Improving the Quality of Complex Control Logic Design using Model Verification

Shobhit Shanker
Senior Application Engineer- Code Generation & Verification

Prashant Hegde
Pilot Engineer
Agenda

- Challenges of testing Complex Control Logic
- Gaps in Traditional Methods of Model Testing
- Improving Quality of the Model by Using Simulink Design Verifier
  - Automatically Generate Test Cases
  - Formally Prove Safety Critical Requirement
  - Detect Design Errors
A Show of Hands…

- How many of you have been involved in testing complex algorithms?
- You spent hours and days weeks writing test cases and manually executing test-cases
Challenges in Testing Complex Control Logic

- Let’s look at the problem we are trying to solve…
  - What is challenge in testing complex control logic?
Difficulties with Complex Logic

- Multiple entry/exit transitions
- Simultaneous transitions
- Execution order
Difficulties with Complex Logic

- Complex conditional statements
Difficulties with Complex Logic

- Synchronization
  - Flag set in one state is being used in another parallel state.
How Do We Test Our Models Today?

- Develop test inputs
  - Types of inputs:
    - Functional requirement
    - High level use case
    - Sensitivity analysis
    - “Edge Case” testing
    - Robustness testing
    - …etc

- Apply the test inputs to model
- Analyze the results
  - Expected output
- Measure coverage
Brief Review on What is Model Coverage...
What is Model Coverage?

- A quick example...

**MC/DC Coverage:** Each condition independently changes the decision outcome.

```plaintext
if (X & Y) 
  Z = 1;
else 
  Z = -1;
end
```
Why is Model Coverage Important?

- It indicates to what extent the test procedure exercise the model structure.
- Reveal model structure that was not exercised during testing.
The quality of your verification activity is based on the input vectors you created. Error or design defect can only be detected if the proper stimulus exists.
Gaps in Simulation Based Testing

- The method itself is inefficient in being exhaustive or complete.
  - A set of functional test case that meets MC/DC coverage objective is only a “minimum” set of test cases.
    - Test cases *can’t* cover every possible combination of different scenarios.
    - Inputs are defined based on what we already knew.

- What about the unknown?
  - Priority, synchronization, timeout…etc

- We are performing *Acceptance* test.
**Simulation Based Testing Pains**

- Writing exhaustive requirement based test cases
- Creating test simulation environment and generating reports
- Missing coverage in the model
- Unexpected behavior in model functionality

*Is there any method/tool to address this?*

---

**Why do we have missing coverage?**

**Simulink Design Verifier**

Generate Test Vector for Missing Coverage
What is Simulink Design Verifier?

- **Formal Method based design analysis tool….
  - No simulation is involved. Model is analyzed based on mathematical equations**

- **Simulink Design Verifier….
  - Points to design flaws or requirement inconsistencies prior to in-depth testing using manually created test vectors.
    - Discovers logic errors / inefficiencies early.
  - Proves that the model meets key behaviors or requirements under all possible circumstances.**

- **Simulink Design Verifier can support simulation based testing method by…**
  - Generation of structural test vectors based on MCDC objectives.
  - Find missing test cases based on MC/DC objectives.
  - Requirement based test vectors with custom objectives to increase test coverage.
Simulink Design Verifier Tour

<table>
<thead>
<tr>
<th></th>
<th>X</th>
<th>Y</th>
<th>X&amp;Y</th>
<th>MC/DC?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Determine Model Design Errors Before you Test

- Ensure algorithmic logic is structurally correct *during* design phase.
- Early Detection
  - Dead logics,
  - Overflow,
  - Divide by Zero

*Transition expression can never be True*
Automatically Generate Test Vectors

- Quickly create a set of test cases that can be used for
  - Meeting the model structural (MCDC) coverage.
  - Equivalence testing criteria in high-integrity application standards
    - ISO26262
    - Software-in-the-Loop
    - Processor-in-the-Loop
- Identify Missing Coverage with respect to functional/Structural use cases and Generate Test vectors only for missing part.
Prove the Correctness of Your Design

- Use custom input conditions and output objectives to:
  - Create “What If” scenarios.
    - Define signal constraints and see if certain output can occur.
    - If sensor values are in range, can DTC be set?
  - Develop additional test cases that would ensure the robustness of the design.
You can ask SLDV the following questions:

- Can “XXX” ever occur?
  - XXX is the Property of interest. It is a description of requirement or behavior created using Simulink modeling construct.

- Is “XXX” always met by my model?
  - A = B always true?
Simulink Design Verifier : Extras

- Exporting generated test vectors into Excel.
- Importing existing test vectors from Excel.
- Comparison of Simulation output with Expected output.
- Use of existing coverage data to generate test vectors only for missing coverage.
- Automatic creation of test harness and report file.
Increasing Confidence In Your Designs

Confidence

Verification Method

Traceability
Modeling Standards Checking
Simulation Testing
Proving Design Correctness
Summary

- Simulink Design Verifier is a *design analysis* tool.

- It has many different uses cases, and can compliment variety of existing Verification and Validation workflow.

- Guided evaluation is the recommended approach.