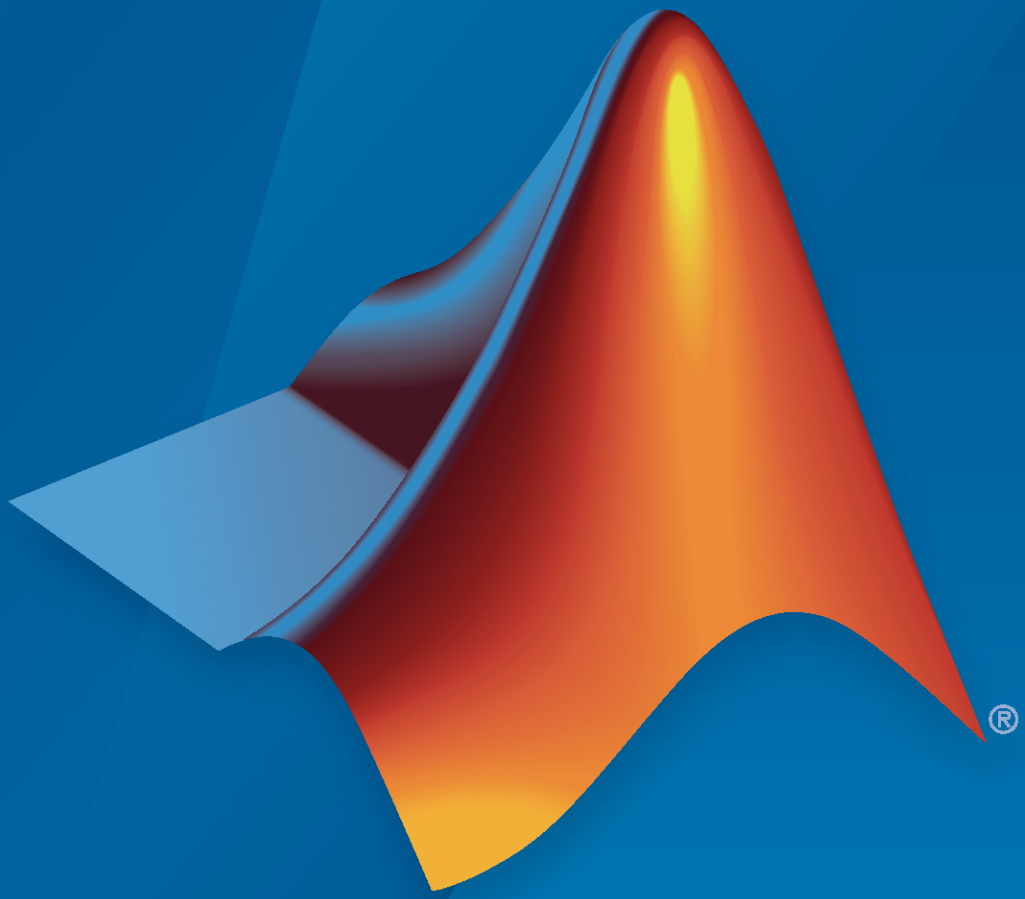
DO Qualification Kit

Model-Based Design Workflow for DO-254



MATLAB® & SIMULINK®

R2022b

MathWorks®

How to Contact MathWorks

Latest news: `www.mathworks.com`

Sales and services: `www.mathworks.com/sales_and_services`

User community: `www.mathworks.com/matlabcentral`

Technical support: `www.mathworks.com/support/contact_us`

Phone: 508-647-7000

The MathWorks, Inc.
1 Apple Hill Drive
Natick, MA 01760-2098

DO Qualification Kit Model-Based Design Workflow for DO-254

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See `www.mathworks.com/trademarks` for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

MathWorks products are protected by one or more U.S. patents. Please see `www.mathworks.com/patents` for more information.

Revision History

September 2018	Online only	New for DO Qualification Kit Version 3.6 (Release 2018b)
March 2019	Online only	Revised for DO Qualification Kit Version 3.7 (Release 2019a)
September 2019	Online only	Revised for DO Qualification Kit Version 3.8 (Release 2019b)
March 2020	Online only	Revised for DO Qualification Kit Version 3.9 (Release 2020a)
September 2020	Online only	Revised for DO Qualification Kit Version 3.10 (Release 2020b)
March 2021	Online only	Revised for DO Qualification Kit Version 3.11 (Release 2021a)
September 2021	Online only	Revised for DO Qualification Kit Version 3.12 (Release 2021b)
March 2022	Online only	Revised for DO Qualification Kit Version 3.13 (Release 2022a)
September 2022	Online only	Revised for DO Qualification Kit Version 3.14 (Release 2022b)

Contents

1 Tool Description

2 DO-254 Hardware Life Cycle

A Acronyms

B References

Tool Description

Tools Overview

This section describes the high-level architecture of the development and verification tools that you can apply in a Model-Based Design workflow for DO-254. The independence aspects of the various tools and how errors in the tools can be detected are described. There are two types of tools used in the workflow, development tools and verification tools.

Development tools:

- Simulink®
- Stateflow®
- Fixed-Point Designer™
- MATLAB®
- HDL Coder™

Verification tools:

- MATLAB Report Generator™
- Simulink Report Generator
- Simulink
- Simulink Design Verifier™
- Simulink Check™
- Simulink Coverage™
- Simulink Test™
- HDL Verifier™

Independence of the Tools

Simulink, Stateflow, and Fixed-Point Designer are separate tools for the development of models. Simulink can be used without Stateflow or Fixed-Point Designer, but when Stateflow or Fixed-Point Designer is used, Simulink is required.

Simulink and Stateflow are tightly integrated and are not independent of each other. There is no requirement for these tools to be independent because they are used together as part of the development of the hardware design.

Simulink and Fixed-Point Designer are tightly integrated and are not independent of each other. There is no requirement for these tools to be independent because they are used together as part of the development of the hardware design.

The Simulink API provides an interface to retrieve data from the model for those tools that cannot access the in-memory data directly. For example, you can use the MATLAB command `get_param` to get data from the model or use the `set_param` command to set a parameter in the model.

See the workflow section of this document, “Hardware Design Process” on page 2-7, which includes the following objectives for the use of Simulink, Stateflow, and Fixed-Point Designer:

- The hardware item conceptual design is developed and is consistent with its requirements.
- Derived requirements produced are sent back to the requirements capture or other appropriate process.
- Requirement omissions and errors are provided to the appropriate process for resolution.

The MATLAB Report Generator and Simulink Report Generator are two separate tools. The MATLAB Report Generator is a prerequisite for the Simulink Report Generator. Simulink Report Generator provides components for reporting on Simulink and Stateflow models and is integrated with the MATLAB Report Generator. These components use the Simulink API to read data from the model loaded in memory. The components cannot write or modify data in the model. For example, when generating the System Design Description document, the report generation components only read data from the model. The System Design Description includes requirements traceability links that you can insert into the models by using Requirements Toolbox™.

See the workflow sections of this document, “Hardware Design Process” on page 2-7 and “Validation and Verification Process” on page 2-13, which include the following objectives for the use of MATLAB Report Generator and Simulink Report Generator:

- Hardware Design Process
  - The hardware item conceptual design is developed and is consistent with its requirements.
  - Derived requirements produced are sent back to the requirements capture or other appropriate process.
- Validation and Verification Process
  - Evidence is provided that the hardware implementation meets the requirements.
  - Traceability is established between hardware requirements, the implementation, and the verification procedures and results.

You can use Requirements Toolbox to author, analyze, and manage requirements within Simulink. You can create rich text requirements with custom attributes and link these requirements to designs,

code, and tests. You can also import requirements from external sources. Use Requirements Toolbox to view requirements and design together and establish links by using drag-and-drop functionality. Use Requirements Toolbox to annotate diagrams with requirements content, analyze requirements traceability, and navigate between requirements, designs, generated code, and tests. You can set up notifications to alert you when requirements change.

See the workflow sections of this document, “Hardware Design Process” on page 2-7 and “Validation and Verification Process” on page 2-13, which include the following objectives for the use of Requirements Toolbox:

- Hardware Design Process
  - Requirements are identified, defined, and documented
  - Derived requirements produced are sent back to the appropriate process.
  - Requirement omissions and errors are provided to the appropriate process for resolution.
  - The hardware item conceptual design is developed and is consistent with its requirements.
  - Derived requirements produced are sent back to the requirements capture.
- Validation and Verification Process
  - Derived hardware requirements against which the hardware is to be verified are correct and complete.
  - Derived requirements are evaluated for their impact on safety.
  - Omissions and errors are sent back to the appropriate processes for resolution.
  - Traceability is established between hardware requirements, the implementation, and the verification procedures and results.

Simulink Design Verifier is a separate tool with these capabilities: design error detection, property proving, and test case generation. Simulink Design Verifier contains formal analysis engines that operate on an internal representation derived from but in a different form than the Simulink model loaded in memory. By using design error detection, you can find specific design errors in the model such as divide-by-zero or numeric overflows. By using property proving, you can prove that user-defined properties are in conjunction with user-defined assumptions. The formal analysis engines are separate and independent of Simulink and Stateflow and do not involve simulation of the model. Simulink Design Verifier can generate test cases based on the model that you can use to verify that the executable object code complies with the model. The basis for the test cases can be a combination of user-defined constraints, coverage criteria for blocks in the model, and user-defined test objectives. HDL Coder ignores the constraint blocks, coverage criteria, and test objective blocks and, therefore, are independent of the coding process. To verify the code by using the generated test cases, you must run the test cases on the model to product the expected results for the code. You can access the completeness of test cases by using the coverage tool and access the expected results through review of the simulation results.

See the workflow section of this document, “Validation and Verification Process” on page 2-13, which includes the following objectives for the use of Simulink Design Verifier:

- Evidence is provided that the hardware implementation meets the requirements.

The Model Advisor checks are provided in several different products: Simulink, HDL Coder, Simulink Code Inspector™, Simulink Check, and Simulink Control Design™. The basic core implementation of Model Advisor checks is done through an engine that uses MATLAB functions and is independent of Simulink, Stateflow, and HDL Coder. The Model Advisor uses the Simulink API to read data from the

model loaded in memory. The Model Advisor can fix issues detected by checks, but you must initiate the fixes and resave the model. You can then rerun the checks to verify the fixes. For custom checks, it is your responsibility not to allow those checks to modify the model.

See the workflow section of this document, “Validation and Verification Process” on page 2-13, which includes the following objectives for the use of Model Advisor:

- Evidence is provided that the hardware implementation meets the requirements.

The coverage capability is provided as part of Simulink Coverage. Model coverage instruments the model before simulation, and then evaluates the coverage criteria as simulation progresses. Simulink Coverage can also merge multiple simulations into a combined coverage report. You can run simulations with coverage enabled and disabled to ensure that there has been no effect on behavior of the model due to the instrumentation.

See the workflow sections of this document, “Hardware Design Process” on page 2-7 and “Validation and Verification Process” on page 2-13, which include the following objectives for the use of Simulink Coverage:

- Hardware Design Process
  - Requirement omissions and errors are provided to the appropriate process for resolution.
- Validation and Verification Process
  - Evidence is provided that the hardware implementation meets the requirements.

Simulink Test is a separate tool that you can use to execute simulations in a batch model and check actual results against expected results. It also provides the capability to author test cases manually or to import test cases in other formats, such as Excel® spreadsheets. Because you manually develop the test cases and expected results, they are independent of the model and HDL code.

See the workflow section of this document, “Validation and Verification Process” on page 2-13, which includes the following objectives for the use of Simulink Test:

- Evidence is provided that the hardware implementation meets the requirements.
- Traceability is established between hardware requirements, the implementation, and the verification procedures and results.

HDL Coder generates portable, synthesizable VHDL® and Verilog® code from MATLAB functions, Simulink models, and Stateflow charts. You can use generated HDL code for Field-Programmable Gate Array (FPGA) programming or Application-Specific Integrated Circuit (ASIC) prototyping and design.

See the workflow sections of this document, “Hardware Design Process” on page 2-7 and “Validation and Verification Process” on page 2-13, which include the following objectives for the use of HDL Coder:

- Hardware Design Process
  - The detailed design is developed from the hardware item requirements and conceptual design data.
  - Derived requirements are sent back to the conceptual design or other appropriate process.
  - Requirement omissions and errors are provided to the appropriate process for resolution.

- Validation and Verification Process
  - Evidence is provided that the hardware implementation meets the requirements.
  - Traceability is established between hardware requirements, the implementation, and the verification procedures and results.

HDL Verifier generates test benches for VHDL and Verilog design verification. You can use MATLAB or Simulink to simulate your design, and then analyze its response by using HDL cosimulation or FPGA-in-the-loop with Xilinx® and Altera® FPGA boards. This approach eliminates authoring standalone Verilog or VHDL test benches.

HDL Verifier also generates components that reuse MATLAB and Simulink models natively in simulators from Cadence®, Mentor Graphics®, and Synopsys®. You can use these components as verification checker models or as stimuli in more complex, test-bench environments such as those that use the Universal Verification Methodology (UVM).

See the workflow sections of this document, “Hardware Design Process” on page 2-7 and “Validation and Verification Process” on page 2-13, which include the following objectives for the use of HDL Verifier:

- Hardware Design Process
  - Requirement omissions and errors are provided to the appropriate process for resolution.
- Validation and Verification Process
  - Evidence is provided that the hardware implementation meets the requirements.

Model and HDL Code Development and Verification

In a workflow where code is generated from the Simulink and Stateflow models, the models are considered to be the hardware conceptual design and architecture. The actual design is the compiled model in memory, as interpreted by the Simulink engine, which is based on input from the model file and data files. Data files can include MATLAB or MAT files that load data into the MATLAB or model workspaces. The model file itself does not represent the hardware conceptual design because the model semantics are not fully included in that file. The model semantics are not complete until the model file has been loaded into memory and the Simulink engine has compiled the model. Some of the model semantics that are determined for the model at compile time, but are not included in the model file, consist of:

- Propagated sample times
- Propagated data types
- Propagated signal dimensions
- Propagated signal types
- Block execution order

The System Design Description report (SDD), which you can create by using Simulink Report Generator, provides a document that details the compiled-for-simulation in-memory representation of the model. The SDD report provides documentation of the hardware conceptual design.

Because the model and code verification activities can occur at different times or on different computers, you must check the consistency of the in-memory representations of the model. You can use an MD5 Checksum computation to check this consistency. The MD5 checksum is based on the in-memory representation of the model and includes data loaded into the workspace from external files that are used by the model. Tunable parameters are not included in the MD5 checksum. The MD5 Checksum value is automatically inserted into the Model Advisor, System Design Description reports. You can also use the Simulink API to access the MD5 Checksum and insert it into reports. A model version number and last saved date, which is updated each time a model is saved, are also included in the report. Externally loaded data does not affect the model version number and last saved date, which is why the MD5 Checksum is required to verify complete consistency of the in-memory representation. The System Design Description report also includes the workspace variables that the model uses at the time the report is generated.

Note The model checksum computation is platform-dependent.

Test Case Development

The DO-254 standard calls out four types of testing, all of which are based on the hardware requirements:

- Functional tests
- System bench tests
- System validation tests
- Aircraft tests

For the HDL code developed from models, the hardware functional test cases and expected results can be the same as the simulation cases and expected results. The test cases are developed from the hardware requirements document and are independent of Simulink, HDL Coder, and the hardware design tools used for the project. The test cases and expected result must also include robustness cases. You can execute these test cases by using the FPGA in-the-loop (FIL) capability in conjunction with the Simulink environment that is used as a test harness, or on a separate hardware test harness.

The conceptual design-based test cases and expected results are based on the models, which represent the conceptual hardware design. You can use Simulink Design Verifier to develop these test cases. Simulink Design Verifier uses the model as its primary input and can input coverage data. You can use model coverage as evidence that hardware requirements tests cover conceptual hardware design, in particular for logical decisions within the models, but also for lookup table data and signal range data. You can then use Simulink Design Verifier to generate tests for the remaining conceptual hardware design that is not covered by hardware requirements testing, such as for derived requirements. You can also insert signal constraints and user-defined test objectives within the models or in model test harnesses to complete the testing. Use test objectives on the inputs to a model to insert test data beyond normal ranges as a way to verify robustness, for example.

The system functional test cases and the aircraft test cases are typically developed manually based on the high-level hardware requirements. These test cases are executed on the final target in an environment independent of the modeling environment.

DO-254 Hardware Life Cycle

DO-254 Hardware Life Cycle Overview

The DO-254 hardware life cycle consists of these processes:

- Planning on page 2-5
- Hardware Design on page 2-7
- Validation and Verification on page 2-13
- Configuration Management on page 2-16
- Process Assurance on page 2-18
- Certification Liaison on page 2-19

The *DO-254, Design Assurance Guidance for Airborne Electronic Hardware* document summarizes the objectives that must be met for each of the life cycle stages. This document provides recommendations and tools that you can use for meeting these objectives in a Model-Based Design process.

Model-Based Design Workflow in DO-254

This section outlines a Model-Based Design workflow that addresses the development and verification activities in a DO-254 hardware life cycle.

Overview of the Model-Based Design Workflow: Hardware Development Activities

Figure 4 illustrates core hardware development artifacts and activities and artifacts. Solid, horizontal arrows indicate the succession of hardware development activities.

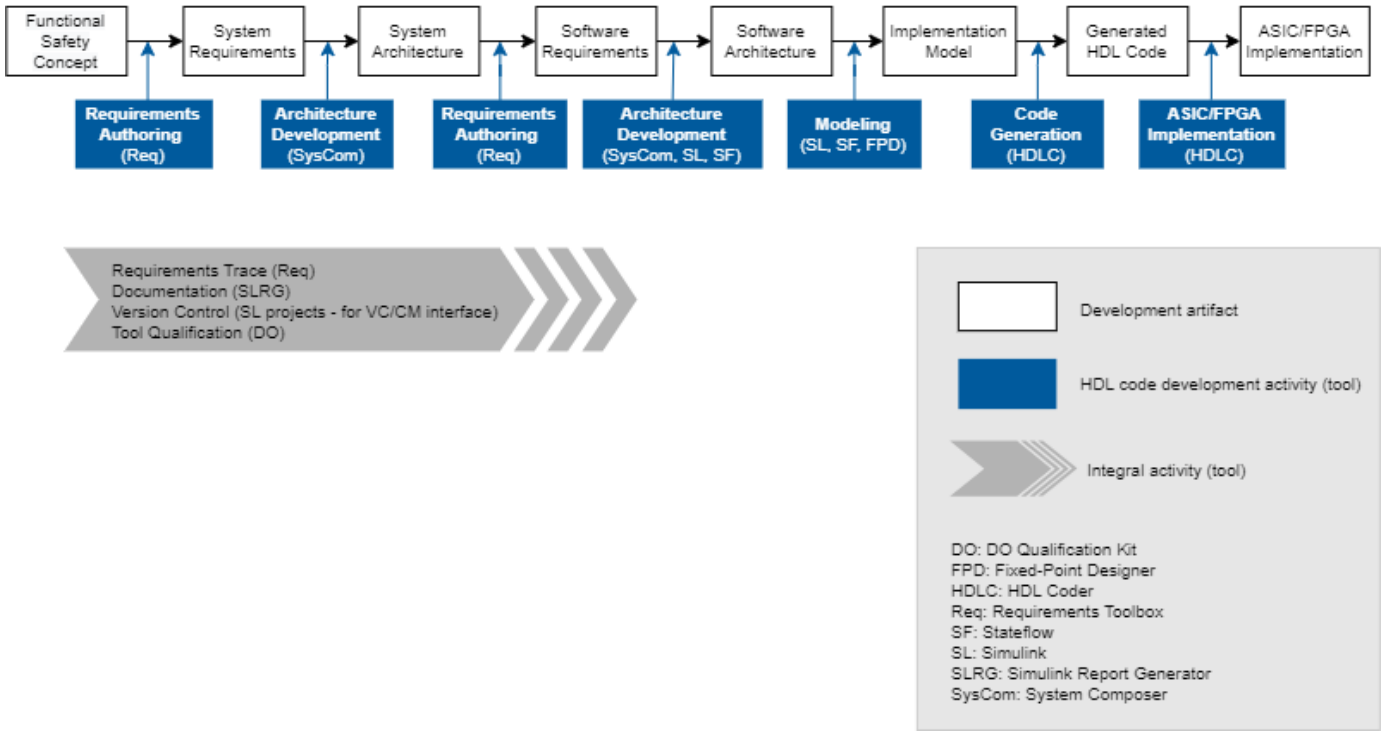


Figure 4: Core Hardware Development Artifacts and Activities

Overview of the Model-Based Design Workflow: Verification Activities

Figure 5 illustrates verification activities that correspond to the core hardware development artifacts and activities shown in Figure 4.

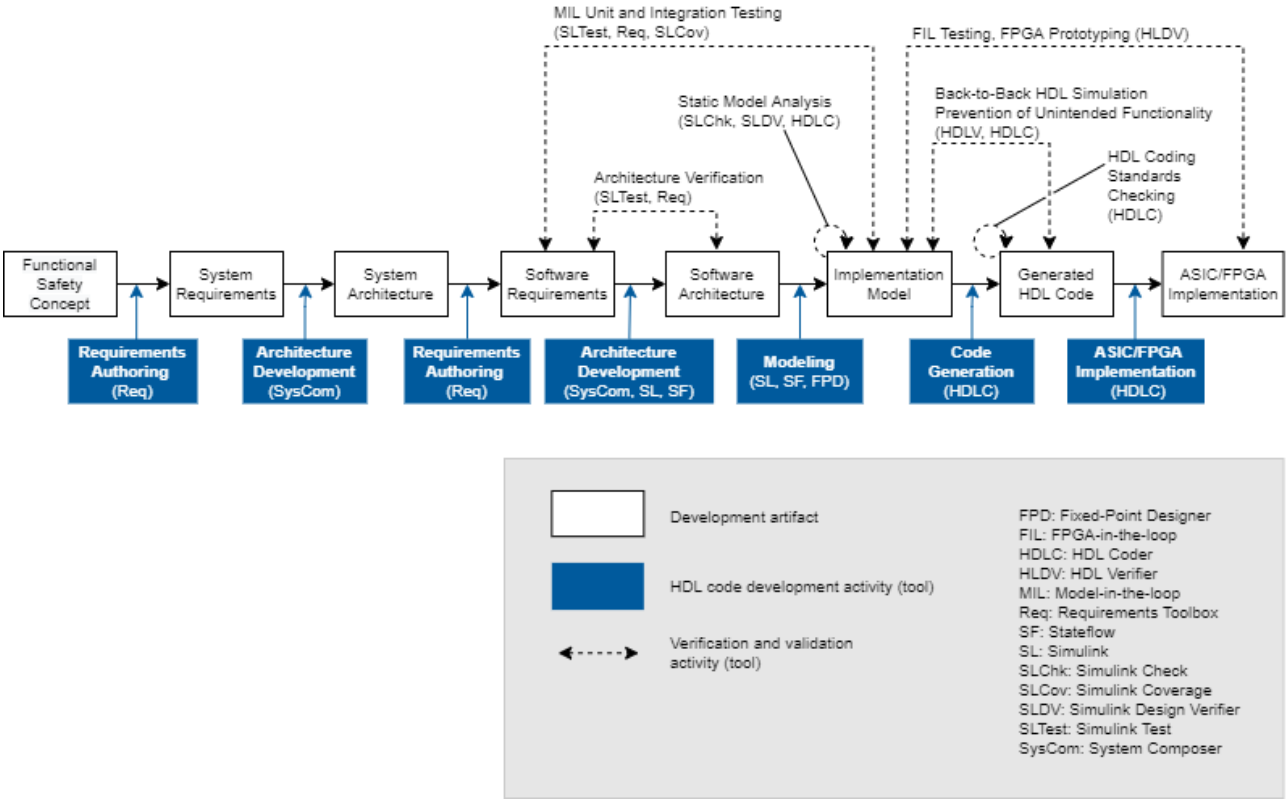


Figure 5: Validation and Verification Artifacts and Activities

Planning Process

This table contains a summary of the planning objectives from DO-254. The table also describes the potential impact to the process when using Model-Based Design.

Table A-1: Planning Process Objectives

	Objective	Ref Sections	Assurance Levels	Process Impact When Using Model-Based Design
1	Hardware design life cycle processes are defined.	4.1(1)	A, B, C, D	Must include Model-Based Design as part of the design process
2	Standards are selected and defined.	4.1(2)	A, B	Must include modeling standards
3	Hardware development and verification environments are selected or defined.	4.1(3)	A, B, C, D	Must include Model-Based Design tools used in the lifecycle processes
4	The means of compliance of the hardware design assurance objectives are proposed to the certification authorities.	4.1(4)	A, B, C, D	Must define credit taken for Model-Based Design in relation to the objectives.

The following sections describe the potential impacts for each objective when using Model-Based Design, if applicable, as compared to traditional development.

Hardware Design Life Cycle Processes Are Defined

You must define Model-Based Design as one of the activities in the hardware development process. Models are typically used to represent the hardware conceptual design. The HDL code represents the hardware detailed design.

You can address change control and configuration management of the models and HDL code during the planning process.

Standards Are Selected and Defined

Because models can represent the conceptual design (see section “Hardware Design Life Cycle Processes Are Defined” on page 2-5), you can use modeling standards to satisfy the requirements standards objectives. Use High-Integrity System Modeling guidelines and the HDL Model Checker rules as a starting point for the project modeling standards. To verify compliance to the modeling standards, use the Model Advisor and/or human reviews.

Hardware Development and Verification Environments Are Selected or Defined

You must define Model-Based Design tools that you use in the development and verification processes. These tools can include:

- MATLAB
- Simulink
- Stateflow
- Fixed-Point Designer
- HDL Coder
- HDL Verifier
- Simulink Check
- Simulink Coverage
- Requirements Toolbox
- Simulink Test
- Simulink Design Verifier
- MATLAB Report Generator
- Simulink Report Generator

The Means of Compliance of the Hardware Design Assurance Objectives Are Proposed to the Certification Authorities

The certification authorities must agree to certification credit taken for the use of Model-Based Design processes or tools. Model-Based Design tools being qualified, either as development or verification tools, must be identified and the tool qualification activities must be defined.

You can use the DO Qualification Kit in the qualification of the MathWorks verification tools.

Hardware Design Process

The following table contains a summary of the hardware design objectives from DO-254. The table also describes the available Model-Based Design tools for satisfying the objectives.

Table A-2: Hardware Design Process Objectives

	Objective	Ref Sections	Assurance Levels	Available Products for Model-Based Design
1	Requirements are identified, defined and documented	5.1.1(1)	A, B, C, D	Requirements Toolbox
2	Derived requirements produced are fed back to the appropriate process.	5.1.1(2)	A, B, C, D	Requirements Toolbox
3	Requirement omissions and errors are provided to the appropriate process for resolution.	5.1.1(3)	A, B, C, D	Requirements Toolbox
4	The hardware item conceptual design is developed and consistent with its requirements.	5.2.1(1)	A, B	Simulink, Stateflow, Fixed-Point Designer, Simulink Report Generator, Requirements Toolbox
5	Derived requirements produced are fed back to the requirements capture or other appropriate process.	5.2.1(2)	A, B	Simulink, Stateflow, Fixed-Point Designer, Simulink Report Generator, Requirements Toolbox
6	Requirement omissions and errors are provided to the appropriate process for resolution.	5.2.1(3)	A, B	Simulink, Stateflow, Simulink Coverage
7	The detailed design is developed from the hardware item requirements and conceptual design data.	5.3.1(1)	A, B, C, D	HDL Coder
8	Derived requirements are fed back to the conceptual design or other appropriate process.	5.3.1(2)	A, B, C, D	HDL Coder
9	Requirement omissions and errors are provided to the appropriate process for resolution.	5.3.1(3)	A, B, C, D	HDL Coder, HDL Verifier
10	A hardware item is produced which implements the hardware detailed design using representative manufacturing processes.	5.4.1(1)	A, B, C, D	Not applicable
11	The hardware item implementation, assembly and installation data is complete	5.4.1(2)	A, B, C, D	Not applicable

	Objective	Ref Sections	Assurance Levels	Available Products for Model-Based Design
12	Derived requirements are fed back to the detailed design or other appropriate process.	5.4.1(3)	A, B, C, D	Not applicable
13	Requirement omissions and errors are provided to the appropriate process for resolution.	5.4.1(4)	A, B, C, D	Not applicable
14	A baseline is established that includes all design and manufacturing data needed to support the consistent replication of the hardware item.	5.5.1(1)	A, B, C, D	Not applicable
15	Manufacturing requirements related to safety are identified and documented and manufacturing controls are established.	5.5.1(2)	A, B, C, D	Not applicable
16	Derived requirements are fed back to the implementation or other appropriate process.	5.5.1(3)	A, B, C, D	Not applicable
17	Errors and omissions are provided to the appropriate process for resolution.	5.5.1(4)	A, B, C, D	Not applicable

The following sections describe the potential impacts for each objective when using Model-Based Design, if applicable, as compared to traditional development.

5.1.1(1) − Requirements Are Identified, Defined and Documented

You can use Requirements Toolbox to develop the hardware requirements. The requirements components in Requirements Toolbox trace to the appropriate system-level requirements, which is developed in accordance with ARP4754. You can use Requirements Toolbox to trace requirements to the model components that implement them.

You can use Requirements Toolbox to generate a Requirements report, which provides the requirements in PDF, HTML, or Microsoft® Word formats.

5.1.1(2) − Derived Requirements Produced Are Fed Back to the Appropriate Process

When using Requirements Toolbox to define hardware requirements, you must use keywords or custom attributes in the requirements set to define any requirements components that do not trace to the system requirements. These are then provided to the systems process.

5.1.1(3) – Requirement Omissions and Errors Are Provided to the Appropriate Process for Resolution

When using Requirements Toolbox to define hardware requirements, the review of requirements set can uncover omissions or errors in the system requirements. These are provided to the system process for resolution.

5.2.1(1) – The Hardware Item Conceptual Design Is Developed and Consistent with Its Requirements

When using models to define the conceptual design, you can develop the conceptual design using Simulink, Fixed-Point Designer, and Stateflow. You can use Requirements Toolbox to trace the components within these models to the appropriate hardware requirements. You must develop models in accordance with the modeling standards defined during the planning process.

You can use Simulink Report Generator to generate a System Design Description report, which provides the design in PDF, HTML, Microsoft Word, or PowerPoint® formats.

5.2.1(2) – Derived Requirements Produced Are Fed Back to the Requirements Capture or Other Appropriate Process

When using models to define the conceptual design, you must identify as derived requirements any Simulink or Stateflow components that do not trace to the hardware requirements. These are provided to the hardware requirements process for evaluation.

5.2.1(3) – Requirement Omissions and Errors Are Provided to the Appropriate Process for Resolution

When using models to define the conceptual design, the simulation and model coverage analysis of Simulink and Stateflow models can uncover omissions or errors in the hardware requirements. These are provided to the hardware requirements process for resolution.

5.3.1(1) – The Detailed Design Is Developed from the Hardware Item Requirements and Conceptual Design Data

When using Simulink and Stateflow models to define the conceptual design, you can use HDL Coder to develop the detailed design. HDL Coder can provide a report that traces the detailed design to the conceptual design.

5.3.1(2) – Derived Requirements Produced Are Fed Back to the Conceptual Design or Other Appropriate Process

You must identify, as derivative requirements, any HDL code that does not trace to the model. These are provided to the conceptual design process.

5.3.1(3) – Requirement Omissions and Errors Are Provided to the Appropriate Process for Resolution

Co-simulation using HDL Verifier can uncover omissions or errors in the conceptual design. These are provided to the conceptual design process.

5.4.1(1) – A Hardware Item Is Produced Which Implements the Hardware Detailed Design Using Representative Manufacturing Processes

The same as for traditional projects.

5.4.1(2) – The Hardware Item Implementation, Assembly and Installation Data Is Complete

The same as for traditional projects.

5.4.1(3) – Derived Requirements Are Fed Back to the Detailed Design or Other Appropriate Process

The same as for traditional projects.

5.4.1(4) – Requirement Omissions and Errors Are Provided to the Appropriate Process for Resolution

The same as for traditional projects.

5.5.1(1) – A Baseline Is Established That Includes All Design and Manufacturing Data Needed to Support the Consistent Replication of the Hardware Item

The same as for traditional projects.

5.5.1(2) – Manufacturing Requirements Related to Safety Are Identified and Documented and Manufacturing Controls Are Established

The same as for traditional projects.

5.5.1(3) – Derived Requirements Are Fed Back to the Implementation or Other Appropriate Process

The same as for traditional projects.

5.5.1(4) – Requirement Omissions and Errors Are Provided to the Appropriate Process for Resolution

The same as for traditional projects.

Validation and Verification Process

The following table contains a summary of the validation and verification objectives from DO-254. The table also describes the available Model-Based Design tools for satisfying the objectives.

Table A-3: Verification and Validation Process Objectives

	Objective	Ref Sections	Assurance Levels	Available Products for Model-Based Design
1	Derived hardware requirements against which the hardware is to be verified are correct and complete.	6.1.1(1)	A, B, C, D	Requirements Toolbox
2	Derived requirements are evaluated for impact on safety.	6.1.1(2)	A, B, C, D	Requirements Toolbox
3	Omissions and errors are fed back to the appropriate processes for resolution.	6.1.1(3)	A, B, C, D	Requirements Toolbox
4	Evidence is provided that the hardware implementation meets the requirements.	6.2.1(1)	A, B, C, D	Simulink Report Generator, Simulink Test, Simulink Coverage, Simulink Design Verifier, HDL Verifier
5	Traceability is established between hardware requirements, the implementation, and the verification procedures and results.	6.2.1(2)	A, B, C	Simulink Report Generator, Requirements Toolbox, Simulink Test, HDL Coder
6	Acceptance test criteria are identified, can be implemented, and are consistent with the hardware design assurance level of the hardware functions.	6.2.1(3)	A, B, C, D	Not applicable
7	Omissions and errors are fed back to the appropriate processes for resolution.	6.2.1(4)	A, B, C, D	Not applicable

The following sections describe the potential impacts for each objective when using Model-Based Design, if applicable, as compared to traditional development.

6.1.1(1) − Derived Hardware Requirements Against Which the Hardware Is to Be Verified Are Correct and Complete

Validation of derived hardware requirements is accomplished using requirements reviews. See “5.1.1(1) − Requirements Are Identified, Defined and Documented” on page 2-9.

6.1.1(2) – Derived Requirements Are Evaluated for Impact on Safety

Derived hardware requirements must feed back to the system safety assessment process for evaluation of the impact on safety.

6.1.1(3) – Omissions and Errors Are Fed Back to the Appropriate Processes for Resolution

The same as for traditional projects. See “5.3.1(3) – Requirement Omissions and Errors Are Provided to the Appropriate Process for Resolution” on page 2-11

6.2.1(1) – Evidence Is Provided That the Hardware Implementation Meets the Requirements

Verification is done at each stage of the design process, conceptual design verification, detailed design verification, and hardware item verification.

When models are used to define the conceptual design, as described in DO-254 Section 2.2.4, compliance with hardware requirements is accomplished by using a combination of model reviews, model analysis, and simulation. For example:

- Use Simulink Report Generator to generate a System Design Description report that includes a trace report to the hardware requirements.
- To verify that hardware requirements are satisfied, you can use Simulink Test and Simulink Coverage to develop test cases from the hardware requirements and execute those test cases on the model.
- To identify unintended functionality in the conceptual design and also assess the completeness of the simulation cases, use the model coverage report in Simulink Coverage.
- Use Simulink Design Verifier to detect design errors.
- Use Simulink Check to verify predefined model standards and also to create customized checks.
- Use Simulink to generate a Model Comparison report, which you can use to explore the differences, view the changes highlighted in the original models, and merge differences.

When you use HDL Coder to generate the detailed design, as described in DO-254 Section 2.2.7, compliance with hardware requirements is accomplished by using a combination of HDL code reviews and co-simulation. To help verify that hardware requirements are satisfied, you can use Simulink Test and HDL Verifier to develop test cases from the hardware requirements and execute those test cases on the co-simulation block. The co-simulation tool can measure coverage of the HDL code during this simulation. HDL Coder comes with a set of industry standard coding rules that you can apply. You can also manually review the HDL code for compliance to the rules selected for the project.

For verification of the hardware, you accomplish compliance with hardware requirements by using FPGA-in-the-loop testing. You can use Simulink Test and HDL Verifier to execute the test cases on the FPGA to verify that the hardware requirements are satisfied. Or you can use a traditional testing environment that does not include Simulink or HDL Verifier to test FPGA or ASIC.

You can use the DO Qualification Kit to qualify these capabilities as a verification tool:

- High-Integrity System Modeling checks in Simulink Check.

- Custom Model Advisor checks.

  Note You are responsible for defining the tool operational requirements, test cases, procedures, and expected results for custom checks.

- Simulink Design Verifier design error detection.
- When used for pass and fail determination, the Test Manager in Simulink Test.
- System Design Description report in Simulink Report Generator.
- Model Coverage in Simulink Coverage.
- Model Comparison report in Simulink.

6.2.1(2) – Traceability Is Established Between Hardware Requirements, the Implementation, and the Verification Procedures and Results

When using models to define conceptual design, traceability to hardware requirements is established by using model reviews that include a requirements report generated by Requirements Toolbox. You can use HDL Coder to generate a report that traces the detailed design to the conceptual design. Use Requirements Toolbox to trace Simulink test harnesses to the hardware requirements.

6.2.1(3) – Acceptance Test Criteria Are Identified, Can Be Implemented and Are Consistent with the Hardware Design Assurance Level of the Hardware Functions

The same as for traditional projects.

6.2.1(4) – Omissions and Errors Are Fed Back to the Appropriate Processes for Resolution

The same as for traditional projects.

Configuration Management Process

The following table contains a summary of the configuration management objectives from DO-254. The table also describes the available Model-Based Design tools for satisfying the objectives.

Table A-8: Configuration Management Process Objectives

	Objective	Ref Sections	Assurance Levels	Available Products for Model-Based Design
1	Configuration items are uniquely identified and documented.	7.1(1)	A, B, C, D	No impact as compared to traditional development
2	Consistent and accurate replication of configuration items is ensured.	7.1(2)	A, B, C, D	No impact as compared to traditional development
3	A controlled method of identifying and tracking modification to the configuration items is provided.	7.1(3)	A, B, C, D	No impact as compared to traditional development

The following sections describe the potential impacts for each objective when using Model-Based Design, if applicable, as compared to traditional development.

7.1(1) − Configuration Items Are Uniquely Identified and Documented

For projects using Model-Based Design, you must configure and identified the following artifacts if they are used on the project:

- Requirements (level above the models)
- Models
- Report Generator files
- Model Trace Reports
- Model advisor reports
- Simulink Test files
- Model test harnesses
- Model test results reports
- Model coverage reports
- Automatically generated HDL code

These artifacts are in addition to or are a substitution of traditional configured items.

7.1(2) − Consistent and Accurate Replication of Configuration Items Is Ensured

The same as for traditional projects. Part of the traceability is covered by Requirements Toolbox.

7.1(3) − A Controlled Method of Identifying and Tracking Modification to the Configuration Items Is Provided

The same as for traditional projects.

Process Assurance

This table contains a summary of the process assurance objectives from DO-254. The table also describes the available Model-Based Design tools for satisfying the objectives.

Table A-9: Process Assurance Objectives

	Objective	Ref Sections	Assurance Levels	Available Products for Model-Based Design
1	Life cycle processes comply with the approved plans.	8.1(1)	A, B, C, D	No impact as compared to traditional development
2	Hardware life cycle data produced complies with approved plans	8.1(2)	A, B	No impact as compared to traditional development
3	The hardware item used for conformance assessment is built to comply with the associated life cycle data.	8.1(3)	A, B, C, D	No impact as compared to traditional development

The following sections describe the potential impacts for each objective when using Model-Based Design, if applicable, as compared to traditional development.

8.1(1) − Life Cycle Processes Comply with the Approved Plans

The same as for traditional projects.

8.1(2) − Hardware Life Cycle Data Produced Complies with Approved Plans

The same as for traditional projects.

8.1(3) − The Hardware Item Used for Conformance Assessment Is Built to Comply with the Associated Life Cycle Data

The same as for traditional projects.

Certification Liaison Process

This table contains a summary of the certification liaison objectives from DO-254. The table also describes the available Model-Based Design tools for satisfying the objectives.

Table A-10: Certification Liaison Process Objectives

	Objective	Ref Sections	Assurance Levels	Available Products for Model-Based Design
1	The PHAC, hardware verification plan and other requested data should be submitted to the certification authority for review at a point in time when the effects of design changes on the program are minimal.	9.1(1)	A, B, C, D	No impact as compared to traditional development
2	Issues identified by the certification authority concerning planning for the hardware aspects of certification are resolved.	9.1(2)	A, B, C, D	No impact as compared to traditional development
3	Agreement on the PHAC should be obtained with the certification authority.	9.1(3)	A, B, C, D	No impact as compared to traditional development
4	Liaison with the certification authority during the design and certification cycle as outlined in the plan should be continued and issues raised by the certification authority resolved in a timely manner.	9.1(4)	A, B, C, D	No impact as compared to traditional development

The following sections describe the potential impacts for each objective when using Model-Based Design, if applicable, as compared to traditional development.

Means of Compliance and Planning

The same as for traditional projects.

Compliance Substantiation

The same as for traditional projects.

Acronyms

Acronyms

ASIC	Application-Specific Integrated Circuit
FIL	FPGA-in-the-loop
FPGA	Field-Programmable Gate Array
HDL	Hardware Description Language
PHAC	Plan for Hardware Aspects of Certification
UVM	Universal Verification Methodology

B

References

Normative References

[1] DO-254, Design Assurance Guidance for Airborne Electronic Hardware. RTCA, 2000.

[2] ARP4754A - Guidelines for Development of Civil Aircraft and Systems. SAE International, 2010.