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 "Using 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 V2.2 (R2008a)	Yes Details	Yes Summary	No	Printable Release Notes: PDF Current product documentation
V2.1 (R2007b)	Yes Details	Yes Summary	No	No
V2.0 (R2007a)	Yes Details	No	No	No
V1.5 (R2006b)	Yes Details	Yes Summary	Bug Reports	No
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

Using Release Notes

Use release notes when upgrading to a newer version to learn about:

- New features

- Changes
- Potential impact on your existing files and practices

Review the release notes for other MathWorks™ products required for this product (for example, MATLAB® or Simulink®) for enhancements, bugs, and compatibility considerations that also might impact you.

If you are upgrading from a software version other than the most recent one, review the 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 release notes for V1.1 and V1.2.

What's in the Release Notes

New Features and Changes

- New functionality
- Changes to existing functionality

Version Compatibility Considerations

When a new feature or change introduces a reported incompatibility between versions, the **Compatibility Considerations** subsection explains the impact.

Compatibility issues reported after the product is released appear under Bug Reports at the MathWorks Web site. Bug fixes can sometimes result in incompatibilities, so you should also review the fixed bugs in Bug Reports for any compatibility impact.

Fixed Bugs and Known Problems

The MathWorks offers a user-searchable Bug Reports database so you can view Bug Reports. The development team updates this database at release time and as more information becomes available. This includes provisions for any known workarounds or file replacements. Information is available for bugs existing in or fixed in Release 14SP2 or later. Information is not available for all bugs in earlier releases.

Access Bug Reports using your MathWorks Account.

About Functions and Properties Being Removed

This section lists functions or properties removed or in the process of being removed. Functions and properties typically go through several stages across multiple releases before being completely removed. This provides time for you to make adjustments to your code.

- Announcement — The Release Notes announce the planned removal, but there are no functional changes; the function runs as it did before.
- Warning — When you run the function, it displays a warning message indicating it will be removed in a future release; otherwise the function runs as it did before.
- Error — When you run the function, it produces an error. The error message indicates the function was removed and suggests a replacement function, if one is available.
- Removal — When you run the function, it fails. The error message is the standard message when MATLAB does not recognize an entry.

Functions and properties might be in a stage for one or more releases before moving to another stage. Functions and properties are listed in the Functions and Properties Being Removed section only when they enter a new stage and their behavior changes. For example, if a function displayed a warning in the previous release and errors in this release, it appears on the list. If it continues to display a warning, it does not appear on the list because there was no change between the releases.

Not all functions and properties go through all stages. For example, a function's impending removal might not be announced, but instead, the first notification might be that the function displays a warning.

The Release Notes include actions you can take to mitigate the effects of function or property removal, such as adapting your code to use a replacement function.

Version 2.2 (R2008a) Filter Design HDL Coder™ Software

This table summarizes what's new in Version 2.2 (R2008a).

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	Printable Release Notes: PDF Current product documentation

New features and changes introduced in this version are:

- "Code Generation Support for Multirate Farrow Sample Rate Converter Filters" on page 4
- "Multifile Test Bench Generation" on page 5
- "Additional CLI Properties Supported" on page 5
- "GUI Support for Storage of FIR Filter Coefficients in RAM or Register File " on page 5
- "generatetb Supports Default Specification of Test Bench Type" on page 8
- "Functions and Properties Being Removed" on page 9
- "ModelSim .do Test Bench Option Deprecated" on page 10
- "ScaleWarnBits Property No Longer Supported" on page 11
- "Summary of GUI Enhancements and Revisions" on page 11

Code Generation Support for Multirate Farrow Sample Rate Converter Filters

The coder now supports HDL code generation for multirate Farrow sample rate converter (`mfilt.farrowsrc`) filters.

The coder also supports code generation for cascades that include a `mfilt.farrowsrc` filter, provided that the `mfilt.farrowsrc` filter is in the last position of the cascade.

See "Generating Code for Multirate Farrow Sample Rate Converters" for further information.

Multifile Test Bench Generation

You can now direct the coder to generate separate files for test bench code, helper functions, and test bench data using the following command–line properties:

- `MultifileTestBench`: This property lets you divide the generated test bench into separate files containing helper functions, data, and HDL test bench code. See `MultifileTestBench` for details.
- `TestbenchDataPostfix`: This property lets you specify a suffix added to the test bench data file name when generating a multi-file test bench. See `TestBenchDataPostFix` for details.

Additional CLI Properties Supported

The following command-line properties are supported in the current release:

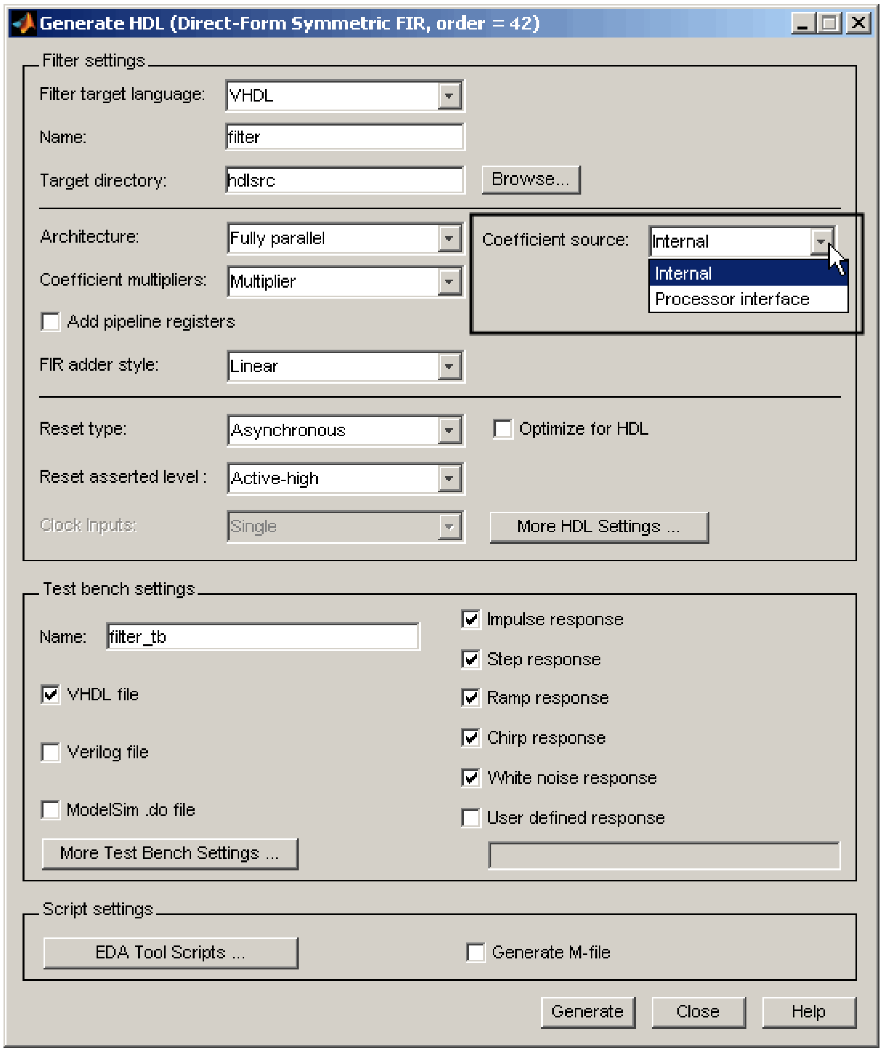
- `HoldInputDataBetweenSamples`: You can apply this property to filters that do not have parallel architectures. In such filters, data can be delivered to the outputs `N` cycles (`N >= 2`) later than the inputs. The `HoldInputDataBetweenSamples` property determines how long (in terms of clock cycles) input data values for these signals are held in a valid state. See `HoldInputDataBetweenSamples` for details.
- `TestBenchReferencePostFix`: This property lets you specify a string appended to the names of reference signals generated in test bench code. See `TestBenchReferencePostFix` for details.

GUI Support for Storage of FIR Filter Coefficients in RAM or Register File

For direct-form FIR filters, the coder now provides two GUI options that let you generate a RAM or register file interface for loading coefficients, and test the interface. These options correspond to the `CoefficientSource` and `TestbenchCoeffStimulus` properties, introduced in the previous release.

The new GUI options are:

- The **Coefficient source** menu on the Generate HDL dialog box (shown in the following figure) lets you select whether coefficients are obtained from the filter object and hard-coded (`Internal`), or from a generated RAM interface (`Processor interface`) The corresponding command-line property is `CoefficientSource`.



Generate HDL (Direct-Form Symmetric FIR, order = 42)

Filter settings

Filter target language: VHDL

Name: filter

Target directory: hdlsrc

Architecture: Fully parallel

Coefficient multipliers: Multiplier

☐ Add pipeline registers

FIR adder style: Linear

Coefficient source: Internal
- Internal
- Processor interface

Reset type: Asynchronous

☐ Optimize for HDL

Reset asserted level : Active-high

Clock Inputs: Single

Test bench settings

Name: filter_tb

✓ VHDL file

☐ Verilog file

☐ ModelSim .do file

✓ Impulse response

✓ Step response

✓ Ramp response

✓ Chirp response

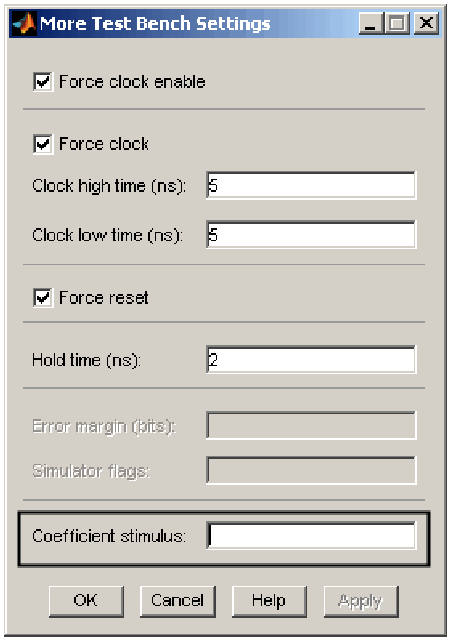
✓ White noise response

☐ User defined response

Script settings

☐ Generate M-file

- The **Coefficient stimulus** option on the More Test Bench Settings dialog box lets you specify how the test bench tests the generated RAM or register file interface. The corresponding command-line property is `TestbenchCoeffStimulus`.



Force clock enable [✓]

Force clock [✓]

Clock high time (ns): 5

Clock low time (ns): 5

Force reset [✓]

Hold time (ns): 2

Error margin (bits):

Simulator flags:

Coefficient stimulus:

For detailed information on these options, see "Specifying Storage of FIR Filter Coefficients in RAM or Register File" in the Filter Design HDL Coder User's Guide.

generatetb Supports Default Specification of Test Bench Type

In previous releases, the `generatetb` function required an explicit argument specifying the test bench type.

In the current release, you can optionally omit the test bench type argument. In this case, the test bench type defaults to the current setting of the `TargetLanguage` property (`'VHDL'` or `'Verilog'`). The `TargetLanguage` property is set by the most recent execution of the `generatehdl` command.

In the following example, `TargetLanguage` is set to `'Verilog'` by the `generatehdl` command. Then, `generatetb` generates a Verilog test bench, by default.

```
>> generatehdl(my_filter,'TargetLanguage','Verilog')
### Starting Verilog code generation process for filter: my_filter
### Starting Verilog code generation process for filter: my_filter
### Generating: H:\hdlsrc\my_filter.v
### Starting generation of my_filter Verilog module
### Starting generation of my_filter Verilog module body
### HDL latency is 2 samples
### Successful completion of Verilog code generation process for filter: my_filter


>> generatetb(my_filter, 'TestBenchName', 'MyFilterTB_V')
### Starting generation of VERILOG Test Bench
### Generating input stimulus
### Done generating input stimulus; length 3312 samples.
### Generating Test bench: H:\hdlsrc\MyFilterTB_V.v
### Please wait .......
### Done generating VERILOG Test Bench
```

See also `generatetb`.

Functions and Properties Being Removed

For more information about the process of removing functions and properties, see “About Functions and Properties Being Removed” in “What’s in the Release Notes” on page 2.

Function or Property Name	What Happens When You Use Function or Property?	Use This Instead	Compatibility Considerations
`'Modelsim'` test bench type argument for `generatetb` function	Warns	No replacement	See “ModelSim .do Test Bench Option Deprecated” on page 10.
`ScaleWarnBits` property	Property is ignored	No replacement	See “ScaleWarnBits Property No Longer Supported” on page 11.

ModelSim .do Test Bench Option Deprecated

The **Modelsim .do file** test bench generation option, and the corresponding`'Modelsim'` test bench type argument for the `generatetb` function, are deprecated in the current release and will not be supported in future releases.

In the current release, the coder displays a warning during test bench generation if this option is specified.

Compatibility Considerations

If your scripts use the `'Modelsim'` test bench type argument for the `generatetb` function, you should remove the `'Modelsim'` argument. The test bench type will then take a default value as described in “generatetb Supports Default Specification of Test Bench Type” on page 8.

See also `generatetb`.

ScaleWarnBits Property No Longer Supported

The `ScaleWarnBits` property is no longer supported. The corresponding GUI option, **Minimum overlap of scale values (bits)** , has been removed from the **Advanced** pane of the More HDL Settings dialog box.

Compatibility Considerations

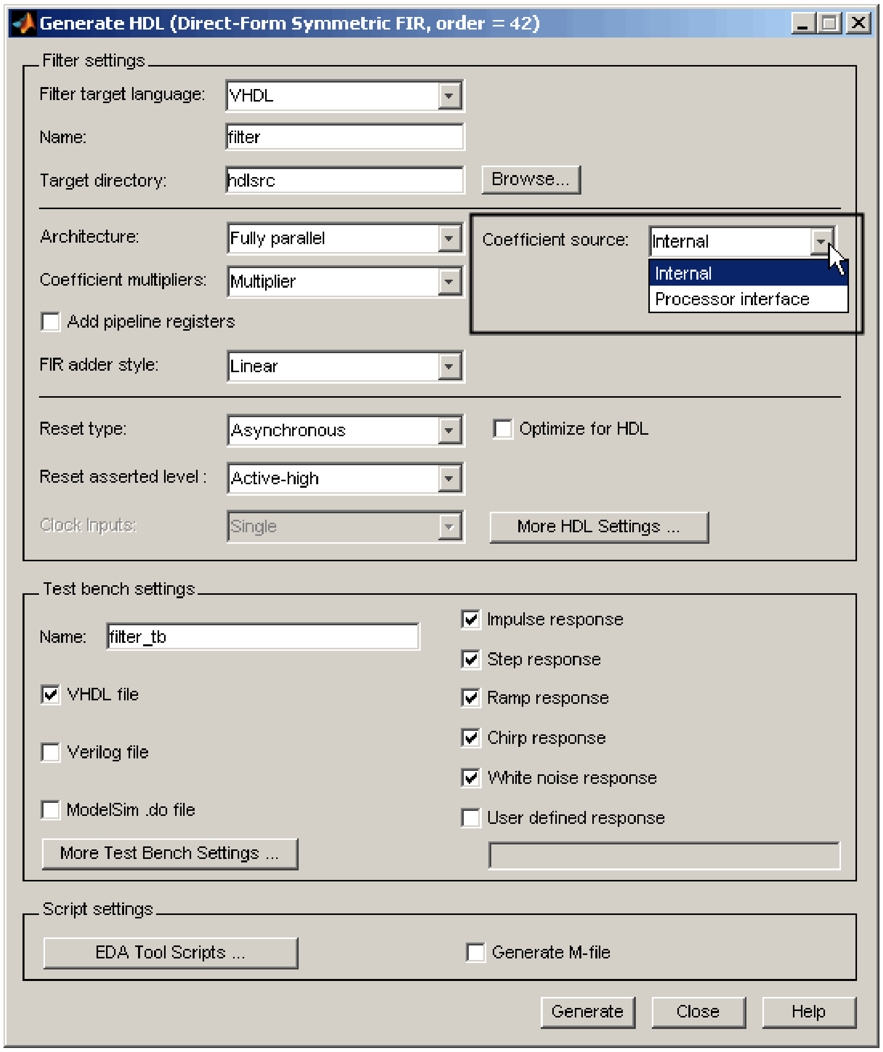
If you have M-files that contain commands that reference the `ScaleWarnBits` property, such references are ignored. Remove references to `ScaleWarnBits` from your code.

Summary of GUI Enhancements and Revisions

This section summarizes revisions and enhancements that have been made to the Filter Design HDL Coder™ GUI.

Generate HDL Dialog Box

The Generate HDL dialog box now includes the **Coefficient source** menu. See "GUI Support for Storage of FIR Filter Coefficients in RAM or Register File " on page 5.


Generate HDL (Direct-Form Symmetric FIR, order = 42)

Filter settings

Filter target language: VHDL

Name: filter

Target directory: hdlsrc

Architecture: Fully parallel

Coefficient multipliers: Multiplier

[ ] Add pipeline registers

FIR adder style: Linear

Coefficient source: Internal

Internal

Processor interface

Reset type: Asynchronous

[ ] Optimize for HDL

Reset asserted level : Active-high

Clock inputs: Single

Test bench settings

Name: filter_tb

[✓] VHDL file

[ ] Verilog file

[ ] ModelSim .do file

[✓] Impulse response

[✓] Step response

[✓] Ramp response

[✓] Chirp response

[✓] White noise response

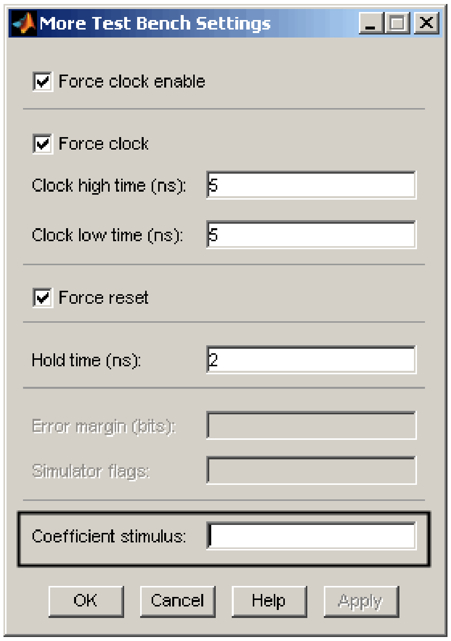
[ ] User defined response

Script settings

[ ] Generate M-file

More Test Bench Settings Dialog Box

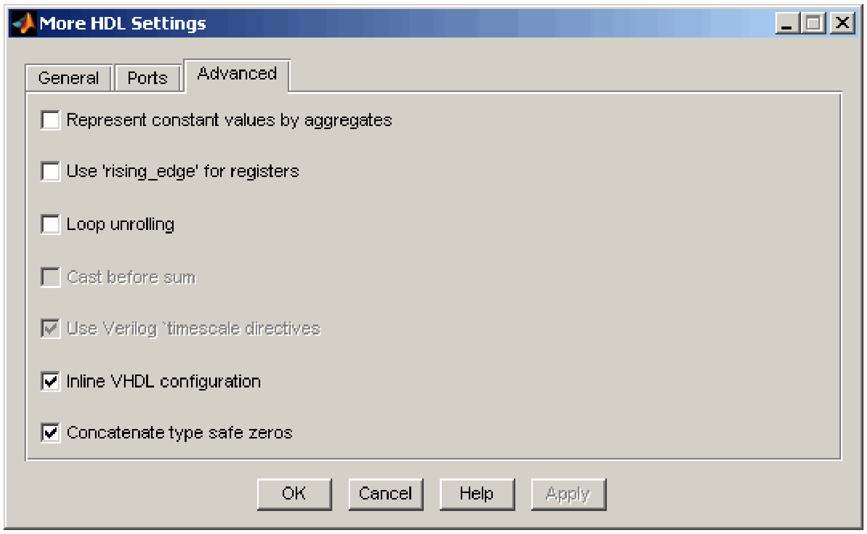
The More Test Bench Settings dialog box now includes the **Coefficient stimulus** option. See "GUI Support for Storage of FIR Filter Coefficients in RAM or Register File " on page 5.



- [✓] Force clock enable
- [✓] Force clock
- Clock high time (ns): 5
- Clock low time (ns): 5
- [✓] Force reset
- Hold time (ns): 2
- Error margin (bits):
- Simulator flags:
- Coefficient stimulus:

More HDL Settings Dialog Box

The **Minimum overlap of scale values (bits)** option has been removed from the **Advanced** pane of the More HDL Settings dialog box. (See "ScaleWarnBits Property No Longer Supported" on page 11.) The following figure shows the default settings for the **Advanced** pane.


More HDL Settings

- [ ] Represent constant values by aggregates
- [ ] Use 'rising_edge' for registers
- [ ] Loop unrolling
- [ ] Cast before sum
- [✓] Use Verilog `timescale directives
- [✓] Inline VHDL configuration
- [✓] Concatenate type safe zeros

Version 2.1 (R2007b) Filter Design HDL Coder™ Software

This table summarizes what's new in Version 2.1 (R2007b).

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:

- "Storage of FIR Filter Coefficients in RAM or Register File" on page 15
- "Generate M-file Option Captures GUI Settings to Generated M-file" on page 16
- "Fixed-point Round Mode Supported for HDL Code Generation" on page 18
- "New Code Generation Properties Supported" on page 18
- "Default Hardware Target for Synthesis Scripts Updated to Virtex-4 " on page 19
- "Summary of GUI Enhancements and Revisions" on page 21

Storage of FIR Filter Coefficients in RAM or Register File

In previous releases, the coder obtained filter coefficients from the filter object and directly coded them into the generated code. An HDL filter realization generated for a particular set of coefficients could not be used with a different set of coefficients.

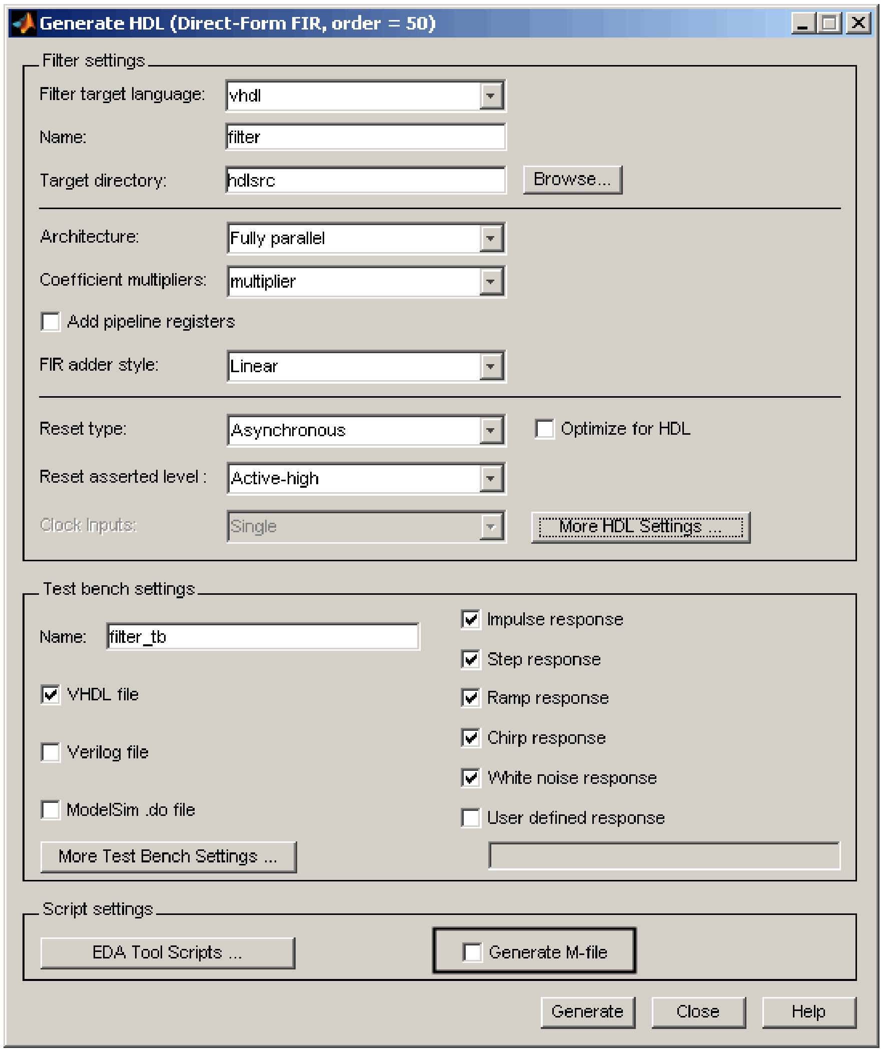
For direct-form FIR filters, the coder now provides two command-line properties that let you generate a RAM or register file interface for loading coefficients, and test the interface. These properties are:

- `CoefficientSource`: This property specifies whether coefficients are directly coded, or stored in RAM or register file.
- `TestbenchCoeffStimulus`: This property specifies how the test bench tests the generated RAM or register file interface and the performance of the filter.

See "Specifying Storage of FIR Filter Coefficients in RAM or Register File" for a detailed description of this feature.

Generate M-file Option Captures GUI Settings to Generated M-file

The new **Generate M-file** option of the Generate HDL dialog box makes command-line scripting of HDL filter code and test bench generation easier. The following figure shows the new option.


Generate HDL (Direct-Form FIR, order = 50)

Filter settings

Filter target language: vhdl

Name: filter

Target directory: hdlsrc

Architecture: Fully parallel

Coefficient multipliers: multiplier

☐ Add pipeline registers

FIR adder style: Linear

Reset type: Asynchronous

☐ Optimize for HDL

Reset asserted level : Active-high

Clock inputs: Single

Test bench settings

Name: filter_tb

☑ VHDL file

☐ Verilog file

☐ ModelSim .do file

☑ Impulse response

☑ Step response

☑ Ramp response

☑ Chirp response

☑ White noise response

☐ User defined response

Script settings

☐ Generate M-file

By default, **Generate M-file** is cleared.

When you select **Generate M-file** and generate code, the coder captures all nondefault HDL code and test bench generation settings from the GUI and writes out an M-file that you can use to regenerate HDL code for the filter.

For detailed information, see "Capturing Code Generation Settings to an M-File".

Fixed-point Round Mode Supported for HDL Code Generation

The coder now supports the fixed-point `Round` rounding mode for HDL code generation. This rounding mode behaves identically to the MATLAB® `round` function.

Compatibility Considerations

In previous releases, the coder did not support this rounding behavior in generated HDL code. When generating code from a filter that had the `RoundMode` property set to `Round`, the coder used `nearest` rounding mode instead. (See "Rounding Behavior in Generated HDL Code" on page 51 for a detailed description of the rounding behavior in previous releases.)

If you have scripts or other programs that generate HDL code from filter objects that have the `RoundMode` property set to `Round`, the behavior of your generated HDL filters may differ from results obtained from previous releases. You may want to update your scripts accordingly.

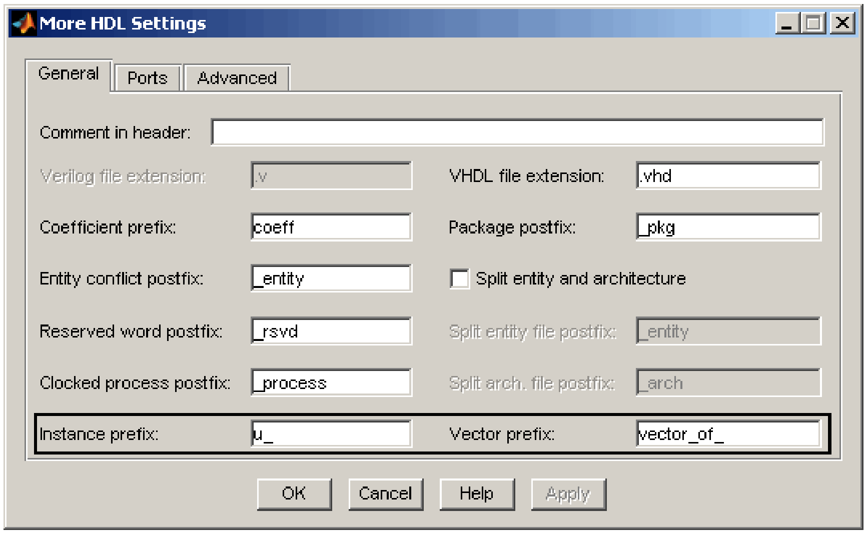
New Code Generation Properties Supported

The coder supports two new code generation properties:

- `InstancePrefix`: This property specifies a string to be prefixed to component instance names in generated code. The default string is `u_`.
- `VectorPrefix`: This property specifies a string to be prefixed to vector names in generated VHDL code. The default string is `vector_of_`.

Note `VectorPrefix` is supported only for VHDL code generation

You can view and edit these new properties via the **Instance prefix** and **Vector prefix** edit fields on the **General** pane of the More HDL Settings dialog box, shown in the following figure.



Comment in header:
Verilog file extension: .v
VHDL file extension: .vhd
Coefficient prefix: coeff
Package postfix: _pkg
Entity conflict postfix: _entity
Split entity and architecture
Reserved word postfix: _rsvd
Split entity file postfix: _entity
Clocked process postfix: _process
Split arch. file postfix: _arch
Instance prefix: u_
Vector prefix: vector_of_

See also:

- "Setting a Prefix for Component Instance Names"
- "Setting a Prefix for Vector Names"

Default Hardware Target for Synthesis Scripts Updated to Virtex-4

The default hardware target string in generated synthesis scripts now specifies:

- `technology` option: `VIRTEX4`

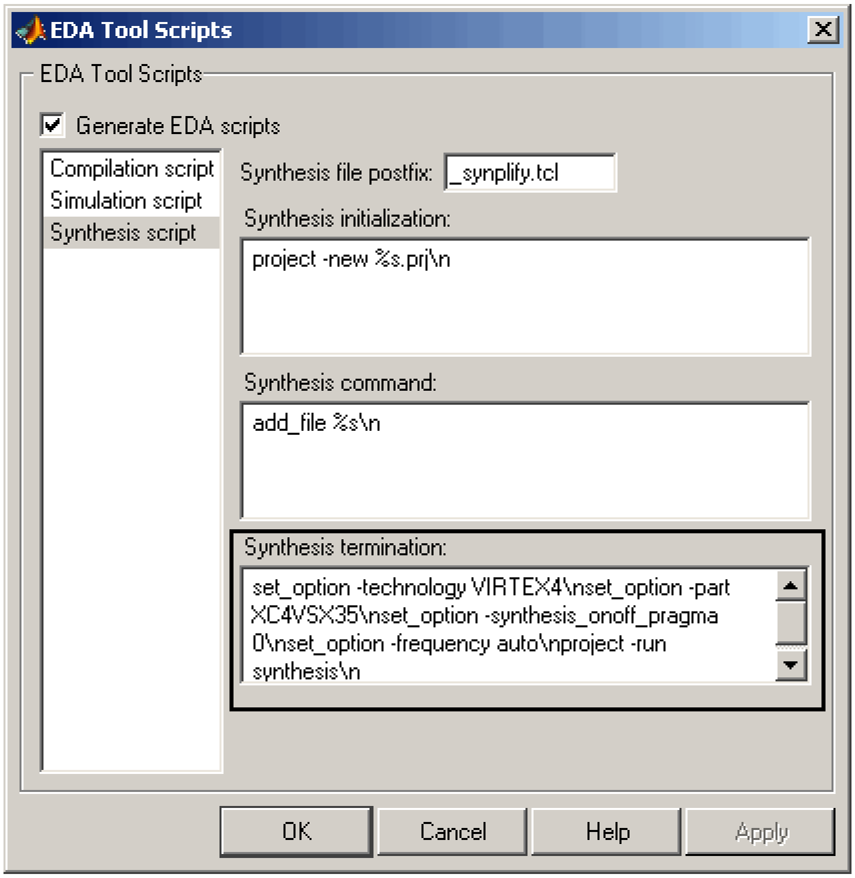
  In previous releases, this option defaulted to `VIRTEX2`.
- `part` option: `XC4VSX35`

  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.



[✓] Generate EDA scripts

Synthesis file postfix: _synplify.tcl

Synthesis initialization: project -new %s.prj\n

Synthesis command: add_file %s\n

Synthesis termination: set_option -technology VIRTEX4\nset_option -part XC4VSX35\nset_option -synthesis_onoff_pragma 0\nset_option -frequency auto\nproject -run synthesis\n

See also "Generating Scripts for EDA Tools".

Compatibility Considerations

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

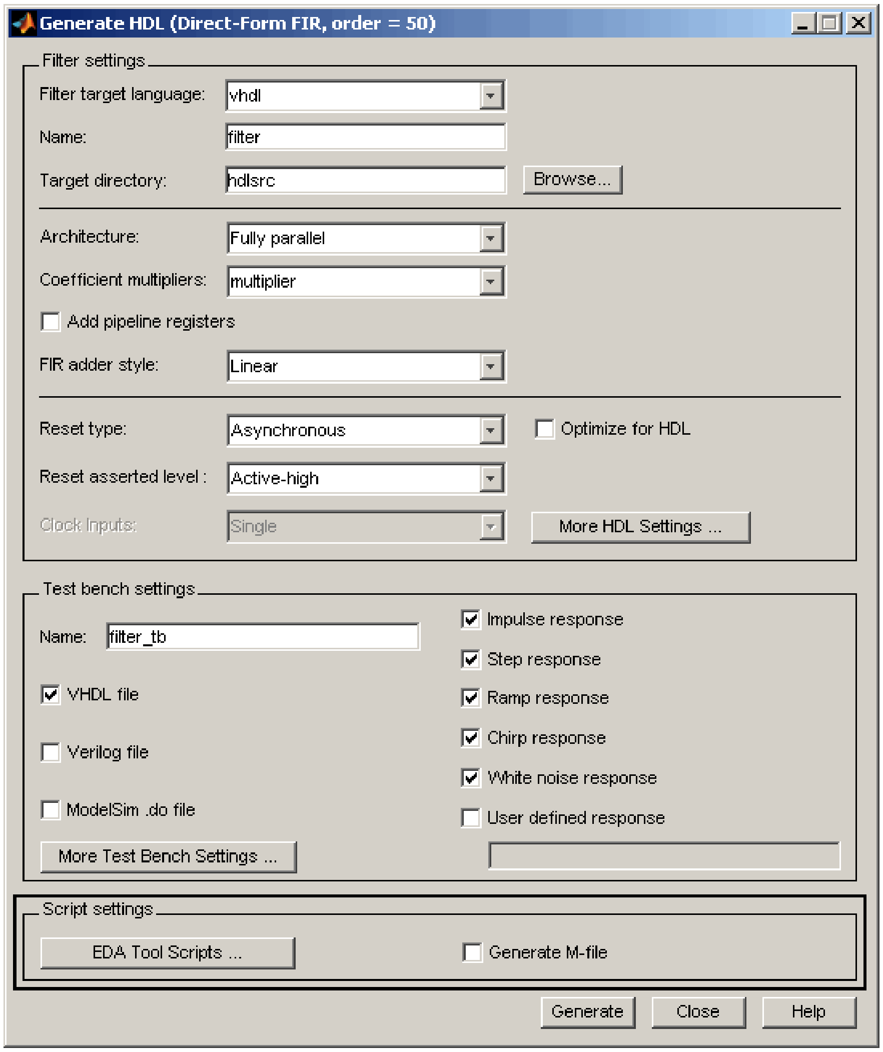
Summary of GUI Enhancements and Revisions

For Version 2.1, revisions and enhancements have been made to the Filter Design HDL Coder™ GUI.

Generate HDL Dialog Box

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

- The new **Generate M-file** option. When you select this option, the code generator captures all nondefault HDL code and test bench generation settings from the GUI and writes out an M-file that you can use to reconstruct the filter and regenerate HDL code. See "Generate M-file Option Captures GUI Settings to Generated M-file" on page 16 for details.
- The **EDA Tool Scripts** button and the **Generate M-file** option are grouped together in a new **Script settings** section.


Generate HDL (Direct-Form FIR, order = 50)

Filter settings

Filter target language: vhdl

Name: filter

Target directory: hdlsrc

Architecture: Fully parallel

Coefficient multipliers: multiplier

☐ Add pipeline registers

FIR adder style: Linear

Reset type: Asynchronous

☐ Optimize for HDL

Reset asserted level : Active-high

Clock Inputs: Single

Test bench settings

Name: filter_tb

☑ VHDL file

☐ Verilog file

☐ ModelSim .do file

☑ Impulse response

☑ Step response

☑ Ramp response

☑ Chirp response

☑ White noise response

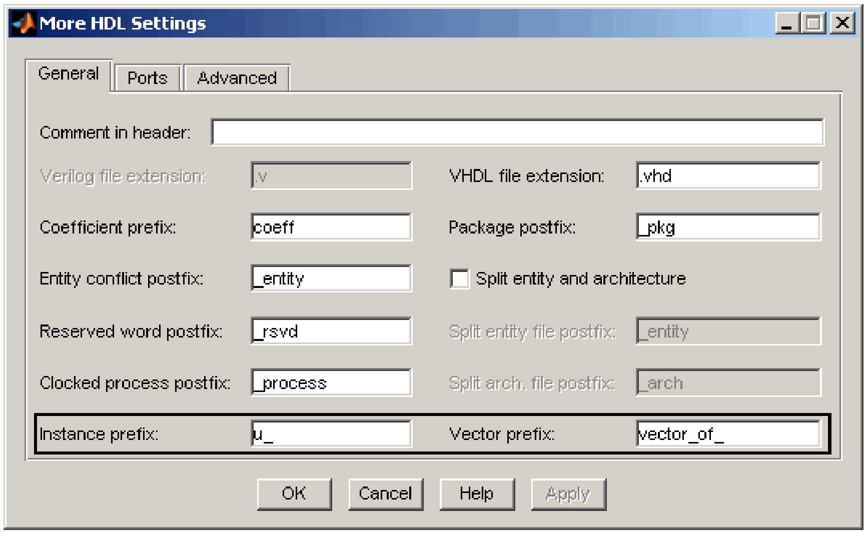
☐ User defined response

Script settings

☐ Generate M-file

More HDL Settings Dialog Box

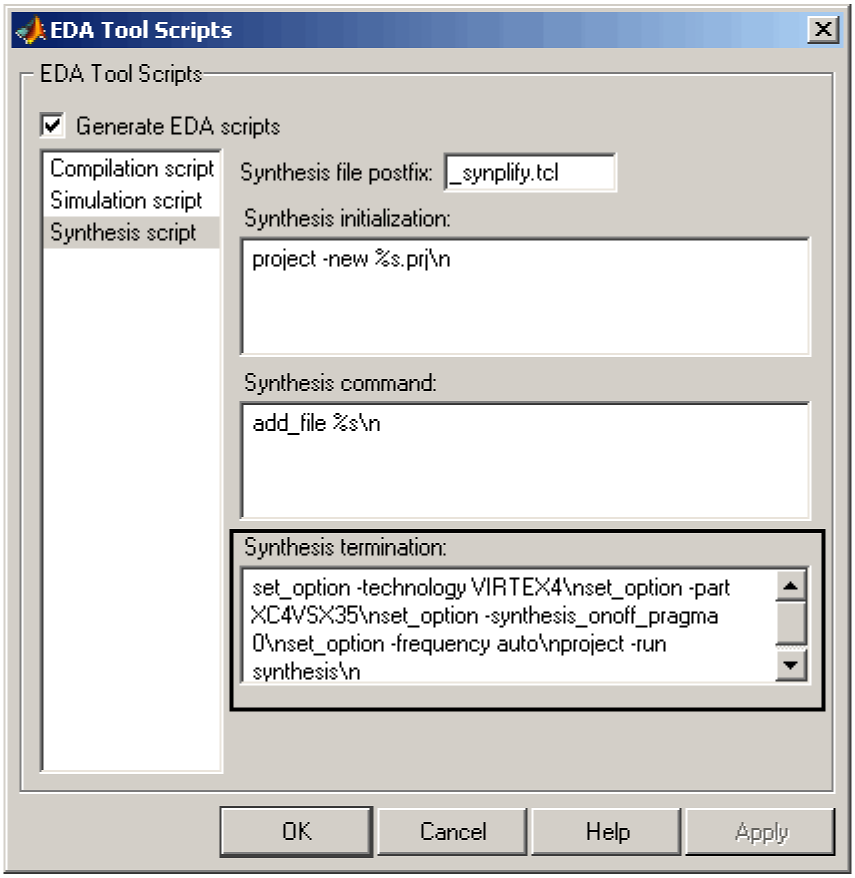
The **General** pane of the More HDL Settings dialog box supports the new **Instance prefix** and **Vector prefix** properties, as shown in the following figure. See “New Code Generation Properties Supported” on page 18 for details.



Comment in header:			
Verilog file extension:	.v	VHDL file extension:	.vhd
Coefficient prefix:	coeff	Package postfix:	_pkg
Entity conflict postfix:	_entity	☐ Split entity and architecture	
Reserved word postfix:	_rsvd	Split entity file postfix:	_entity
Clocked process postfix:	_process	Split arch. file postfix:	_arch
Instance prefix:	u_	Vector prefix:	vector_of_

EDA Tool Scripts Dialog Box

In the EDA Tool Scripts dialog box, the default value for the **Synthesis termination** field has changed (see "Default Hardware Target for Synthesis Scripts Updated to Virtex-4 " on page 19) as shown in the following figure.



EDA Tool Scripts

EDA Tool Scripts

[✓] Generate EDA scripts

Synthesis file postfix: _synplify.tcl

Synthesis initialization:

```
project -new %s.prj\n
```

Synthesis command:

```
add_file %s\n
```

Synthesis termination:

```
set_option -technology VIRTEX4\nset_option -part
XC4VSX35\nset_option -synthesis_onoff_pragma
0\nset_option -frequency auto\nproject -run
synthesis\n
```

Version 2.0 (R2007a) Filter Design HDL Coder™ Software

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 Details below	No	No	No

New features and changes introduced in this version are:

- "Farrow Filter Code Generation" on page 25
- "Code Generation Support for Polyphase Sample Rate Converters (mfilt.firsrc)" on page 26
- "filterbuilder Supports HDL Code Generation" on page 26
- "fdhdltool Function Opens Generate HDL Dialog Box from Command Line" on page 28
- "GUI Enhancements and Revisions" on page 28
- "EDA Tool Scripts Dialog Box" on page 33
- "Multiple Clocks Supported for Multirate Filters with Distributed Arithmetic and Fully Serial Architectures" on page 37

Farrow Filter Code Generation

The coder now 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 filter objects, see the farrow function reference section of the Filter Design Toolbox™ documentation.

The 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 Single-Rate Farrow Filters” in the Filter Design HDL Coder User’s Guide for a complete description of this feature.

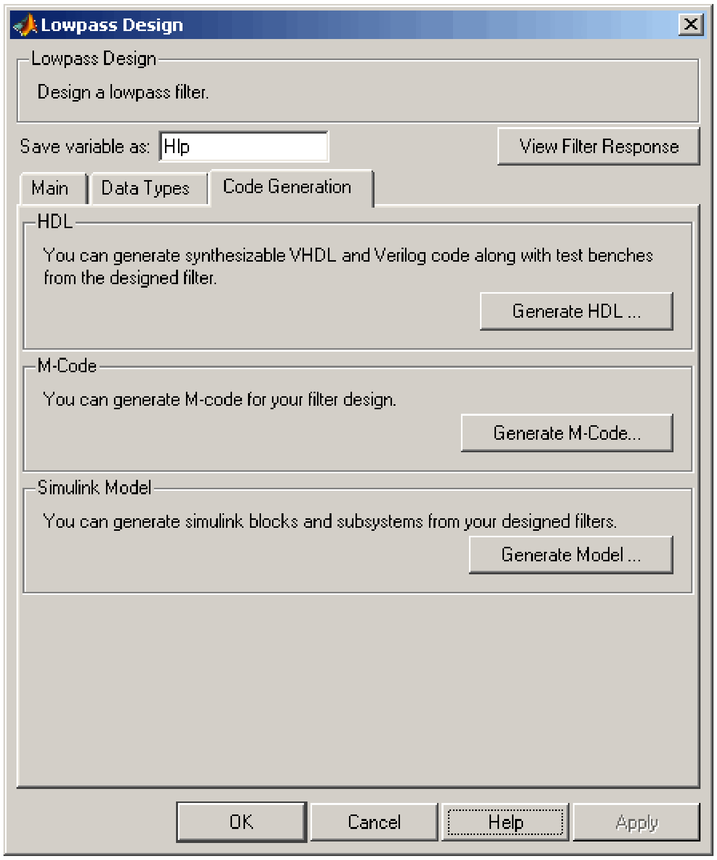
Code Generation Support for Polyphase Sample Rate Converters (mfilt.firsrc)

The coder now 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).


Lowpass Design
Lowpass Design
Design a lowpass filter.
Save variable as: Hlp
View Filter Response
Main
Data Types
Code Generation
HDL
You can generate synthesizable VHDL and Verilog code along with test benches from the designed filter.
Generate HDL ...
M-Code
You can generate M-code for your filter design.
Generate M-Code...
Simulink Model
You can generate simulink blocks and subsystems from your designed filters.
Generate Model ...
OK
Cancel
Help
Apply

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 28 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 Command Line

`fdhdltool` is a convenience function that lets you open the Generate HDL dialog box from the 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);
```

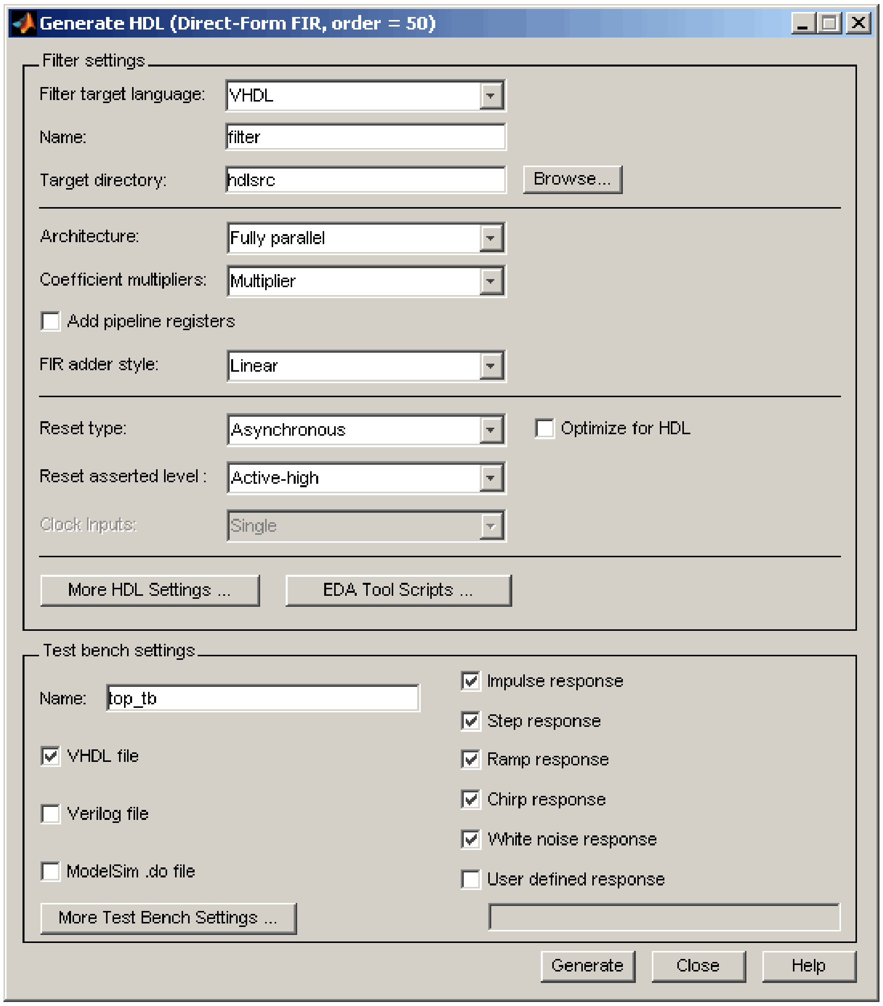
`fdhdltool` operates on a copy of the filter object, rather than the original object in the workspace. Any changes made to the original filter object after `fdhdltool` is invoked will not affect the copy and will not update the Generate HDL dialog box.

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


Generate HDL (Direct-Form FIR, order = 50)

Filter settings

Filter target language: VHDL

Name: filter

Target directory: hdlsrc

Architecture: Fully parallel

Coefficient multipliers: Multiplier

☐ Add pipeline registers

FIR adder style: Linear

Reset type: Asynchronous

☐ Optimize for HDL

Reset asserted level : Active-high

Clock Inputs: Single

Test bench settings

Name: top_tb

☑ VHDL file

☐ Verilog file

☐ ModelSim .do file

☑ Impulse response

☑ Step response

☑ Ramp response

☑ Chirp response

☑ White noise response

☐ User defined response

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

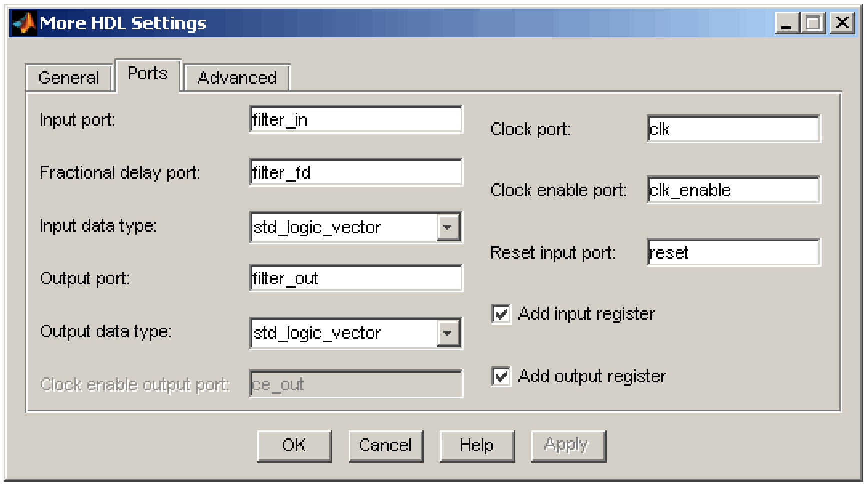
- The new **EDA Tool Scripts** button opens the EDA Tool Scripts dialog box, which lets you set properties that control generation of script files for third-party electronic design automation (EDA) tools. See "EDA Tool Scripts Dialog Box" on page 33.

- 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

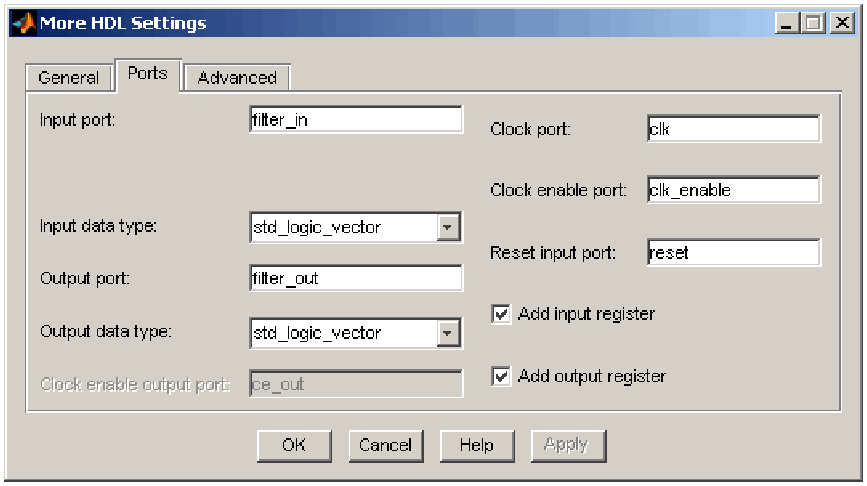
The More HDL Settings dialog box differs slightly from the HDL Settings dialog box, which it replaces.

In the **Ports** pane, when the current filter object is a Farrow filter (see "Farrow Filter Code Generation" on page 25), the new **Fractional delay port** field is displayed, as shown in the following figure.



Input port: filter_in
Fractional delay port: filter_fd
Input data type: std_logic_vector
Output port: filter_out
Output data type: std_logic_vector
Clock enable output port: ce_out
Clock port: clk
Clock enable port: clk_enable
Reset input port: reset
☑ Add input register
☑ Add output register

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



Input port: filter_in

Input data type: std_logic_vector

Output port: filter_out

Output data type: std_logic_vector

Clock enable output port: ce_out

Clock port: clk

Clock enable port: clk_enable

Reset input port: reset

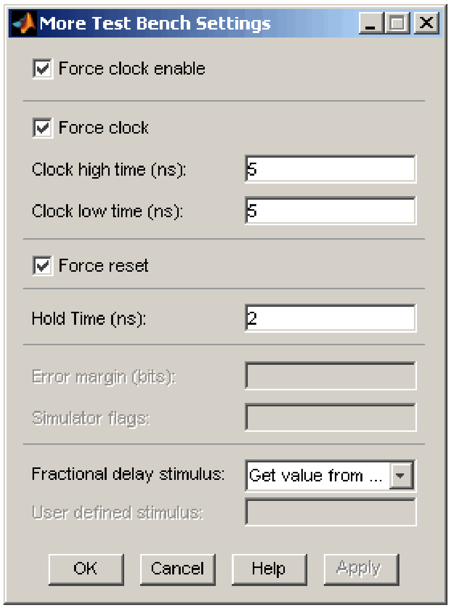
☑ Add input register

☑ Add output register

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 25), the new **Fractional delay stimulus** and **User defined stimulus** options are displayed, as shown in the following figure.



[✓] Force clock enable

[✓] Force clock

Clock high time (ns): 5

Clock low time (ns): 5

[✓] Force reset

Hold Time (ns): 2

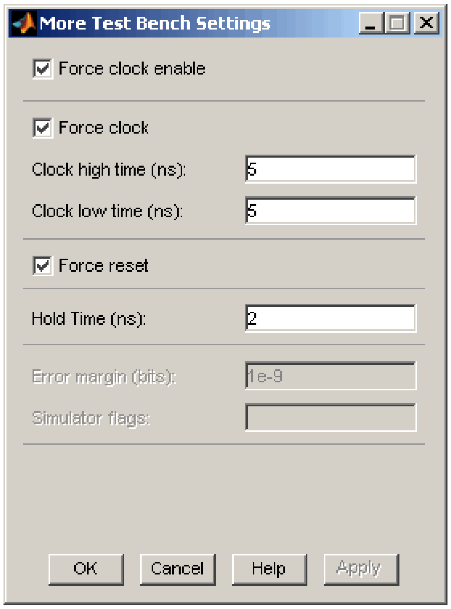
Error margin (bits):

Simulator flags:

Fractional delay stimulus: Get value from ...

User defined stimulus:

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



☑ Force clock enable

☑ Force clock

Clock high time (ns): 5

Clock low time (ns): 5

☑ Force reset

Hold Time (ns): 2

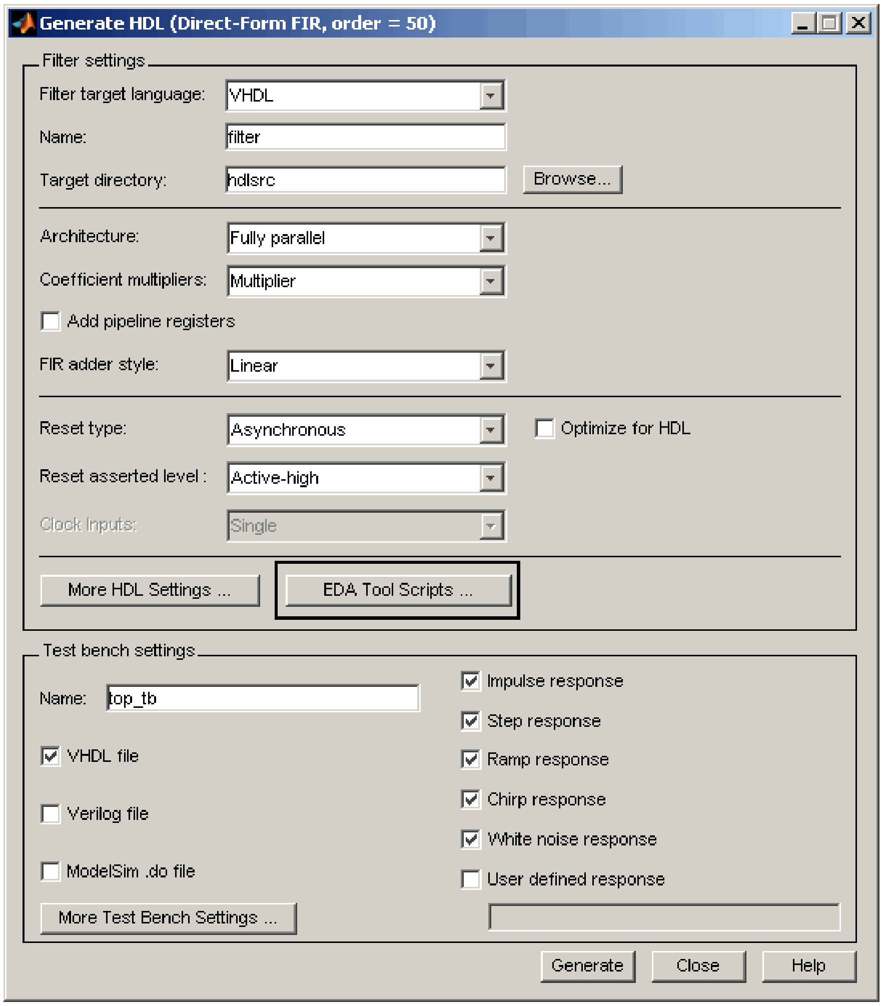
Error margin (bits): 1e-9

Simulator flags:

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).


Generate HDL (Direct-Form FIR, order = 50)

Filter settings

Filter target language: VHDL

Name: filter

Target directory: hdlsrc

Architecture: Fully parallel

Coefficient multipliers: Multiplier

[ ] Add pipeline registers

FIR adder style: Linear

Reset type: Asynchronous

[ ] Optimize for HDL

Reset asserted level : Active-high

Clock Inputs: Single

Test bench settings

Name: top_tb

[✓] VHDL file

[ ] Verilog file

[ ] ModelSim .do file

[✓] Impulse response

[✓] Step response

[✓] Ramp response

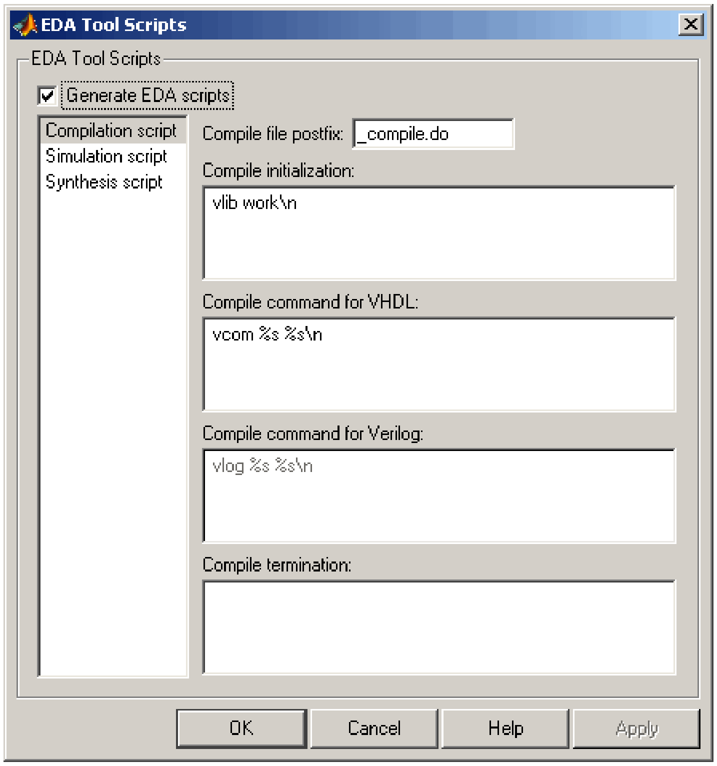
[✓] Chirp response

[✓] White noise response

[ ] User defined response

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.



[✓] Generate EDA scripts

Compile file postfix: _compile.do

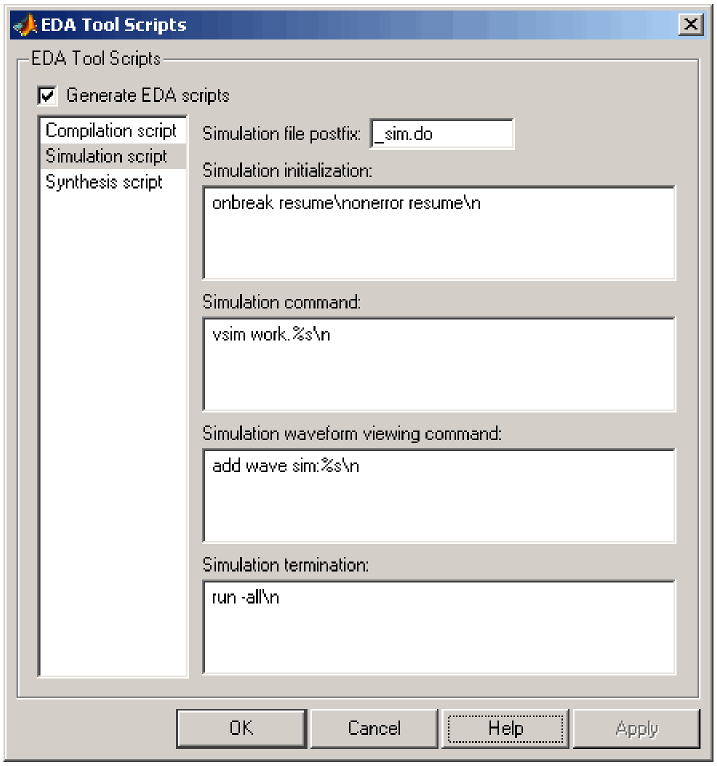
Compile initialization:
```
vlib work\n
```

Compile command for VHDL:
```
vcom %s %s\n
```

Compile command for Verilog:
```
vlog %s %s\n
```

Compile termination:

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



[✓] Generate EDA scripts

Simulation file postfix: _sim.do

Simulation initialization:

```
onbreak resume\nonerror resume\n
```

Simulation command:

```
vsim work.%s\n
```

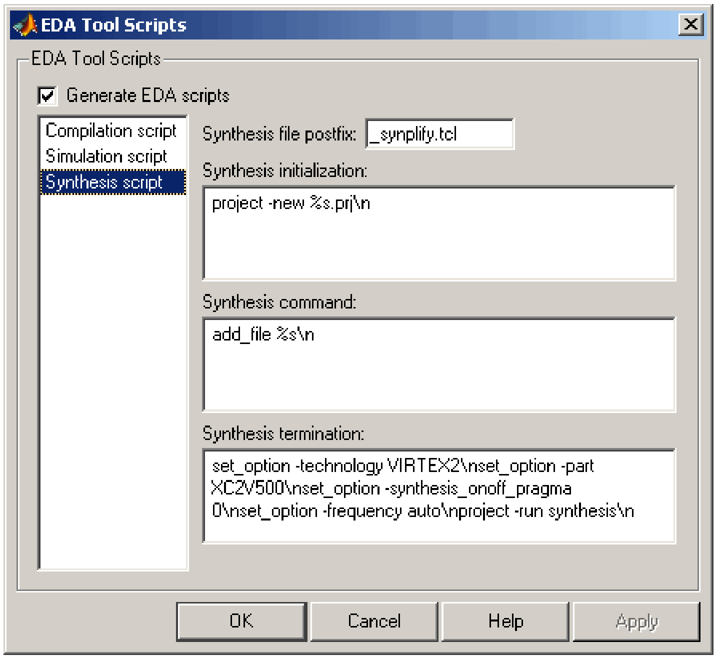
Simulation waveform viewing command:

```
add wave sim:%s\n
```

Simulation termination:

```
run -all\n
```

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



[✓] Generate EDA scripts

Synthesis file postfix: _synplify.tcl

Synthesis initialization:

```
project -new %s.prj\n
```

Synthesis command:

```
add_file %s\n
```

Synthesis termination:

```
set_option -technology VIRTEX2\nset_option -part
XC2V500\nset_option -synthesis_onoff_pragma
0\nset_option -frequency auto\nproject -run synthesis\n
```

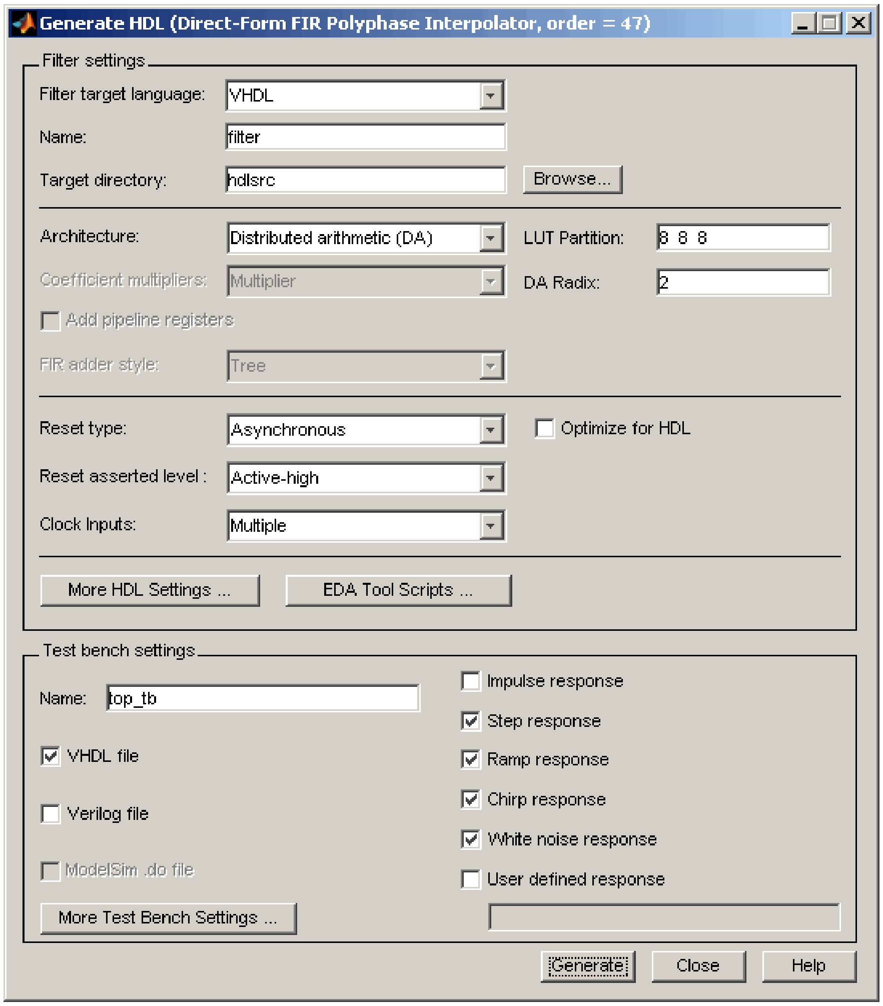
See "Generating Scripts for EDA Tools" for a detailed description of script generation options.

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.

The coder now 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	Browse...	
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	☐ Impulse response
	☑ Step response
☑ VHDL file	☑ Ramp response
☐ Verilog file	☑ Chirp response
	☑ White noise response
☐ ModelSim .do file	☐ 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™ Software

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	No

New features and changes introduced in this version are

- "Distributed Arithmetic Support for FIR Filters" on page 40
- "Multirate Support for Fully Serial Architectures" on page 42
- "Generate HDL Dialog Box Supports All Parallel and Serial Architecture Options" on page 43
- "Enhanced Code Generation for Symmetric Multirate FIR Filters" on page 46
- "EDAScriptGeneration Property Added" on page 46
- "ResetValue Property Merged with ResetAssertedLevel Property" on page 46
- "Clock EnableValue for Test Benches Always Active-High" on page 47

Distributed Arithmetic Support for FIR Filters

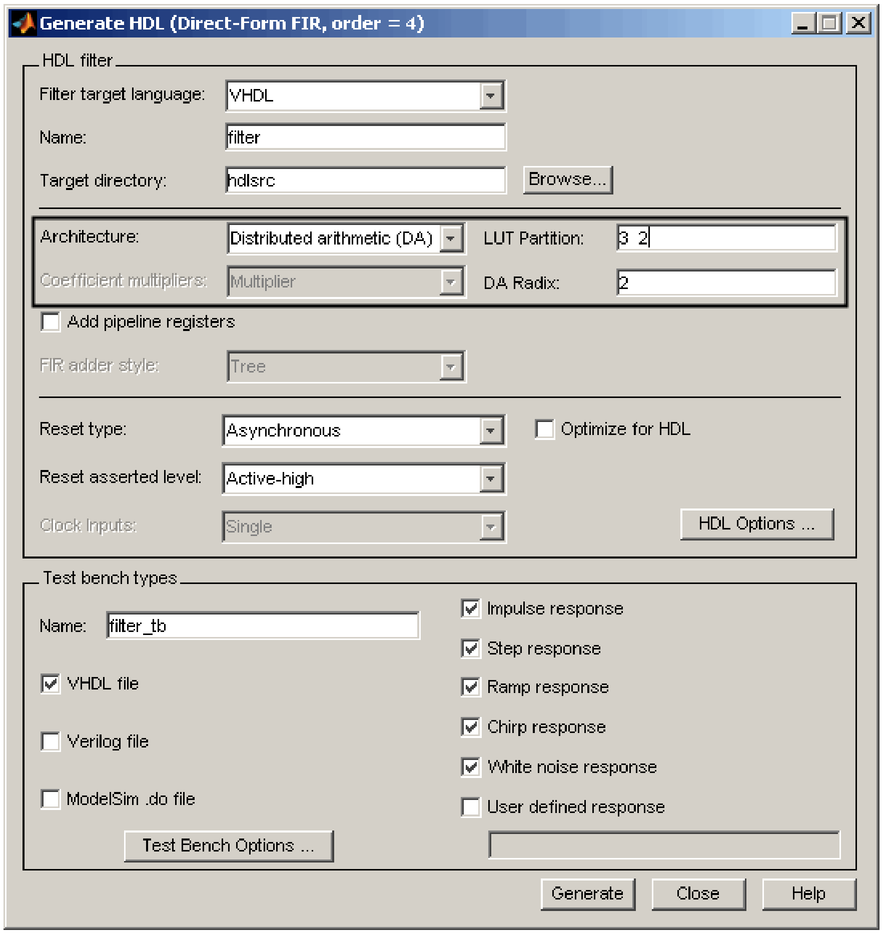
The coder now supports Distributed Arithmetic (DA) in HDL code generated for several single-rate and multirate FIR filter structures. DA is a widely-used technique for implementing sum of products computations without use of multipliers. Designers frequently use DA to build efficient Multiply-Accumulate 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™ documentation 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

The coder adds support for generation of fully serial architectures for the following multirate filter types:

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

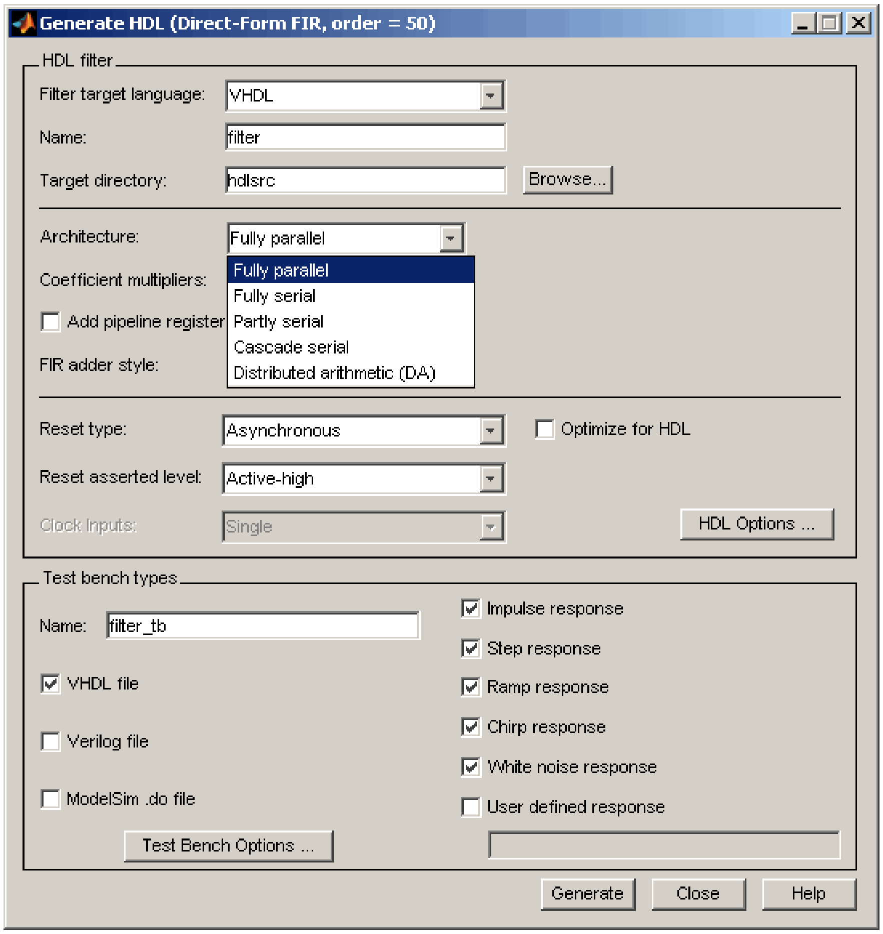
The following table summarizes the filter types that are available for parallel and serial architecture choices. See "Speed vs. Area Optimizations for FIR Filters" in the Filter Design HDL Coder User's Guide for a full description of these options.

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`).

The Generate HDL dialog box now 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.



Generate HDL (Direct-Form FIR, order = 50)

HDL filter

Filter target language: VHDL

Name: filter

Target directory: hdlsrc

Architecture: Fully parallel

Coefficient multipliers:

[ ] Add pipeline register

FIR adder style:

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

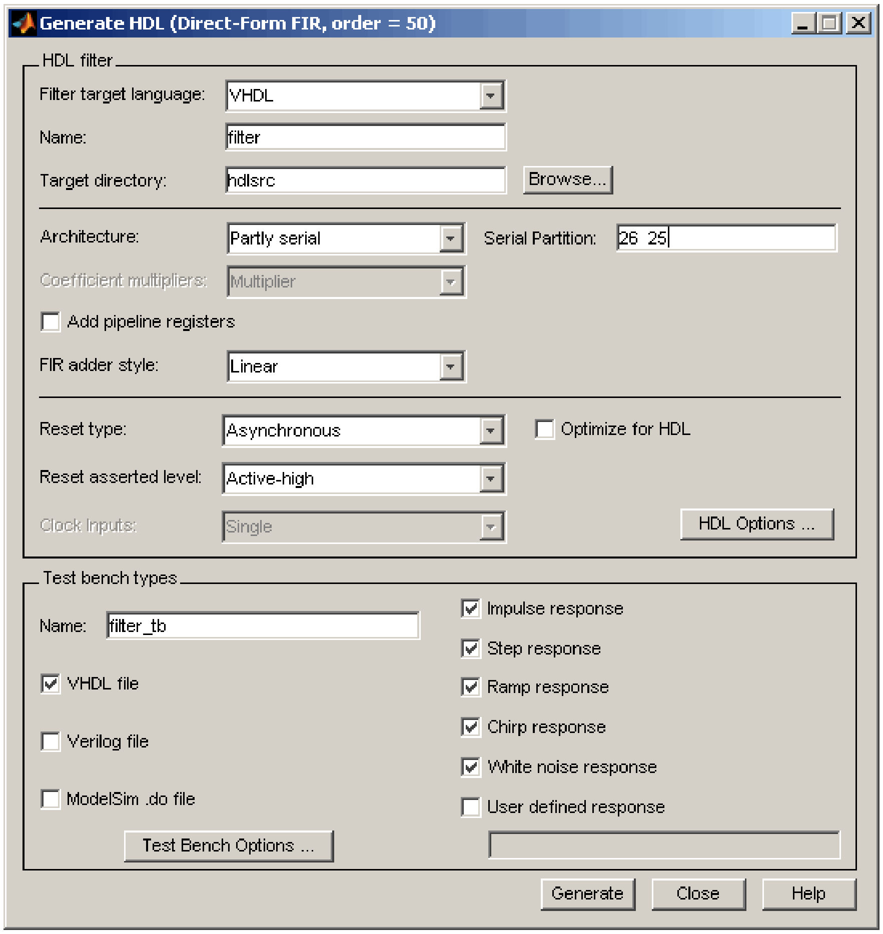
[✓] Ramp response

[✓] Chirp response

[✓] White noise response

[ ] User defined response

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.



Generate HDL (Direct-Form FIR, order = 50)

HDL filter

Filter target language: VHDL

Name: filter

Target directory: hdlsrc

Browse...

Architecture: Partly serial

Serial Partition: 26 25

Coefficient multipliers: Multiplier

[ ] Add pipeline registers

FIR adder style: Linear

Reset type: Asynchronous

[ ] Optimize for HDL

Reset asserted level: Active-high

Clock inputs: Single

HDL Options ...

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

Test Bench Options ...

Generate Close Help

Note The **Architecture** pop-up menu also includes the new `Distributed arithmetic (DA)` option (see "Distributed Arithmetic Support for FIR Filters" on page 40).

Enhanced Code Generation for Symmetric Multirate FIR Filters

In this release, the 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™ Software

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" on page 48
- "Code Generation Support for Delay Filter" on page 50
- "Rounding Behavior in Generated HDL Code" on page 51

Speed vs. Area Tradeoff Options for FIR Filters

The coder now 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:

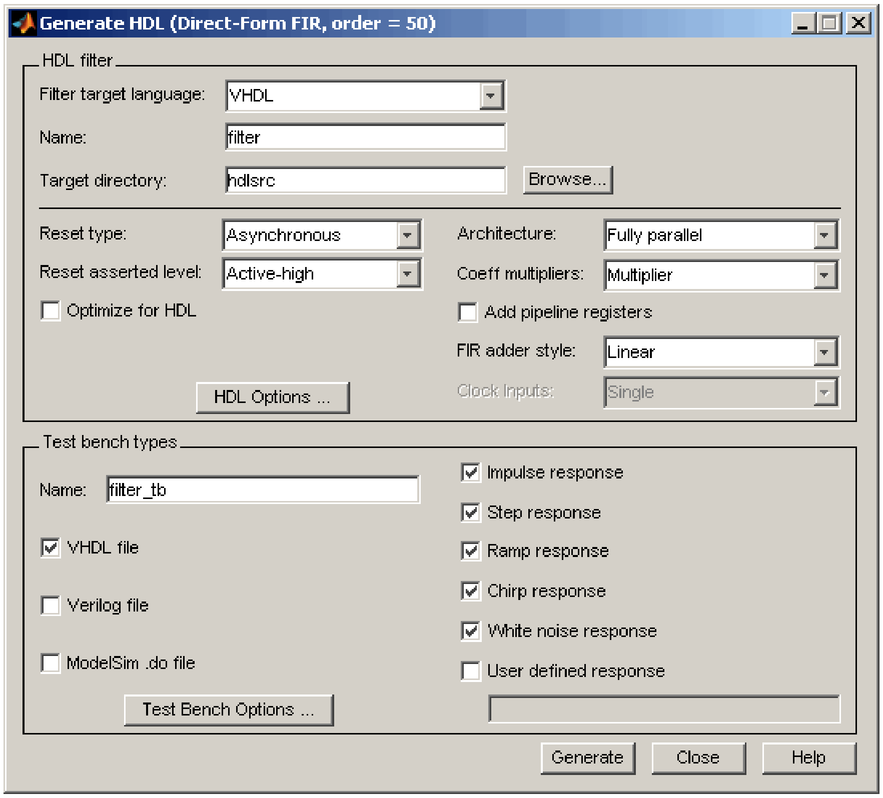
- *Fully parallel*: This is the default option. A fully parallel architecture uses a dedicated multiplier and adder for each filter tap; all taps execute in parallel. A fully parallel architecture is optimal for speed. However, it requires more multipliers and adders than a serial architecture, and therefore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



Generate HDL (Direct-Form FIR, order = 50)

HDL filter

Filter target language: VHDL

Name: filter

Target directory: hdlsrc

Browse...

Reset type: Asynchronous

Reset asserted level: Active-high

[ ] Optimize for HDL

Architecture: Fully parallel

Coeff multipliers: Multiplier

[ ] Add pipeline registers

FIR adder style: Linear

Clock inputs: Single

HDL Options ...

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

Test Bench Options ...

Generate Close Help

The new options are supported for the following filter types:

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

Code Generation Support for Delay Filter

The 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, the 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`, The 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™ Software

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 52
- "Test Bench Generation Improved for Multirate Filters" on page 52

Generating Scripts for EDA Tools

The 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" on page 53
- "Code Generation Support for Interpolating Filters in Cascades" on page 54
- "InitializeRealSignals Property and GUI Option Removed" on page 54

Additional Multirate and Discrete Filter Types Supported

The 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 coder 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 coder now 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.

Compatibility Summary for Filter Design HDL Coder™ Software

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 **V2.2 (R2008a)**	See the **Compatibility Considerations** subheading for this new feature or change: • "ScaleWarnBits Property No Longer Supported" on page 11 • "ModelSim .do Test Bench Option Deprecated" on page 10
V2.1 (R2007b)	See the **Compatibility Considerations** subheading for this new feature or change: • "Fixed-point Round Mode Supported for HDL Code Generation" on page 18 • "Default Hardware Target for Synthesis Scripts Updated to Virtex-4 " on page 19
V2.0 (R2007a)	None

Version (Release)	New Features and Changes with Version Compatibility Impact
V1.5 (R2006b)	See the **Compatibility Considerations** subheading for this new feature or change: • "ResetValue Property Merged with ResetAssertedLevel Property" on page 46 • "Clock EnableValue for Test Benches Always Active-High" on page 47
V1.4 (R2006a)	See the **Compatibility Considerations** subheading for this new feature or change: • "Rounding Behavior in Generated HDL Code" on page 51
V1.3 (R14SP3)	None
V1.2 (R14SP2)	See the **Compatibility Considerations** subheading for this new feature or change: • "InitializeRealSignals Property and GUI Option Removed" on page 54