gr-ettus from Source

Contents

AN-315

This application note is one of a multi-part series which will cover the software development process on the USRP E310, E312 and E313. It will cover building UHD, GNU Radio and gr-ettus from source for the host machine, and cross-compiling UHD, GNU Radio and gr-ettus for the E3xx USRP with RFNoC enabled. It will then cover building a custom RFNoC FPGA image and running an example application RFNoC Fosphor.

Note: Linux only. This application note has been tested using Ubuntu 18.04.

Note: This application note is used with the `release-4` E31x file system.

This application note has four sections:

1. Building UHD / GNU Radio / gr-ettus for the Host machine

2. Cross-compiling UHD / GNU Radio / gr-ettus for the E3xx USRP

3. Building a custom RFNoC FPGA image

4. Running an example application, RFNoC Fosphor

This step is required in order to run the RFNoC Fosphor example. It is possible to operate the E3xx USRP with a RFNoC flowgraph without connecting it to a host machine. All of the processing is performed on the E3xx FPGA, the host is only used to display the FFT/Waterfall.

A more detailed guide to installing UHD / GNU Radio on Linux can be found at the following application note, which can be helpful for reference.
https://kb.ettus.com/Building_and_Installing_the_USRP_Open-Source_Toolchain_(UHD_and_GNU_Radio)_on_Linux

This application note is using a vanilla Ubuntu 18.04 installation for the host operating system. If you are using another operating system, please refer to the above application note for specifics on dependency requirements. If you already have UHD and GNU Radio installed, you should first remove them before proceeding with this application note.

```
$ sudo apt update
$ sudo apt upgrade
```

Switch your default shell on the host computer from `Dash` to `Bash`. In some Linux distributions (e.g. Ubuntu) `Dash` is set as default shell, which may cause some issues. It is recommended to set the shell to `Bash` by running the following commands in the terminal. Choose `No` when prompted by the first command and the second command will validate the that `Bash` will be used.

```
$ sudo dpkg-reconfigure dash
```

Verify Bash is the default shell.

```
$ ll /bin/sh
```

Expected Output:

```
lrwxrwxrwx 1 root root 4 Apr  2 22:00 /bin/sh -> bash*
```

```
$ sudo apt -y install git swig cmake doxygen build-essential libboost-all-dev libtool libusb-1.0-0 libusb-1.0-0-dev libudev-dev libncurses
```

These folders will be used as a work area for cloning the sources and building.

```
$ mkdir -p ~/rfnoc
$ mkdir -p ~/rfnoc/src
$ mkdir -p ~/rfnoc/oe
$ mkdir -p ~/rfnoc/e300
```

```
$ cd ~/rfnoc/src
$ git clone --recursive https://github.com/EttusResearch/uhd
$ cd uhd
$ git checkout v3.14.1.1
$ git submodule update --init --recursive
$ cd host
$ mkdir build-host
$ cd build-host
$ cmake -DENABLE_E300=ON -DENABLE_GPSD=ON -DENABLE_RFNOC=ON ..
$ make -j4
$ sudo make install
$ sudo ldconfig
```

```
$ sudo uhd_images_downloader
```

```
$ cd ~/rfnoc/src
$ git clone --recursive https://github.com/gnuradio/gnuradio
$ cd gnuradio/
$ git checkout maint-3.7
$ git submodule update --init --recursive
$ mkdir build-host
$ cd build-host
$ cmake ..
$ make -j4
$ sudo make install
$ sudo ldconfig
```

```
$ cd ~/rfnoc/src
$ git clone https://github.com/EttusResearch/gr-ettus.git
$ cd gr-ettus
$ mkdir build-host
$ cd build-host
$ cmake ..
$ make -j4
$ sudo make install
$ sudo ldconfig
```

Run the following command to verify UHD was installed properly.

```
$ uhd_usrp_probe --version
```

Expected Output:

```
3.14.1.HEAD-0-g0347a6d8
```

Run the following command to verify GNU Radio was installed properly.

```
$ gnuradio-config-info --version
```

Expected Output:

```
3.7.14.0
```

Download the OE SDK for the E31x release into the `~/rfnoc/src/` directory:

```
$ cd ~/rfnoc/src
$ wget http://files.ettus.com/e3xx_images/e3xx-release-4/oecore-x86_64-armv7ahf-vfp-neon-toolchain-nodistro.0.sh
```

Next, install the SDK to the `~/rfnoc/oe` directory. Enter `~/rfnoc/oe` for the installation directory when prompted.

```
$ bash oecore-x86_64-armv7ahf-vfp-neon-toolchain-nodistro.0.sh
```

Expected Output:

```
Enter target directory for SDK (default: /usr/local/oecore-x86_64): ~/rfnoc/oe
You are about to install the SDK to "/home/user/rfnoc/oe". Proceed[Y/n]?y
Extracting SDK...done
Setting it up...done
SDK has been successfully set up and is ready to be used.
```

Your `~/rfnoc/oe` directory should have a structure such as below:

```
$ ls ~/rfnoc/oe
environment-setup-armv7ahf-vfp-neon-oe-linux-gnueabi
version-armv7ahf-vfp-neon-oe-linux-gnueabi
site-config-armv7ahf-vfp-neon-oe-linux-gnueabi
sysroots
```

Source the OE SDK environment setup file:

```
$ cd ~/rfnoc/oe
$ source ./environment-setup-armv7ahf-vfp-neon-oe-linux-gnueabi
```

Verify the environment has been setup correctly.

```
$ echo $CC
```

Expected Output:

```
arm-oe-linux-gnueabi-gcc -march=armv7-a -mfloat-abi=hard -mfpu=neon --sysroot=/home/user/rfnoc/oe/sysroots/armv7ahf-vfp-neon-oe-linux-gnue
```

- Note: If you open a new terminal at any point, you will need to re-run the source command above to initialize the environment. The compiling environment setup is only needed for terminals that are cross-compiling the sources for the E31x. The terminals used later in this application note which are using SSH to connect to other devices do not need to have the compiling environment initialized.

```
$ cd ~/rfnoc/src/uhd/host
$ mkdir build-arm
$ cd build-arm
$ cmake -DCMAKE_TOOLCHAIN_FILE=../host/cmake/Toolchains/oe-sdk_cross.cmake -DCMAKE_INSTALL_PREFIX=/usr -DENABLE_E300=ON -DENABLE_GPSD=ON -
$ make -j4
$ make install DESTDIR=~/rfnoc/e300
$ make install DESTDIR=~/rfnoc/oe/sysroots/armv7ahf-vfp-neon-oe-linux-gnueabi/
```

Next, we will create an environment setup file that will set the correct system variables to point to our new installation when ran on the E31x.

```
$ cd ~/rfnoc/e300
$ touch setup.env
$ nano setup.env
```

Add the following lines to `setup.env`:

```
LOCALPREFIX=~/newinstall/usr
export PATH=$LOCALPREFIX/bin:$PATH
export LD_LOAD_LIBRARY=$LOCALPREFIX/lib:$LD_LOAD_LIBRARY
export LD_LIBRARY_PATH=$LOCALPREFIX/lib:$LD_LIBRARY_PATH
export PYTHONPATH=$LOCALPREFIX/lib/python2.7/site-packages:$PYTHONPATH
export PKG_CONFIG_PATH=$LOCALPREFIX/lib/pkgconfig:$PKG_CONFIG_PATH
export GRC_BLOCKS_PATH=$LOCALPREFIX/share/gnuradio/grc/blocks:$GRC_BLOCKS_PATH
export UHD_RFNOC_DIR=$LOCALPREFIX/share/uhd/rfnoc/
export UHD_IMAGES_DIR=$LOCALPREFIX/share/uhd/images
```

```
$ mkdir -p ~/rfnoc/e300/usr/share/uhd/images
$ cd ~/rfnoc/e300/usr/share/uhd/images
$ cp -Rv /usr/local/share/uhd/images/usrp_e310_fpga* .
$ cp -Rv /usr/local/share/uhd/images/usrp_e3xx_fpga_idle* .
```

Next, we will mount the `~/rfnoc/e300` folder using SSHFS in order to test our new installation.

- Note: This application note assumes you're using a static IP address of `192.168.10.2` for your E31x device, and that your host has the IP address of `192.168.10.1`. If you're using different IP addresses, the commands below will need to be updated to reflect the correct IPs.

Open a new terminal and verify that you can ping the E31x:

```
$ ping 192.168.10.2
```

Next, SSH into the E31x.

```
$ ssh root@192.168.10.2
```

For this next step, your host computer will need to have OpenSSH Server installed.

Open a third terminal window on the host and run:

```
$ sudo apt -y install openssh-server
```

Returning to the terminal which you're connected to the E31x via SSH, mount the `~/rfnoc/e300` folder onto the E31x.

First create a directory to mount the remote folder to with (running these commands on the E31x):

```
# mkdir -p ~/newinstall
# sshfs username@192.168.10.1:/home/user/rfnoc/e300 newinstall/
```

- Note: The "username" in the above command needs to be updated to reflect your user on the host machine.

When prompted, enter the password of your user to complete the mounting of the remote file system.

Verify the mount was successful, your directory structure should match `~/rfnoc/e300` on the host machine.

```
$ ls ~/newinstall/
```

Expected output:

```
setup.env
usr
```

Next, we will need to setup the system environment on the E31x to use the newly compiled version of UHD.

Verify that the E31x is using the existing UHD installation, using the `which` command:

```
# which uhd_usrp_probe
```

Expected output:

```
/usr/bin/uhd_usrp_probe
```

Next, source the `setup.env` file:

```
# cd ~/newinstall
# source ./setup.env
```

Verify that the newly compiled UHD is being used, again using the `which` command:

```
# which uhd_usrp_probe
```

Expected output:

```
/home/root/newinstall/usr/bin/uhd_usrp_probe
```

Next, run `uhd_usrp_probe`:

```
# uhd_usrp_probe
```

Expect output:

```
# uhd_usrp_probe
[INFO] [UHD] linux; GNU C++ version 4.9.2; Boost_105700; UHD_3.14.1.HEAD-0-g0347a6d8
[INFO] [E300] Loading FPGA image: /home/root/newinstall/usr/share/uhd/images/usrp_e310_fpga_sg3.bit...
[INFO] [E300] FPGA image loaded
[INFO] [E300] Detecting internal GPS....
[INFO] [E300] GPSDO found
[INFO] [E300] Initializing core control (global registers)...
[INFO] [E300] Performing register loopback test...
[INFO] [E300] Register loopback test passed
[INFO] [0/Radio_0] Initializing block control (NOC ID: 0x12AD100000000000)
[INFO] [0/DDC_0] Initializing block control (NOC ID: 0xDDC0000000000000)
[INFO] [0/DUC_0] Initializing block control (NOC ID: 0xD0C0000000000002)
  _____________________________________________________
 /
|       Device: E-Series Device
|     _____________________________________________________
|    /
|   |       Mboard: E3XX SG3
|   |   product: 30675
|   |   revision: 6
|   |   serial: xxxxxxx
|   |   mac-addr: 00:80:2f:25:82:bb
|   |   FPGA Version: 255.0
|   |   FPGA git hash: e39334f
|   |   RFNoC capable: Yes
|   |
|   |   Time sources:  none, internal, external, gpsdo
|   |   Clock sources: internal
|   |   Sensors: temp, ref_locked, gps_locked, gps_time, gps_position, gps_gpgga, gps_gprmc
|   |     _____________________________________________________
|   |    /
|   |   |       RX DSP: 0
|   |   |
|   |   |   Freq range: 0.000 to 0.000 MHz
|   |     _____________________________________________________
|   |    /
|   |   |       RX DSP: 1
|   |   |
|   |   |   Freq range: 0.000 to 0.000 MHz
|   |     _____________________________________________________
|   |    /
|   |   |       RX Dboard: A
|   |   |   ID: E310 MIMO XCVR (0x0110)
|   |   |   Serial: xxxxxxx
|   |   |     _____________________________________________________
|   |   |    /
|   |   |   |       RX Frontend: A
|   |   |   |   Name: FE-RX2
|   |   |   |   Antennas: TX/RX, RX2
|   |   |   |   Sensors: temp, rssi, lo_locked
|   |   |   |   Freq range: 50.000 to 6000.000 MHz
|   |   |   |   Gain range PGA: 0.0 to 76.0 step 1.0 dB
|   |   |   |   Bandwidth range: 200000.0 to 56000000.0 step 0.0 Hz
|   |   |   |   Connection Type: IQ
|   |   |   |   Uses LO offset: No
|   |   |     _____________________________________________________
|   |   |    /
|   |   |   |       RX Frontend: B
|   |   |   |   Name: FE-RX1
|   |   |   |   Antennas: TX/RX, RX2
|   |   |   |   Sensors: temp, rssi, lo_locked
|   |   |   |   Freq range: 50.000 to 6000.000 MHz
|   |   |   |   Gain range PGA: 0.0 to 76.0 step 1.0 dB
|   |   |   |   Bandwidth range: 200000.0 to 56000000.0 step 0.0 Hz
|   |   |   |   Connection Type: IQ
|   |   |   |   Uses LO offset: No
|   |   |     _____________________________________________________
|   |   |    /
|   |   |   |       RX Codec: A
|   |   |   |   Name: E3x0 RX dual ADC
|   |   |   |   Gain Elements: None
|   |     _____________________________________________________
|   |    /
|   |   |       TX DSP: 0
|   |   |
|   |   |   Freq range: 0.000 to 0.000 MHz
|   |     _____________________________________________________
|   |    /
|   |   |       TX DSP: 1
|   |   |
|   |   |   Freq range: 0.000 to 0.000 MHz
|   |     _____________________________________________________
|   |    /
|   |   |       TX Dboard: A
|   |   |   ID: E310 MIMO XCVR (0x0110)
|   |   |   Serial: xxxxxxxx
|   |   |     _____________________________________________________
```

```
|   |   |    /
|   |   |   |       TX Frontend: A
|   |   |   |   Name: FE-TX2
|   |   |   |   Antennas: TX/RX
|   |   |   |   Sensors: temp, lo_locked
|   |   |   |   Freq range: 50.000 to 6000.000 MHz
|   |   |   |   Gain range PGA: 0.0 to 89.8 step 0.2 dB
|   |   |   |   Bandwidth range: 200000.0 to 56000000.0 step 0.0 Hz
|   |   |   |   Connection Type: IQ
|   |   |   |   Uses LO offset: No
|   |   |    _____________________________________________________
|   |   |    /
|   |   |   |       TX Frontend: B
|   |   |   |   Name: FE-TX1
|   |   |   |   Antennas: TX/RX
|   |   |   |   Sensors: temp, lo_locked
|   |   |   |   Freq range: 50.000 to 6000.000 MHz
|   |   |   |   Gain range PGA: 0.0 to 89.8 step 0.2 dB
|   |   |   |   Bandwidth range: 200000.0 to 56000000.0 step 0.0 Hz
|   |   |   |   Connection Type: IQ
|   |   |   |   Uses LO offset: No
|   |   |    _____________________________________________________
|   |   |    /
|   |   |   |       TX Codec: A
|   |   |   |   Name: E3x0 TX dual DAC
|   |   |   |   Gain Elements: None
|   |   |    _____________________________________________________
|   |   |   /
|   |   |   |       RFNoC blocks on this device:
|   |   |   |
|   |   |   |   * Radio_0
|   |   |   |   * DDC_0
|   |   |   |   * DUC_0

[INFO] [E300] Loading FPGA image: /home/root/newinstall/usr/share/uhd/images/usrp_e3xx_fpga_idle_sg3.bit...
[INFO] [E300] FPGA image loaded
```

At this point, it's possible to run the included UHD example programs with the newly compiled UHD.

```
# cd ~/newinstall/usr/lib/uhd/examples/
# ./rx_samples_to_file --freq 100e6 --gain 0 --ant TX/RX --rate 1e6 --duration 10 --null
```

GNU Radio versions that are newer than 3.7.10.2 require `python-six`, which is not included within the `release-4` SDK. In this step `python-six` will be installed into the OE sysroot.

Return to the terminal which is being used to cross-compile, or open a new terminal and source the OE SDK environemnt setup file before proceeding.

```
$ cd ~/rfnoc/src
$ wget https://files.pythonhosted.org/packages/16/d8/bc6316cf98419719bd59c91742194c111b6f2e85abac88e496adefaf7afe/six-1.11.0.tar.gz
$ tar -zxvf six-1.11.0.tar.gz
$ cd six-1.11.0
$ python setup.py install --prefix=~/rfnoc/oe/sysroots/x86_64-oesdk-linux/usr
```

Next, clone and cross-compile GNU Radio for the E31x. On the host machine:

```
$ cd ~/rfnoc/src/gnuradio
$ mkdir build-arm
$ cd build-arm
$ cmake -Wno-dev -DCMAKE_TOOLCHAIN_FILE=~/rfnoc/src/gnuradio/cmake/Toolchains/oe-sdk_cross.cmake -DENABLE_INTERNAL_VOLK=OFF -DENABLE_GR_WX
$ make -j4
$ make install DESTDIR=~/rfnoc/e300/
$ make install DESTDIR=~/rfnoc/oe/sysroots/armv7ahf-vfp-neon-oe-linux-gnueabi/
```

Next, verify your GNU Radio installation was successful. On the E31x terminal, run the command:

```
$ gnuradio-config-info --version
```

Expected output:

```
3.7.13.5
```

Next, clone and cross-compile `gr-ettus`. On the host machine:

```
$ cd ~/rfnoc/src/gr-ettus
$ mkdir build-arm
$ cd build-arm
$ cmake -DCMAKE_TOOLCHAIN_FILE=~/rfnoc/src/gnuradio/cmake/Toolchains/oe-sdk_cross.cmake -DCMAKE_INSTALL_PREFIX=/usr ..
$ make -j4
$ make install DESTDIR=~/rfnoc/e300/
$ make install DESTDIR=~/rfnoc/oe/sysroots/armv7ahf-vfp-neon-oe-linux-gnueabi/
```

At this point you should have a fully functional UHD RFNoC, GNU Radio and gr-ettus installation. However being that the compiled sources are not located on the E31x (and only temporarily mounted via SSHFS) they will need to copied to the E31x's file system.

On the E31x navigate to your home directory and create a folder, `localinstall`

```
# mkdir -p ~/localinstall
```

Next, copy the `etc/`, `setup.env` and `usr/` folder from the remotely mounted folder:

```
# cd ~/localinstall

# cp -v ~/newinstall/setup.env .
# cp -Rv ~/newinstall/etc .
# cp -Rv ~/newinstall/usr .
```

Verify the folder contents:

```
# ls ~/localinstall
```

Expected Output:

```
etc/
setup.env
usr/
```

Next, we will need to update the `setup.env` file that is located within the `~/localinstall/` folder.

Verify your directory location with the `pwd` command:

```
$ pwd
```

Expected Output:

```
/home/root/localinstall
```

Edit the `setup.env` file to update the `PATH` variable to point to your new installation location (`/home/root/localinstall`):

```
$ sed -i 's/newinstall/localinstall/g' setup.env
```

Verify your edit was successful with the command `cat`:

```
$ cat setup.env
```

Expected Output:

```
LOCALPREFIX=~/localinstall/usr
export PATH=$LOCALPREFIX/bin:$PATH
export LD_LOAD_LIBRARY=$LOCALPREFIX/lib:$LD_LOAD_LIBRARY
export LD_LIBRARY_PATH=$LOCALPREFIX/lib:$LD_LIBRARY_PATH
export PYTHONPATH=$LOCALPREFIX/lib/python2.7/site-packages:$PYTHONPATH
export PKG_CONFIG_PATH=$LOCALPREFIX/lib/pkgconfig:$PKG_CONFIG_PATH
export GRC_BLOCKS_PATH=$LOCALPREFIX/share/gnuradio/grc/blocks:$GRC_BLOCKS_PATH
export UHD_RFNOC_DIR=$LOCALPREFIX/share/uhd/rfnoc/
export UHD_IMAGES_DIR=$LOCALPREFIX/share/uhd/images
```

Note, the `LOCALPREFIX` variable has been updated to the new location.

This `setup.env` file needs to be source in order to utilize the new `~/localinstall` location.

```
$ source ./setup.env
```

Verify that the environment is setup correctly:

```
$ which uhd_usrp_probe
```

Expected Output:

```
/home/root/localinstall/usr/bin/uhd_usrp_probe
```

We can now dismount the remotely connected SSHFS folder. On the E31x, run the commands:

```
# cd ~/
# umount ~/newinstall
# rm -rf ~/newinstall
```

The default FPGA image shipped with UHD releases only contains the `Radio`, `DDC`, and `DUC` blocks. In order to explore the additional RFNoC blocks, a custom FPGA image will need to be built.

To build a FPGA image for the USRP E31x, the Xilinx Vivado toolchain must be installed. A future application note will cover a step-by-step install guide for Vivado. The E31x FPGA is a Zynq 7020 and can be built using the free Vivado WebPACK tools. Vivado System and Design Edition will also work.

Note: UHD 3.14.x.x requires Vivado 2017.4.

UHD provides several tools for building FPGA images. The step below assume you have a working installation of Xilinx Vivado WebPACK.

The first step is to source the environment setup file for the Vivado toolchain.

In a new terminal, run:

```
$ source ~/rfnoc/src/uhd/fpga-src/usrp3/top/e300/setupenv.sh
```

Next, you can use either the `uhd_image_builder_gui.py` utility or the command line version `uhd_image_builder.py` to build the FPGA image.

In this step we will build a FPGA image with the following blocks:

- 1x FFT

- 1x Window

- 1x Fosphor

- 2x FIFO

Note: Depending upon your host machine performance, it can take a few hours to build a FPGA image.

```
$ cd ~/rfnoc/src/uhd/fpga-src/usrp3/tools/scripts
$ ./uhd_image_builder.py fft window fosphor -t E310_RFNOC_sg3 -d E310 -m 5 --fill-with-fifos
```

Note: `uhd_image_builder_gui.py` requires Python3.

```
$ cd ~/rfnoc/src/uhd/fpga-src/usrp3/tools/scripts
$ python3 uhd_image_builder_gui.py
```

- Select the `E310_RFNOC_SG3` as the build target

- Select the FFT, Window, and Fosphor block by highlighting them and clicking `>>` to add them to the design

- Click the `Fill with FIFOs` checkbox

- Then click `Generate .bit file`

Once the FPGA image has completed in the previous step, it will need to be copied to to the E31x.

```
$ scp /path/to/e300.bit root@192.168.10.2:~/localinstall/.
```

Note: The location of the generate bit file is printed at the end of the building process.

Example location:

```
/home/user/rfnoc/src/uhd/fpga-src/usrp3/top/e300/build-E310_RFNOC_sg3/e300.bit
```

Next, SSH into the E31x and run `uhd_usrp_probe`, using this FPGA new custom FPGA image. The custom FPGA image path should be passed as a device argument to the UHD application.

```
$ ssh root@192.168.10.2

# source ./localinstall/setup.env
# uhd_usrp_probe --args"fpga=/home/root/localinstall/e300.bit"
```

Expected Output:

```
# uhd_usrp_probe --args "fpga=/home/root/localinstall/e300.bit"
[INFO] [UHD] linux; GNU C++ version 4.9.2; Boost_105700; UHD_3.14.1.HEAD-0-g0347a6d8
[INFO] [E300] Loading FPGA image: /home/root/localinstall/e300.bit...
[INFO] [E300] FPGA image loaded
[INFO] [E300] Detecting internal GPS....
[INFO] [E300] GPSDO found
[INFO] [E300] Initializing core control (global registers)...

[INFO] [E300] Performing register loopback test...
[INFO] [E300] Register loopback test passed
[INFO] [0/Radio_0] Initializing block control (NOC ID: 0x12AD100000000000)
[WARNING] [RFNOC] Can't find a block controller for key FFT, using default block controller!
[INFO] [0/FFT_0] Initializing block control (NOC ID: 0xFF70000000000000)
[INFO] [0/Window_0] Initializing block control (NOC ID: 0xD053000000000000)
[WARNING] [RFNOC] Can't find a block controller for key fosphor, using default block controller!
[INFO] [0/fosphor_0] Initializing block control (NOC ID: 0x666F000000000000)
[INFO] [0/FIFO_0] Initializing block control (NOC ID: 0xF1F0000000000000)
[INFO] [0/FIFO_1] Initializing block control (NOC ID: 0xF1F0000000000000)
  _____________________________________________________
 /
|       Device: E-Series Device
|     _____________________________________________________
|    /
|   |       Mboard: E3XX SG3
|   |   product: 30675
|   |   revision: 6
|   |   serial: xxxxxxx
|   |   mac-addr: 00:80:2f:25:82:bb
|   |   FPGA Version: 255.0
|   |   FPGA git hash: e39334f-dirty
|   |   RFNoC capable: Yes
|   |
|   |   Time sources:  none, internal, external, gpsdo
|   |   Clock sources: internal
|   |   Sensors: temp, ref_locked, gps_locked, gps_time, gps_position, gps_gpgga, gps_gprmc
|   |     _____________________________________________________
|   |    /
|   |   |       RX DSP: 0
|   |   |
|   |   |   Freq range: 0.000 to 0.000 MHz
|   |     _____________________________________________________
|   |    /
|   |   |       RX DSP: 1
|   |   |
|   |   |   Freq range: 0.000 to 0.000 MHz
|   |     _____________________________________________________
|   |    /
|   |   |       RX Dboard: A
|   |   |   ID: E310 MIMO XCVR (0x0110)
|   |   |   Serial: xxxxxxx
|   |   |     _____________________________________________________
|   |   |    /
|   |   |   |       RX Frontend: A
|   |   |   |   Name: FE-RX2
|   |   |   |   Antennas: TX/RX, RX2
|   |   |   |   Sensors: temp, rssi, lo_locked
|   |   |   |   Freq range: 50.000 to 6000.000 MHz
|   |   |   |   Gain range PGA: 0.0 to 76.0 step 1.0 dB
|   |   |   |   Bandwidth range: 200000.0 to 56000000.0 step 0.0 Hz
|   |   |   |   Connection Type: IQ
|   |   |   |   Uses LO offset: No
|   |   |     _____________________________________________________
|   |   |    /
|   |   |   |       RX Frontend: B
|   |   |   |   Name: FE-RX1
|   |   |   |   Antennas: TX/RX, RX2
|   |   |   |   Sensors: temp, rssi, lo_locked
|   |   |   |   Freq range: 50.000 to 6000.000 MHz
|   |   |   |   Gain range PGA: 0.0 to 76.0 step 1.0 dB
|   |   |   |   Bandwidth range: 200000.0 to 56000000.0 step 0.0 Hz
|   |   |   |   Connection Type: IQ
```

```
|    |    |    |   Uses LO offset: No
|    |    |     ------------------------------------------------------
|    |    |    /
|    |    |    |       RX Codec: A
|    |    |    |   Name: E3x0 RX dual ADC
|    |    |    |   Gain Elements: None
|    |    |     ------------------------------------------------------
|    |    |    /
|    |    |    |       TX DSP: 0
|    |    |    |
|    |    |    |   Freq range: 0.000 to 0.000 MHz
|    |    |     ------------------------------------------------------
|    |    |    /
|    |    |    |       TX DSP: 1
|    |    |    |
|    |    |    |   Freq range: 0.000 to 0.000 MHz
|    |    |     ------------------------------------------------------
|    |    |    /
|    |    |    |       TX Dboard: A
|    |    |    |   ID: E310 MIMO XCVR (0x0110)
|    |    |    |   Serial: xxxxxxx
|    |    |    |    ------------------------------------------------------
|    |    |    |   /
|    |    |    |   |       TX Frontend: A
|    |    |    |   |   Name: FE-TX2
|    |    |    |   |   Antennas: TX/RX
|    |    |    |   |   Sensors: temp, lo_locked
|    |    |    |   |   Freq range: 50.000 to 6000.000 MHz
|    |    |    |   |   Gain range PGA: 0.0 to 89.8 step 0.2 dB
|    |    |    |   |   Bandwidth range: 200000.0 to 56000000.0 step 0.0 Hz
|    |    |    |   |   Connection Type: IQ
|    |    |    |   |   Uses LO offset: No
|    |    |    |   |    ------------------------------------------------------
|    |    |    |   |   /
|    |    |    |   |   |       TX Frontend: B
|    |    |    |   |   |   Name: FE-TX1
|    |    |    |   |   |   Antennas: TX/RX
|    |    |    |   |   |   Sensors: temp, lo_locked
|    |    |    |   |   |   Freq range: 50.000 to 6000.000 MHz
|    |    |    |   |   |   Gain range PGA: 0.0 to 89.8 step 0.2 dB
|    |    |    |   |   |   Bandwidth range: 200000.0 to 56000000.0 step 0.0 Hz
|    |    |    |   |   |   Connection Type: IQ
|    |    |    |   |   |   Uses LO offset: No
|    |    |    |   |   |    ------------------------------------------------------
|    |    |    |   |   |   /
|    |    |    |   |   |   |       TX Codec: A
|    |    |    |   |   |   |   Name: E3x0 TX dual DAC
|    |    |    |   |   |   |   Gain Elements: None
|    |    |    |   |   |    ------------------------------------------------------
|    |    |    |   |   |   /
|    |    |    |   |   |   |       RFNoC blocks on this device:
|    |    |    |   |   |   |
|    |    |    |   |   |   |   * Radio_0
|    |    |    |   |   |   |   * FFT_0
|    |    |    |   |   |   |   * Window_0
|    |    |    |   |   |   |   * fosphor_0
|    |    |    |   |   |   |   * FIFO_0
|    |    |    |   |   |   |   * FIFO_1

[INFO] [E300] Loading FPGA image: /home/root/localinstall/usr/share/uhd/images/usrp_e3xx_fpga_idle_sg3.bit...
[INFO] [E300] FPGA image loaded
```

Note: At the end of the `uhd_usrp_probe` output is a list of blocks within this custom FPGA image.

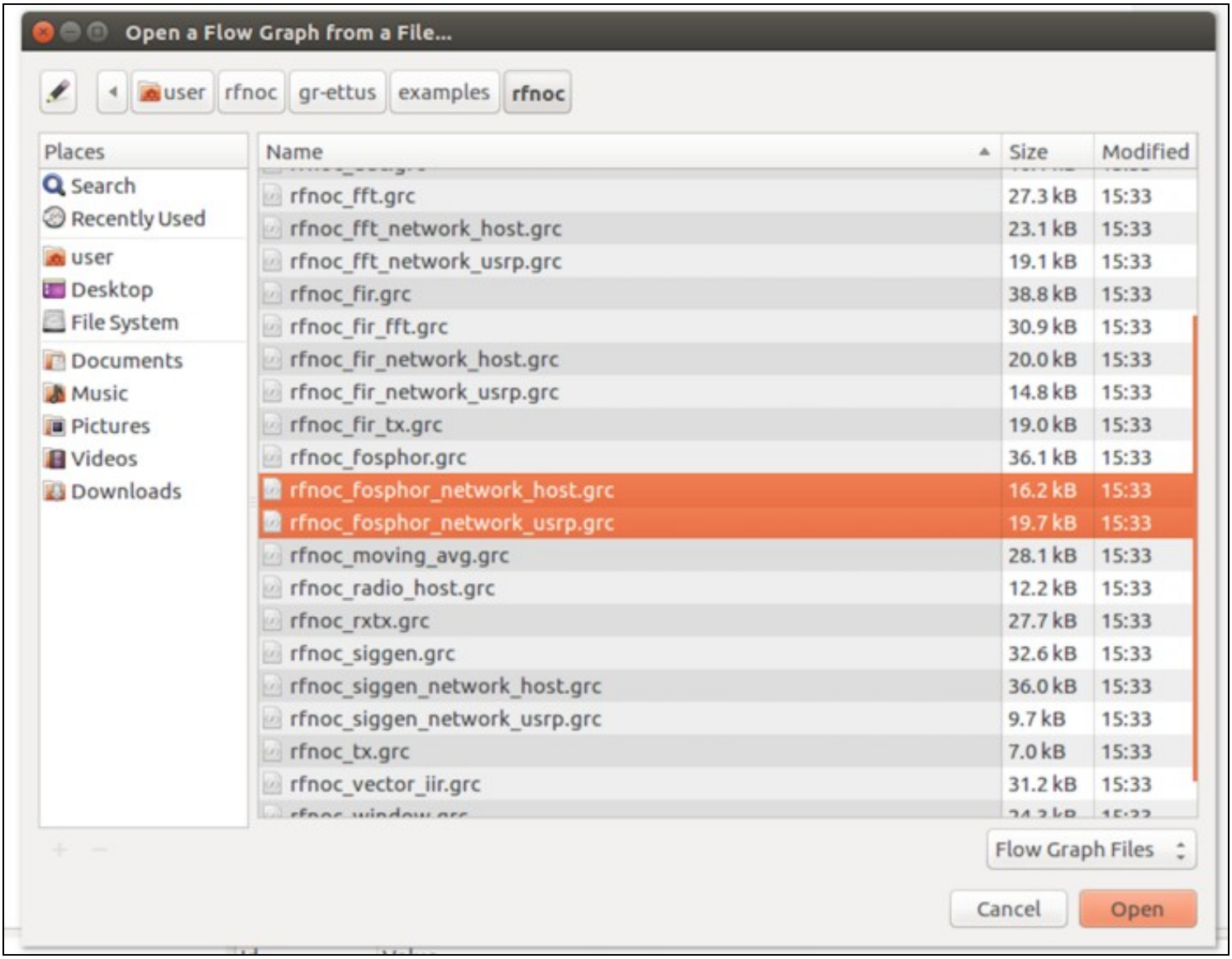
In order to run RFNoC Fosphor, we will first need to generate the Python file to be ran on the E3xx USRP.

On the Host machine, in a new terminal, open GNU Radio:

```
$ gnuradio-companion
```

Within GNU Radio, open the following files:

```
/home/user/rfnoc/src/gr-ettus/examples/rfnoc/rfnoc_fosphor_network_host.grc
/home/user/rfnoc/src/gr-ettus/examples/rfnoc/rfnoc_fosphor_network_usrp.grc
```



Select the `rfnoc_fosphor_network_usrp.grc` flowgraph.

Modify the Variable `ip_addr` and set the value to match the Host's address: `192.168.10.1`

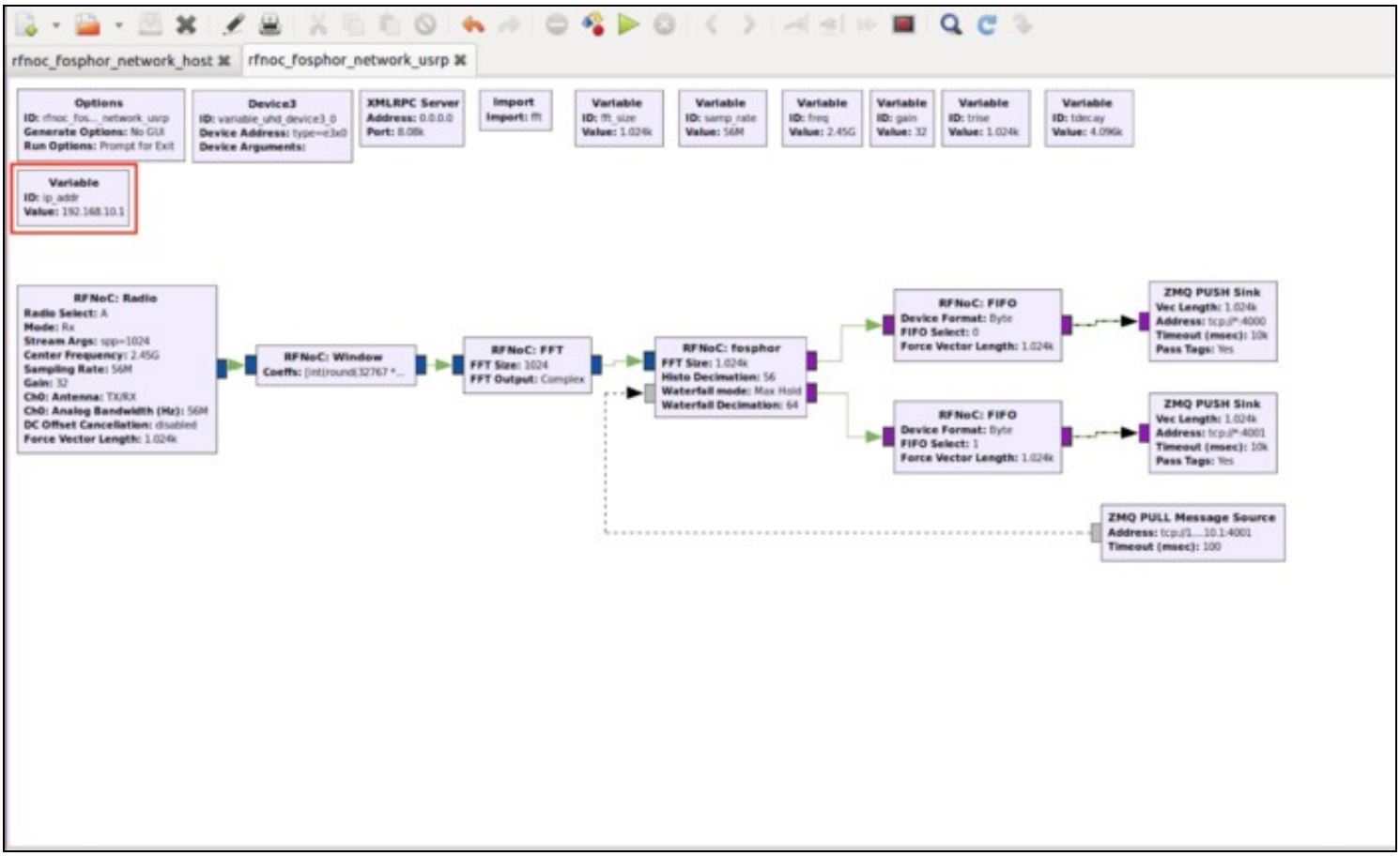
Modify the `device3` block to set the `master_clock_rate` to be `56e6` and FPGA path argument.

Example:

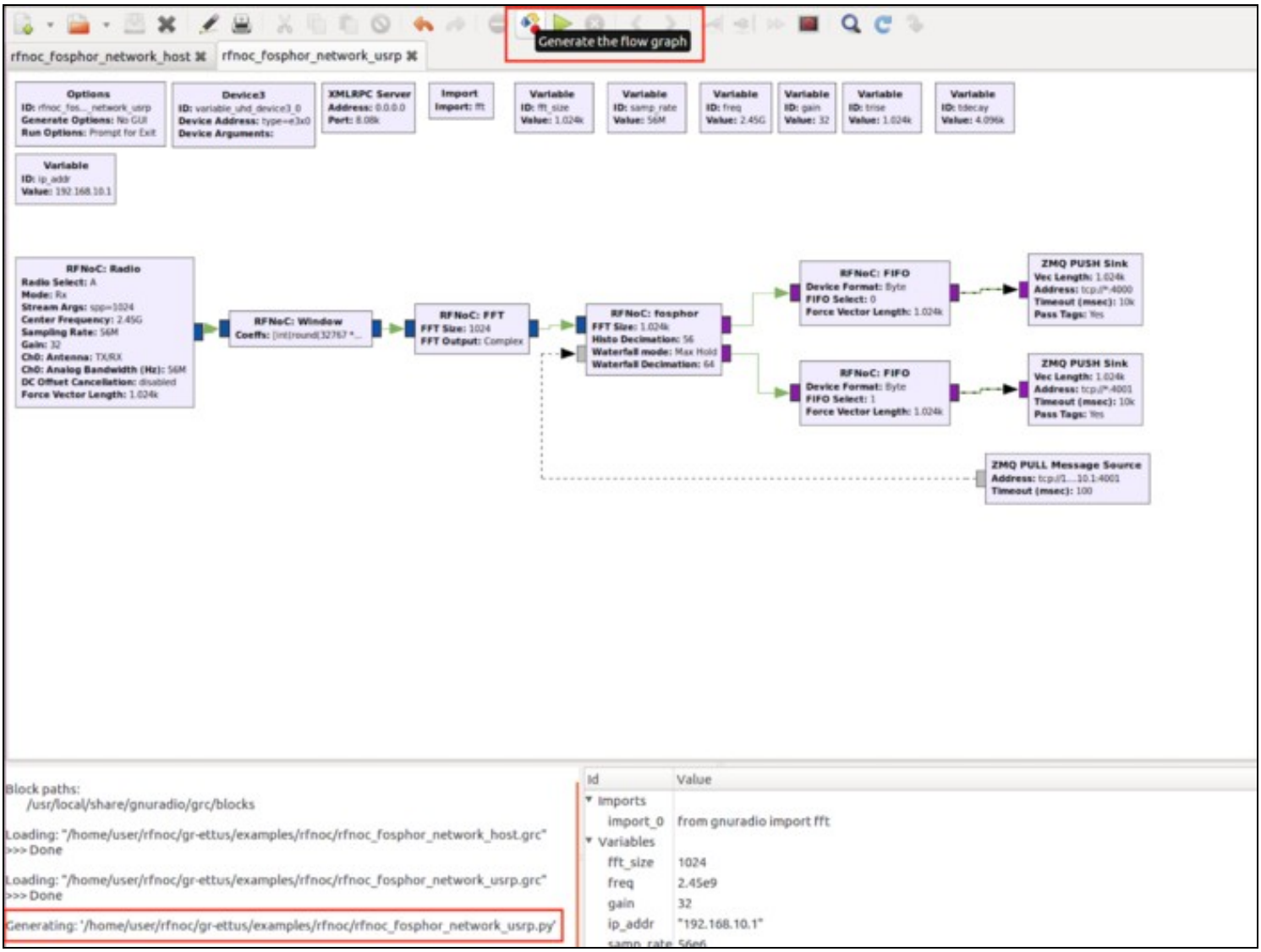
```
master_clock_rate=56e6,fpga=/home/root/localinstall/e300.bit
```

Modify the variable `fft_size` to be `512`

Modify the variable `spp` under the `USRP UHD Source` block to be `512`

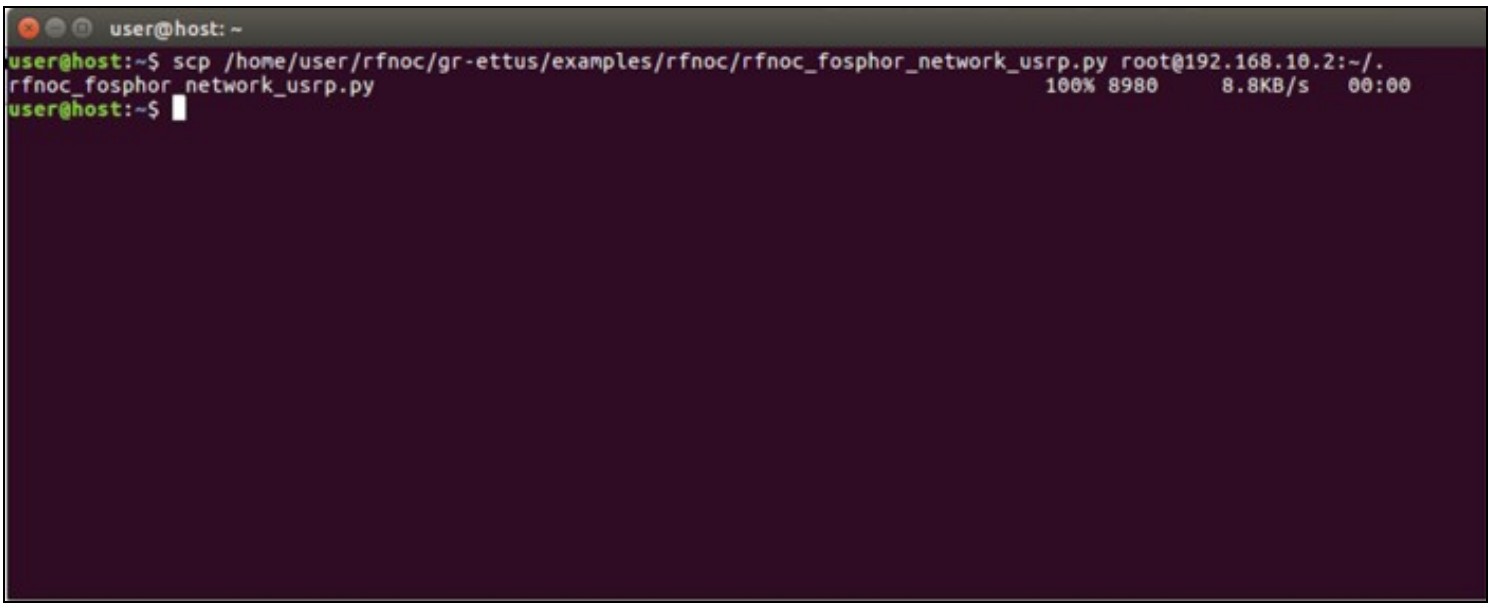


Next, click the `Generate the flow graph` button. Note the file path in the console where it is generating the `rfnoc_fosphor_network_usrp.py` is created.



Copy the generated Python file, `rfnoc_fosphor_network_usrp.py` to the E3xx using the `scp` utility.

```
$ scp /home/user/rfnoc/src/gr-ettus/examples/rfnoc/rfnoc_fosphor_network_usrp.py root@192.168.10.2:~/.
```



You will now need to SSH to the USRP E3xx.

```
$ ssh root@192.168.10.2
```

Verify that the `rfnoc_fosphor_network_usrp.py` file was copied to your E3xx.

```
# ls -al rfnoc_fosphor_network_usrp.py
```

Expected Output:

```
root@ettus-e3xx-sg3:~# ls -al rfnoc_fosphor_network_usrp.py
-rwxr-xr-- 1 root root 8980 Jan 15 04:43 rfnoc_fosphor_network_usrp.py
```

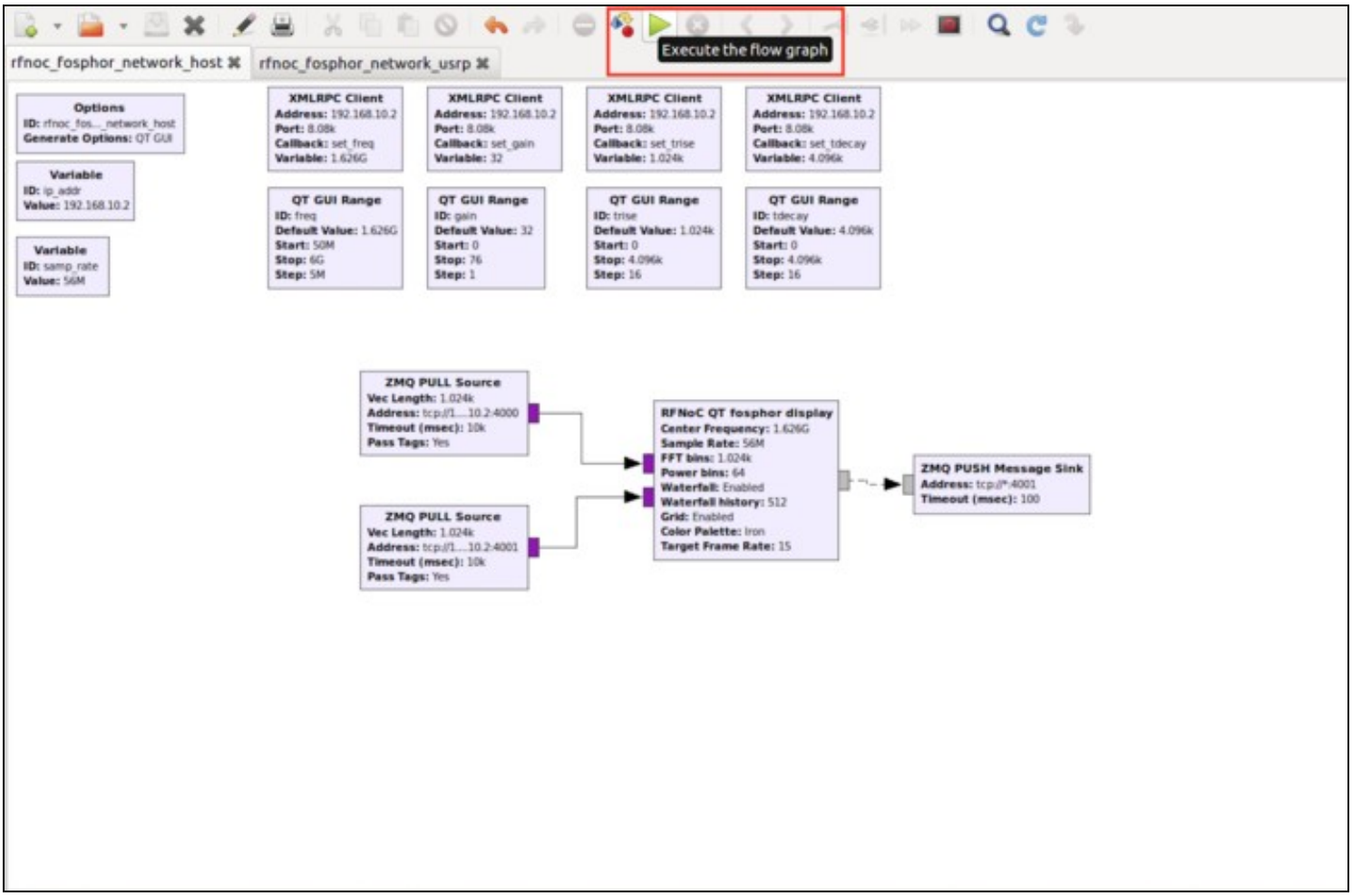
Source the Environment Setup file, `setup.env`.

```
# source ./localinstall/setup.env
```

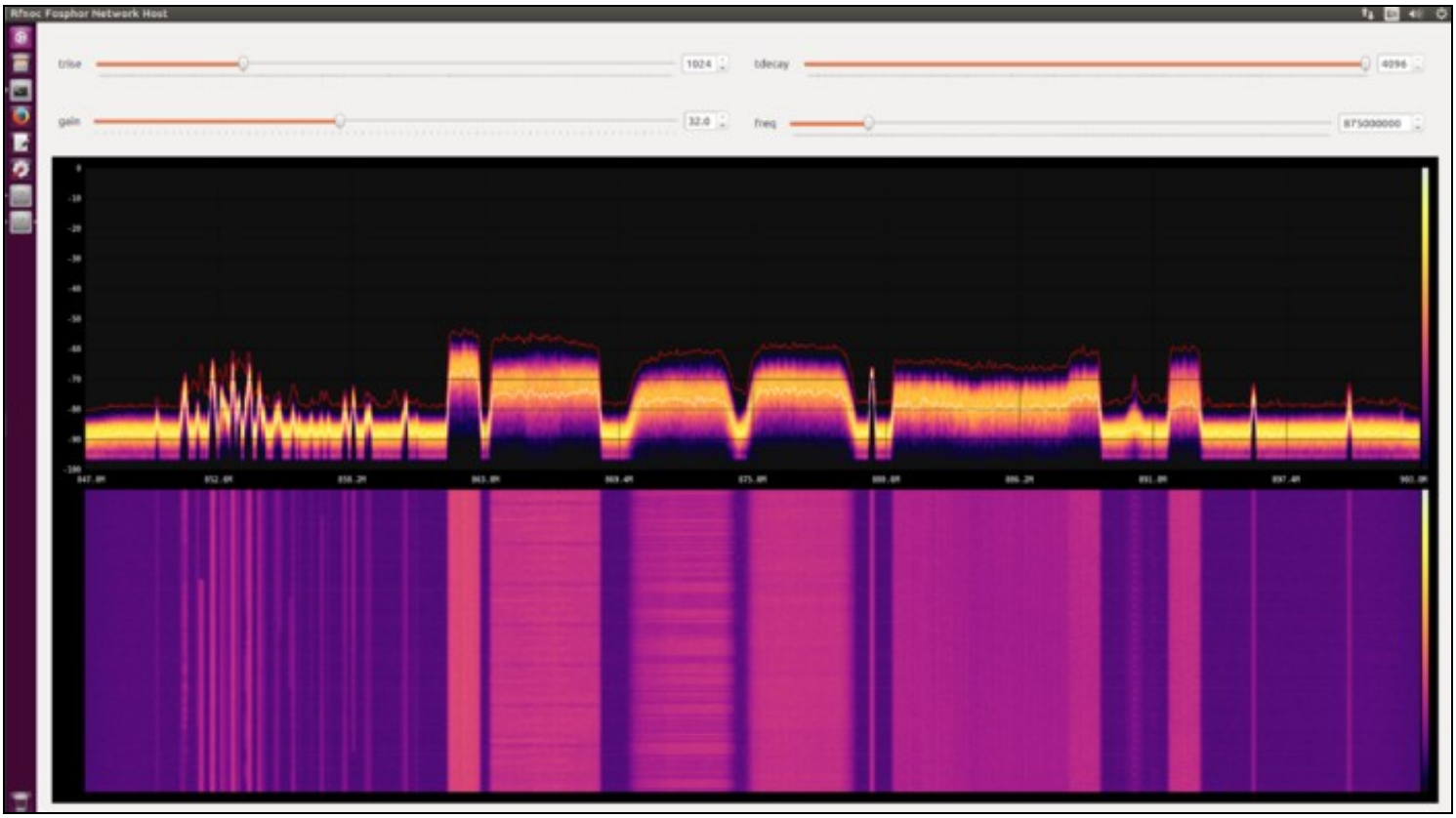
Next, run `rfnoc_fosphor_network_usrp.py`. Leave this window open in the background.

```
# python rfnoc_fosphor_network_usrp.py
```

Return to `gnuradio-companion`, select the `rfnoc_fosphor_network_host.grc` flowgraph, and click the `Execute the flow graph` button.



Maximize the GUI window that has started. RFNoC Fosphor should be running now on the E3xx and be displayed on your host computer. Adjust the Frequency to a strong set of signals, and adjust the Gain slider as needed to produce the best signal to noise ratio for your RF environment.


Rfsoc Fosphor Network Host
trise
1024
tdecay
4096
gain
32.0
freq
875000000