



XAPP1063 (v1.1) December 4, 2008

Reference System: XPS Local Link Tri-Mode Ethernet MAC Performance with VxWorks 6.3

Author: Brian Hill

Abstract

This application note describes how the standard network performance suite **Netperf** is used to measure XPS LL TEMAC performance with Wind River VxWorks 6.3.

Included Systems

Included with this application note two reference systems:

1. PowerPC® 440 Processor Reference System
<https://secure.xilinx.com/webreg/clickthrough.do?cid=113442>
2. PowerPC 405 Processor Reference System
<https://secure.xilinx.com/webreg/clickthrough.do?cid=111453>

Introduction

Many factors can affect Ethernet performance. This application note discusses several of the values which can be manipulated to affect Ethernet performance. The standard network performance suite Netperf is introduced as a means of testing and measuring network performance.

A variation of Netperf 2.1pl3 which has been modified to operate with VxWorks is included with this application note. Building VxWorks with this software will be discussed. A pre-built VxWorks image is also provided.

Hardware and Software Requirements

The hardware and software requirements are:

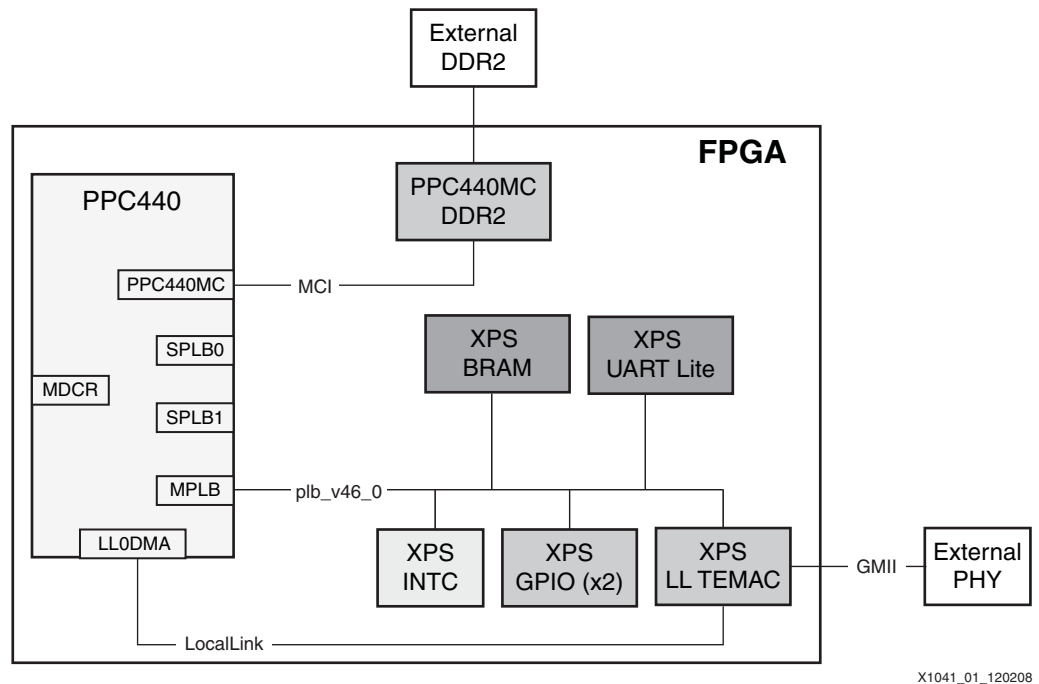
- Xilinx ML405 Development Board for the PowerPC 405 Processor reference system
- Xilinx ML507 Development Board for the PowerPC 507 Processor reference system
- Xilinx Platform USB Cable or Parallel IV Cable
- RS232 Cable
- Ethernet Cable
- Serial Communications Utility Program (e.g. HyperTerminal)
- Xilinx Platform Studio 10.01.03
- ISE® 10.1.3
- Host Computer with Wind River VxWorks 6.3 and Gigabit Ethernet NIC installed

Reference System Specifics

PowerPC 440 Processor Reference System

This application note includes a ML507 reference system shown in [Figure 1](#). The system address map is shown in [Table 1](#).

This reference system contains the IP cores necessary to provide an example of how to set up XPS_LL_TEMAC, how to verify that the core is operational, and how to measure raw Ethernet performance. In addition to the PowerPC 440 processor and the XPS_LL_TEMAC, this system includes the PowerPC 440 MC memory controller for DDR2, the Block RAM memory controller, a UART core, two GPIO cores, and an INTC interrupt controller. The XPS_LL_TEMAC PHY interface signals are connected to the tri-speed Marvell Alaska 88E1111 PHY on the ML507 board. For this reference system the GMII PHY interface type is used.



X1041_01_120208

Figure 1: PowerPC 440 Processor Reference System Block Diagram

Address Map

Table 1: PowerPC 440 Processor Reference System Address Map

Instance	Peripheral	Base Address	High Address
xps_bram_if_cntrl_1	xps_bram_if_cntrl	0xFFFFC000	0xFFFFFFFF
DDR2_SDRAM	ppc440mc_dds2	0x00000000	0x0FFFFFFF
RS232_Uart	xps_uartlite	0x84000000	0x8400FFFF
xps_intc_0	xps_intc	0x81800000	0x8180FFFF
Hard_Ethernet_MAC	xps_ll_temac	0x81C00000	0x81C0FFFF
LEDs_8Bit	xps_gpio	0x81400000	0x8140FFFF
Push_Buttons_Position	xps_gpio	0x81420000	0x8142FFFF

This reference system is discussed in further detail in [XAPP1041](#).

PowerPC 405 Processor Reference System

This application note includes the ML405 reference system shown in [Figure 2](#). The system address map is shown in [Table 2](#).

This reference system contains the IP cores necessary to provide an example of how to set up XPS_LL_TEMAC, how to verify that the core is operational, and how to measure Ethernet performance. In addition to the PowerPC 405 processor (with the PLBv46 Wrapper) and XPS_LL_TEMAC, this system includes controllers for DDR and Block RAM memory, a UART core, two GPIO cores, and an interrupt controller. The XPS_LL_TEMAC PHY interface signals are connected to the tri-speed Marvell Alaska 88E1111 PHY on the ML405 board.

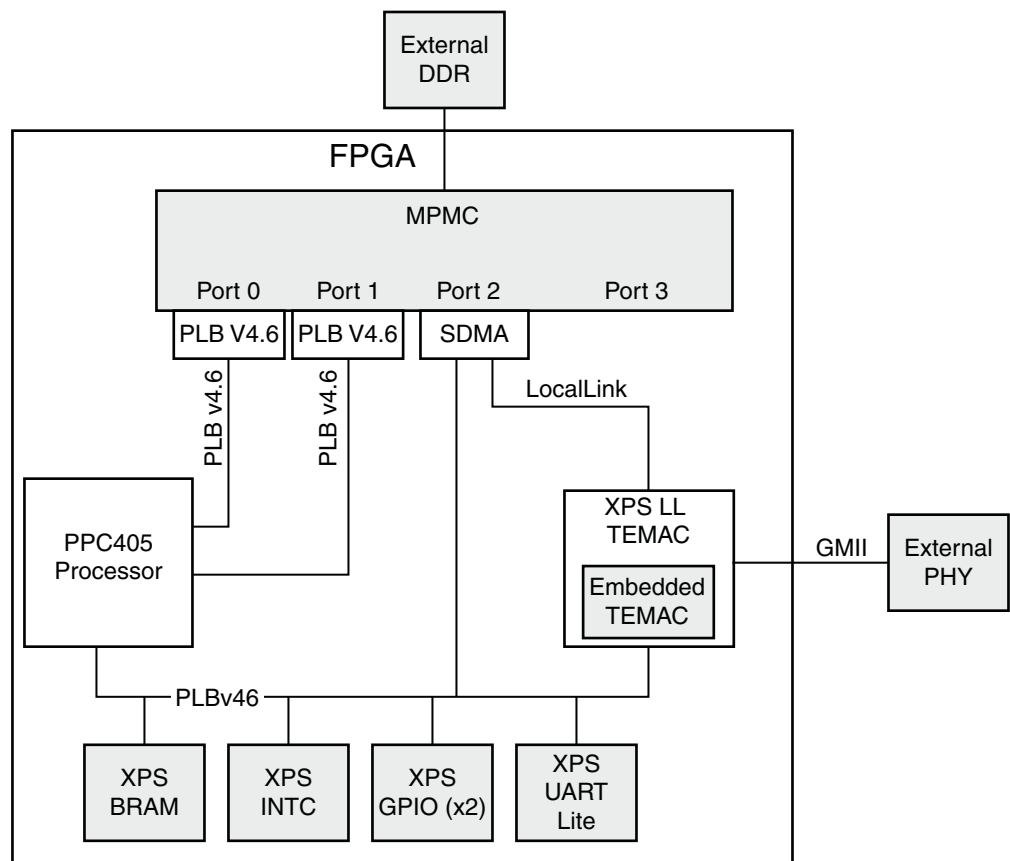


Figure 2: PowerPC 405 Processor Reference System Block Diagram

Address Map

Table 2: PowerPC 405 Processor Reference System Address Map

Instance	Peripheral	Base Address	High Address
xps_bram_if_cntrl_1	xps_bram_if_cntrl	0xFFFFE000	0xFFFFFFFF
DDR_SDRAM	mPMC (4.03.a)	0x00000000	0x07FFFFFF
DDR_SDRAM SDMA	mPMC (4.03.a)	0x84600000	0x8460FFFF
RS232_Uart	xps_uartlite	0x84000000	0x8400FFFF
xps_intc_0	xps_intc	0x81800000	0x8180FFFF
TriMode_MAC_GMII	xps_ll_temac	0x81C00000	0x81C0FFFF
LEDs_4Bit	xps_gpio	0x81400000	0x8140FFFF
Push_Buttons_Position	xps_gpio	0x81420000	0x8142FFFF

This reference system is discussed in further detail in [XAPP1041](#) XPS LL Tri-Mode Ethernet MAC Embedded Systems for MicroBlaze™ and PowerPC Processors.

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:

`netperf`: 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.

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

`netperf/cygwin/bin`: Contains host tools and scripts

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

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

Host System Requirements (Windows XP)

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. This requirement is met with the installation of the Xilinx EDK software.

2. NIC card. If running gigabit rates, the card should include checksum and segmentation offload functionality and provide support for jumbo frames
3. Installation of WindRiver Workbench 2.5 (VxWorks 6.3)

Test Suite Design Notes for Target Side

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 Menu System

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

It is not mandatory that the host OS be Windows XP. 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 checksum offloading
2. Select a socket API (BSD, zbuf, FastUDP)
3. Change network MTU
4. Change SGDMA interrupt coalescing values
5. Instruct the target to run a Netperf client test to the host (for Tx performance testing) and return the results of that test
6. Instruct the target to start its netserver process (for Rx performance testing)
7. Instruct the target to stop its netserver process

Remote control is made to function 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 2.5 (VxWorks 6.3), 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→Workbench 2.5→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 that the application will build with the 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 (highlighted in bold):

```
#include "ppc405_0.h"
#include "config_test.h"

#endif /* INCconfigh */

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

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 (highlighted 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 = C:/path/to/netperf
BSP_DIR = C:/path/to/bsp_ppc405_0
VXWORKS_BEFORE65 = YES
include $(PERF_DIR)/Makefile

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

where `</path/to/netperf>` 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 on what board they are running.

Finally, to build the operating system and the application from the command shell, execute the following commands within the BSP directory:

```
make vxWorks
```

To remove all compiler generated files, run:

```
make clean
```

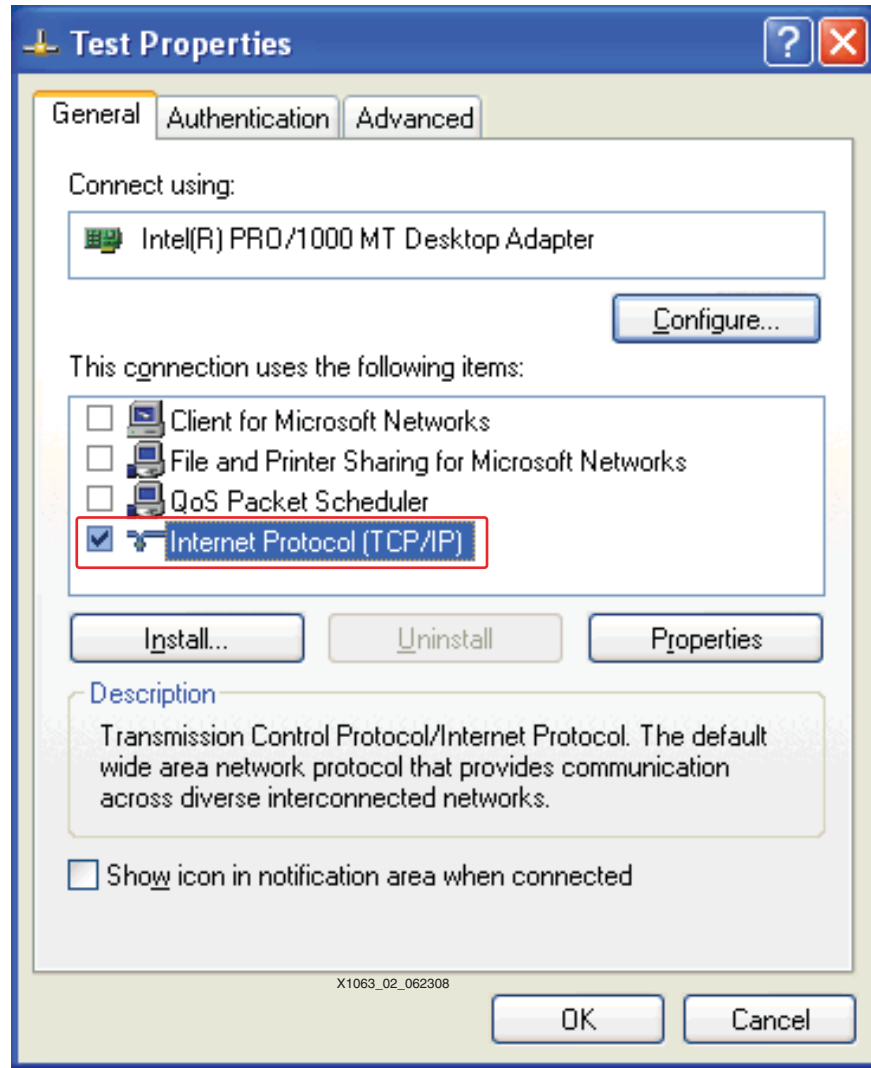
Host Computer Setup

Even on the fastest PC, gigabit line rate may not be achievable. Typical desktop PCs use a NIC which hangs off a 33 Mhz 32-bit PCI bus. The maximum throughput on this PCI setup is 133 MBps. This bandwidth is shared among all devices on the PCI bus. Because a GigE NIC can consume 125 MB 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:

Usually 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 set up (for Windows XP) as shown in [Figure 3](#).

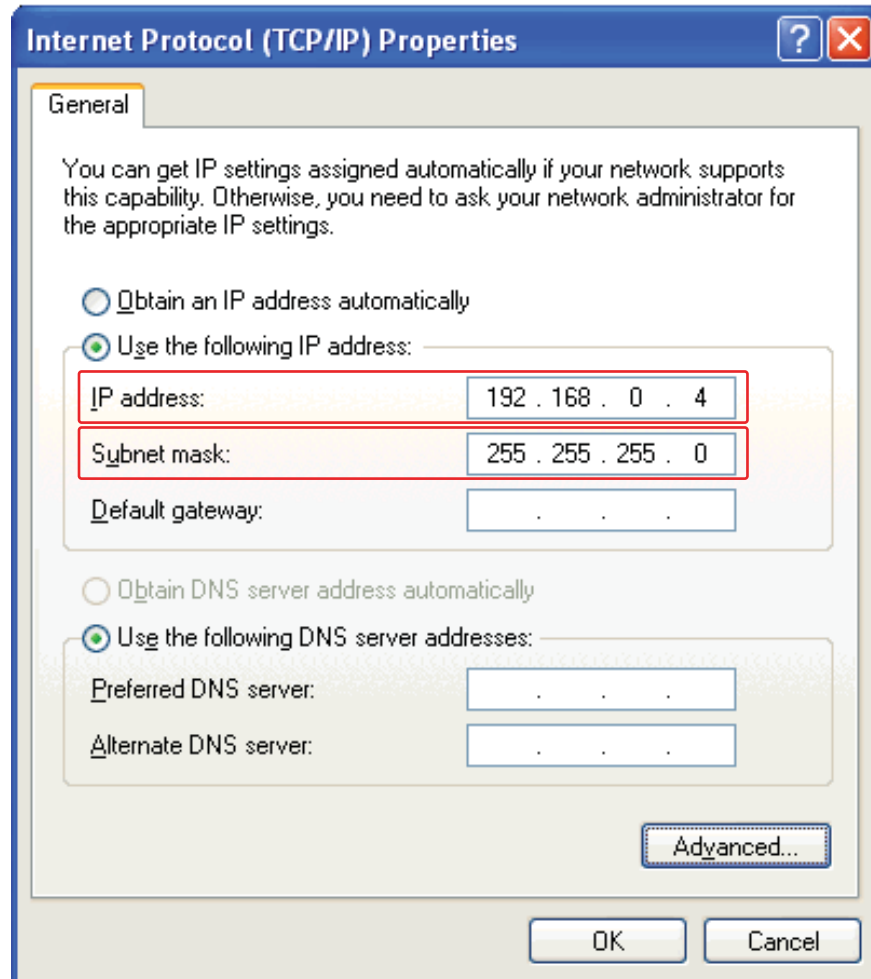


X1063_03_120208

Figure 3: Test Network Properties

In the Test Properties dialog window shown above in [Figure 3](#), select **Internet Protocol (TCP/IP)** only. Unnecessary network traffic may result if other services are selected.

In the Properties for TCP/IP, set up the IP address and Subnet mask only as shown in [Figure 4](#).



X1063_04_120208

Figure 4: TCP/IP Properties

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, modify the registry. Open the registry 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 = 1
- TcpWindowSize = 1057280
- GlobalMaxTcpWindowSize = 1057280

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

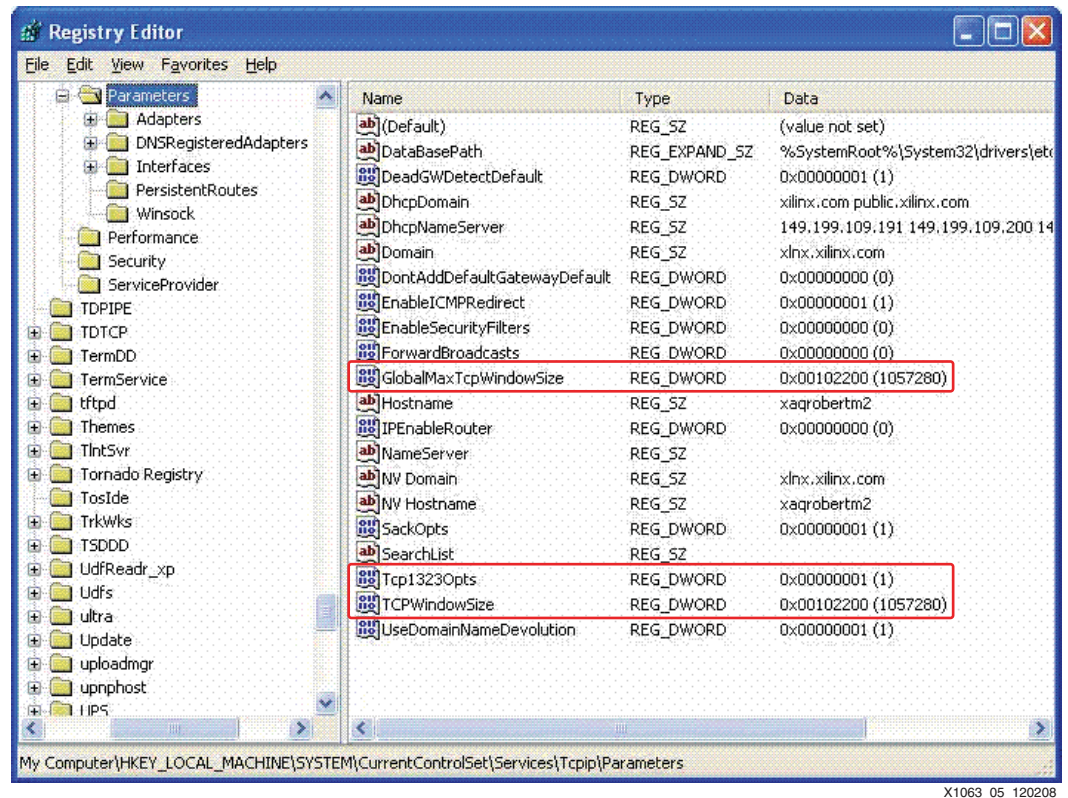


Figure 5: Registry Editor

NIC Card Adjustments

Ensure the host NIC card has been set up for best performance. For Intel Pro type cards, set up 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

Note: The changes to implement this configuration are specific to the hardware present in the host PC. The user should consult the documentation for the NIC in use.

Host Software

Disable any antivirus and firewall software present on the host PC. If these services remain enabled, performance *will* be negatively affected.

Host to Target Board Connection

The host PC and the ML405 should be connected with an Ethernet crossover cable. Intermediate devices such as a router or switch will negatively affect the performance results.

Note: Do not perform network performance tests on a live corporate network! Network stability will be adversely affected.

Running the Tests

When the test starts on the target, the capabilities of the MAC and the TCP/IP stack are examined. The test will not allow, for example, the SGDMA mode if that mode is not supported by hardware. The network is not loaded upon test startup so that 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, DMA interrupt coalescing, 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 MBps, full or half duplex. At present this option is not supported, and the TEMAC is hard coded to operate at 1000 MBps Full Duplex.
- 4) `Set Netperf task priority`: This menu item allows the user to change the priorities of the Netperf client, or server, or both. 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 49 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 `netperf/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 `netperf/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 `netperf/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 `netperf/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 `netperf/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 perform the following commands from the main menu:

`Set New Transport Characteristics`

`Tx UDP Stream`

`Tx TCP Stream`

`Rx Start Netperf Server`

Note: 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 displays 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 displays stack usage for all tasks.

4) `Show socket states`: This menu item runs the VxWorks function `inetstatShow()` which displays socket usage.

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

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

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

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

9) `Show IP stats`: This menu item runs the VxWorks function, `ipStatShow(0)`, which displays IP traffic statistics.

10) `Show UDP stats`: This menu item runs the VxWorks function, `udpstatShow()`, which displays UDP traffic statistics.

11) `Show TCP stats`: This menu item runs the VxWorks function, `tcpstatShow()`, which displays TCP traffic statistics.

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

Running a Target Tx Performance Test

To acquire Tx performance data, start the Netperf server on the host and the Netperf client on the target.

Start the server on the host, by running `netperf/cygwin/bin/netserver` from a EDK/Cygwin shell:

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

The netserver can be left running indefinitely.

Start the client on the target so that it is TCP or UDP type. An example test run for TCP (user data is shown in bold typeface):

```
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>

0:0x108e94:0x0:0x600000:0x7fe0000:0x20000:512:512:32:2:1500
litemac: warning - MII clock is defaulted to 1000 Mbps
litemac: Buffer information:
    0x005FFFE4 bytes allocated for all buffers
    1564 byte cluster size
    4008 Rx buffers, 1st buffer located at 0x0060D400
    4 Tx buffers, 1st buffer located at 0x00C0B900
    0x00020000 bytes allocated for all BDs
    512 RxBDs, BD space begins at 0x07FE0000
    512 TxBDs, BD space begins at 0x07FE8000

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      67.46  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 appear asking what the parameters of the test should be. Entering return for any question selects the default value shown in brackets []. The messages with `litemac:` and buffering information are displayed from the TEMAC driver when it is loaded.

Note: The default test parameters (as used above) will not produce the best performance results.

Running a Target Rx Performance Test

Start the Netperf server on the target and the NetPerf client on the host to acquire Rx performance data. 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 65535 -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 65535   10.00           212.28   -1.00  100.00  -1.000  38.590

```

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 -m65535 -s253952
```

Remote Control Operation

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

Enter the Util menu and select **Enable remote control function** to start the remote control process on the target. When running, the menu status for this command changes from disabled to running. If the network has not been loaded, then it will load 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 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 `netperf/cygwin/bin` directory, scripts are used to cycle through all tests, then collect and display the results in a readable format. Depending on the MAC's capabilities, some of these scripts may take an hour to complete.

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

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

`netperf/cygwin/bin/netserver.exe`: Server side of Netperf performance package. This program is used to test Tx performance on the target.

`netperf/cygwin/bin/rc_client.exe`: This program is used to instruct the target to start a Netperf client test run.

`netperf/cygwin/bin/rc_server.exe`: This program is used to instruct the target to start or stop its netserver process.

`netperf/cygwin/bin/rc_devset.exe`: This program is used to instruct the target to change network/MAC parameters.

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

`netperf/cygwin/bin/txscript_sg.sh`: This program is used to run through a predetermined set of target Tx performance tests in SGDMA mode. Uses `rc_*.exe`, and `netserver.exe`.

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 calculate 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, take care when sending packets to the target for Rx performance tests. Because UDP does not have MTU discovery, 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.

A larger MTU results in better performance because:

1. The processing load is greatly decreased because the number of packets the CPU must process decreases for the same amount of data bytes.
2. Ethernet overhead as a percentage of total data sent will decrease.

TCP Window Size

The larger the TCP window size, the more data can be sent to a peer without requiring an ACK packet to acknowledge that the data had been received. This setting saves on overhead but introduces some risks if a packet gets lost. A packet is more likely to get lost on a connection that may pass through many routers. On a point to point link, the packets are usually dropped

because of a lack of resources on either peer. In either case, a lost or dropped packet causes retransmissions. A bigger TCP window could cause the retransmission of more data which yields a 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 64 KB. RFC 1323 increases this by multiples of 64 KB 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 248 KB (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 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 usually calculated as MTU - 60 bytes. In Netperf the message size is controlled by the `-m` option.

UDP Rx Performance and VxWorks Task Priorities

Because 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. Because 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, 100000 packets may have been received by the stack but only a handful reach 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 enhancing suggestions are implemented and the host computer is capable of transmitting and receiving at the highest rates, the best possible Ethernet throughput numbers for this system are shown below, along with the remote control script command used to determine them.

Test command matrix:

Test	Command
TX TCP 9K MTU	<code>./txscript_sg.sh -M 9000 -m 65536</code>
RX TCP 9K MTU	<code>./rxscript_sg.sh -M 9000 -m 65536</code>
TX UDP 9K MTU	<code>./txscript_sg.sh -M 9000 -m 8968 -u</code>
RX UDP 9K MTU	<code>./rxscript_sg.sh -M 9000 -m 8968 -u</code>

Test	Command
TX TCP 1.5K MTU	<code>./txscript_sg.sh -M 1500 -m 65536</code>
RX TCP 1.5K MTU	<code>./rxscript_sg.sh -M 1500 -m 65536</code>
TX UDP 1.5K MTU	<code>./txscript_sg.sh -M 1500 -m 1468 -u</code>
RX UDP 1.5K MTU	<code>./rxscript_sg.sh -M 1500 -m 1468 -u</code>

Best case results, 9K MTU:

System	TCP 9000 RX	UDP 9000 RX	TCP 9000 TX	UDP 9000 TX
PPC405 300MHz	982.9	991.1	588.5	814.7
PPC440 400MHz	988.3	991.4	988.1	995.6

Best case results, 1.5K MTU:

System	TCP 1500 RX	UDP 1500 RX	TCP 1500 TX	UDP 1500 TX
PPC405 300MHz	185.9	109.5	144.4	148.0
PPC440 400MHz	934.6	306.2	765.5	779.7

If a part with a higher speed grade than the device present on the ML405 is used, it is possible to implement a system which clocks the PowerPC 405 processor at 375 MHz, which will improve Ethernet performance. See [XAPP1041 XPS LL Tri-Mode Ethernet MAC Embedded Systems for MicroBlaze and PowerPC Processors](#) for details on how to create such a system.

Executing the Reference System

To execute these reference systems, the ML405 board must be set up correctly and the bitstream must have been updated and programmed into the Virtex-4 device. Program the bitstream by downloading the pre-built bitstream from the `ready_for_download/` directory under the project root directory or by generating and downloading it from XPS.

Set the terminal software (such as HyperTerminal) as follows: Bits per second **115200**, Data Bits **8**, Parity **None**, and Flow Control **None**.

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

To execute the system using files in the `ready_for_download/` directory 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 command:


```
impact -batch xapp1063.cmd
```
3. Invoke XMD and connect to the processor by using the following command:


```
xmd -opt xapp1063.opt
```
4. Download the executable by using the following command:


```
dow vxWorks
```

Executing the Reference System from EDK

To execute the system using EDK, follow these steps:

1. Open `system.xmp` in EDK.
2. Use **Hardware** → **Generate Bitstream** to generate a bitstream for the system.
3. Download the bitstream to the board by selectin **Device Configuration** → **Download Bitstream**.
4. Launch XMD with **Debug** → **Launch XMD...**
5. Download the executable files by using the following command:

```
down vxWorks
```

Running the Software Application

Use the `run` command to run vxWorks after it has been downloaded. The expected output is shown within the “VxWorks Performance Test Suite” section.

References

1. [XAPP1041](#) *XPS Local Link Tri-Mode Ethernet MAC Embedded Systems for MicroBlaze and PowerPC Processors*.

Revision History

The following table shows the revision history for this document.

Date	Version	Description of Revisions
7/16/08	1.0	Initial Xilinx release.
12/4/08	1.1	Added the ML507 system.

Notice of Disclaimer

Xilinx is disclosing this Application Note to you “AS-IS” with no warranty of any kind. This Application Note is one possible implementation of this feature, application, or standard, and is subject to change without further notice from Xilinx. You are responsible for obtaining any rights you may require in connection with your use or implementation of this Application Note. XILINX MAKES NO REPRESENTATIONS OR WARRANTIES, WHETHER EXPRESS OR IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, IMPLIED WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL XILINX BE LIABLE FOR ANY LOSS OF DATA, LOST PROFITS, OR FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR INDIRECT DAMAGES ARISING FROM YOUR USE OF THIS APPLICATION NOTE.