Effective Verification for DO-254 Projects

David Landoll
Applications Architect
May 1, 2008
Agenda

- Verification Challenges and Safety-Critical Design
- DO-254 Requirements for Verification
- Safety-Critical Verification: Recommendations
- Conclusion
Verifying for Safety Critical

- Know that end product is customer safe
- Track that requirements are *thoroughly* verified
- Ensure verification processes to meet compliance to pertinent standards
- Keep project on schedule
Boeing 787: Integration’s Next Step
From its central processor to its common data network, surveillance system and navigation system, the theme of the Boeing 787 Dreamliner is integration.
James W. Ramsey
Verification Challenges

- Growing Gate Counts
- Highly Integrated Devices
- Very Difficult to Verify!

Major Issues

- Verifying Requirements (100’s, 1000’s, etc)
- Increasing Design Concurrency
- Many Clock Domains (Metastability)
- (Exhaustively) Verifying Safety-Critical Reqs

Is Verification Complete?

HDL Design Files
Questions for the Verification Manager

- How do you know your tests really do comprehensively verify the requirements?
  - Design performs its intended function

- How do you ensure you’re testing the interactions between requirements (i.e., concurrency)?
  - Design has no unintended functionality

- How do you ensure you catch anomalous behaviors that might not be tied to requirements?

- How do you manage your verification effort, measure your progress, and prove that you’re done?
Agenda

- Verification Challenges and Safety-Critical Design
- DO-254 Requirements for Verification
- Safety-Critical Verification: Recommendations
- Conclusion
The purpose of DO-254 is design assurance.

Designs must work as intended.

Quality verification is essential.

Verifying complex designs is very challenging.

Note: In this presentation we will not be talking about testing the physical HW item, even though this is a requirement of “verification” for DO-254.
Verification Independence so designer doesn’t test own code

Requirements-based test on both RTL and Gate-Level design representations (as well as end hardware item)

Traceability from Requirements to tests and results

Coverage to ensure verification is complete

Advanced Methods for level A/B projects

Reporting data for audits and management
Verification

What Does Your Business Require?

An effective verification plan to drive all verification activities

Cost effective methods to ensure profitability

Resources used wisely

Metrics for monitoring progress and completion

Assurance of high quality results

Compliance to DO-254 requirements
Agenda

- Verification Challenges and Safety-Critical Design
- DO-254 Requirements for Verification
- Safety-Critical Verification: Recommendations
- Conclusion
Directed Test

A Traditional Approach

- A good approach for traditional design styles
- Manually-written tests exercise requirements via specified stimulus
- Testbench applies stimulus/checks results
- Log file includes results of test
- Code coverage metrics determine if tests exercise RTL code

Note: This method begins to fail with increased device complexity, integration and a large number of requirements
Traditional Coverage Limitation

Test Bench

- These bugs exist, but are undetected
- Failures only appear if test propagates it to the output

Stimulus

Directed Tests

Expected Output

* Tests Pass
* 100% coverage

Lurking bugs:
Missed by Traditional Coverage

Design Under Test

• Tests Pass
• 100% coverage

* Code coverage
* Branch coverage
* Toggle coverage
Evolution of Verification Methods

- Most aerospace companies use this traditional approach (directed test/code coverage)
- More complex designs can benefit from newer techniques
Automating Test Stimulus vs. Directed Test

- **Directed tests:**
  - Test writer must code each specific scenario to specify intent explicitly
  - Prone to overestimating completeness of testing
  - Doesn’t scale with design complexity

- **Automated test stimulus:**
  - Engine uses constraints and randomness to exercise a wide variety of possible scenarios
  - Completeness driven by progress towards functional coverage goals
  - Scales very efficiently with design complexity
How do you build an Automated Testbench?

1. **Engineer** encodes traffic structure and rules per requirements (Testbench)
2. **SystemVerilog Simulator** then chooses paths (Stimulus), per rules (if any)
3. Coverage measurements assures all paths taken per requirements
Directed Test vs. Automated Test Stimulus

**Directed Test**
- 1 test/scenario (1 day each)
- Immediate progress!

**Automated Test Stimulus**
- Up front infrastructure
- 5X productivity increase!

5 verification engrs
500 requirements

Week2  Week4  Week6  Week8  Week10  Week12  Week14  Week16

Coverage Goal

Add assertions  Set up CR test and cover points  environment
Monitoring and Covering Requirements

Assertion Based Verification

- Assertions are like comments that describe how the design is supposed to work (requirements)
- They *actively monitor the design* to ensure it does!
- Assertions provide traceability to requirements

**Requirement**

“The flight crew shall be aurally warned if the gear is down but not locked”

**Assertion**

```verilog
property RQ62_LANDING_GEAR_LOCK;
  @(posedge clk)
    GEAR_down_notification |->
    ##[1:0] Gear_down_lock_notification;
endproperty
cover property RQ62_LANDING_GEAR_LOCK;
```
Automated Test Generation Applied to DO-254

Modern Testbench Approach

- More complete verification
- Requires fewer directed tests/resources
- Direct link back to requirements
Formal Methods vs. Directed Test

- **Directed tests**
  - Simulation-based method that requires input stimulus
  - Test writer must code a scenario that hits a bug
    - If stimulus doesn’t exercise a bug, the bug is missed

- **Formal Methods**
  - Mathematical analysis done on RTL
    -- no stimulus needed
    - Assertion provides description of requirement to be checked
  - Formal engine analyzes assertion against every possible scenario (state)
    - Exhaustive!

*Note: Formal methods should be used in conjunction with (not as a replacement for) directed test and/or automated testing.*
Example: Formal Model Checking for DO-254

Exhaustively Verify Safety-Specific Requirements

- Formal Model Checking finds all possible scenarios
  - Example: enabling reverse thrusters

- Unexpected paths to this situation are called “sneak paths”
  - Is there any way for some event to happen other than the correct way?

- How to apply:
  - Add an assertion stating that the event cannot happen in implementation
  - Apply formal model checking
  - Investigate/fix all unwanted situations
  - Repeat process until no unwanted paths exist

Requirement
Reverse thrusters shall never fire in mid-air.

Assertion
assert always fire_reverse_thrusters |-> Gear_down_lock_notification @(clk'event and clk = '1')
# Managing Verification for DO-254

<table>
<thead>
<tr>
<th>Needs</th>
<th>Mentor Provides the Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements-based test and traceability</td>
<td>- Verification activities mapped to requirements-driven test plan with links for traceability</td>
</tr>
<tr>
<td>Coverage</td>
<td>- Unified coverage database to store coverage data from a variety of sources, with a variety of metrics</td>
</tr>
<tr>
<td>Verification Mgmt and Reporting data</td>
<td>- Verification management facilitates reporting of progress (coverage) of requirements</td>
</tr>
</tbody>
</table>
Verification Management and Unified Coverage
Quality, Progress and Requirements Traceability

Verification Plan Import (Excel, Word, DocBook)

Generate Coverage
- Code coverage
- Functional coverage
- Assertion coverage

Test Merging

Unified Coverage Database

Simulation
Emulation
Formal
CDC
Verification with Mentor

Advanced Methods

- Assertions
- Auto Test Stimulus
- Functional Coverage
- Verification Management and Unified Coverage

Mentor Leads in Advanced Verification

- Actively monitor adherence to requirements
- Automated stimulus generation to reach many more scenarios than directed test
- Measure coverage against design requirements
- Manage and report on verification progress
- Mathematical analysis to exhaustively prove safety-critical requirements, …
- Check clock-domain crossings to eliminate metastability
- Assure two models are functionally equivalent
- Virtual lab for design and analysis of distributed mechatronic systems

Advanced methods can improve both safety and efficiency!
Agenda

- Verification Challenges and Safety-Critical Design
- DO-254 Requirements for Verification
- Safety-Critical Verification: Recommendations
- Conclusion
Conclusion

- Mentor can help you establish a methodology that is efficient, reusable, and certifiable
  - Industry leading solutions in wide use
  - Supporting DO-254 objectives
  - Scalable methods for the simplest to the most complex safety-critical project

- Applying advanced methods will:
  - Improve verification efficiency and thoroughness
  - Reduce development costs
  - Improve safety of hardware systems
More Information

- Visit our web site: www.mentor.com/go/do-254
- Here you will find numerous resources including the following verification-related publications
  - “Achieving Quality and Traceability in FPGA/ASIC Flows for DO-254 Aviation Projects”
  - “The Use of Advanced Verification Methods to Address DO-254 Design Assurance”
  - “Effective Functional Verification Methodologies for DO-254 Level A/B and Other Safety-Critical Devices”
  - “Assessing the ModelSim Tool for Use in DO-254 and ED-80 Projects”
  - “Automating Clock-Domain Crossing Verification for DO-254 (and other Safety-Critical) Designs”
  - “DO-254 Compliant Design and Verification with VHDL-AMS”