Filter Design HDL Coder[™] Release Notes

MATLAB®



R

How to Contact MathWorks



Latest news:

Phone:

www.mathworks.com

Sales and services: www.mathworks.com/sales_and_services

User community: www.mathworks.com/matlabcentral

Technical support: www.mathworks.com/support/contact_us



 \searrow

508-647-7000

The MathWorks, Inc. 1 Apple Hill Drive Natick, MA 01760-2098

Filter Design HDL Coder[™] Release Notes

© COPYRIGHT 2005-2022 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

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

	R2022a
Bug Fixes	
	R2021b
Bug Fixes	
	R2021a
Bug Fixes	
	R2020b
Default HDL simulation command vsim -novopt has changed to vsim voptargs=+acc	- 4-2
	R2020a
Bug Fixes	

R2019b

Bug Fixes

R2018b

Bug Fixes

R2018a

Bug Fixes

R2017b

Bug Fixes

R2017a

Bug Fixes

R2016b

HDL Code Generation for Variable Fractional Delay: Generate HDL codefrom the dsp.VariableFractionalDelay System object12-2

Additional System Objects: Generate HDL from System objects that model CIC compensators, FIR rate converters, and lowpass, highpass, Farrow, and cascade filters	
Explore architectures, and generate test bench stimulus, for filter System objects	13-2
Removal of mfilt objects	13-2

R2015b

HDL Code Generation from FIR filter, biquad filter, FIR interpolation, FIR	
decimation, CIC interpolation, and CIC decimation System objects	
	14-2

R2015a

Bug Fixes

R2014b

Bug Fixes

R2014a

Bug Fixes

R2013b

 Functions and Function Elements Being Removed
 18-2

Functions and Function Elements Being Removed	19-2
i unctions und i unction Elements Deing Removed	10 4

R2012b

Bug Fixes

R2012a

No New Features or Changes

R2011b

HDL Code Generation for Digital Up and Down Converter System Objects	
	22-2
Multiplier Pipelining Extended to Serial Implementations	22-2
Serial Implementations for IIR Filter (Direct form II Second Order Sections)	22-2
Complex Data Type Support for Serial Architectures of FIR Decimator and	
Interpolators	22-2
Conversion of Error and Warning Message Identifiers	22-3
Functions and Function Elements Being Removed	22-3

R2011a

Distributed Arithmetic and Serial Architectures Supported for Cascaded Filters	23-2
generatetb Function Removed	23-2

Support for Serial Architectures for IIR SOS Filters	23-2
Support for Partly Serial Architecture for FIR Interpolators	23-2
Enhanced Synthesis Script Generation Options	23-2

R2010b

Improved GUI Support for Distributed Arithmetic Architectures	24-2
Enhanced Information Display for Serial Architectures	24-4
Support for Variable Rate CIC Filters	24-5

R2010a

Multiplier Input and Output Pipelining for FIR Filters	25-2
Support for Partly Serial Architecture for FIR Decimators	25-3
Enhancements for Serial Architectures	25-3
GUI Support for Programmable FIR Filter Coefficients	25-4
Option to Suppress Reset Logic Generation for Shift Registers	25-5
GenerateCosimModel 'IN' and 'MQ' Property Values Removed	25-5

R2009b

Graphical User Interface Improved and Revised	26-2
Test Bench GUI Reorganized	26-3
GenerateCosimModel 'IN' and 'MQ' Property Values Replaced	26-4
Extended Complex Data Type Support	26-4 26-5
Generation of Model for Cosimulation Now Supports Multirate Filters	26-5

RAM Based Programmable Coefficients Supported for FIR Filters with	
Serial Architectures	26-5

R2009a

Complex Data Type Support for FIR, CIC, and Other Filter Structures	
Properties Supporting Complex Data Types	27-2 27-2
Generation of Simulink Model for Cosimulation of Generated HDL Code	27-2
Support for Programmable Coefficients for FIR Filters with Serial Architectures	27-3
Support for Programmable Coefficients for IIR Filters	27-3
Default Entity Conflict Postfix Changed	27-3

R2008b

Test Bench Enhancements	28-2
Distributed Arithmetic Restriction Removed for Symmetrical and Asymmetrical FIR Filters	28-3
-novopt Flag Added to the Default Simulation Command in Generated Compilation Scripts	28-3
ModelSim .do Test Bench Option Removed	28-3

R2008a

Code Generation Support for Multirate Farrow Sample Rate Converter Filters	29-2
Multifile Test Bench Generation	29-2
Additional command-line Properties Supported	29-2
GUI Support for Processor Interface for FIR Filter Coefficients	29-2

generatetb Supports Default Specification of Test Bench Type	29-4
Functions and Properties Being Removed	29-5
ModelSim .do Test Bench Option Deprecated	29-5
Discontinued Support for ScaleWarnBits Property	29-5
Summary of GUI Enhancements and Revisions	29-6 29-6 29-6 29-7

R2022a

Version: 3.1.11

R2021b

Version: 3.1.10

R2021a

Version: 3.1.9

R2020b

Version: 3.1.8 Bug Fixes

Version History

Default HDL simulation command vsim -novopt has changed to vsim - voptargs=+acc

The default value of the HDLSimCmd property is now '-voptargs=+acc %s.%s\n'.

Version History

Prior to R2020b, the default HDL simulation command was vsim -novopt %s.%s\n. Mentor Graphics[®] ModelSim[®] versions prior to 10.7 support the former syntax. If you use a more recent ModelSim version, you must use the -voptargs=+acc syntax.

R2020a

Version: 3.1.7

R2019b

Version: 3.1.6

R2019a

Version: 3.1.5

R2018b

Version: 3.1.4

R2018a

Version: 3.1.3

R2017b

Version: 3.1.2

R2017a

Version: 3.1.1

R2016b

Version: 3.1 New Features

HDL Code Generation for Variable Fractional Delay: Generate HDL code from the dsp.VariableFractionalDelay System object

You can now generate HDL code from the dsp.VariableFractionalDelay System object[™]. See Single-Rate Farrow Filters.

R2016a

Version: 3.0 New Features

Bug Fixes

Version History

Additional System Objects: Generate HDL from System objects that model CIC compensators, FIR rate converters, and lowpass, highpass, Farrow, and cascade filters

You can now generate HDL code generation from the following System objects:

- dsp.LowpassFilter
- dsp.HighpassFilter
- dsp.FarrowRateConverter
- dsp.FilterCascade
- dsp.CICCompensationDecimator
- dsp.CICCompensationInterpolator
- dsp.FIRRateConverter

See Generate HDL from Filter System Objects.

Explore architectures, and generate test bench stimulus, for filter System objects

You can call hdlfilterserialinfo, hdlfilterdainfo, and generatetbstimulus on a filter System object. These functions require an input data type argument along with the object.

Removal of mfilt objects

Removed mfilt Object Use This System Object Instead mfilt.cascade dsp.FilterCascade mfilt.cicdecim dsp.CICDecimator mfilt.cicinterp dsp.CICInterpolator mfilt.farrowsrc dsp.FarrowRateConverter mfilt.firdecim dsp.FIRDecimator mfilt.firtdecim dsp.FIRDecimator mfilt.firinterp dsp.FIRInterpolator mfilt.firsrc dsp.FIRRateConverter mfilt.holdinterp dsp.CICInterpolator (with NumSections = 1, approximates mfilt.holdinterp) mfilt.linearinterp dsp.CICInterpolator (with NumSections = 2, approximates mfilt.linearinterp)

mfilt objects will be removed in a future release. Instead, use their System object counterparts.

R2015b

Version: 2.10

New Features

HDL Code Generation from FIR filter, biquad filter, FIR interpolation, FIR decimation, CIC interpolation, and CIC decimation System objects

You can generate HDL code from the following System objects:

- dsp.FIRFilter
- dsp.BiquadFilter
- dsp.FIRDecimator
- dsp.FIRInterpolator
- dsp.CICDecimator
- dsp.CICInterpolator

See Generate HDL from Filter System Objects.

R2015a

Version: 2.9.7

R2014b

Version: 2.9.6

R2014a

Version: 2.9.5

R2013b

Version: 2.9.4 Bug Fixes

Functions and Function Elements Being Removed

	What Happens When You Use The Function or Element?	Use This Instead	Compatibility Considerations
FPGA Automation pane in the Generate HDL dialog box	Error		The FPGA Automation pane has been removed.

R2013a

Version: 2.9.3

New Features

Bug Fixes

Functions and Function Elements Being Removed

	What Happens When You Use The Function or Element?	Use This Instead	Compatibility Considerations
FPGA Automation pane in the Generate HDL dialog box	Warns		In a future release, the FPGA Automation pane will be removed.

R2012b

Version: 2.9.2

R2012a

Version: 2.9.1

No New Features or Changes

R2011b

Version: 2.9

New Features

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', numerictype([], 8,7));
generatehdl(hDDC, 'InputDataType', fi(zeros(1,1), 1, 8, 7));
```

- Input and Output port names may not be set by you and default to ddc_in and ddc_out.
- Inputs and outputs are 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, MultiplierInputPipeline and MultiplierOutputPipeline 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 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 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™.

Version History

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?		Compatibility Considerations
HDL Cosimulation block —Discovery	Errors	Discovery [™] has been removed.	Remove or replace Discovery HDL Cosimulation blocks in existing models.

R2011a

Version: 2.8 New Features

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.

Version History

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

You should replace 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 (dflsos).

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.

R2010b

Version: 2.7

New Features

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.

📣 Generate HDL (Direct-Form FIR, order = 42)	×
Target	
Language: VHDL	
Name: Hlp	
Folder: hdlsrc Browse Generate MATLAB code	
Filter Architecture Global Settings Test Bench EDA Tool Scripts	
Filter Architecture Global Settings Test Bench EDA Tool Scripts Architecture: Distributed arithmetic (DA)	
Address width: 6	
Specify LUT: Address width View details	
<u>view details</u>	_
Coefficient multipliers: Multiplier	
Multiplier input pipeline: 0	
Multiplier output pipeline: 0	
Add pipeline registers	
FIR adder style:	
C Optimize for HDL	
Generate Close Help	

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 impact 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.

wina tabla ia	the cummers	of various filter longther			
ollowing table is the summary of various filter lengths: Total Coefficients Zeros Effective					
43	liciciitis	0	43		
			-15		
filter lengt	n for SerialPar	tition value is 43.			
'DARadix' va	alues with con	responding values of \n folding	g factor and multiple	for LUT sets for the given filte	
olding Fact	or	LUT-Sets Multiple	DARadix		
1		16	2^16		
2		4	2^8 2^4		
8		2	2 4		
16		1	2^1		
611 m		DALL TRANKES - Local Sector			
Address		'DALUTPartition' values: LUT Details	DALUTPartition		
Nidth	Size(bics)		Dricon driction		
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	[99997]		
8	18768	2x256x13, 1x256x14, 1x256x15, 1x256x18, 1x8x10	[888883]		
7	11026	3x128x13, 1x128x14, 1x128x16, 1x128x17, 1x2x9	[7777771]		
6	6354	1x2x9, 4x64x13, 1x64x14, 1x64x15, 1x64x18	[66666666666666666666666666666666666666		
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]		
5		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]		
etails indicat	tes number of	LUTs with their sizes. e.g. 1x	1024x18 implies 1 LU	JT of 1024 18-bit wide location	

See also Distributed Arithmetic for FIR Filters.

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.

Filter Architecture	Global Settings	Test Bench	EDA Tool S	Scripts	
Architecture:	Partly serial	-		Folding factor:	26
Specified by:	Folding factor	-	26 💌	Multiplier: Serial partition:	2 [26 25]
				<u>View details:</u>	

Filter Architecture	Global Settings	Test Bench	EDA Tool Scripts	
Architecture:	Fully serial	•	Folding factor: Multiplier: Serial partition:	1
			View details:	

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	1		
43	0		43			
L) L			1		
Effective filter length for Seri	alPartition value is 43.					
Table of 'SerialPartition' value	s with corresponding	values	of folding factor and num	ber of multipliers for the given filter:		
Folding Factor	Multipliers	Ser	ialPartition]		
1	43	one	s(1,43)]		
2	22	[one	es(1,21)*2, 1]]		
3	15		es(1,14)*3, 1]			
4	11	[one	es(1,10)*4, 3]			
5	9	[5 5	5 5 5 5 5 5 3]			
6	8	[6 6	666661]			
7	7	[7 7	77771]			
8	6	[8 8	8883]			
9	5	[9 9	997]			
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 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	•			
25	2	for	101	11	-	
				ОК		
					_	

See also Selecting Parallel and Serial Architectures in the Generate HDL Dialog Box.

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 Variable Rate CIC Filters.

R2010a

Version: 2.6

New Features

Multiplier Input and Output Pipelining for FIR Filters

Release R2010a lets you specify generation of pipeline stages at multiplier inputs or outputs for 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.

📣 Generate HDL (Direct-Form FIR, order = 50)		×
_Target		
Language: VHDL		
Name: filter		
Folder: hdlsrc Browse	Generate MATLAB code	
Filter Architecture Global Settings Test Bench EDA Tool Scripts	=	
Architecture: Fully parallel		
Coefficient source: Processor interface Coefficient multipliers: Multiplier		
Multiplier input pipeline: 1 Multiplier output pipeline: 1		
Add pipeline registers		
FIR adder style:		
C Optimize for HDL		
0	Generate Close	e Help

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 HDL Filter Realizations 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.

📣 Generate H	HDL (Direct-For	m FIR, order = 8)	×				
Target							
Language:	VHDL	VHDL					
Name:	firfilt_copy	filt_copy					
Folder:	hdlsrc	Browse Generate MATLAB code					
Filter Arch	itecture Glo	obal Settings Test Bench EDA Tool Scripts					
Architecture	:	Partly serial Folding factor: 5 Multiplier: 2					
Specified by	<i>ı</i> :	Folding factor 5 Serial partition: [5 4]					
		View details	- =				
Coefficient s	source:	Internal 💌					
Coefficient r	multipliers:	Multiplier					
Multiplier in	put pipeline:	0					
Multiplier ou	utput pipeline:	0					
🔲 Add pipe	line registers						
FIR adder st	tyle:	Linear 💌					
Optimize for HDL							
			-				
0		Generate Close Help	,				

The **Specified by** pulldown menu lets you define the serial partitioning in 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 HDL Filter Realizations 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.

📣 Generate I	HDL (Direct-Fo	orm FIR, order = 3)	×
Target			^
Language:	VHDL	▼	
Name:	Hd_copy		
Folder:	hdlsrc	Browse Generate MATLAB code	
Filter Arch	itecture G	Slobal Settings Test Bench EDA Tool Scripts	
Architecture	9:	Fully serial Folding factor: 4 Multiplier: 1 Serial partition: [4]	
		View details	=
🗌 Add pipe	line registers	5	
FIR adder s	tyle:	Linear 💌	
Optimize	e for HDL		Ŧ
•		III	Þ
0		Generate Close	Help

For detailed information, see Programmable Filter Coefficients for FIR Filters.

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.

📣 Generate HDL (Direct-Form FIR, order = 42)	×
Target	
Language: VHDL	
Name: Hip	
Folder: hdlsrc	Browse Generate M-file
Filter Architecture Global Settings Test Bench ED	DA 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
Remove reset from: None	
Additional settings - Shift register	
General Ports Advanced	
Comment in header:	
Verilog file extension:	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_	
0	Generate Close Help

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 has discontinued support for the 'IN' and 'MQ' property values. Use the equivalent property values 'Incisive' and 'ModelSim' instead.

Version History

Replace 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.

R2009b

Version: 2.5

New Features

Graphical User Interface Improved and Revised

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

n Generate HDL (Direct-Form FIR, order = 50)	×
Target	
Language: VHDL	
Name: filter	
Folder: hdisrc 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: Clock	enable input port: clk_enable
Reset input port: reset Clock	inputs: Single 💌
Additional settings	
General Ports Advanced	
Comment in header:	
Verilog file extension:	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	
0	Generate Close Help

Version History

Some property labels have changed in the new GUI. The following table 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	e HDL (Dir	ect-Form FI	R, order = 5	0)				×
Target								
Language:	VHDL		•					
Name:	MyFilter							
Folder:	hdlsrc			Brow	se			
Filter archit	ecture							
Architecture	e:	Fully paralle	el	•	Coefficient s	ource:	Internal	•
Coefficient	multipliers:	Multiplier		-				
Add pip	eline registe	ers						
FIR adder s	style:	Linear		•	C Optimize	for HD	L	
HDL te Test be	en generation est bench ench langua ulation block ulation mode Configura e response esponse response	n output ge: Verilog is el for use with	EDA Tool Scrip	•	File name: ModelSim	_	r_tb	
✓ White r	noise respor	nse						
User de	efined respo	onse						
Generate M-file								
Generate	M-file							

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 the 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 the 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 the 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 the 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'.

Version History

Replace 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 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.

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 commandline property) specifies a character vector 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 character vector appended to names generated for the imaginary part of complex signals. The default postfix is '_im'.

See also Using Complex Data and Coefficients.

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 dualport RAM interface for loading coefficients. Previous releases supported programmable coefficients stored in a register file.

For detailed information, see Programmable Filter Coefficients for FIR Filters.

R2009a

Version: 2.4

New Features

Version History

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 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 character vector 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 character vector 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 complete details on complex data type support.

Generation of Simulink Model for Cosimulation of Generated HDL Code

The coder supports generation of a Simulink $^{\ensuremath{\mathbb{R}}}$ 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.

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 test bench coefficients to test the interface. In previous releases, these options were supported only for fully parallel FIR filters.

Programmable coefficients are supported for 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 Programmable Filter Coefficients for FIR Filters.

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 test bench 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.dflsos)
- SOS IIR Direct Form I transposed (dfilt.dfltsos)
- SOS IIR Direct Form II (dfilt.df2sos)
- SOS IIR Direct Form II transposed (dfilt.df2tsos)

For detailed information, see Programmable Filter Coefficients for IIR Filters.

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'.

Version History

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

R2008b

Version: 2.3

New Features

Version History

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.

📣 More Test Bench Settings	×
Force clock	
Clock high time (ns):	5
Clock low time (ns):	5
Hold time (ns):	2
Setup time (ns):	8
Force clock enable	
Clock enable delay (in clock cycles):	1
Force reset	
Reset length (in clock cycles):	2
Hold input data between samples	
Initialize test bench inputs	
Multi-file test bench	
Test bench data file name postfix:	_data
Test bench reference postfix:	_ref
Error margin (bits):	4
Simulator flags:	
Coefficient stimulus:	
Generate cosimulation blocks	
ОК	Cancel Help Apply

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
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

GUI Option	Command-Line Property
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.dfsymfir) 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 the reference page for DARadix and Considerations for Symmetrical and Asymmetrical Filters.

For general information on distributed arithmetic support, see Distributed Arithmetic for FIR Filters.

-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 (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.

Version History

If you are using ModelSim 6.0 or an earlier version, you should set the HDLSimCmd property (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 not 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.

Version History

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.

R2008a

Version: 2.2

New Features

Version History

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 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 character vector 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.

A Generate HDL (Direct-Form FIR, order = 42)	×
Target	
Language: VHDL	
Name: Hd_copy	
Folder: hdlsrc Browse	Generate MATLAB code
Filter Architecture Global Settings Test Bench EDA Tool	Scripts
Architecture: Fully parallel	
Coefficient source: Internal	
Coefficient multipliers: Processor interface	
Multiplier input pipeline: 0	
Multiplier output pipeline: 0	
Add pipeline registers	
FIR adder style:	
Coptimize for HDL	
	Generate Close Help

• 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.

	e HDL (Direct-Form FIR, order = 3)	×
Target		Â
Language:	: VHDL	
Name:	Hd_copy	
Folder:	hdlsrc Browse Generate MATLAB code	
Filter Arch		
	ch Generation Output	
✓ HDL te	est bench	
Test b	bench language: VHDL VHDL VHDL VHDL	
Cosimi	nulation blocks	
Cosimi	nulation model for use with: Mentor Graphics ModelSim	H
Stimuli	Configuration	
Impuls	se response Coefficient stimulus:	
Step re	response	
Ramp	response	
Chirp r	response	
White	e noise response	
🔲 User d	defined response	
		-
	Generate Close Help	

For detailed information on these options, see TestbenchCoeffStimulus.

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
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

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		No replacement	See "ModelSim .do Test Bench Option Deprecated" on page 29-5.
ScaleWarnBits property	Property is ignored	No replacement	See "Discontinued Support for ScaleWarnBits Property" on page 29- 5.

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.

Version History

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 29-4.

See also generatetb.

Discontinued Support for ScaleWarnBits Property

Filter Design HDL Coder does not support the ScaleWarnBits property. The corresponding GUI option, **Minimum overlap of scale values (bits)**, has been removed from the **Advanced** pane of the More HDL Settings dialog box.

Version History

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 29-2.

♦ Generate HDL (Direct-Form FIR, order = 42)	X
-Target	
Language: VHDL	
Name: Hd_copy	
Folder: hdlsrc Browse	Generate MATLAB code
Filter Architecture Global Settings Test Bench EDA Tool Scrip	ts
Architecture: Fully parallel	
Coefficient source: Internal Coefficient multipliers: Processor interface Multiplier input pipeline: 0 Multiplier output pipeline: 0 Add pipeline registers FIR adder style: Linear Coptimize for HDL	
3	Generate Close Help

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 29-2.

📣 Generate I	HDL (Direct-Form FIR, order = 3)		
Target			
Language:	VHDL		
Name:	Hd_copy		
Folder:	hdlsrc Browse 🖾 Generate MATLAB code		
Filter Architecture Global Settings Test Bench EDA Tool Scripts Test Bench Generation Output Image: Construct to the set of the			
Test b	ench language: VHDL File name: Hd_copy_tb		
Cosimu	Cosimulation blocks		
Cosimu	ulation model for use with: Mentor Graphics ModelSim		
Stimuli	Configuration		
Impuls	e response Coefficient stimulus:		
Step re	esponse		
Ramp response			
Chirp response			
White noise response			
User defined response			
	Generate Close Help		

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 "Discontinued Support for ScaleWarnBits Property" on page 29-5.) 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 ∨HDL configuration	
Concatenate type safe zeros	
OK Cancel Help Apply	