PHY IP Design Flow with Interlaken for Stratix V Devices

AN-634-1.0 Application Note

This application note describes implementing and simulating the protocol-specific PHY intellectual property (IP) core in Stratix® V devices using the Interlaken PHY IP interface. You can use the reference design file described in this application note to evaluate the design flow of the PHY IP interface and the Stratix V device family.

The PHY IP interface, starting with the Quartus® software version 10.1, automatically configures the setting parameters in Stratix V devices for the physical coding sublayer (PCS) module, leaving a small number of parameters in the physical media attachment (PMA) module for your control.

The PHY IP design is modular and uses standard interfaces. All PHY IP modules include an Avalon® Memory-Mapped Management Interface to access control and status registers and an Avalon Streaming interface to connect to the MAC block for data transfer. Most of the PCS and PMA functions are implemented as hard logic to save FPGA resources. The PCS and PMA blocks cannot be separated or bypassed. The key interfaces are serial data from external interfaces, parallel data for internal PLD interface, and an internal control interface.

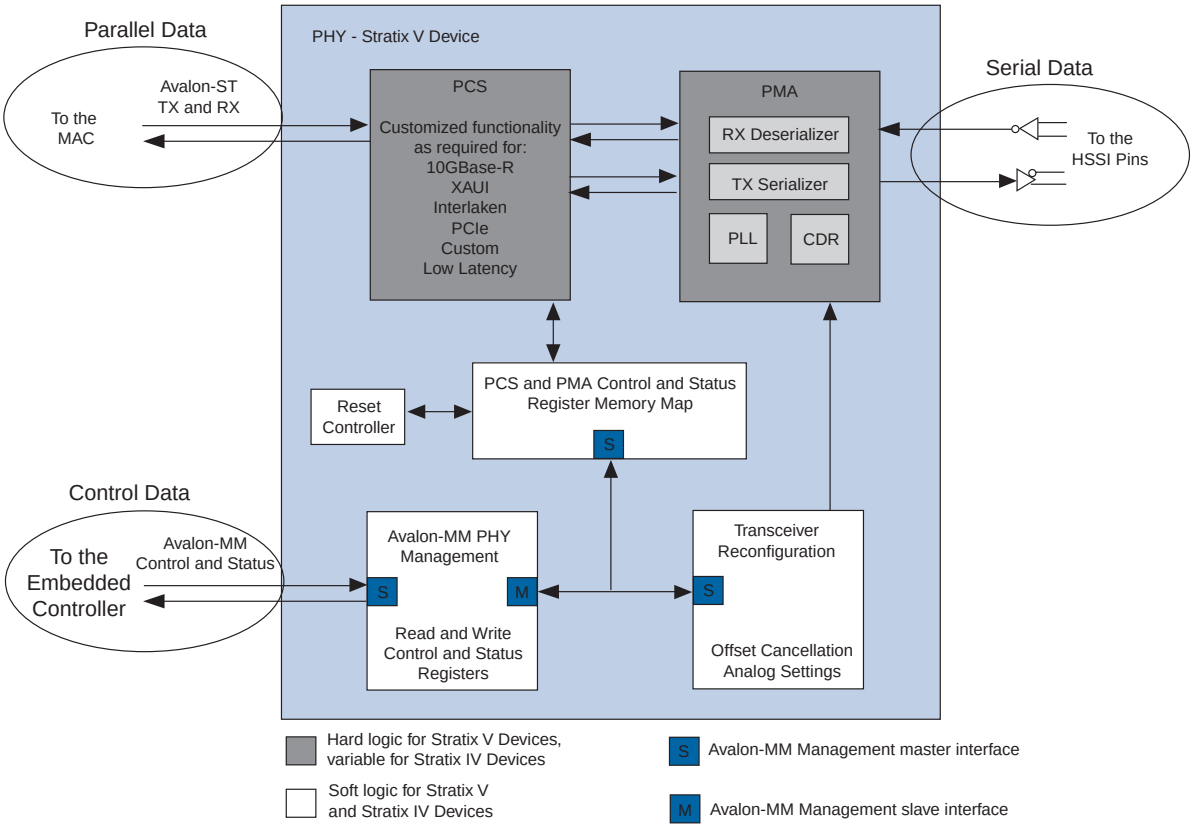
The PHY IP interfaces include the following:

- Avalon-ST TX/RX Interface—The PLD parallel interface for the transmit (TX) and receive (RX) data stream from the data packet of the TX and RX path of the PCS.
- Avalon-ST Ready Interface—Indication signals that the RX and TX paths are active and ready for data reception or transmission.
- Serial RX and TX Interface—The transceiver's input and output serial data.
- Avalon Memory-Mapped (Avalon-MM) Management Interface—Indirect addressing to access the PHY IP's internal registers and settings. The PHY IP expects `phy_mgmt_address[8:0]` as a word address, but because the byte address is the standard interface to a master, the Avalon master takes the byte address (`mgmt_address[9:0]`) and converts it to a word address for the PHY IP.
- Clock Interface—The clock interface includes the user clock output for the RX and TX. Other inputs are the phase-locked loop (PLL), TX and RX clock, and PHY IP management clock.

101 Innovation Drive
San Jose, CA 95134
www.altera.com

Figure 1 shows the typical PHY IP block.

Figure 1. PHY IP Interface



Implementing an Interlaken Interface with the PHY IP

The Interlaken interface protocol allows the design of a narrow high-speed chip-to-chip interface for networking applications. The Interlaken interface's many advantages over XAUI and SPI 4.2 interfaces include low I/O count, flow control, low overhead framing, and extensive integrity checking. Interlaken is scalable and based on the SPI 4.2 structure to support 10-, 40-, and 100-Gbps systems. In an Interlaken configuration, Stratix V devices support a transmission data rate of 3.125, 5.0, 6.25, 6.375, or 10.3125 Gbps per lane.

Reference Design Flow for the Interlaken PHY IP example

This section describes the procedures to run the Interlaken PHY IP reference design with a Stratix V device.

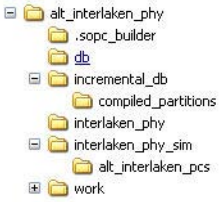
Hardware and Software requirement

Use the Quartus II software version 10.1 or beyond to implement and compile your design. Currently there is no hardware platform; you can only complete a reference design simulation.

The following PHY IP creation shows how to use the MegaWizard™ Plug-In Manager to generate the PHY IP (the PHY IP has already been created as part of the reference design).

Figure 2 shows the reference design directory structure.

Figure 2. Reference Design Directory Structure



Quartus II Software Top-Level Creation

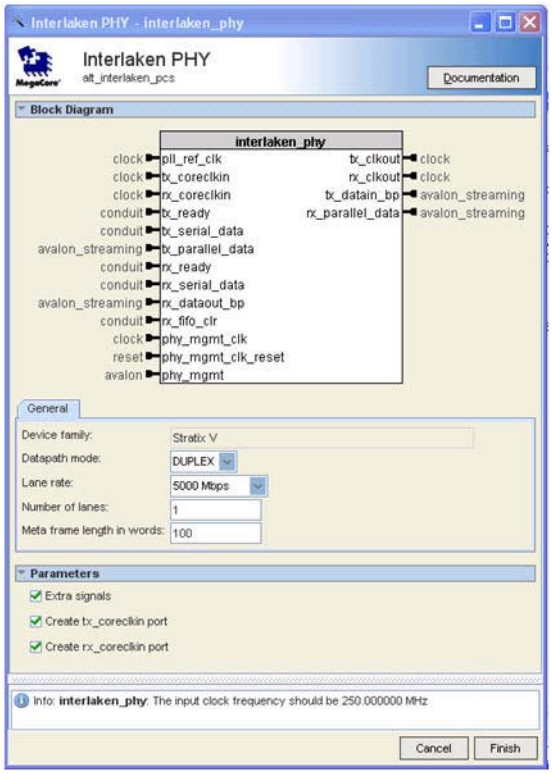
The PHY IP implementation makes the transceiver's protocol-specific interface a simple task when compared with using the ALTGX megafunction, where there are many variables to set for the transceiver.

To create a new Quartus II project file and a top-level design file, follow these steps:

1. Name the top-level verilog file **top_interlaken_phy.v**.
2. Launch the MegaWizard Plug-In Manager to create a new custom megafunction.
3. Select the **Interface Plug-Ins** drop-down menu and click **Interlaken PHY v10.1**. Select the output as verilog file **interlaken_ip**.
4. In the **Interlaken PHY** dialog box, you can select the datapath mode, lane rate, and number of lanes. In this example, select **duplex mode** for the receive and transmit with a **5 Gbps** data rate for one lane, as shown in Figure 3.
5. Set the assigned meta frame length to **100 word**.
6. Select the parameters interface signals. The extra signals are `word_lock`, `sync_lock`, and `CRC32 error` and are part of the `rx_parrellel_data` bus. An `tx_coreclkin` input port is created for driving the write side of the TX FIFO and an `rx_coreclkin` input port is created for driving the write side of the RX FIFO.
7. Click **Finish** to create the PHY IP.

Figure 3 shows the Interlaken PHY IP settings.

Figure 3. Interlaken PHY IP



The top-level verilog design consists of synthesizable code with an Avalon master, pseudo-random binary sequence (PRBS) generator and checker, and a frequency checker. All the necessary design files are included in the reference design package. All the modules were created as part of this reference design. Use an Avalon master for each PHY IP core you use in your design. The top-level design instantiates the following modules:

- ***Interlaken_phy*.v** (generated by the MegaWizard Plug-In Manager)
- ***Custom_avalon_master*.v**
- ***Prbs_generator*.v**
- ***Prbs_checker*.v**
- ***Frequency checker*.v**

Table 1 lists the important status and error signals.

Table 1. Status and Error Signals (Part 1 of 2)

Status and Error Signals	Description
rx_sync_lock	The sync_lock status signal indicates when four sync words are detected.
rx_word_lock	The word_lock status signal indicates when the first alignment pattern is detected.

Table 1. Status and Error Signals (Part 2 of 2)

Status and Error Signals	Description
rx_framing_error	Changes to "1" when a framing error is detected.
rx_crc32_error	Changes to "1" when an CRC32 error is detected.
rx_scrambler_error	Changes to "1" when a scrambler error is detected.
rx_sync_word_error	Changes to "1" when a missing_sync error is detected.
txclkfreq_error	Changes to "1" when the tx_clkout frequency is not within range.
rxclkfreq_error	Changes to "1" when the rx_clkout frequency is not within range.
data_error	Changes to "1" when the PRBS checker detects an error.
verifier_lock	Changes to "1" when the verifier detects a lock.

Custom Avalon Master

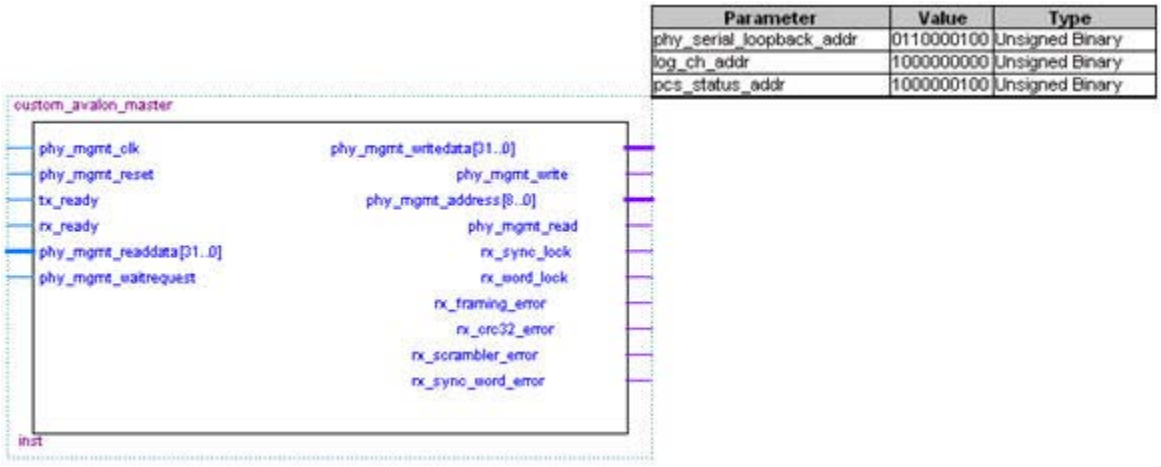
The Quartus II software version 10.1 and before does not allow the SOPC builder or Qsys to create any Avalon interface instance. Therefore, Altera designed a custom Avalon master, called ***custom_avalon_master*.v**, to accept the byte address and to provide the word address to the PHY IP.

The Avalon master accepts the byte address `mgmt_address[9:0]` and delivers the word `address mgmt_address[8:0]` to the PHY IP. The custom Avalon master was designed for an Interlaken interface, but you can use the same master for other PHY IP because they all have a similar structure. In this example, the internal serial loopback feature, which exercises the PMA's control and status register (CSR), is enabled.

The custom Avalon master is the heart of the interface between the FPGA logic core and the PHY IP. You can use this module as a template to modify and adapt to other PHY IP protocols. Each PHY IP protocol type will have a different set of internal registers to read or write through the Avalon master. The input and output data bus width between the PHY IP and the logic core is 32 bits.

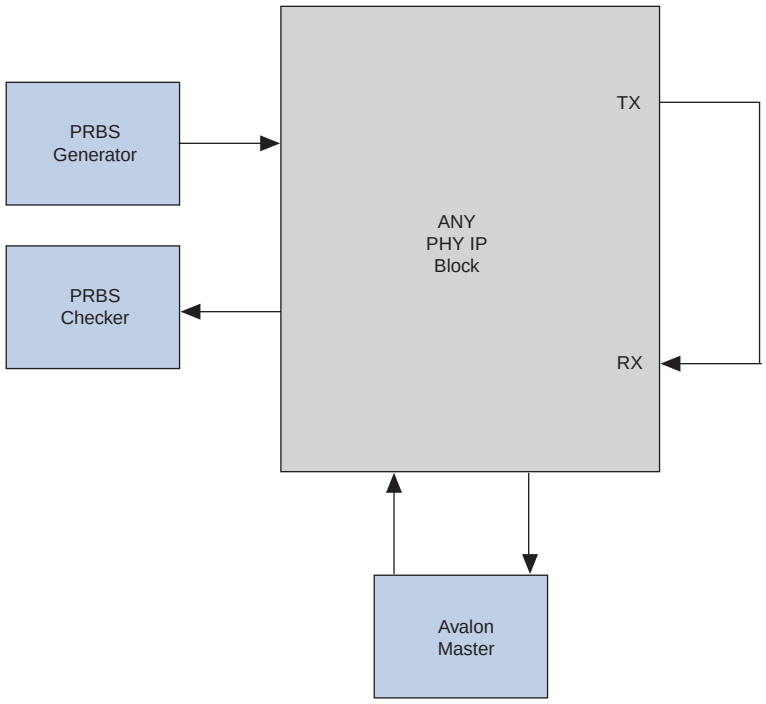
Figure 4 shows the custom_avalon_master signal flow, which was generated into a symbol.

Figure 4. Custom Avalon Master Signal Flow (custom_avalon_master)



Even though this reference design is for the Interlaken protocol, you can reuse some of the key building blocks for other PHY IP protocols. The code you can reuse in the reference design is the PRBS generator, PRBS checker, and, most importantly, the Avalon master module, as shown below in Figure 5.

Figure 5. Key Modules for PHY IP Testing



PRBS Generator and Checker

A 64-bit PRB23 data generator and check modules are used for loopback testing.

Top-Level Testbench

This section describes the register transfer level (RTL) simulation testbench and the simulation results of the reference design.

The RTL simulation testbench provides you with the RTL design verification at the software level. The reference design provides a simple testbench for the loopback environment between the TX and the RX function using a PRBS as a test pattern source. The testbench checks for the status and error signals and displays the required output information.

The device under test (DUT) is the PHY IP. A clock generator was created in the testbench for the PHY IP's clock interfaces, which includes the transceiver reference clock (`pll_ref_clk`) and the data path clock (`phy_mgmt_clk`).

Run the ModelSim Example Script

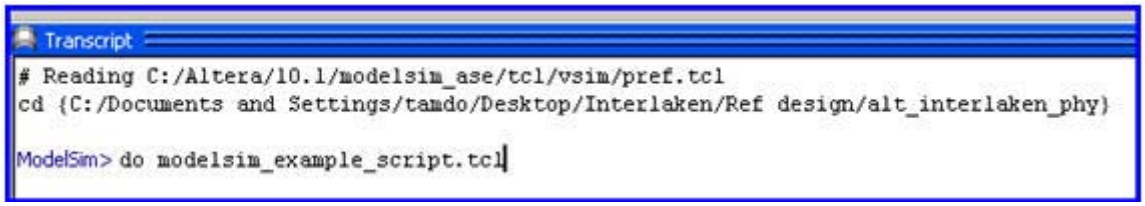
To run a simulation of the reference design in the ModelSim®-Altera software, follow these steps:

1. Start the ModelSim Starter Edition version 6.6c or beyond.
2. Under the File menu, click **Change Directory** and change to the location of the reference design that you have installed.

   ☞ Ensure the ***modelsim_example_script*.tcl** file is located the directory folder.

3. At the ModelSim command prompt, type `do modelsim_example_script.tcl` to run the simulation shown in Figure 6.

Figure 6. ModelSim Script



The **.tcl** script file is an example file for compilation and simulation of the Interlaken PHY instance in ModelSim. All the necessary verilog HDL or VHDL library files are included with the reference design project so that you can pick either language. You can modify this script file and use it to compile and simulate your design depending on new requirements.

Use the top-level testbench file ***top_interlaken_phy_tb*.v** to complete the simulation. This file instantiates the top-level PHY IP project under the test and clock generator. The testbench checks for the status and error signals and displays any error information.

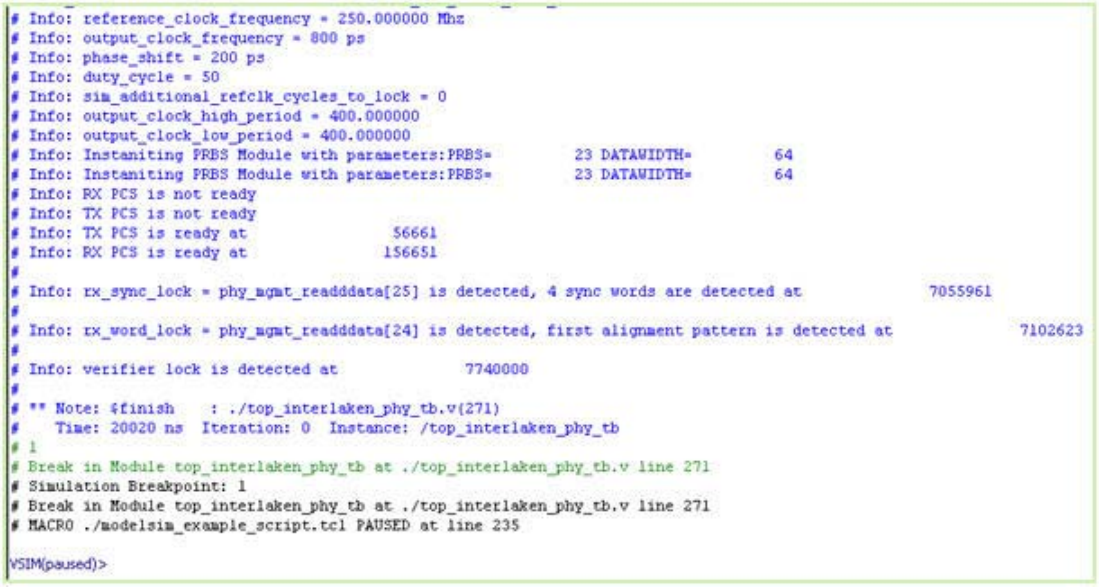
Simulation Results

A successful simulation has the following key status signals:

- TX PCS is ready for this example at 56661 ps
- RX PCS is ready for this example at 156651 ps
- `rx_sync_lock` at 7055961 ps
- `rx_word_lock` at 7102623 ps
- Verifier lock at 7740000 ps

Figure 7 shows the ModelSim simulation's output at the end of a successful simulation.

Figure 7. Successful ModelSim Simulation Results



You can further select the signals that you want to see from the ModelSim Objects window (Figure 8).

To select different signals to display in the Wave window (Figure 9), follow these steps:

1. Highlight the signal that you want to display in the Wave window (Figure 8).
2. On the Add menu, select **To wave** and **Signals in design** (Figure 9).
3. On the VSIM command prompt, type `run 1000` to start the waveform window.

Figure 8. ModelSim's Objects Window

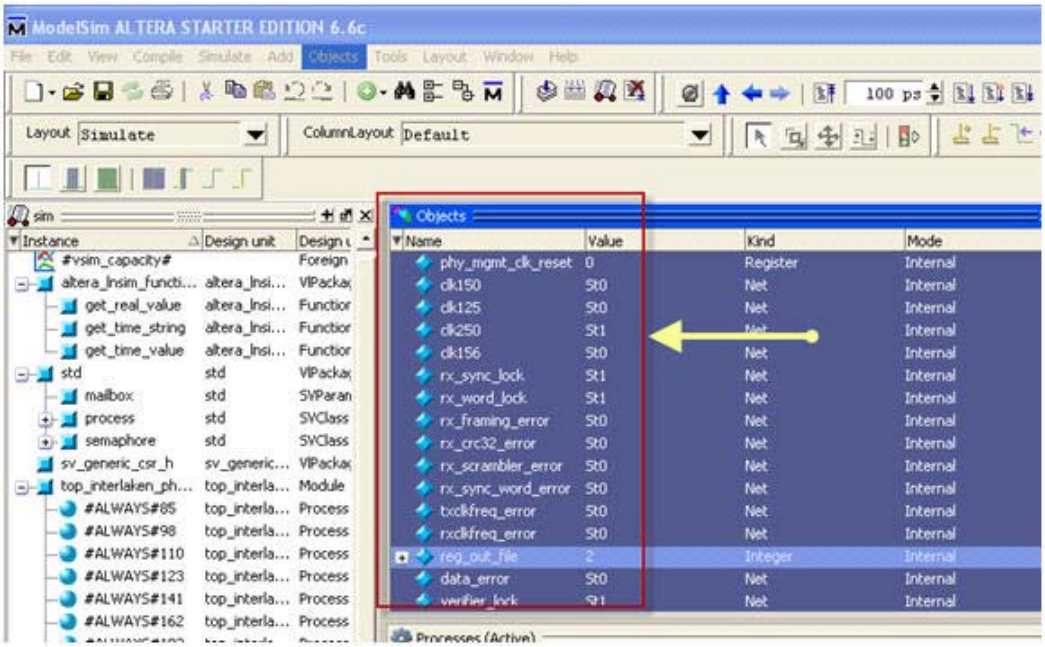
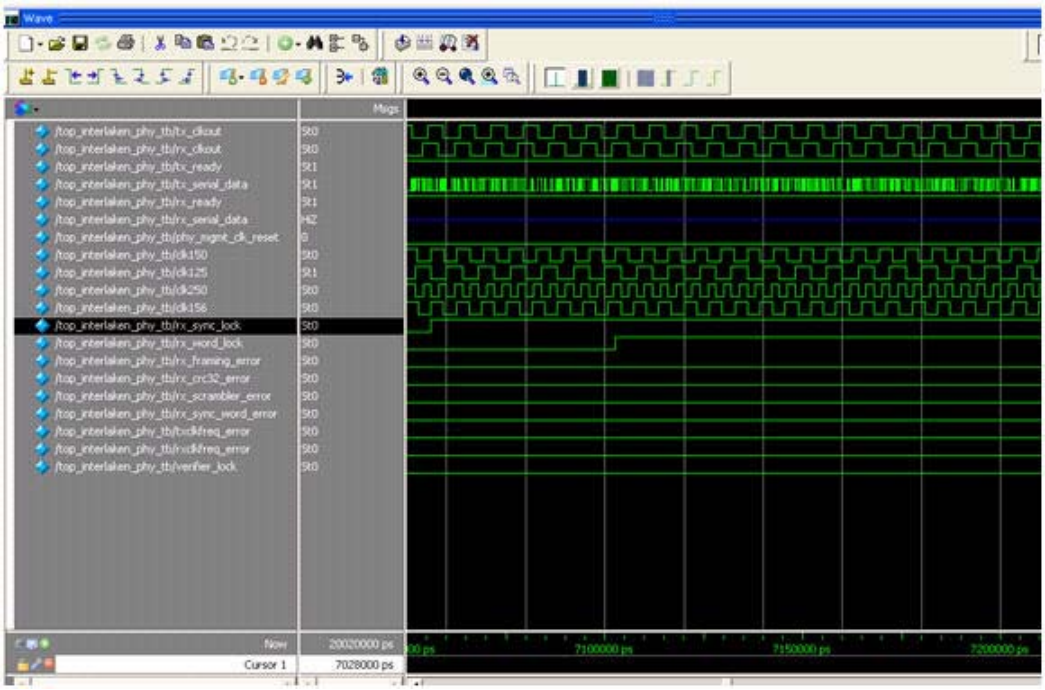


Figure 9. Waveform Window



Document Revision History

Table 2 lists the revision history for this application note.

Table 2. Document Revision History

Date	Version	Changes
December 2010	1.0	Initial release.