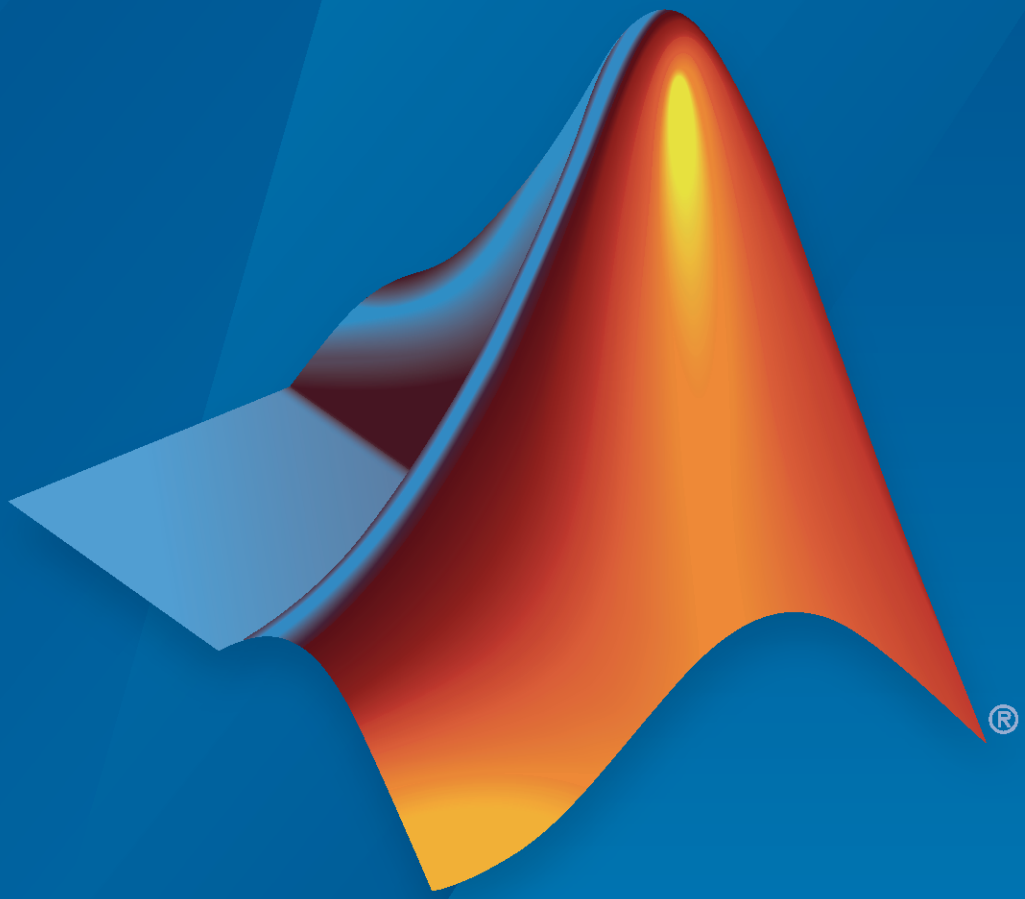
Simulink® Design Verifier™

Getting Started



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

Simulink® Design Verifier™ Getting Started Guide

© COPYRIGHT 2019–2022 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

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 2019	Online only	New for Version 4.2 (Release 2019b)
March 2020	Online only	Revised for Version 4.3 (Release 2020a)
September 2020	Online only	Revised for Version 4.4 (Release 2020b)
March 2021	Online only	Revised for Version 4.5 (Release 2021a)
September 2021	Online only	Revised for Version 4.6 (Release 2021b)
March 2022	Online only	Revised for Version 4.7 (Release 2022a)
September 2022	Online only	Revised for Version 4.8 (Release 2022b)

Contents

1 Simulink Design Verifier Product Description

Simulink Design Verifier Product Description **1-2**

2 Getting Started with Simulink Design Verifier

About Systematic Model Verification Using Simulink Design Verifier . . . **2-2**
When to Use Simulink Design Verifier **2-2**
Using Simulink Design Verifier in a Model-Based Design Workflow **2-2**
Creating Analysis Result Reports **2-3**

Detect Design Errors in Controller Model **2-5**
Step 1: Prepare Model for Design Error Detection **2-5**
Step 2: Perform Design Error Detection Analysis **2-6**
Step 3: Review the Analysis Results **2-7**
Step 4: Fix Design Errors **2-9**

Generate Test Cases for a Simplified Cruise Control Model **2-10**
Analyze a Simple Cruise Control Model **2-10**
Generate Test Cases for Coverage Analysis **2-11**

Simulink Design Verifier Product Description

Simulink Design Verifier Product Description

Identify design errors, prove requirements compliance, and generate tests

Simulink® Design Verifier™ uses formal methods to identify hidden design errors in models. It detects blocks in the model that result in integer overflow, dead logic, array access violations, and division by zero. It can formally verify that the design meets functional requirements. For each design error or requirements violation, it generates a simulation test case for debugging.

Simulink Design Verifier generates test cases for model coverage and custom objectives to extend existing requirements-based test cases. These test cases drive your model to satisfy condition, decision, modified condition/decision (MCDC), and custom coverage objectives. In addition to coverage objectives, you can specify custom test objectives to automatically generate requirements-based test cases.

Support for industry standards is available through IEC Certification Kit (for IEC 61508 and ISO 26262) and DO Qualification Kit (for DO-178).

Getting Started with Simulink Design Verifier

- “About Systematic Model Verification Using Simulink Design Verifier” on page 2-2
- “Detect Design Errors in Controller Model” on page 2-5
- “Generate Test Cases for a Simplified Cruise Control Model” on page 2-10

About Systematic Model Verification Using Simulink Design Verifier

Simulink Design Verifier helps you perform systematic model verification to identify hidden design errors, prove properties, and generate test cases for functional testing. Simulink Design Verifier uses formal methods to test design correctness that increases confidence in your design model that the production code generation uses.

You can perform systematic model verification for scenarios such as:

- Applications that are developed by using model-based design, where you perform design verification to demonstrate that the model satisfies the functional requirements and does not contain unintended functionality.
- Analyzing a subset of a design model that is intended for control software. For open-loop control analysis, formal verification is widely used for rigorous testing of design models.
- Iteratively verifying your model against requirements, checking for design errors, and performing functional testing early in the design cycle and throughout the design process.
- Systematic verification and unit-level testing of small components in isolation or for system-level testing of an integrated design model.

When to Use Simulink Design Verifier

Consider a control engineer who is involved in designing a control system. During the design cycle, the control engineer creates a design model from system requirements. Throughout the development process, the engineer:

- Identifies and eliminates hidden design errors
- Tests model against requirements
- Performs model and code coverage analysis to confirm test completeness
- Resolves missing coverage by using test generation and dead logic detection
- Performs baseline and equivalence testing

Simulink Design Verifier supports these model and code verification processes. It integrates with Requirements Toolbox™, Simulink Coverage™, Simulink Check™, and Simulink Test™ to achieve model and code verification.

Using Simulink Design Verifier in a Model-Based Design Workflow

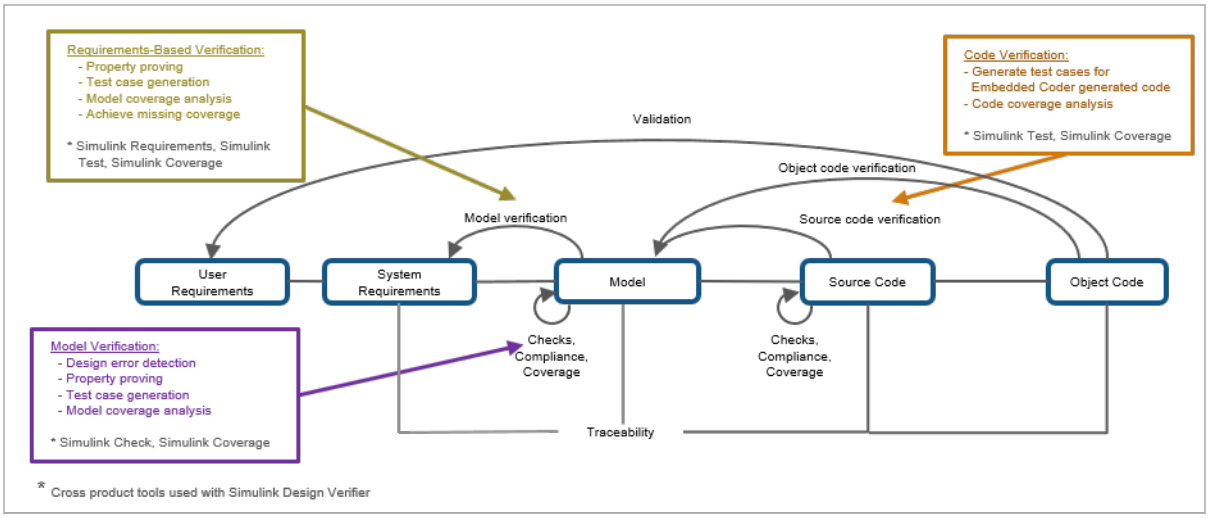
Model verification includes checking against standards, checking for design errors, proving properties, and generating test cases for coverage analysis.

With Simulink Design Verifier, you can:

- Identify hidden design errors, such as integer overflows or division by zero, and generate counterexamples to debug unintended functionalities. You can also justify or exclude model objects from analysis.
- Verify model against requirements by using Requirements Toolbox.
- Achieve model coverage (Simulink Coverage) by generating test cases that satisfies model coverage objectives.

- Perform code coverage (Embedded Coder) analysis by generating test cases for code generated by Embedded Coder®.
- Extend existing test cases and achieve missing coverage.
- Integrate test cases with Simulink Test to perform baseline and equivalence testing.
- Support industry standards through the IEC Certification Kit (for IEC 61508 and ISO 26262) and DO Qualification Kit (for DO-178).

This workflow diagram demonstrates the capabilities of Simulink Design Verifier at various stages of the verification and validation workflow.



For a quick introduction to design error detection and test generation, see “Detect Design Errors in Controller Model” on page 2-5 and “Generate Test Cases for a Simplified Cruise Control Model” on page 2-10.

To learn more about Simulink Design Verifier analysis, see “Detect and Address Bugs”, “Generate Tests”, “Prove Properties in a Model”, and “Review Analysis Results”.

Creating Analysis Result Reports

You can also generate reports and review the analysis results. There are several ways to review the analysis results:

- Review the analysis results at a glance by highlighting the results on the model.
- Create a test harness model to simulate the test cases or debug counterexamples.
- Generate a model coverage report.
- View generated tests in the Simulation Data Inspector.
- Generate an HTML or PDF report that contains detailed information about the analysis results.

See Also

More About

- “Basic Workflow for Simulink Design Verifier”

Detect Design Errors in Controller Model

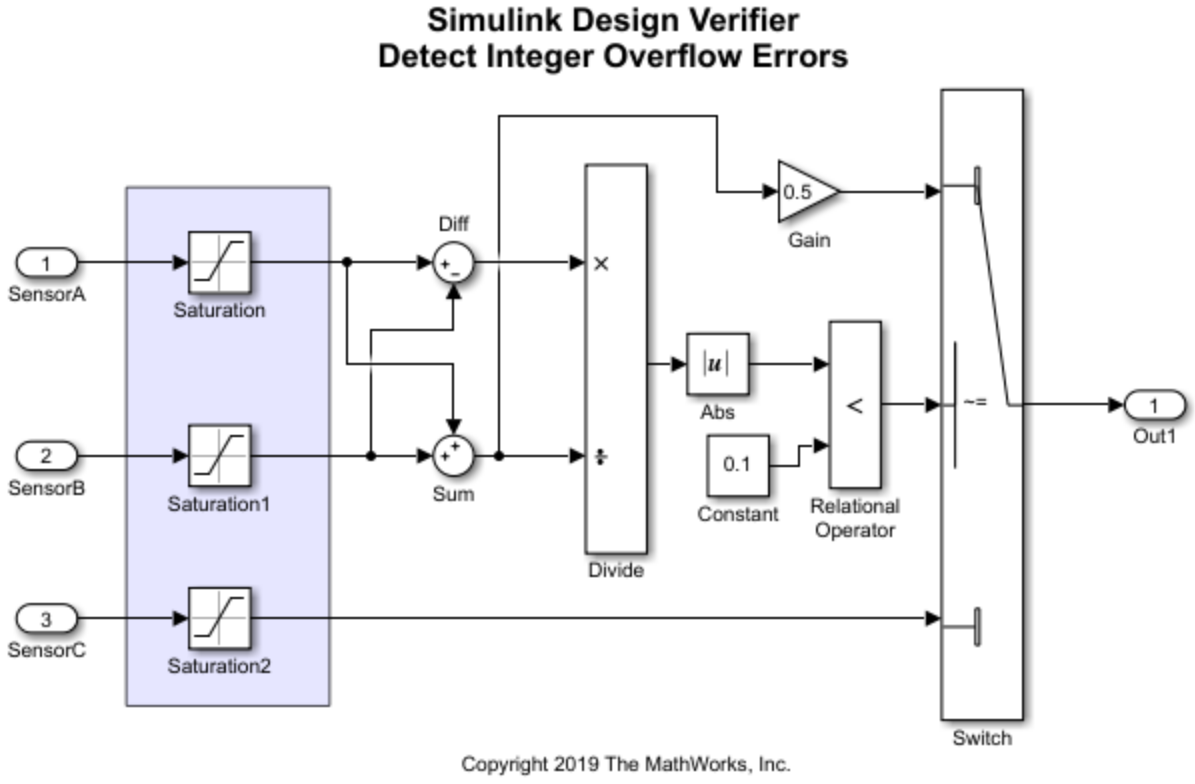
To detect hidden design errors in your model early in the verification process, use design error detection analysis. This tutorial shows how to perform design error detection analysis, review the analysis results, and then fix the identified design errors.

Consider a controller that has three sensor inputs: `SensorA`, `SensorB`, and `SensorC`. The controller algorithm operates according to the equation:

$$If \left|\frac{(SensorA - SensorB)}{(SensorA + SensorB)}\right| > 0.1, then\ output = SensorC$$

$$else\ output = \frac{(SensorA + SensorB)}{2}$$

The algorithm is modeled as:



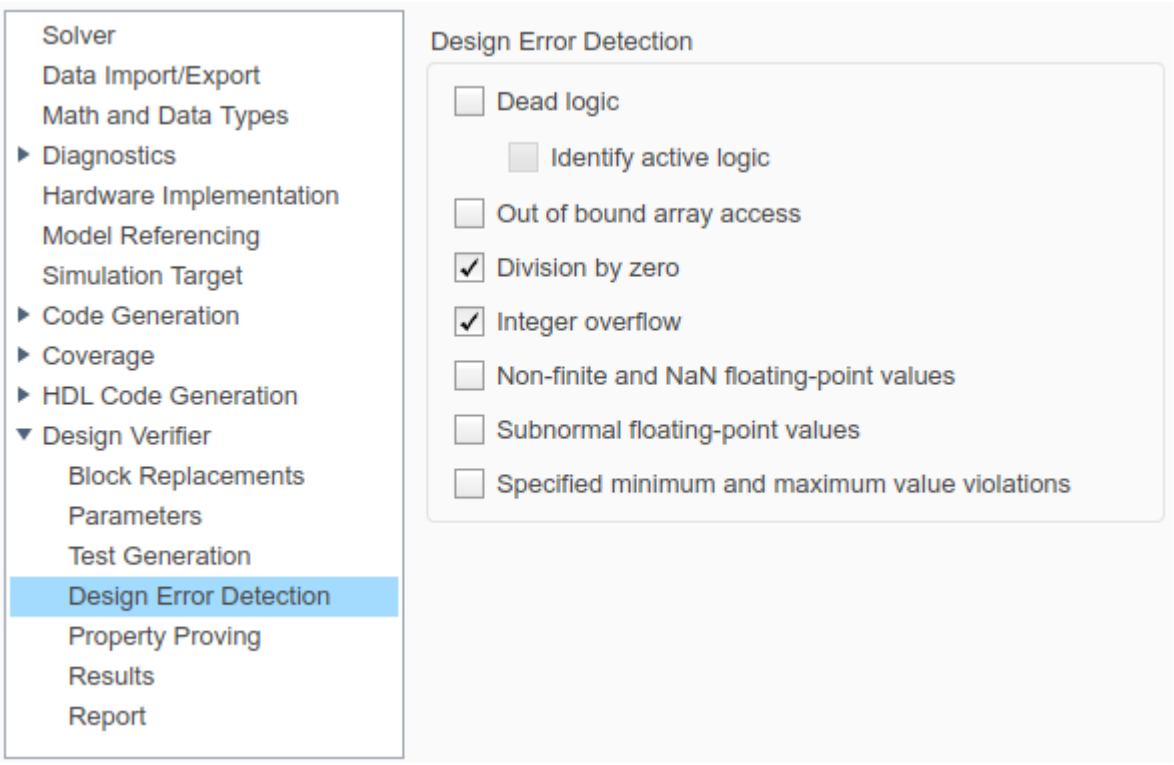
To perform design error detection analysis:

Step 1: Prepare Model for Design Error Detection

1 Open the model sldvexControllerIntegerOverflow.

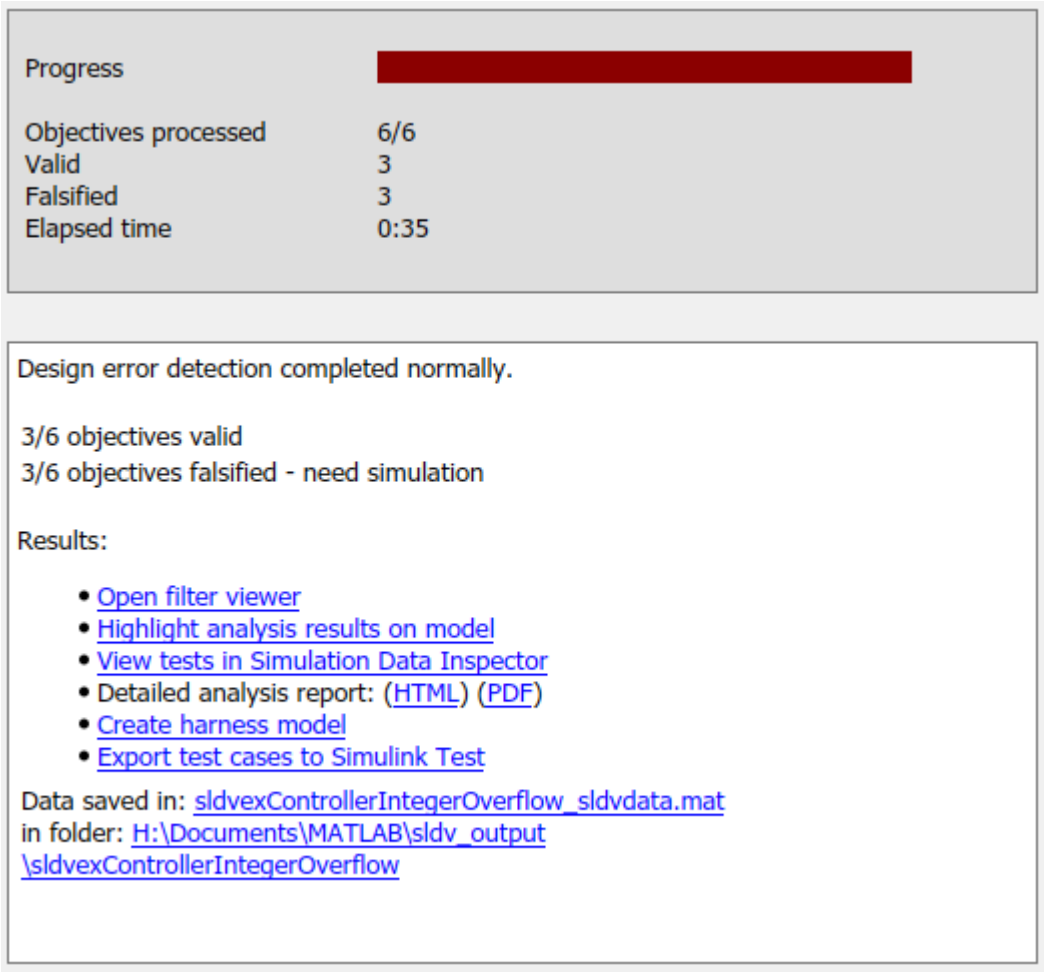
2 On the **Design Verifier** tab, in the **Mode** section, select **Design Error Detection**.

3 Click **Error Detection Settings**. In the Configuration Parameters dialog box, on the **Design Verifier > Design Error Detection** pane, select the checks that you want to perform.



Step 2: Perform Design Error Detection Analysis

To perform design error detection analysis, on the **Design Verifier** tab, click **Detect Design Errors**. The software analyzes the model for design errors and displays the results in the Results window. The results indicate that three out of six objectives were falsified.



Step 3: Review the Analysis Results

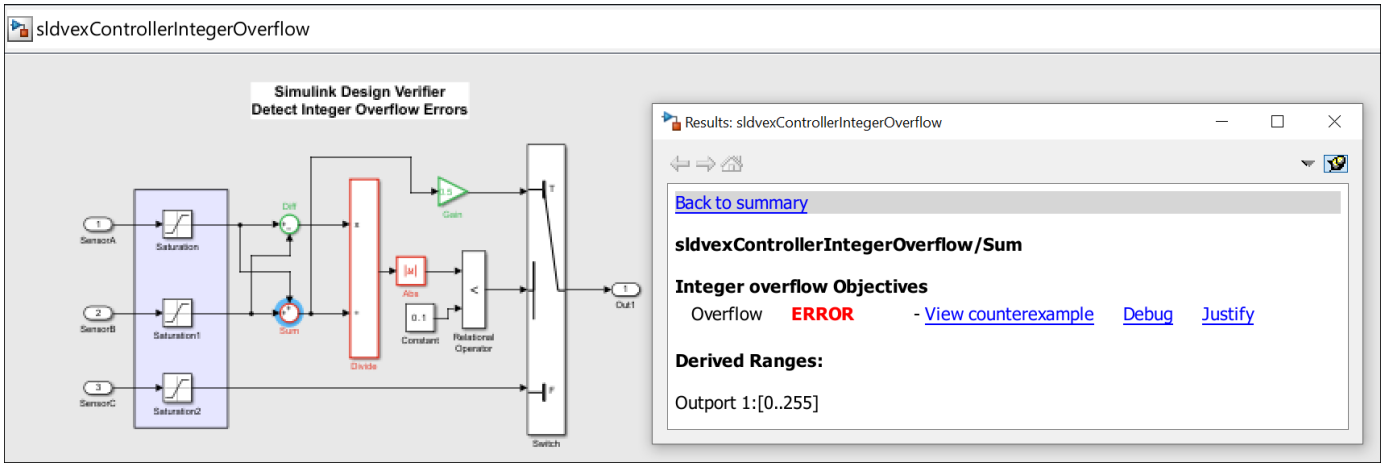
You can review the analysis results by highlighting the results on the model and reviewing the analysis report.

Highlight Analysis Results on the Model

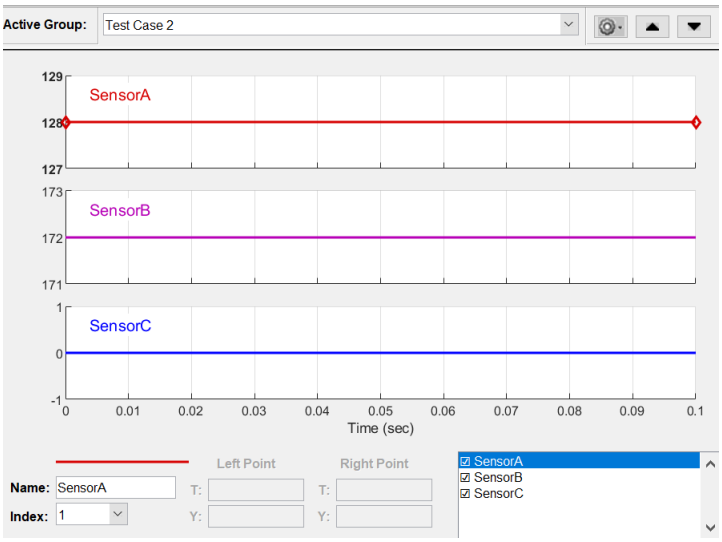
On the **Design Verifier** tab, in the **Review Results** section, click **Highlight in Model**.

The model items highlighted in red are the objectives that resulted in errors. You can replicate the error by simulating the test case.

1 Select the Sum block. The Results window displays the integer overflow objectives of the Sum block.



2 To debug the integer overflow error, click **View counterexample**. The harness model and the Signal Builder block open.



When the input value of `SensorA` is `128` and `SensorB` is `172`, the Sum block output overflows. The accumulator data type of the Sum block is set to an incorrect integer value of `uint8`, it results in overflow errors and division by zero errors on the downstream Divide block.

Review Analysis Report

To view the HTML report, in **Review Results**, click **HTML Report**. The Design Error Detection Objectives section lists the objectives of each model items and their description.

3.1. Objectives Valid

#	Type	Model Item	Description	Analysis Time (sec)
6	Integer overflow	Diff	Overflow	16
9	Integer overflow	Divide	Overflow	41
14	Integer overflow	Gain	Overflow	16

3.2. Objectives Falsified with Counterexamples

#	Type	Model Item	Description	Analysis Time (sec)	Test Case
4	Integer overflow	Sum	Overflow	44	2
8	Division by zero	Divide	Division by zero	41	1
11	Integer overflow	Abs	Overflow	45	3

Step 4: Fix Design Errors

In the tutorial model, the design error detection analysis found integer overflow and division by zero errors in the model. The errors were caused due to a mismatch in the integer data type.

To fix the errors, change the Accumulator data type and Output data type to `uint16` to handle the range of possible signal values. Select the Sum block, and set **Accumulator data type** and **Output data type** to uint16. When you rerun the design error detection analysis, the Results Summary window reports that `6/6` objectives are valid. When you simulate the test case for the Sum block, observe that the test case resolves the error.

See Also

“Design Verifier Pane: Design Error Detection” | “Objectives Status Chapters”

More About

- “What Is Design Error Detection?”
- “Workflow for Detecting Design Errors”

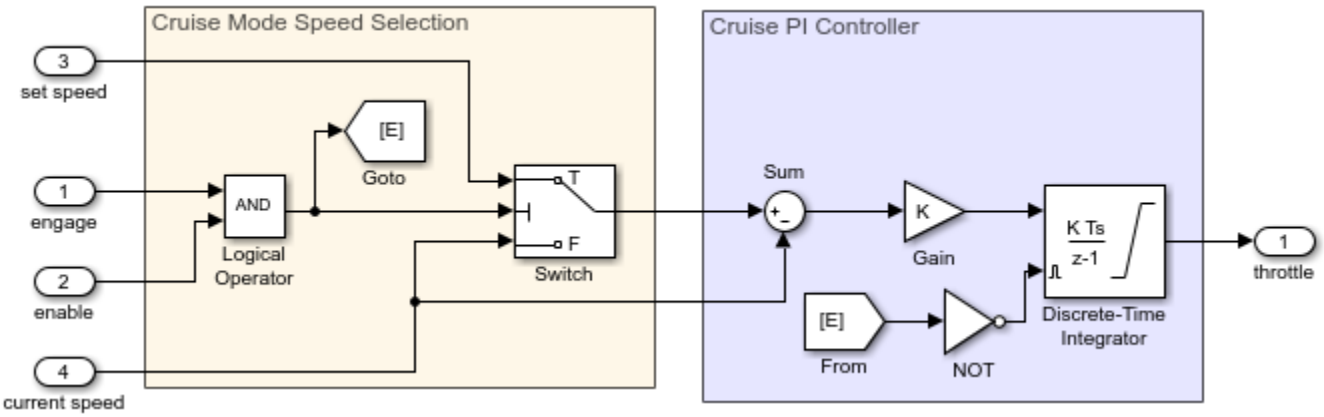
Generate Test Cases for a Simplified Cruise Control Model

With Simulink Design Verifier, you can generate test cases for model coverage and custom objectives. You can also measure coverage of existing requirements-based test cases and extend these test cases to increase coverage or achieve full coverage.

This tutorial explains a simplified cruise control model that controls the throttle speed. You generate test cases that satisfy condition and decision model coverage objectives and then, simulate these test cases to generate the model coverage report.

Analyze a Simple Cruise Control Model

Consider a simplified cruise control model that adjusts the throttle to maintain a steady speed as set by the set speed.



This cruise control model meets these requirements:

- The control system is activated when the `engage` and the `enable` signals are `true`. This condition is defined by the AND block.
- When the system is activated, the Switch block passes `set speed` to the PI controller. The PI controller calculates the `throttle` by integrating the error term defined by the difference `set speed - current speed`.
- `Throttle` continues to increase or decrease until `set speed` is higher or lower than `current speed`.
- When the system is not activated, the Discrete-Time Integrator block resets. The error term is `zero`, which means the `throttle` is in reset position.

When you perform test generation analysis, Simulink Design Verifier generates test cases for the model coverage objectives associated with each model item in the model. The table lists the condition and decision coverage objectives for the associated Model blocks. For more information on model coverage objectives, see “Model Coverage Objectives for Test Generation” and “Model Objects That Receive Coverage” (Simulink Coverage).

Block	Model Coverage Objective	Generated Test Case Description
AND	Condition	Each input value is set to true or false independently.
NOT	Condition	Input is set to true or false independently.
Switch	Decision	Test case demonstrates that the Switch passes both the input signals to output.
Discrete-Time Integrator	Decision	• Test case demonstrates the saturation behavior of the integrator. • Satisfies external reset conditions.

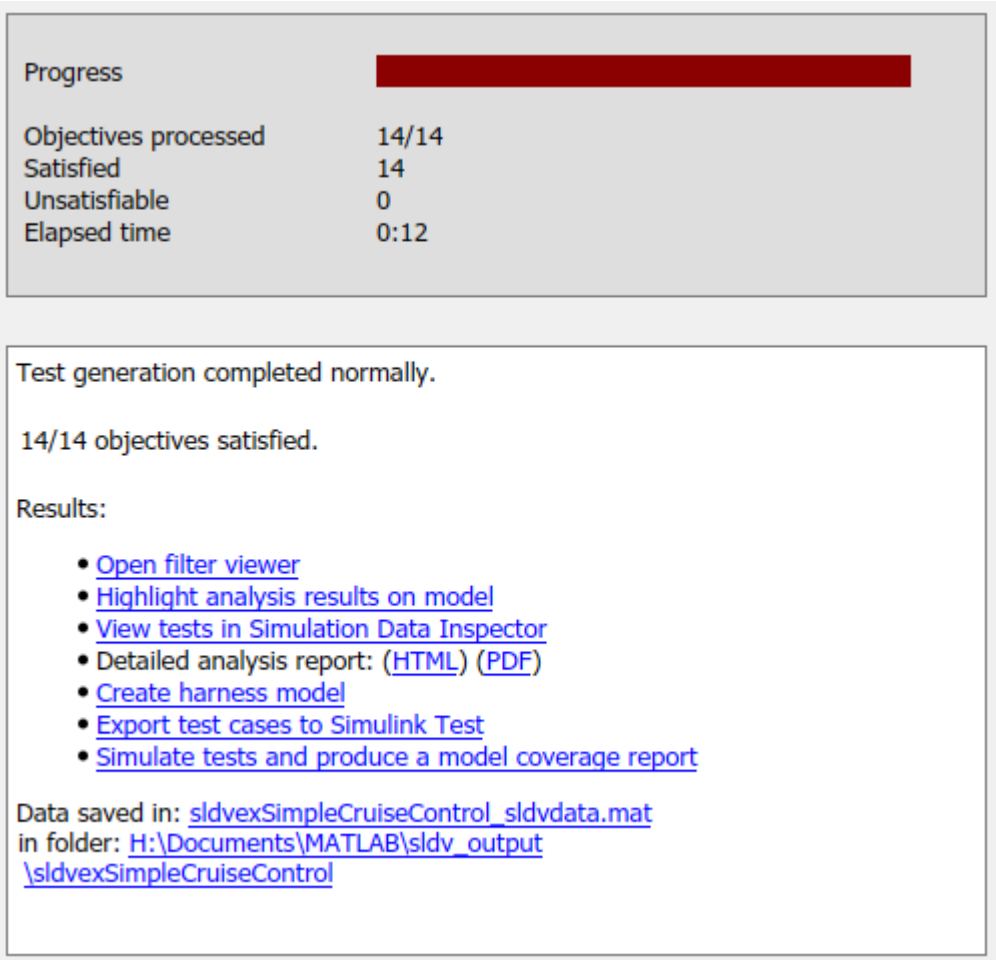
Generate Test Cases for Coverage Analysis

The analysis results give detailed descriptions of the coverage objectives for each model item and generated test cases for all satisfied objectives. You simulate the generated test cases to measure model coverage.

Step 1: Generate Test Cases

1. Open the model sldvexSimpleCruiseControl.
2. On the **Design Verifier** tab, in the **Mode** section, select **Test Generation**.
3. To generate test cases, click **Generate Tests**.

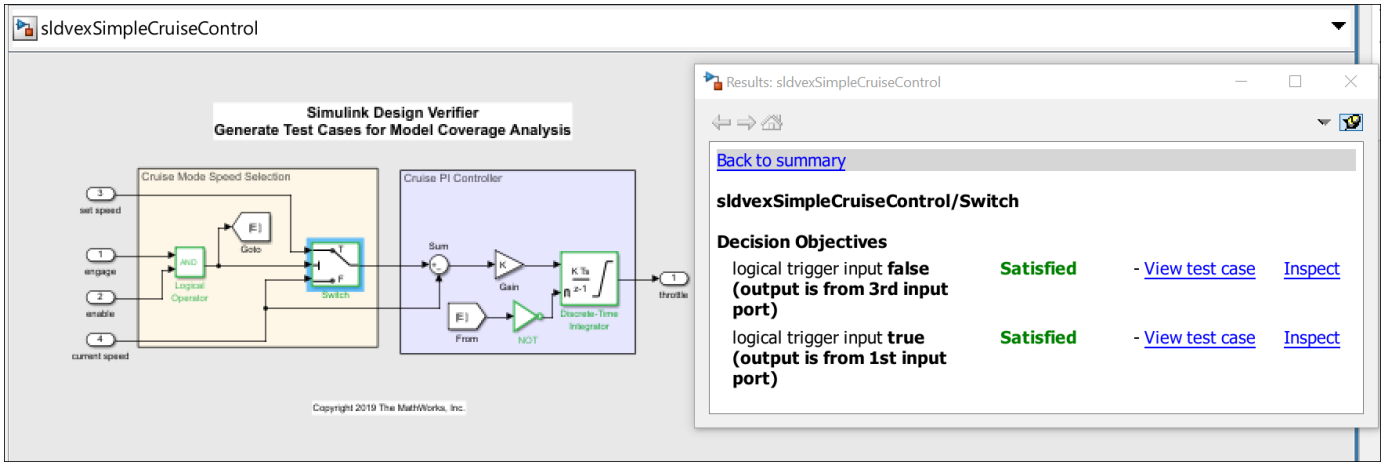
   The Results Summary window displays the results. The result indicates that all 14 objectives are `satisfied`.



Step 2: Review the Analysis Results

1 On the **Design Verifier** tab, in the **Review Results** gallery, click **Highlight in Model**. The model objectives that the software found to be satisfied are highlighted in green.

Click the Switch block. The Results window displays the summary of the satisfied decision objectives.



The summary shows that all the objectives of the Switch block are satisfied.

2 To view the HTML report, in the **Review Results** gallery, click **HTML Report**.

The test objective status section includes detailed descriptions of satisfied objectives for each model item and generated test case.

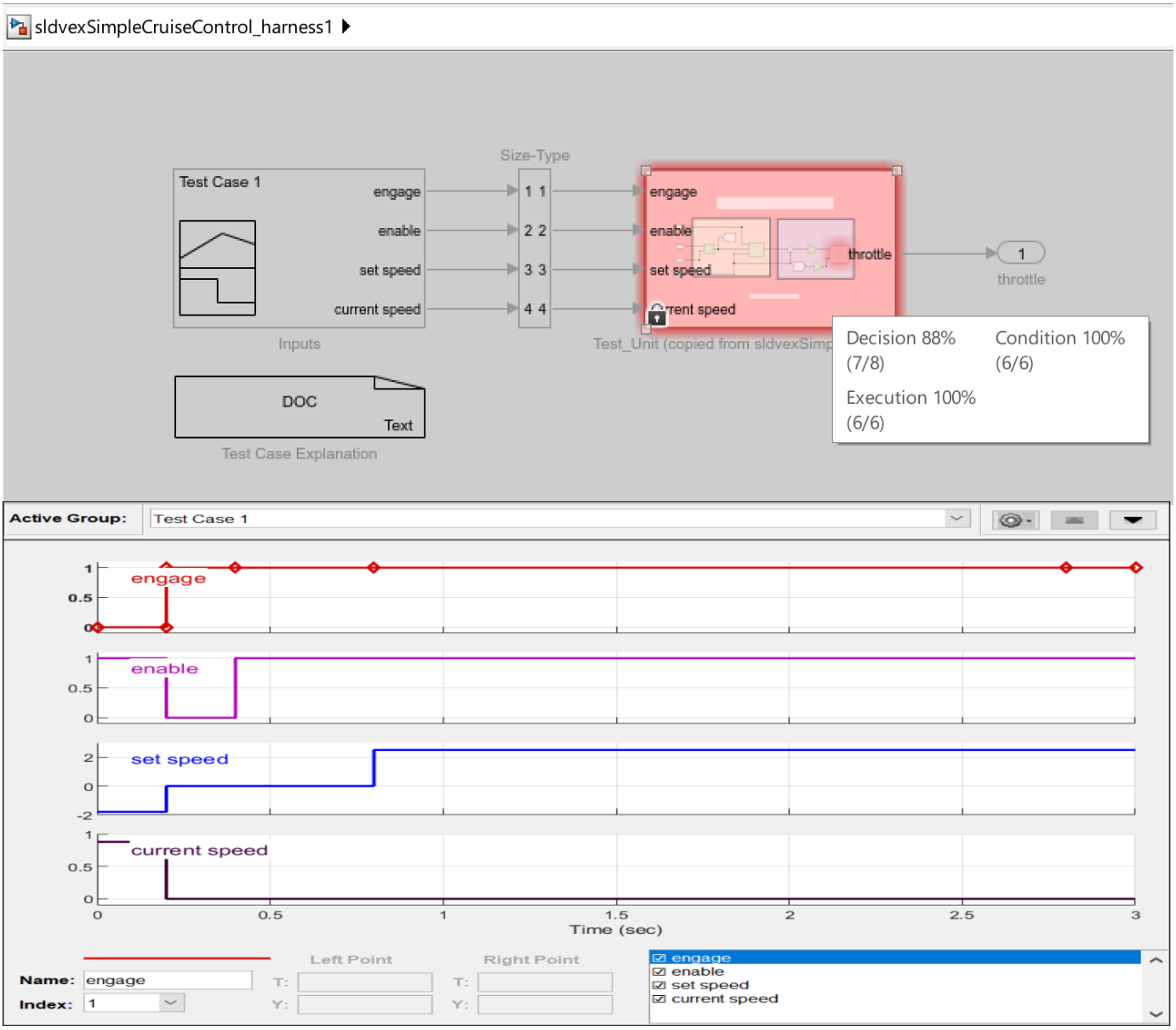
#	Type	Model Item	Description	Analysis Time (sec)	Test Case
1	Condition	Logical Operator	Logic: input port 1 **true**	10	1
2	Condition	Logical Operator	Logic: input port 1 **false**	10	1
3	Condition	Logical Operator	Logic: input port 2 **true**	10	1
4	Condition	Logical Operator	Logic: input port 2 **false**	10	1
5	Condition	NOT	Logic: input port 1 **true**	10	1
6	Condition	NOT	Logic: input port 1 **false**	10	1
7	Decision	Discrete-Time Integrator	Reset **true**	10	1
8	Decision	Discrete-Time Integrator	Reset **false**	10	1
9	Decision	Discrete-Time Integrator	integration result <= lower limit **true**	10	2
10	Decision	Discrete-Time Integrator	integration result <= lower limit **false**	10	1
11	Decision	Discrete-Time Integrator	integration result >= upper limit **true**	11	3
12	Decision	Discrete-Time Integrator	integration result >= upper limit **false**	10	1
13	Decision	Switch	logical trigger input **false (output is from 3rd input port)**	10	1
14	Decision	Switch	logical trigger input **true (output is from 1st input port)**	10	1

Step 3: Simulate Test Cases for Model Coverage Analysis

To view the test case for a model coverage objective, in the Results window, click **View test case**. The harness model and the Signal Builder block opens.

To simulate the test case, in the Signal Builder block, click the button.

The software simulates the test case and highlights the harness model. To view the coverage of the model items, point the cursor to each model object in the harness model.



See Also

“Design Verifier Pane: Test Generation” | “Objectives Status Chapters”

More About

- “Workflow for Test Case Generation”
- “Generate Test Cases for a Subsystem”