Xilinx Open Source Linux on the Avnet V5FX30T Evaluation Board

February, 2009
Version 1.1

Revision History

Version	Description	Date
1.0	Initial draft release	July 28, 2008
1.1	Updated for XPS 10.1.3 and ARCH=powerpc	Feb 3, 2009

Overview

This document describes a PPC440 design implemented and tested on the Avnet Virtex 5FXT Evaluation board. This processor system boots a pre-built Linux kernel image. This kernel image was built using the Linux source tree hosted on the Xilinx git server (http://git.xilinx.com) and the compiler toolchain in the Embedded Linux Developers Kit (ELDK) provided by Denx (http://www.denx.de).

Objectives

This reference design will demonstrate running Xilinx Linux on the Avnet V5FX30T evaluation board.

Requirements

This reference design will require the following software and hardware setup.

Software

The software requirements for this reference design are:

- Windows XP
- Xilinx ISE 10.1 with Service Pack 3
- Xilinx EDK 10.1 with Service Pack 3

Hardware

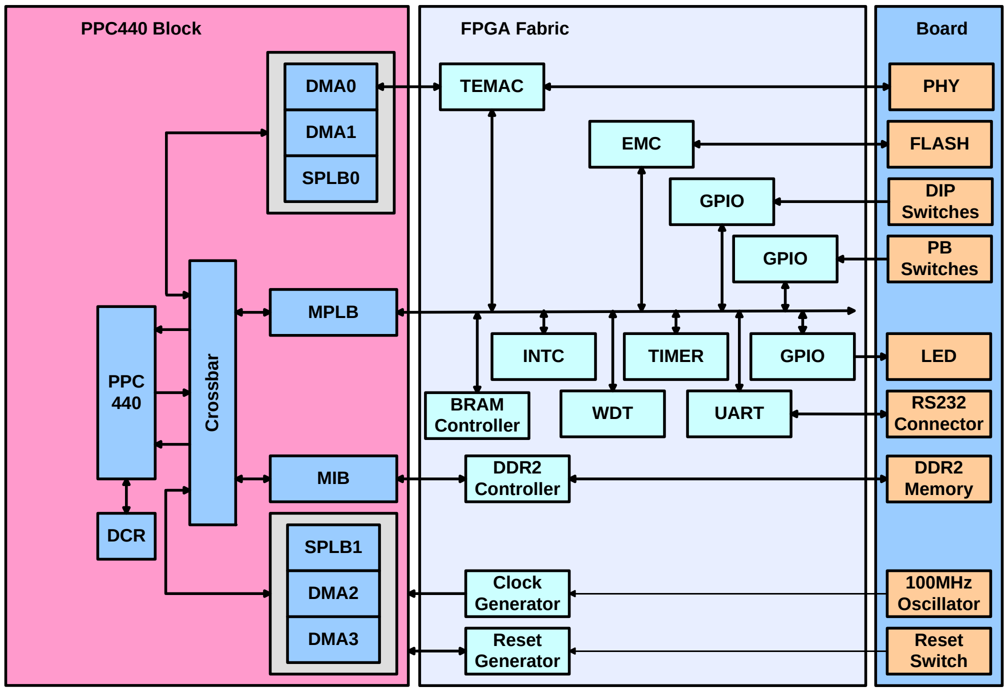
The hardware setup for this reference design is:

- Computer with 1 GB RAM and 1 GB virtual memory (recommended)
- Avnet Virtex-5 FXT evaluation board
- Straight through RS232 cable
- Ethernet cable
- Power supply
- JTAG programming cable (USB or PC4)

Processor System Block Diagram

The following figure shows a high-level block diagram of the processor system in the FPGA. The design consists of:

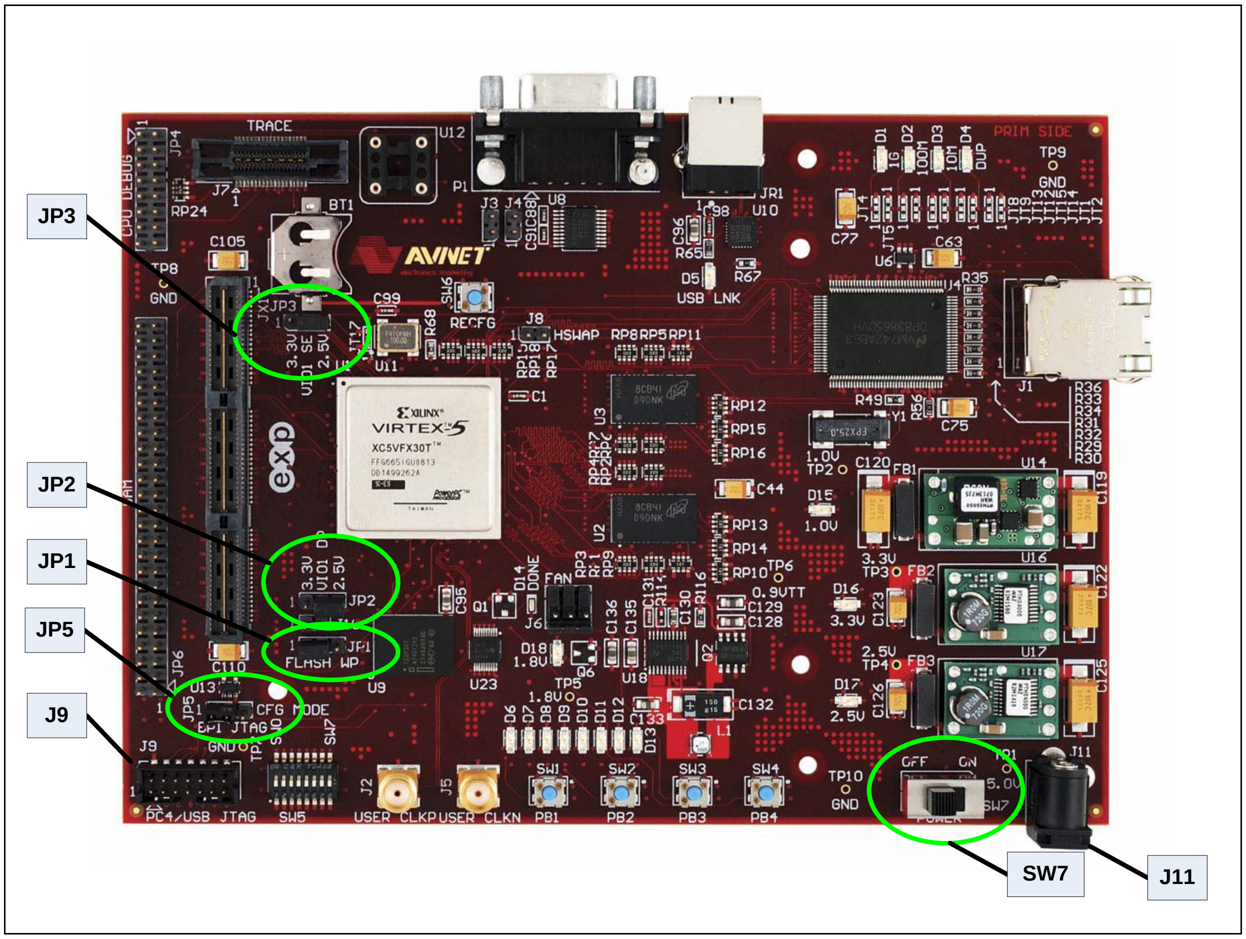
- PowerPC processor
- 64KB of BRAM
- 64MB of DDR2 SDRAM
- 16MB of Flash
- TEMAC Core (implemented with scatter-gather DMA)
- RS232 Port
- 8-position DIP Switch
- Push-button switches
- LEDs
- Timer
- Watchdog timer
- Interrupt Controller



Board Setup

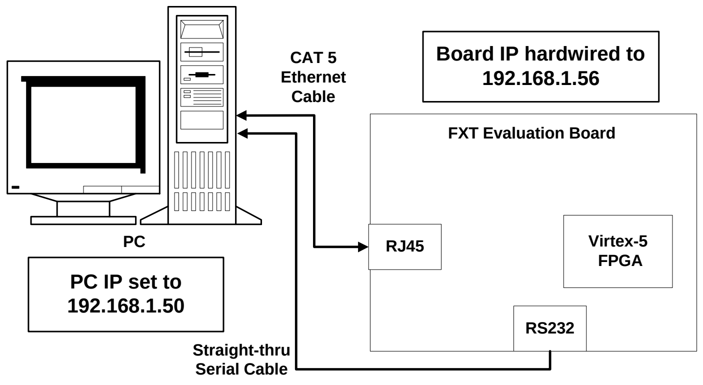
The following figure shows the location of various connectors and jumpers on the Avnet Virtex-5 FXT evaluation board. Perform the following steps to setup the board for running the application software.

1. Verify the Power switch, **SW7**, is in the **OFF** position.
2. Install a jumper on JP3 pins 2-3
3. Install a jumper on JP2 pins 2-3
4. Install a jumper on JP1 pins 1-2
5. Install a jumper on JP5 pins 2-3 (FPGA JTAG mode)
6. Connect the power supply to the J11 connector on the FXT evaluation board and also plug it into the AC outlet.
7. Connect the USB JTAG cable to J9 and the USB port of the PC.
8. Connect a straight through RS232 cable to the board DB-9 connector (P1) and the serial port of the PC. Alternatively, you can use an RS232-USB adapter and connect this adapter to the DB-9 connector and the USB port of the PC. In this case, you must install the RS232-USB driver for the adapter.
9. Slide the power switch to the **ON** position
10. Connect an Ethernet cable to J1 and the Ethernet port of the PC (the Ethernet cable must be connected directly to the PC or via a hub).



PC Setup

Use the Windows XP Control Panel to configure the host PC with an IP address of **192.168.1.50**, with a subnet mask of **255.255.255.0**



Running The Demo Files

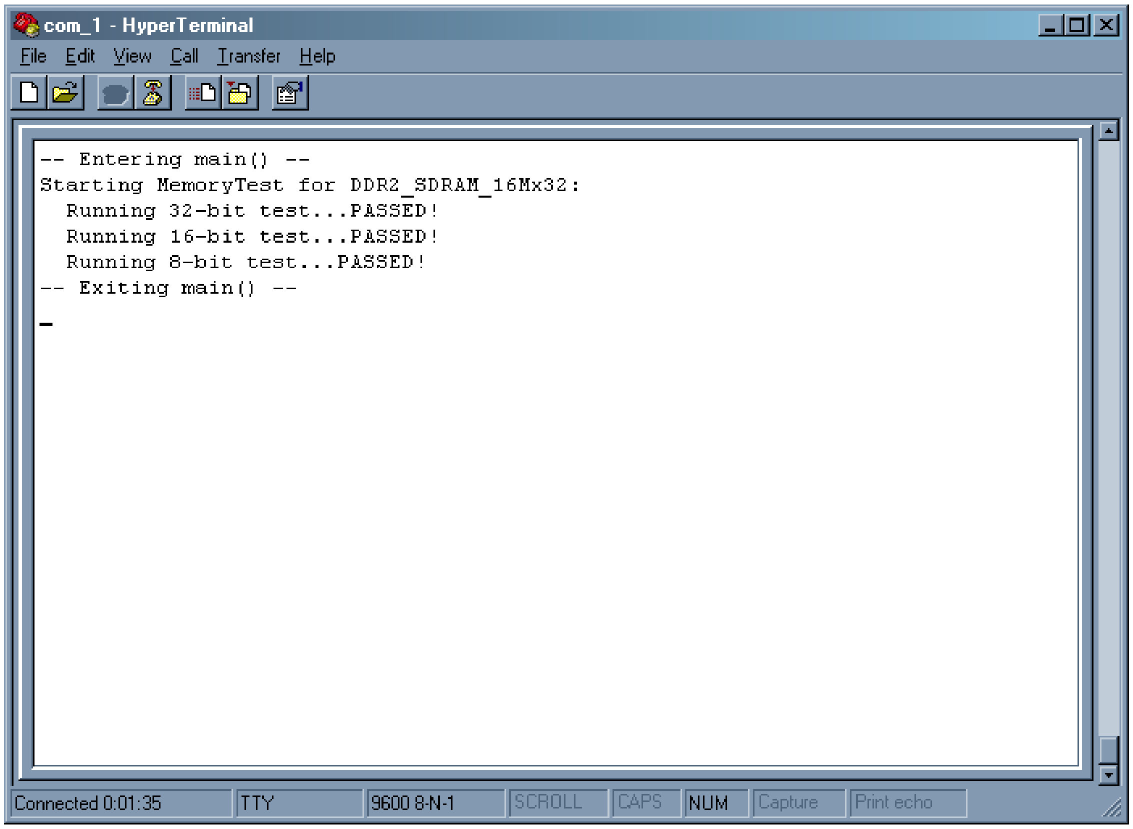
You can load the FPGA and run the test applications without building the design by using the demo script and the pre-built bit and elf files. You must have the Xilinx tools installed on your host, and have the hardware set up and connected as per the previous steps.

1. Start a HyperTerminal session and set the serial port parameters to 9600 baud rate, 8 bits, 1 stop bit, no parity and no flow control.
2. Double-click on **demo_linux.bat** in the demo_files directory. This batch file will load the FPGA with the hardware bistream and TestApp_Memory application and then run XMD to download and boot the Linux kernel image on the board. The command line window that opens will close after executing the script.

Note: There is a known issue that the gigabit (Gb) Ethernet operation on the Avnet V5FX30T board does not work. If you have a Gb-capable NIC on the host and are connecting directly to the board, you need to restrict the transmission rate to 100 Mbps Half-duplex. Change this setting through the Windows host Control Panel. Go to ***Start → Control Panel → Network Connections → Local Area Connection*** and select the ***Properties*** button. On the ***General*** tab select the ***Configure*** button. Now select the ***Advanced*** tab and look for a setting named something like ***Link Speed & Duplex*** and select ***100Mbps/Half Duplex*** among the drop-down options. Alternately you can connect the board and host to a 10/100Mbit router and let auto-negotiate determine the link.

Implementing The Design

1. Start a HyperTerminal session and set the serial port parameters to 9600 baud rate, 8 bits, 1 stop bit, no parity and no flow control.
2. Open the Linux system design in XPS and select **Software > Build All User Applications** to compile the software.
3. Select **Device Configuration > Update Bitstream** from the XPS GUI to build the design.
4. Select **Device Configuration > Download Bitstream** from the XPS GUI to download the **TestApp_Memory** application to the board. The memory test program will run on the board and you should see the following on the HyperTerminal:

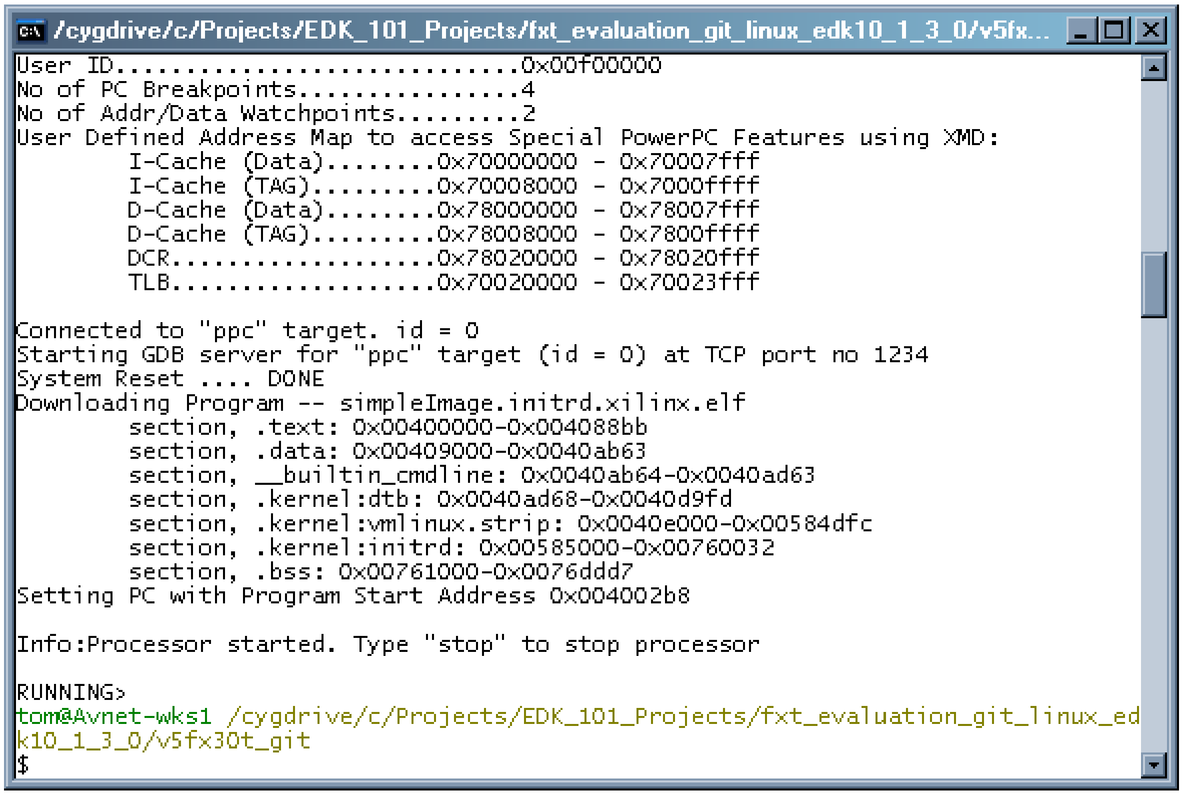


```
-- Entering main() --
Starting MemoryTest for DDR2_SDRAM_16Mx32:
  Running 32-bit test...PASSED!
  Running 16-bit test...PASSED!
  Running 8-bit test...PASSED!
-- Exiting main() --
_
```

5. Run the provided XMD debugger script file to download the Xilinx kernel image to the V5FX30T board. Downloading the kernel image to the board will take a few minutes. Select **Project > Launch EDK Shell** and enter the following command to run the script file:

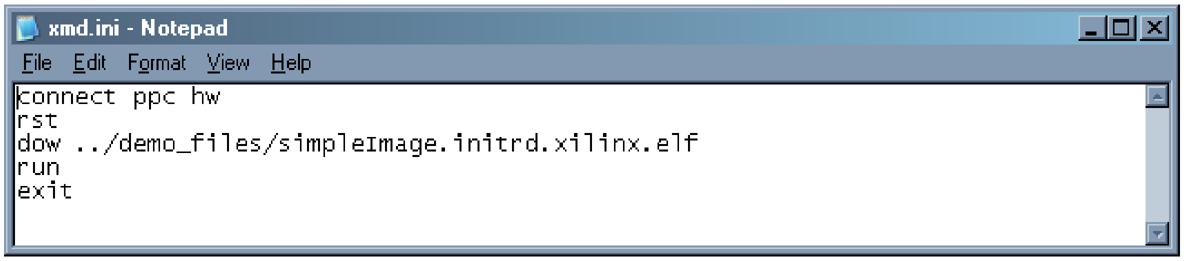
   xmd

   The XMD commands will run and should look similar to the window in the following figure:


/cygdrive/c/Projects/EDK_101_Projects/fxt_evaluation_git_linux_edk10_1_3_0/v5fx...

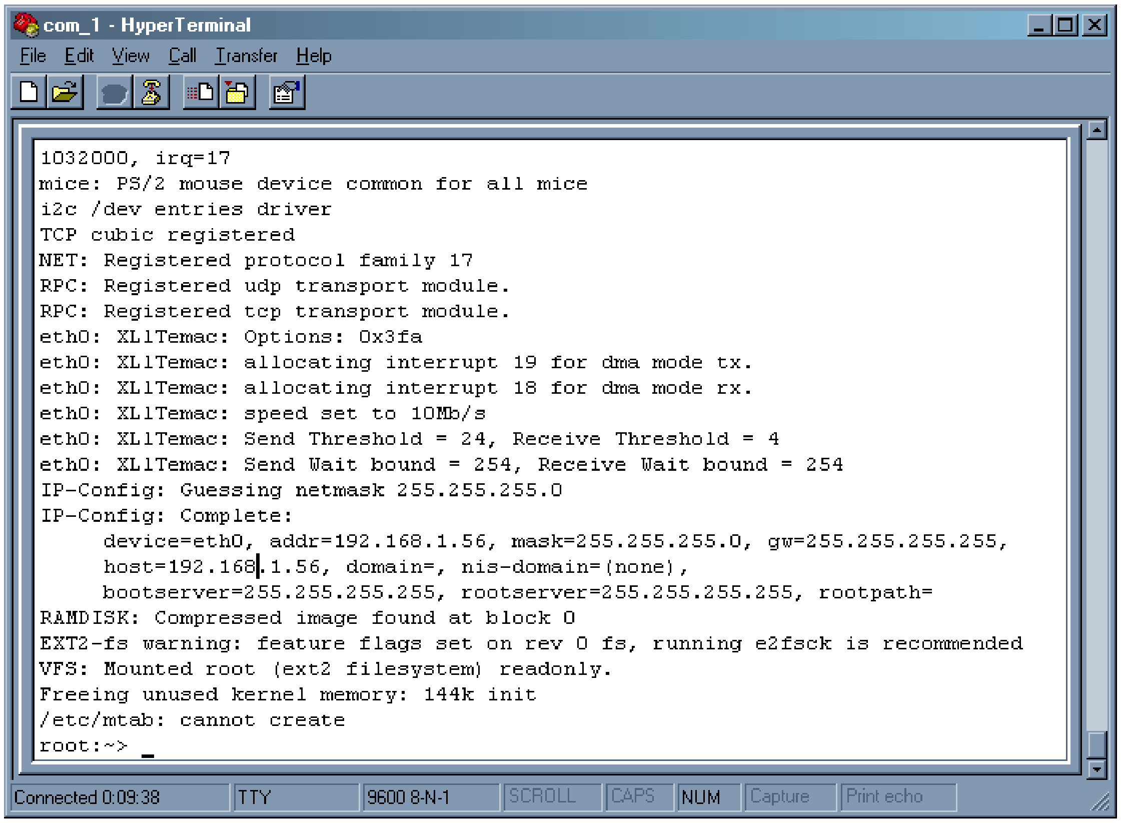
```
User ID.............................0x00f00000
No of PC Breakpoints................4
No of Addr/Data Watchpoints.........2
User Defined Address Map to access Special PowerPC Features using XMD:
        I-Cache (Data)........0x70000000 - 0x70007fff
        I-Cache (TAG).........0x70008000 - 0x7000ffff
        D-Cache (Data)........0x78000000 - 0x78007fff
        D-Cache (TAG).........0x78008000 - 0x7800ffff
        DCR...................0x78020000 - 0x78020fff
        TLB...................0x70020000 - 0x70023fff

Connected to "ppc" target. id = 0
Starting GDB server for "ppc" target (id = 0) at TCP port no 1234
System Reset .... DONE
Downloading Program -- simpleImage.initrd.xilinx.elf
        section, .text: 0x00400000-0x004088bb
        section, .data: 0x00409000-0x0040ab63
        section, __builtin_cmdline: 0x0040ab64-0x0040ad63
        section, .kernel:dtb: 0x0040ad68-0x0040d9fd
        section, .kernel:vmlinux.strip: 0x0040e000-0x00584dfc
        section, .kernel:initrd: 0x00585000-0x00760032
        section, .bss: 0x00761000-0x0076ddd7
Setting PC with Program Start Address 0x004002b8

Info:Processor started. Type "stop" to stop processor

RUNNING>
tom@Avnet-wks1 /cygdrive/c/Projects/EDK_101_Projects/fxt_evaluation_git_linux_ed
k10_1_3_0/v5fx30t_git
$
```

   When XMD was launched it automatically ran the commands in the xmd.ini file in the project directory. The xmd.ini file is a simple text file as shown here:



```
connect ppc hw
rst
dow ../demo_files/simpleImage.initrd.xilinx.elf
run
exit
```

6. Once the Linux kernel is done booting, you should see the following on the HyperTerminal:



```
1032000, irq=17
mice: PS/2 mouse device common for all mice
i2c /dev entries driver
TCP cubic registered
NET: Registered protocol family 17
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
eth0: XLlTemac: Options: 0x3fa
eth0: XLlTemac: allocating interrupt 19 for dma mode tx.
eth0: XLlTemac: allocating interrupt 18 for dma mode rx.
eth0: XLlTemac: speed set to 10Mb/s
eth0: XLlTemac: Send Threshold = 24, Receive Threshold = 4
eth0: XLlTemac: Send Wait bound = 254, Receive Wait bound = 254
IP-Config: Guessing netmask 255.255.255.0
IP-Config: Complete:
     device=eth0, addr=192.168.1.56, mask=255.255.255.0, gw=255.255.255.255,
     host=192.168.1.56, domain=, nis-domain=(none),
     bootserver=255.255.255.255, rootserver=255.255.255.255, rootpath=
RAMDISK: Compressed image found at block 0
EXT2-fs warning: feature flags set on rev 0 fs, running e2fsck is recommended
VFS: Mounted root (ext2 filesystem) readonly.
Freeing unused kernel memory: 144k init
/etc/mtab: cannot create
root:~> _
```

7. The Linux kernel now running on the board mounts a RAMdisk that is running the busybox application. Busybox combines tiny versions of many common Linux utilities into a single small executable, making it very popular for use in embedded systems. You can verify that the OS is running by inspecting all the running process by typing **ps -w** at the prompt. Feel free to poke around and try out a few Linux commands. The ftp and telnet daemons are active in the running kernel (login=root, password=root), so you can also use those applications to interact with the kernel.

Notes and Cautions:

The Linux kernel image provided with this reference design was built from the Linux source tree hosted on the public Xilinx git server. Xilinx provides the Linux source tree for free, and anyone can clone their own copy of it, but **Xilinx DOES NOT PROVIDE SUPPORT** for it. There is **NO** Xilinx hotline or forum support. Also, there is no Avnet support beyond the instructions in this document. Xilinx maintains this source tree in sync with the latest kernel from kernel.org, so you can expect it is quite 'bleeding edge' and perhaps not as stable as a Linux kernel available from a commercial Linux distribution. In fact, you should not consider the Xilinx Linux source tree to be a Linux distribution at all. If you clone the source tree to build your own kernel and find things that are broken your best resource for support will be to subscribe to the linuxppc-dev email reflector. Also, Xilinx does not provide the tool chain to build the kernel. Thankfully, the ELDK from Denx is a free and very good toolchain for the PowerPC. You can download it here. It is beyond the scope of this document to provide instructions for cloning the Xilinx git source tree, installing the toolchain and building the Linux kernel image.