Filter Design HDL Coder™ Release Notes

How to Contact MathWorks

`www.mathworks.com`	Web
`comp.soft-sys.matlab`	Newsgroup
`www.mathworks.com/contact_TS.html`	Technical Support

`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	Sales, pricing, and general information

508-647-7000 (Phone)

508-647-7001 (Fax)

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098

For contact information about worldwide offices, see the MathWorks Web site.

Filter Design HDL Coder™ Release Notes

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See `www.mathworks.com/trademarks` for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

MathWorks products are protected by one or more U.S. patents. Please see `www.mathworks.com/patents` for more information.

Contents

Summary by Version

This table provides quick access to what's new in each version. For clarification, see "Using Release Notes" on page 2.

Version (Release)	New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Latest Version V2.7 (R2010b)	Yes Details	No	No	YesPrintable Release Notes: PDF Current product documentation
V2.6 (R2010a)	Yes Details	Yes Summary	No	No
V2.5 (R2009b)	Yes Details	Yes Summary	No	No
V2.4 (R2009a)	Yes Details	Yes Summary	No	No
V2.3 (R2008b)	Yes Details	Yes Summary	No	No
V2.2 (R2008a)	Yes Details	Yes Summary	No	No
V2.1 (R2007b)	Yes Details	Yes Summary	No	No
V2.0 (R2007a)	Yes Details	No	No	No

Version (Release)	New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
V1.5 (R2006b)	Yes Details	Yes Summary	Bug Reports	No
V1.4 (R2006a)	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®). Determine if enhancements, bugs, or compatibility considerations in other products impact you.

If you are upgrading from a software version other than the most recent one, review the current release notes and all interim versions. For example, when you upgrade from V1.0 to V1.2, review the release notes for V1.1 and V1.2.

What Is 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 release appear under Bug Reports at the MathWorks Web site. Bug fixes can sometimes result in incompatibilities, so review the fixed bugs in Bug Reports for any compatibility impact.

Fixed Bugs and Known Problems

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. Bug Reports include 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 2.7 (R2010b) Filter Design HDL Coder Software

This table summarizes what's new in Version 2.7 (R2010b)

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

New features and changes introduced in this version are:

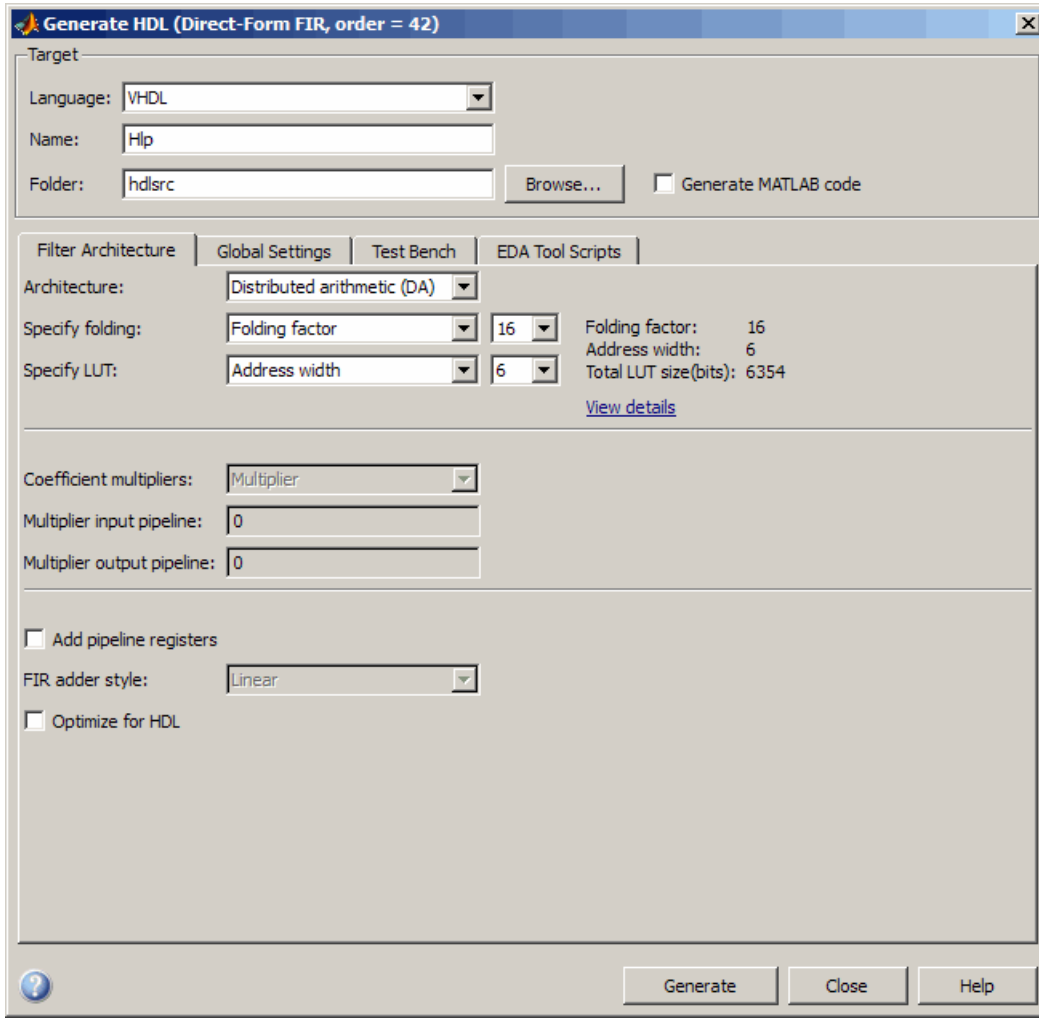
Improved GUI Support for Distributed Arithmetic Architectures

The coder now lets you specify Distributed Arithmetic (DA) architectures for FIR filters more flexibly. In addition, the Generate HDL dialog box now includes informational displays and a hyperlink that lets you display a report showing DA architectural details

In previous releases, you defined a DA architecture in terms of two parameters:

- **LUT Partition**: a vector specifying the number and size of LUT partitions.
- **DA Radix**: the number of bits processed simultaneously, expressed as a power of 2.

In release 2010b you can choose from an expanded set of DA architecture options. The following figure shows the default options.



The **Specify folding** pulldown menu lets you select one of:

- `Folding factor` (default): Select from the pulldown menu to the right of **Specify folding**. The menu contains an exhaustive list of possible folding factors for the filter.
- `DA radix`: Select the number of bits processed simultaneously, expressed as a power of 2.

The **Specify LUT** pulldown menu lets you select one of:

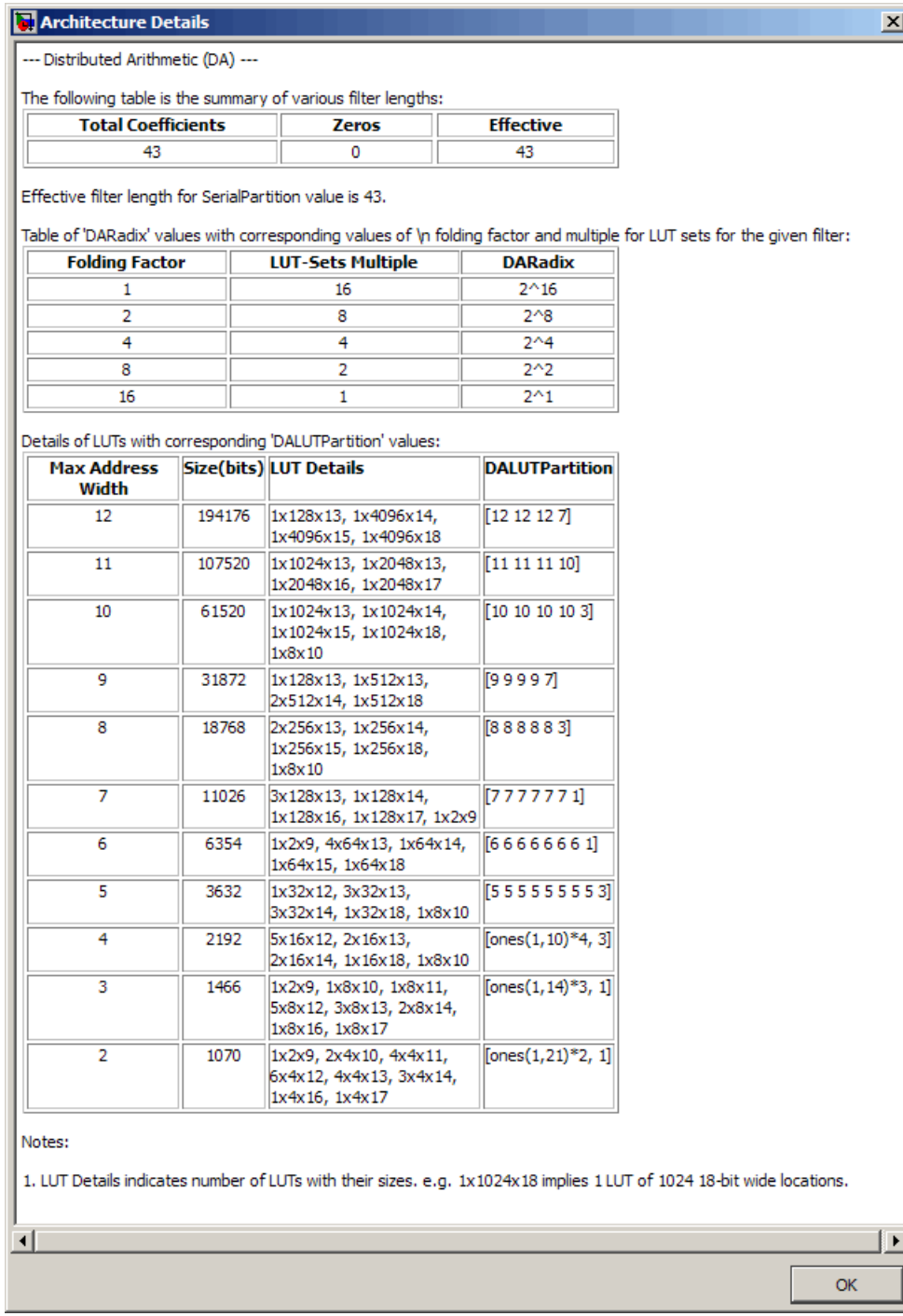
- `Address width` (default): Select from the pulldown menu to the right of **Specify LUT**. The menu contains an exhaustive list of LUT address widths for the filter.
- `Partition`: Enter a vector specifying the number and size of LUT partitions

As you interact with the **Specify folding** and **Specify LUT** options you can see the results of your choice in three display-only fields: `Folding factor,`, `Address width`, and `Total LUT size (bits)`.

In addition, when you click on the **View details** hyperlink, the coder displays a report showing complete DA architectural details for the current filter, including:

- Filter lengths.
- Complete list of applicable folding factors and their effect on the sets of LUTs
- Tabulation of all possible configurations of LUTs with total LUT Size and LUT details.

The following figure shows a typical report.



Architecture Details

--- Distributed Arithmetic (DA) ---

The following table is the summary of various filter lengths:

Total Coefficients	Zeros	Effective
43	0	43

Effective filter length for SerialPartition value is 43.

Table of 'DARadix' values with corresponding values of \n folding factor and multiple for LUT sets for the given filter:

Folding Factor	LUT-Sets Multiple	DARadix
1	16	2^16
2	8	2^8
4	4	2^4
8	2	2^2
16	1	2^1

Details of LUTs with corresponding 'DALUTPartition' values:

Max Address Width	Size(bits)	LUT Details	DALUTPartition
12	194176	1x128x13, 1x4096x14, 1x4096x15, 1x4096x18	[12 12 12 7]
11	107520	1x1024x13, 1x2048x13, 1x2048x16, 1x2048x17	[11 11 11 10]
10	61520	1x1024x13, 1x1024x14, 1x1024x15, 1x1024x18, 1x8x10	[10 10 10 10 3]
9	31872	1x128x13, 1x512x13, 2x512x14, 1x512x18	[9 9 9 9 7]
8	18768	2x256x13, 1x256x14, 1x256x15, 1x256x18, 1x8x10	[8 8 8 8 8 3]
7	11026	3x128x13, 1x128x14, 1x128x16, 1x128x17, 1x2x9	[7 7 7 7 7 7 1]
6	6354	1x2x9, 4x64x13, 1x64x14, 1x64x15, 1x64x18	[6 6 6 6 6 6 6 1]
5	3632	1x32x12, 3x32x13, 3x32x14, 1x32x18, 1x8x10	[5 5 5 5 5 5 5 5 3]
4	2192	5x16x12, 2x16x13, 2x16x14, 1x16x18, 1x8x10	[ones(1,10)*4, 3]
3	1466	1x2x9, 1x8x10, 1x8x11, 5x8x12, 3x8x13, 2x8x14, 1x8x16, 1x8x17	[ones(1,14)*3, 1]
2	1070	1x2x9, 2x4x10, 4x4x11, 6x4x12, 4x4x13, 3x4x14, 1x4x16, 1x4x17	[ones(1,21)*2, 1]

Notes:

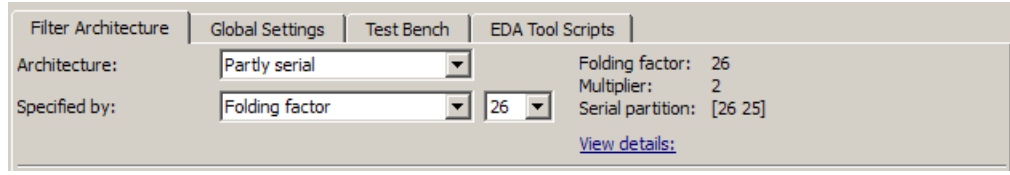
1. LUT Details indicates number of LUTs with their sizes. e.g. 1x1024x18 implies 1 LUT of 1024 18-bit wide locations.

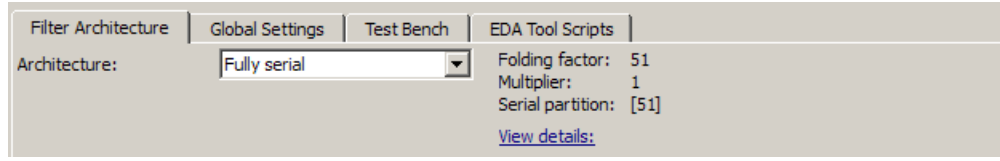
OK

See also “Distributed Arithmetic for FIR Filters” in the Filter Design HDL Coder™ documentation.

Enhanced Information Display for Serial Architectures

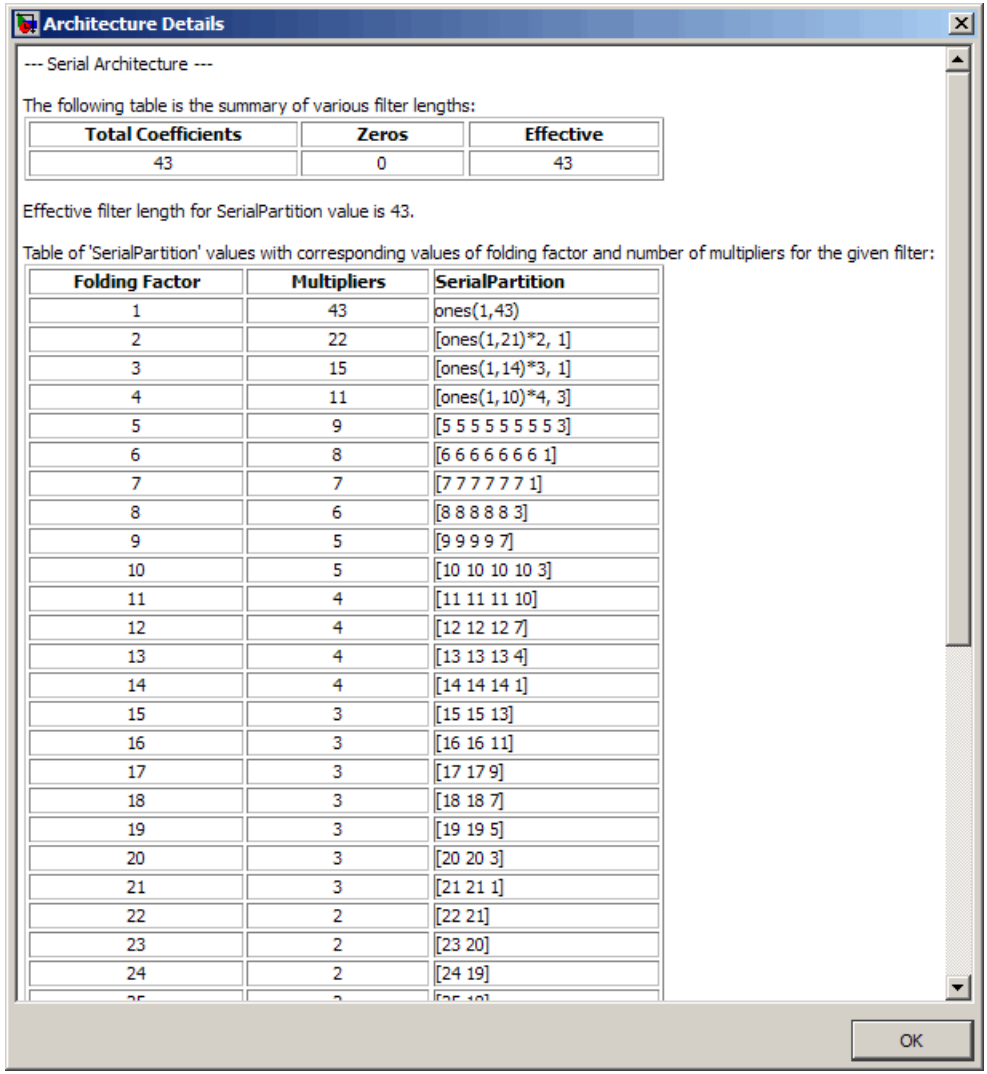
When you select the `Fully serial` or `Partly serial` **Architecture** options, the Generate HDL dialog box now displays a **View details** hyperlink, as shown in the following figures.





When you click on the **View details** link, the coder displays an HTML report in a separate window. The report displays an exhaustive table of possible folding factor, multiplier, and serial partition settings for the current filter. You can use the table to help you choose optimal settings for your design.

The following figure shows a partial view of typical report for a FIR filter of length 51.



See also “Selecting Parallel and Serial Architectures in the Generate HDL Dialog Box” in the Filter Design HDL Coder documentation.

Support for Variable Rate CIC Filters

Release R2010b supports HDL code generation for variable rate CIC filters, including the following filter types:

- CIC Decimators (mfilt.cicdecim)
- CIC Interpolators(mfilt.cicinterp)
- Multi-rate cascade with one CIC stage. (mfilt.cascade)

A variable rate CIC filter has a programmable rate change factor. It is assumed that the filter is designed with the maximum rate expected, and that the Decimation Factor (for CIC Decimators) or Interpolation Factor (for CIC Interpolators) is set to this maximum rate change factor.

Two new options support variable rate CIC filters:

- **Add rate port**: enables generation of `rate` and `load_rate` ports. When the `load_rate` signal is asserted, the `rate` port loads in a rate factor.
- **Testbench rate stimulus** lets you specify a stimulus applied to the rate port.

See also “Generating Code for Variable Rate CIC Filters”

Version 2.6 (R2010a) Filter Design HDL Coder Software

This table summarizes what's new in Version 2.6 (R2010a).

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

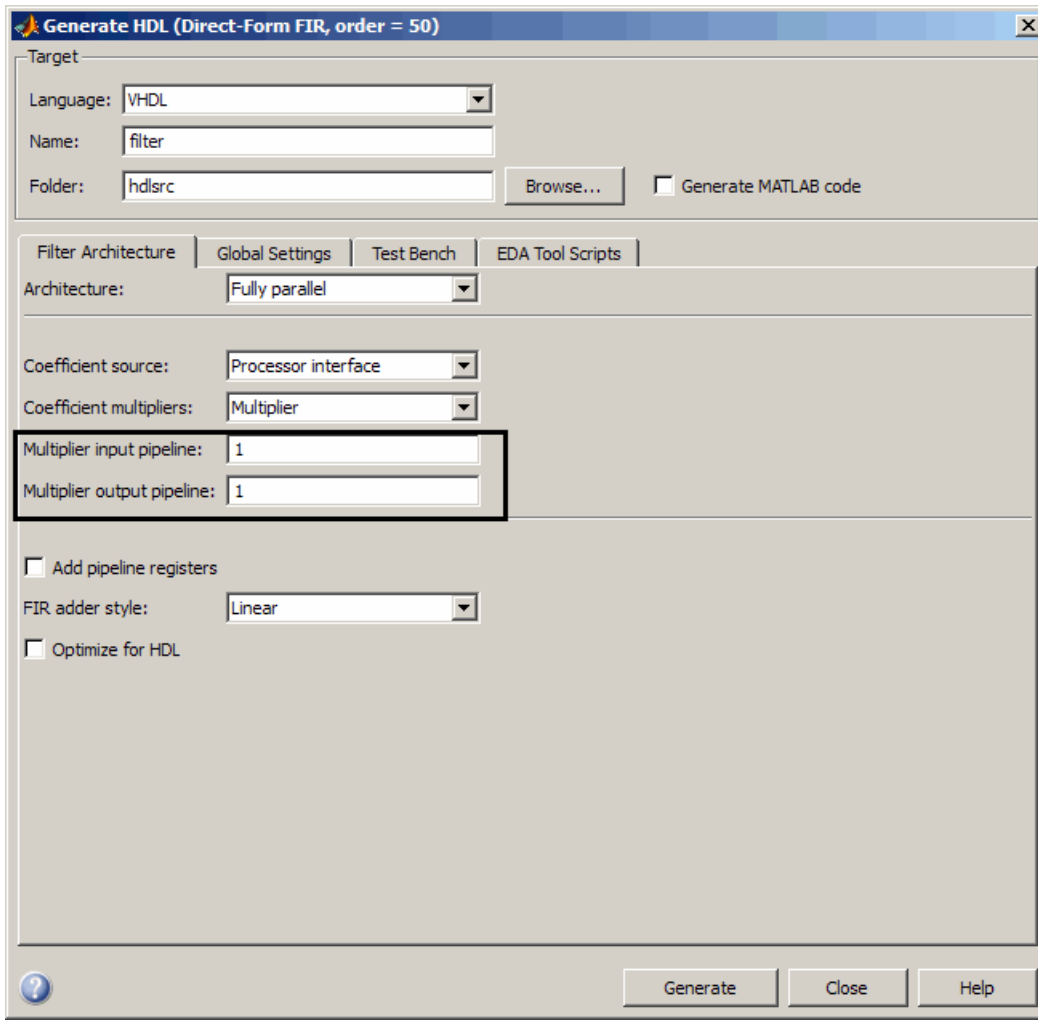
New features and changes introduced in this version are:

Multiplier Input and Output Pipelining for FIR Filters

Release R2010a lets you specify generation of pipeline stages at multiplier inputs or outputs for all FIR filter structures. Multiplier pipelining can help you achieve significantly higher clock rates. You can select input pipelining, output pipelining, or both. You can also specify the desired number of pipeline stages.

The following figure shows the new GUI options for multiplier pipelining options. These are:

- **Multiplier input pipeline**: Enter the desired number of pipeline stages to be added before each multiplier.
- **Multiplier output pipeline**: Enter the desired number of pipeline stages to be added after each multiplier.



The coder enables the **Multiplier input pipeline** and **Multiplier output pipeline** options when **Coefficient multipliers** is set to `Multiplier`.

Alternatively, you can specify the desired number of pipeline stages as `generatehdl` property/value pairs as follows:

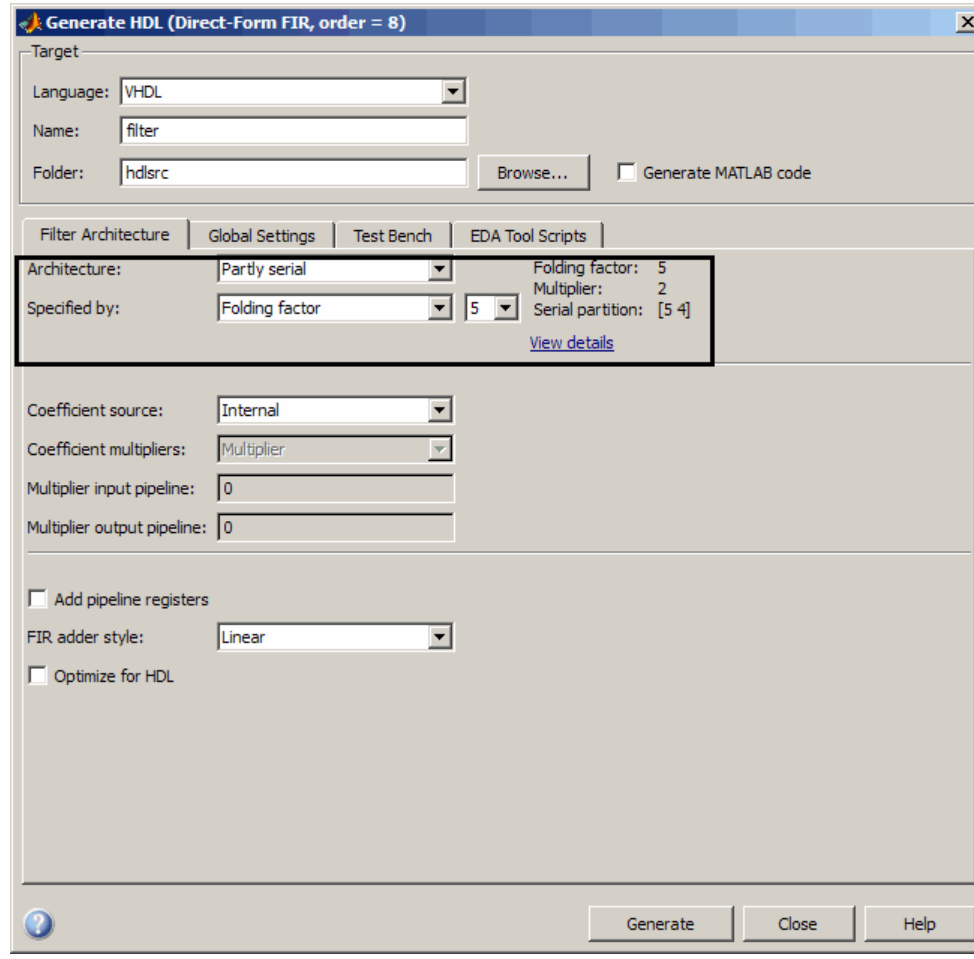
- `'MultiplierInputPipeline', nStages`
- `'MultiplierOutputPipeline', nStages`

Support for Partly Serial Architecture for FIR Decimators

Release R2010a lets you specify a partly serial architecture for FIR decimator filters (`mfilt.firdecim`). See "Speed vs. Area Optimizations for FIR Filters" for detailed information about parallel and serial architectures supported for HDL code generation.

Enhancements for Serial Architectures

When you select the `Partly serial` **Architecture** option, the Generate HDL dialog box now displays additional information and data entry fields related to serial partitioning, as shown in the following figure.



The **Specified by** pulldown menu lets you define the serial partitioning in any the following ways:

- Directly specify a vector of integers having *N* elements, where *N* is the number of serial partitions. Each element of the vector specifies the length of the corresponding partition.

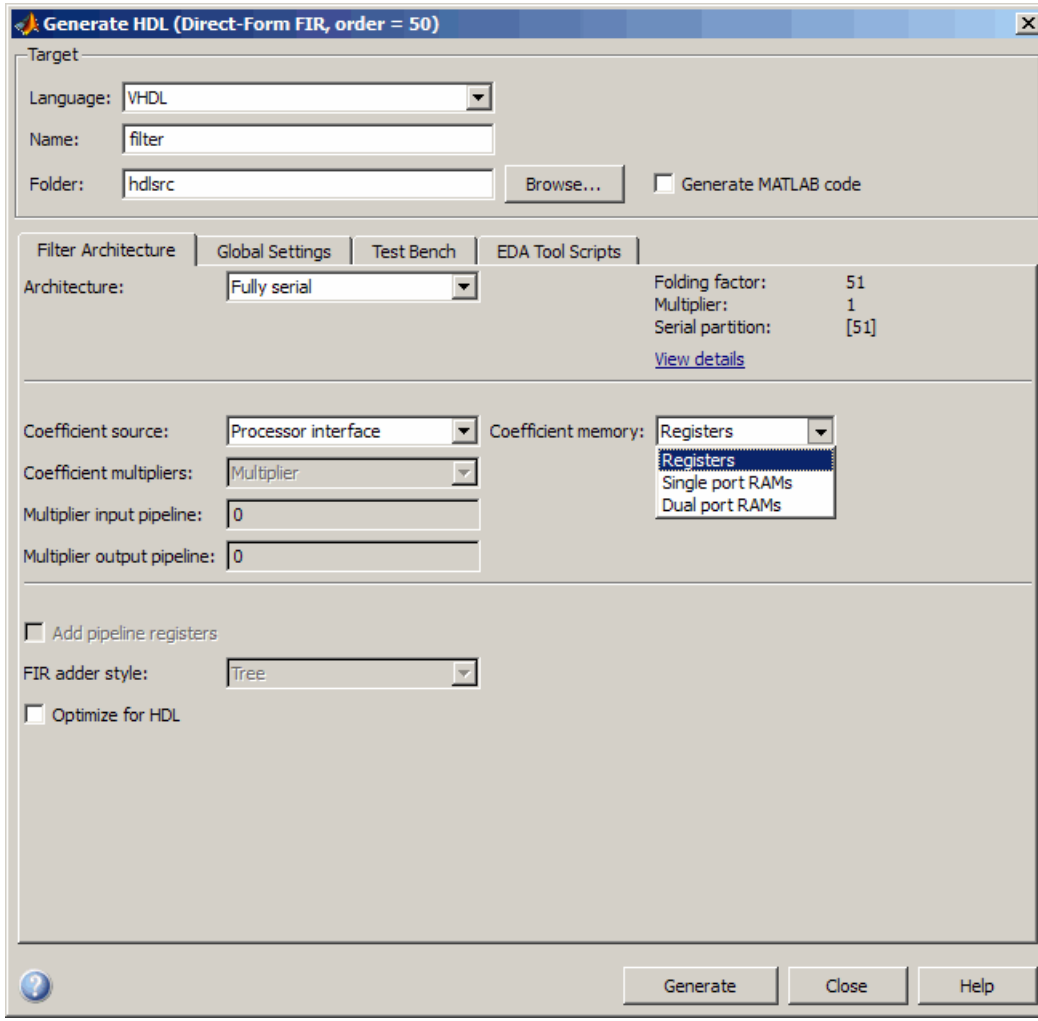
- Specify the desired hardware folding factor *`ff`*, an integer greater than `1`. Given the folding factor, the coder computes the serial partition and the number of multipliers.
- Specify the desired number of multipliers *`nmults`*, an integer greater than `1`. Given the number of multipliers, the coder computes the serial partition and the folding factor.

See “Speed vs. Area Optimizations for FIR Filters” for detailed information about parallel and serial architectures supported for HDL code generation.

The coder also provides the new `hdlgetserialpartition` function to help you define an optimal serial partition for your filter. `hdlgetserialpartition` calculates and displays an exhaustive table of `SerialPartition` values for a given filter, with corresponding values of folding factor and number of multipliers. See hdlfilterserialinfo for further information.

GUI Support for Programmable FIR FIlter Coefficients

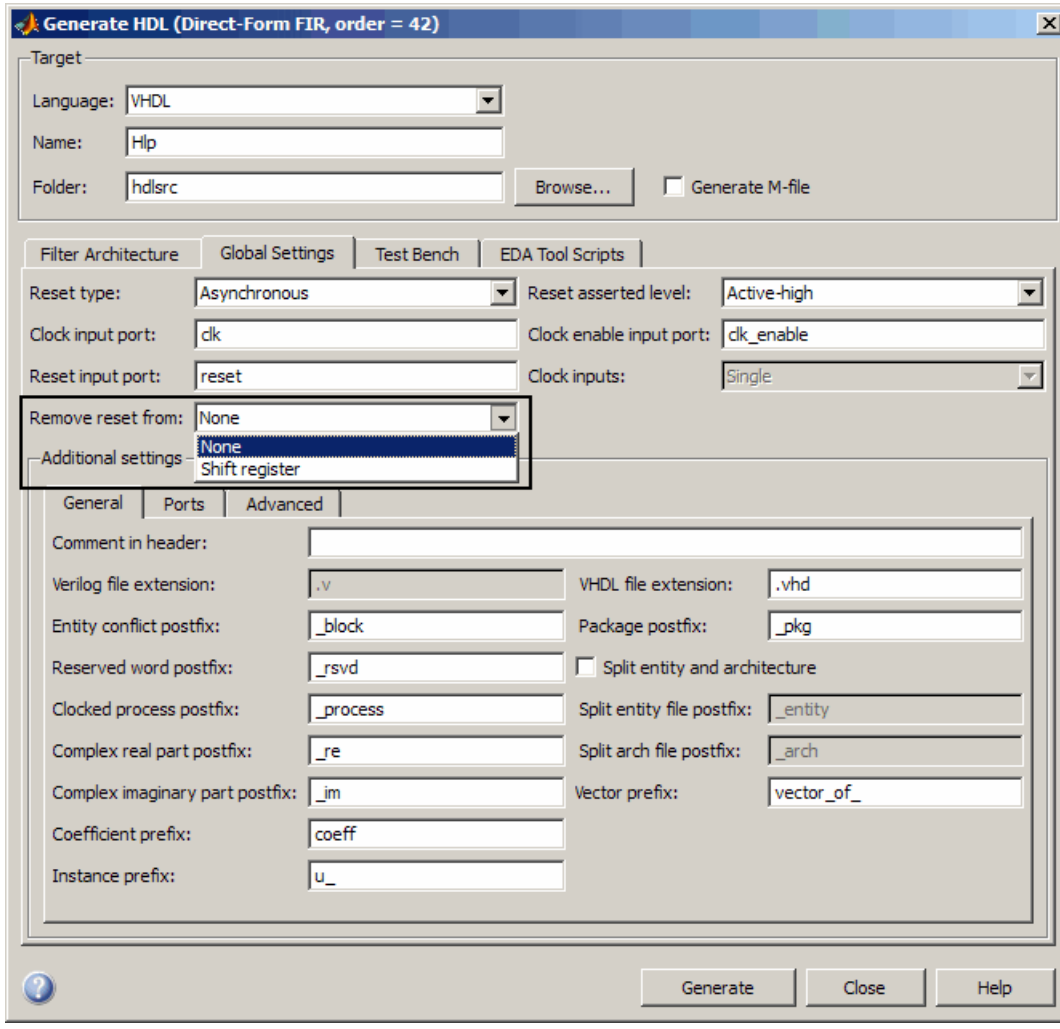
For FIR filters with serial architectures, the **Coefficient memory** pulldown menu now supports generation of a register or RAM based interface for loading coefficients. The following figure shows the **Coefficient memory** pulldown menu.



For detailed information, see “Specifying Programmable Filter Coefficients for FIR Filters” in the Filter Design HDL Coder documentation.

Option to Suppress Reset Logic Generation for Shift Registers

R2010a lets you suppress generation of reset logic for shift registers. To suppress reset logic, select `Shift register` from the **Remove reset from** pulldown in the **Global Settings** pane of the Generate HDL dialog box, as shown in the following figure.



You can also use `generatehdl` function with the property `RemoveResetFrom` to suppress generation of resets from shift registers.

GenerateCosimModel 'IN' and 'MQ' Property Values Removed

Release R2010a no longer supports the `'IN'` and `'MQ'` property values. Use the equivalent property values `'Incisive'` and `'ModelSim'`, as summarized in the following table.

Current Property Value	Deprecated Property Value
`generatehdl(filterObj, 'GenerateCosimModel',' Incisive' );`	`generatehdl(filterObj, 'GenerateCosimModel','IN' );`
`generatehdl(filterObj, 'GenerateCosimModel',' ModelSim' );`	`generatehdl(filterObj, 'GenerateCosimModel','MQ' );`

Compatibility Considerations

Replace any occurrences of 'IN' and 'MQ' in your control files and scripts with the new property values 'Incisive' and 'ModelSim'. In R2010a, the coder raises an error a warning if it encounters the old property values during code generation.

Version 2.5 (R2009b) Filter Design HDL Coder Software

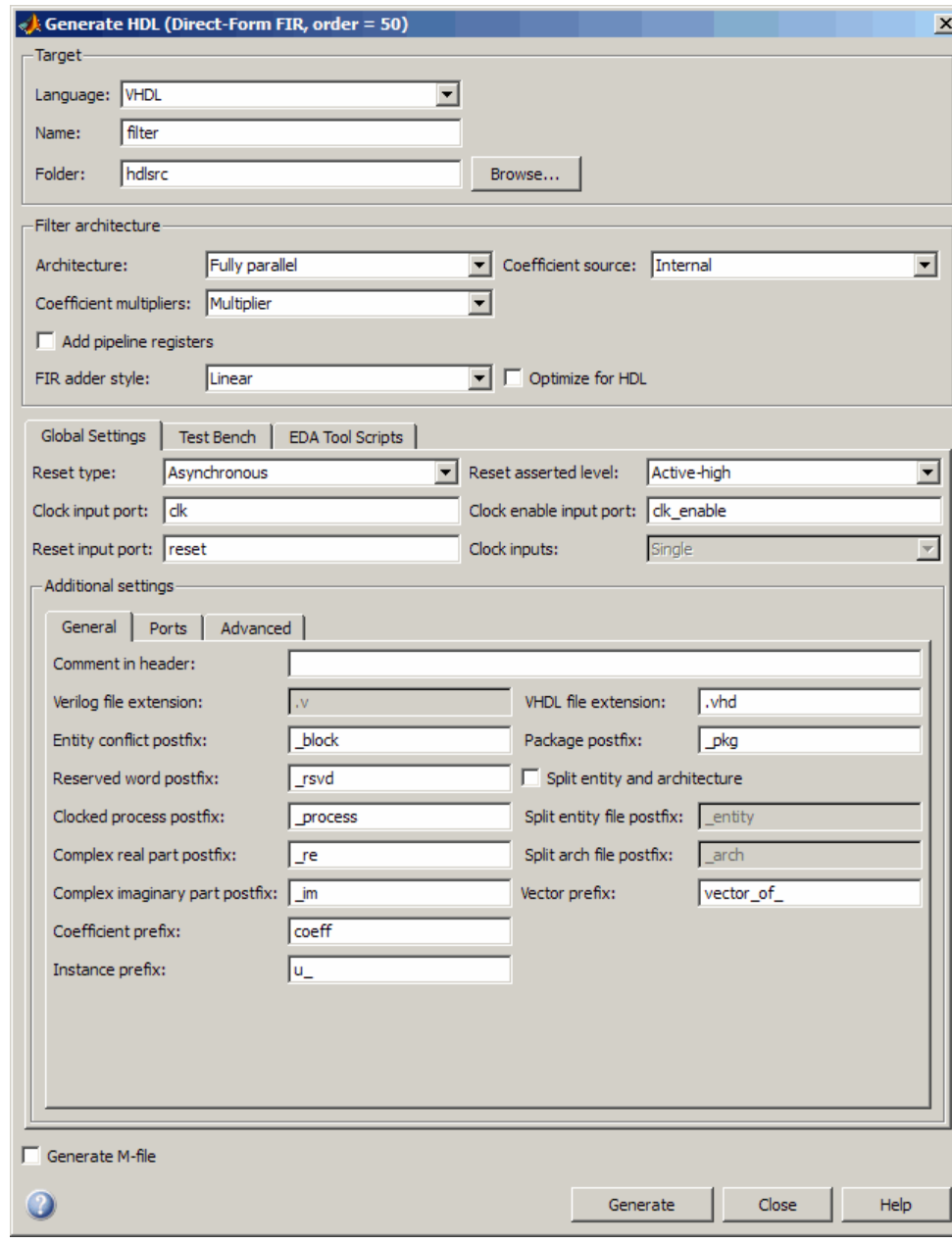
This table summarizes what's new in Version 2.5 (R2009b).

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:

Graphical User Interface Improved and Revised

R2009b includes an improved and revised Filter Design HDL Coder graphical user interface (GUI). The GUI now supports all functions within a single dialog box. The following figure shows the Generate HDL dialog box.


Generate HDL (Direct-Form FIR, order = 50)
Target
Language: VHDL
Name: filter
Folder: hdlsrc
Browse...
Filter architecture
Architecture: Fully parallel
Coefficient source: Internal
Coefficient multipliers: Multiplier
Add pipeline registers
FIR adder style: Linear
Optimize for HDL
Global Settings
Test Bench
EDA Tool Scripts
Reset type: Asynchronous
Reset asserted level: Active-high
Clock input port: clk
Clock enable input port: clk_enable
Reset input port: reset
Clock inputs: Single
Additional settings
General
Ports
Advanced
Comment in header:
Verilog file extension: .v
VHDL file extension: .vhd
Entity conflict postfix: _block
Package postfix: _pkg
Reserved word postfix: _rsvd
Split entity and architecture
Clocked process postfix: _process
Split entity file postfix: _entity
Complex real part postfix: _re
Split arch file postfix: _arch
Complex imaginary part postfix: _im
Vector prefix: vector_of_
Coefficient prefix: coeff
Instance prefix: u_
Generate M-file
Generate
Close
Help

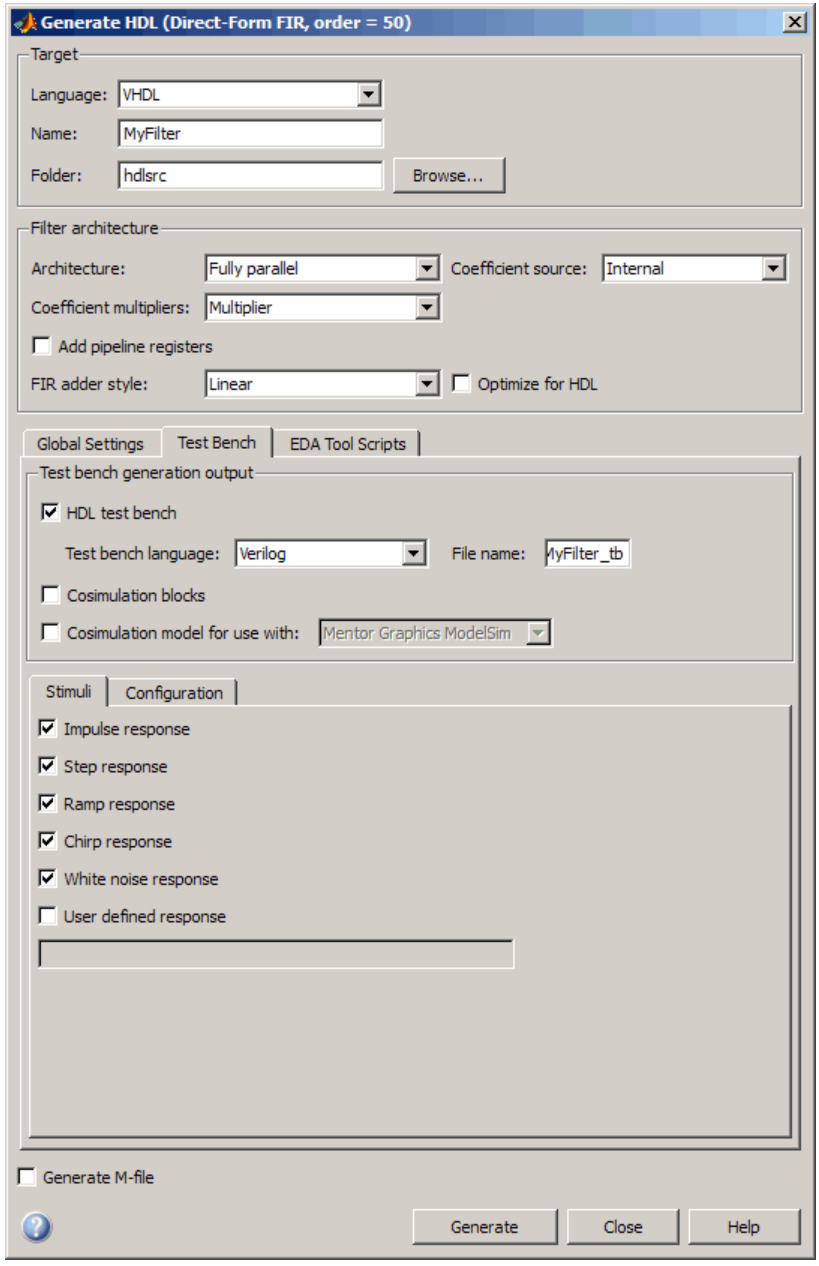
Compatibility Considerations

Some property labels have changed in the new GUI. The following tale lists the previous and current property labels.

Previous Property Label	Current Property Label
Language	**Filter target language**
Folder	**Target directory**

Test Bench GUI Reorganized

The following figure shows the reorganized **Test bench** pane of the Generate HDL dialog box.


Generate HDL (Direct-Form FIR, order = 50)
Target
Language: VHDL
Name: MyFilter
Folder: hdlsrc
Browse...
Filter architecture
Architecture: Fully parallel
Coefficient source: Internal
Coefficient multipliers: Multiplier
Add pipeline registers
FIR adder style: Linear
Optimize for HDL
Global Settings
Test Bench
EDA Tool Scripts
Test bench generation output
HDL test bench
Test bench language: Verilog
File name: MyFilter_tb
Cosimulation blocks
Cosimulation model for use with: Mentor Graphics ModelSim
Stimuli
Configuration
Impulse response
Step response
Ramp response
Chirp response
White noise response
User defined response
Generate M-file
Generate
Close
Help

The new **Testbench generation output** section contains three options:

- **HDL test bench**: Selecting this option enables generation of an HDL test bench, and also enables all options in the **Configuration** section of the **Test Bench** pane.
- **Cosimulation blocks**: Selecting this option enables Generate a model containing HDL Cosimulation block(s) for use in testing the DUT. Selecting this option also enables all options in the **Configuration** section of the **Test Bench** pane.
- **Cosimulation model for use with**: Selecting this option enables generation of a model containing an HDL Cosimulation block for use in testing the DUT, and lets you select the desired cosimulation tool. Selecting this option also enables all options in the **Configuration** section of the **Test Bench** pane.

To configure test bench options and generate test bench code, you must select one or more of the options of the **Testbench generation output** section. If you deselect all three options of the **Testbench generation output** section, the coder disables all options in the **Configuration** section of the **Test Bench** pane.

GenerateCosimModel 'IN' and 'MQ' Property Values Replaced

Release R2009b deprecates the `'IN'` and `'MQ'` property values. Use the equivalent property values `'Incisive'` and `'ModelSim'`, as summarized in the following table.

Current Property Value	Deprecated Property Value
`generatehdl(filterObj, 'GenerateCosimModel','`*`Incisive`*`');`	`generatehdl(filterObj, 'GenerateCosimModel','`*`IN`*`');`
`generatehdl(filterObj, 'GenerateCosimModel','`*`ModelSim`*`');`	`generatehdl(filterObj, 'GenerateCosimModel','`*`MQ`*`');`

Compatibility Considerations

Replace any occurrences of 'IN' and 'MQ' in your control files and scripts with the new property values 'Incisive' and 'ModelSim'. In R2009b, the coder issues a warning if it encounters the old property values during code generation. In subsequent releases, use of the old property values will raise an error.

Extended Complex Data Type Support

The coder now supports use of complex coefficients and complex input signals for additional filter structures. In many cases, you can use complex data and complex coefficients in combination. The following table shows the added filter structures that support complex data and/or coefficients, and the permitted combinations.

Filter Structure	Complex Data	Complex Coefficients	Complex Data and Coefficients
`dfilt.df1sos`	Y	Y	Y
`dfilt.df1tsos`	Y	Y	Y
`dfilt.df2sos`	Y	Y	Y
`dfilt.df2tsos`	Y	Y	Y
`mfilt.holdinterp`	Y	Y	N/A
`mfilt.firsrc`	Y	Y	Y
`mfilt.firtdecim`	Y	Y	Y

The coder also supports use of complex data and complex coefficients in combination for the `mfilt.firdecim` and `mfilt.firinterp` filter structures. The following table summarizes complex data type support for these filter structures.

Filter Structure	Complex Data	Complex Coefficients	Complex Data and Coefficients
`mfilt.firdecim`	Y	Y	Y (newly supported)
`mfilt.firinterp`	Y	Y	Y (newly supported)

For a list of filter structures supporting complex data, coefficients or both, see “Using Complex Data and Coefficients” in the Filter Design HDL Coder documentation.

Additional GUI Support for Complex Data

R2009b adds the following GUI options supporting use of complex data and coefficients.

- The **Input complexity** menu lets you select or disable generation of ports and signal paths for the real and imaginary components of a complex signal. The **Input complexity** setting defaults to `Real`, disabling generation of ports for complex input data. To enable generation of ports for complex input data, set **Input complexity** to `Complex`.
- The **Complex real part postfix** option (corresponding to the `ComplexRealPostfix` command-line property) specifies a string appended to names generated for the real part of complex signals. The default postfix is `'_re'`.
- The **Complex imaginary part postfix** option (corresponding to the `ComplexImagPostfix` command-line line property) specifies a string appended to names generated for the imaginary part of complex signals. The default postfix is `'_im'`.

See also “Using Complex Data and Coefficients” in the Filter Design HDL Coder documentation.

Generation of Model for Cosimulation Now Supports Multirate Filters

The coder now supports generation of cosimulation models for multirate filters. In previous releases, the coder supported generation of cosimulaton models for single-rate models only.

See “Generating a Simulink Model for Cosimulation with an HDL Simulator” for further information.

RAM Based Programmable Coefficients Supported for FIR Filters with Serial Architectures

For FIR filters with serial architectures, the coder now supports generation of a single-port or dual-port RAM interface for loading coefficients. Previous releases supported programmable coefficients stored in a register file.

For detailed information, see “Specifying Programmable Filter Coefficients for FIR Filters” in the Filter Design HDL Coder documentation.

Version 2.4 (R2009a) Filter Design HDL Coder Software

This table summarizes what's new in Version 2.4 (R2009a).

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:

Complex Data Type Support for FIR, CIC, and Other Filter Structures

The coder now supports use of complex coefficients and complex input signals for fully parallel FIR, CIC, and some other filter structures. In many cases, you can use complex data and complex coefficients in combination. The following table shows the filter structures that support complex data and/or coefficients, and the permitted combinations.

Filter Structure	Complex Data	Complex Coefficients	Complex Data and Coefficients
dfilt.dffir	Y	Y	Y
dfilt.dfsymfir	Y	Y	Y
dfilt.dfasymfir	Y	Y	Y
dfilt.dffirt	Y	Y	Y
dfilt.scalar	Y	Y	Y
dfilt.delay	Y	N/A	N/A
mfilt.cicdecim	Y	N/A	N/A
mfilt.cicinterp	Y	N/A	N/A
mfilt.firdecim	Y	Y	N
mfilt.firinterp	Y	Y	N
mfilt.linearinterp	Y	N/A	N/A

Properties Supporting Complex Data Types

The new `InputComplex` code generation property instructs the coder whether or not to generate the appropriate ports and signal paths for the real and imaginary components of a complex signal. To enable generation of ports for complex input data, set `InputComplex 'on'`, as in the following code example:

```
Hd = design(fdesign.lowpass,'equiripple','Filterstructure','dffir');
generatehdl(Hd, 'InputComplex', 'on');
```

Two new code generation properties have been added to help you customize naming conventions for the real and imaginary components of complex signals in generated HDL code. The new properties are:

- The `ComplexRealPostfix` property specifies a string to be appended to the names generated for the real part of complex signals. The default postfix is `'_re'`. See also ComplexRealPostfix.
- The `ComplexImagPostfix` property specifies a string to be appended to the names generated for the imaginary part of complex signals. The default postfix is `'_im'`. See also ComplexImagPostfix.

See “Using Complex Data and Coefficients” in the Filter Design HDL Coder User’s Guide for complete details on complex data type support.

Generation of Simulink Model for Cosimulation of Generated HDL Code

The coder supports generation of a Simulink model that is configured for:

- Simulink simulation of your filter design
- Cosimulation of your design with an HDL simulator

The generated model includes a behavioral model of the filter design, realized in a Simulink subsystem, and a corresponding HDL Cosimulation block, configured to cosimulate the filter design using Simulink. You can generate an HDL Cosimulation block for either of the following EDA Simulator Link™ products:

- EDA Simulator Link (default)
- EDA Simulator Link

See “Generating a Simulink Model for Cosimulation with an HDL Simulator” for further information.

Support for Programmable Coefficients for FIR Filters with Serial Architectures

For FIR filters with serial architectures, the coder now supports generation of a memory interface for loading coefficients, and generation of testbench coefficients to test the interface. In previous releases, these options were supported only for fully parallel FIR filters.

Programmable coefficients are supported for all serial architecture options (fully serial, partly serial, and cascade serial) of the following direct-form FIR filter types:

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

- `dfilt.dfasymfir`

For detailed information, see "Specifying Programmable Filter Coefficients for FIR Filters" in the Filter Design HDL Coder User's Guide.

Support for Programmable Coefficients for IIR Filters

For IIR filters, the coder now supports generation of a memory interface for loading coefficients, and generation of testbench coefficients to test the interface. In previous releases, this option was supported only for FIR filters.

The following IIR filter types support programmable filter coefficients:

- Second-order section (SOS) infinite impulse response (IIR) Direct Form I (`dfilt.df1sos`)
- SOS IIR Direct Form I transposed (`dfilt.df1tsos`)
- SOS IIR Direct Form II (`dfilt.df2sos`)
- SOS IIR Direct Form II transposed (`dfilt.df2tsos`)

For detailed information, see "Specifying Programmable Filter Coefficients for IIR Filters" in the Filter Design HDL Coder documentation.

Default Entity Conflict Postfix Changed

The default value for the **Entity conflict postfix** property (and the corresponding CLI property, `EntityConflictPostfix`) has been changed from `'_entity'` to `'_block'`.

Compatibility Considerations

If your scripts rely on the previous default value (`'_entity'`) for the **Entity conflict postfix** property, you will need to explicitly set the property value to `'_entity'`.

Version 2.3 (R2008b) Filter Design HDL Coder Software

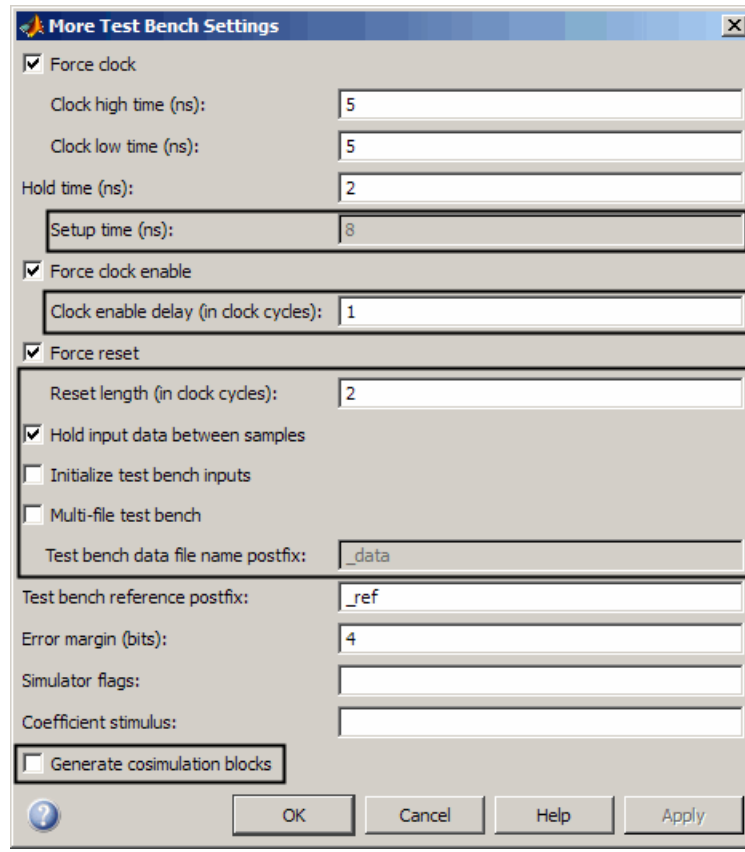
This table summarizes what's new in Version 2.3 (R2008b).

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:

Test Bench Enhancements

The appearance of the More Test Bench Settings dialog box has been revised, and a number of options have been added. The following figure shows the default set of options in the More Test Bench Settings dialog box. Options that have been added to the GUI are highlighted.



Each new option (except **Setup time (ns)**) has a corresponding command-line property. The following table lists the new options and their corresponding command-line properties, and provides hyperlinks to the relevant documentation.

GUI Option	Command-Line Property
Setup time (ns): See "Setting a Hold Time for Data Input Signals" and "Configuring Resets".	This display-only field does not have a corresponding user-settable command-line property.
Clock enable delay (in clock cycles): See "Configuring the Clock".	TestBenchClockEnableDelay

GUI Option	Command-Line Property
Reset length: See "Configuring Resets".	ResetLength
Hold input data between samples: See "Holding Input Data in a Valid State".	HoldInputDataBetweenSamples
Initialize test bench inputs: See "Setting an Initial Value for Test Bench Inputs".	InitializeTestBenchInputs
Multi-file test bench: See "Splitting Test Bench Code and Data into Separate Files".	MultifileTestBench
Test bench data file name postfix: See "Splitting Test Bench Code and Data into Separate Files".	TestBenchDataPostFix
Test bench reference postfix: See "Setting a Postfix for Reference Signal Names".	TestBenchReferencePostFix
Generate cosimulation blocks: See "Generating HDL Cosimulation Blocks for Use with HDL Simulators".	GenerateCoSimBlock

Distributed Arithmetic Restriction Removed for Symmetrical and Asymmetrical FIR Filters

The `DARadix` property specifies the number of bits processed simultaneously in a distributed arithmetic architecture. In previous releases, when generating code for symmetrical (`dfilt.dfsymfir`) or asymmetrical (`dfilt.dfasymfir`) FIR filters, the `DARadix` value was required to be less than or equal to 2. Specification of a `DARadix` value greater than 2 for these filter types caused a warning to be issued during code generation.

In Release 2008b, the coder permits use of `DARadix` values greater than 2 for these filter types. Other requirements for setting the `DARadix` property still apply. For details, see "DARadix Property" and "Considerations for Symmetrical and Asymmetrical Filters" in the Filter Design HDL Coder documentation.

For general information on distributed arithmetic support, see "Distributed Arithmetic for FIR Filters" in the Filter Design HDL Coder documentation.

-novopt Flag Added to the Default Simulation Command in Generated Compilation Scripts

For improved operation with the ModelSim® (Version 6.2 and later) simulator, the default values of the `HDLSimCmd` property string (and the **Simulation Command** GUI option) now includes the `-novopt` flag, as follows:

```
'vsim -novopt work.%s\n'
```

The `-novopt` flag directs the ModelSim simulator not to perform optimizations that remove signals from the simulation view.

Compatibility Considerations

If you are using ModelSim 6.0 or an earlier version, you should set the `HDLSimCmd` property string (or the **Simulation Command** GUI option) to omit the `-novopt` option, as follows:

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

ModelSim .do Test Bench Option Removed

The **Modelsim .do file** test bench generation option, and the corresponding `'Modelsim'` test bench type argument for the `generatetb` function, are no longer supported and have been removed from the current release.

In the current release, `generatetb` displays an error message and terminates test bench generation if the `'Modelsim'` test bench type 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 default to the current setting of the `TargetLanguage` property (`'VHDL'` or `'Verilog'`).

See also `generatetb`.

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	No

New features and changes introduced in this version are:

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 command-line Properties Supported

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

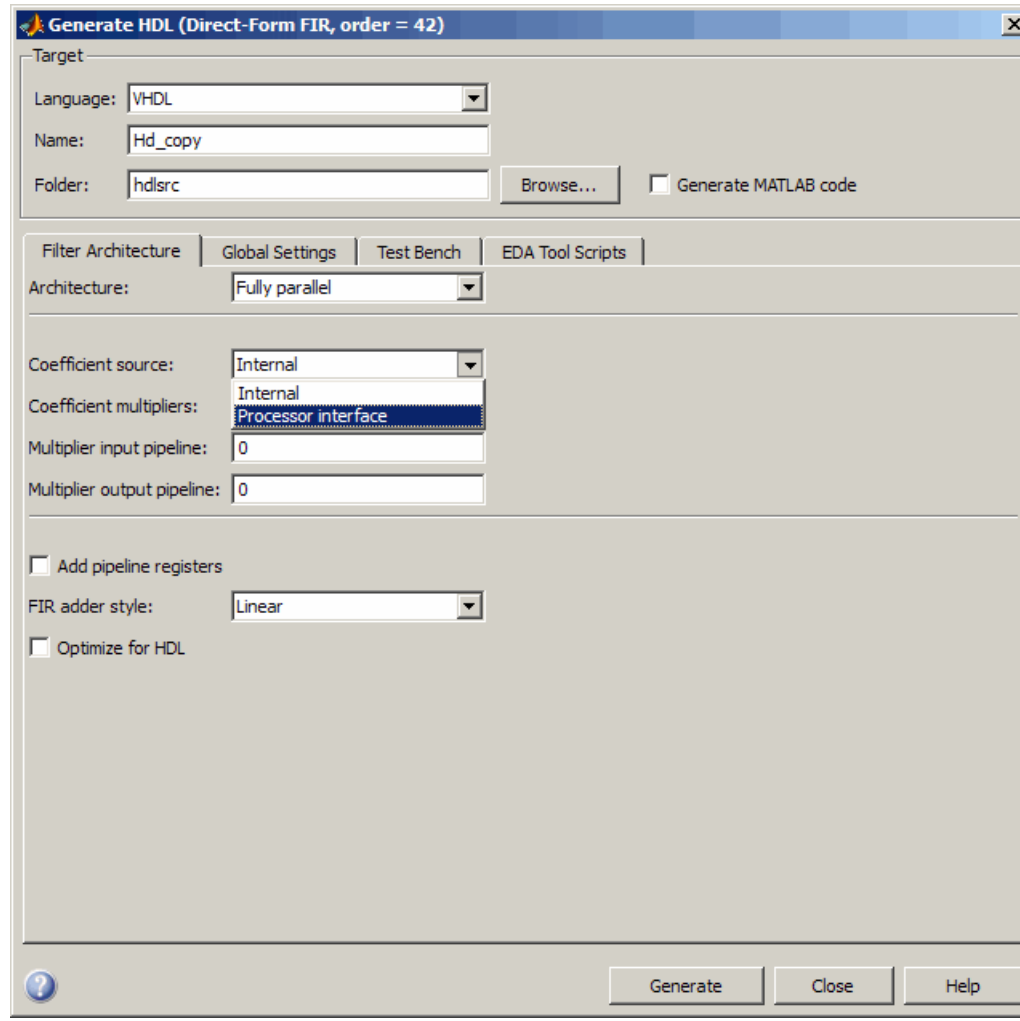
- `HoldInputDataBetweenSamples`: You can apply this property to filters that do not have parallel architectures. In such filters, data can be delivered to the outputs `N` cycles (`N >= 2`) later than the inputs. The `HoldInputDataBetweenSamples` property determines how long (in terms of clock cycles) 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 Processor Interface for FIR Filter Coefficients

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

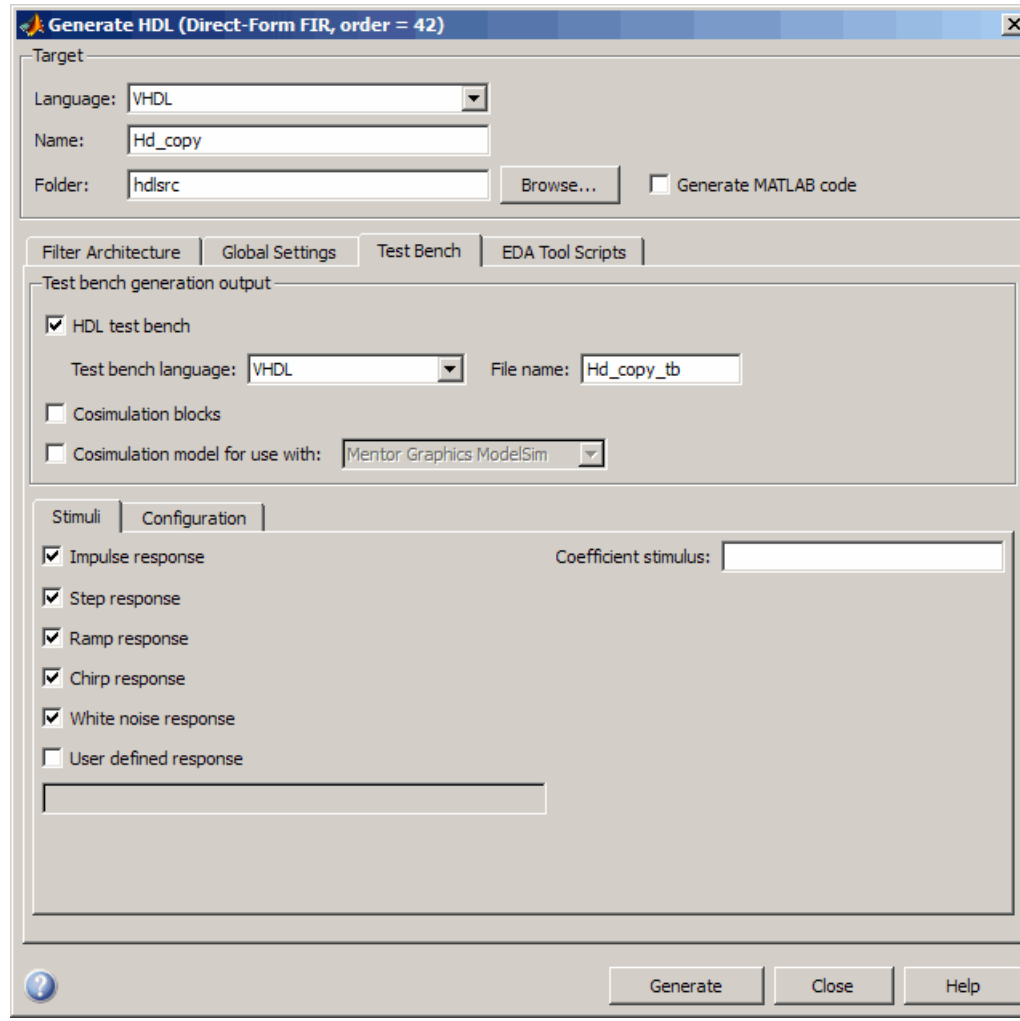
The new GUI options are:

- 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 interface (`Processor interface`). The corresponding command-line property is `CoefficientSource`.



- The **Coefficient stimulus** option on the More Test Bench Settings dialog box lets you specify how the test bench tests the generated

processor interface . The corresponding command-line property is `TestbenchCoeffStimulus`.



For detailed information on these options, see “Specifying Programmable Filter Coefficients for FIR Filters” 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 Is 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 41.
`ScaleWarnBits` property	Property is ignored	No replacement	See "ScaleWarnBits Property No Longer Supported" on page 42.

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 40.

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

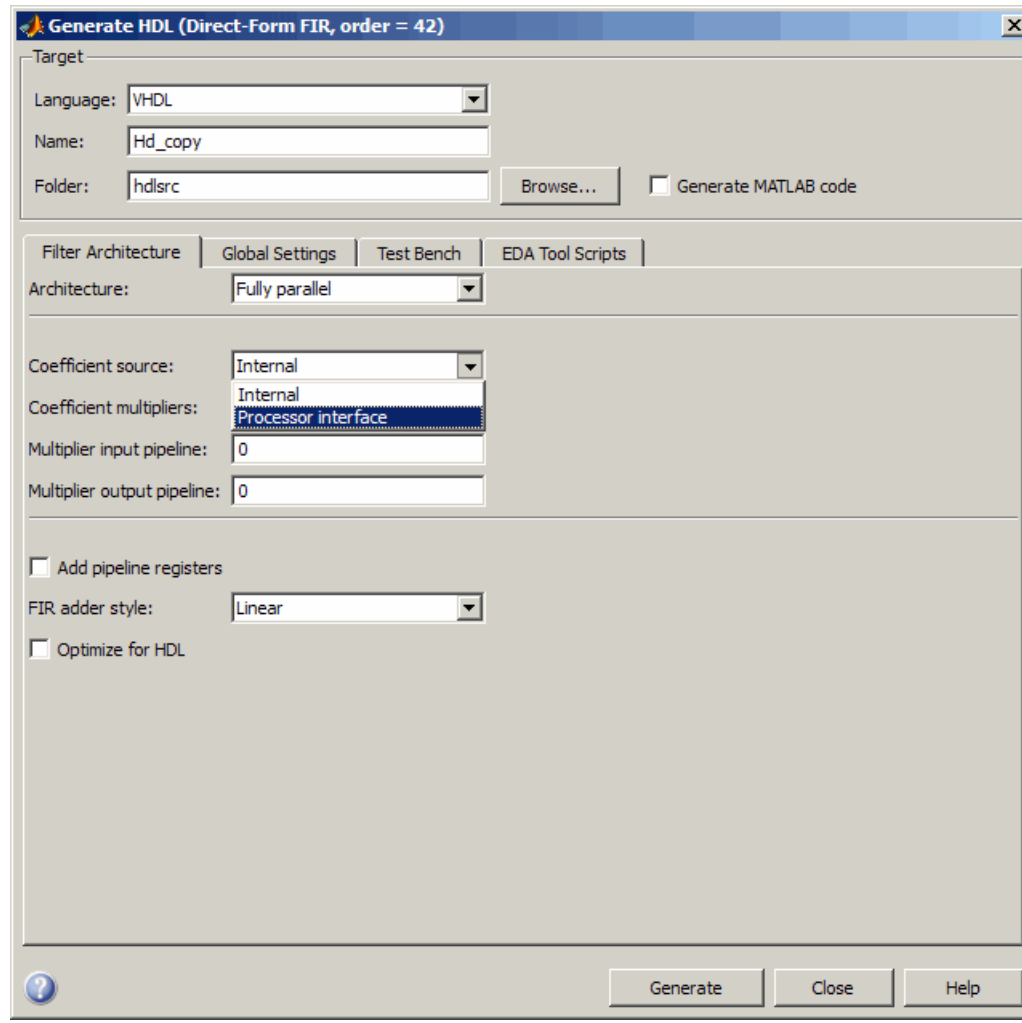
If you have files that contain commands that reference the `ScaleWarnBits` property, such references are ignored. Remove references to `ScaleWarnBits` from your code.

Summary of GUI Enhancements and Revisions

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

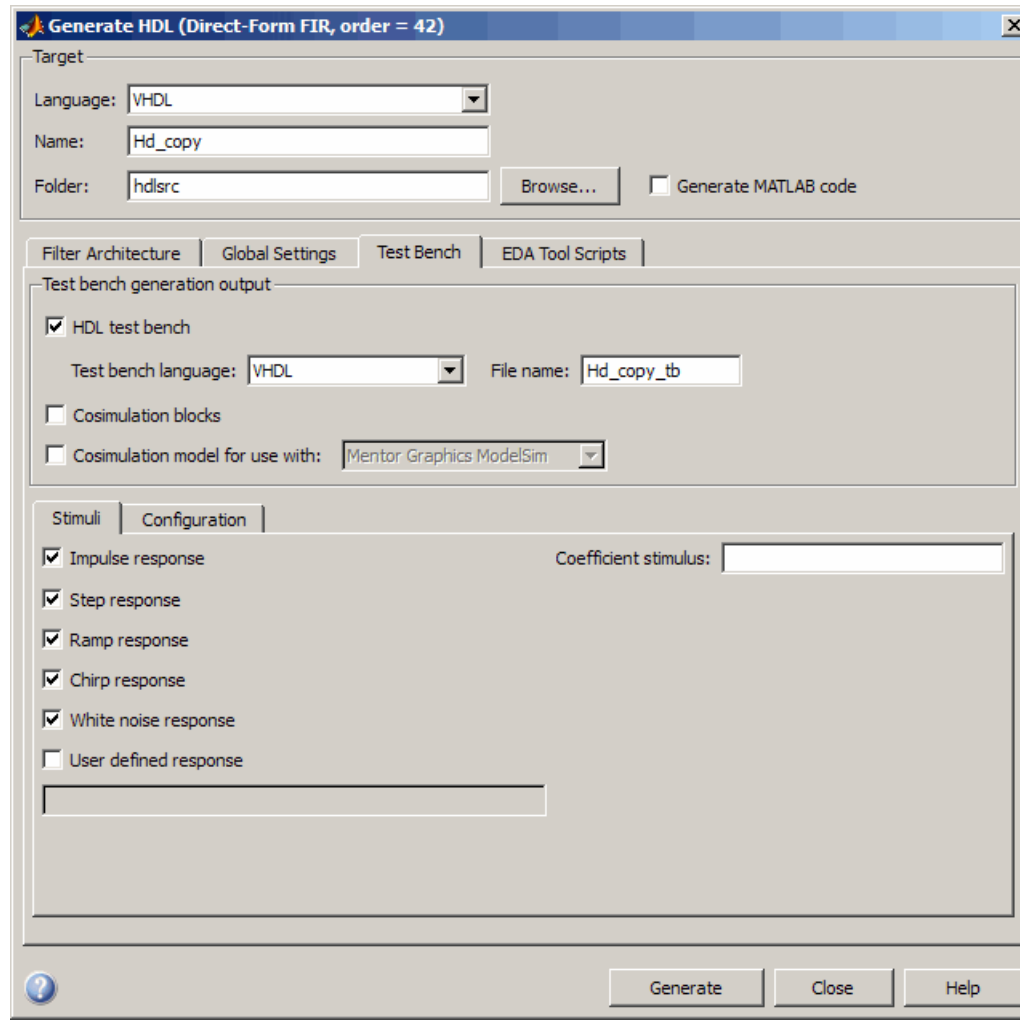
Generate HDL Dialog Box

The Generate HDL dialog box now includes the **Coefficient source** menu. See “GUI Support for Processor Interface for FIR Filter Coefficients” on page 37.



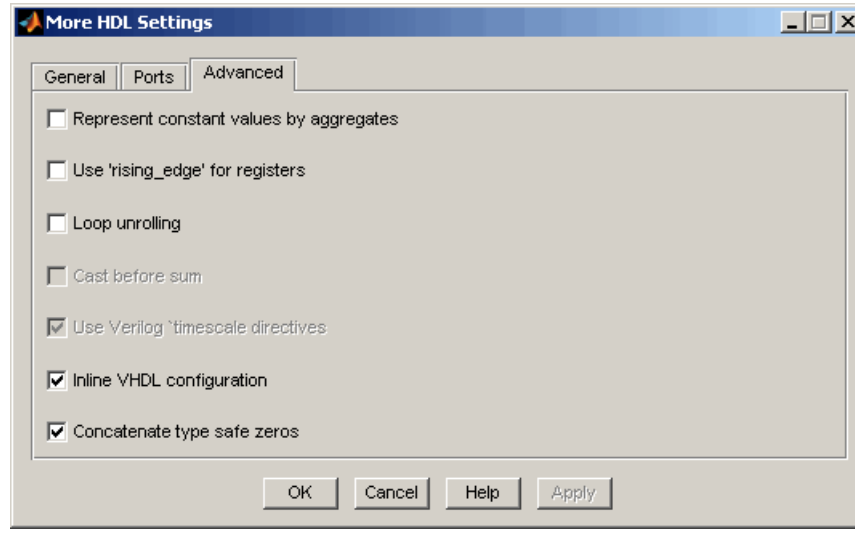
More Test Bench Settings Dialog Box

The More Test Bench Settings dialog box now includes the **Coefficient stimulus** option. See “GUI Support for Processor Interface for FIR Filter Coefficients” on page 37.



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 42.) The following figure shows the default settings for the **Advanced** pane.


More HDL Settings
General
Ports
Advanced
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
OK
Cancel
Help
Apply

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:

Processor Interface for Loading FIR Filter Coefficients

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 processor interface for loading coefficients, and test the interface. These properties are:

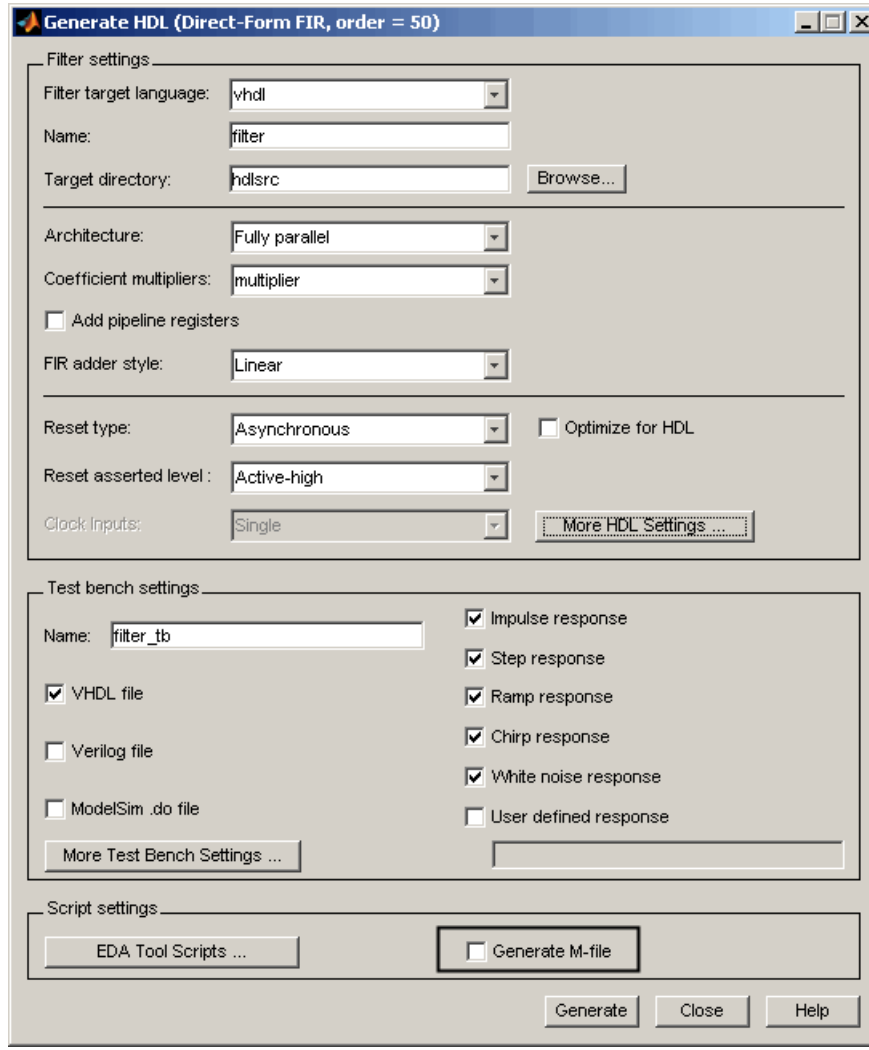
- `CoefficientSource`: This property specifies whether coefficients are directly coded, or loaded via a processor interface.

- `TestbenchCoeffStimulus`: This property specifies how the test bench tests the generated processor interface and the performance of the filter.

See “Specifying Programmable Filter Coefficients for FIR Filters” for a detailed description of this feature.

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



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 a MATLAB file that you can use to regenerate HDL code for the filter.

For detailed information, see "Capturing Code Generation Settings to a Script".

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 82 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.

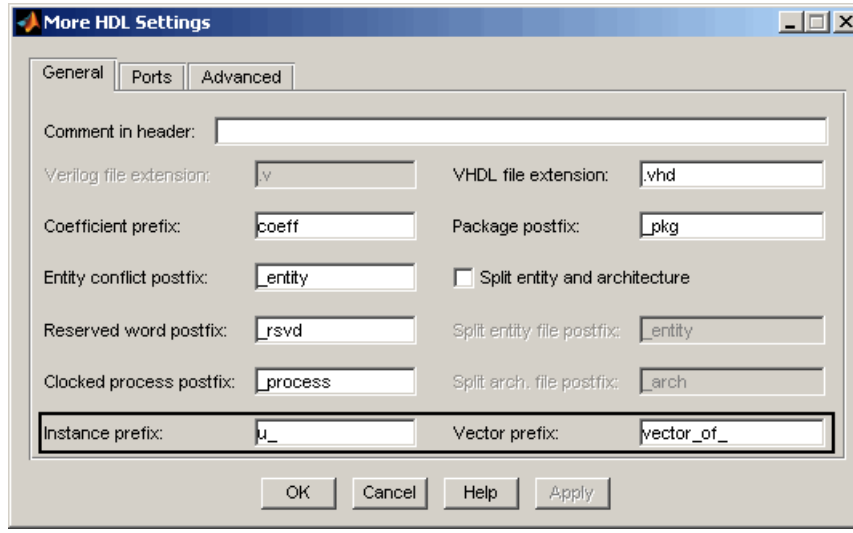
New Code Generation Properties Supported

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.



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`

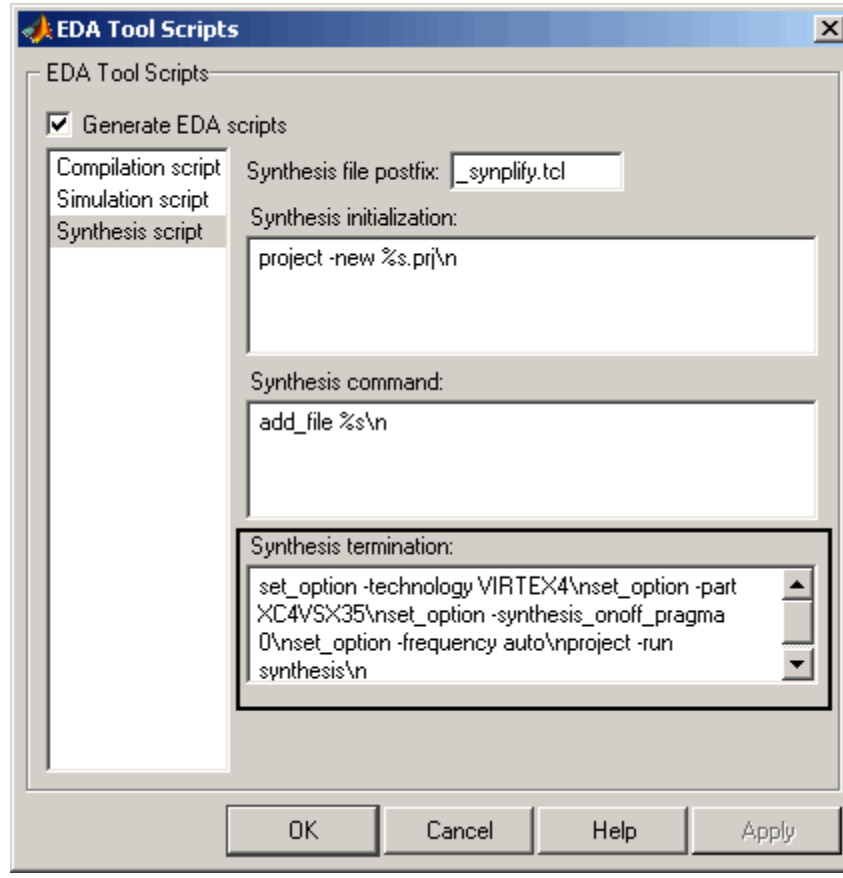
  In previous releases, this option defaulted to `XC2V500`.

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.



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.

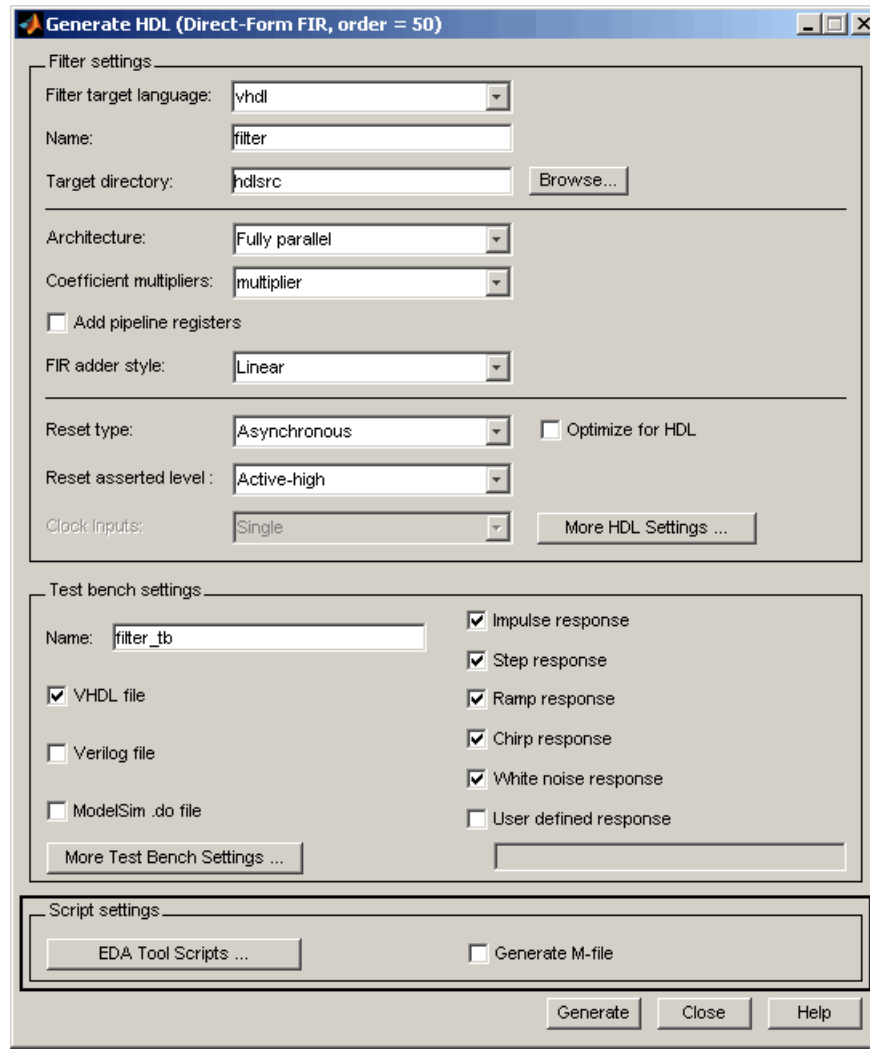
Summary of GUI Enhancements and Revisions

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

Generate HDL Dialog Box

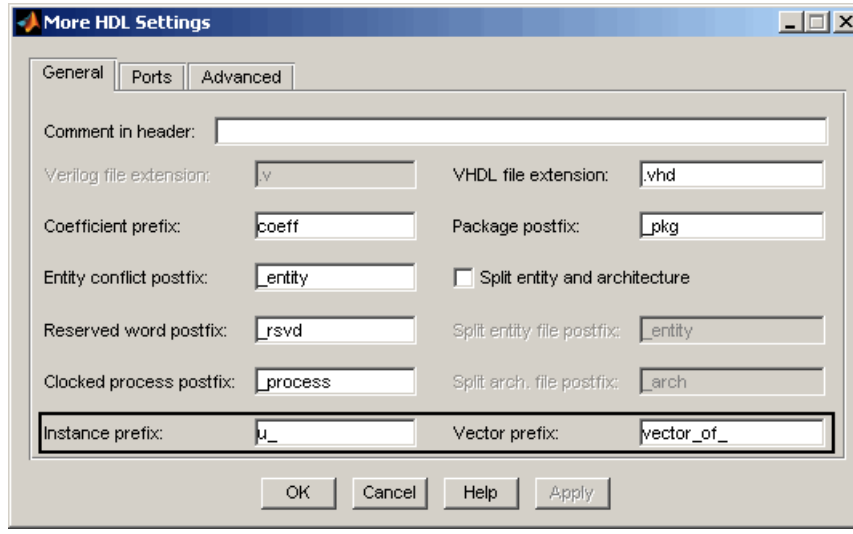
The following figure shows the Generate HDL dialog box. Revisions and enhancements to this dialog box include:

- 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 a file that you can use to reconstruct the filter and regenerate HDL code. See "Generate M-file Option Captures GUI Settings to Generated Command File" on page 47 for details.
- The **EDA Tool Scripts** button and the **Generate M-file** option are grouped together in a new **Script settings** section.



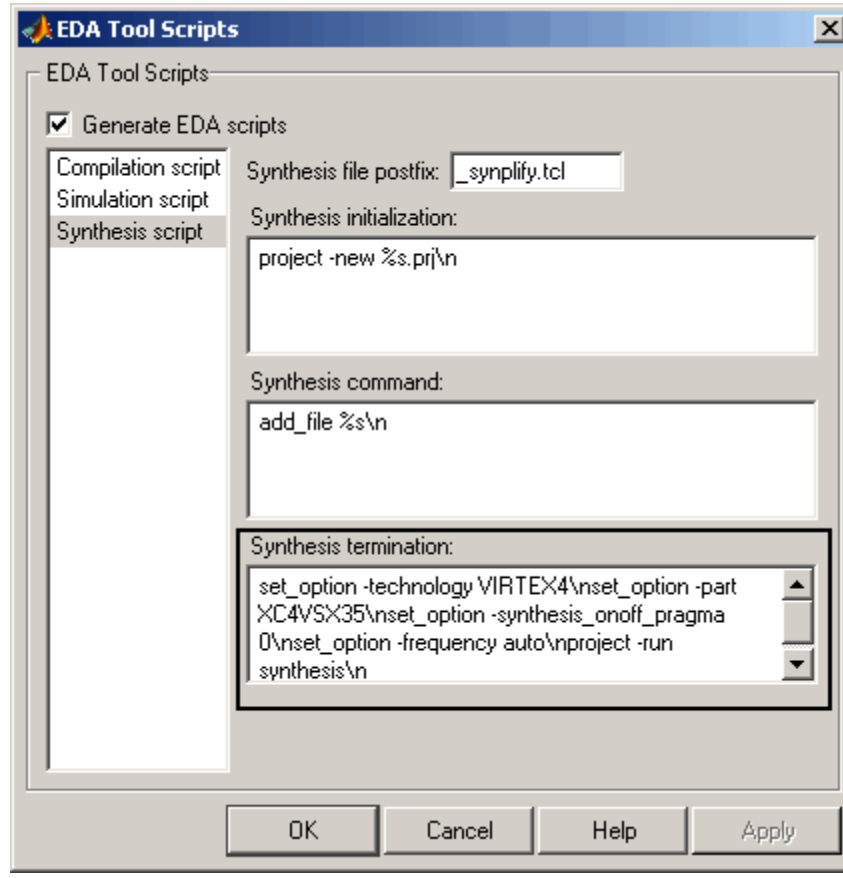
More HDL Settings Dialog Box

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 49 for details.



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 50) as shown in the following figure.


EDA Tool Scripts
EDA Tool Scripts
Generate EDA scripts
Compilation script
Simulation script
Synthesis script
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
OK
Cancel
Help
Apply

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

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 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.

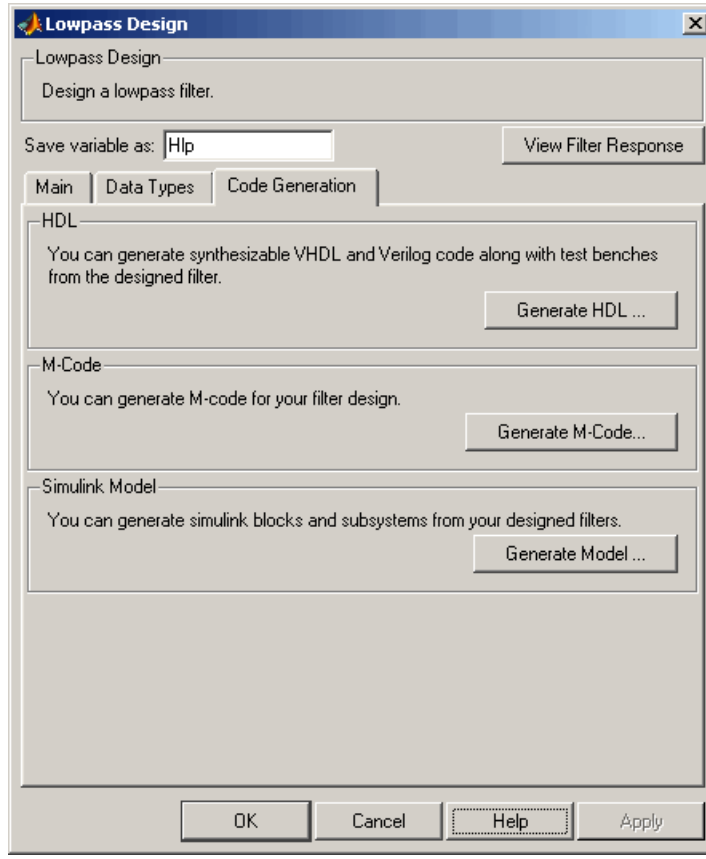
Code Generation Support for Polyphase Sample Rate Converters (mfilt.firsrc)

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).



To generate HDL code from `filterbuilder`:

1. Click the **Code Generation** tab.
2. 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`.
3. 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 59 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 the 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);
```

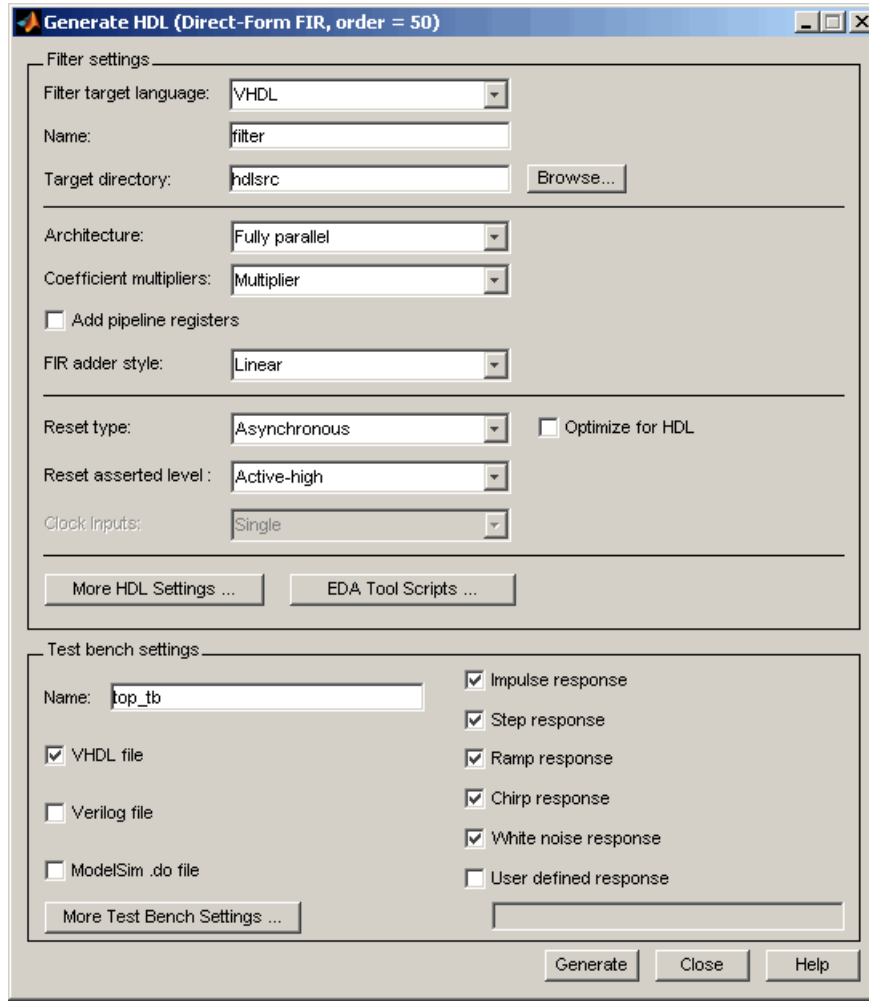
`fdhdltool` operates on a copy of the filter object, rather than the original object in the workspace. Any changes made to the original filter object after `fdhdltool` is invoked will not affect the copy and will not update the Generate HDL dialog box.

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



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

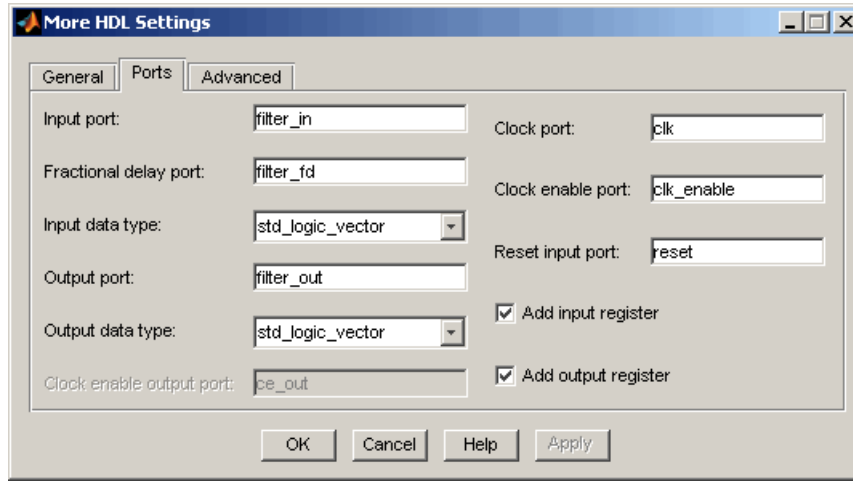
for third-party electronic design automation (EDA) tools. See “EDA Tool Scripts Dialog Box” on page 64.

- 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.

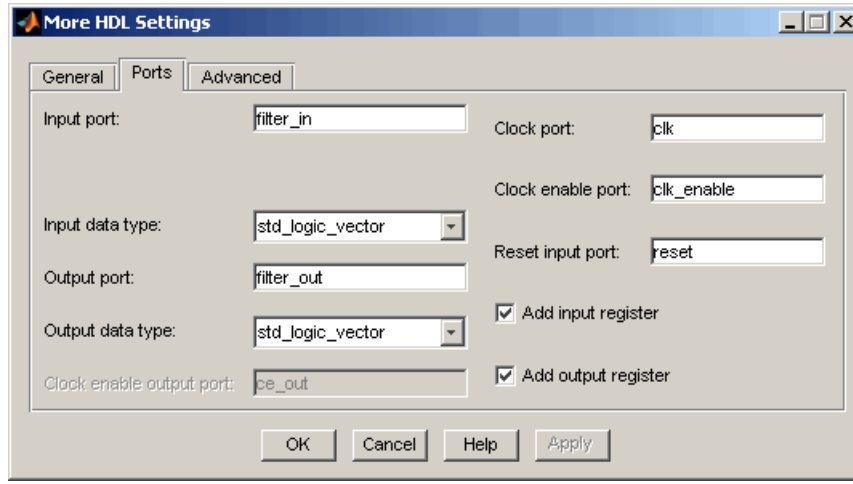
More HDL Settings Dialog Box

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 56), the new **Fractional delay port** field is displayed, as shown in the following figure.



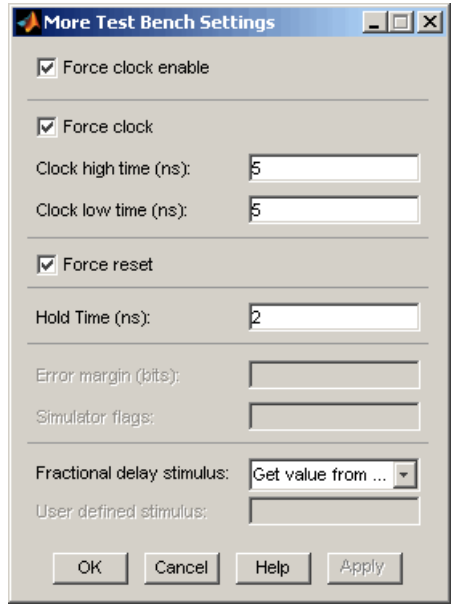
For all other filter types, the **Fractional delay port** field is omitted, as shown in the following figure.



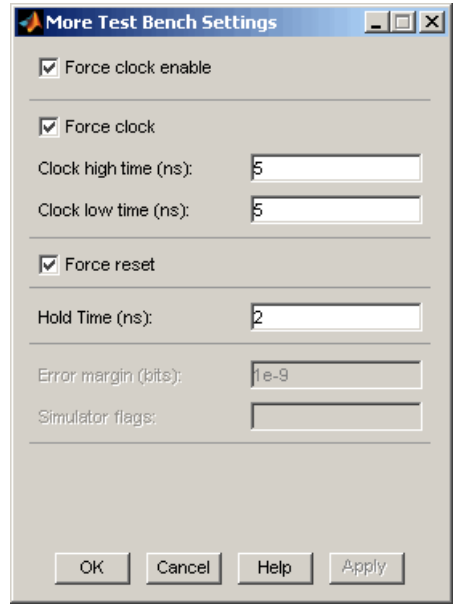
More Test Bench Settings Dialog Box

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



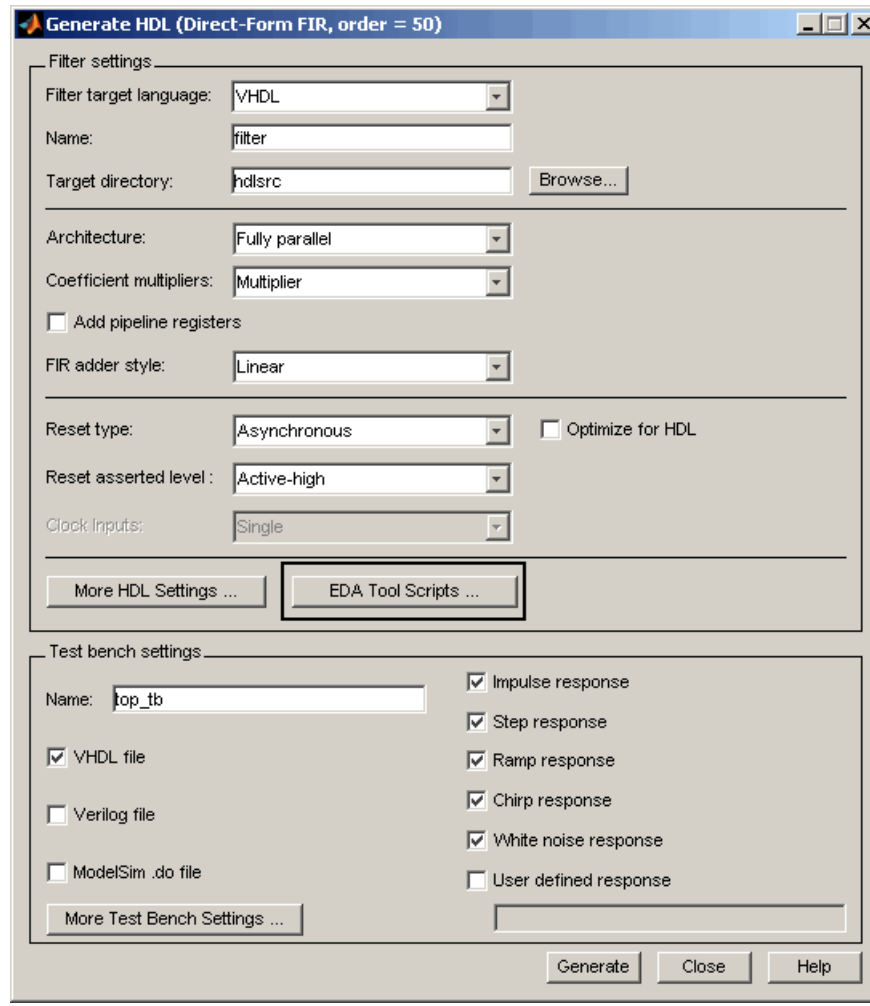
For all other filter types, the **Fractional delay stimulus** and **User defined stimulus** options are omitted, as shown in the following figure.



EDA Tool Scripts Dialog Box

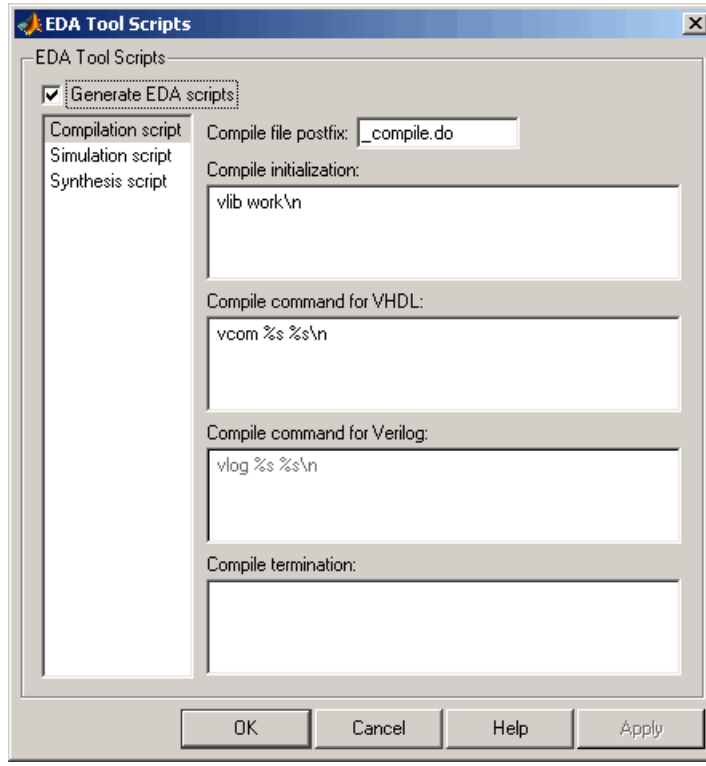
The new EDA Tool Scripts dialog box lets you set all options that control generation of script files for third-party electronic design automation (EDA) tools. In previous releases, script generation options were available only through `generatehdl` properties.

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).

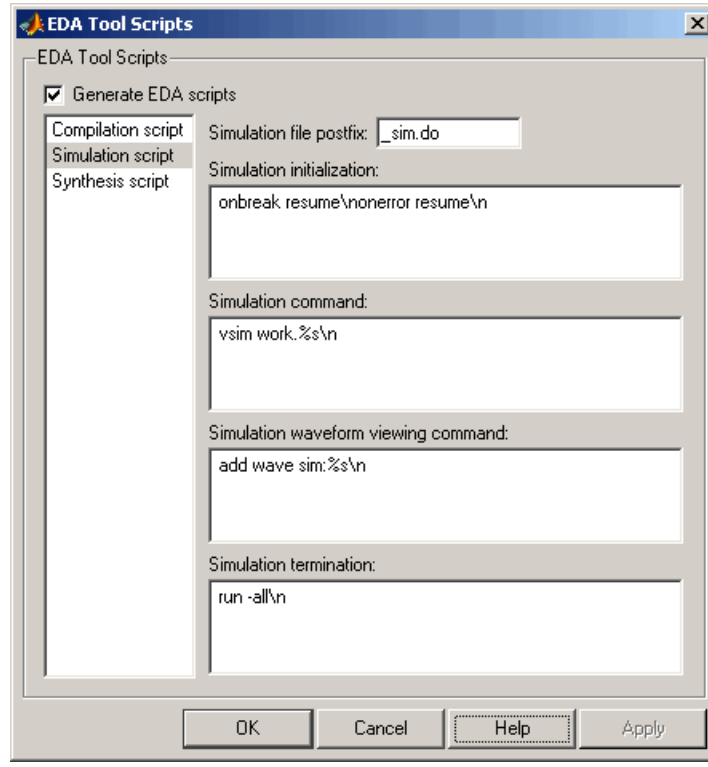


The following figures show the three panes of the EDA Tool Scripts dialog box.

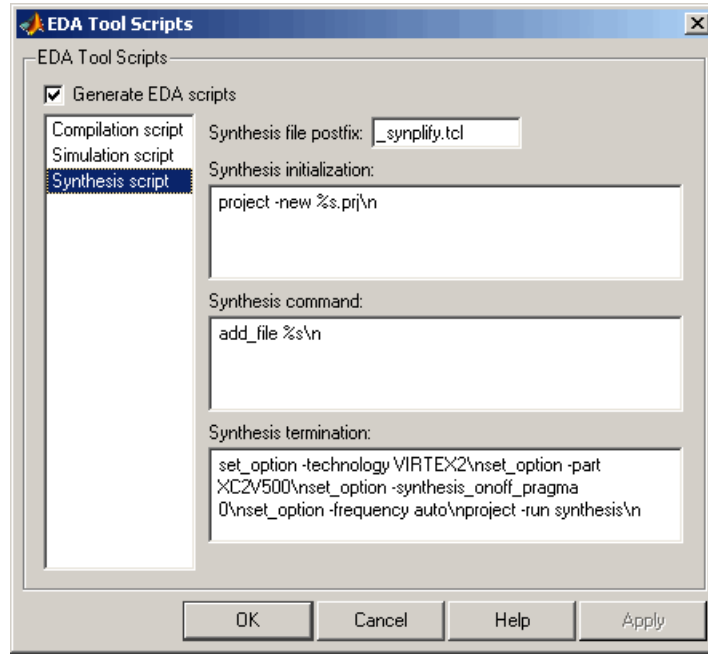
The **Compilation script** pane displays options related to customizing scripts for compilation of generated VHDL or Verilog code.



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.



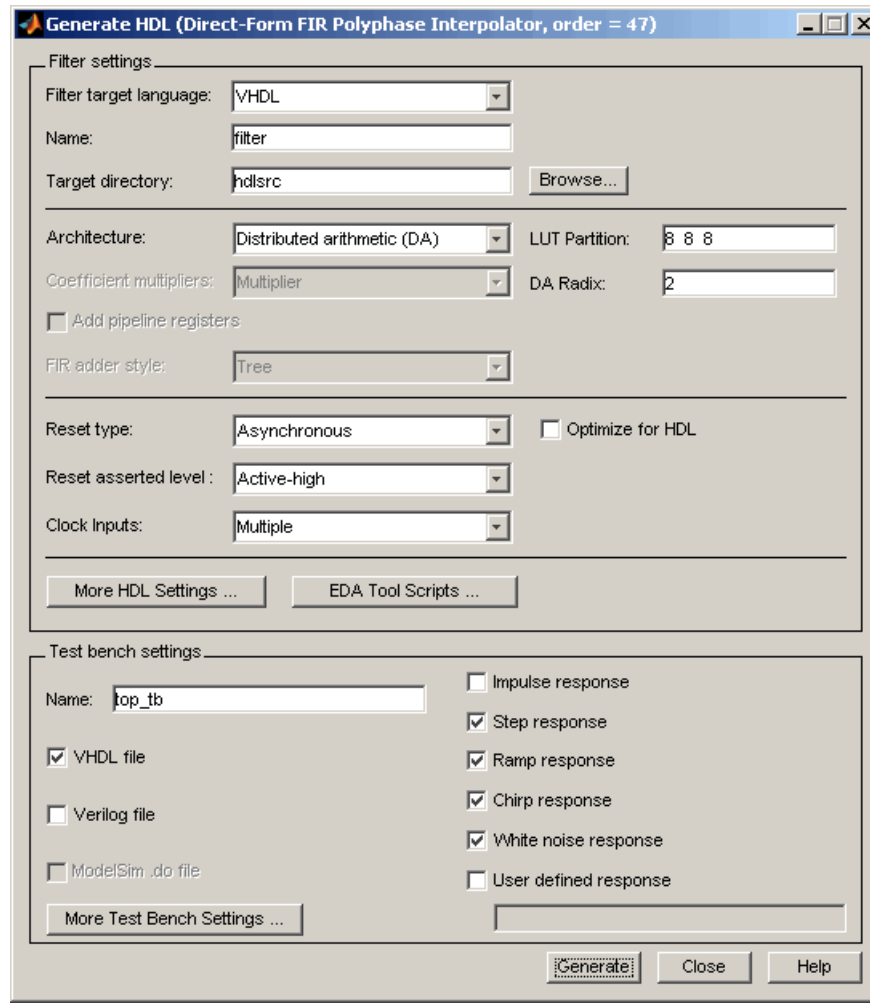
See “Generating Scripts for EDA Tools” for a detailed description of script generation options.

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.

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.



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

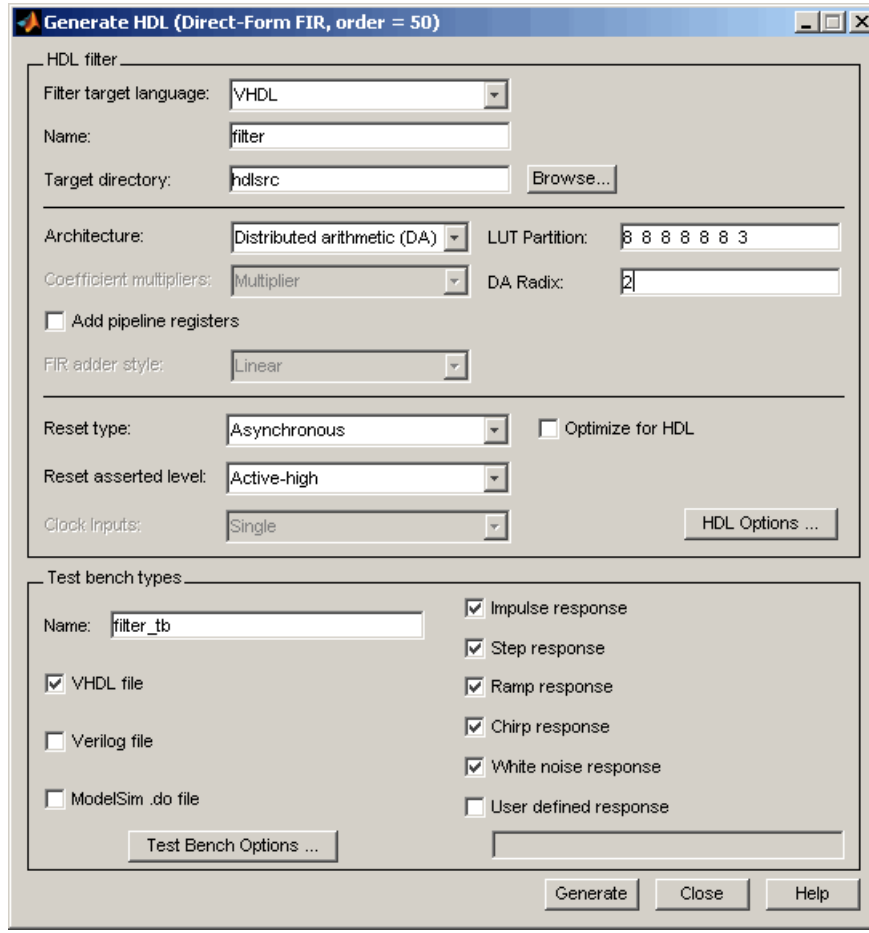
The coder now supports Distributed Arithmetic (DA) in HDL code generated for several single-rate and multirate FIR filter structures. DA is a widely-used technique for implementing sum of products computations without use of multipliers. Designers frequently use DA to build efficient Multiply-Accumulate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



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`

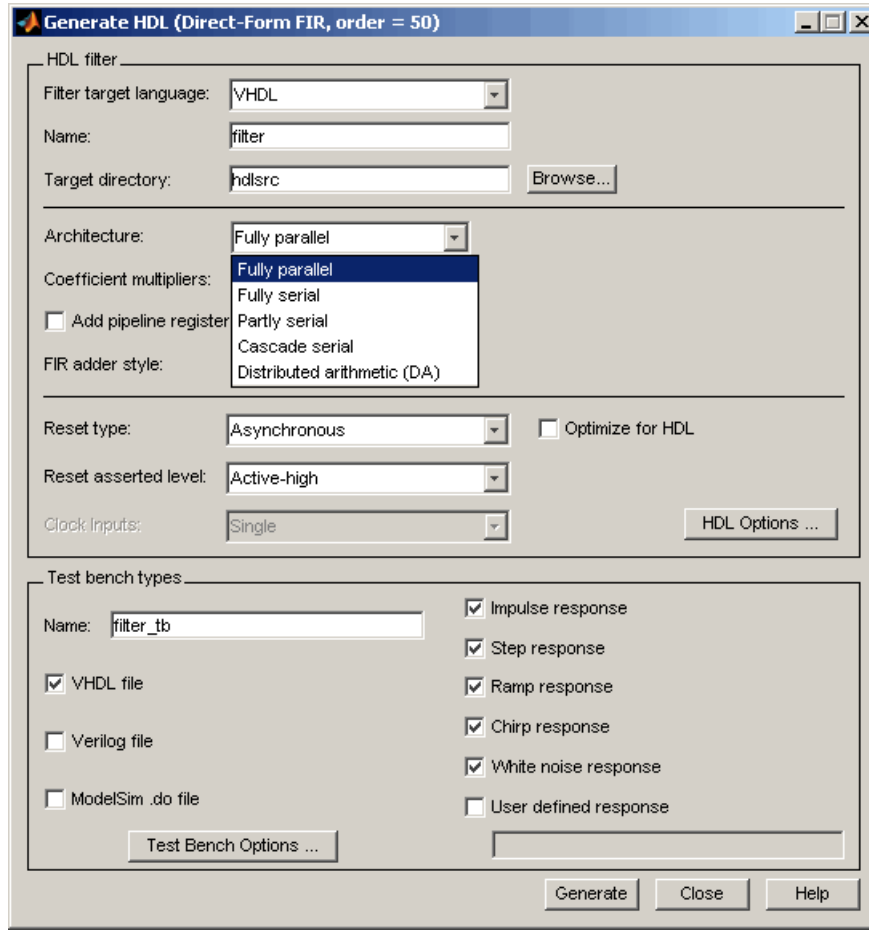
The following table summarizes the filter types that are available for parallel and serial architecture choices. See "Speed vs. Area Optimizations for FIR Filters" in the Filter Design HDL Coder User's Guide for a full description of these options.

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`

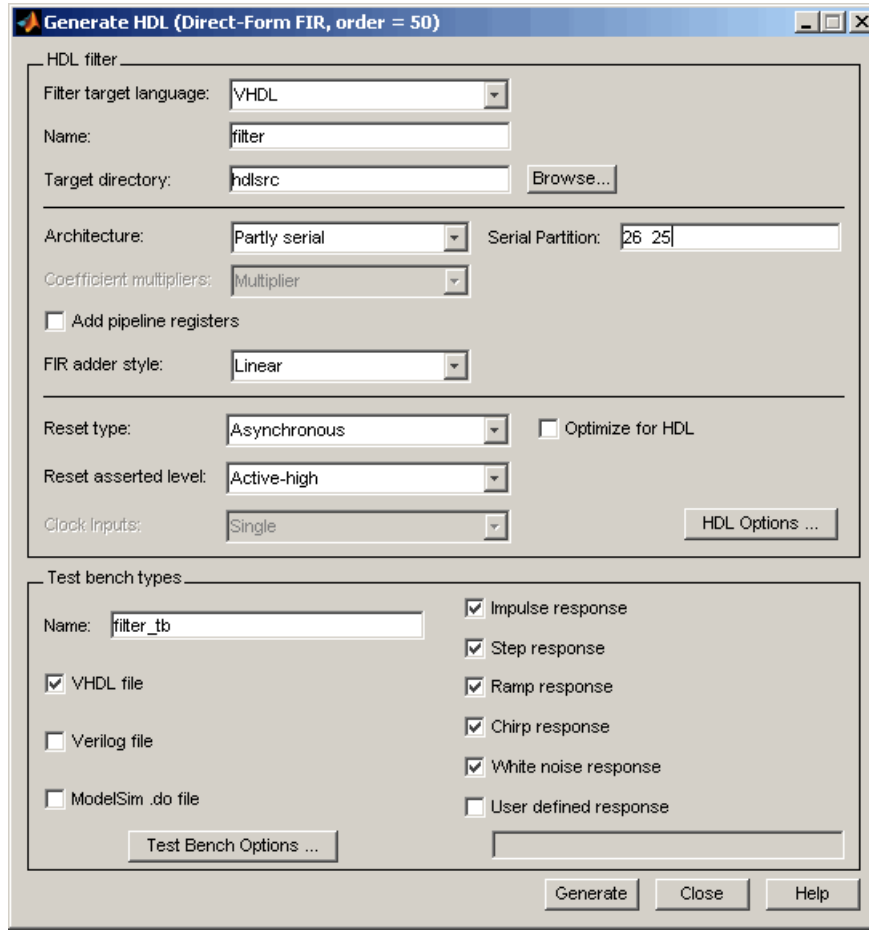
Generate HDL Dialog Box Supports All Parallel and Serial Architecture Options

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.



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 71).

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 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 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

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:

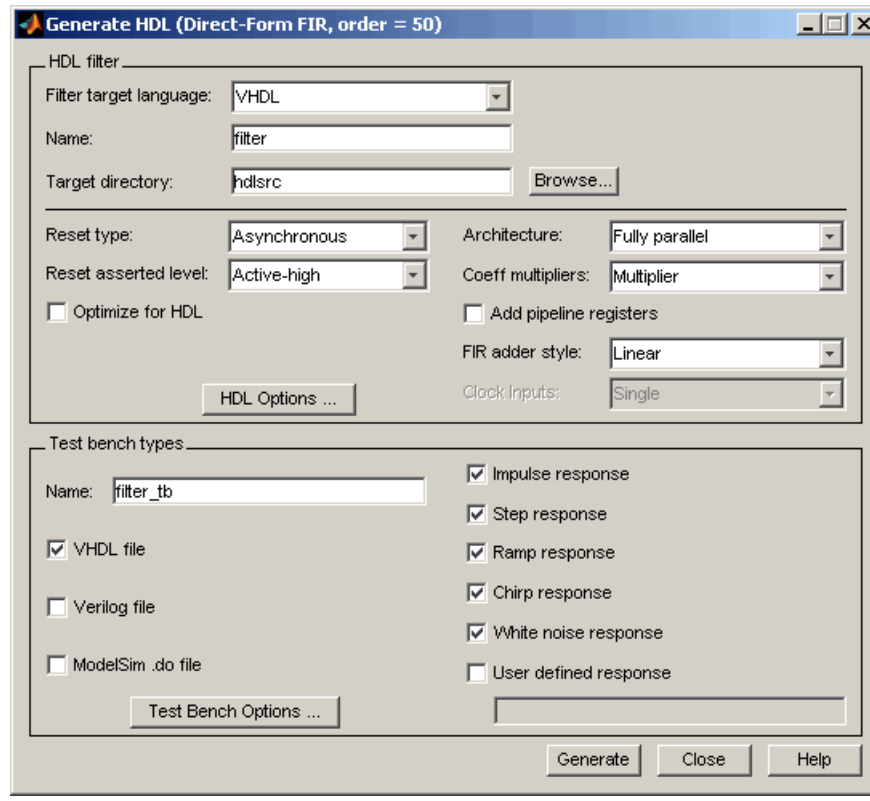
- *Fully parallel*: This is the default option. A fully parallel architecture uses a dedicated multiplier and adder for each filter tap; all taps execute in parallel. A fully parallel architecture is optimal for speed. However, it requires more multipliers and adders than a serial architecture, and therefore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



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.

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 V2.7 (R2010b)	None
V2.6 (R2010a)	See the **Compatibility Considerations** subheading for this new feature or change: • "GenerateCosimModel 'IN' and 'MQ' Property Values Removed" on page 19
V2.5 (R2009b)	See the **Compatibility Considerations** subheading for this new feature or change: • "Graphical User Interface Improved and Revised" on page 20 • "GenerateCosimModel 'IN' and 'MQ' Property Values Replaced" on page 24
V2.4 (R2009a)	See the **Compatibility Considerations** subheading for this new feature or change: • "Default Entity Conflict Postfix Changed" on page 31

Version (Release)	New Features and Changes with Version Compatibility Impact
V2.3 (R2008b)	See the **Compatibility Considerations** subheading for this new feature or change: • "-novopt Flag Added to the Default Simulation Command in Generated Compilation Scripts" on page 35 • "ModelSim .do Test Bench Option Removed" on page 35
V2.2 (R2008a)	See the **Compatibility Considerations** subheading for this new feature or change: • "ScaleWarnBits Property No Longer Supported" on page 42 • "ModelSim .do Test Bench Option Deprecated" on page 41
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 49 • "Default Hardware Target for Synthesis Scripts Updated to Virtex-4" on page 50
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 77 • "Clock EnableValue for Test Benches Always Active-High" on page 78
V1.4 (R2006a)	See the **Compatibility Considerations** subheading for this new feature or change: • "Rounding Behavior in Generated HDL Code" on page 82