ML410 BSB Design
Adding the PLB TEMAC with RGMII Using EDK 8.2i SP1

April 2007

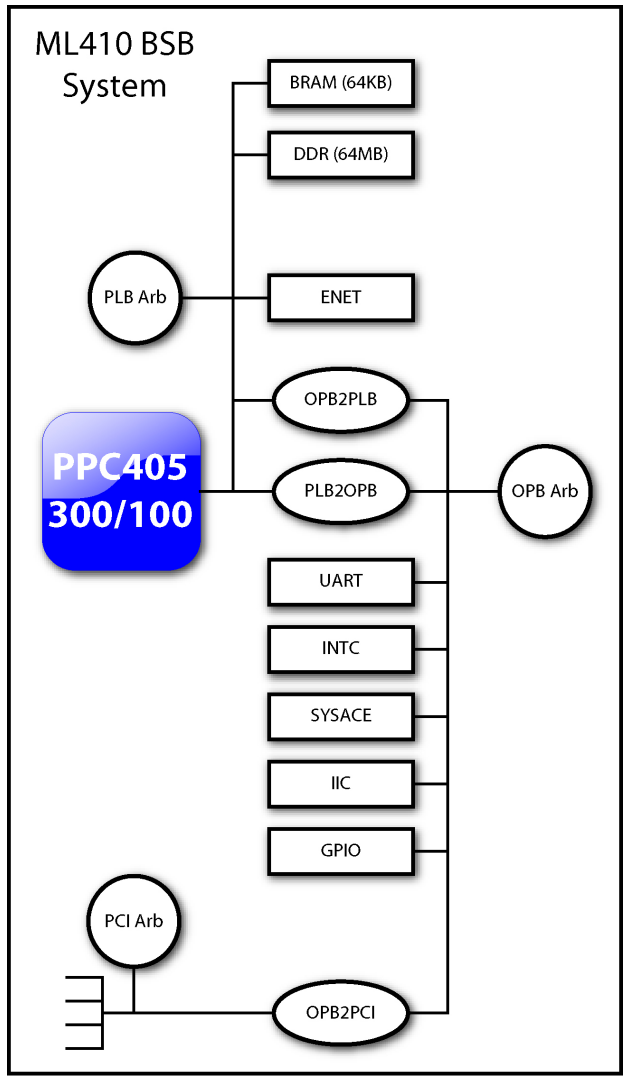
XILINX®

Overview

- Hardware Setup
- Software Requirements
- Generate a Bitstream
- Transfer the Bitstream onto the FPGA
- Loading a Bootloop into the Block RAM

ML410 BSB Hardware

- The ML410 BSB design hardware includes:
  - 64 KB BRAM
  - DDR1 Interface (64 MB)
  - UART
  - Interrupt Controller
  - System ACE Interface
  - IIC
  - GPIO (LEDs and LCD)
  - PLB2OPB Bridge
  - PLB and OPB Arbiters
  - Networking
  - OPB2PLB Bridge
  - OPB2PCI Bridge
  - PCI Arbiter



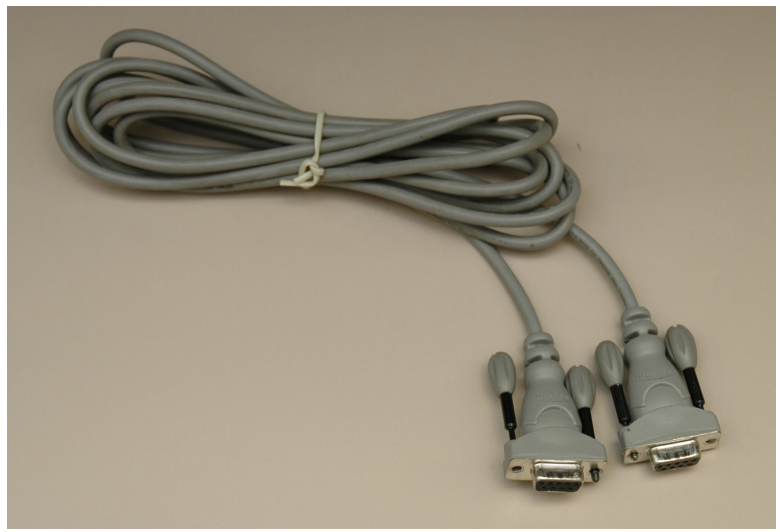
Additional Setup Details

- Refer to ml410_overview_setup.ppt for details on:
  - Software Requirements
  - ML410 Board Setup
    - **Equipment and Cables**
    - **Software**
    - **Network**
  - Terminal Programs
    - **This presentation requires the 9600-8-N-1 Baud terminal setup**

Hardware Setup

- Connect the Xilinx Parallel Cable IV (PC4) to the ML410 board

- Connect the RS232 cable to the ML410 board



ISE Software Requirement

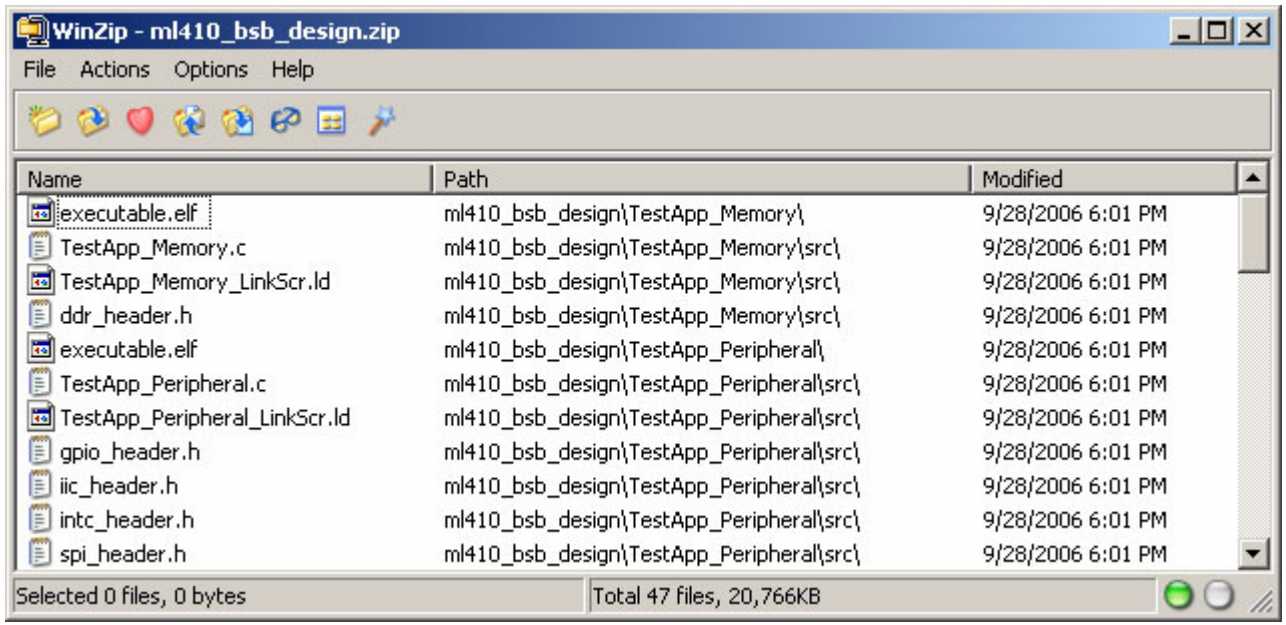
- Xilinx ISE 8.2i SP2 software

EDK Software Requirement

- Xilinx EDK 8.2i SP1 software

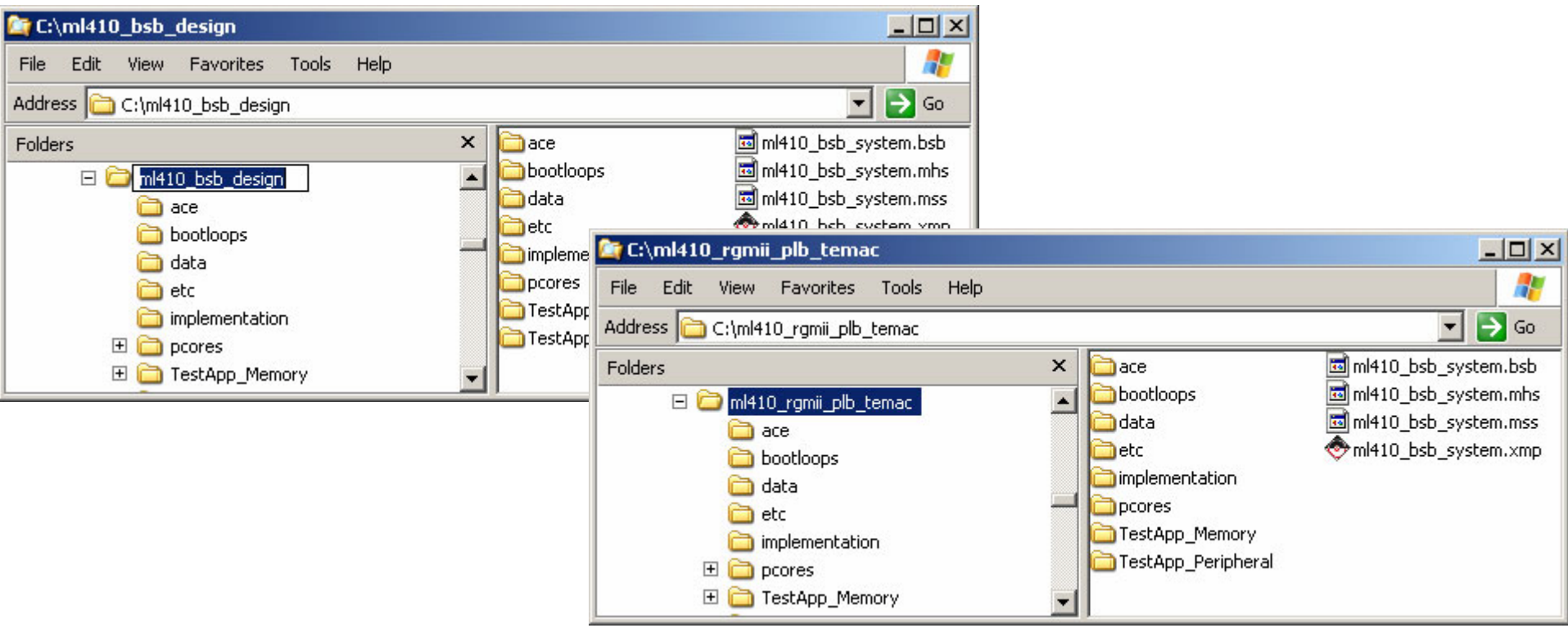
Extracting the Design

- Unzip the ml410_bsb_design.zip file
  - This creates ISE and EDK project directories



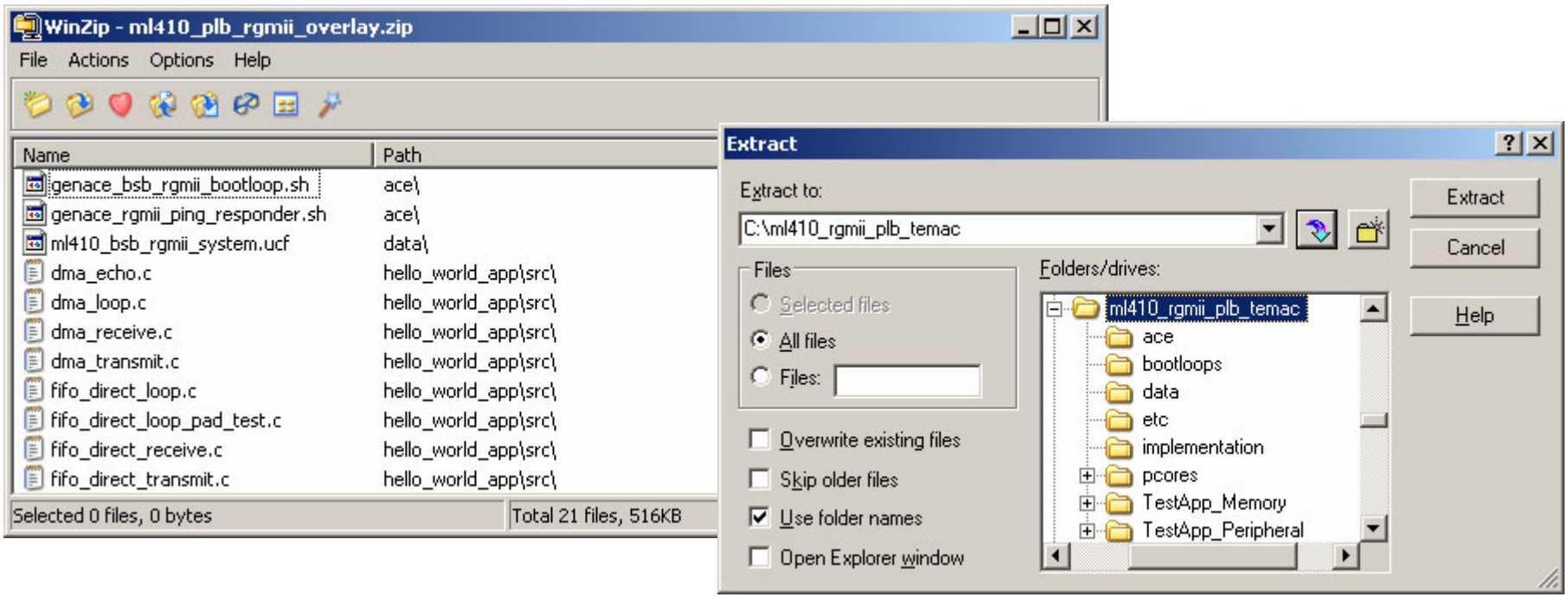
Extracting the Design

- Rename the project directory to ml410_rgmii_plb_temac



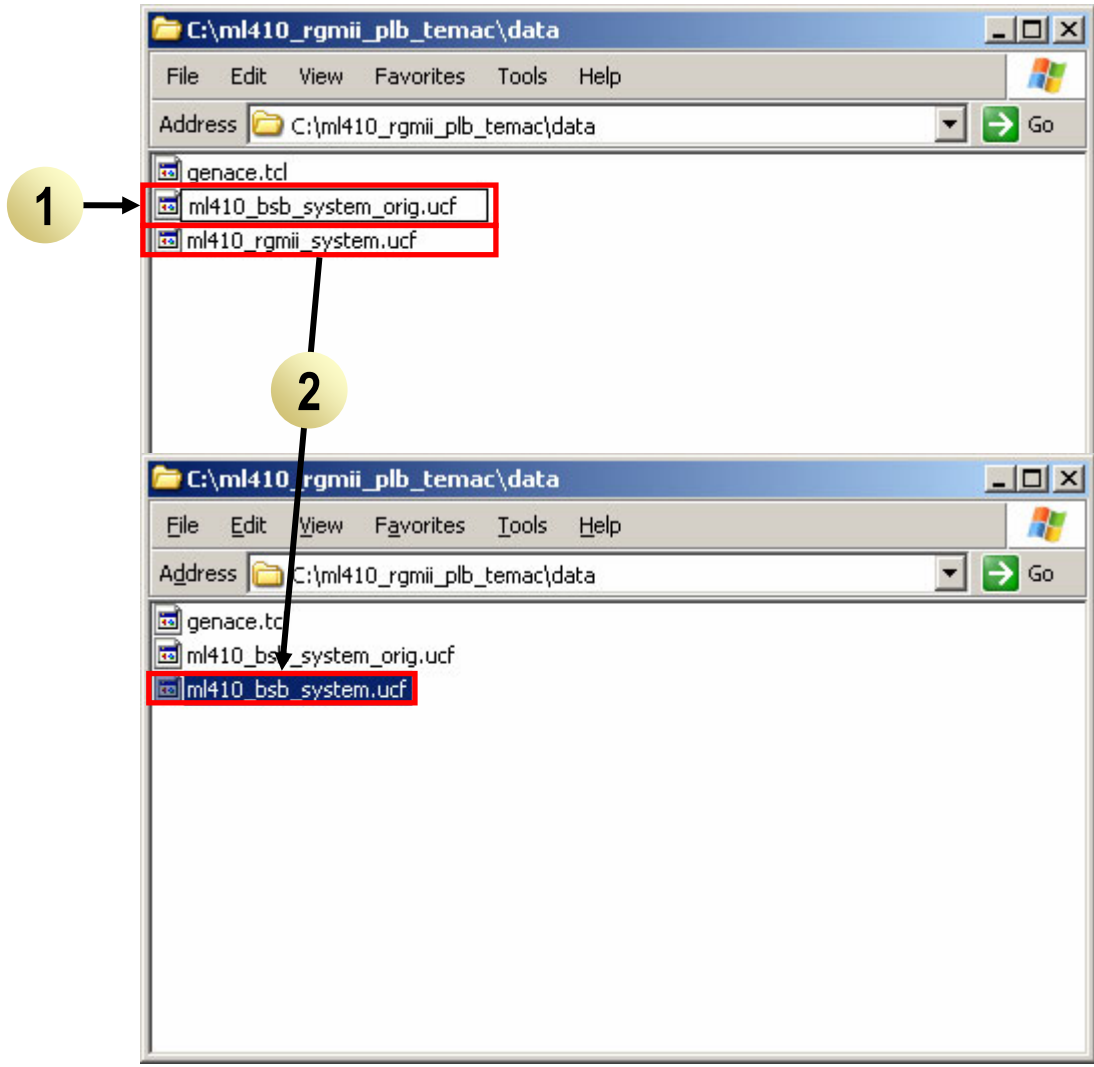
Extracting the Design

- Unzip the ml410_plb_rgmii_overlay.zip file
  - Unzip to the ml410_rgmii_plb_temac directory
  - This adds the PLB TEMAC with RGMII to the design directory



UCF File

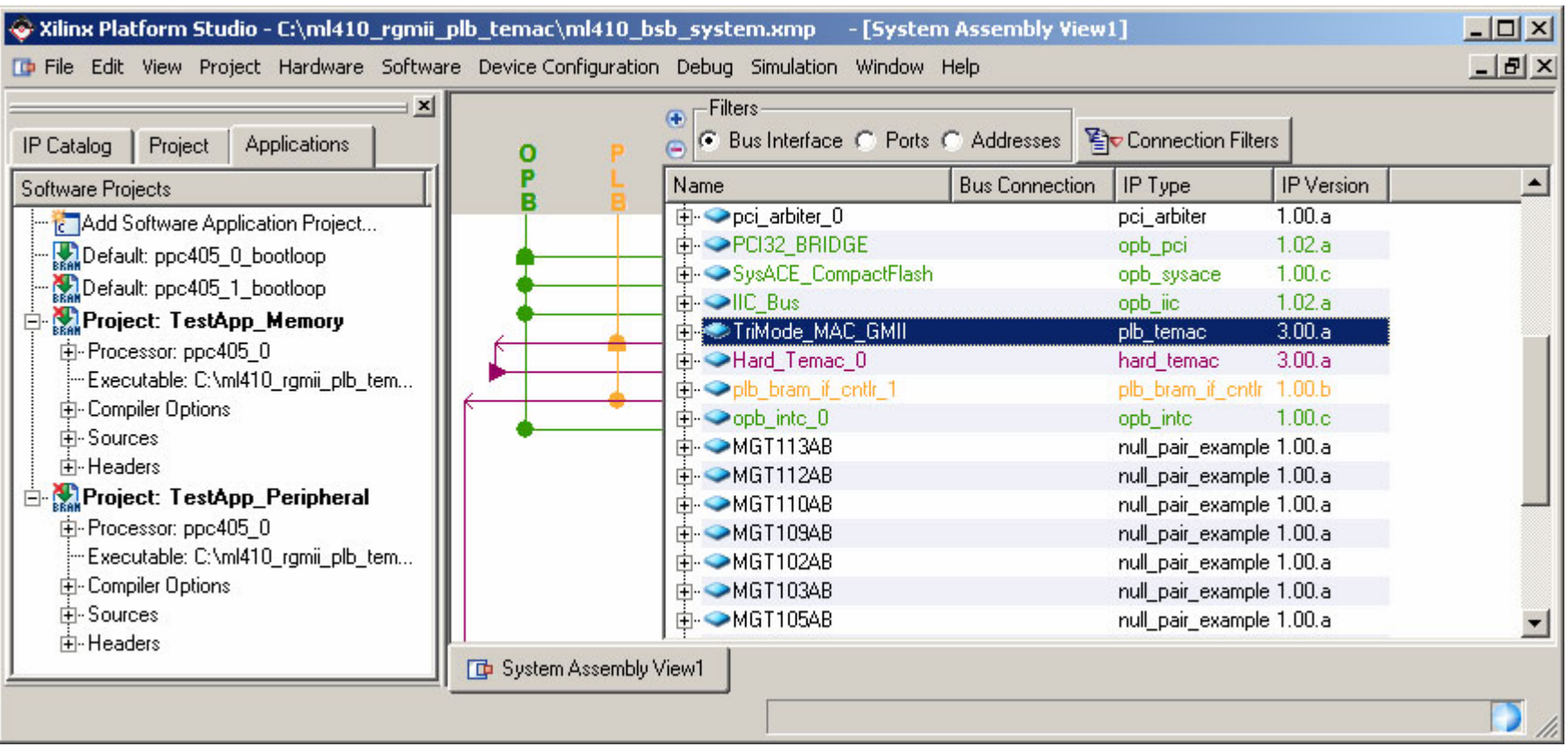
- Rename the ml410_bsb_system.ucf (1)
- Change the file named ml410_rgmii_system.ucf to ml410_bsb_system.ucf (2)



Add and Configure IP

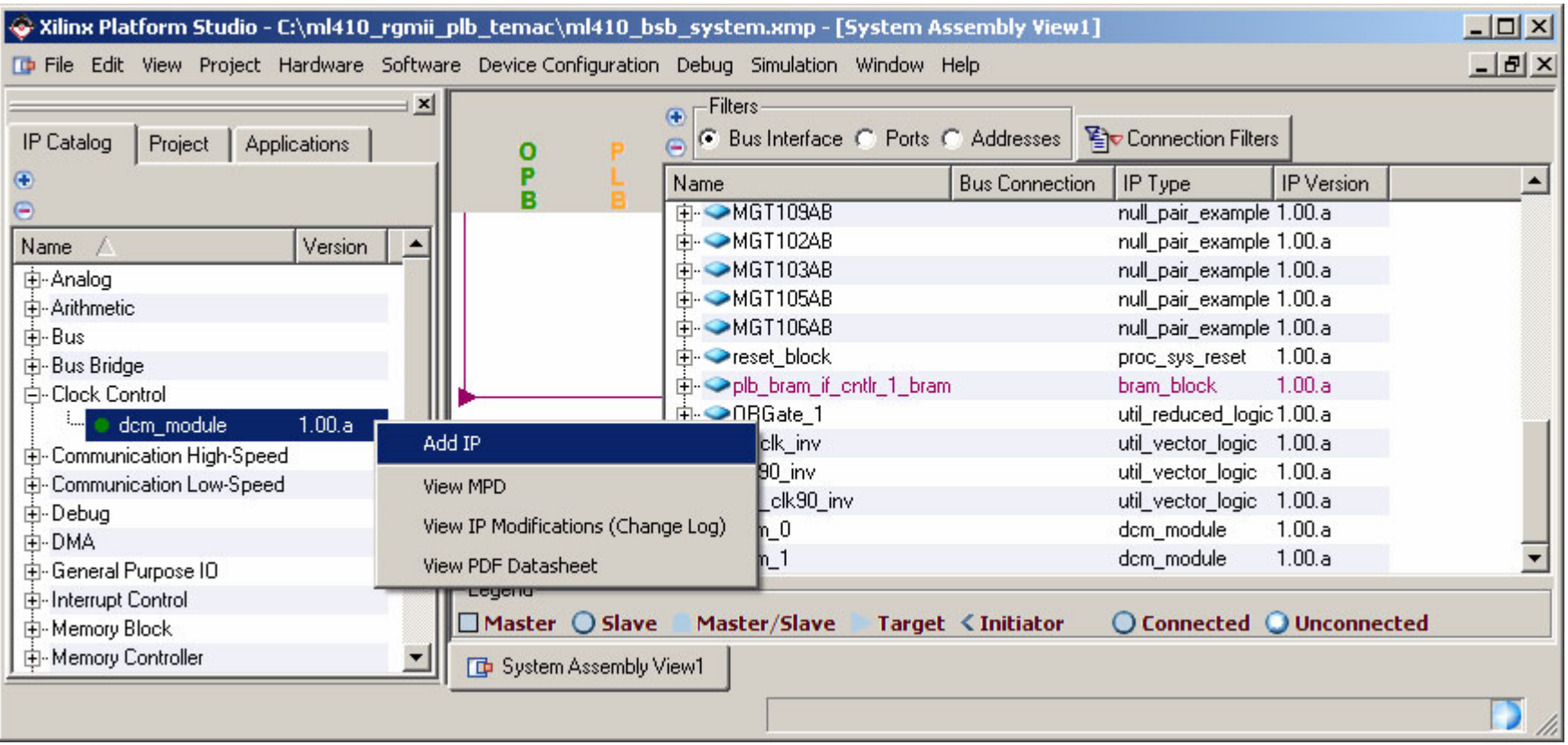
Add RGMII

- Launch EDK project <design path>\ml410_bsb_system.xmp
- Rename the TriMode_MAC_GMII to plb_temac_0



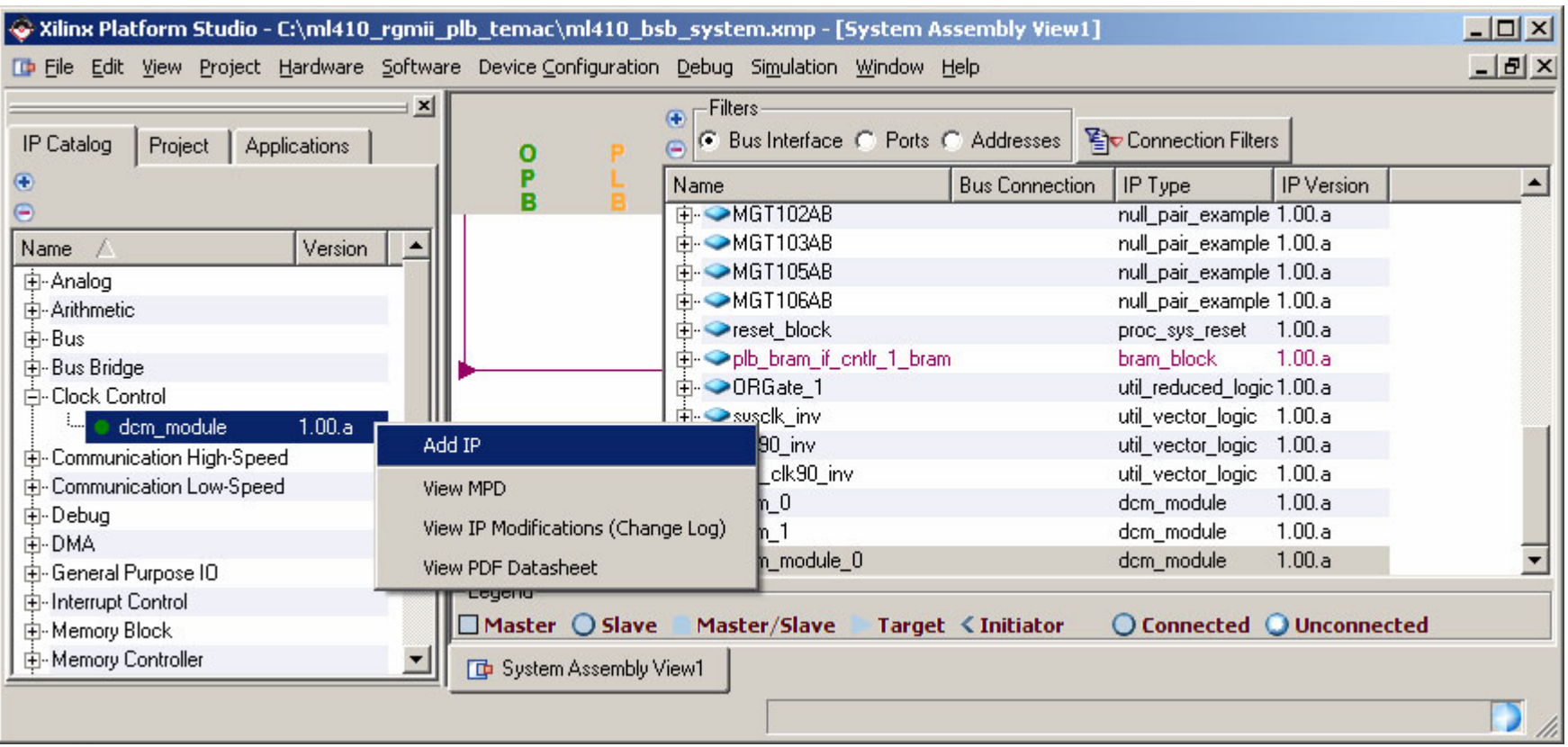
Add RGMII

- Add a dcm



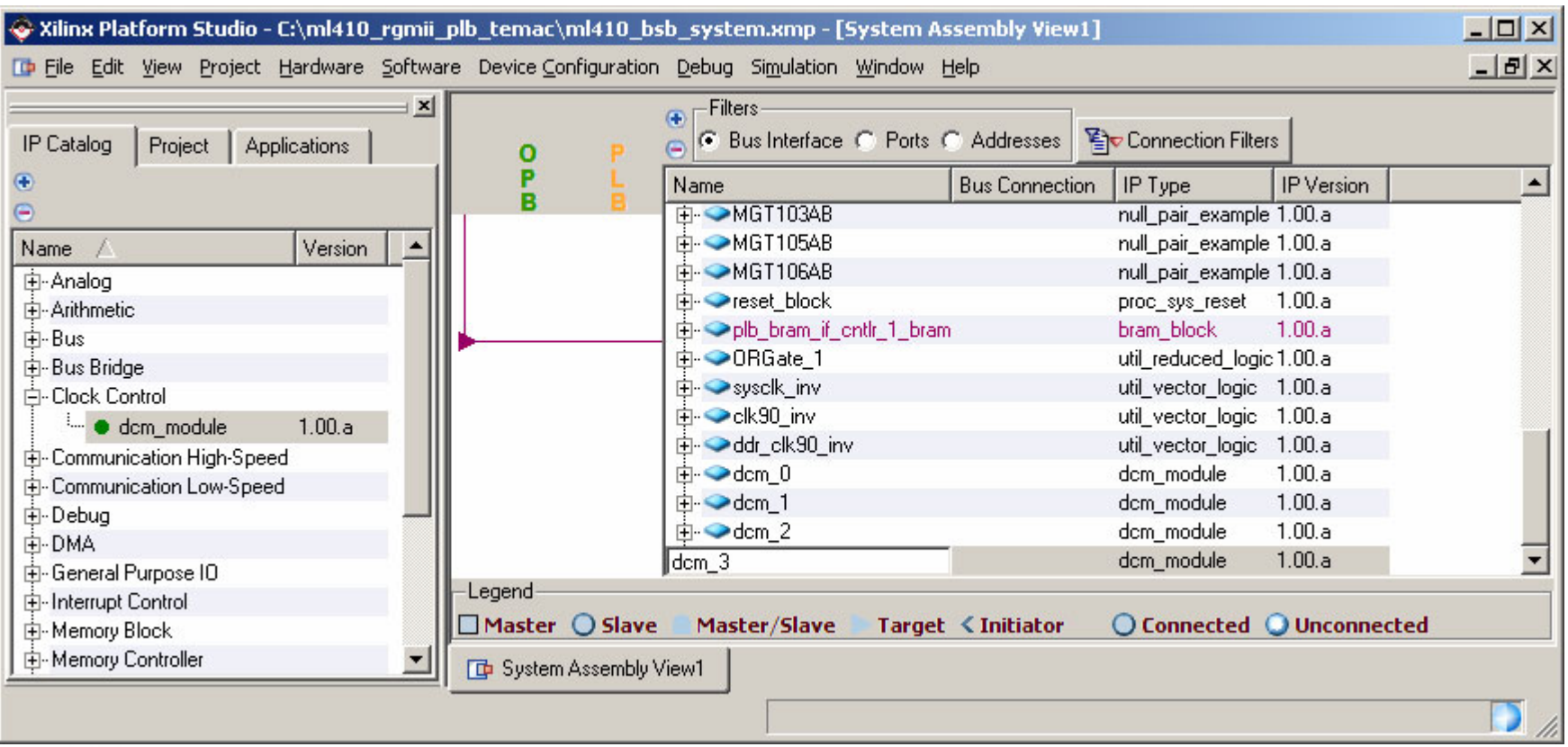
Add RGMII

- Add another dcm



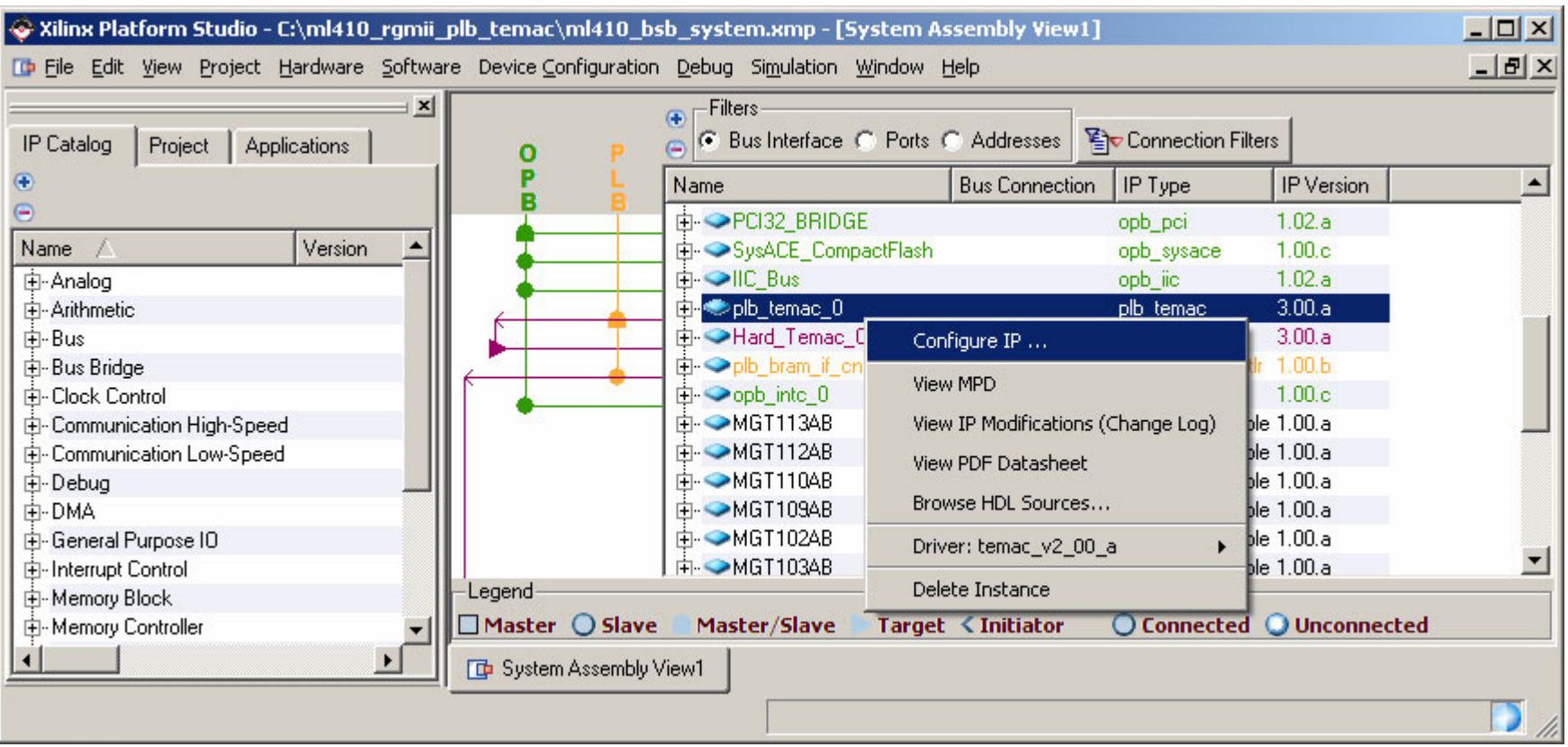
Add RGMII

- Rename the dcm_modules to dcm_2 and dcm_3



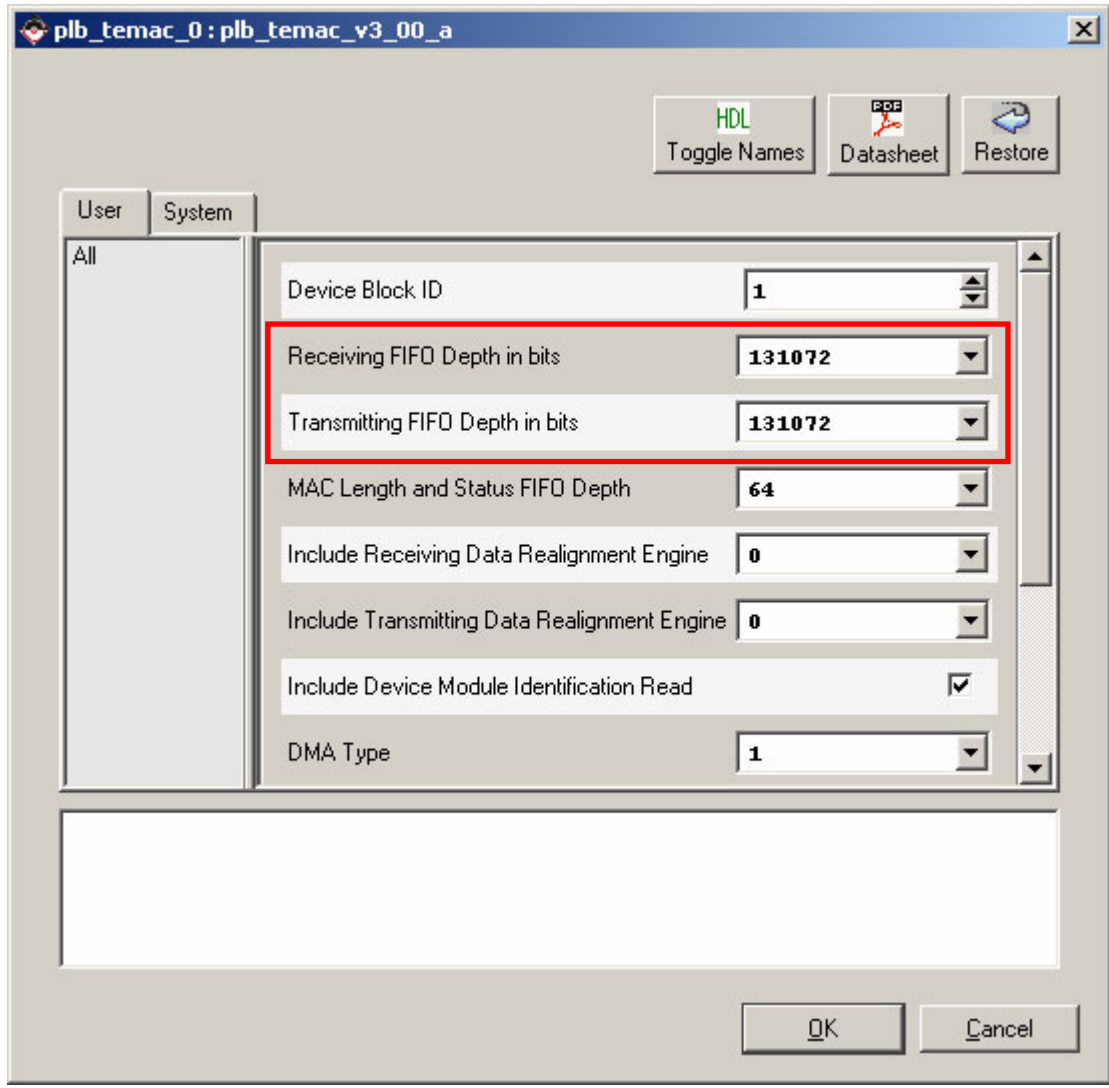
Add RGMII

- Right-click on the plb_temac and select Configure IP …



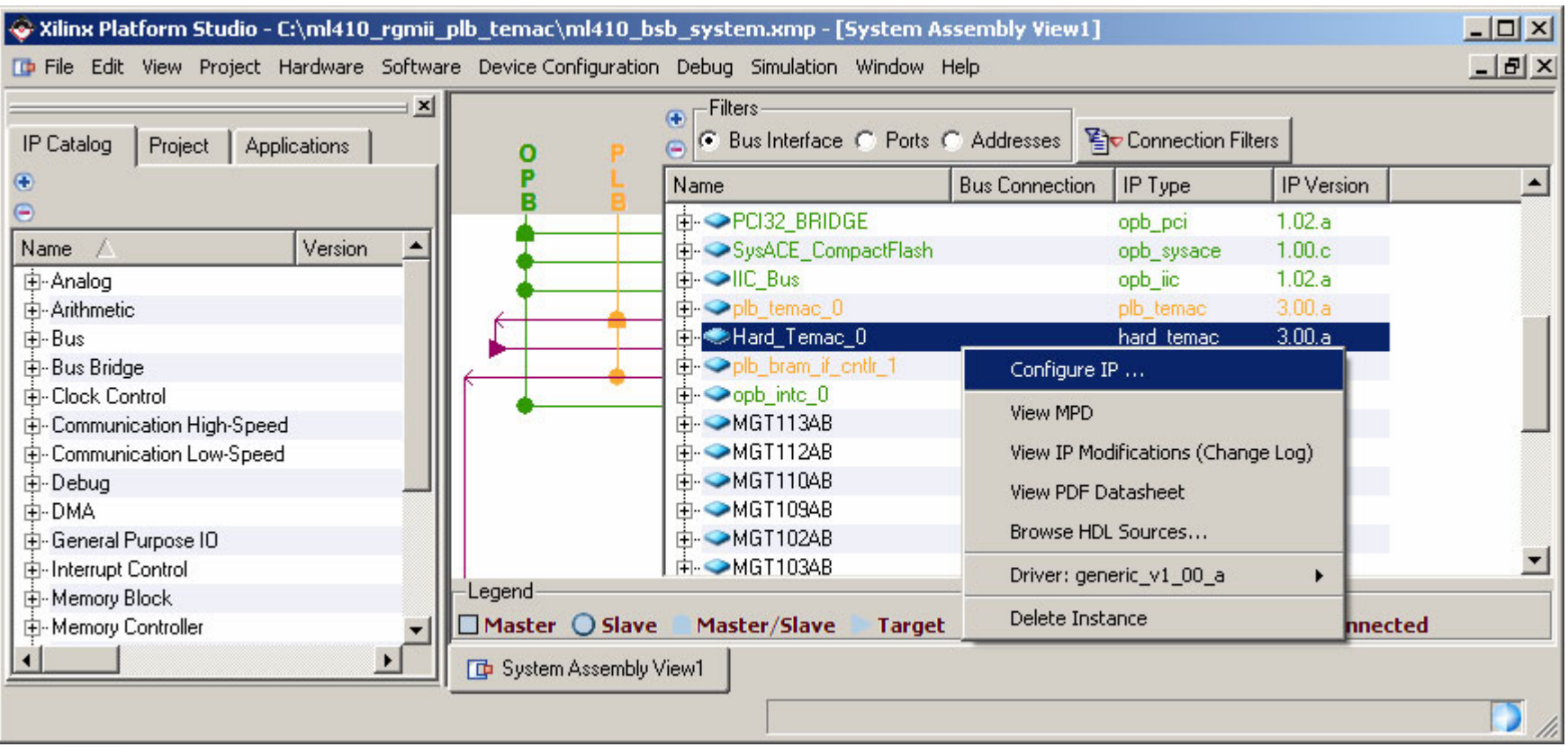
Add RGMII

- Under the User tab, set the following parameters:
  - Receiving FIFO Depth: **131072**
  - Transmitting FIFO Depth: **131072**



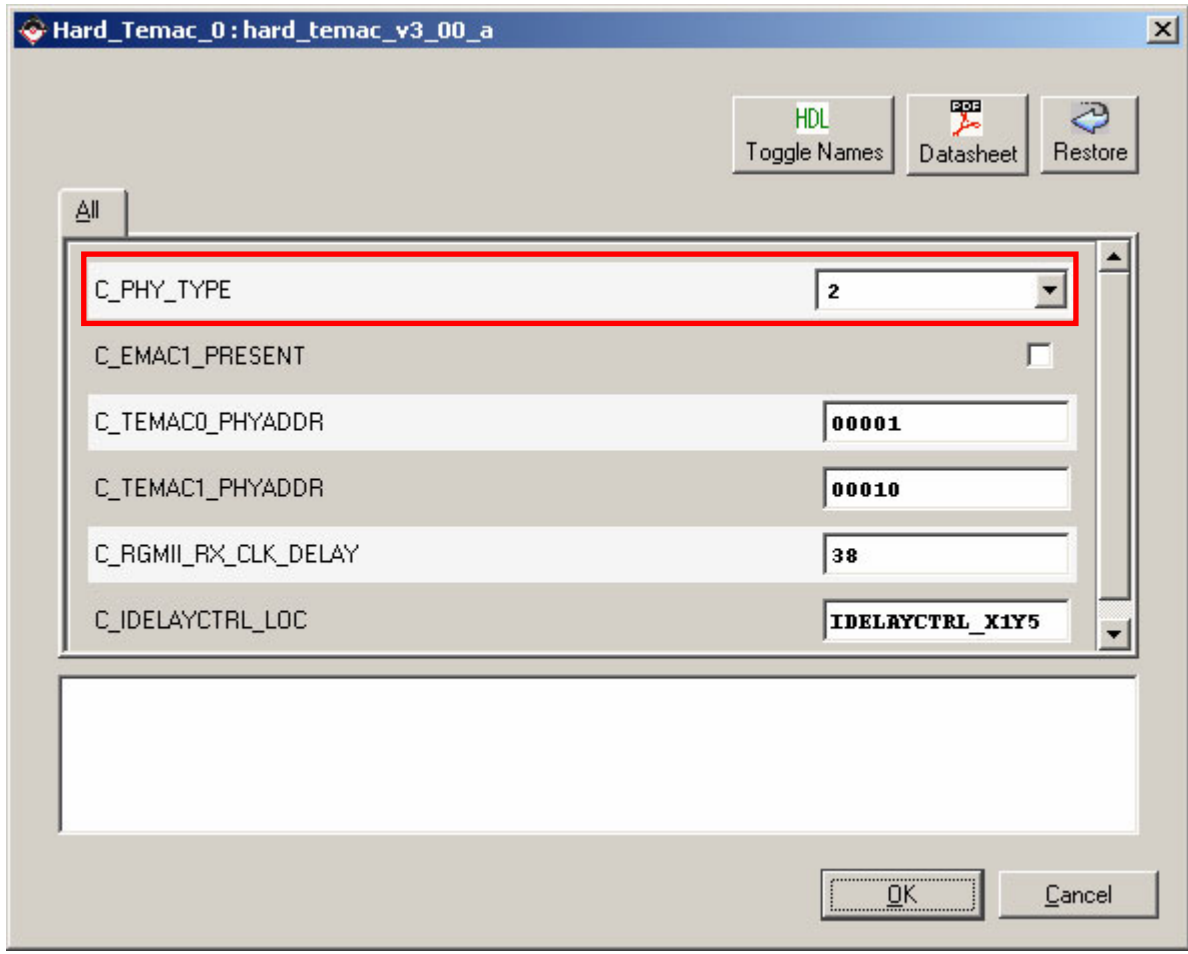
Add RGMII

- Right-click on the Hard_Temac and select **Configure IP …**



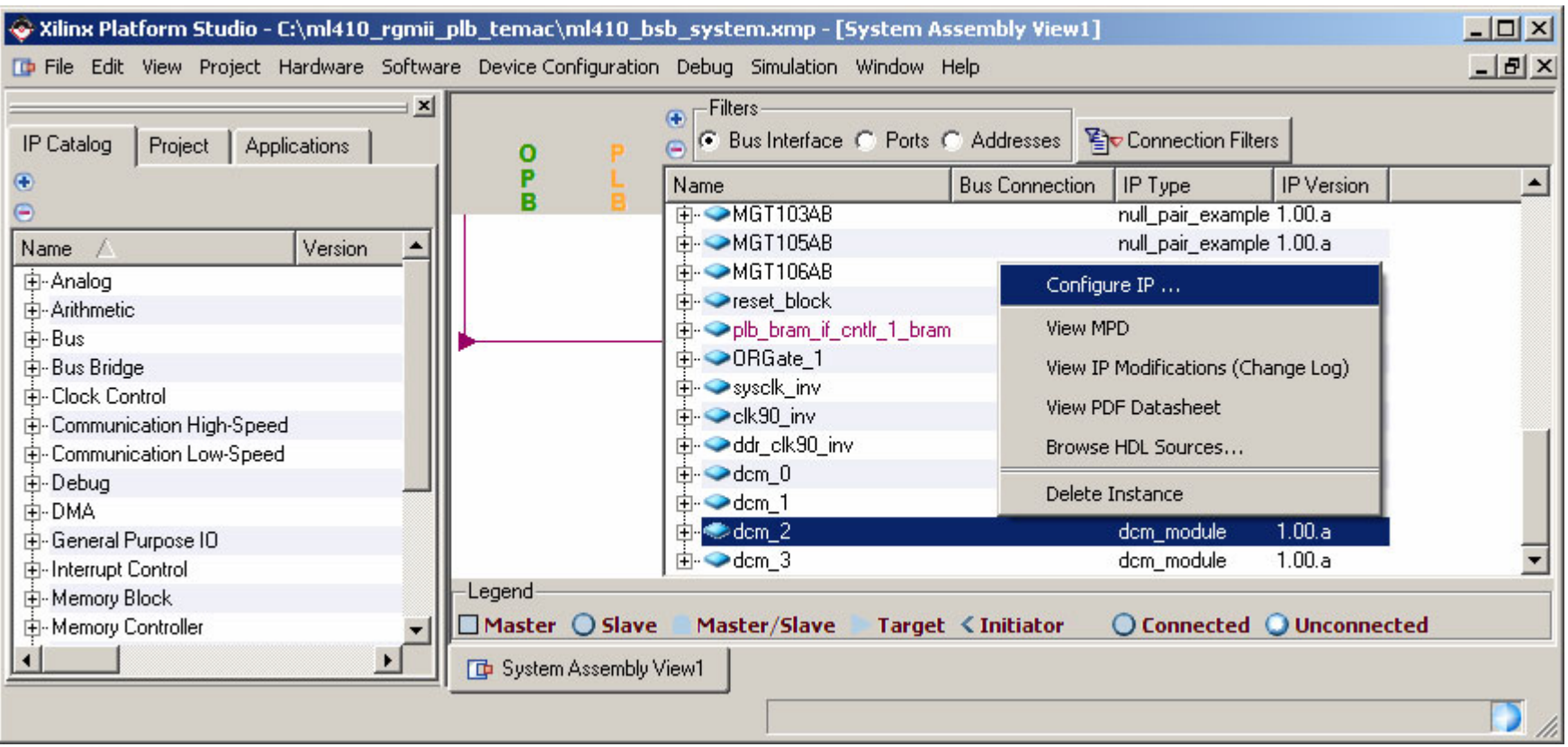
Add RGMII

- Set the C_PHY_TYPE to 2 for RGMII



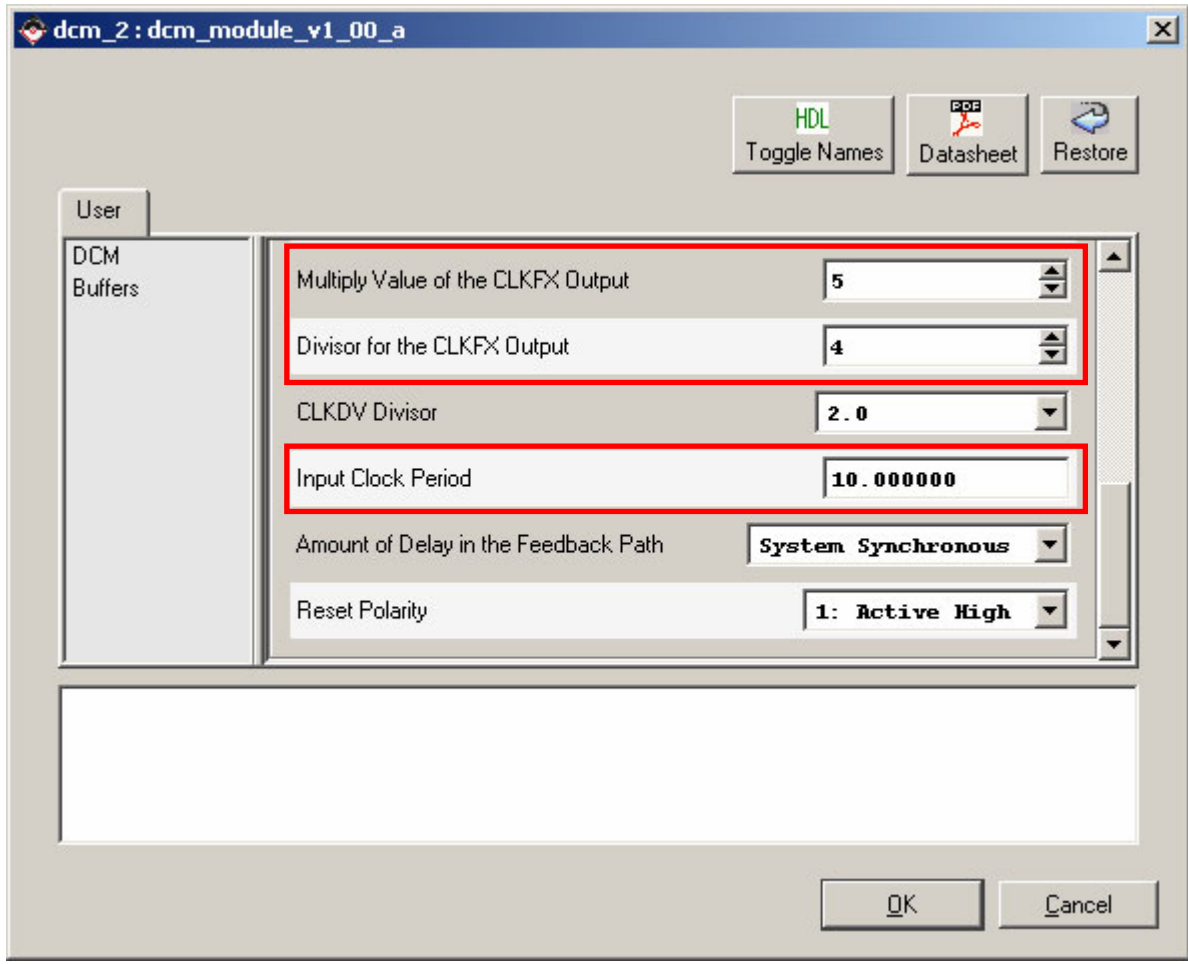
Add RGMII

- Right-click on the dcm_2 and select **Configure IP ...**



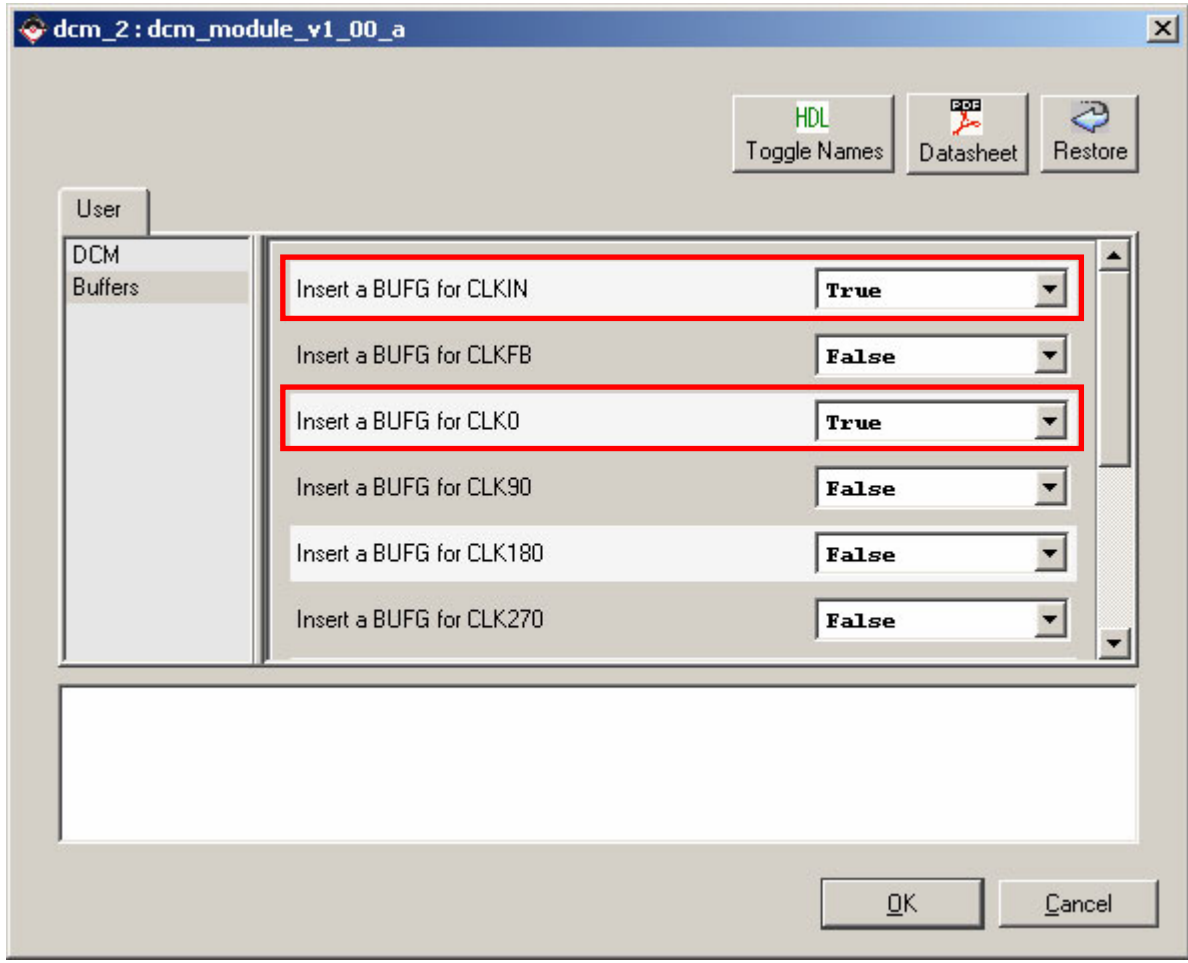
Add PLB TFT Controller

- Make the following settings to dcm_2:
  - Multiply Value of the CLKFX Output = 5
  - Divisor for the CLKFX Output = 4
  - Input Clock Period = 10.000000



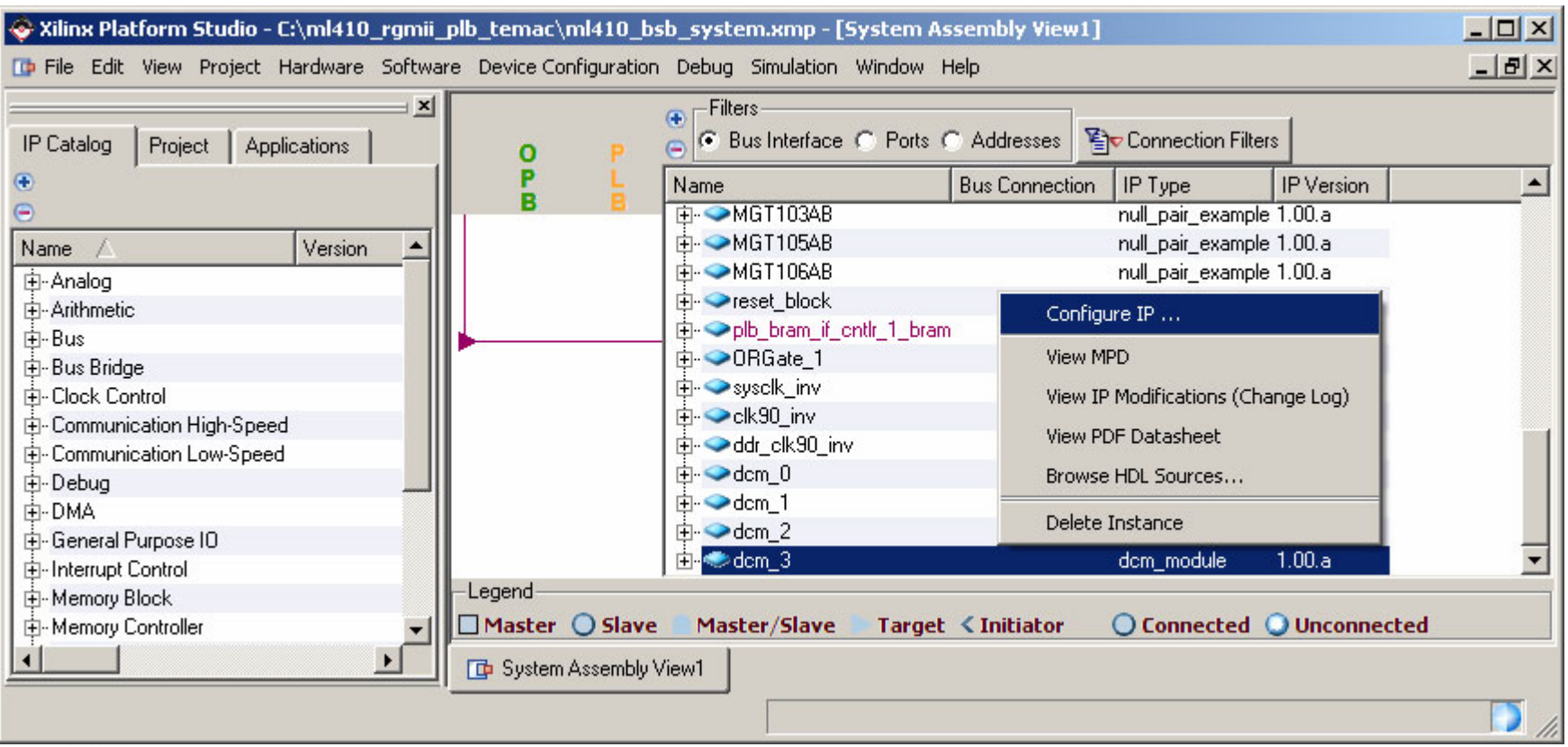
Add PLB TFT Controller

- Set the following buffers to True:
  - CLKIN
  - CLK0



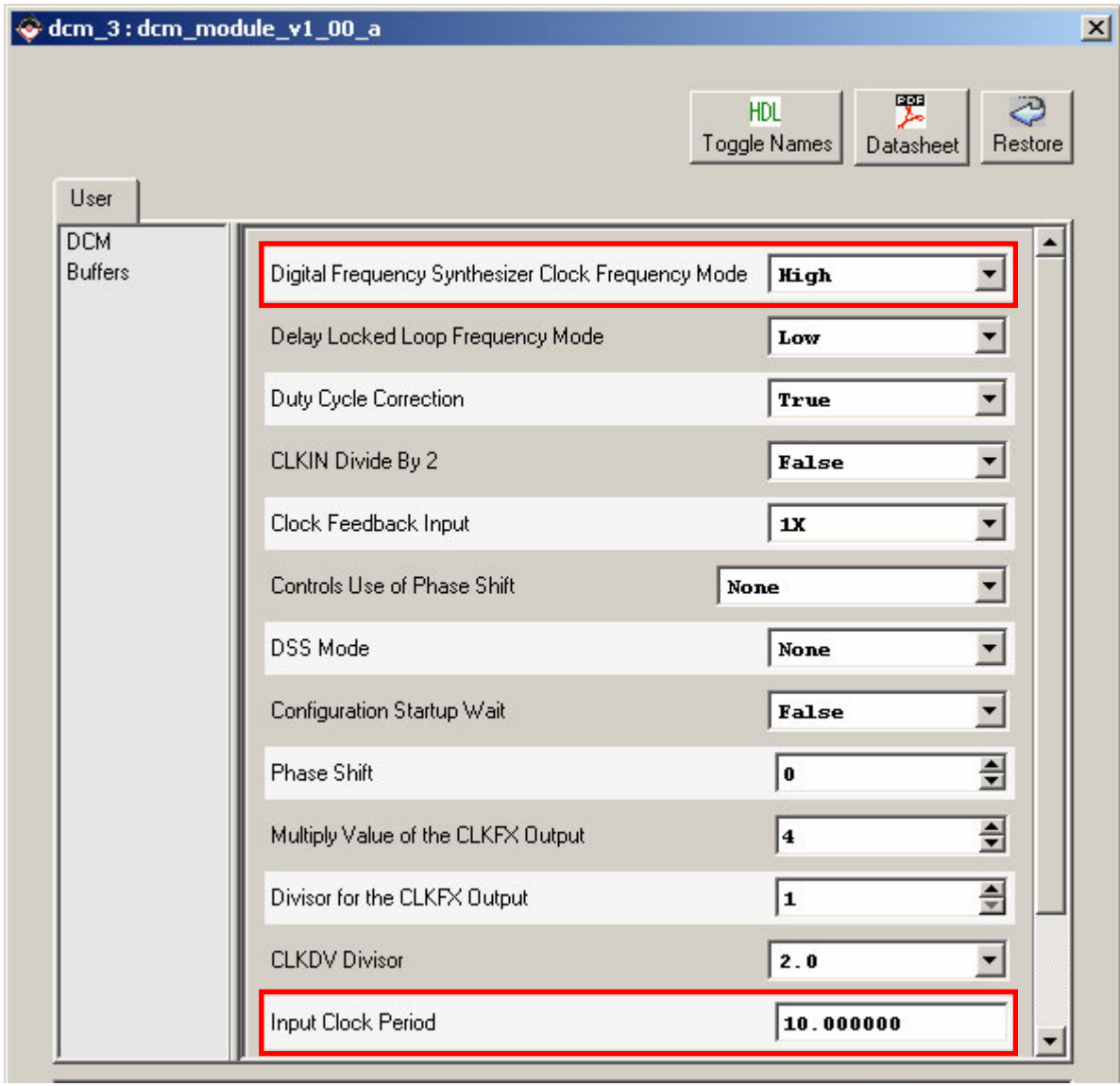
Add RGMII

- Right-click on the dcm_3 and select **Configure IP …**



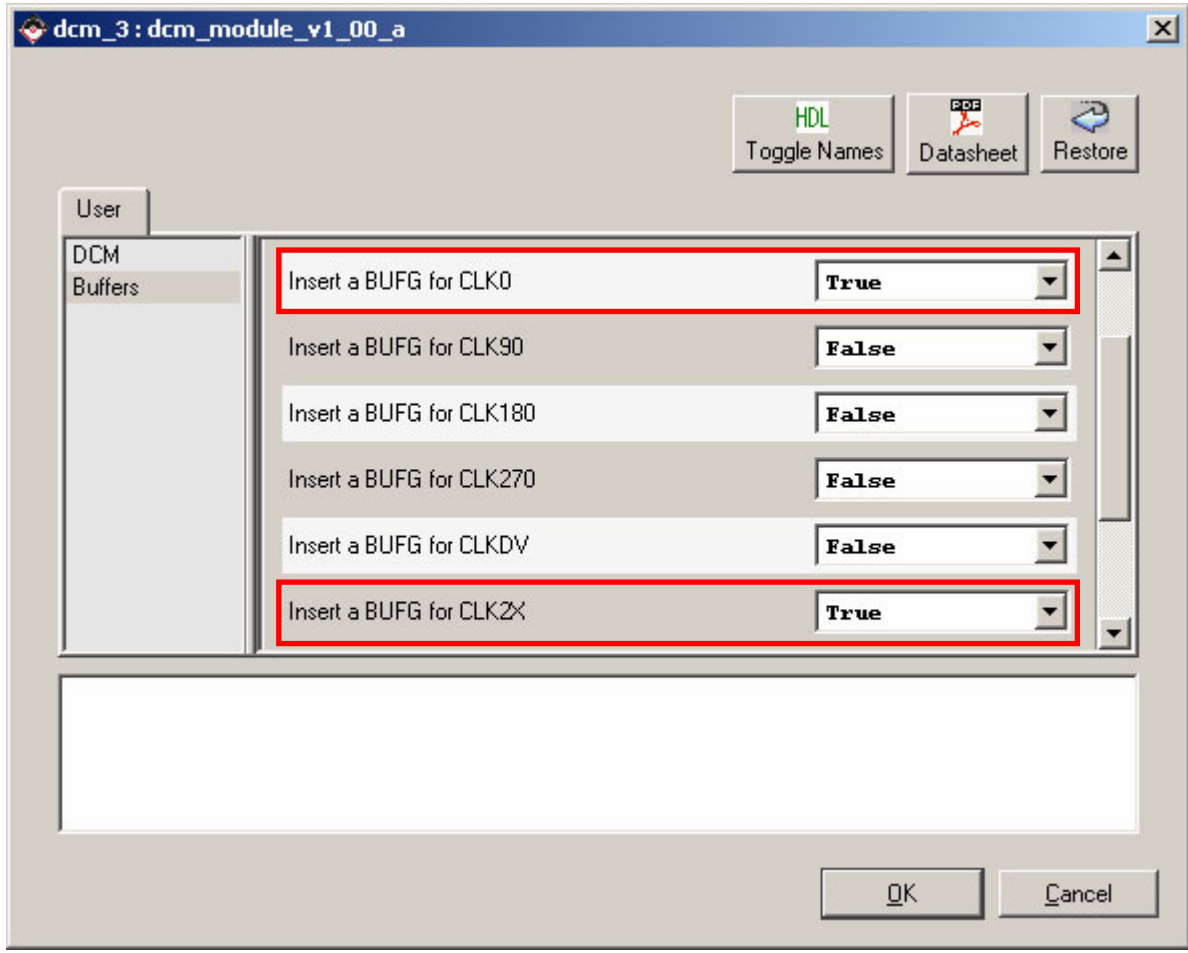
Add PLB TFT Controller

- Make the following settings to dcm_3:
  - Clock Frequency Mode = High
  - Input Clock Period = 10.000000



Add PLB TFT Controller

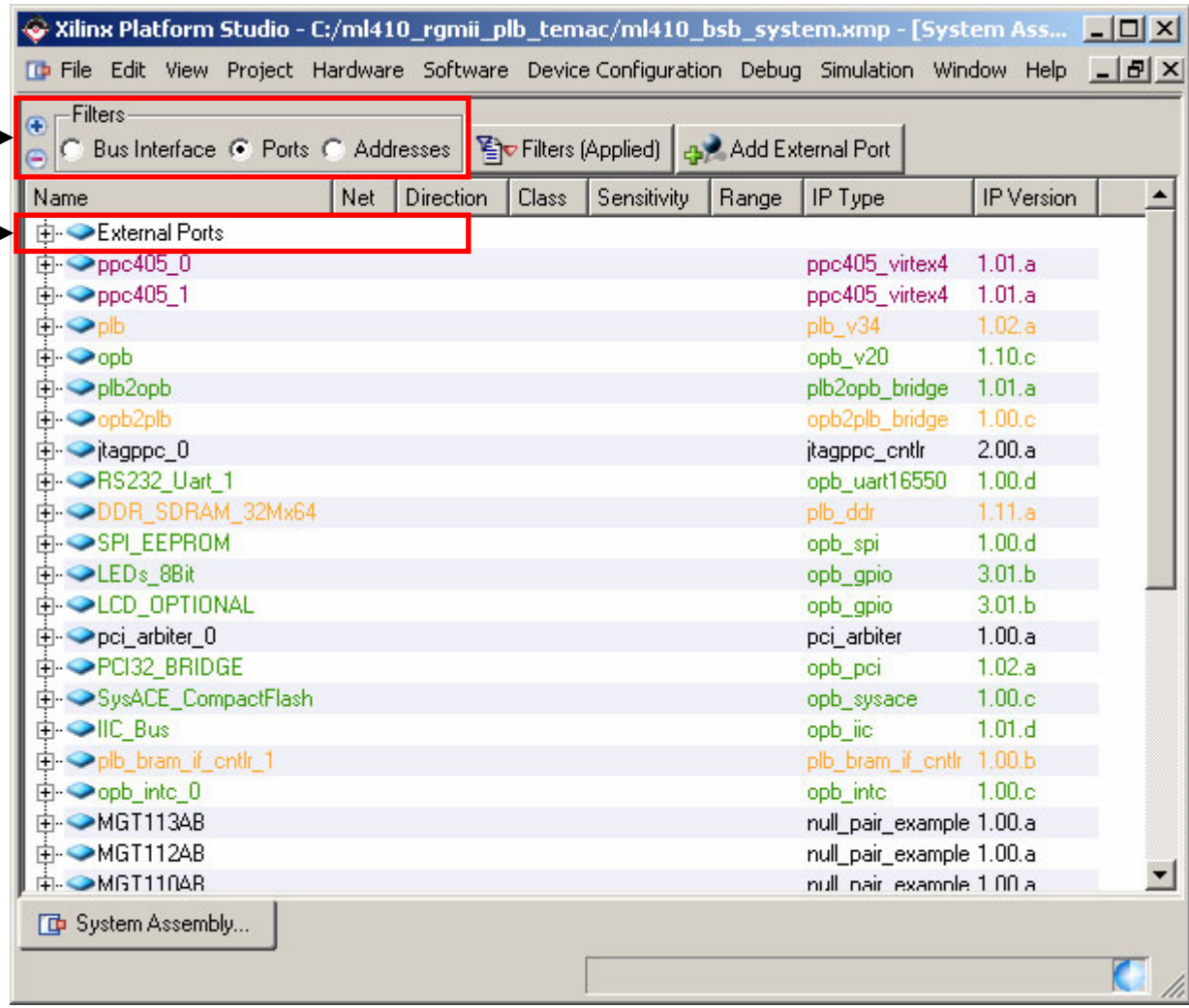
- Set the following buffers to True:
  - CLK0
  - CLK2X



Add External Ports
Connect the Nets

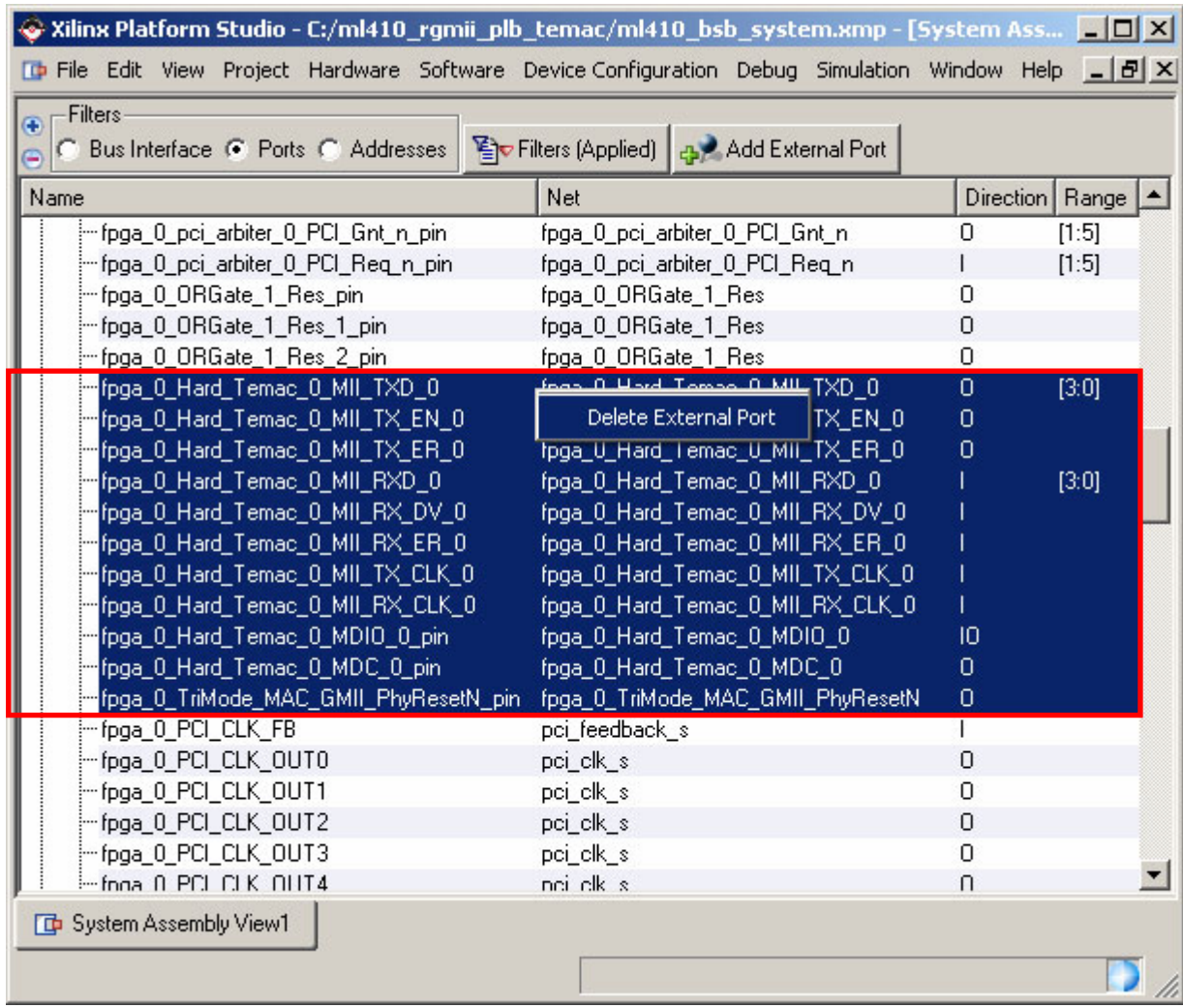
Add RGMII

- Click on the Ports Button (1)
- Expand the External Ports (2)



Add RGMII

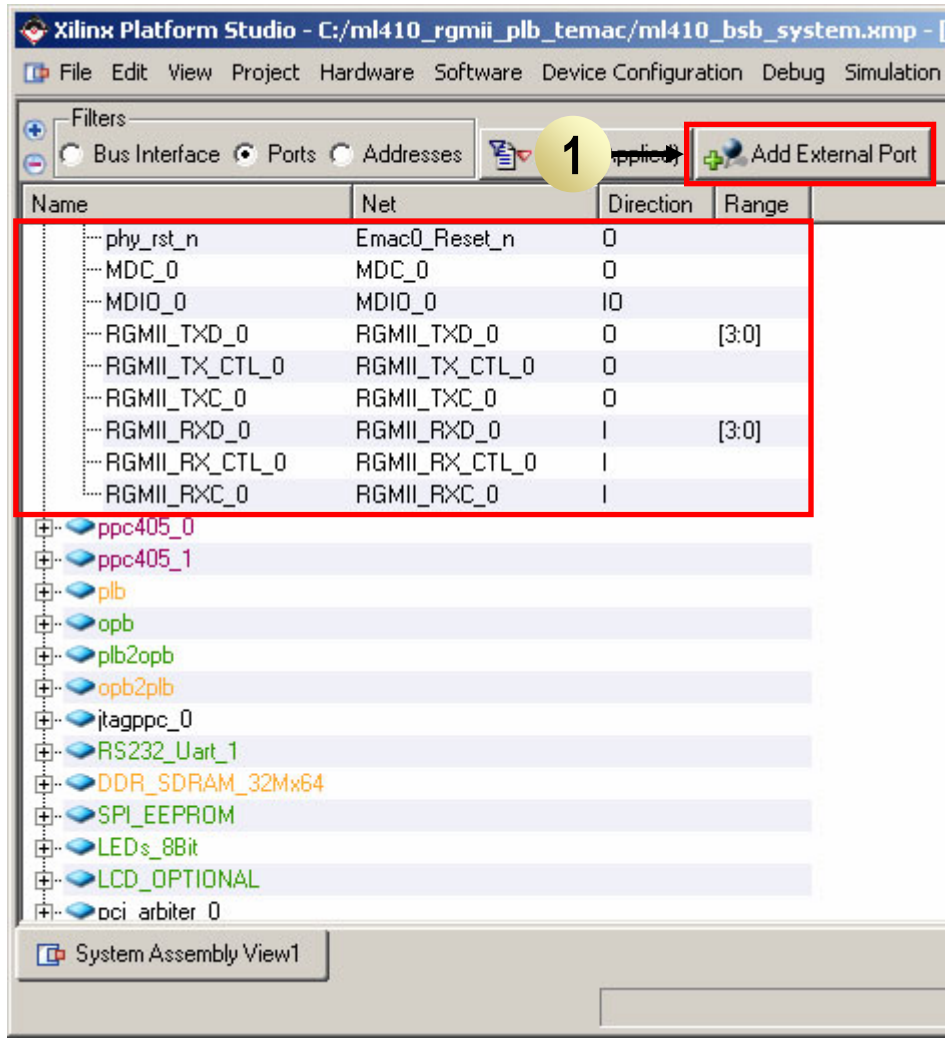
- Delete all the existing Ethernet ports



Note: If using the BSB tool to build the design from scratch, don't add the Ethernet_MAC

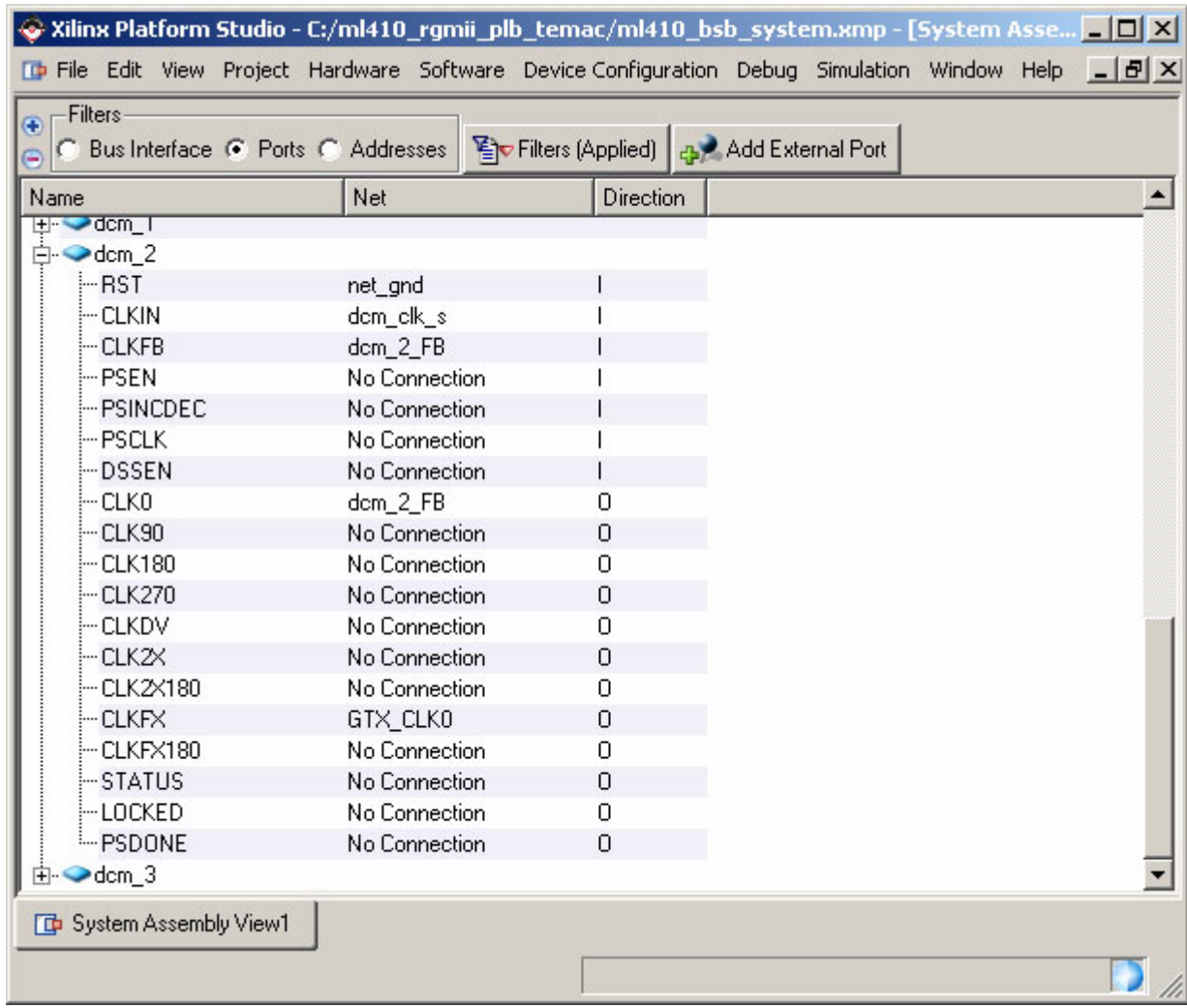
Add RGMII

- Use the Add External Port button (1) to add these Ports:
  - phy_rst_n = Emac0_Reset_n
  - MDC_0 = MDC_0
  - MDIO_0 = MDIO_0
  - RGMII_TXD_0 = RGMII_TXD_0
  - RGMII_TX_CTL_0 = RGMII_TX_CTL_0
  - RGMII_TXC_0 = RGMII_TXC_0
  - RGMII_RXD_0 = RGMII_RXD_0
  - RGMII_RX_CTL_0 = RGMII_RX_CTL_0
  - RGMII_RXC_0 = RGMII_RXC_0
- Set the direction and range as shown



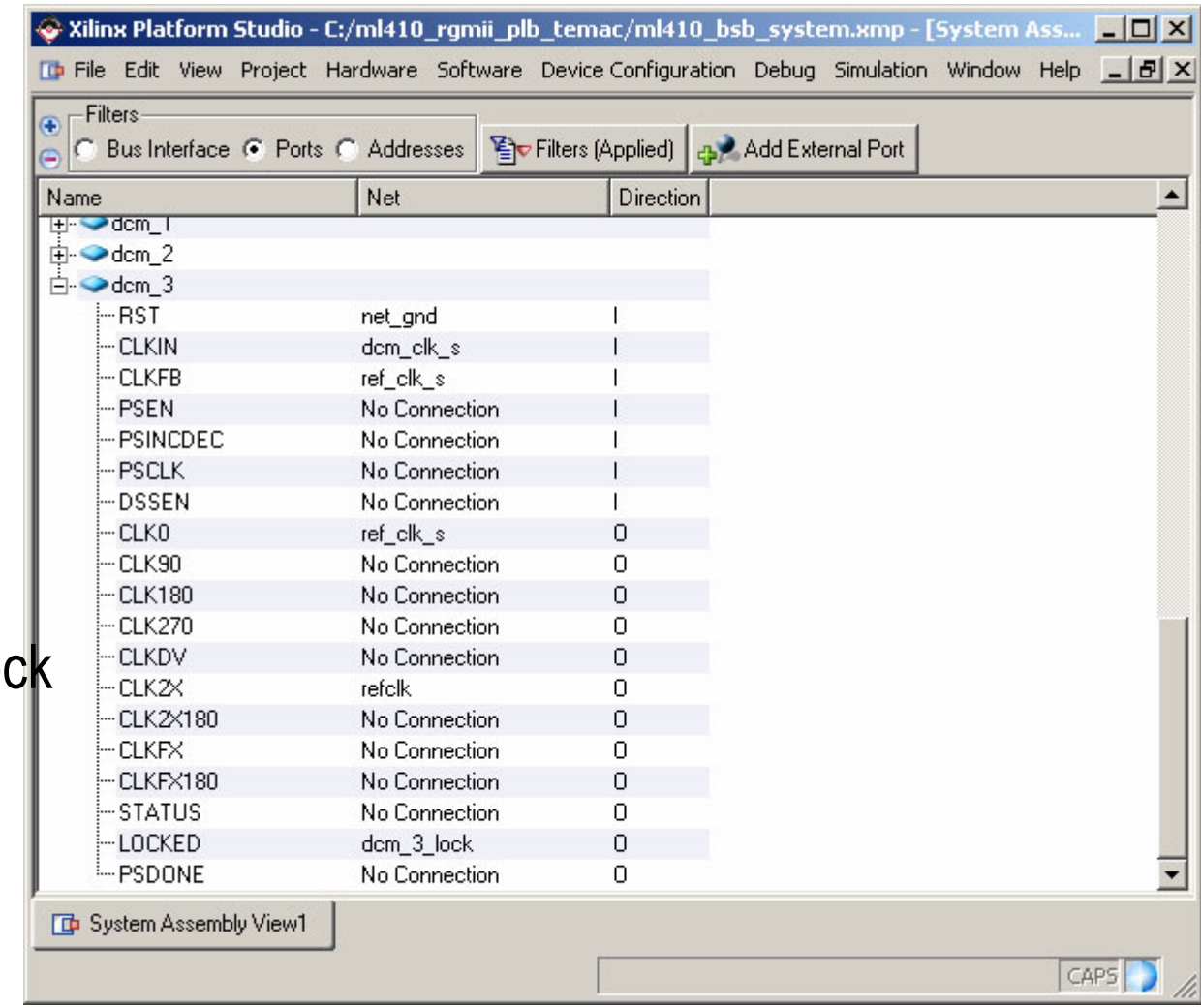
Add RGMII

- Add the following ports to dcm_2:
  - RST = net_gnd
  - CLKIN = dcm_clk_s
  - CLKFB = dcm_2_FB
  - CLK0 = dcm_2_FB
  - CLKFX = GTX_CLK0



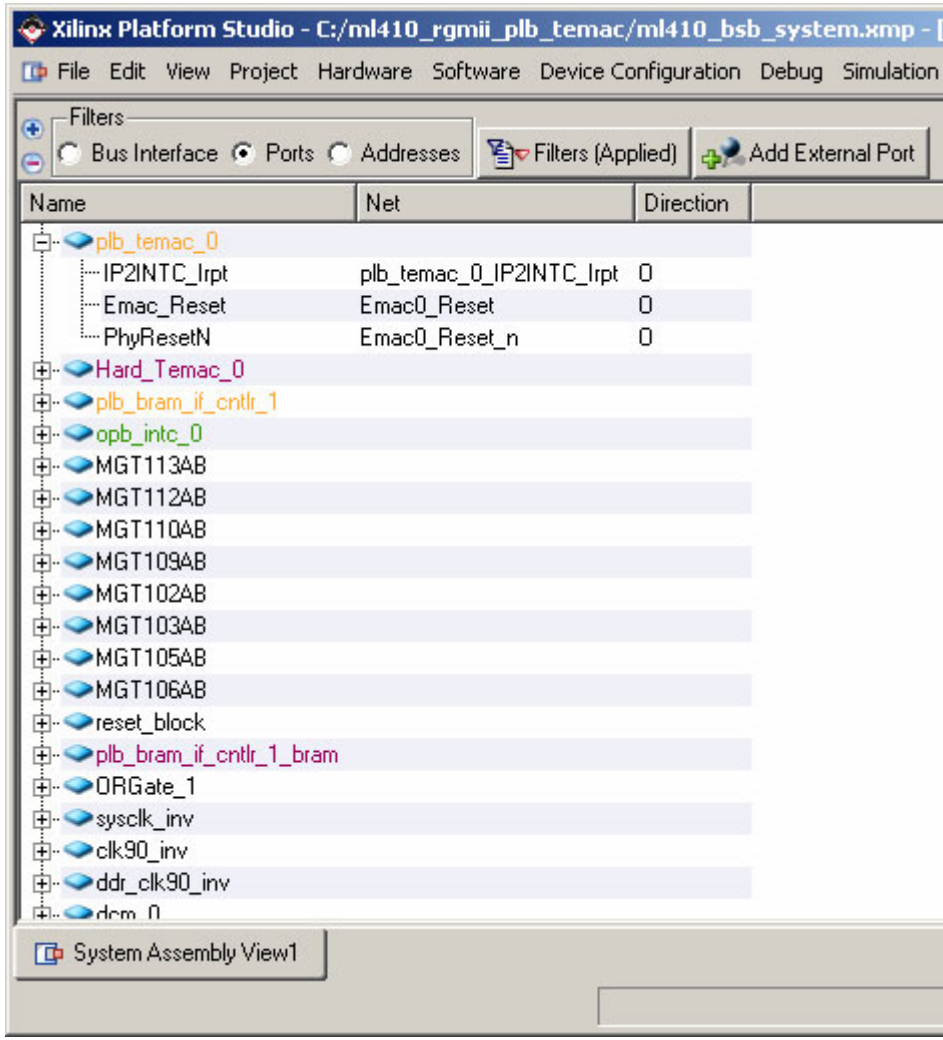
Add RGMII

- Add the following ports to dcm_3:
  - RST = net_gnd
  - CLKIN = dcm_clk_s
  - CLKFB = ref_clk_s
  - CLK0 = ref_clk_s
  - CLK2X = refclk
  - LOCKED = dcm_3_lock



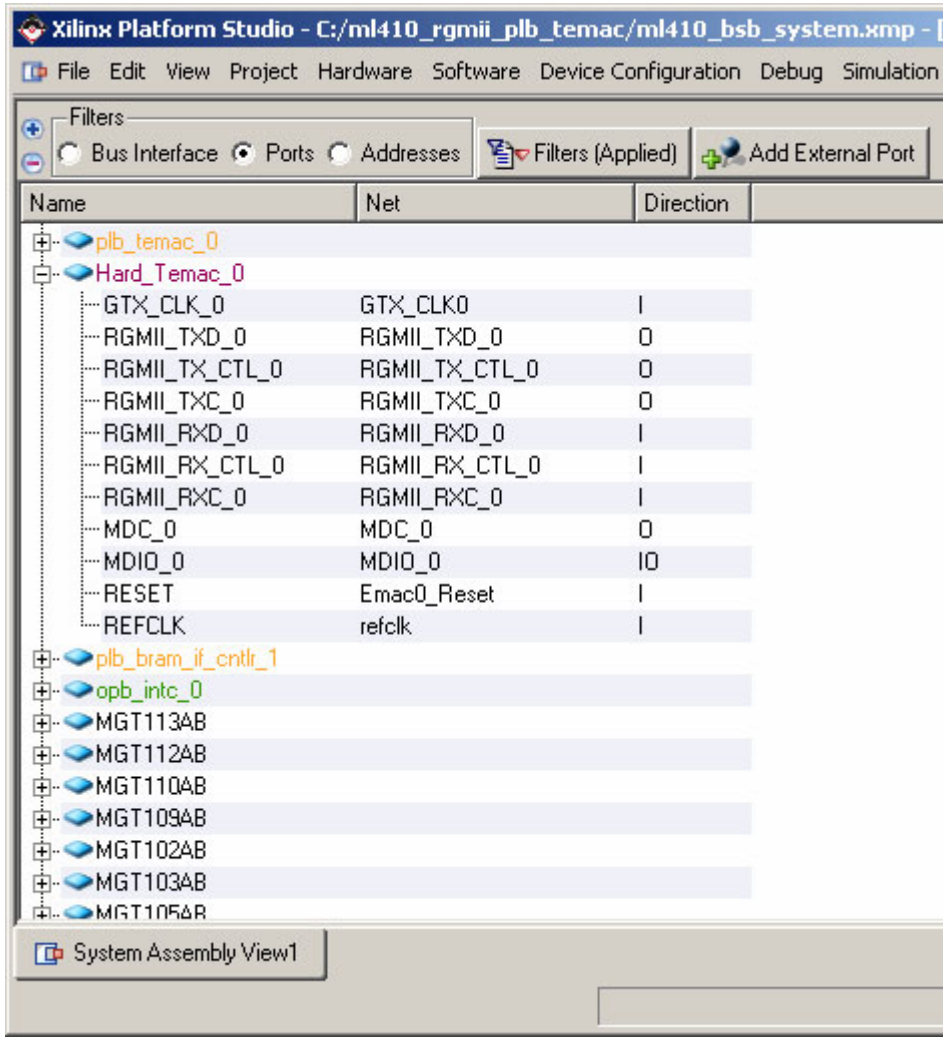
Add RGMII

- Add the following ports to plb_temac:
  - IP2INTC_Irpt = plb_temac_0_IP2INTC_Irpt
  - Emac_Reset = Emac0_Reset
  - PhyResetN = Emac0_Reset_n



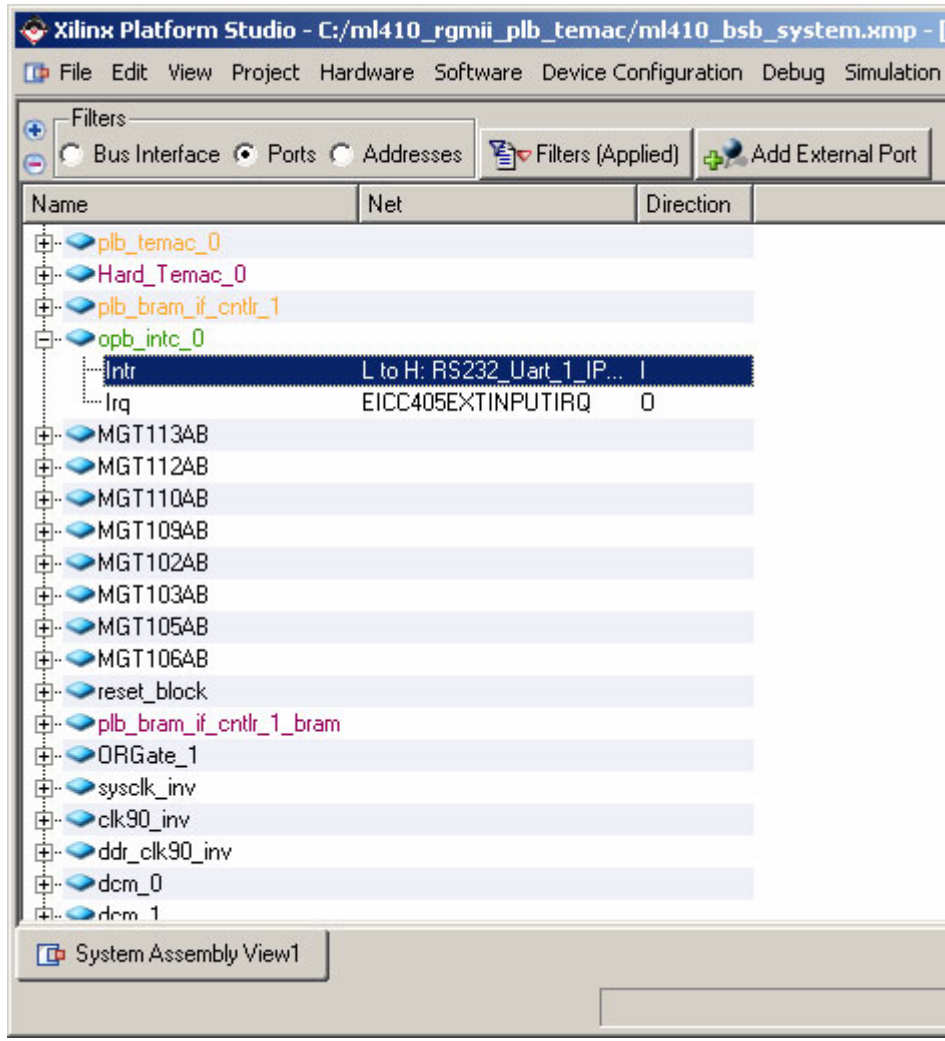
Add RGMII

- Add the following ports to hard_temac:
  - GTX_CLK_0 = GTX_CLK0
  - RGMII_TXD_0 = RGMII_TXD_0
  - RGMII_TX_CTL_0 = RGMII_TX_CTL_0
  - RGMII_TXC_0 = RGMII_TXC_0
  - RGMII_RXD_0 = RGMII_RXD_0
  - RGMII_RX_CTL_0 = RGMII_RX_CTL_0
  - RGMII_RXC_0 = RGMII_RXC_0
  - MDC_0 = MDC_0
  - MDIO_0 = MDIO_0
  - Reset = Emac0_Reset
  - REFCLK = refclk



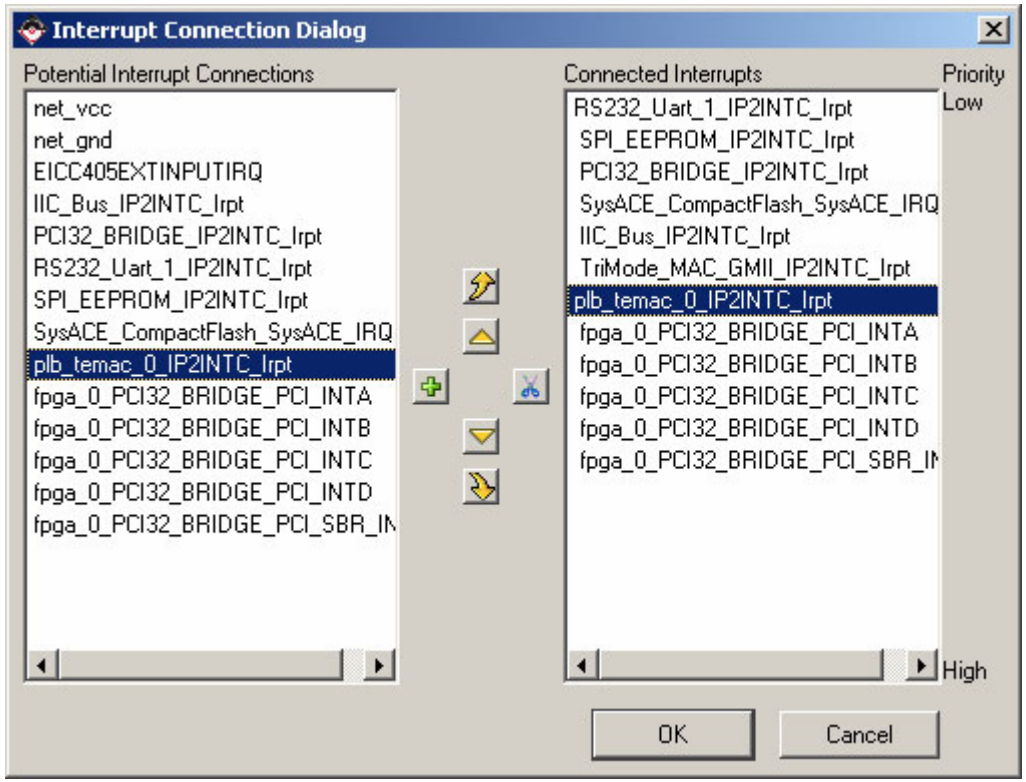
Add RGMII

- Expand opb_intc_0
- Click on “L to H: RS232…”



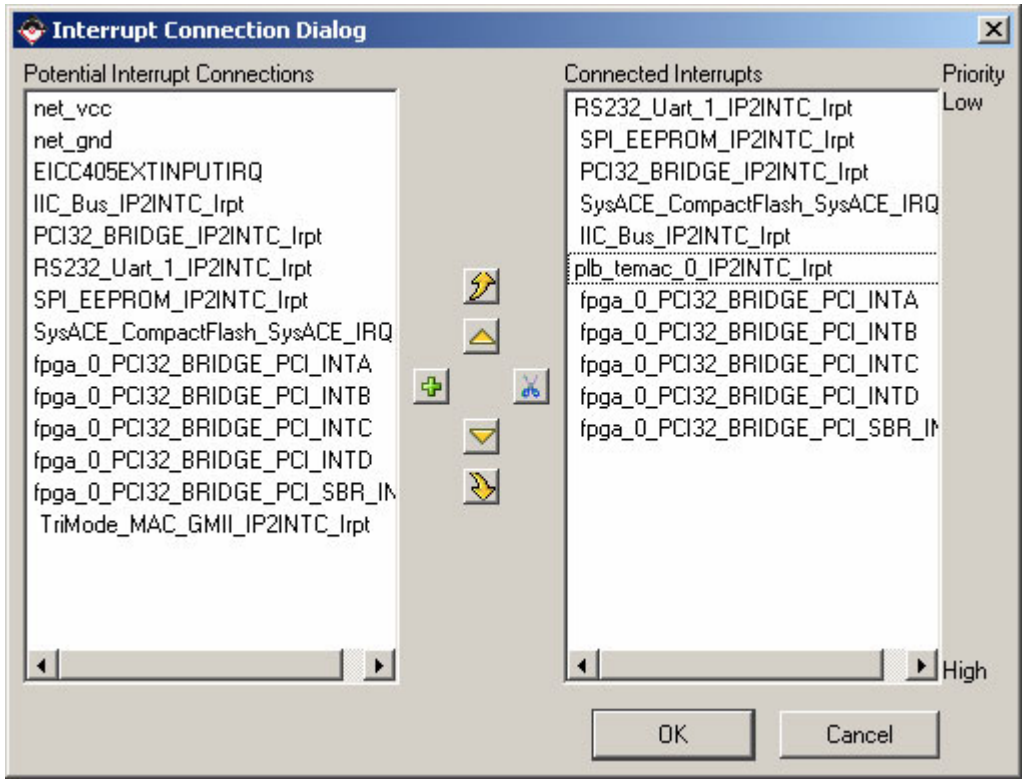
Add RGMII

- Add plb_temac_0_IP2INTC_irpt and move up to the TriMode interrupt



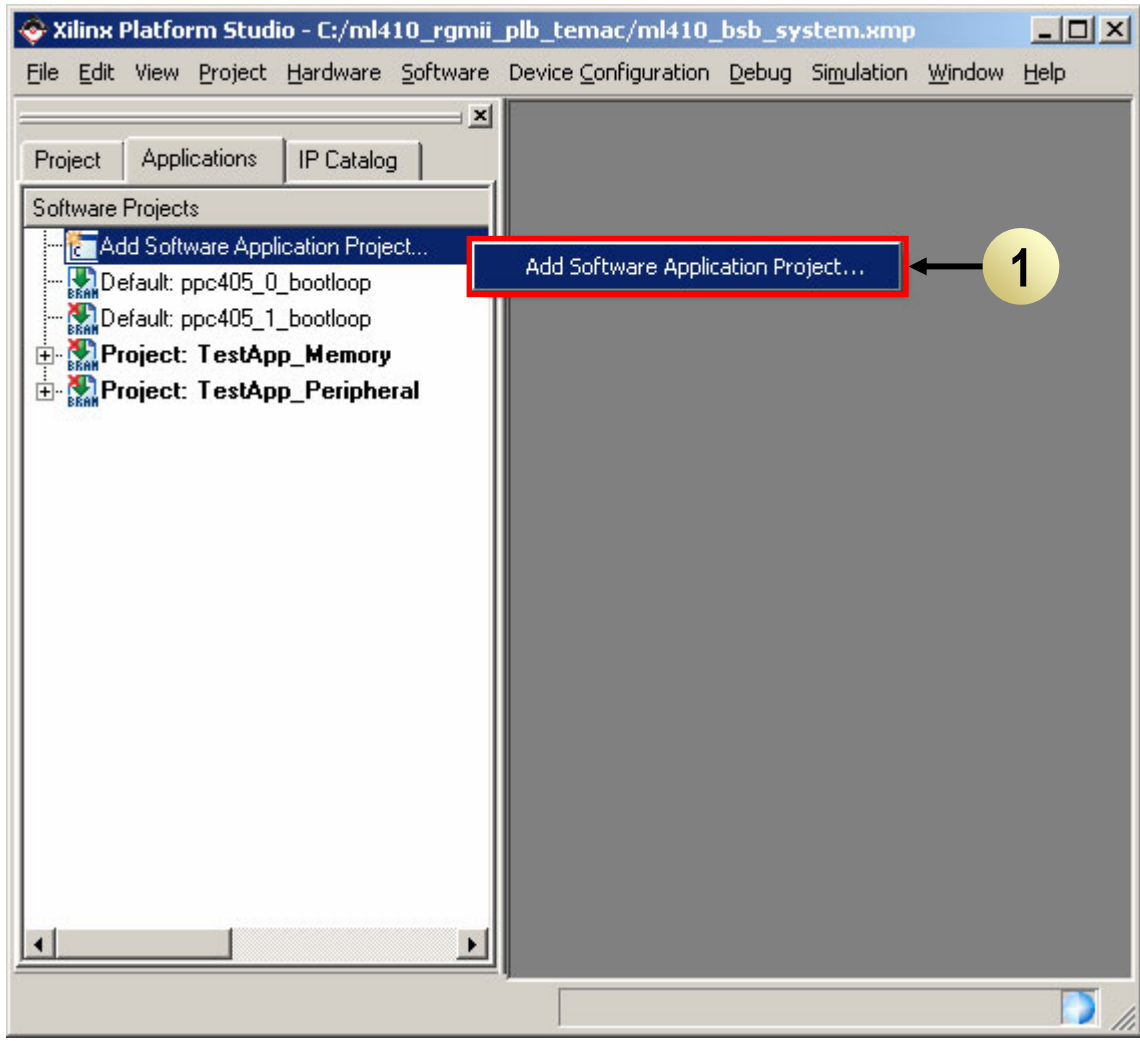
Add RGMII

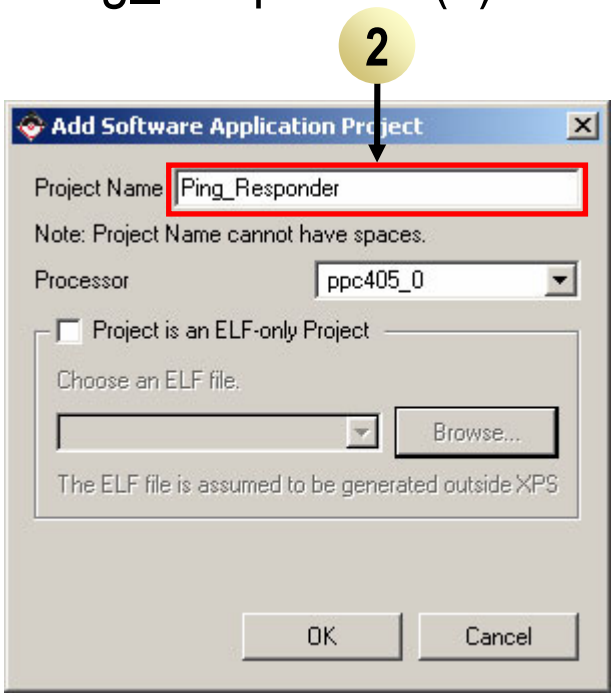
- Delete the TriMode interrupt



Adding Applications

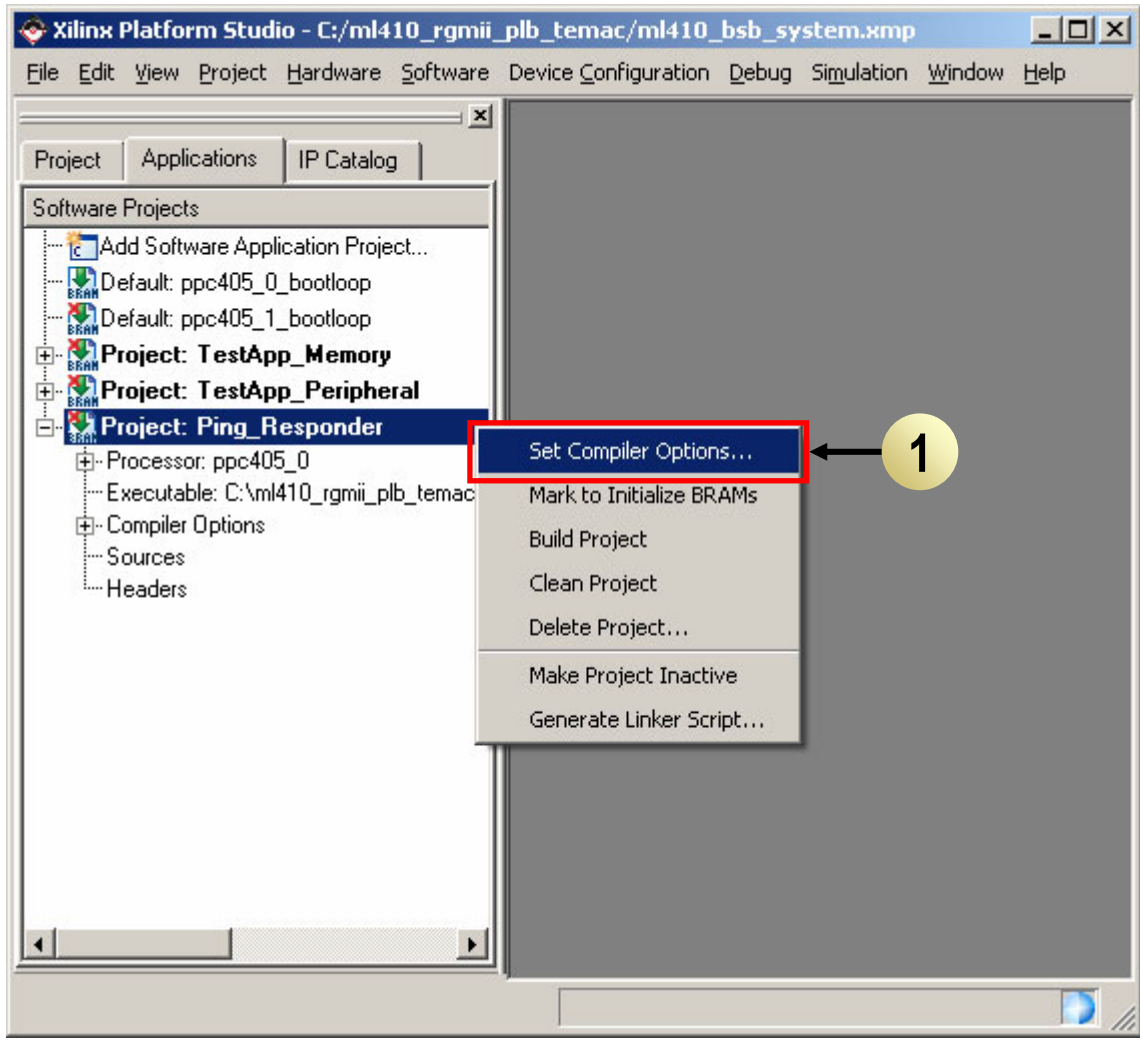
- Right-click on Software Projects and select **Add Software Application Project** (1)
- Name this project Ping_Responder (2)





Adding Applications

- Right-click on Ping_Responder and select **Set Compiler Options…** (1)

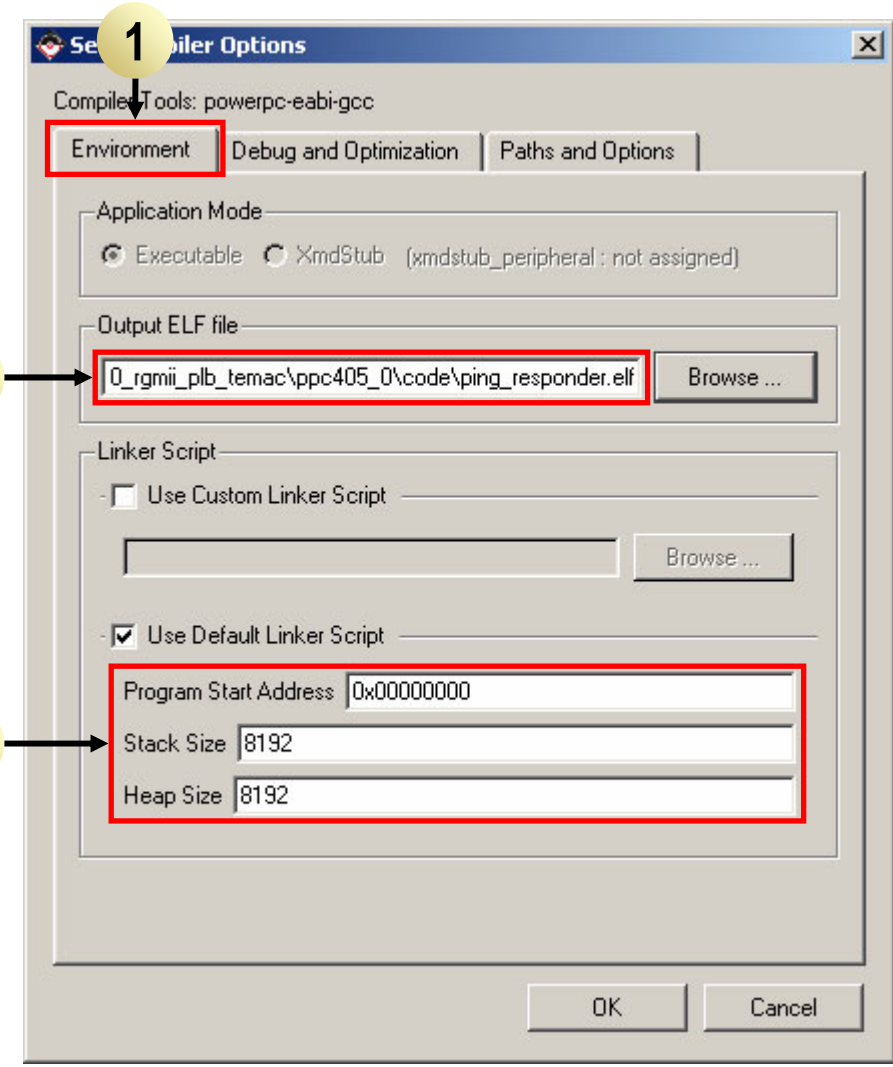


Adding Applications

- Under the Environment tab (1):
- Set the ELF output file to (2):

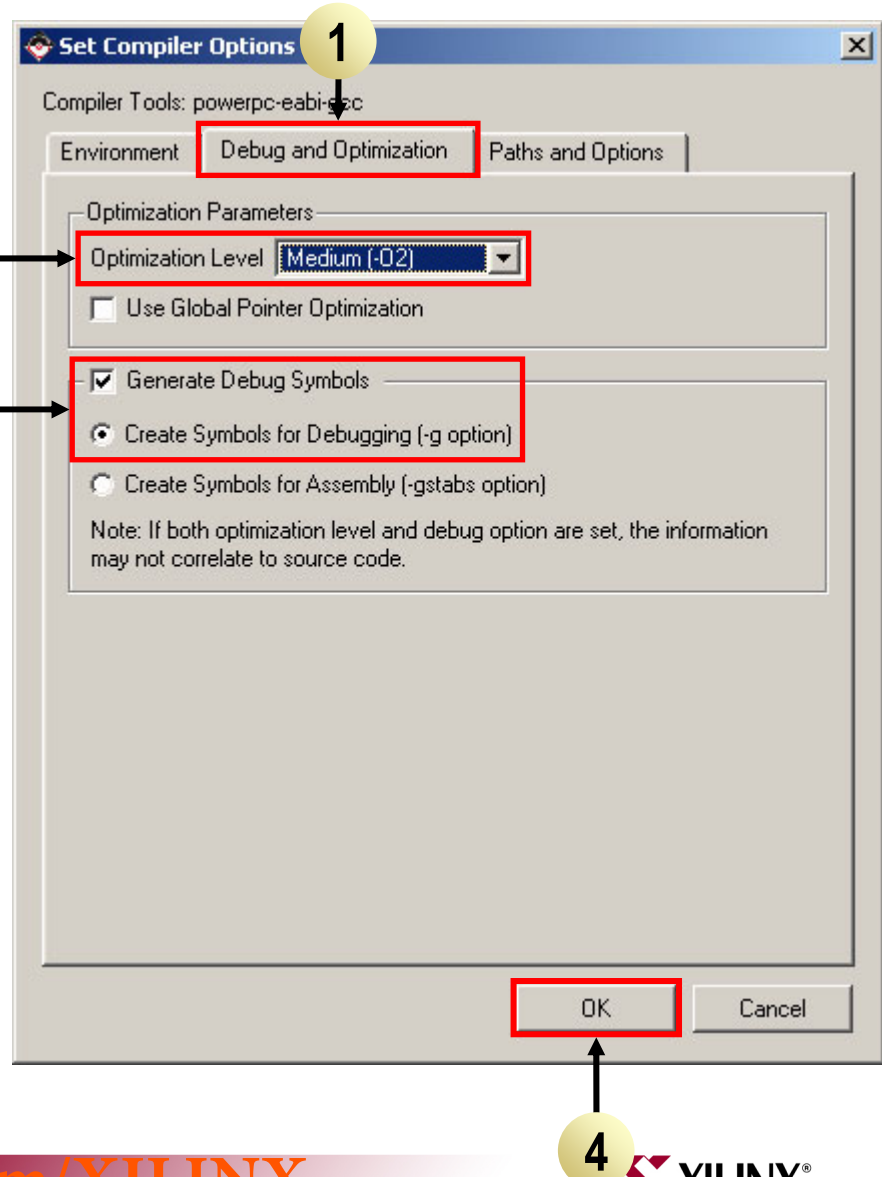
  <design path>\ppc405_0\code\ping_responder.elf
- Set Program Start Address to (3):

  0x00000000
- Set Stack and Heap to **8192** (3)



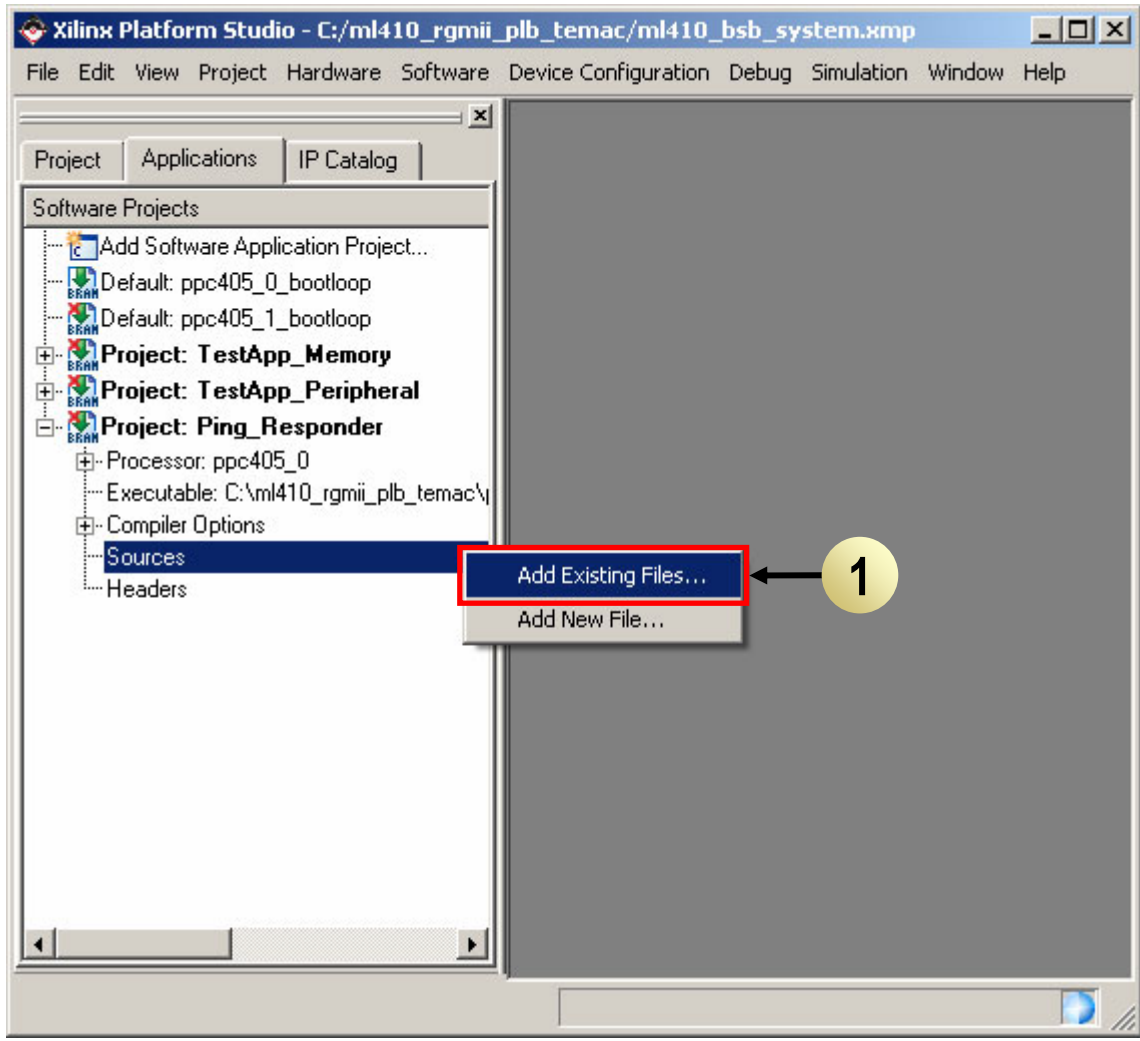
Adding Applications

- Under the Optimization tab (1):
- Set optimization to Medium (2)
- Set debugging options to: Create Symbols for Debugging (-g option) (3)
- Click OK (4)



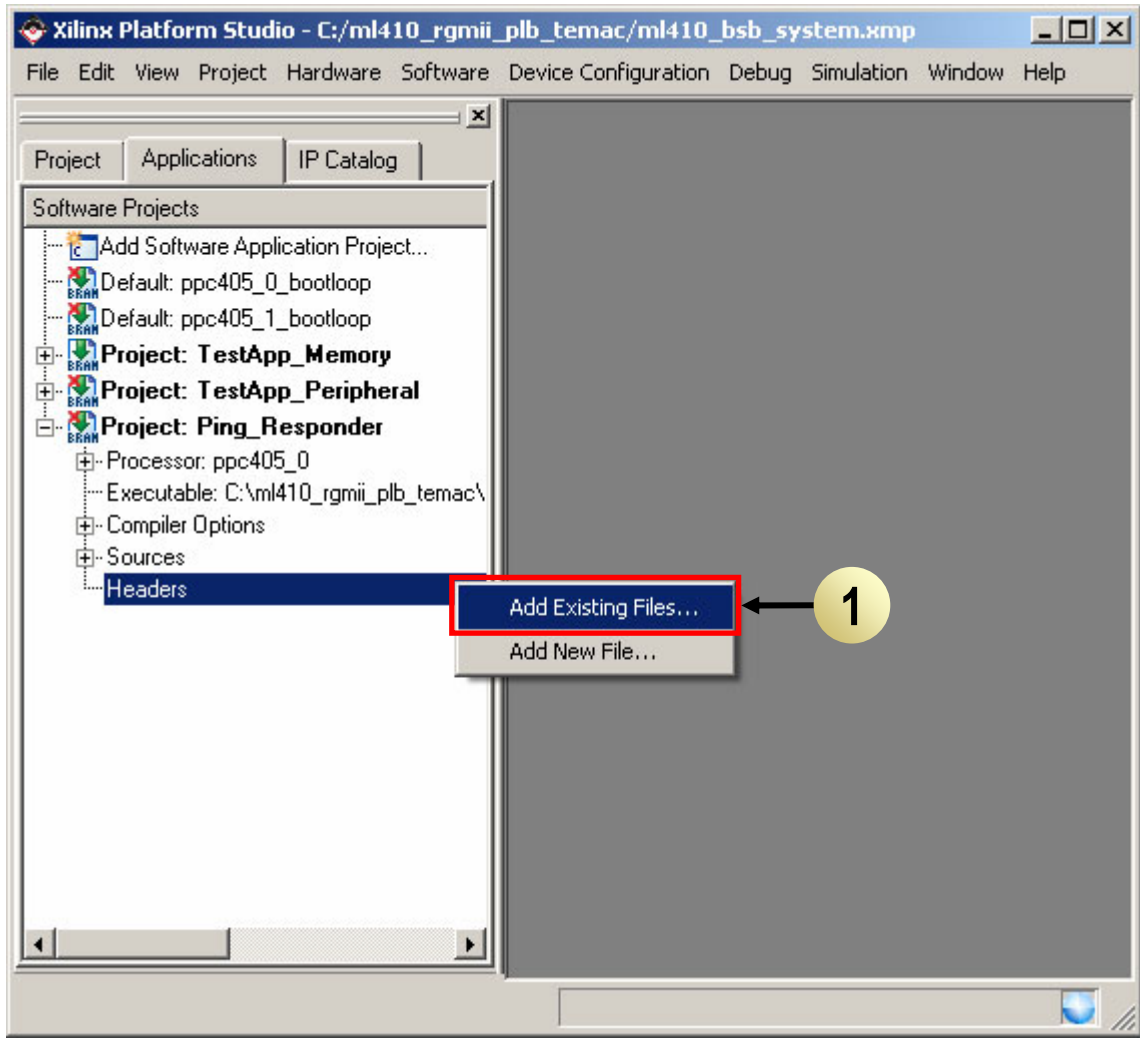
Adding Applications

- In the <design path>\sw\standalone\Ping_Responder\ directory, add these source files to the project (1):
  - ethernet.c
  - mdio.c
  - ping_responder.c
  - serial_io.c



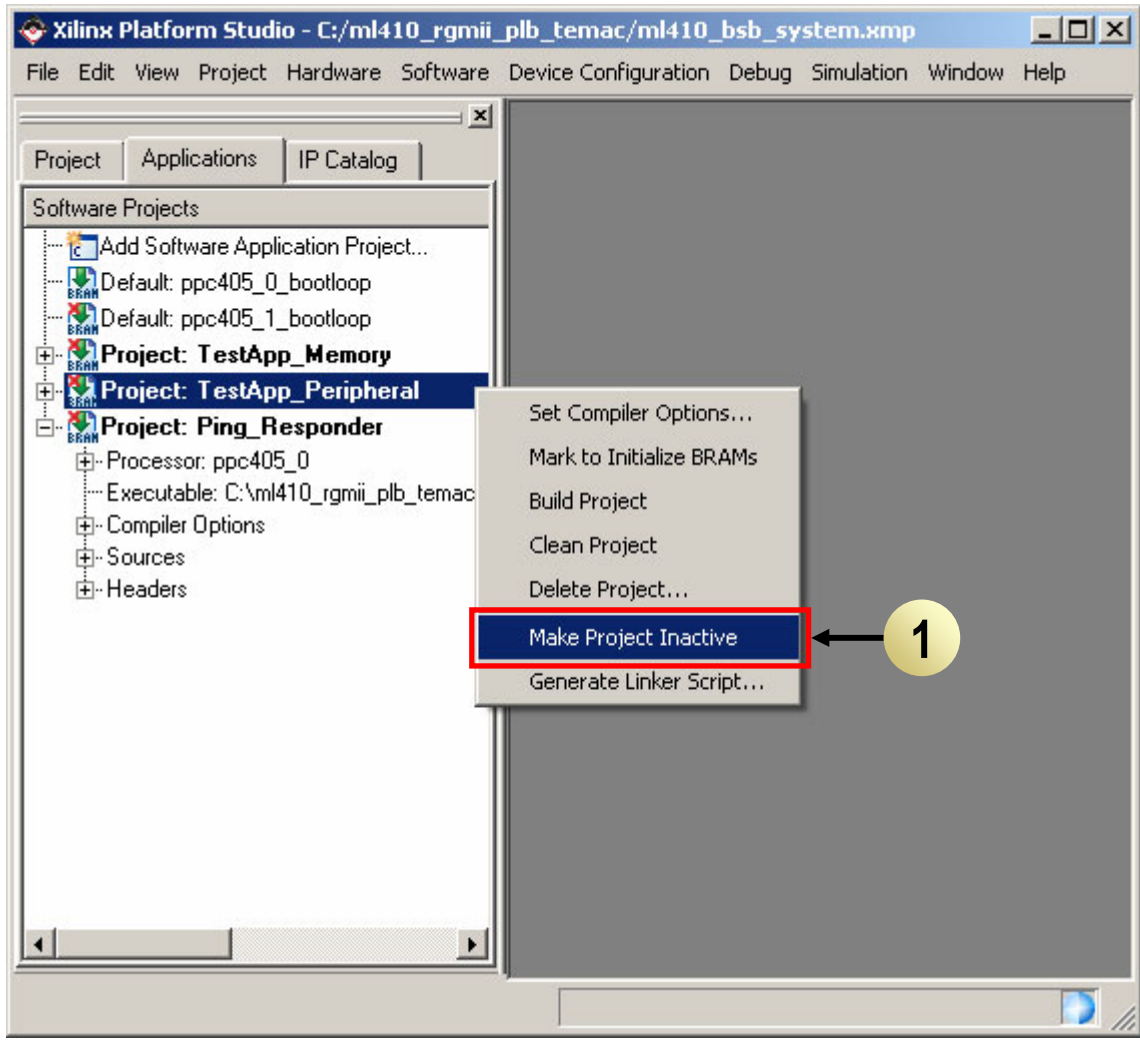
Adding Applications

- In the <design path>\sw\standalone\Ping_Responder\ directory, add these header files to the project (1):
  - ethernet.h
  - ping_responder.h
  - serial_io.h



Adding Applications

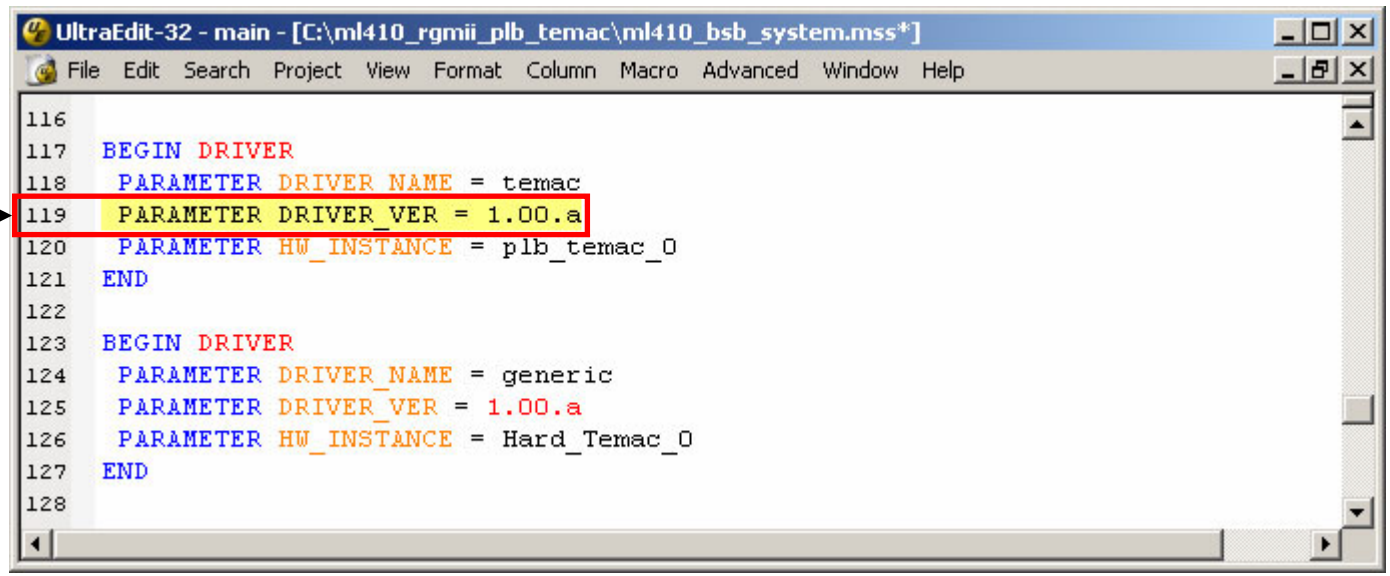
- Right-click on TestApp_Peripheral and select **Make Project Inactive** (1)



Update MSS File

- Close the EDK Project prior to making this edit
  - Reopen the EDK Project when finished
- Change the plb_temac driver version:
  - From PARAMETER DRIVER_VER = 2.00.a
  - To PARAMETER DRIVER_VER = 1.00.a

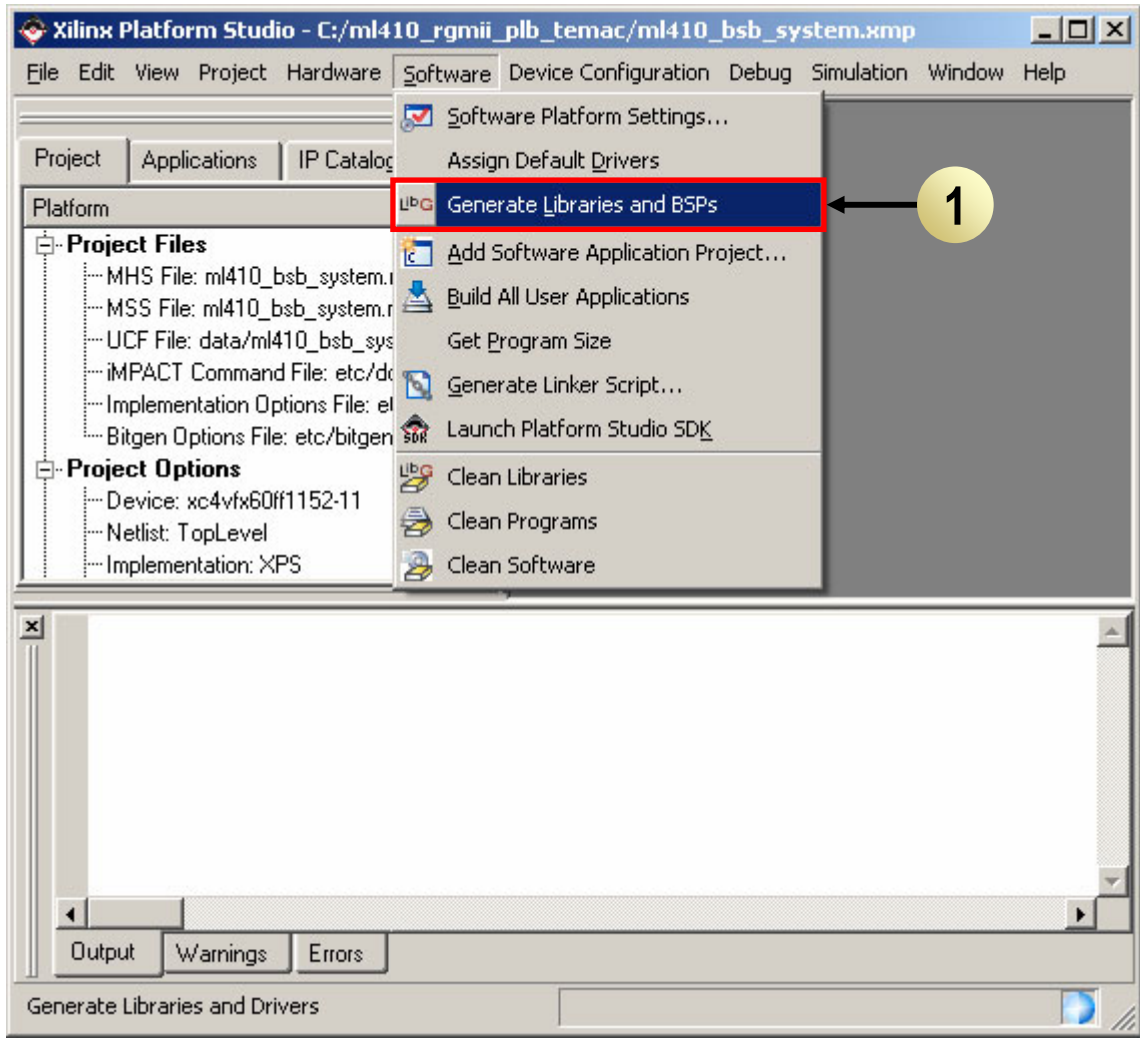
1



XILINX®

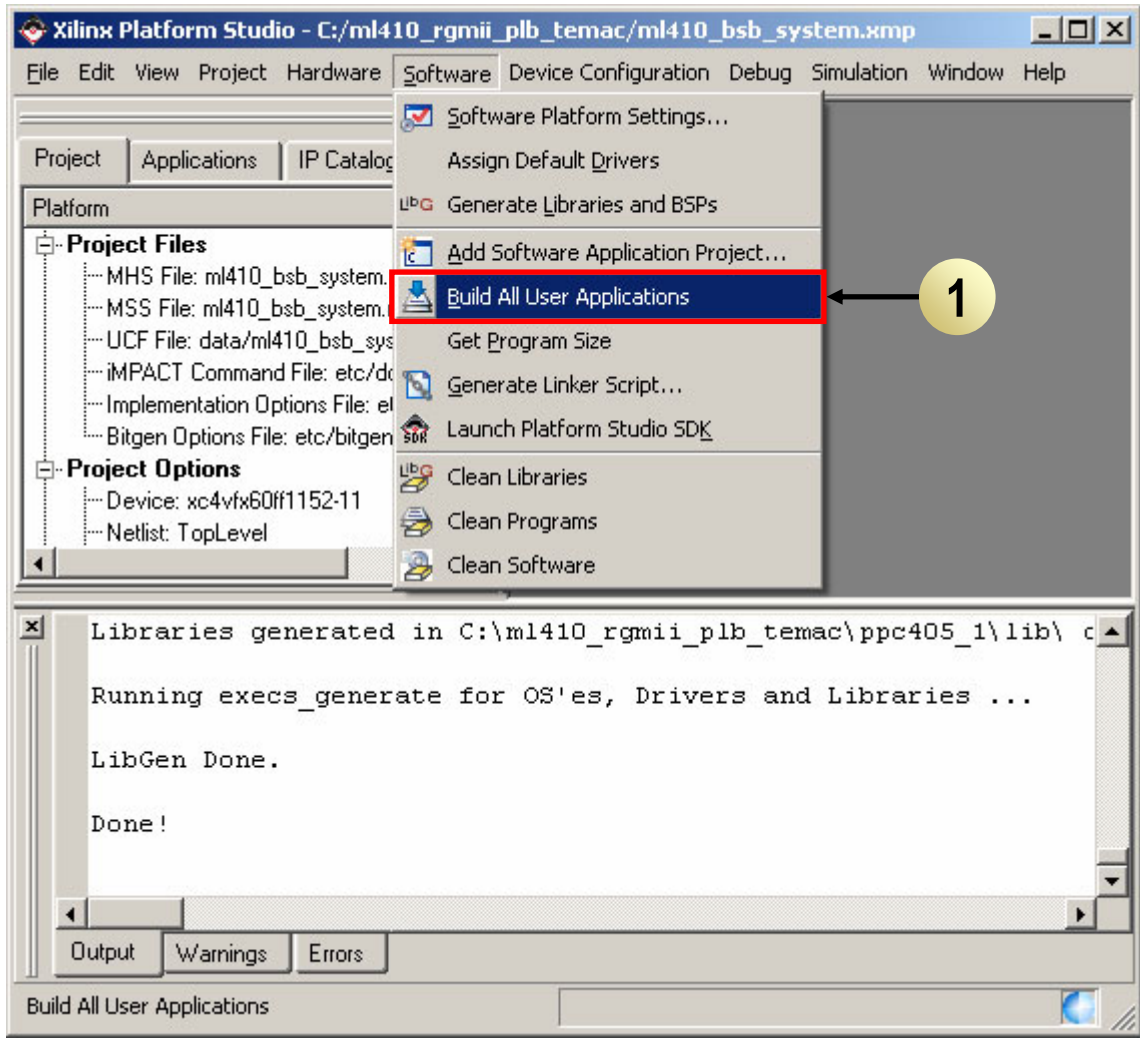
Generate Bitstream

- Generate the libraries needed to create the bitstream
  - Select **Software → Generate Libraries and BSPs** (1)



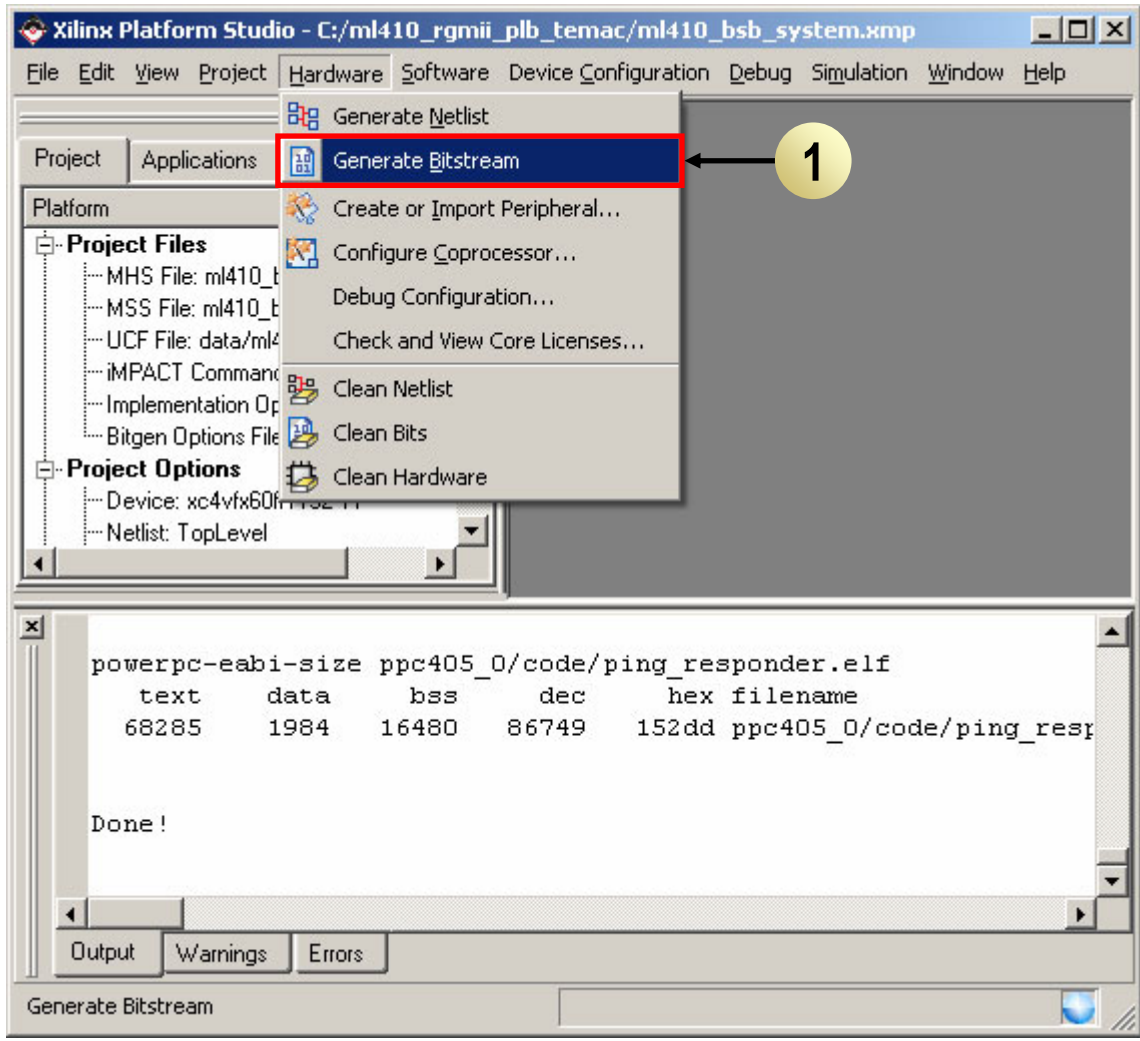
Generate Bitstream

- Compile the applications and create an executable (executable.elf)
  - Select **Software → Build All User Applications** (1)



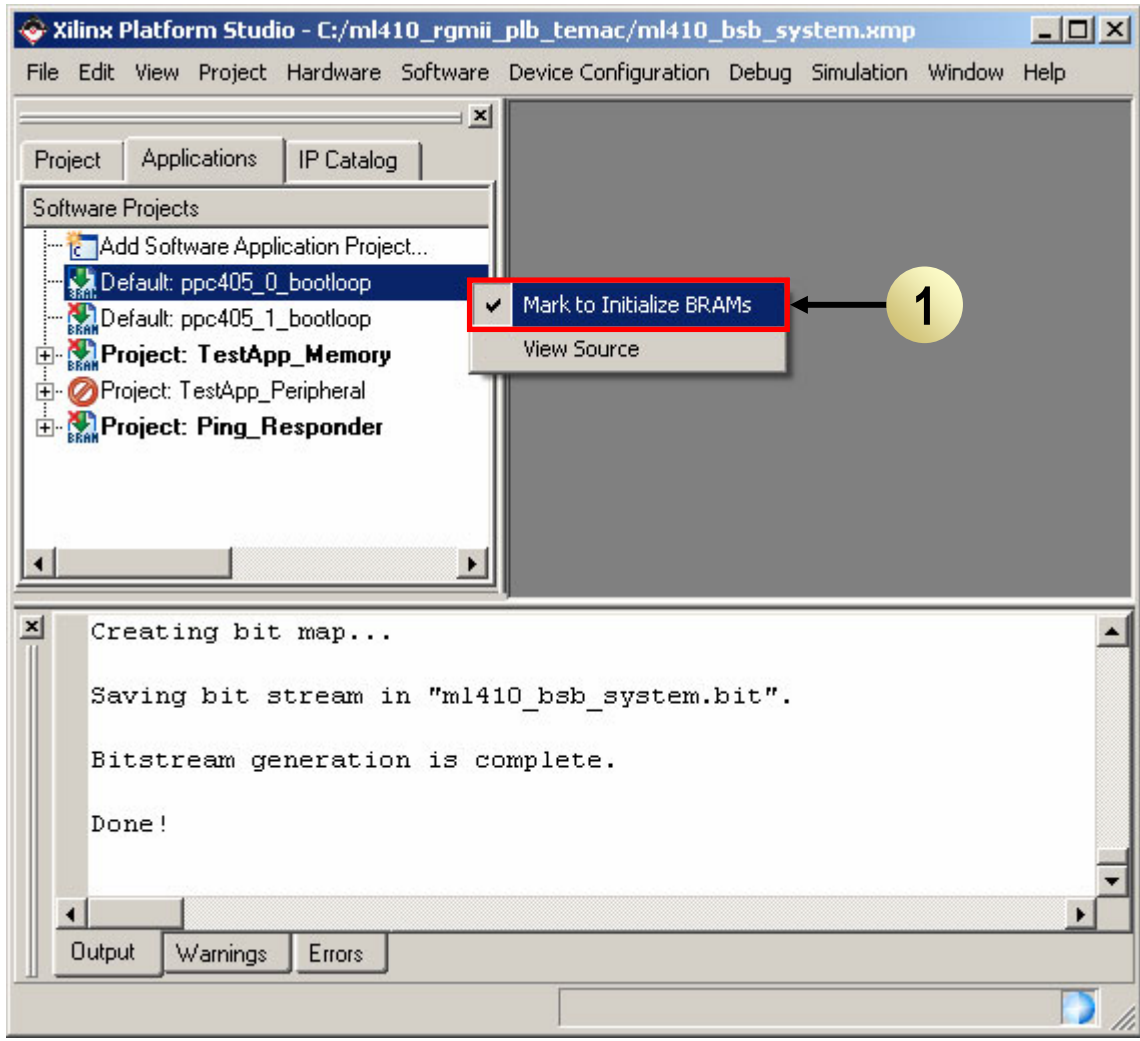
Generate Bitstream

- Create the hardware design (system.bit) that is located in <project directory> /implementation
  - Select **Hardware** → **Generate Bitstream** (1) (Takes roughly 90 minutes)



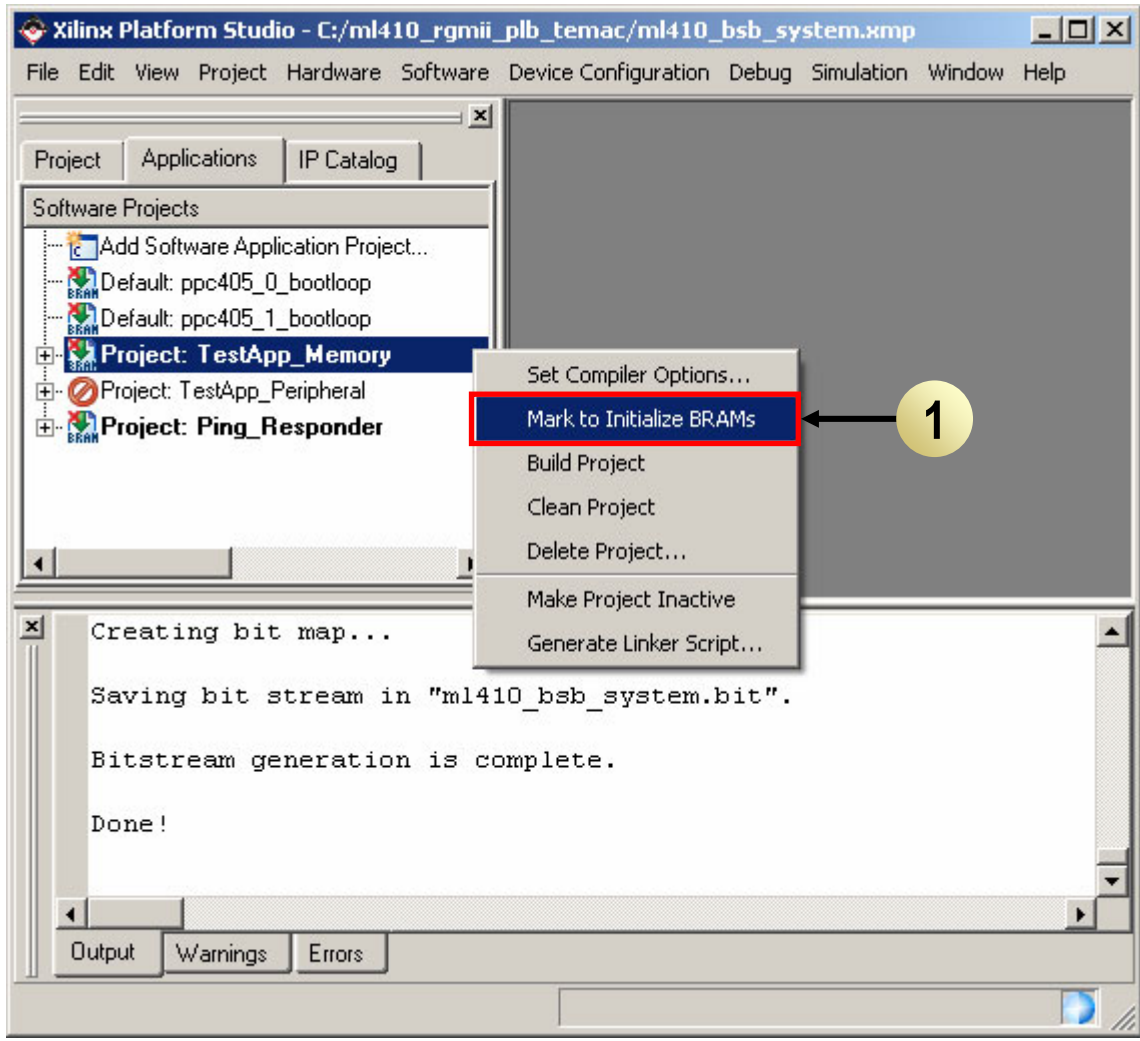
Download the Bitstream

- Right-click the ppc405_0_bootloop project and de-select **Mark to Initialize BRAMs** (1)



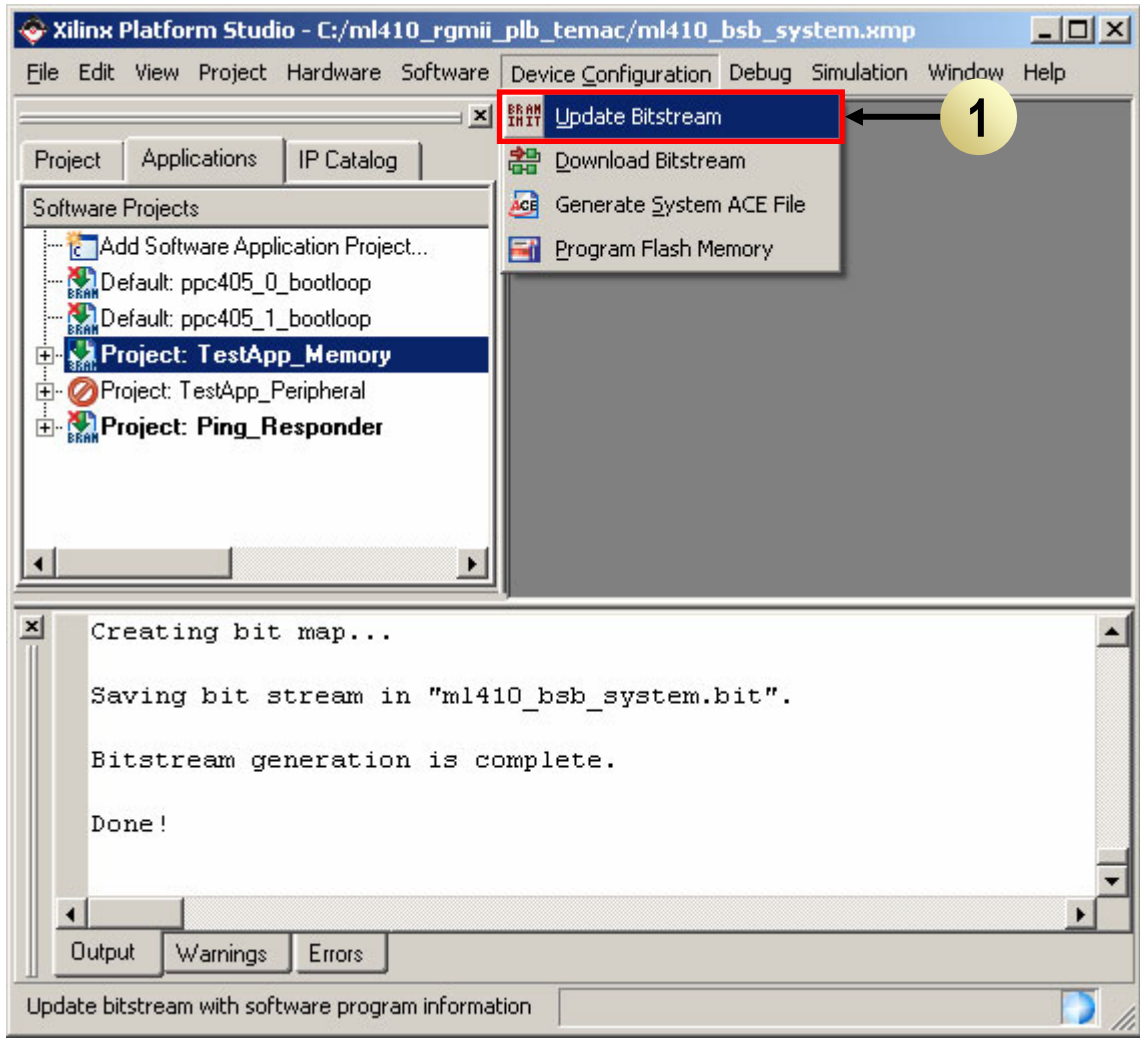
Download the Bitstream

- Right-click the TestApp_Memory project and select **Mark to Initialize BRAMs** (1)
- Now the TestApp_Memory will be instantiated into the block RAM rather than the bootloop ELF



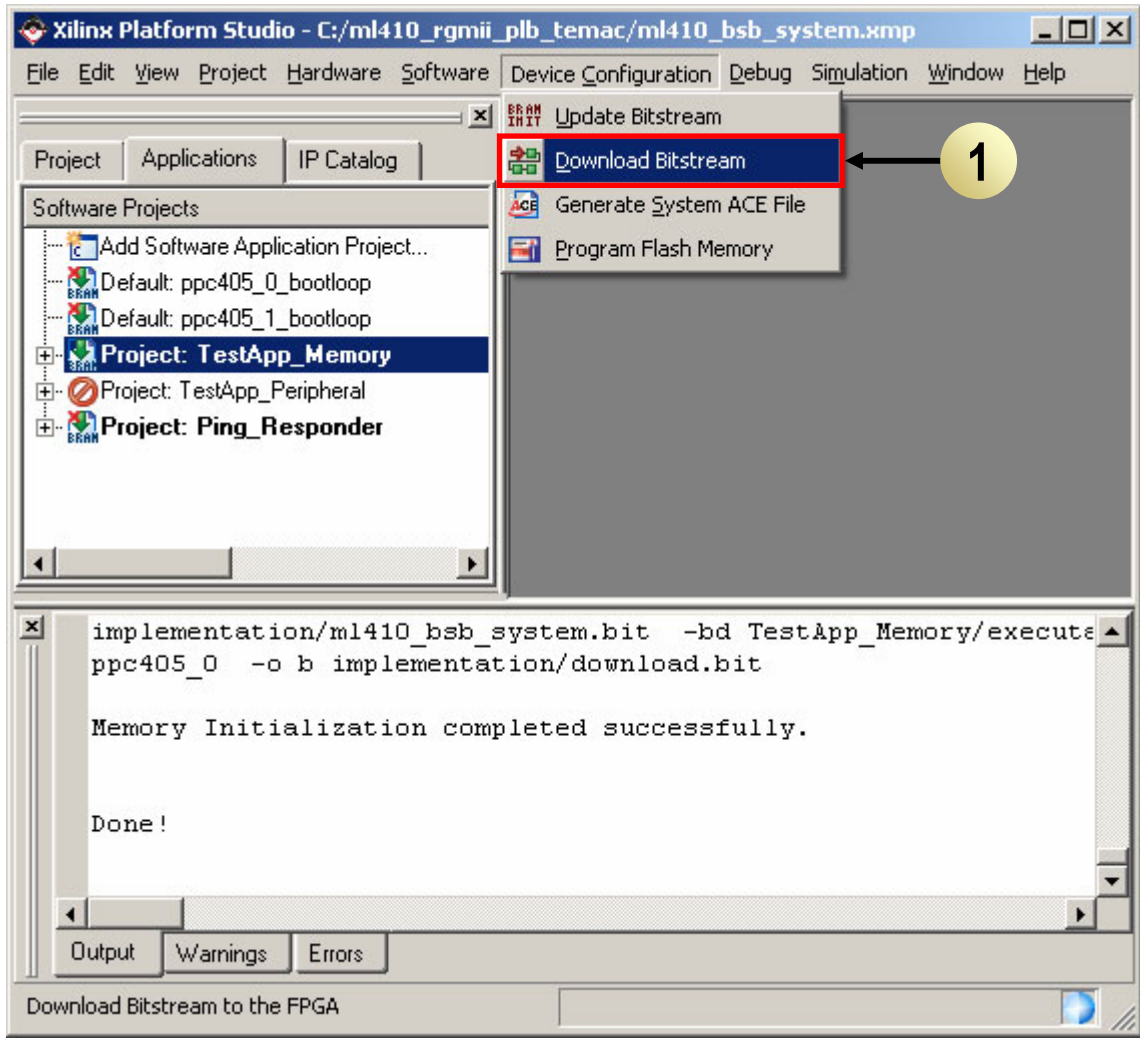
Download the Bitstream

- Update the bitstream (download.bit) with the TestApp_Memory ELF File
  - Select **Device Configuration → Update Bitstream** (1)



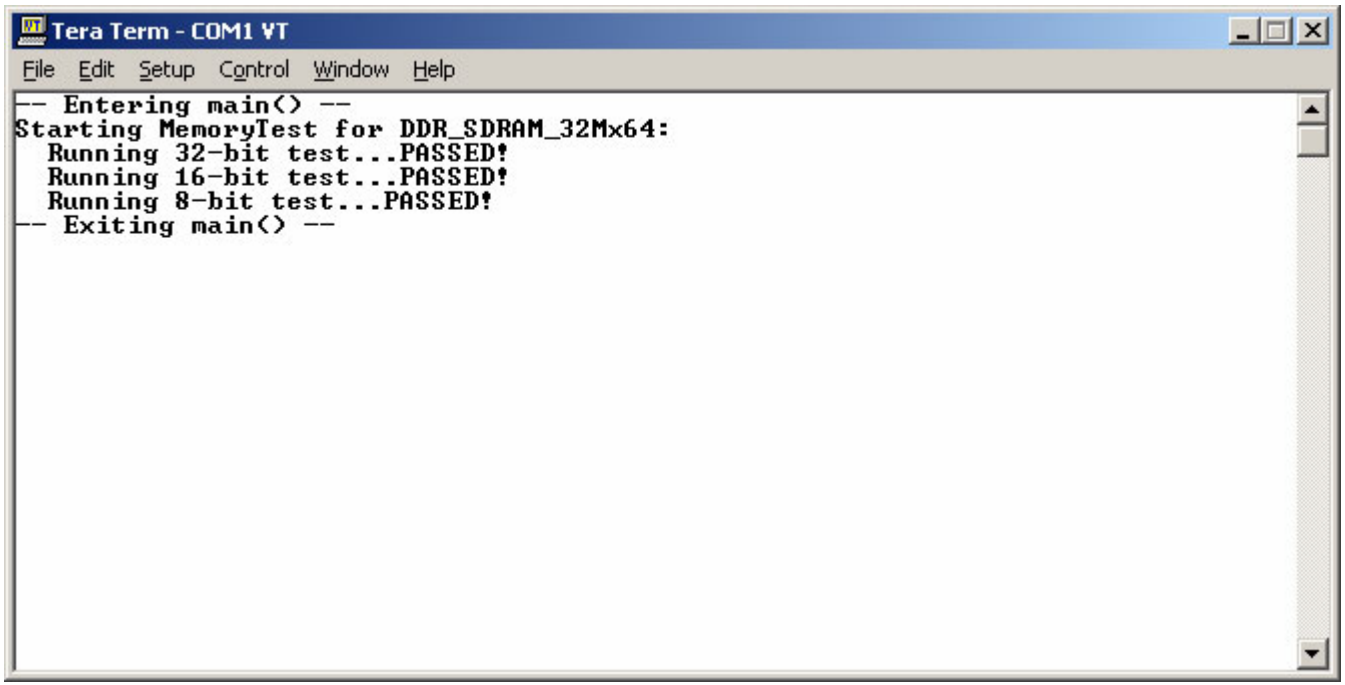
Download the Bitstream

- Download the new bitstream (download.bit)
  - Open a terminal program to view the output of the TestApp executable
  - Select **Device Configuration → Download Bitstream** (1)



Download the Bitstream

- View the output of a successful bitstream download in the terminal window

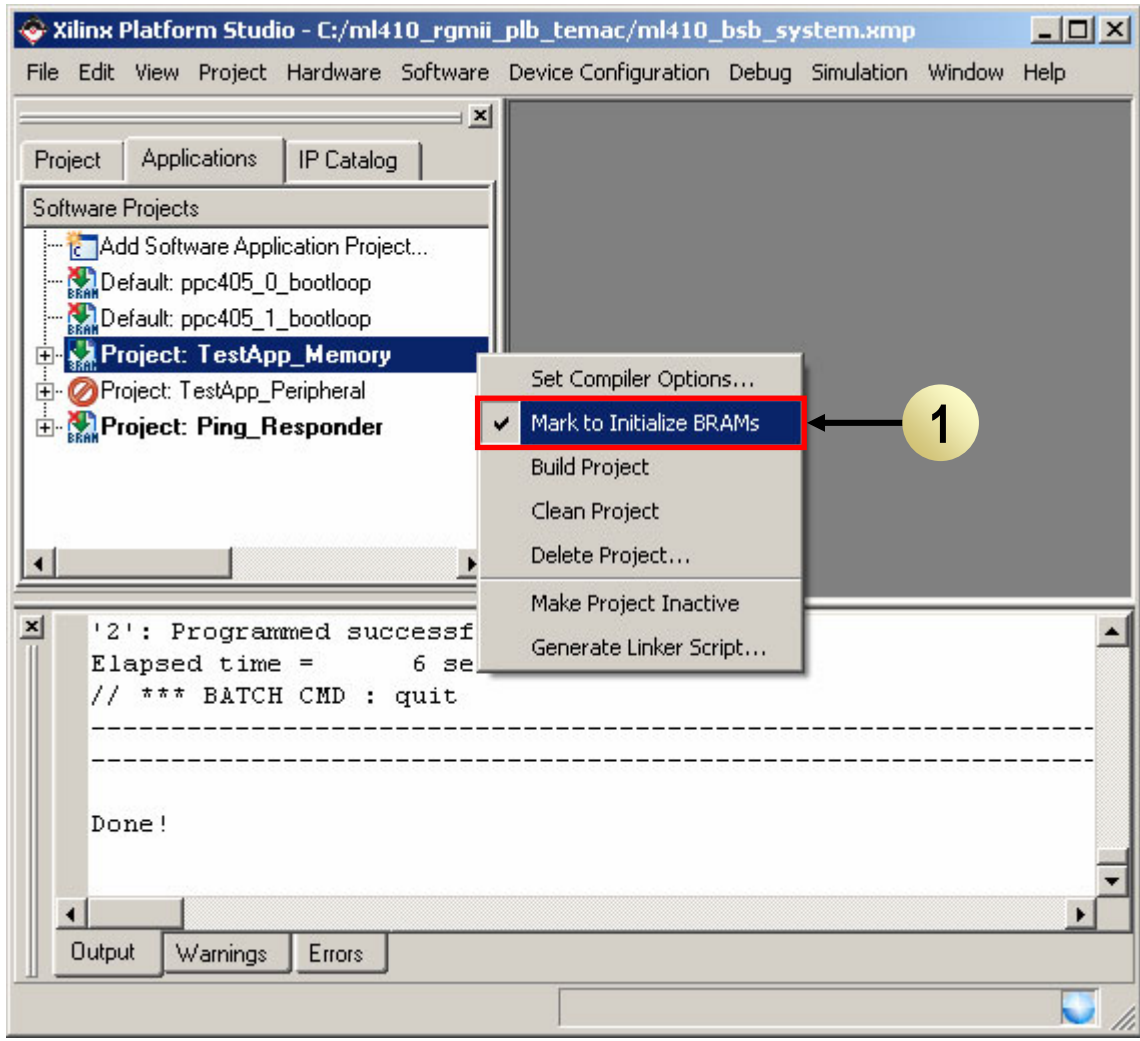


Loading Bootloop into BRAM

- A concatenated software/hardware file, known as an ACE file, is useful for loading large programs, such as a VxWorks or Linux demo, into the external memory
- A bootloop program must be used to occupy the processor until the software is loaded into memory
- The following pages show how to initialize a bootloop program into block RAM and to test its existence

Loading Bootloop into BRAM

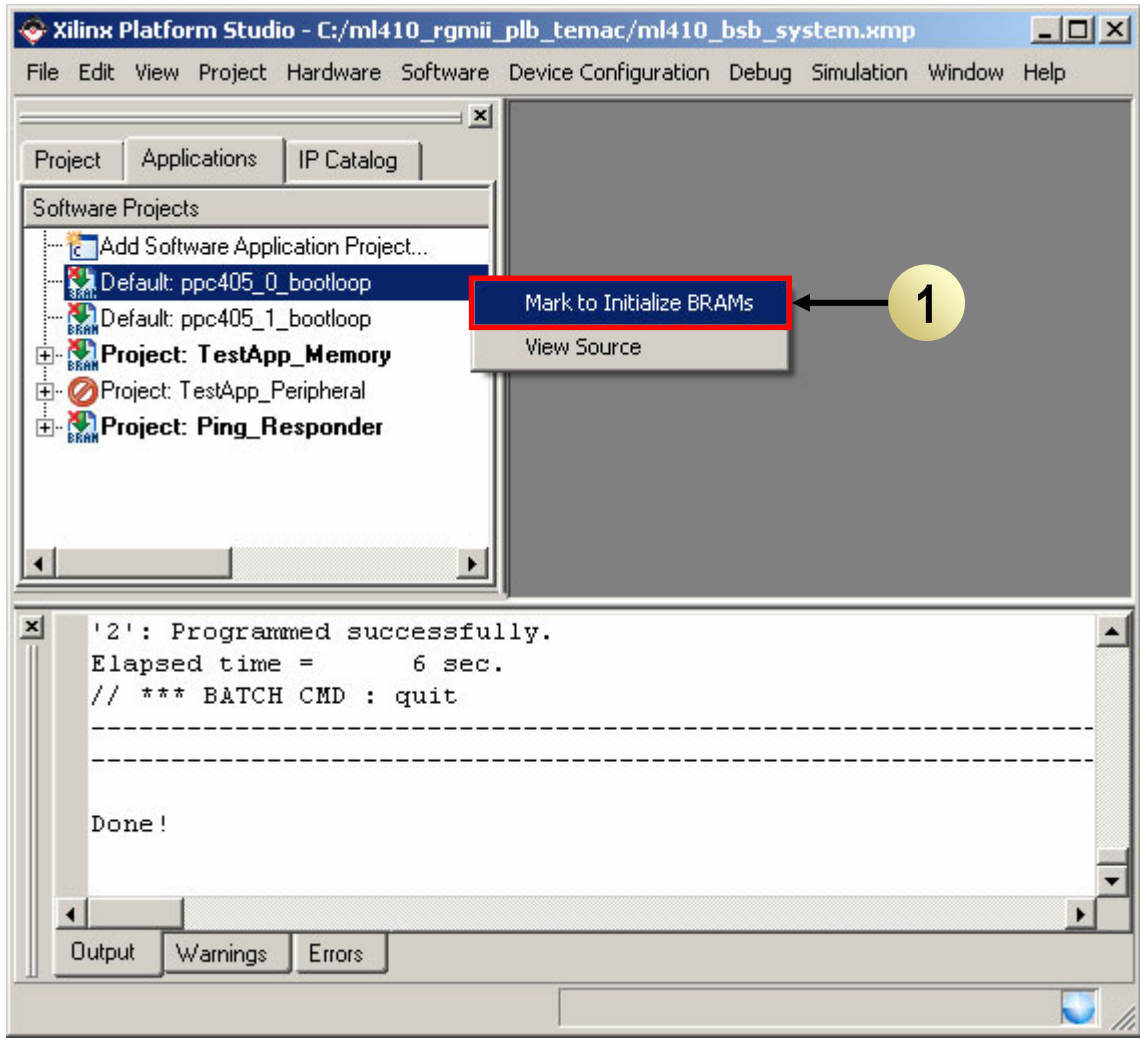
- Right-click the TestApp_Memory project and de-select **Mark to Initialize BRAMs** (1)
- This will prevent the TestApp program from being inserted into the block RAM when the new bitstream is created



XILINX®

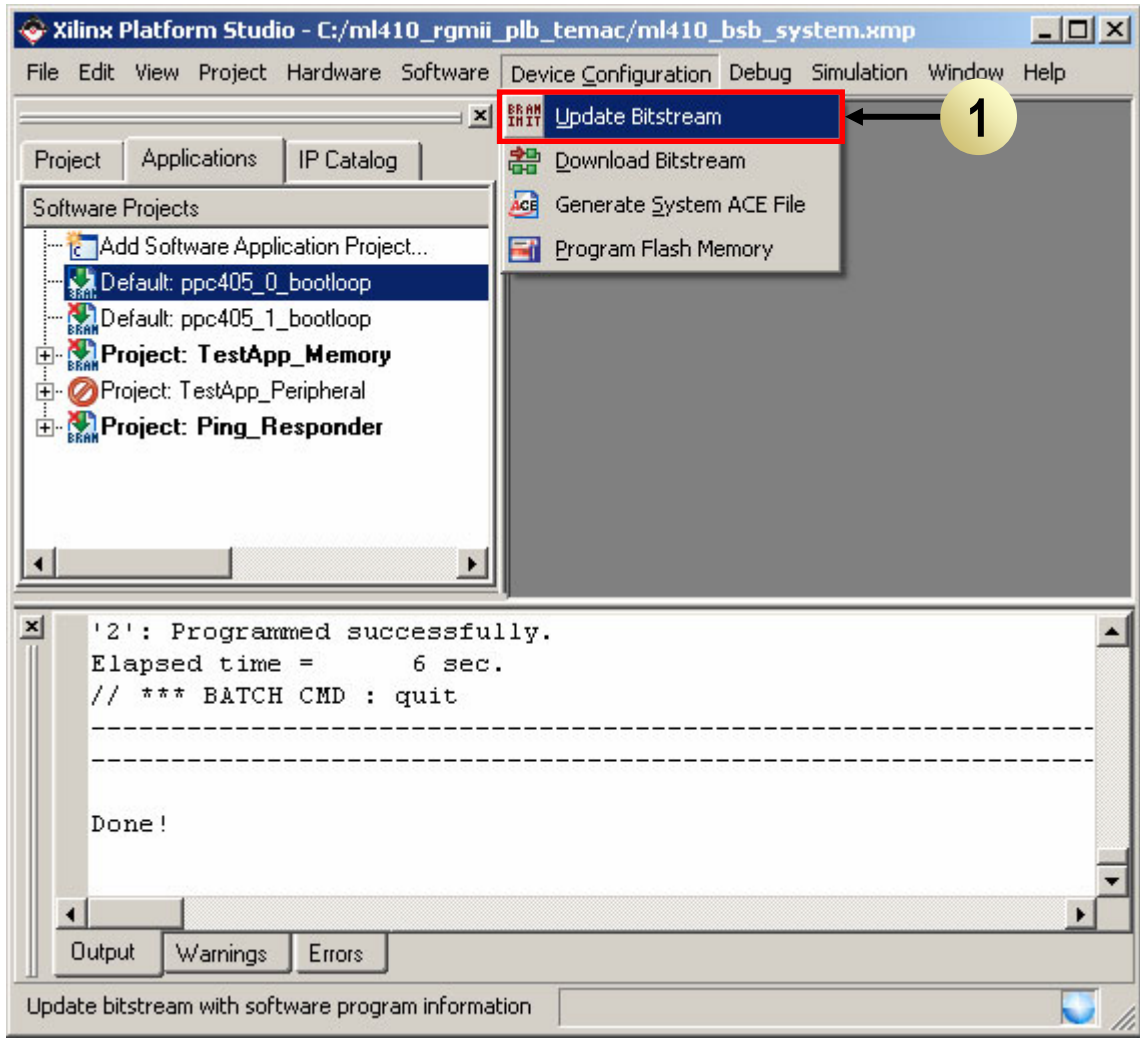
Loading Bootloop into BRAM

- Right-click the ppc405_0_bootloop project and select **Mark to Initialize BRAMs** (1)
- Now bootloop will be instantiated into the block RAM rather than the TestApp project



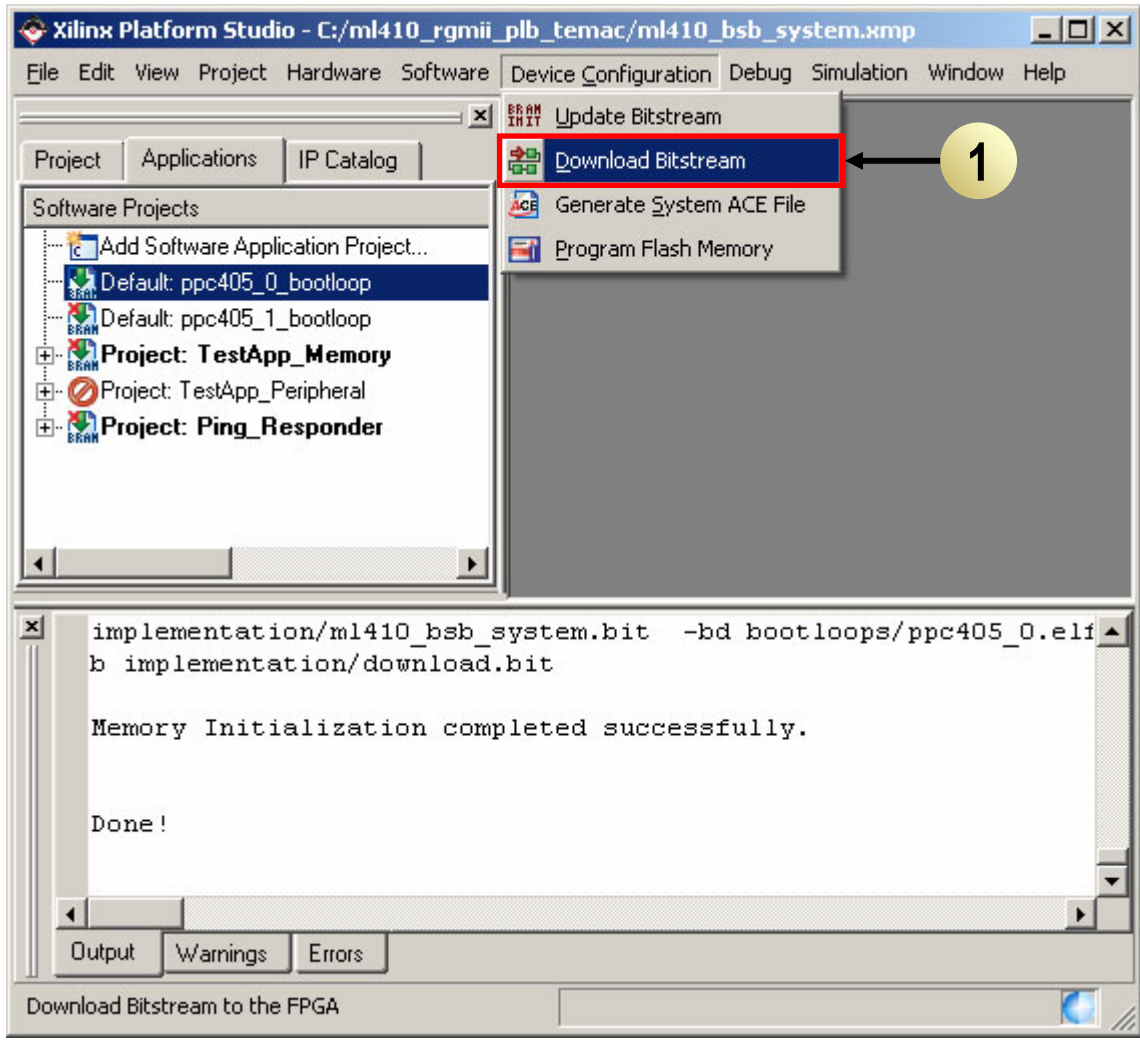
Loading Bootloop into BRAM

- Update the bitstream (download.bit) with a bootloop ELF file (ppc405_0.elf)
  - Select **Device Configuration → Update Bitstream** (1)



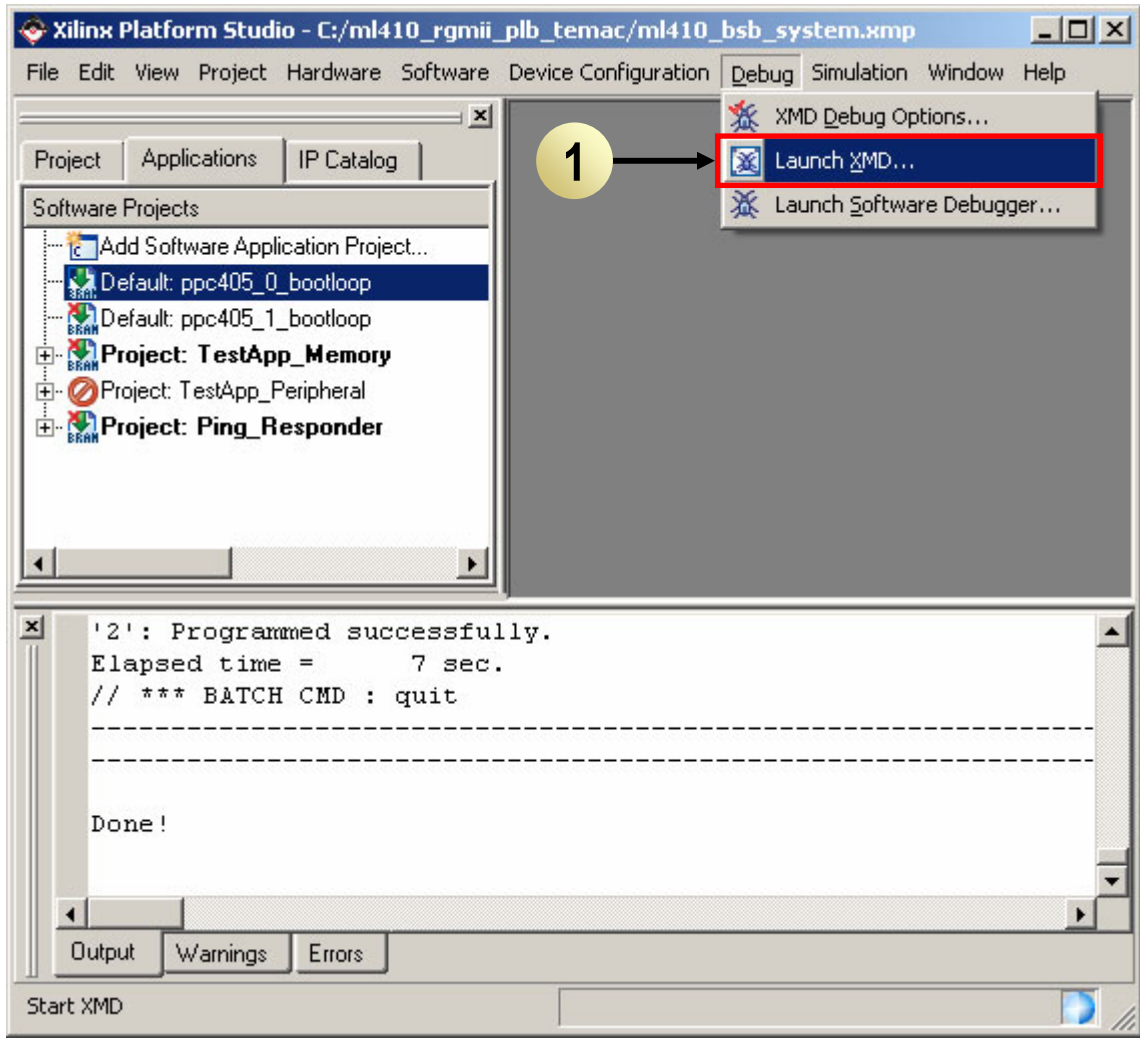
Loading Bootloop into BRAM

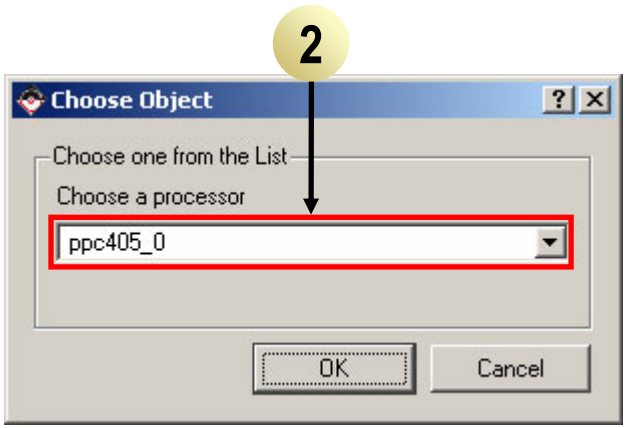
- Load the new design onto the FPGA and load the bootloop program into the block RAM
  - Select **Device Configuration → Download Bitstream** (1)



Loading Bootloop into BRAM

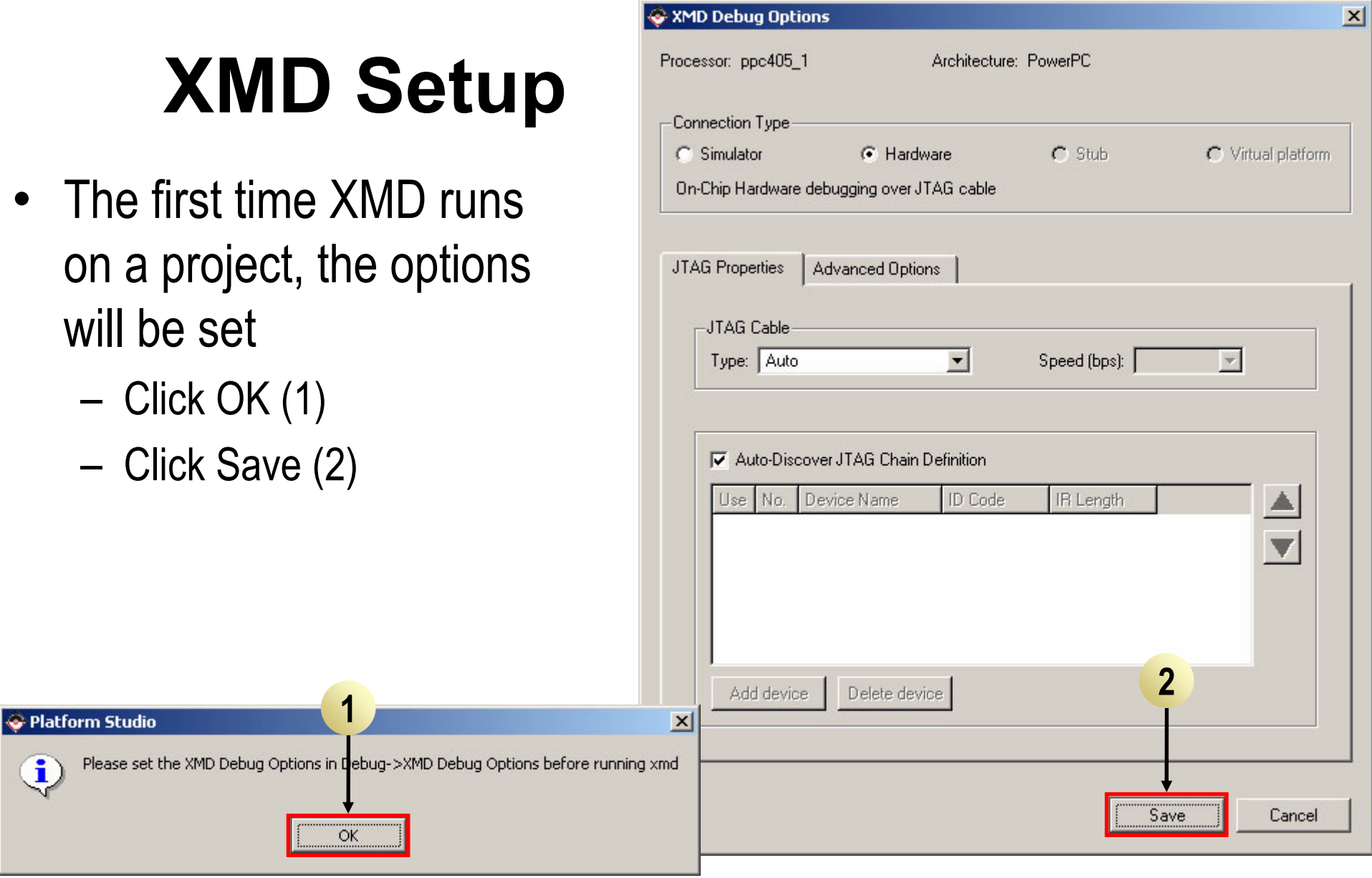
- A memory read can be executed to test if the bootloop was successfully loaded
  - Select **Debug → Launch XMD** (1)
  - Select ppc405_0 (2)





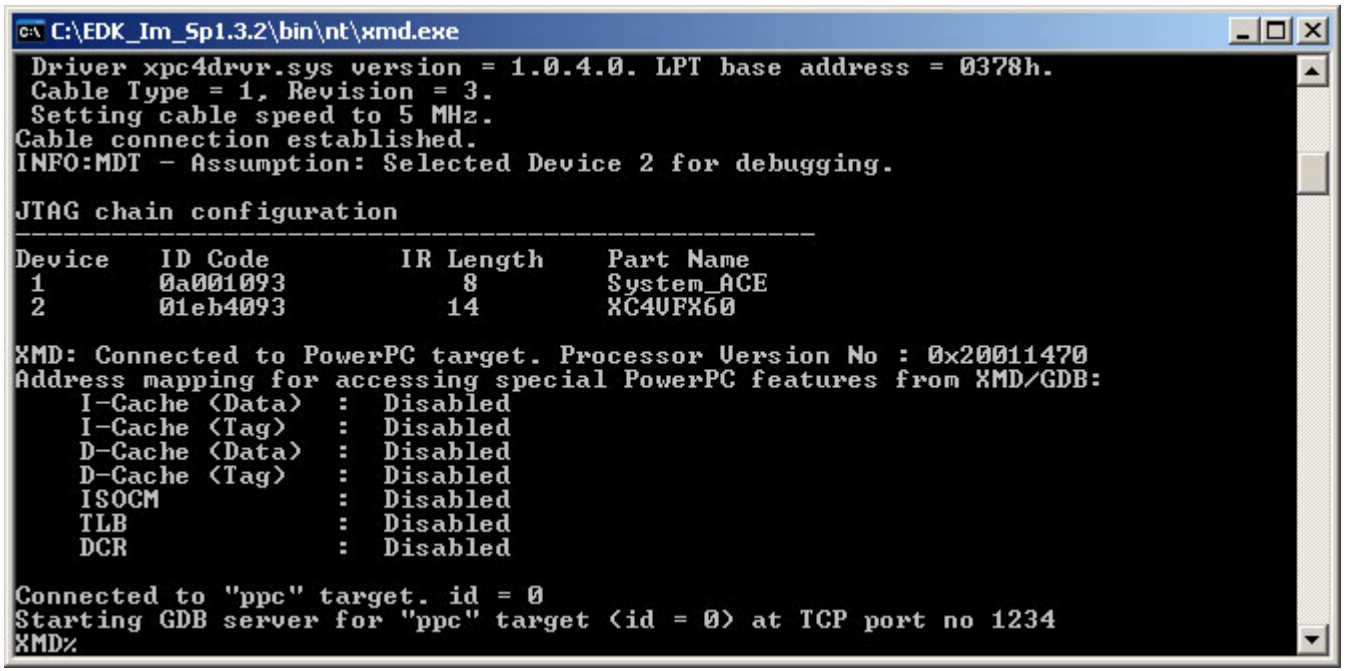
XMD Setup

- The first time XMD runs on a project, the options will be set
  - Click OK (1)
  - Click Save (2)



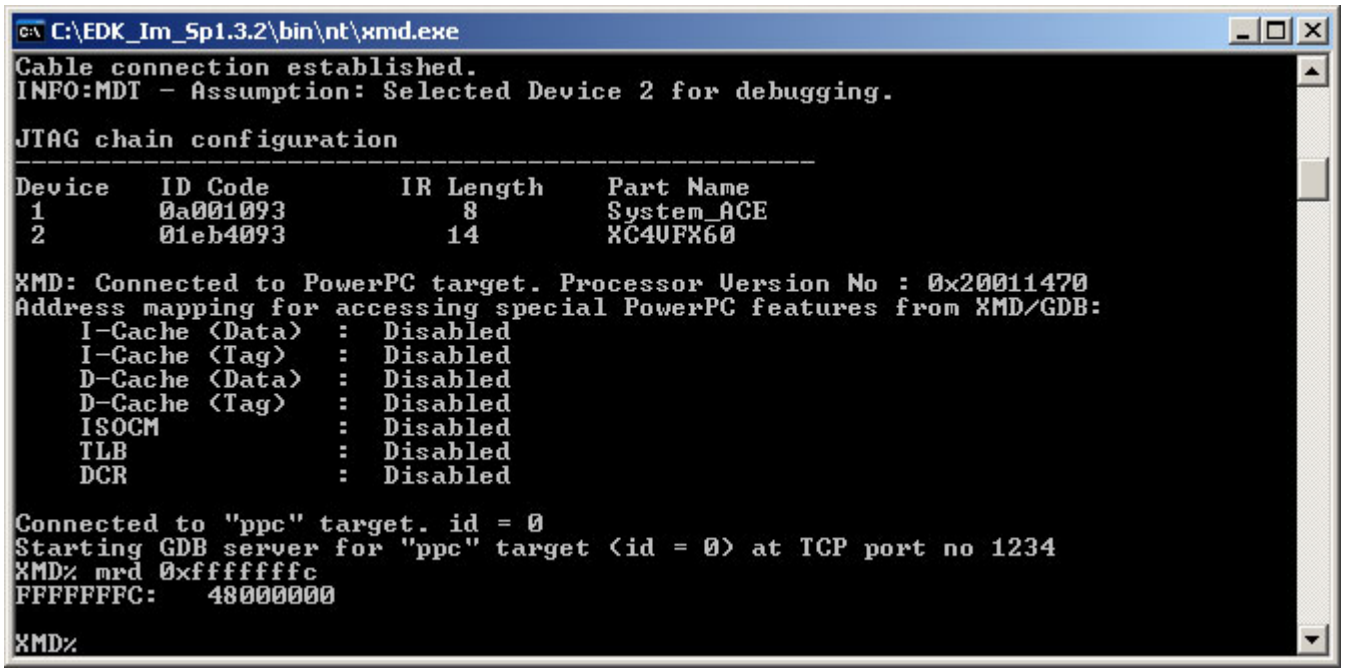
Loading Bootloop into BRAM

- XMD opens and connects to the processor, using the default options



Loading Bootloop into BRAM

- To execute a memory read, type **mrd 0xfffffffc**
- This will read the memory address at the reset vector; the value should be 0x48000000 as shown below

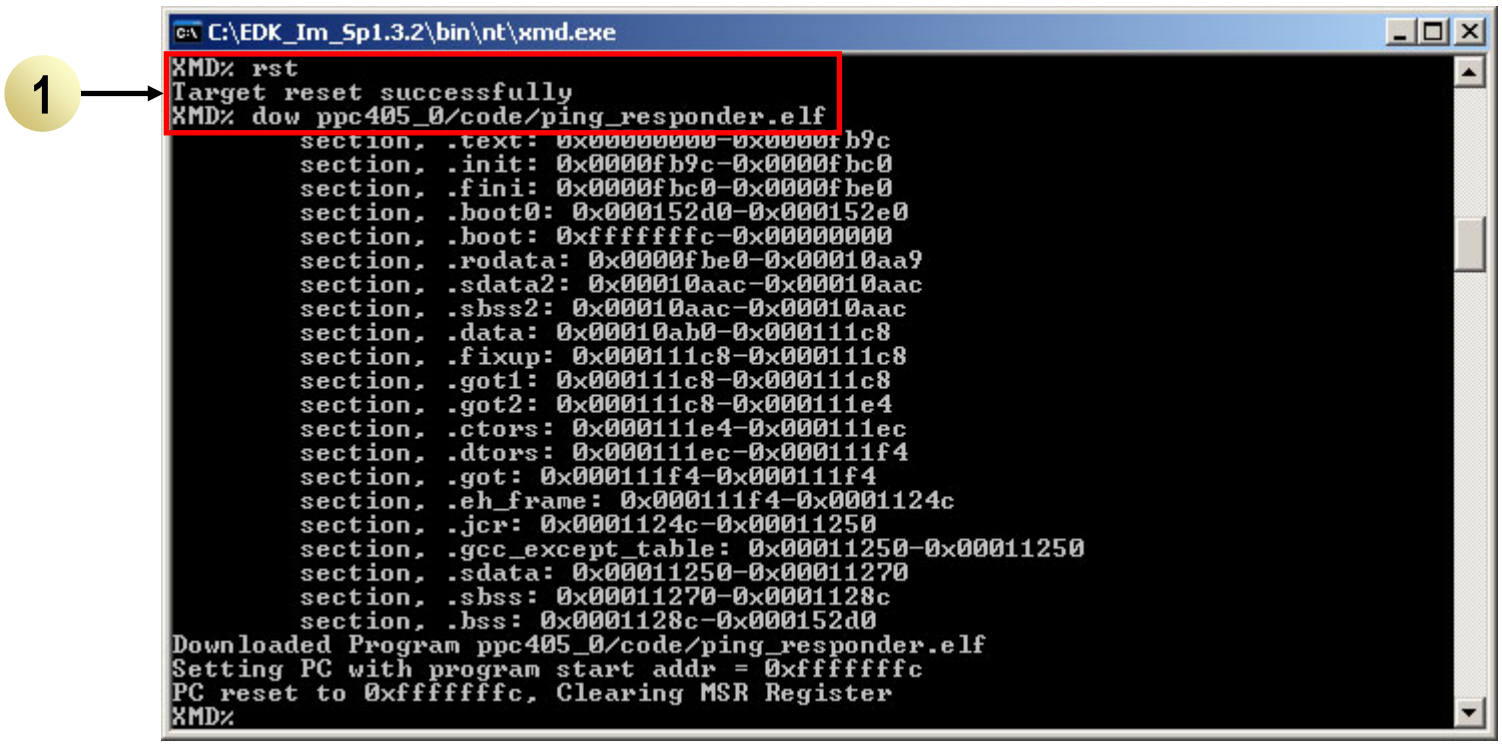


Download ELF File

- Download the ping_responder ELF file from XMD

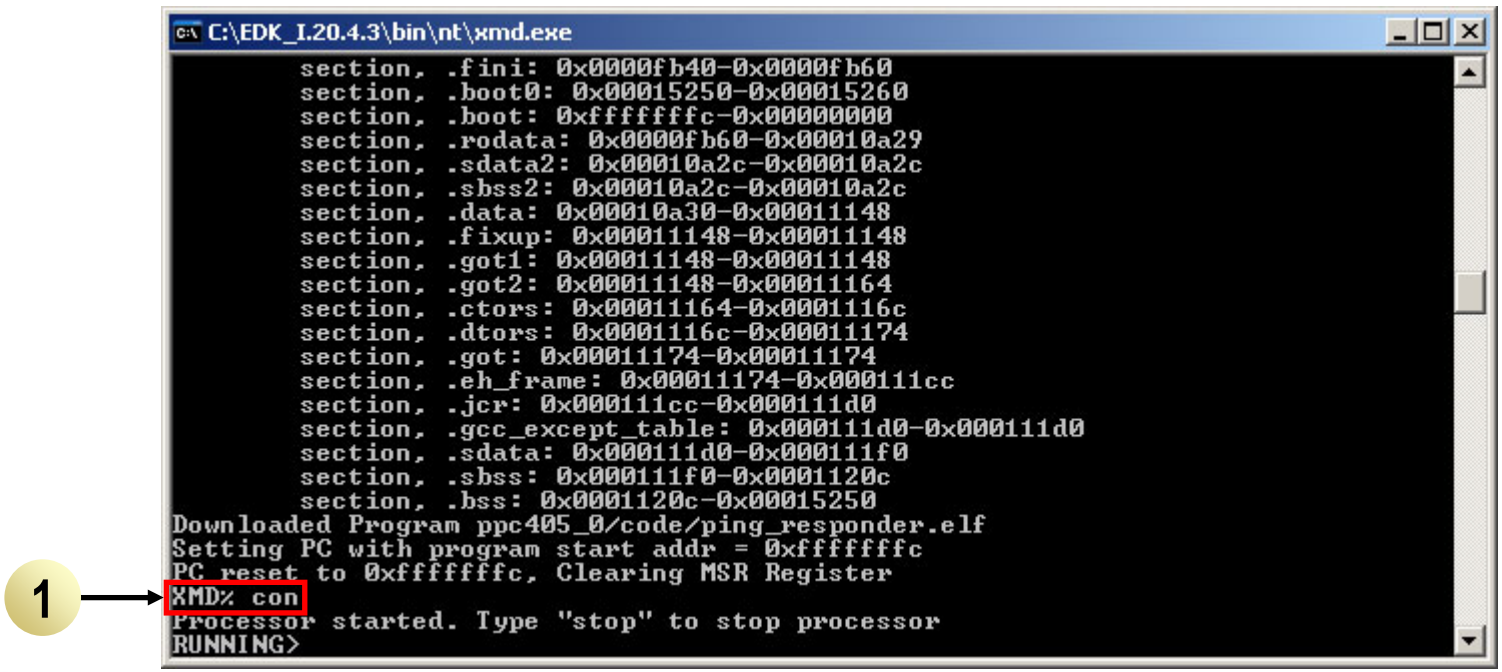
  rst

  dow ppc405_0/code/ping_responder.elf (1)



Run Ping Responder

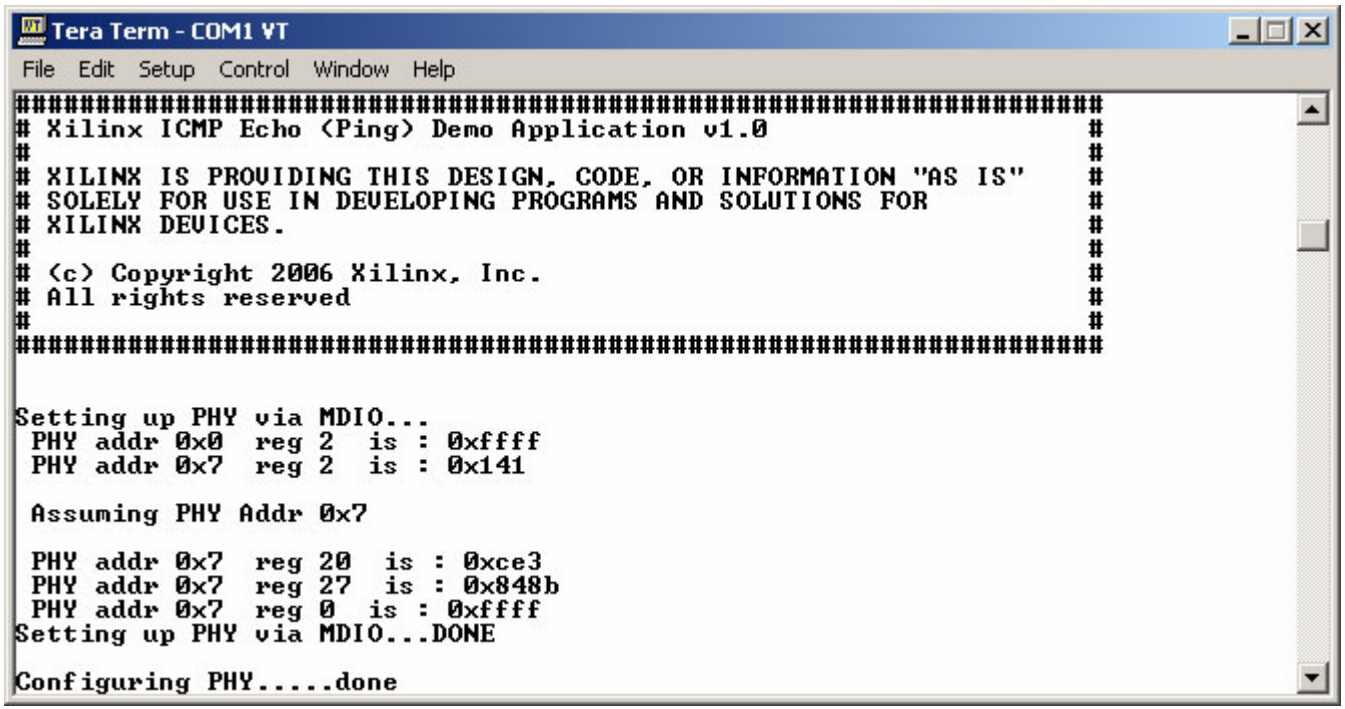
- Open a terminal program
- Enter **con** in the XMD window to start Ping Responder (1)



XILINX®

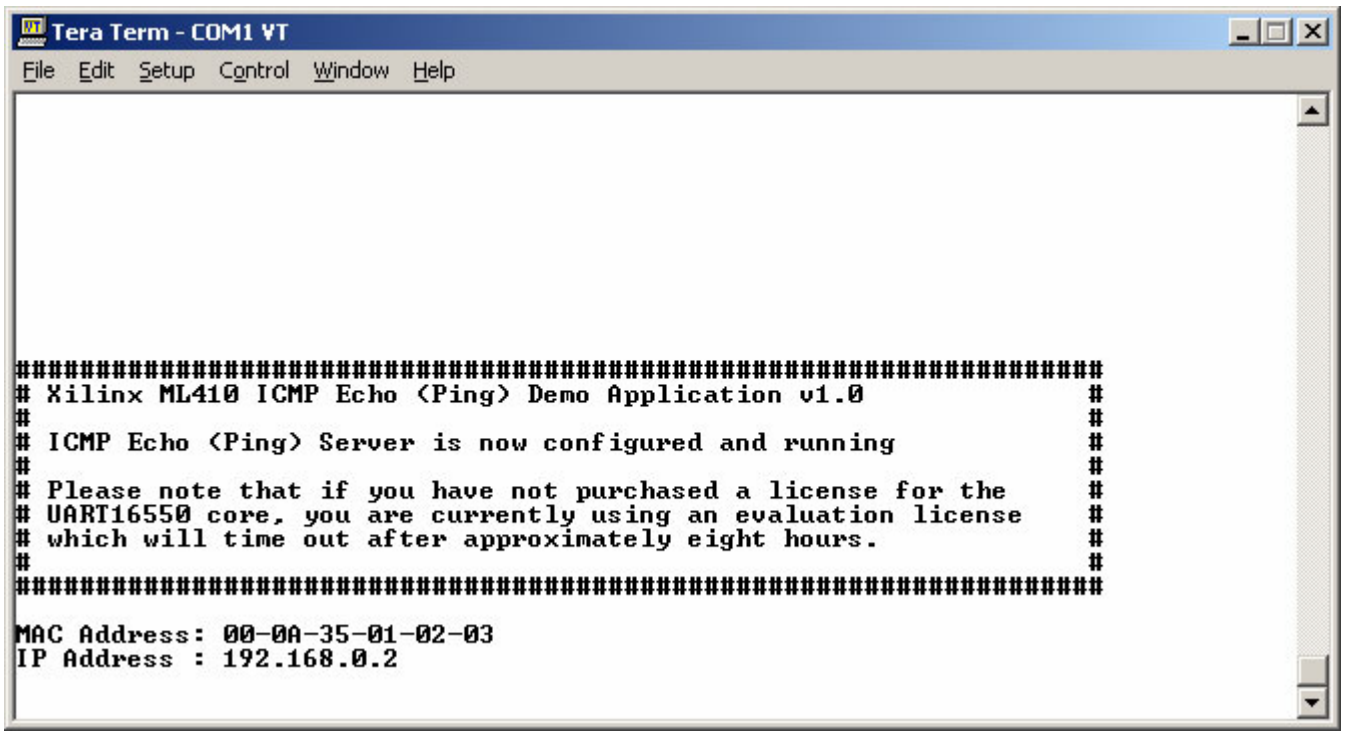
Run Ping Responder

- View the output in the terminal program
- Ping Responder is running in external memory of ML410 (below)



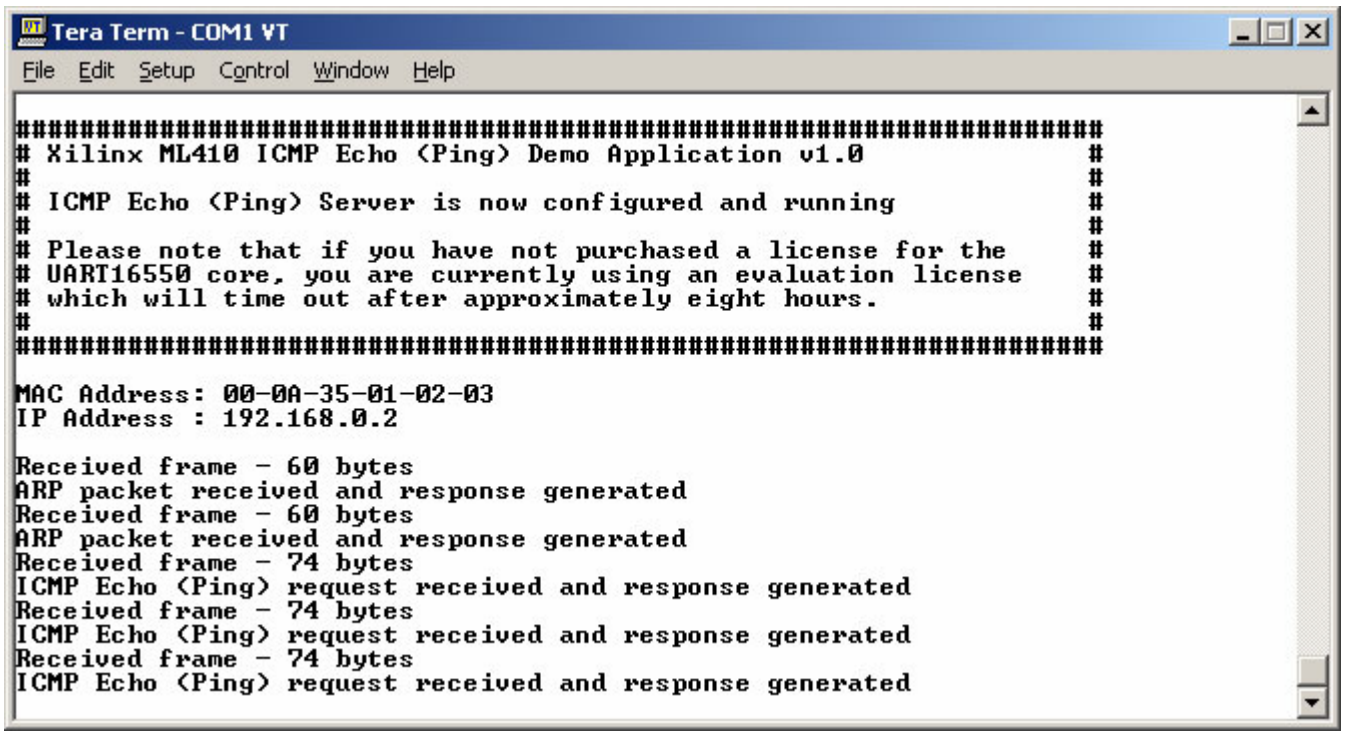
Run Ping Responder

- After setting up the PHY, the program awaits a network ping:



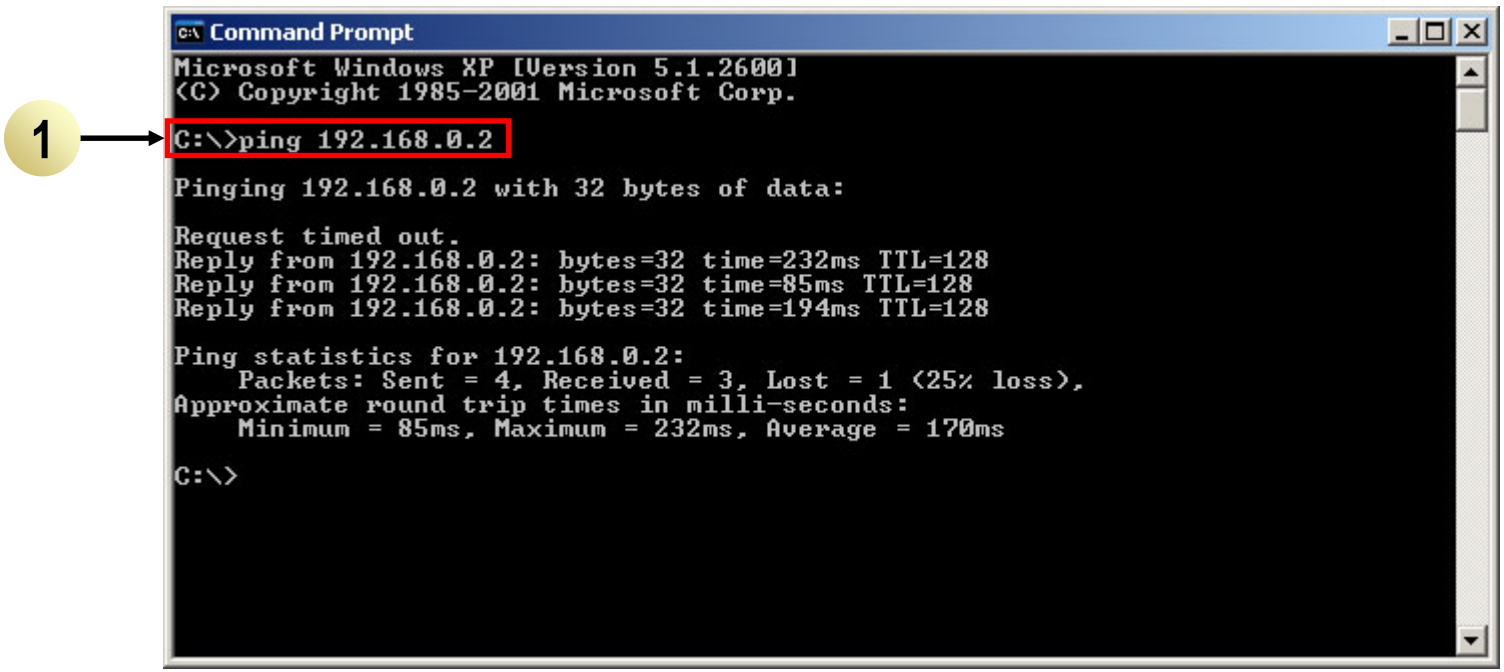
Run Ping Responder

- The program responds to incoming pings
  - It also notes and ignores any other packets on the network



Ping ML410 Target from Host

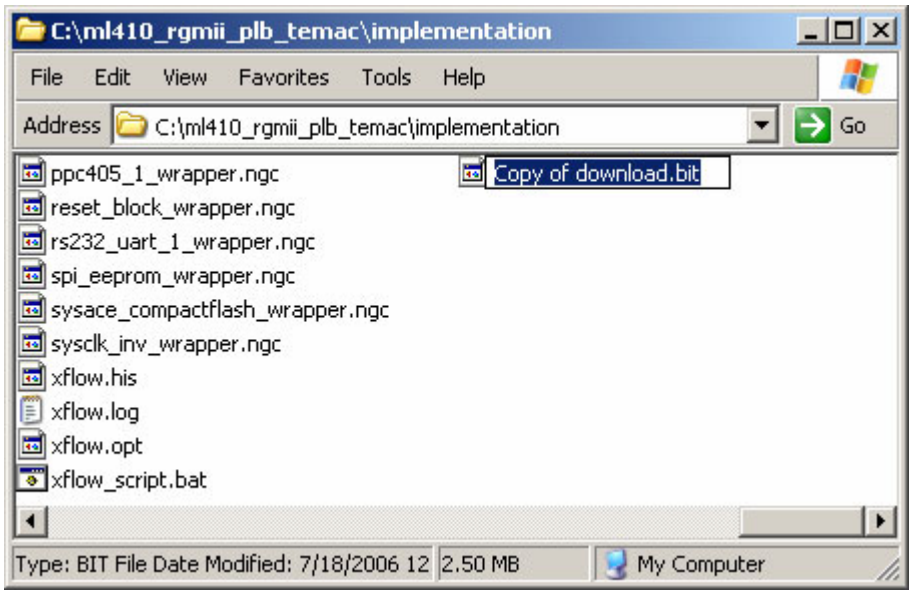
- Open a DOS window on the PC Host
  (**Start** → **Programs** → **Accessories** → **Command Prompt**)
- Type **ping 192.168.0.2** (1)
  - Ping from PC host 192.168.0.1 to ML410 target 192.168.0.2
  - You may see some timeouts on the PC

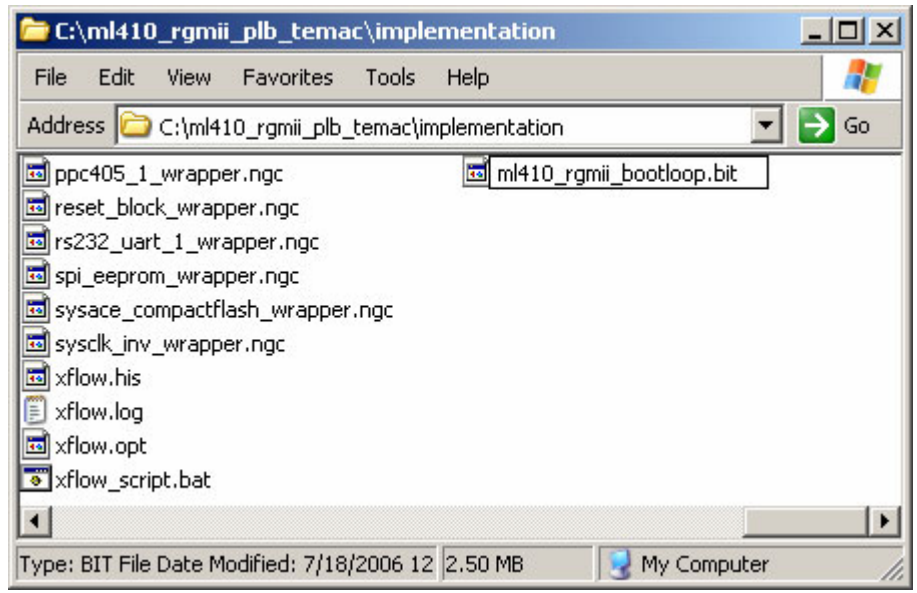


XILINX®

Loading Bootloop into BRAM

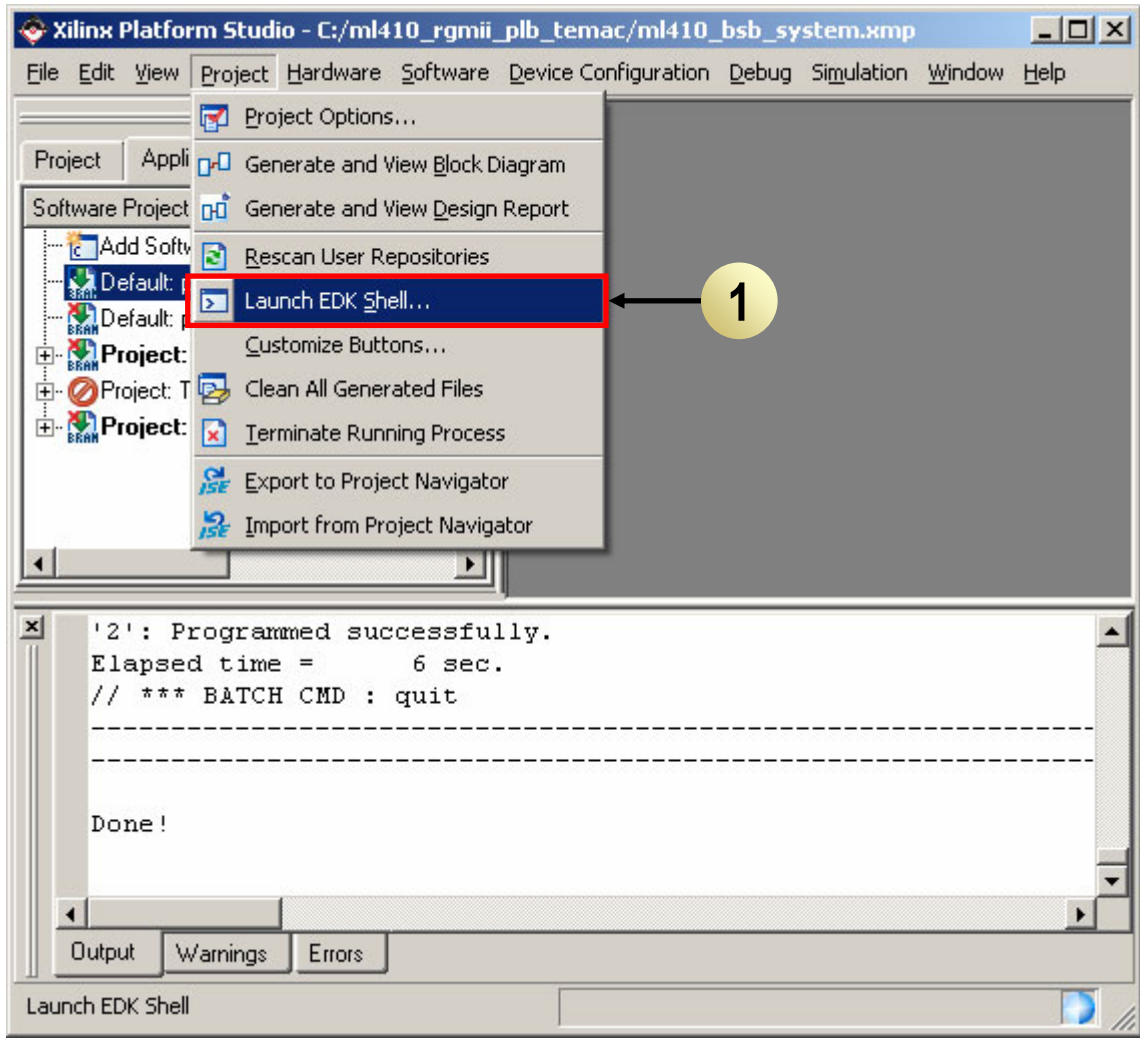
- Make a copy of the updated bitstream (download.bit) and rename it **ml410_rgmii_bootloop.bit**





Create an ACE File

- Open an EDK shell
  - Select **Project → Launch EDK Shell** (1)
  - This shell is used for entering and executing the commands to create a concatenated (HW+SW) ACE file



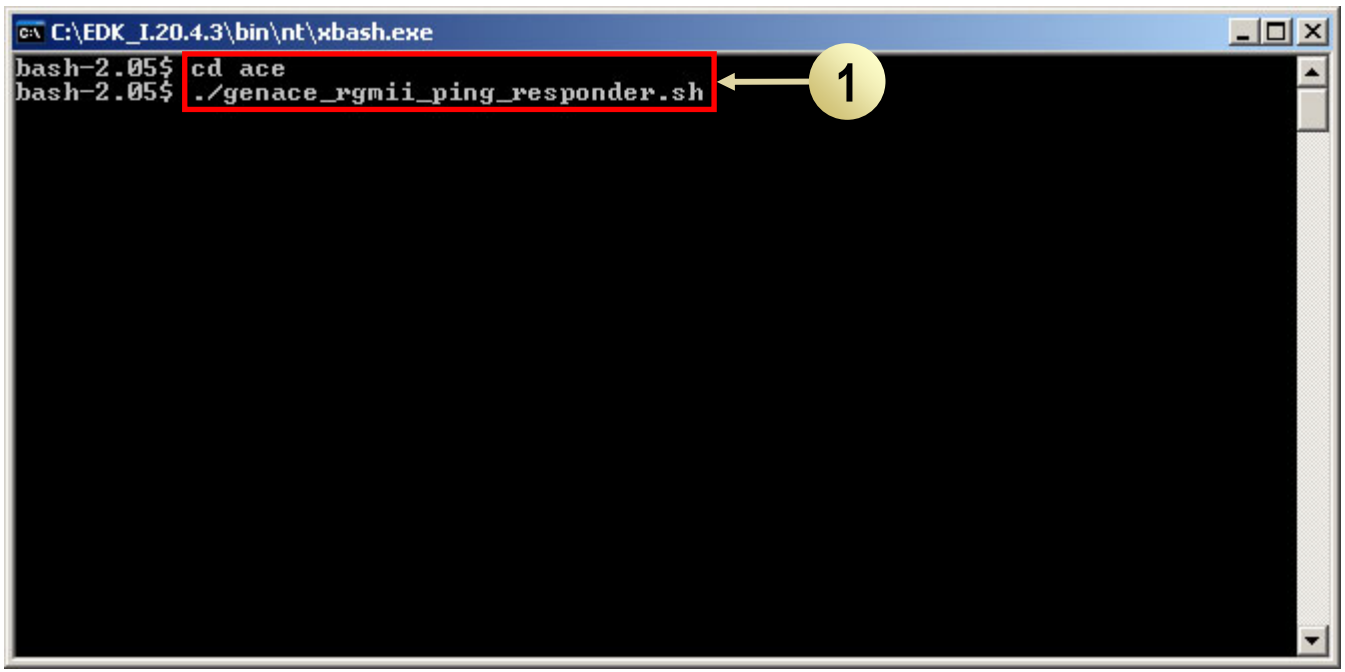
XILINX®

Create an ACE File

- At the bash prompt, type (1):

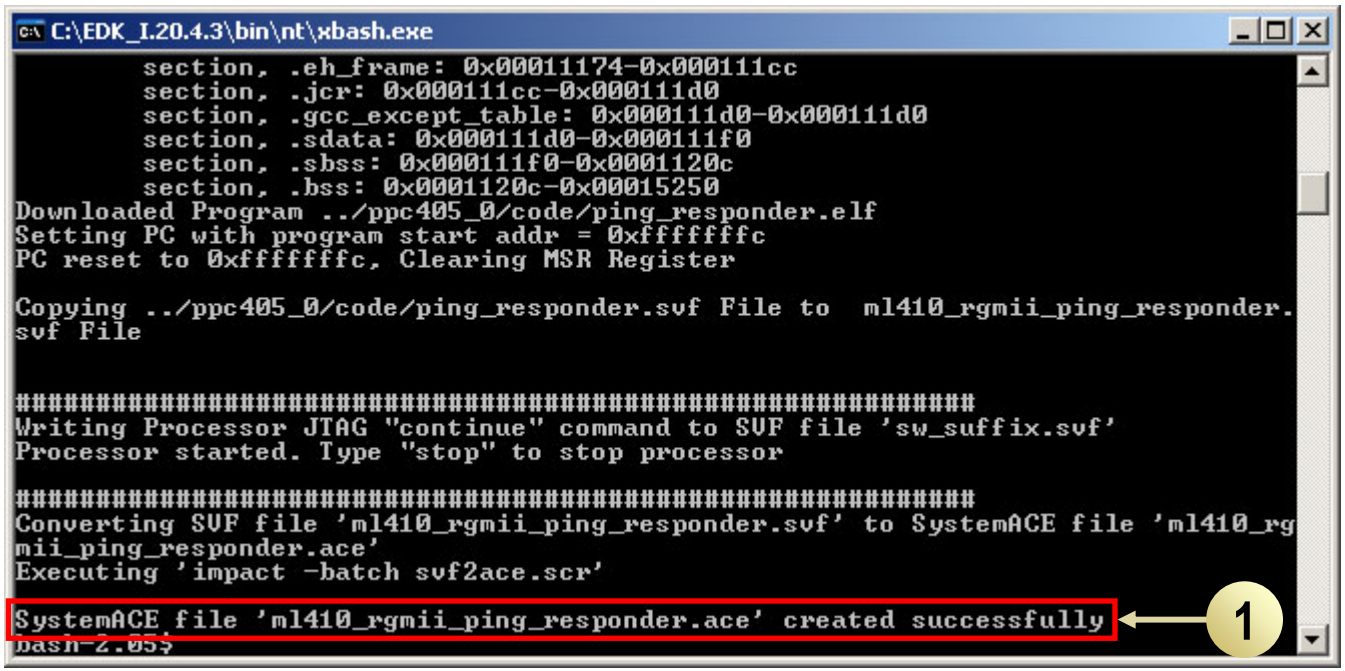
  cd ace

  ./genace_rgmii_ping_responder.sh



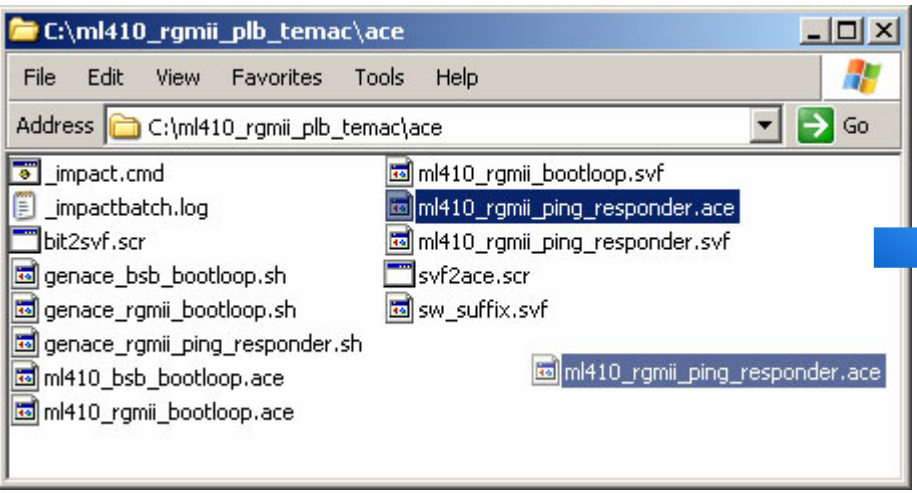
Create an ACE File

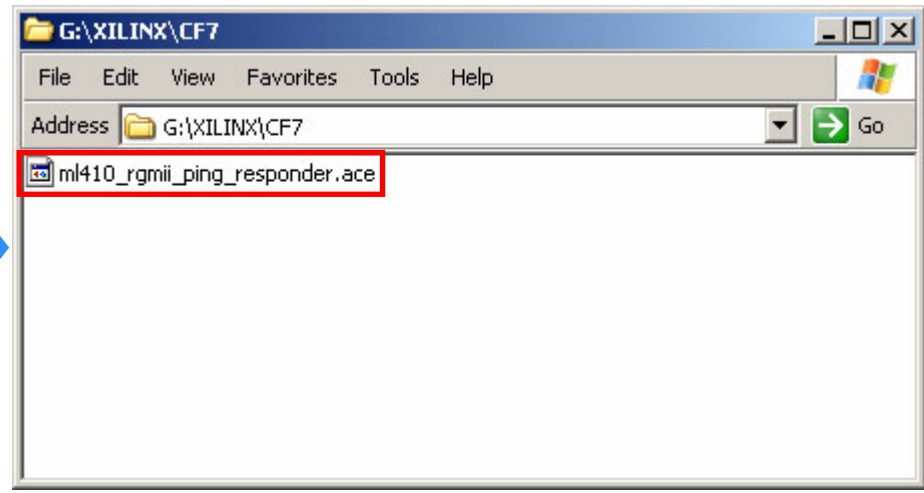
- This creates a concatenated (HW+SW) ACE file
  - Input: ping responder ELF file, ml410_rgmii_bootloop.bit
- Genace_rgmii_ping_responder.sh uses XMD and a genace.tcl script with ML410 appropriate options to generate an ACE file (1)



Run ACE File

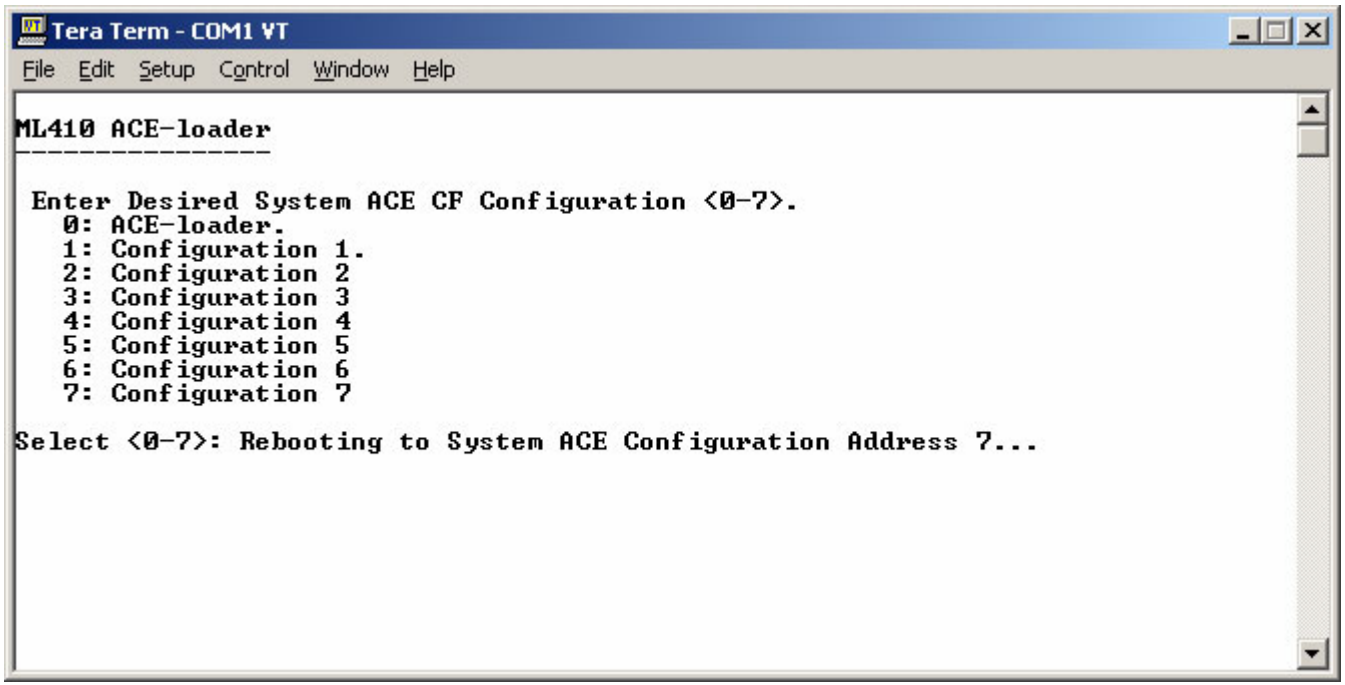
- Copy ml410_rgmii_ping_responder.ace to the xilinx\cf7 directory on your CompactFlash card
  - **Important**: Delete any existing ace files in this CF7 directory
  - **Note**: Use a CompactFlash reader to mount the CompactFlash as a disk drive





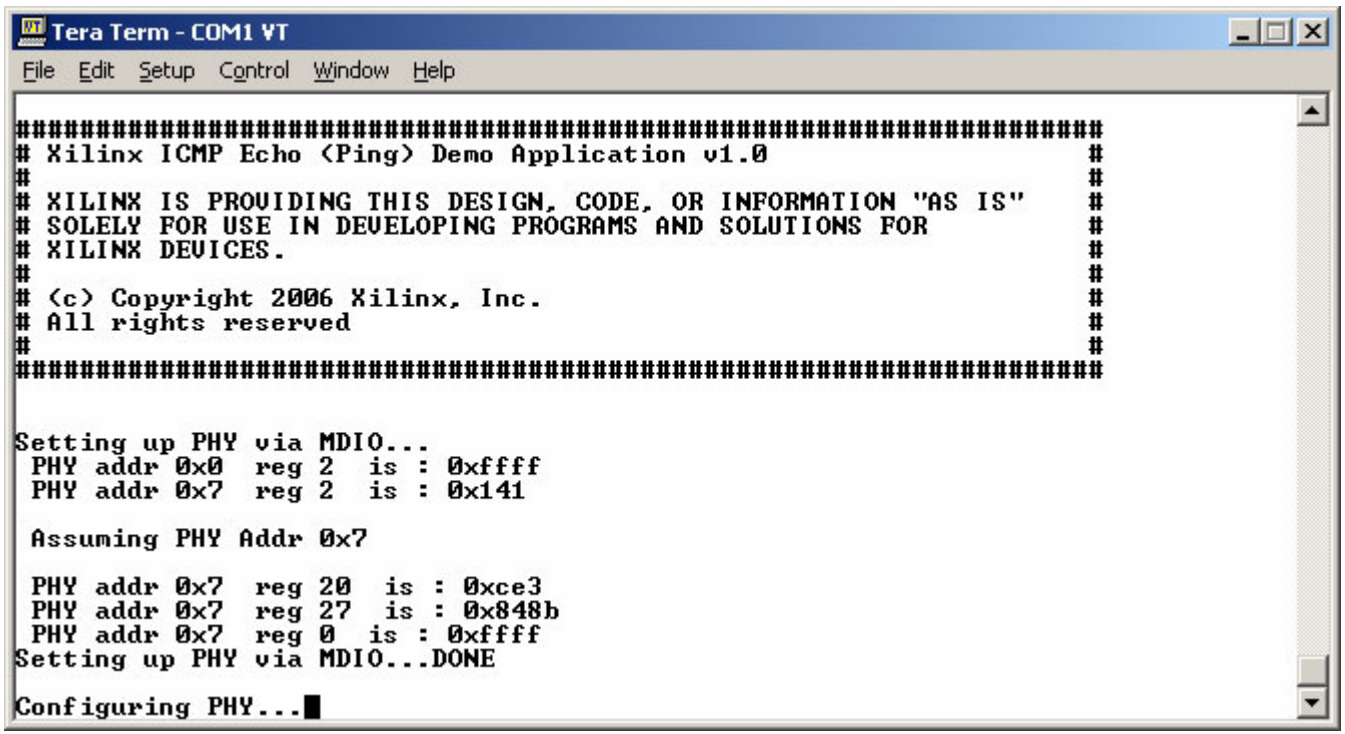
Run ACE File

- Eject the CompactFlash from your PC and insert it back into the ML410
- Type **7** to run the newly created ACE file



Using the ACE File

- Ping Responder output after booting ACE file



Available Documentation

- Platform Studio Documentation
  - Embedded Development Kit (EDK) Resources
    http://www.xilinx.com/ise/embedded_design_prod/platform_studio.htm
  - OS and Libraries Document Collection
    http://www.xilinx.com/ise/embedded/oslib_rm.pdf
- ML410
  - ML410 User's Guide
    http://www.xilinx.com/bvdocs/userguides/ug085.pdf
  - ML410 Overview
    http://www.xilinx.com/ml410
  - ML410 Schematics
    http://www.xilinx.com/products/boards/ml410/docs/ml410_revE.pdf