

CASPER TOOLFLOW

- Installation Guide for Vivado and Vitis according to casper toolflow.

UBUNTU 20.04 LTS

a. Install Ubuntu **20.04 LTS** on your device. (Windows OS are not recommended)

b. For Ubuntu installation follow: <https://www.youtube.com/watch?v=C5deqtXrpgk&t=469s>

- After successfully Ubuntu 20.04, there comes time to install some dependency before installing **MATLAB R2021a**. Note: **Don't install MATLAB before downloading following things**.

Before Installing MATLAB, we must do some configuration, need to install additional libraries.

For Ubuntu 20.04 do the following before installing Matlab and Vivado:

Install the QT4 libraries from the rock-core ppa

e.g.,

sudo add-apt-repository ppa:rock-core/qt4

sudo apt install libqtcore4 libqtgui4

Follow the guidelines of the following image:

Supported MATLAB Versions and Operating Systems

Vitis Model Composer supports the following MATLAB versions:

- R2020a
- R2020b
- R2021a

The following operating systems are supported on x86 and x86-64 processor architectures:

Windows 10 Pro and Enterprise

10.0 1903 Update; 10.0 1909 Update; 10.0 2004 Update; 10.0 20H2; 10.0 21H1

Red Hat Enterprise Workstation/Server 7

7.8; 7.9

Ubuntu Linux

18.04.4 LTS; 18.04.5 LTS; 20.04 LTS

 **Note:** MATLAB 2020a version is not supported on Ubuntu 20 OS.

Prerequisites for using Model Composer on Ubuntu 20 OS

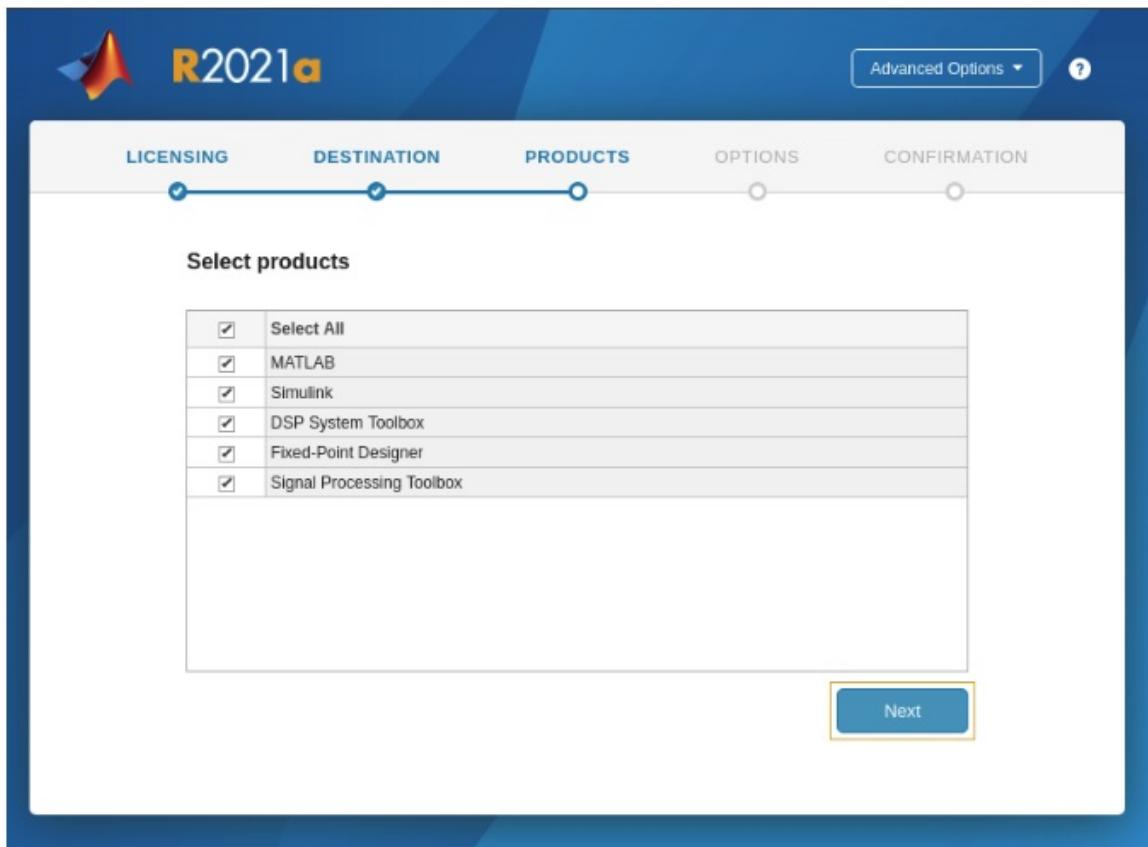
- QT4 library should be installed.
- Ubuntu 20 comes with gcc 7.x to 9.x versions by default; you need to either install gcc 6.x manually or create a symbolic link using the following sudo commands:
 - `sudo ln -s /usr/include/asm-generic /usr/include/asm`
 - `sudo ln -s /usr/include/x86_64-linux-gnu/sys /usr/include/sys`
 - `sudo ln -s /usr/include/x86_64-linux-gnu/bits /usr/include/bits`
 - `sudo ln -s /usr/include/x86_64-linux-gnu/gnu /usr/include/gnu`
- Ubuntu 20 by default comes with dash shell. To avoid any issue that you may encounter while running the downstream AI Engine flows, it is recommended to change the shell from dash to bash using the following sudo command:
 - `sudo dpkg-reconfigure dash`

MATLAB R2021a

a. Download Matlab for linux version from mathworks.com

b. Prior to that make your personal account.

c. I've installed 5 packages of MATLAB: MATLAB, Simulink, DSP System Toolbox, Fixed-Point Designer, Signal processing Toolbox.
Install only required packages (Save storage in your computer)



d. For MATLAB installation follow: <https://www.youtube.com/watch?v=ZNHJkCo5sOc&t=461s>

XILINX VIVADO/VITIS from AMD (we need vitis)

- After successfully installing of Matlab, we are moving toward other software Xilinx Vivado ML edition.: amd.com (make account for this)

Version	We strongly recommend using the latest releases available.
2023.2	
2023.1	
2022.2	
2022.1	
Vivado Archive	
ISE Archive	
CAE Vendor Libraries Archive	

2021

2021.1

Vivado ML Edition - 2021.1 Full Product Installation

Important Information

Vivado ML 2021.1 is now available for download.

Installation guide: Dr. Homin Github

<https://github.com/asiaa/BURSTT/wiki/Installing-mlib-devel-2021>

a. Make sure you download Xilinx Unified Installer 2021.1 Linux Self Extracting Web Installer



After [downloading from amd.com](#), before installing use these 2 commands:

- sudo apt install libtinfo5
- sudo apt install libncurses5

Note: If not, then your final installation won't happen and you end up downloading 45 Gb and can't do final processing, you have to download again.

Installation is very easy. Go to the folder where you downloaded it (VITIS).

- For me, I kept it in download folder. Download>Xilinx_Unified_2021.1_0610_2318_Lin64.bin
– to install it:

CMD: cd *home/Download*

- ls

check if the file you want to install is there or not

```
cd /path/to/matlab/download/matlab_R2021a_glnxa64.zip
mkdir matlab_R2021a
unzip matlab_R2021a_glnxa64.zip -d matlab_R2021a
cd matlab_R2021a
sudo ./install
```

Xilinx_Unified_2021.1_0610_2318_Lin64.bin it will pop up xilinx installer on your window.

RFSOC4x2 is a Dual- Tile

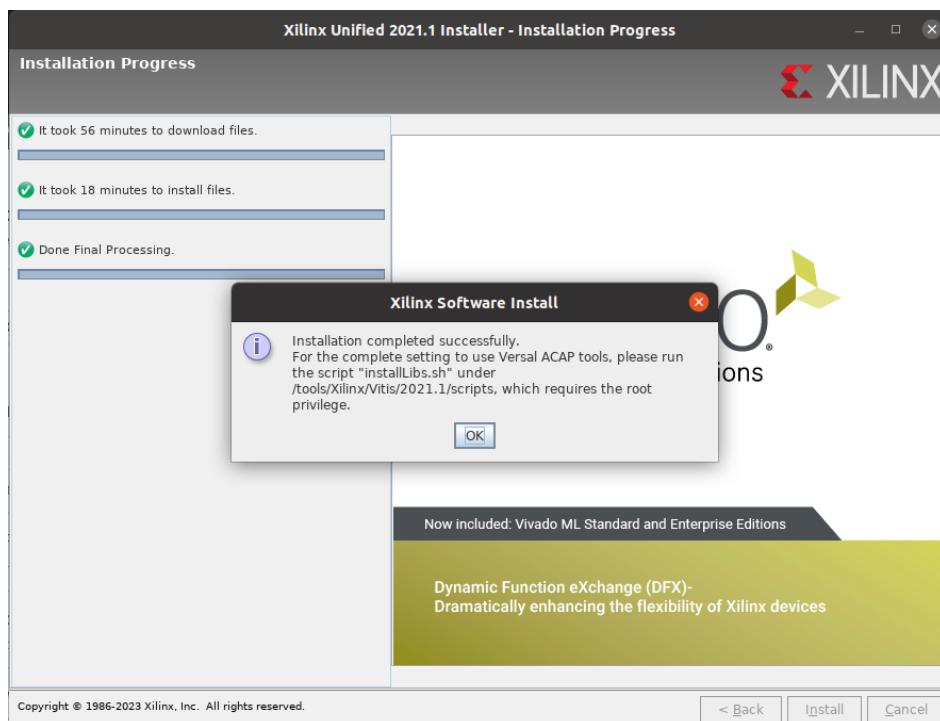
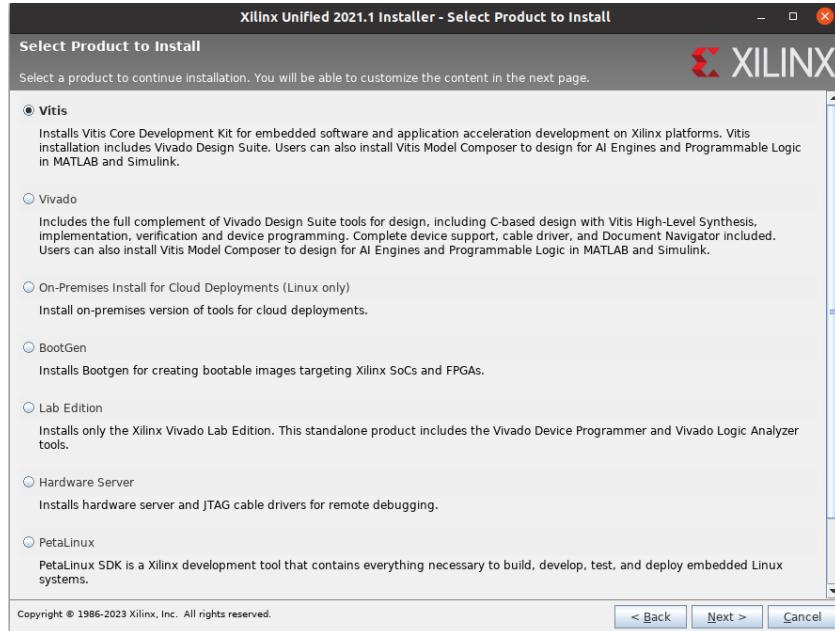
Each ADC tile have 2 inputs. 4 x 2 = 8 inputs

ZCU is quad tile ADC

Each ADC have 4 inputs = 4x4 = 16 inputs

```
bishnu@bishnu:/tools/Xilinx/Vivado/2021.1/bin$ sudo ./vivado
*****
** Vivado v2021.1 (64-bit)
** SW Build 3247384 on Thu Jun 10 19:36:07 MDT 2021
** IP Build 3246043 on Fri Jun 11 00:30:35 MDT 2021
** Copyright 1986-2021 Xilinx, Inc. All Rights Reserved.

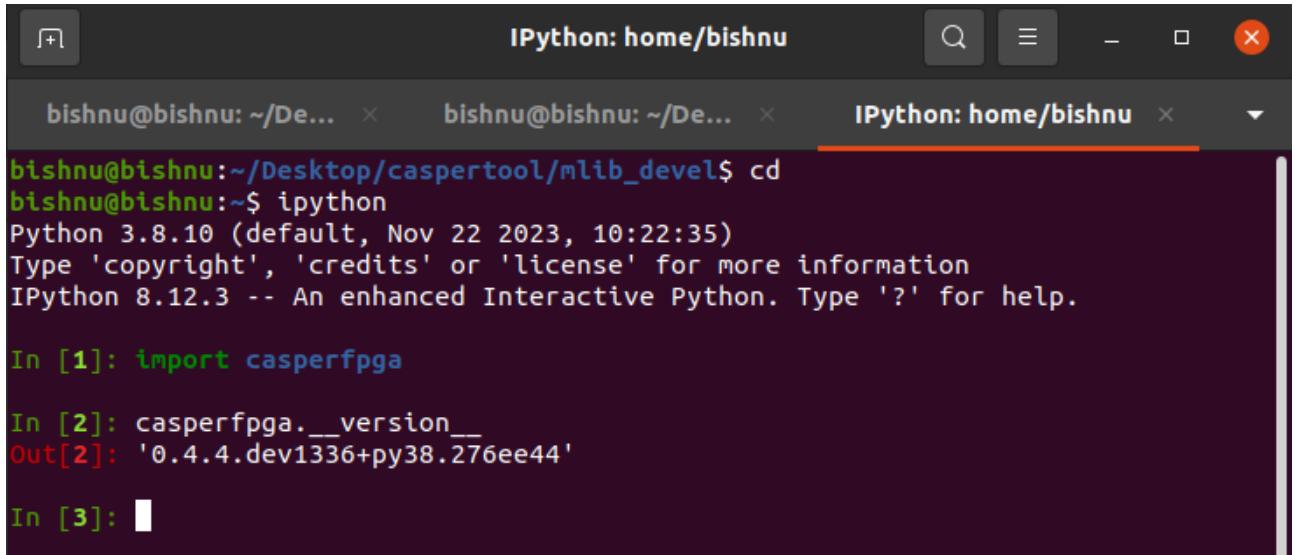
start_gui
```



```
bishnu@bishnu:/tools/Xilinx/Vitis/2021.1/scripts$ ./installLibs.sh
This script must be run as root
bishnu@bishnu:/tools/Xilinx/Vitis/2021.1/scripts$ sudo ./installLibs.sh
[sudo] password for bishnu:
touch: cannot touch '/root/.Xilinx/xinstall/installLibs.sh_2023-12-01_09-04-59':
Permission denied
chmod: cannot access '/root/.Xilinx/xinstall/installLibs.sh_2023-12-01_09-04-59':
: Permission denied
No LSB modules are available.
ubuntu-20 install
Hit:1 https://dl.google.com/linux/chrome/deb stable InRelease
Hit:2 http://tw.archive.ubuntu.com/ubuntu focal InRelease
Get:3 http://tw.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Hit:4 http://tw.archive.ubuntu.com/ubuntu focal-backports InRelease
Get:5 http://tw.archive.ubuntu.com/ubuntu focal-updates/main i386 Packages [913 kB]
Get:6 http://tw.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [299 4 kB]
```

Note: For the complete setting to Versal ACAP tools, please run the script “installLibs.sh” under tools/Xilinx/Vitis/2021.1/scripts using sudo (root privilege).

- Let's start activating Vivado/Vitis. Follow the same you tube link which in mentioned above. Upload license from load license, once you open vivado.
 (Method2: you can use ./vlm command to open license and check about your licenses)



```
bishnu@bishnu:~/Desktop/caspertool/mlib-devel$ cd
bishnu@bishnu:~$ ipython
Python 3.8.10 (default, Nov 22 2023, 10:22:35)
Type 'copyright', 'credits' or 'license' for more information
IPython 8.12.3 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import casperfpga

In [2]: casperfpga.__version__
Out[2]: '0.4.4.dev1336+py38.276ee44'

In [3]:
```

Care should be taken when you make virtual environment.

-----Important read it:

<https://strath-sdr.github.io/tools/matlab/sysgen/vivado/linux/2021/01/28/sysgen-on-20-04.html>

```

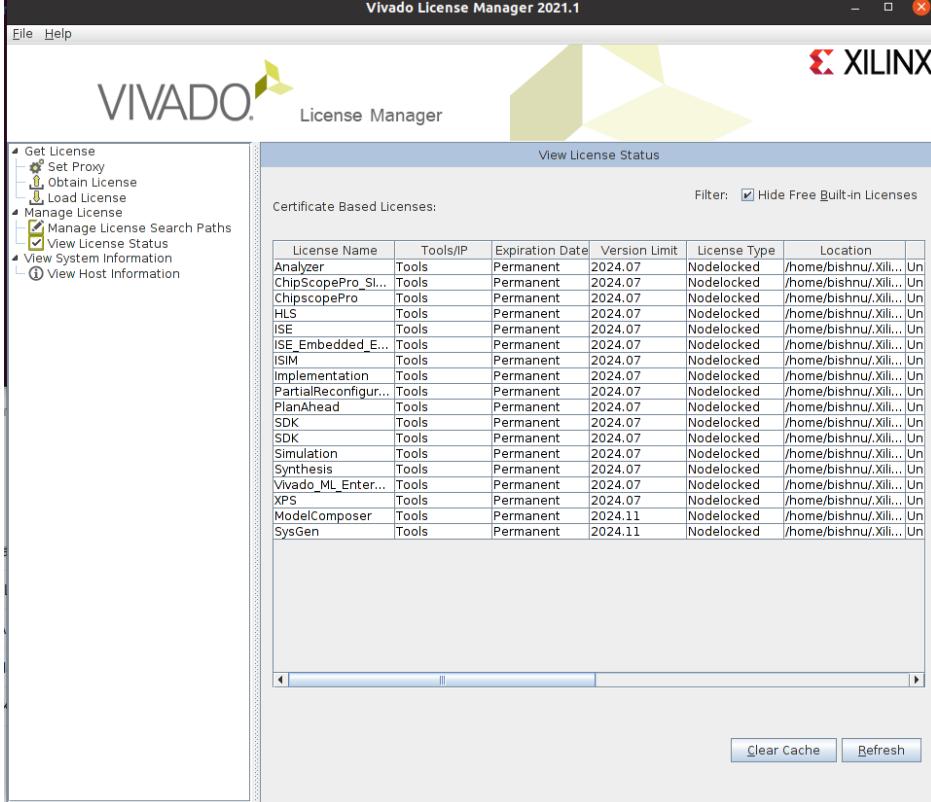
Command Window
  procedure ..sw_cpu_cortexa53..generate  time 19)
invoked from within
":::sw_cpu_cortexa53::generate psu_cortexa53_0"
ERROR: [Hsi 55-1442] Error(s) while running TCL procedure generate()
ERROR: [Common 17-39] 'hsi::generate_target' failed due to earlier errors.

while executing
"hsi::generate_target -dir $jdtc_dir"
  (file "/home/bishnu/Desktop/m_work/tutorials-devel/rfsoc/tut_rfcd/rfsoc4x2/rfsoc4x2_tut_rfcd_real/xsct_gogogo.tcl" line 12)

Traceback (most recent call last):
  File "/home/bishnu/Desktop/caspertool/mlib-devel/jasper_library/exec_flow.py", line 348, in <module>
    vitis.compile()
  File "/home/bishnu/Desktop/caspertool/mlib-devel/jasper_library/toolflow.py", line 1558, in compile
    raise Exception('xsct (Vitis) failed!')
Exception: xsct (Vitis) failed!
Error using jasper (line 23)
Backend build failed! Check log files for more information

```

I redownload the Xilinx-device-tree repository with branch xlnx_rev_v2021.1 and it seems it works. I am currently running RFSoC 4x2 from tutorial devel. Thank you.



When you install Vitis, it will ask to select Design Tools and Devices:

Select all:

Vitis Unified Software platform and all

Select Engineering Sample Devices for ZCU216.

bishnu@bishnu: /tools/Xilinx/Vivado/2021.1/bin\$ cd /tools/Xilinx/Vivado/2021.1/
bishnu@bishnu:/tools/Xilinx/Vivado/2021.1\$ cd bin/
bishnu@bishnu:/tools/Xilinx/Vivado/2021.1/bin\$ ls
apcc loader setupEnv.sh wbtcv xsc
bootgen manage_ipcache svf_utility xar xsdb
bootgen_utility p4bm-vitisnet symbol_server xcd xsim
cdoutil p4bm-vitisnet-cli tcflg xcrg xtclsh
cdoutil_int p4c-sdnet unsetldlibpath.sh xelab xvc_pcie
cs_server p4c-vitisnet unwrapped xip_hls xvhd़
diffbd program_ftdi updatemem xiphlsArgs.sh xvlog
hw_server rdiArgs.sh vcse_server xlicdiag
hw_serverpv run-p4bm-vitisnet vivado xrcserver
ldlibpath.sh setEnvAndRunCmd.sh vlm xrt_server
bishnu@bishnu:/tools/Xilinx/Vivado/2021.1/bin\$./vlm
bishnu@bishnu:/tools/Xilinx/Vivado/2021.1/bin\$

```
$ sudo apt update  
$ sudo apt install software-properties-common  
$ sudo add-apt-repository ppa:rock-core/qt4  
$ sudo apt update  
$ sudo apt install libqtcore4 libqgtgui4  
$ sudo apt install libtinfo-dev
```

```
$ cd /opt/Xilinx/Vivado/2021.1/lib/lnx64.o/Ubuntu  
$ sudo mkdir exclude  
$ sudo mv libgmp.so* exclude  
  
$ cd /opt/Xilinx/Model_Composer/2021.1/lib/lnx64.o/Ubuntu  
$ sudo mkdir exclude  
$ sudo mv libgmp.so* exclude  
  
$ cd /opt/Xilinx/Vivado/2021.1/lib/lnx64.o
```

```
$ sudo mkdir exclude  
$ sudo mv libgmp.so* exclude
```

TL;DR

Vivado brings a conflicting version of `libgmp` into your environment. You can avoid this by moving them to a different folder.

```
sdr@strath$ cd $XILINX_VIVADO/lib/lnx64.o/Ubuntu/  
sdr@strath$ mkdir exclude  
sdr@strath$ mv libgmp.so* exclude/
```

You'll also need to install `qt4` from a PPA (the official repos only have `qt5` now).

```
sdr@strath$ sudo add-apt-repository ppa:rock-core/qt4  
sdr@strath$ sudo apt update  
sdr@strath$ sudo apt install libqtc... libqtgui4
```

That's all. Hopefully you can use System Generator now!

This will solve some installation issue and conflict with Vitis/ Vivado.

To activate Vitis, Vivado, Model Composer, Vivado_HLS follow same steps as shown In below command.

Note: I did same for all, and it works for me, other depends upon you and your requirements.

```
bishnu@bishnu:/tools/Xilinx/Vitis/2021.1$ source /tools/Xilinx/Vitis/2021.1/settings64.sh  
bishnu@bishnu:/tools/Xilinx/Vitis/2021.1$
```

Try this making different folders, and follow this
I think, this information are enough for installation of the Vitis Unified installation.

NEXT CHAPTER SETTING CASPER TOOLFLOW

Virtual environment and install requirements.txt from mlib_devel and casperfpga both in Virtual environment. All the requirements

NOTE: Python3.8 must be installed in Virtual environment.

- Virtual environment enable us to install desired version of libraries and tools without disturbing our main root file. It help to run things smoothly and save Ubuntu environment.

Make a folder in desktop eg: fpga_venv, venv_casper, my_venv or anything.

casperfpga & python venv

- **required module**

```
$ sudo apt install python3.8
$ sudo apt install python3-pip
$ sudo apt install python3.8-venv
```

NOTE: NumPy==1.20 and Tornado==4.5.3 version should be as mentioned in the screenshot images.

Numpy version will impact MATLAB's function.

- Activate virtual environment first.
- Install casperfpga, mlib devel and device tree xlnx branch 2021.

For RFSoC 4x2 kit. If you want to do it for later kit like 4x2, you are upto yourself.

- https://casper-toolflow.readthedocs.io/projects/tutorials/en/latest/tutorials/rfsoc/tut_getting_started.html

- Download links from github of Casper-astro (git clone)
- <https://github.com/casper-astro>
- <https://github.com/Xilinx/device-tree-xlnx/> (v2021.1 branch)
- <https://github.com/casper-astro/mlib-devel>
- <https://github.com/casper-astro/casperfpga>
- <https://github.com/casper-astro/tutorials-devel>

I hope this can help you a lot and decrease your suffering during installation.

Installing casperfpga using a virtual environment

```
$ python3.8 -m venv ./cfpga_venv
$ source ./cfpga_venv/bin/activate
$ git clone https://github.com/casper-astro/casperfpga
$ cd casperfpga/
$ git checkout py38
$ pip install -r requirements.txt
$ pip install .
```

```
### Testing that the installation worked

To check that casperfpga has been installed correctly open an ipython session and import casper

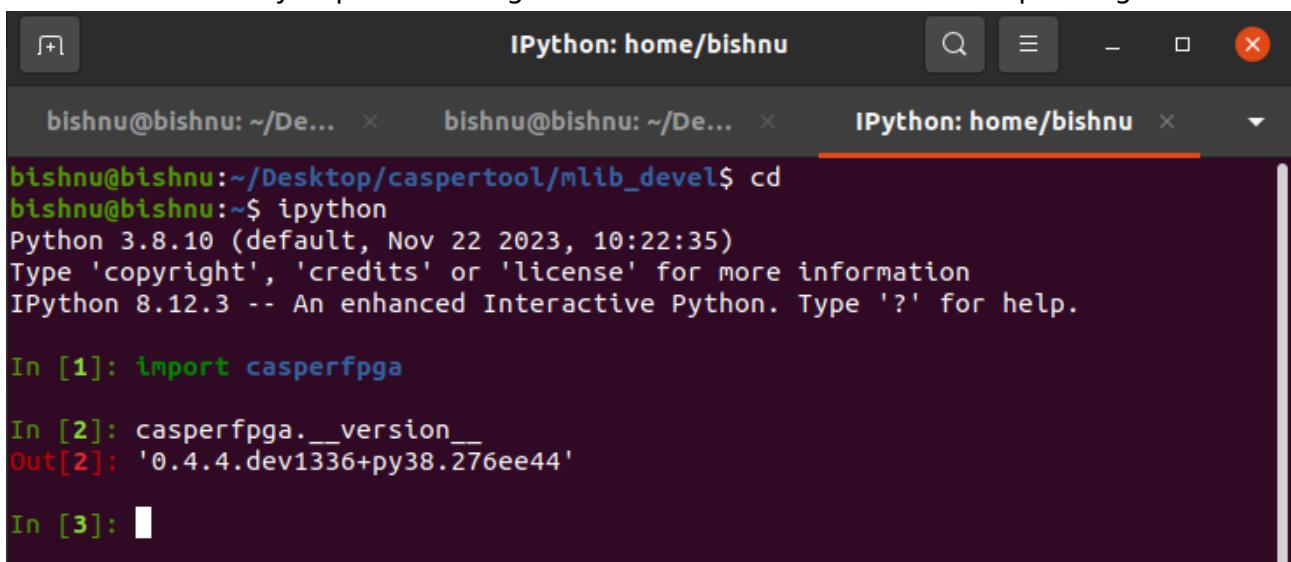
```shell
$ cd ..
$ ipython
```

```
In [1]: import casperfpga
In [2]: casperfpga.__version__
```

- To deactivate virtual environment, just type “deactivate” in the command terminal.

This device tree is very important and save it parallel with mlib-devel. (See image somewhere in the tutorial) [https://github.com/casper-astro/tutorials\\_devel/tree/main](https://github.com/casper-astro/tutorials_devel/tree/main)

**This link** gives a compatibility matrix of supported OS and MATLAB/Vivado combinations, though I've found that developing for SKARAB using MATLAB R2018a and Vivado 2019.1 is the most reliable if you plan on using a SKARAB ADC module. The corresponding



The screenshot shows a terminal window with three tabs. The active tab is titled "IPython: home/bishnu". The command history shows:

```
bishnu@bishnu:~/Desktop/caspertool/mlib-devel$ cd
bishnu@bishnu:~$ ipython
Python 3.8.10 (default, Nov 22 2023, 10:22:35)
Type 'copyright', 'credits' or 'license' for more information
IPython 8.12.3 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import casperfpga

In [2]: casperfpga.__version__
Out[2]: '0.4.4.dev1336+py38.276ee44'

In [3]:
```

mlib-devel branch for these versions can be found [here](#) if you want to use those ones - IIRC this will still require a Python 3.8 environment to run the compile. You will need to use

the [Python 2.7 version of casperfpga](#) to interact with the SKARABs. So you need a Python 3.8 virtual environment for mlib\_devel, and Python 2.7 for casperfpga

- **Installing casperfpga (virtual environment)**

```
change directory to where you want the virtual environment to live
$ cd /home/corr/work #self defined location

create a Python 3.8 virtual environment
$ python3.8 -m venv ./cfpga_venv

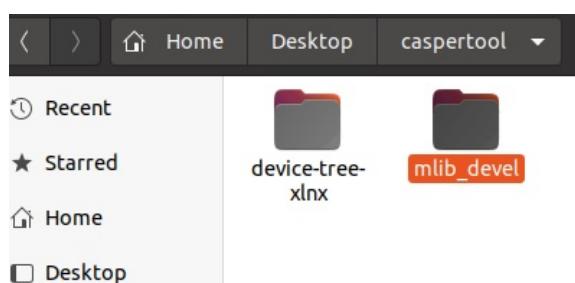
to activate the virtual environment:
$ source ./cfpga_venv/bin/activate

solve the ERROR below (virtual environment)
$ pip3 install wheel
$ sudo apt install python2 #for System Generator
$ pip install --upgrade pip
$ sudo apt install python3-dev build-essential
$ pip3 uninstall tornado
$ pip3 install tornado==4.5.3

get version (for MATLAB ERROR)
$ pip3 show numpy | grep Version
 Version : 1.24.4
#install 1.2的版本
$ pip3 install numpy==1.20

$ git clone https://github.com/casper-astro/casperfpga
$ cd casperfpga/
$ git checkout py38
$ pip install -r requirements.txt
$ pip install . #它會檢查 setup.py 檔案中定義的相依性，並自動下載並安裝所需的套件。

to deactivate the virtual environment:
$ deactivate
```



```
bishnu@bishnu:~/Desktop/caspertool$ git clone -b xlnx_rel_v2021.1 https://github.com/Xilinx/device-tree-xlnx.git
Cloning into 'device-tree-xlnx'...
remote: Enumerating objects: 14828, done.
remote: Counting objects: 100% (6196/6196), done.
remote: Compressing objects: 100% (1492/1492), done.
remote: Total 14828 (delta 3933), reused 6051 (delta 3830), pack-reused 8632
Receiving objects: 100% (14828/14828), 2.94 MiB | 10.69 MiB/s, done.
Resolving deltas: 100% (8270/8270), done.
bishnu@bishnu:~/Desktop/caspertool$
```

```
bin@laptop:~/Desktop/caspertool$./caspertool --gen-jasper dt
INFO: [Hsi 55-2053] elapsed time for repository (/tools/Xilinx/Vitis/2021.1/data/embeddedsw) loading 0 seconds
WARNING: Frequency 33.3333 used instead of 33.333
WARNING: Frequency 33.333 used instead of 27.00
zocl:false
ext_platform:
WARNING: psu_usb_xhci_0: No reset found
WARNING: label 'usb0' found in existing tree
WARNING: psu_usb_xhci_1: No reset found
WARNING: label 'usb1' found in existing tree
hsilog:generate_target: Time (s): cpu = 00:00:16 ; elapsed = 00:00:18 . Memory (MB): peak = 2132.570 ; gain = 0.000 ; free physical = 10310 ; free virtual = 24415
Assembling jasper dt node
INFO: [Hsi 55-2053] Assembling jasper dt node
Created /home/bishnu/Desktop/m_work/tutorials-devel/rfsoc/tut_rfcd/rfsoc4x2/rfsoc4x2_tut_rfcd_cx/outputs/rfsoc4x2_tut_rfcd_cx_2023-12-01_0953.fpg
=====
* Backend complete!
```

- From mlibdevel folder, we start matlab. ./startsg startsg.local to open matlab
- You have to make startsg.local file by yourself according to your requirement.

## Create a New Model

Make sure you are in your previously set up environment and navigate to `mlib_devel`. Start Matlab by executing `startsg`. This will properly load the Xilinx and CASPER libraries into Simulink, so long as your `startsg.local` file is set correctly. Within Matlab, start Simulink by typing `simulink` into Matlab's command line. Create a new blank model and save it with an appropriate name. **The name cannot contain capital letters. If you use capital letters your design will not compile correctly.**

There are also premade starter simulink designs that can help you get started. You can find them in the `tut_platform` folder [here](#).



```
1 export XILINX_PATH=/tools/Xilinx/Vivado/2021.1
2 export MATLAB_PATH=/usr/local/MATLAB/R2021a
3 export COMPOSER_PATH=/tools/Xilinx/Model_Composer/2021.1
4 export PLATFORM=linux
5 #export JASPER_BACKEND=vivado
6 export JASPER_BACKEND=vitis
7 export XLNX_DT_REPO_PATH=/home/bishnu/Desktop/caspertool/device-tree-xlnx
8 export CASPER PYTHON_VENV_ON_START=/home/bishnu/Desktop/m_work/casper_venv
9 #export LD_PRELOAD=${LD_PRELOAD}:"/usr/lib/x86_64-linux-gnu/libexpat.so"
10 |
```

Back-end must be Vitis.

Go to command line terminal, activate your virtual environment, and then take it to mlib devel and type ./startsg startsg.local

- It will pop up matlab,
- Select tutorial devel from your computer, select rfsoc and play .slx file.
- In the Matlab terminal, type jasper\_frontend and then again just jasper.
- You should generate 2 files in the outputs, .dtbo and .fpg file.

---- BEST WISHES----

- This note is a self note. I didn't cite any authors from where I took information.

192.168.50.1

 You

```
sudo dpkg -i libc6_2.31-0ubuntu9.9/ubuntu.focal.custom_amd64.deb
```

 ChatGPT

It looks like you're attempting to install a custom version of the `libc6` package on a system running Ubuntu 20.04 (Focal Fossa). Before proceeding, it's essential to be cautious when installing custom packages, as they may not be officially supported or tested by the Ubuntu maintainers.

The command you provided uses `dpkg` to install the `libc6` package from a `.deb` file. Here's a breakdown of the command:

- `sudo`: Run the command with superuser (administrator) privileges.
- `dpkg -i`: Install the specified Debian package.
- `libc6_2.31-0ubuntu9.9/ubuntu.focal.custom_amd64.deb`: The filename of the Debian package you are installing.

If you encounter any issues during or after the installation, be aware that it might lead to a broken system, as `libc6` is a critical system library. Ensure that the custom package is compatible with your Ubuntu version and architecture.

Before proceeding, you may want to check if there are official updates or security patches available for `libc6` from the standard Ubuntu repositories. If a custom version is necessary, make sure it comes from a reliable and trustworthy source.

If you face any issues or have specific requirements, providing more details or context could help me offer more targeted assistance.

```

Command Window
 procedure ..sw_cpu_cortexa53..generate time 19)
invoked from within
":::sw_cpu_cortexa53::generate psu_cortexa53_0"
ERROR: [Hsi 55-1442] Error(s) while running TCL procedure generate()
ERROR: [Common 17-39] 'hsi::generate_target' failed due to earlier errors.

while executing
"hsi::generate_target -dir $jdtc_dir"
 (file "/home/bishnu/Desktop/m_work/tutorials-devel/rfsoc/tut_rfcd/rfsoc4x2/rfsoc4x2_tut_rfcd_real/xsct_gogogo.tcl" line 12)

Traceback (most recent call last):
 File "/home/bishnu/Desktop/caspertool/mlib-devel/jasper_library/exec_flow.py", line 348, in <module>
 vitis.compile()
 File "/home/bishnu/Desktop/caspertool/mlib-devel/jasper_library/toolflow.py", line 1558, in compile
 raise Exception('xsct (Vitis) failed!')
Exception: xsct (Vitis) failed!
Error using jasper (line 23)
Backend build failed! Check log files for more information

```

The error mentioned below is due to some broken file check cd usr/include/sys and type ls -l and you will see red files like in figure below.

```

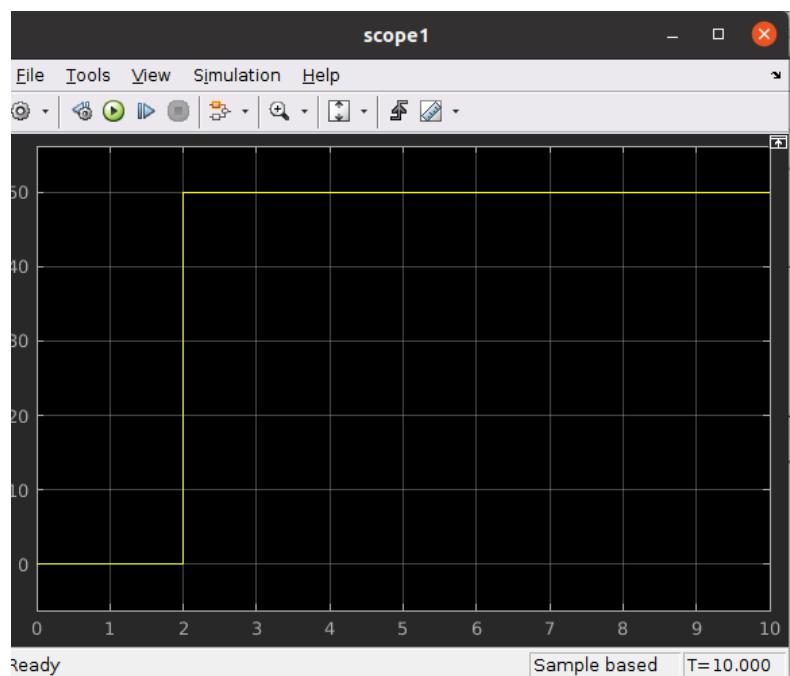
bishnu@bishnu:/usr/include/sys$ ll
total 16
drwxr-xr-x 2 root root 12288 + 4 18:28 .
drwxr-xr-x 8 root root 4096 + 29 11:04 ..
lrwxrwxrwx 1 root root 30 t 26 15:44 acct.h -> ../../x86_64-linux-gnu/sys/acct.h
lrwxrwxrwx 1 root root 30 t 26 15:44 auxv.h -> ../../x86_64-linux-gnu/sys/auxv.h
lrwxrwxrwx 1 root root 33 t 26 15:44 bitypes.h -> ../../x86_64-linux-gnu/sys/bitypes.h
lrwxrwxrwx 1 root root 31 t 26 15:44 cdefs.h -> ../../x86_64-linux-gnu/sys/cdefs.h
lrwxrwxrwx 1 root root 34 t 26 15:44 debugreg.h -> ../../x86_64-linux-gnu/sys/debugreg.h
lrwxrwxrwx 1 root root 29 t 26 15:44 dir.h -> ../../x86_64-linux-gnu/sys/dir.h
lrwxrwxrwx 1 root root 29 t 26 15:44 elf.h -> ../../x86_64-linux-gnu/sys/elf.h
lrwxrwxrwx 1 root root 31 t 26 15:44 epoll.h -> ../../x86_64-linux-gnu/sys/epoll.h
lrwxrwxrwx 1 root root 31 t 26 15:44 errno.h -> ../../x86_64-linux-gnu/sys(errno.h

```

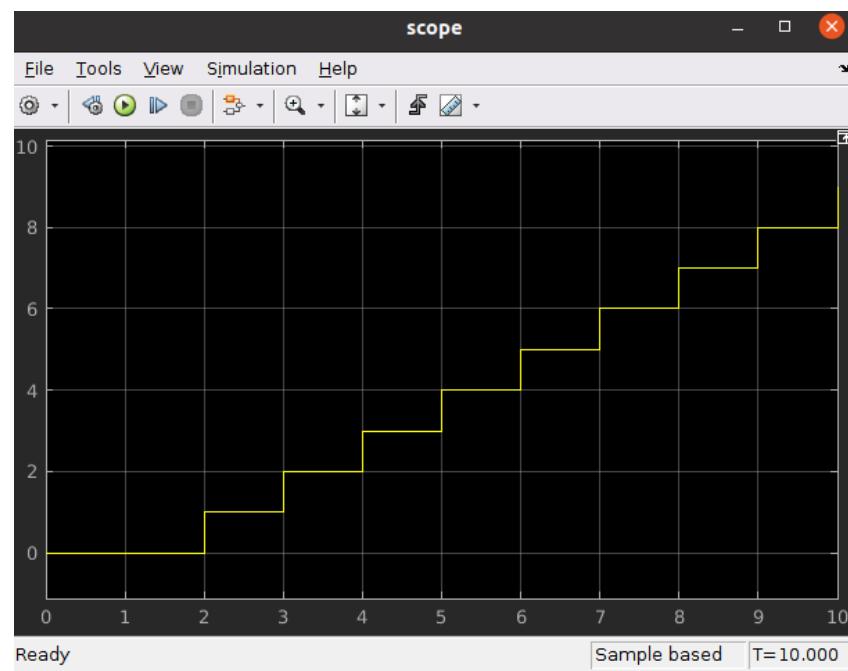
```

bishnu@bishnu:/usr/include/sys$ sudo apt-get install --reinstall libc6-dev
[sudo] password for bishnu:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
 python2-minimal python2.7 python2.7-minimal
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 1 reinstalled, 0 to remove and 0 not upgraded.
Need to get 2519 kB of archives.
After this operation, 0 B of additional disk space will be used.
Get:1 http://tw.archive.ubuntu.com/ubuntu focal-updates/main amd64 libc6-dev amd64 2.31-0ubuntu9.12 [2519 kB]
Fetched 2519 kB in 1s (3337 kB/s)
(Reading database ... 190080 files and directories currently installed.)
Preparing to unpack .../libc6-dev_2.31-0ubuntu9.12_amd64.deb ...
Unpacking libc6-dev:amd64 (2.31-0ubuntu9.12) over (2.31-0ubuntu9.12) ...
Setting up libc6-dev:amd64 (2.31-0ubuntu9.12) ...
bishnu@bishnu:/usr/include/sys$ ls -l
total 456
-rw-r--r-- 1 root root 3302 t 26 15:44 acct.h
-rw-r--r-- 1 root root 1283 t 26 15:44 auxv.h
-rw-r--r-- 1 root root 86 t 26 15:44 bitypes.h
-rw-r--r-- 1 root root 18308 t 26 15:44 cdefs.h
-rw-r--r-- 1 root root 3576 t 26 15:44 debugreg.h
-rw-r--r-- 1 root root 922 t 26 15:44 dir.h
-rw-r--r-- 1 root root 1024 t 26 15:44 elf.h
-rw-r--r-- 1 root root 4411 t 26 15:44 epoll.h
-rw-r--r-- 1 root root 19 t 26 15:44 errno.h

```



Tutorial 1 of RFSoC is done with this signal.



I can run jasper to build [rfsoc4x2\\_tut\\_rfdc\\_real.slx](#) and [rfsoc4x2\\_tut\\_spec.slx](#) examples into .fpg and .dtbo files, and they work. However, if I rename the .slx file or copy it to a new file to make changes, it still builds, but the ADC only returns -1.

When I say ANY changes to the .slx make the ADC return all -1, this has been reduced to even trivial things:

- \* renaming the .slx files with mv
- \* copying an .slx file to a new name with cp
- \* Save As within simulink
- \* copying and pasting the .slx contents into a new file

This is an embarrassing and trivial-sounding issue, but I don't understand enough about simulink or Model Composer to understand how this could be. Is there a filename check or checksum buried in some block in these .slx files or some casperfpga init code? I can't find any.

I've also failed in a similar way with the rfcd when trying to build up "by hand" any project with the RFDC yellow block, including the [RFDC tutorial #2](#).

**This bug is mentioned in the issue of mlib\_devel. I got new lesson today, when you follow any Github, be sure to see the issue someone raised, It will save you and your time a lot.**

**When you change the name of deafult tutorial file, make sure you go inside the rfcd block and change the name of all the 8 blocks inside.**

Hi Jason,

If you look underneath the RFDC yellow block, the gateways should have names which include a prefix which is the model name + RFDC block name. Are these correct in your broken models, or do they reflect a previous model name?

If they are wrong, does creating a new RFDC block and reconfiguring it sort things?

If this turns out to be the problem, please raise an issue on the [mlib\\_devel](#) repo, because this is definitely a bug in the RFDC mask.

Cheers  
Jack

Next step is to image SD card, Image SD card with Casper.

<https://casper.groups.et.byu.net/rfsoc4x2/>

```
navigate to the download location of the compressed tar and unpack it
$ cd </path/to/downloads>
$ tar -xzf zcu216_casper.img.tar.gz

the full uncompressed image `zcu216_casper.img` is now in the current directory
$ ls zcu216_casper.*
zcu216_casper.img zcu216_casper.img.tar.gz

plug in the micro sd card, on OS's like Ubuntu the disk may auto mount,
unmount before proceeding.

Take note of the kernel registered block device
such as `sdb, sdc, sdd, etc.'. This can be done with the `dmesg` utility e.g.,
$ dmesg
[108821.527053] scsi host38: usb-storage 2-2:1.0
[108822.527801] scsi 38:0:0:0: Direct-Access TS-RDF5 SD Transcend TS38 PQ: 0 ANSI: 6
[108822.528460] sd 38:0:0:0: Attached scsi generic sg3 type 0
[108822.829512] sd 38:0:0:0: [sdd] 31116288 512-byte logical blocks: (15.9 GB/14.8 GiB)
[108822.830188] sd 38:0:0:0: [sdd] Write Protect is off
[108822.830197] sd 38:0:0:0: [sdd] Mode Sense: 23 00 00 00
[108822.830867] sd 38:0:0:0: [sdd] Write cache: disabled, read cache: enabled, doesn't support D
[108822.835071] sdd: sdd1 sdd2
[108822.837460] sd 38:0:0:0: [sdd] Attached SCSI removable disk
[109641.322489] sdd: sdd1 sdd2

in this example the sd card block device is `sdd`

flash the sd card with the `dd` utility, wait until this completes. It can take awhile
as we must wait to sync all the I/O, must also have root access
$ sudo dd if=zcu216_casper.img of=/dev/sdd bs=32MB
```

The above link is for ZCU126, follow the same steps for rfsoc 4x2 kit.

After successfully imaging the file, we need to download minicom application.

- open the file from root \$ sudo su

\$ minicom -s

then do configure setting. Make your setting to ttyUSB1

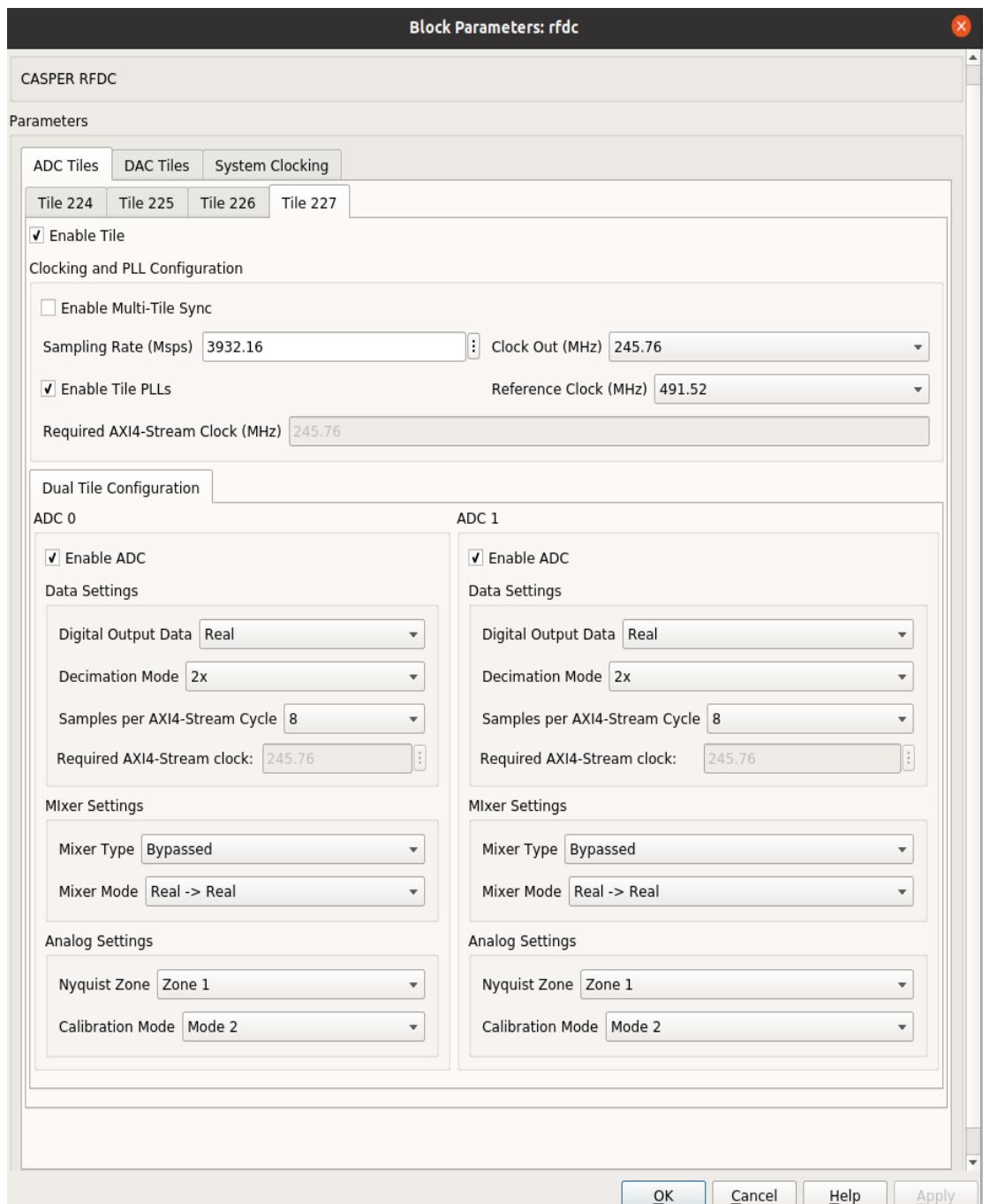
In minicom, change hardware control to No and save it.

Then you can type something in localhost login.

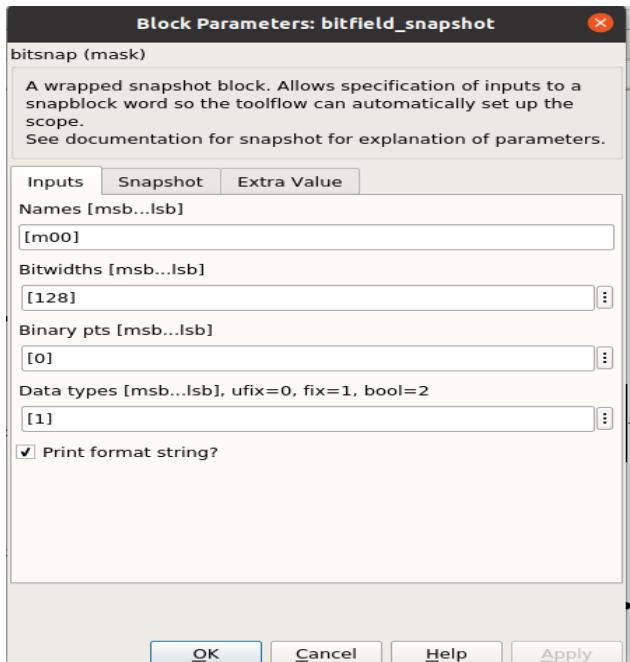
Screenshots are attached in the pages below:

For Tutorial 2 for RFSoC 4x2 kit, for dual tile interface, go to tutorial devil (RFSoC, tut\_rfcd,...) and make a configuration of a block like that. The tutorial 2 is for different block, ZCU111, ZCU216 etc.

Run it, using jasper, and you will generate .dtbo and .fpg file.



Enable tile for all 224, 225, 226, 227 follow it.



```
bishnu@bishnu:~/Desktop/caspertool/mlib-devel$ source startsg startsg.local
Reading local definitions from 'startsg.local'
Found python on start variable
Using MATLAB_PATH=/usr/local/MATLAB/R2021a
Using XILINX_PATH=/tools/Xilinx/Vivado/2021.1
Using COMPOSER_PATH=/tools/Xilinx/Model_Composer/2021.1
Using PLATFORM=lin64
Using MLIB_DEVEL_PATH=/home/bishnu/Desktop/caspertool/mlib-devel
Using JASPER_BACKEND=vitis
Using XML2VHDL_PATH=
Using LD_PRELOAD=
Using DSP_HDL_DEVEL_PATH =
Using python: /home/bishnu/Desktop/m_work/casper_venv/bin/python
(casper_venv) bishnu@bishnu:~/Desktop/caspertool/mlib-devel$ /home/bishnu/Desktop/m_work/casper_venv/bin/python /home/bishnu/Desktop/caspertool/mlib-devel/jasper_library/exec_flow.py -m /home/bishnu/Desktop/m_work/work_file/CT/corner_tur_n_memory.slx --middleware --backend --software --vitis
Starting compile
Adding .gitignore
Starting Toolflow!
Frontend is simulink
```

If you want to compile backend in Python, First go to mlib\_devel folder and run command:  
 source startsg startsg.local

Note: You doesn't need to activate virtual environment of python before. You will face error like:

```
(casper_venv) bishnu@bishnu:~/Desktop/caspertool/mlib-devel$ source startsg startsg.local
```

Reading local definitions from 'startsg.local'

ERROR: Directory /home/bishnu/Desktop/caspertool/casper\_library is not writable by you.

```
(casper_venv) bishnu@bishnu:~/Desktop/caspertool/mlib-devel$ source startsg startsg.local
Reading local definitions from 'startsg.local'
Found python on start variable
Using MATLAB_PATH=/usr/local/MATLAB/R2021a
Using XILINX_PATH=/tools/Xilinx/Vivado/2021.1
Using COMPOSER_PATH=/tools/Xilinx/Model_Composer/2021.1
Using PLATFORM=lin64
Using MLIB_DEVEL_PATH=/home/bishnu/Desktop/caspertool/mlib-devel
Using JASPER_BACKEND=vitis
Using XML2VHDL_PATH=
Using LD_PRELOAD=
Using DSP_HDL_DEVEL_PATH =
Using python: /home/bishnu/Desktop/m_work/casper_venv/bin/python
(casper_venv) bishnu@bishnu:~/Desktop/caspertool/mlib-devel$ /home/bishnu/Desktop/m_work/
casper_venv/bin/python /home/bishnu/Desktop/caspertool/mlib-devel/jasper_library/exec_flo
w.py -m /home/bishnu/Desktop/m_work/tutorials-devel/rfsoc/tut_spec/rfsoc4x2/rfsoc4x2_tut_
spec_cx.slx --middleware --backend --software --vitis
Starting compile
Adding .gitignore
Starting Toolflow!
Frontend is simulink
Setting compile directory: /home/bishnu/Desktop/m_work/tutorials-devel/rfsoc/tut_spec/rfs
oc4x2/rfsoc4x2_tut_spec_cx
/home/bishnu/Desktop/caspertool/mlib-devel/jasper_library/platforms/rfsoc4x2.vaml
```

After you run jasper\_frontend in Matlab, follow this to run in command prompt. You can do all in matlab just by typing jasper, but I believe, this cmd will make matlab free, and you can use matlab for other use as well.

We have to make some configuration for the board.

- Activate root and Change the directory to etc
  - check *etc/dnsmasq.config* file using vim .
  - If it is not installed, try using install dnsmasq using pip command.
- 

- Set the dnsmasq.configuration file and it will help to connect IP and Mac address of the PC
- 

DNS masq file setting

---

bogus-priv  
filterwin2k

```
Interface to listen on (replace eth0 with your interface name)
interface=enp0s31f6
bind-interfaces
```

```

Specify the DHCP range
dhcp-range=192.168.50.2,192.168.50.254,12h
Set a static IP address for a specific device (replace MAC and IP addresses)
dhcp-host=8C:EC:4B:7C:59:C2,192.168.50.198 #My Device
dhcp-host=fc:c2:3d:5b:39:6f,192.168.50.198 #RFSoC 4x2
Enable TFTP (Trivial File Transfer Protocol) for PXE boot (if applicable)
enable-tftp
#tftp-root=/path/to/tftp/root
Set the default gateway address
dhcp-option=3
dhcp-option=option:ntp-server,192.168.50.1

Log DHCP queries
log-dhcp
Log DHCP lease requests
dhcp-leasefile=/var/lib/misc/dnsmasq.leases
dhcp-authoritative

```

The main challenge is that, FPGA board is not showing IP. For this I went to `/var/lib/misc/`  
- vim `dnsmasq.leases`  
There you can find the IP for board.  
- ping this ip and it should show that some data is flowing.

```

bishnu@bishnu:/var/lib/misc$ sudo su
root@bishnu:/var/lib/misc# vim dnsmasq.conf
root@bishnu:/var/lib/misc# vim dnsmasq.conf
root@bishnu:/var/lib/misc# cd..
cd..: command not found
root@bishnu:/var/lib/misc# cd ..
root@bishnu:/var/lib/misc# cd misc/
root@bishnu:/var/lib/misc# vi dnsmasq.leases
root@bishnu:/var/lib/misc# █
```

```
64 bytes from 192.168.50.198: icmp_seq=2 ttl=64 time=0.211 ms
64 bytes from 192.168.50.198: icmp_seq=3 ttl=64 time=0.212 ms
64 bytes from 192.168.50.198: icmp_seq=4 ttl=64 time=0.206 ms
64 bytes from 192.168.50.198: icmp_seq=5 ttl=64 time=0.206 ms
64 bytes from 192.168.50.198: icmp_seq=6 ttl=64 time=0.214 ms
64 bytes from 192.168.50.198: icmp_seq=7 ttl=64 time=0.212 ms
64 bytes from 192.168.50.198: icmp_seq=8 ttl=64 time=0.223 ms
64 bytes from 192.168.50.198: icmp_seq=9 ttl=64 time=0.207 ms
^Z
[3]+ Stopped ping 192.168.50.198
bishnu@bishnu:/var/lib/misc$ ping 192.168.50.198
PING 192.168.50.198 (192.168.50.198) 56(84) bytes of data.
64 bytes from 192.168.50.198: icmp_seq=1 ttl=64 time=0.301 ms
64 bytes from 192.168.50.198: icmp_seq=2 ttl=64 time=0.266 ms
64 bytes from 192.168.50.198: icmp_seq=3 ttl=64 time=0.314 ms
64 bytes from 192.168.50.198: icmp_seq=4 ttl=64 time=0.223 ms
64 bytes from 192.168.50.198: icmp_seq=5 ttl=64 time=0.315 ms
64 bytes from 192.168.50.198: icmp_seq=6 ttl=64 time=0.144 ms
64 bytes from 192.168.50.198: icmp_seq=7 ttl=64 time=0.311 ms
64 bytes from 192.168.50.198: icmp_seq=8 ttl=64 time=0.309 ms
64 bytes from 192.168.50.198: icmp_seq=9 ttl=64 time=0.321 ms
64 bytes from 192.168.50.198: icmp_seq=10 ttl=64 time=0.203 ms
64 bytes from 192.168.50.198: icmp_seq=11 ttl=64 time=0.207 ms
```

\$ ping is to check flow of internet in the devices like FPGA.....

```
(casper_venv) bishnu@bishnu:~/Desktop/m_work$ ipython
Python 3.8.10 (default, Nov 22 2023, 10:22:35)
Type 'copyright', 'credits' or 'license' for more information
IPython 8.12.3 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import casperfpga

In [2]: casperfpga.__version__
Out[2]: '0.4.4.dev1336+py38.276ee44'

In [3]: fpga = casperfpga.CasperFpga('192.168.50.198')

In [4]: fpga.is_connected()
Out[4]: True

In [5]: fpga.upload_to_ram_and_program('/home/bishnu/Desktop/m_work/tutorials_de
....: vel/rfsoc/tut_rfdc/rfsoc4x2/rfsoc4x2_tut_rfdc_cx/outputs/rfsoc4x2_tut_rf
....: dc_cx_2023-12-01_0953.fpg')
....:
....:
....:
....:

^[[AOut[5]: True
```

```
bishnu@bishnu:~$ sudo su
[sudo] password for bishnu:
root@bishnu:/home/bishnu# minicom
minicom: cannot open /dev/modem: No such file or directory
root@bishnu:/home/bishnu# cd /dev/m
mapper/ mcelog mei0 mem mqueue/
root@bishnu:/home/bishnu# cd /dev/
root@bishnu:/dev# ls
autofs i2c-7 sda tty25 tty60 ttyS9
block initctl sda1 tty26 tty61 ttyUSB0
bsg input sda2 tty27 tty62 ttyUSB1
btrfs-control kmsg sda3 tty28 tty63 udmabuf
bus kvm sda5 tty29 tty7 uhid
cdrom log sdb tty3 tty8 uinput
cdrw loop0 sdb1 tty30 tty9 urandom
char loop1 serial tty31 ttynctrlk userio
```

```
Welcome to minicom 2.7.1

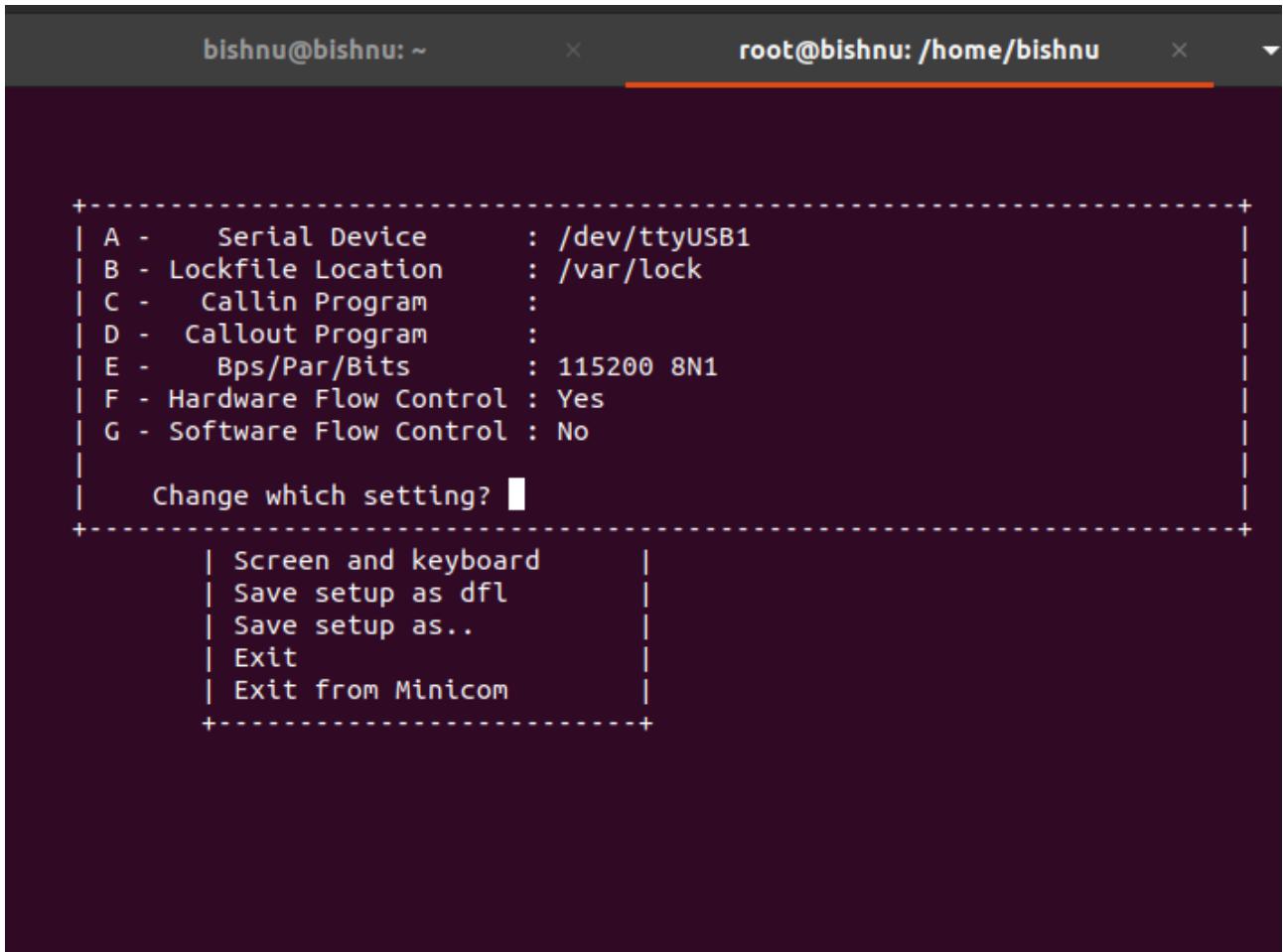
OPTIONS: I18n
Compiled on Dec 23 2019, 02:06:26.
Port /dev/ttyUSB1, 19:06:13

Press CTRL-A Z for help on special keys

Xilinx Zynq MP First Stage Boot Loader
Release 2020.2 Aug 15 2022 - 19:10:45
NOTICE: ATF running on XCZU48DR/silicon v4/RTL5.1 at 0xffffea000
NOTICE: BL31: v2.2(release):xlnx_rebase_v2.2_2020.3
NOTICE: BL31: Built : 19:06:21, Aug 15 2022

U-Boot 2020.01 (Aug 15 2022 - 19:10:42 +0000)
```

For RFSoC4x2 kit, we don't need to fix Mac and other things there. Just go to root mode and run minicom -s and continue it to boot and input login name 'casper' and password 'casper' and ifconfig and you will get MAC id of your RFSoC board in it. I tried to set by myself checking MAC address from 'dnsmasq.leases' and set it but my friend Cliff suggested me that RFSoC board is already MAC setted. We just follow above instructions and get the MAC. Awesome Cliff.



The screenshot shows a terminal window with two tabs. The left tab is titled "bishnu@bishnu: ~" and the right tab is titled "root@bishnu: /home/bishnu". The main area displays the Minicom configuration menu. The configuration settings listed are:

- A - Serial Device : /dev/ttyUSB1
- B - Lockfile Location : /var/lock
- C - Callin Program :
- D - Callout Program :
- E - Bps/Par/Bits : 115200 8N1
- F - Hardware Flow Control : Yes
- G - Software Flow Control : No

Below these settings, there is a prompt: "Change which setting? █". At the bottom of the menu, there is a list of options:

- Screen and keyboard
- Save setup as dfl
- Save setup as..
- Exit
- Exit from Minicom

system status dnsmasq.service

Set Hardware flow Control to 'NO' in Minicom.

Turn on the RFSOC board and then it will ask to press any button to boot, then press any key.

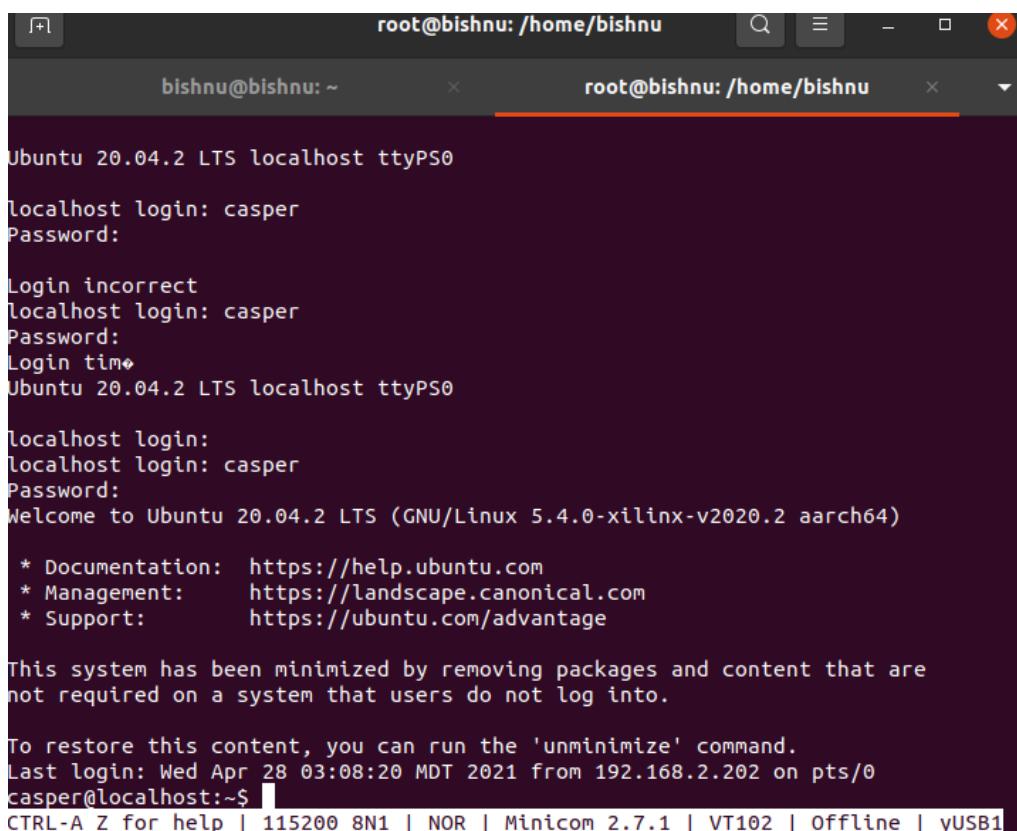
Start setting as in tutorial.

Note: The name of the file should not contain any **CAPITAL LETTER** like RFSOC4x2\_spec.slx, it won't compile properly and will give you an error.

```
U-Boot 2020.01 (Aug 15 2022 - 19:10:42 +0000)

Model: PYNQ RFSoC 4x2
Board: Xilinx ZynqMP
DRAM: 4 GiB
PMUFW: v1.1
EL Level: EL2
Chip ID: zu48dr
NAND: 0 MiB
MMC: mmc@ff160000: 0
In: serial@ff010000
Out: serial@ff010000
Err: serial@ff010000
Bootmode: SD_MODE
Reset reason: EXTERNAL
Net:
ZYNQ GEM: ff0c0000, mdio bus ff0c0000, phyaddr 15, interface rgmii-id
```

[https://casper-toolflow.readthedocs.io/projects/tutorials/en/latest/tutorials/rfsoc/tut\\_getting\\_started.html](https://casper-toolflow.readthedocs.io/projects/tutorials/en/latest/tutorials/rfsoc/tut_getting_started.html)



```
root@bishnu: /home/bishnu
root@bishnu: ~
root@bishnu: /home/bishnu

Ubuntu 20.04.2 LTS localhost ttyPS0

localhost login: casper
Password:

Login incorrect
localhost login: casper
Password:
Login time
Ubuntu 20.04.2 LTS localhost ttyPS0

localhost login:
localhost login: casper
Password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-xilinx-v2020.2 aarch64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Wed Apr 28 03:08:20 MDT 2021 from 192.168.2.202 on pts/0
casper@localhost:~$
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | yUSB1
```

**This setting task is very much complicated.** From getting Started with RFSoC, booting is done to give a MAC address to the board. Booting using minocom -s and follow i2c bus command and follow the steps and fix the MAC address and do restart. Other's command as shown in tutorial may not work in RFSoC4x2 baord.

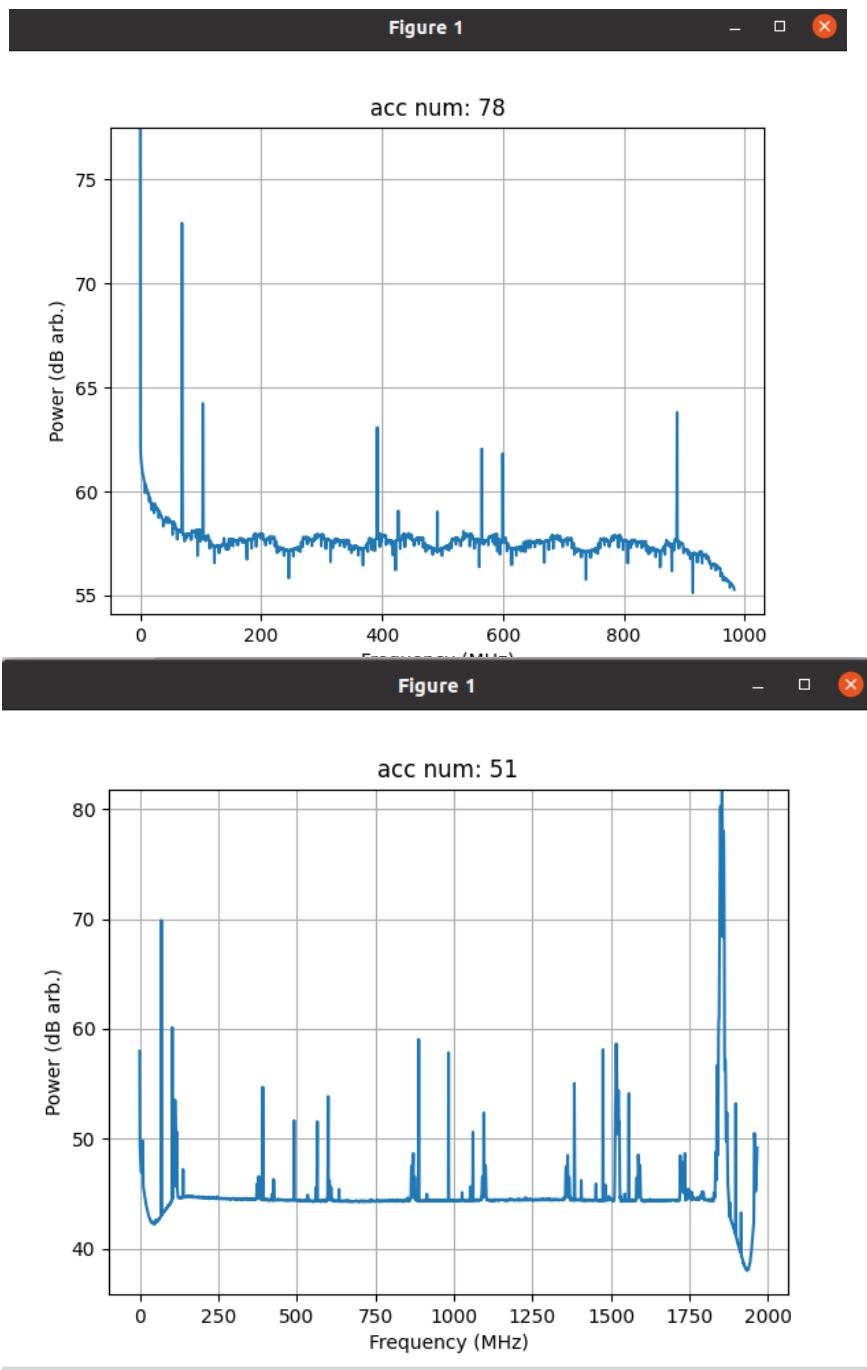
It may make you frustrating to use it:

```
dhcp-host=8C:EC:4B:7C:59:C2,192.168.50.1 #My Device PC (It is the network from my local system router, when there is no internet connection from external sources).
```

```
dhcp-host=fc:c2:3d:5b:39:6f,192.168.50.198 #RFSoC 4x2
```

```
#ping 172.17.31.46 (When there is University ethernet), MAC address: 8C:EC:4B:7C:59:C2 of my PC.
```

```
##fpga: xczu48dr-ffvg1517-2-e
pfb
```



The clock speed of RFSoC 4x2 block

```
RFSoC 4x2
User IP Clock Rate: 245.76, RFPLL PL Clock Rate: 122.88
```

For 100 GbE. We need as mentioned in the tutorials and this branch of github must be installed as Sir suggested.

(casper\_venv)

```
homin@homin-5820:~/rfsoc/python3/casper_venv/mlib-devel/jasper_library/
hdl_sources/onehundred_gbe
```

Please remove the previous version of kutleng\_skarab2\_bsp\_firmware  
and download this one with branch: mb\_rtrs  
[https://github.com/casper-astro/kutleng\\_skarab2\\_bsp\\_firmware](https://github.com/casper-astro/kutleng_skarab2_bsp_firmware)  
branch : mb\_rtrs

If you do this, you can run model .slx file from tutorial devel and then can make .fpg and .dtbo file.  
Those .slx files are the files we use later for processing.

What will happen if you restart your FPGA and Computer? Does it affects IP setting of the FPGA?

- If you restart both system, then make sure the internal IP you have given to PC and FPGA are PING.

- In my case ping 192.168.50.1 (Master for PC from router). This is working fine  
- ping 192.168.50.198 (IP for FPGA from router). This doesn't work.

For this ping to work go to network setting of your Ubuntu and click the name of the network you have assigned to the FPGA. Then ping it to check. I only can connect 2 connection at a time.

There is a trick:

- I changed the name of the network to FPGA and type ip as 192.168.50.198 and dns as 255.255.255.0 and gateway to 192.168.50.198

- I changed the MAC address to the MAC address of the FPGA: fc:c2:3d:5b:39:6f

The name of the network disappear from the Network setting but this make FPGA connecting to the IP .

Each time I want to assign IP to eeprom, it is not happening. Check sudo su:  
/var/lib/misc/dnsmasq.leases

This will assign IP with \* on the top means the IP FPGA is connected.

See the difference between network from router to FPGA and PC.

```
PING 192.168.50.198 (192.168.50.198) 56(84) bytes of data.
64 bytes from 192.168.50.198: icmp_seq=1 ttl=64 time=0.354 ms
64 bytes from 192.168.50.198: icmp_seq=2 ttl=64 time=0.212 ms
64 bytes from 192.168.50.198: icmp_seq=3 ttl=64 time=0.219 ms
64 bytes from 192.168.50.198: icmp_seq=4 ttl=64 time=0.197 ms
```

```
PING 192.168.50.1 (192.168.50.1) 56(84) bytes of data.
```

```
64 bytes from 192.168.50.1: icmp_seq=1 ttl=64 time=0.081 ms
64 bytes from 192.168.50.1: icmp_seq=2 ttl=64 time=0.067 ms
64 bytes from 192.168.50.1: icmp_seq=3 ttl=64 time=0.067 ms
```

64 bytes from 192.168.50.1: icmp\_seq=4 ttl=64 time=0.075 ms

The Network to FPGA is given by PCI Ethernet. Keep it ON.

For my safe side, I stopped connecting internet to my PC. Let run PC without internet, only for FPGA.

This is connection from USB

```
3: enx00e04c680039: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
state UP group default qlen 1000
link/ether 00:e0:4c:68:00:39 brd ff:ff:ff:ff:ff:ff
```

Different cases where I encountered errors:

I run zcu216\_100g\_init.py (I got from Sir Homin) and I the  
FPGA clock = 245.0581725 MHz to 249 MHz range

but it should be

FPGA clock = 200.0581725 MHz

This Frequency is raising error while running zcu216\_accum\_spec2k\_dx4.py which gives images on nearly 200 Mhz frequency, but it does produce error on Frequency 245 MHz.

AWW!! So stressful, I don't know how there is two different frequency arise. Might be clock issue. Let's dig it.

- I checked different FPGA file like .fpg file from tutorials devles and the error is same, I am getting plots but I am not able to see any signal in the plot. It's empty plot. I think this might be python issue. See you python now:

-

--

---

1/12/2024

Dr. Homin asked me to run:

ADC to check whether there is something correct or not.

This is how Dr. Homin helped me to login this FPGA. Although, the idea is simple SSH, but since you don't know how, means it's always a mystery. You should know what to do and How to do. Suppose, if someone ask me can you login to FPGA board, then I can do research how to do that, and I will do that. Instantly, at a point, none of us have idea how to do.

```
(casper_venv) bishnu@bishnu:~/Desktop/m_work/work_file$ ssh -l casper 192.168
.50.198
The authenticity of host '192.168.50.198 (192.168.50.198)' can't be established.
ECDSA key fingerprint is SHA256:EfzCqq6+T+ziX1ZLFNENIqyNJv5HAoXszjBFaoaDagI.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
Host key verification failed.
(casper_venv) bishnu@bishnu:~/Desktop/m_work/work_file$ ssh -l casper 192.168
.50.198
The authenticity of host '192.168.50.198 (192.168.50.198)' can't be established.
ECDSA key fingerprint is SHA256:EfzCqq6+T+ziX1ZLFNENIqyNJv5HAoXszjBFaoaDagI.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.50.198' (ECDSA) to the list of known hosts.
casper@192.168.50.198's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-xilinx-v2020.2 aarch64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
```

```
casper@localhost:~$
casper@localhost:~$
casper@localhost:~$
casper@localhost:~$ ls
bin git share
casper@localhost:~$ cd bin
casper@localhost:~/bin$ ls
bind_spidev.sh init_rfclk_gpio reset_rfpll
display_oled prg_rfpll tcpborphserver3
casper@localhost:~/bin$ cd /lib/firmware/
casper@localhost:/lib/firmware$ ls
rfsoc4x2_LMX_REF_245M76_OUT_491M52.txt tcpborphserver.bin
rfsoc4x2_PL_122M88_REF_245M76.txt tcpborphserver.dtbo
casper@localhost:/lib/firmware$
```

```
This is the code to give which clock file to use.
if (rfdc_rfsoc4x2.progpll('lmk', c[0]) == True):
 print ("Clock PLL success \n")
else :
 print ("Clock PLL failed \n")
 exit_fail()
```

In the FPGA, what I noticed c[0] will take 1<sup>st</sup> .txt clock file and so on. The way me and cliff is using “sudo ./bin/prg\_rfpll -lmk /lib/firmware/bishnu\_LMK04828B\_rfsoc4x2default\_try.txt” is to check whether PLL1 and PLL2 led glows or not.

These two leds give us a clue about the status of the clock file

```
#####
#####
```

When I login to FPGA using “ssh -l casper 192.168.50.198”

the password is casper itself, then we can login into the board and go to cd bin, then cd libfirmware/ then check the files. You will see two clock files, How the above script works is when you give c[0], it will take 1<sup>st</sup> file, when you give c[1], it take 2<sup>nd</sup> file. I have to add the another file of clock, that Dr. Homin gave me. He asked me to avoid confusion about the first and second file, just add

the file he gave me in the txt mode and rename other file extension as .tx only, but don't delete those files, as we may need them.

See, there, I changed the file to .tx extension and I have to now add file that Dr. Homin gave me.

First login to the FPGA board, and then use this command to change the ownership.  
This command changes the ownership of the `/lib/firmware/` directory to the `casper` user and group

```
casper@localhost:/lib/firmware$ ls
rfsoc4x2_LMX_REF_245M76_OUT_491M52.tx tcpborphserver.bin
rfsoc4x2_PL_122M88_REF_245M76.txt tcpborphserver.dtbo
casper@localhost:/lib/firmware$ sudo chown casper:casper /lib/firmware/
[sudo] password for casper:
casper@localhost:/lib/firmware$
```

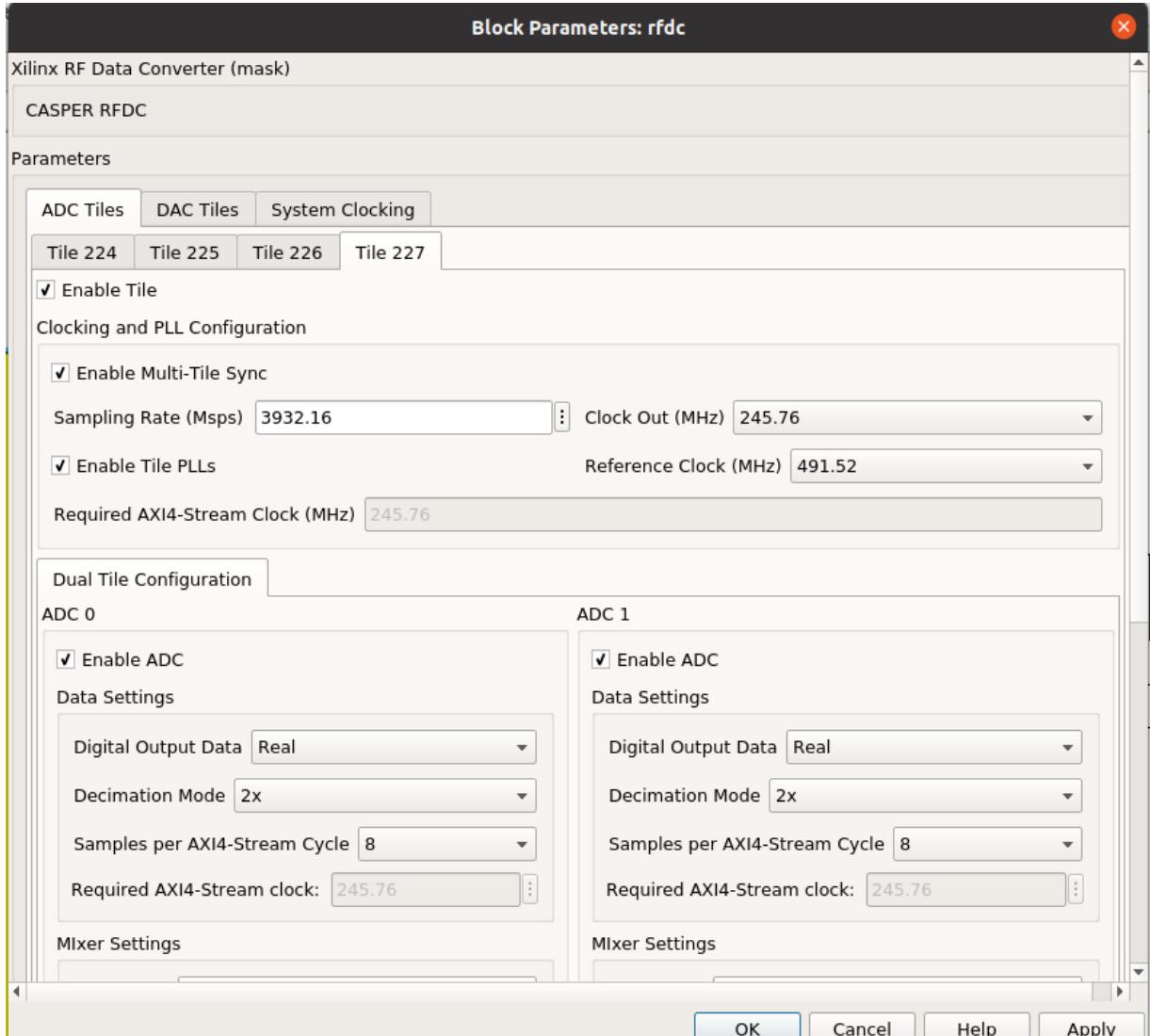
```
casper@localhost:/lib/firmware$ ls
rfsoc4x2_LMX_REF_245M76_OUT_491M52.tx tcpborphserver.bin
rfsoc4x2_PL_122M88_REF_245M76.txt tcpborphserver.dtbo
casper@localhost:/lib/firmware$ sudo mv rfsoc4x2_PL_122M88_REF_245M76.txt rfs
oc4x2_PL_122M88_REF_245M76.tx
casper@localhost:/lib/firmware$ ls
rfsoc4x2_LMX_REF_245M76_OUT_491M52.tx tcpborphserver.bin
rfsoc4x2_PL_122M88_REF_245M76.txt tcpborphserver.dtbo
casper@localhost:/lib/firmware$
```

Now the clock is showing frequency of 400 MHz. Bad Haan!!

```
#####
Login to Casper FPGA 4x2 kit
ssh -l casper 192.168.50.198
#####
/bin$ cd /lib/firmware
You will get two txt file as a clock file
In this folder of FPGA, it will take clock file by chronological order. It is always wise to name the file you use in .txt and other as .tx only. It will avoid much confusion.
scp /home/bishnu/Desktop/LMK04828B_B3000M_200M_MTS_SYSREF_5M_Clk0_20230316.txt
casper@192.168.50.198:/lib/firmware
casper@192.168.50.198's password:
LMK04828B_B3000M_200M_MTS_SYSREF_5M_Clk0_2023 100% 2034 1.8MB/s 00:00
#####
To give access to FPGA folder to paste the file .txt file
sudo chown casper:casper /lib/firmware/
[sudo] password for casper: casper
casper@localhost:/lib/firmware$ sudo chown casper:casper /lib/firmware/
#####
```

Let's Insight into the clock issues.

PPS stands for Pulse Per Second, and it is often used in the context of precision timing and synchronization. In FPGA (Field-Programmable Gate Array) applications, PPS signals are commonly used to provide a precise time reference for systems that require accurate timekeeping.



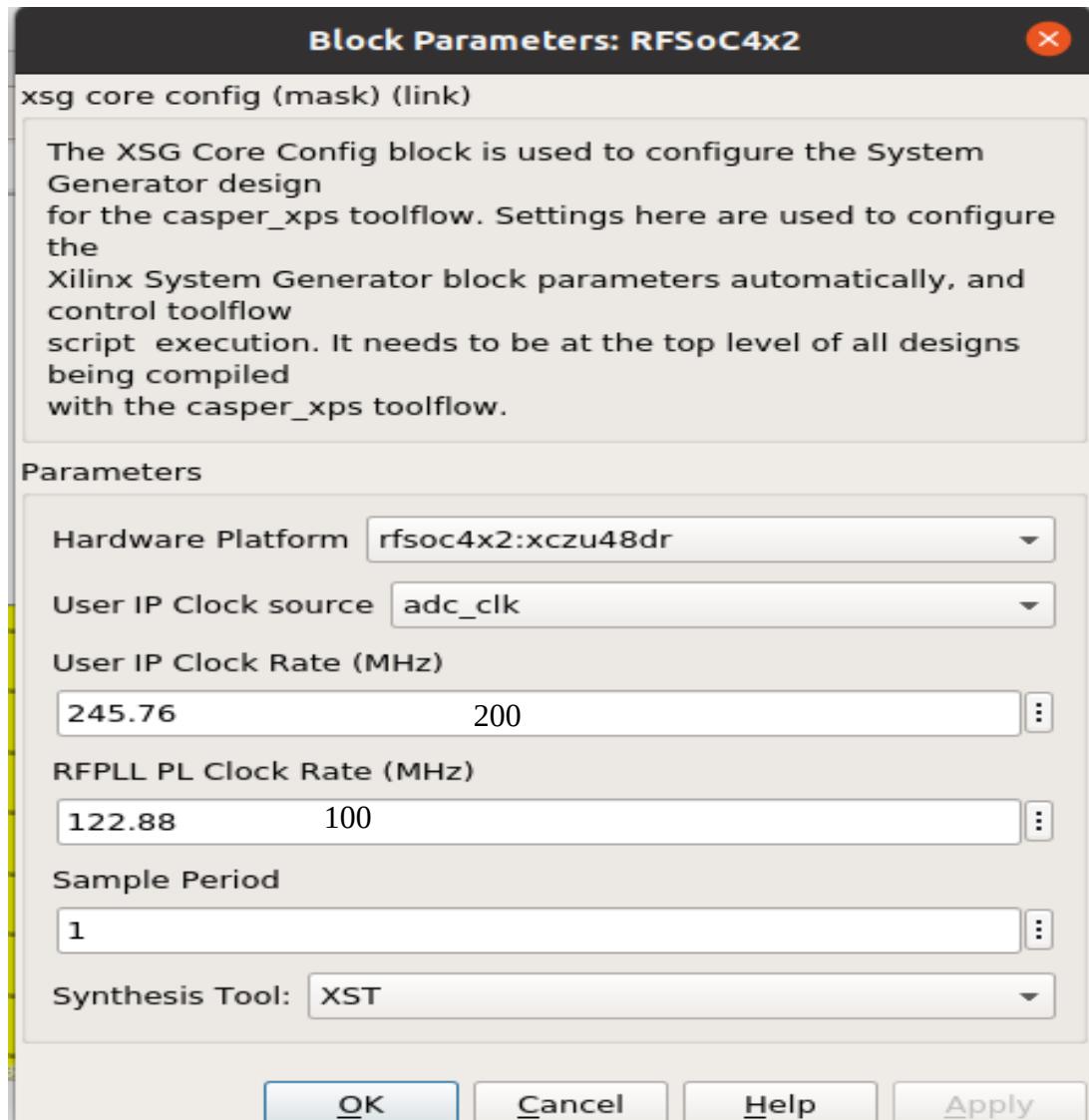
pip install casperfpga  
In the above figure, we can find the information of the clock.

$$3932.16/245.76 = 16 \text{ bits}$$

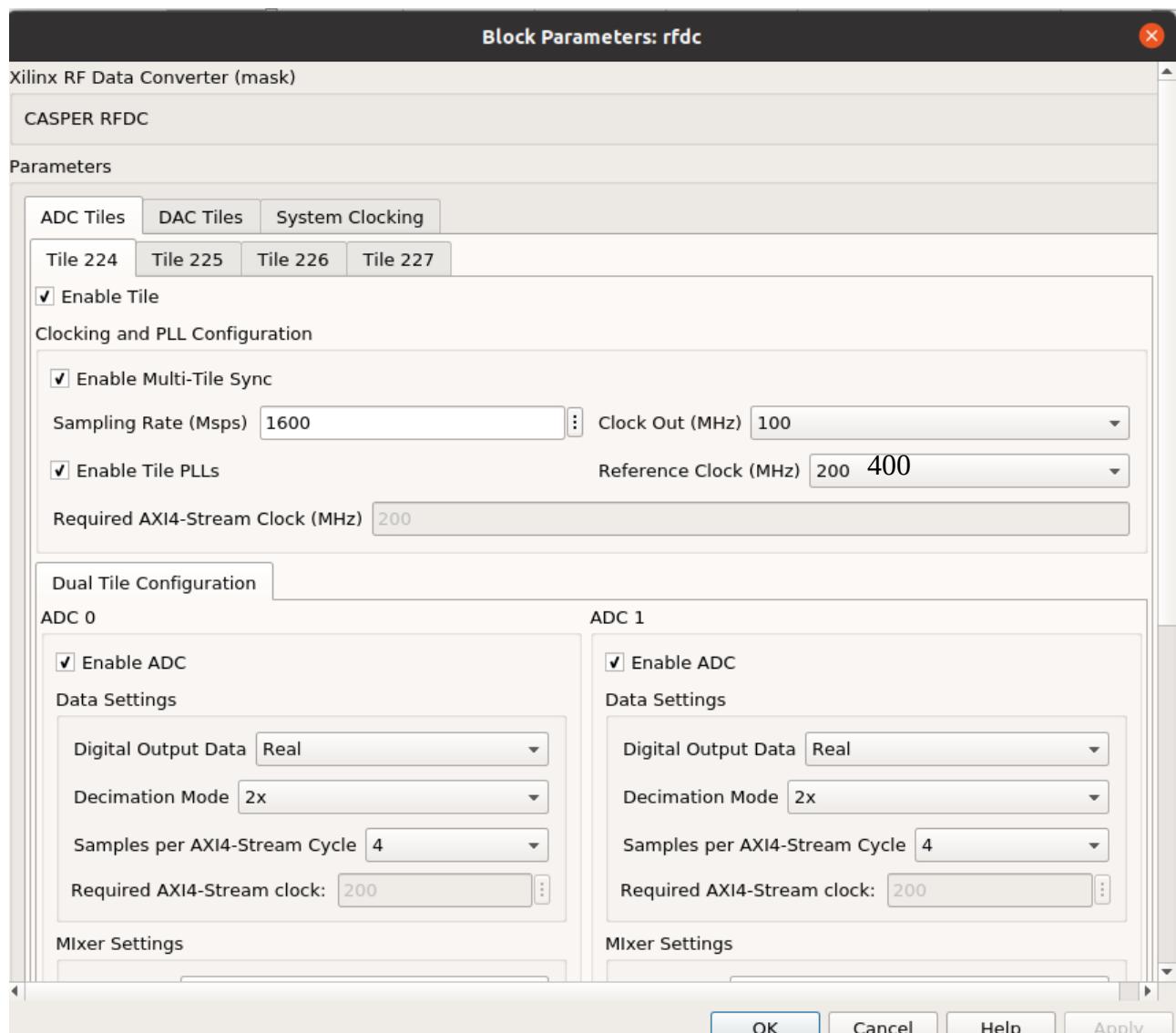
$$3932.16/491.52 = 8 \text{ bits}$$

Required AXI-4 stream clock is 245.76

Reference clock give 8 bits, while clock out is 16 bits as per this RFDC clock setting.  
When you select sampling rate, 3932.16, it will automatically select the clock out 245.76 MHz.



The **User IP Clock Rate** is the desired frequency for the IP of the design. For the RFSoC platform the **adc\_clk** user IP clock source is derived from the **pl\_clk** coming from the first stage PLL in the clocking hierarchy for the RFDC (LMK04828B and LMX2594). In most cases this is an LMK creating the **pl\_clk** in addition to the clock that drives the RFDC tiles. This frequency coming from the LMK as **pl\_clk** (PL SYS REFCLK is to be set 100 Mhz in clock schematic diagram) is what is to be entered into the **RFPLL PL Clock Rate** field. In other words, this is the clock rate the design is expecting to produce the clock frequency for the user IP clock.



Here, the Sampling rate is 1600, so it's clock out is automatically 100 and the reference clock should be 400.

At boot the RFSoC 4x2 the LMK will be programmed to provide a 100 MHz reference to the PL and a 200 MHz reference for the ADC and DAC LMX. The LMX PLLs are programmed to provide a reference of 400 reference to the RFDC ADC and DAC tiles.

Tile 224 corresponding to tile 0, and so on..

If enable tile PLL in not checked, Reference clock will be same as sampling rate.

I HAVE ATTACHED NOISE CABLE AT PORT 1 OF ADC IN FPGA

RFSOC4x2 is a dual tile ADC platform. We need to enable all ADC tiles. Tile 224 (Tile 0), Tile225 (Tile 1), Tlile226 (Tile 2), Tile 227 (Tile 3). Uncheck Enable Multi-Tile Sync. In our design we select Sampling Rate 2000 Msps clock output 100 and reference clock as a 400 (Provide by LMX clock) See the clock configuration of RFSOC4x2 clock schematic diagram. Check Enable Tile PLLs. The Required AXI4-Stream Clock (MHZ) is set automatically when we select Decimation mode and Samples-per AXI4-Stream Cycle. We select

- Decimation mode (2). It drops down the display of the bits.
- Samples-per AXI4-Stream Cycle ( set it to 4). It gives outputs from RFDC block in the multiple of 16.  $4 \times 16 = 64$ . If you select 8, then  $8 \times 16 = 128$ . and so on. This field here displays the effective User IP clock that would be required for the configuration of the decimator and number of samples per clock. These fields are to match for all ADCs within a tile.

**Required AXI4-Stream Clock (MHZ) = Sampling rate / Decimation mode x Samples-per AXI4-Stream Cycle**

- It gives Required AXI4-Stream Clock (MHZ) to 200 Mhz.
- It should match our Platform RFSOC4x2 configuration.
- User IP Clock Rate 200 and RFPLL PL clock rate 100
- Enable both ADCs or can be manipulated
- Digital output real.
- Mixet type Bypassed
- Mixer Mode Real → Real
- Nyquist Zone Zone 1
- Calibration mode mode 2.

**Samples per clock cycle — Number of samples per clock cycle**  
2 (default) | 1 | 3 | 4 | 5 | 6 | 7 | 8 | 10 | 11 | 12 | 14 | 16

Specify the number of samples per clock cycle. Available options for the number of samples per clock cycle vary with the selected hardware board and digital interface type.

The block calculates the stream data width as: 16 x **Samples per clock cycle**.

The block calculates the stream clock frequency as: **Sample rate (MSPS) / Decimation mode (xN) x Samples per clock cycle**.

## #####Snapshot block#####

Snapshot block takes 2 input a write enable and trigger.

For dual-tile platforms in **I/Q** digital output modes, the inphase and quadrature data are produced from different ports. In this mode the first digit of the signal name corresponds to the tile index just as in the quad-tile. But the second digit is **0** for inphase and **1** for quadrature data. In this example then, with **4** sample per clock this is **4** complex samples with the two complex components coming from different ports, **m00\_axis\_tdata** for inphase data ordered **{I3, I2, I1, I0}** and **m01\_axis\_tdata** with quadrature data ordered **{Q3, Q2, Q1, Q0}**. When configured in **Real** digital output mode the second digit is **0** for the first ADC and **2** for the second. With the snapshot block configured to capture  **$2^{14}$  128-bit** words this is a total of  **$2^{16}$**  complex samples for the one port.

M00 = 0 is the ADC tile and 0 is the port of ADC

m01 = 0 is the ADC tile and 1 is the port of ADC

ADC 0 = ADC 224, ADC 1 = ADC 225, ADC 2 = ADC 226, ADC 3 = ADC 227

In complex number :

$x = a + ib$ , Where a is Inphase and b is quadrature.

```
Cx = m00, m01, m02, m03, m20, m21, m22, m23
real = m00, m01, m20, m22
```

#### #####Clock Issue:

```
cd bin/lib/firmware
```

```
LMK04828B_B3000M_200M_MTS_SYSREF_5M_Clk0_20230316.txt tcpborphserver.bin
rfsoc4x2_LMX_REF_245M76_OUT_491M52.txt tcpborphserver.dtbo
rfsoc4x2_PL_122M88_REF_245M76.txt
```

```
FPGA clock = 245.9813525 MHz
```

```
clock counter = 820759135
```

```
RFDC = <casperfpga.rfdc.RFDC object at 0x7f52b92e33d0>
```

```
ADC init. OK
```

```
ADC0: Enabled 1, State: 15 PLL: 1
```

```
ADC1: Enabled 1, State: 12 PLL: 1
```

```
ADC2: Enabled 1, State: 15 PLL: 1
```

```
ADC3: Enabled 1, State: 12 PLL: 1
```

```
DAC0: Enabled 0
```

```
DAC1: Enabled 0
```

```
DAC2: Enabled 0
```

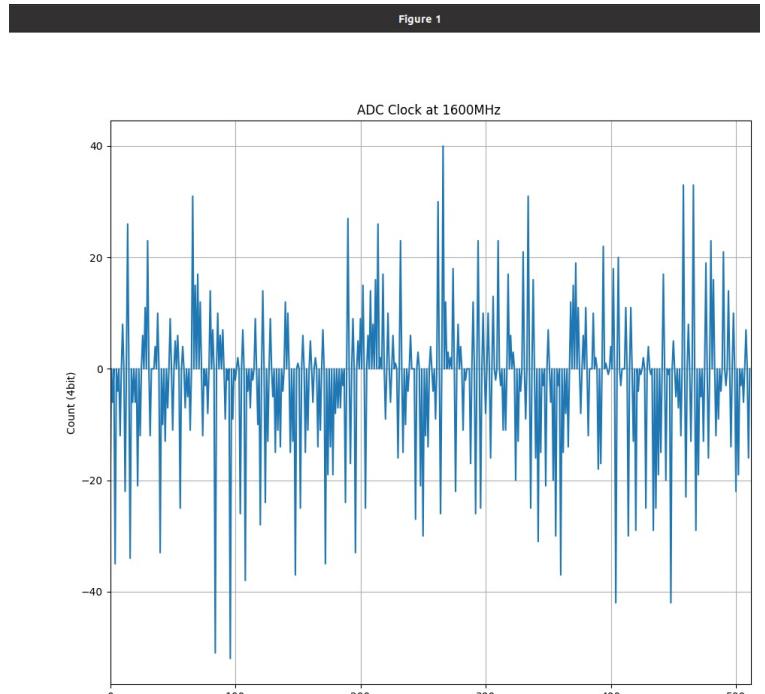
```
DAC3: Enabled 0
```

```
ADC status = True
```

```
Clock PLL success
```

```
Specify the ADC channel ...
```

```
ADC channel = 0
```



```
casper@localhost:/lib/firmware$ ls
```

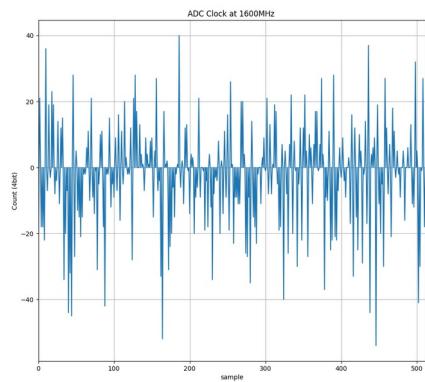
```
LMK04828B_B3000M_200M_MTS_SYSREF_5M_Clk0_20230316.txt tcpborphserver.bin
rfsoc4x2_LMX_REF_245M76_OUT_491M52.txt tcpborphserver.dtbo
rfsoc4x2_PL_122M88_REF_245M76.tx
```

FPGA clock = 245.9951235 MHz

clock counter = 820170106

RFDC = <casperfpga.rfdc.RFDC object at 0x7f1a881b8820>  
ADC init. OK

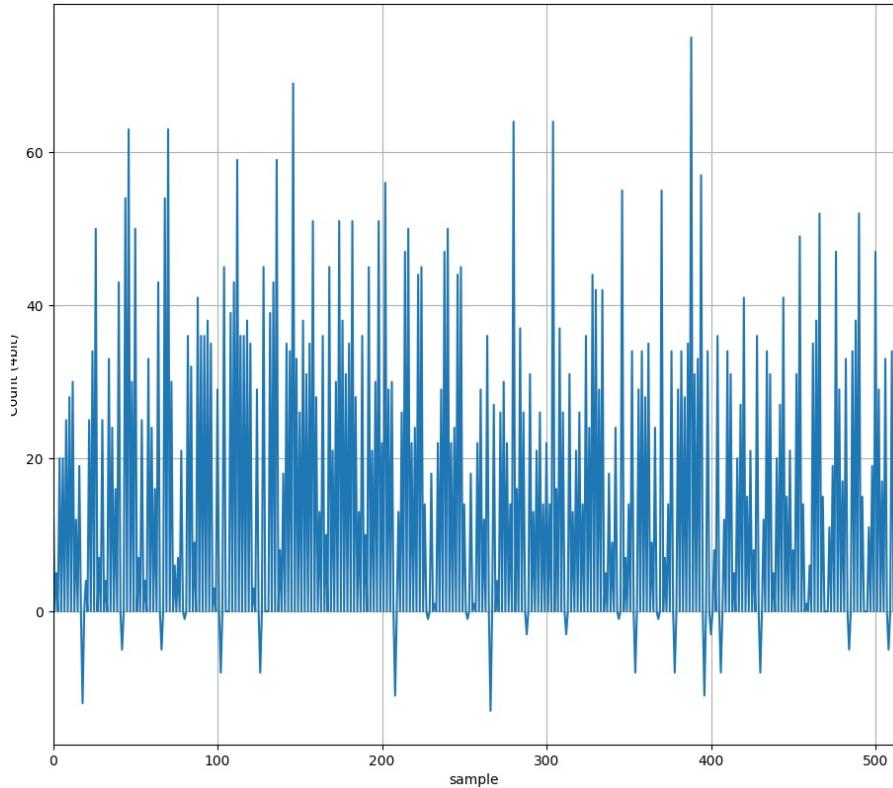
ADC0: Enabled 1, State: 15 PLL: 1  
ADC1: Enabled 1, State: 12 PLL: 1  
ADC2: Enabled 1, State: 15 PLL: 1  
ADC3: Enabled 1, State: 12 PLL: 1  
DAC0: Enabled 0  
DAC1: Enabled 0  
DAC2: Enabled 0  
DAC3: Enabled 0  
ADC status = True



The clock file Dr. Homin gave me. Activate only that.

```
casper@localhost:/lib/firmware$ ls
LMK04828B_B3000M_200M_MTS_SYSREF_5M_Clk0_20230316.txt tcpborphserver.bin
rfsoc4x2_LMX_REF_245M76_OUT_491M52.tx tcpborphserver.dtbo
```

## rfsoc4x2\_PL\_122M88\_REF\_245M76.tx



FPGA clock = 246.2244035 MHz

clock counter = 821606221

RFDC = <casperfpga.rfdc.RFDC object at 0x7f7aaf1e5fa0>  
ADC init. OK

ADC0: Enabled 1, State: 15 PLL: 1

ADC1: Enabled 1, State: 12 PLL: 1

ADC2: Enabled 1, State: 15 PLL: 1

ADC3: Enabled 1, State: 12 PLL: 1

DAC0: Enabled 0

DAC1: Enabled 0

DAC2: Enabled 0

DAC3: Enabled 0

ADC status = True

Clock PLL success

Let's turn off the FPGA, to check whether the poweroff will have any effect on the FPGA settings that I did in the clock.

FPGA clock = 625.7561695 MHz

clock counter = -1931959802

RFDC = <casperfpga.rfdc.RFDC object at 0x7fea596a3310>

ADC init. OK

ADC0: Enabled 1, State: 6 PLL: 0

ADC1: Enabled 1, State: 6 PLL: 0

ADC2: Enabled 1, State: 6 PLL: 0

ADC3: Enabled 1, State: 6 PLL: 0

DAC0: Enabled 0

DAC1: Enabled 0

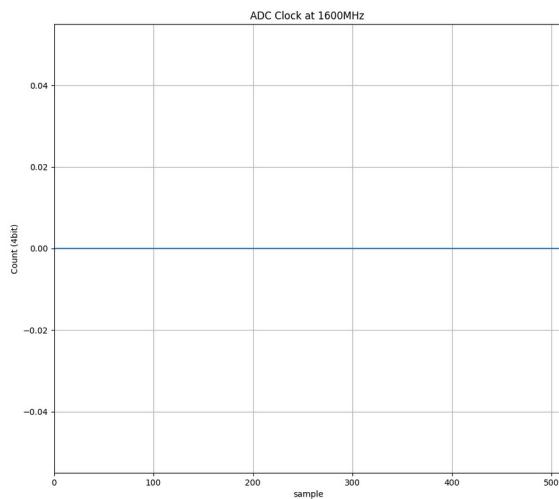
DAC2: Enabled 0

DAC3: Enabled 0

ADC status = True

Clock PLL success

FPGA clock = 400.773024 MHz



See, it's error. The signal is different.

```
casper@localhost:/lib/firmware$ ls
LMK04828B_B3000M_200M_MTS_SYSREF_5M_Clk0_20230316.txt tcpborphserver.bin
rfsoc4x2_LMX_REF_245M76_OUT_491M52.tx tcpborphserver.dtbo
rfsoc4x2_PL_122M88_REF_245M76.tx
```

#####

Let's change the clockfile one by one and turn off FPGA and again on once file is changed.

I checked changing all the clock files from .txt to .tx to make them inactive and check each of them.

The conclusion is that we need both clock file and in .txt version as default in the CASPER Image of the tutorial.

```
casper@localhost:/lib/firmware$ ls
LMK04828B_B3000M_200M_MTS_SYSREF_5M_Clk0_20230316.tx tcpborphserver.bin
rfsoc4x2_LMX_REF_245M76_OUT_491M52.txt tcpborphserver.dtbo
rfsoc4x2_PL_122M88_REF_245M76.txt
```

This is working, I don't know, at least it is giving some signals.

---

-----

Why I am getting 0 signal in the plot of ADC, when I use 200 MHz frequency and sampling rate 1600 Mhz.

- I believe because of the clock files. I am definitely doing good, in circuit design, but somewhere there is error in FPGA settings of the clock files.

## # USER GUIDE TICS PRO

Download the software TICS-PRO from Texas Instruments.

(<https://www.ti.com/tool/TICSPRO-SW>)

- After you download and install it, Now you have to run software for clocks.

- Go to : Select Device >> Clock Generator/Jitter Cleaner (Dual Loop) >> LMK0482x>> LMK04828

- Click “Select Device” → “Clock Generator/ Jitter Cleaner (Dual Loop)” → “LMK0482x” → “LMK04828B”

#####

Play with this and make a clock files.

#####-----#####

Phased locked loop:

It is a non linear feedback system that tracks the phase of input signal and minimize phase errors of the oscillator.

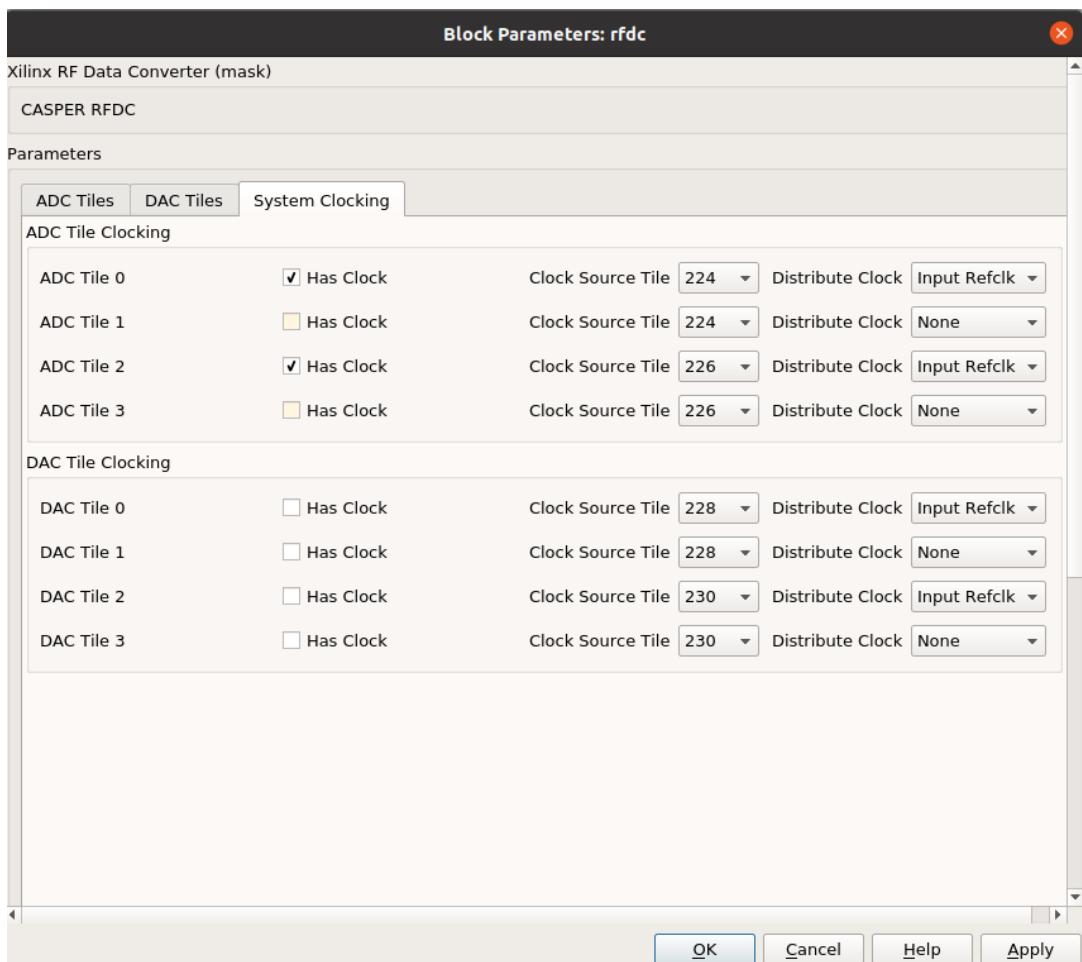
PLL:

- In coherent demodulation PLL is used to removing phase error.

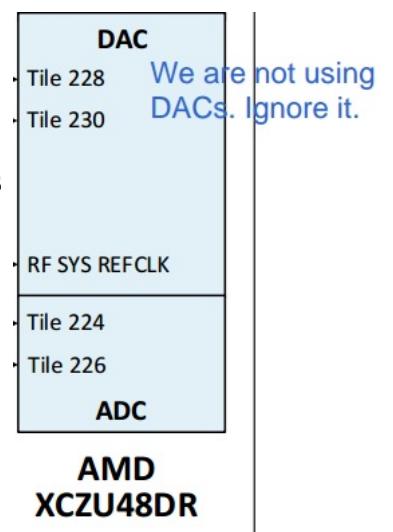
$$c(t)=A_c \sin(\omega_c t + \phi)$$

In this wave equation, %phi is the phase error.

During demodulation, phase error leads distortion in the output side and we need PLL for that.



In this SS from RFSOC4x2 we can see that there are DAC tile 228 and 230 and for ADC there are tile 224 and 226. That's why the SS of above clock have 224, 224, 226, 226 for ADCs and 228, 228, 230, 230 for DACs



Device Information <sup>(1)</sup>		
PART NUMBER	VCO0 FREQUENCY	VCO1 FREQUENCY
LMK04821	1930 to 2075 MHz	2920 to 3080 MHz VCO1 Div = +2 to +8 (+2 = 1460 to 1540 MHz)
LMK04826	1840 to 1970 MHz	2440 to 2505 MHz
LMK04828	2370 to 2630 MHz	2920 to 3080 MHz

(1) For all available packages, see the orderable addendum at the end of the datasheet.

LUTs = Look up Tables.  
phase locked loops (PLL), I/O buffers, block memory (BRAM), Digital Signal Processing (DSP)

VCO = Voltaged Controlled Oscillator

**N Divider:** The N Divider is a programmable counter that divides the input clock frequency by a specified value (N). It is often used in phase-locked loop (PLL) circuits to generate a desired output frequency by dividing the input clock. This allows for flexibility in adjusting the clock frequency according to the application requirements.

**R Divider:** The R Divider is another programmable counter, but it is usually associated with clock multiplication or phase shifting in PLLs. It divides the reference clock by a specified value (R), and the resulting signal is used to compare with a divided version of the feedback clock to adjust the PLL to lock onto the desired frequency.

The Si5395A 12 -output, ultra-high-performance jitter attenuator combines fourth-generation DSPLL and MultiSynth technologies to enable any-frequency clock generation and jitter attenuation for applications like 56G PAM4 SerDes requiring the highest level of jitter performance. The device has an ultra-low jitter of 69 fs and a frequency output range up to 1028 MHz and delivers a 0.09 ps rms phase jitter performance with a 0 ppm error. The loop filter is fully integrated on-chip, eliminating the risk of noise coupling associated with discrete solutions. Further, the jitter attenuation bandwidth is digitally programmable, providing jitter performance optimization at the application level. The Si5395A can be quickly and easily configured using ClockBuilder Pro software.

FPGA use 48 MHz crystal resonator

the LEDs labeled "PLL1" and "PLL2" on the LMK04828 clock synthesizer are likely used as visual indicators to show the status of the internal phase-locked loops (PLLs). The specific behavior of these LEDs (e.g., whether they are lit, blinking, or off) can provide information about the stability and synchronization of the clock signals generated by the LMK04828

A jitter cleaner (a.k.a. jitter attenuator) is a device that is used to reduce the magnitude of noise (jitter) on a given timing signal. Jitter can be described as the undesired deviation from an ideal periodic timing signal.

- sudo ./bin/prg\_rfpll -lmk /lib/firmware/bishnu\_rfsoc4x2default\_try.txt

In FPGA design, "CLK\_P" and "CLK\_N" typically refer to a pair of clock signals used in a differential clocking scheme. These pairs are often associated with High-Speed Transceiver (HSTx) or High-Speed Receiver (HSRx) interfaces that use differential signaling. Here's what these terms generally mean:

1. **CLK\_P (Clock Positive):** This is the positive or non-inverted signal in a differential pair. It carries the actual clock signal.
2. **CLK\_N (Clock Negative):** This is the negative or inverted signal in a differential pair. It is the complement of the positive signal.

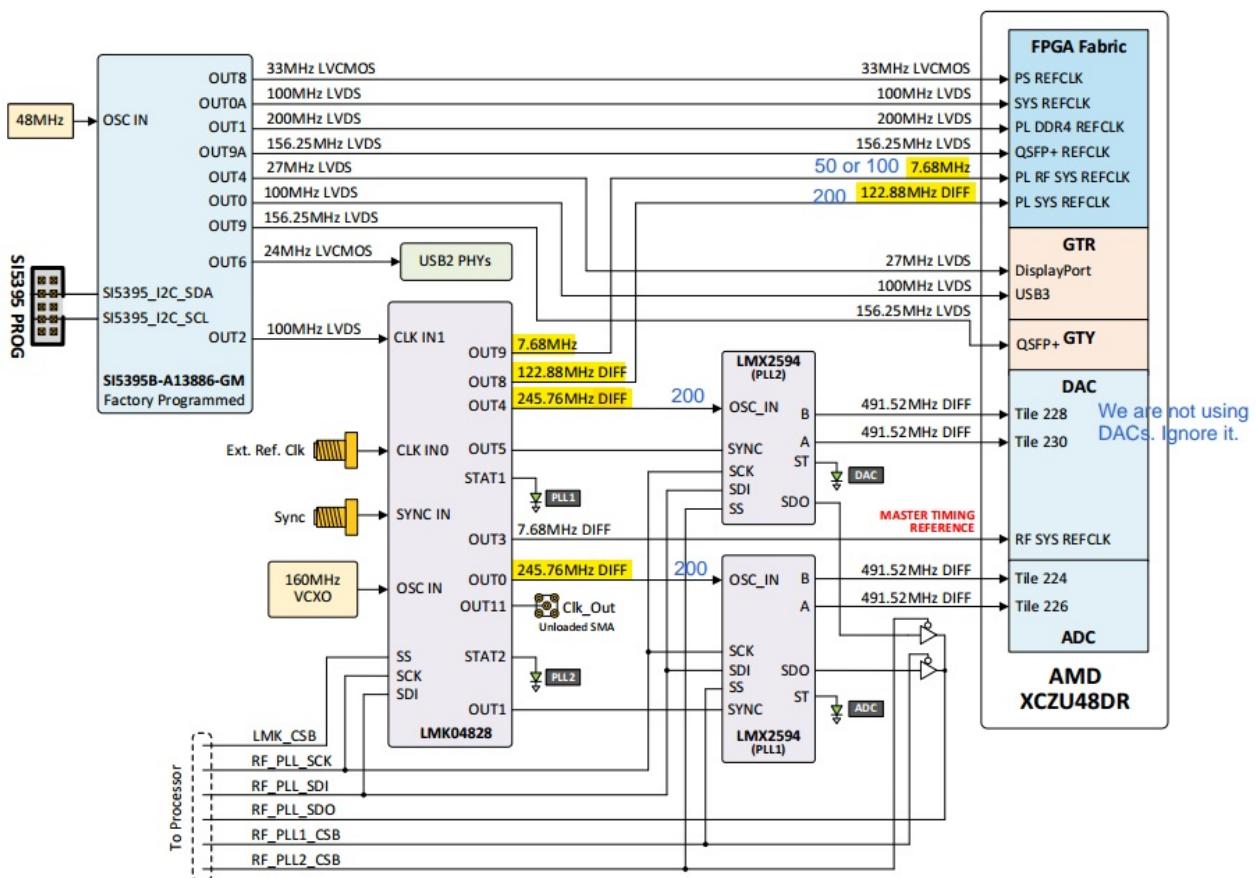
In the context of FPGA or ASIC design, a multiplexer is often used to route data or signals from multiple sources to a single destination based on control signals.

##

When you download the same file and want to delete both of them.

```
bishnu@bishnu:~/Downloads$ rm bishnu_LMK04828B_rfsoc4x2default_try
(1).txt bishnu_LMK04828B_rfsoc4x2default_try (2).txt
bishnu_LMK04828B_rfsoc4x2default_try.txt
bash: syntax error near unexpected token `'
```

```
bishnu@bishnu:~/Downloads$ rm 'bishnu_LMK04828B_rfsoc4x2default_try
(1).txt' 'bishnu_LMK04828B_rfsoc4x2default_try (2).txt'
bishnu_LMK04828B_rfsoc4x2default_try.txt
```



Here CLK IN 1 in LMK04828B is 100MHz. RFSOC\_RM\_A3

Output 2 from Si5395B, input to the CLK IN1 must be 10 MHZ, but in RFSOC\_RM\_A3 it is 100MHz (The one we were following until I checked the websites and got new user manual). The one extra '0' cost my 2 weeks to figure it out.

In electronics, it happens. The new tutorial is RFSOC\_4x2\_A\_4. Now follow it.

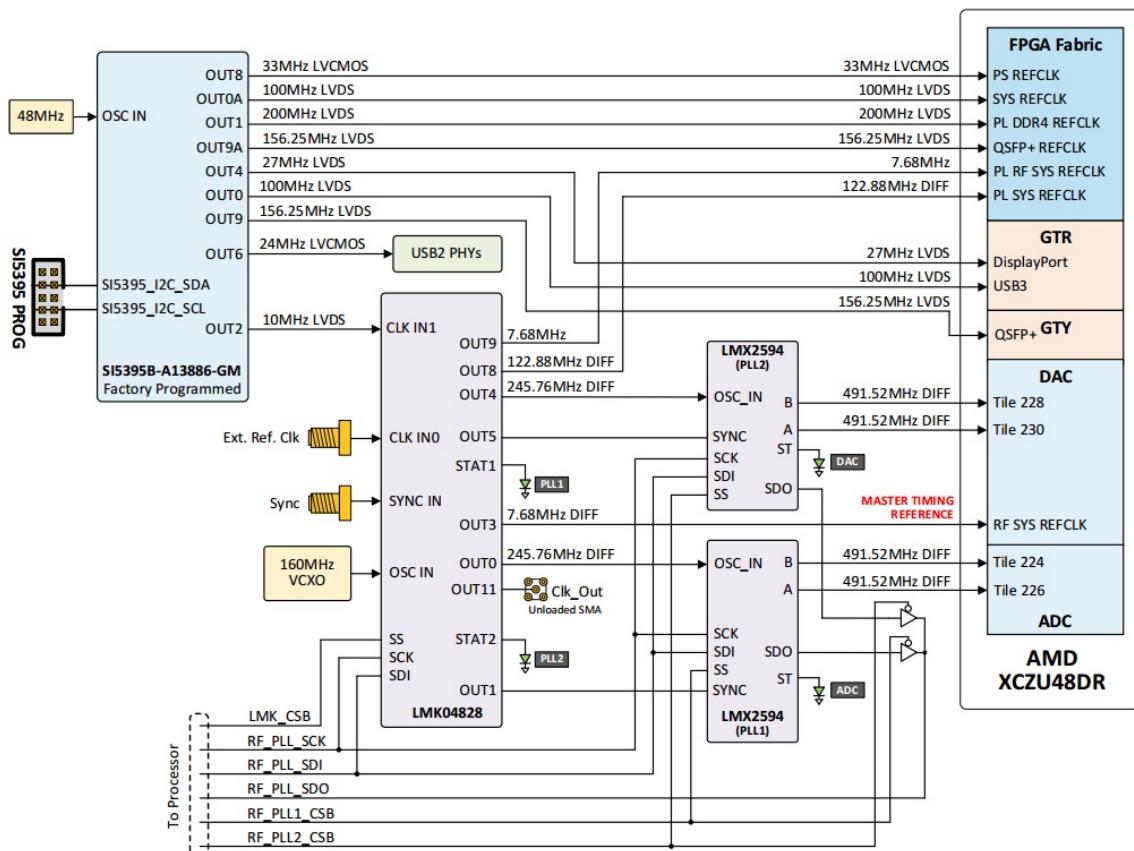


Figure 5: RFSoC Clocking Scheme

```
RFSoC 4x2
In [17]: c = rfdc_rfsoc4x2.show_clk_files()

In [18]: c
Out[18]: ['rfsoc4x2_LMX_REF_245M76_OUT_491M52.txt',
 'rfsoc4x2_PL_122M88_REF_245M76.txt']

In [19]: rfdc_rfsoc4x2.progpll('lmk', c[1])
Out[19]: True

In [20]: rfdc_rfsoc4x2.progpll('lmx', c[0])
Out[20]: True
```

These are the clock file, we tried to follow tutorial for LMK this is the result.

```
sudo ./bin/prg_rfpll -lmk
/lib/firmware/rfsoc4x2_LMX_REF_245M76_OUT_491M52.txt
loaded the following config:
```

```
0x700000, 0x6f0000, 0x6e0000, 0x6d0000, 0x6c0000, 0x6b0000, 0x6a0000,
0x690021, 0x680000,
0x670000, 0x663f80, 0x650011, 0x640000, 0x630000, 0x620200, 0x610888,
0x600000, 0x5f0000,
0x5e0000, 0x5d0000, 0x5c0000, 0x5b0000, 0x5a0000, 0x590000, 0x580000,
0x570000, 0x560000,
0x55d300, 0x540001, 0x530000, 0x521e00, 0x510000, 0x506666, 0x4f0026,
0x4e00e5, 0x4d0000,
0x4c000c, 0x4b0940, 0x4a0000, 0x49003f, 0x480001, 0x470081, 0x46c350,
0x450000, 0x4403e8,
0x430000, 0x4201f4, 0x410000, 0x401388, 0x3f0000, 0x3e0322, 0x3d00a8,
0x3c0000, 0x3b0001,
0x3a8001, 0x390020, 0x380000, 0x370000, 0x360000, 0x350000, 0x340820,
0x330080, 0x320000,
0x314180, 0x300300, 0x2f0300, 0x2e07fc, 0x2dc0df, 0x2c1f20, 0x2b0000,
0x2a0000, 0x290000,
0x280000, 0x270001, 0x260000, 0x250104, 0x240140, 0x230004, 0x220000,
0x211e21, 0x200393,
0x1f43ec, 0x1e318c, 0x1d318c, 0x1c0488, 0x1b0002, 0x1a0db0, 0x190624,
0x18071a, 0x17007c,
0x160001, 0x150401, 0x14c848, 0x1327b7, 0x120064, 0x110117, 0x100080,
0x0f064f, 0x0e1e40,
0x0d4000, 0x0c5001, 0x0b00a8, 0x0a10d8, 0x090604, 0x082000, 0x0740b2,
0x06c802, 0x0500c8,
0x040c43, 0x030642, 0x020500, 0x010809, 0x00241c, 0x000000, 0x000000,
0x000000, 0x000000,
0x000000, 0x000000, 0x000000, 0x000000, 0x000000, 0x000000, 0x000000,
0x000000, 0x000000,
0x000000
```

rfsoc4x2 does not support register readback, only led status

```
Let's insight into some of the clock issues.
```

Changing clock file: LMX and LMK

The first one is LMK and then it is LMX

- Login to casper using :ssh -l casper@192.168.50.198

```
###LMK##
```

- sudo ./bin/prg\_rfpll -lmk /lib/firmware/rfsoc4x2\_PL\_122M88\_REF\_245M76.txt  
###LMX##

- sudo ./bin/prg\_rfpll -lmx

```
/lib/firmware/rfsoc4x2_LMX_REF_245M76_OUT_491M52.txt
```

When the clock files are compatible for FPGA, it automatically turn the LEDs inside it.

```
Moving file from your pc to FPGA or any other PC using bash
command. #####
scp[space]path_to_your_file_in_your_pc[space]nameofhostpc@IP:path_of_your_
target
```

```
scp
/home/bishnu/Desktop/m_work/work_file/clockfile/
LMK04828Bclockfile_updated.txt casper@192.168.50.198:/lib/firmware
```

#

- 7 SYSREF (System Reference) clocks

- **LVPECL (Low Voltage Positive Emitter-Coupled Logic):**

- Differential signaling standard.
- Typically used in applications requiring high-speed data transmission.
- Employs a positive emitter-coupled logic architecture.

**2. LVDS (Low Voltage Differential Signaling):**

- A widely used differential signaling standard.
- Commonly used for high-speed communication between integrated circuits over short distances.
- LVDS transmits data differentially, reducing electromagnetic interference.

**3. HSDS (High-Speed Data Serial):**

- Generic term referring to high-speed serial data transmission.
- Can encompass various specific standards such as HDMI (High-Definition Multimedia Interface) or other high-speed serial communication protocols.

**4. LCPECL (Low Current Positive Emitter-Coupled Logic):**

- Similar to LVPECL but optimized for lower power consumption.
- Uses a positive emitter-coupled logic architecture.
- Suitable for applications where power efficiency is a critical factor.

I2C = Inter integrated circuit bus developed by Philips - 1982

SPI = serial peripheral interface bus developed by Motorola - 1979

I2C and SPI are both communication protocol.

For SPI

we need 4 wire for 2 way communication

MOSI = Master Out Slave In

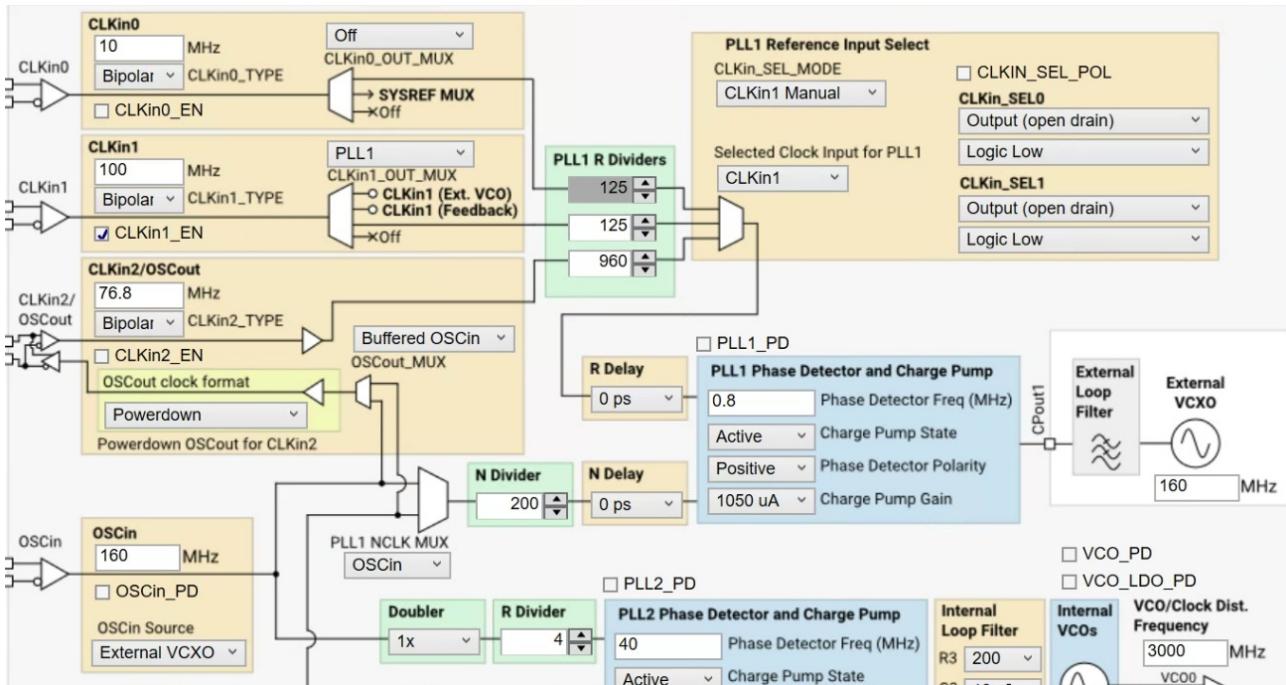
MISO = Master In Slave Out

SCLK = Serial CLock

SS = Slave Select

I2c have half duplex and SPI have half and full duples

## # Clock setting TICS pro



- First CLKin0, CLKin1, CLKin2/OSCout, In this parameters, enable which you want. Like for RFSOC, in general case, if you are not giving them external clock, use only CLKin1, that is check that parametes. CLK0 is off, CLK1 is PLL1, and CLK2 is buffered.

### DCLK (Data Clock):

- DCLK is often used to denote the clock signal that synchronizes the transfer of data within a digital system.
- It is commonly associated with the timing of data transmission, indicating when each bit of data should be sampled or read.

### 2. SDCLK (Serial Data Clock):

- SDCLK is related to serial data communication.
- In serial communication, data is sent bit by bit over a single communication line, and SDCLK is the clock signal that determines the timing of these individual bits.

The main clock file is LMK04828B. It gives led light to both PLL1 and PLL1 as can be clearly seen in RFSOC clock diagram as well.

Note: There is new manual for RFSoC4x2 and the version A4, please follow it not A3.

When I change input of CLK IN 1 in LMK04828B to 10 MHz, both PLL1 and PLL2 are locked.

---

## 100 Gbe connection

- NIC card from Mellanox technology
- Insert it into slots of your PC
- Connect 100GBE ethernet wire from PC to FPGA.

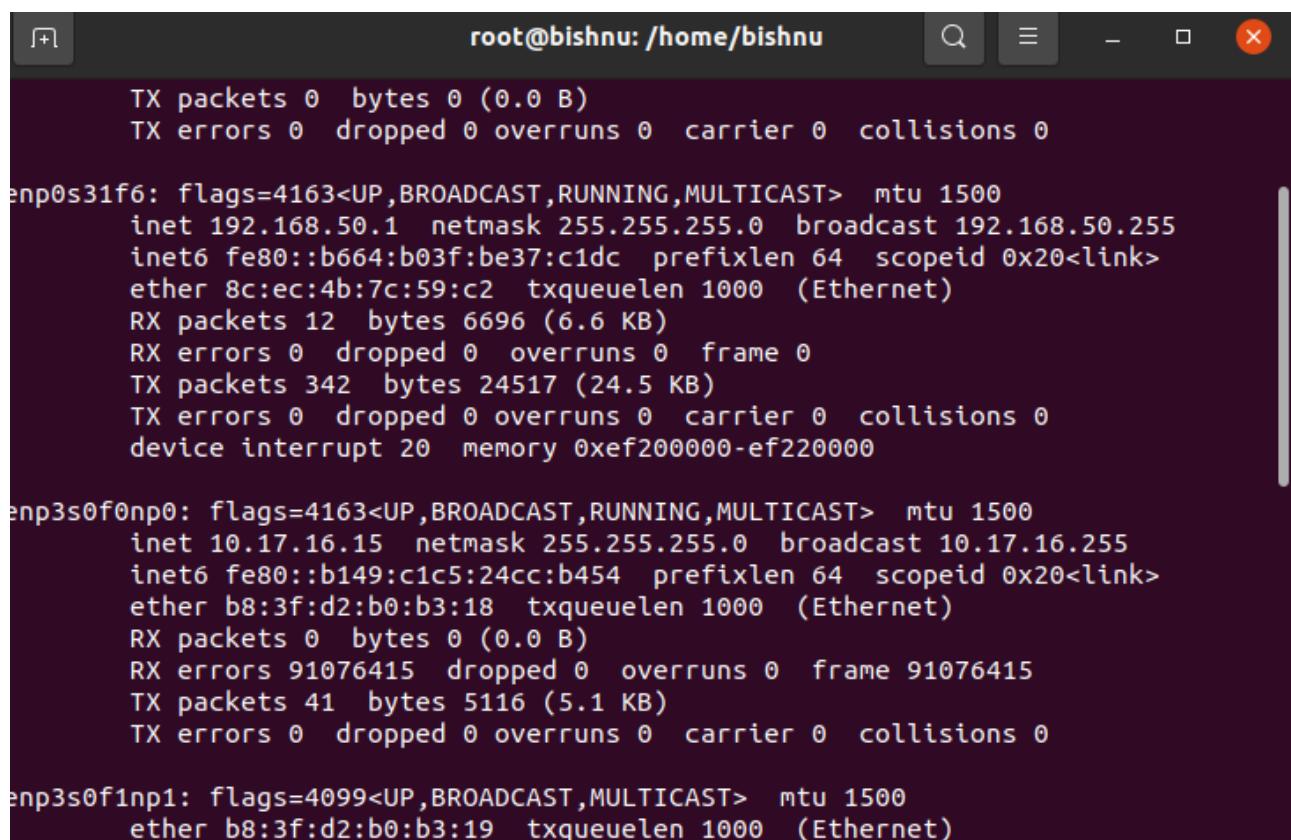
- \$sudo su

- \$ifconfig:

you will see

```
$sudo ip addr add 10.17.16.15/24 broadcast 10.17.16.255 dev
enp3s0f0np0
```

Setting IP to the 100 Gbe network.



The screenshot shows a terminal window with a dark background and light-colored text. At the top, it says "root@bishnu: /home/bishnu". Below that, there is a list of network interfaces and their configurations. The interfaces listed are enp0s31f6, enp3s0f0np0, and enp3s0f1np1. Each interface has its flags, MTU, IP address, subnet mask, broadcast address, MAC address, and various statistics like RX and TX packets, errors, and collisions. The enp3s0f0np0 interface is specifically highlighted with a yellow background.

```
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp0s31f6: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
 inet 192.168.50.1 netmask 255.255.255.0 broadcast 192.168.50.255
 inet6 fe80::b664:b03f:be37:c1dc prefixlen 64 scopeid 0x20<link>
 ether 8c:ec:4b:7c:59:c2 txqueuelen 1000 (Ethernet)
 RX packets 12 bytes 6696 (6.6 KB)
 RX errors 0 dropped 0 overruns 0 frame 0
 TX packets 342 bytes 24517 (24.5 KB)
 TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
 device interrupt 20 memory 0xef200000-ef220000

enp3s0f0np0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
 inet 10.17.16.15 netmask 255.255.255.0 broadcast 10.17.16.255
 inet6 fe80::b149:c1c5:24cc:b454 prefixlen 64 scopeid 0x20<link>
 ether b8:3f:d2:b0:b3:18 txqueuelen 1000 (Ethernet)
 RX packets 0 bytes 0 (0.0 B)
 RX errors 91076415 dropped 0 overruns 0 frame 91076415
 TX packets 41 bytes 5116 (5.1 KB)
 TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp3s0f1np1: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
 ether b8:3f:d2:b0:b3:19 txqueuelen 1000 (Ethernet)
```

There are two slots for 100 Gbe ethernet, one slot is enp3s0f0np0 and enp3s0f1np1. I am using enp3s0f0np0. At the beginning, mtu is 1500, we need to change it to more than 9000. say 9690.

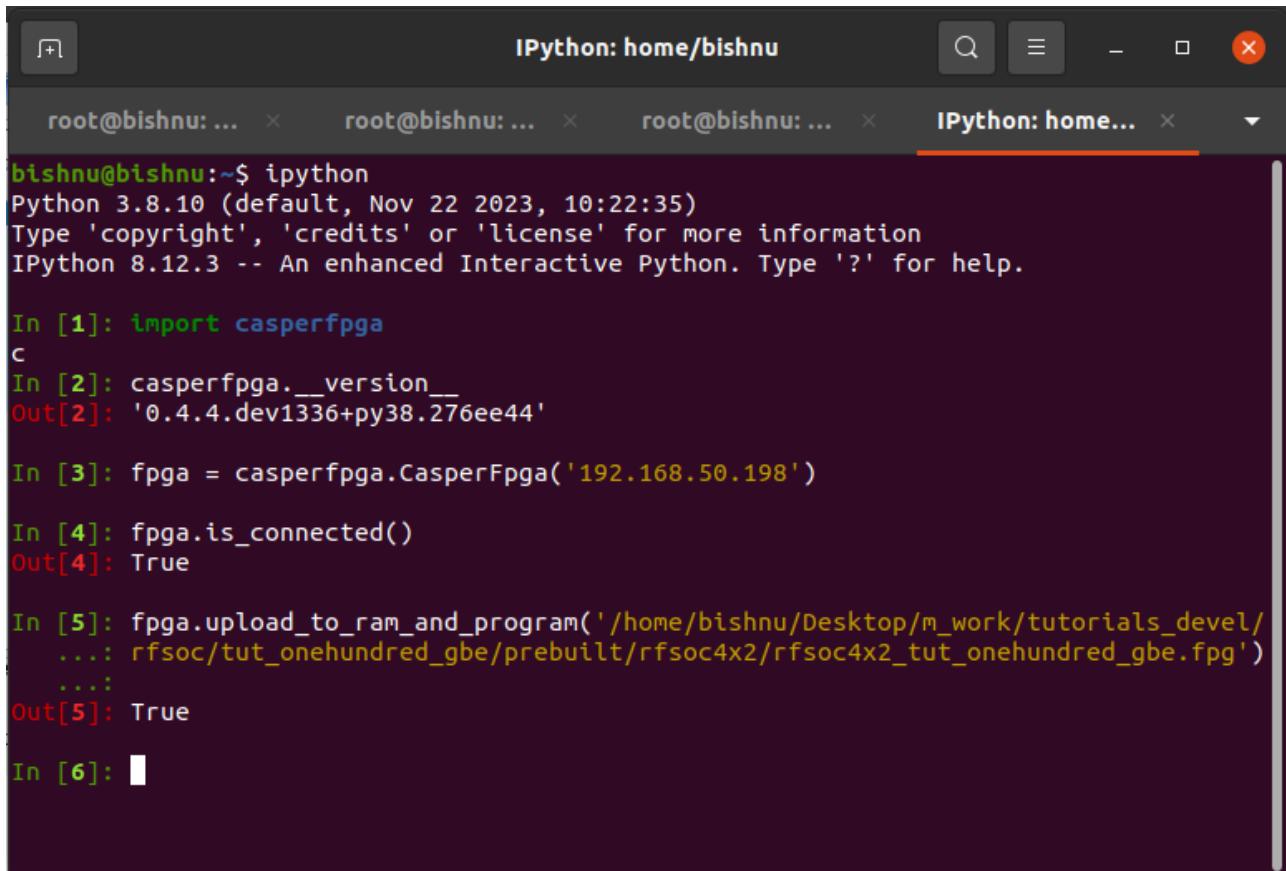
\$sudo su

\$ifconfig enp3s0f0np0 mtu 9690

\$ wireshark

Note: Wireshark must be used in root mode. \$ sudo su

Every time FPGA is shut down, make sure to load (fpga.upload\_to\_ram\_and\_program) .fpg file so that wireshark can show some signal and we can capture data.



```
IPython: home/bishnu
root@bishnu: ... × root@bishnu: ... × root@bishnu: ... × IPython: home...
bishnu@bishnu:~$ ipython
Python 3.8.10 (default, Nov 22 2023, 10:22:35)
Type 'copyright', 'credits' or 'license' for more information
IPython 8.12.3 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import casperfpga
c
In [2]: casperfpga.__version__
Out[2]: '0.4.4.dev1336+py38.276ee44'

In [3]: fpga = casperfpga.CasperFpga('192.168.50.198')

In [4]: fpga.is_connected()
Out[4]: True

In [5]: fpga.upload_to_ram_and_program('/home/bishnu/Desktop/m_work/tutorials_devel/
.... rfsoc/tut_onehundred_gbe/prebuilt/rfsoc4x2/rfsoc4x2_tut_onehundred_gbe.fpg')
....'
Out[5]: True

In [6]:
```

If we don't upload .fpg file as shown in figure above, wireshark won't show anything and you keep on going wasting time and time.

Note: When you turn-off your PC, make sure you change ifconfig file for your 100GBE network to more than mtu = 9000.

```
root@bishnu:/home/bishnu# sudo ip addr add 10.17.16.15/24 broadcast 10.17.16.255 dev
enp3s0f0np0
```

```
root@bishnu:/home/bishnu# ifconfig enp3s0f1np1 mtu 9660
```

sudo ip addr add 10.17.16.15/24 broadcast 10.17.16.255 dev enp3s0f0np0

Set IP to the network ( Use root interface)

```
$sudo ip address add <IP_ADDRESS>/<SUBNET_MASK> dev
<INTERFACE_NAME>
```

```
- $sudo ip address add 192.168.1.100/24 dev eth0
```

After assigning IP bring up the interface using:

```
$sudo ip link set <INTERFACE_NAME> up
```

```
- $sudo ip link set eth0 up
```

```
root@bishnu:/home/bishnu# ip address add 10.17.16.15/255.255.255.0 dev
enp3s0f0np0
```

Up and down network :

```
$sudo ip link set enp3s0f0np0 down
```

```
$sudo ip link set enp3s0f0np0 up
```

**\*\* Wireshark issue is solved \*\***

Next is streaming from the RFDC

The source and Destination of the Wireshark depends upon the .fpg file you upload.

```
(casper_venv) bishnu@bishnu:~/Desktop/m_work/tutorials-devel/rfsoc/tut_onehundre
d_gbe/py$ python tut_100g_listener.py 192.168.50.198 -n 256 -b /home/bishnu/Desktop/m
_work/tutorials-devel/rfsoc/tut_onehundred_gbe/rfsoc4x2_stream_rfcd_100g/ou
tputs/rfsoc4x2_stream_rfcd_100g_2024-02-19_1135.fpg -a 3
```

```
(casper_venv) bishnu@bishnu:~/Desktop/m_work/tutorials-devel/rfsoc/tut_onehundre
d_gbe/py$ sudo /home/bishnu/Desktop/m_work/casper_venv/bin/python tut_100g_catch
er.py enp3s0f0np0
opening ethernet interface enp3s0f0np0
configured to catching 256 packets each with 2048 time samples
```

Terminte with keyboard interrupt here, this will also close the listener  
^Cnding packet sequence 59...

all done!

```
(casper_venv) bishnu@bishnu:~/Desktop/m_work/tutorials-devel/rfsoc/tut_onehundre
d_gbe/py$
```

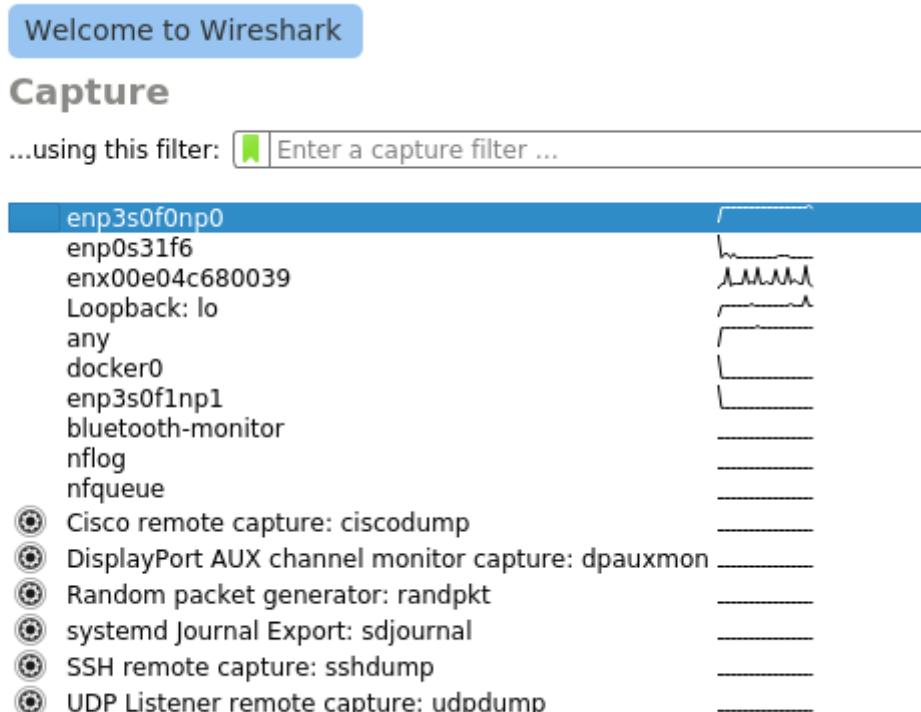
Use **root** to use wire shark as well as listener.:

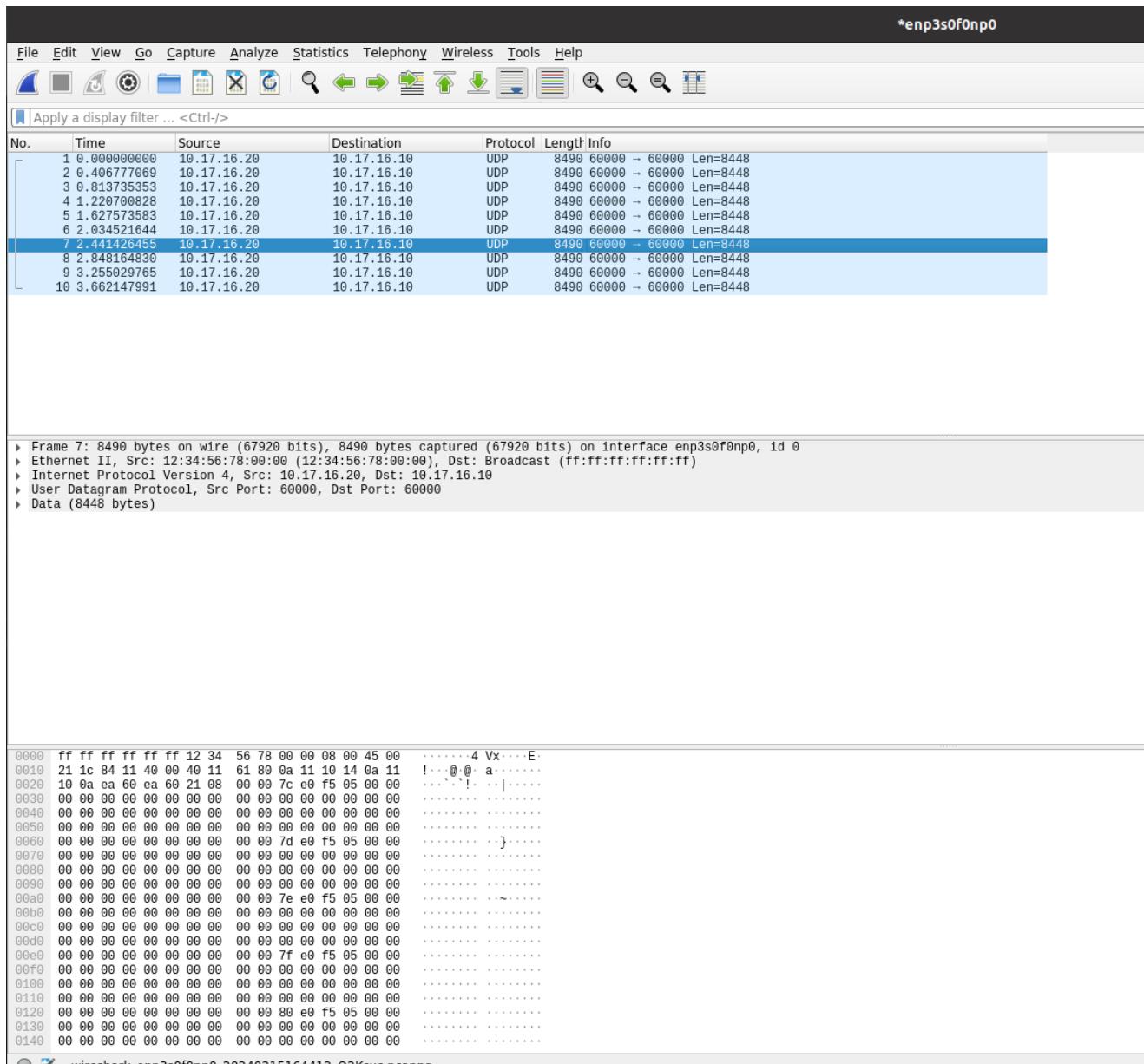
```
sudo /home/bishnu/Desktop/m_work/casper_venv/bin/python
tut_100g_catcher.py enp3s0f0np0
```

Just python like in tutorial does't work all the time.

**Specify the Python interpreter: Instead of relying on the default python command, you can specify the full path to the Python interpreter within your virtual environment when running the script with sudo. For example:**

```
sudo /home/bishnu/Desktop/m_work/casper_venv/bin/python
tut_100g_catcher.py enp3s0f0np0
```





See the 100 Gbe network, enp3s0f0np0, the increment curve is going on and on and on.

```
#####
```

#### - Default-

At boot the RFSoC 4x2 the LMK will be programmed to provide a 122.88 MHz reference to the PL and a 245.76 MHz reference for the ADC and DAC LMX. The LMX PLLs are programmed to provide a reference of 491.52 reference to the RFDC ADC and DAC tiles.

#### - Customize for our need

At boot the RFSoC 4x2 the LMK will be programmed to provide a 100 MHz reference to the PL and a 200 MHz reference for the ADC and DAC LMX. The LMX PLLs are programmed to provide a reference of 400 reference to the RFDC ADC and DAC tiles.

### #Some Warning and ERROR in RFSOC4x2 design

- When you design the simulink block, make sure you give the name of the design at first before using any blocks from simulink library.
- Then start download blocks from the simulink library.
- Make sure you look deep inside the RFDC block, which is indeed very important.
- The name of the blocks inside the RFDC must match with the file name, else you will get only -1 data plots after compiling that designs.
- From tutorial devel files, if you change the name of the file, then the designed is compiled but the plot is only -1.
- This is caused due to the conflict of file name and the name of blocks inside the RFDC.

## VIVADO design opening

open\_bd\_design

```
{/home/bishnu/Desktop/m_work/work_file/mux_ok/my_adc/my_adc.srcs/sources_1/bd/design_adc/design_adc.bd}
```

##### How to add RFSOC4x2 board in your Vivado design suits #####

1. <https://www.realdigital.org/hardware/rfsoc-4x2>

2. This is the website of the real digital. From here, Download the board files for rfsoc4x2.

- RFSoC Reference Materials
- RFSoC Board Files
- BOM
- Gen3 Mechanical Drawing Mounting Holes
- Reference Designators
- Gerbers

3. Once you download the board files from this website, unzip.

4. Copy this unzipped files into the Vivado.

```
bishnu@bishnu:~/Downloads$ sudo cp -r rfsoc4x2_board_files
/tools/Xilinx/Vivado/2021.1/data/xhub/boards/XilinxBoardStore/boards/Xilinx
```

[sudo] password for bishnu:

```
bishnu@bishnu:~/Downloads$ cp rfsoc4x2_board_files /tools/Xilinx/Vivado/2021.1
/data/xhub/boards/XilinxBoardStore/boards/Xilinx
cp: -r not specified; omitting directory 'rfsoc4x2_board_files'
bishnu@bishnu:~/Downloads$ cp -r rfsoc4x2_board_files /tools/Xilinx/Vivado/20
21.1/data/xhub/boards/XilinxBoardStore/boards/Xilinx
cp: cannot create directory '/tools/Xilinx/Vivado/2021.1/data/xhub/boards/Xili
nxBoardStore/boards/Xilinx/rfsoc4x2_board_files': Permission denied
bishnu@bishnu:~/Downloads$ sudo cp -r rfsoc4x2_board_files /tools/Xilinx/Viva
do/2021.1/data/xhub/boards/XilinxBoardStore/boards/Xilinx
[sudo] password for bishnu:
bishnu@bishnu:~/Downloads$
```

You can see the board name inside the Vivado design suite, before and after adding this files.

```
bishnu@bishnu:/tools/Xilinx/Vivado/2021.1/data/xhub/boards/XilinxBoardStore/boards/Xilinx$ ls
ac701 kcu1500 vc709 vcu129 zc702 zcu1275
k26c kv260 vck190 vcu1525 zc706 zcu1285
k26i li-imx274-mipi vcu108 vermeo_t1_mpsoc zcu102 zcu208
kc705 som240 vcu110 vermeo_t1_rfsoc zcu104 zcu208ld
kcu105 sp701 vcu118 vmk180 zcu106 zcu216
kcu116 vc707 vcu128 xm105 zcu111 zcu216ld
bishnu@bishnu:/tools/Xilinx/Vivado/2021.1/data/xhub/boards/XilinxBoardStore/boards/Xilinx$ pwd
/tools/Xilinx/Vivado/2021.1/data/xhub/boards/XilinxBoardStore/boards/Xilinx
bishnu@bishnu:/tools/Xilinx/Vivado/2021.1/data/xhub/boards/XilinxBoardStore/boards/Xilinx$ ls
ac701 kv260 vck190 vermeo_t1_mpsoc zcu104 zcu216
k26c li-imx274-mipi vcu108 vermeo_t1_rfsoc zcu106 zcu216ld
k26i rfsoc4x2_board_files vcu110 vmk180 zcu111
kc705 som240 vcu118 xm105 zcu1275
kcu105 sp701 vcu128 zc702 zcu1285
kcu116 vc707 vcu129 zc706 zcu208
kcu1500 vc709 vcu1525 zcu102 zcu208ld
bishnu@bishnu:/tools/Xilinx/Vivado/2021.1/data/xhub/boards/XilinxBoardStore/boards/Xilinx$
```

Now, if you search rfsoc4x2 in vivado, in board names, you can find it there also. Now you can design your rfsoc4x2 board there.

## # Opening python interface for compiling jasper\_frontend file.

```
(casper_venv) bishnu@bishnu:~/Desktop/caspertool/mlib-devel$
source ./startsg startsg.local
```

Reading local definitions from 'startsg.local'

Found python on start variable

Using MATLAB\_PATH=/usr/local/MATLAB/R2021a

Using XILINX\_PATH=/tools/Xilinx/Vivado/2021.1

Using COMPOSER\_PATH=/tools/Xilinx/Model\_Composer/2021.1

Using PLATFORM=lin64

Using MLIB\_DEVEL\_PATH=/home/bishnu/Desktop/caspertool/mlib-devel

Using JASPER\_BACKEND=vitis

Using XML2VHDL\_PATH=

Using LD\_PRELOAD=

Using DSP\_HDL\_DEVEL\_PATH =

Using python: /home/bishnu/Desktop/m\_work/casper\_venv/bin/python

## # Opening Matlab

```
- (casper_venv)
```

```
bishnu@bishnu:~/Desktop/caspertool/mlib-devel$./startsg
startsg.local
```

Reading local definitions from 'startsg.local'

Found python on start variable

Using MATLAB\_PATH=/usr/local/MATLAB/R2021a

Using XILINX\_PATH=/tools/Xilinx/Vivado/2021.1

Using COMPOSER\_PATH=/tools/Xilinx/Model\_Composer/2021.1

Using PLATFORM=lin64

Using MLIB\_DEVEL\_PATH=/home/bishnu/Desktop/caspertool/mlib-devel

Using JASPER\_BACKEND=vitis

Using XML2VHDL\_PATH=

```
Using LD_PRELOAD=
Using DSP_HDL_DEVEL_PATH =
Using python: /home/bishnu/Desktop/m_work/casper_venv/bin/python
Changing to directory: /home/bishnu/Desktop/caspertool/mlib-devel
Starting model_composer
Launching Model Composer and System Generator tools under a single MATLAB
session!
```

```
=====
=====
```

```
Model Composer v2021.1
```

```
Build 3247384 on Thu Jun 10 19:42:21 MDT 2021
```

```
Copyright (C) 1986-2021 Xilinx, Inc. All Rights Reserved.
```

```
=====
=====
```

## # Virtual Environment

```
bishnu@bishnu:~/Desktop/m_work$ source casper_venv/bin/activate
(casper_venv) bishnu@bishnu:~/Desktop/m_work$
```

## Control of counter or other yellow block registers

```
fpga.read_uint ('trigger', 0)
```

```
fpga.read_uint('trigger', 1)
fpga.read_uint('trigger', 0) # This is practise done by my PI. But in tutorial it is just 0 and 1.
```

Look inside for ADC block for more informations

Lastly, let's test the controllable counter:

```
In [9]: fpga.read_uint('counter_value')
Out[9]: 0

In [10]: fpga.write_int('counter_control',1)

In [11]: fpga.read_uint('counter_value')
Out[11]: 1103388123

In [12]: fpga.read_uint('counter_value')
Out[12]: 1849175237

In [13]: fpga.read_uint('counter_value')
Out[13]: 2590065552

In [14]: fpga.write_int('counter_control',0)

In [15]: fpga.read_uint('counter_value')
Out[15]: 1159837158

In [16]: fpga.read_uint('counter_value')
Out[16]: 1159837158

In [17]: fpga.write_int('counter_control',2)

In [18]: fpga.read_uint('counter_value')
Out[18]: 0
```

Note that all blocks from the "Simulink" library (usually white), will not be compiled into hardware. They are present for simulation only and expect continuous signals, not discreet.

Only Xilinx blocks (usually are blue with Xilinx logo) will be compiled to hardware.

For this reason, you need to use "gateway" blocks whenever connecting a Simulink scope or constant for simulations. Some of the CASPER blocks do this for you with "sim\_in" and "sim\_out", like the software registers.

```
In [605]: fpga.upload_to_ram_and_program("/home/bishnu/Desktop/m_work/work_file/
...: quantum_project/5_snapshots/homin_design/trigger/64_bit_adc_counter/tr
...: igger/outputs/trigger_2025-01-03_1517.fpg")
Out[605]: True

In [606]: fpga.write_int("trigger", 5000)

In [607]: fpga.write_int("trigger", 4000)

In [608]: fpga.write_int("trigger", 3000)

In [609]: fpga.write_int("trigger", 2000)

In [610]: fpga.write_int("trigger", 1000)

In [611]: fpga.write_int("trigger", 500)

In [612]: fpga.write_int("trigger", 50)

In [613]: fpga.write_int("trigger", 500)

In [614]: █
```

Roach Board support 2 ADCs, but RFSOC 4x2 board only support 1 RFDC.