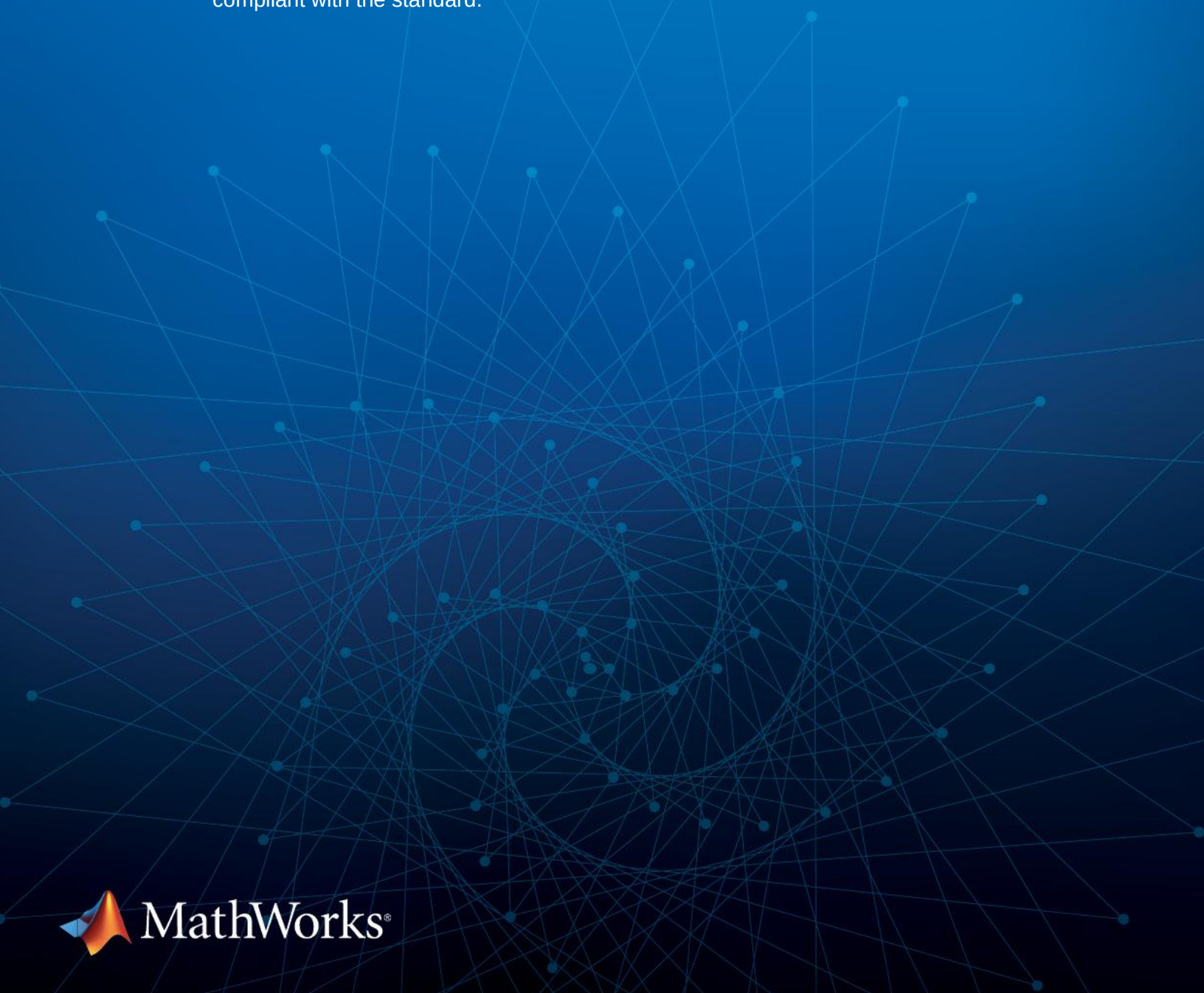
WHITE PAPER

Enabling Model-Based Design for DO-254 Certification Compliance

Featuring MATLAB and Simulink with Siemens EDA Tools

The increasing prevalence and cost of projects that need to comply with the DO-254 standard is forcing companies to evaluate their development processes. This white paper shows how MathWorks and Siemens EDA provide methodologies and a toolchain compliant with the standard.



MathWorks®

Introduction

The increasing prevalence and cost of projects that need to comply with the DO-254 standard is forcing companies to evaluate their development processes.

This white paper shows a development approach to compliance using Model-Based Design. It covers how a DO-254 workflow using Model-Based Design promotes a consistent requirements-oriented project view and increases reuse of design and verification efforts through all phases of the DO-254 life cycle.

Background

The purpose of DO-254 (formally known as RTCA/DO-254 or ED80) is to provide guidance for the development of airborne electronic hardware. The Federal Aviation Administration (FAA), European Aviation Safety Agency (EASA), and other worldwide aviation safety authorities require this standard to ensure that complex electronic hardware used in aircraft systems works as specified under all foreseeable conditions, avoiding faulty operation and potential air disasters.

DO-254 compliance is now common on commercial and military aviation projects. However, companies often struggle with the requirements and costs of DO-254 compliance. Engineers can use Model-Based Design for requirements analysis, algorithm design, automatic HDL code generation, and verification, to produce airborne electronic hardware that adheres to the DO-254 standard. The Model-Based Design approach for DO-254 combines automation tools from both MathWorks and Siemens EDA for design and verification to support a development process that goes from concept through implementation. This approach streamlines the development process and reduces costs.

Simulink, from MathWorks, is the starting point for the Model-Based Design flow. The Simulink environment allows engineers to manage requirements, test sets, architecture and behavior modeling, formal verification, and conformance to modeling standards. Engineers can also perform HDL code generation and verification. This approach delivers two main benefits:

- Finding and fixing errors earlier in the design process is better than finding them later, during implementation and testing.
- Designs, tests, and analyses can be reused throughout the development process and easily communicated among team members.

Siemens EDA offers industry-leading tools that span the design workflow. The tools focus on chip-level solutions for HDL design and verification. They also include capabilities for consistently managing and tracking requirements from design concept through implementation.

Model-Based Design promotes a requirements-oriented project view and greater integration and reusability among conceptual design, detailed design, and implementation.

DO-254 Overview

DO-254 Compliance and Life Cycle

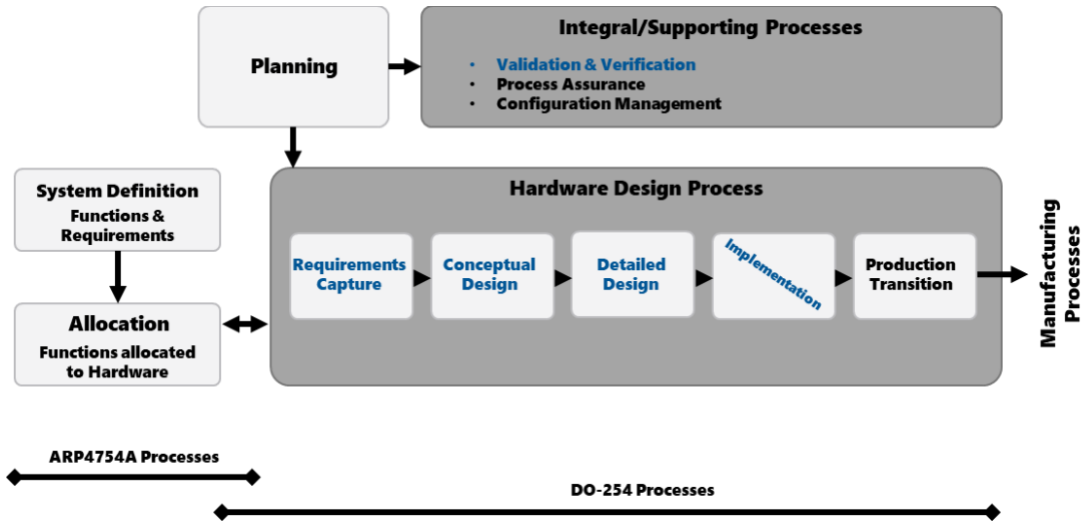


Figure 1 DO-254 compliance lifecycle and associated processes.

Figure 1 shows the DO-254 life cycle and lists the processes that must be performed and documented as a design moves from phase to phase in the life cycle. The following processes, which appear in blue in Figure 1, are discussed in this paper:

- **Requirements Capture - Management and Tracing** – DO-254 demands that design elements and verification artifacts link back to the requirements that they support. Traceability provides proof that a design has implemented the intended function and that it has been thoroughly verified to ensure it performs this function under all foreseeable conditions.
- **Design Processes** – This corresponds to the **Conceptual design, detailed design and implementation in Figure 1**. During the Conceptual Design Phase, the high-level strategy for implementing the functionality expressed in the requirements is conceptualized and documented. During the Detailed Design Phase, the conceptual design is elaborated and refined into the design (such as HDL code) that will be implemented in the hardware device.
- **Integral and Supporting Processes**
  - **Conformance to Design Standards** – In a compliant-development process, pertinent standards must be developed for each phase. As a design moves from phase to phase in the life cycle, it is necessary to show that these standards are being met.
  - **Verification and Validation**– At each phase in the design, the designer must ensure that the current version of the design (conceptual design, HDL code, netlist, hardware) achieves requirements and matches the previous version. Many different techniques and tools ranging from simulation to advanced analysis can be used to perform verification of the design at different phases. Different verification methods at a high level with attention to how design activities and artifacts can be reused throughout the process are shown.

DO-254 Workflow Using Model-Based Design

Figure 2 shows a high-level DO-254 workflow using Model-Based Design.

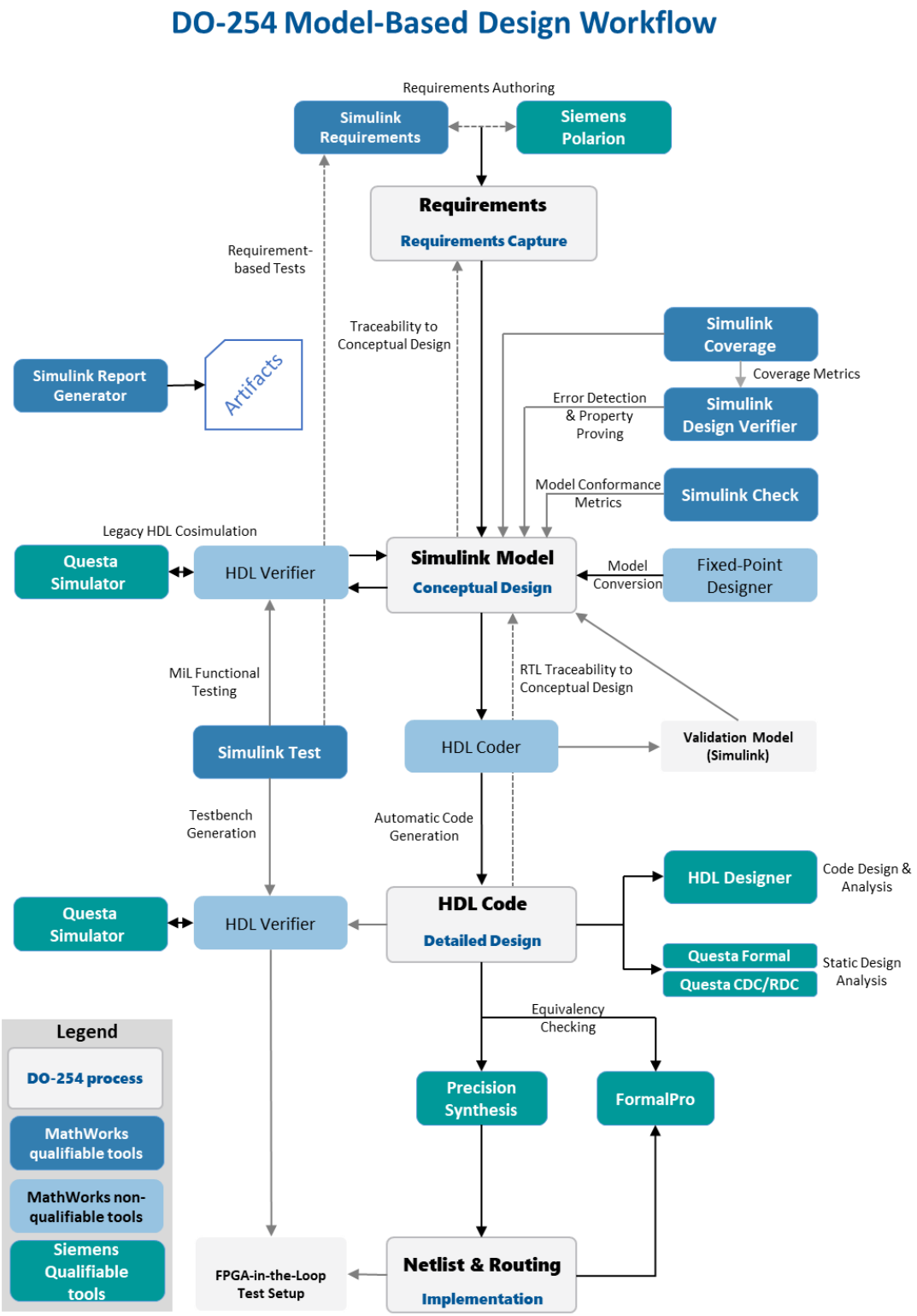


Figure 2. DO-254 workflow with Model-Based Design.

In this workflow, engineers collect and manage requirements with Siemens EDA Polarion® that can then be exported to Simulink Requirements™. From these requirements, an executable Simulink model is created to explore a conceptual design. This conceptual model links directly to requirements at different levels in Polarion and Simulink Requirements.

Using verification and validation tools from MathWorks, engineers can perform functional testing and formal analysis at the conceptual model level and create tests for functional, unit, requirements-based, and regression testing. In Simulink engineers can elaborate the model by adding implementation attributes such as data-streaming and fixed-point effects. The elaborated model allows engineers to verify that the design meets requirements and conformance to model standards and becomes the specification model for HDL implementation. From this fully verified Simulink model, a detailed design in HDL can be generated using HDL Coder™, and SystemVerilog testbench components can be generated using HDL Verifier™.

Further verification of the detailed HDL design can be performed in concert with Siemens EDA verification solutions. Using HDL cosimulation with HDL Verifier, a Simulink testbench may be used with a design-under-test (DUT) simulated in the Questa® HDL simulator to verify that the DUT correctly implements the specification model. The test vectors created at the conceptual model level with Simulink Test™ are applied to the Simulink testbench during cosimulation. HDL code coverage is measured in Questa to determine the effectiveness of the test vectors.

The entire Simulink testbench can be exported to the Siemens verification environment by generating SystemVerilog DPI-C components representing the stimulus, reference model, and checker. For organizations using the Universal Verification Methodology (UVM), HDL Verifier can generate either individual UVM verification components or complete UVM verification environments.

From this stage forward, Siemens EDA provides the primary environment for additional HDL development, code checking, coverage closure, code visualization, and review. Questa Formal perform static design, automated coverage analysis and Siemens FormalPro™ performs logical equivalency for model checking. Questa CDC and RDC perform clock and reset domain crossing checks for metastability and glitch scenarios. FPGA synthesis and integration with FPGA vendor place and route tools is accomplished by Precision RTL®.

Requirements Capture

DO-254 projects are requirements-driven projects. Requirements define the intended function of a device, and a DO-254 compliant process ensures that a device performs its intended function. System requirements allocated to a hardware item must be reviewed, captured, managed, and traced to the pertinent design activities. Likewise, derived requirements, those derived from design decisions throughout the process, must go through these same processes. Therefore, a DO-254 project shall have mechanisms for:

- Capturing requirements, as per the first phase of the DO-254 life cycle
- Managing changes to requirements that occur throughout a program
- Tracing requirements to design and verification activities that occur throughout a program in the different phases

Companies that serve the aerospace market often use enterprise-level requirements management systems such as Siemens Polarion ALM. This provides a database mechanism to

store and manage requirements and can support large complex systems. It is essential that the design and verification work links back to these requirements, regardless of their source environment. In DO-254, this linking is called requirements tracing. Capturing a static set of requirements can be achieved relatively simply. However, establishing a requirements-driven design flow and managing requirements as they evolve throughout a project is a much more daunting challenge. A requirements-driven design flow requires entering requirements, tracking changes to requirements, and linking to design and verification artifacts.

MathWorks and Siemens have applied their expertise in design automation tools to automate requirements management and tracing. The Polarion and Simulink Requirements integration establishes traceability between design elements and verification artifacts at model and code levels. It also validates requirements by facilitating requirements reviews, offering coverage information, guiding verification activities based on requirements status, and providing certification artifacts. Polarion and Simulink Requirements integrate with both MathWorks HDL Coder and Siemens native environments of HDL development, verification, and synthesis, and are flexible enough to adapt to nearly any other tool that would be used in a DO-254 development process. Designers link requirements information to specific blocks, subsystems, and entire models. This information is then automatically passed through to HDL code generated by HDL Coder. This process is clarified within the “Detailed Design” section.

In addition to traceability and validation support, these requirements tools assist project management by creating a visual depiction of project status, which shows the requirements that have and have not been designed and verified. They can also generate the traceability matrices required to meet DO-254 traceability objectives. In essence, combining Polarion and Simulink Requirements, it is possible to have a requirements-oriented project management environment from concept through implementation that supports the traceability needs of DO-254 projects. Figure 3 depicts a requirements-driven flow established with the above-mentioned tools.

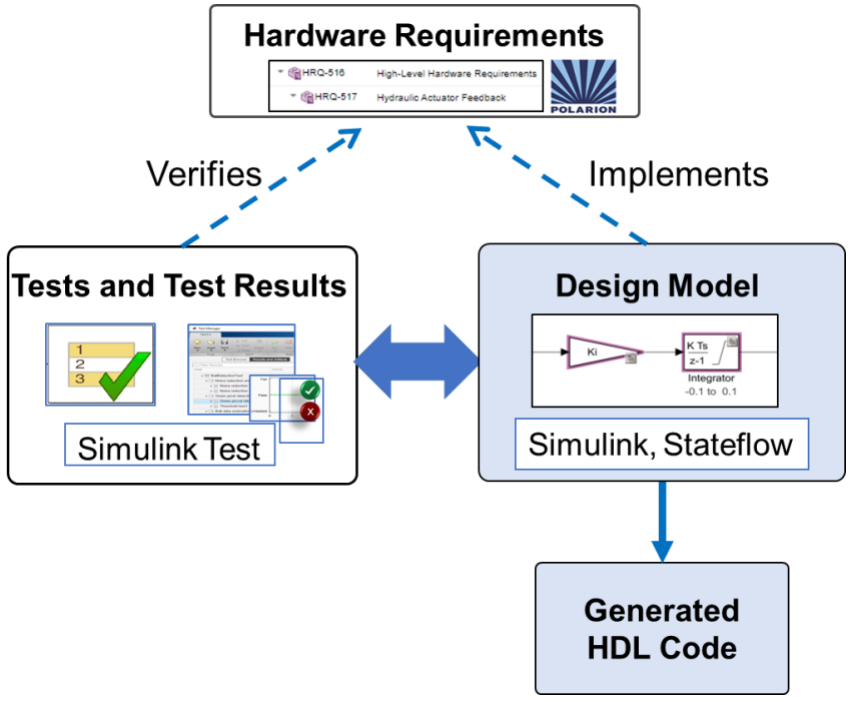


Figure 3. Requirements-driven workflow.

Conceptual Design

Once requirements are firmly established, the next step in the process is for a design engineer to develop a conceptual design that is consistent with and achieves the high-level requirements captured in the previous phase. This section discusses how Simulink and other MathWorks model-level tools can be used to develop and verify the conceptual design.

Conceptual Model Design

Simulink is an industry-standard tool for designing, implementing, and verifying aerospace systems. It serves as the main platform for Model-Based Design. From the requirements captured previously, a design engineer constructs an executable version of the design. It is important to highlight that it is also possible to build an architecture view using System Composer™. This last part will cover the needs to comply with the ARP4754A standard. Simulink enables engineers to build up these algorithmic models in an intuitive graphical manner.

Figure 4 shows an example of an aircraft control that was developed in Simulink using Model-Based Design. Stateflow® is used to develop finite-state machines and logic. Additional blocksets provide higher level functionality for application-specific tasks. Simulating the larger system and environment in which the hardware will operate enables engineers to fully test a system before implementation. For example, consider the design of the actuation control algorithms. These algorithms can be designed independently within Simulink or developed as part of a larger system-level aircraft model, also in Simulink. The system-level aircraft simulation model can include the logic algorithm, a six-degree-of-freedom airframe model with environmental effects, sensor models, and actuator models.

Having a model enables engineers to test their designs earlier in the process and quickly evaluate what-if scenarios. In this example, the system-level model lets an engineer test the control laws under varying conditions, sensor failures, and pilot inputs.

As confidence increases in the design, models are elaborated to specify architecture and include implementation effects. Fixed-Point Designer™, for instance, lets engineers model and analyze a design to help them choose optimal fixed-point word lengths. Simulink enables these effects to be simulated and compared back to a reference design to ensure that requirements are still being met.

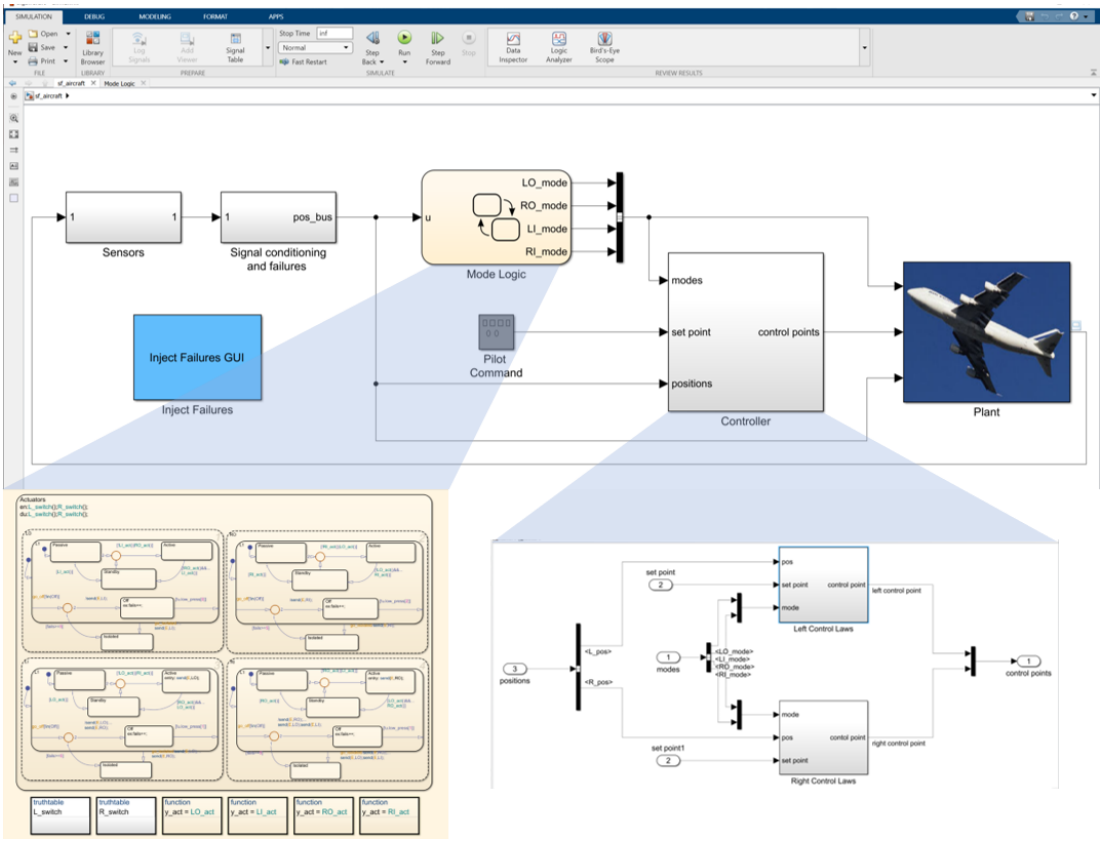


Figure 4. Aircraft control model including Simulink and Stateflow.

Traceability in the Conceptual Model

In a workflow with Model-Based Design, all elements of the conceptual design should be traceable to the requirements they satisfy. As introduced before, MathWorks and Siemens provide traceability support via Simulink Requirements and Polarion. This traceability is preserved in the generated HDL and extends throughout HDL analysis and testing.

Verifying the Conceptual Model

The conceptual design must be analyzed to verify whether it meets requirements. Several MathWorks products can assist in this task. For example, MATLAB can be used to script execution, conduct parameter sweeps, and perform analysis on simulation outputs. These tasks can be run in parallel on multi-core machines or clusters using parallel and distributed computing. MathWorks has also developed tools specifically to aid in system verification. Simulink Test is a platform that can be used to create and execute tests of the Simulink model. Tests can be authored to demonstrate that specific functional requirements are being satisfied. Simulink Test can be used with Simulink Report Generator™ to automatically generate artifacts and with Simulink Coverage™ to generate tests and requirements coverage reports. These artifacts can be used for certification evidence.

The generated tests can be reused later in the design process. Simulation helps validate that requirements are satisfied by enabling a design to be exercised over a range of conditions. While simulation is essential, it can be a challenge to ensure that a set of simulations fully exercises a design over all conditions. To ensure a complete functional test coverage, formal analysis can be used in conjunction with simulation to generate test cases and to perform

property proving. These techniques use mathematically rigorous procedures to simplify and search through all model's possible execution paths to find test cases and counter examples.

This systematic analysis provides deeper understanding of the behavior of designs. For example, consider the takeoff-abort algorithm discussed above. Typically, this type of logic in software or hardware involves several sensor inputs, such as airspeed, acceleration, and pilot input. Using formal property proving, an engineer can use a commercial formal verification tool to verify a certain system behavior such as, "Prove to me that this logic will never engage if the airspeed and acceleration are within certain ranges." Simulink Design Verifier™ lets a developer define these mission-critical properties and prove that certain scenarios cannot happen under any conditions at model level.

Throughout testing, model coverage can be a useful metric to assess how fully tests are exercising a model. Simulink Coverage can track and report on model coverage. These coverage metrics should first be gathered using functional based tests executed against the model. Although functional tests are used to ensure that design requirements are met, they often do not exercise 100% of the design. Simulink Design Verifier uses formal methods to automatically generate test cases to complement functional tests and significantly improve the modified condition/decision coverage (MC/DC) of the design at the model level. Even if not a requirement for certification, this model-level coverage testing is very useful to validate and verify a design. If test cases are not achieving 100% coverage, it may be an indication that additional requirements are needed, design elements are unnecessary, or that a design is inherently difficult to test. These insights are valuable in refining requirements, developing a conceptual design, and creating tests. Significant savings are realized by fixing these errors early in the conceptual design phase. Note that testing must also be exercised on HDL and later stages of design. However, as discussed, the test cases generated on the conceptual model can be reused in HDL-level testing.

Conforming to Conceptual Model Design Standards

As discussed previously, the development of and adherence to design and coding styles is required by DO-254. Conceptual design standards can be developed and applied to the Simulink model. Modeling standards are equivalent to coding standards and can dictate aesthetic and functional aspects of the model. Model Advisor is a standard feature of Simulink that can execute pre-packaged sets of model checks. Simulink Check™ enables the customization and deployment of these checks within an organization. A DO-254 specific set of rules is now available in Simulink Check to support the adherence to the style.

It is important to note that these checks are static, which means that design engineers are not executing the model, but rather looking at it statically and analyzing its characteristics. Typical characteristics include settings, data types, code generator settings, and HDL settings. This static process can detect simple mistakes, such as a missing connection for a block input or output. It can also detect more complex and serious issues, such as block settings that may result in an overflow in a fixed-point operation. The HDL detailed design must also conform to standards. Checking that HDL code conforms to acceptable standards is discussed in the next section.

Detailed Design

The detailed design process is generally agreed upon to begin at the HDL stage of development. This development can involve handwritten code or automatically generated code with HDL Coder. Automatically generating HDL can increase efficiency by reducing the amount of hand coding required and enabling faster design iterations. The workflow discussed below

includes automatic HDL generation. Note that the verification activities at code level discussed throughout the DO-254 workflow may be used whether hand coding or automatic code generation is used.

Generated HDL from the Conceptual Model

In a workflow using Model-Based Design, the generated HDL code is read into HDL Designer from Siemens for independent assessment and integration with either existing HDL or portions of the design. Within HDL Designer, the HDL code is examined via code reviews, automatically checked against HDL coding standards, and visualized for ease of understanding. HDL Coder also generates scripts for HDL Designer to perform linting on generated HDL code.

Traceability in Detailed Design

Just as conformance of the conceptual design to design standards was demonstrated at the Simulink model level, conformance of the detailed design to HDL coding standards must also be demonstrated. HDL Coder offers a variety of ways to customize the generated code to comply with process requirements, such as naming conventions, optimizations, and reset style. HDL Designer has an integrated design rules checking engine that is sometimes referred to as a linting tool. This feature provides several different rule sets that can be used as-is or customized. One of these is called the “DO-254 rule set,” which contains several coding rules that can be used to meet the DO-254 objective of defining a set of HDL coding standards (as specified in Order 8110-105). HDL Designer automatically checks HDL code to ensure that it conforms to this rule set.

Reviewing the Code

HDL code must be examined by independent review to ensure that it conforms to HDL coding standards and correctly implements the required functionality. The HDL Designer rules checking engine ensures conformance to HDL standards. Both HDL Coder and HDL Designer assist with reviewing HDL code to ensure that it implements the required functionality. The HTML code generation report generated by HDL Coder facilitates navigating from HDL code back to the blocks in the Simulink conceptual model and to requirements. This navigation is bidirectional. The graphical conceptual model in conjunction with the generated HDL code helps reviewers more quickly understand and analyze a design. A traceability report is also generated to aid in this review process.

HDL Designer helps facilitate code reviews by providing features to visualize the HDL code. These visualizations, along with the rules checking results, examination of the requirements links, and functional verification (described in next section) provide independent output assessment of the generated code.

Verifying the HDL Model

Verification must also be performed at the detailed design level. This task is required for both handwritten code and automatically generated code. As with the conceptual model, multiple verification techniques can be employed. These techniques range from basic simulation to advanced formal methods. There is a high degree of flexibility in how verification activities are performed.

Verification of the detailed design compared to the conceptual design is performed using HDL Verifier, either by performing HDL cosimulation of Simulink with Questa or by generating SystemVerilog DPI-C verification components for use within the Questa simulator environment. An organization should consider the utility of each of these methods and discuss their

applicability with the certification liaison. The sections below focus on how to reuse testing data and artifacts developed earlier in the conceptual design process.

Simulating the HDL Model

The section entitled “Conceptual Design” discussed how simulation is an important element of verifying the conceptual design. Simulation is also an essential tool for verifying the detailed HDL design. Questa Simulator is a leading simulator in the military and aerospace industry. Questa simulates HDL designs with an emphasis on debugging. It also provides built-in code coverage analysis in support of the DO-254 elemental analysis method for level A/B designs. Questa combines the Questa simulation engine with advanced verification capabilities from languages such as SystemVerilog, PSL, and SystemC. These capabilities include:

- Transaction-level modeling
- Constrained random testing
- Object-oriented programming (OOP) techniques for testbench creation
- Automated test stimulus
- Dynamic assertion-based verification, including an assertion debugger
- A unified coverage database (UCDB), which has been donated and accepted as an Accellera standard
- A verification management environment for ease of managing and reporting on project verification activities

It also has a deep integration with Polarion, delivering round-trip traceability from verification artifacts stored in UCDB to requirements. These advanced verification methods aid in verifying devices of substantial complexity that contain concurrent behaviors. Thus, Questa is typically used on complex devices, including ASICs and large FPGAs.

Verifying the HDL Model – Conceptual Design Test Case Reuse

During the conceptual design phase, the simulation engine is Simulink. During the detailed design phase, the simulation engine is Questa. While the type of simulation employed has changed, there are several ways to leverage the verification activities performed in the conceptual design phase in the detailed design phase. The two simulators are connected through HDL cosimulation and HDL testbench generation.

HDL Verifier from MathWorks enables tests authored in the MATLAB, Simulink, and Simulink Test environments to be executed against HDL code simulated in Questa. HDL cosimulation lets engineers easily reuse the Simulink test cases and analysis routines developed during conceptual design to ensure that the specification model and HDL are functionally equivalent.

The aircraft control model discussed earlier was designed and simulated as part of a larger system-level aircraft model. From this Simulink conceptual model, an HDL detailed design of the algorithm was generated using HDL Coder. HDL Verifier allows the designer to run the system-level model and tests from the conceptual design against the generated HDL running in Questa and to perform HDL code coverage analysis using Questa.

HDL Verifier also includes the ability to generate HDL testbench files as SystemVerilog DPI-C components based on tests performed on the conceptual design. In the HDL generation options, the designer can specify that testbench files be generated along with the algorithmic HDL also supporting the Universal Verification Methodology (UVM). In this manner tests

performed on the Simulink conceptual design can be reused when hardware engineers do not have access to Simulink.

Both cosimulation and testbench generation promote test case reuse and enable engineers to quickly test the detailed design (HDL), reducing design iteration time and associated costs. They also allow engineers to leverage the analysis capabilities of both Simulink and Questa: high-level functional testing can be quickly performed and analyzed in the Simulink design environment, while detailed analysis can be performed in Questa.

Verifying the HDL Model – Advanced Analysis

Use of Model-Based Design in DO-254 workflows promotes the concept of reuse in design and verification. Reuse achieved through HDL cosimulation and testbench generation is well suited for functional testing. However, there are additional analyses available to the designer in Siemens design environments. These advanced analysis techniques are discussed below.

Clock-Domain Crossing Analysis

Integrating multiple functions into a single Integrated Circuit (IC) is commonplace today. ICs typically consist of multiple asynchronous clock and reset domains. Designers must implement dedicated hardware to correctly move data from one domain to another. Improper implementation may result in metastability and glitches, a leading cause of device failure. The problems associated with signals that cross clock domains are extremely difficult and expensive to debug and fix because they typically are not detected until a failure occurs in the lab or field. Questa CDC and Questa RDC deliver advanced structural and formal analysis capabilities at multiple points through the design lifecycle to ensure correct clock and reset domain crossing hardware is implemented. A design with two or more asynchronous clock or reset domains should use Questa CDC/RDC during the design process to help reduce the likelihood of metastability.

Formal Verification (HDL Model Checking)

Model checking is a formal technique that analyzes a design against its requirements, which are written as assertions. Model checking was discussed earlier in the section entitled “Verifying the Conceptual Model.” The same concept of exhaustively proving safety-critical properties is true at this level of design as well. In this case, the model is an HDL version of the detailed design. Model checking exhaustively proves that a design performs its intended function, and it is mentioned in DO-254 Appendix B as an acceptable method of advanced verification for level A/B devices. Questa Formal Autocheck identifies scenarios such as combinatorial loops, FSM deadlocks, and arithmetic overflows. Questa Formal Covercheck compliments Autocheck and reveals root causes of code coverage gaps.

Synthesizing the HDL

Synthesis is a transformation of HDL code into a technology-based netlist. Design synthesis is at the heart of all modern PLD, FPGA, and ASIC design flows. Designers, and in turn their synthesis tools, have historically tended to focus on achieving four main design goals: timing performance, design area, power consumption, and tool run time. However, in military and aerospace applications where design assurance is critical, a synthesis tool must consider additional aspects.

Precision Synthesis, an FPGA vendor-independent synthesis product from Siemens, balances aspects of safe synthesis with performance, optimization, and timing goals. It ensures that circuitry intended for proper operation, such as specialized reset circuitry and special state machine encoding, are preserved during synthesis. It also supports the DO-254 principle of

repeatability, providing a means to generate a deterministic and repeatable netlist given a consistent environment and conditions. In addition, it provides integration with the Siemens FormalPro logical equivalency checking tool to provide an added measure of assurance for the generated netlist. More information on FormalPro appears in the next section. Placement and routing of the netlist into a physical device depends on specific knowledge about the target FPGA device. This process requires tools provided by the FPGA vendor. Precision Synthesis has integration with the FPGA vendor software and can directly launch these tools from the Precision environment.

Verifying the Netlist

As described in the introduction, verification is needed at each phase in the DO-254 life cycle to ensure that the design meets requirements and matches the previous version. This design assurance is paramount, especially for DO-254 Level A/B designs. Appendix B of DO-254 states: “As the design assurance level increases, the approach needed to verify that a given design meets its safety requirements may need overlapping, layered combinations of design assurance methods.” There are several ways to perform this verification on the post-synthesis gate-level design and ensure that it is equivalent to the HDL detailed design.

Static Timing Analysis

Precision Synthesis performs internal static timing analysis as a part of the synthesis process. At this point, the analysis is done with estimations only since the actual physical location of the circuitry is not yet known. During the place and route process, FPGA vendor tools such as Xilinx Vivado, Intel Quartus, or Microchip Libero run final timing analysis when physical placement in the target device is known. ASIC implementation tools also have static timing analysis built in, but signoff analysis is typically performed using standalone tools.

Gate-Level Simulation with Timing

Questa supports verification of the gate-level netlist. Verification can be done at the output of synthesis with timing estimates or by including the final timing information back annotated from the place and route procedure. In either case, the HDL testbench environment used during HDL simulation is leveraged to perform gate-level simulations. Comparing and contrasting HDL and GLS design models provide a means of independent assessment. EDA Simulator Link also supports cosimulation at this level of design.

Logical Equivalency Checking

In a DO-254 compliant workflow, repeating functional verification at the gate level is generally accepted as the means to validate synthesis results and to verify the results of HDL simulation. However, for large and complex designs, this repetition can be incredibly time-consuming. A faster approach for verifying synthesis results is a type of formal verification known as logical equivalency checking, or LEC. Siemens Formal Pro compares one model to another to determine whether they are functionally equivalent. This comparison is typically done on the input and output of a process. For example, FormalPro compares the HDL fed into synthesis with the netlist generated to determine if they are functionally equivalent. This same process is often used to compare the input to place and route (i.e., the synthesized netlist) with the output of place and route. This formal methods approach enables faster verification than gate-level simulation.

FPGA-in-the-Loop Testing

FPGA-in-the-loop testing is a technique for verifying that a netlist programmed into an actual FPGA device conforms to the conceptual design model in Simulink. HDL Verifier automates the

process of connecting an FPGA board to a Simulink session on a host computer and transmitting signals between Simulink and the board using PCI-Express, Ethernet, or JTAG for communication. When used with HDL Coder, the entire process of synthesis, place and route, programming, and setting up host/board communication is automated. FPGA-in-the-loop is supported for boards with FPGAs from Xilinx, Intel, and Microchip Technology.

Implementation and Production Transition

The DO-254 workflow using Model-Based Design has centered on the requirements capture, conceptual design, and detailed design phases of development. DO-254 compliance entails a broader scope of activities including implementation, such as programming the FPGA device and production transition (handing off of the data and artifacts required to produce a repeatable, identical final hardware item). The design and verification artifacts outlined above can be reused in these phases.

Conclusion

Aerospace companies and their suppliers need to adopt hardware design workflows to comply with DO-254, supporting documents and related standards. MathWorks and Siemens EDA offer tools that may be used in the Hardware Design Process as defined in DO-254, spanning Requirements Capture, Conceptual Design, and Detailed Design. These tools may also be used for the supporting DO-254 process of Validation & Verification.

MathWorks and Siemens EDA collaborate with FPGA vendors including Xilinx, Intel, and Microchip Technology, to enable handoff to tools offered by these semiconductor vendors for the downstream phases of Implementation and Production Transition.

Learn More

MathWorks

- Support for DO-254 Applications
- FPGA, ASIC, and SoC Development
- Hardware Design with MATLAB and Simulink
- High-Level Synthesis
- UVM Verification
- Verilog Test Bench and VHDL Test Bench

Siemens EDA

- Siemens EDA Functional Safety
- Functional Safety Verification Academy