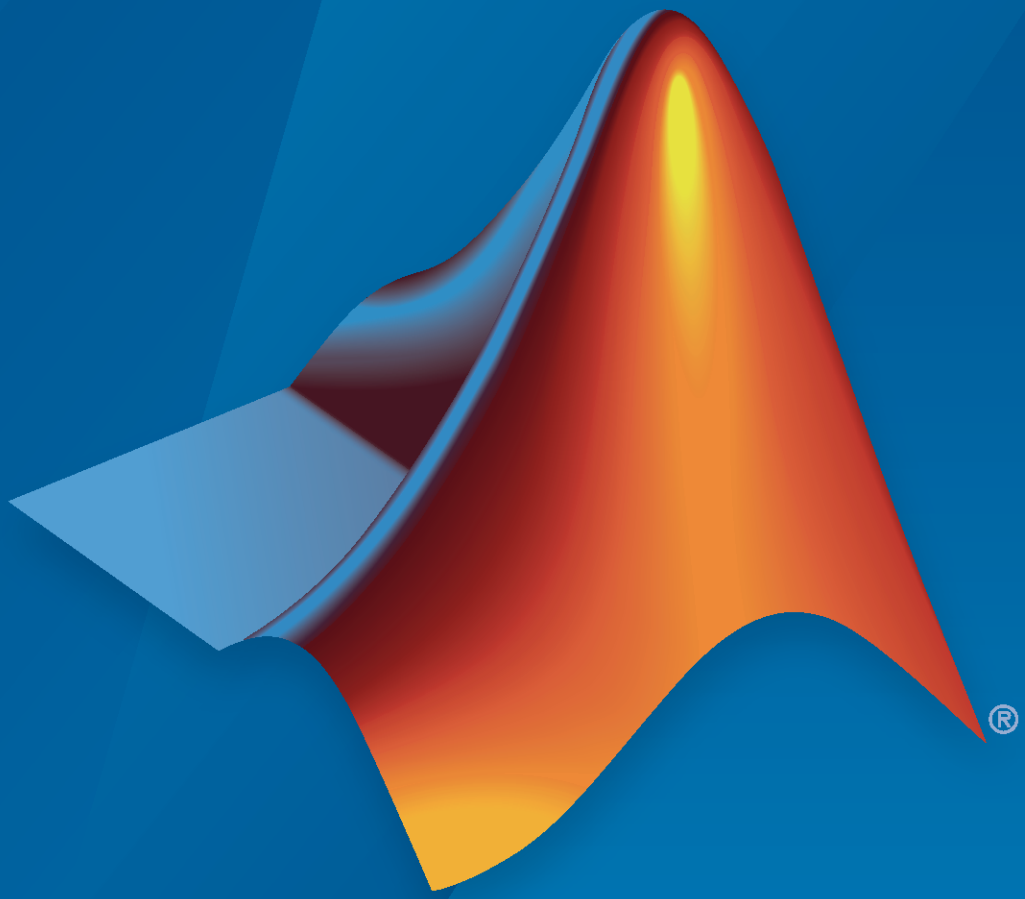
Wireless HDL Toolbox™ Release Notes



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

Wireless HDL Toolbox™ Release Notes

Contents

R2022a

R2021b

R2021a

R2020b

R2020a

R2019b

R2019a

R2018b

R2018a

R2017b

R2022a

DVB-S2 Receiver Reference Application: Implement DVB-S2 receiver on FPGA or ASIC

The “DVB-S2 HDL Receiver” reference application example builds on the “DVB-S2 HDL PL Header Recovery” example and shows how to implement an end-to-end receiver for real-time applications that conforms to the Digital Video Broadcasting Satellite Second Generation (DVB-S2) standard. The application example shows how to implement PL header recovery, symbol demodulation, deinterleaving, forward error correction frame (FECFRAME) decoding, and stream recovery. The DVB-S2 receiver in this example decodes the FECFRAME using the low-density parity-check (LDPC) decoder and Bose-Chaudhuri-Hocquenghem (BCH) decoder HDL blocks according to the DVB-S2 standard. The example implements the stream recovery to decode the baseband frame (BBFRAME) and to get the baseband header (BBHEADER) information and data field decoding. The example also shows how to handle radio frequency (RF) impairments to support DVB-S2 waveforms.

This example is designed using Simulink® blocks and supports HDL code generation with HDL Coder™.

Hardware Acceleration of 5G NR SIB1 Recovery Example: Deploy polar, CRC, and LDPC decoders to FPGA

As an extension to the “NR HDL SIB1 Recovery” example, the “Hardware Accelerators for NR SIB1 Recovery” example shows how to use hardware accelerators to offload decode operations from the processor to the FPGA logic. The decoding subsystems include a blind search of the control channel using polar, a cyclic redundancy check (CRC), and LPDC decoding.

WLAN HDL Receiver Reference Example Enhancements: Support for 40 MHz bandwidth option

The “HDL Implementation of WLAN Receiver” reference example now supports a single-input single-output (SISO) 40 MHz bandwidth option for high-throughput (HT) and very-high-throughput (VHT) frame formats. You can configure 20 MHz and 40 MHz bandwidth options during the compile time and can modify the modulation and coding scheme and frame formats parameters during the run time.

This example is designed using Simulink blocks and supports HDL code generation with HDL Coder.

DVB-S2 HDL LDPC Encoder Example: Implement LDPC Encoder according to DVB-S2 standard

The “DVB-S2 HDL LDPC Encoder” example shows the implementation of an LDPC encoder according to the DVB-S2 standard. The DVB-S2 LDPC Encoder model in this example supports FEC frames of type normal and short for all the supported code rates according to the standard. This example is designed using Simulink blocks and supports HDL code generation with HDL Coder.

CCSDS RS Encoder Block: Encode message into RS codeword according to CCSDS standard

The CCSDS RS Encoder block encodes a message into a Reed-Solomon (RS) codeword according to the Consultative Committee for Space Data Systems (CCSDS) standard. The block supports RS

codewords in the format (255, *k*), where *k* is a message length of 239 or 223. The block also supports shortened message lengths and interleaving depth values of 1, 2, 3, 4, 5, and 8. This block provides an interface and architecture for HDL code generation with HDL Coder.

DVB-S2 BCH Decoder Block: Decode and recover message from BCH codeword

The DVB-S2 BCH Decoder block decodes and recovers messages from a BCH codeword according to the DVB-S2 standard. The block supports scalar inputs and forward error correction (FEC) frames of type normal and short. The block uses hardware-friendly control signals and supports HDL code generation with HDL Coder.

DVB-S2 LDPC Decoder Block: Implement decoding of LDPC codes according to DVB-S2 standard

The DVB-S2 LDPC Decoder block implements layered belief propagation with min-sum approximation and normalized min-sum approximation algorithms for decoding LDPC codes according to the DVB-S2 standard. When you set the **Algorithm** parameter to `Min-sum`, the block uses minimal hardware resources. When you set the **Algorithm** parameter to `Normalized min-sum`, the block provides improved bit error ratio (BER) performance.

The block supports early termination to achieve a faster convergence speed. To use this feature, set the **Decoding termination criteria** parameter to `Early`. The block supports scalar values through the input/output (I/O) interface. It also supports FEC frames of type normal and short with all the code rates supported by the DVB-S2 standard. The block uses hardware-friendly control signals and supports HDL code generation with HDL Coder.

DVB-S2 Symbol Demodulator Block Enhancements

The DVB-S2 Symbol Demodulator block now supports demodulation of a complex constellation symbol to a set of data bits. Also, the block now provides parameter options to select:

- Output type — Scalar or Vector
- Decision type — Hard or Approximate log-likelihood ratio
- Noise variance input port

This block provides an interface and architecture for HDL code generation with HDL Coder.

Symbol Demodulator Block: Demodulate complex constellation symbol to LLR values or data bits

The Symbol Demodulator block demodulates a complex constellation symbol to a set of log-likelihood ratio (LLR) values or data bits. The block supports BPSK, QPSK, 8-PSK, 16-PSK, 16-QAM, 32-PSK, 64-QAM, and 256-QAM modulations. The block supports scalar and vector outputs and provides an input port to specify the noise variance. It also provides parameter options to specify gray mapping and to set the minimum distance between symbols.

This block provides an interface and architecture for HDL code generation with HDL Coder.

Functionality being removed or changed

ltehdlFramesToSamples function has been removed

The `ltehdlFramesToSamples` function has been removed. Use the `whdlFramesToSamples` function instead. The functionality of the two functions is the same. To update your code, replace instances of `ltehdlFramesToSamples` with `whdlFramesToSamples`.

ltehdlSamplesToFrames function has been removed

The `ltehdlSamplesToFrames` function has been removed. Use the `whdlSamplesToFrames` function instead. The functionality of the two functions is the same. To update your code, replace instances of `ltehdlSamplesToFrames` with `whdlSamplesToFrames`.

R2021b

Version: 2.3

New Features

Bug Fixes

5G SIB1 Reference Application: Implement 5G NR SIB1 recovery on SoC or ASIC

The NR HDL SIB1 Recovery reference application builds on the NR HDL MIB Recovery design and shows how to implement system information block type 1 (SIB1) decoding. The design is partitioned between hardware and software and consists of:

- A Simulink model for recovering the master information block (MIB) and for OFDM-demodulating the bandwidth part that carries SIB1. This model supports HDL code generation with HDL Coder.
- MATLAB® code to decode SIB1 from the OFDM grid that is returned by the Simulink model. The MATLAB code supports C code generation with Embedded Coder® for targeting embedded processors.

The NR HDL Downlink Receiver MATLAB Reference example (renamed from "NR HDL Cell Search and MIB Recovery MATLAB Reference") now includes cell search, MIB recovery, and SIB1 recovery.

WLAN HDL Receiver Example: Detect frame format and decode signal and data field according to WLAN standards

The HDL Implementation of WLAN Receiver example is an extension to the WLAN HDL Time and Frequency Synchronization example that was introduced in R2021a. Using the HDL Implementation of WLAN Receiver example, you can detect frame format and decode signal and data fields according to wireless local area network (WLAN) standards. This example supports a single-input single-output (SISO) 20 MHz bandwidth option for non-high-throughput (non-HT), high-throughput mixed mode (HT-MM), and very-high-throughput (VHT) frame formats.

This example is designed using Simulink blocks and supports HDL code generation with HDL Coder.

DVB-S2 PL Header Recovery Example: Implement DVB-S2 HDL receiver synchronization and PL header recovery system on FPGA or ASIC

The DVB-S2 HDL PL Header Recovery example implements Digital Video Broadcasting Satellite Second Generation (DVB-S2) receiver synchronization and a physical layer (PL) header recovery system that can handle radio frequency (RF) impairments to support DVB-S2 waveforms. The example shows how to perform frame synchronization and time, frequency, and phase offset estimation and correction. It also shows how to decode the PL header information.

This example is designed using Simulink blocks and supports HDL code generation with HDL Coder.

HDL Digital Predistorter with LMS Coefficient Estimation: Implement DPD with LMS-based coefficient estimation on FPGA or ASIC

The HDL Implementation of Digital Predistorter with LMS Coefficient Estimation example shows the implementation of a digital predistorter (DPD) with an least mean squares (LMS) based coefficients estimator on an FPGA. This example has a hardware-friendly interface for the Xilinx® Zynq® UltraScale+™ RFSoC ZCU111 evaluation board.

This example is designed using Simulink blocks and supports HDL code generation with HDL Coder.

WLAN LDPC Decoder Block: Implement decoding of LDPC codes according to WLAN standard

The WLAN LDPC Decoder block implements layered belief propagation with min-sum approximation and normalized min-sum approximation algorithms for decoding low-density parity-check (LDPC) codes according to these WLAN standards: 802.11n, 802.11ac, 802.11ax, and 802.11ad. When you set the **Algorithm** parameter to `Min-sum`, the block uses minimum hardware resources. When you set the **Algorithm** parameter to `Normalized min-sum`, the block provides improved bit error ratio (BER) performance.

The block supports scalar and vector values through the input/output (I/O) interface. The block supports early termination to achieve a faster convergence speed. To use this feature, set the **Decoding termination criteria** parameter to `Early`. The block uses hardware-friendly control signals and supports HDL code generation with HDL Coder.

CCSDS RS Decoder Block: Decode and recover messages from RS codeword according to CCSDS standard

The CCSDS RS Decoder block decodes and recovers messages from a Reed-Solomon (RS) codeword according to the Consultative Committee for Space Data Systems (CCSDS) standard. The block supports RS codewords (255, *k*), where *k* is a message length of 239 or 223. The block also supports shortened message lengths and interleaving depth values 1, 2, 3, 4, 5, and 8. This block provides an interface and architecture for HDL code generation with HDL Coder.

DVBS2 Symbol Demodulator Block: Demodulate complex constellation symbols to LLR values

The DVBS2 Symbol Demodulator block demodulates complex constellation symbols to a set of log-likelihood ratio (LLR) values. The block supports pi/2-BPSK, QPSK, 8-PSK, 16-APSK, and 32-APSK modulation types. It also supports multiple code rates according to the DVB-S2 standard. You can configure the modulation type and code rate during runtime while using this block.

This block provides an interface and architecture for HDL code generation with HDL Coder.

APP Decoder Block: Decode coded LLR values using MAP decoding algorithm

The APP Decoder block decodes coded LLR values using the maximum a-posteriori probability (MAP) decoding algorithm and serves as a basic building block to implement a turbo decoder. The block accepts soft inputs and provides soft outputs, which can be useful for iterative decoding. You can set the **Algorithm** parameter to `Log MAP (max)`* or `Max Log MAP (max)` option to select the required decoding algorithm. The block supports terminated and truncated modes and these decoding rates: 1/2, 1/3, 1/4, 1/5, 1/6, and 1/7.

This block provides an interface and architecture for HDL code generation with HDL Coder.

5G NR Parity-Aided Polar Codes: Encode and decode short-length uplink PUCCH messages

The NR Polar Encoder and NR Polar Decoder blocks now support parity-aided uplink control messages on the physical uplink control channel (PUCCH). To use this feature, set the **Link direction** parameter to `Uplink` and the **K** parameter to be in the range from 18 to 25.

R2021a

5G NR HDL MIB Recovery for FR2 Reference Application: Implement 5G NR MIB recovery for millimeter wave frequencies on FPGA or ASIC

The NR HDL MIB Recovery for FR2 reference application builds on the NR HDL MIB Recovery reference application by adding support for millimeter wave frequencies. The 5G NR HDL reference applications now come with Simulink cache files for faster simulation. This design supports HDL code generation with HDL Coder and is ready for deployment to hardware.

OFDM Transmitter and Receiver Reference Applications Enhancements: Implement interleaver and deinterleaver blocks in custom OFDM wireless communications system on FPGA or ASIC

The HDL OFDM Transmitter and HDL OFDM Receiver reference applications now include interleaver and deinterleaver blocks in their designs. This release includes these enhancements:

- Interleaver and deinterleaver blocks are added to the header and data chain blocks in transmitter and receiver designs to improve the burst error performance. For more information about interleaving and deinterleaving, see the HDL Interleaver and Deinterleaver example.
- The header field cyclic redundancy check (CRC) polynomial length is increased from 8 bit to 16 bit to improve the error detection performance.
- An input valid port is added to the transmitter to access payload data from an external source.

These designs support HDL code generation with HDL Coder and are ready for deployment to hardware.

The HDL OFDM MATLAB References example is enhanced to match the new functionality. Use this MATLAB reference to verify Simulink models.

HDL Interleaver and Deinterleaver Example: Design and implement interleaver and deinterleaver blocks for wireless communications system

The HDL Interleaver and Deinterleaver example shows how to design and implement interleaver and deinterleaver blocks for a wireless communications system.

This example is designed using Simulink blocks and supports HDL code generation with HDL Coder.

WLAN HDL Time and Frequency Synchronization Example: Perform packet detection and time and frequency synchronization according to WLAN standard

The WLAN HDL Time and Frequency Synchronization example performs packet detection and time and frequency synchronization operations according to the wireless local area network (WLAN) standards 802.11a, 802.11b, 802.11g, 802.11n, 802.11ac, and 802.11ax. These operations are essential for proper demodulation and decoding of packet information. This example supports 20, 40, and 80 MHz bandwidth options.

This example is designed using Simulink blocks and supports HDL code generation with HDL Coder.

Digital Predistorter Example Enhancements: Add OFDM transmitter and receiver to model design

The HDL Implementation of Digital Predistorter example now includes OFDM transmitter and receiver reference modules in the design model to show the implementation for OFDM wireless communications systems. The Digital Predistorter subsystem supports HDL code generation with HDL Coder.

5G NR CRC Encoder and Decoder Blocks: Implement CRC generation and detection according to 5G NR standard

The NR CRC Encoder block generates cyclic redundancy check (CRC) code bits and appends them to input data. The NR CRC Decoder block detects errors in input data using CRC bits. The blocks support these six cyclic generator polynomials specified in the 5G NR standard: CRC6, CRC11, CRC16, CRC24A, CRC24B, and CRC24C. These blocks use hardware-friendly control signals and support HDL code generation with HDL Coder.

OFDM Equalizer Block: Equalize OFDM data using channel estimate and noise variance

The OFDM Equalizer block equalizes OFDM data using a channel estimate and noise variance in the frequency domain. The block uses zero forcing (ZF) and minimum mean square error (MMSE) equalization methods. It uses hardware-friendly control signals and supports HDL code generation with HDL Coder.

To calculate channel estimates to provide as an input to the block, use the OFDM Channel Estimator block.

5G NR LDPC Decoder Improvements: Support multiple code rates and early termination

The NR LDPC Decoder block now supports multiple code rates to achieve high throughputs with a higher degree of code-rate flexibility.

Also, the block now supports early termination to achieve a faster convergence speed. To use this feature, set the **Decoding termination criteria** parameter to `Early`. The block uses hardware-friendly control signals and supports HDL code generation with HDL Coder.

OFDM Modulator Enhancement: Support windowing for frame-based input

The OFDM Modulator block now supports windowing for frame-based inputs. The block uses hardware-friendly control signals and supports HDL code generation with HDL Coder.

Functionality being removed or changed

`ltehdlFramesToSamples` function has been removed
Errors

`ltehdlFramesToSamples` function has been removed. Use the `whdlFramesToSamples` function instead. The functionality of the two functions is the same. To update your code, replace instances of `ltehdlFramesToSamples` with `whdlFramesToSamples`.

ltehdlSamplesToFrames function has been removed
Errors

`ltehdlSamplesToFrames` function has been removed. Use the `whdlSamplesToFrames` function instead. The functionality of the two functions is the same. To update your code, replace instances of `ltehdlSamplesToFrames` with `whdlSamplesToFrames`.

R2020b

Version: 2.1

New Features

Bug Fixes

5G NR HDL MIB Recovery Reference Application: Implement 5G NR MIB recovery subsystem on FPGA or ASIC

The NR HDL MIB Recovery reference application builds on the NR HDL Cell Search reference application by decoding the broadcast channel and recovering the master information block (MIB). This design supports HDL code generation with HDL Coder and is ready for deployment to hardware.

The NR HDL Cell Search and MIB Recovery MATLAB Reference example shows how to model the hardware algorithms used in these reference applications to help develop the HDL-friendly Simulink models. Use this MATLAB reference to verify Simulink models.

OFDM Transmitter and Receiver Reference Applications: Implement custom OFDM wireless communication system on FPGA or ASIC

The HDL OFDM Transmitter and HDL OFDM Receiver reference applications implement an orthogonal frequency-division multiplexing (OFDM) based wireless communication system designed using Simulink blocks. These designs support HDL code generation with HDL Coder and are ready for deployment to hardware.

The HDL OFDM MATLAB References example shows how to model the hardware algorithms used in these reference applications to help develop the HDL-friendly Simulink models. Use this MATLAB reference to verify Simulink models.

AWGN Channel Example: Implement AWGN generator on hardware to accelerate BER performance evaluation of wireless communication systems

The HDL Implementation of AWGN Generator example shows the implementation of an additive white Gaussian noise (AWGN) generator on hardware to accelerate the bit error rate (BER) performance evaluation of wireless communication systems under an AWGN channel. This example includes a MATLAB reference code for verifying the Simulink model.

The design supports HDL code generation with HDL Coder and is ready for deployment to hardware.

Digital Predistorter Example: Implement digital predistorter to correct nonlinearities and memory effects from a power amplifier

The HDL Implementation of Digital Predistorter example shows the implementation of a digital predistorter for correcting the nonlinearities and memory effects from a power amplifier. The digital predistorter subsystem supports HDL code generation with HDL Coder.

NR CRC Generator Example: Implement CRC Generator according to 5G NR standard

The Encode Streaming Data Using General CRC Generator HDL Optimized Block for 5G NR Standard example shows the implementation of a 5G New Radio (NR) cyclic redundancy check (CRC) generator for encoding streaming data. The NR CRC Generator subsystem supports HDL code generation with HDL Coder.

5G NR Polar Decoder Improvements: Select longer list lengths and use target RNTI to improve decoding performance

In previous releases, the NR Polar Decoder block used a list length of 2. You can now select a list length of 2, 4, or 8 using the **List length** parameter. Choosing a longer list length can improve error correction performance, but can use more hardware resources and increase the decoding latency.

Also, this block now includes an optional port to provide a target RNTI. Providing a target RNTI value can improve decoding performance of downlink control information (DCI) messages.

5G NR Polar Encoder Improvements: Specify message length and rate-matched length using parameters

The NR Polar Encoder block now enables you to set the message length and rate-matched length values using the **Message length (K)** and **Rate-matched length (E)** parameters, respectively. This option uses fewer hardware resources than setting these values using input ports.

5G NR LDPC Decoder and Encoder Improvements: Support scalar input and implement normalized min-sum approximation algorithm

The NR LDPC Decoder block now supports scalar values through the I/O interface. This option uses fewer hardware resources but increases the decoding latency and reduces the throughput compared to using the block with vector inputs.

Also, the block now offers a layered belief propagation with normalized min-sum approximation algorithm to help improve decoding performance. When you set the **Algorithm** parameter to `Normalized min-sum`, the block uses more hardware resources and improves its BER performance when compared to setting the parameter to `Min-sum` option.

The NR LDPC Encoder block now supports scalar values through the I/O interface. This option uses fewer hardware resources but increases the encoding latency and reduces the throughput compared to using the block with vector inputs.

RS Encoder: Encode message data to RS codeword

The RS Encoder block encodes message data to a Reed-Solomon (RS) codeword. The block supports the continuous processing of frames and uses hardware-friendly control signals to handle frame data. The block supports HDL code generation with HDL Coder.

OFDM Modulator Throughput Enhancement: Increase throughput using frame-based input

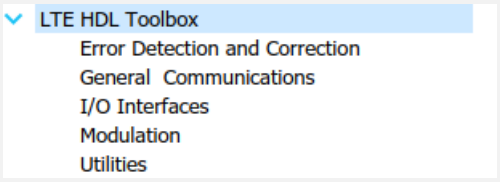
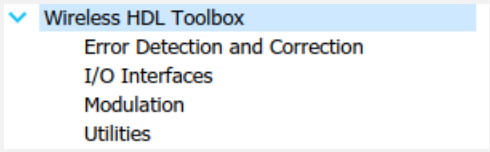
The OFDM Modulator block now supports a frame-based input and windowing for scalar inputs. The block uses hardware-friendly control signals and supports HDL code generation with HDL Coder.

R2020a

LTE HDL Toolbox name change to Wireless HDL Toolbox

The Wireless HDL Toolbox™ name reflects the expansion of the toolbox to include algorithms for 5G New Radio (NR) and general communications designs along with LTE algorithms.

This table shows the updated library hierarchy.

Old Libraries	Updated Libraries
	

All blocks from the LTE HDL Toolbox™ libraries are available in the Wireless HDL Toolbox libraries. The blocks formerly in the **General Communications** library are now split between the **Error Detection and Correction** and **Modulation** libraries. This table shows the updated location of each of these blocks.

Block	Updated Library
Convolutional Encoder	Error Detection and Correction
Depuncturer	Error Detection and Correction
OFDM Demodulator	Modulation
Puncturer	Error Detection and Correction
Viterbi Decoder	Error Detection and Correction

5G NR Signal Synchronization Reference Application: Use primary and secondary synchronization signals (PSS and SSS) to detect connection to valid cell

The NR HDL Cell Search reference application performs primary and secondary signal synchronization (PSS and SSS) as per the 5G NR standard. This design supports HDL code generation with HDL Coder and is ready for deployment to hardware.

The NR HDL Cell Search MATLAB Reference example shows how to model the 5G NR cell search hardware algorithm in MATLAB as a step towards developing a Simulink HDL implementation. The NR HDL Cell Search example uses this MATLAB reference to verify the Simulink reference application model.

5G NR Polar Decoder and Encoder Blocks: Implement polar error correction algorithm according to 5G New Radio (NR) standard

The NR Polar Encoder and NR Polar Decoder blocks implement polar error correction codes according to the 5G NR standard. Both blocks support HDL code generation with HDL Coder.

5G NR LDPC Decoder and Encoder Blocks: Implement low-density parity checking according to 5G New Radio (NR) standard

The NR LDPC Decoder block implements layered belief propagation with the min-sum approximation algorithm for decoding low-density parity-check (LDPC) codes. The NR LDPC Encoder block encodes LDPC codes as per the 5G NR standard. Both blocks support channel coding of downlink and uplink shared channels and paging channel as per the 5G NR standard. These blocks use hardware-friendly control signals and support HDL code generation with HDL Coder.

OFDM Modulator Block: Modulate orthogonal frequency division multiplexed symbols for custom communication protocols

The OFDM Modulator block supports standard and custom communication protocols. The block supports HDL code generation with HDL Coder.

OFDM Channel Estimator Block: Estimate OFDM channel using LS channel estimation algorithm

The OFDM Channel Estimator block estimates an OFDM channel by using the least square (LS) channel estimation algorithm. The block supports averaging and interpolation techniques to improve the accuracy of LS estimation. The block uses hardware-friendly control signals and supports HDL code generation with HDL Coder.

RS Decoder Block: Decode Reed-Solomon codeword to recover message data

The RS Decoder block decodes and recovers a message vector from a Reed-Solomon (RS) codeword. The block supports continuous processing of frames and contains a control bus to handle this frame data. This block provides an interface and architecture for HDL code generation with HDL Coder.

High-throughput OFDM Demodulation: Increase throughput using frame-based input

The OFDM Demodulator block now supports a frame-based input. The block uses hardware-friendly control signals and supports HDL code generation with HDL Coder.

Functionality being removed or changed

`ltehdlFramesToSamples` function will be removed

Warns

`ltehdlFramesToSamples` will be removed in a future release. Use `whdlFramesToSamples` instead. To update your code, replace the instances of `ltehdlFramesToSamples` with `whdlFramesToSamples`. The functionality of the two functions is the same.

`ltehdlSamplesToFrames` function will be removed

Warns

`ltehdlSamplesToFrames` will be removed in a future release. Use `whdlSamplesToFrames` instead. To update your code, replace the instances of `ltehdlSamplesToFrames` with `whdlSamplesToFrames`. The functionality of the two functions is the same.

R2019b

Version: 1.4

New Features

Bug Fixes

MIB and SIB1 Decoder Enhancements: Increase robustness of receiver reference applications

The LTE HDL MIB Recovery and LTE HDL SIB1 Recovery reference applications now support 2x transmit diversity decoding. These enhancements increase the decoding success rate for low quality input signals.

OFDM Demodulator Block: Demodulate orthogonal frequency division multiplexed symbols for custom communication protocols

The OFDM Demodulator block demodulates OFDM symbols for custom communication protocols. The block provides an interface and architecture for HDL code generation and hardware deployment.

FFT 1536 Block: Optimize resource usage for implementing LTE signals with 15 MHz bandwidth option

The LTE standard provides multiple transmission bandwidth options ranging from 1.4 MHz to 20 MHz. For the 15 MHz bandwidth option, you can use a 1536-point fast Fourier transform (FFT) for OFDM modulation and demodulation. The FFT 1536 block implements a 1536-point FFT by using a single 512-point FFT. The block provides an interface and architecture for HDL code generation and hardware deployment.

LTE and 5G NR Symbol Demodulator Blocks: Demodulate complex PSK or QAM symbols for LTE or 5G NR

The LTE Symbol Demodulator and NR Symbol Demodulator blocks demodulate complex phase-shift keying (PSK) or quadrature amplitude modulation (QAM) symbols by using either hard or soft decision decoding. They return message bits or log-likelihood ratios (LLR) based on the modulation type. Each block support these modulation types specified by LTE standard TS 36.211 and 3GPP 5G standard TS 38.211, respectively.

- LTE Symbol Demodulator: BPSK, QPSK, and 16, 64, and 256 QAM
- NR Symbol Demodulator: pi/2-BPSK, BPSK, QPSK, and 16, 64, and 256 QAM

Each block provides an interface and architecture for HDL code generation and hardware deployment.

Convolutional Encoder and Puncturer Blocks: Encode bit streams with continuous, terminated, and truncated modes for custom communication protocols

You can customize the parameters of the Convolutional Encoder and Puncturer blocks to support most communication protocols, including continuous, terminated, and truncated modes. These blocks use hardware-friendly control signals and support HDL code generation with HDL Coder.

LTE Multiantenna Transmitter Reference Application: Generate waveforms for transmission on 1, 4, or 8 antennas

The LTE HDL PBCH Transmitter example is extended to support 1, 4, or 8 antennas.

This example generates the baseband waveform specified by LTE standard TS 36.211. The waveform includes the primary synchronization signal (PSS), secondary synchronization signal (SSS), cell-specific reference signals (Cell-RS), and the master information block (MIB) for transmission through the physical broadcast channel (PBCH). The design supports HDL code generation with HDL Coder and is ready for deployment to hardware.

Sample Rate Conversion for LTE Receiver: Convert data sample rate to LTE sample rate of 30.72 Msps

The Sample Rate Conversion for an LTE Receiver example converts input waveforms to the 30.72 Msps sample rate required by the LTE receiver. The model is compatible with the other LTE receiver reference applications and supports HDL code generation with HDL Coder.

R2019a

Version: 1.3

New Features

Bug Fixes

OFDM Modulator Block: Modulate orthogonal frequency division multiplexing symbols according to the LTE standard

The OFDM Modulator block implements an algorithm for modulating LTE signals specified by LTE standard TS 36.212. It provides an interface and architecture for HDL code generation and hardware deployment. The block modulates an encoded resource grid into time-domain OFDM samples.

LTE Transmitter Reference Application: Generate baseband waveform including PSS, SSS, cell-specific reference, and MIB

The LTE HDL PBCH Transmitter example generates the baseband waveform specified by LTE standard TS 36.211. The waveform includes the primary synchronization signal (PSS), secondary synchronization signal (SSS), cell-specific reference signals (Cell-RS), and the master information block (MIB) for transmission through the Physical Broadcast Channel (PBCH). The design supports HDL code generation with HDL Coder and is ready for deployment to hardware.

LTE and 5G NR Symbol Modulator Blocks: Modulate message bits according to the LTE or 5G standard

The LTE Symbol Modulator block and the NR Symbol Modulator block map groups of bits to complex data symbols according to a dynamic modulation scheme. These supported modulation schemes are specified by LTE standard TS 36.211 and 3GPP 5G standard TS 38.211.

- LTE Symbol Modulator: BPSK,QPSK,16/64/256-QAM
- NR Symbol Modulator: pi/2-BPSK, BPSK,QPSK,16/64/256-QAM

Each block provides an interface and architecture for HDL code generation and hardware deployment.

1536-Point FFT Example: Implement a 1536-point FFT using a single 512-point FFT block

The LTE standard provides multiple transmission bandwidth options ranging from 1.4 MHz to 20 MHz. For 15 MHz transmission bandwidth, the standard requires an FFT length of 1536 for OFDM modulation and demodulation. The HDL Implementation of Streaming 1536-point FFT example shows how to implement a 1536-point FFT using a single 512-point FFT HDL Optimized block.

Variable sample-rate input for OFDM Demodulator block

The OFDM Demodulator block now has a parameter that enables you to choose between applying 30.72 MHz input samples for all NDLRBs or applying input samples at a sample rate determined by the NDLRB of each cell.

TDD Support for SIB Recovery Reference Application: Recover System Information Block type 1 (SIB1) data from time-division duplex LTE signals

The LTE HDL SIB1 Recovery reference application now shows how to decode SIB1 data for LTE networks that use either TDD or FDD. The design supports HDL code generation with HDL Coder and is ready for deployment to hardware.

MIB recovery with strongest cell detection

Previously, the LTE HDL MIB Recovery reference application selected the first detected cell that met a threshold. This reference application now uses a longer synchronization window and tests several cells before choosing the strongest cell ID to decode.

Merge of MIB and MIB TDD reference applications into single model

The LTE HDL MIB Recovery reference application now integrates the decoding of TDD signals, previously included in the LTE Cell Search and MIB Recovery Using TDD Mode reference application.

Separation of LTE PSS and SSS detection into LTE Cell Search reference application

The PSS and SSS detection subsystems used by the LTE HDL MIB Recovery and LTE HDL SIB1 Recovery reference applications are now described in a separate reference application. See LTE HDL Cell Search.

Accelerate Measurement of Bit-Error-Rate (BER)

The Accelerate BER Measurement for LTE HDL Turbo Decoder example shows how to speed up measurement of the bit-error-rate (BER) for the LTE HDL Toolbox Turbo Decoder by using `parsim` to parallelize the simulations across EbNo points.

R2018b

Version: 1.2

New Features

Bug Fixes

SIB1 Reference Application: Implement an LTE System Information Block type 1 (SIB1) recovery subsystem on your FPGA or ASIC

The LTE HDL SIB1 Recovery Example reference application continues the LTE receiver design from the LTE HDL Cell Search and MIB Recovery reference application to perform SIB1 decoding. This design supports HDL code generation with HDL Coder and is ready for deployment to hardware.

Viterbi Decoder and Depuncturer Blocks: Decode convolutionally encoded bit streams using the Viterbi algorithm with puncturing, terminated, and truncated modes

The Viterbi Decoder block supports continuous, terminated, and truncated modes using hardware-friendly control signals. This block provides an optional reset port for use with continuous mode and supports punctured code rates by providing an erasure port. The Depuncturer block accepts a puncture vector as either a port or property and provides an erasure output signal.

Both blocks support HDL code generation with HDL Coder.

Reset port for OFDM Demodulator block

The OFDM Demodulator block now has a parameter to enable a reset input port.

Variable-size FFT example with arbitrary input pattern

The HDL Implementation of a Variable-Size FFT example now includes a second model that shows how to handle input data that does not conform to the input valid pattern expected by the original model. The new model shows how to process data that is continuously valid and padded with zeros at the end of each symbol.

R2018a

Version: 1.1

New Features

Bug Fixes

TDD Support for MIB Recovery Reference Application: Recover master information block data from time-division duplex LTE signals

This reference application shows PSS and SSS signal detection, duplex mode detection, and MIB decoding for LTE networks that use TDD or FDD. The design supports HDL code generation with HDL Coder and is ready for deployment to hardware. See LTE Cell Search and MIB Recovery Using TDD Mode.

5G Filtered OFDM Modulation Example: Implement a filtered orthogonal frequency division multiplexing transmitter in hardware

This example implements an F-OFDM modulator suitable for use in 5G transmitter designs, and verifies the design using the 5G Library for LTE System Toolbox®. The example shows how to convert from double to fixed-point types and minimize the resource use of the design on an FPGA. See HDL Code Generation for Filtered OFDM (F-OFDM) Transmitter.

Gold Sequence Generator Block: Generate LTE-specific Gold sequences for channel estimation and descrambling

The Gold Sequence Generator block returns Gold sequences generated using the polynomial and shift length specified by LTE standard TS 36.212. LTE systems use a Gold sequence generator for reference symbols and for scrambling/descrambling data, such as in MIB and SIB coding and decoding.

OFDM Demodulator Block: Demodulate orthogonal frequency division multiplexing symbols according to the LTE standard

The OFDM Demodulator block implements an algorithm for demodulating LTE signals specified by LTE standard TS 36.212, providing an interface and architecture for HDL code generation and hardware deployment. The block demodulates time-domain OFDM samples and returns the resource grid that is used for cell ID detection, MIB recovery, SIB1 recovery, and further decoding.

Variable-Size FFT Example: Implement flexible-bandwidth FFT using a single FFT block

This example shows how to implement an FFT that can be used for different LTE standard bandwidths. The design uses a controller around an FFT HDL Optimized block. See HDL Implementation of a Variable-Size FFT.

R2017b

Version: 1.0

New Features

Standard-compliant LTE Simulink blocks

LTE HDL Toolbox blocks implement the CRC, Turbo, and Convolutional encoding and decoding algorithms specified by LTE standard TS 36.212, using hardware-friendly architectures and interfaces. For the list of algorithms provided in this product, see HDL-Optimized System Design.

Downlink detector and MIB recovery reference application

The product includes a reference application that performs PSS and SSS signal detection and MIB decoding. This design supports HDL code generation with HDL Coder and is ready for deployment to hardware. See LTE Cell Search and MIB Recovery.

HDL code generation support

The encoding and decoding blocks in LTE HDL Toolbox support HDL code generation. For the list of blocks, see HDL-Optimized System Design. To generate HDL code from these designs, you must have an HDL Coder license. HDL Coder also enables you to generate scripts and test benches for use with third-party HDL simulators.

Frame-to-sample and sample-to-frame conversions

Hardware designs must use a streaming data interface, while LTE Toolbox™ scripts operate on frame-based data. LTE HDL Toolbox provides frame-to-sample and sample-to-frame conversion capability that enables verification of hardware algorithms against LTE Toolbox designs. For details of the streaming interface, see Streaming Sample Interface.

Convert framed data in MATLAB to a sample stream using `ltehdlFramesToSamples`, then import the samples to Simulink for HDL-compatible design. For an example of this workflow, see Verify Turbo Decoder with Streaming Data from MATLAB.

Alternatively, import framed data from MATLAB to Simulink and convert the frames to a sample stream using the Frame to Samples block. For an example of this workflow, see Verify Turbo Decoder with Framed Data from MATLAB.

FPGA-in-the-loop verification on Xilinx or Intel boards

If you have an HDL Verifier™ license, you can use the FPGA-in-the-loop (FIL) feature to verify your HDL design against models in MATLAB and Simulink, while your design runs on an FPGA board. The FIL blocks provide efficiency improvements for streaming data across the interface between Simulink and the FPGA board. See FPGA-in-the-Loop.

Algorithm prototyping on Xilinx Zynq-based hardware

The Communications Toolbox™ Support Package for Xilinx Zynq-Based Radio enables you to design, prototype, and verify practical wireless communications systems on Xilinx Zynq-based radio hardware. LTE HDL Toolbox designs have compatible interfaces to fit into the SDR support package reference design. See Prototype LTE Algorithms on Hardware.