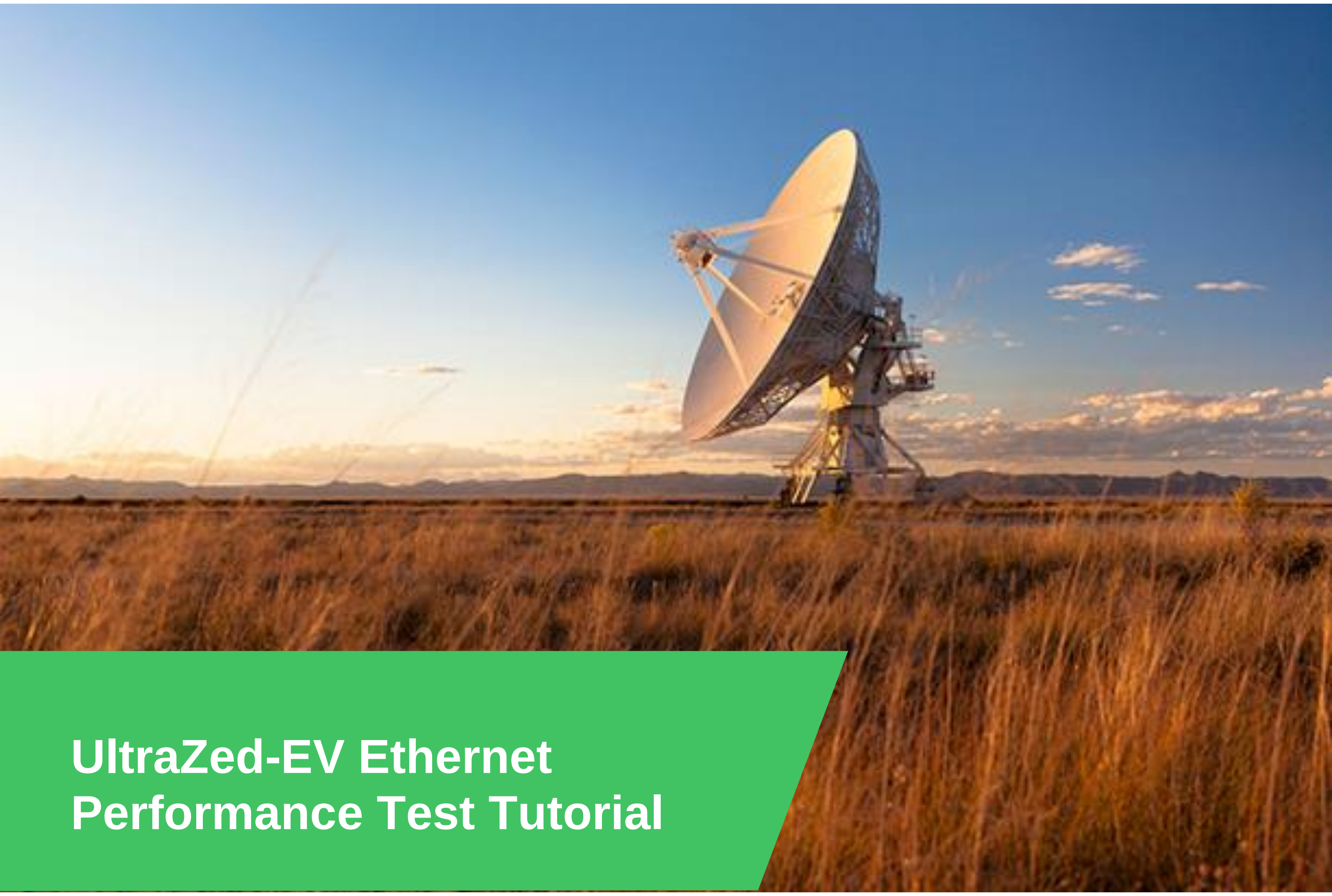


UltraZed-EV Ethernet Performance Test Tutorial

Introduction

Network connectivity is becoming increasingly commonplace with embedded applications. As data throughput demands increase, the ability to objectively measure networking performance when using an embedded operating system becomes essential.

DISCLAIMER: This tutorial is provided for reference/educational purposes only and may not reflect results observed with other test equipment.

There are a number of factors which can impact network performance and throughput in addition to transmission overheads, including latency, receive window size, and system limitations such that the calculated throughput typically does not reflect the maximum achievable throughput.

If several systems are communicating simultaneously, the throughput between any pair of nodes can be substantially lower than nominal network bandwidth.

Design Objectives

This UltraZed tutorial offers system developers an example of how to:

- Target a prebuilt Xilinx release of PetaLinux to UltraZed
- Perform Ethernet performance tests on Zynq platform running a prebuilt open source Linux build created with Xilinx PetaLinux Tools

Experiment Setup

This tutorial builds upon the concepts and lab activities of the Avnet UltraZed Tutorials which cover the use of Xilinx Vivado Design Suite in creating/testing a basic Zynq UltraScale MPSoC hardware platform and running software applications. Please refer back to this reference material on the UltraZed community website for further information on how to configure the underlying UltraZed hardware platform.

The experiments in this tutorial use **iperf3** which is a tool for performing network throughput measurements by testing either TCP or UDP throughput.

For the example test network configuration that was used in this tutorial, see the section **Troubleshooting: Network Connection** further in this document for further information.

The instructions in this tutorial assume that the cross build platform is a CentOS V7.0 installation running natively on an x86-based host. Though other systems may work using the same or similar instructions, those systems are not supported.

Example Design Requirements

Software

The software used to test this reference design is:

- CentOS V7.0 64-bit installer image (Exact version used for this tutorial is CentOS-7.0-1406-x86_64-bin-DVD.iso)
- Xilinx Vivado Design Suite 2017.4 (Free WebPACK license and download from Xilinx website)

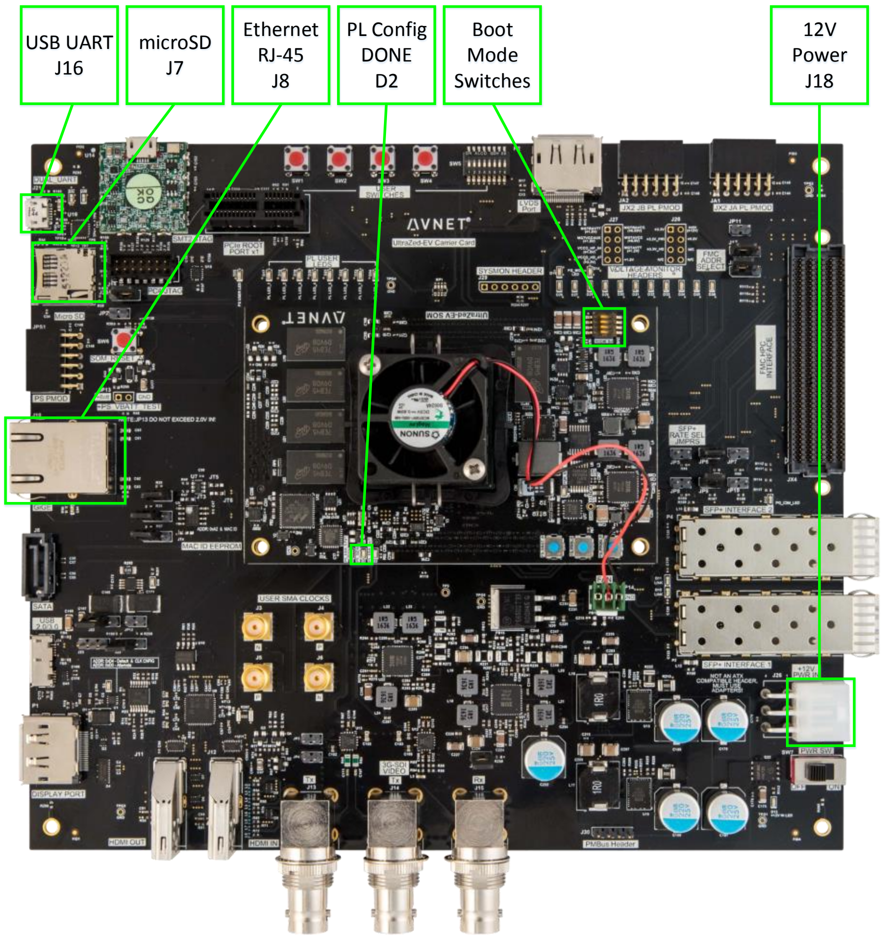
Hardware

The hardware setup used to test this reference design includes:

- Lenovo ThinkPad T420 Laptop
    - Intel® Core i5-2540M CPU - 2.60 GHz
    - 4GB DDR3 Memory
    - Intel® 82579LM Gigabit Ethernet Adapter
    - SD card slot on PC or external USB SD card reader
- Stock Avnet UltraZed-7EV SOM Rev. 1 (AES-ZU7EV-1-SOM-I-G)
- Avnet UltraZed-EV Carrier Card (AES-ZUEV-CC-G)
- USB cable (Type A to Micro-USB Type B)
- 8GB MicroSD card
- Category 5e Ethernet patch cable

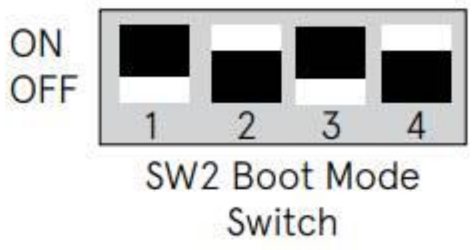
Experiment 0: Setting Up the UltraZed-EV Hardware

Refer to the following figure and perform the following steps to set up the board when using the UltraZed-EV and EV Carrier.



1. Plug the UltraZed-EV SOM onto the EV Carrier Card via JX1/JX2/JX3 connectors and connect the fan to the fan header (JP14) on the EV Carrier Card.

2. Set the UltraZed-EV SOM Boot Mode switch (SW2) (MODE[3:0] = SW2[1:4]) to OFF, ON, OFF, and ON positions (Boot Mode set to SD card, MODE[3:0] = 0x5).



3. Connect the USB-UART port on the EV Carrier Card (J16) to a free USB port on your PC.

4. Connect the 12V power cable, but do not turn on the board yet.

Experiment 1: Setup Linux for UltraZed

The experiments in this tutorial are based upon the Linux build that is provided by Avnet as part of the UltraZed-EV Carrier Card - Out-of-Box Design which is available for download from the ultrazed.org website.

http://avnet.me/ultrazed-ev-reference-designs

1. Copy the **BOOT.BIN** and **image.ub** files from the accompanying archive into the root folder of the microSD card to match the SOM and Carrier you are using.

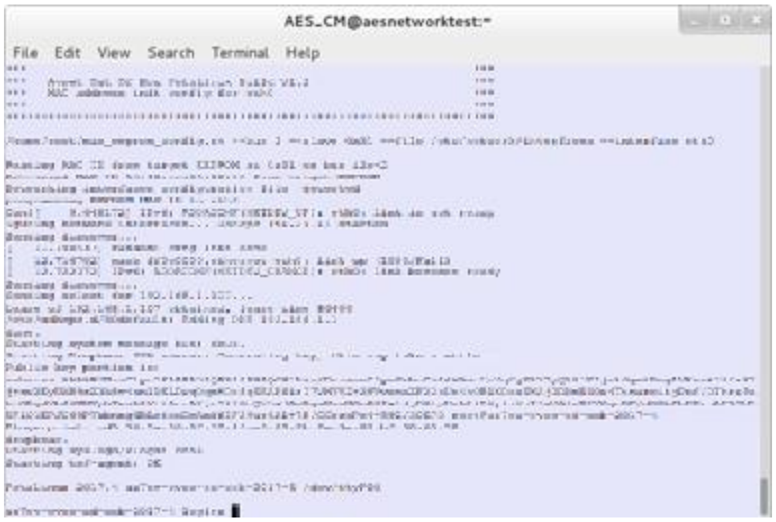
   Replace any existing versions of these files that may already be on the microSD card.

2. Insert the microSD card, prepared using the steps abve, into the UltraZed Carrier microSD card cage (J7).

3. Set the UltraZed Carrier power switch (SW7) to the ON position. The UltraZed system will power on and the Power Good LED should illuminate.

4. Launch a terminal program with the 115200/8/n/1/n settings. For the example output shown here, the minicom terminal was used. For information on setting up minicom to use with the UltraZed USB-UART port, see the section **Appendix II: Troubleshooting Serial Connection** later in this document for further information.

5. You should observe terminal output from U-Boot and then Linux output appear in the **minicom** window.



```
[illegible]
```

Experiment 2: Building iperf3 for Zynq UltraScale+

Now that the Linux can be booted on UltraZed, **iperf3** is provided as part of the Linux build image provided for the OOB design so this experiment is optional.

If you want to be able to configure and build a newer version of **iperf3** for the Zynq UltraScale+ target platform that we will test on the local network, the following experiment can help explain how to do this, otherwise you can proceed to Experiment 3.

1. Change the current working directory to the training user home directory:

```
$ cd ~
```

2. The Zynq cross toolchain supplied with Vivado Design Suite is built to run on a 32-bit x86 machine and the 32-bit version of glibc libraries are needed to run it. Use the following command if the **glibc.i686** package has not already been installed on your CentOS system:

```
$ sudo yum install glibc.i686
```

3. This experiment also uses the Git distributed revision control system to retrieve the **iperf3** source code from a github.com repository. Use the following command if the **git** package has not already been installed on your CentOS system:

```
$ sudo yum install git
```

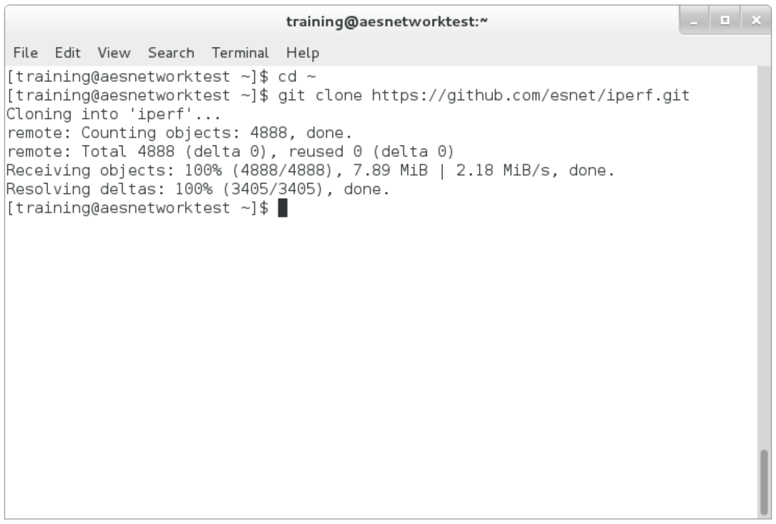
4. Clone the **iperf3** git repository to the local disk using the following command:

```
$ git clone https://github.com/esnet/iperf.git
```

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

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

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


training@aesnetworktest:~
File Edit View Search Terminal Help

```
[training@aesnetworktest ~]$ cd ~
[training@aesnetworktest ~]$ git clone https://github.com/esnet/iperf.git
Cloning into 'iperf'...
remote: Counting objects: 4888, done.
remote: Total 4888 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (4888/4888), 7.89 MiB | 2.18 MiB/s, done.
Resolving deltas: 100% (3405/3405), done.
[training@aesnetworktest ~]$
```

6. Rename the **iperf3** source folder so that we can tell the Zynq UltraScale+ version from the X86 version that is built in the next experiment.

```
$ mv iperf iperf_zynq
```

7. Change from the home directory into the iperf3_zynq source directory.

```
$ cd iperf_zynq/
```

8. Checkout the code changes related to the 3.1.7 version tag.

```
$ git checkout -b 3.1.7 3.1.7
```

9. Configure the **iperf3** source build tree for a Zynq ARM target device.

```
$  ./configure --build=i686-linux --host=aarch64-linux-gnu \
--target=aarch64-linux-gnu --enable-static
```

10. Build the **iperf3** tools for the Zynq ARM target device using make.

```
$  make
```

11. Create a staging folder and install the binaries into system folders.

```
$ mkdir build

$ make install DESTDIR=~/iperf_zynq/build/
```

The **iperf3** application is now built for the Zynq UltraScale+ 64-bit ARM target device and the executable **~/iperf_zynq/build/usr/local/bin/iperf3** file is ready to copy over to your microSD card FAT32 partition for the remaining experiments.

Experiment 3: Building iperf3 for CentOS 7.0

Now that the embedded target software has been setup, **iperf3** must be configured and built for the CentOS 7.0 Linux system that will be used in server mode to test on the local network.

1. Change the current working directory to the training user home directory:

```
$ cd ~
```

2. This experiment uses the x86 GCC compiler tools that are part of the **gcc** package. Use the following command if the **gcc** package has not already been installed on your CentOS system:

```
$ sudo yum install gcc
```

3. This experiment also uses the Git distributed revision control system to retrieve the **iperf3** source code from a github.com repository. Use the following command if the **git** package has not already been installed on your CentOS system:

```
$ sudo yum install git
```

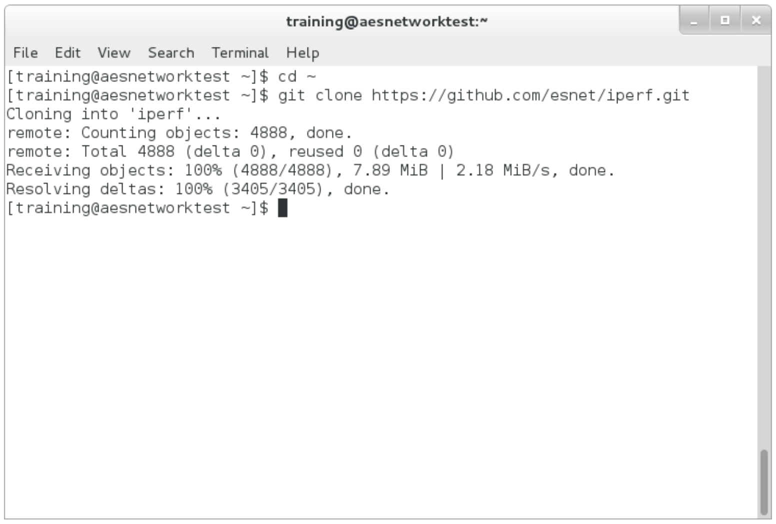
4. Clone the **iperf3** git repository to the local disk using the following command:

```
$ git clone https://github.com/esnet/iperf.git
```

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

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

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


training@aesnetworktest:~
File Edit View Search Terminal Help

```
[training@aesnetworktest ~]$ cd ~
[training@aesnetworktest ~]$ git clone https://github.com/esnet/iperf.git
Cloning into 'iperf'...
remote: Counting objects: 4888, done.
remote: Total 4888 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (4888/4888), 7.89 MiB | 2.18 MiB/s, done.
Resolving deltas: 100% (3405/3405), done.
[training@aesnetworktest ~]$
```

6. Rename the **iperf3** source folder so that we can tell the Zynq version from the x86 version that was built in the previous experiment.

```
$ mv iperf iperf_x86
```

7. Change from the home directory into the **iperf3** x86 source directory.

```
$ cd iperf_x86/
```

8. Checkout the code changes related to the 3.1.7 version tag.

```
$ git checkout -b 3.1.7 3.1.7
```

9. Configure the **iperf3** source build tree for the local machine.

```
$ ./configure
```

10. Build the **iperf3** tools using make.

```
$  make
```

11. Install the binaries into system folders.

```
$ sudo make install
```

The **iperf3** application is now built for the local CentOS machine and ready to run in server mode for the remaining experiments.

Experiment 4: Running Network Performance Tests

After the embedded target software has been setup and test tools are installed on the host Linux PC, UltraZed-EV SOM and Carrier Card can be booted with Linux to a login prompt. From there, the SD card can be mounted and the network tests launched using either the built-in **iperf3** or the binaries built from source code in the earlier Experiment 2.

1. Insert the microSD card, prepared using the steps above, into the UltraZed-EV Carrier J7 slot.

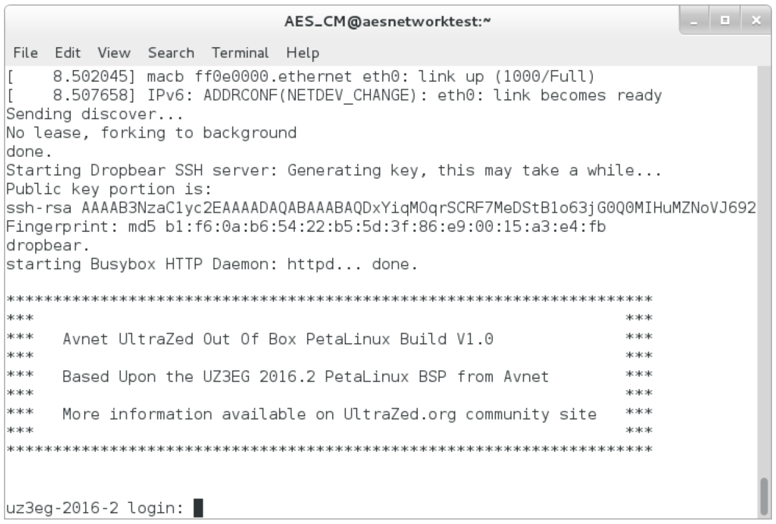
2. Close or disconnect the terminal that may have previously been open on your PC.

3. Connect one end of the Category 5e patch cable to UltraZed-EV Carrier Ethernet port at J8 and connect the other end to the test machine used to run the **iperf3** server built in Experiment 3 above.

4. Set the UltraZed-EV Carrier power switch SW7 to the ON position. The UltraZed system will power on and the Power Good LED (D12) should illuminate.

5. Launch a terminal program with the 115200/8/n/1/n settings. For the example output shown here, the **minicom** terminal was used. For information on setting up **minicom** to use with the UltraZed USB-UART port, see section **Appendix II: Troubleshooting Serial Connection** later in this document for further information.

   You should observe terminal output from U-Boot and then Linux output appear in the minicom window.



```
[    8.502045] macb ff0e0000.ethernet eth0: link up (1000/Full)
[    8.507658] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
Sending discover...
No lease, forking to background
done.
Starting Dropbear SSH server: Generating key, this may take a while...
Public key portion is:
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDxYiqMOqrSCRF7MeDStB1o63jG0Q0MIHuMZNoVJ692
Fingerprint: md5 b1:f6:0a:b6:54:22:b5:5d:3f:86:e9:00:15:a3:e4:fb
dropbear.
starting Busybox HTTP Daemon: httpd... done.

*********************************************************************
***                                                               ***
***   Avnet UltraZed Out Of Box PetaLinux Build V1.0              ***
***                                                               ***
***   Based Upon the UZ3EG 2016.2 PetaLinux BSP from Avnet        ***
***                                                               ***
***   More information available on UltraZed.org community site   ***
***                                                               ***
*********************************************************************


uz3eg-2016-2 login: █
```

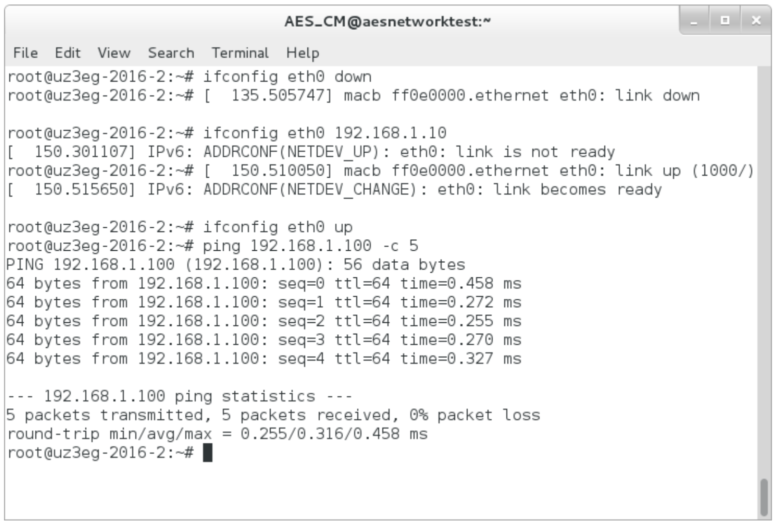
6. Use the terminal window to enter the login **root** along with password **root** in order to gain access to the Zynq Linux command prompt.

7. The RAM disk file system runs DHCP by default to obtain an IP address. Bring the Ethernet interface down, reconfigure to a static IP address of 192.168.1.10, then bring the interface back up using the following commands:

```
# ifconfig eth0 down

# ifconfig eth0 192.168.1.10

# ifconfig eth0 up
```

8. Ping the host CentOS PC address to verify that the network is configured correctly. If the ping command does not return a response from the PC, verify your network cable connection matches the one shown in the section **Troubleshooting: Network Connection** further in this document make.

```
#  ping 192.168.1.100 –c 5
```

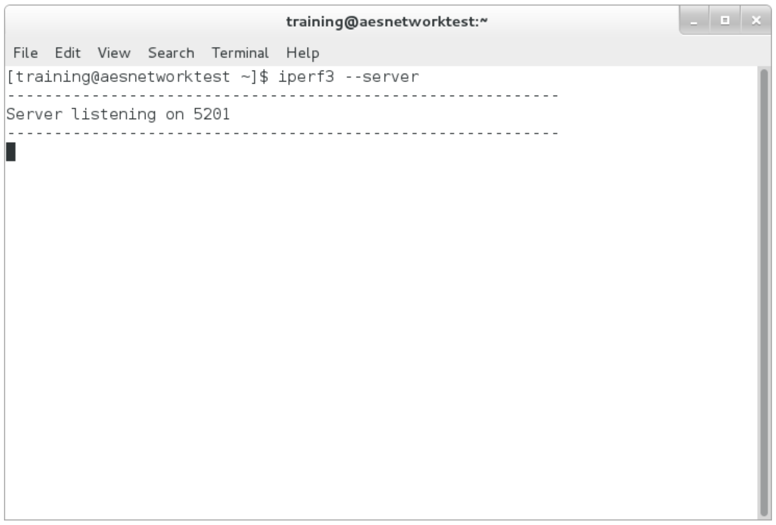


```
root@uz3eg-2016-2:~# ifconfig eth0 down
root@uz3eg-2016-2:~# [  135.505747] macb ff0e0000.ethernet eth0: link down

root@uz3eg-2016-2:~# ifconfig eth0 192.168.1.10
[  150.301107] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
root@uz3eg-2016-2:~# [  150.510050] macb ff0e0000.ethernet eth0: link up (1000/)
[  150.515650] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready

root@uz3eg-2016-2:~# ifconfig eth0 up
root@uz3eg-2016-2:~# ping 192.168.1.100 -c 5
PING 192.168.1.100 (192.168.1.100): 56 data bytes
64 bytes from 192.168.1.100: seq=0 ttl=64 time=0.458 ms
64 bytes from 192.168.1.100: seq=1 ttl=64 time=0.272 ms
64 bytes from 192.168.1.100: seq=2 ttl=64 time=0.255 ms
64 bytes from 192.168.1.100: seq=3 ttl=64 time=0.270 ms
64 bytes from 192.168.1.100: seq=4 ttl=64 time=0.327 ms

--- 192.168.1.100 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.255/0.316/0.458 ms
root@uz3eg-2016-2:~#
```

9. Now that the network connection has been validated, we need to prepare the host CentOS PC to run **iperf3** in server mode so that the client that we run on UltraZed can connect properly.

   Open a new terminal window for the host CentOS PC and start **iperf3** in server mode using the following command:

```
$  iperf3 --server
```


training@aesnetworktest:~
File Edit View Search Terminal Help

```
[training@aesnetworktest ~]$ iperf3 --server
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

10. Switch back to the UltraZed serial terminal window and mount the FAT32 partition of the microSD card to the **/media/** folder so that we can gain access to the **iperf3** executable that was copied over in Experiment 2 above.

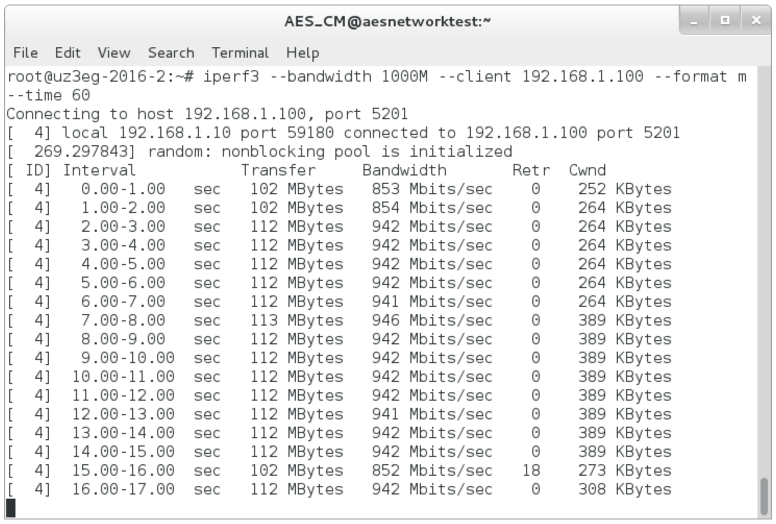
```
#  mount /dev/mmcblk1p1 /media/
```

11. At the UltraZed serial terminal window change to the **/media/** folder and list the contents of that folder. If the microSD card was mounted successfully, you should see the **iperf3** executable listed along with the other files used to boot Linux on UltraZed.

```
#  cd /media

# ls
```

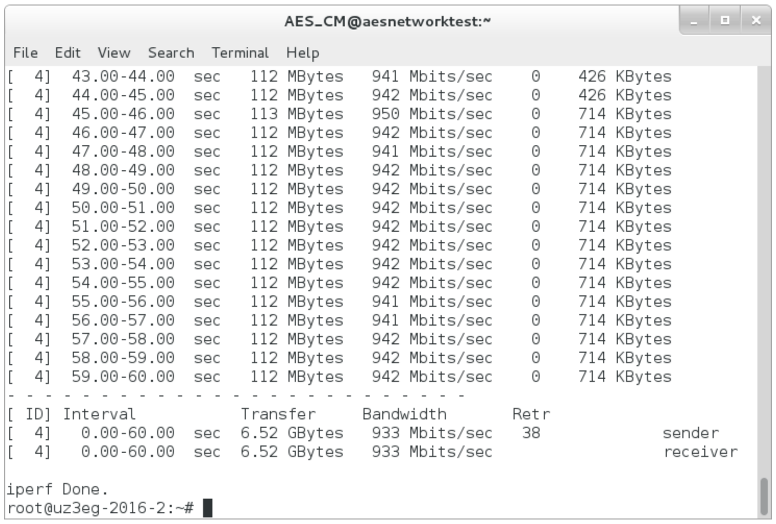
12. At the UltraZed serial terminal launch **iperf3** with a TCP throughput test for 60 seconds using the following command:

```
# ./iperf3 --bandwidth 1000M --client 192.168.1.100 --format m --time 60
```



```
root@uz3eg-2016-2:~# iperf3 --bandwidth 1000M --client 192.168.1.100 --format m
--time 60
Connecting to host 192.168.1.100, port 5201
[  4] local 192.168.1.10 port 59180 connected to 192.168.1.100 port 5201
[  269.297843] random: nonblocking pool is initialized
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   102 MBytes   853 Mbits/sec    0    252 KBytes
[  4]   1.00-2.00   sec   102 MBytes   854 Mbits/sec    0    264 KBytes
[  4]   2.00-3.00   sec   112 MBytes   942 Mbits/sec    0    264 KBytes
[  4]   3.00-4.00   sec   112 MBytes   942 Mbits/sec    0    264 KBytes
[  4]   4.00-5.00   sec   112 MBytes   942 Mbits/sec    0    264 KBytes
[  4]   5.00-6.00   sec   112 MBytes   942 Mbits/sec    0    264 KBytes
[  4]   6.00-7.00   sec   112 MBytes   941 Mbits/sec    0    264 KBytes
[  4]   7.00-8.00   sec   113 MBytes   946 Mbits/sec    0    389 KBytes
[  4]   8.00-9.00   sec   112 MBytes   942 Mbits/sec    0    389 KBytes
[  4]   9.00-10.00  sec   112 MBytes   942 Mbits/sec    0    389 KBytes
[  4]  10.00-11.00  sec   112 MBytes   942 Mbits/sec    0    389 KBytes
[  4]  11.00-12.00  sec   112 MBytes   942 Mbits/sec    0    389 KBytes
[  4]  12.00-13.00  sec   112 MBytes   941 Mbits/sec    0    389 KBytes
[  4]  13.00-14.00  sec   112 MBytes   942 Mbits/sec    0    389 KBytes
[  4]  14.00-15.00  sec   112 MBytes   942 Mbits/sec    0    389 KBytes
[  4]  15.00-16.00  sec   102 MBytes   852 Mbits/sec   18    273 KBytes
[  4]  16.00-17.00  sec   112 MBytes   942 Mbits/sec    0    308 KBytes
```

13. If the **iperf3** client connects successfully to the server running on the host CentOS PC, you should see throughput test progress output appear once every second for 60 seconds total. Upon completion, a throughput summary report will be shown which summarizes the total amount of data transferred along with the average bandwidth measured over the course of the test.



```
[  4]  43.00-44.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  44.00-45.00  sec   112 MBytes   942 Mbits/sec    0    426 KBytes
[  4]  45.00-46.00  sec   113 MBytes   950 Mbits/sec    0    714 KBytes
[  4]  46.00-47.00  sec   112 MBytes   942 Mbits/sec    0    714 KBytes
[  4]  47.00-48.00  sec   112 MBytes   941 Mbits/sec    0    714 KBytes
[  4]  48.00-49.00  sec   112 MBytes   942 Mbits/sec    0    714 KBytes
[  4]  49.00-50.00  sec   112 MBytes   942 Mbits/sec    0    714 KBytes
[  4]  50.00-51.00  sec   112 MBytes   942 Mbits/sec    0    714 KBytes
[  4]  51.00-52.00  sec   112 MBytes   942 Mbits/sec    0    714 KBytes
[  4]  52.00-53.00  sec   112 MBytes   942 Mbits/sec    0    714 KBytes
[  4]  53.00-54.00  sec   112 MBytes   942 Mbits/sec    0    714 KBytes
[  4]  54.00-55.00  sec   112 MBytes   942 Mbits/sec    0    714 KBytes
[  4]  55.00-56.00  sec   112 MBytes   941 Mbits/sec    0    714 KBytes
[  4]  56.00-57.00  sec   112 MBytes   941 Mbits/sec    0    714 KBytes
[  4]  57.00-58.00  sec   112 MBytes   942 Mbits/sec    0    714 KBytes
[  4]  58.00-59.00  sec   112 MBytes   942 Mbits/sec    0    714 KBytes
[  4]  59.00-60.00  sec   112 MBytes   942 Mbits/sec    0    714 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-60.00  sec  6.52 GBytes   933 Mbits/sec   38             sender
[  4]   0.00-60.00  sec  6.52 GBytes   933 Mbits/sec                  receiver

iperf Done.
root@uz3eg-2016-2:~# 
```

14. Continue to experiment with **iperf3** command settings until you have collected the type of information that you need for your own network performance testing. A sample report along with the **iperf3** commands used to generate the results is provided in the **UltraZed-EV - 1000Mbps Ethernet Performance Test Report.pdf** document provided in the archive accompanying this tutorial.

15. When you have completed your network performance testing be sure to un-mount the microSD card to avoid corruption of the FAT32 file system.

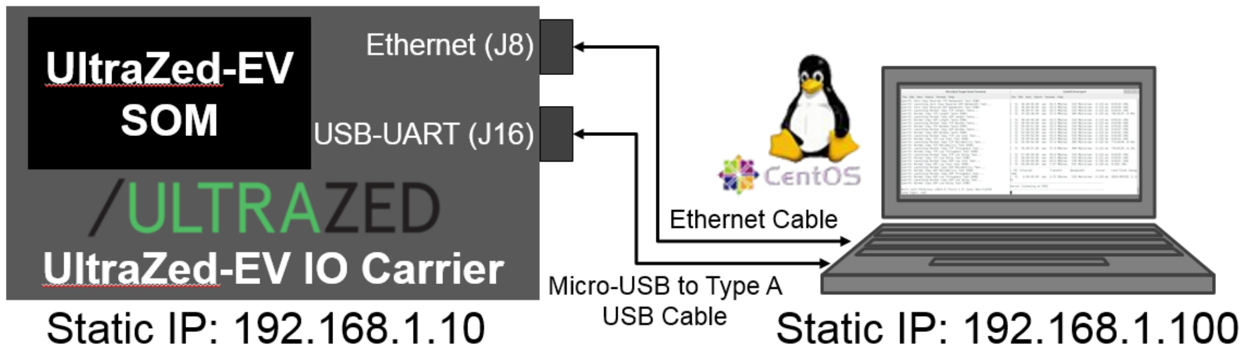
    Alternatively, you can simply shutdown the UltraZed Linux system cleanly by using the Linux **shutdown -h now** command.

```
#  cd /

# umount /media/
```

Appendix I: Troubleshooting Network Connection

This section provides troubleshooting information for the network connection used in this UltraZed Ethernet Performance Test Tutorial.

1. The basic configuration for the UltraZed Ethernet Performance Test Tutorial is shown below:



2. Make sure that Ethernet adapter on Linux host PC is configured as follows:
   - IPv4 Address: 192.168.1.100
   - Subnet Mask: 255.255.255.0
   - Default Gateway: 192.168.1.1

3. Make sure the wireless internet adapter of the Linux host PC is disabled otherwise there may be a routing conflict that prevents the UltraZed Linux host from being reached.

4. Make sure the firewall of the Linux host PC is configured for **iperf3** traffic otherwise there may be a routing conflict that prevents the target UltraZed Linux host from being reached.

Appendix II: Troubleshooting Serial Connection

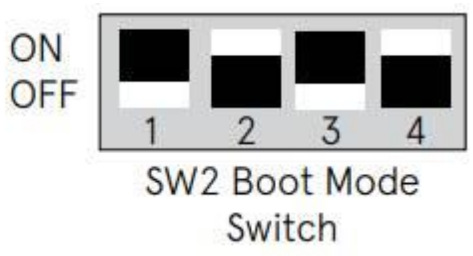
This section provides troubleshooting information for the USB-UART serial connection used in this UltraZed Open Source Linux SATA Performance Test Tutorial. The experiments in this tutorial use **minicom** as the serial terminal application which is recommended for this tutorial but other serial terminal applications might work as well.

1. Use the following command if the **minicom** package has not already been installed on your CentOS system:

```
$ sudo yum install minicom
```

2. Insert a bootable microSD card prepared with the prebuilt binaries available with this tutorial into the UltraZed-EV Carrier Card microSD slot J7 slot.

3. Set the UltraZed-EV SOM Boot Mode switch (SW2) (MODE[3:0] = SW2[4:1]) to ON, OFF, ON, and OFF positions (Boot Mode set to SD Card, MODE[3:0] = 0xA).



4. Make sure the UltraZed-EV Carrier Card power switch SW7 is in the OFF position.

5. Insert the UltraZed-7EV SOM module onto the UltraZed-EV Carrier Card using the JX1, JX2, and JX3 connectors and connect the fan to the fan header (JP14) on the EV Carrier Card.

6. Close or disconnect the terminal that may have previously been open on your PC.

7. Plug in the UltraZed USB-UART cable between the host PC and the UltraZed-EV Carrier Card USB-UART port (J16).

8. Insert the appropriate country plug into the 12V AC/DC adapter. Plug it into the J18 2x3 power connector. (NOTE – this 2x3 connector is NOT compatible with ATX power supplies.)

9. Set the UltraZed-EV Carrier power switch SW7 to the ON position. The UltraZed system will power on and the Power Good LED (D3) should illuminate.

10. Check the kernel output log for signs that the USB-UART device has enumerated and note the ttyUSB device that is enumerated. USB-UART device should enumerate as **/dev/ttyUSB0** or similar.

```
$ dmesg
```

11. Create system default udev rules to give USB-UART devices sufficient permissions for all users similar.

```
$ sudo cp /lib/udev/rules.d/50-udev-default.rules /etc/udev/rules.d/
```

12. Edit the system default udev rules with vi text editor.

```
$ sudo vi /etc/udev/rules.d/50-udev-default.rules
```

13. Edit the system default udev rules with **vi** text editor.

```
$ sudo vi /etc/udev/rules.d/50-udev-default.rules
```

14. Add the following 2 lines to **50-udev-default.rules** somewhere just after the 'tty' section.

```
# relax the permissions just for ttyUSB0
KERNEL=="ttyUSB0",              MODE="0666"
```

15. Save the changes to the **50-udev-default.rules** file and exit **vi** text editor.

16. Add the current user to the **dialout** group which gives permission to access USB-UART serial tty devices.

```
$ sudo usermod -a -G dialout <Current Username>
```

17. Run **minicom** with elevated privileges so that the configuration can be modified.

```
$ sudo minicom -s
```

18. Change minicom 'Serial port setup' settings and perform the following:
    a. Set the **Serial Device** setting to **/dev/ttyUSB0** or to match the device name from Step 6 above.
    b. Set the **Hardware Flow Control** setting to **No**.
    c. Set the **Software Flow Control** setting to **No**.
    d. Save the settings by selecting the **Save setup as dfl** option.

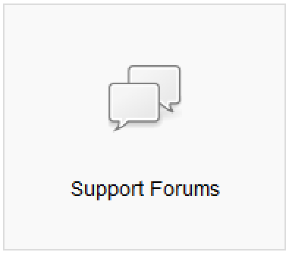
19. Exit from **minicom**.

20. Run **minicom** again in normal user mode and check for terminal output.

```
$ minicom
```

Appendix III: Getting Support

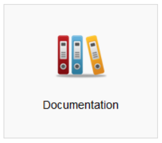
Avnet Support

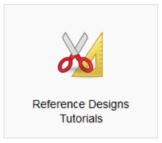


- Technical support is offered online through the ultrazed.org website support forums. UltraZed users are encouraged to participate in the forums and offer help to others when possible.
  http://ultrazed.org/forums/zed-english-forum
  http://ultrazed.org/forums/software-application-development

- For questions regarding the UltraZed community website, please direct questions to the ultrazed.org Web Master (webmaster@ultrazed.org).

- To access the most current collateral for the UltraZed, visit the community support page (www.ultrazed.org/content/support) and click one of the icons shown below:





  - UltraZed-EV SOM Documentation
    http://avnet.me/ultrazed-ev-som-documentation

  - UltraZed-EV Carrier Card
    - Documentation
      http://avnet.me/ultrazed-ev-carrier-documentation

    - Reference Designs
      http://avnet.me/ultrazed-ev-reference-designs

Xilinx Support

For questions regarding products within the Product Entitlement Account, send an email message to the Customer Service Representative in your region:

- Canada, USA and South America - isscs_cases@xilinx.com
- Europe, Middle East, and Africa - eucases@xilinx.com
- Asia Pacific including Japan - apaccase@xilinx.com

For technical support, including the installation and use of the product license file, contact Xilinx Online Technical Support at www.xilinx.com/support. The following assistance resources are also available on the website:

- Software, IP and documentation updates
- Access to technical support Web tools
- Searchable answer database with over 4,000 solutions
- User forums

Internet Support

Here are some helpful links regarding some of the Linux software applications mentioned in this tutorial:

iperf3

- Active network performance measurement
  https://software.es.net/iperf/
- FAQ
  https://software.es.net/iperf/faq.html#faq

Revision History

Date	Version	Revision
21 Jun 18	01	Initial Release