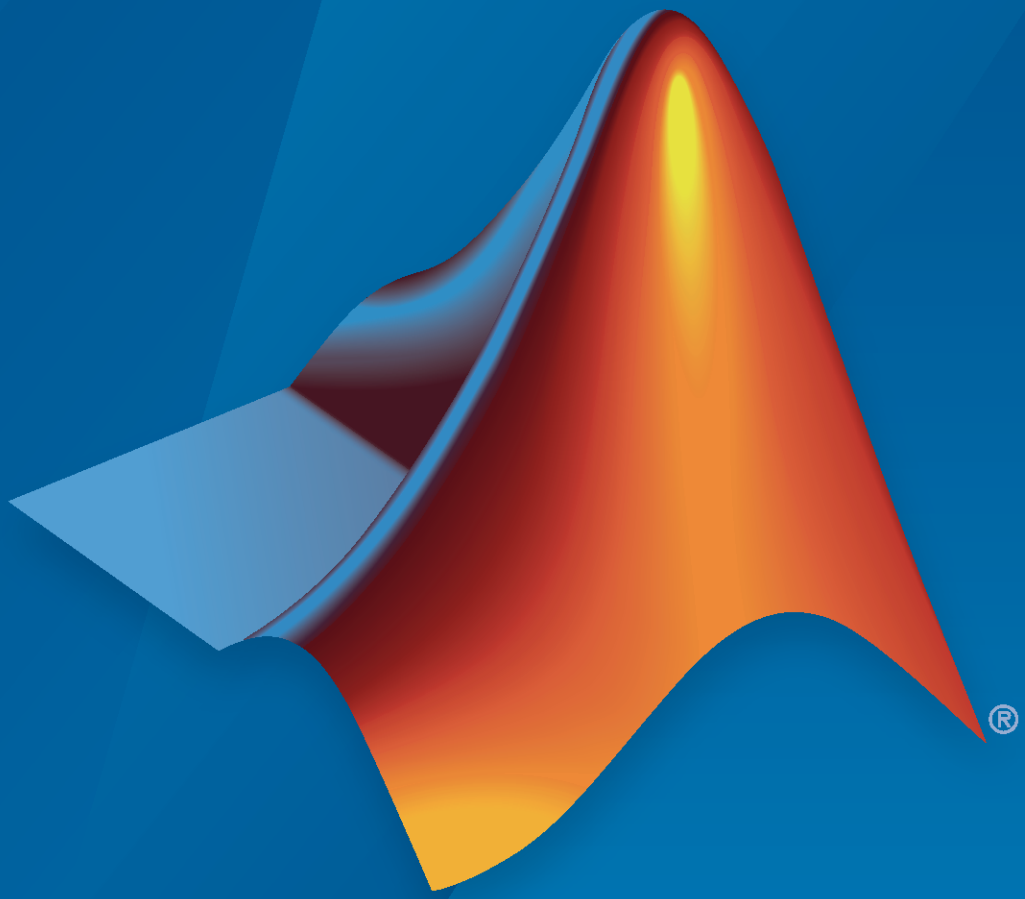
Simulink® Design Verifier™ Release Notes



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® Design Verifier™ Release Notes

Contents

R2020b

R2020a

R2019b

R2019a

R2018b

R2018a

R2017b

R2017a

R2016b

R2016a

R2015aSP1

R2015b

R2015a

R2014b

R2014a

R2013b

R2013a

R2012b

R2012a

R2011b+

R2011b

R2011a

R2010bSP1

R2010b

R2010a

R2009bSP1

R2009b

R2009a

R2008b

R2008a+

R2008a

R2007b+

R2007b

R2007a+

R2020b

Version: 4.4

New Features

Bug Fixes

System object analysis

You can now analyze the models that contain System object™ in MATLAB® System blocks and MATLAB Function blocks. You can:

- Generate test cases for condition, decision and MCDC objectives
- Check for errors, including out-of-bound array access, division-by-zero, integer overflow, dead logic, non-finite and NaN floating-point values, subnormal floating-point values, and data store access violations
- Run property proving analysis

For more information see, “Supported and Unsupported Simulink Blocks in Simulink Design Verifier”.

Detect violations of high-integrity systems modeling guidelines

You can now detect violations of the following High-Integrity Systems Modeling (HISM) checks:

- hisl_0002: Usage of Math Function blocks (rem and reciprocal).
- hisl_0003: Usage of Square Root blocks
- hisl_0004: Usage of Math Function blocks (natural logarithm and base 10 logarithm)
- hisl_0028: Usage of Reciprocal Square Root blocks

For details, see “Design Error Detection Pane Overview”.

For more information about high-integrity systems modeling checks, see “Model Checks for High Integrity Systems Modeling”.

To detect violations of HISM checks:

1. In the **Design Verifier** tab, in the **Mode** section, select **Design Error Detection**.
2. Click **Error Detection Settings**.
3. Under the **Design Error Detection** section of the **Design Verifier** configuration parameters, select the checks that you want to run in **High-Integrity Systems Modeling Checks**.

You can also detect violations of HISM checks by using the Model Advisor.

For more information on using the Model Advisor, see “High Integrity System Modeling Checks” and “Simulink Design Verifier Checks”.

Test case generation: Simplified parameters, progress indicators, and additional report details

R2020b simplifies the options and reports for generating test cases for missing coverage.

In the Configuration Parameters dialog box, the following parameters are updated under **Design Verifier > Test Generation > Advanced Parameters > Add tests for the missing coverage**:

- **Extend using existing coverage data** – Extends the coverage by generating new tests when you provide a coverage data file.
- **Extend using existing test data** – Extends the coverage by adding new tests when you provide a test data file.

- **Separate objectives satisfied with the existing test/coverage data in the report** – Impacts the reporting of the test objectives status.

The **Test Objectives Status** section of Simulink® Design Verifier™ analysis report provides additional information for objectives satisfied with existing test or coverage data. When enabled, the **Test Objectives Status** section shows separate tables for:

- Objectives satisfied from current analysis
- Objectives satisfied using existing test or coverage data

The progress bar in the Results Summary window starts with a preloaded count of objectives satisfied by the test and coverage data and shows the progress for current analysis. The Results Summary window also includes a new status to show the objectives count satisfied by the coverage and test data.

To configure test case generation for missing coverage:

1. In the **Design Verifier** tab, in the **Mode** section, select **Test Generation**.
2. Click **Test Generation Settings**.
3. Under the **Test Generation** section of the **Design Verifier** configuration parameters, select **Advanced Parameters**.

For more information, see “Design Verifier Pane: Test Generation”.

Usability improvements for dead logic detection

R2020b simplifies the options and analysis results for dead logic detection.

- You can perform a partial check for dead logic using the **Dead logic (partial)** parameter. The partial check does not replace an exhaustive analysis. You can run a partial check when you need to more quickly debug errors, before investing the computing time to perform an exhaustive analysis. To perform an exhaustive analysis, enable **Dead logic (partial) > Run exhaustive analysis** in the Configuration Parameters dialog box.
- The Results Summary window shows a notification when Simulink Design Verifier runs a partial check for dead logic.
- The dead logic detection reflects the validation of active logic in the Results Summary window.

To detect dead logic in your model:

1. In the **Design Verifier** tab, in the **Mode** section, select **Design Error Detection**.
2. Click **Error Detection Settings**.
3. Under the **Design Error Detection** section of the **Design Verifier** configuration parameters, select **Dead logic (partial)**.

For more information, see “Dead Logic Detection” and “Design Error Detection Pane Overview”.

R2020a

Version: 4.3

New Features

Bug Fixes

Design error detection for specified block input range violations

You can use design error detection analysis to detect specified input range violations for these blocks:

- n-D Lookup Table
- Interpolation Using Prelookup
- Prelookup
- Direct Lookup Table (n-D)
- Multiport Switch
- Trigonometric Function, when the **Approximation method** parameter is set to `CORDIC`

To detect specified block input range violations in your model, on the **Design Verifier** tab, in the **Mode** section, select **Design Error Detection**. Click **Error Detection Settings**. In the Configuration Parameters dialog box, on the **Design Verifier > Design Error Detection** pane, select **Specified block input range violations**.

You can also detect block input range violations by using the Model Advisor. For more information, see Detect Block Input Range Violations.

Validate test cases or counterexamples in parallel for faster simulations

If you have a Parallel Computing Toolbox™ license, you can validate test cases or counterexamples in parallel across multiple workers on the same machine for faster simulations. To enable the use of parallel computing, on the **Design Verifier** tab, in the **Prepare** section, select **Settings**. In the Configuration Parameters dialog box, on the **Design Verifier** pane, under **Advanced parameters**, select **Validate test cases or counterexamples with parallel computing**.

By default, Simulink Design Verifier uses fast restart mode to simulate the model when you use `sldvruntest`. To also enable the use of parallel computing for faster simulations, set the `useParallel` option in `sldvruntest`. For more information see, sldvruntest.

Improved reporting to capture dead logic causes

Simulink Design Verifier now reports possible causes for dead logic in the Results Inspector window. These causes are reported when dead logic occurs due to:

- Short-circuiting of a Logical Operator block during analysis
- Conditional execution of a block when the Conditional input branch execution (Simulink) parameter is set to **On**

Improved performance for missing coverage workflows

Simulink Design Verifier analysis performance while using existing test cases or existing coverage data is improved. In some cases, the analysis is faster and more objectives are satisfied. For more information, see Defining and Extending Existing Tests Cases and Missing Coverage in Subsystems and Model Blocks.

Analyze models that contain bus element port blocks

Simulink Design Verifier now supports analysis of models that contain In Bus Element and Out Bus Element blocks in top-level models. Before R2020a, analysis was only supported when the bus element port blocks were in the Subsystem block. For more information, see Simplify Subsystem and Model Interfaces with Buses (Simulink).

Support for C Script blocks

Simulink Design Verifier now supports analysis of models that contain C Script blocks.

Analyze models that contain instance-specific parameters

Simulink Design Verifier supports analysis of models that are configured to use instance-specific parameters for referenced models. For more information, see Parameterize a Referenced Model (Simulink).

Support for Stateflow Variants

Simulink Design Verifier supports analysis of models that contain Variant Transitions in Stateflow®. For more information, see Software Configurations Using Variant Transitions (Stateflow) documentation.

R2019b

Version: 4.2

New Features

Bug Fixes

Detect data store memory access violations

In R2019b, you can use design error detection analysis to detect these access violations in Data Store Memory blocks:

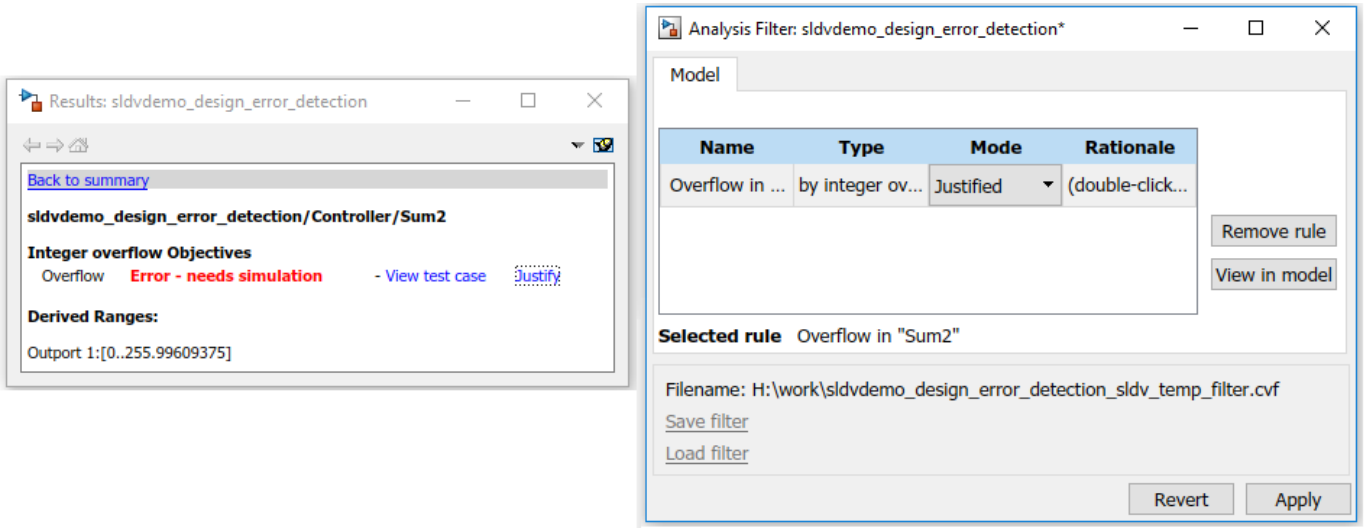
- Read-before-write
- Write-after-read
- Write-after-write

To detect data store memory access violations in your model, on the **Design Verifier** tab, select **Settings**. In the Configuration Parameters dialog box, on the **Design Verifier > Design Error Detection** pane, select **Data store access violations**. For more information, see Detect Data Store Access Violations in a Model.

You can also detect data store access violations by using the Model Advisor. For more information, see Detect Data Store Access Violations.

Enable justification capability in design error detection analysis

In R2019b, you can justify design error detection objectives by using the Analysis Filter window. To open the Analysis Filter window, after Simulink Design Verifier analysis completes, in the Results Inspector window, click **Justify**. For more information, see Filter Objectives by Using Analysis Filter Viewer.



Alternatively, you can justify objectives by using a filter file. On the **Design Verifier** tab, select **Settings**. In the Configuration Parameters dialog box, on the **Design Verifier** pane, in **Advanced parameters**, select **Ignore objectives based on filter**, then browse to the **Filter file**. For more information, see Design Verifier Pane. Before R2019b, the option **Ignore objectives based on filter** was available in Design Verifier Pane: Test Generation.

Detect design errors in custom code

Simulink Design Verifier now supports design error detection in C/C++ custom code in models and Stateflow charts. For more information, see Detect Design Errors in C/C++ Custom Code.

Share model representation cache files for faster analysis

In R2019b, performing Simulink Design Verifier analysis on a model creates a Simulink cache file (`.slxc`). For faster analysis, you can share this cache file. Simulink Design Verifier can reuse the model representation for analysis from this cache file if no changes are detected in the model. By default, the SLXC file appears in your current folder, but you can control the default location. For more information, see Share Simulink Cache File for Faster Analysis.

By default, the **Rebuild model representation** option is set to `If change is detected`. The analysis always reuses the model representation when no change is detected in the model. For more information, see Reuse Model Representation for Analysis.

Debug property proving violations using Model Slicer

You can now use Model Slicer along with Simulink Design Verifier to debug property proving violations. To launch the **Model Slicer**, open **Apps** and, under **Model Verification, Validation, and Test**, select **Design Verifier**. In the **Mode** section, select **Property Proving**, then click the **Debug with Slicer** button. For more information, see Debug Property Proving Violations by Using Model Slicer.

Enable test case extension in the presence of parameter configurations

In R2019b, Simulink Design Verifier supports extending test cases in the presence of parameter configurations. You can apply the parameter configurations by using the parameter table or parameter configuration file. For more information see, Extend Existing Test Cases After Applying Parameter Configurations.

Run dead logic analysis in conjunction with other design error checks

In R2019b, you can perform dead logic analysis in conjunction with other design error checks. For more information, see Design Verifier Pane: Design Error Detection.

Support for complex data type

Simulink Design Verifier now supports the use of complex data types in a model for design error detection, test generation, and property proving analysis.

Simulink Design Verifier contextual tabs in the Simulink Toolstrip

In R2019b, the new Simulink Toolstrip includes contextual tabs for Simulink Design Verifier. To open the Simulink Design Verifier tab:

1. Open a Simulink model.

2. On the **Apps** tab, click the arrow on the far right of the **Apps** section .
3. In the **Model Verification, Validation, and Test** gallery, click **Design Verifier**.

The following table shows the mapping from R2019a Simulink Editor to the new **Design Verifier** tab.

R2019a Simulink Editor Menu Bar Item	Design Verifier Tab Equivalent
Analysis > Design Verifier > Check Compatibility > Model	**Check Compatibility**
Analysis > Design Verifier > Check Compatibility > Enable 'Treat as Atomic Unit' to Analyze	Click the Subsystem block and select **Convert > Convert to Atomic Subsystem**.
Analysis > Design Verifier > Check Compatibility > Selected Subsystem	To check the compatibility of a Subsystem block, first convert the Subsystem block to an Atomic Subsystem. Click the Subsystem block, then click **Check Compatibility**.
Analysis > Design Verifier > Detect Design Errors > Model	On the **Design Verifier** tab, in the **Mode** section, select **Design Error Detection**. Click **Detect Design Errors**.
Analysis > Design Verifier > Detect Design Errors > Selected Subsystem	Click the Subsystem block, then click **Detect Design Errors**.
Analysis > Design Verifier > Generate Tests > Model	On the **Design Verifier** tab, in the **Mode** section, select **Test Generation**. Click **Generate Tests**.
Analysis > Design Verifier > Generate Tests > Selected Subsystem	Click the Subsystem block, then click **Generate Tests**.
Analysis > Design Verifier > Generate Tests > Advisor	On the **Design Verifier** tab, in the **Mode** section, select **Test Generation**. Click **Advisor**.
Analysis > Design Verifier > Prove Properties > Model	On the **Design Verifier** tab, in the **Mode** section, select **Property Proving**. Click **Prove Properties**.
Analysis > Design Verifier > Prove Properties > Selected Subsystem	Click the Subsystem block, then click **Prove Properties**.
Analysis > Design Verifier > Results > Active	On the **Design Verifier** tab, in the **Review Results** section, select **Results Summary**.
Analysis > Design Verifier > Results > Load	On the **Design Verifier** tab, in the **Review Results** section, select **Load Earlier Results**.
Analysis > Design Verifier > Options	On the **Design Verifier** tab, in the **Prepare** section, click **Settings**. The name of the **Settings** button is related to the **Mode** that you select. For example, if you select **Test Generation** mode, set the analysis options by clicking **Test Generation Settings**.

R2019a

Version: 4.1

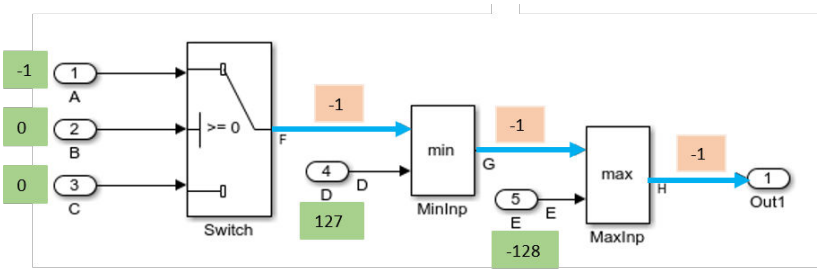
New Features

Bug Fixes

Test generation for enhanced MCDC objectives

Enhanced MCDC is an extension of modified condition decision coverage. For a test block, enhanced MCDC generates test cases that avoid masking effects from downstream blocks, so that the test block has an effect on the output.

For example, a model consists of a cascade of Switch, Min, and Max blocks. The test case generated for the enhanced MCDC coverage ensures that the Switch test block is not masked by the downstream Min and Max blocks. The masking occurs if the other input to the Min block has a smaller value than the Switch output or, if the other input to the Max block has a larger value than the output of the Min block.



To generate test cases for enhanced MCDC, in the Configuration Parameters dialog box, on the **Design Verifier** > **Test Generation** pane, for **Model coverage objectives**, select `Enhanced MCDC`. Then perform test generation analysis. For more information, see Enhanced MCDC Coverage in Simulink Design Verifier.

Default Auto test case generation strategy

In R2019a, you can improve test case generation performance by using the `Auto` option available for the **Test suite optimization** option on the **Design Verifier** > **Test Generation** pane. By default, the test suite optimization option is set to `Auto`. This option adapts the analysis strategy to your model to achieve better analysis performance and precision.

The `CombinedObjectives`, `CombinedObjectives (Nonlinear Extended)`, and `LargeModel` strategies are replaced by the `Auto` strategy. The `LargeModel (Nonlinear Extended)` option is renamed to `Legacy LargeModel (Nonlinear Extended)` in the **Design Verifier** > **Test Generation** pane. For more information, see Test suite optimization.

Performance improvement by reusing the model representation

Before analyzing a model, Simulink Design Verifier performs a compatibility check and saves the model representation. You can reduce the analysis time by reusing the model representation when you re-analyze the model. To reuse the representation, the model must not change, else the representation is rebuilt.

To enable the reuse of the model representation, in the Configuration Parameters dialog box, on the **Design Verifier** pane in the **Advanced parameters** section, set **Rebuild model representation** option to `If change is detected`. By default, the software always rebuilds the model representation during the model analysis. For more information, see Model Representation for Analysis.

The time that the software takes to build or reuse the model representation is reported in the Simulink Design Verifier reports.

Signal Editor block source for test harness models

Simulink Design Verifier now supports the Signal Editor block as the source block for a test harness model. With the Signal Editor block, you can create test input data scenarios and switch between the scenarios during simulation. In the Configuration Parameters dialog box, on the **Design Verifier > Results** pane, select Generate separate harness model after analysis. Then, select `Signal Editor` as the Harness source option. For more information, see Simulink Design Verifier Harness Models.

Model Slicer moves to Simulink Check

Previously, the Model Slicer was available with Simulink Design Verifier. In R2019a, the tool is available with Simulink Check™. In R2018b, you accessed the Model Slicer by selecting **Analysis > Design Verifier > Model Slicer**. In R2019a, you access the Model Slicer by selecting **Analysis > Model Slicer**. For more information on the Model Slicer, see Model Simplification with Dependency Analysis (Simulink Check).

Improved performance for test case extension workflows

In R2019a, test case extension analysis is improved with faster loading of existing test cases for a Simulink model that consists of input bus signals. Or, when existing test cases have repeated input values over time.

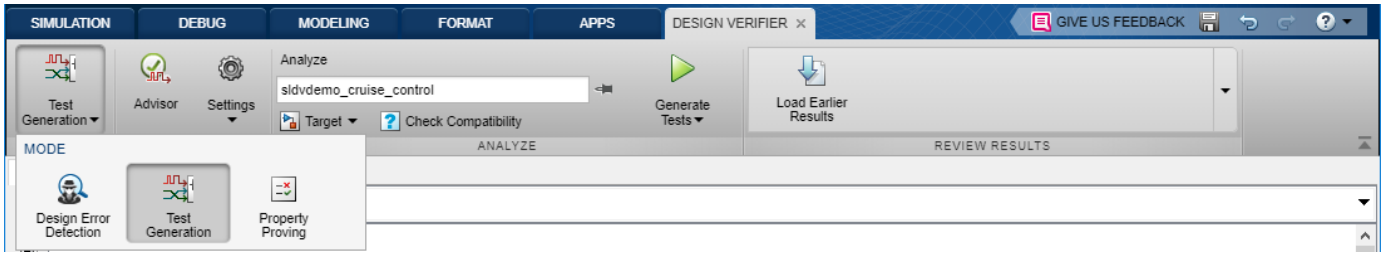
Simulink Design Verifier contextual tabs in the Simulink Toolstrip

In R2019a, you have the option to turn on the Simulink Toolstrip. For more information, see Simulink Toolstrip Tech Preview replaces menus and toolbars in the Simulink Desktop.

The Simulink Toolstrip includes contextual tabs, which appear only when you need them. The Simulink Design Verifier contextual tabs include options for completing actions that apply only to Simulink Design Verifier.

To access the Simulink Design Verifier tab:

1. Open a Simulink model.
2. Activate the Simulink Toolstrip by using the **Simulink Preferences**. See, Simulink Toolstrip Tech Preview replaces menus and toolbars in the Simulink Desktop.
3. On the MATLAB Toolstrip, from the **APPS** tab, select **Design Verifier**.
4. Set up Design Verifier for your analysis mode.



5 Perform an analysis and explore the results.

R2018b

Version: 4.0

New Features

Bug Fixes

Floating-Point Design Error Detection: Detect occurrences of non-finite, NaN, and subnormal floating-point values

You can use design error detection analysis to detect the occurrences of non-finite, NaN, and subnormal floating-point values. To detect non-finite, NaN, and subnormal floating-point values in your model, from the Simulink Editor, select **Analysis > Design Verifier > Options**. In the Configuration Parameters dialog box, on the **Design Verifier > Design Error Detection** pane, select Non-finite and NaN floating-point values and Subnormal floating-point values.

You can also detect the floating-point values by using these Model Advisor checks:

- Detect Non-finite and NaN Floating-Point Values
- Detect Subnormal Floating-Point Values

For more information, see Detect Non-Finite, NaN, and Subnormal Floating-Point Values.

Stateflow Custom Code Support: Analyze and generate tests for C/C++ constructs within Stateflow charts

Use Simulink Design Verifier to perform design error detection and property proving and generate tests for C/C++ constructs within Stateflow charts. For more information on how to enable custom code support, see Coverage for Custom C/C++ Code in Simulink Models (Simulink Coverage) and Test Generation for Custom Code in a Stateflow Chart.

C Caller Block Support: Analyze and generate tests for C/C++ code in blocks

Use Simulink Design Verifier to perform design error detection and property proving and generate tests for the 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 (Simulink Coverage) and Test generation with a C Caller block.

Improved Floating-Point Analysis: Reduce rational approximation for models containing single- and double-precision floating-point arithmetic

When you use Simulink Design Verifier to analyze models, Simulink Design Verifier attempts to reduce the use of rational approximation if the model uses single-precision floating-point values or double-precision floating-point values. This feature applies to all analysis modes. You can disable this feature from the Design Verifier Pane of the model configuration parameters.

For more information on how Simulink Design Verifier introduces approximations, see Floating-Point to Rational Number Conversion.

Export-Function Model Support: Analyze and generate tests for models by automatically creating schedulers that invoke model functions

Simulink Design Verifier now supports design error detection, test generation, and property proving for export-function models. The software automatically creates schedulers that invoke the export-

function models and then perform analysis on scheduled model. For more information, see Analyze Export-Function Models.

Model Slicer: Leverage fast restart to streamline model debugging workflows

Model Slicer now supports the model debugging workflows by using the fast restart methodology. You can use the fast restart mode to:

- Perform multiple simulations efficiently with different inputs, without recompiling the model.
- Debug a simulation by stepping through the major time steps of a simulation and inspecting how slice changes.

For more information, see Debug Slice Simulation by Using Fast Restart Mode.

Model Slicer: Highlight and generate slice for subsystem bus interfaces

Model Slicer now supports highlighting and slice generation for models containing In Bus Element and Out Bus Element ports. You can use bus element port blocks to interface bus signals to subsystems. For more information, see Simplify Subsystem Bus Interfaces (Simulink).

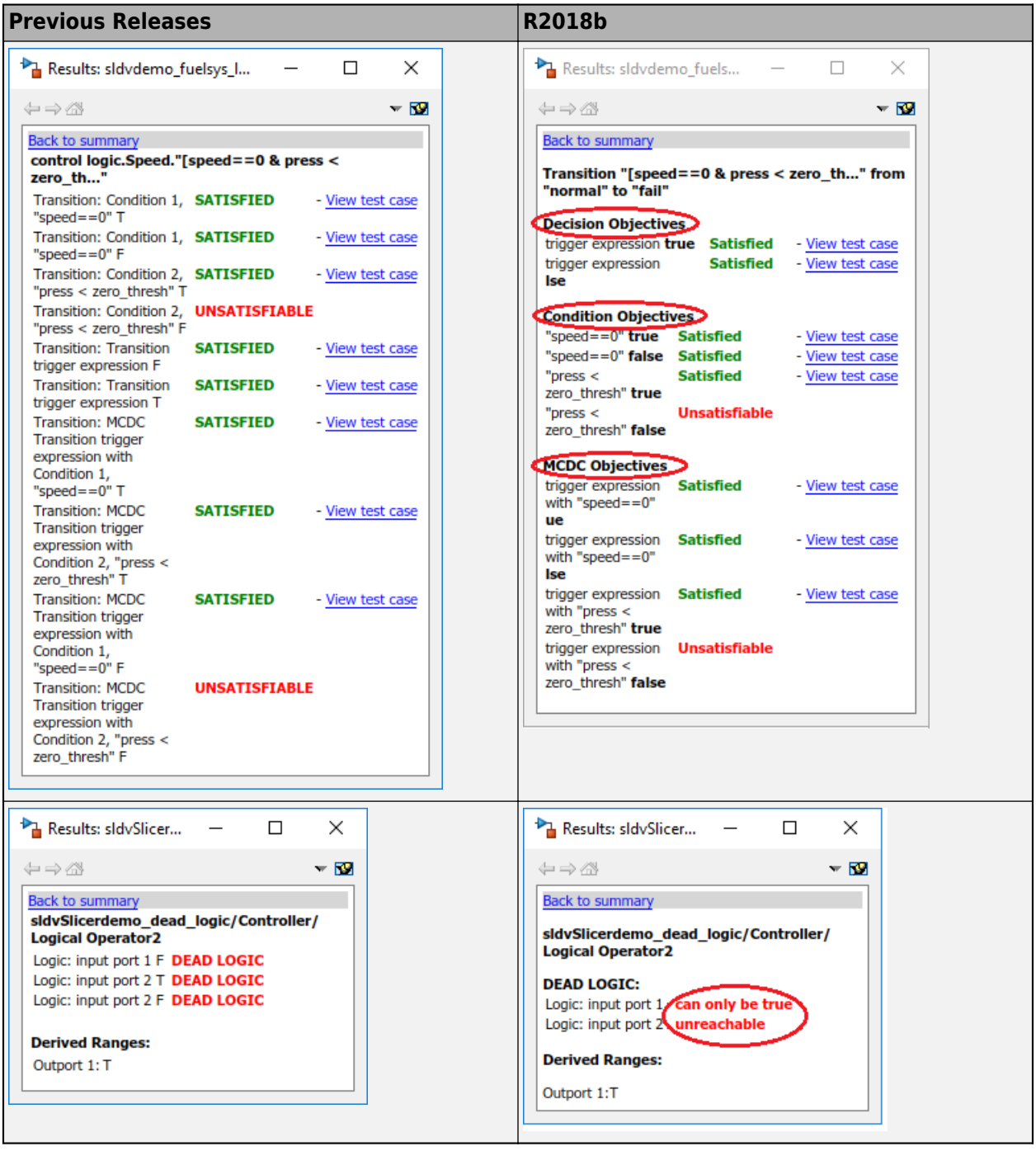
Enhance Model Coverage of Models from Previous Releases

Leverage the latest release capabilities of Simulink Design Verifier on a model from a previous release. Use the workflows described in Enhance Model Coverage of Older Release Models.

Improved Reporting for Test Generation and Dead Logic Analysis Results

In R2018b, reporting Simulink Design Verifier analysis results in the Results Inspector window and reports is improved for better clarity and understanding.

This table shows an example of the test generation and dead logic analysis results in the Results Inspector window in previous releases and R2018b.


Previous Releases
R2018b
Results: sldvdemo_fuelsys_l...
Back to summary
control logic.Speed."[speed==0 & press < zero_th..."
Transition: Condition 1, "speed==0" T SATISFIED - View test case
Transition: Condition 1, "speed==0" F SATISFIED - View test case
Transition: Condition 2, "press < zero_thresh" T SATISFIED - View test case
Transition: Condition 2, "press < zero_thresh" F UNSATISFIABLE
Transition: Transition trigger expression F SATISFIED - View test case
Transition: Transition trigger expression T SATISFIED - View test case
Transition: MCDC Transition trigger expression with Condition 1, "speed==0" T SATISFIED - View test case
Transition: MCDC Transition trigger expression with Condition 2, "press < zero_thresh" T SATISFIED - View test case
Transition: MCDC Transition trigger expression with Condition 1, "speed==0" F SATISFIED - View test case
Transition: MCDC Transition trigger expression with Condition 2, "press < zero_thresh" F UNSATISFIABLE
Results: sldvdemo_fuels...
Back to summary
Transition "[speed==0 & press < zero_th..." from "normal" to "fail"
Decision Objectives
trigger expression true Satisfied - View test case
trigger expression lse Satisfied - View test case
Condition Objectives
"speed==0" true Satisfied - View test case
"speed==0" false Satisfied - View test case
"press < zero_thresh" true Satisfied - View test case
"press < zero_thresh" false Unsatisfiable
MCDC Objectives
trigger expression with "speed==0" ue Satisfied - View test case
trigger expression with "speed==0" lse Satisfied - View test case
trigger expression with "press < zero_thresh" true Satisfied - View test case
trigger expression with "press < zero_thresh" false Unsatisfiable
Results: sldvSlicer...
Back to summary
sldvSlicerdemo_dead_logic/Controller/ Logical Operator2
Logic: input port 1 F DEAD LOGIC
Logic: input port 2 T DEAD LOGIC
Logic: input port 2 F DEAD LOGIC
Derived Ranges:
Outport 1: T
Results: sldvSlicer...
Back to summary
sldvSlicerdemo_dead_logic/Controller/ Logical Operator2
DEAD LOGIC:
Logic: input port 1 can only be true
Logic: input port 2 unreachable
Derived Ranges:
Outport 1:T

R2018a

Incremental Test Generation for Generated Code: Generate additional tests from within the Test Manager to increase coverage of code generated by Embedded Coder

If you have Simulink Test™, you can generate tests to achieve additional coverage for code generated with Embedded Coder® from the coverage results pane in the Test Manager. After executing tests, view the cumulative coverage results in the Test Manager results pane. Select the coverage result and click **Add Tests for Missing Coverage**.

For more information, see Generate Test Cases for Embedded Coder Generated Code and Code Coverage Test Generation.

Dead Logic Refinement for Model Slicer: Improve model slice generation by using static analysis

In R2018a, you can analyze a model slice by refining the dead logic. Use existing Simulink Design Verifier data file to highlight and refine the functional dependencies without resimulating the model. For more information, see Refine Dead Logic for Dependency Analysis.

Export-Function Models: Analyze export-function models driven by scheduler

In R2018a, you can run a Simulink Design Verifier analysis on export-function models that are driven by a scheduler. For more information, see Analyze Export-Function Models.

Precision Improvements: Reduce rational approximations for model analysis

When you analyze models for test case generation, property proving, or dead logic detection, Simulink Design Verifier attempts to reduce the use of rational approximation if the model uses single-precision floating-point values but no double-precision floating-point values. You can disable this feature from the Design Verifier Pane of the model configuration parameters.

For more information on how Simulink Design Verifier introduces approximations, see Floating-Point to Rational Number Conversion.

Test Generation: Avoid run-time errors by using test case refinement

If division by zero or an array out-of-bounds read or write is possible in your model, Simulink Design Verifier satisfies more objectives by generating test cases without division by zero or array out-of-bounds errors.

Compatibility Considerations

In R2018a, if division by zero or an array out-of-bounds read or write is possible in your model, the test or proof objective status might be reported as satisfied or falsified, respectively. Prior to R2018a, these objectives might have been reported as Undecided Due to Division by Zero or Undecided Due to Runtime Error.

Minimum and maximum constraints specified on unused input signals

For test case generation, Simulink Design Verifier considers the minimum and maximum constraints specified on the unused input signals. For more information, see Minimum and Maximum Input Constraints.

Compatibility Considerations

Before R2018a, if your model contained unused input signals with minimum and maximum constraints, simulating the generated test cases produced an error or warning indicating the constraint violation.

Lookup Tables: Use enumerated data types

In lookup tables, Simulink Design Verifier software now supports n-D Lookup Table and Prelookup blocks for enumerated parameters, constants, and inputs.

R2017b

Model Slicer: Investigate and refine active slice-time windows with the Model Slicer Data Inspector

In R2017b, you can use the Model Slicer Data Inspector to:

- Inspect logged signals after model simulation as a part of the Model Slicer workflow.
- Specify the simulation time interval by using the graphical plot to refine the highlighted model.

For more information, see Refine Highlighted Model Slice by Using Model Slicer Data Inspector.

Modeling Support for Secure Coding Standards: Check model for compliance with secure coding requirements in CERT C, CWE, ISO/IEC TS 17961 standards to improve security of generated code

The checks in the Model Advisor for design error detection are included as part of a new set of checks designed to check the model or subsystem for compliance with secure coding requirements in CERT C, CWE, and ISO/IEC TS 17961 standards. To execute these checks, Select and Run Model Advisor Checks (Simulink) and select **By Task > Modeling Guidelines for Secure Coding (CERT C, CWE, ISO/IEC TS 17961)**.

The following table summarizes the checks.

Check	Description	Addresses Secure Coding Standards
Detect Dead Logic	Identifies logic that stays inactive during simulation.	• CERT C, MSC07-C • CWE, CWE-561
Detect Integer Overflow	Identifies operations that exceed the data type range for integer or fixed-point operations.	• ISO/IEC TS 17961: 2013, intoflow • CERT C, INT30-C and INT32-C • CWE, CWE-190
Detect Division by Zero	Identifies operations in the model that cause division-by-zero errors.	• ISO/IEC TS 17961: 2013, diverr • CERT C, INT33-C and FLP03-C • CWE, CWE-369
Detect Out Of Bound Array Access	Detects operations that access outside the bounds of an array index	• ISO/IEC TS 17961: 2013, invptr • CERT C, ARR30-C • CWE, CWE-118

Check	Description	Addresses Secure Coding Standards
Detect Violation of Specified Minimum and Maximum Values	Checks the specified minimum and maximum values (the design ranges) on intermediate signals throughout the model and on the output ports. If the analysis detects that a signal exceeds the design range, the results identify where in the model the errors occurred.	• CERT C, API00-C • CWE, CWE-628

For information about the secure coding standards organizations, see Secure Coding Standards (Embedded Coder).

Multiword Support: Analyze models containing multiword fixed-point data types up to 128 bits

Simulink Design Verifier now supports multiword fixed-point data types up to 128 bits.

Side-Effect-Free Behavior for Design Error Detection Checks

Before R2017b, run-time design error checks (excluding dead logic) could have side effects on other downstream checks. This behavior caused upstream design errors to mask cascaded effects downstream, resulting in Simulink Design Verifier reporting a lower number of error objectives than actually occurred.

In R2017b, downstream design error checks are unaffected by upstream design errors. Consequently, in designs where such cascaded side effects previously existed, you might see more reported error objectives than those reported before R2017b. For more information, see Highlighted Results on the Model.

When you debug counter-examples in simulation, set the relevant **Diagnostics** model configuration parameters to `warning` instead of `error` so that all errors are flagged without halting the simulation.

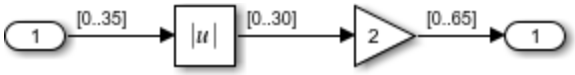
Compatibility Considerations

Simulink Design Verifier analysis and highlighting behavior for models with multiple design errors might be different in R2017b than in previous releases.

- Some objects with design errors that are downstream of other design errors, or which rely on the results of objects with design errors, are now analyzed without side effects from the upstream design errors. Before R2017b, such objects could have been shown as valid. To simplify the debugging of cascaded errors, fix upstream design errors first.
- Derived ranges now report the actual computed range without considering any impacts or side effects of violated design errors, including **Check specified intermediate minimum and maximum values**. Derived ranges now match the results from prior releases for design error detection with all design error checks disabled.
- Because the number of actual reported design errors for a model might increase compared with releases before R2017b, you might see an increased number of reported design errors for a model when you run Simulink Design Verifier analysis as compared with previous releases. This

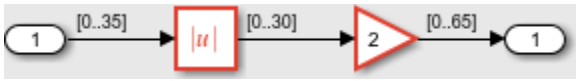
increased number of errors also results in an increased number of test cases for the reported design errors.

Consider the following model, which contains specified minimum and maximum values and two subsequent design errors:



The results of design error detection analysis for this model for **Check specified minimum and maximum values** are different for R2017b than for prior releases. In R2017a and earlier, the Gain block shows no design errors, even though the computed value for its output exceeds `65`. This occurs due to a side effect of the Abs block, which restricts its output range to `[0..30]`:

In R2017b, the downstream design error check is unaffected by the upstream design error, so the Gain block correctly shows a design error:



Result highlighting on the model during analysis

In R2017b, Simulink Design Verifier highlights the model objects automatically during analysis. When the software updates the objective status, the model blocks are highlighted. See Highlight Results on Model Automatically.

Model Slicer: Use Model reference block as slicing context

In R2017b, you can create a slice of a referenced model for debugging and refinement. See Isolate Referenced Model for Functional Testing.

Impact of approximation reported for individual objective status

In R2017b, Simulink Design Verifier identifies the presence of approximation effects and reports them for individual objective status. See Reporting Approximations Through Validation Results.

Compatibility Considerations

In R2017b, Simulink Design Verifier report and highlighting status can differ from previous releases.

- The objectives that approximations impact are marked with a different status when compared to previous releases. For example, for test generation analysis, if the software identifies an objective that approximations impact, the objective status is marked as Undecided with test case. In releases before R2017b, this objective can be marked as Satisfied. For more information see, Objectives Status Chapters.

- The objectives that approximations impact are highlighted in orange. For more information see, Orange Highlighting on Model.

Fast Restart Mode: Identify presence of approximation effects

In R2017b, during analysis, Simulink Design Verifier locks the model in fast restart mode. The model is simulated to identify the presence of approximation effects. See Reporting Approximations Through Validation Results.

R2017a

Activity-Based Time Slicing: Visualize the effect of state activity timing on Model Slicer highlighting for simulations

Visualize the effect of Stateflow states and transitions on model simulation using Model Slicer. You can constrain the Model Slicer highlighting to the simulation intervals in which selected states and transitions are simultaneously active. For more information, see Highlight Active Time Intervals by Using Activity-Based Time Slicing and Using Model Slicer with Stateflow.

Bus Element Port Support: Detect design errors, generate tests, and prove properties for models containing Bus Element ports

Simulink Design Verifier now supports design error detection, test generation, and property proving for models containing In Bus Element and Out Bus Element ports.

Incremental Test Generation: Generate additional tests from within the Test Manager to increase coverage

If you have Simulink Test, you can generate tests to achieve additional coverage from the coverage results pane in the Test Manager. After executing tests, view the cumulative coverage results in the Test Manager results pane. Select the coverage result and click **Add Tests for Missing Coverage**. For an example, see Perform Functional Testing and Analyze Test Coverage.

Improved Analysis Startup Time: Continue analysis after compatibility check

After you perform a Simulink Design Verifier compatibility check on a model, you can continue Simulink Design Verifier design error detection, test generation, or property proving from the compatibility check window.

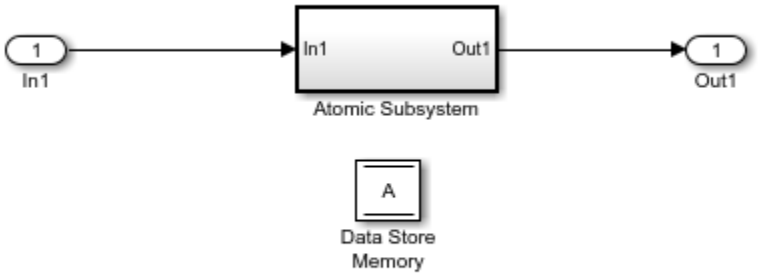
Use Subsystem Inport and Outport blocks as starting points for Model Slicer

You can use Model Slicer to highlight functional dependencies in models by using Subsystem Inport and Outport blocks as starting points. You can use Model Slicer to generate a model slice by using a Subsystem Outport block as a starting point and setting the signal propagation to `upstream`.

Refined sldvextract behavior for models with Data Store Memory blocks

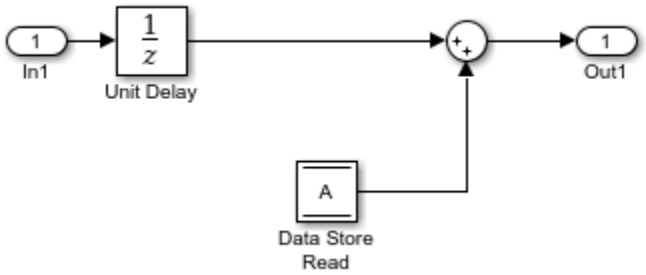
If you use `sldvextract` on a subsystem which references Data Store Memory blocks which are not defined within the subsystem itself, Simulink Design Verifier inserts corresponding Data Store Memory blocks into the extracted model. Simulink Design Verifier also inserts Data Store Write or Data Store Read blocks and inports or outports into the extracted model depending on the contents of the subsystem to be extracted. Consider the following example for more detailed information.

A model contains a Data Store Memory block `A`, and a subsystem.

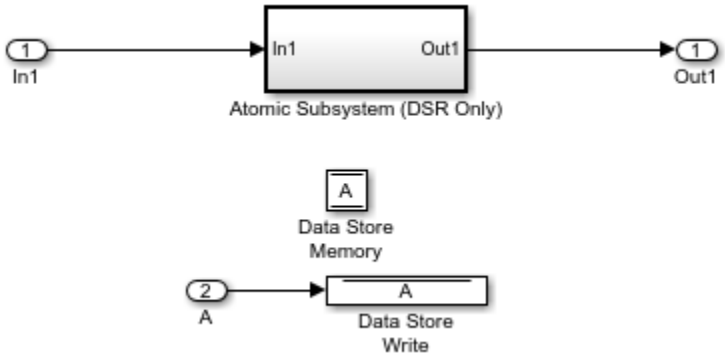


Using `sldvextract` on this subsystem results in different Simulink Design Verifier behaviors depending on the contents of the subsystem:

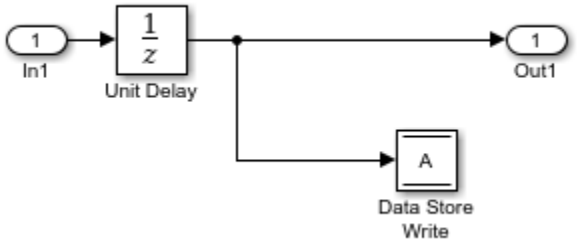
- The subsystem to be extracted contains Data Store Read blocks, but does not contain any Data Store Write blocks.



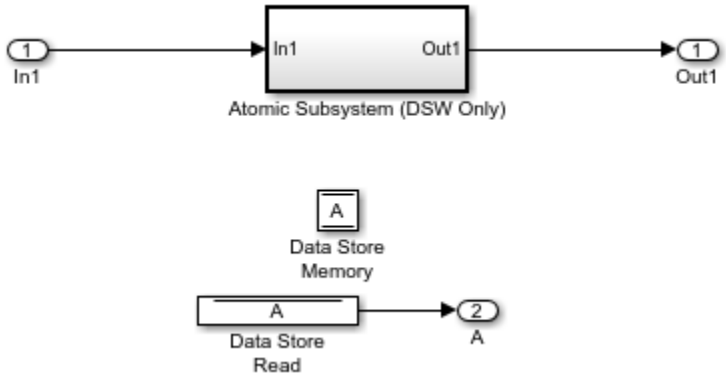
  Simulink Design Verifier inserts corresponding Data Store Write blocks with inports into the extracted model.



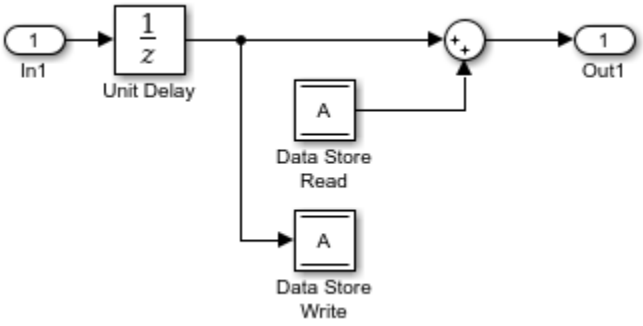
- The subsystem to be extracted contains Data Store Write blocks, but does not contain any Data Store Read blocks.



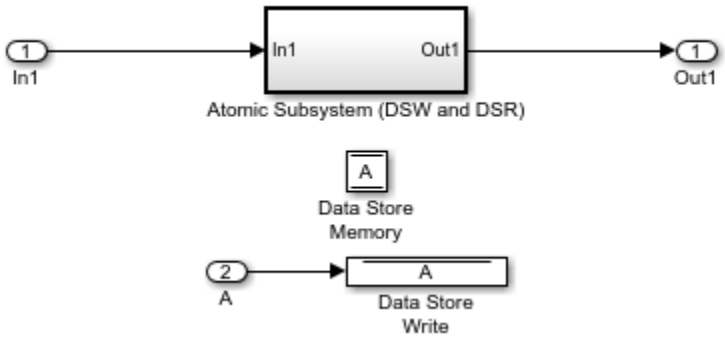
Simulink Design Verifier inserts corresponding Data Store Read blocks with outports into the extracted model.



- The subsystem to be extracted contains both Data Store Write and Data Store Read blocks.



Simulink Design Verifier inserts corresponding Data Store Write blocks with inports into the extracted model.



Interpret verify() statements as proof objectives for Simulink Design Verifier analysis

If your model or test harness contains a Simulink Test `verify()` statement in a Test Assessment or Test Sequence block, Simulink Design Verifier property proving analysis interprets the `verify()` statement as a proof objective. This allows your `verify()` statements to be used for both functional testing and formal analysis, without having to add Proof Objective blocks to the model. Also, for `verify()` statements falsified, you can create counterexamples that falsify the objective during simulation. For more information about property proving, see Prove Properties in a Model. For more information about `verify()` statements, see Assess Simulation Using Logical Statements.

Respect the value of the Simulink Verification and Validation CovLogicBlockShortCircuit model parameter during Simulink Design Verifier analysis

Simulink Design Verifier can consider logic blocks as short-circuiting during analysis, depending on the value you set for the Simulink Verification and Validation™ `CovLogicBlockShortCircuit` Model Parameters (Simulink). For more information, see Logic Operations Short-Circuiting.

Compatibility Considerations

Prior to R2017a, Simulink Design Verifier considered all Logical Operator blocks as non-short-circuiting. When the Simulink Verification and Validation `CovLogicBlockShortCircuit` model parameter is set to `'on'`, Simulink Design Verifier analysis may produce different results than those results from R2016b or earlier. To enforce the same Simulink Design Verifier logic block short-circuiting analysis behavior as that before R2017a, set `CovLogicBlockShortCircuit` to `'off'`.

R2016b

Version: 3.2

New Features

Bug Fixes

Model Slicer for Stateflow: Highlight active states and transitions for specified simulation time window

You can use Model Slicer to highlight active Stateflow states and transitions for a specified simulation time window. See Highlight Functional Dependencies and Using Model Slicer with Stateflow.

Model Slicer: Highlight unexpected behavior in test harnesses created by Simulink Test

You can use Model Slicer to highlight functional dependencies in test harnesses created by Simulink Test. See Highlight Functional Dependencies.

MCDC Test Generation: Generate tests from MCDC coverage for cascaded Simulink logic blocks

Use Simulink Design Verifier to generate tests to satisfy MCDC coverage objectives for models containing cascaded Simulink logic blocks if Treat Simulink logic blocks as short-circuited is enabled. See Analyzing MCDC for Cascaded Logic Blocks, Logical Operator Cascade Patterns, and Modified Condition and Decision Coverage in Simulink Design Verifier.

Incremental Test Generation: Generate tests to increase coverage for model objects in a test harness

Use Simulink Design Verifier to incrementally increase coverage for model objects in a test harness. See Increase Coverage for Referenced Models in a Test Harness.

Dead Logic Detection: Precisely detect dead logic without rational approximations

Improved precision of dead logic detection by removing approximation effects, such as relational boundary effects.

Detect Run-Time Errors in Supported S-Functions

When you run a design error detection analysis, Simulink Design Verifier supports the following subset of run-time error detection for supported S-Functions:

- Division by zero
- Out of bound array, including invalid pointer access
- Dead logic detection

For more information, see What Is Design Error Detection? and Support Limitations for S-Functions.

Generate Tests for Relational Boundary Code Coverage for Supported S-Functions

When you generate tests for models with compatible S-Functions, Simulink Design Verifier can create test cases for relational boundary objectives for the S-Functions. For more information, see Configuring S-Function for Test Case Generation and Relational Boundary.

Analyze Models that Contain Cell Arrays

Simulink Design Verifier supports analysis of models that contain cell arrays.

Detect Dead Logic Only without Floating-Point to Rational Number Conversion Approximation or While Loop Approximation

When you detect dead logic only, Simulink Design Verifier analyzes your model without floating-point to rational number conversion approximation or while loop approximation. For more information about the types of design error detection for dead logic in Simulink Design Verifier, see Dead Logic Detection. For more information about approximations in Simulink Design Verifier, see Approximations.

R2016a

Version: 3.1

New Features

Bug Fixes

Test Generation: Automatically generate tests for C/C++ S-Functions

Simulink Design Verifier generates tests for Decision, Condition, and MCDC objectives for C/C++ code within S-Functions. For more information, see:

- Configuring S-Function for Test Case Generation
- Workflow for Test Case Generation

Model Slicer: Time window adjustment without the need to rerun simulations

When you Refine Highlighted Model with Model Slicer, you can refine the slice highlight time window without the need to resimulate the model.

Variant Reducer: Create sliced models based on active variant configurations

You use the **Variant Manager** or Model Slicer to generate a simplified standalone model including only selected variant configurations. For more information, see Simplification of Variant Systems.

Overflow Detection: Automatically find overflow errors for fixed-point types with nonstandard word length

Design error detection in Simulink Design Verifier supports fixed-point types with nonstandard word length. For more information, see Detect Integer Overflow and Division-by-Zero Errors.

Simulink Functions: Perform verification of models that contain Simulink Functions

Simulink Design Verifier supports Simulink Functions.

Report Generation: Generate analysis reports in PDF format

Simulink Design Verifier can generate a PDF report that contains detailed information about the analysis results if you have Simulink Report Generator™. For more information, see Simulink Design Verifier Reports.

Model Slicer: Slice by using nonzero start time for simulation time window

When you Refine Highlighted Model with Model Slicer, you can use a nonzero start time for the simulation time window.

Model Slicer: Support for root-level inports as starting points for downstream highlighting

Model Slicer supports root-level inports as starting points for downstream highlighting. Model Slicer now treats these root-level inports as nonvirtual blocks. For more information, see Highlight Functional Dependencies.

R2015aSP1

Version: 2.8.1

Bug Fixes

R2015b

Version: 3.0

New Features

Bug Fixes

Analysis of C S-functions

Simulink Design Verifier provides the option to incorporate C S-function behavior for all modes of analysis.

Model Slicer API

Simulink Design Verifier adds an API for programmatically analyzing functional dependencies using Model Slicer.

Analyze minimum and maximum ranges specified for bus elements

Simulink Design Verifier analysis considers minimum and maximum values specified for bus elements in Simulink models. Minimum and maximum values specified for bus elements in Stateflow charts are not considered. See “Specify Input Ranges for Bus Elements”.

Model Advisor checks for design error detection

The Model Advisor includes additional Simulink Design Verifier design error detection checks:

- Detect integer overflow
- Detect division by zero
- Detect out of bound array access
- Detect violation of minimum and maximum values

Test Generation Advisor improvements

Test Generation Advisor reduces batch component extraction time and adds the ability to save results, reload results, and continue using previous analysis results.

Generate test inputs and export them to test cases in Simulink Test

You can generate test inputs using Simulink Design Verifier analysis and export them to test cases in Simulink Test. This option appears in the Simulink Design Verifier results dialog box. You can also generate test cases from Simulink Design Verifier analysis results files. See “Export Test Cases to Simulink Test”.

Support for Discrete Filter Blocks and Discrete Transfer Fcn Blocks

Simulink Design Verifier now supports Discrete Filter blocks and Discrete Transfer Fcn blocks for all **Reset Mode** settings.

R2015a

Version: 2.8

New Features

Bug Fixes

Isolate important model content and reduce model complexity based on design interests with Model Slicer

Model Slicer facilitates determining model block diagram dependencies, and generating simplified models that maintain simulation behavior. You use Model Slicer to analyze your model beginning with one or more starting points and an analysis direction. You can exclude model items and signal paths, and consider simulation effects in your analysis.

Model Slicer highlights items that your starting points depend on, and/or the items that depend on your starting points. You can generate a standalone, simplified model from certain model highlights.

You can access the Model Slice Manager by selecting **Design Verifier > Model Slicer**. See Model Simplification with Dependency Analysis.

Load results from previous Test Generation Advisor analysis

Test Generation Advisor loads the previous model analysis results after closing the advisor window or the model.

“Parameter table” replaces “Parameter configuration table”

The table that you access in the Model Configuration Parameters dialog box, on the **Design Verifier > Parameters** pane, is now called “Parameter table.” Use this table to specify parameter value ranges or constraints. The function of the table has not changed.

R2014b

Test generation for relational boundary values

Simulink Design Verifier generates tests that satisfy a relational boundary objective.

The relational boundary objective applies to blocks that have an explicit or implicit relational operation. The objectives also apply to Stateflow transitions and MATLAB function blocks that contain a relational operation.

For these blocks, the tests check the relational operations with:

- Equal operand values. This part of relational boundary objective applies only if both operands are integers or fixed-point numbers.
- Operand values that differ by a certain tolerance. This part of relational boundary objective applies to all operands. For integer and fixed-point operands, the tolerance is fixed. For floating-point operands, you can either use a predefined tolerance or specify your own tolerance value.

For more information, see Design Verifier Pane: Test Generation.

Fast dead logic detection and Model Advisor check

Dead logic detection is optimized for faster performance. It is integrated with the Model Advisor.

In releases prior to R2014b and after R2012b, design error detection included a single analysis option to detect dead logic. This previous dead logic detection also reported active logic.

Design error detection for dead logic now comprises two analysis options:

- Detection of dead logic only. Does not report active logic or undecided objectives. Available in the Model Advisor and in the Configuration Parameters dialog box under **Design Verifier > Options > Design Error Detection**.
- Detection of active logic. Runs concurrently with dead logic detection. In rare cases, can also find additional dead logic. Available in the Configuration Parameters dialog box under **Design Verifier > Options > Design Error Detection**.

By default, detection of active logic is turned off. However, if you have run dead logic detection on a model using a previous release, when you load your model in the new release, both detection of dead logic and active logic are turned on. Therefore, you have the same results of dead logic detection as before.

For more information, see:

- Detect Dead Logic Only
- Detect Dead and Active Logic

Analysis for arrays of buses, For Each block, and For Each Subsystem block

Analysis supports arrays of buses.

Analysis supports For Each and For Each Subsystem blocks, with the following limitation:

- When For Each Subsystem contains one or more Simulink Design Verifier Test Condition, Test Objective, Proof Assumption, or Proof Objective blocks, this block is not supported.

Test Generation Advisor to guide component analysis

Test Generation Advisor runs preliminary analysis on your model. For each component, it shows recommended Simulink Design Verifier options for test generation analysis. Test Generation Advisor also reports the number and status of test objectives for each component.

For each component, Test Generation Advisor reports one of the following statuses:

- Analyzable, meaning that you can use Simulink Design Verifier to analyze this component.
- Complex, meaning that you can use Simulink Design Verifier to analyze this component, but the analysis can be time-consuming. It is possible that the component does not receive full coverage from generated tests. For more information, see Sources of Model Complexity.
- Incompatible, meaning that this component is incompatible with Simulink Design Verifier. For more information, see Check Model Compatibility.

To open Test Generation Advisor, from the Simulink Editor, select **Analysis > Design Verifier > Generate Tests > Advisor**.

Improved test generation performance for lookup tables and timers

Test generation analysis includes optimizations to improve performance for 1-D Lookup Table and 2-D Lookup Table blocks. These optimizations are in effect for floating-point lookup tables using linear interpolation.

R2014a

Version: 2.6

New Features

Bug Fixes

Parameter Configuration Table for constraint specification and management

You can use the Parameter Configuration Table to specify parameters to treat as variables during analysis. Previously, you specified parameter configurations only with a MATLAB code file.

Use the Parameter Configuration Table to:

- Autogenerate value ranges for parameters in your model.
- Enter your own value ranges for parameters in your model.
- Highlight objects in your model that have parameters configured to act as variables during analysis.
- Import and export parameter configurations from MATLAB code files.

For more information, see Define Constraint Values for Parameters and Store Parameter Constraints in MATLAB Code Files.

Compatibility check integrated with Model Advisor

You can check the compatibility of your model for Simulink Design Verifier analysis using the Model Advisor. See Check compatibility with Simulink Design Verifier.

For more information on using the Model Advisor, see Consult the Model Advisor.

Condition coverage test generation for Relational Operator blocks

Analysis supports test case generation for condition coverage for Relational Operator, Compare to Constant, and Compare to Zero blocks. For more information, see Model Objects That Receive Coverage in the Simulink Verification and Validation documentation.

For Each Subsystem block analysis

Analysis supports For Each and For Each Subsystem blocks, with the following limitations:

- When For Each Subsystem contains another For Each Subsystem, not supported.
- When For Each Subsystem contains one or more Simulink Design Verifier Test Condition, Test Objective, Proof Assumption, or Proof Objective blocks, not supported.

Parameter handling for Simulink data dictionary

Analysis supports models with parameters stored in data dictionaries. For more information, see What Is a Data Dictionary?.

Japanese language localization support

Simulink Design Verifier is available in Japanese for Japanese localized systems. For more information, see Internationalization.

R2013b

Highlighting of partial results in model during analysis for visualizing progress

Before Simulink Design Verifier analysis on your model is complete, you can view its progress. Seeing the analysis progress can help you identify components of your model that take a longer time to analyze than others.

During analysis, in the Simulink Design Verifier log window, click **Highlight** to view the analysis results. The Simulink Design Verifier Results window shows the elapsed time of the analysis. Highlighting appears on model objects for which the analysis has begun. For more information, see Highlighted Results on the Model.

Summarizing and highlighting of prior analysis results

You can load previously generated analysis results for a model. In the Simulink Editor, select **Analysis > Design Verifier > Results > Load** to choose the data file with the results of a previous analysis. The Simulink Design Verifier Results window opens with options that you can use to explore the results. For more information, see Load Previous Results.

You can also use the `sldvloadresults` and `sldvhighlight` functions to explore previous analysis results for a model.

Performance improvements for test generation with input constraints

For test case generation analysis, especially when using **Nonlinear Extended** test suite optimizations, there are improved heuristics for handling input constraints defined using the Test Condition block.

Analysis time information for objectives in results window, report, and data file

The Simulink Design Verifier Results window, generated analysis reports, and data files include information about time spent analyzing objectives.

Mac i64 support

Simulink Design Verifier supports the 64-bit Intel® Macintosh platform. For more information, see System Requirements.

Improved while loop bound detection

If your model contains a `while` loop, Simulink Design Verifier tries to detect a conservative constant bound that allows the `while` loop to exit. If the software cannot find a constant bound, it performs a `while` loop approximation. With this approximation, the analysis does not prove objectives to be valid or unsatisfiable and it does not prove dead logic. The generated analysis report notes this approximation.

The behavior of the `while` loop approximation is consistent in all modes of analysis, as described in the following table.

Analysis Mode	While Loop Approximation
Design Error Detection	Sets number of `while` loop iterations to 3. Does not report dead logic or valid objectives.
Test Case Generation	Sets number of `while` loop iterations to 3. Does not report unsatisfiable objectives.
Property Proving	Sets number of `while` loop iterations to 3. Does not report valid objectives.

Compatibility Considerations

In previous versions of Simulink Design Verifier, if the software did not detect a bound for a `while` loop, it terminated the analysis or performed an approximation, as described in the following table.

Analysis Mode	While Loop Approximation (pre-R2013b)
Design Error Detection	Set number of `while` loop iterations to 3.
Test Case Generation	Set number of `while` loop iterations to 3.
Property Proving	Terminated analysis.

This generated analysis report noted this approximation.

Internationalization support for Simulink Design Verifier Options pane

Japanese internationalization is supported for the Simulink Design Verifier **Options** pane. For more information, see Internationalization.

Additional status information for undecided objectives

Simulink Design Verifier reports objectives that the analysis was unable to decide because of nonlinearities or division by zero. The software also reports objectives that the analysis was unable to decide because of stubbing. Objectives that have status Undecided Due to Stubbing can include objectives that, in releases prior to R2013b, were marked as Satisfied – No Test Case or Falsified – No Counterexample. For more information, see Objectives Undecided.

Example of property proving using Truth Table

The Property Proving Using MATLAB Truth Table Block example shows how to verify safety properties using Simulink Design Verifier property proving analysis. The example model uses `sldv.prove` and `sldv.assume` functions in a Truth Table block that uses MATLAB as the action language.

To see this example, at the MATLAB command prompt, type:

```
sldvexSBRVerificationTruthTableExample
```

Continuous state-space block family not stubbable

The Continuous Simulink blocks State-Space, Transfer Fcn, and Zero-Pole are not supported and not stubbable for Simulink Design Verifier analysis. For information on blocks that are supported for Simulink Design Verifier analysis, see Simulink Block Support.

Compatibility Considerations

If you have a model that contains one or more continuous State-Space, Transfer Fcn, or Zero-Pole blocks, your model is incompatible with Simulink Design Verifier. Consider analyzing smaller portions of your model to work around this incompatibility, as described in Extract Subsystems for Analysis.

R2013a

Version: 2.4

New Features

Bug Fixes

Detection of out-of-bound array access design errors

Before you simulate a model, you can use design error detection analysis to find out of bound array access errors. To detect out of bound array access errors in your model, from the Simulink Editor, select **Analysis > Design Verifier > Options**. In the Configuration Parameters dialog box, on the **Design Verifier > Design Error Detection** pane, select **Out of bound array access**. On the **Design Verifier** pane, click **Detect Errors**.

For more information, see Detect Out of Bound Array Access Errors.

R2012b

Support for Discrete Filter Blocks and Discrete Transfer Fcn Blocks

Simulink Design Verifier now supports Discrete Filter blocks and Discrete Transfer Fcn blocks if their **Reset Mode** is set to `None`. Discrete Filter blocks and Discrete Transfer Fcn blocks with their **Reset Mode** set to any other option are not supported.

R2012a

Version: 2.2

New Features

Bug Fixes

Design Error Detection For Dead Logic

You can now use design error detection to find dead logic in your model. Simulink Design Verifier uses multiple analysis engines, including Polyspace® and Prover®, to identify the dead logic. Previously, to uncover dead logic in your model, you had to analyze the results of test case generation. To conduct design error detection analysis, in the Configuration Parameters dialog box, on the **Design Verifier > Design Error Detection** pane, select **Dead Logic**. See Detecting Dead Logic.

Filtering Model Objects From Model Coverage

Simulink Design Verifier now allows you to filter certain model objects from model coverage during test case generation. The objects that you specify to exclude are stored in an external file. In the Configuration Parameters dialog box, on the **Design Verifier > Test Generation** pane, select **Ignore objectives based on filter**. In the **Coverage filter file** field, enter the file name.

For more information about coverage filtering, see Excluding Model Objects From Coverage in the Simulink Verification and Validation documentation.

Improved Property Proving For Look-Up Tables

Simulink Design Verifier now automatically optimizes property proving for Look-Up Table blocks, improving performance.

R2011b+

Version: 2.1.1

New Features

Bug Fixes

sldvtimer Function Available For Generating Test Cases

In R2011b+, the `sldvtimer` function is available to identify, change, and display timer optimizations. When generating test cases, you can use the function to:

- Identify where you can apply timer optimization in your model.
- Determine if Simulink Design Verifier applied timer optimizations during test generation.
- Configure timer optimizations.

R2011b

Checking Specified Design Minimum and Maximum Values

If you have specified minimum and maximum values on your Simulink model, the Simulink Design Verifier software can analyze your model to make sure that the intermediate and output signals do not violate the specified values. If the analysis finds a violation, the software creates a test case that demonstrates the violation.

This check is a new option for the design error detection analysis mode. For detailed information, see Checking for Specified Intermediate Minimum and Maximum Signal Values.

Improved Support for Trigonometric Functions

For a test case generation analysis, the `CombinedObjectives (Nonlinear)` and `LargeModel (Nonlinear Extended)` test suite optimizations have improved support for trigonometric and nonlinear math functions.

Improved Support for Large Lookup Tables

All the Simulink Design Verifier analysis modes have improved support for large lookup tables.

Optimized Handling for Extending Existing Test Cases

Existing test cases are now represented in a more memory-efficient way internally in the test generation engine. This improvement makes it possible to extend significantly longer existing test cases with more time steps.

Support for Trigger and Enable Ports for Model Blocks

The Simulink Design Verifier software fully supports root-level Trigger and Enable ports for referenced models.

Changed Format for sldvruntest and sldvruncgvtest Output

The output format for `sldvruntest` and `sldvruncgvtest` has changed in R2011b. In R2011b, the output argument contains the following data for each test case executed in an array of `Simulink.SimulationOutput` objects.

Field	Description
`tout_sldvruncgvtest`	Simulation time
`xout_sldvruncgvtest`	State data
`yout_sldvruncgvtest`	Output signal data
`logsout_sldvruncgvtest`	Signal logging data for: • Signals that are connected to outports • Signals that are configured for logging on the model

Compatibility Considerations

If you have scripts that depend on the output from `sldvruntest` and `sldvruncgvtest`, you can temporarily specify the output format. Use the nonvisible field `outputFormat` in the `runOpts` structure that `sldvruntestopts` creates as follows:

```
runOpts = sldvruntestopts;
runOpts.outputFormat = 'TimeSeries';
sldvruntest(model_name, sldvData, runOpts);
```

or

```
runOpts = sldvruntestopts;
runOpts.outputFormat = 'StructureWithTime';
sldvruntest(model_name, sldvData, runOpts);
```

Conversion of Error and Warning Message Identifiers

For R2011b, error and warning message identifiers have changed in Simulink Design Verifier.

Compatibility Considerations

If you have scripts or functions that use message identifiers that changed, you must update the code to use the new identifiers. Typically, message identifiers are used to turn off specific warning messages, or in code that uses a `try`/`catch` statement and performs an action based on a specific error identifier.

For example, the `SLDV:InvalidConfSet` identifier has changed to `SLDV:configcomp_get:InvalidConfSetRef`. If your code checks for `SLDV:InvalidConfSet`, you must update it to check for `SLDV:configcomp_get:InvalidConfSetRef` instead.

To determine the identifier for a warning, run the following command just after you see the warning in the MATLAB command window:

```
[MSG,MSGID] = lastwarn;
```

This command saves the message identifier to the variable `MSGID`.

To determine the identifier for an error that appears at the MATLAB prompt, run the following commands just after you see the error:

```
exception = MException.last;
MSGID = exception.identifier;
```

Note Warning messages indicate a potential issue with your code. While you can turn off a warning, a suggested alternative is to change your code so it runs warning-free.

R2011a

Version: 2.0

New Features

Bug Fixes

Automatic Detection of Overflow and Divide-by-Zero Design Errors

If your Simulink model performs arithmetic operations, the Simulink Design Verifier software can analyze the model to identify design errors that occur at run time. The analysis detects two types of errors:

- Integer or fixed-point data overflow
- Division by zero

After the analysis is complete, you can create a harness model that includes test cases for each error.

For more information, see Detecting Integer Overflow and Division-by-Zero Errors.

Improved Analysis Results Workflow

After you analyze a Simulink model using the Simulink Design Verifier software, you can choose how you want to review the results. You have the following options:

- Highlight the analysis results on the model.
- Generate the detailed analysis report.
- Create the harness model with the generated test cases in the Signal Builder.
- Simulate the test cases and produce a model coverage report (test case generation only).

Improved Support for Nonlinear Arithmetic and Math Operations

The Simulink Design Verifier test-generation software includes improved support for nonlinear arithmetic and math operations that:

- Improves test generation for models containing nonlinear operations.
- Improves support of math operations such as trigonometric functions.
- Improves scalability to very large models.

To use these new strategies, in the Configuration Parameters dialog box, on the **Design Verifier > Test Generation** pane, set the **Test suite optimization** parameter to one of the following:

- `CombinedObjectives (Nonlinear Extended)`
- `LargeModel (Nonlinear Extended)`

New Capability to Highlight Analysis Results on the Model

When a Simulink Design Verifier analysis is complete, you can specify that the software use color highlighting on the model to indicate the analysis results for individual objects. When you click an object, you see the detailed results specific to that object.

You can generate a detailed analysis report or create a harness model that contains test case signals at any time.

After you run a design error detection analysis, the model is highlighted by default. After you run a test case generation or property-proving analysis, you can highlight the model.

For more information, see Highlighted Results on the Model.

New Capability to Review Model Analysis Results in Model Explorer

If you close the Simulink Design Verifier analysis results so you can fix the causes of problems in your model, you might need to review the analysis results again. As long as your model remains open, you can view the results of your most recent analysis results in the Model Explorer by selecting **Tools > Design Verifier > Latest Results**.

From the Model Explorer, you can:

- Highlight the analysis results on the model.
- Generate the detailed analysis report.
- Create the harness model with the generated test cases in the Signal Builder.
- Simulate the test cases and produce a model coverage report (test case generation only).

For more information, see Reviewing Analysis Results in the Model Explorer.

New Temporal Operator Blocks

The Simulink Design Verifier block library includes three new blocks that allow you to define temporal properties on Boolean signals in your model:

- Detector — Detect true duration on input and construct output true duration based on output type
- Extender — Extend true duration of input
- Within Implies — Capture within implication if observed input is `true` within each true duration of first input

Support for Simulink Blocks

The Simulink Design Verifier software now supports the following Simulink blocks:

- Discrete Derivative
- Function-Call Feedback Latch (new block)
- Probe
- Rate Limiter Dynamic
- Trigonometric Function — Supported when **Function** is `sin`, `cos`, or `sincos` and **Approximation method** is `CORDIC`.
- Variant Subsystem
- Weighted Sample Time
- Weighted Sample Time Math

R2010bSP1

Version: 1.7.1

Bug Fixes

R2010b

Support for 64-Bit Windows Operating Systems

The Simulink Design Verifier software can now execute in 64-bit mode on 64-bit Windows® systems.

New Support for Specified Input Minimum and Maximum Values as Analysis Constraints

The Simulink Design Verifier software can consider the specified minimum and maximum values for input signals during a test case generation or property proving analysis. This feature allows you to:

- Constrain the test cases based on the specified minimum and maximum values.
- Assume the specified minimum and maximum values during property-proving analysis.

New Built-In Support for Automating Test Execution in SIL/PIL Mode via the CGV API

The new `sldvruncgvtest` function allows you to execute test cases in Software-in-the-Loop (SIL) or Processor-in-the-Loop (PIL) mode using the Code Generation Verification (CGV) API methods.

New Support for Extracting and Analyzing Stateflow Atomic Subcharts

The Simulink Design Verifier software now provides support for Stateflow atomic subcharts. Atomic subcharts make it easier to isolate parts of a Stateflow chart for development and analysis.

The Simulink Design Verifier support for atomic subcharts allows you to:

- Generate test cases and prove properties for an atomic subchart in a Stateflow chart within a Simulink model.
- Use `sldvextract` to extract the contents of an atomic subchart and create a model for the Simulink Design Verifier software to analyze.

New Capability to Eliminate Unused Signals from the Generated Harness

To improve the performance of harness model creation during a Simulink Design Verifier analysis, from the Signal Builder block, the software can omit signals that have no effect on the output of the model.

`sldvmakeharness` supports this capability if you set the `usedSignalsOnly` harness model option to `true`.

Support for Simulink Blocks

The Simulink Design Verifier software now supports the following Simulink blocks:

- Dead Zone and Dead Zone Dynamic
- Lookup Table Dynamic
- Probe (Partial support)

- Width

sldvlogsignals Replaces sldvlogdata

The `sldvlogsignals` function replaces the `sldvlogdata` function. Use `sldvlogsignals` to:

- Simulate a Simulink model and in that model, log all the inputs to a specified Model block.
- Simulate all or some of the test cases in a harness model created by:
  - Simulink Design Verifier analysis
  - `sldvmakeharness`
  - `slvnvmakeharness`

Compatibility Considerations

The `sldvlogsignals` function replaces the `sldvlogdata` function. Currently, if you use the `sldvlogdata` function, it automatically redirects to `sldvlogsignals`. Update your scripts to use `sldvlogsignals`.

sldvmergeharness Replaces sldvharnessmerge

The `sldvmergeharness` function replaces the `sldvharnessmerge` function. `sldvmergeharness` combines the test cases and initializations from any specified harness models into a single harness model.

Compatibility Considerations

Currently, if you use the `sldvharnessmerge` function, it automatically redirects to `sldvmergeharness`. Update your scripts to use `sldvmergeharness`.

R2010a

Version: 1.6

New Features

Bug Fixes

Generate Test Cases for Missing Coverage Data

The Simulink Design Verifier software now offers the option to isolate test generation to objectives that are not satisfied in simulation coverage results. If you simulate your model, but do not achieve 100% coverage, you can analyze the model using the Simulink Design Verifier test-generation capability to find test cases that achieve the missing coverage.

If you select the **Ignore objectives satisfied in existing coverage data** parameter in the Configuration Parameters dialog box, you can import the coverage data file; the analysis eliminates all objectives satisfied in the coverage results.

sldvlogdata Function for Logging Test Cases During Simulation

With the new `sldvlogdata`function, you can:

- Simulate a Simulink model and in that model, log all the inputs to a Model block.
- Simulate all or some of the test cases in a harness model created by the Simulink Design Verifier software and log all the inputs to the test unit.

You can save logged data to a MAT-file and use that file as input to the Simulink Design Verifier software for extending tests. This allows you to generate more realistic test cases and extends the analysis to complete the test suite.

Extend Existing Test Cases

The Simulink Design Verifier software now offers the option to extend existing test cases with additional time steps to generate complete test suites. This allows the software to generate test cases for parts of your model that are hard to analyze.

If you enable the **Extend existing test cases** parameter in the Configuration Parameters dialog box, the software imports the logged test cases from a MAT-file. If you also enable the **Ignore objectives satisfied by existing test cases** parameter, the analysis generates results, ignoring the coverage objectives satisfied by the logged test cases. Otherwise, the analysis efficiently creates a complete test suite.

Demo Library and Models to Support Temporal Properties Specification

The Simulink Design Verifier software includes a new **Temporal Property Specification** demo category that includes:

- A Temporal Operator Blocks demo library that contains the following blocks and examples:
  - Detector — Detects a user-specified length of true duration on the input signal and constructs an output true duration of length based on the output type.
  - Extender — Extends the true duration of the input signal by a fixed number of time steps or indefinitely.
  - Within Implies — Captures the within implication by observing whether the second input is true for at least one time step within each true duration of the first input.
  - Temporal Property Specification examples — A library model that includes examples that use the Detector, Extender, and Within Implies blocks

- Two demo models that contain these blocks:
  - Debounce Temporal Properties
  - Power Window Controller Temporal Properties

Support for Stateflow Absolute-Time Temporal Logic Operators

The Simulink Design Verifier software now supports the Stateflow absolute-time temporal logic operators. For more information, see Operators for Absolute-Time Temporal Logic.

Support for Simulink Blocks

The Simulink Design Verifier software now fully supports the following blocks:

- Backlash
- Cosine
- Discrete Derivative
- Sine

The Simulink Design Verifier software now provides improved support for the following blocks:

- Interpolation Using Prelookup
- Lookup Table (n-D)

For more information, see Simulink Block Support.

R2009bSP1

Version: 1.5.1

Bug Fixes

R2009b

New Functions for Verification Objectives and Constraints

Use these four new functions to specify objectives and constraints within an Embedded MATLAB® script. You can use these functions instead of the corresponding Simulink Design Verifier blocks.

Function	Purpose	Corresponding Block
`sldv.assume`	Proof assumption	Proof Assumption
`sldv.condition`	Test condition	Test Condition
`sldv.prove`	Proof objective	Proof Objective
`sldv.test`	Test objective	Test Objective

These functions:

- Identify mathematical relationships for objectives and constraints in a form that can be more natural than using block parameters
- Support specifying multiple constraints without complicating the model
- Provide access to the power of the MATLAB software
- Support separation of verification and model design

Compatibility Considerations

The following functions will be removed in a future release:

- `dv.assume`
- `dv.condition`
- `dv.prove`
- `dv.test`

So that your models with those functions will work in future releases, replace these functions with the corresponding new function added in this release. For example, replace `dv.assume` with `sldv.assume`.

Support for Enumerated Signals and Parameters

The Simulink Design Verifier software now supports Simulink models with enumerations. All the Simulink Design Verifier library blocks support enumerated parameters, constants, and inputs.

New Option to Stop Simulation on Proof Violation

The Simulink Design Verifier software allows you to stop a model simulation if it encounters a property violation. You enable this capability by inserting a Proof Objective block into a model and setting the **Stop simulation when the property is violated** parameter. If the simulation detects a violation of the property specified in the Proof Objective block, it terminates with an error.

Therefore, you can now verify a counterexample that was detected during a Simulink Design Verifier analysis.

New sldvmakeharness Function

With the new `sldvmakeharness` function, you can:

- Create a test harness model from existing Simulink Design Verifier analysis data.
- Create an empty test harness model directly from a Simulink model.

New sldvreport Function

You can now generate and customize a report from existing Simulink Design Verifier analysis data with the new `sldvreport` function.

New Support for Simulink Blocks

The Simulink Design Verifier software now supports the following blocks and parameters:

- Direct Lookup Table (n-D)
- Discrete Transfer Fcn
- Lookup Table and Lookup Table (2-D) — Except when the **Lookup method** block parameter specifies `Interpolation-Extrapolation` and the block's input and output signals do not have the same floating-point data type.
- Math Function — All signal types now support the `hermitian` and `transpose` function parameter settings
- Rate Limiter — For signals of all data types
- Shift Arithmetic — For all parameters and signals of all data types
- Tapped Delay
- Transfer Fcn Direct Form II
- Transfer Fcn Direct Form II Time Varying

Support for New Blocks

The Simulink Design Verifier software supports the following new Simulink blocks:

- Discrete PID Controller
- Discrete PID Controller (2 DOF)
- Enumerated Constant

R2009a

Version: 1.4

New Features

Bug Fixes

Automatic Stubbing for Unsupported Operations

Automatic stubbing allows you to complete a test-generation or property-proving analysis even if the model contains blocks or functions that the Simulink Design Verifier software does not support, like S-functions and C math operations.

By default, this feature is unavailable. To enable automatic stubbing before running an analysis, on the Configuration Parameters **Design Verifier** main pane, select **Automatic stubbing of unsupported blocks and functions**. In addition, if the compatibility check finds unsupported blocks that automatic stubbing can handle, you can enable automatic stubbing at that time.

Long Test Case Optimization

`Long test cases` is a new option for the **Test suite optimization** parameter. The `Long test cases` option instructs the Simulink Design Verifier software to create fewer but longer test cases that each satisfy multiple test objectives. With this option, you can customize the analysis results, run a more efficient analysis, and create easier-to-review results, in both Signal Builder and in the HTML report that the software generates.

New Support for Blocks

The Simulink Design Verifier software now supports models containing the following blocks:

- Combinatorial Logic
- Decrement Time To Zero
- Discrete Filter
- Fixed-Point State-Space
- Integer Delay
- Model blocks that reference other models
- Prelookup
- Relay

Analyzing External Functions for Embedded MATLAB Function Blocks

If your model contains an Embedded MATLAB Function block that calls any external functions, the Simulink Design Verifier software can now accumulate coverage results for those functions.

Enhanced Block Replacement Capability for Subsystems and Model Blocks

You can write your own replacement rules to replace subsystem or Model blocks that reference another model with the Simulink Design Verifier block replacement capability. The software replaces a subsystem or Model block with a different subsystem or with a built-in block as defined in the block replacement rules.

New Implies Block

The new Implies block simplifies property specification. You can now specify conditions that produce a given response. For example, you can quickly create expressions indicating that pressing the brake pedal implies the cruise control must be inactive.

You can use the Implies block in any model, not just when running the Simulink Design Verifier software.

New Property-Proving Examples and Demos

The Simulink Design Verifier block library includes four new example models that demonstrate how to define complex properties for property-proving analysis.

In addition, the following demo models are shipping with R2009a:

- `sldvdemo_sbr_design.mdl` — Finding property violations
- `sldvdemo_sbr_verification.mdl` — Proving that properties are valid
- `sldvdemo_thrustrvs_verification.mdl` — Analyzing model and properties to prove correctness or to identify counterexamples
- `sldvdemo_cruise_control_fxp_verification.mdl` — Proving properties for fixed-point arithmetic with block replacements
- `sldvdemo_cruise_control_verification.mdl` — Supporting model reference and verification subsystems

sldvisactive Function

The `sldvisactive` function checks whether the Simulink Design Verifier software is actively translating the model. This function is called from the masked initialization of masked subsystems and other model or block callbacks to configure the model for Simulink Design Verifier analysis.

For example, the mask initialization of the Environment Controller block invokes the `sldvisactive` function to output the signal at its Sim port when you start analyzing a model that contains the block.

R2008b

Simulink Bus Signals and Bus Objects Support

Simulink Design Verifier now supports Simulink buses and bus objects:

- The root Inport and Outport blocks accept bus signals.
- Nonvirtual buses are propagated through the model elements.
- The test harness model reconstructs the bus signals from the underlying bus elements.

Fixed-Point Data Support

Simulink Design Verifier blocks now support fixed-point parameters and inputs. These blocks include:

- Test Condition
- Test Objective
- Assumption
- Proof Objective

The `Slvd.Point` and `Sldv.Interval` constructors now accept fixed-point data.

Generating Test Harness Model with Model Reference

To use this option, select **Reference input model in generated harness** in the **Design Verifier > Results** pane of the Configuration Parameters dialog box. Simulink Design Verifier software then uses model reference to run the original model from the test harness.

Generating SystemTest TEST-File

To use this option, select **Save test harness as SystemTest TEST-File** in the **Design Verifier > Results** pane of the Configuration Parameters dialog box. The software creates a TEST-file instead of a test harness model. Using a TEST-file allows you to run the test cases in the SystemTest™ environment and configure the model coverage settings using the SystemTest software.

Improved Search Algorithms

This release includes search algorithms for the following two modes that improve the performance and the quality of the results:

- Test case generation — The combined objectives options minimizes the number of test cases by generating cases that address more than one test objective.
- Property proving — Proving that model properties are valid.

New Data File Format

When the Simulink Design Verifier software completes an analysis, it creates a data file. Now the data file supports bus input ports and includes more information about the analyzed model. For more information, see Simulink Design Verifier Data Files.

Compatibility Considerations

To convert an `sldvData` structure from the old format to the new format, use the `Sldv.DataUtils.convertToCurrentFormat` utility with the following syntax:

```
new_sldvData = Sldv.DataUtils.convertToCurrentFormat(model, old_sldvData)
```

The arguments used for this conversion comprise:

- `model` — The name of the model that was analyzed
- `old_sldvData` — The name of an `sldvData` structure created using the old (pre-R2008b) format

To convert an `sldvData` structure in the new format to the old format, use the `Sldv.DataUtils.convertToOldFormat` utility with the following syntax:

```
old_sldvData = Sldv.DataUtils.convertToOldFormat(model, new_sldvData)
```

The arguments used for this conversion comprise:

- `model` — The name of the model that was analyzed
- `new_sldvData` — The name of an `sldvData` structure created using the format that is new with R2008b

New HTML Report

The HTML report that Simulink Design Verifier software generates has been enhanced. Now, when you select **Generate report of the results** in the **Design Verifier > Report** pane of the Configuration Parameters dialog box, the report generated has several improvements:

- The report generates faster and is easier to understand.
- The report can display expected outputs.
- The software generates a report that reflect the analysis settings (for example, test case generation vs. property proving).

Blocks with No Input Ports Limitation

If a Simulink model has any blocks with no input ports, Simulink Design Verifier software cannot generate the test harness.

R2008a+

Version: 1.2.1

Bug Fixes

R2008a

Version: 1.2

New Features

Bug Fixes

Embedded MATLAB Subset Support

This release provides support for the Embedded MATLAB Function block in the Simulink software and Embedded MATLAB functions in the Stateflow software. For more information, see Support Limitations for MATLAB for Code Generation in the Simulink Design Verifier User's Guide.

Enhanced Support for Stateflow Truth Tables

Previous releases support only the Stateflow Classic truth tables. However, this release introduces support for Embedded MATLAB truth tables in the Stateflow software, which includes support for the Truth Table block. See Truth Table Functions for Decision-Making Logic for more information.

New Simulink Design Verifier Data File Options

This release introduces new options on the **Design Verifier > Results** pane of the Configuration Parameters dialog box:

- **Include expected output values** — Simulates the model using the test case signals and includes the output values in the Simulink Design Verifier data file.
- **Randomize data that does not affect outcome** — Assigns random values instead of zeros to input signals that have no impact on test or proof objectives.

New Test Suite Optimization Setting

In this release, the **Test suite optimization** parameter that appears on the **Design Verifier > Test Generation** pane of the Configuration Parameters dialog box includes a new setting: `Large model`. This test generation strategy is tailored to large, complex models that contain nonlinearities and many test objectives.

R2007b+

Version: 1.1.1

Bug Fixes

R2007b

Version: 1.1

New Features

Bug Fixes

Fixed-Point Data Type Support

This release provides support for fixed-point data types. For more information, see Fixed-Point Support Limitations in the Simulink Design Verifier User's Guide.

R2007a+

Version: 1.0

New Features

Introducing the Simulink Design Verifier Software

The Simulink Design Verifier software extends the Simulink and Stateflow products with formal methods that help you confirm your model and chart behavior. The Simulink Design Verifier software performs a mathematically rigorous analysis of your model to identify all of its possible execution pathways. Subsequently, the software can

- **Generate Tests**

  The Simulink Design Verifier software can generate tests that satisfy your model's coverage objectives, including decision coverage, condition coverage, and modified condition/decision coverage (MCDC). You can even customize the tests that it generates by using Simulink Design Verifier blocks that allow you to specify your own objectives and to constrain signal values. After the software completes its analysis, it produces a test harness model with a Signal Builder block that contains test signals. Simply simulate the test harness model to confirm that the test signals achieve your model's objectives.
- **Prove Properties**

  The Simulink Design Verifier software can prove that signals in your model attain particular values or ranges. Use Simulink Design Verifier blocks to specify values and ranges that you desire signals to attain, or to constrain the values of other signals. If the software disproves any of the values or ranges given the constraints you specify, it produces a test harness model with a Signal Builder block that contains signals comprising counterexamples. Simply simulate the test harness model to confirm that the counterexamples falsify your model's properties.

The Simulink Design Verifier software documents its analysis results in an HTML report. Also, it produces a data file containing the analysis results, which you can postprocess for your own analyses and reports.

In short, the Simulink Design Verifier software gives you confidence in the behavior of your Simulink models and Stateflow charts.

Version 1.0 of the Simulink Design Verifier software was released in a Web-downloadable form after R2007a.