Contents

automotive ECU challenges
State of the art in ECU design
Model based design
System Level Design Tools
System under Design

- automotive ECU’s
- Distributed system
- Distributed function
- Mechatronic
- Hard real time
- Complex communication
- Mechanics/Electrics Codesign

General structure of an electronic control unit
Embedded systems in a car

Relatively high production volumes (5,000 – 1,000,000)
High number of variants (countries, customers),
Reusability
Long term availability: > 15 years

tough operating conditions
- Temperature range: -40°C … +125°C … +175°C
- Supply voltage: 6V … 14V … 28V … (42V)
- Mechanical stress: acceleration, vibration
- Chemical stress: humidity, oil, exhaust gases, road salt …
- Electromagnetic compatibility

High reliability: << 1 ppm/h Failure rate

Performance, Reliability, Safety, Security, Costs, Weight, 3D shape and volume
Energy Consumption (5% of fuel for EE-Systems)
Diagnosis and Maintainability (Service, Updates, Lifelong-Guaranty)

Design Process

multiple interleaving design processes
Concurrent engineering
distributed between OEM and supplier
requires
comprehensive life cycle model (V-Model)
strictly controlled design methodology
supporting computer aided design tools
**V-Model for automotive ECU’s**

- System oriented Process steps
- Application Software oriented Process steps
- ECU oriented Process Steps

**Assuring Design Quality**

**Preventive Measures** (fight against failure origins)
- Design Methodology (life cycle, design cycle, V-model)
- Training
- Tools (Hardware/Software)
- Reliable Component Vendors
- Technology
  - Quality of Component Libraries
  - Experience, Service, Delivery Time, Costs

**Measures for Failure Detection** (get rid of failure effects)
- Validation and Verification:
  - Rule Checking
  - Simulation (System, Module, Component)
  - formal Verification
  - Test

Copyright © 2009 KIT Karlsruhe kmg
Verification and Validation in Systems Engineering

Verification
1. The process of determining whether the product of a given phase of the system development cycle fulfills the requirements established during the previous phase. "Am I building the product right?"
2. The act of reviewing, inspecting, testing, checking, auditing, comparing or otherwise establishing and documenting whether items, processes, services or documents conform to specified requirements.
3. Formal proof of correctness.
4. In systems engineering a generalized term that can mean:
   • test (using precision instrumentation)
   • demonstration (a functional test)
   • analysis (or simulation) or
   • examination (or documentation)

Validation
Determination of the correctness of the final system (SW, HW) produced from a development project with respect to the user’s needs and requirements. “Am I building the right product?”

Typical Designflow

System-Analysis
- executable Specs, models

System Design
- Customer Requirements
- Technical Requirements
- System Architecture
- Interfaces
- Modeling and Simulation
- Real Time Requirements

Rapid Prototyping
- Hardware Platform
- Code Generation
- Real Time Operating System
- configurable Interfaces

Detailed HW/SW-Design
- SW-Design, Data Dictionary
- HW-Drawings
- HW-Analysis Report

Preliminary HW/SW-Design
- HW-Architecture, SW-Architecture
- Interface Description

HW/SW-Requirements Analysis

HW/SW-Implementation
- Test, Integration, Test
  - SW-Modules, Data Dictionary, SW-Comp
  - HW-Component, HW-Module
  - HW-Realization Documents

System Integration, Test
- Calibration, Application
- Transition to Utilization

Idea

Copyright © 2009 KIT Karlsruhe kmg
Rapid Prototyping - Hardware in the Loop

Besides review, inspection, audits other means for verification and validation:
\[ \text{MiL, SiL, Simulation, Rapid Prototyping, Component Test, Integration Test (HiL)} \]
\[ \text{System Test, Drive Test, Maintenance Test, Life Time Tests} \]

ECU development for passenger cars:
3 Prototypes
Typical Designflow

What are the supporting system level design tools?

System-Analysis
executable Specs, models

System Design
Customer Requirements
Technical Requirements
System Architecture
Interfaces
Modeling and Simulation
Real Time Requirements

Idea

Interface Description

Detailed HW/SW-Design
SW-Design, Data Dictionary
HW-Drawings
HW-Analysis Report

HW/SW-Implementation
Test, Integration, Test
SW-Modules, Data Dictionary, SW-Comp
HW-Component, HW-Module
HW-Realization Documents

Rapid Prototyping
Hardware Platform
Code Generation
Real Time Operating System
configurable Interfaces

System Integration, Test
Calibration, Application
Transition to Utilization

Contents

automotive ECU challenges
State of the art in ECU design
Model based design
System Level Design Tools

Copyright © 2009 KIT Karlsruhe kmg
Challenges

Still increasing complexity (more comfort and safety functions coming)

number of ECU’s must not increase, should decrease!
requires redistribution (mapping) of software onto fewer hardware platforms
eventually new, more flexible architectures (multicore, dynamically reconfigurable?)

Today’s E/E architecture in a car is characterized by an assembly of (too)
many locally optimized subsystems

Only OEM can go for global optimum, this requires new tools for early
design space exploration in close cooperation with supplier

Ensure manufacturer/supplier interface for requirements specification

System specification as basis for cooperative design process

hundreds of pages, mainly textual descriptions, many drawings

expensive iteration cycles due to
- incomplete
- wrong
- ambiguous
- inconsistent system specification

Formal Specifications, executable, Model Based Design
Model based design: modeling issues

Modeling for complete system including system environment (ECU, car, driver, road, weather conditions)

Domain specific models for Subsystems and Components (closed loop control, reactive systems, probabilistic systems, software intensive systems)

Different abstraction levels, Parameter variation and boundaries (functional and non-functional data for early design space exploration)

Use of characterized libraries (reuse, variant design)

Model verification through extensive testing

Model characterization

Model documentation

Macro modeling

Meta modeling

Matlab/Simulink, ASCET, Statemate, Stateflow, UML, SysML
Model Based Design on System Level:
Models for Executable Specification and Analysis (Simulation) of Physical phenomena
Only to some extend models of computation

Abstraction Levels for Modeling and Simulation
Modeling

Heterogeneous systems: different domains and levels of abstraction

**Simulators:**
- Saber, Eldo, Spectre, VHDL-AMS and Verilog-AMS simulators,
  Matlab, MatrixX, ...
- ANSYS, MAFIA, CAPA, ABAQUS, Nastran, FEMLAB, ...

Complex, Heterogeneous: mechanical, electrical, fluidical, optical phenomena and their couplings

Multi Domain Multi Physics Systems

Tight coupling between: complete system, system environment
subsystems, components
manufacturer and technology dependencies

Multi View, Multi Abstraction Level Approach

Long iteration cycles: design, manufacturing and test very expensive and time consuming
Simulation is a must

Complicated non-linearities: strong functional influence, small signal behavior not sufficient:
electrostatic force prop. \( V^2 \), capacitance prop. \( 1/d \), Hysteresis,
several working points

Dynamic systems: extremely different time constants: > 10 orders of magnitude,
stiff differential equation systems

Time and space derivatives: FEM Simulation and Analysis, 3D fields and waves
System Level Modeling and Simulation

Mechanics
Electrostatics
Electromagnetic Fields
Thermal problems
Multi-Physics Problems

Modeling

Different ways to model a technical systems

„Physically oriented“

Technical System

Geometrical structure

Mathematical description: PDE

Discretization (FEM, FDM, …)

DAE, ODE, algebraic equations

Order reduction

Simulation in time or frequency domain

Simulation results

Black-box model generation

Parameterizable analytical element models

Numerically generated behavioral models

Modeling languages used: Spice, SABER, VHDL-AMS, SystemC_AMS, Modelica

Copyright © 2009 KIT Karlsruhe kmg
Courtesy Peter Schwarz, FhG Dresden
Requirements for new system level tools

Model based design as a basis.
- Is accepted in research and predevelopment,
- not yet standard in ECU development

Design space exploration means
- distribution of hardware and software under consideration of sensor/actuator locations
- computation performance as well as communication performance
- Codesign not only for hardware and software but also function, variants, safety, security, cost …

Metrics and parameters used are domain specific
- therefore, domain specific system level tools are required
- interfacing seamlessly with component specific tools
  (meet in the middle)

A lot of model transformations are required

System Design: Meet-in-the-Middle – Strategy

(DEBYS – design by specification project 1989)

- CA-tools for Systems Engineers
- Evaluate Design Alternatives
- Architectural Synthesis
- High Design Productivity and Efficiency
- Short Development Cycles
- Mostly Manufacturer- and Technology Independent

- Closely related to Technology
- Mostly Manufacturer- and Technology Dependent
- CA-Tools dedicated and highly flexible
- For highly qualified Specialists only
- Optimization and Characterization of all Parameters
- Result: quality-assured Library for Variant Design
Model Based Design: Meet-in-the-Middle Strategy

**TOP-DOWN:**
- Concepts
- Formal Specification
- Successive Detailing

**BOTTOM-UP:**
- Abstraction
- Macro modeling

**Requirements-Description**

**Executable Model**

**Formal Model**

**DFD**
**CFD**
**StateCharts**
**ER Diagrams**

**Mathematical Model**

**Graphs**
**Equations**
**Transfer-Functions**

**Knowledge based Selection of Models**
**Verify and Validate (by Simulation)**
**Macro Models**

**Real Technical System**

"Meet-In-The-Middle" - Strategy

**System Specification**
**System Design**
**System Verification**

**Requirements Analysis**
- Functional/nonfunctional
- Requirements/Constraints
- Design Space Exploration

**Project Planning**
- Time/Cost plan
- Personnel and Tools
- Activities and Relations

**Requirements Specification**
- Formal System Model
- Function/Behavior/Architecture
- Verification and Validation

**Bottom-Up Information Base „Capabilities“**
- Subsystems, Components, Variants
- Rich Component Models
- Reliability, Safety, Security
- Technological Alternatives

**Data Exchange Formats**
- XMI, MOF, STEP...

**Interdisciplinary design environment system integration**

**Component Design and Characterization**

- Software
- Digital-Hardware
- Analog-Hardware
- Micro-mechanics
- Micro-optics
- other Sensors/Acutators

- Wiring Harness
- PIT

Information Basis Technologies, Materials, Components

Copyright © 2009 KIT Karlsruhe kmg
"Meet-In-The-Middle" - Strategy

System Design Environment:
- Requirements Engineering
- Sortie and Life Mission Analysis, Feature/Function Analysis,
  Architecture Synthesis and Optimization
  (Design Space Exploration, Reuse, Estimation, Evaluation of Alternatives,
  pre planned product enhancements life cycle considerations, DFX,
  Computation/Communication,
  Safety, Security, Dependability, ISO26262
- Cost Issues product and life cycle,
  System partitioning, HW, SW
- Baseline, Product Line, Variants, Project Planning

System Design Environment:
- Verifications and Validation
  (Reviews, Rule Checking, System Simulation,
  Formal Verification)
- Compositional model building
- System Integration
  (models, 3D PIT, mechatronics)
- Bio., Phys., Chem. compatibility
- Parasitics extraction and analysis
- Interface to component design
- Test and Test Systems support

Component Design Environment:
- Dedicated tools
- On-Line Model Generation
- 4D models (FEM, FBM)
- Component Analysis and Component Optimization
  (Cross) sensitivity analysis
- Verification and Validation
  (Model Level)
- Macro modeling
  -(multi level, mixed mode)

Component Design Environment:
- Parasitics Extraction and Analysis
- Interface to system level design
- Compositional model support
- Characterized Libraries
- Component test and system test support

Information Basis Technologies, Materials, Components
"Meet-In-The-Middle" - Strategy

System Integration Design Environment:  
- Packaging and Interconnect  
- Technology selection  
- On-line model generation  
- Component composition analysis and composition optimization  
- (Cross) sensitivity analysis  
- Verification and Validation (composition model level top down vs. bottom up)

System Integration Design Environment:  
- Macro modeling (multi level, mixed mode)  
- Bto., Phys., Chem. Compatibility  
- Parasitics Extraction and Analysis  
- Interface to system level design  
- Compositional model support  
- Characterized Libraries  
- Compositional component test and system test support

Component Design and Characterization

<table>
<thead>
<tr>
<th>Software</th>
<th>Digital-Hardware</th>
<th>Analog-Hardware</th>
<th>Micro-mechanics</th>
<th>Micro-optics</th>
<th>Sensors/Actuators</th>
<th>Wiring Harness, PIT</th>
</tr>
</thead>
</table>

Information Basis Technologies, Materials, Components

Modeling for heterogeneous electronic embedded systems

Architecture  
Modelling with UML  
- Real-time Studio (ARTISAN)  
- Rhapsody in C++ (i-Logix)  
- Rose (Rational Software, IBM)  
- Together (Borland)  
- Poseidon (Gentleware)  
- MagicDraw (NoMagic)  
- Ameos (Aonix)  
- TAU2 (Telelogic)

Signal flow oriented  
Modelling with block diagrams  
- ASCET (ETAS)  
- MATLAB/Simulink (The MathWorks)  
- MATRIXx (National Instruments)

Event driven  
Modelling with state charts  
- Rhapsody in C++ (i-Logix)  
- Statemate (i-Logix)  
- Stateflow (The MathWorks)  
- ASCET (ETAS)

Heterogeneous modeling requires integration platform  
e.g. ETAS Integrio, Vector DaVinci
ITIV/FZI Tool integration platform (model transformation)

GeneralStore CASE-Tool Integration Platform

Tool chains
other Challenges

- Algorithm Integration
- C, C++, Matlab
- SDL, SPW, Cossap
- Functional Network
- Does the functionally integrated design work?
- Executable Functional Specification
- Architecture
- Performance
- CPU, DSP, Bus, I/O, Memory, HW, SW, RTOS
- Unambiguous Structure
- Are Partitioning & Performance Sufficient?
- Executable Performance Specification
- Detailed design

Goal (AUTOSAR)

- Hardware A
- Hardware B
- Functional Integration
- ECU Library
- Vehicle A
- Vehicle B
- Mapping A
- Mapping B
- Code Generation
- Function Library
- Seat Adjustment A
- Seat Adjustment B
- Lighting
- Seat Heating A
- Seat Heating B
- Air Conditioning
Evolution of hardware/software architectures in a car

1. Add-on
   - ECU

2. Networking
   - Vehicle specific data bus
   - CAN-Bus Architecture

3. Integration
   - Standardized software modules (OSEK/VDX)
   - Open System Architecture

4. Vehicle Module Orientation
   - (highly integrated)

Evolution led to open system architectures with modular software architecture:
Milestones: CAN, OSEK/VDX (AUTOSAR)

Another Challenge: Upcoming X-by-Wire Systems

Delayed for 4 to 5 years

First driver assistance systems overruling driver currently being introduced (truck emergency brake system)

EN 61508 norm for safety critical electronic control systems not yet finally adapted for car industry (new standard 26262 for 2009)

System Redundancy required:
   - HW redundancy: sensors, actuators, ECU’s, busses (Flexray) doubled
   - Information redundancy: error detection codes used
   - Time redundancy: all messages send twice on each bus
   - Software Redundancy: two version programming?

Certification required as in aerospace industry?
Distributed ECU’s in cars - design challenges

Still increasing complexity (more comfort and safety functions coming)

Today's E/E architecture in a car is characterized by an assembly of (too) many locally optimized subsystems

Only OEM can go for global optimum

new system level design exploration tools are required

Requirements for new system level tools

Model based design as a basis.
  Is accepted in research and predevelopment, not yet standard in ECU development

Design space exploration means
  distribution of hardware and software under consideration of sensor/actuator locations
  computation performance as well as communication performance
  Co-design not only for hardware and software but also function, safety, security

Metrics and parameters used are domain specific
  therefore, domain specific system level tools are required interfacing seamlessly with component specific tools (meet in the middle).

A lot of model transformations are required
Virtual Development

E/E Architecture Development with PREEvision®
Architecture Views in Concept Development

Electronical


SW

HW

Electrical

Physical Geometrical

Function Architecture
Functions and Subfunctions? Interaction?

Software Architecture
Software Structure? Standards?

Network Architecture
How do ECUs communicate? Performance Characteristics

Component Architecture
How many ECUs, what performance? Variants, scalability?

Power Supply/Power Distribution
Electrical Power Supply System? Sources and Sinks, Dynamic Behavior, Idle State Power Consumption (Ignition Off Drain Current)?

Component Topology
Where are EE Components Located? Assembly/Maintenance/Recycling?

Interfaces Relationships Mapping Processes

Views

List of Features
FL

Feature Function Network
FFN

Functions Network
FN

Components and Networking
KN

Topology
TOP

Which features?
What is the concept behind the features? How do they interact?

Detailed Specification of Functions architecture
Comp. Architecture Network Infrastructure Power Distribution Topology wiring harness
Views onto automotive E/E-systems

Typical domain specific views
- Features
- Functions
- Components
- Component locations and wiring

Design space exploration needs domain specific metrics and parameters

Model-Based E/E Architecture Design (Inputs)

Features | Requirements | Design Objectives | Sales & Marketing

Variant Configuration & Management
- Technical Concepts
- Equipments
- Platforms

Model-Based Design

Electronic / Electric Architecture Model (Domain Specific Notation)
Model-Based E/E Architecture Design (Outputs)

Electronic / Electric Architecture Model (Domain Specific Notation)

Model-Based Design

Model Optimization - Consistency Checks - Refactorings - …

Architecture Handbook
Metric Results
Benchmarks
Various Exports

E/E-Architecture Trade-off Studies: e.g. Door Control

Alternative 1

Door FL Actuator
Door FL Switch
Door RL Actuator

Body Control Module

Alternative 2

Door FL Actuator
Door FL Switch
Door RL Actuator

Body Control Module

Network Connected
EE-Architecture alternative solutions

System Oriented

Topology Oriented

Central Control Unit

Central Control Unit with Specific Sub Busses

Architectural Evaluation & Benchmark

- Metrics
- Counting Metrics (weight, length etc.)
- Cost Metrics
- Complexity Metrics
- Power consumption metrics
- Bus load
- ...

Copyright © 2009 KIT Karlsruhe kmg
E/E Architecture Layer Model

Requirements  Feature-Functionality-Network

Function Network

Hardware Architecture

Topology

Feature List  FL

Feature list trace to, trace from relationships within feature list or in other layer

Feature list is part of model
Detailed View onto Seat Module

Connecting Layers by Mapping

Funktionsnetz (FN)  Vernetzung (NET)

Mappings

Distribution of Functions by Mapping
Functions onto Components

⇒ Assignment of Signals onto Bussystems

Copyright © 2009 KIT Karlsruhe kmg
Power Supply Concept

Fuse Strategy

Schematic Drawings with internal and external components
Connecting NET→TOP by Mappings

Mapping of ECU's onto Topology-Layer

Vernetzung (NET)

ECU

ECU

ECU

GW

Topologie (TOP)

ECU

ECU

ECU

EB1

EB2

EB3

EB4

EB5

EB6

EB7

Bauraum

Schematics linked to wiring harness

Schematics

Synthesis

Wiring Harness

Copyright © 2009 KIT Karlsruhe kmg
Harness routing, component positioning

Mapping between Components and Topology

EE-Architecture Concept Tool  PreeVision (www.aquintos.com)

User Checks
• AQ-Checks (88 checks included)

User Algorithms
• Dijkstra (Algorithm included)

User Plugins
• AQ-Demo Plugin (All eclipse plug-ins included)

User Templates
• AQ-Demotemplate (All templates included)

User Metrics
• AQ-Demo Metrics (All metrics included)

Consistency Check
Rule Engine

Signal- and Wire-Router and Weight Algorithms

Eclipse Framework
JAVA Application interface (API)

Report Generator (OpenOffice) pdf, html

Benchmark with Metrics

Variant Propagation
Rule Engine

Model-Based E/E-
Architecture Development:
Variant Management
Graphical Modeling
(10 Editors, 10 views)

Model Query
Engine

Label and Object
Schema (Color, …)

Import and Export
Engine

Multuser Support

MySQL

ORACLE

User Queries

AQ-Model Queries (99 queries included)

User Schemas

Default Schema (1 schema included)

Userconfiguration

AQ-Interfaces (3 interfaces included)

MySQL®

Oracle®
Important issue: efficient M2M-Transformation

Optimized Transformer-Engine with Interfaces to
- ETAS ASCET® (>= 5.1)
- The Mathworks MATLAB®/Simulink®/Stateflow® (R13 – R16)
- Fully integrated in PREEvision (for model consistency checks, variant propagation…)

Model-based Specification of Transformation Rules
- Rule Set modeled with UML
- Maintainability, Readability
- Automated Code Generation of the Rule-Set, no manual design process behind

Purpose of M2M Transformation
- Model data migration
- Model-Refactoring
- Model-Optimization
- Model-Verification

M2M Engines Architecture

Source Model Tool A

Rule-Model UML

Target Model Tool B

Importer

Transformator

Exporter
HiL-Test System

<table>
<thead>
<tr>
<th>System under Test</th>
<th>HiL-Test System</th>
<th>Testing Host</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Functional Layer</strong></td>
<td>Simulation</td>
<td>Host-Com</td>
</tr>
<tr>
<td><strong>Basic SW Layer</strong></td>
<td>Signal Pool</td>
<td>Test Software</td>
</tr>
<tr>
<td>Routing Layer</td>
<td>ECU Basic Software</td>
<td>Operating System</td>
</tr>
<tr>
<td><strong>Hardware Layer</strong></td>
<td>Computation Unit</td>
<td></td>
</tr>
<tr>
<td>ECU Hardware</td>
<td>IO Generation</td>
<td>PC-IF</td>
</tr>
<tr>
<td></td>
<td>Signal Conditioning</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fault Insertion Unit</td>
<td></td>
</tr>
<tr>
<td><strong>Physical Layer</strong></td>
<td>ECU Hardware</td>
<td>PC Hardware</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Simulation of functional Environment

Design Flow

Projection and Faithfulness

Data from Model of EE-Architecture

Executable Models / Implementations from former production series

Special Requirements to Tests and Test System
Identification and propagation of Information through the EE-Model

Cut out methodology using model query rule chains

Determination of information of interest to extract from the EEA model
  □ Representation of extracted information in XML file format ensures easy handling for further processing
  □ Representation of extracted information has to be based on a context specific meta model

Model query rule chains find specified correlations and combinations of model artefacts
  □ Model query rule chains find sequentional AND connections between chain links
  □ Every possible chain of interest needs a dedicated model query rule chain
  □ Redundancy caused by identical hits of different chains has to be erased before generating the final representation of the extracted information

Copyright © 2009 KIT Karlsruhe kmg
Extraction out of EE-Model – Component Network

- Type of connection
- Bandwidth
- Messages, Composition
- Timing Information
- Signal Encoding

Extraction out of EE-Model – Function-Network

- Interface
- Data Element
- Function Type
- Type of Connection
- Bandwidth
- Signals
- Communication Matrix
HiL Architecture Synthesis

Contents

automotive ECU challenges
State of the art in ECU design
Model based design
System Level Design Tools
Tools used for ECU design

- specification support (Doors, QFD/Capture)
- reactive systems (SDL, Stateflow, Statemate)
- closed loop control systems (ASCET-SD, Matlab/Simulink, MatrixX)
- software systems (Real-time Studio, Rhapsody in C++, Rose, Together, Poseidon, MagicDraw, Ameos TAU2)
- performance analysis (SES/Workbench, Foresight)
- rapid prototyping, HiL (dSPACE, ETAS, IPG, Quickturn)
- application, test, diagnosis (ETAS, Hitex, Vector, RA)
- architecture evaluation (Aquintos Preevision)
- C-Verifier (PolySpace)
- ASIC Design (Cadence, Mentor, Synopsys)

System Level Tool Support

- Power electronics
- Actuators
- Sensors
- Real environment (optical, mechanical, thermal, electrical, magnetic)
- Special interfaces
- Analog signal processing
- Microcontroller DSP
- Real Time Operating System
- Digital Signal Processing
- Communication with other Systems
- System Control
- Power Supply

Not seamless

somehow satisfying support: standard hardware platforms, software, RTOS, Sensors und Actuators
More domain specific system level tools required

System Design Environment:
- Requirements Engineering
- Sortie and Life Mission Analysis,
- Architecture Synthesis and Optimization
- Estimation, Evaluation of Alternatives,
- life cycle considerations, DFX,
- Safety, Security, Dependability, ISO26262
- Cost Issues product and life cycle,
- System partitioning, HW, SW
- Baseline, Product Line, Variants, Project Planning

System Design Environment:
- Verification and Validation
- System Integration
- Parasitics extraction and analysis
- Interface to component design
- Test and Test Systems support

Conclusion (1)
- What system level tools should provide
  - Documentation (readable for men, specific for application domain)
  - Data exchange between all designers across company boundaries
  - Data exchange between computer aided tools supporting distributed databases
  - Intellectual Property, reusable in libraries
  - Parameterized for variant design
  - Supporting standards and guidelines (e.g. HIS, Autosar)
  - Testable (Fault models, automatic Model validation), quality assured
    (automatic generation of test pattern and test bench) and documented
    (what is modeled, but also what is not modeled)
  - Seamless in design flow
    (Analysis, Design, Verification, Integration, Validation, Test, Application, Diagnosis)
  - Reviews, Rule Checking, Simulation, Formal Verification, Model Checking
  - Synthesis, automatic, interactive optimizing (e.g. RP-Code, Production Code)
  - allow access for automatic parameter-extraction
Conclusion (2)

Design studies show:

- Model based methodologies and tools are well performing and promising
- Seamless design flow only partially given (e.g. digital hardware, software).
- Interfaces for Modeling, Simulation, Characterization mostly manual
- hard problem for design of embedded systems
  - Cross sensitivity of Components (insufficient characterization)
  - Safety, Security, Function-Codesign
  - According modeling is really time and cost consuming
  - Mixed-Mode, Multi-Level-Simulation required
  - Formal Verification und Validation not possible?!
    - Non functional requirements
    - Time-, frequency- und parameter-domain
  - Module / System-Integration und –Test
  - Cross-sensitivities, EMC, Certification

Model based system design is possible, but there are many design and analysis steps still missing, especially in early design phases.

Conclusion (3)

Industrial design practice shows:

- Challenges for the design of embedded systems
  - many modeling techniques from computer science not adequate: FSM, Hybrid Automata, LSC, MSC, Petri nets, process algebra, Statecharts, Temporal Logic, Timed Automata, Z...
  - Is academic willing to prove their research results for real designs?!
  - Seamless flow required with respect to industrial life cycle processes, therefore support of standard interfaces must be done also by academics
  - There exist large libraries in different description methodologies that can’t be neglected
  - There exist standard RTOS (OSEK/VDX) and bus systems
  - There exist tight cost boundaries
  - New algorithms and tools must be made commercially available
  - Engineering constraints, adequate description methods according to De-Facto-Standards (tools) must be obeyed: Matlab, ASCET, Statemate, Doors, Saber, VHDL, C, Assembler
  - Formal methods are not yet scaling for many real industrial problems
  - Required from industry: availability of real requirements, constraints, cost numbers etc. for research
- Required: more close cooperation between system manufacturer, (tier 1) suppliers, EDA companies and academics
Thank you very much for your attention

Contact:
Klaus Müller-Glaser
Karlsruhe Institute of Technology KIT
klaus.mueller-glaser@kit.edu
http://www.itiv.kit.edu