Modellbasiertes Echtzeittesten und automatische Codegenerierung
Agenda

- Introduction to Model-Based Design
- Rapid Control Prototyping
- Processor in the Loop – PIL
- Hardware in the Loop - HIL
Model-Based Design
Development Process
Model-Based Design

Development Process

- Requirements
- System Design
  - Environment
  - Physical Components
  - Algorithms
- Component Design
- Rapid Control Prototyping
- Verification and Validation
- Subsystem Implementation
  - DSP
  - FPGA
  - ASIC
  - Embedded Software
  - Digital Electronics
  - C, C++
  - VHDL, Verilog
  - MCU, DSP, FPGA, ASIC
- Integration & Test
  - SIL/PIL
  - HIL
  - Complete Integration & Test
- Code Verification and Validation
- Integration testing
- User Acceptance Testing

Research
- Data Analysis
- Algorithm Development
- Data Modeling
Model-Based Design
Rapid Prototyping

Requirements

System Design
- Environment
- Physical Components
- Algorithms

Component Design

Verification and Validation

System-Level Specification

Research
- Data Analysis
- Algorithm Development
- Data Modeling

Rapid Control Prototyping

Integration 
- Testing

User Acceptance Testing

Complete Integration & Test

System-Level Integration & Test
HIL

Code Verification and Validation

Subsystem Integration & Test
SIL/PIL

Implementation
- Embedded Software
- Digital Electronics
- MCU
- DSP
- FPGA
- ASIC

Integration

Subsystem Implementation
Why do Rapid Prototyping?

- Test, verify, validate, and prove your design with hardware under test
- Evaluate new ideas using a production independent development platform
- Reduce costs, shorten time-to-market, and minimize risks
Rapid Controller Prototyping

- **HOST**
- **TARGET**
- **DEVICE**

`Test Suite` → `Controller Model` → `Hardware (Plant/System)` → `Verification`
XPC Target workflow

Real-time execution of Simulink models
Demo – Rapid Control Prototyping
Rapid Prototyping

- Prove your design
- Evaluate new ideas
- Reduce costs
- Shorten time-to-market
- Minimize risks
Model-Based Design
Automatic Code Generation

Requirements

System Design
- Environment
- Physical Components
- Algorithms

Component Design

Verification and Validation

Integration testing

Code Verification and Validation

User Acceptance Testing

Complete Integration & Test

System-Level Integration & Test HIL

Subsystem Integration & Test SIL/PIL

Research
- Data Analysis
- Algorithm Development
- Data Modeling

Rapid Control Prototyping

Implementation

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

Digital Electronics
- MCU
- DSP
- FPGA
- ASIC

Integration

Subsystem Implementation
Model-Based Design
Automatic Code Generation

- C/C++, VHDL/Verilog and PLC Code
- Support for Fixed Point Data Format
  - Automatic scaling
  - Supported in Simulation and Code
- Easy integration of legacy C/C++ Code
- System development independent of the target

C, C++
VHDL, Verilog
Structured Text

MCU
DSP
FPGA
ASIC
PLC

DSP & µC
FPGA & ASIC
HDL Coder (VHDL, Verilog)
Embedded Coder (C, C++)
PLC Coder (Structured Text)
Demo – Code Generation
Model-Based Design
Continuous Verification and Validation

Requirements

System Design
- Environment
- Physical Components
- Algorithms

Component Design

Verification and Validation

System-Level Specification

Research
- Data Analysis
- Algorithm Development
- Data Modeling

Rapid Control Prototyping

Code Verification and Validation

User Acceptance Testing

Integration testing

Complete Integration & Test

System-Level Integration & Test

HIL

Complete Integration & Test

Subsystem Integration & Test

SIL/PIL

Subsystem Implementation

Implementation
- Embedded Software
- Digital Electronics
- MCU, DSP, FPGA, ASIC

Integration

Processing

Verify

Generate
Why do Processor in the Loop (PIL)?

- Verify numerical output of code
- Profiling
  - Execution
  - Coverage
  - Stack
- Verify behaviour of target specific code
- Investigate compiler settings and optimizations
Use PIL Simulation to Verify

Simulink

Test Signals

Controller Model

Code Generation

Embedded Processor

Controller C Code

PIL Implementation

Serial / TCP/IP

Verifications
Demo - PIL

Controller Algorithm for Permanent Magnet Syncl

Copyright 2010-2012 The MathWorks, Inc.

Mode Scheduler

Motor Control

velocity
position_valid
controller_mode
velocity_command
position_command

Embedded Processor

Models hardware on the embedded processor, including the controller algorithm software specification and peripherals.

motor_on
command_type
command_value

sensors

System Inputs

Command Channel

motor_on
command_type
command_value

PWM compare

Drivers Peripherals

Motor Commands

Model Description: Controller
Specifies controller software
The controller outputs control signals based on sensor data.
Why do Processor in the Loop (PIL)?

- Verify numerical output of generated (or legacy) code
- Coverage, Execution and Stack profiling
- Verify behavior of target specific code
- Investigate effects of compiler settings and optimizations
Processor in the Loop (PIL)

- Verify numerical output of code
- Profiling
- Verify target specific code
- Investigate compiler settings and optimizations
Model-Based Design
Continuous Verification and Validation
Hardware in the Loop
Why do Hardware in the Loop (HIL)?

- Substitute for unavailable parts of the system
- Test the system for safety and performance
- Minimize expensive downtime for the rest of the system
- Test operation and failure conditions that are difficult to replicate
Conclusion

- Model-Based Design
  - Core of the Development Process

- Rapid Control Prototyping
  - Fast Evaluation of Design

- Processor in the Loop – PIL
  - Early Test and Verification

- Hardware in the Loop – HIL
  - System level integration test