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
- 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

<table>
<thead>
<tr>
<th>End Product Markets</th>
<th>2003</th>
<th>2004</th>
<th>2005</th>
<th>2006</th>
<th>2007</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microprocessor/DSP</td>
<td>18.9%</td>
<td>16.0%</td>
<td>13.1%</td>
<td>10.5%</td>
<td>14.7%</td>
</tr>
<tr>
<td>Computer, Peripheral</td>
<td>22.9%</td>
<td>21.6%</td>
<td>18.5%</td>
<td>24.2%</td>
<td>19.0%</td>
</tr>
<tr>
<td>Wired Network</td>
<td>11.2%</td>
<td>5.2%</td>
<td>5.8%</td>
<td>4.8%</td>
<td>5.2%</td>
</tr>
<tr>
<td>Wireless Network</td>
<td>13.1%</td>
<td>10.4%</td>
<td>13.1%</td>
<td>7.3%</td>
<td>6.9%</td>
</tr>
<tr>
<td>Multimedia</td>
<td>25.6%</td>
<td>34.2%</td>
<td>33.8%</td>
<td>37.9%</td>
<td>31.9%</td>
</tr>
<tr>
<td>Automotive</td>
<td>1.9%</td>
<td>3.0%</td>
<td>3.9%</td>
<td>4.0%</td>
<td>4.3%</td>
</tr>
<tr>
<td>Others</td>
<td>6.4%</td>
<td>9.7%</td>
<td>11.9%</td>
<td>11.3%</td>
<td>16.1%</td>
</tr>
</tbody>
</table>

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

- **Phase 3: Feedback and Standardization** (2008-2009)
  - Promote SystemC AMS Language Reference Manual as OSCI standard

- AMS WG status and drafts will be announced via [www.systemc.org](http://www.systemc.org)
Positioning SystemC AMS Extensions

Functional

Architecture

Implementation

missing abstraction

Specification

RF

SoC

Interface

SystemC

VHDL-AMS, Verilog-AMS

SystemVerilog, VHDL, Verilog
Positioning SystemC AMS Extensions

Functional

Architecture

Implementation

SystemC AMS extensions

VHDL-AMS, Verilog-AMS

SystemC

SystemVerilog, VHDL, Verilog

Specification
Positioning SystemC AMS Extensions

non-conservative
system descriptions

conservative
system descriptions

VHDL-AMS,
Verilog-AMS

SystemC
AMS extensions

SystemC
AMS extensions

SystemVerilog,
VHDL, Verilog

synchronization
using channels, signals,
interfaces, ports
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:

Quizzzzz

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 >> 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:

Quizzzz

AMS system modeling – What is it?
Abstraction!

<table>
<thead>
<tr>
<th>Time</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock ticks</td>
<td>Causal</td>
</tr>
<tr>
<td>(synchronous syst.)</td>
<td>Synchronous</td>
</tr>
<tr>
<td>Discrete (integer value f(MRT))</td>
<td>Discrete</td>
</tr>
<tr>
<td>Continuous (real value)</td>
<td>Continuous/Signal flow</td>
</tr>
<tr>
<td></td>
<td>Continuous/Conservative</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data</th>
<th>Primitives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tokens ((un)interpreted)</td>
<td>Processor, memory, bus,</td>
</tr>
<tr>
<td></td>
<td>RF emitter/receiver, PLL,</td>
</tr>
<tr>
<td></td>
<td>sensor, actuator</td>
</tr>
<tr>
<td>Enumerated (symbols, alphabet)</td>
<td>ALU, register, control,</td>
</tr>
<tr>
<td></td>
<td>converter, filter, VCO</td>
</tr>
<tr>
<td>Logic values</td>
<td>Logical gates, Op-Amp</td>
</tr>
<tr>
<td>Integer values</td>
<td>Transistor, R, C, source</td>
</tr>
<tr>
<td>Real values</td>
<td></td>
</tr>
</tbody>
</table>

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

**design abstraction**  
**modeling formalism**  
**use cases**

- functional
  - data flow

- architecture
  - signal flow
  - SystemC AMS extensions
  - electrical networks

- implementation

use cases:
- executable specification
- virtual prototyping
- architecture exploration
- integration validation
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**

<table>
<thead>
<tr>
<th>SystemC methodology-specific elements</th>
<th>AMS methodology-specific elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction Level Modeling</td>
<td>elements for AMS design refinement, etc.</td>
</tr>
<tr>
<td>Cycle/bit Accurate Modeling</td>
<td>Electrical Linear Networks (ELN) modules terminals nodes</td>
</tr>
<tr>
<td>etc.</td>
<td>Linear Signal Flow (LSF) modules ports signals</td>
</tr>
<tr>
<td></td>
<td>Timed Data Flow (TDF) modules ports signals</td>
</tr>
<tr>
<td></td>
<td>User-defined AMS extensions modules ports signals (e.g. additional solvers/simulators)</td>
</tr>
</tbody>
</table>

**Synchronization layer**

**SystemC Language Standard**
### SystemC AMS methodology-specific elements

- **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 2/2 elements

- 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)

<table>
<thead>
<tr>
<th>SystemC methodology-specific elements</th>
<th>AMS methodology-specific elements elements for AMS design refinement, etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction Level Modeling</td>
<td></td>
</tr>
<tr>
<td>Cycle/bit Accurate Modeling</td>
<td></td>
</tr>
<tr>
<td>etc.</td>
<td></td>
</tr>
<tr>
<td>Electrical Linear Networks (ELN) modules terminals nodes</td>
<td>Linear Signal Flow (LSF) modules ports signals</td>
</tr>
<tr>
<td>Timed Data Flow (TDF) modules ports signals</td>
<td>User-defined AMS extensions modules ports signals (e.g. additional solvers/simulators)</td>
</tr>
<tr>
<td>Linear DAE solver</td>
<td>Scheduler</td>
</tr>
<tr>
<td>Synchronization layer</td>
<td></td>
</tr>
<tr>
<td>SystemC Language Standard</td>
<td></td>
</tr>
</tbody>
</table>

- **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

- 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

- Predefined converter ports and converter modules/primitives
- To synchronize between TDF, LSF and/or ELN and SystemC
Synchronization with 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_vsourc`
# 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
}

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

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

SCACTOR(mixer) {}
```

<table>
<thead>
<tr>
<th>TDF primitive model: no hierarchy</th>
<th>Attributes specify timed semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>processing() function executed at each activation</td>
<td>AMS constructor</td>
</tr>
</tbody>
</table>

Note: AMS language constructs currently under discussion – subject to change
Code example – Lowpass filter in ELN 1/2

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

SC_MODULE used for hierarchical structure

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

nodes to connect components

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

network primitives (components)

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

primitive converter modules from/to TDF

...
# Code example – Lowpass filter in ELN 2/2

```c
SCCTOR(lp_filter_eln)
{
  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);

  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);

  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);
}
```

**note:** AMS language constructs currently under discussion – subject to change
Code example – top-level (RF front-end)

```cpp
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>> > ctrl1_config;

    sca_tdf::sca_signal<double> if_sig;
    sc_core::sc_signal<double> ctrl1_gain;

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

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

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

        ctrl1 = new agc_ctrl("ctrl1"); // SystemC module
        ctrl1->out(ctrl1_gain);
        ctrl1->config(ctrl1_config);
    }
};
```

- **SC_MODULE** used for hierarchical structure
- **Usage of different signals**
- **High-level mixer model** (TDF module)
- **LPF close to implementation level (ELN module)**
- **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
  - OSCl 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!

  - Targeted for 2nd half of 2008

- SystemC AMS-TLM interaction...
Thank You

Martin Barnasconi, AMS WG Chairman
June 9, 2008