HDL Coder

Generate Verilog and VHDL code for FPGA and ASIC designs

HDL Coder™ generates portable, synthesizable Verilog® and VHDL® code from MATLAB® functions, Simulink® models, and Stateflow® charts. The generated HDL code can be used for FPGA programming or ASIC prototyping and design.

HDL Coder provides a workflow advisor that automates the programming of Xilinx® and Altera® FPGAs. You can control HDL architecture and implementation, highlight critical paths, and generate hardware resource utilization estimates. HDL Coder provides traceability between your Simulink model and the generated Verilog and VHDL code, enabling code verification for high-integrity applications adhering to DO-254 and other standards.

**Key Features**

- Target-independent, synthesizable VHDL and Verilog code
- Code generation support for MATLAB functions, System objects, and Simulink blocks
- Mealy and Moore finite-state machines and control logic implementations using Stateflow
- Workflow advisor for programming Xilinx and Altera application boards
- Resource sharing and retiming for area-speed tradeoffs
- Code-to-model and model-to-code traceability for DO-254
- Legacy code integration
Generating HDL code from MATLAB or Simulink with HDL Coder. You can generate synthesizable VHDL and Verilog code from MATLAB functions, Simulink models, or a combination of the two.

Generating HDL Code

HDL Coder lets you generate synthesizable HDL code for FPGA and ASIC implementations in a few steps:

- Model your design using a combination of MATLAB code, Simulink blocks, and Stateflow charts.
- Optimize models to meet area-speed design objectives.
- Generate HDL code using the integrated HDL Workflow Advisor for MATLAB and Simulink.
- Verify generated code using HDL Verifier™.

HDL Code Generation from MATLAB

The HDL Workflow Advisor in HDL Coder automatically converts MATLAB code from floating-point to fixed-point and generates synthesizable VHDL and Verilog code. This capability lets you model your algorithm at a high level using abstract MATLAB constructs and System objects while providing options for generating HDL code that is optimized for hardware implementation. HDL Coder provides a library of ready-to-use logic elements, such as counters and timers, which are written in MATLAB.

HDL Code Generation from Simulink

The HDL Workflow Advisor generates VHDL and Verilog code from Simulink and Stateflow. With Simulink, you can model your algorithm using a library of more than 200 blocks, including Stateflow charts. This library provides complex functions, such as the Viterbi decoder, FFT, CIC filters, and FIR filters, for modeling signal processing and communications systems and generating HDL code.
HDL Coder Workflow Advisor for Simulink. You can generate HDL code to program Xilinx and Altera FPGAs by connecting directly to Xilinx ISE and Altera Quartus II.

Optimizing HDL Code

In MATLAB or Simulink, you can optimize HDL code to achieve speed-area objectives by employing distributed pipelining, streaming, and resource sharing. In MATLAB, you can use advanced loop optimizations, such as loop streaming and loop unrolling, for a MATLAB design containing for-loops or matrix operations. You can map a persistent array or matrix variables in MATLAB code to block RAMs. In Simulink, you can implement multichannel designs and serialization techniques common to signal processing and multimedia applications.
HDL Workflow Advisor for MATLAB, which provides optimization options, such as RAM mapping, pipelining, resource sharing, and loop unrolling.

Area-speed optimization. Replacing four multipliers with one multiplier reduces the design area at the cost of increasing the data rate by a factor of four.

**Automating FPGA Design**

The HDL Workflow Advisor in HDL Coder automates the workflow for implementing your MATLAB algorithms and Simulink models into Xilinx and Altera FPGAs. The HDL Workflow Advisor integrates all steps of the FPGA design process, including:

- Checking the Simulink model for HDL code generation compatibility
- Generating HDL code, an HDL test bench, and a cosimulation model
- Performing synthesis and timing analysis through integration with Xilinx ISE and Altera Quartus II
- Estimating resources used in the design
Back annotating the Simulink model with critical path timing

Back annotating a Simulink model with critical path timing. The HDL Workflow Advisor highlights critical path timing in Simulink to help identify speed bottlenecks and improve design performance.

You can view a postsynthesis timing report and back annotate the Simulink model to identify timing-constraint bottlenecks. This integration with synthesis tools enables rapid design iterations and significantly reduces FPGA design cycle time.

**Verifying HDL Code**

HDL Coder generates VHDL and Verilog test benches for rapid verification of generated HDL code. You can customize an HDL test bench using a variety of options that apply stimuli to the HDL code. You can also generate script files to automate the process of compiling and simulating your code in HDL simulators.

HDL Coder works with HDL Verifier to automatically generate two types of cosimulation models:

- HDL cosimulation model, for performing HDL cosimulation with Simulink and an HDL simulator, such as Cadence Incisive or Mentor Graphics ModelSim and Questa
- FPGA-in-the-loop (FIL) cosimulation model, for verifying your design with Simulink and an FPGA board
Automatically generated FPGA-in-the-loop (FIL) model for video sharpening. FIL simulation lets you efficiently perform design space exploration on your hardware.

**Documenting and Tracing HDL Code**

HDL Coder documents generated code in an HTML report that contains hyperlinked HDL code and a table of generated HDL files. Hyperlinks in the HDL code let you navigate to the corresponding MATLAB algorithm or Simulink blocks that generated the code.

HDL Coder supports code traceability for high-integrity applications that adhere to standards such as DO-254 by enabling you to:

- Navigate to MATLAB code from generated HDL code
- Navigate between Simulink blocks and generated HDL code for bidirectional tracing
- Insert user-controlled comments and descriptions to improve code readability
Code Generation Report in MATLAB, which lets you navigate to MATLAB code from generated VHDL and Verilog code.

Using Simulink Verification and Validation™ with HDL Coder enables you to embed system requirements as comments within HDL code generated from Simulink or Stateflow. As a result, you can achieve complete transparency throughout the entire workflow, from system requirements to generated HDL code.

HDL Code Generation Report in Simulink, which lets you navigate between model and generated HDL code.

**HDL Coding Standards**

The development processes used for industrial FPGA and ASIC applications, such as DO-254 in the aerospace industry, may recommend the use of certain RTL coding guidelines. HDL Coder seeks to generate VHDL and Verilog code that meets common industry coding guidelines such as RMM and STARC. HDL Coder also
generates reports that help you identify unsuitable constructs in your Simulink models and MATLAB code so that you can adapt your models to get generated RTL that satisfies these coding guidelines.

HDL Coder can also generate third-party lint tool scripts that can be used to check your generated HDL code. Code generated with HDL Coder can be checked with industry-standard lint tools like Atrenta SpyGlass, Real Intent Ascent Lint, and Synopsys Leda. HDL Coder generates custom scripts that allow integration with any lint tool.

Code generated using HDL Coder follows RTL coding principles by:

- Avoiding FSM state reachability and coding issues
- Avoiding differences between simulation and synthesis semantics
- Avoiding operations with expensive implementation costs
- Avoiding downstream tool flow issues
- Following naming and RTL coding conventions in the generated code
- Enforcing RTL modeling clarity and reducing complexity
- Enforcing checks for clock bundles (clocks, enables, resets) and control signals
- Enabling testability and traceability of the code

### Resources

<table>
<thead>
<tr>
<th>Product Details, Examples, and System Requirements</th>
<th>Online User Community</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trial Software</td>
<td>Training Services</td>
</tr>
<tr>
<td><a href="http://www.mathworks.com/trialrequest">www.mathworks.com/trialrequest</a></td>
<td><a href="http://www.mathworks.com/training">www.mathworks.com/training</a></td>
</tr>
<tr>
<td>Sales</td>
<td>Third-Party Products and Services</td>
</tr>
<tr>
<td>Technical Support</td>
<td>Worldwide Contacts</td>
</tr>
<tr>
<td><a href="http://www.mathworks.com/support">www.mathworks.com/support</a></td>
<td><a href="http://www.mathworks.com/contact">www.mathworks.com/contact</a></td>
</tr>
</tbody>
</table>