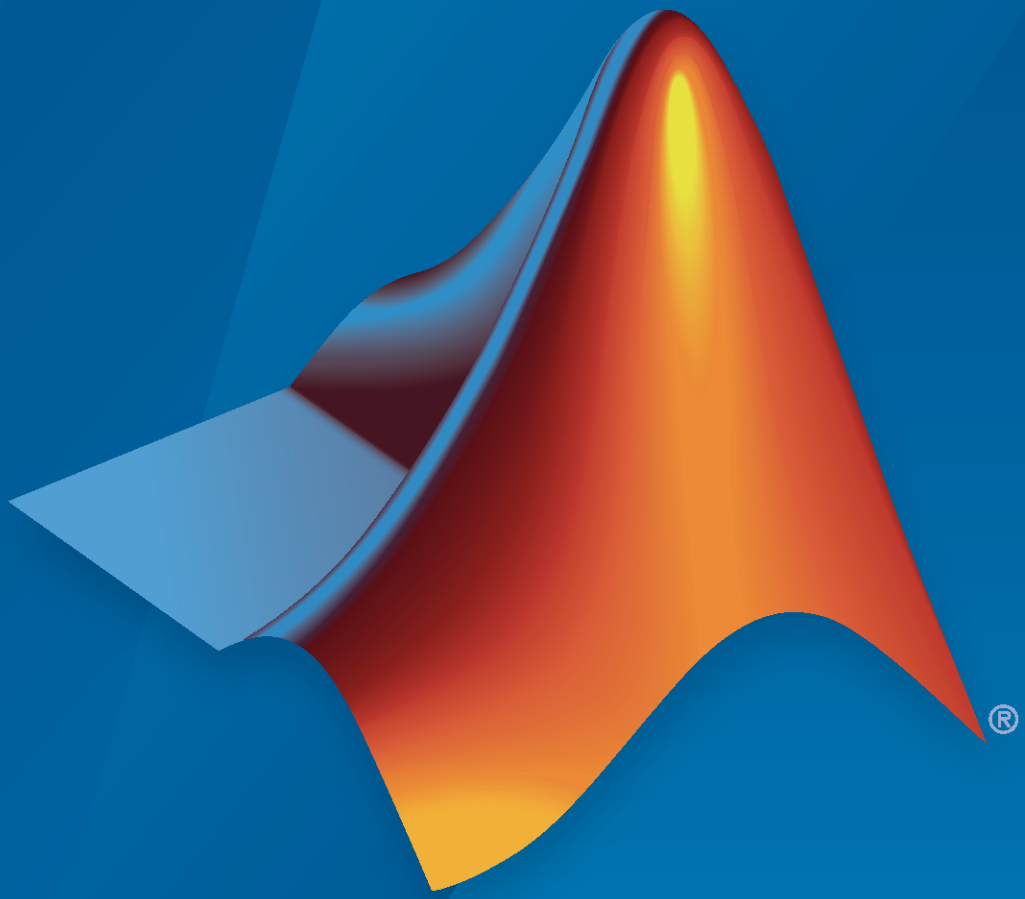
Simulink® Coverage™

Getting Started



MATLAB® & SIMULINK®

R2023a

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™ Getting Started

Revision History

September 2017	Online only	New for Version 4.0 (Release 2017b)
March 2018	Online only	Revised for Version 4.1 (Release 2018a)
September 2018	Online only	Revised for Version 4.2 (Release 2018b)
March 2019	Online only	Revised for Version 4.3 (Release R2019a)
September 2019	Online only	Revised for Version 4.4 (Release R2019b)
March 2020	Online only	Revised for Version 5.0 (Release R2020a)
September 2020	Online only	Revised for Version 5.1 (Release R2020b)
March 2021	Online only	Revised for Version 5.2 (Release R2021a)
September 2021	Online only	Revised for Version 5.3 (Release R2021b)
March 2022	Online only	Revised for Version 5.4 (Release R2022a)
September 2022	Online only	Revised for Version 5.5 (Release R2022b)
March 2023	Online only	Revised for Version 5.6 (Release R2023a)

Contents

1 Simulink Coverage Product Description

2 Getting Started with Simulink Coverage

Simulink Coverage Product Description

Simulink Coverage Product Description

Measure test coverage in models and generated code

Simulink® Coverage™ performs model and code coverage analysis that measures testing completeness in models and generated code. It applies industry-standard metrics such as decision, condition, modified condition/decision coverage (MCDC), and relational boundary coverage to assess the effectiveness of simulation testing in models, software-in-the-loop (SIL), and processor-in-the-loop (PIL). You can use missing coverage data to find gaps in testing, missing requirements, or unintended functionality.

Simulink Coverage produces interactive reports showing how much of your model, C/C++ S-functions, MATLAB® functions, and code generated by Embedded Coder® has been exercised. You can highlight coverage results in blocks and subsystems to visualize gaps in testing. To assess testing completeness, you can accumulate coverage data from multiple test runs, as well as view coverage achieved through unit and system tests. Coverage outcomes can be traced to requirements and tests. You can apply filters to exclude blocks from coverage and justify missing coverage in reports.

Support for industry standards is available through DO Qualification Kit and IEC Certification Kit.

Getting Started with Simulink Coverage

Structural Coverage Metrics

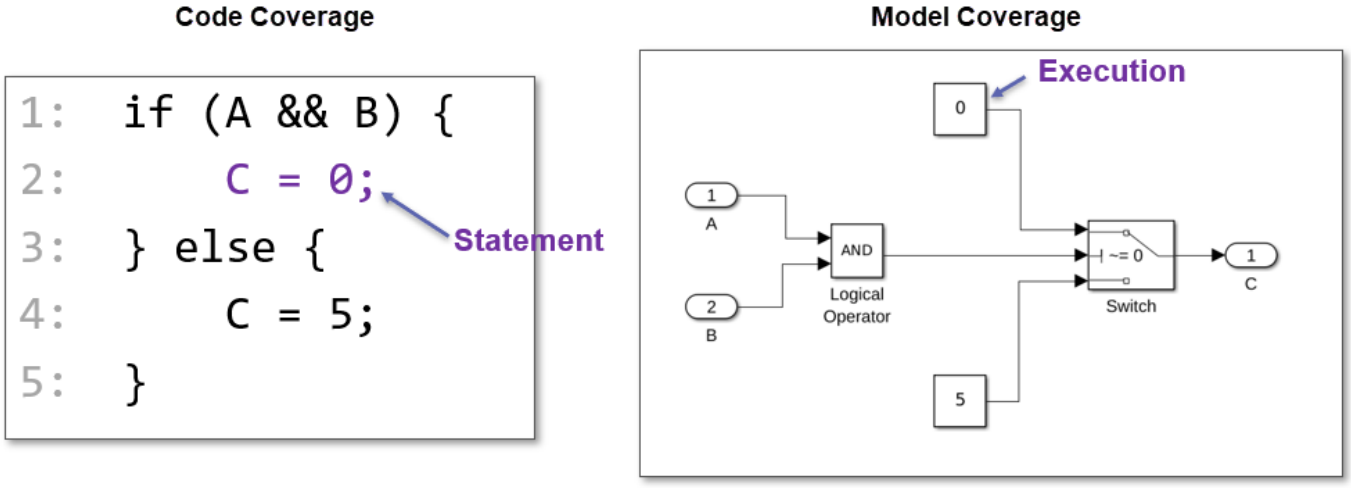
When Simulink Coverage measures coverage for a model, it uses the same principles used in code coverage. In code coverage, you create tests to show that every line of code executes to show that there is no unexpected behavior in your program. Simulink Coverage applies the same measurement to your Simulink model. Simulink Coverage analyzes your model and reports which blocks and logical branches of the model are active during the simulation.

Structural coverage metrics analyze the structural elements of your model, such as blocks and Stateflow® charts, and report how much of the model and, if applicable, logical branches execute. The four structural coverage metrics for model coverage are block execution coverage, decision coverage, condition coverage, and modified condition decision coverage (MCDC).

For more information about these and other coverage metrics, see “Types of Model Coverage”.

Block Execution Coverage

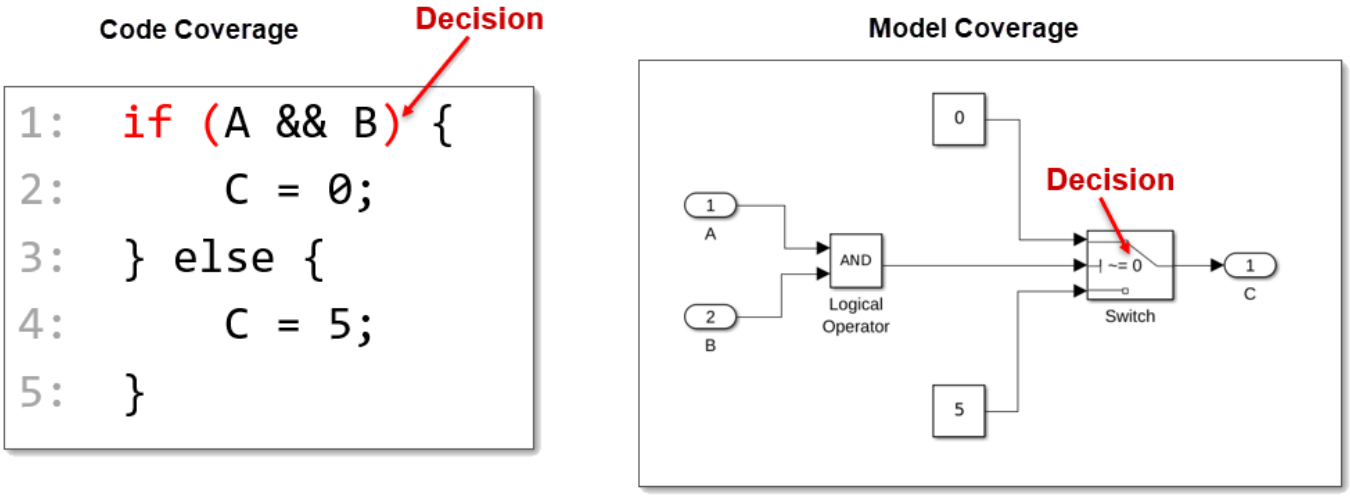
Block execution coverage tells you whether each block executes during simulation. The code coverage metric that block execution coverage is most similar to is statement coverage.



Most blocks receive execution coverage, although there are some types of blocks which do not receive any coverage metrics. For more information, see “Model Objects That Do Not Receive Coverage”.

Decision Coverage

Decision coverage analyzes decision points in your code or model. In code coverage, a decision is a Boolean expression composed of one or more conditions and zero or more Boolean operators. In model coverage, a decision is a place in your model where the value of one or more input signals decide the output signal of a block.

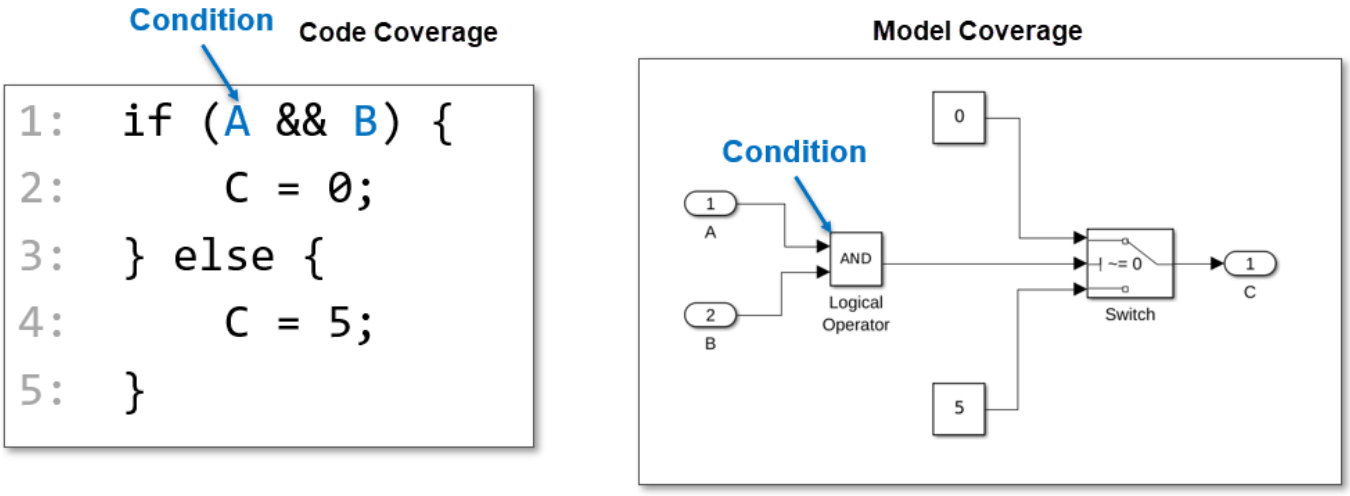


In code coverage, the decision `A && B` has 100% decision coverage if the decision is `true` for at least one time step, and false for at least one time step, regardless of the outcomes of the individual conditions inside the decision. Similarly, in model coverage, a Switch block has 100% decision coverage if the `true` case is executed for at least one time step, and the `false` case is executed for at least one time step.

Condition Decision Coverage

Condition Decision coverage analyzes both decision points and the conditions that make up the decisions. In code coverage, a condition is a Boolean expression that does not contain Boolean operators. In other words, it is a Boolean expression that cannot be broken down into simpler Boolean expressions. An example of a condition is an expression like `A`, where the outcome is `true` or `false`.

In model coverage, condition coverage analyzes blocks that output the logical combination of their input. An example of a condition in a model is the Logical Operator block. A Logical Operator block with the operator set to `And` would output `true` if all of its input signals are `true`. This is what is meant by the logical combination of its inputs.



The expression `A` has 100% coverage if the condition is `true` for at least one time step, and `false` for at least one time step. Similarly, the Logical Operator block has 100% condition coverage if each of its input signals is `true` for at least one time step, and `false` for at least one time step. In this example model, the signals `A` and `B` are Boolean values in the model which represent the same conditions as the code example.

Modified Condition Decision Coverage (MCDC)

MCDC is condition decision coverage, except each condition must independently affect the decision outcome.

In code coverage, MCDC analyzes code to test that every entry and exit point of a program is invoked at least once, and every condition and decision has taken every possible outcome at least once. In addition, each condition must independently affect the decision outcome.

Using the same example with the decision `A && B`, to achieve 100% MCDC coverage, the conditions must show `TT`, `TF`, and `FT`. You need these three outcomes in order to show that each condition independently affects the outcome of the block. The `FF` outcome is not needed because this does not change the decision outcome from the `TF` and `FT` cases.

In model coverage, MCDC looks for decision reversals that occur because one condition outcome changes from `true` to `false` or from `false` to `true`. An example of a block that receives MCDC coverage is the Logical Operator block.

An And block would have 100% MCDC coverage if each input condition outcome changes, independently affecting the outcome of the decision. For example, an And block with two input conditions must show three outcomes. The `true` case where both input conditions are true, and two false cases, where condition 1 is false while condition 2 is true, and vice-versa.

See Also

Related Examples

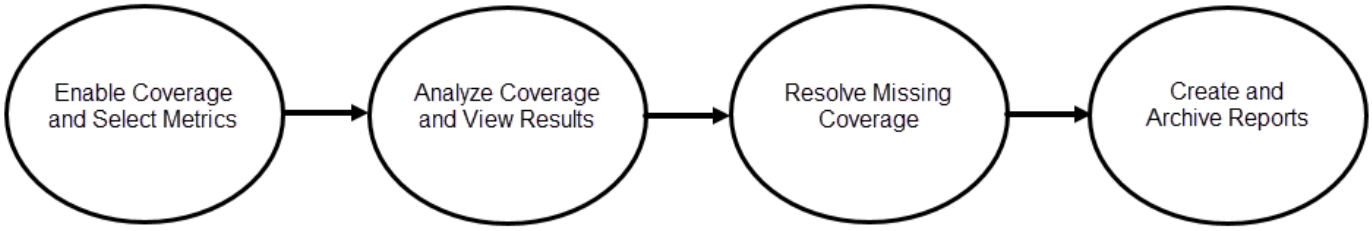
- “Use Simulink Coverage to Analyze Your Model” on page 2-6

- “Types of Model Coverage”
- “Model Objects That Receive Coverage”

Use Simulink Coverage to Analyze Your Model

Tip If you have a Simulink Test license, you can collect and aggregate coverage directly in the Test Manager. See “Analyze Model Coverage by Using the Test Manager in Simulink Test” on page 2-24 for more information.

Use Simulink Coverage to gain insight into which parts of your model are exercised during simulation to uncover bugs in your model or gaps in your testing. Your workflow usually looks like:



Enable Coverage and Select Metrics

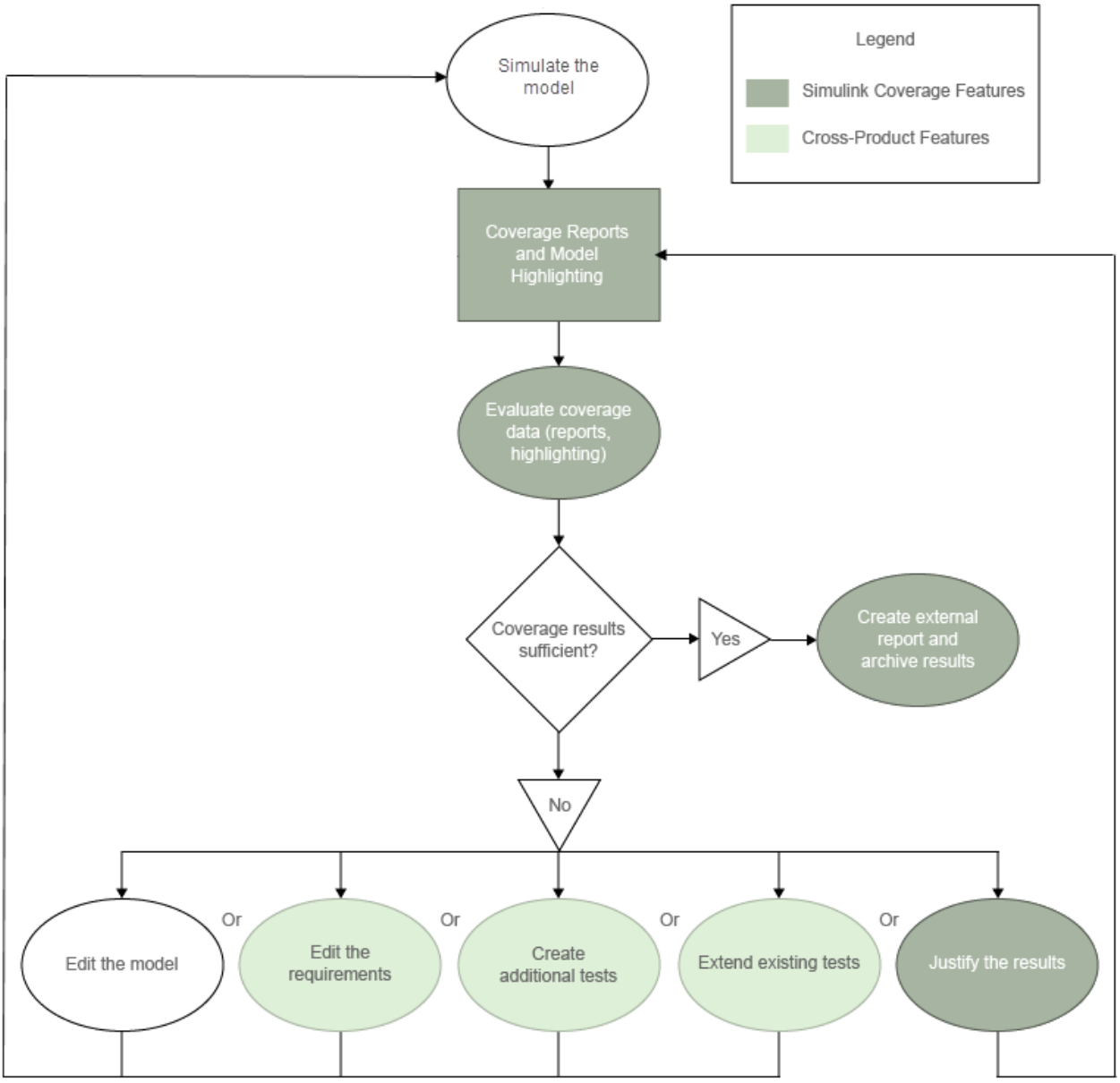
First, you need to enable coverage and select the coverage metrics that best apply to your model. For more information about selecting metrics, see “Enable Coverage and Select Metrics” on page 2-9. For detailed information about specific blocks, see “Model Objects That Receive Coverage”.

Analyze Coverage and View Results

When you simulate your model with coverage enabled, Simulink Coverage reports on, and highlights, the model elements that are exercised during simulation. Use the model coloring in Simulink and the **Coverage Details** pane to browse the coverage results. You can also use the Coverage Results Explorer to look at individual tests and aggregated coverage and see which tests exercise which model elements. Coverage aggregation, which is also called cumulative coverage, allows you to run multiple tests, each of which might target a specific part of your model, and then combine the coverage from each of your simulations into a single report.

Address Missing Coverage

After you view your model coverage results, you might find that your model does not have 100% coverage. You can increase your model coverage by:



- **Editing the model** — Your model might contain unintended functionality that is not part of your required design. Remove the unintended functionality.
- **Editing the requirements** — You might have an incorrect requirement or a missing requirement. You can use Requirements Toolbox™ to author and edit requirements in Simulink. If you add a requirement, you must add a test case for that requirement. For more information, see “Test Model Against Requirements and Report Results”.
- **Editing the tests** — Your existing tests might not fully exercise the intended simulation inputs. You can use Simulink Coverage to create additional tests to model these inputs. For more information, see “Automating Model Coverage Tasks” and “Generate Test Cases” (Simulink Design Verifier). If you have a Simulink Design Verifier™ license, you may be able to automatically

generate additional tests to exercise untested parts of your design. For more information, see “Incrementally Increase Test Coverage Using Test Case Generation”

- **Filtering missing coverage results** — Parts of your model might not be exercised during simulation according to your design, such as subsystems that only activate during failures. To achieve full coverage in such cases, you can filter out results for model elements you do not intend to exercise. For more information, see “Create, Edit, and View Coverage Filter Rules”.

Create and Archive Reports

After collecting coverage and addressing unsatisfied outcomes, you can create HTML reports for archiving results. In the Coverage Results Explorer, you can create a report from the aggregated coverage results or from individual test cases.

When you create a standalone coverage report, by default, the coverage tables list the satisfied and unsatisfied outcomes for only the model elements that have incomplete coverage. This is different than the report displayed in the **Coverage Details** pane, which shows the coverage tables by default. You can change the report settings in the Configuration Parameters window or in the Coverage Results Explorer. See “Access, Manage, and Aggregate Coverage Results” for more information.

To get started with this tutorial, see

- STEP 1: “Enable Coverage and Select Metrics” on page 2-9
- STEP 2: “Analyze Coverage and View Results” on page 2-11
- STEP 3: “Address Missing Coverage” on page 2-16
- STEP 4: “Create and Archive Reports” on page 2-23

See Also

More About

- “Types of Model Coverage”
- “Types of Code Coverage”
- “View Coverage Results in Simulink Canvas”

Enable Coverage and Select Metrics

Before you analyze coverage results for your model, you need to enable coverage and decide which coverage metrics you want to see.

Open the Model

Open the `slvnvdemo_powerwindow` model. The `slvnvdemo_powerwindow` model contains a power window controller and a low-order plant model. The controller is implemented with a Stateflow® chart.

Choose Your Coverage Metrics

On the **Apps** tab, select **Coverage Analyzer**. Then, on the **Coverage** tab, click **Settings** to open the Configuration Parameters window. In the left pane, click **Coverage**. Select **Enable coverage analysis**. By default, the scope of the coverage analysis is set to **Entire system**. If you want to collect coverage data for a specific referenced model or subsystem, select your desired coverage scope and click **Select Models** or **Select Subsystem**. For this example, select **Entire system**.

The **Structural coverage level** parameter has four settings:

- `Block Execution`

  Whether each block is executed during simulation.
- `Decision`

  Analyzes decision points in your model. A decision is a place in your model where one or more input signals decide the output signal of a block. Selecting `Decision` coverage also collects `Block Execution` coverage.
- `Condition Decision`

  Analyzes blocks that output the logical combination of their input. A condition is a Boolean expression that does not contain Boolean operators. Selecting `Condition Decision` coverage also collects `Block Execution` coverage.
- `Modified Condition Decision Coverage (MCDC)`

  MCDC is condition decision coverage, except each condition must independently affect the decision outcome. Selecting `Modified Condition Decision Coverage (MCDC)` also collects `Condition Decision` and `Block Execution` coverage.

For this example, set **Structural coverage level** to `Modified Condition Decision Coverage (MCDC)`.

For a full list of coverage metrics, including advanced metrics, see “Types of Model Coverage”.

Command-Line Information

You can also perform the steps in this example programmatically. Use `sim` with a `Simulink.SimulationInput` object to set the coverage model parameters for a simulation.

```
openExample('slcoverage/GetStartedWithSimulinkCoverageExample')
modelName = 'slvnvdemo_powerwindow';
open_system(modelName)
simIn = Simulink.SimulationInput(modelName);
simIn = setModelParameter(simIn,'CovEnable','on');
simIn = setModelParameter(simIn,'CovScope','EntireSystem');
simIn = setModelParameter(simIn,'CovMetricStructuralLevel','MCDC');
simIn = setModelParameter(simIn,'CovSaveSingleToWorkspaceVar','on');
simIn = setModelParameter(simIn,'CovSaveName','covData');
```

See Also

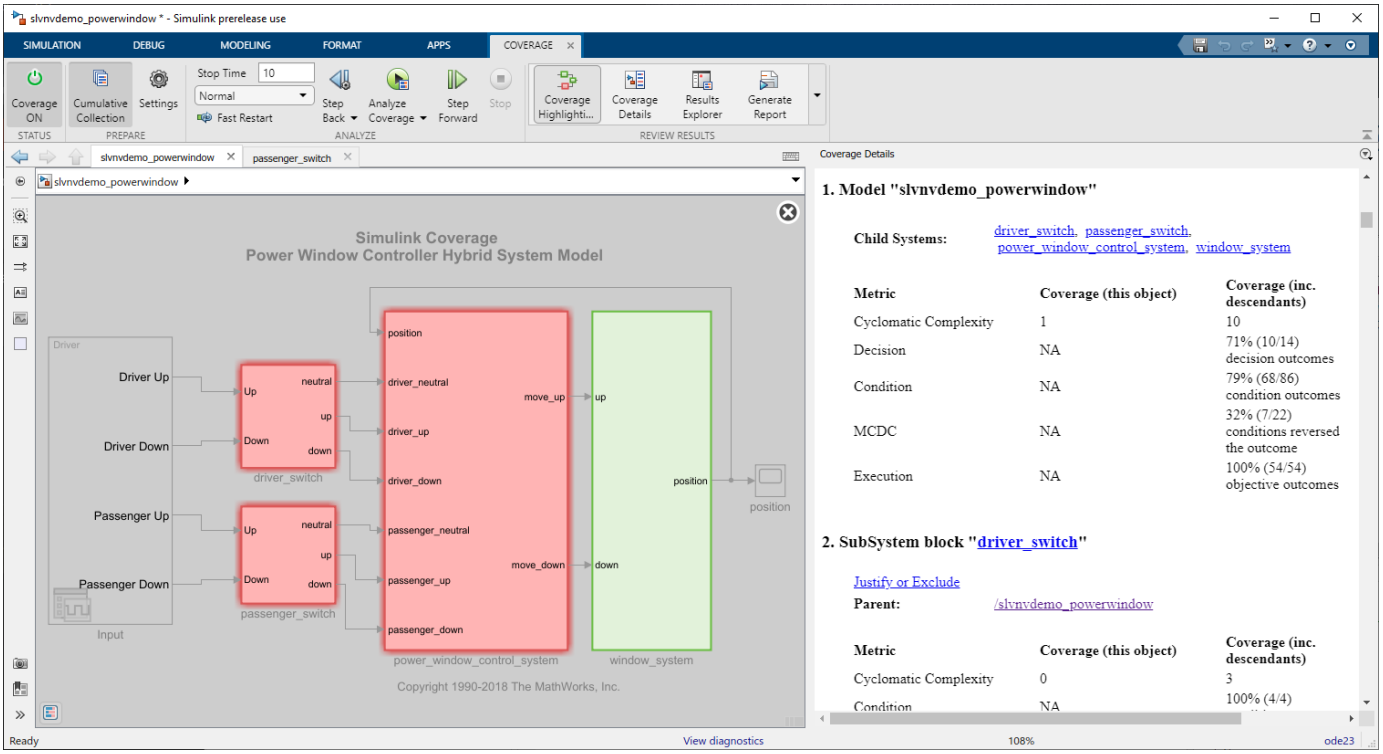
More About

- “Types of Model Coverage”
- “Specify Coverage Options”
- “Coverage Settings”
- “Model Objects That Receive Coverage”

Analyze Coverage and View Results

After you have enabled coverage and selected the metrics you want, you must simulate the model to collect the coverage results.

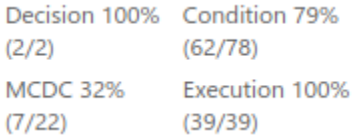
First, simulate the model by clicking the **Analyze Coverage** button. When the simulation completes, Simulink Coverage highlights the model objects based on their coverage completeness.



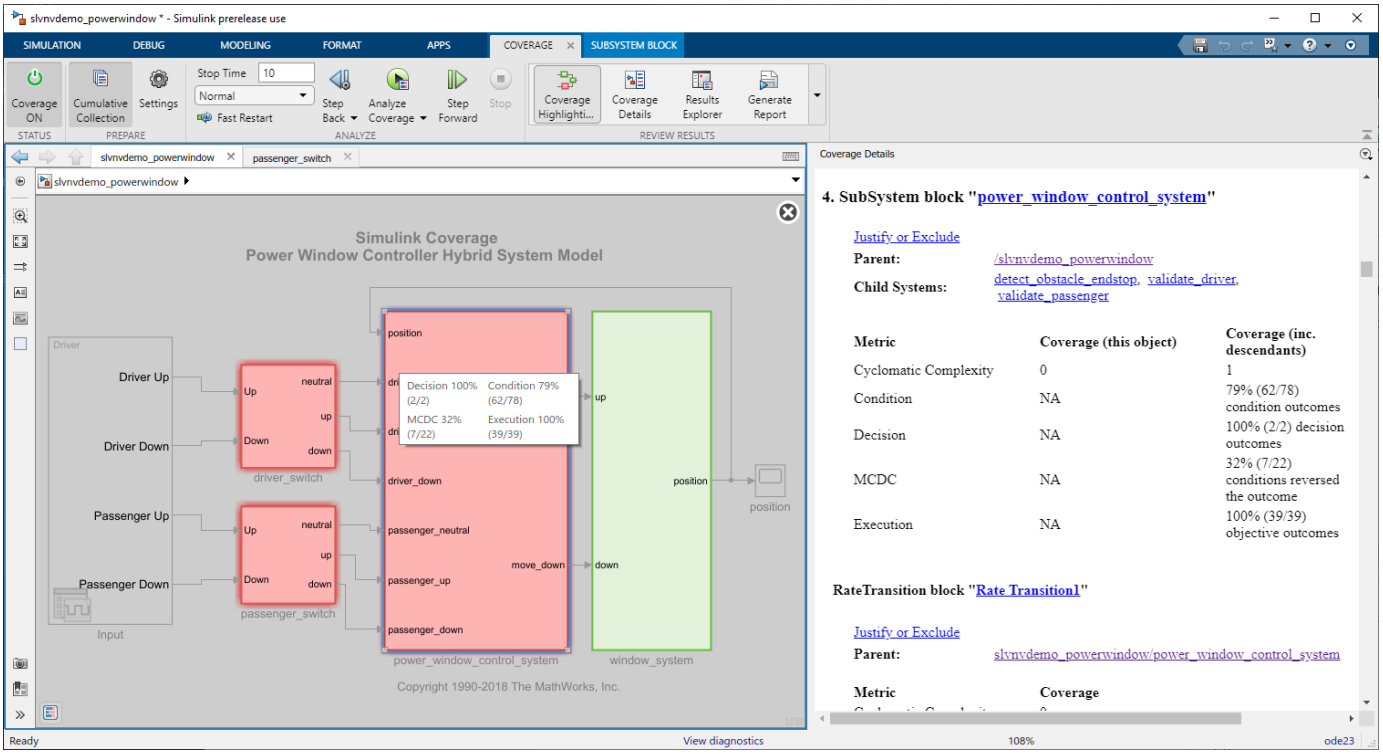
The **Coverage Details** pane opens automatically and displays the coverage report. The report automatically opens in the **Details** section which lists the coverage results for the model objects. You can scroll up to view the summary section and model information, or scroll down to browse the individual blocks.

The `window_system` subsystem is colored green because every block inside the subsystem received 100% coverage. The other subsystems are colored red because one or more blocks inside did not receive full coverage.

Point to a subsystem to display a short summary of coverage for each relevant metric. For example, point to the `power_window_control_system` subsystem block.

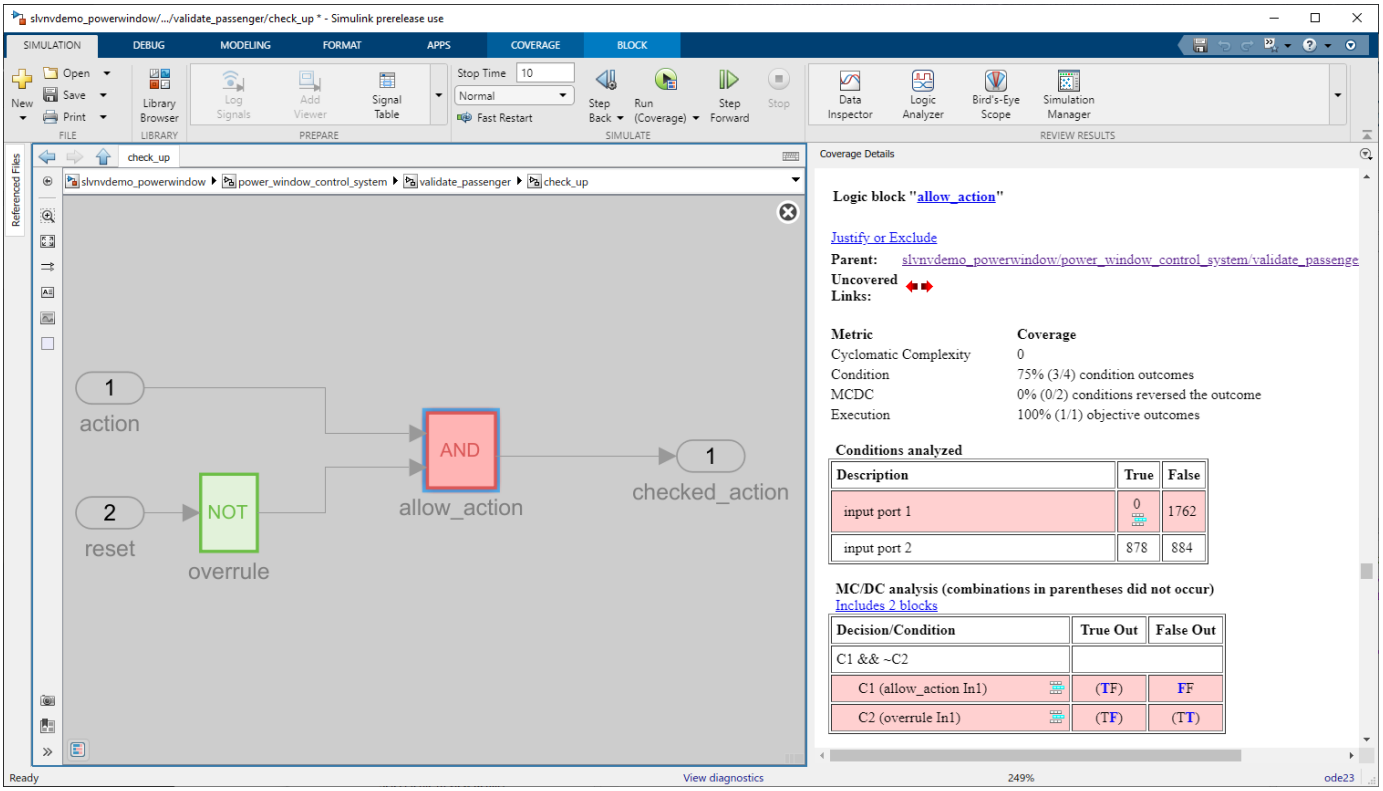


If you left click on a block, the **Coverage Details** pane displays the section of the report for that block. For example, click the `power_window_control_system` subsystem.



You can enter a subsystem to view the granular coverage results for the contents of that subsystem. Double-click the `power_window_control_system` subsystem to view the coverage results for the blocks it contains. Double-click on `validate_passenger` and then on `check_up`.

Click on the And block `allow_action` to display the relevant section of the report.



Because the And block outputs the logical combination of the two input signals, if both input signals are true, the output signal is true. You can see from the condition table that input signal 1 was `False` at every time step, while input signal 2 was `True` for 878 time steps, and `False` for 884 time steps.

Logic block "allow_action"

Justify or Exclude

Parent: slvnvdemo_powerwindow/power_window_control_system/validate_passenger/check_up

Uncovered Links:

Metric	Coverage
Cyclomatic Complexity	0
Condition	75% (3/4) condition outcomes
MCDC	0% (0/2) conditions reversed the outcome
Execution	100% (1/1) objective outcomes

Conditions analyzed

Description	True	False
input port 1	0	1762
input port 2	878	884

Because there are four condition outcomes and one of them did not occur, the `allow_action` block received 75% condition coverage.

Below the **Conditions analyzed** table is the **MC/DC analysis** table.

MC/DC analysis (combinations in parentheses did not occur)

Includes 2 blocks

Decision/Condition	True Out	False Out
C1 && ~C2		
C1 (allow_action In1)	(TF)	FF
C2 (overrule In1)	(TF)	(TT)

The MCDC analysis table lists decision reversals that occur because one condition outcome changes from `true` to `false` or from `false` to `true`. This table reports that the only the `FF` case occurred during the simulation for first condition, and neither MCDC objective occurred for the second condition, resulting in 0% MCDC reported.

Command-Line Information

To perform the same steps from the command line, enter:

```
simOut = sim(simIn);
covData = simOut.covData;
cvmodelview(covData);
```

See Also

Related Examples

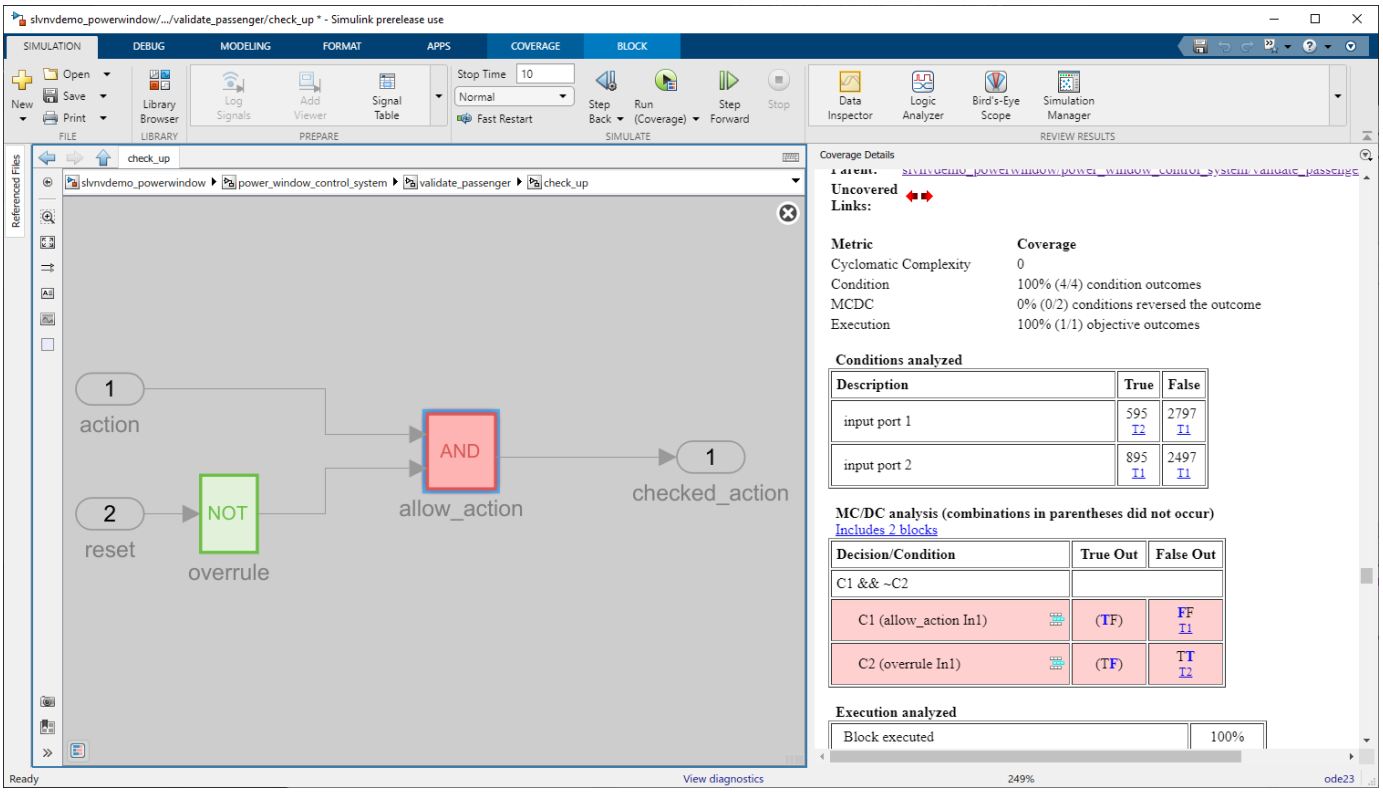
- “Model Objects That Receive Coverage”
- “View Coverage Results in Simulink Canvas”
- “Accessing Coverage Data from the Results Explorer”

Address Missing Coverage

After you view your model coverage results, you might find that your model does not have 100% coverage. You can address missing coverage in your model.

Add a Test Case to Improve Coverage

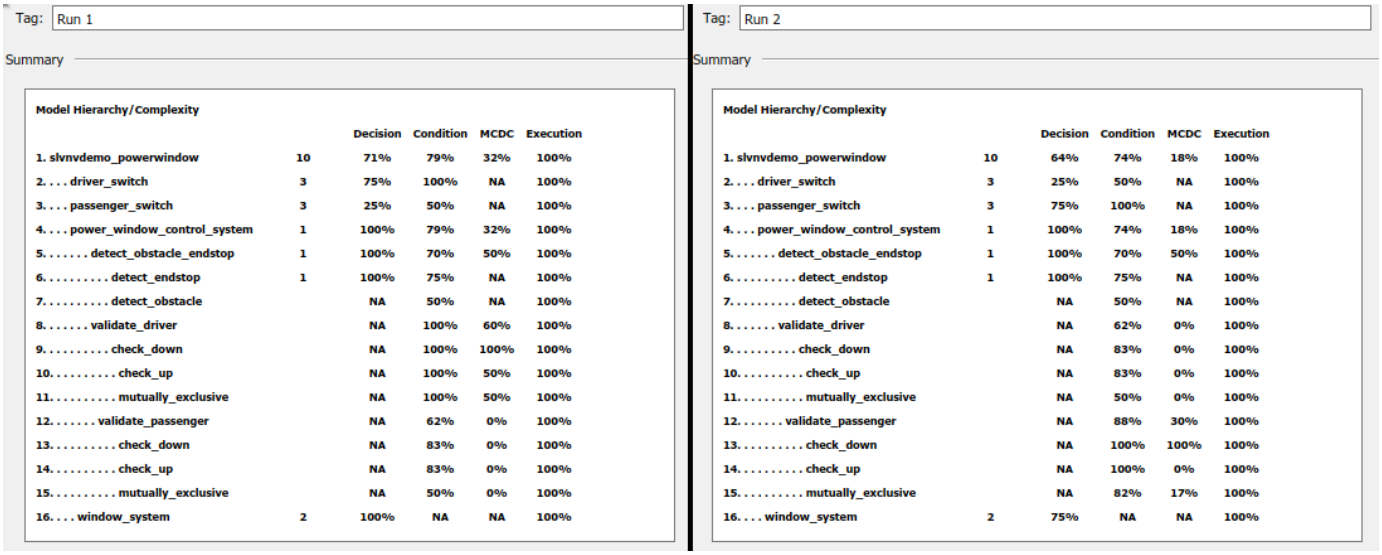
In the `slvnvdemo_powerwindow` model, the And block `power_window_control_system/validate_passenger/check_up/allow_action` has 75% condition coverage. The `true` case of the first condition did not occur because, in the Signal Editor block `Input` at the model root level, the **Active scenario** parameter is set to `Driver`. Change **Active scenario** to `Passenger`. Simulate the model again by clicking **Analyze Coverage**.



The And block `allow_action` now receives 100% condition coverage. The first condition was true for 595 time steps, and false for 2797 time steps. Additionally, the **T1** and **T2** links in the condition and MCDC tables link to the tests that were used to test each objective. For example, the `true` case of the first condition was satisfied by test run 2, **T2**. Click the link to scroll to the **Aggregated Tests** section of the report.

Review Coverage Data in the Coverage Results Explorer

You can also review coverage data by using the Coverage Results Explorer. On the **Coverage** tab, click **Results Explorer**. The run data is in the left pane, under **Current Cumulative Data**. Click **Run 1** and **Run 2** to compare the coverage results.



Tag: Run 1

Summary

Model Hierarchy/Complexity

		Decision	Condition	MCDC	Execution
1. slvnvdemo_powerwindow	10	71%	79%	32%	100%
2. . . . driver_switch	3	75%	100%	NA	100%
3. . . . passenger_switch	3	25%	50%	NA	100%
4. . . . power_window_control_system	1	100%	79%	32%	100%
5. . . . . detect_obstacle_endstop	1	100%	70%	50%	100%
6. . . . . detect_endstop	1	100%	75%	NA	100%
7. . . . . detect_obstacle		NA	50%	NA	100%
8. . . . . validate_driver		NA	100%	60%	100%
9. . . . . check_down		NA	100%	100%	100%
10. . . . . check_up		NA	100%	50%	100%
11. . . . . mutually_exclusive		NA	100%	50%	100%
12. . . . . validate_passenger		NA	62%	0%	100%
13. . . . . check_down		NA	83%	0%	100%
14. . . . . check_up		NA	83%	0%	100%
15. . . . . mutually_exclusive		NA	50%	0%	100%
16. . . . window_system	2	100%	NA	NA	100%

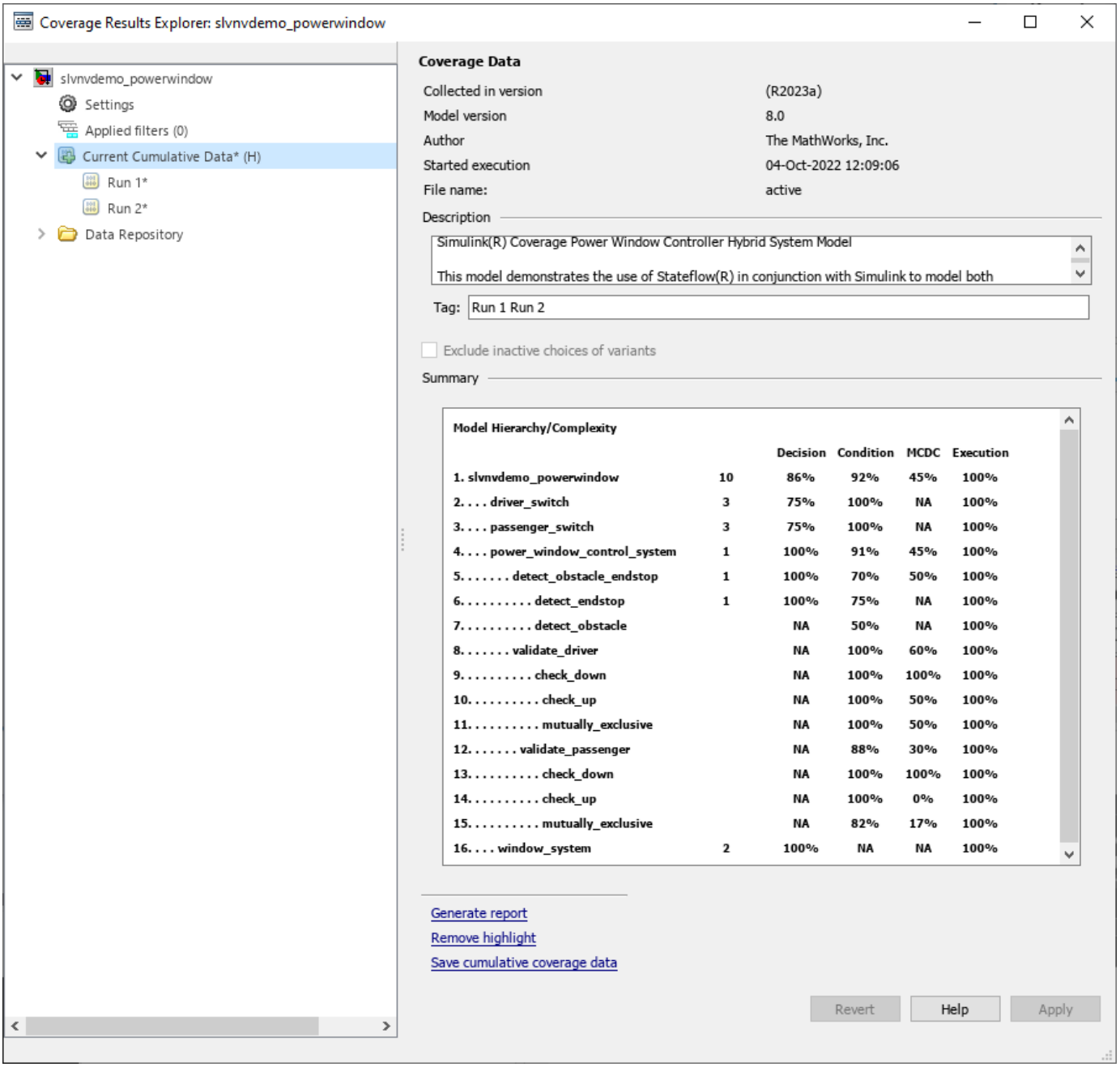
Tag: Run 2

Summary

Model Hierarchy/Complexity

		Decision	Condition	MCDC	Execution
1. slvnvdemo_powerwindow	10	64%	74%	18%	100%
2. . . . driver_switch	3	25%	50%	NA	100%
3. . . . passenger_switch	3	75%	100%	NA	100%
4. . . . power_window_control_system	1	100%	74%	18%	100%
5. . . . . detect_obstacle_endstop	1	100%	70%	50%	100%
6. . . . . detect_endstop	1	100%	75%	NA	100%
7. . . . . detect_obstacle		NA	50%	NA	100%
8. . . . . validate_driver		NA	62%	0%	100%
9. . . . . check_down		NA	83%	0%	100%
10. . . . . check_up		NA	83%	0%	100%
11. . . . . mutually_exclusive		NA	50%	0%	100%
12. . . . . validate_passenger		NA	88%	30%	100%
13. . . . . check_down		NA	100%	100%	100%
14. . . . . check_up		NA	100%	0%	100%
15. . . . . mutually_exclusive		NA	82%	17%	100%
16. . . . window_system	2	75%	NA	NA	100%

Click **Current Cumulative Data** to see the aggregated results of these two runs.



The aggregated results show higher coverage percentages than either of the two individual test cases because the test cases satisfy different objectives in some blocks.

Filter Coverage Outcomes

If you analyze the coverage report and find that you are missing coverage that is not possible to fix by changing the model or a test case, you can filter the missing outcomes so that they are not reported as missing coverage. Some possible reasons you might want to filter coverage outcomes include:

- A block is tested by a different test suite, and is not applicable to the current coverage analysis.
- A block is intended to catch edge cases that you think should not occur anyway. This type of model design is sometimes called defensive coding.

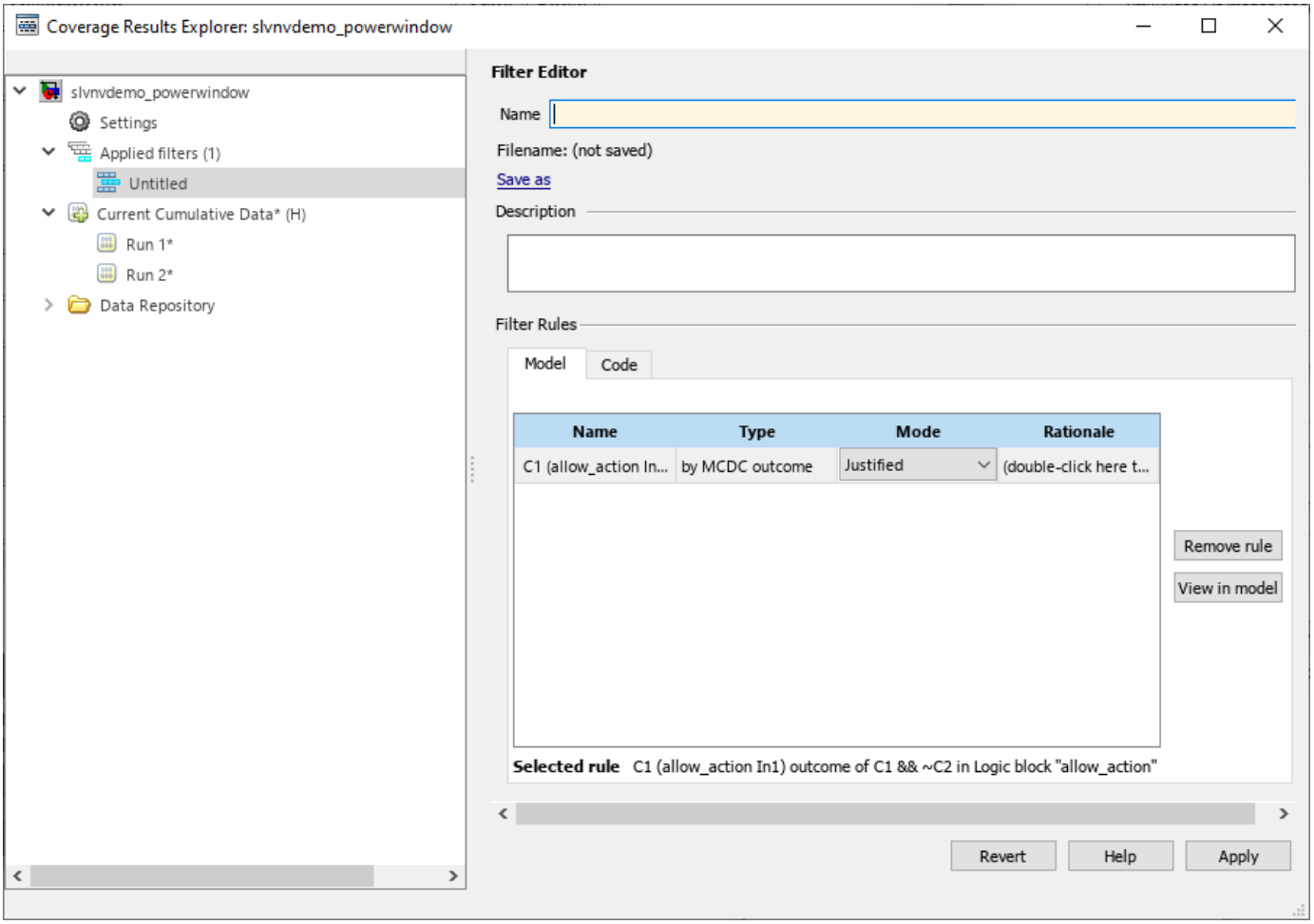
There are two types of coverage filters:

- An exclusion filter rule can be applied to a model element, and causes that element to be ignored by coverage analysis. The excluded model elements appear dimmed in the highlighted model, like other elements that are not applicable to the metrics you selected.
- A justification filter rule can be applied to a coverage outcome that is not satisfied. This filter rule allows Simulink Coverage to analyze the rest of the model element but does not report the justified outcome as missing coverage. This filter rule allows you to improve your coverage for a model object without excluding it entirely.

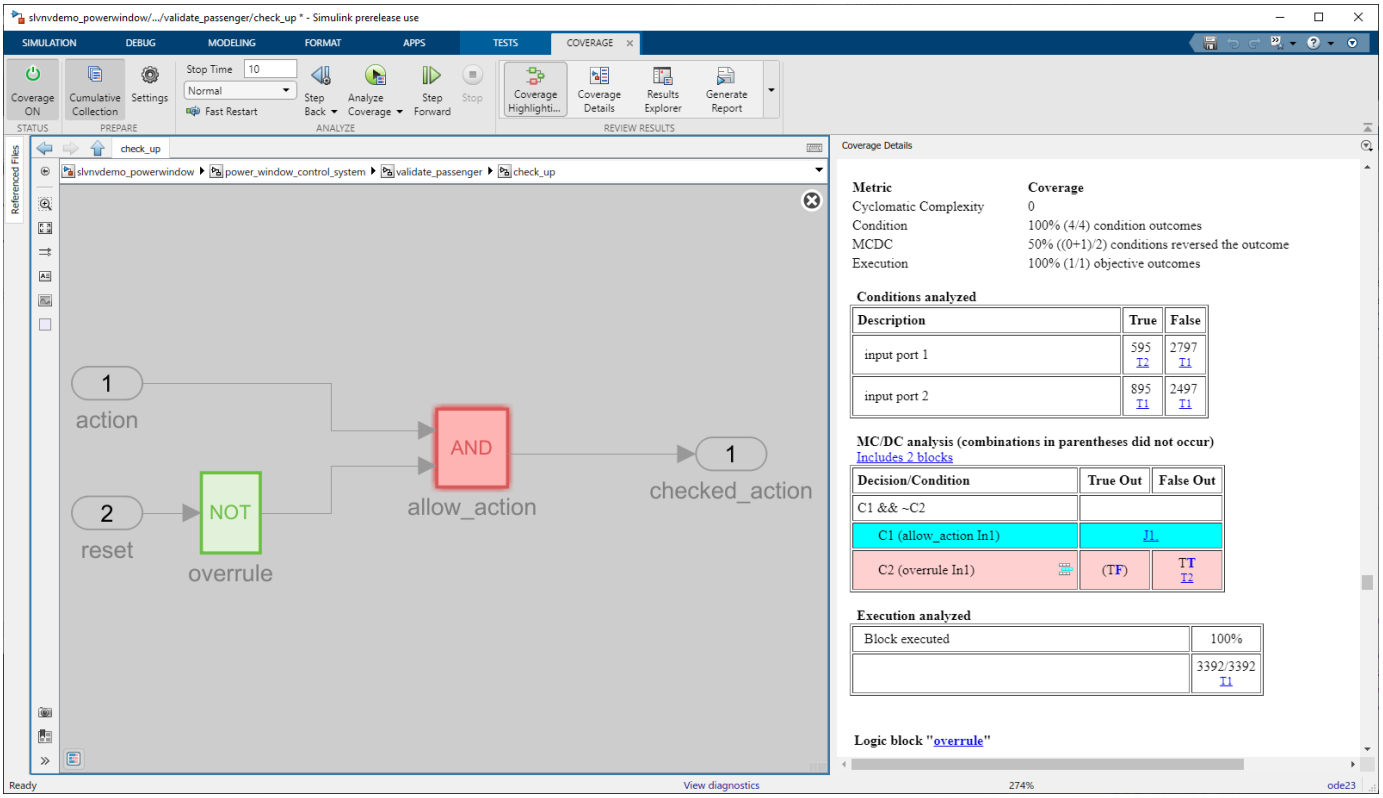
Suppose that the And block condition 1 MCDC outcome has been tested by a different test suite and is not applicable for this case. You can justify the outcome so that it is not reported as missing coverage.

Click the And block `allow_action` to scroll to the relevant section in the coverage report. The MCDC condition `C1 (allow_action In1)` is incomplete because the `TF` case did not occur. To justify the `C1 (allow_action In1)` MCDC outcome, click the Add justification rule icon  .

The Coverage Results Explorer opens the **Filter Editor** pane with a new untitled filter file. The filter file contains a justification rule for the specified outcome. You can add multiple filter rules to the same filter file.



In the **Name** field, enter `slvnvdemo_powerwindow_filter`. Under **Filter Rules**, double-click the **Rationale** field and enter `Tested in a different test suite`. Click **Apply**, then save the file. The model and coverage report automatically update to indicate that the outcome is justified.



In the **Coverage Details** pane, the justified outcome is highlighted in cyan and links to the justification rationale. Clicking **J1** brings you to the section of the report titled **Objects Filtered from Coverage Analysis**. This section of the report only appears if you apply one or more filters to the coverage data.

Command-Line Information

To programmatically add a test and aggregate coverage, enter:

```
blockPath = [modelName,'/Input'];
set_param(blockPath,'ActiveScenario','Passenger')
simOut2 = sim(simIn);
covDataRun2 = simOut2.covData;
cvmodelview(covDataRun2);
aggregatedCovData = covData + covDataRun2;
```

To programmatically filter coverage outcomes, enter:

```
filt = slcoverage.Filter;
setFilterName(filt,'slvnvdemo_powerwindow_filter');
blockPath = [modelName,'/power_window_control_system/validate_passenger/check_up/allow_action'];
sel = slcoverage.MetricSelector(slcoverage.MetricSelectorType.MCDCOutcome,blockPath,1,1);
rule = slcoverage.FilterRule(sel,'Tested in a different test suite');
addRule(filt,rule);
save(filt,'slvnvdemo_powerwindow_filter')
aggregatedCovData.filter = 'slvnvdemo_powerwindow_filter';
```

See Also

Related Examples

- “Access, Manage, and Aggregate Coverage Results”
- “Trace Coverage Results to Associated Test Cases”
- “Coverage Filter Rules and Files”
- “Create, Edit, and View Coverage Filter Rules”
- “Creating and Using Coverage Filters”

Create and Archive Reports

After collecting coverage and addressing unsatisfied outcomes, you can archive the results by creating HTML reports. In the Coverage Results Explorer, you can create a report from the aggregated coverage results, or from individual test cases.

When you create a standalone coverage report, by default, the coverage tables that list satisfied and unsatisfied outcomes appear for all model objects that receive coverage analysis. This is consistent with the report displayed in the **Coverage Details** pane. You can change this and other settings in the Configuration Parameters dialog box or in the Coverage Results Explorer. See “Access, Manage, and Aggregate Coverage Results” for more information.

To create a report for the aggregated results, in the Coverage Results Explorer click **Current Cumulative Data** and, at the bottom of the window, click **Generate Report**.

To create a report for an individual run, click on the run name and click **Generate Report**.

Command-Line Information

To generate an aggregated results report programmatically, enter:

```
cvhtml('aggregated_coverage_report',aggregatedCovData)
```

See Also

Related Examples

- “Access, Manage, and Aggregate Coverage Results”
- “Types of Coverage Reports”
- “Top-Level Model Coverage Report”

Analyze Model Coverage by Using the Test Manager in Simulink Test

If you have a Simulink Test™ license, you can collect coverage directly from the Test Manager. For more information about Simulink Test, see “Get Started with Simulink Test” (Simulink Test).

When you analyze coverage for a model, Simulink Coverage reports on and highlights the model objects in your design that were exercised during simulation. Use the Test Manager to explore tests and aggregated coverage to see which tests exercise certain model objects. You can run multiple tests in a test suite, analyze coverage for each test, then view the aggregated results for the result set in the Test Manager. Use the coverage reports and model highlighting to learn which parts of your model activated during the specified test cases.

Open the Model

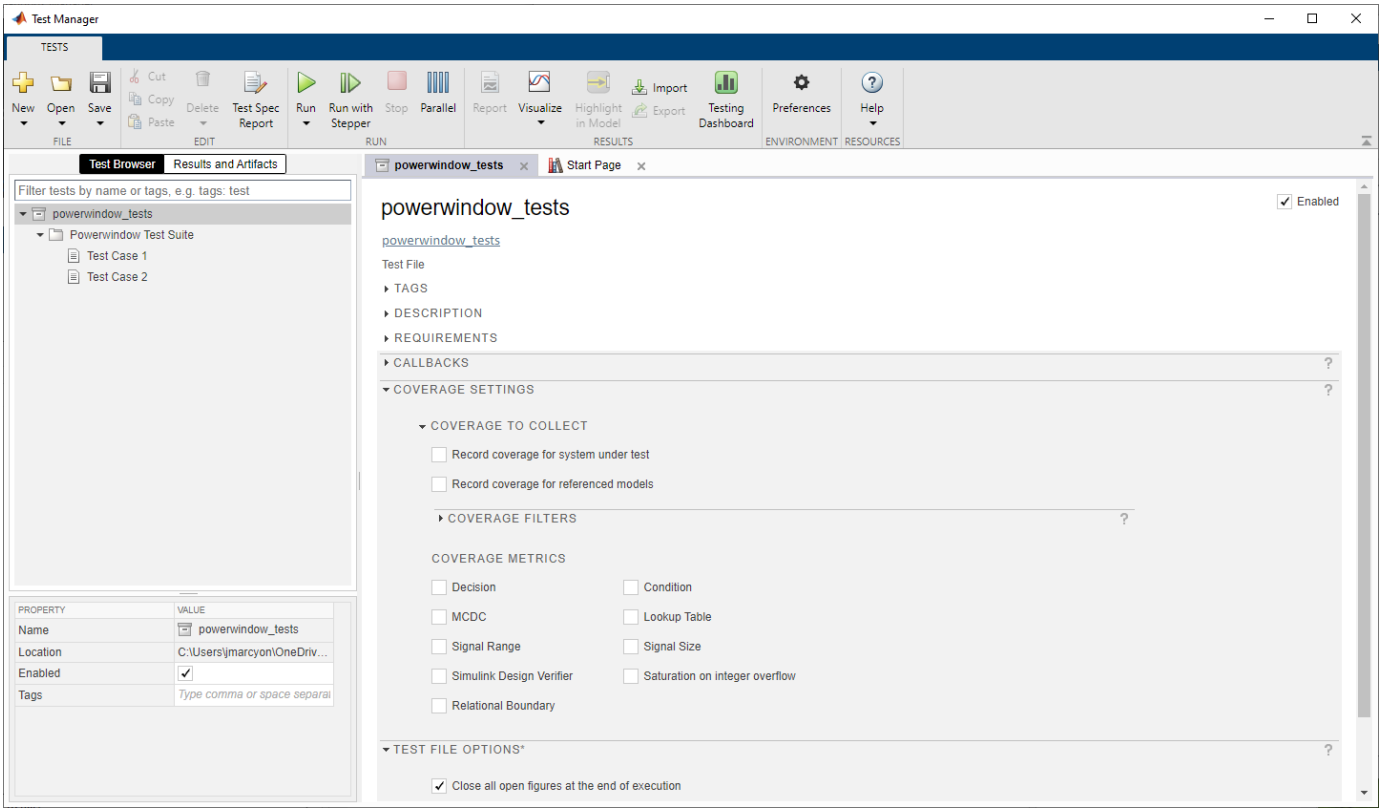
Open the `slvnvdemo_powerwindow` model. The `slvnvdemo_powerwindow` model contains a power window controller and a low-order plant model. The controller is implemented with a Stateflow® chart.

Choose Your Coverage Metrics

On the **Apps** tab, click **Simulink Test** to open the **Tests** tab. Click **Simulink Test Manager**.

This example includes a test file. In the Test Manager, click **Open**. In the Open File window, select `powerwindow_tests.mldatx`.

In the **Test Browser** pane, click `powerwindow_tests`. Then expand the coverage options by clicking **Coverage Settings**.



To enable coverage for the model, under **Coverage to Collect**, select **Record coverage for system under test**. Under **Coverage Metrics**, you can select the metrics you want to report. For information about the coverage metrics, see “Structural Coverage Metrics” on page 2-2.

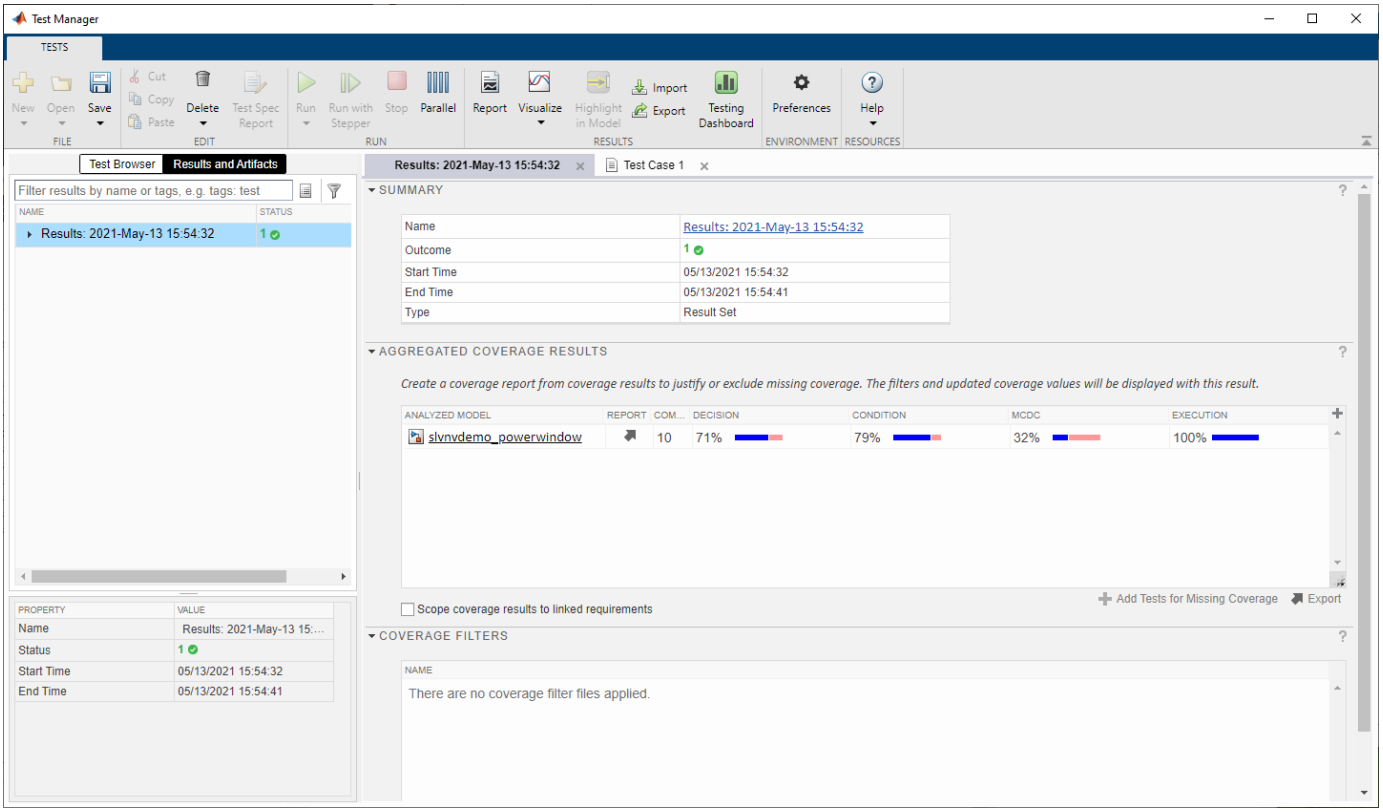
For this example, select **Decision**, **Condition**, and **MCDC**.

Run A Test and Collect Coverage

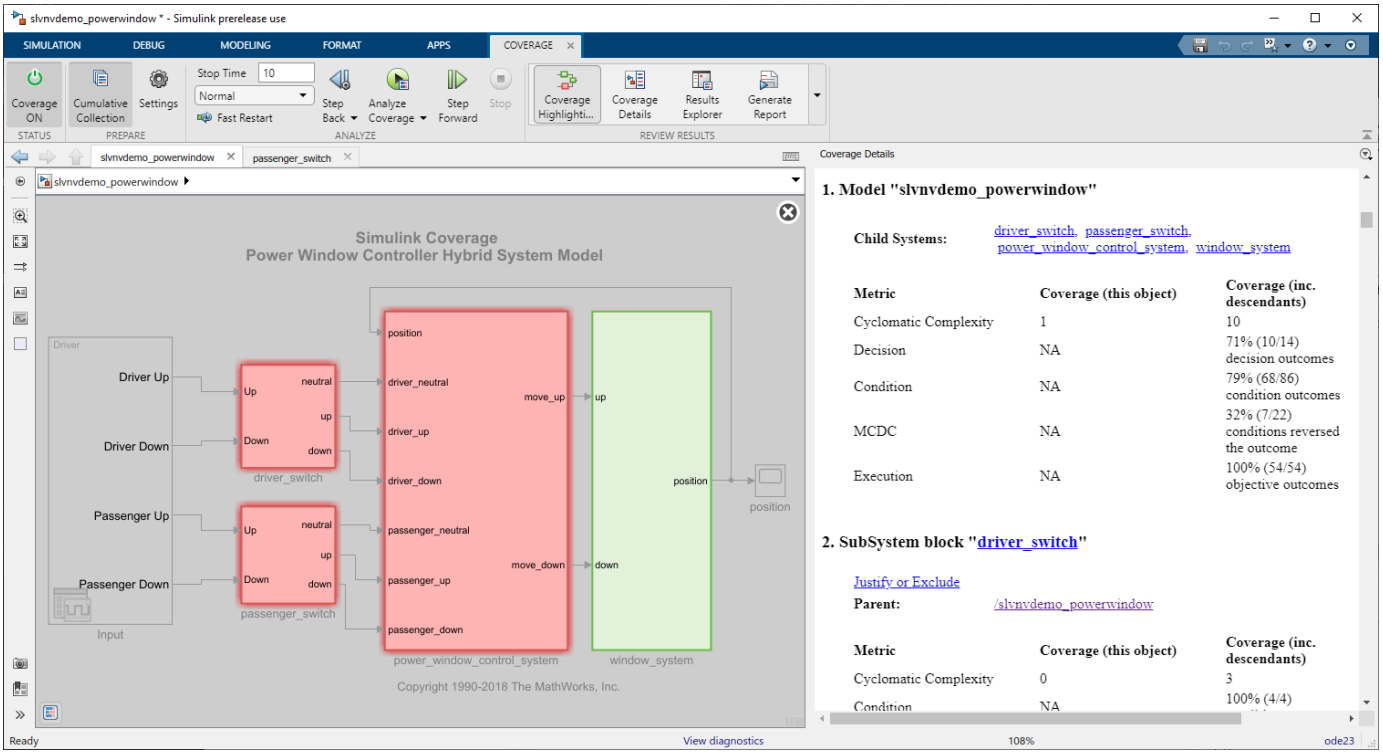
There are two tests in the `powerwindow_tests` test file. **Test Case 1** has the Signal Editor scenario set to `Driver`, and is the default setting in the model. **Test Case 2** has the Signal Editor scenario set to `Passenger`.

In the Test Manager, you can run individual tests or entire test suites by selecting them in the **Test Browser** and clicking **Run**. Select **Test Case 1** and click **Run**.

When the test finishes, the **Results and Artifacts** tab opens. Click the results for the test to see a summary of the coverage results.



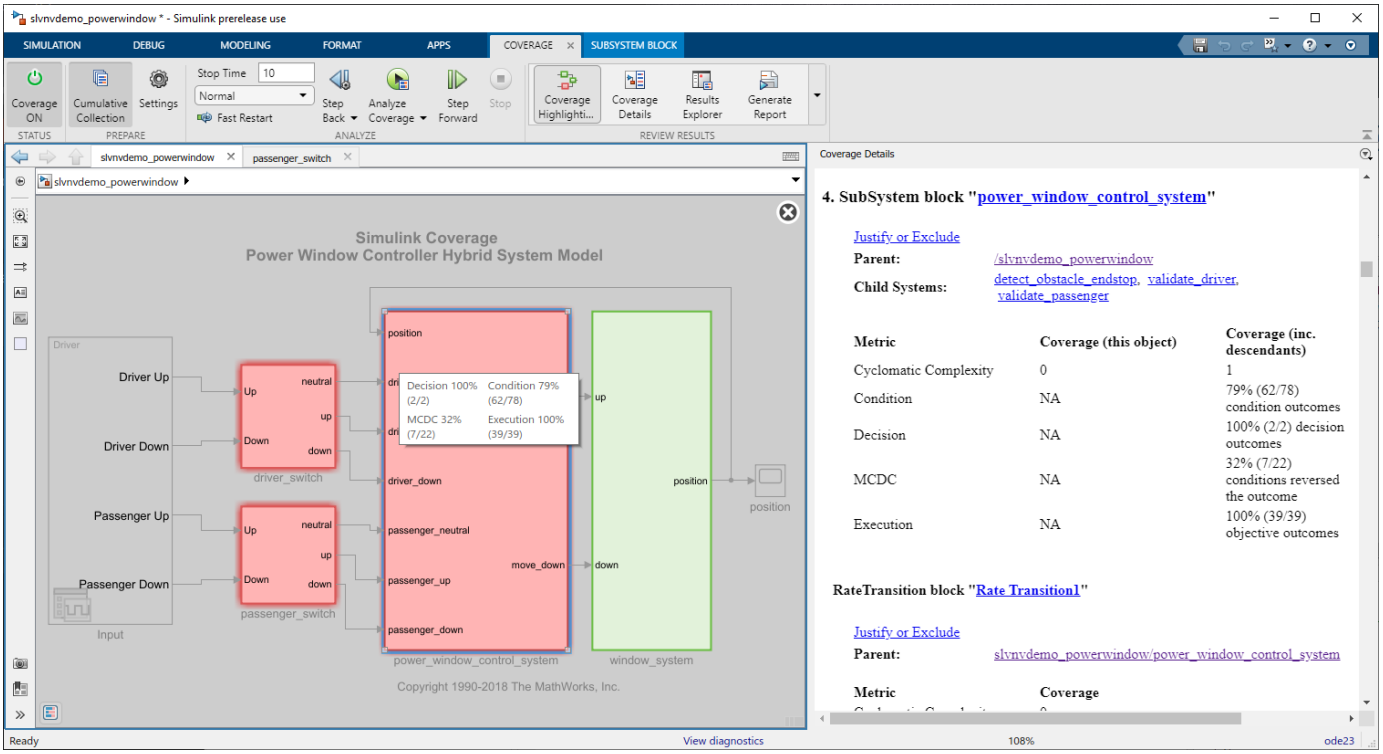
Test Case 1 is the case where the Signal Editor scenario is set to `Driver`. You can click the name of the model in the **Analyzed Model** column of the **Coverage Results** table to display the coverage results in the model with model highlighting enabled. The **Coverage Details** pane also opens in the Simulink window.



The `window_system` subsystem is green because every block inside the subsystem received 100% coverage. The other subsystems are red because one or more blocks inside did not receive full coverage.

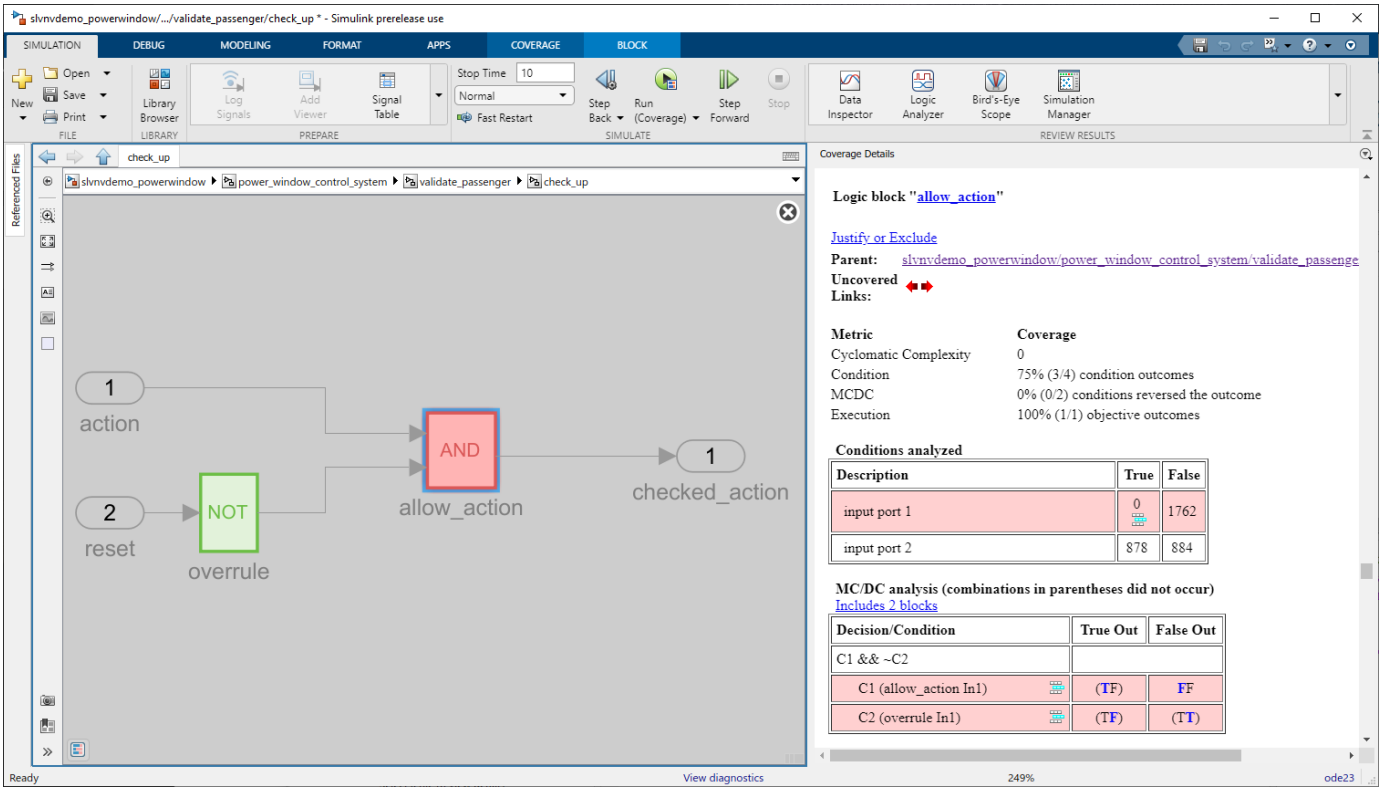
You can point to a subsystem to see a short summary of coverage for each relevant metric. For example, point to the `power_window_control_system` subsystem block.

If you left click on a block, the **Coverage Details** pane displays the section of the report for that block. For example, click on the `power_window_control_system` subsystem.



You can enter a subsystem to view the granular coverage results for the contents of that subsystem. Double-click the `power_window_control_system` subsystem and view the coverage results for the blocks it contains. Double-click on `validate_passenger`, then `check_up`.

Click the And block `allow_action` to display the relevant section of the report.



Click the And block to see the condition and MCDC results. Because the And block outputs the logical combination of the two input signals, if both input signals are true, the output signal is true. You can see from the condition table that input signal 1 was `false` at every time step, while input signal 2 was `true` for 878 time steps and `false` for 884 time steps. Because there are four condition outcomes and one of them did not occur, the `allow_action` block received 75% condition coverage.

The MCDC analysis table lists decision reversals that occur because one condition outcome changes from `true` to `false` or from `false` to `true`. This table reports that only the `FF` case occurred for the first condition, and neither MCDC objective occurred for the second condition, which results in 0% MCDC reported.

Run Multiple Tests and View Aggregated Coverage

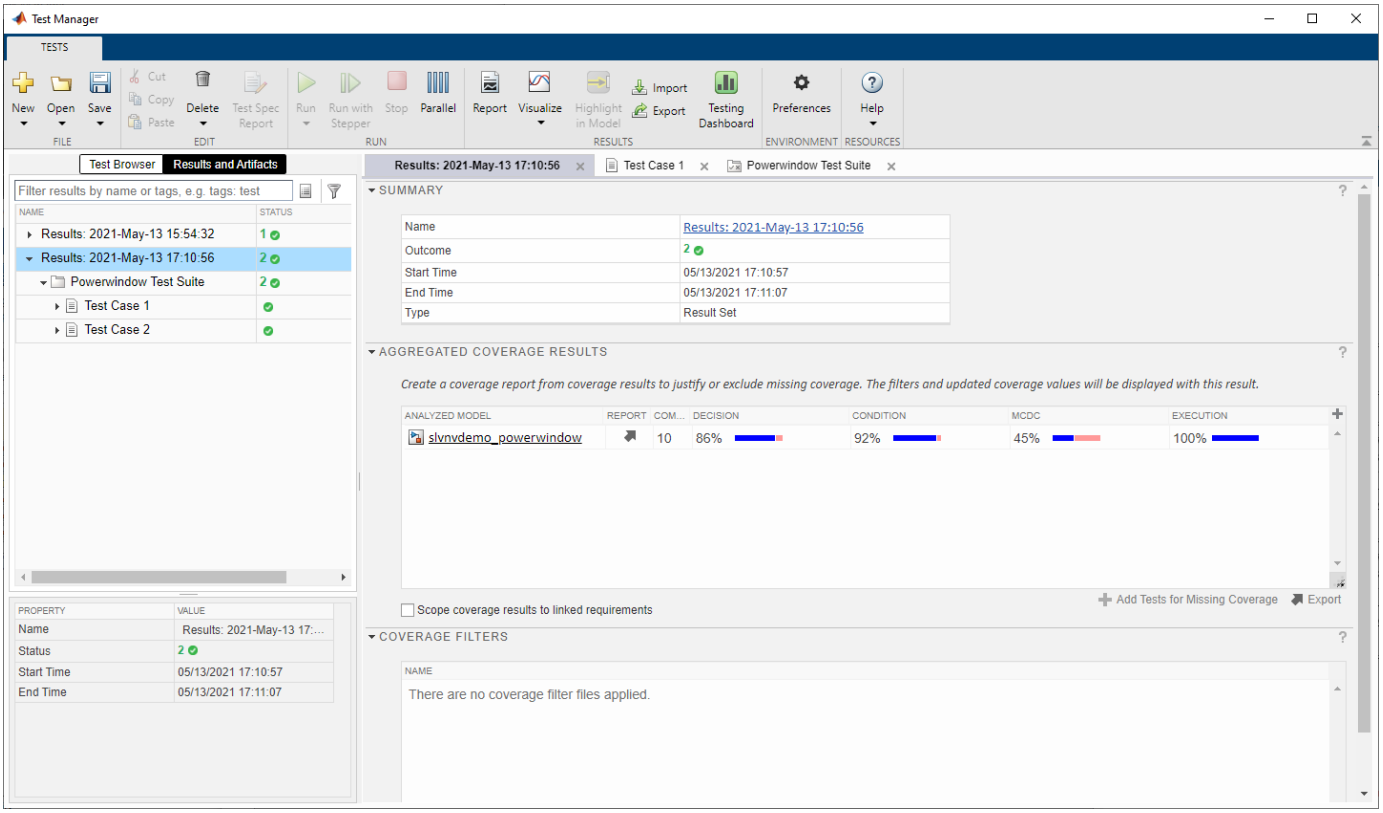
In the `slvnvdemo_powerwindow` model, the And block `power_window_control_system/validate_passenger/check_up/allow_action` has 75% condition coverage because the Signal Editor block `Input` at the model root level has the **Active scenario** set to `Driver`. This missing coverage can be resolved by including **Test Case 2** in the coverage analysis.

In the Test Manager, open the **Test Browser** tab. Click **Powerwindow Test Suite**, then click **Run** to run the tests in the test suite. When the tests finish, the Test Manager opens the **Results and Artifacts** pane. Expand results for the new test. You can view the coverage results for each individual test by expanding each test case.

Select **Powerwindow Test Suite**, then click the model name, `slvnvdemo_powerwindow`, to open the model with coverage highlighting. Navigate to `power_window_control_system/validate_passenger/check_up` and click the And block. In the coverage report, the condition coverage is now 100%. Additionally, the **T1** and **T2** link in the condition and MCDC tables link to the

test that satisfies each objective. For example, the `true` case of the first condition was satisfied by test case 2, **T2**. Click the link to scroll to the **Aggregated Tests** section of the report.

In the Test Manager, in the **Aggregated Coverage Results** table, the overall coverage results improved. When you click on the test suite or the test file name, you see the aggregated coverage results for the tests inside.



The aggregated results show higher coverage percentages than either of the two individual test cases because the test cases satisfy different objectives in some blocks.

Filter Coverage Outcomes

If you analyze the coverage report and find that you are missing coverage that is not possible to fix by changing the model or a test case, you can filter the missing outcomes so that they are not reported as missing coverage. Although you cannot filter coverage directly from the Test Manager, you can apply an existing filter and view the changes in your coverage results. Some possible reasons you might want to filter coverage outcomes include:

- A block is tested by a different test suite, and is not applicable to the current coverage analysis.
- A block is intended to catch edge cases that you think should not occur anyway. This type of model design is sometimes called defensive coding.

There are two types of coverage filters:

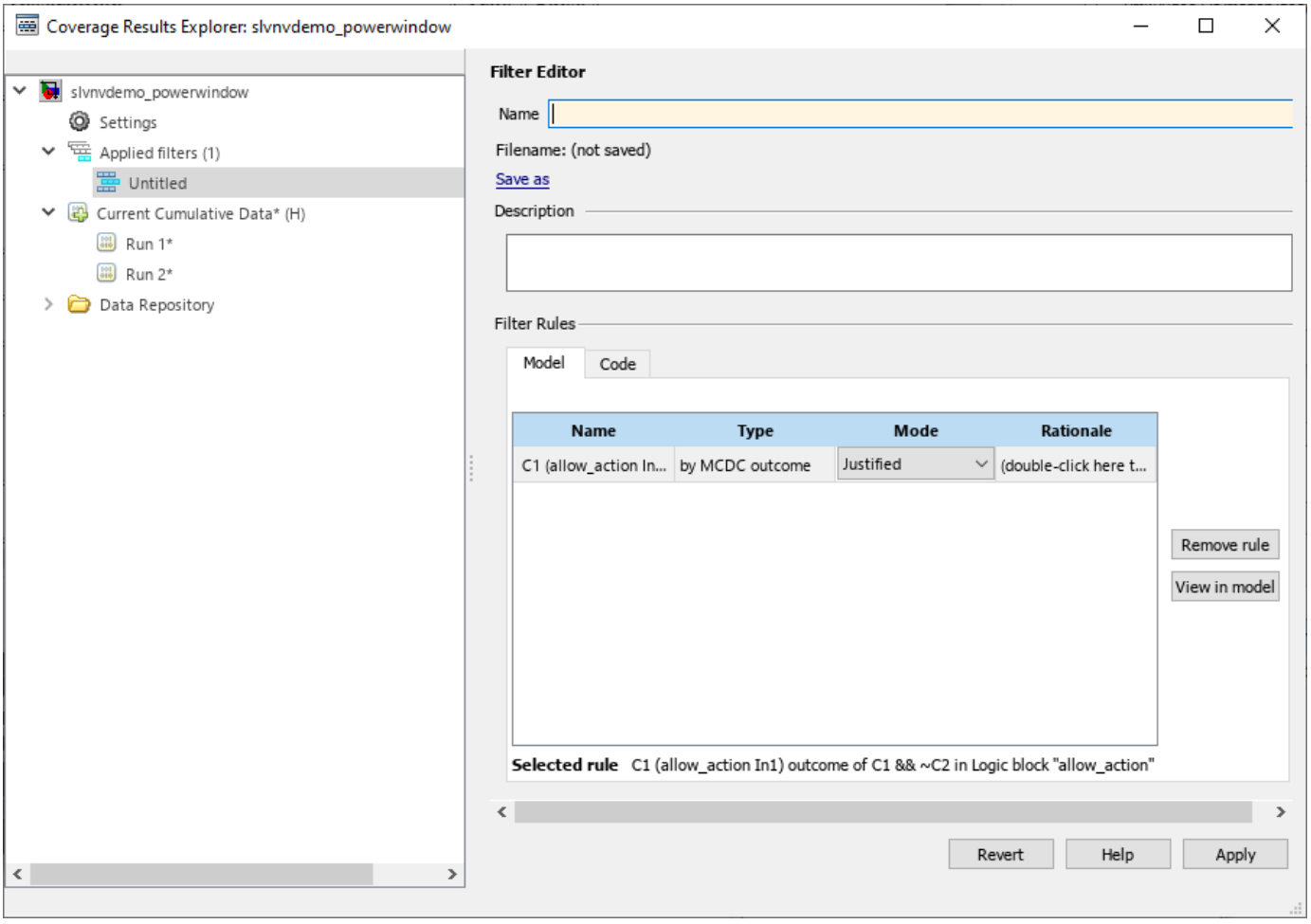
- An exclusion filter rule can be applied to a model element, and causes that element to be ignored by coverage analysis. The excluded model elements appear dimmed in the highlighted model, like other elements that are not applicable to the metrics you selected.

- A justification filter rule can be applied to a coverage outcome that is not satisfied. This filter rule allows Simulink Coverage to analyze the rest of the model element but does not report the justified outcome as missing coverage. This filter rule allows you to improve your coverage for a model object without excluding it entirely.

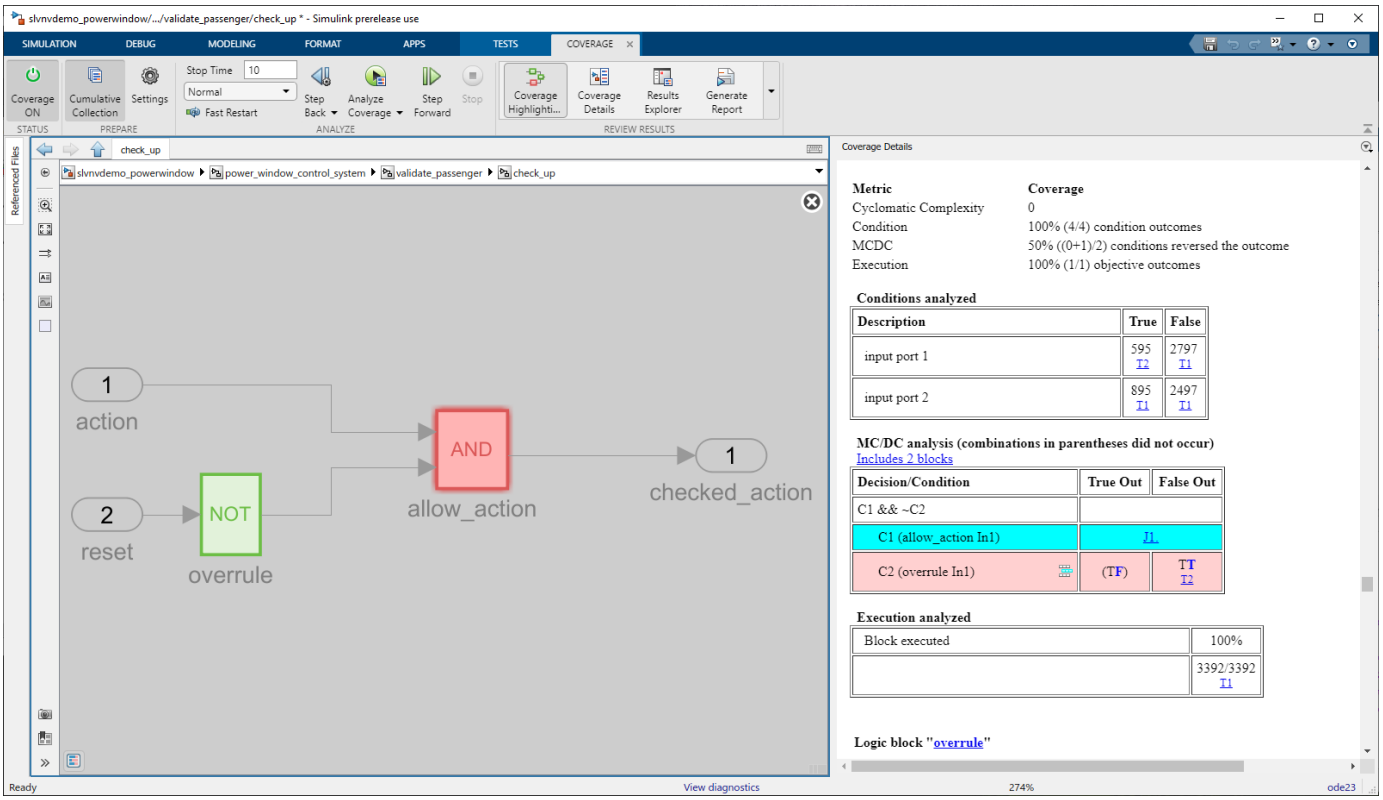
Suppose that the And block condition 1 MCDC outcome has been tested by a different test suite and is not applicable for this case. You can justify the outcome so that it is not reported as missing coverage.

In the model, navigate to the subsystem `power_window_control_system/validate_passenger/check_up` and click the And block `allow_action` to scroll to the relevant section in the coverage report. The MCDC condition `C1 (allow_action In1)` is incomplete because the `TF` case did not occur. To justify the `C1 (allow_action In1)` MCDC outcome, click the Add justification rule icon .

The Coverage Results Explorer opens the **Filter Editor** pane with a new untitled filter file. The filter file contains a justification rule for the specified outcome. You can add multiple filter rules to the same filter file.



In the **Name** field, enter `slvnvdemo_powerwindow_filter`. Under **Filter Rules**, double-click the **Rationale** field and enter `Tested in a different test suite`. Click **Apply**, then save the file. The model and coverage report automatically update to indicate that the outcome is justified.



In the **Coverage Details** pane, the justified outcome is highlighted in cyan and links to the justification rationale. Clicking **J1** brings you to the section of the report titled **Objects Filtered from Coverage Analysis**. This section of the report only appears if you apply one or more filters to the coverage data.

Go back to the Test Manager. Click the result set. In the **Aggregated Coverage Results** section, the coverage filter has been applied automatically, and the MCDC coverage result is now 50%.

Note You must click on a the top-level results to see the applied coverage filters. The filter is not included in test suite or test case results.

To apply another coverage filter file, click the results set, expand the **Coverage Filters** section, and then click **Add**.

Create and Archive Reports

To create a coverage report, click the arrow in the **Report** column. Alternatively, you can click **Export** under the **Aggregated Coverage Results** table. Then use `cvhtml` to create a coverage report. For example, export the results with the variable name `coverageData` and then enter:

```
cvhtml('testManager_covData',coverageData)
```

See Also

Related Examples

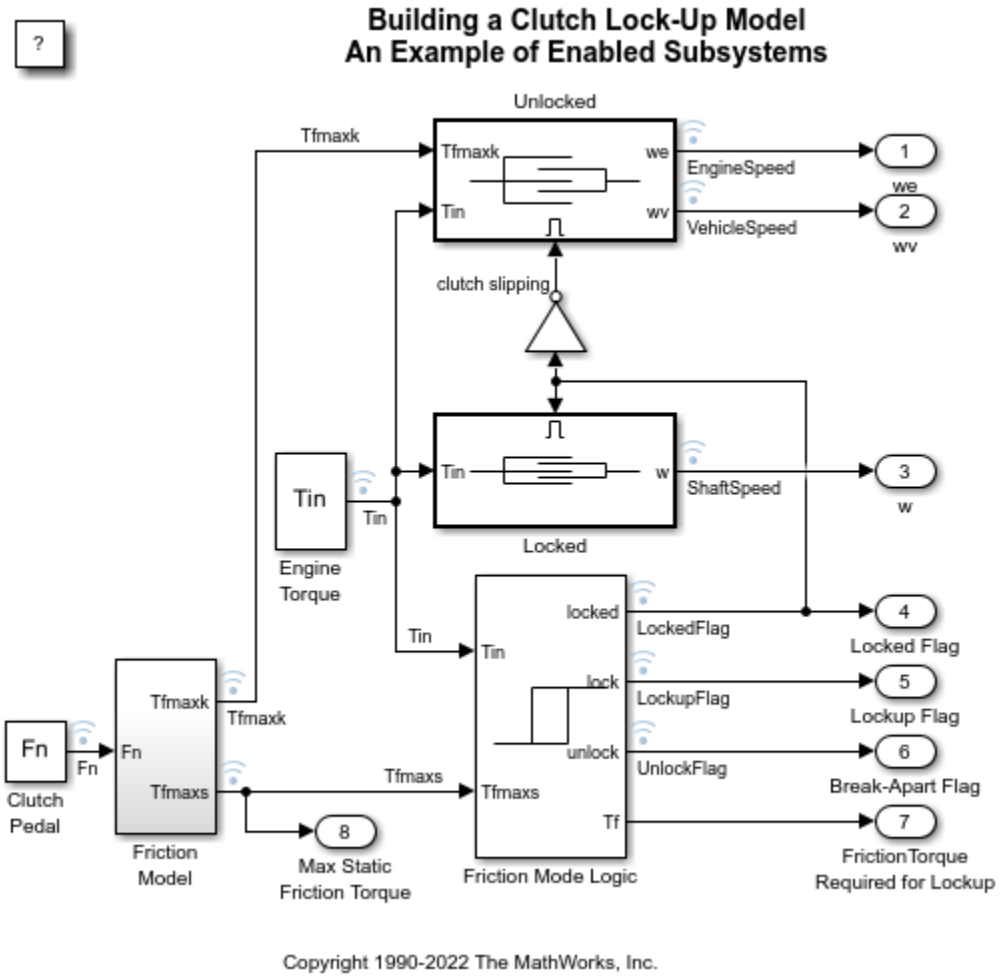
- “Trace Coverage Results to Requirements”
- “Assess Coverage Results from Requirements-Based Tests”
- “Trace Coverage Results to Associated Test Cases”

Basic Operation of the Model Coverage Tool

This example shows how to use the Configuration Parameters dialog to enable coverage for a Simulink® model and adjust the type of information that is reported.

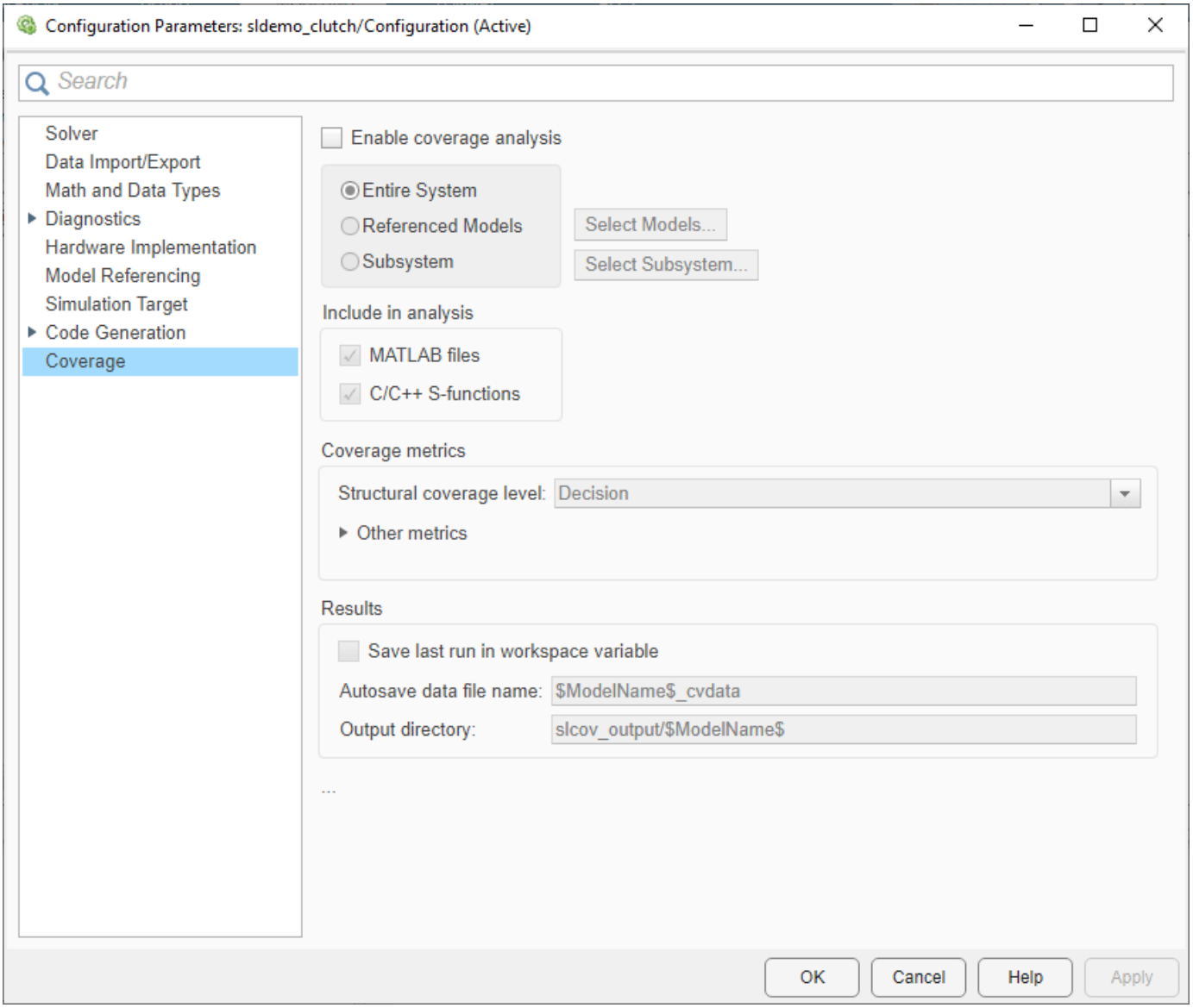
Open Example Model

The `sldemo_clutch` example model that ships with Simulink provides a good introduction to model coverage. This model contains several blocks that have intrinsic decisions, places where control flow can take more than one path.



Open Coverage Settings Dialog

You can find coverage settings under the **Coverage** pane of the Configuration Parameters dialog box. To navigate to this pane, from the **Modeling** tab, click **Model Settings**.



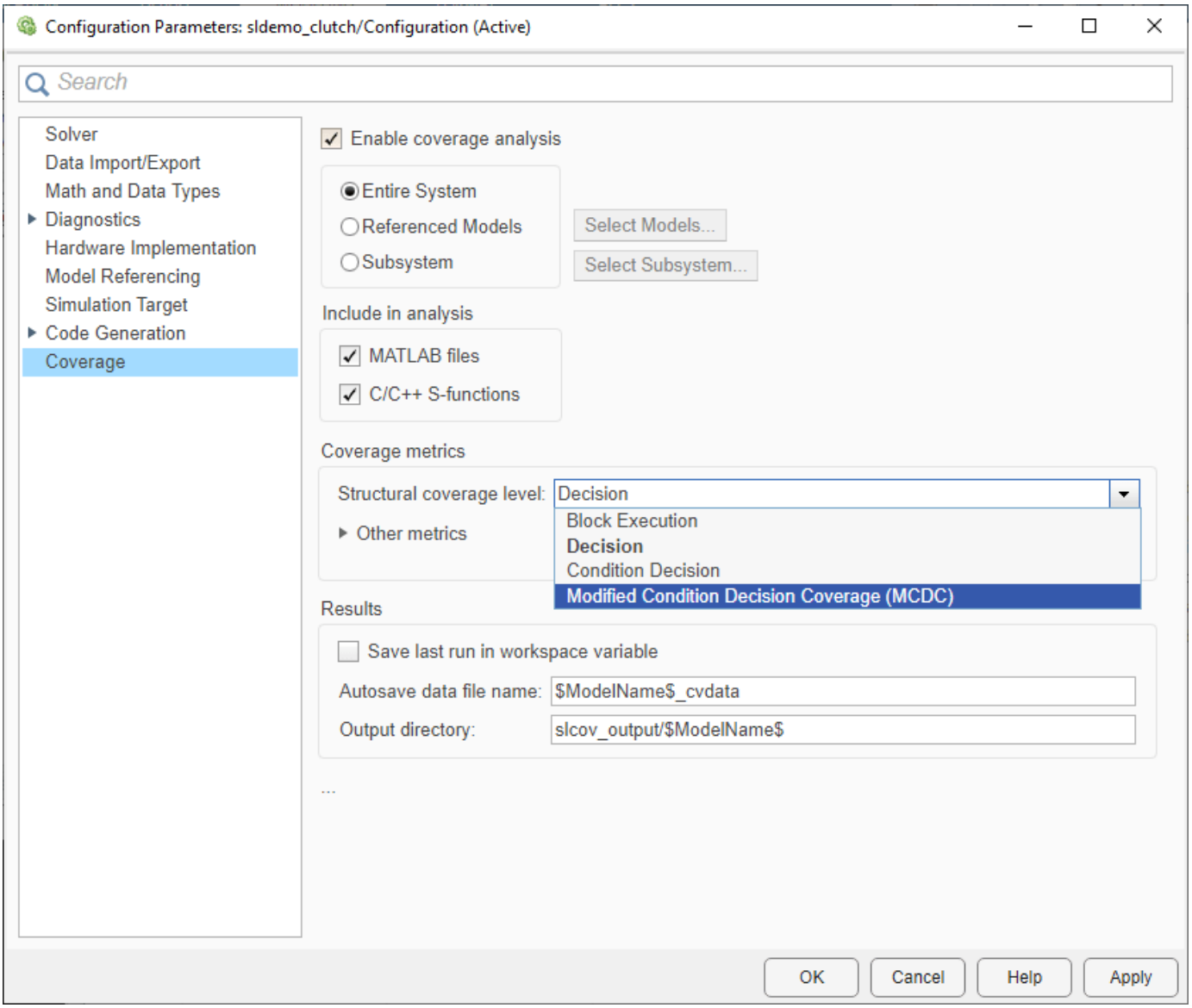
Enable Multiple Coverage Metrics

To enable coverage, select **Enable coverage analysis**. This setting enables the other options in the Coverage pane.

The Coverage metrics section controls the types of coverage information to collect during simulation.

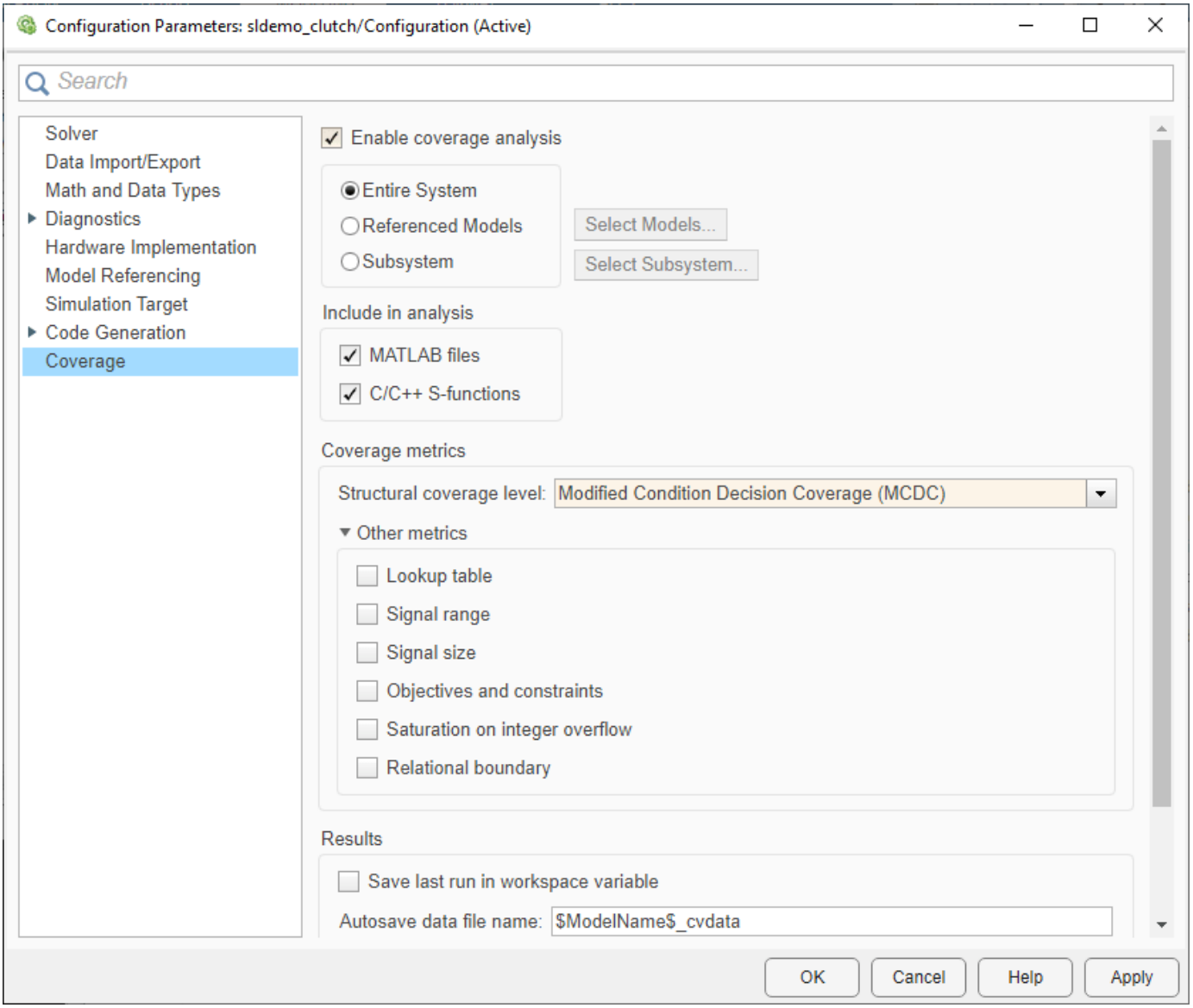
The **Structural coverage level** drop-down menu allows you to select which structural coverage metrics to analyze during the simulation. These are ordered from the least rigorous (*Block Execution*) to the most rigorous (*Modified Condition Decision (MCDC)*).

For this example, select the *Modified Condition Decision (MCDC)* structural coverage level. The resulting report also includes decision and condition coverage results.



You can find additional coverage metrics under the **Other metrics** toggle panel. Click the black arrow to expand this panel and see the available metrics. Selecting all coverage metrics provides the most coverage information.

For this example, select **Signal Range** and **Lookup Table**.

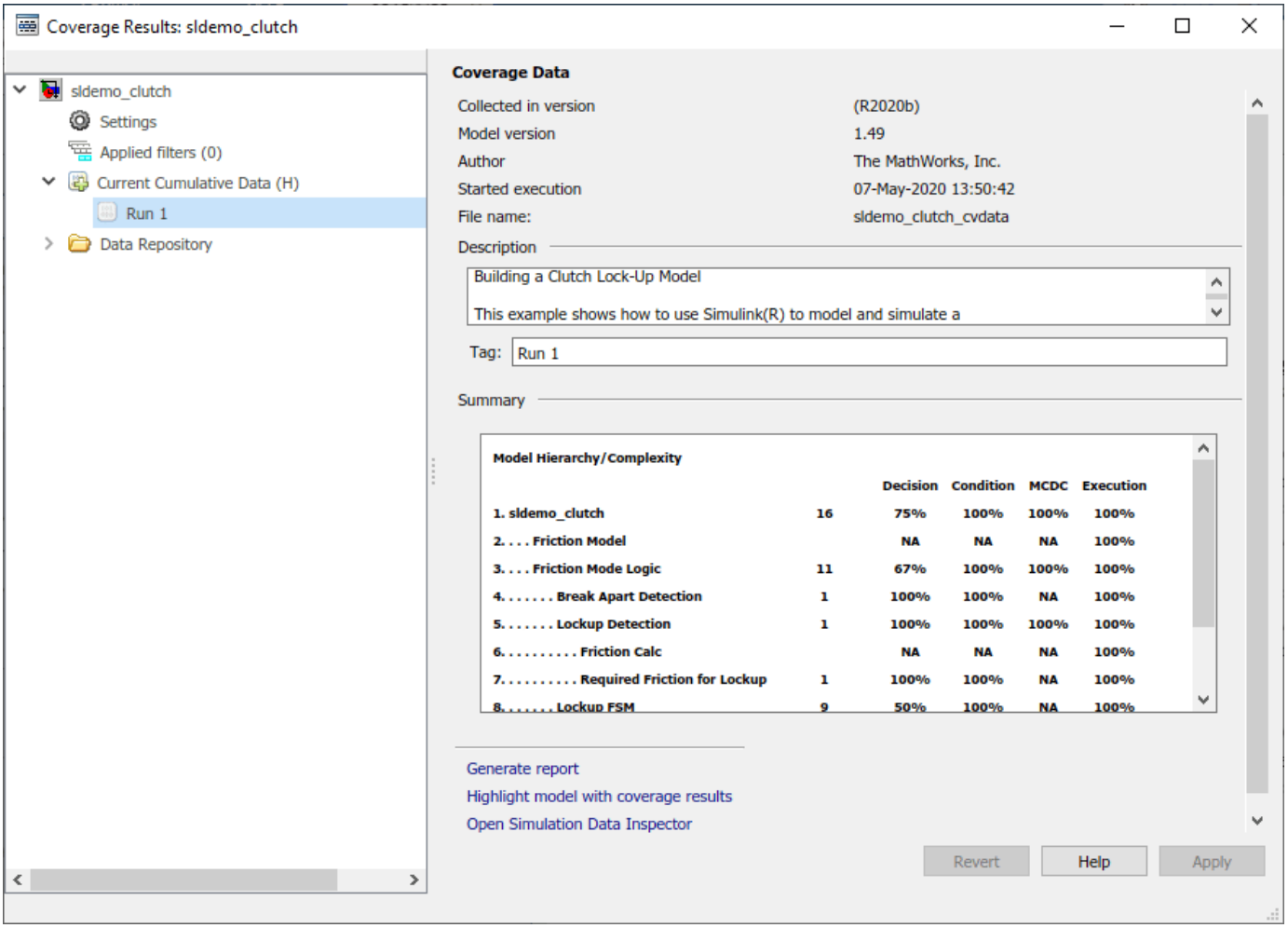


Click OK to apply your selected settings and close the Coverage Settings dialog box.

Run Simulation and Explore Results

Once you enable coverage analysis for your model, simulate the model to collect coverage information. At the end of the simulation, the model **Coverage Details** pane opens, showing results for the model. Additionally, the model displays a coverage highlighting overlay that visually indicates coverage completeness for each object in the model.

Click the **Run (Coverage)** button to simulate the model and collect coverage.



Produce Coverage Report from Results Explorer

After the simulation completes, in the **Coverage** tab of the toolstrip, click **Results Explorer**. The Coverage Results Explorer offers various functionality for processing, displaying, and reporting coverage results for your model.

Click the **Generate report** link at the bottom of the Coverage Results Explorer window. This produces an HTML report of coverage information that displays within the built-in MATLAB® web browser.

Coverage Report for sldemo_clutch

Table of Contents

Analysis Information

Coverage Data Information

Collected in version	(R2020b)

Model Information

Model version	1.49
Author	The MathWorks, Inc.
Last saved	Wed Apr 15 08:09:26 2020

Simulation Optimization Options

Default parameter behavior	tunable
Block reduction	off
Conditional branch optimization	on

Coverage Options

Analyzed model	sldemo_clutch
Logic block short circuiting	on
MCDC mode	masking

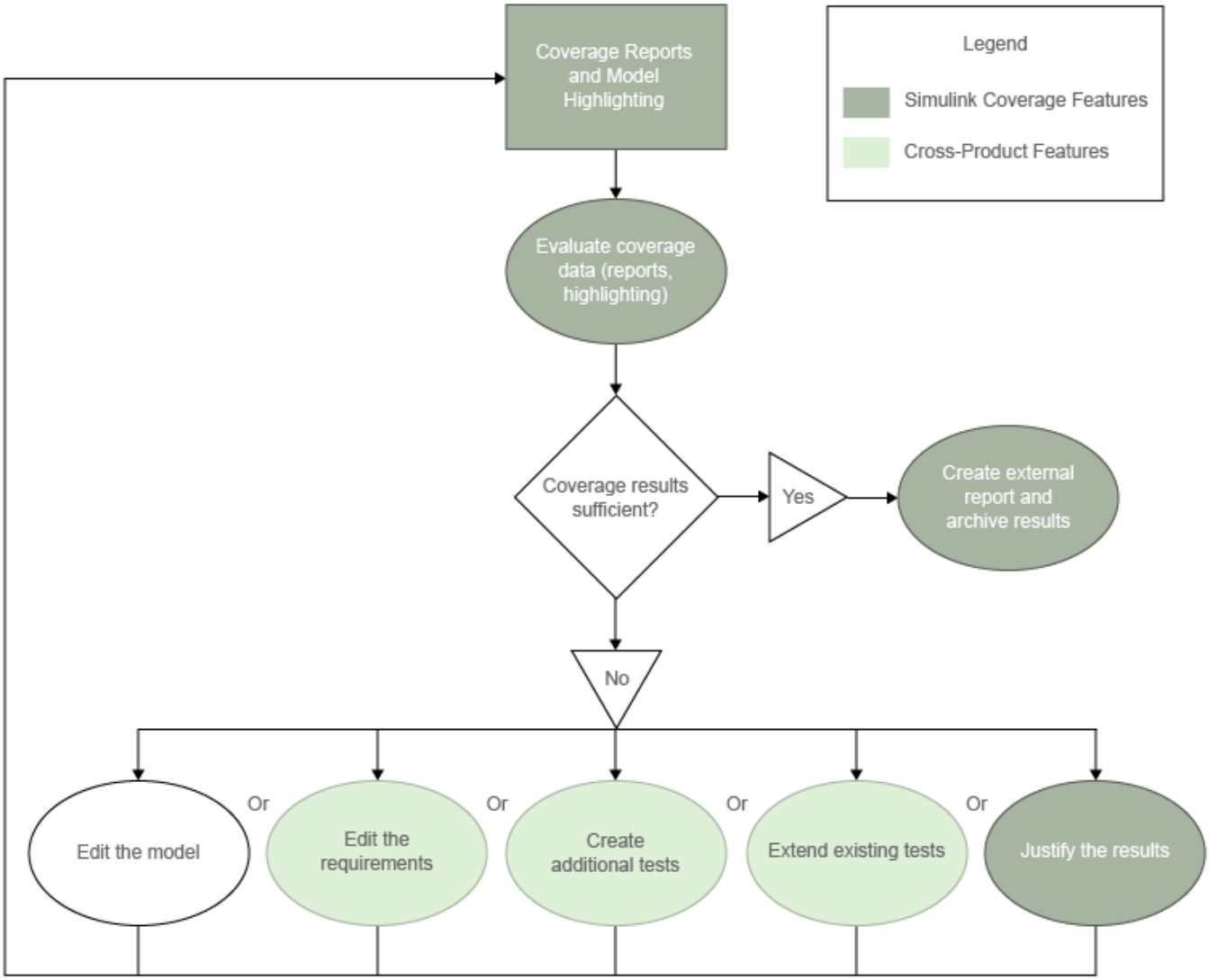
Tests

Test#	Started execution	Ended execution	Description
Test 1	07-May-2020 13:50:42	07-May-2020 13:50:44	Building a Clutch Lock-Up Model This example shows how to use Simulink(R) to model and simulate a rotating clutch system. Although modeling a clutch system is difficult because of topological changes in the system dynamics during lockup, this example shows how Simulinks enabled subsystems feature easily handles such problems. We illustrate how to employ important Simulink modeling concepts in the creation of the clutch simulation. Designers can apply these concepts to many models with strong discontinuities and constraints that may change dynamically. In the example, we use enabled subsystems to build the clutch model. Two enabled subsystems model the clutch dynamics in either the locked or unlocked position. After running the simulation, a GUI opens. Checking any of the boxes on the GUI produces a plot of any of the selected variables (versus time).

How to Address Coverage Testing Gaps

In this section...

After you generate a model coverage report, you can use Simulink Coverage to help you address coverage testing gaps for your model.



Recording and Evaluating Coverage

After you enable coverage recording and simulate your model, you then evaluate the coverage data to find gaps in your testing. Coverage reports and model highlighting help you to understand which parts of your design are sufficiently exercised by your test cases.

For more information, see:

- “Specify Coverage Options”
- “View Coverage Results in Simulink Canvas”
- “Types of Coverage Reports”
- “Overview of Model Coverage Highlighting”

Addressing Coverage Testing Gaps

If your coverage tests do not sufficiently exercise your model, there are a few steps you can take to increase your coverage:

- **Edit the model** — Your model might contain unintended functionality that is not part of your required design. Remove the unintended functionality.
- **Edit the requirements** — Your requirements might not sufficiently detail the testing required to exercise all parts of your design. You can use Requirements Toolbox to author and edit requirements from within the Simulink environment. For more information, see “Test Model Against Requirements and Report Results”.
- **Create additional tests** — Your existing tests might not fully exercise the intended simulation inputs according to your requirements. You can use Simulink Coverage to create additional tests to model these inputs. For more information, see “Automating Model Coverage Tasks” and “Generate Test Cases” (Simulink Design Verifier).
- **Extend existing tests** — Your existing tests might not fully exercise all parts of your design. If you have a Simulink Design Verifier license, you may be able to automatically generate additional tests to exercise untested parts of your design. For more information, see “Incrementally Increase Test Coverage Using Test Case Generation”.
- **Justify the results** — Parts of your model might not be exercised during simulation according to your design, such as subsystems that only enable during failures. To achieve full coverage in such cases, you can justify the missing coverage. For more information, see “Create, Edit, and View Coverage Filter Rules”.

Reporting and Archiving Results

If your coverage tests sufficiently exercise your model, you can archive the reported results. For more information, see:

- “Top-Level Model Coverage Report”
- “Access, Manage, and Aggregate Coverage Results”
- “Code Coverage Report”
- “Export Model Coverage Web View”
- `cvhtml`

See Also

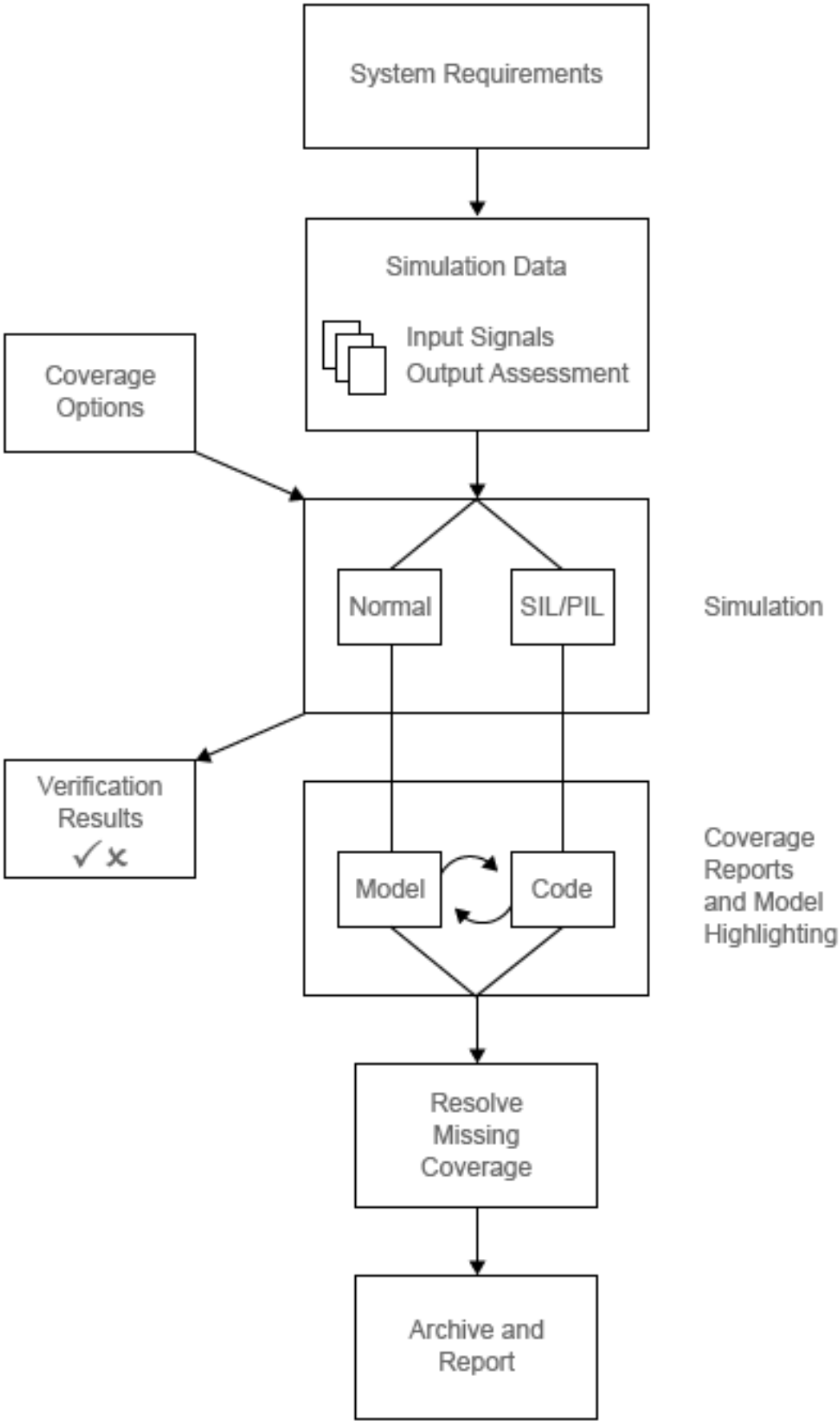
More About

- “Perform Functional Testing and Analyze Test Coverage”

Simulink Coverage in End-to-End Systematic Verification

In this section...

You can use Simulink Coverage to increase confidence in your design and testing through end-to-end systematic verification.


System Requirements
Simulation Data
Input Signals
Output Assessment
Coverage Options
Normal
SIL/PIL
Simulation
Verification Results
Model
Code
Coverage Reports and Model Highlighting
Resolve Missing Coverage
Archive and Report

System Requirements

End-to-end systematic verification begins with system requirements. The system requirements detail the model design and the testing and verification specifications.

If you have a Requirements Toolbox license, you can author and manage system requirements, link Simulink blocks to requirements, and link test cases to the requirements that they verify from within the Simulink environment. For more information, see “Test Model Against Requirements and Report Results”.

Simulation Data

To record coverage, you first specify input signals for your model. For more information, see “Create and Run Test Cases” and “Perform Functional Testing and Analyze Test Coverage”.

If you have a Simulink Design Verifier license, you can also specify expected output signals for the specified input signals. For more information, see “Perform Functional Testing and Analyze Test Coverage” (Simulink Design Verifier). You can also automatically generate test case for a system. For more information, see “Workflow for Test Case Generation” (Simulink Design Verifier).

Simulation

After you “Specify Coverage Options” for your model, you then simulate the model and record model coverage. For more information on the types of model coverage, see “Types of Model Coverage”. You can record several runs of coverage data and accumulate the results. For more information, see “Access, Manage, and Aggregate Coverage Results” and “Cumulative Coverage Data”.

If you have an Embedded Coder license, you can also record code coverage for models in software-in-the-loop (SIL) mode and processor-in-the-loop (PIL) mode. The code coverage reports tell you which parts of your generated code are exercised by your tests, and also which elements of your model are associated with which parts of your generated code. For more information on enabling SIL or PIL code coverage for a model, see “Specify Code Coverage Options”.

If you have a Simulink Design Verifier license, you can model and verify design requirements using property proving. For more information, see “Specify and Verify Design Requirements” (Simulink Design Verifier).

Coverage Reports and Model Highlighting

After you record coverage for a simulation, you then review the coverage reports and model highlighting to identify gaps in your testing, with the goal of gaining confidence that your tests and verification are complete and adequate. For more information on coverage reports and model highlighting, see:

- “View Coverage Results in Simulink Canvas”
- “Top-Level Model Coverage Report”
- “Code Coverage Report”
- “Access, Manage, and Aggregate Coverage Results”

If you have an Embedded Coder license and you record code coverage for models in software-in-the-loop (SIL) mode and processor-in-the-loop (PIL) mode, code coverage reports tell you which parts of

your generated code are exercised by your tests, and also which elements of your model are associated with which parts of your model. For more information on enabling SIL or PIL code coverage for a model, see “Code Coverage for Models in Software-in-the-Loop (SIL) Mode and Processor-in-the-Loop (PIL) Mode”.

Resolve Missing Coverage

If you find that parts of your design are not fully tested, you can take steps to resolve the missing coverage and gain confidence that your tests and verification are complete and adequate. For more information, see “How to Address Coverage Testing Gaps” on page 2-40.

Archive and Report

When you are confident in the testing and verification of your design, you can archive and report the results of the coverage recording. For more information, see:

- “Top-Level Model Coverage Report”
- “Code Coverage Report”
- “Access, Manage, and Aggregate Coverage Results”
- “Export Model Coverage Web View”
- `cvhtml`