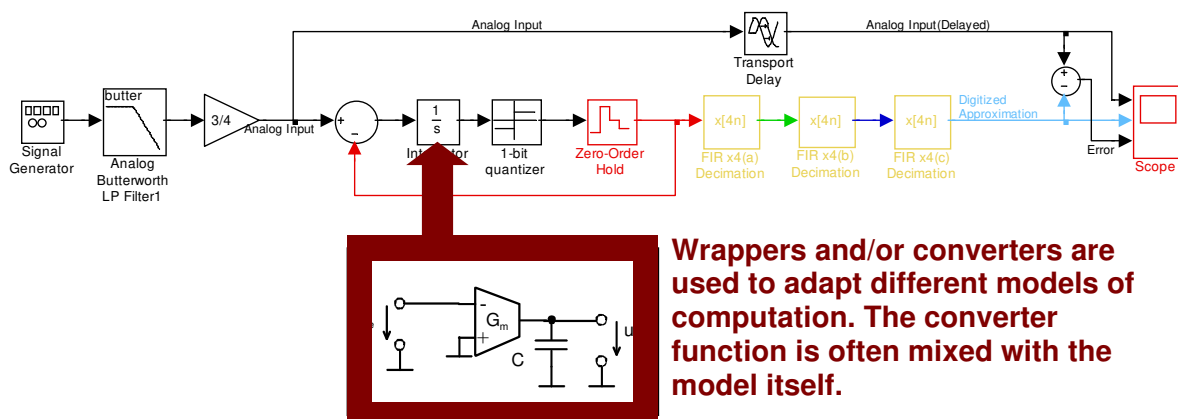


# Top-down Refinement of Mixed-Signal Systems with converter modules

Christoph Grimm, Markus Damm, Jan Haase, Florian Brame  
TU Vienna, Institute of Computer Technology

SystemC-AMS provides the designer with a tool that supports the system- and architecture level modelling and simulation of RF-, analog and mixed-signal systems. However, especially for analog components design often still needs large bottom up fractions to allow designers an early performance estimation of architectures. A known solution for this issue is mixed-level simulation. In mixed-level simulation, some parts of a model are modelled in an abstract way for example by a transfer function, others in a very concrete way, e.g. by a net list. Mixing of different levels of abstraction leads models that combine different models of computation (MoC). There are several ways to deal with the necessary conversion between MoC:

1. Manual approach like in VHDL there is no support for conversion between analog and digital MoC. A designer has to write code (converter, ...) to make a conversion. This is error prone and time consuming.
2. Automatic approach as in Verilog-AMS, where converter modules are automatically inserted between different domains, if the designer provides them, and wishes the simulator to do this.
3. Manual and structured approach as in TLM/SystemC, where transactors translate between different levels of abstraction of communication.



In the last years we have implemented a first demonstrator similar in function to the second and/or third approach. It supports the translation between different MoC and even allows the integration of external simulators. Actually, Dolphin/Smash (VHDL, SPICE) and Matlab/Simulink are supported. The demonstrator library specifically supports a design methodology that successively refines an executable specification to a mixed-signal circuit by adding properties and structures while being able to simulate each single design refinement. The presentation gives an overview of different approaches to handle the coupling of different MoC, describes the current implementation and also gives an overview of the design methodology.



# Top-down Refinement of Mixed-Signal Systems with Converter Channels

Institute of Computer Technology  
Embedded Systems Group

Institut für  
Computertechnik

**ICT**

Institute of  
Computer Technology



Christoph Grimm

**Markus Damm**

Jan Haase

Florian Brame

- **AN**alysis and **D**esign of run-time **RE**configurable heterogeneous **S**ystems
- EU-founded project
- Duration: June 2006 – May 2009
- Volume: 31 person years
- <http://andres.offis.de>

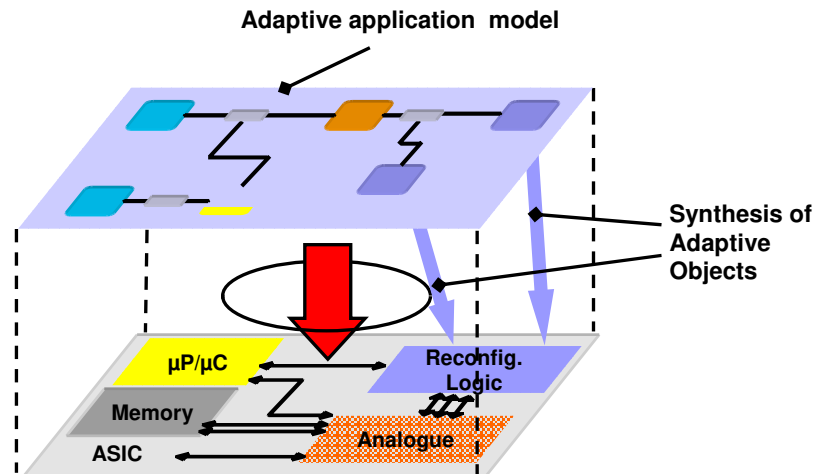


### Motivation:

- (Self-)Adaptivity is gaining importance for the development of flexible and efficient systems
- Systems are heterogeneous in nature
- Adaptivity has to be considered in different domains: digital hardware, analogue hardware, and software

### Goals:

- Theoretical framework to specify Adaptive Heterogeneous Embedded Systems (AHES)
- Modelling of adaptive systems using domain specific Models of Computation in one integrated framework based on SystemC
- Automatic synthesis of components and interfaces (digital hardware and software)



### Three SystemC-extensions:

- **SystemC-AMS:** Signal processing & Analogue
- **HetSC:** Software
- **OSSS+R:** Hardware Synthesis

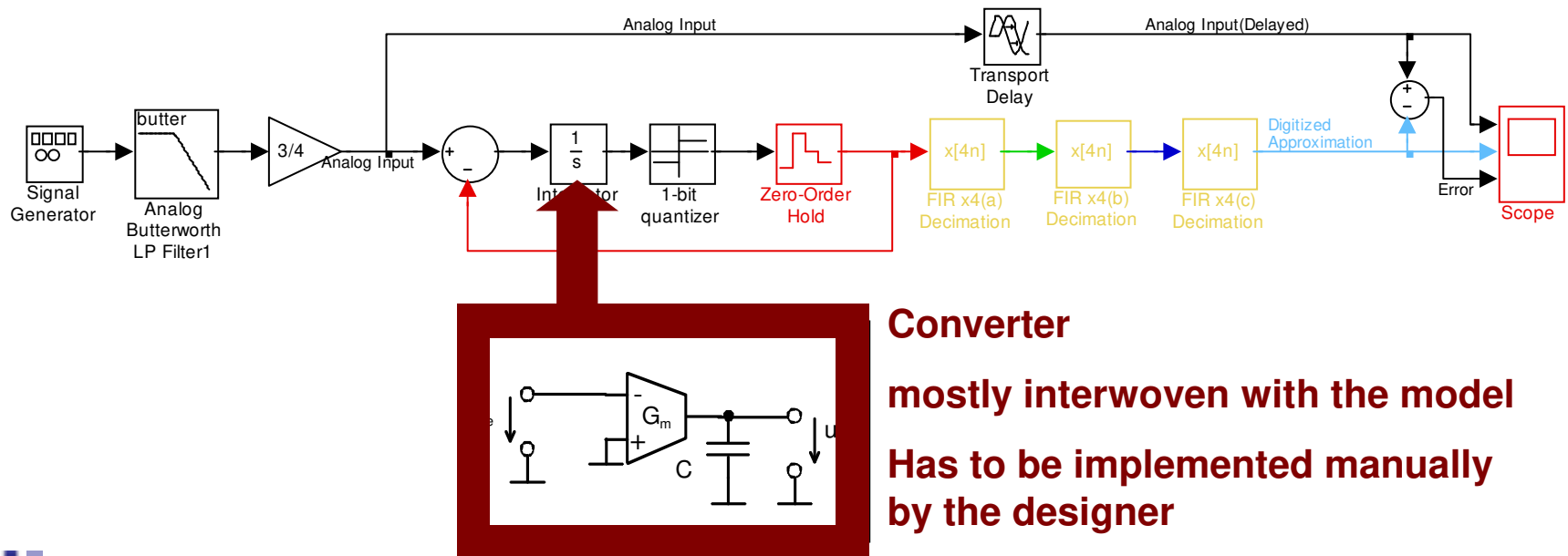
Theoretical Backbone: **ForSyDe**

# Benefits of top-down design

- Productivity-gain of 10 and more...  
[Collet Intl. , DAC '98; Numetrics]
- Highest productivity
  - 3-6 month for a mixed-signal project
  - Almost no redesigns
- Reasons for lower productivity
  - Specification errors
  - Bottom-up approach, late simulation
- But: analogue systems "need bottom up"
  - Mixed level simulation integrates "local bottom up" into an overall top-down approach

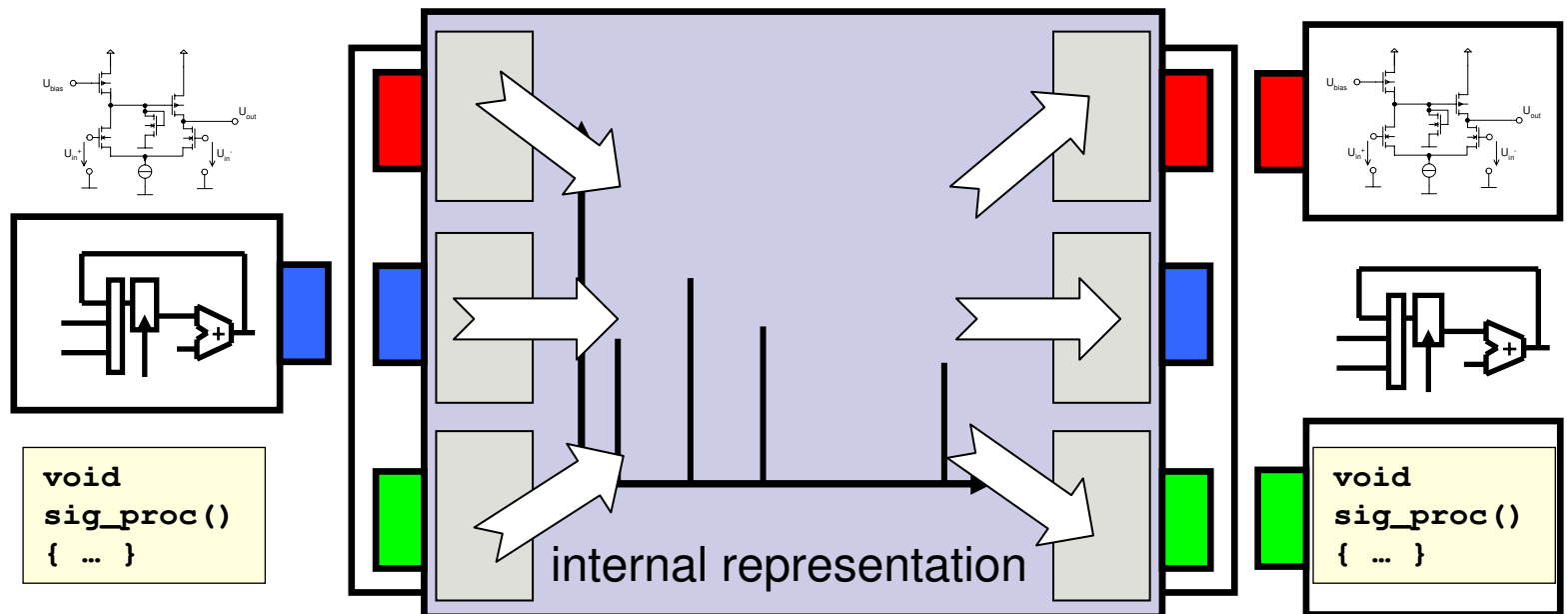
# Mixed level Simulation

- Refinement of single system components
- Simulation together with the "higher level" parts
- Benefit: Simulation time & accuracy
- Problem: Converters are needed



# Polymorphic signals

- Enable the communication between components using different MoCs / Interfaces
- Grimm/Schroll/Waldschmidt, Kluwer 2005;  
Schroll/Grimm/Waldschmidt, FDL 2004
- Prototype: University of Frankfurt/Main (based on SystemC-AMS)



# Converter channels within ANDRES

- New implementation of polymorphic signals
  - Different approach (e.g. no general inner representation)
- Connecting modules using different MoCs:
  - SystemC: DE
  - SystemC-AMS: SDF, Continuous time (LN)
  - HetSC: PN, KPN, CSP, SR, SDF
- Connecting module-ports with different data types
  - implicit conversion
- Minimal engagement by the designer
  - Design space exploration without the need to worry about custom converter modules for each variant (e.g. bit-widths)
  - Mixed Level Simulation
- No adaptation during simulation time (regarding reconfiguration)
- No simulator coupling (yet)



# Current state: usage

- Instantiation:

```
andres_convchannel<DT_W, DT_R1 , ..., DT_R5> name;
```

- Connecting a polysignal **sig**:

- to output ports: `module.out(sig.source_MOC())`
- to input ports: `module.in(sig)`

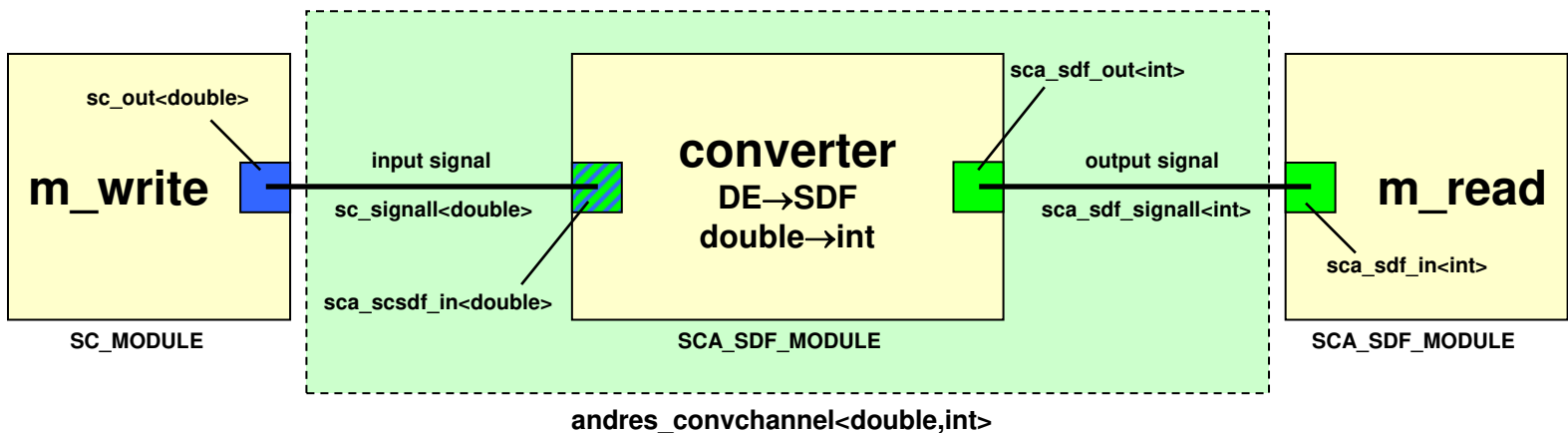
- Code-example:

```
andres_convchannel<double, int> sig;  
  
producer_sdf_double prodsdf("prodsdf");  
  prodsdf.out(sig.source_sdf());  
  
consumer_sc_double conssc("conssc");  
  conssc.in(sig);  
  
consumer_sdf_int conssdf("conssdf");  
  conssdf.in(sig);
```

# Current state: internal structure (1)

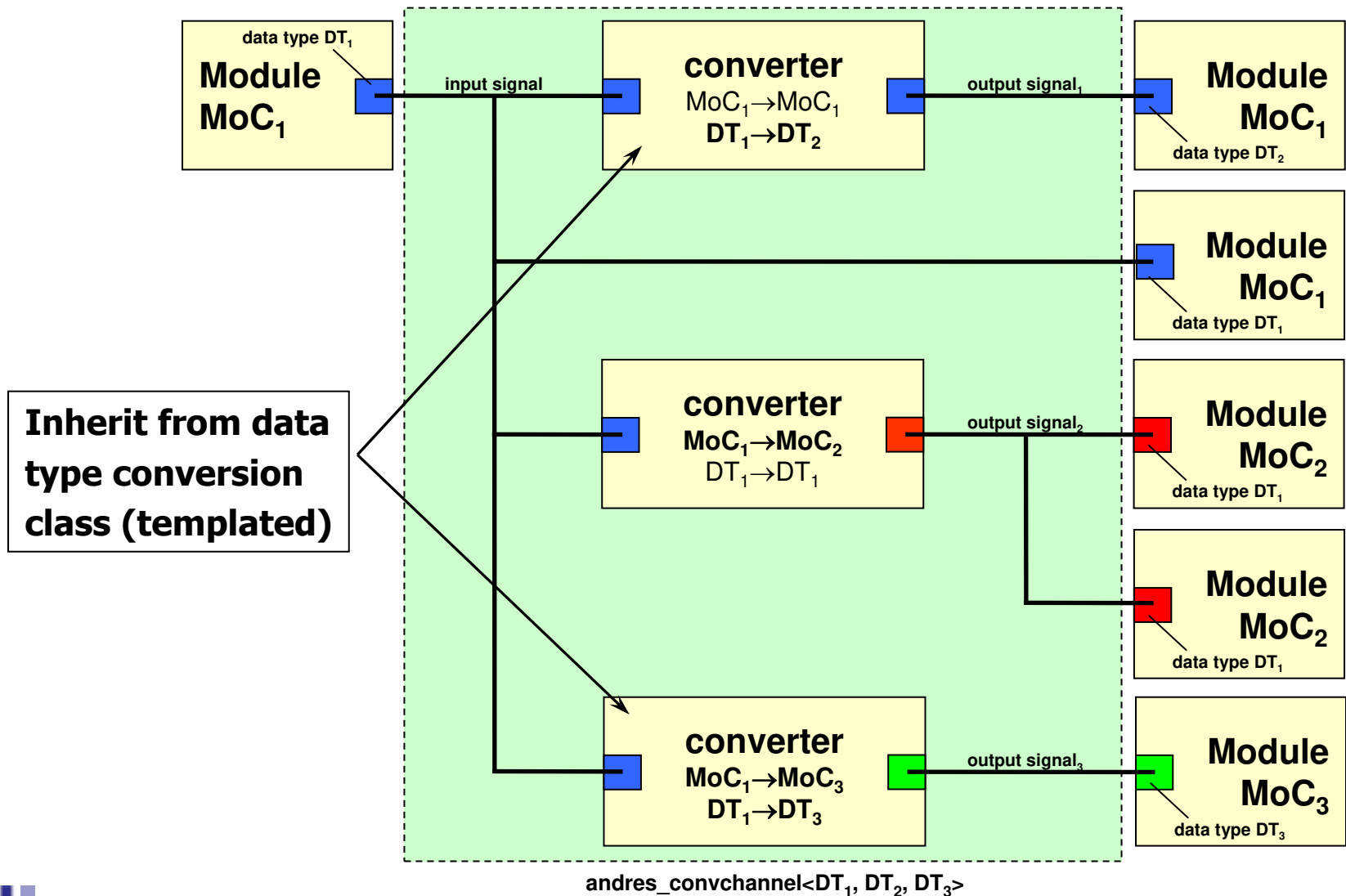
Four cases when connecting two modules:

1. MoCs and data types of the ports disagree
2. MoCs different, data types are the same
3. MoCs agree, data types of the ports disagree
4. Both MoCs and port data types agree.



**Goal:** Doing only what's necessary for any case

# Current state: internal structure (2)




# Value range scaling

- Problem: conversion between data types with very different value ranges
  - Example: converting `double` to `sc_uint<8>`
  - A simple conversion takes absolute values and would "discard" values not lying within the interval `[0,255]`
  - If the polymorphic signal plays the role of an A/D converter within a model, this is definitely not what we want
- Solution: Option for value range scaling
  - `sig.setRangeScaling( min , max )`
  - Scales the values of the writing side within the interval `[min,max]` to the (inherent) value scale of the reading side

# Automatic data type conversion

- Problematic conversions
  - Converting from `sc_logic`
  - Converting integer types to `double`, `float` etc: rounding, floor or ceiling-like conversions?
  - Converting Numerical values to `Bool` & co
- Possible: Conversion customizable by designer
- Should conversions with potential loss of information be included at all? (Your opinion?)

- Integration of HetSC-MoCs
  - A closer look at the conversion semantics is needed
  - No convenient constructs like the SC-SDF converter port available
  - Mostly untimed MoCs
- Integration of SystemC-AMS Linear Networks
  - Straightforward, since several converters are available
- Developing a theoretical framework
  - Based on the MoC-formalism of Axel Jantsch (ForSyDe)
- Later Future: Simulator coupling & TLM transactors



Thank you  
for your  
attention!