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 1

Version 2.9 (R2011b) Filter Design HDL Coder Software 4

Version 2.8 (R2011a) Filter Design HDL Coder Software 8

Version 2.7 (R2010b) Filter Design HDL Coder Software 11

Version 2.6 (R2010a) Filter Design HDL Coder Software 18

Version 2.5 (R2009b) Filter Design HDL Coder Software 26

Version 2.4 (R2009a) Filter Design HDL Coder Software 34

Version 2.3 (R2008b) Filter Design HDL Coder Software 38

Version 2.2 (R2008a) Filter Design HDL Coder Software 42

Compatibility Summary for Filter Design HDL Coder Software 52

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
Latest Version V2.9 (R2011b)	Yes Details	Yes Summary	Bug Reports
V2.8 (R2011a)	Yes Details	Yes Summary	No
V2.7 (R2010b)	Yes Details	No	No
V2.6 (R2010a)	Yes Details	Yes Summary	No
V2.5 (R2009b)	Yes Details	Yes Summary	No
V2.4 (R2009a)	Yes Details	Yes Summary	No
V2.3 (R2008b)	Yes Details	Yes Summary	No
V2.2 (R2008a)	Yes Details	Yes Summary	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.

Documentation on the MathWorks Web Site

Related documentation is available on `mathworks.com` for the latest release and for previous releases:

- Latest product documentation
- Archived documentation

Version 2.9 (R2011b) Filter Design HDL Coder Software

This table summarizes what's new in Version 2.9 (R2011b)

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes See Summary.	Bug Reports

New features and changes introduced in this version are:

- "HDL Code Generation for Digital Up and Down Converter System Objects" on page 4
- "Multiplier Pipelining Extended to Serial Implementations" on page 5
- "Serial Implementations for IIR Filter (Direct form II Second Order Sections)" on page 5
- "Complex Data Type Support for Serial Architectures of FIR Decimator and Interpolators" on page 6
- "Conversion of Error and Warning Message Identifiers" on page 6
- "Functions and Function Elements Being Removed" on page 7

HDL Code Generation for Digital Up and Down Converter System Objects

With release R2011b, You can generate HDL code for DUC and DDC System objects. This capability is limited to code generation at the command line only.

Note the following:

- The property InputDataType has been added to the command line interface. This property accepts only numerictype values.

```
hDDC = dsp.DigitalDownConverter; generatehdl(hDDC, 'InputDataType', nu
```

- Input and Output port names may not be set by you and default to "ddc_in" and "ddc_out".
- Inputs and outputs are always registered.
- A data valid signal is generated at the top DDC/DUC level. For DDC it is "ce_out" and is tied to the corresponding "ce_out" signal from the decimating filtering cascade. For DUC, it is "ce_out_valid" and is tied to the corresponding "ce_out_valid" signal from the interpolating filtering cascade.
- Filtering stages in DDC and DUC can be implemented only in the default fully parallel architectures. Optimization and architecture-specific properties such as SerialPArtition, DALUTPArtition, DARAdix, AddpipelineRegisters, MultiplierInputPipleine and MultiplierOutputPipelines are not supported.

Multiplier Pipelining Extended to Serial Implementations

The Generate HDL dialog box has these new parameter options—**MultiplierInputPipeline** and **MultiplierOutputPipeline**—to support multiplier pipeline registers for serial architectures. Benefits of this feature include:

- Additional pipeline stages added to multipliers
- User-control over how many and where pipeline stages are added
- Works with all supported FIR-based structures

Serial Implementations for IIR Filter (Direct form II Second Order Sections)

Effective with R2011b, you can specify a partly or fully serial architecture for SOS IIR Direct Form II (dfilt.df2sos) filters.

You can specify serial architecture using `generatehdl` via properties FoldingFactor and NumMultipliers, or through the Generate HDL dialog box

(on the **Filter Architecture** tab). For more details, see "Serial Architectures for IIR SOS Filters" and "Specifying Serial Architectures for IIR SOS Filters".

Complex Data Type Support for Serial Architectures of FIR Decimator and Interpolators

Complex data for serial architecture of FIR Decimator (mfilt.firdecim) and FIR interpolator (mfilt.firinterp) is supported via the property InputComplex.

- For the `generatehdl` function, the property name is InputComplex and you can set it to on or off'. To generate the code for complex input data, issue the following command:

```
generatehdl(Hd, 'InputComplex', 'on')
```

- For Simulink® HDL Coder™ filter blocks, the software detects the complex input types or complex coefficients from the model and generates code accordingly if the block supports HDL code generation.

Conversion of Error and Warning Message Identifiers

For R2011b, error and warning message identifiers have changed in Filter Design HDL Coder™.

Compatibility Considerations

If you have scripts or functions that use message identifiers that changed, you must update the code to use the new identifiers. Typically, message identifiers are used to turn off specific warning messages.

For example, the `hdlshared:hdlfilter:abstractdffir:setimplementation:symmetrywarning` identifier has changed to `HDLShared:hdlfilter:symmetrywarning`. If your code checks for `hdlshared:hdlfilter:abstractdffir:setimplementation:symmetrywarning`, you must update it to check for `HDLShared:hdlfilter:symmetrywarning` instead.

To determine the identifier for a warning that appears at the MATLAB prompt, run the following command just after you see the warning:

```
[MSG,MSGID] = lastwarn;
```

This command saves the message identifier to the variable `MSGID`.

Note Warning messages indicate a potential issue with your model or code. While you can turn off a warning, a suggested alternative is to change your model or code so it runs warning-free.

Functions and Function Elements Being Removed

Function or Function Element Name	What Happens When you use the Function or Element?	Use This Instead	Compatibility Considerations
HDL Cosimulation block—Discovery	Errors	Support for Synopsys® Discovery™ has been removed.	Remove or replace all Discovery™ HDL Cosimulation blocks in existing models.

Version 2.8 (R2011a) Filter Design HDL Coder Software

This table summarizes what's new in Version 2.8 (R2011a)

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

New features and changes introduced in this version are:

- "Distributed Arithmetic and Serial Architectures Supported for Cascaded Filters" on page 8
- "generatetb Function Removed" on page 9
- "Support for Serial Architectures for IIR SOS Filters" on page 9
- "Support for Partly Serial Architecture for FIR Interpolators" on page 9
- "Enhanced Synthesis Script Generation Options" on page 10

Distributed Arithmetic and Serial Architectures Supported for Cascaded Filters

The coder now supports Distributed Arithmetic (DA) and serial architectures for cascade filters. You can use this feature only with the command-line interface (`generatehdl`). See "Architecture Options for Cascaded Filters" for more details and some examples.

generatetb Function Removed

R2011a removes the `generatetb` function. To generate a test bench for your HDL filter code, you should instead use the `generatehdl` function. Set the `GenerateHDLTestbench` property to `'on'`, as shown in the following example.

```
generatehdl(Hlp, 'GenerateHDLTestbench', 'on');
```

See the GenerateHDLTestbench reference page for details.

Compatibility Considerations

In Release R2011a, the `generatetb` function will continue to operate, but it will display a warning message when called.

You should replace all calls to `generatetb` in your scripts with calls to `generatehdl`. When you do so, enable test bench generation with the `GenerateHDLTestbench` property, as shown in the preceding example.

See also “Enabling Test Bench Generation”.

Support for Serial Architectures for IIR SOS Filters

Effective with R2011a, you can specify a partly or fully serial architecture for IIR SOS filter structures. At this time, Filter Design HDL Coder supports only Direct Form I SOS filters (df1sos).

You can specify serial architecture using `generatehdl` or through the Generate HDL dialog box (on the **Filter Architecture** tab). For more details, see “Serial Architectures for IIR SOS Filters” and “Specifying Serial Architectures for IIR SOS Filters”.

Support for Partly Serial Architecture for FIR Interpolators

Effective with R2011a, you can specify a partly serial architecture for FIR interpolator filters (`mfilt.firinterp`). See “Speed vs. Area Optimizations for HDL Filter Realizations” for detailed information about parallel and serial architectures supported for HDL code generation.

Enhanced Synthesis Script Generation Options

The **Synthesis script** options in the **EDA Tool Scripts** pane now includes the options **Choose synthesis tool** and **Synthesis file postfix**.

These options let you specify generation of a tool-specific synthesis script. See "Automation Scripts for Third-Party Synthesis Tools" for details.

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
Yes Details below	No	No

New features and changes introduced in this version are:

- "Improved GUI Support for Distributed Arithmetic Architectures" on page 11
- "Enhanced Information Display for Serial Architectures" on page 15
- "Support for Variable Rate CIC Filters" on page 17

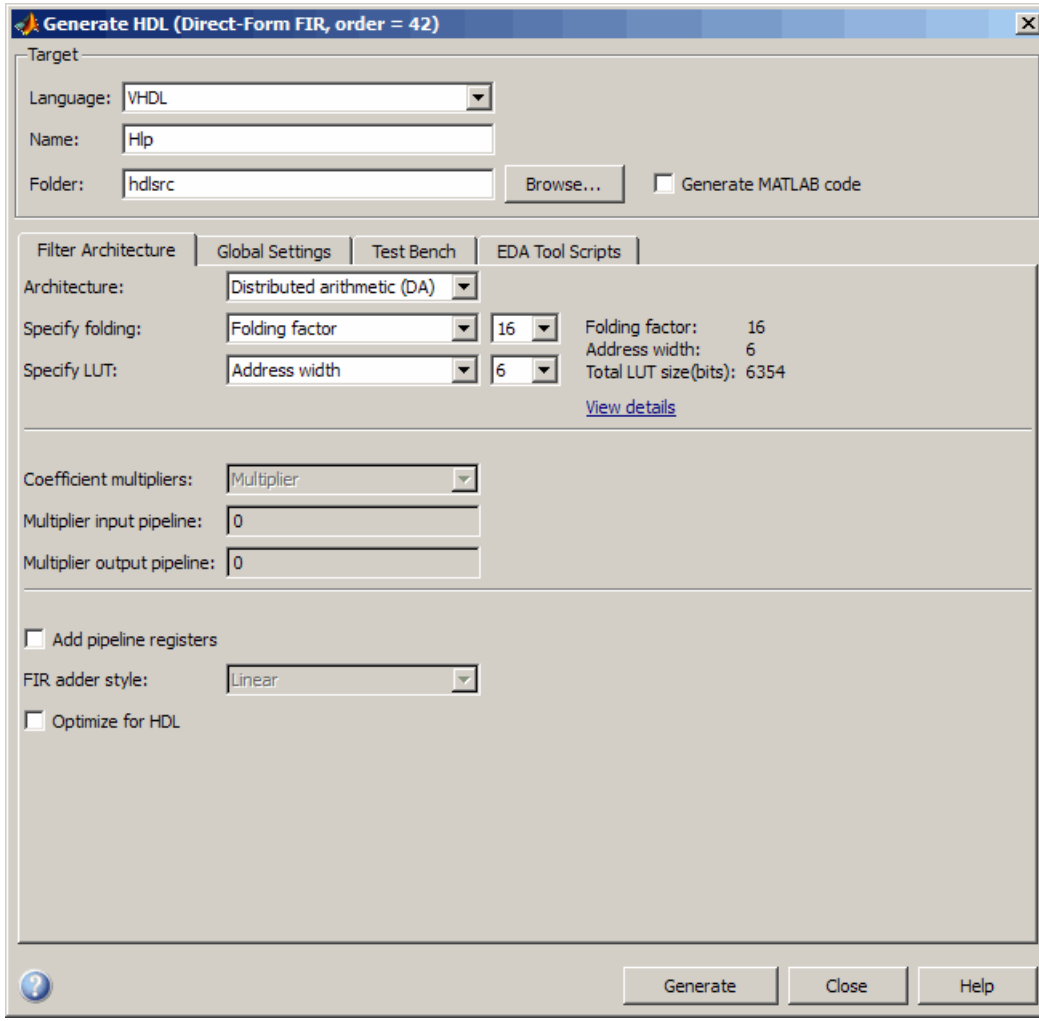
Improved GUI Support for Distributed Arithmetic Architectures

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 folding factor options 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:

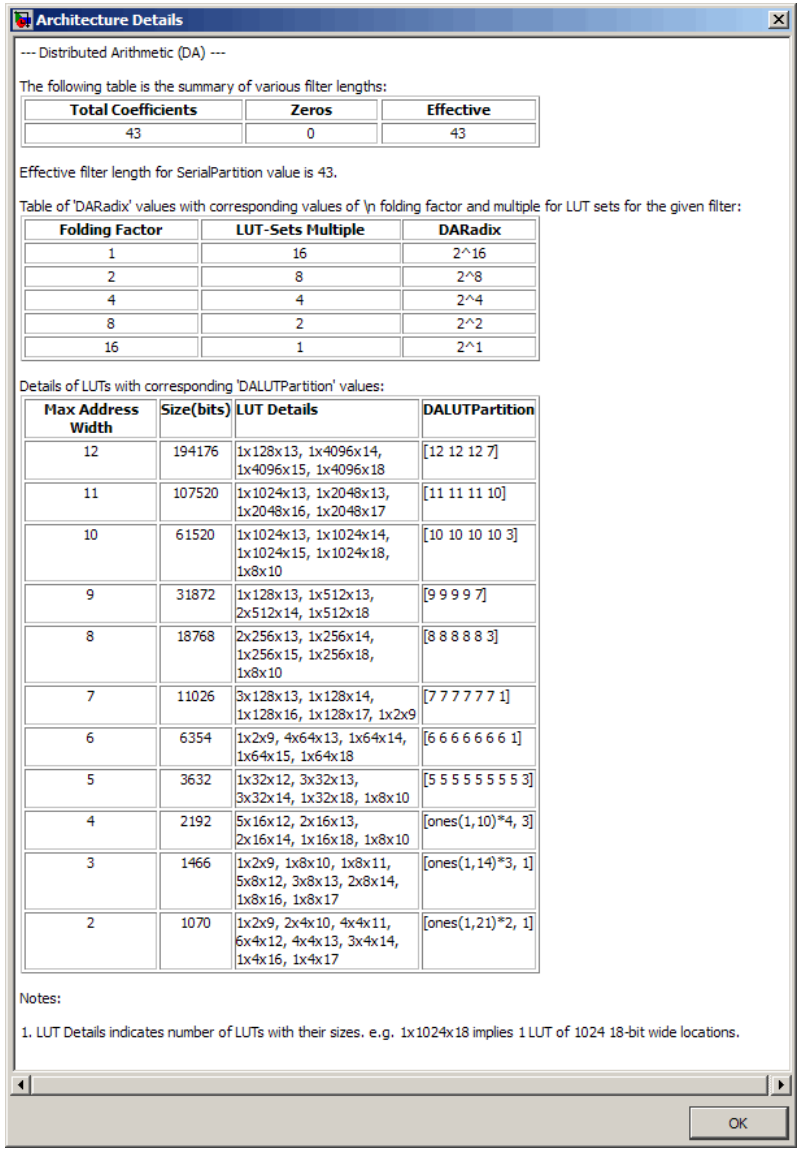
- `Address width` (default): Select from the pulldown menu to the right of **Specify LUT**. The menu contains an exhaustive list of LUT address widths for the filter.
- `Partition`: Enter a vector specifying the number and size of LUT partitions

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 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:

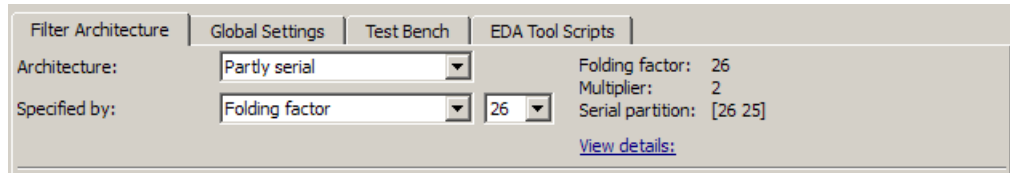
1. LUT Details indicates number of LUTs with their sizes. e.g. 1x1024x18 implies 1 LUT of 1024 18-bit wide locations.

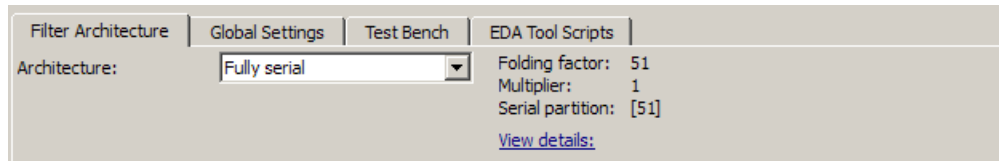
OK

See also in the Filter Design HDL Coder documentation.

Enhanced Information Display for Serial Architectures

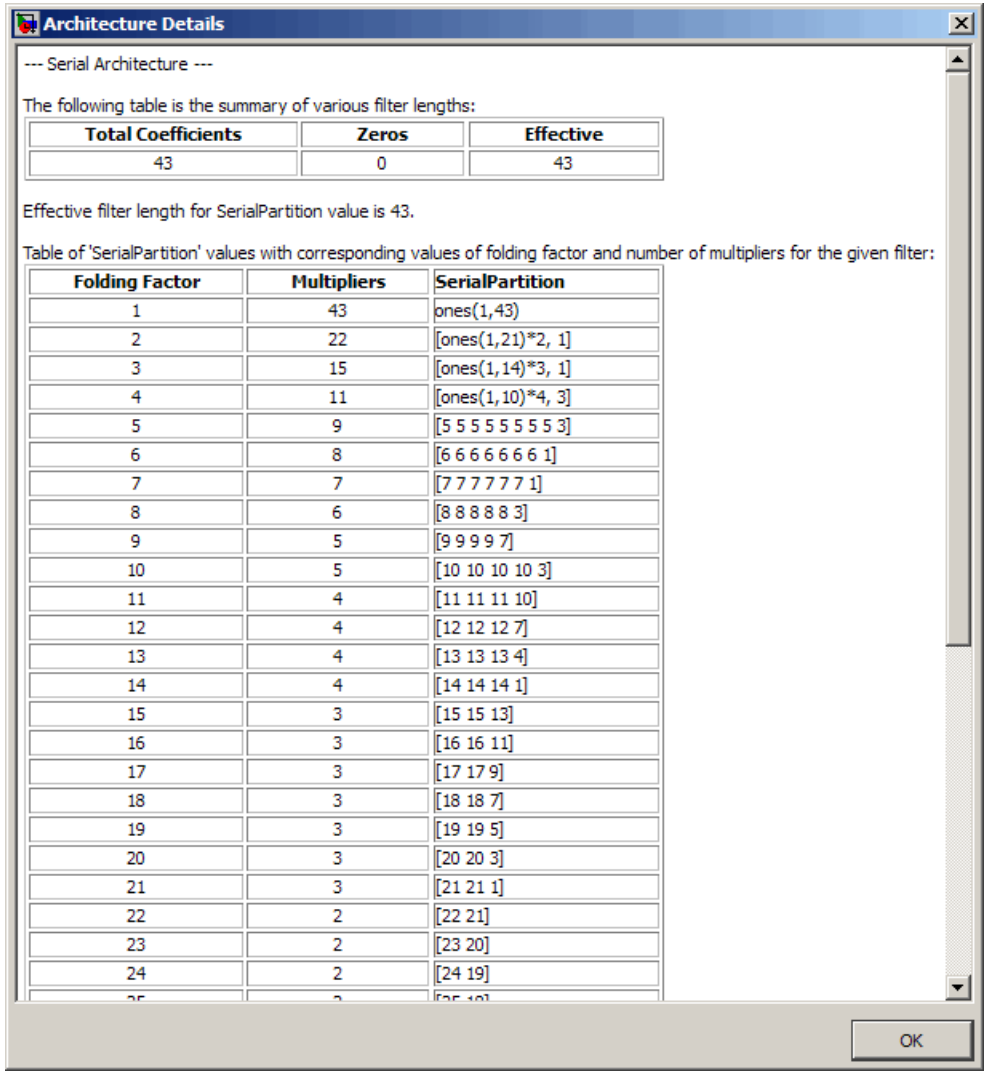
When you select the `Fully serial` or `Partly serial` **Architecture** options, the Generate HDL dialog box now displays a **View details** hyperlink, as shown in the following figures.





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 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.



Architecture Details

--- Serial Architecture ---

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 'SerialPartition' values with corresponding values of folding factor and number of multipliers for the given filter:

Folding Factor	Multipliers	SerialPartition
1	43	ones(1,43)
2	22	[ones(1,21)*2, 1]
3	15	[ones(1,14)*3, 1]
4	11	[ones(1,10)*4, 3]
5	9	[5 5 5 5 5 5 5 5 3]
6	8	[6 6 6 6 6 6 6 1]
7	7	[7 7 7 7 7 7 1]
8	6	[8 8 8 8 8 3]
9	5	[9 9 9 9 7]
10	5	[10 10 10 10 3]
11	4	[11 11 11 10]
12	4	[12 12 12 7]
13	4	[13 13 13 4]
14	4	[14 14 14 1]
15	3	[15 15 13]
16	3	[16 16 11]
17	3	[17 17 9]
18	3	[18 18 7]
19	3	[19 19 5]
20	3	[20 20 3]
21	3	[21 21 1]
22	2	[22 21]
23	2	[23 20]
24	2	[24 19]

OK

See also 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

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
Yes Details below	Yes—Details labeled as **Compatibility Considerations**, below. See also Summary.	No

New features and changes introduced in this version are:

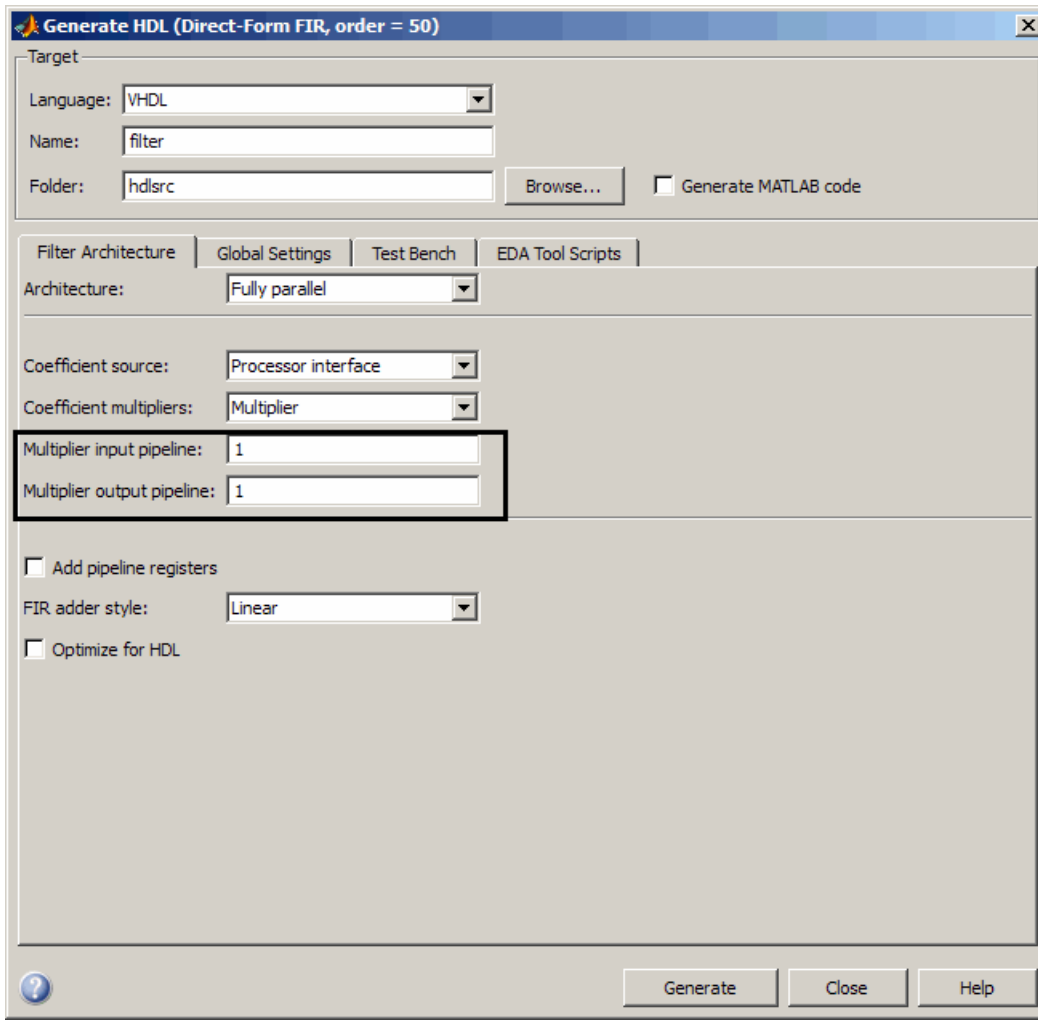
- "Multiplier Input and Output Pipelining for FIR Filters" on page 18
- "Support for Partly Serial Architecture for FIR Decimators" on page 20
- "Enhancements for Serial Architectures" on page 20
- "GUI Support for Programmable FIR FIlter Coefficients" on page 22
- "Option to Suppress Reset Logic Generation for Shift Registers" on page 23
- "GenerateCosimModel 'IN' and 'MQ' Property Values Removed" on page 25

Multiplier Input and Output Pipelining for FIR Filters

Release R2010a lets you specify generation of pipeline stages at multiplier inputs or outputs for all supported 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:

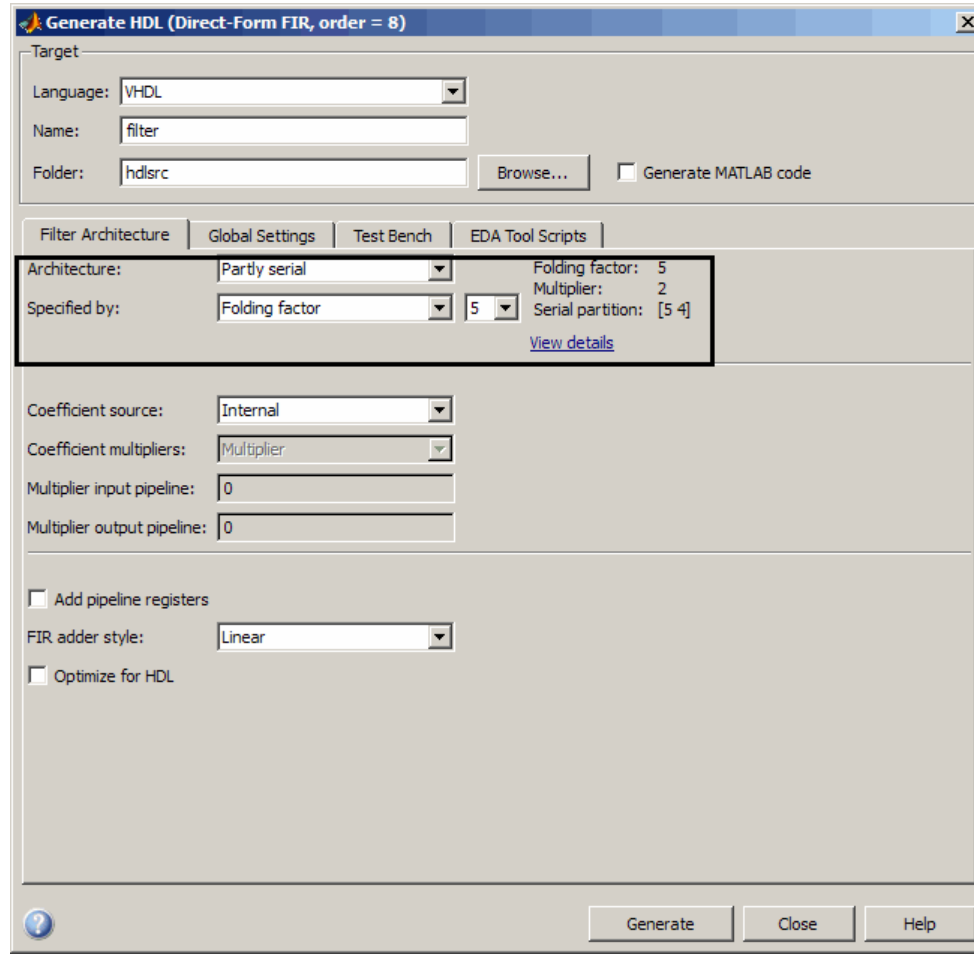
- `'MultiplierInputPipeline', nStages`
- `'MultiplierOutputPipeline', nStages`

Support for Partly Serial Architecture for FIR Decimators

Release R2010a lets you specify a partly serial architecture for FIR decimator filters (`mfilt.firdecim`). See 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.

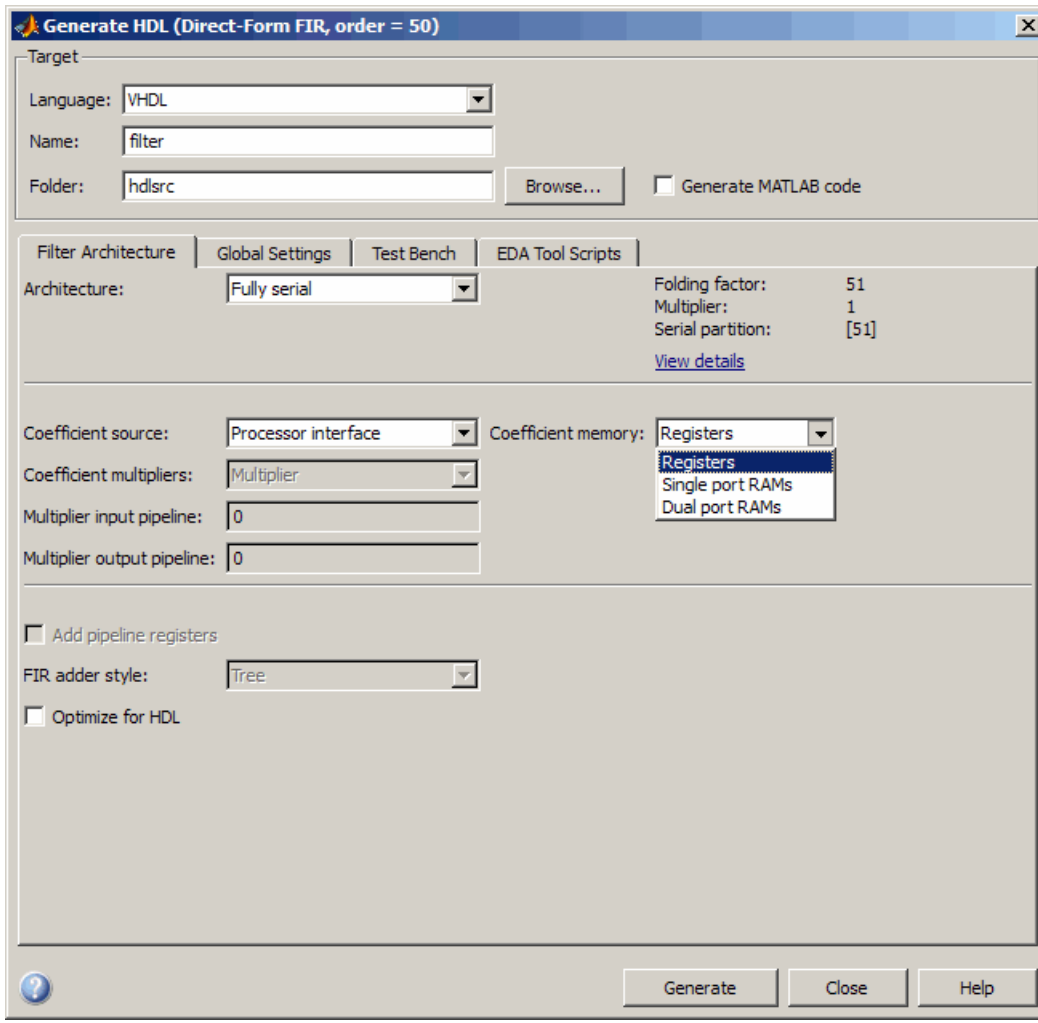
- Specify the desired hardware folding factor *`ff`*, an integer greater than `1`. Given the folding factor, the coder computes the serial partition and the number of multipliers.
- Specify the desired number of multipliers *`nmults`*, an integer greater than `1`. Given the number of multipliers, the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

See 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.

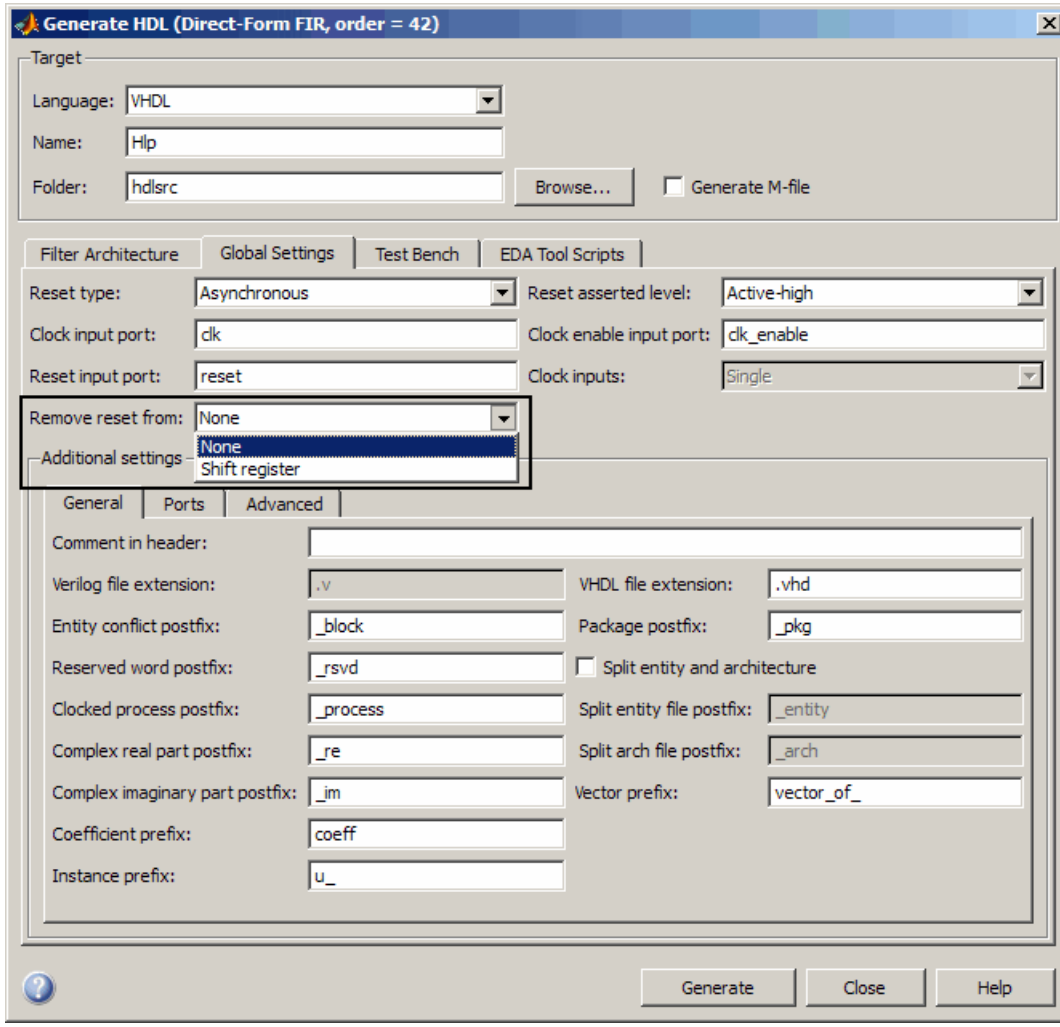


For detailed information, see in the Filter Design HDL Coder documentation.

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 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.

Version 2.5 (R2009b) Filter Design HDL Coder Software

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

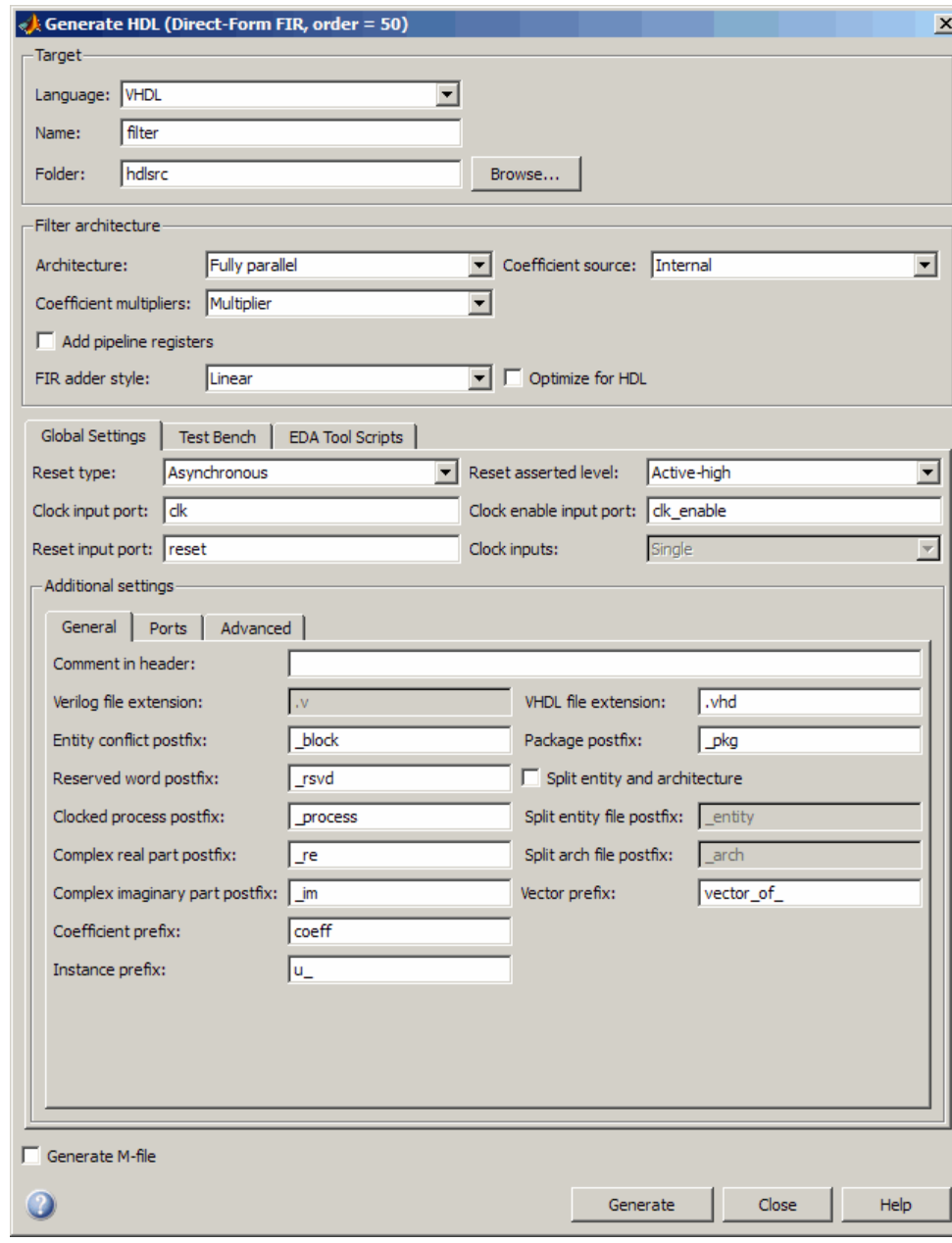
New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes—Details labeled as **Compatibility Considerations**, below. See also Summary.	No

New features and changes introduced in this version are:

- "Graphical User Interface Improved and Revised" on page 26
- "Test Bench GUI Reorganized" on page 28
- "GenerateCosimModel 'IN' and 'MQ' Property Values Replaced" on page 30
- "Extended Complex Data Type Support" on page 31
- "Generation of Model for Cosimulation Now Supports Multirate Filters" on page 32
- "RAM Based Programmable Coefficients Supported for FIR Filters with Serial Architectures" on page 32

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

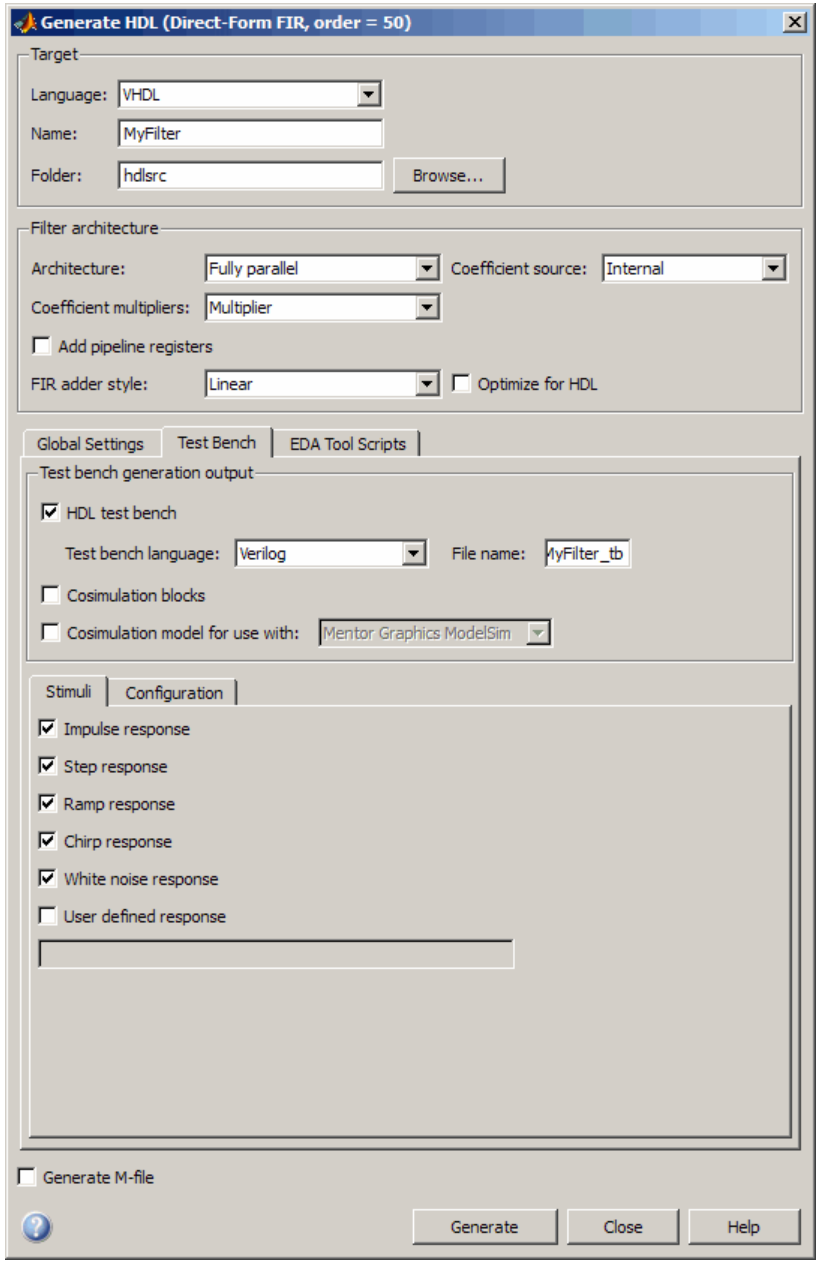
Compatibility Considerations

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 the 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 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 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 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 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
Yes Details below	Yes—Details labeled as **Compatibility Considerations**, below. See also Summary.	No

New features and changes introduced in this version are:

- "Complex Data Type Support for FIR, CIC, and Other Filter Structures" on page 34
- "Generation of Simulink Model for Cosimulation of Generated HDL Code" on page 36
- "Support for Programmable Coefficients for FIR Filters with Serial Architectures" on page 36
- "Support for Programmable Coefficients for IIR Filters" on page 37
- "Default Entity Conflict Postfix Changed" on page 37

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

Filter Structure	Complex Data	Complex Coefficients	Complex Data and Coefficients
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 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 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 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 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 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).

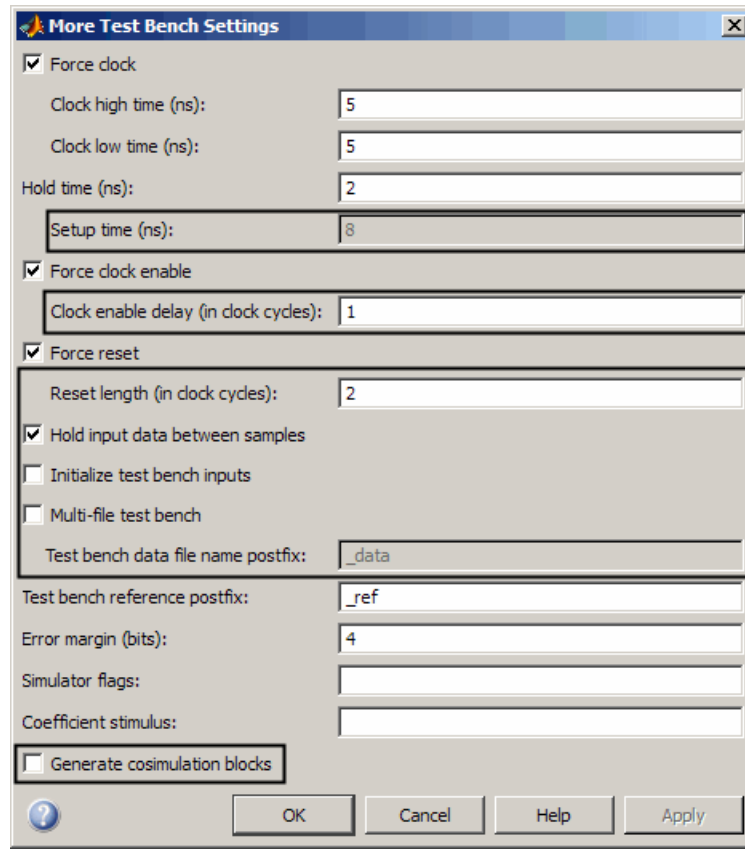
New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes—Details labeled as **Compatibility Considerations**, below. See also Summary.	No

New features and changes introduced in this version are:

- "Test Bench Enhancements" on page 38
- "Distributed Arithmetic Restriction Removed for Symmetrical and Asymmetrical FIR Filters" on page 40
- "-novopt Flag Added to the Default Simulation Command in Generated Compilation Scripts" on page 40
- "ModelSim .do Test Bench Option Removed" on page 41

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 and .	This display-only field does not have a corresponding user-settable command-line property.
Clock enable delay (in clock cycles): See .	TestBenchClockEnableDelay

GUI Option	Command-Line Property
Reset length: See .	ResetLength
Hold input data between samples: See .	HoldInputDataBetweenSamples
Initialize test bench inputs: See .	InitializeTestBenchInputs
Multi-file test bench: See .	MultifileTestBench
Test bench data file name postfix: See .	TestBenchDataPostFix
Test bench reference postfix: See .	TestBenchReferencePostFix
Generate cosimulation blocks: See .	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 and in the Filter Design HDL Coder documentation.

For general information on distributed arithmetic support, see 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
Yes Details below	Yes—Details labeled as **Compatibility Considerations**, below. See also Summary.	No

New features and changes introduced in this version are:

- "Code Generation Support for Multirate Farrow Sample Rate Converter Filters" on page 42
- "Multifile Test Bench Generation" on page 43
- "Additional command-line Properties Supported" on page 43
- "GUI Support for Processor Interface for FIR Filter Coefficients" on page 43
- "generatetb Supports Default Specification of Test Bench Type" on page 46
- "Functions and Properties Being Removed" on page 46
- "ModelSim .do Test Bench Option Deprecated" on page 47
- "ScaleWarnBits Property No Longer Supported" on page 48
- "Summary of GUI Enhancements and Revisions" on page 48

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 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.

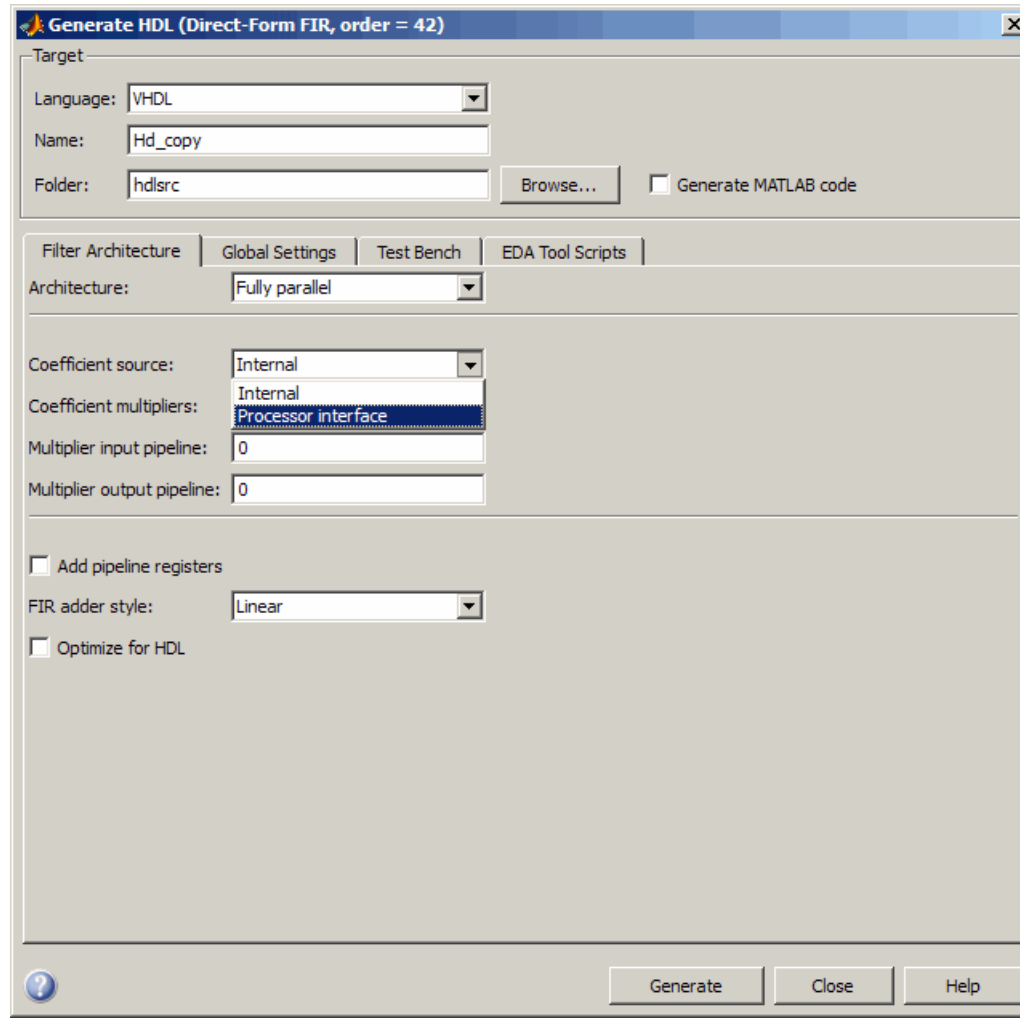
GUI Support for Processor Interface for FIR Filter Coefficients

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.

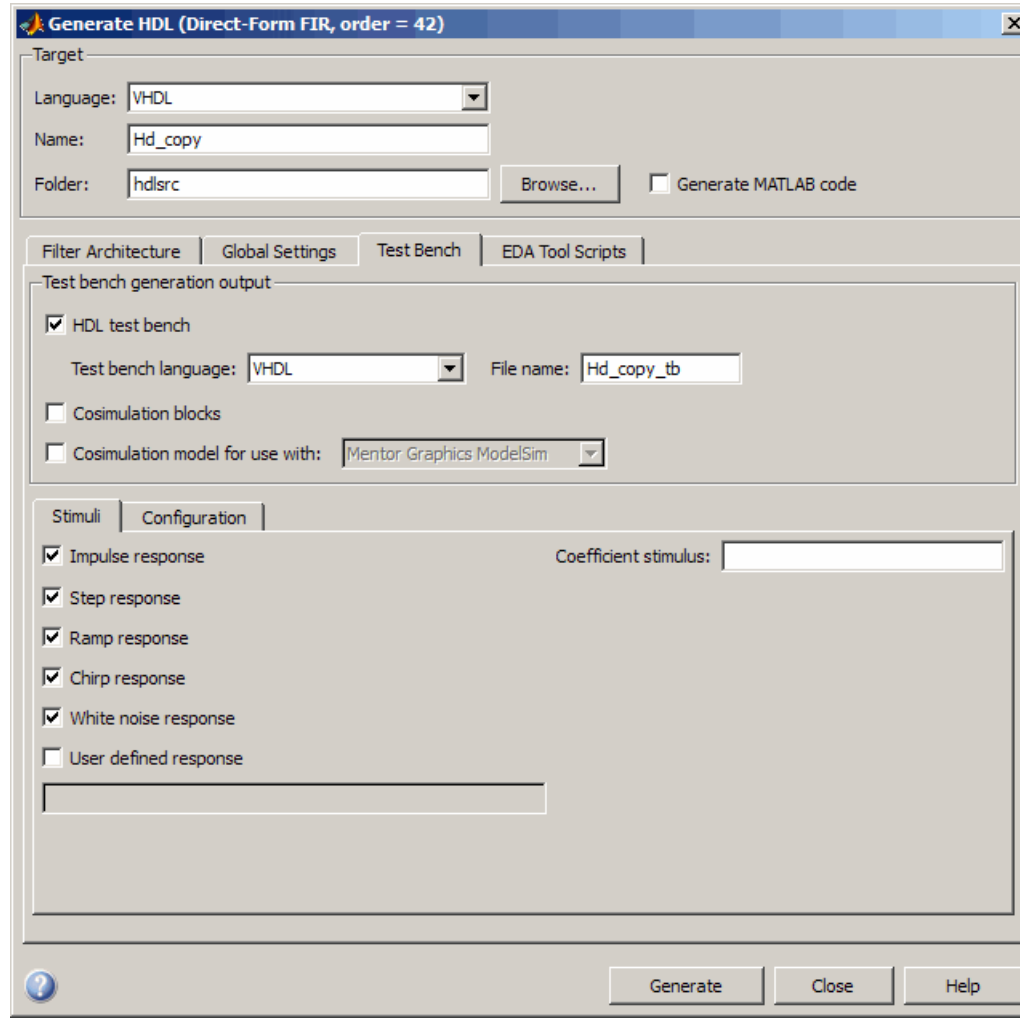
The new GUI options are:

- 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 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 47.
`ScaleWarnBits` property	Property is ignored	No replacement	See "ScaleWarnBits Property No Longer Supported" on page 48.

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 46.

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

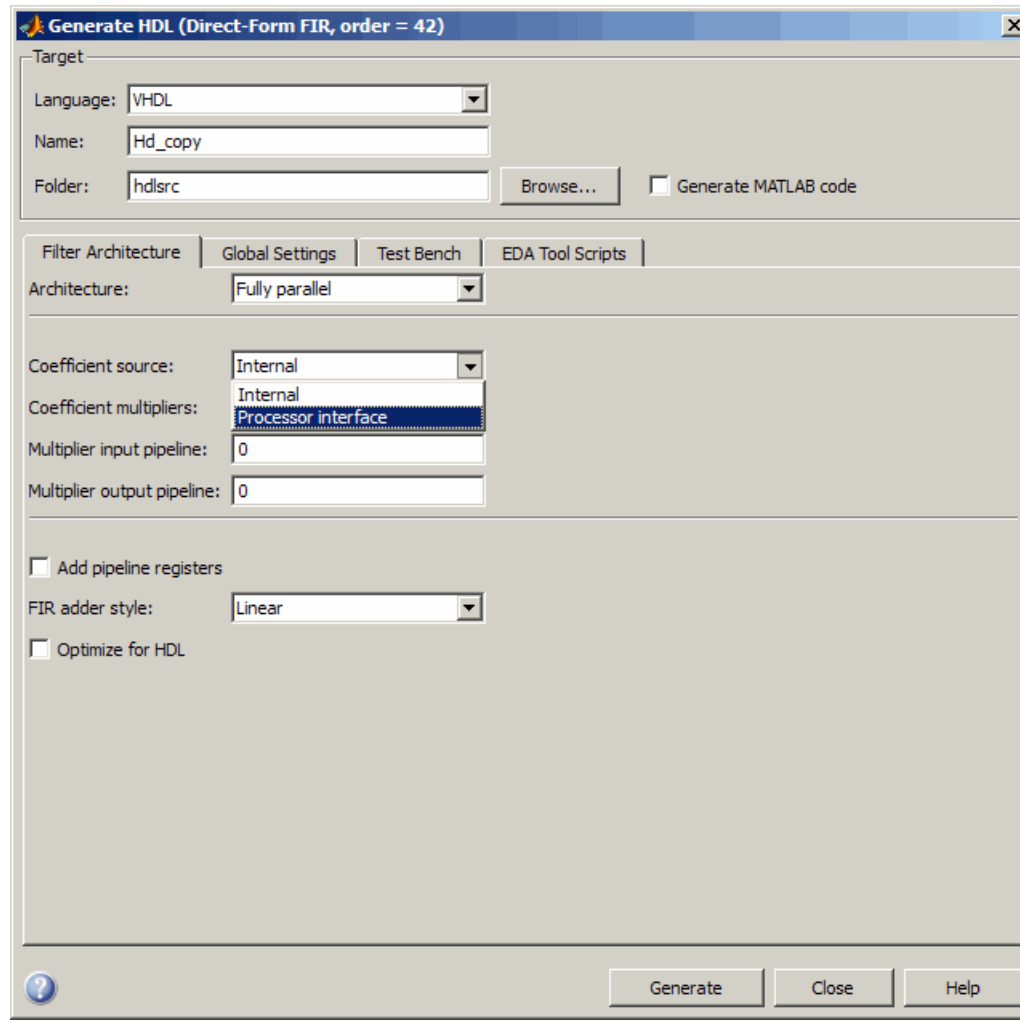
If you have files that contain commands that reference the `ScaleWarnBits` property, such references are ignored. Remove references to `ScaleWarnBits` from your code.

Summary of GUI Enhancements and Revisions

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

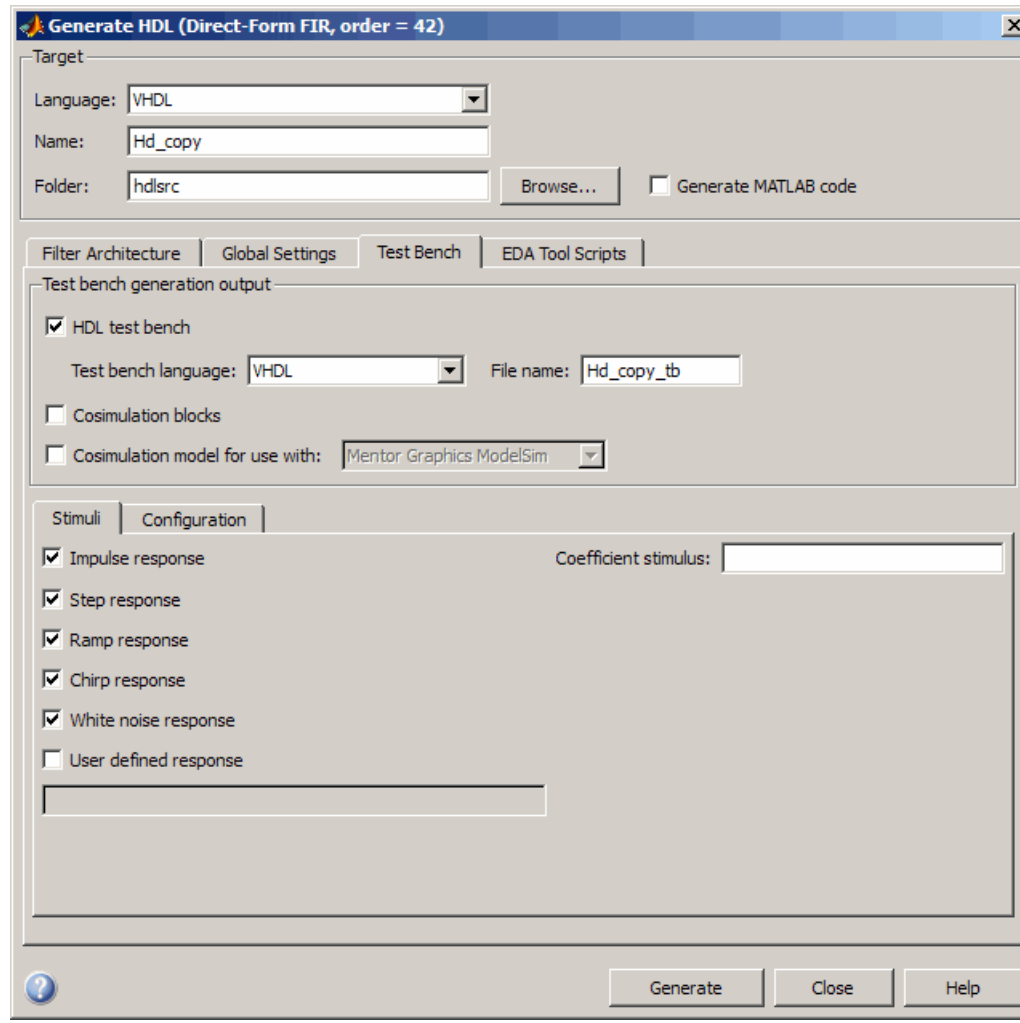
Generate HDL Dialog Box

The Generate HDL dialog box now includes the **Coefficient source** menu. See “GUI Support for Processor Interface for FIR Filter Coefficients” on page 43.



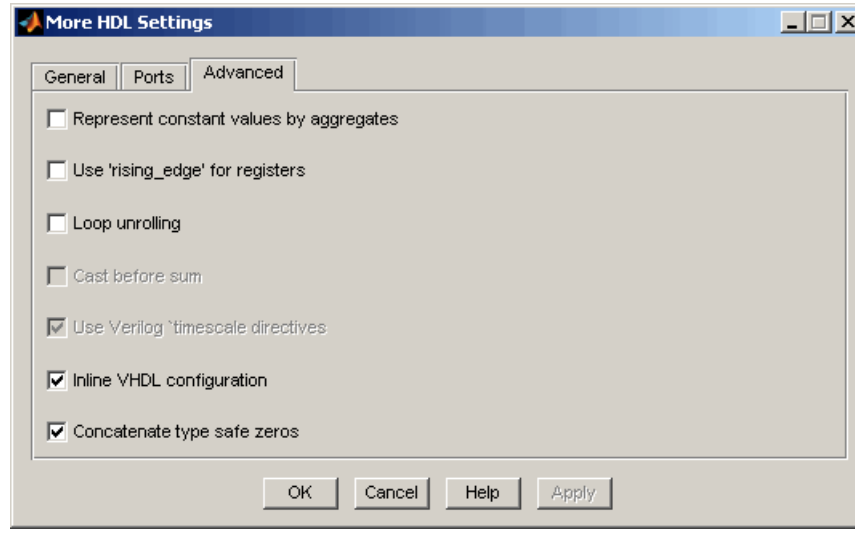
More Test Bench Settings Dialog Box

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 43.



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 48.) 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

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 V2.9 (R2011b)	See the **Compatibility Considerations** subheading for this new feature or change: • "Conversion of Error and Warning Message Identifiers" on page 6 • "Functions and Function Elements Being Removed" on page 7
V2.8 (R2011a)	See the **Compatibility Considerations** subheading for this new feature or change: • "generatetb Function Removed" on page 9
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 25

Version (Release)	New Features and Changes with Version Compatibility Impact
V2.5 (R2009b)	See the **Compatibility Considerations** subheading for this new feature or change: • "Graphical User Interface Improved and Revised" on page 26 • "GenerateCosimModel 'IN' and 'MQ' Property Values Replaced" on page 30
V2.4 (R2009a)	See the **Compatibility Considerations** subheading for this new feature or change: • "Default Entity Conflict Postfix Changed" on page 37
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 40 • "ModelSim .do Test Bench Option Removed" on page 41
V2.2 (R2008a)	See the **Compatibility Considerations** subheading for this new feature or change: • "ScaleWarnBits Property No Longer Supported" on page 48 • "ModelSim .do Test Bench Option Deprecated" on page 47