LwIP Applications For the Arty Evaluation Board

Vivado version 2015.2
Document version 1.0
August, 2015

Table of Contents

Overview

This document describes a basic MicroBlaze^{TM} design implemented and tested on the Avnet/Digilent Arty Evaluation Board. This example design utilizes the light-weight IP (lwIP) protocol stack in raw API mode, with the Xilinx 10/100 soft AXI_Ethernetlite MAC in simple FIFO Interrupt mode. This example design is based on Xilinx Applications Note XAPP1026 (www.xilinx.com/support/documentation/application_notes/xapp1026.pdf) with small modifications to the source software to match the board hardware and to boost performance.

Lightweight IP (lwIP) is an open source TCP/IP networking stack for embedded systems. The Xilinx® Software Development Kit (SDK) provides lwIP software customized to run on Xilinx embedded systems containing a MicroBlaze™ processor. This tutorial describes how to utilize the lwIP library to add networking capability to an embedded system. In particular, lwIP is utilized to develop these applications:

- TCP/IP echo server
- HTTP web server
- TFTP server
- Receive and transmit throughput tests.

This tutorial shows how to build a MicroBlaze Hardware Platform and then create, build, and run a lwIP-enabled software project with networking capability on the Avnet/Digilent Arty Evaluation Board.

Objectives

When you have completed this tutorial, you will know how to do the following:

- Build a MicroBlaze hardware platform capable of running Ethernet networking applications.
- Set up an SDK workspace.
- Create a BSP that includes the lwIP TCP/IP stack and support for the webserver file system.
- Add example lwIP software applications.
- Test several networking applications.

Description of Included lwIP Applications

The applications listed below are provided as software examples for lwIP raw API implementation to demonstrate the performance and utility of the Ethernet link. The RxTest and TxTest applications communicate with a publicly available Internet Performance monitor application (IPerf). A windows executable for this application is included with this reference design.

Echo server – implements a simple application to echo whatever is sent to the program over the network.

RxTest – implements an IPerf server application. IPerf is run in client mode on the PC and sends a stream of packet data to RxTest, running on the development board. RxTest receives the data at the application level, so the data propagates all the way through the lwIP stack. RxTest simply discards the data once it is received. The lwIP stack sends an acknowledgement packet back to IPerf, and IPerf calculates and displays the throughput based on the average round trip times of each packet and the data sent.

TxTest – implements an IPerf client application. IPerf is run in server mode on the PC and receives a stream of packet data sent by TxTest running on the development board. IPerf calculates and displays the throughput based on the amount of data received from the client.

TFTP server – implements a Trivial File Transfer Protocol (TFTP) server to send and receive files over the network.

HTTP server – implements a simple webserver to demonstrate how a remote embedded system can be monitored and controlled over an Ethernet network. The Xilinx Memory File System (xilmfs) is used to store a collection of files in the memory of the Arty Evaluation Board and be served by the webserver application. The LEDs and DIP switches on the Arty board are toggled and monitored using HTTP POST commands and files in the filesystem, such as an image of the Arty Evaluation board showing the locations of the LEDs and switches, are accessed using HTTP GET commands.

Reference Design Requirements

The following items are required for proper completion of this tutorial.

Software

The software requirements for this reference design are:

- Linux, Windows 7, Windows 8.1
  (ug973-vivado-release-notes-install-license.pdf)
- The IPerf software application supplied with this design is a Windows binary, and thus the Windows requirement. IPerf is also available as a Linux utility, and the method for acquiring it in Linux (yum, apt-get, etc.) will depend on your distribution.
- Xilinx Vivado Design Edition 2015.2, with SDK

Hardware

The hardware setup used by this reference design includes:

- Computer with 1 GB RAM and 1 GB virtual memory (recommended)
  (www.xilinx.com/design-tools/vivado/memory.htm)
- Avnet/Digilent Arty Evaluation Board
- USB-A to USB-micro B cable
- Cat-5 Ethernet Cable
- Host PC with 10/100 compatible Ethernet NIC
- Host PC or network router to act as DHCP server (optional)
- Power supply (optional)

Supplied Files

The following directory structure is included with this reference design:

demo: Contains the files to run receive test program:

- **config_fpga.bat**: Batch file to configure the FPGA with the "bootloop" bitstream of the MicroBlaze processor system.
- **cp_from_sdk.bat**: Batch file to copy hardware bitstream and software executables from the SDK workspace
- **demo_raw_apps.bat**: Batch file to run the commands to load the FPGA bitstream of the hardware design, download the webserver file system, and download the software application.
- **design_1_wrapper.bit**: The golden FPGA bitstream of the hardware design.
- **design_1_wrapper.mmi**: The golden BRAM map file to integrate the bootloop executable with the FPGA bitstream.
- **download.bit**: The golden FPGA bitstream of the hardware design required to run the lwIP raw mode networking applications.
- **download_bit.tcl**: TCL command file for downloading the bitstream to the board.
- **image.mfs:** Formatted webserver filesystem.
- **load_bits.tcl**: TCL command file for downloading the bitstream to the board.
- **lwip_raw_apps.elf**: The golden MicroBlaze executable for the lwIP raw mode applications.
- **mb_bootloop_le.elf:** Application binary for MicroBlaze "bootloop" software application to hold the CPU in a known idle state.
- **mbrst.opt:** XMD options file to reset the MicroBlaze and re-download the software executable without downloading the webserver file system.
- **run_iperf_cli.bat**: Batch script to run IPerf in client mode for RxTest.
- **run_iperf_ser.bat**: Batch script to run IPerf in server mode for TxTest.
- **sleep.exe:** Utility to pause execution of the batch file.
- **xmd.ini**: Command file used by XMD to download the webserver file system and start execution of the lwIP raw mode software applications.

doc: Contains the documentation for this reference design.

- **7A35T_Arty_lwIP_EthernetLite_VIV2015_2.pdf**: This document.

IPerf: Contains the Internet Performance program to run on a windows PC

IPI: Empty folder to use to create new Vivado IPI hardware project for this design.

IPI_repo: Repository of files and IP needed to create the MicroBlaze hardware platform.

memfs: Collection of files for the webserver filesystem.

sdk_repo: Contains the software applications and lwIP library source files.

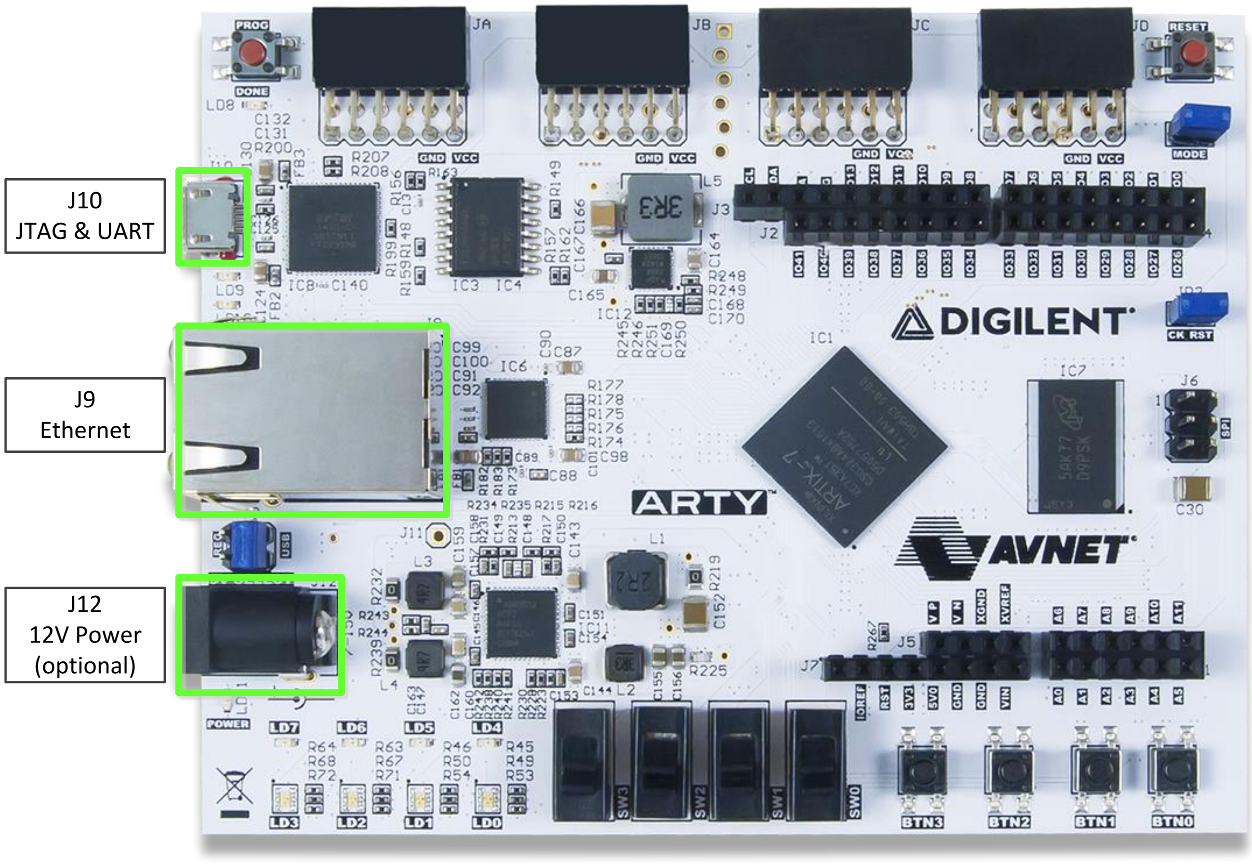
SDK_Workspace: Empty folder to use to create new SDK workspace for this design.

IPI_solution.zip: Contains pre-built XPS project for this design.

sdk_solution.zip: Contains pre-built SDK project for this design.

Setting Up the Arty Evaluation Board

Refer to the following figure and perform the following steps to set up the board for running the applications.



1. Plug a USB cable into the PC and the combination JTAG & UART port (J10) (this will also power the board). If the USB port on the hub or host PC cannot supply enough power you may alternatively power the board via the 12V power barrel jack (J12).

2. Plug an Ethernet cable into the PC (or router) and port J9.

PC Setup

Follow the steps below to configure the Ethernet NIC of the development host PC to establish an Ethernet connection directly with the Arty board.

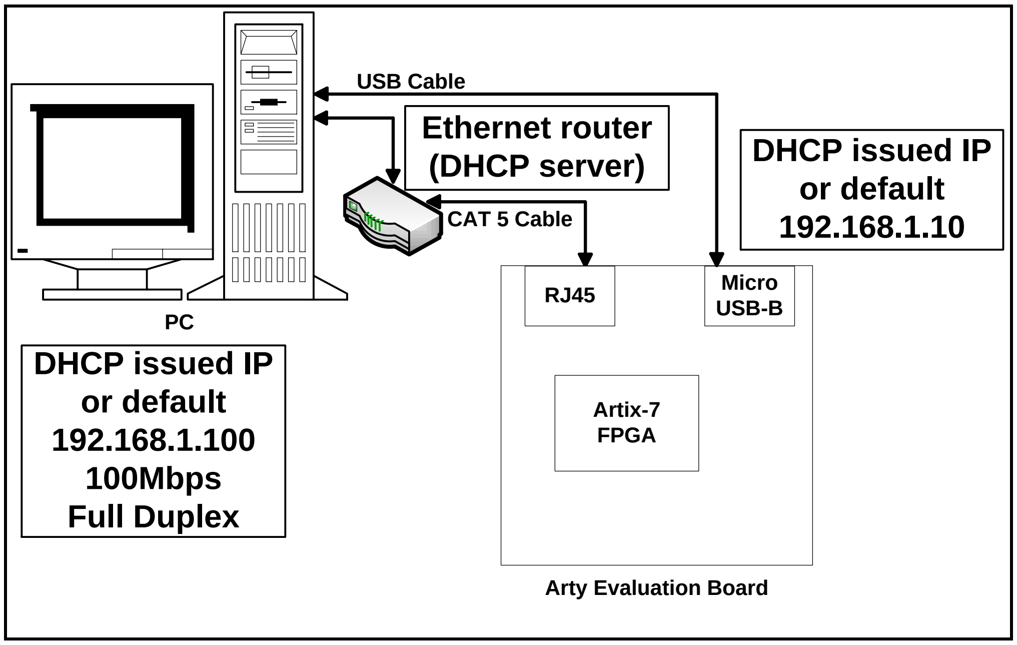

USB Cable
Ethernet router (DHCP server)
DHCP issued IP or default 192.168.1.10
CAT 5 Cable
RJ45
Micro USB-B
PC
DHCP issued IP or default 192.168.1.100 100Mbps Full Duplex
Artix-7 FPGA
Arty Evaluation Board

Figure 1 – Network Connection Setup

1. If you choose to connect the Arty board directly to the host PC and are not running a DHCP server you will need to configure the host PC with an IP address of **192.168.1.100** and a subnet mask of **255.255.255.0**.

2. If you choose to connect the Arty board directly to the host PC and your PC has a 1000 Mbps capable NIC, you will have to manually set the line rate to **100 Mbps full-duplex**. The axi_ethernetlite MAC does not support 1000 Mbps line speeds.

Installing the UART Driver and Virtual COM Port

The USB UART driver is built into the device driver for the JTAG interface and is included with the Xilinx Vivado tools installation. As long as the Vivado tools are installed, the USB UART will be recognized when the board is plugged into the host PC. See Appendix I: Determining the Virtual COM Port for information on identifying the COM port in use on the host PC.

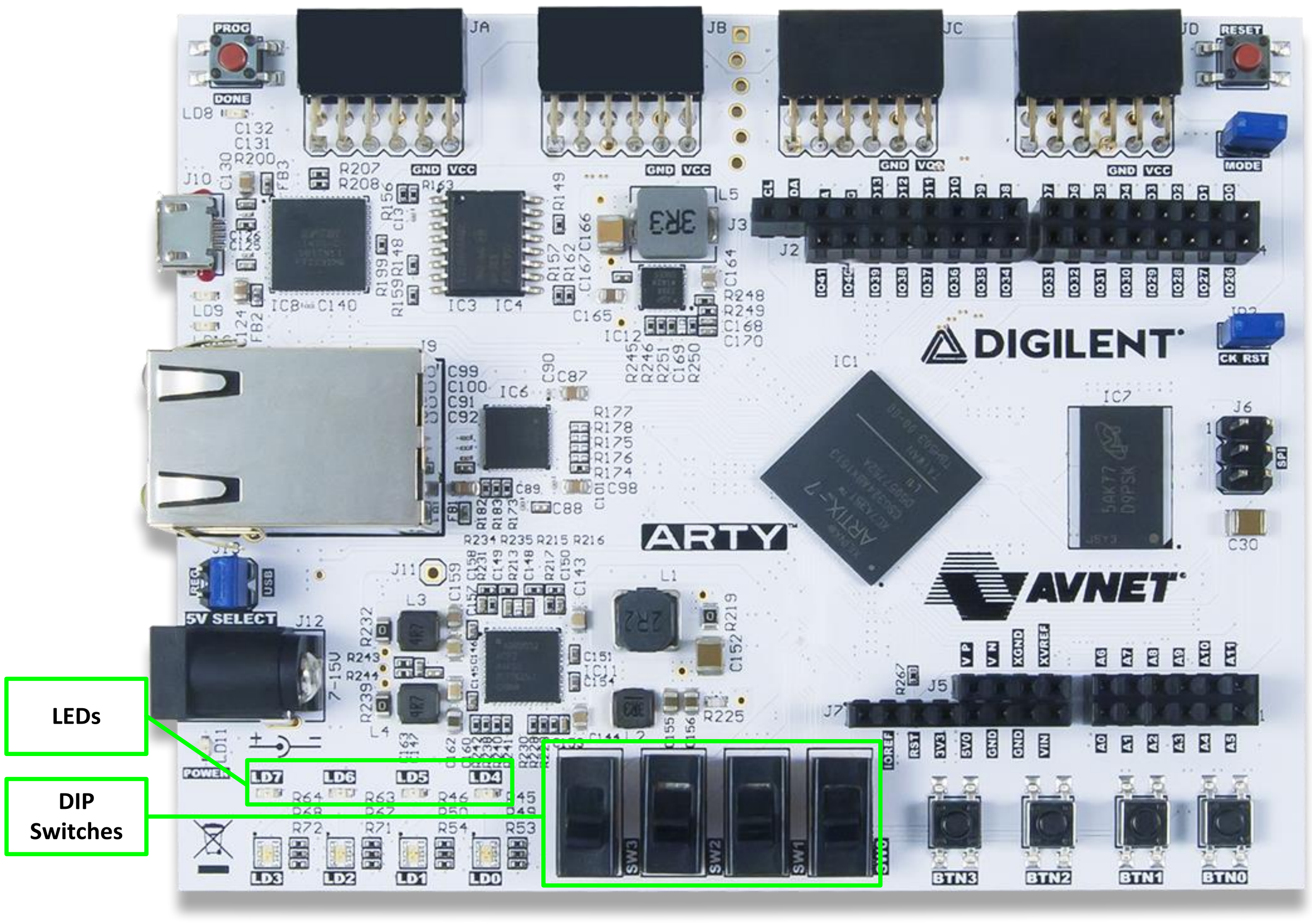
Installing a Serial Console on a Windows 7 Host

Starting with Windows 7, Microsoft no longer includes the HyperTerminal terminal emulator software. However, this example design requires use of terminal emulation software for a serial console connection to the Arty board. A suitable free and open-source replacement for HyperTerminal is TeraTerm. Download and install instructions for TeraTerm can be found at http://en.sourceforge.jp/projects/ttssh2. As an alternative the Terminal applet in the Xilinx SDK may also be used.

Running the Demo Files

You can load the FPGA and run the software application without building the design by using the demo scripts and the pre-built bitstream and elf files. You must have the Xilinx tools (including the SDK) installed on your host, and have the hardware set up and connected as described in Setting Up the Arty Evaluation Board.

Refer to the figure below when running the LwIP webserver. Note the location of the DIP switches and LEDs.



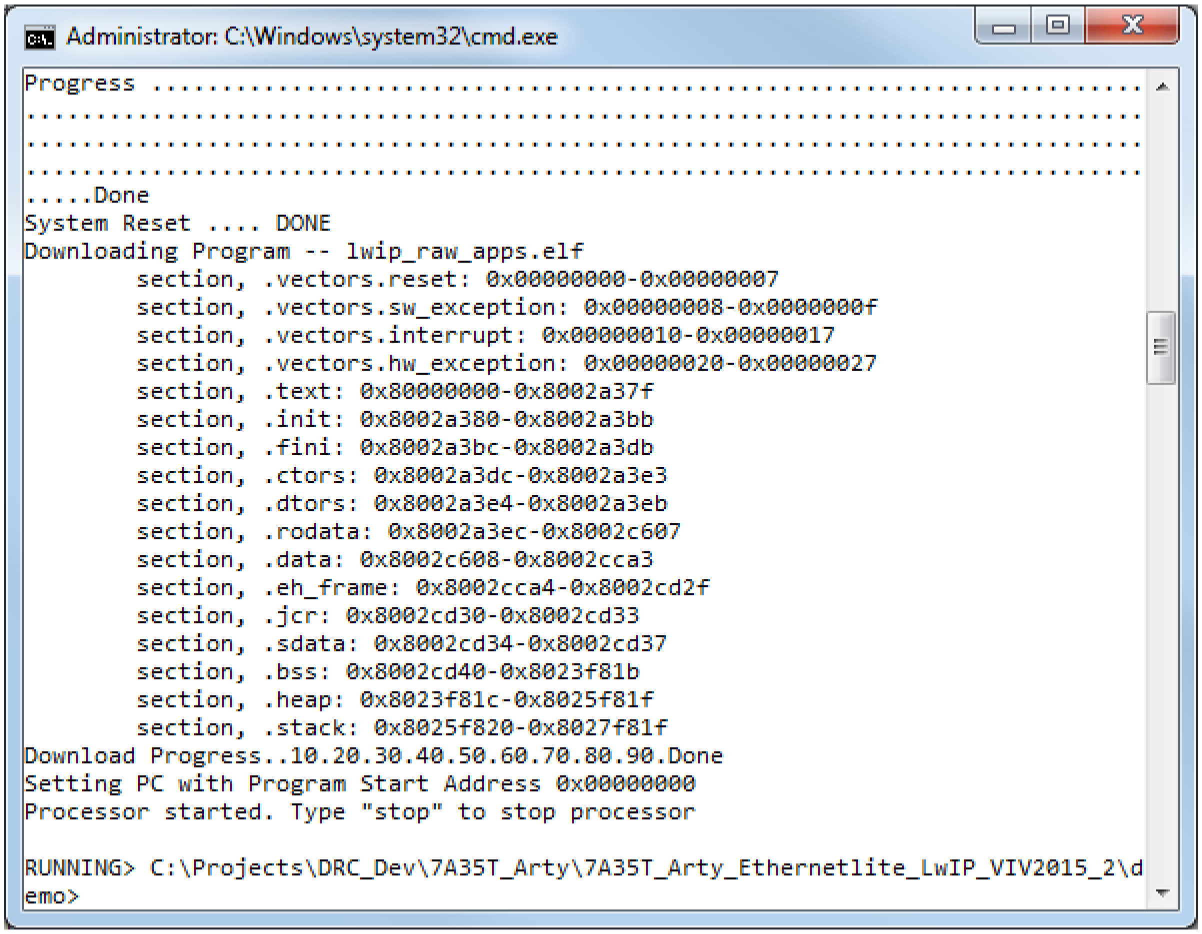
Applications Download

1. Start a serial terminal session and set the serial port parameters to **115200** baud rate, **no** parity, **8** bits, **1** stop bit and no flow control.

2. Open a command window in the **<installation>\demo** folder and enter:

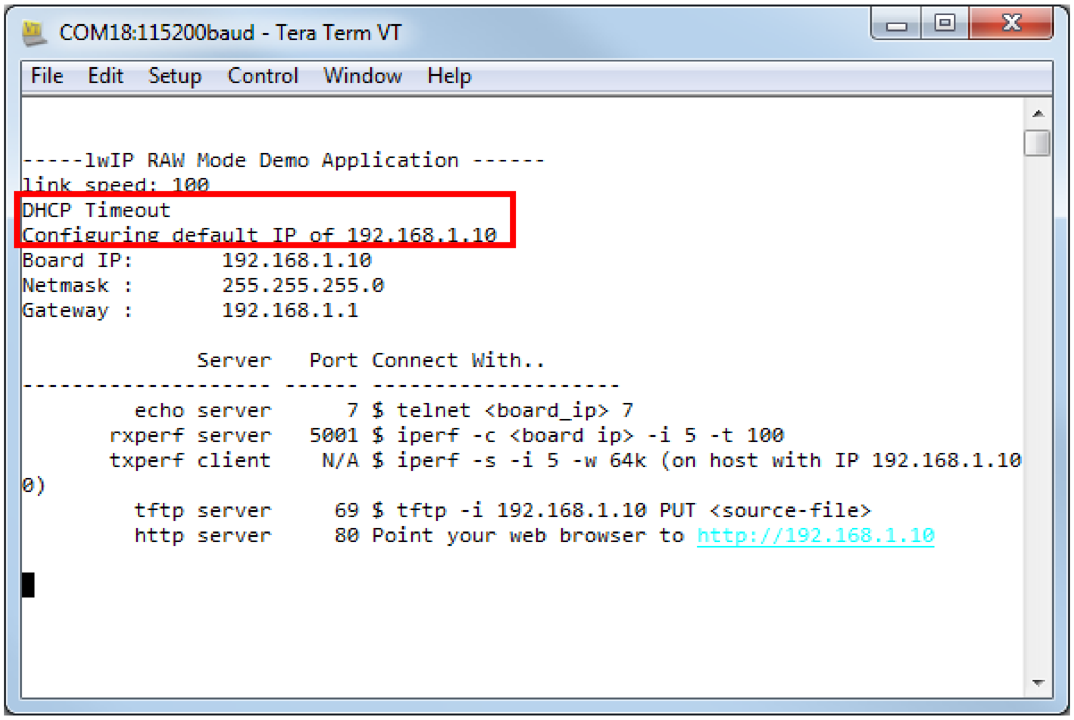
```
demo_raw_apps.bat
```

3. The FPGA bitstream will be downloaded, followed by the webserver file system and the executable file for the software applications. Do not close the command window.



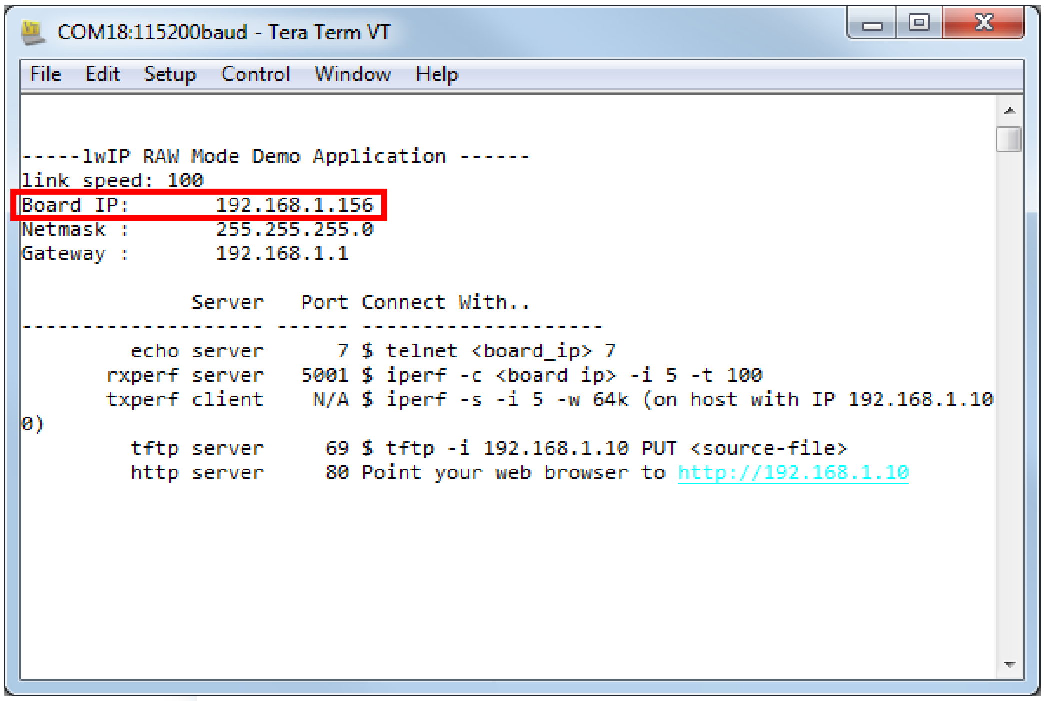
```
Progress ................................................................................
................................................................................
................................................................................
................................................................................
.....Done
System Reset .... DONE
Downloading Program -- lwip_raw_apps.elf
        section, .vectors.reset: 0x00000000-0x00000007
        section, .vectors.sw_exception: 0x00000008-0x0000000f
        section, .vectors.interrupt: 0x00000010-0x00000017
        section, .vectors.hw_exception: 0x00000020-0x00000027
        section, .text: 0x80000000-0x8002a37f
        section, .init: 0x8002a380-0x8002a3bb
        section, .fini: 0x8002a3bc-0x8002a3db
        section, .ctors: 0x8002a3dc-0x8002a3e3
        section, .dtors: 0x8002a3e4-0x8002a3eb
        section, .rodata: 0x8002a3ec-0x8002c607
        section, .data: 0x8002c608-0x8002cca3
        section, .eh_frame: 0x8002cca4-0x8002cd2f
        section, .jcr: 0x8002cd30-0x8002cd33
        section, .sdata: 0x8002cd34-0x8002cd37
        section, .bss: 0x8002cd40-0x8023f81b
        section, .heap: 0x8023f81c-0x8025f81f
        section, .stack: 0x8025f820-0x8027f81f
Download Progress..10.20.30.40.50.60.70.80.90.Done
Setting PC with Program Start Address 0x00000000
Processor started. Type "stop" to stop processor

RUNNING> C:\Projects\DRC_Dev\7A35T_Arty\7A35T_Arty_Ethernetlite_LwIP_VIV2015_2\d
emo>
```

4. When the executable has finished loading and is ready to run you should see the following in your serial terminal window:
   a. At startup lwIP is configured to fetch an IP address from a DHCP server. If there is no DHCP server present the DHCP request will timeout and the board will default to an IP address of 192.168.1.10:


COM18:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
-----lwIP RAW Mode Demo Application ------
link speed: 100
DHCP Timeout
Configuring default IP of 192.168.1.10
Board IP:       192.168.1.10
Netmask :       255.255.255.0
Gateway :       192.168.1.1

              Server   Port Connect With..
-------------------- ------ --------------------
         echo server      7 $ telnet <board_ip> 7
       rxperf server   5001 $ iperf -c <board ip> -i 5 -t 100
       txperf client    N/A $ iperf -s -i 5 -w 64k (on host with IP 192.168.1.10
0)
         tftp server     69 $ tftp -i 192.168.1.10 PUT <source-file>
         http server     80 Point your web browser to http://192.168.1.10
```

   b. If a DHCP server is found, a dynamic IP address will be assigned:


COM18:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
-----lwIP RAW Mode Demo Application ------
link speed: 100
Board IP:       192.168.1.156
Netmask :       255.255.255.0
Gateway :       192.168.1.1

              Server   Port Connect With..
-------------------- ------ --------------------
         echo server      7 $ telnet <board_ip> 7
       rxperf server   5001 $ iperf -c <board ip> -i 5 -t 100
       txperf client    N/A $ iperf -s -i 5 -w 64k (on host with IP 192.168.1.10
0)
         tftp server     69 $ tftp -i 192.168.1.10 PUT <source-file>
         http server     80 Point your web browser to http://192.168.1.10
```

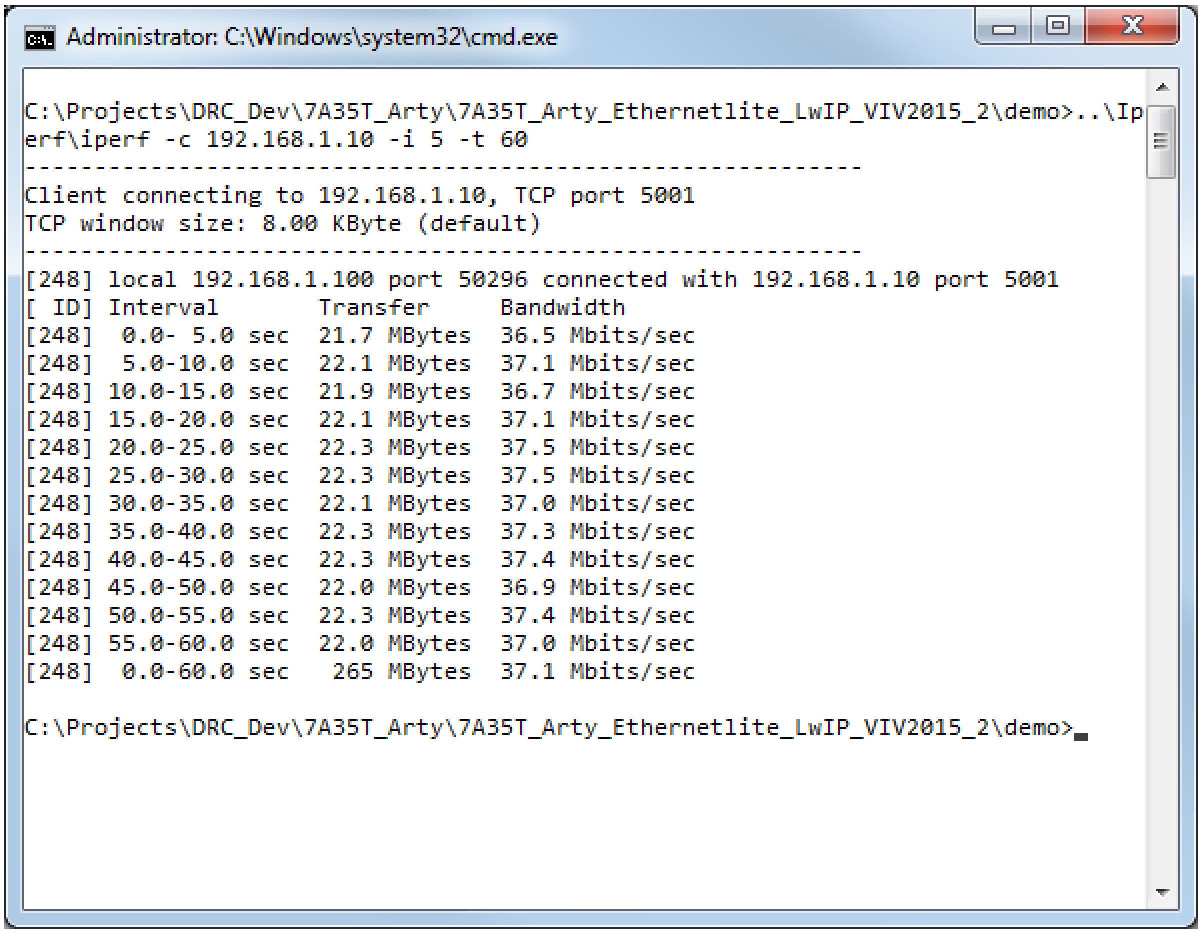
RxTest Demo

The IPerf application on the host PC will connect with the RxTest application on the board, and data transmission will commence. Statistics are printed out every 5 seconds in the command window by IPerf for the duration of the 60 second test. The actual bandwidth values you see may vary slightly from those shown below, depending on your host system[1].

1. In the open command window enter:

```
run_iperf_cli.bat
```

2. You can run the client on the host as many times as you like by running the **<installation>\demo\run_iperf_cli.bat** batch file from the command prompt. Do not close the command window.



```
C:\Projects\DRC_Dev\7A35T_Arty\7A35T_Arty_Ethernetlite_LwIP_VIV2015_2\demo>..\Ip
erf\iperf -c 192.168.1.10 -i 5 -t 60
------------------------------------------------------------
Client connecting to 192.168.1.10, TCP port 5001
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[248] local 192.168.1.100 port 50296 connected with 192.168.1.10 port 5001
[ ID] Interval       Transfer     Bandwidth
[248]  0.0- 5.0 sec  21.7 MBytes  36.5 Mbits/sec
[248]  5.0-10.0 sec  22.1 MBytes  37.1 Mbits/sec
[248] 10.0-15.0 sec  21.9 MBytes  36.7 Mbits/sec
[248] 15.0-20.0 sec  22.1 MBytes  37.1 Mbits/sec
[248] 20.0-25.0 sec  22.3 MBytes  37.5 Mbits/sec
[248] 25.0-30.0 sec  22.3 MBytes  37.5 Mbits/sec
[248] 30.0-35.0 sec  22.1 MBytes  37.0 Mbits/sec
[248] 35.0-40.0 sec  22.3 MBytes  37.3 Mbits/sec
[248] 40.0-45.0 sec  22.3 MBytes  37.4 Mbits/sec
[248] 45.0-50.0 sec  22.0 MBytes  36.9 Mbits/sec
[248] 50.0-55.0 sec  22.3 MBytes  37.4 Mbits/sec
[248] 55.0-60.0 sec  22.0 MBytes  37.0 Mbits/sec
[248]  0.0-60.0 sec   265 MBytes  37.1 Mbits/sec

C:\Projects\DRC_Dev\7A35T_Arty\7A35T_Arty_Ethernetlite_LwIP_VIV2015_2\demo>
```

[1] Host machine used to generate the results was an HP desktop with 3.4 GHz Core i7 CPU, 16 GB RAM and an Intel 82579LM network adaptor, running Windows 7. The host was directly connected with the board with a Cat-5E Ethernet cable.

TxTest Demo

The IPerf application on the host PC will connect with the TxTest application on the board, and data transmission will commence. Statistics are printed out every two seconds in the command window by IPerf until stopped by the user. The actual bandwidth values you see may vary slightly from those shown below, depending on your host system.[2]

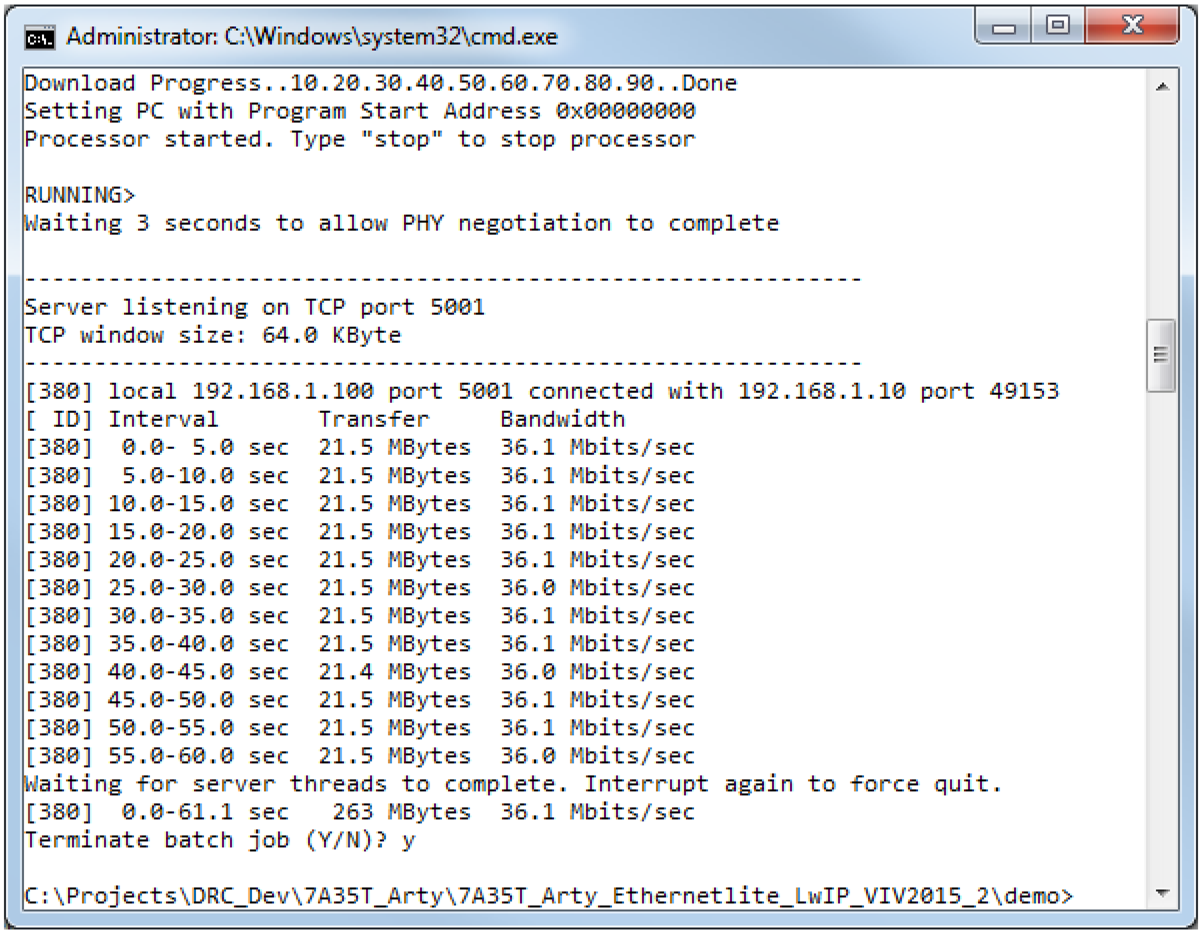
1. In the open command window enter:

```
run_iperf_ser.bat
```

2. The board will transmit packets to the host until you type stop in the open command line window for txtest. Do not close the command window.

3. To stop the server on the host side, enter:

```
<CTRL> - C
<CTRL> - C
  y ↵
```



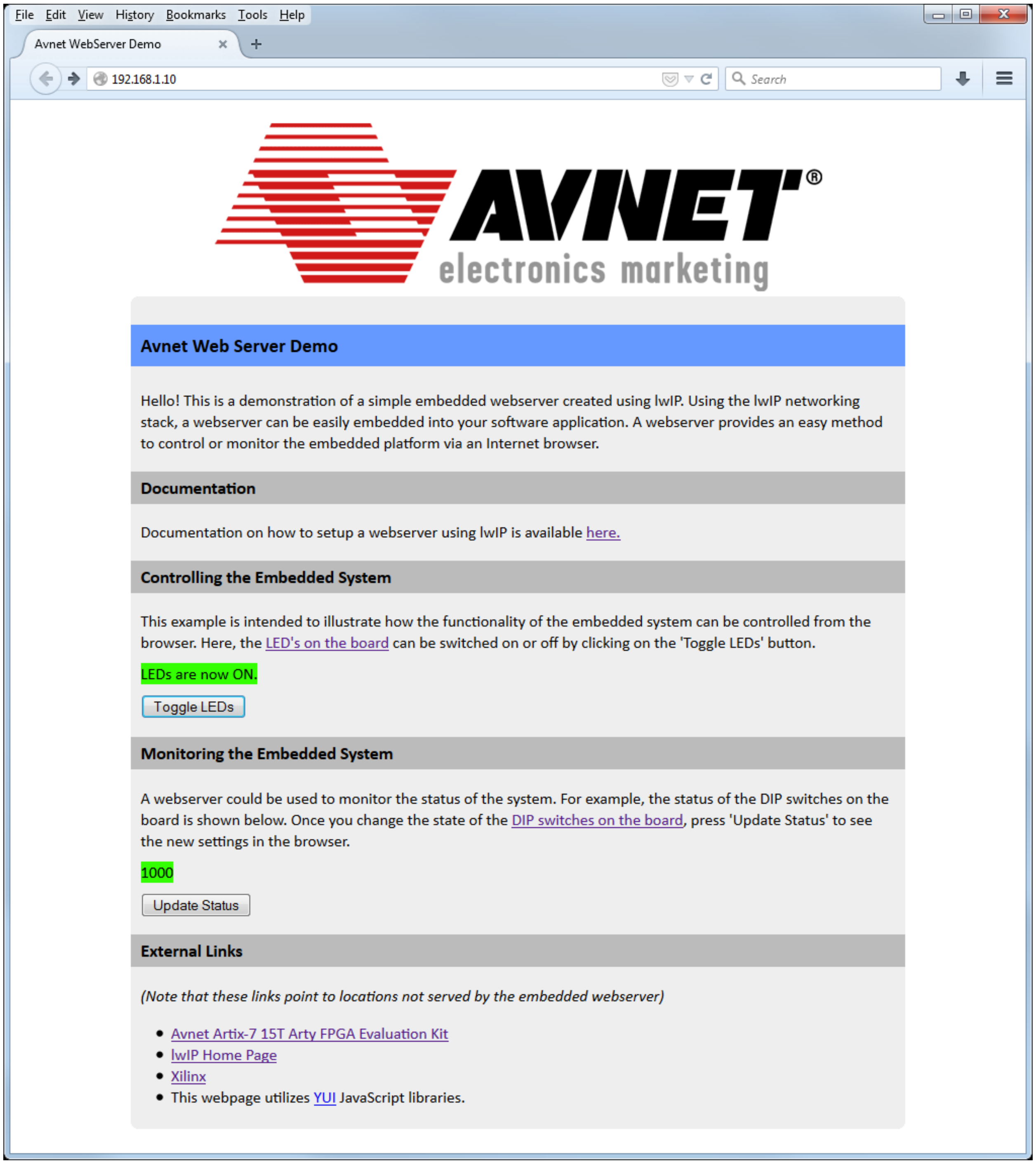
```
Download Progress..10.20.30.40.50.60.70.80.90..Done
Setting PC with Program Start Address 0x00000000
Processor started. Type "stop" to stop processor

RUNNING>
Waiting 3 seconds to allow PHY negotiation to complete

------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 64.0 KByte
------------------------------------------------------------
[380] local 192.168.1.100 port 5001 connected with 192.168.1.10 port 49153
[ ID] Interval       Transfer     Bandwidth
[380]  0.0- 5.0 sec  21.5 MBytes  36.1 Mbits/sec
[380]  5.0-10.0 sec  21.5 MBytes  36.1 Mbits/sec
[380] 10.0-15.0 sec  21.5 MBytes  36.1 Mbits/sec
[380] 15.0-20.0 sec  21.5 MBytes  36.1 Mbits/sec
[380] 20.0-25.0 sec  21.5 MBytes  36.1 Mbits/sec
[380] 25.0-30.0 sec  21.5 MBytes  36.0 Mbits/sec
[380] 30.0-35.0 sec  21.5 MBytes  36.1 Mbits/sec
[380] 35.0-40.0 sec  21.5 MBytes  36.1 Mbits/sec
[380] 40.0-45.0 sec  21.4 MBytes  36.0 Mbits/sec
[380] 45.0-50.0 sec  21.5 MBytes  36.1 Mbits/sec
[380] 50.0-55.0 sec  21.5 MBytes  36.1 Mbits/sec
[380] 55.0-60.0 sec  21.5 MBytes  36.0 Mbits/sec
Waiting for server threads to complete. Interrupt again to force quit.
[380]  0.0-61.1 sec   263 MBytes  36.1 Mbits/sec
Terminate batch job (Y/N)? y

C:\Projects\DRC_Dev\7A35T_Arty\7A35T_Arty_Ethernetlite_LwIP_VIV2015_2\demo>
```

[2] Host machine used to generate the results was an HP desktop with 3.4 GHz Core i7 CPU, 16 GB RAM and an Intel 82579LM network adaptor, running Windows 7. The host was directly connected with the board with a Cat-5E Ethernet cable.

HTTP Server Demo

The web server implements only a subset of the HTTP 1.1 protocol and demonstrates the control and monitoring an embedded system via a browser.

1. Open an HTML browser and point to the URL `http://192.168.1.10`, where **`192.168.1.10`** is the IP address of the board. The webserver demo should appear in your browser as shown in the following figure.


File Edit View History Bookmarks Tools Help
Avnet WebServer Demo
192.168.1.10
Search
AVNET electronics marketing

Avnet Web Server Demo

Hello! This is a demonstration of a simple embedded webserver created using lwIP. Using the lwIP networking stack, a webserver can be easily embedded into your software application. A webserver provides an easy method to control or monitor the embedded platform via an Internet browser.

Documentation

Documentation on how to setup a webserver using lwIP is available here.

Controlling the Embedded System

This example is intended to illustrate how the functionality of the embedded system can be controlled from the browser. Here, the LED's on the board can be switched on or off by clicking on the 'Toggle LEDs' button.

LEDs are now ON.

Toggle LEDs

Monitoring the Embedded System

A webserver could be used to monitor the status of the system. For example, the status of the DIP switches on the board is shown below. Once you change the state of the DIP switches on the board, press 'Update Status' to see the new settings in the browser.

1000

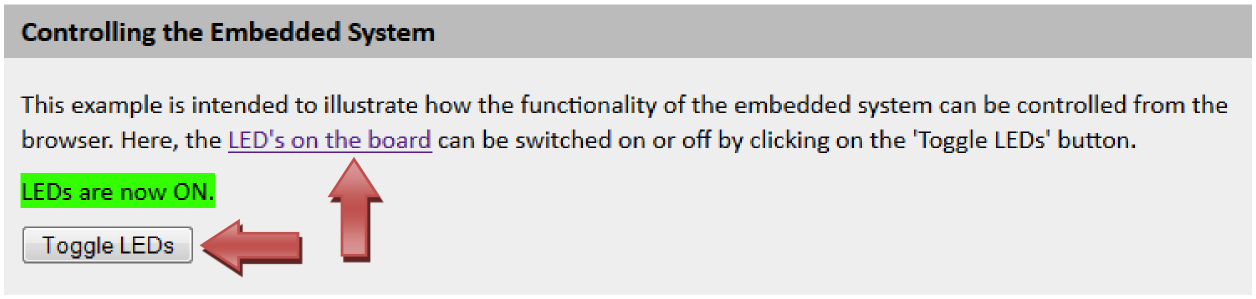
Update Status

External Links

(Note that these links point to locations not served by the embedded webserver)

- Avnet Artix-7 15T Arty FPGA Evaluation Kit
- lwIP Home Page
- Xilinx
- This webpage utilizes YUI JavaScript libraries.

a) You can remotely control the embedded system, in this case the LEDs on the board, by clicking on the button labeled "**Toggle LEDs.**" to turn the board LEDs ON or OFF. The current status of the LEDs is displayed in the web page and you can click the hyperlink labeled "**LEDs on the board**" to pop up a browser window showing a picture of the Arty board with the locations of the LEDs and DIP switches highlighted.

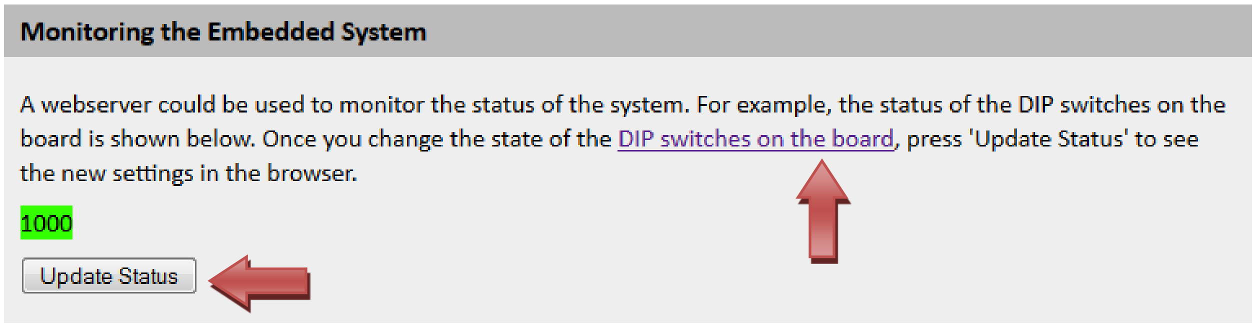


Controlling the Embedded System

This example is intended to illustrate how the functionality of the embedded system can be controlled from the browser. Here, the LED's on the board can be switched on or off by clicking on the 'Toggle LEDs' button.

LEDs are now ON.

b) You can monitor the state of the embedded system, in this case the DIP switch settings, by toggling them on and off while clicking on "**Update Status**". You should see the new value of the switch settings being displayed on the web page. The current status of the switches is displayed in the web page and you can click the hyperlink labeled "**DIP switches on the board**" to pop up a browser window showing a picture of the Arty board with the locations of the LEDs and DIP switches highlighted.

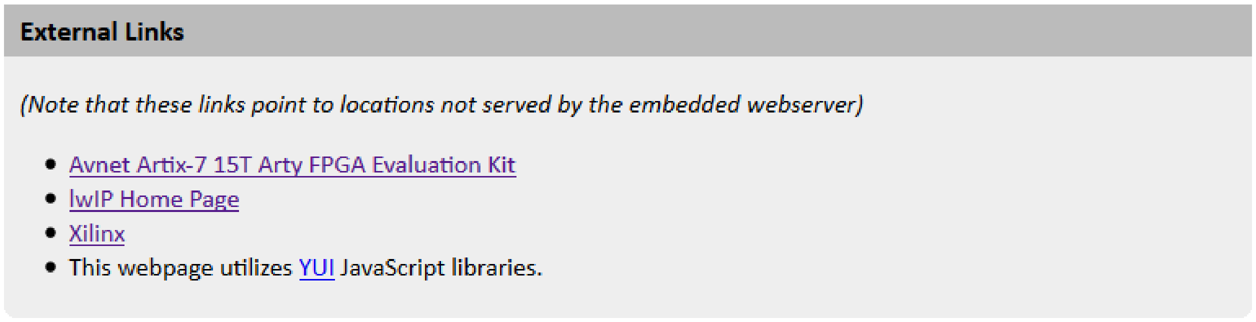


Monitoring the Embedded System

A webserver could be used to monitor the status of the system. For example, the status of the DIP switches on the board is shown below. Once you change the state of the DIP switches on the board, press 'Update Status' to see the new settings in the browser.

1000

c) If the webserver is also connected to the internet you can also click on the links below to access external websites for more information.



External Links

(Note that these links point to locations not served by the embedded webserver)

- Avnet Artix-7 15T Arty FPGA Evaluation Kit
- lwIP Home Page
- Xilinx
- This webpage utilizes YUI JavaScript libraries.

Echo Server Demo

This is a simple application to echo whatever is sent to the program over the network.

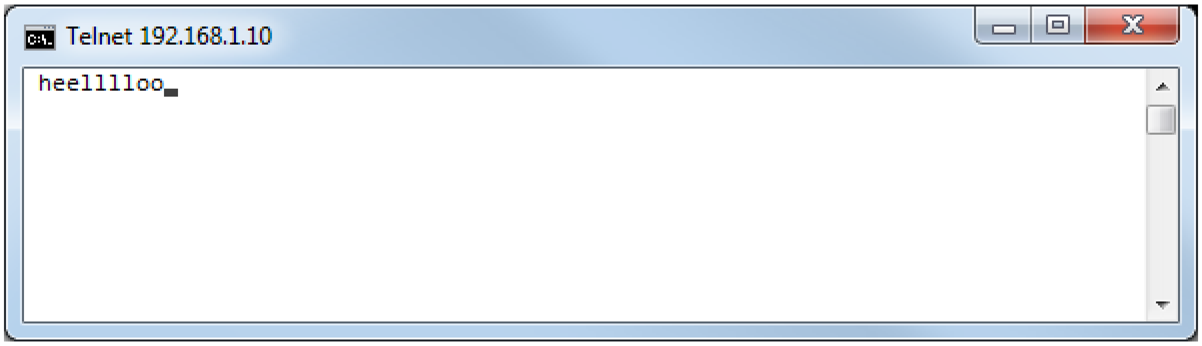
1. Open a command window in the **<installation>\demo** folder and enter:

   ```
   telnet 192.168.1.10 7
   ```

   The syntax for the telnet command is:

   ```
   telnet <target IP address> <IP port number>
   ```

2. The echo server will repeat, or echo, every character sent to it as you can see below. Try this out for yourself. Close this command window when you are done.



```
heellllloo_
```

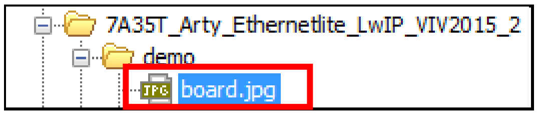
TFTP Server Demo

TFTP is a popular UDP based protocol in embedded systems for sending and receiving files over an Ethernet network. Although UDP does not guarantee reliable delivery of packets, TFTP implements a mechanism to guarantee packets are not lost during a file transfer. The Xilinx Memory Filesystem (xilmfs) used with this design is not large enough to write to (TFTP **put** command), so we'll demonstrate how to fetch a file from a remote filesystem using TFTP.

1. Open a command window in the **<installation>\demo** folder and enter:

   ```
   tftp -i 192.168.1.10 get images/board.jpg
   ```

2. You will see the file is downloaded from the board to the **<installation>\demo** folder.:



3. All the files in the **<installation>\memfs** folder are in memory on the Arty board and can be retrieved using TFTP. Experiment with retrieving other files. Close this command window when you are done.

Create the MicroBlaze System

As mentioned previously, this example design requires that the Xilinx Vivado and SDK tools be installed, licensed, and ready to run. It is beyond the scope of this document to show how to build this processor system from scratch. Rather, it is provided pre-built and ready to synthesize and implement.

Hardware Design Block Diagram

The following figure shows a high-level block diagram of the hardware design. The design requires:

- MicroBlaze processor
- 16KB of BRAM
- 32KB of instruction and data cache
- 256MB of DDR3 SDRAM
- Xilinx 10/100 AXI_Ethernetlite IP
- UART Port
- LEDs and DIP switches (not used by RxTest or TxTest)
- Interrupt Controller
- Timer

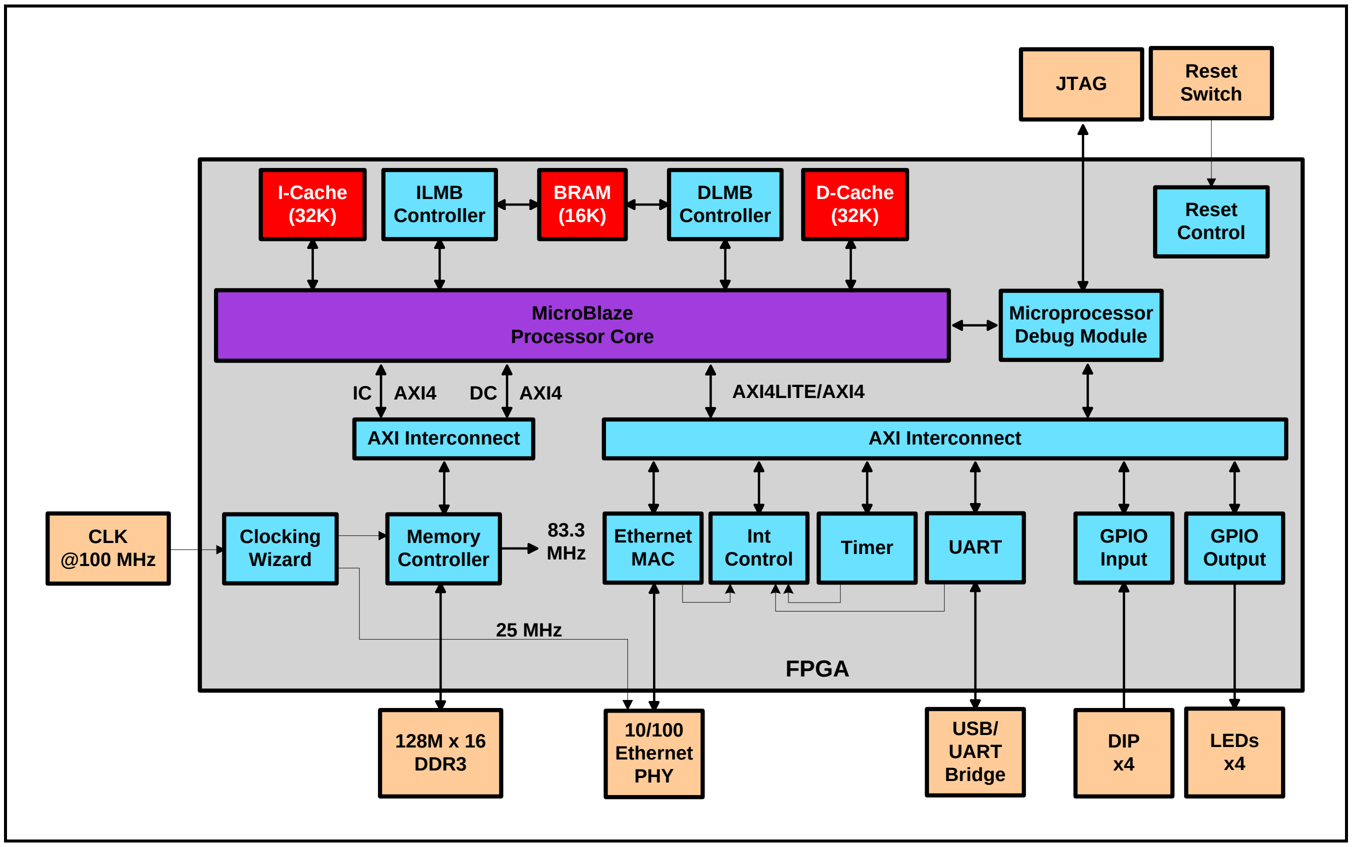


Figure 2 – Reference Design Block Diagram

Description of Hardware Modifications

This design was created using the Vivado IP Integrator Block Design flow with the following modifications:

- MicroBlaze processor settings:
    - Select the Typical Predefined Configuration
    - Select performance (throughput) optimized MicroBlaze
    - Enable the barrel shifter
    - Enable the MUL32 integer multiplier
    - Enable the integer divider
    - Enable the pattern comparator
    - Enable reversed load/store and swap instructions
    - Specify 32KB for instruction and data cache
    - Configure the i-cache and d-cache with an 8 word cache line
    - Specify 16KB for local memory size
- AXI_Ethernetlite MAC settings
    - Enable second Rx and Tx buffers
    - Change the AXI interface from AXI4Lite to AXI4
- XADC Wizard settings
    - Enable Temp Bus status port
- Clocking Wizard settings
    - Set clk_out1 to 200MHz (DDR3 reference)
    - Set clk_out2 to 166.667MHz (DDR3 controller)
    - Set clk_out3 to 25MHz (Ethernet PHY reference)
    - Disable the reset input
    - Disable the locked output
- AXI_UARTLite
    - Set baud rate to 115200
- Other settings
    - Add a axi_timer with interrupt

Vivado IP Integrator

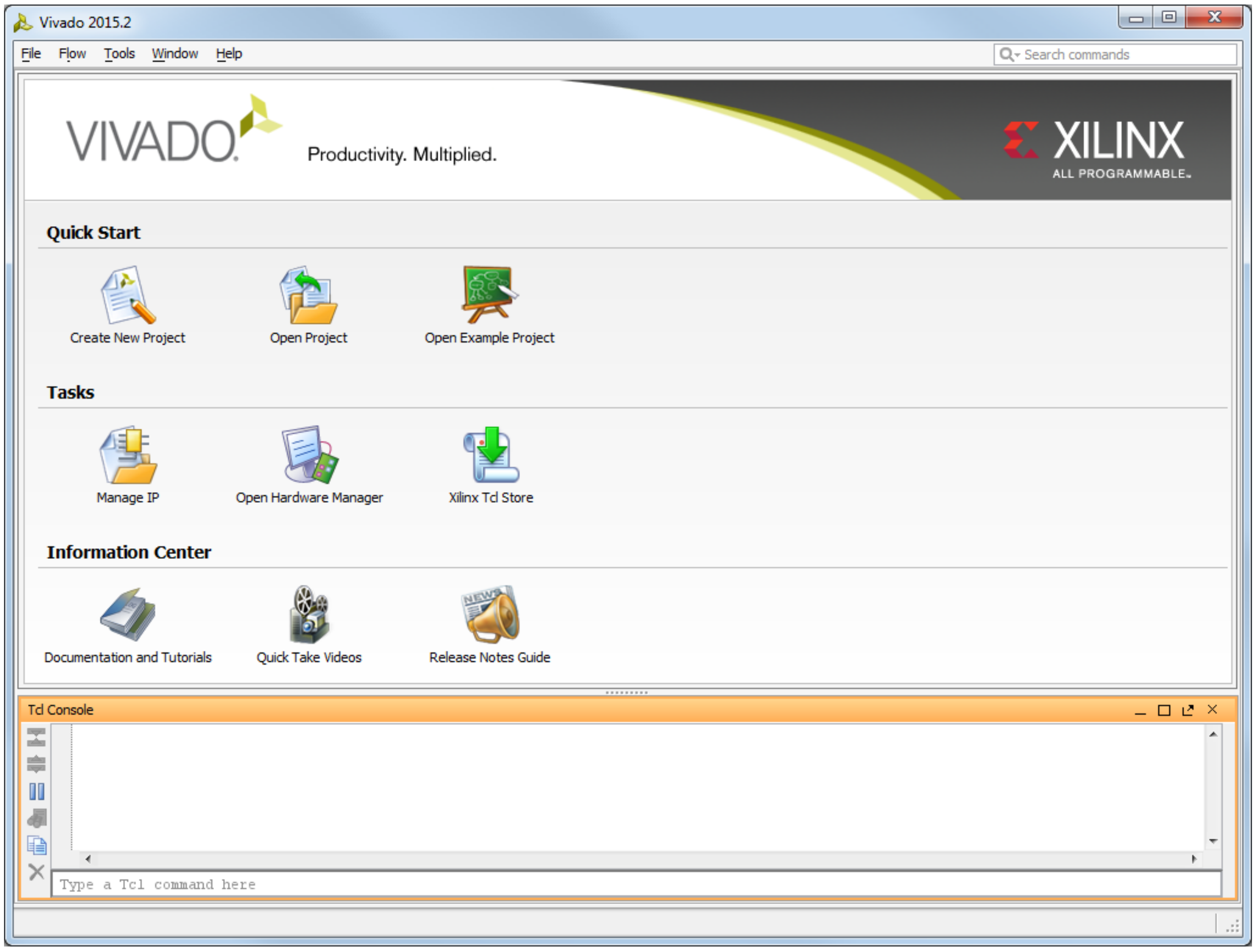
Vivado IP Integrator (IPI) provides a rich graphical environment in which to create and customize MicroBlaze processor systems. The integrated TCL command window allows for running simple commands. In fact, most functions and tasks in the Vivado GUI are run as TCL commands. The TCL command window can also be used to automate complex tasks like creating a MicroBlaze system from scratch that is capable of running the LwIP Ethernet applications. Follow the steps below to import and implement a pre-built known-good MicroBlaze system block design.

Install Board Definition Files

1. If you haven't already done so, please follow the instructions in Appendix II to install the board definition XML file set for the Arty board. The board definition file identifies the interfaces on the Arty board and allows us to target the board directly in the Vivado tools by selecting it from a menu.

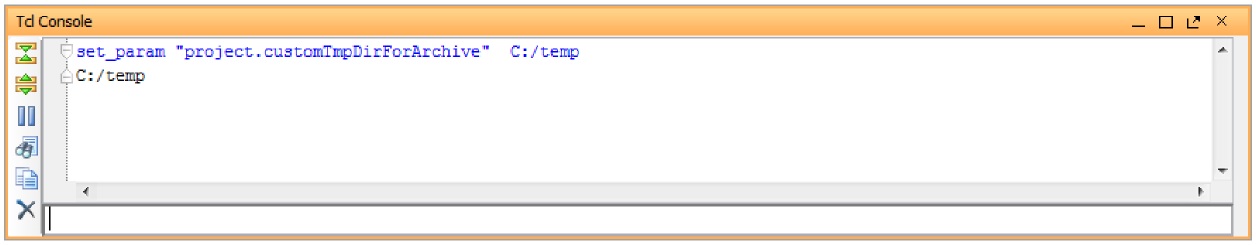
Create the Vivado IPI Block Design Project

1. Navigate to the Windows Desktop or Start menu and launch Vivado.



2. If you are using the Vivado Design Suite on a Windows®-7 host, there is a known issue that Vivado file path names often exceed the maximum allowed by Windows. Please see Appendix III for more details. A workaround for this is to change the temp folder that Vivado uses. Create your own temporary directory named C:\temp, and force Vivado to use the new folder with the following TCL command:

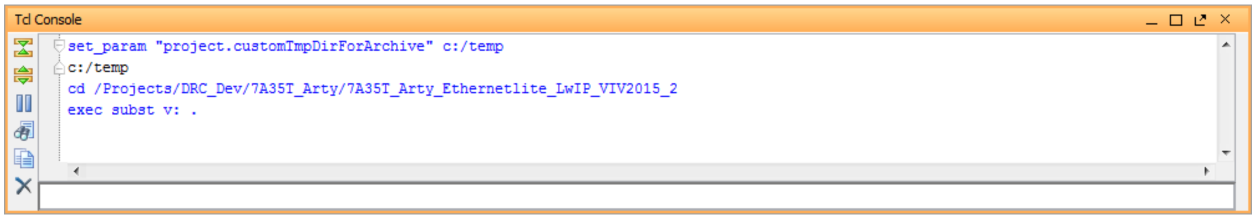
```
set_param “project.customTmpDirForArchive”  C:/temp
```



```
set_param "project.customTmpDirForArchive"  C:/temp
C:/temp
```

3. Another workaround for the 260 character path length limitation is to use the Windows **subst** command to substitute a drive letter for a long file path. Substitute the drive letter V: (or any other unused drive letter on your Windows PC) for the path where the zip archive of this example design was extracted. This can be done in the Vivado TCL Console:

```
cd <installation>
exec subst V:  .
```

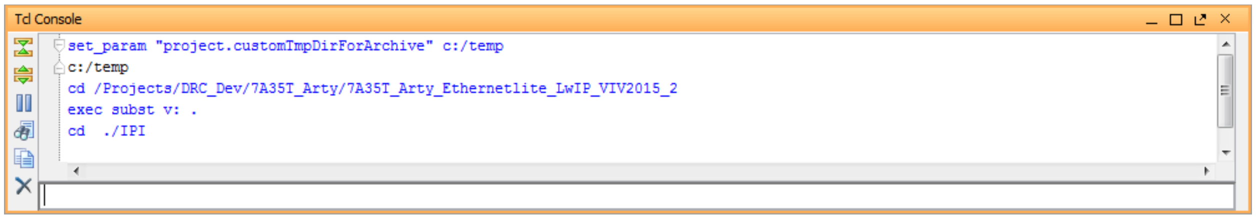


```
set_param "project.customTmpDirForArchive" c:/temp
c:/temp
cd /Projects/DRC_Dev/7A35T_Arty/7A35T_Arty_Ethernetlite_LwIP_VIV2015_2
exec subst v: .
```

Import and Recreate the Block Design

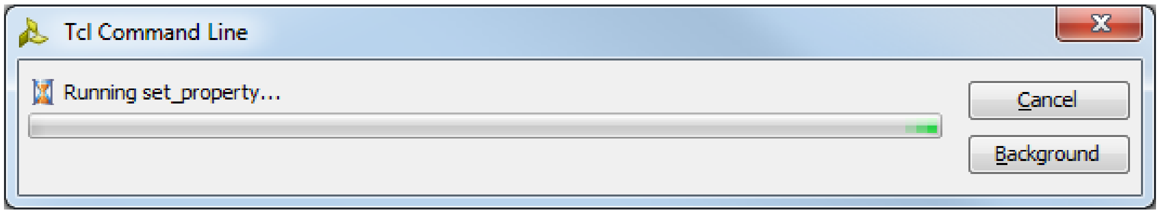
1. At this point we can create the Vivado project and import the pre-built MicroBlaze processor system block design. Open the **TCL Console** by selecting the tab at the bottom of the Vivado window and enter the following commands to create the block design:

```
cd ./IPI
source ./design_1.tcl
```

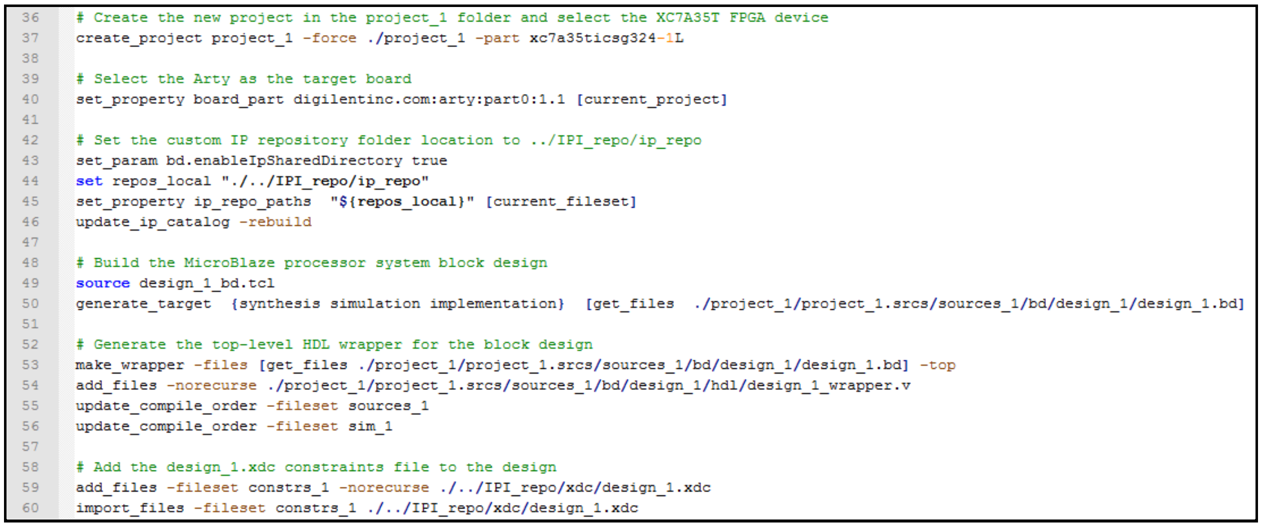


```
set_param "project.customTmpDirForArchive" c:/temp
c:/temp
cd /Projects/DRC_Dev/7A35T_Arty/7A35T_Arty_Ethernetlite_LwIP_VIV2015_2
exec subst v: .
cd  ./IPI
```

2. You will see the following window as the block design is imported and recreated. This process may take a few minutes.

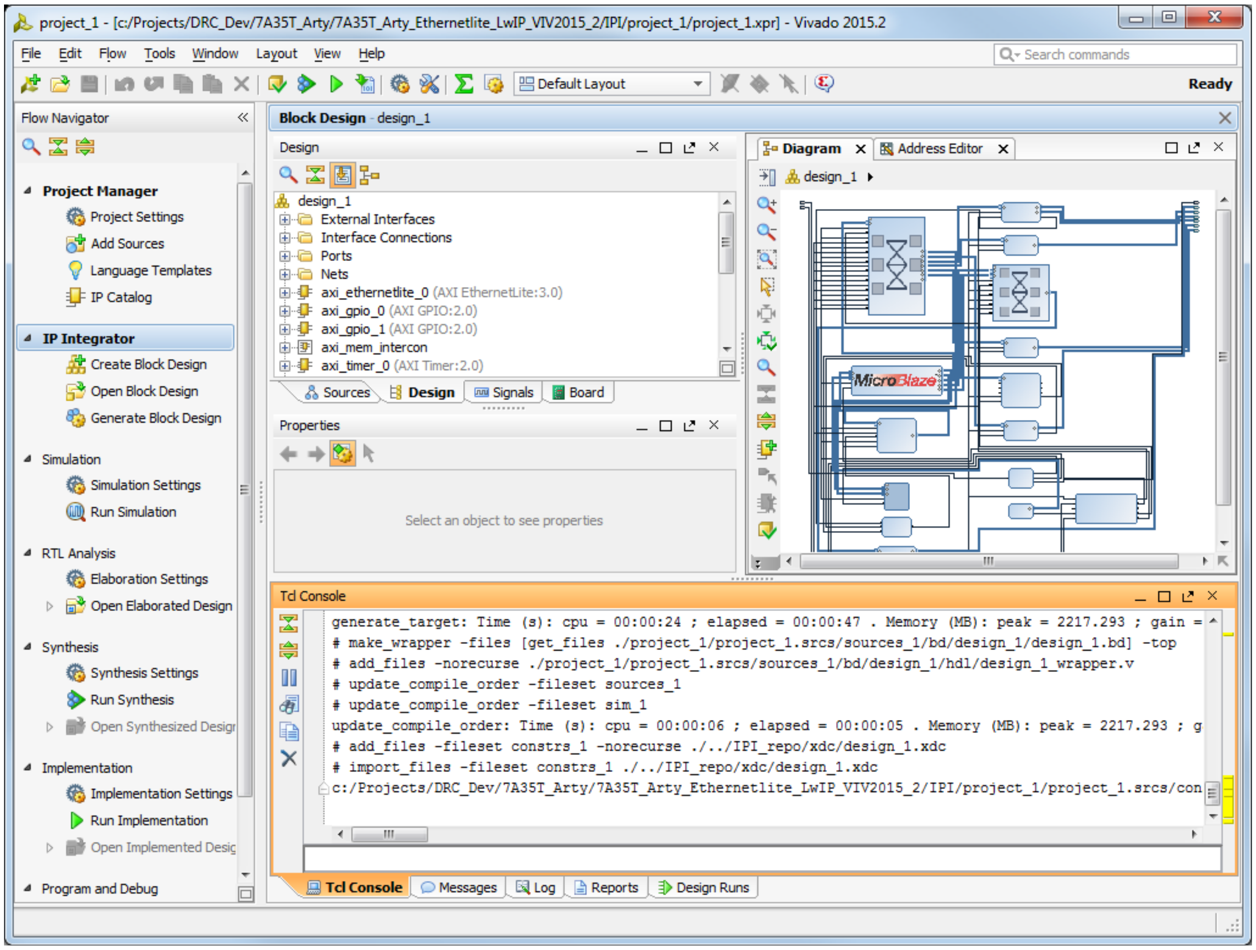


3. Open the **design_1.tcl** file in your favorite text editor and identify the important Vivado tasks that the script automates for us:
    - Create the new project in the **project_1** folder and select the **XC7A35T** FPGA device
    - Select the **Arty** as the target board
    - Set the custom IP repository folder location to **../IPI_repo/ip_repo**
    - Build the MicroBlaze processor system block design:
    - Generate the top-level HDL wrapper for the block design
    - Add the **design_1.xdc** constraints file to the design



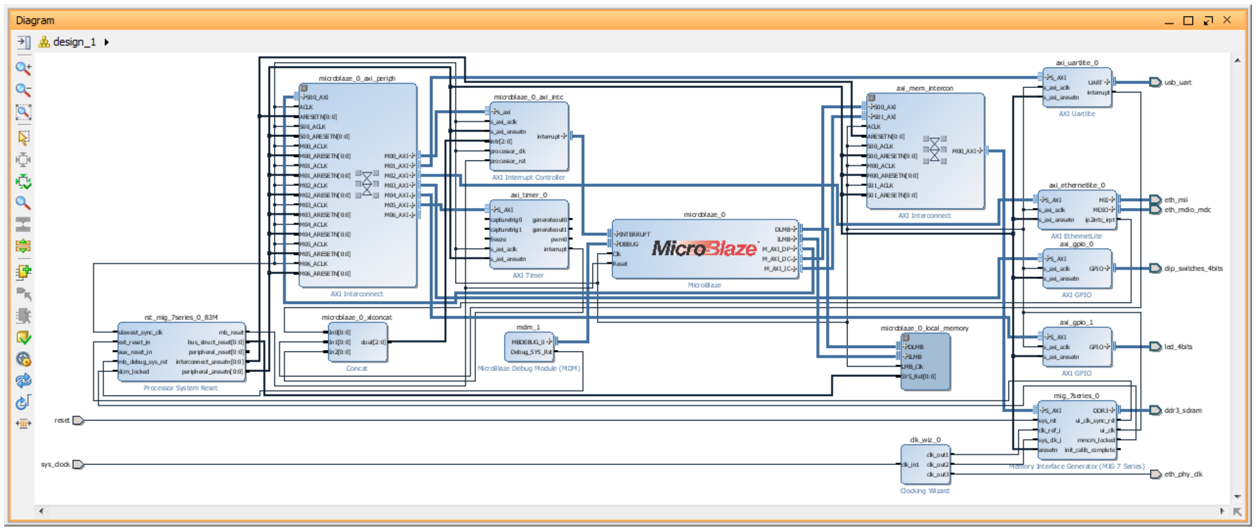
```
# Create the new project in the project_1 folder and select the XC7A35T FPGA device
create_project project_1 -force ./project_1 -part xc7a35ticsg324-1L

# Select the Arty as the target board
set_property board_part digilentinc.com:arty:part0:1.1 [current_project]

# Set the custom IP repository folder location to ../IPI_repo/ip_repo
set_param bd.enableIpSharedDirectory true
set repos_local "./../IPI_repo/ip_repo"
set_property ip_repo_paths  "${repos_local}" [current_fileset]
update_ip_catalog -rebuild

# Build the MicroBlaze processor system block design
source design_1_bd.tcl
generate_target  {synthesis simulation implementation}  [get_files  ./project_1/project_1.srcs/sources_1/bd/design_1/design_1.bd]

# Generate the top-level HDL wrapper for the block design
make_wrapper -files [get_files ./project_1/project_1.srcs/sources_1/bd/design_1/design_1.bd] -top
add_files -norecurse ./project_1/project_1.srcs/sources_1/bd/design_1/hdl/design_1_wrapper.v
update_compile_order -fileset sources_1
update_compile_order -fileset sim_1

# Add the design_1.xdc constraints file to the design
add_files -fileset constrs_1 -norecurse ./../IPI_repo/xdc/design_1.xdc
import_files -fileset constrs_1 ./../IPI_repo/xdc/design_1.xdc
```

4. When the design is done building you should see a Vivado window similar to this:

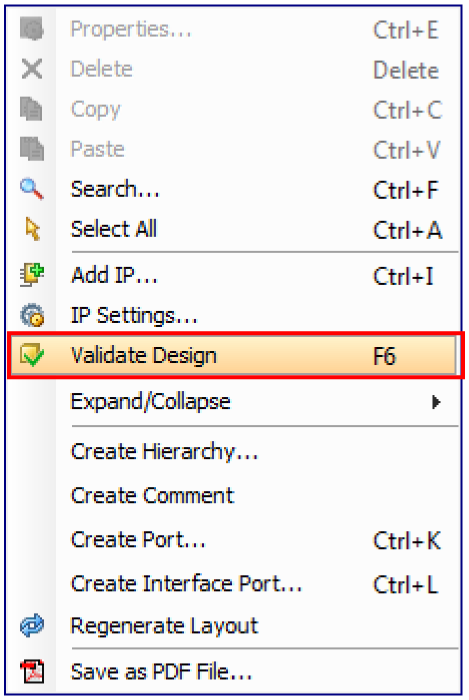


```
generate_target: Time (s): cpu = 00:00:24 ; elapsed = 00:00:47 . Memory (MB): peak = 2217.293 ; gain =
# make_wrapper -files [get_files ./project_1/project_1.srcs/sources_1/bd/design_1/design_1.bd] -top
# add_files -norecurse ./project_1/project_1.srcs/sources_1/bd/design_1/hdl/design_1_wrapper.v
# update_compile_order -fileset sources_1
# update_compile_order -fileset sim_1
update_compile_order: Time (s): cpu = 00:00:06 ; elapsed = 00:00:05 . Memory (MB): peak = 2217.293 ; g
# add_files -fileset constrs_1 -norecurse ./../IPI_repo/xdc/design_1.xdc
# import_files -fileset constrs_1 ./../IPI_repo/xdc/design_1.xdc
c:/Projects/DRC_Dev/7A35T_Arty/7A35T_Arty_Ethernetlite_LwIP_VIV2015_2/IPI/project_1/project_1.srcs/con
```

5. The Block Design should look similar to this:

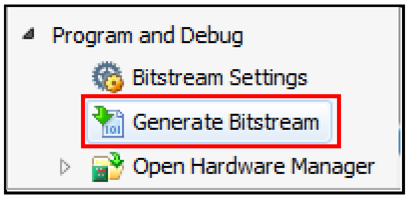


6. Right click in the block design **Diagram** window and select **Validate Design**. There should not be any errors or warnings. Click **OK** to continue:

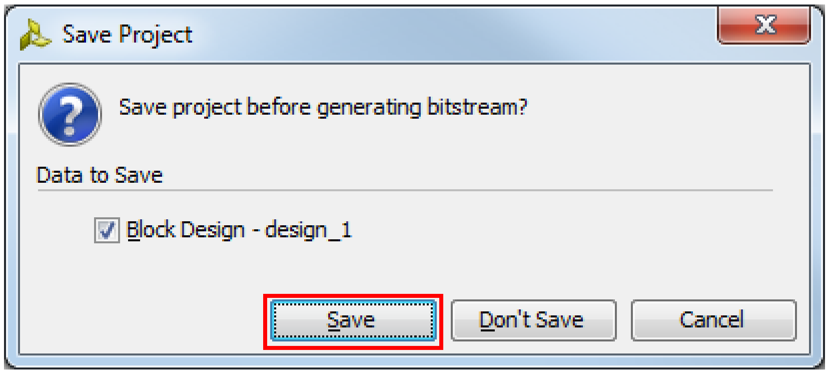


Implement the Design

1. We are now ready to build the design. Navigate to the **Project Manager** pane and click **Generate Bitstream**. This will run all previous steps to generate the IP netlists and wrappers and synthesize the design. You will see a window reminding you that synthesis hasn't been completed yet. Click **OK** to continue.

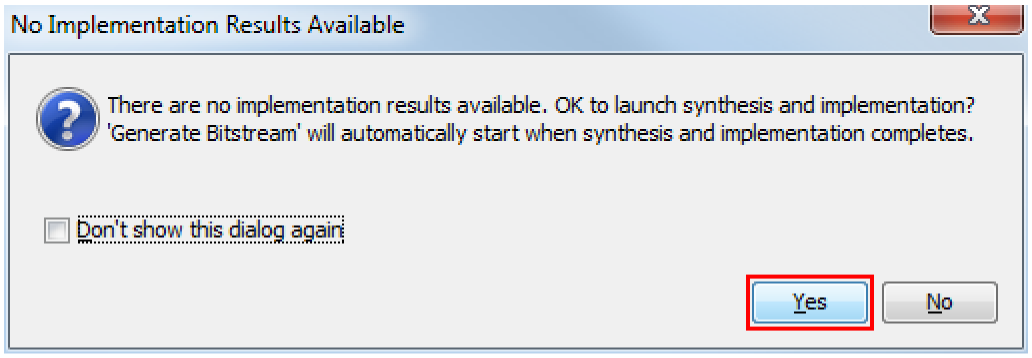


2. Click **Save** if you are prompted to save the block design:



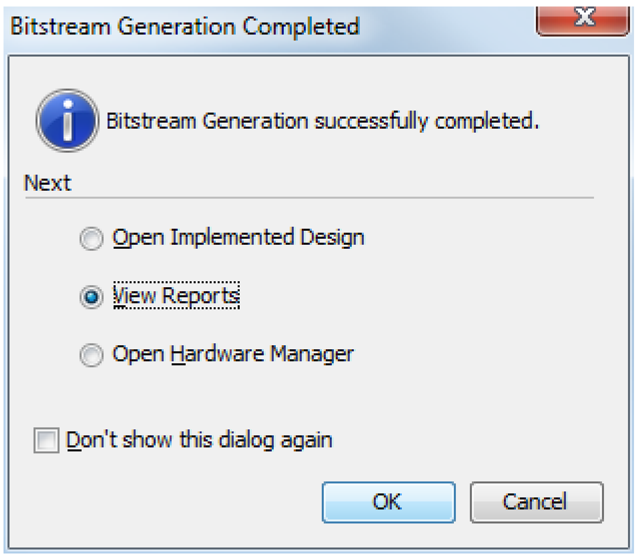
Save project before generating bitstream?

3. You will see a window reminding you that synthesis and implementation hasn't been completed yet. Click **Yes** to continue. This will take a few minutes.


No Implementation Results Available
Don't show this dialog again
Yes
No

There are no implementation results available. OK to launch synthesis and implementation?
'Generate Bitstream' will automatically start when synthesis and implementation completes.

4. When prompted select **View Reports** and click **OK**.



Bitstream Generation Completed

Bitstream Generation successfully completed.

Next

( ) Open Implemented Design

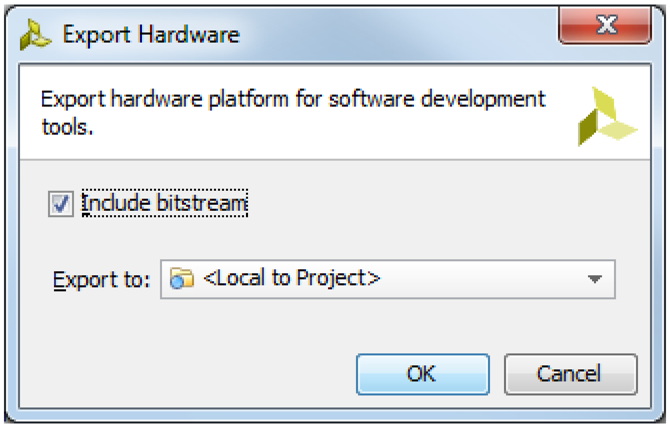
(●) View Reports

( ) Open Hardware Manager

☐ Don't show this dialog again

Export Vivado Hardware Design to the SDK

1. All software aspects of the design are performed inside a SDK Workspace. To generate an empty workspace based on the hardware platform built in Vivado navigate to **File → Export → Export Hardware** from the Vivado GUI. Accept the default **<Local to Project>** export directory location and check the box to **Include bitstream** then click **OK** to continue.



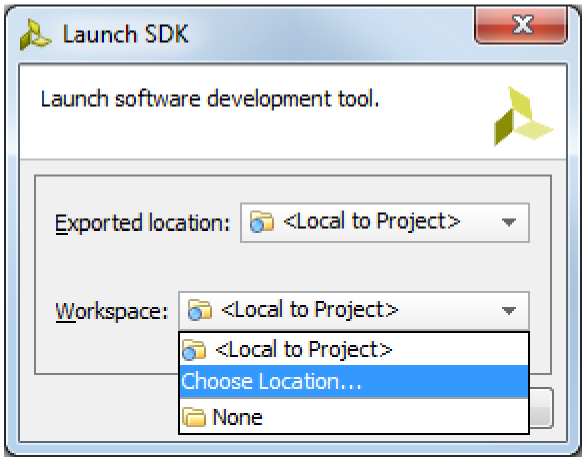
Export Hardware

Export hardware platform for software development tools.

☑ Include bitstream

Export to: <Local to Project>

2. Navigate to **File → Launch SDK** from the Vivado GUI. In the Launch SDK window click on the Workspace drop list and select **Choose Location**.



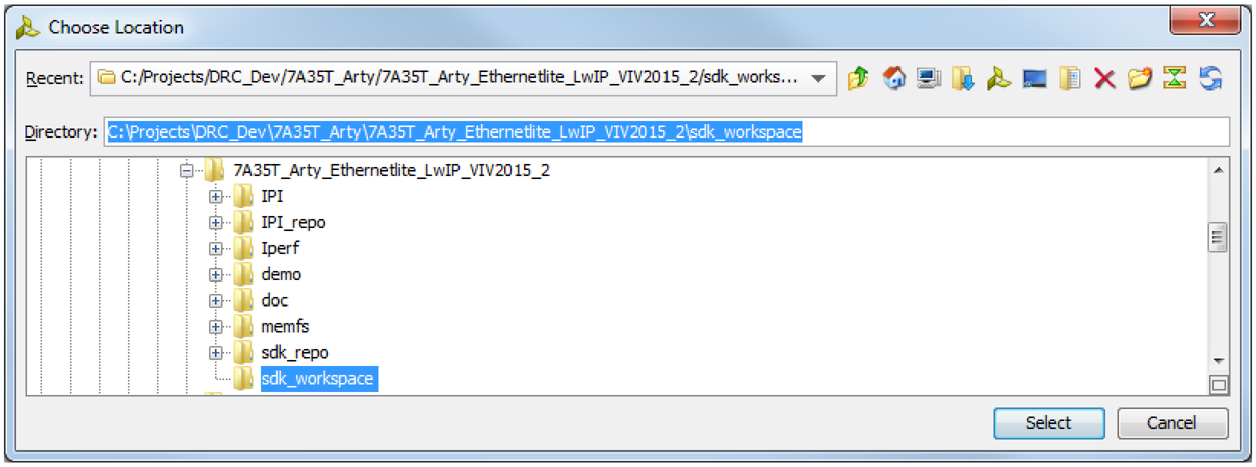
Launch SDK

Launch software development tool.

Exported location: <Local to Project>

Workspace: <Local to Project>

3. Navigate to the **<installation>\sdk_workspace** folder to create a new SDK workspace. ***Make sure there are NO SPACES in this path***. The Xilinx SDK does not tolerate spaces in this file path. Click **Select** and then click **OK** to continue.

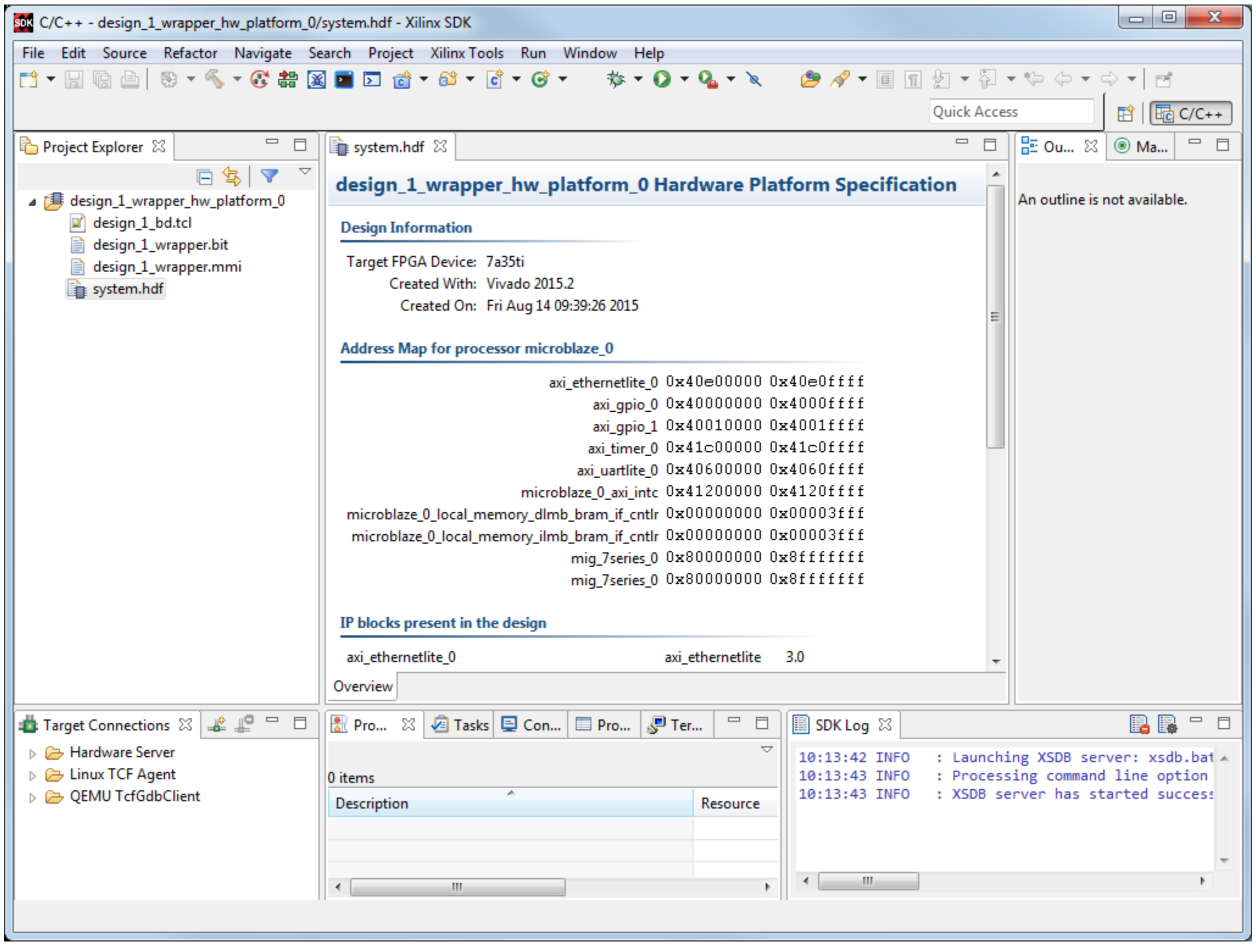


Recent: C:/Projects/DRC_Dev/7A35T_Arty/7A35T_Arty_Ethernetlite_LwIP_VIV2015_2/sdk_works...

Directory: C:\Projects\DRC_Dev\7A35T_Arty\7A35T_Arty_Ethernetlite_LwIP_VIV2015_2\sdk_workspace

```
7A35T_Arty_Ethernetlite_LwIP_VIV2015_2
    IPI
    IPI_repo
    Iperf
    demo
    doc
    memfs
    sdk_repo
    sdk_workspace
```

4. After a few seconds the SDK will show a GUI similar to the one shown below:



```
design_1_wrapper_hw_platform_0
    design_1_bd.tcl
    design_1_wrapper.bit
    design_1_wrapper.mmi
    system.hdf
```

design_1_wrapper_hw_platform_0 Hardware Platform Specification

Design Information

Target FPGA Device: 7a35ti
Created With: Vivado 2015.2
Created On: Fri Aug 14 09:39:26 2015

Address Map for processor microblaze_0

axi_ethernetlite_0	0x40e00000	0x40e0ffff
axi_gpio_0	0x40000000	0x4000ffff
axi_gpio_1	0x40010000	0x4001ffff
axi_timer_0	0x41c00000	0x41c0ffff
axi_uartlite_0	0x40600000	0x4060ffff
microblaze_0_axi_intc	0x41200000	0x4120ffff
microblaze_0_local_memory_dlmb_bram_if_cntlr	0x00000000	0x00003fff
microblaze_0_local_memory_ilmb_bram_if_cntlr	0x00000000	0x00003fff
mig_7series_0	0x80000000	0x8fffffff
mig_7series_0	0x80000000	0x8fffffff

IP blocks present in the design

axi_ethernetlite_0	axi_ethernetlite	3.0

An outline is not available.

```
Hardware Server
Linux TCF Agent
QEMU TcfGdbClient
```

0 items

Description	Resource

```
10:13:42 INFO  : Launching XSDB server: xsdb.bat
10:13:43 INFO  : Processing command line option
10:13:43 INFO  : XSDB server has started success
```

Create the SDK Workspace

Description of SDK Software BSP and Linker Modifications

The Xilinx Software Development Kit (SDK) is used for all software tasks. The software platform settings were modified from the default settings as follows:

- lwip library is generated
  - DHCP option **lwip_dhcp** is set to **true**
  - lwIP memory option **memp_n_pbuf** is set to **1024**
  - lwIP memory option **memp_n_tcp_seg** is set to **1024**
  - emac option **n_tx_descriptors** is set to **256**
  - emac option **n_rx_descriptors** is set to **256**
  - Pbuf option **pbuf_pool_size** is set to **1024**
  - emac option **phy_link_speed** is set to **CONFIG_LINKSPEED100**
  - TCP option **tcp_mss** is set to **1458**
  - TCP option **tcp_wnd** is set to **4096**

- xilmfs library is generated
  - **base_address** is set to **0x84000000**
  - **init_type** is set to **MFSINIT_IMAGE**
  - **need_utils** parameter is set to **true**
  - **numbytes** is set to **0xA00000**

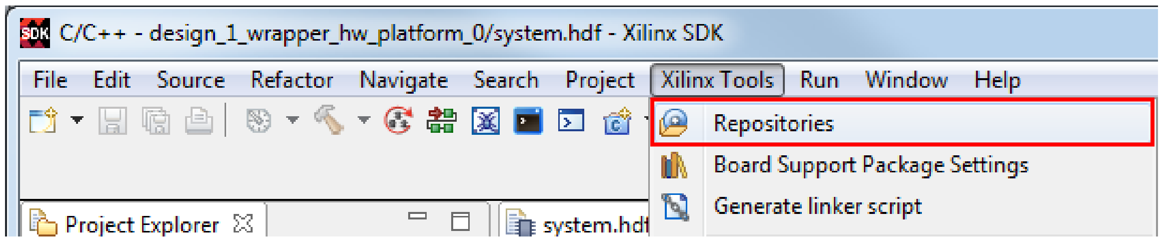
- Custom linker script
  - All code sections reside in DDR3
  - Heap and Stack reside in DDR3
  - Stack Size and Heap Size are both set to **x20000 (128KB)**

- Application uses the lwIP and xilmfs software libraries.

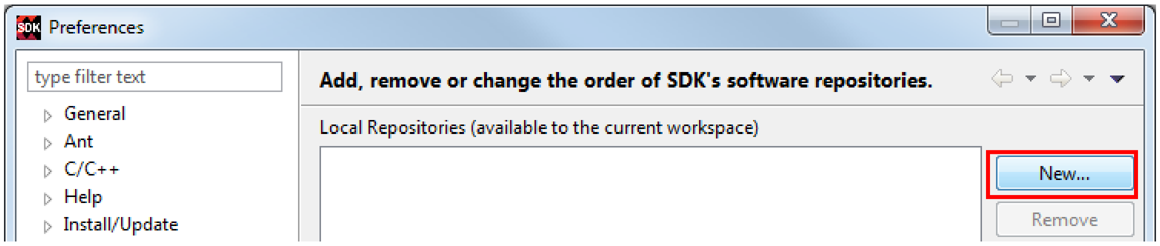
Add Custom Repository

Adding a custom repository to the SDK workspace is a convenient way to integrate software libraries and applications into your software development. We need to add a repository to the SDK workspace for the lwIP TCP/IP protocol stack library and the lwIP software applications. This is a patched version of the Xilinx SDK lwIP stack. This patched library fixes an issue that caused the SDK to create a lwIP BSP that does not work with the Arty board. Including the lwIP applications in a repository greatly simplifies the task of adding new software sources or application projects to the SDK workspace.

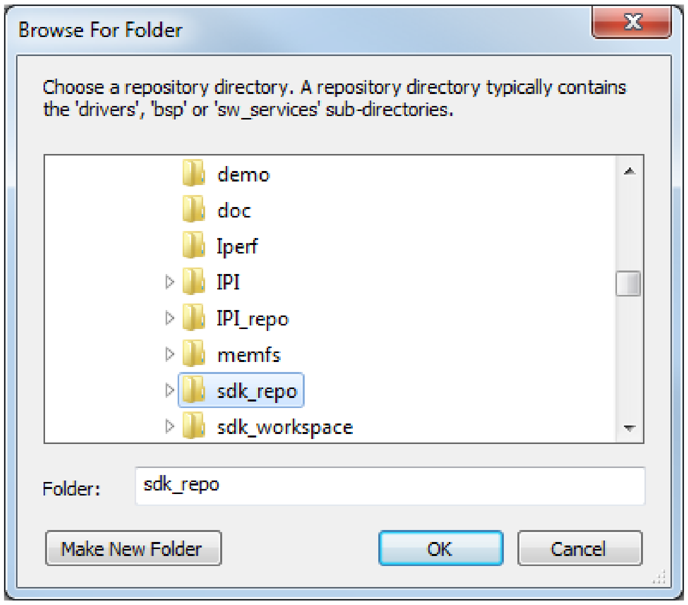
1. In the SDK GUI navigate to **Xilinx Tools → Repositories.**


C/C++ - design_1_wrapper_hw_platform_0/system.hdf - Xilinx SDK
File Edit Source Refactor Navigate Search Project Xilinx Tools Run Window Help
Repositories
Board Support Package Settings
Generate linker script
Project Explorer
system.hdf

2. Under **Local Repositories** click on **New** to create a repository that will only be visible and usable by this particular SDK workspace.

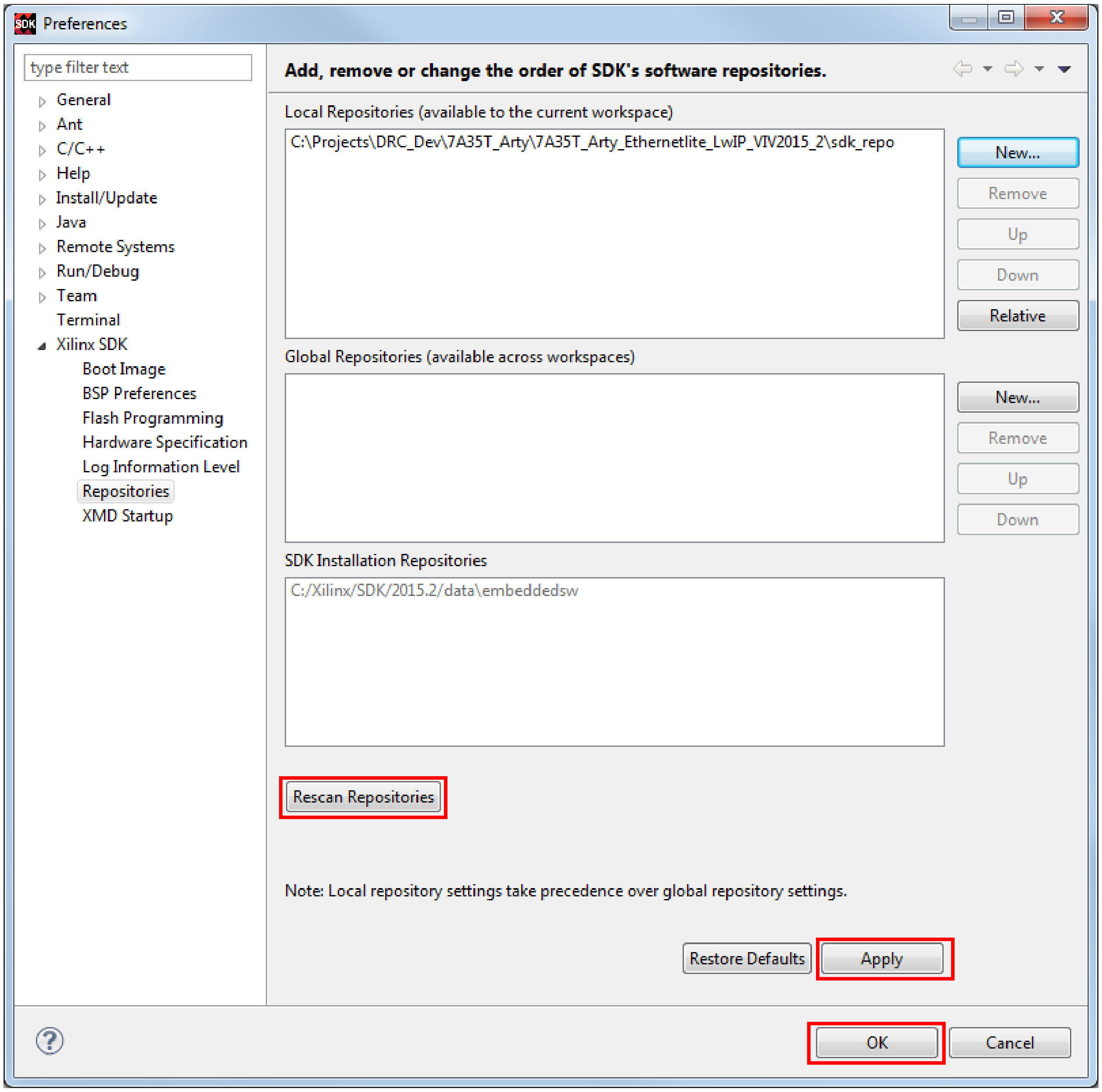

Preferences
type filter text
General
Ant
C/C++
Help
Install/Update
Add, remove or change the order of SDK's software repositories.
Local Repositories (available to the current workspace)
New...
Remove

3. Navigate to **<installation>\sdk_repo** and click **OK**.


Browse For Folder

Choose a repository directory. A repository directory typically contains the 'drivers', 'bsp' or 'sw_services' sub-directories.

4. Click on **Rescan Repositories**, then click **Apply**, then click **OK**. The new repository is now ready to use.



Add, remove or change the order of SDK's software repositories.

Local Repositories (available to the current workspace)

C:\Projects\DRC_Dev\7A35T_Arty\7A35T_Arty_Ethernetlite_LwIP_VIV2015_2\sdk_repo

Global Repositories (available across workspaces)

SDK Installation Repositories

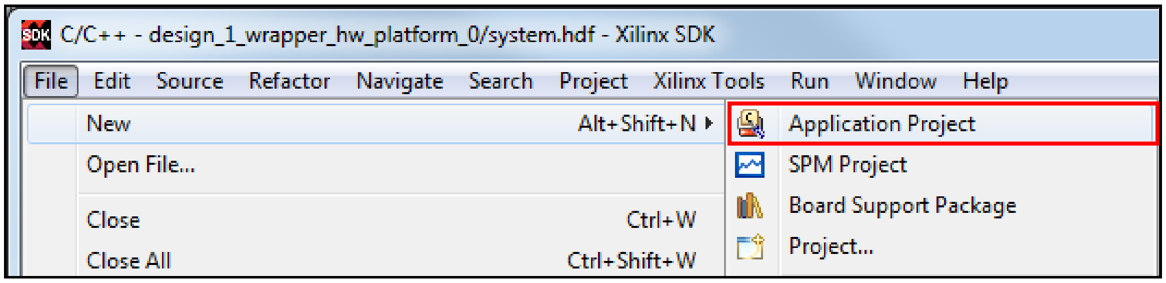
C:/Xilinx/SDK/2015.2/data\embeddedsw

Note: Local repository settings take precedence over global repository settings.

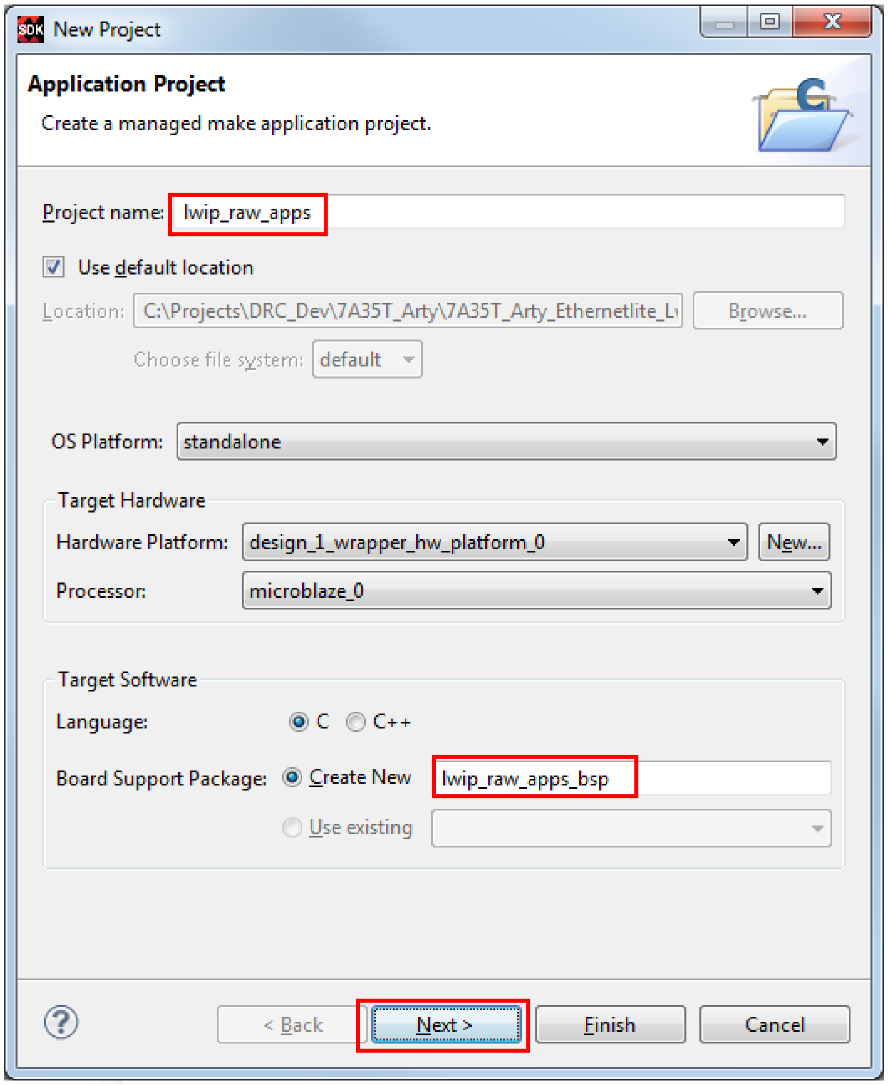
Create the Software Application

The next step is to create the suite of lwIP software applications.

1. Go to **File** → **New** → **Application Project**


C/C++ - design_1_wrapper_hw_platform_0/system.hdf - Xilinx SDK
File Edit Source Refactor Navigate Search Project Xilinx Tools Run Window Help
New Alt+Shift+N
Open File...
Close Ctrl+W
Close All Ctrl+Shift+W
Application Project
SPM Project
Board Support Package
Project...

2. Name the project **lwip_raw_apps** and accept the **default location** and **Target Hardware** settings. Accept the default **OS Platform** and **Language**. Accept the default to let the SDK **Create New** board support package named **lwip_raw_apps_bsp**. Click **Next** to continue.



Project name: lwip_raw_apps

[✓] Use default location

Location: C:\Projects\DRC_Dev\7A35T_Arty\7A35T_Arty_Ethernetlite_L Browse...

Choose file system: default

OS Platform: standalone

Target Hardware

Hardware Platform: design_1_wrapper_hw_platform_0 New...

Processor: microblaze_0

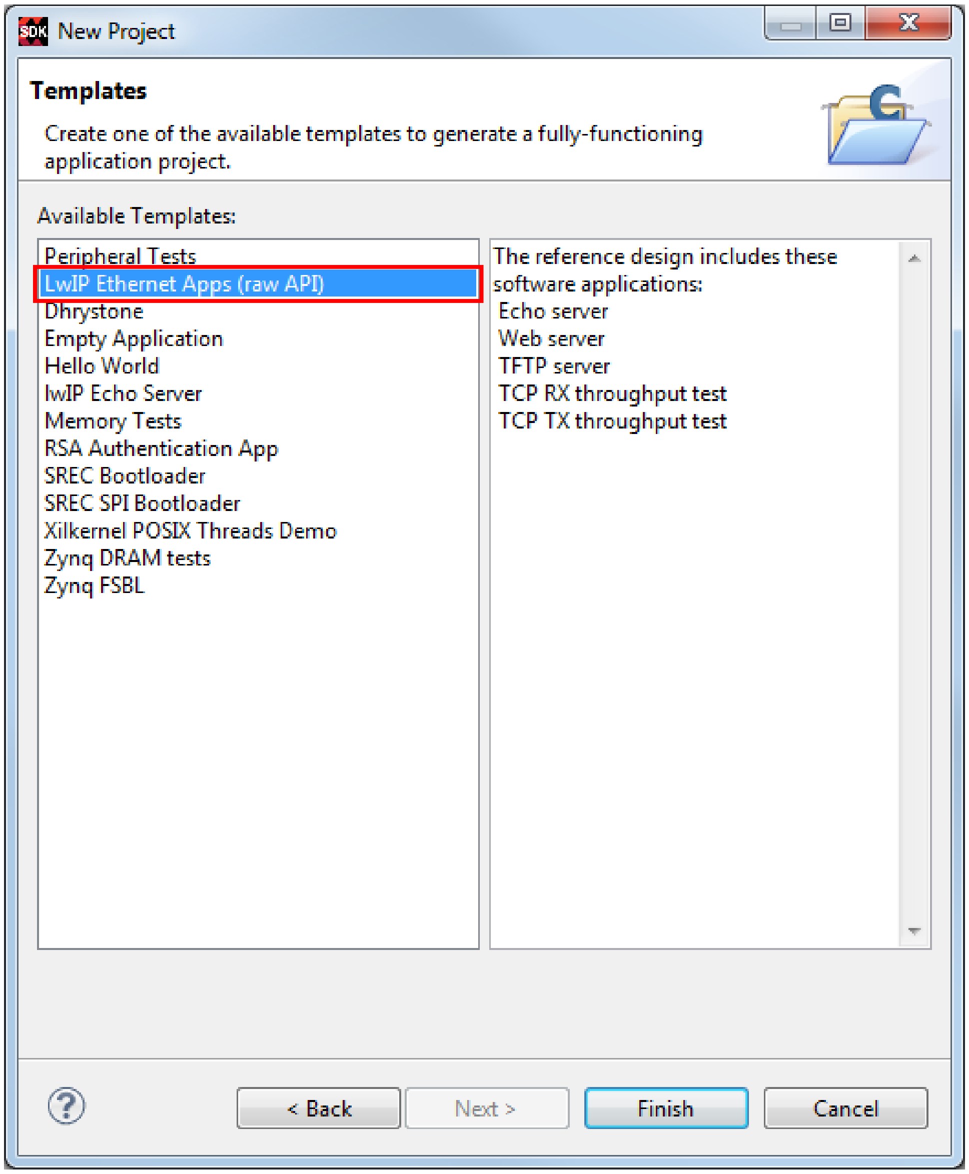
Target Software

Language: [● C] C++

Board Support Package: [● Create New] lwip_raw_apps_bsp

Use existing

3. Select the **LwIP Ethernet Apps (raw API)** template and click **Finish** to continue. This application appears in this list because it is in the repository we added earlier. The source code for the applications and BSP, along with the BSP settings, will be added to the workspace and compiled.



Templates

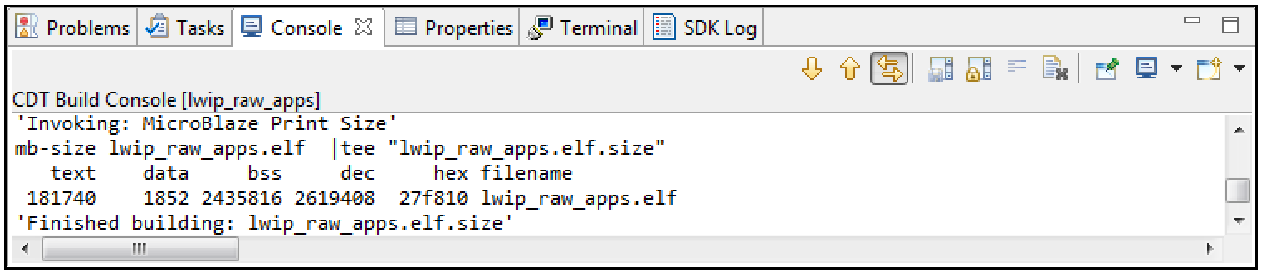
Create one of the available templates to generate a fully-functioning application project.

Available Templates:

Peripheral Tests
LwIP Ethernet Apps (raw API)
Dhrystone
Empty Application
Hello World
lwIP Echo Server
Memory Tests
RSA Authentication App
SREC Bootloader
SREC SPI Bootloader
Xilkernel POSIX Threads Demo
Zynq DRAM tests
Zynq FSBL

The reference design includes these software applications:
Echo server
Web server
TFTP server
TCP RX throughput test
TCP TX throughput test

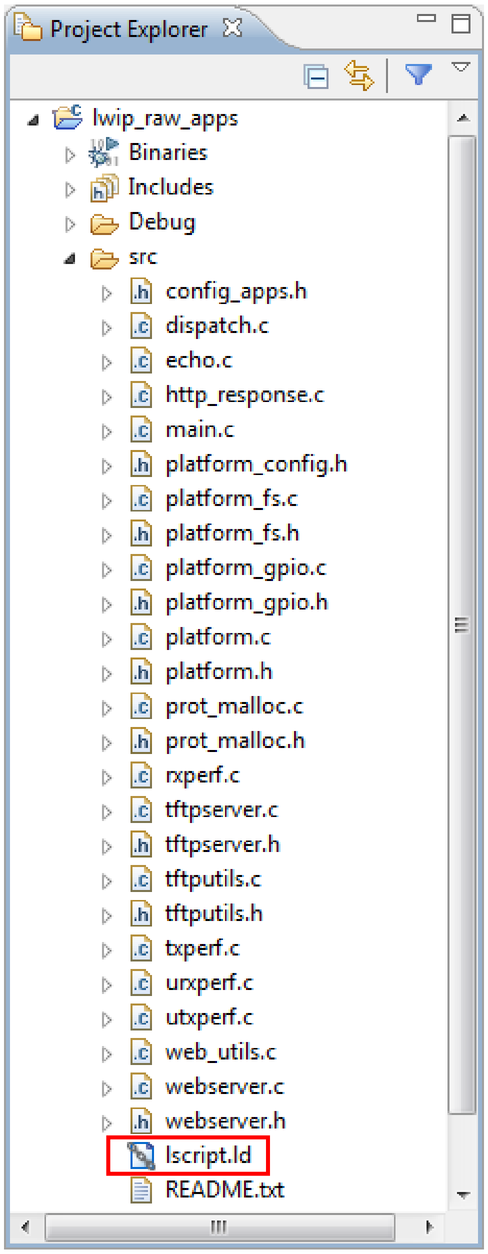
4. When compilation is complete the SDK should display a Console tab similar to this:



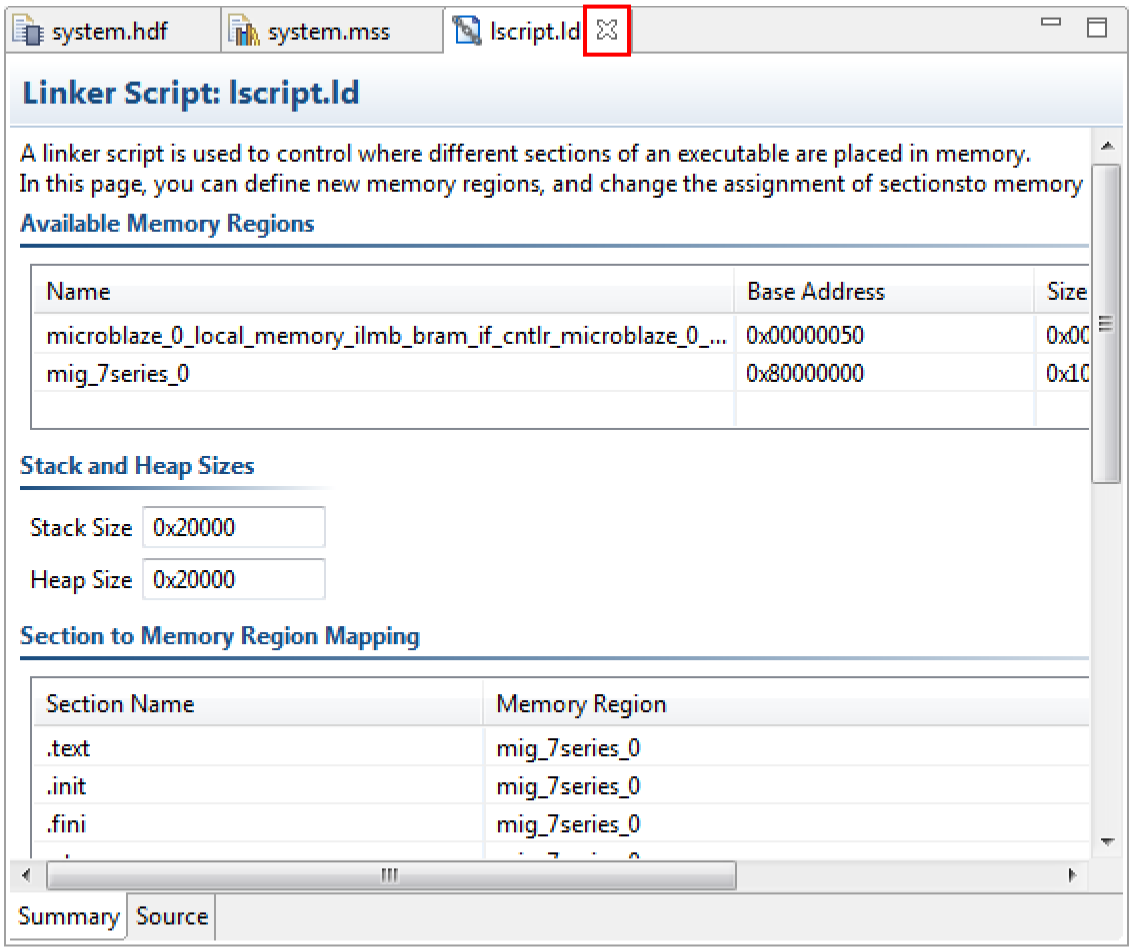
```
CDT Build Console [lwip_raw_apps]
'Invoking: MicroBlaze Print Size'
mb-size lwip_raw_apps.elf  |tee "lwip_raw_apps.elf.size"
   text    data     bss     dec     hex filename
 181740    1852 2435816 2619408  27f810 lwip_raw_apps.elf
'Finished building: lwip_raw_apps.elf.size'
```

Examine the Linker Script

1. Navigate the **lwip_raw_apps** software application project source tree in the **Project Explorer** pane and double-click on the **lscript.ld** linker script:



2. Notice that the **Stack Size** and **Heap Size** are both set to `0x20000 (128KB)` and that all of the code sections are mapped to be located in **mig_7series_0**, the DDR3 system RAM (you will need to scroll the window to see all of the code sections). This would also be a good time to verify the base address of the DDR3 is correct. Edit this field if you need to and save the file when you are done. Click the ☒ on the file tab to close the file.



Linker Script: lscript.ld

A linker script is used to control where different sections of an executable are placed in memory.
In this page, you can define new memory regions, and change the assignment of sectionsto memory

Available Memory Regions

Name	Base Address	Size
microblaze_0_local_memory_ilmb_bram_if_cntlr_microblaze_0_...	0x00000050	0x0C
mig_7series_0	0x80000000	0x1C

Stack and Heap Sizes

Stack Size 0x20000

Heap Size 0x20000

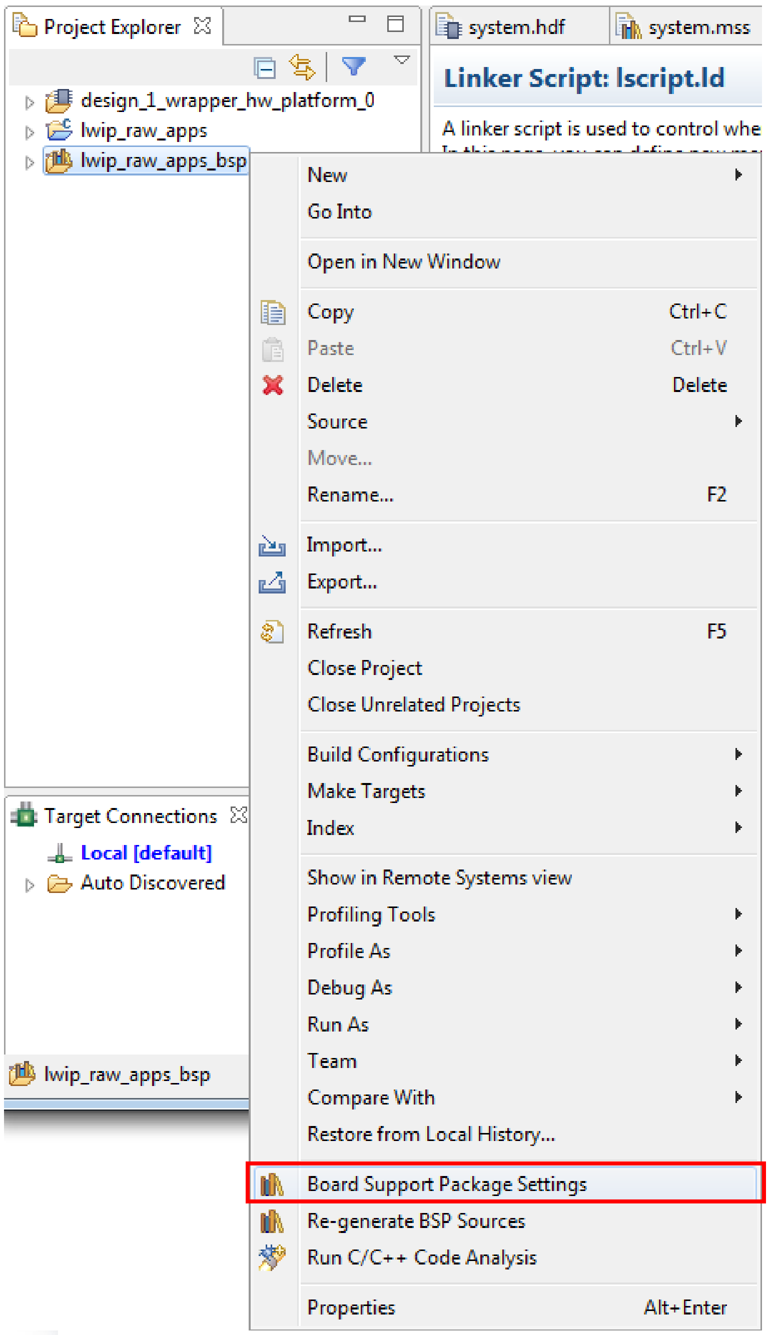
Section to Memory Region Mapping

Section Name	Memory Region
.text	mig_7series_0
.init	mig_7series_0
.fini	mig_7series_0

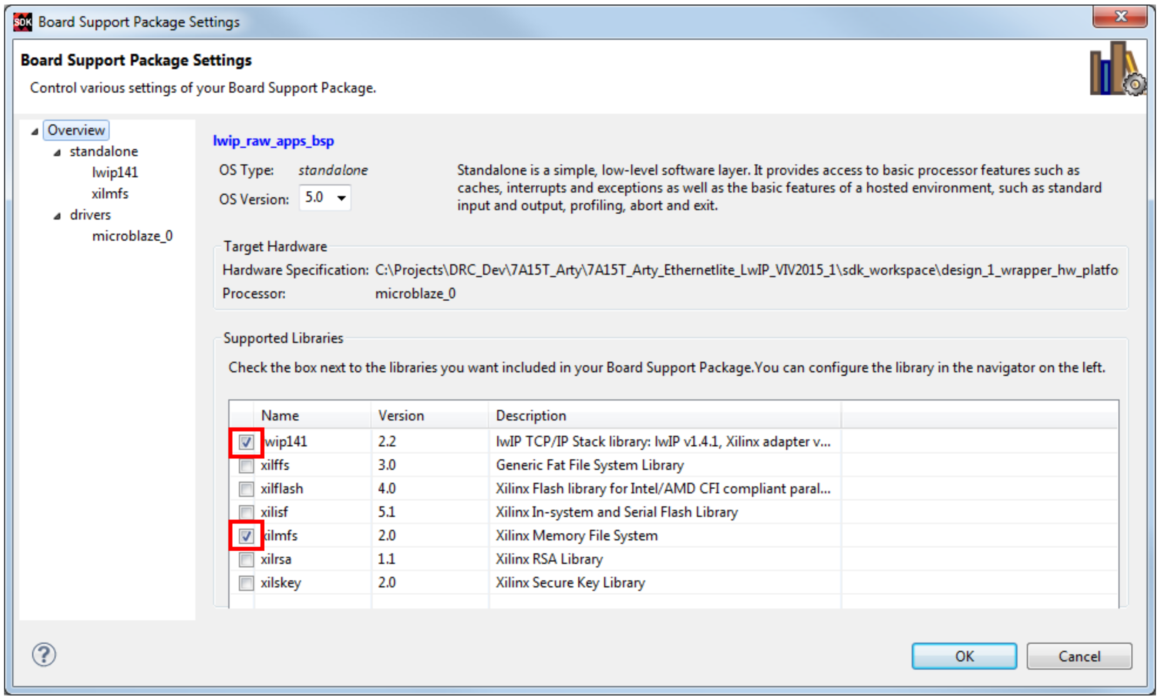
Examine the Board Support Package Settings

Any software application built in the SDK requires a Support Package (BSP) on which individual projects can be built. Multiple BSPs and multiple application projects can be held in a single SDK workspace. The BSP required to run the lwIP applications was built when we created the software application, and it includes many settings that significantly impact Ethernet performance. We will examine those settings in the steps below.

1. Right click on the **lwip_raw_apps_bsp** in the **Project Explorer** pane and select **Board Support Package Settings**



2. Notice that the **lwip141** and **xilmfs** libraries are already selected in the BSP. These were selected automatically when we create the lwip applications software project from the repository we added earlier. **Do not** click **OK** yet.



Board Support Package Settings

Control various settings of your Board Support Package.

lwip_raw_apps_bsp

OS Type: standalone

OS Version: 5.0

Standalone is a simple, low-level software layer. It provides access to basic processor features such as caches, interrupts and exceptions as well as the basic features of a hosted environment, such as standard input and output, profiling, abort and exit.

Target Hardware

Hardware Specification: C:\Projects\DRC_Dev\7A15T_Arty\7A15T_Arty_Ethernetlite_LwIP_VIV2015_1\sdk_workspace\design_1_wrapper_hw_platfo

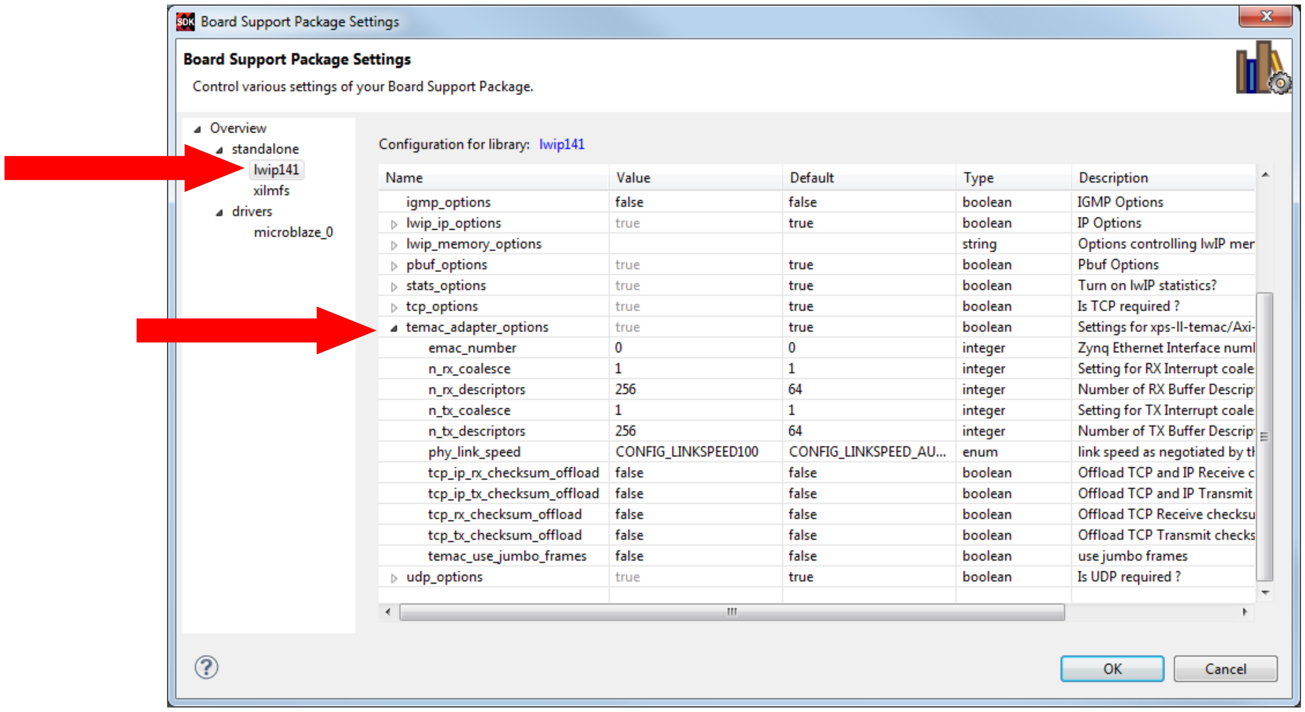
Processor: microblaze_0

Supported Libraries

Check the box next to the libraries you want included in your Board Support Package.You can configure the library in the navigator on the left.

	Name	Version	Description
✓	lwip141	2.2	lwIP TCP/IP Stack library: lwIP v1.4.1, Xilinx adapter v...
	xilffs	3.0	Generic Fat File System Library
	xilflash	4.0	Xilinx Flash library for Intel/AMD CFI compliant paral...
	xilisf	5.1	Xilinx In-system and Serial Flash Library
✓	xilmfs	2.0	Xilinx Memory File System
	xilrsa	1.1	Xilinx RSA Library
	xilskey	2.0	Xilinx Secure Key Library

3. In the left panel, select **lwip141** for the standalone OS to see the detailed parameters available for the lwIP library and click to expand the tree for the **temac_adapter_options** settings.



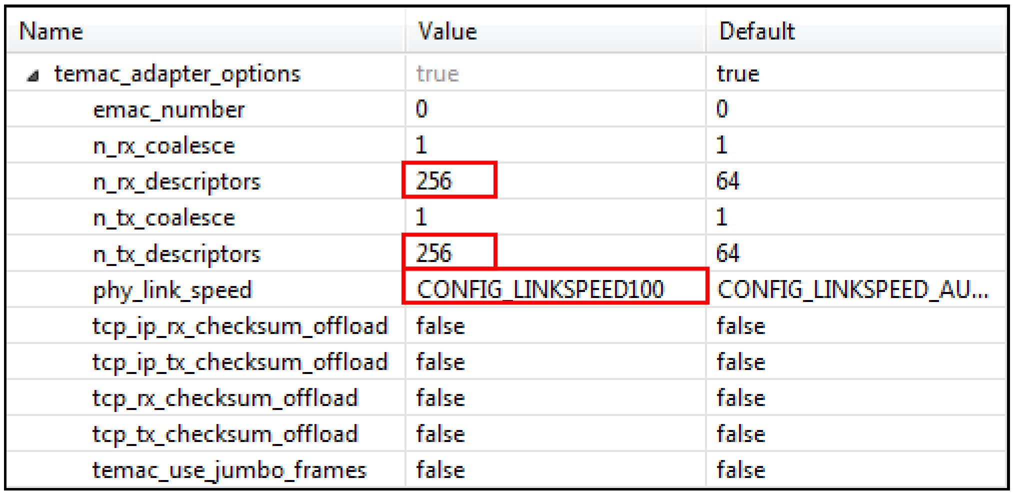
Board Support Package Settings

Control various settings of your Board Support Package.

Configuration for library: lwip141

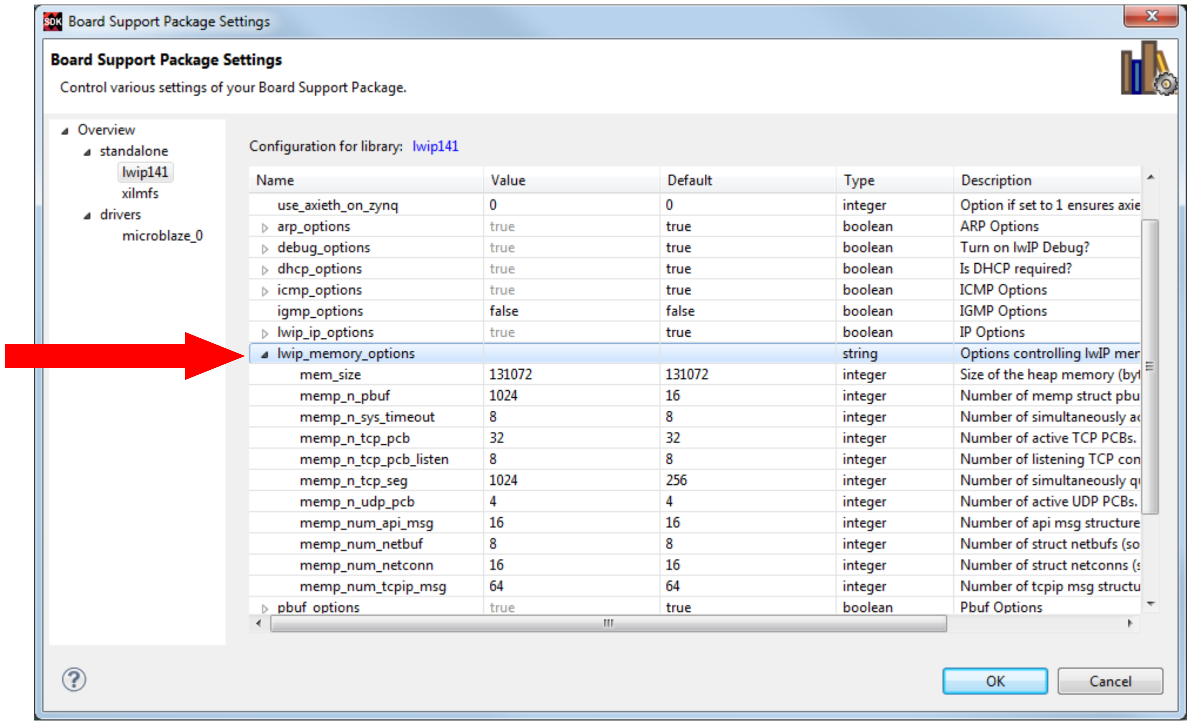
Name	Value	Default	Type	Description
igmp_options	false	false	boolean	IGMP Options
▷ lwip_ip_options	true	true	boolean	IP Options
▷ lwip_memory_options			string	Options controlling lwIP mer
▷ pbuf_options	true	true	boolean	Pbuf Options
▷ stats_options	true	true	boolean	Turn on lwIP statistics?
▷ tcp_options	true	true	boolean	Is TCP required ?
◢ temac_adapter_options	true	true	boolean	Settings for xps-ll-temac/Axi-
emac_number	0	0	integer	Zynq Ethernet Interface numl
n_rx_coalesce	1	1	integer	Setting for RX Interrupt coale
n_rx_descriptors	256	64	integer	Number of RX Buffer Descrip
n_tx_coalesce	1	1	integer	Setting for TX Interrupt coale
n_tx_descriptors	256	64	integer	Number of TX Buffer Descrip
phy_link_speed	CONFIG_LINKSPEED100	CONFIG_LINKSPEED_AU...	enum	link speed as negotiated by tl
tcp_ip_rx_checksum_offload	false	false	boolean	Offload TCP and IP Receive c
tcp_ip_tx_checksum_offload	false	false	boolean	Offload TCP and IP Transmit
tcp_rx_checksum_offload	false	false	boolean	Offload TCP Receive checksu
tcp_tx_checksum_offload	false	false	boolean	Offload TCP Transmit checks
temac_use_jumbo_frames	false	false	boolean	use jumbo frames
▷ udp_options	true	true	boolean	Is UDP required ?

4. To maximize Ethernet throughput it is necessary to increase the number of Tx and Rx descriptors to be used. The optimal value for the number of **n_tx_descriptors** and **n_rx_descriptors** for this example design is **256**. We also need to set the **phy_link_speed** to **CONFIG_LINKSPEED100**. Normally it would be desirable to leave this set for auto-negotiation, but this setting is PHY dependent and has only been tested with Marvell PHYs used on Xilinx development boards. The maximum throughput of the axi_ethernetlite peripheral is 100Mbps, so we set the link speed to 100Mbps here. This setting must be correct and must match the capabilities of the Ethernet MAC and PHY in order to transmit and receive packets. **Do not** click **OK** yet.



Name	Value	Default
◢ temac_adapter_options	true	true
emac_number	0	0
n_rx_coalesce	1	1
n_rx_descriptors	256	64
n_tx_coalesce	1	1
n_tx_descriptors	256	64
phy_link_speed	CONFIG_LINKSPEED100	CONFIG_LINKSPEED_AU...
tcp_ip_rx_checksum_offload	false	false
tcp_ip_tx_checksum_offload	false	false
tcp_rx_checksum_offload	false	false
tcp_tx_checksum_offload	false	false
temac_use_jumbo_frames	false	false

5. Click to expand the tree for the **lwip_memory_options:**



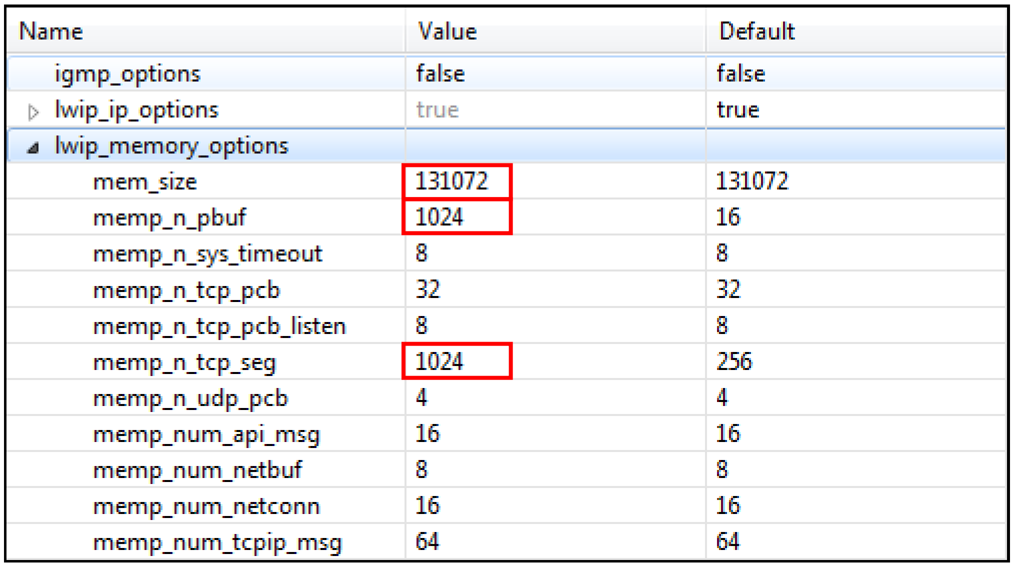
Board Support Package Settings

Control various settings of your Board Support Package.

Configuration for library: lwip141

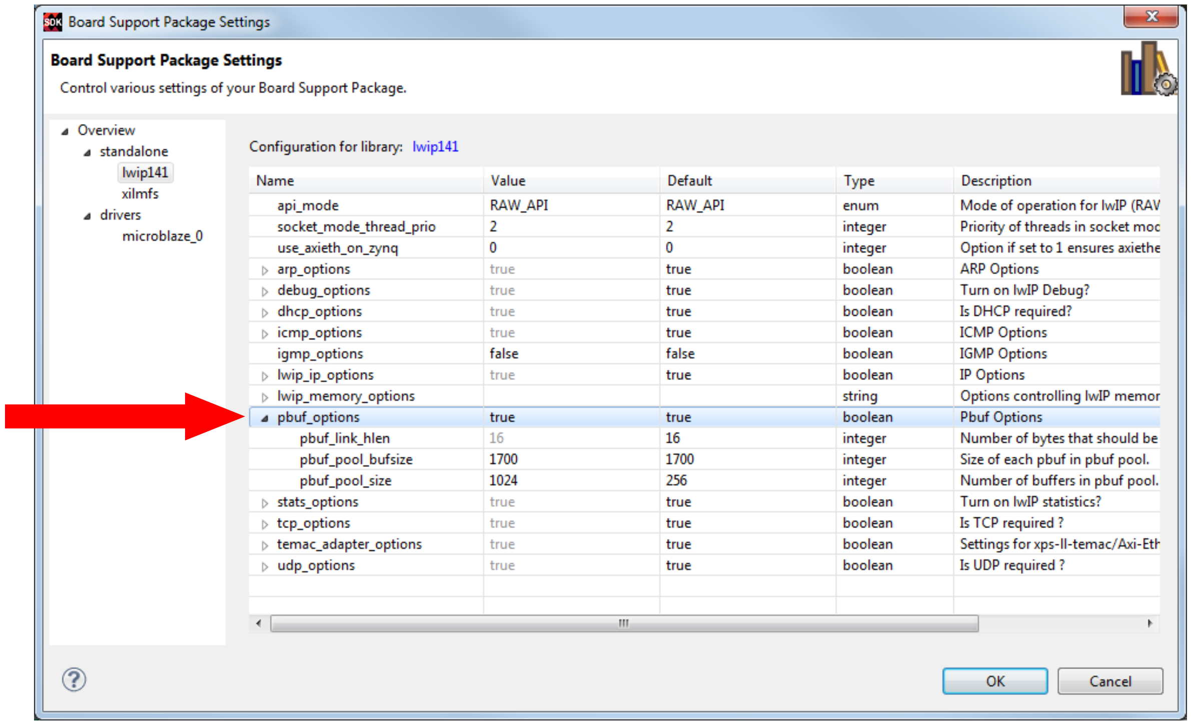
Name	Value	Default	Type	Description
use_axieth_on_zynq	0	0	integer	Option if set to 1 ensures axie
▷ arp_options	true	true	boolean	ARP Options
▷ debug_options	true	true	boolean	Turn on lwIP Debug?
▷ dhcp_options	true	true	boolean	Is DHCP required?
▷ icmp_options	true	true	boolean	ICMP Options
igmp_options	false	false	boolean	IGMP Options
▷ lwip_ip_options	true	true	boolean	IP Options
◢ lwip_memory_options			string	Options controlling lwIP mer
mem_size	131072	131072	integer	Size of the heap memory (byt
memp_n_pbuf	1024	16	integer	Number of memp struct pbu
memp_n_sys_timeout	8	8	integer	Number of simultaneously ac
memp_n_tcp_pcb	32	32	integer	Number of active TCP PCBs.
memp_n_tcp_pcb_listen	8	8	integer	Number of listening TCP con
memp_n_tcp_seg	1024	256	integer	Number of simultaneously qu
memp_n_udp_pcb	4	4	integer	Number of active UDP PCBs.
memp_num_api_msg	16	16	integer	Number of api msg structure
memp_num_netbuf	8	8	integer	Number of struct netbufs (so
memp_num_netconn	16	16	integer	Number of struct netconns (s
memp_num_tcpip_msg	64	64	integer	Number of tcpip msg structu
▷ pbuf_options	true	true	boolean	Pbuf Options

6. Our webserver file system is essentially a RAMdisk because it resides in system DDR3 memory and serves a few files while also allowing us to monitor the status of board GPIO. To maximize this file serving performance we need to set the number of **memp_n_pbuf** and **memp_n_tcp_seg** to **1024**. Notice the value of **mem_size**. This is equivalent to the amount of heap space that we specified in the linker script. **Do not** click **OK** yet.



Name	Value	Default
igmp_options	false	false
lwip_ip_options	true	true
lwip_memory_options		
mem_size	131072	131072
memp_n_pbuf	1024	16
memp_n_sys_timeout	8	8
memp_n_tcp_pcb	32	32
memp_n_tcp_pcb_listen	8	8
memp_n_tcp_seg	1024	256
memp_n_udp_pcb	4	4
memp_num_api_msg	16	16
memp_num_netbuf	8	8
memp_num_netconn	16	16
memp_num_tcpip_msg	64	64

7. Click to expand the tree for the **pbuf_options:**



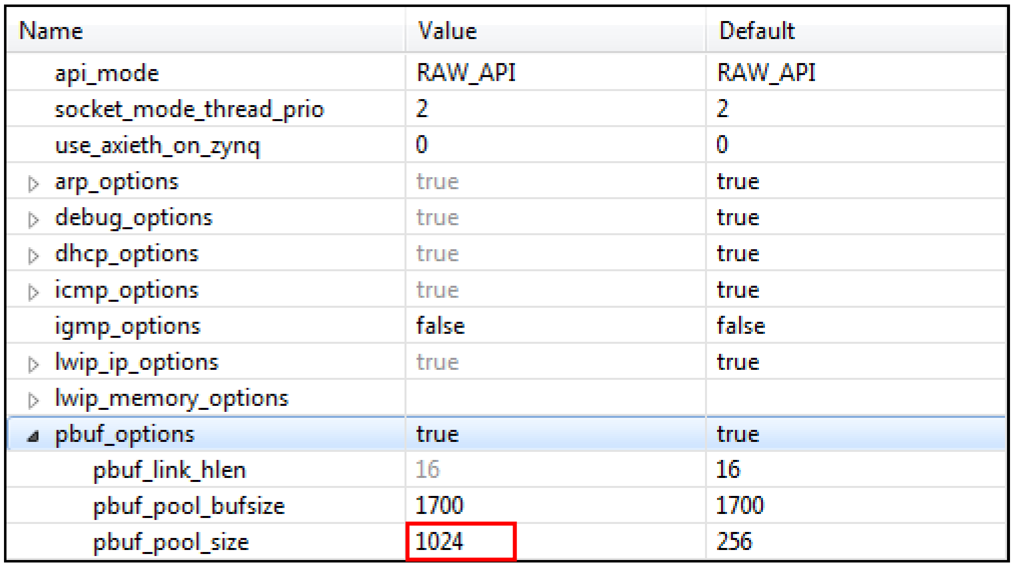
Board Support Package Settings

Control various settings of your Board Support Package.

Configuration for library: lwip141

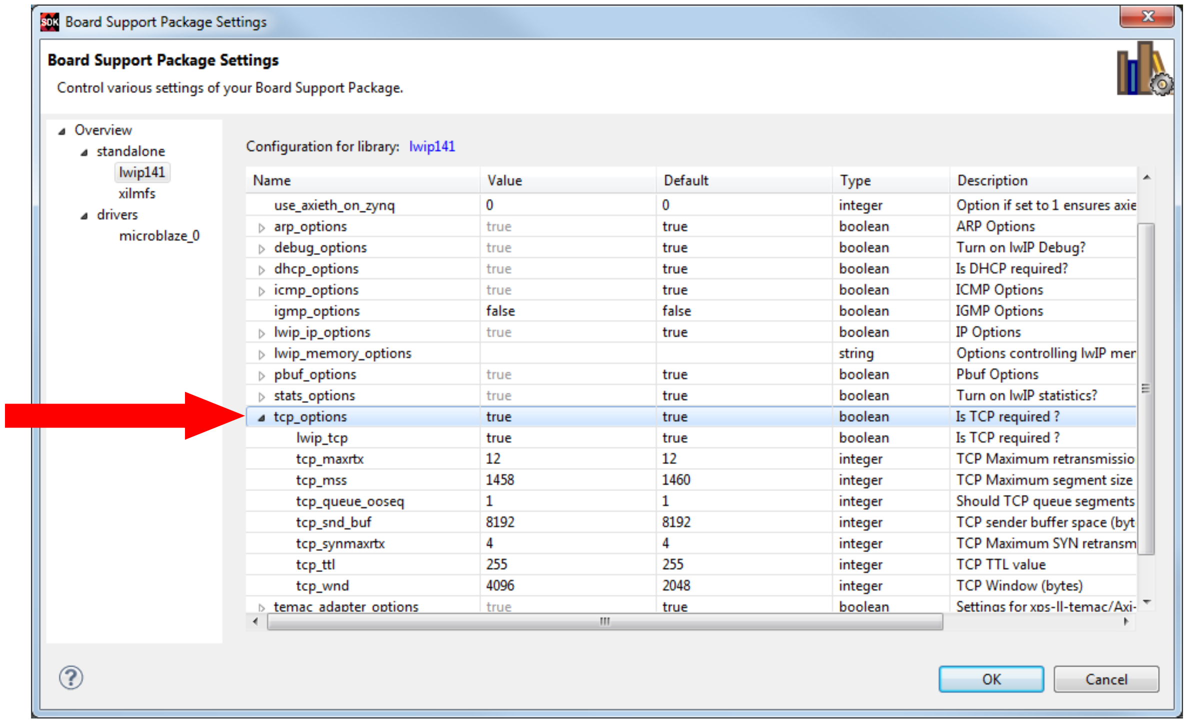
Name	Value	Default	Type	Description
api_mode	RAW_API	RAW_API	enum	Mode of operation for lwIP (RAV
socket_mode_thread_prio	2	2	integer	Priority of threads in socket moc
use_axieth_on_zynq	0	0	integer	Option if set to 1 ensures axiethe
arp_options	true	true	boolean	ARP Options
debug_options	true	true	boolean	Turn on lwIP Debug?
dhcp_options	true	true	boolean	Is DHCP required?
icmp_options	true	true	boolean	ICMP Options
igmp_options	false	false	boolean	IGMP Options
lwip_ip_options	true	true	boolean	IP Options
lwip_memory_options			string	Options controlling lwIP memor
pbuf_options	true	true	boolean	Pbuf Options
pbuf_link_hlen	16	16	integer	Number of bytes that should be
pbuf_pool_bufsize	1700	1700	integer	Size of each pbuf in pbuf pool.
pbuf_pool_size	1024	256	integer	Number of buffers in pbuf pool.
stats_options	true	true	boolean	Turn on lwIP statistics?
tcp_options	true	true	boolean	Is TCP required ?
temac_adapter_options	true	true	boolean	Settings for xps-ll-temac/Axi-Eth
udp_options	true	true	boolean	Is UDP required ?

8. Packet buffers (Pbufs) carry packets across various layers of the TCP/IP stack. To increase our Ethernet performance we need to increase the **pbuf_pool_size** to **1024**. **Do not** click **OK** yet.



Name	Value	Default
api_mode	RAW_API	RAW_API
socket_mode_thread_prio	2	2
use_axieth_on_zynq	0	0
▷ arp_options	true	true
▷ debug_options	true	true
▷ dhcp_options	true	true
▷ icmp_options	true	true
igmp_options	false	false
▷ lwip_ip_options	true	true
▷ lwip_memory_options		
◢ pbuf_options	true	true
pbuf_link_hlen	16	16
pbuf_pool_bufsize	1700	1700
pbuf_pool_size	1024	256

9. Click to expand the tree for the **tcp_options** settings:



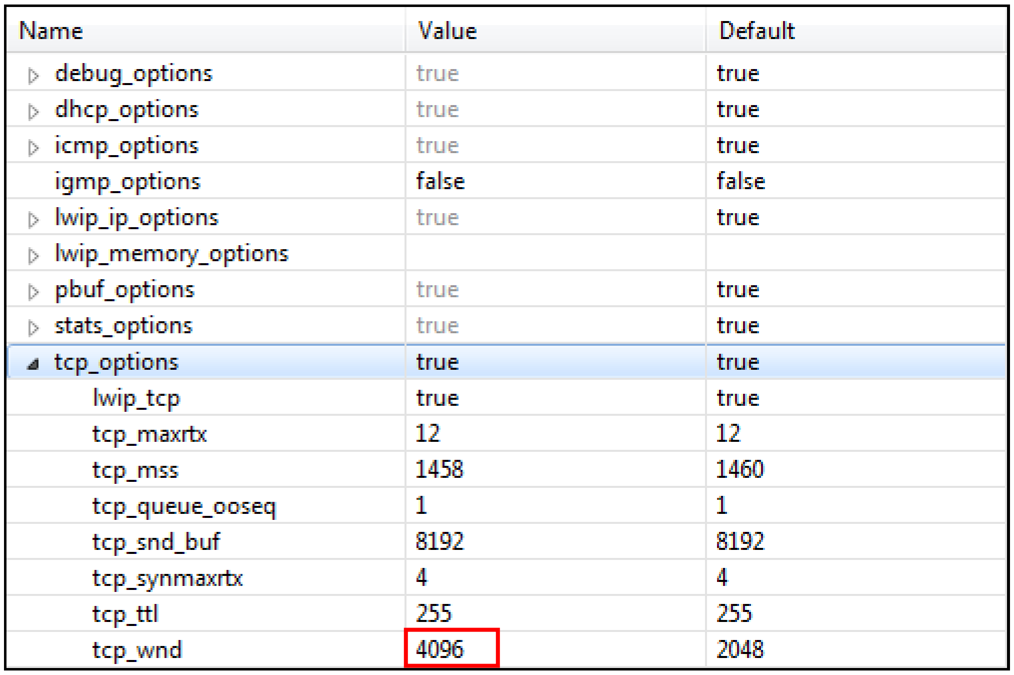
Board Support Package Settings

Control various settings of your Board Support Package.

Configuration for library: lwip141

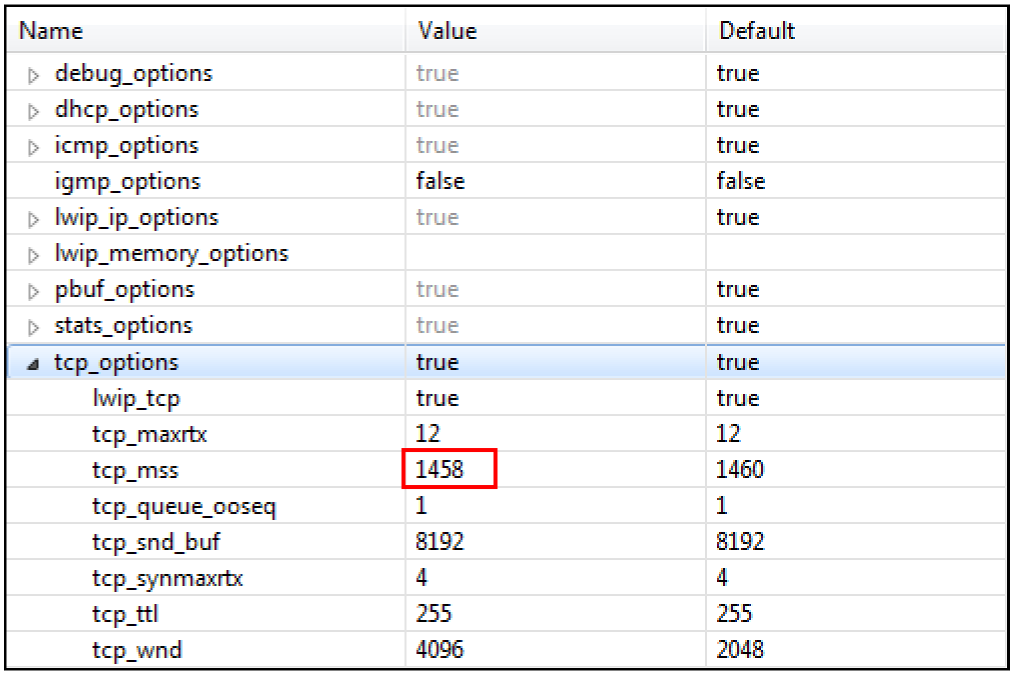
Name	Value	Default	Type	Description
use_axieth_on_zynq	0	0	integer	Option if set to 1 ensures axie
▷ arp_options	true	true	boolean	ARP Options
▷ debug_options	true	true	boolean	Turn on lwIP Debug?
▷ dhcp_options	true	true	boolean	Is DHCP required?
▷ icmp_options	true	true	boolean	ICMP Options
igmp_options	false	false	boolean	IGMP Options
▷ lwip_ip_options	true	true	boolean	IP Options
▷ lwip_memory_options			string	Options controlling lwIP mer
▷ pbuf_options	true	true	boolean	Pbuf Options
▷ stats_options	true	true	boolean	Turn on lwIP statistics?
◢ tcp_options	true	true	boolean	Is TCP required ?
lwip_tcp	true	true	boolean	Is TCP required ?
tcp_maxrtx	12	12	integer	TCP Maximum retransmissio
tcp_mss	1458	1460	integer	TCP Maximum segment size
tcp_queue_ooseq	1	1	integer	Should TCP queue segments
tcp_snd_buf	8192	8192	integer	TCP sender buffer space (byt
tcp_synmaxrtx	4	4	integer	TCP Maximum SYN retransm
tcp_ttl	255	255	integer	TCP TTL value
tcp_wnd	4096	2048	integer	TCP Window (bytes)
▷ temac_adapter_options	true	true	boolean	Settings for xps-ll-temac/Axi-

10. To maximize the use of the hardware Rx ping-pong buffers it is necessary to increase the size of the TCP window. A standard Ethernet frame is 1518 bytes long, so each buffer can hold a single frame. By default, the receive window is set at 2048 bytes, meaning only one frame can be in transit before an acknowledgement is sent to the host. We can double the size of **tcp_wnd** to **4096** bytes to allow a second frame in transit, which will better utilize the available bandwidth. Experiment with this parameter to see how higher and lower values affect the throughput on your system. **Do not** click **OK** yet.



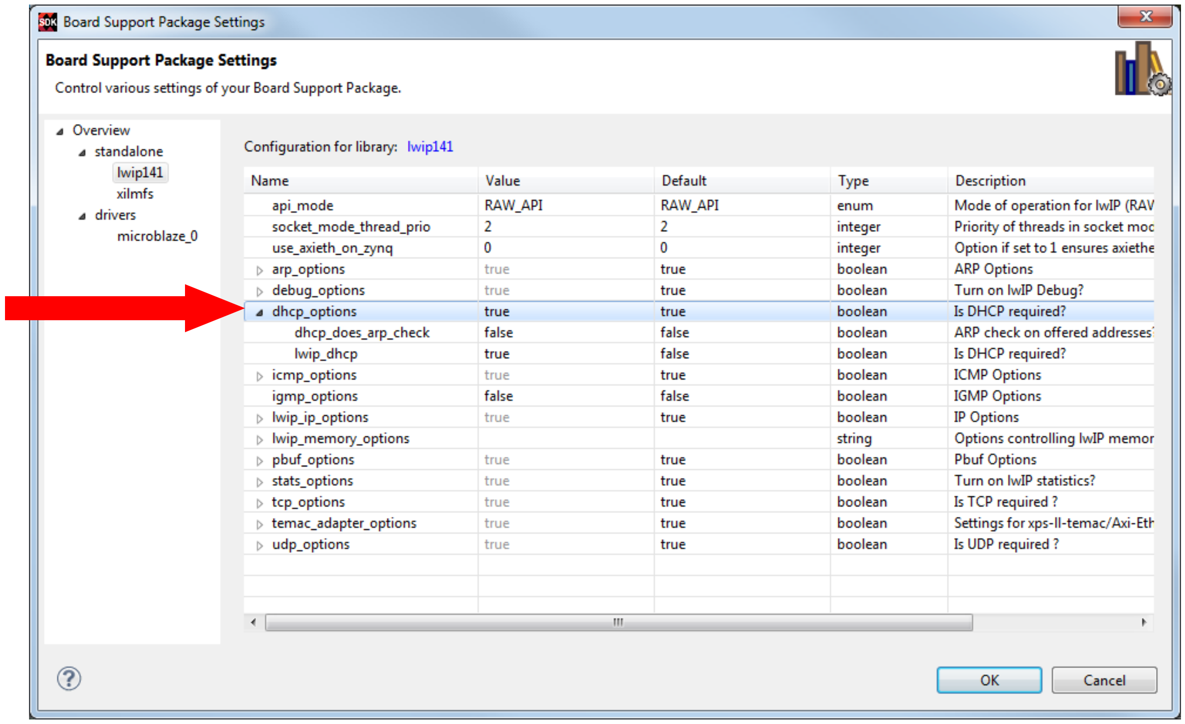
Name	Value	Default
▷ debug_options	true	true
▷ dhcp_options	true	true
▷ icmp_options	true	true
igmp_options	false	false
▷ lwip_ip_options	true	true
▷ lwip_memory_options		
▷ pbuf_options	true	true
▷ stats_options	true	true
◢ tcp_options	true	true
lwip_tcp	true	true
tcp_maxrtx	12	12
tcp_mss	1458	1460
tcp_queue_ooseq	1	1
tcp_snd_buf	8192	8192
tcp_synmaxrtx	4	4
tcp_ttl	255	255
tcp_wnd	4096	2048

11. To maximize the Tx throughput it is necessary to make the maximum segment size slightly smaller than the default. The maximum segment size controls the amount of payload bytes per IP packet. Ideally this value is as large as possible to maximize the transmit efficiency. For this MicroBlaze system connected over Ethernet to a Windows host computer, the best transmit throughput was achieved with a **tcp_mss** value of **1458** bytes. Experiment with this parameter to see how higher and lower values affect the throughput on your system. **Do not** click **OK** yet.



Name	Value	Default
▷ debug_options	true	true
▷ dhcp_options	true	true
▷ icmp_options	true	true
igmp_options	false	false
▷ lwip_ip_options	true	true
▷ lwip_memory_options		
▷ pbuf_options	true	true
▷ stats_options	true	true
◢ tcp_options	true	true
lwip_tcp	true	true
tcp_maxrtx	12	12
tcp_mss	1458	1460
tcp_queue_ooseq	1	1
tcp_snd_buf	8192	8192
tcp_synmaxrtx	4	4
tcp_ttl	255	255
tcp_wnd	4096	2048

12. Click to expand the tree for the **dhcp_options** settings:



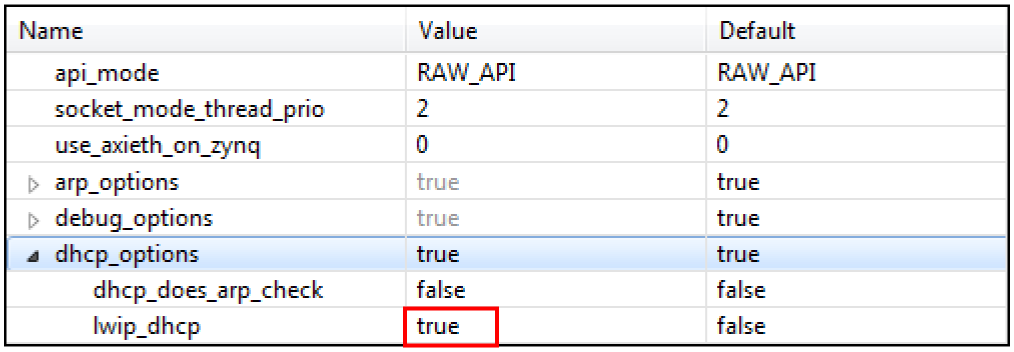
Board Support Package Settings

Control various settings of your Board Support Package.

Configuration for library: lwip141

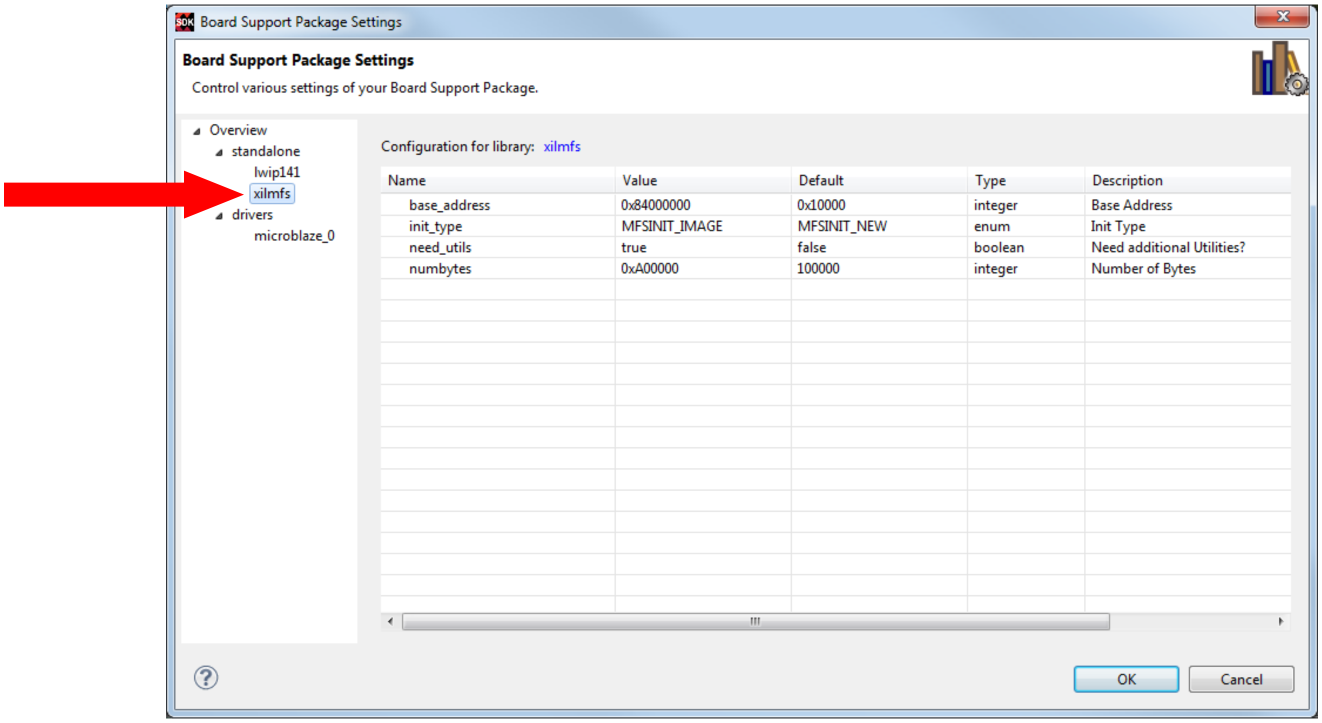
Name	Value	Default	Type	Description
api_mode	RAW_API	RAW_API	enum	Mode of operation for lwIP (RAV
socket_mode_thread_prio	2	2	integer	Priority of threads in socket moc
use_axieth_on_zynq	0	0	integer	Option if set to 1 ensures axiethe
arp_options	true	true	boolean	ARP Options
debug_options	true	true	boolean	Turn on lwIP Debug?
dhcp_options	true	true	boolean	Is DHCP required?
dhcp_does_arp_check	false	false	boolean	ARP check on offered addresses
lwip_dhcp	true	false	boolean	Is DHCP required?
icmp_options	true	true	boolean	ICMP Options
igmp_options	false	false	boolean	IGMP Options
lwip_ip_options	true	true	boolean	IP Options
lwip_memory_options			string	Options controlling lwIP memor
pbuf_options	true	true	boolean	Pbuf Options
stats_options	true	true	boolean	Turn on lwIP statistics?
tcp_options	true	true	boolean	Is TCP required ?
temac_adapter_options	true	true	boolean	Settings for xps-ll-temac/Axi-Eth
udp_options	true	true	boolean	Is UDP required ?

13. Even though this example design prescribes connecting the Arty board directly to the host PC with a static IP address, the lwIP stack is capable of fetching its IP address from a DHCP server (probably an Ethernet router) on the network. To enable this capability set **lwip_dhcp** to **true**. If a DHCP server is not found when the lwIP stack initializes at power on, it will default to an IP address of 192.168.1.10. This is the behavior seen in the demos described earlier in this tutorial. **Do not** click **OK** yet.



Name	Value	Default
api_mode	RAW_API	RAW_API
socket_mode_thread_prio	2	2
use_axieth_on_zynq	0	0
arp_options	true	true
debug_options	true	true
dhcp_options	true	true
dhcp_does_arp_check	false	false
lwip_dhcp	true	false

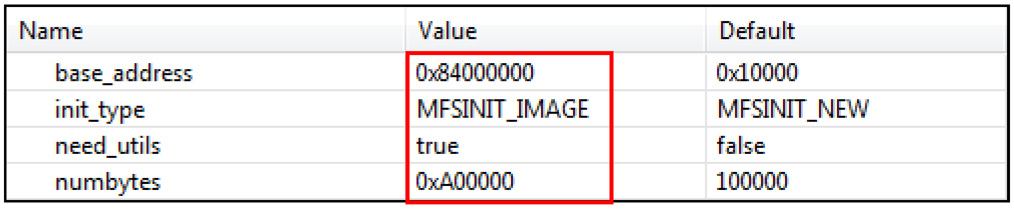
14. In the left panel, select **xilmfs** for the standalone OS to see the detailed parameters available for the xilmfs library.


Board Support Package Settings
Board Support Package Settings
Control various settings of your Board Support Package.
Overview
standalone
lwip141
xilmfs
drivers
microblaze_0
OK
Cancel

Configuration for library: xilmfs

Name	Value	Default	Type	Description
base_address	0x84000000	0x10000	integer	Base Address
init_type	MFSINIT_IMAGE	MFSINIT_NEW	enum	Init Type
need_utils	true	false	boolean	Need additional Utilities?
numbytes	0xA00000	100000	integer	Number of Bytes

15. In this window specify the following settings:
    - Set **numbytes** to **`0xA00000 (10485760)`**
    - Set **base_address** to **`0x84000000`.** This is the base address of the system RAM plus an offset above and beyond where the application code is located.
    - Set **init_type** to **MFSINIT_IMAGE**
    - Set **need_utils** to **true**



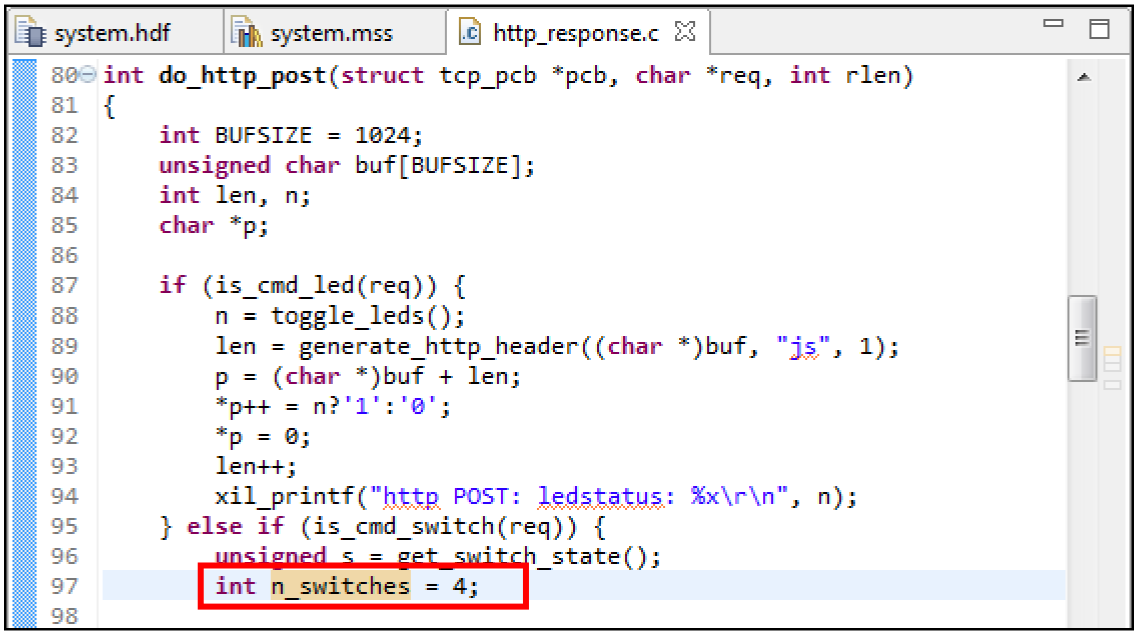
Name	Value	Default
base_address	0x84000000	0x10000
init_type	MFSINIT_IMAGE	MFSINIT_NEW
need_utils	true	false
numbytes	0xA00000	100000

Click **OK** to save the changes, and the software platform will automatically compile and link.

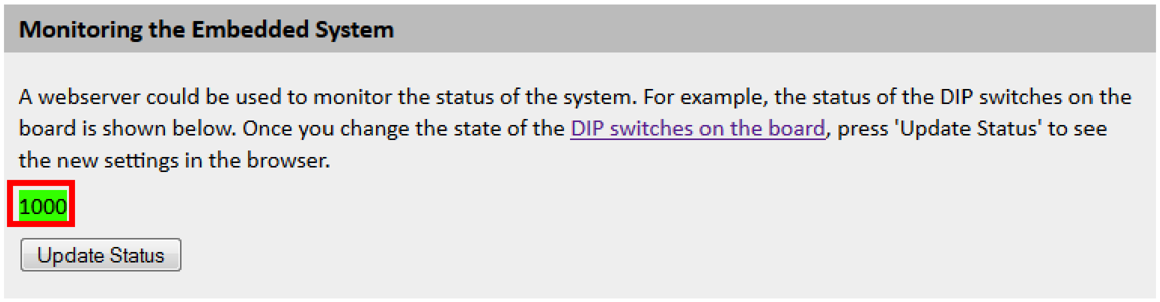
Description of lwIP Source Code Edits

The Xilinx source code for the lwIP applications does not match the hardware on the Arty board, and prints status messages to the serial console that diminish the performance of the webserver. Simple edits were made to the following files:

http_response.c: A small edit is made to the **http_response.c** source file to change the value of the **n_switches** variable to 4 to match the number of the DIP switches on the Arty board. This change is made so the webserver will report the correct number of switches when the Update Status button is pressed in the web browser.



```
int do_http_post(struct tcp_pcb *pcb, char *req, int rlen)
{
    int BUFSIZE = 1024;
    unsigned char buf[BUFSIZE];
    int len, n;
    char *p;

    if (is_cmd_led(req)) {
        n = toggle_leds();
        len = generate_http_header((char *)buf, "js", 1);
        p = (char *)buf + len;
        *p++ = n?'1':'0';
        *p = 0;
        len++;
        xil_printf("http POST: ledstatus: %x\r\n", n);
    } else if (is_cmd_switch(req)) {
        unsigned s = get_switch_state();
        int n_switches = 4;
```

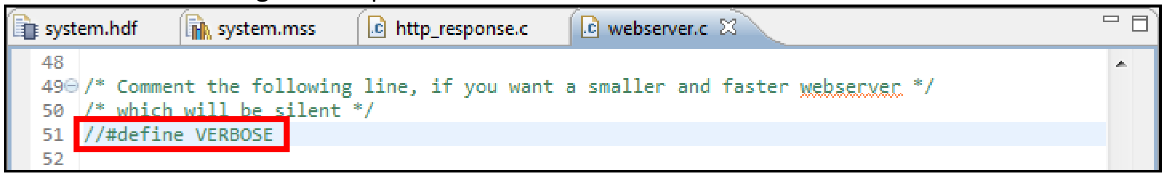


Monitoring the Embedded System

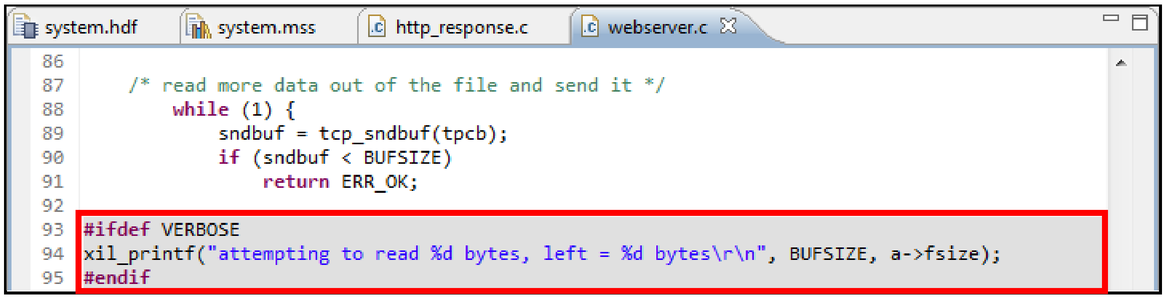
A webserver could be used to monitor the status of the system. For example, the status of the DIP switches on the board is shown below. Once you change the state of the DIP switches on the board, press 'Update Status' to see the new settings in the browser.

1000

webserver.c: Every time a link is clicked in the web browser to display the picture of the Arty board or display the Xilinx XAPP1026 documentation a status message is printed to the serial console displaying a streaming status of how many bytes of the file have been fetched and how many more are needed before the file will be displayed. This serial communication at 9600 baud or even 115200 baud to display these messages in the serial console is much slower than the transmission of TCP/IP packets and significantly slows down the performance of the HTTP server. A couple of small edits were made to this file to allow you to choose to display these messages or not. Remove the comment "//" on line 51 to allow these status messages to be printed to the serial console:

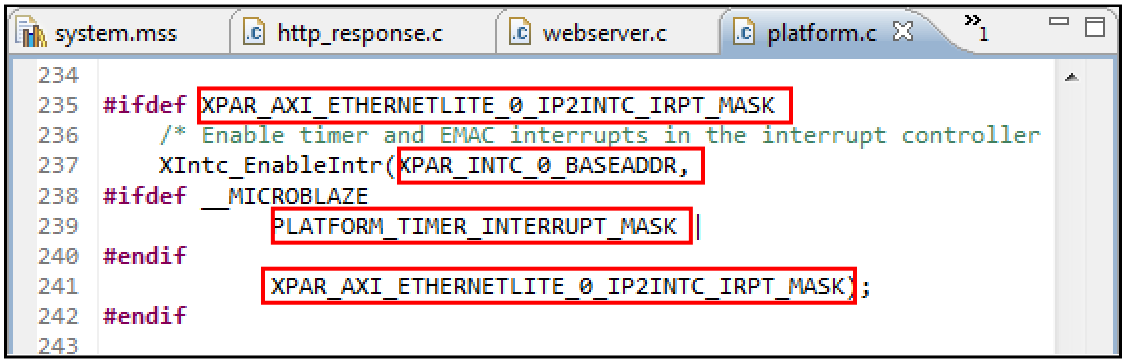


```
48
49 /* Comment the following line, if you want a smaller and faster webserver */
50 /* which will be silent */
51 //#define VERBOSE
52
```



```

    /* read more data out of the file and send it */
        while (1) {
            sndbuf = tcp_sndbuf(tpcb);
            if (sndbuf < BUFSIZE)
                return ERR_OK;

#ifdef VERBOSE
xil_printf("attempting to read %d bytes, left = %d bytes\r\n", BUFSIZE, a->fsize);
#endif
```

platform.c: Because these lwIP applications were designed to be run on many different FPGA and Zynq development boards, and because the Xilinx SDK and Vivado tools are not always consistent in the naming of peripheral instances, some IP parameter definitions and declarations in the source code are often incorrect. One block of code that often needs changing when targeting the lwIP applications to a new hardware platform is the enabling of the timer and EMAC interrupts in the interrupt controller. It is always a good idea to check that the values for the EMAC IP2INTC interrupt mask (for the Arty board this is the AXI EthernetLite, XPAR_AXI_ETHERNETLITE_0_IP2INTC_IRPT_MASK), the base address for the interrupt controller (XPAR_INTC_0_BASEADDR), and the PLATFORM_TIMER_INTERRUPT_MASK are known to the software and defined in the xparameters.h file. You may need to change any or all of these definitions to match your hardware.



```

#ifdef XPAR_AXI_ETHERNETLITE_0_IP2INTC_IRPT_MASK
    /* Enable timer and EMAC interrupts in the interrupt controller
    XIntc_EnableIntr(XPAR_INTC_0_BASEADDR,
#ifdef __MICROBLAZE
                    PLATFORM_TIMER_INTERRUPT_MASK |
#endif
                   XPAR_AXI_ETHERNETLITE_0_IP2INTC_IRPT_MASK);
#endif

```

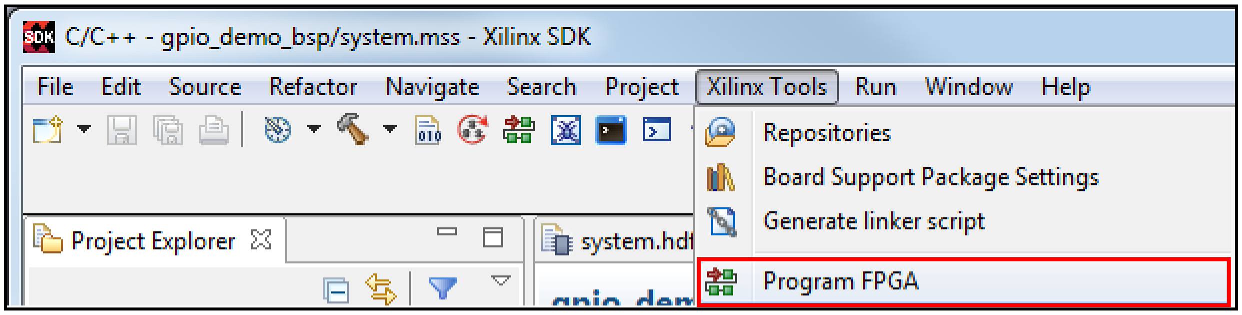
Running the Software Applications

To prepare for running the **lwip_raw_apps** software application later we will configure the FPGA now with our hardware bitstream that includes a very small **bootloop** software application that holds the MicroBlaze processor in a known-good ready-to-use state.

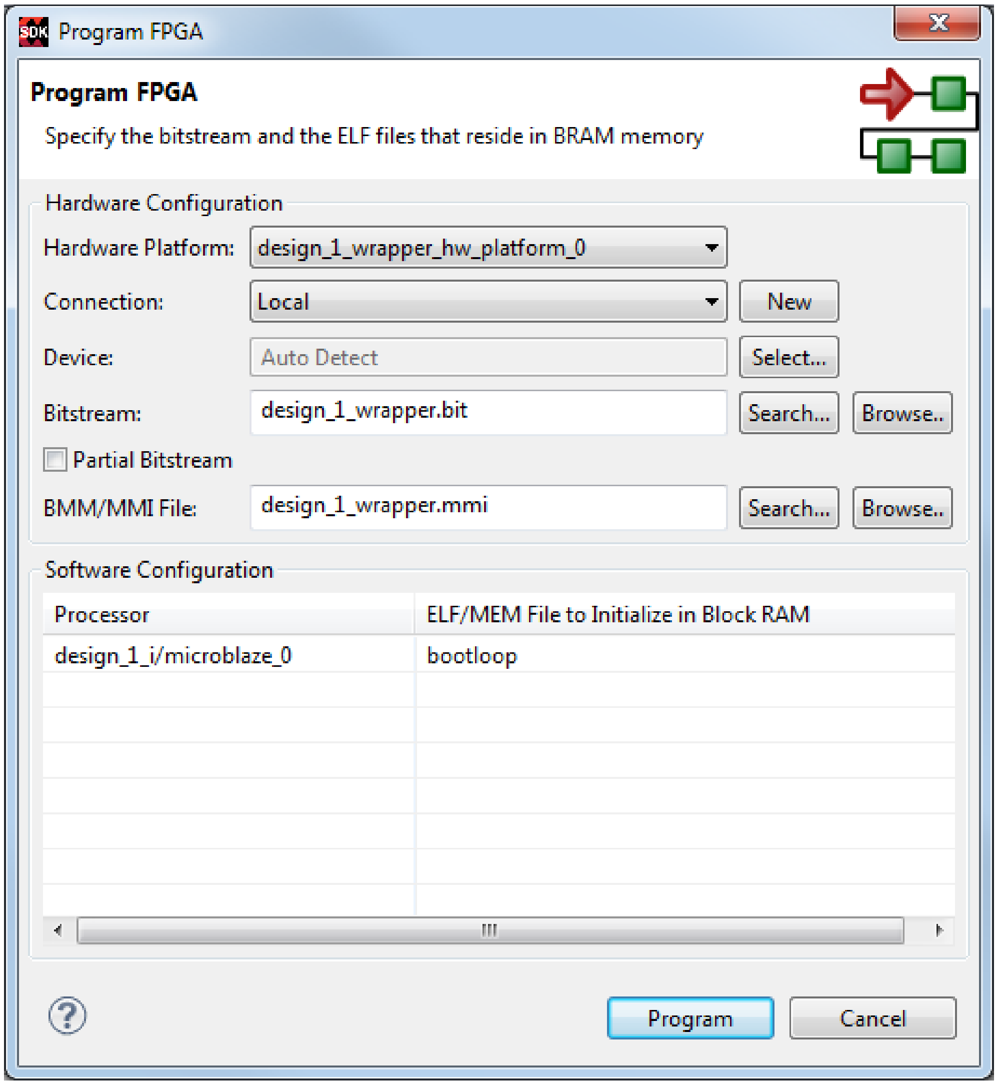
1. If not already done, connect the Arty board to the host PC as described earlier in Setting Up the Arty Evaluation Board.

2. Start a serial terminal session using your terminal software of choice and set the serial port parameters to **115200** baud rate, **no** parity, **8** bits, **1** stop bit and no flow control.

3. In the SDK main menu, select **Xilinx Tools → Program FPGA** or click on the on the SDK toolbar:



4. Accept the defaults for the **Hardware Platform** and **Connection** and verify that the **bootloop** application is selected as the **ELF/MEM File to Initialize in Block RAM**. Click **Program** to continue:



Program FPGA

Program FPGA

Specify the bitstream and the ELF files that reside in BRAM memory

Hardware Configuration

Hardware Platform: design_1_wrapper_hw_platform_0

Connection: Local

Device: Auto Detect

Bitstream: design_1_wrapper.bit

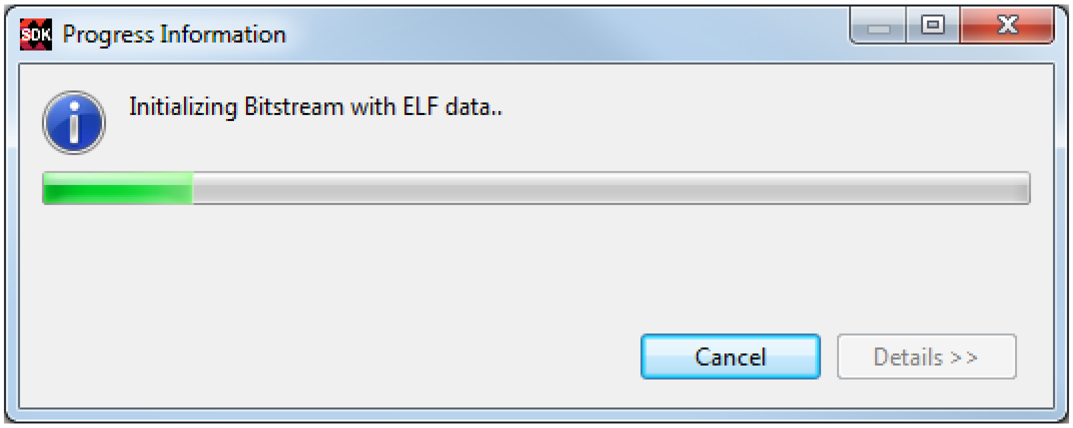
Partial Bitstream

BMM/MMI File: design_1_wrapper.mmi

Software Configuration

Processor	ELF/MEM File to Initialize in Block RAM
design_1_i/microblaze_0	bootloop

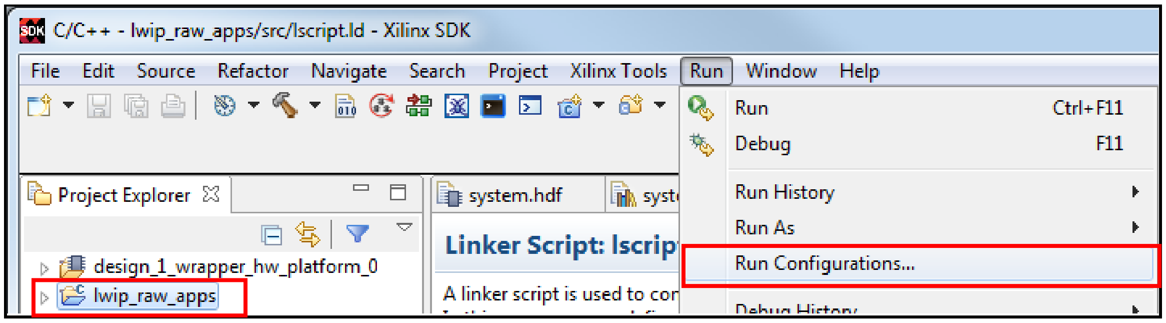
5. You will see the following window while the FPGA bitstream is downloaded:



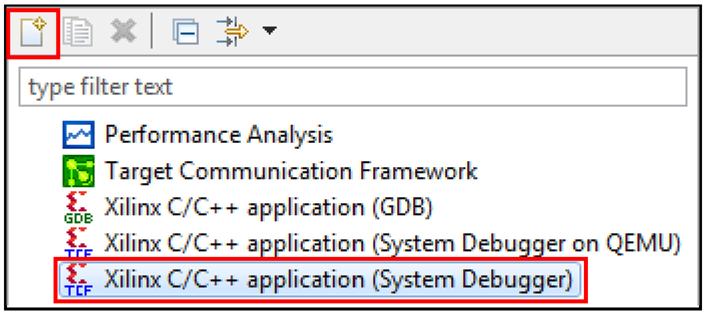
Progress Information

Initializing Bitstream with ELF data..

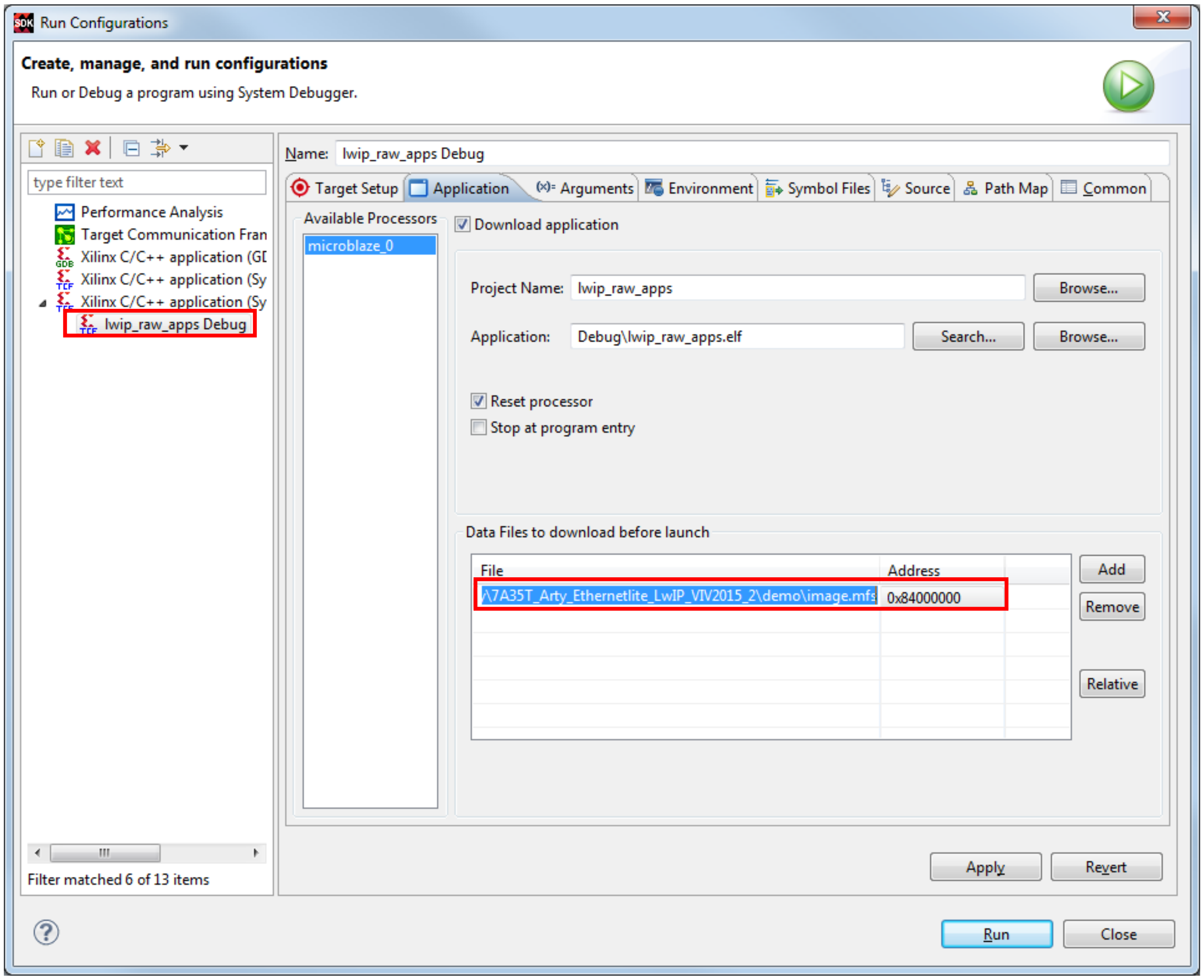
6. One of the lwIP applications is a small webserver that requires a filesystem to be resident in system memory. Before we can use the webserver we must first setup the Xilinx System Debugger (TCF) options to automatically download this filesystem to the board when we download and run the **lwip_raw_apps** software application. Click on the **lwip_raw_apps** application in the **Project Explorer** pane and in the SDK main menu, select **Run → Run Configurations…**


C/C++ - lwip_raw_apps/src/lscript.ld - Xilinx SDK
File Edit Source Refactor Navigate Search Project Xilinx Tools Run Window Help
Project Explorer
design_1_wrapper_hw_platform_0
lwip_raw_apps
system.hdf
Linker Script: lscrip
A linker script is used to con
Run Ctrl+F11
Debug F11
Run History
Run As
Run Configurations...
Debug History

7. Click on the Xilinx C/C++ application (System Debugger) option in the left panel and press the **New** to create a new Debug run configuration for the **lwip_raw_apps** software application.


type filter text
Performance Analysis
Target Communication Framework
Xilinx C/C++ application (GDB)
Xilinx C/C++ application (System Debugger on QEMU)
Xilinx C/C++ application (System Debugger)

8. Select the new **lwip_raw_apps Debug** run configuration. Click on the **Application** tab to specify the path to the webserver filesystem and the memory location where it is to be stored on the Arty board. Click on the Add icon and navigate to the **<installation>\demo** folder then double-click the **image.mfs** file to add it to the run configuration. Note that this memory address (**`0x84000000`**) must be the same as was specified earlier in the configuration of the BSP. This run configuration will download the filesystem along with the application executable. This saves us an extra step that we would otherwise have to do manually. Click the **Run** button to continue.


Run Configurations

Create, manage, and run configurations

Run or Debug a program using System Debugger.

type filter text

- Performance Analysis
- Target Communication Fran
- Xilinx C/C++ application (GI
- Xilinx C/C++ application (Sy
- Xilinx C/C++ application (Sy
  - lwip_raw_apps Debug

Filter matched 6 of 13 items

Name: lwip_raw_apps Debug

Target Setup | Application | Arguments | Environment | Symbol Files | Source | Path Map | Common

Available Processors: microblaze_0

☑ Download application

Project Name: lwip_raw_apps

Application: Debug\lwip_raw_apps.elf

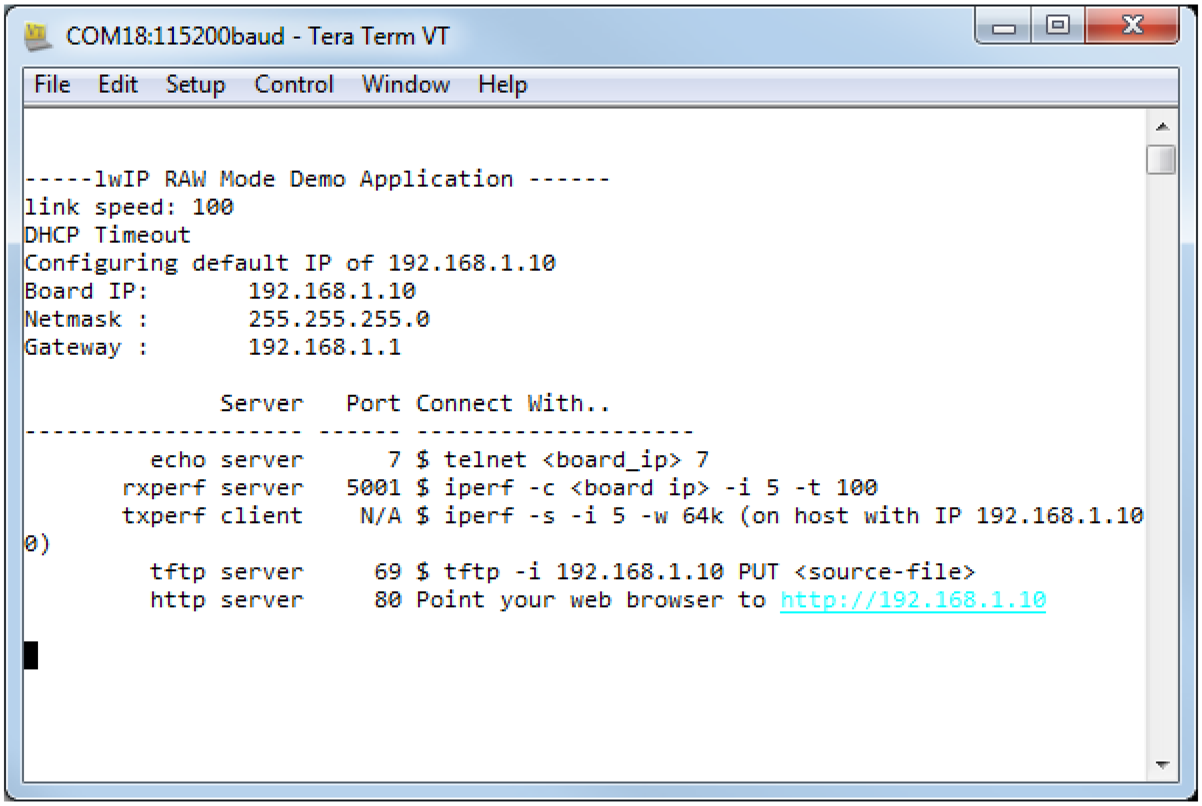
☑ Reset processor

☐ Stop at program entry

Data Files to download before launch

File	Address
\7A35T_Arty_Ethernetlite_LwIP_VIV2015_2\demo\image.mfs	0x84000000

9. You should see the following on the serial console:


COM18:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
-----lwIP RAW Mode Demo Application ------
link speed: 100
DHCP Timeout
Configuring default IP of 192.168.1.10
Board IP:       192.168.1.10
Netmask :       255.255.255.0
Gateway :       192.168.1.1

              Server   Port Connect With..
-------------------- ------ --------------------
         echo server      7 $ telnet <board_ip> 7
       rxperf server   5001 $ iperf -c <board ip> -i 5 -t 100
       txperf client    N/A $ iperf -s -i 5 -w 64k (on host with IP 192.168.1.10
0)
         tftp server     69 $ tftp -i 192.168.1.10 PUT <source-file>
         http server     80 Point your web browser to http://192.168.1.10
```

10. You are now ready to run the lwIP software applications. The steps to run the applications are the same as running the demos you probably used earlier, except the steps of downloading the bitstream, application executable and filesystem are already completed. Open a command window in the **<installation>\demo** folder and run the **cp_from_sdk.bat** batch file script to copy the new bitstream and software ELF files to the demo folder. Go back and rerun the demos, starting with the RxTest Demo. This concludes this design tutorial.

Modify the Webserver Filesystem

After using the HTTP server demo you may want to experiment with creating your own filesystem for the webserver. Feel free to edit the **<installation>\memfs\index.html** to change the existing text and links or to add your own custom text and links to new files. If you are also familiar with JavaScript you can edit the provided JavaScript files to add new functionality or modify the existing methods to change the actions performed by the webserver. When you are done making your changes to the filesystem be sure to open a command window in the **<installation>\memfs** folder and at the command prompt enter

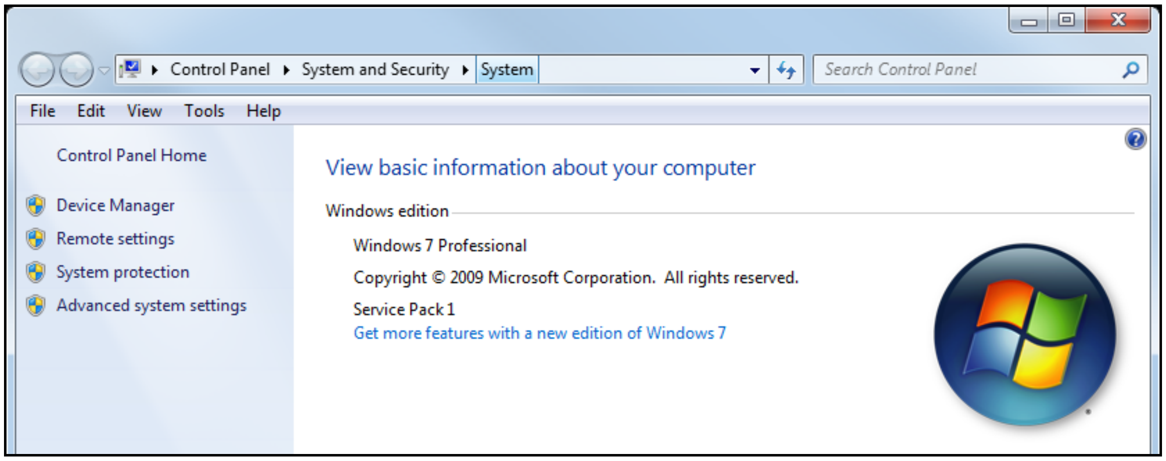
```
create_mfs.bat
```

to run the batch file to create a new Xilinx Memory Filesystem with your changes and copy the new filesystem image to the **<installation>\demo** folder. You can see the effects of your changes by running the **demo_raw_apps.bat** batch file as described in the Applications Download section earlier in this document. If you want to preserve the original filesystem be sure to copy it to a new filename first.

Appendix I: Determining the Virtual COM Port

Now you can connect the evaluation board's USB-to-UART port to one of the USB ports on your PC. The new hardware detection will pop up and enumeration of the driver will be started. Once finished a virtual COMx port is created and you are ready to setup a connection using Windows HyperTerminal or comparable serial terminal emulation utility. Follow these instructions to determine the COMx port assigned to the USB-to-UART bridge:

1. Open the Device Manager by right-clicking on Computer, select **Properties**, then click on the **Device Manager**.



View basic information about your computer

Windows edition

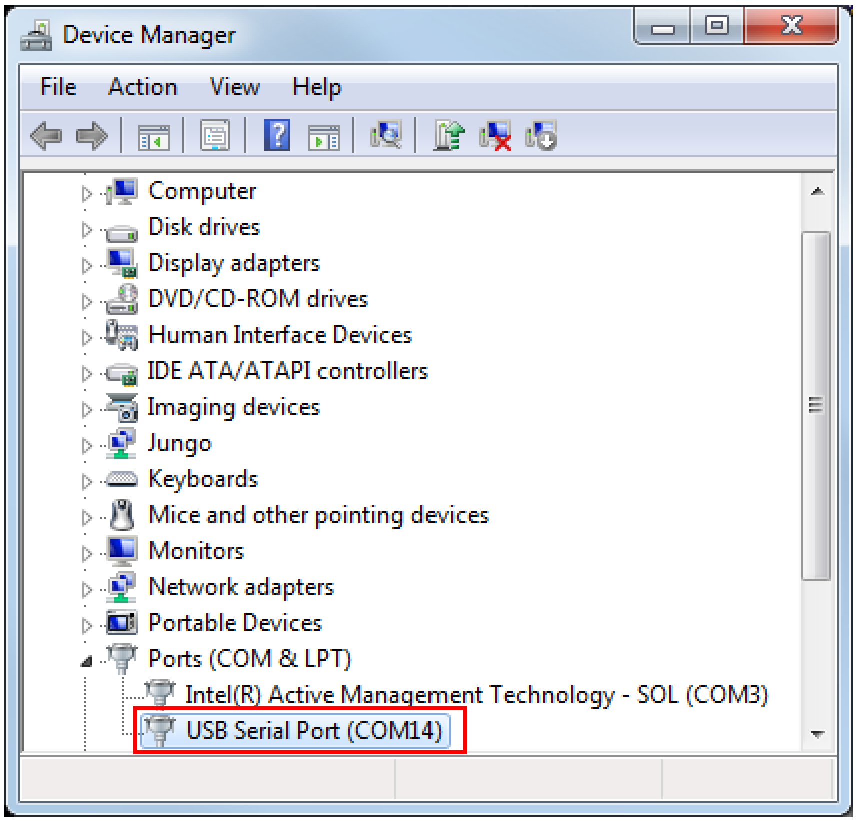
Windows 7 Professional

Copyright © 2009 Microsoft Corporation. All rights reserved.

Service Pack 1

Get more features with a new edition of Windows 7

2. In the Device Manager, scroll down to Ports and expand the list. You will see the USB Serial Port and its assigned COM port. In the example below, it is COM14. Make note of this COM port number for use with the serial terminal you will use elsewhere in this design tutorial. This concludes these USB UART driver and virtual COM port installation instructions.

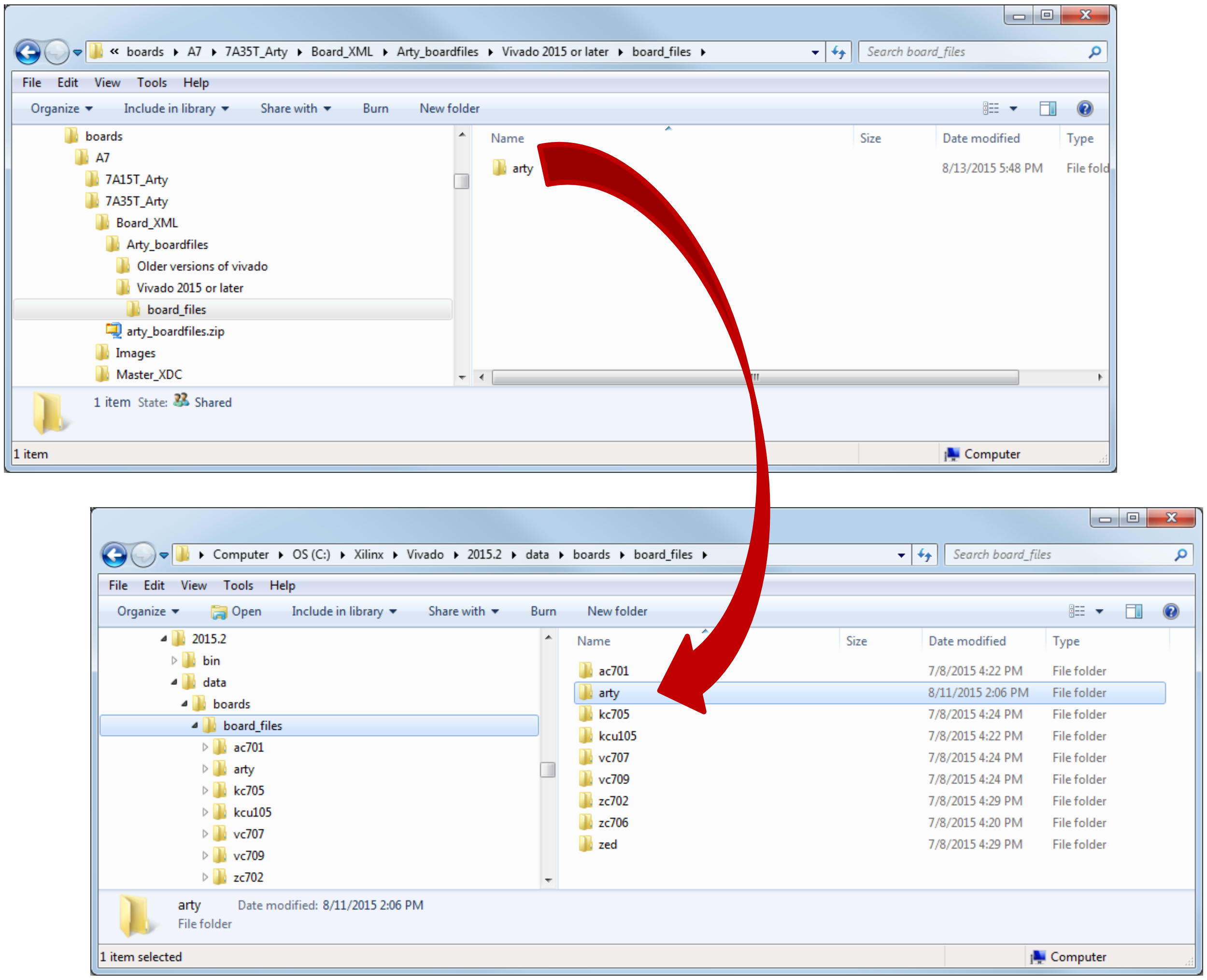


Appendix II: Installation of Board XML Files

Many Avnet evaluation boards can now be targeted directly in the Xilinx Vivado tools. This greatly simplifies the task of building the MicroBlaze processor system and integrating system peripherals and board interfaces into your own custom designs. Follow the instructions below to install the board XML files for the Arty board into your Vivado installation folders:

1. Using your favorite web browser, download the zip archive of board XML files from the Digilent wiki site for the Arty Evaluation Board:
   https://reference.digilentinc.com/_media/arty_boardfiles.zip

2. Extract the zip file in the download folder and in Windows Explorer, copy the **arty** folder from the **<installation>\Arty_boardfiles\Vivado 2015 or later\board_files** folder to **<Vivado_install>\Xilinx\Vivado\2015.2\data\boards\board_files**:



« boards ▸ A7 ▸ 7A35T_Arty ▸ Board_XML ▸ Arty_boardfiles ▸ Vivado 2015 or later ▸ board_files ▸

Name	Size	Date modified	Type
arty		8/13/2015 5:48 PM	File fold

boards
A7
7A15T_Arty
7A35T_Arty
Board_XML
Arty_boardfiles
Older versions of vivado
Vivado 2015 or later
board_files
arty_boardfiles.zip
Images
Master_XDC

1 item State: Shared

▸ Computer ▸ OS (C:) ▸ Xilinx ▸ Vivado ▸ 2015.2 ▸ data ▸ boards ▸ board_files ▸

Name	Size	Date modified	Type
ac701		7/8/2015 4:22 PM	File folder
arty		8/11/2015 2:06 PM	File folder
kc705		7/8/2015 4:24 PM	File folder
kcu105		7/8/2015 4:22 PM	File folder
vc707		7/8/2015 4:24 PM	File folder
vc709		7/8/2015 4:24 PM	File folder
zc702		7/8/2015 4:29 PM	File folder
zc706		7/8/2015 4:20 PM	File folder
zed		7/8/2015 4:29 PM	File folder

2015.2
bin
data
boards
board_files
ac701
arty
kc705
kcu105
vc707
vc709
zc702

arty Date modified: 8/11/2015 2:06 PM
File folder

1 item selected

Appendix III: Windows 260 Character Path Limit

If you are using the Vivado Design Suite on a Windows-7 host, you may run into issues resulting from Vivado pathnames exceeding the maximum allowed. Vivado projects create a very deep file hierarchy, and it becomes very easy to violate the Windows limit if the project is not extracted near the root of the drive. This can even happen when the archive is decompressed, depending on where you choose to place the project in your existing file hierarchy.

It is not always convenient to place every Vivado project at the root of a drive. To work around this limitation, the recommended procedure is to place the Vivado archive in a shared folder on your host machine, then use Windows Explorer to map a network drive to the directory where the archive will be decompressed. This allows the Vivado project to be mapped to the root of the virtual (mapped) directory, eliminating any path issues.

Vivado also makes use of the Windows temp folder, which may be located several folders deep from the root drive, and this can also cause problems. You can create your own temporary directory in C:\temp, and force Vivado to use the new folder with the following TCL:

```
set_param “project.customTmpDirForArchive”  C:/temp
```

For further information, see the Xilinx answer record at:

http://www.xilinx.com/support/answers/52787.html.

Appendix IV: Getting Help and Support

Avnet Website

- Evaluation Kit home page with Documentation and Reference Designs
  http://em.avnet.com/arty

- Avnet support forums
  http://community.em.avnet.com/

Xilinx Website

- Xilinx Application Note 1026 – LwIP Applications
  http://www.xilinx.com/support/documentation/application_notes/xapp1026.pdf

- Details on the Artix-7 FPGA family are included in the following Xilinx documents:
  - *Artix-7 Family Overview (DS180)*
  - *Artix-7 FPGA Data Sheet (DS181)*
  - *Artix-7 FPGA Configuration User Guide (UG470)*

- For more detailed information about lwIP and XilMFS, please refer to the Xilinx OS and Libraries Document Collection located here:
  www.xilinx.com/support/documentation/sw_manuals/xilinx2015_2/oslib_rm.pdf

- For more detailed information about SDK, please refer to the *SDK Help* HTML:
  www.xilinx.com/support/documentation/sw_manuals/xilinx2015_2/SDK_Doc/index.html

- Xilinx support forums
  http://www.xilinx.com/support.html

Digilent Website

- Arty home page
  http://www.artyboard.com

- Arty Wiki
  https://reference.digilentinc.com/arty

Other Web Resources

- LwIP Wiki
  http://lwip.wikia.com/wiki/LwIP_Wiki

- Tune LwIP TCP for best performance
  http://lwip.wikia.com/wiki/Tuning_TCP

- LwIP Home
  http://savannah.nongnu.org/projects/lwip/

Revision History

Version	Description	Date
1.0	Initial release for Vivado 2015.2	14 Aug 2015