Tutorial:

Ubuntu on the

Zynq™-7000 SoC

Featuring the Avnet ZedBoard

May 2013
Version 02

Table of Contents

Lab Setup for Xilinx 14.4 Tools

To complete the Tutorial labs, the following setups are recommended.

Software

- Xilinx ISE WebPACK 14.4 (Free license and download from Xilinx website)
- Cypress CY7C64225 USB-to-UART Bridge Driver
- Tera Term (Exact version used for this Tutorial is v4.75)
- VMware Player V5.0.0 (Exact version used is VMware-player-5.0.0-812388.exe)
- Virtual Machine Linux environment. For instructions on how to build this from scratch, see the **Avnet Speedway – Implementing Linux on the Zynq-7000 SoC** located at:
  http://www.zedboard.org/course/implementing-linux-zynq-7000-soc
- Adobe Reader for viewing PDF content (Exact version used is Adobe Reader X 10.1.4)
- 7-zip file archive utility

Hardware

- PC workstation with at least 5 GB RAM, 30GB free hard disk space, Windows 7 64-bit operating system, and Internet access to download updates from the source code repository and software mirror sites.
- Available SD card slot on PC or external USB-based SD card reader
- Avnet ZedBoard (**AES-Z7EV-7Z020-G)** – *included in kit*
- 2 USB cables (Type A to Micro-USB Type B) – *1 included in kit*
- 4GB SD card – *included in kit*
- CAT-5 Ethernet cable
- 4-port USB Hub
- USB Keyboard
- USB Mouse
- HDMI Cable
- HDMI Monitor with speakers
  - or DVI Monitor with an HDMI-DVI adapter and external speakers or headphones connected to Line Out on the ZedBoard

Technical Support

For technical support with any of the labs, please contact your local Avnet/Silica FAE or visit the ZedBoard.org support forum:

http://www.zedboard.org/forum

Additional technical support resources are listed below.

ZedBoard Kit support page with Documentation and Reference Designs

http://www.zedboard.org/content/support

For Xilinx technical support, you may contact your local Avnet/Silica FAE or Xilinx Online Technical Support at www.support.xilinx.com . On this site you will also find the following resources for assistance:

- Software, IP, and Documentation Updates
- Access to Technical Support Web Tools
- Searchable Answer Database with Over 4,000 Solutions
- User Forums
- Training - Select instructor-led classes and recorded e-learning options

Contact your Avnet/Silica FAE or Avnet Support for any additional questions regarding the ZedBoard reference designs, kit hardware, or if you are interested in designing any of the kit devices into your next design.

http://www.em.avnet.com/techsupport

Tutorial Notes

Each lab in the tutorial begins with a brief description of the objective and a set of general instructions. If these are sufficient you may proceed on your own. For those with less experience, detailed instructions for all steps are provided.

Lab instructions using the SD card assume that the card is connected to the host system via a USB adapter.

Tutorial Prerequisites

The labs in this document assume that you are using a Windows 7 Host system and a VMWare player running the Linux environment from the Implementing Linux on the Zynq-7000 SoC course.

The labs assume that you have a USB adapter to read/write the SD card that will be used for booting Ubuntu on the ZedBoard. If you have an SD card slot the operations will be much the same, but you won't have to worry about USB devices being registered between the Windows host machine and the Virtual Machine.

It would be an advantage to have some knowledge of the Xilinx Tools and the general architecture and operation of the Zynq device. A good overview can be found in the Avnet Speedway Introduction to Zynq.

Tutorial Overview

This tutorial provides a means to integrate several different technologies on a single platform. Using the Avnet ZedBoard, we have the power of an 800 MHz Arm-9 Cortex processor chip with dual 32-bit cores, combined with the unrivalled flexibility of a Xilinx FPGA to implement custom hardware systems. We use a Linux kernel as the foundation operating system running on the processor cores, but also add on a fully featured desktop from Ubuntu, contained in the root file system. The desktop allows the ZedBoard to function as a personal computer using a USB Keyboard and mouse, along with an HDMI monitor.

With Linux and Ubuntu in place, a vast array of applications can be installed and used, just as if a standard desktop PC were available. This includes the potential to develop in a native application development environment on the ARM system.

Why do we want a desktop computer in an embedded system? Because this is the future of mobile computing. By the end of 2013, the first smartphones will be commercially available that implement an Ubuntu desktop and an Android environment running side by side on top of a Linux kernel. This is possible because the Ubuntu desktop and the Java VM use a common Linux kernel.

Lab 1 - FPGA Hardware Platform

Lab Overview

The hardware platform used for Ubuntu was created with the Xilinx Design Tools, version 14.4. For the sake of convenience we will begin with the source files for a Xilinx Platform Studio project, but if you would like additional information on creating a Zynq processor system completely from scratch, please refer to the Avnet Speedway Introduction to Zynq.

When you have completed Lab 1, you will know how to do the following:

- Build a project with Xilinx Platform Studio to encapsulate the hardware design in a system.bit file.

Experiment 1: Obtain and Build the Hardware Platform

This experiment shows how to download the hardware source files for the Ubuntu system from ADI, and how to generate the system.bit file.

Experiment 1 General Instruction:

Download a hardware platform from the ADI website that provides the components necessary to run a desktop platform.

http://wiki.analog.com/resources/fpga/xilinx/kc705/adv7511

Generate a bitstream using Xilinx Platform Studio.

Experiment 1 Step-by-Step Instructions:

1. Using the web browser of your choice, browse to the website URL:

   http://wiki.analog.com/resources/fpga/xilinx/kc705/adv7511

   Download and save the **Zed HDL Reference Design** from the link:

   cf_adv7511_zed_edk_14_4_2013_02_05.tar.gz

2. Use a decompression utility such as 7-zip to extract the two folders into:

C:\ZedBoard\Zynq_Ubuntu

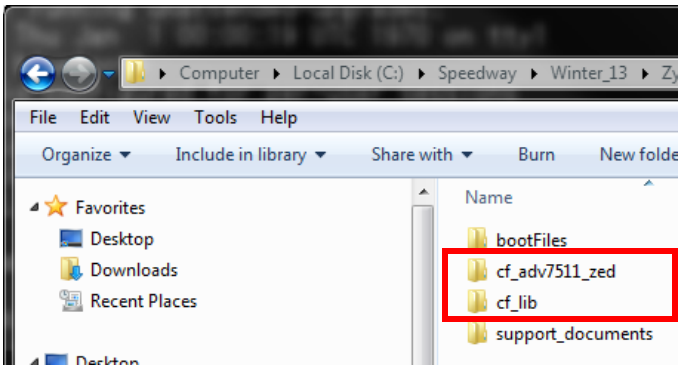


Figure 1 – Hardware Source Folders

You will extract two folders:

- cf_adv7511_zed
- cf_lib

3. Browse into the **cf_adv7511_zed** folder and double-click on the **system.xmp** file. This will launch Xilinx Platform Studio.

4. In the Navigator panel on the left side of XPS, click on the Generate BitStream icon. The build process to create a **system.bit** file may take 20-40 minutes depending on the capabilities of your host system

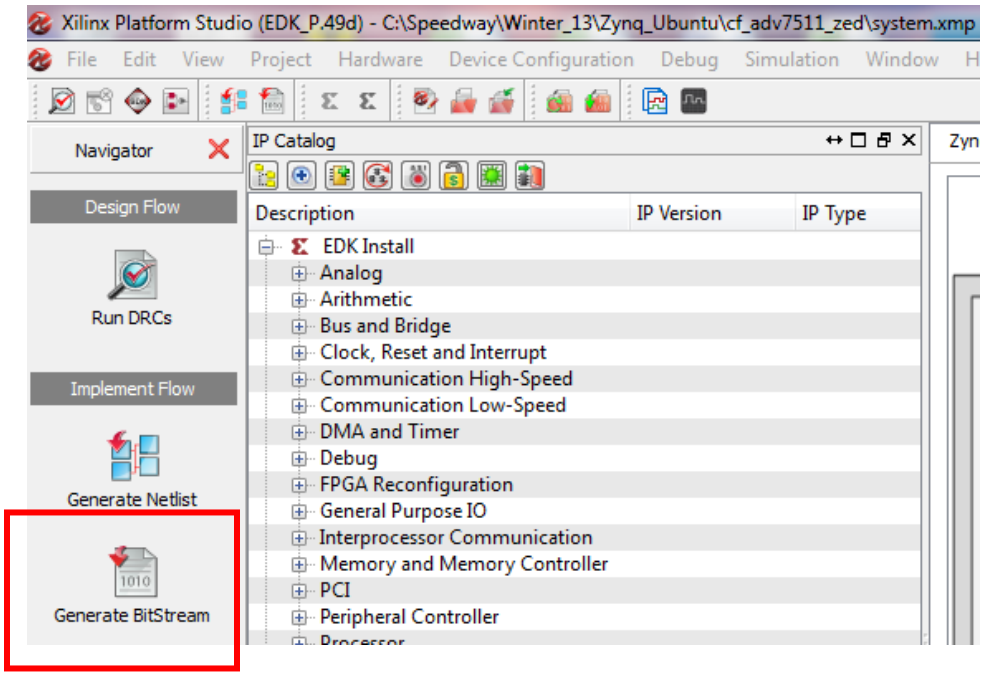


Figure 2 – Generate Bitstream in Xilinx Platform Studio

5. Once the bitstream has been generated successfully, we will make preparations to create software in later labs by initializing the Software Development Kit environment. We will also save the system.bit file created for later use in a boot image that will be used to start the software running in the ZedBoard at power up. The system.bit file is created at:

   C:\ZedBoard\Zynq_Ubuntu\cf_adv7511_zed\implementation\system.bit

   Using Windows Explorer, copy the system.bit file and browse to:

   C:\ZedBoard\Zynq_Ubuntu

   Create a new folder called **bootfiles** and paste the bitfile in at that location. We will be using it later when we create a boot image for the ZedBoard.

6. In the Navigator panel click on the SDK Export Design icon.

Figure 3 – Xilinx Software Development Kit Icon

7. Accept the default location for the hardware description files, ensure the checkbox for the bitstream and BMM files is ticked, and click on the Export and Launch SDK button.

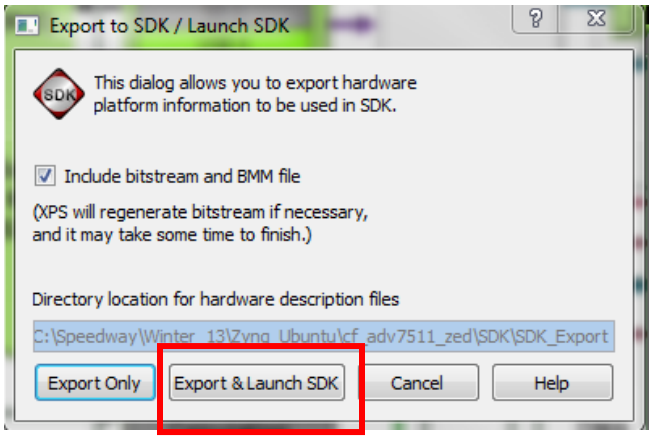


Figure 4 – Export Hardware to the SDK

8. When the hardware export completes and the SDK launches, you will be asked to select a workspace. Make sure that the workspace path you select is within the directory of your hardware project – it is entirely possible that the last workspace used by the SDK will be shown in the Workspace path. If this is the case, browse to the correct folder, which will be the path of your hardware project appended with:

cf_adv7511_zed\SDK\SDK_Workspace

When you are satisfied the workspace path is correct, click the OK button to proceed.

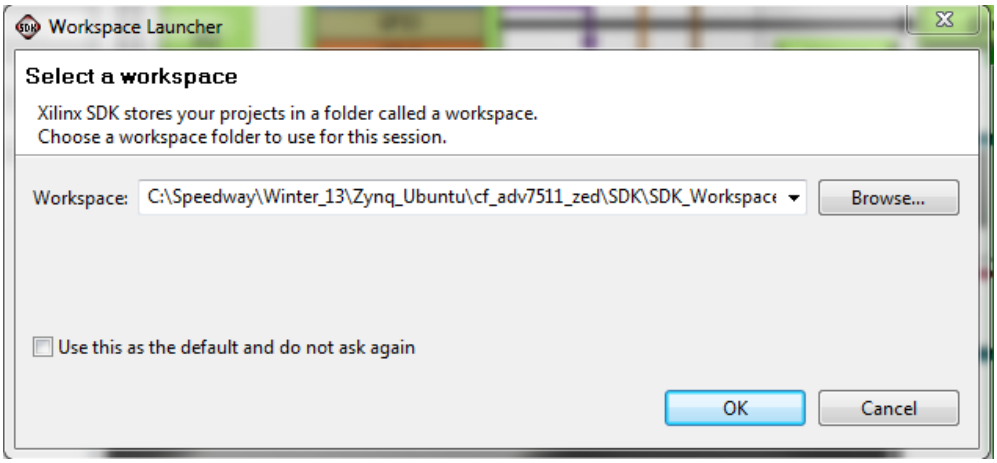


Figure 5 – Select an SDK Workspace

9. When the SDK launch is complete, you should see the following SDK screen representing the hardware platform.

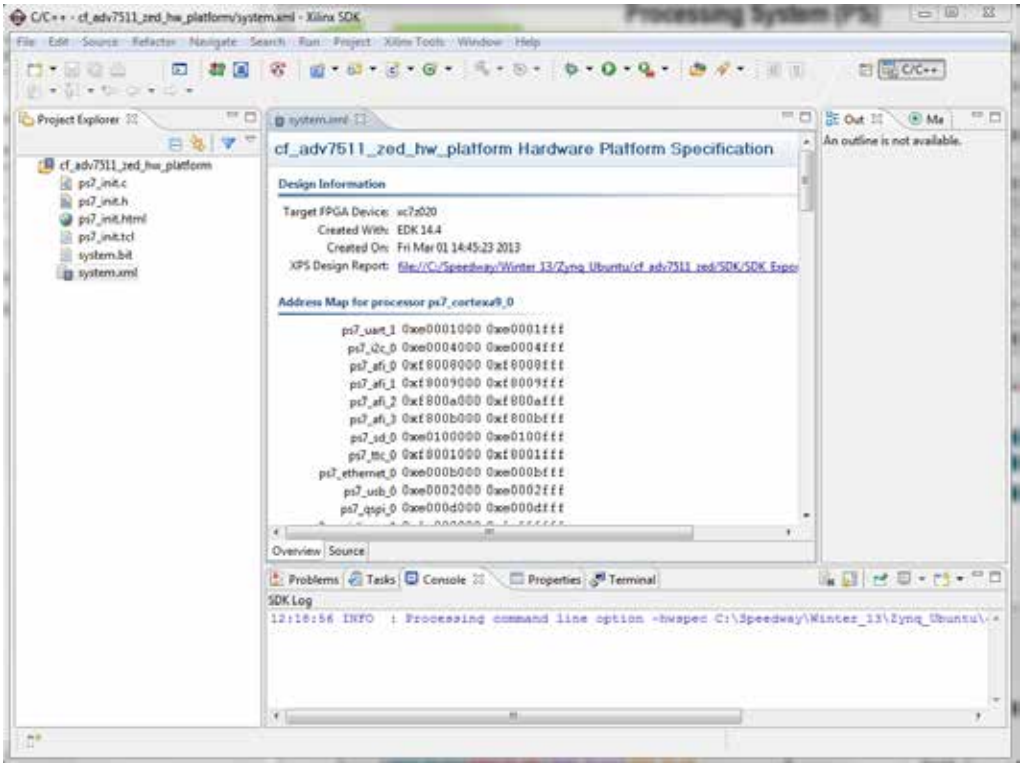


Figure 6 – Xilinx Software Development Kit

10. In order to boot the ZedBoard, we are going to need a bootstrap loader to bring up an environment that can be eventually used to load an operating system. To do that we will create a new Application project for a 1st stage boot loader.

    Select **File -> New -> Application Project** from the menu. Enter **zynq_fsbl_0** for the Project Name and click **Next**.

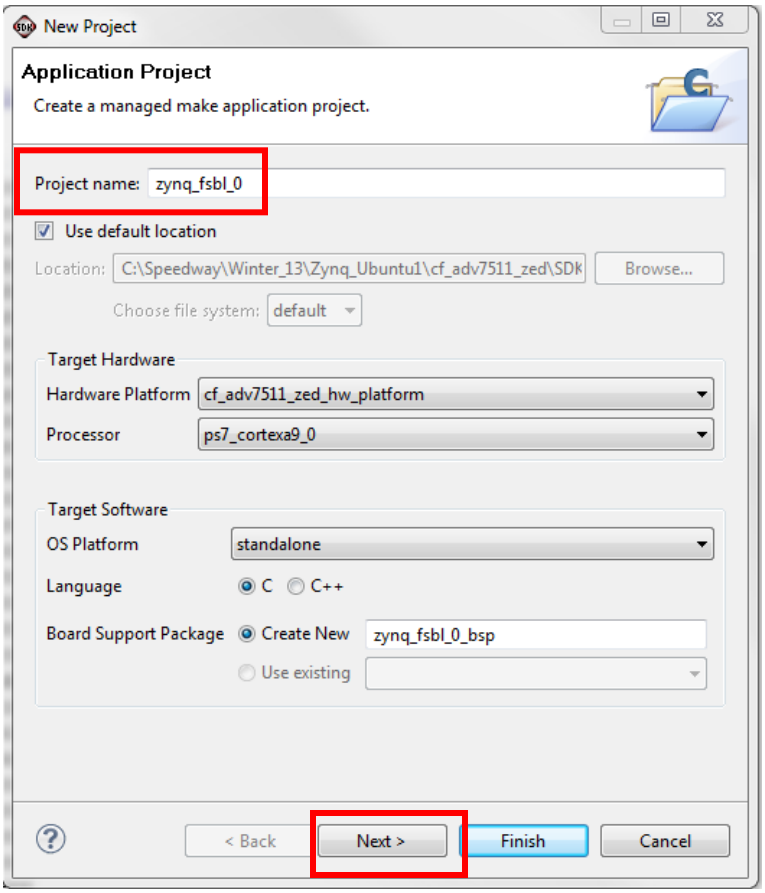


Figure 7 – First Stage Bootloader Project

11. Select **Zynq FSBL** from the Available Templates list and click **Finish**.

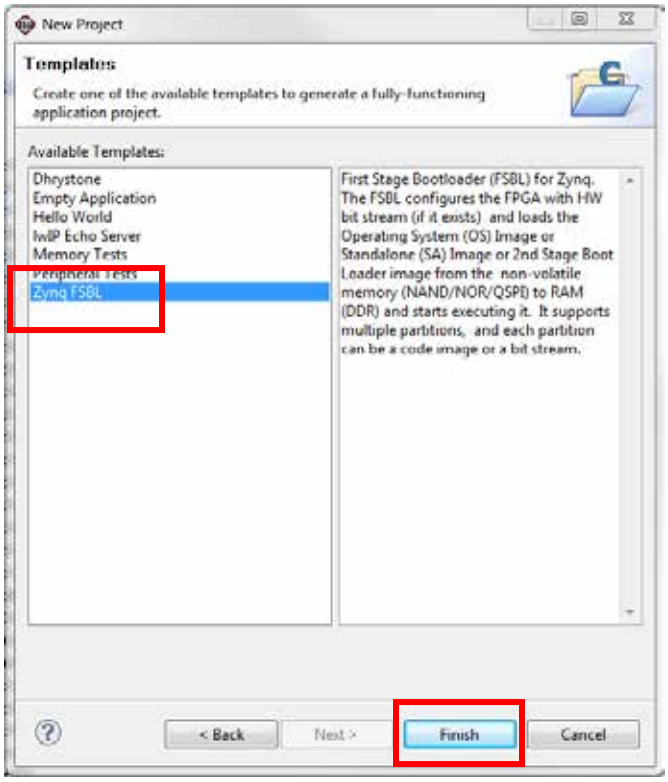


Figure 8 – First Stage Boot Loader Template

The SDK will create a zynq_fsbl_0 project and a zynq_fsbl_0_bsp (Board Support Package) project, and will automatically build the software.

12. The elf file for the bootstrap loader is created at:

    C:\ZedBoard\Zynq_Ubuntu\cf_adv7511_zed\SDK\SDK_Workspace\zynq_fsbl_0\Debug\zynq_fsbl_0.elf

    Using Windows Explorer, copy the zynq_fsbl_0.elf and browse to the folder created earlier to hold the system.bit file:

    C:\ZedBoard\Zynq_Ubuntu\bootfiles

    Paste the elf file in at that location. We will be using it later when we create a boot image for the ZedBoard.

13. You may close the SDK and XPS applications.

Lab 2 - Create the Second Stage Boot Loader (U-boot)

Lab Overview

Typically when booting up an embedded processing system, there are a series of low-level device-specific operations that are executed to bring the system to a basic operating state. Tasks to be performed may include processor register initialization, memory initialization, peripheral detection and verification, and cache activation. At this stage there are very few resources available, and hence the bootstrap or first-stage loader and its associated software must be as compact as possible.

Once the processing system is fully operational, more complex operations, such as loading and running an operating system can be performed. This is beyond the capabilities of the bootstrap loader, so as its final task it loads and passes control to a larger and more capable second stage loader. For the purposes of running Linux on our embedded system, U-boot provides all the capabilities needed to act as our second stage.

Why create the second stage image before we have the first stage loader in place? The initial bootstrap loader image must contain the executable for the second stage loader so that it may complete its final task. Therefore, before we can create the first stage image, we must have the second stage boot loader built.

When you have completed this lab, you will know how to:

- Retrieve U-Boot source code from the Xilinx repository
- Configure U-Boot for the ZedBoard target
- Build U-Boot for the ZedBoard target

Experiment 1: Clone the Xilinx U-Boot Git Repository

This experiment shows how to clone the Xilinx U-Boot Git repository for Zynq. To successfully complete this lab, you will need Internet access to retrieve the repository information from the Xilinx website.

Experiment 1 General Instruction:

Make a local copy of the Xilinx U-Boot Git repository in your home directory, and check out the branch compatible with the Linux kernel we intend to build.

On your Linux host, enter the following commands:

```
$ cd ~

$ git clone git://git.xilinx.com/u-boot-xlnx.git

$ cd u-boot-xlnx

$ git checkout -b xilinx-v14.4 xilinx-v14.4
```

Experiment 1 Step-by-Step Instructions:

1. If the CentOS virtual machine is not already open, launch the VMware Player application by selecting **Start à All Programs à VMware à VMware Player**. If the CentOS virtual machine is already open, skip ahead to Step 4.

Figure 9 – The VMware Player Application Icon

2. Select the virtual machine named **Ubuntu-CentOS-6.3-amd64-ZedBoard-linux** from the selections on the left and click on the **Play virtual machine** button to the right.

   If prompted for whether the virtual machine has been copied or moved, click on the **Moved** button.

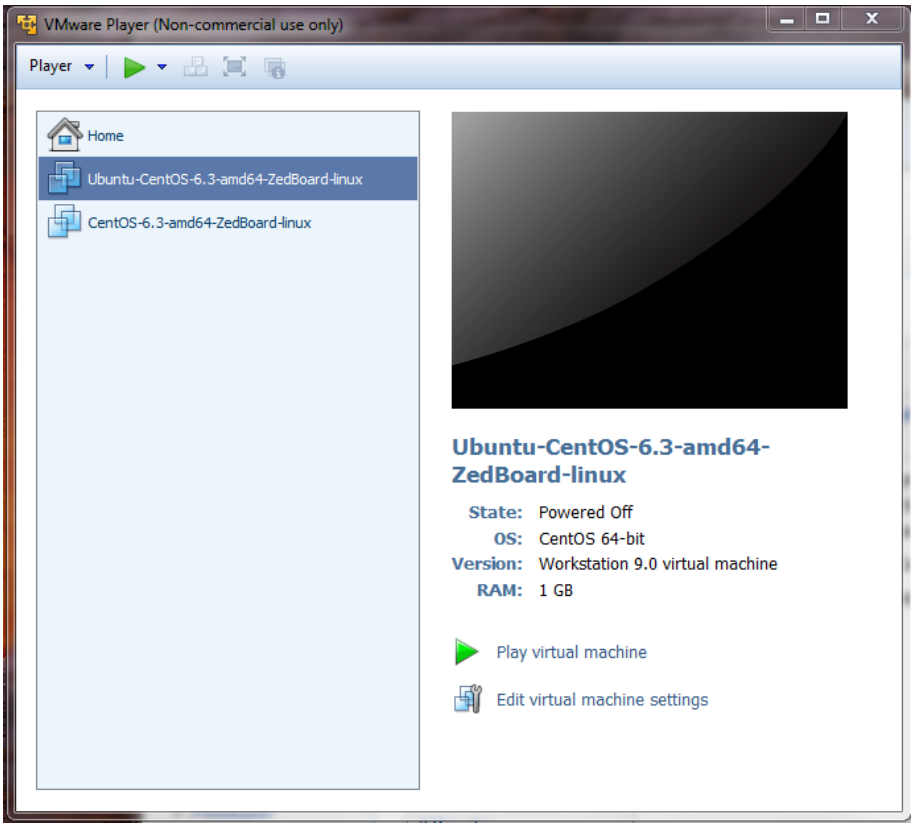


Figure 10 – The VMware Player Application

3. If prompted for a workstation login, click on the user entry **training** and enter the password **Avnet** in order to log into the system.

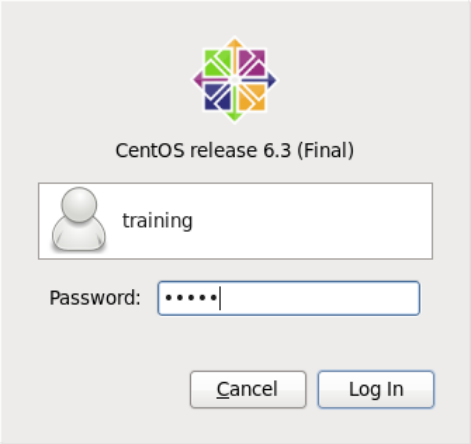


Figure 11 – The CentOS Workstation Login

4. If a terminal is not already open on the desktop of the CentOS guest operating system, open a terminal window through the **Applications à System Tools à Terminal** menu item.

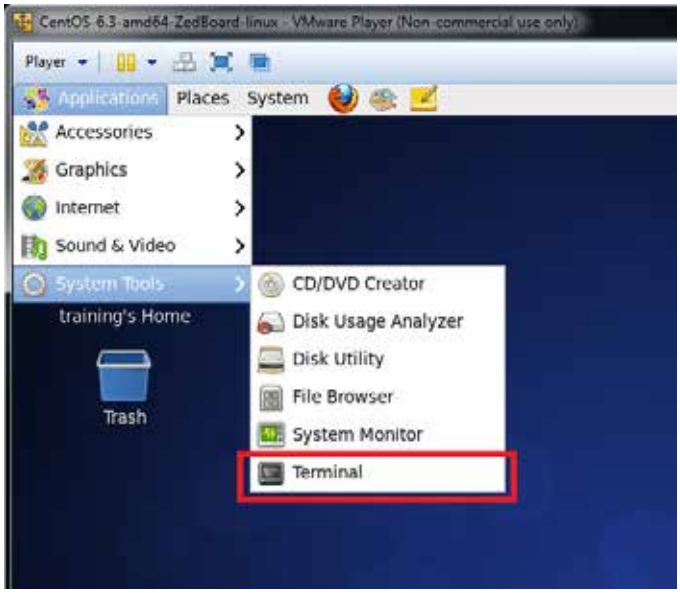


Figure 12 – Launching the CentOS Terminal from the Desktop

5. The Xilinx U-Boot Git repository is located at the following URL:

   git://git.xilinx.com/u-boot-xlnx.git

   To get a working copy of the codebase, clone the remote repository to your local machine. Cloning creates the repository and checks out the latest version, which is referred to as HEAD.

   Use the following Git command to clone the repository.

```
$ git clone git://git.xilinx.com/u-boot-xlnx.git
```

6. Wait until the clone operation completes, this could take 5-20 minutes depending upon your connection speed.

   The clone command sets up a few convenience items for you by:

   - Keeping the address of the original repository
   - Aliasing the address of the original repository as origin so that changes can be easily sent back (if you have authorization) to the remote repository

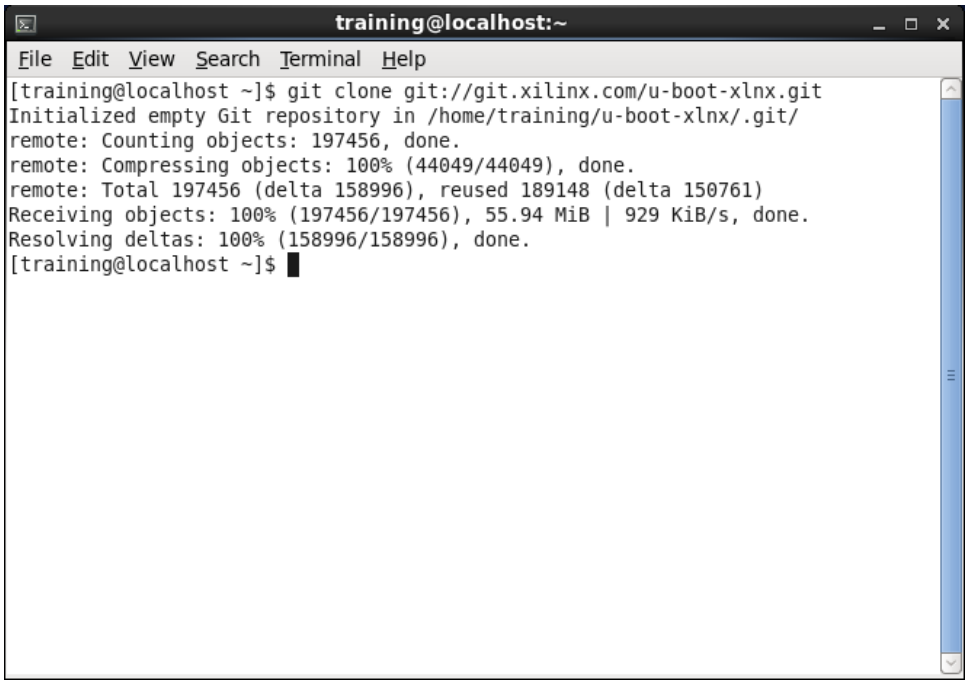


Figure 13 – Using the Git Clone Command

7. Change into the U-Boot source folder.

```
$ cd u-boot-xlnx
```

8. Checkout the code changes related to the 14.2 tools release, which are kept under the **xilinx-14.2-build1** tag. U-boot and Linux kernel versions are closely related, so it is essential that that we match the correct U-boot release to the version of the kernel we intend to use.

```
$ git checkout -b xilinx-v14.4 -build1-trd xilinx-v14.4
```

Experiment 2: Configuring U-Boot for the ZedBoard Target

A working copy of the U-Boot repository is now available on the local development machine and can be used to configure the source tree so that it can be built for a ZedBoard target platform.

U-Boot, like Linux, is maintained for many different processor systems and must be configured for a designated target platform before being compiled. The make files included in the U-boot source tree look for a target board name supplied on the command line, and use that to locate the corresponding configuration file in the source tree. This is done using the **make config** command format:

make <BOARDNAME>_config

Where **<BOARDNAME>** is the name of the configuration file (but without the .h) found in the **include/configs/** folder.

Experiment 2 General Instruction:

Configure the parameters for building U-Boot source so the executable created is suitable for the Avnet ZedBoard. Edit the **zynq_zed.h** file to adjust the boot parameters for our boot requirements. On your Linux host, enter the following commands:

```
$ cd ~/u-boot-xlnx/

$ make distclean

$ make zynq_zed_config
```

Experiment 2 Step-by-Step Instructions:

1. If the CentOS virtual machine is not already open, launch the VMware Player application by selecting **Start à All Programs à VMware à VMware Player**. If the CentOS virtual machine is already open, skip ahead to Step 4.

Figure 14 – The VMware Player Application Icon

2. Select the virtual machine named **Ubuntu-CentOS-6.3-amd64-ZedBoard-linux** from the selections on the left and then click on the **Play virtual machine** button to the right.

If prompted for whether the virtual machine has been copied or moved, click on the **Moved** button.

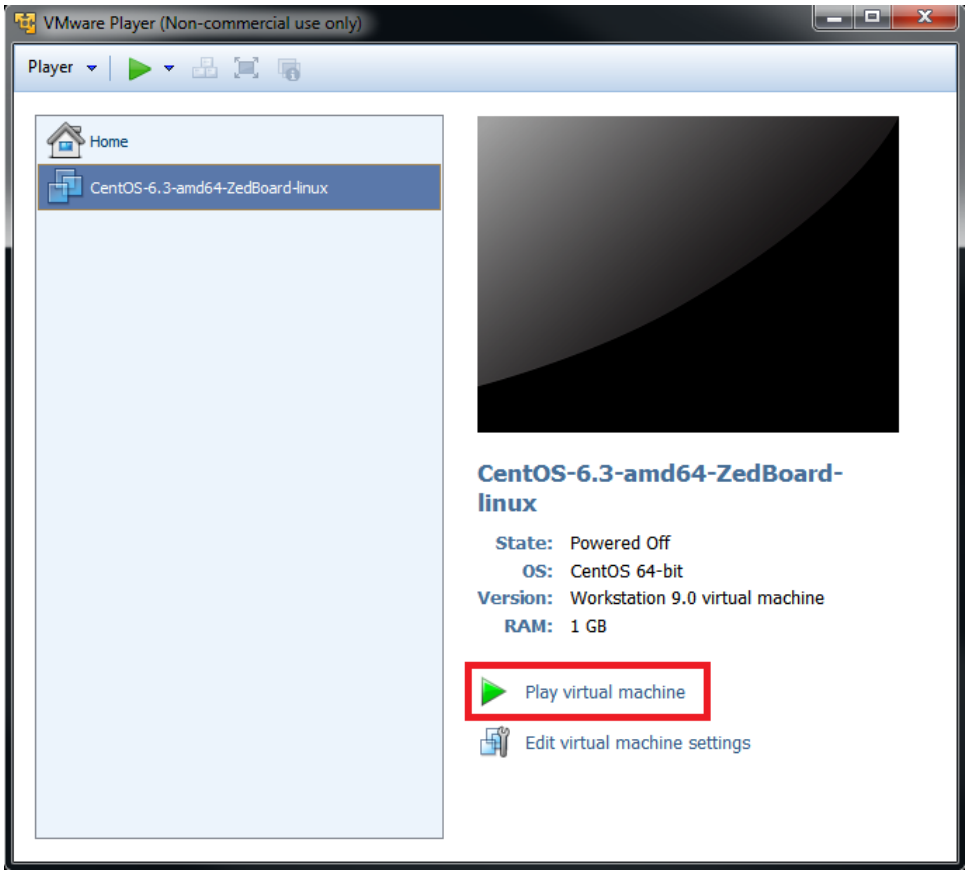


Figure 15 – The VMware Player Application

3. If prompted for a workstation login, click on the user entry **training** and enter the password **Avnet** in order to log into the system.

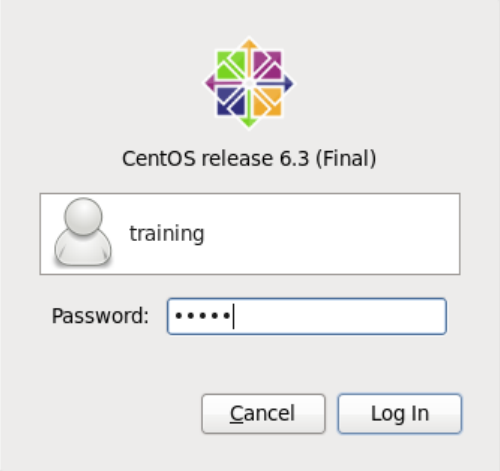


Figure 16 – The CentOS Workstation Login

4. If a terminal is not already open on the desktop of the CentOS guest operating system, open a terminal window through the **Applications à System Tools à Terminal** menu item.

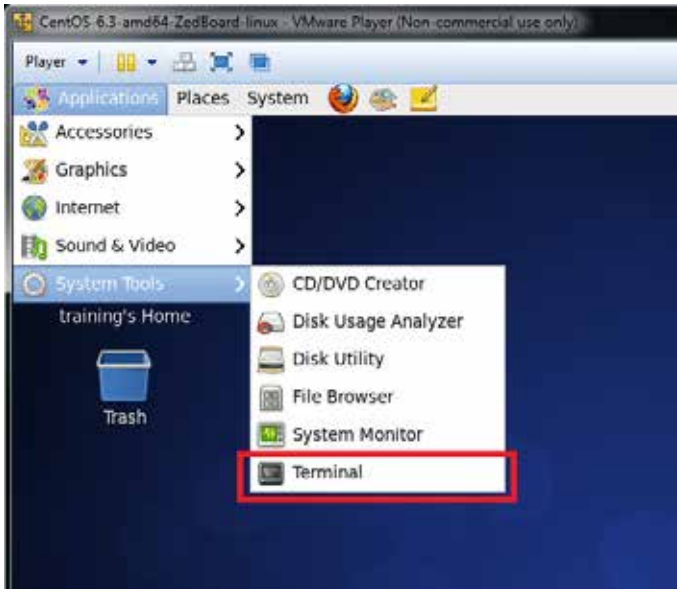


Figure 17 – Launching the CentOS Terminal from the Desktop

5. Change from the home directory into the U-Boot source directory.

```
$ cd u-boot-xlnx/
```

6. For good measure (sometimes a necessity) run a make distribution clean command against the U-Boot source code. This command will remove all intermediary files created by config as well as any intermediary files created by make and it is a good way to clean up any stale configurations.

```
$ make distclean
```

7. A ZedBoard configuration is included for the ZedBoard target and the ZedBoard specific header can be seen in the **/include/configs/zynq_zed.h** file. This file references a second file where parameters common to all Zynq boards are defined. This file is:

   /include/configs/zynq_common.h.

   You will need to open the zynq_common.h in the editor of your choice and edit the sdboot default configuration line (search for **sdboot** to locate it) that copies the root file system from the SD card to memory. This is how Xilinx boots their kernel, but the ADI kernel uses the root file system in place on the SD card, so it does not get copied. Change the sdboot entry to the following:

```
"sdboot=echo Copying Linux kernel from SD to RAM...RFS in ext4;" \
    "mmcinfo;" \
    "fatload mmc 0 0x3000000 ${kernel_image};" \
    "fatload mmc 0 0x2A00000 ${devicetree_image};" \
    "bootm 0x3000000 - 0x2A00000\0" \
```

   Save the file and close your editor.

8. Configure U-Boot for the ZedBoard target by using our modified ZedBoard default configuration.

```
$ make zynq_zed_config
```

Experiment 3: Building U-Boot from the Configured Source

Now that the source tree has been configured for the ZedBoard target platform, it can be built using the cross toolchain. The resulting executable file will be created in your working directory with the name **u-boot**. You will need this file for inclusion in the first-stage loader image created in Lab 4 - Create the First Stage Boot Loader.

Experiment 3 General Instruction:

Build U-Boot for the ZedBoard target platform using the cross toolchain. The source configuration was performed in the previous experiment, so the executable file can be built with a single command. In your Linux system, enter:

```
$ cd ~/u-boot-xlnx

$ make
```

Experiment 3 Step-by-Step Instructions:

1. In the previous experiment, the U-Boot source tree was configured for a ZedBoard target platform.

   First, make sure the **/home/training/u-boot-xlnx/** folder is the current working directory by using the pwd command. If the working directory shown is not the **/home/training/u-boot-xlnx/** folder, change directories to that folder.

```
$ pwd
```

2. Once the working directory is the **/home/training/u-boot-xlnx/** folder, build the U-Boot source with the make command.

   The build process should take anywhere from 5 to 10 minutes to finish and should complete successfully. If the build is successful and the console output looks similar to that shown in Figure 13, proceed to **Step 4**.

```
$ make
```

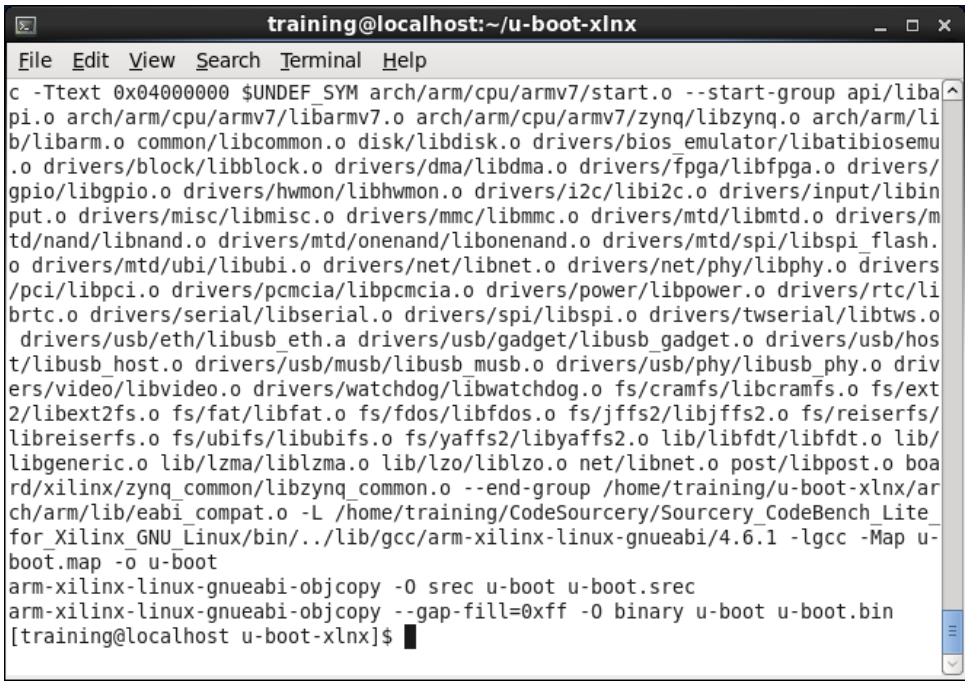


Figure 18 – U-Boot Build Completed

3. If you run into strange errors regarding problems locating your cross toolchain or errors with segmentation faults during the build, perform this step to correct the problem, otherwise skip to the next numbered step.

   The terminal **PATH** environment variable may not be set correctly. One way to resolve this is to pick up the updated user profile using the source command from the **/home/training/** folder.

   Return to the U-Boot source folder, then clean the source tree with **make distclean** and go back to Step 2 to rebuild. This should help resolve any segmentation faults encountered with the toolchain.

```
$ cd ~

$ source .bash_profile

$ cd u-boot-xlnx/

$ make distclean

$ make zynq_zed_config
```

   The terminal PATH environmental variable tells the terminal shell which directories to search for executable files in response to commands issued by a user. This environment variable increases both the convenience and the safety of command line based operating systems and is widely considered to be the single most important environmental variable. The PATH environment variable can be checked by echoing it to the command line. Check it to be sure that it contains a path to the Sourcery CodeBench installed /bin folder.

```
$ echo $PATH
```

4. Once the build has completed successfully, open a file browser window through the **Applications à System Tools à File Browser** menu item.

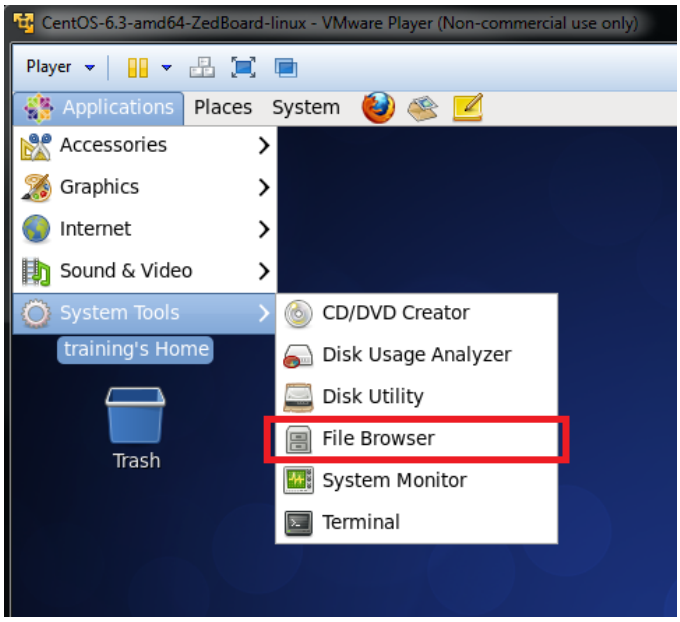


Figure 19 – CentOS File Browser

5. Locate the file **/home/training/u-boot-xlnx/u-boot** which is the target executable file in the Extended Linker Format (ELF) needed to execute on Zynq. Right click on this file and select the **Copy** option which will place the selected file in the Virtual Machine clipboard.

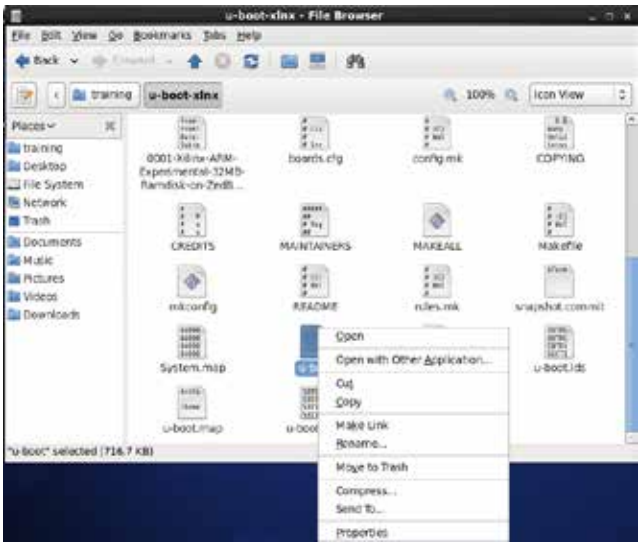


Figure 20 – Copying U-Boot Executable to Virtual Machine Clipboard

6. Paste the U-Boot executable into the host operating system by using Windows Explorer to navigate to the following folder:

C:\ZedBoard\Zynq_Ubuntu\bootFiles

Then paste the **u-boot** file into this folder. Rename the **u-boot** file to **u-boot.elf** so that it has the appropriate ELF extension needed for the SDK tools to see it.

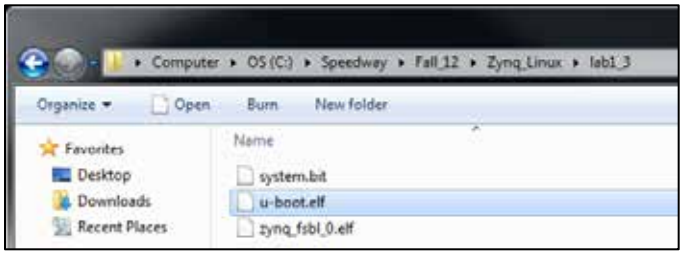


Figure 21 – U-Boot Executable Copied to the Host Machine and Renamed

Lab 3 – Partition the SD Card for Booting

Lab Overview

The ZedBoard will be booted from data contained on an SD card, and once the Linux environment is operational the root file system will also reside on the SD card. In this lab we will prepare the SD card with the required partition formats.

Experiment 1: Format the SD Card for Ubuntu Booting

In this experiment we will create two partitions on the SD card. The first partition will be in FAT32 format, which can be accessed by either a Windows or Linux operating system. This partition will contain the files used for initial bring-up of the ZedBoard. The second partition will be in ext4 format, usable only by a Linux OS. On this partition we will place the root file system, required by the Linux kernel to complete the OS boot process.

Because the second partition is Linux specific, we will prepare the SD card from the CentOS VM using a utility called the Gnome Partition Utility (GParted).

Experiment 1 General Instruction:

Use GParted to create 2 empty partitions on a 4 MB SD card. The first partition will be 52 MB FAT32 format, and the second partition will use the remaining space in ext4 format.

Note: All existing data on the SD card will be destroyed. Make sure you back up any data you need to retain prior to the formatting operation.

Experiment 1 Step-by-Step Instruction:

1. Go to Appendix I and follow the instructions to install GParted. Then enter:
   a. **sudo yum install gparted**
   b. Respond 'y' to all queries

2. With the VM running, insert the SD card into a compatible read/write slot on your computer. If the USB device is registered by Windows, close any pop-ups that appear. From the VM, you may see a window to indicate that a new removable device is available. Click OK to acknowledge and close the window.

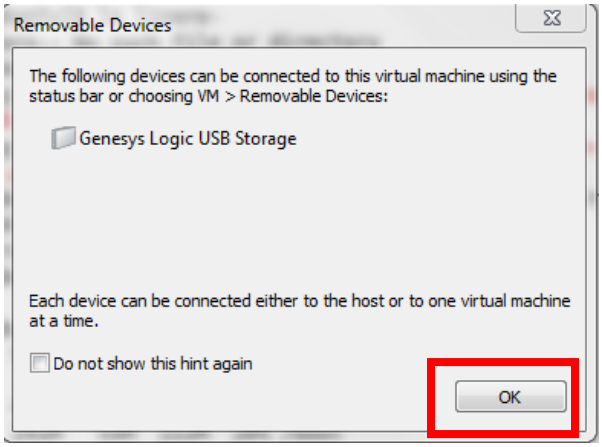


Figure 22 – USB Detection by Virtual Machine

3. In the VM, look at the upper right-hand corner and locate the icon that applies to the USB device. Right-click on the icon and select “Connect” from the drop-down menu.

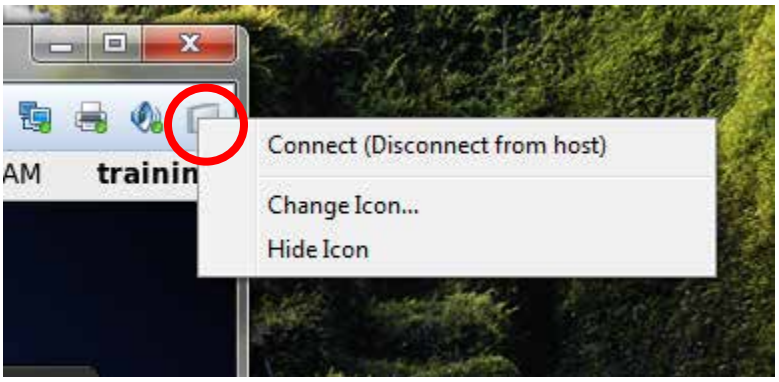


Figure 23 – Connect USB Device to Virtual Machine

4. Click OK to acknowledge the device will be disconnected from the Windows host and connected to the Virtual Machine.

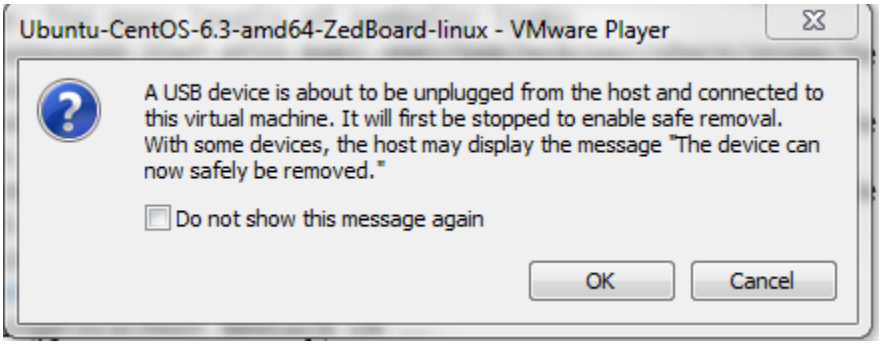


Figure 24 – USB Transfer Warning from Virtual Machine

5. With the device connected to the VM, you may see browser windows activate to view files on a non-blank SD card. Close any browser windows associated with the SD card, as we will be repartitioning and formatting the entire card. **Note that any existing data on the card will be LOST**.

6. To run the GParted Partition Editor, in the VM menu select **Applications | System Tools | GParted Partition Editor**.

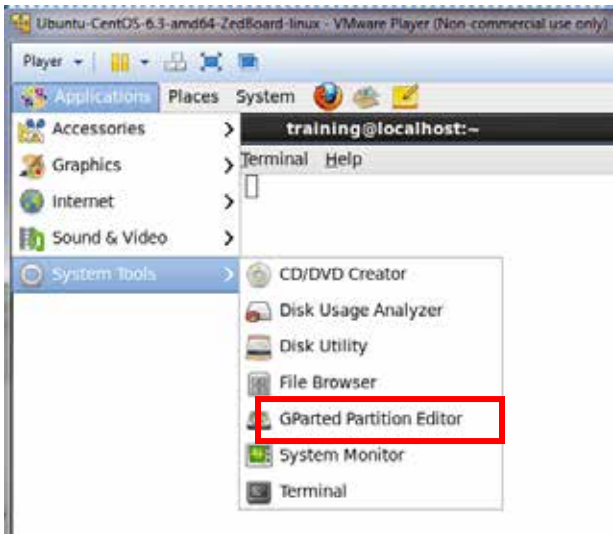


Figure 25 – Launch GParted Partition Editor

7. Enter the root password when prompted. For the CentOS VM created in Avnet Speedways, the password is “Avnet”. Click OK.

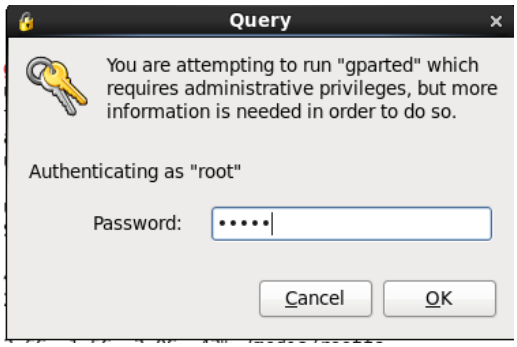


Figure 26 – Enter Root Password

8. GParted will by default display the partitions on your CentOS system. **You DO NOT want to change anything here!** In the upper right, open the drop-down menu to display the available devices, and select the one corresponding to the SD card. This will generally be labelled as **/dev/sdb**, with a size shown that is slightly smaller than the full size of your SD card. Click on this device to select it.

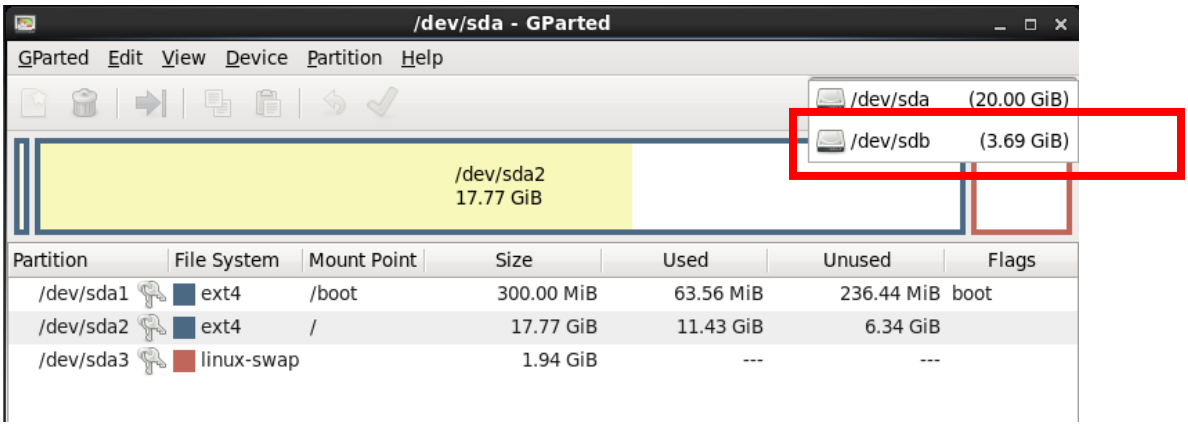


Figure 27 – GParted SD Card Selection

9. If your SD card has been used before, it may contain a file system already. Any existing system not specifically used for the Ubuntu boot process should be removed so we can start with a clean slate. In the example below we have a 4 MB SD card with a FAT32 file system that was created on a Windows 7 system. If the card is empty, you may skip to ***step 15***.

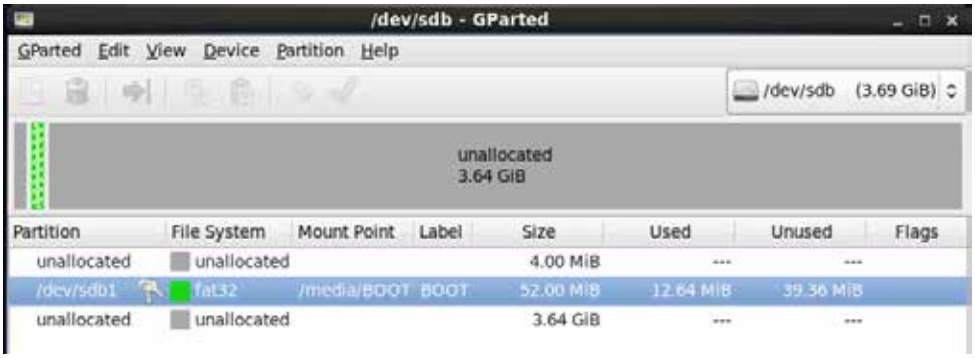


Figure 28 – Existing SD Card Partitions

10. Right-click on the file system to be removed and select **Unmount**.

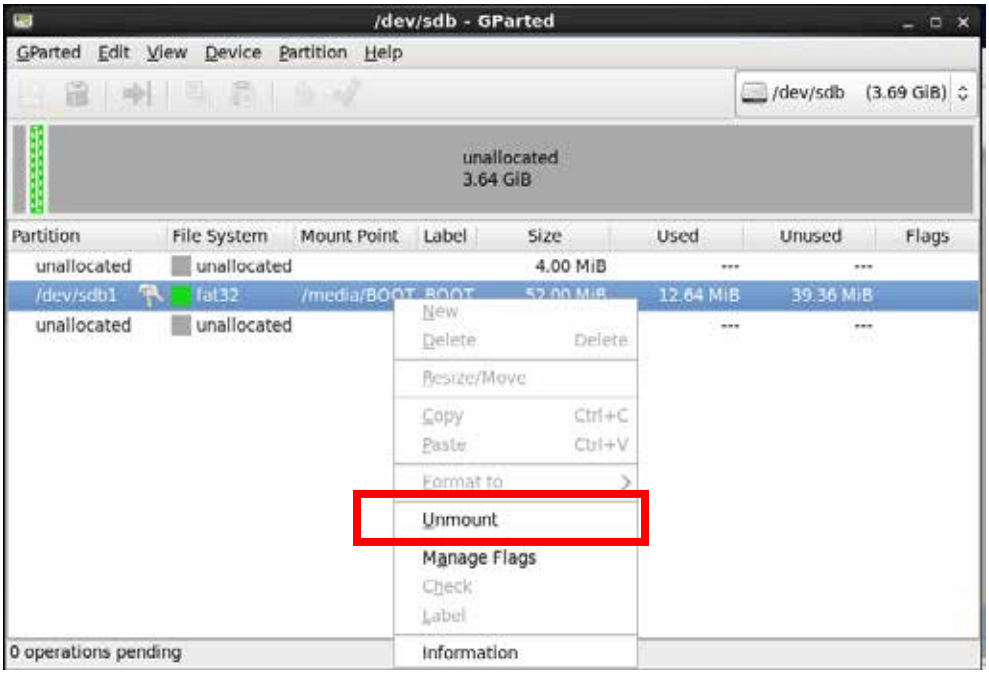


Figure 29 – Unmount an Existing Partition

11. Right click on the unmounted file system and select **Delete**. You will see a pending Delete action appear in a new window at the bottom of GParted. Actions are queued and then executed serially with a single command.

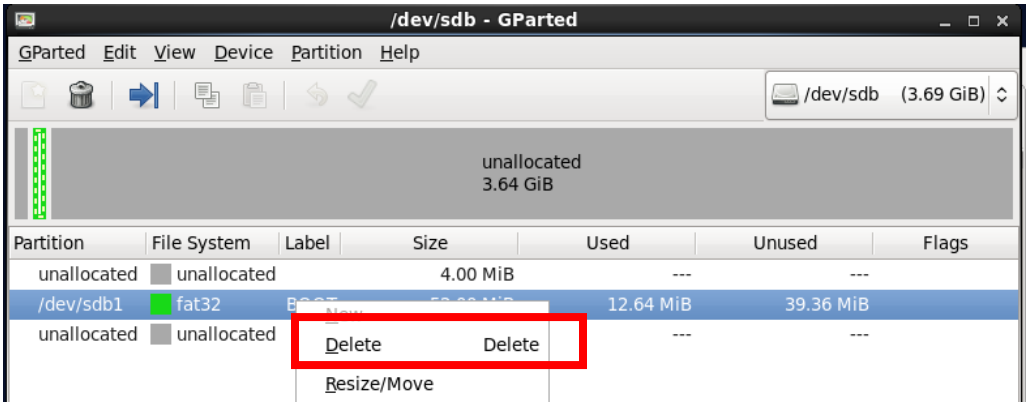


Figure 30 – Delete Existing SD Card Partition

12. You will now see GParted as shown below, with the entire SD card unallocated and one action pending. Click on the Edit menu and select **Apply All Operations** from the dropdown list.

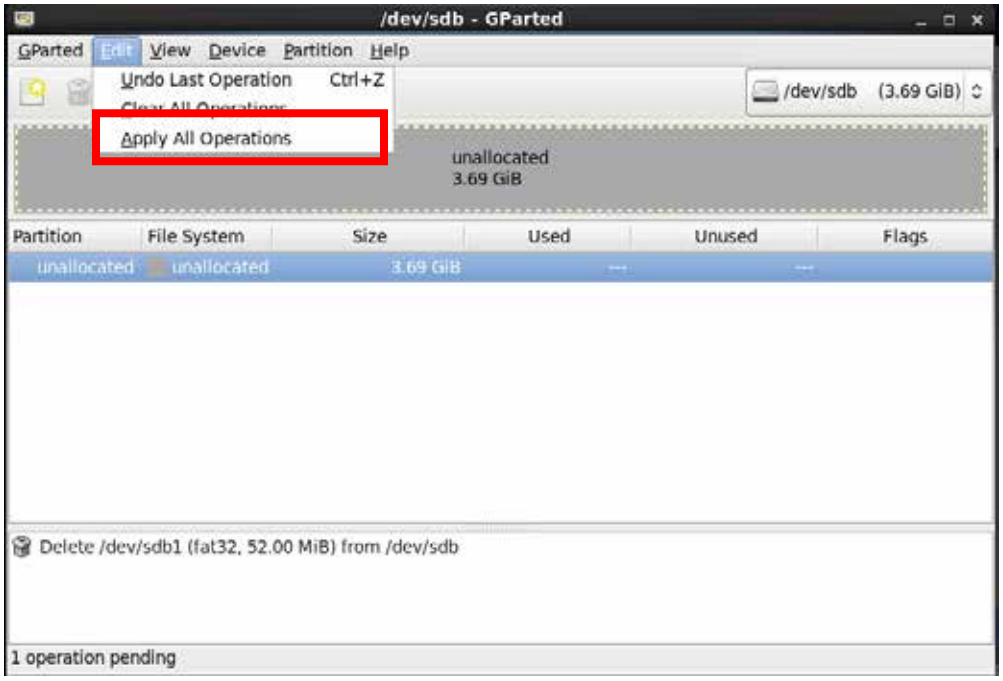


Figure 31 – GParted Perform Pending Operations

13. Click Apply to proceed with the operation. All data on the device will be permanently lost.

Figure 32 – GParted Verify Partition Operations

NOTE: If you apply operations in GParted and you get back an error saying the resource is busy, make sure you have no open browser windows for the /media directory. Close them if you do, restart GParted, and the actions should work.

14. Once the operations have completed, you will see a message indicating that all operations were successful. Click Close.

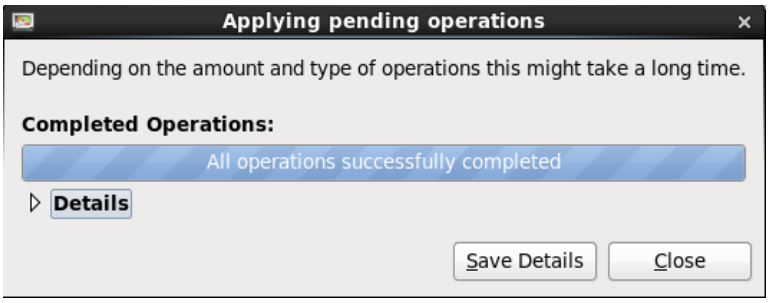


Figure 33 – GParted Operations Successful

15. Your SD card should now appear with the full space completely unallocated. Right click on the unallocated space and select **New**.

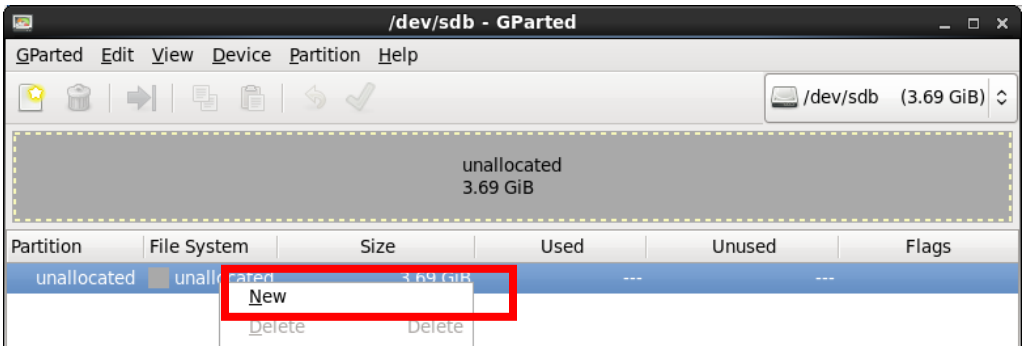


Figure 34 – Create New SD Card Partition

16. In the Create New Partition window, enter the parameters shown below and click the **Add** button. The operation will queue up in the lower window in GParted.

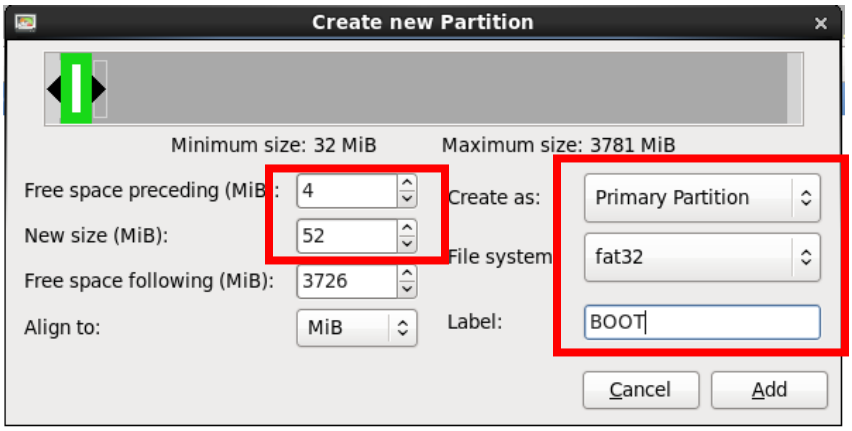


Figure 35 – Create FAT32 Partition

17. Right-click on the unallocated space below your new FAT32 file system and select **New**.

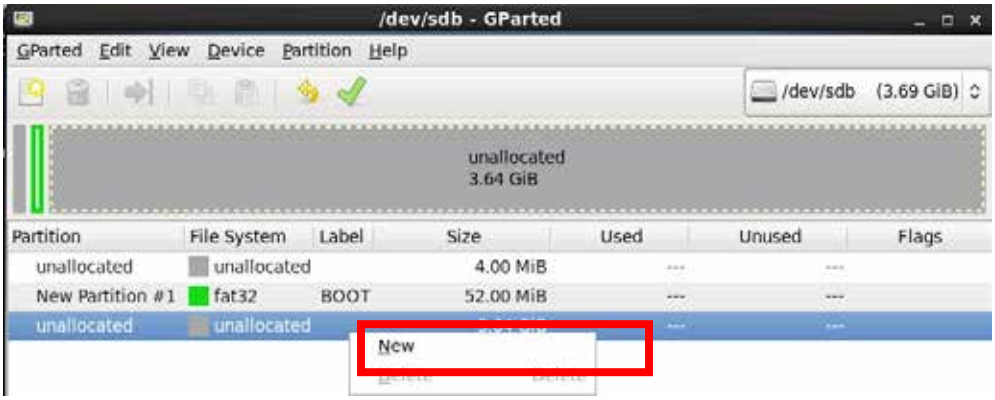


Figure 36 – Create New SD Card Partition

18. In the Create New Partition window, enter the parameters shown below and click the **Add** button. The operation will queue up in the lower window in GParted. The New size will automatically default to the remaining free space on your SD card, which is what we want.

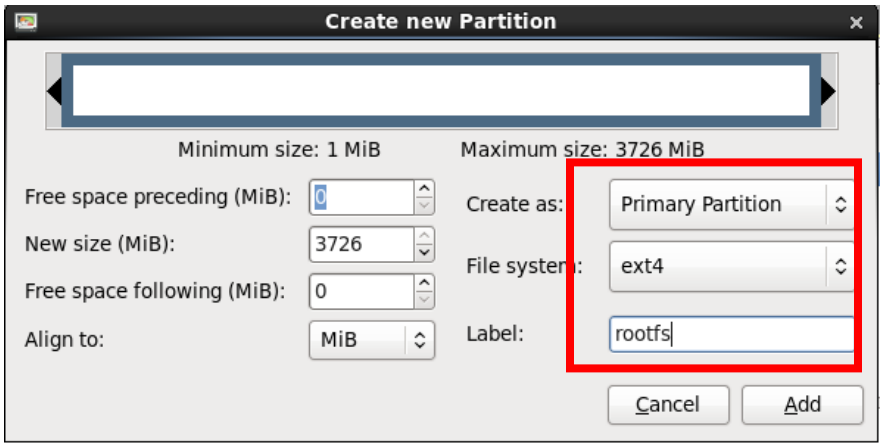


Figure 37 – Create New ext4 Partition

19. The GParted window should now look very similar to the one shown below. If you are satisfied that you will be creating 2 partitions, the first a fat32 of about 52 MB called **BOOT**, and the second an ext4 partition of 3.6 GB or more called **rootfs**, then select Edit from the menu and click on **Apply All Operations**.

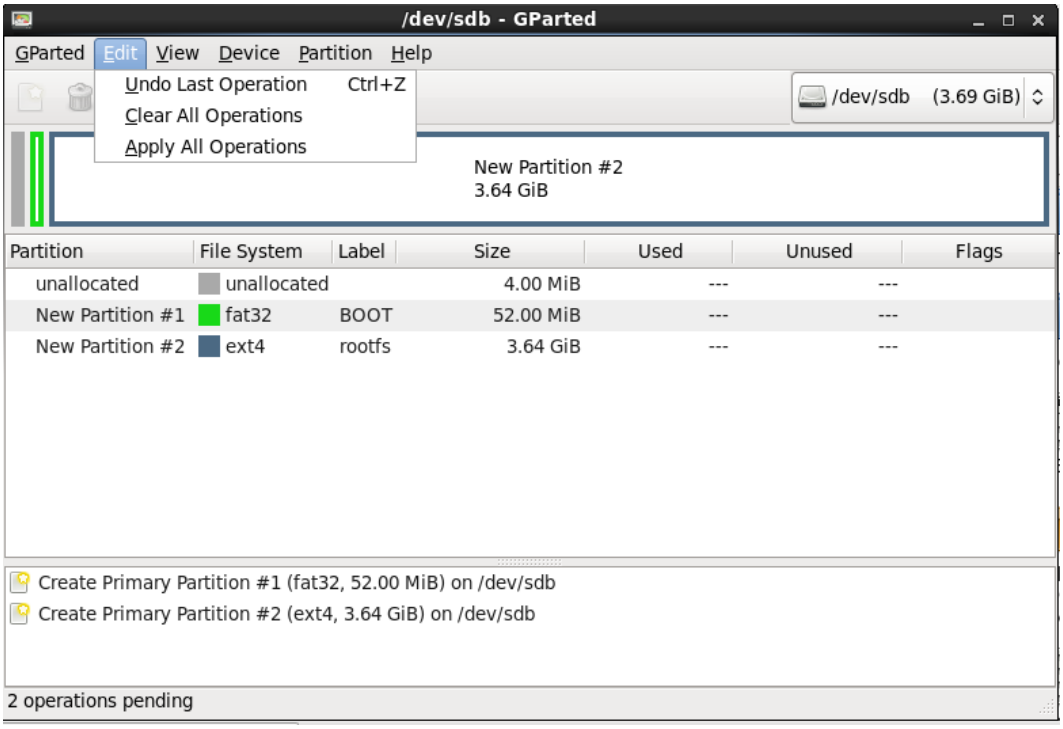


Figure 38 – Partition SD Card with FAT32 and ext4

20. Click **Apply** to proceed with the operation. All data on the device will be permanently lost. Depending on the size of your SD card, the operations may take a few minutes or more to complete.

Figure 39 – Verify Partition Creation Operation

21. Once the operations have completed, you will see a message indicating that all operations were successful. Click **Close**.

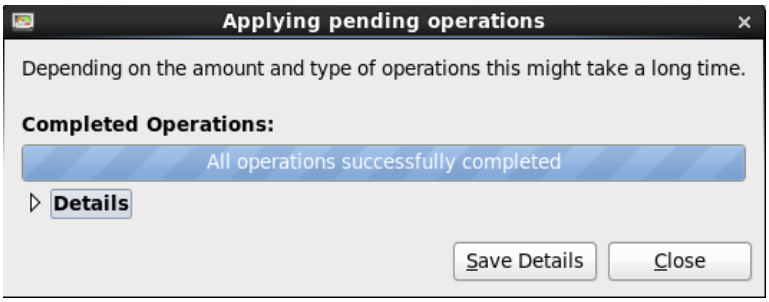


Figure 40 – Partition Creation Successful

22. Your SD card is now ready for the files that will be used to boot Ubuntu on the ZedBoard. Your partition information should look similar to the information shown below. Right-click on the drive icon at the upper right of the VM window and select **Disconnect**.

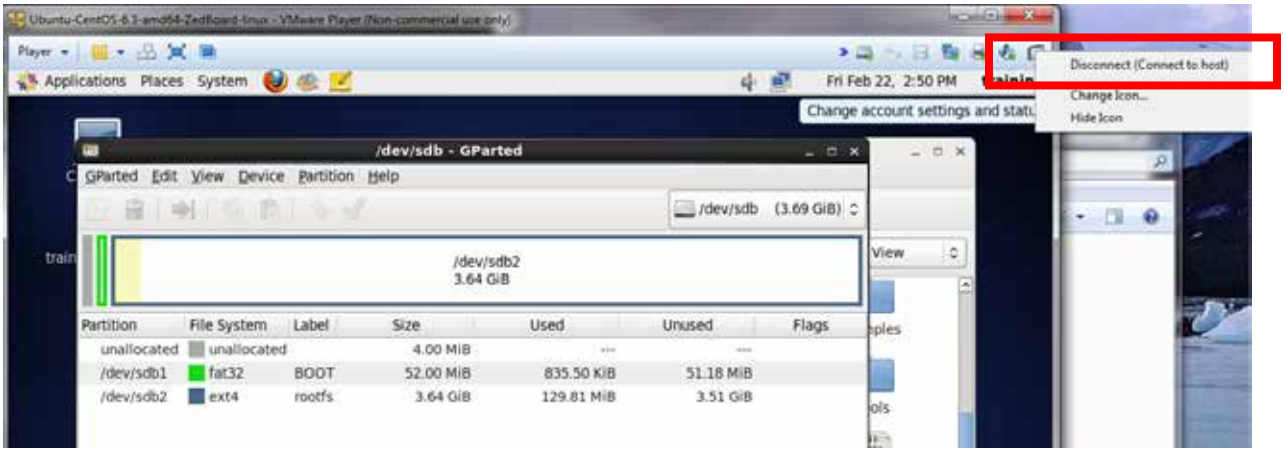


Figure 41 – Disconnect USB Device

23. Close the GParted window.

24. Back in Windows you can open Windows Explorer to locate the removable drive, which will now appear as a FAT32 drive named BOOT. Note that Windows does not recognize the ext4 file system format, so it is not available from the Windows host. Right-click on BOOT, select **Eject** and you can remove the SD card. The card is now ready for use in subsequent steps.

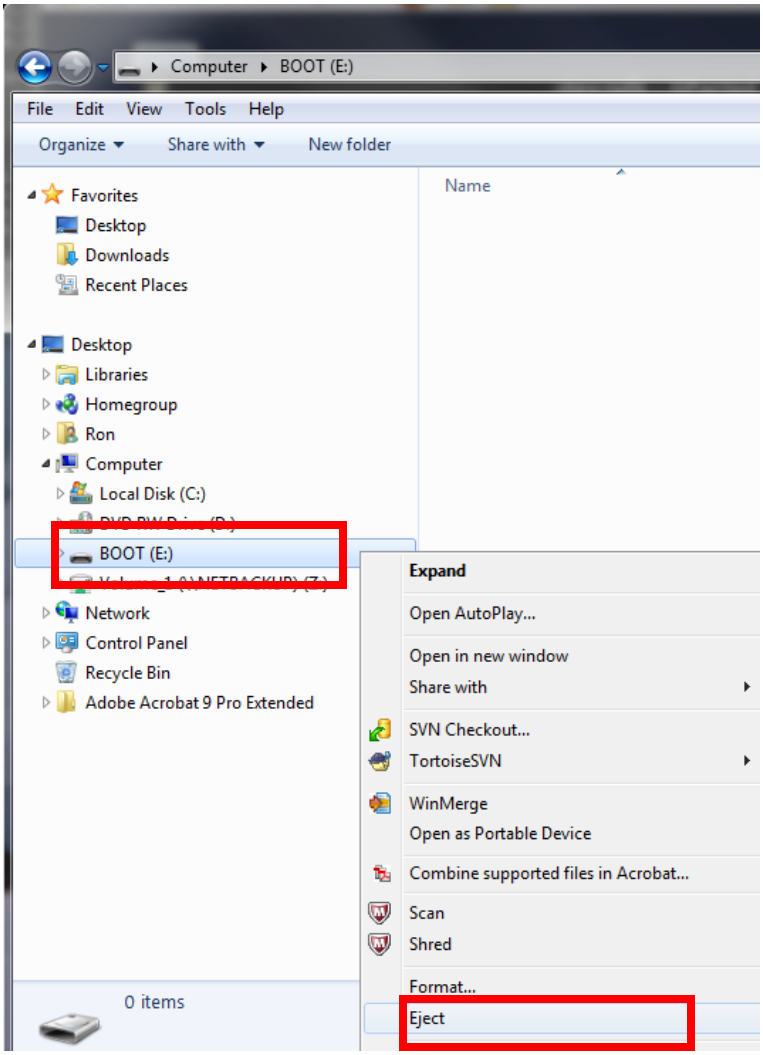


Figure 42 – Remove SD Card from Host

Lab 4 - Create the First Stage Boot Loader

Lab Overview
At this point all the components necessary for initial bring-up of the ZedBoard have been created. We have the hardware bitstream file and the First Stage Boot Loader program, created in "Lab 1 - FPGA Hardware Platform". In the second lab we configured and built U-boot for the ZedBoard, to serve as the second stage loader that will ultimately launch the operating system.

In this lab we will use the Xilinx SDK to combine the three components into a single boot image, and test that image to verify we can complete a first stage boot and see a U-boot prompt on the console.

When you have completed this lab, you will know how to do the following:

- Create a boot image for ZedBoard
- Boot ZedBoard to U-Boot prompt

Experiment 1: Create the Boot Image

This experiment shows how to export a boot image using SDK.

Boot Image Format

The Zynq BootROM is capable of booting the processor from several different non-volatile memory types but it requires a data structure referred to as the Boot Image Format (BIF) for instructions on how to parse the different boot components. For the most part, simple Linux systems require only three components within the boot image:

1. FSBL (Stage 1 boot loader)
2. Programmable Logic (PL) Hardware Bitstream
3. U-Boot (Stage 2 boot loader)

Refer to Xilinx UG585 document, the Zynq Technical Reference Manual, for further details.

Boot Medium

Although ZedBoard can boot from either Quad-SPI Flash or SD card, for the purposes of this lab, we will use the SD card as the boot medium. This gives us the advantage of being able to write directly to the FAT32 partition on the SD card from a PC.

Experiment 1 General Instruction:

Launch Xilinx Software Development Kit (SDK) and open the workspace from the project in the Lab 1.1 directory. Use SDK **Create Zynq Boot Image** tool to create the Zynq boot image file.

Experiment 1 Step-by-Step Instructions:

1. Launch Xilinx Software Development Kit (SDK). **Start à All Programs à Xilinx Design Tools à ISE Design Suite 14.4 à EDK à Xilinx Software Development Kit**.

Figure 43 – The SDK Application Icon

2. Set or switch the workspace to:

   C:\ZedBoard\Zynq_Ubuntu\cf_adv7511_zed\SDK\SDK_Workspace

   and click the **OK** button.

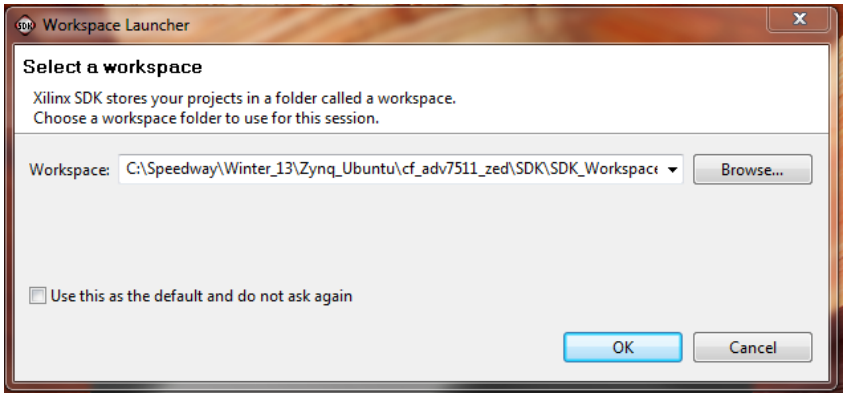


Figure 44 – Switching to the Appropriate SDK Workspace

3. Once SDK is open, launch the **Create Zynq Boot Image** tool using the **Xilinx Tools à Create Boot Image** menu item.

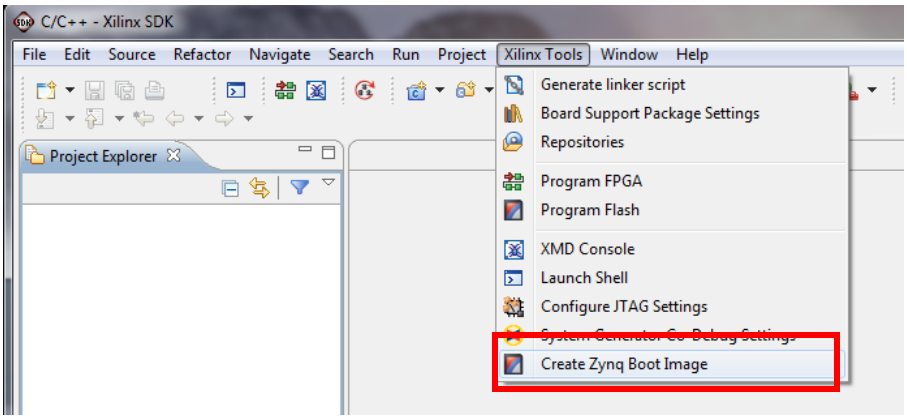


Figure 45 – Launching the Create Zynq Boot Image Tool

4. A new Boot Image Format (BIF) file must be created. In the **Bif file** drop down menu, select the **Create a new bif file...** option.

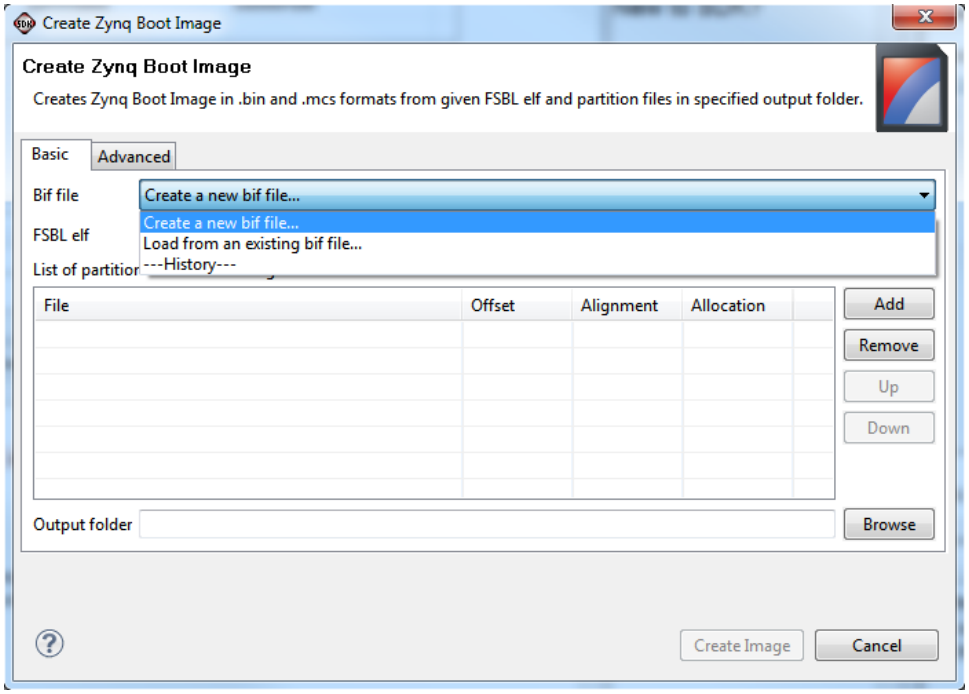


Figure 46 – Creating a New BIF File

5. Specify the **FSBL elf** file by browsing to the following file:

   C:\ZedBoard\Zynq_Ubuntu\bootFiles\zynq_fsbl_0.elf

   Keep in mind the importance of order for files placed into a boot image. Specify the hardware bitstream by clicking the **Add** button and selecting the file below. Click the **Open** button to add this file to the list of partitions in the boot image:

   C:\ZedBoard\Zynq_Ubuntu\bootFiles\system.bit

   Specify the application executable last (in our case it is the second stage boot loader U-Boot) by clicking the **Add** button and selecting the following file. Then click the **Open** button to add this file to the list of partitions in the boot image:

   C:\ZedBoard\Zynq_Ubuntu\bootFiles\u-boot.elf

   Browse to the following **Output folder**:

   C:\ZedBoard\Zynq_Ubuntu\bootFiles

   Click the **Create Image** button which will background launch Bootgen, a tool for constructing boot images for Zynq-7000 AP SoC configuration.

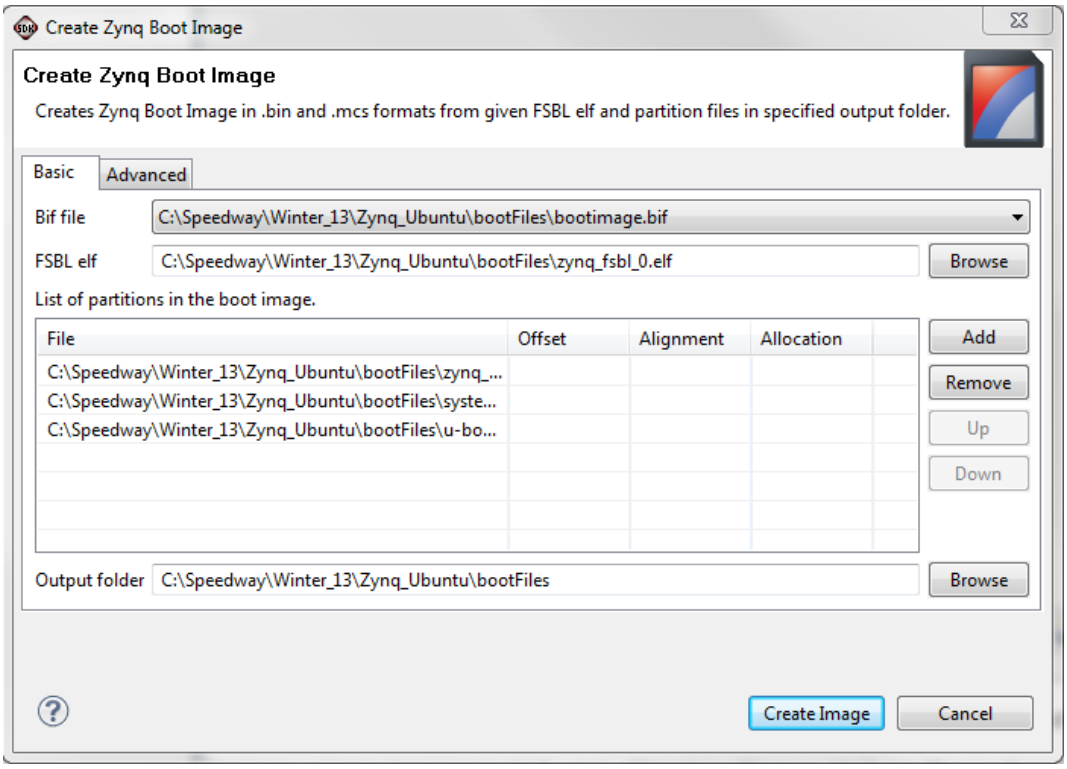


Figure 47 – Selecting Appropriate Boot Image Partitions

6. Bootgen merges the BIT and ELF files into a single boot image with the format defined in the Boot Image Format (BIF) file to be loaded into Zynq devices at boot time. Evaluate the output seen in the SDK console; notice how bootgen is called against the **bootimage.bif** file, which contains the format which defines which files are integrated and what order they are added to the binary output file **u-boot.bin**.

   It is important to note that the order of the images should always be the FSBL first, followed by the Programmable Logic bitstream, and then finally the software application file (in our case it is the second stage boot loader). The BIF file is automatically created by the Zynq Boot Image Creation tool prior to the Bootgen operation.

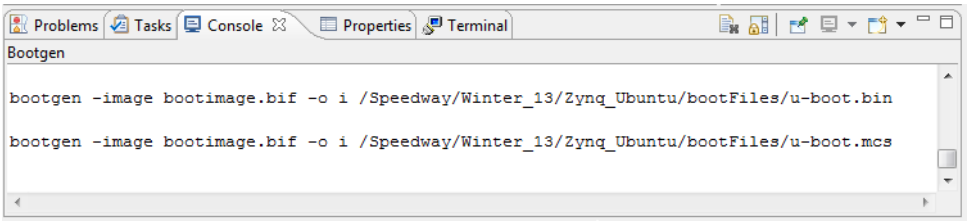


Figure 48 – Bootgen Running in the Background

At this point you may close the Xilinx SDK.

7. BootGen will create a **u-boot.bin** file in the specified output folder. The BootROM on Zynq will only be able to locate a boot image file on the SD card if it is named **boot.bin**. Rename the **u-boot.bin** file to the **boot.bin** filename.

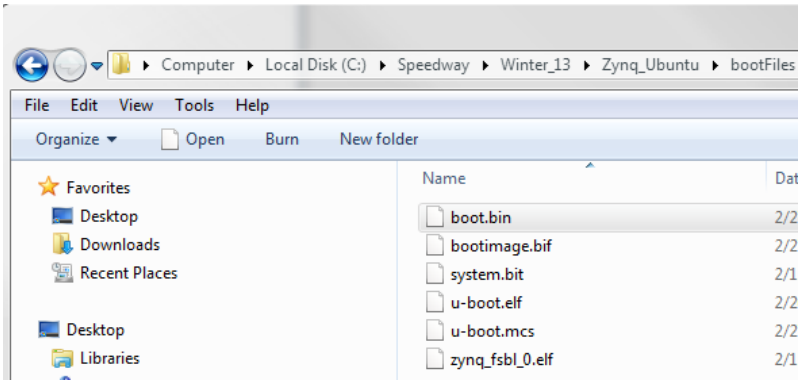


Figure 49 – The Resulting Boot Image file Renamed to boot.bin

8. Insert the SD card into the PC or SD card reader and wait for it to enumerate as a Windows drive. If prompted by Windows when inserting the SD card, select the **Continue without scanning** option.

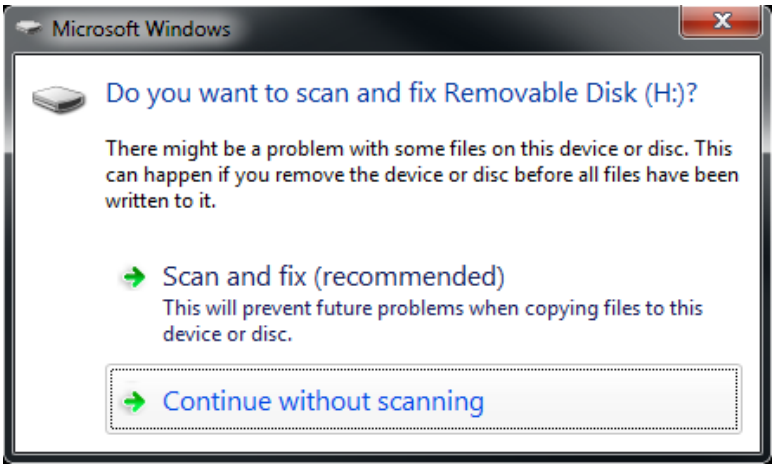


Figure 50 – Windows Prompt for Scanning and Fixing an SD Card

9. Copy the **boot.bin** file from the **bootfiles** folder to the top level of the SD card. Replace any existing versions of the **boot.bin** file that may be on the SD card.

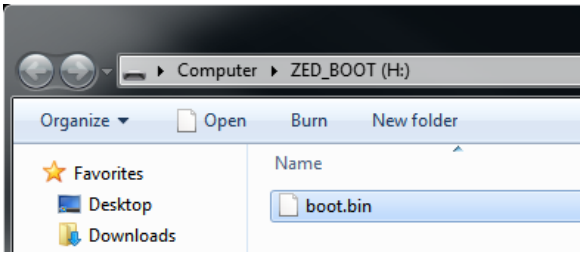


Figure 51 – The Boot Image File Copied to the SD Card

Note: You may encounter problems writing to the SD card even when the write-protect button is in the "off" position. With the SD card face up, the tiny white plastic switch on the left side should be positioned towards the card edge connector (the shortest side).

For some SD cards, on the opposite side from the switch, there is a tiny notch cut out. This notch allows the SD cards to work in portable consumer electronics, but may not work in USB card readers or computers. Some SD cards have this extra built-in feature for copy protection, which makes it useless for custom applications.

A strategically placed piece of electrical tape can work in overcoming this feature. Be very precise in placing the tape over the notch, but not over the

adjacent brass electrical contact. Place the "modified" SD card in your reader, and you should have no further problems with formatting, reading or writing.

Once you have bypassed the protection feature of the SD card, install your files as needed. If you get the same error again after applying the tape, add a second layer. Double check that you did not cover the connection edge and that the plastic tab is "forward"; closest to the edge you insert. Sometimes it takes several tries, but it will usually work with some added patience. If you have existing files on the SD card, insert it, right click on the files you wish to remove or overwrite, and select the **Properties** option. Then unselect the **Read Only** check by the box. Try again to delete or add files; if you are prompted with a message indicating that the disk is write protected, check the tape again.

Experiment 2: Boot ZedBoard Using New Boot Image

ZedBoard can now be booted using the Boot Image that was copied to the SD card in the previous exercise.

Experiment 2 General Instruction:

Boot ZedBoard using the SD card with the Zynq boot image and observe the terminal output on a console (Tera Term or similar) on your host system.

Experiment 2 Step-by-Step Instructions:

1. Connect 12 V power supply to ZedBoard barrel jack (J20).

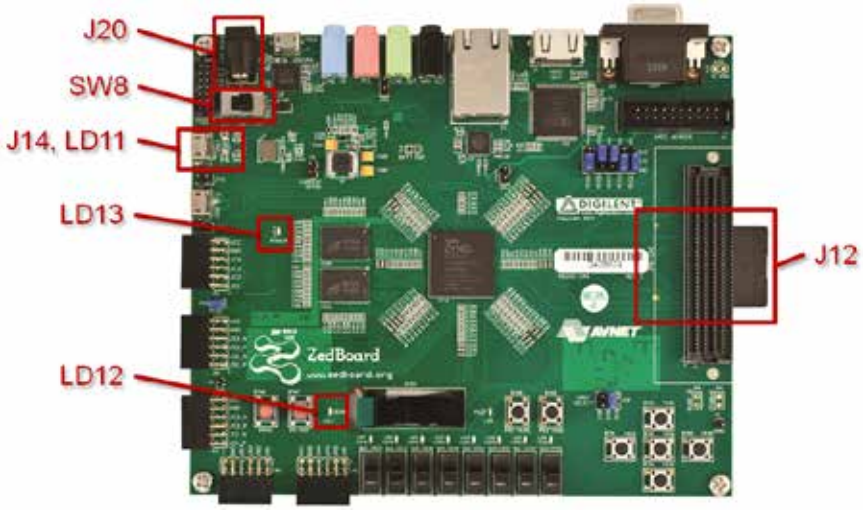


Figure 52 – ZedBoard Hardware Reference

2. Connect the USB-UART port of the ZedBoard (J14), labeled UART, to a PC using the MicroUSB cable.

3. Insert the 4GB SD card included with ZedBoard into the SD card slot (J12) located on the underside of ZedBoard PCB.

4. Verify the ZedBoard boot mode (JP7-JP11) and MIO0 (JP6) jumpers are set to SD card mode as described in the Hardware Users Guide:

   http://www.zedboard.org/sites/default/files/ZedBoard_HW_UG_v1_6.pdf

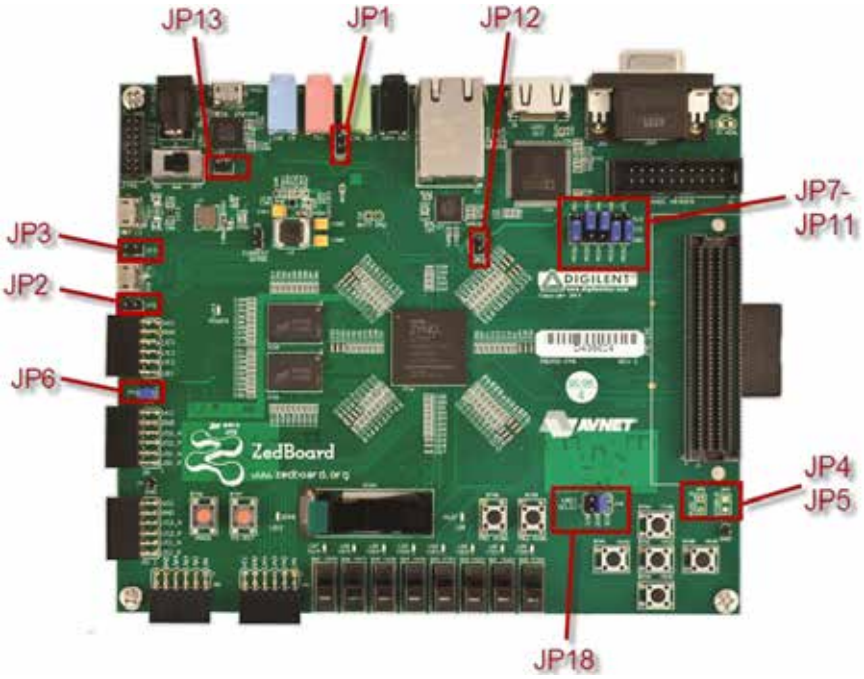


Figure 53 – ZedBoard Jumper Settings

5. Turn power switch (SW8) to the ON position. ZedBoard will power on and the Green Power Good LED (LD13) should illuminate.

6. The PC may pop-up a dialog box asking for driver installation.

   ZedBoard has a USB-UART bridge based on the Cypress CY7C64225 chipset. Use of this feature requires that a USB driver be installed on your Host PC.

   If Windows recognizes the USB-UART and loads the software driver, go ahead and proceed to the next step. However, if the host PC did not recognize the USB-UART and enumerate it as a COM port device refer to the USB-UART Setup Guide document in the link below for instructions on installing this driver. When driver installation is complete, continue to the next step.

   http://www.zedboard.org/sites/default/files/CY7C64225_Setup_Guide_1_1.pdf

7. Wait approximately 15 seconds. The blue Done LED (LD12) should illuminate. Use the Windows Device Manager to determine the COM Port.

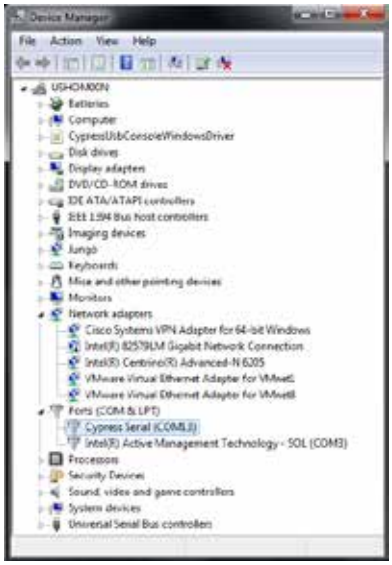


Figure 54 – Device Manager Showing Enumerated USB-UART as COM13

Note: Each unique USB-UART device attached will enumerate under the next available COM port. Here in this example, the Cypress CY7C64225 USB-UART device is enumerated as COM13.

8. On the PC, open a serial terminal program. For this demo, Windows 7 was used which does not come with a built in terminal application such as HyperTerm. Tera Term was used in this example which can be downloaded from the Tera Term project on the SourceForge Japan page:

   http://ttssh2.sourceforge.jp

9. Once Tera Term is installed, Tera Term can be accessed from the desktop or Start menu shortcuts.

Figure 55 – Tera Term Icon

10. To configure baud rate settings, open the Serial Port Setup window from the **Setup à Serial port** menu selection. Select the USB-UART COM port enumeration that matches the listing found in Device Manager.

    Also set the Baud rate option to **115200**, the Data width option to **8-bit**, the Parity option to **none**, the Stop bit option to **1 bit**, and the flow control to **none**.

    Finally, assign the transmit delay parameters to **10 msec/char** and **100 msec/line**, and then click **OK**. This setting will help with later lab exercises were many lines of text will be sent to the console in rapid succession.

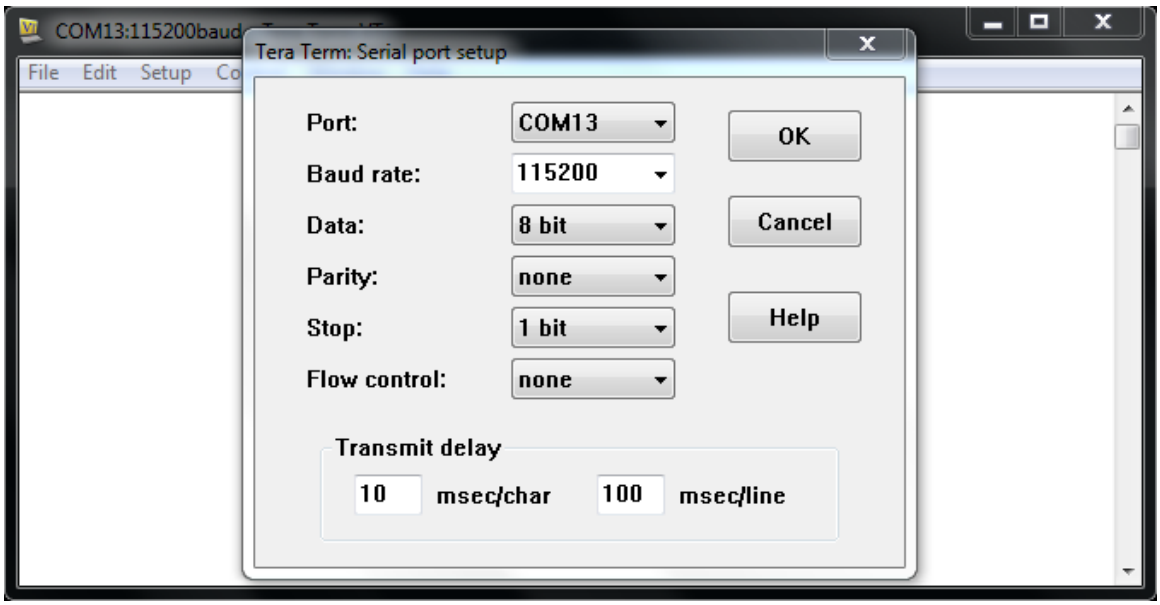


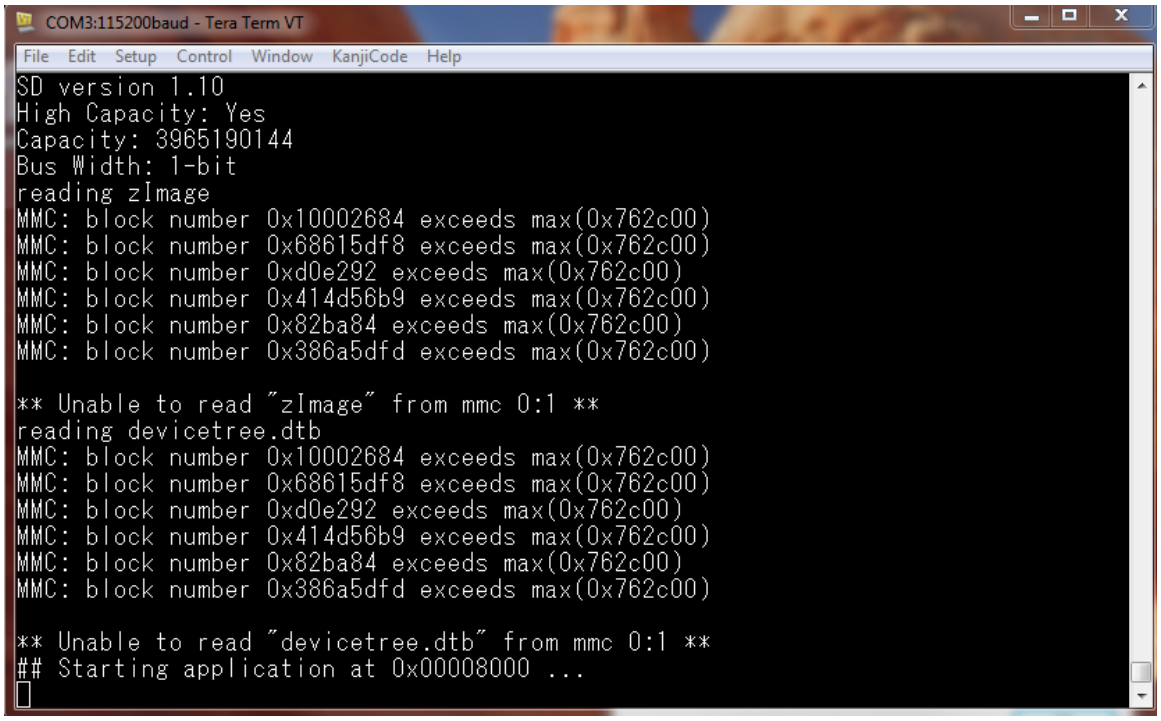
Figure 56 – Tera Term Serial Port Setup Page

11. Optionally, the terminal settings can be saved for later use. To do this, use the **Setup à Save** setup menu selection and overwrite the existing TERATERM.INI file.

12. Power cycle the ZedBoard and monitor the Tera Term window carefully.

    When the terminal output from U-Boot and a countdown is observed, **press any of the keyboard keys to interrupt the boot process**.

    If you fail to interrupt the u-boot countdown, you will see that u-boot dutifully tries to look for a Linux kernel image and its associated device file, and then jumps to the location in memory where the Linux image should have been loaded. We have not created these files as yet, so of course u-boot cannot find them. If you get to this point, simply power cycle the board and be careful to interrupt the countdown when it appears.



    If the amber USB-Link Status (LD11) does not flicker to indicate activity, and no output is displayed on the terminal after 10 seconds, check the driver installation to determine if the device driver is recognized and enumerated successfully and that there are no errors reported by Windows.

Take a few minutes to explore the U-Boot environment and command line options. Use the **?** command entry to display a listing of the supported U-Boot commands. Use any of the listed commands to explore U-Boot's capabilities.

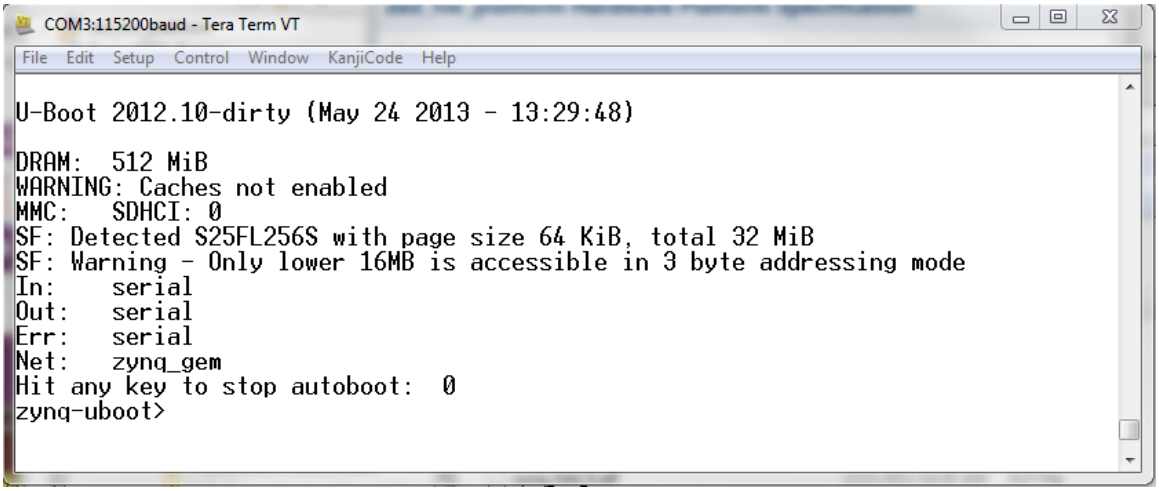


Figure 57 – ZedBoard U-Boot Prompt

Note: If you have run U-boot on the ZedBoard previously, and you saved the environment variables using the **saveenv** command, you will need to update the non-volatile memory at this time so the new **sdboot** command will be used. When we changed the source code, we updated the default environment, but U-boot only uses that when it cannot locate environment variables in non-volatile memory.

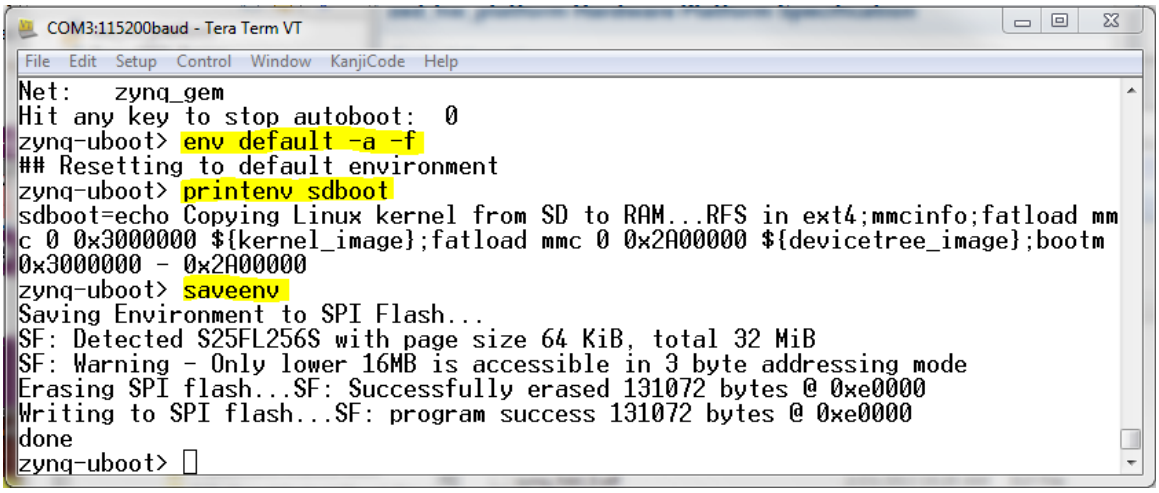
See the next page to perform this action.

At the zynq-uboot prompt, enter:

env default –a –f
printenv sdboot

Check to make sure the sdboot command reflects our source changes. If you are sure it is correct, save the environment by entering:

saveenv



Lab 5 – Configure and Build the Linux Kernel

Lab Overview
ADI provides a freely downloadable Linux kernel solution that has been tested on Xilinx Zynq-7000 Programmable SoC development boards. The source files are hosted on an open source repository site called GitHub.

In this lab, the process to rebuild the kernel from the source repository is explored.

When you have completed this lab, you will know how to do the following:

- Retrieve Linux kernel source code from ADI repository
- Configure the kernel for the ZedBoard target
- Build the kernel for the ZedBoard target

Experiment 1: Clone the ADI Linux Kernel Git Repository

This experiment shows how to make a local copy of the ADI Linux kernel Git repository for Zynq. To successfully complete this lab, you will need Internet access to retrieve the repository information from the GitHub website.

Experiment 1 General Instruction:

Make a local copy of the ADI Linux kernel Git repository in your home directory, and check out the branch we intend to build.

On your Linux host, enter the following commands:

```
$ cd ~

$ git clone \

git://github.com/analogdevicesinc/linux.git ubuntu

$ cd ubuntu

$ git checkout xcomm_zynq
```

Experiment 1 Step-by-Step Instructions:

1. If the CentOS virtual machine is not already open, launch the VMware Player application by selecting **Start à All Programs à VMware à VMware Player**. If the CentOS virtual machine is already open, skip ahead to Step 4.

Figure 58 – The VMware Player Application Icon

2. Select the virtual machine named **Ubuntu-CentOS-6.3-amd64-ZedBoard-linux** from the selections on the left and click on the **Play virtual machine** button to the right.

   If prompted for whether the virtual machine has been copied or moved, click on the **Moved** button.

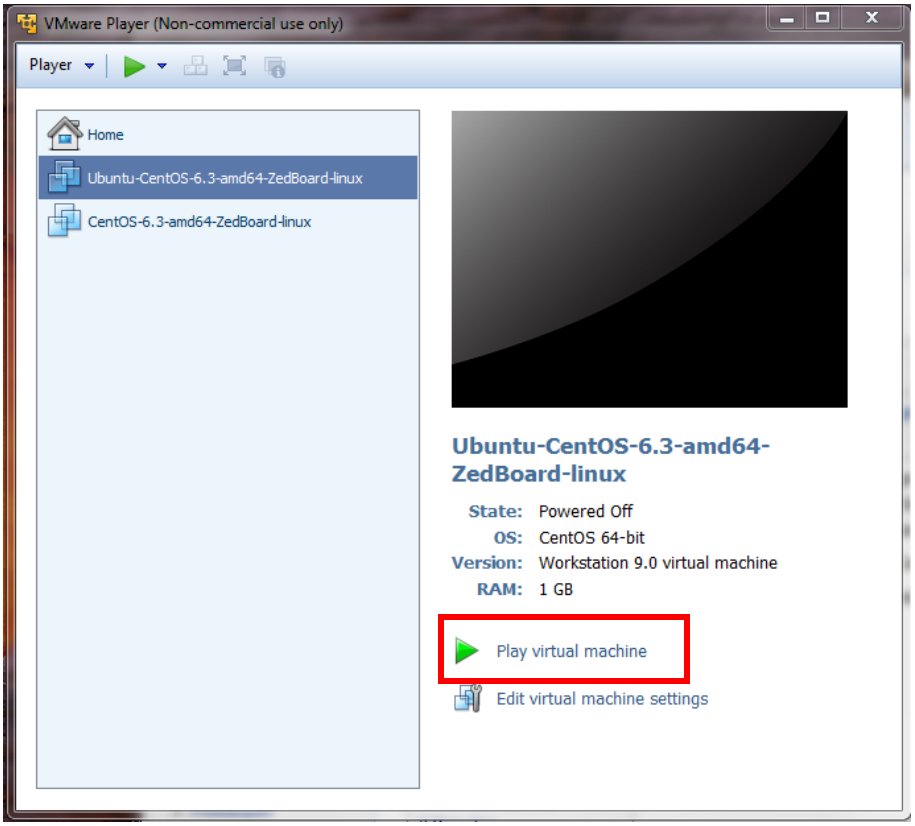


Figure 59 – The VMware Player Application

3. If prompted for a workstation login, click on the user entry **training** and enter the password **Avnet** in order to log into the system.

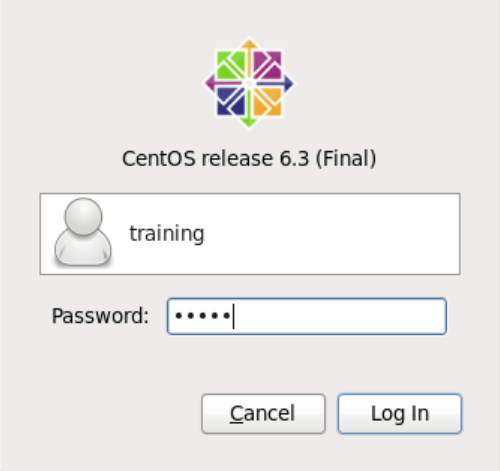


Figure 60 – The CentOS Workstation Login

4. If a terminal is not already open on the desktop of the CentOS guest operating system, open a terminal window through the **Applications à System Tools à Terminal** menu item.

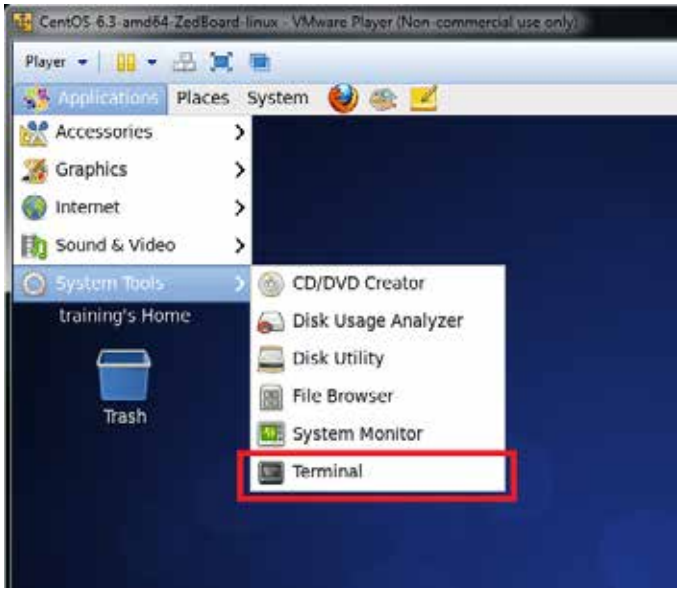


Figure 61 – Launching the CentOS Terminal from the Desktop

5. The Xilinx U-Boot Git repository is located at the following URL:

   git://github.com/analogdevicesinc/linux.git

   To get a working copy of the codebase, clone the remote repository to your local machine. Cloning creates the repository and checks out the latest version, which is referred to as HEAD.

   Use the following Git command to clone the repository. .

```
$ cd ~

$ git clone \

git://github.com/analogdevicesinc/linux.git ubuntu
```

6. Wait until the clone operation completes, this could take 15-30 minutes depending upon your connection speed.

   The clone command sets up a few convenience items for you by:

   - Keeping the address of the original repository
   - Aliasing the address of the original repository as origin so that changes can be easily sent back (if you have authorization) to the remote repository

   This copies the repository to a new directory at **/home/training/ubuntu**.

7. Change into the top level source folder.

```
$ cd ubuntu
```

8. Checkout the code branch compatible with the Xilinx 14.4 tools. This sets up the **xcomm_zynq** branch for remote tracking.

```
$ git checkout xcomm_zynq
```

Experiment 2: Configure and Build the Linux Kernel

This experiment shows how to configure the source branch to target the Xilinx Zynq SoC, and to build an executable file.

Experiment 2 General Instruction:

Clean, configure and build the Linux kernel source for the ARM architecture of the Zynq SoC.

On your Linux host, enter the following commands:

```
$ make ARCH=arm distclean

$ make ARCH=arm zync_xcomm_adv7511_defconfig

$ make ARCH=arm

$ cd arch/arm/boot

$ gzip zImage

$ mkimage –A arm –a 0x8000 –e 0x8000 –n ‘Linux kernel’ \
  -T kernel –d zImage.gz uImage
```

Experiment 2 Step-by-Step Instructions:

1. Change from the home directory into the ADI Linux kernel source directory.

```
$ cd ~/ubuntu/
```

2. For good measure (sometimes a necessity), run a make distribution clean command against the kernel source code. This command will remove all intermediary files created by config as well as any intermediary files created by make and it is a good way to clean up any stale configurations.

```
$ make ARCH=arm distclean
```

3. A Zynq configuration file is included for Zynq targets using the ADI adv7511 HDMI video transmitter on the board. The file is located at:

/arch/arm/configs/zync_xcomm_adv7511_defconfig

This file defines a common configuration for a number of ADI reference designs that may or may not include HDMI video and wireless communication. Our design includes HDMI video, but no wireless comms. Configure the Linux Kernel for the Zynq target by using this default configuration file. Take note that "**zynq"** is misspelled in the filename as "**zync**".

```
$ make ARCH=arm zync_xcomm_adv7511_defconfig
```

4. Build the kernel source with the make command.

   The build process should take about 10 to 20 minutes to complete. If the build is successful and the console output looks similar to that shown in the Figure below, proceed to the next step.

```
$ make ARCH=arm
```

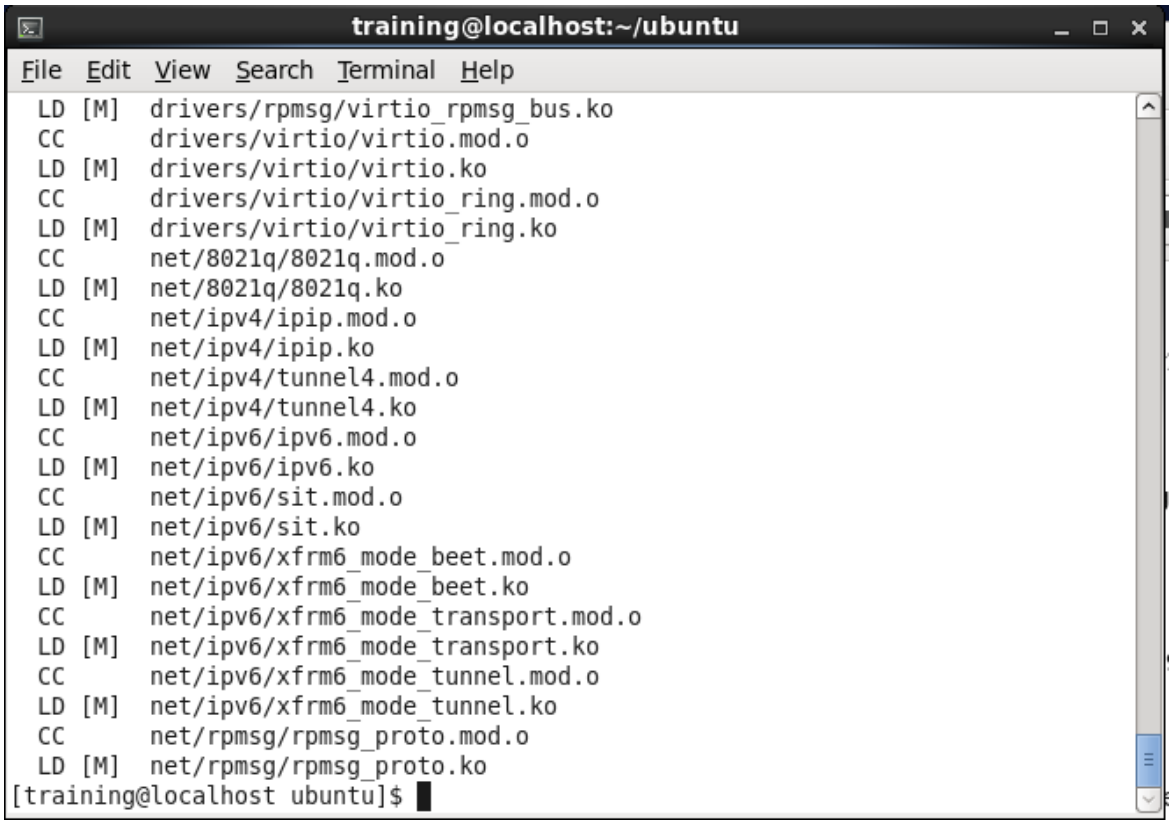


Figure 62 – Linux Kernel Build Completed

4. The final step in the build process is to convert the zImage file to a compress uImage. First, change into the boot directory where the zImage file was created. We need to compress the executable file using the gzip utility, and then use the mkimage utility to create a file that encapsulates boot information that will be read by u-boot at run time.

    a. Change to directory arch/arm/boot

```
$ cd arch/arm/boot
```

    b. Compress the zImage file. This creates a file zImage.gz in place.

```
$ gzip zImage
```

    c. The mkimage utility is included in the u-boot directory structure at:

~/u-boot-xlnx/tools/mkimage

    This directory is probably not in your execution path, so for convenience simply copy the **mkimage** file to a location that is in your execution path, such as **~/bin**.

```
$ cp ~/u-boot-xlnx/tools/mkimage  ~/bin/.
```

    d. Run the mkimage utility to create a uImage file that U-boot will use at run time. The utility has a myriad of options, but the only parameters of interest to us are shown below.

    The mkimage program uses the following parameters:

-A arm	Set the architecture to Arm
-O u-boot	Program type
-T standalone / kernel	Operating system
-C gzip	Compression method
-a 40000	Load address at 0x40000
-e 40004	Execution address at 0x40004
-n 'any text'	Image description
-d <input file>	Input file
<output file>	Output file

Run mkimage using our zImage.gz file as input using as shown to create the uImage file we require for booting.

```
$ mkimage –A arm –a 0x8000 –e 0x8000 –n ‘Linux kernel’ \

-T kernel –d zImage.gz uImage
```

5. If a file browser window is not already open from a previous exercise, open a file browser window through the **Applications à System Tools à File Browser** menu item.

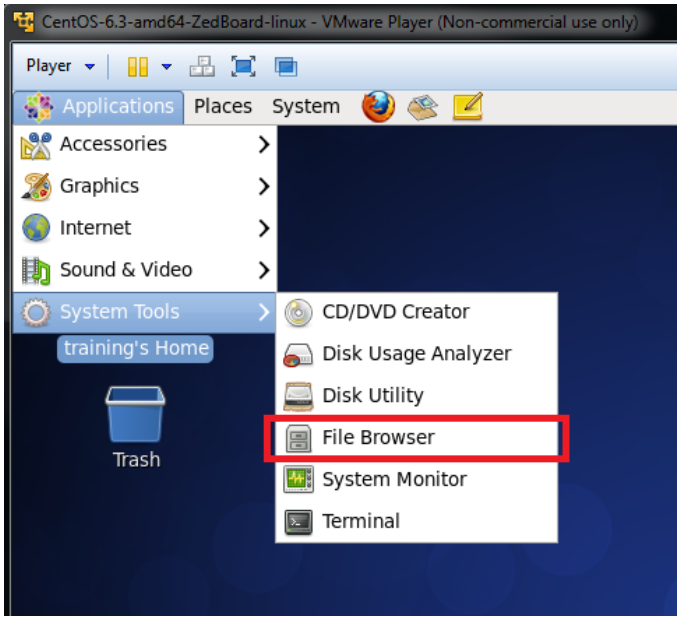


Figure 63 – CentOS File Browser

6. Locate the file **/home/training/ubuntu/arch/arm/boot/uImage,** which is the target executable image needed to execute on Zynq. Right click on this file and select the **Copy** option which will place the selected file in the Virtual Machine clipboard.

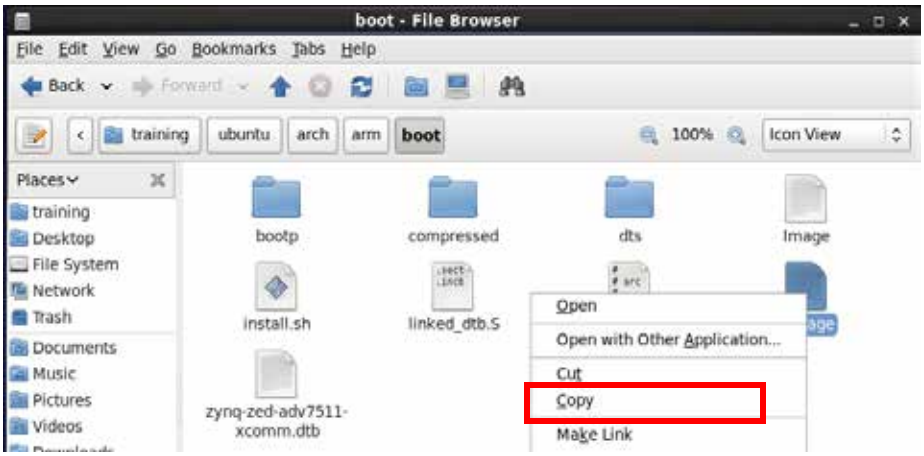


Figure 64 – Copying Kernel uImage to Virtual Machine Clipboard

7. Paste the kernel image into the boot folder under the host operating system by using Windows Explorer to navigate to the following folder:

C:\ZedBoard\Zynq_Ubuntu\bootFiles

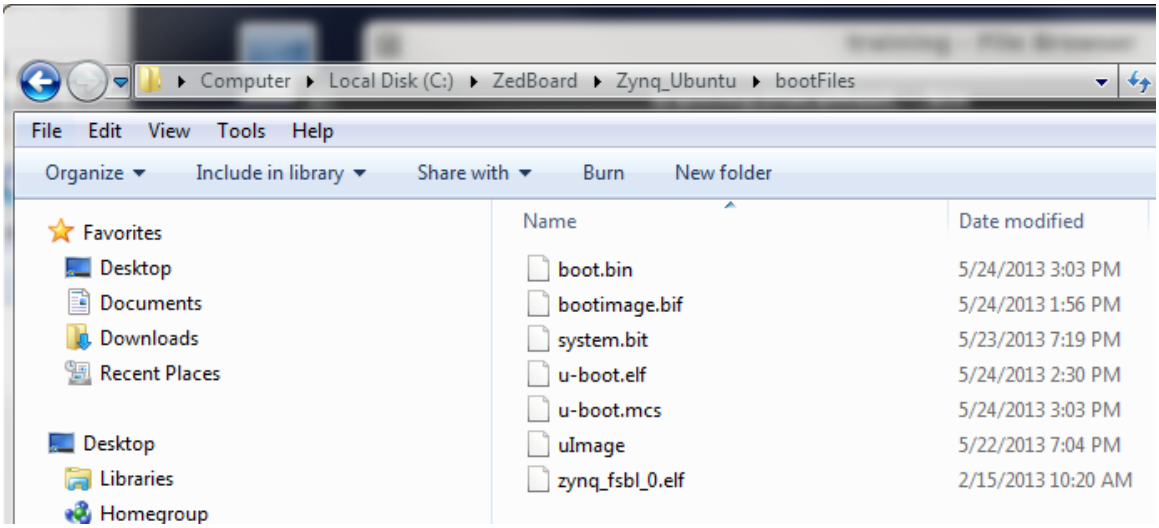


Figure 65 – Kernel uImage Copied to the Host Machine

8. Insert the SD card into the PC or SD card reader and wait for it to enumerate as a Windows drive. If prompted by Windows when inserting the SD card, select the **Continue without scanning** option.

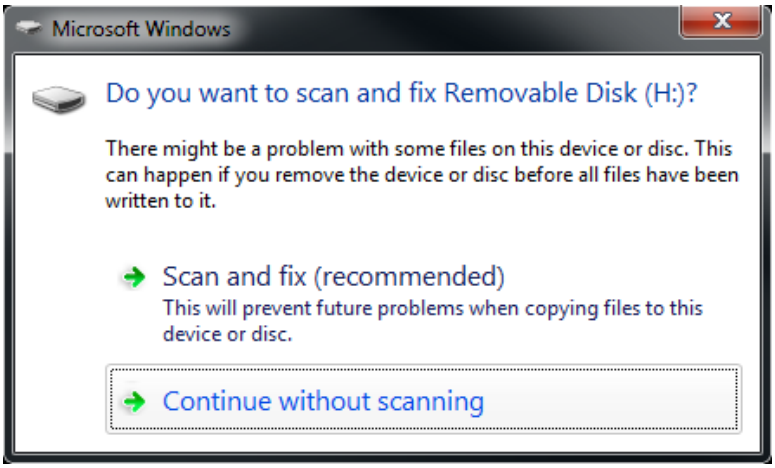


Figure 66 – Windows Prompt for Scanning and Fixing an SD Card

9. Copy the **uImage** file from the **bootfiles** folder to the top level of the SD card. Replace any existing versions of the **uImage** file that may be on the SD card.

Lab 6 – Install the Device Tree

Lab Overview
The Device Tree is a file which contains a data structure describing the hardware system. The information is used by Linux during the kernel boot process to map device parameters such as device type, memory location and interrupt signals.

ADI provides a complete device tree source file for the Ubuntu system in the Git kernel repository at:

~/ubuntu/arch/arm/boot/dts/zynq-zed-adv7511.dts

When you have completed this lab, you will know how to do the following:

- Compile the device tree source file for use by the Linux kernel at boot time

Experiment 1: Compile the Device Tree Source File

This experiment shows how to compile the device tree source file and copy it to your host system for inclusion in the boot directory.

Experiment 1 General Instruction:

Use the makefile in the local Ubuntu repository to compile the device tree source file to its binary equivalent.

On your Linux host, enter the following commands:

```
$ cd ~/ubuntu

$ make ARCH=arm zynq-zed-adv7511.dtb
```

Copy the devicetree.dtb file from the output directory to the boot file directory on the host machine.

Experiment 1 Step-by-Step Instructions:

1. The location of the device tree source is hard-coded into the makefile in the root directory of your local ubuntu repository. You can compile the device tree by providing an output file name in the root directory that matches the name of the source file (the only difference is the .dtb versus .dts endings).

```
$ cd ~/ubuntu

$ make ARCH=arm zynq-zed-adv7511.dtb
```

   The output shows the location of the compiled binary file (.dtb) and the original source file (.dts).

```
  DTC  arch/arm/boot/zynq-zed-adv7511.dtb
DTC: dts->dtb on file arch/arm/boot/dts/zynq-zed-adv7511.dts
```

2. If a file browser window is not already open from a previous exercise, open a file browser window through the **Applications à System Tools à File Browser** menu item.

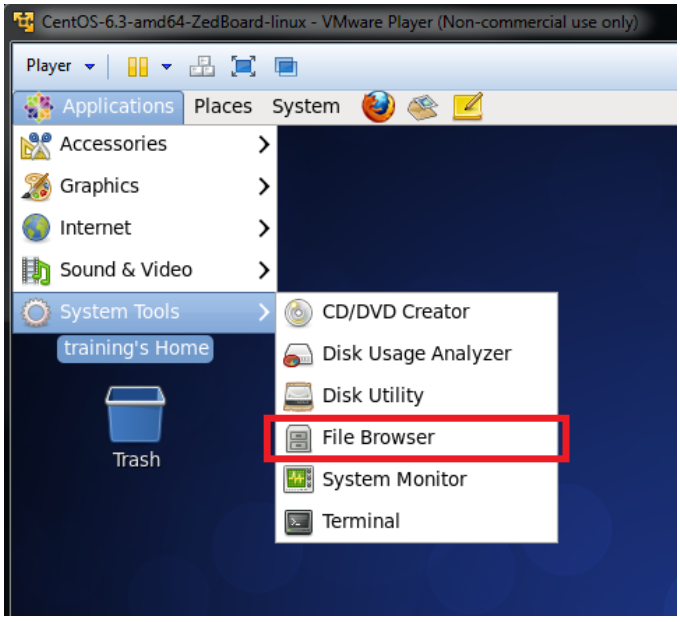


Figure 67 – CentOS File Browser

3. Locate the compiled device tree file at:

~/ubuntu/arch/arm/boot/zynq-zed-adv7511.dtb

Right click on this file and select the **Copy** option which will place the selected file in the Virtual Machine clipboard.

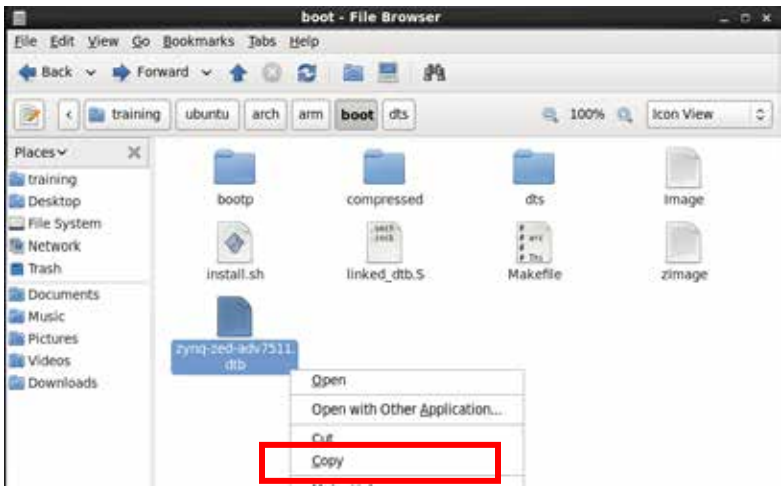


Figure 68 – Copying Kernel Image to Virtual Machine Clipboard

4. Paste the compiled devicetree file into the boot folder under the host operating system by using Windows Explorer to navigate to the following folder:

C:\ZedBoard\Zynq_Ubuntu\bootFiles

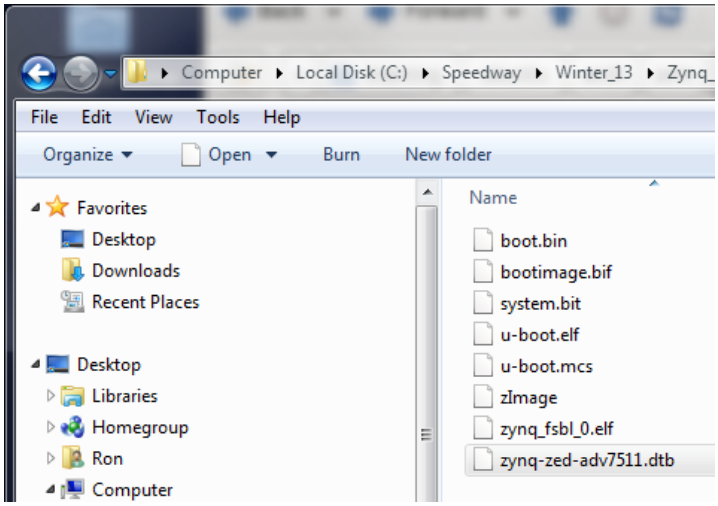


Figure 69 – Compiled Device Tree Copied to the Host Machine

5. Insert the SD card into the PC or SD card reader and wait for it to enumerate as a Windows drive. If prompted by Windows when inserting the SD card, select the **Continue without scanning** option.

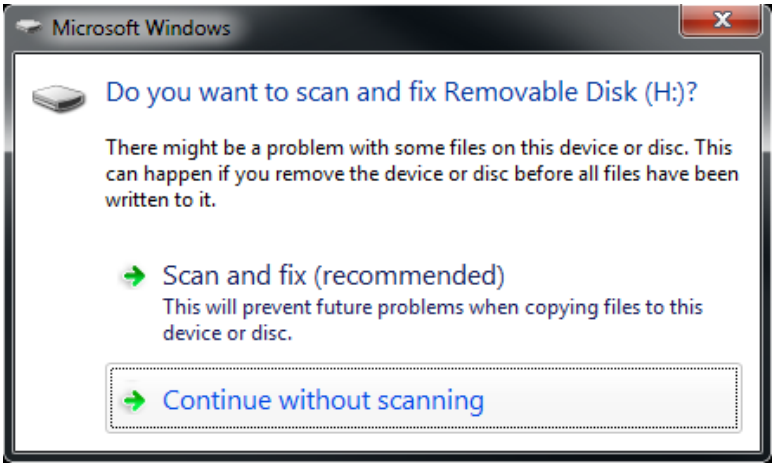


Figure 70 – Windows Prompt for Scanning and Fixing an SD Card

6. Copy the **zynq-zed-adv7511.dtb** file from the **bootfiles** folder to the top level of the SD card. Delete any existing versions of file **devicetree.dtb**, and rename the file you just copied to **devicetree.dtb**.

7. Remove the SD card from your system.

Lab 7 – Obtain and Load the Root File System

Lab Overview

The Root File System is contained on the same partition as the root directory in the Linux kernel, and it is the file system on which all other file systems are mounted. The RFS is separate from the Linux kernel, but is required by the kernel to complete the boot process. All file data used by Linux and any users is contained in the Root File System.

The separation of kernel and file system adds additional flexibility to Linux distributions. Distributions largely use a common Linux kernel (with slight variations to support specific elements important to the distribution), but the large part of what makes distributions unique is determined by the makeup of the root file system. In the case of Ubuntu and Android, for example, both use a common kernel but include feature differences at the root file system level. For Ubuntu, RFS elements provide a desktop-style graphical UI, while Android relies on a Java Virtual Machine to allow for ease of application development. Both systems can co-exist on the same hardware platform because they make use of a common Linux kernel.

The creation of a complete Root File System from scratch is a complex procedure, and beyond the scope of this tutorial. However, if you wish to understand how an RFS is constructed, refer to the Avnet Speedway Implementing Linux on the Zynq-7000 SoC, Lab 2.2, *Creating a Basic Root File System*. The Speedway Lab provides step by step instructions for creating a Root File System from scratch, including creating a complete development environment. While the content of Ubuntu and Android may differ, the process for importing the required features is the same.

In general, it is best to start with a Root File System that contains much of what you already need for your implementation, and then customize it to your specific requirements. Linaro provides a rich root file system that will coexist with our kernel to provide a desktop experience, which serves the purposes of this tutorial.

When you have completed this lab, you will know how to do the following:

- Obtain the Root File System image from the Linaro website
- Install the Root File System to an SD card

You will need Internet access from your Linux host to retrieve the RFS.

Experiment 1: Download and Install the Root File System

Experiment 1 General Instruction:

Download the RFS image from the Linaro website, and install it on a pre-formatted SD card that will be used to boot Ubuntu on the ZedBoard.

On your Linux host, enter the following commands:

```
$ cd ~

$ wget http://releases.linaro.org/11.12/ubuntu/oneiric-images/ubuntu-desktop/linaro-o-ubuntu-desktop-tar-20111219-0.tar.gz

$ sudo tar --strip-components=3 –C /media/rootfs -xzpf \
  linaro-o-ubuntu-desktop-tar-20111219-0.tar.gz \
  binary/boot/filesystem.dir
```

Experiment 2 Step-by-Step Instructions:

1. Download a prebuilt root file system image from Linaro to your local user top-level directory.

```
$ cd ~

$ wget
http://releases.linaro.org/11.12/ubuntu/oneiric-images/ubuntu-desktop/linaro-o-ubuntu-desktop-tar-20111219-0.tar.gz
```

2. With the VM running, insert the pre-formatted SD card created in Lab 3 – Partition the SD Card for Booting into a compatible slot on your host machine. If the USB device is registered by Windows, close any pop-ups that appear. From the VM, you may see a window to indicate that a new removable device is available. Click OK to acknowledge and close the window.

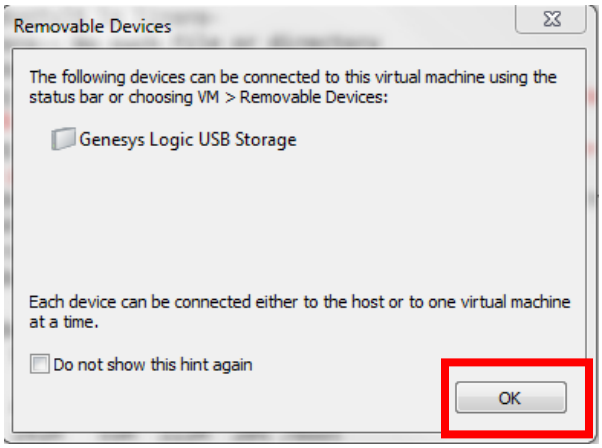


Figure 71 – USB Device Detection by Virtual Machine

3. In the VM, look at the upper right-hand corner and locate the icon that applies to the USB device. Right-click on the icon and select "Connect" from the drop-down menu.

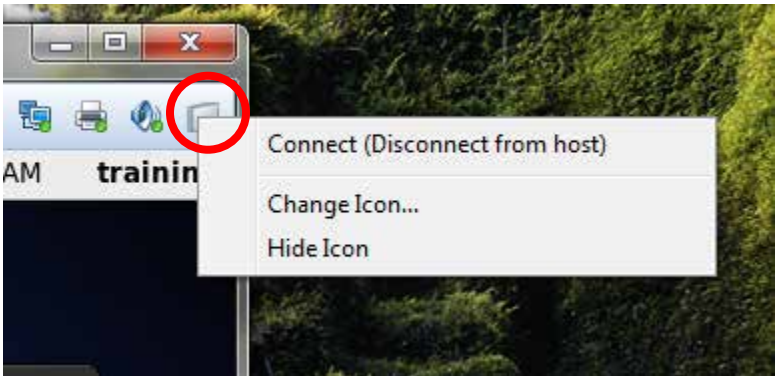


Figure 72 – Connect USB Device to Virtual Machine

4. Click OK to acknowledge the device will be disconnected from the Windows host and connected to the Virtual Machine.

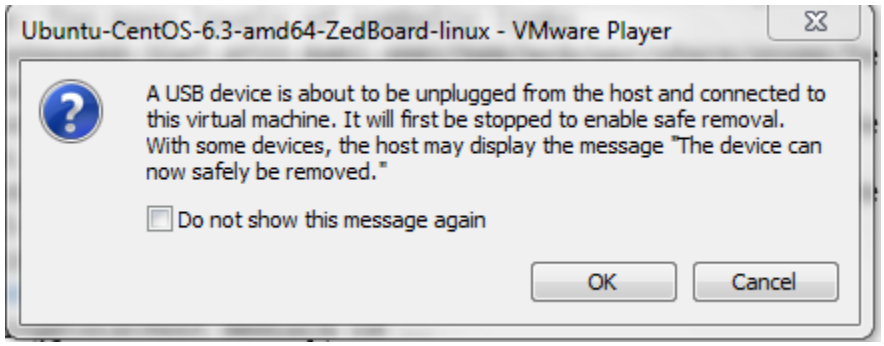


Figure 73 – Accept USB Device Connection to VM

5. With the device connected to the VM, you may see browser windows activate to view files on a non-blank SD card. Close any browser windows associated with the SD card.

6. In the Linux Terminal Window, extract the root file system onto the SD card. This process will take 5 to 15 minutes to complete.

```
$ sudo tar --strip-components=3 –C /media/rootfs -xzpf
linaro-o-ubuntu-desktop-tar-20111219-0.tar.gz
```

7. Disconnect the SD Card from the Linux system.

Lab 8 – Boot Ubuntu

Lab Overview

If you have successfully completed all the preceding labs, you will have an SD card configured with 2 partitions. The first partition is in FAT32 format, accessible from either a Windows or Linux host machine, and it will contain all of the files required to bring up the board and start the Linux kernel booting. The second partition is in ext4 format, invisible to Windows but used by Linux systems, and it holds the root file system created by Linaro.

This root file system contains the Ubuntu desktop environment. This higher level system uses a common Linux kernel, and can therefore coexist on the same processor system with other higher level functions, such as an Android Java VM. In fact, this is the basis for much of the current development in the mobile marketplace; the initiative to replace all existing computer systems with mobile devices has begun. If a cellphone can have a sufficiently powerful processor system, there is no reason it cannot act as a computer in its own right. An Android OS provides the app-rich environment currently familiar to high-end smartphone owners, while the Ubuntu desktop allows access to applications normally seen only on PCs or tablets. This merging of the mainstream computing with the mobile world is the next step forward in portable computing.

When you have completed Lab 8, you will know how to do the following:

- Configure the ZedBoard to boot from an SD card
- Connect all the external devices necessary to allow the ZedBoard to function as a desktop computer. This is equivalent to placing your smartphone in a dock.
- Boot the Ubuntu desktop to provide a graphical desktop environment
- Make post-installation changes to the environment to add an updated video driver, and to correct an issue with the audio

Experiment 1: Boot the ZedBoard to the Ubuntu Desktop

This experiment describes the device connections required for a desktop boot. Board setup should be the same as in Lab 4 - Create the First Stage Boot Loader. You will need the following additional equipment for this lab:

- HDMI or DVI capable monitor capable of 1600 X 1200 resolution
- USB 2.0 Powered Hub
- USB Keyboard
- USB Mouse
- HDMI cable, with HDMI to DVI adapter if a DVI monitor is used

Experiment 1 General Instruction:

Configure the ZedBoard for booting from the SD card. Connect the external peripherals to the ZedBoard required to use the system as a desktop computer. At a minimum you need a keyboard, a mouse, an HDMI monitor and a serial connection for a console. Apply power to the board to boot the desktop.

Experiment 1 Step-by-Step Instructions:

1. Configure the ZedBoard for booting from the SD card, as described in Lab 4 - Create the First Stage Boot Loader.

2. Connect the HDMI/DVI monitor to the HDMI output on the ZedBoard using the HDMI cable, and adapter if required.

3. Connect the USB Hub to the USB-OTG port on the ZedBoard using the Micro-USB to USB Adapter cable supplied with the board.

4. Connect the USB keyboard to an open port on the USB Hub.

5. Connect the USB mouse to an open port on the USB Hub.

6. If you have not done so as part of Step 1, connect the USB serial port to your host system for use with Tera Term as the boot console. Tera Term will not be able to detect the Comm port until the board is powered and output has been sent, so wait to start Tera Term until you see the blue configuration LED illuminate after powering the ZedBoard.

7. Apply power to the ZedBoard, and in about 15 seconds the blue configuration LED will illuminate. With Tera Term operating, you will be able to see U-boot load the images from the SD card and transfer control to the Linux kernel. The remainder of the console output comes from the kernel boot process, which will end with a command prompt.

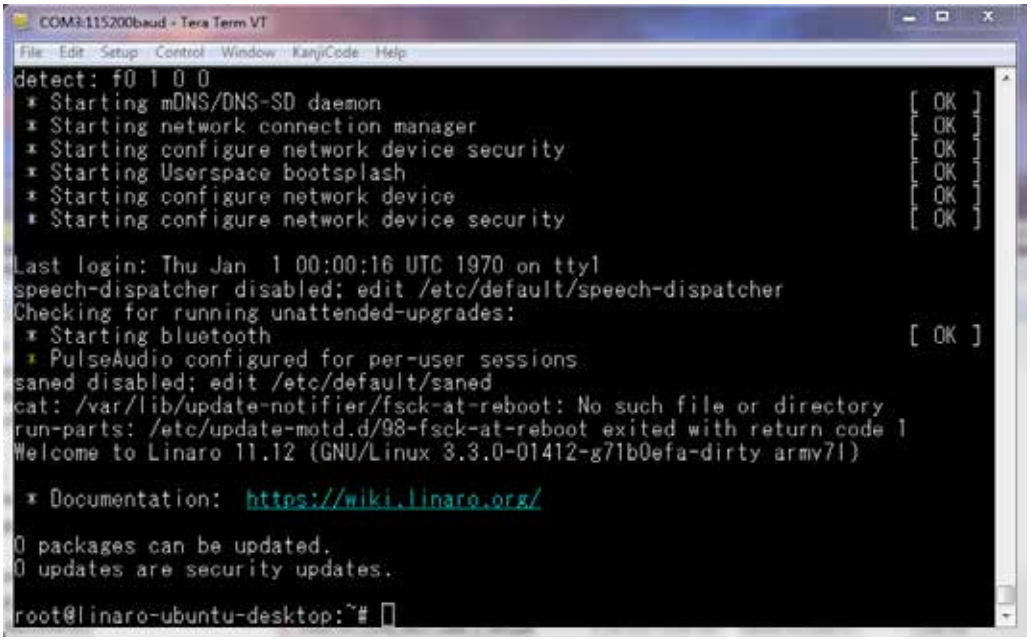


Figure 74 – Linaro Console Boot

8. Once the command prompt is displayed, you will see the Ubuntu desktop display on the HDMI monitor. You can interact with the desktop via the USB mouse and keyboard attached to the USB Hub. Note that the screen resolution may not be optimal – this is addressed in the next section.

Figure 75 – Ubuntu Default Desktop

9. There are a couple of deficiencies in this installation, as documented on the ADI wiki site:

   http://wiki.analog.com/resources/tools-software/linux-drivers/platforms/zynq

   You may check this URL for the latest information, but at the time of writing the following information was provided for video and audio modifications:

Post-installation tweaks

After the system has been installed it is time to do some post-installation tweaks to the system. None of them is required to get a basic working system, but they improve the overall video and audio experience quite a bit.

Enable xf86-video-modesetting Xorg driver

In the default installation, the Ubuntu desktop may not be able to take advantage of the full resolution of your monitor. The xf86-video-modesetting driver has been written to take advantage of the new Kernel Mode Setting (KMS) API of the DRM layer. This allows a user to switch between different screen resolutions at runtime (using the Xservers xrandr interface) and adds plug-and-play support for monitors.

Unfortunately the current Linaro Ubuntu distribution does not contain a package for xf86-video-modesetting driver. So it becomes necessary to manually download and build it. Open up a terminal on the target system (use the Dashboard and search for "terminal" to locate a suitable program) and run the following commands:

Download and install xf86-video-modesetting

```
> sudo apt-get install xserver-xorg-dev libdrm-dev xutils-dev
> wget http://xorg.freedesktop.org/archive/individual/driver/xf86-video-modesetting-0.5.0.tar.bz2
> tar -xjf xf86-video-modesetting-0.5.0.tar.bz2
> cd xf86-video-modesetting-0.5.0
> ./configure --prefix=/usr
> make
> sudo make install
```

To enable the modesetting driver, you will need root access to use the vi editor to create **/etc/X11/xorg.conf** and add following lines. The easiest way to accomplish this is to use the Tera Term console, which is already logged in as root.

```
Section "Device"
  Identifier "ADV7511 HDMI"
  Driver "modesetting"
EndSection
```

Once this file is created, the driver changes will take effect on the next boot. You will then be able to click on the System Operations icon in the upper right of the display:

Figure 76 – Click on the System Operations Icon

Select **Display** from the drop-down menu to access resolution settings for the monitor. In the example below, an HP LP2065 20" monitor was used:

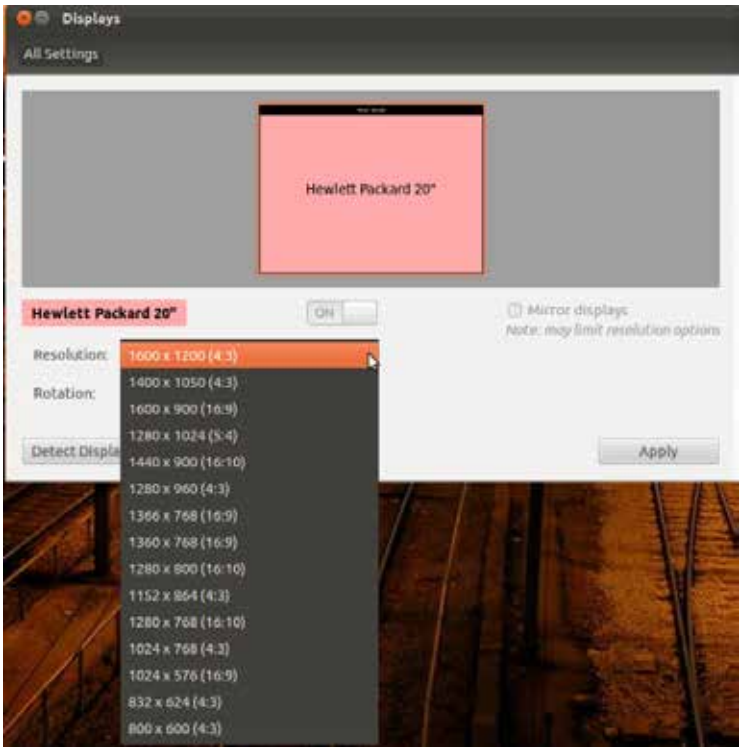


Figure 77 – Display Settings after Driver Update

The root file system exists on the SD card, so changes made will persist across subsequent boots.

Fixing issues with Pulse Audio

PulseAudio is the audio daemon used by default on the Linaro Ubuntu installation. Unfortunately PulseAudio's 'glitch-free' algorithm seems to cause audio glitches on this particular platform. To get seamless audio experience it is necessary to disable the glitch-free feature. To disable the 'glitch-free' feature of pulse audio open up a terminal on the target system and run the following command (all on one line):

> **sudo sed -i 's,load-module module-udev-detect,load-module module-udev-detect tsched=0,' /etc/pulse/default.pa**

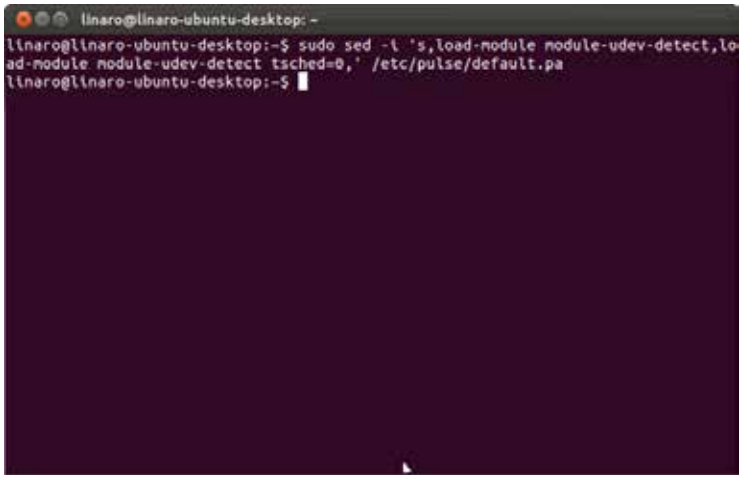


Figure 78 – Update Pulse Audio in Terminal Window

10. This is now a desktop computer environment, so you should follow a standard shutdown procedure before removing power from the ZedBoard. To shut down the graphics system, select **Shutdown** from the drop down menu off the System (gear) icon at the upper right in the Ubuntu desktop.

11. To stop the command line console and shut down Linux, enter **poweroff** at the command prompt in the console window (Tera Term). You can now switch off power to the ZedBoard.

Lab 9 – Ubuntu Demo

Lab Overview
The Ubuntu desktop is one of the most popular user interfaces for Linux PCs. In the lab you will become familiar with the basics of the Ubuntu GUI.

When you have completed Lab 9, you will know how to do the following:

- Navigate the Ubuntu desktop
- Locate the fundamental features of Ubuntu

Experiment 1: The Basic Desktop

This experiment introduces the Ubuntu desktop, which uses similar concepts and constructs common to most mouse-based GUIs, including the traditional Windows desktop.

Experiment 1 General Instruction:

Explore the Ubuntu desktop.

Experiment 1 Step-by-Step Instructions:

1. If you have not already done so, boot your Ubuntu desktop on the ZedBoard using the SD card files created in the previous labs. You should have the following hardware connected to your system to make full use of the desktop:
    a. HDMI Monitor (or DVI Monitor with HDMI-to-DVI adapter)
        i. (Optional) HDMI Monitor with Speakers
    b. USB Hub
    c. USB Mouse
    d. USB Keyboard
    e. (Optional) Ethernet cable to connect to a DHCP router (or similar) for Internet access via Firefox

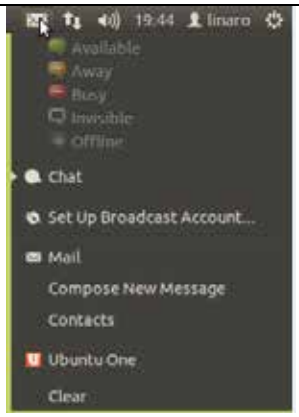
Once booted, your basic unmodified desktop should appear as shown in the figure below. You are automatically logged in as the default user, with password "***linaro***". You will need to enter this password if you leave your desktop unattended, as by default it will automatically lock after a few minutes.

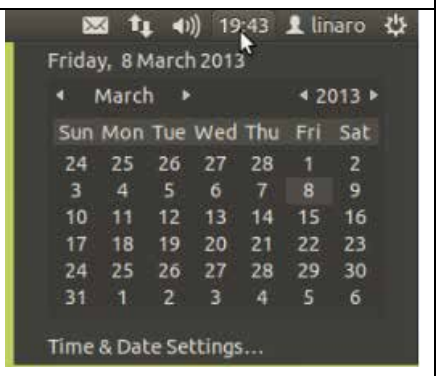
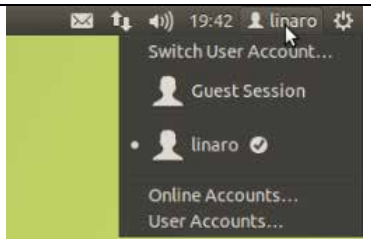
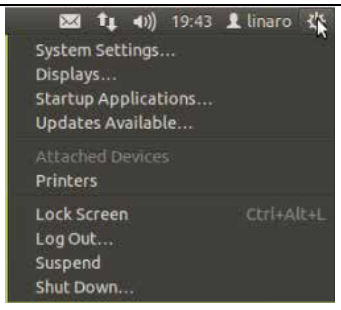
Figure 79 – Ubuntu Default Desktop

There are two control areas visible on the desktop, one at the upper right corner and a series of icons down the left hand side.

2. The control icons at the upper left are quick links to basic OS functions that should be familiar to everyone.

Viewed from left to right, clicking on each icon in turn will produce a drop-down menu to access functions for:

Email and Web services		

Networking		
Sound Controls		
System clock and calendar functions		
User Account information		
System operations, including Log Off and Shut Down		

3. The primary area to access desktop applications is the **launchpad** at the left of the display. There is a set of default icons, plus each application you launch will appear in this area while it is running. You may right click on an application icon to either lock it or remove from the launchpad.

   The applications represented by each icon are shown below:

	**Dashboard** – quick access to any applications not present in the launchpad
	**File Browser** – Equivalent to Windows Explorer
	**Firefox** – the default web browser
	**Ubuntu Software Center** – manage application installation for your Ubuntu system.
	**Ubuntu One** – access to the online Ubuntu community
	**System Settings** – Equivalent to Windows Control Panel
	**Snapshot** – Application used to take the screenshots for this manual. This is not in the launchpad by default, but was locked in place as described above.
	**Workspaces** – divides the screen into 4 separate work areas that can be individually populated.

4. Dashboard

The Ubuntu Dashboard is your central location for locating and running applications. You can select one of the applications groups on the top level to display installed software by group, but it is generally easier to locate applications by simply typing a name into the search box at the top.

Figure 80 – Ubuntu Dashboard

For example, if you wish to locate a screen capture program, you might begin by typing "screen" into the search area. As you type, matching application names are displayed, so that after a few characters, we find what we are looking for.

Figure 81 – Locate Applications with Dashboard

5. Ubuntu Software Center

   From this panel you can view and manage the software application portfolio on your Ubuntu system. You can view installed software, see the installation history, and locate new applications on the Internet which may be downloaded.

   Note that not all applications will install on the Ubuntu system running on the ZedBoard, due to the restrictions imposed on the OS by the embedded system.

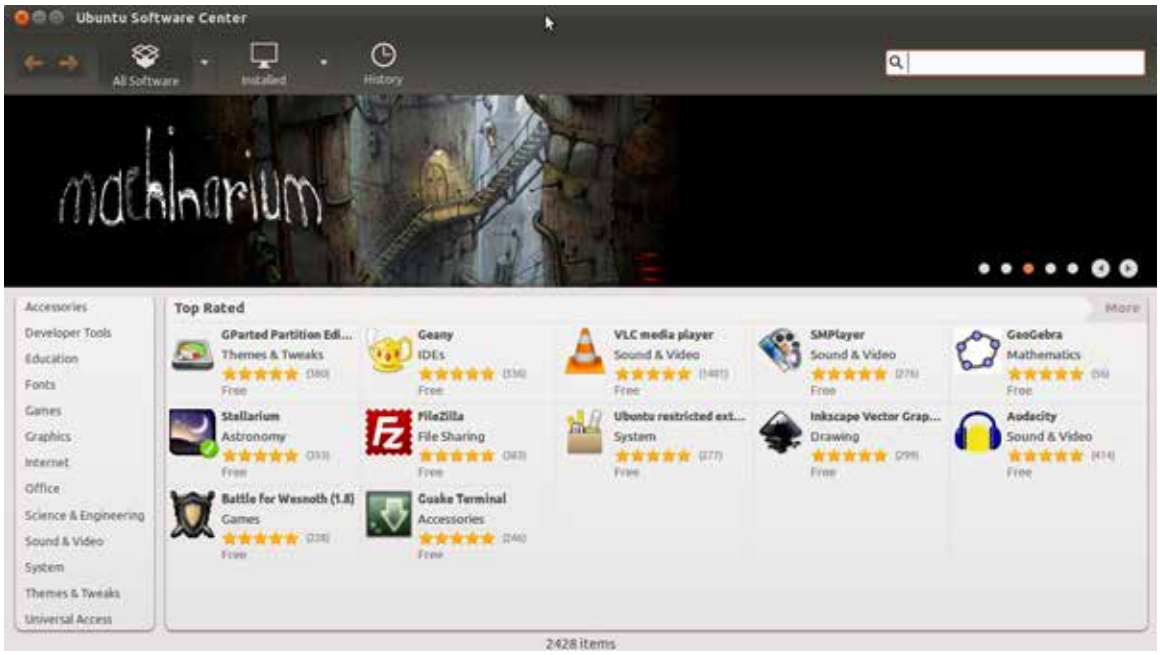


Figure 82 – Ubuntu Software Center

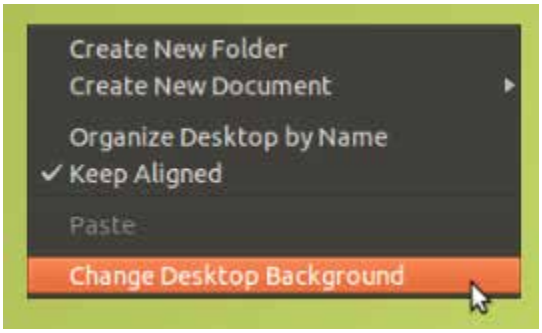
6. Ubuntu One

   Ubuntu is the top Linux choice for desktop environments, and as such there is an active online community. If you are interested in exploring Ubuntu further, you can join and participate online by clicking the **Learn More** or **Join Now** buttons. You will need Internet access to participate.

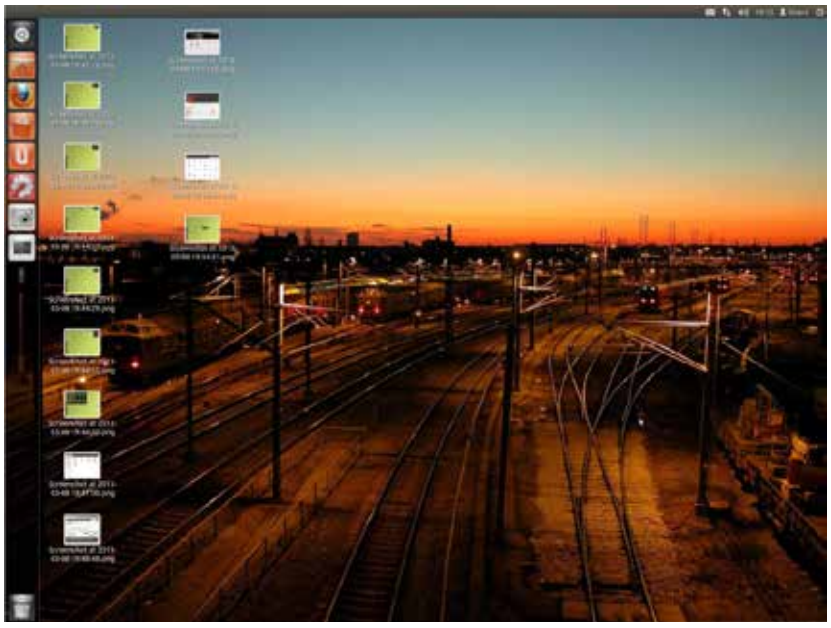
Figure 83 – Ubuntu One

7. Customization

Of course, standard personalization familiar to desktop users is available in Ubuntu. For a simple example, right-click on the desktop to produce the pop-up menu below:



Select **Change Desktop Background** and you can pick from a variety of themes, as well as adding your own. An example of a customized desktop background for Ubuntu is shown below.



8. Games

In the default installation, there are a number of well-known games, plus you can use the Ubuntu Software Center to locate and download many more from the Internet. You can use the Dashboard filter on the right to see only the application types you are looking for – in this example, Games.

Figure 84 – Ubuntu Games

Experiment 2: Sound

If you are using an HDMI monitor incorporating a speaker system, then you will be able to hear sound in the default configuration. However, if you are using a DVI monitor, you will want to plug external speakers or headphones into the Line Out receptacle on the ZedBoard. Unfortunately, in the standard configuration, there is no driver in Unbuntu for the Analog Devices ADAU1761 audio codec, so we must modify the default setup to drive the line out signal.

Experiment 2 General Instruction:

Install the driver for the Analog Devices ADAU1761 audio codec and test the setup using the Ubuntu Sound application via external speakers.

At the time of writing, the audio codec driver is a work in progress, and has not been fully integrated with the mainstream repository. The simplest way to get it to work is to check out the **audio_zynq** branch, which includes the driver modifications, and build a new zImage and devicetree. Instructions for this are given in the Step-by-Step listing below.

Experiment 1 Step-by-Step Instructions:

1. If you wish to create a fresh repository, you may do so on your CentOS VM by optionally cloning the ADI repository to a new directory (as was done for the original kernel build). If you wish to use the same directory structure you have been using, you can skip to step 2 and check out the remote branch for the audio codec. Instructions in this section assume you have created a new repository. To clone the repository, enter the following commands in a terminal window:

```
$ cd ~

$ git clone \

git://github.com/analogdevicesinc/linux.git audio-linux

$ cd audio-linux
```

2. The branch in the repository containing the driver code for the Adau1761 is called **audio_zynq**, so we must check out the remote repository to obtain the most recent source changes.

```
$ git checkout –b remotes/origin/audio_zynq
```

3. Configure and build the updated kernel.

```
$ make ARCH=arm distclean

$ make ARCH=arm zync_xcomm_adv7511_defconfig

$ make ARCH=arm
```

4. Compile the associated device tree.

```
$ make ARCH=arm zynq-zed-adv7511.dtb
```

5. Insert the SD card into your host system, and using copy and paste from the CentOS VM, copy the zImage and zynq-zed-adv7511.dtb files. Note that you may want to preserve the original zImage and devicetree.dtb files, so make sure they are backed up before you do the copy. Use Windows Explorer to paste the two files to the FAT32 partition of your SD card. Location of the files on the CentOS VM from your top level repository directory is:

   arch/arm/boot/zImage

   arch/arm/boot/zynq-zed-adv7511.dtb

6. Rename the compiled device tree file on the SD card to **devicetree.dtb**.

7. Once Ubuntu is running on the ZedBoard, you will need to configure the audio driver to produce sound through the adau1761 codec. There are a number of configuration settings to be completed, so the simplest way to do this is via a configuration file, *zed_audio.state*. At the time of this writing, this file is still in development status. It is discussed in the thread below and can be downloaded from the second link. Download and decompress the file.

   http://ez.analog.com/thread/17243
   http://ez.analog.com/servlet/JiveServlet/download/80077-14002/zed_audio.state.zip

   We need this file to be available in the root file system for Ubuntu on the ZedBoard, after it boots. You can do this two ways: prior to booting the ZedBoard (see a. below) or after the ZedBoard is booted to the Ubuntu desktop (see b. below):

   a. With the SD card still attached to your host system, use the CentOS VM to mount the ext4 partition on the SD card and copy the zed_audio.state file to the **Desktop** directory.

b. Boot the ZedBoard to the Ubuntu desktop, and place the zed_audio.state file in a location that is accessible to the network the ZedBoard is attached to. You can use the File Browser application in the Launch area to copy the file from the network location to the **Desktop** directory.

8. Insert the SD card into your ZedBoard. Your system should be set up for an Ubuntu boot, with connections for the monitor, mouse, keyboard and Ethernet, just as it was when booting the mainline Ubuntu in previous labs. In addition, you should connect a set of powered speakers or a set of headphones to the line out jack on your ZedBoard.

9. Apply power to the ZedBoard, and once the blue light indicating that the FPGA has been configured illuminates, optionally start Tera Term on your Windows host to receive output as the boot console. The system will boot to the Ubuntu desktop, as usual.

   Note: At the time of writing, the audio branch is still under development. The kernel may throw numerous error messages during boot, and continue to report errors on the Tera Term console after Ubuntu is operational. These errors will be corrected by open source developers in due course.

10. Open a terminal window on the Ubuntu Desktop. Change to the **Desktop** directory and configure the audio driver with the zed_audio.state file you copied there earlier (using one of the two methods described above). Enter the following commands:

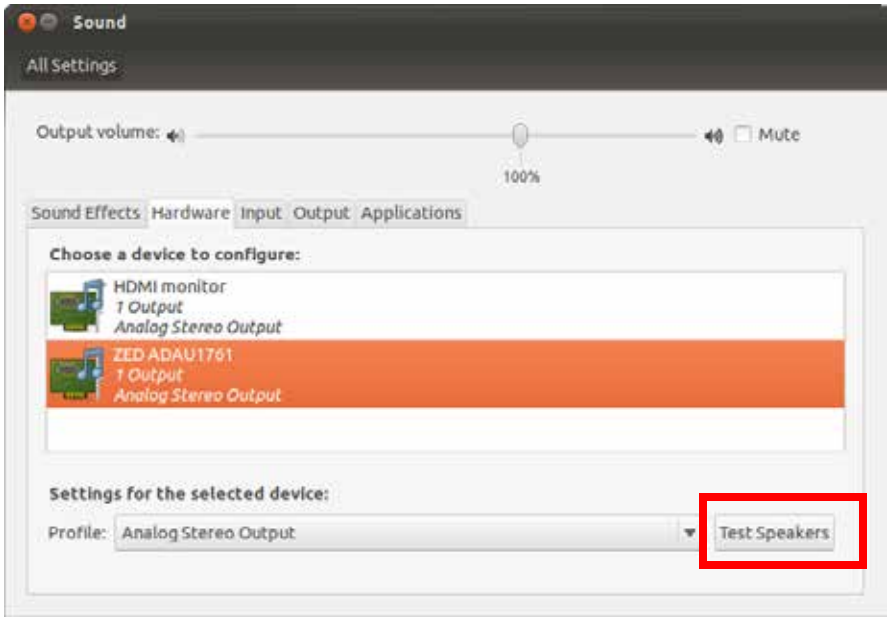
```
$ cd Desktop

$ alsactl restore –f zed_audio.state
```

If you receive an error, reboot the ZedBoard and try again.

11. In the Ubuntu desktop, click on the sound icon in the upper right of the display

Select **Sound Settings** from the drop-down menu.

12. In the **Sound** panel, select the **Hardware** tab.



You will see two sound devices – select ZED ADAU1761 and click the **Test Speakers** button.

13. Click on the Test buttons for the Front Left and Front Right speakers. You may need to turn up the audio settings to maximum in both Ubuntu and externally on your powered speakers or headphones (if an external audio gain dial is available). You will hear "Front Left" or "Front Right" from the corresponding speaker in response to the button press.

This completes activation of the Adau1761 audio codec. You may close all open windows.

Appendix I - Installing RHEL EPEL Repo on Centos 6.x

http://www.rackspace.com/knowledge_center/article/installing-rhel-epel-repo-on-centos-5x-or-6x

- Article ID: **1272**
- Last updated on **February 13, 2013**
- Authored by: **Rackspace Support**
- (25 Comments)

How to install RHEL EPEL repository on Centos 6.x

The following article will describe how to configure a CentOS 5.x-based or Centos 6.x-based system to use Fedora Epel repos and third party **remi** package repos. These package repositories are not officially supported by CentOS, but they provide much more current versions of popular applications like PHP or MYSQL.

Install the extra repositories

The first step requires downloading some RPM files that contain the additional YUM repository definitions. The instructions below point to the 64-bit versions that work with our Cloud Server instances.

```
$ wget
http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-
release-6-8.noarch.rpm

$ wget
http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-
release-6-8.noarch

$ sudo rpm -Uvh remi-release-6*.rpm epel-release-6*.rpm
```

Once installed you should see some additional repo definitions under the */etc/yum.repos.d* directory.

```
$ ls -1 /etc/yum.repos.d/epel* /etc/yum.repos.d/remi.repo
```

/etc/yum.repos.d/epel.repo

/etc/yum.repos.d/epel-testing.repo

Enable the remi repository

The remi repository provides a variety of up-to-date packages that are useful or are a requirement for many popular web-based services. That means it generally is not a bad idea to enable the remi repositories by default.

First, open the */etc/yum.repos.d/remi.repo* repository file using a text editor of your choice:

```
$ sudo vim /etc/yum.repos.d/remi.repo
```

Edit the **[remi]** portion of the file so that the *enabled* option is set to *1*. This will enable the remi repository.

```
$ sudo vim /etc/yum.repos.d/remi.repo
```

```
name=Les RPM de remi pour Enterprise Linux $releasever - $basearch

#baseurl=http://rpms.famillecollet.com/enterprise/$releasever/remi/$basearch/

mirrorlist=http://rpms.famillecollet.com/enterprise/$releasever/remi/mirror

enabled=1

gpgcheck=1

gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-remi

failovermethod=priority
```

Figure 85 – Tera Term Icon

You will now have a larger array of yum repositories from which to install.

Revision History

Date	Version	Revision
16 Apr 13	01	Initial Draft
24 May 13	02	Updated for kernel uImage & compatible U-boot

Resources

http://www.zedboard.org

http://www.xilinx.com/zynq

http://wiki.analog.com/resources/tools-software/linux-drivers/platforms/zynq

http://www.xilinx.com/planahead

http://git-scm.com/

http://www.denx.de/wiki/U-Boot

http://www.linaro.org/

http://www.ubuntu.com/