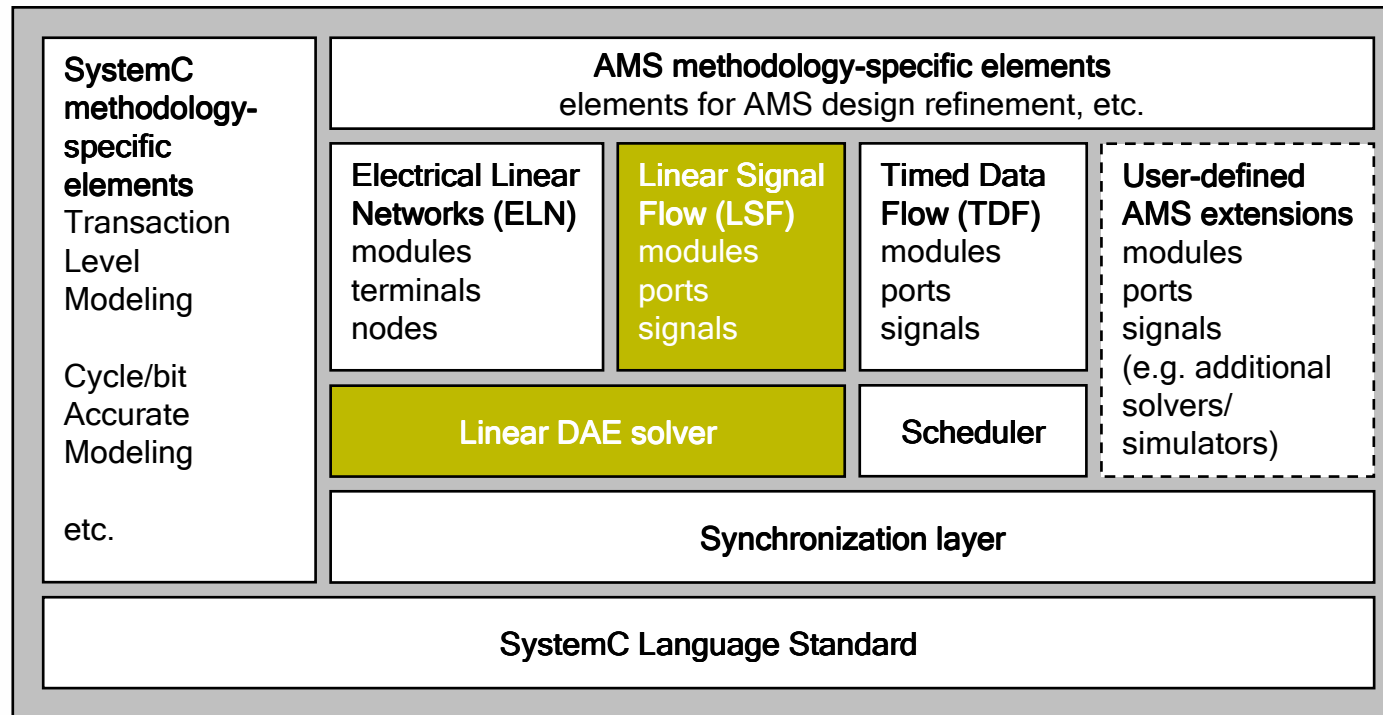
- **New modeling and design refinement methodology**
 - Different ways to write and partition models: mix and match abstraction levels with models of computation
- **Using namespaces**
 - Clearly identify the used model of computation
 - Unified and common set of predefined classes, (converter) ports and signals
- **Examples**
 - Module `sca_tdf::sca_module` `sca_lsf::sca_module`
 - Input port `sca_tdf::sca_in` `sca_lsf::sca_in`
 - Signals `sca_tdf::sca_signal` `sca_lsf::sca_signal`

Timed Data Flow (TDF)



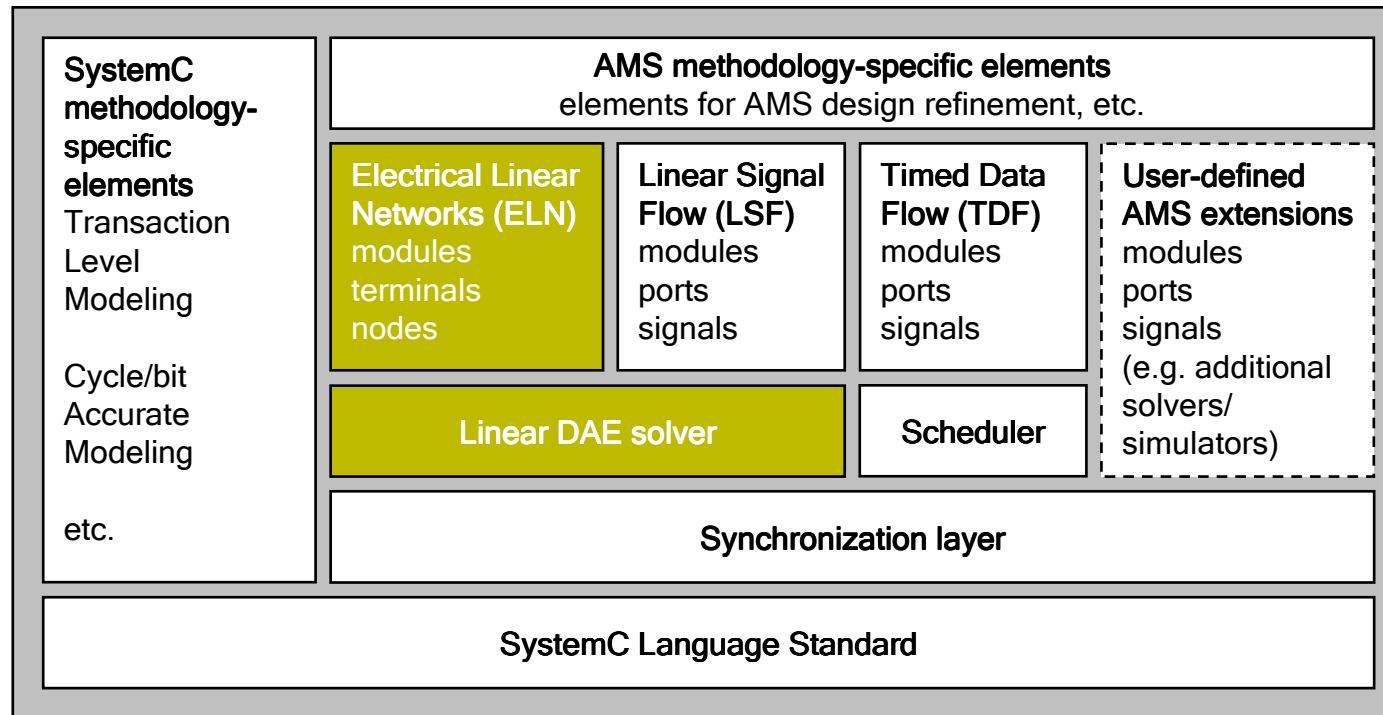
- **Timed Data Flow - efficient simulation of discrete-time behavior**
 - Data flow simulation accelerated using static scheduling
 - Schedule is activated in discrete time steps, introducing timed semantics
 - Support of static non-linear behavior

Linear Signal Flow (LSF)



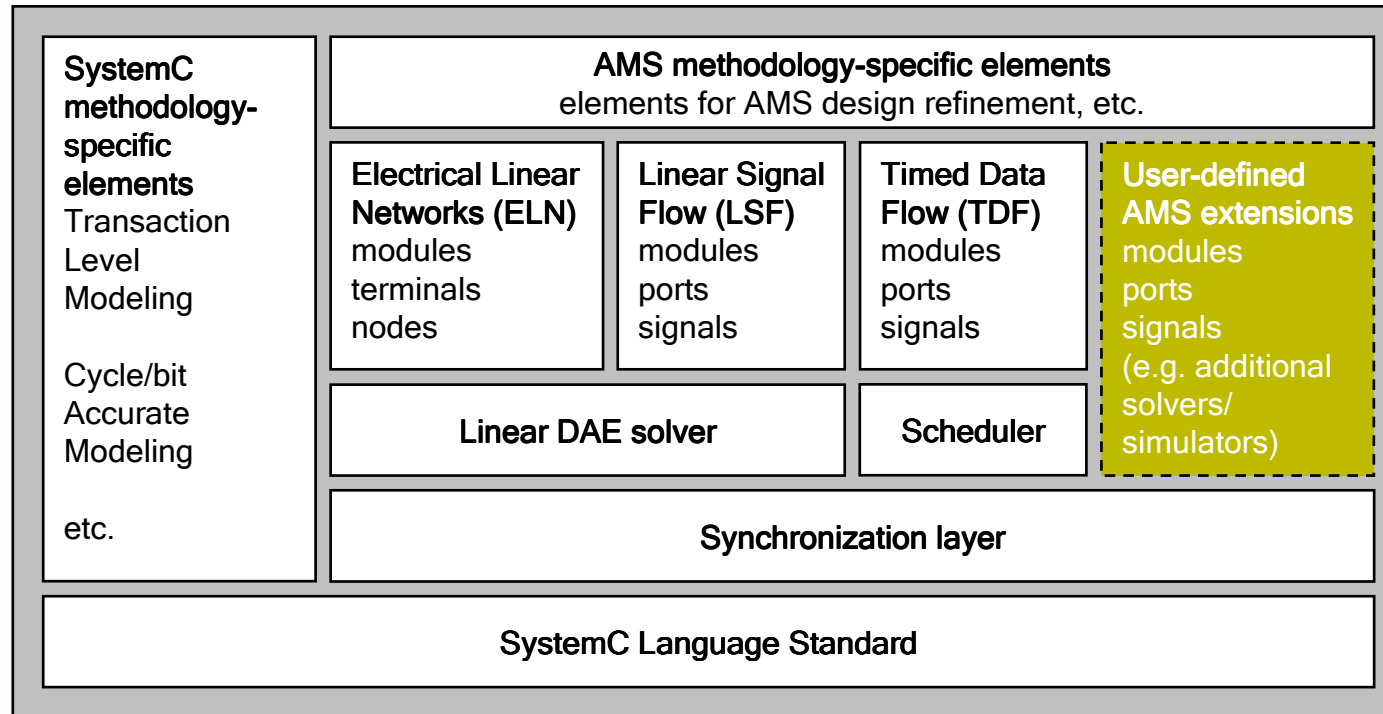
- **Linear Signal Flow - simulation of continuous-time behavior**
 - Differential and Algebraic Equations solved numerically at appropriate time steps
 - Primitive modules defined for adders, integrators, differentiators, transfer functions, etc.

Electrical Linear Networks (ELN)



- **Electrical Linear Networks - simulation of network primitives**
 - Network topology results in equation system which is solved numerically
 - Primitive modules defined for linear components (e.g. resistors, capacitors) and switches

User-defined AMS extensions

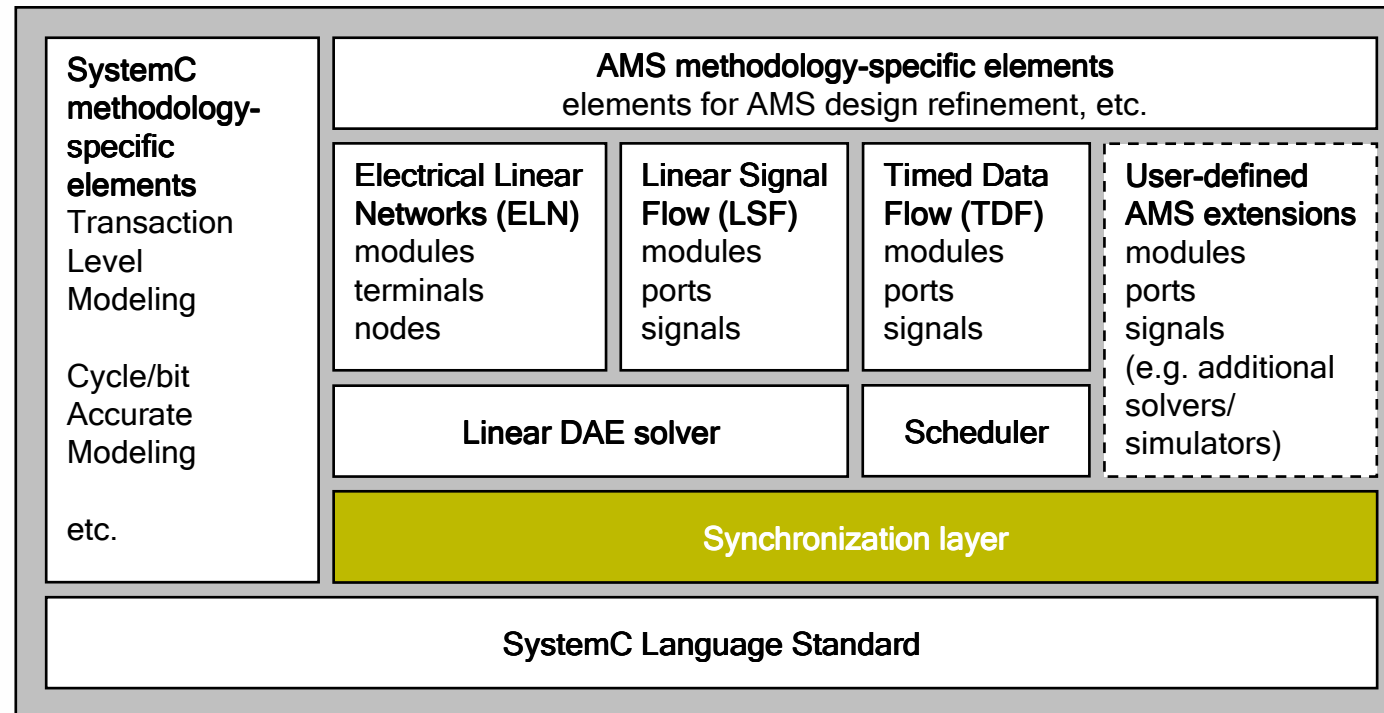


- **User-defined AMS extensions**

- Additional simulators and solvers linked in a C++ manner (e.g. shared object)
- Or using the synchronization layer defining the communication with SystemC

Synchronization with SystemC

1/2



■ Synchronization with SystemC

- Predefined converter ports and converter modules/primitives
- To synchronize between TDF, LSF and/or ELN and SystemC

- **Predefined converter ports in TDF**
 - To establish a connection to a SystemC channel of class `sc_core::sc_signal<T>`
 - Reading or writing values during the first delta cycle of the current SystemC time step
 - Predefined ports: `sca_tdf::sc_in`, `sca_tdf::sc_out`
- **Predefined primitive modules defined in LSF and ELN**
 - Converter modules defined as sources and sinks
 - To read or write values to SystemC ports of class `sc_core::sc_in<double>` or `sc_core::sc_out<double>`
 - To read or write values to TDF ports of class `sca_tdf::sca_in<double>` or `sca_tdf::sca_out<double>`
 - Example of predefined converter primitive module (source): `sca_lsf::sca_sc_source`, `sca_eln::sca_sc_vsource`

Code example – mixer function in TDF

```
SCA_TDF_MODULE(mixer) // TDF primitive module definition
{
  sca_tdf::sca_in<double> rf_in, lo_in; // TDF in ports
  sca_tdf::sca_out<double> if_out;      // TDF out ports
}
```

**TDF primitive
module:
no hierarchy**

```
void set_attributes()
{
  set_timestep(1.0, SC_US); // time between activations
}
```

**Attributes specify
timed semantics**

```
void processing()
{
  if_out.write( rf_in.read() * lo_in.read() );
}
```

**processing()
function executed
at each activation**

```
SCA_CTOR(mixer) {}
};
```

**AMS
constructor**

note: AMS language constructs currently under discussion – subject to change

Code example – Lowpass filter in ELN 1/2

```
SC_MODULE(lp_filter_eln)
{
  sca_tdf::sca_in<double> in;
  sca_tdf::sca_out<double> out;
```

SC_MODULE used
for hierarchical
structure

```
sca_eln::sca_node in_node, out_node; // node declarations
sca_eln::sca_node_ref gnd;           // reference node
```

**nodes to connect
components**

```
sca_eln::sca_r *r1; // resistor
sca_eln::sca_c *c1; // capacitor
```

**network primitives
(components)**

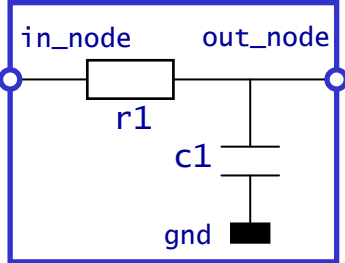
```
sca_eln::sca_tdf_vsource *v_in;
sca_eln::sca_tdf_vsink *v_out;
```

**primitive
converter
modules from/to
TDF**

⋮

note: AMS language constructs currently under discussion – subject to change

Code example – Lowpass filter in ELN 2/2

<pre> : : SC_CTOR(lp_filter_eln)</pre>	normal constructor
<pre>{ v_in = new sca_eln::sca_tdf_vsource("v_in", 1.0); v_in->inp(in); v_in->p(in_node); v_in->n(gnd);</pre>	TDF input is converted to voltage
<pre> r1 = new sca_eln::sca_r("r1", 10e3); // 10kOhm resistor r1->p(in_node); r1->n(out_node); c1 = new sca_eln::sca_c("c1", 100e-6); // 100uF capacitor c1->p(out_node); c1->n(gnd);</pre>	
<pre> v_out = new sca_eln::sca_tdf_vsink("v_out", 1.0); v_out->p(out_node); v_out->n(gnd); v_out->outp(out); } };</pre>	output voltage converted to TDF signal

note: AMS language constructs currently under discussion – subject to change

Code example – top-level (RF front-end)

```
SC_MODULE(frontend)
{
  sca_tdf::sca_in<double> rf, loc_osc;
  sca_tdf::sca_out<double> if_out;
  sc_core::sc_in<sc_dt::sc_bv<3> > ctrl_config;
```

SC_MODULE used for hierarchical structure

```
sca_tdf::sca_signal<double> if_sig;
sc_core::sc_signal<double> ctrl_gain;
```

usage of different signals

```
mixer* mixer1;
lp_filter_eln* lpf1;
agc_ctrl* ctrl1;
```

```
SC_CTOR(frontend) {
```

```
  mixer1 = new mixer("mixer1"); // TDF module
  mixer1->rf_in(rf);
  mixer1->lo_in(loc_osc);
  mixer1->if_out(if_sig);
```

High-level mixer model (TDF module)

```
  lpf1 = new lp_filter_eln("lpf1"); // ELN module
  lpf1->in(if_sig);
  lpf1->out(if_out);
```

LPF close to implementation level (ELN module)

```
  ctrl1 = new agc_ctrl("ctrl1"); // SystemC module
  ctrl1->out(ctrl_gain);
  ctrl1->config(ctrl_config);
```

easy to combine with normal SystemC modules !

```
  }
};
```

note: AMS language constructs currently under discussion – subject to change

Conclusions

- **The SystemC AMS extensions enhance the available SystemC standard with support for Linear Electrical Networks, Linear Signal flow, and Timed Data Flow modeling**
 - Essential features to model telecommunication, automotive and imaging sensor applications
- **New language constructs support the creation of AMS models at higher levels of abstraction**
 - Building a foundation for new AMS design methodologies based on SystemC
 - Essential for executable specification, architecture exploration, integration validation and virtual prototyping use cases
- **All features currently being defined in the AMS Language Reference Manual**
 - OSCI and the AMS Working Group are committed in defining a unified and standardized system-level design language in the AMS domain

Acknowledgements

- **Members of the OSCI AMS Working Group for their contribution to the AMS extensions**
- **Special thanks to**
 - Karsten Einwich
 - Christoph Grimm
 - Alain Vachoux

**For their continued support and major contribution to the
Whitepaper and AMS Language Reference Manual**

What's next...

- **Whitepaper introducing the SystemC AMS Extensions will be announced at DAC!**
 - Announcement and whitepaper will become available on www.systemc.org
- **AMS discussion forum on www.systemc.org**
 - Start interacting with SystemC community to discuss the concepts as defined in the SystemC AMS Extension
 - Sign-up to this email-reflector soon!
- **Public Draft of the SystemC AMS Language Reference Manual**
 - Targeted for 2nd half of 2008
- **SystemC AMS-TLM interaction...**



Thank You

Martin Barnasconi, AMS WG Chairman

June 9, 2008