Modeling of Automotive Wheel-Speed Sensor IC's in SystemC

North American SystemC User's Group meeting XVIII, San Francisco, CA

6th June 2012

Dr. Fritz Rasbornig

Wolfgang Granig

Dr. Simon Hainz

Infineon Technologies Austria AG

Sense & Control

email: Fritz.Rasbornig@infineon.com

Agenda

- **Outline:**
  - Time before System(C) models
  - Overview of Infineon's SystemC models
  - A look at SystemC models for customers
  - How Infineon is modeling a SystemC model
  - An example of a SystemC model for customers and IFX internally
    - Black-box approach with embedded models in Matlab/Simulink
    - VHDL and SystemC co-simulations
  - New approaches …
  - Summary
  - Discussions and questions …

Look into History

- Starting point in the past (Time before SystemC models):
  - Discussions with customer:
    - Review of data-sheet regarding chip-behavior (Telecons, face2face meetings etc.)
    - Possible misinterpretation of specifications
    - Verification of digital algorithm behavior on silicon-level → "late-surprises" → needed redesign-steps → loose of time and money

  - Internal discussions with concept-engineer vs. chip-designer etc.)
    - Review of internal documentation
    - Possible misinterpretation of specifications internally
    - No "golden/reference" model available for pre-silicon verification

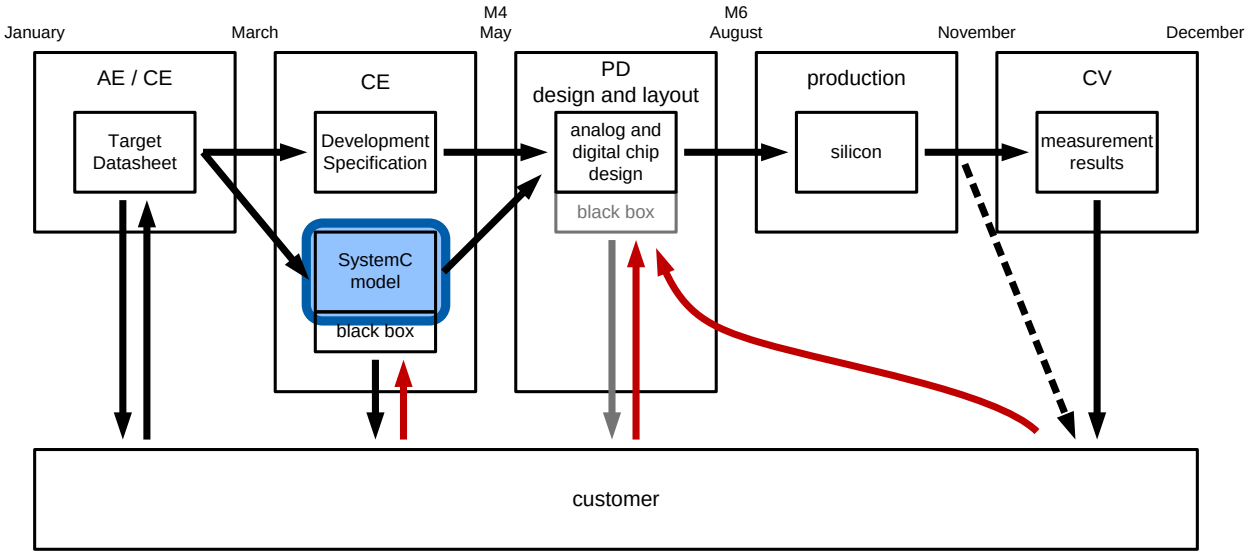
General Aspects

Overview of SystemC models today:

- Infineon (department Sense & Control) is using SystemC-models for mechanical, thermal and electrical investigations for automotive sensor products for several years
- SystemC models of sensor products (e.g. speed-, angle and linear Hall sensor) were delivered to several customers
- SystemC models are also used for internal co-simulation with VHDL → internal verification ("Executable Specification")
- Design-flow with the usage of SystemC-models:

Example Wheel Speed Sensor Project:

AE=Application Engineering
CE=Concept Engineering
PD=Product Design
CV=Comonent Verification

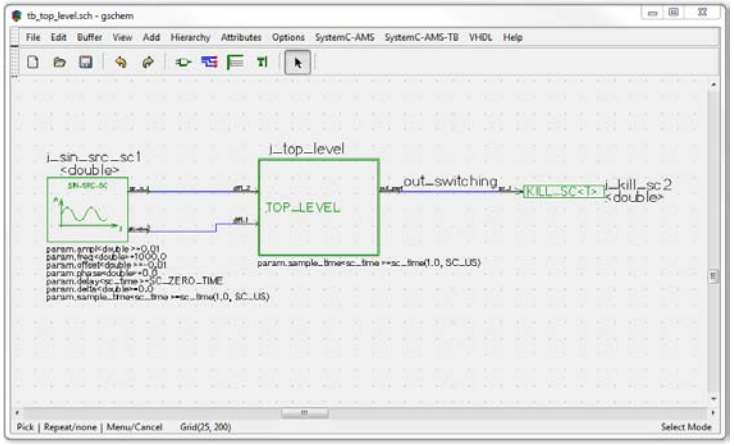


Motivation for SystemC/SystemC-AMS Models

- **Advantages of using SystemC models for customers and Infineon:**
  - Allow customer testing of chip features in a very early design step (before tape-out)
  - Reduction of development costs (e.g. avoiding redesigns etc.)
    - Time to market / First time right
  - Increase internal understanding of chip architecture and concept
  - Avoiding miss-interpretation of the specification
  - An accurate system model can deliver:
    - Understanding of the internal digital-algorithms, state-machines and output-protocols
    - Implementation of main signal processing flow (of analog- and digital-section)

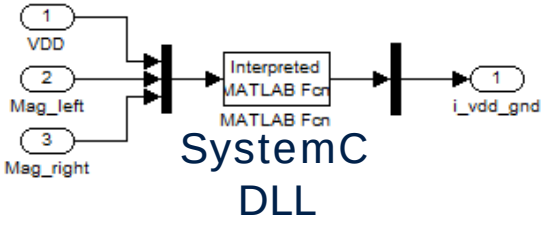
Customer & Internal Model of an ABS Speed-Sensor (TLE4941plusC)

- Analog blocks modeled „ideal“ → fast simulations results:
  - No internal offsets
  - Mismatch of transistors not considered (e.g. process variations)
  - Implementation of temperature dependence (optional)
  - UV/OV behavior can be seen
  - Current interface (7mA/14mA)
- Digital Core (detailed implemented):
  - System clocking
  - Digital algorithm implemented detailed
  - ADC/DAC granularity (typical signal-ranges)
  - Digital output-protocol



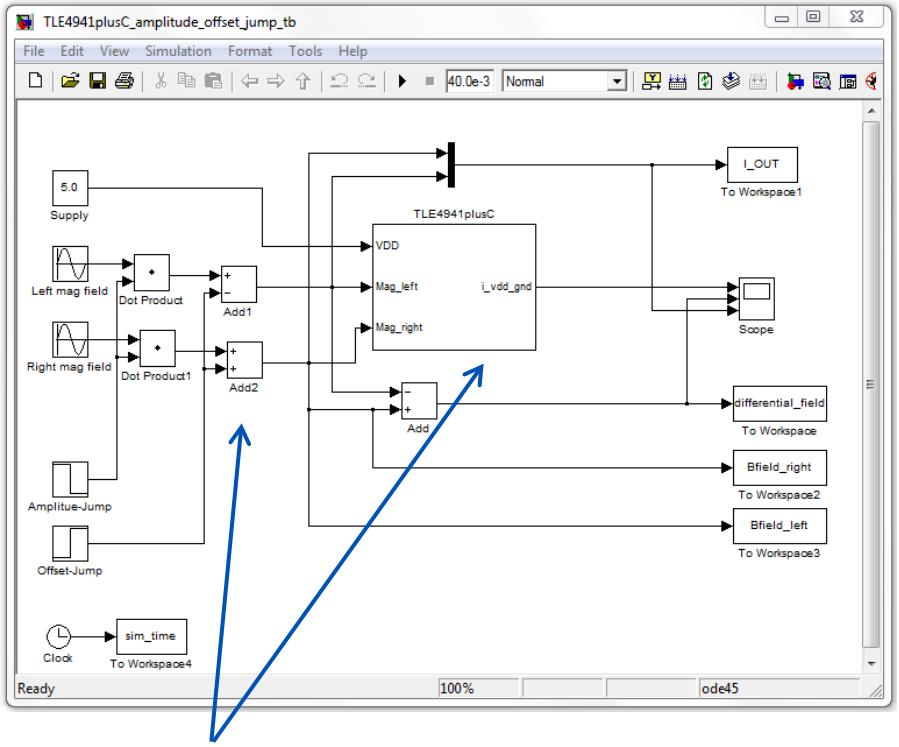
SystemC/SystemC-AMS Models for Customers

- **SystemC models for customers:**
  - Blackbox approach of embedded SystemC models in Matlab/Simulink → usage of a DLL
  - Customers can use their own specific testbenches in Matlab or Simulink
  - Customers can simulate behavior of IC, but have no detailed information about Infineon IP (algorithms etc.)
    - Simulating sensor-model with ECU-model long before silicon is available
  - Customers are requesting for system models to check the behavior of the IC
  - Optional Microsoft Excel interface → Providing input-file and/or storing simulation data in MS Excel format



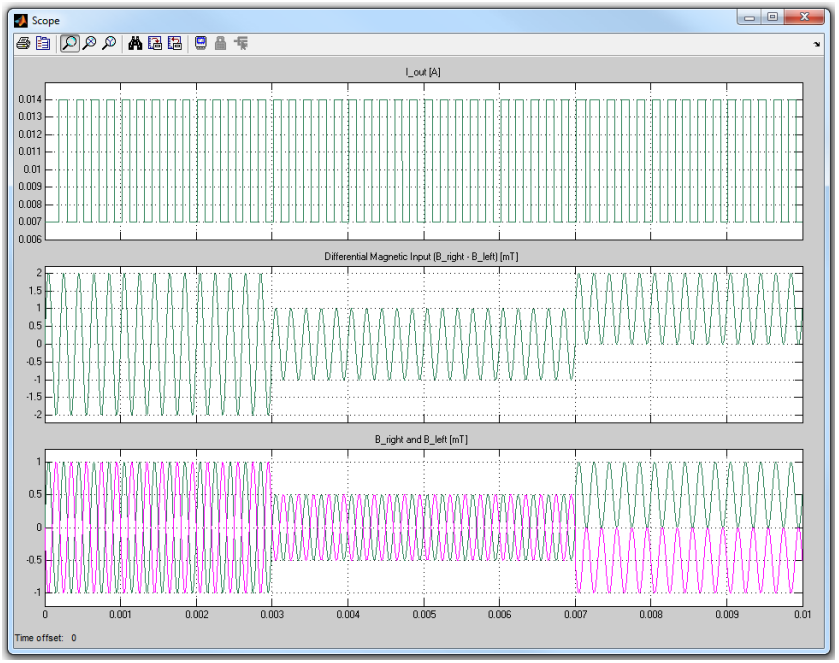
Embedded SystemC Model in Simulink

- **Simulink model of TLE4941plusC and simulation results:**



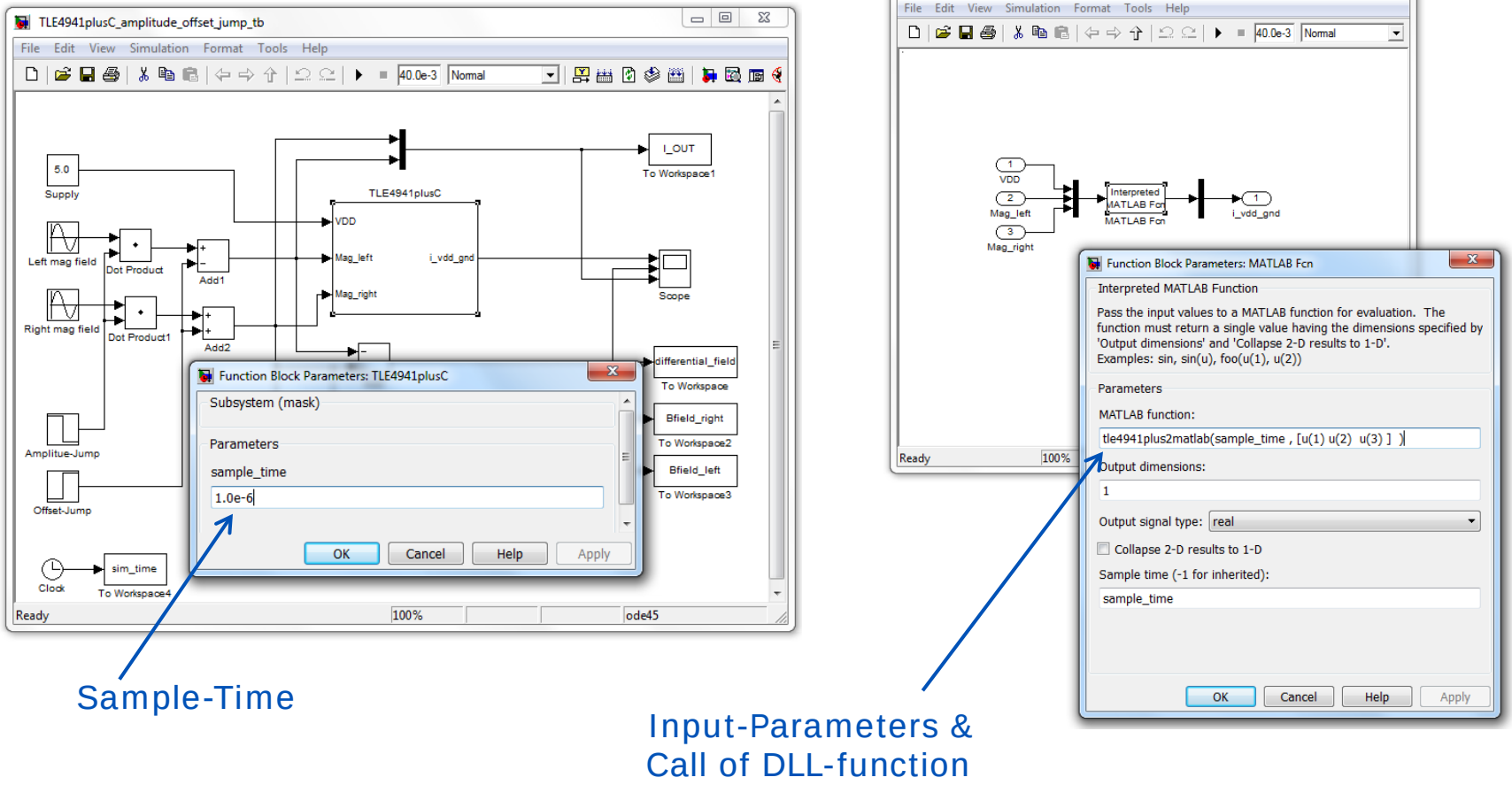
Simulink Top-level testbench & embedded SystemC model (DLL)

Simulation results:



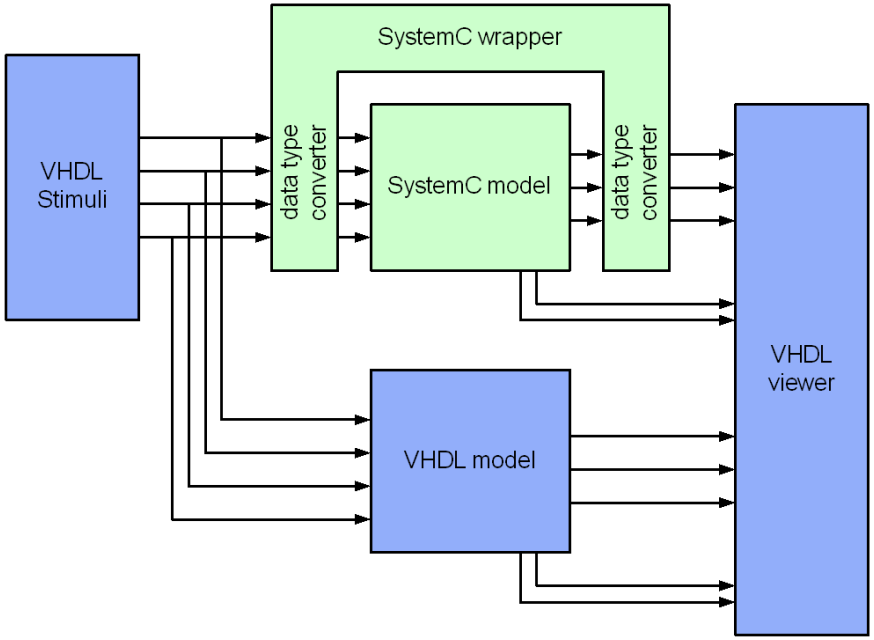
Detailed View of a SystemC Model in Simulink

- A “look” under the mask of an embedded SytemC / Simulink customer model:



VHDL & SystemC co-simulation

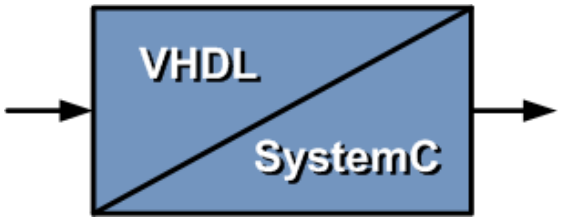
- VHDL & SystemC environment for verification of implementation:
  - VHDL and SystemC should get equal stimuli from VHDL test-bench
  - Output ports and internal signals of VHDL and SystemC should be traced by ModelSim



  - SystemC wrapper (possibly) required to convert data types such as SystemC-double to SystemC-integer
  - Additionally, a data type conversion (possibly) required in VHDL test-bench to convert data types such as VHDL-analog to VHDL-integer

VHDL2SystemC Converter

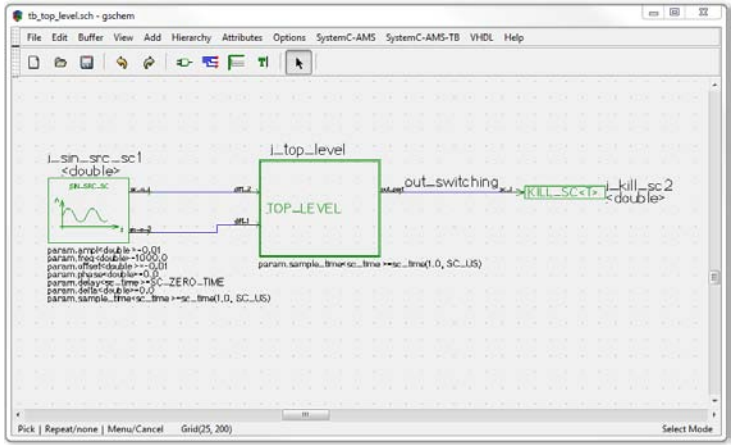
- New approach → Usage of VHDL2SystemC converter:
  - "Quasi" automatic conversion of digital-core units possible → bit-true conversion
  - Additionally analog basic blocks could be converted → optional
  - Advantages for customer and IFX:
    - Faster conversion of VHDL-code to System-C
    - Lower risk of misinterpretation of VHDL-code
    - Delivery of a SystemC model possible before tape-out

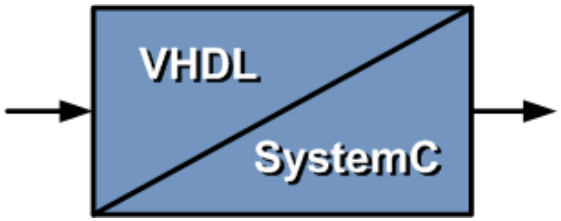


  - Points which have to be still reviewed/considered:
    - Behavior of the model should be the same as the behavior of the VHDL simulations
    - Digital- and (analog-) top-level wire connections have to be done manually

Summary and Outlook

- **Summary:**
  - SystemC models are used internally for VHDL-co-simulation as well as for customer-models for golden reference
  - Very positive customer response to this methodology
  - This approach results in faster development-cycles and prevent possible expensive and time-consuming re-designs.
  - Next step is to use a “quasi” automatic conversion of digital blocks with a VHDL to SystemC converter, to generate models with implementation details, fast and efficient.





Thank you for Paying Attention!
Do you have any Questions?

Special thanks to FHG Dresden who support us in all SystemC activities!

Fraunhofer IIS

ENERGY EFFICIENCY
MOBILITY
SECURITY

Innovative semiconductor solutions for energy efficiency, mobility and security.