Simulink® HDL Coder Release Notes

Contents

Summary by Version

This table provides quick access to what's new in each version. For clarification, see "About Release Notes" on page 1.

Version (Release)	New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Latest Version V1.2 (R2007b)	Yes Details	Yes Summary	Bug Reports	Printable Release Notes: PDF Current product documentation
V1.1 (R2007a)	Yes Details	No	Bug Reports	No
V1.0 (R2006b)	Yes Details	Not applicable	Bug Reports	No

About Release Notes

Use release notes when upgrading to a newer version to learn about new features and changes, and the potential impact on your existing files and practices. Release notes are also beneficial if you use or support multiple versions.

If you are not upgrading from the most recent previous version, review release notes for all interim versions, not just for the version you are installing. For example, when upgrading from V1.0 to V1.2, review the New Features and Changes, Version Compatibility Considerations, and Bug Reports for V1.1 and V1.2.

New Features and Changes

These include

- New functionality

- Changes to existing functionality
- Any version compatibility considerations associated with each new feature or change

Version Compatibility Considerations

When a new feature or change introduces a reported incompatibility between versions, its description includes a **Compatibility Considerations** subsection that details the impact. For a list of all new features and changes that have reported compatibility impact, see the "Compatibility Summary for Simulink HDL Coder" on page 18.

Compatibility issues that are reported after the product has been released are added to Bug Reports at the MathWorks Web site. Because bug fixes can sometimes result in incompatibilities, also review fixed bugs in Bug Reports for any compatibility impact.

Fixed Bugs and Known Problems

MathWorks Bug Reports is a user-searchable database of known problems, workarounds, and fixes. The MathWorks updates the Bug Reports database as new problems and resolutions become known, so check it as needed for the latest information.

Access Bug Reports at the MathWorks Web site using your MathWorks Account. If you are not logged in to your MathWorks Account when you link to Bug Reports, you are prompted to log in or create an account. You then can view bug fixes and known problems for R14SP2 and more recent releases.

The Bug Reports database was introduced for R14SP2 and does not include information for prior releases. You can access a list of bug fixes made in prior versions via the links in the summary table.

Related Documentation at Web Site

Printable Release Notes (PDF). You can print release notes from the PDF version, located at the MathWorks Web site. The PDF version does not support links to other documents or to the Web site, such as to Bug Reports. Use the browser-based version of release notes for access to all information.

Product Documentation. At the MathWorks Web site, you can access complete product documentation for the current version and some previous versions, as noted in the summary table.

Version 1.2 (R2007b) Simulink HDL Coder

This table summarizes what's new in V1.2 (R2007b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes Summary	Bug Reports	Printable Release Notes: PDF Current product documentation

New features and changes introduced in this version are:

HDL Code Generation for Single-Clock Multirate Models

Simulink® HDL Coder now supports HDL code generation for single-clock, single-tasking multirate models. Your model can include blocks running at multiple sample rates:

- Within in the device under test (DUT)
- In the test bench driving the DUT

- In both the test bench and the DUT

Multirate code generation support is described in detail in "Generating HDL Code for Multirate Models" in the Simulink HDL Coder documentation.

Additional Blocks Supported for Multirate Code Generation

The following blocks, frequently used in construction of multirate models, are now supported for HDL code generation:

- Signal Attributes/Rate Transition
- Signal Processing Blockset/Signal Operations/Downsample
- Signal Processing Blockset/Signal Operations/Upsample

New Properties Added in Support of Multirate Code Generation

To support multirate code generation, two new `makehdl` properties have been added. The new properties are:

- `HoldInputDataBetweenSamples`: This property determines how long (in terms of base rate clock cycles) data values for subrate signals are held in a valid state. See `HoldInputDataBetweenSamples` for details.
- `ClockEnableDelay`: This property specifies a delay time `D` (a number of base-rate clock cycles) applied to `clk_enable` signals when the model's reset input signal is asserted. This delay time is in addition to the hold time specified by the `HoldTime` property. See `ClockEnableDelay` for details.

Requirements and Restrictions for Multirate Code Generation

Certain requirements and restrictions apply to the use of multirate models for HDL code generation. See "Configuring Multirate Models for HDL Code Generation" for further information.

Additional Blocks Supported for HDL Code Generation

Simulink HDL Coder now supports the following blocks for HDL code generation:

- Additional Math & Discrete/Additional Discrete/Unit Delay Enabled
- Math Operations/Divide
- Math Operations/Math Function (`sqrt` function only)
- Signal Attributes/Rate Transition
- Signal Processing Blockset/Signal Operations/Downsample
- Signal Processing Blockset/Signal Operations/Upsample
- Dual Port RAM (For information on this new block, see also "Dual Port RAM Block Supported for Simulation and Code Generation" on page 6.)

See "Summary of Block Implementations" for a complete listing of blocks that are currently supported for HDL code generation.

Dual Port RAM Block Supported for Simulation and Code Generation

Simulink HDL Coder 1.2 provides the Dual Port RAM Block for use in simulation and code generation.

The Dual Port RAM block lets you:

- Simulate the behavior of a dual-port RAM with registered outputs in your Simulink model.
- Generate an interface to the inputs and outputs of the RAM in HDL code.

See "Generating an Interface for the Dual Port RAM Block" for full details.

Block Implementation Parameters Include Output Pipelining

Simulink HDL Coder 1.2 supports *block implementation parameters*, which let you control details of the code generated for specific block implementations. Block implementation parameters are passed as property/value pairs to `forEach` or `forAll` calls in a code generation control file.

Supported Block Implementation Parameters

Block implementation parameters supported in the current release include:

- `'OutputPipeline', nStages`: This parameter lets you specify a pipelined implementation for selected blocks. The parameter value (`nStages`) specifies the number of pipeline stages (pipeline depth) in the generated code. `OutputPipeline` is supported by most Simulink HDL Coder block implementations.
- Interface generation parameters let you customize features of an interface generated for the following block types:
  - simulink/Ports & Subsystems/Model
  - built-in/Subsystem
  - lfilinklib/HDL Cosimulation
  - modelsimlib/HDL Cosimulation

  For example, you can specify generation of a black box interface for a subsystem, and pass in parameters that specify the generation and naming of clock, reset, and other ports in HDL code. Interface generation parameters are described in "Customizing the Generated Interface".

For more information on block implementation parameters, see the following sections in the Simulink HDL Coder documentation:

- "Specifying Block Implementations and Parameters in the Control File"
- "Block Implementation Parameters"
- "Summary of Block Implementations"

Using hdlnewforeach to Find Block Implementation Parameters

Given a selection of one or more blocks from your model, the `hdlnewforeach` function returns information about the available HDL implementations for each block.

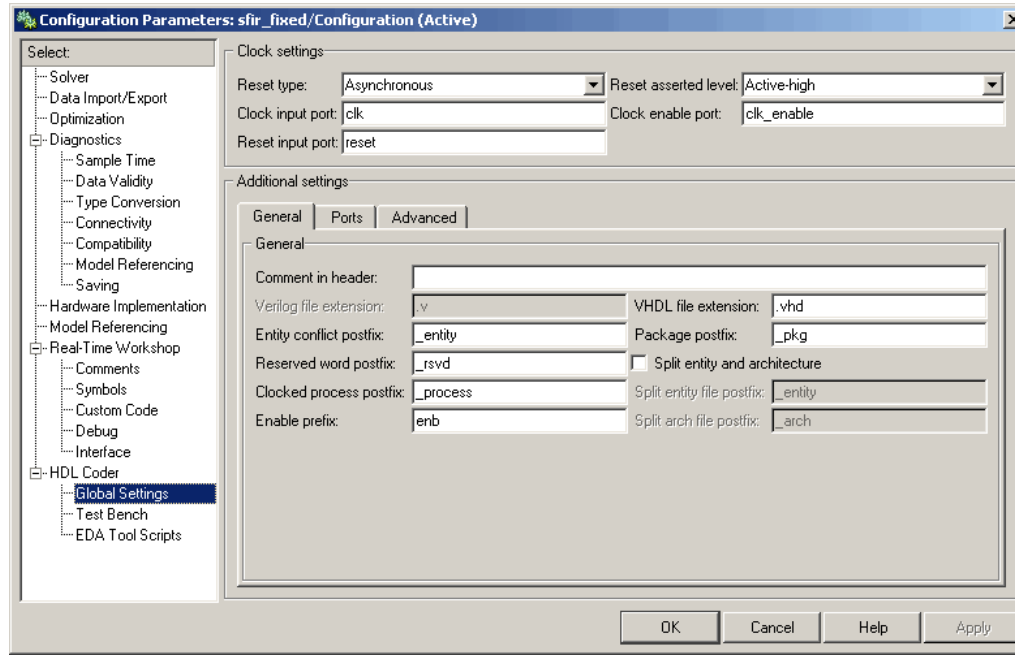
In the current release, the information returned by `hdlnewforeach` has been expanded. `hdlnewforeach` now returns an optional cell array of strings specifying the parameter(s) corresponding to each block implementation.

See “Generating Selection/Action Statements with the hdlnewforeach Function” for details.

Summary of GUI Updates

The following updates have been made to the Simulink HDL Coder GUI:

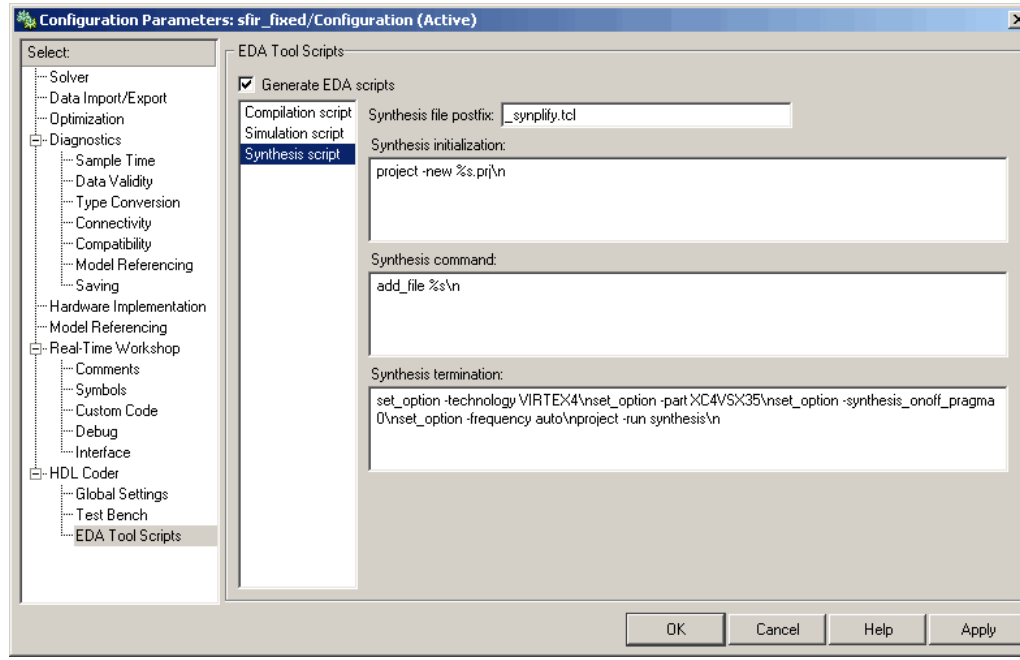
- The **Enable prefix** option is now supported by the GUI as well as by the `EnablePrefix` command-line property. See “Enable prefix” for details on this option.



- The default value for the **Synthesis termination** field of the EDA Tool Scripts dialog box has changed, as shown in the following figure. The default hardware target string in generated synthesis scripts now specifies
  - `technology` option: `VIRTEX4`

    In previous releases, this option defaulted to `VIRTEX2`.
  - `part` option: `XC4VSX35`

In previous releases, this option defaulted to `XC2V500`.



See also “Default Hardware Target for Synthesis Scripts Updated to Virtex-4 ” on page 10.

Digital Filter Block Restriction Removed

In previous releases, Filter Design HDL Coder was required to generate HDL code for the Digital Filter block when the **Dialog parameters** option was selected in the **Coefficient source** option group. This requirement has been removed.

In the current release, the HDL code generation requirements for the Digital Filter block vary according to the **Coefficient source** option you select, as follows:

- **Dialog parameters**: No additional toolboxes or blocksets required for HDL code generation.
- **Discrete-time filter object**: Filter Design HDL Coder required.

- **Input port(s)**: This option is not supported for HDL code generation.

See also “Summary of Block Implementations”.

Support for New Embedded MATLAB Bitwise Functions

Simulink HDL Coder supports the new Embedded MATLAB™ fixed-point bitwise functions introduced in R2007b. Many of these functions map directly to HDL bitwise operators, resulting in very efficient HDL code. See “Using Fixed Point Bitwise Functions” for examples of the use of these functions in HDL code generation.

For general information on Embedded MATLAB bitwise functions, see “Bitwise Functions” in the Fixed-Point Toolbox documentation.

Compatibility Considerations

In previous releases, the return type of the `bitget` function was `ufix8`. For more efficient HDL code generation, the return data type of the `bitget` function has been changed to `ufix1`. If your existing Embedded MATLAB code performs type casts to adapt values returned from `bitget` for HDL code generation, you may be able to eliminate these type casts.

Default Hardware Target for Synthesis Scripts Updated to Virtex-4

The default hardware target string in generated synthesis scripts now specifies

- `technology` option: `VIRTEX4`

  In previous releases, this option defaulted to `VIRTEX2`.

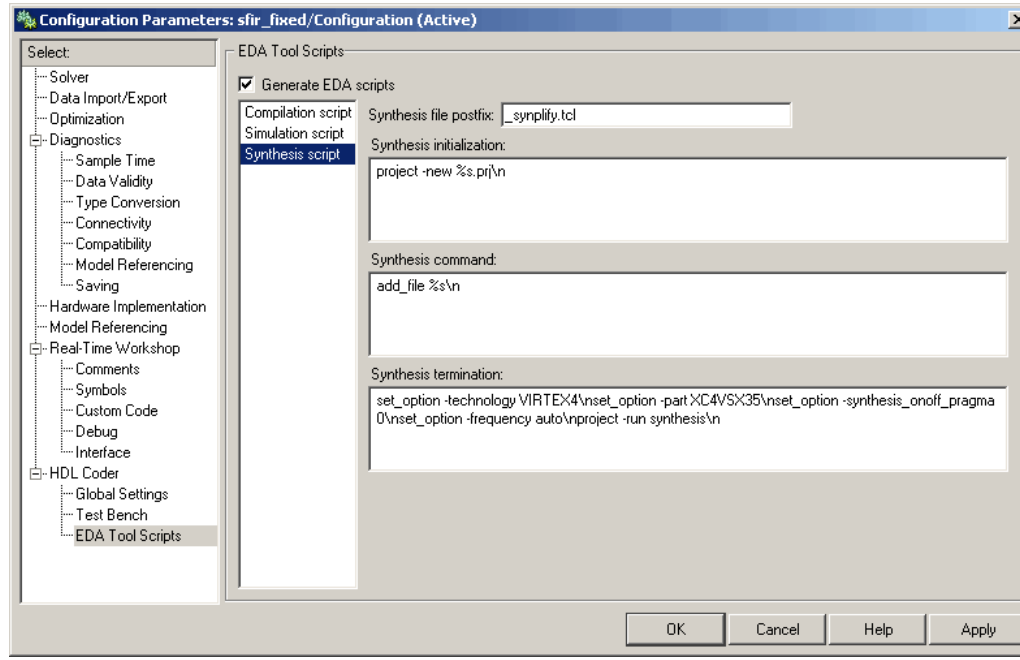
- `part` option: `XC4VSX35`

  In previous releases, this option defaulted to `XC2V500`.

These updates affect the default value for the `HDLSynthTerm` property. The default is:

```
['set_option -technology VIRTEX4\n',...
'set_option -part XC4VSX35\n',...
'set_option -synthesis_onoff_pragma 0\n',...
'set_option -frequency auto\n',...
'project -run synthesis\n']
```

The default value for the `HDLSynthTerm` property appears in the **Synthesis termination** field of the EDA Tool Scripts dialog box, as shown in the following figure.



See also “Generating Scripts for HDL Simulators and Synthesis Tools”.

Compatibility Considerations

If you have existing models that generate synthesis scripts using the previous defaults for `technology` or `part`, you may want to update your models and regenerate scripts.

Version 1.1 (R2007a) Simulink HDL Coder

This table summarizes what's new in V1.1 (R2007a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	No	Bug Reports	No

New features and changes introduced in this version are

Sign Block Supported for HDL Code Generation

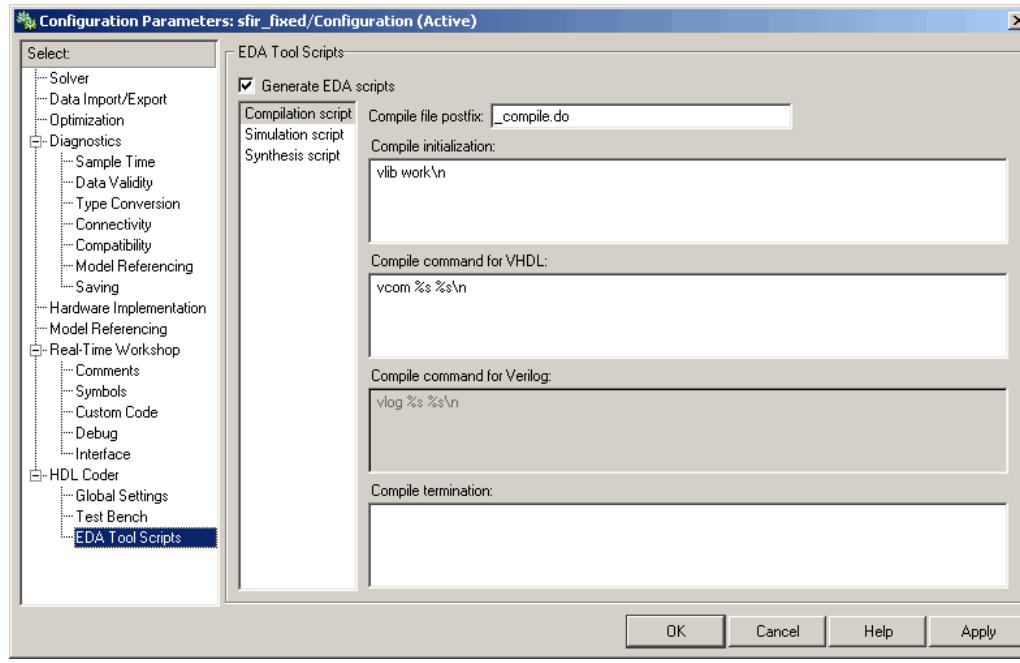
The Sign block (Simulink/Math Operations/Sign) is now supported for HDL code generation. See "Summary of Block Implementations" in the *Simulink HDL Coder User's Guide* for further information.

Link for Cadence Incisive HDL Cosimulation Block Supported

Simulink HDL Coder now supports HDL code generation for the Link for Cadence® Incisive® HDL Cosimulation Block. You can use the HDL Cosimulation block with Simulink HDL Coder to generate an interface to your manually written or legacy HDL code. When an HDL Cosimulation block is included in a model, Simulink HDL Coder generates a VHDL or Verilog interface, depending on the selected target language. See "Code Generation for HDL Cosimulation Blocks" in the *Simulink HDL Coder User's Guide* for details.

GUI Support for Generation of EDA Tool Scripts

The new **EDA Tool Scripts** pane of the Simulink HDL Coder GUI (shown in the following figure) lets you set all options that control generation of script files for third-party electronic design automation (EDA) tools. In previous releases, script generation options were available only through `makehdl` and `makehdltb` properties.



The list on the left of the **EDA Tool Scripts** pane lets you select from the following categories of options:

- **Compilation script**: Options related to customizing scripts for compilation of generated VHDL or Verilog code.
- **Simulation script**: Options related to customizing scripts for HDL simulators.
- **Synthesis script**: Options related to customizing scripts for synthesis tools.

See “Generating Scripts for HDL Simulators and Synthesis Tools” in the *Simulink HDL Coder User’s Guide* for detailed information on the **EDA Tool Scripts** options and on script generation in general.

Embedded MATLAB Function Block Supported for HDL Code Generation

Simulink HDL Coder now supports synthesizable HDL code generation from the Embedded MATLAB Function block. See “Generating HDL Code with the Embedded MATLAB Function Block” for detailed information.

We are interested in getting your feedback on this introductory feature. Please send your responses to: hdlcoder_feedback@mathworks.com.

Stateflow HDL Code Generation Updates

This section describes some limitations on the use of Stateflow® charts in HDL code generation have been removed in the current release. These are:

Restriction on Reading from Output Ports Removed

In the previous release, reading from output ports was disallowed. This restriction has been relaxed. You can now read from output ports if outputs are registered. (Outputs are registered if the **Initialize Outputs Every Time Chart Wakes Up** option is deselected.)

Stateflow HDL Code Generator Fully Supports Simulink Fixed Point Data Type

In the previous release, fixed-point data type support for Stateflow HDL code generation was limited to fixed point without scaling. This limitation has been removed. You can now use Simulink fixed-point data types without restriction in Stateflow charts intended for HDL code generation.

Version 1.0 (R2006b) Simulink HDL Coder

This table summarizes what's new in V1.0 (R2006b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	No	Bug Reports	No

Introduction to Simulink HDL Coder

Simulink HDL Coder lets you generate hardware description language (HDL) code based on models developed in Simulink and finite-state machines developed in Stateflow. Simulink HDL Coder brings the Simulink Model-Based Design approach into the domain of application-specific integrated circuit (ASIC) and field programmable gate array (FPGA) development. Using Simulink HDL Coder, system architects and designers can spend more time on fine-tuning algorithms and models through rapid prototyping and experimentation and less time on HDL coding.

Simulink HDL Coder generates bit-true and cycle-accurate, synthesizable Verilog and VHDL code from Simulink models and Stateflow diagrams. The automatically generated HDL code is target independent.

You can simulate and synthesize the automatically generated HDL using industry standard tools and then map it into field-programmable gate arrays (FPGAs) or application-specific integrated circuits (ASICs). You can also use the automatically generated HDL code to verify existing HDL code using formal or functional verification tools.

Simulink HDL Coder also generates test benches, enabling rapid verification of the generated HDL code using HDL simulation tools.

Version 1.0 of Simulink HDL Coder includes these features:

- Generation of synthesizble VHDL or Verilog code from Simulink models and Stateflow charts

- Code generation configured and initiated via graphical user interface, MATLAB® command-line interface, or M-file programs
- Test bench generation (VHDL or Verilog) for validating generated code
- Generation of models that are bit-true and cycle-accurate with respect to generated HDL code
- Numerous options for controlling the contents and style of the generated HDL code and test bench
- Block support:
  - Simulink built-in
  - Signal Processing Blockset
  - Link for ModelSim HDL Cosimulation block
  - Stateflow chart
  - User-selectable optimized block implementations provided for commonly used blocks
- Code generation control files support:
  - Selection of alternate block implementations for specific blocks or sets of blocks in the model
  - Setting of code generation options
  - Selection of the model or subsystem from which code is to be generated.
  - Definition of default or template HDL code generation settings for your organization
- Generation of subsystem-based identification comments and mapping files for easy tracing of HDL entities back to corresponding elements of the original model
- Generation of interfaces to existing HDL code via:
  - Black box subsystem implementation
  - Cosimulation with ModelSim HDL simulator (requires link for ModelSim)

- Compatibility checker utility that examines your model for HDL code generation compatibility, and generates HTML report with hyperlinks to problematic blocks
- Generation of scripts for EDA tools:
  - ModelSim
  - Synplify
- Model features supported for code generation in Version 1.0:
  - Real data types only (fixed-point and double) (Complex data types are not supported.)
  - Fixed-step, discrete, single-rate models
  - Scalar and vector ports (row or column vectors only)

For More Information

See the Simulink HDL Coder User's Guide for comprehensive information on Simulink HDL Coder.

Compatibility Summary for Simulink HDL Coder

This table summarizes new features and changes that might cause incompatibilities when you upgrade from an earlier version, or when you use files on multiple versions. Details are provided in the description of the new feature or change.

Version (Release)	New Features and Changes with Version Compatibility Impact
Latest Version V1.2 (R2007b)	See the **Compatibility Considerations** subheading for this new feature or change: • "Default Hardware Target for Synthesis Scripts Updated to Virtex-4 " on page 10 • "Support for New Embedded MATLAB Bitwise Functions" on page 10
V1.1 (R2007a)	None
V1.0 (R2006b)	None