Sneak Preview of the Upcoming SystemC AMS 2.0 Standard

Martin Barnasconi, AMSWG chairman,
18th NASCUG, DAC 2012
Analog Mixed Signal applications

Communications

Imaging

Automotive
Outline

- Introduction: What is SystemC AMS?
- Timed Data Flow (TDF) modeling style
- Use cases and requirements for Dynamic TDF
- Example: DC motor control
- SystemC AMS 2.0 Dynamic TDF features
- Code example
- Summary and outlook
What is SystemC AMS?

Functional Architecture Implementation

Analog/Mixed-Signal Digital HW/SW

Specification

SystemC

CPU Interface

AMS

VHDL, Verilog

VHDL-AMS, Verilog-AMS

SystemC

AMS extensions

2.0draft

RF

Dig
SystemC AMS – History

- **1999**: Open SystemC Initiative (OSCI) announced
- **2000**: SystemC 1.0 released (sourceforge.net)
- **2002**: OSCI SystemC 1.0.2
- **2005**: IEEE Std 1666-2005 LRM
- **2005**: SystemC Transaction level modeling (TLM) 1.0 released
- **2007**: SystemC 2.2 released
- **2009**: SystemC TLM 2.0 standard
- **2009**: SystemC Synthesizable Subset Draft 1.3
- **2011**: IEEE Std 1666-2011 LRM
- **2012**: SystemC 2.3
- **~2000**: First C-based AMS initiatives (AVSL, MixSigC)
- **2002**: SystemC-AMS study group started
- **2005**: First SystemC-AMS PoC released by Fraunhofer
- **2006**: OSCI AMSWG installed
- **2008**: SystemC AMS Draft 1 LRM
- **2010**: SystemC AMS 1.0 standard
- **2010**: SystemC AMS 1.0 PoC released by Fraunhofer
- **2012**: SystemC AMS 2.0 draft standard
- **today**: SystemC AMS 2.0
Model abstraction and formalisms

**Use cases**
- Executable specification
- Virtual prototyping
- Architecture exploration
- Integration validation

**Model abstractions**
- Discrete-time
  - static non-linear
  - Non-conservative behavior
- Continuous-time
  - dynamic linear
  - Conservative behavior

**Modeling formalism**
- Timed Data Flow (TDF)
- Linear Signal Flow (LSF)
- Electrical Linear Networks (ELN)
SystemC AMS language elements

SystemC Language Standard (IEEE Std 1666-2011)

Mixed Signal Virtual Prototypes
written by the end user

SystemC methodology-specific elements
Transaction-level modeling (TLM), Cycle/Bi-bit-accurate modeling, etc.

AMS methodology-specific elements
elements for AMS design refinement, etc.

Scheduler

Linear DAE solver

Synchronization layer

Timed Data Flow (TDF)
modules
ports
signals

Linear Signal Flow (LSF)
modules
ports
signals

Electrical Linear Networks (ELN)
modules
terminals
nodes

Mixed Signal Virtual Prototypes
written by the end user

SystemC Language Standard (IEEE Std 1666-2011)
Timed Data Flow (TDF) modeling style

- **TDF is based on synchronous dataflow**
  - A module is executed if enough samples are available at its input ports
  - The number of read/written samples are constant for each module activation
  - This enables *static* scheduling, which follows the signal flow direction

- **TDF characteristics**
  - The samples of a TDF signal are “tagged” in time using a time step attribute defined prior to simulation
  - TDF samples have a constant and fixed time step during simulation (AMS 1.0 only)

- **TDF is a very efficient discrete-time modeling style**
Use cases and requirements (1)

- **Abstract modelling of sporadically changing signals**
  - E.g. power management that switches on/off AMS subsystems

- **Abstract description of reactive behaviour**
  - AMS computations driven by events or transactions

- **Capture behaviour where frequencies (and time steps) change dynamically**
  - Often the case for clock recovery circuits or capturing jitter

- **Modelling systems with varying (data) rates**
  - E.g. multi-standard / software-defined radio (SDR) systems

This requires a dynamic and reactive Timed Data Flow modeling style

- Basically introduce variable time step instead of fixed/constant time step
## Use cases and requirements (2)

<table>
<thead>
<tr>
<th>Use cases</th>
<th>Requirements</th>
<th>Application examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstraction of sporadically changing signals</td>
<td>Switch on/off AMS computations</td>
<td>Power management unit</td>
</tr>
<tr>
<td>Abstract description of reactive behavior</td>
<td>Detect analog zero- or threshold crossing</td>
<td>Sensor circuits; alarm mode of systems</td>
</tr>
<tr>
<td>Capture behavior where frequencies (and time steps) change dynamically</td>
<td>Request and response caused by digital event or transaction</td>
<td>AMS embedded in digital HW/SW virtual prototype</td>
</tr>
<tr>
<td>Modeling systems with varying (data) rates</td>
<td>Changeable time step of AMS computations</td>
<td>VCO, PLL, PWM, Clock recovery circuits</td>
</tr>
<tr>
<td></td>
<td>Changeable time step and/or data rate</td>
<td>Communication systems, multi-standard radio interfaces (e.g. cognitive radios)</td>
</tr>
</tbody>
</table>
Example: DC motor control

- Functional model in the Laplace domain modelled in SystemC AMS
- To achieve high accuracy, many module activations are necessary when using fixed time steps (AMS 1.0)
- Introducing *Dynamic TDF* to only compute when necessary, due to dynamic time step mechanism (AMS 2.0)
DC motor control loop behavior

\[ i_{\text{meas}}(t) \]

\[ i_{\text{ref}} \]

\[ v_{\text{drv}}(t) \]

\[ t_{\text{ramp}} \]

\[ t_{\text{duty}} \]

\[ t_{\text{period}} \]
New AMS 2.0 Dynamic TDF features

New callback and member functions to support Dynamic TDF:

- **change_attributes()**
  - callback provides a context, in which the time step, rate, or delay attributes of a TDF cluster may be changed

- **request_next_activation(...)**
  - member function to request a next cluster activation at a given time step, event, or event-list

- **does_attribute_changes(), does_no_attribute_changes()**
  - member functions to mark a TDF module to allow or disallow making attribute changes itself, respectively

- **accept_attribute_changes(), reject_attribute_changes()**
  - member functions to mark a TDF module to accept or reject attribute changes caused by other TDF modules, respectively
Example of Pulse Width Modulator (1)

```c++
// pwm_dynamic.h
#include <cmath>
#include <systemc-ams>

SCA_TDF_MODULE(pwm) // for dynamic TDF, we can use the same helper macro to define the module class
{
    sca_tdf::sca_in<double> in;
    sca_tdf::sca_out<double> out;

    pwm( sc_core::sc_module_name nm, ... )
    : in("in"), out("out") {}

    void set_attributes()
    {
        does_attribute_changes(); // module allowed to make changes to TDF attributes
        accept_attribute_changes(); // module allows attribute changes made by other modules
    }

    void change_attributes() // new callback to change attributes during simulation
    {
        double t = get_time().to_seconds(); // current time
        double t_pos = std::fmod( t, t_period); // time position inside pulse period
        ...
    }
```
Example of Pulse Width Modulator (2)

```cpp
if ( t_pos < t_ramp ) {
    // rising edge
    request_next_activation( t_ramp - t_pos, sc_core::SC_SEC );
} else if ( t_pos < t_ramp + t_duty ) {
    // plateau
    request_next_activation( ( t_ramp + t_duty ) - t_pos, sc_core::SC_SEC );
} else if ( t_pos < t_ramp + t_duty + t_ramp ) {
    // falling edge
    request_next_activation( ( t_ramp + t_duty + t_ramp ) - t_pos, sc_core::SC_SEC );
} else {
    // return to initial value
    request_next_activation( t_period - t_pos, sc_core::SC_SEC );
}
}

void processing()
{
    ... // PWM behavior
}

private:
    ... // member variables
};
```

Dynamic TDF features indicated in red
## TDF vs. Dynamic TDF comparison

<table>
<thead>
<tr>
<th>TDF model of computation variant</th>
<th>( t_step ) (ms)</th>
<th>( t_ramp ) (ms)</th>
<th>( t_period ) (ms)</th>
<th>Time accuracy (ms)</th>
<th>#activations per period</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional TDF</td>
<td>0.01 (fixed)</td>
<td>0.05</td>
<td>5.0</td>
<td>0.01 (( = t_step ))</td>
<td>500</td>
</tr>
<tr>
<td>Dynamic TDF</td>
<td>variable</td>
<td>0.05</td>
<td>5.0</td>
<td>defined by ( sc_set_time_resolution() )</td>
<td>4</td>
</tr>
</tbody>
</table>

- **Comparison of the two variants of the TDF model of computation**
  - Conventional PWM TDF model uses a fixed time step that triggers too many unnecessary computations.
  - When using Dynamic TDF, the PWM model is only activated if necessary.
Summary and outlook

- **SystemC AMS 2.0 draft standard available**
  - Introducing Dynamic Timed Data Flow to facilitate a more reactive and dynamic behavior for AMS computations
  - Enabling a tighter, yet efficient and time-accurate, synchronization for AMS signal processing and control systems
  - Dynamic TDF is based on the same data flow semantics as TDF, thus maintaining the same model abstraction and speed

- **New language constructs and semantics defined in the SystemC AMS 2.0 draft language reference manual**
  - Available under SystemC Open Source license at [www.accellera.org](http://www.accellera.org)

- **Public review has been conducted. The AMS Working group is now preparing the final release**
  - SystemC AMS 2.0 release expected in 2H/2012
More information

- www.accellera.org/downloads/standards/systemc/ams
- www.accellera.org/activities/committees/systemc-ams/ams_2_draft_public_review/
- www.accellera.org/community/articles/amsdynamicctdf
- www.systemc-ams.org
Thank you