# LogiCORE Aurora v2.9

## *Getting Started Guide*

**UG173 (v2.9) March 24, 2008**

XILINX®

# Revision History

The following table shows the revision history for this document.

| Date | Version | Revision |
|---|---|---|
| 04/28/05 | 1.1 | Initial Xilinx release. |
| 01/10/06 | 2.0 | LogiCORE Aurora v2.4 release. |
| 09/12/06 | 2.5 | LogiCORE Aurora v2.5 release. |
| 11/30/06 | 2.5.1 | LogiCORE Aurora v2.5.1 release.<br>Updated Chapter 2, "Installing and Licensing the Core." |
| 03/01/07 | 2.6 | LogiCORE Aurora v2.6 release. |
| 05/17/07 | 2.7 | LogiCORE Aurora v2.7 release. Updated "Generating the Core." Added "Using ChipScope Pro Cores with the Aurora Core." |
| 08/22/07 | 2.7.1 | LogiCORE Aurora v2.7.1 release. |
| 10/10/07 | 2.8 | LogiCORE Aurora v2.8 release. Added "Using ChipScope Pro Cores with the Aurora Core" and "Using the Timer-Based Simplex Mode." |
| 03/24/08 | 2.9 | LogiCORE Aurora v2.9 release. Added ISIM support. |

# Table of Contents

# About This Guide

The *LogiCORE Aurora v2.9 Getting Started Guide* provides information about generating a LogiCORE™ Aurora core using Virtex™-II Pro or Virtex-4 RocketIO™ transceivers. The information includes customizing and simulating the core using the provided example design, and running the design files through implementation using the Xilinx tools.

## Contents

This guide contains the following chapters:

- Preface, "About this Guide" introduces the organization and purpose of this guide, a list of additional resources, and the conventions used in this document.
- Chapter 1, "Introduction" describes the core and related information, including recommended design experience, additional resources, technical support, and submitting feedback to Xilinx.
- Chapter 2, "Installing and Licensing the Core" provides information about installing and licensing the core.
- Chapter 3, "Quick Start Example Design" provides an overview of the Aurora protocol and core, and gives a step-by-step tutorial on how to generate Aurora designs with the CORE Generator™ tool.

## Additional Resources

For additional information, go to http://www.xilinx.com/support. The following table lists some of the resources you can access from this website or by using the provided URLs.

| Resource | Description/URL |
|---|---|
| Tutorials | Tutorials covering Xilinx design flows, from design entry to verification and debugging<br>http://www.xilinx.com/support/techsup/tutorials/index.htm |
| Answer Browser | Database of Xilinx solution records<br>http://www.xilinx.com/xlnx/xil_ans_browser.jsp |
| Application Notes | Descriptions of device-specific design techniques and approaches<br>http://www.xilinx.com/xlnx/xweb/xil_publications_index.jsp?category=Application+Notes |
| Data Sheets | Device-specific information on Xilinx device characteristics, including readback, boundary scan, configuration, length count, and debugging<br>http://www.xilinx.com/xlnx/xweb/xil_publications_index.jsp |

| Resource | Description/URL |
|---|---|
| Problem Solvers | Interactive tools that allow you to troubleshoot your design issues<br>http://www.xilinx.com/support/troubleshoot/psolvers.htm |
| Tech Tips | Latest news, design tips, and patch information for the Xilinx design environment<br>http://www.xilinx.com/xlnx/xil_tt_home.jsp |

# Conventions

This document uses the following conventions. An example illustrates each convention.

## Typographical

The following typographical conventions are used in this document:

| Convention | Meaning or Use | Example |
|---|---|---|
| `Courier font` | Messages, prompts, and program files that the system displays | `speed grade: - 100` |
| **`Courier bold`** | Literal commands that you enter in a syntactical statement | **`ngdbuild`** *design_name* |
| **Helvetica bold** | Commands that you select from a menu | **File → Open** |
| | Keyboard shortcuts | **Ctrl+C** |
| Italic font | Variables in a syntax statement for which you must supply values | **`ngdbuild`** *design_name* |
| | References to other manuals | See the *Development System Reference Guide* for more information. |
| | Emphasis in text | If a wire is drawn so that it overlaps the pin of a symbol, the two nets are *not* connected. |
| Square brackets [ ] | An optional entry or parameter. However, in bus specifications, such as **`bus[7:0]`**, they are required. | **`ngdbuild`** [*`option_name`*] *`design_name`* |
| Braces { } | A list of items from which you must choose one or more | **`lowpwr =`**{**`on`**\|**`off`**} |
| Vertical bar \| | Separates items in a list of choices | **`lowpwr =`**{**`on`**\|**`off`**} |
| Vertical ellipsis<br>.<br>.<br>. | Repetitive material that has been omitted | `IOB #1: Name = QOUT'`<br>`IOB #2: Name = CLKIN'`<br>.<br>.<br>. |
| Horizontal ellipsis ... | Repetitive material that has been omitted | **`allow block`** *block_name loc1 loc2 ... locn;* |

## Online Document

The following conventions are used in this document:

| Convention | Meaning or Use | Example |
|---|---|---|
| Blue text | Cross-reference link to a location in the current document | See the section "Additional Resources" for details.<br>Refer to "Title Formats" in Chapter 1 for details. |
| Red text | Cross-reference link to a location in another document | See Figure 2-5 in the *Virtex-II Platform FPGA User Guide.* |
| Blue, underlined text | Hyperlink to a website (URL) | Go to http://www.xilinx.com for the latest speed files. |

XILINX®

Chapter 1

# Introduction

This chapter introduces the Aurora core and provides related information, including recommended design experience, additional resources, technical support, and how to submit feedback to Xilinx.

The LogiCORE™ Aurora core is a high-speed serial solution based on the Aurora protocol and the Virtex™-II Pro and Virtex-4 RocketIO™ Multi-Gigabit Transceiver (MGT). The core is delivered as open-source code and supports both Verilog and VHDL design environments. Each core comes with an example design and supporting modules.

## About the Core

The Aurora core is a CORE Generator™ IP core, included in the latest IP Update on the Xilinx IP Center. For detailed information about the core, see http://www.xilinx.com/aurora. For information about system requirements, installation, and licensing options, see Chapter 2, "Installing and Licensing the Core."

## Recommended Design Experience

Although the Aurora core is a fully verified solution, the challenge associated with implementing a complete design varies depending on the configuration and functionality of the application. For best results, previous experience building high-performance, pipelined FPGA designs using Xilinx implementation software and user constraints files (UCF) is recommended.

Contact your local Xilinx representative for a closer review and estimation for your specific requirements.

## Related Xilinx Documents

Prior to generating an Aurora core, users should be familiar with the following:

- Documents located on the Aurora product page: http://www.xilinx.com/aurora
  - SP002: *Aurora Protocol Specification*
  - UG058: *Aurora Bus Functional Model User Guide*
- Documents located on the LocalLink product page: http://www.xilinx.com/locallink
  - SP006: *LocalLink Interface Specification*
- Xilinx RocketIO Transceiver User Guides:
  - UG024: *RocketIO Transceiver User Guide* (for Virtex™-II Pro transceivers)
  - UG076: *Virtex-4 RocketIO Multi-Gigabit Transceiver User Guide*
- ISE™ documentation
  http://www.xilinx.com/support/sw_manuals/xilinx9/index.htm

# Additional Core Resources

For detailed information and updates about the Aurora core, see the following documents, located on the Aurora product page at http://www.xilinx.com/aurora.

- DS128: *(LogiCORE) Aurora v2.9 Data Sheet*
- UG061: *LogiCORE Aurora v2.9 User Guide*
- UG173: *LogiCORE Aurora v2.9 Getting Started Guide*
- Aurora v2.9 Release Notes

# Technical Support

For technical support, go to www.xilinx.com/support. Questions are routed to a team of engineers with expertise using the Aurora core.

Xilinx will provide technical support for use of this product as described in the *LogiCORE Aurora v2.9 User Guide*. Xilinx cannot guarantee timing, functionality, or support of this product for designs that do not follow these guidelines.

# Feedback

Xilinx welcomes comments and suggestions about the Aurora core and the accompanying documentation.

## Core

For comments or suggestions about the Aurora core, please submit a WebCase from www.xilinx.com/support. Be sure to include the following information:

- Product name
- Core version number
- List of parameter settings
- Explanation of your comments

## Document

For comments or suggestions about this document, please submit a WebCase from www.xilinx.com/support. Be sure to include the following information:

- Document title
- Document number
- Page number(s) to which your comments refer
- Explanation of your comments

Chapter 2

# Installing and Licensing the Core

This chapter provides instructions for installing the Aurora core in the CORE Generator tool and obtaining a free license to use the core.

## Before You Begin

Before installing the Wizard, you must have a MySupport account and the ISE 10.1 software installed on your system. If you have already completed these steps, go to "Installing the Core," page 14, otherwise, do the following:

1. Click **Login** at the top of the Xilinx home page, then follow the onscreen instructions to create a MySupport account.
2. Install the ISE 10.1 software and the applicable Service Pack. ISE Service Packs can be downloaded from www.xilinx.com/support/download.htm.

## System Requirements

**Windows**

- Windows XP Professional SP1, SP2, 32-bit, 64-bit
- Windows Vista Business 32-bit

**Solaris**

- Sun Solaris 9, 10

**Linux**

- Red Hat Enterprise WS 3.0/4.0/5.0 (32-bit or 64-bit)

**Software**

- ISE 10.1 with applicable Service Pack

Check the release notes for the required Service Pack; ISE Service Packs can be downloaded from http://www.xilinx.com/support/sw_manuals/xilinx9/index.htm.

# Installing the Core

You can install the Wizard in two ways: using the CORE Generator WebUpdate facility, which lets you select from a list of updates, or by performing a manual installation after downloading the core from the web.

## Automated Installation Using WebUpdate

***Note:*** To use this installation method behind a firewall, you must know your proxy settings. If necessary, contact your administrator to determine the proxy host address and port number before you begin.

1. From the main CORE Generator window, choose **Tools** → **Software Update...**
2. Click **OK** to close the CORE Generator tool and start WebUpdate.
3. If necessary, click **Advanced...** to specify a proxy host.
4. Click **Check for Updates**.
5. Ensure **ISE 10.1 IP Update 0** is selected in the list of software updates.
6. Click **OK**.
7. WebUpdate downloads and installs the selected software updates. Restart your computer when the install is finished.
8. To confirm the installation, from the main CORE Generator window, choose **Help** → **About Xilinx CORE Generator**.
9. Look for the following lines in the About dialog box:

```
Updates installed:

ISE 10.1 IP Update 0
```

## Manual Installation

1. Close the CORE Generator application if it is running.
2. Download the ZIP file from the following location and save it to a temporary directory: http://www.xilinx.com/support/download/index.htm.

   ***Note:*** Before you can access this page and the files listed on it, you must be registered for CORE Generator IP Updates access.
3. Extract the ZIP archive file `ise_101_ip_update0.zip` to a temporary location.

   **For Windows**

   - Extract the ZIP archive file using WinZip 7.0 SR-1 or later.

     ***Note:*** When extracting the files using WinZip, you must check the **Use Folder Names** option.

   **For UNIX**

   - Extract the ZIP archive file using unzip.

     ***Note:*** You might need system administrator privileges to install the update.
4. In the root level of the extracted directory structure, run the setup (`.exe`) executable to install the update.
5. Restart the CORE Generator tool; it automatically detects and displays the newly installed IP cores.
6. Determine whether the installation was successful by verifying that the new cores are visible in the main CORE Generator window.

# Obtaining Your License

To obtain your license for the Aurora core, perform the following steps:

- Navigate to the Aurora product page: http://www.xilinx.com/aurora
- Click the **Aurora LogiCORE** link at the bottom of the page
- Click **Order and Register**

Follow the onscreen instructions to review and electronically sign the Aurora License Agreement and download your license file for the Aurora core.

# Installing Your License File

After selecting a license option, an email will be sent to you that includes instructions for installing your license file. In addition, information about advanced licensing options and technical support is provided.

Chapter 3

# Quick Start Example Design

The quick start instructions are a step-by-step procedure for generating an Aurora core, implementing the core in hardware using the accompanying example design (aurora_example), and simulating the core with the provided demonstration testbench (example_tb). To learn more about the example design provided with the Aurora core, see the *LogiCORE Aurora v2.9 User Guide*.

## Overview

The quick start example consists of the following components:

- An instance of the Aurora core generated using the default parameters
  - Full-duplex with a single RocketIO™ MGT
  - Both flow control options
  - LocalLink interface
  - Virtex-II Pro target device
- A top-level example design with user constraints file (UCF) for an ML321 board
- A demonstration testbench (example_tb) to simulate two instances of the example design

The Aurora example design has been tested with Synplicity and XST for synthesis and ModelSim for simulation.

Figure 3-1 shows a block diagram of the default Aurora example design.

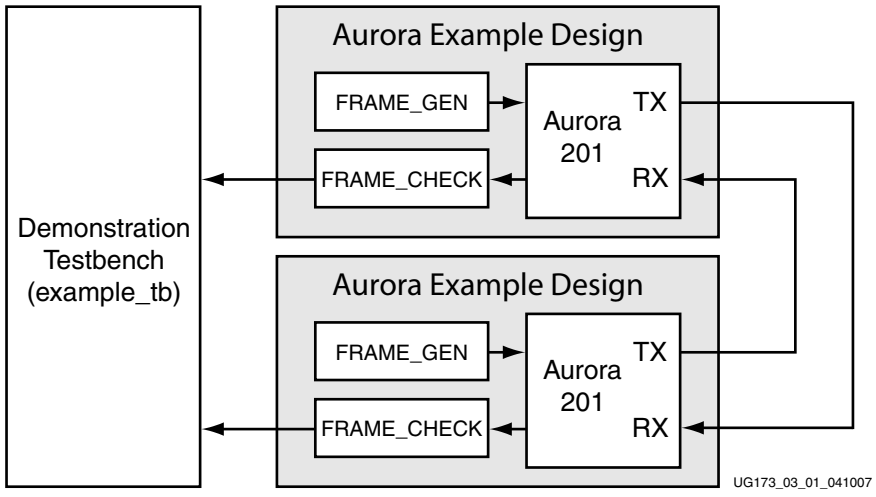



*Figure 3-1:* **Example Design**

# Generating the Core

To generate an Aurora core with default values using the CORE Generator tool:

1. Start the CORE Generator tool.

   For help starting and using the CORE Generator tool, see *CORE Generator Help* in the ISE documentation.

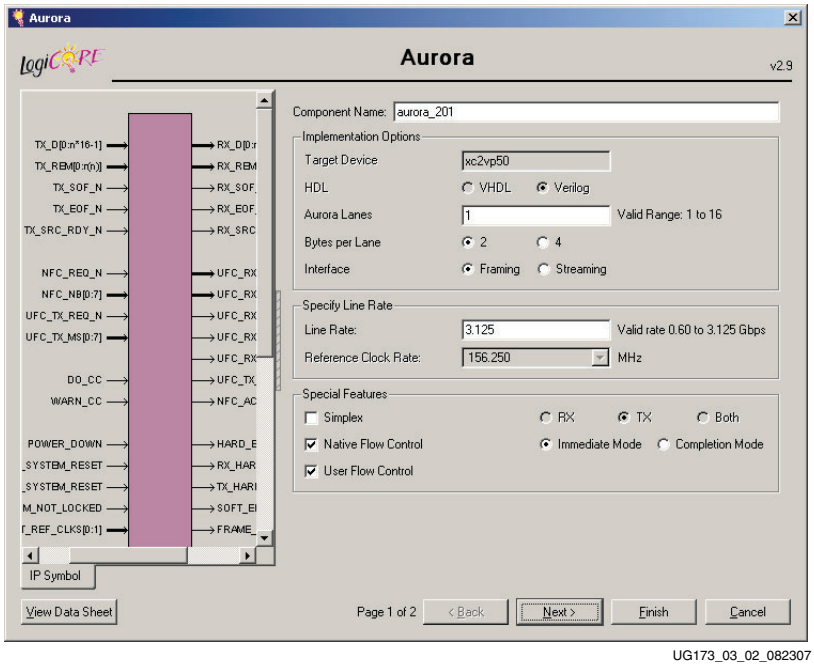
2. Choose **File** → **New Project**.
3. Type a location and a directory name. This example uses the following location and directory name:

   `/Projects/aurora/201_v2_9`

4. Click **OK** to create the directory.
5. To set project options:
   - On the **Part** tab, select a **Family** and **Device** that support the Aurora core, such as Virtex2P and xc2vp7.

     ***Note:*** If an unsupported silicon family is selected, the Aurora core appears light grey in the taxonomy tree and cannot be customized. For a list of supported architectures, see the *LogiCORE Aurora v2.9 User Guide*.

   - No further project options need to be set.
   - Optionally, on the **Generation** tab, set the **Design Entry** pulldown to **Verilog**.
6. After creating the project, locate the Aurora core in the taxonomy tree under `/Communication_&_Networking/Serial_Interfaces`.
7. Double-click the **core**. If the license file is not properly configured, the CORE Generator tool reports an error. See Chapter 2, "Installing and Licensing the Core."




UG173_03_02_082307

*Figure 3-2:* **CORE Generator Aurora Customization Screen**

8. In the Component Name field, enter a name for the core instance. This example uses the `name` **`aurora_201`**.

9. Click **Finish**.

   The core and its supporting files, including the example design, are generated in the project directory. For detailed information about the example design files and directories, see the *LogiCORE Aurora v2.9 User Guide*.

# Implementing the Example Design

After the core is generated, the design can be processed by the Xilinx implementation tools. The generated output files include several scripts to assist the user in running the Xilinx software.

From the command prompt, navigate to the project directory and type the following:

For Windows

```
ms-dos> cd aurora_201\scripts
ms-dos> xilperl make_aurora.pl -example -m -p -b -win
```

For UNIX

```
unix-shell% cd aurora_201/scripts
unix-shell% xilperl make_aurora.pl -example -m -p -b
```

These commands execute a script that synthesizes, builds, maps, place-and-routes the example design and produces a bitmap file. The resulting files are placed in the scripts directory. See the *LogiCORE Aurora v2.9 User Guide* for information on how to use the make_aurora.pl build script to create an ISE project for the Aurora core.

# Using ChipScope Pro Cores with the Aurora Core

## Description

The ChipScope™ Pro ICON and ILA cores aid in debugging and validating the design in board. To assist with debugging, these cores can be added to the Aurora core from CORE Generator™ tool.

## Usage

Add *_cscope* at the end of the module name while generating the Aurora core.

For example: aurora_201_cscope

Coregen adds the ICON and ILA cores to the Aurora example design and are found in the example top-level module. The user must generate the respective `.edn` files using the current ChipScope Pro tool.

# Using the Timer-Based Simplex Mode

## Description

In simplex mode, the Aurora core RX channel partner communicates to the TX channel partner via sideband signals named RX_RESET, RX_ALIGNED, RX_VERIFY, and RX_BONDED. The Aurora core has been enhanced to include a timer-based simplex mode. This simplex mode does not rely on sideband signals, thereby saving trace routing at the board level, resulting in cost savings. In the timer-based simplex mode, the sideband signals are generated by a set of timers whose values were taken from sample Aurora design simulations. These timer values can be modified according to the Aurora core configuration and user requirements.

### Usage

Add *_simplextimer* at the end of the module name while generating the Aurora core.

For example: aurora_201_simplextimer

The CORE Generator tool generates the Timer-based Simplex Aurora core.

## Using the ISE Flow to Generate the Aurora Core

1. Invoke ISE 10.1.
2. Select **File** → **New Project...** to invoke the New Project Wizard.
3. Enter the Project Name and Project Location.
4. Set the Top-Level Source Type to **HDL**.
5. Click **Next**.
6. Select the Device and specify the Preferred Language.
7. Click **Next**.
8. Click **New Source** to invoke the New Source Wizard.
9. Select **IP (Coregen & Architecture Wizard)**.
10. Enter a File name and click **Next**.
11. Select **Communication & Networking** → **Serial Interfaces** → **Aurora v2.9** and click **Next**.
12. Click **Finish**.
13. If necessary, click **Yes** to create the new directory.
14. Click **Next** twice to reach the Project Summary.
15. Click **Finish** to invoke CORE Generator and the LogiCORE Aurora v2.9 GUI.
16. Provide the necessary parameters for the Aurora configuration (refer to the *LogiCORE Aurora v2.9 User Guide*) and click **Finish**.
17. Note that CORE Generator has added an XCO file to the source file tree for the project.
18. Add the top level source files from the `examples` and `src` directories.
19. Add the constraints file from the `ucf` directory.
20. Perform synthesis and implementation steps to generate a `.bit` file.
21. Use IMPACT to transfer the `.bit` file to the target device.

When using the ISE flow to generate and synthesize the Aurora CORE, it is important to set the Hierarchy Separator value to '/' and the Bus Delimiter value to '[ ]'.

To set these values:

1. Right-click **Synthesize - XST** in the Process tree and select **Properties**.
2. Select the **Synthesis Options** category.
3. Set the Property display level to **Advanced**.
4. Select '**/**' in the Hierarchy Separator list.
5. Select '**[ ]**' in the Bus Delimiter list.
6. Click **OK**.

# Simulating the Example Design

The Aurora core provides a quick way to simulate and observe the behavior of the core using the provided example design. Prior to simulating the core, the functional (gate-level) simulation models must be generated. You must compile all source files in the following directories to a single library as shown in Table 3-1. Refer to *Simulating Your Design* in the *Synthesis and Verification Design Guide* for ISE 10.1 for instructions to compile simulation libraries for ISE software.

*Table 3-1:* **Required Simulation Libraries**

| HDL | Library | Source Directories |
|---|---|---|
| Verilog | UNISIMS_VER | <*Xilinx dir*>/verilog/src/unisims<br><*Xilinx dir*>/smartmodel/<*OS*>/wrappers/mtiverilog |
| VHDL | UNISIM | <*Xilinx dir*>/vhdl/src/unisims<br><*Xilinx dir*>/smartmodel/<*OS*>/wrappers/mtivhdl |

**Notes:**

OS refers to the following operating systems: nt, lin, lin64 or sol

The Aurora core provides a command line script to simulate the example design. To run a VHDL or Verilog ModelSim simulation of the Aurora core, use the following instructions:

1. Launch the ModelSim simulator and set the current directory to:

   ***<project directory>/aurora_201/scripts***

2. Set the MTI_LIBS variable:

   `modelsim>` **`setenv MTI_LIBS <path to compiled libraries>`**

3. Launch the simulation script:

   `modelsim>` **`do example_test.do`**

The ModelSim script compiles the example design and testbench, and adds the relevant signals to the wave window. After the design is compiled and the wave window is displayed, run the simulation for about 20 µs to see the Aurora core power up, followed by Aurora channel initialization and data transfer. Data transfer begins after the CHANNEL_UP signal goes High.

***Note:*** Run the simulation for about 180 µs for Virtex-4 cores.

# Simulating the Example Design using ISIM simulator

The Aurora core provides a command line script to simulate the example design using the ISIM simulator. To run a VHDL or Verilog ISIM simulation of the Aurora core, use the following instructions:

1. Generate a core and set the current directory to:

   ***<project directory>/scripts***

2. Launch the simulation script:

   **`perl simulate_isim.pl`**

The Perl script compiles the example design and testbench, and adds the relevant signals to the wave window. After the design is compiled, the wave window is displayed and the simulation is ran for about 20 µs to see the Aurora core power up, followed by Aurora channel initialization and data transfer. Data transfer begins after the CHANNEL_UP signal goes High.

# Testing DUT-BFM Automatic Compliance of Example Design

For DUT-BFM automatic compliance testing, obtain and place the Aurora bus functional model shared library (abfm.sl) in the `scripts` directory. Refer to the *Aurora Bus Functional Model User Guide* for more information about obtaining the Aurora bus functional model 8B/10B shared library.

From the command prompt, navigate to the project directory and type the following:

**For Windows**

```
ms-dos> cd aurora_201\scripts
ms-dos> xilperl run_aurora_sim.pl -a abfm.sl
```

**For UNIX**

```
unix-shell% cd aurora_201/scripts
unix-shell% xilperl run_aurora_sim.pl -a abfm.sl
```

Automatic compliance has three flavors:

- Lane Init compliance checks compliance for all the activities in Lane Init, Channel Bonding (for multi-lane designs) and Channel Verification state machines.
- Channel-up compliance checks compliance of all UPDU, UFC and NFC packet transmissions.
- Soft Error compliance checks compliance of error handling mechanism of the Aurora Protocol.

At the end of the test, BFM displays log message whether DUT passed or failed any compliance criteria. Refer to the *Aurora Bus Functional Model User Guide* for detailed information about the automatic compliance state machines and error messages given in each state in case compliance fails in that state.

The Aurora v2.9 single-lane design is tested and is compliant with Channel-up and Soft Error compliance.

# Example Design Hierarchy

The hierarchy for the design used in this quick start example is as follows:

```
example_tb
|___aurora_example
    |___aurora_201
    |   |___aurora_lane
    |   |   |___lane_init_sm
    |   |   |___chbond_count_dec
    |   |   |___sym_gen
    |   |   |___sym_dec
    |   |   |___error_detect
    |   |
    |   |___phase_align
    |   |___global_logic
    |   |   |___channel_init_sm
    |   |   |___idle_and_ver_gen
    |   |   |___channel_error_detect
    |   |
    |   |___tx_ll
    |   |   |___tx_ll_datapath
    |   |   |___tx_ll_control
    |   |
    |   |___rx_ll
    |       |___rx_ll_nfc
    |       |___ufc_filter
    |       |___rx_ll_pdu_datapath
    |       |___rx_ll_ufc_datapath
    |
    |___standard_cc_module
    |___frame_gen
    |___frame_check
```

***Note:*** The mgt_wrapper is present in the same hierarchy as aurora_lane, phase_align, and global_logic, if a Virtex-4 device is selected.