Codegenerierung für Embedded Systeme leicht gemacht – So geht’s!

Tobias Kuschmider
MathWorks
München, 9.07.2014
Agenda

- Model-Based Design – An Introduction
- Use of Production Code Generation for Implementation
- Proving Functional Equivalence between Model and Code
- Summary
Demonstration – Model-Based Design
Agenda

- Model-Based Design – An Introduction
- Use of Production Code Generation for Implementation
- Proving Functional Equivalence between Model and Code
- Summary
Model-Based Design

- Executable Specifications
- Design with Simulation
- Continuous Test and Verification
- Automatic Code Generation
Model-Based Design
Development Process

- Requirements
- System Design
  - Environment
  - Physical Components
  - Algorithms
- Component Design
- Research
  - Data Analysis
  - Algorithm Development
  - Data Modeling
- Executable Specifications
- Design with Simulation
- Continuous Test and Verification
- Models
- Automatic Code Generation
- Implementation
- Code Verification and Validation
- Integration testing
- Complete Integration & Test
- Subsystem Integration & Test
- User Acceptance Testing

Subsystems:
- DSP
- FPGA
- ASIC
- Embedded Software
- Digital Electronics
- C, C++
Model-Based Design
Continuous Verification and Validation

- Requirements
- System Design
  - Environment
  - Physical Components
  - Algorithms
- Component Design
- Research
  - Data Analysis
  - Data Modeling
  - Algorithm Development

- Verification and Validation
- Code Verification and Validation
- Integration testing
- Subsystem Integration & Test
- Complete Integration & Test
- User Acceptance Testing

- Implementation
  - Embedded Software
    - C, C++
  - Digital Electronics
    - VHDL, Verilog
  - MCU, DSP, FPGA, ASIC

- Subsystem Implementation

- System-Level Specification
- System-Level Integration & Test
Model-Based Design
Development Process

Requirements

Requirements capturing in Word, DOORS, etc.

System Design
- Environment
- Physical Components
- Algorithms

Component Design

Research
- Data Analysis
- Data Modeling
- Algorithm Development

Subsystem Design

Implementation
- Embedded Software
  - C, C++
- Digital Electronics
  - VHDL, Verilog
- MCU
- DSP
- FPGA
- ASIC

Subsystem Implementation

Code Verification and Validation

Integration testing

System-Level Integration & Test

User Acceptance Testing

Complete Integration & Test

System-Level Specification

Subsystem Integration & Test
Model-Based Design
Multi-Domain Modeling and Algorithm Development

Methods for modeling systems in different domains

- Data Flow (Block diagram)
- Physical Modeling (Schematic)
- Event-Driven Systems
- Programing Language (Textual)

Research
- Data Analysis
- Algorithm Development

System-Level Specification
- Requirements
- System Design
- Physical Components
- Environment
- Algorithms

Complete Integration & Test
- User Acceptance Testing
- System-Level Integration & Test
Model-Based Design

Early Concept Verification

- Executable specifications
- Predict dynamic system behaviour by simulation
  - System & environment models
  - Less physical prototypes
- Use of simulation results for system design
  - Fast What-If studies
  - Short iteration cycles

Requirements

System-Level Specification

Research

• Executable specifications
• Predict dynamic system behaviour by simulation
  - System & environment models
  - Less physical prototypes
• Use of simulation results for system design
  - Fast What-If studies
  - Short iteration cycles
Demonstration – Motor Control, Design
Agenda

- Model-Based Design – An Introduction
- Use of Production Code Generation for Implementation
- Proving Functional Equivalence between Model and Code
- Summary
Model-Based Design
Development Process

Requirements

System Design
- Environment
- Physical Components
- Algorithms

System-Level Specification

User Acceptance Testing

Integration testing

Complete Integration & Test

System-Level Test

Code Verification and Validation

Subsystem Integration & Test

Component Design

Subsystem Design

Research
- Data Analysis
- Data Modeling
- Algorithm Development

Implementation

Subsystem Implementation

Embedded Software
- C, C++
- VHDL, Verilog

Digital Electronics
- MCU, DSP
- FPGA, ASIC
Model-Based Design
Automatic Code Generation

- C/C++, VHDL and PLC-Code Generation from one model
- Support for Fixed Point Data Format
  - Automatic scaling
  - Supported in Simulation and Code-Generation
- Easy integration of legacy C/C++-Code
- System development independent of the target
Demonstration – Motor Control, Code Generation
Codegeneration Report

- Automatically generated
- HTML links between model and code
- Contains all relevant information
Plattform Independent ANSI ISO C-Code

- Plattform independent C/C++ Code
- Navigation between model and code ensuring traceability

```c
/* Gain: '<S14>/number_of_pole_pairs' */
sin_coefficient = ctrlParams.Pmsm.PolePairs * B.Switch_fr;

/* Trigonometry: '<S14>/Trigonometric Function1' */
cos_coefficient = (real32_T)cos(sin_coefficient);
sin_coefficient = (real32_T)sin(sin_coefficient);
```
Optimized Code for ARM Cortex-M

- Code Replacement Tables
- Use of CMSIS library for code optimization

```c
/* Gain: '<S14>/number of pole pairs' */

sin_coefficient = ctrlParams.PmsgPolePairs * B.Switch_fr;

/* Trigonometry: '<S14>/Trigonometric Function1' */
cos_coefficient = arm_cos_f32(sin_coefficient);
sin_coefficient = arm_sin_f32(sin_coefficient);
```
Agenda

- Model-Based Design – An Introduction
- Use of Production Code Generation for Implementation
- Proving Functional Equivalence between Model and Code
- Summary
Model-Based Design
Continuous Verification and Validation

Requirements
System Design
- Environment
- Physical Components
- Algorithms

Component Design

User Acceptance Testing
Integration testing
Complete Integration & Test
System-Level Test

Code Verification and Validation

Implementation
- Embedded Software
- Digital Electronics
- C, C++
- VHDL, Verilog
- MCU, DSP, FPGA, ASIC

Subsystem Implementation

Subsystem Integration & Test

Research
- Data Analysis
- Data Modeling
- Algorithm Development

System-Level Specification

Integration & Test

Verification and Validation
Model-Based Design
Continuous Verification and Validation

Requirements

System Design
- Environment
- Physical Components
- Algorithms

Component Design

Research
- Data Analysis
- Algorithm Development

Subsystem Design

Implementation
- Embedded Software
- Digital Electronics
- C, C++
- VHDL, Verilog
- MCU, DSP, FPGA,ASIC

Component Tests
- Test vectors and expected outputs

System Tests
- Test vectors and expected responses

System validation/Acceptance Test

Integration testing

Code Verification and Validation

User Acceptance Testing

Complete Integration & Test

System-Level Integration & Test

Subsystem Integration & Test

Subsystem Implementation

Component tests in simulation

Run component tests on target

Run system tests on integrated controller

System tests in simulation

Component Design

Subsystem Design

Requirements
Model-Based Design
Subsystem-Level Integration & Testing

Processor-In-The-Loop Simulation
• Co-Simulation of real hardware and simulated environment
• Testing functional Equivalence

System-Level Specification
Subsystem Design
Processor - In - The - Loop Simulation
• Co-Simulation of real hardware and simulated environment
• Testing functional Equivalence

User Acceptance Testing
Complete Integration & Test
Integration testing
Verification and Validation
Subsystem Integration & Test
System Analysis

Requirements

System-Level Integration & Test
Demonstration – STM32F4 – Discovery, Equivalence Testing
Proving Functional Equivalence

- Pure simulation vs. Processor-In-The-Loop simulation
- Proof of functional equivalence successful
ARM Cortex-M4: Add Peripheral Blocks, Generate Code, and Deploy!

STM32F4xx Blockset
Agenda

- Model-Based Design – An Introduction
- Use of Production Code Generation for Implementation
- Proving Functional Equivalence between Model and Code
- Summary
Benefits of Model-Based Design

- Models: Core of the Development Process
- Unambiguous Description of Requirements (Executable Specification)
- Fast Evaluation of Design Variants
- Frontloading - Early Test and Verification
- Automatic Code Generation

⇒ Better Cooperation, Communication and Collaboration
⇒ Higher Product Quality