

# **Filter Design HDL Coder Release Notes**

---



|  |           |
|--|-----------|
| <b>Summary by Version</b> .....                                | <b>1</b>  |
| <b>Version 2.0 (R2007a) Filter Design HDL Coder</b> .....      | <b>4</b>  |
| <b>Version 1.5 (R2006b) Filter Design HDL Coder</b> .....      | <b>19</b> |
| <b>Version 1.4 (R2006a) Filter Design HDL Coder</b> .....      | <b>27</b> |
| <b>Version 1.3 (R14SP3) Filter Design HDL Coder</b> .....      | <b>31</b> |
| <b>Version 1.2 (R14SP2) Filter Design HDL Coder</b> .....      | <b>32</b> |
| <b>Version 1.1 (R14SP1) Filter Design HDL Coder</b> .....      | <b>34</b> |
| <b>Compatibility Summary for Filter Design HDL Coder</b> ..... | <b>36</b> |



## Summary by Version

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

| <b>Version (Release)</b>            | <b>New Features and Changes</b> | <b>Version Compatibility Considerations</b> | <b>Fixed Bugs and Known Problems</b> | <b>Related Documentation at Web Site</b>                      |
|-------------------------------------|---------------------------------|---|--------------------------------------|---|
| <b>Latest Version V2.0 (R2007a)</b> | Yes<br>Details                  | No  | No                                   | Printable Release Notes: PDF<br>Current product documentation |
| V1.5 (R2006b)                       | Yes<br>Details                  | Yes<br>Summary                              | Bug Reports                          | No  |
| V1.4 (R2006a)                       | Yes<br>Details                  | Yes<br>Summary                              | Bug Reports                          | No  |
| V1.3 (R14SP3)                       | Yes<br>Details                  | No  | Bug Reports                          | No  |
| V1.2 (R14SP2)                       | Yes<br>Details                  | Yes<br>Summary                              | Bug Reports                          | No  |
| V1.1 (R14SP1)                       | Yes<br>Details                  | Yes<br>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 36.

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 2.0 (R2007a) Filter Design HDL Coder

This table summarizes what's new in Version 2.0 (R2007a).

| New Features and Changes | Version Compatibility Considerations | Fixed Bugs and Known Problems | Related Documentation at Web Site                                 |
|--------------------------|--------------------------------------|-------------------------------|---|
| Yes<br>Details below     | No                                   | No                            | Printable Release Notes: PDF<br><br>Current product documentation |

New features and changes introduced in this version are:

- “Farrow Filter Code Generation” on page 4
- “Code Generation Support for Polyphase Sample Rate Converters (mfilt.firsrc)” on page 5
- “filterbuilder Supports HDL Code Generation” on page 5
- “fdhdltool Function Opens Generate HDL Dialog Box from MATLAB Command Line” on page 7
- “GUI Enhancements and Revisions” on page 7
- “EDA Tool Scripts Dialog Box” on page 12
- “Multiple Clocks Supported for Multirate Filters with Distributed Arithmetic and Fully Serial Architectures” on page 16

### Farrow Filter Code Generation

Filter Design HDL Coder 2.0 supports HDL code generation for Farrow filters. The Farrow filter structures supported are:

- `farrow.fd`
- `farrow.linearfd`

A Farrow filter differs from a conventional filter because it has a fractional delay input in addition to a signal input. The fractional delay input enables



the use of time-varying delays, as the filter operates. The fractional delay input receives a signal taking on values between 0 and 1.0. For general information how to construct and use Farrow filters in MATLAB, see the `farrow` function reference section of the Filter Design Toolbox documentation.

Filter Design HDL Coder provides `generatetb` and `generatehdl` properties and equivalent GUI options that let you :

- Define the fractional delay port name used in generated code.
- Apply a variety of test bench stimulus signals to the fractional delay port, or define your own stimulus signal.

See “Generating Code for Farrow Filters” in the Filter Design HDL Coder User’s Guide for a complete description of this feature.

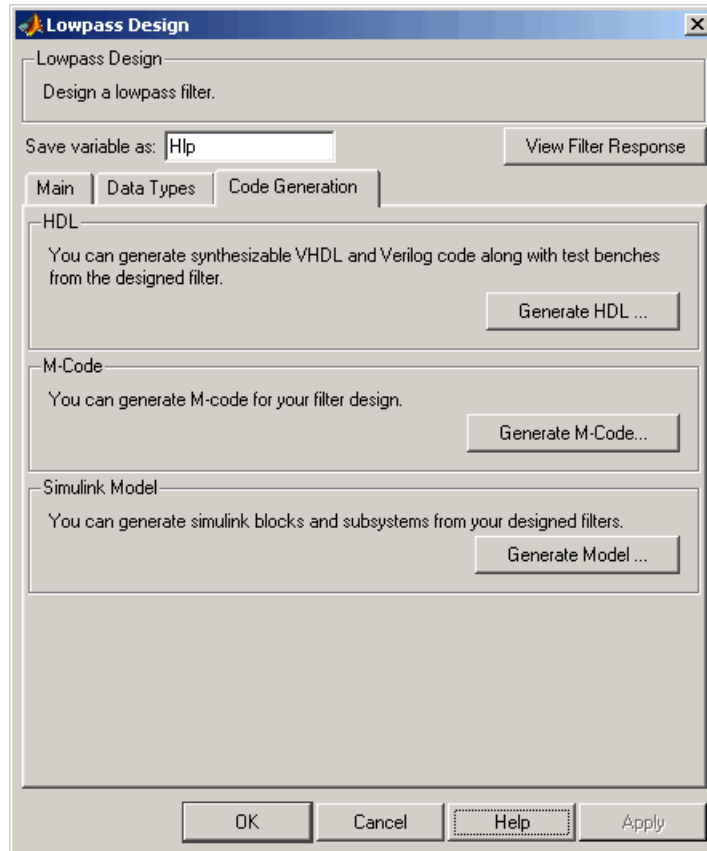
## **Code Generation Support for Polyphase Sample Rate Converters (`mfilt.firsrc`)**

Filter Design HDL Coder 2.0 supports code generation for direct-form FIR polyphase sample rate converters (`mfilt.firsrc`). `mfilt.firsrc` is a multirate filter structure that combines an interpolation factor and a decimation factor, allowing you to perform fractional interpolation or decimation on an input signal.

For detailed information on this feature, see “Generating Code for Polyphase Sample Rate Converters” in the Filter Design HDL Coder User’s Guide.

## **filterbuilder Supports HDL Code Generation**

You can now use the `filterbuilder` tool to generate HDL code for any filter object designed in `filterbuilder`. The `filterbuilder` GUI now includes a **Code Generation** pane (shown in the following figure).



- 1 To generate HDL code from filterbuilder:
- 2 Click the **Code Generation** tab.
- 3 In the **Code Generation** pane, click the **Generate HDL** button. This opens the Generate HDL dialog box, passing in the current filter object from filterbuilder.
- 4 Set the desired code generation and test bench options and generate code in the Generate HDL dialog box.

See also “GUI Enhancements and Revisions” on page 7 to learn about changes that have been made to the Generate HDL dialog box and its subordinate dialog boxes.

## **fdhdltool Function Opens Generate HDL Dialog Box from MATLAB Command Line**

`fdhdltool` is a convenience function that lets you open the Generate HDL dialog box from the MATLAB command line.

The command syntax is

```
fdhdltool(Hd)
```

where `Hd` is a filter object.

The `fdhdltool` function is particularly useful when you need to use the Filter Design HDL Coder GUI to generate HDL code for filter structures that are not supported by `FDATool` or `filterbuilder`. For example, the following commands create a Farrow linear fractional delay filter object `Hd`, which is passed in to the `fdhdltool` function.

```
D = .3;  
Hd = farrow.linearfd(D);  
Hd.arithmetic = 'fixed';  
fdhdltool(Hd);
```

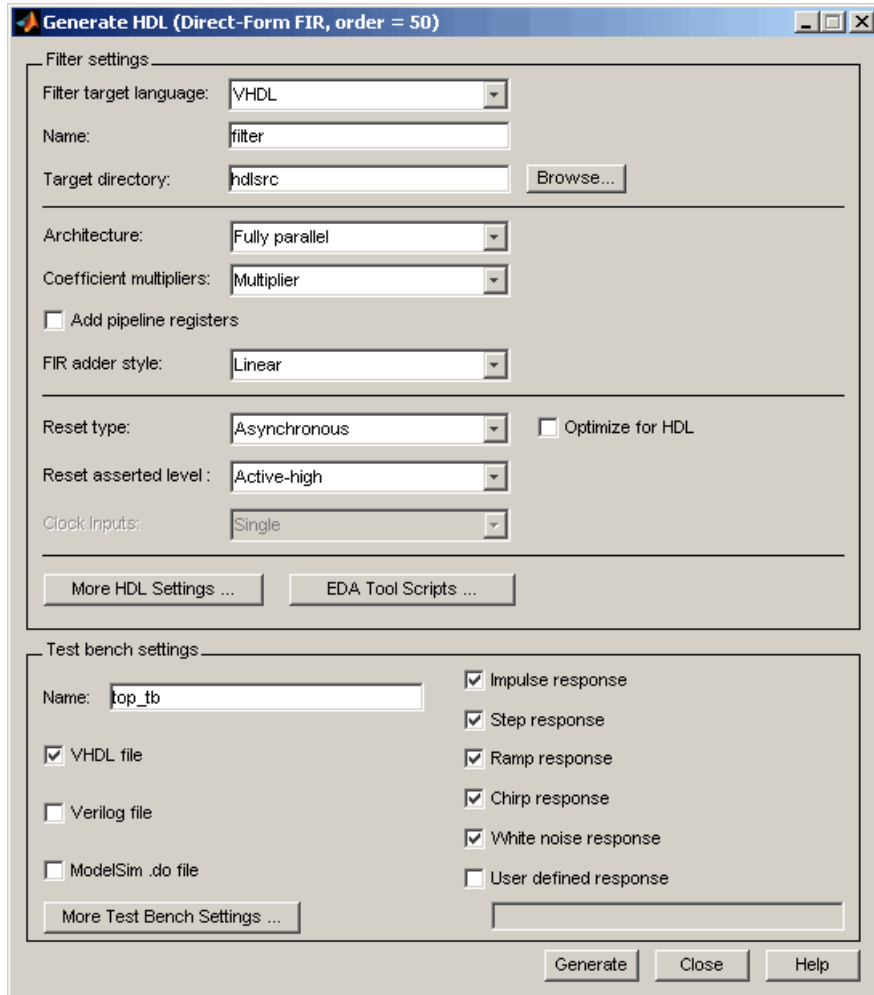
`fdhdltool` operates on a copy of the filter object, rather than the original object in the MATLAB workspace. Any changes made to the original filter object after `fdhdltool` is invoked will not affect the copy and will not update the Generate HDL dialog box.

The naming convention for the copied object is *filt\_copy*, where *filt* is the name of the original filter object.

## **GUI Enhancements and Revisions**

For release 2.0, significant revisions and enhancements have been made to the Filter Design HDL Coder GUI.

## Generate HDL Dialog Box



The preceding figure shows the Generate HDL dialog box. Revisions and enhancements to this dialog box include:

- The new **EDA Tool Scripts** button opens the EDA Tool Scripts dialog box, which lets you set properties that control generation of script files

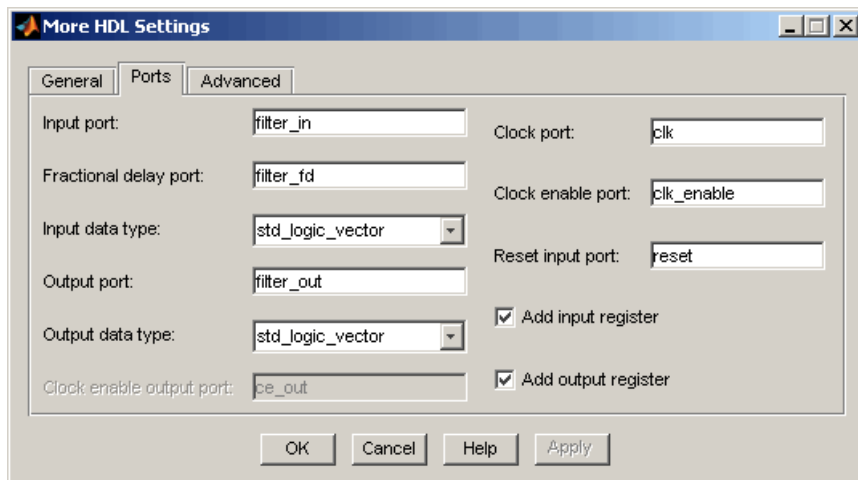
for third-party electronic design automation (EDA) tools. See “EDA Tool Scripts Dialog Box” on page 12.

- The **More HDL Settings** button opens the More HDL Settings dialog box, which replaces the HDL Options dialog box.
- The **More Test Bench Settings** button opens the More Test Bench Settings dialog box, which replaces the Test Bench Options dialog box.

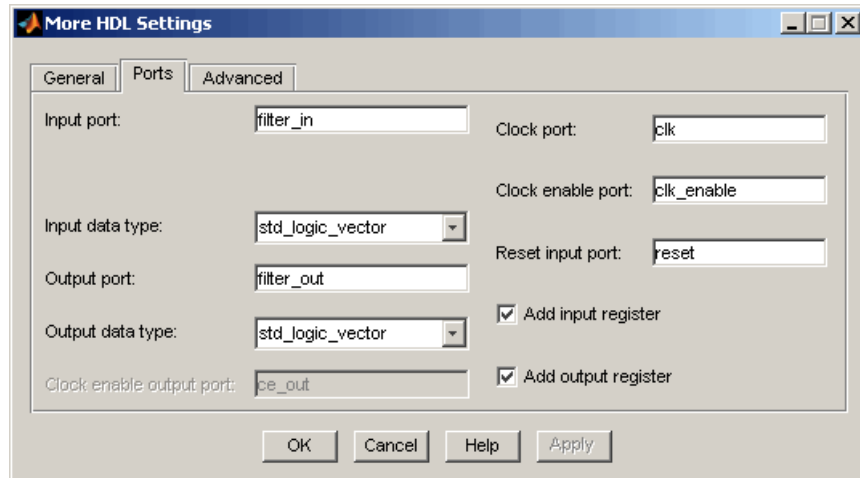
### More HDL Settings Dialog Box

The More HDL Settings dialog box differs slightly from the HDL Settings dialog box, which it replaces.

In the **Ports** pane, when the current filter object is a Farrow filter (see “Farrow Filter Code Generation” on page 4), the new **Fractional delay port** field is displayed, as shown in the following figure.



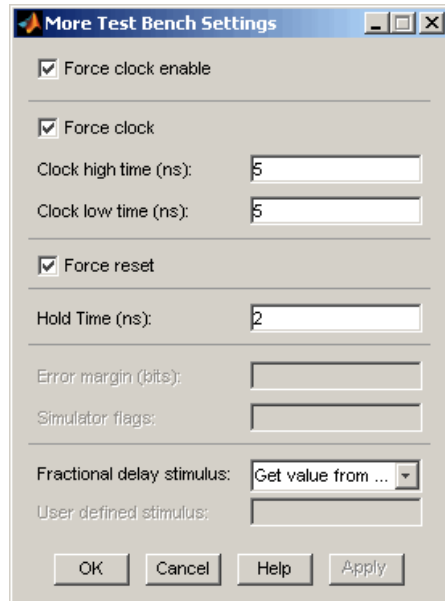
For all other filter types, the **Fractional delay port** field is omitted, as shown in the following figure.



### More Test Bench Settings Dialog Box

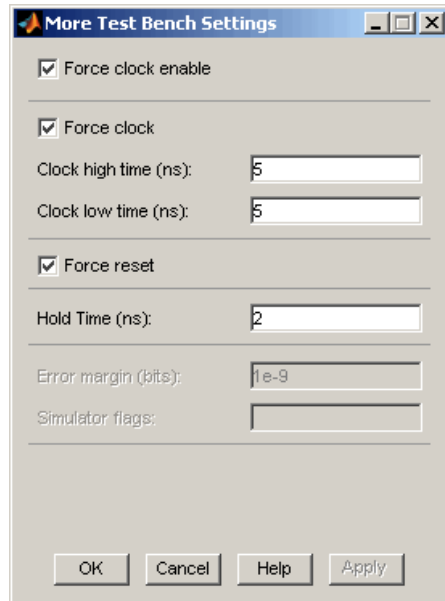
The More Test Bench Settings dialog box differs slightly from the Test Bench Settings dialog box, which it replaces.

When the current filter object is a Farrow filter (see “Farrow Filter Code Generation” on page 4), the new **Fractional delay stimulus** and **User defined stimulus** options are displayed, as shown in the following figure.



The image shows a dialog box titled "More Test Bench Settings" with a blue header bar containing a small icon and standard window controls (minimize, maximize, close). The dialog is divided into several sections by horizontal lines. The first section contains a checked checkbox labeled "Force clock enable". The second section contains a checked checkbox labeled "Force clock", followed by two input fields: "Clock high time (ns):" with the value "5" and "Clock low time (ns):" with the value "5". The third section contains a checked checkbox labeled "Force reset", followed by an input field "Hold Time (ns):" with the value "2". The fourth section contains an empty input field "Error margin (bits):". The fifth section contains an empty input field "Simulator flags:". The sixth section contains a dropdown menu "Fractional delay stimulus:" with the selected option "Get value from ...". The seventh section contains an empty input field "User defined stimulus:". At the bottom of the dialog are four buttons: "OK", "Cancel", "Help", and "Apply".

For all other filter types, the **Fractional delay stimulus** and **User defined stimulus** options are omitted, as shown in the following figure.

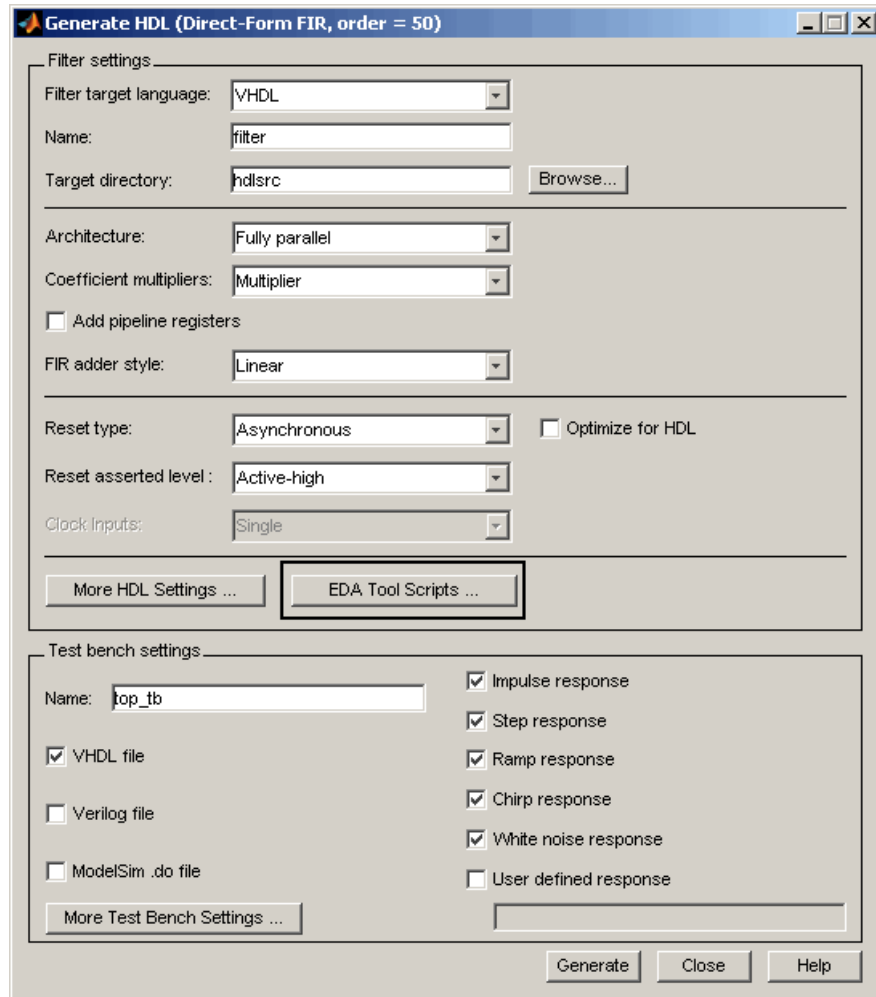


## EDA Tool Scripts Dialog Box

The new EDA Tool Scripts dialog box 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 `generatehdl` properties.

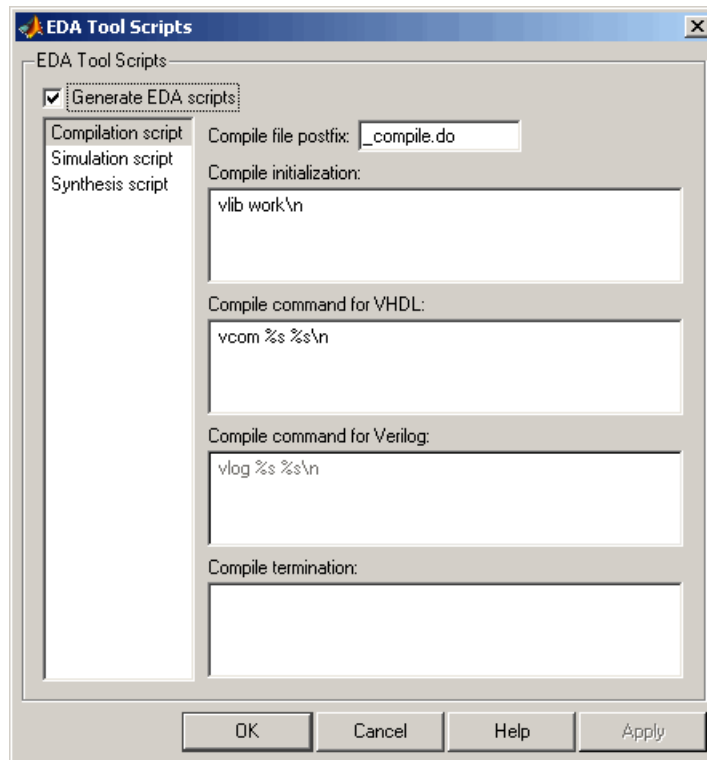
To open the EDA Tool Scripts dialog box, click on the **EDA Tool Scripts** button in the Generate HDL dialog box (shown in the following figure).



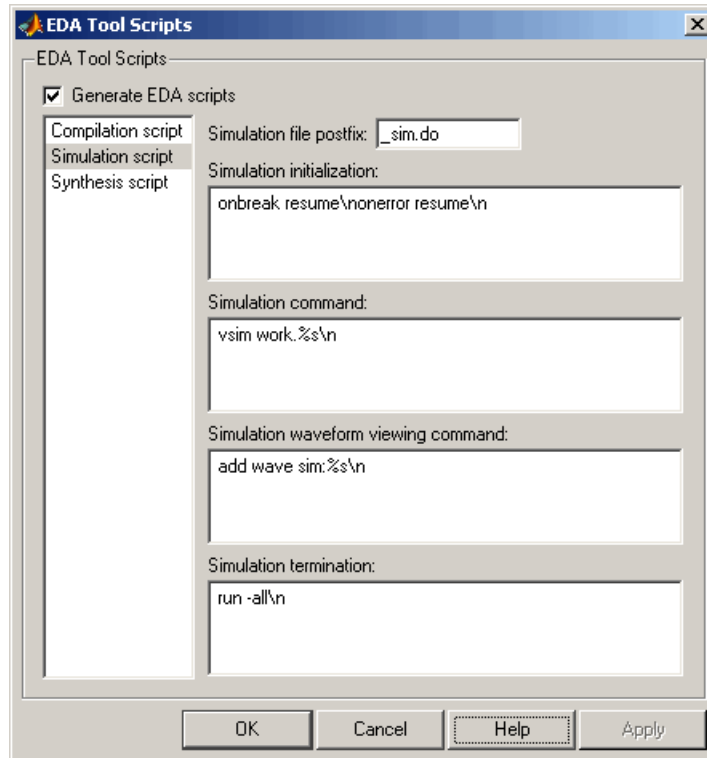


The following figures show the three panes of the EDA Tool Scripts dialog box.

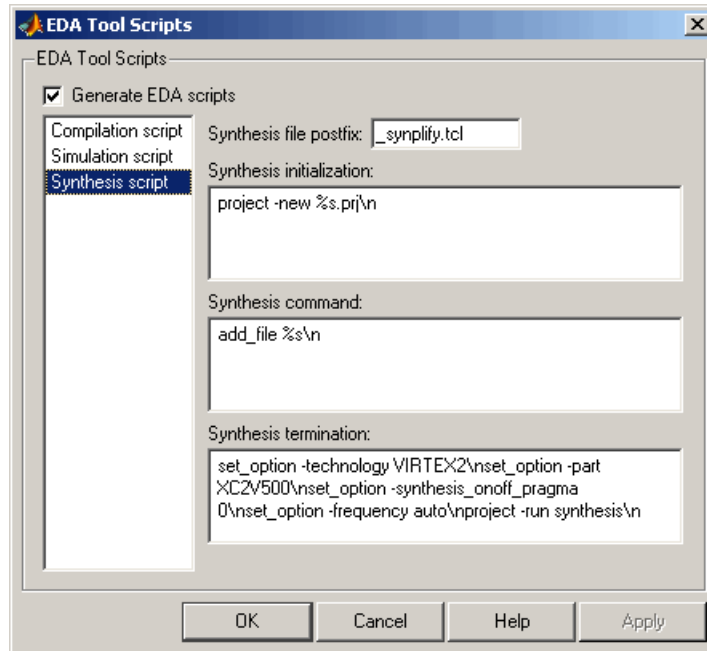
The **Compilation script** pane displays options related to customizing scripts for compilation of generated VHDL or Verilog code.



The **Simulation script** pane displays options related to customizing scripts for HDL simulators.



The **Synthesis script** pane displays options related to customizing scripts for synthesis tools.



See “Generating Scripts for EDA Tools” for a detailed description of script generation options.

## Multiple Clocks Supported for Multirate Filters with Distributed Arithmetic and Fully Serial Architectures

In previous releases, for multirate filters with a distributed arithmetic (DA) or fully serial architecture specified, the **Clock inputs** options was set to Single and disabled.

Filter Design HDL Coder 2.0 supports specification of either single or multiple clock inputs for multirate filters with a DA or fully serial architecture.

For example, in the following figure, the **Clock inputs** option was set to Multiple for a direct-form FIR polyphase interpolator (`mfilt.firinterp`) with a DA architecture.

**Generate HDL (Direct-Form FIR Polyphase Interpolator, order = 47)**

**Filter settings**

Filter target language: VHDL

Name: filter

Target directory: hdlsrc

---

Architecture: Distributed arithmetic (DA) LUT Partition: 8 8 8

Coefficient multipliers: Multiplier DA Radix: 2

Add pipeline registers

FIR adder style: Tree

---

Reset type: Asynchronous  Optimize for HDL

Reset asserted level: Active-high

Clock Inputs: Multiple

---

**Test bench settings**

Name: top\_tb

VHDL file  Impulse response

Verilog file  Step response

ModelSim .do file  Ramp response

Chirp response

White noise response

User defined response

---

**Note** For multirate filters with the Partly serial architecture option selected, the **Clock inputs** options is set to Single and disabled.

---

See also:

- “Distributed Arithmetic for FIR Filters” in the *Filter Design HDL Coder User’s Guide* for a complete description of DA related options and properties.
- “Speed vs. Area Optimizations for FIR Filters” in the *Filter Design HDL Coder User’s Guide* for a complete description of serial architectures.

## 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<br>Details below     | Yes—Details labeled as <b>Compatibility Considerations</b> , below. See also Summary. | Bug Reports                   | No                                |

New features and changes introduced in this version are

- “Distributed Arithmetic Support for FIR Filters” on page 19
- “Multirate Support for Fully Serial Architectures” on page 21
- “Generate HDL Dialog Box Supports All Parallel and Serial Architecture Options” on page 22
- “Enhanced Code Generation for Symmetric Multirate FIR Filters” on page 25
- “EDAScriptGeneration Property Added” on page 25
- “ResetValue Property Merged with ResetAssertedLevel Property” on page 25
- “Clock EnableValue for Test Benches Always Active-High” on page 26

### 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.



**Generate HDL (Direct-Form FIR, order = 4)**

HDL filter

Filter target language: VHDL

Name: filter

Target directory: hdlsrc

---

Architecture: Distributed arithmetic (DA) LUT Partition: 3 2

Coefficient multipliers: Multiplier DA Radix: 2

Add pipeline registers

FIR adder style: Tree

---

Reset type: Asynchronous  Optimize for HDL

Reset asserted level: Active-high

Clock inputs: Single

---

Test bench types

Name: filter\_tb

VHDL file

Verilog file

ModelSim .do file

Impulse response

Step response

Ramp response

Chirp response

White noise response

User defined response

## 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`

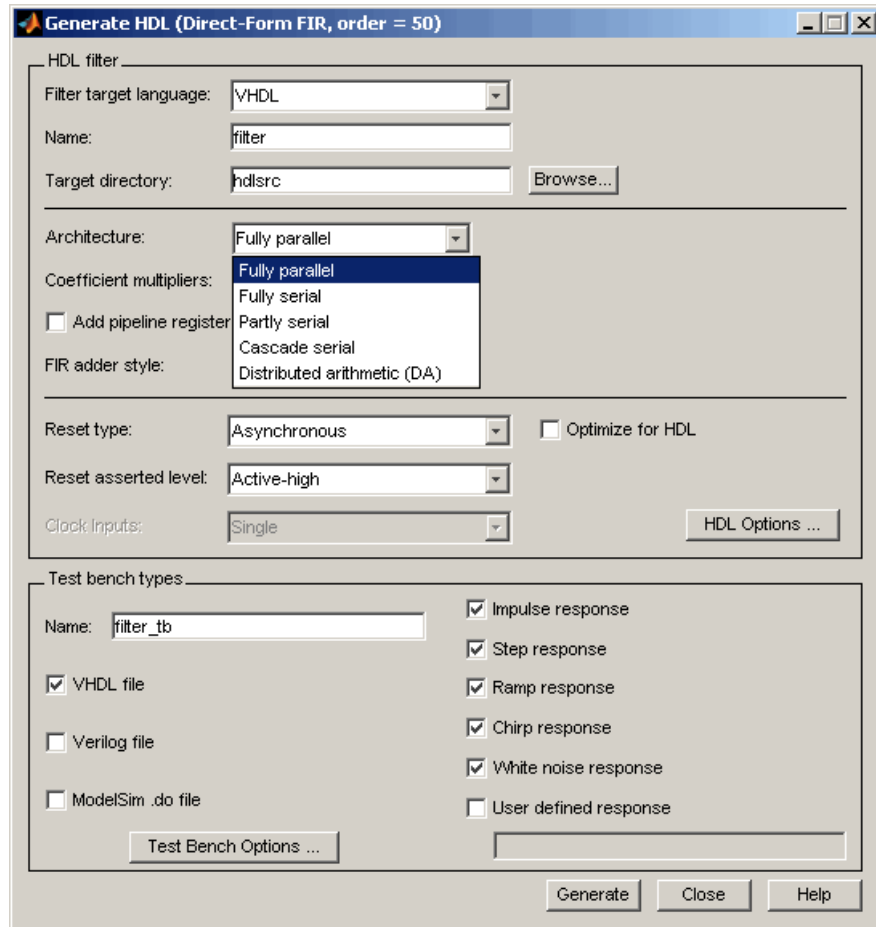
The following table summarizes the filter types that are available for parallel and serial architecture choices in Filter Design HDL Coder 1.5. See “Speed vs. Area Optimizations for FIR Filters” in the Filter Design HDL Coder User’s Guide for a full description of these options.

| <b>Architecture</b>      | <b>Available for Filter Types...</b>   |
|--------------------------|--|
| Fully parallel (default) | All filter types that are supported for HDL code generation  |
| Fully serial             | <ul style="list-style-type: none"> <li>• <code>dfilt.dffir</code></li> <li>• <code>dfilt.dfsymfir</code></li> <li>• <code>dfilt.dfasymfir</code></li> <li>• <code>mfilt.firdecim</code></li> <li>• <code>mfilt.firinterp</code></li> </ul> |
| Partly serial            | <ul style="list-style-type: none"> <li>• <code>dfilt.dffir</code></li> <li>• <code>dfilt.dfsymfir</code></li> <li>• <code>dfilt.dfasymfir</code></li> </ul>  |
| Cascade serial           | <ul style="list-style-type: none"> <li>• <code>dfilt.dffir</code></li> <li>• <code>dfilt.dfsymfir</code></li> <li>• <code>dfilt.dfasymfir</code></li> </ul>  |

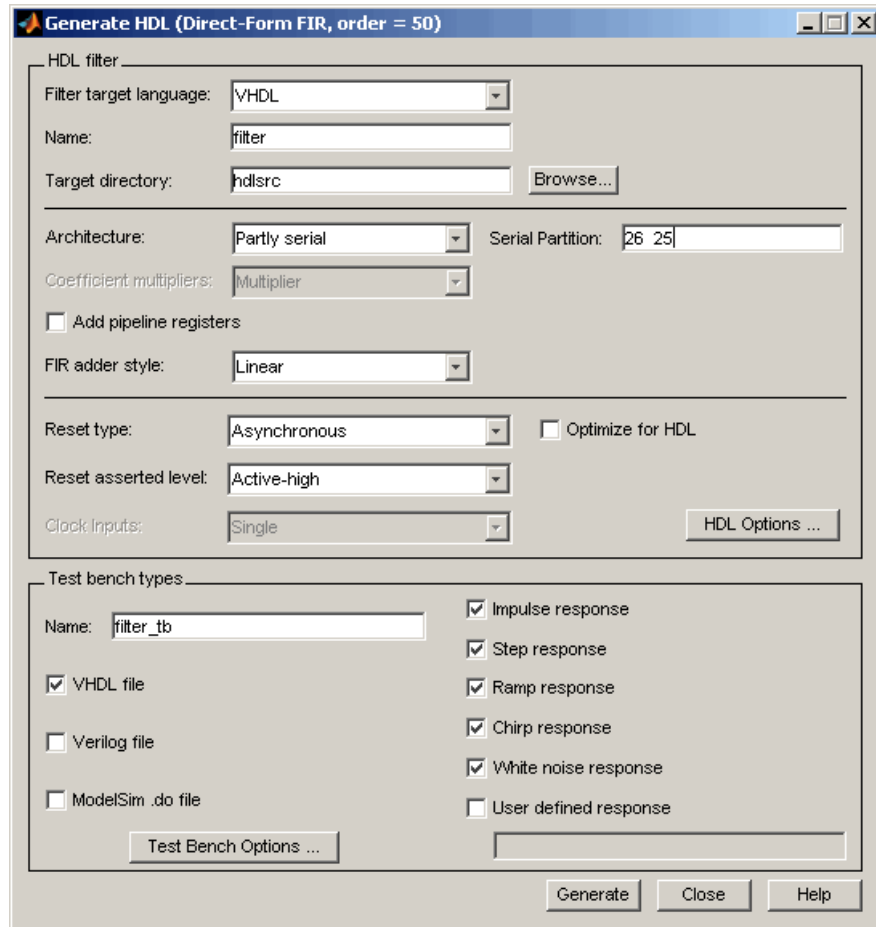
## Generate HDL Dialog Box Supports All Parallel and Serial Architecture Options

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 19).

## 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 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 User’s Guide 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<br>Details below     | Yes—Details labeled as <b>Compatibility Considerations</b> , 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” on page 27
- “Code Generation Support for Delay Filter” on page 29
- “Rounding Behavior in Generated HDL Code” on page 30

### 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.

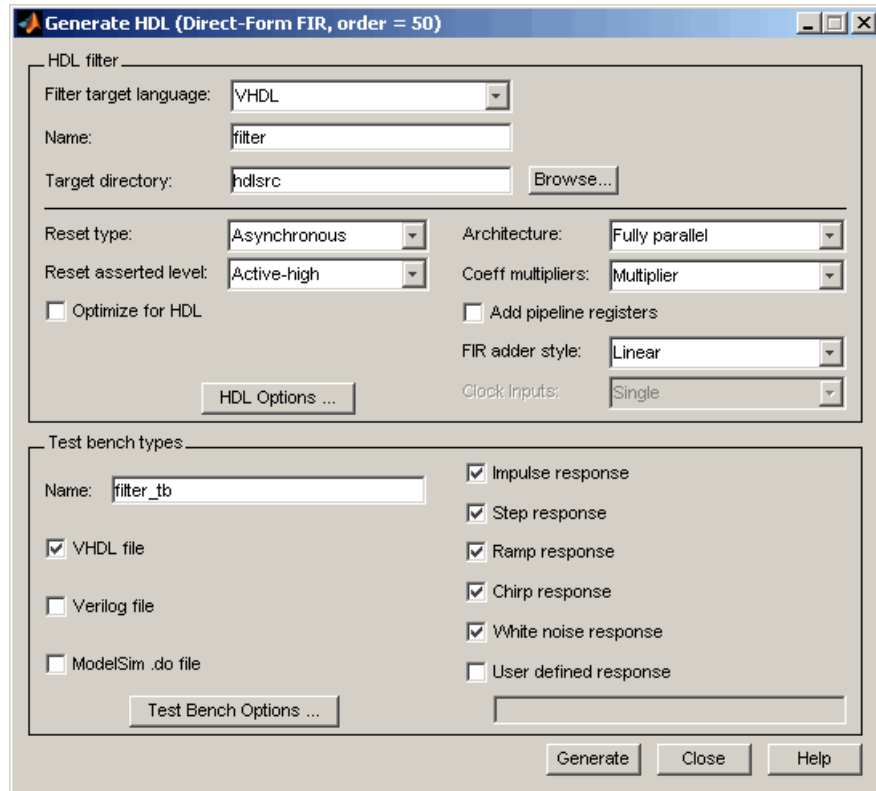
In a partly serial architecture, the filter taps are grouped into a number of serial *partitions*. The taps within each partition execute serially, but the partitions execute in parallel with respect to one another. The outputs of the partitions are summed at the final output.

- *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<br>Details below     | No                                   | Bug Fixes                     | No                                |

New features and changes introduced in this version are

- “Generating Scripts for EDA Tools” on page 31
- “Test Bench Generation Improved for Multirate Filters” on page 31

### 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<br>Details below     | Yes—Details labeled as <b>Compatibility Considerations</b> , below. See also Summary. | Bug Fixes                     | No                                |

New features and changes introduced in this version are

- “Additional Multirate and Discrete Filter Types Supported” on page 32
- “Code Generation Support for Interpolating Filters in Cascades” on page 33
- “InitializeRealSignals Property and GUI Option Removed” on page 33

### 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

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<br>Details below     | Yes—Details labeled as <b>Compatibility Considerations</b> , below. See also Summary. | No                            | No                                |

New features and changes introduced in this version are

- “Multirate Filter Support” on page 34
- “Cascade Filter Support” on page 34
- “CoeffPrefix Property Replaces CoeffName Property” on page 35

### 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  |
|---|---|
| <b>Latest Version</b><br><b>V2.0 (R2007a)</b> | None  |
| V1.5<br>(R2006b)                              | See the <b>Compatibility Considerations</b> subheading for this new feature or change: <ul style="list-style-type: none"> <li>• “ResetValue Property Merged with ResetAssertedLevel Property” on page 25</li> <li>• “Clock EnableValue for Test Benches Always Active-High” on page 26</li> </ul> |
| V1.4<br>(R2006a)                              | See the <b>Compatibility Considerations</b> subheading for this new feature or change: <ul style="list-style-type: none"> <li>• “Rounding Behavior in Generated HDL Code” on page 30</li> </ul>   |
| V1.3<br>(R14SP3)                              | None  |



| <b>Version (Release)</b> | <b>New Features and Changes with Version Compatibility Impact</b>   |
|--------------------------|---|
| V1.2<br>(R14SP2)         | See the <b>Compatibility Considerations</b> subheading for this new feature or change: <ul style="list-style-type: none"><li>• “InitializeRealSignals Property and GUI Option Removed” on page 33</li></ul> |
| V1.1<br>(R14SP1)         | See the <b>Compatibility Considerations</b> subheading for this new feature or change: <ul style="list-style-type: none"><li>• “CoeffPrefix Property Replaces CoeffName Property” on page 35</li></ul>      |