XAPP546 (v1.0) December 14, 2004

High Performance TCP/IP on Xilinx FPGA Devices Using the Treck Embedded TCP/IP Stack

Author: Satish Narayanaswamy

Summary

TCP/IP is a popular communications protocol software stack that allows reliable data communications between two hosts. Most people already use TCP/IP everyday to check E-mail, browse the Web, or transfer files using PCs. TCP/IP is being used more and more in the embedded world as well. Treck, Inc. is a leading provider of embedded TCP/IP stacks that allow Xilinx FPGAs to communicate in a wide range of networking environments. Treck's dual Ipv4/Ipv6 TCP/IP stack provides Ipv4 functionality today and allows a Xilinx FPGA to support Ipv6 networks of the future. Treck, Inc. also provides optional protocols, such as an embedded Web Server, FTP, IPSEC, DHCP, and more, to enhance the functionality of Xilinx FPGAs.

This document describes how to get started using the Treck TCP/IP stack using the Xilinx EDK tools. An evaluation version of the Treck TCP/IP stack is included as part of the application note. An example TCP application uses the Treck TCP/IP stack to send TCP data over Gigabit Ethernet on the Virtex-II Pro ™ ML300 Development Board to a remote PC-based server.

Introduction

The Treck TCP/IP stack offers a high performance TCP/IP software solution that can be used with the PowerPC™ 405 processor inside the Virtex-II Pro series of Xilinx FPGAs.

Some features of the Treck TCP/IP stack include:

- Zero-copy send and receive, which help to deliver maximum throughput for bridging applications
- Jumbo frames support, in the case of Gigabit Ethernet devices
- TCP checksum offload support for devices that support TCP checksum offload in hardware
- Fully RFC-compliant TCP/IP stack for maximum interoperability
- Standard sockets interface API

The Treck TCP/IP stack can be used with or without any operating system software. This application note discusses the use of the Treck TCP/IP stack on a standalone system (without an operating system)

This application note provides the Treck library as a binary file for evaluation purposes. The Treck library allows full functionality of the stack for a limited period of time before it times out and requires a restart of the system to continue evaluation.

Contact Treck at www.treck.com for information on purchasing the sources for the Treck TCP/IP stack. An example TCP client and server application is also available as part of this application note. Xilinx EDK tools are used to compile and link the client application with the Treck library to create a complete TCP/IP application for the ML300 board.

The client application uses Treck TCP/IP on the ML300 board to transmit TCP data to a remote PC. The server application running on the PC prints the TCP throughput every second on the console. The sources, as well as Windows and Linux binaries, are included for the server application.

Treck TCP/IP on the GSRD System

The Gigabit System Reference Design (GSRD) system enables high performance TCP/IP termination using a high performance multi-port memory controller, a Gigabit Ethernet MAC that incorporates TCP checksum offload capability in hardware, and a Communications Direct Memory Access Controller (CDMAC) that supports scatter-gather DMA operations.

Refer to XAPP535 and XAPP536 for more information on GSRD (http://www.xilinx.com/gsrd) and to download the design files.

The Treck TCP/IP stack has been successfully used on the GSRD system on a Virtex-II Pro ML300 development platform. Throughput rates of 785 Mb/s have been achieved using the Treck TCP/IP stack on this architecture. This section has step-by-step instructions for using the Treck TCP/IP stack on a standalone system using the GSRD design files.

Hardware Requirements

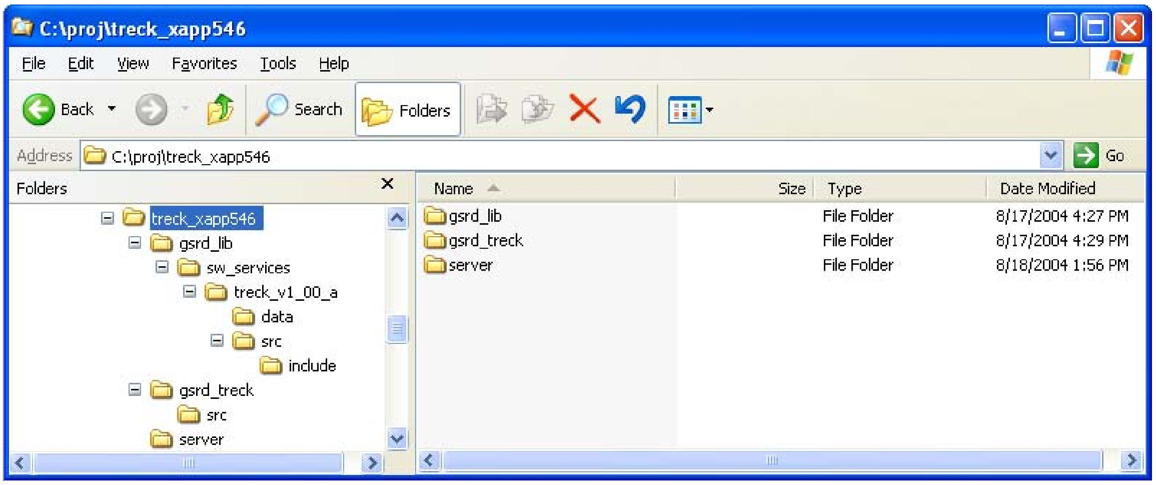
- ML300 board and programming cable, RS-232 serial cable, and Fiber Gigabit Ethernet cable
- GSRD design
- PC with Fiber Gigabit Ethernet card and RS-232 port

Software Requirements

- Xilinx EDK and ISE tools. At this time, the tools used to build the hardware and stack are EDK 6.3, and ISE 6.3.

The Treck software files (available as a separate download) have the following directory structure (Figure 1):

- The **gsrd_lib** directory contains the evaluation binary of the Treck TCP/IP stack.
- The **gsrd_treck** directory contains an example application that transmits TCP data via the Gigabit Ethernet peripheral using the GSRD design.
- The **server** directory contains sources and binaries for the server application to be run on a PC.



Name	Size	Type	Date Modified
gsrd_lib		File Folder	8/17/2004 4:27 PM
gsrd_treck		File Folder	8/17/2004 4:29 PM
server		File Folder	8/18/2004 1:56 PM

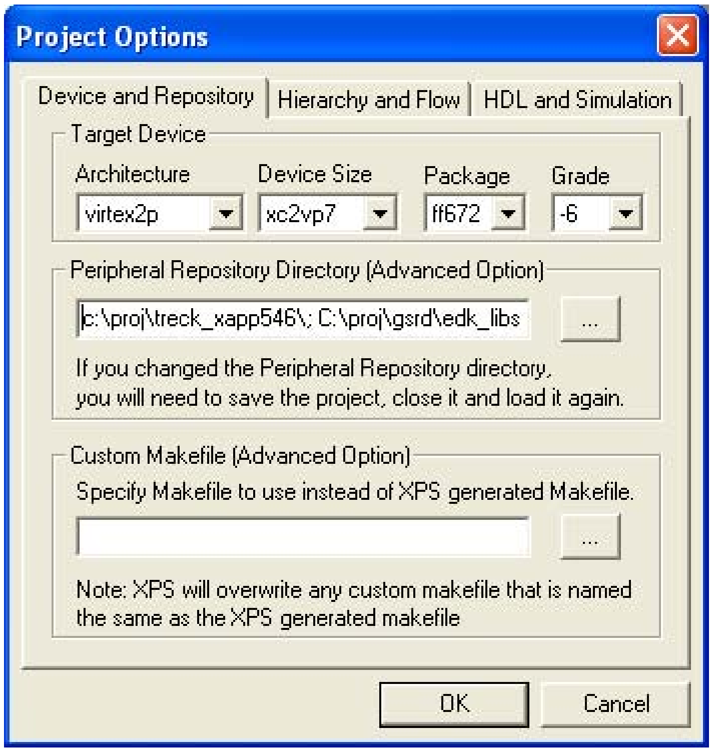
Figure 1: **File Directory Structure**

Compiling the Treck Stack and Application Using the GSRD Design

Follow these steps:

1. Download the GSRD design from the Xilinx website (http://www.xilinx.com/gsrd). Unzip the zip file into a local directory on the PC, for instance, `c:\proj\`. The GSRD zip file creates a new directory called `gsrd` under `c:\proj`.
2. Download the Treck related files (available as a download as part of this application note) and unzip them in a directory on the local PC, for instance `C:\proj`. The Treck zip file creates a new directory called `treck_xapp546` under `c:\proj`.
3. Double-click on the `c:\proj\gsrd\projects\ml300_gemac_tft\system.xmp` file to launch the XPS application (errors display during the first launch of the GSRD projects, which can be ignored for now.)
4. Under Options-Project Options (Figure 2), set up the peripheral repository directory path to point to the GSRD EDK libraries, as well as the path to the Treck software files. Use a semi-colon as a delimiter for these two paths.

   `(c:\proj\treck_xapp546\; C:\proj\gsrd\edk_libs\)`


Project Options

Device and Repository | Hierarchy and Flow | HDL and Simulation

Target Device

Architecture	Device Size	Package	Grade
virtex2p	xc2vp7	ff672	-6

Peripheral Repository Directory (Advanced Option)

c:\proj\treck_xapp546\; C:\proj\gsrd\edk_libs

If you changed the Peripheral Repository directory, you will need to save the project, close it and load it again.

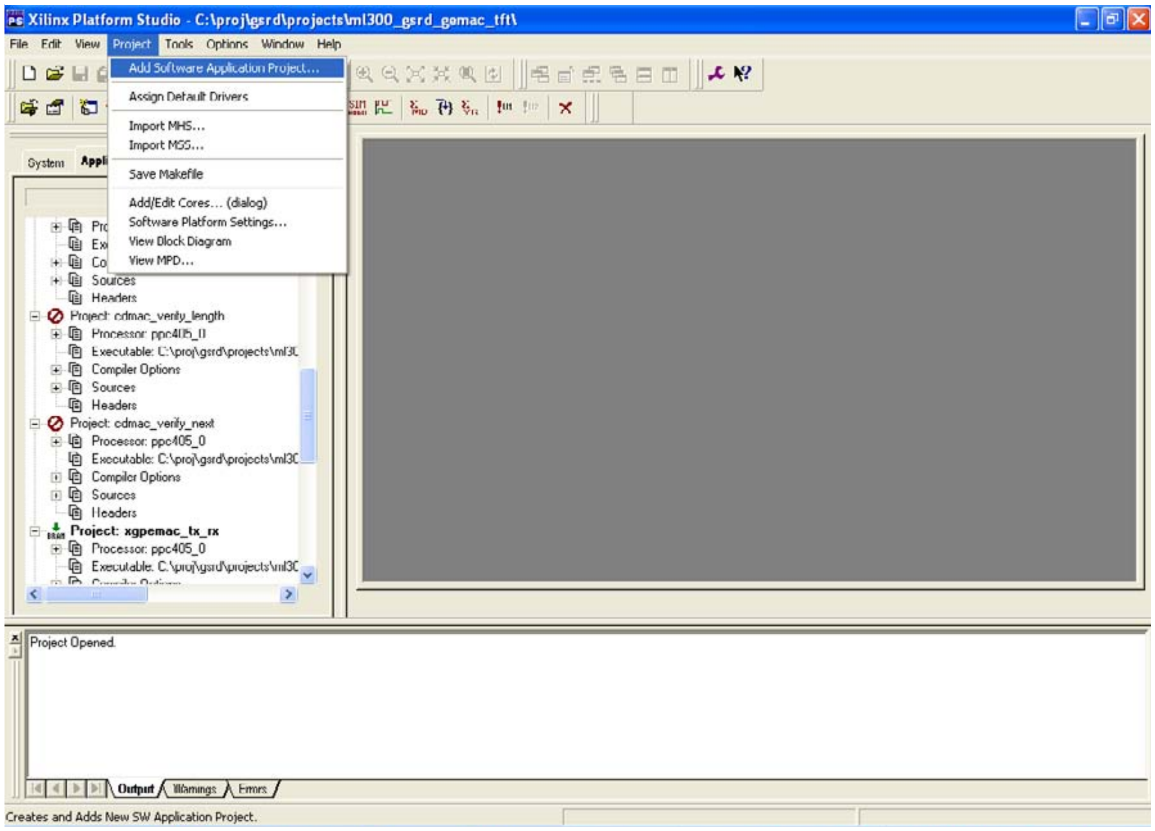
Custom Makefile (Advanced Option)

Specify Makefile to use instead of XPS generated Makefile.

Note: XPS will overwrite any custom makefile that is named the same as the XPS generated makefile

Figure 2: **Project Options**

5. A prompt box confirms your new settings. Click on OK to accept the changes.
6. Close the project (File-Close Project) and re-open the same project (File-Recent Projects-`C:\proj\gsrd\projects\ml300_gsrd_gemac_tft\system.xmp`). This causes the above changes to take effect. No errors should be displayed during project startup this time.
7. Click on Project-Add software application project (Figure 3) to add the Treck application to the project.

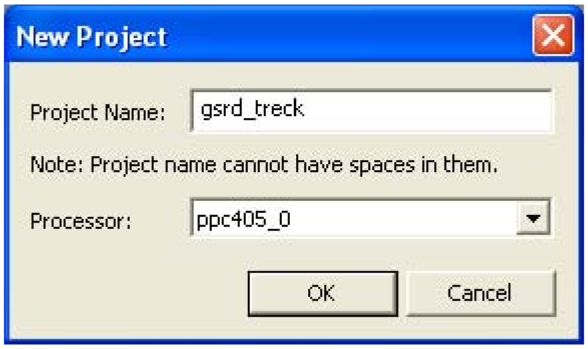

Xilinx Platform Studio - C:\proj\gsrd\projects\ml300_gsrd_gemac_tft\
File Edit View Project Tools Options Window Help
Add Software Application Project...
Assign Default Drivers
Import MHS...
Import MSS...
Save Makefile
Add/Edit Cores... (dialog)
Software Platform Settings...
View Block Diagram
View MPD...
System
Project: cdmac_verify_length
Processor: ppc405_0
Compiler Options
Sources
Headers
Project: cdmac_verify_next
Project: xgpemac_tx_rx
Output
Warnings
Errors

Project Opened.

Creates and Adds New SW Application Project.

Figure 3: **Add Software Application Project**

8. This launches a "New Project" dialog box (Figure 4). Fill in the name of the Treck application as `gsrd_treck`. Make sure the application is created for the PowerPC processor (ppc405_0) and click on OK.



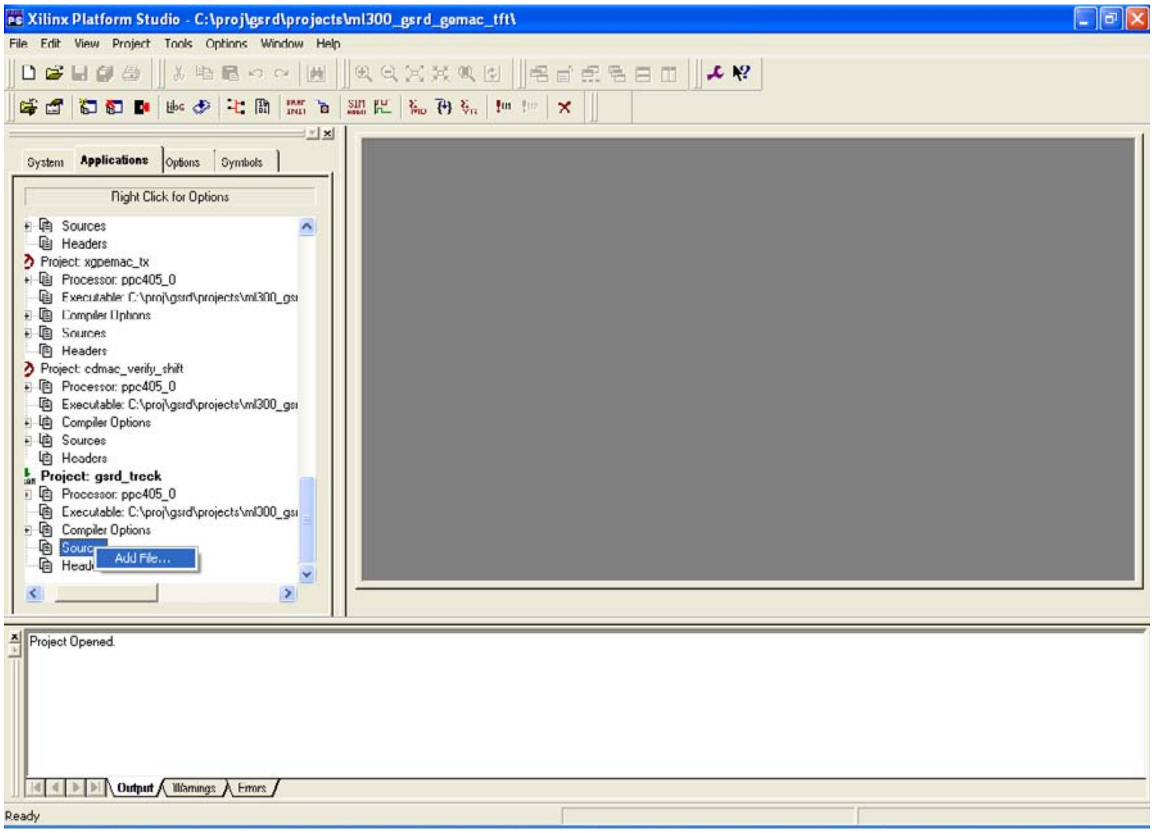
Project Name: gsrd_treck

Note: Project name cannot have spaces in them.

Processor: ppc405_0

Figure 4: **New Project**

9. Right-click on Sources for the newly created `gsrd_treck` software project and choose "Add File" (Figure 5).

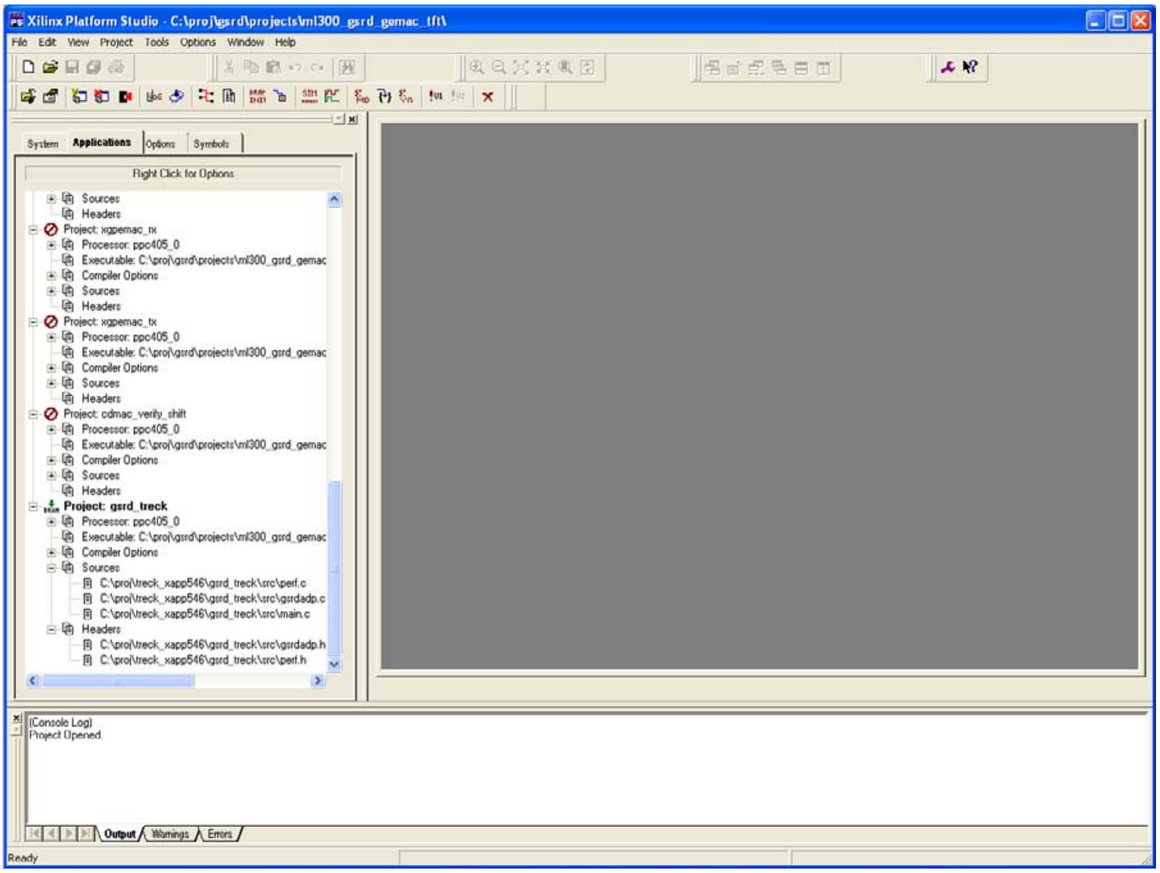


Project Opened.

Figure 5: **Add a File**

10. Browse the Treck software sources (`c:\proj\treck_xapp546\gsrd_treck\src directory`) and add `perf.c`, `gsrdadp.c`, and `main.c` to the software project. (TIP: Hold down the CTRL key to select multiple files within the Browse window.)
11. Similarly, right click on the "Headers" option under the `gsrd_treck` project and choose "Add file". Add `gsrdadp.h` and `perf.h` to the list of header files in this project.

At the end of this step, the project should look like the following screenshot (Figure 6).


Xilinx Platform Studio - C:\proj\gsrd\projects\ml300_gsrd_gemac_tft\
File Edit View Project Tools Options Window Help
System Applications Options Symbols
Right Click for Options
Output Warnings Errors
Ready

```
   Sources
   Headers
Project: xgpemac_rx
   Processor: ppc405_0
   Executable: C:\proj\gsrd\projects\ml300_gsrd_gemac
   Compiler Options
   Sources
   Headers
Project: xgpemac_tx
   Processor: ppc405_0
   Executable: C:\proj\gsrd\projects\ml300_gsrd_gemac
   Compiler Options
   Sources
   Headers
Project: cdmac_verify_shift
   Processor: ppc405_0
   Executable: C:\proj\gsrd\projects\ml300_gsrd_gemac
   Compiler Options
   Sources
   Headers
Project: gsrd_treck
   Processor: ppc405_0
   Executable: C:\proj\gsrd\projects\ml300_gsrd_gemac
   Compiler Options
   Sources
      C:\proj\treck_xapp546\gsrd_treck\src\perf.c
      C:\proj\treck_xapp546\gsrd_treck\src\gsrdadp.c
      C:\proj\treck_xapp546\gsrd_treck\src\main.c
   Headers
      C:\proj\treck_xapp546\gsrd_treck\src\gsrdadp.h
      C:\proj\treck_xapp546\gsrd_treck\src\perf.h
```

```
(Console Log)
Project Opened
```

Figure 6: **Project Screenshot**

12. Set the compiler options for the project Right click on Compiler Options for the `gsrd_treck` project and choose “Set compiler options” (Figure 7).

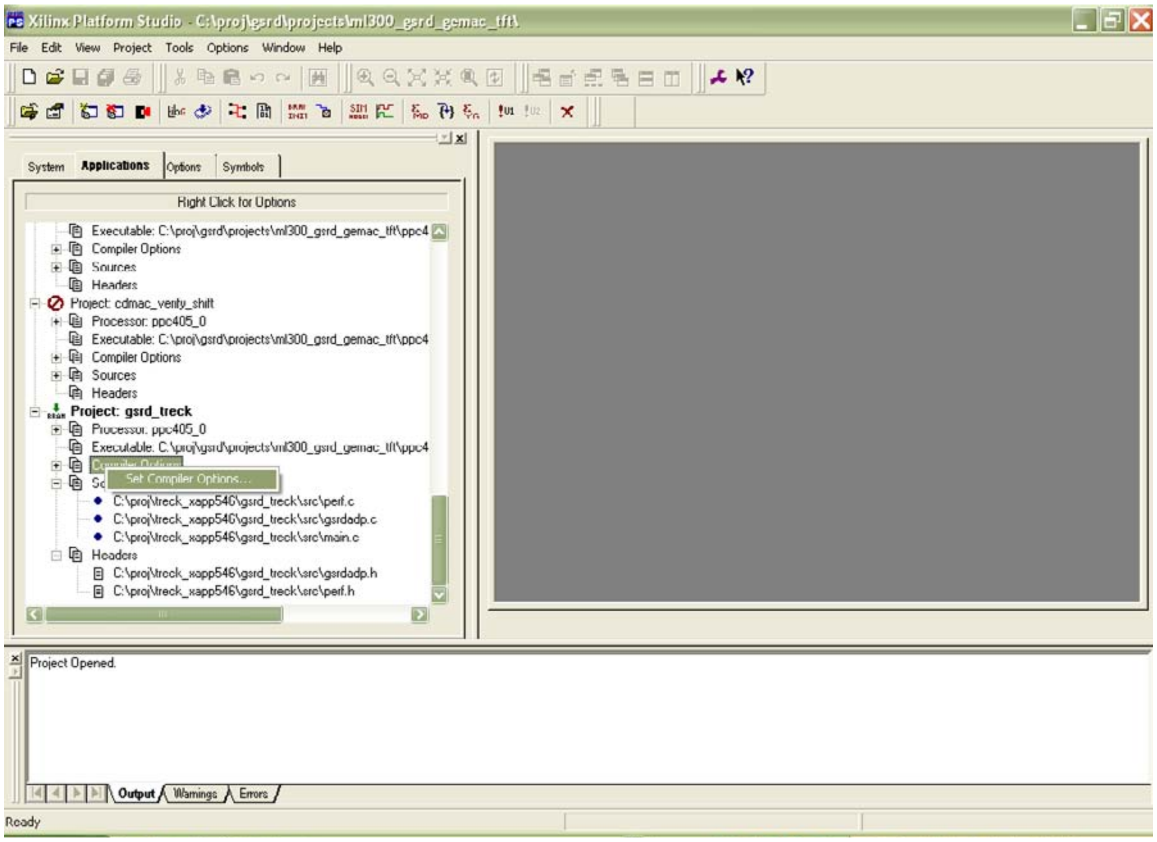
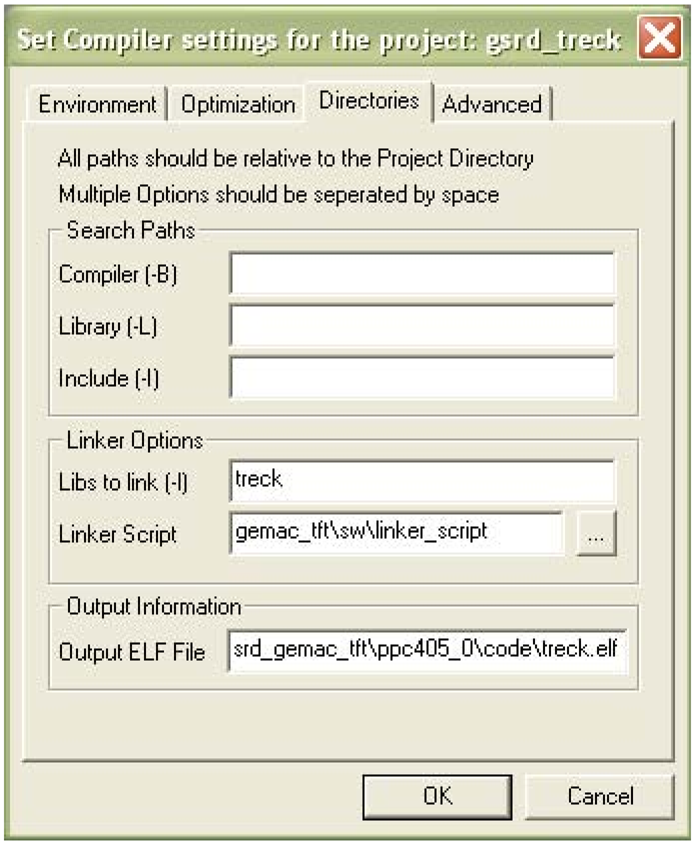

Xilinx Platform Studio - C:\proj\gsrd\projects\ml300_gsrd_gemac_tft\
File Edit View Project Tools Options Window Help
System Applications Options Symbols
Right Click for Options
Executable: C:\proj\gsrd\projects\ml300_gsrd_gemac_tft\ppc4
Compiler Options
Sources
Headers
Project: cdmac_verify_shift
Processor: ppc405_0
Executable: C:\proj\gsrd\projects\ml300_gsrd_gemac_tft\ppc4
Compiler Options
Sources
Headers
Project: gsrd_treck
Processor: ppc405_0
Executable: C:\proj\gsrd\projects\ml300_gsrd_gemac_tft\ppc4
Compiler Options
Set Compiler Options...
C:\proj\treck_xapp546\gsrd_treck\src\perf.c
C:\proj\treck_xapp546\gsrd_treck\src\gsrdadp.c
C:\proj\treck_xapp546\gsrd_treck\src\main.c
Headers
C:\proj\treck_xapp546\gsrd_treck\src\gsrdadp.h
C:\proj\treck_xapp546\gsrd_treck\src\perf.h
Project Opened.
Output Warnings Errors
Ready

Figure 7: **Set Compiler Options**

13. The “Set Compiler Options” window has a Directories tab where the linker script for the project can be set. Browse to: `c:\proj\gsrd\projects\ml300_gsrd_gemac_tft\sw` directory and choose `linker_script` as the linker script for the project.

    Also, set the linker option “Libs to link (-l)” to “treck”. This includes the Treck library in the linking process (Figure 8).

    The “output ELF file” can optionally be changed to: `c:\proj\gsrd\projects\ml300_gsrd_gemac_tft\ppc405_0\code\treck.elf`



All paths should be relative to the Project Directory
Multiple Options should be seperated by space

Search Paths

Compiler (-B)

Library (-L)

Include (-I)

Linker Options

Libs to link (-l) treck

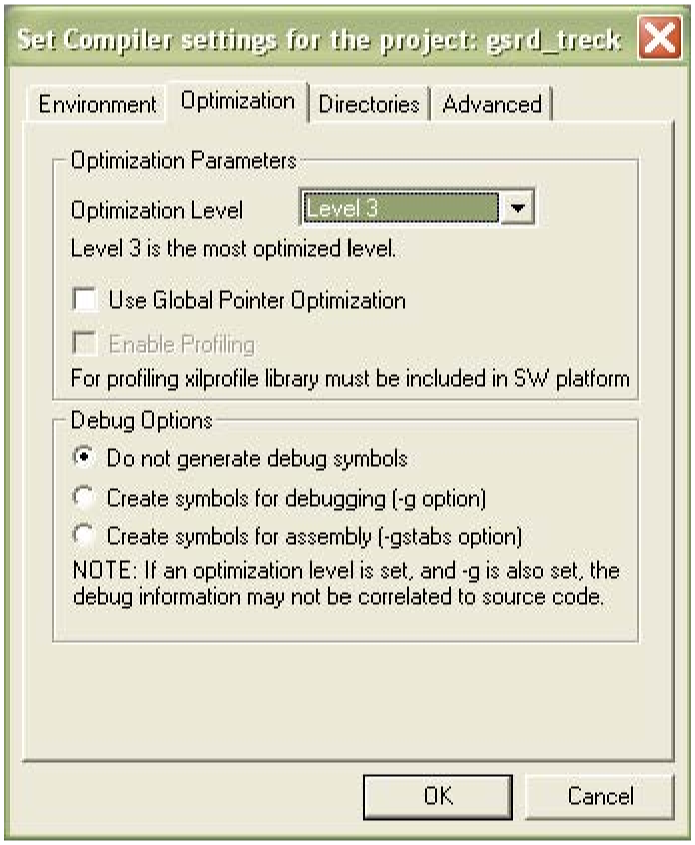
Linker Script gemac_tft\sw\linker_script

Output Information

Output ELF File srd_gemac_tft\ppc405_0\code\treck.elf

Figure 8: **Set the Linker Options**

14. Under the Optimization tab (Figure 9), optionally change the Optimization level to "Level 3" and choose "Do not generate debug symbols" under debug options for maximum performance.
15. Click on the OK button.
16. Import the MSS file for the GSRD Treck project. Under the Project menu, choose the “Import MSS” button. Browse to the `C:\proj\treck_xapp564\gsrd_treck` directory and choose the `system_treck.mss` file.


Set Compiler settings for the project: gsrd_treck

Optimization Parameters

Optimization Level: Level 3

Level 3 is the most optimized level.

[ ] Use Global Pointer Optimization

[ ] Enable Profiling

For profiling xilprofile library must be included in SW platform

Debug Options

[✓ Do not generate debug symbols]

Create symbols for debugging (-g option)

Create symbols for assembly (-gstabs option)

NOTE: If an optimization level is set, and -g is also set, the debug information may not be correlated to source code.

Figure 9: **Set the Optimization Level**

Test Configuration Options

Change the IP address/subnet mask if needed for the Treck initialization of the Gigabit Ethernet interface on the ML300 board. Double-click on the `perf.h` header file for the `gsrd_treck` project. The relevant lines to configure the network interface settings are:

```
#define TM_OUR_IP_ADDRESS      "192.168.42.101"
#define TM_OUR_IP_NETMASK      "255.255.255.0"
#define TM_GATEWAY_IP_ADDRESS  "192.168.42.100"
```

In addition, the application sends TCP data to a remote server which prints throughput information every second. The server IP address is configured through this line in `perf.h`.

```
#define SERV_HOST_ADDR  "192.168.42.100"
```

Other configuration options in `perf.h` include:

- JUMBO_FRAME_TEST for enabling 9000 byte jumbo Ethernet frames for maximum performance (note that the remote PC server and any intermediate switches in the network must also support jumbo frames for this to work).
- ZEROCOPY_TEST enables maximum performance by using the Treck zero-copy features. The user buffer is transmitted directly over Ethernet via a DMA operation, bypassing an intermediate buffer copy into Treck's local buffers. This constant is enabled by default.

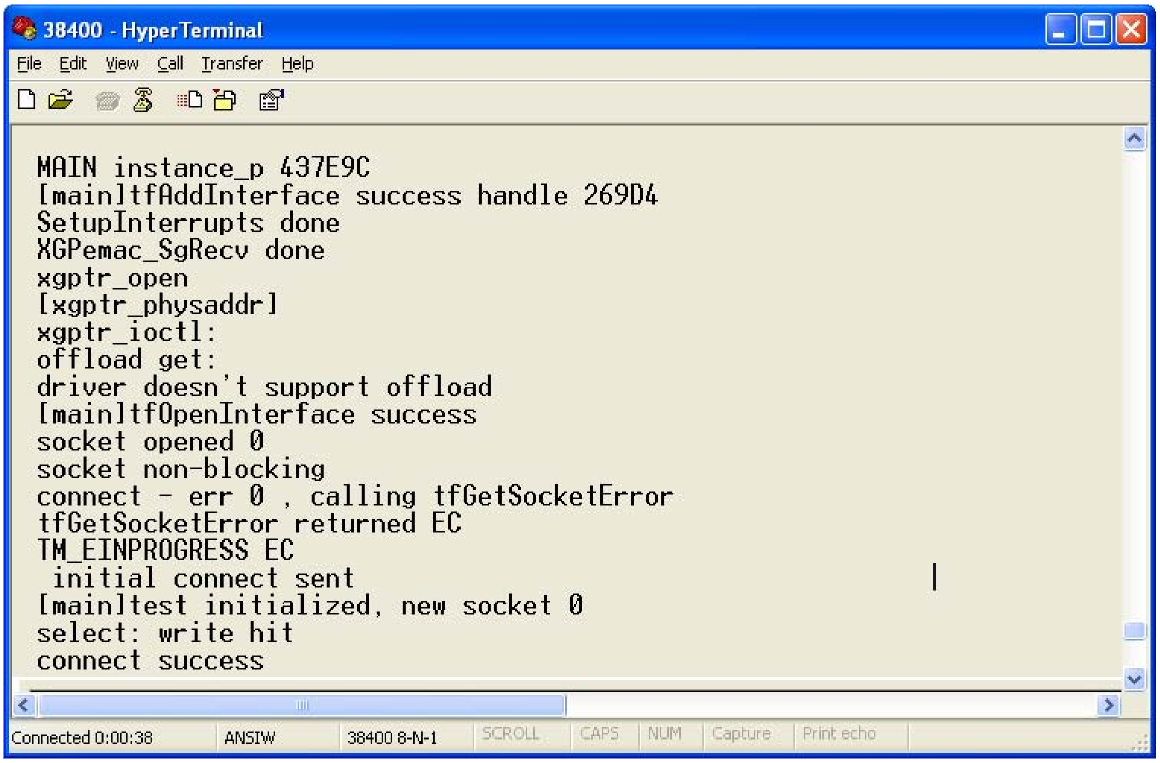
Save the header file after making any changes.

Building the Treck Library and the Software Applications

1. Under the Tools menu, choose "`Generate libraries and BSP`". Once this is completed, choose `Tools-Build all User Applications` to generate the executable `.elf` file in the `gsrd_treck` directory.
2. Download the Treck application to the ML300 board. This application initializes the Treck TCP/IP stack, initializes the Gigabit Ethernet interface with the IP address above. Refer to the GSRD documentation for information on how to download to the ML300 board.
3. Run the server application on a PC (running Windows or Linux) before running the Treck application on the ML300 board.

   The `serv.exe` application under the `c:\treck_xapp546\server` directory is built for Windows (it requires the user to download and install `cygwin` application www.cygwin.com). The `linux_serv` executable is for a Linux 2.4 kernel. The source for these two executables is `serv.c`. The command line to build the server application on a lInux system is "`gcc -o linux_serv -lpthread serv.c`".

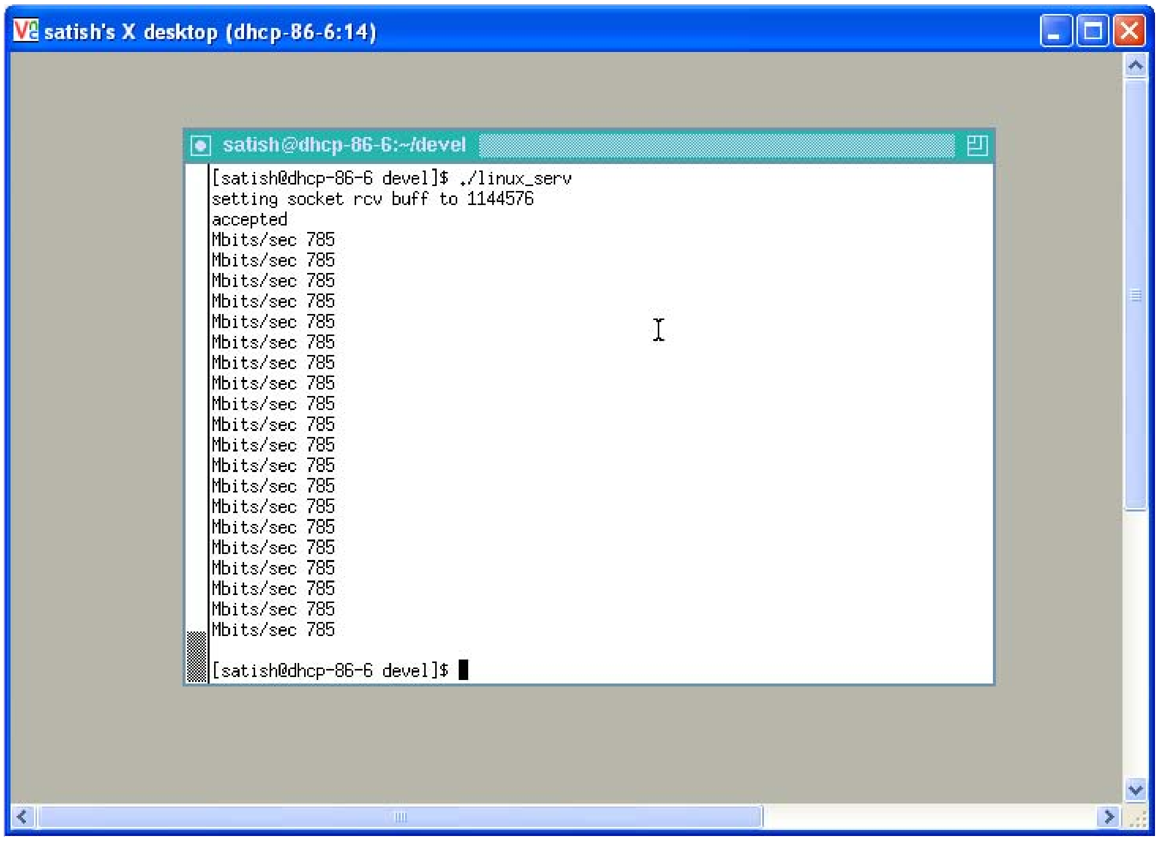
   The ML300 console (Figure 10) should display some debug messages on initialization. If the client is able to connect to the server, a "connect success" message displays on the console. No further output displays on the ML300 console, but the server screen should display the throughput information every second.



```
MAIN instance_p 437E9C
[main]tfAddInterface success handle 269D4
SetupInterrupts done
XGPemac_SgRecv done
xgptr_open
[xgptr_physaddr]
xgptr_ioctl:
offload get:
driver doesn't support offload
[main]tfOpenInterface success
socket opened 0
socket non-blocking
connect - err 0 , calling tfGetSocketError
tfGetSocketError returned EC
TM_EINPROGRESS EC
 initial connect sent
[main]test initialized, new socket 0
select: write hit
connect success
```

Figure 10: **Screenshot of ML300 Console**

Also see screenshot (Figure 11) of Linux server (P-III 733 Mhz, 512MB RAM, SysKonnect Gigabit Ethernet card) running the `linux_serv` application.



```
[satish@dhcp-86-6 devel]$ ./linux_serv
setting socket rcv buff to 1144576
accepted
Mbits/sec 785
Mbits/sec 785
Mbits/sec 785
Mbits/sec 785
Mbits/sec 785
Mbits/sec 785
Mbits/sec 785
Mbits/sec 785
Mbits/sec 785
Mbits/sec 785
Mbits/sec 785
Mbits/sec 785
Mbits/sec 785
Mbits/sec 785
Mbits/sec 785
Mbits/sec 785
Mbits/sec 785
Mbits/sec 785
Mbits/sec 785
Mbits/sec 785

[satish@dhcp-86-6 devel]$ 
```

Figure 11: **Linux Server**

Performance

The example application uses the Treck zero-copy feature and the checksum offload feature with scatter-gather DMA operations in the GSRD hardware design to obtain the following performance numbers.

Frame Size	Remote Host TCP Receive Window	Transmit Performance (Mb/s)
1500	1 Mbyte	425
9000	1 Mbyte	785

An important factor in determining TCP throughput is tuning the receiver system's TCP window size (size of the receive TCP socket buffer of the server application running on Windows/Linux).

The attached server program sets the receiving socket buffer size to be 1 Mbyte. The Linux or Windows PC system limits might have to be tuned to enforce the increased socket buffer sizes, even if the application reports that it has successfully set the socket buffer size to 1 Mbyte.

Refer to `http://www.psc.edu/networking/perf_tune.html` for information on tuning the server system for a larger window size.

A Linux shell script `perf.sh` is available as part of the application note which configures a Linux 2.4 system to enable large socket buffers.

Reference Design Files

The reference design files can be downloaded from the Xilinx website: xapp546.zip

Conclusion

The Treck embedded TCP/IP stack is well-suited for TCP/IP applications running on Xilinx FPGAs. Its support of zero-copy applications and checksum offload in hardware is utilized in a high-performance architecture like GSRD. Treck also offers different protocols and applications like IPSEC, IPV6, HTTP, and Telnet. The combination of the Treck TCP/IP stack and the flexible Xilinx FPGA hardware platform offer an ideal solution for TCP/IP termination at high data rates.

References

- Gigabit System Reference Design (GSRD)
  www.xilinx.com/gsrd
  http://www.xilinx.com/bvdocs/appnotes/xapp536.pdf
- High Performance Multi-Ported Memory Controller Application Note
  http://www.xilinx.com/bvdocs/appnotes/xapp535.pdf
- Treck TCP/IP Stack
  http://www.treck.com
- Treck TCP/IP Stack API Documentation
  http://www.treck.com/trusr40c.pdf
- Enabling High-Performance Data Transfers
  http://www.psc.edu/networking/perf_tune.htm
- Cygwin Linux Emulation Software For Windows
  /http://www.cygwin.com

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
12/14/04	1.0	Initial Xilinx release.