Modeling Communication Systems Using the SystemC AMS Building Block Library

Jiong Ou\textsuperscript{1}, Farooq Muhammad\textsuperscript{1}, Christoph Grimm\textsuperscript{1} and Martin Barnasconi\textsuperscript{2}

Vienna University of Technology\textsuperscript{1}

NXP Semiconductors\textsuperscript{2}

13.06.2010
Outline

- A brief introduction to the SystemC AMS extensions
- Overview of the AMS Building Block Library
- Application example: Modeling of OFDM Transceiver System
- Conclusions and Future work
SystemC AMS extensions

1. SystemC AMS is not SPICE

2. SystemC AMS is not for circuit design – it’s for overall system modeling!

3. Used in appropriate way, SystemC AMS yields
   • high simulation performance
   • increased design productivity
SystemC AMS extensions

Functional

Architecture

Implementation

NEW

Specification

SoC Interface

SystemC

AMS extensions

RF

VHDL-AMS, Verilog-AMS

SystemVerilog, VHDL, Verilog
AMS building block library - motivation

- **Problems:**
  - *Slow simulation* of AMS communication systems
  - Modeling of different parts of a system needs **a serious investment in time**
  - *Limitations and constraints* of closed (proprietary) models not always clear

- **Possible Solutions:**
  - Modeling in **Timed Data Flow** (TDF)
  - Provides **building blocks** for various AMS, RF, and digital functions
  - Using **open model-based design** approach
Available modules

- **Signal sources:**
  - Sine/Cosine, bit stream (uniformly), random number (Gaussian) ...

- **Signal processing:**
  - Basic mathematic modules: adder, multiplier, integrator ...
  - Analog modules: LNA, mixer, PLL, Butterworth/Chebyshev filter ...
  - Modulation processes: AM, BASK, M-FSK, M-PSK, DBPSK, OQPSK, QAM, OFDM ...
  - DSP algorithm: FFT/IFFT ...
  - Converter: A/D converter, D/A converter, P2S, S2P ...

- **Analysis tools:** eye diagram, scatterplot ...
Non-ideal properties of analog

- **LNA:**
  - intermodulation products (IP3)
  - output limitation

- **Mixer:**
  - intermodulation products (IP3)
  - output limitation

- **AD/DA converter:**
  - gain error
  - offset error
  - output limitation

- **General modules of non-idealities:**
  - Saturation
  - Dead zone
  - Columb function
Parameterization of modules

- Generally: Adaptivity of modules is realized by parametrization of modules; realization could be DSP SW, FPGA, maybe analog
  - setting parameters by instantiation

<table>
<thead>
<tr>
<th>Name value</th>
<th>Type</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>sc_module_name</td>
<td>-</td>
<td>name of instant module</td>
</tr>
<tr>
<td>_gain</td>
<td>double</td>
<td>-</td>
<td>gain in dB</td>
</tr>
<tr>
<td>_ip3</td>
<td>double</td>
<td>-</td>
<td>IP3 in dBm</td>
</tr>
<tr>
<td>_ideal</td>
<td>bool</td>
<td>-</td>
<td>true for simulation of ideal LNA, otherwise false</td>
</tr>
</tbody>
</table>

- parameter adjustments during simulation possible for some modules
User-specific model extensions

- The **open** nature of the AMS building block library enables making model extensions
  - without changing the design architecture or structure!
- Example: Add non-ideal effects to the model (e.g., noise)

Model-based design with Matlab/Simulink®

Open model-based design approach

```c
void processing() {
    out.write(ltf(...)+noise());
}
```
Building block example: LNA header file

```cpp
SCA_TDF_MODULE (lna)
{
public:
    sca_tdf::sca_in<double> in; // Input port
    sca_tdf::sca_out<double> out; // Output port

private:
    double gain; // Gain in dB
    double ip3;  // Third Input Intercept Point in dBm

    // Coefficients of output polynomial v = a*i - b*i*i - c*i*i*i
    double a, b, c;

    bool ideal; // Ideal LNA or not, true --> ideal...

public:
    // Constructor: name, gain in dB, ip3 in dBm, ideal (bool)
    lna (sc_core::sc_module_name n, double _gain, double _ip3, bool _ideal);
... 

private:
    void processing(); // Timed Data Flow (TDF) processing method
};
```
Example: OFDM transceiver (1)

- OFDM: Orthogonal frequency-division multiplexing
Example: OFDM transceiver (2)

- **Structure of the application:**

![Diagram of OFDM transceiver]

- Source
- s2p
- IFFT
- OFDM Transmitter
- Quadrature Mixer
- OFDM Receiver
- Voltage Detector
- Quadrature Mixer
- Drain
- p2s
- FFT
- s2p

Air channel
Example: OFDM transceiver (2)

```c
/***/
rand_bool i_stimuli("stimuli",16);
i_stimuli.out(sig_stimuli);
i_stimuli.out.set_timestep(1/freq_bit,SC_SEC);
/***/
ofdm_se<8> i_tran("transmitter",freq_carrier,const1_dim,freq_bit,data_rate,ampl_se);
i_tran.in(sig_stimuli);
i_tran.out(sig_out);
/***/
air i_air("air",attent,"gauss_white",n_va);
i_air.in(sig_out);
i_air.out(sig_noise);
/***/
ofdm_re<8> i_receiver("receiver",freq_carrier,const1_dim,freq_bit,data_rate,ampl_re);
i_receiver.in(sig_noise);
i_receiver.out(sig_received);
/***/
drain drn("drn");
drn.in(sig_received);
```
Simulation results (1)

Parameters for simulation: (no noise, no attenuation)

constl_dim = 16,  n_va = 0.0,  ampl_se = 1.0,  ampl_re = 1.0,  atten = 0.0

Parameters for simulation: (with strong white noise, no attenuation)

constl_dim = 16,  n_va = 90.0,  ampl_se = 1.0,  ampl_re = 1.0,  atten = 0.0
Simulation results (2)

Parameters for simulation: (with strong Gaussian noise and 50% attenuation)

\[
\begin{align*}
\text{constl\_dim} &= 16, \\
\text{n\_va} &= 90.0, \\
\text{ampl\_se} &= 1.0, \\
\text{ampl\_re} &= 1.0, \\
\text{atten} &= 0.5
\end{align*}
\]

Parameters for simulation: (with strong white noise and 50% attenuation)

\[
\begin{align*}
\text{constl\_dim} &= 16, \\
\text{n\_va} &= 90.0, \\
\text{ampl\_se} &= 20.0, \\
\text{ampl\_re} &= 0.1, \\
\text{atten} &= 0.5
\end{align*}
\]

With higher transmission power, we can reproduce the correct signals again!
Simulation results (3)

Parameters for simulation: (with strong Gaussian noise and 50% attenuation)

\[\text{constl\_dim} = 4, \quad \text{n\_va} = 90.0, \quad \text{ampl\_se} = 1.0, \quad \text{ampl\_re} = 1.0, \quad \text{atten} = 0.5\]

Or, we can also slow down the transmission to improve the Bit Error Rate (BER).
Conclusions and future work

- It is convenient to model communication systems using the AMS building block library
  - Open Source building block library is published and can be downloaded from [http://www.systemc-ams.org/](http://www.systemc-ams.org/)
  - Already used by several companies for research purpose

- Extend the library with technology dependent information
  - Evaluation of area and power consumption
  - Enabling architecture exploration use cases
Thank you

If you would like to download a copy of the presentation slides in PDF format, click the Attachments link at the top of the presentation viewer.