

# Ethernet on the Zynq ZC706

18-545 Advanced Digital Design



Terence An, Eddie Nolan, Dale Zhang

December 12, 2015

# Chapter 1

## Introduction

This paper is a guide to start building ethernet on the Zynq ZC706 board. It was originally written as the final project for a 18-545 project which didn't complete because they struggled to build an ethernet adapter in programmable logic and have it properly communicate with the Processing System. The intention of this paper is to aid future groups in completing an ethernet adapter, as well as providing the necessary background and deterring groups from fruitless avenues. This guide will expect a very minimal understanding of Vivado because most students in 18-545 have had very limited exposure to Vivado. We will attempt to provide the pertinent references as needed.

That being said, going through Lab 2 in *Vivado Design Suite Tutorial* [4] will probably be the fastest way to understand the work flow. Also, chapter 2 and chapter 4 of *UltraFast Design Methodology Guide for the Vivado Design Suite* [3] will be superbly helpful in learning to use Vivado, especially for using IP in Vivado. Finally, if you still want more details on using IP, you can refer to the Vivado guide on *Designing with IP* [2] and *Designing IP Subsystems Using IP Integrator* [1].

## Chapter 2

# Ethernet Background

# Chapter 3

## Basic Approaches

# Chapter 4

## Ethernet PL Guide

# Chapter 5

## PetaLinux Networking

### 5.0.1 Introduction

This section will focus on the attempts we made to implement the software-firmware stack that Xilinx provides through their application notice XAPP1082 and through a guide on the Xilinx wiki. This involves creating a bootable SD card containing a copy of PetaLinux, Xilinx's embedded Linux distribution, that is patched to support communicating over the SFP Ethernet port, and flashing the FPGA fabric with a corresponding bitstream that implements the hardware Ethernet support. Although our eventual goal was to modify the kernel and firmware to support Ethernet traffic analysis, we ran into bugs that we were unable to fully diagnose and solve.

### 5.0.2 Hyperlinks to Relevant Documentation

- XAPP1082 PDF
- Xilinx Wiki Guides:
  - Zynq Ethernet Guide
  - Zynq BootSetup Guide
- Zynq ZC706 User Guide
- PetaLinux Tools Reference Guide
- Installation Guide for PetaLinux 2014.4 (Not by Xilinx)

### 5.0.3 Supplies

- Hardware:
  - It's necessary to be able to connect to a wired Ethernet connection and write SD cards with the computer you are using for development.
  - If you want to switch between different copies of Petalinux, it's convenient to have multiple SD cards, so you don't have to reflash them repeatedly. The copy of Petalinux we used was only 200MB so large storage capacity is unnecessary.

- It’s helpful to have an Ethernet switch or router so you can test the device without needing access to the full Internet.
- Software:
  - I installed the Petalinux tools on two Ubuntu LTS virtual machines. You should have virtualization software and enough disk space available on your computer for 2 virtual machines.
  - We also were able to use lab computers that were configured to use the version of Vivado that corresponded to the version of XAPP1082 we used.
  - The specific versions that our project used were XAPP1082 version 3, Petalinux and Vivado version 2014.4, Xubuntu 14.04 LTS for installing the Petalinux tools. Our lab machines ran RHEL.

## 5.1 Using Petalinux

### 5.1.1 Creating Boot Media

The Zynq Boot Setup Guide linked above demonstrates the way to format bootable SD cards. You need to create 2 partitions, `boot` and `root`, and copy 2 files, `BOOT.bin` and `image.ub`, onto the `'boot'` partition, but the partition table needs to be configured in a specific way using `fdisk` commands. The relevant part of the guide is the section under “SD Boot”; ignore the sections that describe JTAG and QSPI booting. You can find relevant precompiled `BOOT.bin` and `image.ub` files in the XAPP1082 software release: `$XAPP_HOME/ready_to_test/pl_eth_noCso`.

### 5.1.2 Building XAPP1082’s Petalinux Configuration

The version of the Petalinux Tools that we used wouldn’t install project properly on a 64-bit virtual machine because of issues with 32-bit library support that we were unable to resolve. However, the Xilinx SDK, which is required to create the boot image after the kernel has been compiled, only supports 64-bit versions of Linux. We were able to get around this by creating a 32-bit VM for creating the project and building the kernel, and a separate 64-bit VM for creating the boot image.

#### Setting up the 32-bit VM

For Ubuntu, the Petalinux Tools Reference Guide required the following packages to be installed:

```
tofrodo iproute gawk gcc git-core make net-tools libncurses5-dev tftpd
zlib1g-dev flex bison
```

The Petalinux Tools are distributed as a self-extracting archive/installer. The Xilinx Wiki Zynq Ethernet Guide contains a download link. Both the Petalinux Tools and the XAPP1082 software release should be extracted. The provided shell scripts help to automate steps that are described in the Wiki guide. The directories on the first lines should be modified to reflect the user’s configuration.

### Setting up the 64-bit VM

To set up the 64-bit VM, you need to install all the same packages as the 32 bit VM needs. In addition to extracting the Petalinux installer, you also need to install the Xilinx SDK that corresponds to your version of Petalinux. After you have finished configuring and building the project on the 32-bit VM, you can copy the `xapp1082_pl_eth` project directory to the 64-bit VM, then use the provided shell script to create a bootable image (more detail is available in the wiki guide). The required files for creating a bootable SD card will be in the project directory in the subdirectory `images/linux`.

### Note on Petalinux Setup

In an attempt to diagnose the main problem that we encountered, we needed to introduce a change to the Xilinx ethernet driver so that it would print debug information. We did this by text-editing the patch file that adds driver support to the kernel, although there are many other ways of modifying the Petalinux configuration and software that are described in more detail in the documentation. If there is no need to customize Petalinux in any way, precompiled images are provided in the XAPP1082 software release, and the Petalinux Tools don't need to be used.

### 5.1.3 Connecting to the UART of the ZYNQ ZC706

### 5.1.4 Petalinux Networking Setup and CMU Netreg

### 5.1.5 Bugs and Roadblocks

### 5.1.6 Other Notes

- Make sure jumper J17 is set (see XAPP1082 PDF)

### 5.1.7 Code Listing

#### Shell Scripts for the 32-bit VM

```
1_create_project_config_kernel.sh
```

```
#!/bin/bash
XAPP_HOME=~/.545/xapp1082_2014_4
source ~/.545/petalinux-v2014.4-final/settings.sh

cd $PETALINUX
petalinux-create -t project -s $XAPP_HOME/software/petalinux
    /bsp/xapp1082_pl_eth.bsp
cd $PETALINUX/xapp1082_pl_eth
petalinux-config
```

```
2_apply_patch_build_kernel.sh
```

```
#!/bin/bash
XAPP_HOME=~/.545/xapp1082_2014_4
source ~/.545/petalinux-v2014.4-final/settings.sh
```



```
cd $PETALINUX/xapp1082_pl_eth/build/linux/kernel/download/  
linux-xlnx/  
git am $XAPP_HOME/software/patch/0001-ethernet-xilinx-Add-  
XAPP1082-support.patch  
cd $PETALINUX/xapp1082_pl_eth  
cp subsystems/linux/configs/kernel/xapp1082_defconfig  
subsystems/linux/configs/kernel/config  
petalinux-build -v
```

### Shell Script for the 64-bit VM

3\_create\_boot\_image.sh

```
#!/bin/bash  
source ~/545/petalinux-v2014.4-final/settings.sh  
source /opt/Xilinx/SDK/2014.4/settings64.sh  
XAPP1082_PL_ETH_PROJECT_DIRECTORY=$PETALINUX/xapp1082_pl_eth  
  
cd $XAPP1082_PL_ETH_PROJECT_DIRECTORY/images/linux  
petalinux-package --boot --fsbl=zynq_fsbl.elf --fpga=  
$PETALINUX/xapp1082_pl_eth/subsystems/linux/hw-  
description/pl_eth_sfp.bit --u-boot
```

# Chapter 6

## Alternate Approaches

# Chapter 7

## More on PL

# Chapter 8

## Miscellaneous

### 8.1 Personal Statements

#### 8.1.1 Dale Zhang

For me, this class was challenging for a multitude of reasons. First off, I hadn't touched Verilog since I took 18-240 a few years ago, and I also wasn't very comfortable with HDL. In addition, in the context of our project, I had very little knowledge on ethernet and networking, so I had to learn a lot about how our project worked as I went along.

Over the course of the semester, there are definitely a few things I wish I had done differently. Since Terence and Eddie were both far more knowledgeable about Ethernet and networking, I often took a backseat to them when the group was making decisions. However, at some points in the semester, instead of asking for their help understanding some of the concepts driving our design, I would try to do it myself, without very much success. This led to me spending far more time on some tasks than I should've. This definitely limited my effectiveness as a team member.

Another mistake we made as a team was underestimating how much work actually needed to go into this project. Towards the beginning of the semester, we didn't put in much lab time outside of class periods and mandatory lab time. It first really caught up to us around mid semester with the first status meeting, where we saw how far behind we were, and how much more time we would need to commit for the rest of the semester.

Some advice I'd have for anyone planning to pursue an FPGA Ethernet project in the future is not to spend too much time trying to do research, and to start actually working on the board as soon as possible. In addition, ethernet on FPGA is not very well documented, and much of the documentation available is incorrect or incomplete.

For the class in general, it's definitely better to spend the long hours working on your project earlier in the semester, before your other classes have started to pick up. In addition, at the beginning of the semester, try and pick a project that you can be passionate about and that you would really like to see succeed. At times, I felt very unmotivated to go in and work on the project simply because I wasn't particularly excited about our final product.

To all future students reading this, good luck with the class and have fun!

# Bibliography

- [1] Xilinx. *Designing IP Subsystems Using IP Integrator*. 2015. URL: [http://www.xilinx.com/support/documentation/sw\\_manuals/xilinx2015\\_4/ug994-vivado-ip-subsystems.pdf](http://www.xilinx.com/support/documentation/sw_manuals/xilinx2015_4/ug994-vivado-ip-subsystems.pdf).
- [2] Xilinx. *Designing with IP*. 2015. URL: [http://www.xilinx.com/support/documentation/sw\\_manuals/xilinx2015\\_4/ug896-vivado-ip.pdf](http://www.xilinx.com/support/documentation/sw_manuals/xilinx2015_4/ug896-vivado-ip.pdf).
- [3] Xilinx. *UltraFast Design Methodology Guide for the Vivado Design Suite*. 2015. URL: [http://www.xilinx.com/support/documentation/sw\\_manuals/ug949-vivado-design-methodology.pdf](http://www.xilinx.com/support/documentation/sw_manuals/ug949-vivado-design-methodology.pdf).
- [4] Xilinx. *Vivado Design Suite Tutorial*. 2015. URL: [http://www.xilinx.com/support/documentation/sw\\_manuals/xilinx2015\\_4/ug888-vivado-design-flows-overview-tutorial.pdf](http://www.xilinx.com/support/documentation/sw_manuals/xilinx2015_4/ug888-vivado-design-flows-overview-tutorial.pdf).

Appendix A

Appendix