Ferry Bridge FPGA

User Guide

October 2018

Intel Confidential

Document Number: 548166, Rev. 1.1

Contents

Figures

Tables

Revision History

Date	Revision	Description
October 2018	1.1	Release of document corresponding to R1.4.1 release.
August 2017	1.0	Initial release of document corresponding to R1.4.0 release

§

1 Introduction

1.1 Scope

The document describes how to set up the Ferry Bridge Environment and then make and run a test.

1.2 Terminology

Table 1-1. Terminology

Term	Description
BBU	Baseband Unit
BS	Base Station
CPRI	Common Public Radio Interface
DL	Downlink
DPDK	Data Plane Development Kit
FB	Ferry Bridge
FEDK	Front-End Development Kit
FFT	Fast Fourier Transform
FPGA	Field-Programmable Gate Array
LTE	Long Term Evolution
PDSCH	Physical Downlink Shared Channel
PRACH	Physical Random Access Channel
PUSCH	Physical Uplink Shared Channel
UE	User Equipment
UL	Uplink
VM	Virtual Machine

1.3 Related Documentation

Table 1-2. Related Documentation

Document Name	Document No.
Ferry Bridge Release Notes	571743
Ferry Bridge Programmer's Guide	574223

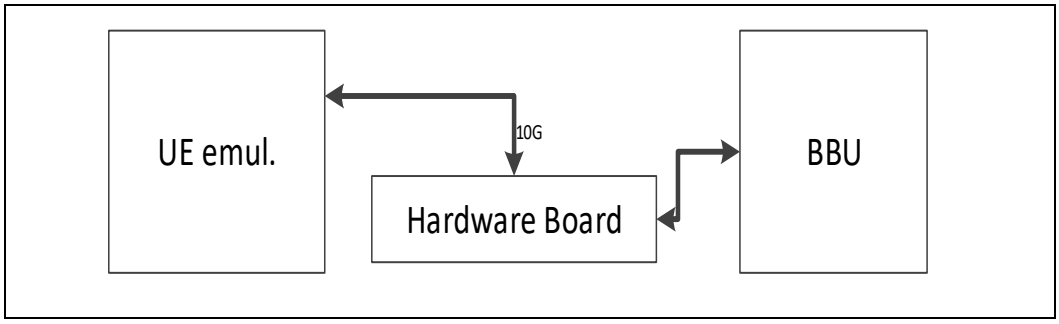
NOTE: Contact your Intel representative for the latest version of the document.

§

2 System Overview

The system setup is shown below where the Ferry Bridge Field Programmable Gate Arrray (FPGA) hardware is used as a PC peripheral board. The board supported are the Ferry Bridge AMC Card and the Terasic* Card. In both cases the setup contains two Intel server machines and one hardware board as shown in the following figure.

Figure 2-1. System Setup – HW Board



The role of each entity is as follows:

BBU: Runs Long Term Evolution (LTE) code on the Base Station (BS) side. When using the sample application this is the REC mode.

UE Emulator: Runs LTE code on the User Equipment (UE) side. When using the sample application this is the RE mode.

§

3 Software Configuration

3.1 Operating System Configuration

The OS can be Fedora* 14 or 16. It can also run on a virtual machine; the validated system is WR OVP 6.

For a Fedora system, you need the rt patch to ensure real-time disposal. Otherwise, you can use the below command.

Note: The command is for an eight core per socket machine. If a core number is changed per socket, the mask 01010101 in red text below should be changed accordingly.

```
sysctl -w kernel.sched_rt_runtime_us=-1 # no time for non-rt process
chkconfig --level 12345 irqbalance off # interrupts are not distributed to all cores

for i in ` ls  /proc/irq |grep -v default_smp_affinity | grep -v 0 |grep -v 2 `
    do
        echo "01010101" > /proc/irq/$i/smp_affinity
done
```

An OVP system is predefined and can ensure real-time disposal.

3.2 Intel® C++ Compiler

Intel® C++ Compiler is used to compile the Intel® DPDK and LTE code. Version 18.0.1 should be used.

For DPDK builds, the setup script under tools is named `setup.sh` and it helps to compile and insert the mode.

For hardware driver builds, the Makefile is in `ferrybridge/lib`.

The Sample Application is in `ferrybridge/app`.

3.3 DPDK Configuration

Install the Intel® Data Plane Development Kit (DPDK) on the BBU and UE machines. After copying Intel® DPDK source code to the `DPDKInstall` directory, generate the user space driver by running `DPDKInstall/tools/setup.sh` and install.

Both the BBU and UE emulators use Intel® DPDK for high-speed packet communication. Use Intel® DPDK version 18.08.

Configure the Intel® DPDK environment first:

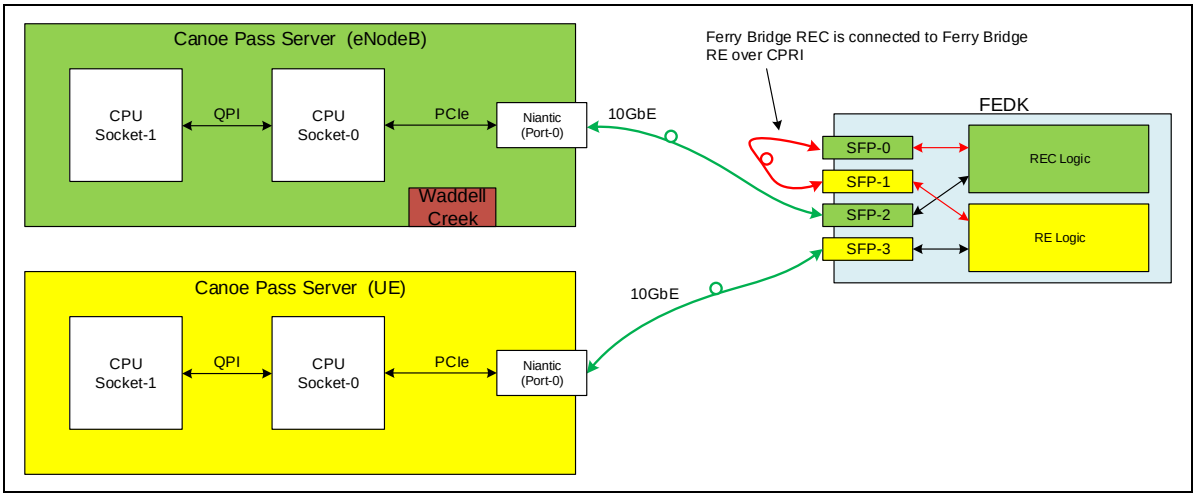
```
export RTE_SDK=DPDK installation directory
export RTE_TARGET=x86_64-default-linuxapp-icc
```

§

4 Hardware Configuration

The version 1.4.1 Evaluation consists of two servers and one Ferry Bridge Front-End Development Kit (FEDK) or the Terasic* Card. The FEDK performs two functions, BB (eNodeB) processing and CPRI termination at the UE side. Each function is independent. The CPRI cable connects the eNodeB side to the UE as shown in the diagram below.

Figure 4-1. Hardware Configuration for FEDK card

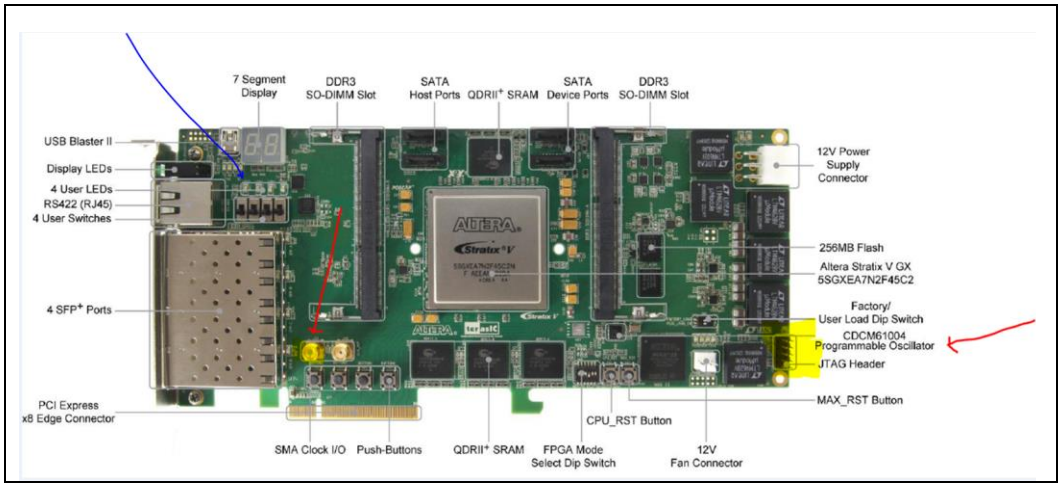


4.1 Terasic* Board Setup

To use the Terasic* board, the OXCO timing module must be installed on the board itself. If the OXCO module is not already installed follow the below steps to install.

1. Plug the module into the programmable oscillator location at the end of board highlighted in yellow in Figure xx.
2. Screw the cable into the SMA connector (highlighted in yellow with red arrow)
3. Once the LED (highlighted with Blue arrow) lights up, the PLL is locked on the OXCO 30.72MHz clock.

Figure 4-2. Terasic* Board



4.2 Hardware Setup

The two Canoe Pass servers are identical. The software installation determines that one acts as an eNodeB and the other as a UE.

1. Connect an Ethernet cable from `Eth-port-0` of each Canoe Pass server to the host PC.
   a. Eth-0 of eNodeB has a static IP address of 192.168.1.2.
   b. Eth-0 of UE has a static IP address of 192.168.1.3.
2. Connect an optical cable between SFP Ports 0 and 1 on the FEDK.
3. Connect an optical cable between eNodeB 10 GbE `port-0` and SFP-`port-2` on the FEDK.
4. Connect an optical cable between the UE 10 GbE `port-0` and SFP-`port-3` on the FEDK.
5. Connect the AC adaptor to the DC power inlet on the rear of the FEDK.
6. Power on the FEDK. The FPGA is automatically configured at power-on.

Figure 4-3. eNodeB and UE 10GbE Port

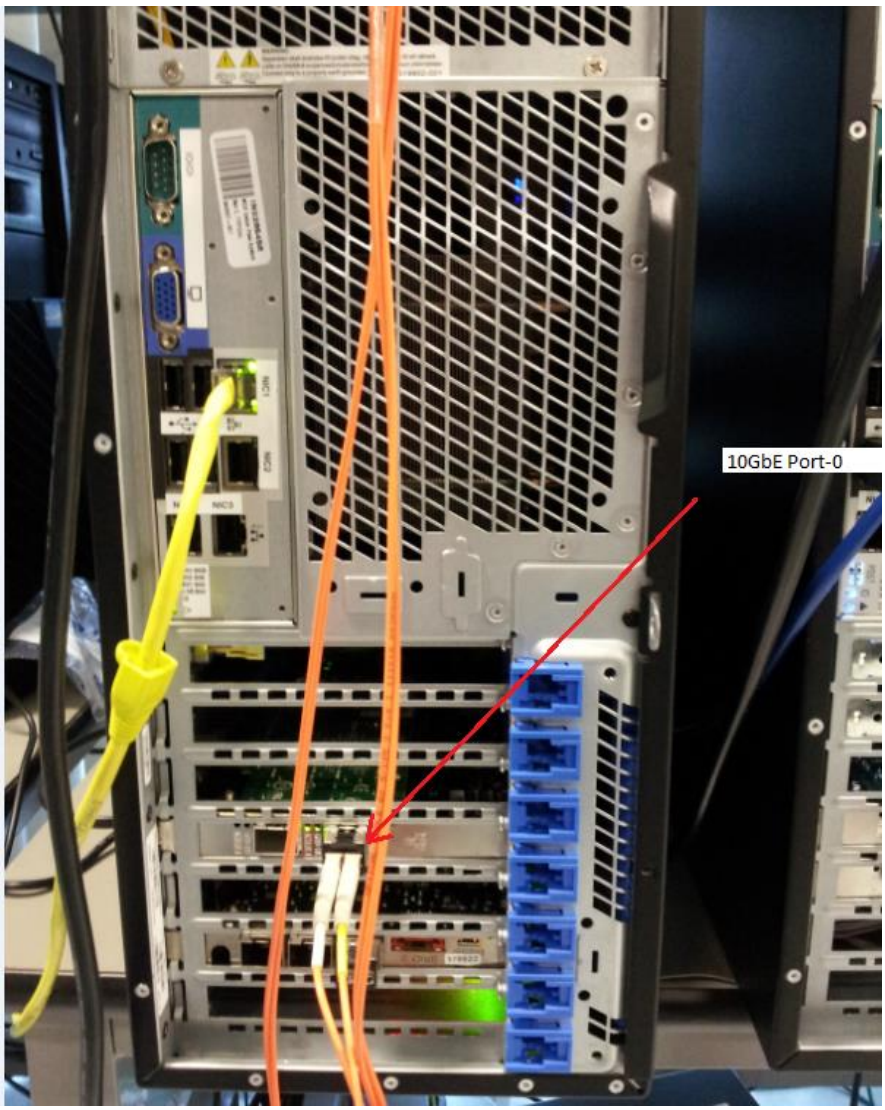
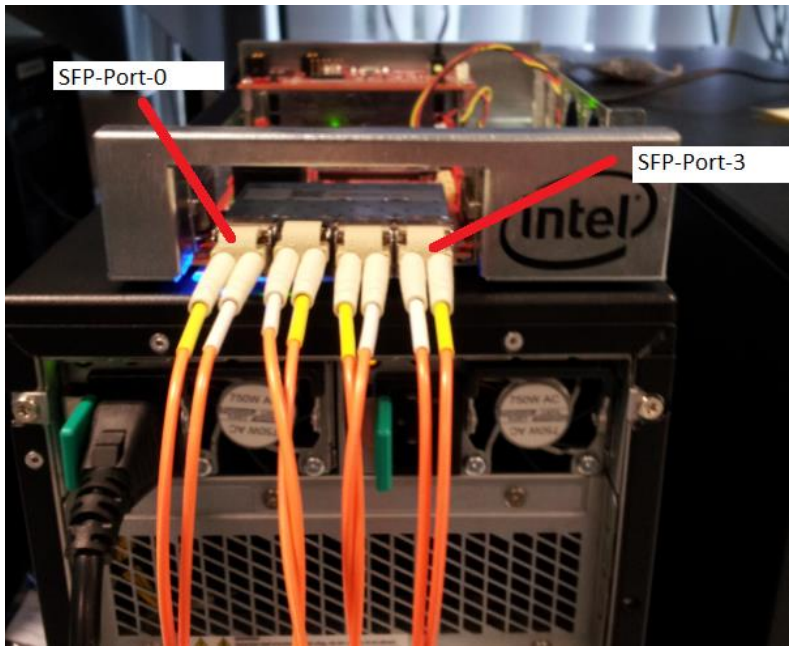


Figure 4-4. Intel® Front-End Development Kit



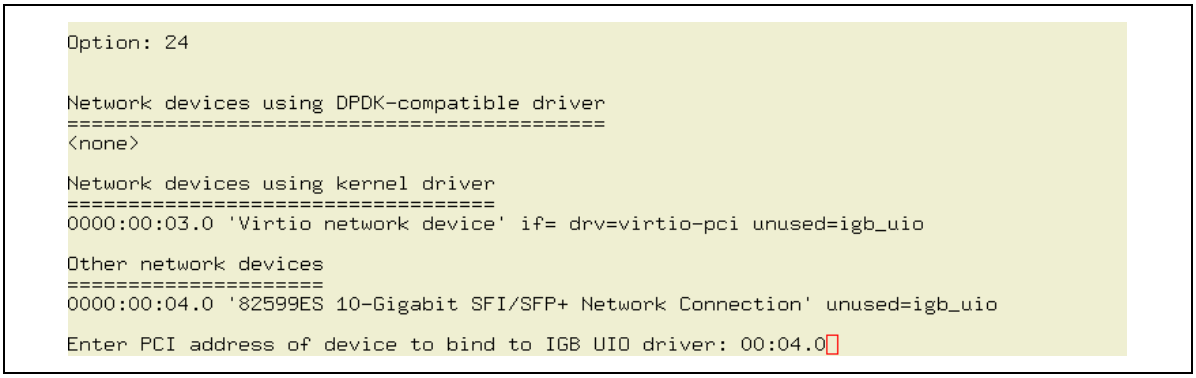
4.2.1 Configuring DPDK

The following configuration must be conducted on the eNodeB and UE VMs.

1. `cd /opt/intel/dpdk-18.08/usertools`
2. `./dpdk-setup.sh`
   - To build the DPDK module with gcc, select option 16.
   - To insert the module, select option 18.
   - To bind NIC ports to the module, select option 24. When prompted to enter the PCI address of the NIC port, enter `00:04.0`.

Note: This is the same for both VMs (`00:04.0`).

Figure 4-5. Configuring DPDK



a. To display the ports bound to NICs (for verification), select option 23.
b. To exit the script, select option 35.

Note: Every time the system is rebooted, follow the above steps for DPDK.

4.2.2 Ferry Bridge Build and Execution

Enter the following commands to build and run Ferry Bridge.

1. `export RTE_SDK=/opt/intel/dpdk-18.08/`
2. `export RPE_DIR=/opt/intel/ferrybridge/`
3. `cd /opt/intel/ferrybridge/lib`
4. Build Ferry Bridge library.
   a. `make clean`
   b. `make`
5. `cd /opt/intel/ferrybridge/app`
6. Build Ferry Bridge sample application.
   a. `make clean`
   b. `make`

4.2.3 Executing the Ferry Bridge SampleApp

There are two configuration files in `app/` directory: `config_file_rec.dat` and `config_file_re.dat`. Use configuration file fields as described below.

General SampleApp setup:

- `cpriMode` – select CPRI mode in which SampleApp will work (0 – REC, 1 – RE)
- `lteMode` – select LTE mode in which SampleApp will work (0 – LTE Normal, 1 – CPRI Bypass, 3 – LTE Bypass)
- `antNum` – select number of antennas configuration, 2x2 and 4x4 supported (2 – 2x2, 4 – 4x4)
- `numRbs` – number of Resource Blocks for each symbol; 100, 50 and 25 supported; if LTE Bypass or CPRI Bypass mode used `numRbs` should be set to 0
- `dlFftMode` – enable Downlink FFT Bypass if set to 1 (no FFT on DL side) – only for debug purposes
- `cpriLinkRate` – set CPRI Link Rate configuration to one of 4 available values (0 – default for given antennas number, 2.4G for two antennas and 6.14G for four antennas; 2 – 2.4G, 4 – 4.9G, 6 – 6.14G)

LTE specific setup (available only in LTE Normal mode):

- `enableRS` – set to 1 to enable Resource Signal feature
- `rsOffset` – Resource Signal offset for each antenna; if `antNum=2` then `rsOffset=`XX, if `antNum=4` then `rsOffset=`XXXX where X is number <0,5>
- `rsQpskVector` – 32-bits hexadecimal value of Resource Signal QPSK Vector to be used when RS feature is enabled
- `enablePrach` – set to 1 to enable PRACH feature
- `nRbPrach` – PRACH resource blocks number used for NCO calculations (part of PRACH configuration); 0 <= nRB <= 97
- `prachConfigIndex` – PRACH configuration index (0 – 15)
- `enablePbch` – set to 1 to enable PBCH feature
- `enablePss` – set to 1 to enable PSS feature

Start SampleApp using command:

```
./build/rpeSampleApp <config_file_name> -c1e -n2 -- -p1
```

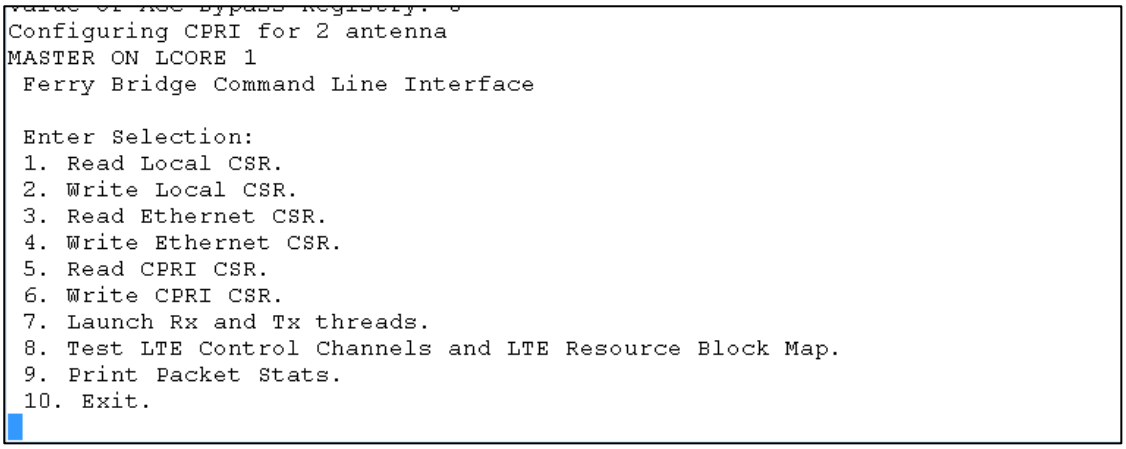
For example, REC application startup command line:

```
./build/rpeSampleApp config_file_rec.dat -c1e -n2 -- -p1
```

And RE application:

```
./build/rpeSampleApp config_file_re.dat -c1e -n2 -- -p1
```

Figure 4-6. Executing the Ferry Bridge SampleApp (1 of 2)



The menu above appears. This menu is the same for REC and RE applications.

- To run traffic, select option 7. This option must be run on both the REC and RE in order to establish the CPRI link.
- To send only two radio frames, select option 8
- To view the current status of the test, select option 9. (Note: The test continues to run.)

Figure 4-7. Executing the Ferry Bridge SampleApp (2 of 2)

```
############### Packet Stats#################
Total Number of Tx Packets                                  : 3760082
Total Number of Rx Packets                                  : 4278643
Total Number of Rx PUSCH Packets                            : 3630362
Number of Valid PUSCH Packets                               : 3630362
Number of inValid PUSCH Packets                             : 0
Number of inValid header PUSCH Packets                      : 0
Number of inValid data PUSCH Packets                        : 0
Total Timing Packets Rx's                                   : 129657
Valid Timing Packets Rx's                                   : 129657
InValid Timing Packets Rx's                                 : 0

Total PRACH Packets Port 0 Ant 0 Rx's                 : 129657
Valid PRACH Packets Port 0 Ant 0 Rx's                 : 129657
Invalid PRACH Packets Port 0 Ant 0 Rx's               : 0
Total PRACH Packets Port 0 Ant 1 Rx's                 : 129657
Valid PRACH Packets Port 0 Ant 1 Rx's                 : 129657
Invalid PRACH Packets Port 0 Ant 1 Rx's               : 0
Total PRACH Packets Port 0 Ant 2 Rx's                 : 129657
Valid PRACH Packets Port 0 Ant 2 Rx's                 : 129657
Invalid PRACH Packets Port 0 Ant 2 Rx's               : 0
Total PRACH Packets Port 0 Ant 3 Rx's                 : 129657
Valid PRACH Packets Port 0 Ant 3 Rx's                 : 129657
Invalid PRACH Packets Port 0 Ant 3 Rx's               : 0

Average time for 129657 pkts                    : 1000
Max time                                  : 1000
Min time                                  : 994
 Ferry Bridge Command Line Interface

 Enter Selection:
 1. Read Local CSR.
 2. Write Local CSR.
 3. Read Ethernet CSR.
 4. Write Ethernet CSR.
 5. Read CPRI CSR.
 6. Write CPRI CSR.
 7. Launch Rx and Tx threads.
 8. Test LTE Control Channels and LTE Resource Block Map.
 9. Print Packet Stats.
 10. Exit.
```

- To stop the test, select option 10. The packet stats are output at the end of the test.

§

Appendix A Updating FPGA Image

The EEPROM that stores the FPGA image in the FEDK can be updated using the embedded USB-Blaster* functionality on the FEDK and the Altera Quartus* programming tool. The Quartus Programmer can be installed on the UE server. Intel uses version 13.1; however, later versions should also be supported.

A.1 Obtain the Quartus* Programmer Installer

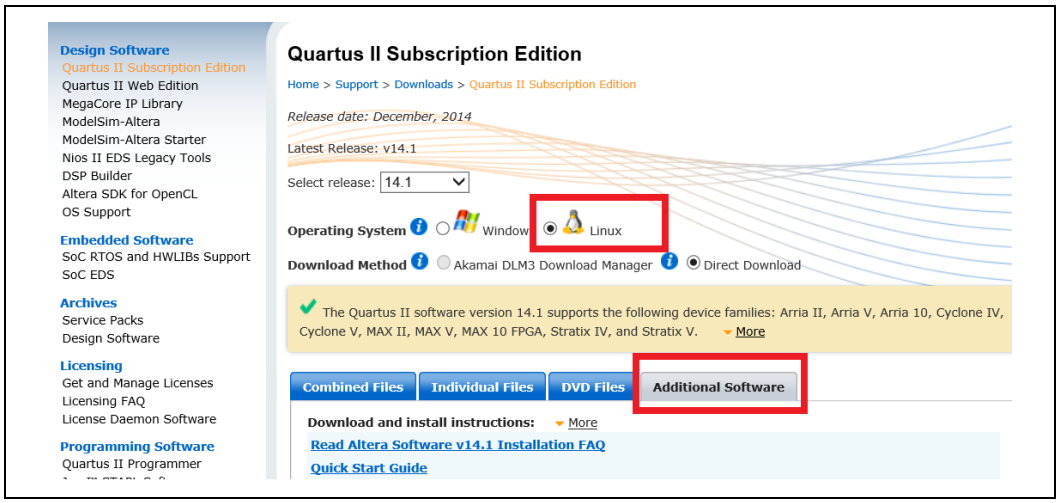
The Quartus Programmer can be downloaded for free from the Altera website at the following location:

http://dl.altera.com/?product=qprogrammer#tabs-4

To download the tool, you must first register with Altera.

1. On the download portal, click the **Additional Software** tab.

Figure 4-8. Download Portal



2. Select **Quartus II Programmer and Tools** and save the `QuartusProgrammerSetup-13.1.0.162.run` file. (You can download a later version. This 13.1 version is used by Intel.)
3. Log in to the Host UE server and create directory `/opt/altera/`. (NB ensure you are not logged in to VM.)
4. Copy the `QuartusProgrammerSetup-13.1.0.162.run` file to the `/opt/altera` directory on the Host UE server.
5. `cd /opt/altera`
6. `chmod 755 QuartusProgrammerSetup-13.1.0.162.run`

7. Install the software:
   a. `./ QuartusProgrammerSetup-13.1.0.162.run`
   b. Press **Enter** several times to scroll through the license agreement and enter **accept** at the final prompt.
8. Select the install directory to be `/opt/altera/`.

The command-line version of the Altera tool is now installed at `/opt/altera/`.

A.2 Upgrade the FPGA Image

The R1.1.0 FPGA image is located at `/opt/altera/fpga_bin`.

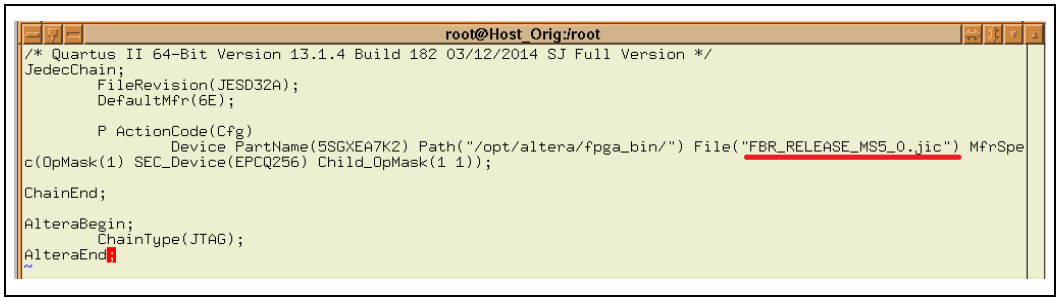
To upgrade the FPGA EEPROM image on the FEDK:

1. Connect the USB cable to the micro-USB port at the rear of the FEDK.
2. Connect the USB cable to the left-front USB port on the Canoe Pass server.

Figure 4-9. Left-Front USB Port on the Canoe Pass Server

3. The .jic image file is located in the `/opt/intel/fpga_bin` directory on the host-eNodeB server.
4. Edit the `AMC_EEPROM.cdf` file to point to the updated .jic image file.

Figure 4-10. AMC_EEPROM.cdf File



5. Execute the following command to verify connectivity with the FPGA and identify the USB port utilized. (USB-Blaster [1-1.4] is shown in the example below.)

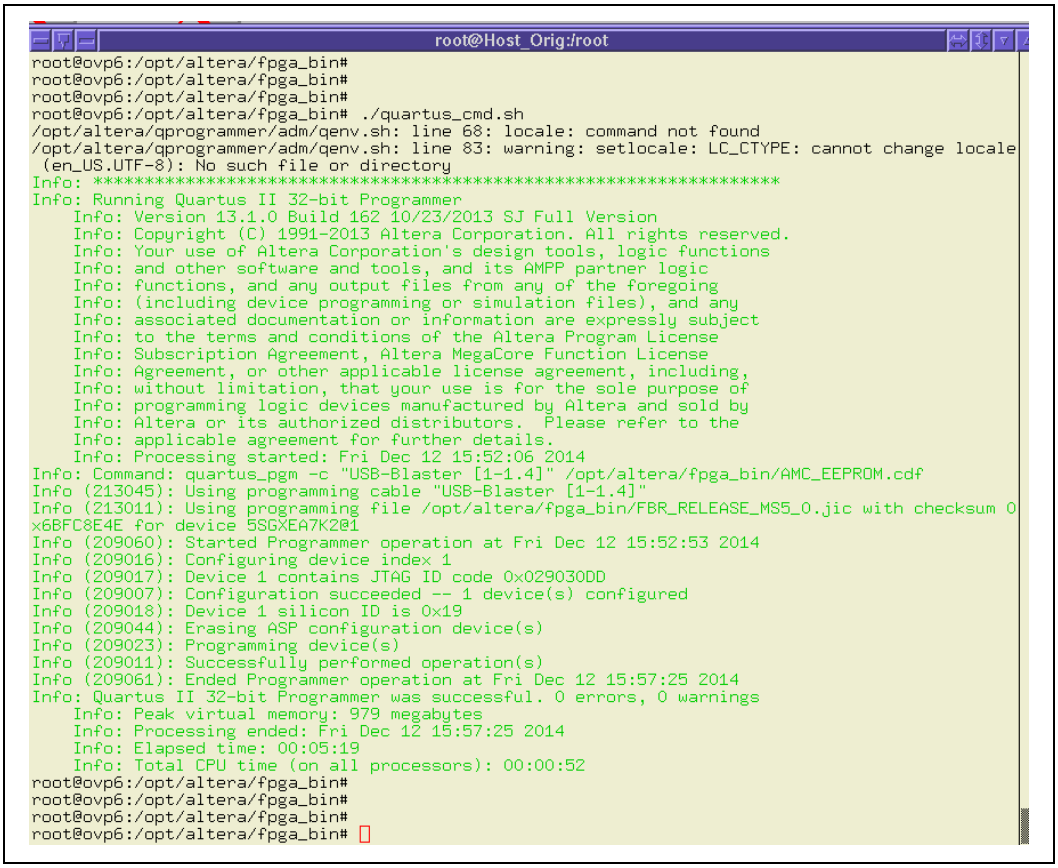
```
/opt/altera/qprogrammer/bin/quartus_pgm –a
```

6. Execute the Quartus programming command:

```
/opt/altera/qprogrammer/bin/quartus_pgm -c \"USB-Blaster [1-1.4]\"
/opt /intel/fpga_bin/AMC_EEPROM.cdf
```

The programming process takes approximately 15 minutes.

Figure 4-11. Programming Process



7. Once completed, remove the USB cable, and power-cycle the FEDK to load the updated configuration from the EEPROM to the FPGA.

A.3 Updating using Terasic* Dev Card

The download of the image to the Terasic* card is done using a different procedure from the AMC/FEDK card and is performed using flash images.

Download and install the NIOS command line tools from Altera.

1. Locate where `SOPC_KIT_NIOS2` is installed on your system and add it to your `${PATH}` if it is not the case already.
2. Ensure USB cable is connected from PC to the target DE5-NET dev card.
3. Enter `source ./flash_program_bashrc_ub2.bash`

   This file downloads the parallel flash loader and programs three sections to the on-board flash:

   - fbr_flash_hw.flash
   - fbr_flash_sw.flash
   - S5_OptionBits.flash

This procedure may take up to 15-minutes.

4. Power-cycle the card to load the image stored in on-board flash.

§