XAPP941 (v1.1) June 15, 2007

Reference System: PLB Tri-Mode Ethernet MAC

Author: Robert McGee and Norbert Melnikov

Abstract

This application note describes a reference system illustrating how to build an embedded PowerPC™ system using the Virtex™-4 PLB Tri-Mode Ethernet Media Access Controller (PLB_TEMAC). This reference system has the PLB_TEMAC configured to use Scatter/Gather DMA and support a Gigabit Media Independent Interface (GMII) PHY. Furthermore, it enables such performance enhancing features as hardware data realignment and checksum offloading. This reference system includes two software test suites that provide examples of how to measure performance and verify functionality of the PLB_TEMAC IP core. These applications are a simple standalone Echo Server application that loops back received data and a VxWorks Board Support Package (BSP) that includes functionality to test the throughput performance of the system. The reference system is targeted for the ML405 evaluation platform.

Included Systems

Included with this application note is one reference system:

- www.xilinx.com/bvdocs/appnotes/xapp941.zip

Introduction

Using Ethernet MACs in embedded microprocessor systems is becoming increasingly prevalent. In fact, the usage is so high that Xilinx has integrated an Ethernet MAC into the fabric of the Virtex-4 FX family of FPGAs. This Ethernet MAC is capable of transmitting and receiving data at 10, 100, and 1000 Mbps and supports interfacing to MII, GMII, Reduced GMII (RGMII), Serial GMII (SGMII), and 1000BASE-X, all while consuming no FPGA resources since the Ethernet MAC is embedded in each Virtex-4 FX device. Xilinx also provides a parameterizable Processor Local Bus (PLB) interface that provides a connection to the hard TEMAC. This PLB_TEMAC core is complete with variable size FIFOs and a Scatter/Gather DMA engine to make building embedded PowerPC systems much easier. The reference system described in this application note has the PLB_TEMAC configured to use Scatter/Gather DMA, include receive (Rx) and transmit (Tx) hardware Data Realignment Engine (DRE) and Checksum Offloading (CSO), and interface to the PHY through GMII.

Hardware and Software Requirements

The hardware and software requirements are:

- Xilinx ML405 Development Board
- Xilinx Platform USB Cable or Parallel IV Cable
- RS232 Cable
- Serial Communications Utility Program (e.g. HyperTerminal)
- Xilinx Platform Studio 9.1.01i
- Xilinx ISE 9.1.03i
- Cygwin 1.5.19-4 or higher (if running VxWorks performance measurement suite)
- VxWorks 5.5 or higher (if running VxWorks performance measurement suite)

Reference System Specifics

This ml405_ppc_plb_temac reference system is composed of two separate EDK projects. The first project, `vxworks_performance.xmp`, is set up for VxWorks so that the VxWorks Performance Test Suite can be executed. The second project, `system.xmp`, is a project that uses standalone software to run the Echo Server application. Both of these EDK projects use the same hardware configuration.

This reference system contains only the IP cores necessary to provide an example of how to set up PLB_TEMAC, how to verify that the core is operational and how to measure TCP/UDP throughput using VxWorks. In addition to the PowerPC processor and PLB_TEMAC, this system includes DDR and Block RAM memory on the PLB and a UART and an interrupt controller on the OPB.The PLB_TEMAC PHY interface signals are connected to the tri-speed PHY on the ML405 board. See Figure 1 for the block diagram and Table 1 for the address map of this system.

Block Diagram

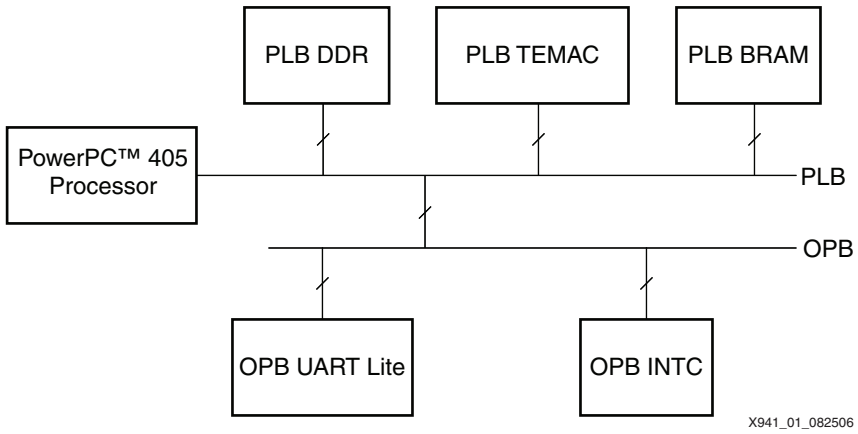

Figure 1: **Reference System Block Diagram**

Address Map

Table 1: **Reference System Address Map**

Peripheral	Instance	Base Address	High Address
PLB_DDR	DDR_SDRAM_64Mx32	0x00000000	0x03FFFFFF
OPB_UARTLite	RS232_Uart	0x40600000	0x4060FFFF
OPB_INTC	opb_intc_0	0x41200000	0x4120FFFF
PLB_TEMAC	plb_temac_0	0x80000000	0x8000FFFF
PLB_BRAM	plb_bram_if_cntrl_1	0xFFFFC000	0xFFFFFFFF

Configuring the PLB_TEMAC Core

The ml405_ppc_plb_temac reference system configures the PLB_TEMAC to use Scatter/Gather DMA and turns on DRE and CSO on both the receive and transmit channels. This reference system uses only one PLB_TEMAC instance, even though the core allows for connecting up two instances of the TEMAC. The reason for this is that each processor block in Virtex-4 FX devices actually includes two Ethernet MACs, but they share a unified host interface. Therefore, in order to enable both the Ethernet MACs from the same processor block, it is necessary to set up the appropriate parameters. The host interface for the PLB_TEMAC in this reference system is controlled through the PLB rather than the DCR. Figure 2 shows a

complete setting of the PLB_TEMAC parameters in the ml405_ppc_plb_temac reference system.

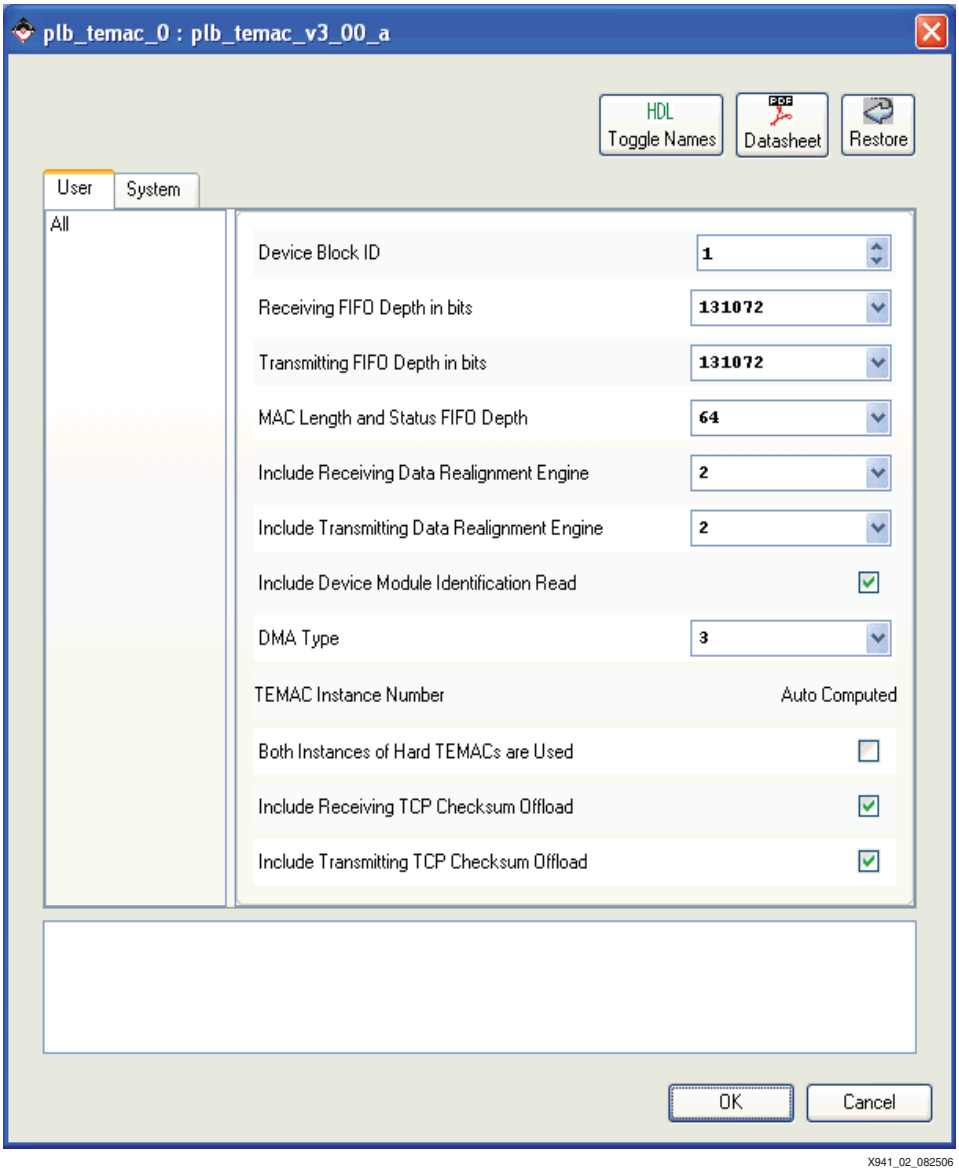

Figure 2: **PLB_TEMAC Parameters**

VxWorks Performance Test Suite

The intent of this software application is to allow users to run the network performance test suite for VxWorks. The user should be familiar with VxWorks and the C programming language. Some background knowledge of TCP/IP stacks is very helpful.

Introduction

This TCP/UDP test suite can be used to characterize the performance capabilities of the VxWorks TCP/IP and Xilinx driver stack running on a PPC based Virtex-4 system. The test suite is comprised of three parts: a client/server application on the target, a compatible client/server application on the host computer, and various host tools and scripts to automate data gathering. The target client/server application is Netperf version 2.1pl3 (www.netperf.org).

This application can measure Ethernet throughput and CPU utilization. There are no dependencies on hardware. The target system's Netperf application has been extended to use the faster zero copy network functionalities of VxWorks.

Directory Layout

The test suite components are laid out as follows:

`ml405_ppc_plb_temac` - VxWorks front-end source code. Contains the part of the target application that implements the menu system, PHY-specific source code, MAC-specific source code, and other utilities.

`ml405_ppc_plb_temac/netperf-2.1pl3-vxworks-revb` - Contains the VxWorks port of Netperf

`ml405_ppc_plb_temac/cygwin/bin` - Contains host tools and scripts

`ml405_ppc_plb_temac/cygwin/src/netperf-2.1pl3-cygwin` - Contains source code for cygwin version of Netperf

`ml405_ppc_plb_temac/cygwin/src/rc` - Contains source code for remote control of target MAC settings and client startup

Host System Requirements (WinXP)

In order to test at Gigabit transfer rates, the host computer must be fast enough to handle line rate data transfers for TCP and UDP. The following items are also needed:

1) Installation of cygwin version 1.5.19-4 or higher

2) (optional) Installation of cygwin version of gcc to rebuild host tools

3) NIC card. If running gigabit rates, the card should include checksum and segmentation offload functionality and provide support for jumbo frames

4) Installation of WindRiver Tornado 2.2 (VxWorks 5.5), or Workbench 2.3 (VxWorks 6.x)

Target System Requirements (PPC405)

The ml405_ppc_plb_temac reference system already meets all the target system requirements, however, in general, the following features should be provided by the target hardware system to provide maximum performance:

1) Bursting must be enabled where possible for memory systems

2) PPC data and instruction caches must be included

3) The inclusion of MMU is not required

4) The inclusion of at least 16MB of external memory

5) The inclusion of at least 16KB of BRAM (for SGDMA capable systems)

Test Suite Design Notes for Target Side

VxWorks 5.5 and 6.1 are both supported by the test suite application transparently. Checksum offloading is not implemented for VxWorks 5.5. Only VxWorks 6.1 provides support for checksum offloading.

The target application consists of a front end that implements a menu system, MAC & PHY specific utilities, a remote control subsystem, plus the Netperf application. A VxWorks Board Support Package (BSP) is not included. The BSP can be generated by the EDK tools. Once the application and BSP have been integrated, the application, BSP, and VxWorks kernel are all linked into a single monolithic object.

Netperf

The tester interacts with the target by using various menus on the console serial port. The menus allow the tester to change MAC characteristics, start Netperf client-side tests, start and stop Netperf server-side tests, and display various networking statistics.

The Netperf port is based off the Netperf standard 2.1pl3 Unix release. This Netperf release was initially modified by Tom Pavel of the Stanford Linear Accelerator Center to implement the BSD TCP stream tests. Further porting work was done by Xilinx, Inc. by adding implementations of:

1) UDP send and receive tests

2) TCP/UDP zbuf socket API tests

3) UDP fastUDP API tests

4) CPU profiling

Due to the added features, this port of Netperf is not backwards compatible with the original 2.1pl3 version.

The Netperf application is built using a makefile. The executable itself is loaded by the operating system. Runtime support such as clearing the BSS section is managed by the operating system. This test suite's version of Netperf has BSS and data sections mixed with the kernel and application's BSS and data. At boot time, the operating system will set up all BSS and data sections properly and Netperf will execute normally the first time it is run. On subsequent runs, Netperf will fail because BSS and data sections contain old data. To overcome this limitation, a linker script is used when building Netperf. The BSS and data sections are collected in one place and marked so that they may be found at run time. A new section was created to contain a copy of the BSS and data for access during runtime. At startup, the application (not the operating system) will make a copy of the BSS and data. Each time Netperf is run the application will reset the BSS and data with the copy.

Startup

When the VxWorks kernel and the application are started, the configuration table entry for the MAC is scanned and a set of device capabilities is determined. Any request to change the device mode is validated against these capabilities.

After boot and application startup is complete, the network is in an unloaded state until one of the menu commands starts the Netperf client, starts the Netperf server, or starts the remote control subsystem.

VxWorks BSP

An EDK generated BSP can be the basis BSP for the test suite. Only minor modifications are required to integrate the test suite application into the BSP. For more information, see the BSP Setup and Image Compilation section of this application note

Test Suite Design Notes for Host Side

The host OS does not need to be WinXP. If Linux is desired, then the host tools will need to be recompiled and host scripts possibly adjusted for whatever shell is being used.

Netperf

A cygwin port of Netperf is provided. This is an unmodified port available on the Internet.

Scripts

Most of the scripts automate data gathering by invoking remote control commands to place the MAC into the desired mode and collecting and formatting test results.

Test Suite Design Notes for Remote Control

The remote control subsystem is part of the test suite that allows a host computer to take control of various aspects of the target system normally managed by user interaction with the console menus. This functionality allows all performance test measurements to be coordinated and automated from the host computer. The following actions can be controlled:

1) Enable/disable Tx and/or Rx DRE

2) Enable/disable Tx and/or Rx checksum offloading

3) Select a socket API (BSD, zbuf, FastUDP)

4) Change network MTU

5) Change SGDMA interrupt coalescing values

6) Instruct the target to run a Netperf client test to the host (for Tx performance testing) and return the results of that test

7) Instruct the target to start its netserver process (for Rx performance testing)

8) Instruct the target to stop its netserver process

Remote control works by setting up a UDP port on the target system to listen for commands from the host. When a command comes in, it is processed and the status is sent back to the host. Some commands may cause the entire target network to be reloaded. In this case, the host computer must wait long enough for the network to come back up before getting a response.

BSP Setup and Image Compilation

The BSP and application must be compiled from the command line. It is possible to make projects for Workbench 2.3 (VxWorks 6.1), but it requires creating complex subprojects and makefile extensions within the IDE.

Setting Up a Development Shell

To compile the BSP and test suite, the user will need to invoke a command shell that contains the correct environment. When using the VxWorks Development Shell, the environment is already set up. This shell can be started by choosing:

start->Wind River->VxWorks 6.1->VxWorks Development Shell

Setting up the Test Suite for the Hardware

The file `xps_user.h` contains constant declarations that may vary from system to system. The user must modify this file to ensure the application will build with their hardware and network topography.

Setting up and Compiling the VxWorks BSP

There are two files to modify for a BSP emitted from the EDK tools: `config.h` and `Makefile`.

In `config.h` insert the following code near the end of the file (hilighted in bold):

```
#include "ppc405_0.h"
#include "<path_to_project>/config_test.h"


#endif /* INCconfigh */


#if defined(PRJ_BUILD)
#include "prjParams.h"
```

```
#endif
```

where **<path_to_project>** is the absolute path of the VxWorks BSP directory.

This modification will override global constants and certain constants already in config.h. Constants overridden include enabling the data caches, greatly increasing the number of network buffers, and specifying the entry point of the application.

In `Makefile` insert the following lines (hilighted in bold):

```
#
#
# Xilinx Chip Support Package modifications end here


DEV_MAC_SRC = temac_200a.c
DEV_PHY_SRC = phy_ marvell_88e111.c
DEV_SRC_DEFINE = -DML300_PRODUCTION_REV1
PERF_DIR = h:/<path_to_perfcode>
BSP_DIR = $(subst ,/,$(GETCWD))
include $(PERF_DIR)/Makefile


## Only redefine make definitions above this point, or the
expansion of
## makefile target dependencies may be incorrect.
```

where **<path_to_perfcode>** is the absolute path to the root directory of where the performance suite source code resides.

The DEV_MAC_SRC and DEV_PHY_SRC constants inform the make system which MAC and PHY code to compile. This code must be present in the base directory of the test suite. DEV_SRC_DEFINE is optional and passes predefined constants into the compilation. This can be used, for example, to let the MAC or PHY source code know what board they are being run on.

Finally, to build the operating system and the application from the command shell:

```
make vxWorks
```

In order to remove all compiler generated files, run:

```
make clean
```

Host Computer Setup

Even on the fastest PC, gigabit line rate may not be achievable. Typical desktop PCs use a NIC which hangs off a 33Mhz 32 bit PCI bus. The maximum throughput on this PCI setup is 133 MBytes/sec. This bandwidth is shared amongst all devices on the PCI bus. Since a GigE NIC can consume 125MB per second on each of the Tx and Rx channels, the PCI bus becomes the bottleneck. A PC with a PCI-X, or PCI-Express will most likely not pose a bandwidth problem. Even when using regular PCI, it is still possible to do things to help performance such as terminating any unneeded processes, unplugging USB hardware and using the fastest settings available on the NIC.

Looking at the target's CPU utilization for a Rx UDP test is the best way of determining if the host PC is not up to the task. If the Rx throughput is not at line rate and the CPU utilization is

low and the target received nearly all UDP packets sent from the host, then the target is waiting on the PC.

Setting up the TCP/IP Address of the Host:

Typically the host computer will have two network cards: one for the normal work related connection, and another for a dedicated test network. The test network settings should be setup (for WinXP) as shown in Figure 3:

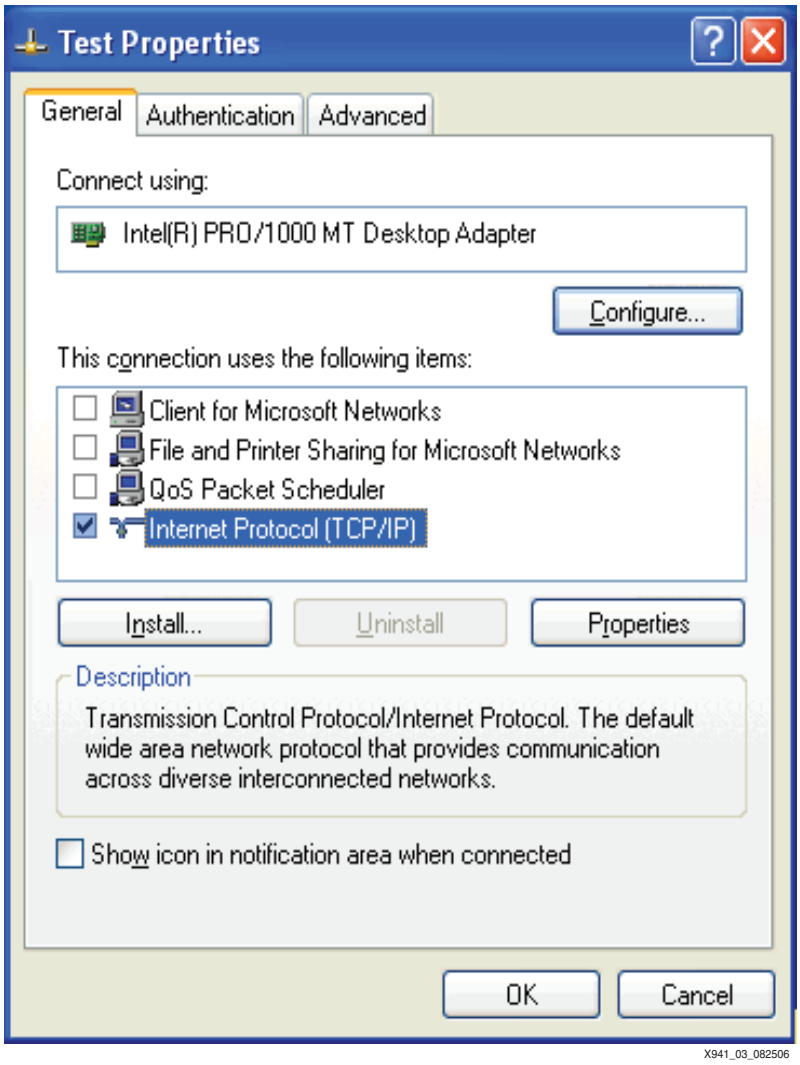

Figure 3: **Test Network Properties**

In the figure above, only select **Internet Protocol (TCP/IP)**. Unnecessary network traffic may result if other services are selected.

In the Properties for TCP/IP, only the IP address and mask need to be set up. Everything else can be turned off. See Figure 4.

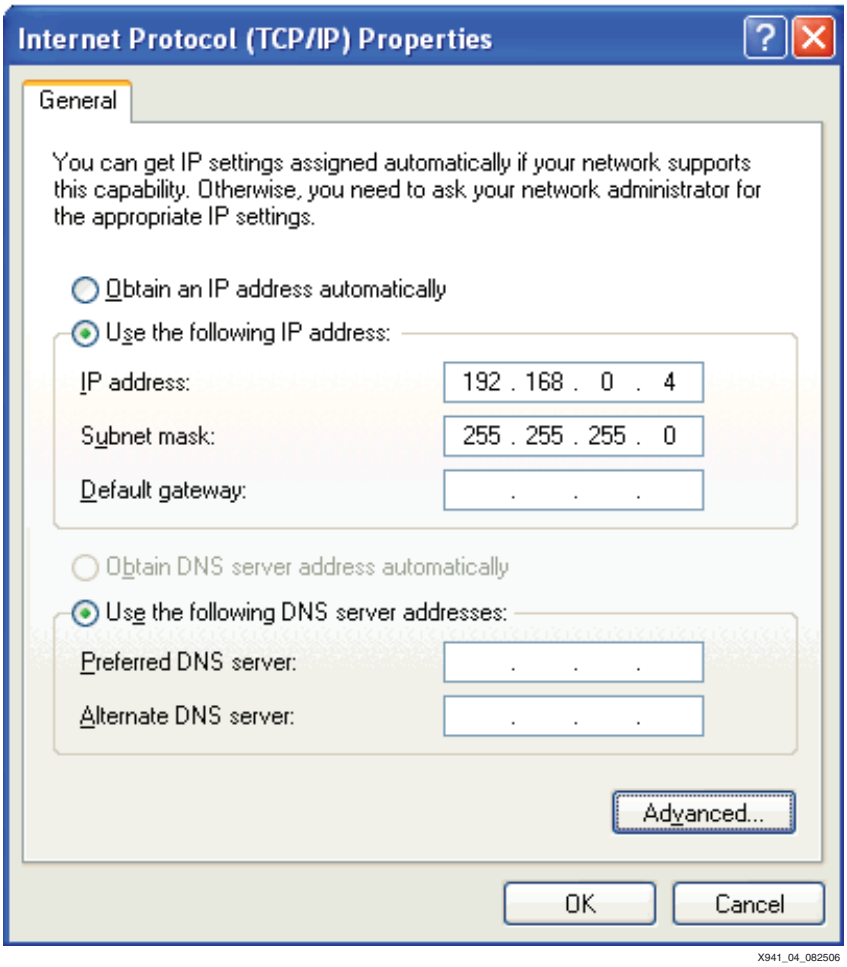

Figure 4: **TCP/IP Properties**

Note that the IP address should match what is specified in the target's test suite application `xps_user.h`:

```
#define HOST_IP_ADDRESS      "192.168.0.4"
```

Modifying the Registry to Achieve Improved Performance

This step is optional, however, it results in significantly improved performance.

The WinXP stack supports TCP window scaling protocols (RFC 1323) but it is not enabled by default. To enable this option, the registry must be modified. Registry can be opened for editing by entering **`regedit`** in:

start->Run

Create new REG_DWORD keys located at HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters. The following parameters must be set (see Figure 5):

- Tcp1323Opts = 3
- TcpWindowSize = 1057280
- GlobalMaxTcpWindowSize = 1057280

The PC must be rebooted for these new settings to take effect.

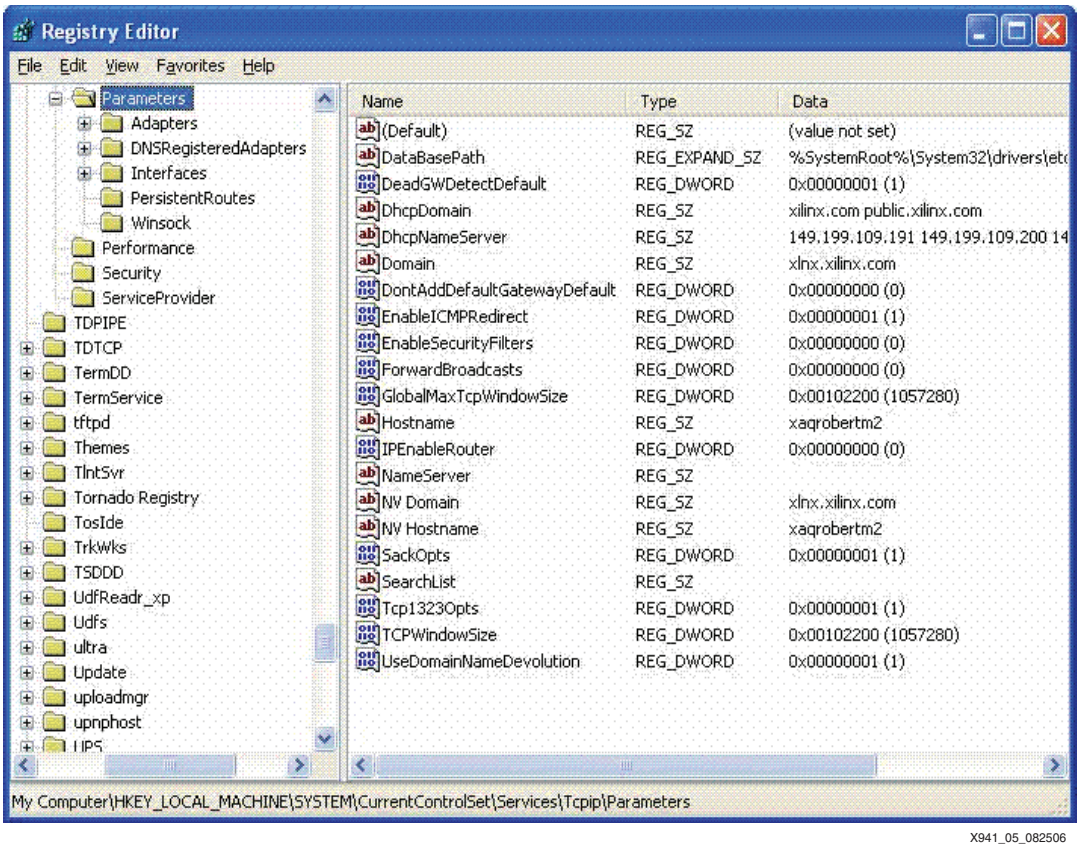

Figure 5: **Registry Editor**

NIC Card Adjustments

Make sure the host NIC card has been set up for best performance. For Intel Pro type cards, setup the following:

9014 byte frame size (i.e. 9000 byte MTU)

Segmentation/checksum offload enable

Interrupt moderation rate of medium for TCP tests and high for UDP tests

Rebuilding cygwin Tools

In a cygwin shell, cd to the `ml405_ppc_plb_temac/cygwin` directory and enter:

```
make
```

This will compile the sources for Netperf and the host based remote control source, and copy the executables to the bin directory.

In order to remove the executables and all compiler generated files:

```
make clean
```

IMPORTANT: These commands must be issued from a fresh cygwin shell. If the cygwin shell had previously been setup for VxWorks development, the make process will fail.

Running the Tests

When the test starts on the target, capabilities of the MAC and the TCP/IP stack are examined. The test will not allow, for example, SGDMA mode if that mode is not supported by hardware.

The network is not loaded upon test startup so operations, such as pinging the target, will fail initially. When a specific test is selected, the network will be loaded with the specified MAC mode and MTU. After this point, the target can be pinged.

Main Menu Description

`1) Set New Transport Characteristics` - This menu selection allows the user to change how the MAC is loaded, which socket API to use, and the network MTU.

`2) Show Transport Characteristics` - This menu item prints out the current operating mode of the MAC.

`3) Set Link speed` - This menu item switches the link speed to either 10, 100, or 1000 MB per second, full or half duplex. The link settings allowed depends on the MAC and PHY. If the network is loaded, the change takes place immediately. To force the link to the selected parameters is dependent on how the PHY code is implemented. One method is to tell the PHY to advertise only for the selected properties and allow it to auto negotiate to them. The host computer's NIC must be capable of the selected speed and duplex.

`4) Set Netperf task priority` - This menu item allows the user to change the priorities of the Netperf client and/or server. This may be necessary in some circumstances with UDP tests. The task priority of the VxWorks network task daemon is fixed at 50. The range of priorities is from 0 to 255 with 0 being the highest priority. Do not set the Netperf processes to 0 as this may interfere with critical OS services. To set a process to a higher priority than the network task daemon, a value of 40 is recommended.

`5) Tx UDP Stream` - This menu item starts the Netperf client UDP_STREAM test to the host computer (which must be running the Netperf server `ml405_ppc_plb_temac/cygwin/bin/netserver`). The user is prompted for the socket size and the message size. Generally the largest socket size allowed is preferred and a message size that is close to the network MTU is desired for maximum performance. The goal of this test is to send UDP packets as fast as possible to the host. When the test is complete, a test report is generated.

See `cygwin/src/netperf-2.1pl3-cygwin/netperf.ps` for information regarding report formatting.

`6) Tx TCP Stream` - This menu item starts the Netperf client TCP_STREAM test to the host computer (which must be running the Netperf server `ml405_ppc_plb_temac/cygwin/bin/netserver`). The user is prompted for the socket size and the message size. Generally the largest socket size allowed is preferred and a message size that is a multiple of the TCP MSS is desired for maximum performance. The TCP MSS is usually the MTU - 60 bytes. The goal of this test is to send TCP packets as fast as possible to the host. When the test is complete, a test report is generated.

See `cygwin/src/netperf-2.1pl3-cygwin/netperf.ps` for information regarding report formatting.

`7) Tx Canned UDP & TCP menu` - This menu item starts a submenu that allows the user to run a series of TCP_STREAM and UDP_STREAM tests based on the capabilities of the MAC to the host computer (which must be running the Netperf server `ml405_ppc_plb_temac/cygwin/bin/netserver`). Depending on the MAC capabilities, this test may take a long time to complete. As the tests progress, test summary information is displayed.

`8) Rx Start Netperf Server` - This menu item starts the server process which will accept client connections from the host to perform TCP_STREAM or UDP_STREAM tests. Results from these tests are reported on the host machine. The server process is executed based on the current MAC and network settings. The server process is killed when:

- `Set New Transport Characteristics` menu item is selected.
- `Set Netperf task priority` menu item is selected and a new priority is entered.

`Tx UDP Stream` menu item is selected.

`Tx TCP Stream` menu item is selected.

`Tx Canned UDP & TCP menu` menu item is selected.

`9) Util menu` - This menu item switches to a utility menu described below.

Util Menu Description

`1) Enable remote control function` - This menu item starts the remote control process that waits for and processes commands from the host system. This allows the host to essentially perform the following commands from the main menu:

```
Set New Transport Characteristics
Tx UDP Stream
Tx TCP Stream
Rx Start Netperf Server
```

When the `Util menu` is exited, then the remote control process is killed.

`2) Show running tasks` - This menu item runs the VxWorks function taskShow(0,2) which will display the status of all tasks (processes) in the system.

`3) Show call stack usage` - This menu item runs the VxWorks function checkStack(0) which will display stack usage for all tasks.

`4) Show socket states` - This menu item runs the VxWorks function inetstatShow() which will display socket usage.

`5) Show net sys pool stats` - This menu item runs the VxWorks function netStackSysPoolShow() which will display the network system memory pool statistics.

`6) Show net data pool stats` - This menu item runs the VxWorks function netStackDataPoolShow() which will display the network data pool statistics.

`7) Show End netbuf stats` - This menu item runs the VxWorks function netPoolShow() which will display the MAC driver's memory pool statistics.

`8) Show IF stats` - This menu item runs the VxWorks function ifShow(NULL) which will display interface driver level statistics and status.

`9) Show IP stats` - This menu item runs the VxWorks function ipStatShow(0) which will display IP traffic statistics.

`10) Show UDP stats` - This menu item runs the VxWorks function udpstatShow() which will display UDP traffic statistics.

`11) Show TCP stats` - This menu item runs the VxWorks function tcpstatShow() which will display TCP traffic statistics.

`12) Set host/target IP addresses` - This menu item lets the user change the host and/or target IP address from the default specified in `xps_user.h`.

Running a Target Tx Performance Test

To acquire Tx performance data, the Netperf server needs to be started on the host and the Netperf client on the target.

The server can be started on the host, by running `ml405_ppc_plb_temac/cygwin/bin/netserver` from a cygwin shell:

```
$ ./netserver
Starting netserver at port 12865
```

The netserver can be left running indefinitely.

On the target, the client can be started so that it is of TCP or UDP type. An example test run for TCP (user data shown in bold):

```
Main Menu:
1   - Set New Transport Characteristics
2   - Show Transport Characteristics
3   - Set Link Speed (1000 FD)
4   - Set Netperf task priority
5   - Tx UDP Stream
6   - Tx TCP Stream
7   - Tx Canned UDP & TCP menu
8   - Rx Start Netperf Server (stopped)
9   - Util menu
100- Exit
Enter selection: 6

Size in bytes of local socket buf [253952]: <return>
Size in bytes of remote socket buf [253952]: <return>
Size in bytes of message [8192]: <return>
Number of seconds to run test [10]: <return>
temac: warning - MII clock is defaulted to 1000 Mbps
temac: Buffer information:
     0x005FFFE4 bytes allocated for all buffers
     1564 byte cluster size
     4008 Rx buffers, 1st buffer located at 0x005FE860
          4 Tx buffers, 1st buffer located at 0x00BFCD60
     0x00004000 bytes allocated for all BDs
          128 RxBDs, BD space begins at 0xFFFFC000
          128 TxBDs, BD space begins at 0xFFFFE000
TCP STREAM TEST to 192.168.0.4
Recv   Send    Send                  Utilization Service Demand
Socket Socket Message Elapsed        Send   Recv  Send    Recv
Size   Size   Size  Time Throughput  local  remote  local remote
bytes  bytes  bytes  secs. 10^6bits/s  % U  % U   us/KB us/KB

253952 253952   8192   10.00   74.02   100.00  -1.00  0.000 -1.000
```

The output above shows user interaction with the main menu. Once the TCP action is selected, a series of questions is asked as to what the parameters of the test should be. Entering return for any question uses the default value shown in brackets [ ]. The messages with `temac:` and buffering information are displayed from the temac driver when it is loaded.

Running a Target Rx Performance Test

To acquire Rx performance data, the Netperf server needs to be started on the target and the Netperf client on the host. On the target side, the server is enabled with the current MAC and network parameters in effect. Different parameters can be chosen by executing the menu command `Set New Transport Characteristics`.

```
Main Menu:
1   - Set New Transport Characteristics
2   - Show Transport Characteristics
3   - Set Link Speed (1000 FD)
4   - Set Netperf task priority
5   - Tx UDP Stream
6   - Tx TCP Stream
7   - Tx Canned UDP & TCP menu
8   - Rx Start Netperf Server (stopped)
9   - Util menu
100- Exit
Enter selection: 8
Starting netserver at port 12865


Main Menu:
1   - Set New Transport Characteristics
2   - Show Transport Characteristics
3   - Set Link Speed (1000 FD)
4   - Set Netperf task priority
5   - Tx UDP Stream
6   - Tx TCP Stream
7   - Tx Canned UDP & TCP menu
8   - Rx Start Netperf Server (running)
9   - Util menu
100- Exit
```

As a result of choosing to start the Rx Netperf Server, the status message should have changed from `stopped` to `running`.

On the host, the Netperf client to the target can be started as follows:

```
$ ./netperf -l10 -H 192.168.0.2 -C -t TCP_STREAM -- -m 14400 -s253952 -S253952
TCP STREAM TEST to 192.168.0.2
Recv   Send    Send                  Utilization Service Demand
Socket Socket Message Elapsed        Send   Recv  Send    Recv
Size   Size   Size  Time Throughput  local  remote  local remote
bytes  bytes  bytes  secs. 10^6bits/s  % U  % U   us/KB us/KB
```

```
253952 253952 14400  10.00  254.59     -1.00  81.00 -1.000 26.063
```

It is also possible to use the `tcp_stream` script which abstracts some of the command line clutter:

```
$ ./tcp_stream -l10 -H 192.168.0.2 -m14400 -s253952
```

Remote Control Operation

Remote control mode allows the host computer to manipulate the state of the MAC, PHY, and select other network attributes.

To start the remote control process on the target, enter the `Util menu` and select **Enable remote control function**. When running, the menu status for this command changes from `disabled` to `running`. If the network has not been loaded then it will at this time. To kill the remote control process, exit the `Util menu`.

While remote control is active, only the host computer can change MAC, PHY, and network characteristics. When a command is received from the host, a description of the command is displayed on the target's console.

On the host, in the `ml405_ppc_plb_temac/cygwin/bin` directory, scripts are used to cycle through all tests, collect, and display the results in a readable format. Depending on the MAC's capabilities, some of these scripts could take an hour to complete.

These scripts use as building blocks the commands `rc_devset.exe, rc_client.exe,` and `rc_server.exe`. These applications are what actually communicate commands to the target. Running these applications without any arguments provides additional info about these applications.

Summary of Host Command Tools and Scripts

The list below summarizes what is available on the host. Other specialized scripts may be present. The user can examine any script and use it as a template to create new ones for specialized data gathering.

`cygwin/bin/netperf.exe` - Client side of Netperf performance package. This program is used to test Rx performance on the target. It can be rebuilt by issuing **`make netperf`** in the cygwin directory. Command line options and output descriptions can be found in `cygwin/src/netperf-2.1pl3-cygwin/netperf.ps`

`cygwin/bin/netserver.exe` - Server side of Netperf performance package. This program is used to test Tx performance on the target. It can be rebuilt by issuing **`make netperf`** in the cygwin directory.

`cygwin/bin/rc_client.exe` - This program is used to tell the target to start a Netperf client test run. It can be rebuilt by issuing **`make rc`** in the cygwin directory.

`cygwin/bin/rc_server.exe` - This program is used to tell the target to start/stop its netserver process. It can be rebuilt by issuing **`make rc`** in the cygwin directory.

`cygwin/bin/rc_devset.exe` - This program is used to tell the target to change network/MAC parameters. It can be rebuilt by issuing **`make rc`** in the cygwin directory.

`cygwin/bin/rxscript_fifo.sh` - This program is used to run through a canned set of target Rx performance tests in FIFO mode. Uses `rc_*.exe`, and `netperf.exe`.

`cygwin/bin/rxscript_sg.sh` - This program is used to run through a canned set of target Rx performance tests in SGDMA mode. Uses `rc_*.exe`, and `netperf.exe`.

`cygwin/bin/txscript_fifo.sh` - This program is used to run through a canned set of target Tx performance tests in FIFO mode. Uses `rc_*.exe`, and assumes `netserver.exe` is running on the host.

`cygwin/bin/txscript_sg.sh` - This program is used to run through a canned set of target Tx performance tests in SGDMA mode. Uses `rc_*.exe` and assumes `netserver.exe` is running on the host.

Performance Tips and Observations

Host MTU Size Selection

In most circumstances the host MTU can be set to the same size as the jumbo MTU on the target.

For TCP tests, the protocol contains MTU discovery options that figure out the correct MTU. If the target is set for 1500 byte MTU and the host is set for 9000, then TCP will use 1500.

For UDP tests, care must be taken when sending packets to the target for Rx performance tests. UDP does not have MTU discovery so it will use the largest packet it can. If the target is set for 1500 byte MTU and the host for 9000, and a 5000 byte UDP message is sent to the target, the MAC hardware on the target will drop the packet. The solution, in this case, is to lower the host MTU to 1500 bytes.

TCP Window Size

The bigger the TCP window size, the more data can be sent to a peer without requiring an ACK packet to acknowledge the data had been received. This saves on overhead but introduces some risks if a packet gets lost. The likelihood of a lost packet is greater on a connection that may pass through many routers. On a point to point link, the dropped packets are typically due to lack of resources on either peer. In either case, a lost or dropped packet causes retransmissions. A bigger TCP window results in potentially having to retransmit more data which in turn yields lower throughput. Using a large TCP window on a point to point link is desirable if there is bulk data transfer activity. To avoid running out of resources, system tuning is needed until requirements are met.

The largest standard TCP window is 64KB. RFC 1323 increases this by multiples of 64KB by defining a window scaling option. In theory, with RFC 1323 the largest window size extends into the Gbps range. For VxWorks, the largest window size seems to be 248KB (253952 bytes). Using larger windows can cause the network to crash.

The window size can be controlled by the socket buffer size. These are key options for Netperf.

TCP Window Size and FIFO Depth

For systems without SGDMA, the processor reads frame data directly from the packet FIFOs. If the processor is busy or the peer has much higher throughput capacity, then the FIFO may fill up and overflow causing TCP retransmissions.

To avoid dropped packets in this situation, limit the TCP window size to less than the depth of the packet FIFO. When the peer sends enough data to fill the window, it will wait for an ACK before sending more, thus preventing the packet FIFO from being overrun and dropping packets. This strategy works only when there is a single connection between peers. If there are other types of traffic being received, then the packet FIFO may still overflow.

There are many variables that determine the desired window size. TCP message size, TCP MSS size, size of the Ethernet/IP/TCP headers, and MTU size are some of these variables. Since it is difficult to calculate the optimal setting for the TCP window size, the fastest way is to determine it experimentally. A good starting point is to set the window size to half the packet FIFO depth and increase from there until retransmissions start to occur.

TCP MSS

The MSS is the maximum number of TCP payload bytes present in a single packet. Keeping the length of the message in the socket call to a multiple of the MSS will allow the TCP protocol to keep packet lengths at or near the MTU. The MSS is typically calculated as MTU - 60 bytes. In Netperf the message size is controlled by the `-m` option.

UDP Rx Performance and VxWorks Task Priorities

Since UDP has no flow control, a fast host computer can potentially send UDP packets to the target much faster than the target can process. This results in many dropped packets and lower performance numbers. Packets can be dropped by hardware or by the TCP/IP stack. To find out if packets were dropped by the stack, run the UDP statistics command on the util menu. If there is a large number of dropped packets indicated in the statistics then changing task priorities may help.

The reason why UDP packets get dropped by the stack is typically resource related at the socket level. The consumer of data (the netserver process) reads the data, increments its performance counters, and waits for the next message. When packets come in very fast, the VxWorks network task daemon process starves the netserver process. Since the netserver process rarely runs, the socket buffer will begin to fill up. Once the buffer is full, the stack has no place else to put subsequent data so it drops the packet. In extreme conditions, there may have been 100000 packets received by the stack but only a handful made it to the netserver process.

By increasing the priority of the netserver process above that of the VxWorks network task daemon, netserver is able to receive all packets that get passed to the stack from the driver. When packets are dropped, they are dropped at the driver level. Performance numbers should increase dramatically as a result of increasing the priority.

Note that WindRiver discourages this practice, but if it meets a requirement without adversely affecting other parts of the system then it does not produce any harm. If UDP performance is satisfactory without the priority modification, then there is no need to do it.

Best Case Performance Results

If all of these performance tips are implemented and the host computer is capable of transmitting at the highest rates, the best possible Ethernet throughput numbers for this system are following:

TCP Transmit- 591 MB per second with 100% CPU utilization

TCP Receive - 868 MB per second with 91% CPU utilization

UDP Transmit - 623 MB per second with 100% CPU utilization

UDP Receive - 885 MB per second with 100% CPU utilization

Standalone Echo Server Application

In addition to the VxWorks Performance Test Suite, this reference system comes with a standalone Echo Server application. This application transmits out whatever data was received by swapping the source and destination addresses in the packet data and retransmitting the data.

Echo Server Application Design Notes

This application does not use an operating system, it runs under `Standalone BSP`.

Both the receive and transmit channels use 128 buffer descriptors each. The buffer descriptors are contained in BRAM.

The instruction and data caches are only enabled for external RAM and disabled for BRAM. Enabling caches in areas where the buffer descriptors are stored can cause the system to not work.

The MAC address of the TEMAC is hard-coded to be `06-05-04-03-02-01`.

Maximum length for any received frame can be 9024 bytes.

The application keeps track of CPU utilization, number of frames received/transmitted/dropped. These statistics are displayed every 5 seconds. At the time of outputting the counts, no new frames are received.

The number of frames received may differ from the number of frames transmitted due to interrupt coalescing.

Frames may be dropped if the received frames count is extremely large and the frame sizes are small. This scenario could lead to running out of buffer descriptors.

This performance measurement application does not verify the correctness of data.

This application is menu driven which is self explanatory.

A typical output for the Echo Server application is given below (user input in in bold):

```
L1 NETWORK PERFORMANCE TEST SUITE
Created on Aug 3 2006
PPC405 core          : 300 MHz
PLB bus              : 100 MHz

Initializing...
    Rx Frame Space: 0x0003AA20..0x0003CD48
    Tx Frame Space: 0x0003CD48..0x0003F070
    RxBd Space    : 0xFFFFC000..0xFFFFE000
    TxBd Space    : 0xFFFFE000..0x00000000
    MAC Tx Packet FIFO depth: 16384 bytes
    MAC Rx Packet FIFO depth: 16384 bytes
    MAC event FIFO depth    : 64 entries

Main Menu (speed=1000 Mbps, duplex=full):
4   - Tx SG DMA
5   - Tx Canned
9   - Rx SG DMA
20 - Start echo server
10 - Switch to 10 Mbps (full duplex)
100 - Switch to 100 Mbps (full duplex)
1000 - Switch to 1000 Mbps (full duplex)
99- Exit
Enter selection: 20
New Rx packet threshold count [32]:
New Tx packet threshold count [32]:
New Rx packet waitbound timer [1]:
New Tx packet waitbound timer [1]:

Frame storage:
   Base address : 0x03DCC000
   Frame count  : 256
```

```
   Frame length : 9024 bytes
Press any key to stop

CPU        Rx        Tx  Dropped
Load   Frames    Frames   Frames  Errors
----  -------  -------  -------  ------
  24   402432   402400        0       0
  27   402528   402534        0       0
  27   402528   402529        0       0
  27   402528   402528        0       0
  27   402528   402528        0       0
  27   402528   402527        0       0
  26   402528   402528        0       0
  27   402528   402528        0       0
  27   402528   402529        0       0
  27   402528   402528        0       0
  27   402528   402528        0       0
  27   402528   402527        0       0
  27   402528   402528        0       0
  27   402528   402529        0       0
  27   402528   402527        0       0
  27   402528   402529        0       0
```

Executing the Reference System

To execute this reference system, the ML405 board must be set up correctly and the bitstreams must have been updated and be ready for download. Pre-built bitstreams that have been verified to work properly are also available in the `ready_for_download/` directory under the project root directory. This directory contains the hardware bitstream, the VxWorks image and executable.elf for the Echo Server application. A HyperTerminal (or any other serial communications utility) must be connected to the COM port that is connected to the UART terminal on the ML405 board. The terminal settings must have the baud rate set to **115200** and data bits to **8**.

See Figure 6 for the HyperTerminal settings. The UART terminal is used to capture the results of the tests.

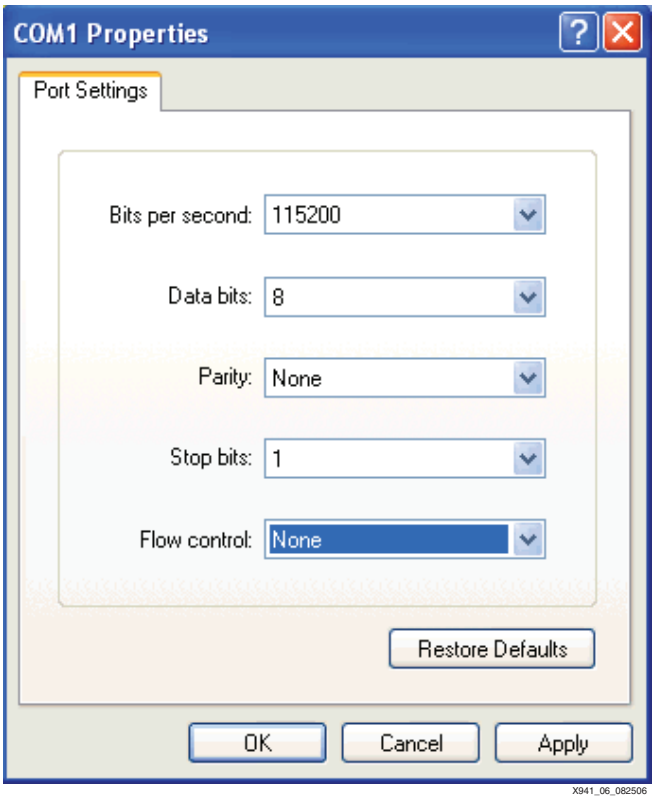

Figure 6: **HyperTerminal Settings**

Executing the Reference System using the Pre-Built Bitstream and the Compiled Software Applications

To execute the system using files inside the `ready_for_download/` in the project root directory, follow these steps:

1. Change directories to the `ready_for_download` directory.
2. Use iMPACT to download the bitstream by using the following:

   ```
   impact -batch xapp941.cmd
   ```

3. Invoke XMD and connect to the PowerPC 405 processor by the following command:

   ```
   xmd -opt xapp941.opt
   ```

4. Download the executables by the following command:

   ```
   dow executable.elf
   ```

Executing the Reference System from EDK

To execute the system using EDK, follow these steps:

1. Open `vxworks_performance.xmp or system.xmp` inside EDK.
2. Use **Hardware→Generate Bitstream** to generate a bitstream for the system.
3. Use **Software→Build All User Applications** to build the software applications.
4. Download the bitstream to the board with **Device Configuration→Download Bitstream**.
5. Launch XMD with **Debug→Launch XMD...**

6. Depending on the project pened, download the respective executables using the following command:
   a. VxWorks Performance Test Suite

      ```
      dow vxWorks
      ```

   b. Standalone Echo Server application

      ```
      dow executable.elf
      ```

The expected output for each of these test suites is described in detail in the respective sections for "VxWorks Performance Test Suite" and "Standalone Echo Server Application".

References

PLB TEMAC Product Specification, Xilinx DS489

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
11/09/06	1.0	Initial Xilinx release.
6/15/07	1.1	Updated for EDK 9.1.01i.