XAPP1127 (v1.0) December 15, 2008

XPS LL Tri-Mode Ethernet MAC Performance with Monta Vista Linux

Author: Brian Hill

Abstract

This application note describes how the standard network performance suite **Netperf** is used to measure XPS LL TEMAC performance with MontaVista Linux 4.0.

Included Systems

Included with this application note is a ML507 reference system:

PowerPC® 440 Processor Reference System

Introduction

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

Netperf 2.4.4 source is included with this application note. Pre-built Linux and Cygwin images are also provided.

Hardware and Software Requirements

The hardware and software requirements are:

- Xilinx ML507 Development Board for the PowerPC 440 Processor reference system
- Xilinx Platform USB Cable or Parallel IV Cable
- RS232 Cable
- Ethernet Cable
- Serial Communications Utility Program, such as HyperTerminal
- Xilinx Platform Studio 10.1.03
- ISE® 10.1.3
- Host Computer with a Gigabit Ethernet NIC installed
- Monta Vista Linux 4.0

Reference System Specifics

PowerPC 440 Processor Reference System

This application note includes a ML507 reference system. See Figure 1 for a diagram of the system and Table 1 for the system Address Map.

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..

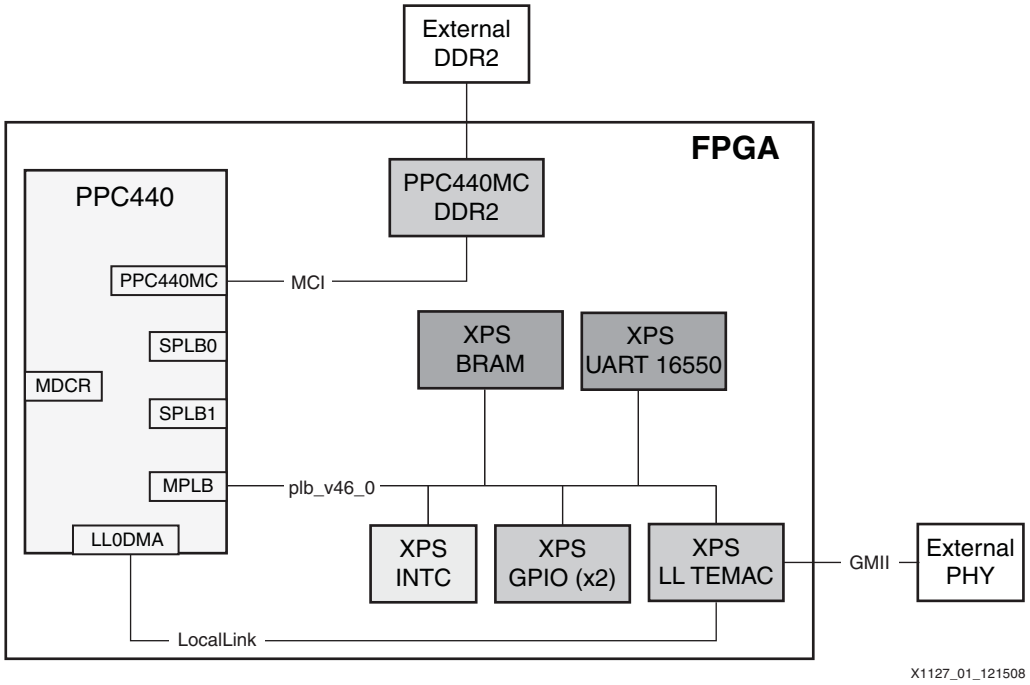

Figure 1: **Reference System Block Diagram**

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_ddr2	0x00000000	0x0FFFFFFF
xps_uart16550_0	xps_uart16550	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_5Bit	xps_gpio	0x81420000	0x8142FFFF
xps_iic_0	xps_iic	0x81600000	0x8160FFFF

Performance Test Suite

The intent of the enclosed software applications and scripts is to allow users to run the Netperf network performance test suite for MontaVista Linux (MVL). The user should be familiar with MontaVista Linux 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 MVL TCP/IP and Xilinx driver stack running on a PowerPC based Virtex®-5 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 tools and scripts to automate data gathering. The target client/server application is Netperf version 2.4.4 (www.netperf.org). This application can measure Ethernet throughput and CPU utilization. There are no dependencies on hardware.

Directory Layout

The directories in `xapp1127.zip` are arranged as follows:

`ready_for_download`:

Pre-built binaries for the embedded board.

`netperf/cygwin/bin`:

Pre-built binaries and scripts for the host PC

`netperf/netperf-2.4.4`:

Original netperf source used to build embedded and host netperf binaries.

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 1.5.17 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. MontaVista Linux 4.0 (if the user wishes to compile their own binary rather than using the one provided).

Test Suite Design Notes for Target Side

The target application consists of a bootable MontaVista Linux image. The ramdisk within this image contains front end scripst to control the Netperf application and appropriately configure the TEMAC for each performance run. A MVL Board Support Package (BSP) is not included. The BSP can be generated by the EDK tools.

Ramdisk Image

The ramdisk embedded within the bootable MVL image has the directory structure shown below.

/opt/netperf

netperf

netserver

set_net_params.sh

tcp_stream

udp_stream

These binaries and scripts comprise the core of the test suite. Each are discussed in detail in the "where <val> is the desired value (1-255). The process of tuning these values is best performed with scripts. The tcp_stream and udp_stream scripts provided with this application note are discussed in the "Running the Tests" section on page 12." section on page 11 of the document.

MVL 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 on page 4 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, the host tools must be recompiled and host scripts may need to be adjusted for whatever shell is being used.

Netperf

A Cygwin Netperf binary is provided. It has been built from from unmodified source available on the Internet.

Scripts

The scripts automate data gathering by invoking remote shell commands to place the MAC into the desired mode and for collecting and formatting test results.

BSP Setup and Image Compilation

The user may use the following steps to compile their own MVL bootable image rather than using the image provided.

Generate the BSP

1. Copy an MVL source tree to a new location.

2. Choose linux_2_6 as the OS in Software Platform Settings.

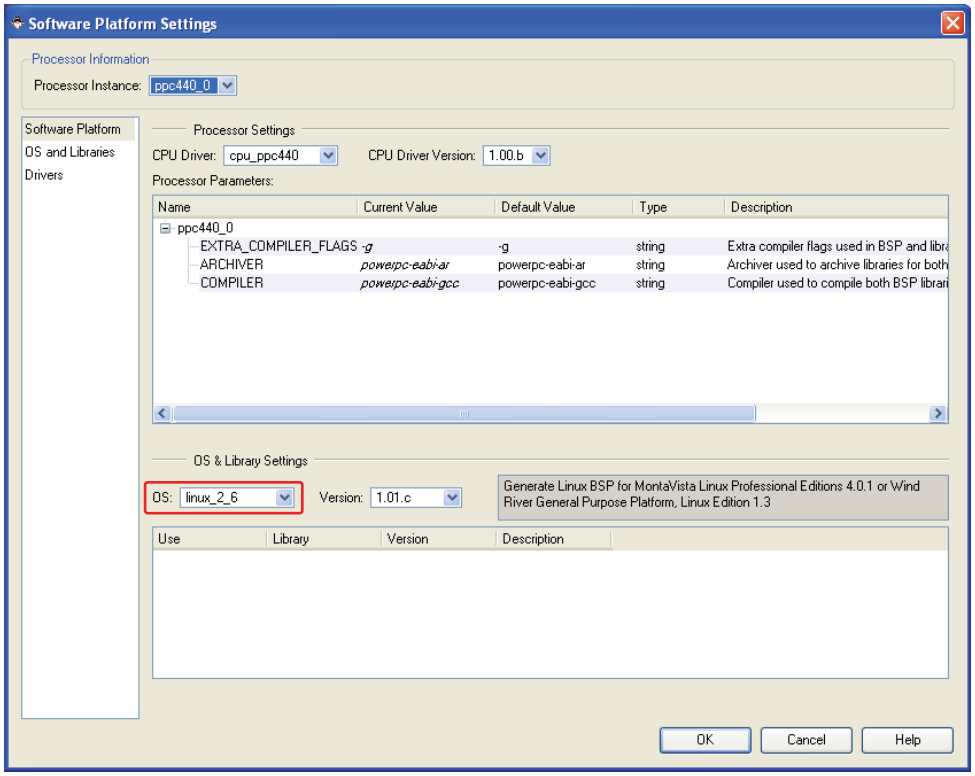

Figure 2: **Software Platform Settings: Software Platform**

3. Configure the MVL BSP in the OS and Libraries section.

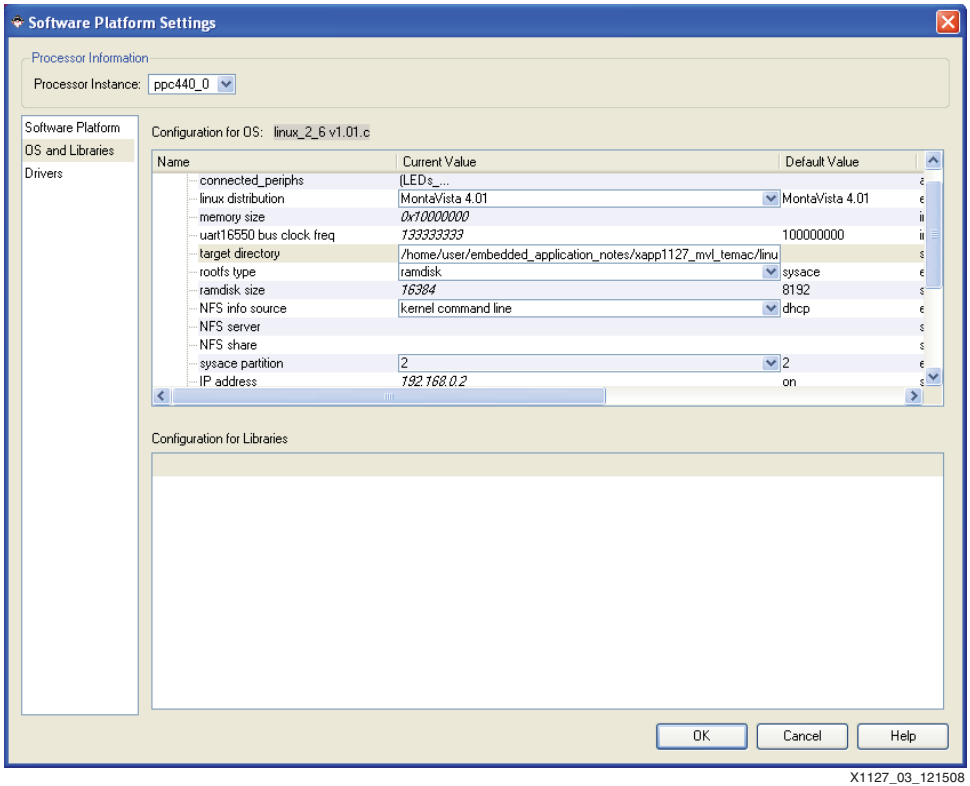

Figure 3: **OS and Libraries**

The user must pay particular attention to the following settings in OS and Libraries:

a. connected_periphs:

   Add all available peripherals in the system to the BSP

b. memory size:

   Ente the value `0x10000000`.

c. uart16550 bus clock freq:

   Enter the value `133333333`.

d. target directory:

   Specify the directory where the MontaVista Linux kernel is located. This is the location to where the files were copied in step 1 of this series of steps.

   Note: Files in this directory will be overwritten when the BSP is generated. This directory should be a **copy** which does **not** contain **any original files**.

e. rootfs type:

   Choose ramdisk. A ramdisk image `ramdisk.image.gz` is provided in the `ready_for_download` area.

f. ramdisk size:

   Enter the value `16384`.

g. IP address:

   Enter 192.168.0.2.

4. Choose **Software→Generate Libraries and BSPs**.

Setting up and Compiling the MVL image

The following steps are used to generate a bootable MVL image containing the test suite used in this application note.

Note: MontaVista Linux must be installed and properly configured before the user proceeds with the following instructions. All commands are assumed to be in the user's PATH.

1. Configure the kernel.

   The kernel is configured as desired by the user. Within the top of the MVL kernel tree, use the following command:

   $ **`make ARCH=ppc menuconfig`**

   This generates or modifies the kernel `.config` file. The configuration which was used to generate the bootable image provided with this application note is provided in the `ready_for_download/config_mvl4` file. The user may choose to copy this file to `.config` in the MVL kernel directory rather than generating a new configuration.

2. The supplied ramdisk image `ready_for_download/ramdisk.image.gz` is copied to the appropriate location in the user's MVL kernel tree, `<MVL kernel directory>/arch/ppc/boot/images/`.

3. The bootable kernel image is generated using the command:

   $ **`make ARCH=ppc CROSS_COMPILE=ppc_440- zImage.initrd`**

   The bootable image will be: `<MVL>/arch/ppc/boot/images/zImage.initrd.elf`

Host Computer Setup

Even on the fastest PC, gigabit line rate may not be achievable. Typical desktop PCs use a NIC which is connected to 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 MBps 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 improve the performance by 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 able to manage 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:

The host computer usually 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 4.

In the Test Properties dialog window shown in Figure 4, select **Internet Protocol (TCP/IP)** only. If other services are selected, unnecessary network traffic may result.

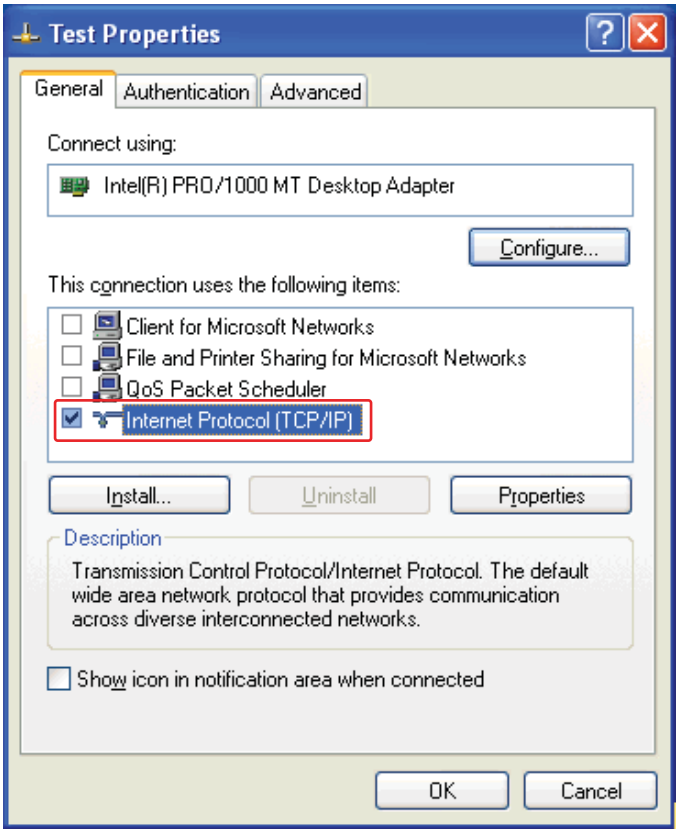

Figure 4: **Test Network Properties**

In the Internet Protocol (TCP/IP) Properties window, set up the IP address and Subnet mask only as shown in Figure 5.

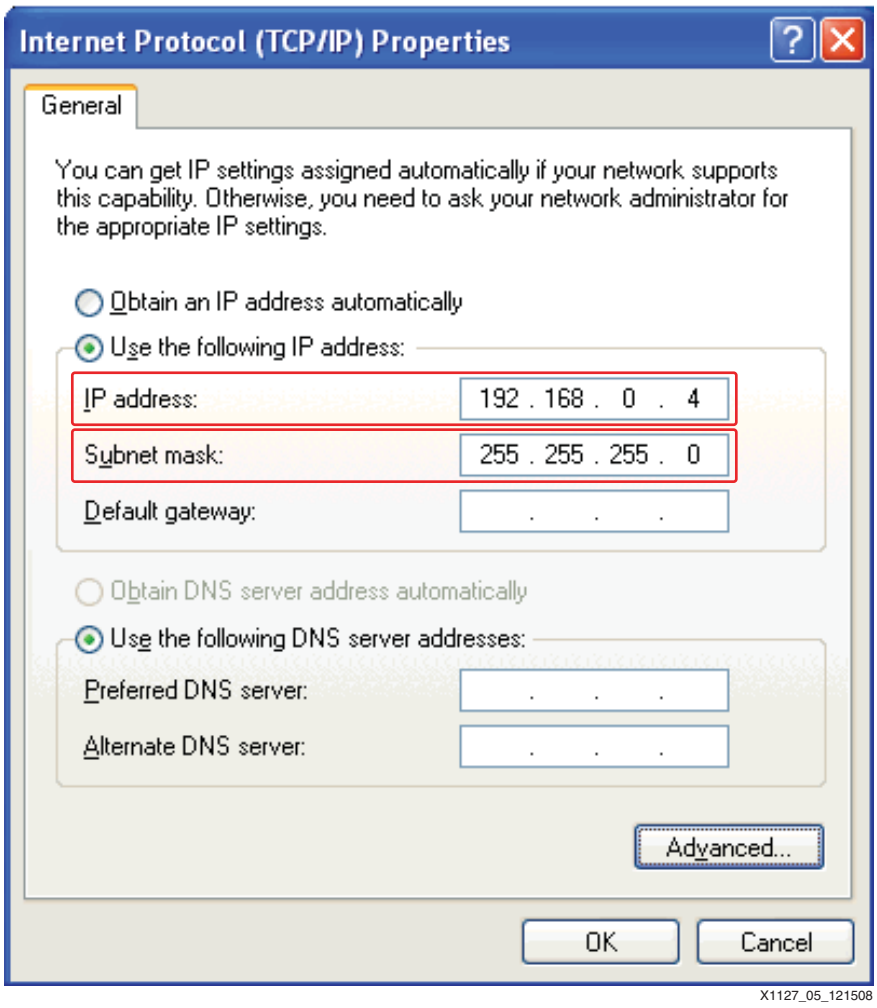

Figure 5: **TCP/IP Properties**

Modifying the Registry to Achieve Improved Performance

The following step is optional. However, it results in 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 6.

- Tcp1323Opts = 1
- TcpWindowSize = 1057280
- GlobalMaxTcpWindowSize = 1057280

Reboot the PC so that the new settings will take effect.

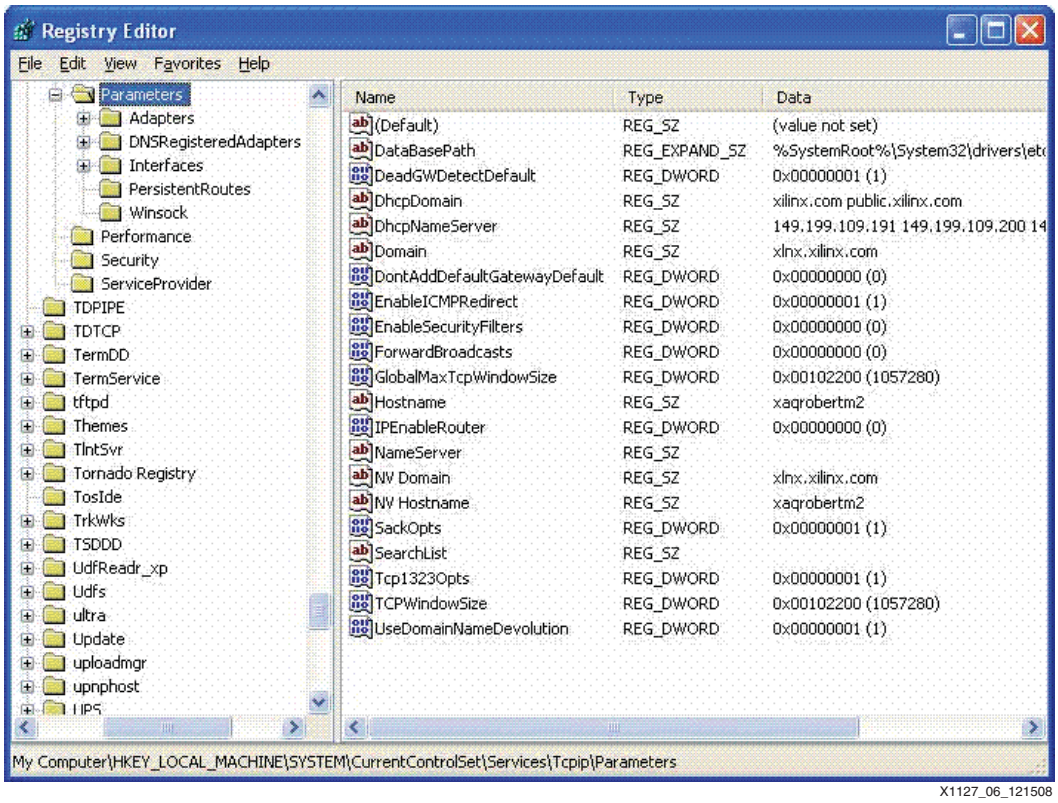

Figure 6: **Registry Editor**

NIC Card Adjustments

Set up the host NIC card for the best performance. For Intel Pro type cards, set up the following:

9014 byte frame size (for example, 9000 byte MTU)

Segmentation/checksum offload enable

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

Note: These parameters will be specific to whichever NIC driver the user has installed on the PC.

Host Software

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

Host to Target Board Connection

The host PC and the ML507 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.

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, 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 that 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. In this case, the solution is to lower the host MTU to 1500 bytes.

A larger MTU results in better performance. The performance improves because:

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

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 that the data had been received, thereby saving on overhead but introducing some risks if a packet becomes 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 most likely caused by a 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 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.

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 typically calculated as MTU - 60 bytes. In Netperf the message size is controlled by the `-m` option.

TEMAC Coalescing Values

The behavior of the DMA engine which services the XPS LL TEMAC is configured with the number of packets to process before raising an interrupt indicating that the processors should service the hardware descriptors. This directly affects the processor load dedicating to servicing interrupts. Finding the correct balance for receive and transmit coalesce values is critical to tuning ethernet performance.

The Linux utility `ethtool` is a standard way to configure ethernet hardware settings. TEMAC coalescing values are changed on the target board with the following commands:

```
# /usr/sbin/ethtool -C eth0 tx-frames <val>
# /usr/sbin/ethtool -C eth0 rx-frames <val>
```

where <val> is the desired value (1-255). The process of tuning these values is best performed with scripts. The tcp_stream and udp_stream scripts provided with this application note are discussed in the "Running the Tests" section on page 12.

Executing the Reference System

To execute these reference systems, the ML507 board must be set up correctly and the bitstream must have been updated and programmed into the Virtex-5 device. Programming the bitstream can be done by either 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 **9600**, 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, use the following steps:

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

   ```
   impact -batch xapp1127.cmd
   ```

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

   ```
   xmd -opt xapp1127.opt
   ```

4. Download the executable by using the following command:

   ```
   dow zImage.initrd.elf
   ```

Executing the Reference System from EDK

To execute the system using EDK, use the following steps:

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

   ```
   dow zImage.initrd.elf
   ```

Running the Software Application

Use the **`run`** command to run Linux after it has been downloaded. The expected output is shown within the "Running the Tests" section on page 12.

Running the Tests

The test suite consists of the netperf executables and scripts which automate testing. Both the host and the target have netperf executables compiled from the same source, and both use the same shell scripts.

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 an EDK/Cygwin shell:

```
$ cd netperf/cygwin/bin
$ ./netserver
Starting netserver at port 12865
```

The netserver can be left running indefinitely.

The target board should be configured and booted as outlined in the "Executing the Reference System" section on page 11. Once Linux has booted on the target, the user will login as **`root`**. There is no password.

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 is shown in bold typeface):

```
# /opt/netperf/netperf -H 192.168.0.4 C -t TCP_STREAM -- -m 65535 -s253952
-S253952
TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 192.168.0.4
(192.168.0.4) port 0 AF_INET
Recv   Send    Send
Socket Socket  Message  Elapsed
Size   Size    Size     Time     Throughput
bytes  bytes   bytes    secs.    10^6bits/sec

253952 208896  65535    10.00     346.81
```

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

Running a Target RX Performance Test

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

Target:

```
# /opt/netperf/netserver
Starting netserver at port 12865
Starting netserver at hostname 0.0.0.0 port 12865 and family AF_UNSPEC
```

Host:

```
$ netperf -l10 -H 192.168.0.2 -C -t TCP_STREAM -- -m 65535 -s253952 -
S253952
TCP STREAM TEST from xyz.xilinx.com (0.0.0.0) port 0 AF_INET to 192
-168-0-2.xilinx.com (192.168.0.2) port 0 AF_INET
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      % S      us/KB   us/KB

208896 253952  65535    10.25      291.94   -1.00    85.16    -1.000  23.897
```

Scripted Operation

The scripts provided with this application note will repeatedly run netperf, iterating through TEMAC coalescing values to determine which configuration yields the best throughput. The two scripts, `tcp_stream` and `udp_stream`, are run on both the host and the target. When run on the host, the targets receive performance is tested. When run on the target, the targets transmit performance is tested.

The scripts will use various executables to achieve the task:

- **`awk`** is used to sift through the results and find the best performance
- **`rsh`** is used on the host to run commands (**`ifconfig`**, **`ethtool`**) on the target.
- **`ifconfig`** is used on the target to configure the TEMAC MTU.
- **`ethtool`** is used on the target to configure TEMAC coalescing parameters

EXAMPLE USAGE:

To test the target 192.168.0.2 UDP receive performance with a 1500 byte MTU, the following is run in the host:

```
$ udp_stream 192.168.0.2 1500 host
```

Notice that the script is explicitly told that it is runing on the host. It is expected that `netserver` already be running on the target.

To test the target UDP transmit performance to host 192.168.0.4 with a 1500 byte MTU, the following is run in the target:

```
$ udp_stream 192.168.0.4 1500 embedded
```

Again, it is expected that `netserver` already be running on the host.

Summary of Command Tools and Scripts

The list below summarizes the performance suite. 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`: Client side of Netperf performance package. This program is used to test RX performance on the target when run from the host, and TX performance of the target when run on the target.

`netserver`: Server side of Netperf performance package.

`tcp_stream`: Script which repeatedly runs netperf with different target Temac coalescing values

`udp_stream`: Script which repeatedly runs netperf with different target Temac coalescing values

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.

Command prompt guide:

Command Prompt	Environment
#	ML507
$	Host PC

Command matrix used to generate test results:

Test	Command
TX TCP 9K MTU	`# ./tcp_stream 192.168.0.4 9000 embedded`
RX TCP 9K MTU	`$ ./udp_stream 192.168.0.2 9000 host`
TX UDP 9K MTU	`# ./udp_stream 192.168.0.4 9000 embedded`
RX UDP 9K MTU	`$ ./tcp_stream 192.168.0.2 9000 host`
TX TCP 1.5K MTU	`# ./tcp_stream 192.168.0.4 1500 embedded`
RX TCP 1.5K MTU	`$ ./tcp_stream 192.168.0.2 1500 host`
TX UDP 1.5K MTU	`# ./udp_stream 192.168.0.4 1500 embedded`
RX UDP 1.5K MTU	`$ ./udp_stream 192.168.0.2 1500 host`

Best case results, 9K MTU (MBs):

TCP 9000 RX	UDP 9000 RX	TCP 9000 TX	UDP 9000 TX
714.2	992.3	741.1	520.5

Best case results, 1.5K MTU (MBs):

TCP 1500 RX	UDP 1500 RX	TCP 1500 TX	UDP 1500 TX
209.1	307.0	393.2	491.9

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
12/15/08	1.0	Initial Xilinx release.

Notice of Disclaimer