Towards Co-Design of HW/SW/Analog/RF Systems

Prof. Dr. Christoph Grimm

Many thanks to Markus Damm, Jan Haase, Javier Moreno, Thomas Herndl, Stefan Mahlknecht, Josef Wenninger, Sumit Adhikari
Why Co-Design?

**Simple Answer:** better performances, higher productivity!

- Higher productivity of top-down approach:
  - Ressource budgeting (delay, power, ...) enables concurrent design
  - Issues are detected earlier – „late fails“ less likely

- Better Performances
  - Architecture/system level defines performances today!
  - Architecture can be improved incrementally

*HW/SW ok, but:*

*Why and how also include AMS, RF, application?*
*Which languages? Which tools?*
Virtual Prototyping and Power Profiling

1. Why going beyond HW/SW Co-Design?
2. How including AMS/RF?
3. Which tools?
4. Conclusion
Input:
Functional model, fixed constraints (deadlines, power, etc.), stimuli

HW/SW Co-Design
1. Executable specification
2. Architecture exploration || Virtual Prototyping
3. HW Development || SW Development
4. Integration validation

Output:
Processor(s) + Software that guarantee fixed deadlines, power, etc. in the worst case
1\textsuperscript{st} Chance missed: Optimization across domain borders

**Partitioning Analog/SW**

- **Direct:**
  - ADC, sampling
  - SW: Polling ADC

- **Optimized:**
  - 2 analog comparators → IRQ
  - SW: ISR

---

**Software**

```c
If ( ub.read() )
  out.write( -1.0 );
If ( lb.read() )
  out.write( 1.0 );
```

---

**Analog**

- **integ dt**
- **ul**
- **ll**

---

**Analog? SW?**
2nd Chance missed: Optimization across layers

**Usage** of hardware defined by
- Scenario & Application
- Communication protocols
- Architecture

**X-Layer optimization:**
- Know application and scenarios
- Develop power management strategy
- Match with RT-Level infrastructure
- Match with technology
- Battery for years!

**Power** consumed in hardware,
Battery for days.
3\textsuperscript{rd} Chance missed: Co-Optimization with Application

Instead of fixed deadlines, more flexible quality of control (Chakraborty, FDL 2012):

\[
\text{Gradient of QoC: } \mathcal{P}(D) = \frac{J_0 - J(D)}{J_0}
\]

**Required:** Co-simulation of whole system with environment
⇒ SystemC + Scenario Environment (analog!)

⇒ Benefit: Use of processor features like CACHE and BRANCH PREDICTION in real-time ES.
X-Layer, X-Domain dependencies

<table>
<thead>
<tr>
<th>Property of System</th>
<th>Depends from its component’s properties</th>
<th>Dependencies across layers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>linear</td>
<td>Additive from bottom-up</td>
</tr>
<tr>
<td>Delay</td>
<td>linear on path</td>
<td>Additive on path from bottom-up</td>
</tr>
<tr>
<td>Power</td>
<td>linear</td>
<td>Bottom-up and top-down, non-linear (activity-driven)</td>
</tr>
<tr>
<td>Accuracy, Dependability</td>
<td>nonlinear</td>
<td>Bottom-up and top-down, non-linear (functionality-driven)</td>
</tr>
</tbody>
</table>
Virtual Prototyping and Power Profiling

1. Why going beyond HW/SW Co-Design?
2. How include AMS/RF?
3. Which tools?
4. Conclusion
HW/SW/Analog/RF Co-Design flow (today)

- Method development
- HW/SW Co-Design
- Integration/ validation ...
- Platforms, IP
- Analog design
- Previous design
- Knowledge, experience, ...
- Previous design
- Method development
Concrete Objectives for including AMS/RF

1. Capture existing knowledge, make it available
   - during method development
   - Introduce *overall* architecture exploration phase

2. Detect design risks *early*
   - without existing design
   - But: quantified impact of „problems“ and design risks on outcome

3. **Systematically** do overall architecture exploration
   - Early and close-to-optimal resource budgeting (less re-designs)
   - Improve system partitioning (better performances)
HW/SW/Analog/RF Co-Design flow (1)

Overall architecture exploration:
- Partitioning/Mapping
- Analysis of risks
- Resource budgeting

Method development

HW / SW Co-Design

Analog design

Integration validation

Platforms, IP

Previous design

Knowledge, experience, ...
HW/SW/Analog/RF Co-Design flow (2)

- Method development
- Overall architecture exploration
- Templates of Building blocks
  - Incl. power, inaccuracies, noise, ...
- Characterized models:
  - Specification of intended budgets
  - Validation of system integration
- Integration validation

- Platforms, IP
- Previous design

Characterization:
- Achieved performances

Knowledge, experience, ...
Requirements for tools

- **Design Language** needed
  - HW from method development down to RT level
  - SW and OS modeling – in C!
  - AMS from method development down to circuit level
  - Mixing layers and domains -> single language

- **Means/tools** to
  - Capture experiences, design results
  - Bring it method development & architecture exploration
  - Evaluation of performances (functionality, power, accuracy, ...) at architecture level
Virtual Prototyping and Power Profiling

1. Why going beyond HW/SW Co-Design?
2. How including AMS and RF?
3. Which tools? SystemC!
4. Conclusion
Requirements for tools

- **Design Language** needed
  - HW from method development down to RT level
  - SW and OS modeling – in C!
  - AMS from method development down to maco models
  - Mixing layers and domains - single language!

- **Means/tools** to
  - capture experiences, design results
  - Bring it method development & architecture exploration
  - Evaluation of performances (functionality, power, accuracy, ...) at architecture level
SystemC + TLM + AMS extensions!
+ Home-brew: sca-comlib

Building block library for communication systems
- Blocks à la Matlab communication systems toolkit
- Power models
- Parameterizable models of non-idealities
- Sources, sinks, analysis tools for power, accuracy,...

Available for free from TU Kaiserslautern
Bringing expert’s knowledge to architecture level

Complete model with assumed/measured parameters!

<table>
<thead>
<tr>
<th>Parameter</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></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>
Power estimation with SystemC (AMS)

- Annotate code with calls that change power states
- Group software into activities for analysis
- Visualization

- Power optimization: Use of TX, RX more relevant than e.g. CPU!

Figure: Quanto: Tracking Energy in Networked Embedded Systems; Rodrigo Fonseca, Prabal Dutta, Philip Levis, Ion Stoica; Proceedings of the Eighth USENIX Symposium on Operating System Design and Implementation (OSDI'08)
X-Layer optimization of power management

Execution of TDF processes controlled by clk(enable) signals (events), SW by Interrupts (events)

Ch. Grimm: Virtual Prototyping and Power Profiling
X-Domain partitioning

Add Instruction Set Simulator(s) of processors to be used

Protocol Processor ASIC, DSP

‡P Memory

Software

Applikation Phy-App Layer

Control & Power Management

PowerDown

Clk, Enable

Interrupt

Ch. Grimm: Virtual Prototyping and Power Profiling
Example: Power profiling of TPMS with In-Car WSN; 18 8-Bit-uC with Firmware + Transceiver + Sensors
Virtual Prototyping and Power Profiling

1. Why going beyond HW/SW Co-Design?
2. How include AMS/RF?
3. Which tools?
4. Conclusion
Conclusion

- SystemC enables HW/SW/AMS/RF Co-Design
  - TLM, AMS extensions
  - also needed: design libraries
  - Can be extended easily to model power

- SystemC AMS 2.0 is huge step ahead
  - Quasi-dynamic data flow significantly broadens scope
  - Key for modeling power management, etc.