Filter Design 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 V1.5 (R2006b)	Yes Details	Yes Summary	Bug Reports	Printable Release Notes: PDF Current product documentation
V1.4 (R2006a)	Yes Details	Yes Summary	Bug Reports	No
V1.3 (R14SP3)	Yes Details	No	Bug Reports	No
V1.2 (R14SP2)	Yes Details	Yes Summary	Bug Reports	No
V1.1 (R14SP1)	Yes Details	Yes Summary	No	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
- Changes to system requirements (complete system requirements for the current version are at the MathWorks Web site)
- Any version compatibility considerations associated with each new feature or change

Version Compatibility Considerations

When a new feature or change introduces a known 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 compatibility impact, see the "Compatibility Summary for Filter Design HDL Coder" on page 21.

Compatibility issues that become known 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 1.5 (R2006b) Filter Design HDL Coder

This table summarizes what's new in Version 1.5 (R2006b).

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details labeled as **Compatibility Considerations**, below. See also Summary.	Bug Reports	Printable Release Notes: PDF Current product documentation

New features and changes introduced in this version are

Distributed Arithmetic Support for FIR Filters

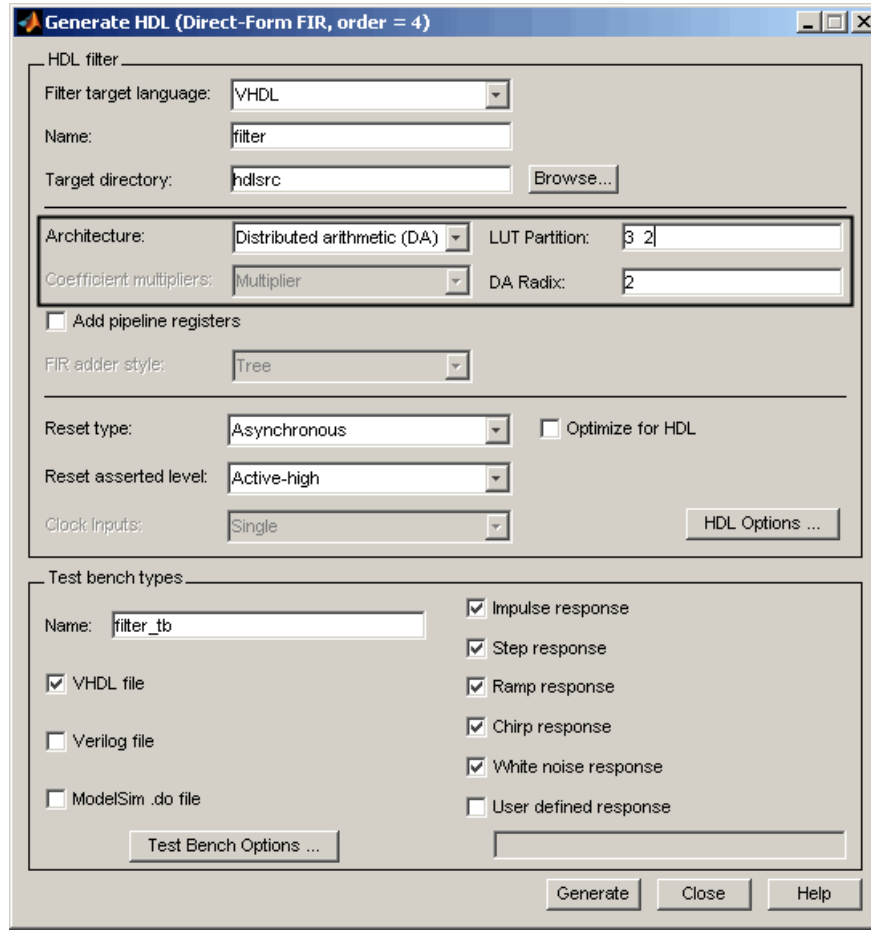
Filter Design HDL Coder now supports Distributed Arithmetic (DA) in HDL code generated for several single-rate and multirate FIR filter structures. DA is a widely-used technique for implementing sum of products computations without use of multipliers. Designers frequently use DA to build efficient Multiply-Accumulate Circuitry (MAC) for filters and other DSP applications.

DA code generation is supported for fixed-point realizations of the following FIR filter structures:

- `dfilt.dffir`
- `dfilt.dfsymfir`
- `dfilt.dfasymfir`
- `mfilt.firdecim`
- `mfilt.firinterp`

You can enable and control DA code generation using `generatehdl` properties provided for that purpose, or by selecting the `Distributed Arithmetic (DA)` option from the **Architecture** pop-up menu in the Generate HDL dialog box (shown in the following figure).

See “Distributed Arithmetic for FIR Filters” in the Filter Design HDL Coder User’s Guide for a complete description of DA related options and properties.



Multirate Support for Fully Serial Architectures

Filter Design HDL Coder 1.5 adds support for generation of fully serial architectures for the following multirate filter types:

- `mfilt.firdecim`
- `mfilt.firinterp`

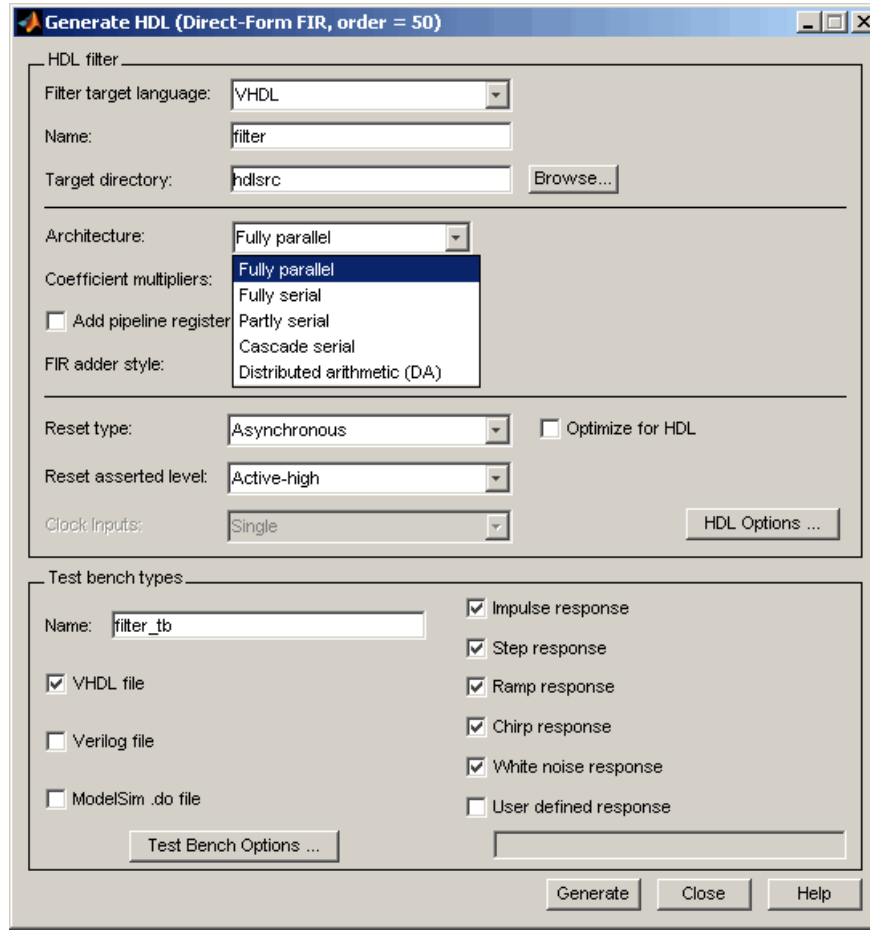
The following table summarizes the filter types that are available for parallel and serial architecture choices in Filter Design HDL Coder 1.5. See "Speed vs. Area Optimizations for FIR Filters" in the Filter Design HDL Coder User's Guide for a full description of these options.

Architecture	Available for Filter Types...
`Fully parallel` (default)	All filter types that are supported for HDL code generation
`Fully serial`	• `dfilt.dffir` • `dfilt.dfsymfir` • `dfilt.dfasymfir` • `mfilt.firdecim` • `mfilt.firinterp`
`Partly serial`	• `dfilt.dffir` • `dfilt.dfsymfir` • `dfilt.dfasymfir`
`Cascade serial`	• `dfilt.dffir` • `dfilt.dfsymfir` • `dfilt.dfasymfir`

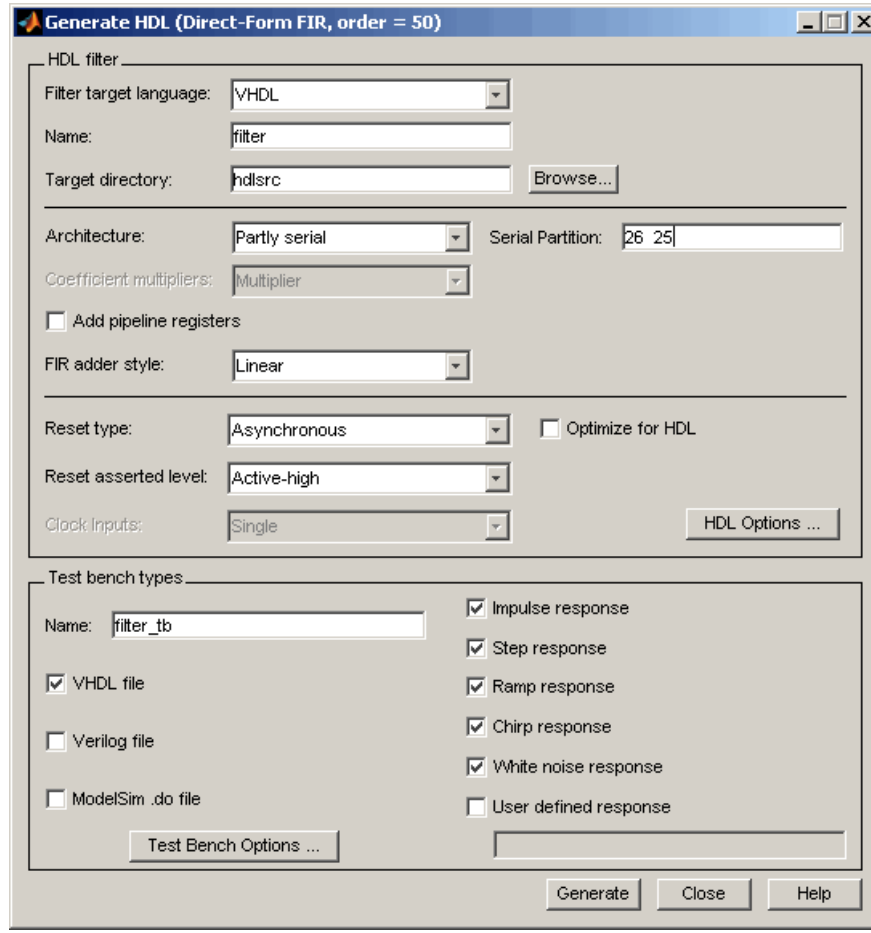
Generate HDL Dialog Box Supports All Parallel and Serial Architecture Options

Previously, the **Architecture** pop-up menu on the HDL Options dialog box provided a choice between two basic (`Fully parallel` or `Fully serial`) architectures. Other architecture options were available only by setting `generatehdl` properties (`ReuseAccum` and `SerialPartition`).

In Filter Design HDL Coder 1.5, the Generate HDL dialog box supports the full range of architecture options. As shown in the following figure, the **Architecture** pop-up menu now includes `Partly serial` and `Cascade serial` options.



When the `Partly serial` or `Cascade serial` option is selected, the Generate HDL dialog box displays the **Serial Partition** field (shown in the following figure). See “Speed vs. Area Optimizations for FIR Filters” in the Filter Design HDL Coder User’s Guide for a full description of serial and parallel architecture options.



Note The **Architecture** pop-up menu also includes the new `Distributed arithmetic (DA)` option (see “Distributed Arithmetic Support for FIR Filters” on page 4).

Enhanced Code Generation for Symmetric Multirate FIR Filters

In this release, Filter Design HDL Coder enhances code generation for Direct-Form FIR Polyphase Decimator (`mfilt.firdecim`) filters by using the symmetry in polyphase coefficients for each FIR subfilter. The code generator inserts adders before multipliers to sum the input samples that correspond to to the symmetric taps.

EDAScriptGeneration Property Added

The `EDAScriptGeneration` property controls the generation of script files. By default, `EDAScriptGeneration` is set `'on'`. To disable script generation, set `EDAScriptGeneration` to `'off'`, as in the following example:

```
generatehdl(Hd,'EDAScriptGeneration','off')
```

See “Generating Scripts for EDA Tools” in the Filter Design HDL Coder documentation for further information.

ResetValue Property Merged with ResetAssertedLevel Property

In previous releases, the `ResetValue` property (or the **Reset value** option in the Test Bench Options dialog box) allowed test bench reset input signal levels (`active-high` or `active-low`) to be set independently from the level specified for resets in the generated filter code.

In this release, the `ResetValue` property has been merged with the `ResetAssertedLevel` property (**Reset asserted level** menu in the **HDL filter** pane of the Generate HDL dialog box). The **Reset asserted level** setting determines the rest level for both filter and test bench reset input signals, ensuring consistency among reset signals.

Compatibility Considerations

If you have existing M-file scripts or saved FDATool settings that rely on setting the `ResetValue` property independently of `ResetAssertedLevel`, you should change them to use only `ResetAssertedLevel`.

Clock EnableValue for Test Benches Always Active-High

The clock enable value for test benches is now always active-high. The `ClockEnableValue` property and the corresponding **Clock enable value** option in the Test Bench Options dialog box have been removed. Setting an active-low clock enable value for test benches is no longer supported.

Compatibility Considerations

You should remove any code that sets or references the `ClockEnableValue` property from your existing M-file scripts.

Version 1.4 (R2006a) Filter Design HDL Coder

This table summarizes what's new in V1.4 (R2006a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details labeled as **Compatibility Considerations**, below. See also Summary.	Bug Reports at Web site	No

New features and changes introduced in this version are

Speed vs. Area Tradeoff Options for FIR Filters

Filter Design HDL Coder 1.4 provides options that extend your control over speed vs. area tradeoffs in the realization of single-rate direct-form FIR filter designs.

This release note summarizes the new options. See "Speed vs. Area Optimizations for FIR Filters" in the Filter Design HDL Coder User's Guide for full details and examples. Further examples are given in the HDL Serial Architectures for FIR Filters demo (`hdlserialfir.m`).

To achieve the desired speed vs. area tradeoff, you can either specify a *fully parallel* architecture for generated HDL filter code, or choose one of several *serial* architectures. The following architectures are supported:

- *Fully parallel*: This is the default option. A fully parallel architecture uses a dedicated multiplier and adder for each filter tap; all taps execute in parallel. A fully parallel architecture is optimal for speed. However,

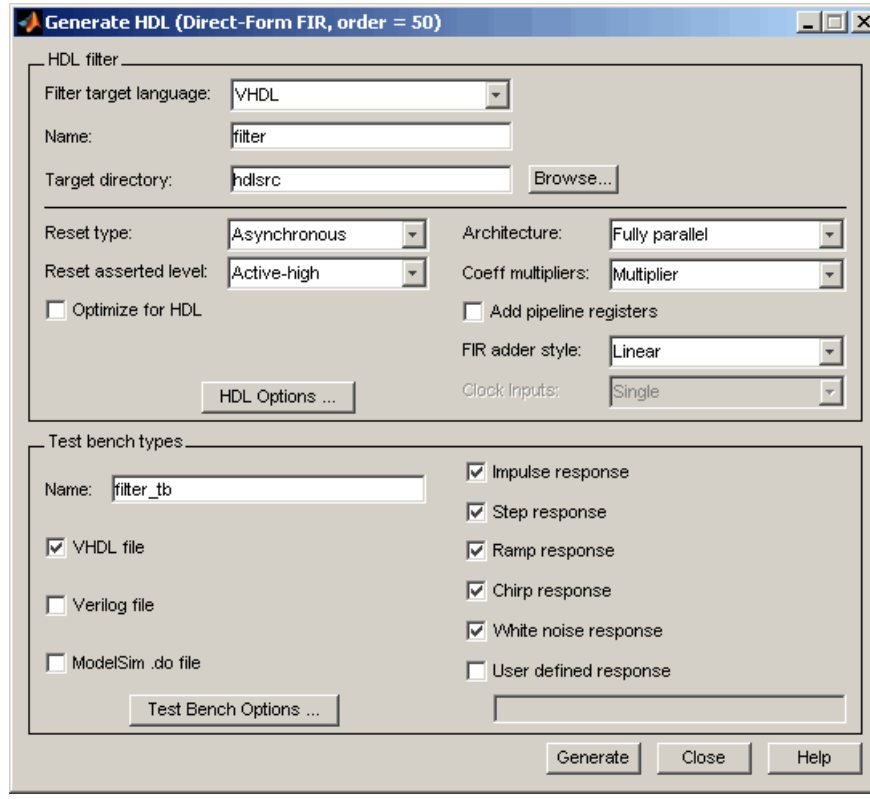
it requires more multipliers and adders than a serial architecture, and therefore consumes more chip area.

- *Fully serial*: A fully serial architecture conserves area by reusing multiplier and adder resources sequentially. For example, a four-tap filter design would use a single multiplier and adder, executing a multiply/accumulate once for each tap. The multiply/accumulate section of the design runs at four times the filter's input/output sample rate. This saves area at the cost of some speed loss and higher power consumption.
- *Partly serial*: Partly serial architectures cover the full range of speed vs. area tradeoffs that lie between the two extreme cases, fully parallel and fully serial architectures.

  In a partly serial architecture, the filter taps are grouped into a number of serial *partitions*. The taps within each partition execute serially, but the partitions execute in parallel with respect to one another. The outputs of the partitions are summed at the final output.
- *Cascade-serial*: A cascade-serial architecture closely resembles a partly serial architecture. As in a partly serial architecture, the filter taps are grouped into a number of serial partitions that execute in parallel with respect to one another. However, the accumulated output of each partition is cascaded to the accumulator of the previous partition. The output of all partitions is therefore computed at the accumulator of the first partition. This technique is termed *accumulator reuse*. No final adder is required, which saves area.

The full range of parallel/serial architecture options is supported by new properties passed in to the `generatehdl` command.

Alternatively, you can use the new **Architecture** option on the HDL Options dialog box (see the following figure) to choose between the basic `Fully Parallel` or `Fully Serial` architectures.



The new options are supported for the following filter types:

- `dfilt.dffir`
- `dfilt.dfsymfir`
- `dfilt.dfasymfir`

Code Generation Support for Delay Filter

Filter Design HDL Coder now supports code generation for the Delay filter type (`dfilt.delay`). See the Signal Processing Toolbox documentation for information on this filter type.

The Delay filter is often used in a cascade with other filter types. See "Generating Code for Cascade Filters" Filter Design HDL Coder User's Guide for general considerations on using cascade filters in code generation.

Rounding Behavior in Generated HDL Code

In Release 2006a, filter objects (and fixed-point arithmetic in general) support a fixed-point rounding mode (`Round`) that behaves identically to the MATLAB® `round` function. However, Filter Design HDL Coder does not support this rounding behavior in generated HDL code. When generating code from a filter that has the `RoundMode` property set to `Round`, Filter Design HDL Coder uses `Nearest` rounding mode instead. A warning is issued when code generation is initiated, as shown in the following example.

```
b = [0.05 0.9 0.05];
Hd = dfilt.dffir(b);
Hd.arithmetic = 'fixed';
Hd.FilterInternals = 'SpecifyPrecision';
Hd.RoundMode = 'Round';
generatehdl(Hd);
Warning: RoundMode 'round' is not supported for HDL generation. Using 'nearest' instead.
.
.
.
### Successful completion of VHDL code generation process for filter: Hd
```

If you are generating code from a fixed-point filter created in FDATool, this situation does not occur because the FDATool **Round towards** menu does not include the `Round` option.

Compatibility Considerations

Before generating HDL code from your existing filter objects, check the `RoundMode` property and if it is set to `Round`, use another mode to avoid the warning.

Version 1.3 (R14SP3) Filter Design HDL Coder

This table summarizes what's new in V1.3 (R14SP3):

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

New features and changes introduced in this version are

- "Generating Scripts for EDA Tools" on page 16
- "Test Bench Generation Improved for Multirate Filters" on page 16

Generating Scripts for EDA Tools

Filter Design HDL Coder now supports generation of script files for third-party Electronic Design Automation (EDA) tools. These scripts let you compile and simulate generated HDL code and/or synthesize generated HDL code.

Using the defaults, you can automatically generate scripts for the following tools:

- Mentor Graphics ModelSim SE/PE HDL simulator
- The Synplify family of synthesis tools

See "Generating Scripts for EDA Tools" in the Filter Design HDL Coder User's Guide for a detailed description.

Test Bench Generation Improved for Multirate Filters

The speed of generation of large test bench files for multirate filters has been improved significantly for this release.

Version 1.2 (R14SP2) Filter Design HDL Coder

This table summarizes what's new in V1.2 (R14SP2):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details labeled as **Compatibility Considerations**, below. See also Summary.	Bug Fixes	No

New features and changes introduced in this version are

Additional Multirate and Discrete Filter Types Supported

The Filter Design HDL Coder now adds code generation support for the following multirate and discrete filter types:

- Direct-Form FIR Polyphase Interpolator (`mfilt.firinterp`)
- Direct-Form FIR Polyphase Decimator (`mfilt.firdecim`)
- FIR Hold Interpolator (`mfilt.holdinterp`)
- FIR Linear Interpolator (`mfilt.linearinterp`)
- Discrete-Time Scalar (`dfilt.scalar`)

For a complete list of filter structures supported for code generation, see "Key Features and Components" in the Filter Design HDL Coder online documentation.

Code Generation Support for Interpolating Filters in Cascades

In the previous release, only decimators and/or single-rate filter structures could be included in a cascade for code generation purposes.

The Filter Design HDL Coder 1.2 now supports code generation for cascades that include interpolators. You can generate code for cascades that combine the following filter types:

- Decimators and/or single-rate filter structures
- Interpolators and/or single-rate filter structures

Code generation for cascades that include both decimators and interpolators is not currently supported, however.

See also “Generating Code for Cascade Filters” in the Filter Design HDL Coder online documentation.

InitializeRealSignals Property and GUI Option Removed

The Filter Design HDL Coder Version 1.2 always initializes signals of type `REAL` with a value of `0.0`.

In previous releases, initialization code for real signals was generated optionally. Generation of such initialization code was controlled by the `InitializeRealSignals` property and the corresponding **Initialize real signals** option in the **Advanced** pane of the HDL Options dialog box. The **Initialize real signals** option is no longer supported and has been removed from the **Advanced** pane. The `InitializeRealSignals` property is set to `'on'` and is no longer user settable.

Compatibility Considerations

Consider removing code that sets the `InitializeRealSignals` property.

Version 1.1 (R14SP1) Filter Design HDL Coder

This table summarizes what's new in V1.1 (R14SP1):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details labeled as **Compatibility Considerations**, below. See also Summary.	No	No

New features and changes introduced in this version are

Multirate Filter Support

The Filter Design HDL Coder now supports code generation for several types of multirate filters:

- Cascaded Integrator Comb (CIC) interpolation (`mfilt.cicdecim`)
- Cascaded Integrator Comb (CIC) decimation (`mfilt.cicinterp`)
- Direct-Form Transposed FIR Polyphase Decimator (`mfilt.firtdecim`)

For details, see "Generating Code for Multirate Filters" in the Filter Design HDL Coder User's Guide.

Cascade Filter Support

The Filter Design HDL Coder now supports code generation for the following types of cascade filters:

- Multirate cascade of filter objects (`mfilt.cascade`)
- Cascade of discrete-time filter objects (`dfilt.cascade`)

See “Generating Code for Cascade Filters” in the Filter Design HDL Coder User’s Guide for details.

CoeffPrefix Property Replaces CoeffName Property

The `CoeffName`property has been renamed to `CoeffPrefix`. The purpose of the `CoeffPrefix` remains unchanged. This property specifies a string to be used as the prefix for filter coefficient names.

Compatibility Considerations

For backward compatibility, `CoeffName` is still supported. Existing code that uses `CoeffName` will run without change. However, The MathWorks recommends updating your code to use the current property name.

Compatibility Summary for Filter Design 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 V1.5 (R2006b)	See the **Compatibility Considerations** subheading for this new feature or change: • "ResetValue Property Merged with ResetAssertedLevel Property" on page 10 • "Clock EnableValue for Test Benches Always Active-High" on page 11
V1.4 (R2006a)	See the **Compatibility Considerations** subheading for this new feature or change: • "Rounding Behavior in Generated HDL Code" on page 15
V1.3 (R14SP3)	None

Version (Release)	New Features and Changes with Version Compatibility Impact
V1.2 (R14SP2)	See the **Compatibility Considerations** subheading for this new feature or change: • "InitializeRealSignals Property and GUI Option Removed" on page 18
V1.1 (R14SP1)	See the **Compatibility Considerations** subheading for this new feature or change: • "CoeffPrefix Property Replaces CoeffName Property" on page 20