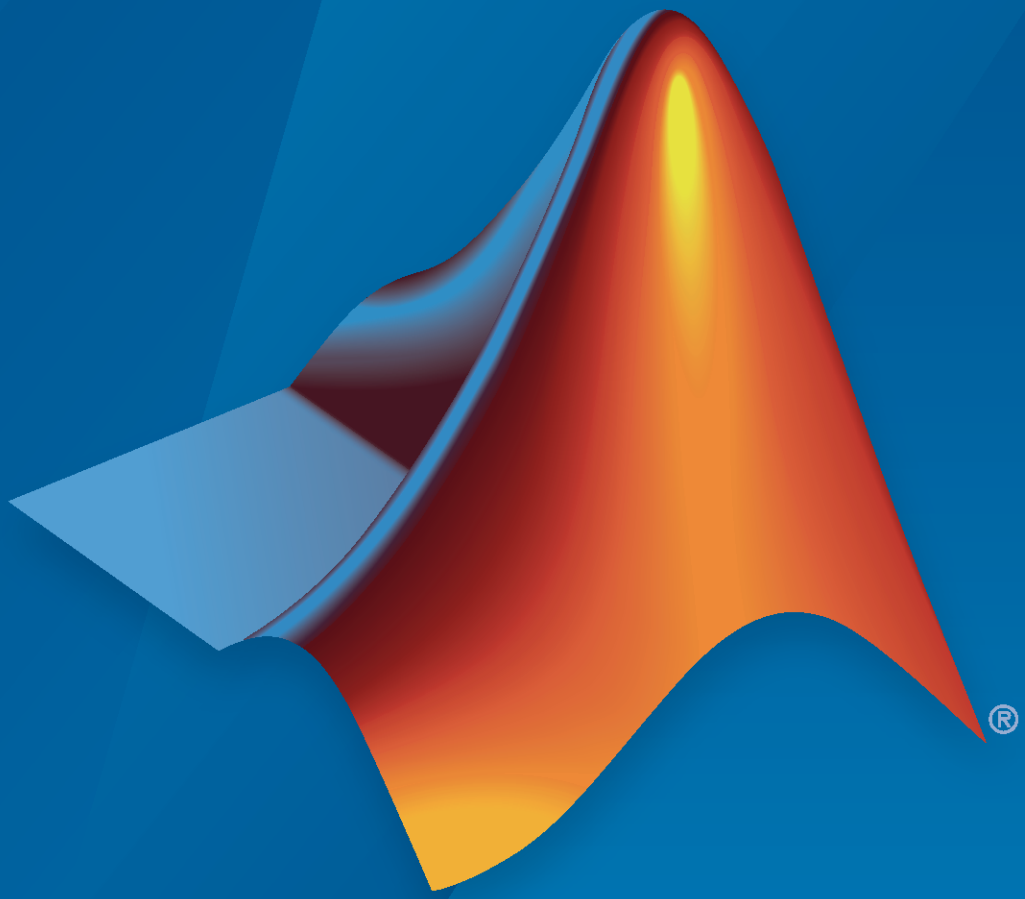
DSP HDL Toolbox™ Release Notes



MATLAB® & SIMULINK®

MathWorks®

How to Contact MathWorks

Latest news: `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`

Phone: 508-647-7000

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

DSP HDL Toolbox™ Release Notes

Contents

R2023a

R2022b

R2022a

R2023a

Version: 1.2

New Features

Bug Fixes

LMS Filter: Minimize error between observed and desired signal

The new LMS Filter block minimizes the error between the observed signal and the desired signal by using mean square error (MSE) criteria. This block provides an interface and architecture for HDL code generation with HDL Coder.

This functionality is also available as the `dsphdl.LMSFilter` System object™.

Minimum resource architecture for biquad filter

The Biquad Filter block now has the option to set the **Filter structure** parameter to `Direct form I fully serial` to implement a fully serial architecture that uses only one multiplier. When you select this option, the block stores the numerator, denominator, and scale values in the same ROM, and you can control the data type of the ROM by using the **Coefficients** data type parameter.

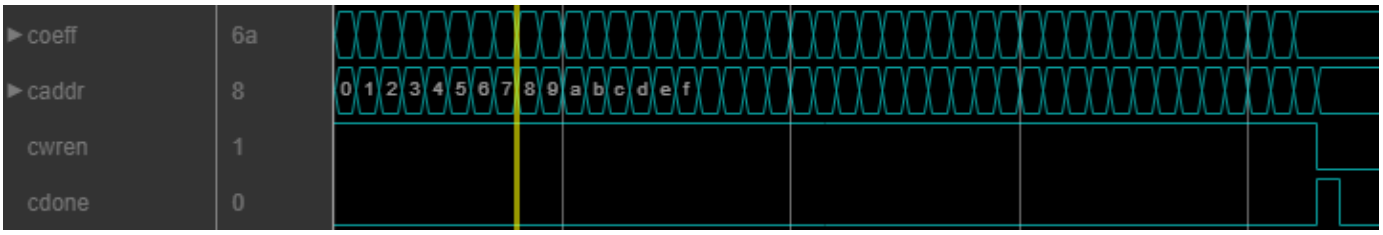
When you select this architecture, the block has an output **ready** signal that indicates when the block is ready for new input. You must only apply input data when the output **ready** signal is `1` (`true`).

Multichannel support for FIR filter

The Discrete FIR Filter block now supports multichannel input. Specify coefficients as an L-by-K matrix, where L is the filter length and K is the number of channels. You can supply input data as a 1-by-K row vector or as scalar input with the channels interleaved in time. The filter shares resources between channels, even if the filter coefficients are different across the channels. If the input data channels have enough cycles between valid input samples, the block can implement the multichannel filter as a single fully serial FIR filter.

Load FIR coefficients using memory-style interface

The Discrete FIR Filter block now offers an optional memory-style interface to load coefficients. To use this interface, set the **Coefficient source** parameter to `Input port (Memory interface)`. You can use this coefficient interface with any filter architecture.



Frequency-domain filtering for FPGA

The “Frequency-Domain Filtering in HDL” example shows how a frequency-domain filter implementation can sometimes be more resource-efficient than a time-domain implementation.

R2022b

Version: 1.1

New Features

Bug Fixes

Partly serial filter architecture for Farrow Filter

The Farrow Rate Converter block now has the **Minimum number of cycles between valid input samples** parameter to enable resource sharing with a partly serial systolic architecture. The internal FIR filter for each filter stage is implemented with a partly serial systolic filter with the sharing factor that you specify.

Increased resource sharing for FIR Interpolator

In R2022a, the minimum number of multipliers that a partly serial FIR Interpolator block could use was *InterpolationFactor* multipliers, which occurred when **Minimum number of cycles between valid input samples** was equal to the filter length. Increasing **Minimum number of cycles between valid input samples** above the filter length resulted in no additional resource sharing. In R2022b, when the **Minimum number of cycles between valid input samples** is greater than the filter length, the block interleaves the coefficients to share multipliers between the polyphase branches. To implement the filter with the minimum number of multipliers, set **Minimum number of cycles between valid input samples** to `Inf`. For real input and real coefficients, this configuration of the filter uses a single multiplier. For complex input, the filter uses three multipliers.

Upsample and downsample frame-based signals

The Upsampler block adds zeros between samples of your input signal to result in a higher output sample rate. The Downsampler block removes samples from your input signal to result in a lower output sample rate. These blocks operate on scalar or frame-based signals and provide hardware-friendly control signals. The blocks support HDL code generation with HDL Coder™.

This functionality is also available as `dsphdl.Upsampler` and `dsphdl.Downsampler` System objects.

High-throughput peak detection example

The Gigasamples-per-Second Correlator and Peak Detector example implements a vector-input correlator and peak detector. The system is suitable for applications such as lidar and mm-wave radar. The model uses the Discrete FIR Filter block.

Near-field communication example

The NFC Digital Downconverter example decimates a 100 megasamples per second ADC signal to 424 kilosamples per second for a near-field communication (NFC) system.

R2022a

Version: 1.0

New Features

Model and simulate hardware-optimized implementations of DSP algorithms

- Algorithms use hardware-friendly control signals such as valid, reset, and backpressure signals.
- Select from hardware architecture options to quickly explore throughput and resource tradeoffs.
- Model realistic pipelining and latency in Simulink®.
- Algorithms such as FFT, FIR filter, decimation and interpolation, Farrow rate conversion, and NCO, are available as Simulink blocks or MATLAB® System objects.

Enable gigasamples-per-second (GSPS) throughput by using optional parallel processing

DSP HDL Toolbox™ blocks and System objects can accept vector input and implement parallel architectures to achieve high sample throughputs with lower clock rates.

Generate HDL code for FPGAs, ASICs, and SoCs (requires HDL Coder)

DSP HDL Toolbox algorithm implementations are optimized to fit well into DSP blocks on FPGAs and are pipelined to minimize critical path and increase synthesized clock frequency.