



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

April 2007



www.BDTIC.com/XILINX

XILINX

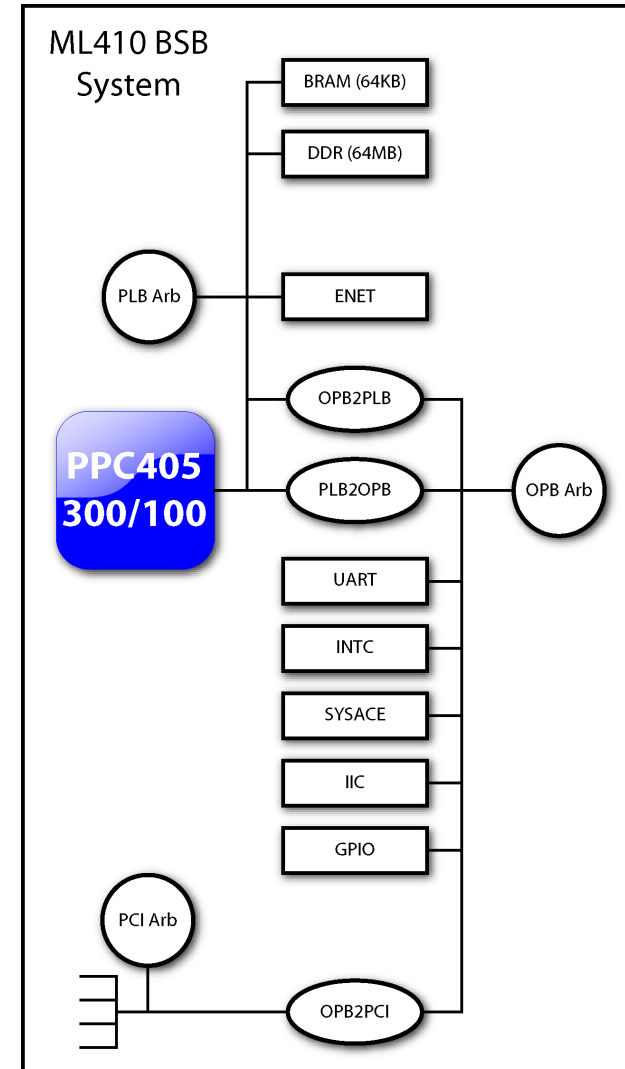
Overview

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



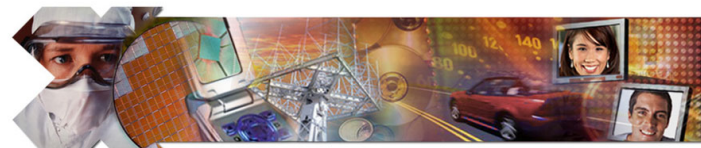
ML410 BSB Hardware

- 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**



ML410 Overview and Setup

Overview of the Hardware Designs and Software Applications
How to set up the equipment, software, CompactFlash,
network, and terminal programs



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



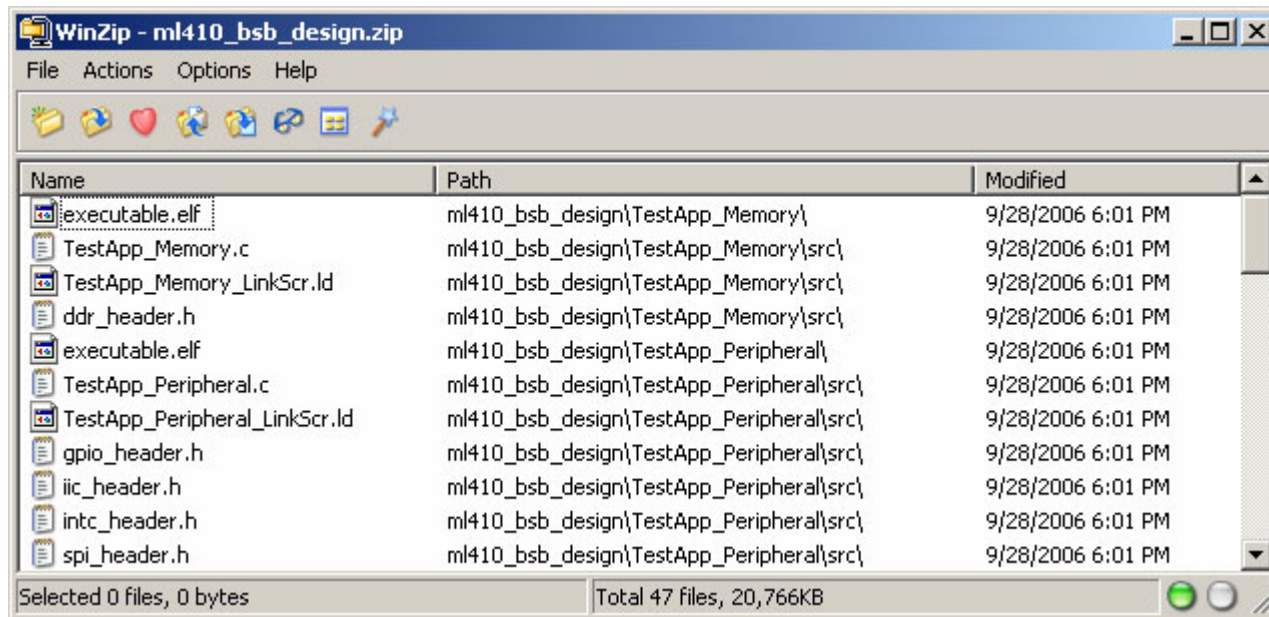
EDK Software Requirement

- Xilinx EDK 8.2i SP1 software



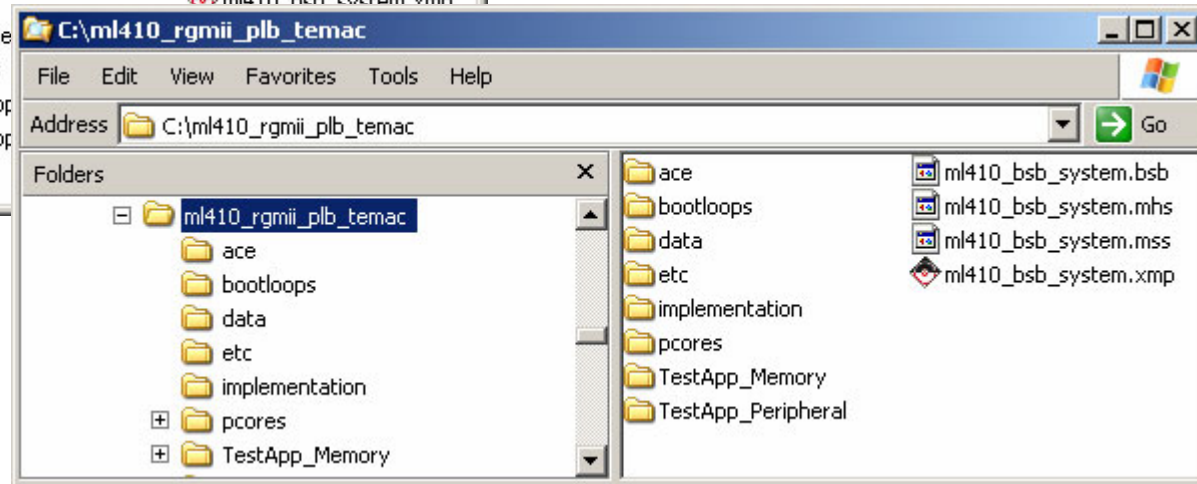
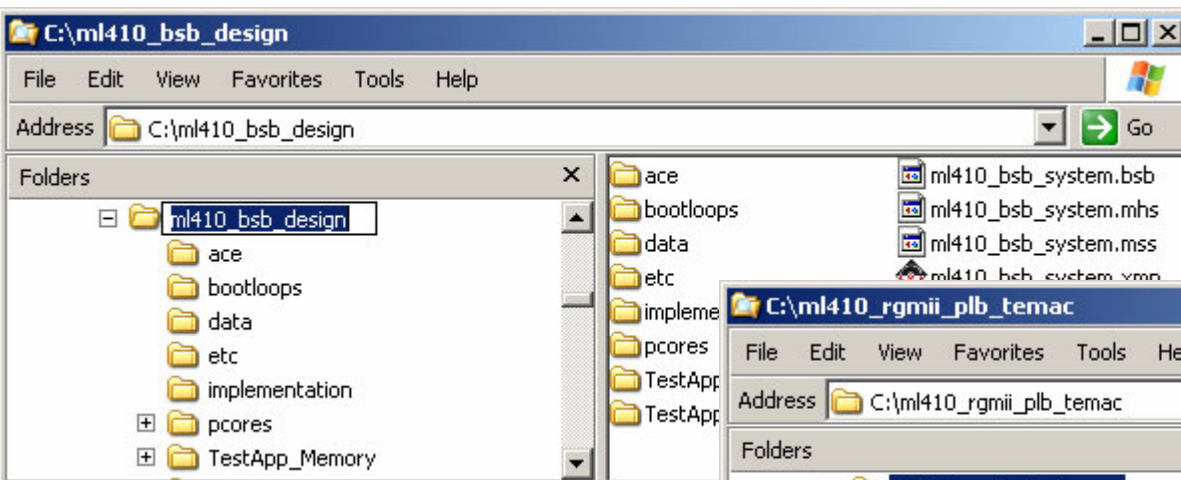
Extracting the Design

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



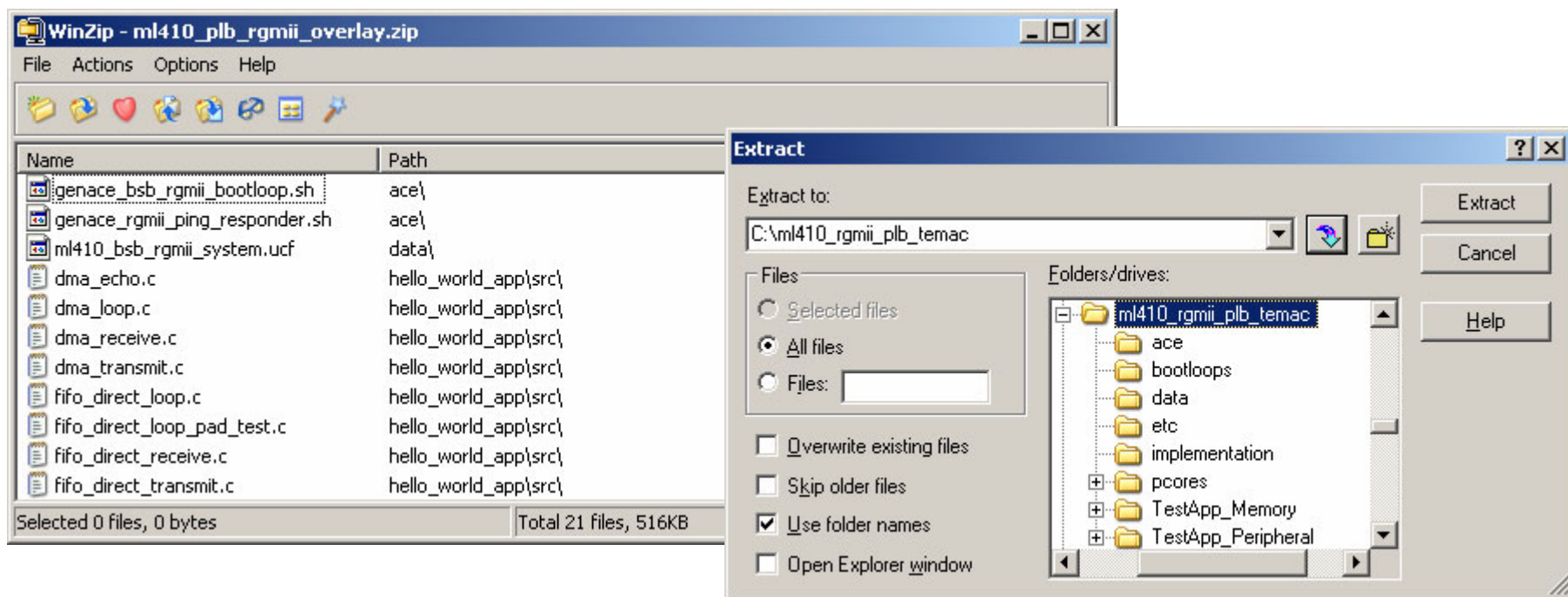
Extracting the Design

- Rename the project directory to ml410_rgmii_plb_temac



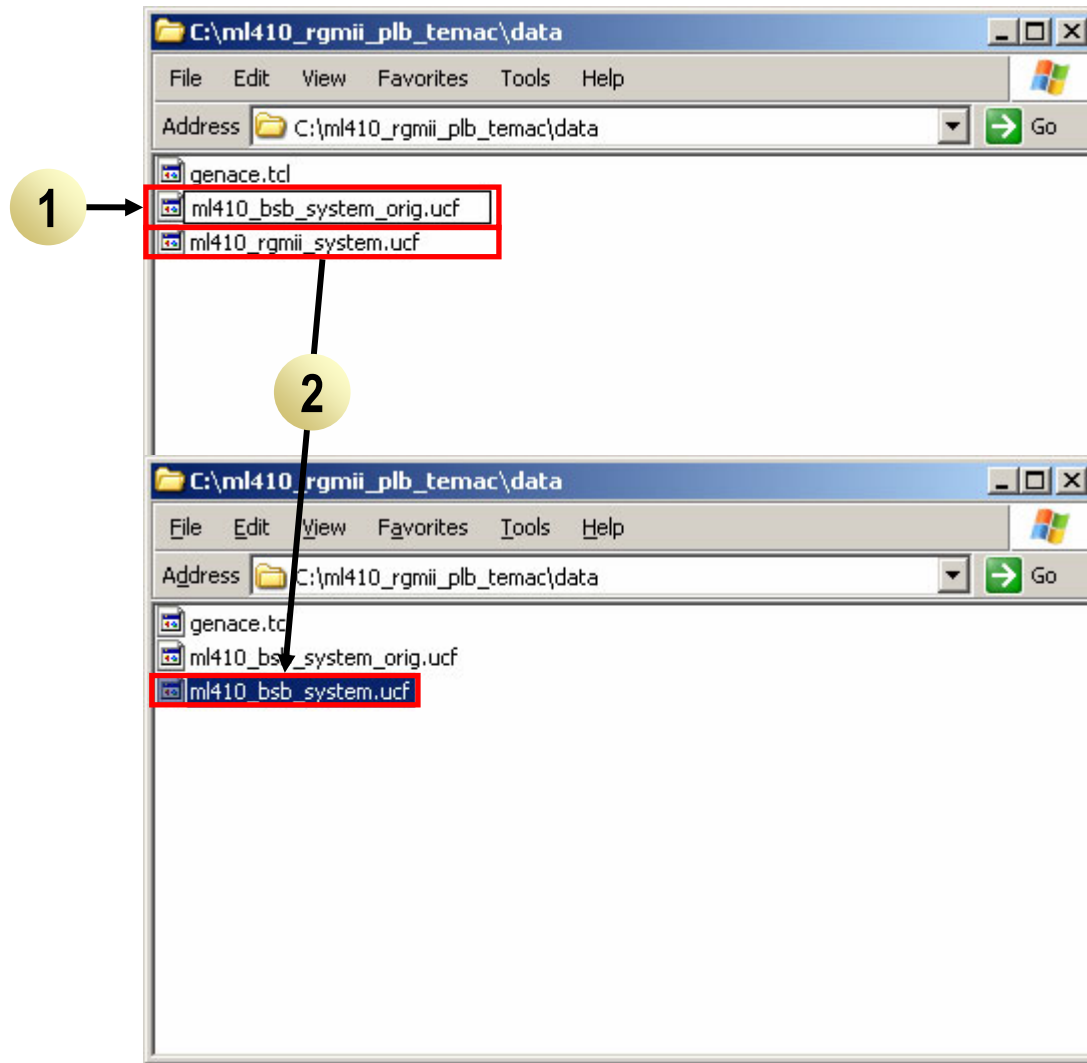
Extracting the Design

- 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

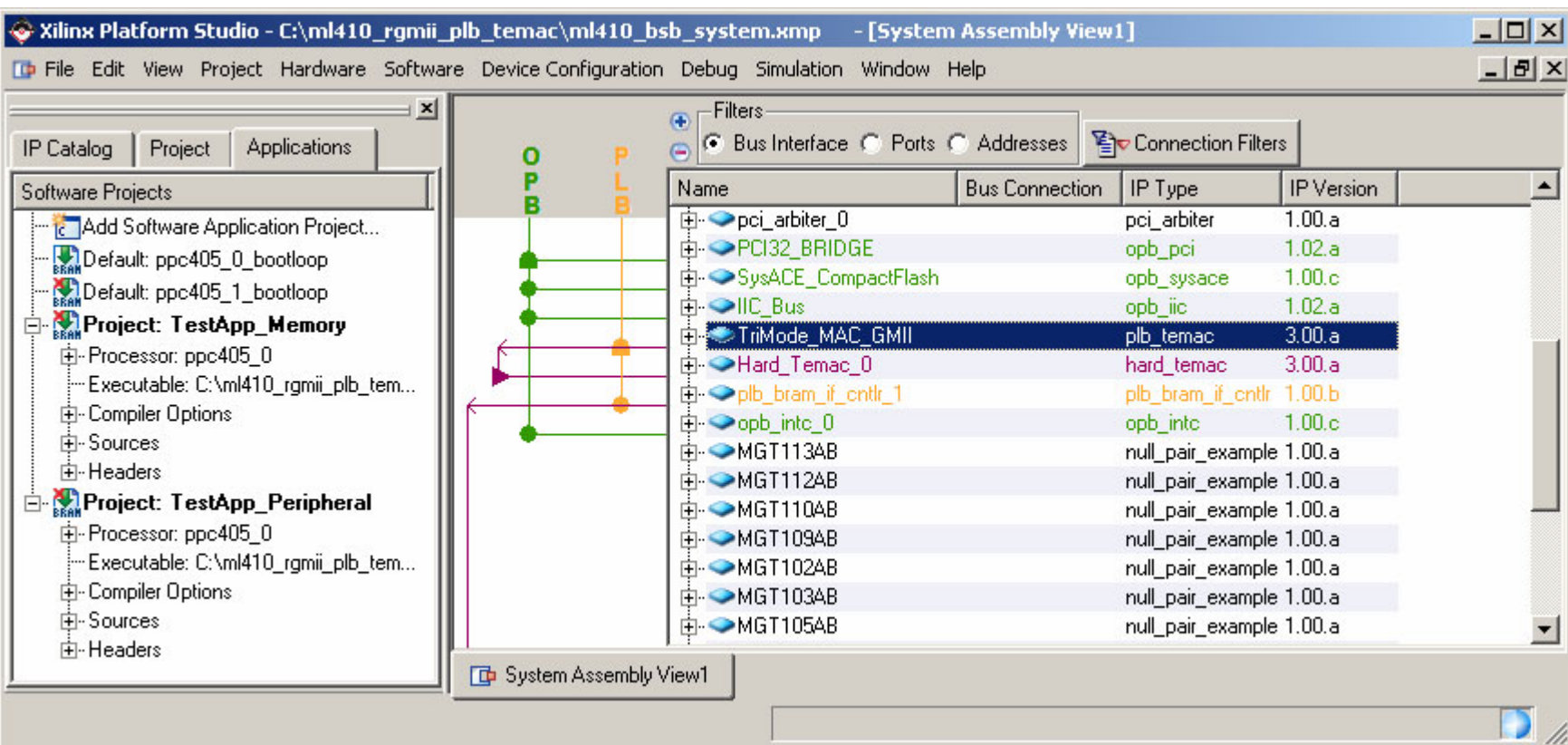
- Rename the ml410_bsb_system.ucf (1)
- Change the file named ml410_rgmii_system.ucf to ml410_bsb_system.ucf (2)



Add and Configure IP

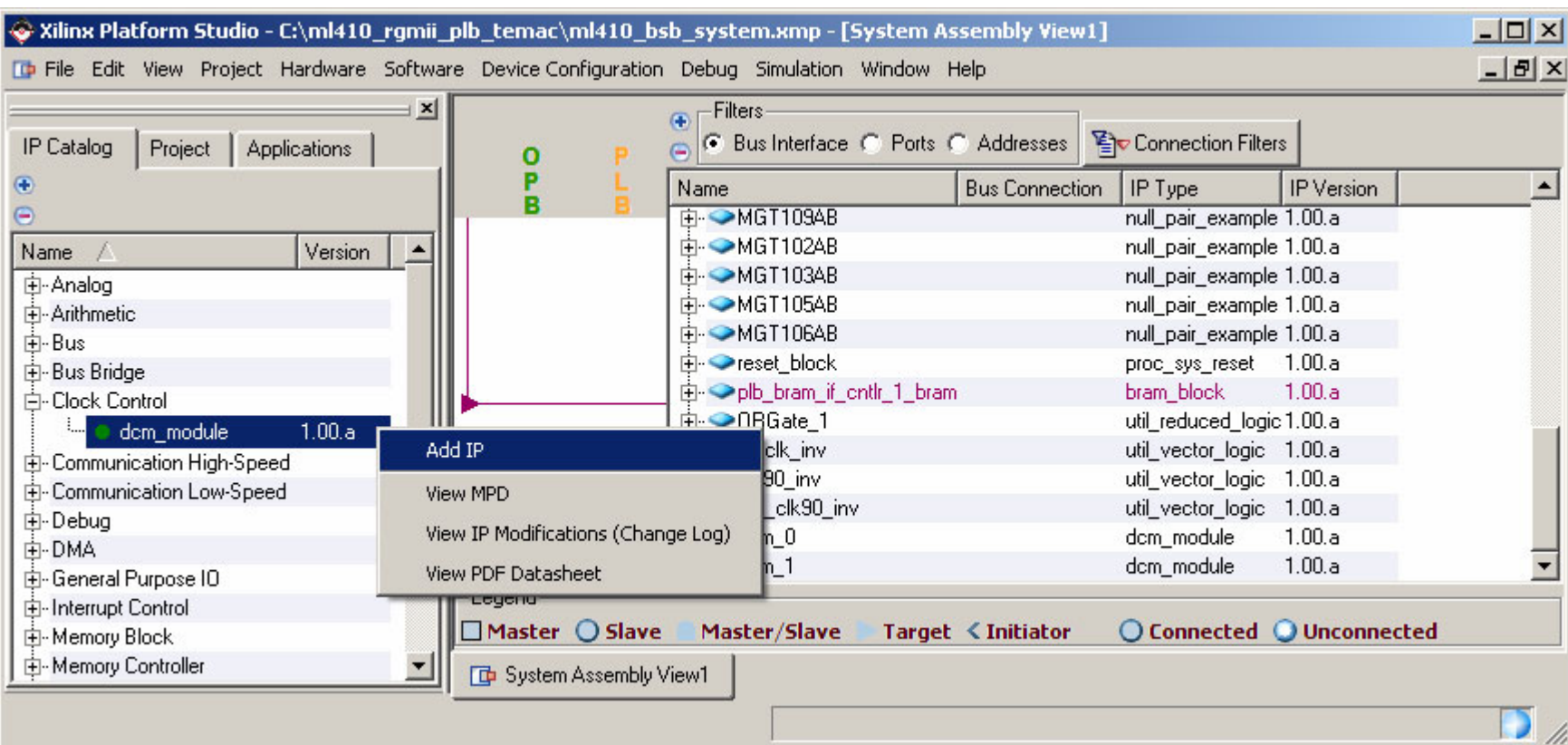
Add RGMII

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



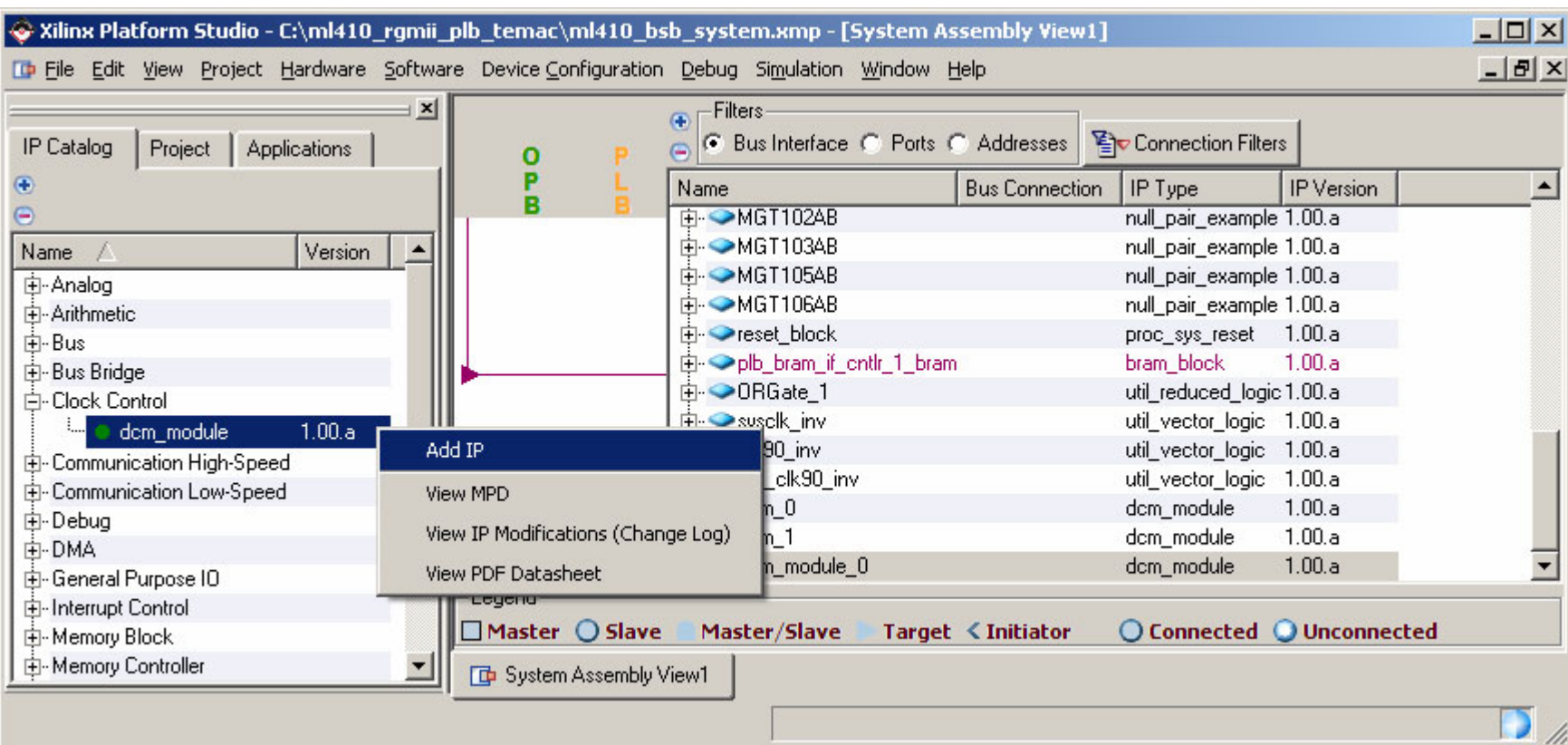
Add RGMII

- Add a dcm



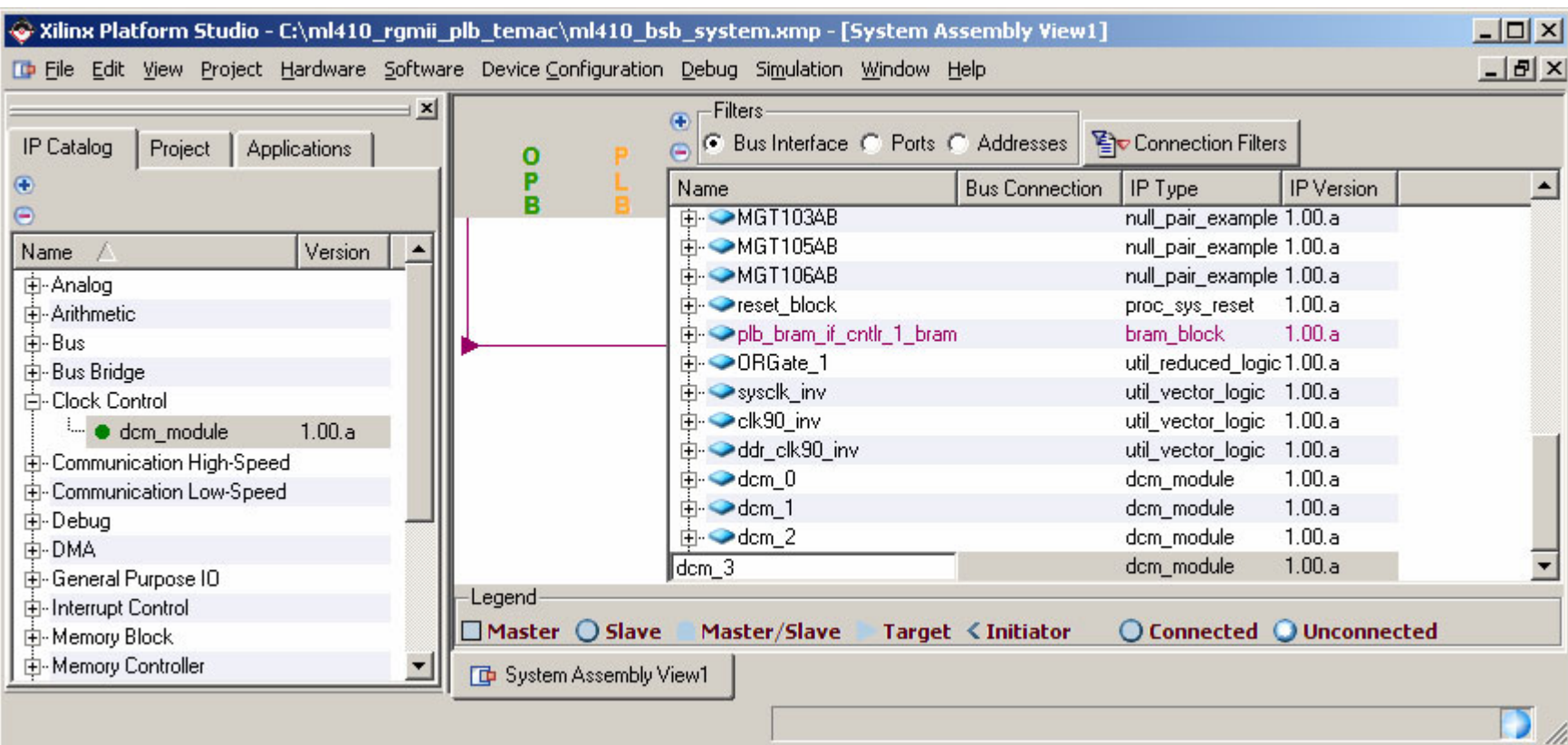
Add RGMII

- Add another dcm



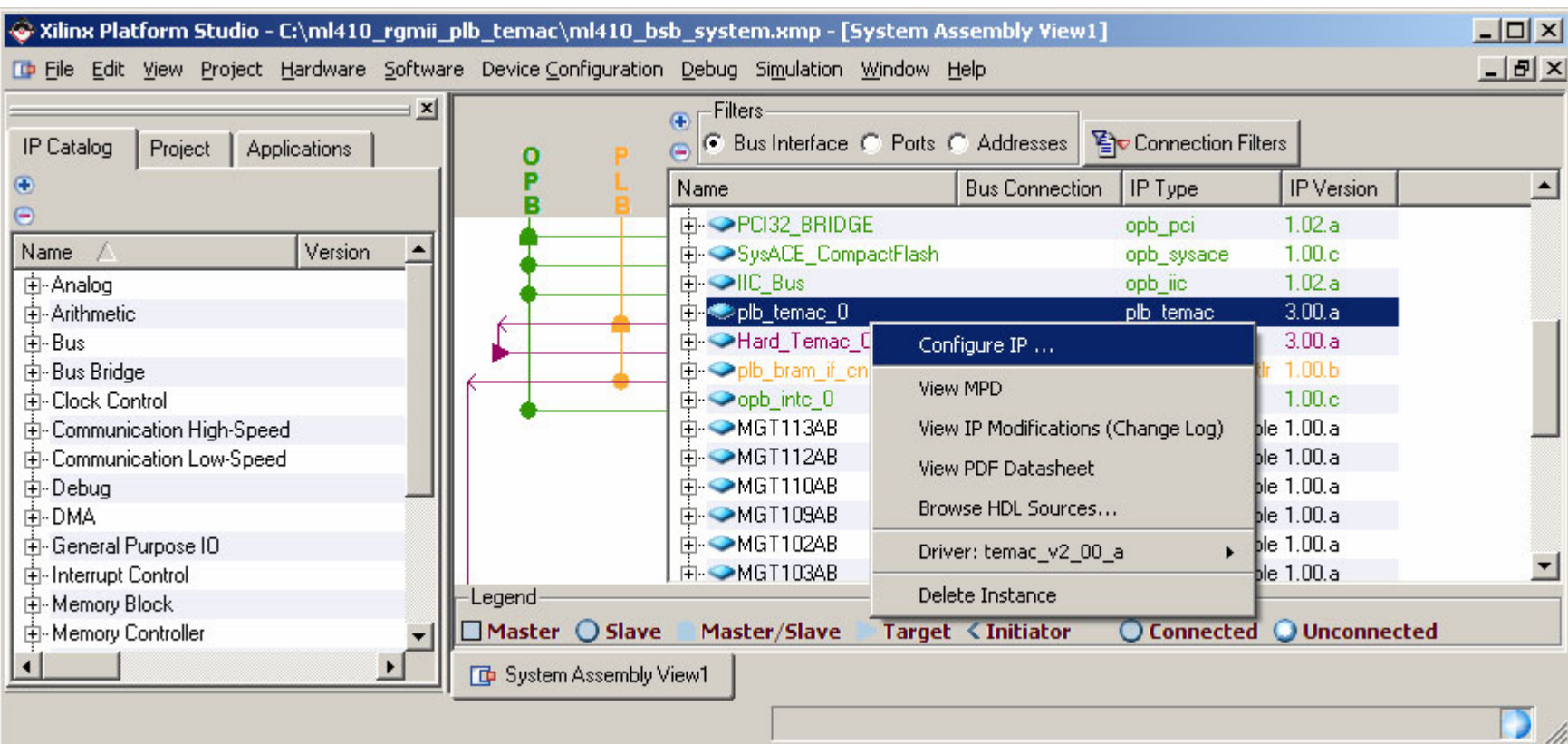
Add RGMII

- Rename the dcm_modules to dcm_2 and dcm_3



Add RGMII

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



Add RGMII

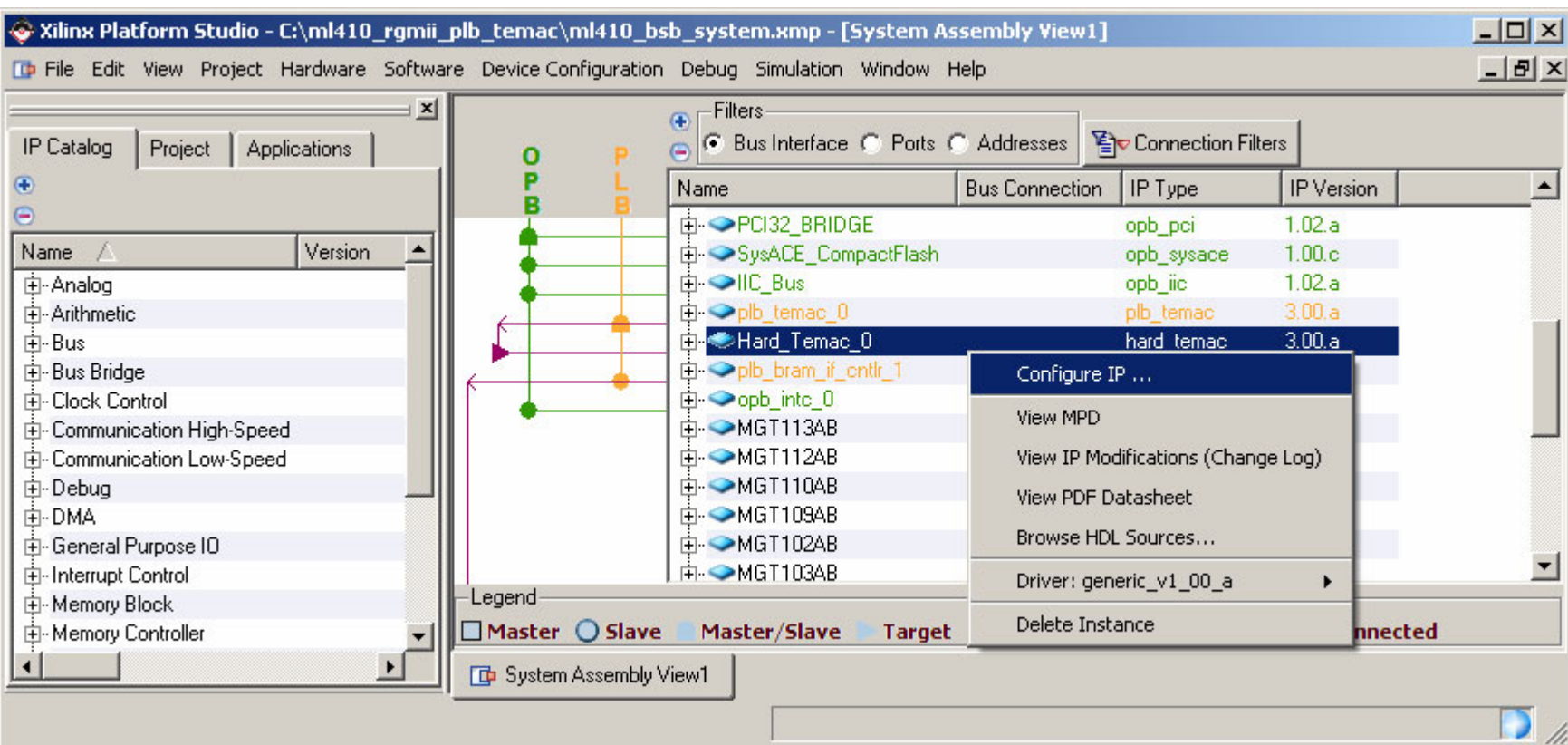
- Under the User tab, set the following parameters:
 - Receiving FIFO Depth: **131072**
 - Transmitting FIFO Depth: **131072**

The screenshot shows the configuration window for 'plb_temac_0 : plb_temac_v3_00_a'. The 'User' tab is active. The 'Receiving FIFO Depth in bits' and 'Transmitting FIFO Depth in bits' are highlighted with a red box and set to 131072. Other parameters are as follows:

Parameter	Value
Device Block ID	1
Receiving FIFO Depth in bits	131072
Transmitting FIFO Depth in bits	131072
MAC Length and Status FIFO Depth	64
Include Receiving Data Realignment Engine	0
Include Transmitting Data Realignment Engine	0
Include Device Module Identification Read	☒
DMA Type	1

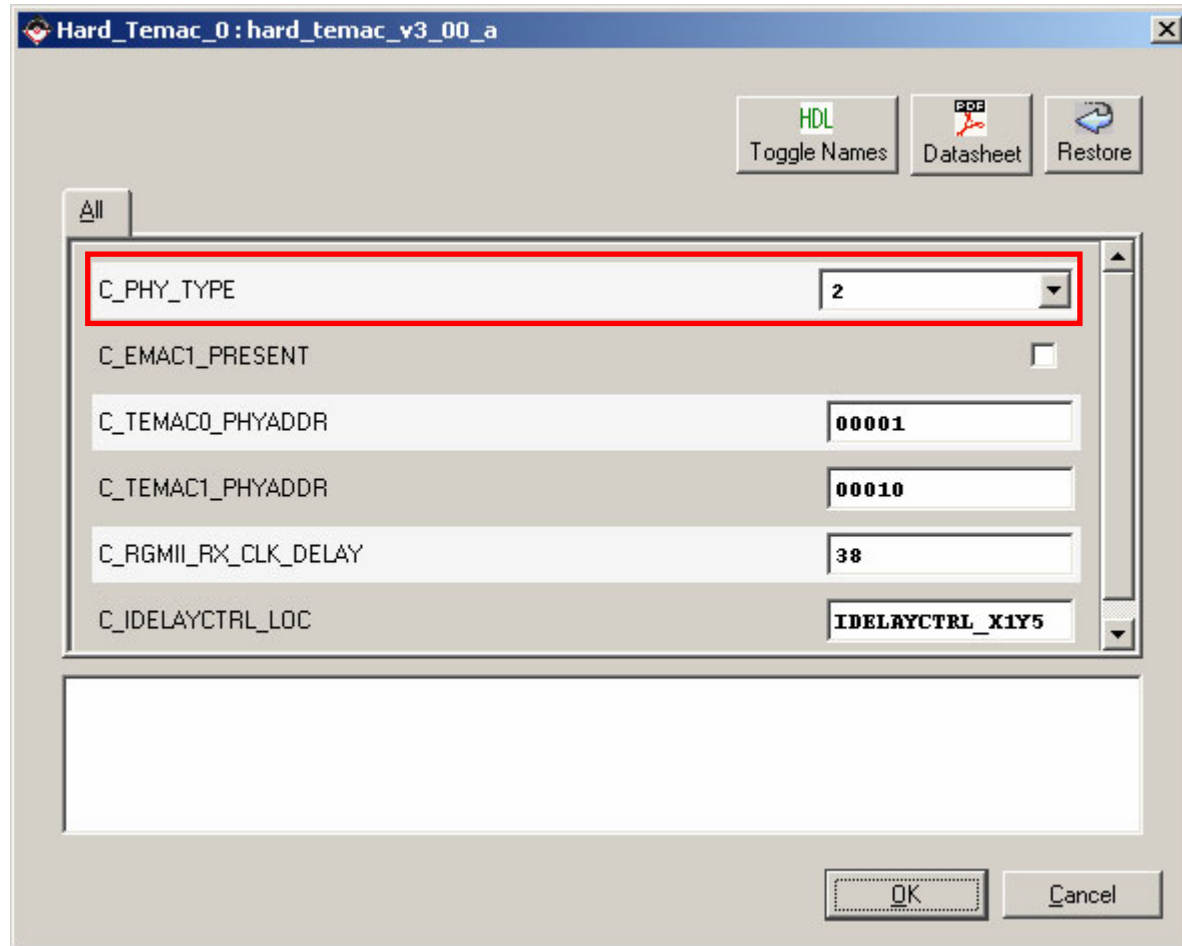
Add RGMII

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



Add RGMII

- Set the C_PHY_TYPE to 2 for RGMII



Hard_Temac_0:hard_temac_v3_00_a

Toggle Names Datasheet Restore

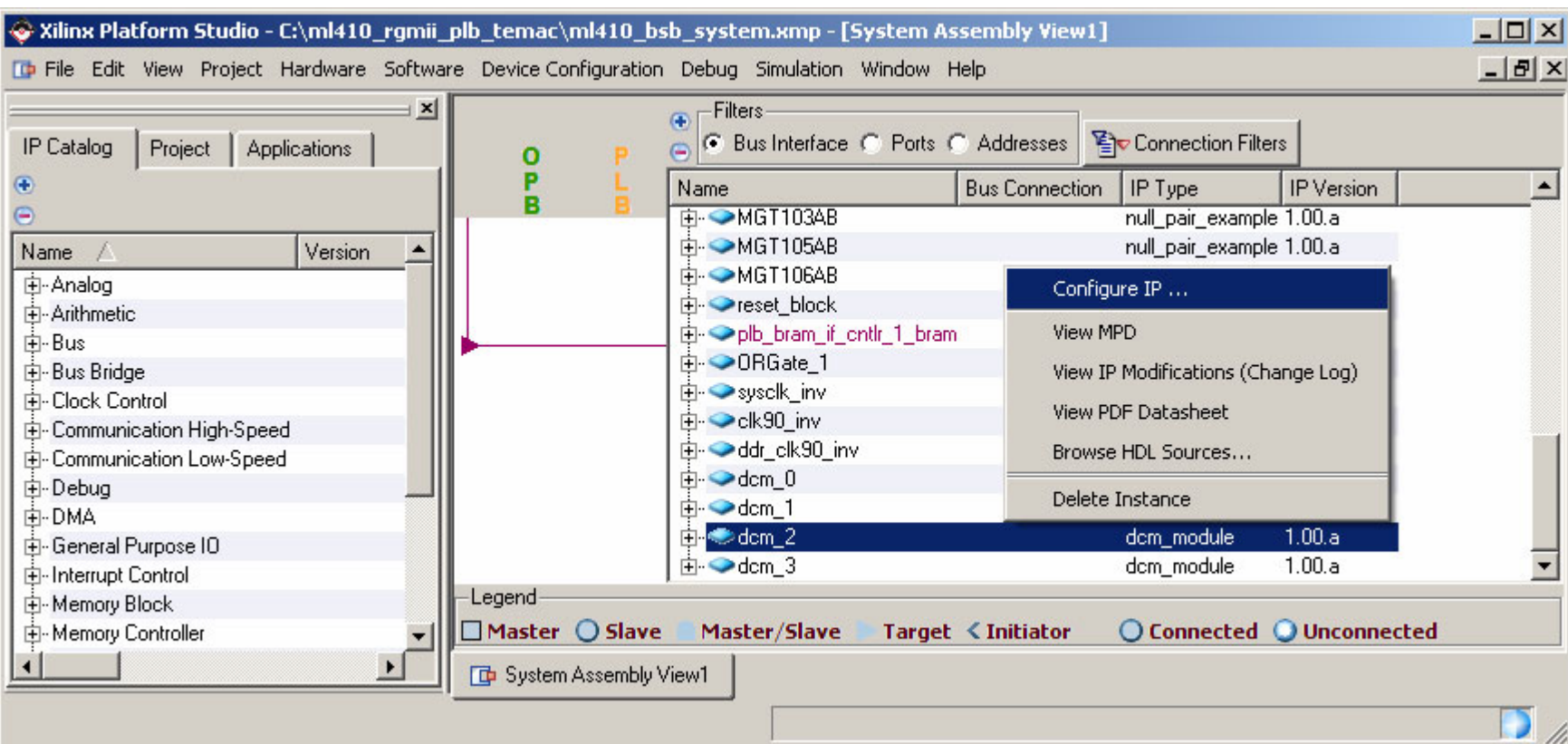
All

C_PHY_TYPE	2
C_EMAC1_PRESENT	☐
C_TEMAC0_PHYADDR	00001
C_TEMAC1_PHYADDR	00010
C_RGMII_RX_CLK_DELAY	38
C_IDELAYCTRL_LOC	IDELAYCTRL_X1Y5

OK Cancel

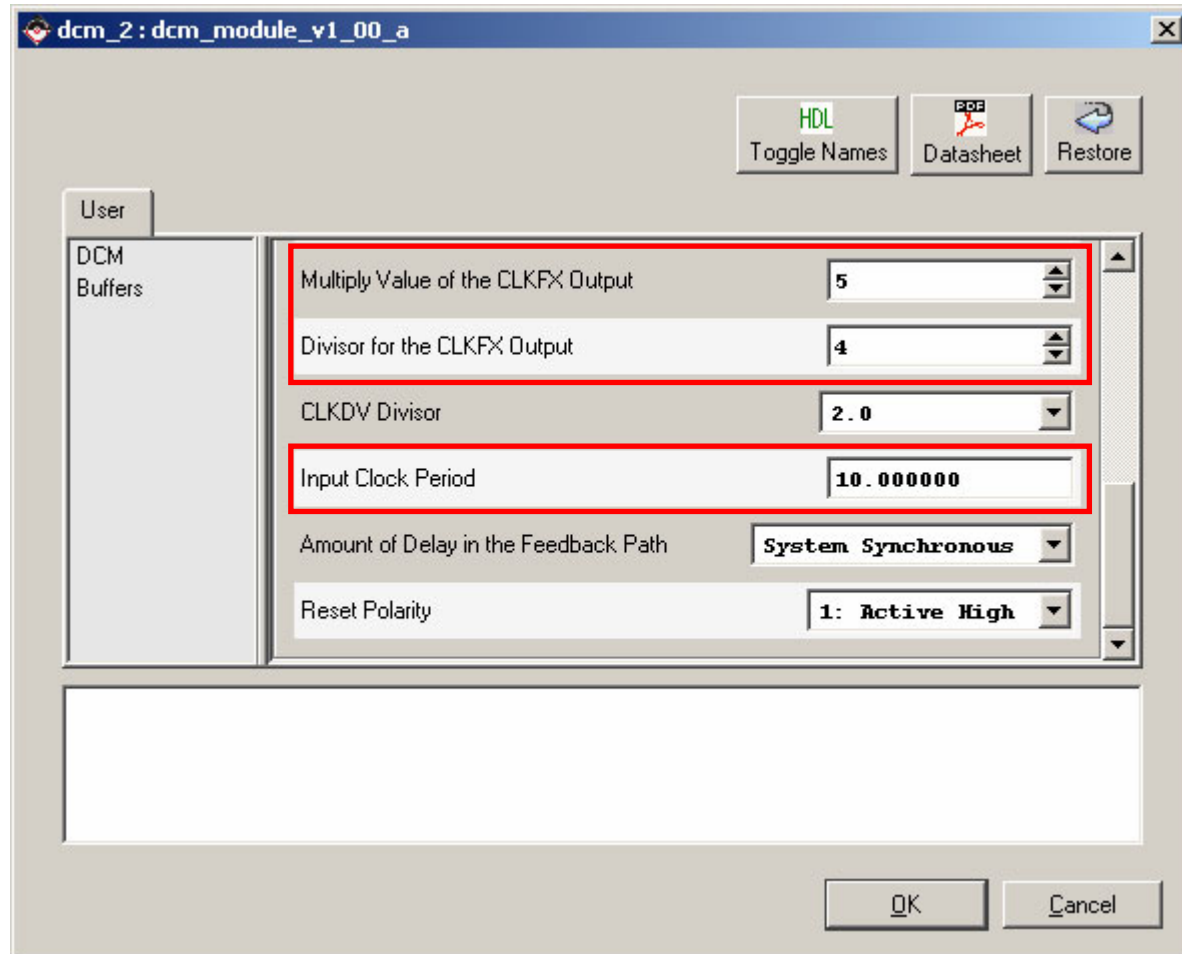
Add RGMII

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



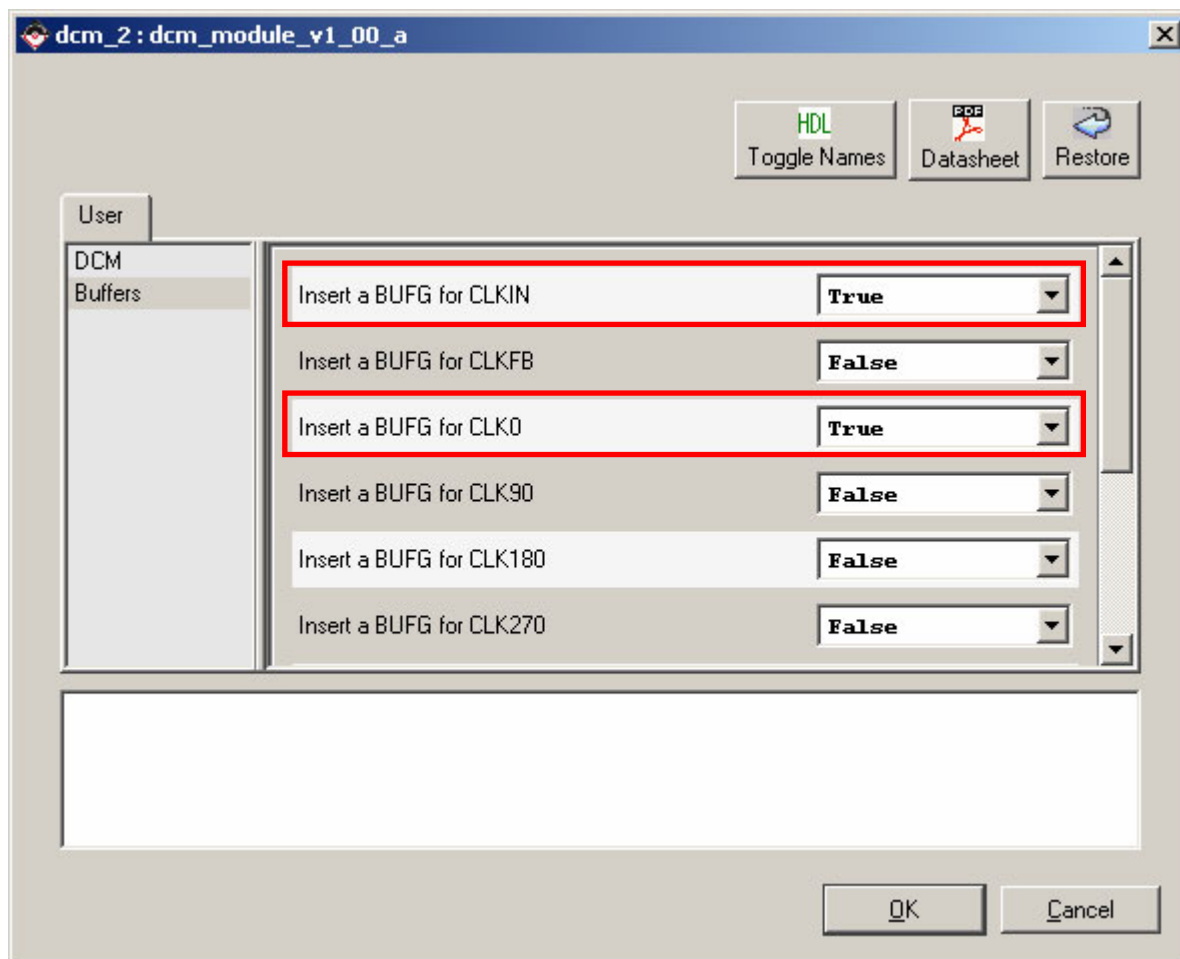
Add PLB TFT Controller

- 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



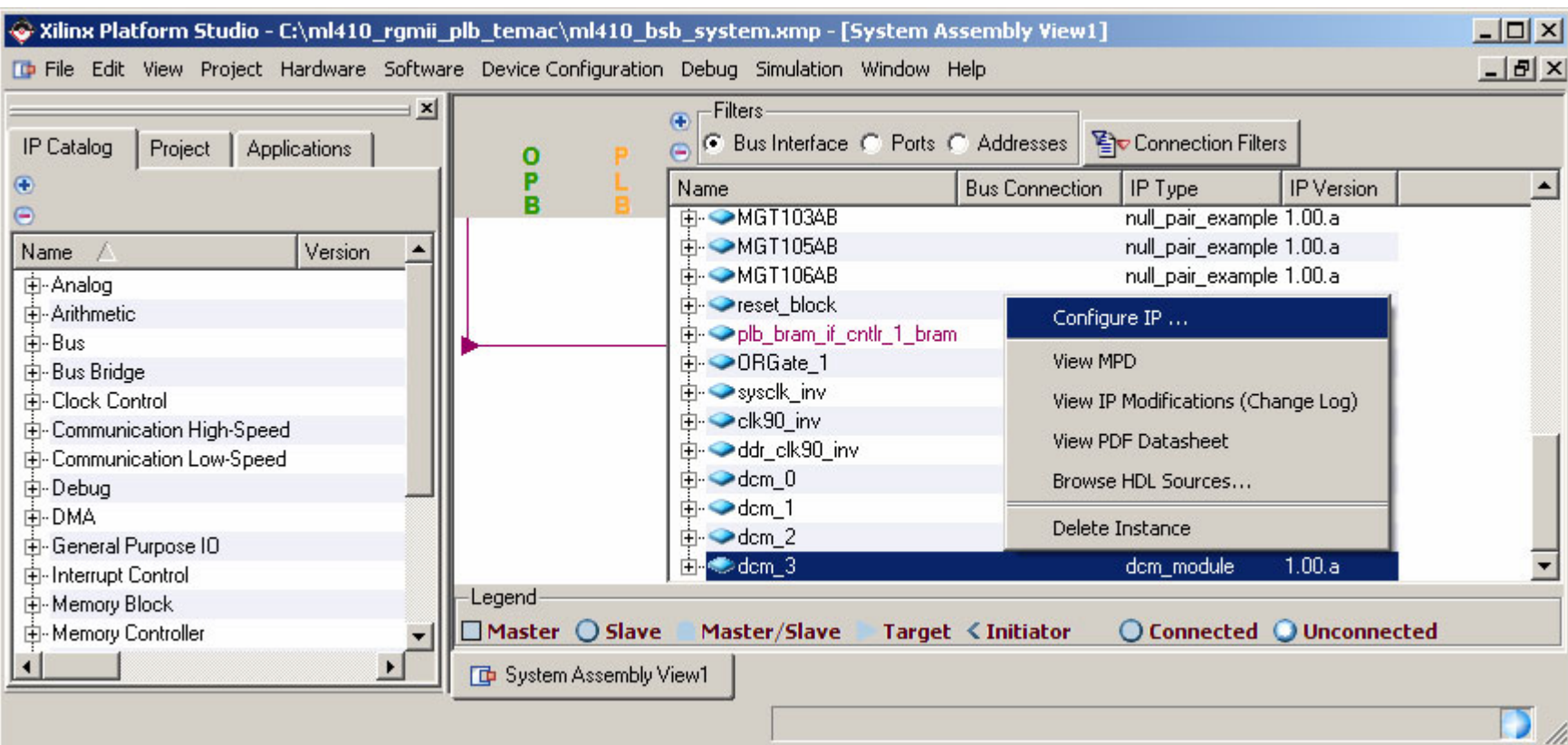
Add PLB TFT Controller

- 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

dc3 : dcm_module_v1_00_a

Toggle Names Datasheet Restore

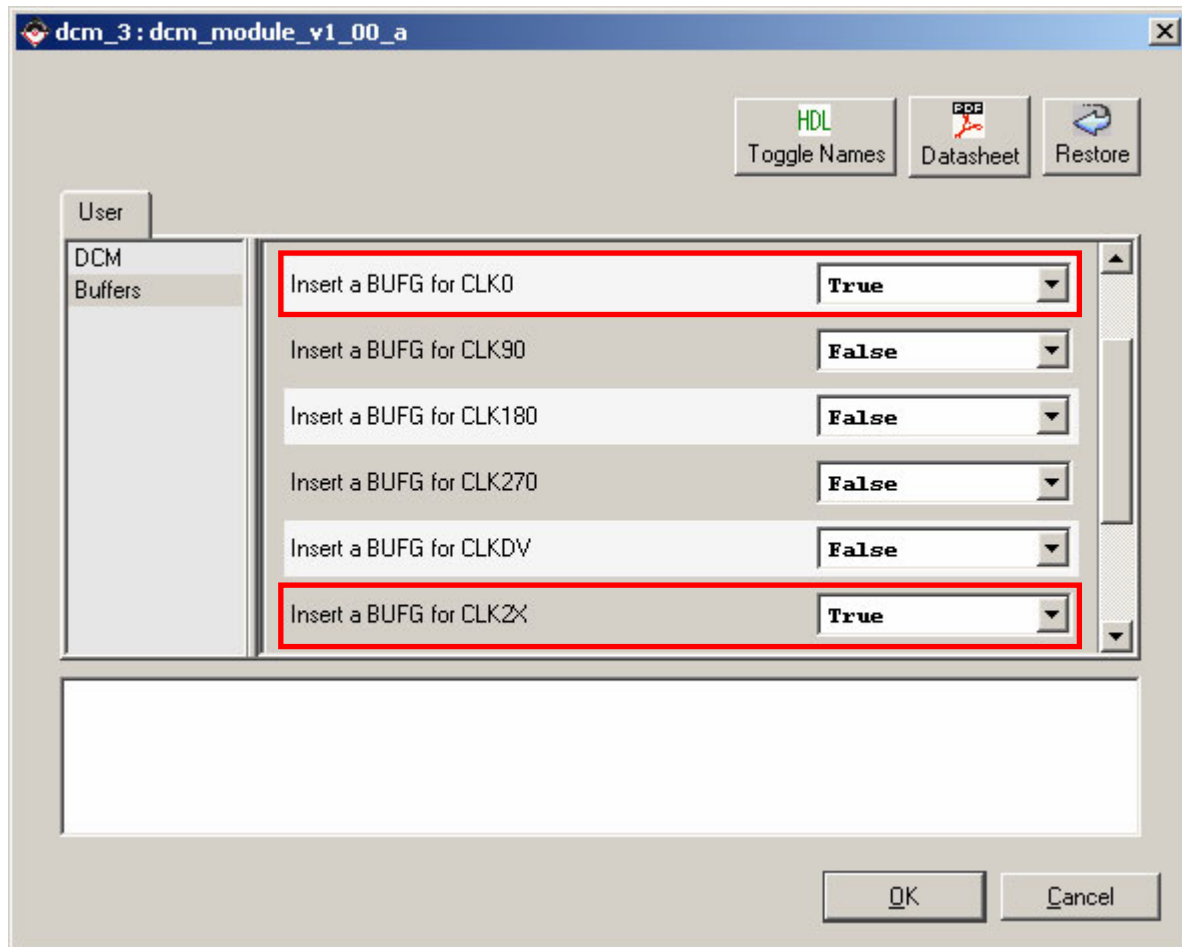
User

DCM Buffers

Digital Frequency Synthesizer Clock Frequency Mode	High
Delay Locked Loop Frequency Mode	Low
Duty Cycle Correction	True
CLKIN Divide By 2	False
Clock Feedback Input	1X
Controls Use of Phase Shift	None
DSS Mode	None
Configuration Startup Wait	False
Phase Shift	0
Multiply Value of the CLKFX Output	4
Divisor for the CLKFX Output	1
CLKDV Divisor	2.0
Input Clock Period	10.000000

Add PLB TFT Controller

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

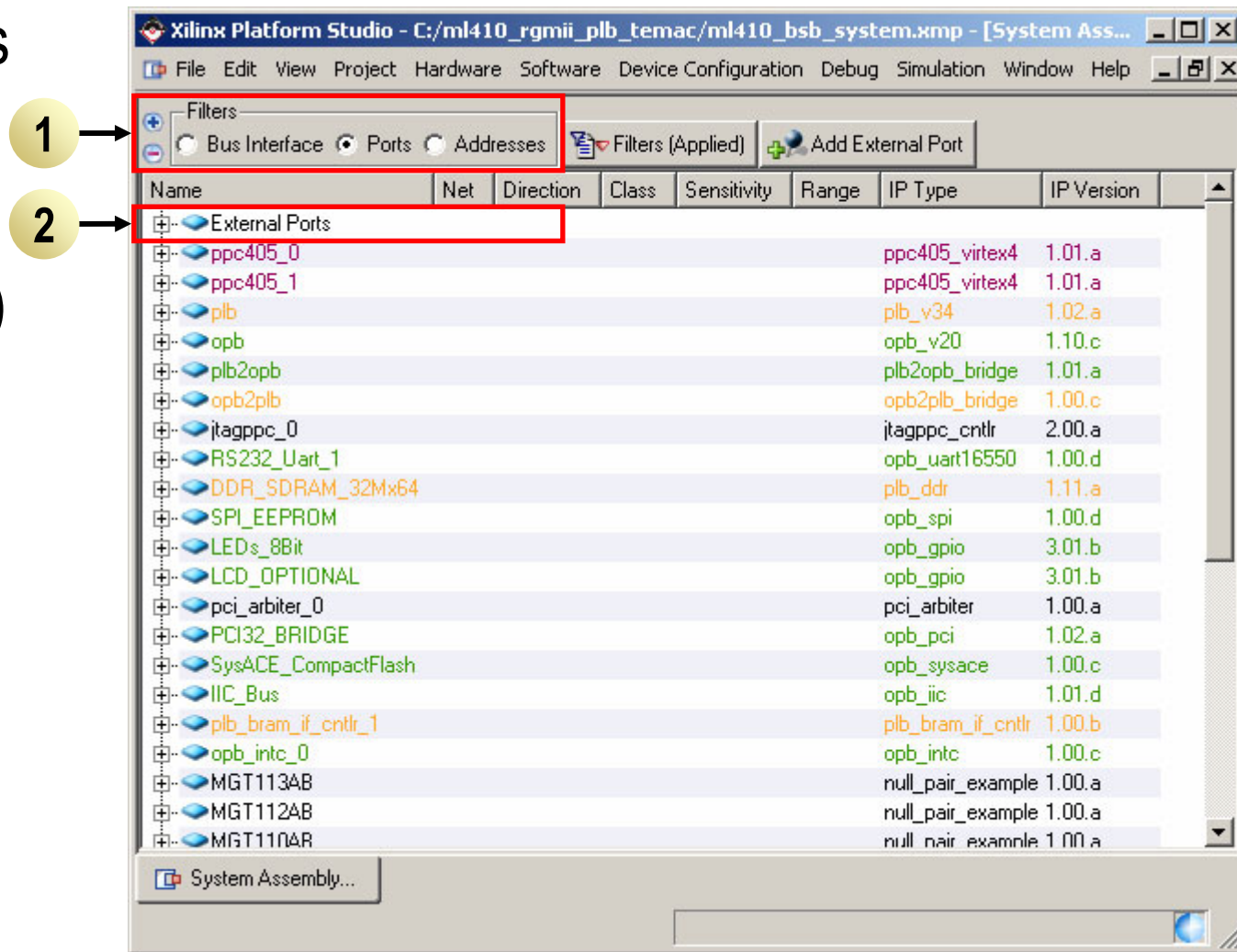


Add External Ports Connect the Nets



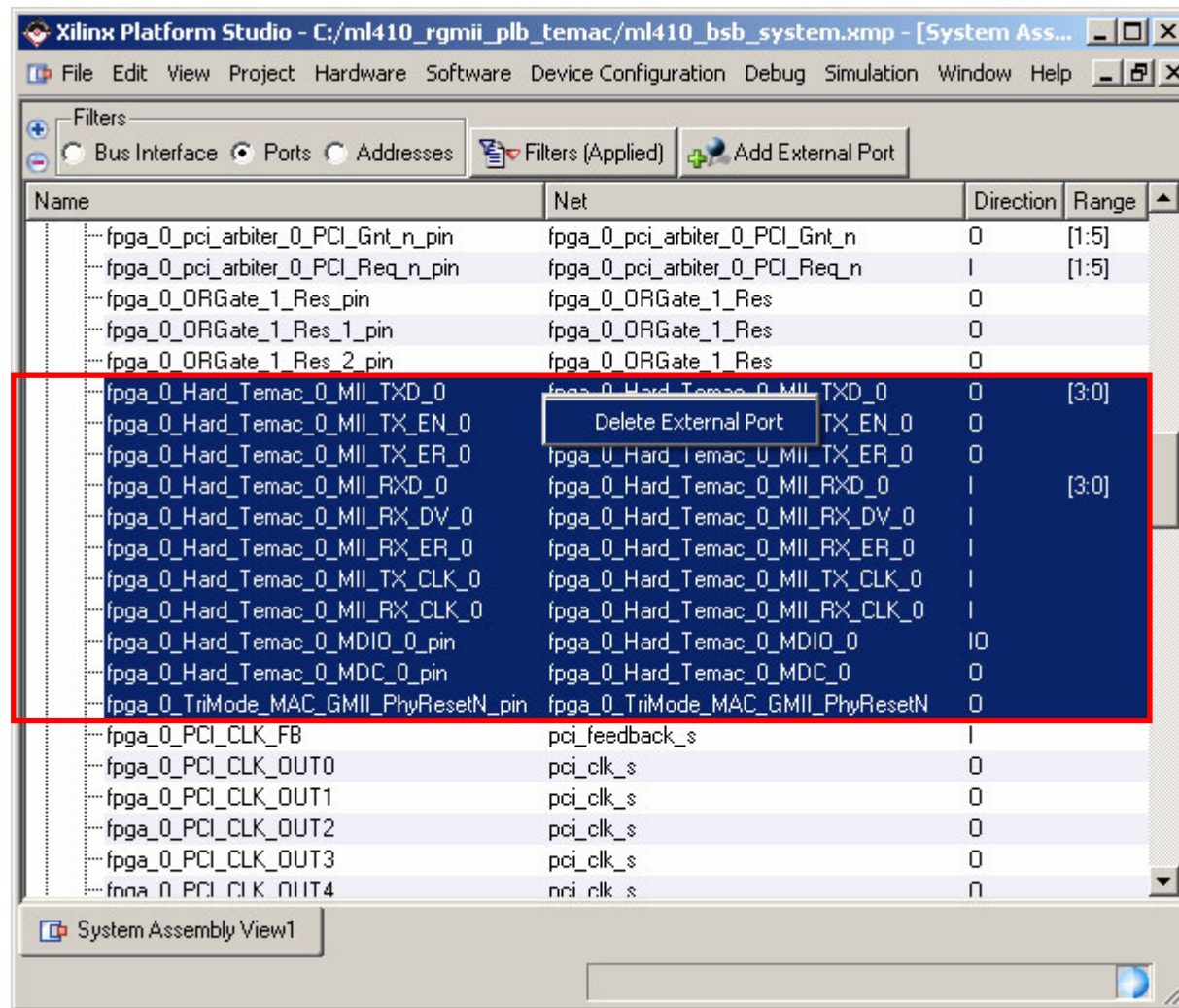
Add RGMII

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



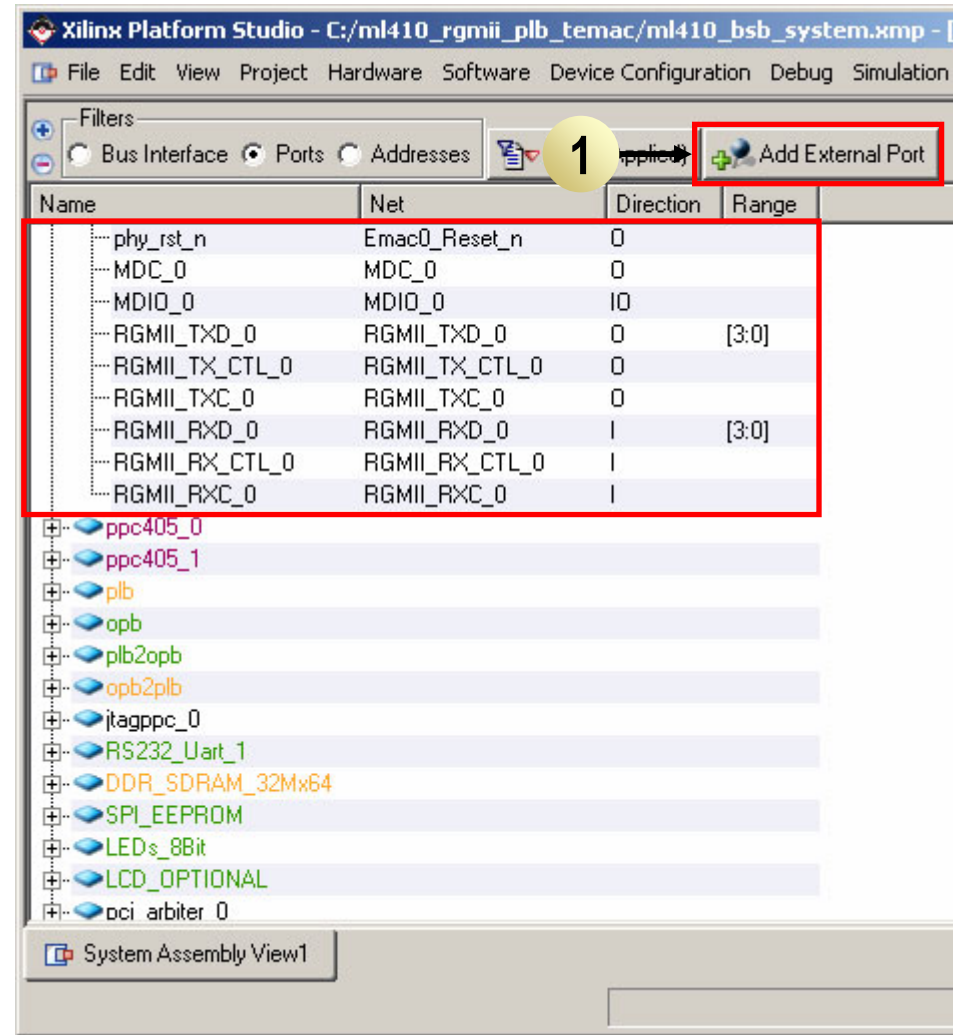
Add RGMII

- Delete all the existing Ethernet ports



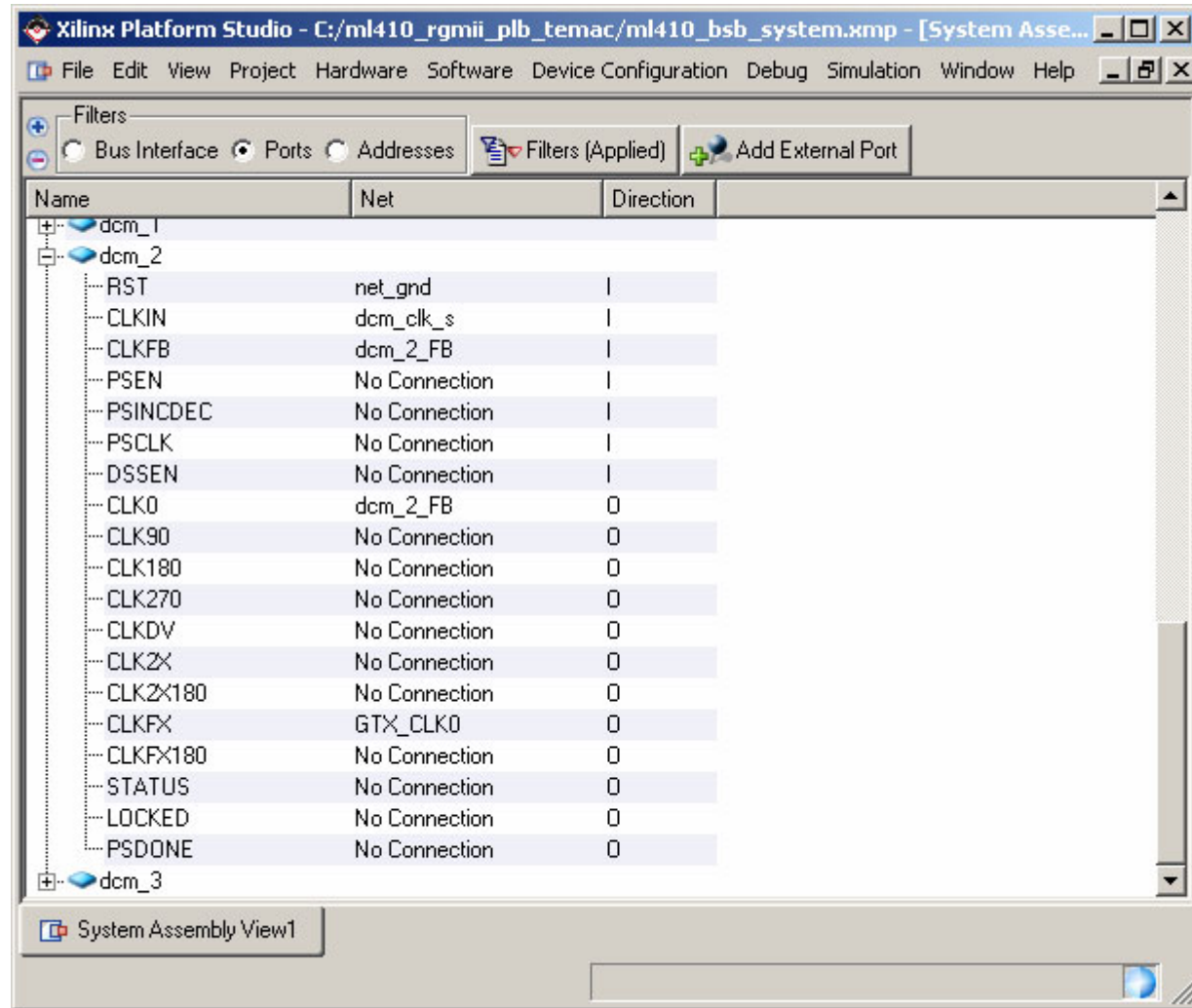
Add RGMII

- 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



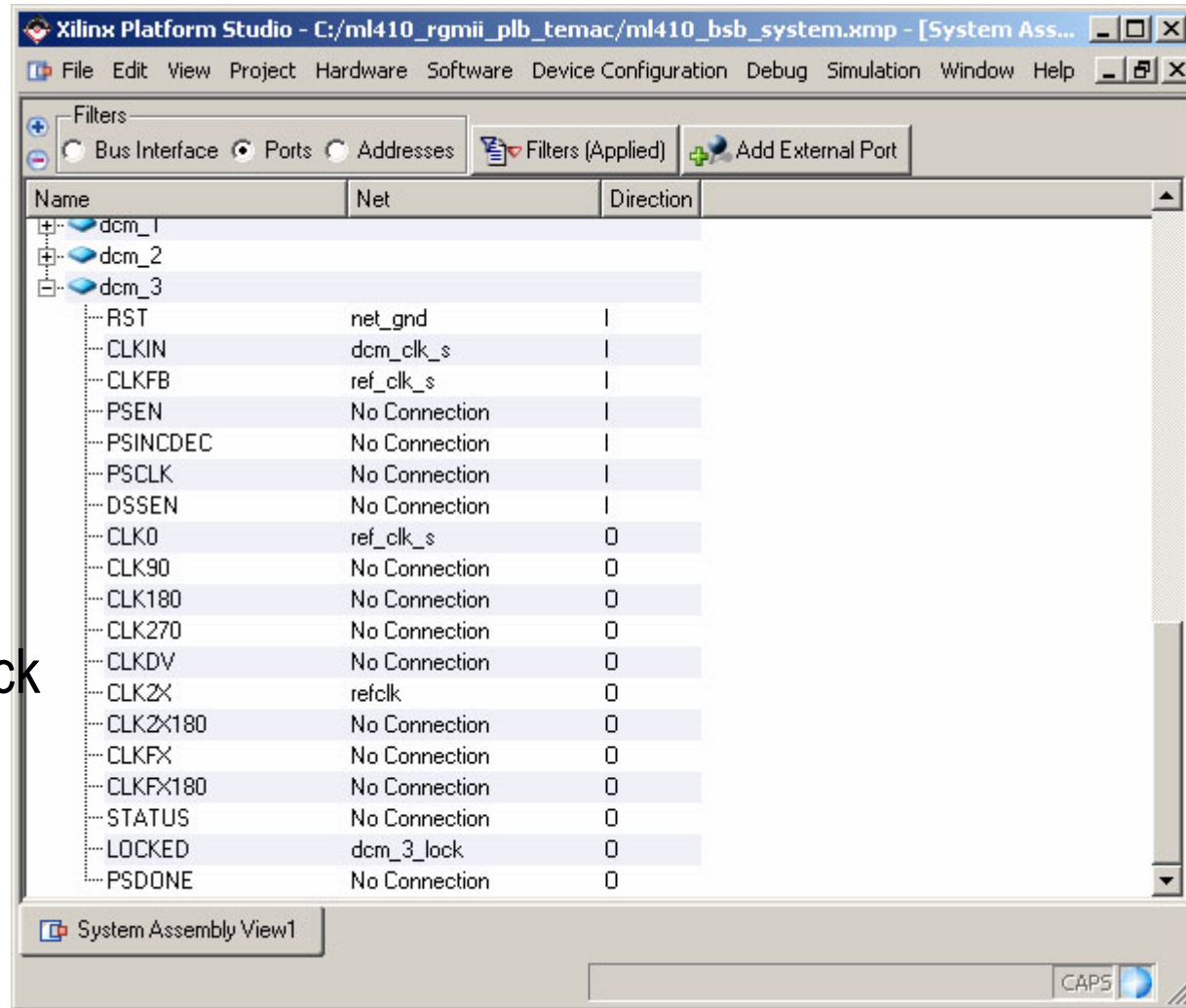
Add RGMII

- 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



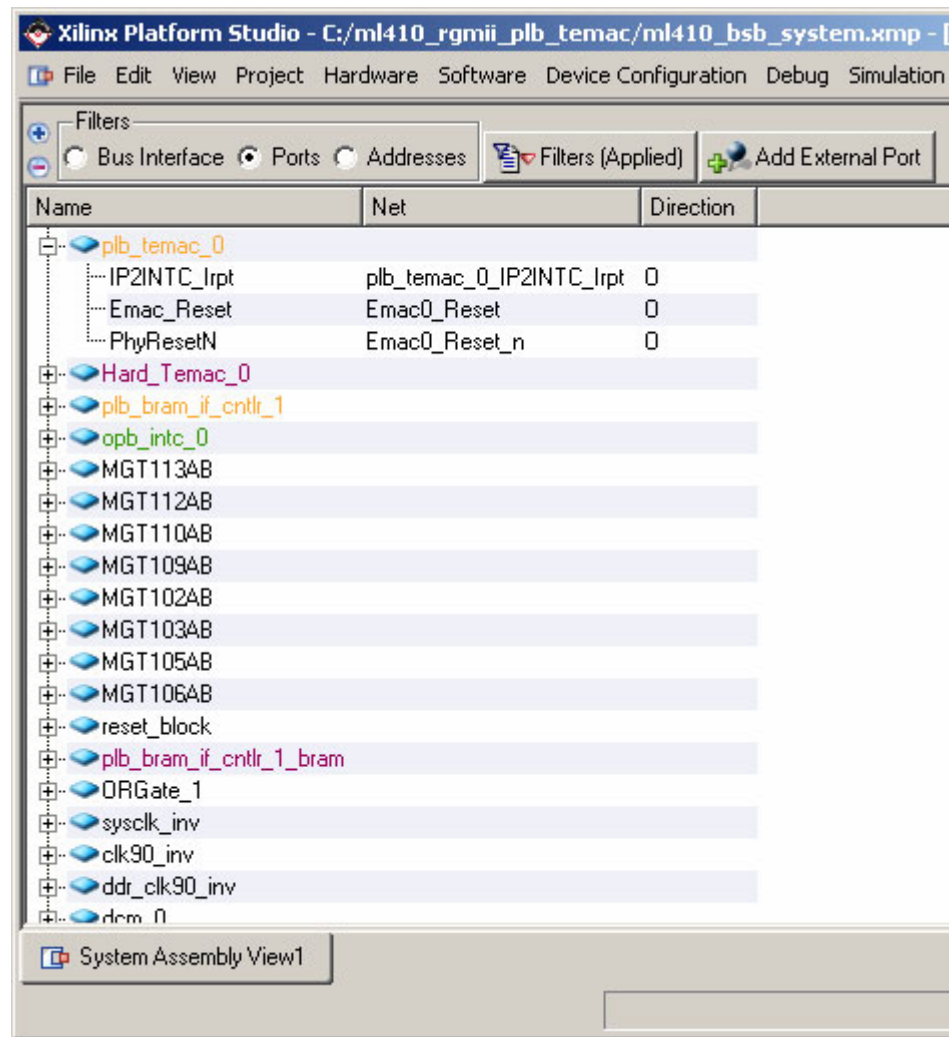
Add RGMII

- 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



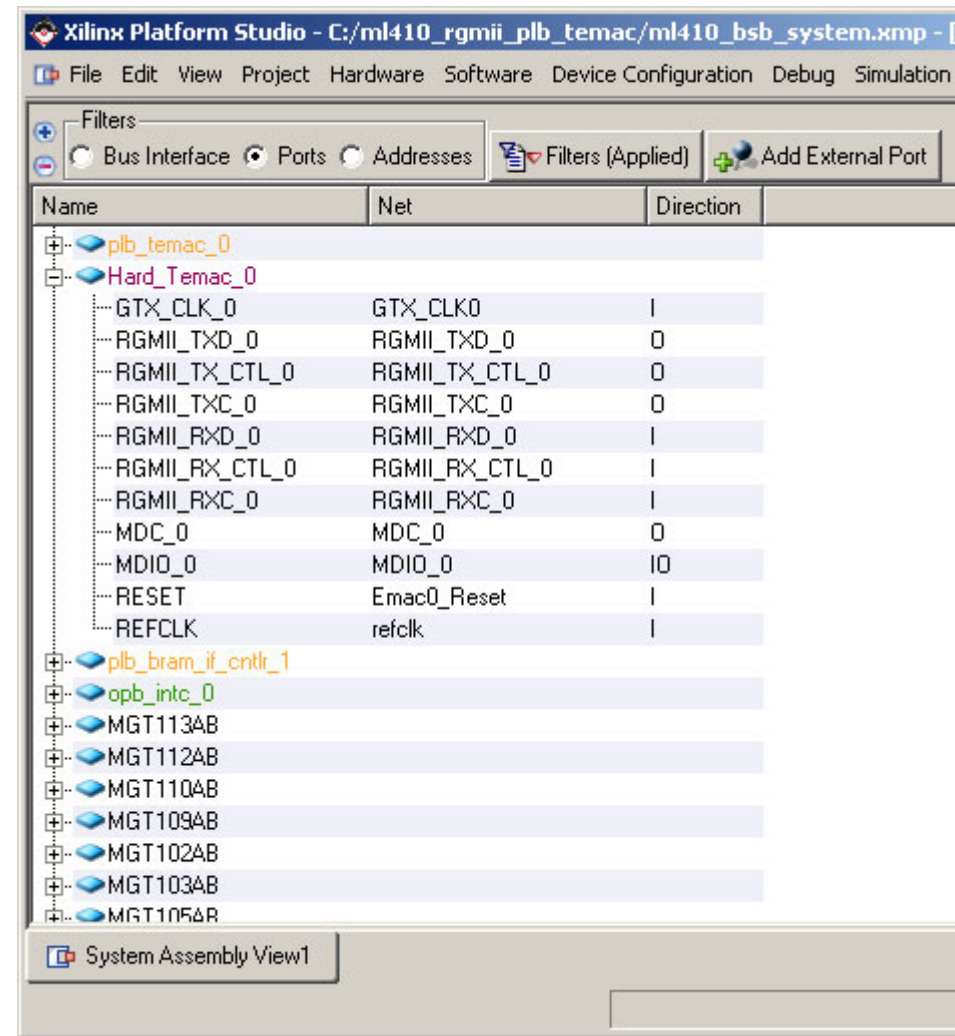
Add RGMII

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



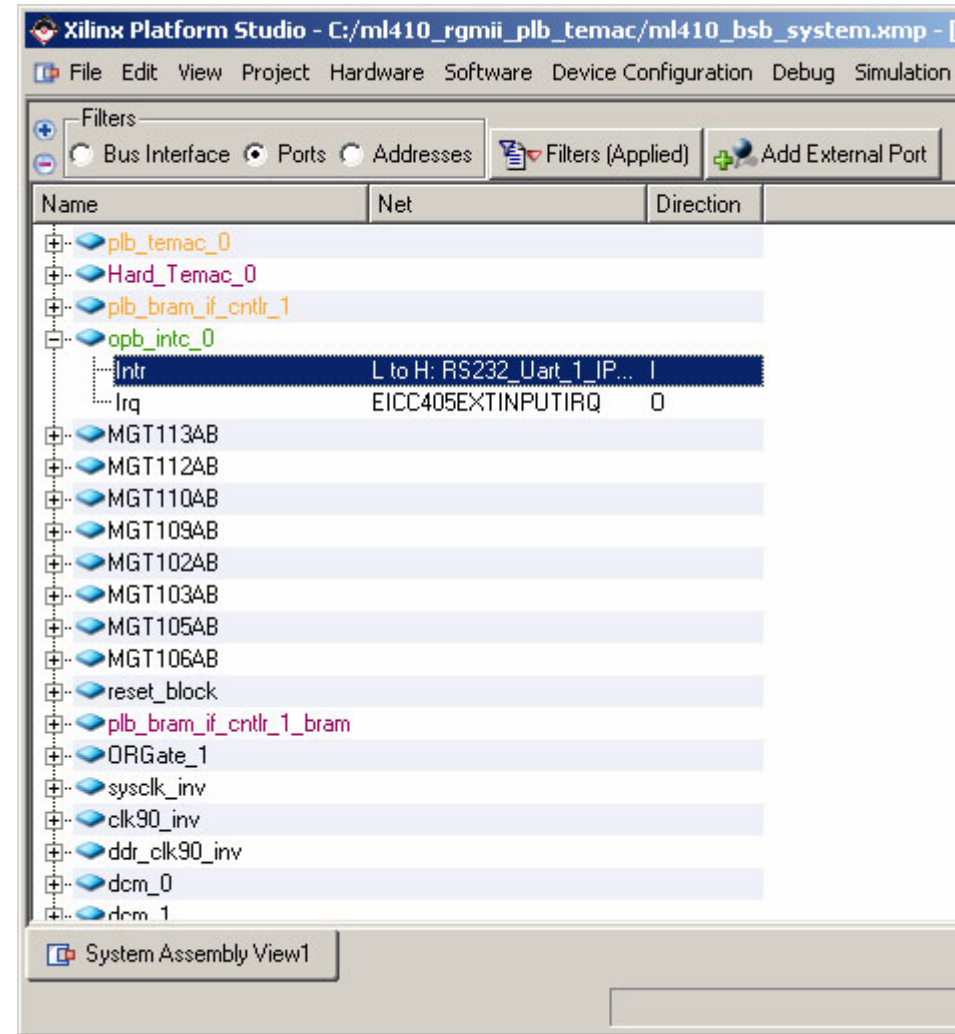
Add RGMII

- 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



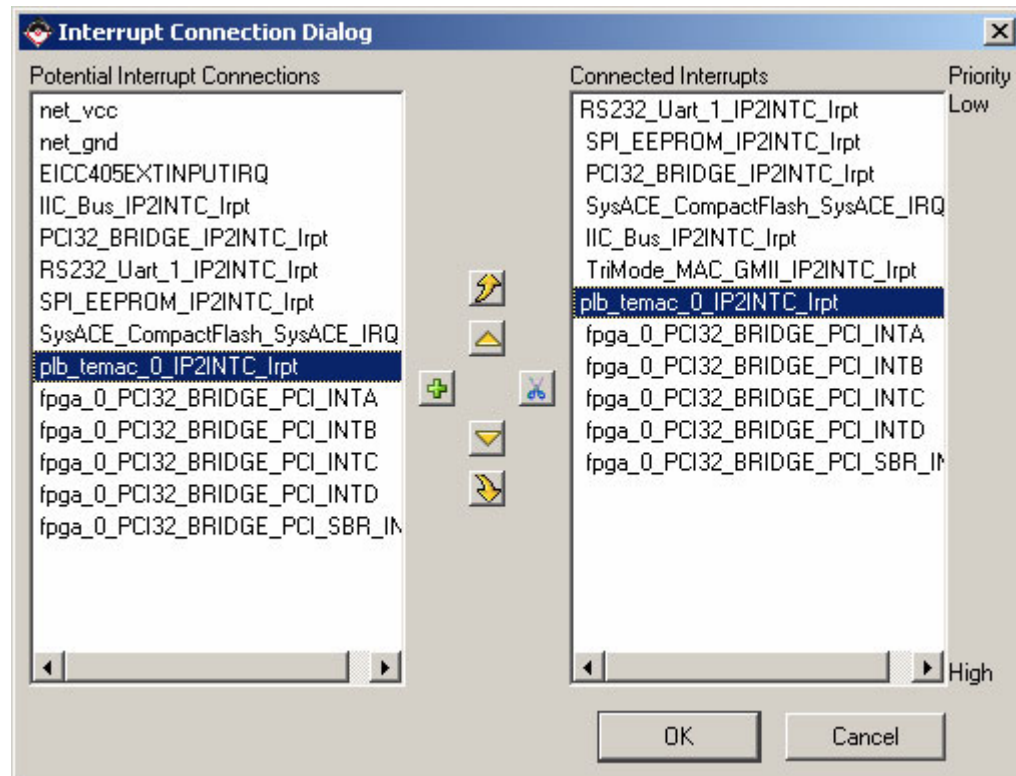
Add RGMII

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



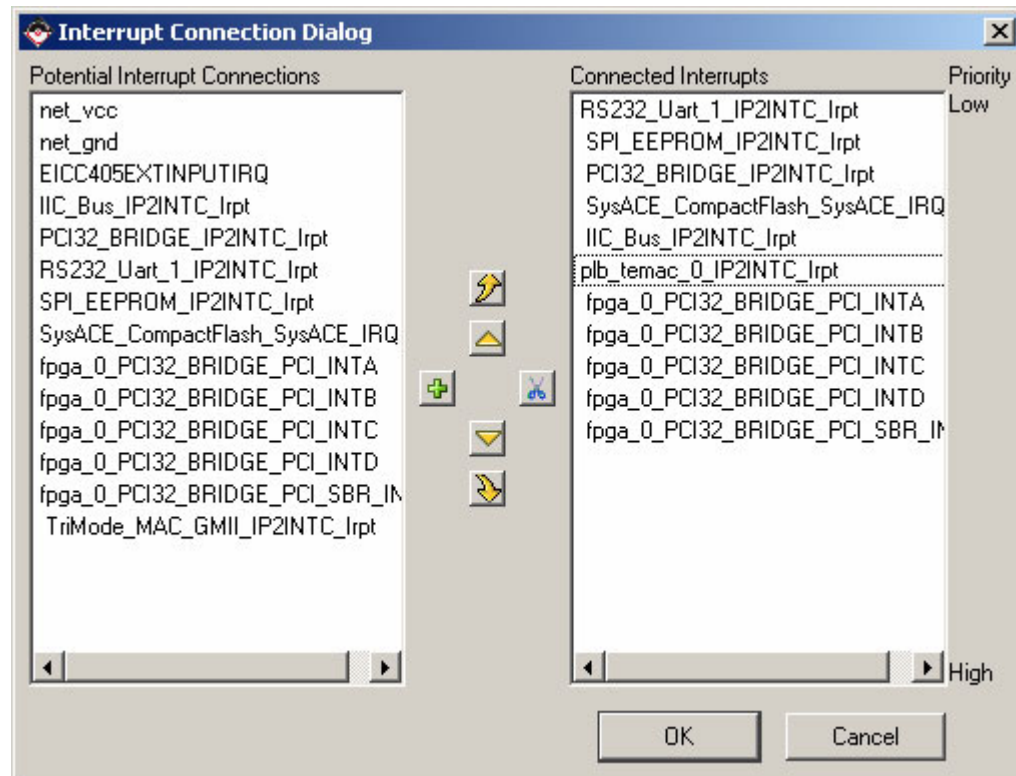
Add RGMII

- Add plb_temac_0_IP2INTC_irpt and move up to the TriMode interrupt



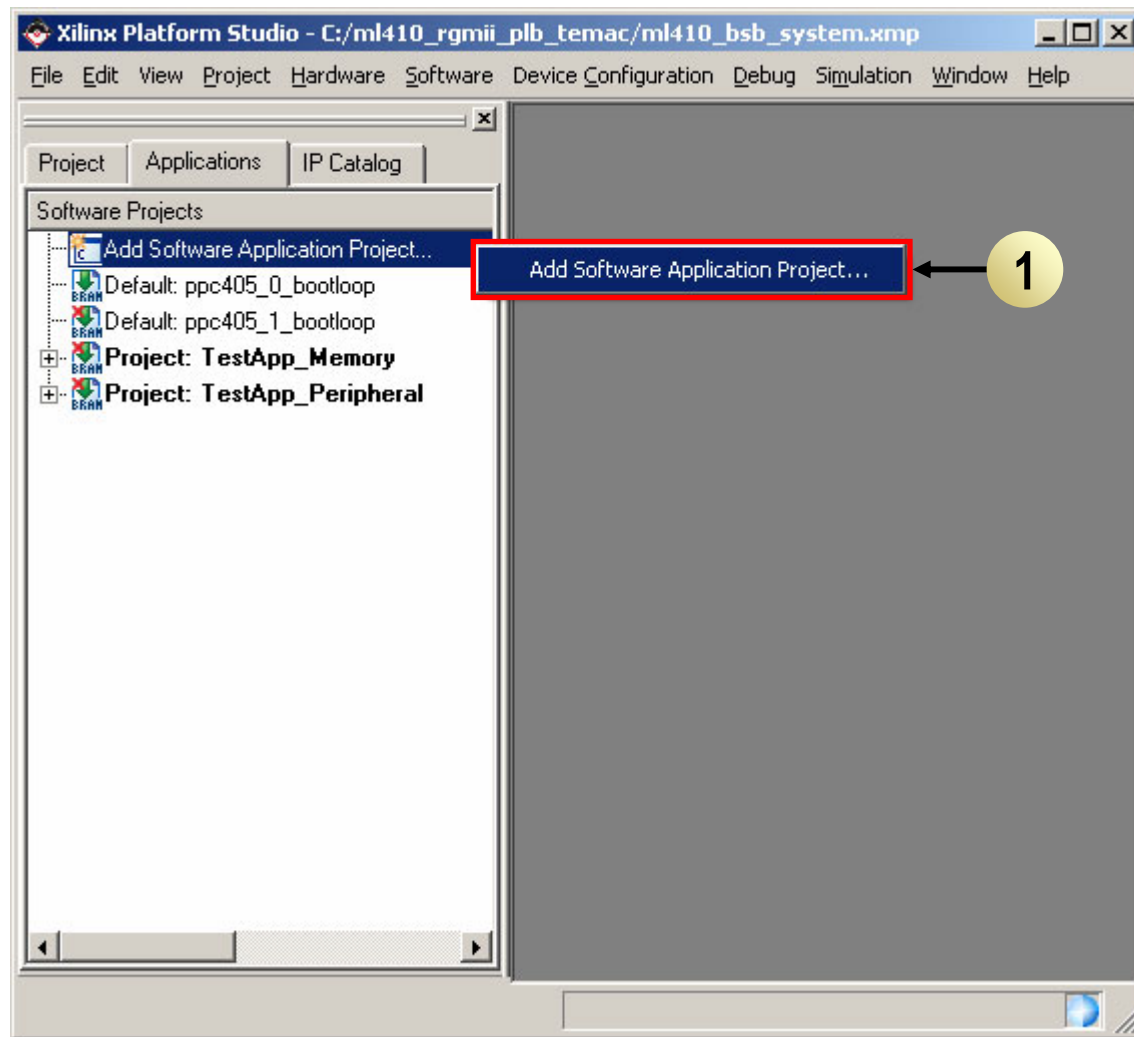
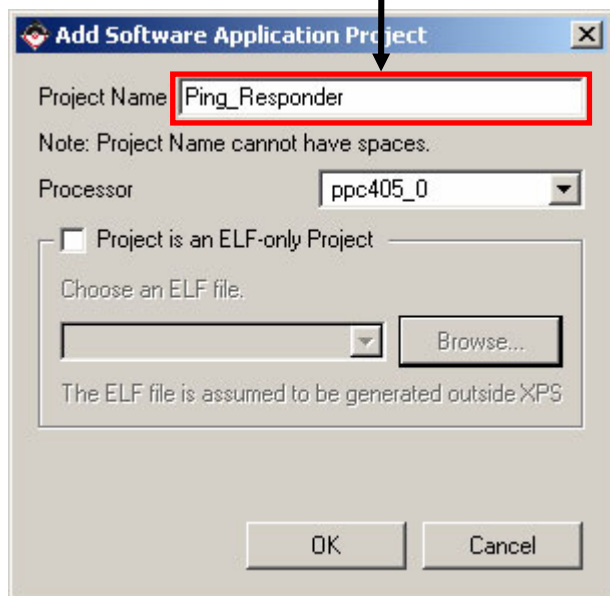
Add RGMII

- Delete the TriMode interrupt



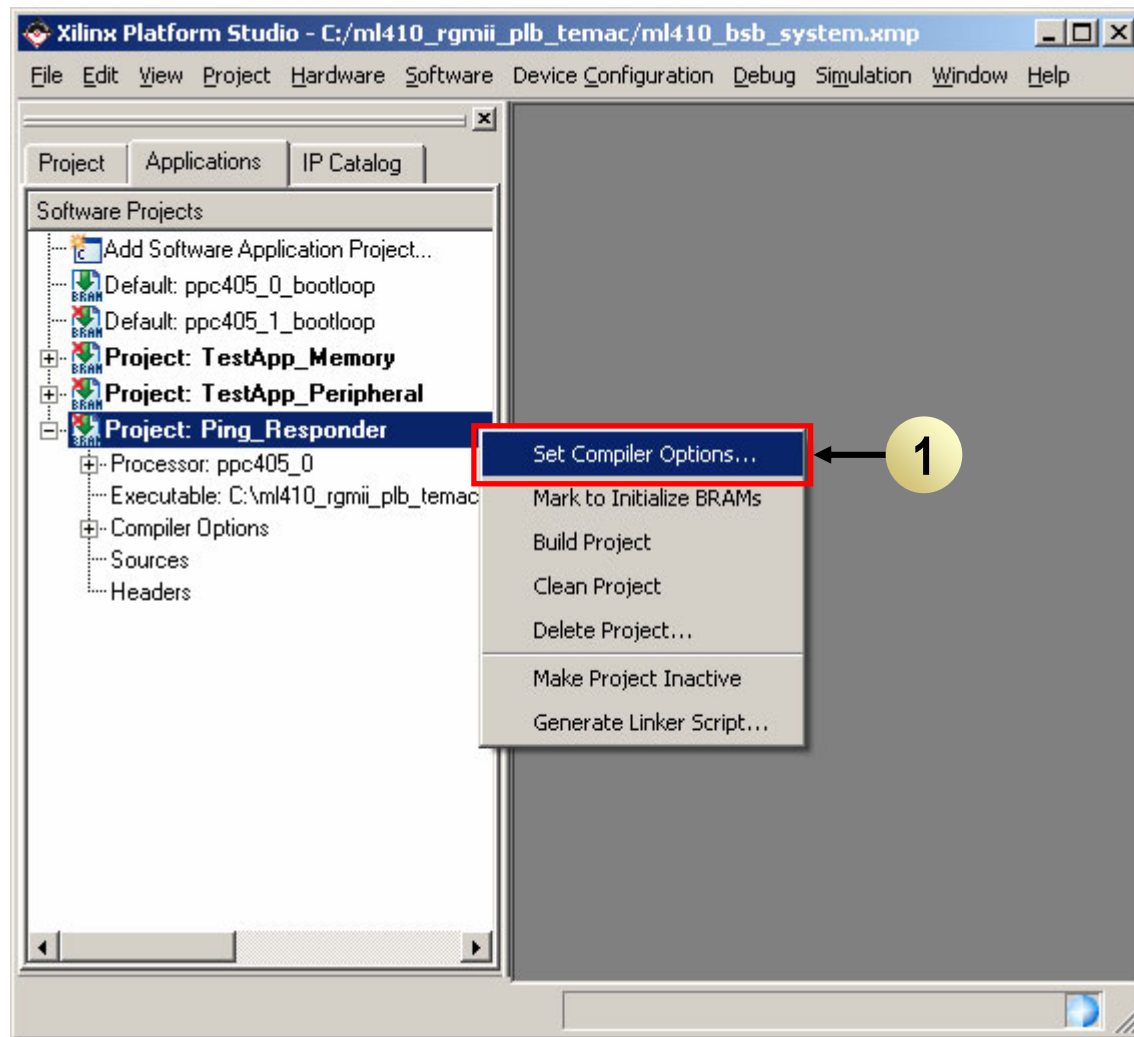
Adding Applications

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



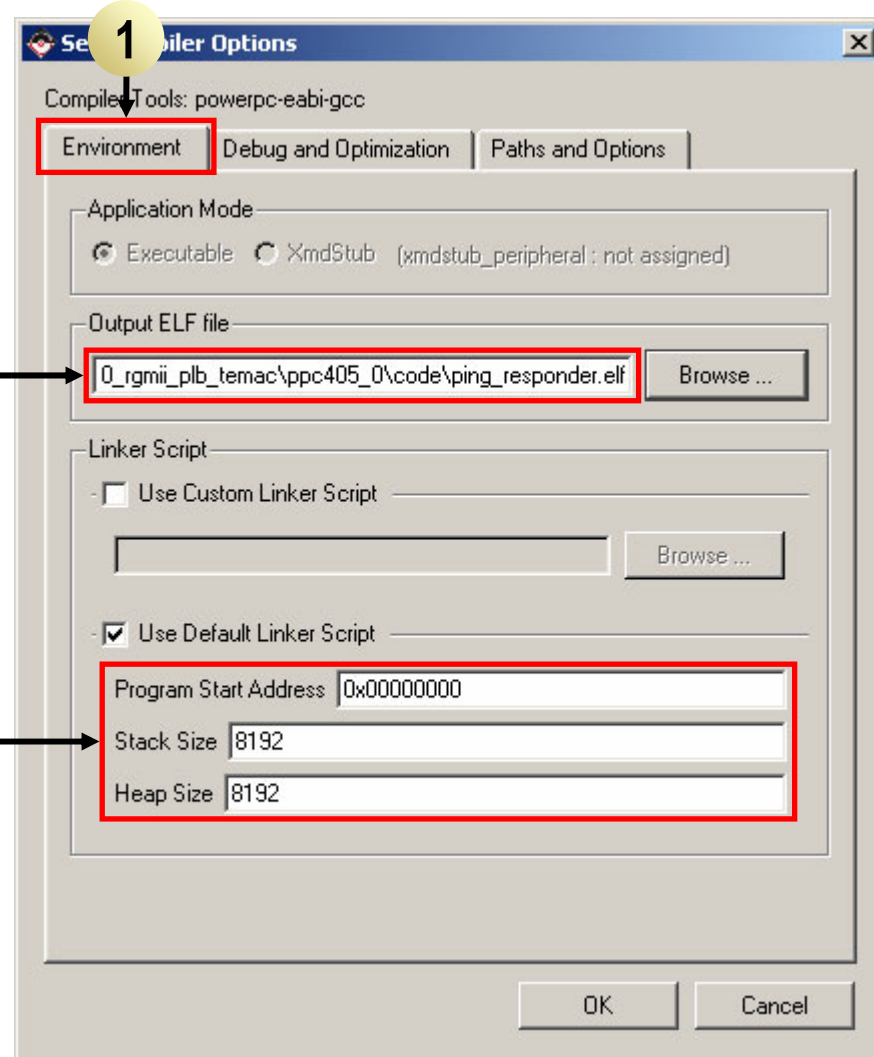
Adding Applications

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



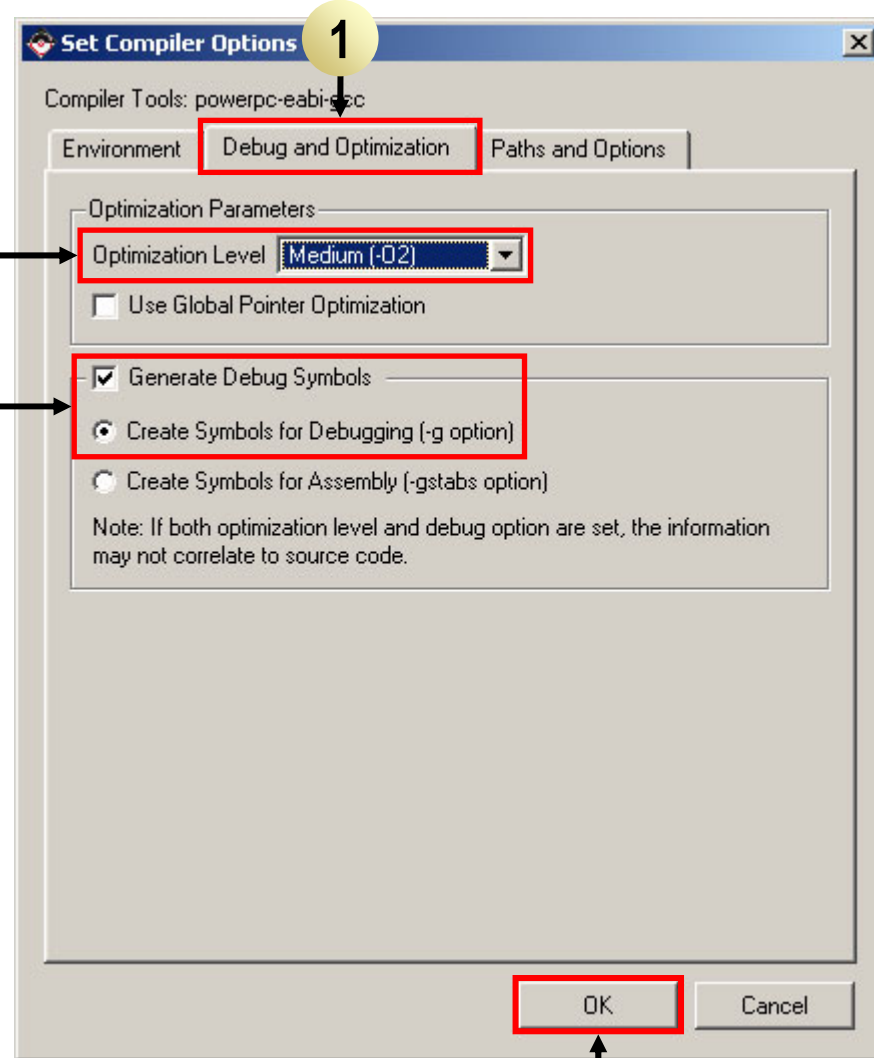
Adding Applications

- 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)



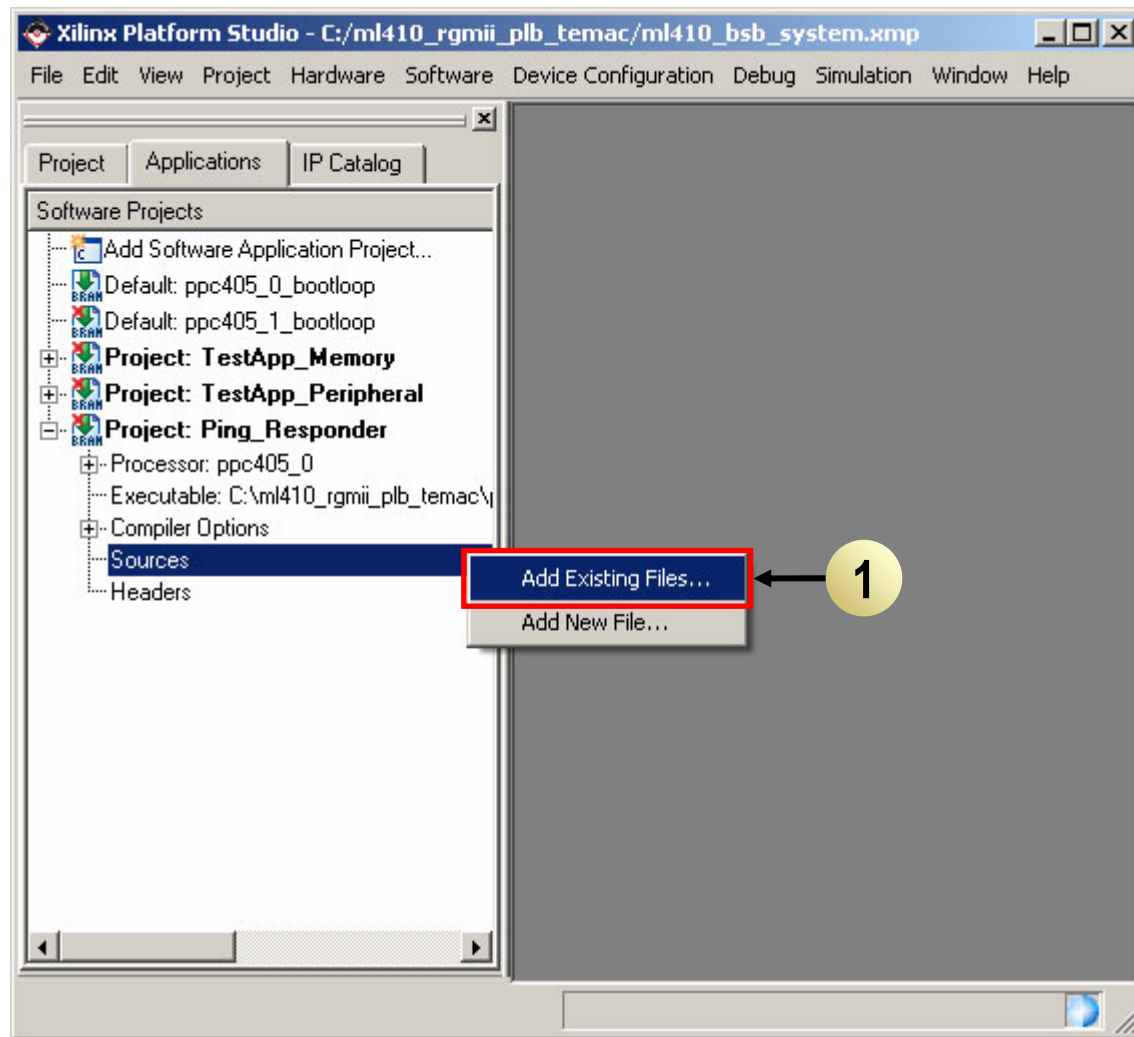
Adding Applications

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



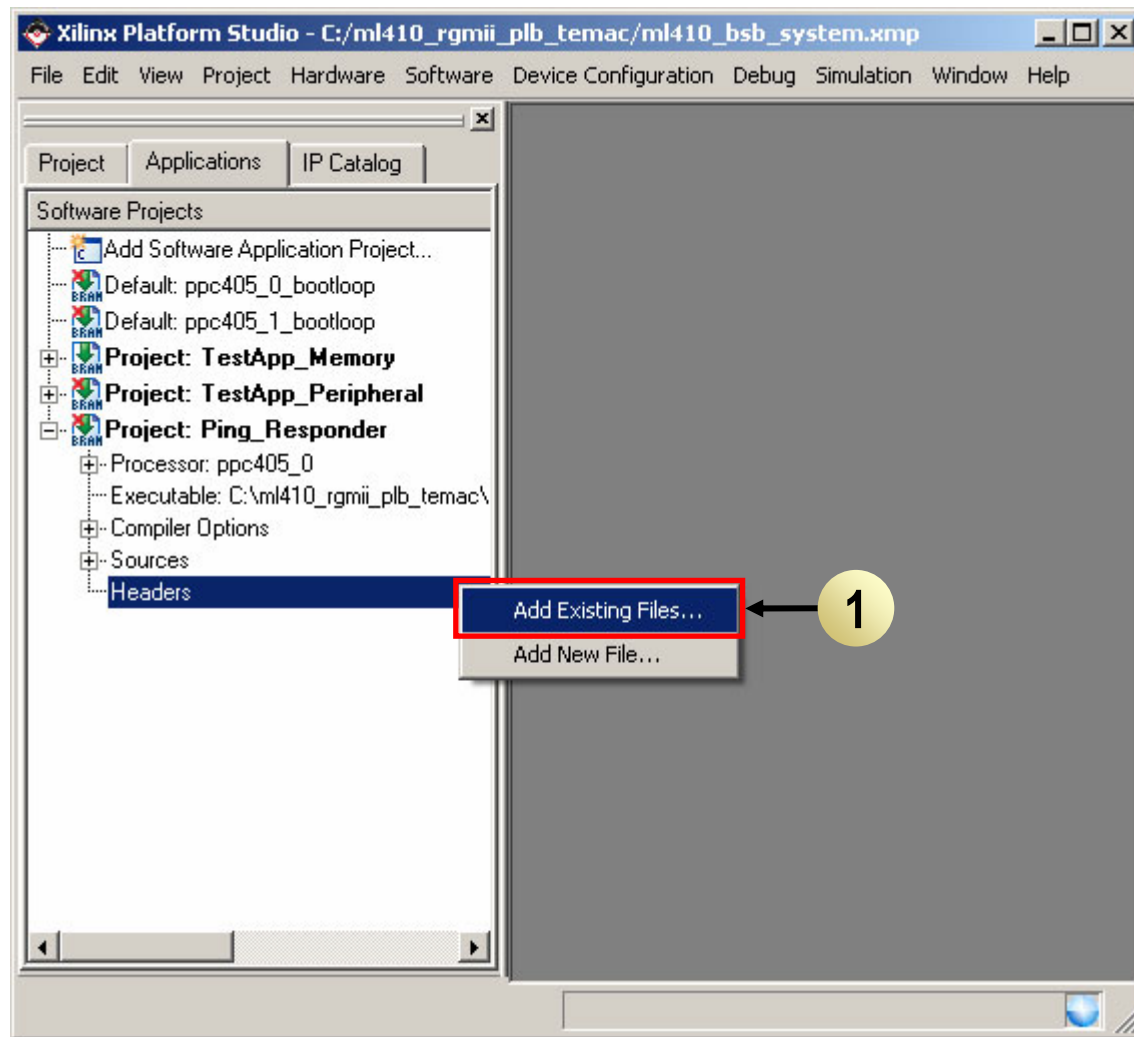
Adding Applications

- 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



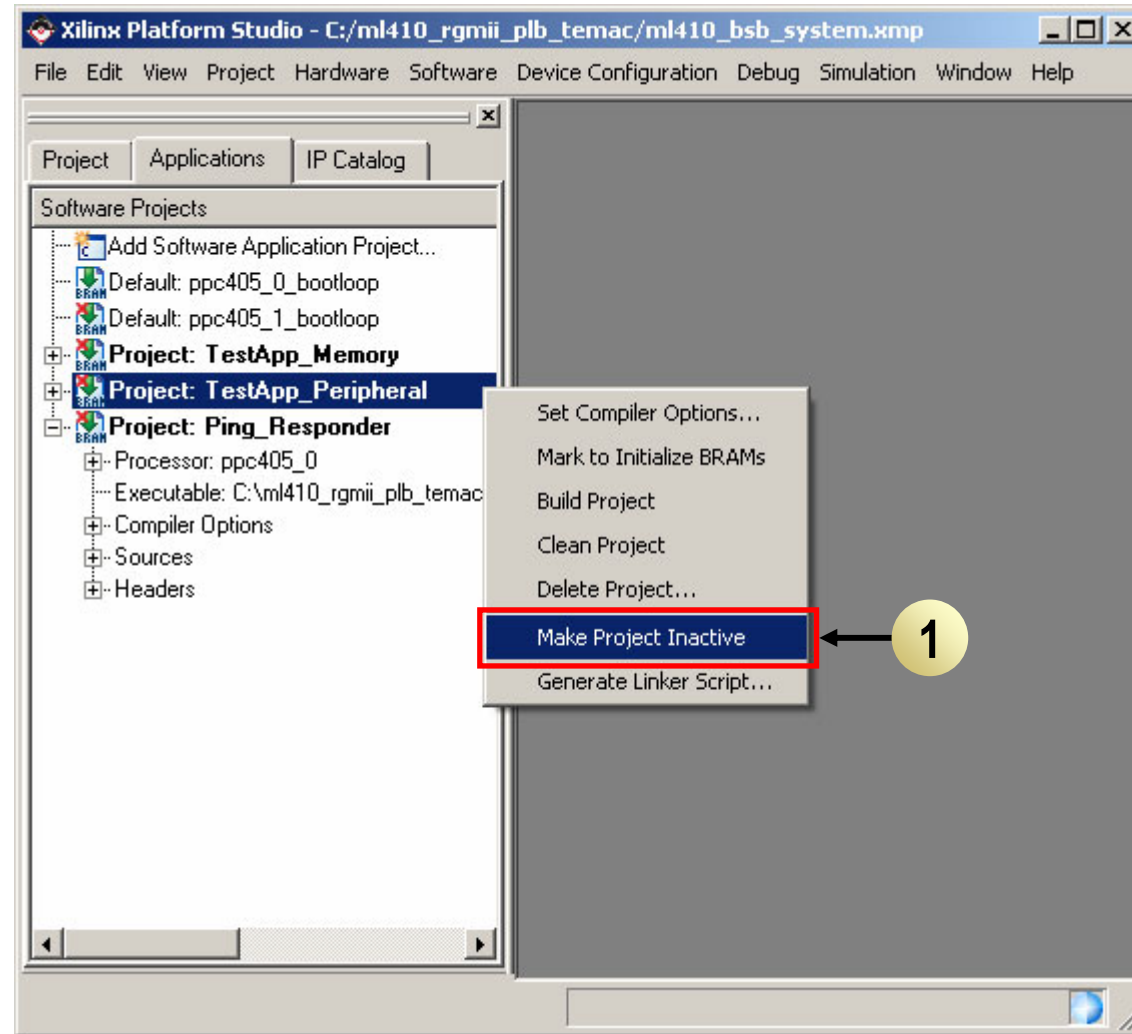
Adding Applications

- 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



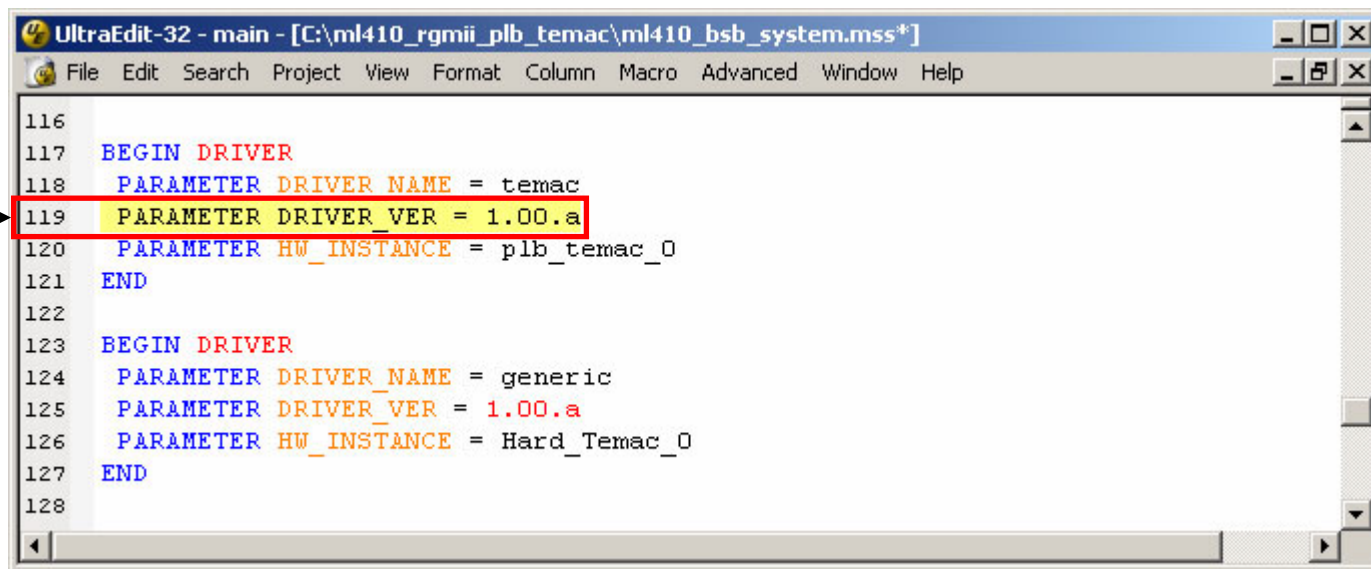
Adding Applications

- 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`

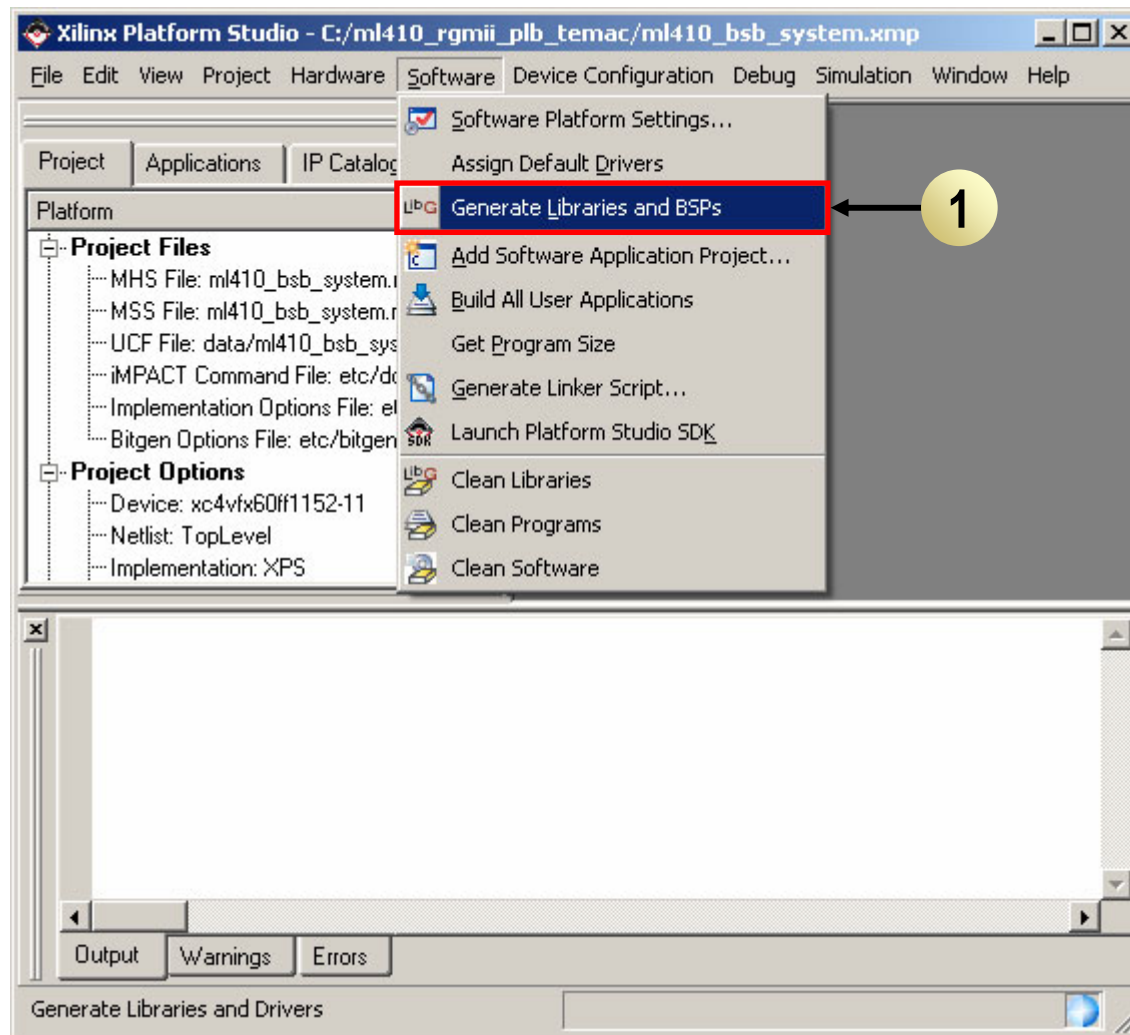


The screenshot shows the UltraEdit-32 editor window with the file `C:\ml410_rgmii_plb_temac\ml410_bsb_system.mss*` open. The editor displays a Verilog-like code snippet. Line 119, `PARAMETER DRIVER_VER = 1.00.a`, is highlighted in yellow and enclosed in a red rectangular box. A yellow circle with the number '1' and an arrow points to this line. The code snippet is as follows:

```
116
117 BEGIN DRIVER
118   PARAMETER DRIVER_NAME = temac
119   PARAMETER DRIVER_VER = 1.00.a
120   PARAMETER HW_INSTANCE = plb_temac_0
121 END
122
123 BEGIN DRIVER
124   PARAMETER DRIVER_NAME = generic
125   PARAMETER DRIVER_VER = 1.00.a
126   PARAMETER HW_INSTANCE = Hard_Temac_0
127 END
128
```

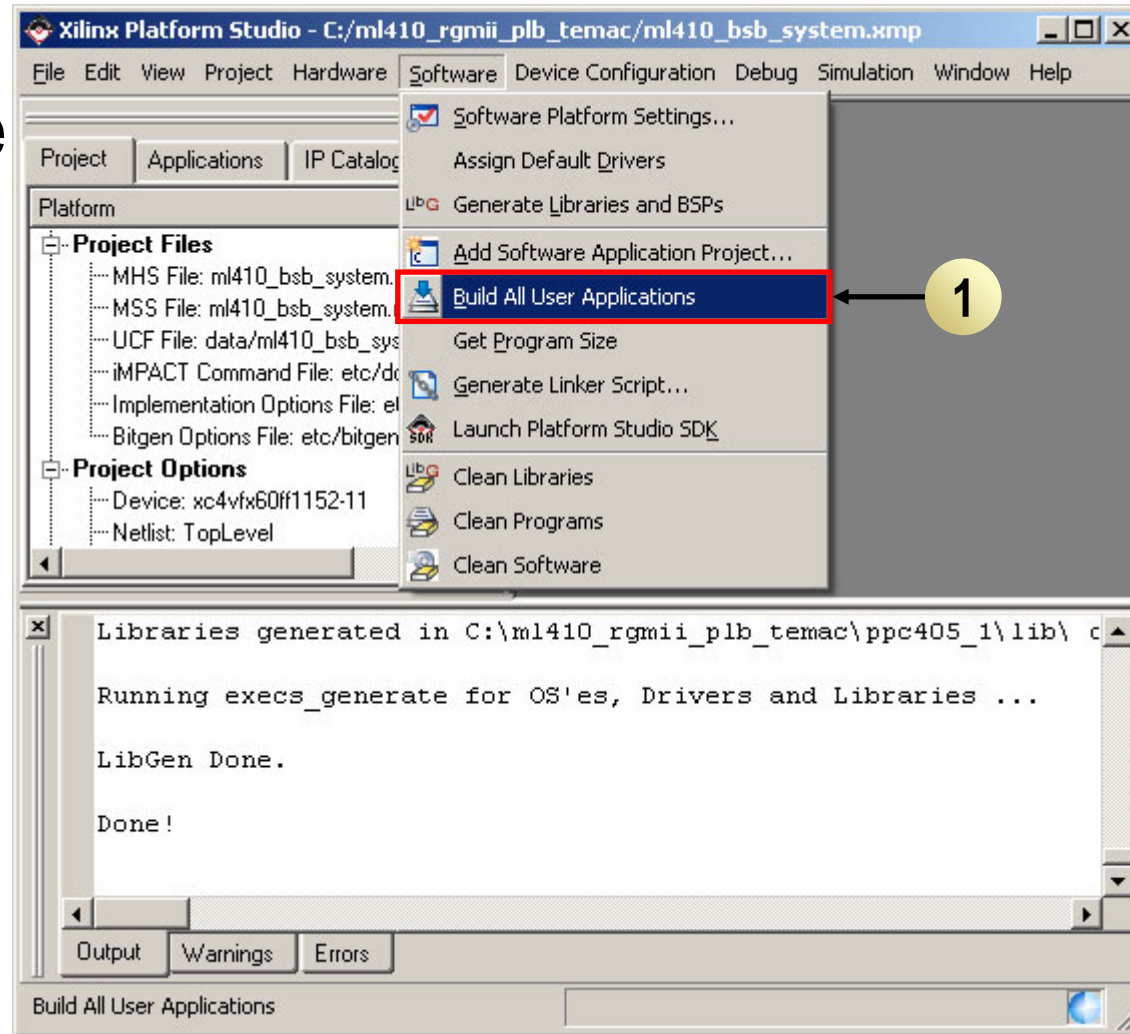
Generate Bitstream

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



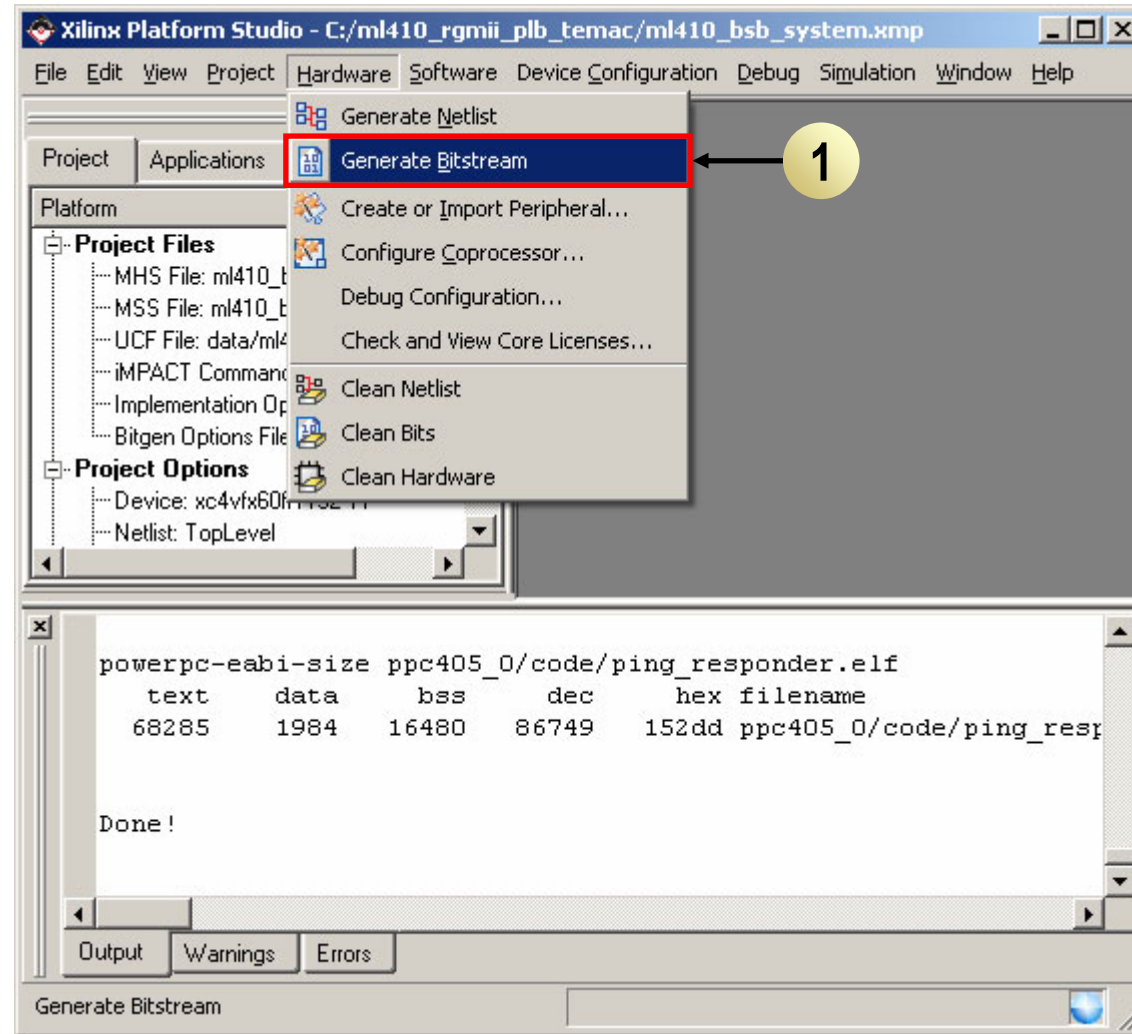
Generate Bitstream

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



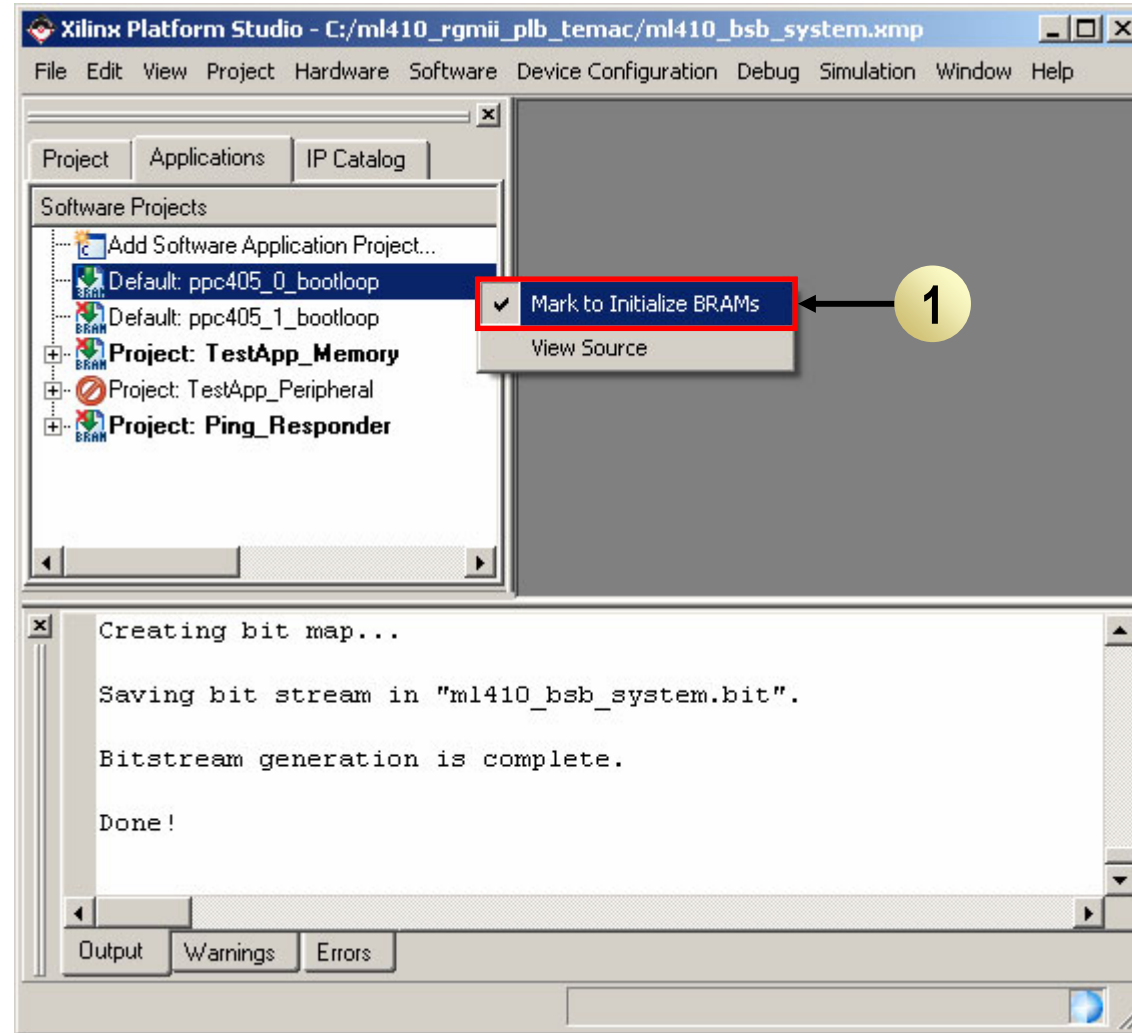
Generate Bitstream

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



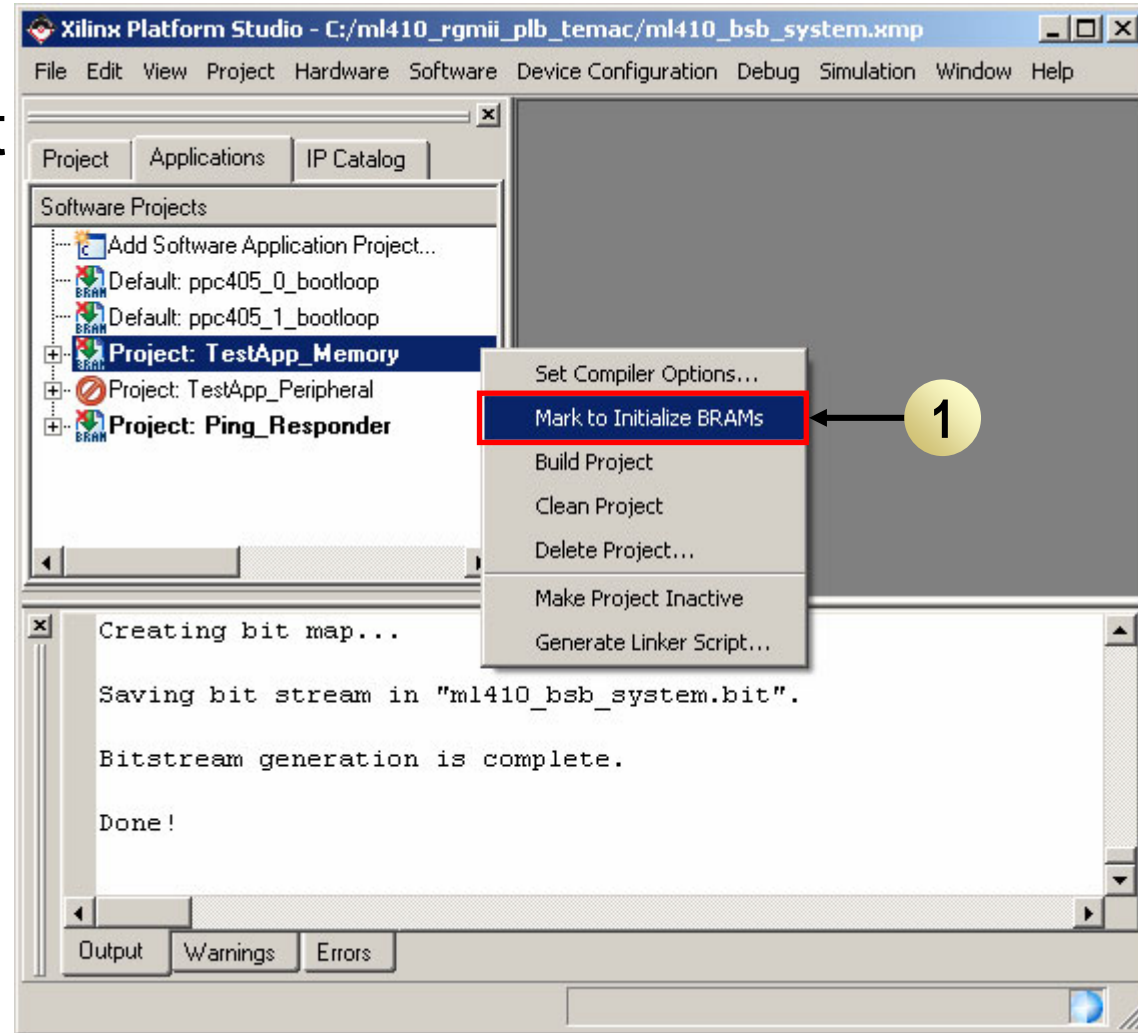
Download the Bitstream

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



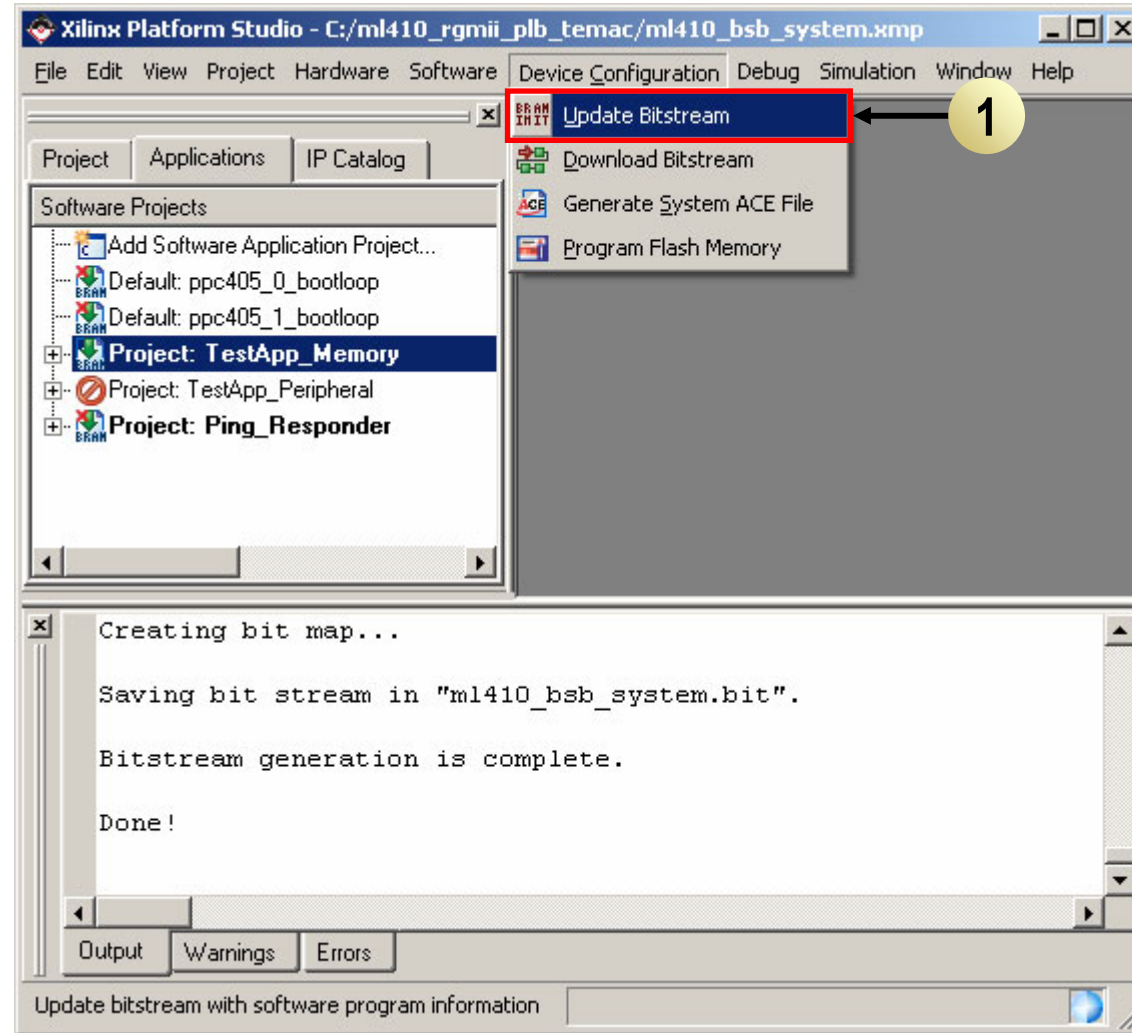
Download the Bitstream

- 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



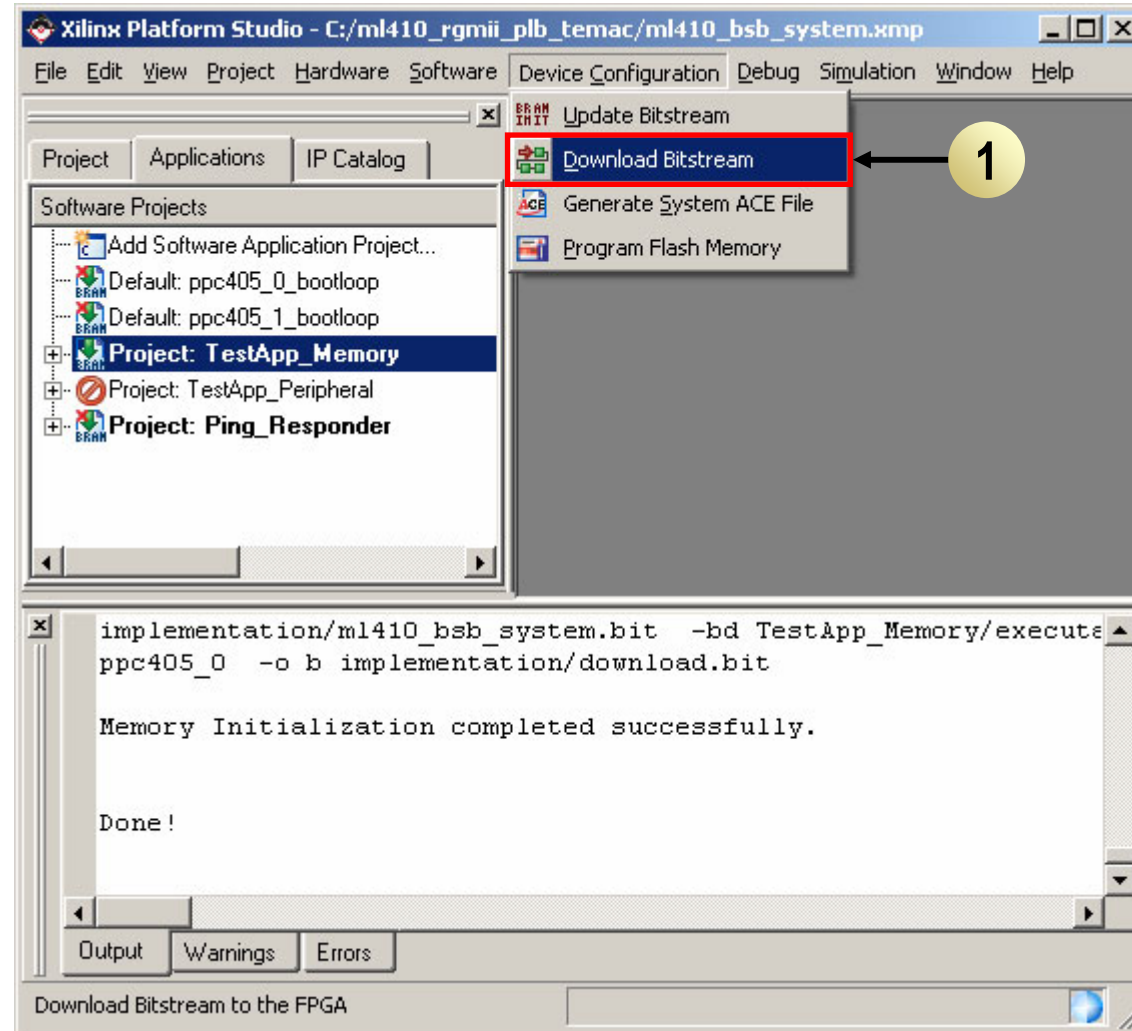
Download the Bitstream

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



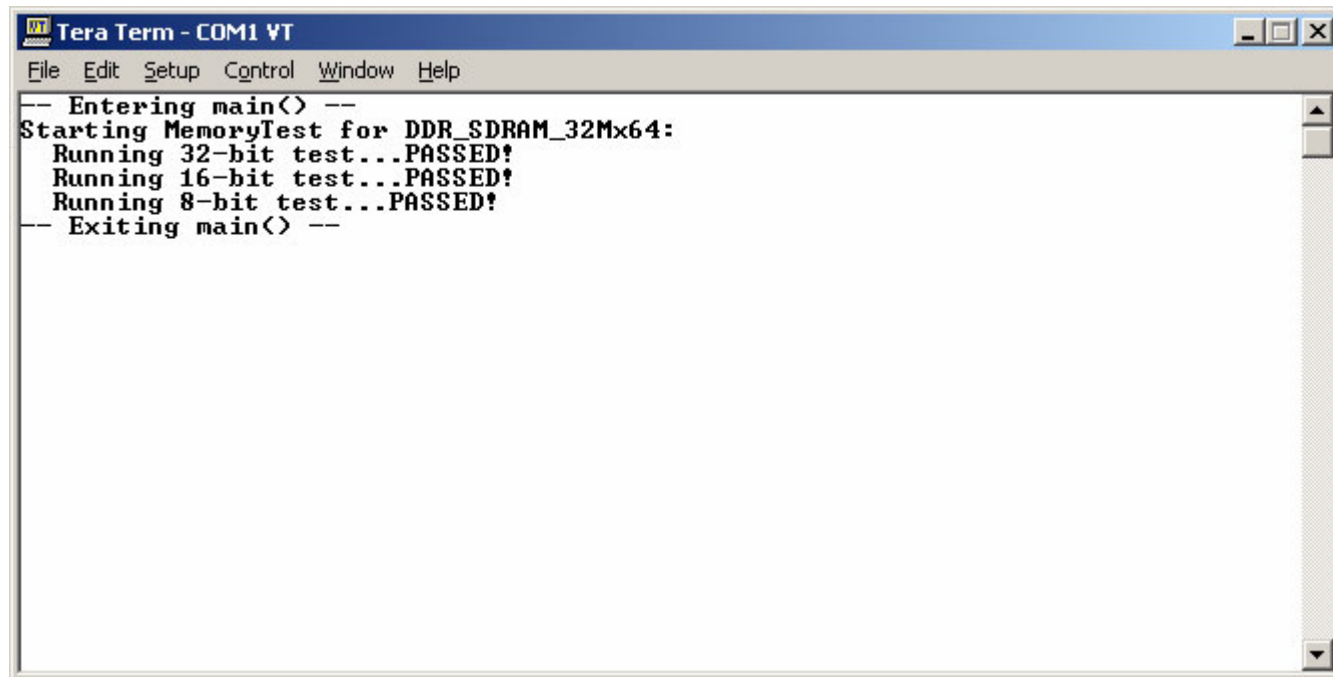
Download the Bitstream

- 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



```
-- Entering main() --  
Starting MemoryTest for DDR_SDRAM_32Mx64:  
Running 32-bit test...PASSED!  
Running 16-bit test...PASSED!  
Running 8-bit test...PASSED!  
-- Exiting main() --
```

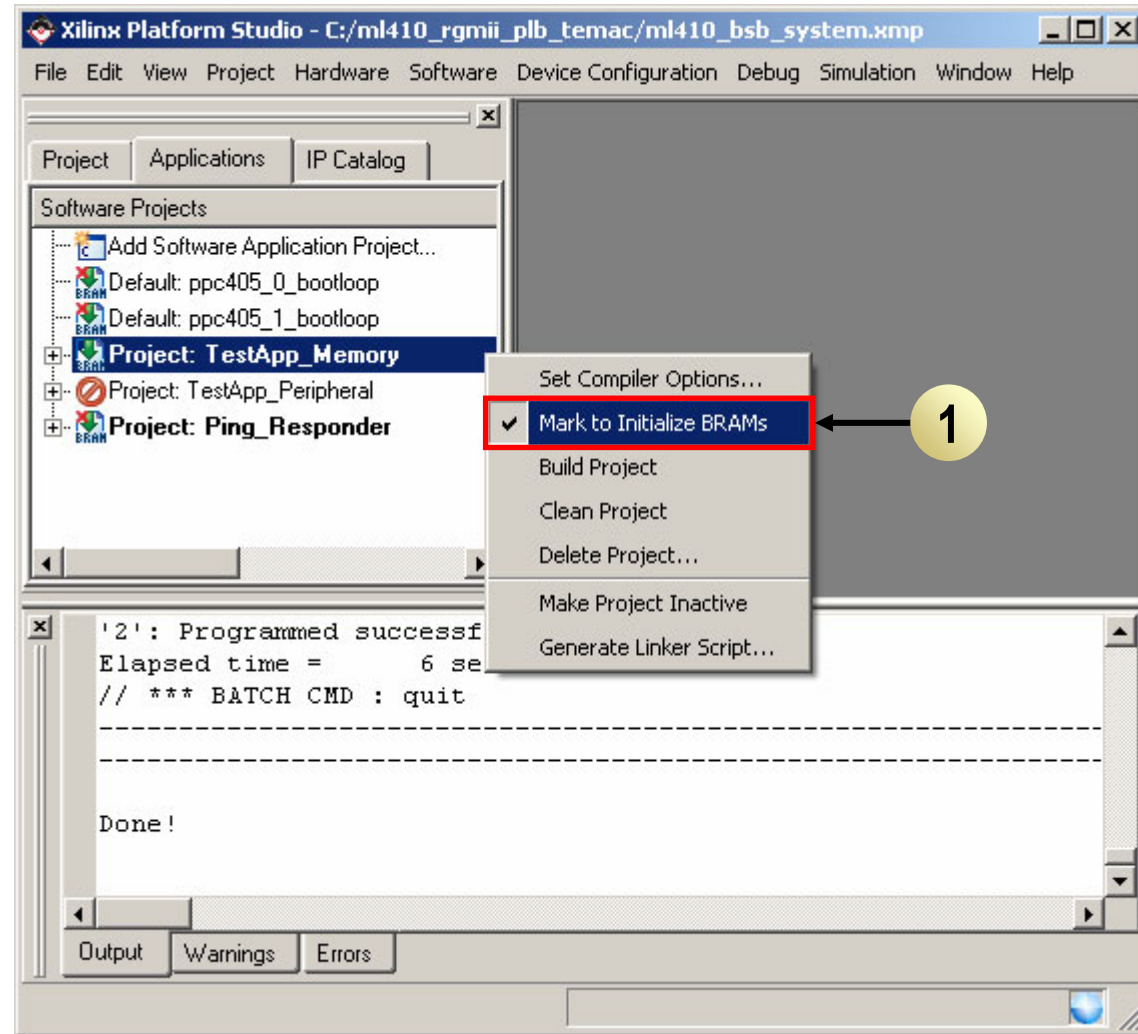
Loading Bootloop into BRAM

- 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



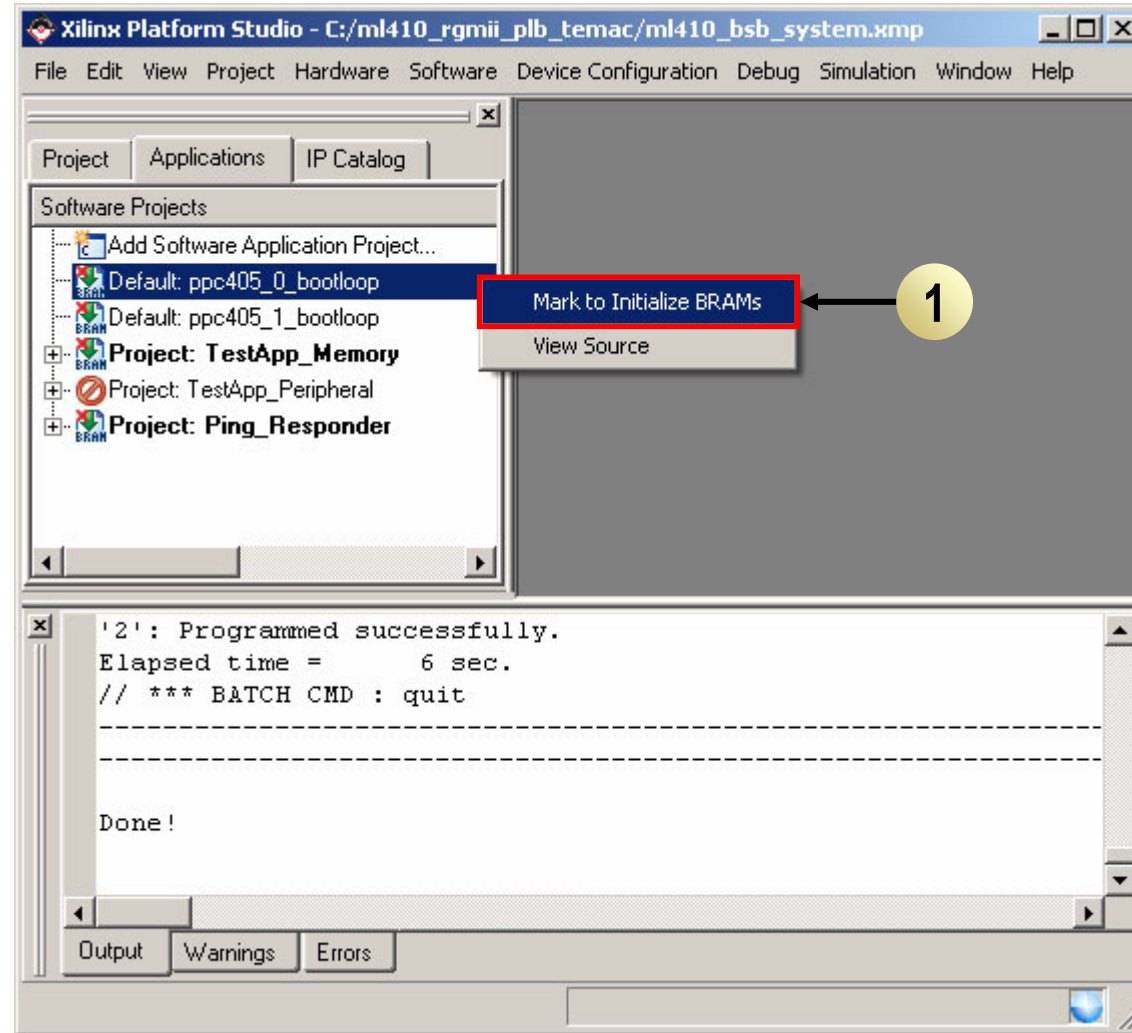
Loading Bootloop into BRAM

- 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



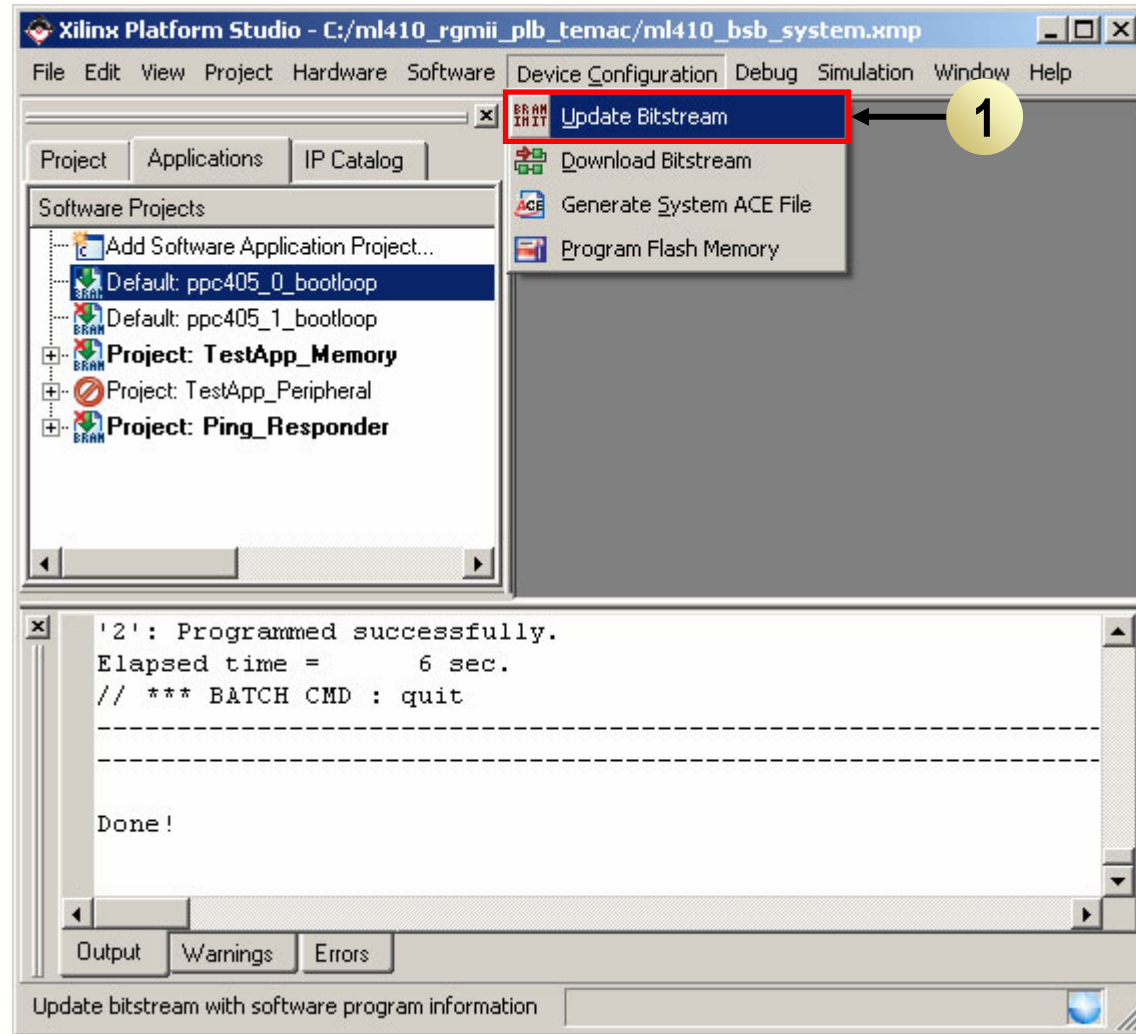
Loading Bootloop into BRAM

- 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



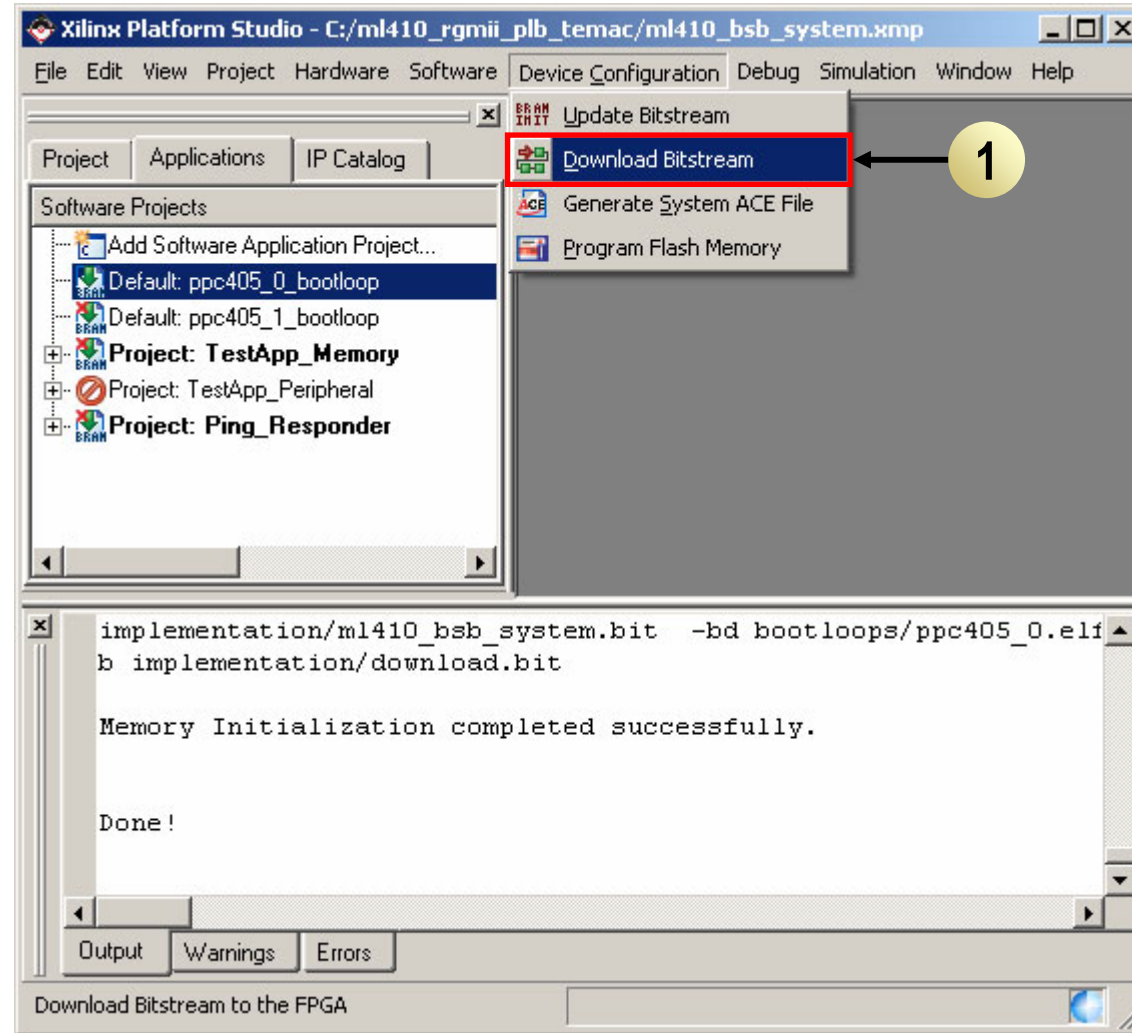
Loading Bootloop into BRAM

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



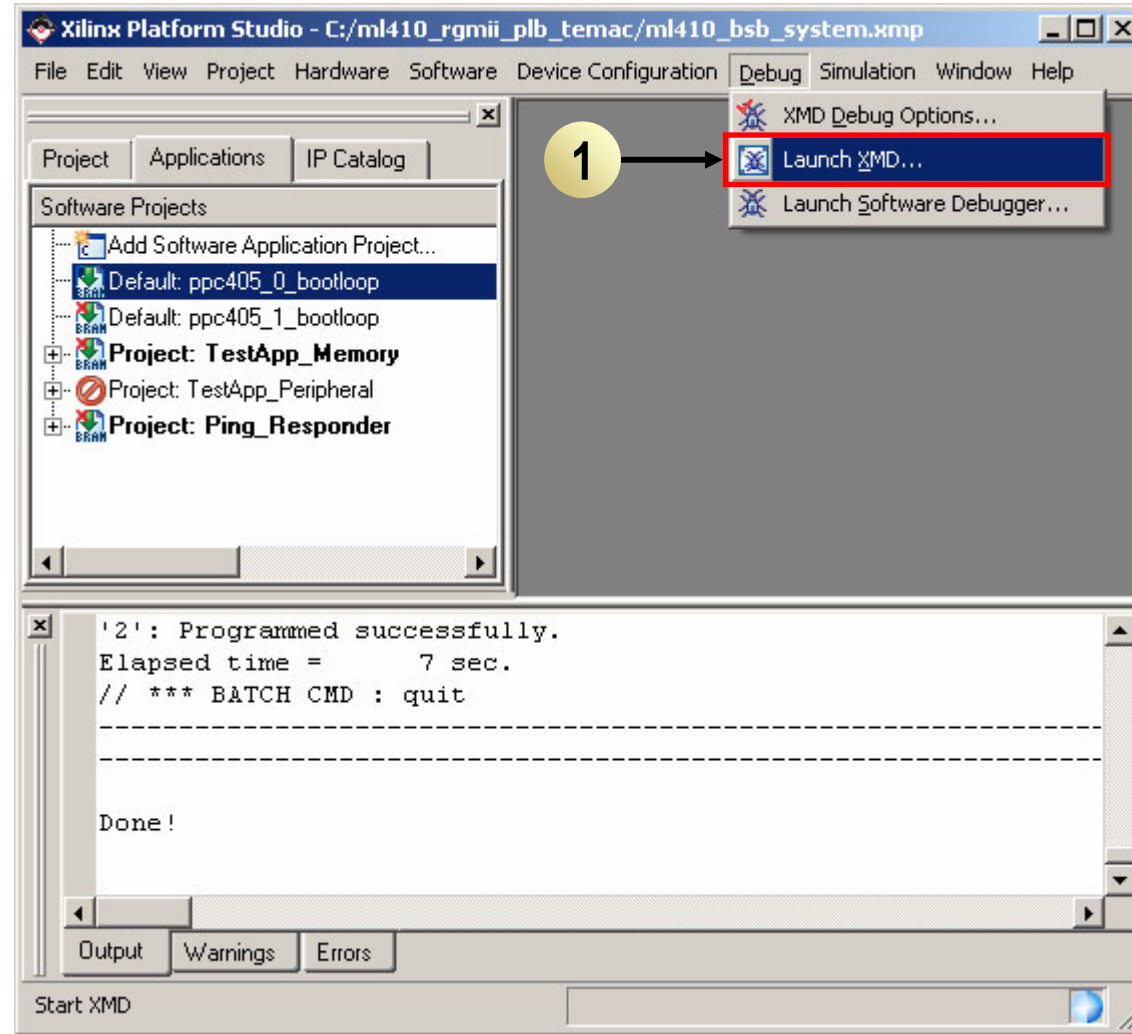
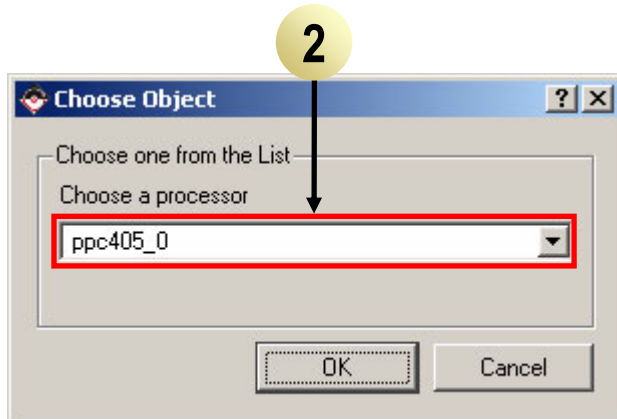
Loading Bootloop into BRAM

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



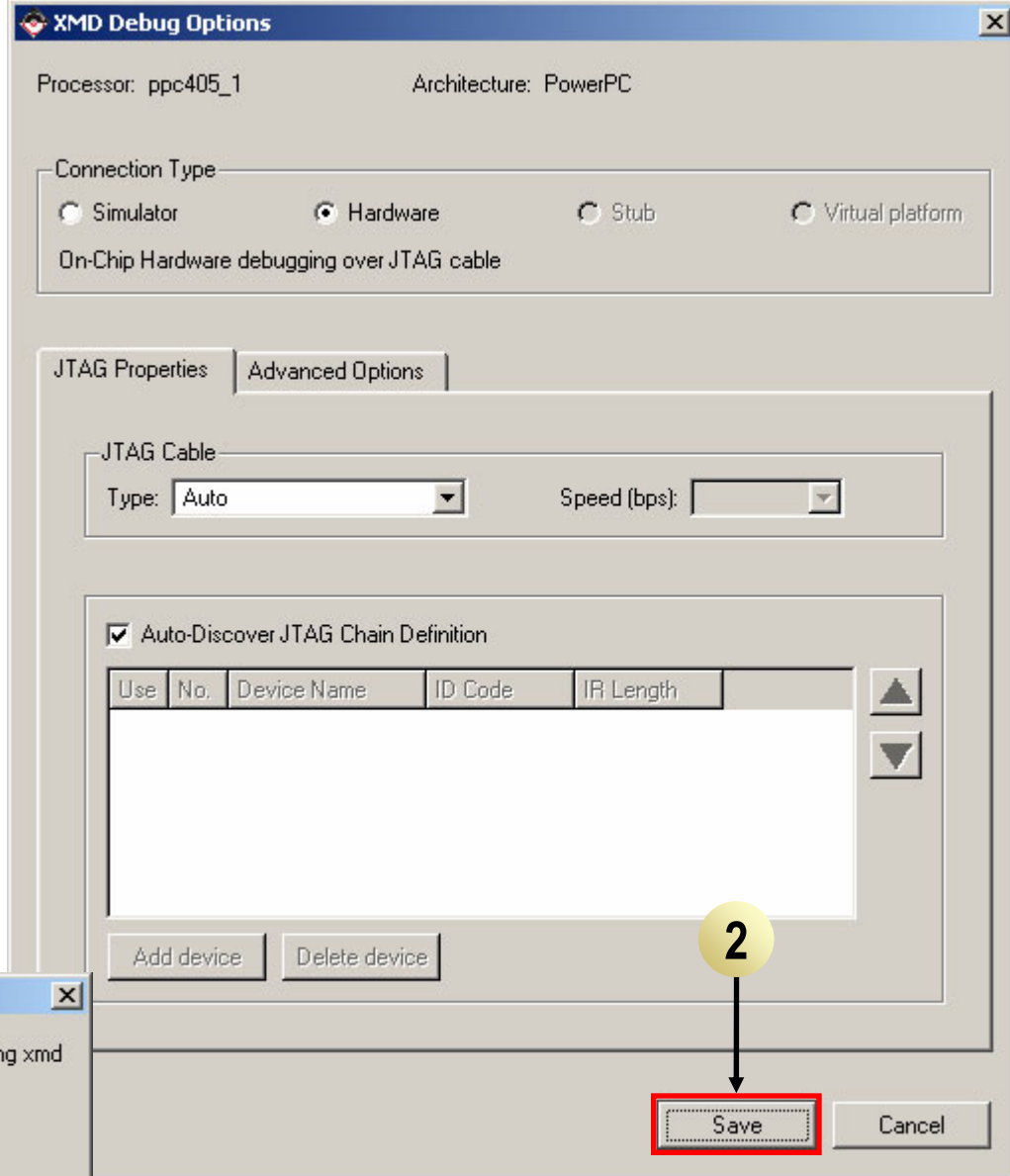
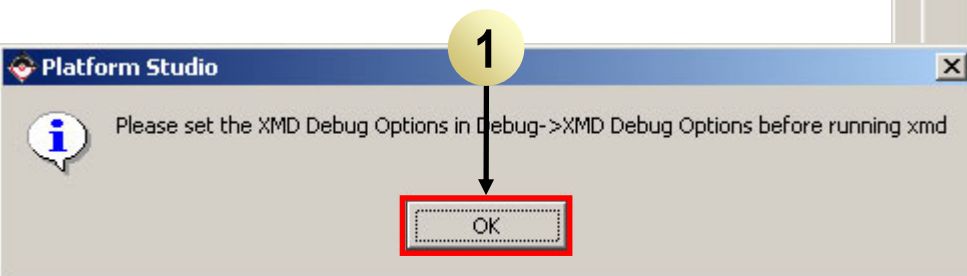
Loading Bootloop into BRAM

- 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

```
C:\EDK_Im_Sp1.3.2\bin\nt\xmd.exe
Driver xpc4drv.sys version = 1.0.4.0. LPT base address = 0378h.
Cable Type = 1, Revision = 3.
Setting cable speed to 5 MHz.
Cable connection established.
INFO:MDT - Assumption: Selected Device 2 for debugging.

JTAG chain configuration
-----
Device  ID Code      IR Length  Part Name
  1      0a001093      8      System_ACE
  2      01eb4093     14      XC4VFX60

XMD: Connected to PowerPC target. Processor Version No : 0x20011470
Address mapping for accessing special PowerPC features from XMD/GDB:
  I-Cache <Data> : Disabled
  I-Cache <Tag>  : Disabled
  D-Cache <Data> : Disabled
  D-Cache <Tag>  : Disabled
  ISOCM          : Disabled
  TLB            : Disabled
  DCR            : Disabled

Connected to "ppc" target. id = 0
Starting GDB server for "ppc" target <id = 0> at TCP port no 1234
XMD%
```

Loading Bootloop into BRAM

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

```
C:\EDK_Im_Sp1.3.2\bin\nt\xmd.exe
Cable connection established.
INFO:MDT - Assumption: Selected Device 2 for debugging.

JTAG chain configuration
-----
Device  ID Code      IR Length  Part Name
  1      0a001093         8  System_ACE
  2      01eb4093        14  XC4VFX60

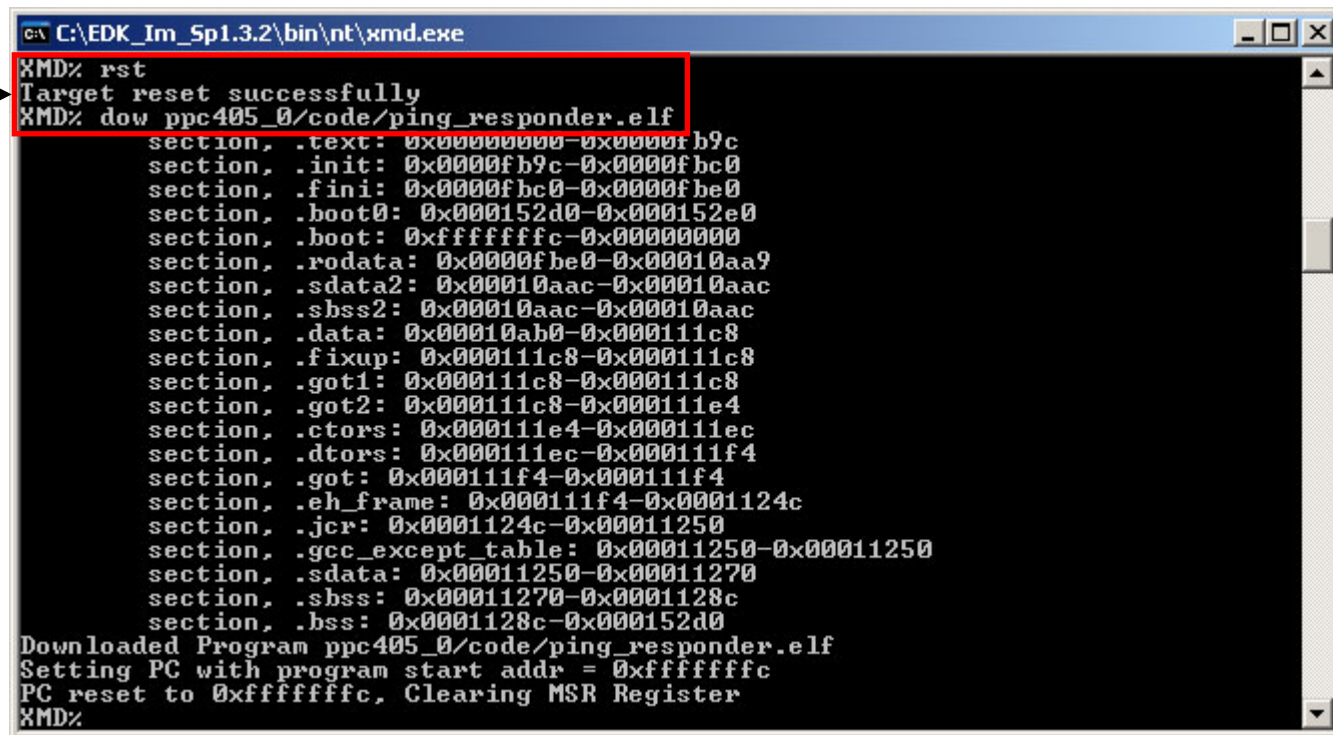
XMD: Connected to PowerPC target. Processor Version No : 0x20011470
Address mapping for accessing special PowerPC features from XMD/GDB:
  I-Cache <Data> : Disabled
  I-Cache <Tag>  : Disabled
  D-Cache <Data> : Disabled
  D-Cache <Tag>  : Disabled
  ISOCM          : Disabled
  TLB            : Disabled
  DCR            : Disabled

Connected to "ppc" target. id = 0
Starting GDB server for "ppc" target (id = 0) at TCP port no 1234
XMD% mrd 0xffffffffc
FFFFFFFFC:  48000000

XMD%
```

Download ELF File

- Download the ping_responder ELF file from XMD
rst
dow ppc405_0/code/ping_responder.elf (1)

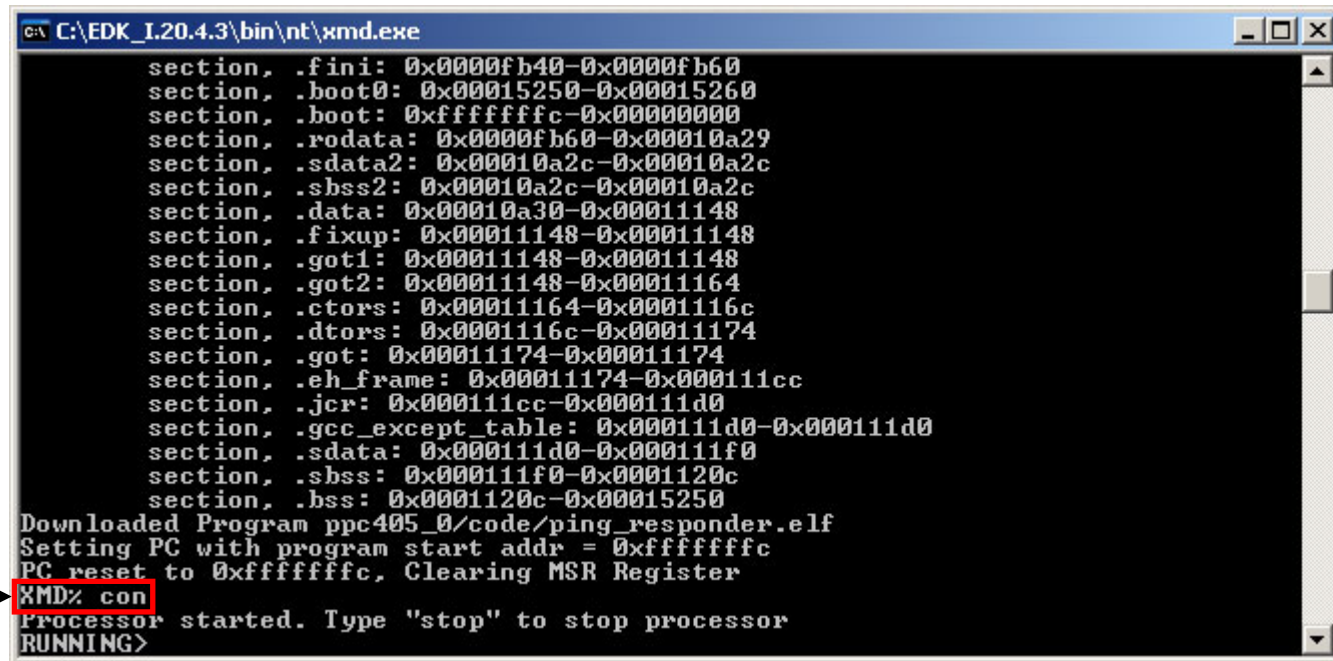


A screenshot of a Windows command prompt window titled "C:\EDK_Im_Sp1.3.2\bin\nt\xmd.exe". The window shows the execution of the XMD command line interface. A yellow circle with the number "1" and an arrow points to the command "XMD% dow ppc405_0/code/ping_responder.elf", which is highlighted with a red rectangular box. The output of the command is a list of memory sections for the program, including .text, .init, .fini, .boot0, .boot, .rodata, .sdata2, .sbss2, .data, .fixup, .got1, .got2, .ctors, .dtors, .got, .eh_frame, .jcr, .gcc_except_table, .sdata, .sbss, and .bss, each with its start and end addresses in hexadecimal. Below the list, the console shows the messages: "Downloaded Program ppc405_0/code/ping_responder.elf", "Setting PC with program start addr = 0xffffffff", and "PC reset to 0xffffffff, Clearing MSR Register". The prompt "XMD%" is visible at the bottom.

```
C:\EDK_Im_Sp1.3.2\bin\nt\xmd.exe
XMD% rst
Target reset successfully
XMD% dow ppc405_0/code/ping_responder.elf
section, .text: 0x00000000-0x00000b9c
section, .init: 0x0000fb9c-0x0000fbc0
section, .fini: 0x0000fbc0-0x0000fbe0
section, .boot0: 0x000152d0-0x000152e0
section, .boot: 0xffffffff-0x00000000
section, .rodata: 0x0000fbe0-0x00010aa9
section, .sdata2: 0x00010aac-0x00010aac
section, .sbss2: 0x00010aac-0x00010aac
section, .data: 0x00010ab0-0x000111c8
section, .fixup: 0x000111c8-0x000111c8
section, .got1: 0x000111c8-0x000111c8
section, .got2: 0x000111c8-0x000111e4
section, .ctors: 0x000111e4-0x000111ec
section, .dtors: 0x000111ec-0x000111f4
section, .got: 0x000111f4-0x000111f4
section, .eh_frame: 0x000111f4-0x0001124c
section, .jcr: 0x0001124c-0x00011250
section, .gcc_except_table: 0x00011250-0x00011250
section, .sdata: 0x00011250-0x00011270
section, .sbss: 0x00011270-0x0001128c
section, .bss: 0x0001128c-0x000152d0
Downloaded Program ppc405_0/code/ping_responder.elf
Setting PC with program start addr = 0xffffffff
PC reset to 0xffffffff, Clearing MSR Register
XMD%
```

Run Ping Responder

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

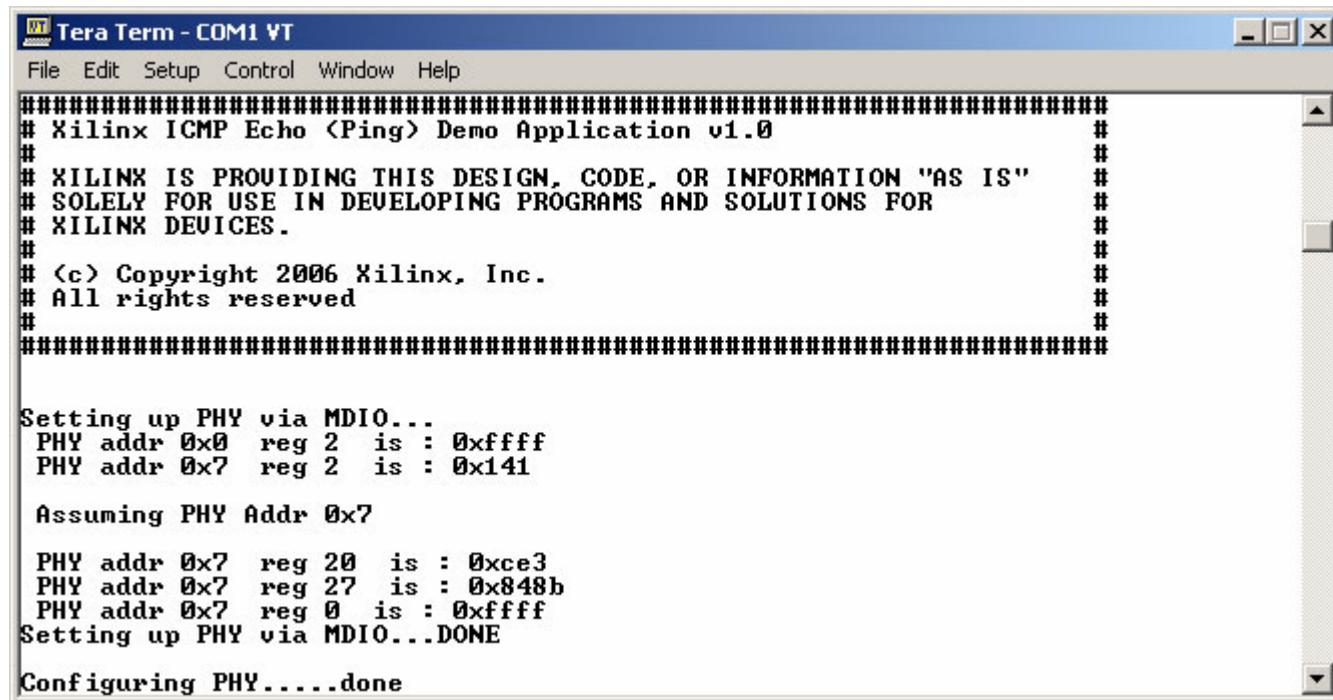


```
C:\EDK_I.20.4.3\bin\nt\xmd.exe
section, .fini: 0x0000fb40-0x0000fb60
section, .boot0: 0x00015250-0x00015260
section, .boot: 0xffffffffc-0x00000000
section, .rodata: 0x0000fb60-0x00010a29
section, .sdata2: 0x00010a2c-0x00010a2c
section, .sbss2: 0x00010a2c-0x00010a2c
section, .data: 0x00010a30-0x00011148
section, .fixup: 0x00011148-0x00011148
section, .got1: 0x00011148-0x00011148
section, .got2: 0x00011148-0x00011164
section, .ctors: 0x00011164-0x0001116c
section, .dtors: 0x0001116c-0x00011174
section, .got: 0x00011174-0x00011174
section, .eh_frame: 0x00011174-0x000111cc
section, .jcr: 0x000111cc-0x000111d0
section, .gcc_except_table: 0x000111d0-0x000111d0
section, .sdata: 0x000111d0-0x000111f0
section, .sbss: 0x000111f0-0x0001120c
section, .bss: 0x0001120c-0x00015250
Downloaded Program ppc405_0/code/ping_responder.elf
Setting PC with program start addr = 0xffffffffc
PC reset to 0xffffffffc, Clearing MSR Register
XMD% con
Processor started. Type "stop" to stop processor
RUNNING>
```

1

Run Ping Responder

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



```
Tera Term - COM1 VT
File Edit Setup Control Window Help
#####
# Xilinx ICMP Echo (Ping) Demo Application v1.0
#
# XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS"
# SOLELY FOR USE IN DEVELOPING PROGRAMS AND SOLUTIONS FOR
# XILINX DEVICES.
#
# (c) Copyright 2006 Xilinx, Inc.
# All rights reserved
#
#####

Setting up PHY via MDIO...
PHY addr 0x0 reg 2 is : 0xffff
PHY addr 0x7 reg 2 is : 0x141

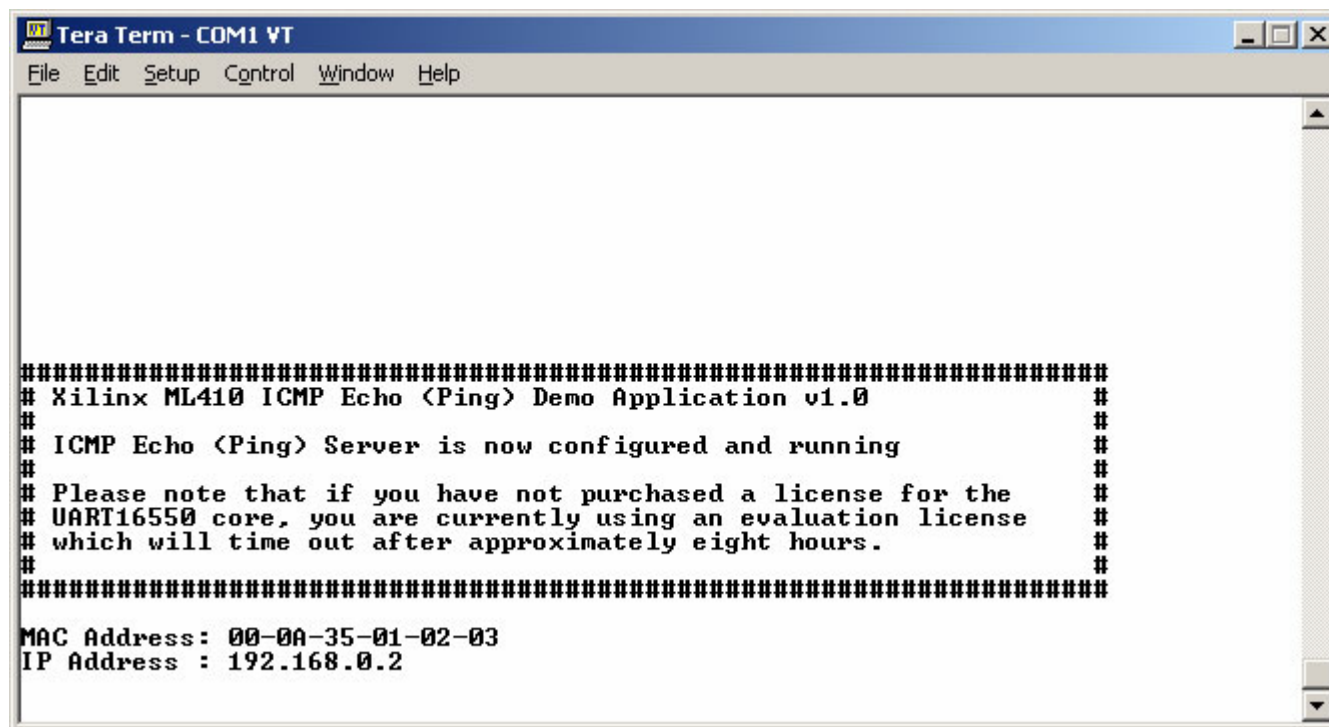
Assuming PHY Addr 0x7

PHY addr 0x7 reg 20 is : 0xce3
PHY addr 0x7 reg 27 is : 0x848b
PHY addr 0x7 reg 0 is : 0xffff
Setting up PHY via MDIO...DONE

Configuring PHY....done
```

Run Ping Responder

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

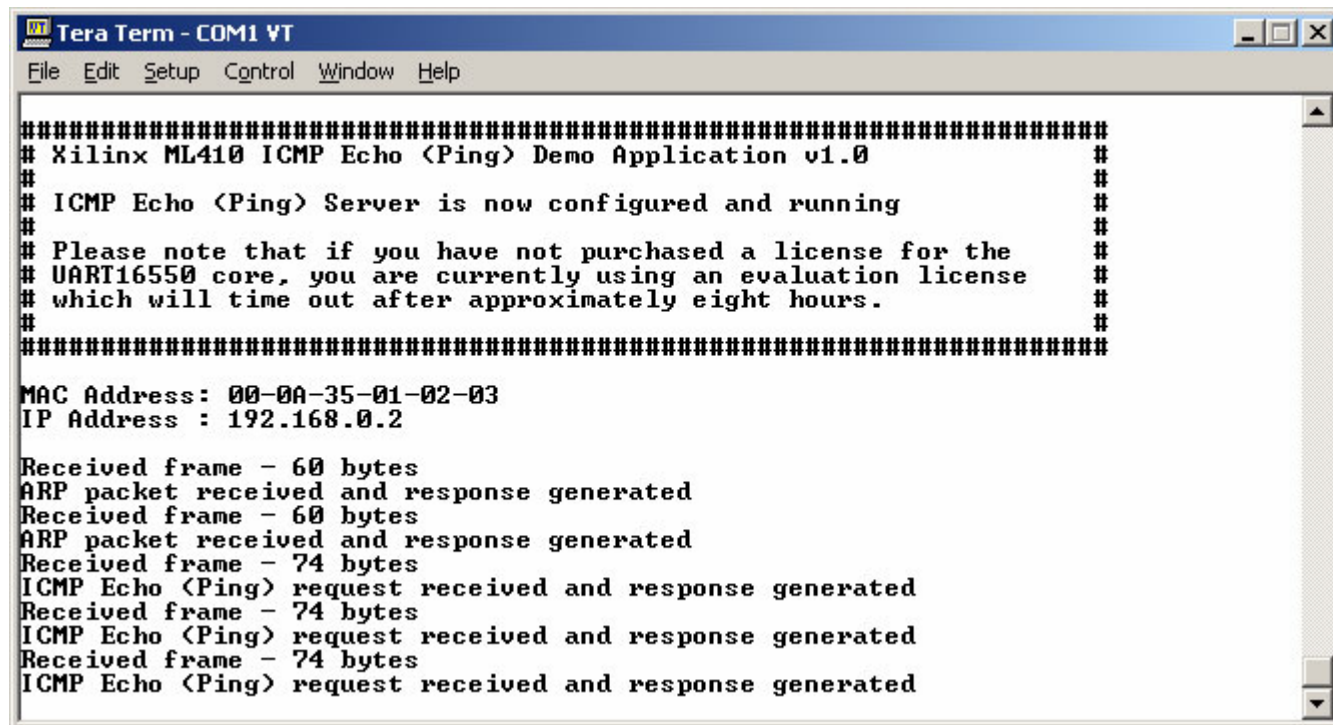


```
Tera Term - COM1 VT
File Edit Setup Control Window Help

#####
# Xilinx ML410 ICMP Echo <Ping> Demo Application v1.0      #
#                                                         #
# ICMP Echo <Ping> Server is now configured and running   #
#                                                         #
# Please note that if you have not purchased a license for the
# UART16550 core, you are currently using an evaluation license
# which will time out after approximately eight hours.     #
#                                                         #
#####
MAC Address: 00-0A-35-01-02-03
IP Address : 192.168.0.2
```

Run Ping Responder

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



```
Tera Term - COM1 VT
File Edit Setup Control Window Help

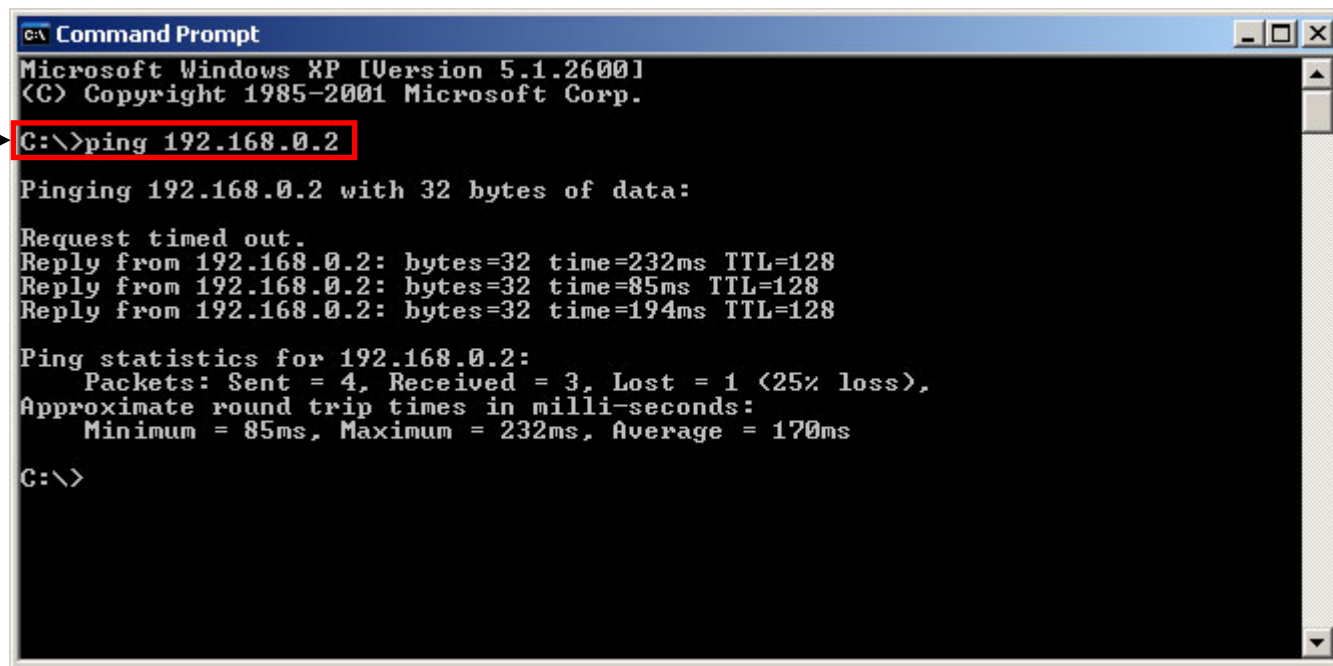
#####
# Xilinx ML410 ICMP Echo (Ping) Demo Application v1.0
#
# ICMP Echo (Ping) Server is now configured and running
#
# Please note that if you have not purchased a license for the
# UART16550 core, you are currently using an evaluation license
# which will time out after approximately eight hours.
#
#####

MAC Address: 00-0A-35-01-02-03
IP Address : 192.168.0.2

Received frame - 60 bytes
ARP packet received and response generated
Received frame - 60 bytes
ARP packet received and response generated
Received frame - 74 bytes
ICMP Echo (Ping) request received and response generated
Received frame - 74 bytes
ICMP Echo (Ping) request received and response generated
Received frame - 74 bytes
ICMP Echo (Ping) request received and response generated
```

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



```
C:\ Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\>ping 192.168.0.2

Pinging 192.168.0.2 with 32 bytes of data:

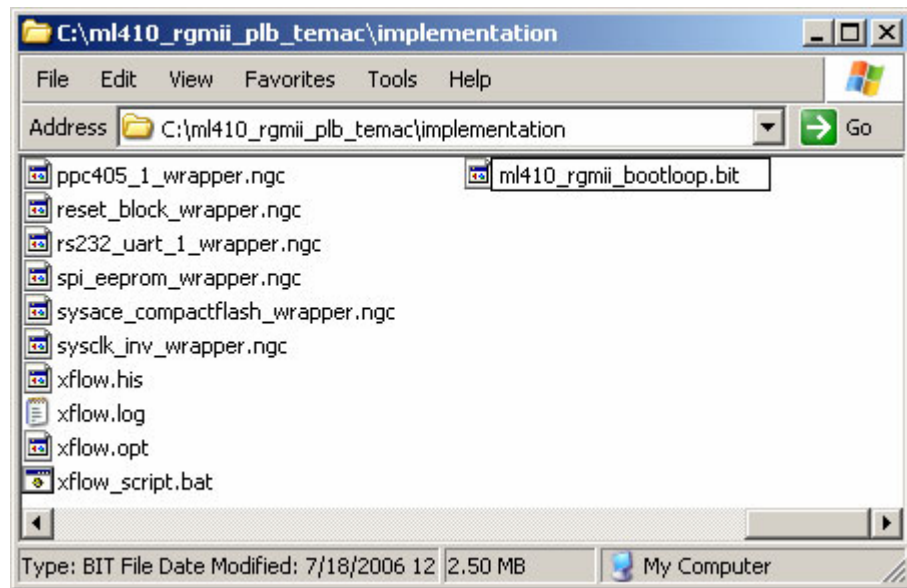
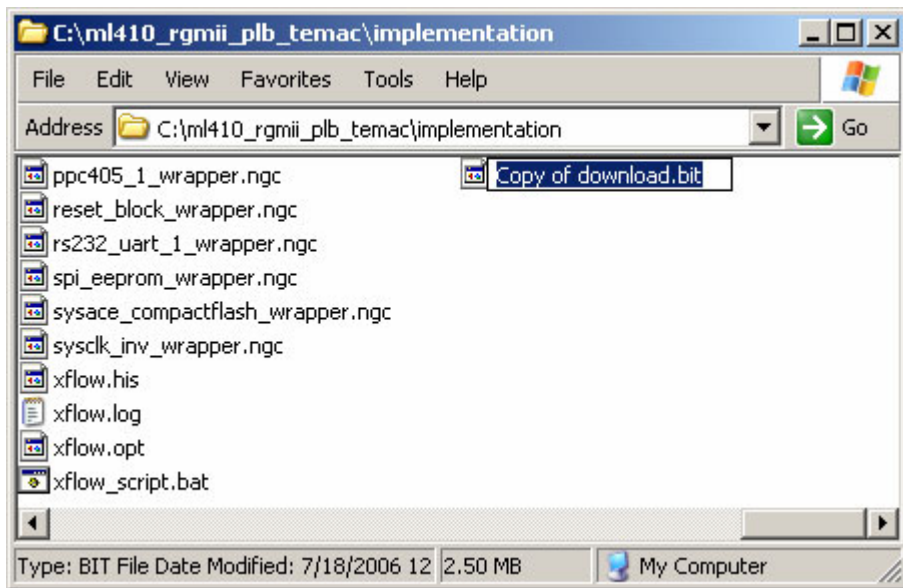
Request timed out.
Reply from 192.168.0.2: bytes=32 time=232ms TTL=128
Reply from 192.168.0.2: bytes=32 time=85ms TTL=128
Reply from 192.168.0.2: bytes=32 time=194ms TTL=128

Ping statistics for 192.168.0.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 85ms, Maximum = 232ms, Average = 170ms

C:\>
```

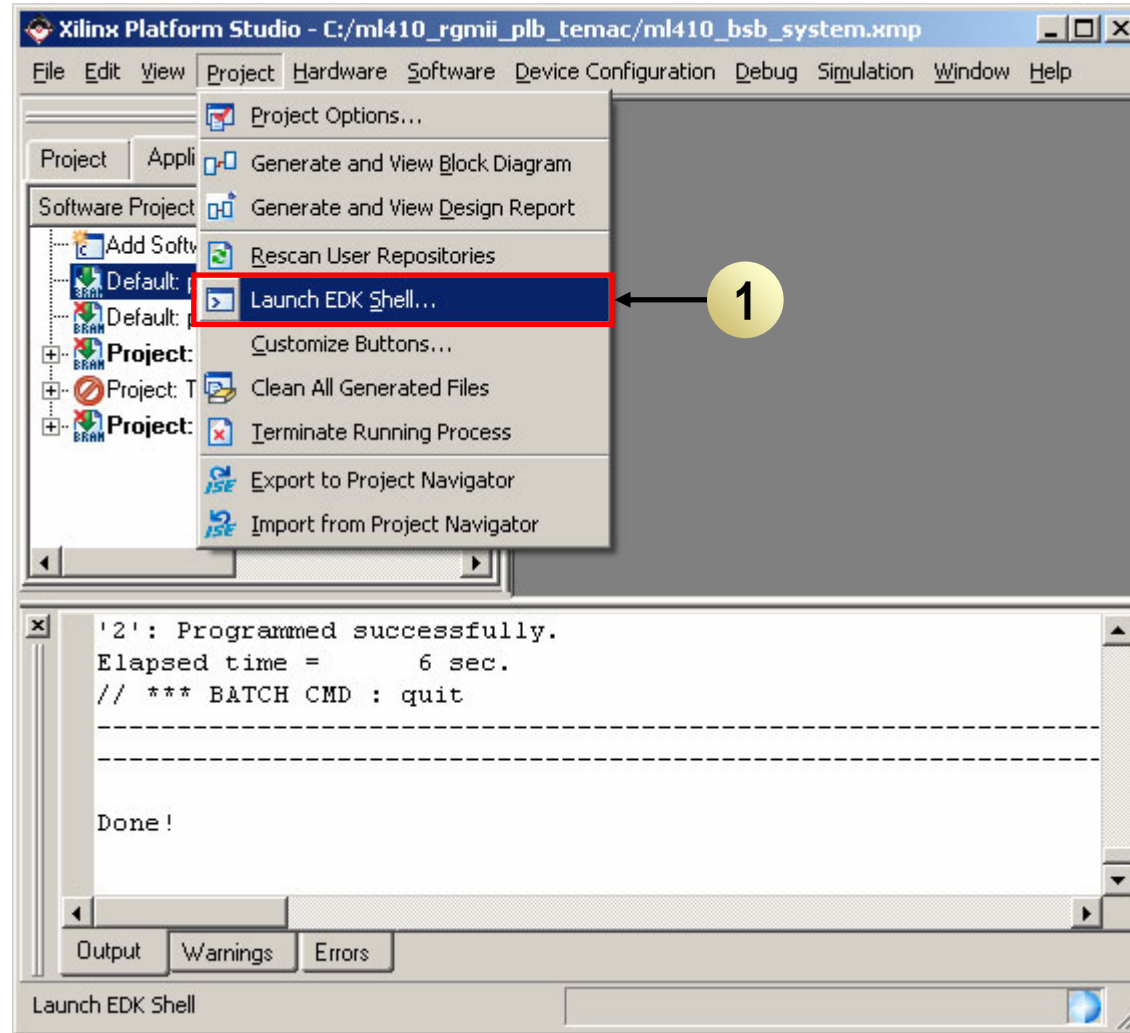
Loading Bootloop into BRAM

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



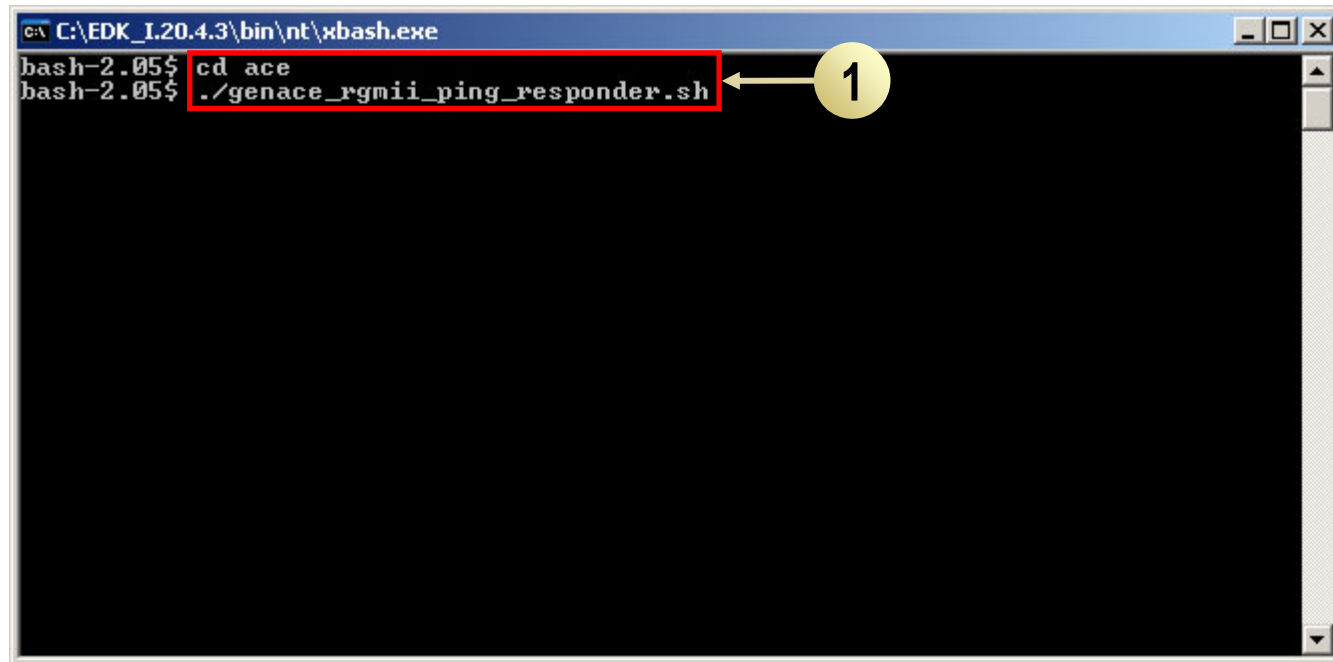
Create an ACE File

- 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



Create an ACE File

- At the bash prompt, type (1):
`cd ace`
`./genace_rgmii_ping_responder.sh`

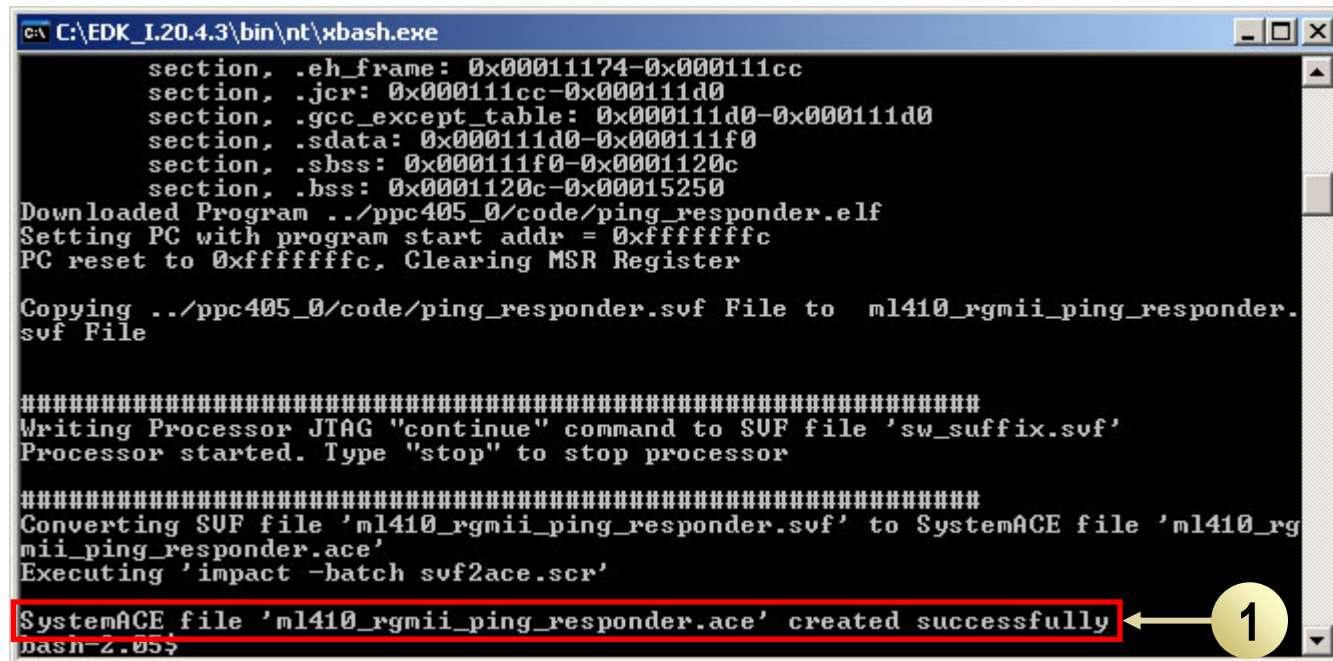


A screenshot of a terminal window titled "C:\EDK_I.20.4.3\bin\nt\xbash.exe". The terminal shows a bash prompt "bash-2.05\$" followed by the command "cd ace". The next line shows the prompt "bash-2.05\$" followed by the command " ./genace_rgmii_ping_responder.sh". A red rectangular box highlights the command " ./genace_rgmii_ping_responder.sh". A yellow circle with the number "1" and an arrow points to the command, indicating it is the step to be performed.

```
C:\EDK_I.20.4.3\bin\nt\xbash.exe
bash-2.05$ cd ace
bash-2.05$ ./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)



```
C:\EDK_I.20.4.3\bin\nt\xbash.exe
section, .eh_frame: 0x00011174-0x000111cc
section, .jcr: 0x000111cc-0x000111d0
section, .gcc_except_table: 0x000111d0-0x000111d0
section, .sdata: 0x000111d0-0x000111f0
section, .sbss: 0x000111f0-0x0001120c
section, .bss: 0x0001120c-0x00015250
Downloaded Program ../ppc405_0/code/ping_responder.elf
Setting PC with program start addr = 0xffffffffc
PC reset to 0xffffffffc, Clearing MSR Register

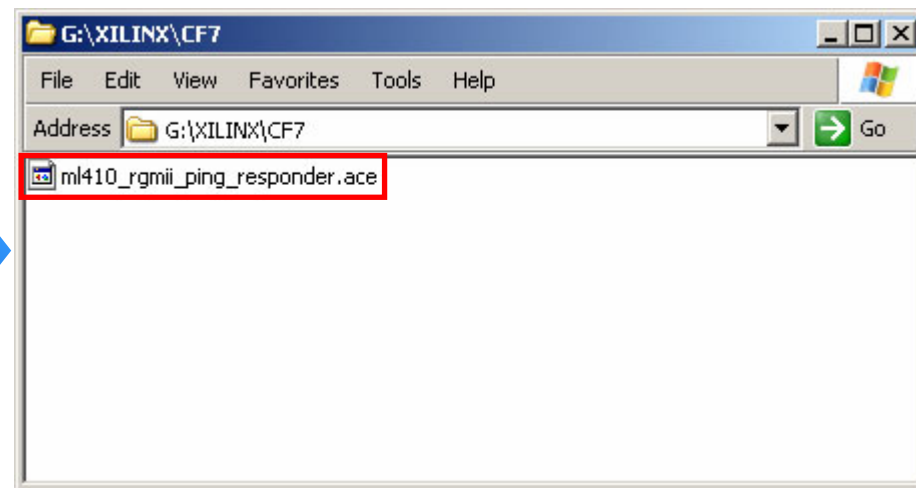
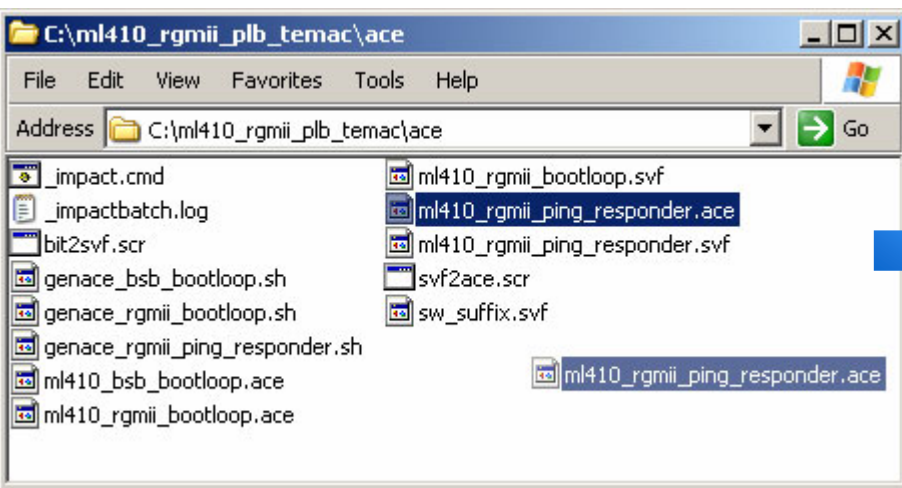
Copying ../ppc405_0/code/ping_responder.svf File to ml410_rgmii_ping_responder.
svf File

#####
Writing Processor JTAG "continue" command to SVF file 'sw_suffix.svf'
Processor started. Type "stop" to stop processor

#####
Converting SVF file 'ml410_rgmii_ping_responder.svf' to SystemACE file 'ml410_rg
mii_ping_responder.ace'
Executing 'impact -batch svf2ace.scr'
SystemACE file 'ml410_rgmii_ping_responder.ace' created successfully
bash-2.05$
```

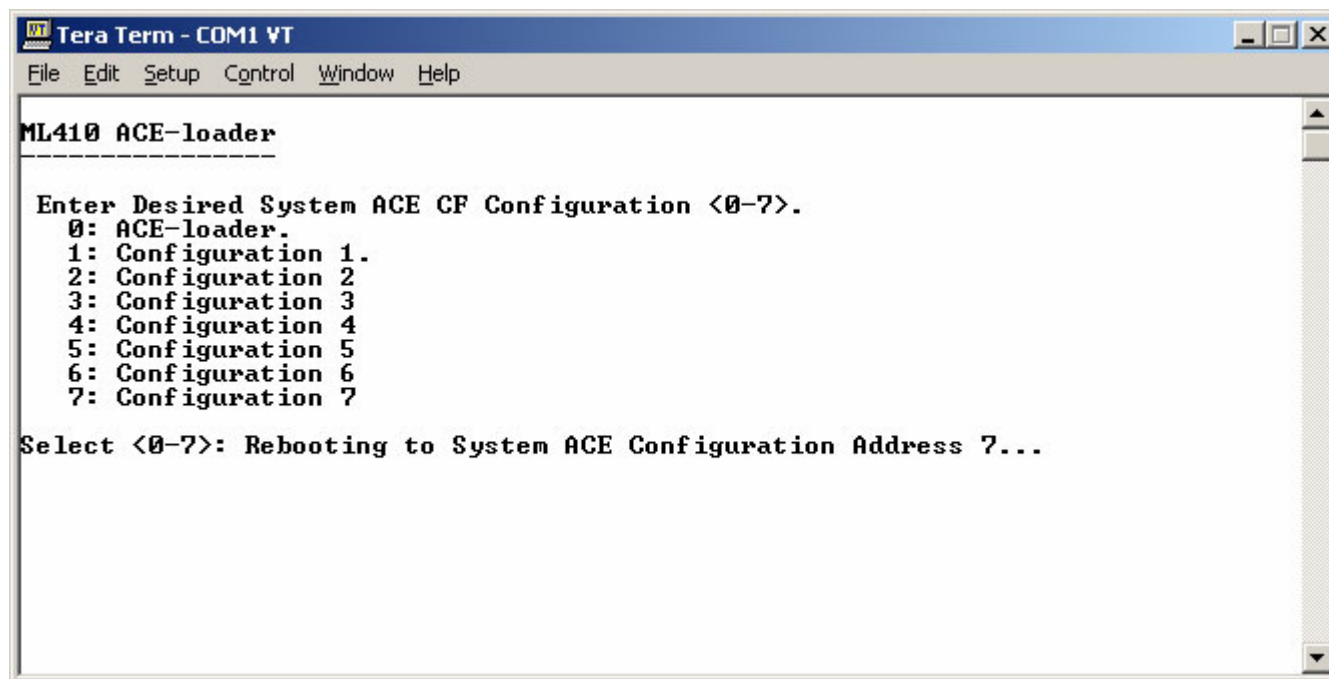
Run ACE File

- 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



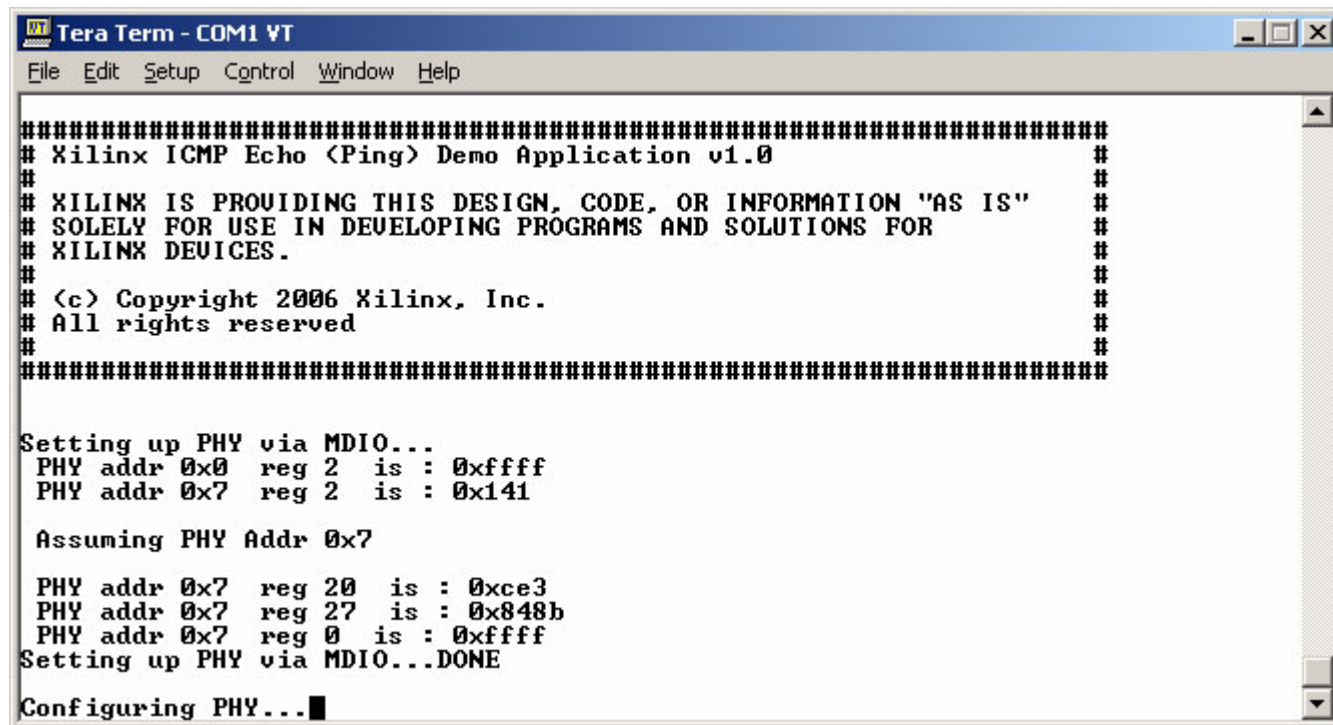
The screenshot shows a terminal window titled "Tera Term - COM1 VT". The menu displayed is:

```
ML410 ACE-loader
-----
Enter Desired System ACE CF Configuration <0-7>.
0: ACE-loader.
1: Configuration 1.
2: Configuration 2.
3: Configuration 3.
4: Configuration 4.
5: Configuration 5.
6: Configuration 6.
7: Configuration 7.

Select <0-7>: Rebooting to System ACE Configuration Address 7...
```

Using the ACE File

- Ping Responder output after booting ACE file



```
Tera Term - COM1 VT
File Edit Setup Control Window Help

#####
# Xilinx ICMP Echo (Ping) Demo Application v1.0 #
#
# XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS" #
# SOLELY FOR USE IN DEVELOPING PROGRAMS AND SOLUTIONS FOR #
# XILINX DEVICES. #
#
# (c) Copyright 2006 Xilinx, Inc. #
# All rights reserved #
#
#####

Setting up PHY via MDIO...
PHY addr 0x0 reg 2 is : 0xffff
PHY addr 0x7 reg 2 is : 0x141

Assuming PHY Addr 0x7

PHY addr 0x7 reg 20 is : 0xce3
PHY addr 0x7 reg 27 is : 0x848b
PHY addr 0x7 reg 0 is : 0xffff
Setting up PHY via MDIO...DONE

Configuring PHY...█
```

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

