Ubuntu on

Zynq®-7000 All Programmable SoC Tutorial

For
ZedBoard and
Zynq Mini-ITX Development Kits

Version 3
July 2014

Table of Contents

Tutorial Prerequisites

This software tutorial relies on a specific development environment. All the necessary files and documentation for creating this can be found at:

http://www.zedboard.org/

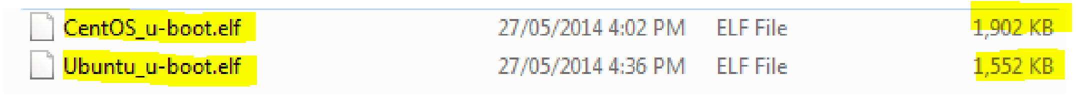
The following requirements must be met before the tutorial instructions can be completed:

1. Instructions for software builds under Linux are documented using a CentOS development environment and Code Sourcery toolchain, encapsulated inside a Virtual Machine running on a Windows 7 host. For instructions to obtain and install the free VMware player, see Appendix I - Installing the VMware Player. For instructions on installing the OS and setting up the development environment, see **Appendix II - Installing CentOS as the VMware OS**.

2. If you prefer to use an Ubuntu development environment and the Linaro toolchain, instructions for installing the OS and setting up the environment are found in

3. **Appendix III - Installing Ubuntu as the VMWare** OS. You will notice some GUI differences in the Ubuntu Unity interface when compared to the CentOS instruction Figures, so experience with the Ubuntu desktop may be helpful. All command line instructions are identical in both environments.

   The Linaro toolchain produces ARM executable files that are more compact than the corresponding versions created by the Code Sourcery toolchain. For example, the Linaro U-boot executable is approximately 20% smaller.



```
CentOS_u-boot.elf     27/05/2014 4:02 PM    ELF File    1,902 KB
Ubuntu_u-boot.elf     27/05/2014 4:36 PM    ELF File    1,552 KB
```

Figure 1 – Toolchain Executable Size Comparison

4. The Linux kernel and Ubuntu desktop require a hardware platform targeted to the development board you are using. To obtain the hardware platform created with Vivado® Design Suite, please see the following Avnet reference materials:

   ZedBoard HDMI Bare Metal Reference Design Using ADV7511 and ADI IP

   or

   Mini-ITX HDMI Bare Metal Reference Design Using ADV7511 and ADI IP

   To locate the reference designs manually, log into the *www.Zedboard.org* site, select the *Support* tab and click the *View* button for the appropriate target.

It is 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 Speedways **Developing Zynq® - 7000 AP SoC Software** and **Developing Zynq - 7000 AP SoC Hardware**.

To locate the Speedways manually, log into the *www.Zedboard.org* site, select the *Support* tab, and click on the *Trainings and Videos* link.

Lab Setup for Xilinx Tools

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

Software

- Xilinx® SDK 2013.4 (Free license and download from Xilinx website)
- USB-to-UART Bridge Driver (See setup guides below)
  - ZedBoard: Cypress CY7C64225 http:/www.zedboard.org/documentation/1521
  - Zynq Mini-ITX: Silicon Labs CP210x http://www.zedboard.org/documentation/2056
- Tera Term (Version used is v4.75)
- VMware Player V5.0.0 (Version used is VMware-player-5.0.0-812388.exe)
- Adobe Reader for viewing PDF content (Version used is Adobe Reader X 10.1.4)
- 7-zip file archive utility

Hardware

- 64-bit host computer with at least 1.3/3/5 GB RAM (typical z7020/z7045/z7100) and up to 1.9/5/10 GB RAM (peak) for Windows-7 or Linux operating systems available to Vivado Design
  - http://www.xilinx.com/design-tools/vivado/memory.htm#zynq-7000
- SD/microSD card slot on PC or external USB-based SD/microSD card reader
- Avnet Zynq Development Kit (choose one)
  - Avnet ZedBoard Kit (**AES-Z7EV-7Z020-G)**
    - 2 USB cables (Type A to Micro-USB Type B) – *1 included in kit*
    - Micro-USB to USB Adapter cable
    - 4GB SD card – *included in kit*
    - 4-port powered USB Hub
    - CAT-5 Ethernet cable
  - Avnet Mini-ITX Z7045 Kit (**AES-MINI-ITX-7Z045-G**) or
    Avnet Mini-ITX Z7100 Kit (**AES-MINI-ITX-7Z100-G**)
    - 200W ATX Power Supply - *included in kit*
    - CAT-5 Ethernet cable - *included in kit*
    - 2 USB cables (Type A to Micro-USB Type B) - *included in kit*
    - 4 GB microSD card - *included in kit*
- USB Keyboard
- USB Mouse
- HDMI Cable
- HDMI Monitor with speakers or DVI Monitor with an HDMI-DVI adapter
- Headphones and microphone (or combined headset) with 1/8” audio jacks
- Male-to-male 1/8” audio jack connector cable (ADAU1761 audio jack tests)
- Optional powered external speakers (in place of headphones)

Technical Support

For technical support with any of the labs, please 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/design/1521/11

Zynq Mini-ITX Kit support page with Documentation and Reference Designs:

http://www.zedboard.org/design/2056/17

Contact your Avnet/Silica FAE or Avnet Support for any additional questions regarding the 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

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

Tutorial Format

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/microSD card assume that the card is connected to the host system via a USB adapter.

Tutorial Overview

This tutorial provides a means to integrate several different technologies on a single platform. Using Avnet Development Boards featuring the Zynq®-7000 All Programmable SoC, we have the power of an 800 MHz Cortex-A9 processor chip with dual 32-bit cores, combined with the unrivalled flexibility of Xilinx Programmable Logic to implement custom hardware systems. We use a Linux kernel as the foundation operating system, but also add a fully featured desktop from Ubuntu, contained in the root file system. The desktop allows the target 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, mimicking the environment of a standard desktop PC. This includes the potential to develop applications natively on the embedded ARM system.

Why do we want a desktop computer in an embedded system? Because this is the future of mobile computing. Soon the first smartphones will be commercially available implementing 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 Android use a common Linux kernel.

In fact, in December 2013, Canonical (the company that specializes in the Ubuntu version of Linux) announced that it had signed its first deal to supply a smartphone manufacturer with its mobile operating system. The first shipments with Ubuntu on high end smartphones were slated for 2014.

Lab 1 - FPGA Hardware Platform

Lab Overview

The hardware platform used for Ubuntu is created with Xilinx Vivado Design Suite. In this software tutorial we begin with an SDK project based on the exported hardware design.

1. Download the standalone **HDMI Bare Metal Reference Design using ADV7511 and ADI IP** for your target from the location provided in the **Tutorial Prerequisites** section.

2. At a minimum, follow the reference design instructions to **Create the SDK Workspace**. However, if you choose to complete the entire standalone reference design for your selected target[1], you will learn how to:

    - Design a hardware system capable of supporting an operating system
    - Synthesize, implement and generate a bitstream to encapsulate the hardware design using Vivado Design Suite
    - Export the relevant design files to a Xilinx SDK project
    - Test the hardware design on your selected target to ensure proper functionality of all system components

You may use any path you choose for the SDK workspace, but for the purposes of this tutorial the path used will be:

C:\ZedBoard\SDK_Ubuntu
or
C:\mitx\SDK_Ubuntu

3. The SDK workspace contains a description of the hardware project, including a bitstream to initialize the Zynq All Programmable SoC. Use Windows Explorer to copy the **system_top.bit** file to a new location to be used to gather the files we need for booting.

    Following the suggested path above, copy:

    C:\<Avnet target>\SDK_Ubuntu\hw_platform_0\system_top.bit

    to a new path:

    C:\<Avnet target>\bootfiles\system_top.bit

[1] Recommended procedure.

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

Lab Overview
Typically when booting 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.

A second stage loader is generally required to load an operating system. 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 program. For the purposes of running Linux on our embedded system, U-boot provides all the features needed to act as the second stage.

U-boot is the de-facto loader for embedded Linux because it has been adopted by almost every distribution. It runs on a wide range of processors and has been ported to innumerable boards. Online information is readily available and there are many support forums to aid developers in porting to new architectures. U-boot includes a command line interface and many configuration options, including the ability to pass parameters to the kernel to alter how the boot will proceed. It can load the kernel, the root file system (RFS) and the device tree and perform validation on the installation before passing control to the kernel for execution.

U-boot has already been ported to the Zynq architecture, so migrating it to a new Zynq board is a matter of considering the following points:

1. **<top level>/boards.cfg** - you must add a line to this file to identify your board to the configuration makefile (mkconfig).
2. **<top_level>/include/configs** – **zynq_<board_name>.h** and **zynq-common.h** are the two files that will contain the configuration information for U-boot. If your Zynq board does not exist in the directory, copy an existing file to create the zynq_<board_name>.h file, which must match the name, minus the .h, of the entry in boards.cfg. For example, for the Zynq Mini-ITX, copy the **zynq_zed.h** file (configuration for the Avnet ZedBoard) to create a new file **zynq_mitx.h**[2].
3. The **zynq_common.h** file should be identical for every Zynq board, but you may want to override some defaults, such as extending the default boot delay from 3 seconds, or changing the boot parameters controlled by jumper settings on the

[2] For the Zynq Mini-ITX, a configuration file has been included in the Supplied Files.

board. For example, by default the **sdboot** parameter assumes the RFS will be loaded to RAM by U-boot. For Ubuntu we chose to boot from an ext4 partition already present on the SD/microSD card. To override the default parameters, do not change the zynq-common.h file – instead change the zynq_<board_name>.h file and insert the changes/additions after the #include that references zynq-common.h. Parameters of the same name will override the parameters in the common file.

Why create the second stage image before we have the first stage loader in place? The initial bootstrap loader image contains 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 executable available.

When you have completed this lab, you will have learned to:

- Retrieve U-boot source code from the Xilinx repository
- Configure U-boot for the Avnet Zynq target
- Build U-boot for the Avnet Zynq 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:

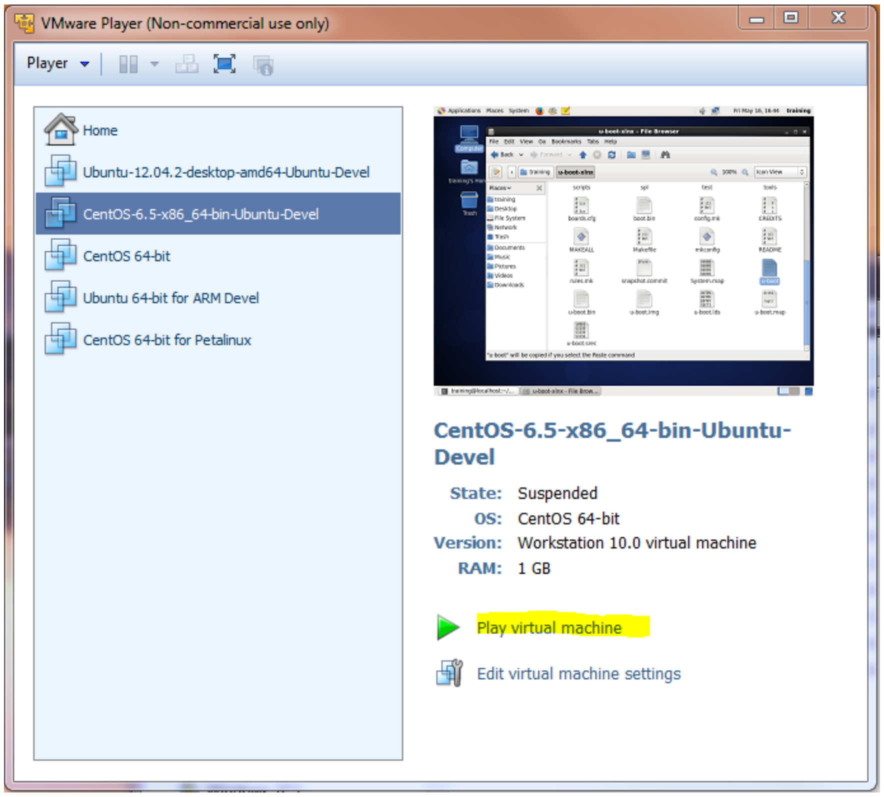
```
$ cd ~

$ git clone git://github.com/Xilinx/u-boot-xlnx.git

$ cd u-boot-xlnx

$ git checkout -b xilinx-v2013.4
```

Experiment 1 Step-by-Step Instruction:

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

Figure 2 – The VMware Player Application Icon

2. Select your virtual machine from the selections on the left and click on the **Play virtual machine** button to the right.


VMware Player (Non-commercial use only)
Player
Home
Ubuntu-12.04.2-desktop-amd64-Ubuntu-Devel
CentOS-6.5-x86_64-bin-Ubuntu-Devel
CentOS 64-bit
Ubuntu 64-bit for ARM Devel
CentOS 64-bit for Petalinux
Play virtual machine
Edit virtual machine settings

CentOS-6.5-x86_64-bin-Ubuntu-Devel

State: Suspended
OS: CentOS 64-bit
Version: Workstation 10.0 virtual machine
RAM: 1 GB

Figure 3 – The VMware Player Application

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

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.

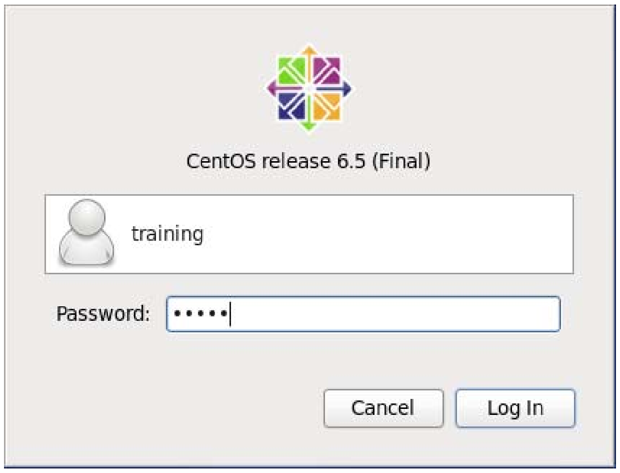


Figure 4 – The CentOS Workstation Login

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

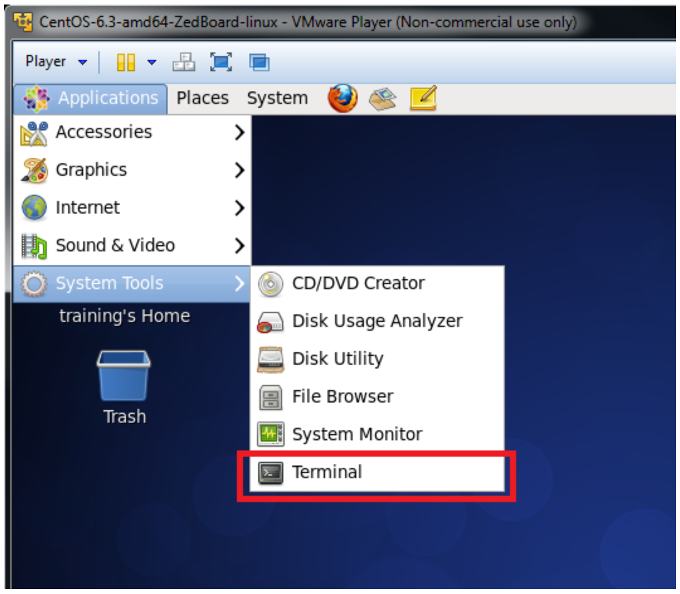


Figure 5 – Launching the CentOS Terminal from the Desktop

5. The Xilinx U-boot Git repository can be browsed at:

https://github.com/Xilinx

To retrieve 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.

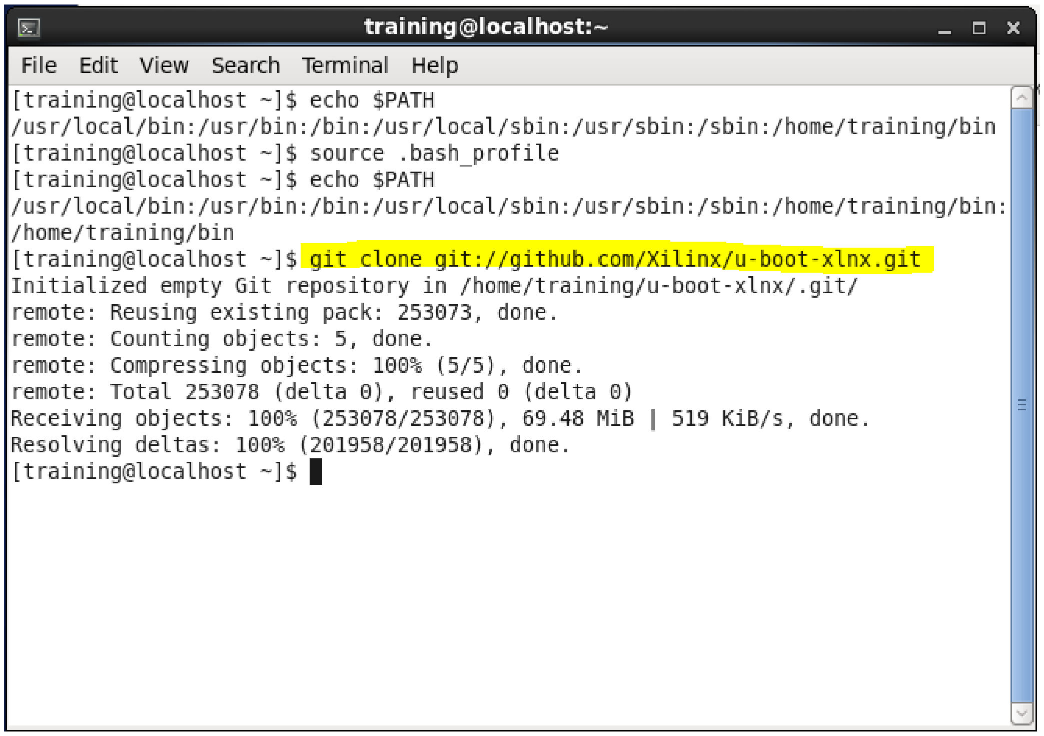
Use the following Git command to clone the repository.

```
$ git clone git://github.com/Xilinx/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 convenient 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



```
[training@localhost ~]$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/training/bin
[training@localhost ~]$ source .bash_profile
[training@localhost ~]$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/training/bin:
/home/training/bin
[training@localhost ~]$ git clone git://github.com/Xilinx/u-boot-xlnx.git
Initialized empty Git repository in /home/training/u-boot-xlnx/.git/
remote: Reusing existing pack: 253073, done.
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 253078 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (253078/253078), 69.48 MiB | 519 KiB/s, done.
Resolving deltas: 100% (201958/201958), done.
[training@localhost ~]$ 
```

Figure 6 – 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 selected tools release, which are kept under a designated Git 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. The Linux kernel that we will be using was ported for Ubuntu by Analog Devices, and at the time of writing was version 3.14.0. This is compatible with the U-boot version in the Xilinx repository from release 2013.4.

```
$ git checkout -b xilinx-v2013.4
```

Experiment 2: Configuring U-boot for the Avnet Zynq 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 Zynq 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 Zynq target. Edit the **.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    (ZedBoard ONLY)

$ make zynq_mitx_config   (Mini-ITX Board ONLY)
```

Experiment 2 Step-by-Step Instruction:

1. If the virtual machine is not already open, launch the VMware Player application and open a Terminal window. See the Step-by-Step Instruction 1 through 4 of **Lab 2, Experiment 1,** if you don't remember how to do this.

2. Change from the home directory into the U-boot source directory.

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

3. 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 previous configurations as well as any residual files left by prior builds.

```
$ make distclean
```

4. A board configuration is included for each Avnet Zynq target. The top level board-specific headers can be seen in:

   /include/configs/zynq_zed.h (ZedBoard ONLY)

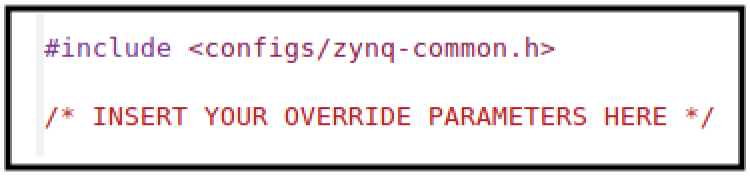
   /include/configs/zynq_mitx.h[3] **(Mini-ITX Board ONLY)**

   These files reference a second file where parameters common to all Zynq boards are defined. This file is:

   /include/configs/zynq-common.h.

   The **zynq-common.h** file provides default configuration assignments for parameters we may wish to override. For example, U-boot can pass a parameter to the Linux kernel to tell it where to find the root file system. By default, it expects the RFS to be located in RAM, but for Ubuntu we prefer it to remain on removable media (SD or microSD). To override the sdboot parameter, copy the entire #define section for CONFIG_EXTRA_ENV_SETTINGS[4] from zynq-common.h to the zynq_<board_name>.h file, inserting it after the #include for the zynq-common.h file:

[3] If the current kernel source does not include this file, and you did not create one in an earlier step, you may use the version included in the *Supplied Files*.
[4] Note that the entire section is one long line, with the backslash indicating continuation at the end of each line in the file.



```
#include <configs/zynq-common.h>

/* INSERT YOUR OVERRIDE PARAMETERS HERE */
```

Figure 7 – Override U-boot Parameters

Edit the sdboot "line" as shown in following code snippet. Note there is no ramdisk load, since it exists on the ext4 partition already. And the bootm command has been changed to remove the load address used by a ramdisk.

Updated:

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

Original:

```
"sdboot=if mmcinfo; then " \
                "run uenvboot; " \
                "echo Copying Linux from SD to RAM... && " \
                "fatload mmc 0 0x3000000 ${kernel_image} && " \
                "fatload mmc 0 0x2A00000 ${devicetree_image} && " \
                "fatload mmc 0 0x2000000 ${ramdisk_image} && " \
                "bootm 0x3000000 0x2000000 0x2A00000; " \
```

Refining the constant value will cause the compiler to generate numerous warnings. Eliminate the warnings by placing the following code immediately prior to the updated constant:

```
#ifdef CONFIG_EXTRA_ENV_SETTINGS
#undef CONFIG_EXTRA_ENV_SETTINGS
#endif
```

Other parameters you may wish to override are the boot delay and boot prompt. The original versions are:

```
#define CONFIG_SYS_PROMPT              "zynq-uboot> "
#define CONFIG_BOOTDELAY               3 /* -1 to Disable autoboot */
```

You may add these lines (and corresponding #ifdef blocks) with your preferred changes after the CONFIG_EXTRA_ENV_SETTINGS entry. Save the file and close your editor.

5. Configure U-boot for the Avnet Zynq target by using our modified configuration.

```
$ make zynq_zed_config    (ZedBoard ONLY)

$ make zynq_mitx_config   (Mini-ITX Board ONLY)
```

Experiment 3: Building U-boot from the Configured Source

Now that the source tree has been configured for the Avnet Zynq 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 later 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 Avnet Zynq 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 Instruction:

1. In the previous experiment, the U-boot source tree was configured for an Avnet Zynq target platform.

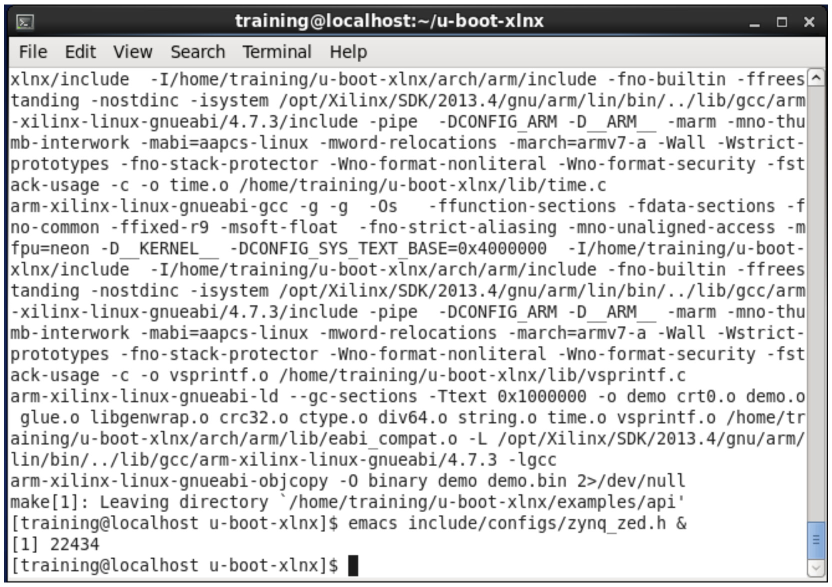
   First, make sure the **/home/training/u-boot-xlnx/** folder is the current working directory by using the pwd command. If the working directory shown is not the **/home/training/u-boot-xlnx/** folder, change directories to that folder.

```
$ pwd

$ cd ~/u-boot-xlnx   (if necessary)
```

2. 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 below, you may continue with the next step.

```
$ make
```


training@localhost:~/u-boot-xlnx
File Edit View Search Terminal Help

```
xlnx/include  -I/home/training/u-boot-xlnx/arch/arm/include -fno-builtin -ffrees
tanding -nostdinc -isystem /opt/Xilinx/SDK/2013.4/gnu/arm/lin/bin/../lib/gcc/arm
-xilinx-linux-gnueabi/4.7.3/include -pipe  -DCONFIG_ARM -D__ARM__ -marm -mno-thu
mb-interwork -mabi=aapcs-linux -mword-relocations -march=armv7-a -Wall -Wstrict-
prototypes -fno-stack-protector -Wno-format-nonliteral -Wno-format-security -fst
ack-usage -c -o time.o /home/training/u-boot-xlnx/lib/time.c
arm-xilinx-linux-gnueabi-gcc -g -g  -Os   -ffunction-sections -fdata-sections -f
no-common -ffixed-r9 -msoft-float  -fno-strict-aliasing -mno-unaligned-access -m
fpu=neon -D__KERNEL__ -DCONFIG_SYS_TEXT_BASE=0x4000000  -I/home/training/u-boot-
xlnx/include  -I/home/training/u-boot-xlnx/arch/arm/include -fno-builtin -ffrees
tanding -nostdinc -isystem /opt/Xilinx/SDK/2013.4/gnu/arm/lin/bin/../lib/gcc/arm
-xilinx-linux-gnueabi/4.7.3/include -pipe  -DCONFIG_ARM -D__ARM__ -marm -mno-thu
mb-interwork -mabi=aapcs-linux -mword-relocations -march=armv7-a -Wall -Wstrict-
prototypes -fno-stack-protector -Wno-format-nonliteral -Wno-format-security -fst
ack-usage -c -o vsprintf.o /home/training/u-boot-xlnx/lib/vsprintf.c
arm-xilinx-linux-gnueabi-ld --gc-sections -Ttext 0x1000000 -o demo crt0.o demo.o
 glue.o libgenwrap.o crc32.o ctype.o div64.o string.o time.o vsprintf.o /home/tr
aining/u-boot-xlnx/arch/arm/lib/eabi_compat.o -L /opt/Xilinx/SDK/2013.4/gnu/arm/
lin/bin/../lib/gcc/arm-xilinx-linux-gnueabi/4.7.3 -lgcc
arm-xilinx-linux-gnueabi-objcopy -O binary demo demo.bin 2>/dev/null
make[1]: Leaving directory `/home/training/u-boot-xlnx/examples/api'
[training@localhost u-boot-xlnx]$ emacs include/configs/zynq_zed.h &
[1] 22434
[training@localhost u-boot-xlnx]$
```

Figure 8 – U-boot Build Completed

3. When the make has completed, the U-boot hierarchy contains a tool for converting binary files into image files that can be loaded by U-boot. We will need this tool later when building the Linux kernel, so it is useful to place this tool into a directory that is part of the execution search path.

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

4. Now that we have a successful build, open a file browser window through the **Applications→System Tools→File Browser** menu item.

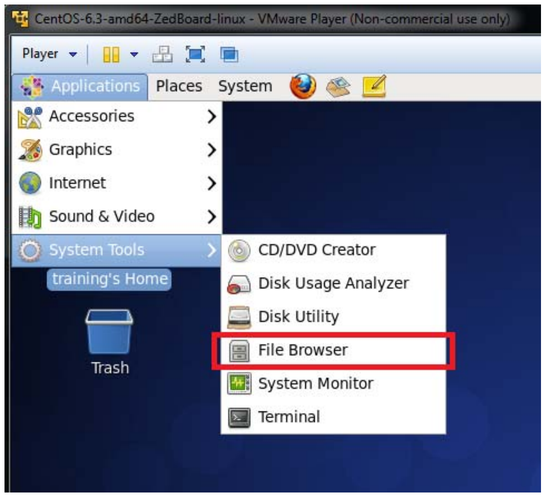
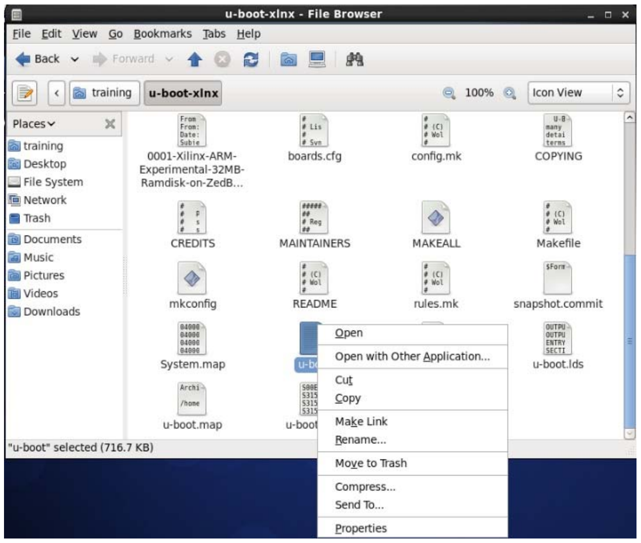


Figure 9 – CentOS File Browser

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.



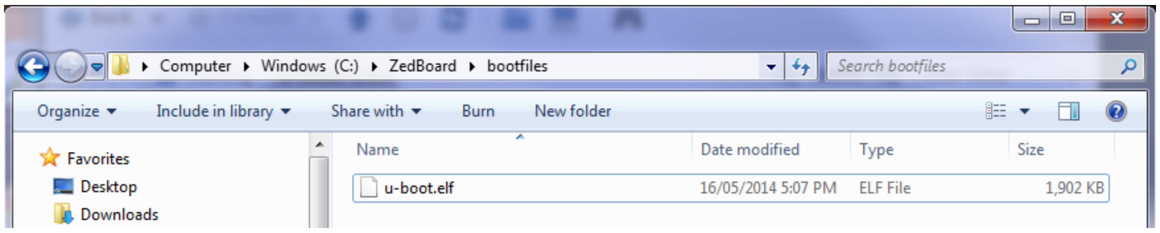
"u-boot" selected (716.7 KB)

Figure 10 – Copying U-boot Executable to Virtual Machine Clipboard

6. Paste the U-boot executable into the host operating system, using Windows Explorer to create the following folder path:

   C:\<Avnet_target>[5]\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.



Name	Date modified	Type	Size
u-boot.elf	16/05/2014 5:07 PM	ELF File	1,902 KB

Figure 11 – u-boot Executable Copied to the Host Machine and Renamed

[5] Avnet target is one of: ZedBoard or Mini-ITX

Lab 3 – Partition the SD Card for Booting

Lab Overview

The Avnet Zynq target 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 Avnet Zynq target. 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 Virtual Machine using a utility called the Gnome Partition Utility (GParted).

Experiment 1 General Instruction:

Use GParted to create two empty partitions on a 4 MB (or larger) SD card. The first partition will be 52MB to 400 MB (depending on the capacity of your media) FAT32 format, and the second partition will use the remaining space in ext4 format.

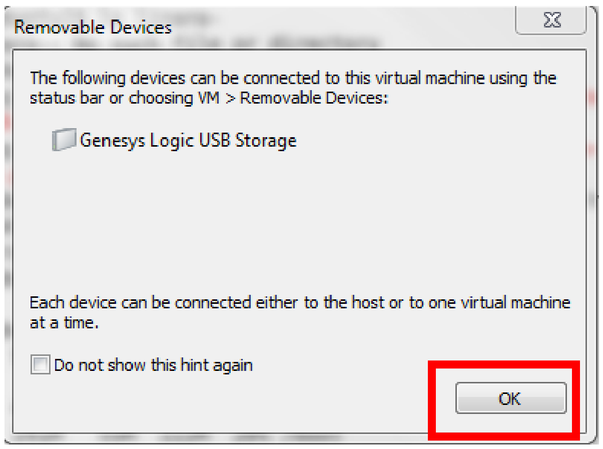
Note: All existing data on the SD card will be destroyed. Make sure you back up any data you need to retain prior to the formatting operation.

It should be noted here that not all SD cards of the same capacity offer the same performance, and even cards of the same advertised capacity may differ between manufacturers. This can create boot issues if you are copying an image from one card to another. If you are using multiple cards, it is best to stick to a single manufacturer to reduce boot problems resulting from bad images.

The data transfer rate of SD cards affects the performance of Linux significantly when the root file system resides on the removable media. Both SD and microSD cards come in different classes, from 4 to 10 at the time of writing, and within a specific manufacturer the transfer rate is generally higher with a higher class. This is not always the case between manufacturers, however. For the best performance, use a card that states its maximum transfer rate. During testing, we used a 16 MB class 10 card with a 30 MBps stated transfer rate to create a system with very good response from the Ubuntu Desktop.

Experiment 1 Step-by-Step Instruction:

1. 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.


Removable Devices

The following devices can be connected to this virtual machine using the status bar or choosing VM > Removable Devices:

Genesys Logic USB Storage

Each device can be connected either to the host or to one virtual machine at a time.

Figure 12 – USB Detection by Virtual Machine

2. 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.

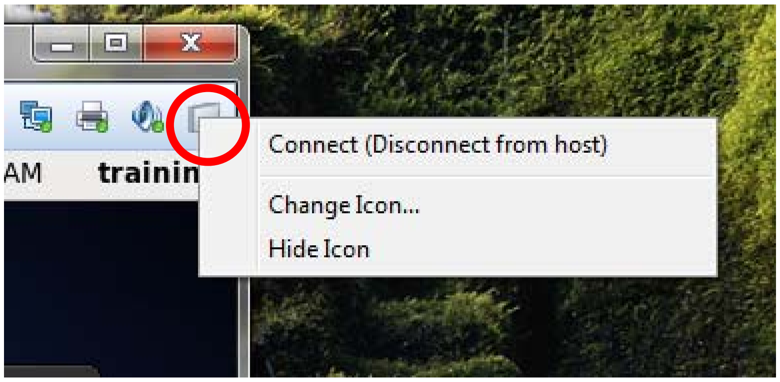
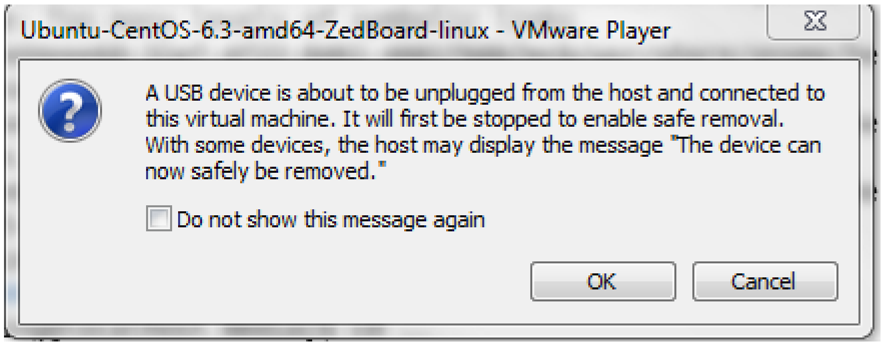


Figure 13 – Connect USB Device to Virtual Machine

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


Ubuntu-CentOS-6.3-amd64-ZedBoard-linux - VMware Player

A USB device is about to be unplugged from the host and connected to this virtual machine. It will first be stopped to enable safe removal. With some devices, the host may display the message "The device can now safely be removed."

Do not show this message again

Figure 14 – USB Transfer Warning from Virtual Machine

4. 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**.

5. To run the GParted Partition Editor:

   CentOS: in the VM menu select **Applications | System Tools | GParted Partition Editor**.

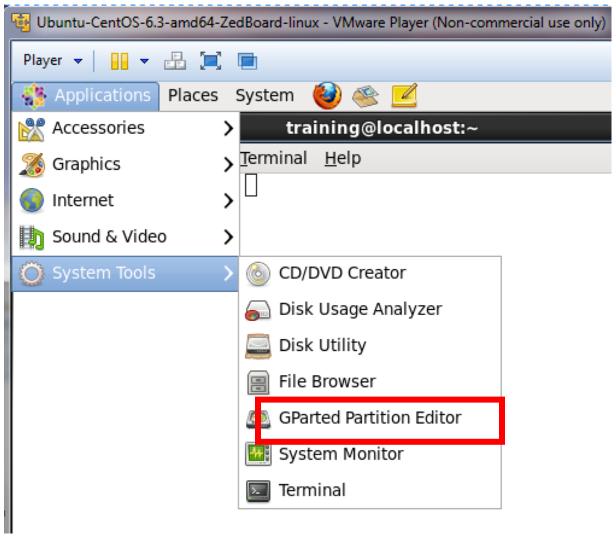

Ubuntu-CentOS-6.3-amd64-ZedBoard-linux - VMware Player (Non-commercial use only)

Figure 15 – Launch GParted Partition Editor in CentOS

Ubuntu: Click on the Dashboard icon and start typing the first few characters of gparted into the search field. When the GParted Partition Editor appears, double-click the icon to launch it.

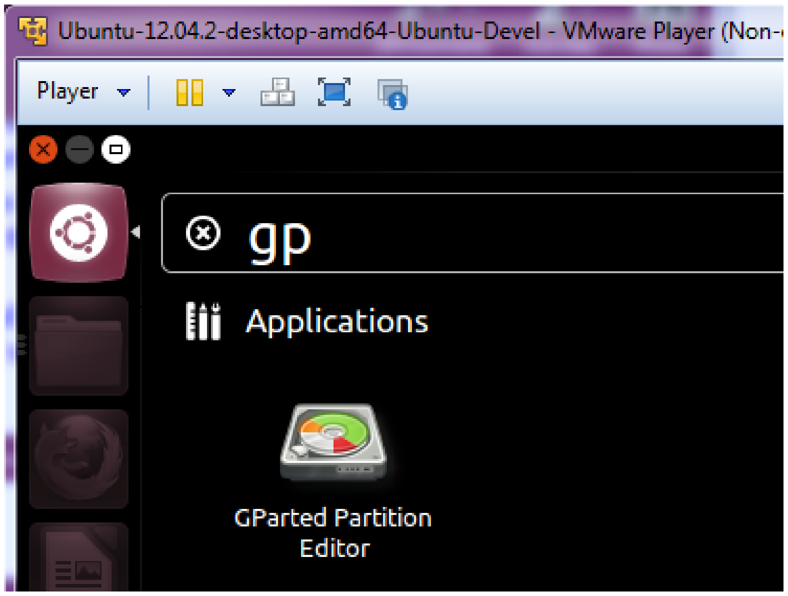
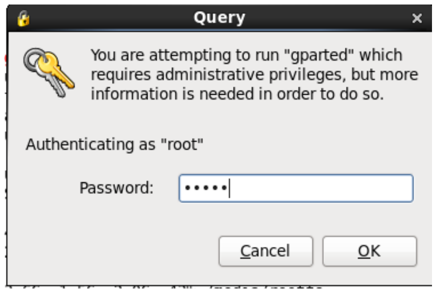

Ubuntu-12.04.2-desktop-amd64-Ubuntu-Devel - VMware Player (Non-
Player
gp
Applications
GParted Partition Editor

Figure 16 – Launch GParted Partition Editor in Ubuntu

6. Enter the root password when prompted. Click **OK** or **Authenticate**.

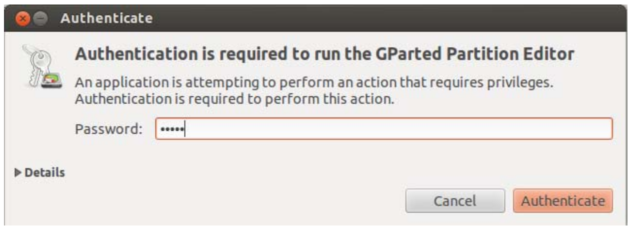


You are attempting to run "gparted" which requires administrative privileges, but more information is needed in order to do so.

Authenticating as "root"

Password: •••••

CentOS



Authentication is required to run the GParted Partition Editor

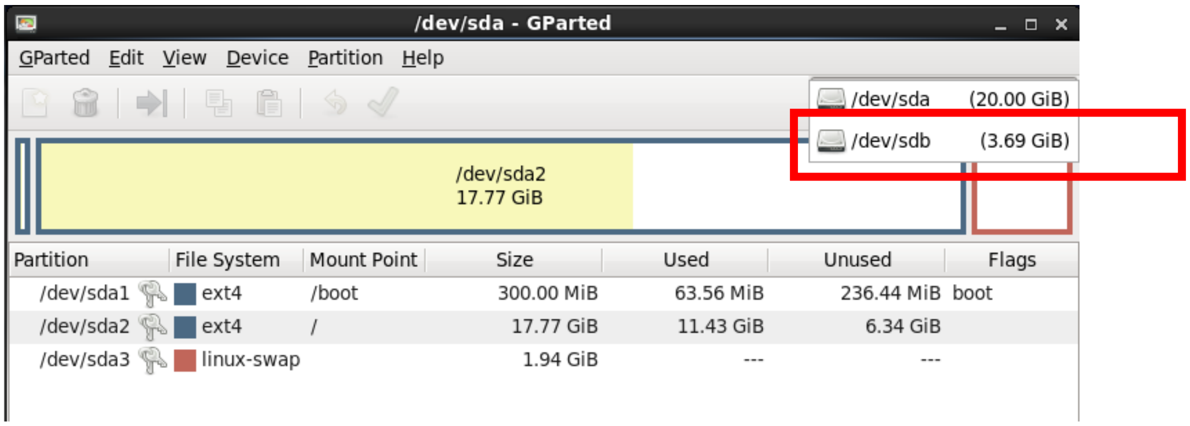
An application is attempting to perform an action that requires privileges. Authentication is required to perform this action.

Password: •••••

Ubuntu

Figure 17 – Enter Root Password

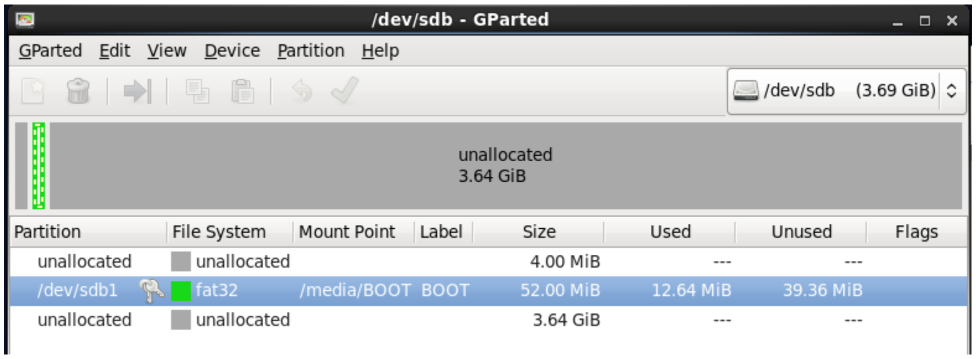
7. GParted will by default display the partitions on your Linux development system. **You DO NOT want to change anything here!** In the upper right, open the drop-down menu to display the available devices, and select the one corresponding to the SD card. This will generally be labelled as **/dev/sdb**, with a size shown that is slightly smaller than the full size of your SD card. Click on this device to select it (This example shows a 4 GB capacity SD/microSD card).



Partition	File System	Mount Point	Size	Used	Unused	Flags
/dev/sda1	ext4	/boot	300.00 MiB	63.56 MiB	236.44 MiB	boot
/dev/sda2	ext4	/	17.77 GiB	11.43 GiB	6.34 GiB	
/dev/sda3	linux-swap		1.94 GiB	---	---	

Figure 18 – GParted SD Card Selection

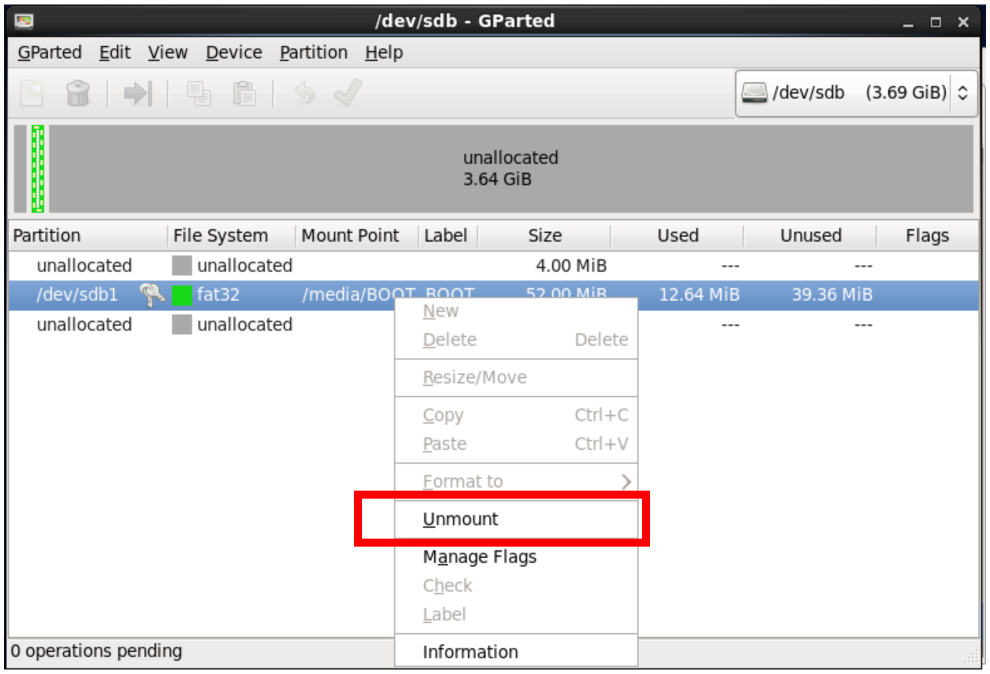
8. If your SD card has been used before, it may contain a file system already. Any existing system not specifically used for the Ubuntu boot process should be removed so we can start with a clean slate. In the example below we have a 4 GB SD card with a FAT32 file system that was created on a Windows 7 system. If the card is empty, you may skip to ***step 14***.



Partition	File System	Mount Point	Label	Size	Used	Unused	Flags
unallocated	unallocated			4.00 MiB	---	---	
/dev/sdb1	fat32	/media/BOOT	BOOT	52.00 MiB	12.64 MiB	39.36 MiB	
unallocated	unallocated			3.64 GiB	---	---	

Figure 19 – Existing SD Card Partitions

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



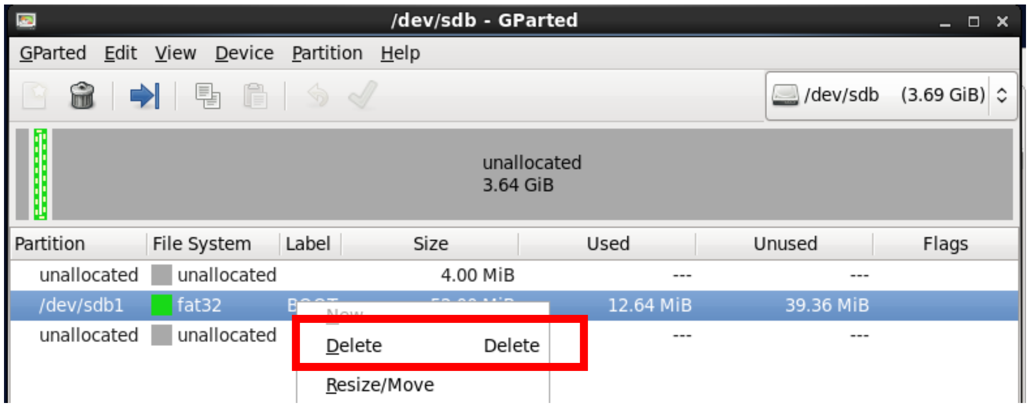
/dev/sdb - GParted

Partition	File System	Mount Point	Label	Size	Used	Unused	Flags
unallocated	unallocated			4.00 MiB	---	---	
/dev/sdb1	fat32	/media/BOOT	BOOT	52.00 MiB	12.64 MiB	39.36 MiB	
unallocated	unallocated				---	---	

0 operations pending

Figure 20 – Unmount an Existing Partition

10. 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.

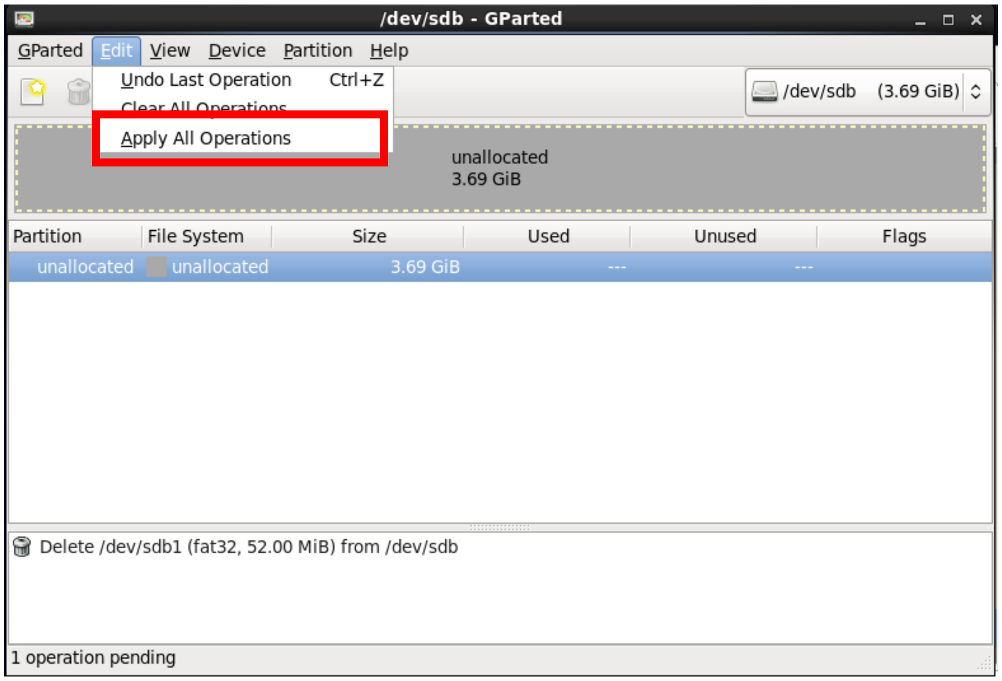


/dev/sdb - GParted

Partition	File System	Label	Size	Used	Unused	Flags
unallocated	unallocated		4.00 MiB	---	---	
/dev/sdb1	fat32	BOOT	52.00 MiB	12.64 MiB	39.36 MiB	
unallocated	unallocated			---	---	

Figure 21 – Delete Existing SD Card Partition

11. 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.



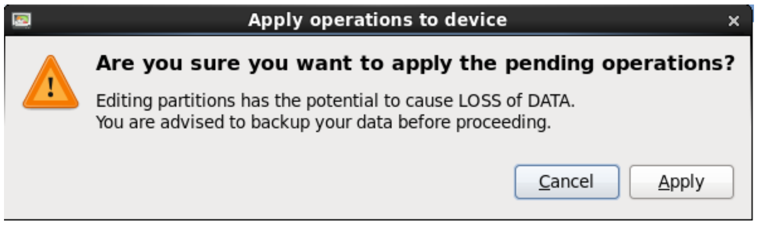
Partition	File System	Size	Used	Unused	Flags
unallocated	unallocated	3.69 GiB	---	---	

Delete /dev/sdb1 (fat32, 52.00 MiB) from /dev/sdb

1 operation pending

Figure 22 – GParted Perform Pending Operations

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



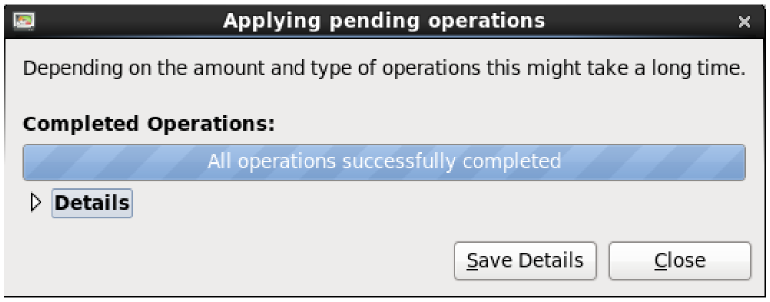
Are you sure you want to apply the pending operations?

Editing partitions has the potential to cause LOSS of DATA.
You are advised to backup your data before proceeding.

Figure 23 – 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.

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


Applying pending operations

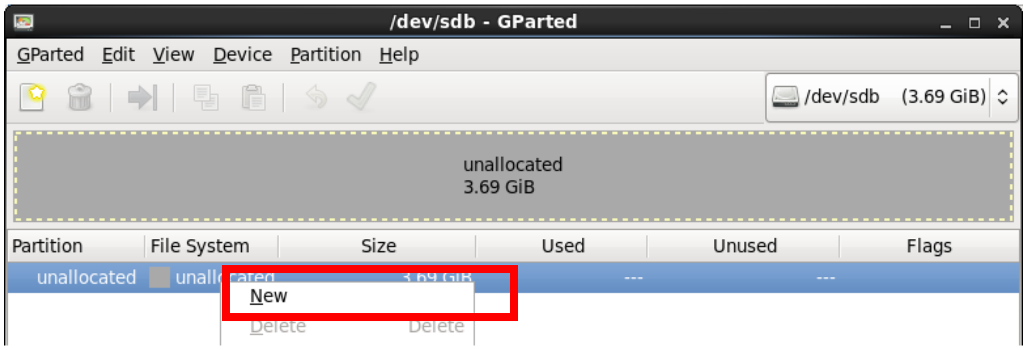
Depending on the amount and type of operations this might take a long time.

Completed Operations:

All operations successfully completed

Figure 24 – GParted Operations Successful

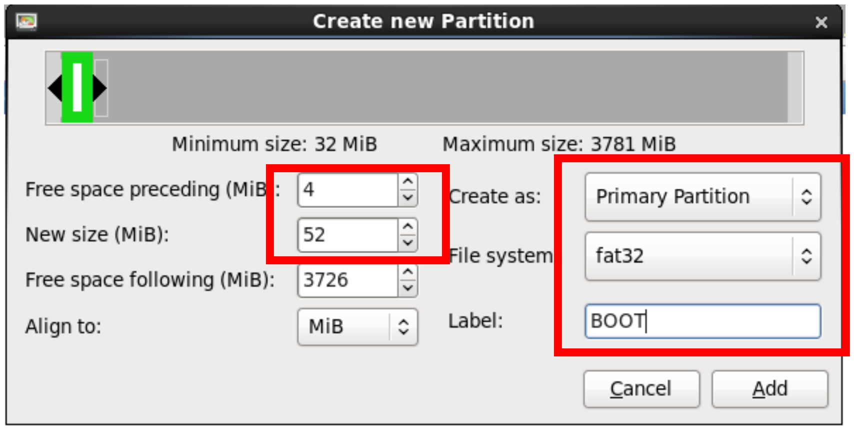
14. Your SD card should now appear with the full space completely unallocated. Right click on the unallocated space and select **New**.



Partition	File System	Size	Used	Unused	Flags
unallocated	unallocated	3.69 GiB	---	---	

Figure 25 – Create New SD Card Partition

15. In the *Create New Partition* window, enter the parameters shown below[6] and click the **Add** button. The operation will queue up in the lower window in GParted.

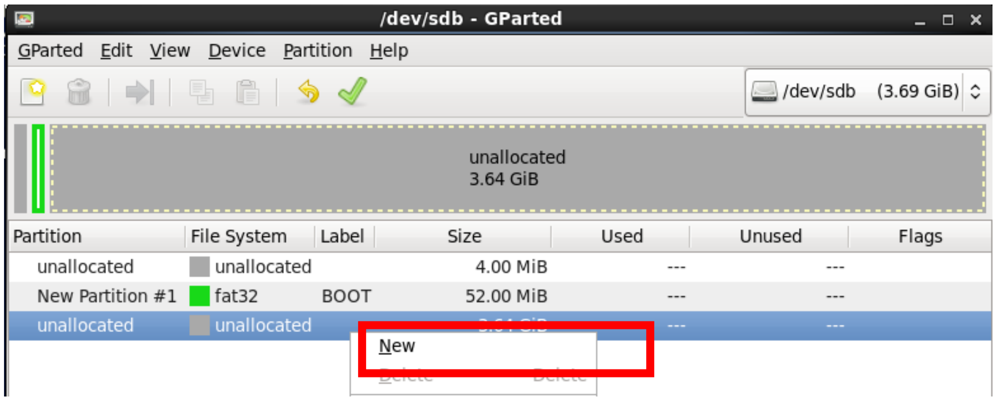


Minimum size: 32 MiB Maximum size: 3781 MiB

Free space preceding (MiB): 4
New size (MiB): 52
Free space following (MiB): 3726
Align to: MiB
Create as: Primary Partition
File system: fat32
Label: BOOT

Figure 26 – Create FAT32 Partition

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

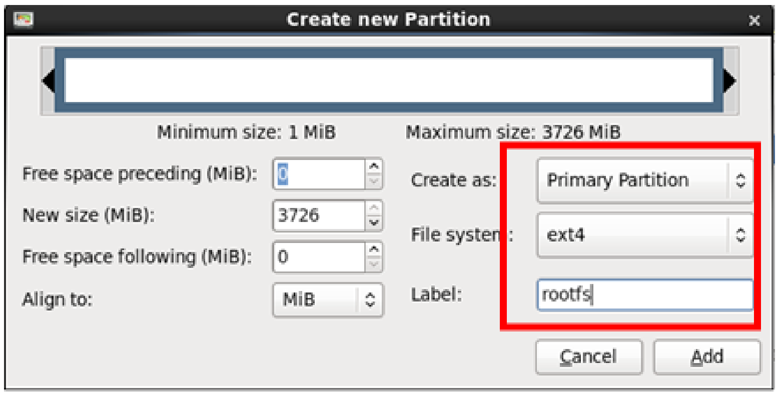


Partition	File System	Label	Size	Used	Unused	Flags
unallocated	unallocated		4.00 MiB	---	---	
New Partition #1	fat32	BOOT	52.00 MiB	---	---	
unallocated	unallocated		3.64 GiB	---	---	

Figure 27 – Create New SD Card Partition

[6] 52 MB is the minimum recommended FAT32 partition size for the boot images, suitable for a 4 GB SD/microSD card. If you have a higher capacity, you may want to increase this partition size to allow you more flexibility on the files you can save there. 400 MB is typically large enough to accommodate any extra files you wish to retain.

17. 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.



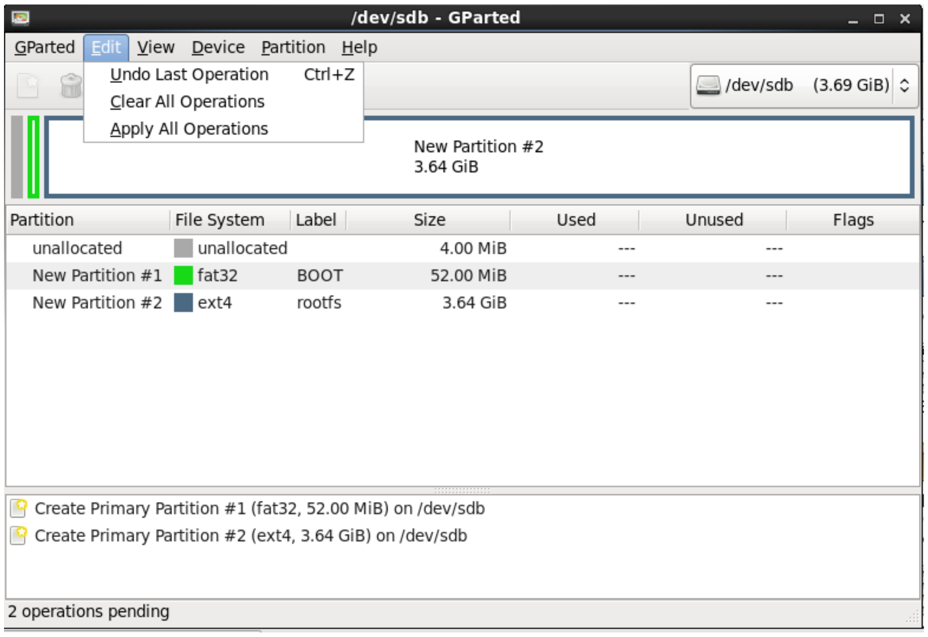
Create new Partition

Minimum size: 1 MiB Maximum size: 3726 MiB

Free space preceding (MiB): 0
New size (MiB): 3726
Free space following (MiB): 0
Align to: MiB
Create as: Primary Partition
File system: ext4
Label: rootfs

Figure 28 – Create New ext4 Partition

18. 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.**



/dev/sdb - GParted

/dev/sdb (3.69 GiB)

New Partition #2
3.64 GiB

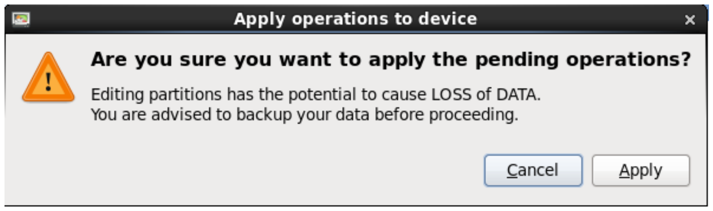
Partition	File System	Label	Size	Used	Unused	Flags
unallocated	▨ unallocated		4.00 MiB	---	---	
New Partition #1	🟩 fat32	BOOT	52.00 MiB	---	---	
New Partition #2	🟦 ext4	rootfs	3.64 GiB	---	---	

Create Primary Partition #1 (fat32, 52.00 MiB) on /dev/sdb
Create Primary Partition #2 (ext4, 3.64 GiB) on /dev/sdb

2 operations pending

Figure 29 – Partition SD Card with FAT32 and ext4

19. 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.



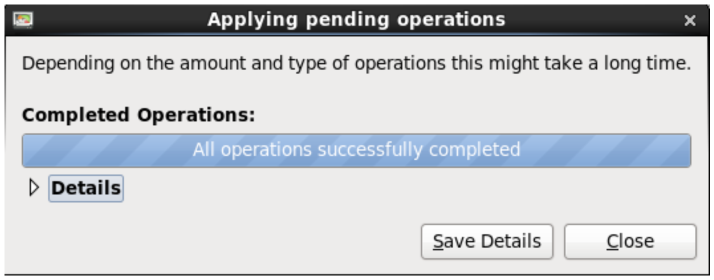
Apply operations to device

Are you sure you want to apply the pending operations?

Editing partitions has the potential to cause LOSS of DATA.
You are advised to backup your data before proceeding.

Figure 30 – Verify Partition Creation Operation

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



Applying pending operations

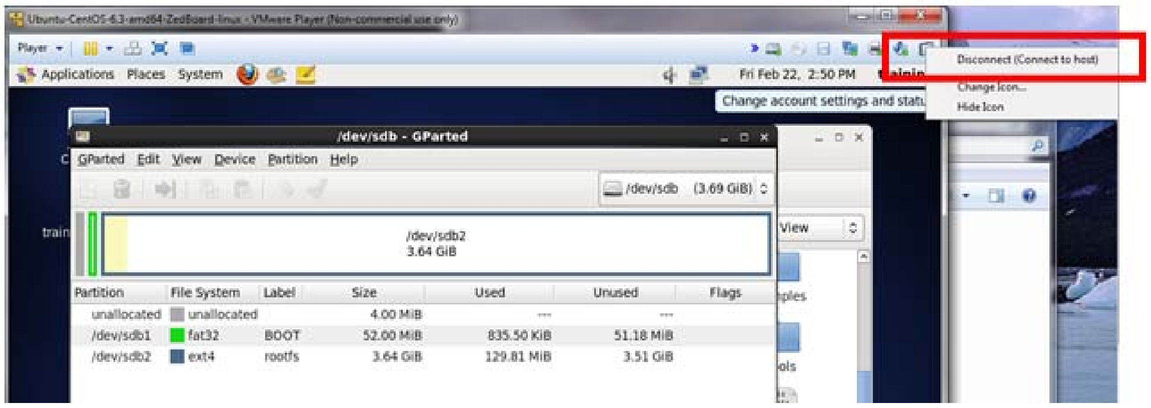
Depending on the amount and type of operations this might take a long time.

Completed Operations:

All operations successfully completed

Figure 31 – Partition Creation Successful

21. Your SD card is now ready for the files that will be used to boot Ubuntu on the Avnet Zynq target. 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**.



Ubuntu-CentOS-6.3-amd64-ZedBoard-linux - VMware Player (Non-commercial use only)
Player
Applications Places System
Fri Feb 22, 2:50 PM
Disconnect (Connect to host)
Change Icon...
Hide Icon
Change account settings and status
/dev/sdb - GParted
GParted Edit View Device Partition Help
/dev/sdb (3.69 GiB)
/dev/sdb2
3.64 GiB

Partition	File System	Label	Size	Used	Unused	Flags
unallocated	unallocated		4.00 MiB	---	---	
/dev/sdb1	fat32	BOOT	52.00 MiB	835.50 KiB	51.18 MiB	
/dev/sdb2	ext4	rootfs	3.64 GiB	129.81 MiB	3.51 GiB	

Figure 32 – Disconnect USB Device

22. Close the *GParted* window.

23. 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 media. The card is now ready for use in subsequent labs.

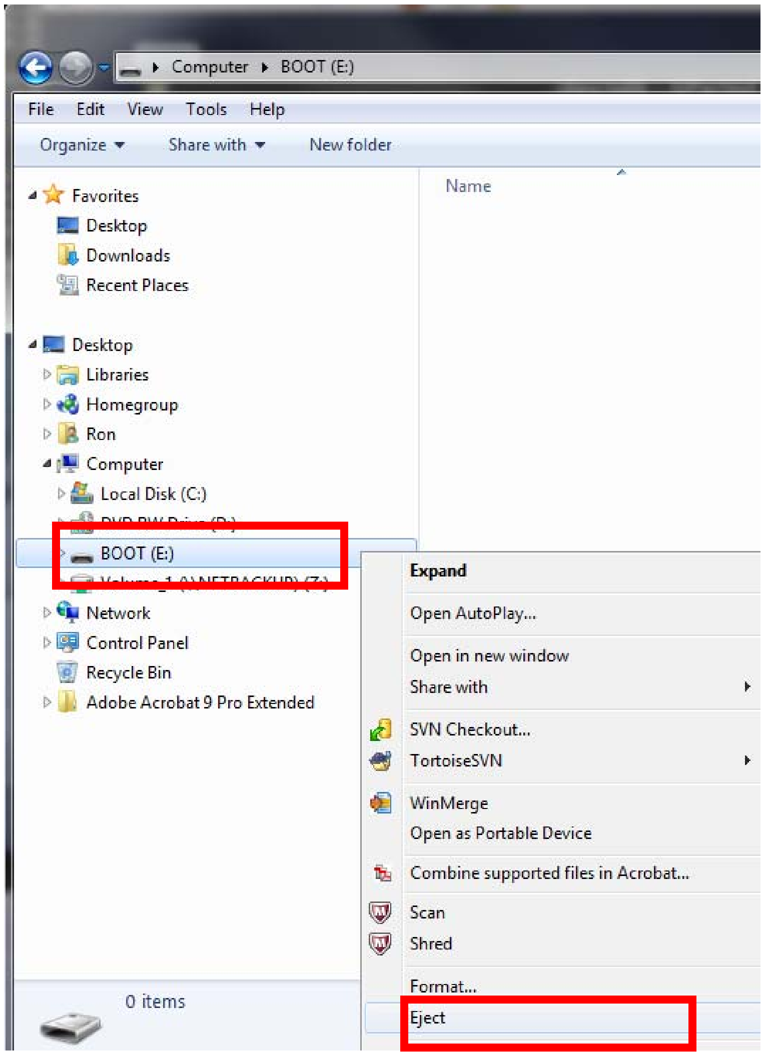


Figure 33 – Remove SD Card from Host

Lab 4 - Create the First Stage Boot Loader

Lab Overview

At this point all the components necessary for initial bring-up of the Avnet Zynq target have been created, except for the First Stage Boot Loader (FSBL). We have the hardware bitstream file from “Lab 1 - FPGA Hardware Platform”, and in the second lab we configured and built U-boot to serve as the second stage loader that will ultimately launch the operating system. In Lab 3 we initialized the boot media.

In this lab we will use the Xilinx SDK to create the FSBL and combine it with the other two 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:

- Build an FSBL for the Avnet target
- Create a boot image
- Boot the Avnet target to a U-boot prompt

Experiment 1: Create the First Stage Boot Loader

Experiment 1 General Instruction:

Launch Xilinx Software Development Kit (SDK) and open the workspace containing the Standalone platform from Lab 1. Create an FSBL project, allow the SDK to generate the default source code, and if necessary, modify the code to customize it for your selected target. An example of required customization is shown in the Step-by-Step Instruction for the Avnet Mini-ITX board (ZedBoard does not require any customization).

Experiment 1 Step-by-Step Instruction:

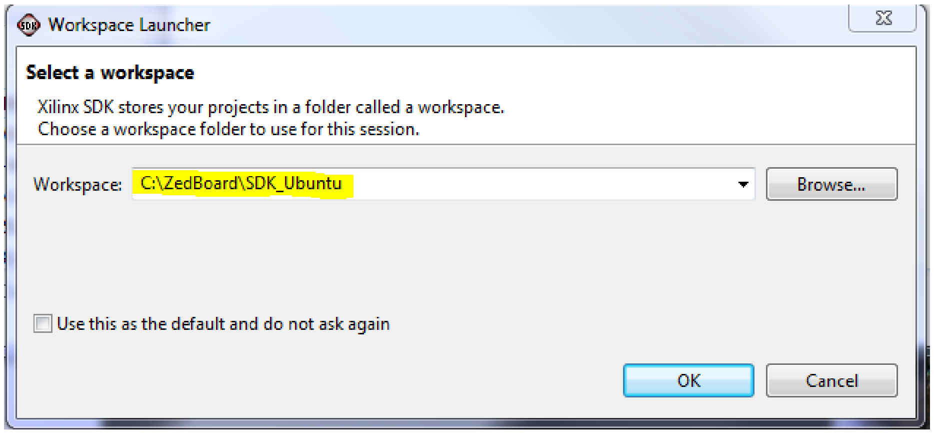
1. Launch Xilinx Software Development Kit (SDK). **Start → All Programs → Xilinx Design Tools → Vivado 2013.4 → SDK → Xilinx SDK 2013.4**.

Figure 34 – The SDK Application Icon

2. Set or switch the workspace to the path for your Avnet target. This is the path to the SDK workspace from Lab 1. The suggested values were:

C:\ZedBoard\SDK_Ubuntu
or
C:\mitx\SDK_Ubuntu

Click the **OK** button.


Workspace Launcher

Select a workspace

Xilinx SDK stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\ZedBoard\SDK_Ubuntu

Figure 35 – Select the SDK Workspace (ZedBoard)

3. When the SDK launch is complete, your SDK project should have, at a minimum, the hw_platform_0 project installed. If you have the first stage boot loader project from the *HDMI Standalone Reference Design*, you may skip to Step 6.

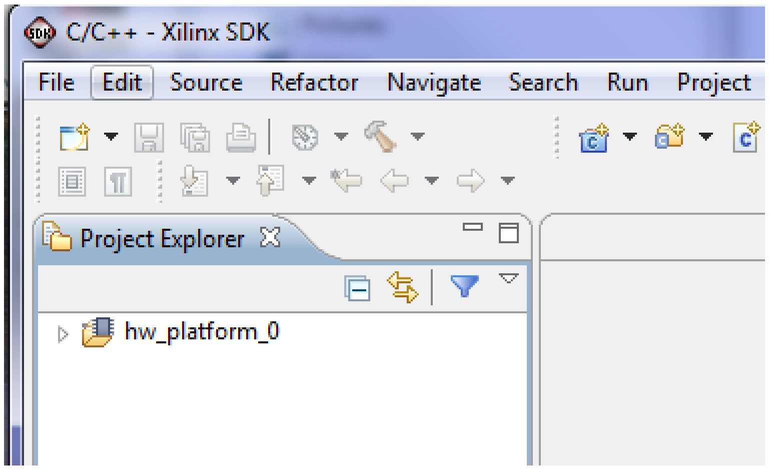
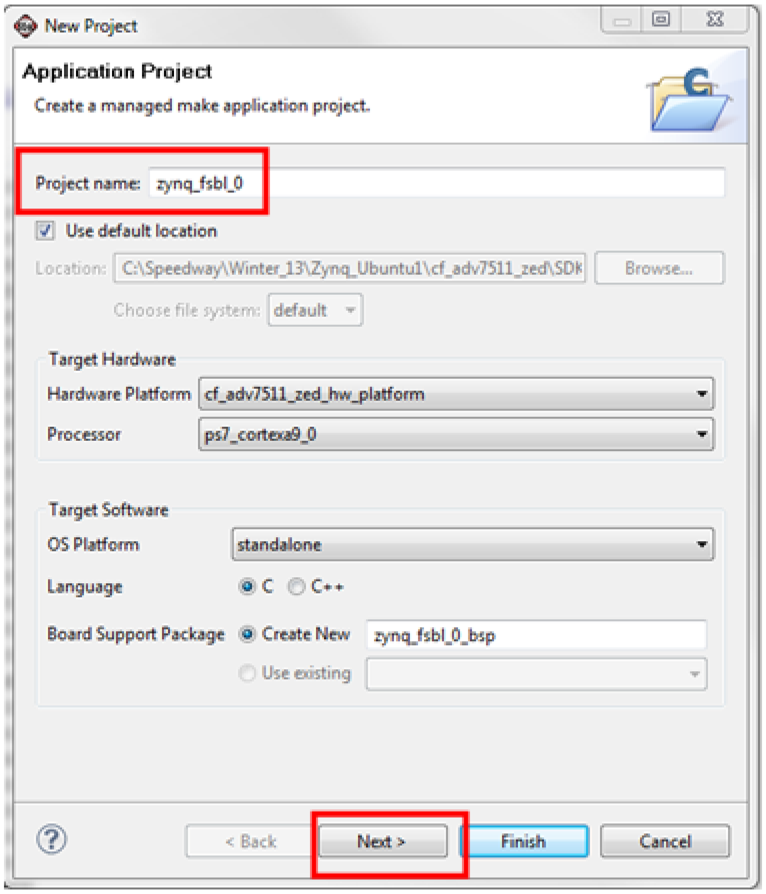


Figure 36 – Xilinx Software Development Kit

4. In order to boot the Avnet target, we need a bootstrap loader to create the processor environment used to load an operating system. To do that we can 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*, accept the remaining default values and click **Next**.


New Project

Application Project

Create a managed make application project.

Project name: zynq_fsbl_0

Use default location

Location: C:\Speedway\Winter_13\Zynq_Ubuntu1\cf_adv7511_zed\SDK

Choose file system: default

Target Hardware

Hardware Platform: cf_adv7511_zed_hw_platform

Processor: ps7_cortexa9_0

Target Software

OS Platform: standalone

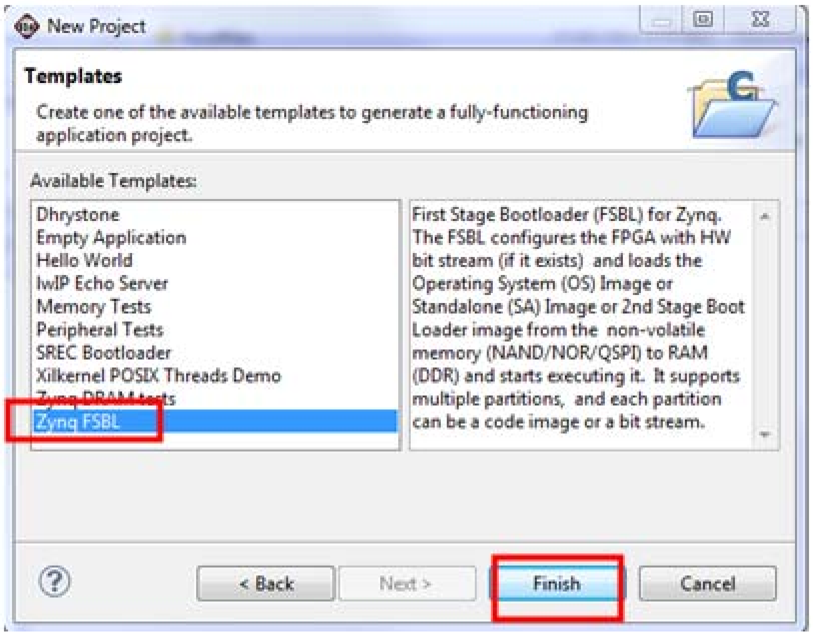
Language: [✓ C] C++

Board Support Package: [✓ Create New] zynq_fsbl_0_bsp

Use existing

Figure 37 – First Stage Bootloader Project

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



Create one of the available templates to generate a fully-functioning application project.

Available Templates:

Dhrystone
Empty Application
Hello World
lwIP Echo Server
Memory Tests
Peripheral Tests
SREC Bootloader
Xilkernel POSIX Threads Demo
Zynq DRAM tests
Zynq FSBL

First Stage Bootloader (FSBL) for Zynq. The FSBL configures the FPGA with HW bit stream (if it exists) and loads the Operating System (OS) Image or Standalone (SA) Image or 2nd Stage Boot Loader image from the non-volatile memory (NAND/NOR/QSPI) to RAM (DDR) and starts executing it. It supports multiple partitions, and each partition can be a code image or a bit stream.

Figure 38 – First Stage Boot Loader Template

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

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

C:\<Avnet_target>\SDK_Ubuntu\zynq_fsbl_0\Debug\zynq_fsbl_0.elf

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

C:\<Avnet_target>\bootfiles

Paste the elf file in at that location. We will be using it in the next Experiment when we create a boot image for the Avnet target.

Experiment 2: 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 using a data structure called the Boot Image Format (BIF). For the most part, Linux systems accessing Programmable Logic IP require 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 Avnet Zynq Development boards can boot from Quad-SPI Flash, Ethernet or removable SD/microSD media, for the purposes of this lab we will use the SD/microSD card as the boot medium. This gives us the advantage of being able to write directly to the FAT32 partition of the SD card from a PC, and is typical for a development system.

Experiment 2 General Instruction:

Launch Xilinx Software Development Kit (SDK) and open the workspace used in Lab 4, Experiment 1. Use the SDK **Create Zynq Boot Image** tool to create the Zynq boot image file (**boot.bin**).

Experiment 2 Step-by-Step Instruction:

1. If the SDK project is not still open from the previous lab, launch it using steps 1 and 2 from Lab 4, Experiment 1.

2. With the SDK open, launch the **Create Zynq Boot Image** tool using the **Xilinx Tools→Create Boot Image** menu item.

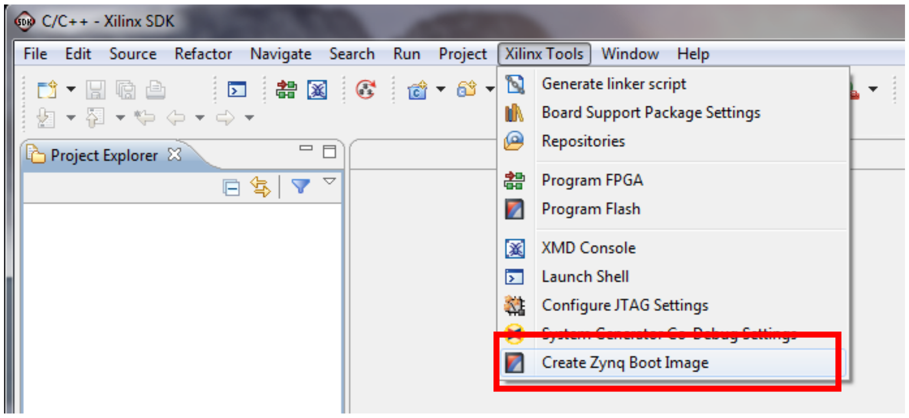
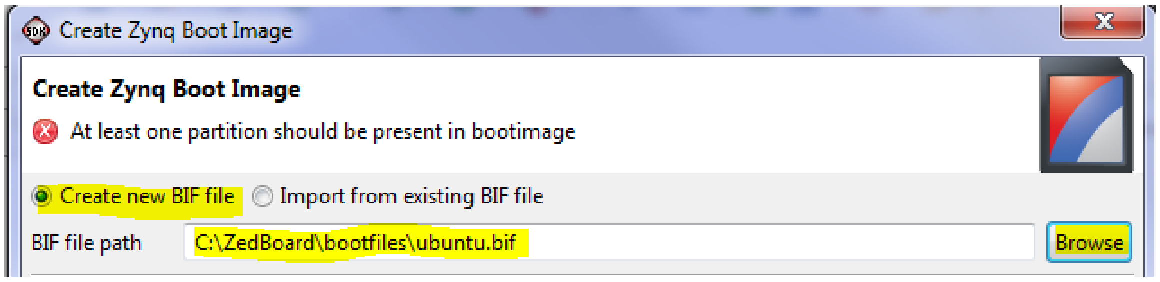


Figure 39 – Launching the Create Zynq Boot Image Tool

3. A new Boot Image Format (BIF) file must be created. Click the **Create new BIF file** radio button.

   Click the **Browse** button to select a location for the new file. A convenient place is within the **bootfiles** folder created earlier. Name the file **ubuntu.bif** to identify its purpose.

C:\<Avnet target>\bootfiles\ubuntu.bif

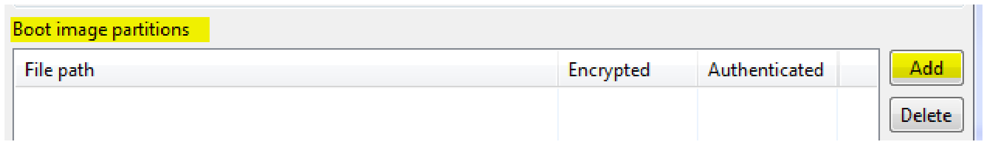


At least one partition should be present in bootimage

BIF file path C:\ZedBoard\bootfiles\ubuntu.bif

Figure 40 – Creating a New BIF File

4. The next step is to select the files that will make up our boot image. In the *Boot image partitions* section, click the **Add** button.

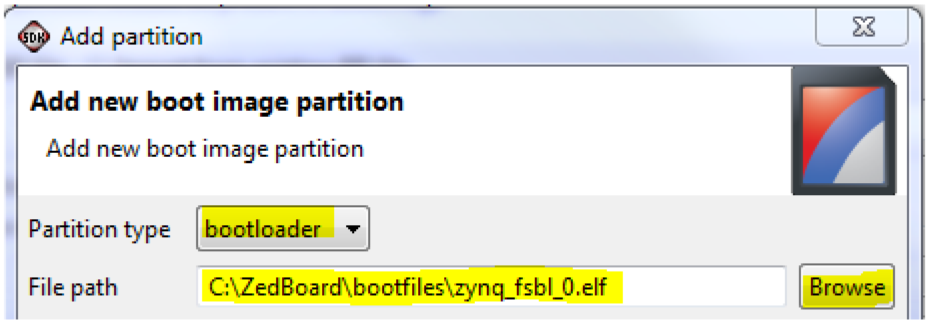

Boot image partitions

File path	Encrypted	Authenticated

Figure 41 – Add Boot Image Partitions

5. Click the **Browse** button to select the First Stage Boot Loader file from the saved location in Experiment 1. The Partition type must be set to **bootloader**.

C:\<Avnet target>\bootFiles\zynq_fsbl_0.elf


Add partition

Add new boot image partition

Add new boot image partition

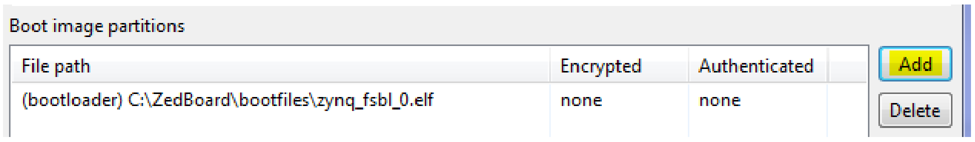
Partition type: bootloader

File path: C:\ZedBoard\bootfiles\zynq_fsbl_0.elf

Figure 42 – Add First Stage Boot Loader

Click the **OK** button to accept this file.

6. Keep in mind the importance of order for files placed into a boot image. Click the **Add** button to begin selection of the hardware bitstream.

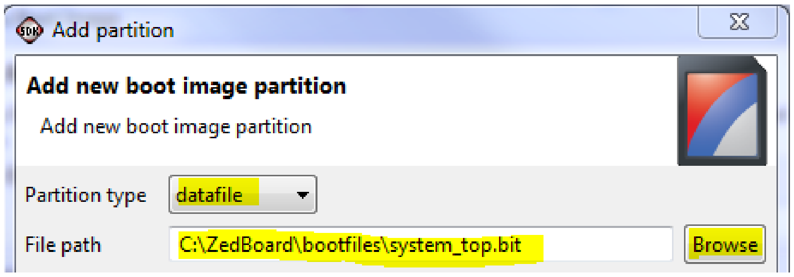

Boot image partitions

File path	Encrypted	Authenticated
(bootloader) C:\ZedBoard\bootfiles\zynq_fsbl_0.elf	none	none

Figure 43 – Add Next Boot Image Partition

7. Ensure the *Partition type* is set to **datafile**. Click the **Browse** button to select the bitstream file from the save location in Lab 1.

C:\<Avnet target>\bootFiles\system_top.bit



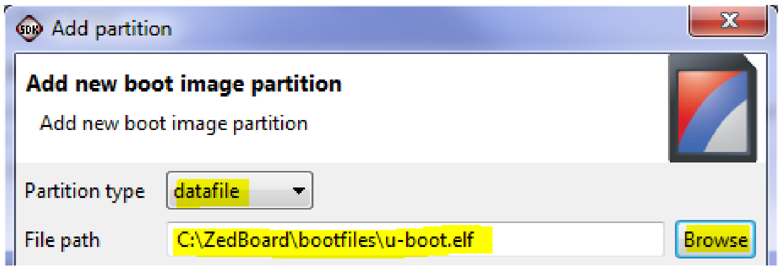
Partition type: datafile
File path: C:\ZedBoard\bootfiles\system_top.bit

Figure 44 – Add Bitstream

Click the **OK** button to accept this file.

8. Specify the application executable last (in our case it is the second stage boot loader U-boot). Click the **Add** button and again ensure the *Partition type* is set to **datafile**. Click the **Browse** button to select the ELF file from the save location in Lab 2.

C:\<Avnet target>\bootFiles\u-boot.elf



Partition type: datafile
File path: C:\ZedBoard\bootfiles\u-boot.elf

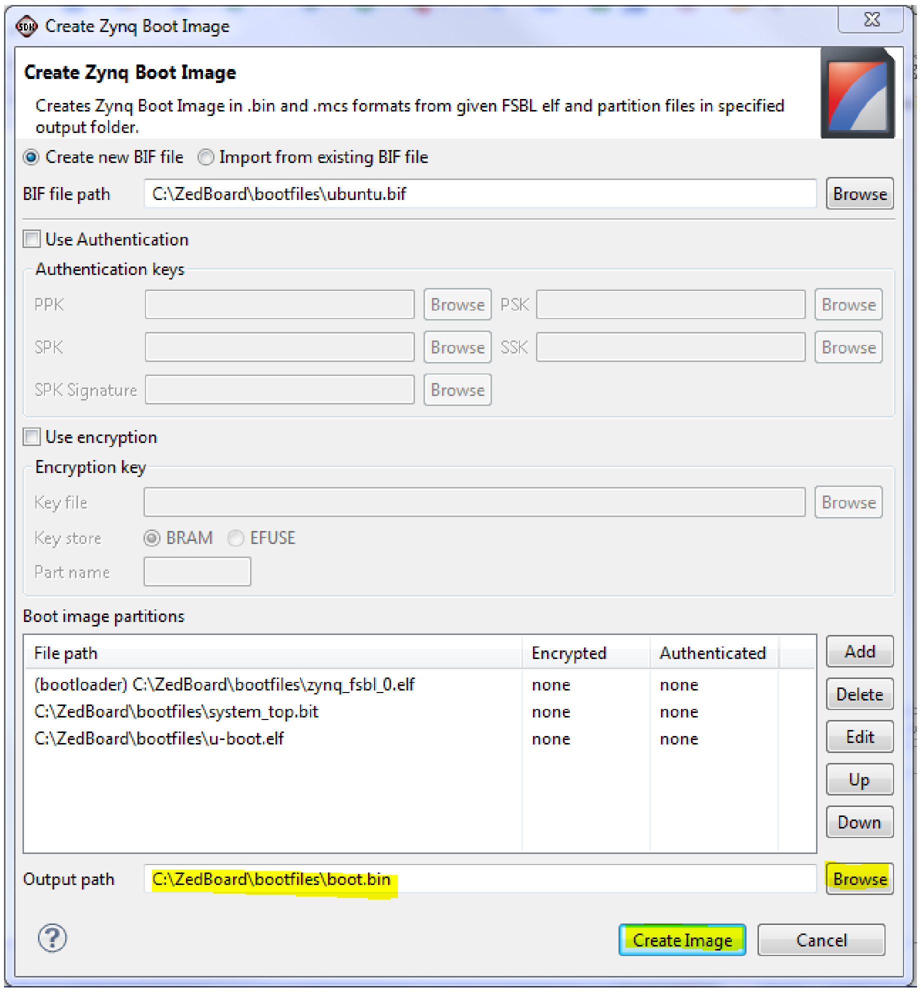
Figure 45 – Add U-Boot

Click the **OK** button to accept this file.

9. Click the **Browse** button adjacent to the *Output path* field and name the file **boot.bin**. Use the same bootFiles folder as the path.

C:\<Avnet target>\bootFiles\boot.bin

Click the **Create Image** button to launch *Bootgen*, the tool for building a boot image for the Zynq-7000 All Programmable SoC.



Creates Zynq Boot Image in .bin and .mcs formats from given FSBL elf and partition files in specified output folder.

[✓ Create new BIF file] Import from existing BIF file

BIF file path: C:\ZedBoard\bootfiles\ubuntu.bif

Use Authentication

Use encryption

Key store: [✓ BRAM] EFUSE

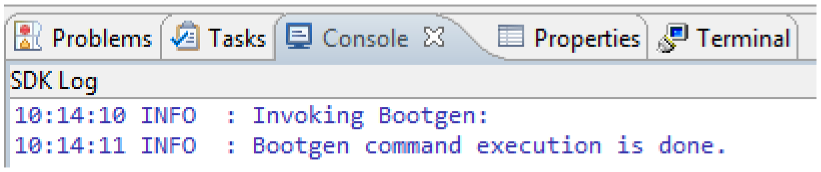
Boot image partitions

File path	Encrypted	Authenticated
(bootloader) C:\ZedBoard\bootfiles\zynq_fsbl_0.elf	none	none
C:\ZedBoard\bootfiles\system_top.bit	none	none
C:\ZedBoard\bootfiles\u-boot.elf	none	none

Output path: C:\ZedBoard\bootfiles\boot.bin

Figure 46 – Create Boot Image

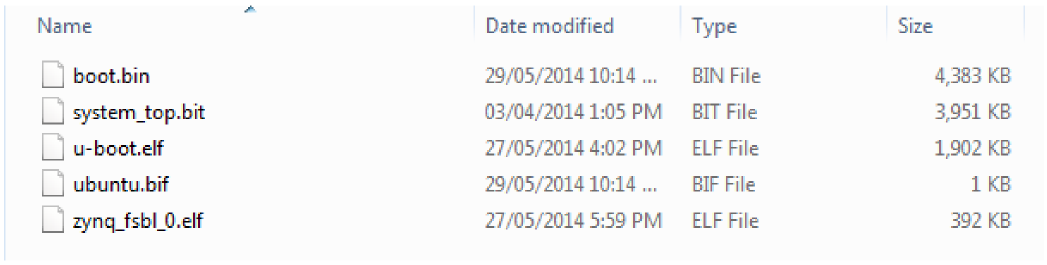
10. Bootgen merges the BIT and ELF files into a single binary boot image using the partition format specified in the Boot Image Format (BIF) file.



```
10:14:10 INFO  : Invoking Bootgen:
10:14:11 INFO  : Bootgen command execution is done.
```

Figure 47 – Bootgen Complete

Your bootfiles folder will now contain the following files:



Name	Date modified	Type	Size
boot.bin	29/05/2014 10:14 ...	BIN File	4,383 KB
system_top.bit	03/04/2014 1:05 PM	BIT File	3,951 KB
u-boot.elf	27/05/2014 4:02 PM	ELF File	1,902 KB
ubuntu.bif	29/05/2014 10:14 ...	BIF File	1 KB
zynq_fsbl_0.elf	27/05/2014 5:59 PM	ELF File	392 KB

boot.bin: Boot image file used to initialize the Zynq All Programmable SoC and execute u-boot.

system_top.bit: The bitstream used to load the Programmable Logic. This file is contained in a datafile partition in the boot image.

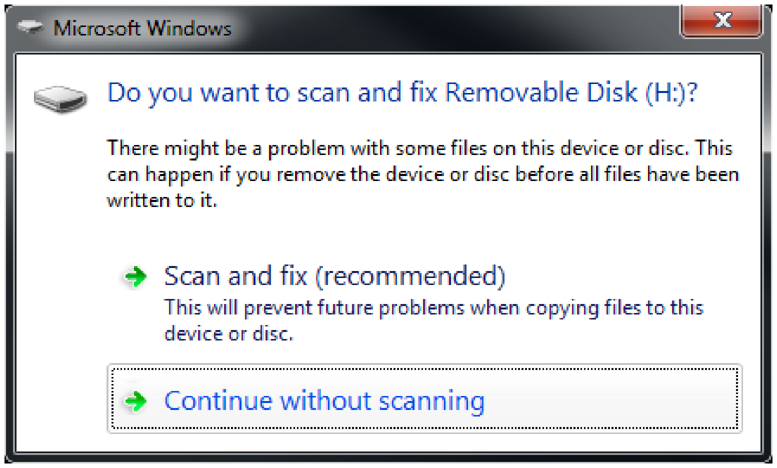
u-boot.elf: The second stage boot loader. This file is contained in a datafile partition in the boot image.

ubuntu.bif: The partition format of the boot image file. This file may be referenced in the Create Zynq Boot Image tool if you wish to recreate the boot image later.

zynq_fsbl_0.elf: The first stage boot loader that initializes the Processor System. This file is contained the bootloader partition in the boot image.

At this point you may close the Xilinx SDK.

11. Insert the SD/microSD card into the PC or card reader and wait for it to enumerate as a Windows drive. If prompted by Windows security software when inserting the card, select the **Continue without scanning** option. Select this option whenever this message appears after inserting your SD/microSD card.


Microsoft Windows

Do you want to scan and fix Removable Disk (H:)?

There might be a problem with some files on this device or disc. This can happen if you remove the device or disc before all files have been written to it.

Scan and fix (recommended)
This will prevent future problems when copying files to this device or disc.

Continue without scanning

Figure 48 – Windows Prompt to Scan and Fix Removable Media

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

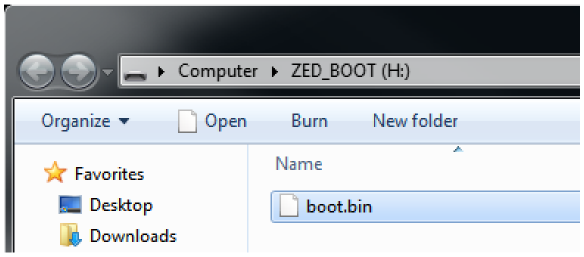


Figure 49 – The Boot Image File Copied to the SD Card

Note for SD cards: 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.

If none of these procedures work for you, try a different SD card.

13. With the **boot.bin** file in place on the SD/microSD card, you may eject the device from your Windows host and remove the card.

Experiment 3: Prepare the Avnet Target for Booting

In this experiment we will set up the hardware with all connections necessary for booting from the SD/microSD card. Select the procedure specific to the Avnet target you are using.

Experiment 3a General Instruction for ZedBoard:

Make all necessary connections to the ZedBoard in preparation for booting from the SD card.

Experiment 3a is for ZedBoard only. For the Mini-ITX, skip to Experiment 3b.

Experiment 3a Step-by-Step Instruction:

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

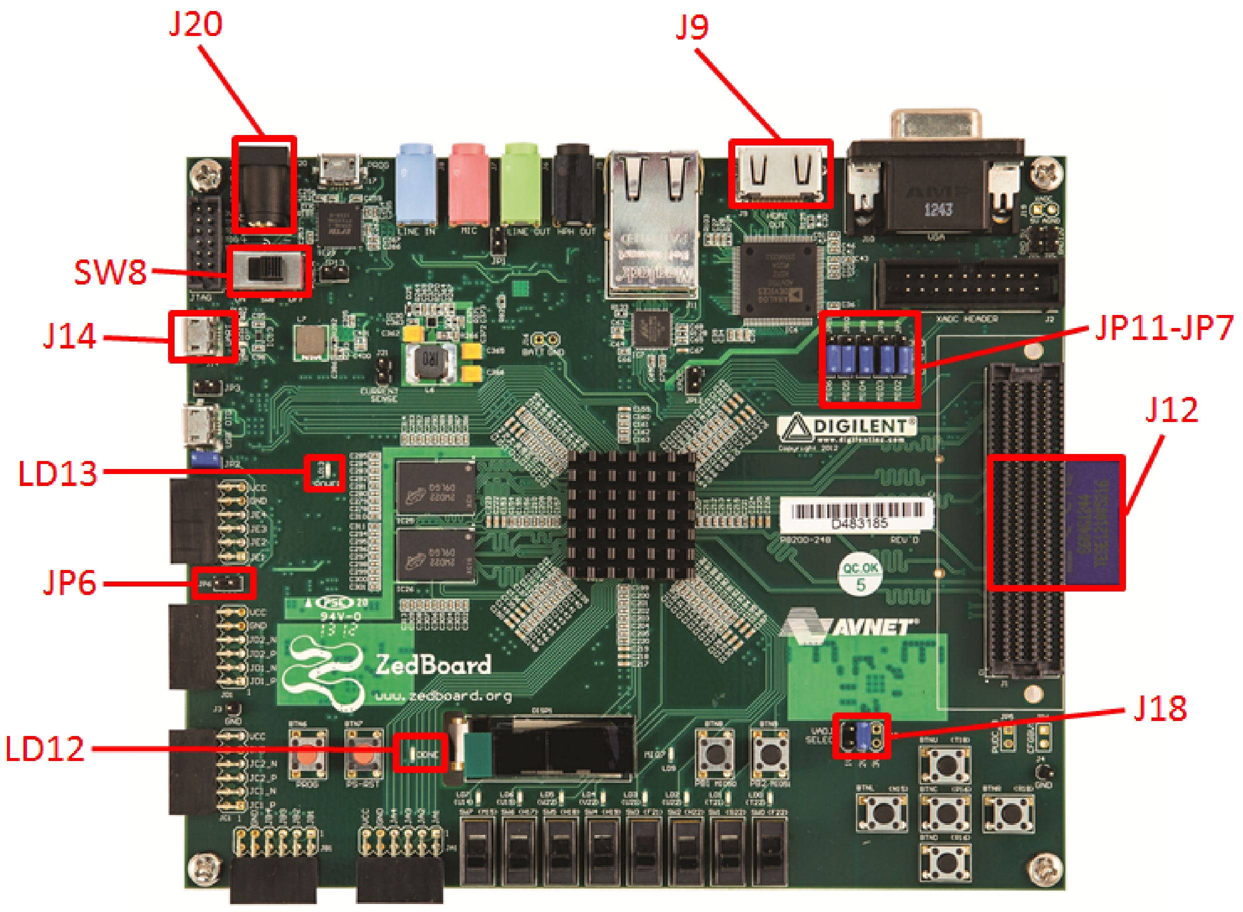


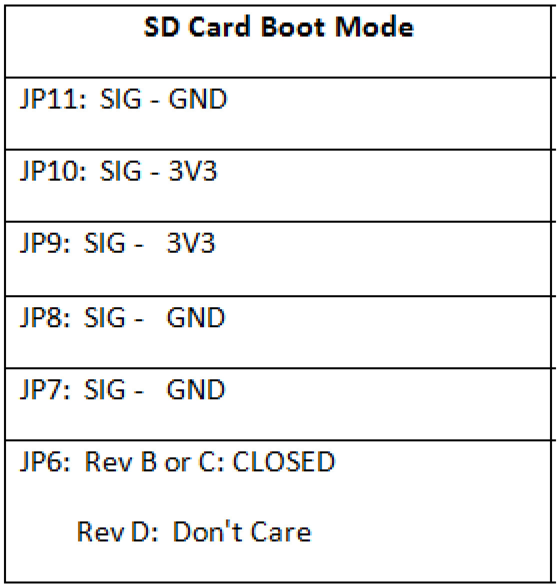
Figure 50 – ZedBoard Jumper Locations

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

3. Insert the SD card from Experiment 2 into the ZedBoard SD card slot (J12) located on the underside of the PCB.

4. Insert jumper on J18 2V5 to select 2.5V for Vadj.

5. To boot from the SD card, set the ZedBoard boot mode (JP7-JP11) and MIO0 (JP6) jumpers to SD card mode (shown in the Figure above).



SD Card Boot Mode
JP11: SIG - GND
JP10: SIG - 3V3
JP9: SIG - 3V3
JP8: SIG - GND
JP7: SIG - GND
JP6: Rev B or C: CLOSED Rev D: Don't Care

Figure 51 – SD Card Boot Mode Jumper Settings

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

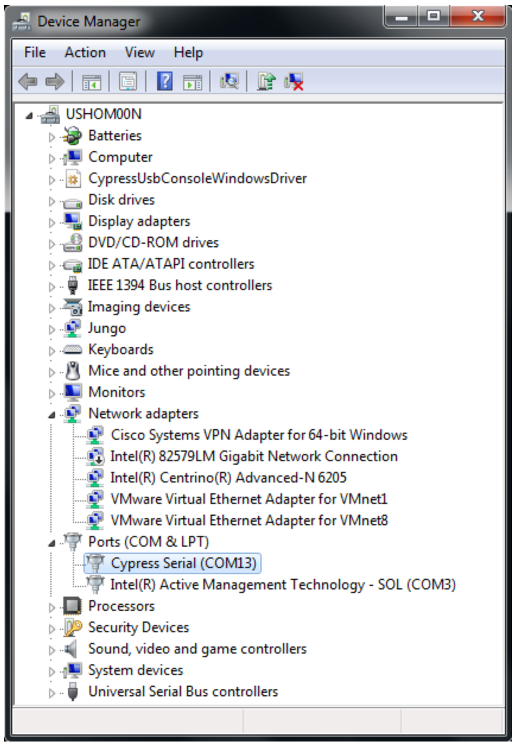
7. The PC may pop-up a dialog box asking for driver installation.

   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 *Cypress CY7C64225 USB-to-UART Setup Guide* in the link below for instructions on installing this driver. When driver installation is complete, continue to the next step.

   http://www.zedboard.org/documentation/1521

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

9. 

```
USHOM00N
  Batteries
  Computer
  CypressUsbConsoleWindowsDriver
  Disk drives
  Display adapters
  DVD/CD-ROM drives
  IDE ATA/ATAPI controllers
  IEEE 1394 Bus host controllers
  Imaging devices
  Jungo
  Keyboards
  Mice and other pointing devices
  Monitors
  Network adapters
    Cisco Systems VPN Adapter for 64-bit Windows
    Intel(R) 82579LM Gigabit Network Connection
    Intel(R) Centrino(R) Advanced-N 6205
    VMware Virtual Ethernet Adapter for VMnet1
    VMware Virtual Ethernet Adapter for VMnet8
  Ports (COM & LPT)
    Cypress Serial (COM13)
    Intel(R) Active Management Technology - SOL (COM3)
  Processors
  Security Devices
  Sound, video and game controllers
  System devices
  Universal Serial Bus controllers
```

Figure 52 – Device Manager Showing Enumerated USB-UART

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

10. Skip to Experiment 4.

Experiment 3b General Instruction for Mini-ITX:

Make all necessary connections to the Zynq Mini-ITX in preparation for booting from the microSD card.

Experiment 3b is for the Mini-ITX only. For ZedBoard, refer to Experiment 3a.

Experiment 3b Step-by-Step Instruction:

1. Connect the ATX power supply to the Mini-ITX, as shown in the figure below. The male four-connector cable from the power supply should be inserted first, followed by the larger connector. Apply downward pressure to the large connector until it snaps into place, securing the smaller connector when it does so.

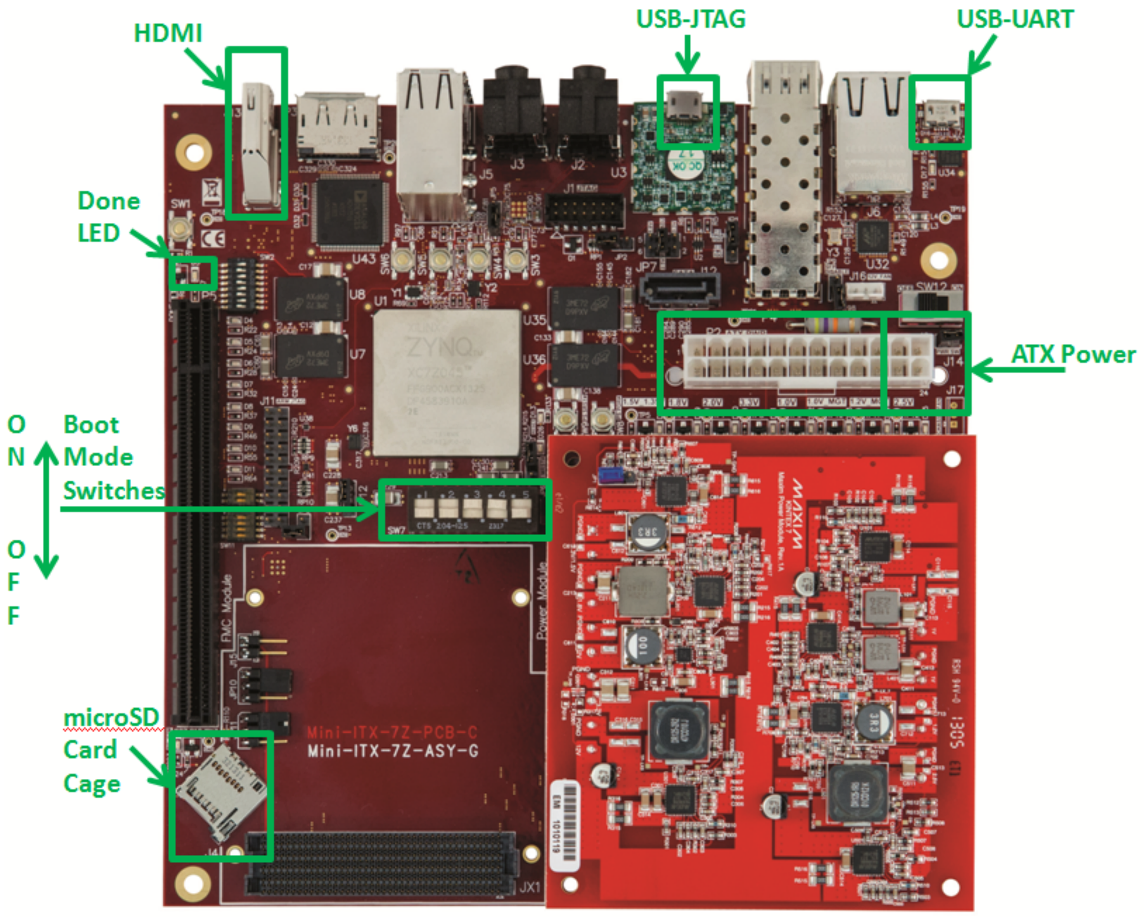


Figure 53 – Zynq Mini-ITX Board Components

2. Connect the USB-UART port of the Zynq Mini-ITX (J7), labeled UART, to a PC using a USB-to-microUSB cable.

3. Connect an HDMI cable between the HDMI monitor and the J13 connector on the Zynq Mini-ITX.

4. Verify the Mini-ITX boot mode switch block (SW7), labeled PS BOOT, are set to JTAG Boot mode as shown in the figure below. All Switches must be in the OFF position (away from the numbers).

microSD Boot Mode	**JTAG Boot Mode**
1: OFF	1: OFF
2: OFF	2: OFF
3: ON	3: OFF
4: ON	4: OFF
5: OFF	5: OFF

Figure 54 – Mode Switch Block (SW7) Settings

5. Turn power switch (SW12) to the ON position. The Zynq Mini-ITX will power on and the cooling fan on the Zynq device should start running. A row of 8 green LEDs next to the power module and a row of 8 red LEDs next to the PCIe connector will illuminate.

6. The PC may pop-up a dialog box asking for driver installation. The Zynq Mini-ITX has a USB-UART bridge based on the Silicon Labs CP210x 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 *Silicon Labs CP210x USB-to-UART Setup Guide* in the link below for instructions on installing this driver. When driver installation is complete, continue to the next step.

   http://www.zedboard.org/documentation/2056

   When driver installation is complete, continue to the next step.

7. Use the Windows Device Manager to determine the COM Port.

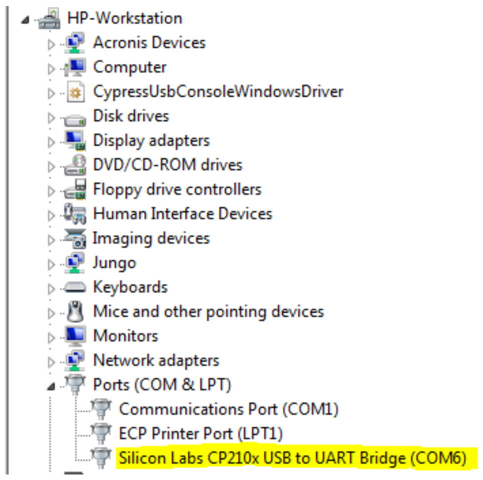

HP-Workstation
Acronis Devices
Computer
CypressUsbConsoleWindowsDriver
Disk drives
Display adapters
DVD/CD-ROM drives
Floppy drive controllers
Human Interface Devices
Imaging devices
Jungo
Keyboards
Mice and other pointing devices
Monitors
Network adapters
Ports (COM & LPT)
Communications Port (COM1)
ECP Printer Port (LPT1)
Silicon Labs CP210x USB to UART Bridge (COM6)

Figure 55 – Device Manager Showing Enumerated USB-UART

Note: Each unique USB-UART device attached will enumerate under the next available COM port. In this example, the Silicon Labs CP210x USB-to-UART device is enumerated as COM6.

Experiment 4: Boot the Avnet Target with Boot Image

The Avnet Zynq target can now be booted using the Boot Image that was copied to the SD/microSD card in the previous exercise.

Experiment 4 General Instruction:

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

Experiment 4 Step-by-Step Instruction:

1. 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

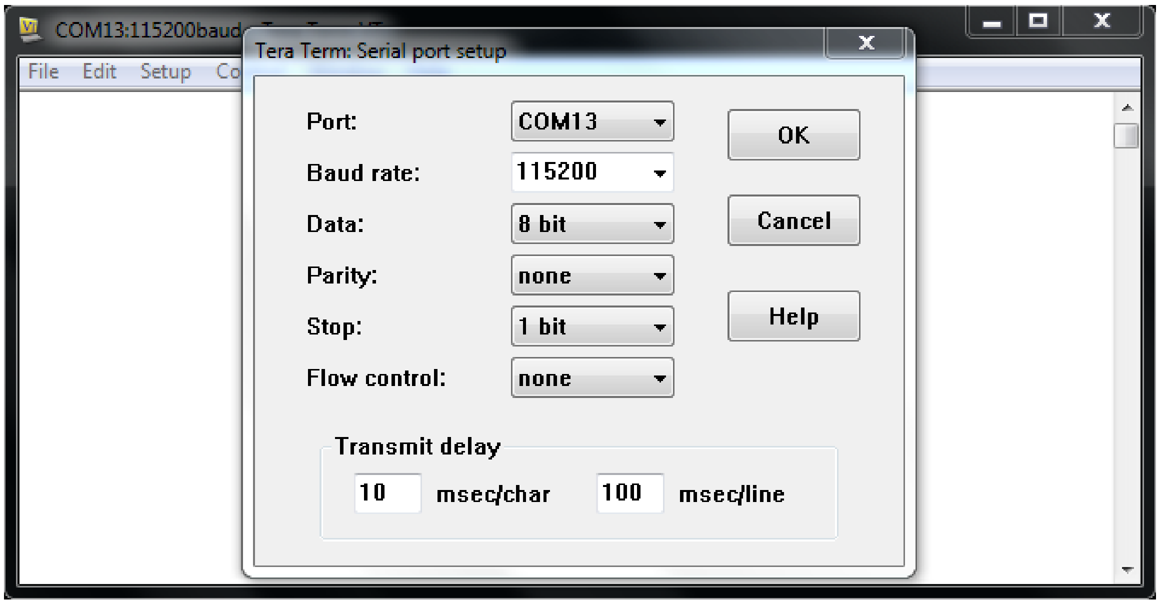
2. Once Tera Term is installed, Tera Term can be accessed from the desktop or Start menu shortcuts.

Figure 56 – Tera Term Icon

3. To configure baud rate settings, open the Serial Port Setup window from the **Setup→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 where many lines of text will be sent to the console in rapid succession.



Port: COM13
Baud rate: 115200
Data: 8 bit
Parity: none
Stop: 1 bit
Flow control: none
Transmit delay
10 msec/char 100 msec/line

Figure 57 – Tera Term Serial Port Setup Page

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

5. Power cycle the Avnet target 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** and stop at the U-boot command prompt.

   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, but is unable to locate it. At this point you will see the error:

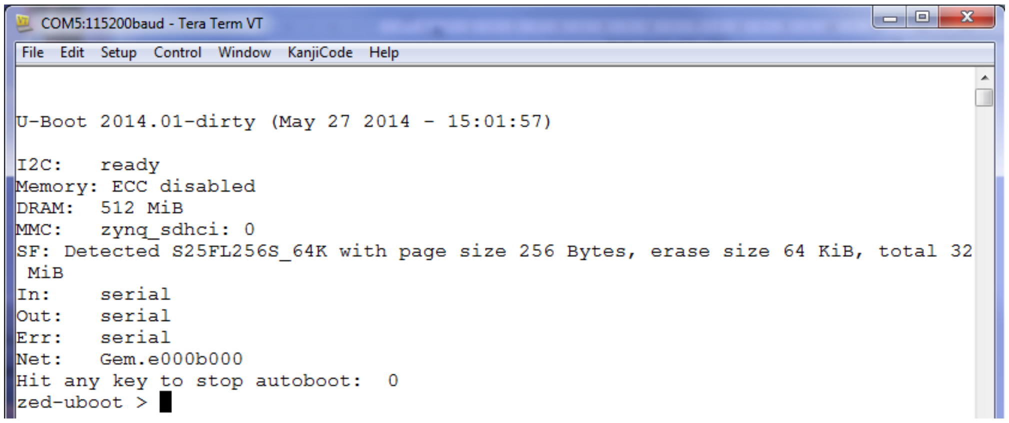
   **** Unable to read file uImage ****

   U-boot will return to its command prompt.

   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.

   On some versions of ZedBoard, you may need to disconnect the serial console and wait for the DONE LED to illuminate before you can reconnect.

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.


COM5:115200baud - Tera Term VT
File Edit Setup Control Window KanjiCode Help

```
U-Boot 2014.01-dirty (May 27 2014 - 15:01:57)

I2C:   ready
Memory: ECC disabled
DRAM:  512 MiB
MMC:   zynq_sdhci: 0
SF: Detected S25FL256S_64K with page size 256 Bytes, erase size 64 KiB, total 32
 MiB
In:    serial
Out:   serial
Err:   serial
Net:   Gem.e000b000
Hit any key to stop autoboot:  0
zed-uboot >
```

Figure 58 – Zynq U-boot Prompt

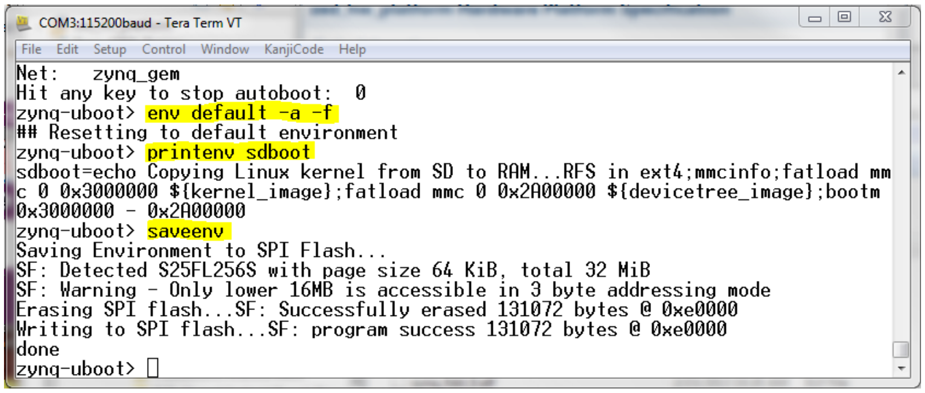
Note: If you have run U-boot on the Avnet target previously, and you saved the environment variables using the **saveenv** command, you may need to update the non-volatile memory at this time so the new **sdboot** command will be used. When we changed the U-boot source code, we updated the default environment,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

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


COM3:115200baud - Tera Term VT
File Edit Setup Control Window KanjiCode Help

```
Net:   zynq_gem
Hit any key to stop autoboot:  0
zynq-uboot> env default -a -f
## Resetting to default environment
zynq-uboot> printenv sdboot
sdboot=echo Copying Linux kernel from SD to RAM...RFS in ext4;mmcinfo;fatload mm
c 0 0x3000000 ${kernel_image};fatload mmc 0 0x2A00000 ${devicetree_image};bootm
0x3000000 - 0x2A00000
zynq-uboot> saveenv
Saving Environment to SPI Flash...
SF: Detected S25FL256S with page size 64 KiB, total 32 MiB
SF: Warning - Only lower 16MB is accessible in 3 byte addressing mode
Erasing SPI flash...SF: Successfully erased 131072 bytes @ 0xe0000
Writing to SPI flash...SF: program success 131072 bytes @ 0xe0000
done
zynq-uboot>
```

Lab 5 – Configure and Build the Linux Kernel

Lab Overview

Analog Devices (ADI) maintains a freely downloadable Linux kernel that has been tested on Xilinx Zynq-7000 Programmable SoC development boards, and is suitable for hosting the Ubuntu Desktop. 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 learn to do the following:

- Retrieve Linux kernel source code from the GitHub repository
- Configure the kernel for the Avnet target
- Build the kernel for the Avnet 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 \

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

$ cd ubuntu

$ git checkout xcomm_zynq
```

Experiment 1 Step-by-Step Instruction:

1. If the virtual machine is not already open, launch the VMware Player application by following Steps 1 through 4 in Lab 2, Experiment 1.

2. The Linux 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 commands to clone the repository. Make sure you use the **https:** directive instead of the old **git:** string. Failure to do this will cause the process to hang without any indication as to the cause.

```
$ cd ~

$ git clone \

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

3. Wait until the clone operation completes, this could take an hour or more depending upon your connection speed.

   The clone command sets up a few convenient 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**.

4. Change into the top level source folder.

```
$ cd ubuntu
```

5. Check out the code branch compatible with the FPGA platform[7]. 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, including extensions for the Analog Devices ADV7511 HDMI transmitter and ADAU1761 Audio Codec, 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 zynq_xcomm_adv7511_defconfig

$ make ARCH=arm uImage LOADADDR=0x8000
```

Experiment 2 Step-by-Step Instruction:

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
```

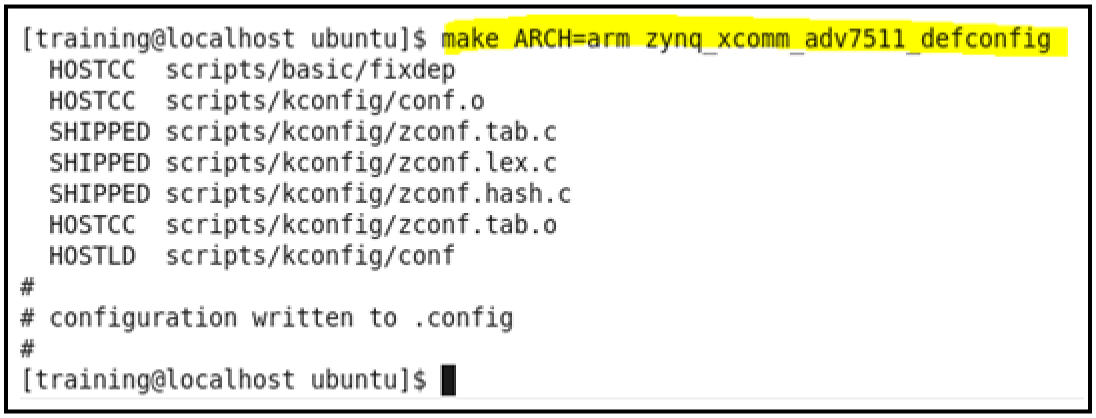
3. A Zynq configuration file is included for Zynq targets using the ADI adv7511 HDMI video transmitter on the board. The file is located at:

/arch/arm/configs/zynq_xcomm_adv7511_defconfig

[7] The kernel branch referenced here is compatible with Xilinx tools release 14.2 and higher.

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.

```
$ make ARCH=arm zynq_xcomm_adv7511_defconfig
```



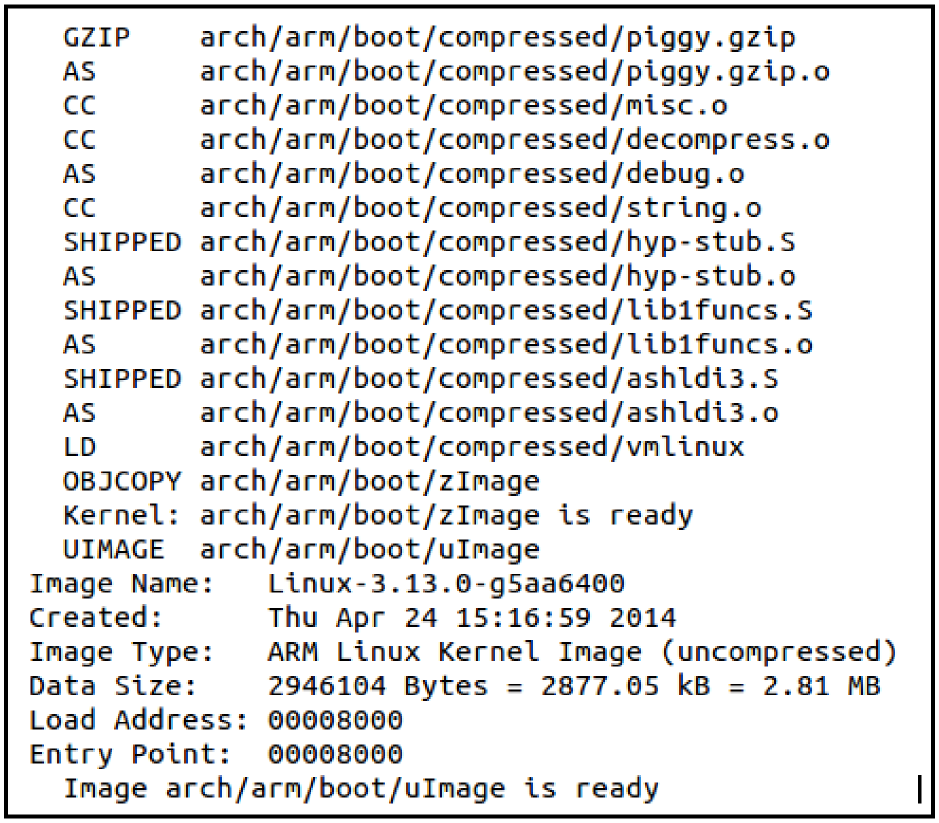
```
[training@localhost ubuntu]$ make ARCH=arm zynq_xcomm_adv7511_defconfig
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  SHIPPED scripts/kconfig/zconf.tab.c
  SHIPPED scripts/kconfig/zconf.lex.c
  SHIPPED scripts/kconfig/zconf.hash.c
  HOSTCC  scripts/kconfig/zconf.tab.o
  HOSTLD  scripts/kconfig/conf
#
# configuration written to .config
#
[training@localhost ubuntu]$ 
```

Figure 59 – Kernel Configuration

4. Build the kernel source and generate the compressed image file expected by our version of U-boot with the make command.

   The build process should take about 15 to 25 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 uImage LOADADDR=0x8000
```



```
  GZIP    arch/arm/boot/compressed/piggy.gzip
  AS      arch/arm/boot/compressed/piggy.gzip.o
  CC      arch/arm/boot/compressed/misc.o
  CC      arch/arm/boot/compressed/decompress.o
  AS      arch/arm/boot/compressed/debug.o
  CC      arch/arm/boot/compressed/string.o
  SHIPPED arch/arm/boot/compressed/hyp-stub.S
  AS      arch/arm/boot/compressed/hyp-stub.o
  SHIPPED arch/arm/boot/compressed/lib1funcs.S
  AS      arch/arm/boot/compressed/lib1funcs.o
  SHIPPED arch/arm/boot/compressed/ashldi3.S
  AS      arch/arm/boot/compressed/ashldi3.o
  LD      arch/arm/boot/compressed/vmlinux
  OBJCOPY arch/arm/boot/zImage
  Kernel: arch/arm/boot/zImage is ready
  UIMAGE  arch/arm/boot/uImage
Image Name:   Linux-3.13.0-g5aa6400
Created:      Thu Apr 24 15:16:59 2014
Image Type:   ARM Linux Kernel Image (uncompressed)
Data Size:    2946104 Bytes = 2877.05 kB = 2.81 MB
Load Address: 00008000
Entry Point:  00008000
  Image arch/arm/boot/uImage is ready
```

Figure 60 – Linux Kernel Build Completed

If your build fails to locate the *mkimage* tool, please see the final step in *Lab 2, Experiment 1* to correct the problem. You may rebuild the entire kernel again, or you can manually create the image with the following commands:

```
cd arch/arm/boot
gzip -9 zImage
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→System Tools→File Browser** menu item.

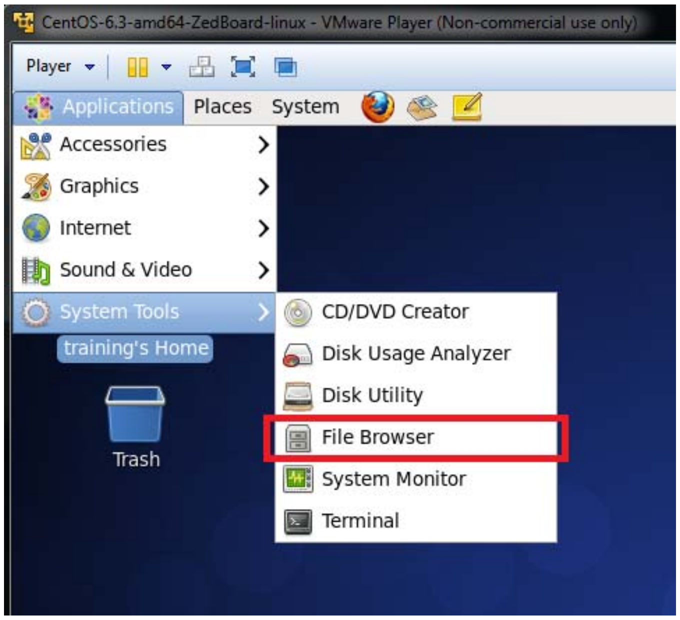


Figure 61 – CentOS File Browser

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

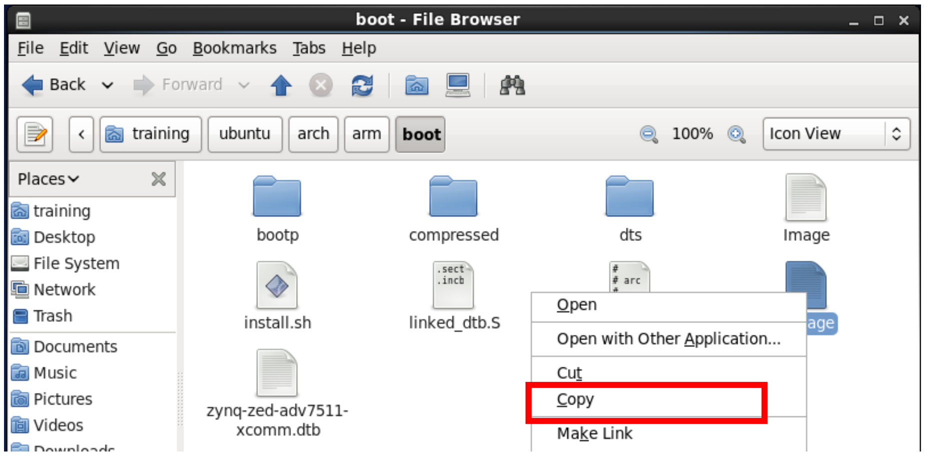
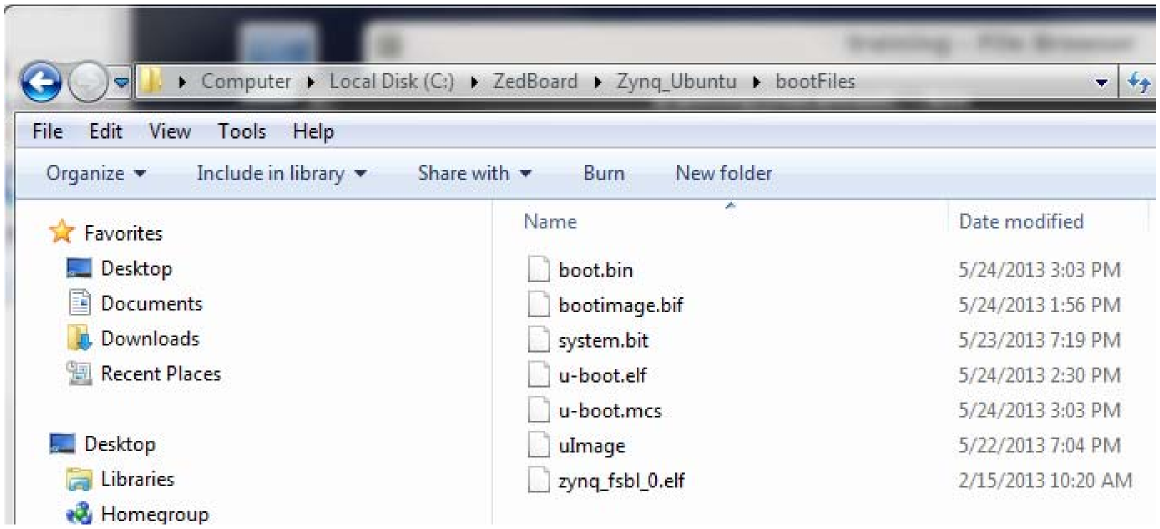


Figure 62 – Copying Kernel uImage to Virtual Machine Clipboard

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:\<Avnet target>\bootFiles


Computer ▸ Local Disk (C:) ▸ ZedBoard ▸ Zynq_Ubuntu ▸ bootFiles
File Edit View Tools Help
Organize Include in library Share with Burn New folder
Favorites
Desktop
Documents
Downloads
Recent Places
Desktop
Libraries
Homegroup

Name	Date modified
boot.bin	5/24/2013 3:03 PM
bootimage.bif	5/24/2013 1:56 PM
system.bit	5/23/2013 7:19 PM
u-boot.elf	5/24/2013 2:30 PM
u-boot.mcs	5/24/2013 3:03 PM
uImage	5/22/2013 7:04 PM
zynq_fsbl_0.elf	2/15/2013 10:20 AM

Figure 63 – 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.

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. Although we have already initialized and booted the ARM processor cores into a standalone running state, Linux has no knowledge of this and by default assumes that it must perform all initialization on its own. To do this it needs to set up virtual memory, print to the console, and locate all of the installed hardware in the system and load software drivers.

These operations are carried out by writing to registers, but the Linux kernel needs a method to discover the parameters associated with the current hardware system. In a fixed system this could be handled with static header files or kernel configuration at build time, but for many dynamic devices it is much more desirable to obtain the configuration information at run time. This is further complicated by the endless customization available on an FPGA, which would quickly result in an unmanageable number of possible kernel header and configuration combinations.

On a PC, the device information is supplied by the BIOS, but the ARM processors don't have anything comparable.

So the chosen solution is a **device tree**, also referred to as **Open Firmware** (abbreviated **OF**) or **Flattened Device Tree** (**FDT**). This is a text file which contains all the hardware information about the system, such as device addresses, interrupt vectors and bus addresses, compiled into byte code format. U-boot copies the compiled data into a known address in the RAM before jumping to the kernel's entry point.

The top level device tree source files for the Avnet Ubuntu system targets are located in the Git kernel repository at:

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

Mini-ITX: ~/ubuntu/arch/arm/boot/dts/zynq-mitx-adv7511.dts[8]

These top level files contain the details for the specific FPGA platforms used in the design, and would be the files to change if IP were added, removed or altered in the system. There are two further levels of device tree files for the Ubuntu

[8] If this file is not included in the Linux kernel source, you may copy it and the companion zynq-mitx.dtsi file from the Supplied files.

implementation, which together define the entire system, all the way down to the ARM processor cores. Information is structured so the least volatile entries are made at the lowest level, and are therefore common to all Zynq systems and should require no changes for the dual core Zynq devices. The middle layer is used to provide a convenient place for board specific parameters, such as the board name, available memory and Ethernet PHY configuration.

The middle layer device tree files are located at:

ZedBoard: ~/ubuntu/arch/arm/boot/dts/zynq-zed.dtsi

Mini-ITX: ~/ubuntu/arch/arm/boot/dts/zynq-mitx.dtsi

The Zynq common layer device tree file is found at:

~/ubuntu/arch/arm/boot/dts/zynq.dtsi

When you have completed this lab, you will have learned to:

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

Experiment 1: Compile the Device Tree Source Files

This experiment shows how to compile the device tree source files to a single binary output product 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    (For ZedBoard)

                                       or

$ make ARCH=arm zynq-mitx-adv7511.dtb   (For Mini-ITX)
```

Rename the result file to **devicetree.dtb** and copy it from the output directory to the boot file directory on the host machine.

Experiment 1 Step-by-Step Instruction:

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 top level source file (the only difference is the .dtb versus .dts endings).

```
$ cd ~/ubuntu

$ make ARCH=arm zynq-zed-adv7511.dtb          (ZedBoard)

                                   or

$ make ARCH-arm zynq-mitx-adv7511.dtb         (Mini-ITX)
```

   The binary *(.dtb)* is created in the same directory as the source *(.dts)* file.

   ZedBoard Output:
   DTC arch/arm/boot/dts/zynq-zed-adv7511.dtb

   Mini-ITX Output:
   DTC arch/arm/boot/dts/zynq-mitx-adv7511.dtb

2. Rename the compiled device tree to the name expected by U-boot when it copies the file to RAM at load time.

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

$ mv zynq-zed-adv7511.dtb devicetree.dtb (ZedBoard)

                                   or

$ mv zynq-mitx-adv7511.dtb devicetree.dtb (Mini-ITX)
```

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

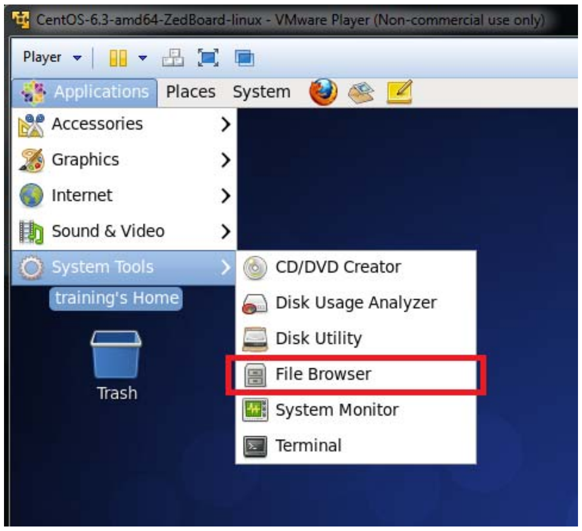


Figure 64 – CentOS File Browser

4. Locate the compiled device tree file at:

~/ubuntu/arch/arm/boot/dts/devicetree.dtb

Right click on this file and select the **Copy** option to place it in the Virtual Machine clipboard.

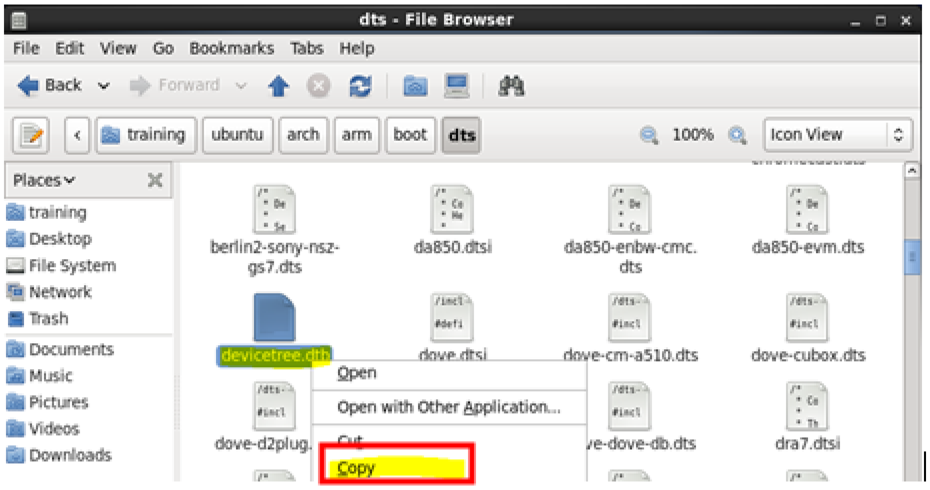
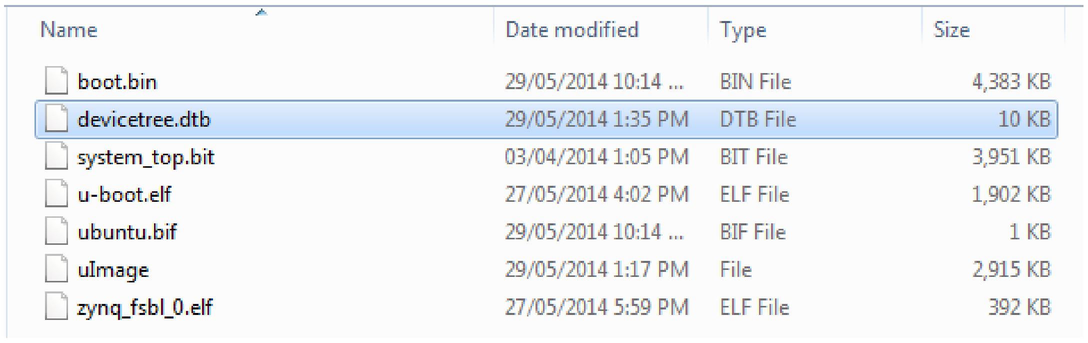


Figure 65 – Copying Kernel Image to Virtual Machine Clipboard

5. Paste the *devicetree.dtb* file into the **bootfiles** folder under the host operating system by using Windows Explorer to navigate to the following folder:

C:\ZedBoard\Zynq_Ubuntu\bootFiles



Name	Date modified	Type	Size
boot.bin	29/05/2014 10:14 ...	BIN File	4,383 KB
devicetree.dtb	29/05/2014 1:35 PM	DTB File	10 KB
system_top.bit	03/04/2014 1:05 PM	BIT File	3,951 KB
u-boot.elf	27/05/2014 4:02 PM	ELF File	1,902 KB
ubuntu.bif	29/05/2014 10:14 ...	BIF File	1 KB
uImage	29/05/2014 1:17 PM	File	2,915 KB
zynq_fsbl_0.elf	27/05/2014 5:59 PM	ELF File	392 KB

Figure 66 – Compiled Device Tree Copied to the Host Machine

6. Insert the SD card into the PC or SD card reader and wait for it to enumerate as a Windows drive.

7. Copy the **devicetree.dtb** file from the **bootfiles** folder to the top level of the SD/microSD card, overwriting any existing versions of the file.

8. Eject the removable drive and remove the SD/microSD 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/12.11/ubuntu/precise-
images/ubuntu-desktop/linaro-precise-ubuntu-desktop-
20121124-560.tar.gz

$ sudo tar --strip-components=3 –C /media/rootfs -xzpf \
  linaro-precise-ubuntu-desktop-20121124-560.tar.gz \
  binary/boot/filesystem.dir
```

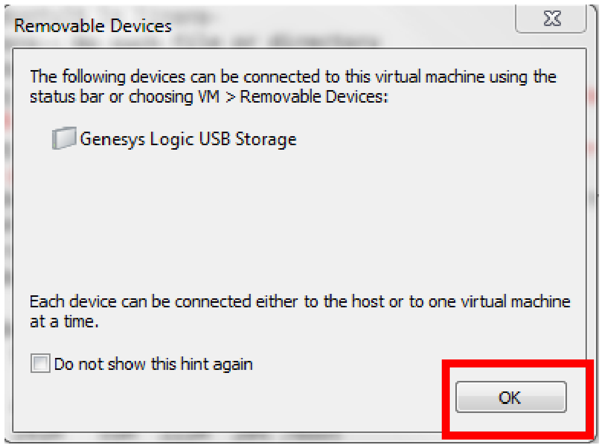
Experiment 2 Step-by-Step Instruction:

1. Download a prebuilt root file system image from Linaro to your local user top-level directory. The suggested version (known working) is at the URL below:

```
$ cd ~

$ wget
http://releases.linaro.org/12.11/ubuntu/precise-
images/ubuntu-desktop/linaro-precise-ubuntu-desktop-
20121124-560.tar.gz
```

Depending on your connection speed, this download will average from 10 to 30 minutes to complete.

2. With the VM running, insert the pre-formatted SD card created in Lab 3 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.



Removable Devices

The following devices can be connected to this virtual machine using the status bar or choosing VM > Removable Devices:

Genesys Logic USB Storage

Each device can be connected either to the host or to one virtual machine at a time.

Do not show this hint again

OK

Figure 67 – 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.

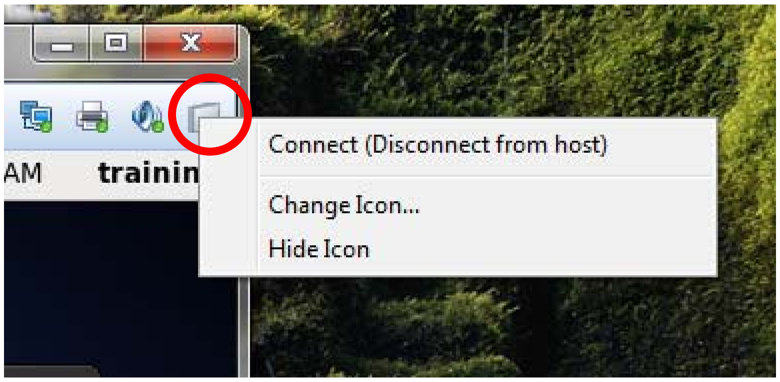
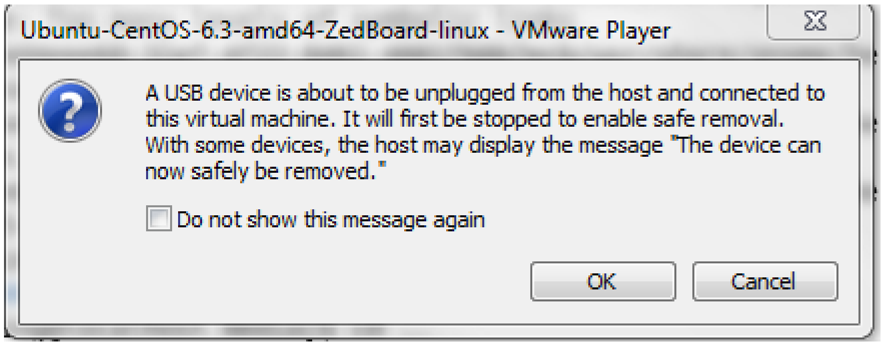


Figure 68 – Connect USB Device to Virtual Machine

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


Ubuntu-CentOS-6.3-amd64-ZedBoard-linux - VMware Player

A USB device is about to be unplugged from the host and connected to this virtual machine. It will first be stopped to enable safe removal. With some devices, the host may display the message "The device can now safely be removed."

Do not show this message again

Figure 69 – 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-precise-ubuntu-desktop-20121124-560.tar.gz
binary/boot/filesystem.dir
```

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/microSD card configured with 2 partitions. The first partition is in FAT32 format, accessible from a Windows or Linux host, and it contains all the files required to initialize 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.

The root file system contains the Ubuntu desktop environment. This higher level software uses a common Linux kernel, and can therefore coexist on the same processor system with other higher level packages, 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 mobile phone 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 have learned to:

- Configure the Avnet target to boot from an SD/microSD card
- Connect all the external devices necessary to allow the Avnet target to function as a desktop computer.
- 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 codec

Experiment 1: Boot the Avnet Target 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 (ZedBoard only)
- USB Keyboard
- USB Mouse
- HDMI cable, with HDMI to DVI adapter if a DVI monitor is used

Experiment 1 General Instruction:

Configure the target for booting from the SD card. Connect the external peripherals necessary 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 Instruction:

1. Configure the Avnet target for booting from the SD/microSD card, as described in **Lab 4 - Create the First Stage Boot Loader**.

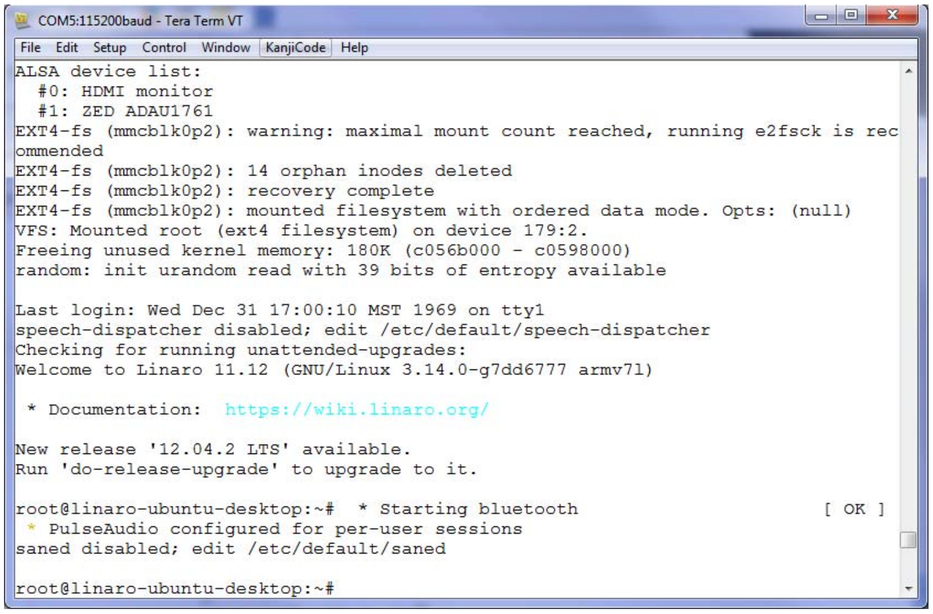
2. Connect the HDMI/DVI monitor to the HDMI output on the target using the HDMI cable, and adapter if required.

3. ZedBoard only: Connect the USB Hub to the USB-OTG port on the ZedBoard using a Micro-USB to USB Adapter cable included in the ZedBoard Kit.

4. Connect the USB keyboard and USB mouse to open ports on the USB Hub (ZedBoard), or to two of the USB ports in the USB stack on the Zynq Mini-ITX board.

5. 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 (or equivalent) as the boot console.

6. Apply power to the target, and wait for the blue configuration LED to illuminate. Connect Tera Term to the USB-UART COM port. You will see U-boot load the images from the SD/microSD 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.


COM5:115200baud - Tera Term VT
File Edit Setup Control Window KanjiCode Help

```
ALSA device list:
  #0: HDMI monitor
  #1: ZED ADAU1761
EXT4-fs (mmcblk0p2): warning: maximal mount count reached, running e2fsck is rec
ommended
EXT4-fs (mmcblk0p2): 14 orphan inodes deleted
EXT4-fs (mmcblk0p2): recovery complete
EXT4-fs (mmcblk0p2): mounted filesystem with ordered data mode. Opts: (null)
VFS: Mounted root (ext4 filesystem) on device 179:2.
Freeing unused kernel memory: 180K (c056b000 - c0598000)
random: init urandom read with 39 bits of entropy available

Last login: Wed Dec 31 17:00:10 MST 1969 on tty1
speech-dispatcher disabled; edit /etc/default/speech-dispatcher
Checking for running unattended-upgrades:
Welcome to Linaro 11.12 (GNU/Linux 3.14.0-g7dd6777 armv7l)

 * Documentation:  https://wiki.linaro.org/

New release '12.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

root@linaro-ubuntu-desktop:~#  * Starting bluetooth                       [ OK ]
 * PulseAudio configured for per-user sessions
saned disabled; edit /etc/default/saned

root@linaro-ubuntu-desktop:~#
```

Figure 70 – Linaro Console Boot

7. 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. Note that the screen resolution may not be optimal – this is addressed in the next section.

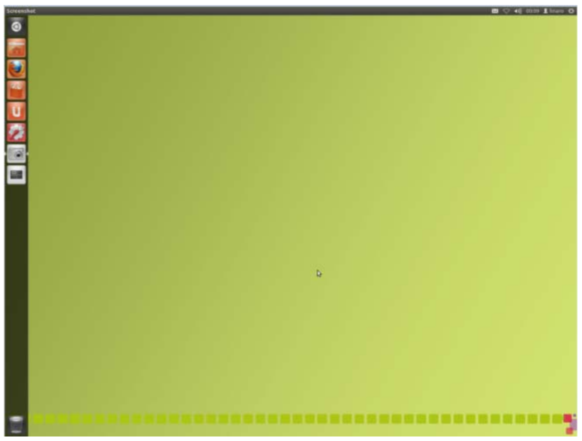


Figure 71 – Ubuntu Default Desktop

8. 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 are required to get a basic working system, but they improve the overall video and audio experience quite a bit.

Set the System Clock

The video configuration in the following section may fail if the system clock is not current, as new files will be created with a timestamp that is older than the files they will replace. To set the system clock, click on the time displayed in the upper right of the monitor.

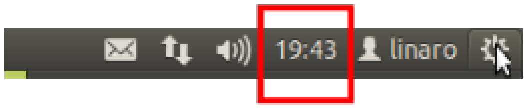


Figure 72 – Ubuntu Desktop Clock

A calendar will be displayed. At the bottom, click on the **Time and Date Settings**.

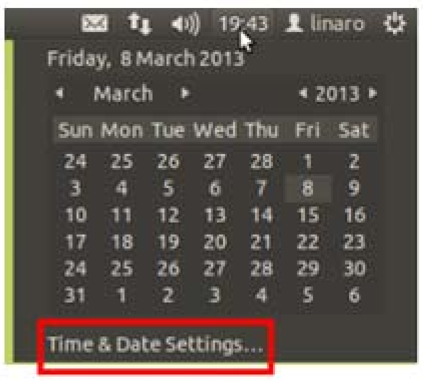
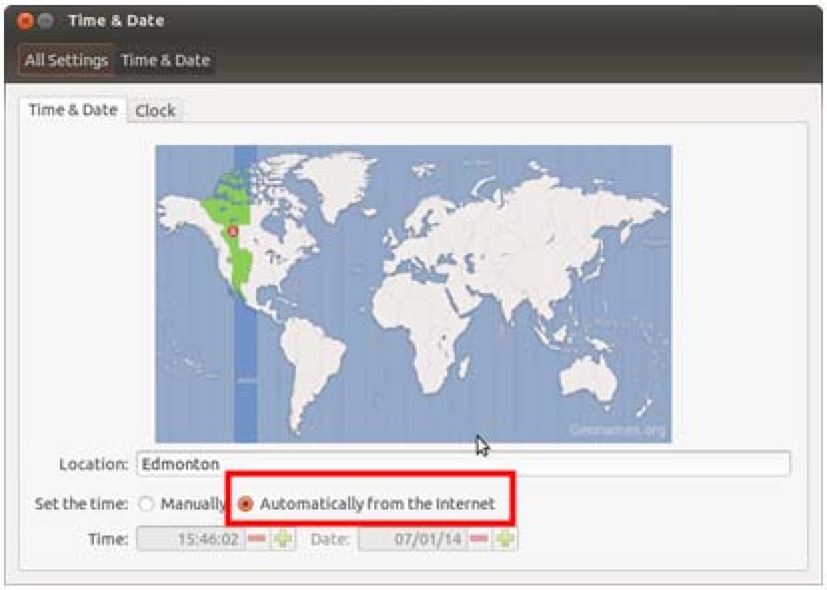


Figure 73 – Ubuntu Desktop Time and Date Settings

Use the *Time & Date* panel to set the system time. If your target has an Internet connection, instead of setting the time manually you may select the radio button to set the time *Automatically from the internet*.



Location: Edmonton
Set the time: ○ Manually ● Automatically from the Internet
Time: 15:46:02 Date: 07/01/14

Figure 74 – Set Ubuntu Current Time

Once the time is current, close the *Time & Date* panel.

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 is 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
```

If you would like to use the Ubuntu console, you will find the root user is disabled by default in Ubuntu. To enable the root user, enter:

sudo passwd

- Enter new root password
- Confirm new root password

To become the root user, enter **su** and the root password at the prompt. Type **exit** at the prompt to return to the linaro user.

Once this file is created[9], 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:

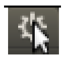


Figure 75 – Click on the System Operations Icon

[9] To exit vi, hit the **<ESC>** key, type a colon <:> and enter **wq** on the command line.

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

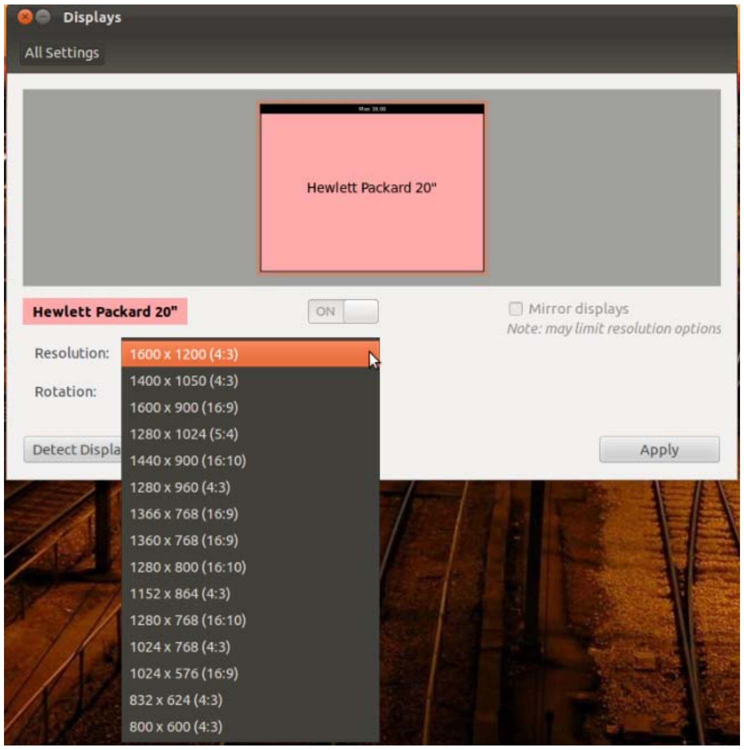


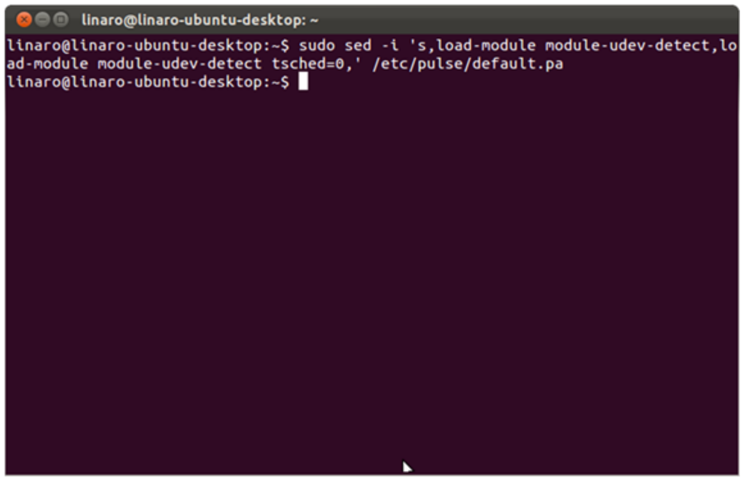
Figure 76 – Display Settings after Driver Update

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
```



```
linaro@linaro-ubuntu-desktop:~$ sudo sed -i 's,load-module module-udev-detect,lo
ad-module module-udev-detect tsched=0,' /etc/pulse/default.pa
linaro@linaro-ubuntu-desktop:~$
```

Figure 77 – Update Pulse Audio in Terminal Window

System Shudown

This is now a desktop computer environment, so you should follow a standard shutdown procedure before removing power from the Avnet target. 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.

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 target board.

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. If your display device has HDMI audio capabilities, use it to demonstrate HDMI audio from Ubuntu applications.

Experiment 1 Step-by-Step Instruction:

1. If you have not already done so, boot your Ubuntu desktop on the Avnet target using the SD/microSD 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 powered hub (ZedBoard only)
    c. USB Mouse
    d. USB Keyboard
    e. (Optional) Ethernet cable to connect to a DHCP router (or similar) for Internet access via Firefox

    Although audio is available over HDMI by default, there are some additional configuration steps required to employ audio over the Analog Devices ADAU1761 codec (used with the audio jacks). Audio coverage with the codec deferred to Experiment 2 in this lab.

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.

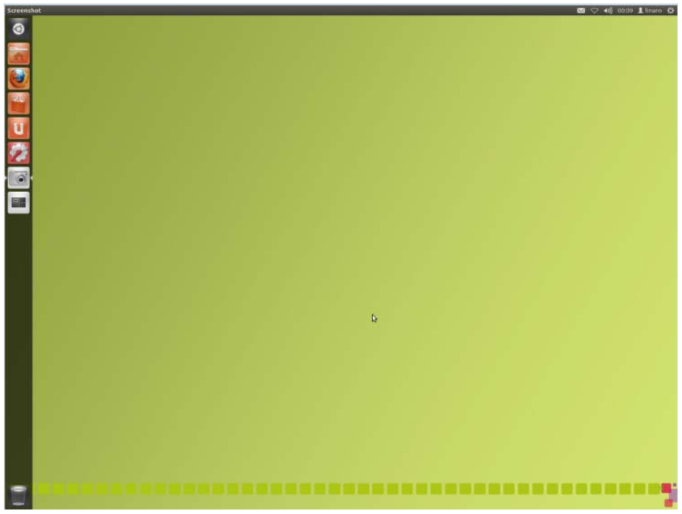


Figure 78 – Ubuntu Default Desktop

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.

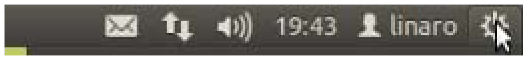
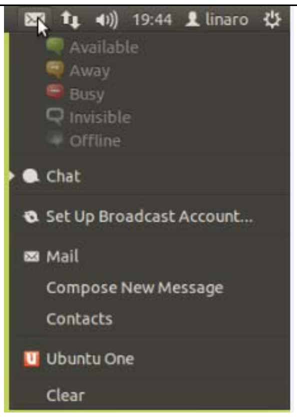
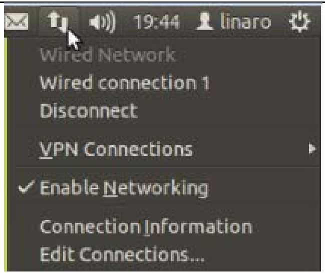
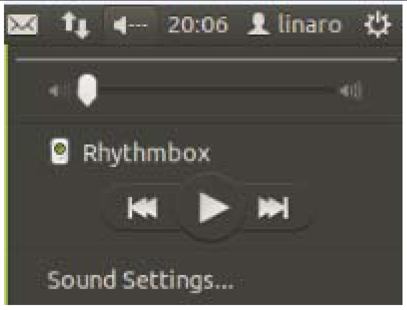
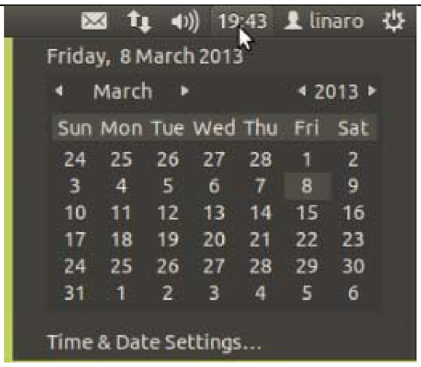
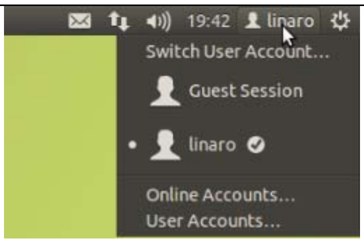
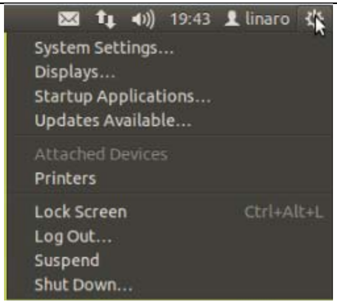


Figure 79 – Ubuntu Control Icons

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		 19:44 linaro Available Away Busy Invisible Offline Chat Set Up Broadcast Account... Mail Compose New Message Contacts Ubuntu One Clear
Networking		 19:44 linaro Wired Network Wired connection 1 Disconnect VPN Connections ✓ Enable Networking Connection Information Edit Connections...
Sound Controls		 20:06 linaro Rhythmbox Sound Settings...
System clock and calendar functions		 19:43 linaro Friday, 8 March 2013 March 2013 Time & Date Settings...

User Account information		 19:42 linaro Switch User Account… Guest Session linaro Online Accounts… User Accounts…
System operations, including Log Off and Shut Down		 19:43 linaro System Settings… Displays… Startup Applications… Updates Available… Attached Devices Printers Lock Screen Ctrl+Alt+L Log Out… Suspend 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 in the Dashboard section

	below.
	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.

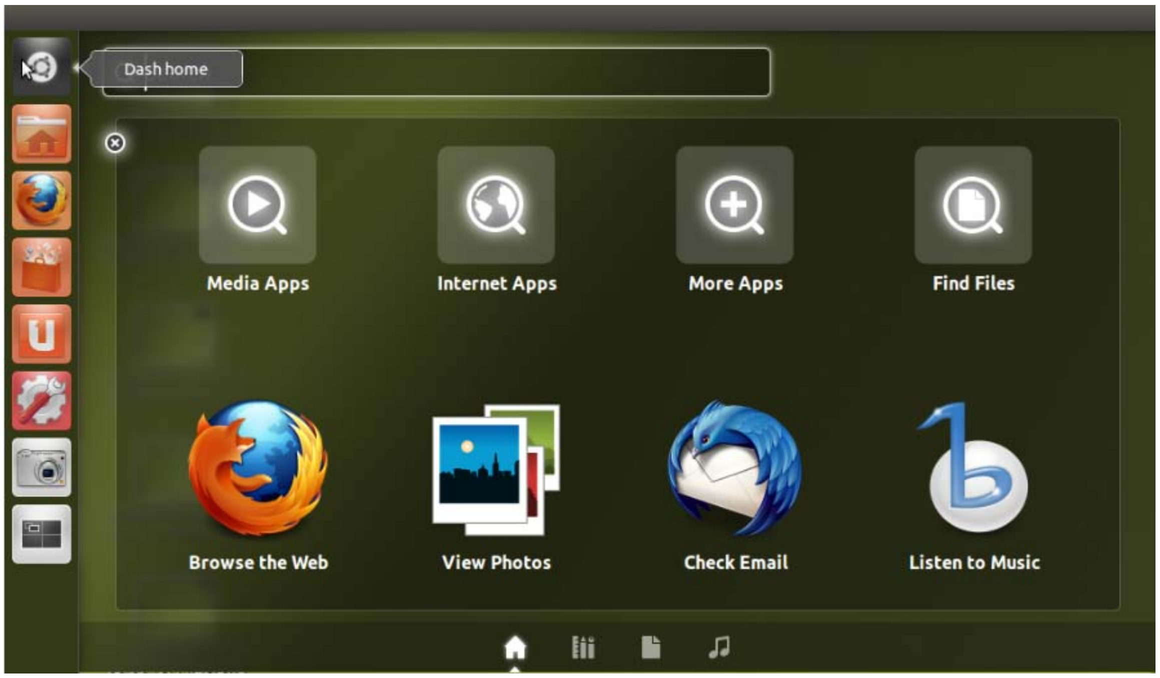


Figure 80 – Ubuntu Dashboard

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.

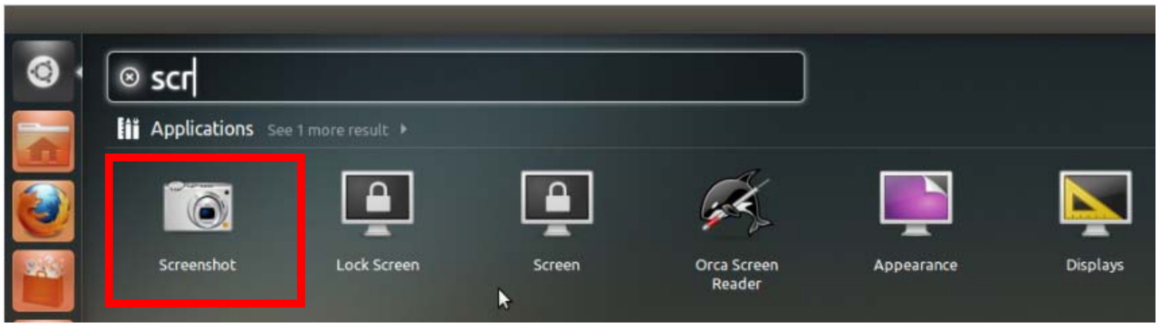


Figure 81 – Locate Applications with Dashboard

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.

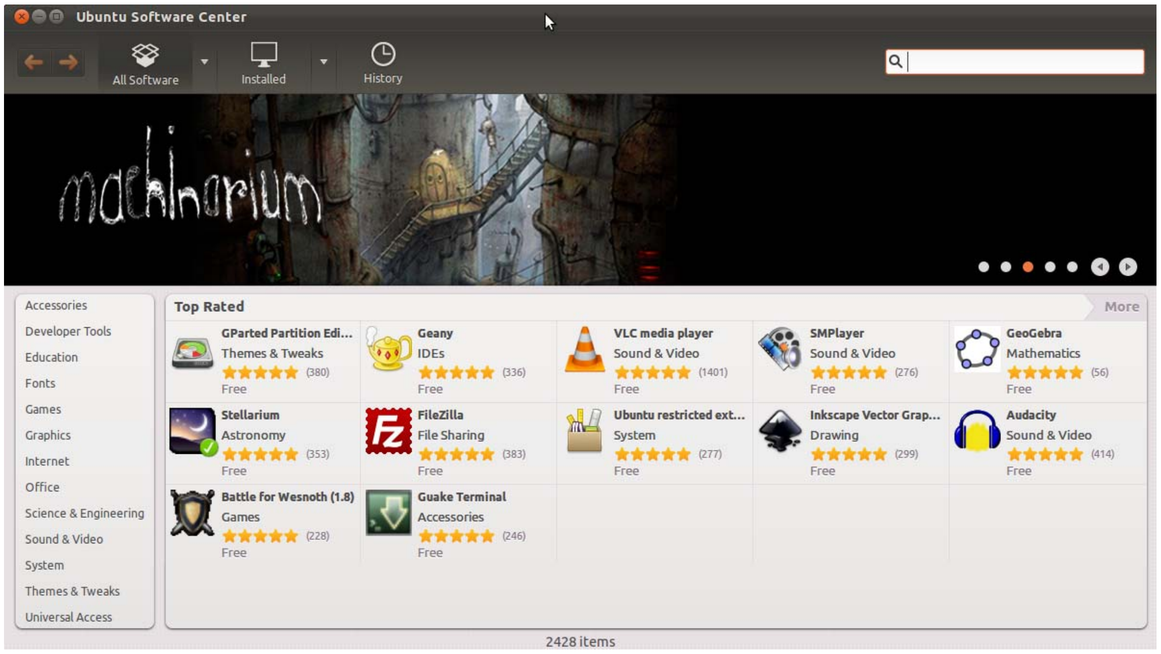
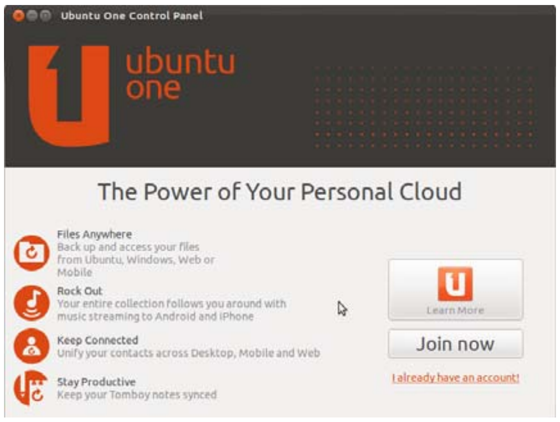

Ubuntu Software Center
All Software
Installed
History
machinarium
Accessories
Developer Tools
Education
Fonts
Games
Graphics
Internet
Office
Science & Engineering
Sound & Video
System
Themes & Tweaks
Universal Access
Top Rated
More
GParted Partition Edi... Themes & Tweaks (380) Free
Geany IDEs (336) Free
VLC media player Sound & Video (1401) Free
SMPlayer Sound & Video (276) Free
GeoGebra Mathematics (56) Free
Stellarium Astronomy (353) Free
FileZilla File Sharing (383) Free
Ubuntu restricted ext... System (277) Free
Inkscape Vector Grap... Drawing (299) Free
Audacity Sound & Video (414) Free
Battle for Wesnoth (1.8) Games (228) Free
Guake Terminal Accessories (246) Free
2428 items

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.



Ubuntu One Control Panel
ubuntu one
The Power of Your Personal Cloud
Files Anywhere
Back up and access your files from Ubuntu, Windows, Web or Mobile
Rock Out
Your entire collection follows you around with music streaming to Android and iPhone
Keep Connected
Unify your contacts across Desktop, Mobile and Web
Stay Productive
Keep your Tomboy notes synced
Learn More
Join now
I already have an account!

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:

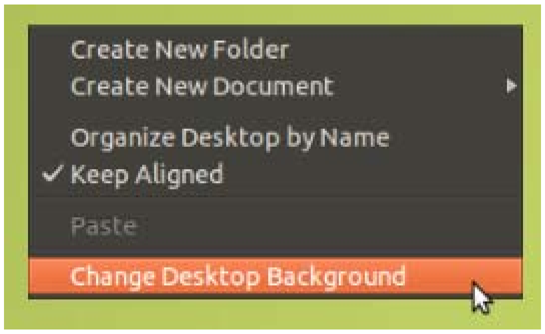


Figure 84 – Ubuntu Desktop Pop-up Menu

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.

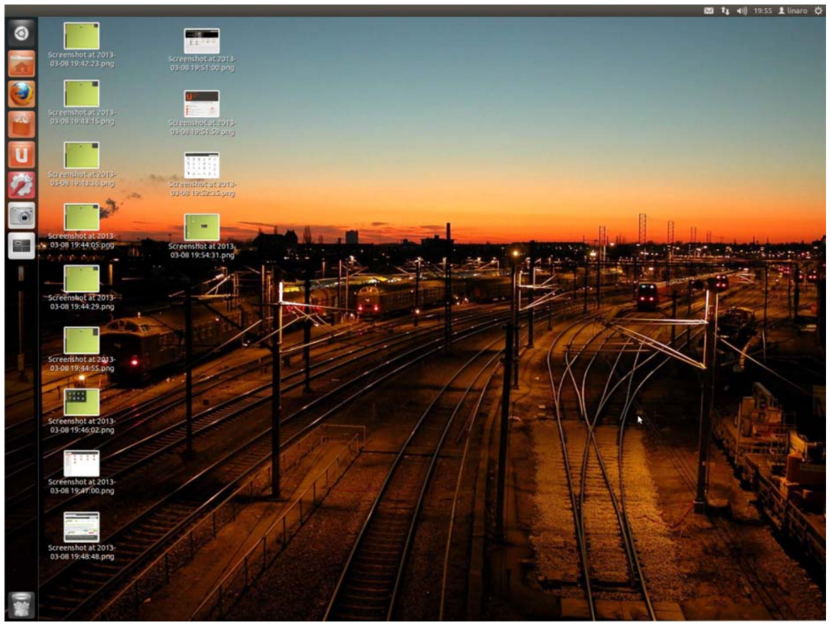


Figure 85 – Ubuntu Sample Custom Background

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.

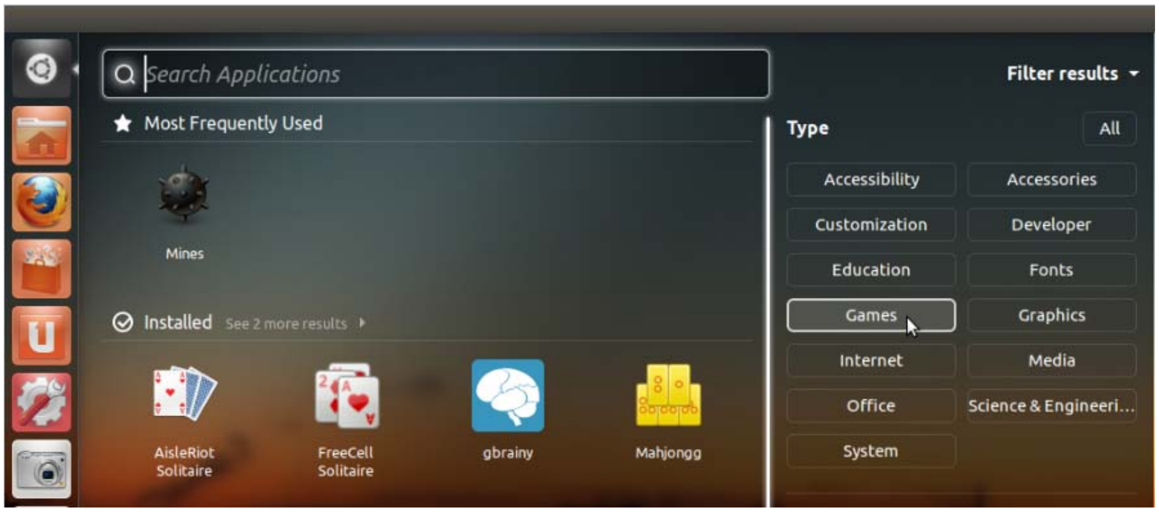


Figure 86 – Ubuntu Games

9. HDMI Audio

HDMI audio is provided by the Analog Devices ADV7511 transmitter using time division multiplexing to “mix” the video and audio signals on the HDMI cable. If you are using an HDMI device capable of processing audio signals received over the HDMI cable, then you will be able to hear audio on that device in the default configuration. You can test this by using the built-in Sound panel available in the Ubuntu desktop.

1. At the top right of the Ubuntu desktop, locate the **Sound** icon:

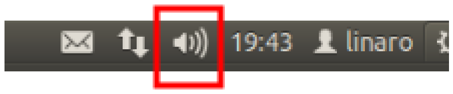


Figure 87 – Ubuntu Sound Icon

2. Click the **Sound icon** to expand the selections, and click on **Sound Settings**.

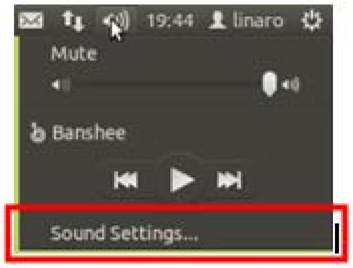
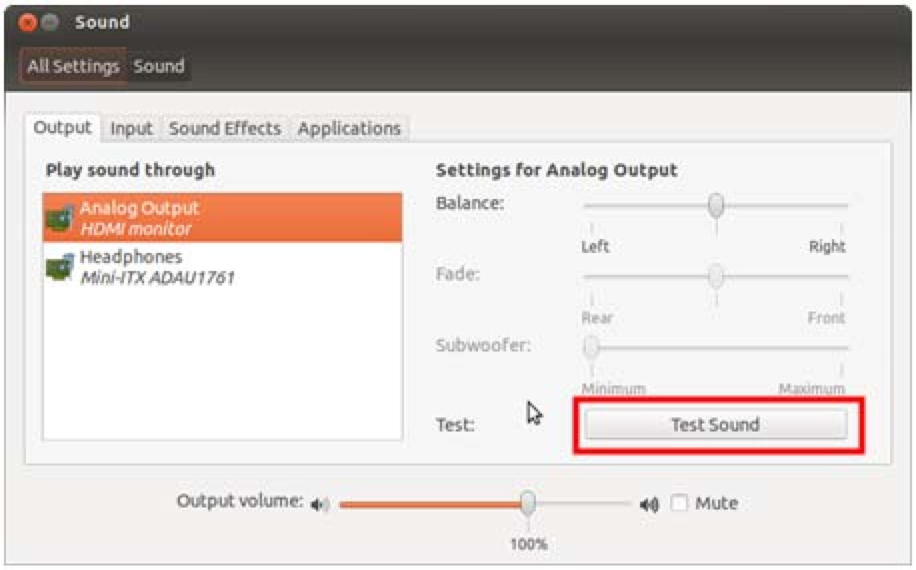


Figure 88 – Ubuntu Sound Settings

3. In the Sound panel, select the Output tab and ensure the default sound device selected is the **HDMI monitor**. As you can see in the display, you may also select the Headphones to play sound through that ADAU1761 audio codec. However, until we configure the codec parameters in the following experiment, the results may be less than optimal, so we will defer this until Experiment 2 is completed.[10]



Play sound through

Analog Output
HDMI monitor

Headphones
Mini-ITX ADAU1761

Settings for Analog Output

Balance: Left Right

Fade: Rear Front

Subwoofer: Minimum Maximum

Test:

Output volume: 100%

Figure 89 – Ubuntu Sound Panel, Output Tab

[10] In some cases you may find there are no devices listed in the Sound window, even though the monitor and ADAU1761 were detected by ALSA in the boot log. This appears to be an intermittent problem with PulseAudio. It may be possible to correct the issue by uninstalling and reinstalling PulseAudio, but this is not guaranteed to resolve the issue. In any event, the command line audio tests (later) will still function correctly, and this problem should be corrected in a new PulseAudio release in the future.

4. To hear audio from the HDMI device, click on the **Test Sound** button (see previous Figure) to open the Speaker Testing panel.

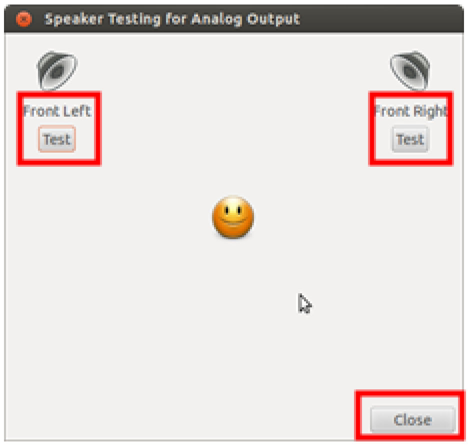
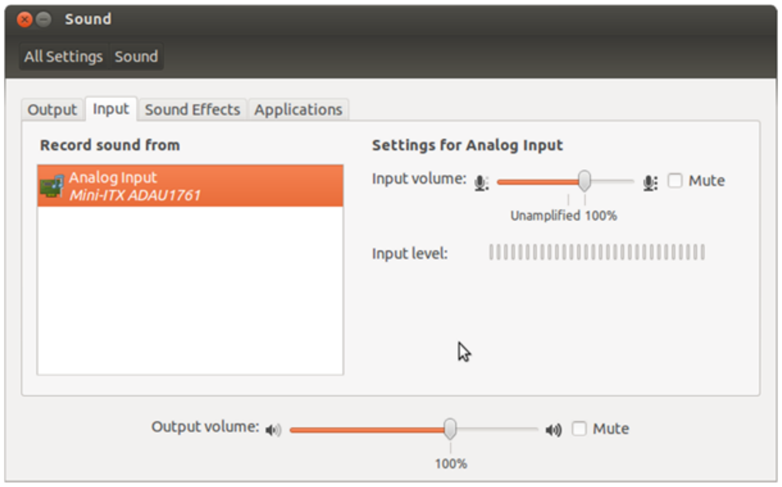


Figure 90 – Ubuntu Speaker Testing Panel

Alternately click the Test buttons for "Front Left" and "Front Right". You should hear sound[11] coming from the associated HDMI speaker.

5. While not required for this audio test, it is instructive to view the Input tab while we have the Sound panel open. Click on the Input tab, and note that by default the audio input device selected is the Analog Devices ADAU1761.



Analog Input
Mini-ITX ADAU1761

Figure 91 – Ubuntu Sound Panel, Input Tab

[11] The actual sound may vary depending on the specific release of the Root File System, but at a minimum you should hear the default system sound – a water drop, or sonar ping, or something similar – clearly audible from the selected HDMI speaker.

You may close the Sound panel at this point by clicking on the **Close** icon at the upper left.

6. With the HDMI audio output verified, we can now try a more prolonged test by playing a Waveform Audio (.wav) file using one of the Ubuntu Desktop applications. There are many applications for playing audio files on Ubuntu, some installed by default and many more available from the Internet. For simplicity, in this instance we will simply use whatever default audio application is currently in place to process the .wav format.

   To prevent any possibility of copyright violation, you will need to supply your own .wav file to play. Again, these can be downloaded from the Internet, created on any Windows platform with Windows Media Player installed, or generated on a host Linux system.

   Place your .wav file on the Ubuntu Desktop using the File Browser icon from the Launchpad on the left of the display.

7. Double-click on your .wav file. The default application will launch, and the audio will start playing over the HDMI speakers. In this instance, the default player was the Movie application.

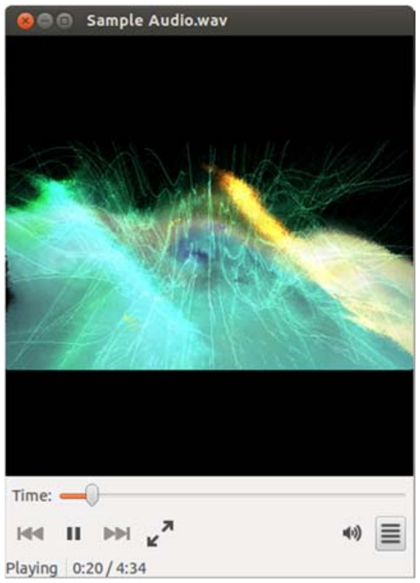


Figure 92 – Playing Audio with Ubuntu Movie Player

This completes the HDMI audio section. You may close the default audio player.

Experiment 2: Audio using the ADAU1761 Codec

The Analog Devices ADAU1761 audio codec is capable of processing stereo input and output using the four audio jacks located on the perimeter of the Avnet targets. If you wish to use headphones, external speakers, or process incoming audio on the Ubuntu system, you will need to use this codec, and the associated software drivers.

There are four jacks that are used with standard 1/8" audio plugs on the Avnet targets:

1. **Headphone out**: used for output to standard unpowered headphones, earbuds or headsets.
2. **Microphone in**: used with a standard unpowered microphone or headset to capture audio input.
3. **Line out**: used with powered external speakers, or as input to an external amplifier.
4. **Line in**: used to receive audio input from another device, such as the headphone output from a PC, or line out from standard audio equipment.

Audio using the jacks is routed through the Analog Devices ADAU1761 audio codec. The Linux drivers supplied in the Linaro kernel make use of the Advanced Linux Sound Architecture (ALSA) to produce high quality stereo output and receive stereo input for recording or playback. The ALSA architecture includes a software mixer which must be configured using an initialization (state) file that creates a path in the codec from the audio jacks to the processing system, and controls various settings such as channels, volume, voltage and signal mixing. Once the configuration is complete, the initialization file can be saved to a default location so the system will be automatically configured at boot time.

The test procedure to validate the audio jacks requires the use of a headphones, a microphone (or a headset with audio output and input jacks) and a male-to-male 1/8" audio cable. This cable is used to connect a PC headphone jack to the input jacks on the Avnet target.

Experiment 2 General Instruction:

Use built-in ALSA command line applications for playback and recording to demonstrate the use of the ADAU1761 audio codec, via the audio jacks on the Avnet target.

Experiment 2 Step-by-Step Instruction:

1. Before we can use the audio codec, it is necessary to initialize the device via the device drivers, to establish a circuit from the audio jacks to the SoC, and to initialize the audio parameters for playback and recording. There are many separate parameters associated with this task, so the easiest way to do this is to use an initial configuration file. This file was supplied as part of the tutorial package, located at:

   C:\<Avnet_target>\Zynq_Ubuntu\adau1761_golden.state

   Copy the initialization file to the Ubuntu Desktop using any convenient method[12].

2. The configuration file is a simple hierarchical text file, with each control identified and initial parameters provided in separate blocks. Consider the example snippets below:

```
state.monitor {
     control {
     }
}
```

   The first block identifies the HDMI monitor as a sound device, but there are no ALSA configuration parameters associated with it. As we have seen, an audio-capable HDMI monitor will produce audio by default under Ubuntu.

```
state.ADAU1761 {
     control.1 {
          iface MIXER
          name 'Digital Capture Volume'
          value.0 255
          value.1 255
          comment {
               access 'read write'
               type INTEGER
               count 2
```

[12] For example, you can access the file on your local network directly from the Ubuntu desktop over the Ethernet connection, or you can copy the file to a USB thumb drive and plug it into one of the open USB ports on your embedded system.

```
                range '0 - 255'
                dbmin -9563
                dbmax 0
                dbvalue.0 0
                dbvalue.1 0
          }
    }
```

The next block introduces the ADAU1761 as the second audio device. There are 38 separate control parameters associated with the device, from Digital Playback Volume (shown below) to Lineout Playback Switch for selection of specific audio jacks, through mixer controls that can be used to mix audio tracks from one input to another. There are typically two values for each control, corresponding to the left and right stereo settings, and the initial value is provided next to the identifiers. In the comment block, the ranges and parameters for various audio configuration elements are provided, and these are specific to the ADAU1761 and can be procured from the Hardware manual for the device.

```
    control.2 {
          iface MIXER
          name 'Digital Playback Volume'
          value.0 255
          value.1 255
          comment {
                access 'read write'
                type INTEGER
                count 2
                range '0 - 255'
                dbmin -9563
                dbmax 0
                dbvalue.0 0
                dbvalue.1 0
          }
    }
```

To illustrate these points, the control block labeled 2 is identified as an interface to the ALSA mixer device for Digital Playback Volume.

```
          iface MIXER
          name 'Digital Playback Volume'
```

In the comment block, we can see the values can be seen and changed by the driver:

```
          access 'read write'
```

The number of channels for the control is 2, indicating stereo:

```
count 2
```

The valid values for the volume are 0 (mute) through 255:

```
range '0 - 255'
```

Referring back to the initialization section above the comment block, we can see that by default the Digital Playback Volume is set to the maximum allowed for each channel:

```
value.0 255        (Left channel)
value.1 255        (Right channel)
```

Further down in the file initial volume controls can be set independently for headphones, microphone and line in, as well as dynamic volume control during operation through the Ubuntu Desktop. You may wish to explore this file using any text editor to experiment with the effect different values have on the input and output audio.

3. To initialize the audio codec for the first time, we can use a terminal window to send command line instructions to the software. Using the launchpad at the left of the Ubuntu Desktop, open a terminal window.

4. Change directories to the desktop, and issue the command:

   alsactl restore –f adau1761_golden.state

   This instructs the ALSA architecture to perform the device initialization specified in the named file. The values supplied in the file can be changed using a text editor, and you may issue the command above each time you change the file in order to adopt your updated audio configuration.

   Next, we will validate the operation from each of the audio jacks, and once everything is tested the final step will be to save the initial configuration permanently, so it will be the new boot default.

5. In this step, you will connect the headphone output of your host PC to the audio input jacks on the Avnet target. The audio will be captured and re-routed out the playback jacks to verify that all four connections are operational.

a. Connect the 1/8” male-to-male audio plug to the headphone output of your host PC.

Figure 93 – Male Audio Jack in Host PC Headphone Output

b. Connect the other end of the male-to-male audio plug to the microphone jack of the Avnet target.

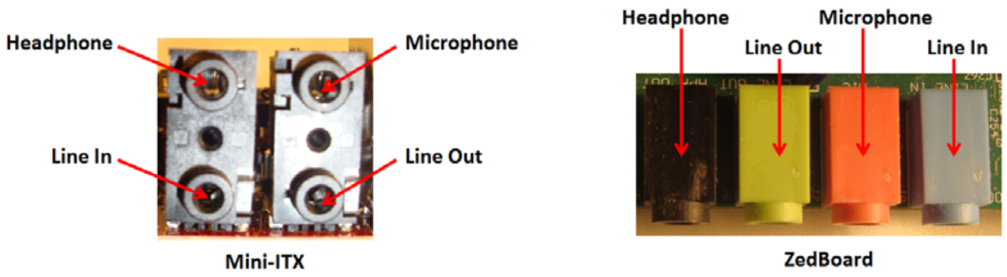


Figure 94 – Male Audio Jack in Microphone Input

c. Plug your headphones into the headphone jack of the Avnet target.

d. On the host PC, select any audio file for playback through the headphone output. An .mp3 or .wav music file is a good choice for this test, as it provides persistent audio at various ranges throughout. Double-click on the audio file on the host PC to initiate playback using Windows Media Player.

e. In the terminal window of the Ubuntu Desktop, we will use the ALSA command line audio recorder and audio player together to send the incoming audio to the ADAU1761, and route it back out through your headphones. This will validate the microphone and headphone jacks.

In the Terminal window, enter:

arecord –Dplughw:ADAU1761 –f S32 –r 48000 | aplay –Dplughw:ADAU1761

This command selects the ADAU1761 as the input device for recording, and the output device for playback. You should hear the audio originating on your host PC through the headphones connected to the Avnet target.

f. While the audio is playing, unplug the microphone jack from the Avnet target, and insert it into the **Line In** receptacle. Once again, you should hear the audio from your host PC through the headphones.

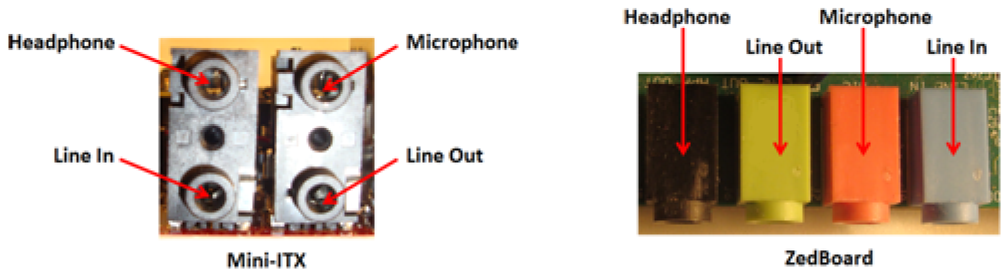


Figure 95 – Male Audio Jack to Line In

g. While the audio continues to play, unplug the headphone jack and insert it into the Line Out receptacle. You will hear the audio from your host PC, but this time at a lower volume.

If all of these tests functioned as described, you have validated the hardware and software connections on your Avnet target for the audio codec.

6. If you used a .wav file to test the HDMI audio in Experiment 1, you can also play that same file through the audio codec. You can do this from the command line using the ALSA player with the following command. You may have the headphones plugged into the **Line Out** or the **Headphone** receptacle, with superior volume in the latter.

   In the terminal window, enter:

   - **aplay –Dplughw:ADAU1761 –r 48000 –f S32 <audio_file>.wav**

7. As described earlier, you may use a text editor to experiment with the audio configurations in the adau1761_golden.state file. Once you have a set of parameters you would like to save as the default, you may do this from the terminal window, using the following command:

   alsactl store

   This command saves the current audio parameters to the default configuration file at:

/var/lib/alsa/asound.state

This file will be used by the Linux kernel at boot time to configure the ADAU1761 audio codec.

This completes the Ubuntu tutorial.

Appendix I - Installing the VMware Player

This experiment shows how to install VMware Player which will enable the use of a Linux virtual machine workstation to be used for the cross build platform.

General Instruction:

Install VMware Player using the official VMware installer. For legal distribution reasons, the VMware player installation executable cannot be included with any public Avnet materials. To obtain a free legal copy of VMware player, please download a copy from the VMware website:

http://www.vmware.com/products/player/overview.html

The version downloaded may differ from the version shown in this documentation.

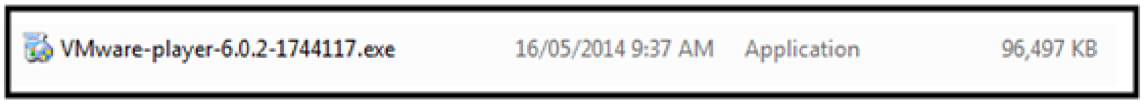
Step-by-Step Instructions:

1. To download a free legal copy of **VMware Player**, please download a copy from the VMware website:

   http://www.vmware.com -> Downloads

   The version downloaded may differ from the version shown in this documentation.

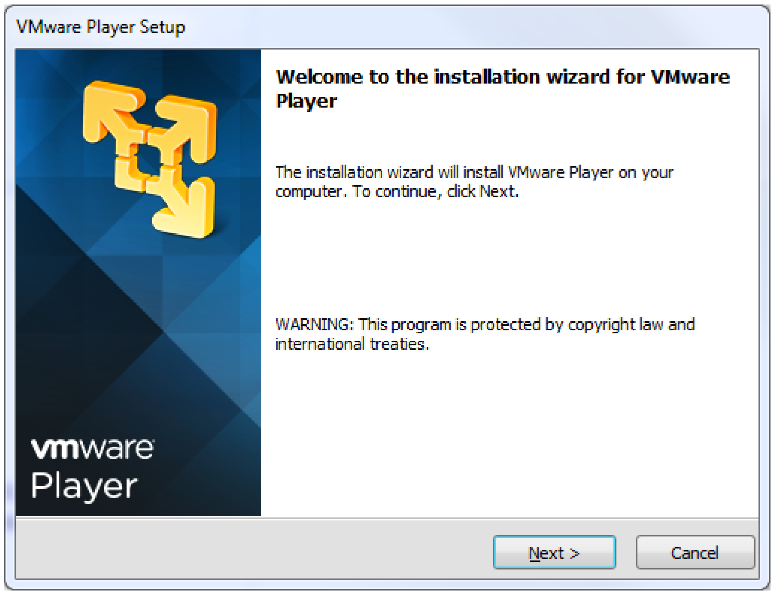
2. Launch the VMware Player installer from Windows Explorer by double-clicking on the self-extracting executable. Allow the installer to make changes to your computer, if so prompted.



VMware-player-6.0.2-1744117.exe 16/05/2014 9:37 AM Application 96,497 KB

Figure 96 – VMware Player installer for Windows

3. Once the VMware Player installation wizard appears, click on the **Next** button.



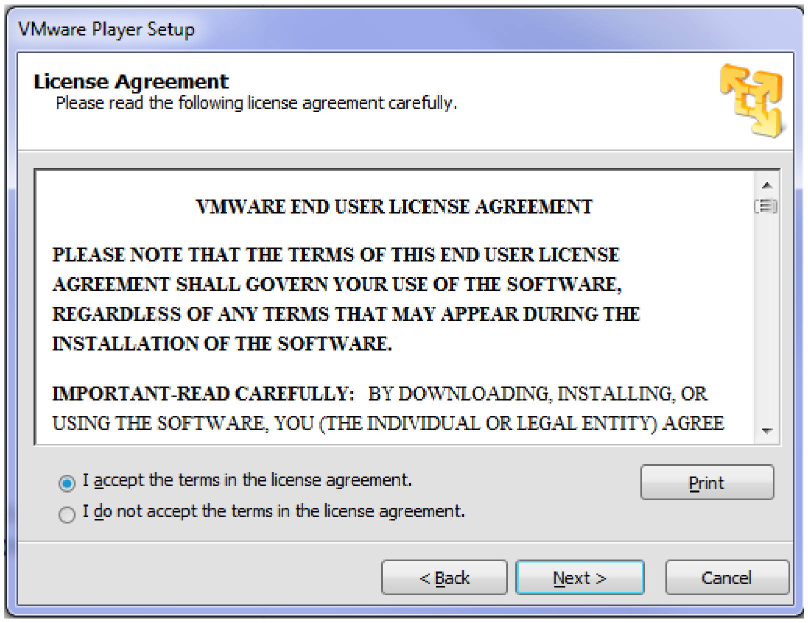
Welcome to the installation wizard for VMware Player

The installation wizard will install VMware Player on your computer. To continue, click Next.

WARNING: This program is protected by copyright law and international treaties.

Figure 97 – VMware Player Installation Wizard Welcome

4. Read the license agreement and select the "I accept..." radio button to proceed. Click the **Next** button.



License Agreement
Please read the following license agreement carefully.

VMWARE END USER LICENSE AGREEMENT

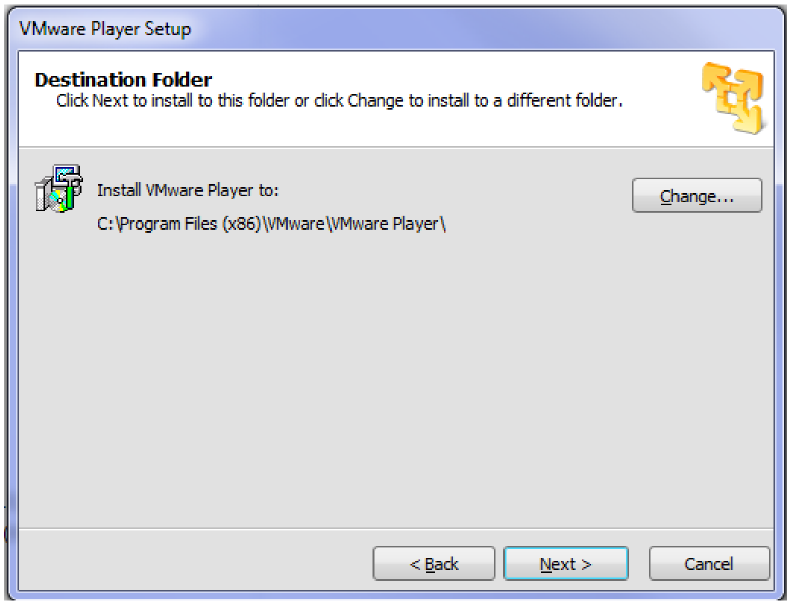
PLEASE NOTE THAT THE TERMS OF THIS END USER LICENSE AGREEMENT SHALL GOVERN YOUR USE OF THE SOFTWARE, REGARDLESS OF ANY TERMS THAT MAY APPEAR DURING THE INSTALLATION OF THE SOFTWARE.

IMPORTANT-READ CAREFULLY: BY DOWNLOADING, INSTALLING, OR USING THE SOFTWARE, YOU (THE INDIVIDUAL OR LEGAL ENTITY) AGREE

[✓ I accept the terms in the license agreement.]
I do not accept the terms in the license agreement.

Figure 98 – VMware Player License Agreement

5. In the *Destination Folder* dialog, select the folder that the VMware player application files will installed into. Once the destination folder has been specified correctly, click the **Next** button. It is strongly recommended that the default suggested destination folder is used.

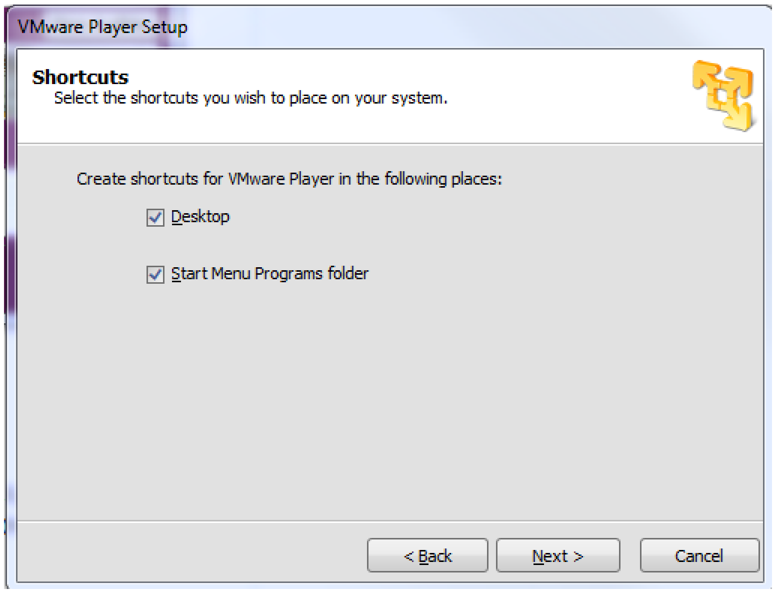

VMware Player Setup

Destination Folder
Click Next to install to this folder or click Change to install to a different folder.

Install VMware Player to:
C:\Program Files (x86)\VMware\VMware Player\

Figure 99 – VMware Player Destination Folder Selection

6. Select your preference for shortcuts and click the **Next** button.

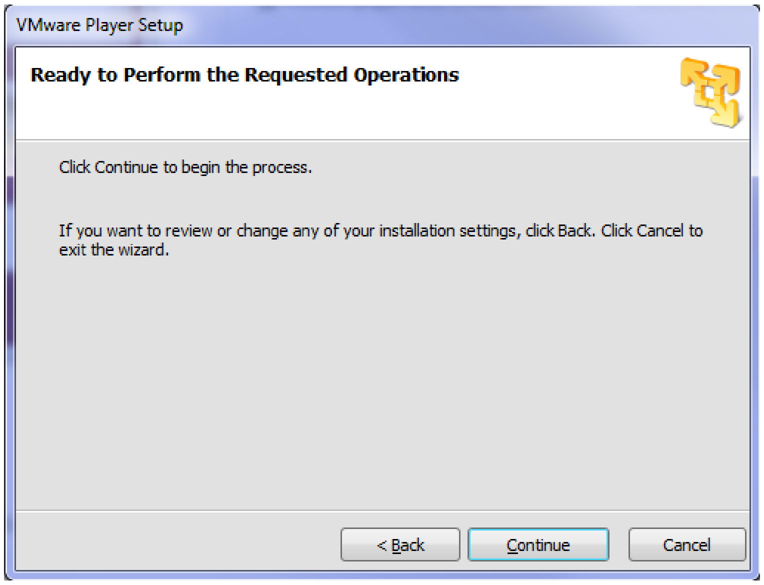

VMware Player Setup

Shortcuts
Select the shortcuts you wish to place on your system.

Create shortcuts for VMware Player in the following places:
☑ Desktop
☑ Start Menu Programs folder

Figure 100 – VMware Player Shortcuts Selection

7. Click the **Continue** button.



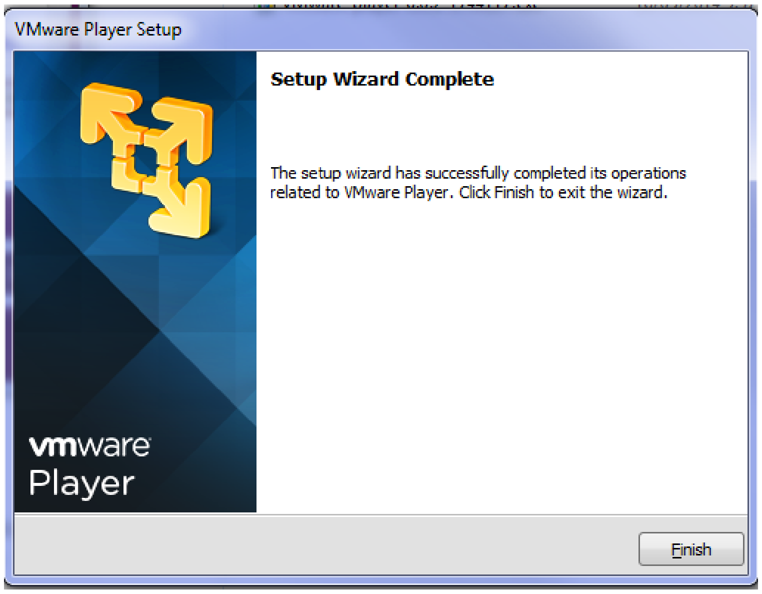
Ready to Perform the Requested Operations

Click Continue to begin the process.

If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

Figure 101 – VMware Player Ready to Install

8. Once the install process has completed, click the **Finish** button to complete the install. Reboot the host machine at this point only if prompted to do so.



Setup Wizard Complete

The setup wizard has successfully completed its operations related to VMware Player. Click Finish to exit the wizard.

Figure 102 – VMware Player Installation Complete

Appendix II - Installing CentOS as the VMware OS

General Instruction:

Obtain the CentOS installation media image(s) from one of the official servers listed on http://www.centos.org and create a new VMware virtual machine using the downloaded CentOS media as the source image.

Important

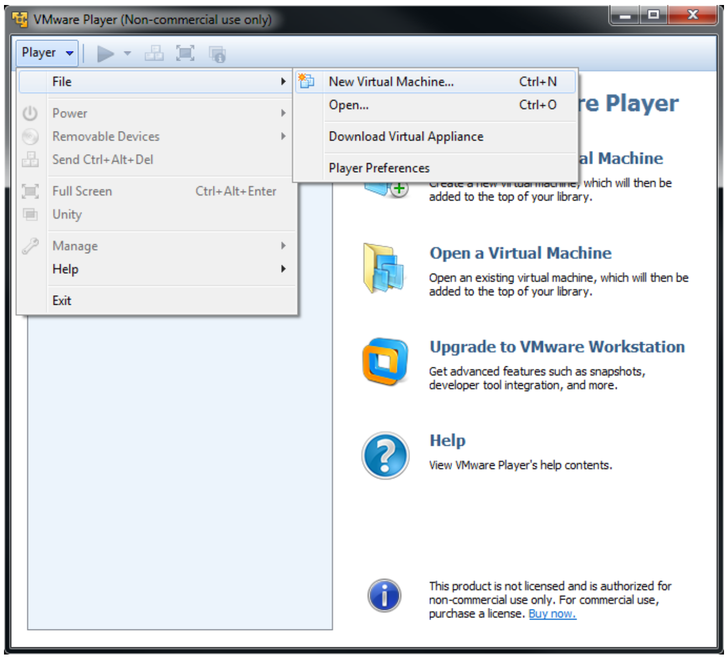
Installation of a 64-bit guest operating system is only recommended on a 64-bit host operating system. Also, virtualization technology (AMD-V or VT-x) must be supported by the processor and enabled in the BIOS.

The iso file used to prepare this documentation is:

CentOS-6.5-x86_64-bin-DVD1.iso

Step-by-Step Instructions:

1. Launch VMware player and select menu **Player → File → New Virtual Machine**, or select the action item **Create a New Virtual Machine** at the right of the window.


VMware Player (Non-commercial use only)
Player
File
Power
Removable Devices
Send Ctrl+Alt+Del
Full Screen Ctrl+Alt+Enter
Unity
Manage
Help
Exit
New Virtual Machine... Ctrl+N
Open... Ctrl+O
Download Virtual Appliance
Player Preferences
re Player
al Machine
Open a Virtual Machine
Upgrade to VMware Workstation
Help

which will then be added to the top of your library.

Open an existing virtual machine, which will then be added to the top of your library.

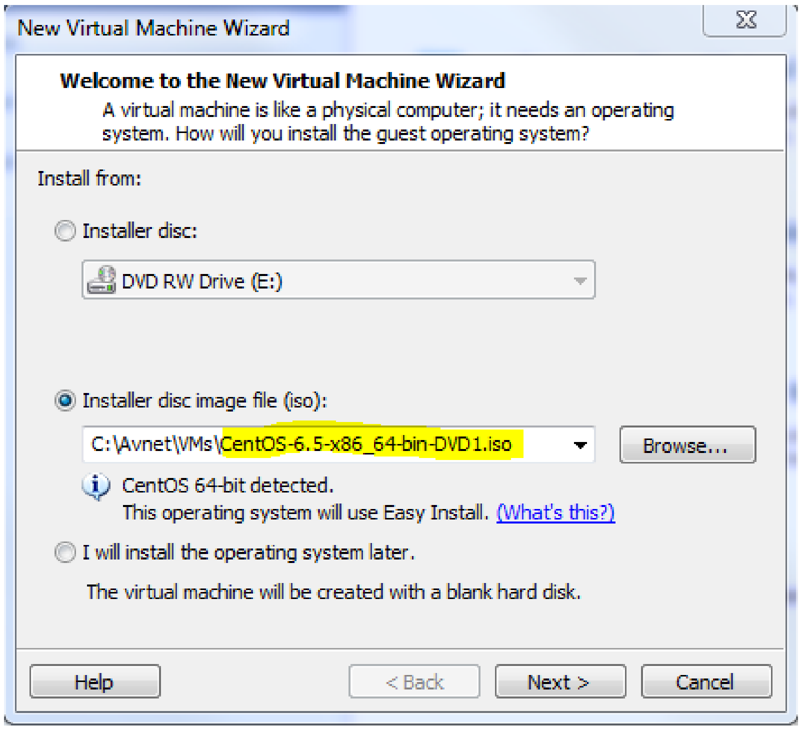
Get advanced features such as snapshots, developer tool integration, and more.

View VMware Player's help contents.

This product is not licensed and is authorized for non-commercial use only. For commercial use, purchase a license. Buy now.

Figure 103 – Using VMware Player to Create a New Virtual Machine

2. In the *New Virtual Machine Wizard* select the **Installer disc image file (iso)** option, locate your CentOS installation media image file, and click the **Next** button.



New Virtual Machine Wizard

Welcome to the New Virtual Machine Wizard

A virtual machine is like a physical computer; it needs an operating system. How will you install the guest operating system?

Install from:

( ) Installer disc:

DVD RW Drive (E:)

(●) Installer disc image file (iso):

C:\Avnet\VMs\CentOS-6.5-x86_64-bin-DVD1.iso

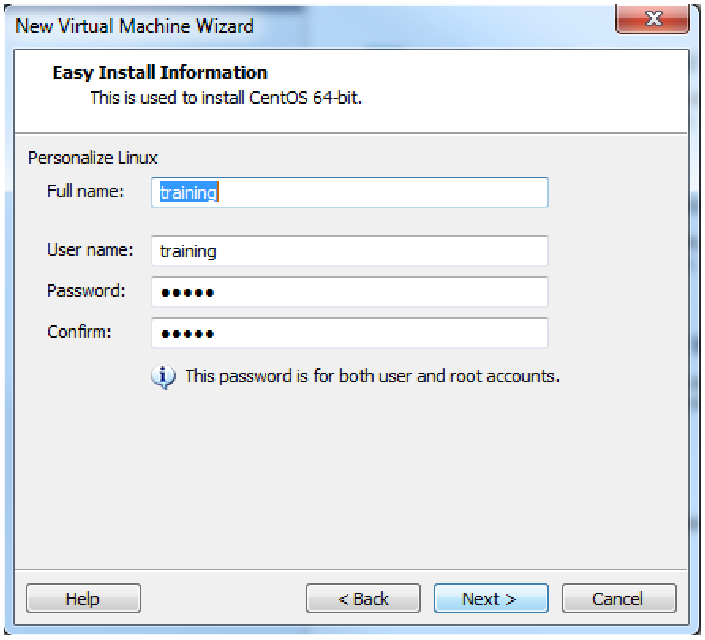
CentOS 64-bit detected.
This operating system will use Easy Install. (What's this?)

( ) I will install the operating system later.

The virtual machine will be created with a blank hard disk.

Figure 104 – Selecting the Install Media for New Virtual Machine

3. Under *Easy Install Information*, enter in information that is appropriate to your own install and then click the **Next** button. The User Name and password will be the primary login account for your development system. For the purposes of this documentation, we use the name "training".



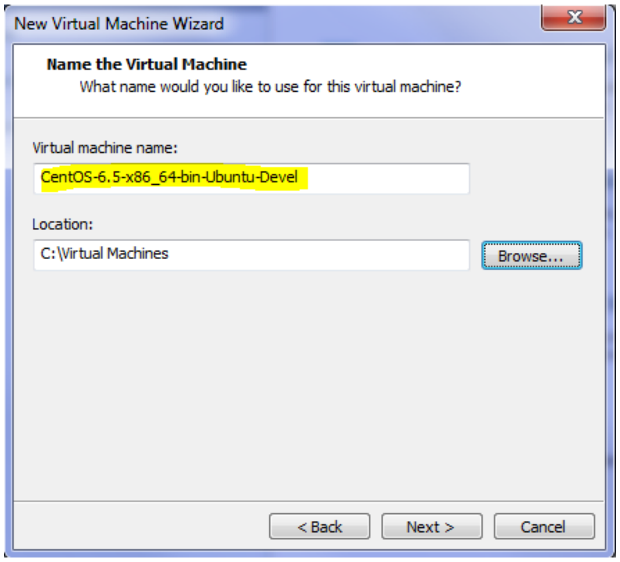
New Virtual Machine Wizard

Easy Install Information
This is used to install CentOS 64-bit.

Personalize Linux
Full name: training
User name: training
Password: ●●●●●
Confirm: ●●●●●
This password is for both user and root accounts.

Figure 105 – CentOS Personalization

4. In the *Name the Virtual Machine* window, name the virtual machine **CentOS-6.5-x86_64-bin-Ubuntu-Devel**, select an appropriate path, and click the **Next** button.



New Virtual Machine Wizard

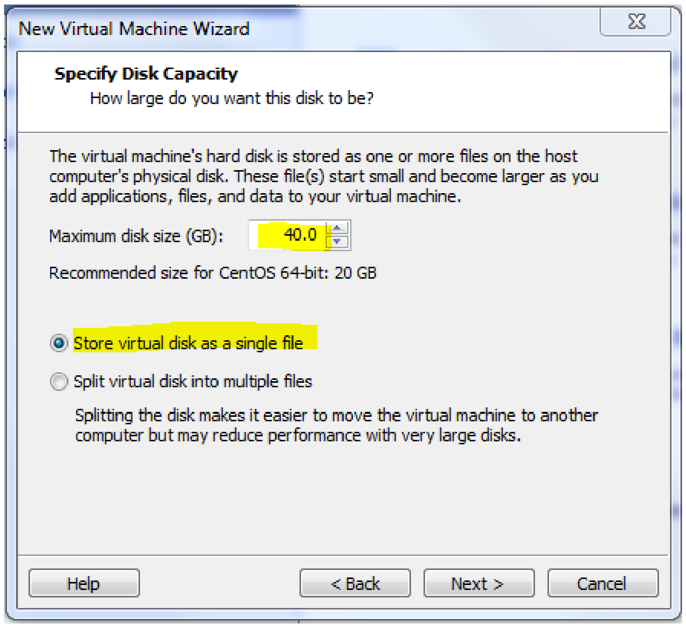
Name the Virtual Machine
What name would you like to use for this virtual machine?

Virtual machine name:
CentOS-6.5-x86_64-bin-Ubuntu-Devel

Location:
C:\Virtual Machines

Figure 106 – Naming the New Virtual Machine

5. In the *Specify Disk Capacity* window, use a size of **40.0 GB** or more to give plenty of room for tools you may want to install later. Select the option **Store virtual disk as a single file**, and then click the **Next** button.



Specify Disk Capacity
How large do you want this disk to be?

The virtual machine's hard disk is stored as one or more files on the host computer's physical disk. These file(s) start small and become larger as you add applications, files, and data to your virtual machine.

Maximum disk size (GB): 40.0

Recommended size for CentOS 64-bit: 20 GB

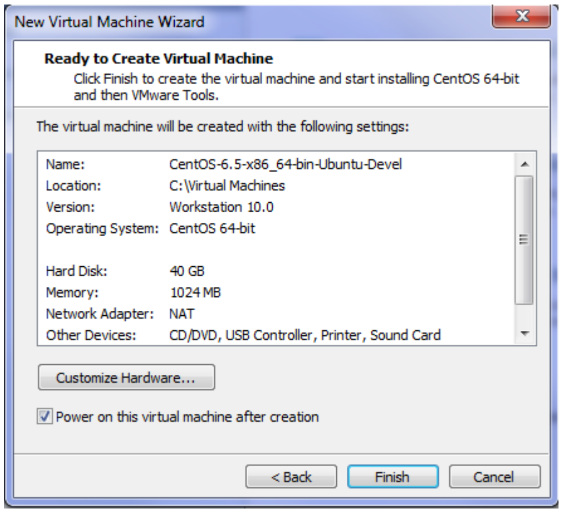
[✓ Store virtual disk as a single file]

Split virtual disk into multiple files

Splitting the disk makes it easier to move the virtual machine to another computer but may reduce performance with very large disks.

Figure 107 – Virtual Machine Disk Capacity

6. In the *Ready to Create Virtual Machine* window, verify the settings for the machine. Be certain that at least 1024MB is allocated for the virtual machine memory and then click the **Finish** button.



Ready to Create Virtual Machine
Click Finish to create the virtual machine and start installing CentOS 64-bit and then VMware Tools.

The virtual machine will be created with the following settings:

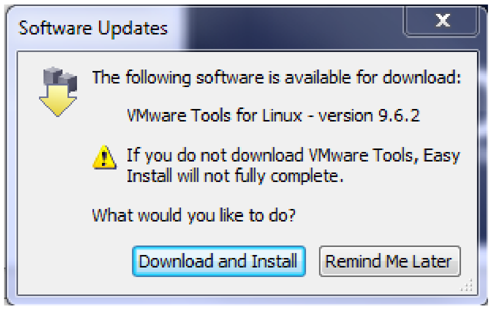
Name:	CentOS-6.5-x86_64-bin-Ubuntu-Devel
Location:	C:\Virtual Machines
Version:	Workstation 10.0
Operating System:	CentOS 64-bit
Hard Disk:	40 GB
Memory:	1024 MB
Network Adapter:	NAT
Other Devices:	CD/DVD, USB Controller, Printer, Sound Card

[✓] Power on this virtual machine after creation

Figure 108 – Virtual Machine Settings Review

The virtual machine will be created and CentOS installed in the new machine. The virtual machine creation and install will take 20-40 minutes to complete depending upon the host machine performance. If they security software installed in the PC, there may be additional prompts to allow changes to the system during the course of the virtual machine creation.

If prompted to download and install *VMware Tools for Linux*, click the **Download and Install** button to allow the download to complete and the VMware Update Launcher to run on the host OS. The latest version of VMware Tools is recommended to enhance the performance of the virtual machine guest operating system and improve virtual machine management.



Software Updates

The following software is available for download:

VMware Tools for Linux - version 9.6.2

If you do not download VMware Tools, Easy Install will not fully complete.

What would you like to do?

Figure 109 – Download and Install VMware Tools for Linux Prompt

7. Once the installation process is complete, login with your user account information specified earlier to view the CentOS desktop.

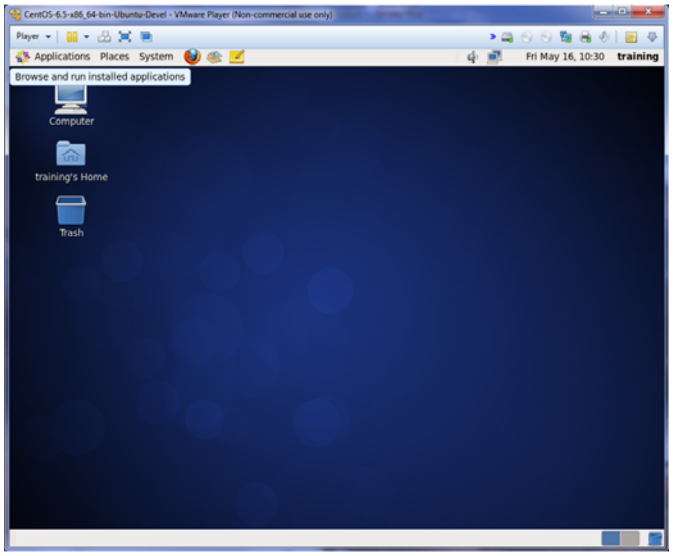

CentOS-6.5-x86_64-bin-Ubuntu-Devel - VMware Player (Non-commercial use only)
Player
Applications Places System
Fri May 16, 10:30 training
Browse and run installed applications
Computer
training's Home
Trash

Figure 110 – New Virtual Machine Following Installation

8. If you get a notification that the **VMware Tools for Linux** could not be installed automatically, these tools must be installed manually using the remaining steps in this section.

   If you do not get a notification, the tools were installed automatically in the background during the CentOS install, skip ahead to **Set up the Build Environment under CentOS**.

9. In the CentOS guest operating system, open a terminal window through the **Applications→System Tools→Terminal** menu item.

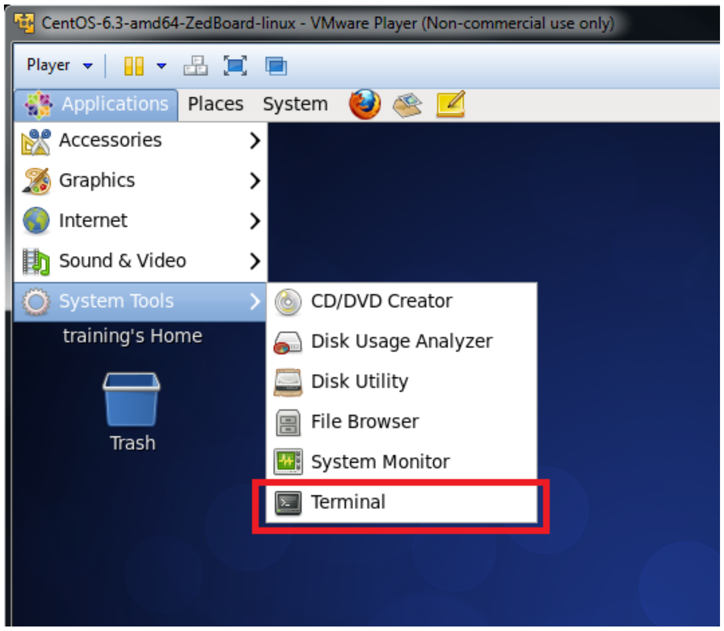
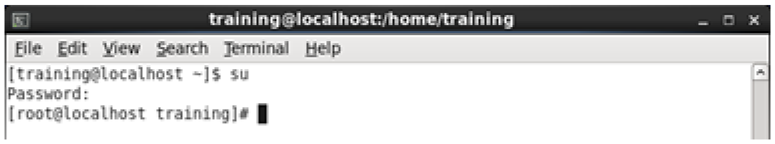


Figure 111 – Launching the CentOS Terminal from the Desktop

10. Take on root privileges by running the superuser elevation command **su** and entering the password specified earlier.

```
$ su
```


training@localhost:/home/training
File Edit View Search Terminal Help

```
[training@localhost ~]$ su
Password:
[root@localhost training]# 
```

Figure 112 – Elevating to Superuser Privileges

11. Install the VMware Tools for Linux prerequisites.

```
# yum install make gcc kernel-devel perl
```

12. On the host operating system, from the VMware Player menu bar, select the **Player→Manage→Install VMware Tools** option.

13. Run the mount command with no arguments to determine whether your Linux distribution automatically mounted the VMware Tools virtual CD-ROM image.

    If the CD-ROM device is mounted, the CD-ROM device and its mount point are listed as something like this:

```
/dev/cdrom on /mnt/cdrom type iso9660 (ro,nosuid, nodev)
```

14. If the VMware Tools virtual CD-ROM image is not mounted, mount the CD-ROM drive.

    If a mount point directory does not already exist, create it.

```
# mkdir /mnt/cdrom
```

15. Mount the virtual CD-ROM drive.

```
# mount /dev/cdrom /mnt/cdrom
```

16. Change to the **/tmp** folder as a working directory.

```
# cd /tmp
```

17. List the contents of the mount point directory and note the filename of the VMware Tools tar installer.

```
# ls mount-point
```

18. Uncompress the installer.

```
# tar zxpf /mnt/cdrom/VMwareTools-x.x.x-yyyy.tar.gz
```

    Note: The value x.x.x is the product version number, and yyyy is the build number of the product release. If you attempt to install a tar installation over an RPM installation, or the reverse, the installer detects the previous installation and must convert the installer database format before continuing.

19. Unmount the CD-ROM image.

```
# umount /dev/cdrom
```

20. Run the installer and configure VMware Tools.

```
# cd vmware-tools-distrib./vmware-install.pl
```

Respond to the prompts by pressing **Enter** to accept the default values

Follow the instructions at the end of the install script.

Depending on the VMware Tools for Linux features you use, these instructions can include restarting the X session, restarting networking, logging in again, and starting the VMware User process. You can alternatively reboot the guest operating system to accomplish all these tasks.

Set up the Build Environment under CentOS

In order to retrieve and build the latest code for U-Boot and the Linux Kernel from open source repositories, the build environment must be configured correctly. The Git SCM tools must be installed along with the Sourcery CodeBench cross toolchain for Zynq.

General Instruction:

Install the Git SCM tools followed by the Sourcery CodeBench cross toolchain for Zynq.

Step-by-Step Instructions:

1. In the CentOS guest operating system, open a terminal window through the **Applications→System Tools→Terminal** menu item.

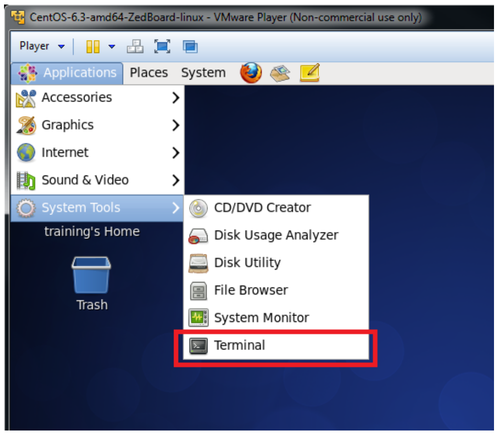
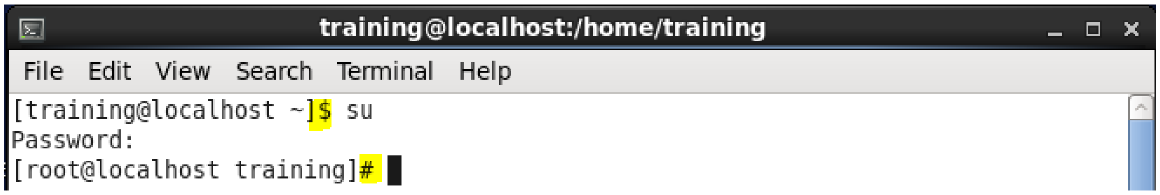


Figure 113 – Launching the CentOS Terminal from the Desktop

2. Take on root privileges by running the superuser elevation command **su** and entering your login password. The prompt changes from $ to # to indicate the privilege elevation succeeded.

```
$ su
```


training@localhost:/home/training
File Edit View Search Terminal Help

```
[training@localhost ~]$ su
Password:
[root@localhost training]#
```

Figure 114 – Elevating to Superuser Privileges

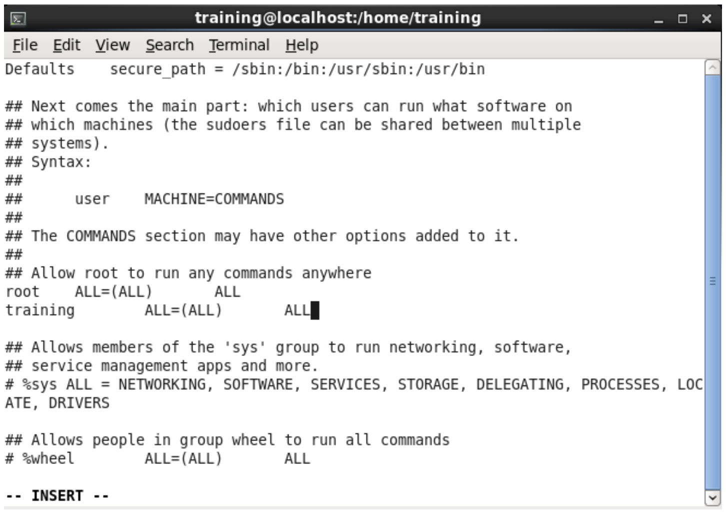
3. Use visudo text editor to edit the /etc/sudoers file.

```
# visudo
```

4. Add the **training** user to the sudoers list by inserting the following line to the users section of the sudoers file as shown in the Figure below. The users section is located towards the end of the file.

```
training    ALL=(ALL) ALL
```

To add text from within the vi editor, **press the `I` key** on the keyboard to enter vi insert mode.


training@localhost:/home/training
File Edit View Search Terminal Help

```
Defaults    secure_path = /sbin:/bin:/usr/sbin:/usr/bin

## Next comes the main part: which users can run what software on
## which machines (the sudoers file can be shared between multiple
## systems).
## Syntax:
##
##      user    MACHINE=COMMANDS
##
## The COMMANDS section may have other options added to it.
##
## Allow root to run any commands anywhere
root    ALL=(ALL)       ALL
training        ALL=(ALL)       ALL

## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
# %sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PROCESSES, LOC
ATE, DRIVERS

## Allows people in group wheel to run all commands
# %wheel        ALL=(ALL)       ALL

-- INSERT --
```

Figure 115 – Adding training Account to /etc/sudoers File

5. Exit the vi editor and save changes to the sudoers file by using the write-quit key sequence:

```
<ESC> :wq
```

6. The **training** user will now have sufficient privileges to do important systems tasks using the sudo command. Exit the superuser mode.

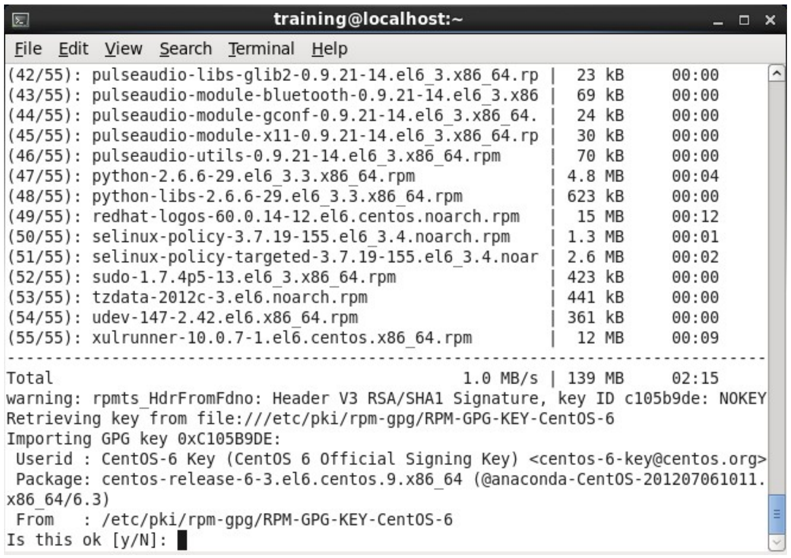
```
# exit
```

7. Next, the system should be revised with the latest updates. Use the yum package manager to install the system updates from the appropriate mirrors and when prompted accept the download and installation of all recommended packages. These updates can take several minutes and may present several user prompts before completion.

If prompted to allow download of packages, accept the download by pressing the **Y** key followed by the **Enter** key.

If prompted for the import of the GPG key accept the import by pressing the **Y** key followed by the **Enter** key.

```
$ sudo yum update
```



```
(42/55): pulseaudio-libs-glib2-0.9.21-14.el6_3.x86_64.rp |  23 kB     00:00
(43/55): pulseaudio-module-bluetooth-0.9.21-14.el6_3.x86 |  69 kB     00:00
(44/55): pulseaudio-module-gconf-0.9.21-14.el6_3.x86_64. |  24 kB     00:00
(45/55): pulseaudio-module-x11-0.9.21-14.el6_3.x86_64.rp |  30 kB     00:00
(46/55): pulseaudio-utils-0.9.21-14.el6_3.x86_64.rpm     |  70 kB     00:00
(47/55): python-2.6.6-29.el6_3.3.x86_64.rpm              | 4.8 MB     00:04
(48/55): python-libs-2.6.6-29.el6_3.3.x86_64.rpm         | 623 kB     00:00
(49/55): redhat-logos-60.0.14-12.el6.centos.noarch.rpm   |  15 MB     00:12
(50/55): selinux-policy-3.7.19-155.el6_3.4.noarch.rpm    | 1.3 MB     00:01
(51/55): selinux-policy-targeted-3.7.19-155.el6_3.4.noar | 2.6 MB     00:02
(52/55): sudo-1.7.4p5-13.el6_3.x86_64.rpm                | 423 kB     00:00
(53/55): tzdata-2012c-3.el6.noarch.rpm                   | 441 kB     00:00
(54/55): udev-147-2.42.el6.x86_64.rpm                    | 361 kB     00:00
(55/55): xulrunner-10.0.7-1.el6.centos.x86_64.rpm        |  12 MB     00:09
--------------------------------------------------------------------------------
Total                                           1.0 MB/s | 139 MB     02:15
warning: rpmts_HdrFromFdno: Header V3 RSA/SHA1 Signature, key ID c105b9de: NOKEY
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6
Importing GPG key 0xC105B9DE:
 Userid : CentOS-6 Key (CentOS 6 Official Signing Key) <centos-6-key@centos.org>
 Package: centos-release-6-3.el6.centos.9.x86_64 (@anaconda-CentOS-201207061011.
x86_64/6.3)
 From   : /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6
Is this ok [y/N]: 
```

Figure 116 – Yum Prompt to Import GPG Key

8. The ncurses-devel package and Git SCM tool will be installed next using the package manager and when prompted accept the download and installation of all recommended packages. When prompted, accept the defaults to install all packages.

```
$ sudo yum install ncurses-devel git
```

9. If you plan on using this as your own development machine to submit patches, configure Git with your contact info and customize your preferences. By setting your name and email address, they will be embedded into any of the commits that are generated from this system.

   Note: Setting the identity only needs to be done once and Git will always use this information unless overridden with a different name or e-mail address for specific projects. To perform the override, repeat these same commands but remember to omit the --global flags when in the target project.

```
$ git config --global user.name "<your name>"
$ git config --global user.email "<your email>"
```

10. Set your editor preference (default is vi or vim) if desired. On this reference system, the vi editor will be used.

```
$ git config --global core.editor vi
```

11. If there is a preference for a particular diff tool used to resolve merge conflicts, this should also be set here. On this reference system, the vimdiff tool will be used.

```
$ git config --global merge.tool vimdiff
```

12. In order to use Sourcery CodeBench on an x86 64-bit Linux host system, the 32-bit system libraries must be installed. The 32-bit libraries are available as a series of packages that can be installed using yum on the command line. When prompted, accept the defaults to install all packages.

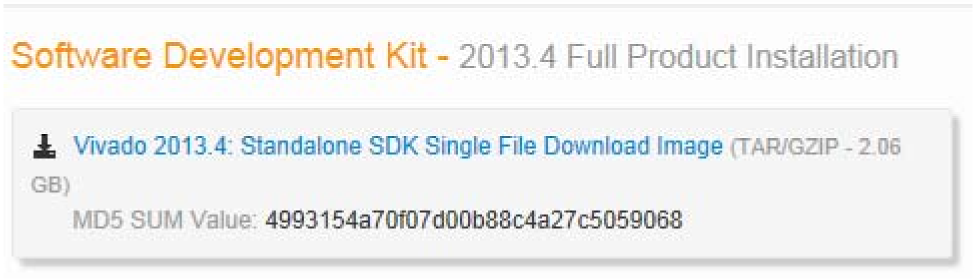
```
$ sudo yum install glibc-devel.i686 gtk2-devel.i686 \
libcanberra.i686 \
libcanberra-gtk2.i686 PackageKit-gtk-module.i686 \
GConf2.i686 ncurses-libs.i686 xulrunner.i686
```

13. The Sourcery CodeBench cross toolchain is now part of the Xilinx SDK, and is not available as a standalone installer. Download the standalone SDK installer from:

http://www.xilinx.com/support/download/index.htm

Note: Downloading the installer file from the URL above requires an account login to the Xilinx website so a web browser such as Firefox must be used.

Select the Standalone SDK from the download list (unless you also wish to install the Vivado Design Suite).


Software Development Kit - 2013.4 Full Product Installation

Vivado 2013.4: Standalone SDK Single File Download Image (TAR/GZIP - 2.06 GB)

MD5 SUM Value: 4993154a70f07d00b88c4a27c5059068

Figure 117 – Standalone SDK Installation

14. Create a new directory **/home/training/temp**.

```
$ mkdir /home/training/temp
```

Save or move the tarball to the **/home/training/temp** folder of your development system.

15. Decompress the SDK tarball.

```
$ cd /home/training/temp

$ tar -xvf Xilinx_SDK_2013.4_1210_1.tar
```

16. Change into the installation directory and run the installer.

```
$ cd Xilinx_SDK_2013.4_1210_1

$ sudo ./xsetup
```

Follow the installer instruction screens, accepting all License Agreements and selecting all defaults. You may choose to install the Cable Drivers if you plan to download via JTAG from this VM to a target board, but this is not required for the Ubuntu tutorial.

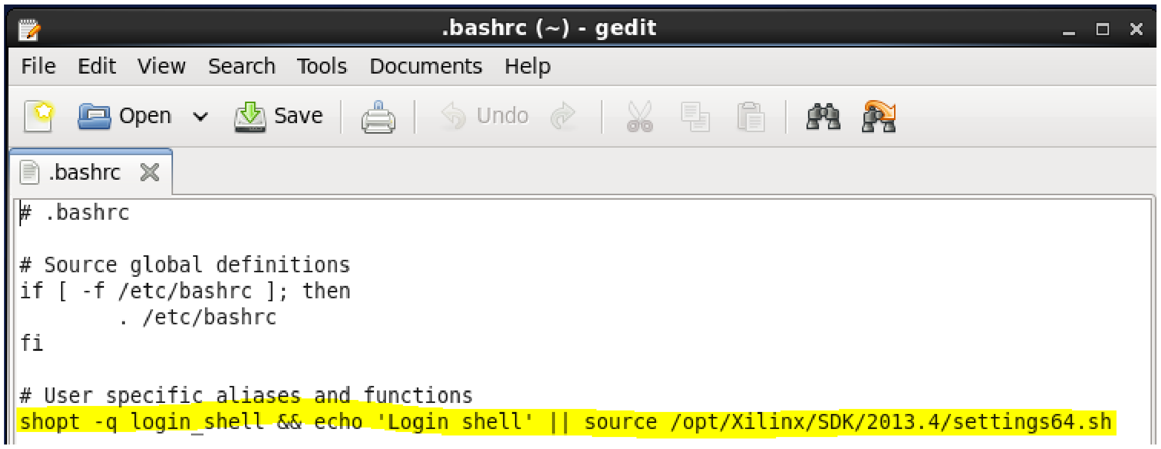
17. When the installer completed, click the **Finish** button and enter the command below to update your $PATH variable. You may optionally delete the entire **temp** directory at this point.

```
$ source /opt/Xilinx/SDK/2013.4/settings64.sh
```

If you would like this command to execute each time you open a new terminal window, you can place it in your hidden **.bashrc** file in the **/home/training** directory. However, it must be placed in a conditional statement, because if .bashrc is referenced during the login shell (as is often the case), it will prevent X11 applications such as File Manager from executing. You may use gedit to add the following line to your .bashrc file.

shopt -q login_shell && echo 'Login shell' || source /opt/Xilinx/SDK/2013.4/settings64.sh

```
$ cd ~

$ gedit .bashrc
```



```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# User specific aliases and functions
shopt -q login_shell && echo 'Login shell' || source /opt/Xilinx/SDK/2013.4/settings64.sh
```

Figure 118 – Editing the .bashrc File

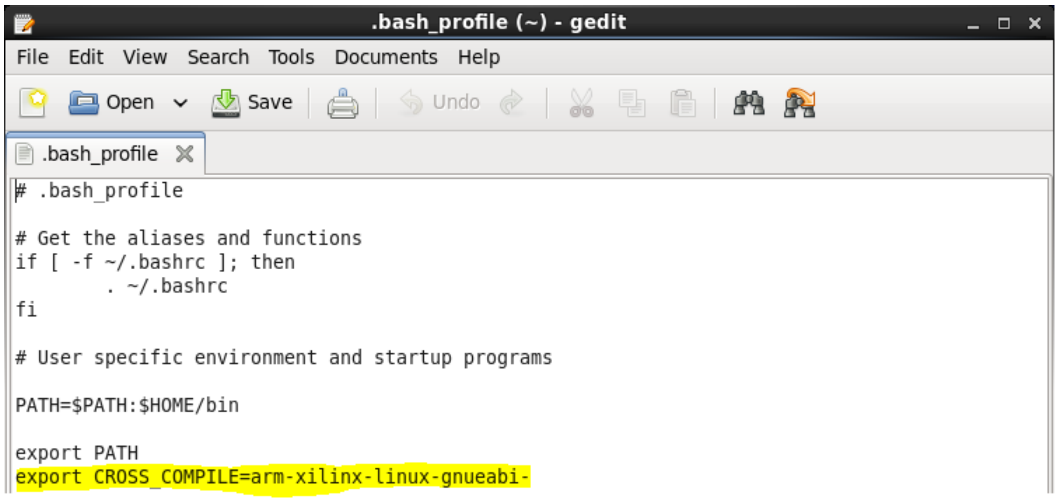
18. Many programs refer to environment variable CROSS_COMPILE to invoke the GNU tools that are used to build the software package for an embedded target.

    Use the **gedit** text editor to open the hidden file **.bash_profile** file found in the **/home/training** directory**.**

```
$ cd ~

$ gedit .bash_profile
```

Add the following line to the bash shell user profile. You may also optionally add the source command from the previous step as shown.

```
export CROSS_COMPILE=arm-xilinx-linux-gnueabi-
```


.bash_profile (~) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
.bash_profile

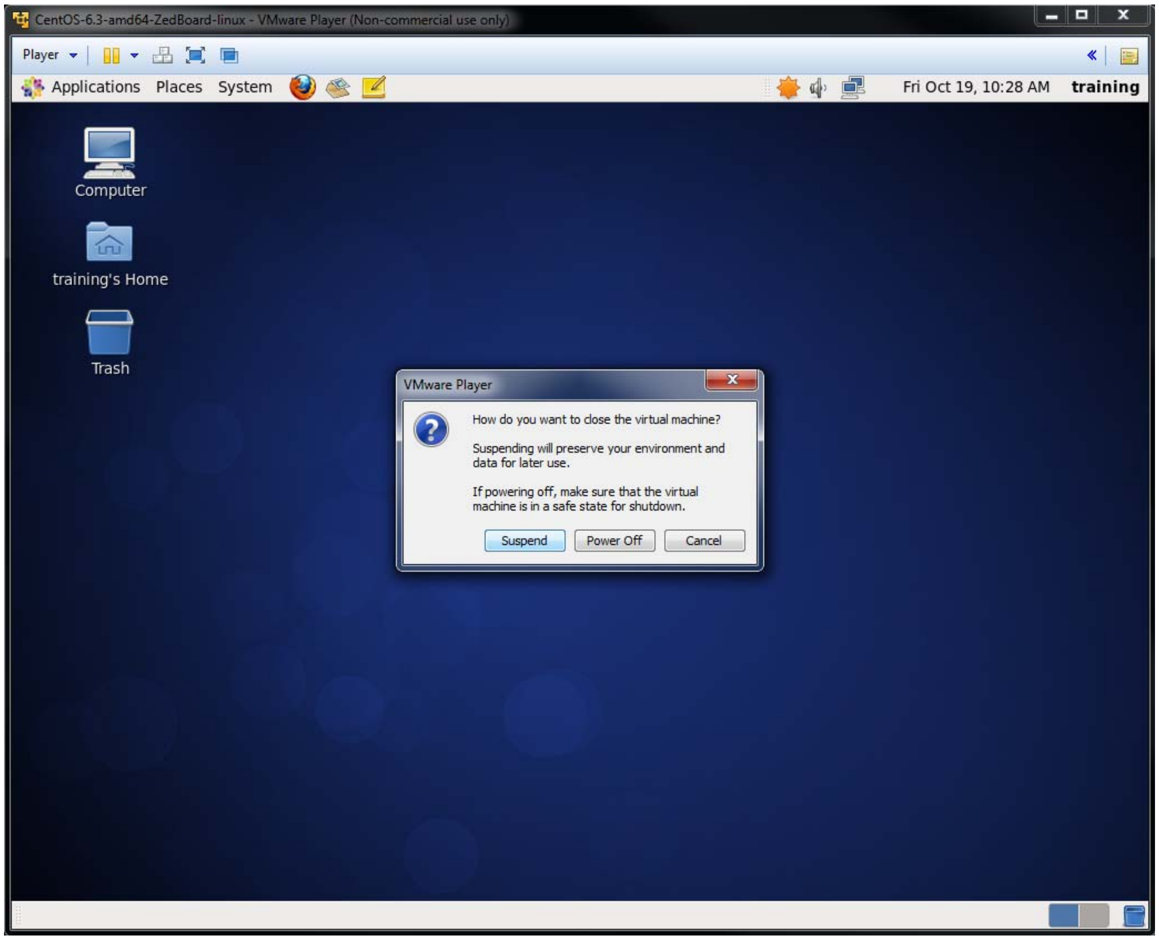
```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin

export PATH
export CROSS_COMPILE=arm-xilinx-linux-gnueabi-
```

Figure 119 – Editing the .bash_profile File

Save the changes made to the bash shell user profile and exit **gedit** using the **File→Quit** menu option. Close the terminal window and open a new terminal window. The .bashrc file executes each time a new terminal window is opened so the environment is automatically established.

19. This document uses the Gnome Partition Utility to prepare the SD/microSD for booting. To install this utility on CentOS:
    a. Go to **Appendix IV - Installing RHEL EPEL Repo on Centos 6.x** and follow the instructions to install the RHEL libraries. Then at your Terminal prompt enter:
        i. **sudo yum install gparted**
        ii. Respond 'y' to all queries

20. This completes the setup of the virtual machine. If you wish to save your virtual machine work at any point in time, the virtual machine operating system can be suspended and the virtual machine window closed. To do this, click on the window close **X** button in the upper right hand corner of the virtual machine window and click on the **Suspend** button when prompted. This will close the virtual machine window but it will also persist the state that the CentOS desktop is left in so that your work can be resumed by re-launching the virtual machine.


CentOS-6.3-amd64-ZedBoard-linux - VMware Player (Non-commercial use only)
Player
Applications Places System
Fri Oct 19, 10:28 AM training
Computer
training's Home
Trash
VMware Player
Suspend
Power Off
Cancel

How do you want to close the virtual machine?

Suspending will preserve your environment and data for later use.

If powering off, make sure that the virtual machine is in a safe state for shutdown.

Figure 120 – Suspending the Virtual Machine

Appendix III - Installing Ubuntu as the VMWare OS

General Instruction:

Obtain the Ubuntu installation media image(s) from http://www.ubuntu.com/download and create a new VMware virtual machine using the downloaded Ubuntu media as the source image.

Important

Installation of a 64-bit guest operating system is only recommended on a 64-bit host operating system. Also, virtualization technology (AMD-V or VT-x) must be supported by the processor and enabled in the BIOS.

The iso file used to prepare this documentation is:

ubuntu-12.04.2-desktop-amd64.iso

Step-by-Step Instructions:

1. Launch VMware player and select menu **Player → File → New Virtual Machine**, or select the action item **Create a New Virtual Machine** at the right of the window.

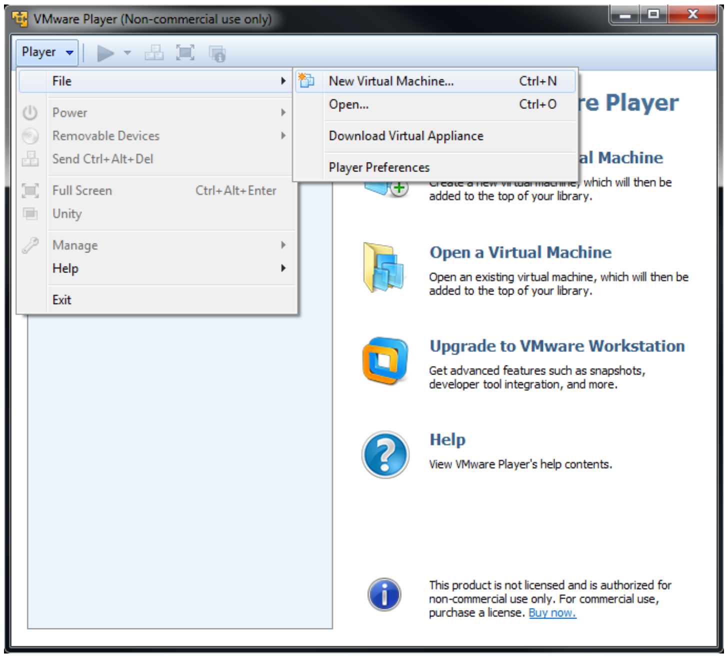
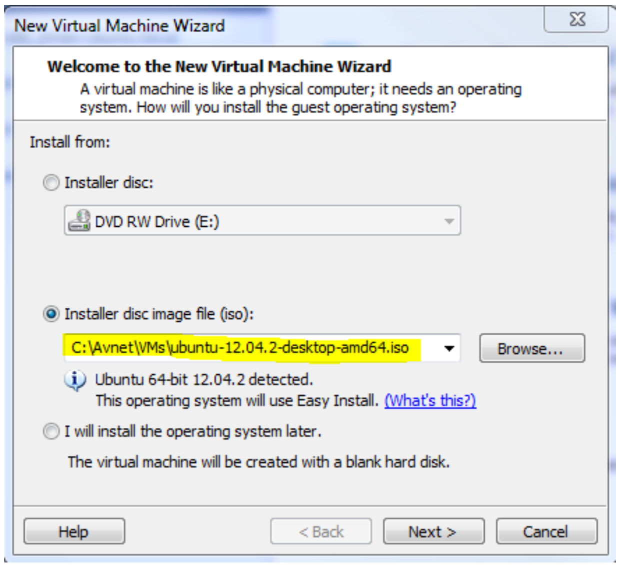

VMware Player (Non-commercial use only)
Player
File
Power
Removable Devices
Send Ctrl+Alt+Del
Full Screen Ctrl+Alt+Enter
Unity
Manage
Help
Exit
New Virtual Machine... Ctrl+N
Open... Ctrl+O
Download Virtual Appliance
Player Preferences
re Player
al Machine
which will then be added to the top of your library.
Open a Virtual Machine
Open an existing virtual machine, which will then be added to the top of your library.
Upgrade to VMware Workstation
Get advanced features such as snapshots, developer tool integration, and more.
Help
View VMware Player's help contents.
This product is not licensed and is authorized for non-commercial use only. For commercial use, purchase a license. Buy now.

Figure 121 – Using VMware Player to Create a New Virtual Machine

2. In the *New Virtual Machine Wizard* select the **Installer disc image file (iso)** option, locate your Ubuntu installation media image file, and click the **Next** button.



Welcome to the New Virtual Machine Wizard
A virtual machine is like a physical computer; it needs an operating system. How will you install the guest operating system?

Install from:

○ Installer disc:
DVD RW Drive (E:)

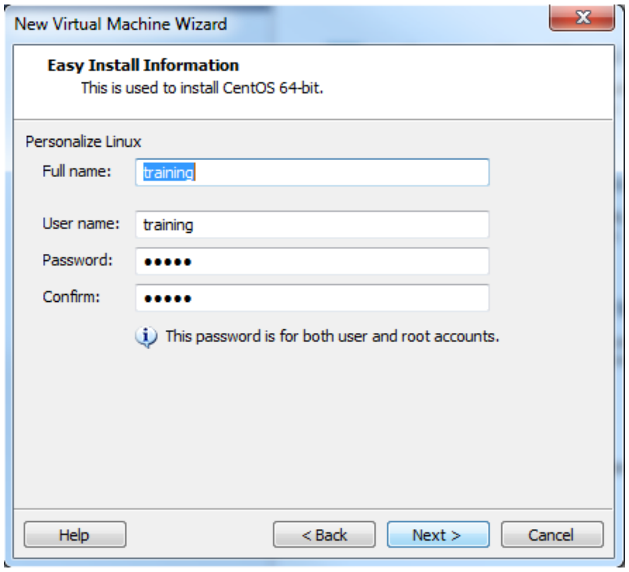
● Installer disc image file (iso):
C:\Avnet\VMs\ubuntu-12.04.2-desktop-amd64.iso

Ubuntu 64-bit 12.04.2 detected.
This operating system will use Easy Install. (What's this?)

○ I will install the operating system later.
The virtual machine will be created with a blank hard disk.

Figure 122 – Selecting the Install Media for New Virtual Machine

3. Under *Easy Install Information*, enter in information that is appropriate to your own install and then click the **Next** button. The User Name and password will be the primary login account for your development system. For this documentation, the name "training" is used.



Easy Install Information
This is used to install CentOS 64-bit.

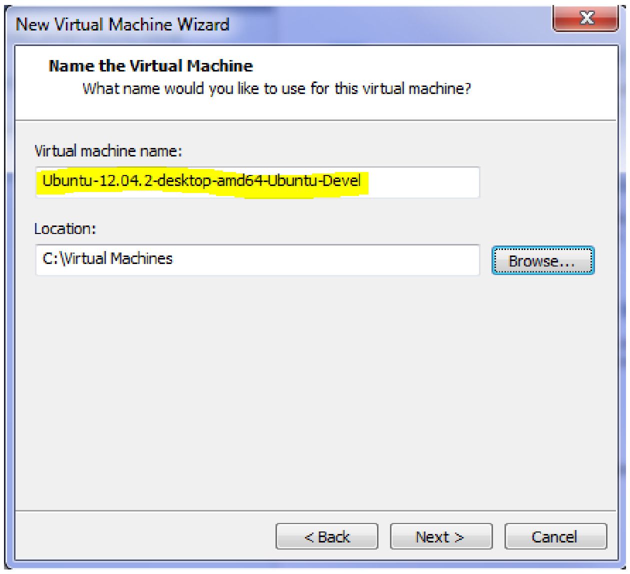
Personalize Linux

Full name:	training
User name:	training
Password:	●●●●●
Confirm:	●●●●●

This password is for both user and root accounts.

Figure 123 – Ubuntu Personalization

4. In the *Name the Virtual Machine* window, name the virtual machine **Ubuntu-12.04.2-desktop-amd64-Ubuntu-Devel**, select an appropriate path, and click the **Next** button.


New Virtual Machine Wizard

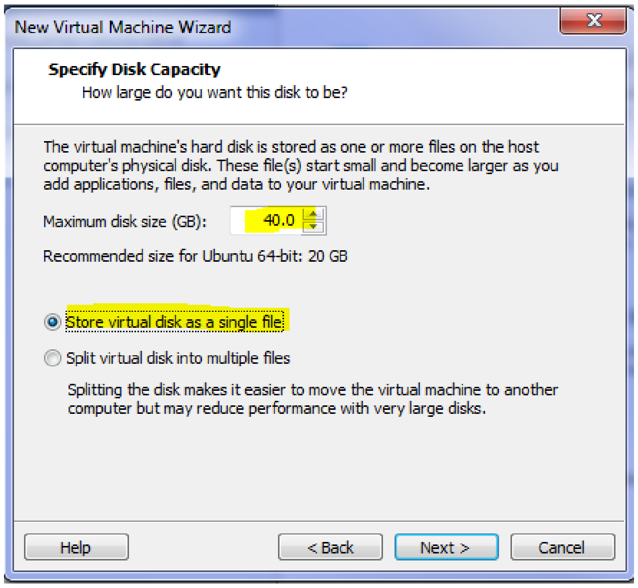
Name the Virtual Machine
What name would you like to use for this virtual machine?

Virtual machine name:
Ubuntu-12.04.2-desktop-amd64-Ubuntu-Devel

Location:
C:\Virtual Machines

Figure 124 – Naming the New Virtual Machine

5. In the *Specify Disk Capacity* window, use a size of **40.0 GB** or more to give plenty of room for tools you may want to install later. Select the option **Store virtual disk as a single file**, and then click the **Next** button.


New Virtual Machine Wizard

Specify Disk Capacity
How large do you want this disk to be?

The virtual machine's hard disk is stored as one or more files on the host computer's physical disk. These file(s) start small and become larger as you add applications, files, and data to your virtual machine.

Maximum disk size (GB): 40.0

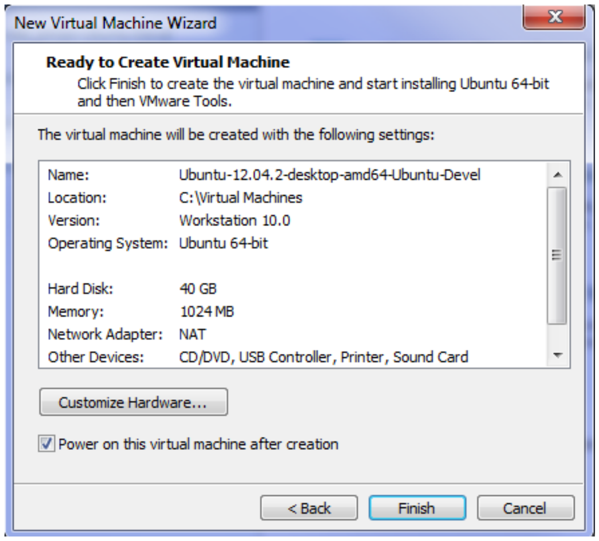
Recommended size for Ubuntu 64-bit: 20 GB

[✓ Store virtual disk as a single file]
Split virtual disk into multiple files
Splitting the disk makes it easier to move the virtual machine to another computer but may reduce performance with very large disks.

Figure 125 – Virtual Machine Disk Capacity

6. In the *Ready to Create Virtual Machine* window, verify the settings for the machine. Be certain that at least 1024MB is allocated for the virtual machine memory and then click the **Finish** button.

   The virtual machine will be created and Ubuntu installed in the new machine. The virtual machine creation and install will take 20-40 minutes to complete depending upon the host machine performance. If they security software installed in the PC, there may be additional prompts to allow changes to the system during the course of the virtual machine creation.



New Virtual Machine Wizard

Ready to Create Virtual Machine

Click Finish to create the virtual machine and start installing Ubuntu 64-bit and then VMware Tools.

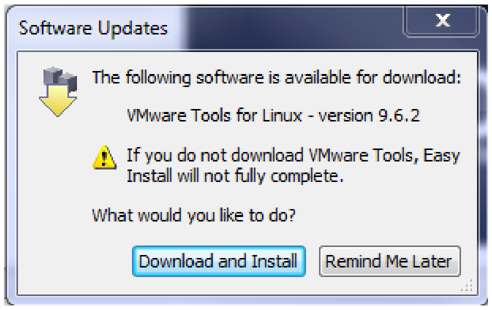
The virtual machine will be created with the following settings:

Name:	Ubuntu-12.04.2-desktop-amd64-Ubuntu-Devel
Location:	C:\Virtual Machines
Version:	Workstation 10.0
Operating System:	Ubuntu 64-bit
Hard Disk:	40 GB
Memory:	1024 MB
Network Adapter:	NAT
Other Devices:	CD/DVD, USB Controller, Printer, Sound Card

☑ Power on this virtual machine after creation

Figure 126 – Virtual Machine Settings Review

If prompted to download and install *VMware Tools for Linux*, click the **Download and Install** button to allow the download to complete and the VMware Update Launcher to run on the host OS. The latest version of VMware Tools is recommended to enhance the performance of the virtual machine guest operating system and improve virtual machine management.



Software Updates

The following software is available for download:

VMware Tools for Linux - version 9.6.2

If you do not download VMware Tools, Easy Install will not fully complete.

What would you like to do?

Figure 127 – Download and Install VMware Tools for Linux Prompt

7. Once the installation process is complete, login with your user account information specified earlier to view the Ubuntu desktop.

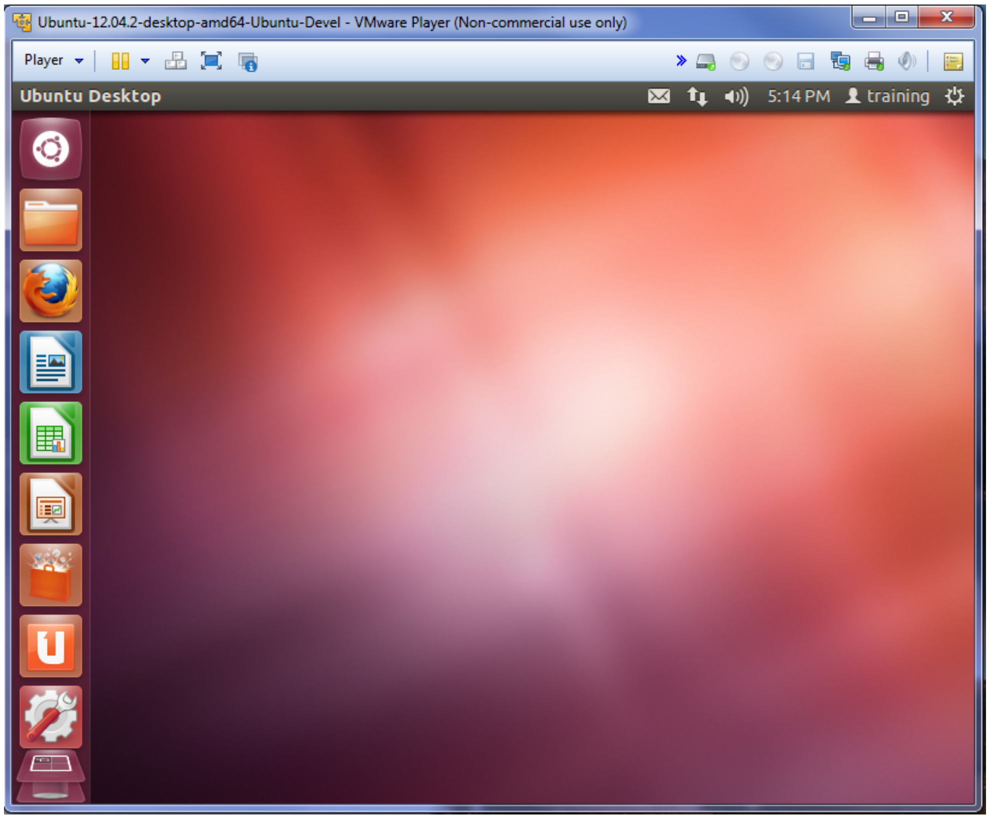

Ubuntu-12.04.2-desktop-amd64-Ubuntu-Devel - VMware Player (Non-commercial use only)
Player
Ubuntu Desktop
5:14 PM
training

Figure 128 – New Virtual Machine Following Installation

Set up the Build Environment under Ubuntu

General Instruction:

Install the Git SCM tools followed by the Linaro maintainer's tool chain.

Step-by-Step Instruction:

1. In the Ubuntu guest operating system, click on the Dashboard icon in the Launch panel and type the first few characters of **terminal** into the text box. Click on the Terminal icon to open a new command line window.

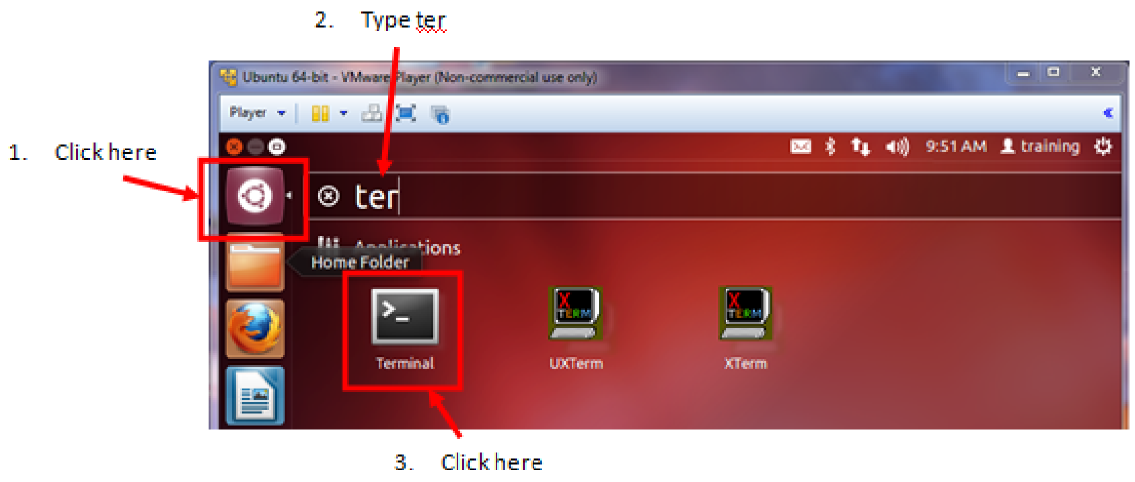
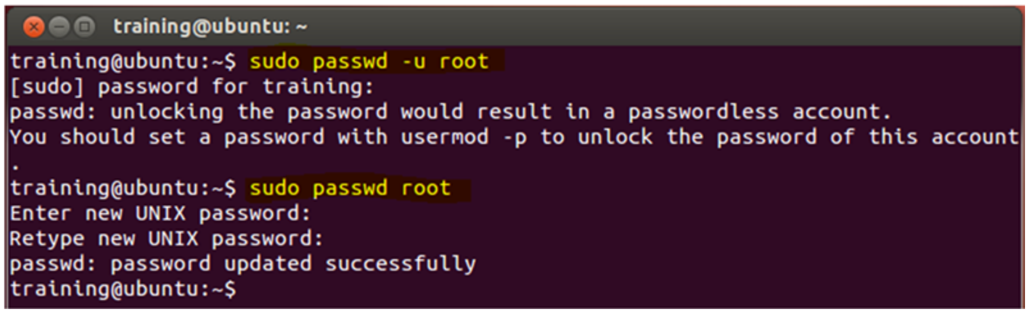


Figure 129 – Launch Ubuntu Terminal Window

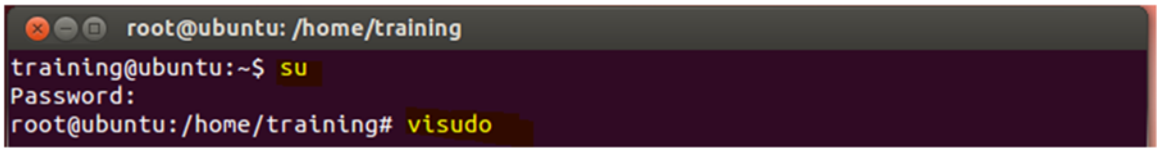
2. In Ubuntu, the root user is disabled by default. You can enable it, and set the root password, as shown below. For the purposes of this tutorial, it is recommended the su password be set to **Avnet**.



```
training@ubuntu: ~
training@ubuntu:~$ sudo passwd -u root
[sudo] password for training:
passwd: unlocking the password would result in a passwordless account.
You should set a password with usermod -p to unlock the password of this account
.
training@ubuntu:~$ sudo passwd root
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
training@ubuntu:~$
```

Figure 130 – Enable Ubuntu root Account

3. Take on root privileges by running the superuser elevation command **su**. Use the password you set in the previous step. Use the visudo text editor to modify the /etc/sudoers file.

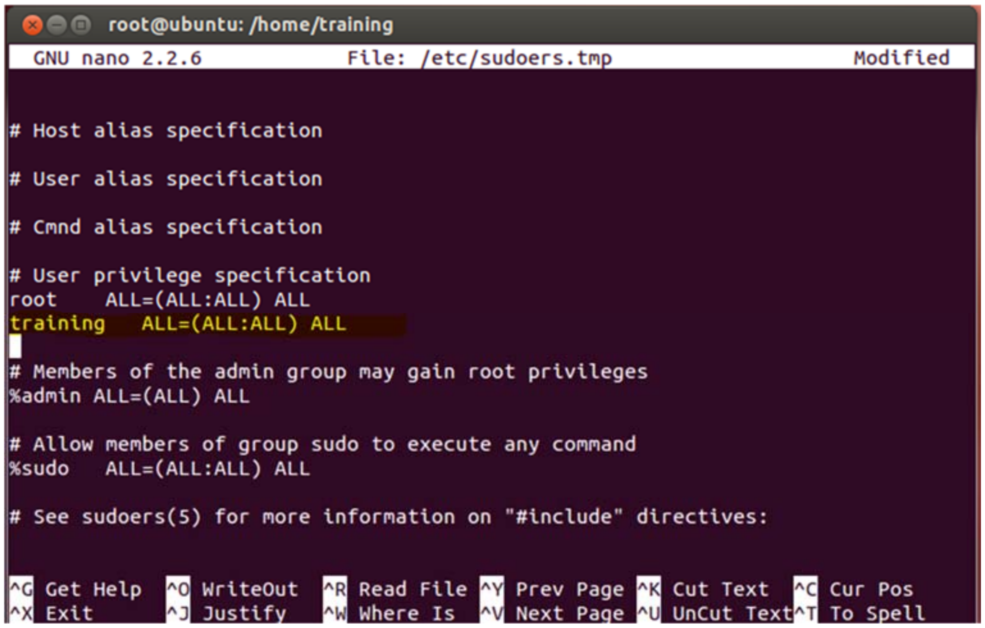


```
root@ubuntu: /home/training
training@ubuntu:~$ su
Password:
root@ubuntu:/home/training# visudo
```

Figure 131 – Elevate to Superuser Privileges and Execute Visudo

4. Add the training user to the sudoers list by inserting the following line to the users section of the sudoers file as shown in the Figure below. The users section is located near the end of the file.

```
training      ALL=(ALL:ALL)   ALL
```



```
root@ubuntu: /home/training
  GNU nano 2.2.6             File: /etc/sudoers.tmp                     Modified


# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
training   ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:


^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Figure 132 – Add training Account to sudoers File

5. Exit the visudo editor by typing **<CTRL>-X** and respond **Y** when asked if you want to save the file. Hit the **<ENTER>** key to accept the default format.

6. The training user will now have sufficient privileges to perform system tasks using the sudo command. Exit the superuser mode.

```
# exit
```

7. Notice how the command prompt has just changed from a # character back to a normal $ character. Next, the system should be updated with the latest

packages. Use the package manager to install the system updates from the appropriate mirrors (of course you must have Internet access for this) and when prompted accept the download and installation of all recommended packages. These updates can take a few minutes and may present several user prompts before completion.

For all user prompts, accept the action by pressing the **Y** key followed by the **<ENTER>** key.

```
$ sudo apt-get update
$ sudo apt-get install
```

8. The ncurses-devel package and Git SCM tool will be installed next using the package manager. When prompted, accept the download and installation of all recommended packages.

```
$ sudo apt-get install libncurses5-dev
$ sudo apt-get install git
```

9. If you plan on using the VM as your own development machine to submit patches, configure Git with your contact information and customize your preferences. By setting your name and email address, they will be embedded into any of the commits that are generated from this system.

   Note: Setting the identity only needs to be done once and Git will always use this information unless overridden with a different name of e-mail address for specific projects. To perform the override, repeat these same commands but remember to omit the –global flags when in the target project.

```
$ git config --global user.name "<your user id here>"
$ git config --global user.email "<your e-mail address here>"
```

10. Set your editor preference (default is vi or vim) if desired.

```
$ git config --global core.editor <"your preferred editor command here">
```

11. If there is a preference for a particular diff tool used to resolve merge conflicts, this should also be set here. In this example, the vimdiff tool is specified.

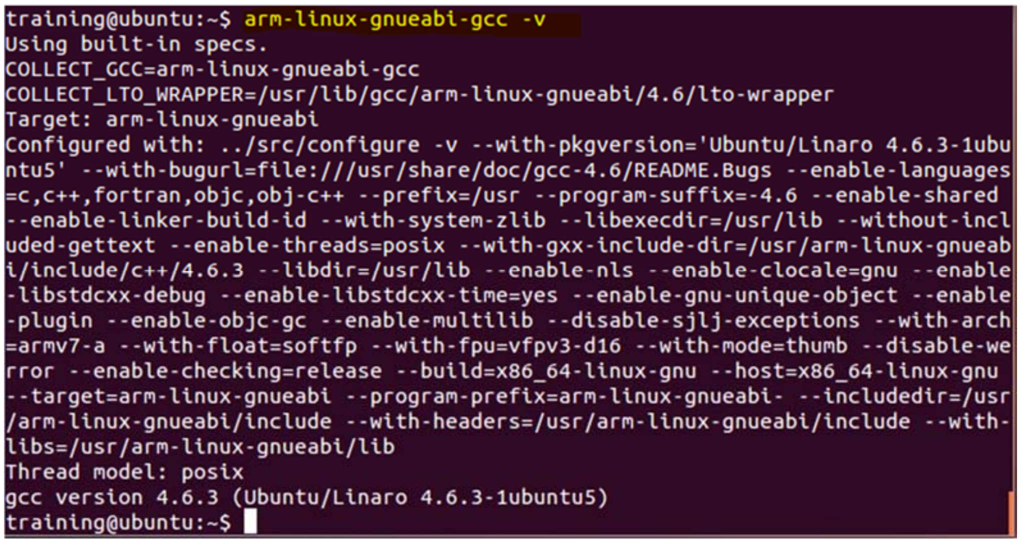
```
$ git config --global merge.tool vimdiff
```

12. Install the Linaro ARM tool chain. When prompted, accept the defaults to install all packages.

```
$ sudo apt-get install gcc-arm-linux-gnueabi
```

13. To validate that your tool chain installation is complete, request the version information which should provide a response similar to the one shown in the figure below.

```
$ arm-linux-gnueabi-gcc -v
```



```
training@ubuntu:~$ arm-linux-gnueabi-gcc -v
Using built-in specs.
COLLECT_GCC=arm-linux-gnueabi-gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/arm-linux-gnueabi/4.6/lto-wrapper
Target: arm-linux-gnueabi
Configured with: ../src/configure -v --with-pkgversion='Ubuntu/Linaro 4.6.3-1ubu
ntu5' --with-bugurl=file:///usr/share/doc/gcc-4.6/README.Bugs --enable-languages
=c,c++,fortran,objc,obj-c++ --prefix=/usr --program-suffix=-4.6 --enable-shared
--enable-linker-build-id --with-system-zlib --libexecdir=/usr/lib --without-incl
uded-gettext --enable-threads=posix --with-gxx-include-dir=/usr/arm-linux-gnueab
i/include/c++/4.6.3 --libdir=/usr/lib --enable-nls --enable-clocale=gnu --enable
-libstdcxx-debug --enable-libstdcxx-time=yes --enable-gnu-unique-object --enable
-plugin --enable-objc-gc --enable-multilib --disable-sjlj-exceptions --with-arch
=armv7-a --with-float=softfp --with-fpu=vfpv3-d16 --with-mode=thumb --disable-we
rror --enable-checking=release --build=x86_64-linux-gnu --host=x86_64-linux-gnu
--target=arm-linux-gnueabi --program-prefix=arm-linux-gnueabi- --includedir=/usr
/arm-linux-gnueabi/include --with-headers=/usr/arm-linux-gnueabi/include --with-
libs=/usr/arm-linux-gnueabi/lib
Thread model: posix
gcc version 4.6.3 (Ubuntu/Linaro 4.6.3-1ubuntu5)
training@ubuntu:~$
```

Figure 133 – Installation of Linaro ARM Tool Chain Successful

14. The final step is to edit your default shell profile to set environment variables that identify the Linaro toolchain. You may create a new file in your home directory called .bash_profile with the gedit command.

```
$ gedit .bashrc
```

15. Add the following lines to end of the file.

```
PATH=$PATH:$HOME/bin
export PATH
export CROSS_COMPILE=arm-linux-gnueabi-
```

Save the changes and exit **gedit** using the **File -> Quit** menu option.

16. Close the terminal window. The .bashrc file will execute whenever a new terminal window is opened to restore your build environment variables.

17. This document uses the Gnome Partition Utility to prepare the SD/microSD card for booting. On Ubuntu, install this utility by entering the following commands at the Terminal prompt:
    - **a. sudo apt-get install gparted**
    - b. Respond 'y' to all queries

This completes the setup of the Virtual Machine and the build environment under Ubuntu. If you wish to save your virtual machine work at any point in time, you may close the window with the X button at the upper right and click the **Suspend** button when prompted. The next time you open the VM, the current state will be restored.

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

Installation of these packages is required on CentOS in order to install the GParted Partition Editor.

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://rpms.famillecollet.com/enterprise/remi-
release-6.rpm

$ 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
/etc/yum.repos.d/remi.repo

Enable the remi repository

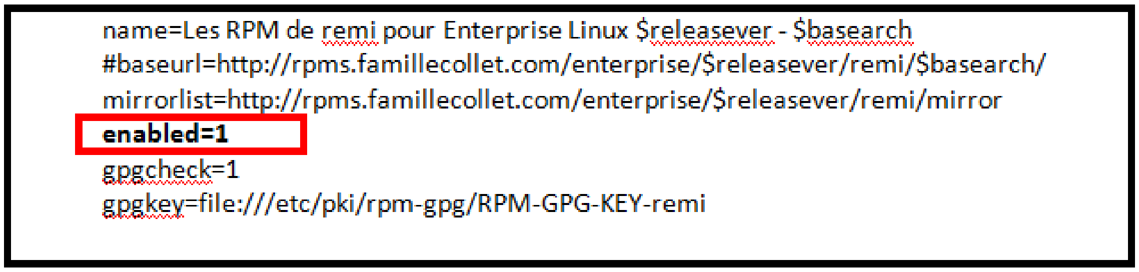
The remi repository provides a variety of up-to-date packages that are useful or are a requirement for many popular web-based services. That means it generally is not a bad idea to enable the remi repositories by default.

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
```

Figure 134 – Tera Term Icon

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

© 2011-2013 Rackspace US, Inc.

Appendix V - Supplied Files

Pre-built binary files for the FAT32 portion of the SD card are included in the event that you encounter problems while performing the tutorial experiments. You may copy these files to your SD/micro SD card to verify that you have your environment is correctly configured.

Using these binaries, you will be able to boot the Linux kernel to the point where it attempts to reference the root file system on the ext4 partition, at which point there will be a kernel panic. You can follow the steps in this tutorial to create an ext4 partition and load the root file system to it, at which point you will have a complete Ubuntu desktop system to test.

The root file system cannot be included due to the size of the system, and further you will need to configure the bootable media with both FAT32 and ext4 partitions before it can be loaded. Detailed steps for obtaining the root file system and extracting it to a properly formatted SD/microSD card are covered in this tutorial document.

adau1761: Contains files used in the audio codec validation tests.

- **adau1761_golden.state**: Advanced Linux Sound Architecture configuration file for the ADAU1761 audio codec.
- **pipeaudio**: Script for testing codec stereo record and playback. Copy this script to the Ubuntu desktop and execute it from a Terminal window. An external audio source is required (see Lab 9, Experiment 2)
- **tonetest**: Script for basic audio tones. Use this script to validate stereo audio playback using the Headphone jack. Copy this script to the Ubuntu desktop and execute it from a Terminal window.

sd_card: Contains the files to run test programs from the SD card:

- **devicetree.dtb**: Compiled hardware map for Ubuntu platform.
- **boot.bin**: Boots hw platform and executes the U-boot loader for Ubuntu.
- **uImage**: Compressed kernel image for Linaro kernel used to access the Ubuntu desktop.

Ubuntu_on_Zynq_Tutorial.pdf: This document.

boot_source[13]**:** Contains files used in the creation of the sd card images.

- **u-boot**: Source files that will eventually be included in the Xilinx GIT repository.
  - **boards.cfg**: Adds the Zynq Mini-ITX to u-boot targets.
  - **zynq_mitx.h**: Add a default u-boot configuration for the Zynq Mini-ITX.
- **system_top.bin**: Bitstream for the hardware platform.
- **u-boot.elf**: Compiled u-boot executable.
- **zynq_fsbl.elf:** Compiled first stage bootloader executable.
- **zynq-mitx.dtsi:** Second level device tree source for Zynq Mini-ITX platform.
- **zynq-mitx-adv7511.dts:** Top level device tree source for Zynq Mini-ITX platform.

[13] Zynq Mini-ITX targets only.

Resources

ZedBoard.org

ZedBoard documentation page
http://www.zedboard.org/documentation/1521

ZedBoard reference designs and tutorials
http://www.zedboard.org/design/1521/11

Zynq Mini-ITX documentation page
http://www.zedboard.org/documentation/2056

Zynq Mini-ITX reference designs and tutorials
http://www.zedboard.org/design/2056/17

ZedBoard Forum Support (English)
http://www.zedboard.org/forums/zed-english-forum

ADI

ADV7511 Product Page
http://www.analog.com/en/audiovideo-products/analoghdmidvi-interfaces/adv7511/products/product.html

ADAU1761 Product Page
http://www.analog.com/en/audiovideo-products/audio-signal-processors/adau1761/products/product.html

Wiki Instruction page
http://wiki.analog.com/resources/fpga/docs/hdl
http://wiki.analog.com/resources/fpga/docs/hdl/vivado
http://wiki.analog.com/resources/fpga/xilinx/kc705/adv7511 (standalone software)

ADI HDL Reference Designs HDL User Guide
http://wiki.analog.com/resources/fpga/docs/hdl/github

ADI Support
http://ez.analog.com/community/fpga

ADV7511 Xilinx Evaluation Boards Reference Designs (ZedBoard)
http://wiki.analog.com/resources/fpga/xilinx/kc705/adv7511

Xilinx Website

Xilinx Design Tools Installation and Licensing Guide
http://www.xilinx.com/support/documentation/sw_manuals/xilinx2013_4/ug973-vivado-release-notes-install-license.pdf

Zynq
http://www.xilinx.com/products/silicon-devices/soc/zynq-7000/index.htm

Xilinx Silicon and Tools Forum Support
http://forums.xilinx.com/

Other Internet Resources

http://www.hdmi.org/
http://en.wikipedia.org/wiki/HDMI
http://www.linaro.org/
http://www.ubuntu.com/

Revision History

Date	Version	Revision
16 Apr 13	01	Initial Draft (EDK/SDK 14.4)
24 May 13	02	Updated for kernel uImage & compatible U-boot
10 July 14	03	Added Zynq Mini-ITX + Updated to Vivado/SDK 2013.4