HDL Coder™ Release Notes

How to Contact MathWorks

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

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

508-647-7000 (Phone)

508-647-7001 (Fax)

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

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

HDL Coder™ Release Notes

Contents

Summary by Version

This table provides quick access to what's new in each version. For clarification, see "Using Release Notes" on page 1.

Version (Release)	New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
New Product V3.0 (R2012a)	Yes Details	Yes Summary	Bug Reports

Using Release Notes

Use release notes when upgrading to a newer version to learn about:

- New features
- Changes
- Potential impact on your existing files and practices

Review the release notes for other MathWorks® products required for this product (for example, MATLAB® or Simulink®). Determine if enhancements, bugs, or compatibility considerations in other products impact you.

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

What Is in the Release Notes

New Features and Changes

- New functionality
- Changes to existing functionality

Version Compatibility Considerations

When a new feature or change introduces a reported incompatibility between versions, the **Compatibility Considerations** subsection explains the impact.

Compatibility issues reported after the product release appear under Bug Reports at the MathWorks Web site. Bug fixes can sometimes result in incompatibilities, so review the fixed bugs in Bug Reports for any compatibility impact.

Fixed Bugs and Known Problems

MathWorks offers a user-searchable Bug Reports database so you can view Bug Reports. The development team updates this database at release time and as more information becomes available. Bug Reports include provisions for any known workarounds or file replacements. Information is available for bugs existing in or fixed in Release 14SP2 or later. Information is not available for all bugs in earlier releases.

Access Bug Reports using your MathWorks Account.

Documentation on the MathWorks Web Site

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

- Latest product documentation
- Archived documentation

Version 3.0 (R2012a) HDL Coder

This table summarizes what's new in V3.0 (R2012a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes—Details labeled as **Compatibility Considerations**, below. See also Summary.	Bug Reports

New features and changes introduced in this version are:

Product Name Change and Extended Capability

HDL Coder™ replaces Simulink HDL Coder and adds the HDL code generation capability directly from MATLAB.

To generate HDL code from MATLAB, you need the following products:

- HDL Coder
- MATLAB Coder™
- Fixed-Point Toolbox™
- MATLAB

To generate HDL code from Simulink, you need the following products:

- HDL Coder
- MATLAB Coder
- Fixed-Point Toolbox
- Simulink Fixed Point™

- Simulink
- MATLAB

Code Generation from MATLAB

You can now generate HDL code directly from MATLAB code.

This workflow provides:

- Verilog or VHDL code generation from MATLAB code.
- Test bench generation from MATLAB scripts.
- Automated conversion from floating point code to fixed point code.
- Automated HDL verification through integration with ModelSim® and ISim.
- HDL code generation for a subset of System objects from the Communications System Toolbox™ and DSP System Toolbox™.
- A traceability report mapping generated HDL code to your original MATLAB code.

The MATLAB to HDL workflow provides the following automated HDL code optimizations:

- Area optimizations: RAM mapping for persistent array variables, loop streaming, resource sharing, and constant multiplier optimization.
- Speed optimizations: input pipelining, output pipelining, and distributed pipelining.

The coder can also generate a resource utilization report, with RAM usage and the number of adders, multipliers, and muxes in your design.

See also “HDL Code Generation from MATLAB”.

Code Generation from Any Level of Subsystem Hierarchy

You can now generate HDL code from a subsystem at any level of the subsystem hierarchy. In previous releases, you could generate HDL code from the top-level subsystem only.

This feature also enables you to check any level subsystem for code generation compatibility, and to automatically generate a testbench.

Automated Subsystem Hierarchy Flattening

You can now generate code with a flattened subsystem hierarchy, while preserving hierarchy in nested subsystems.

This option enables you to perform more extensive area and speed optimization on the flattened component. It also enables you to reduce the number of HDL output files.

See also “Hierarchy Flattening”.

Support for Discrete Transfer Fcn Block

You can now generate HDL code from the Discrete Transfer Fcn block.

For details, see “Discrete Transfer Fcn Requirements and Restrictions”.

User Option to Constrain Registers on Output Ports

A new property, `ConstrainedOutputPipeline`, enables you to specify the number of registers you wish to have on an output port without introducing additional delay on the input to output path. The coder redistributes existing delays within your design to try to meet the constraint. This behavior is different from the `OutputPipeline` property, which introduces additional delay on the input to output path.

If the coder is unable to meet the constraint using existing delays, it reports the difference between the number of desired and actual output registers in the timing report.

Distributed Pipelining for Sum of Elements, Product of Elements, and MinMax Blocks

The Sum of Elements, Product of Elements, and MinMax blocks can now participate in distributed pipelining if their architecture is set to `Tree`.

MATLAB Function Block Enhancements

Multiple Accesses to RAMs Mapped from Persistent Variables

You can now perform multiple reads and writes to a persistent variable, and the persistent variable will still be mapped to RAM. In previous releases, a RAM mapped from a persistent variable could be accessed only once.

Streaming for MATLAB Loops and Vector Operations

You can now perform streaming on MATLAB loops and loops created from vector operations for improved area efficiency.

For details, see "Loop Optimization".

Loop Unrolling for MATLAB Loops and Vector Operations

You can now unroll user-written MATLAB loops and loops created from vector operations. This enables the coder to perform area and speed optimizations on the unrolled loops.

For details, see "Loop Optimization".

Automated Code Generation from Xilinx System Generator for DSP Blocks

You can now automatically generate HDL code from subsystems containing Xilinx® System Generator for DSP blocks.

For details, see "Code Generation with Xilinx System Generator Subsystems".

Altera Quartus II 11.0 Support in HDL Workflow Advisor

The HDL Workflow Advisor has now been tested with Altera® Quartus II 11.0. In previous releases, the HDL Workflow Advisor was tested with Altera Quartus II 9.1.

Automated Mapping to Xilinx and Altera Floating Point Libraries

The coder can now map Simulink floating point operations to synthesizable floating point Altera Megafunctions and Xilinx LogiCORE IP Floating Point Operator v5.0 blocks. To learn more, see "FPGA Target-Specific Floating-Point Library Mapping".

For a list of supported Altera Megafunction blocks, see "Supported Altera Floating-Point Library Blocks".

For a list of supported Xilinx LogicCORE IP blocks, see "Supported Xilinx Floating-Point Library Blocks".

Vector Data Type for PCI Interface Data Transfers Between xPC Target and FPGA

In the FPGA Turnkey workflow, you can now use vector data types with the **Scalarize Vector Ports** option to automatically generate PCI DMA transfers on the PCI interface between xPC Target and FPGA. You no longer need to manually insert multiplexers, demultiplexers and provide synchronization logic for vector data transfers.

If the **Scalarize Vector Ports** option is disabled when the code generation subsystem has vector ports, the coder displays an error.

New Global Property to Select RAM Architecture

There is a new global property, `RAMArchitecture`, that enables you to generate RAMs either with or without clock enables. This property applies to every RAM in your design, and replaces the block level property, `RAMStyle`. By default, RAMs are generated with clock enables.

To generate RAMs without clock enables, set RAMArchitecture to `'WithoutClockEnable'`. To generate RAMs with clock enables, either use the default, or set `RAMArchitecture` to `'WithClockEnable'`. For more information, see “Implement RAMs With or Without Clock Enable”.

Compatibility Consideration

The coder now ignores the block level property, `RAMStyle`.

If a block’s `RAMStyle` property is set, the coder generates a warning.

Turnkey Workflow for Altera Boards

HDL Workflow Advisor now supports Altera FPGA design software and the following Altera development kits and boards:

- Altera Arria II GX FPGA development kit
- Altera Cyclone III FPGA development kit
- Altera Cyclone IV GX FPGA development kit
- Altera DE2-115 development and education board

This workflow has been tested with Altera Quartus II 11.0.

HDL Support For Bus Creator and Bus Selector Blocks

Release R2012a provides HDL code generation for the Bus Creator and Bus Selector blocks. You must use these blocks for your buses if you want HDL support.

HDL Support For HDL CRC Generator Block

Release R2012a provides HDL code generation for the new HDL CRC Generator block.

HDL Support for Programmable Filter Coefficients

When using filter blocks to generate HDL code, can specify coefficients from input port(s).. This feature applies to FIR and BiQuad filter blocks only. Fully Parallel and all serial architectures are supported.

Follow these directions to use programmable filters:

1. Select `Input port(s)` as coefficient source from the filter block mask.
2. Connect the coefficient port with a vector signal.
3. Specify the implementation architecture and parameters from the HDL coder property interface.
4. Generate HDL code.

Notes

- For fully parallel implementations, the coefficients ports are connected to the dedicated MAC directly.
- For serial implementation, the coefficients ports first go to a mux, and then to the MAC. The mux decides the coefficients that used at current time instant
- For Discrete FIR filters, this feature is not supported under the following conditions:
  - Implementations having coefficients specified by dialog parameters (for example, complex input and coefficients with serial architecture)
  - Filters using DA architecture
  - CoeffMultipliers specified as `csd` or `factored-csd`
- For Biquad filters, this feature is not supported when CoeffMultipliers are specified as `csd` or `factored-csd`.

Synchronous Multiclock Code Generation for CIC Decimators and Interpolators

You can specify multiple clocks in one of the following ways:

- Use the model-level parameter `ClockInputs` with the function `makehdl` and specify the value as 'Multiple'.
- In the Clock settings section of the **Global Settings** pane in the HDL Code Generation Configuration Parameters dialog box, set **Clock inputs** to `Multiple`.

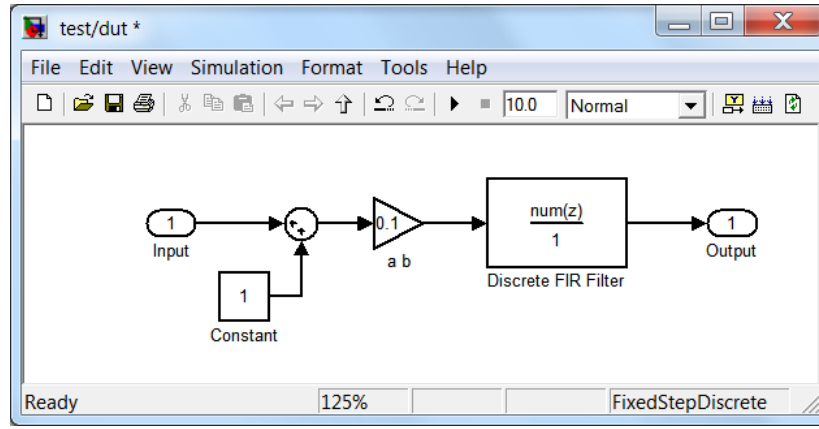
When you use single-clock mode, HDL code generated from multirate models uses a single master clock that corresponds to the base rate of the DUT. When you use multiple-clock mode, HDL code generated from multirate models use one clock input for each rate in the DUT. The number of timing controllers generated in multiple-clock mode depends on the design in the DUT.

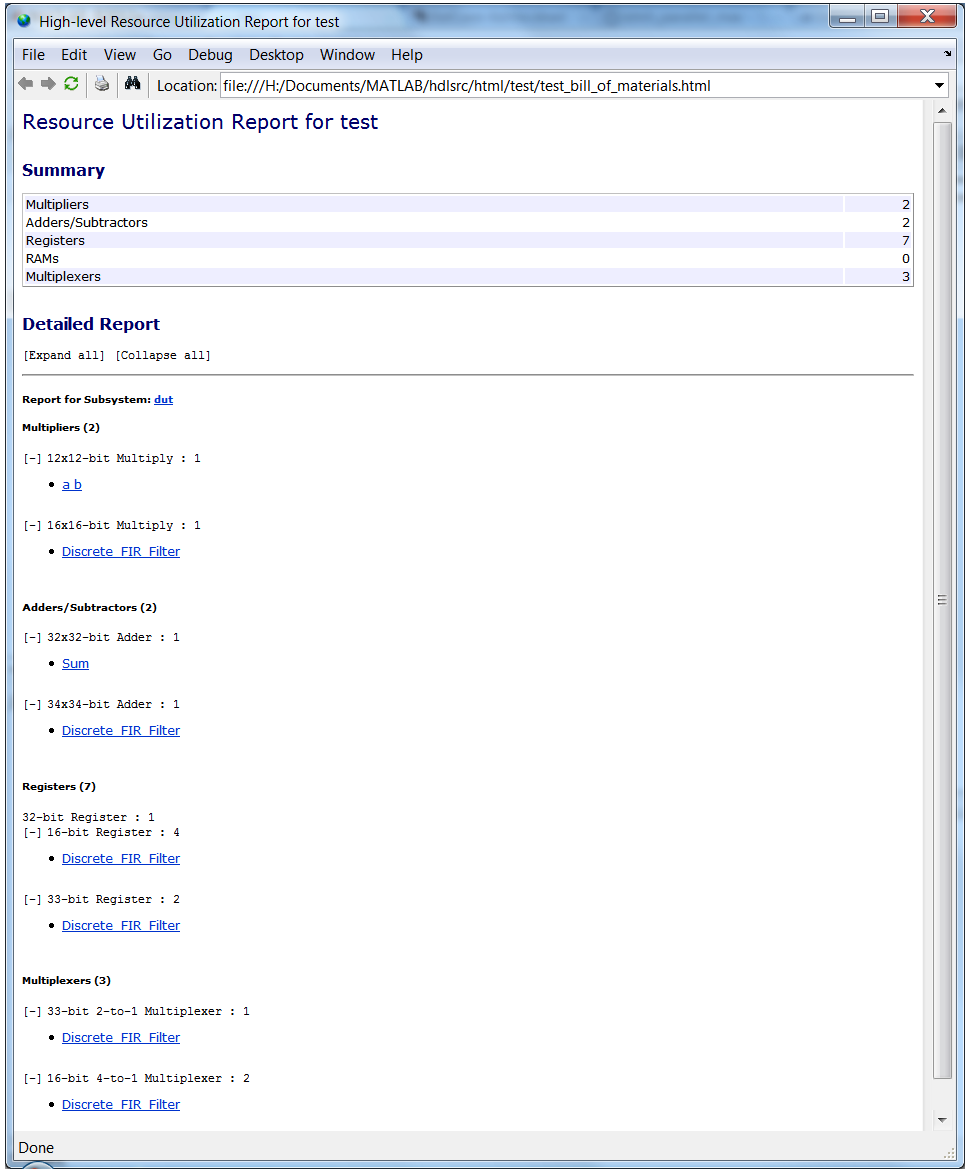
The `ClockInputs` parameter supports the values 'Single' and 'Multiple', where the default is 'Single'. In the default single-clock mode, the coder behavior is unchanged from previous releases.

Filter Block Resource Report Participation

Resource reports include the HDL resource usage for filter blocks. The report includes adders, subtractors, multipliers, multiplexers, registers. This feature covers all filter blocks, and all implementations for the block.

You can turn on the report feature using the command line (`ResourceReport`) or GUI (**Generate resource utilization report**). The following illustrations show a report for a model that includes a Discrete FIR Filter block.



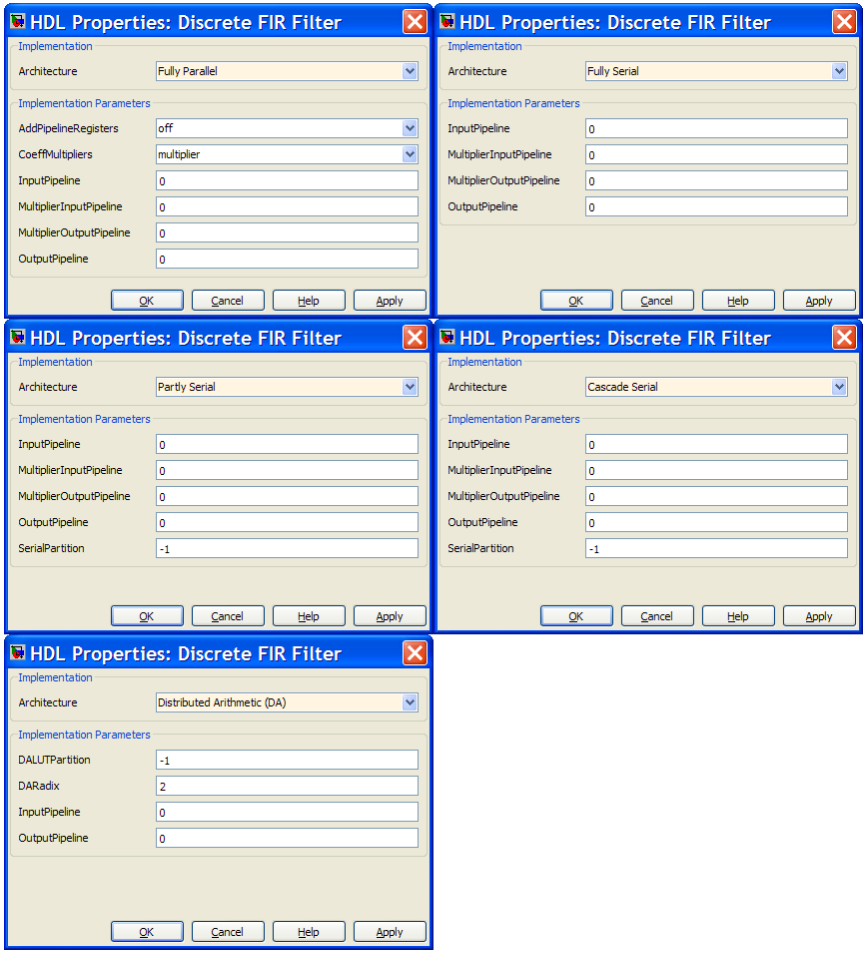

High-level Resource Utilization Report for test
File Edit View Go Debug Desktop Window Help
Location: file:///H:/Documents/MATLAB/hdlsrc/html/test/test_bill_of_materials.html
Resource Utilization Report for test
Summary
Multipliers 2
Adders/Subtractors 2
Registers 7
RAMs 0
Multiplexers 3
Detailed Report
[Expand all] [Collapse all]
Report for Subsystem: dut
Multipliers (2)
[-] 12x12-bit Multiply : 1
a b
[-] 16x16-bit Multiply : 1
Discrete_FIR_Filter
Adders/Subtractors (2)
[-] 32x32-bit Adder : 1
Sum
[-] 34x34-bit Adder : 1
Discrete_FIR_Filter
Registers (7)
32-bit Register : 1
[-] 16-bit Register : 4
Discrete_FIR_Filter
[-] 33-bit Register : 2
Discrete_FIR_Filter
Multiplexers (3)
[-] 33-bit 2-to-1 Multiplexer : 1
Discrete_FIR_Filter
[-] 16-bit 4-to-1 Multiplexer : 2
Discrete_FIR_Filter
Done

HDL Block Properties Interface Allows Choice of Filter Architecture

You can choose from several filter architectures for FIR Decimation and Discrete FIR Filter blocks. Choices are:

- Fully Parallel
- Distributed Architecture (DA)
- Fully Serial
- Party Serial
- Cascade Serial

The availability of architectures depends on the transfer function type and filter structure of filter blocks. For Partly Serial and DA, specify at least **SerialPartition** and **DALUTPartition**, respectively, so that these architectures are inferred. For example, if you select `Distributed Architecture (DA)`, make sure to also set **DALUTPartition**.



HDL Support for FIR Filters With Serial Architectures and Complex Inputs

HDL support for serial implementations of a FIR block with complex inputs.

HDL Support for External Reset Added for Proportional-Integral-Derivative (PID) and Discrete Time Integrator (DTI) Blocks

External reset support added for level mode.

Compatibility Summary for HDL Coder

This table summarizes new features and changes that might cause incompatibilities when you upgrade from an earlier version, or when you use files on multiple versions. Details are provided in the description of the new feature or change.

Version (Release)	New Features and Changes with Version Compatibility Impact
Latest Version **V3.0 (R2012a)**	See the **Compatibility Considerations** subheading for this new feature or change: • "New Global Property to Select RAM Architecture" on page 8