Design Refinement of Embedded Analog/Mixed-Signal Systems with SystemC AMS extensions

Prof. Dr. habil. Christoph Grimm
Chair Embedded Systems
Vienna University of Technology

*partially supported by
EC FP 6 „ANDRES“, No. IST-5-033511
MEDEA+ Project GEODES/FI-IT SNOPS
Beyond Moore: E-AMS

- Tightly interwoven **DSP-software** and **analog circuit** functionality: **Embedded Analog/Mixed-Signal System (E-AMS)**

Design of E-AMS

**Functional**
- (executable) Specification: 
  - Matlab/Simulink, Ptolemy [Lee, UCB], OMNET, C/C++

**Architecture**
- HW/SW Co-Design: 
  - SystemC (OSCI, IEEE 1666)

**Implementation**
- Digital: VHDL,...
- Software: C/C++
- Analog design: VHDL-AMS, Verilog-AMS, SPICE

**Methodology & Language**
for „Analog/Digital Co-Design“

**Integration validation**
- FastSPICE, ...

C. Grimm: Refinement of E-AMS Systems using SystemC AMS extensions
Design refinement of E-AMS systems

1.) SystemC AMS extensions

2.) Design refinement of E-AMS

3.) Outlook
SystemC AMS extensions

- OSCI AMS WG driven by academia and industry

- Chair: Martin Barnasconi (NXP),
  Co-Chair: Christoph Grimm (TU Wien)
## SystemC AMS extensions

<table>
<thead>
<tr>
<th>SystemC methodology-specific elements</th>
<th>AMS methodology-specific elements for AMS design refinement, etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction Level Modeling</td>
<td>Electrical Linear Networks (ELN) modules terminals nodes</td>
</tr>
<tr>
<td>Cycle/bit Accurate Modeling</td>
<td>Linear Signal Flow (LSF) modules ports signals</td>
</tr>
<tr>
<td>etc.</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>
<tr>
<td></td>
<td>Linear DAE solver</td>
</tr>
<tr>
<td></td>
<td>Scheduler</td>
</tr>
<tr>
<td></td>
<td>Synchronization</td>
</tr>
<tr>
<td></td>
<td>SystemC Language Standard</td>
</tr>
</tbody>
</table>

Timed Data Flow in SystemC AMS extensions

“cluster“ := set of connected TDF modules

\[ \text{order of computation} \]

\[ \text{cluster period} \]

\[ \text{serial} \quad f1 \quad \text{deser} \]

\[ \text{delay=1} \]

\[ \text{rate=8} \]

\[ T=1 \quad t0=0 \]
### Timed Data Flow Block: Serializer

<table>
<thead>
<tr>
<th>TDF Module: primitive module!</th>
<th>SCA_TDF_MODULE(serial)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>sca_tdf::sca_in&lt;sc_bv&lt;8&gt; &gt; in;</td>
</tr>
<tr>
<td></td>
<td>sca_tdf::sca_out&lt;bool&gt; out;</td>
</tr>
<tr>
<td>Attributes specify timed semantics</td>
<td>void set_attributes()</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>out.set_rate(8);</td>
</tr>
<tr>
<td></td>
<td>//out.set_delay(1);</td>
</tr>
<tr>
<td></td>
<td>//out.set_timestep(1, SC_MS);</td>
</tr>
<tr>
<td>processing() describes computation</td>
<td>void processing()</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>for (int i=7; i &gt;= 0 ; i-- )</td>
</tr>
<tr>
<td></td>
<td>out.write(in.get_bit(i), i);</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>SCA_CTOR(serial);</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
</tbody>
</table>
Interfacing Timed Data Flow and SystemC (DE)

Converter ports towards discrete event domain

```cpp
sca_tdf::sc_in < <type> >
sca_tdf::sc_out < <type> >
```

Note: Time in MR – TDF may run ahead DE time!

```cpp
sc_time sca_get_time()
```
SC_MODULE(lp_filter_eln) {
    sca_tdf::sca_in<double> in;
    sca_tdf::sca_out<double> out;
    sca_eln::sca_node in_node, out_node; // nodes
    sca_eln::sca_node_ref gnd;           // reference
    sca_eln::sca_r *r1;                 // resistor
    sca_eln::sca_c *c1;                 // capacitor
    sca_eln::sca_tdf2v *v_in;           // converter TDF->U
    sca_eln::sca_v2tdf *v_out;          // converter U->TDF
    SC_CTOR(lp_filter_eln) {
        v_in = new sca_eln::sca_tdf2v("v_in", 1.0); // scale factor 1.0
        v_in->ctrl(in); v_in->p(in_node); v_in->n(gnd);
        r1 = new sca_eln::sca_r("r1", 10e3);       // 10kOhm resistor
        r1->p(in); 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_v2tdf("v_out", 1.0); // scale factor 1.0
        v_out1->p(out_node); v_out1->n(gnd); v_out->ctrl(out);
    }
};
Design refinement of E-AMS systems

1.) SystemC AMS extensions

2.) Design refinement of E-AMS

3.) Outlook
Design Refinement is a top-down design methodology that stepwise and interactively augments an executable, functional specification with properties of intended implementation at architecture level.

- Similar, but not the same:
  - Extreme programming, refactoring (SW Engineering)
  - Property refinement (Formal method to ensure safety properties)
What is needed for „Design Refinement“?

1. A *single* language that combines functional level with architecture/implementation level:

   ➔ **SystemC AMS extensions** (OSCI)

2. Design methodology specific support library

   ➔ **HEAVEN** (TU Vienna’s Heterogeneous Embedded Systems Analysis/Refinement Environment)
(executable) Specification:

Matlab/Simulink, Ptolemy [Lee, UCB], OMNET, C/C++

1.) **Refinement of computation**
   Algorithms, data types, physical effects/accuracy

2.) **Refinement of structure**
   Mapping functional blocks to concrete processors

3.) **Refinement of interfaces**
   Signals, synchronization, bus protocols

Digital: VHDL,...
Software: C/C++
Analog design: VHDL-AMS, Verilog-AMS, SPICE

System integration, Postlayout, Test

*FastSPICE, ...*
1.) Refinement of computation

**Objective:** Evaluate impact of non-ideal behavior of assumed architecture/implementation using functional model

**Method:** Add non-ideal effects to executable specification by modifying block-behavior, e.g.:

```c
void processing() // Mixer with distortions and noise
{
  double rf = in1.read(); double lo = in2.read();
  double rf_dist = (alpha - gamma * rf * rf) * rf;
  double mix_dist = rf_dist * lo;
  if_out.write( mix_dist + my_noise() );
}
```

C. Grimm: Refinement of E-AMS Systems using SystemC AMS extensions 16
2.) Refinement of structure

- **Objective:** Compare performance of different A/D/HW/SW partitionings more accurately
- Map functions to assumed or existing processor (=change structure, MoC)

![Diagram of system architecture](image)

MoC: TDF

MoC: ELN

MoC: TLM
Integration in functional model „interactive“?

HW/SW: DSP, RAM, CPU

messages (transactions) to Processors, ...

Signal/dataflow: Numbers

Digital circuit:
- pinaccurate, single bits

Analog circuit:
- Electrical nodes, physical sizes

Electrical nodes, physical sizes
2.) Refinement of structure

Converter channel from HEAVEN library:
- Automatically do conversion of signal types, ranges, ...
- Enable interactive replacement of functional blocks by implementation

Refinement of structure, re-partitioning

- **Method 2 (top-down):**
  Re-write functional part that is re-partitioned in new model of computation that matches intended realization (cumbersome ...)
  - Analog: Signal flow, Network
  - Digital HW: TDF, DE (or TLM)
- *Slightly simplified by “static polymorphism”*

```cpp
SCA_MoC_MODULE(par2ser)
{
    sca_MoC::sca_in<sc_bv<8>> in;
    sca_MoC::sca_out<bool> out;
    ...
    SCA_CTOR(par2ser)
}
```
3.) Refinement of interfaces

- **Intention:** Prepare validation of system integration - All ports accurately as in implementation (same types, same number, clocks, enable-signals, ...)

- **Method:**
  Adapter classes translate between abstract data flow and (DE) pin-accurate protocols or analog nodes (requires appropriate models inside, known from SystemC, TLM, ...)
HEAVEN bundles support for refinement

- Converter channels
  - MoC supported: TDF, TLM, DE, Analog circuits
  - Data types: double, bitvector, logic_vector
  - Simulators: CADENCE, Matlab Simulink

- Adapter classes (at the moment: SSIO)

- Functional blocks & behavioral models for communication systems

- Analysis tools
  - Power estimation
  - Eye diagrams, Trellis diagrams, Constellation diagrams, ...
Analysis Tools: Estimation Power Consumption

- Estimation via
  - Usage of functionality
  - ISS
- “Calibration” by
  - Circuit simulation
  - Measurement
  - Data sheets
- “Usage profile” in log file
- Post Processing: Analysis, Visualization
Typical Simulation Scenario: Wireless Sensor Network

- User interactivity
- Energy: Light source
- RF-channel
- Environment properties: Temperature, Vibration
- Node properties
- Position thickness material: Obstacle
- Velocity direction: Dynamics
- Sensor node
- Environment
Design Refinement aims at higher productivity
- Hard to measure / compared …

Statistics of industrial application
- Executable spec, Integration validation w/ Verilog-AMS
  ➔ Today most SoC projects „first time right“
- Architecture exploration by refinement (+SystemC AMS?)
  ➔ Target: 30% reduction of design time
Design refinement of E-AMS systems

1.) SystemC AMS extensions

2.) Design refinement of E-AMS

3.) Outlook
Summary, Future work

- **SystemC AMS extensions** provide appropriate means for architecture exploration of E-AMS systems

- Outlook SystemC AMS
  - 1st Draft of LRM available on [www.systemc.org](http://www.systemc.org)
  - public review – please comment!
Summary, Future Work

- **Design refinement** could increase design productivity by
  a) Quickly available first models
  b) Immediate analysis/verification after changing/adding property

- **HEAVEN library:**
  - *Communication with CADENCE Design Framework*
References

- www.systemc.org
  (OSCI members)
- www.systemc-ams.org
  (For information from former SystemC-AMS SG, provides some information for the public)