LatticeMico32 Tri-Speed Ethernet MAC Gigabit Demo for the ECP5/ECP5-5G Versa Development Kit

User Guide

FPGA-UG-02007-1.1

August 2022

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults and associated risk the responsibility entirely of the Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Contents

Figures

Tables

1. Introduction

This document describes the LatticeMico32 Tri-Speed Ethernet Media Access Controller (MAC) IP demonstration for the ECP5™/ECP5-5G™ Versa Development Kit. The demonstration shows the ability of the Tri-Speed Ethernet MAC IP core to function in a real network environment operating at a link speed of either 10 or 100 megabits per second or 1000 megabits per second (gigabit). It is designed to be simple and easy-to-use, requiring no test equipment, lengthy setup, nor complex explanation. Because searching the Web is a familiar procedure to everyone, this demonstration uses a Web server to demonstrate the Tri-Speed Ethernet MAC IP core, using the open-source Lightweight IP (lwIP) network stack.

1.1. Prerequisites

The hardware, software, cable and general requirements for this demonstration are provided in the following sections.

1.1.1. Hardware Requirements

This demonstration requires the following hardware components:

- ECP5/ECP5-5G Versa Development Kit
- PC with a Gigabit Ethernet-capable (1000 Base-T) network card for running the demonstration in gigabit mode

1.1.2. Software Requirements

This demonstration requires the following software components:

- Internet browser on the gigabit-capable PC for accessing the Web server on the ECP5/ECP5-5G Versa Evaluation Board. This demonstration uses Internet Explorer for demonstrating the Web server functionality.
- LatticeMico32 System for optionally rebuilding or debugging the LatticeMico32 Web server software application
- Diamond Programmer System software for downloading the FPGA bitstream
- Lattice Diamond® Design Software for modifying the platform and regenerating the bitstream

1.1.3. Cable Requirements

The preferred cable for the gigabit Ethernet demonstration application is a category 5e (Cat 5e) or above twisted pair cable.

1.1.4. General Requirements

This demonstration requires some knowledge of the following:

- Familiarity with the LatticeMico32 System flow and usage, including the ability to download and debug LatticeMico32 software applications. The LatticeMico32 Tutorial is a good starting point for becoming familiar with LatticeMico32 System usage.
- Familiarity with basic TCP/IP networking concepts and troubleshooting skills and experience establishing basic network connectivity (for example, using a hub, switch, or swapped cable)
- Familiarity with IP addressing (and subnet mask), MAC addresses, the ping utility, and the ability to configure IP addresses on a PC

1.2. ECP5 Versa Development Kit

The demonstration runs on an ECP5/ECP5-5G Versa Development Kit connected to another host computer (demonstration PC) on the same network or through a crossover cable. The demonstration PC physically connects directly to the ECP5/ECP5-5G Versa Development board using a category 5e network cable. The setup is depicted in Figure 1.1.

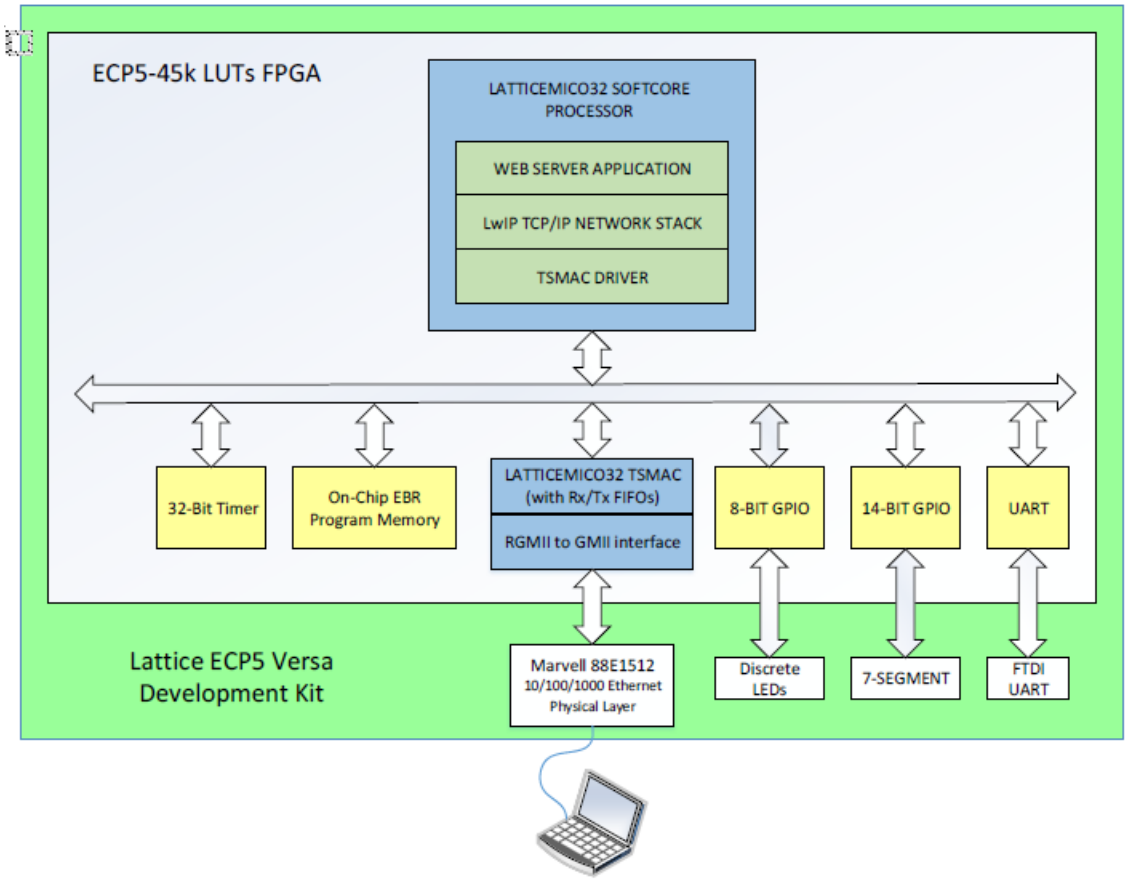

Figure 1.1. LatticeMico32 Tri-Speed Ethernet MAC Gigabit Demonstration Setup

1.3. Purpose

This simple demonstration shows the ability of the Tri-Speed Ethernet MAC to be configured with a MAC address, receive 802.3 frames (both physical and broadcast), filter these packets, and pass them to higher-protocol software.

The demonstration illustrates the following LatticeMico32 Tri-Speed Ethernet MAC features:

- Run-time software configurability for a 10BASE-T, 100BASE-TX, or 1000BASE-T link
- Real-world 802.3 Ethernet frames received and transmitted
- Acceptance of broadcast messages (destination MAC address ff.ff.ff.ff.ff.ff)
- Packet filtering based on configured MAC address
- Automatic padding of short frames
- Error counters and statistics

1.4. Application Space

This Web-server demonstration uses the Lightweight IP (lwIP) network stack, a comprehensive and highly configurable network stack that can be used as the basis for other applications requiring advanced network-stack functionality, such as IP fragmentation and reassembly, DHCP functionality, and TCP, UDP, and ARP functionality. See http://savannah.nongnu.org/projects/lwip/ for more information on lwIP network-stack usage considerations in embedded devices.

The LatticeMico32 Tri-Speed Ethernet MAC is a drag-and-drop module within LatticeMico32 System, facilitating system integration. Its simple software control and data-transfer operations make it a lightweight solution for application development. Although this demonstration uses polled mode, you can obtain higher throughput by using DMA transfers from memory to the Tri-Speed Ethernet MAC FIFO channels.

1.5. Limitations

The following cautions apply to this demonstration:

- The bitstream included with the demo design has no time-out restrictions. Rebuilding the demo without a valid TSMAC IP license will only allow evaluation operation for about an hour because the Tri-Speed Ethernet MAC evaluation version contains a built-in timer.
- This demonstration does not demonstrate how to dynamically configure the Tri-Speed Ethernet MAC according to link-speed changes at run time (for example, plugging from a 1000BASE-T to a 100BASE-TX connection or vice-versa). When changing link rates, press the reset button (FPGA GSRN) on the ECP5 Versa Evaluation board to allow the software to recognize and dynamically configure the new link rate.
- It is recommended that only one client at a time communicate with the Web server.
- Only hard-coded, static pages are displayed.
- Expect low throughput with dropped packets on a busy network with a lot of broadcast traffic, especially when debugging the application.

The ECP5UM-45F device has 108 block RAMs. About two thirds of these block RAMs are used by the TSMAC and are for the Mico platform. This demonstration is a comprehensive demonstration with significant debug features that enable you to become familiar with communication between the LatticeMico32 Tri-Speed Ethernet MAC and the Marvell 88E1512 10/100/1000 Ethernet PHY device. The pre-built application uses optimization for performance in processing TCP/IP packets. The Mico32 debug module is included in the Mico32 platform build so the debugger can attach and download a new application program. The USB UART is used for simple diagnostic output, as well as the board LEDs. The Mico32 Application code is deployed entirely into on-chip EBR so the entire application is up and running on power-up or on a reset of the FPGA.

2. Installing the Demonstration

Lattice distributes the demonstration package as an install executable. Download and then run the installation file. The demonstration package includes the MSB platform, a pre-built bitstream, and the C/C++ SPE Ethernet application project which you must import into LatticeMico32 C/C++ SPE. Additionally, the demonstration project contains a zipped lwIP source distribution file and a tool for generating embedded C source files from HTML pages.

You can either download (fast program) the pre-built bitstream into the ECP5 FPGA device each demonstration run or program it into the on-board SPI configuration Flash.

2.1. Installation

The LatticeMico32 Tri-Speed Ethernet MAC gigabit demonstration installs by default into the **C:\Lattice_DevKits\DK-ECP5&ECP5-5G-TSMAC-Demo** directory.

Note: If you intend to unzip the zipped demonstration file in a different directory, ensure that there are no spaces in the directory path or in the directory name. You will also need to ensure that the path names to resource files are regenerated in MSB components to point to the new location (EBR memory initialization file, TSMAC IP directory, etc.).

Figure 2.1 displays the directory hierarchy.

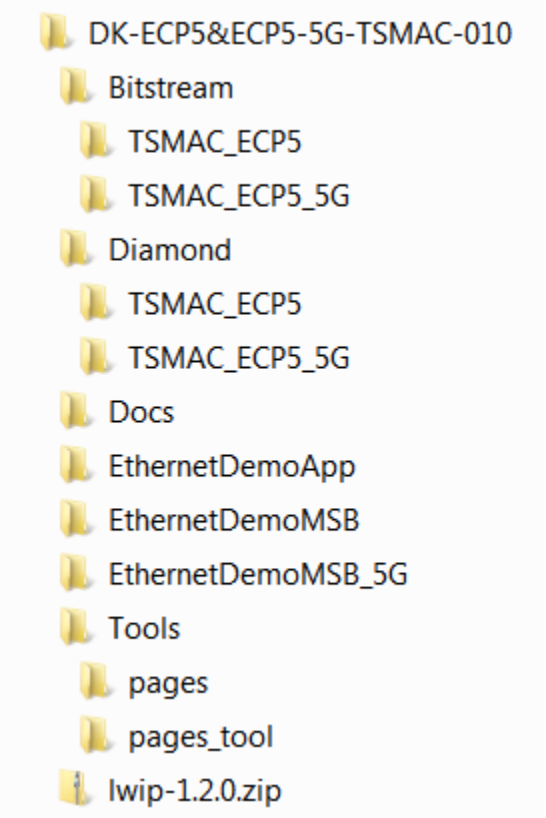

Figure 2.1. Demonstration Directory Structure

The directories and directory contents for the demonstration are as follows:

- Bitstream/TSMAC_ECP5 - Contains the pre-built demonstration bitstream for ECP5 Versa Development board with the read-to-run Web Server demo application.
- Bitstream/TSMAC_ECP5_5G - Contains the pre-built demonstration bitstream for ECP5-5G Versa Development board with the read-to-run Web Server demo application.

- Diamond/TSMAC_ECP5 – Contains the Diamond project for ECP5 Versa Development board used for building the bitstream
- Diamond/TSMAC_ECP5_5G – Contains the Diamond project for ECP5-5G Versa Development board used for building the bitstream
- Docs – Contains the user guide for the demonstration
- EthernetDemoApp – Contains the Ethernet demonstration C/C++ SPE project
- EthernetDemoMSB – Contains the Ethernet demonstration MSB project for ECP5 Versa Development board
- EthernetDemoMSB_5G – Contains the Ethernet demonstration MSB project for ECP5_5G Versa Development board
- Tools/pages – Contains the Web pages used for the project
- Tools/page_tools – Contains a tool and its source code for converting Web pages to a C source file that can be used with the C/C++ SPE Ethernet demonstration project

2.2. Environment Setup

The demonstration consists of two parts:

- A pre-built FPGA bitstream containing the LatticeMico32 Tri-Speed Ethernet MAC platform and Web Server software application which requires no modification
- An Ethernet demonstration software application, to modify and experiment with, that you can import into C/C++ SPE

If you are familiar with importing a software application into C/C++ SPE, you can move to the The Demonstration section after importing the Ethernet application.

2.2.1. Importing Ethernet Demonstration Software Application into C/C++ SPE

The steps in this section show you how to import the included Ethernet application software into the LatticeMico32 C/C++ SPE environment. It is assumed that you have unzipped the demonstration package with C:\ as the root.

1. Switch to the C/C++ SPE perspective.
2. Choose **File > Import** to activate the Select dialog box, shown in Figure 2.2.

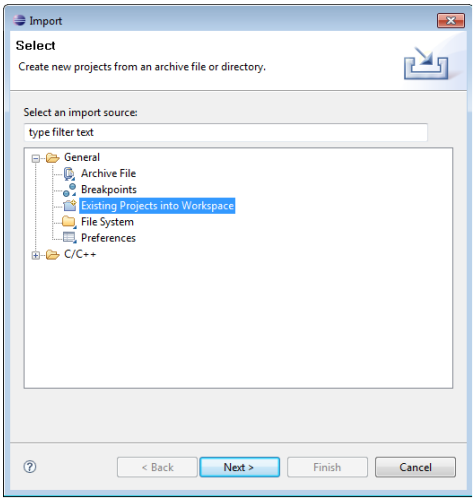

Figure 2.2. Select Dialog Box

3. Select **Existing Projects into Workspace**.
4. Click **Next** to activate the Import Projects dialog box, shown in Figure 2.3.

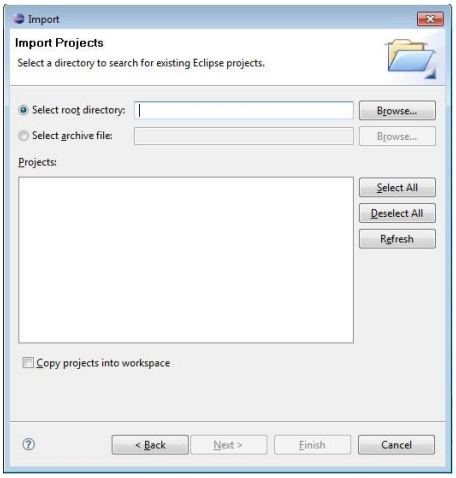

Figure 2.3. Import Projects Dialog Box

5. Select **Select root directory**, if it is not already selected.
6. Select the **Browse** button to activate the Browse For Folder dialog box.
7. In the Browse For Folder dialog box, browse to the **<demo_directory>/EthernetDemoApp** directory.
8. Select **OK**.

   The Import Project dialog box now shows EthernetDemoApp as one of the available projects in the Projects pane. If you do not see any listing in the Projects pane, you have not selected the directory correctly in the previous step.
9. Select **Finish**.

   The C/C++ Projects view in the C/C++ SPE perspective now lists the EthernetDemoApp project, as shown in Figure 2.4.

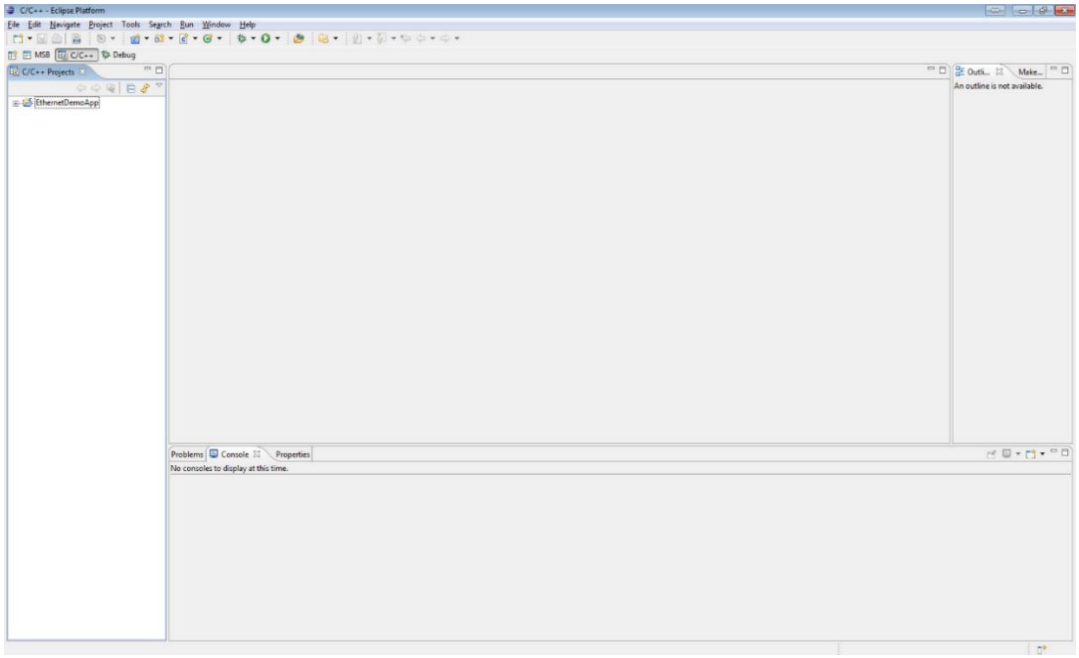

Figure 2.4. Ethernet Demonstration Application Listed in C/C++ Projects View

10. Rebuild the imported project, EthernetDemoApp, to verify a successful build.

    The project is configured so that debugging is turned off.

You have now successfully imported the Ethernet demonstration application into C/C++ SPE.

3. The Demonstration

In this section, you will learn how to set up and run the demonstration.

3.1. Board Setup

The ECP5/ECP5-5G Versa Evaluation Board DIP switches (SW3) should all be in the OFF position (towards top of the board).

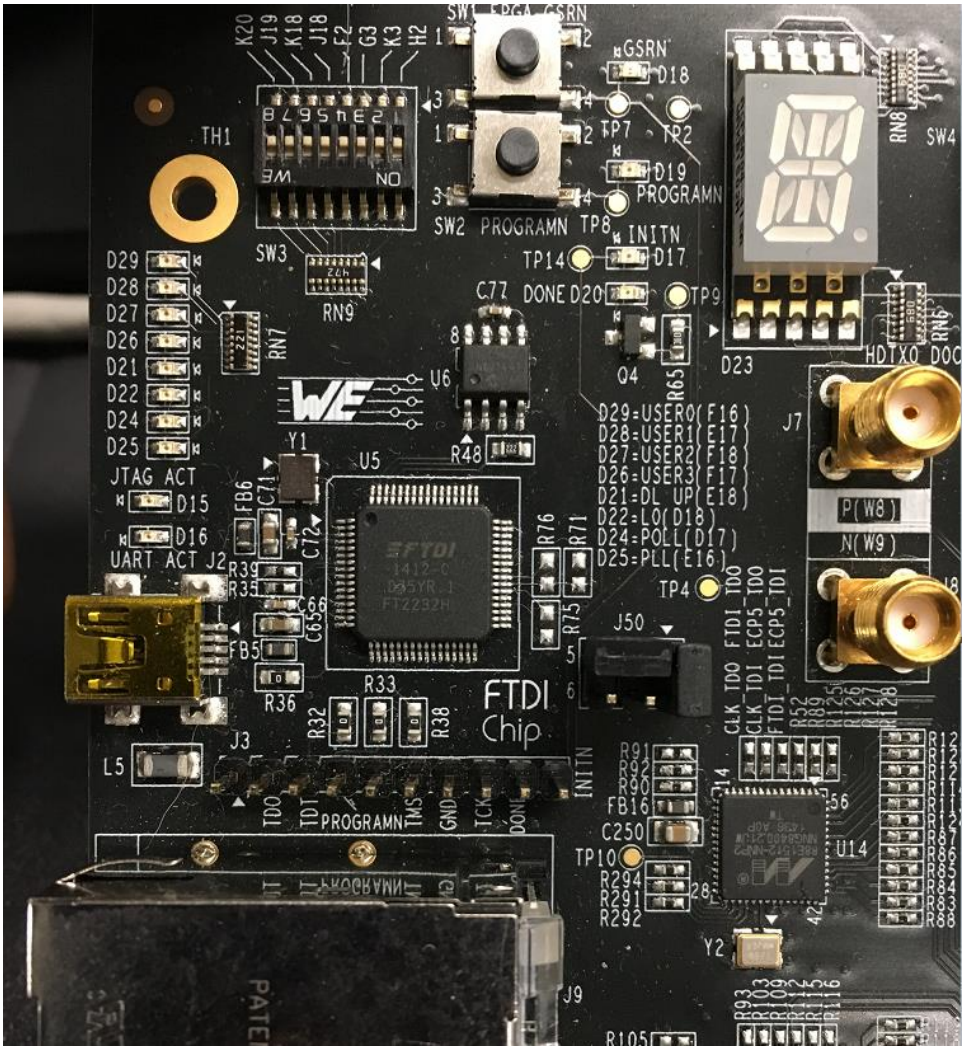

Figure 3.1. ECP5 Versa Evaluation Board

3.2. Network Cable Setup

For gigabit operation, it is recommended that you use a category 5e or above twisted pair cable. Insert one end of the category 5e twisted pair cable in the RJ-45 connector (which is closer to the USB connector) into the ECP5/ECP5-5G Versa Development Kit , and insert the other end of this cable into the RJ-45 connector of the gigabit Ethernet network card of the host PC.

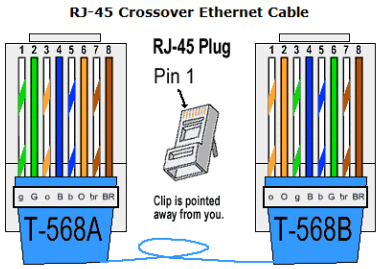

Figure 3.2. Twisted Pair Ethernet Cable

See Appendix A. Alternate Network Cable Setup if you do not have access to a gigabit Ethernet network card on the host PC but would still like to run the demonstration over a gigabit link.

You also can run the demonstration on a PC with a 100 Mbps network card without any modifications to the FPGA bitstream or the software application.

3.3. Host PC Network Parameters Setup

The host PC IP address must be configured so that it can communicate with the software application on the ECP5/ECP-5G Versa Evaluation Board that has a default IP address of 192.168.33.175. The IP address on the host PC must be of the format 192.168.33.xyz, where xyz can range from 2 to 254. The following example shows 192.168.33.30.

Warning: Do not use this method when connected on a company-wide network. You will probably cause conflicts with a machine assigned the same IP address. This method is only intended for direct connection between the test PC and the ECP5 Versa Evaluation Board.

1. On the Windows platform, change the PC IP address by selecting Start > Control Panel > Network and Sharing Center > Change Adapter Settings > Local Area Connection > Properties > Internet Protocol Version 4(TCP/IPv4) > Properties > Use the following IP address .
2. In the dialog box shown in Figure 3.3, select Use the following IP address, and enter the appropriate numbers for IP address, Subnet mask, and Default gateway.

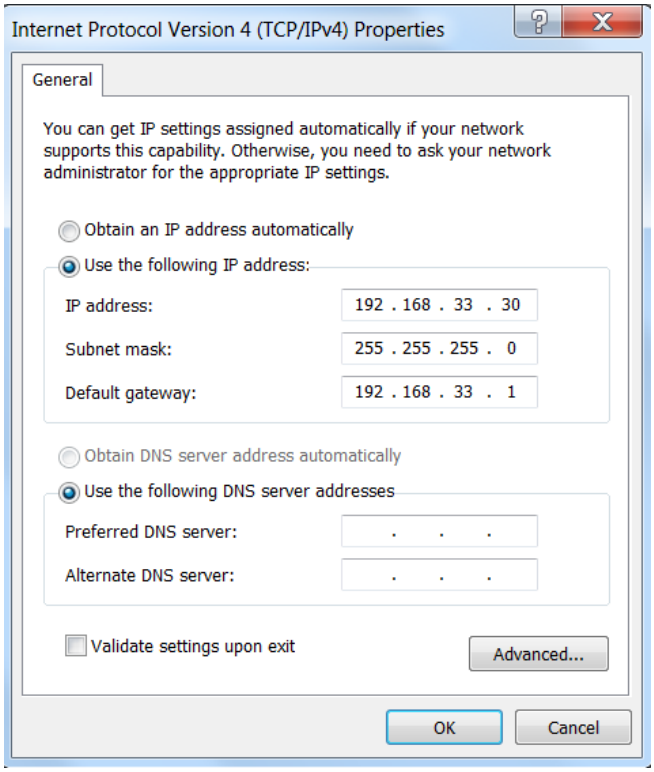

Figure 3.3. Setting Fixed IP Address Network Connectivity

3. Click **OK**.

3.4. LatticeMico32 Application Network Parameters Setup

Following are the default network parameters for the LatticeMico32 software application:

- Default IP address: 192.168.33.175
- Default gateway IP address: 192.168.33.1
- Default subnet mask: 255.255.255.0
- Default MAC address: 00-01-02-03-04-05

3.5. Running the Demonstration

This section guides you through the process of performing the demonstration. For visual status indication, this demonstration uses eight discrete LEDs (D25, D24, D22, D21, D26, D27, D28, and D29) on the board located next to the USB port. Paying attention to the LEDs, as well as to the C/C++ SPE project debug output, can help you detect problems in the proper operation of the hardware.

3.5.1. LED Indicators

The software controls eight discrete LEDs towards the left side of the board. Their functions are as follows:

Figure 3.4. LED Indicators

- D25 – Blinks approximately every second as the software passes through the main loop looking for incoming packets or processing the network-stack timer functions.
- D24 – Set when a packet has been received by the processor.
- D22– Set when a packet has been sent to the Tri-Speed Ethernet MAC by the processor from the Tri-Speed Ethernet MAC.
- D21 – Set when an ARP query for the processor's IP address is received by the network stack.
- D26 – Set when a finger protocol packet has been received by the processor.
- D27 – Set when an HTTP (Web page) protocol packet has been received by the processor.
- D28 – Set when an HTTP (Web page) protocol packet has been completely sent by the processor.
- D29 – Set when the software is preparing the Marvell 88E1512 10/100/1000 Ethernet PHY device for operation. Once preparation is completed, this LED is turned off and all other LEDs indicate operational status. While this LED is lit, you can refer to the C/C++ SPE debugger output to view the software activity.

If all eight LEDs blink simultaneously on and simultaneously off approximately every second, the software has encountered a fatal error when configuring the Marvell 88E1512 10/100/1000 Ethernet PHY device.

3.5.2. Downloading the FPGA Bitstream

The pre-built bitstream, ECP5_TSMAC_demo_Trispeed_impl .bit, is located in the Bitstream directory. Use the Diamond Programmer file located in the Bitstream directory to download the demonstration bitstream into the ECP5 device using the USB cable. Open Diamond Programmer and you will be presented with the Getting Started window. You can choose the cable type or select Detect Cable button to auto detect the cable. The Windows settings is shown in Figure 3.5.

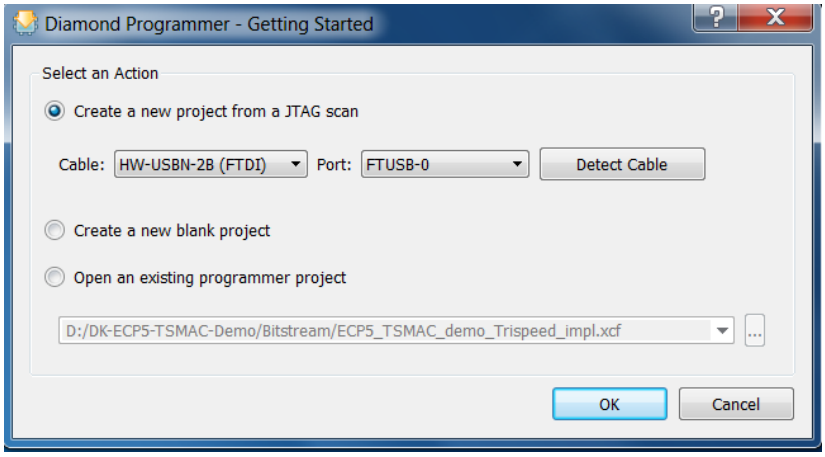

Figure 3.5. USB Port Settings

When USB setup has completed, select the Program button as shown in Figure 3.6 to program the bistream into the FPGA. When the FPGA bitstream has been successfully downloaded, the eight discrete LEDs near the USB connector are lit .

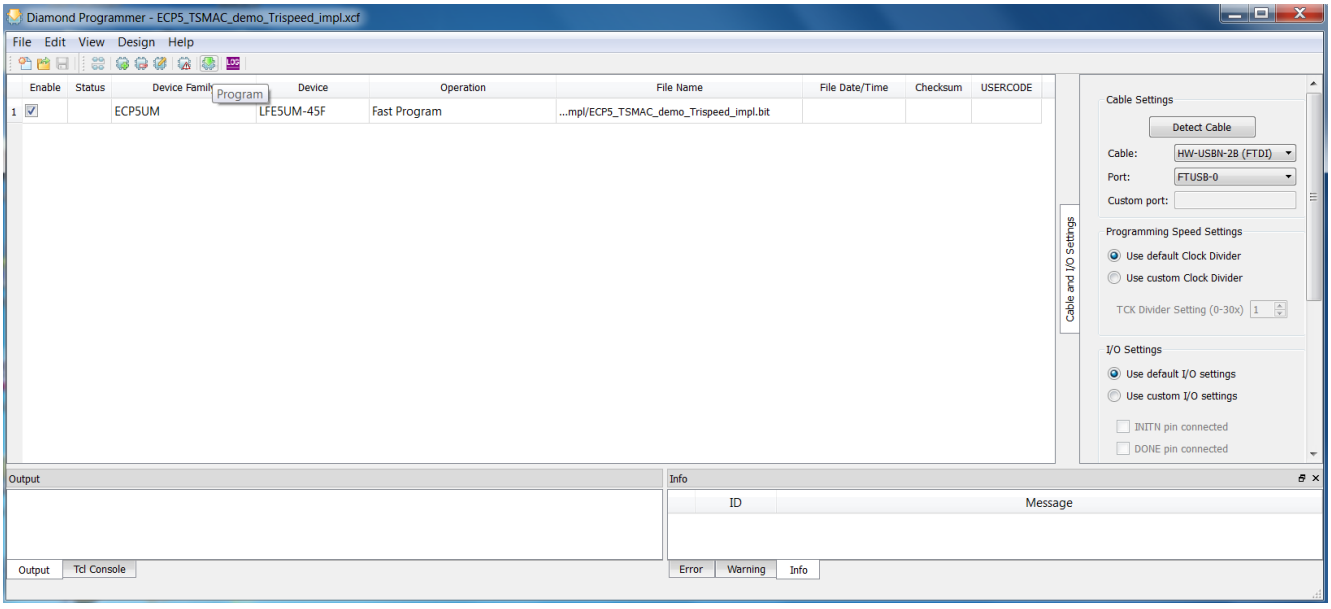

Figure 3.6. USB Port Settings

3.5.3. Pinging the Board

Pinging the board is the first step in ensuring proper network configuration and a properly functioning LatticeMico32 Tri-Speed Ethernet MAC platform and application. The ECP5/ECP5-5G Versa Evaluation Board responds to standard network ping commands from a host computer (demonstration PC). The ping must succeed before continuing, as shown in Figure 3.7, because other demonstrations will fail if the ping does not succeed.

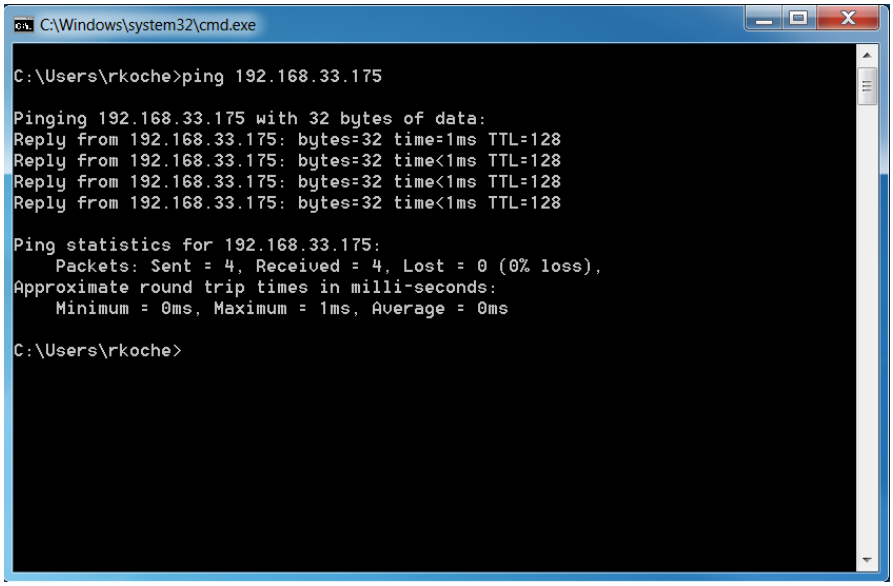

Figure 3.7. Successful Ping Console Output

The following LED diagnostics apply:

- D25 should toggle about every second. If D25 appears solid and is not blinking, the software has malfunctioned.
- D24 should light up to indicate incoming packets.
- D22 should light up on a successful ping, indicating that the board is sending packets back, most likely ping replies.
- D21 may light up in conjunction with D24 and D22. ARP requests usually are sent out by network entities if they do not already know the MAC address of a target with a specific IP address. Also, the network entities, especially PCs, tend to cache ARP replies for a period of time. If they already know the MAC address for an IP address, they may not send out an ARP request.

3.5.4. Accessing the Web Server

In addition to the LED diagnostics for the ping test, you may observe the following when the PC accesses the Web server:

- The 14-segment display will increment for each IP packet sent back to the PC.
- D27 lights up when the Web server receives an HTTP request.
- D28 lights up when the Web server has completed sending all the associated data for a Web page. The gap between the time that D27 lights up and the time that D29 lights up depends on the amount of data to send for a Web request.

The software running on the CPU inside ECP5/ECP5-5G implements a Web server that serves pages over an HTTP connection. Open Internet Explorer and simply enter the board's IP address in the address field.

The the proxy server on the Web browser must be disabled since it will be directly connecting to the ECP5 Versa Evaluation Board instead of going through a proxy server.

If you see the message shown in Figure 3.8, you must disable the proxy setting.

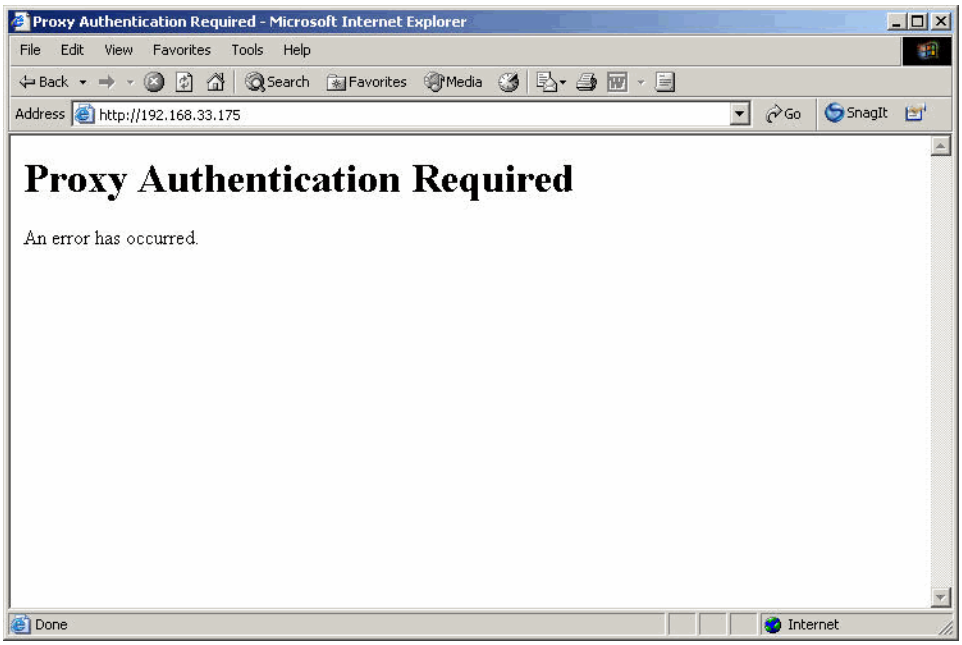

Figure 3.8. Proxy Authorization Errors

Perform the following steps to disable the proxy:

1. In Internet Explorer, choose Tools > Internet Options > Connections > LAN Settings.
2. In the dialog box shown in Figure 3.9, deselect Use a proxy server for your LAN.

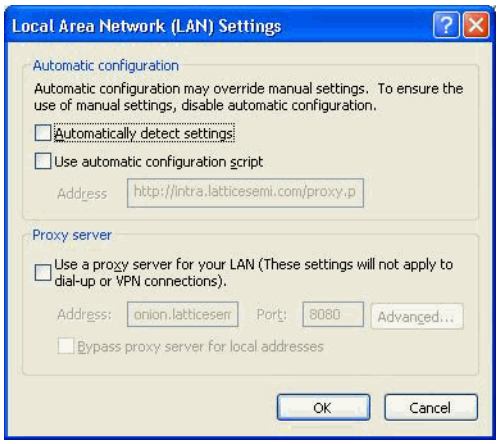

Figure 3.9. Proxy Disabled

3. Click **OK**.
4. Close and start Internet Explorer again.

If you have not disabled the proxy server and are trying to access the board's Web server through a twisted wire connection or a local hub, you may see the Internet Explorer window shown in Figure 3.10.

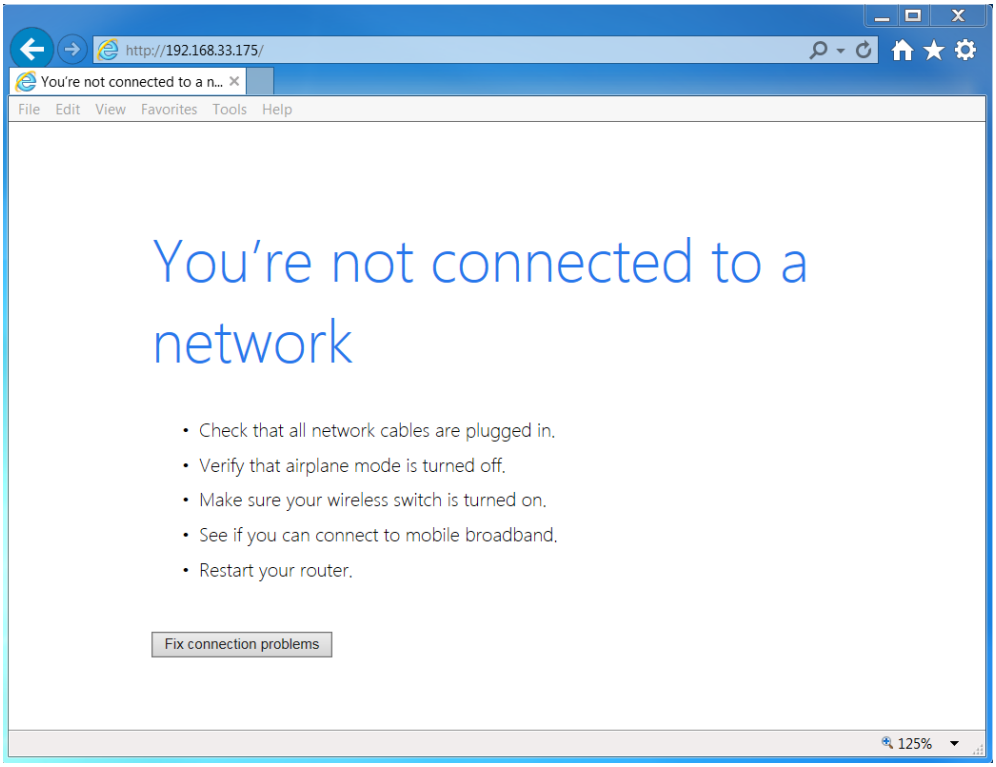

Figure 3.10. Connection Error in Internet Explorer

You must disable the proxy in Internet Explorer, as noted earlier.

Once the proxy is disabled, the host PC should be able to access the Web pages described in this section.

To access the home page, enter the board IP address in Internet Explorer, such as http://192.168.33.175/. Alternatively, you can enter http://192.168.33.175/index.html.

You may have to select Refresh in Internet Explorer if the page was the actively displayed page on Internet Explorer or if Internet Explorer was set to cache the Web page.

The home page has a link to the LatticeMico32 Tri-Speed Ethernet MAC product page, shown in Figure 3.11, below the displayed architecture image, also served by the Web server. You can click this hyperlink to retrieve the product summary page.

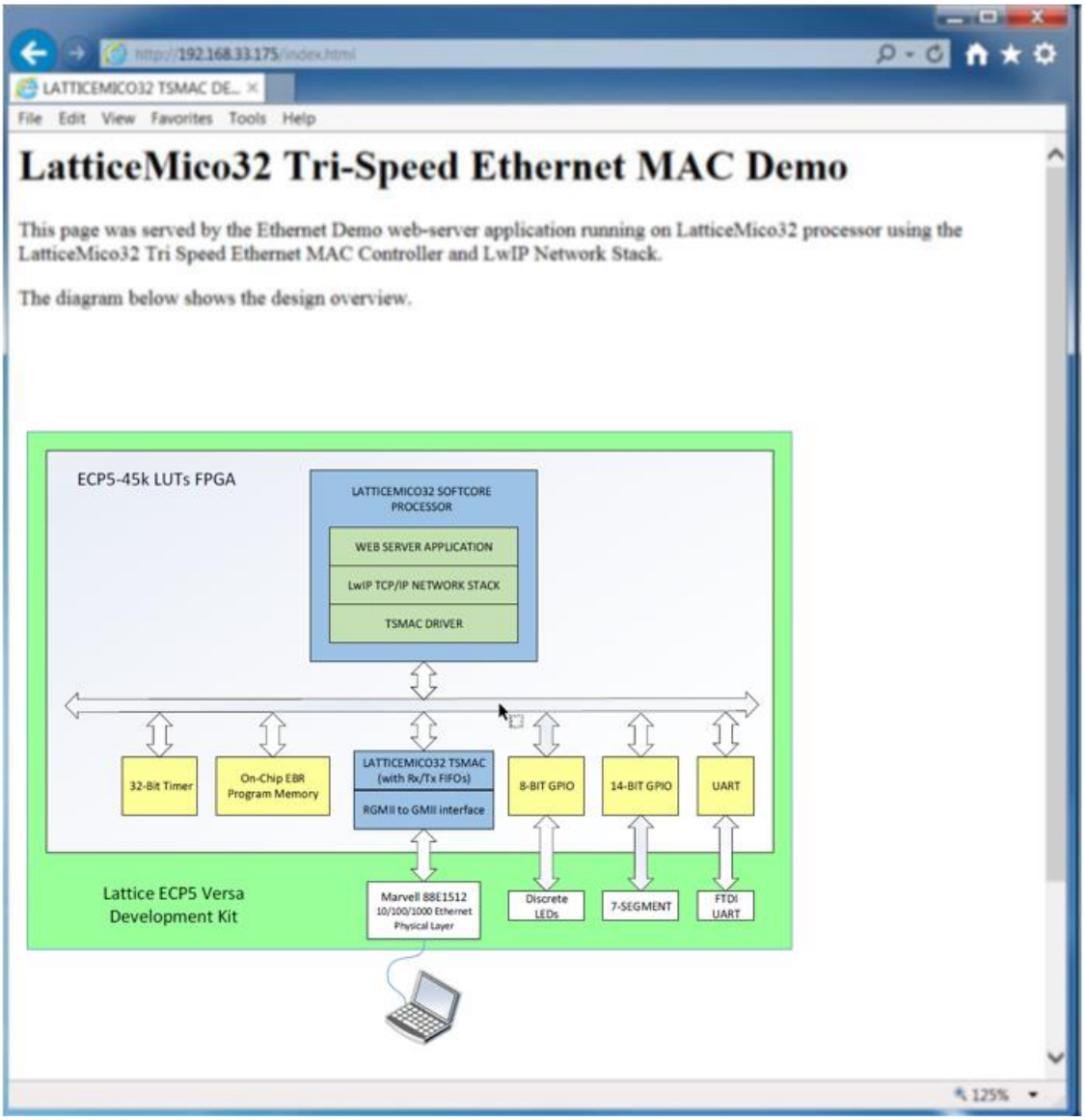

Figure 3.11. Web Server's Home Page

Figure 3.12 illustrates the product page. You can access this page by either clicking the link in the home page or by typing http://192.168.33.175/mac.html in the Internet Explorer address field.

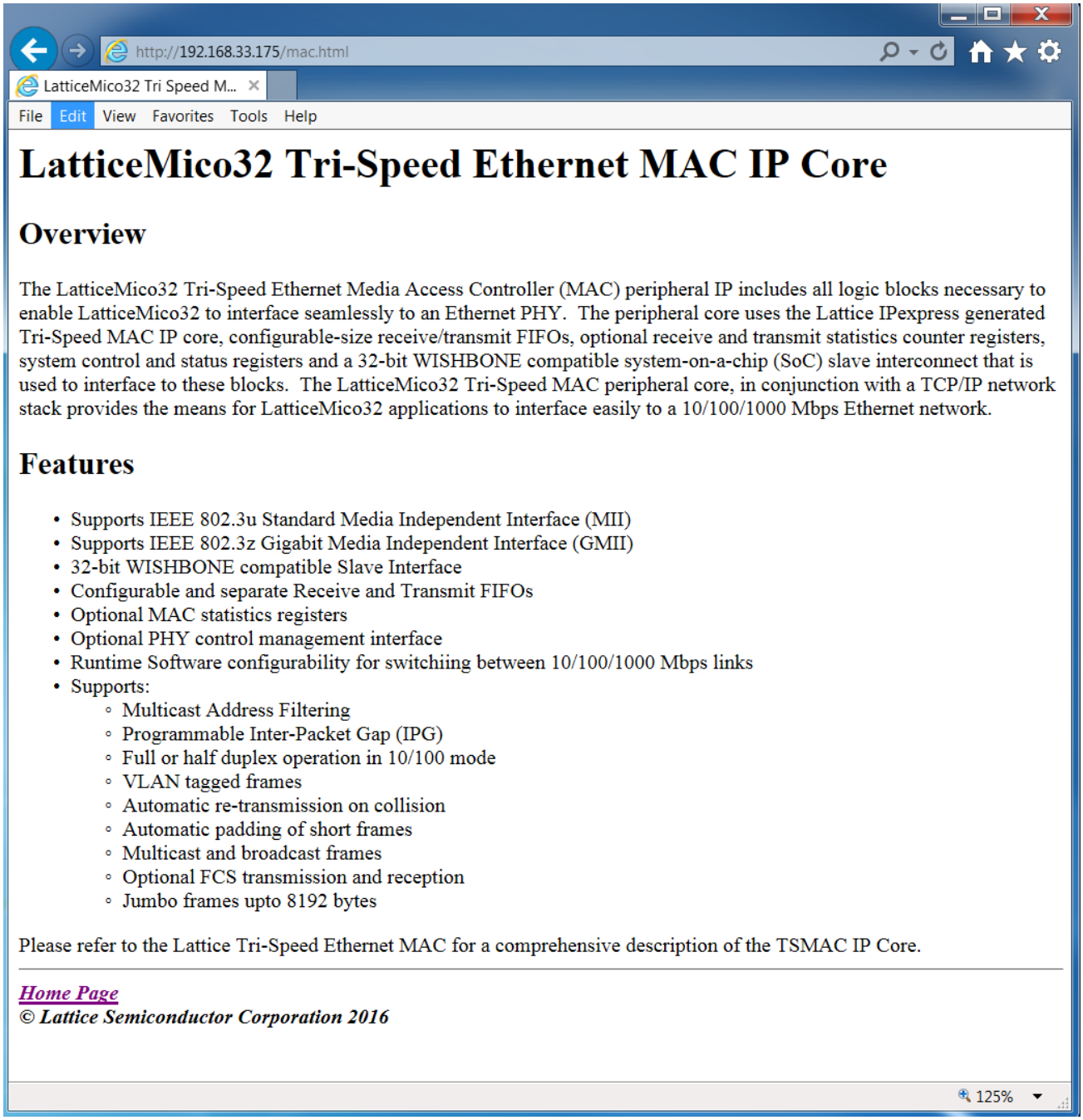

Figure 3.12. Tri-Speed Ethernet MAC Product Page

If you attempt to access a page that does not exist, a “file not found” page will appear. You can also access this page by typing http://192.168.33.175/404.html as the address in Internet Explorer.

4. Design Details

This section describes the demonstration system design and the software design.

4.1. Design Top Level

The platform Verilog file is not the design's top-level source file nor is the platform the top-level module.

The top_level.v file, located in the \soc directory, is the top-level file. It implements the top-level module that instantiates the platform as well as the rgmii2gmii.v module. It enables the connection of the Marvell 88E151210/100/1000 Ethernet PHY device's strap-up signals and provides the critical synthesis constraint for the Tri-Speed Ethernet MAC's RXD signals.

4.2. Clock Sources

The ECP5 Versa Evaluation Board has a 100 MHz oscillator clock source for the FPGA. The demonstration FPGA top-level design uses this clock source directly for the MSB platform.

The Tri-Speed Ethernet MAC requires a 125 MHz clock source for gigabit operation. The Marvell 88E1512 10/100/1000 Ethernet PHY device on the ECP5 Versa Evaluation Board is capable of outputting a 125 MHz clock when it negotiates a gigabit link. This output is used as the 125 MHz clock source required by the Tri-Speed Ethernet MAC IP. When the Marvell 88E1512 10/100/1000 Ethernet PHY device negotiates a 100 Mbps link, the clock output from this device falls to 25 MHz.

4.3. MSB Platform

Figure 4.1 shows the MSB platform used for the LatticeMico32 Tri-Speed Ethernet MAC gigabit demonstration. This platform is targeted to an ECP5 device with a 100 MHz processor clock. You can use the constraints file provided with the demonstration if you intend to recreate the platform. The current version of Mico System Builder doesn't support the "ts_mac_core" IP generation for ECP5-5G device. To create a MSB for ECP5-5G device, change the device to ECP5.

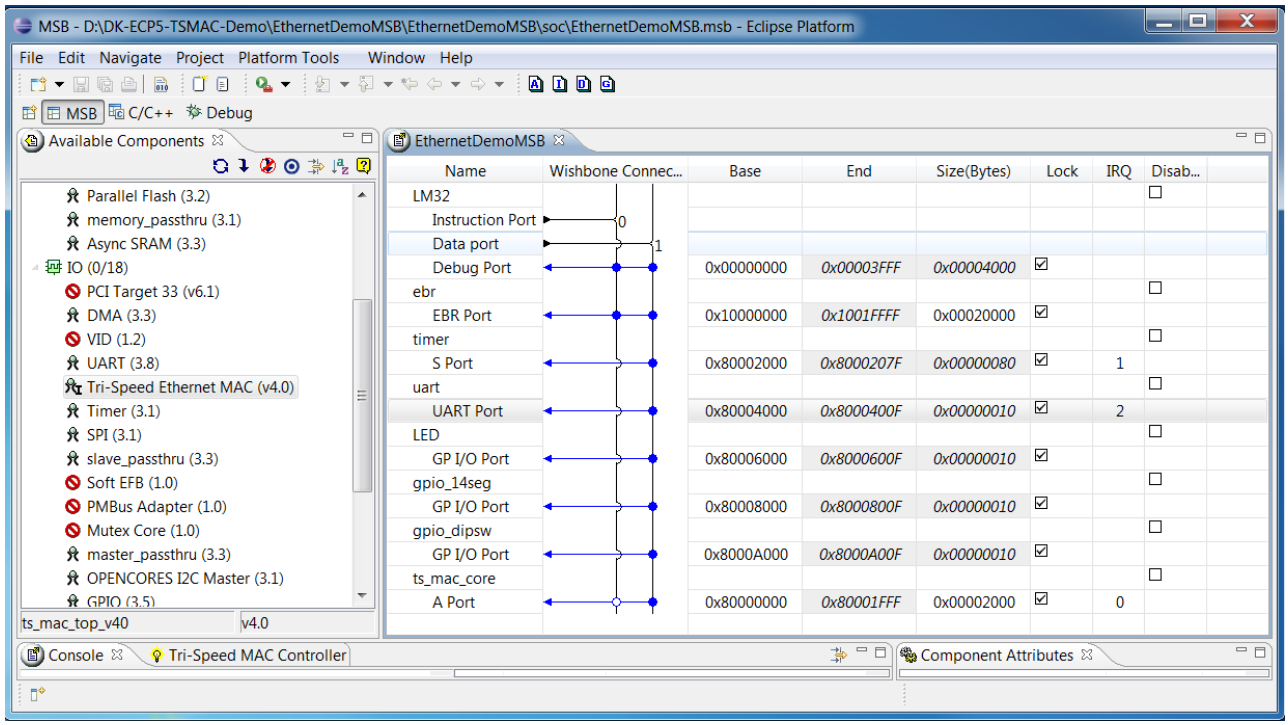

Figure 4.1. LatticeMico32 Tri-Speed Ethernet MAC Gigabit Demonstration Platform

Refer to Platform Configuration Dependency section before making any changes to the platform if you plan to use the Tri-Speed Ethernet MAC Gigabit Demonstration software application.

4.4. Software

The Ethernet demonstration C application consists of three parts:

- lwIP network stack for handling network packets and associated protocols, such as TCP/IP, ICMP, and ARP queries and replies, and ARP cache management
- User application for initializing the network stack, calling periodic network-stack maintenance, such as TCP slow and fast timer functions, and calling the network-stack function for handling incoming packets.
- Ethernet MAC driver for configuring the LatticeMico32 Tri-Speed Ethernet MAC and implementing transmit and receive functions.

4.4.1. Demonstration Application File Organization

Figure 4.2 shows the structure of the Ethernet demonstration application directory.

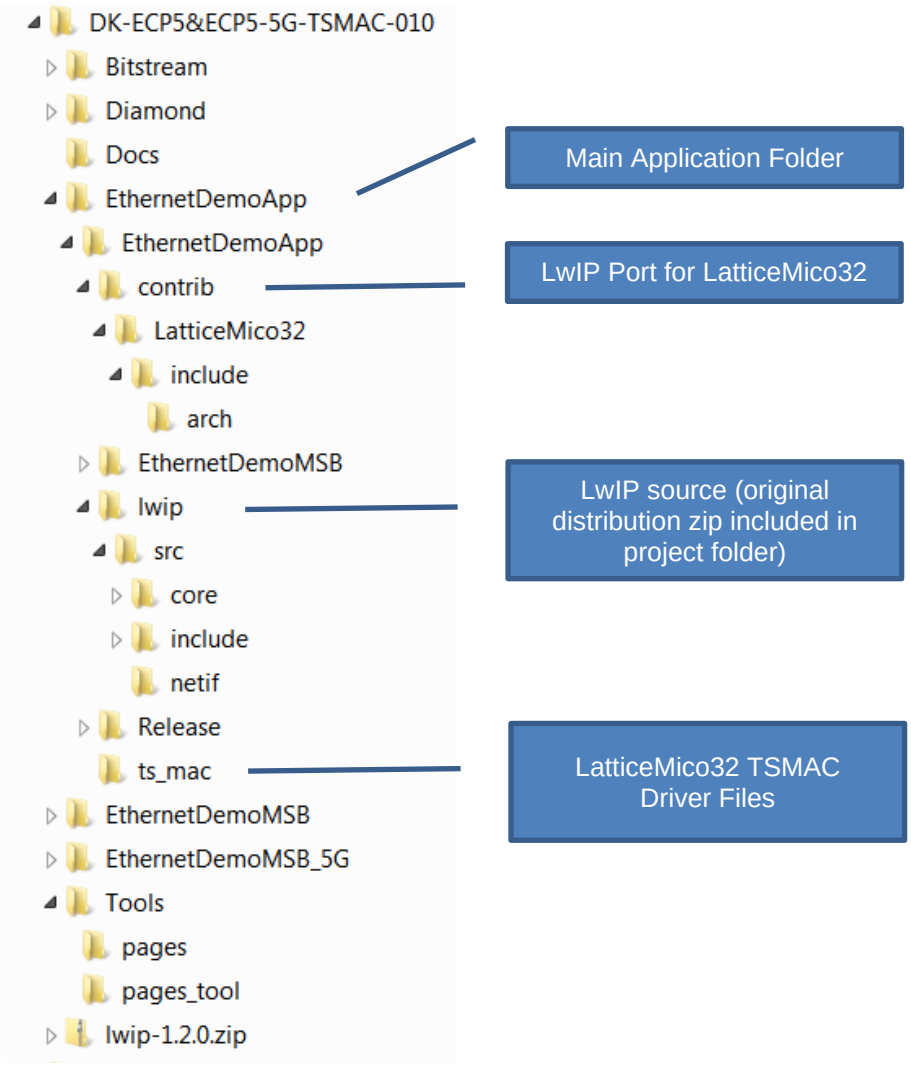

Figure 4.2. Organization of the Ethernet Demonstration Application Directory

The Release directory in the main application directory is generated as part of building the managed-build project. It contains the built executable and the intermediate object files. The EthernetDemo directory in the main application directory is the dynamically generated platform library directory generated by the managed-build process. Refer to the LatticeMico32 Software Developer User Guide for details on the managed-build process.

LwIP Port-Files for LatticeMico32 (contrib directory): This directory contains lwIP-required port files for LatticeMico32. The arch subdirectory contains three files.

- *cc.h* – Contains the mapping for lwIP data types to the processor-native data types, compiler structure-packing attributes and macro declarations for the debug and assert invocation required when debugging the lwIP operation.
- *perf.h* – Contains performance-stamping macro declarations. For this demonstration, these macros do not map to any functionality.
- *sys_arch* – Contains definitions when a system layer is used. For this demonstration, the system layer is not used. Refer to the lwIP documentation for more information.

These files should not be modified unless it is absolutely necessary.

LwIP Source Files (lwIP directory): LwIP is a comprehensive lightweight TCP/IP network stack. Because this demonstration is a managed-build project, it requires the source files to be part of the project. LwIP distribution contains additional sources that are not required. To avoid having the managed-build process automatically compile such sources, the essential files for the demonstration have been placed in this directory, lwip, maintaining the lwIP distribution hierarchy. The original lwIP distribution (release 1.2.0) from which this directory is populated is kept in the main application directory, named lwip-1.2.0.zip. More information on lwIP, including development status and licensing, is available at http://savannah.nongnu.org/projects/lwip/.

Basic lwIP configuration is controlled through the declarations in the files contained in the main application directory. The files in this directory should not be modified unless it is absolutely necessary, for example, to enhance functionality or to add or remove modules.

The following files have been modified from the package for this demonstration:

- \lwIP\src\netif\etharp.c – Modified to update LED status on receiving an ARP request

The following files/directories from the package are not used:

- \lwIP\src\api*
- \lwIP\src\core\inet6.c
- \lwIP\src\core\ipv6*
- \lwIP\src\include\ipv6*
- \lwIP\src\netif\ethernetif.c (this file is ported to LatticeMico32 in the ts_mac directory)
- \lwIP\src\netif\loopif.c
- \lwIP\src\netif\slipif.c
- \lwIP\src\netif\ppp*

LatticeMico32 Tri-Speed Ethernet MAC Driver Files (ts_mac directory): Figure 4.3 shows the list of files in the ts_mac directory. Basic Tri-Speed Ethernet MAC functionality configuration is controlled through definitions in files that reside in the main application directory. Do not modify the files in this directory unless you modify the driver, that is, implement MIIM control of the Marvell 88E1512 10/100/1000 Ethernet PHY device or a different MAC data-transfer mechanism.

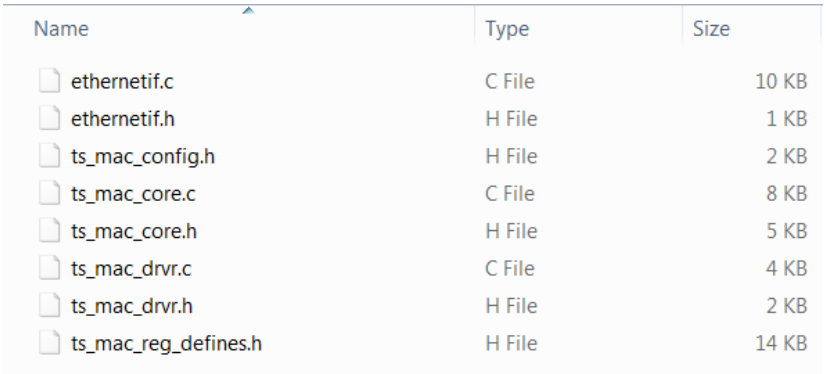

Figure 4.3. LatticeMico32 Tri-Speed Ethernet MAC Driver Files

The Tri-Speed Ethernet MAC driver files are the following:

- *ts_mac_drvr.h* – This file declares driver functions available for external modules. The functions declared are those that are required for lwIP usage.
- *ts_mac_drvr.c* – This file implements driver functionality for initializing the Tri-Speed Ethernet MAC, transmitting and receiving packets, and some query functions. These functions are created for lwIP usage but can be used as reference code for creating custom drivers.
- *ts_mac_core.h* – This file declares core functionality required by the Tri-Speed Ethernet MAC driver functions.
- *ts_mac_core.c* – This file implements functions required by the Tri-Speed Ethernet MAC driver functions.
- *ts_mac_reg_defines.h* – This file contains absolutely essential Tri-Speed Ethernet MAC-specific register definitions. It is used only by the Tri-Speed Ethernet MAC driver and core routines.
- *ts_mac_config.h* – This file contains Tri-Speed Ethernet MAC-specific default values used by the Tri-Speed Ethernet MAC driver as part of Tri-Speed Ethernet MAC configuration.
- *ethernetif.h* – This file declares functions in ethernetif.c available for external modules.
- *ethernetif.c* – This file connects lwIP-specific transmit, receive, and initialization routines required by lwIP usage to the Tri-Speed Ethernet MAC-specific routines.

Files in Main Application Directory: Figure 4.4 shows the files in the main application directory.

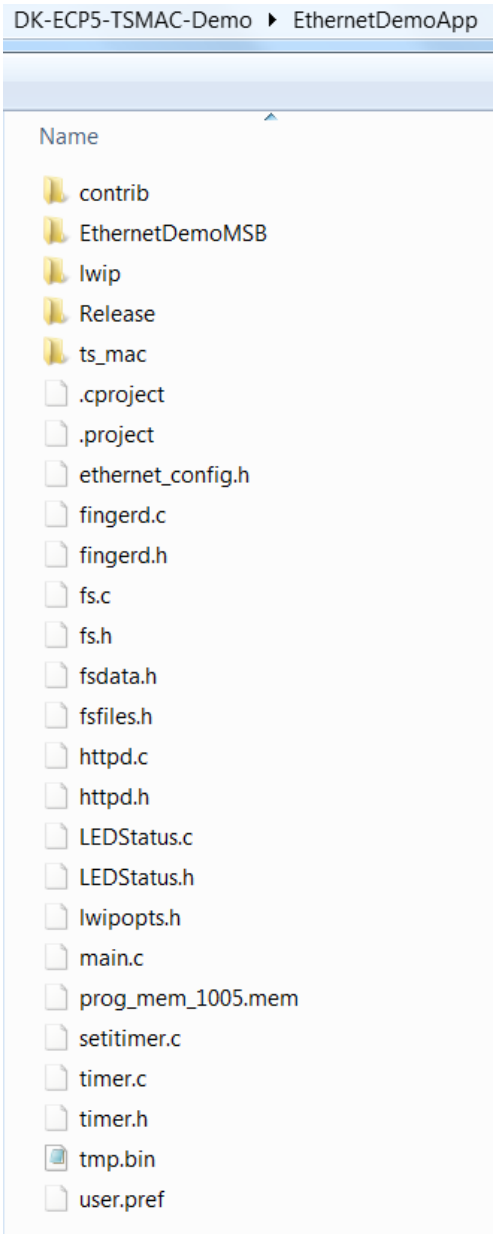

Figure 4.4. Files in Main Application Directory

In Figure 4.4, .cproject, .project, and user.pref are C/C++ SPE-related files and should not be modified in any way. Modifying these files can render the demonstration project unusable.

The lwip-1.2.0.zip file contains the original unmodified lwIP source code as obtained from the lwIP web site. The source in this zip file is used for the demonstration.

The main application directory contains two sets of files:

- Files that provide base functionality and do not require modification
- Files that provide configuration data and may require modification

The following files provide base functionality:

- *fs.h* – This file is part of the HTML file-system implementation and should not be modified unless you make changes for debugging or enhancement.

- *fs.c* – This file is part of the HTML file-system implementation and should not be modified unless you make changes for debugging or enhancement.
- *fsdata.h* – This file is part of the HTML file-system implementation and should not be modified unless you make changes for debugging or enhancement.
- *httpd.c* – This file implements the HTTP server functionality.
- *httpd.h* – This file declares functions exposed by httpd.c to external modules.
- *fingerd.c* – This file implements the finger-server functionality.
- *fingerd.h* – This file declares functions exposed by fingerd.c to external modules.
- *timer.c* – This file implements the functionality for multiple timers, which are needed for periodically invoking lwIP functionality, such as periodic update of the ARP cache and TCP retransmission of unacknowledged packets. The implementation in this file relies on a single periodic callback, which is provided by the LatticeMico32 timer driver, using the platform timer component.
- *timer.h* – This file declares functions exposed by timer.c for external modules.
- *setitimer.c* – This file contains functions that set up the single platform timer and periodically invoke the callback function provided by timer.c. The functionality in this file uses the first timer component that it finds in the platform.
- *MicoUartService.c* – This file overrides the functions implemented in the default MicoUartService.c file that is populated by the managed-build process in the platform library directory. It inserts a \r character (carriage-return) before sending a \n (new line) character, because the Hyperterminal program on Windows does not allow treating a new line as a carriage return followed by a new line. If you intend to use a different console application on the development PC or do not need this functionality, you can delete this file.
- *main.c* – This file implements the main() function that performs the appropriate initialization and contains the main control loop.
- *LEDStatus.c* – This file implements the LED status functionality.
- *LEDStatus.h* – This file declares the functions available to other modules for setting and controlling the respective LED status. It also declares constants for LED definitions.

The following files provide configuration data. They configure the demonstration application parameters, such as network parameters and debug settings for the application, lwIP, and the embedded HTML pages. These files are C header files.

Note: Since C/C++ SPE is not able to track dependent files, you must rebuild the Ethernet demonstration application if you modify any of the header files (.h file), rather than just build it.

- *ethernet_config.h* – This file contains the basic Ethernet demonstration network configuration data, namely, the MAC address, IP address for the ECP5 Versa Evaluation Board, gateway IP address, and subnet mask. Modify this file as needed.
- *lwipopts.h* – This file controls compile-time settings for lwIP, such as enabling and disabling features like UDP and TCP. It also contains debug settings for lwIP. This file is included in the opt.h lwIP configuration file located in /lwip/src/include/lwip directory. The opt.h header file contains default lwIP configuration settings. These settings are used only if they are not already declared in lwipopts.h, so lwipopts.h provides a convenient means of overriding the default lwIP configuration. See the opt.h header file for a complete list of configurable settings.
- *fsfiles.h* – This file defines the embedded HTML pages and the associated file system. Only the fs.c source file includes this file. Originally this file was a C source file named fsdata.c included by the fs.c file; however, since the C/C++ SPE managed build tends to include all C source files for compilation (resulting in a compilation error), fsdata.c was renamed to fsfiles.h. Including this file in any other C source file results in a compilation error, because this file contains data structure instances.

4.4.2. Configurable LwIP Network-Stack Parameters

The opt.h lwIP header file, located in the /lwip/src/includes/lwip/ directory, contains default lwIP-configurable parameters. You can override these parameters by defining them in the lwipopts.h header file located in the main application directory.

Some of the configurations defined in the lwipopts.h header file for reducing the network-stack code size are the following:

- UDP support is turned off because the Ethernet demonstration application does not require UDP.
- UDP checksum generation for transmit and check for receive is turned off. When UDP is turned off, it overrides these settings.
- LwIP statistics are turned off.
- IP reassembly and Packet fragmentation are turned off.
- IP options are turned off. LwIP therefore discards packets that contain an options field in the IP header.

4.4.3. Configurable Network and MAC Parameters

Changing the Ethernet demonstration application's network parameters does not require you to generate a new bitstream.

The configurable network parameters are defined in the ethernet_config.h header file in the SPE project, shown in Figure 4.5.

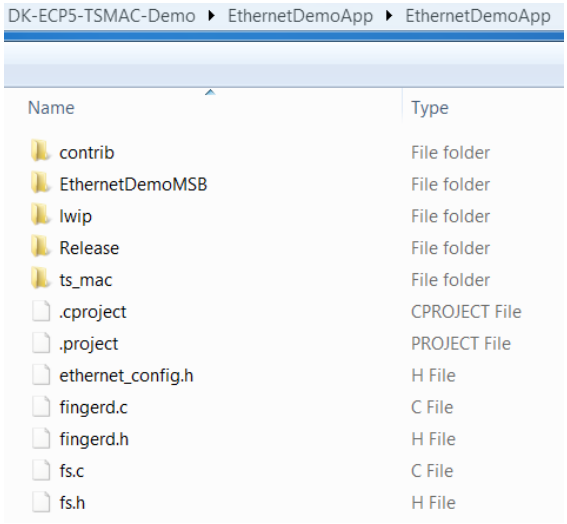

Figure 4.5. Location of the Network Parameters File

The contents of the network parameters file are shown in Figure 4.6. The lwIP network stack used by the Ethernet demonstration application has built-in DHCP support that can be enabled or disabled at compile time. This demonstration does not use DHCP, because it is disabled (LWIP_DHCP is set to 0 in lwipopts.h header file). The demonstration therefore uses the parameters defined in the LWIP_DHCP conditional. You can modify the provided MAC address, IP address, gateway IP address, and the subnet mask in this file.

```
/*
 * This file defines constants that control
 * application-configuration
 */

#ifndef ETHERNET_CONFIG_H_
#define ETHERNET_CONFIG_H_
#include "lwip/opt.h"

/*
 * MAC Address (Software configured)
 * (e.g. 00-01-02-03-04-05)
 */
#define MAC_CFG_MAC_ADDR_UPPER_16    (0x0001)
#define MAC_CFG_MAC_ADDR_LOWER_32    (0x02030405)


/*
 * IP Addresses when not using DHCP
 */
#if !LWIP_DHCP
/* Host ip-address: 192.168.33.175 */

#define HST_IP_ADDR_0    (192)
#define HST_IP_ADDR_1    (168)
#define HST_IP_ADDR_2    (33)
#define HST_IP_ADDR_3    (175)

/* Gateway IP Address: 192.168.33.1 */
#define GW_IP_ADDR_0     (192)
#define GW_IP_ADDR_1     (168)
#define GW_IP_ADDR_2     (33)
#define GW_IP_ADDR_3     (1)

/* Subnet: 255.255.255.0 */
#define SUBNET_MASK_0    (255)
#define SUBNET_MASK_1    (255)
#define SUBNET_MASK_2    (255)
#define SUBNET_MASK_3    (0)

#endif
#endif /*ETHERNET_CONFIG_H_*/
```

Figure 4.6. Contents of the Network Parameters File

Once you modify this file, be sure to rebuild (using the Rebuild command) the project rather than build it (using the Build command), because C/C++ SPE does not correctly identify dependent files when modifying header files. Once the rebuilding is finished, the generated executable contains code to configure the Tri-Speed Ethernet MAC and the lwIP network stack with the new parameters.

4.4.4. Platform Configuration Dependency

This application has the following platform dependencies:

- The LatticeMico32 Tri-Speed Ethernet MAC MSB component instance must be named ts_mac_core. If you rename the component instance, you must modify the low_level_init function in the ethernetif.c source file to correct the MAC base-address definition.
- The application must include at least one 32-bit timer instance. If it contains a single timer instance, it must not be used by the application. If there are more instances, the first instance of the timer is used by the Ethernet demonstration application for the network timer facility. To change this behavior, modify the setitimer.c source file.
- The application must include an 8-bit output GPIO named LED, which is used by the LEDStatus.c file to update the status of the LEDs according to the application activity. If you plan to remove the LED GPIO, you must make the appropriate changes to the LEDStatus.c file to avoid compilation errors.
- The finger daemon reports Tri-Speed Ethernet MAC counter values. If you do not plan to use the Tri-Speed Ethernet MAC statistics module, you must disable the finger daemon initialization in the main() function. Disabling the finger daemon initialization automatically excludes all finger code.

4.4.5. Setting Up the Marvell 88E1512 10/100/1000 Mbps Energy Efficient Ethernet Transceiver PHY Device

The main module performs the following steps to configure the Marvell 88E1512 10/100/1000 Ethernet PHY device in this demonstration before entering the main loop:

1. Issues a soft reset to the Marvell 88E1512 10/100/1000 Ethernet PHY device.
2. Configures the Marvell 88E1512 10/100/1000 Ethernet PHY device to auto-negotiate.
3. Requests the Marvell 88E1512 10/100/1000 Ethernet PHY device to auto-negotiate.
4. Queries the Marvell 88E1512 10/100/1000 Ethernet PHY device for the link speed, once auto-negotiation is complete.
5. Configures the ethernetif structure for allowing the lwIP Ethernet drivers to configure the Tri-Speed Ethernet MAC for correct operation.

4.4.6. Main Control Loop

The entire control application exists in one forever loop. This is the standard approach to small control systems. Interrupts are not used, nor is an operating system or executive, although lwIP can be ported for use with an operating system. The program looks for a packet to be read from the Tri-Speed Ethernet MAC. If pending, the packet is read by the functions in ethernetif.c, and appropriate lwIP calls are made to process this incoming packet. Once the check for a received packet is processed, the main loop calls the appropriate lwIP timer functions that must be called periodically if their associated timers have elapsed.

4.4.7. Web Server Execution

The Web server application is implemented in the httpd.c source file. Initialization of the Web server, which resides in httpd_init and is invoked by main(), provides a callback function to lwIP that is called when an incoming HTTP connection is accepted by the lwIP network stack. This call-accept callback function, in turn, provides the callback function that lwIP calls when it receives a packet for this accepted connection. In effect, the Web server is part of the main control loop, because the main control loop calls the lwIP functions when it receives packets from the Tri-Speed Ethernet MAC.

4.4.8. Finger Server Execution

The finger application is implemented in the fingerd.c source file. The finger server is initialized in the main() function by calling the fingerd_init function. The finger server has an architecture very similar to that of the Web server and can be used as reference code for other applications. Processing the finger request and sending the data is performed through callback functions called when network data arrives or is sent. In effect, the finger server is also part of the main control loop, because the main control loop calls the lwIP functions when it receives packets from the Tri-Speed Ethernet MAC.

4.4.9. Tri-Speed Ethernet MAC Driver

The software uses a driver to configure and control the Tri-Speed Ethernet MAC IP and the external Marvell 88E1512 10/100/1000 Ethernet PHY device through the host interface registers. The driver also handles the FIFO interface to send and receive Ethernet packets. If you plan to use these drivers as is in a multi-tasking situation, you must modify the provided drivers for protection against calls from being accessed simultaneously by multiple threads.

The driver is implemented in the tsmac_init function in the ts_mac_drvr.c source file. This function performs the following Tri-Speed Ethernet MAC initializations:

- Sets the operation mode.
- Sets a maximum packet size limit for the Tri-Speed Ethernet MAC.
- Sets MAC RX/TX FIFO thresholds.
- Sets the MAC address.

4.4.10. TCP/IP Software Stack

The Ethernet demonstration software application uses the open-source lwIP network stack. The following description is taken from the lwIP home page (http://savannah.nongnu.org/projects/lwip/):

LwIP is a small independent implementation of the TCP/IP protocol suite that has been developed by Adam Dunkels at the Computer and Networks Architectures (CNA) lab at the Swedish Institute of Computer Science (SICS).

The focus of the lwIP TCP/IP implementation is to reduce resource usage while still having a full-scale TCP. This makes lwIP suitable for use in embedded systems with tens of kilobytes of free RAM and room for around 40 kilobytes of code ROM.

The lwIP implementation offers the following features:

- IP (Internet Protocol), including packet forwarding over multiple network interfaces
- ICMP (Internet Control Message Protocol) for network maintenance and debugging
- UDP (User Datagram Protocol), including experimental UDP-lite extensions
- TCP (Transmission Control Protocol) with congestion control, RTT estimation, fast recovery, and fast retransmit
- Specialized raw API for enhanced performance
- Optional Berkeley-like socket API
- DHCP (Dynamic Host Configuration Protocol)
- PPP (Point-to-Point Protocol)
- ARP (Address Resolution Protocol) for Ethernet

You can find additional information on lwIP at http://savannah.nongnu.org/projects/lwip/.

4.5. Modifying Web Pages

Perform the following steps to modify the Web pages or to provide additional Web pages. The Web page for “File Not Found” is embedded in the Web server so that if the embedded file system is not created correctly, the Web server will still be able to serve this page.

1. Place your new Web pages or modified Web pages in the /tools/pages directory of the demonstration installation.
2. If you are adding new Web pages, add the names of the new Web page files to files.txt. Do not leave a new line at the end of the file list.
3. Start the LatticeMico32 System SDK shell and execute the makepages_lwIP.exe program, as shown in Figure 4.7.

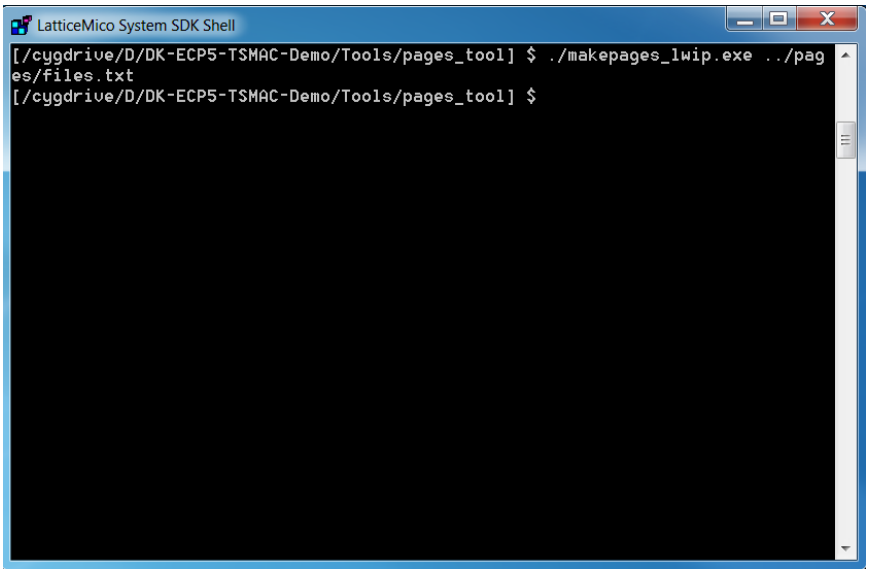

Figure 4.7. Running makepages_lwip.exe from LatticeMico312 System SDK Shell

The output of a successful run of makepages_lwIP.exe is a C source file, fsdata.c.

4. Rename this file to fsfiles.h.
5. Replace the provided fsfiles.h file in your EthernetDemoApp project, which was imported in C/C++ SPE, and rebuild the project. You must rebuild the project rather than build it.

To verify your modifications to the Web pages, run the Ethernet demonstration application and try to access the Web pages. If you do not find the Web page, the Web the server will issue a 404 (page not found) error.

You may have to select Refresh in Internet Explorer if the page was the actively displayed page on Internet Explorer or if Internet Explorer was set to cache the Web page.

5. Troubleshooting

Following are some commonly encountered problems in the demonstration and possible solutions.

The board or demonstration does nothing:

1. Make sure that the software heartbeat (D25) is blinking.
2. Make sure that the decimal point on the 7 segment LED is blinking. This confirms that the hardware is working correctly with the board clock running.
3. Reload the bitstream from the demonstration Bitstream directory.

You cannot ping the board:

1. Make sure that the visual indicators provided in LED Indicators light up as indicated.
2. Make sure that the IP address configuration is set correctly on the ECP5/ECP5-5G Versa Evaluation Board, as well as on the host computer.
3. Make sure that the subnet masks match up; the network stack has a strong relationship to the subnet mask.
4. Make sure that you are using either a hub or a crossover cable.
5. If the problem persists, turn on lwIP debugging and debug the application. The starting point is in the receive routine ethernetif_input function, called from the main() function.

The Web browser does not display pages:

1. If pinging the board succeeds, the network configuration is correct.
2. Make sure that the proxy, if used, is disabled on the browser.
3. For extreme network activity that causes the demonstration software to lose packets, use a direct connection to the board.

Unable to connect to the UART port

1. If unable to connect to the UART port, make sure that the UART ports of ECP5 are connected to the design UART port. Confirm this by assigning the correct pins in the constraints file.
2. If the hardware pins are connected correctly, make sure that in the EthernetDemoMSB, under Properties in the platform section, the STDIO direction is set to RS-232 (UART).
3. After confirming the above settings and the UART is still not enabled, confirm that the FTDI chip is programmed so as to unable the UART port on the FTDI.

Rebuilding the MSB Platform

1. When rebuilding the MSB platform, the top level Verilog file generated by the Mico System Builder has to be updated manually to add the “gbit_en” signal as an output port.

If you need to analyze the network traffic, connect the ECP5/ECP5-5G Versa Evaluation Board directly to the host computer, using either a cable or a hub. Then download and install the WireShark (formerly known as Ethereal) network protocol analyzer on the host computer. Allow it to install WinPCap as part of the installation. WireShark is a powerful network protocol analyzer that enables you to capture, display, and analyze network traffic at the host computer’s network interface.

6. Third-Party Software

Table 6.1 lists the third-party software tools and source code that are used in this demonstration. These packages are all freely available to the general public from various Web sites on the Internet.

Table 6.1. Third-Party Software Tools Used in Demonstration

Software	Source/Author	License	Description
lwIP	Adam Dunkels	BSD style (Open Source)	TCP/IP network stack (C language) Project page: http://savannah.nongnu.org/projects/lwip/

7. References

The following documents provide more information on topics discussed in this guide:

- LatticeMico32 Software Developer User's Guide
- LatticeMico32 Tutorial
- LatticeMico32 Processor Reference Manual
- LatticeMico32 Tri-Speed Ethernet Media Access Controller data sheet, accessible through the LatticeMico32 MSB interface
- LatticeMico32 MSB and C/C++ online Help, accessible through the LatticeMico32 System Help
- ECP5 Versa Development Board User Guide (FPGA-EB-02021)
- ECP5-5G Versa Development Board User Guide (FPGA-EB-02048)

Appendix A. Alternate Network Cable Setup

If you do not have access to a PC with a gigabit-capable network card, you can use a gigabit switch, such as Netgear's ProSafe 5 Port Gigabit Switch (Model GS105) to perform the demonstration.

Figure A.1 shows the cable connectivity when using a gigabit switch.

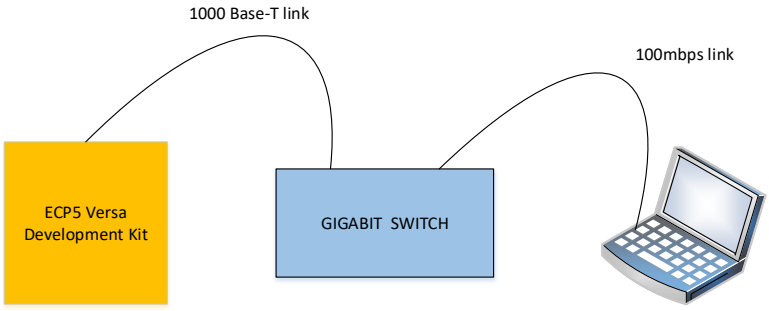

Figure A.1. Cable Connectivity for a Gigabit Switch

If you also do not have access to a gigabit switch, you can connect the ECP5 Versa Evaluation Board to a PC capable of providing 100 megabit-per-second connectivity or to a hub or switch capable of providing 100 megabit-per-second network connectivity.

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

Revision History

Revision 1.1, August 2022

Section	Change Summary
All	• Changed title from “LatticeMico32 Tri-Speed Ethernet MAC Gigabit Demo for the ECP5 and ECP5-5G Versa Evaluation Boards” to “LatticeMico32 Tri-Speed Ethernet MAC Gigabit Demo for the ECP5/ECP5-5G Versa Development Kit”. • Minor changes in formatting and style.
Introduction	Changed from “Lattice ECP5-45K FPGA” to “ECP5-45k LUTs FPGA” in Figure 1.1.
The Demonstration	Changed from “Lattice ECP5-45K FPGA” to “ECP5-45k LUTs FPGA” in Figure 3.11.

Revision 1.0, October 2016

Section	Change Summary
All	Initial release.

www.latticesemi.com