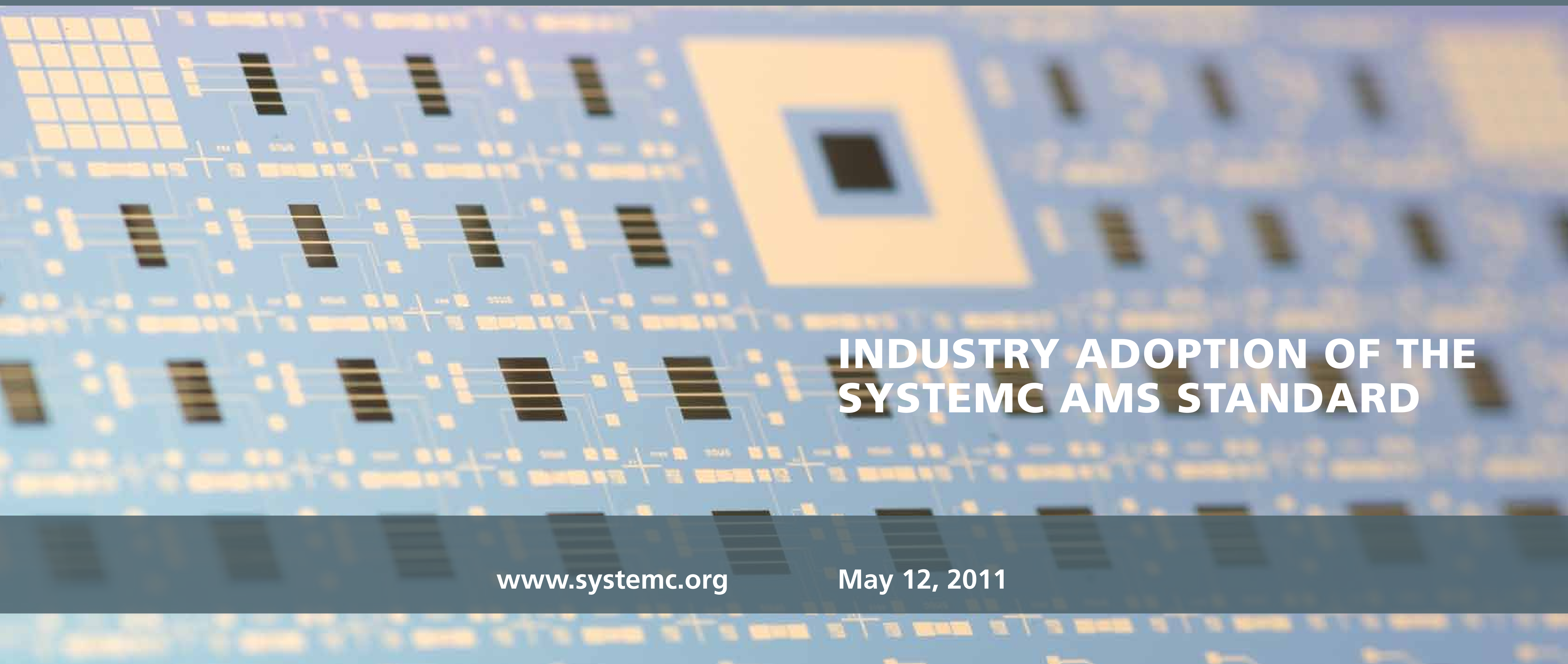
OPEN SYSTEMC INITIATIVE

SYSTEMC AMS DAY 2011



INDUSTRY ADOPTION OF THE SYSTEMC AMS STANDARD

www.systemc.org

May 12, 2011

OPEN SYSTEMC
INITIATIVE

SystemC AMS Day 2011

Industry Adoption of the SystemC AMS Standard

May 12, 2011

Dresden

The SystemC AMS Day is sponsored by:

ARM, Cadence, Forte Design Systems, Mentor Graphics, Synopsys, XtremeEDA

Content

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Institut für Computertechnik
Institute of Computer Technology

A Range Based System Simulation and Refinement Design Flow

TU Vienna, Chair of Embedded Systems

F. Schupfer, M. Svarc, C. Radojicic, C. Grimm

Institut für Computer Technik

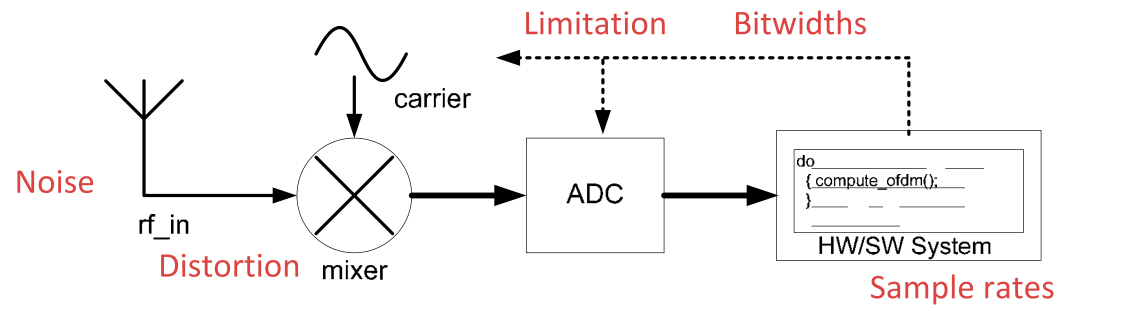
Overview

- Introduction
- Refinement methodology
- MARC/SYCYPHOS Design environment, examples
- Future work

Embedded Mixed-Signal Systems

Embedding „Cyber" and „physical" systems

Application SW Stack



- *Verification of system functionality and parameters (accuracy, power) requires system simulation over long time period*
- *Problems/Errors often detected too late, during/after design*

4 Major Problems in Verification of AMS Systems

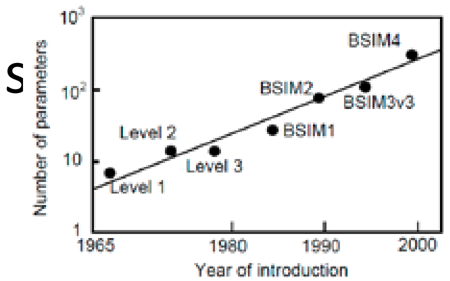


Fig. 2. The increase of MOSFET model parameters

1. Incomplete specification
2. Models too abstract: Power, accuracy, … ?
3. Process variations
4. Insufficient verification coverage, system integration

More accurate models … (2., 3.)

More abstract models needed for more simulation runs … (3., 4.)

State of the Art, Related Work

Mainstream tries to use RT/circuit level models and simulation

- Mixed-Level, Multi-Run, Monte-Carlo, etc.
- Design of Experiments [Rafaila]
- **Earlier estimation of power + accuracy needed!**

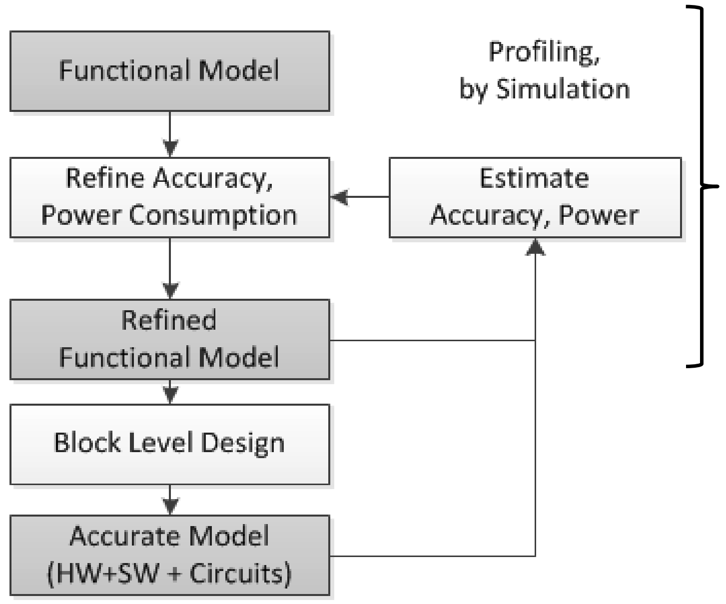
SystemC AMS Methodology enables modeling and simulation of embedded mixed-signal systems at functional, architecture level.

- Power consumption?
- Accuracy?

Overview

- Introduction
- Refinement methodology
- MARC/SYCYPHOS Design environment, examples
- Future work

Profiling/Refinement at Functional Level

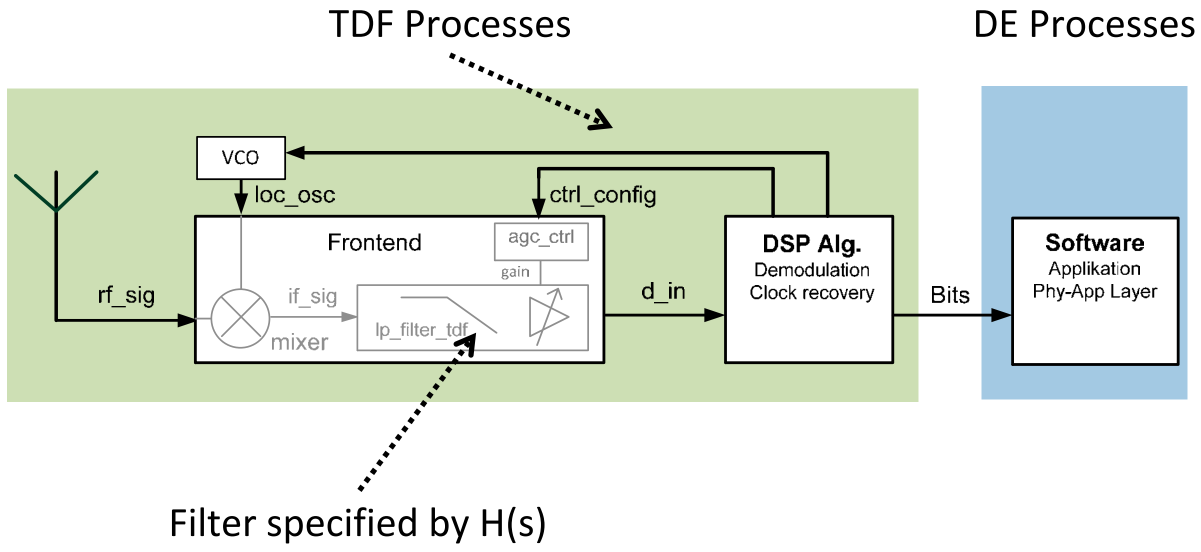


Supported by tools and libraries based on SystemC AMS and TLM extensions:

***Why** is accuracy reduced? Risks?*
=> Accuracy budgeting

***Why** is power consumed?*
=> Power budgeting

Start: Functional Model



TDF: Filter in Receiver

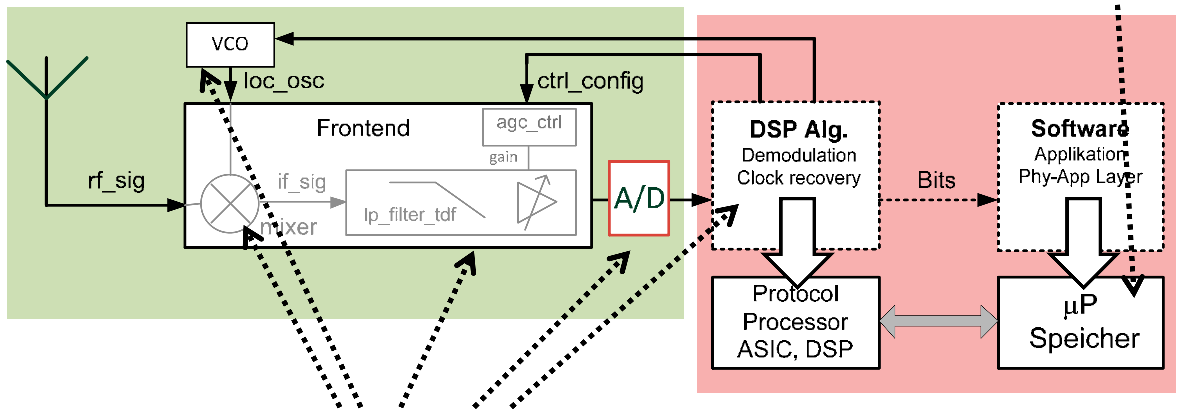

```
SCA_TDF_MODULE(lp_filter_tdf)
{
  sca_tdf::sca_in<double>  in;
  sca_tdf::sca_out<double> out;
  sca_tdf::sca_in<double>   gain;
  sca_tdf::sca_ltf_nd ltf;        // computes transfer function
  sca_util::sca_vector<double> num, den; // coefficients

  void initialize()
  {
    num(0) = 1.0;
    den(0) = 1.0; den(1) = 1.0/(2.0*M_PI*1.0e4);
  }
  void processing()
  {
    out.write( ltf(num, den, in.read() * gain.read() ) );
  }
  SCA_CTOR(lp_filter_tdf) {}
};
```


Refined functional model

TDF Model

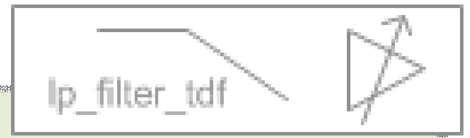
Transaction Level Model
Instruction Set Simulator (**ISS**)



Model of non-ideal properties in frontends, ADC, DSP

Model HW/SW Partitioning, Allocation

TDF: Filter in Receiver




```
SCA_TDF_MODULE(lp_filter_tdf)
{
  sca_tdf::sca_in<AAF>  in;
  sca_tdf::sca_out<AAF> out;
  sca_tdf::sca_in<AAF>   gain;
  sca_tdf::sca_ltf_nd ltf;        // computes transfer function
  sca_util::sca_vector<double> num, den; // coefficients

  void initialize()
  {
    num(0) = 1.0;
    den(0) = 1.0; den(1) = 1.0/(2.0*M_PI*1.0e4);
  }
  void processing()
  {
    out.write( ltf(num, den, in.read() * gain.read() ) + noise() );
  }
  SCA_CTOR(lp_filter_tdf) {}
};
```

From budgeting

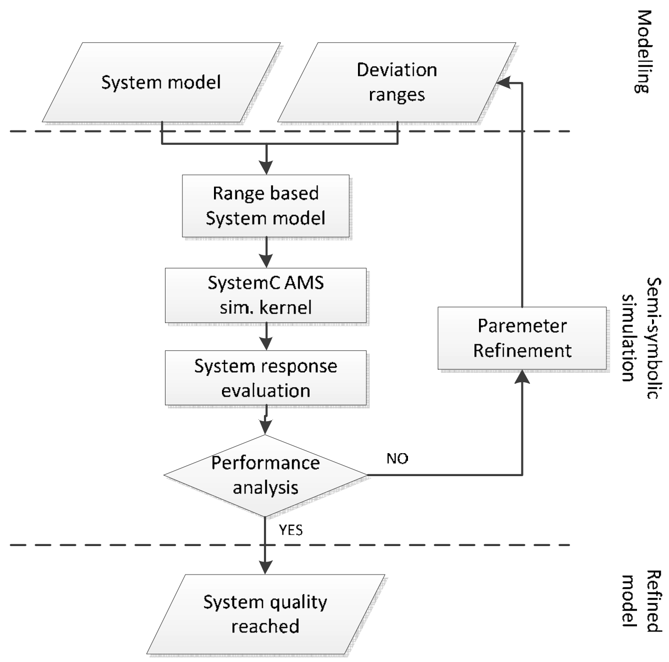
From characterization

Accuracy Profiling and Iterative Refinement

Range based model

Semi-symbolic simulation with SystemC AMS

Refined system model



SystemC (TLM, AMS) based analysis

- SystemC – based tracing of power and accuracy

„Body“	AAF extension	(or Power ext.)	(or Air ext.)
Sample (or package)	Partial deviations	Accumulated Power	Routing history
			Trace through network

Profiling Accuracy

Power Profiling

Tracing messages through HW and. Network

Range-Based Simulation

Affine Arithmetic – general

- Enhancement of Intervall Arithmetic [Comba]
- Accurate range-based computations for linear systems

Affine Arithmetic - simulation

- Static and dynamic deviations [Heupke, Grimm]
- SystemC AMS integration [Heupke, Grimm]
- Transistor level solver [Grabowski, Grimm]

Affine Arithmetic [Andrade et al.]

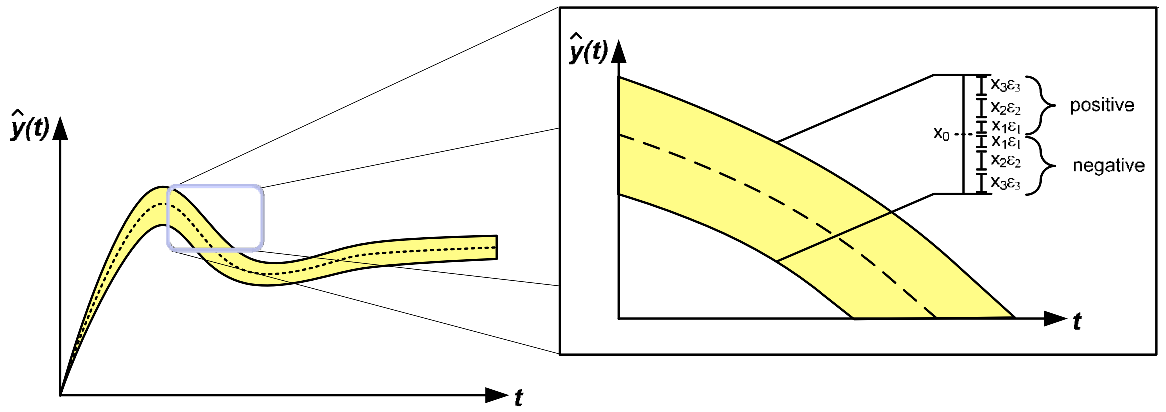
Improves Interval Arithmetics by conserving *correlations* in a symbolic way

Affine Arithmetics represents a size $\hat{x}$ by

- an ideal, numerical 'central value' x_0, and
- n partial deviations x_i scaled by noise symbols $\epsilon_i \in [-1, 1]$

$$\hat{x} = x_0 + \sum_{i=1}^{n} x_i \epsilon_i$$

Graphical representations

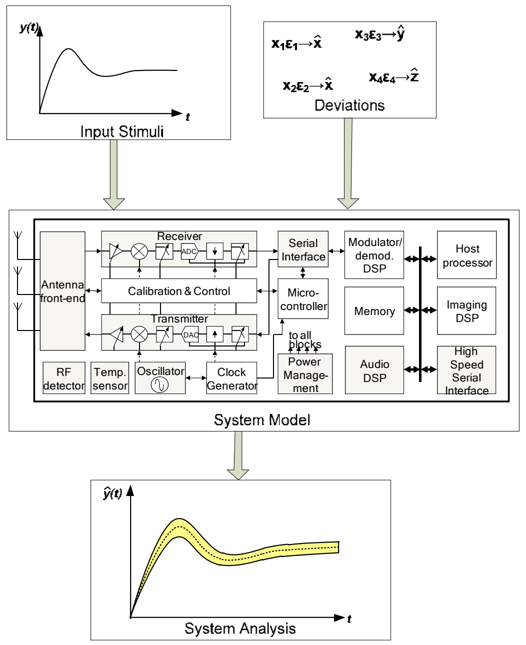


Range based system response

Signal construction by sub-ranges

Semi-symbolic Simulation

- Numeric simulation extended by symbolic representatives
- Multiple simulation results by one run
- Range dependency preservation
- Guaranteed, conservative result inclusion



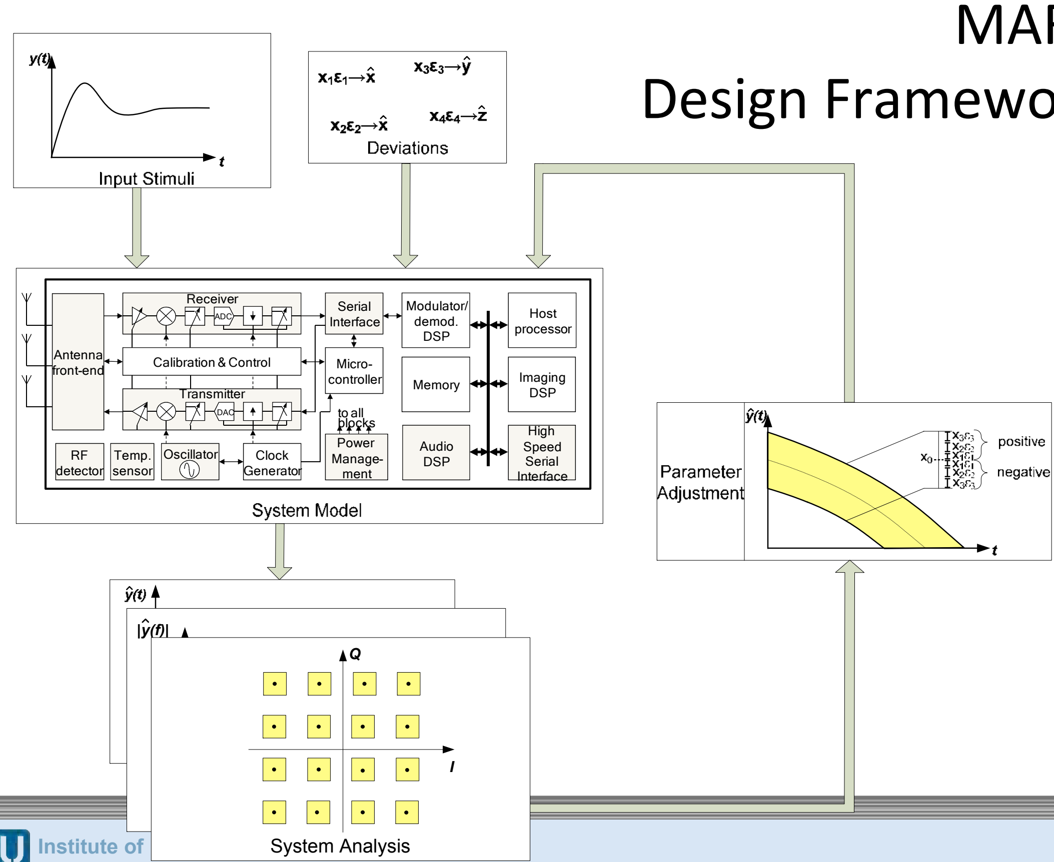
Overview

- Introduction
- Refinement methodology
- **MARC/SYCYPHOS Design environment, examples**
- Future work

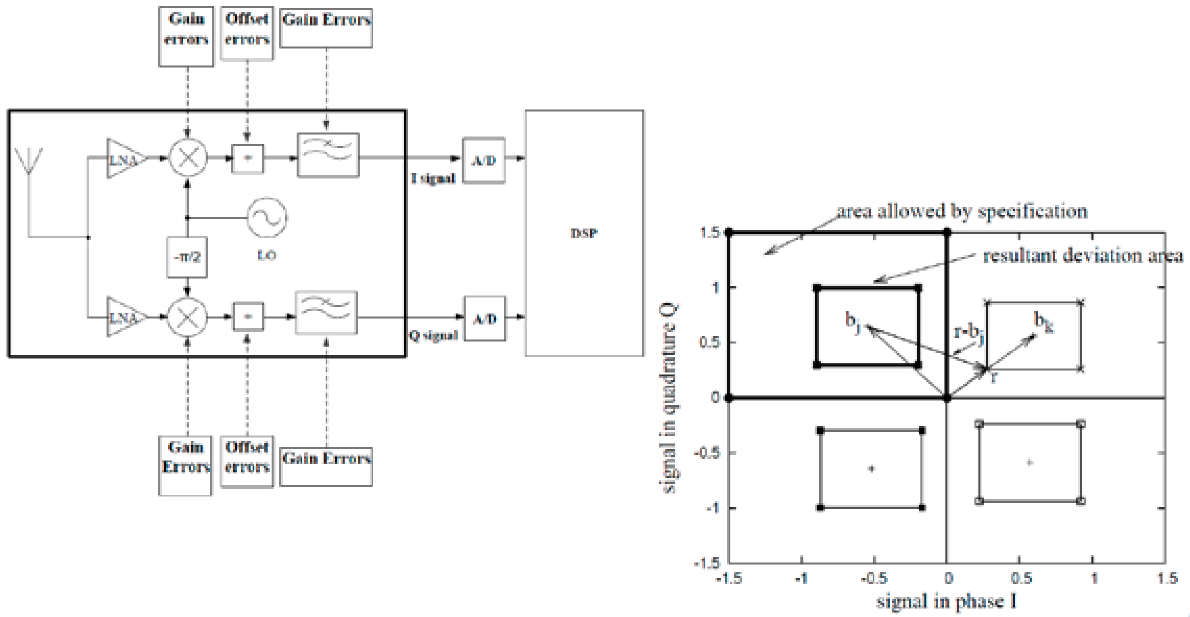
Libraries & Tools

1. Library of functional blocks
   - Blocks for receiver/transmitter (serializer, modulators, mixers, ACD, …)
   - Non-ideal properties (Noise, offset, nonlinearities, …)
   - Models von processors (ISS)
2. Profiling tools
   - **Accuracy profiling**
   - Power (see poster)

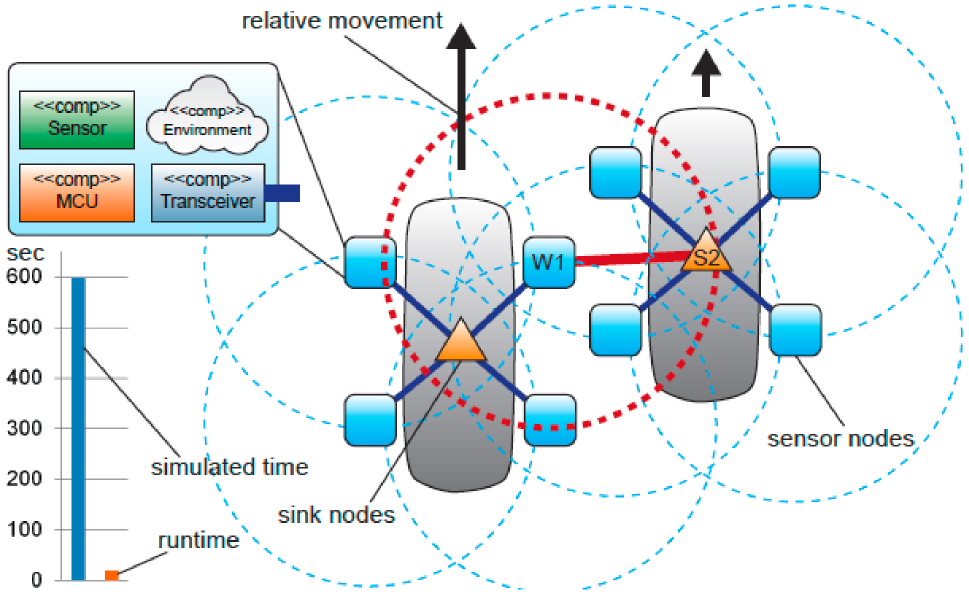
MARC Design Framework



Example: Profiling Accuracy in 4-QAM receiver

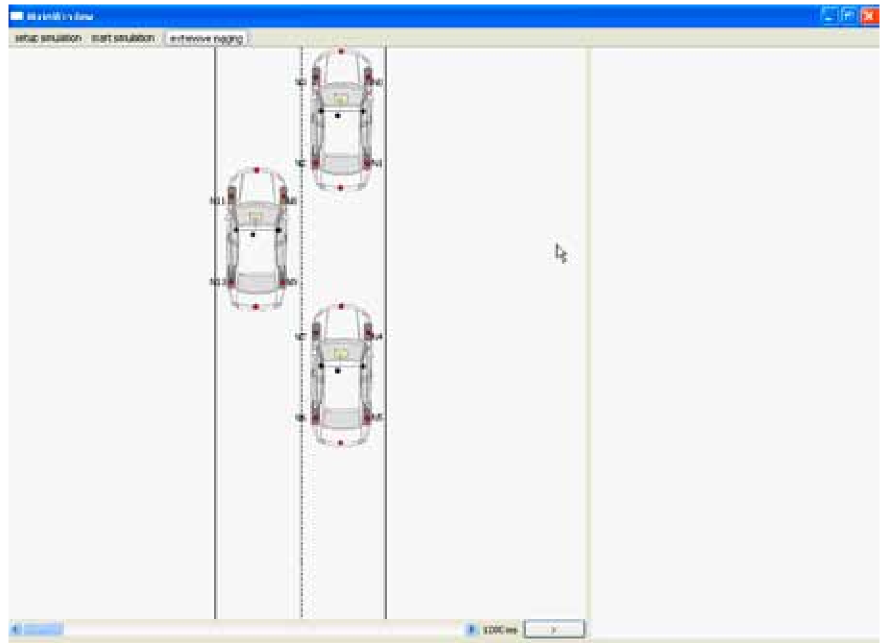


Poster: Power Profiling of In-Car WSN;
18 8-Bit uC with Firmware + Transceiver + Sensors



www.ict.tuwien.ac.at

Poster: Power Profiling of In-Car WSN;
18 8-Bit uC with Firmware + Transceiver + Sensors



www.ict.tuwien.ac.at

Overview

- Introduction
- Refinement methodology
- MARC/SYCYPHOS Design environment, examples
- **Conclusion, Future work**

www.ict.tuwien.ac.at

Conclusion, Outlook

- Range based refinement methodology
  - Complements Worst-Case Analysis
  - Single run, traceable deviations influence
  - Refinement information = recommendations, maybe automation?
- Planned extensions
  - Automated management of ressource "accuracy"
  - "Expert-models" that include typical risks as kind of IP-Knowledge from recent projects

Future work: SYCYPHOS/MARC

- Synthesis of Cyber Physical Systems and Applications *integrates* all TUV Tools:
  - Modeling of scenarios and high-level communication in cyber and physical worlds
  - Modeling of accuracy, robustness, power consumption in microelectronic systems
  - ***Challenge:***
    Automatical analysis, verification, and improvement of accuracy, resilience/adaptivity, power consumption

Thank you for your attention

Affine Arithmetic: System Simulation

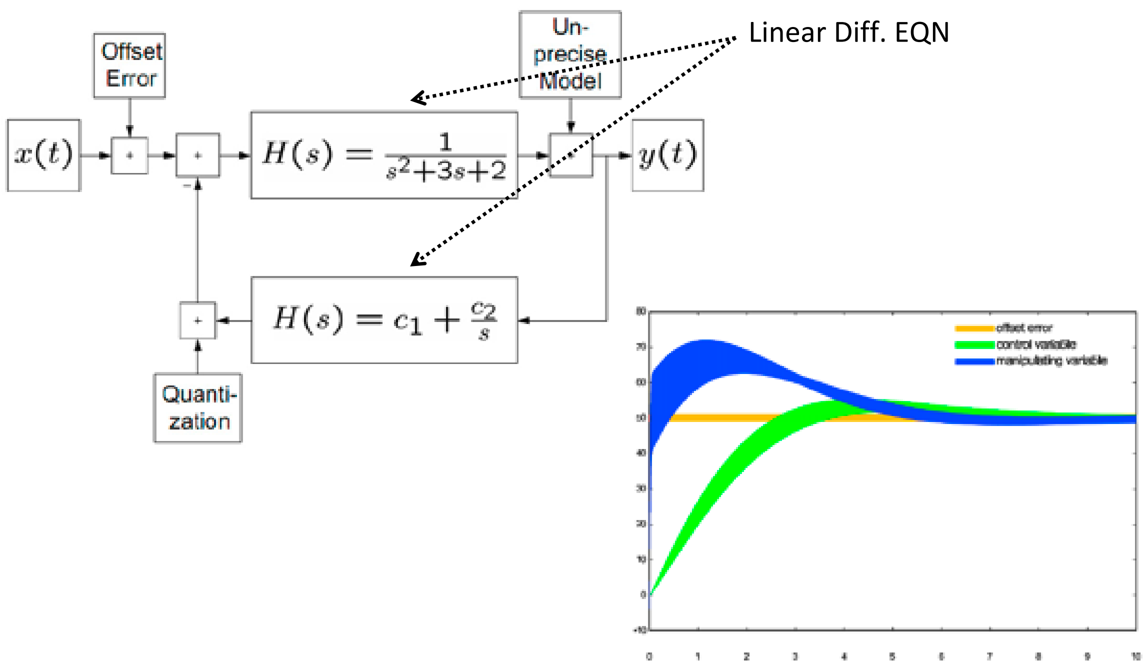
System Simulation, SystemC AMS

- Directed signal flow; *output = f(input, state)*
- Models of Computation: Synchronous & Dynamic Data flow, KPN, Discrete event modeling, Signal flow

System Simulation with AA straight forward:

- Class library provides abstract data type AAF and associated linear and nonlinear operations
- Number of noise terms increases with each nonlinear operations → „Garbage collection“

Affine Arithmetic: System Simulation



Circuit Simulation with Affine Arithmetic

Component models | Netlist | Parameters p

Modified Nodal Analysis (MNA)

$\underline{F}(\underline{x}, \dot{\underline{x}}, \underline{p}, t) = \underline{0}$

Numerical Integration

$$LTE = \left|\frac{\delta t}{2}\ddot{x}\right|$$

$\underline{F}(\underline{x}, \underline{p}, t) = \underline{0}$

Symbolic Differentiation

$$\mathrm{J} = \frac{\partial \underline{F}(\underline{x}, \underline{p}, t)}{\partial \underline{x}} \qquad \mathrm{P} = \frac{\partial \underline{F}(\underline{x}, \underline{p}, t)}{\partial \underline{p}}$$

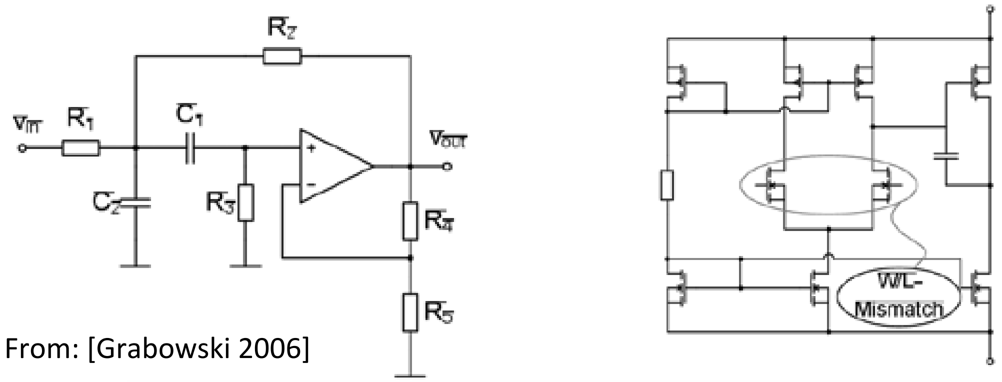
Computation of Affine ASPs

Computaton of Affine ASP as follows:

1. Compute x_0 by existing Newton-Raphson iteration:
   $$\underline{F}(\underline{x}_0, \underline{p}_0, t) = \underline{0} \quad \rightarrow \quad \hat{\underline{x}} = \underline{x}_0$$
2. Compute $x_i\epsilon_i$ by sensitivity analysis:
   $$\mathrm{J}|_{\underline{x}_0,\underline{p}_0}\Delta\underline{x} + \mathrm{P}|_{\underline{x}_0,\underline{p}_0}\Delta\underline{p} = \underline{0} \quad \rightarrow \quad \hat{\underline{x}} = \underline{x}_0 + \sum \underline{x}_p\epsilon_p$$
3. Compute NL ϵ_{i+1} (in n-dim space) by approximation scheme in vector/matrix form (Grabowski 2006, 2007, 2008).

$$\hat{\underline{x}} = \underline{x}_0 + \sum \underline{x}_p\epsilon_p + \underline{x}_{epd}\epsilon_{epd,i}$$

Affine Arithmetic, Circuit Simulation



From: [Grabowski 2006]

AAF:
5 sec.
50 MC runs:
50 sec.