

ISim Hardware Co-Simulation Tutorial: Processing Live Ethernet Traffic Through Virtex-5 Embedded Ethernet MAC

UG819 (v13.3) November 11, 2011



Xilinx is disclosing this user guide, manual, release note, and/or specification (the “Documentation”) to you solely for use in the development of designs to operate with Xilinx hardware devices. You may not reproduce, distribute, republish, download, display, post, or transmit the Documentation in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx. Xilinx expressly disclaims any liability arising out of your use of the Documentation. Xilinx reserves the right, at its sole discretion, to change the Documentation without notice at any time. Xilinx assumes no obligation to correct any errors contained in the Documentation, or to advise you of any corrections or updates. Xilinx expressly disclaims any liability in connection with technical support or assistance that may be provided to you in connection with the Information.

THE DOCUMENTATION IS DISCLOSED TO YOU “AS-IS” WITH NO WARRANTY OF ANY KIND. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE DOCUMENTATION, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT OF THIRD-PARTY RIGHTS. IN NO EVENT WILL XILINX BE LIABLE FOR ANY CONSEQUENTIAL, INDIRECT, EXEMPLARY, SPECIAL, OR INCIDENTAL DAMAGES, INCLUDING ANY LOSS OF DATA OR LOST PROFITS, ARISING FROM YOUR USE OF THE DOCUMENTATION.

© Copyright 2002-2012 Xilinx Inc. All Rights Reserved. XILINX, the Xilinx logo, the Brand Window and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners. The PowerPC name and logo are registered trademarks of IBM Corp., and used under license. All other trademarks are the property of their respective owners.

Revision History

The following table shows the revision history for this document.

Date	Version	
03/18/2011	13.1	Initial release
07/06/2011	13.2	Release updates
10/04/2011	13.3	Updated with modifications to match release.

Table of Contents

Tutorial	5
Introduction.....	6
Step 1: Generating a Design in CORE Generator	8
Step 2: Creating a Testbench	21
Step 3: Creating a Custom Constraints File.....	21
Step 4: Compiling the Design for Hardware Co-Simulation	24
Step 5: Running ISim Hardware Co-Simulation	28
Appendix A Additional Resources.....	31
Appendix B Determining the Ethernet Port	33
Determining the Ethernet Port	33

Tutorial

This tutorial is separated into five sections that contain the steps you need to run an Ethernet design through ISim Hardware Co-Simulation (HWCoSim). Perform the steps in the presented order.

This tutorial contains the following sections:

1. Using the CORE Generator™ tool, generate a Virtex®-5 Ethernet MAC example design using the Virtex-5 Embedded Tri-mode Ethernet MAC Wrapper.
2. Binding the Ethernet MAC wrapper to a packet processor by creating a testbench.
3. Creating a custom constraints file to specify which ports on the example design are controlled by ISim and which are mapped to external I/Os.
4. Compiling the testbench for ISim simulation with the example design targeted for hardware co-simulation.
5. Connecting the target FPGA board to your computer, and running the ISim simulation.

Introduction

This tutorial describes how to use ISim Hardware Co-Simulation (HWCosim) to capture live Ethernet traffic through the Virtex®-5 FPGA Embedded Tri-mode Ethernet MAC and process the captured packets from your Hardware Description Language (HDL) testbench at runtime.

When developing an FPGA design that uses Ethernet, it is often challenging to verify the whole design including the Ethernet Machine Access Control (MAC), Multi-Gigabit Transceivers (MGT) (if SGMII is used to interface between the Ethernet MAC and PHY), and Ethernet PHY. Traditionally, one either simulates the whole design in software, or runs the whole design in hardware. A full software simulation approach is useful in two aspects:

- It offers full visibility into the design
- It allows the testbench or design to be changed and reverified in a rapid manner.

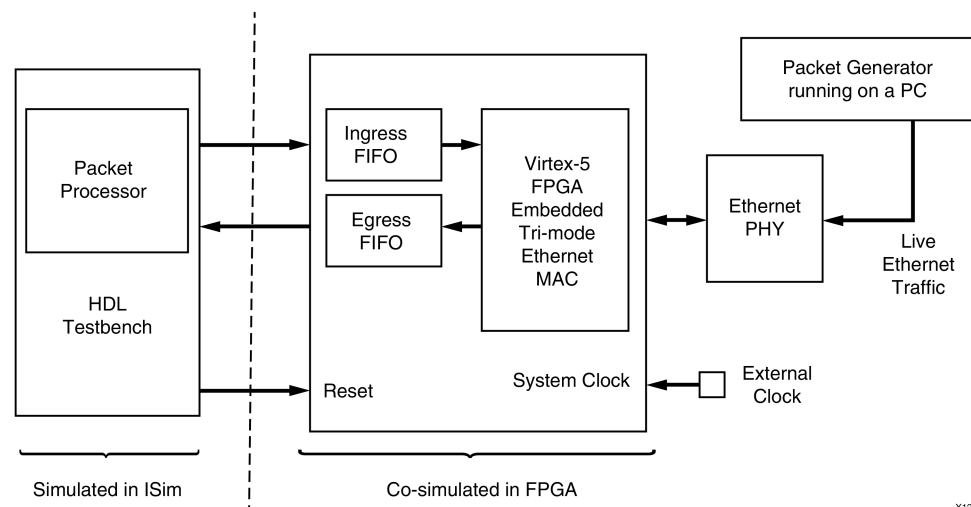
The challenges are setting a testbench that covers the Ethernet MAC, MGT, and PHY, and achieving a reasonable simulation speed.

Often, simulation details of MGT and PHY are bypassed altogether.

Running the design in hardware addresses these problems, but at the cost of reduced visibility into the design, and the complexity of setting up and changing the testbench in hardware.

ISim HWCosim gives you the flexibility to run a portion of your design in hardware while simulating the rest in software. For example, the Ethernet MAC, MGT, and PHY are good candidates to put in hardware so that they are modeled exactly and simulated quickly. The testbench and the application logic, such as the packet processor in your design, which are still under development, need to be simulated in software so you can change, verify, and debug them. The following figure shows how a design for Ethernet packet processing can be partitioned to leverage the ISim Hardware Co-Simulation features.

Partitioning an Ethernet Design for ISim Hardware Co-Simulation



X12262

Prerequisites

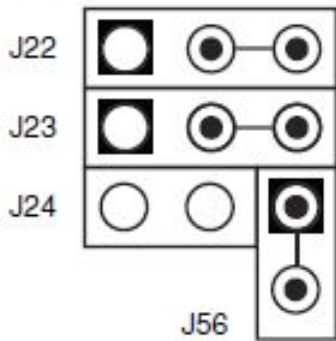
This tutorial requires the following software and hardware:

- ISE® Design Suite
- Virtex-5 ML506 Evaluation Kit
- Design Files: [rdf0127_live_emac_tutorial.zip](http://www.xilinx.com/support/documentation/tutorials/tutorials_rdf0127_live_emac_tutorial.zip)

Note Refer to the [ML506 Evaluation Platform User Guide \(UG347\)](#) to connect your ML506 board to your computer using a JTAG cable. This tutorial uses the SGMII mode to interface with the Ethernet PHY. As shown in the following figure, you change the jumper J22, J23, and J24 on the ML506 board to select SGMII as the default PHY interface.

PHY Interface Mode Jumpers on ML506

SGMII to copper; no clock



- J22: Jumper over pins 2-3
- J23: Jumper over pins 2-3
- J24: No jumper

Tutorial Files

File	Description
full_compile.bat	Windows batch file to fully compile the design for Hardware Co-Simulation (HWCosim) 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 testbench for hardware co-simulation with the Fuse command line.
incr_compile.sh	Linux shell script to incrementally compile the testbench for hardware co-simulation with the Fuse command line.
ipcore_dir	CORE Generator™ directory that contains the Virtex-5 Embedded Tri-mode Ethernet Machine Access Control (MAC) wrapper and FIFO cores.
init.tcl	Custom simulation command file that tells ISim to initialize the simulation.
iseconfig	Directory containing ISE configuration files.
run_isim.bat	Windows batch file to launch the ISim simulation.

File	Description
run_isim.sh	Linux shell script to launch the ISim simulation.
simple_arp.v	A sample packet processor that responds to Address Response Protocol requests.
v5emac_hwcosim.ucf	Custom constraints file for hardware co-simulation that indicates which ports on the v5emac_top module to be mapped to external I/Os and which ports are controlled from the testbench.
v5emac_ml50x.gise	ISE project file.
v5emac_ml50x.xise	ISE project for this tutorial.
v5emac_tb.prj	Hardware co-simulation board support file for PEEP.
v5emac_tb.v	Top-level testbench that instantiates the packet processor.
v5emac_top.v	Wrapper that instantiates the Ethernet MAC core, ingress FIFO, egress FIFO, and packet count FIFO.
v5emac_tb.wcfg	Custom waveform configuration file.
_xmsgs	Directory containing message files.

Note Before starting the tutorial, copy all the data files to your current working directory.

Determining the Ethernet Port

If you have multiple Ethernets, you need to determine the Ethernet port. See [Determining the Ethernet Port](#) for more information.

Tutorial

This tutorial is separated into five sections that contain the steps you need to run an Ethernet design through ISim Hardware Co-Simulation (HWCoSim). Perform the steps in the presented order.

This tutorial contains the following sections:

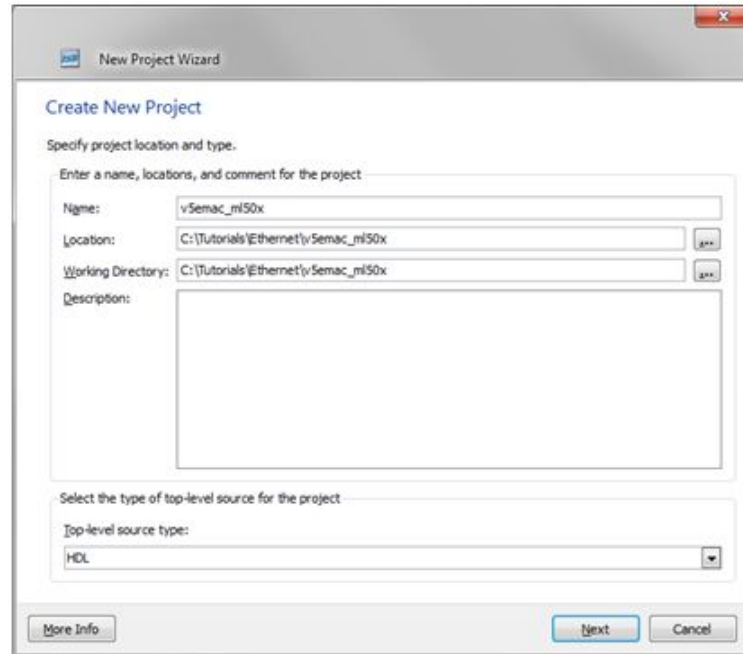
1. Using the CORE Generator™ tool, generate a Virtex®-5 Ethernet MAC example design using the Virtex-5 Embedded Tri-mode Ethernet MAC Wrapper.
2. Binding the Ethernet MAC wrapper to a packet processor by creating a testbench.
3. Creating a custom constraints file to specify which ports on the example design are controlled by ISim and which are mapped to external I/Os.
4. Compiling the testbench for ISim simulation with the example design targeted for hardware co-simulation.
5. Connecting the target FPGA board to your computer, and running the ISim simulation.

Step 1: Generating a Design in CORE Generator

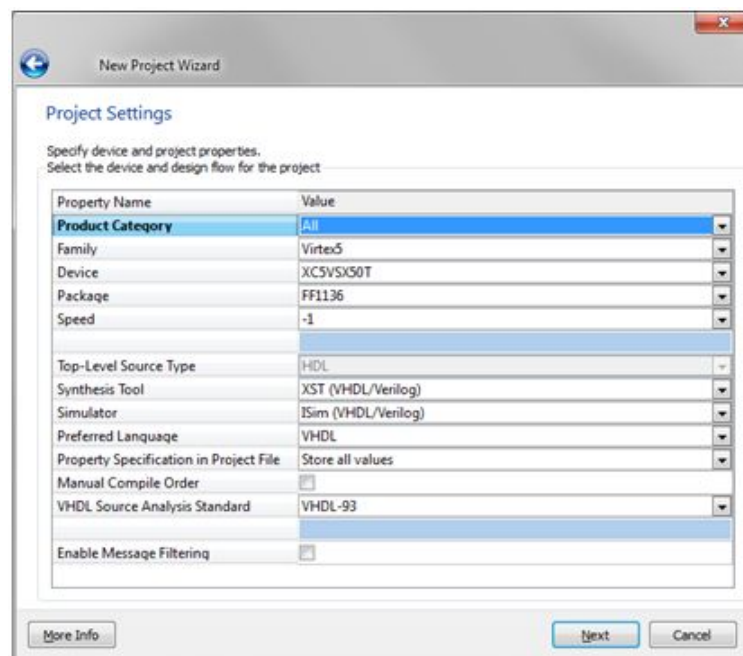
The Virtex®-5 FPGA has an embedded Tri-Mode Ethernet MAC (EMAC) block that provides a simple and reliable way to interface with an Ethernet PHY interface. The Virtex-5 Embedded Tri-Mode Ethernet MAC wrapper in the CORE Generator™ software simplifies the configuration of the Ethernet MAC block. In this tutorial, you use the example design generated by the CORE Generator tool and create an ISim (HWCoSim) testbench that runs on the Virtex-5 FPGA ML506 Evaluation Kit.

Note The figures in this tutorial are based on the Virtex-5 Embedded Tri-Mode Ethernet MAC wrapper version 1.8. The CORE Generator interface might look different in later versions of the tool.

1. Launch ISE® Project Navigator.
2. Select **File > New Project** to open the New Project Wizard. Enter a project name (**v5emac_m150x**) and location, and click **Next**.

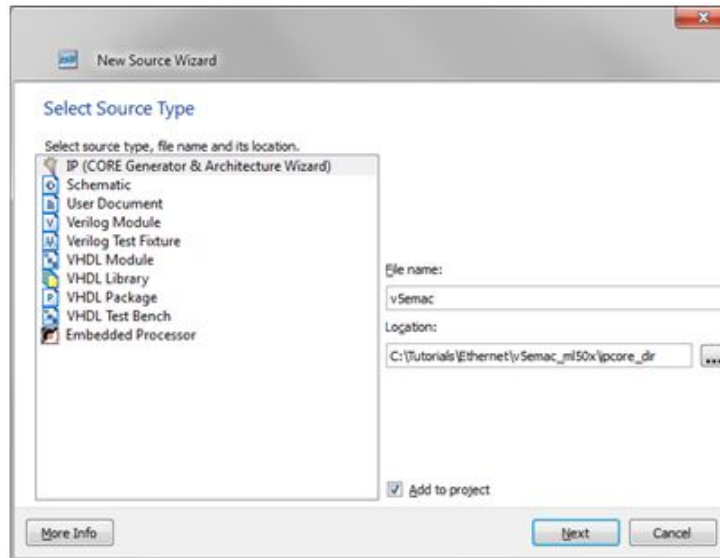


3. On the Project Settings page, select the part for the ML506 board, which is a Virtex-5 device **XC5VSX50T**, package **FF1136**, and speed **-1**. Select **ISim** as the simulator and **VHDL** as the preferred language. Click **Next** and then **Finish** to complete the project creation.



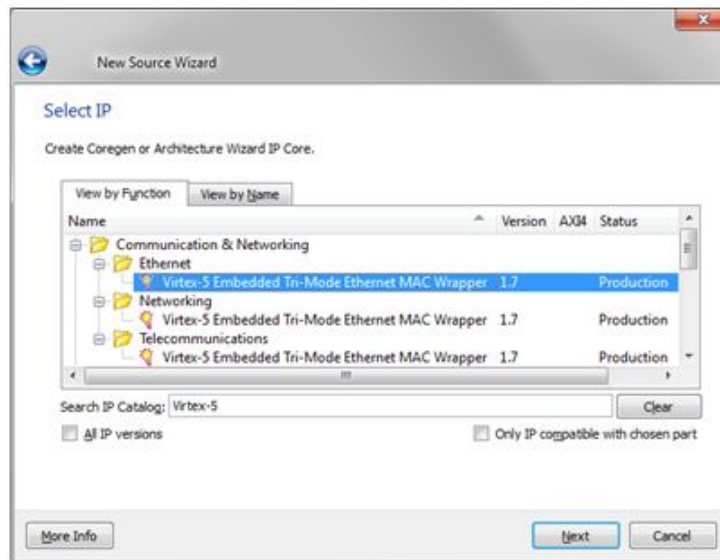
4. Select **Project > New Source** to open the New Source Wizard.

5. Select **IP (CORE Generator & Architecture Wizard)** and name the IP **v5emac**, and click **Next**.

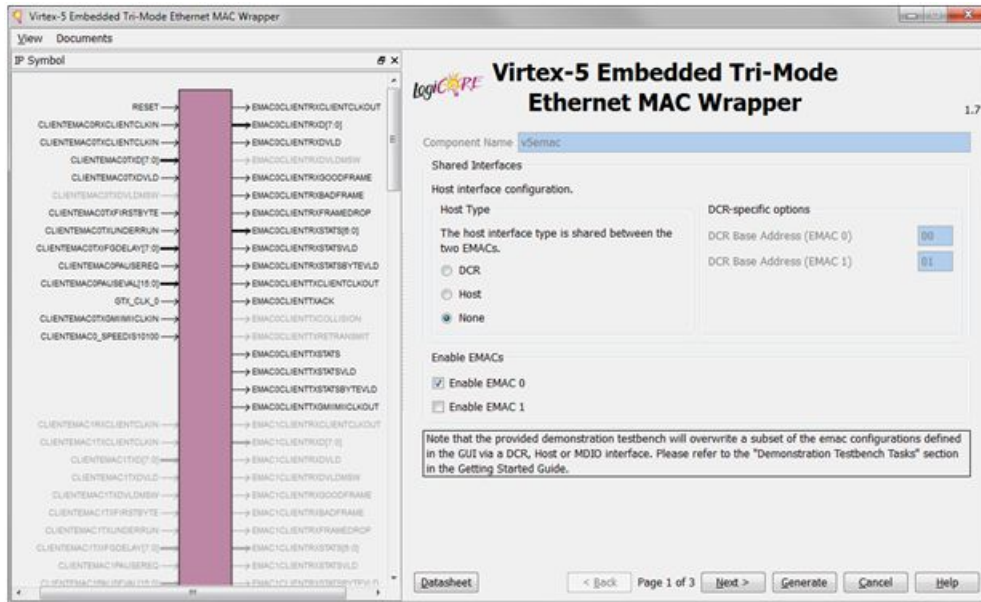


6. Select **Virtex-5 Embedded Tri-Mode Ethernet MAC Wrapper version 1.8** or later from the IP list using one of the selection methods:
 - By Function (Communications & Networking)
 - By Name (Embedded Tri-Mode Ethernet MAC Wrapper version 1.8)

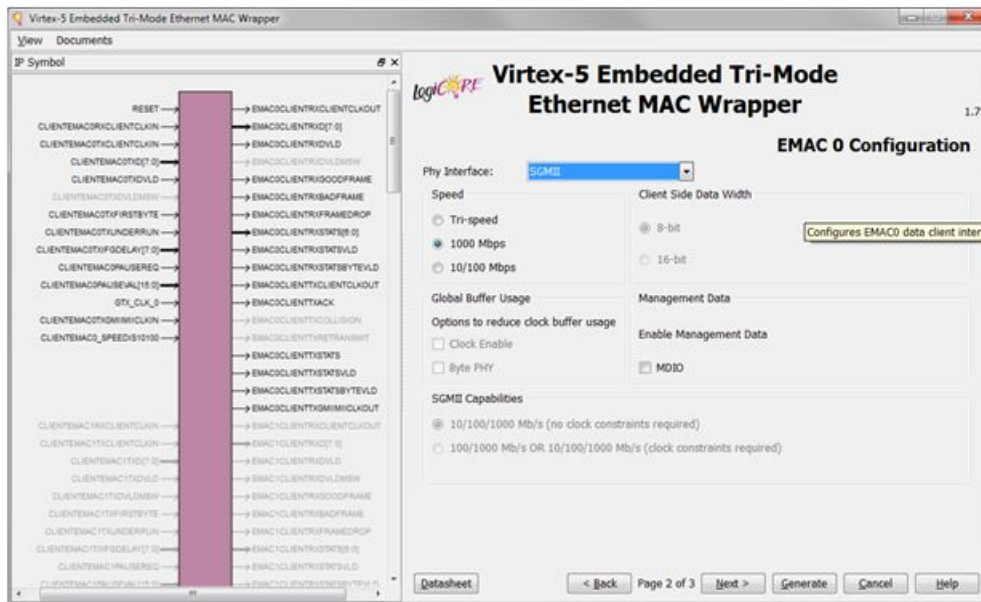
Click **Next** and then **Finish**.



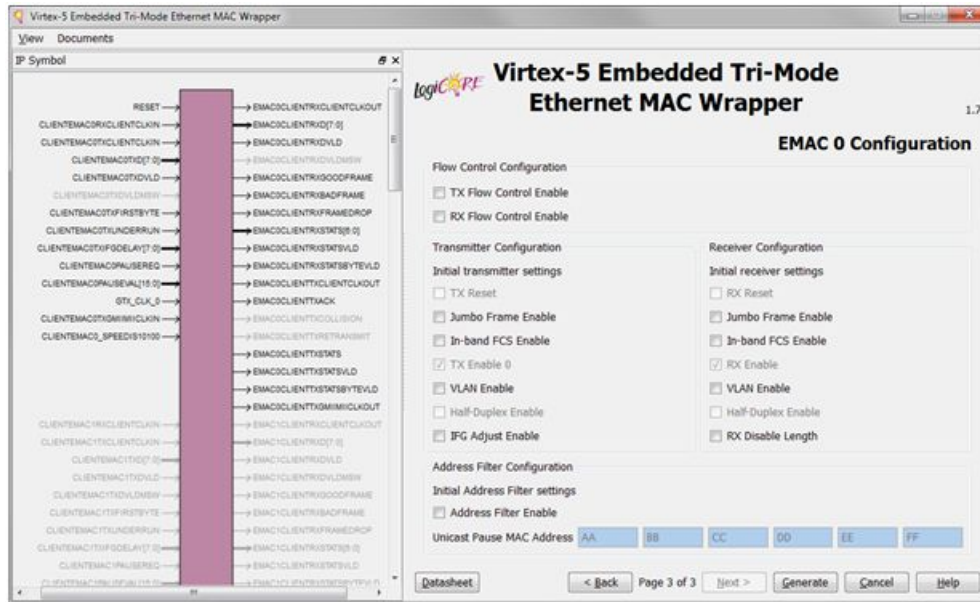
7. When the **Virtex-5 Embedded Tri-Mode Ethernet MAC Wrapper** dialog box opens, set the **Host Type** to **None**. Ensure that **Enable EMAC 0** is checked, and **Enable EMAC 1** is unchecked. Click **Next**.



8. Set **Phy Interface** to **SGMII**, and the **Speed** to **1000 Mbps**. Leave the **MDIO** checkbox unchecked. Click **Next**.



9. Use the default settings for the flow control, transmitter, receiver, and address filter configuration. Click **Generate** to generate the core.



Note After the Ethernet MAC wrapper core is generated, use the LocalLink submodule `v5emac_locallink` in the example design generated in the `ipcore_dir/v5emac/example_design` directory.

In this tutorial, you do not use the entire example design `v5emac_example_design`, because you need to partition the design into two asynchronous halves:

- One simulated in lock-step with the ISim testbench
- The other free-running on FPGA through hardware co-simulation

The packet processing module is simulated in ISim to achieve a full debug visibility and faster turnaround for modifications. The Ethernet MAC and MGT are free-running on FPGA to interface with the external Ethernet PHY chip on the ML506 board. This allows the Ethernet MAC to receive and transmit Ethernet packets on a live Ethernet connection.

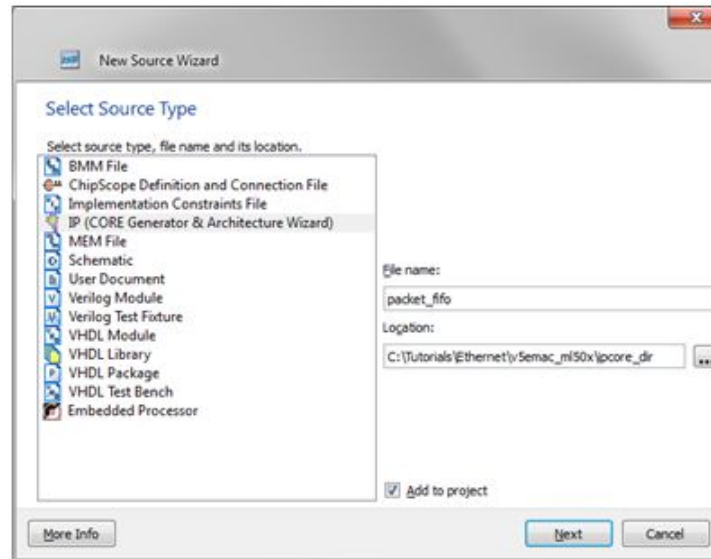
10. Choose **Project > Add Source**. Go to the `ipcore_dir/v5emac/example_design` directory.

Add the following HDL files to the ISE project (all HDL files in the example design except `v5_emac_example_design.vhd`, which is not used in this tutorial):

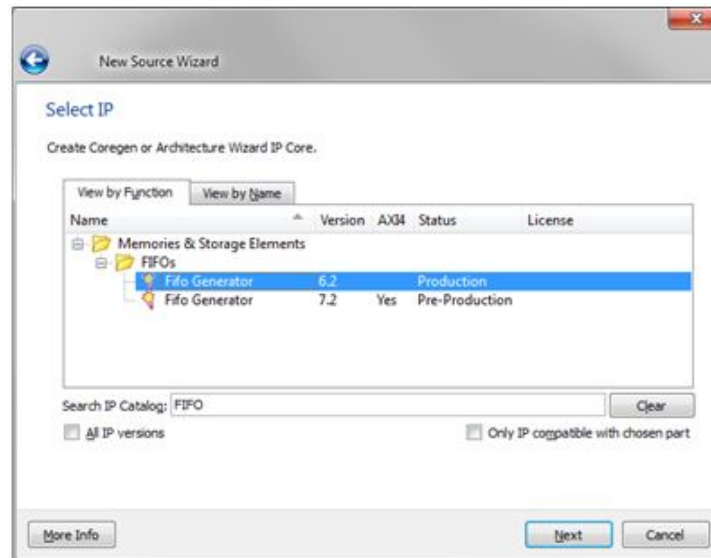
```
v5emac.v
v5emac_block.v
v5emac_locallink.v
client/address_swap_module_8.v
client/fifo/eth_fifo_8.v
client/fifo/rx_client_fifo_8.v
client/fifo/tx_client_fifo_8.v
physical/gtp_dual_1000X.v
physical/rocketio_wrapper_gtp.v
physical/rocketio_wrapper_gtp_tile.v
physical/rx_elastic_buffer.v
```

Typically, you could partition the design across the LocalLink interface where the LocalLink FIFOs serves as asynchronous buffers for crossing clock domains. However, the LocalLink FIFOs in the example design are not large enough and do not work well with the emulated clock generated by ISim. Therefore, you complement the LocalLink FIFOs with a pair of asynchronous FIFOs for buffering ingress and egress packets.

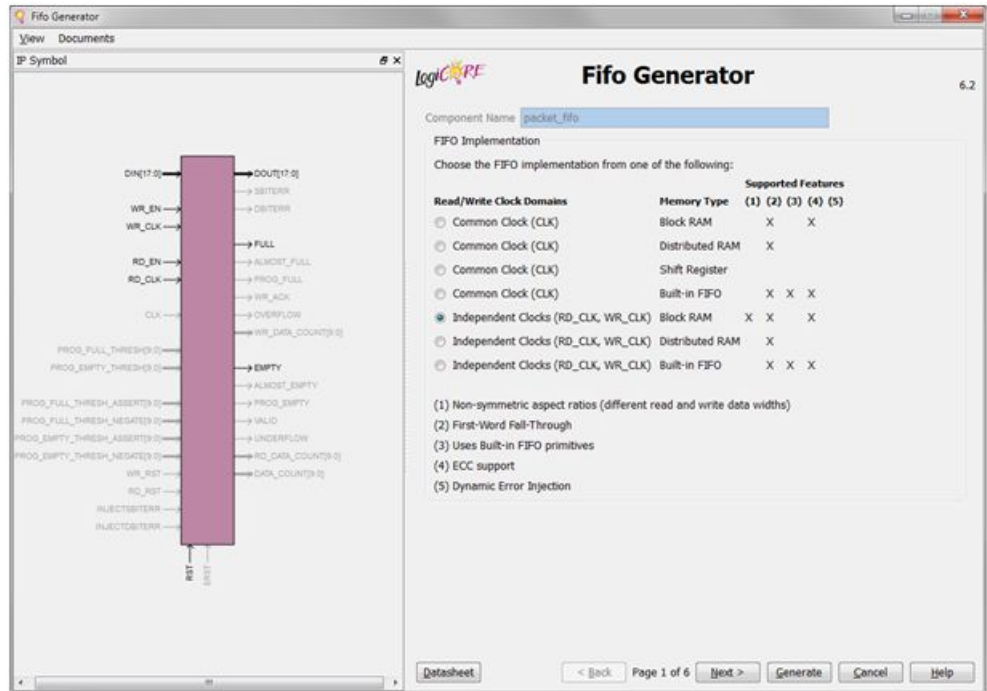
11. Choose **Project > New Source** to open the New Source Wizard. Select **IP (CORE Generator & Architecture Wizard)** and name the IP **packet_fifo**. Click **Next**.



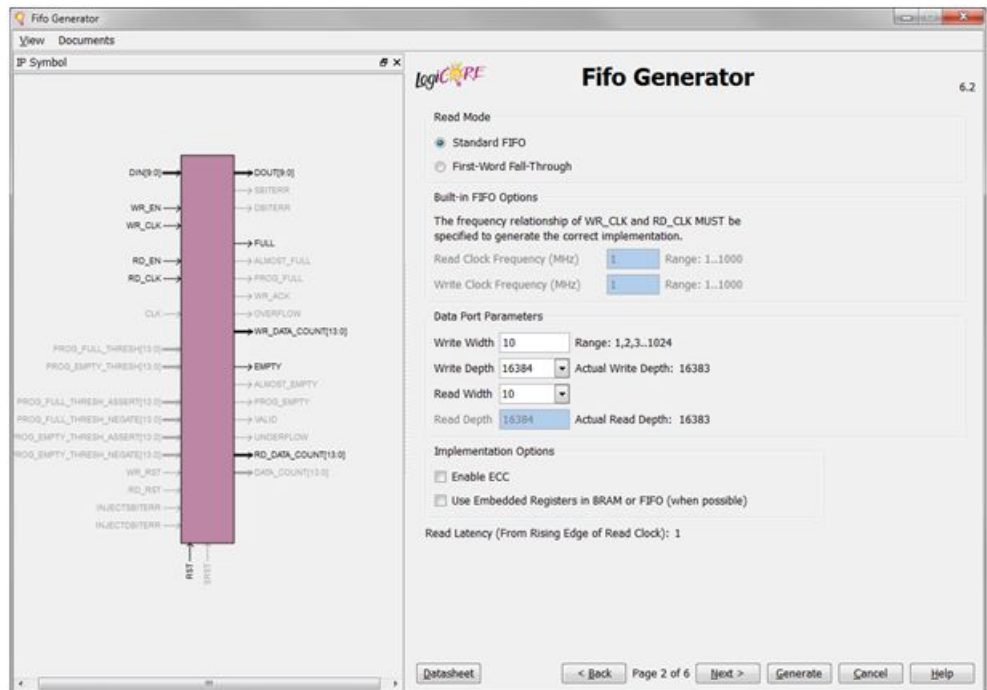
12. Select **Fifo Generator version 8.3** from the IP list. Click **Next** and **Finish**.



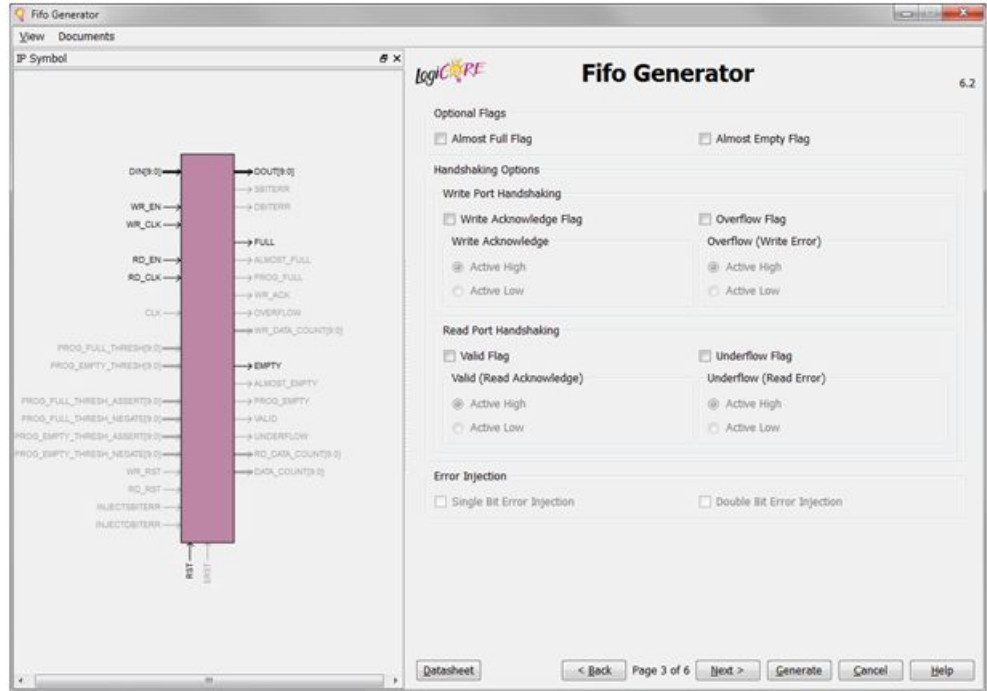
13. When the FIFO Generator core GUI opens, set the FIFO implementation to **Independent Clocks (RD_CLK, WR_CLK) Block RAM**. Click **Next**.



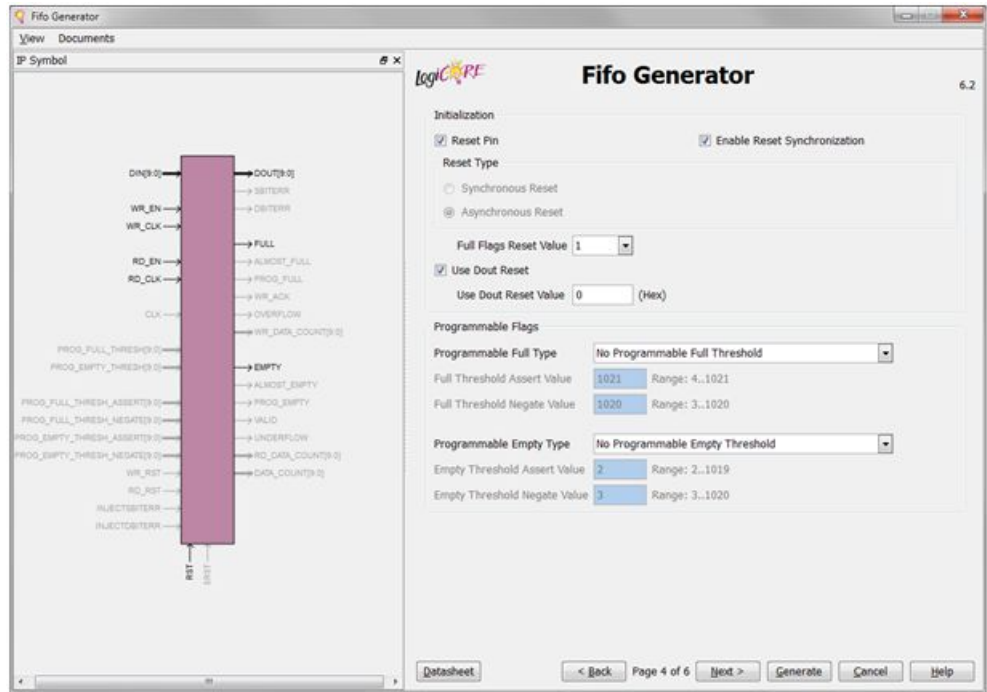
- Set the Read Mode to **First-Word Fall-Through**. Set the **Write Width** and **Read Width** to 10, and **Write Depth** to 16384. Click **Next**.



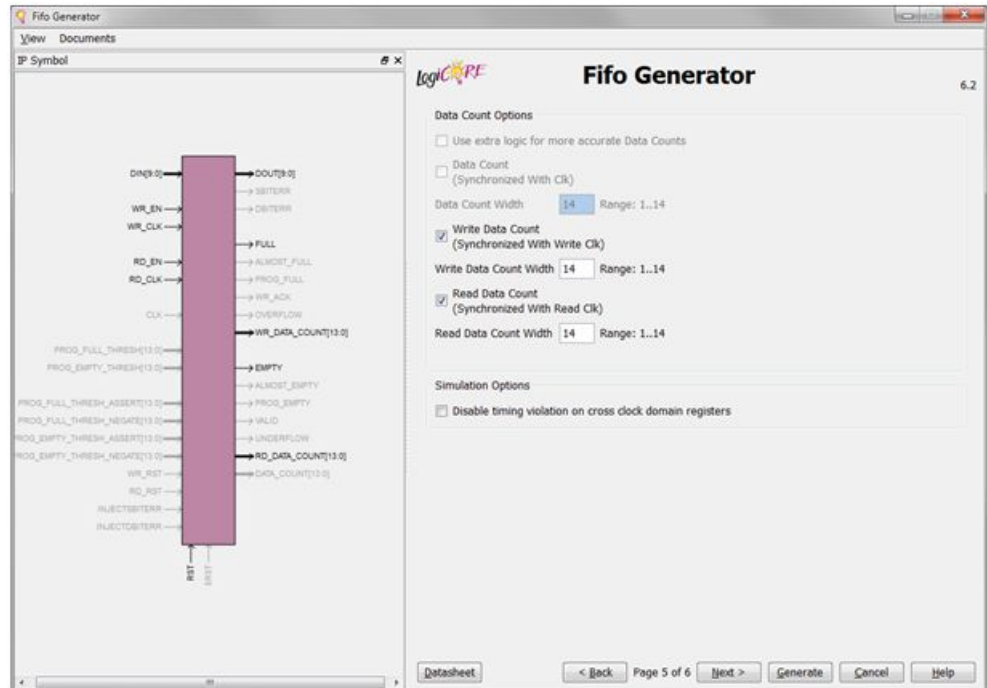
- Use the default settings for **Optional Flags**, **Handshaking Options**, and **Error Injection**. Click **Next**.



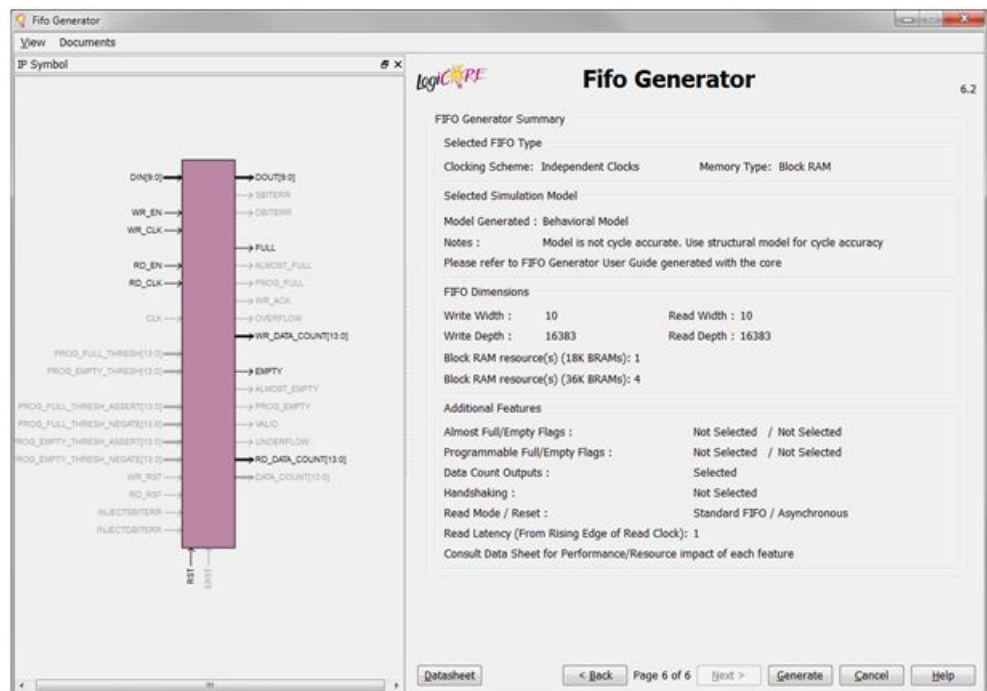
16. Use the default settings for **Initialization and Programmable Flags**. Click **Next**.



17. Set the **Write Data Count** and **Read Data Count** to 5. Click **Next**.

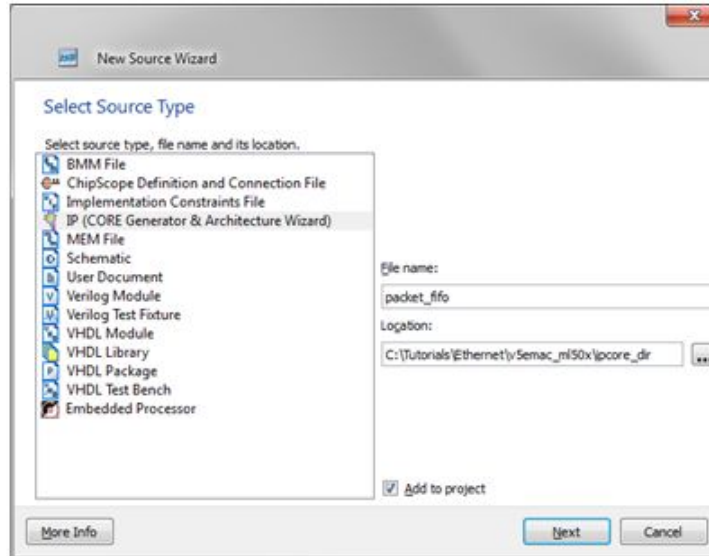


18. Click **Generate** to generate the FIFO core.

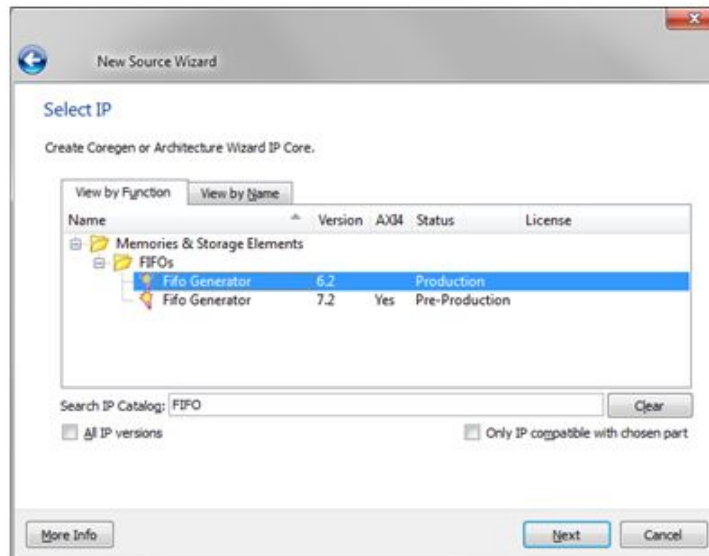


Note In this tutorial, the egress FIFO forwards a packet to the TX LocalLink First In First Out (FIFO) only when there is at least one complete packet in the egress FIFO. For this purpose, add another FIFO to monitor the packet count in the egress FIFO.

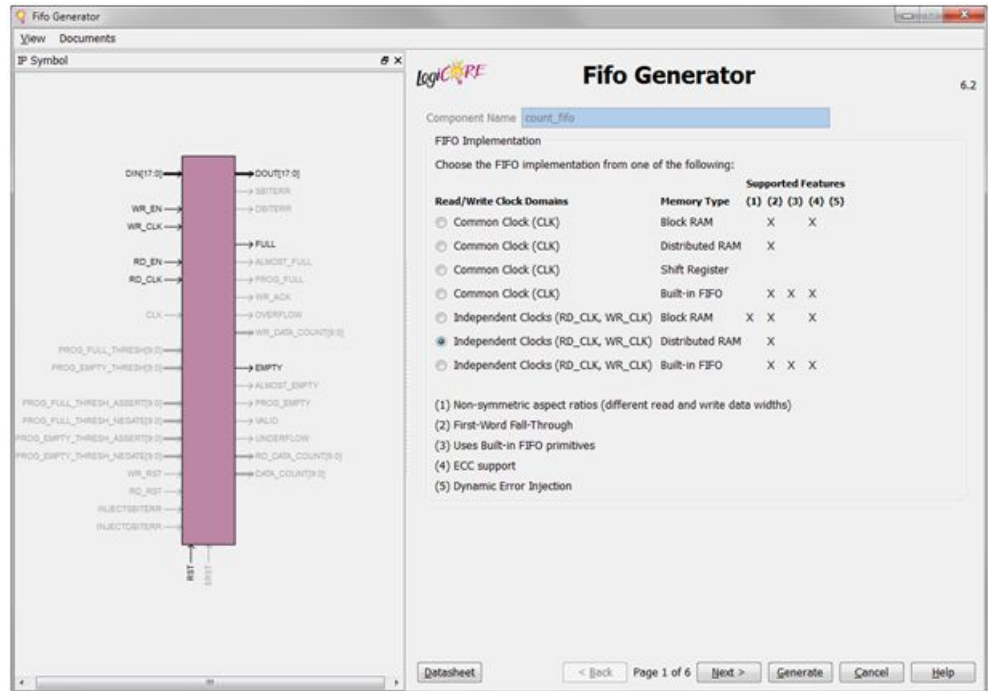
19. Select **Project > New Source** to open the New Source Wizard. Select **IP (CORE Generator & Architecture Wizard)** and name the IP **count_fifo**. Click **Next**.



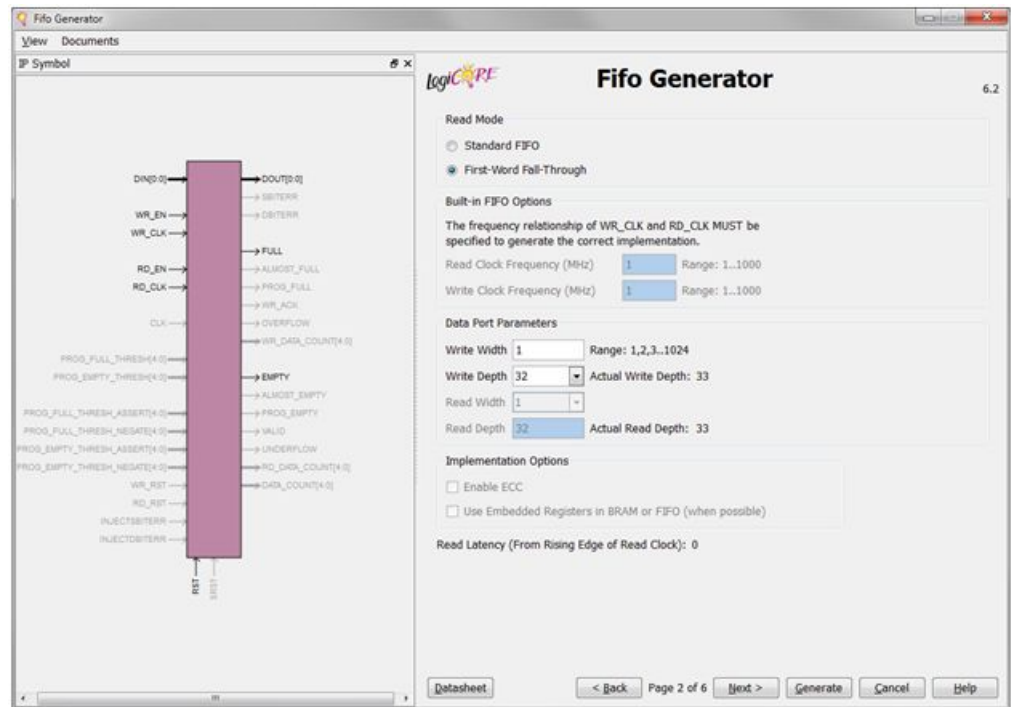
20. Select **Fifo Generator version 8.3** from the IP list. Click **Next** and then **Finish**.



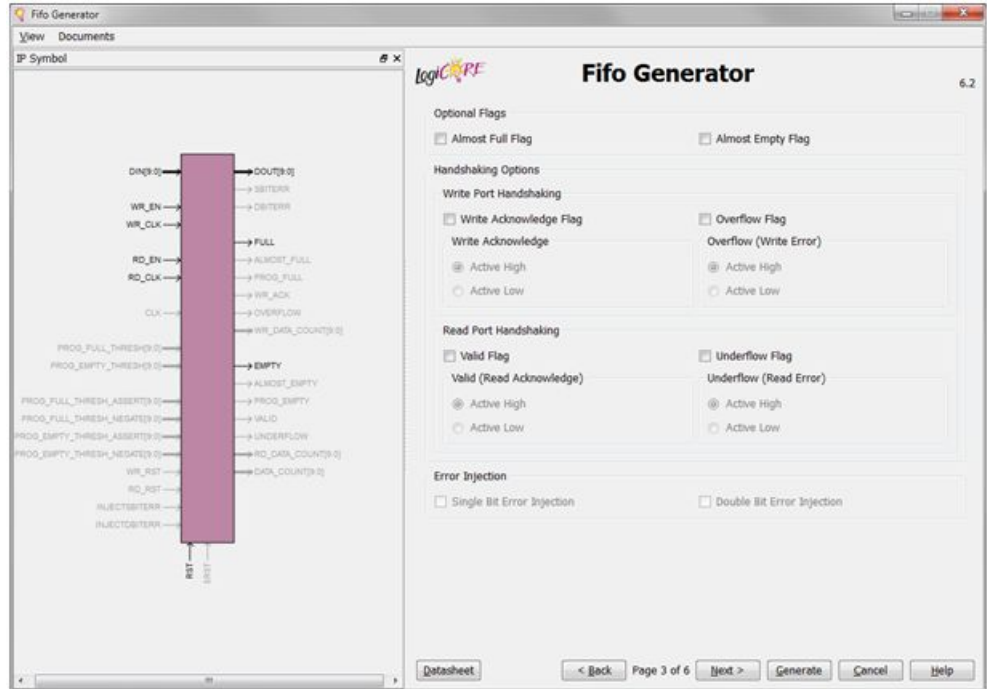
21. When the FIFO Generator GUI opens, set the **FIFO implementation** to **Independent Clocks (RD_CLK, WR_CLK) Distributed RAM**. Click **Next**.



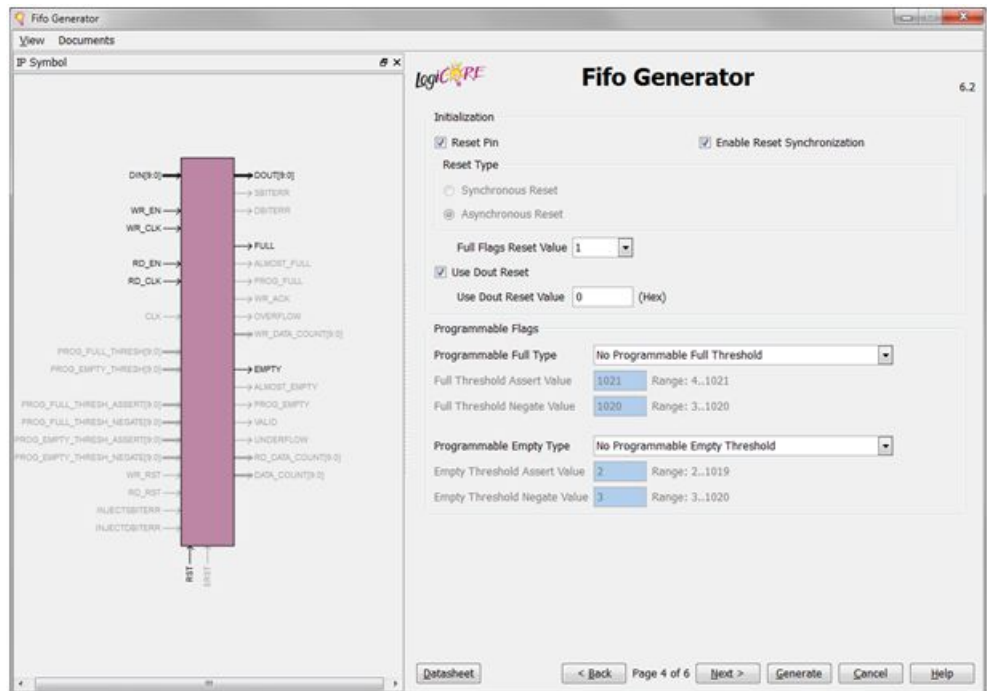
22. Set the Read Mode to First-Word Fall-Through. Set the Write Width and Read Width to 1, and Write Depth to 32. Click Next.



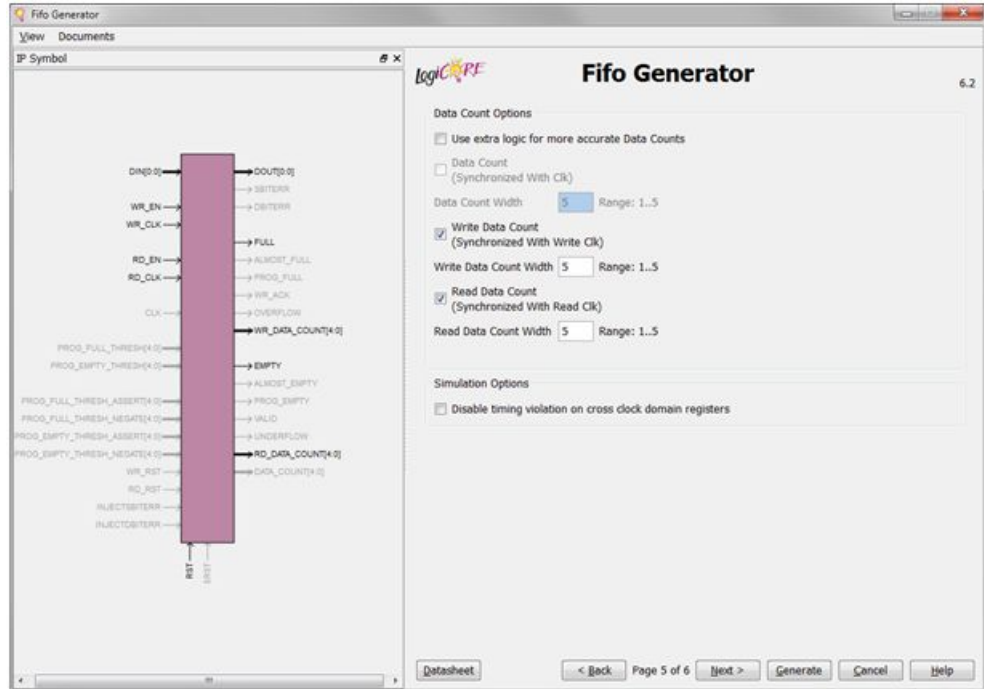
23. Use the default settings for Optional Flags, Handshaking Options, and Error Injection. Click Next.



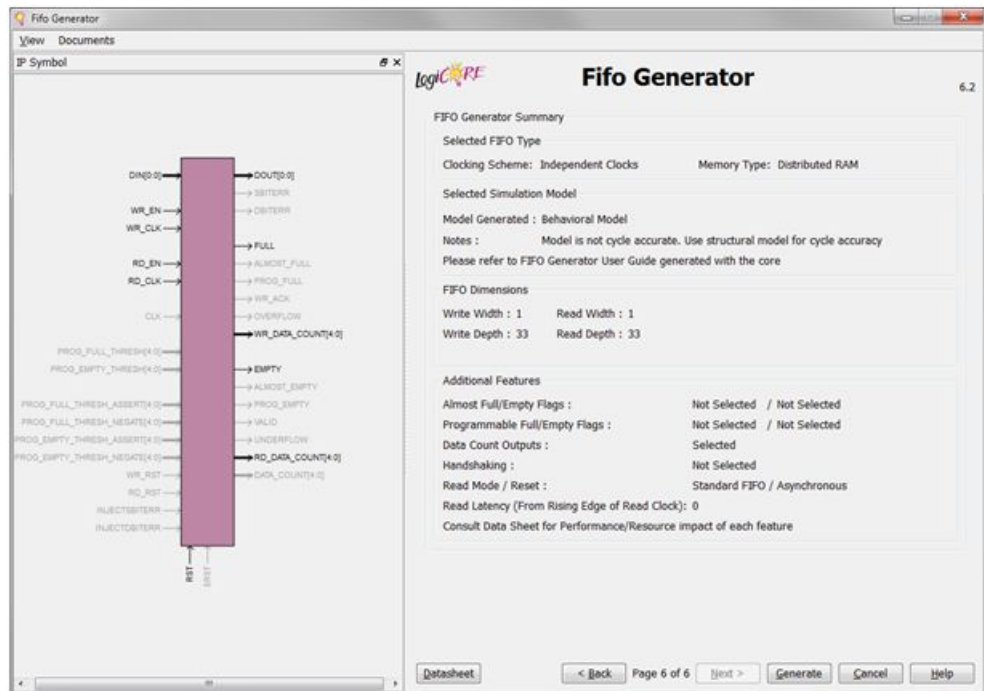
24. Use the default settings for **Initialization and Programmable Flags**. Click **Next**.



25. Ensure that the **Write Data Count** and **Read Data Count** are set to 5. Click **Next**.



26. Click **Generate** to generate the FIFO core.



Next, you Add a top-level module `v5emac_top` that instantiates the `v5_emac_localink` module, ingress FIFO, egress FIFO, and egress packet count FIFO. You can use the completed `v5emac_top.vhd` provided in this tutorial.

27. Select **Project > Add Source**, and add the `v5emac_top.vhd`.

Step 2: Creating a Testbench

Now, you can add a VHDL testbench module `v5emac_tb.v` that binds the `v5emac_top` instance to a packet processor. You can use the completed `v5emac_tb.v` file.

Select **Project > Add Source**. Add the `v5emac_tb.v` and `simple_arp.v` provided in this tutorial.

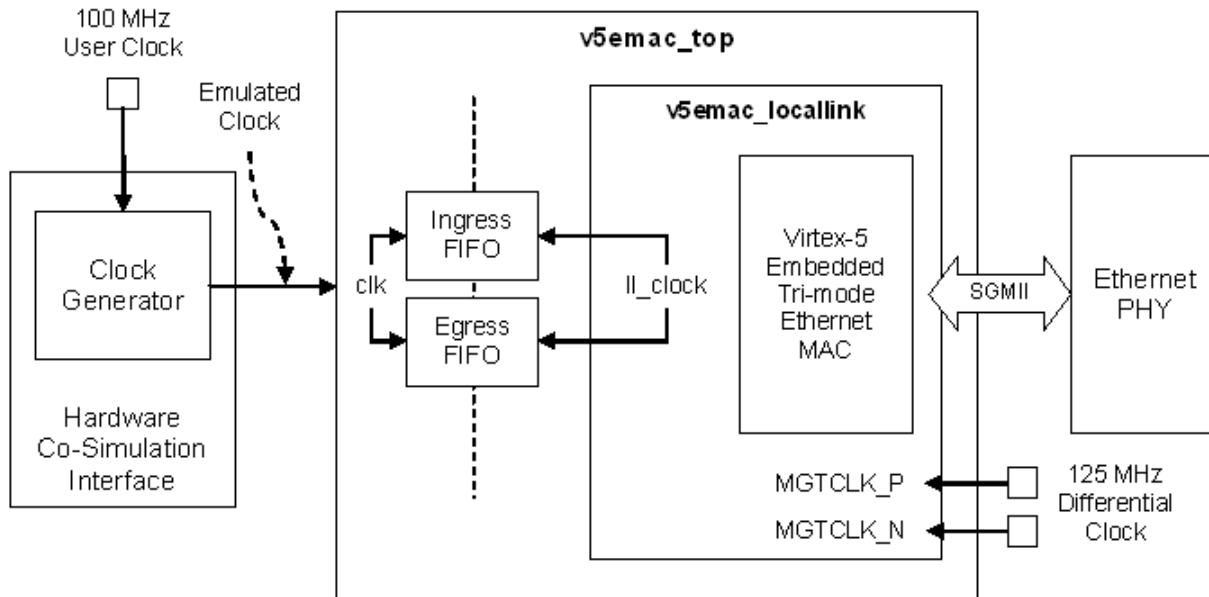
Step 3: Creating a Custom Constraints File

Partitioning the Design into Lock-Step and Free-Running Portions

The key concept of this tutorial is to partition the design into two portions:

- A free-running portion that interfaces with the external Ethernet PHY through the Virtex®-5 Embedded Ethernet MAC. It connects to external I/Os and clocks, and runs at the full clock speed required by the Ethernet interface
- A lock-step portion that is driven by the HDL testbench through ISim. It is synchronized to ISim, and receives stimuli and clock events virtually over the Hardware Co-Simulation (HwCoSim) interface. As a result, it runs at a much lower speed.

The following figure shows how the Ethernet design is clocked under HwCoSim. The HwCoSim interface is inserted automatically during the compilation. It generates an emulated clock based on the 100 MHz user clock on the ML506 board. The emulated clock corresponds to the clock event on the `clk` signal in the testbench, and drives the `clk` port of `v5emac_top` running in hardware. The MGT clock for the Ethernet MAC is received from the 125 MHz differential clock on the ML506 board. The ingress and egress FIFO provide an asynchronous packet buffer for crossing domains between the emulated clock, `clk` and the LocalLink interface clock, `ll_clock`.



Mapping Ports to External I/Os and Clocks

You can provide a custom constraints file, in Xilinx® UCF format, to instruct the ISim compiler about which ports of the instance under Hardware Co-Simulation (HWCoSim) are to be mapped to FPGA IOBs, and which ports are controlled by the HDL testbench. The ISim compiler looks for LOC constraints in the provided UCF file.

- A port with a LOC constraint is mapped to the corresponding FPGA Input Output Buffer (IOB).
- A port without a LOC constraint is mapped to the hardware co-simulation interface and is accessible from the HDL testbench.

The partitioning of a design into a free-running portion and a lock-step portion happens implicitly, based upon how clock ports are mapped.

- When the clock port is mapped to an FPGA IOB using a LOC constraint, the logic driven by this clock belongs to the free-running portion.
- When the clock port has no LOC constraint assigned, the hardware co-simulation interface toggles the value on this port when a corresponding clock event occurs in the testbench. The logic driven by this clock belongs to the lock-step portion.

Because the free-running and lock-step portion run at different speeds with separate clocks, the design should handle clock domain crossing between the two portions. The ISim HWCoSim compilation does not modify the design internals, and the assumption is that the design can handle the speed difference and synchronization between the two portions.

The following table lists the ports on the `v5emac_top` module that are mapped to external I/Os, and those are controlled by the testbench.

Partition of ports on the `v5emac_top` module

Ports Mapped to External I/Os	Ports Controlled by Testbench
CTXP_0 TXN_0 RXP_0 RXN_0 MGTCLK_P MGTCLK_N PHY_RST_N	clk reset resetdone ingress_sof_n ingress_eof_n ingress_data ingress_rd_count ingress_re ingress_empty egress_sof_n egress_eof_n egress_data egress_wr_count egress_we egress_full count_we_o count_wr_count

The example design provided by Virtex-5 Embedded Tri-Mode Ethernet MAC wrapper has a UCF file, `ipcore_dir/v5emac/example_design/v5emac_example_design.ucf`. You use it as a template to create the custom constraints file for HwCoSim.

1. Copy `ipcore_dir/v5emac/example_design/v5emac_example_design.ucf` to the ISE® project directory where `v5_emac_top.v` is located. Name the copied file `v5emac_hwcosim.ucf`.

2. Modify the `v5emac_hwcosim.ucf` file as follows for the ML506 board:

- a. Comment out the `AREA_GROUP` constraints for the embedded Ethernet MAC.

```
#INST v5_emac_11/* AREA_GROUP = AG_v5_emac ;
#AREA_GROUP "AG_v5_emac" RANGE = CLOCKREGION_X1Y2,CLOCKREGION_X1Y3 ;
```

Change the `LOC` constraints of `MGTCLK_P` and `MGTCLK_N` to `P4` and `P3`, respectively, if the default values are different.

```
INST "MGTCLK_N" LOC = "P3" ;
INST "MGTCLK_P" LOC= "P4" ;
```

- b. Change the `LOC` constraint of `GTP` primitive from `GTP_DUAL_X0Y2` to `GTP_DUAL_X0Y3`.

```
INST "*GTP_DUAL_1000X_inst?GTP_1000X?tile0_rocketio_wrapper_i?gtp_
dual_i" LOC = "GTP_DUAL_X0Y3" ;
```

- c. Enable auto-negotiation by default on `EMAC0` of the Ethernet MAC primitive.

```
INST "*?v5_emac" EMAC0_PHYINITAUTONEG_ENABLE = TRUE ;
```

- d. Add the `LOC` constraint for `TXN_0`, `TXP_0`, `RXN_0`, `RXP_0`, and `PHY_RST_N` to match the pin assignments on the ML506 board.

```
INST "TXN_0" LOC = "N2" ;
INST "TXP_0" LOC = "M2" ;
INST "RXN_0" LOC = "P1" ;
INST "RXP_0" LOC = "N1" ;
INST "PHY_RST_N" LOC = "J14" ;
```

3. Next, modify the `v5emac_hwcosim.ucf` file for ISim HwCoSim requirements.

- a. Add a wildcard character (*) at the beginning of the hierarchical path for the following constraints. This is required because the `v5emac_locallink` wraps as a submodule when it is compiled for HwCoSim.

```
NET "*clk125" TNM_NET = "clk_gtp" ;
```

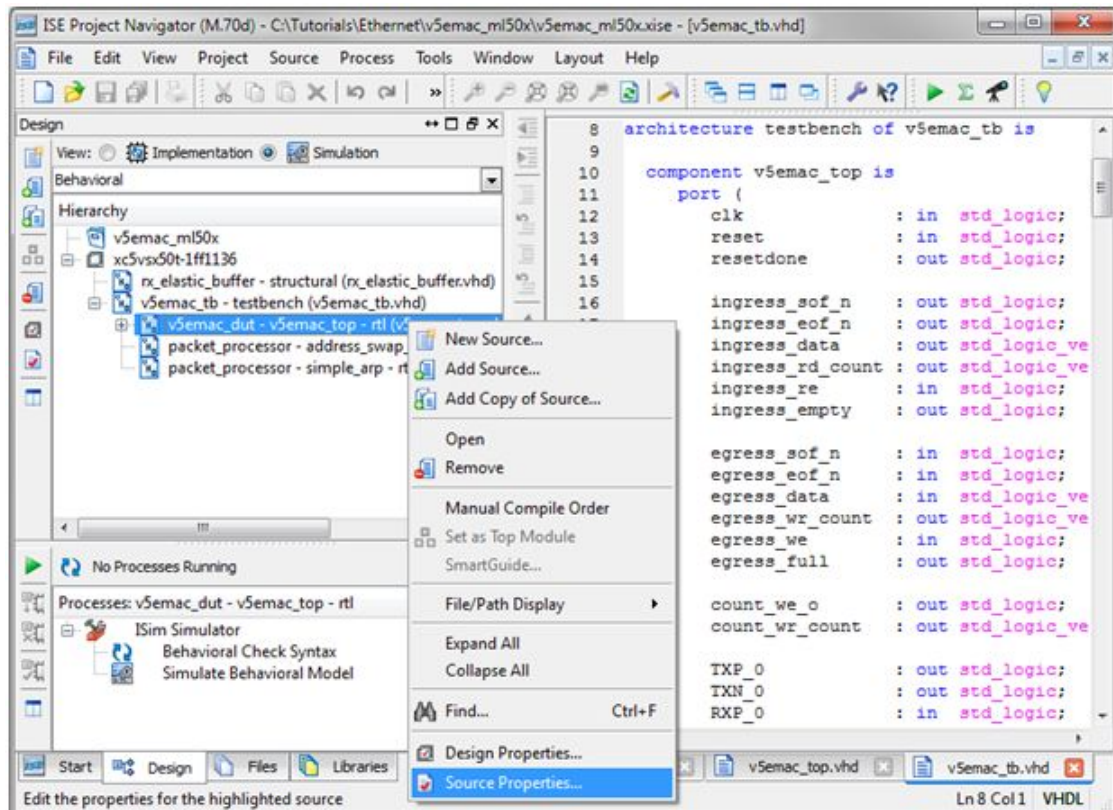
- b. Put a timing ignore constraints (TIG) on the `resetdone` signal to avoid any timing error, because that is monitored by the ISim testbench.

```
NET "*resetdone" TIG ;
```

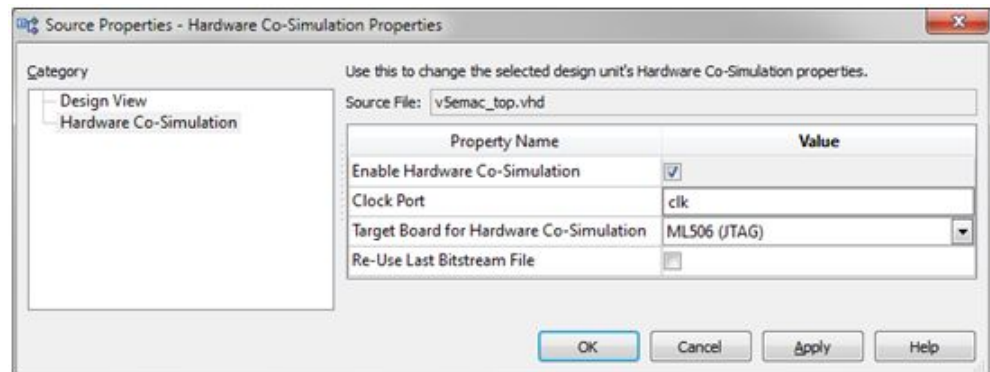
Step 4: Compiling the Design for Hardware Co-Simulation


After you create the testbench and the custom constraints file, you can compile the design for Hardware Co-Simulation (HwCoSim) using the ISim compiler. You do this in Project Navigator by enabling hardware co-simulation on a selected instance in your design. The selected instance, including its submodules, are co-simulated in hardware during the ISim simulation. Other modules are simulated in software.

1. Switch to the **Simulation View** in Project Navigator. Right-click the **v5emac_dut – v5emac_top** instance from the Hierarchy Pane, and click **Source Properties**.



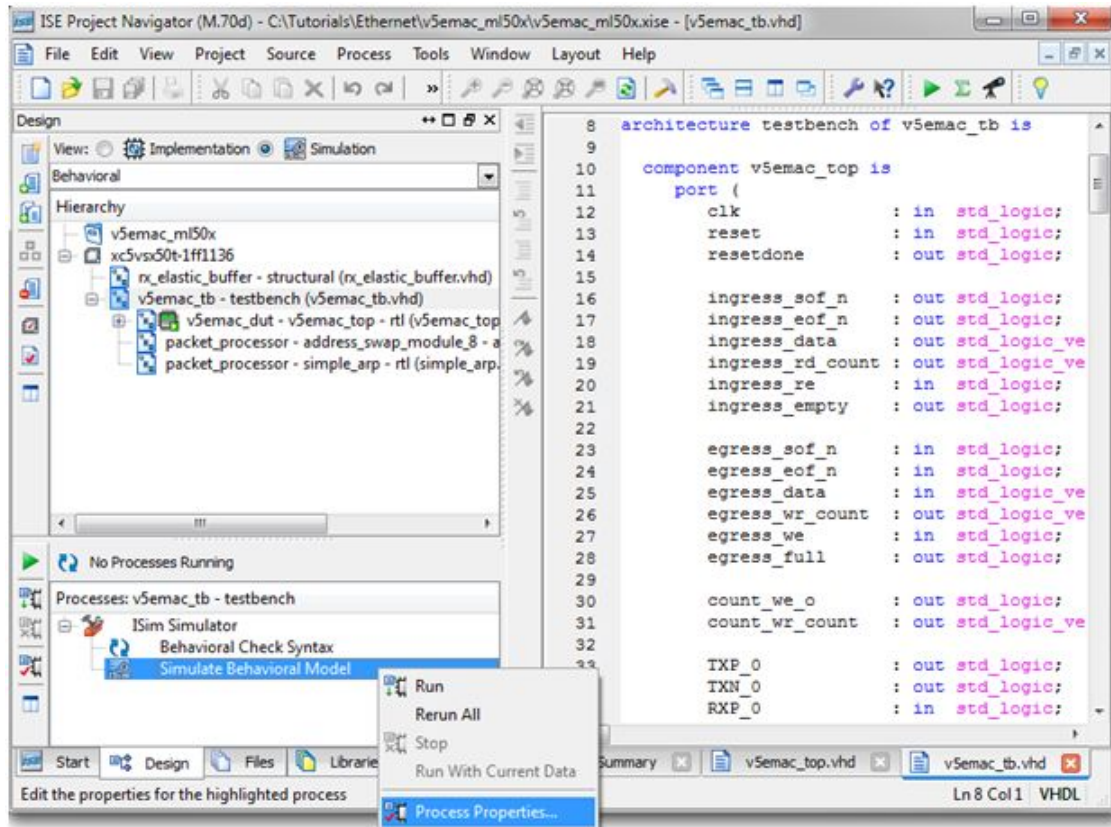
2. Select the **Hardware Co-Simulation** category. Check the **Enable Hardware Co-Simulation** checkbox. Set the **Clock Port** to **clk**. Select **ML506 (JTAG)** as the **Target Board for Hardware Co-Simulation**.



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

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

3. Select the **v5emac_tb** instance from the Hierarchy Pane. In the Processes Pane, right-click **Simulate Behavioral Model**, and click **Process Properties**.



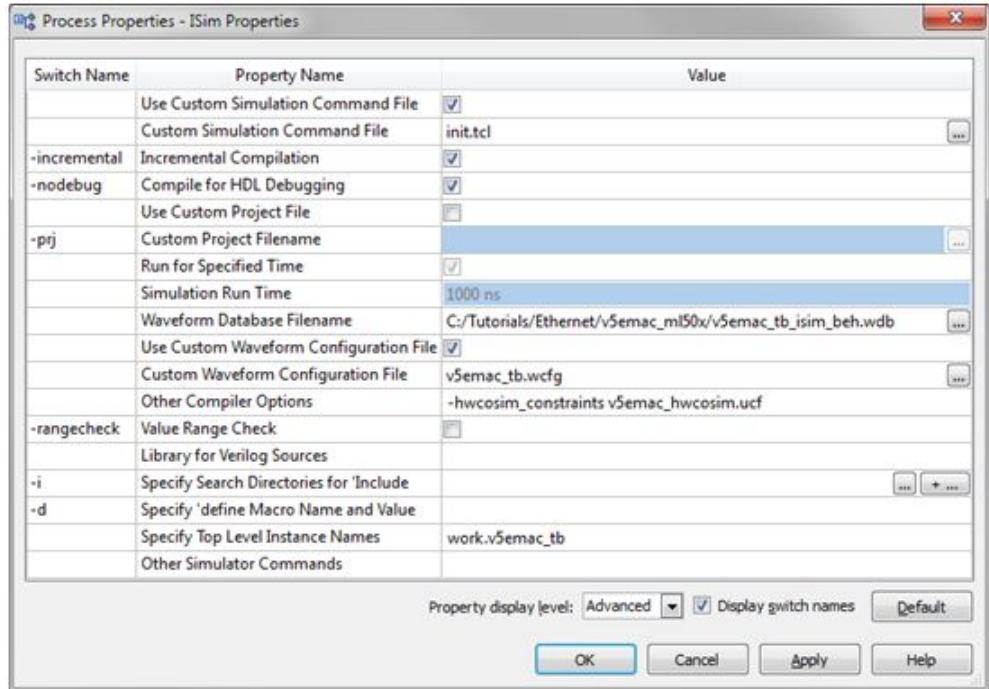
4. Change the **Property** display level to **Advanced**. Set the following properties for the **Simulate Behavioral Model** process:

- Check **Use Custom Simulation Command File**.
- Set **Custom Simulation Command File** to **init.tcl**.
- To set this option, the **Use Custom Waveform Configuration File** needs to be checked.
- Set **Other Compiler Options** to **-hwcosim_constraints v5emac_hwcosim.ucf**.

The `init.tcl` script executes when ISim starts. It runs the simulation for 50ns to perform an initial reset of the design.

The `v5emac_tb.wcfg` file provides a customized waveform configuration view for this tutorial.

Note The custom constraints file for hardware co-simulation is provided to the ISim compiler through the `-hwcosim_constraints` switch. Specify this property through the **Other Compiler Options**.



5. Run the **Simulate Behavioral Model** process for the v5emac_tb instance.

Compiling from the Command Line

You can invoke the ISim compiler through the Fuse command line tool. As in the pure software simulation flow, you need to provide Fuse a project file, the design top-level module(s), and other optional arguments, such as libraries to link in and library search paths. To compile the design for Hardware Co-Simulation (HWCosim), provide the extra arguments listed below:

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


- `-hwcosim_instance` specifies the full hierarchical path of the instance to co-simulate in hardware.
- `-hwcosim_clock` specifies the port name of the clock input for the instance.
 - This is the clock in the lock-step portion, to be controlled by the testbench.
 - 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. The following Virtex®-5 boards are supported by default:
 - `ml501-jtag`: Xilinx® ML501 Evaluation Platform
 - `ml505-jtag`: Xilinx ML505 Evaluation Platform
 - `ml506-jtag`: Xilinx ML506 Evaluation Platform
 - `ml507-jtag`: Xilinx ML507 Evaluation Platform
 - `ml510-jtag`: Xilinx ML510 Evaluation Platform
 - `xupv5-jtag`: Xilinx XUPV5-LX110T Evaluation Platform
- `-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.

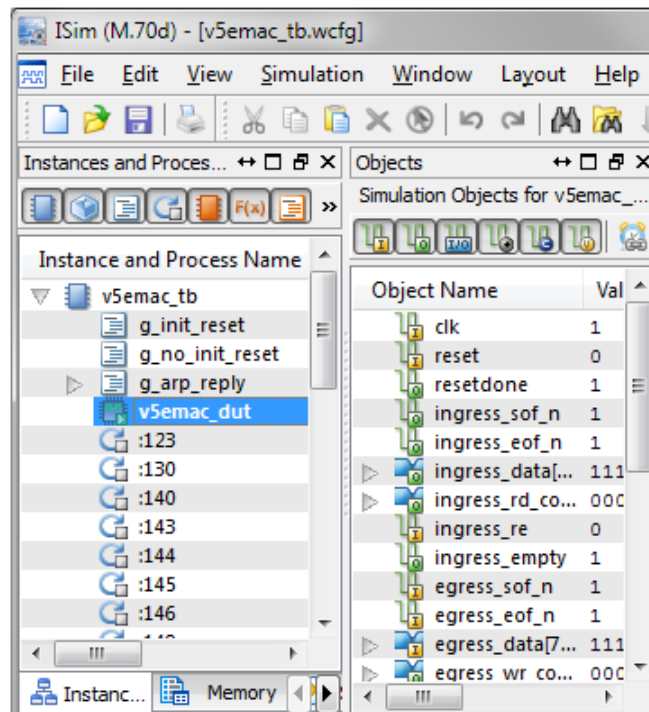
For example, to compile the EMAC design for this tutorial, you can run the Fuse command line as follows:

```
fuse -prj v5emac_tb.prj v5emac_tb
      -o v5emac_tb.exe
      -hwcosim_instance /v5emac_tb/v5emac_dut
      -hwcosim_clock clk
      -hwcosim_board ml506-jtag
      -hwcosim_constraints v5emac_hwcosim.ucf
```

Step 5: Running ISim Hardware Co-Simulation

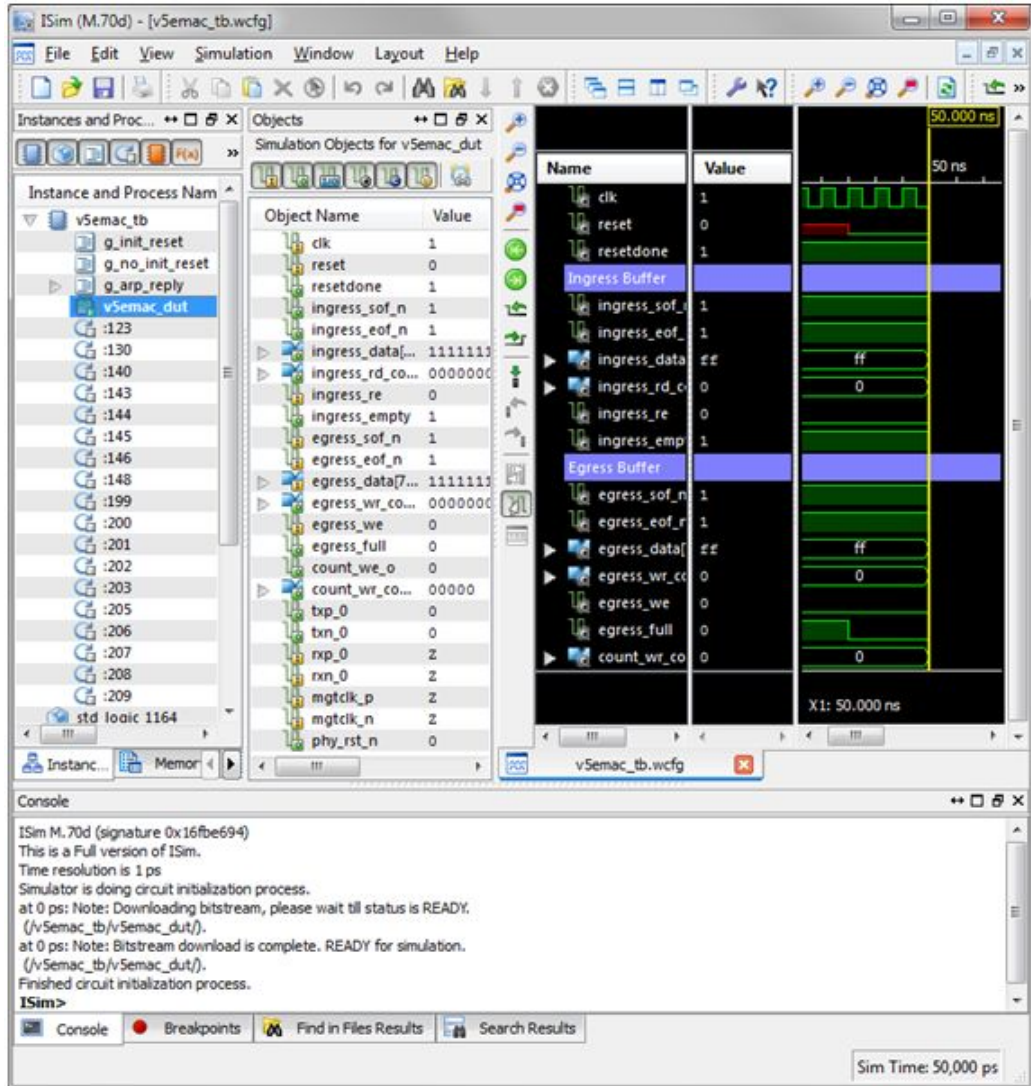
The simulation executable generated by the ISim compiler runs in the same way in both the pure software simulation and the hardware co-simulation flow. Project Navigator automatically launches the simulation executable in GUI mode after the compilation finishes.

In the Instances and Processes view, a special icon  indicates that an instance is selected for hardware co-simulation. As the instance runs in hardware, you cannot expand it to see its internal signals and submodules.

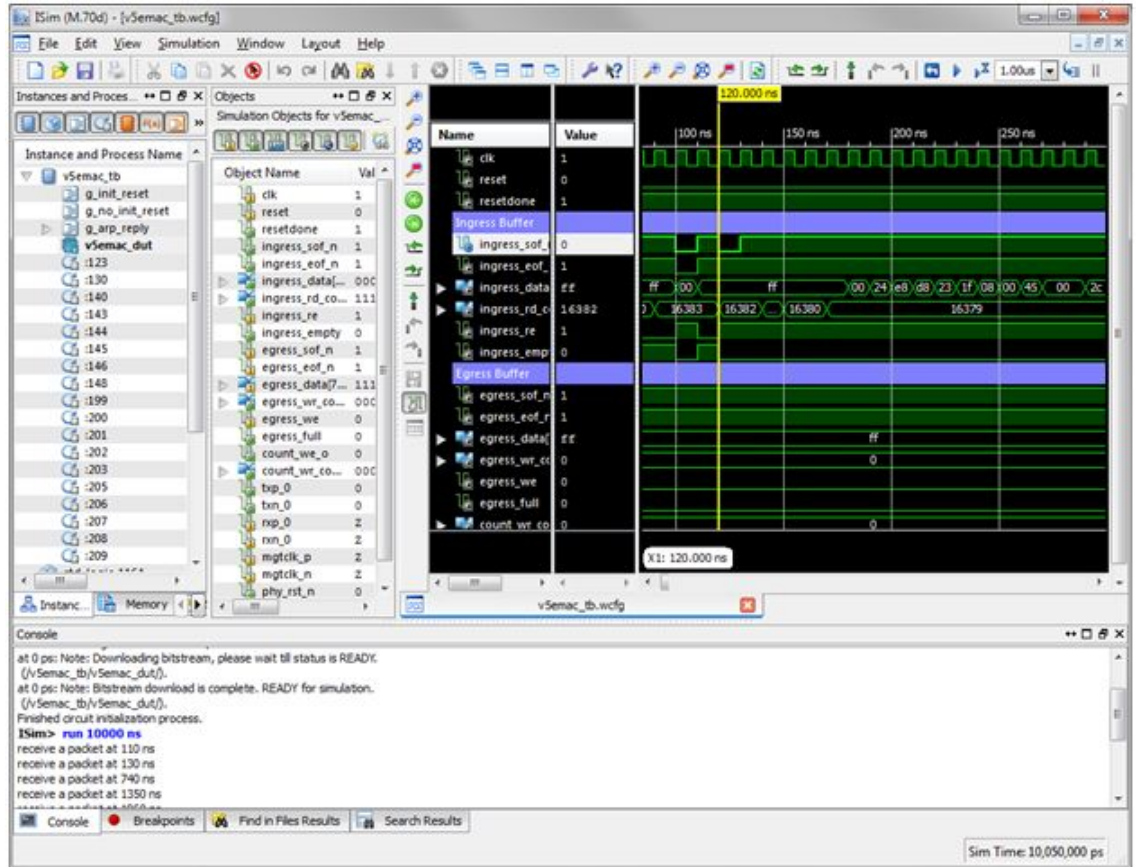


Before the simulation starts, ISim programs the FPGA with the bitstream file generated for hardware co-simulation. The message in the ISim console window reads: `Downloading bitstream, please wait till status is READY. After the FPGA is configured, the console shows Bitstream download is complete. READY for simulation.` From this point, you can run the simulation and interact with the ISim interface the same way you do in the software simulation flow.

The testbench initially resets the system by asserting the reset signal. The `resetdone` signal transits from low to high quickly after the reset is de-asserted. This is because the reset process takes place in hardware at full speed. It could take a much longer time if the same process is simulated in software.



If the Ethernet MAC receives packets, the packets are forwarded from RX LocalLink FIFO to ingress FIFO. When the testbench runs for a few thousand nanoseconds, it starts to read packets out from the ingress FIFO. The ISim console prints out a message like receive a packet at 110 ns when there is a packet read from the ingress FIFO. You can also observe the packet data (ingress_data) from the ISim waveform. If you run the ISim simulation continuously (using the **Run All** command), you can observe the packet stream and how the packet processor processes the packets. As a validity check, you can install a third-party packet sniffer such as Wireshark (<http://www.wireshark.org>) to compare the packets captured by the ISim testbench, and the ones captured by the sniffer.



Now you can modify your packet processor in your testbench and recompile the ISim testbench with the **Enable Incremental Implementation** turned on in the Hardware Co-Simulation (HwCoSim) properties. This substantially speeds up the develop-compile-debug cycle for your HDL design.

Additional Resources

- **Xilinx Glossary** - http://www.xilinx.com/support/documentation/sw_manuals/glossary.pdf
- **Xilinx Documentation** - <http://www.xilinx.com/support/documentation>
- **Xilinx Support** - <http://www.xilinx.com/support/documentation>
- [*Virtex®-5 ML506 Evaluation Platform User Guide \(UG347\)*](#)
- [*Virtex®-6 ML605 Documentation*](#)
- [*Spartan®-6 Boards and Kits*](#)
- [*ISim User Guide \(UG660\)*](#)
- [*ISE Hardware Co-Simulation Tutorials: Accelerating Floating Point FFT Simulation \(UG817\)*](#)
- [*ISE Hardware Co-Simulation Tutorial: Interacting with Spartan-6 Memory Controller and On-Board DDR2 Memory \(UG818\)*](#)
- [*ISE Hardware Co-Simulation Tutorial: Processing Live Ethernet Traffic Through Virtex-5 Embedded Ethernet Mac \(UG819\)*](#)

Determining the Ethernet Port

Determining the Ethernet Port

To run an Ethernet-based Hardware Co-Simulation 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  
<##:##:##:##:##:##> <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.