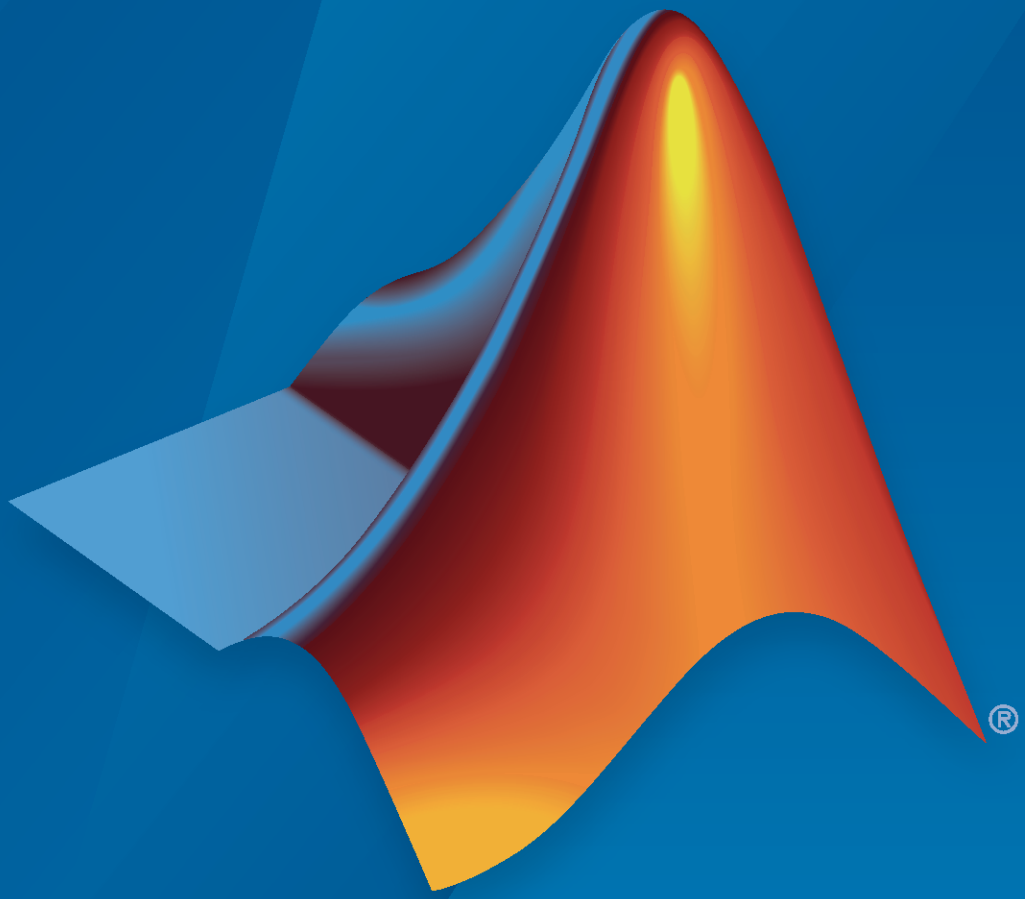
Simulink® Coverage™ Release Notes



MATLAB® & SIMULINK®

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® Coverage™ Release Notes

Contents

R2021a

R2020b

R2020a

R2019b

R2019a

R2018b

R2018a

R2017b

R2021a

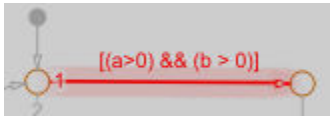
Version: 5.2

New Features

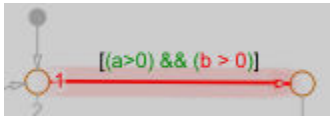
Bug Fixes

Show granular highlighting of model coverage results for Stateflow

In R2021a, coverage highlighting now distinguishes satisfied and unsatisfied coverage objectives within a Stateflow state or transition that uses MATLAB® as the action language. This new behavior matches what is already displayed in the coverage report. For example, consider a transition, `[(a>0) && (b>0)]`, where the condition `a>0` is satisfied and the condition `b>0` is not. In R2020b and before, the transition is entirely colored red.



In R2021a, the satisfied condition is colored green, while the unsatisfied condition is colored red.



For more information, see “Model Coverage Display for Stateflow Charts”.

Collect coverage for observer models

In R2021a, you can collect coverage on observer models and models referenced by observer models. Observer models are treated similarly to model references. When you set **Scope of coverage analysis** to **Referenced Models**, the Select Models for Coverage Analysis dialog now lists observer models in addition to model references. Observer models are distinguished from model references by an **(Observer)** tag.

See “Observer Model” for more information.

Justify unsatisfied outcomes for code coverage

In R2021a, you can justify unsatisfied code coverage outcomes without excluding other outcomes. Once you have generated code coverage results from a model in software-in-the-loop (SIL) or processor-in-the-loop (PIL) mode, you can justify unsatisfied outcomes from within the code coverage report the same way that you would justify unsatisfied outcomes in a model coverage report. The code coverage report now has the same layout as the model coverage report.

For more information, see the new example “Use Justification Rules to Filter Code Coverage Outcomes” and the updated reference pages `slcoverage.CodeSelector` and `slcoverage.SFcnSelector`.

Trace code coverage results to associated test cases

In R2021a, you can trace aggregated code coverage results to associated test cases for models simulated in software-in-the-loop (SIL) or processor-in-the-loop (PIL) mode. The aggregated coverage report now links to the test cases associated with each code coverage outcome. You can collect the aggregated coverage results by using either the Simulink® Test™ Manager or the Coverage Results Explorer. This enhancement means code coverage now links to associated test cases the same way model coverage does. For more information, see “Trace Coverage Results to Associated Test Cases”.

Collect code coverage for atomic subsystems

In R2021a, you can collect code coverage for atomic subsystems by using the Simulink Test Manager to compare normal and software-in-the-loop (SIL) or processor-in-the-loop (PIL) test results. You can create the tests manually or by using the Test for Model Component wizard.

This feature requires Simulink Test and Embedded Coder® licenses.

Collect coverage for tests from multiple releases in Simulink Test

In R2021a, you can use an equivalence test in the Simulink Test Manager to compare coverage across two releases. Coverage results are returned in result sets and you can view both individual and aggregated coverage for the releases. You can export coverage to a MATLAB workspace variable and include it in a generated report. For releases that support coverage filtering, you can automatically add missing coverage. Multiple release coverage supports up to the previous six releases.

While you can load coverage data from previous releases using Simulink Coverage™, you need a Simulink Test license to run tests and collect coverage on them from multiple releases.

R2020b

Load and view coverage data from previous releases

In R2020b, you can load and view coverage data from previous releases as far back as R2017b.

The `cvdata` object now contains a `dbVersion` property that identifies the origin release of the coverage data.

```
[~, cvd] = cvload('myCvData_19b');
cvd = cvd{1};
cvd.dbVersion
ans =
     '(R2019b)'
```

The coverage report contains a new section about the coverage data, which includes a field titled **Collected in version**.

After you import coverage data from a previous release, you can extract information by using the same API that you would use normally. Model highlighting and filtering continue to work as expected. You can aggregate coverage data from two or more `cvdata` objects if the `dbVersion` properties match.

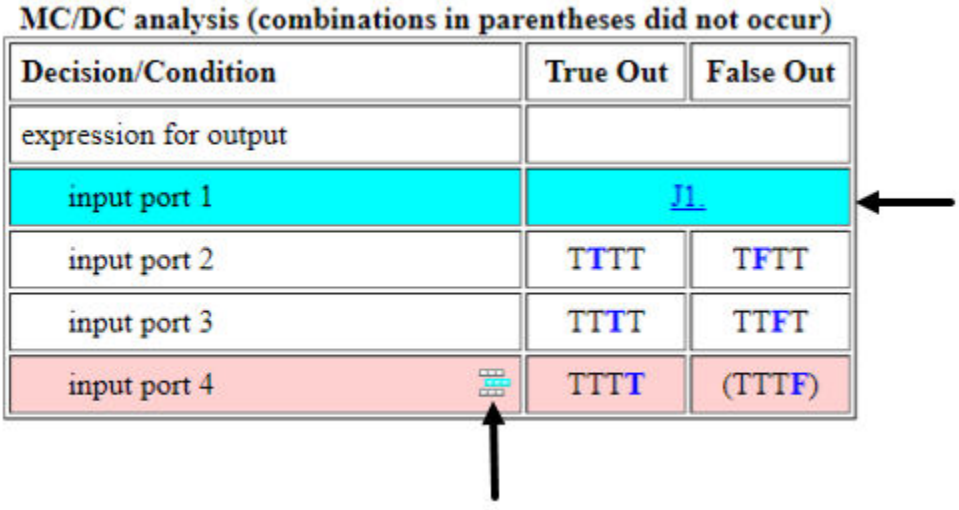
The Test Manager in Simulink Test can show coverage reports for imported coverage data from a previous release. Simulink Design Verifier™ can use coverage data collected in a previous release to generate test cases to achieve missing coverage. For more information, see Achieve Missing Coverage in Referenced Model (Simulink Design Verifier).

It is not possible to export coverage data from the current release to a previous release format.

Justify missing coverage for individual MCDC objectives

In R2020b, you can justify missing coverage for individual MCDC objectives.

The following image shows the model coverage report with a missing coverage objective justified after MCDC analysis.



The input port 1 expression is a missing coverage objective that has been justified after MCDC analysis. The input port 4 expression shows a missing coverage objective and the **Add justification rule** icon.

To justify an MCDC outcome, follow the same procedure as justifying a decision or condition outcome. For a coverage filtering example, see Creating and Using Coverage Filters. For an example that justifies an unsatisfied MCDC objective outcome, see Filter Coverage Results Using a Script.

Collect and view requirements testing data by using the Model Testing Dashboard

The Model Testing Dashboard collects and displays metric data on the status and quality of your requirements-based testing. If you have Simulink Check™ and Simulink Test licenses, you can assess the testing status of a model by using the dashboard to view:

- Summary data on requirements, tests, and traceability between requirements and tests
- Status and results for the latest test runs
- Model coverage measurements achieved by tests and justifications
- A list of the latest artifacts of the project, organized by the associated models

For more information, see “Model Testing Dashboard: Track completeness of requirements-based testing for compliance to standards such as ISO 26262” (Simulink Check).

Simplified coverage configuration parameters

In R2020b, in the Configuration Parameters window, the **Coverage > Results** tab is removed. The parameters previously located under **Results** are now listed under **Coverage**.

Functionality being removed or changed

CovShowResultsExplorer, CovHighlightResults, and CovHtmlReporting are removed

In R2020b, three Simulink Coverage configuration parameters are removed:

- `CovShowResultsExplorer` — To open the Coverage Results Explorer, in the **Apps** tab, click **Coverage Analyzer**. Then click **Results Explorer**.
- `CovHighlightResults` — To highlight coverage results in your model, use `cvmodelview`.
- `CovHtmlReporting` — To create a coverage report, use `cvhtml`.

Compatibility Considerations

Using `CovShowResultsExplorer`, `CovHighlightResults`, or `CovHtmlReporting` does not cause errors in your simulation, but they are ignored and a warning is generated.

This change affects only models run using `sim`. These parameters were already ignored by `cvsim` and simulations run using the **Run** button.

R2020a

Scope model coverage to requirements-based tests

Starting in R2020a, you can scope coverage results to linked requirements-based tests. From the Simulink Test Manager, select **Scope coverage results to linked requirements**. This setting scopes the aggregated coverage results such that each test only contributes coverage for the corresponding model elements that implement the requirements verified by that test. This improves confidence that model elements are covered by the intended test cases.

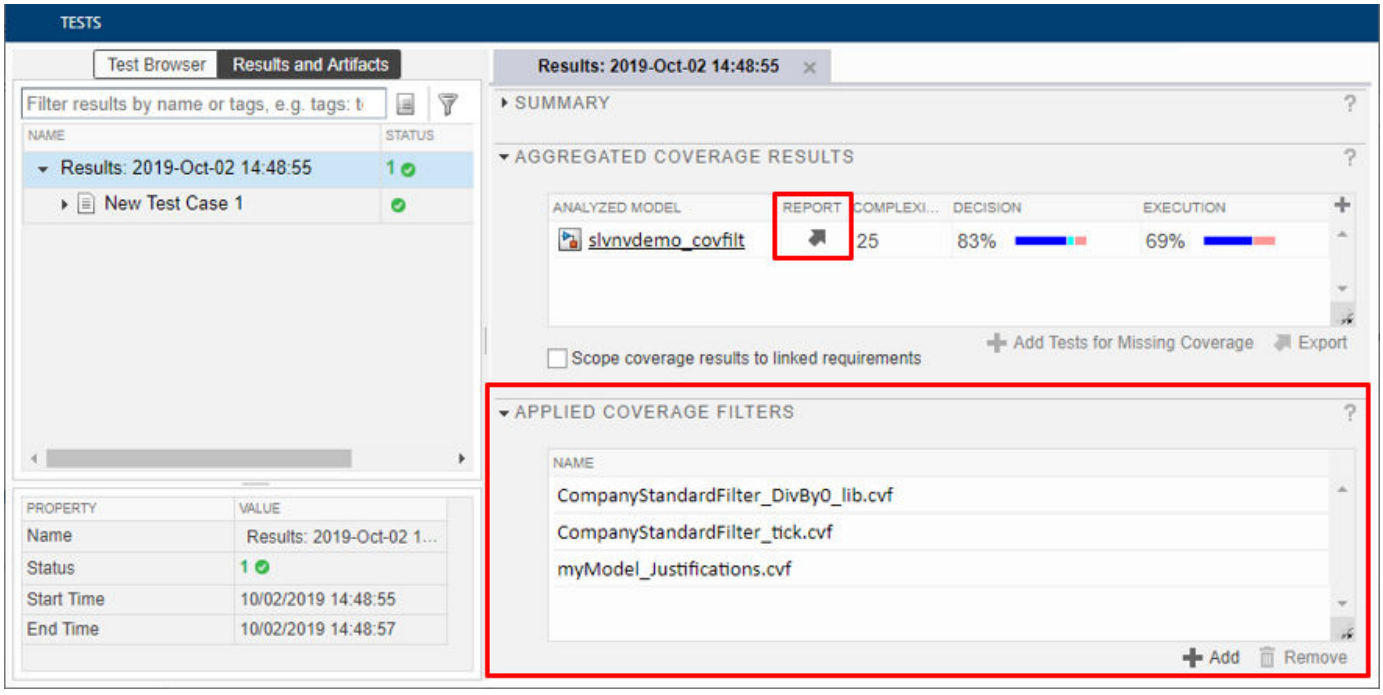
For more information on scoping coverage results to linked requirements, see Assess Coverage Results from Requirements-Based Tests.

For more information on collecting coverage by using the Simulink Test Manager, see Test Coverage for Requirements-Based Testing (Simulink Test). For more information on creating requirements links, see Link Blocks and Requirements (Simulink Requirements).

Manage and view multiple coverage filters

You can apply multiple coverage filters to coverage data:

- Programmatically. For more information, see Use Multiple Filter Files for a Simulation.
- From the **Results Explorer**. For more information, see Creating and Using Coverage Filters.
- From the **Results and Artifacts** pane of the Simulink Test Manager. You can add or remove existing coverage filters from the **Applied Coverage Filters** section or create new filter rules directly from a report created by using the **Report** button in the **Aggregated Coverage Results** section.



- From coverage results. You can create new filter rules from a coverage report by right-clicking a block with model coverage highlighting or from the **Coverage Details** pane. For more information, see View Coverage Results in a Model.

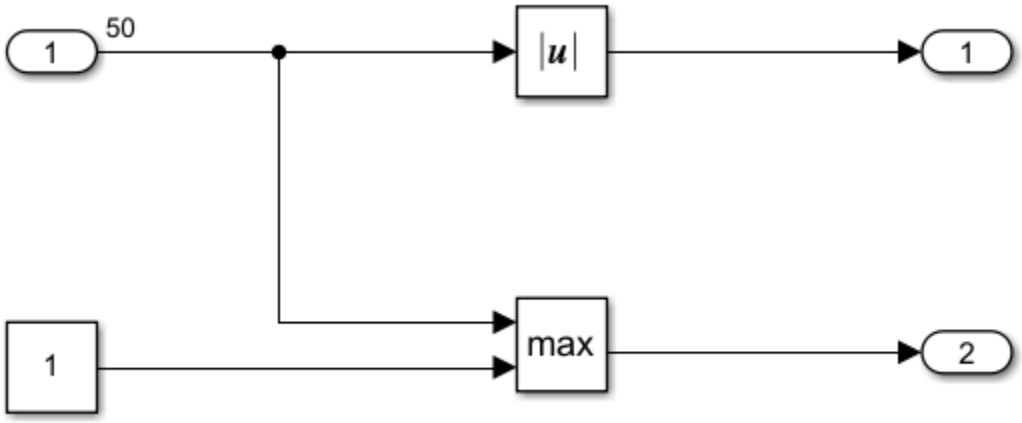
Coverage support for Stateflow variant transitions

Simulink Coverage records coverage for active Stateflow® variants. For more information, see Software Configurations Using Variant Transitions (Stateflow).

Enhanced calculation of cyclomatic complexity

In R2019b, the cyclomatic complexity metric calculation counted each element of a vectorized operation as a separate decision point. As a result, a simple model with a signal that is a vector or matrix could have a high cyclomatic complexity result. In R2020a, the cyclomatic complexity calculation considers a vectorized operation as a single decision point, thereby lowering the cyclomatic complexity result for some models.

For example, consider the following model which has a vector signal of size 50. This signal branches to two signals. Two blocks perform calculations on each signal.



In R2019b, the coverage report stated that this model had a cyclomatic complexity of 101. In R2020a, the coverage report states this model has a cyclomatic complexity of 3.This is a more reasonable value for a simple model. For more information, see Cyclomatic Complexity.

Compatibility Considerations

Previously, models with vectorized operations had higher cyclomatic complexity values than they have in R2020a.

R2019b

Version: 4.4

New Features

Bug Fixes

Unit-to-System Test Coverage Aggregation: View system test coverage achieved from unit tests in new Aggregated Tests section of coverage report

If you use Simulink Test to record coverage for multiple subsystems from the same model, you can aggregate the coverage data for the subsystems into top-level model coverage. You access the aggregated results in the Simulink Test Manager results tab for a given test run.

For more information on recording coverage for test cases in Simulink Test, see Trace Coverage Results to Associated Test Cases and Aggregated Tests.

Requirements-to-Test-Case Traceability in Coverage Report: View Simulink Requirements links and coverage details for each Simulink block when generating coverage reports from Simulink Test Manager

When you collect aggregated coverage from the Simulink Test Manager for a model with associated Simulink Requirements™ links, the coverage report now includes a summary of the associated requirements links for each model element.

For more information, see Trace Coverage Results to Requirements by Using Simulink Test and Simulink Requirements and Requirement Testing Details.

Test Case Traceability of Coverage Results: Trace coverage results to relevant simulations in Simulink Test Manager and Coverage Results Explorer

When you collect aggregated coverage for a model, the coverage report now includes a summary of the associated tests in the Results Explorer or the Simulink Test Manager and links to the first test case which executed each coverage outcome for model elements.

For more information, see Trace Coverage Results to Associated Test Cases and Aggregated Tests.

Simulink Test Integration

If you record coverage from the Simulink Test Manager, the coverage report includes the associated Simulink Test test cases in the **Aggregated Test** summary. The coverage details section for each model element also links to Simulink Test test cases associated with each coverage outcome.

Coverage Toolstrip: Access common coverage features from the new Simulink Toolstrip, including model highlighting, coverage details, and report generation

The Simulink toolstrip includes contextual tabs, which appear when you open the **Coverage Analyzer** app, under **Verification, Validation, and Test**. The Simulink Coverage contextual tab includes options for completing actions that apply only to Simulink Coverage.

For more information, see “Simulink Toolstrip: Access and discover Simulink capabilities when you need them”.

View code coverage information in code view

If you have an Embedded Coder license, you can view code coverage information in the Code view. For more information, see “Code coverage information in Code view” (Embedded Coder).

Extract subsystem coverage data from system-level coverage data

You can now use `extract` to extract the coverage data for a subsystem. For an example, see Create HTML Coverage Report for a Subsystem from Model Coverage Data.

R2019a

Version: 4.3

New Features

Bug Fixes

Lookup table breakpoint value changes in coverage data

In R2019a, when you record coverage for multiple runs of a model with a lookup table, and then change the breakpoint values for the lookup table between runs, the coverage data from the previous runs is compatible with the updated model if the dimensions of the lookup table are the same as in the previous runs.

Simulink Coverage contextual tabs in the Simulink Toolstrip Tech Preview

In R2019a, you can turn on the Simulink toolstrip. See “Simulink Toolstrip Tech Preview replaces menus and toolbars in the Simulink Desktop” for more details.

The Simulink toolstrip includes contextual tabs, which appear when you open the **Coverage Analyzer** app, under **Verification, Validation, and Test**. The Simulink Coverage contextual tab includes options for completing actions that apply only to Simulink Coverage. Documentation does not reflect the addition of the Simulink Coverage contextual tabs.

R2018b

Version: 4.2

New Features

Model Coverage Visualization: Gain enhanced perspective of coverage results through model highlighting and pop-ups within the Simulink Editor

The model coverage perspective allows you to hover over a model element and view a coverage summary for the model element in a tool tip. The **Coverage Details** window allows you to view detailed coverage information for model elements without leaving the Simulink Editor. For more information, see View Coverage Results in a Model.

Parallel Simulation Support: Accelerate coverage analysis through use of parsim

You can now leverage the `parsim` function to record model coverage for multiple simulation runs in parallel. For more information, see Record Coverage in Parallel Simulations by Using Parsim and parsim.

Stateflow Custom Code Support: Collect coverage on elements of Stateflow charts where C/C++ code is used

Simulink Coverage records code coverage for elements of Stateflow charts where C/C++ code is used. For more information on how to enable custom code support, see Coverage for Custom C/C++ Code in Simulink Models.

C Caller Block Support: Perform code coverage analysis for custom C/C ++ code in Simulink models

Simulink Coverage records code coverage for custom C/C++ code in C Caller blocks. For more information on how to enable custom code support, see Coverage for Custom C/C++ Code in Simulink Models.

Coverage Filtering API: Create filter rules for custom C/C++ code in normal mode and generated code in SIL or PIL modes

You can use model coverage commands to filter custom C/C++ code in normal mode and generated code in SIL or PIL modes. For more information, see `slcoverage.CodeSelector` and Automate Coverage Workflows.

R2018a

Version: 4.1

New Features

Fine-grained filtering for relational boundary metrics: Control coverage results for individual design elements

To achieve complete coverage when you record saturate on integer overflow coverage or relational boundary coverage for a model, you can exclude or justify incomplete coverage outcomes from the coverage report.

For more information on coverage filtering, see Create, Edit, and View Coverage Filter Rules. For more information on saturate on integer overflow coverage and relational boundary model coverage, see Types of Model Coverage.

Stateflow Just-In-Time (JIT) Compilation Mode: Reduce model update time when recording coverage

When you record coverage for Stateflow charts, Stateflow uses just-in-time (JIT) compilation technology to improve model update performance. For more information on JIT compilation technology, see Speed Up Simulation.

R2017b

Version: 4.0

New Features

Compatibility Considerations

Simulink Verification and Validation Packaging: Moved model and generated code coverage functionality and component verification functions such as slvnvmakeharness to Simulink Coverage

The model and generated code coverage functionalities and component verification functions of Simulink Verification and Validation™ have been transitioned to Simulink Coverage. For an introduction to the product, a basic Simulink Coverage workflow, and an outline of how Simulink Coverage fits into a systematic, end-to-end verification workflow, see the Getting Started with Simulink Coverage category. For coverage-related release notes for Simulink Verification and Validation prior to R2017b, see https://www.mathworks.com/help/releases/R2017a/slvnv/release-notes.html.

Coverage Filtering API: Filtering choices for coverage justifications that include specified decisions, conditions, and outcomes

You can use a command-line API to create filtering rules for blocks. Selection criteria for filtering includes filtering by individual block ID, filtering for all blocks of the same type, filtering certain decisions, conditions, and outcomes of a block, and more. You can also filter S-Function C++ code by code coverage outcome. For more information and examples, see Automate Coverage Workflows.

Logical Expressions in Assignment Statements: Record Condition and MCDC coverage for logical expressions in assignments in Stateflow and MATLAB Function Blocks

In models where logical expressions are assigned to variables – to break up complicated logic, to reuse common subexpressions, etc. – Simulink Coverage now records Condition and MCDC coverage for the logical expressions in assignment statements. For a detailed example of how Simulink Coverage records Condition and MCDC coverage for models where logical expressions are assigned to variables, see Coverage for MATLAB® Function Blocks.

Compatibility Considerations

Models that use logical expressions in assignment statements in Stateflow and MATLAB Function blocks record an increased number of Condition and MCDC objectives than previously recorded.

Function and Function Call Coverage: Collect SIL & PIL coverage as required by ISO 26262

Simulink Coverage introduces two new metrics for measuring Statement coverage for code.

- Function Coverage: Function coverage determines whether all the functions of your code have been called during simulation.
- Function Call Coverage: Function call coverage determines whether all function calls in your code have been executed.

The new metrics are reported in the top-level Summary and in the Details section of the HTML Coverage Report when you record code coverage.