SIEMENS
Ingenuity for life

Siemens Digital Industries Software

# What is "Verification" in the Context of DO-254 (Avionics) Programs?


## Executive summary

If you are a hardware design or verification engineer, you probably have a good idea of what verification entails. However, add compliance to RTCA/DO-254 as a requirement, and suddenly the definition of "verification" may not be so clear. This paper clarifies the scope, expectations, and nuances of DO-254 verification.



Michelle Lange, Jeff Reeve, Tammy Reeve – Patmos Engineering Services
With Jacob Wiltgen – Siemens Digital Industries Software

If you are a hardware design or verification engineer, you probably have a good idea of what verification entails. However, add compliance to RTCA/DO-254 as a requirement, and suddenly the definition of "verification" may not be so clear. First, the term "verification" must be understood alongside the synergistic term "validation." Next, in a DO-254 context, verification spans a wider scope than it does traditionally, so understanding this is crucial. Also, while "advanced verification" is required for the more safety-critical designs, it might not mean what you think it means. Add to this DO-254 terms and concepts like requirements-based testing, elemental analysis, robustness testing, target testing, and independence, and suddenly the realm of verification might feel quite foreign. If DO-254 verification is on your horizon, keep reading to understand the scope, expectations, and nuances of DO-254 verification.

## Verification and Validation

One of the first clarifications you need when understanding "verification" in the scope of DO-254, is how verification and validation are both intricately synergistic and yet subtly different. RTCA/DO-254 defines validation as "The process of determining that the requirements are the correct requirements and that they are complete" and defines verification as "The evaluation of an implementation of requirements to determine that they have been met." In simple terms, validation ensures the item is correctly defined while verification ensures the item operates as per its (validated) definition. Together, validation and verification (referred to as V&V) ensure the hardware item is what it is supposed to be and does what it should do.

## Verification Objectives & Activities

The next step in understanding verification (and how it is distinct from, yet related to, validation) is to understand the DO-254 objectives related to V&V. DO-254 Section 6.0 covers the "Validation and Verification Process", which is a supporting process (meaning it occurs throughout and alongside the development life cycle, as opposed to being its own unique development phase). The primary objectives of validation are to ensure the hardware requirements are correct and complete, and to evaluate their impact on safety. (To muddy the waters, these objectives are often met via reviews, analysis or test, which are considered to be verification activities). The primary objectives of verification are to provide evidence (usually achieved through reviews, analysis and/or test) that the hardware implementation meets the (validated) requirements, and that the requirements are traceable to their implementation as well as the tests and results that demonstrate compliance.

The activities that formally roll up under verification to meet the primary objectives include identifying the requirements that must be verified, identifying one or more methods of verification (usually reviews, analysis or test) for each, holding the reviews and/or analysis, creating and performing the tests (and/or simulations), establishing traceability between all these elements, and ensuring that every requirement has been sufficiently covered. Of course, all of this needs to be documented and reviewed (which is another facet of verification).

## Requirements-Based Verification

What was just described is requirements-based verification. In other words, all these activities stem from the validated requirements. This ensures that the design does what the requirements say it should. But there is more to DO-254 verification than this. Due to safety concerns, especially at the higher design assurance levels (DALs), the design may need more verification to ensure it does not include any functions in it besides those defined by the requirements or behave in unexpected ways. With requirements-based testing you are only testing to ensure the design does what you expect it to do. When safety is concerned, you must go beyond this and ensure it does not do anything you don't want it to do. Two additional types of verification compliment requirements-based verification to achieve this -- advanced verification and robustness testing.

## Advanced Verification

DO-254 Appendix B states that for designs of the highest DALs (i.e., DAL A or B), you must do some additional verification referred to as advanced verification. Verification engineers in other industries may think this means techniques like assertion-based verification, formal methods, emulation, or any of a myriad of state-of-the-art techniques for verifying today's complex hardware designs. While DO-254 does not rule out any of these techniques, the de-facto standard for achieving advanced verification in a DO-254 context is a technique called elemental analysis. This is basically code coverage (yes, a decades-old technique).

To understand elemental analysis, it helps to think of the HDL code as a series of "elements" that describe aspects of the design. For example, an HDL statement (or a condition, or branch or decision made in the code) describes some element of the design's behavior. These

are the elements in elemental analysis. Elemental analysis ensures that when you perform requirements-based verification, you in fact are exercising all of the design elements. If not, this tells you something about your design, its requirements, or verification thoroughness. Your design may have elements (i.e., functions or behaviors) that you are unaware of (which may indicate a missing requirement), that should not be there (such as extra code, dead or even malicious code), that were not properly traced to a requirement, and/or were simply not tested thoroughly enough. So elemental analysis, as a form of advanced verification, provides an understanding that the design does only what it should, and nothing more, and that each of the requirements that defines the design's behavior is thoroughly tested.

## Robustness Testing

But is this enough? How will the design react given unexpected conditions? This is where robustness testing comes into play. Well established in the realm of airborne software (i.e., DO-178C), robustness testing is barely mentioned in the original DO-254 document. However, more recent guidance has clarified the expectation of robustness testing for hardware. AMC 20-152A, a very new document from the European Aviation Safety Agency (EASA) that applies some newer guidance and clarifies some aspects of DO-254, describes robustness as "the expected behavior of the design under abnormal and boundary/worst case operating conditions of the inputs and internal design states." It goes on to explain the abnormal and boundary conditions and the associated expected behavior of the design should be defined as requirements and verified to ensure that the behavior of the design, even under these unexpected conditions, is known. This verification method basically adds additional requirements to requirements-based verification.

## The Role of Simulation

Every engineer knows that today's designs are very complex and "testing" them usually involves simulation. It is important to understand the definitions of testing and simulation in a DO-254 context. Simulation is considered to be an analysis method. Analysis methods may include techniques such as functional simulation, timing simulation, clock domain crossing analysis, logical equivalency checking, assertion-based verification, static timing analysis, and signal integrity analysis among others. Analysis techniques, especially these, are typically performed on a development model of the design, such as the HDL code or the gate-level netlist. Simulation is a special type of analysis used to verify the design's response to both normal and abnormal conditions. Simulation is run on a design model, with a model of the design's interfaces, and is ideal to verify the design under conditions that may be difficult or impossible to create on the actual physical hardware implementation. If you are using simulation for verification credit of requirements-based test, you typically need to justify the use of this technique by demonstrating that an overlapping set of tests run on the physical board (i.e., "target testing") yields the same results. In doing so, you validate the modeling of the design's external interfaces used in simulation.

## Target Testing

In contrast to analysis, the word "test" in a DO-254 context means verifying the physical. Simulation has a key role in DO-254 verification, but performing simulation alone is never enough in safety-critical and airborne applications. Physical testing, often referred to as target testing in a DO-254 context, is also required.

Target testing is a realm of much confusion and sometimes misinformation, with some vendors claiming they have target testing systems that provide full DO-254 verification credit with the push of a button. While it sounds very desirable, steer clear of the hype and instead implement a DO-254 verification strategy that meets compliance expectations. It is important to fully understand what is meant by target testing. The first concept to understand is on-target testing vs. off-target testing.

*On-target testing* means that the device (e.g., FPGA, PLD, ASIC) is being tested in a production-equivalent circuit card and (usually) with production-equivalent software. In other words, on-target testing verifies the design is working correctly in the final hardware implementation exactly as it will be configured and installed on the aircraft. This is the most desirable type of test. On-target testing can also be used to independently verify the tool outputs, demonstrating that these tools did not introduce or fail to detect design errors. This is important in the context of tool qualification (a topic for another article, as it is too complex to do justice here).

*Off-target testing* means that the physical hardware is tested in an environment that varies in some ways from the actual hardware environment installed on the aircraft. You may get verification credit for this type of testing but will likely have to analyze and justify the installation vs. testing environment differences to ensure that the assumptions built into the test environment are correct. Therefore, when looking at industry

solutions for DO-254 verification and testing, you should understand that no pre-packaged solution provides 100% of DO-254 verification, and that every deviation from the installed hardware and environment must be analyzed and justified for verification/testing credit to be accepted.

## Independence

One final aspect of verification worth mentioning is the concept of independence. This is quite intuitive really. Independence simply means that the person who created the design should not be the one verifying it. Any engineer who has ever tried to verify his/her own design knows this is a good idea. Designs with higher DALs require verification (and validation) to be performed with independence.

## DO-254 Verification Example

While there is no-one-size-fits-all DO-254 verification solution, what follows is an example (taken from a summary of an actual project) of DO-254 verification methods, including a short description of the HDL simulation aspects.

### FPGA Testbench and Test Cases

DO-254 functional verification is based on the requirements allocated to the FPGA from the systems and board-level definitions. An independent verification team develops the test bench and test cases, and performs simulation using a requirements-based verification methodology. This is performed on the completed HDL for the FPGA design. Test cases are developed to cover each requirement, with traceability between requirements and test (which will be used to analyze coverage).

The test bench development includes an architectural plan for constructing the simulation test bench and description of advanced testbench requirements and design. The Hardware Test Cases and Procedures document contains a top-level diagram and description of the simulation testbench design and method for recording simulation results. Standards for the simulation test bench design include a coding standard to ensure readability and consistency of the simulation test code.

Development of the simulation test bench involves the following items:

- FPGA Requirements
- FPGA Verification and Validation plan
- Verification standards and test bench coding standards
- Interface control diagram (ICD) and/or schematics showing devices and interfaces to FPGA
- Datasheets for devices and interfaces to FPGA

### Typical Verification Methods and Tools

| TOOL | OS HOST | SUPPORTED PROCESSES |
|---|---|---|
| *Siemens EDA QuestaSim* | *Red Hat Enterprise Linux* | *Verification Tool:*<br>*Functional verification*<br>*Structural verification*<br>*Code coverage for elemental analysis* |
| *Actel Libero* | *Red Hat Enterprise Linux* | *Design and Verification Tool:*<br>*FPGA implementation*<br>*Timing verification* |
| *Siemens EDA Questa CDC* | *Red Hat Enterprise Linux* | *Verification Tool:*<br>*Clock-domain crossing verification* |
| *Siemens EDA DesignChecker (feature of HDL Designer)* | *Red Hat Enterprise Linux* | *Verification Tool: HDL code linting* |

Table 1 - Example FPGA DAL A Project Tools and Use

In the test bench architecture shown in Figure 1, the Test Harness instantiates the DUT, Bus Functional Models (BFMs), and Monitors. The same test bench is used for both RTL functional simulations and gate-level simulations. Test case files are read by the Test Harness. The Test Harness uses these files to create stimulus and drive the test through a sequence of transactions, perform scoreboarding, and create log files to document the results. The BFMs form the driving and response to/from specific DUT interfaces. The monitors typically continuously check the expected versus actual results.

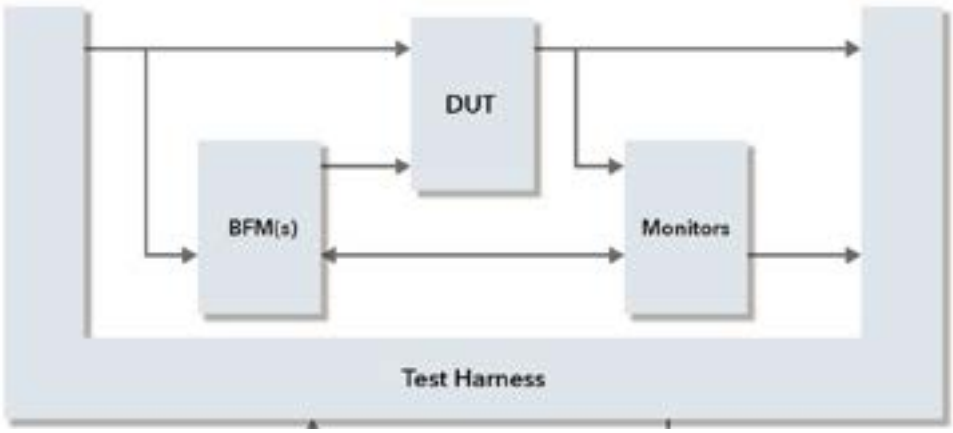


Figure 1 - Example Test Bench Architecture

**Documentation**
The documentation (found in the Hardware Test Cases and Procedures document) will include a top-level diagram (such as the above) and a description that includes the interfaces and connection of blocks of the test bench, including any ports or function calls. Each test bench block requires a brief description of the block purpose. For System Verilog testbenches, documentation includes a UML diagram and description of the functional attributes. All BFMs need to be documented with details sufficient to validate the modules. The documentation of the simulation environment includes the tools and versions along with a description to sufficiently convey how to execute the tests (using the tools) and generate the results.

Verification results are recorded in the Hardware Verification Results document, which will be reviewed to ensure the procedures and results are as expected. This document summarizes the results of reviews, analysis (including simulation and code coverage), and test. Note that automated test benches automatically generate pass/fail results, a manual review of the actual and expected results is required as an independent assessment of the automation.

**Code Coverage**
A DAL A FPGA requires additional Advanced Verification techniques as per DO-254 Appendix B. Once a complete set of simulation test cases is developed, reviewed, and passes, this becomes the basis for running elemental analysis (see DO-254 3.3.1). A code coverage report is created by aggregating the UCDB coverage database for all test cases. This will be reviewed for coverage for the statements, decisions & finite state machines (see AMC 20-152A CD-9 and section B.2.1.5) and gaps in coverage analyzed to determine if there is dead code, missing requirements, and/or missing or incomplete tests.

**Physical Test**
Physical test cases are defined to ensure full verification of the hardware implementation. Sufficient requirements-based test cases at the device level in the target environment are developed and documented in a separate section of the Hardware Test Cases and Procedures document (or a separate document) to ensure validation of the simulation aspects, including BFMs. A combination of simulation and physical tests is optimal to show coverage of FPGA-level requirements. Simulation is essential because it is not always possible to have visibility at the board level of internal functions. Physical test is essential to demonstrate that simulation accurately models the physical implementation and its environment.

**Requirements Tracing and Coverage Analysis**
A Requirements Traceability Matrix (RTM) includes a trace of requirements to the design implementation, test cases and results. Ideally a board-level (physical test) or system-level test also t races to each FPGA requirement to show overlap of the simulation testing.

**Other Aspects of the Verification**
Prior to simulation, linting is commonly applied to the HDL. Typically, the linter is set up with a customized set of compliance checks that verify coding guidelines,

un-synthesizable constructs, undriven signals, out of range indexing, etc.

Prior to design synthesis, the RTL should be evaluated for clock domain crossing (CDC) issues. Proper synchronization techniques should be employed on all clock domain crossing instances and verified.

Once RTL simulation is completed, the design is synthesized, placed and routed (PAR). This process is guided by a timing constraint file that communicates the timing intentions of the design to the point tools used for PAR. The timing constraints include clock definitions, Input/Output delay, timing exceptions, multi-cycle paths, etc. Static Timing analysis verifies the PAR process results in a design that meets the timing criteria captured in the timing constraint file. The resulting netlist and back-annotation file are used for gate-level simulations. The same RTL testbench is used for gate-level simulation. Gate-level simulations are useful for verifying the design operates at desired frequencies with actual delays (max/min) in place, and help reveal glitch detection on edge-sensitive signals, power-up/reset initialization, etc.

More modern and automated techniques of verification, such as assertion-based verification, emulation, using the Questa UCBD, and so on may also be employed in a DO-254 program. Addressing the details of how these techniques may be used is a large topic, best left for another paper. If such techniques are chosen, it's advisable to consult a DO-254 specialist to ensure the tools and methods planned ensure compliance.

## About Patmos Engineering Services

Patmos Engineering Services is an independent engineering consulting company founded and incorporated by Jeff Reeve and Tammy Reeve in January 2000. Patmos has been certified by the Washington State Office of Minority and Women's Business Enterprises (OMWBE) as a Women's Business Enterprise in Engineering Consulting Services, NAICS Code 541330. Patmos offers a unique skillset for digital design (FPGA, ASIC, board level) as well as FAA DER review and approval authority for programmable devices and software. Specifically, Patmos offers:

- DO-254 and DO-178C compliance auditing for FAA, EASA and FAA-EASA coordination
- DO-254 (TR-101), DO-178C (TR-102) and ARP 4754A (TR-103) training
- Process evaluation ("gap analysis") and advising
- Support for commercial and military compliance
- Support for unmanned aerial systems (UAS) i.e., drones
- Support for certifiable IP
- Support for TSO and STC application

In the aerospace domain, the Patmos team supports both commercial and military avionics design and certification programs and is DDTC registered with the United States Department of State Bureau of Military affairs. Outside of the aerospace domain, Patmos has developed a diversity of designs for fields including medical, commercial, and consumer products. The Patmos team has a combined experience in digital hardware design and certification of over 40 years. The goal of Patmos is to provide integrity and honesty in engineering practices and activities. Patmos is a partner with Siemens EDA in support of DO-254 and DO-178C programs.

Patmos Engineering Services:
https://www.patmos-eng.com/about-us/

**Siemens Digital Industries Software**

**Headquarters**
Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
+1 972 987 3000

**Americas**
Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
+1 314 264 8499

**Europe**
Stephenson House
Sir William Siemens Square
Frimley, Camberley
Surrey, GU16 8QD
+44 (0) 1276 413200

**Asia-Pacific**
Unit 901-902, 9/F
Tower B, Manulife Financial Centre
223-231 Wai Yip Street, Kwun Tong
Kowloon, Hong Kong
+852 2230 3333

## About Siemens Digital Industries Software

Siemens Digital Industries Software is driving transformation to enable a digital enterprise where engineering, manufacturing and electronics design meet tomorrow. Xcelerator, the comprehensive and integrated portfolio of software and services from Siemens Digital Industries Software, helps companies of all sizes create and leverage a comprehensive digital twin that provides organizations with new insights, opportunities and levels of automation to drive innovation. For more information on Siemens Digital Industries Software products and services, visit siemens.com/software or follow us on LinkedIn, Twitter, Facebook and Instagram. Siemens Digital Industries Software – Where today meets tomorrow.

**siemens.com/eda**



83647-C1 4/21 TGB