HyperTransport Single-Ended Slave Core

Implementation Guide

UG032 (v1.2) January 7, 2003

XILINX®

HyperTransport Single-Ended Slave Core
UG032 (v1.2) January 7, 2003

The following table shows the revision history for this document.

Date	Version	Revision
08/12/02	1.0	Initial release (draft).
08/30/02	1.1	Changed Xilinx Tools version.
01/07/03	1.2	Edited "XC2V3000-FF1152" on page 8.

Table of Contents

Chapter 1

Getting Started

Summary

The Xilinx LogiCORE™ HyperTransport™ 8-bit single-ended slave core provides a fully verified, pre-implemented HyperTransport interface. This interface supports Verilog HDL.

This implementation guide is intended to serve as a reference during the implementation phase of a project using the LogiCORE Interface. Since this guide is comprehensive in nature, some portions may not apply to your design.

This guide covers the supported design flows for the HyperTransport Interface targeting the Virtex™-II architecture. An example design located in the directory **<Install Path>/HT_8_SES_v1_0/Example** is included with the single-ended slave interface to demonstrate the design flows. Taking the time to synthesize and implement this example design can help you understand the Xilinx HyperTransport design flow.

Documentation

For more details on the LogiCORE HyperTransport single-ended slave interface, refer to the following documentation located in the Xilinx HyperTransport Lounge, accessible from: http://www.xilinx.com/systemio/htses/index.htm

- LogiCORE HyperTransport Data Sheet
- LogiCORE HyperTransport Design Guide

These documents are also located in the directory: **<Install Path>/HT_8_SES_v1_0/Docs**.

Further information is available at the HyperTransport Consortium web site at: http://www.hypertransport.org

The *HyperTransport I/O Link Specification* can be downloaded from the HyperTransport Consortium web site.

Support

The fastest method of obtaining technical support from the LogiCORE HyperTransport interface is through: http://support.xilinx.com

Questions can be routed to the team of engineers with specific expertise in using the LogiCORE HyperTransport interface. Xilinx provides technical support for use of the LogiCORE product as described in the *LogiCORE HyperTransport Design Guide* and the *LogiCORE HyperTransport Implementation Guide*. Xilinx cannot guarantee timing, functionality, or support of the LogiCORE product for designs that do not follow these guidelines.

Family-Specific Considerations

The supplied core supports an 8-bit data bus width at 200 MHz DDR (400 Mb/s) or 400 MHz DDR (800 Mb/s) operation when targeting a supported part and footprint. This release supports the following devices and footprints:

- XC2V3000-FF1152
- XC2V4000-FF1152
- XC2V6000-FF1152

Refer to Table 1-1 for a list of supported device and interface combinations.

Table 1-1: **Device and Interface Selection**

Supported Device	Operation Speed	Constraints File
XC2V3000FF1152-4	200 MHz DDR (400 Mb/s)	xc2v3000ff1152_200.ucf
XC2V4000FF1152-4	200 MHz DDR (400 Mb/s)	xc2v4000ff1152_200.ucf
XC2V6000FF1152-4	200 MHz DDR (400 Mb/s)	xc2v6000ff1152_200.ucf
XC2V3000FF1152-5	400 MHz DDR (800 Mb/s)	xc2v3000ff1152_400.ucf
XC2V4000FF1152-5	400 MHz DDR (800 Mb/s)	xc2v4000ff1152_400.ucf
XC2V6000FF1152-5	400 MHz DDR (800 Mb/s)	xc2v6000ff1152_400.ucf

The wrapper file, `ses_top.v`, located in the **<Install_Path>/HT_8_SES_v1_0/Verilog/template** directory, contains an instance of the HyperTransport interface.

When you are ready to begin a new design, modify this wrapper to include all user I/O elements and modules. Instructions on allowable modifications are located in the wrapper file.

The constraints file, located in the **<Install_Path>/HT_8_SES_v1_0/Implement/ucf** directory, contains various constraints required for the HyperTransport interface. The constraints file must always be used while processing a design and is specific to the target device.

The design is pin locked. Movement of the HyperTransport interface to other locations on the device invalidates the supplied pinout and will likely result in the inability to place and route to the clock frequencies required to support the interface.

Running the Example Design

This chapter describes the design flow for the example design provided in the **<Install_Path>/HT_8_SES_v1_0/Example** directory.

Tools Support

The LogiCORE Physical Layer supports the following tools:

- Simulation: Cadence Verilog XL and Model Technology ModelSim
- Synthesis: Synplicity Synplify v7.10 or higher
- Xilinx Tools: Foundation/Alliance v5.1i

Example

The example provided allows users to synthesize and implement the HyperTransport single-ended slave interface.

The demo test bench design consists of several files and two distinct components, including an example HyperTransport host and an example user application. The demo testbench is written in Verilog design language. The user application along with associated files is fully synthesizable. The host model is for simulation purposes only and will not synthesize. The host model and SES application are instantiated in a testbench top level called, `test_top.v`. The demo testbench architecture is shown in Figure 2-1 .

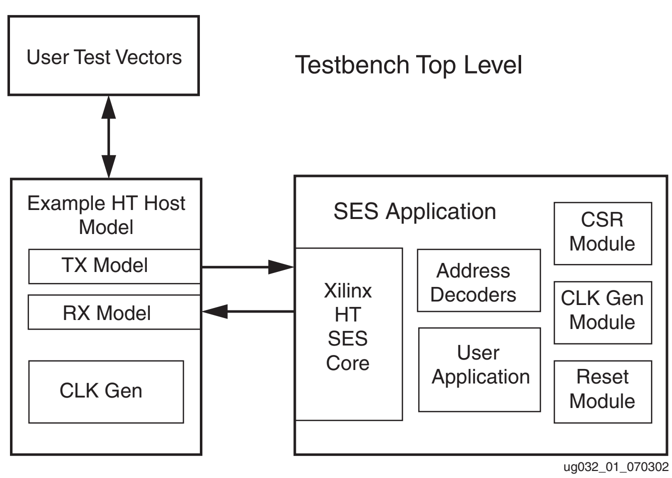

Figure 2-1: **Example Design System Architecture Block Diagram**

HT Host Model

The HT host model is designed to provide the bare minimum of functionality to demonstrate some transactions over a HyperTransport link. The host is composed of five major files. The file hierarchy is as follows:

- HyperTransport Host Top Level (`ht_host_top.v`)
  - Tx Model (`ht_host_tx.v`)
    - CRC Generation Module (`crc_check.v`)
  - Rx Model (`ht_host_rx.v`)
    - CRC Generation Module (`crc_check.v`)
  - Clock Generation Module (`clk_gen.v`)

HyperTransport Host Top Level

This file instantiates the lower level files. The file name for this module is `ht_host_top.v.`

Tx Model

This file models the Tx port of a HyperTransport host. It produces read and write requests. In the provided example, the following HyperTransport commands are supported:

- Posted Writes 64 bytes long
- Non-posted Reads 64 bytes long
- Host Buffer available data
- Link CRC generation and transmission

The Model also generates the required link CRC values and transmits them at the required time as described in the *HyperTransport Specification*. The file name for this module is `ht_host_tx.v`

CRC Generation Module

This module generates a running CRC on the link to which it is attached. An interrupt timer is included to notify the link model that it is time to send the CRC or check the incoming CRC. This module is used in both the Tx and Rx models. The file name for this module is `crc_check.v`.

Rx Model

This file models the Rx port of a HyperTransport host. It accepts input responses and processes them as required. The supported responses are as follows:

- Read responses of 64 bytes long
- Slave buffer available data
- Link CRC generation and verification

Input read response data is collected and used to verify the returned data to what was expected. The model generates a CRC value of the link and compares it with the input CRC data. Errors are flagged through the simulation log file. Input buffer data is also accepted and interpreted. This is done in the manner described in the HyperTransport Specification. The file name for this module is `ht_host_rx.v`.

Clock Generation

This module produces the required clock signals for the Host model to communicate with the SES application and to process data. The file name for this module is `clk_gen.v`.

SES Application

The SES application contains a synthesizable example using the Xilinx SES Core. The example consists of the SES Core, a user application, CSR modules, clock and reset modules and address decoders. The file hierarchy of the SES application is as follows:

- SES Application Top Level (`ses_top.v`)
  - SES Core (`ldtsx_LDTSESX_PHYCORE.v/ ldtsx_LDTSESX.v`)
  - Address Decoders (`ses_addec.v`)
    - Address Decoder Submodule (`ses_addec_fixaddec.v`)
    - Address Decoder Submodule (`ses_addec_brpriaddec.v`)
  - Clock Generation Module (`ses_clk_gen.v`)
  - Reset Module (`ses_reset.v`)
  - User Application (`ses_user_app_.v`)
  - CSR Module (`ses_csr.v`)

SES Application Top Level

This contains the instantiations of the SES application modules. The file name is `ses_top.v`.

SES Core

This is the Core simulation file or Core instantiation file. For synthesis, the instantiation file must be used. This file is called `ldtsx_LDTSESX_PHYCORE.v`. For simulation, the simulation core file must be used. This file is called `ldtsx_LDTSESX.v`.

Address Decoders

The address decoders decode the incoming address on request packets and notify the SES Core if this is within the address space of the application. The file names for the address decoders are `ses_addec.v`, `ses_addec_fixaddec.v`, and `ses_addec_brpriaddec.v`.

Clock Generation Module

This module contains the instantiations of the DCMs and Global buffers necessary to generate the required clocks for the core. The file name for this module is `ses_clk_gen.v`.

Reset Module

This module contains the reset logic for the SES application. The file name is `ses_reset.v`.

Functional Simulation

Functional simulation of the HyperTransport single-ended slave interface is supported by the following tools:

- Cadence Verilog-XL
- Model Technology ModelSim (PE/EE/SE)

The example design includes a simple test bench. This testbench can be leveraged to create larger test benches. The example design is a simple user application which is intended for use as a training vehicle and design flow test.

Cadence Verilog-XL

Before attempting functional simulation, ensure that the Verilog-XL environment is properly configured for use.

1. To begin, move into the example functional simulation directory:

   ```
   cd <Install Path>/HT_8_SES_v1_0/Example/sim_scripts/func_sim
   ```

2. Edit the `test_tb.f` file. This file lists command line arguments for Verilog-XL and is shown below:

   ```
   ../../Verilog/sim_model/glbl.v
   ../../Verilog/sim_model/ldtsx_LDTSESX.v
   ../verilog_ex_src/buffer_fifo.v
   ../verilog_ex_src/main_memory.v
   ../verilog_ex_src/tag_fifo.v
   ../verilog_ex_src/ses_addec_fixaddec.v
   ../verilog_ex_src/ses_addec_bripriaddec.v
   ../verilog_ex_src/ses_addec.v
   ../verilog_ex_src/ses_clk_gen.v
   ../verilog_ex_src/ses_reset.v
   ../verilog_ex_src/ses_user_app.v
   ../verilog_ex_src/ses_csr.v
   ../verilog_ex_src/ses_top.v
   ../verilog_ex_src/clk_gen.v
   ../verilog_ex_src/crc_check.v
   ../verilog_ex_src/ht_host_tx.v
   ../verilog_ex_src/ht_host_rx.v
   ../verilog_ex_src/ht_host_top.v
   ../verilog_ex_src/test_top.v
   +libext+.vmd+.v
   -y <Xilinx Install Path>/verilog/src/unisims
   -y <Xilinx Install Path>/verilog/src/simprims
   -y <Xilinx Install Path>/verilog/src/XilinxCoreLib
   ```

   Modify the library search path by changing <**Xilinx Install Path**> to match the Xilinx installation directory. Save the file.

   Most of the files listed are related to the example design and its testbench. For other test benches, the following subset must be used for proper simulation of the single-ended slave interface.

   ```
   +licq_all+
   ../../Verilog/sim_model/glbl.v
   ../../Verilog/sim_model/ldtsx_LDTSESX.v
   ../../Verilog/template/ses_top.v
   +libext+.vmd+.v
   -y <Xilinx Install Path>/verilog/src/unisims
   ```

```
-y <Xilinx Install Path>/verilog/src/simprim
```

3. To run the Verilog-XL simulation:

   ```
   verilog -f test_tb.f
   ```

   Verilog-XL processes the simulation files and exits. The testbench prints status messages to the console. After the simulation completes, view the `verilog.log` file to check for errors.

4. The Signalscan browser can be used to view the simulation results. Signalscan is started with the following command:

   ```
   signalscan
   ```

Model Technology ModelSim

Before attempting functional simulation, ensure that the ModelSim environment is properly configured for use.

1. To begin, move into the example functional simulation directory:

   ```
   cd <Install Path>/HT_8_SES_v1_0/Example/sim_scripts/func_sim
   ```

2. Edit the `test_tb.f` file. This file lists command line arguments for ModelSim and is shown below:

   ```
   +licq_all+
   ../../Verilog/sim_model/glbl.v
   ../../Verilog/sim_model/ldtsx_LDTSESX.v
   ../verilog_ex_src/buffer_fifo.v
   ../verilog_ex_src/main_memory.v
   ../verilog_ex_src/tag_fifo.v
   ../verilog_ex_src/ses_addec_fixaddec.v
   ../verilog_ex_src/ses_addec_bripriaddec.v
   ../verilog_ex_src/ses_addec.v
   ../verilog_ex_src/ses_clk_gen.v
   ../verilog_ex_src/ses_reset.v
   ../verilog_ex_src/ses_user_app.v
   ../verilog_ex_src/ses_csr.v
   ../verilog_ex_src/ses_top.v
   ../verilog_ex_src/clk_gen.v
   ../verilog_ex_src/crc_check.v
   ../verilog_ex_src/ht_host_tx.v
   ../verilog_ex_src/ht_host_rx.v
   ../verilog_ex_src/ht_host_top.v
   ../verilog_ex_src/test_top.v
   +libext+.vmd+.v
   -y <Xilinx Install Path>/verilog/src/unisims
   -y <Xilinx Install Path>/verilog/src/simprims
   -y <Xilinx Install Path>/verilog/src/XilinxCoreLib
   ```

   Modify the library search path by changing <**Xilinx Install Path**> to match the Xilinx installation directory. Save the file.

   Most of the files listed are related to the example design and its testbench. For other testbenches, the following subset must be used for proper simulation of the single-ended slave interface.

   ```
   +licq_all+
   ../../Verilog/sim_model/glbl.v
   ../../Verilog/sim_model/ldtsx_LDTSESX.v
   ../../Verilog/template/ses_top.v
   ```

```
+libext+.vmd+.v
-y <Xilinx Install Path>/verilog/src/unisims
-y <Xilinx Install Path>/verilog/src/simprim
```

3. Invoke ModelSim and ensure that the current directory is set to:

   ```
   <Install Path>/HT_8_SES_v1_0/Example/sim_scripts/func_sim
   ```

4. Create the work directory by typing the following in the ModelSim command window:

   ```
   vlib work
   ```

5. To run the simulation, use the following command:

   ```
   do modelsim.do
   ```

   This compiles all modules, loads them into the simulator, displays the waveform viewer, and runs the simulation.

Synthesis

Synthesis of the HyperTransport single-ended slave interface is supported by Synplicity Synplify v7.10.

The **hypertransport.prj** file points to all of the files necessary to run through the example. This file is provided in the **<Install_Path>/HT_8_SES_v1_0/Example/syn_scripts** directory.

The project file sets the include directory to the **/Example/verilog_ex_src** directory. The include directory allows Synplicity to read in the various files necessary to synthesize the example design.

The synthesis project can be run in one of two ways. The first method uses the script capability of Synplicity. Ensure that the Synplify environment is properly configured for use and enter the following at the command line:

```
synplify_pro -batch hyptertransport.prj
```

The second option to synthesize the project requires that the Synplify GUI be started. Once the GUI is started, load the project file **hypertransport.prj**. Click the run button to run through the synthesis flow.

Both methods result in an EDIF file being generated in the **syn_results** directory. This EDIF file will be used as the top-level design during implementation

Note: It is important that the top-level instance name stay the same and that the ldtsx_LDTSESX_PHYCORE module always be instantiated in the top level. These restrictions are necessary in order to ensure that the **ucf** file contains the correct hierarchy. Advanced users may change the hierarchy structure but must also modify the **ucf** file so that the design works correctly.

FPGA Implementation

A script file is provided to run the example EDIF file through the Xilinx tools. This script file sets all options necessary to correctly implement the design. To use the script file, ensure that Xilinx environment is properly configured.

From the **<Install_Path>/HT_8_SES_v1_0/Example/fpga_scripts** directory, run the appropriate script from the command line:

- run_m4 – Unix
- run_m4.bat – MS DOS

It may be helpful to open the script file and become familiar with the different commands used during implementation. For more information on the implementation commands and associated options, see the *Development System Reference Guide* section of the 5.1i software manuals.

Timing Simulation

Timing simulation of the example is not supported for this release.

Chapter 3

Setting Up the User Design

This chapter describes the user design flow. A design directory has been provided in the **<Install_Path>/HT_8_SES_v1_0/Implement** directory. This directory contains subdirectories for the user to synthesize and implement in. The user is not provided with a simulation directory. However, instructions on the location and the use of the simulation model are provided below.

The **<Install_Path>/HT_8_SES_v1_0/Implement/design_src** directory contains the design netlist that will be instantiated in the user design. See "Synthesis" for instructions on how to set up the user design to instantiate the single-ended slave interface.

Tools Support

The LogiCORE single-ended slave interface supports the following tools:

- Simulation: Cadence Verilog XL and Model Technology ModelSim
- Synthesis: Synplicity Synplify v7.10
- Xilinx Tools: Foundation/Alliance v5.1i

Functional Simulation

The HyperTransport interface Verilog simulation model is located at:

<Install_Path>/HT_8_SES_v1_0/Verilog/sim_model/ldtsx_LDTSESX.v.

To simulate with this model, point to the simulation libraries found in the Xilinx Foundation/Alliance 5.1i release. An example of this can be found in the **ldt_tb.f** file located in the **<Install_Path>/HT_8_SES_v1_0/implement/func_sim** directory.

A **do** file provided for use with ModelSim calls the **ldt_tb.f** file. When using Cadence Verilog-XL, use a command line command similar to: `verilog -f ldt_tb.f`.

See the "Functional Simulation"section under Chapter 2, "Running the Example Design" for detailed instructions on using Verilog-XL or ModelSim when simulating the HyperTransport core.

Synthesis

Xilinx provides an example Synplicity project which loads the top-level file, the configuration file, and the black box instantiation of the core. The user can add any other necessary files. The project file, **hypertransport.prj**, is provided in the **<Install_Path>/HT_8_SES_v1_0/Implement/syn_scripts** directory.

The synthesis project can be run in one of two ways. The first method uses the script capability of Synplicity. Ensure that the Synplify environment is properly configured for use and enter the following at the command line:

`synplify_pro -batch hypertransport.prj`.

The second option to synthesize the project requires that the Synplify GUI be started. Once the GUI is started, load the project file **hypertransport.prj**. Click the run button to run through the synthesis flow.

Both methods result in an EDIF file being generated in the **syn_results** directory. This EDIF file will be used as the top-level design during implementation.

Xilinx provides an example design using the HyperTransport LogiCORE to help users become familiar with the synthesis flow. See the "Synthesis" section under Chapter 2, "Running the Example Design" for more information.

Note: It is important that the top-level instance name stays the same and that the ldtsx_LDTSESX_PHYCORE module always be instantiated in the top level. These restrictions are necessary in order to ensure that the **ucf** file contains the correct hierarchy. Advanced users may change the hierarchy structure but must also modify the **ucf** file so that the design works correctly.

FPGA Implementation

A script file is provided to run the EDIF file through the Xilinx tools. This script file sets all options necessary to correctly implement the design. To use the script file, ensure that the Xilinx environment is properly configured.

From the **<Install_Path>/HT_8_SES_v1_0/Implement/fpga_scripts** directory, run the appropriate script from the command line:

- run_m4 – Unix
- run_m4.bat – MS DOS

It may be helpful to open the script file and become familiar with the different commands used during implementation. For more information on the implementation commands and associated options, see the *Development System Reference Guide* section of the 5.1i software manuals.