Implementation and Verification

Daniel MARTINS
Application Engineer
MathWorks
Daniel.Martins@mathworks.fr
Agenda

- Benefits of Model-Based Design
- Verification at Model level
- Code generation
- Verification at Code level
- Certification process
Agenda

- Benefits of Model-Based Design
  - Verification at Model level
  - Code generation
  - Verification at Code level
  - Certification process
Why do complex systems fail?

- Insufficient specification
  - Design errors
  - Software coding errors
- Mechanical failure
- Human errors
Benefits of Model-Based Design

Research
- Data Analysis
- Algorithm Development
- Data Modeling

Design
- Environment
- Physical Components
- Algorithms

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

Requirements
- Environments for Test, Verification & Validation
  - Continuous V&V
Agenda

- Benefits of Model-Based Design
- Verification at Model level
- Code generation
- Verification at Code level
- Certification process
Is the algorithm model complete?

- Traceability to Requirements?
- Satisfies Modeling Standards?
- Free of design errors?
- Represents the intended behavior?
- Tested thoroughly?
Creating links between textual documents and model objects

During the normal mode of operation the Fault Tolerant Fuel Control System shall determine the fuel rate which is injected at the valves.

1.1.1. Stoichiometric mixture ratio
During normal model of operation the System shall maintain the stoichiometric mixture target ratio of 14.6.

1.1.2. Oxygen Sensor (EGO)
The System shall determine the amount of residual oxygen present in the exhaust gas (EGO) by reading the value of the EGO sensor. During a calibratable warm up period the oxygen sensor correction shall be disabled.

1.1.3. High Oxygen Level
If the EGO sensor determines a high oxygen level present in the exhaust gas, the System shall increase the fuel rate in order to maintain the stoichiometric mixture target ratio.

1.1.4. Fuel-rich Mixture
If the EGO sensor determines a low oxygen level present in the exhaust gas, the System shall decrease the fuel rate in order to maintain the stoichiometric mixture target ratio.

1.1.5. Manifold Pressure Sensor (MAP)
The System shall use readings from the MAP sensor to calculate air density and determine the equivalent air flow mass.
Static analysis of models against a set of checks
- for simulation
- for code generation
- Requirements Consistency

Modeling Standards Checks for:
- MAB Style Guidelines
- DO-178B, IEC 61508, ISO 26262, EN 50128

Modeling Standards Checking Overview
Simulink Verification and Validation
Identifying Design Errors Early

Simulink Design Verifier

Automatic identification of hard-to-find design inconsistencies in the model without running simulation

- Integer overflow
- Division by zero
- Out of bound arrays
- Dead logic
- Assertion violation
Verify the model against the requirements
Simulink Design Verifier

Explicit descriptions of required behavior – functional or safety requirement

Express Properties

Analyze model

Provide counter example for falsified property
Model Coverage Report
Simulink Verification and Validation

Coverage metrics identifies untested portions of your model
Agenda

- Benefits of Model-Based Design
- Production code generation
- Verification of Model, Code and Executable
- Code certification
- Certification process
C/C++ Code generation for Microcontrollers/DSP

- Code efficiency
- Code integration for C and C++
- Code generation from MATLAB
- AUTOSAR 4.x

Generated code is smaller than production hand code.

Table 2 shows ROM and RAM comparisons between hand code and auto code for a floating-point component in some typical powertrain software.

<table>
<thead>
<tr>
<th></th>
<th>Hand Code</th>
<th>Auto Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM</td>
<td>6408</td>
<td>6192</td>
</tr>
<tr>
<td>RAM</td>
<td>132</td>
<td>112</td>
</tr>
</tbody>
</table>

The auto code has less size of ROM and RAM compared to that of hand code. The auto code is readable and peer reviewed, and checked with the QAC static analysis tool. Most importantly, the auto code is implemented in a real-world powertrain application.

CONCLUSION

A custom data class allowing data type and data scaling information to be incorporated into the model is

MATLAB®: A modeling environment

DEFINITIONS, ACRONYMS

MBDG: Model-Based Development.
Multi-target Model (Generic: contains no data type information).
Target Specific Model: A generic specific data dictionary loaded and linked to the model.
ROM: Read Only Memory.
RAM: Random Access Memory.
KAM: Keep Alive Memory.
HDL Code Generation for ASIC/FPGA

- Generates Target-independent HDL Code
  - IEEE 1376 compliant VHDL®
  - IEEE 1364-2001 compliant Verilog®

- Optimization
  - Frequency driven timing optimization through automatic pipelining
  - Reduced area with multirate delay balancing
  - Resource sharing and streaming without over clocking

- Support for
  - Vivado, ISE, Quartus
  - Any Altera and Xilinx boards
  - Xilinx Zynq-7000 and Altera SoC platforms
Code Generation for PLCS

- Generates IEC 61131-3 structured text:
  - Simulink models
  - Stateflow charts
  - MATLAB code

- Support Several IDEs
  - 3S-Smart Software Solutions CoDeSys
  - B&R Automation Studio
  - Beckhoff TwinCAT
  - Rockwell Automation RSlogix 5000
  - Siemens Step 7
  - Rexroth IndraWorks, OMRON Sysmac Studio
Agenda

- Benefits of Model-Based Design
- Verification at Model level
- Code generation
- Verification at Code level
- Certification process
Is the code accurate and complete?

- Satisfies Coding Standards?
- Free of Run-time errors?
- Represents the intended behavior?
Static code analysis with Polyspace Products

- Prove the absence of run-time errors in software
  - 25 checks (non initialized pointers, out of bound arrays, …)
- Identify software defects via static analysis
  - 59 checks (memory leak, race condition, deadlock, …)
- Takes advantage of MATLAB platform

Optional

Required

Product stack
Test the Code on the Production Processor:
Simulink with Processor-in-the-loop

Execution
• Host/Target
• Nonreal-time
Test the ECU with a real-time environment

*Simulink Real-Time*

- MathWorks Simulink Real-Time
- MathWorks Instrumentation
- MathWorks real-time kernel
- FPGA-based software and hardware solutions
- Speedgoat real-time target machines
- Speedgoat I/O modules and protocol support
- Speedgoat tools and driver library
Agenda

- Benefits of Model-Based Design
- Verification at Model level
- Code generation
- Verification at Code level
- Certification process
Development Processes for High-Integrity Applications

- High integrity applications development follows standards and guidelines

- Standards and Guidelines have objectives for development process activities
  - Impose additional constraints on development
  - Require creation of additional artifacts
  - Require more thorough verification, validation and testing activities

- Require evidence that the objectives were met.
Automated Certification Process

- **Requirements**
  - Requirements Management Interface
  - System Design Description

- **Models**
  - MATLAB Simulink Stateflow
  - Model Advisor
  - Model Coverage
  - Processor-in-the-loop
  - Code Coverage

- **Source Code**
  - Embedded Coder
  - Polyspace Bug Finder & Code Prover
  - Simulink Code Inspector
  - SLDV Test Generation
  - Processor-in-the-loop
  - Code Coverage

- **Object Code**
  - Compiler

**Standards**
- IEC 61508
- IEC 62304
- ISO 26262
- EN 50128
- DO 178C
- DO 330/331
- DO 278
- DO 254
Key takeaways

- Faster development from requirements to code
- Better quality through continuous verification
- Stronger confidence with the support of international standards
Products cited

- **Model Verification**
  - Simulink Verification & Validation
  - Simulink Design Verifier

- **Code Generation**
  - Embedded Coder
  - HDL Coder
  - PLC Coder

- **Code Verification**
  - Polyspace Bug Finder
  - Polyspace Code Prover
  - Simulink Real-Time

- **Certification process**
  - DO Qualification Kit (for DO-178)
  - IEC Certification Kit (for ISO 26262 and IEC 61508)
Questions?