ISim Hardware Co-Simulation Tutorial: Accelerating Floating Point FFT Simulation

UG817 (v 14.3) October 16, 2012

This tutorial document was last validated using the following software version: ISE Design Suite 14.3

If using a later software version, there may be minor differences between the images and results shown in this document with what you will see in the Design Suite.

Notice of Disclaimer

Revision History

The following table shows the revision history for this document.

Date	Version	Description
03/18/2011	13.1	Initial release
07/06/2011	13.2	Release updates
11/04/2011	13.3	Updated with modifications to match release. Added: • Revision History (this topic) • Additional Resources Appendix • Determining the Ethernet Port Appendix Updated: • Creating a Testbench • Graphics throughout
01/18/2012	13.4	Miscellaneous filename changes throughout.
04/24/2012	14.1	Changed FPGA device to Kintex™-7 KC705. Changed screen shots to reflect the new device.
10/16/2012	14.3	Updated Version and Date.

Table of Contents

Revision History 2

Chapter 1 Introduction 5

Prerequisites 5

Tutorial Files 6

Chapter 2 Tutorial 7

Step 1: Generating a FFT Core in CORE Generator 7

Step 2: Creating a Test Bench 12

Step 3: Compiling the Design for Hardware Co-Simulation 13

Step 4: Running ISim Hardware Co-Simulation 16

Appendix A Additional Resources 17

Appendix B Determining the Ethernet Port 19

Introduction

This tutorial describes how to use ISim Hardware Co-Simulation (HWCoSim) to accelerate the simulation of Floating Point Unit (FPU) Fast Fourier Transform (FFT) and verify the FFT implementation on a Kintex™-7 KC705 board.

Digital Signal Processing (DSP) designs are typically very time-consuming to simulate in software due to their data and computation intensiveness.

- A fast, bit-accurate model is often used to speed up the simulation of a DSP function, but it does not provide any cycle accuracy and is not straightforward to integrate with other Register Transfer Level (RTL) modules.
- A behavioral RTL model provides bit-and-cycle accuracy but it is relatively slower to simulate. A structural Register Transfer Level (RTL) or gate-level model is even much slower to simulate.
- Sometimes an IP does not provide a fast, bit-accurate model or even a behavioral RTL model, which leaves the slowest structural/gate-level simulation as the only choice.

ISim hardware co-simulation provides an additional means to simulate DSP functions by off loading intensive computations to FPGA. It can bring synthesizable HDL code, synthesized or protected netlists such as IP cores generated by CORE Generator™ software into FPGA for co-simulation. That solves the problem of getting bit-and-cycle accurate simulation models as well as bolstering the simulation performance. For many complex DSP designs, not only does it accelerate the simulation of a design, it verifies the implementation of the design on actual hardware. ISim hardware co-simulation complements RTL, post-synthesis, and post-implementation simulation to complete the verification tool suite.

Prerequisites

- ISE Design Suite version 14.1 or above
- Kintex™-7 FPGA KC705 Evaluation Kit
- Tutorial Files: kc705_fp_fft_hwCoSim_tutorial.zip

Tutorial Files

File	Description
fp_fft_top.v	Wrapper that instantiates the floating point FFT core.
fp_fft_tb.vhd	Top-level VHDL test bench that generates test vectors to exercise the FFT core.
isim_run.tcl	Custom simulation command file that measures the ISim simulation time.
custom_wave_config.tcl	Custom waveform configuration file.
fp_fft_tb.prj	ISim project file for the command line flow.
full_compile.bat	Windows batch file to fully compile the design for hardware co-simulation with the `Fuse` command line.
full_compile.sh	Linux shell script to fully compile the design for hardware co-simulation with the `Fuse` command line.
incr_compile.bat	Windows batch file to incrementally compile the test bench for hardware co-simulation with the Fuse command line.
incr_compile.sh	Linux shell script to incrementally compile the test bench for hardware co-simulation with the `Fuse` command line.
run_isim.bat	Windows batch file to launch the ISim simulation.
run_isim.sh	Linux shell script to launch the ISim simulation.

Note When performing this tutorial, all data files must be copied to your current working directory.

Tutorial

This tutorial describes how to use ISim Hardware Co-simulation to accelerate the simulation of Floating Point Unit (PFU) Fast Fourier Transform (FFT) and verify the FFT implementation on a Xilinx® KC705 board.

This tutorial has four sections with the steps you need to run an FFT design through ISim hardware co-simulation. Perform the steps in the order that they are presented. These sections are as follows:

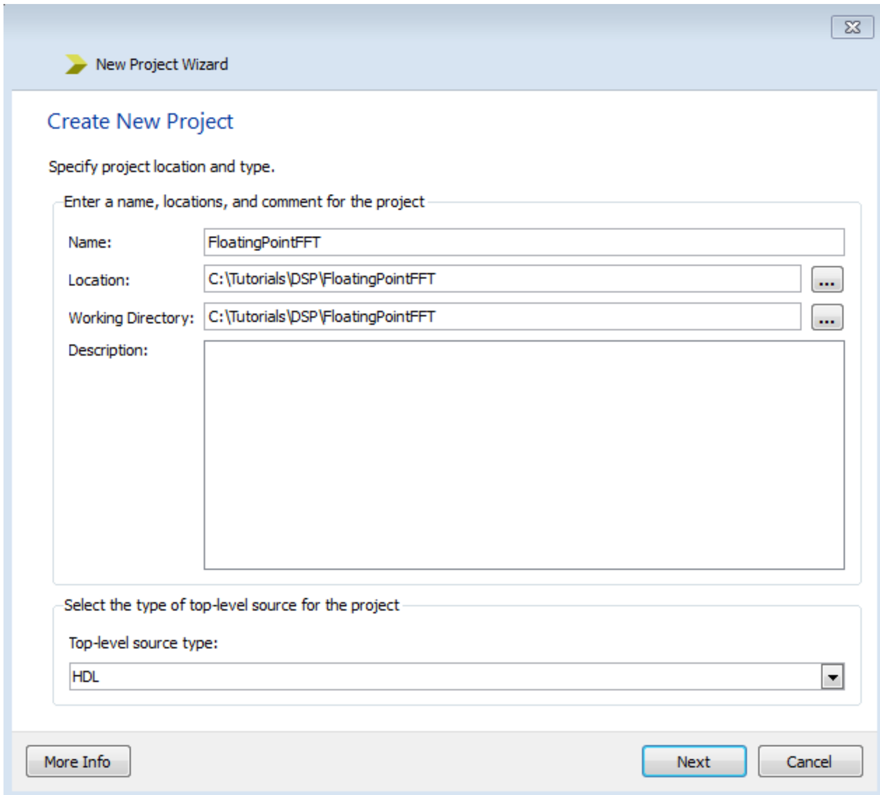
1. Generating a FFT Core in CORE Generator™
2. Creating a Test Bench
3. Compiling the Design for Hardware Co-Simulation
4. Running ISim Hardware Co-Simulation

Step 1: Generating a FFT Core in CORE Generator

In this tutorial, you use the Fast Fourier Transform (FFT) IP core generated from CORE Generator™ tool and create an ISim Hardware Co-Simulation (HWCoSim) test bench that runs on the Kintex™-7 FPGA KC705 Evaluation Kit.

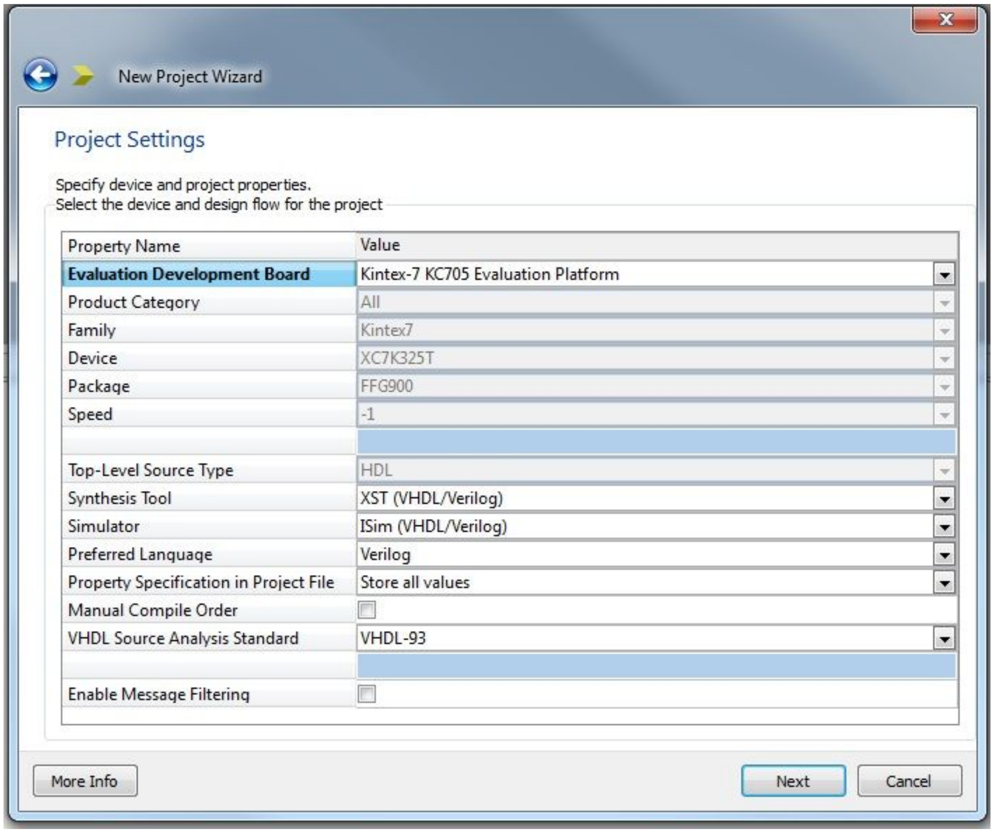
Note The screen shots captured in this tutorial are based on the Fast Fourier Transform version 8.0. The CORE Generator GUI might look different in later versions of the tool.

1. Launch the ISE® Design Suite Project Navigator.
2. Select **File > New Project** to open the New Project Wizard. Enter a project name (`FloatingPointFFT`) and location. Click **Next**.

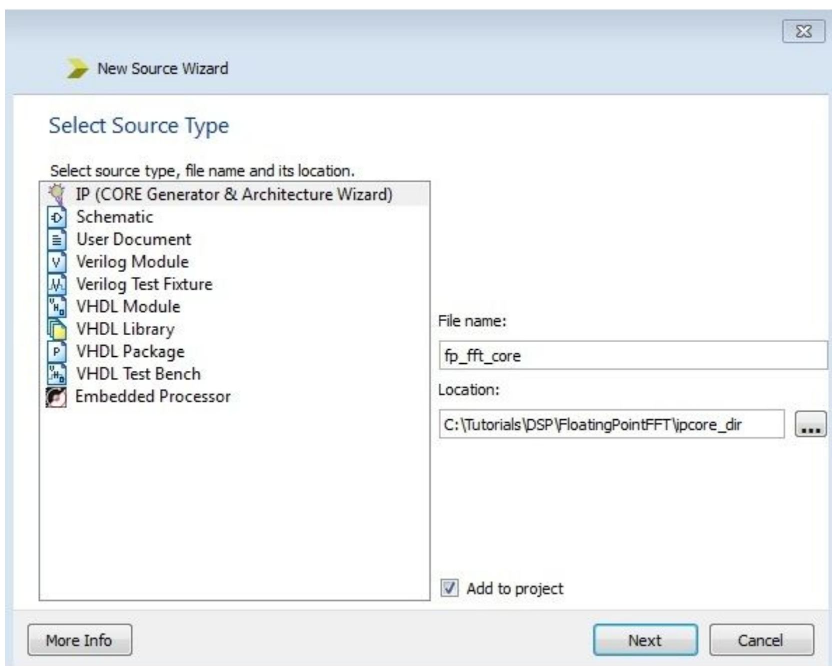


3. On the **Project Settings** page:

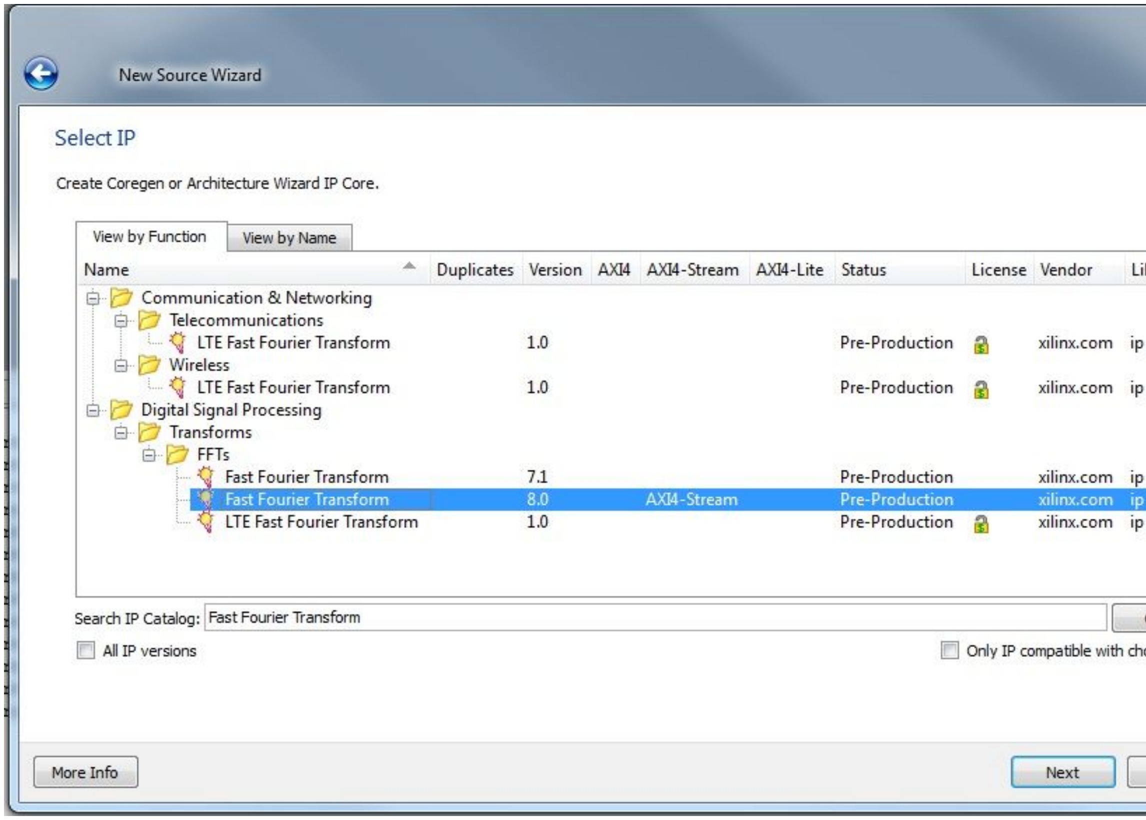
    - Choose Evaluation Development Board as KC705 board, which is a Kintex-7 KC705 Evaluation Platform.
    - Ensure the following default value settings:
        – Family: Kintex7
        – Device: XC7K325T
        – Package: FFG900
        – Speed: –1
    - Ensure the following HDL source-specific settings:
        – Synthesis Tool: XST (VHDL/Verilog)
        – Simulator: ISim (VHDL/Verilog)
        – Preferred Language: Verilog
    - In the Project Summary page, ensure all project settings are correct and click **Finish** to complete the project creation



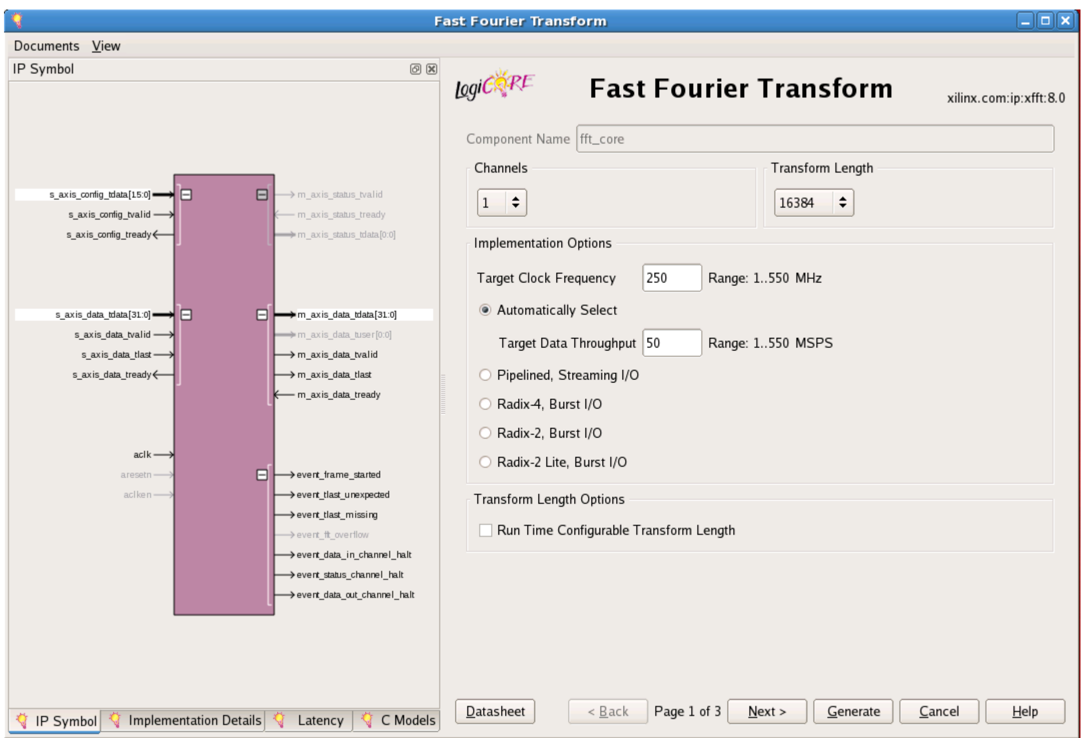
4. Choose **Project > New Source** to open the New Source Wizard. Select **IP (CORE Generator & Architecture Wizard)** and name the IP as **`fp_fft_core`**. Click **Next**.



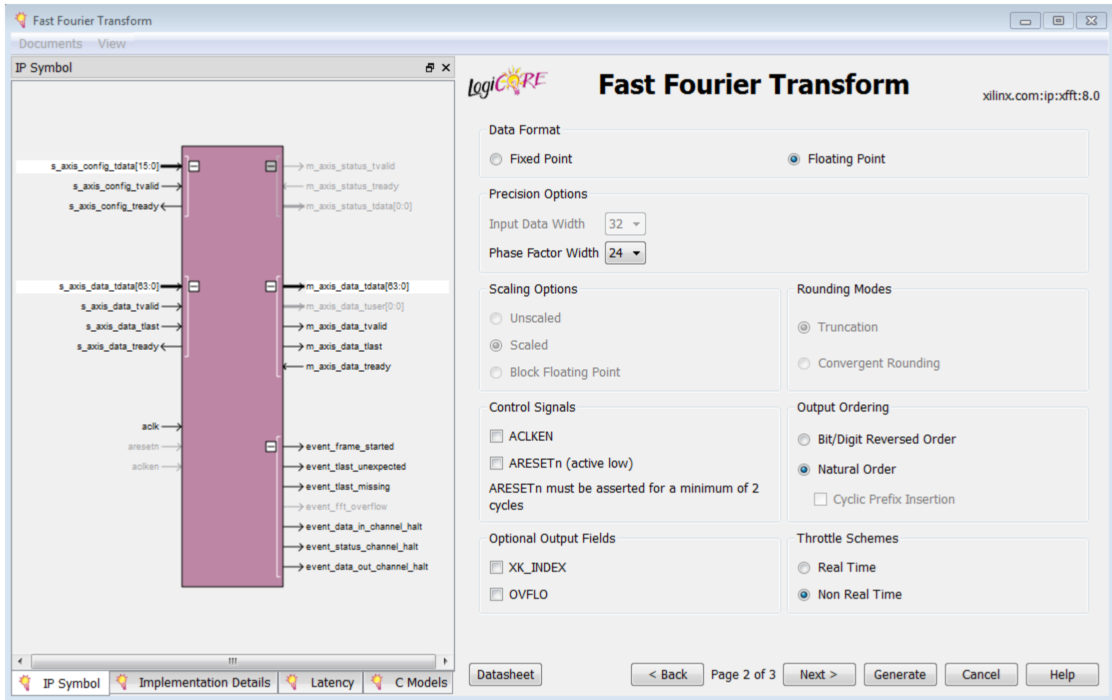
5. From Select IP Wizard, choose Fast Fourier Transform 8.0 under **Digital Signal Processing -> Transforms -> FFTs**. Click **Next,** and then click **Finish**.



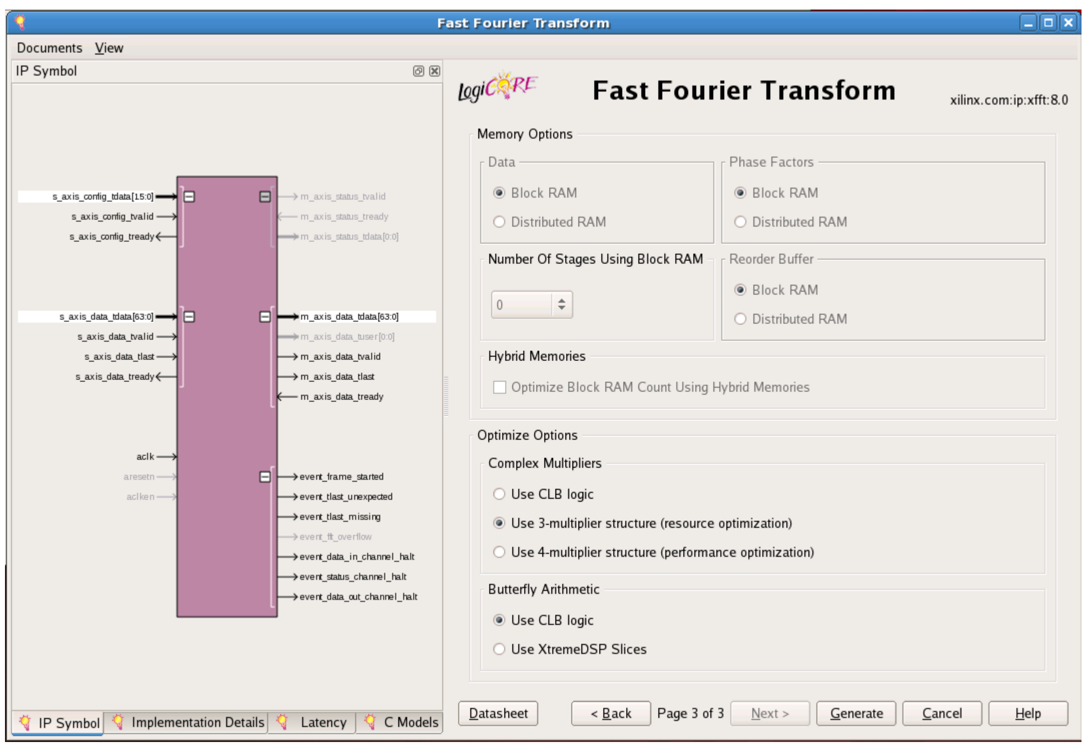
6. When the FFT core GUI opens, set the **Transform Length** to **16384**and click **Next**.



7. Set the **Data Format** to **Floating Point**, and the **Output Ordering** to **Natural Order**. Click **Next**.



8. Choose **Use 3-multiplier structure (resource optimization)** for the **Complex Multipliers** option. Click **Generate** to generate the core.



9. After the CORE Generator finishes successfully, Add a top-level module, `fp_fft_top`, that instantiates the generated `fp_fft_core` IP core. This is required because ISim hardware co-simulation only supports a Hardware Description Language (HDL) top-level module. You can use the completed `fp_fft_top.v` provided in this tutorial. Choose **Project > Add Source > fp_fft_top.v**, select **Open** and then click **OK**.

Step 2: Creating a Test Bench

1. Add a VHDL test bench module `fp_fft_tb` that generates test vectors to exercise the `fp_fft_top` instance. You can use the completed `fp_fft_tb.vhd` file provided in this tutorial.
2. Select **Project > Add Source > fp_fft_tb.vhd**.
3. Click **Open**, and then click **OK**.

The VHDL test bench has a constant `IP_DATA` which is an array of depth 16384 of records `T_IP_SAMPLE` for storing the 32-bit real and imaginary components of the FFT input. When the simulation starts, the test bench generates the FFT input vector using `create_ip_table` function, for the first frame.

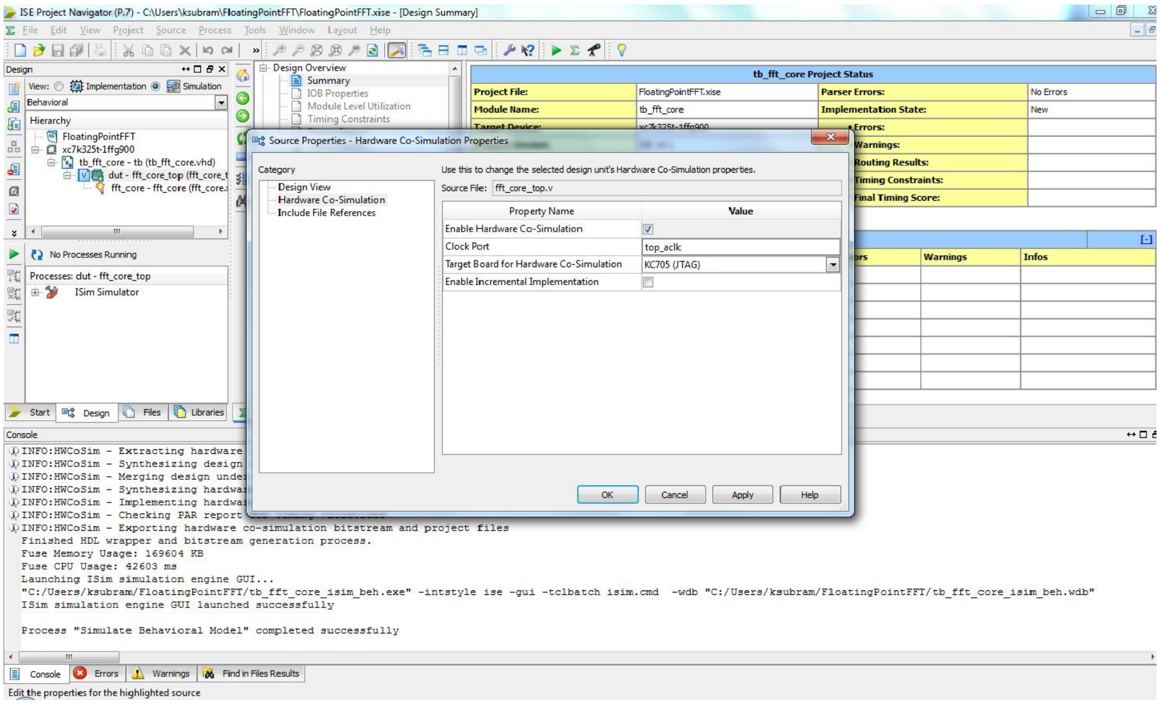
The operations performed by the demonstration test bench are appropriate for the configuration of the generated core, and are a subset of the following operations:

- Frame 1: Drive a frame of pre-generated input data from the constant array `IP_DATA`.
- Frame 2: Configure an inverse transform; drive the output of Frame 1 as a frame of input data.
- Configure Frame 3: A forward transform while the previous transform is running.
- Frame 3: Drive the output of Frame 2 as a frame of input data; de-assert AXI `TVALID` (and `TREADY`, if present) signals occasionally to demonstrate AXI handshaking
- Frames 4-7: Run these frames back-to-back, as quickly as possible:
  - Queue up configurations for a forward transform (Frame 4) followed by a reverse transform (Frame 5), both with a smaller point size (if the point size is configurable) and a short cyclic prefix (if available).
  - Frame 4: Drive a frame of pre-generated input data.
  - Frame 5: Drive the output of Frame 1 as a frame of input data; simultaneously configure Frame 6: A forward transform with maximum point size, a longer cyclic prefix (if available) and a zero scaling schedule (if fixed scaling is used).
  - Frame 6: Drive a frame of pre-generated input data; simultaneously configure Frame 7: An inverse transform with maximum point size, no cyclic prefix and default scaling schedule (if fixed scaling is used)
  - Frame 7: drive the output of frame 1 as a frame of input data.
- Wait until all frames are complete.

Step 3: Compiling the Design for Hardware Co-Simulation

After you create the test bench, you can compile the design for hardware co-simulation using the ISim compiler. This can be done in Project Navigator by enabling hardware co-simulation on a selected instance in your design. The selected instance, including its sub-modules, are co-simulated in hardware during the ISim simulation. Other modules are simulated in software.

1. In Project Navigator, switch to the **Simulation View**. From the Hierarchy Pane, right-click the `fp_fft_top` instance, and click **Source Properties**.

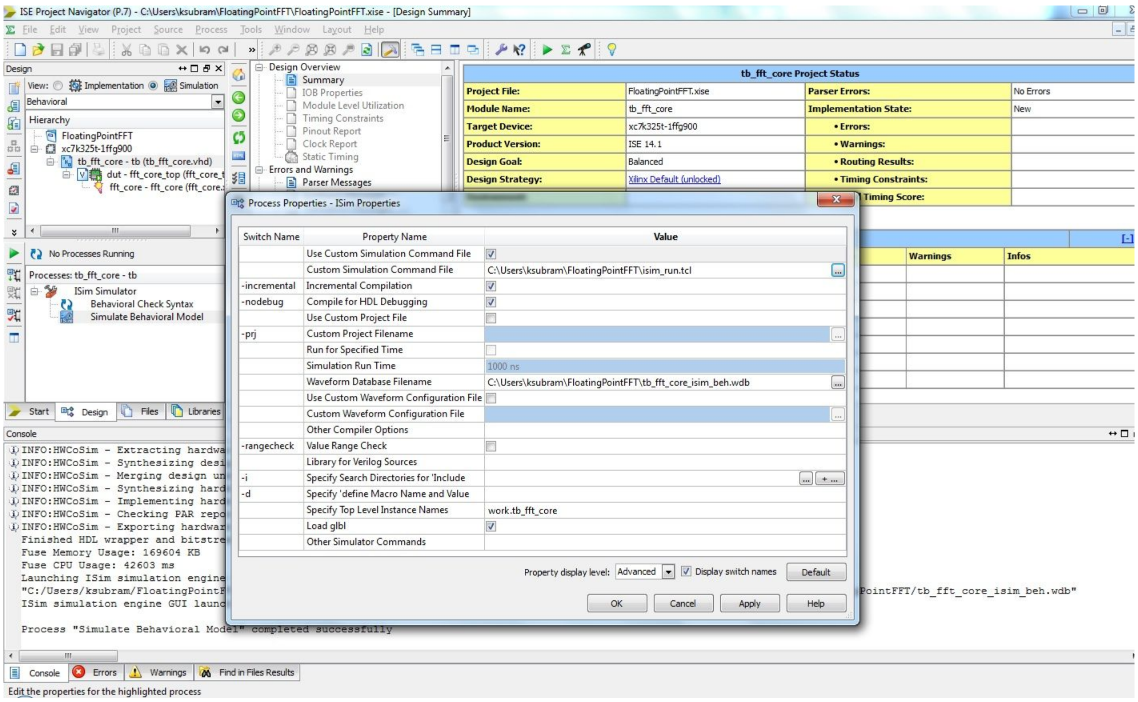


2. In the **Source Properties** dialog box, do the following:
   - Select the **Hardware Co-Simulation** category.
   - Check the **Enable Hardware Co-Simulation** check box.
   - Set the Clock Port to **top_aclk.**
   - Select **KC705 (JTAG)** as the target board.
   - Leave the **Enable Incremental Implementation** option unchecked, and Click **OK**.

Note The enabled instance for hardware co-simulation is now marked with a special icon. 

You can use the **Enable Incremental Implementation** option after the design is compiled for hardware co-simulation. If the instance selected for hardware co-simulation does not change in subsequent runs, you can turn on this option to skip the synthesis, implementation, and bitstream generation for hardware co-simulation. It allows the test bench or any portion simulated in software to be modified and simulated again quickly.

3. From the Hierarchy Pane, select the `fp_fft_tb` instance.
4. In the Process Pane, right-click **Simulate Behavioral Model** and click **Process Properties**.
   a. Check the **Use Custom Simulation Command File** check box.
   b. Browse and add `isim_run.tcl` as the runtime tcl command file.
   c. Disable the **Run for Specific Time** option.
   d. In the **Process Properties > ISim Properties** dialog box, change the **Property display level** to **Advanced**, and click **OK**.
   e. Review the values, and click **OK**.



5. In the Process Pane, run the **Simulate Behavioral Model** process for the `dut - fp_fft_tb` instance.

Compiling the Design on the Command Line

You can invoke the ISim compiler through the `Fuse` command line tool. You must provide a project file, the design top level module(s), and other optional arguments such as libraries and library search path to which to link. To compile the design for hardware co-simulation, you must also provide the following arguments:

```
fuse -prj <project file> <top-level modules>
     -hwcosim_instance <instance>
     -hwcosim_clock <clock>
     -hwcosim_board <board>
     -hwcosim_constraints <constraints file>
     -hwcosim_incremental [0|1]
```

- `-hwcosim_instance`: Specifies the full hierarchical path of the instance to co-simulate.
- `-hwcosim_clock`: Specifies the port name of the clock input for the instance.
  - This is the clock in the lock-step portion, that is to be controlled by the test bench.
  - For a design with multiple clocks, specify the fastest clock using this option so that ISim can optimize the simulation. Other clock ports are treated as regular data ports.
- `-hwcosim_board`: Specifies the identifier of the hardware board to use for co-simulation.
- `-hwcosim_constraints` (optional): Specifies the custom constraints file that provides additional constraints for implementing the instance for hardware co-simulation. We also use the constraints file to specify which ports of the instance are mapped to external I/Os or clocks.
- `-hwcosim_incremental` (optional): Specifies whether Fuse should reuse the last generated hardware co-simulation bitstream and skip the implementation flow.

For example, to compile the FFT design for this tutorial, you can run the `Fuse` command line as follows:

```
fuse -prj fp_fft_tb.prj fp_fft_tb
     –o fp_fft_tb.exe
     -hwcosim_instance /fp_fft_tb/dut
     -hwcosim_clock top_aclk
     -hwcosim_board kc705-jtag
```

Step 4: Running ISim Hardware Co-Simulation

The simulation executable generated by the compiler runs in the same way in both software simulation and hardware co-simulation flow. Project Navigator automatically launches the simulation executable in GUI mode after the compilation finishes.

In the Instances and Processes Pane, the instance selected for hardware co-simulation is indicated with a special icon. As the instance runs in hardware, you cannot expand it to see its internal signals and sub-modules.

Before the simulation starts, ISim programs the FPGA with the bitstream file generated for hardware co-simulation.

Notice the message in the ISim console window:

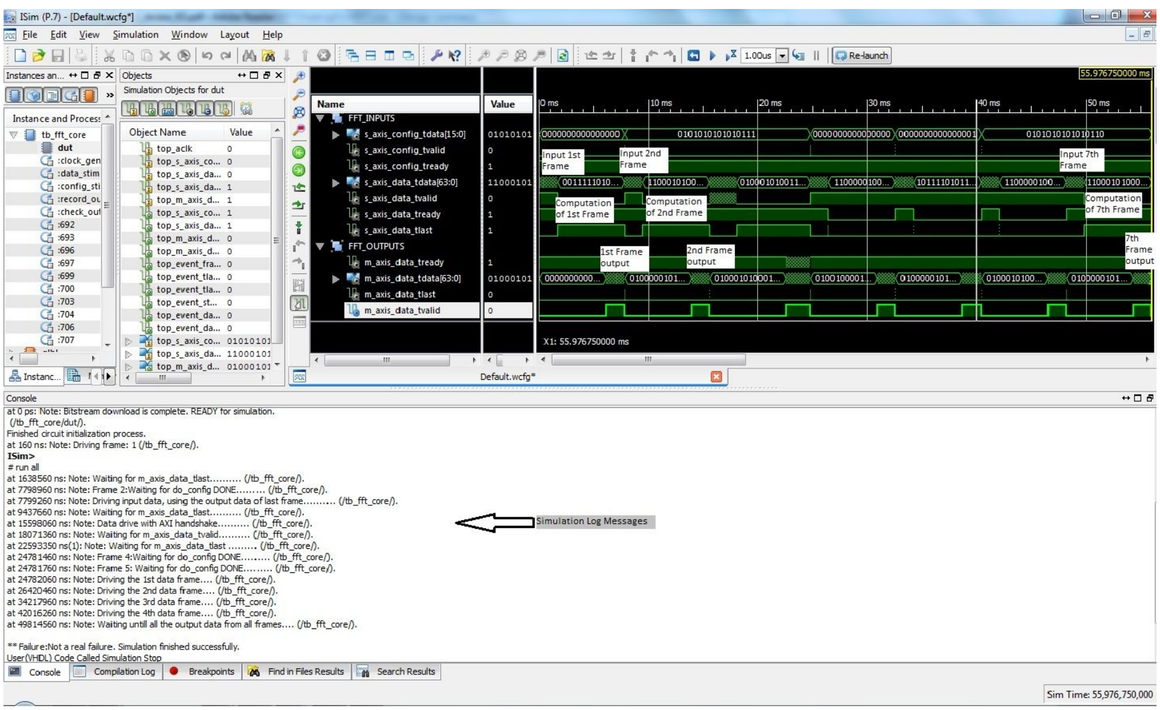
`“Downloading bitstream, please wait till status is READY”.`

After the FPGA is configured, the console shows:

`“Bitstream download is complete.  READY for simulation.”`

Note If your computer has more than one Ethernet port, you must identify the one to use to the simulation process. See Determining the Ethernet Port, for more information.

From this point, you can run the simulation and interact with the ISim GUI the same way you do in the software simulation flow.



Appendix A

Additional Resources

- **Xilinx Glossary -** http://www.xilinx.com/support/documentation/sw_manuals/glossary.htm
- **Xilinx Documentation -** http://www.xilinx.com/support/documentation
- **Xilinx Support -**http://www.xilinx.com/support
- *7 Series User Guides*
- *ISim User Guide (UG660)*

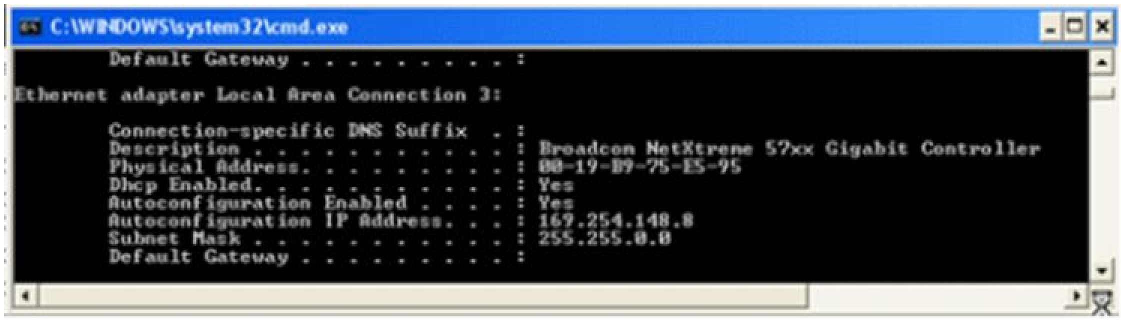
Determining the Ethernet Port

To run an Ethernet-based Hardware Co-Simulation (HWCoSim) when multiple Ethernet interfaces are present, you must select the Ethernet interface that you want to co-simulate.

If you ran a previous hardware co-simulation using the Point-to-Point interface option, you see the following error message:

```
"ERROR: In process wrapper AHIL_INITIALIZE
Failed to open hardware co-simulation instance.
Error in Point-to-point Ethernet Hardware Co-simulation.
There are multiple Ethernet interfaces available.
Please select an interface."
```

Use the following steps to determine the Ethernet port, set and verify the Ethernet address, and verify that the simulation runs. Refer to the following figure for Step 1.



1. Determine the Ethernet port to which the co-simulation board is connected.
   a. On your system command prompt, open a command terminal window (**`cmd`**)
   b. In the command window, type **`ipconfig -all`** to list all Ethernet ports and connections.
   c. Locate the physical address of the Ethernet port connected to the co-simulation board.
   d. Convert the physical address delimiter from a dash (-) to a colon (:). For example: `00:19:B9:75:E5:95`
2. Set and verify the correct Ethernet port in ISim, as follows:
   a. Open the ISim GUI.
   b. Select the Design under Test (DUT).
   c. Go to the Tcl console.
   d. In the Tcl console, enter the following commands:
      i. Set the Ethernet address:

```
hwcosim set ethernetInterfaceID
<##:##:L#:##:L#:##> <physical address>
```

      ii. Verify the Ethernet address:

```
hwcosim get ethernetInterfaceID
```

      iii. Verify that the simulation runs:

```
run 10us
```

The following figure outlines the process within the ISim GUI.