MATLAB EXPO 2019

Planning Simulink Model Architecture and Modeling Patterns for ISO 26262 Compliance

Dave Hoadley

ISO 26262 “Road Vehicles - Functional Safety”

- ISO 26262 is a functional safety standard for road vehicles
- MathWorks has seen an increased interest in ISO 26262 compliant workflows
  - Increase in System Complexity
  - Demand from ADAS and AD related applications
- ISO 26262 facilitates modern software engineering concepts

Challenges with ISO 26262

- Do I have an ISO 26262 compliant **workflow**?
- How to **efficiently** reach unit testing coverage criteria?
- How to achieve **Freedom from Interference**?
- Can we use **AUTOSAR** and meet ISO 26262 at the same time?
- Is **Simulink suitable for use** for ISO 26262?

ISO 26262-6:2018 notes Simulink and Stateflow as Suitable for Software Architecture, Design and as basis for Code Generation

Table 5 — Notations for software unit design

Notations		ASIL			
		A	B	C	D
1a	Natural language[a]	++	++	++	++
1b	Informal notations	++	++	+	+
1c	Semi-formal notations[b]	+	+	++	++
1d	Formal notations	+	+	+	+

[a] Natural language can complement the use of notations for example where some topics are more readily expressed in natural language or provide an explanation and rationale for decisions captured in the notations.

EXAMPLE To avoid possible ambiguity of natural language when designing complex elements, a combination of an activity diagram with natural language can be used.

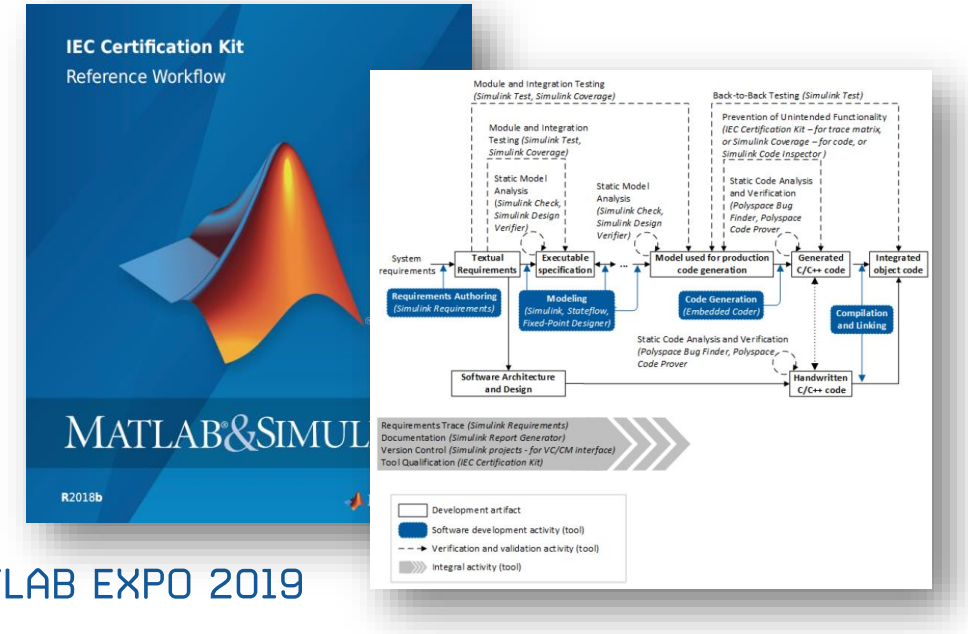
[b] Semi-formal notations can include pseudocode or modelling with UML®, SysML®, Simulink® or Stateflow®.

NOTE UML®, SysML®, Simulink® and Stateflow® are examples of suitable products available commercially. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO of these products.

NOTE In the case of model-based development with automatic code generation, the methods for representing the software unit design are applied to the model which serves as the basis for the code generation.

MathWorks Support

- IEC Certification Kit
  - Model-Based Design Reference Workflow
    - Proven in use
  - Tool Qualification Package
    - Software Tool Criteria Evaluation Report
    - Software Tool Qualification
    - Tool Validation Suite



LG Chem Develops ISO 26262 ASIL C AUTOSAR-compliant Software for a Hybrid Vehicle Battery Management System for the Volvo XC90

"*Model-Based Design enables us to **increase component reuse, reduce manual coding, improve communication with our customers**, and ultimately deliver **higher-quality** BMS in **less time**.*"

- Won Tae Joe, LG Chem

MathWorks Support for ISO 26262
Certification Kit

- Applicable Model-Based Design Tools and Processes

IEC Certification Kit
Model-Based Design for ISO 26262

Methods		ASIL A	ASIL B	ASIL C	ASIL D	Applicable Model-Based Design Tools and Processes	Comments
1b	Generation and analysis of equivalence classes	+	++	++	++	Simulink Design Verifier – Test case generation	The analysis of equivalence classes can be based on the interfaces of the model. Automatic test case generation in combination with Test Objective blocks can be used to generate test cases and test sequences for given equivalence classes.
1c	Analysis of boundary values	+	++	++	++	Simulink Design Verifier – Test case generation	The analysis of boundary values can be based on the interfaces of the model. Automatic test case generation in combination with Test Objective blocks can be used to generate test cases and test sequences for given boundary values.
1d	Error guessing	+	+	+	+		

Table 12 – Structural Coverage Metrics at the Software Unit Level

Methods		ASIL A	ASIL B	ASIL C	ASIL D	Applicable Model-Based Design Tools and Processes	Comments
1a	Statement coverage	++	++	+	+	Simulink Coverage – Code coverage analysis	During SIL execution, Simulink Coverage can measure the statement coverage of the generated code.
1b	Branch coverage	+	++	++	++	Simulink Coverage – Model coverage analysis Simulink Coverage – Code coverage analysis Simulink Design Verifier – Test case generation	During model testing, Simulink Coverage can collect decision coverage (also known as branch coverage) at the model level. During SIL execution, Simulink Coverage can measure the decision coverage of the generated code. Simulink Design Verifier can generate test cases that satisfy decision coverage at the model level.
1c	MC/DC (Modified Condition/Decision Coverage)	+	+	+	++	Simulink Coverage – Model coverage analysis Simulink Design Verifier – Test case generation	During model testing, Simulink Coverage verification can collect MC/DC coverage at the model level. Simulink Design Verifier can be used to generate test cases that satisfy MC/DC coverage at the model level.
						Simulink Coverage – Code coverage analysis	During SIL and PIL execution, Simulink Coverage can measure MC/DC coverage of the generated code.

Mapping between ISO requirements to Model-Based Design toolchain

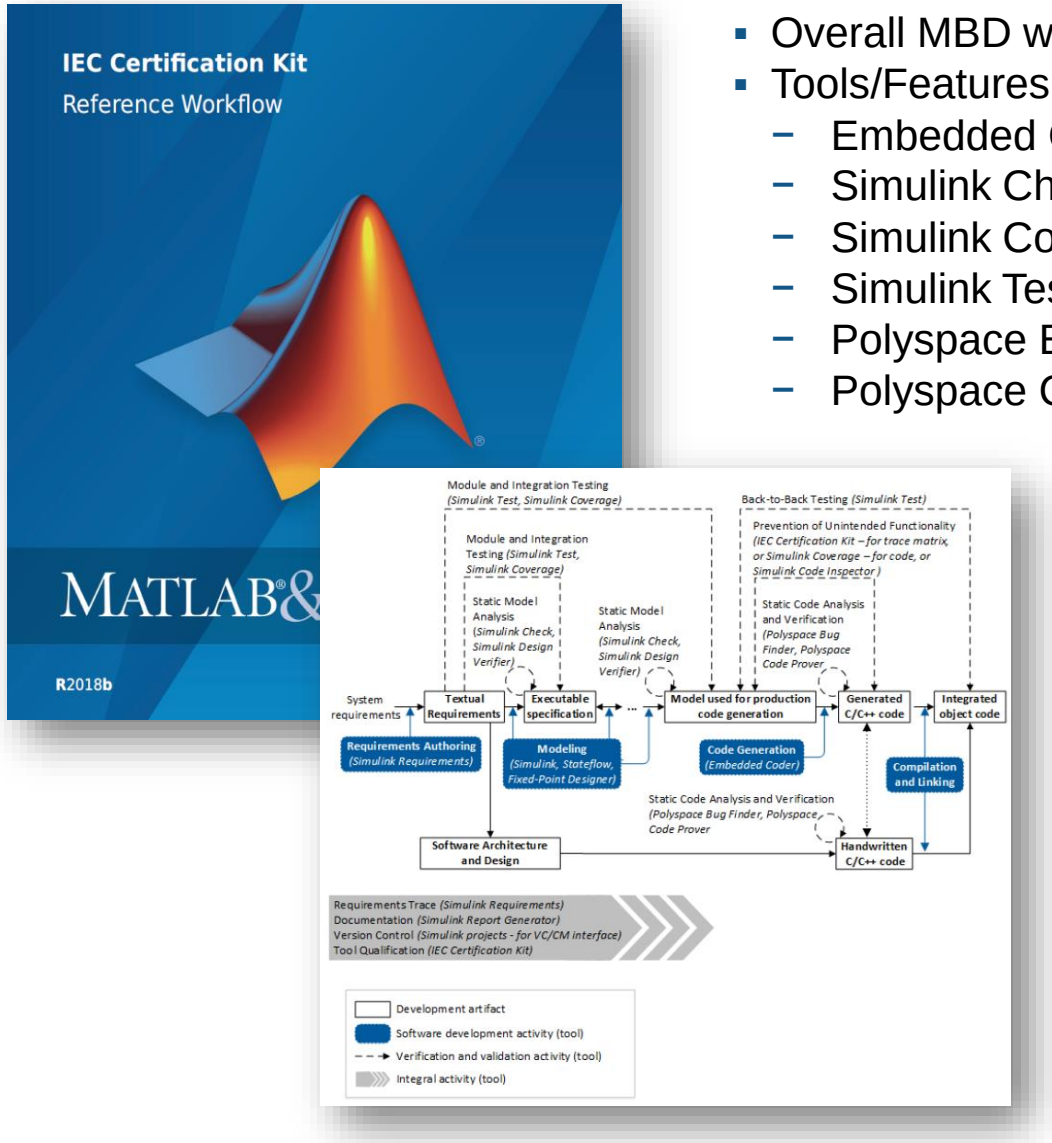
Methods		ASIL A	ASIL B	ASIL C	ASIL D	Applicable Model-Based Design Tools and Processes	Comments
1c	MC/DC (Modified Condition/Decision Coverage)	+	+	+	++	Simulink Coverage – Model coverage analysis Simulink Design Verifier – Test case generation	During model testing, Simulink Coverage verification can collect MC/DC coverage at the model level. Simulink Design Verifier can be used to generate test cases that satisfy MC/DC coverage at the model level.
						Simulink Coverage – Code coverage analysis	During SIL and PIL execution, Simulink Coverage can measure MC/DC coverage of the generated code.

ISO Requirement — Model-Based Design

MathWorks Support for ISO 26262
Certification Kit

- Applicable Model-Based Design Tools and Processes

- Model-Based Design reference workflow



- Overall MBD workflow
- Tools/Features:
  - Embedded Coder
  - Simulink Check
  - Simulink Coverage
  - Simulink Test
  - Polyspace Bug Finder
  - Polyspace Code Prover

MathWorks Support for ISO 26262
Certification Kit

- Applicable Model-Based Design Tools and Processes
- Model-Based Design reference workflow
- Tool Qualification Package
  - Software Tool Criteria Evaluation Report
  - Software Tool Qualification Report

IEC Certification Kit
Simulink® Test™
ISO 26262 Tool Qualification Package

R2018b

Tool Confidence Level determination
+ Other qualification artifacts

Potential Malfunction or Erroneous Output	Use Cases	TI	Justification for TI	Prevention / Detection Measures	TD	Justification for TD	TCL
[SLTEST_E1] Incorrect behavior of test harness	[SLTEST _UC1] [SLTEST _UC2]	TI2	Incorrect behavior of test harness could prevent errors in an object under test from being detected.	[SLTEST_M1] Requirements-based testing	TD1	The test cases and expected results are derived from requirements independen of the model under test and the test environment. The independence provides a high degree of confidence that errors will be detected using the actual results from the model under tes in the test environment	TCL1
[SLTEST_E2] Incorrect run of test procedure	[SLTEST _UC1] [SLTEST _UC2]	TI2	Incorrect run of test procedure could prevent errors in an object under test from being detected.	[SLTEST_M1] Requirements-based testing	TD1	Requirements-based testing will detect incorrect run of test procedure, see TD justification for [SLTEST_E1]	TCL1
[SLTEST_E3] Erroneous assessment of test results – passed test indicated as failed	[SLTEST _UC3]	TI1	Nuisance only, failed tests have to be manually reviewed and explained by user	-	-	-	TCL1
		TI2		None	TD3	-	TCL3

Modeling Best Practices for ISO 26262

- Architecture
- Signal Routing
- Data Definition
- Code Generation Configuration

(Excerpts from our white paper

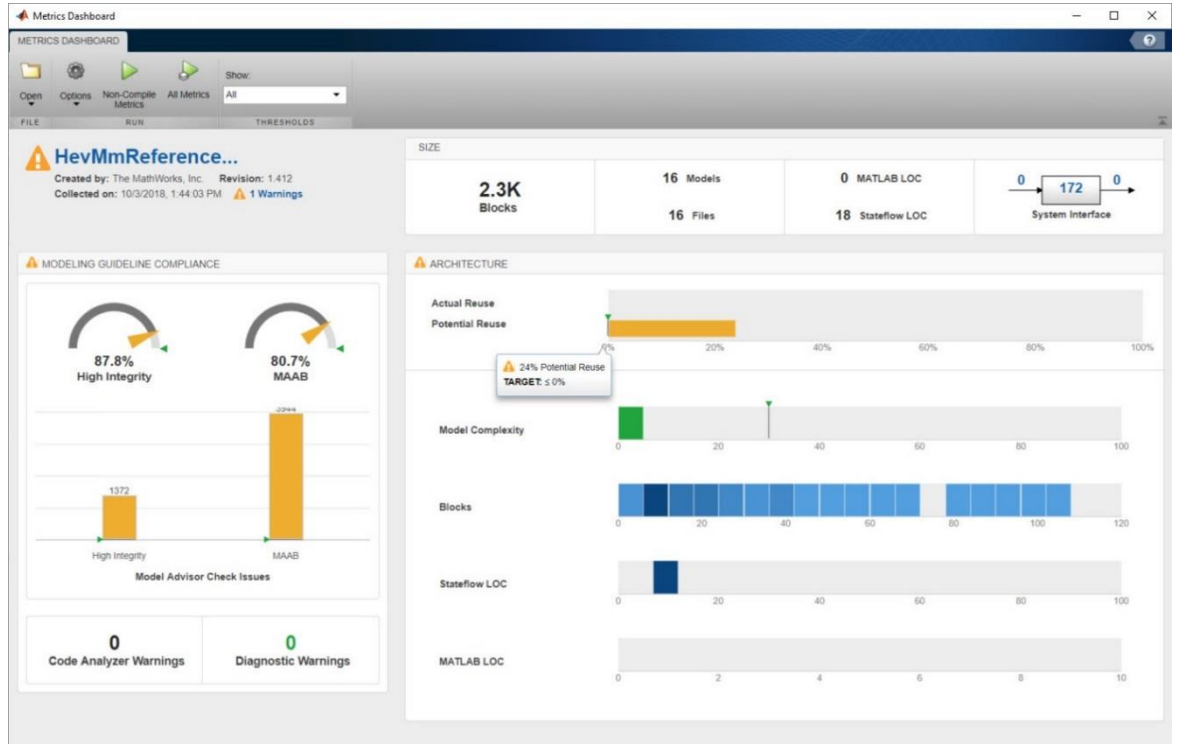
– Please request www.mathworks.com/services/consulting/contact.html)

Use Model Metrics to Monitor Unit Complexity

Architecture

- Issues:
  - Model verification gets increasingly difficult
  - Unable to efficiently achieve unit coverage
- Best Practice:
  - Monitor complexity metrics
    - Interfaces
    - Reusable libraries
    - Cyclomatic complexity (<=30)*
    - Number of elements (<500)*
    - Style and standards conformance
- Reference:
  - *Paper: Model Quality Objectives
    - Authors: Jérôme Bouquet(Renault), Stéphane Faure(Valeo), Florent Fève(Valeo), Ursula Garcia(Bosch), François Guérin(MathWorks), Thierry Hubert(PSA), Florian Levy(Renault), Stéphane Louvet(Bosch), Patrick Munier(MathWorks), Pierre-Nicolas Paton(Delphi), Alain Spiewek(Delphi), and Yves Touzeau(Renault)

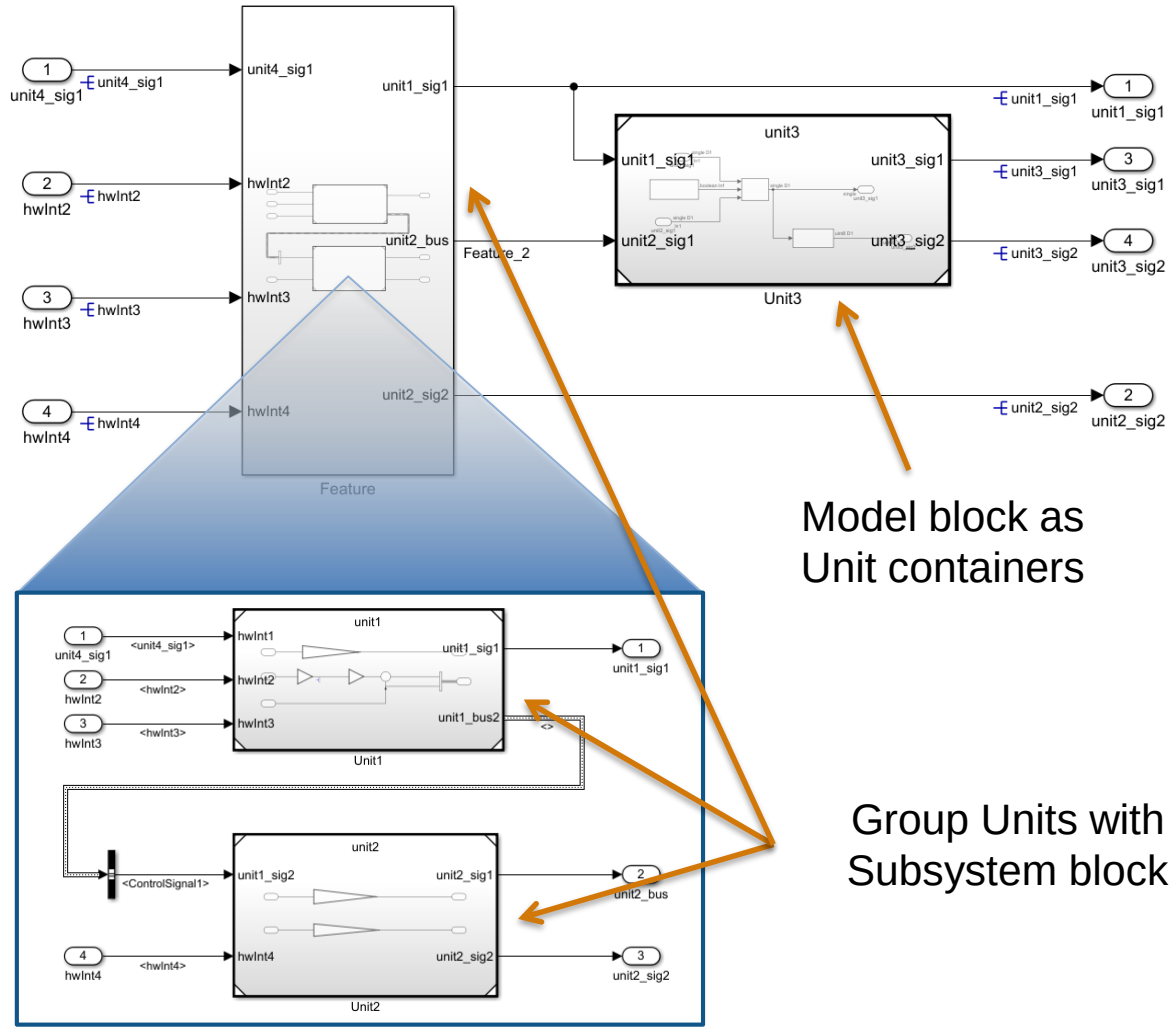
Model Metric Dashboard



Use Model Reference for Unit Level Model

Simulink Architecture

- Issues:
  - Poor modularity of algorithm (reuse)
  - Unable to preform unit level testing
  - Configuration Management difficulties
  - Unable to achieve Freedom from Interference
- Best Practice
  - Use Model Reference for unit level model
  - Group units to form functional hierarchy (features/components) with virtual Subsystems

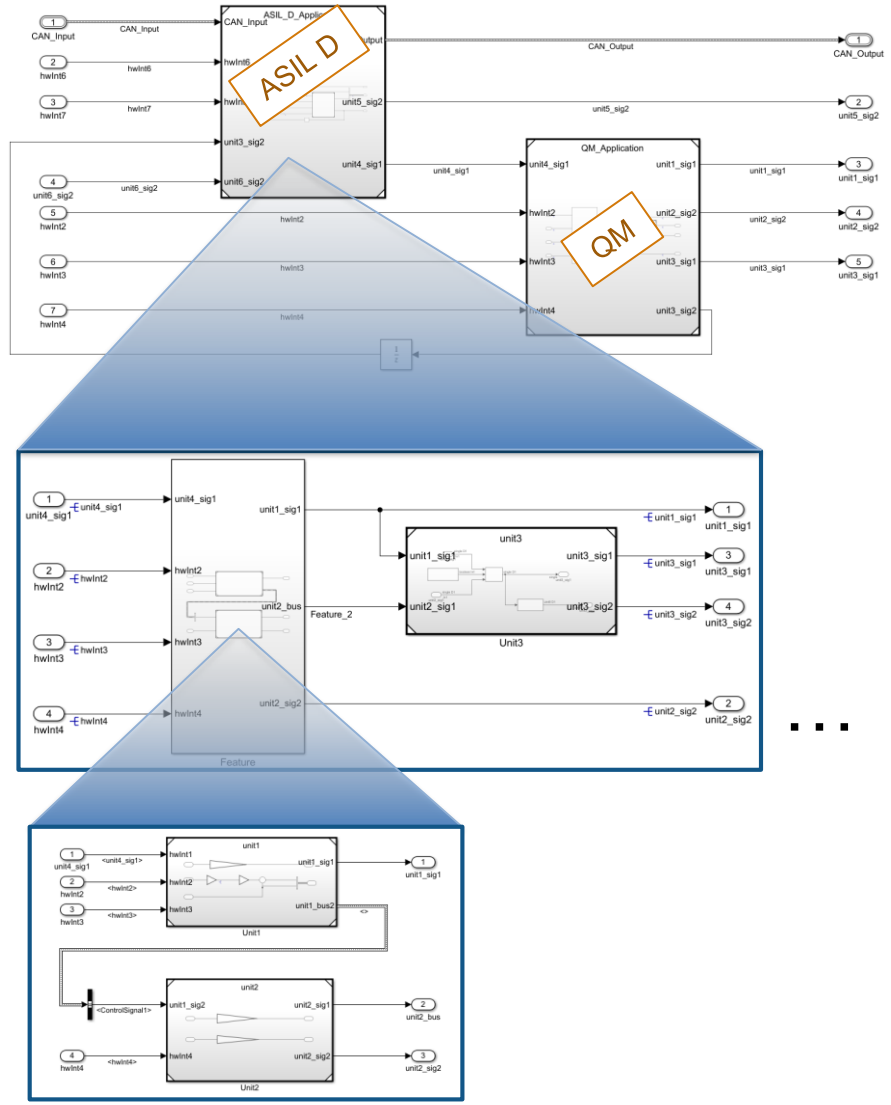


Split ASIL and QM Levels at Top Level of Control Model

Simulink Architecture

- Issues:
  - Difficulty in achieving Freedom from Interference
  - Complexity in code integration
- Best Practice:
  - Code generation should be done at as high as level as possible.

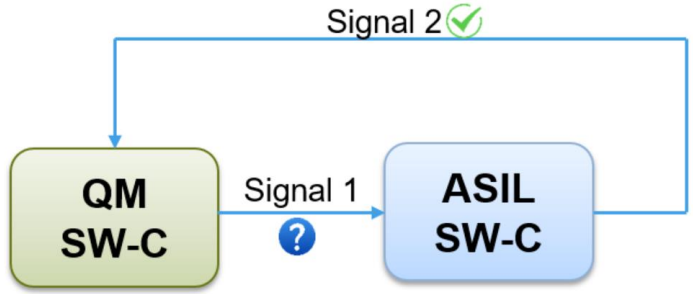
Model Hierarchy	Modeling Pattern
Top level (ASIL / QM)	Model Reference
Integration	Subsystem (multiple layers)
Unit	Model Reference



Data Protection Between ASIL and QM Levels

Code Generation Configuration

- Issues:
  - How to provide signal protection between ASIL and QM functions?

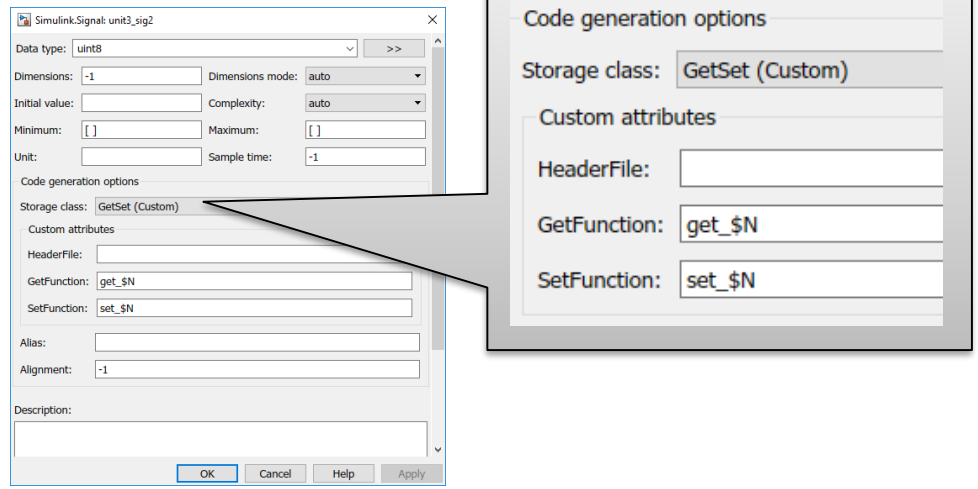


Data Protection Between ASIL and QM Levels

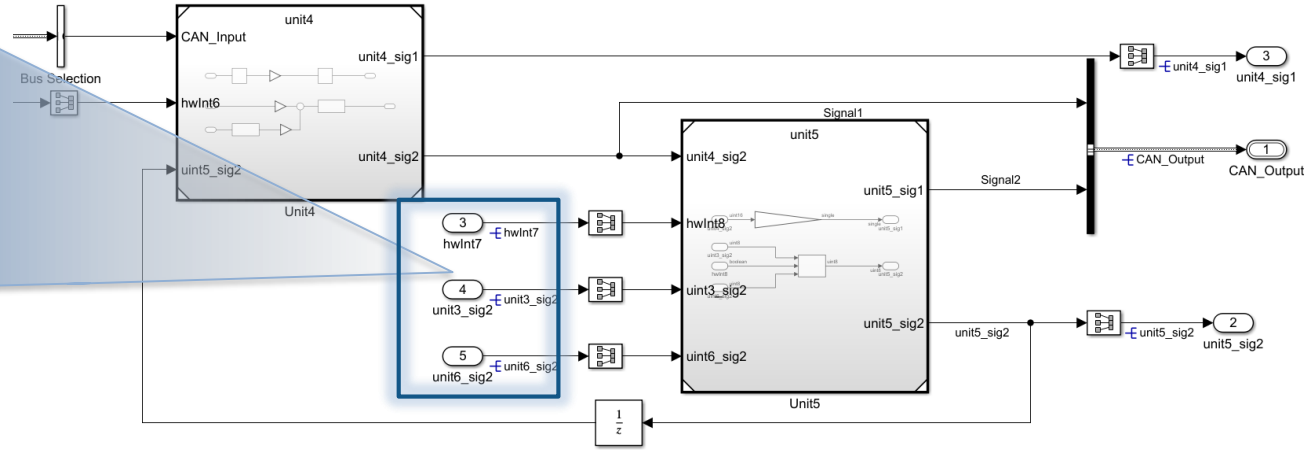
Code Generation Configuration

- Issues:
  - How to provide signal protection between ASIL and QM functions?
- Best Practice
  - Use Get/Set storage class for signals between ASIL and QM levels

Get/Set Storage Class



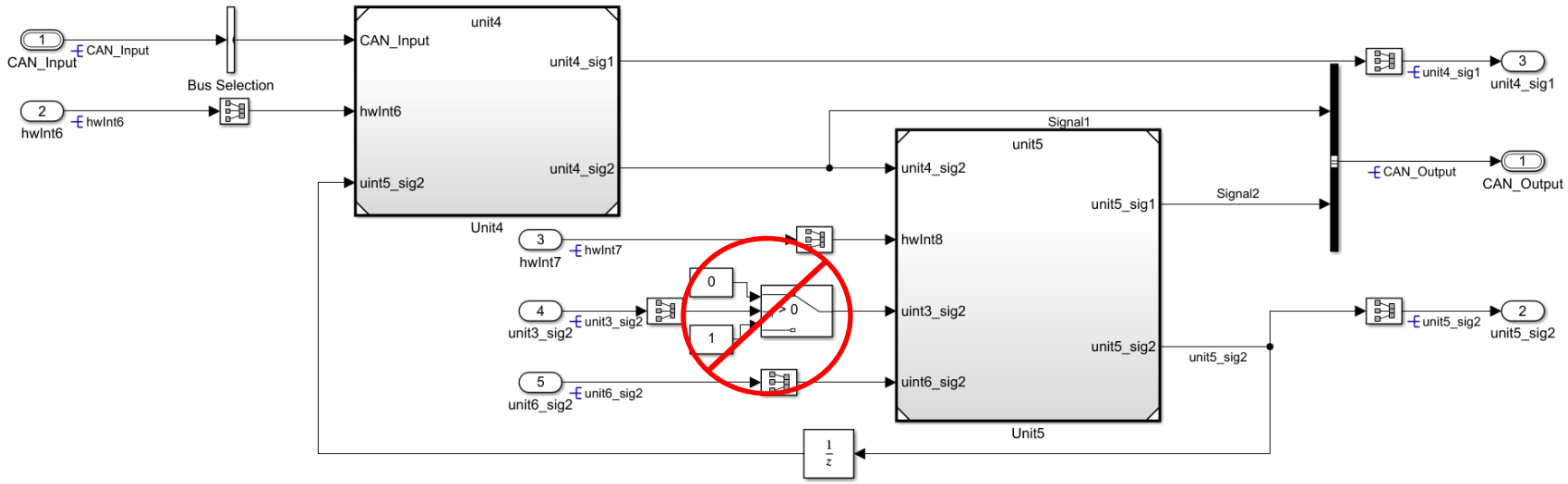

```
106  /* SignalConversion: '<Root>/Signal Conversion2' incorporates:
107   *  Inport: '<Root>/hwInt7'
108   */
109  rtb_SignalConversion2 = get_hwInt7();
110
111  /* SignalConversion: '<Root>/Signal Conversion3' incorporates:
112   *  Inport: '<Root>/unit3_sig2'
113   */
114  rtb_UnitDelay = get_unit3_sig2();
115
116  /* SignalConversion: '<Root>/Signal Conversion4' incorporates:
117   *  Inport: '<Root>/unit6_sig2'
118   */
119  rtb_SignalConversion4 = get_unit6_sig2();
120
121  /* ModelReference: '<Root>/Unit5' incorporates:
122   *  UnitDelay: '<Root>/Unit Delay'
123   */
```



Eliminate Algorithm Content at Integration Level

Architecture

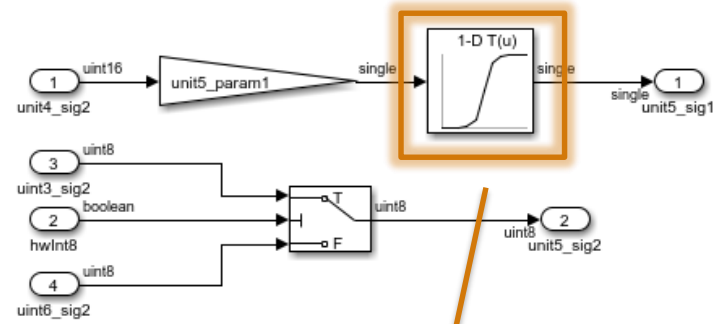
- Issues:
  - Complexity in integration level testing
  - Difficult tracing of requirement ⇔ design ⇔ test
- Best Practice:
  - Ensure only virtual blocks are at the integration level
  - Reference (MAAB/JMAAB): db_0143: Similar block types on the model levels

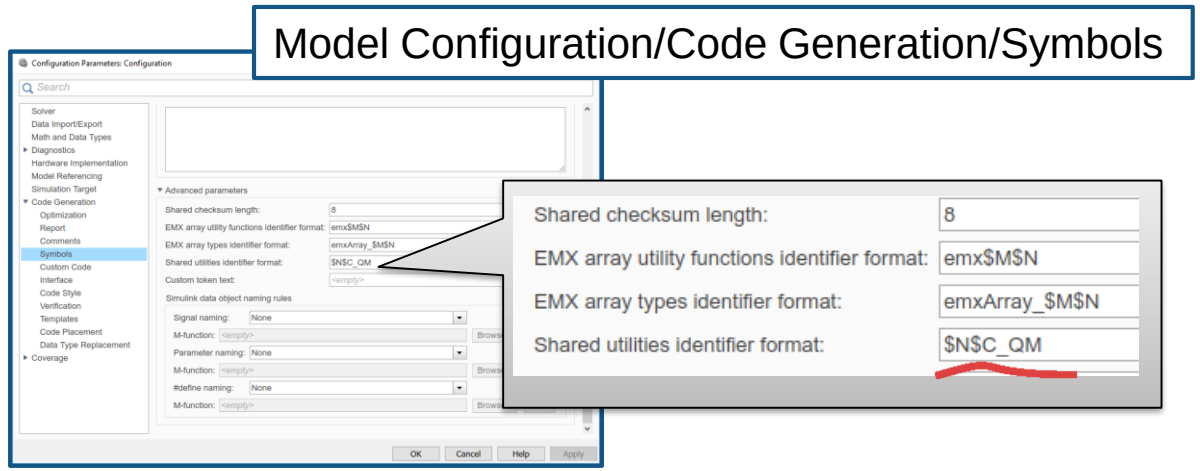


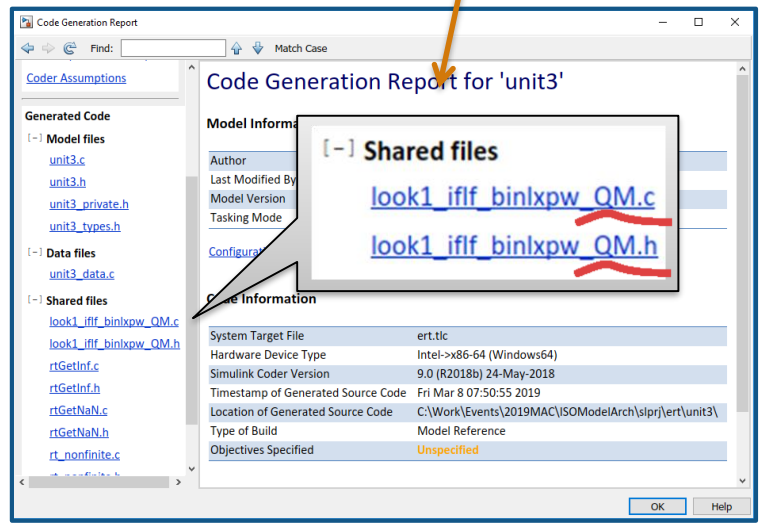
Use Different Name Token for Shared Utility

Code Generation Configuration

- Issues:
  - ASIL and QM level uses the same shared utility code
- Best Practice:
  - Configure Shared Utility Identifier



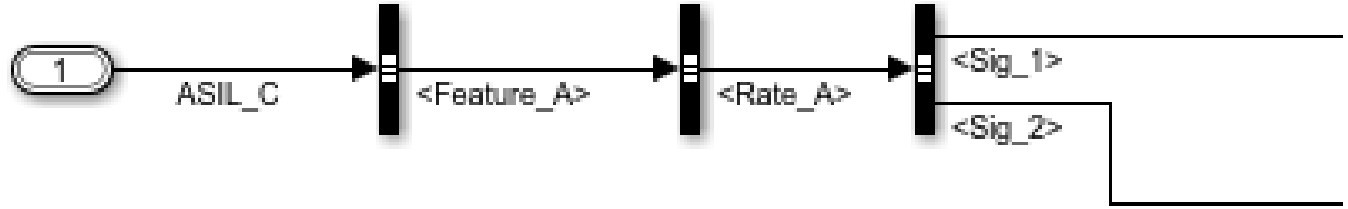




Design Bus Hierarchy

Signal Routing

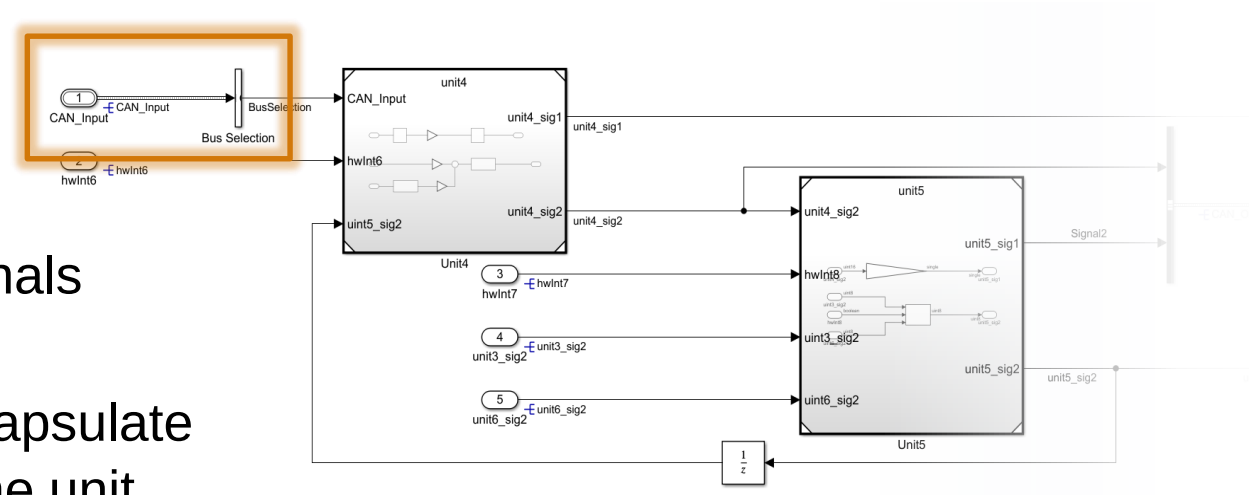
- Issues:
  - Inefficient bus segmentation
  - Inconsistent bus grouping by developers
  - Modeling difficulty from splitting and recreating bus signals
- Best Practice:
  - Bus hierarchy should be designed as a function of ASIL levels, QM, and rates at a minimum.



Pass Only Used Signal into Unit

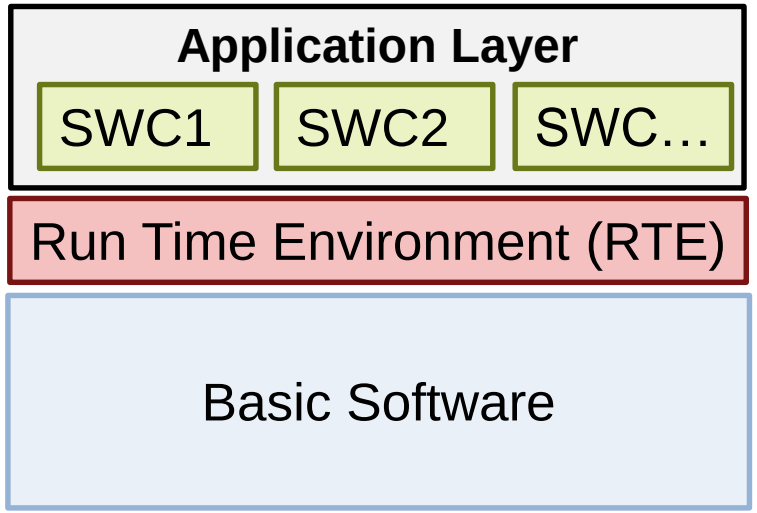
Signal Routing

- Issues:
  - Hundreds/Thousands input signals causing difficulties in verification flow
- Best Practice:
  - Use Bus Selector to send only used signals into Unit
  - Add additional virtual Subsystem to encapsulate the Bus manipulation before and after the unit



AUTOSAR Implications

- AUTOSAR
  - adds complexity due to additional tool ecosystem
  - but makes some things simpler
    - Get/Set function would be implemented using Send/Receiver port with RTE protection
- Best Practices discused are consistent with our AUTOSAR Blockset
- Reference Workflow shown in IEC Certification Kit supports AUTOSAR
  - Simulation
  - Code generation
  - Verification



Layered AUTOSAR Architecture

MathWorks Support

ISO 26262 Consulting Services

- Process establishment
  - Development Processes
  - Verification process
  - Gap analysis
- Tool qualification support
  - Analyze customer specific tools
  - Provide guidance on tool qualification activities

ISO 26262 Process Deployment Advisory Service

MathWorks Consulting Services works with you to migrate your existing process—whether based on manual methods or Model-Based Design—to a process framework for using Model-Based Design with ISO 26262. Customized to your specific environment, tools, and application, the ISO 26262 Process Deployment Advisory Service identifies gaps in your current processes, develops a road map to a more optimized process framework using Model-Based Design, and works with you to deploy that road map.

"We leveraged MathWorks consultants to apply Model-Based Design for ISO 26262 on our new Integrated Restraints and Braking Controller (IRBC) developed with Simulink, Stateflow, Simulink Design Verifier, and Embedded Coder for production code generation and verification."

Rich Rakes, Lead Engineer, Autoliv

Summary

- Modeling Best Practice for ISO 26262
  - Use Model Reference for Unit Level Model
  - Split ASIL and QM Levels at Top Level of Model
  - Eliminate Algorithm Content at Integration Level
  - Use Model Metrics to Monitor Unit Complexity
  - Pass Only Used Signal into Unit
  - Design Bus Hierarchy
  - Modeling Construct for Data
  - Data Protection Between ASIL and QM Levels
  - Partition Different ASIL levels and QM to Separate Memory Section
  - Use Different Name Token for Shared Utility
  - AUTOSAR Implications
- Further information?
  - Please see mathworks.com/services/consulting/proven-solutions/iso26262.html
  - Contact me dhoadley@mathworks.com
  - Stop by the ISO 26262 table