



Design refinement for wireless sensor networks (using SystemC AMS extensions)

Institut für
Computertechnik

ICT

Institute of
Computer Technology

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

Computer Technology



Heinz Zemanek and his „Mailüfterl“: First transistor computer. (TU Vienna/ICT, 1955)



Today, computers are very small and very cheap – the end of development?

More than Moore!

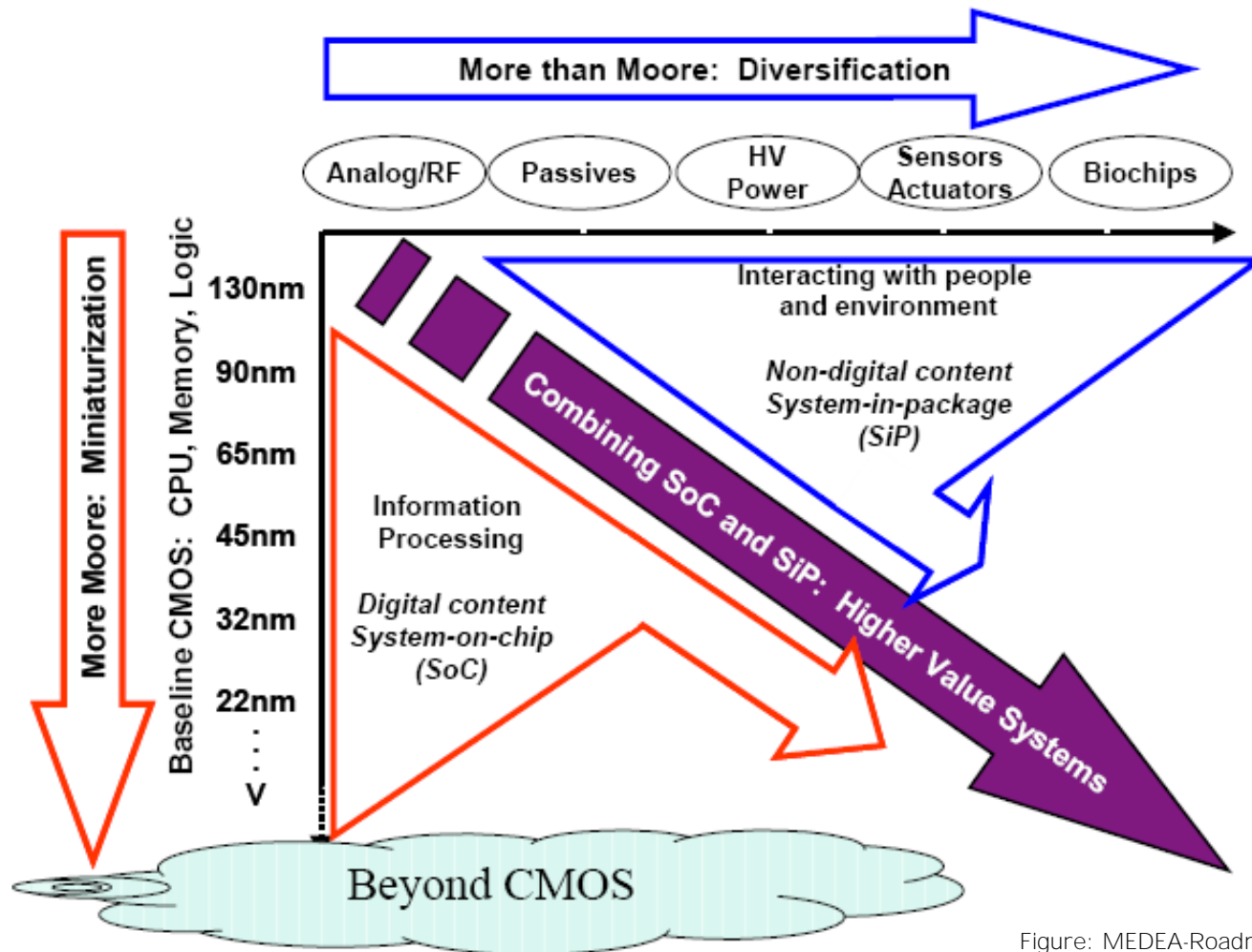


Figure: MEDEA-Roadmap

Wireless ultra-low power sensor networks

- **Introduction**
- Design challenges
- OSCI SystemC AMS extensions
- Support for refinement of sensor networks

Sensor networks (e.g. in ambient assisted living)

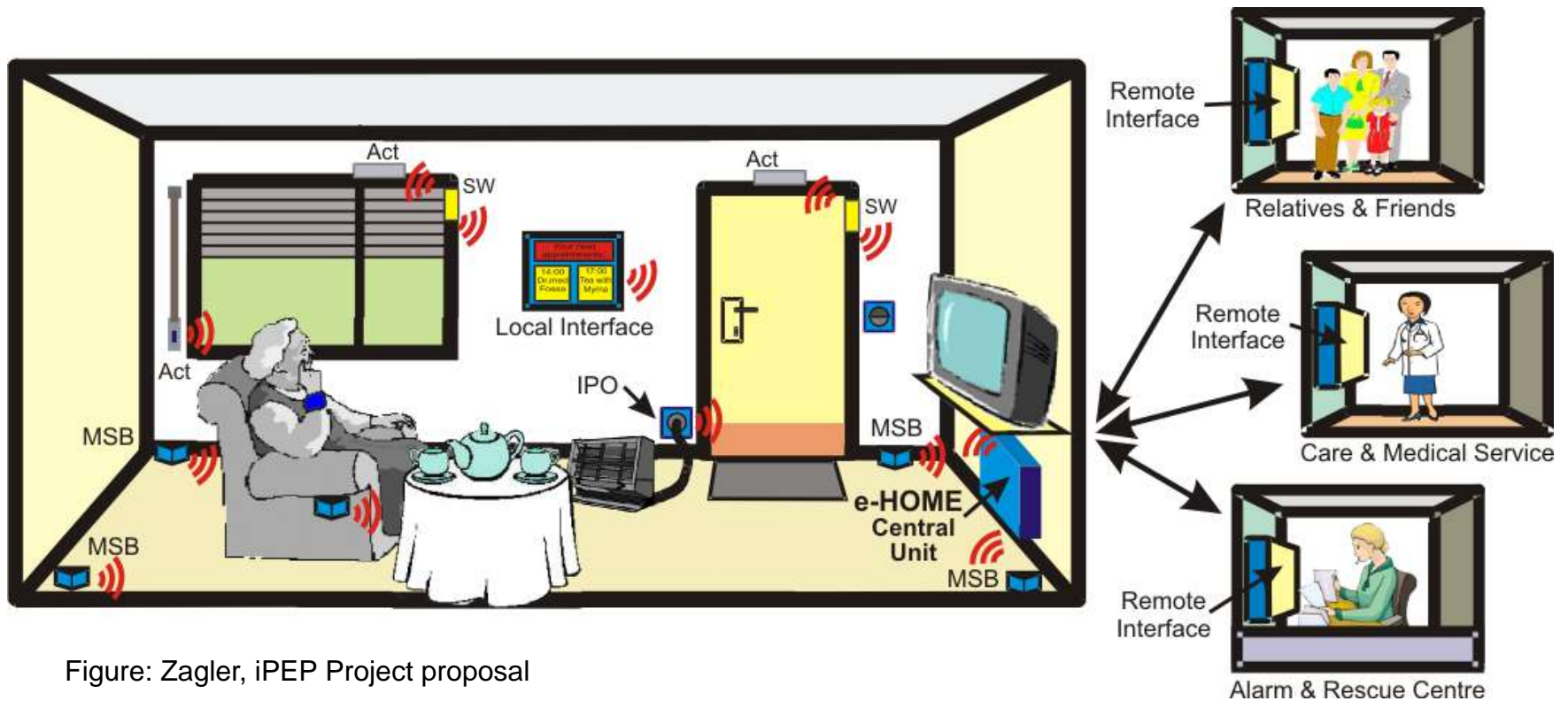
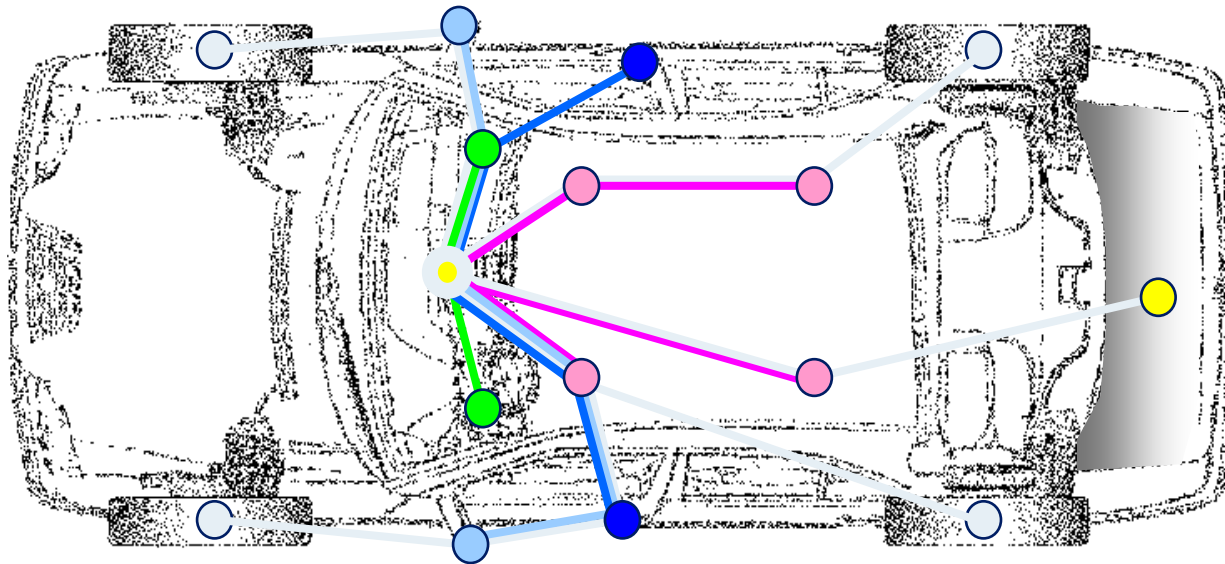


Figure: Zagler, iPEP Project proposal

Sensor networks (e.g. in automotive application)

- Tire pressure metering ...



- Sensors will be embedded in tire and „harvest“ energy
- Multi-hop protocol increases dependability

Ultra-low power wireless sensor network

(Ultra-low power wireless) „Sensor networks“

- Growing interest from industry
- Energy autonomy for „lifetime“

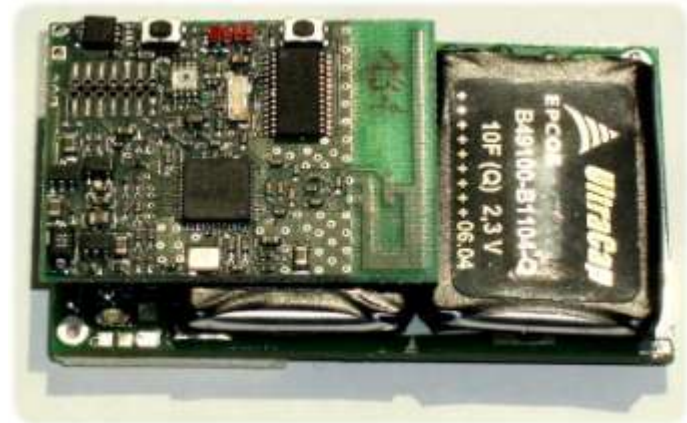
Some typical requirements:

- mikrowatt radio (average consumption: some μA)
- low duty cycle ($< 1\%$)
- low throughput (1-10kbps)
- Often helpful: very cheap, very small ...

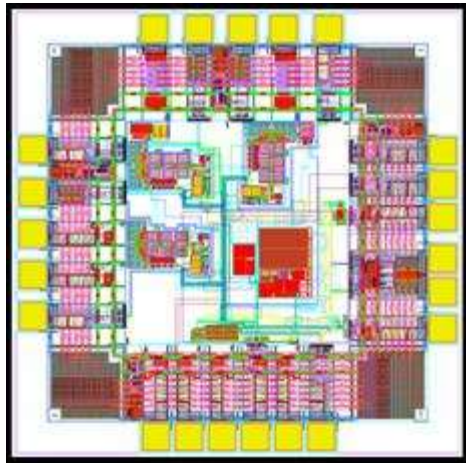
Ultra-low power wireless sensor network node

Typical ingredients

- Sensing unit
- Signal conditioning
- ADC, DAC
- DSP
- Protocol Stack (e.g ZIGBEE simplified/adapted)
- Energy harvester , energy scavenging
e.g. Solar cell, Antenna, MEMS, ...

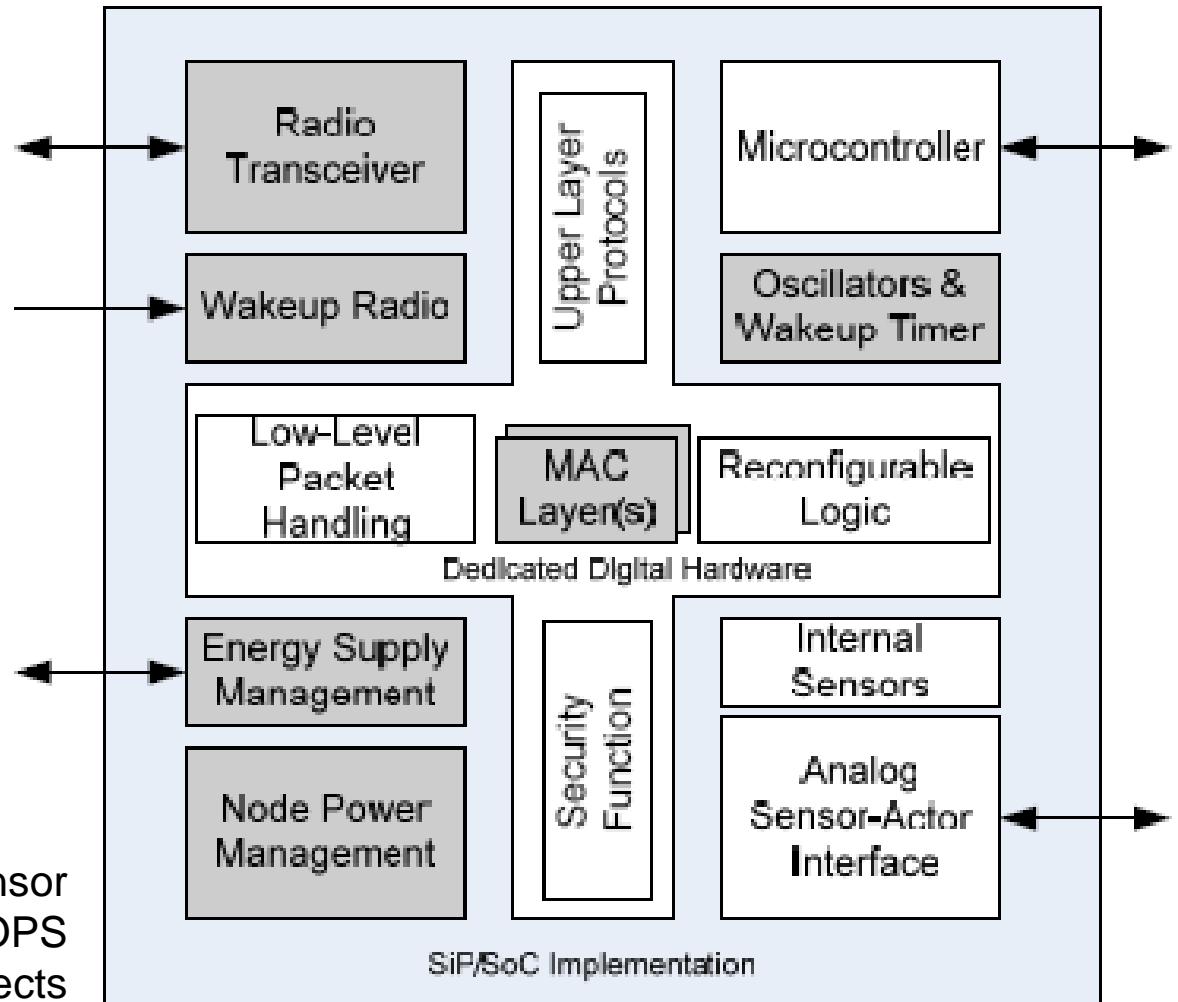


Architecture from ICT's PAWIS node



Wakeup radio designed by Johann Glaser/ICT

Architecture of sensor nodes from PAWIS/SNOPS projects



Wireless ultra-low power sensor networks

- Introduction
- **Design challenges**
- OSCI SystemC AMS extensions
- Support for refinement of sensor networks

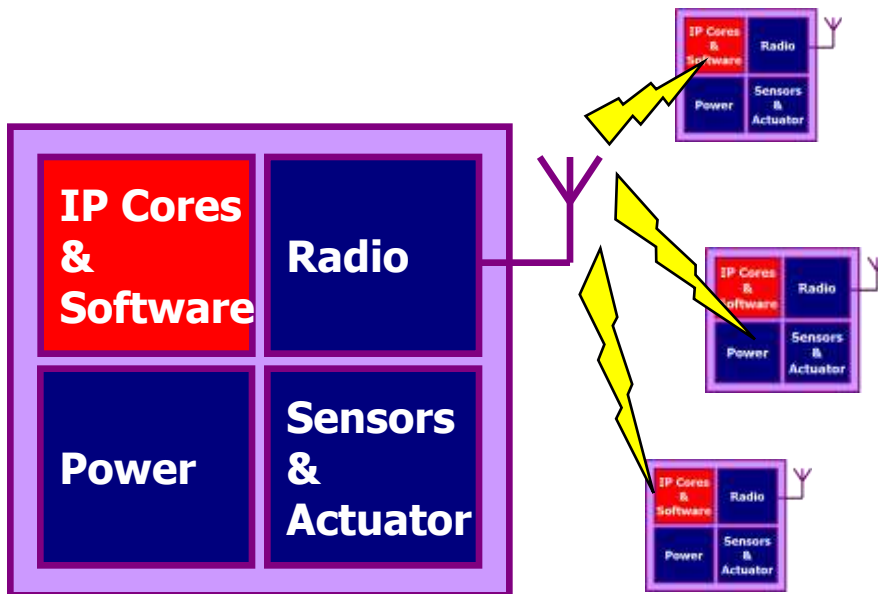
Design challenges (1)

- Process technology: Tradeoff size - leakage current
- Analog/digital partitioning
- Voltage matching problem: Combination of different technologies requires different voltages
- DC/DC converters & Power management: very low quiescent current!
- Oscillator start-up time

Design challenges (2)

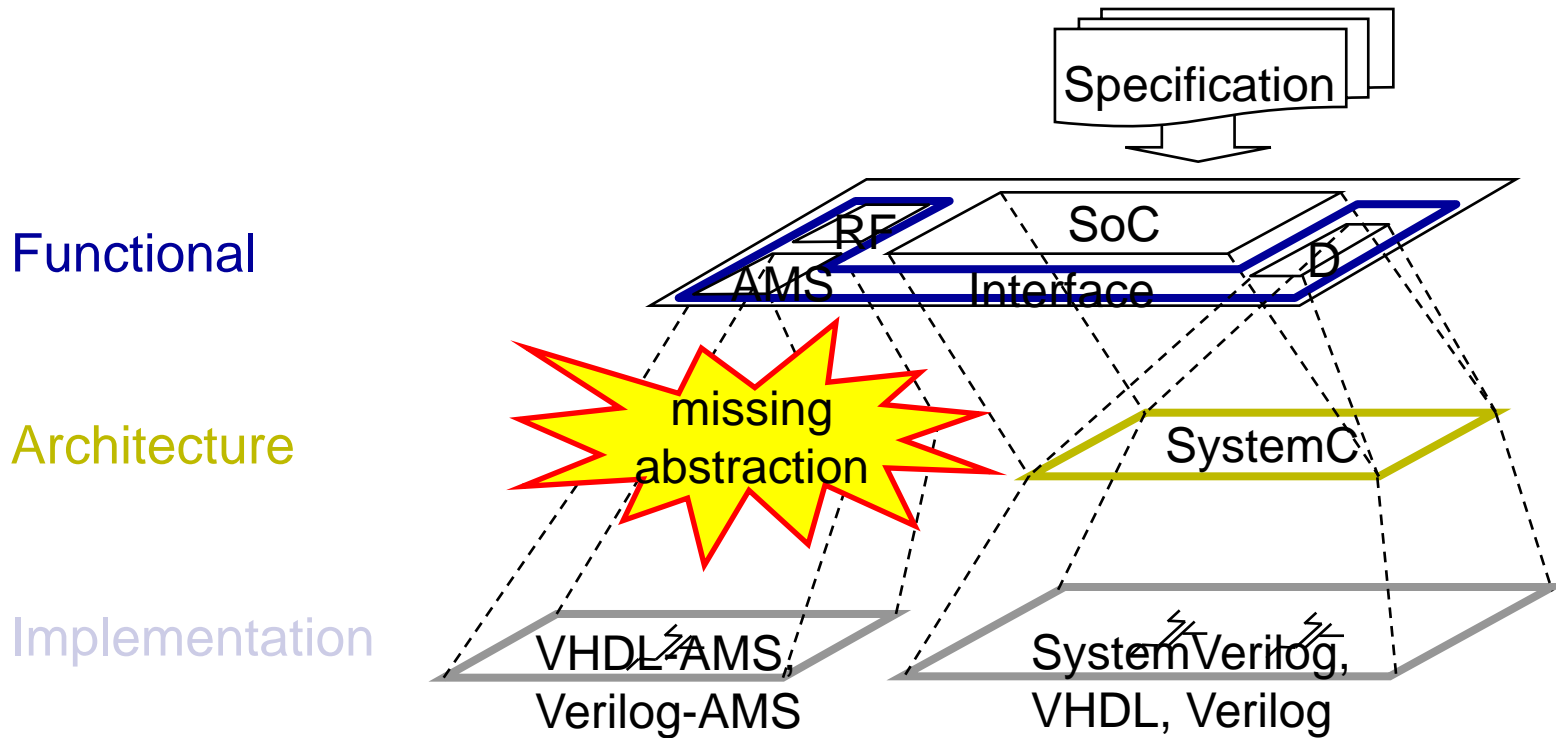
- Choice of modulation scheme
 - ON/OFF, ASK: simple + very energy efficient hardware, but not robust against noisy environment
 - FSK, more complex ones: less simple, but more robust at higher data rates
 - Combinations: Wake-up, control: ASK; rest: FSK+...
- Wakeup problem
- Optimization at protocol level
 - Multi-hop protocols - star network architecture
 - Which offers appropriate performance at lowest cost/power consumption?

Even *much* more than More than Moore

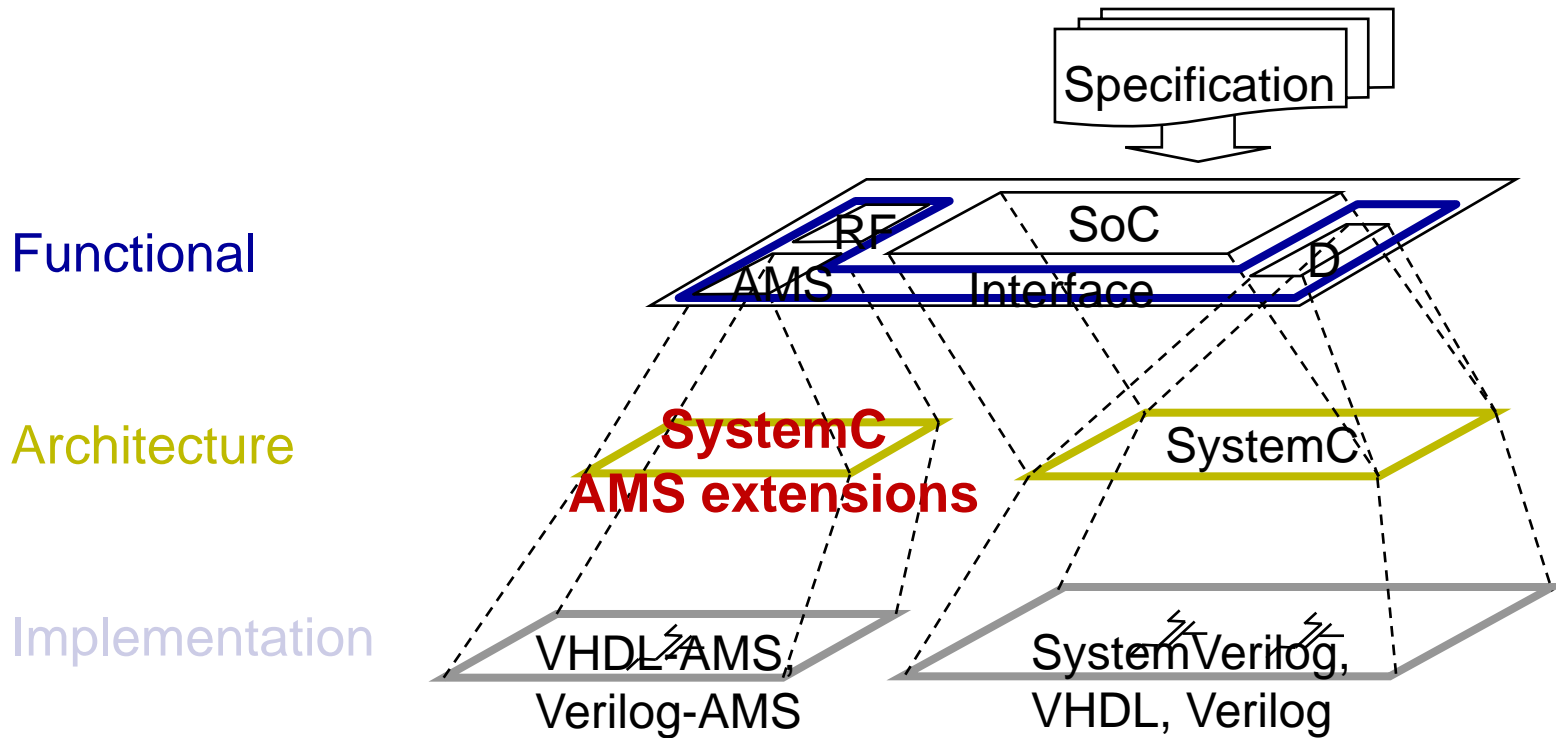


- **Network level**
OMNET++?
- **Functional level**
Simulink, Ptolemy, ...
- **Architecture level**
SystemC?
AMS extensions?
- **Block level, circuit level**
VHDL-AMS, Verilog-AMS,
...

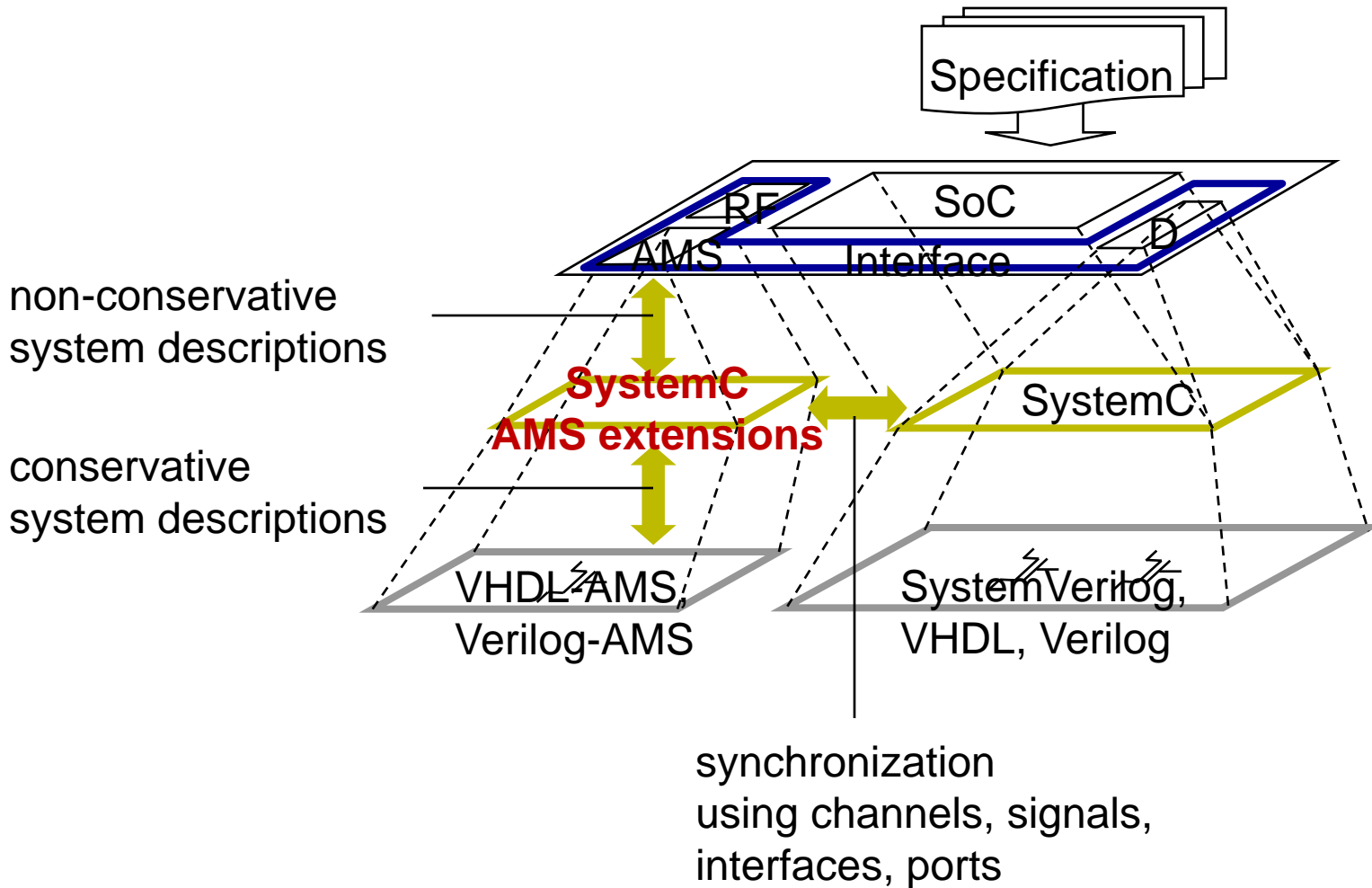
Gap at architecture level



Gap at architecture level



Gap at architecture level



Wireless ultra-low power sensor networks

- Introduction
- Design challenges and architecture gap
- **OSCI SystemC AMS extensions**
 - use cases
 - language
- Methods & support for design of sensor networks

Current OSCI AMS WG Roster



- Steady growth in AMS WG: 44 individuals from 19 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 Working Group scope

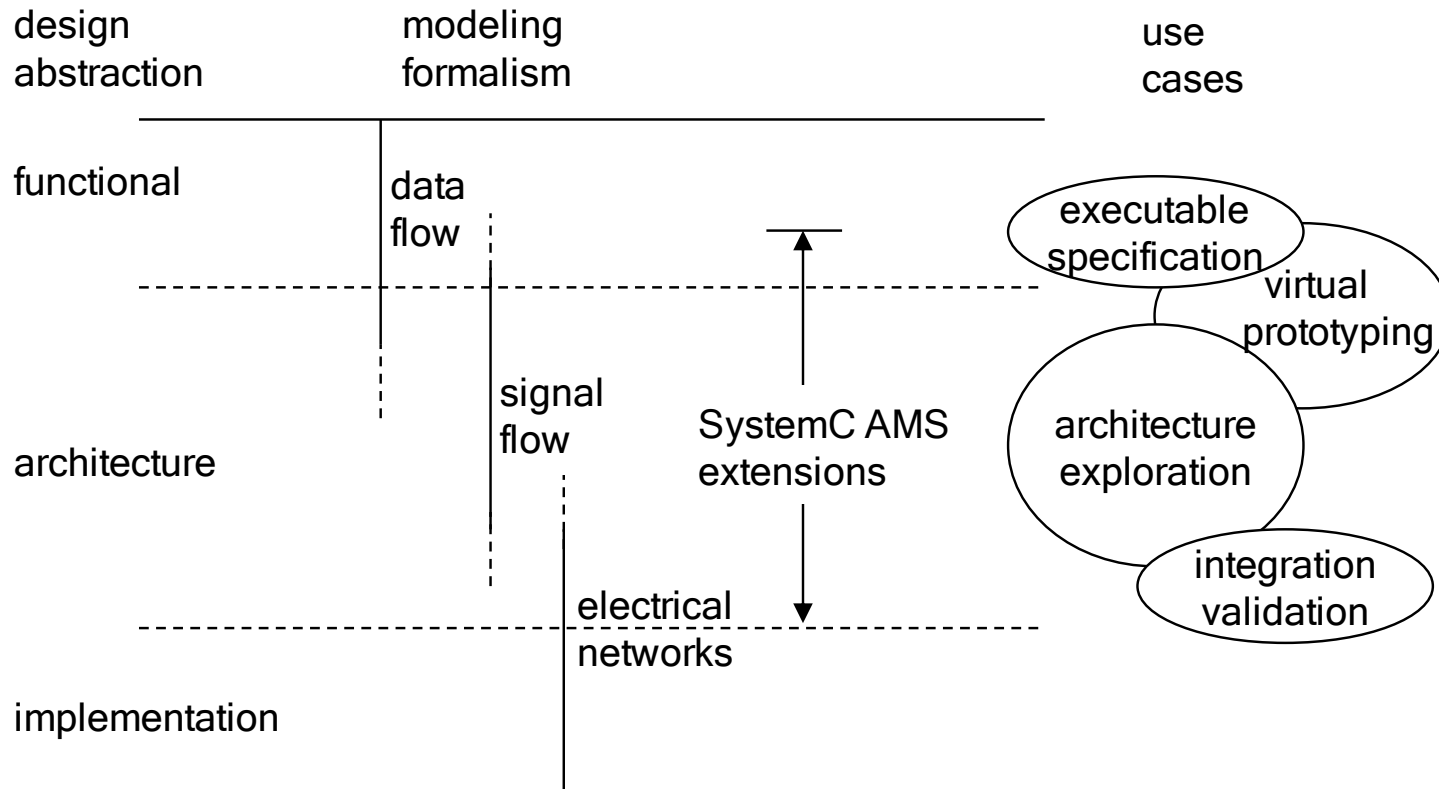
- Embedded analog/
mixed-signal systems
 - Heterogeneous systems including analog, mixed-signal, RF and digital HW/SW
- Targeted application domains
 - Wireless
 - Wired
 - Automotive
 - Imaging sensors

End Product Markets	2003	2004	2005	2006	2007
Microprocessor/DSP	18.9%	16.0%	13.1%	10.5%	14.7%
Computer, Peripheral	22.9%	21.6%	18.5%	24.2%	19.0%
Wired Network	11.2%	5.2%	5.8%	4.8%	5.2%
Wireless Network	13.1%	10.4%	13.1%	7.3%	6.9%
Multimedia	25.6%	34.2%	33.8%	37.9%	31.9%
Automotive	1.9%	3.0%	3.8%	4.0%	4.3%
Others	6.4%	9.7%	11.9%	11.3%	18.1%

source: SystemC Trends report, April 2007

focus of AMS
WG

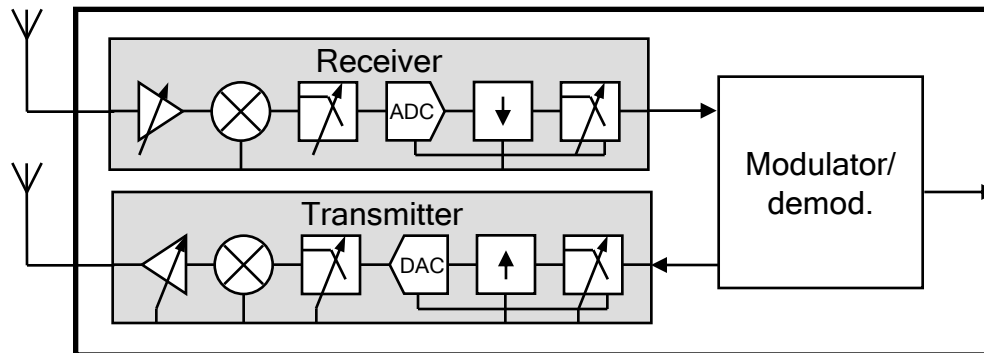
Between functional model and implementation



[Grimm/Barnasconi/Vachoux/Einwich: An Introduction to Modeling Embedded Analog/Mixed-Signal Systems using SystemC AMS Extensions. OSCI, June 2008]

Use case: executable specification

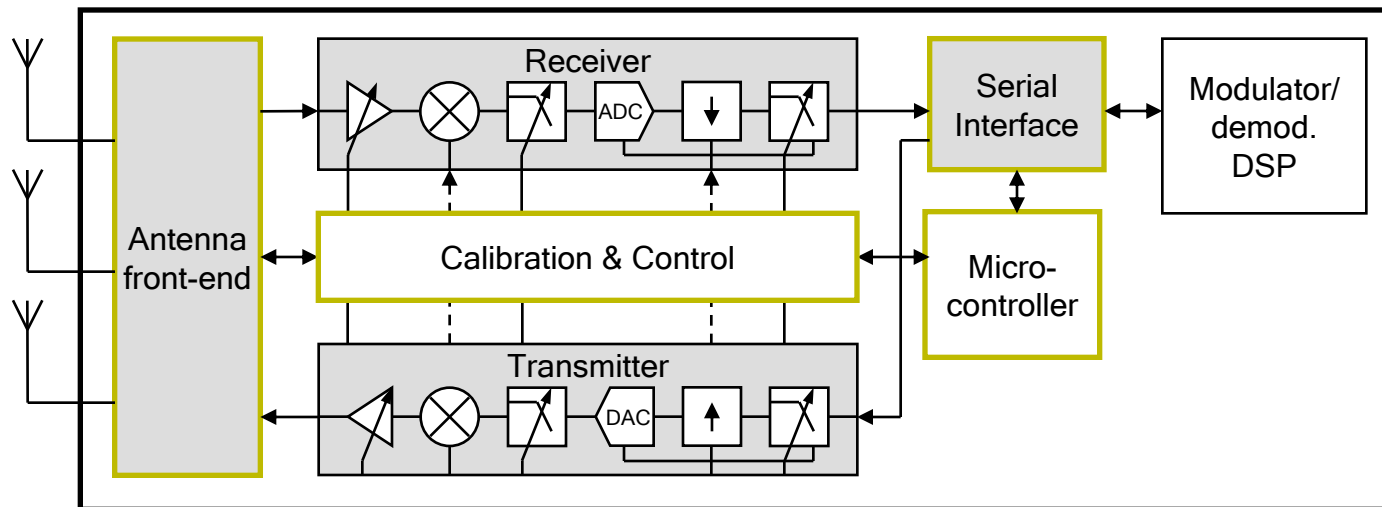
- Objective: verify the system specification by creating an executable description of the system using simulation
- Highest level of design abstraction applied, using functional models and signal processing algorithms
- Timed Data Flow and Linear Signal Flow modeling formalisms used



Transceiver based on functional models

Use case: architecture exploration

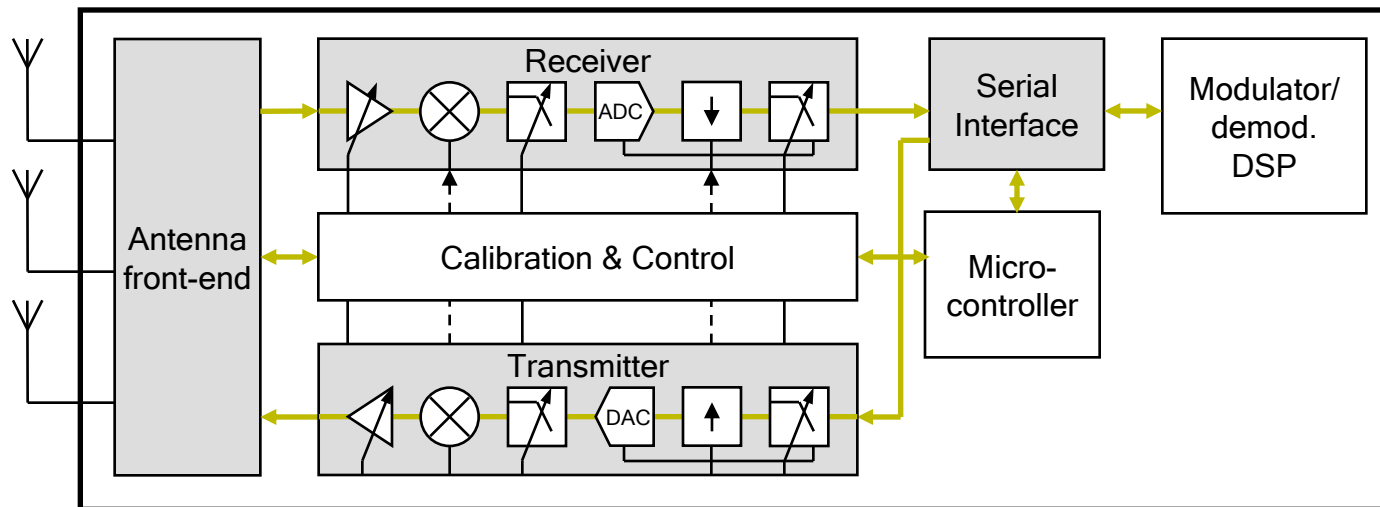
- Objective: determine, evaluate and dimension the key properties of the system architecture
- Two phases
 - Refinement of executable specification by adding non-ideal properties
 - Introduce interfaces components and architectural elements to match the final implementation



Transceiver including architecture elements (MCU, serial interface, ...)

Use case: integration validation

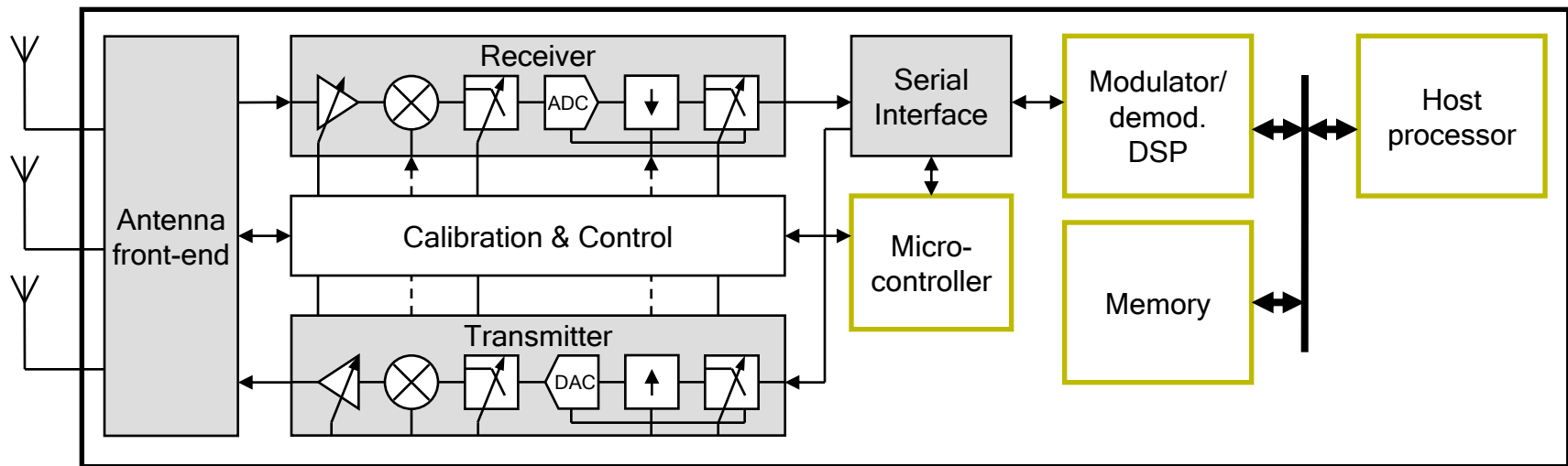
- Objective: accurate modeling of interfaces of all subsystems
 - analog circuits/subsystems: introduce electrical nodes
 - digital circuits/subsystems: bit/cycle accurate signals and busses
 - HW/SW subsystem (CPU, MCU): TLM interfaces



Transceiver subsystems modeled with accurate interfaces

Use case: virtual prototyping

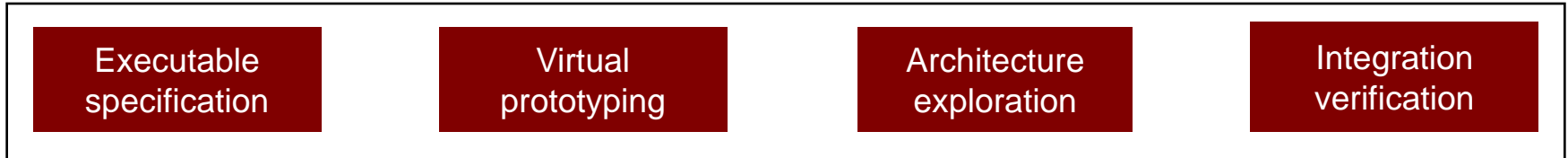
- Objective: SW development using a high-level untimed or timed model that represents the AMS sub-system
- Interoperability with SystemC TLM extensions expected
- Modeling formalism used:
Timed Data Flow, incorporating untimed models if needed



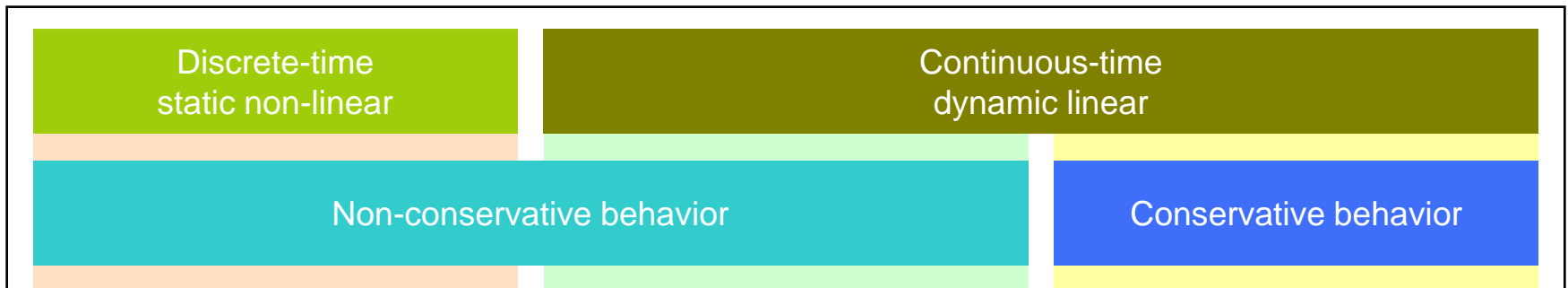
Transceiver subsystem connected to digital virtual platform

Model abstraction and formalisms

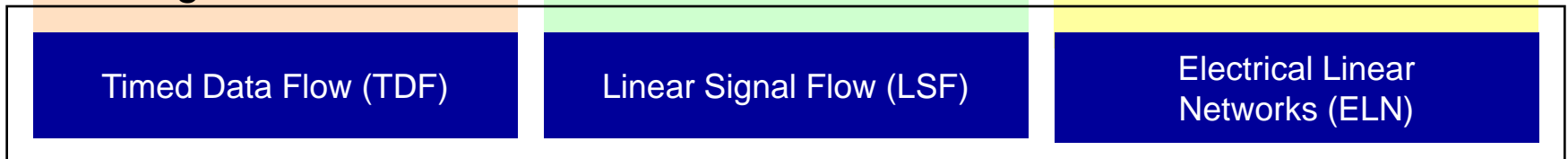
Use cases



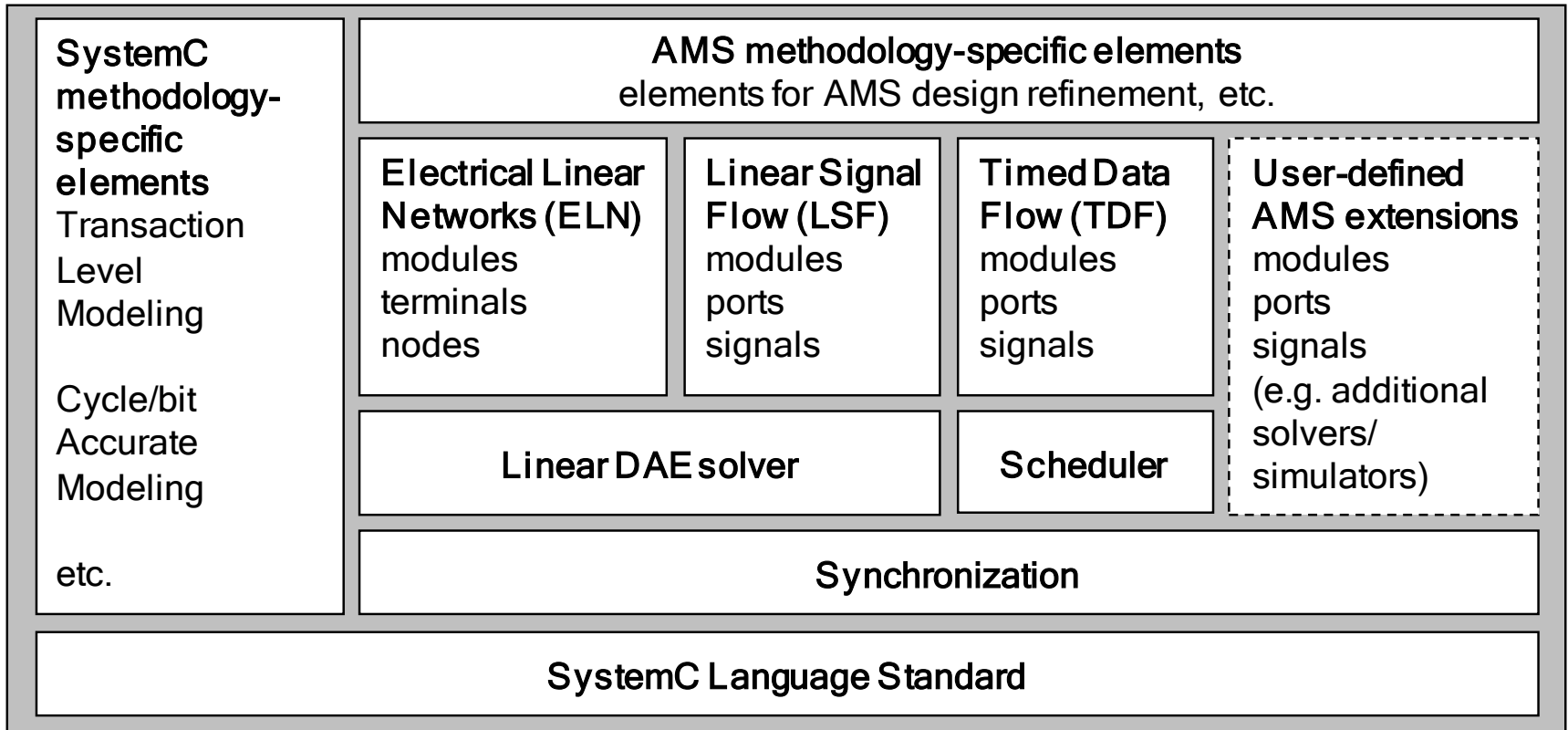
Model abstractions



Modeling formalism



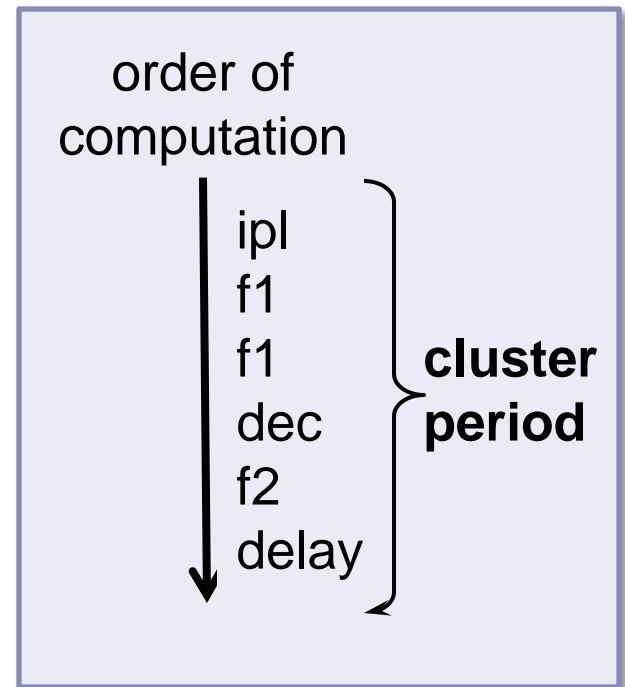
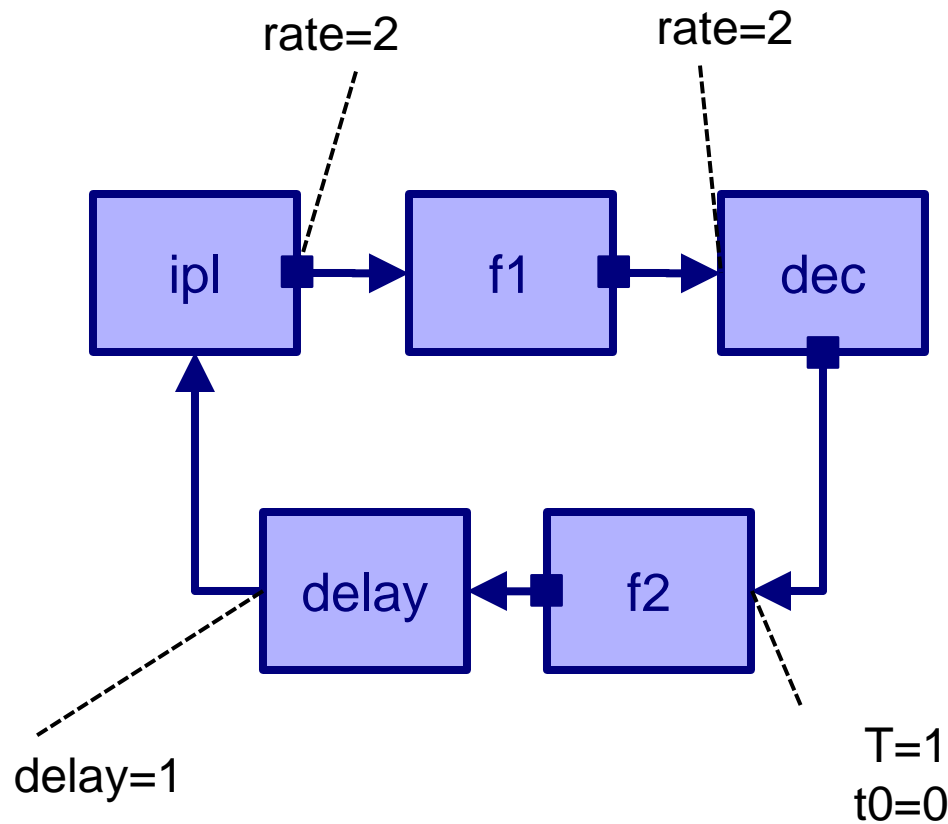
SystemC AMS extensions



[Grimm/Barnasconi/Vachoux/Einwich: An Introduction to Modeling Embedded Analog/Mixed-Signal Systems using SystemC AMS Extensions. OSCI, June 2008]

Timed Data Flow

„cluster“ := set of connected TDF modules



Timed Data Flow: **Serializer Example**

TDF Module:
primitive module!

```
SCA_TDF_MODULE(par2ser)
{
    sca_tdf::sca_in<sc_bv<8> > in;
    sca_tdf::sca_out<bool>      out;
```

Attributes specify
timed semantics

```
void set_attributes()
{ out.set_rate(8);
  out.set_delay(1);
  out.set_timestep(1, SC_MS);}
```

processing()
describes
computation

```
void processing()
{
    for (int i=7; i >= 0 ; i-- )
        out.write(in.get_bit(i), i);
}
SCA_CTOR(par2ser);
}
```

Timed Data Flow: Connecting to SystemC

Converter **ports** towards
discrete event domain

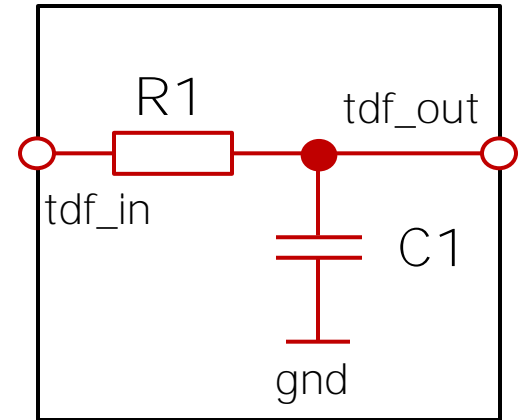
```
sca_tdf::sc_in < <type> >  
sca_tdf::sc_out < <type> >
```

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

```
sc_time sca_get_time()
```

Linear Electrical Networks

```
SC_MODULE(lp_filter_e1n)
{
  sca_tdf::sca_in<double> in;
  sca_tdf::sca_out<double> out;
  sca_e1n::sca_node in_node, out_node; // nodes
  sca_e1n::sca_node_ref gnd;          // reference
  sca_e1n::sca_r *r1;                  // resistor
  sca_e1n::sca_c *c1;                  // capacitor
  sca_e1n::sca_tdf2v *v_in;            // converter TDF->U
  sca_e1n::sca_v2tdf *v_out;          // converter U->TDF
  SC_CTOR(lp_filter_e1n) {
    v_in = new sca_e1n::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_e1n::sca_r("r1", 10e3);        // 10kOhm resistor
    r1->p(in); r1->n(out_node);
    c1 = new sca_e1n::sca_c("c1", 100e-6);      // 100uF capacitor
    c1->p(out_node); c1->n(gnd);
    v_out = new sca_e1n::sca_v2tdf("v_out", 1.0); // scale factor 1.0
    v_out->p(out_node); v_out->n(gnd); v_out->ctrl(out);
  }
};
```

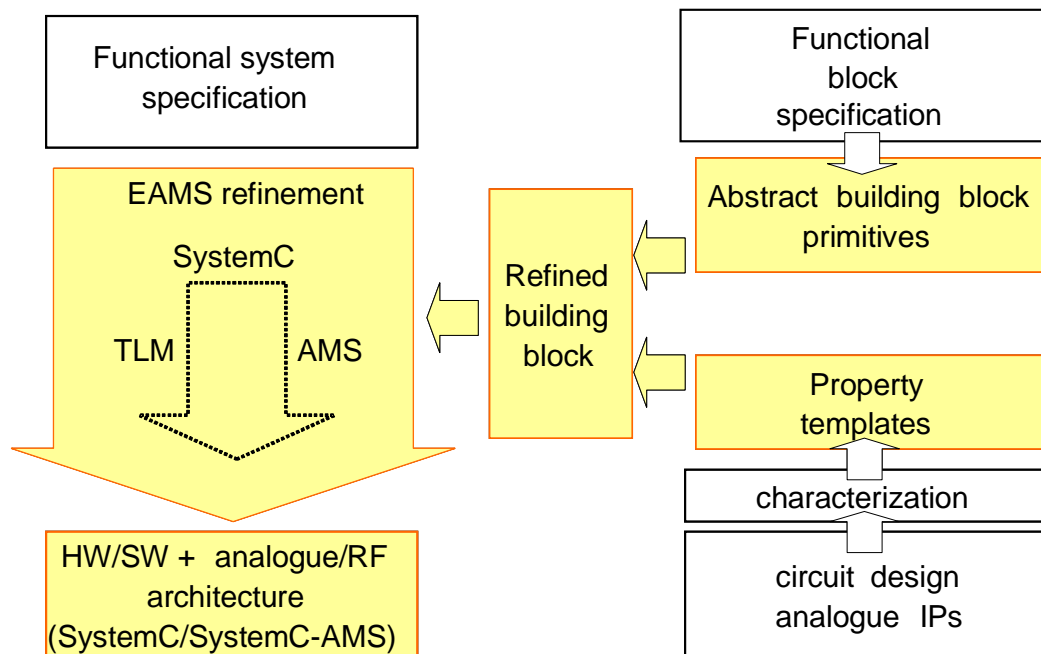


Wireless ultra-low power sensor networks

- Introduction
- Design challenges and architecture gap
- OSCI SystemC AMS extensions
- **Support for refinement of sensor networks**
 1. **Methodology**
 2. **Methodology specific support**

„Design Refinement“ methodology

- **Remember the challenge:** Design at architecture level, considering abstract overall-system (e.g. network protocols, ...) and at the same time circuit level properties in some parts in a top-down methodology.



„Design Refinement“ methodology

Design Refinement is a top-down methodology that successively augments and integrates properties of an implementation into a functional model, and analyzes their impact.

- Related, but not the same:
 - Extreme programming (SW engineering method)
 - Property refinement (Formal method, maintains safety properties)

Design refinement - classification

- Functional level (starting point)
- Architecture level
 - *Refinement of computation -> Computation accurate model*
Algorithms, data types, physical effects/accuracy
 - *Refinement of structure -> Structure accurate model*
Mapping to processors or functional blocks
 - *Refinement of interfaces -> Interface accurate model*
Signals, synchronization, bus protocols
- Implementation

Refinement of computation

Objective: Evaluate impact of non-ideal behavior of an assumed architecture to overall functionality (e.g. performance figures).

Method:

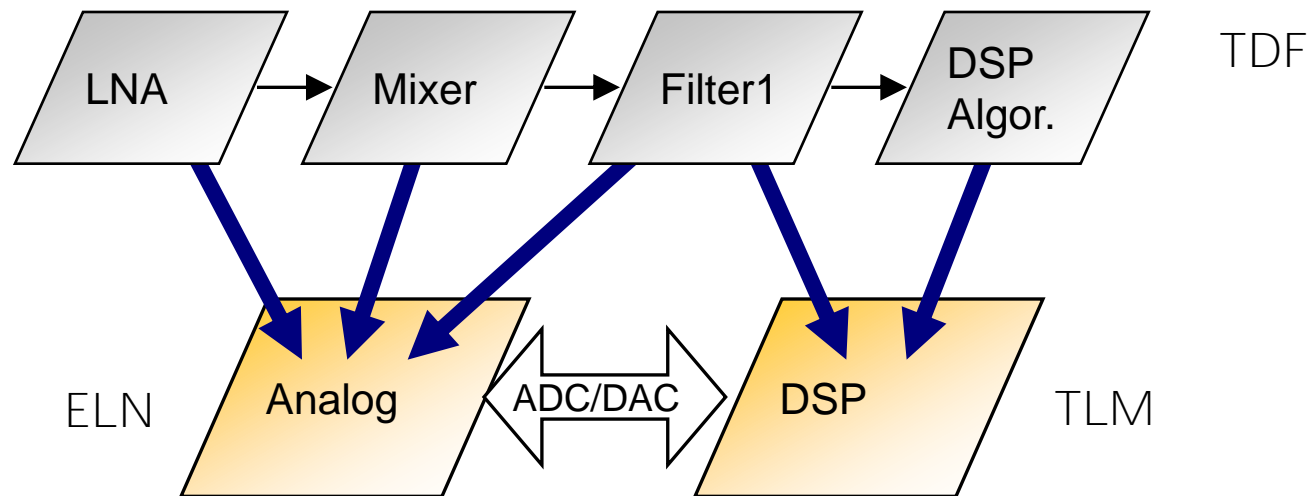
- Add non-ideal effects to executable, *functional spec*
- non-ideal behavior can be written directly in C-code!

```
void processing() // Mixer refined 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() );
}
```

Refinement of structure, re-partitioning

- **Objective:** Compare performance of different A/D/HW/SW partitionings more accurately.

Map functions of specification to processors, adapt structure of functional spec to architecture structure.



Refinement of structure, re-partitioning

- **Method 1:**

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)

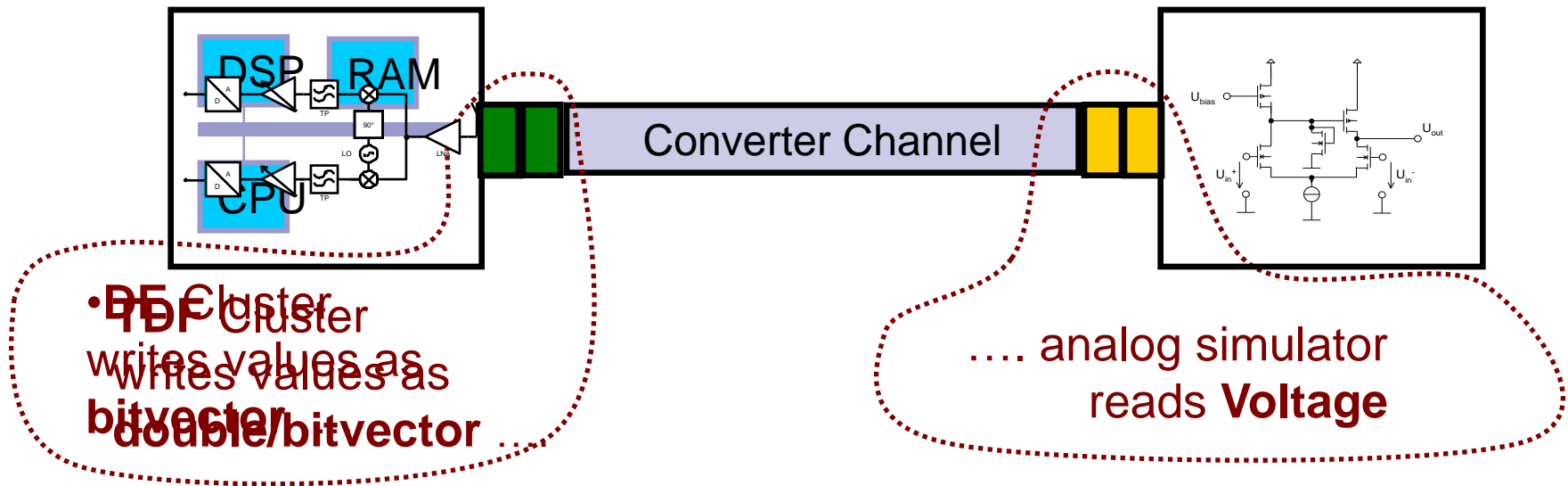
- *Slightly simplified by „static polymorphism“*

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

Refinement of structure, re-partitioning

■ Method 2:

Bottom-up integration of available models (e.g. library)



„**Converter channel**“ is hierarchical channel that separates modeling artefacts to translate MoC from „real“ design.

Interface accurate models

- **Intention:** Used for verification of system integration
All ports accurately as in implementation (same types, same number, clocks, enable-signals, ...)
- **Method 1:**
Use adapter classes that translate between abstract data flow and pin-accurate protocols or analog nodes
(requires appropriate models inside)
- **Method 2:**
Clock signals or events determine activation of TDF cluster
(allows combination of functional TDF description with complex control signals for validation of control signals)

Design methodology specific library

Application domain: Design refinement of *communication systems* and *sensor networks*.

Components:

- Signal sources
- Converters
- Mappers / Demappers
- Modulators / Demodulators
- Misc (e.g. FFT, ...)

- Analysis tools

... and more:

Support for design refinement

- Converter channels
- Adapter classes
- All modules highly configurable

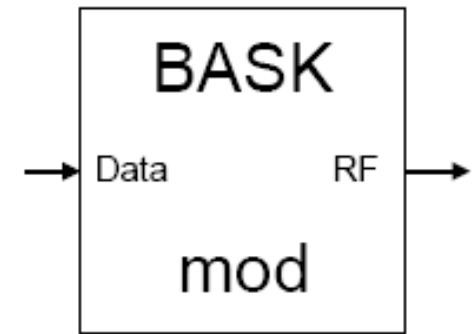
Component: BASK modulator

■ Binary Amplitude Shift Keying Modulator

Class definition:

```
andres_bb_mod_bask(sc_module_name nm,  
    double _freq,  
    double _ampl1,  
    double _ampl0,  
    double _phi,  
    int _rate) ;
```

Interfaces: sca_sdf_in<bool> in ;
sca_sdf_out<double> out ;



Component: IQ mapper

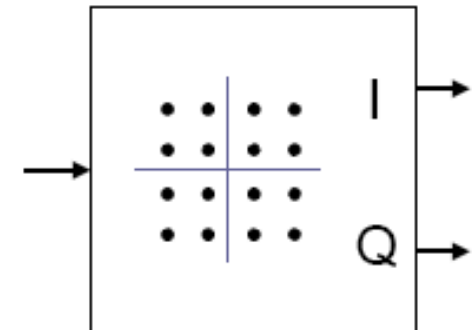
■ IQ mapper (4,16, 64, ...)

Class definition:

```
andres_bb_mod_iq(sc_module_name nm,  
    double _freq,  
    double _d_ampl,  
    double _d_phi,  
    int    _rate);
```

Interfaces:

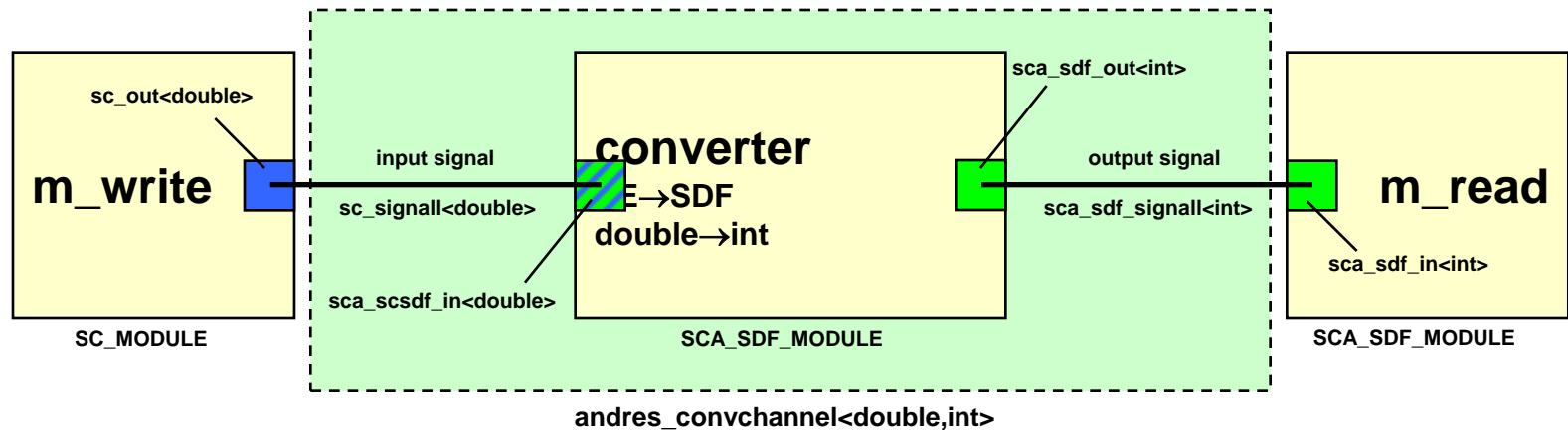
```
sca_sdf_in<double>  i  ;  
sca_sdf_in<double>  q  ;  
sca_sdf_out<double> out ;
```



Methodology specific support: converter channels

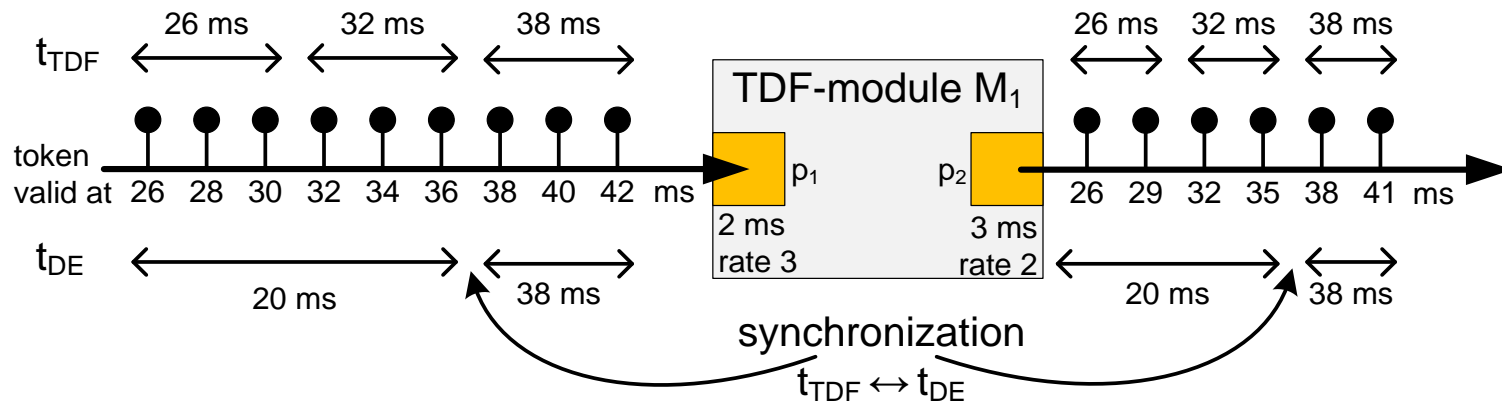
Converter channels specifically support the refinement of structure / re-partitioning.

- Converter channel adapts
 - MoC, data type, sample rates
 - after refinement steps

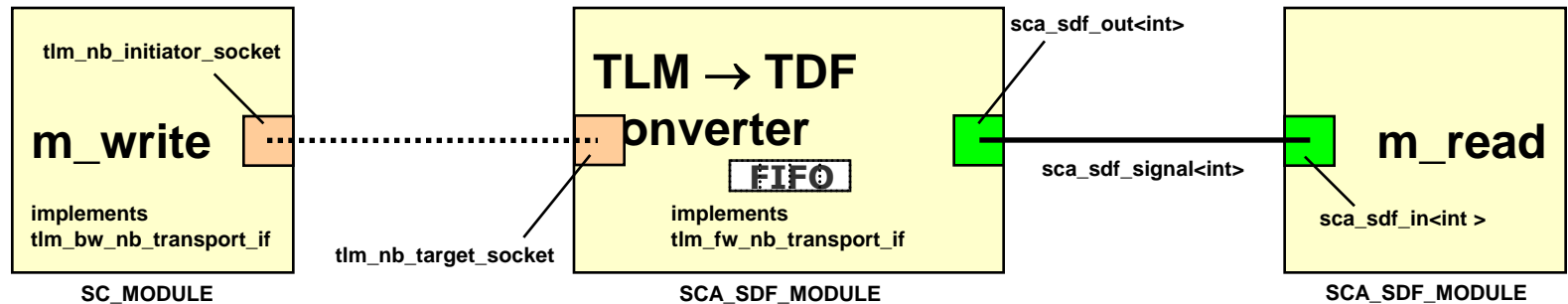


Coupling SystemC-AMS TDF and LT-TLM?

- SystemC-AMS is basically *strictly* timed, so connecting with *loosely* timed models seems futile...
- ... **but** TDF has also a kind of “time-warp”.
- A source process with datarate > 1 **also sends values “to the future”** with respect to the current SystemC (and even SystemC-AMS!) simulation time.
- A drain process with datarate > 1 , it **also receives “future values.”**
- SystemC-AMS time is always \geq SystemC time

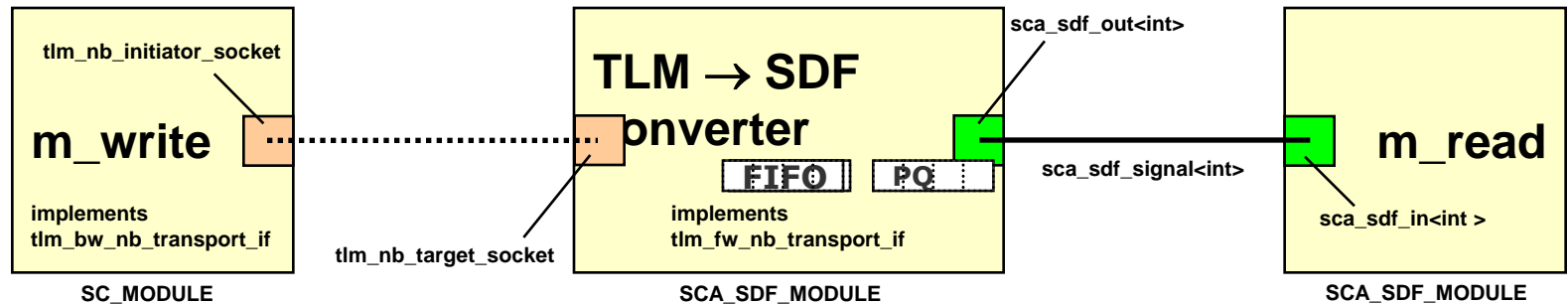


Conversion TLM → TDF



- **Basic idea:** Stream the data of the input transactions to a TDF signal with datarate r_t
- The transaction data are **buffered** within an internal FIFO
- At every `sig_proc()` execution, r_t many values are taken from the buffer and written to the TDF output port
- Empty buffer \Rightarrow send default values or throw error / warning
- If a transaction causes a buffer overflow we return the transaction with a `TLM_GENERIC_ERROR_RESPONSE`

Conversion TLM → TDF



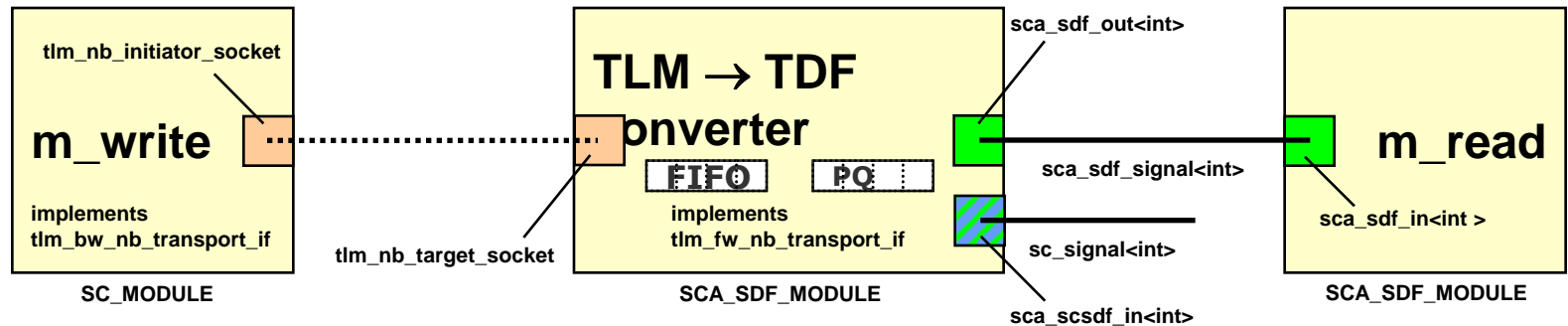
■ Problems:

- The **transactions may arrive** out of order with respect to their time stamps.

⇒ We use a **payload event queue** (PEQ) to store the transactions *before* writing their data to the buffer, which we only do when necessary (i.e. when the buffer has less data than the datarate).

- The PEQ sorts the transactions according to their time stamp.
- the data from a transaction is written into the buffer as late as possible.

Conversion TLM → TDF

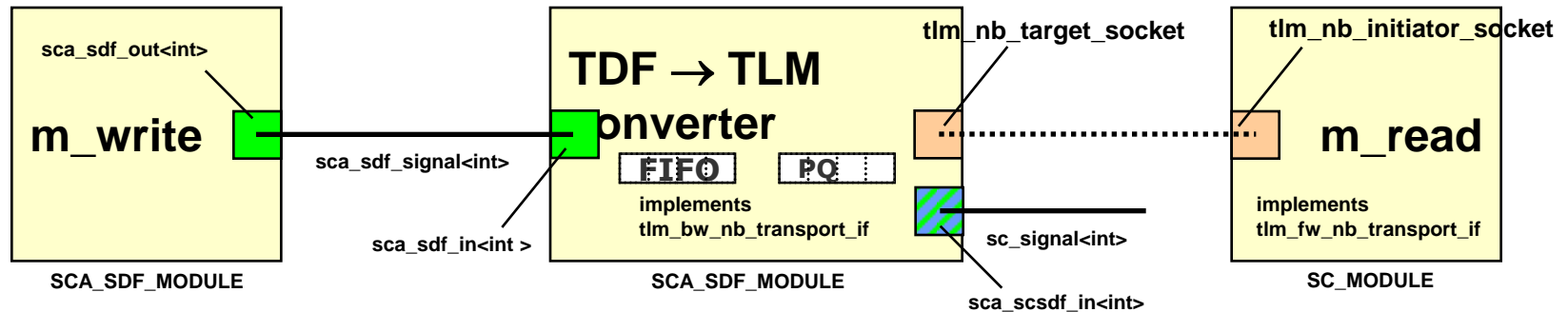


■ ...more problems:

- The **converter may run ahead** so far that the initiators might not had the chance to produce sufficient transactions to fill the buffer (even though they would).

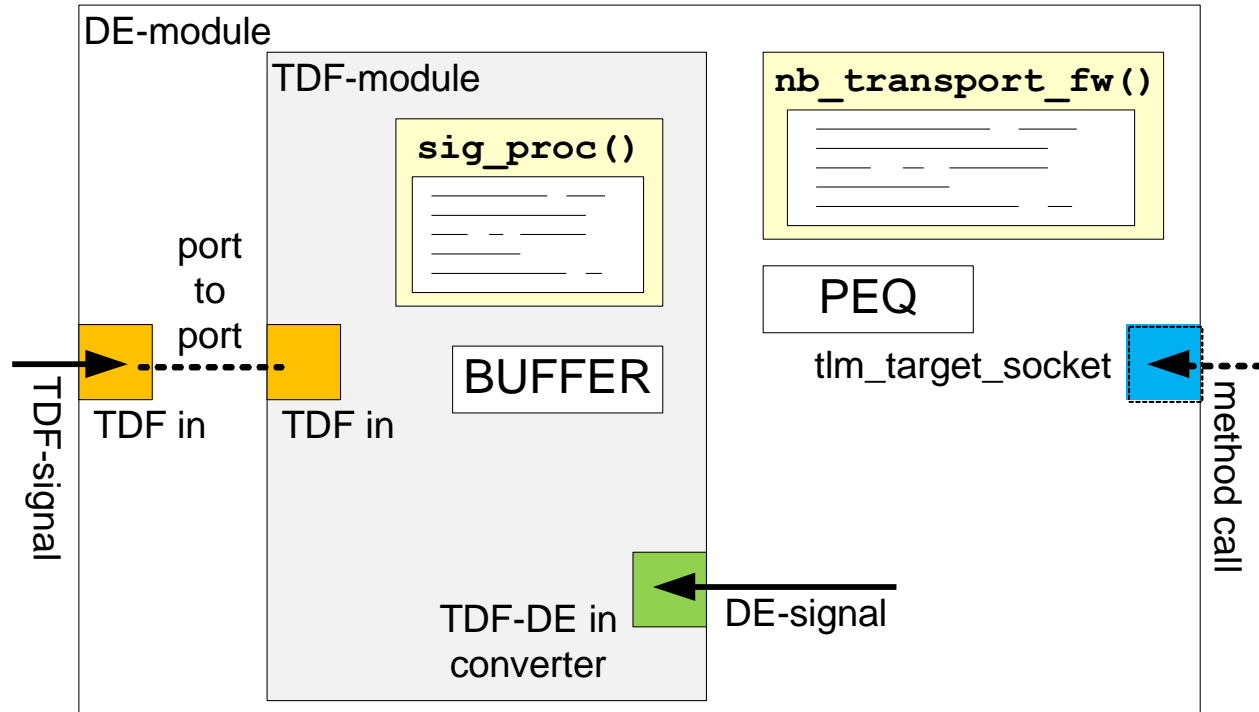
⇒ We have to trigger t_{DE} - t_{TDF} synchronization.

Conversion TDF → TLM



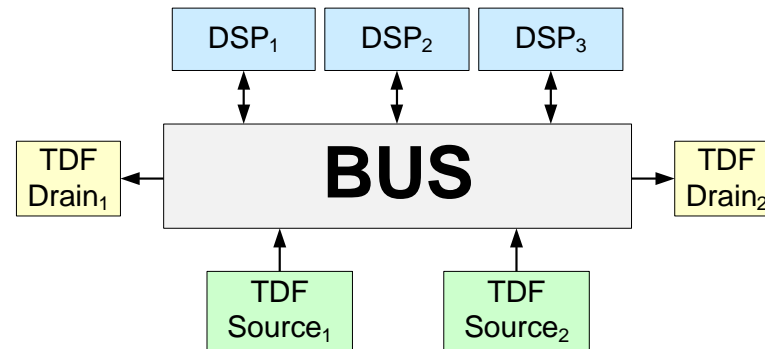
- **Basic idea:** Bundle the streaming data from the TDF side into transactions.
- The TLM side sends **read** transactions (requests) to the converter, who copies the requested data into the transaction and returns it.
- We use an internal buffer and PEQ again
- $t_{DE}-t_{TDF}$ synchronization needs due to full internal buffer. Access to internal converter port gives the TLM side a chance to produce sufficient **read** transactions.
- Still buffer overflow? \Rightarrow data loss or error/warning

TDF → TLM converter architecture



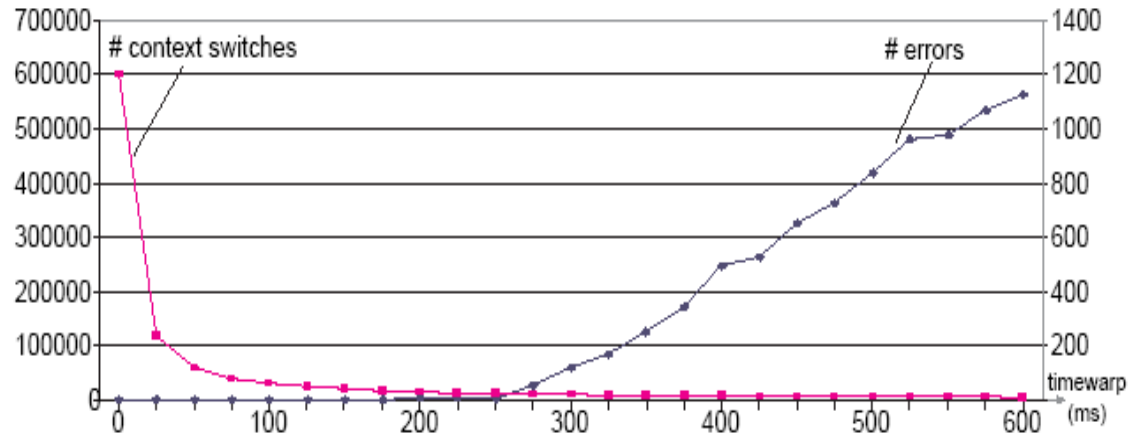
- Basically pretty similar to the TLM → TDF converter.

Experimental results



- Toy example: Three DSPs access two TDF drains and two TDF sources via a bus concurrently.
- If a DSP read from source 1 (source 2), it processes the data and writes the results to drain 1 (drain 2).
- Possible scenario: Software Defined Radio. Two different modulation schemes are used.
- Goal here: Testing functional correctness, observing simulation speed / simulation accuracy trade-offs.

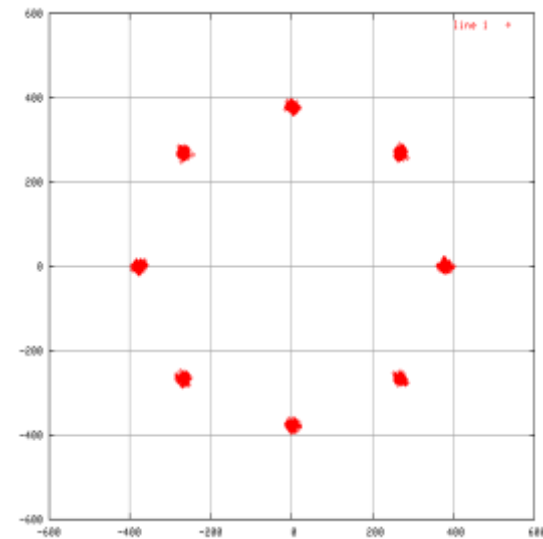
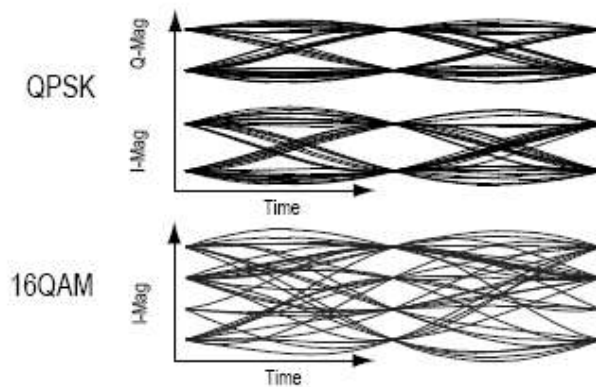
Experimental results



- We took the number of context switches (of the DSPs) as a measure for the simulation performance.
- The Arrival of an data packet in the wrong order constitutes as an error.
- About 12000 data packets were processed by the system

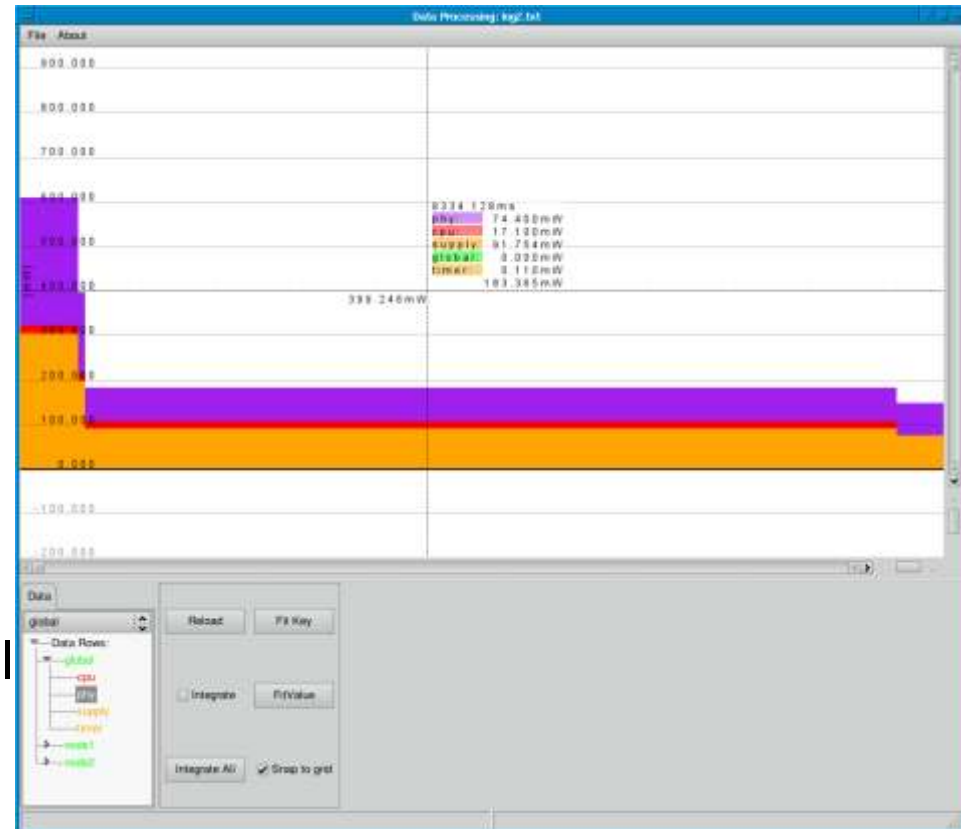
Analysis tools : debugging

- Tools to analyze signal quality by plotting
 - eye diagrams
 - trellis diagrams
 - constellation diagrams
 - ... as **SVG file**

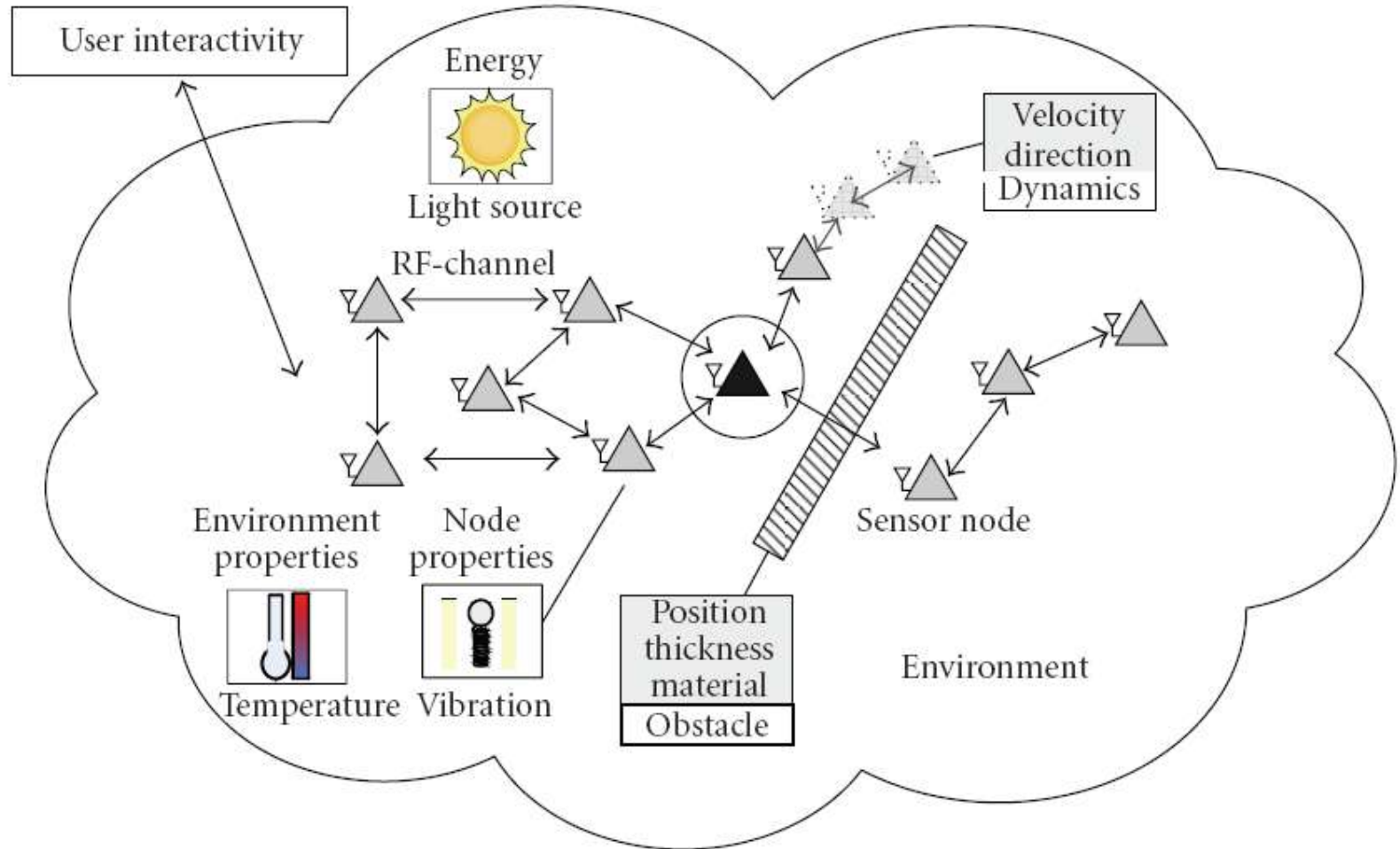


Analysis tools: Power metering

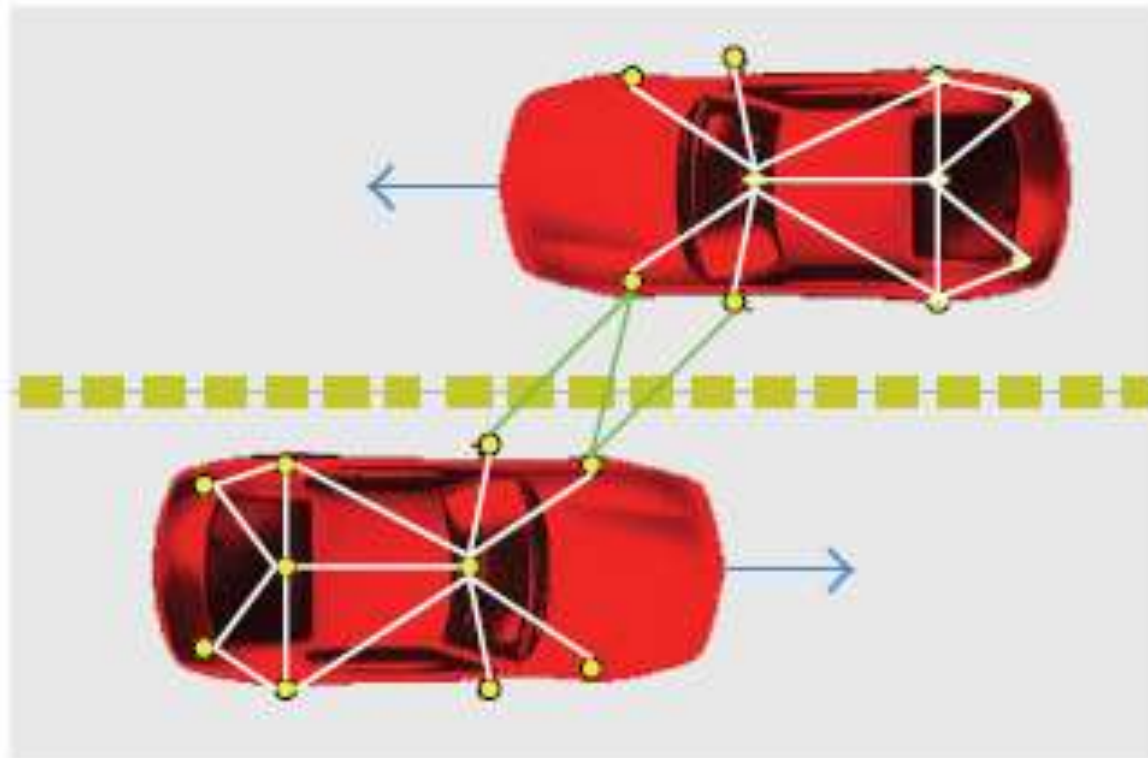
- Consumers
 - HW module usage
 - CPU usage
- Power consumption/use by
 - circuit-level simulation
 - measurement
 - data sheets
- Usage profile collected in log file
- Post Processing
 - Analysis
 - Visualization



Simulation scenario

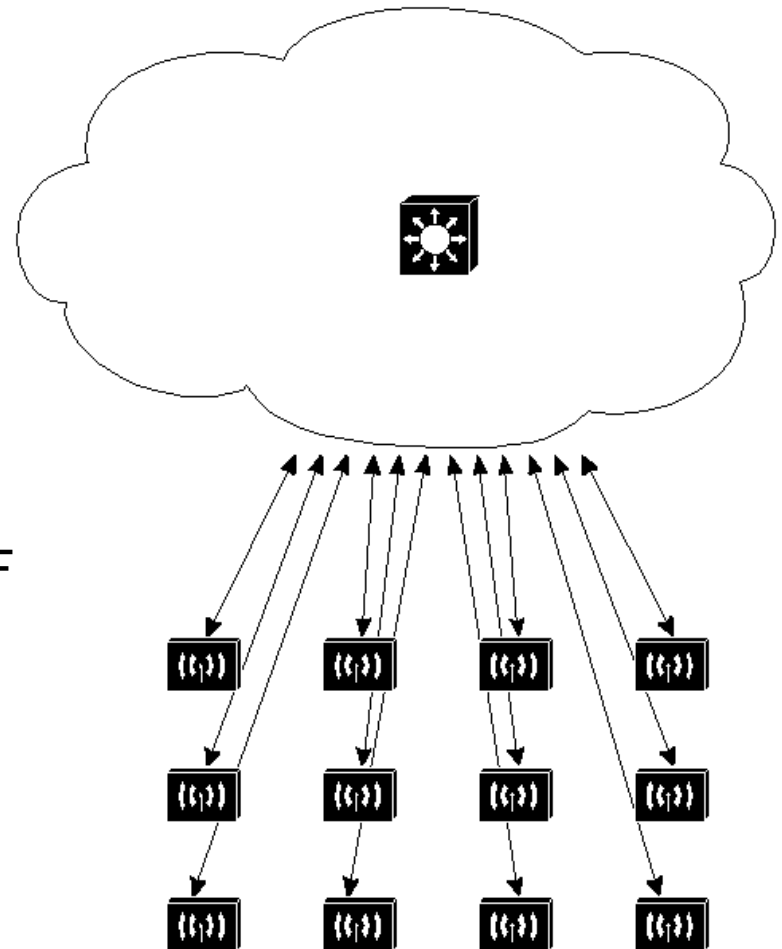


Dynamic scenarios



Library object „air“

- Global “Air” object
- Acts like a Switch
- Considers 3D arrangement, Obstacles
- Attenuation, distortion, noise
- Every node connects to it
- “RF Messages” are distributed
- Use BB equivalent instead of real RF
- *First version: PAWIS project, ongoing work in SNOPS project.*




Wireless ultra-low power sensor networks

- Introduction
- Design challenges and architecture gap
- OSCI SystemC AMS extensions
- Support for refinement of sensor networks

- SystemC AMS is gaining impact for architecture-level design of complex heterogeneous systems
- Well applicable for E-AMS systems, where HW/SW and AMS are functionally interwoven
- Design refinement modeling strategy integrates architecture properties successively into functional model
 - Quickly available first models
=> Early feedback on feasibility or potential issues
 - Immediate analysis/verification after changing/adding property
=> Optimization / debugging more efficient

Some references

- www.systemc.org, AMS WG (OSCI members)
- www.systemc-ams.org (For information from former SystemC-AMS SG, provides some information for the public)
- *Ch. Grimm, M. Barnasconi, A. Vachoux, K. Einwich: An Introduction to Modeling Embedded Analog/Mixed-Signal Systems using SystemC AMS Extensions.* OSCI, June 2008
- *Ch. Grimm: Modeling and Refinement of Mixed Signal Systems with SystemC.* In: *SystemC: Methodologies and Applications.* Kluwer Academic Publisher (KAP), June 2003.
- *Johann Glaser, Daniel Weber, Sajjad A.Madani, and Stefan Mahlkecht: Power Aware Simulation Framework for Wireless Sensor Networks and Nodes.* In: Ch. Grimm (editor) EURASIP Journal on Embedded Systems; Special issue on C-Based Design of Heterogeneous Systems; vol. 2008, Article ID 369178.



Thank you for your attention!

Your:

- questions
- comments
- ideas
- objections