DESCRIPTION

To create an outdoor cat detection system to work as doorbell.

STARTING ALGORITHM

The inspiration for the algorithm comes from:

https://pyimagesearch.com/2020/06/22/turning-any-cnn-image-classifier-into-an-object-detector-with-keras-tensorflow-and-opencv/

And the CNN classifier has been trained as per:

https://keras.io/examples/vision/image_classification_from_scratch/

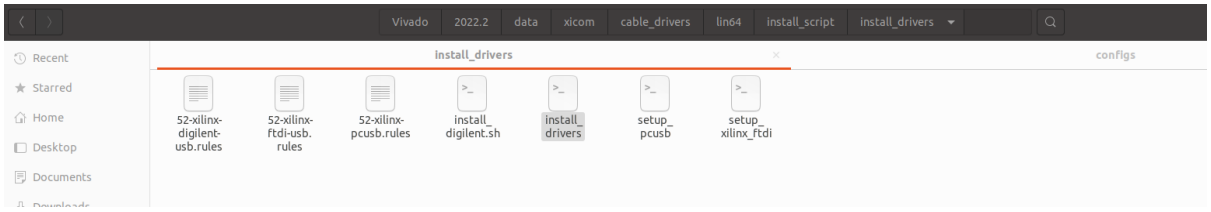
A good tutorial can be found also at:

https://python.plainenglish.io/cat-dog-classification-with-cnn-84af3ae98c44

JTAG DRIVERS INSTALLATION

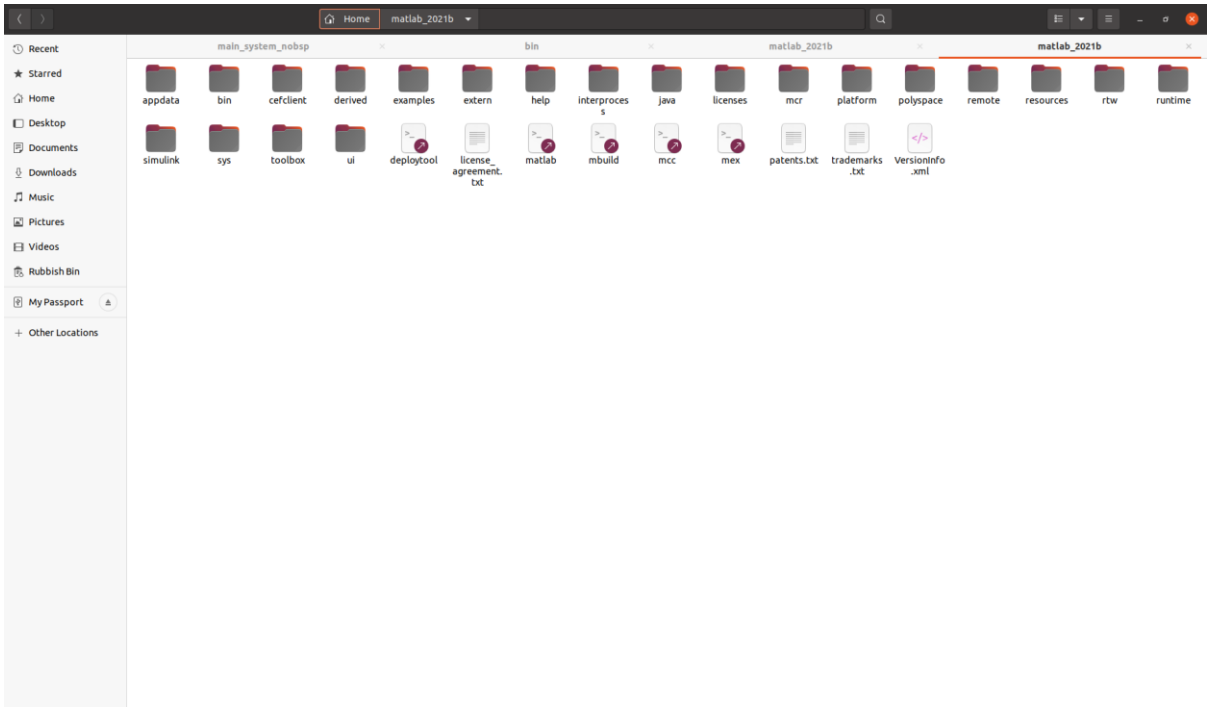
In order to allow Xilinx tools to recognize the board, cable drivers must be installed first.

To do so, go in the folder below and once there, alunch the “install_drivers” script:



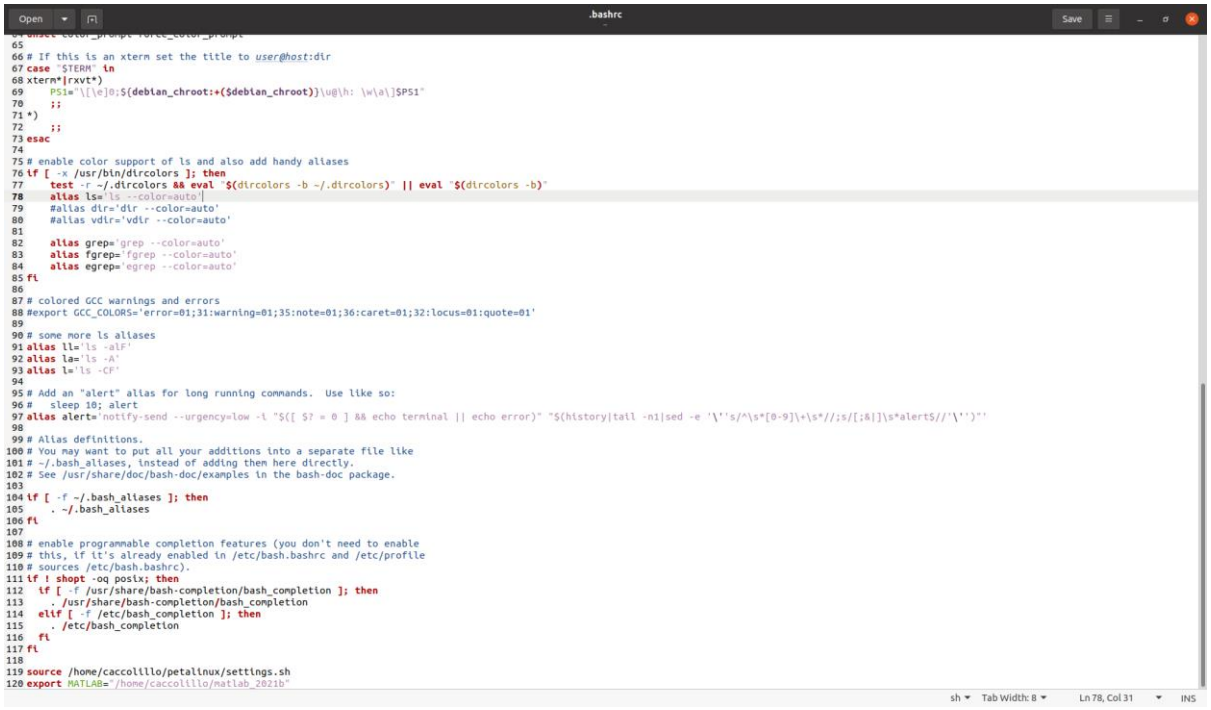
MODEL COMPOSER MATLAB/SIMULINK INSTALLATION

MATLAB 2021b has been installed in the position below:

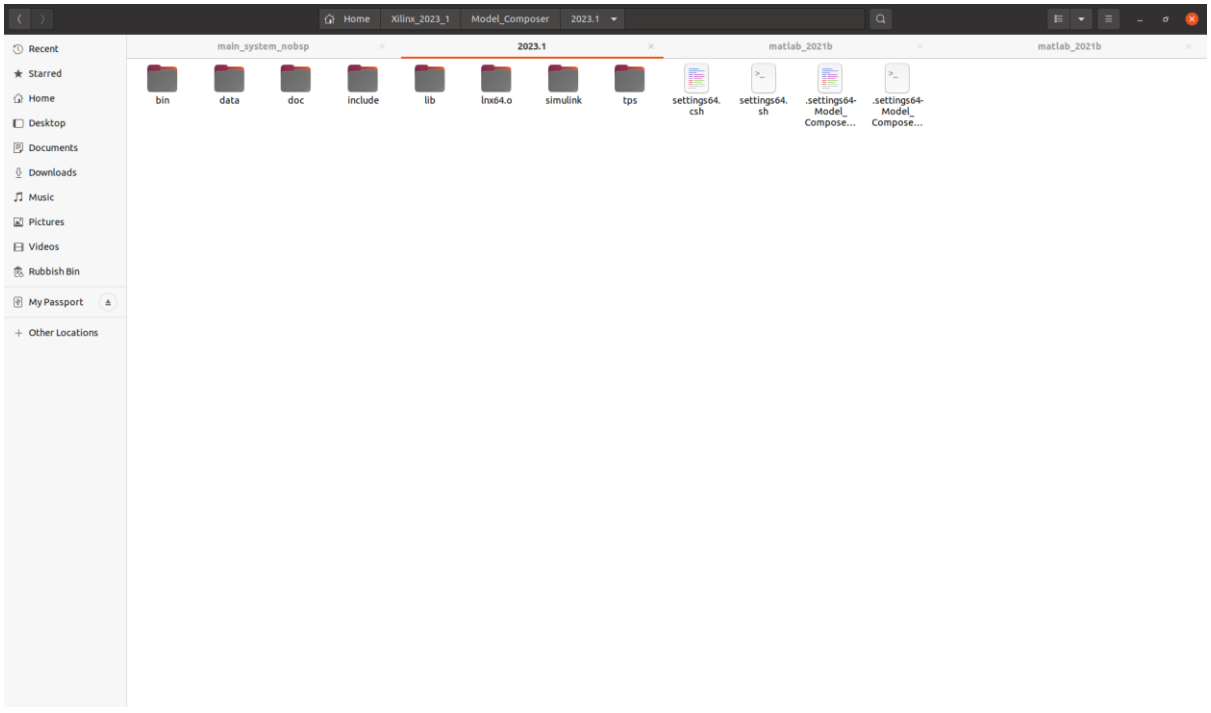


And model composer in the following one:

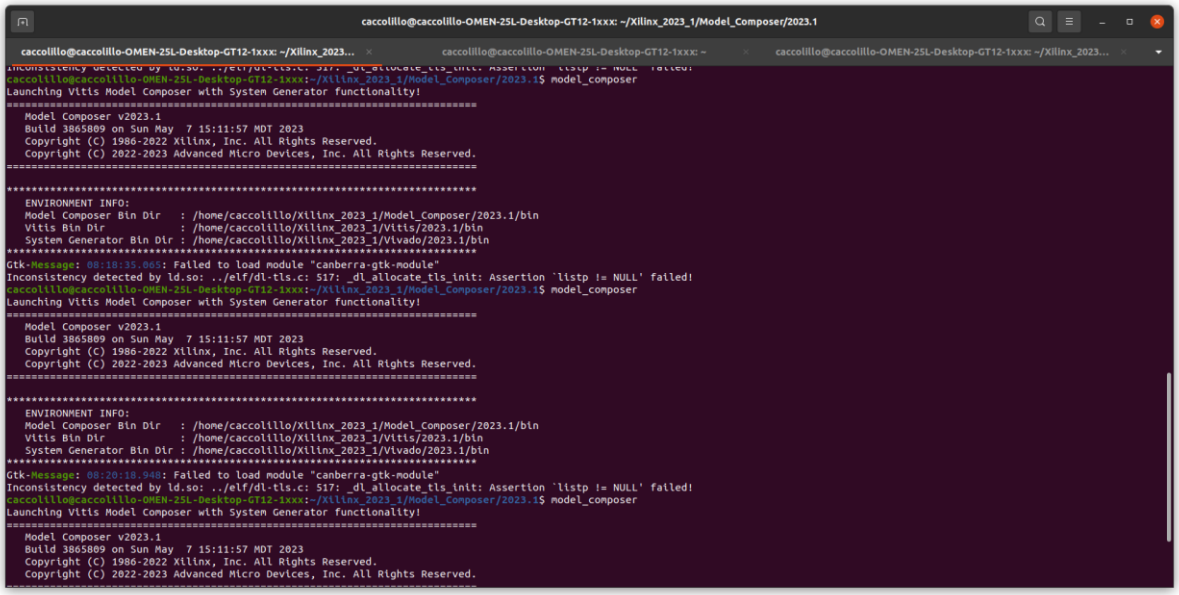
The matlab path has been added to the .bashrc file in the home folder:



To start model composer, run the script “settings64.sh”:



and on the command line give “model_composer”:



To make Simulink work, the workaround below must be followed:

https://uk.mathworks.com/matlabcentral/answers/1454674-why-does-matlab-crash-on-linux-with-inconsistency-detected-by-ld-so-elf-dl-tls-c-597-_dl_allo

Solution:
1. Install the gtk-2.0+ version of canberra by running the following command in the terminal:

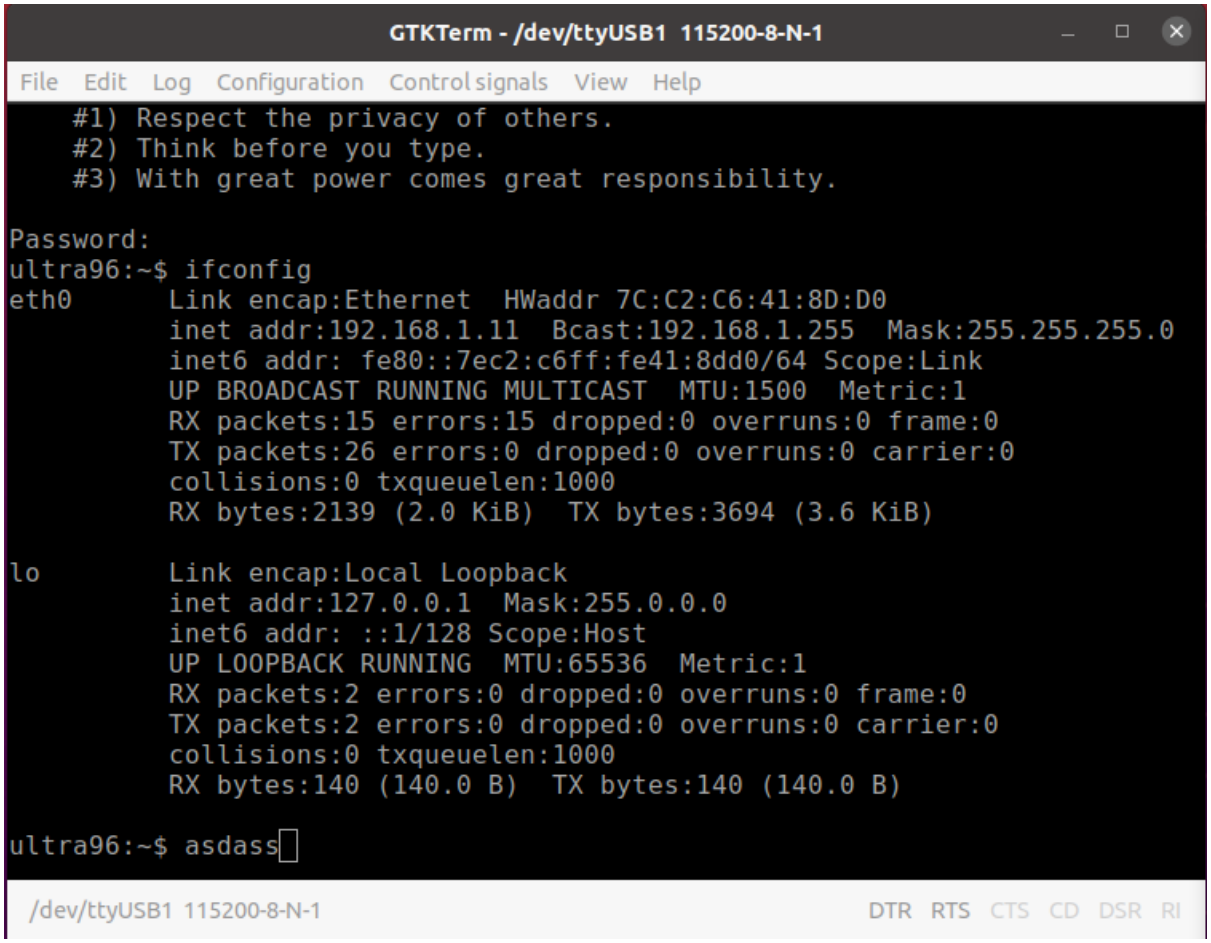
sudo apt-get install libcanberra-gtk-module –y

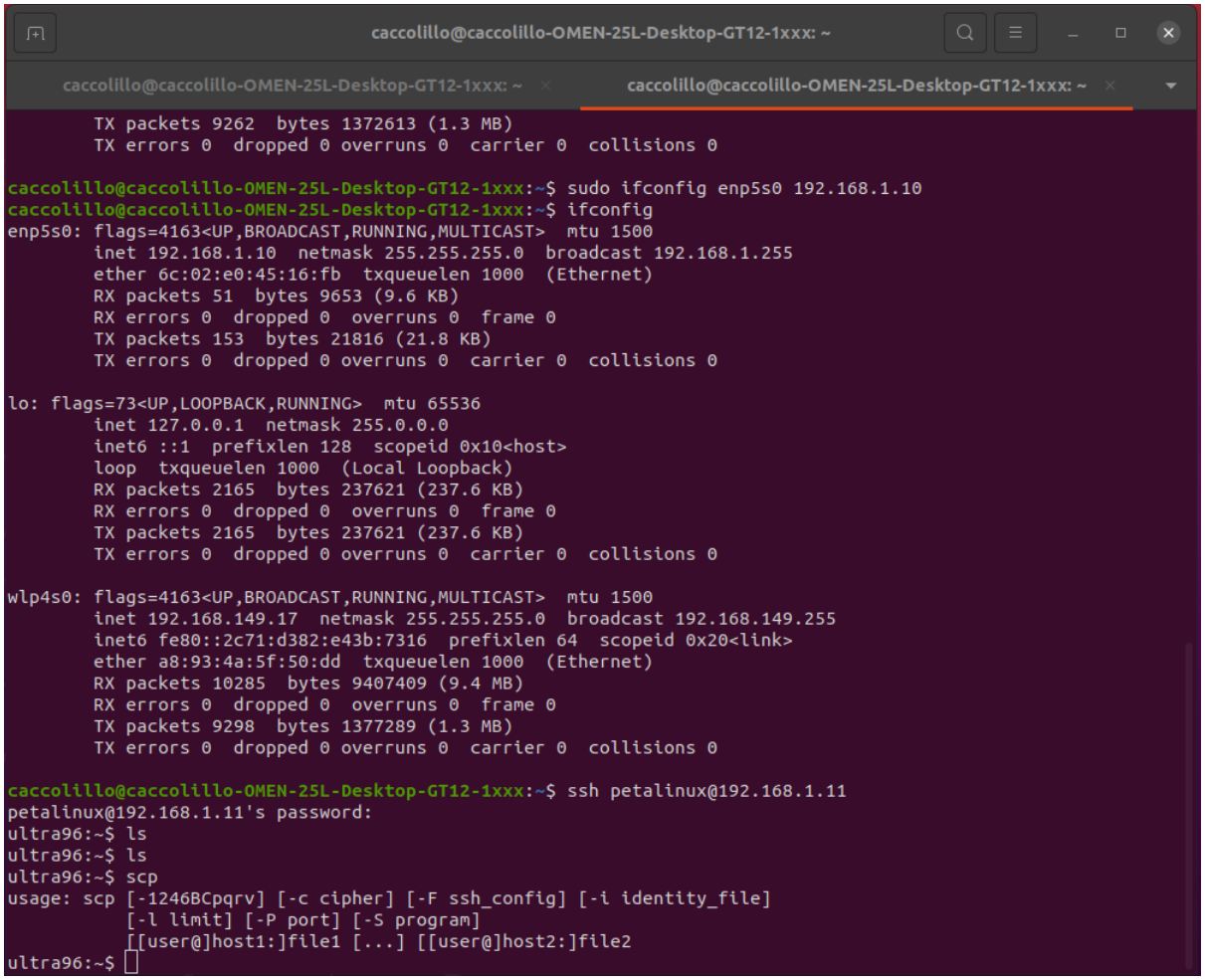
2. Add a soft connection, run the following command in the terminal (many people are missing this step):

sudo ln -s /usr/lib/x86_64-linux-gnu/gtk-2.0/modules/libcanberra-gtk-module.so /usr/lib/libcanberra-gtk-module.so

SSH and SCP to transfer files to the board

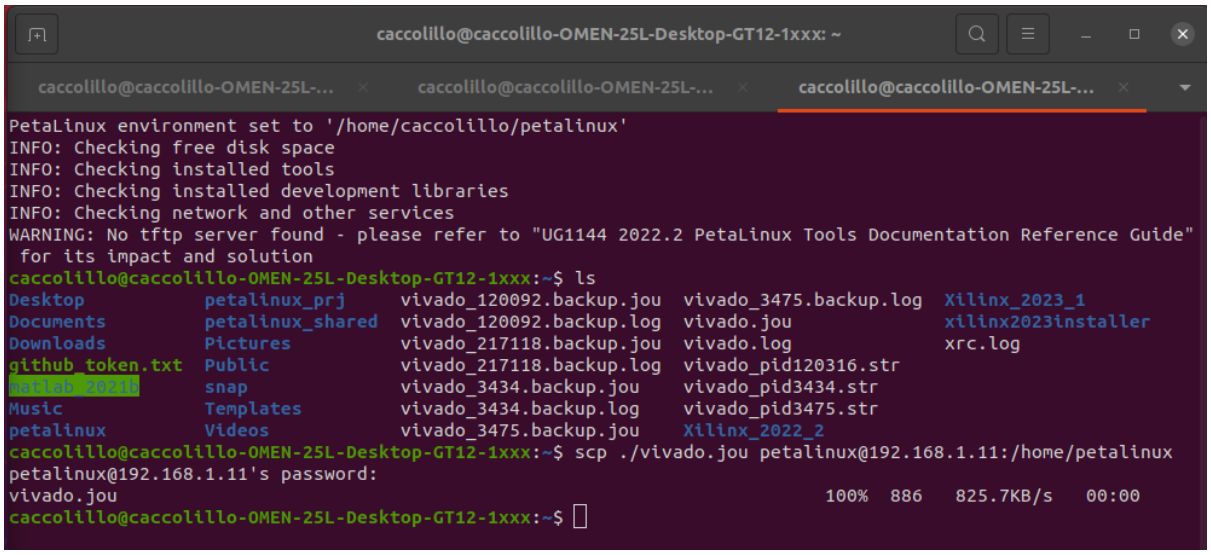
To transfer files from the host PC to the target, configure the static IP addresses of the UBUNTU PC and the Ultra 96 V2:


GTKTerm - /dev/ttyUSB1 115200-8-N-1
File Edit Log Configuration Control signals View Help
#1) Respect the privacy of others.
#2) Think before you type.
#3) With great power comes great responsibility.
Password:
ultra96:~$ ifconfig
eth0 Link encap:Ethernet HWaddr 7C:C2:C6:41:8D:D0
inet addr:192.168.1.11 Bcast:192.168.1.255 Mask:255.255.255.0
inet6 addr: fe80::7ec2:c6ff:fe41:8dd0/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:15 errors:15 dropped:0 overruns:0 frame:0
TX packets:26 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:2139 (2.0 KiB) TX bytes:3694 (3.6 KiB)
lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:2 errors:0 dropped:0 overruns:0 frame:0
TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:140 (140.0 B) TX bytes:140 (140.0 B)
ultra96:~$ asdass
/dev/ttyUSB1 115200-8-N-1
DTR RTS CTS CD DSR RI



On the linux pc connect to the Ultra96 v2 with “ssh petalinux@192.168.1.11”.

To transfer a file use the following syntax:



When it complains about the SSH key, use the trick below:

```
caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx: ~/Documents/Cat_Doorbell/python
caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx:~/Documents/Cat_Doorbell/python$ scp ./get4frames.py petalinux@192.168.1.12:/home/petalinux/
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
SHA256:G+V9VD5jIbGaTb31peno1kCUHGbHc8APKbvniccx+0M.
Please contact your system administrator.
Add correct host key in /home/caccolillo/.ssh/known_hosts to get rid of this message.
Offending RSA key in /home/caccolillo/.ssh/known_hosts:2
  remove with:
  ssh-keygen -f "/home/caccolillo/.ssh/known_hosts" -R "192.168.1.12"
RSA host key for 192.168.1.12 has changed and you have requested strict checking.
Host key verification failed.
lost connection
caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx:~/Documents/Cat_Doorbell/python$ ssh-keygen -R 192.168.1.12
# Host 192.168.1.12 found: line 2
/home/caccolillo/.ssh/known_hosts updated.
Original contents retained as /home/caccolillo/.ssh/known_hosts.old
caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx:~/Documents/Cat_Doorbell/python$ scp ./get4frames.py petalinux@192.168.1.12:/home/petalinux/
```

To copy everything from the remote to a local directory, use:

```
caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx:~/Documents/Cat_Doorbell/python/tmp$ scp petalinux@192.168.1.12:/home/petalinux/*.* ./
petalinux@192.168.1.12's password:
driver_img_avg_prj.elf                100%   80KB  16.2MB/s   00:00
frame1.png                            100%  839KB  21.5MB/s   00:00
frame1.txt                            100%   12MB  22.8MB/s   00:00
frame2.png                            100%  867KB  21.9MB/s   00:00
frame2.txt                            100%   12MB  22.8MB/s   00:00
frame3.png                            100%  877KB  22.1MB/s   00:00
frame3.txt                            100%   12MB  22.9MB/s   00:00
frame4.png                            100% 1103KB  22.3MB/s   00:00
frame4.txt                            100%   13MB  22.8MB/s   00:00
get4frames.py                         100% 2013     2.0MB/s   00:00
outputframe.txt                       100%   10MB  22.8MB/s   00:00
caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx:~/Documents/Cat_Doorbell/python/tmp$
```

THE BOARD

The algorithm has to be ported on the ULTRA96 board, using a COTS webcam as image acquisition device:

https://community.element14.com/products/devtools/avnetboardscommunity/w/boards/23079/ultra96-v2

The training material is at:

https://community.element14.com/products/devtools/avnetboardscommunity/avnetboard-training/technical-training-courses/w/documents/27905/ultra96-v2-hardware-technical-training-2021-2-lesson-1?ICID=ultra96-2021-hardwaretraining

THE OUTLINE

Use:

- Linux
- Image acquisition
- PYNQ
- OpenAMP
- Python on Zynq
- HLS for image processing acceleration
- Adding HLS devices to the device tree
- System Verilog testbench
- Verification plan
- Vivado
- Vitis
- TCL
- Bash

- C++/C

BOARD VIVADO INSTALLATION

To install the board under Vivado, follow the directions at:

https://github.com/Avnet/bdf

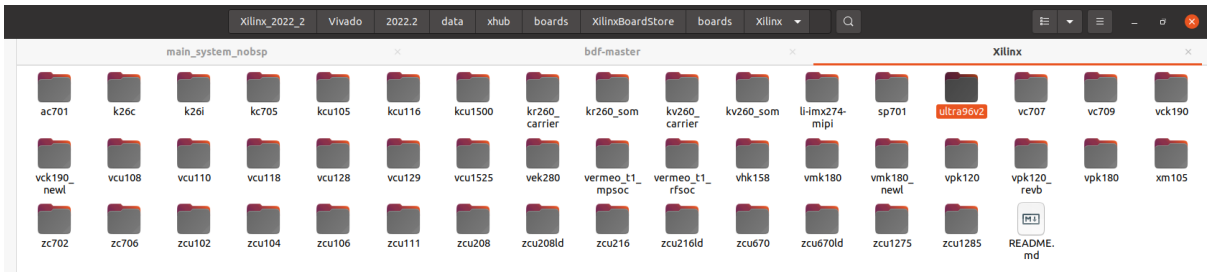
Download the repo as a zip file, then follow:

https://support.xilinx.com/s/article/The-board-file-location-with-the-latest-Vivado-tools?language=en_US

And:

Xilinx Vivado Installation and Ultra96 V2 BDF setup (Windows OS)

In this way, the board will be supported by Vivado:



CREATING A BOOTABLE IMAGE ON A MICRO-SD

Go in the petalinux image folder and create the BOOT.BIN boot file as per instructions below:

10. Change directory into **images/linux** within the lab1 project.

```
cd images/linux
```

11. Once in the project enter:

```
$ petalinux-package --boot --fsbl zynqmp_fsbl.elf --u-boot u-
boot.elf --pmufw pmufw.elf --fpga system.bit
```

Figure 6 Generate BOOT.BIN

Create a bootable micro-SD card as per instructions in the following:

14. Now that the SD card is in the Virtual machine, we can use fdisk and disks to create and format our partitions. Open disks:

Figure 8 Open Disk Application

15. In disks we can see our SD card:

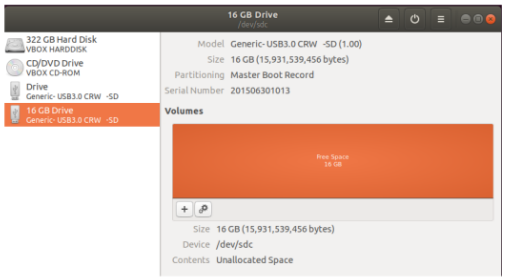


Figure 9 SD card in Disk Application

Note if you already have a partition on the drive, we recommend you delete the partitions before continuing. Simply press the "minus" button (only shows if you have partitions) on the selected partition to delete them. Continue if you have "Free Space"

16. We are going to create the first Partition as FAT32. Simply press the plus button. We recommend that you have at least 1 GB for the boot files and leave the reset for the Linux OS.

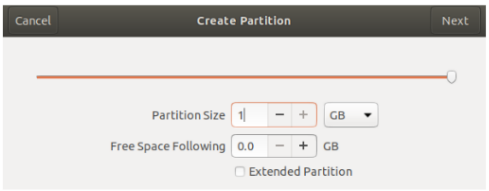


Figure 10 Create new partition 1GB

17. Next add a name for the boot partition. Leave the settings as default and press Create when done.

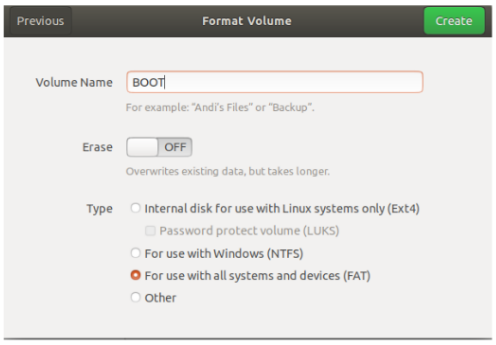


Figure 11 Name the new partition and FAT32 type

18. Now we create the second partition in the same way. Click on the free space and then the plus button.

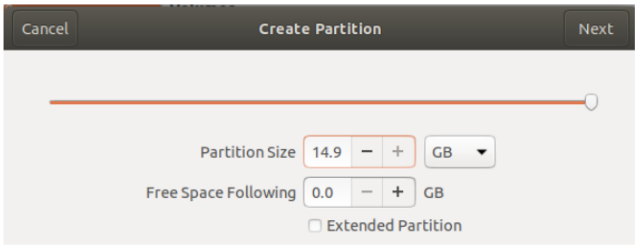


Figure 12 Create second partition with remaining free space

We are going to use the rest of the free space on the SD card for the Linux OS. Click next.

19. Name the Linux File System for the partition and **select the file system type as Ext4** click create once done.

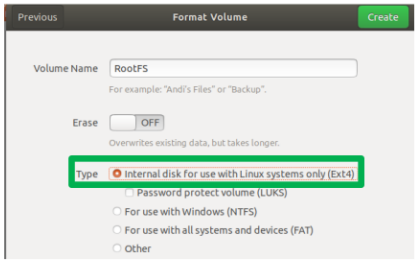


Figure 13 Name the Linux File System and EXT4

20. Open a terminal and enter:

```
sudo fdisk -l
```

This command will show all the drives on the virtual machine. Look for the SD card as we need the device name.

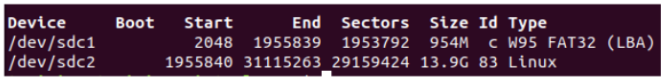


Figure 14 Check in fdisk

21. Navigate into the images/Linux folder of our PetaLinux project. In our case

```
cd ~/AvnetTTC/home/training/AvnetTTC-u96/MPSoC_PetaLInux/2021.2/Lab1
```

21. Navigate into the images/Linux folder of our PetaLinux project. In our case

```
cd ~/AvnetTTC/home/training/AvnetTTC-u96/MPSoC_PetaLInux/2021.2/Lab1
```

22. Next is to enter the command below. Make sure the correct partition is selected. Our ext4 partition is mounted to /dev/sdc2. This took just under an hour to complete

sudo dd if=images/linux/rootfs.ext4 of=/dev/sdc2 status=progress

Figure 15 Flash Completed

While we wait for the flashing process to complete, we are going to copy the boot files into the boot partition that we created earlier. The boot files that we need to copy over is the boot.scr, BOOT.BIN and image.ub.

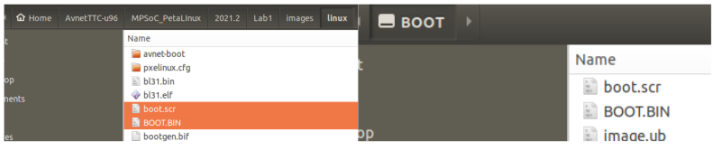


Figure 16 Copy the boot files in the BOOT partition

Insert the micro-SD card and boot as follows:

1. Power down the Ultra96 board by pressing and holding the power down, then change the Boot Mode switch on Ultra96 to 01 for SD Card as shown below.

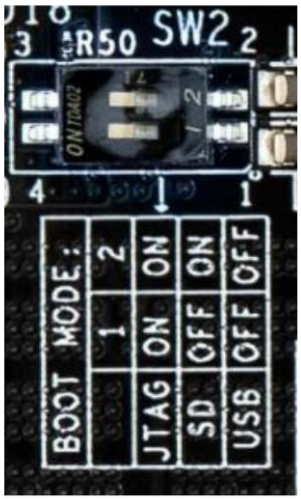


Figure 17 Set switch to SD

2. Plug-in the USB Cable to the USB to JTAG/UART pod
3. Open Terra Term and connect to the serial port at 115200, No Parity and 1 Stop
4. Power on the Ultra96 by pressing the power on button. You should see the u-boot and PetaLinux boot messages scroll in the terminal window.

The port to use for the serial terminal is /dev/ttyUSB1.

PETALINUX

The petalinux distribution gets built as per the tutorial:

https://www.hackster.io/LogicTronix/petalinux-2020-2-bsp-for-ultra96v2-with-wifi-vnc-server-4764ea

The board support package file, can be gathered from Avnet website:

https://www.avnet.com/wps/portal/us/products/avnet-boards/avnet-board-families/ultra96-v2/

Also the following guide can be followed:

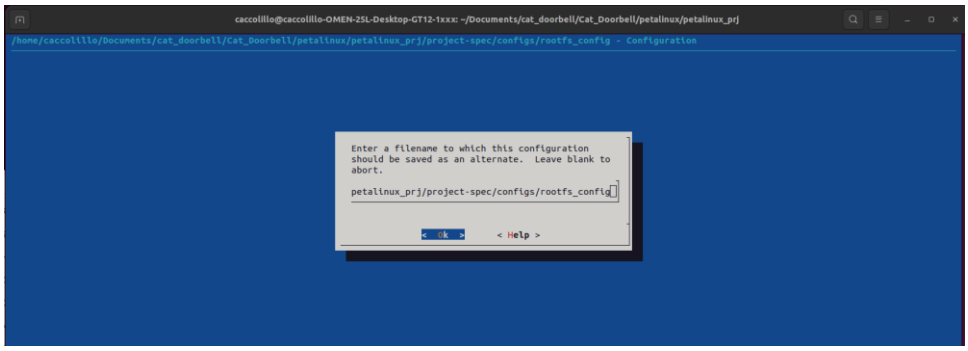
https://www.96boards.org/documentation/consumer/ultra96/ultra96-v2/build/peta-linux.md.html#install-petalinux-sdk

https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18842385/How+to+format+SD+card+for+SD+boot

Some guidance about how to enable Python and deep learning packages, is available at:

https://support.xilinx.com/s/article/1171977?language=en_US

To know where the configuration files for the kernel and the root filesystem are, it will suffice to have a look at the save menu:



An Ubuntu distribution is available at:

https://github.com/ikwzm/ZynqMP-FPGA-Ubuntu20.04

To generate the device tree when PL is involved, follow the guide:

https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/341082130/Quick+guide+to+Debugging+Device+Tree+Generator+Issues

Petalinux user manual is available at:

https://docs.xilinx.com/r/2022.1-English/ug1144-petalinux-tools-reference-guide/Installation-Requirements

And specifies which is the supported OS and Vivado release as well.

Petalinux 2022.2 supports Ubuntu 20.04 LTS and VIvado 2022.2.

To get it working, has been necessary install Ubuntu 20.04 LTS in virtual box.

In order to allow the board to get connected to the wifi network, the following file has been modified to include the network name and the wifi password:

https://logictronix.com/machine-learning-with-fpga/vitisai-dpu-dnndk/deephi-dnndk-tutorial-for-ultra96/

In order to get the Wifi and all the rest working, login as:

USER = root

PASSWORD = root

```
fonts                   lsb-release          pulse          skel                wpa_supplicant.conf.old

ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1

network={
        ssid="VodafoneMobileWiFi-6B1862"
        psk="3122407191"
}
~
~
~
~
~
"./wpa_supplicant.conf" [dos] 19L, 187B written
u96v2-sbc-base-2022-2:~# cat ./wpa_supplicant.conf
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1

network={
        ssid="VodafoneMobileWiFi-6B1862"
        psk="3122407191"
}
u96v2-sbc-base-2022-2:~# ls
/dev/ttyUSB1 115200-8-N-1                                                    DTR RTS CTS CD DSR
```

And running the script wifi.sh in the root folder, the internet connection is achieved:

```
u96v2-sbc-base-2022-2:~# ls
bt.sh  pmic-prog  usb_gadget_ethernet.sh  wifi.sh  wpa_supplicant.conf  wpa_supplicant.conf~
u96v2-sbc-base-2022-2:~# ./wifi.sh
Stopping Connection Manager
ifconfig: SIOCGIFFLAGS: No such device
Successfully initialized wpa_supplicant
nl80211: kernel reports: Authentication algorithm number required[ 1089.924856] power up request for already
 powered up source Wifi

udhcpc: started, v1.34.1
udhcpc: broadcasting discover
udhcpc: broadcasting discover
udhcpc: broadcasting select for 192.168.217.175, server 192.168.217.154
udhcpc: lease of 192.168.217.175 obtained from 192.168.217.154, lease time 3599
/etc/udhcpc.d/50default: Adding DNS 192.168.217.154
u96v2-sbc-base-2022-2:~# ping www.google.com
PING www.google.com (142.250.178.4): 56 data bytes
64 bytes from 142.250.178.4: seq=0 ttl=114 time=44.521 ms
64 bytes from 142.250.178.4: seq=1 ttl=114 time=43.148 ms
64 bytes from 142.250.178.4: seq=2 ttl=114 time=48.147 ms
64 bytes from 142.250.178.4: seq=3 ttl=114 time=47.646 ms
64 bytes from 142.250.178.4: seq=4 ttl=114 time=43.286 ms
64 bytes from 142.250.178.4: seq=5 ttl=114 time=44.862 ms
^C
--- www.google.com ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 43.148/45.268/48.147 ms
u96v2-sbc-base-2022-2:~#     dnf install python3-pip

^[[AOE Remote Repo: sswreleases sswboa     [===                              ] ---  B/s |   0  B     --:--
OE Remote Repo: sswreleases sswboardfeeds zzsoc all                          45 kB/s |  92 kB     00:02
OE Remote Repo: sswreleases sswboardfeeds zzsoc aarch64                     753 kB/s | 3.7 MB     00:05
OE Remote Repo: sswreleases sswboardfeeds zzsoc plnx_aarch64                 19 kB/s |  31 kB     00:01
Package python3-pip-21.2.4-r0.0.cortexa72_cortexa53 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
/dev/ttyUSB1 115200-8-N-1                                                    DTR RTS CTS CD DSR RI
```

Now packages can be installed from the net with the dnf command:

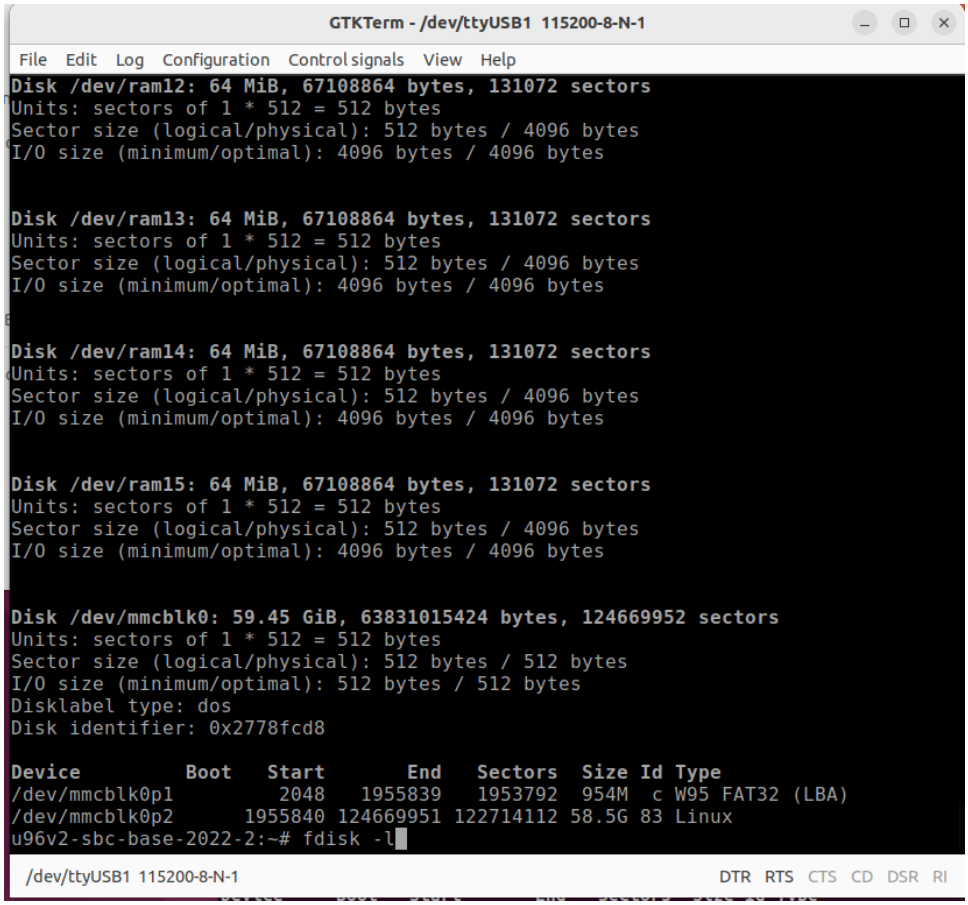
https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18841870/Install+and+run+applications+through+DNF+on+target?showComments=true&showCommentArea=true

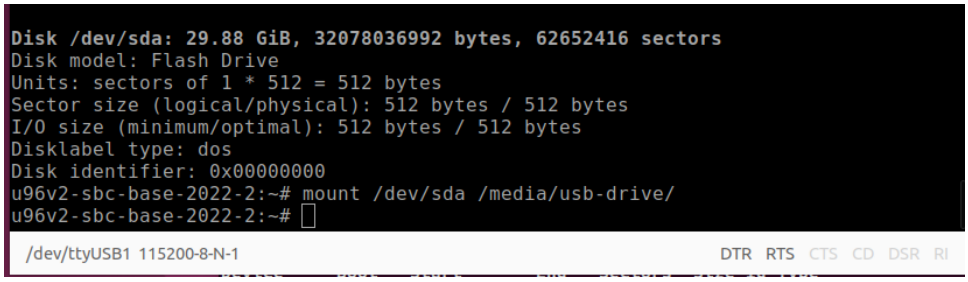
Because all gets installed in the ramdisk, and this is not big enough, issues about insufficient space gets raised.

To circumvent this, when using pip3 install, it must be instructed to use another location, such as an USB memory drive.

So the USB memory stick gets mounted, as per the following instructions:

https://linuxconfig.org/howto-mount-usb-drive-in-linux





And by following these links:

https://stackoverflow.com/questions/40755610/ioerror-errno-28-no-space-left-on-device-while-installing-tensorflow

https://askubuntu.com/questions/1326304/cannot-install-pip-module-because-there-is-no-space-left-on-device

https://superuser.com/questions/1652496/pip-install-gives-no-free-space

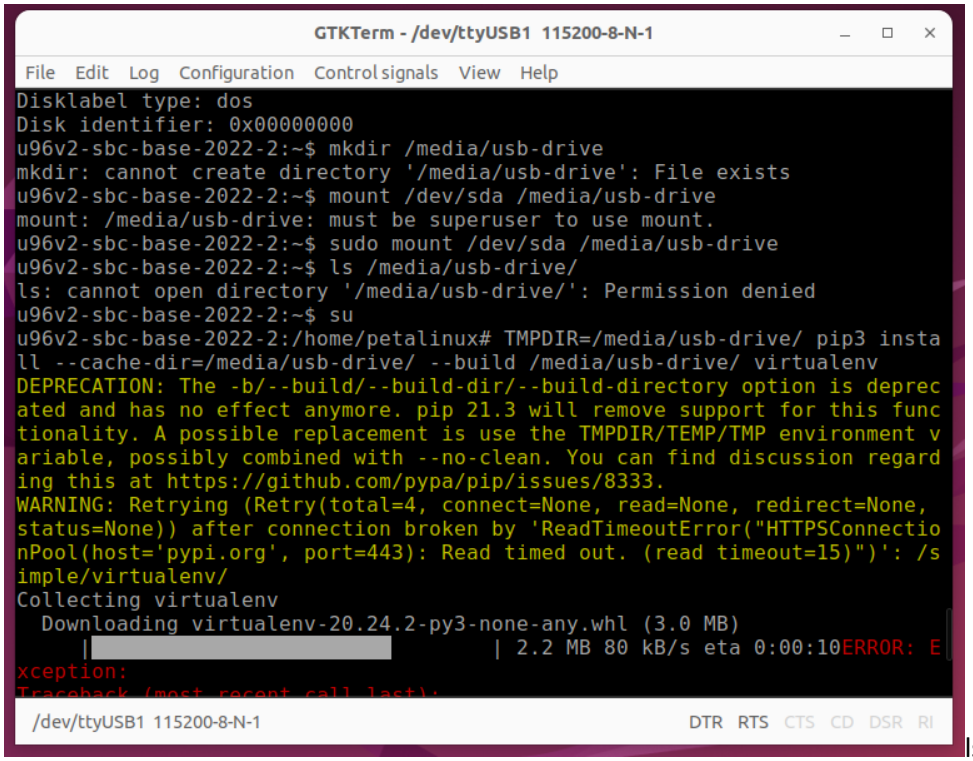
The mount point gets used as temporary workspace for pip3, to circumvent issues about limited space.

Even in this case, the space is not enough.

The virtual environment solution specified below is going to be used:

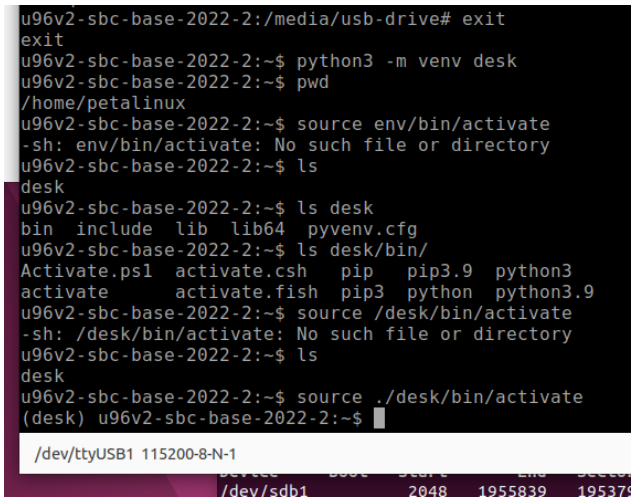
https://wiki.dave.eu/index.php/DESK-XZ7-AN-0001:_Using_Python_with_BORA

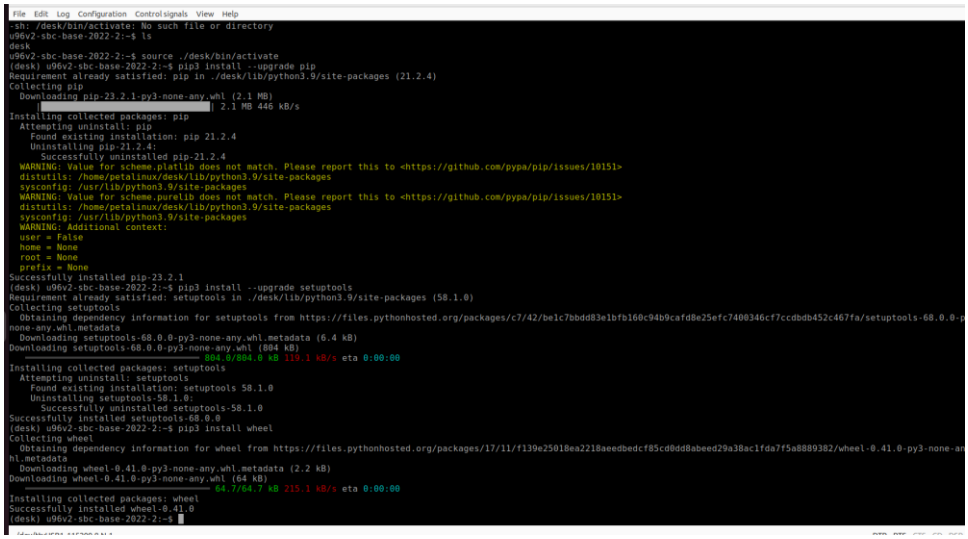
To install the virtual environment:



ls

Exit as su and create the virtual environment in the home folder:





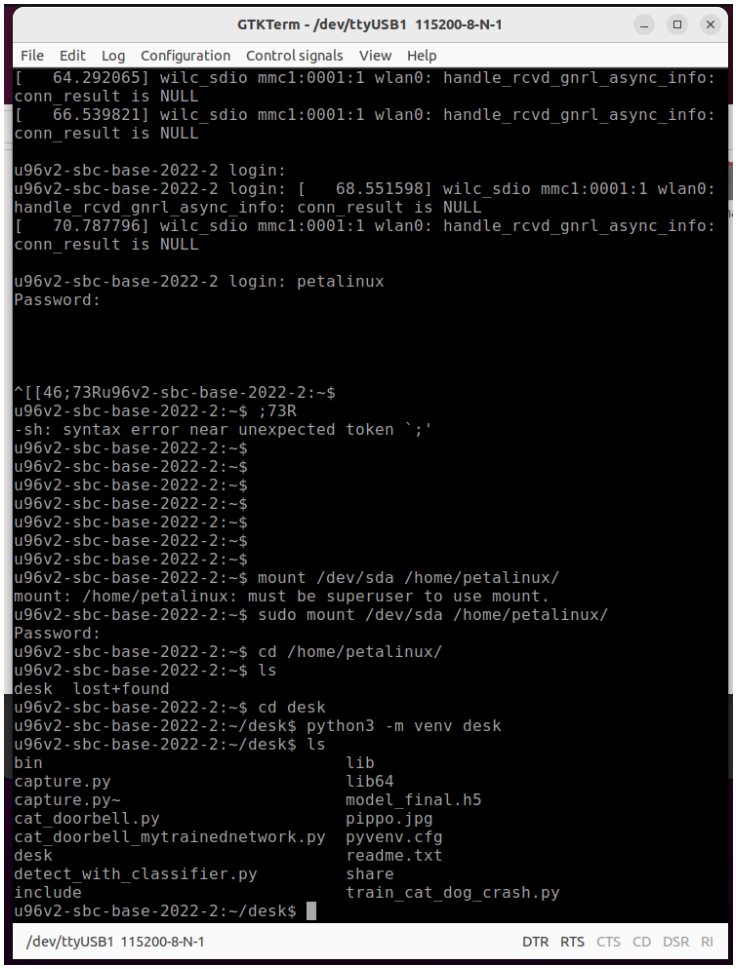
It keeps on complaining about not having enough space.

An attempt to move the home folder to the USB drive can be done:

https://ostechnix.com/move-home-directory-to-new-partition-in-linux/

The issue turned out to be that the USB pendrive had to be formatted in ext4 filesystem.

The pendrive is mounted in the home folder, and all starts fine:



To make everything work, is necessary using a monitor because of a failing QT issue.

By launching the script from a command shell on the video of the Ultra96 Linux distribution, all works as on the PC.

PYTHON APPLICATION PROFILING

The next step to follow, after having in place the Python script, is to find how to measure the time of execution of the main pieces of code, by using timers.

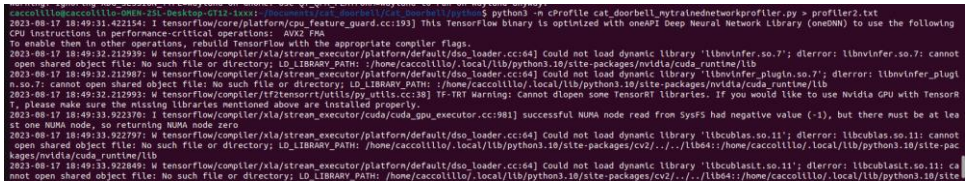
https://realpython.com/python-profiling/

https://www.geeksforgeeks.org/profiling-in-python/

It can be done via a profiler:

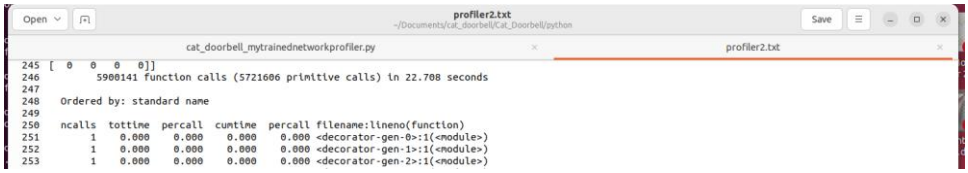
https://stackoverflow.com/questions/582336/how-do-i-profile-a-python-script

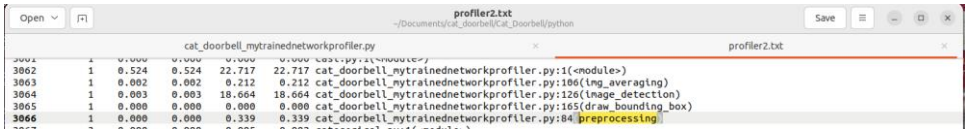
By using:



We get a text file with the profiling outcome.

From there, we get data about the key functions:



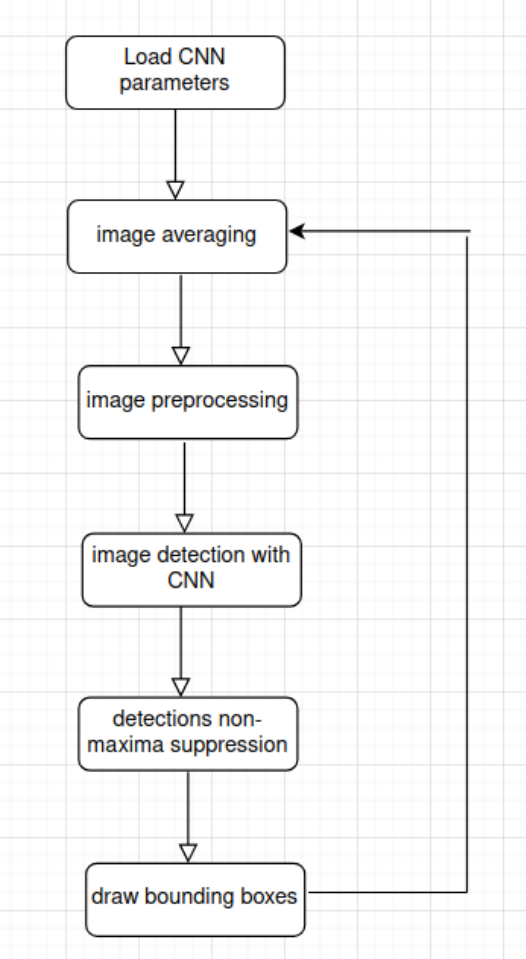


It comes out that the most computing intensive tasks are:

Function	Execution time	Description
Image_detection	18.664 s	CNN cat detector
Preprocessing	0.339 s	Illumination correction with CLAHE, and denoising (via OPENCV)
Img_averaging	0.212 s	Averages 4 consecutive frames

TASK PARTITIONING

A high-level flow chart of the python script application follows:



TASK ALLOCATION

On the Zynq US+, there are several options to implement the key tasks, in the following we will name a few.

Vitis Vision Library:

https://www.xilinx.com/products/design-tools/vitis/vitis-libraries/vitis-vision.html

Pynq:

http://www.pynq.io/

XRT for C/C++/Python:

https://www.xilinx.com/products/design-tools/vitis/xrt.html#overview

https://xilinx.github.io/XRT/2021.2/html/pyxrt.html

Linux CMA for HW accelerators:

https://support.xilinx.com/s/article/000034737?language=en_US

https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18842412/Accessing+BRAM+In+Linux

Kernels in Vitis

https://docs.xilinx.com/r/en-US/ug1393-vitis-application-acceleration

https://docs.xilinx.com/r/en-US/Vitis-Tutorials-Hardware-Acceleration

The Python code of “Image_detection” is:

```
def image_detection(image,min_conf,stepSize):
  #stepSize = 100 #stride (10)
  locs = np.array([0,0,0,0,0])
  entry = np.array([0,0,0,0,0])
  first_det = True
  (w_width, w_height) = (150, 200) # search window size
  for x in range(0, image.shape[1] - w_width , stepSize):
     for y in range(0, image.shape[0] - w_height, stepSize):
        window = image[x:x + w_width, y:y + w_height, :]
        cv2.imshow('roi' , np.array(window, dtype = np.uint8 ) )
        cv2.waitKey(1)
        time.sleep(0.12)
        tmp = image.copy()
        #cv2.rectangle(tmp, (x, y), (x + w_width, y + w_height), (255, 0, 0), 2) # draw rectangle on image
        #cv2.imshow('scan window' , np.array(tmp, dtype = np.uint8 ) )
        #cv2.waitKey(1)
        time.sleep(0.12)
        #classify with resnet50
        image_resnet50 = cv2.resize(window, (180, 180)) #resize in the format expected by resnet50
        image_resnet50 = img_to_array(image_resnet50) #convert to numpy array
        #cv2.imshow('scan window resnet' , np.array(image_resnet50, dtype = np.uint8 ) )
        #cv2.waitKey(1)
        #time.sleep(0.12)

        image_resnet50 = np.expand_dims(image_resnet50, axis=0) #adds a dimension to the image
        preds = model.predict(image_resnet50)[0] #predicts the ROI by using resnet50
        print(preds)
        score = float(preds[0])
        #print(f"This image is {100 * (1 - score):.2f}% cat and {100 * score:.2f}% dog.")
        prob_cat = 1 - score
        prob_dog = score

        # filter out weak detections by ensuring the predicted probability
        # is greater than the minimum probability
        if prob_cat >= min_conf:
            entry = [x, y, x + w_width, y + w_height, prob_cat]
            locs = np.vstack((locs,entry))
  return locs
```

And it relies upon a CNN, it is natural to accelerate it with Vitis AI.

The Python code of “preprocessing” is:

```
#image pre-processing function
def preprocessing(frame):
  tmp = frame.copy()
  #illumination correction using CLAHE
  img = cv2.cvtColor(tmp, cv2.COLOR_RGB2Lab)
  #configure CLAHE
  clahe = cv2.createCLAHE(clipLimit=10,tileGridSize=(8,8))
  #0 to 'L' channel, 1 to 'a' channel, and 2 to 'b' channel
  img[:,:,0] = clahe.apply(img[:,:,0])
  img = cv2.cvtColor(img, cv2.COLOR_Lab2RGB)
  cv2.imshow("CLAHE", img)
  #cv2.waitKey(0)
  #cv2.destroyAllWindows()
  #image denoising
  tmp2 = frame.copy()
  noiseless_image_colored = cv2.fastNlMeansDenoisingColored(tmp2,None,20,20,7,21)
  cv2.imshow("Denoised", img)
  #cv2.waitKey(0)
  #cv2.destroyAllWindows()

  return noiseless_image_colored
```

And as it relies upon OpenCV, it is natural to use the Vitis Vision libraries and an HW Kernel accelerator.

The Python code of “Img_averaging” is:

```
#frame averaging
def img_averaging(vid):
    # Capture the video frame
    # by frame
    ret, frame = vid.read()
    np_frame = np.array(frame).astype(np.float32)

    ret, frame1 = vid.read()
    np_frame1 = np.array(frame1).astype(np.float32)

    ret, frame2 = vid.read()
    np_frame2 = np.array(frame2).astype(np.float32)

    ret, frame3 = vid.read()
    np_frame3 = np.array(frame3).astype(np.float32)

    average_frame = (np_frame+np_frame1+np_frame2+np_frame3)/4
    average_frameint = average_frame .astype(np.uint8)
    return average_frameint
```

And it can be done via OpenAMP on the R5 real time cores.

Then summarizing, we will have the following HW acceleration methods in use:

Function	Hw acceleration method
Image_detection	Vitis AI
preprocessing	Vitis Vision Libraries
Img_averaging	Open AMP on the R5 core

OPEN AMP: accelerating “Img_averaging”

The home page for the Open AMP project is at:

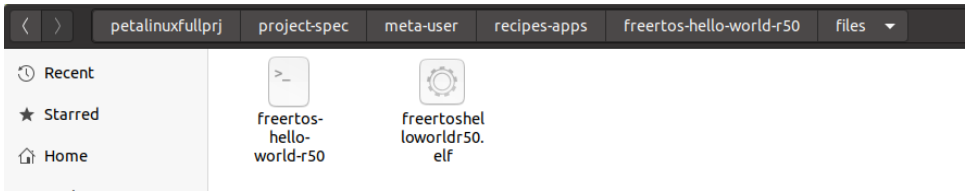
https://www.openampproject.org/

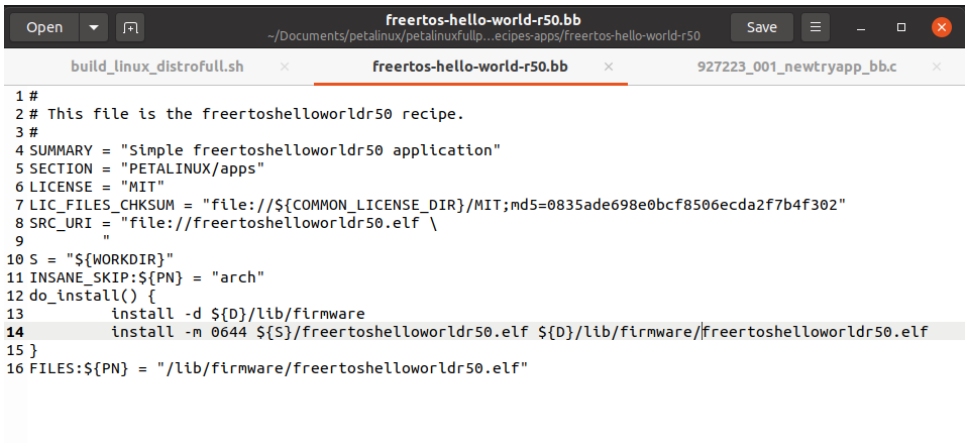
And Xilinx is among the key members.

A good tutorial about its usage on the board is at:

https://www.hackster.io/LogicTronix/openamp-in-xilinx-mpsoc-fpga-running-petalinux-baremetal-e025ca

A sample elf file for the r5 has been generated in vitis, and the instructions have been followed to create the petalinux application recipe:





The tutorial:

https://www.hackster.io/timothy-vales/openamp-on-the-snickerdoodle-black-691912

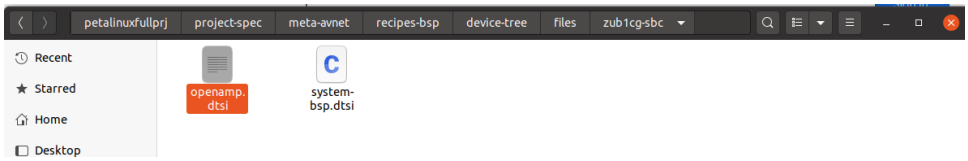
Has been followed so to configure the kernel and the root filesystem.

The bitbake recipe has been changed as per:

https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/2423488521/OpenAMP+Project+Build+Process

By replacing “ INSANE_SKIP:${PN}” to “INSANE_SKIP_${PN} ”.

The dts file has been found at:



And the Xilinx wiki is at:

https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18841718/OpenAMP

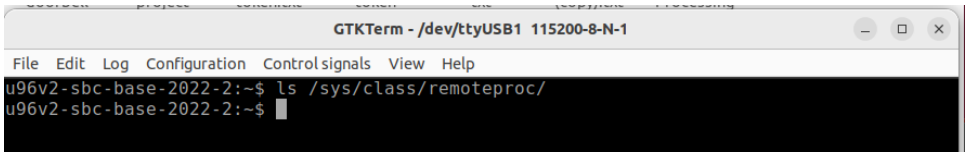
Other useful links follow:

https://support.xilinx.com/s/article/826083?language=en_US

https://www.hackster.io/sasha-falkovich/openamp-on-kria-kv260-ccbb30

https://support.xilinx.com/s/article/826083?language=en_US

The first step to check, is seeing if OpenAMP is enabled in the Petalinux distribution generated by the Avnet BSP:



The folder “remoteproc” is empty.

Petalinux must be rebuilt by enabling the missing bits.

If the build fail, check this thread:

https://support.xilinx.com/s/question/0D54U00006sZafqSAC/error-petalinuximageminimal10r0-doimagecpio-executionerrorerror-logfile-of-failure-stored-in?language=en_US

To get rid of a cpio error message.

Vitis Vision Libraries: accelerating “preprocessing”

A quickstart video introducing the library, is available at:

https://www.xilinx.com/video/software/vitis-hls-l1-library-wizard.html

Follow it to clone from git the library of interest, and to use it.

A decent video with a tutorial about how to use the libraries, is at:

https://www.youtube.com/watch?v=hyhlSFa74bs

And at:

https://www.youtube.com/watch?v=a9ozwIT98rc

A tutorial using petalinux, is at:

https://www.xilinx.com/video/software/vitis-embedded-acceleration-platform-creation-demystified.html

Vitis AI: accelerating “Image_detection”

End toolsuite selection

After having had many issues with Petalinux, Ubuntu 22.04 LTS has been chosen, for it the following guide has been followed:

https://www.hackster.io/whitney-knitter/vivado-vitis-petalinux-2023-1-install-on-ubuntu-22-04-ab28da

How to debug R5 applications on Linux via VITIS

TO debug R5 applications on a running Petalinux distribution, setup the TCF agent as per the tutorial below:

https://www.hackster.io/whitney-knitter/debugging-your-custom-linux-applications-using-vitis-67c022

In order to make it work, follow the kernel build instructions below, needed to not keep in idle the R5:

https://support.xilinx.com/s/article/69143?language=en_US

For a step by step procedure to set up the debug for bare metal r5 debug on a running linux system, look at:

https://support.xilinx.com/s/article/1225679?language=en_US

How to debug applications on Linux via VITIS

To debug Linux applications from within Vitis, follow the following tutorial:

https://www.xilinx.com/video/software/building-linux-application-vitis.html

https://www.hackster.io/whitney-knitter/debugging-your-custom-linux-applications-using-vitis-67c022

OPENMP in Vitis

https://xilinx.github.io/Vitis-Tutorials/2021-2/build/html/docs/Hardware_Acceleration/Introduction/06-meet-the-other-shoe.html

Vitis tutorials and HW accelerators

According to AVnet:

https://community.element14.com/technologies/fpga-group/b/blog/posts/avnet-hdl-git-howto-vivado-2020-2-and-later

Reference designs are available on their servers.

BSPs are provided:

https://community.element14.com/technologies/fpga-group/b/blog/posts/petalinux-2021-1-bsps-now-available-for-avnet-soms-and-sbcs-244855059

And they support the “vector addition example”.

This example is described at the home page about HW acceleration on Vitis:

https://xilinx.github.io/Vitis_Accel_Examples/2021.1/html/simple_vadd.html

https://docs.xilinx.com/r/en-US/Vitis-Tutorials-Vitis-Platform-Creation/Test-2-Run-Vector-Addition-Application?tocId=8w7OXVMcu2r75uTH6UGdhA

The landing page about these tutorials is at:

https://github.com/Xilinx/Vitis-Tutorials/tree/2023.2/Getting_Started/Vitis

And a webinar can be found at:

https://www.youtube.com/watch?v=a9ozwIT98rc

The official tutorial for the HW accelerator is at:

https://docs.xilinx.com/r/en-US/Vitis-Tutorials-Hardware-Acceleration/Design-Tutorials?tocId=ySpgAKBHTPrnnilZGgczRA

And the manual for HW accelerators under vitis:

https://docs.xilinx.com/r/en-US/ug1393-vitis-application-acceleration/Getting-Started-with-Vitis

It is possible to develop the HW accelerator in Vitis HLS, and then import the generated xo file in Vitis as per the tutorial:

https://xilinx.github.io/Vitis-Tutorials/2020-1/docs/build/html/docs/Getting_Started/Vitis_HLS/using_the_kernel.html

And an in-depth explanation of the whole flow is at:

https://www.hackster.io/whitney-knitter/vitis-acceleration-flow-on-kv260-software-in-petalinux-b2456c

https://www.hackster.io/whitney-knitter/vitis-acceleration-flow-on-kv260-vitis-platform-c3537e

Official Xilinx Vitis in depth tutorial are at:

https://github.com/Xilinx/Vitis-Tutorials

And the video about it:

https://www.xilinx.com/video/software/vitis-embedded-acceleration-platform-creation-demystified.html

And an in depth tutorial is at:

https://docs.xilinx.com/r/en-US/Vitis-Tutorials-Hardware-Acceleration/Open-the-IP-Packager

The page for the 2022.2 Vitis release is at:

https://docs.xilinx.com/r/2022.2-English/Vitis-Tutorials-Hardware-Acceleration/Design-Tutorials?tocId=ySpgAKBHTPrnnilZGgczRA

In order to use XRT, it has to be downloaded from the GIT psge:

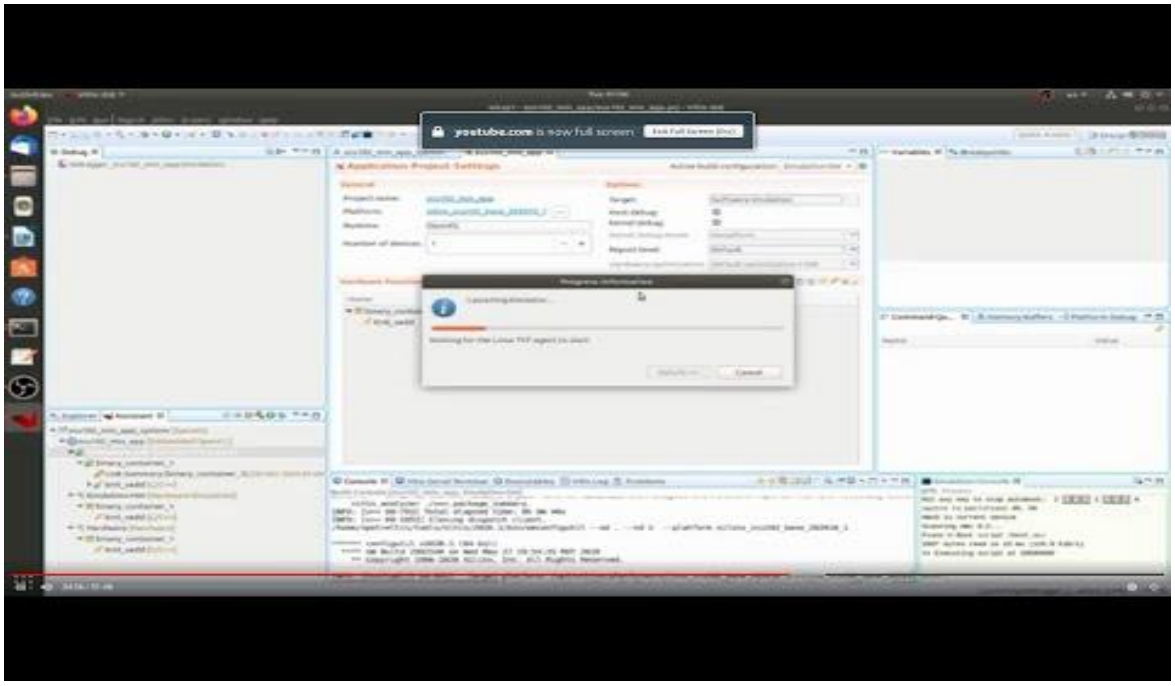
https://github.com/Xilinx/XRT

And installed as per the instructions at the wiki page:

https://xilinx.github.io/XRT/master/html/build.html

A good set of tutorials about vitis accelerated flow are on youtube at:

https://www.youtube.com/watch?v=6-Yox8fcUxc&list=PLXUuQj2gQ4s8HjchuPAXtkTg37cYvIKKp



How to create and debug Linux applications in VITIS

To create and debug Linux applications in Vitis follow the tutorial below:

https://support.xilinx.com/s/article/1141772?language=en_US

And the video:

https://www.xilinx.com/video/software/building-linux-application-vitis.html

In order to develop, libraries are needed (e.g. Vitis will complain about missing /usr/lib folder).

So the following tutorial can be followed:

https://support.xilinx.com/s/article/1175461?language=en_US

In order to get something meaningful, libraries must be added to the Petalinux image, to accomplish this, follow the tutorial below:

https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18842346/Linux+Applications+with+Petalinux+Libraries

Also worth considering building an SDK after having generated a Linux image:

https://docs.xilinx.com/r/en-US/ug1144-petalinux-tools-reference-guide/SDK-Generation-Target-Sysroot-Generation

Vish-list image processing and enhancement python

https://www.toptal.com/opencv/python-image-processing-in-computational-photography

Vitis and GIT

Vitis has an integrated GIT pluigin.

Instructions about how to use it are at:

https://support.xilinx.com/s/article/1173362?language=en_US

https://www.youtube.com/watch?v=_2VYeujilJk

And on the UG1553, at the git integration chapter:

https://docs.amd.com/r/en-US/ug1553-vitis-ide/Git-Integration

HOWTO to remove issues with the interrupt controller

Describes the issues of the interrupt controller hanging up the kernel during booting:

https://community.element14.com/products/devtools/avnetboardscommunity/avnetboard-forums/f/software-application-development/51813/ultra96-v2-petalinunx-boot-error-with-vivado-custom-design

How to solve /amba/axi-interrupt-ctrl issue on ultra96v2 (xilinx.com)

HOWTO build the vivado design and the project used by the BSP

It contains the instructions to get the Vivado design used in the BSP:

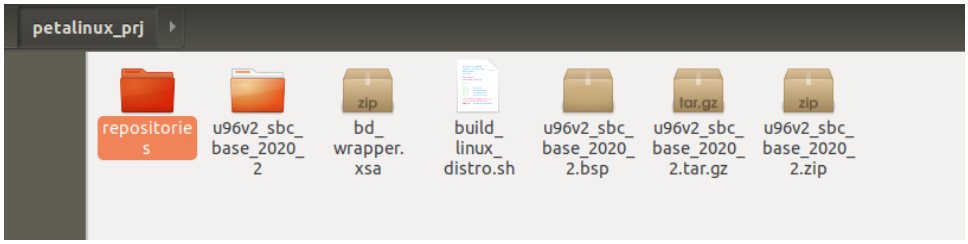
Ultra96-V2 - Building the foundational designs - Hackster.io

HOWTO accelerate the build by using cached source files

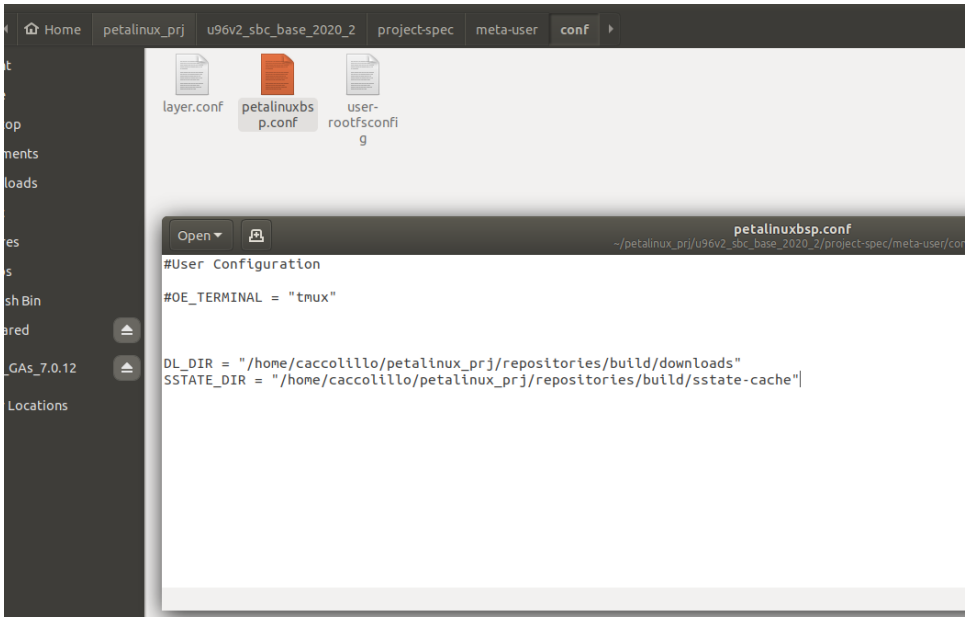
Describes how to speed up the build by using cached source files:

Accelerating PetaLinux BSP Build Time - element14 Community

In my case, I've saved and renamed an old build of petalinux:



And in the new projects, once configured and imported the xsa file, I edit the file "petalinuxbsp.conf" as follows:



HOWTO add to the device tree a custom HLS peripheral

By using the BSP, custom HW does not get added to the device tree.

To get the device tree compiler and use it, follow the guide below:

Build Device Tree Compiler (dtc) - Xilinx Wiki - Confluence (atlassian.net)

To see an example about how to integrate an HLS block in petalinux, have a look at:

Zedboard: Linux UIO Application using Vitis IDE - Hackster.io

Another example of usage of HLS in Petalinux, is at:

Integrating an HLS accelerator into Petalinux – FPGAWORLD (wordpress.com)

Yet another in-depth guide about how to get the device tree for HLS and use it, is at:

KV260 DPU-TRD Petalinux 2022.1 Vivado Flow - Hackster.io

The following guide has been followed to install the device tree compiler:

https://www.hackster.io/whitney-knitter/vitis-acceleration-flow-on-kv260-software-in-petalinux-b2456c#toc-install-device-tree-generator-4

And to create the dts file.

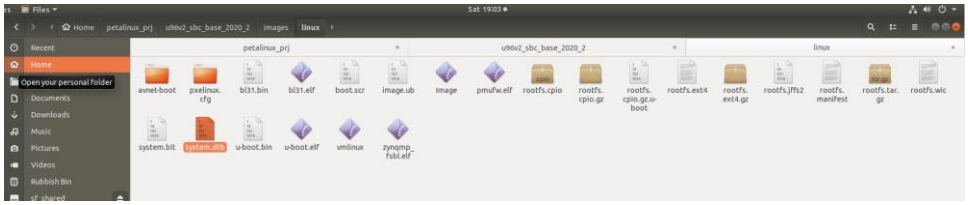
Instructions about how to add a node in the device tree, are at:

https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18842482/Device+Tree+Tips?showComments=true&showCommentArea=true#DeviceTreeTips-9Howtoaddordeletenodesandpropertiesin PetaLinux

At the following link, instructions about how to debug device tree:

[PetaLinux Image Debug Series: Debugging the Device Tree Generator (xilinx.com)]

At the end of the build, the dtb (device tree blob) file is in the folder below:



And it can be decompiled as per the following guide:

[Decompiling the Device Tree (DTB) for Embedded Linux – Silicon Blade Consultants]

```
caccolillo@caccolillo-VirtualBox:~/petalinux_prj/u96v2_sbc_base_2020_2$ pwd
/home/caccolillo/petalinux_prj/u96v2_sbc_base_2020_2
caccolillo@caccolillo-VirtualBox:~/petalinux_prj/u96v2_sbc_base_2020_2$ dtc -I dtb system.dtb -o pippo.dts
pippo.dts: Warning (ranges_format): /zynqmp_ipi1 has empty "ranges" property but its #address-cells (1) differs from / (2)
pippo.dts: Warning (ranges_format): /zynqmp_ipi1 has empty "ranges" property but its #size-cells (1) differs from / (2)
pippo.dts: Warning (unit_address_vs_reg): Node /amba/axi-interrupt-ctrl has a reg or ranges property, but no unit name
pippo.dts: Warning (unit_address_vs_reg): Node /amba_pl@0 has a unit name, but no reg property
pippo.dts: Warning (unit_address_vs_reg): Node /memory has a reg or ranges property, but no unit name
pippo.dts: Warning (unit_address_vs_reg): Node /zynqmp-rpu has a reg or ranges property, but no unit name
pippo.dts: Warning (unit_address_vs_reg): Node /zynqmp-rpu/r5@0 has a unit name, but no reg property
pippo.dts: Warning (unit_address_format): Node /amba/spi@ff040000/spidev@0x00 unit name should not have leading "0x"
pippo.dts: Warning (unit_address_format): Node /amba/spi@ff040000/spidev@0x00 unit name should not have leading 0s
pippo.dts: Warning (unit_address_format): Node /amba/spi@ff050000/spidev@0x00 unit name should not have leading "0x"
pippo.dts: Warning (unit_address_format): Node /amba/spi@ff050000/spidev@0x00 unit name should not have leading 0s
pippo.dts: Warning (unit_address_format): Node /krnl_img_averaging@0xa00a0000 unit name should not have leading "0x"
pippo.dts: Warning (pci_device_reg): Node /amba/pcie@fd0e0000/legacy-interrupt-controller missing PCI reg property
pippo.dts: Warning (simple_bus_reg): Node /amba/axi-interrupt-ctrl simple-bus unit address format error, expected "a0090000"
pippo.dts: Warning (simple_bus_reg): Node /amba/zyxclmm_drm missing or empty reg/ranges property
pippo.dts: Warning (gpios_property): Could not get phandle node for /__symbols__:gpio(cell 0)
caccolillo@caccolillo-VirtualBox:~/petalinux_prj/u96v2_sbc_base_2020_2$ aver
```

The system-user.dtsi file used to create the dts file is:

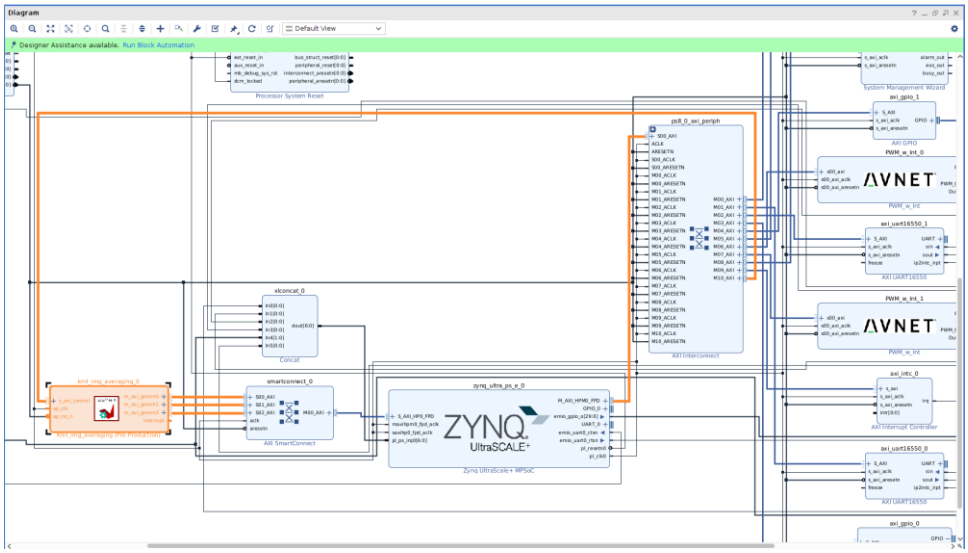
```
/include/ "system-conf.dtsi"
/ {
    amba_pl: amba_pl@0 {
            krnl_img_averaging@0xa00a0000 {
            /* This is a place holder node for a custom IP, user may need to update the entries */
            clock-names = "s_axi_aclk";
            clocks = <&zynqmp_clk 71>;
            compatible = "xlnx,krnl-img-averaging-1.0";
            reg = <0x0 0xa00a0000 0x0 0x10000>;
            xlnx,s-axi-control-addr-width = <0x7>;
            xlnx,s-axi-control-data-width = <0x20>;
            };
    };
};
```

Open ▾ system-user.dtsi [Read-Only]
~/petalinux_prj/u96v2_sbc_base_2020_2/pro...ec/meta-user/recipes-bsp/device-tree/files

```
/include/ "system-conf.dtsi"
/ {
    amba_pl: amba_pl@0 {
        krnl_img_averaging@0xa00a0000 {
        /* This is a place holder node for a custom IP, user may need to update the entries */
        clock-names = "s_axi_aclk";
        clocks = <&zynqmp_clk 71>;
        compatible = "xlnx,krnl-img-averaging-1.0";
        reg = <0x0 0xa00a0000 0x0 0x10000>;
        xlnx,s-axi-control-addr-width = <0x7>;
        xlnx,s-axi-control-data-width = <0x20>;
        };
    };
};
```

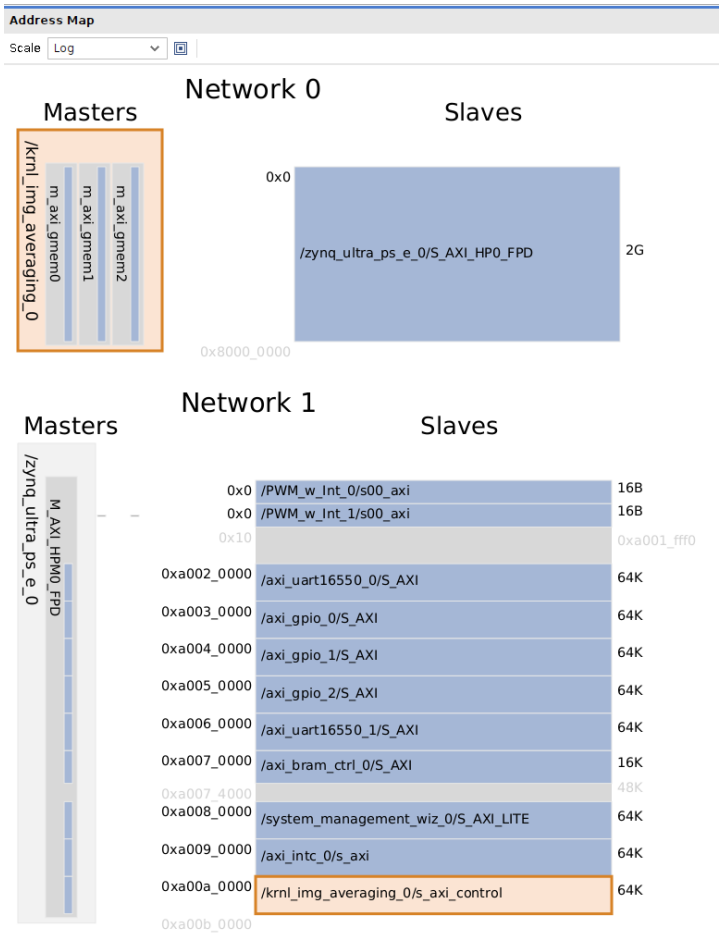
C development under LINUX for a custom HLS peripheral

The image averaging HLS IP core has been integrated and connected in Vivado:



It has three AXI master ports used to get the two source images and write back the result one to the DDR memory, and an AXI slave port for management purposes.

The address map for the peripheral follows:



The AXI registers for the peripherals are mapped starting at 0xa00a0000.

By looking at the automatically generated device drivers in HLS, we get the register set internal details:

Open — xkrnl_img_averaging_hw.h
~/Documents/Cat_Doorbell/vitis_hls/im...sc/drivers/krnl_img_averaging_v1_0/src

```
// ==========================================================
// Vitis HLS - High-Level Synthesis from C, C++ and OpenCL v2022.2 (64-bit)
// Tool Version Limit: 2019.12
// Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.
// ==========================================================
// control
// 0x00 : Control signals
//        bit 0  - ap_start (Read/Write/COH)
//        bit 1  - ap_done (Read)
//        bit 2  - ap_idle (Read)
//        bit 3  - ap_ready (Read/COR)
//        bit 4  - ap_continue (Read/Write/SC)
//        bit 7  - auto_restart (Read/Write)
//        bit 9  - interrupt (Read)
//        others - reserved
// 0x04 : Global Interrupt Enable Register
//        bit 0  - Global Interrupt Enable (Read/Write)
//        others - reserved
// 0x08 : IP Interrupt Enable Register (Read/Write)
//        bit 0 - enable ap_done interrupt (Read/Write)
//        bit 1 - enable ap_ready interrupt (Read/Write)
//        others - reserved
// 0x0c : IP Interrupt Status Register (Read/TOW)
//        bit 0 - ap_done (Read/TOW)
//        bit 1 - ap_ready (Read/TOW)
//        others - reserved
// 0x10 : Data signal of in1
//        bit 31~0 - in1[31:0] (Read/Write)
// 0x14 : Data signal of in1
//        bit 31~0 - in1[63:32] (Read/Write)
// 0x18 : reserved
// 0x1c : Data signal of in2
//        bit 31~0 - in2[31:0] (Read/Write)
// 0x20 : Data signal of in2
//        bit 31~0 - in2[63:32] (Read/Write)
// 0x24 : reserved
// 0x28 : Data signal of out_r
//        bit 31~0 - out_r[31:0] (Read/Write)
// 0x2c : Data signal of out_r
//        bit 31~0 - out_r[63:32] (Read/Write)
// 0x30 : reserved
// 0x34 : Data signal of scale_by_4
//        bit 31~0 - scale_by_4[31:0] (Read/Write)
// 0x38 : reserved
// 0x3c : Data signal of size
//        bit 31~0 - size[31:0] (Read/Write)
// 0x40 : reserved
// (SC = Self Clear, COR = Clear on Read, TOW = Toggle on Write, COH = Clear on Handshake)

```

A petalinux distribution gets generated for this HW release integrating the image averaging IP, and a quick test with devmem is done, to ascertain that the internal registers can be accessed:

```
File  Edit  Log  Configuration  Control signals  View  Help
root@u96v2-sbc-base-2020-2:~# devmem 0xa00a0000
0x00000081
root@u96v2-sbc-base-2020-2:~# devmem 0xa00a0000 32 0x00000000
root@u96v2-sbc-base-2020-2:~# devmem 0xa00a0000
0x0000000E
root@u96v2-sbc-base-2020-2:~# devmem 0xa00a0000 32 0x00000004
root@u96v2-sbc-base-2020-2:~# devmem 0xa00a0000
0x00000006
root@u96v2-sbc-base-2020-2:~#
```

```
root@u96v2-sbc-base-2020-2:~# devmem 0xa00a003c
0x00000000
root@u96v2-sbc-base-2020-2:~# devmem 0xa00a003c 32 0x000f0000
root@u96v2-sbc-base-2020-2:~# devmem 0xa00a003c
0x000F0000
root@u96v2-sbc-base-2020-2:~#
```

It proves the device is where it's meant to be and can be accessed and managed by petalinux.

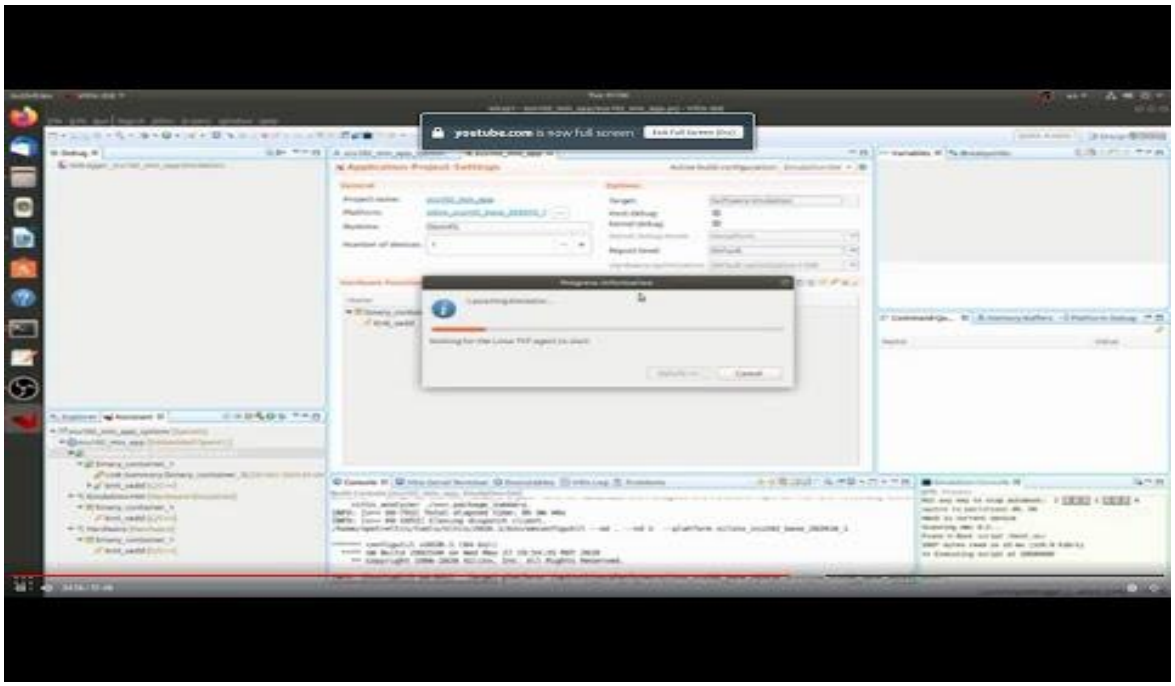
Now a C application under Linux can be developed to manage the IP core.

The guidance at:

https://www.hackster.io/whitney-knitter/debugging-your-custom-linux-applications-using-vitis-67c022

https://xilinx.github.io/Vitis-Tutorials/master/docs-jp/docs/Vitis_Platform_Creation/Design_Tutorials/03_Edge_VCK190/step4.html

https://www.youtube.com/watch?v=6-Yox8fcUxc&list=PLXUuQj2gQ4s8HjchuPAXtkTg37cYvIKKp



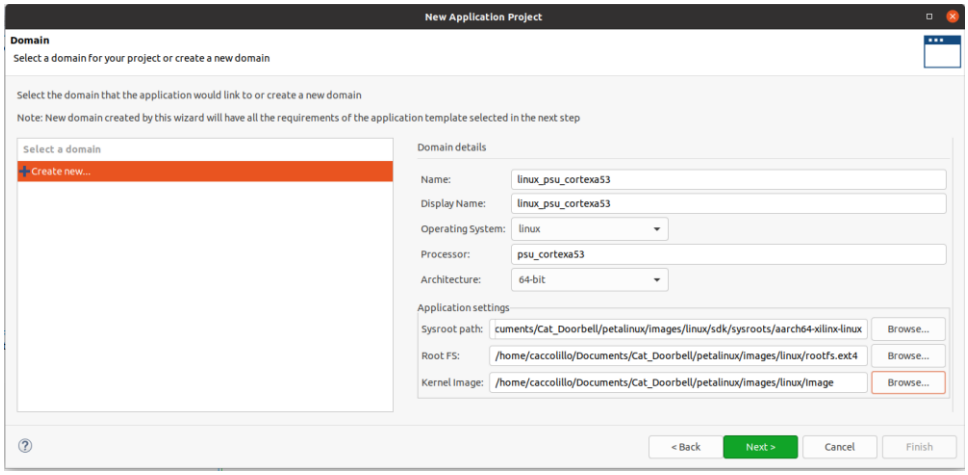
Shall be followed. It shows how to write and test on a live Linux system a C application.

Once confident about it, a petalinux recipe for permanent deployment can be found at:

https://www.hackster.io/whitney-knitter/custom-application-creation-in-petalinux-on-the-zynqberry-15ff77

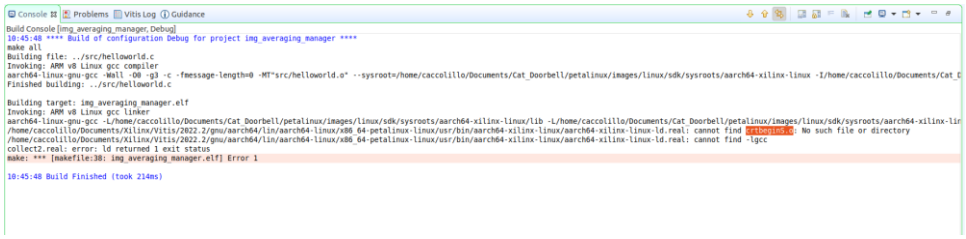
A new application project gets created, and the xsa file generated in Vivado gets used.

The SMP option has to be used, and the sysroot, root fs and the image gets specified as follow:



Where sysroot is generated as sdk in petalinux.

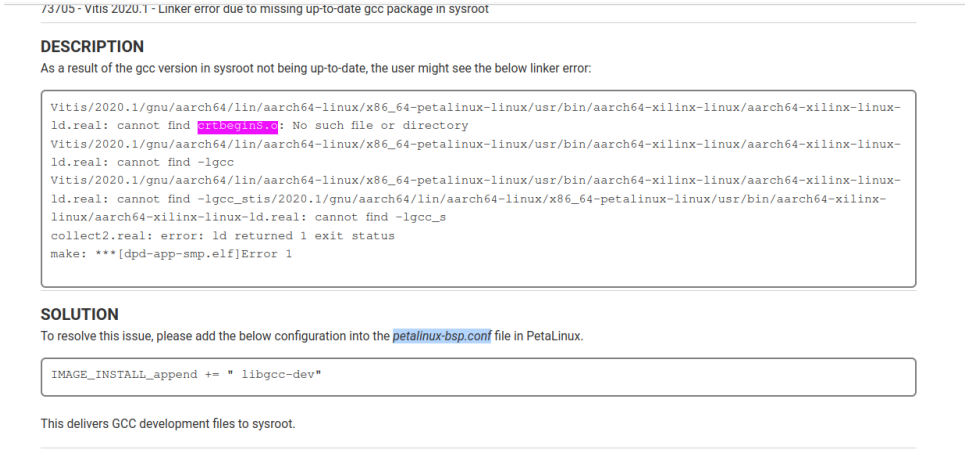
A trivial helloworld application has been chosen as starting point, and when built, gives the following error:



By looking online:

https://support.xilinx.com/s/article/73705?language=en_US

It comes out that petalinux has to be restarted and petalinux-bsp.conf changed as follows:



Despite this modification, Vitis still complaints about the presence of the same error.

By looking online, the following posts have been looked at:

https://support.xilinx.com/s/question/0D52E00006hpLY6SAM/petalinux-build-in-libgcc?language=en_US

https://support.xilinx.com/s/question/0D52E00006hpl3YSAQ/petalinux-20183-missing-libgccsso1-while-using-pthread?language=en_US

https://support.xilinx.com/s/question/0D52E00006rAMJCSA4/libgccsso1-not-installed-even-though-libgcc-selected-in-rootfs-configuration?language=en_US

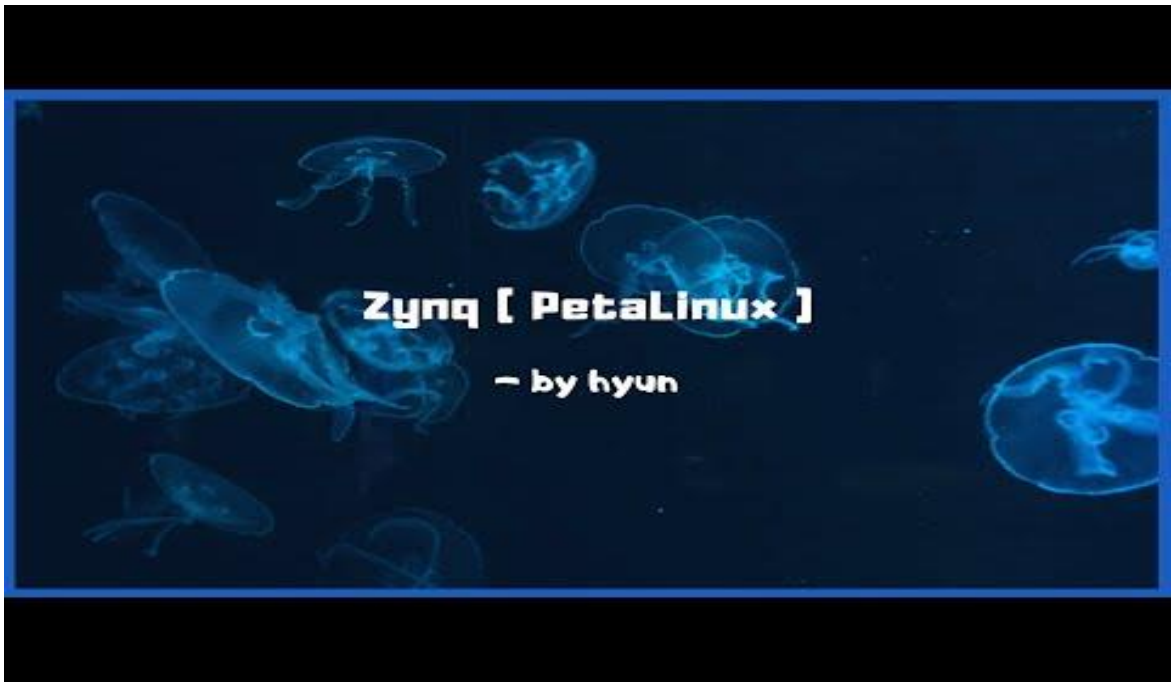
https://www.hackster.io/fabioc9675/empowering-dune-creating-petalinux-2022-2-os-image-for-kria-2fbca1

Trying again by following this Xilinx tutorial:

Building a Linux Application in the Vitis IDE (xilinx.com)

Here is shown how to develop natively on the Ultra967 v2:

Twitch_SpringSales_UK (youtube.com)



BSP PETALINUX issues and build without a BSP

After many attempts, builds have started failing, by using the AVNET BSP for Petalinux.

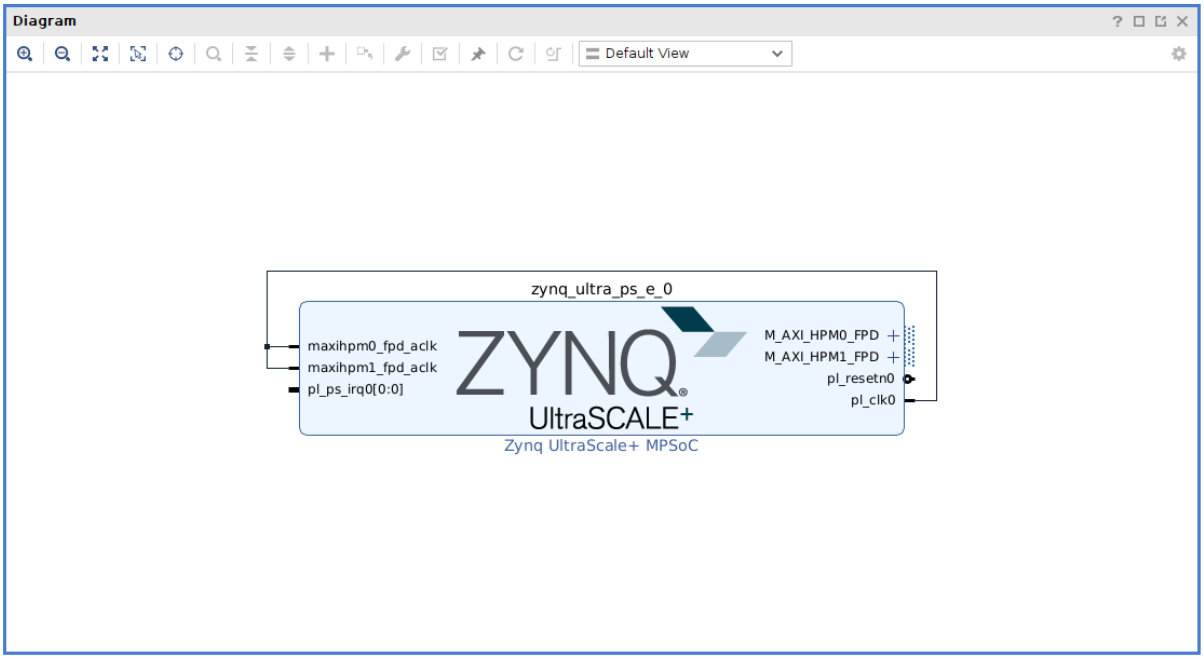
This was due mainly to issues with the Microchip WiFi driver.

In light of this, the choice to go BSP-less has been taken.

The method of using a template for the Zynq Ultrascale+ described in the blog below has been used:

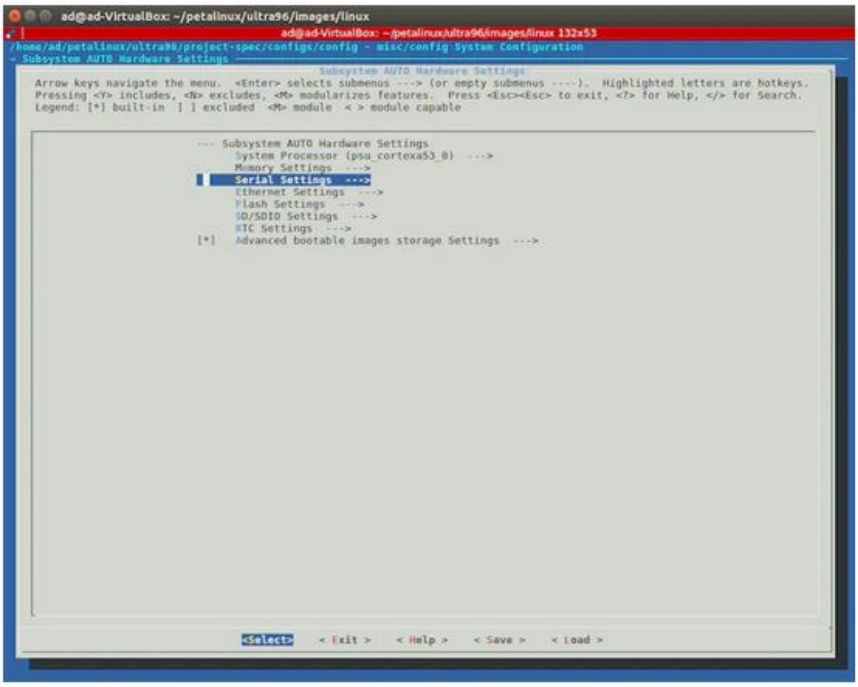
https://www.hackster.io/adam-taylor/two-methods-of-building-petalinux-for-the-ultra96-77c8e0

And the Vivado project has been reverted back, with a block design which is hosting only a Zynq ultrascale+ with default settings:

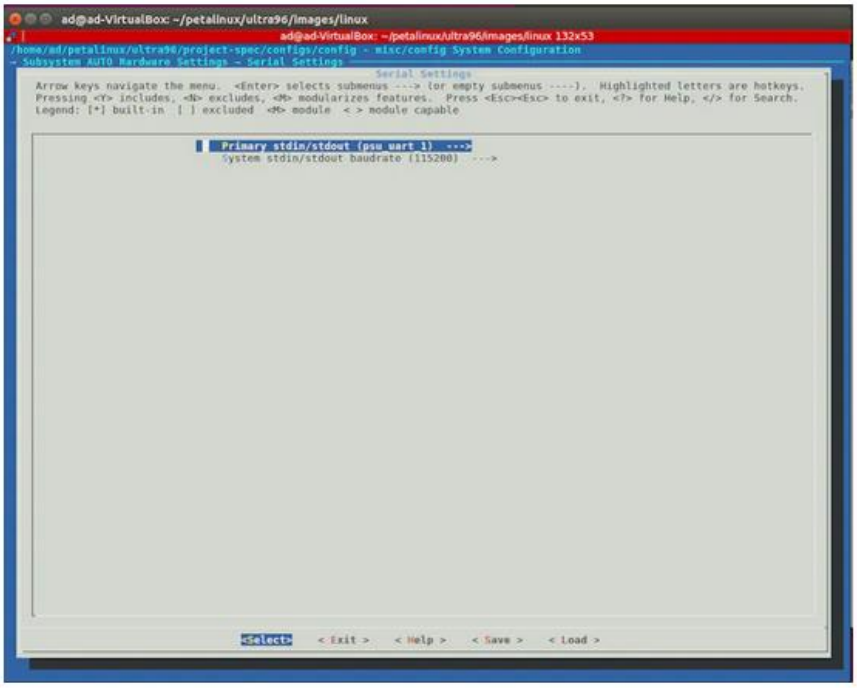


No fan control is in place: all is relying upon the usage of some sort of pull-up resistor on the PCB, as the fan is spinning at full speed anyway.

Once built the XSA, the zynq ultrascale + template has been used as is, with the exception of changing the stdio to the following way:

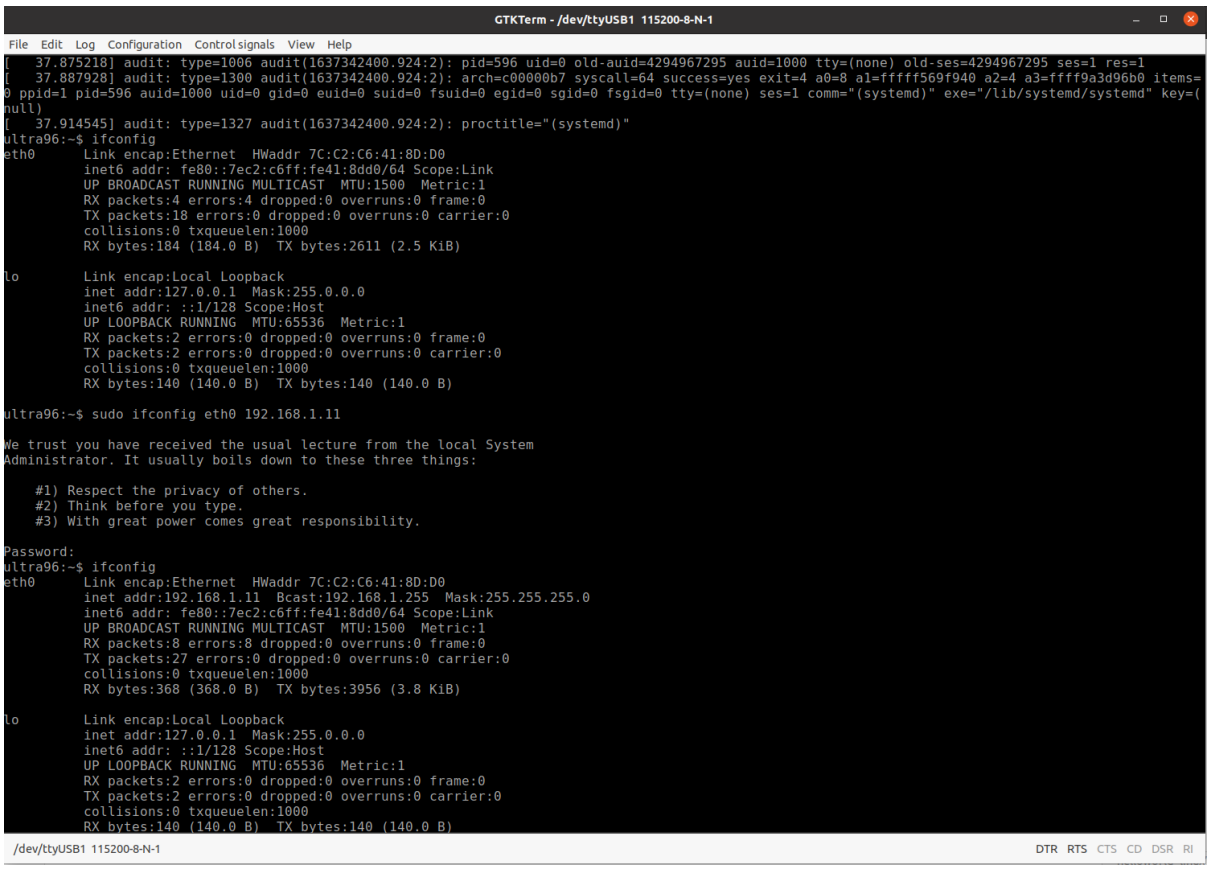


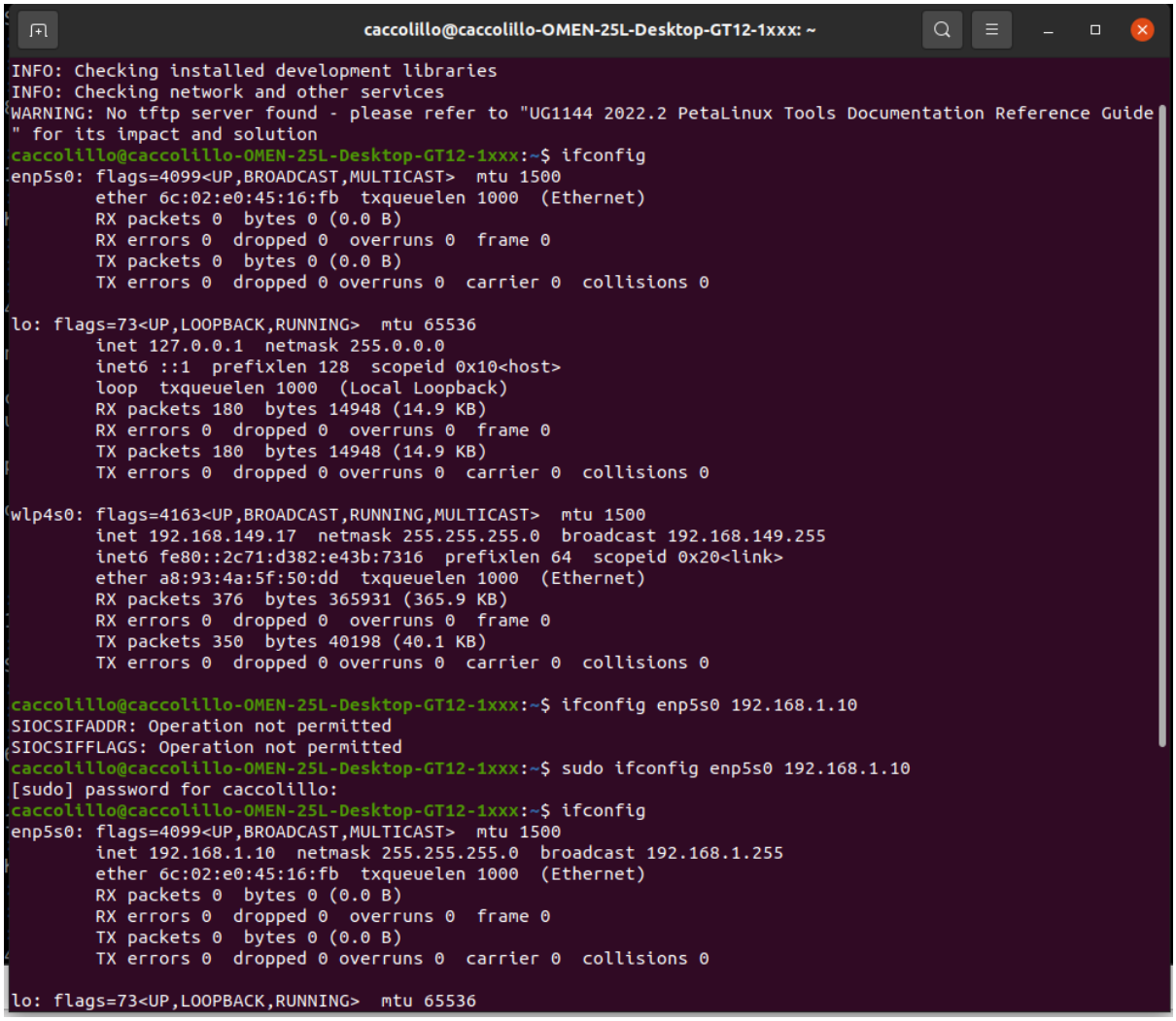
Selecting Serial interfaces



Selecting UART 1

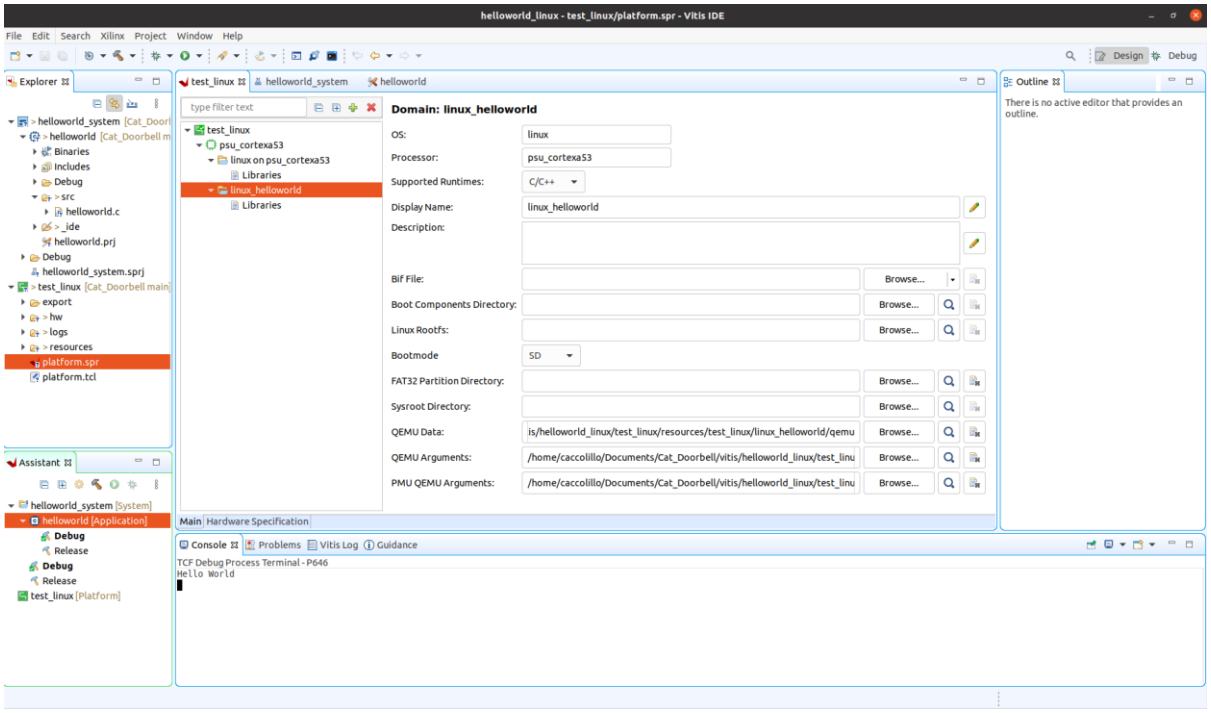
Plugging the ethernet cable between the PC and the USB to ethernet dongle on the ULTRA96 v2, static IP addresses get set:

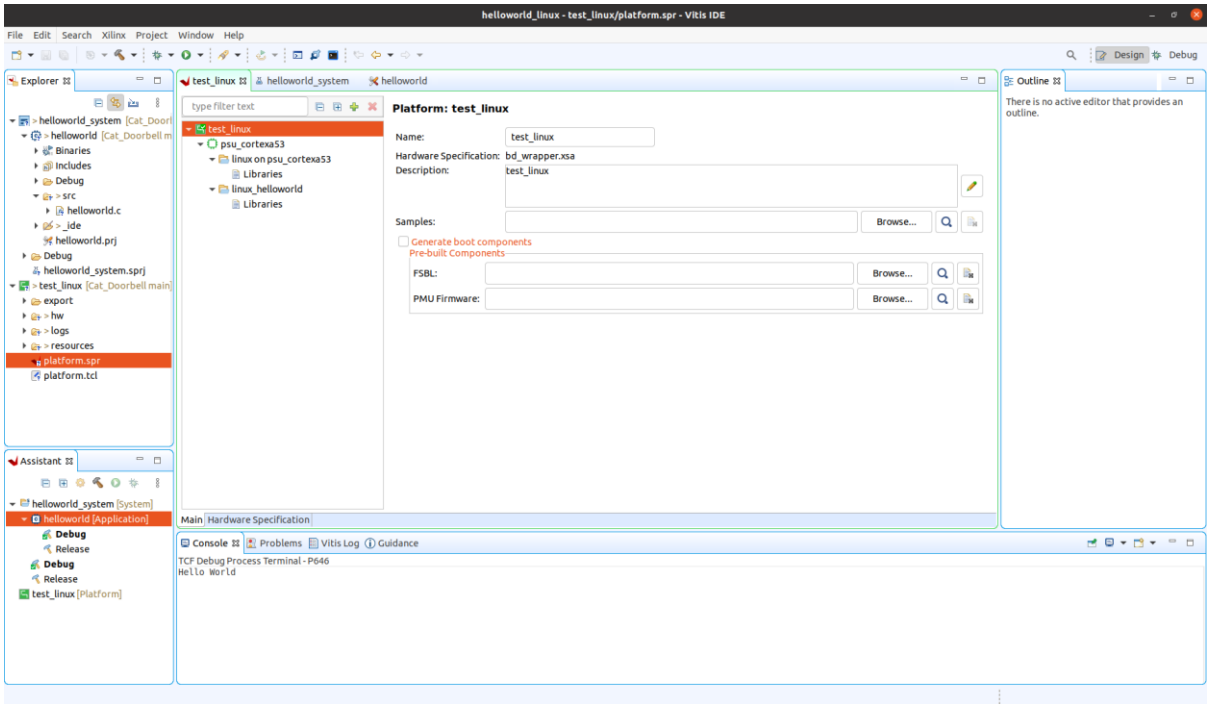




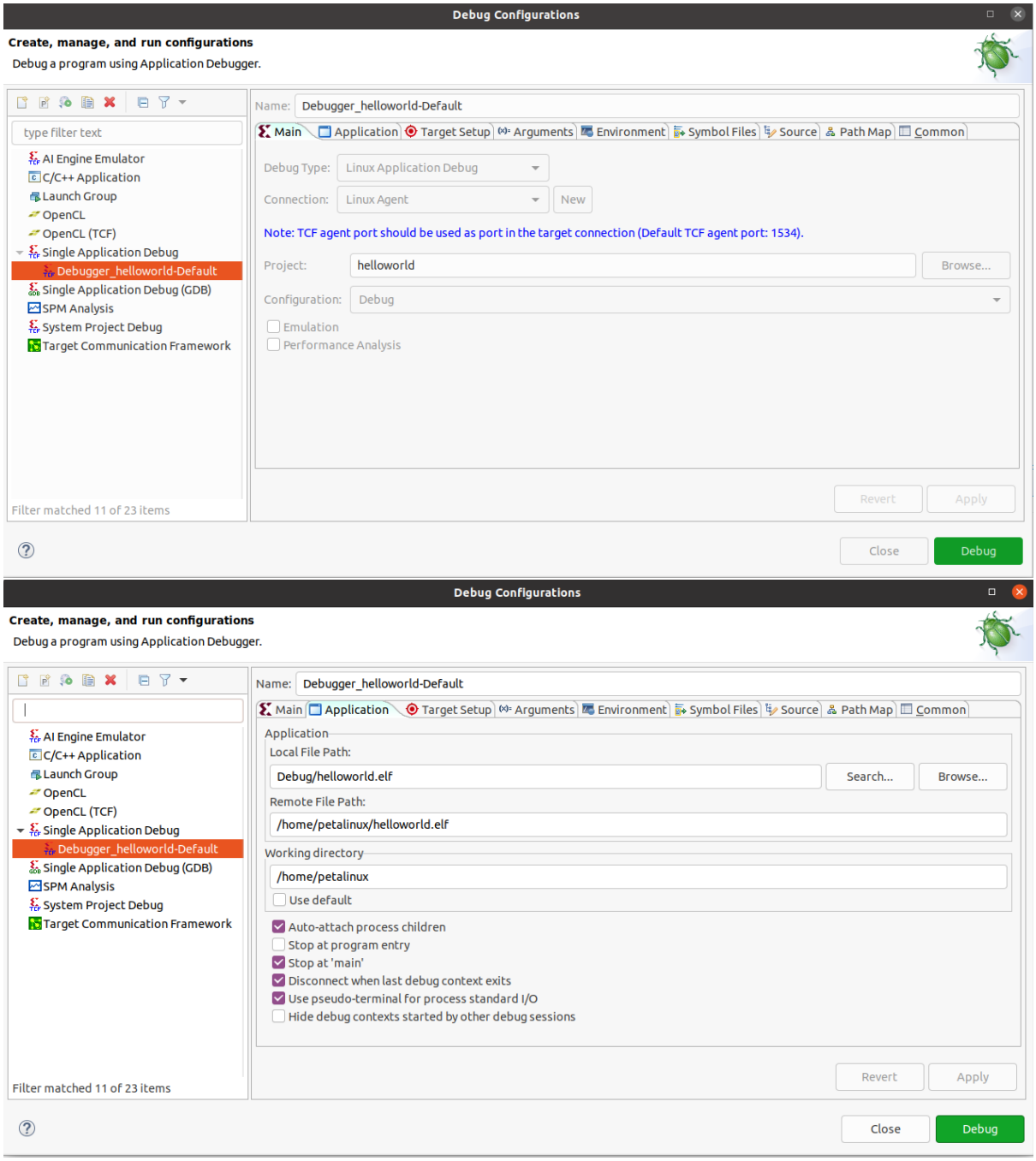
The vitis environment gets set as per the following tutorial to debug via a TCF connection on the ULTRA96 V2 board:

https://xilinx.github.io/Embedded-Design-Tutorials/docs/2021.2/build/html/docs/Introduction/Zynq7000-EDT/4-linux-for-zynq.html

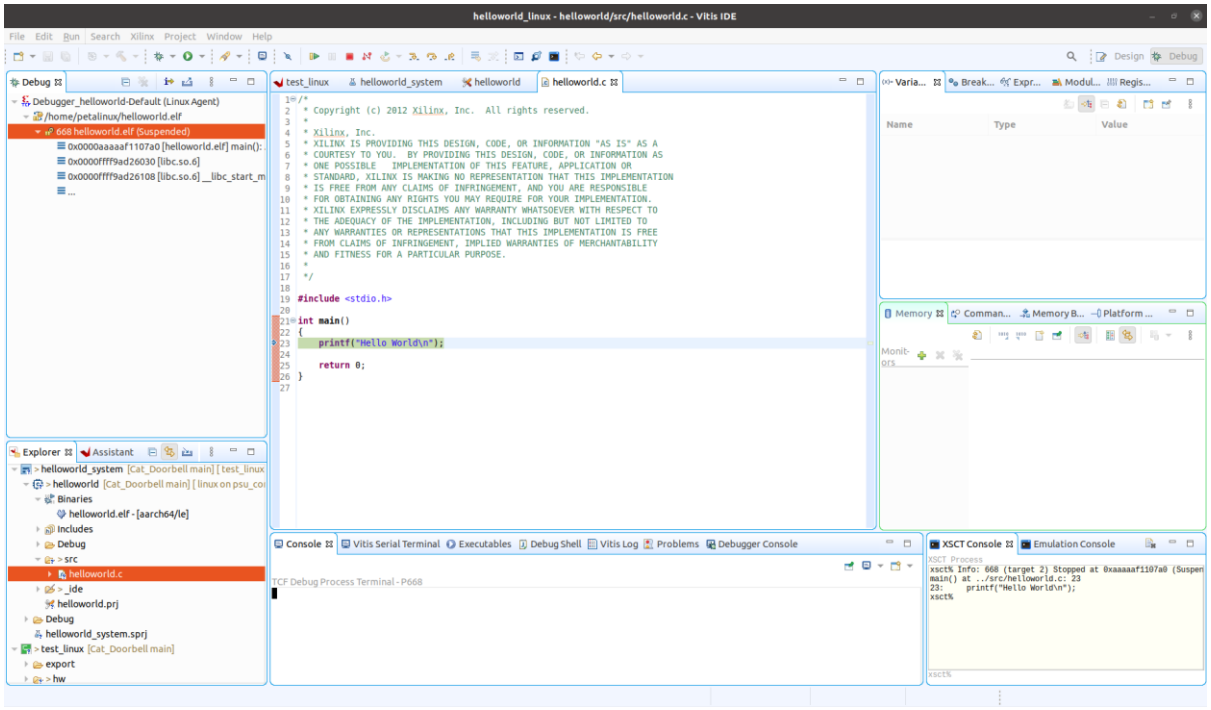


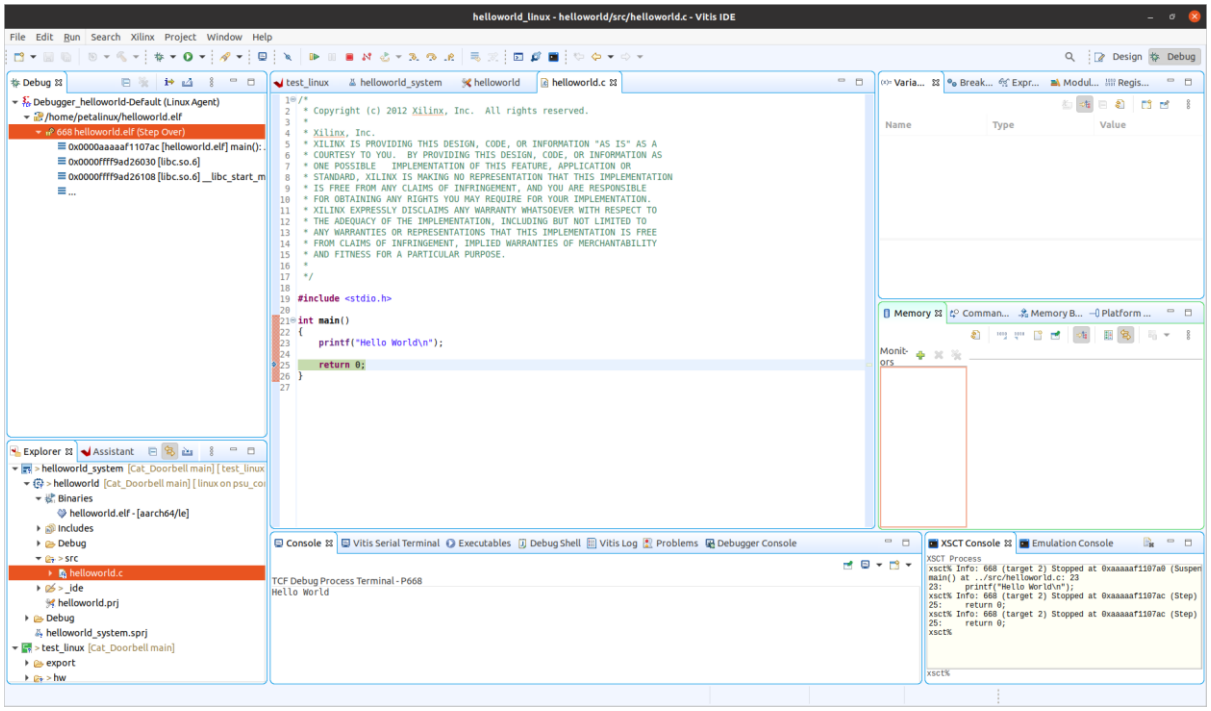


The debugger has been configured as follows:

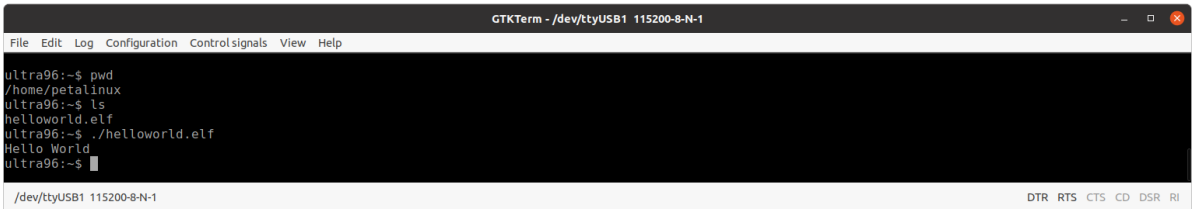


once launched the debug process, the ”hello world” gets written in the console window:





By looking at the Ultra96 v2 shell, the elf file gets downloaded in the folder of choice set up in the debug configuration, and can also be executed locally:

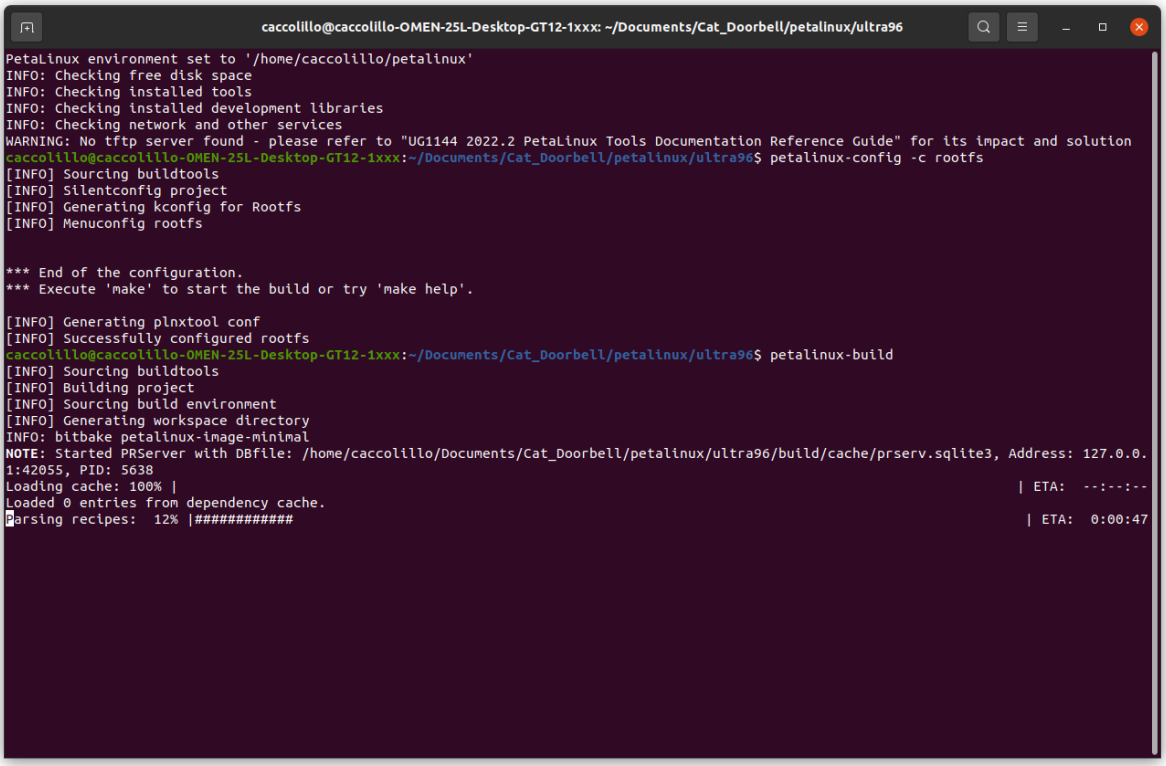


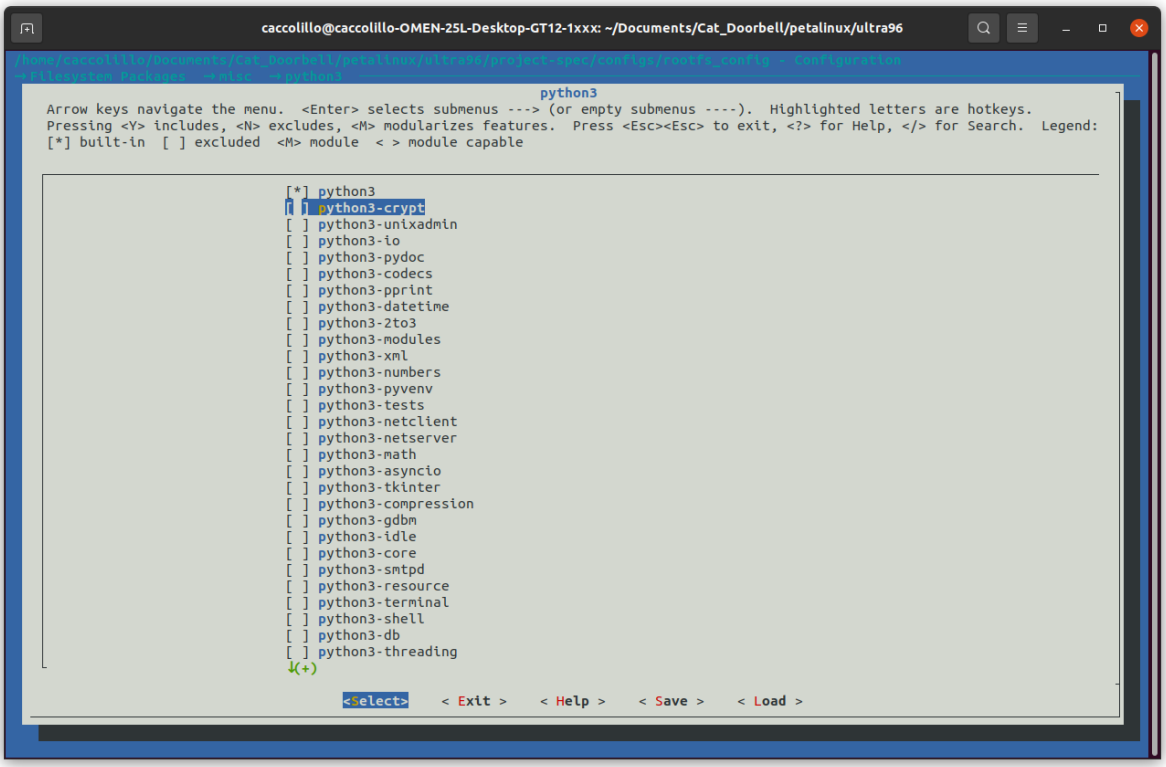
Python 3 interpreter is not installed as a default on the petalinux build from the template, so it has to be added.

To do so follow:

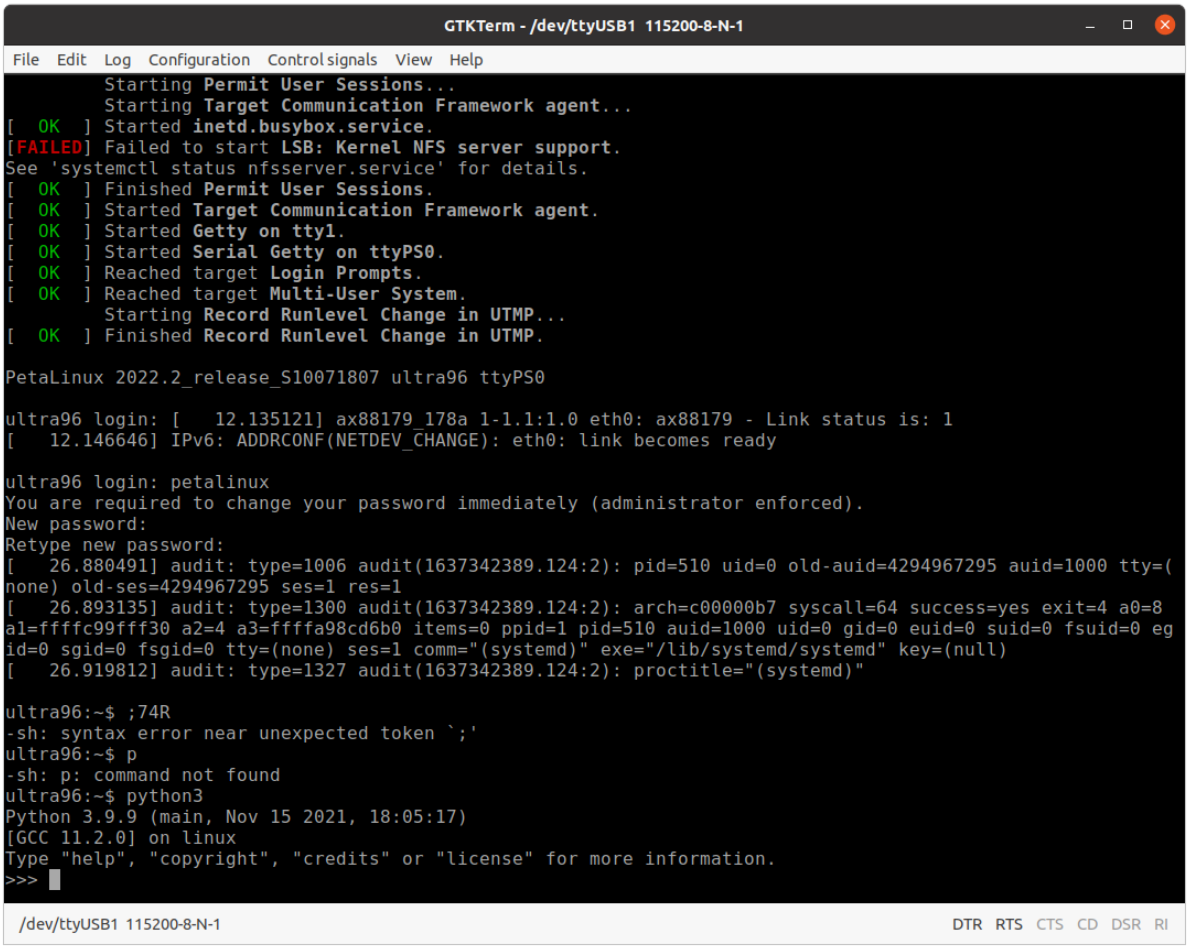
https://support.xilinx.com/s/question/0D52E00006ihQL7SAM/petalinux-and-python3?language=en_US

So it gets enabled as per pictures below:





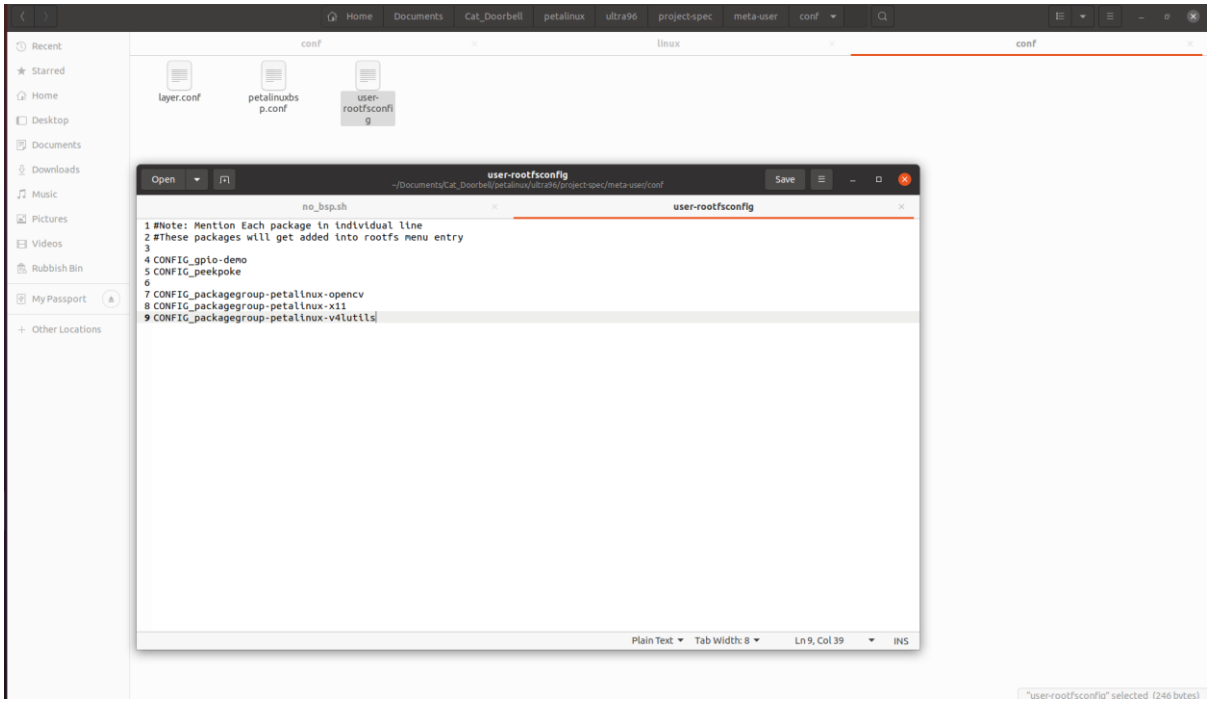
And on the ULTRA96 v2 console, the python interpreter can be started:



To enable the webcam, the tutorial below has been followed:

https://www.hackster.io/whitney-knitter/using-a-usb-web-camera-with-the-minized-5783b1

In doing so, the user-rootfsconfig has been edited as follows:

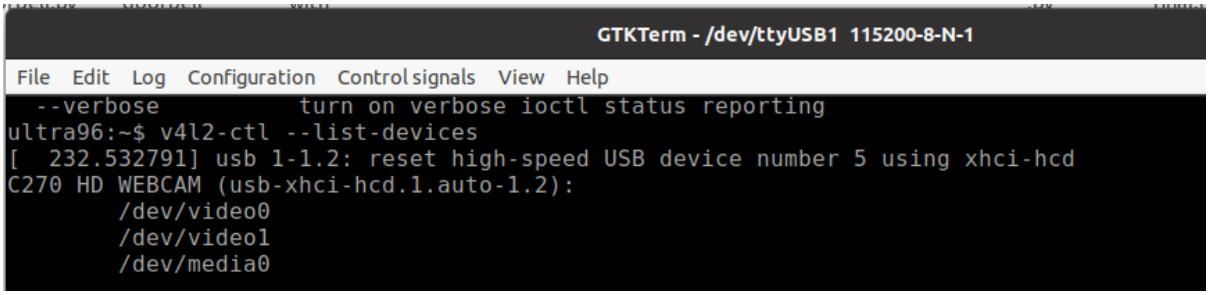


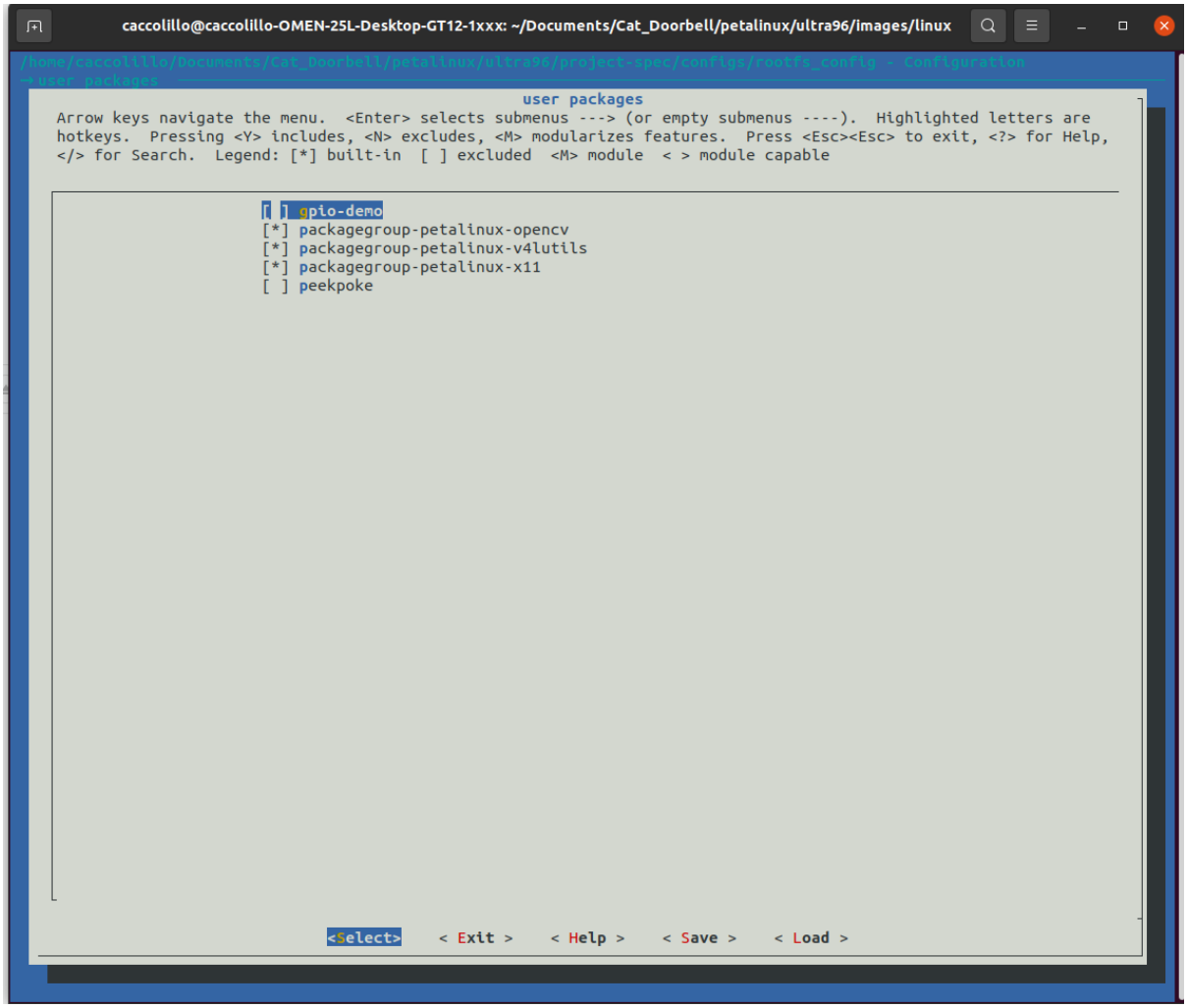
Then launch the root filesystem configuration editor GUI with “petalinux-config -c rootfs”, and then enable the:

- packagegroup-petalinux,
- packagegroup-petalinux-display-debug,
- packagegroup-petalinux-opencv,
- packagegroup-petalinux-self-hosted,
- packagegroup-petalinux-utils,
- packagegroup-petalinux-v4lutils,
- packagegroup-petalinux-x11.

Then navigate to the user packages tab to enable all of the custom options that were added to the *user-rootfsconfig* file.

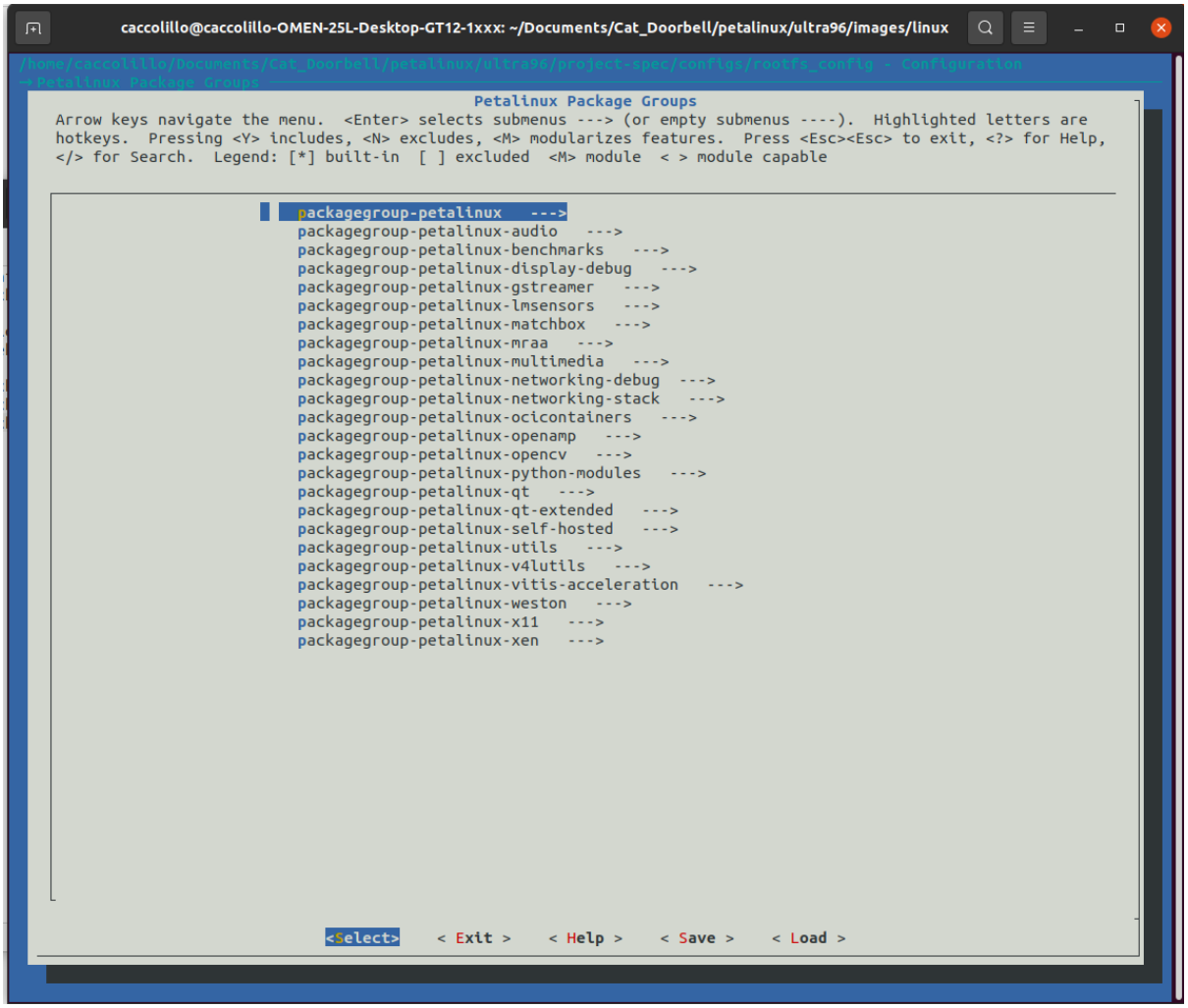
Compiling the image and running the board, by issuing the following command, the USB webcam gets recognized correctly:



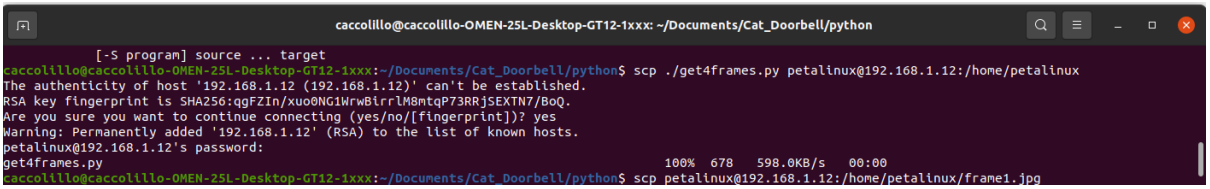


Save and exit the root filesystem configuration editor then build the PetaLinux project:

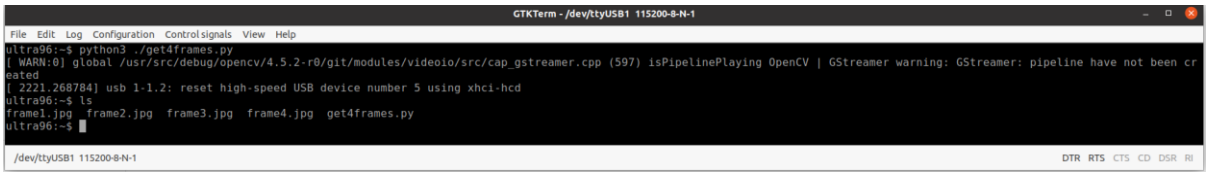
petalinux-build



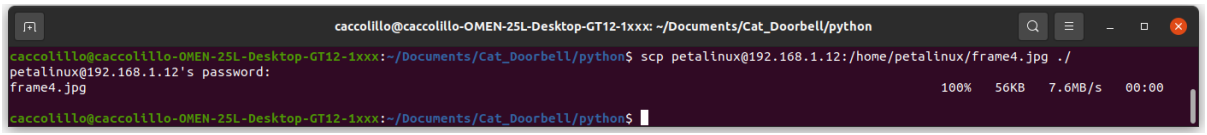
The board gets programmed with the new image, the ETHERNET static addresses on the ULTRA 96 and the HOST pc get set, via scp the python script gets transferred to the ULTRA96:



And then it gets executed on the board:



After the execution, the four frames files are created, as expected, and they can be collected always via scp, and brought in the host pc, with the command:



Now it's time to create the Vitis "driver" C application to manage the image averaging kernel HLS IP-core.

A source of inspiration comes from:

https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18842018/Linux+User+Mode+Pseudo+Driver

In order to get usable memory for the hardwarte accelerator, the technique of using the CMA memory shall be used:

https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18841683/Linux+Reserved+Memory

The system-user.dtsi has been hence changed as per below:

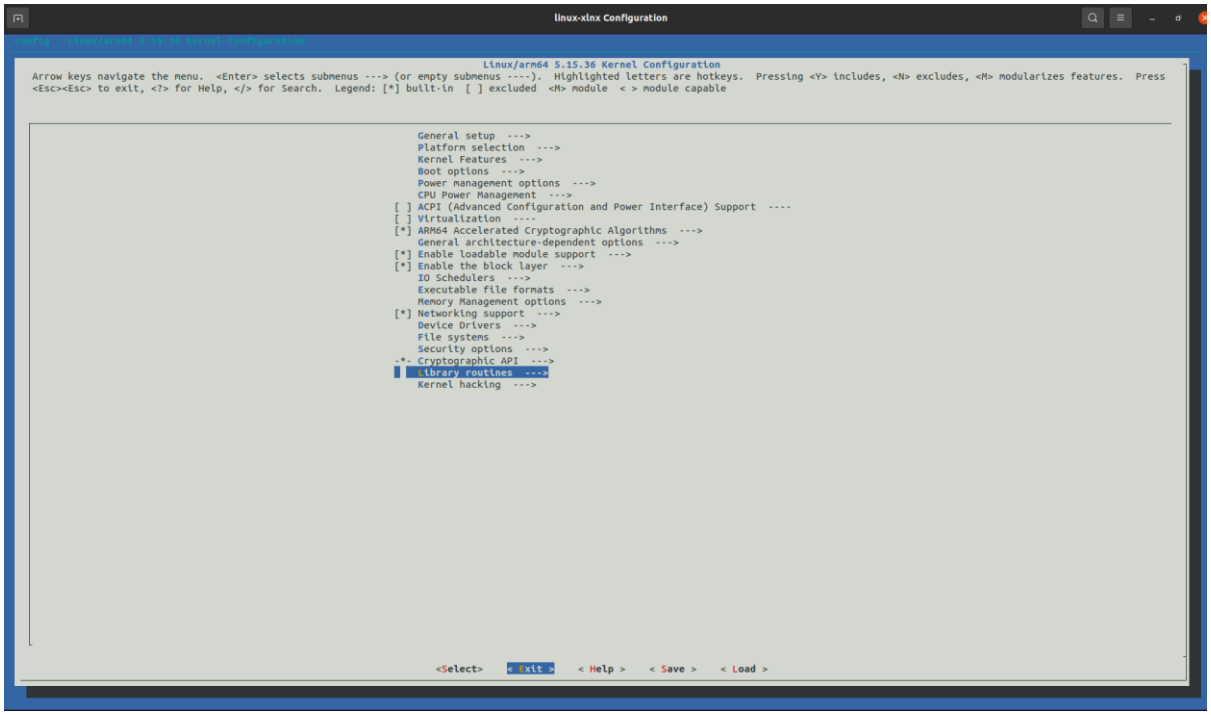
```
/include/ "system-conf.dtsi"
/ {
reserved-memory {
    #address-cells = <2>;
    #size-cells = <2>;
    ranges;
    reserved: buffer@0 {
      compatible = "shared-dma-pool";
      reusable;
      reg = <0x0 0x70000000 0x0 0x10000000>;
      linux,cma-default;
    };
  };
};
```

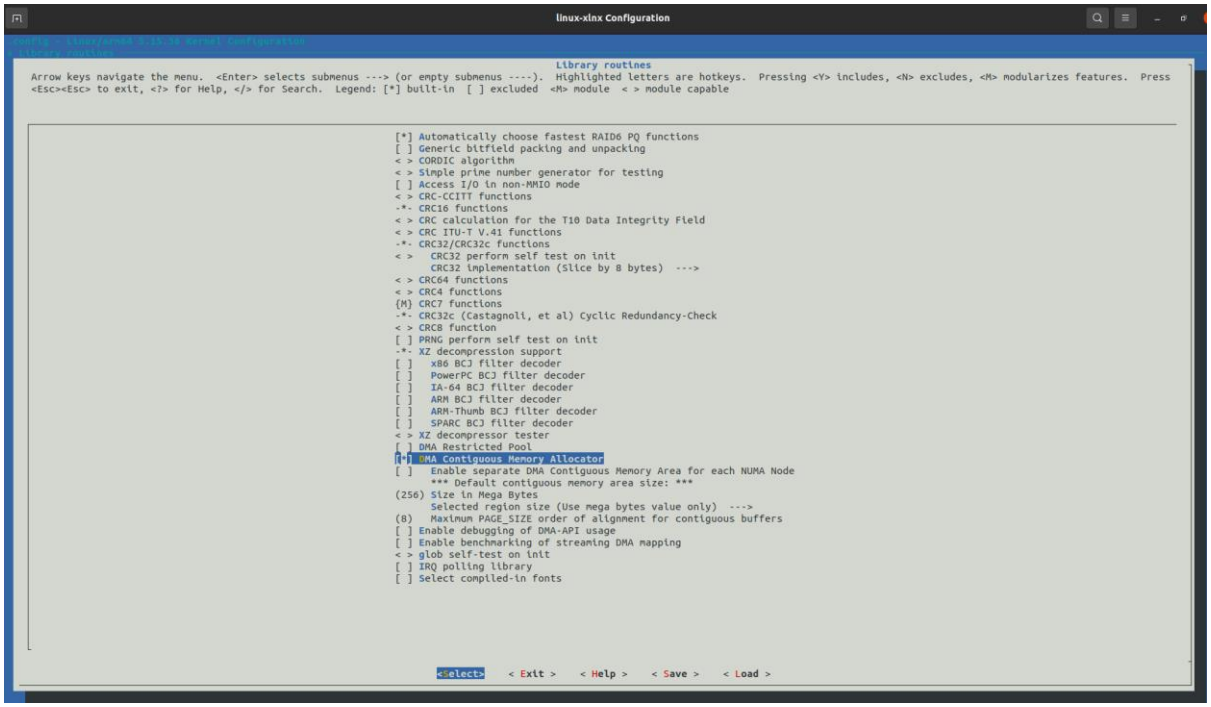
Open | system-user.dtsi | ~/Documents/Cat_Doorbell/petalinux/ultra9...ec/meta-user/recipes-bsp/device-tree/files | Save

```
/include/ "system-conf.dtsi"
/ {

reserved-memory {
      #address-cells = <2>;
      #size-cells = <2>;
      ranges;

      reserved: buffer@0 {
         compatible = "shared-dma-pool";
         reusable;
         reg = <0x0 0x70000000 0x0 0x10000000>;
         linux,cma-default;
      };
   };


};
```

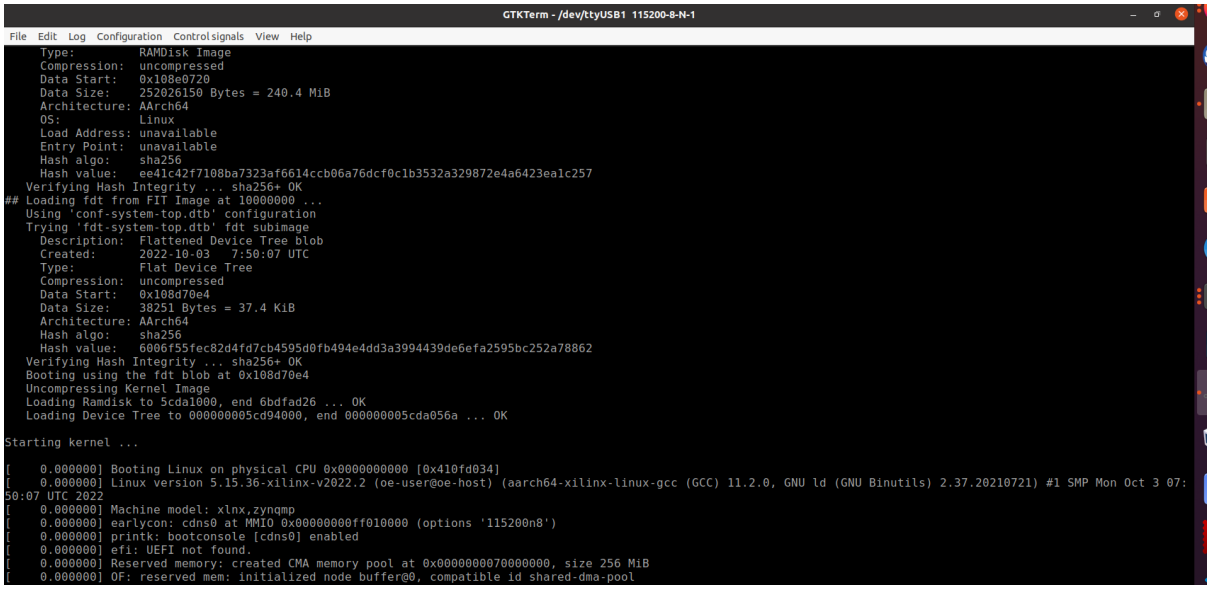
And the CMA has been enabled in the kernel:

petalinux-config -c kernel





After building petalinux and flashing the image, the system boots up with a correct CMA region:



VITIS VISION LIBRARIES L1

Instructions about how to make work OpenCV under Vitis to be used with Vision libraries are at:

https://support.xilinx.com/s/article/Vitis-Libraries-Compiling-and-Installing-OpenCV?language=en_US

An example about how to get a working Vitis vision example is at:

https://support.xilinx.com/s/article/000035905?language=en_US

A tutorial about how to use the L1 in Vivado, Vitis HLS and Vitis is at:

https://support.xilinx.com/s/article/000035309?language=en_US

Several interesting tutorials are at:

https://community.element14.com/technologies/fpga-group/b/blog/posts/learning-xilinx-zynq-try-to-make-my-own-accelerated-opencv-function---1-vitis-hls

https://community.element14.com/technologies/fpga-group/b/blog/posts/learning-xilinx-zynq-hardware-accelerated-software

https://www.linkedin.com/pulse/vitis-20201-beauty-acceleration-martin-kellermann

https://www.xilinx.com/video/software/vitis-hls-l1-library-wizard.html

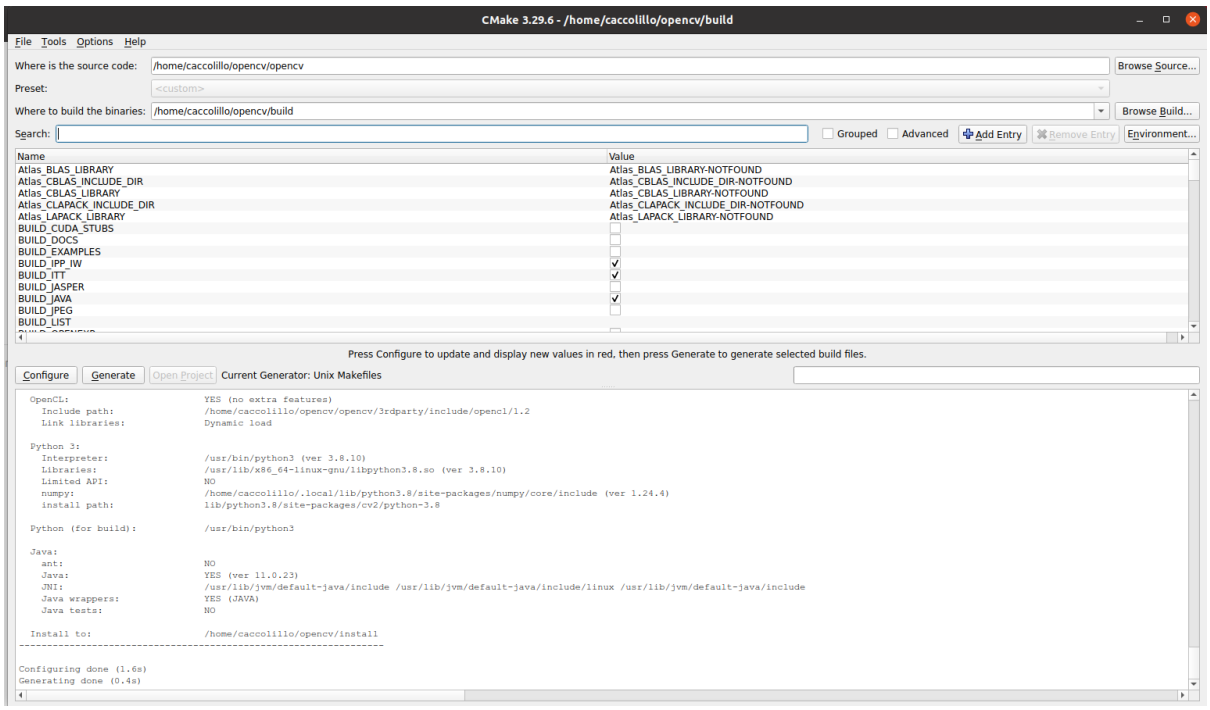
https://docs.amd.com/r/2022.2-English/Vitis-Tutorials-Getting-Started/Step-1-Create-Vivado-Design-and-Generate-XSA

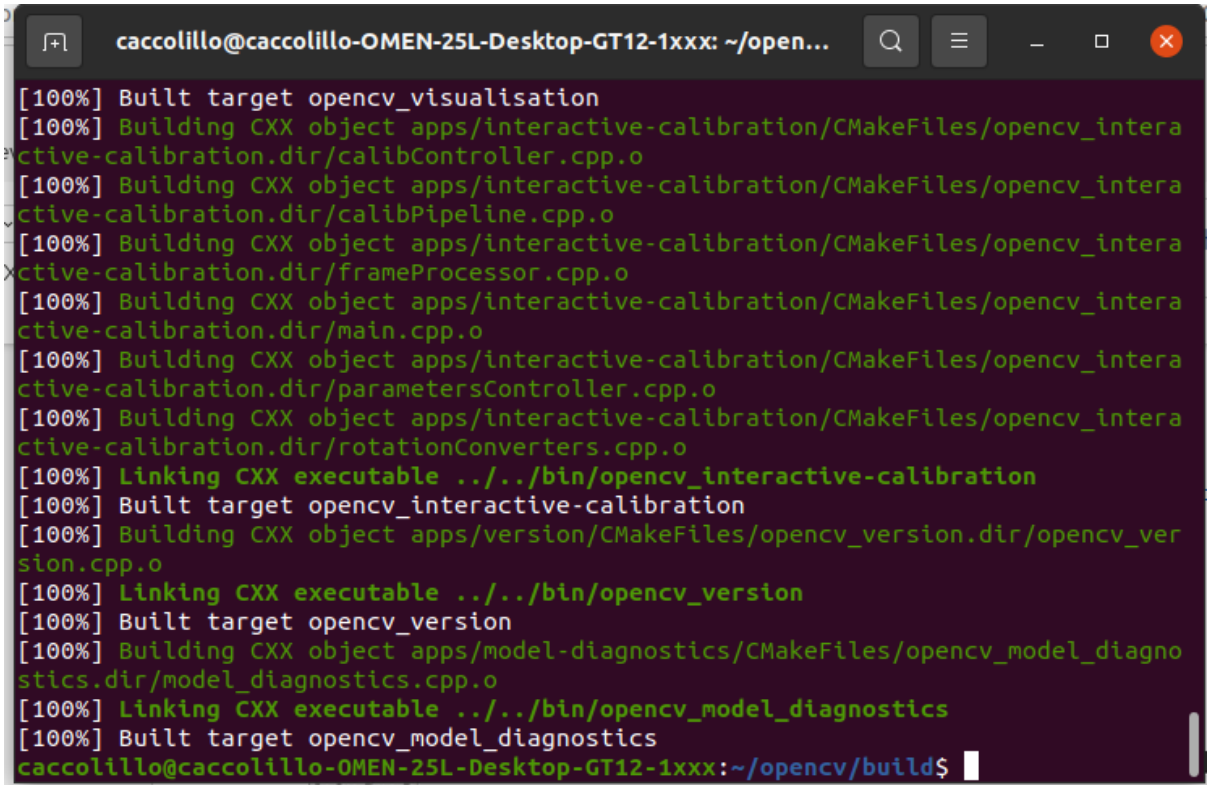
https://github.com/Xilinx/Vitis_Libraries

To compile the OpenCV libraries, the easier way has been to follow the windows like in Linux:

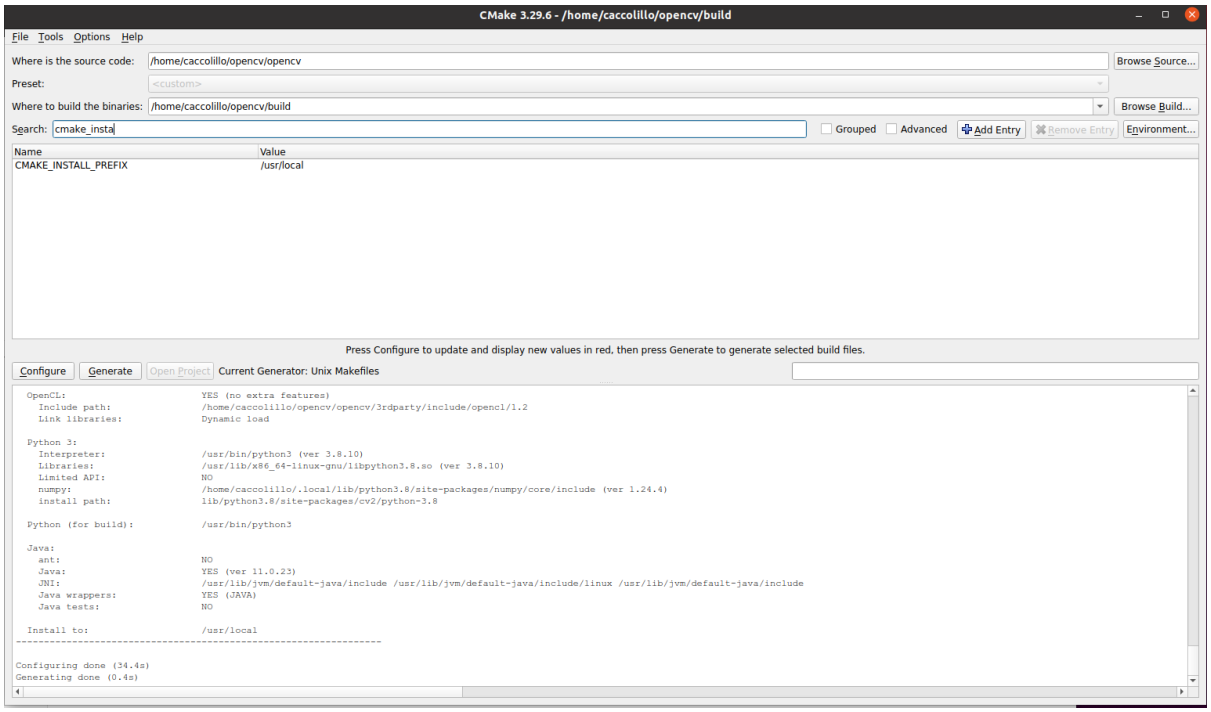
https://support.xilinx.com/s/article/000035890?language=en_US

By using the GUI.





The libraries are installed in /usr/local as per picture below:



Once done export in ".bashrc" as per:

https://support.xilinx.com/s/article/Vitis-Libraries-Compiling-and-Installing-OpenCV?language=en_US

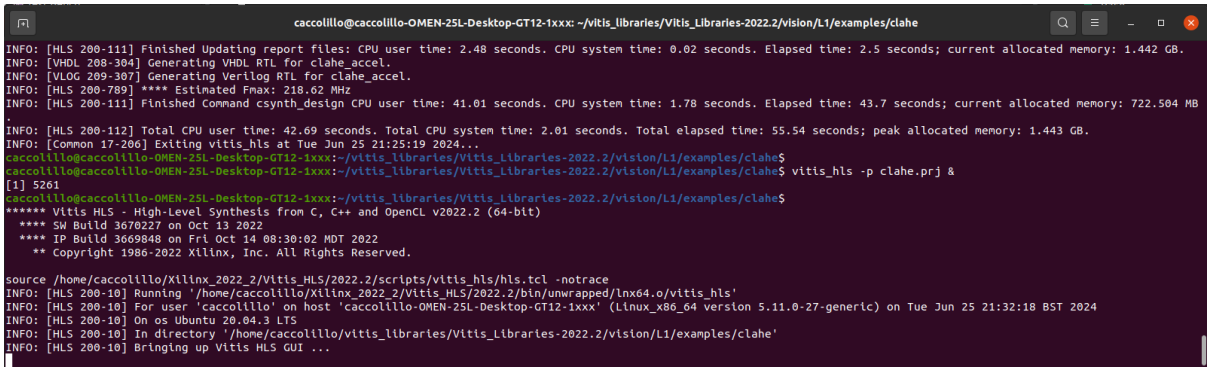
```
Open                                                    .bashrc
 72     ;;
 73 esac
 74
 75 # enable color support of ls and also add handy aliases
 76 if [ -x /usr/bin/dircolors ]; then
 77     test -r ~/.dircolors && eval "$(dircolors -b ~/.dircolors)" || eval "$(dircolors -b)"
 78     alias ls='ls --color=auto'
 79     #alias dir='dir --color=auto'
 80     #alias vdir='vdir --color=auto'
 81
 82     alias grep='grep --color=auto'
 83     alias fgrep='fgrep --color=auto'
 84     alias egrep='egrep --color=auto'
 85 fi
 86
 87 # colored GCC warnings and errors
 88 #export GCC_COLORS='error=01;31:warning=01;35:note=01;36:caret=01;32:locus=01:quote=01'
 89
 90 # some more ls aliases
 91 alias ll='ls -alF'
 92 alias la='ls -A'
 93 alias l='ls -CF'
 94
 95 # Add an "alert" alias for long running commands.  Use like so:
 96 #   sleep 10; alert
 97 alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo error)" "$(history|tail -n1|sed -e '\''s/^\s*[0-9]\+\s*//;s/[;&|]\s*alert$//'\'')"'
 98
 99 # Alias definitions.
100 # You may want to put all your additions into a separate file like
101 # ~/.bash_aliases, instead of adding them here directly.
102 # See /usr/share/doc/bash-doc/examples in the bash-doc package.
103
104 if [ -f ~/.bash_aliases ]; then
105     . ~/.bash_aliases
106 fi
107
108 # enable programmable completion features (you don't need to enable
109 # this, if it's already enabled in /etc/bash.bashrc and /etc/profile
110 # sources /etc/bash.bashrc).
111 if ! shopt -oq posix; then
112   if [ -f /usr/share/bash-completion/bash_completion ]; then
113     . /usr/share/bash-completion/bash_completion
114   elif [ -f /etc/bash_completion ]; then
115     . /etc/bash_completion
116   fi
117 fi
118
119
120 source /home/caccolillo/petalinux/settings.sh
121 source /home/caccolillo/Xilinx_2022_2/Vitis/2022.2/settings64.sh
122 source /home/caccolillo/Xilinx_2022_2/Vitis_HLS/2022.2/settings64.sh
123 source /home/caccolillo/Xilinx_2022_2/Vivado/2022.2/settings64.sh
124
125 export MATLAB="/home/caccolillo/matlab_2021b"
126 ifconfig enp5s0 192.168.1.10
127 PATH=$PATH:/home/caccolillo/temp/cmake-3.28.1/bin
```

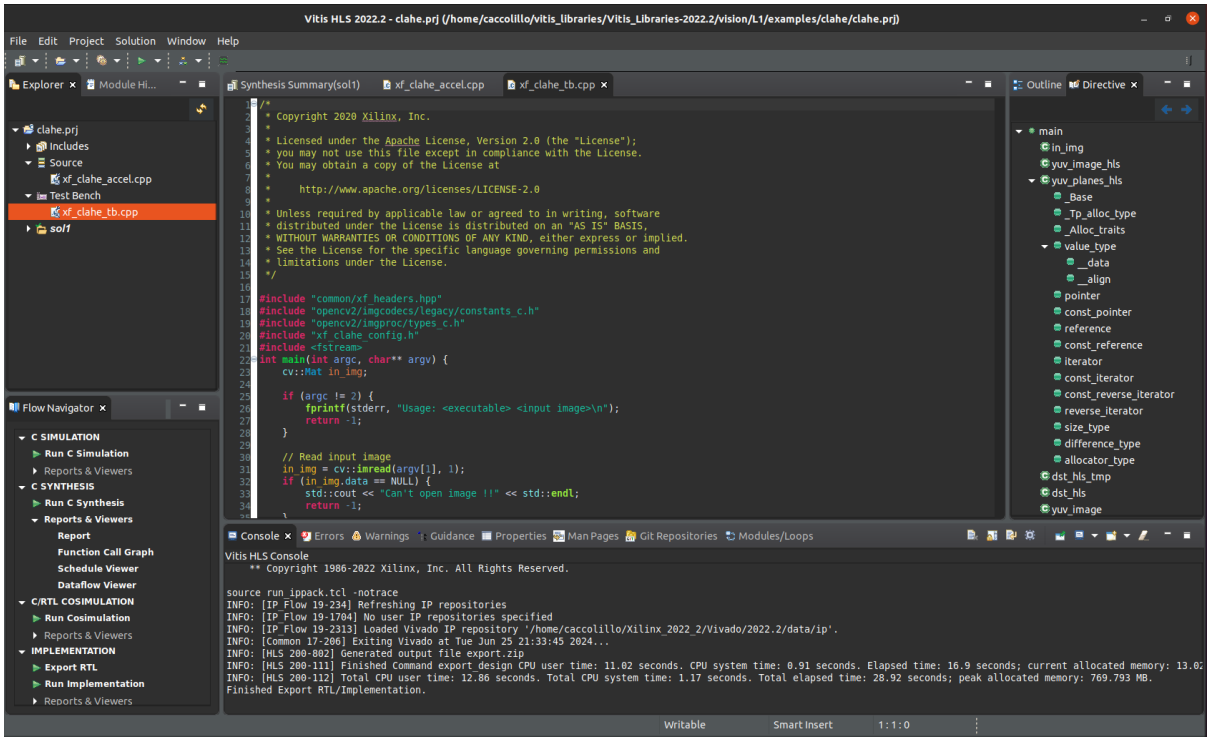
The tutorial below has been followed to create the “clahe”:

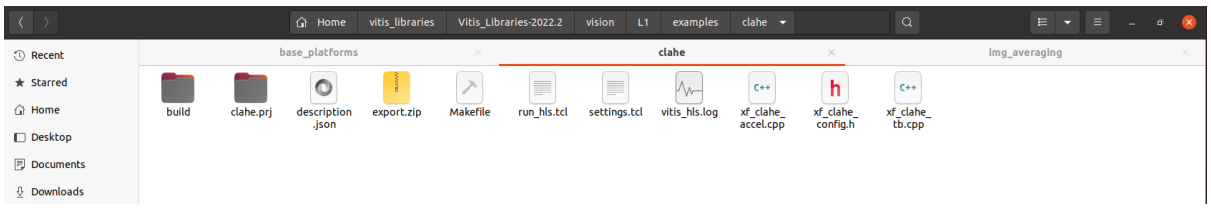
https://xilinx.github.io/Vitis-Tutorials/2022-1/build/html/docs/Getting_Started/Vitis_Libraries/README.html

```
caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx: ~/vitis_libraries/Vitis_Libraries-2022.2/vision/L1/examples/clahe

INFO: [HLS 200-112] Total CPU user time: 12.44 seconds. Total CPU system time: 0.84 seconds. Total elapsed time: 24.62 seconds; peak allocated memory: 754.879 MB.
INFO: [Common 17-206] Exiting vitis_hls at Tue Jun 25 21:23:17 2024...
make: *** [Makefile:250: runhls] Error 1
caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx:~/vitis_libraries/Vitis_Libraries-2022.2/vision/L1/examples/clahe$ make run CSIM=0 CSYNTH=1 COSIM=0 XPART=xczu3eg-sbva484-1-e
XPART is directly set to xczu3eg-sbva484-1-e
Configured: settings.tcl
----
set XPART xczu3eg-sbva484-1-e
set CSIM 0
set CSYNTH 1
set COSIM 0
set VIVADO_SYN 0
set VIVADO_IMPL 0
set XF_PROJ_ROOT "/home/caccolillo/vitis_libraries/Vitis_Libraries-2022.2/vision"
set OPENCV_INCLUDE "/home/caccolillo/opencv/install/include/opencv4"
set OPENCV_LIB "/home/caccolillo/opencv/install/lib"
set CUR_DIR "/home/caccolillo/vitis_libraries/Vitis_Libraries-2022.2/vision/L1/examples/clahe"
----
vitis_hls -f run_hls.tcl;

****** Vitis HLS - High-Level Synthesis from C, C++ and OpenCL v2022.2 (64-bit)
  **** SW Build 3670227 on Oct 13 2022
  **** IP Build 3669848 on Fri Oct 14 08:30:02 MDT 2022
```

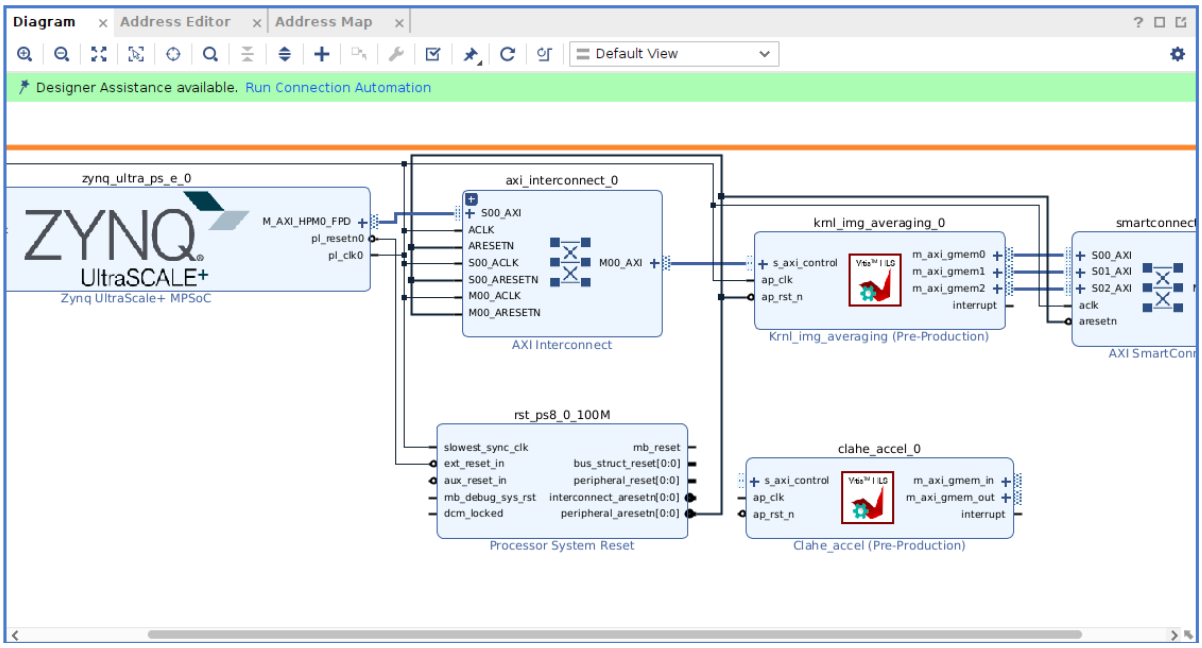
Opening the vitis_hls gui end exporting the IP, a zip file gets created:







By adding the repo in the main vivado project, the IP can be imported and added to the block design:



By trying to run the testbench in Vitis HLS, the tool complains about OpenCV, to sort this out, the following ar# must be followed:

https://support.xilinx.com/s/article/75727?language=en_US

VITIS CLAHE SW implementation

The CLAHE algorithm pure sw implementation has been attempted in Vitis.

Tutorials about how to use OpenCV in vitis are available at:

https://beetlebox.org/getting-started-with-computer-vision-for-vitis-embedded-systems-part-3-using-opencv-and-file-transfers/

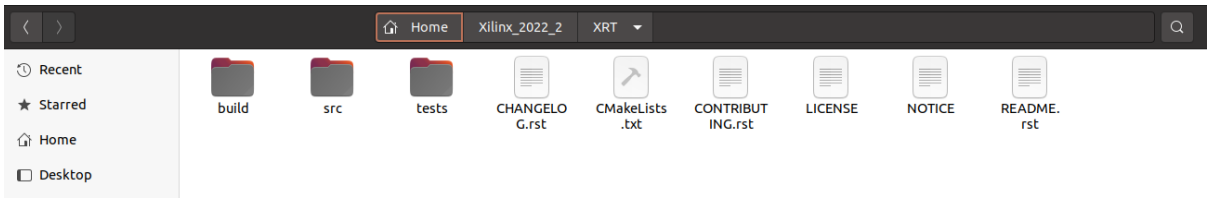
https://beetlebox.org/getting-started-with-computer-vision-for-vitis-embedded-systems-part-4-vitis-vision-library/

XRT has been downloaded and installed as per:

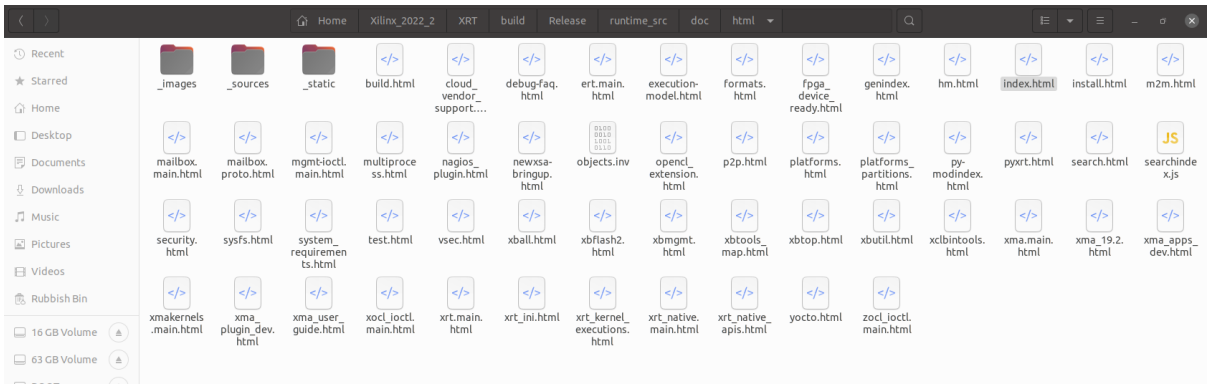
https://xilinx.github.io/XRT/master/html/test.html

https://github.com/Xilinx/XRT/tree/2022.2?tab=readme-ov-file

In the folder:



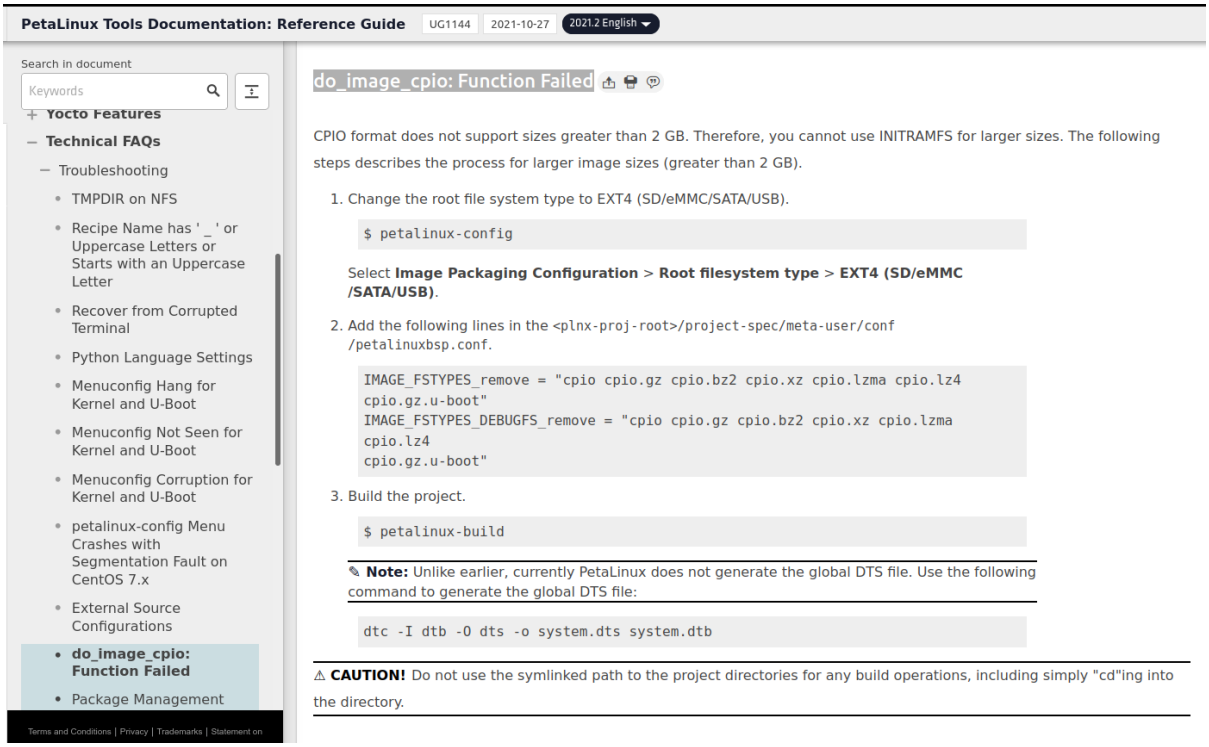
At the end, the documentation is in:



PETALINUX do_image_cpio: Function Failed

If during petalinux build this error appears, the solution is at:

https://docs.amd.com/r/2021.2-English/ug1144-petalinux-tools-reference-guide/do_image_cpio-Function-Failed



ADD OPENCV AND OTHER LIBRARIES TO VITIS

In order to use openCV, XRT and other ROOTFS components, the following tutorial has been followed:

https://support.xilinx.com/s/article/1175461?language=en_US

https://support.xilinx.com/s/article/69166?language=en_US

https://support.xilinx.com/s/article/1141772?language=en_US

https://support.xilinx.com/s/article/1138750?language=en_US

https://support.xilinx.com/s/article/1138667?language=en_US

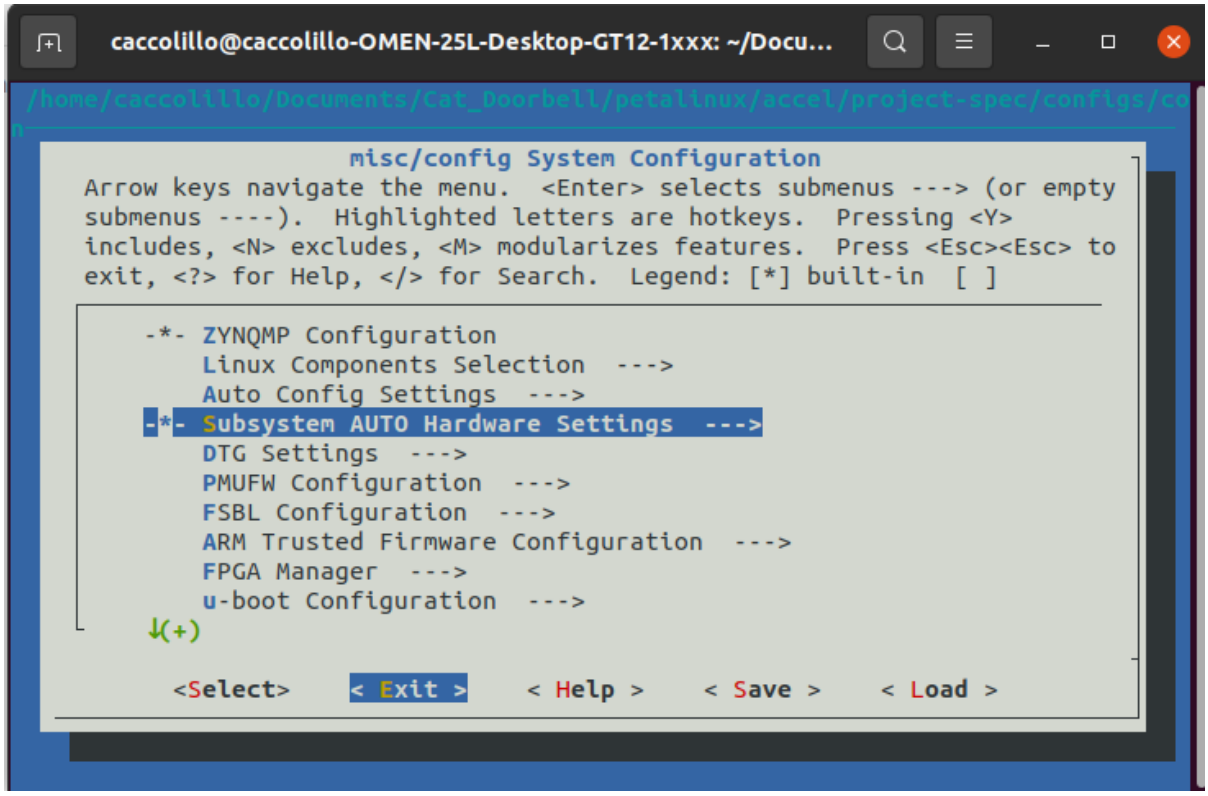
After having created SDK, we get a confirmation about openCV:

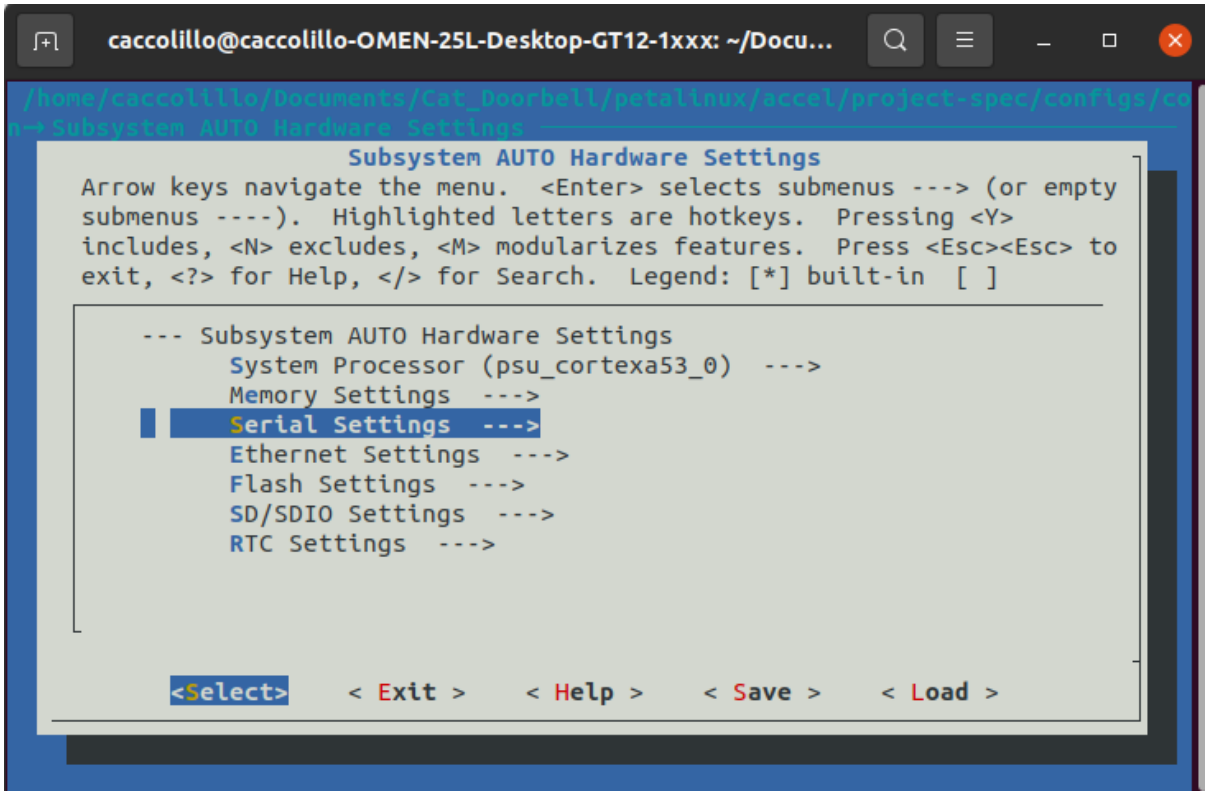
```
caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx:~/Documents/Cat_Doorbell/petalinux/accel/images/linux$ find ./ -name "*opencv*"
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_xfeatures2d.so.4.5
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_stereo.so
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_xphoto.so.4.5
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_face.so.4.5
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_bioinspired.so.4.5.2
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_hfs.so.4.5
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_objdetect.so
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_ccalib.so.4.5
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_line_descriptor.so.4.5.2
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_ts.so.4.5
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_rgbd.so
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_imgproc.so.4.5
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_superres.so.4.5.2
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_rapid.so
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_core.so.4.5
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_dnn_objdetect.so
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_wechat_qrcode.so.4.5.2
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_dnn_superres.so.4.5
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_photo.so
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_alphamat.so
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_reg.so
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_videoio.so
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_ccalib.so
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_imgcodecs.so.4.5
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_ml.so
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_xphoto.so.4.5.2
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_video.so
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_xobjdetect.so
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_objdetect.so.4.5.2
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_surface_matching.so.4.5
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_structured_light.so
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_optflow.so.4.5
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_highgui.so.4.5
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_ximgproc.so.4.5.2
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_datasets.so.4.5
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_fuzzy.so.4.5.2
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_img_hash.so
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_dnn.so
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_img_hash.so.4.5
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_intensity_transform.so.4.5
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_plot.so
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_features2d.so
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_surface_matching.so
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_plot.so.4.5
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_aruco.so.4.5
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_xfeatures2d.so.4.5.2
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_xobjdetect.so.4.5.2
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_line_descriptor.so
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_flann.so.4.5
./sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib/libopencv_ts.so
```

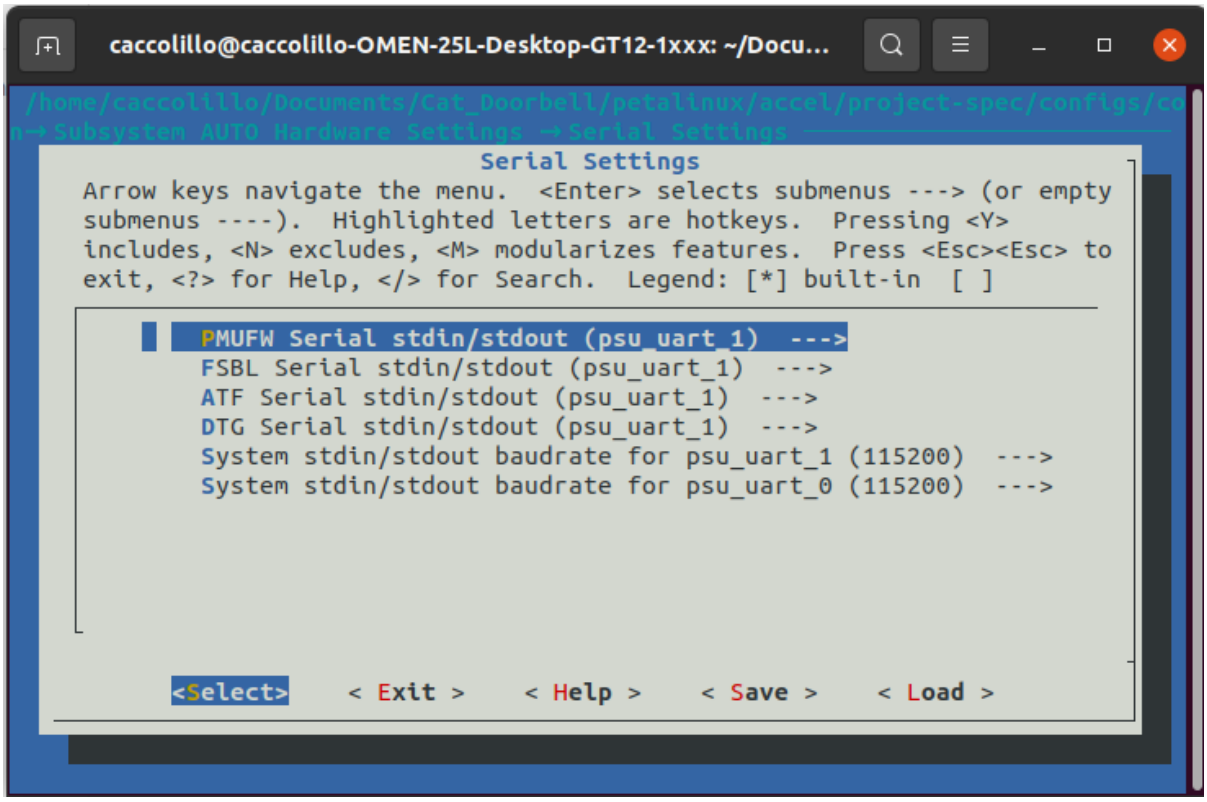
END PETALINUX CONFIGURATION SUPPORTING OPENCV, XRT AND ROOT FILESYSTEM ON MICROSD

petalinux-config:

-set stdin stdout uart to psu_uart_1

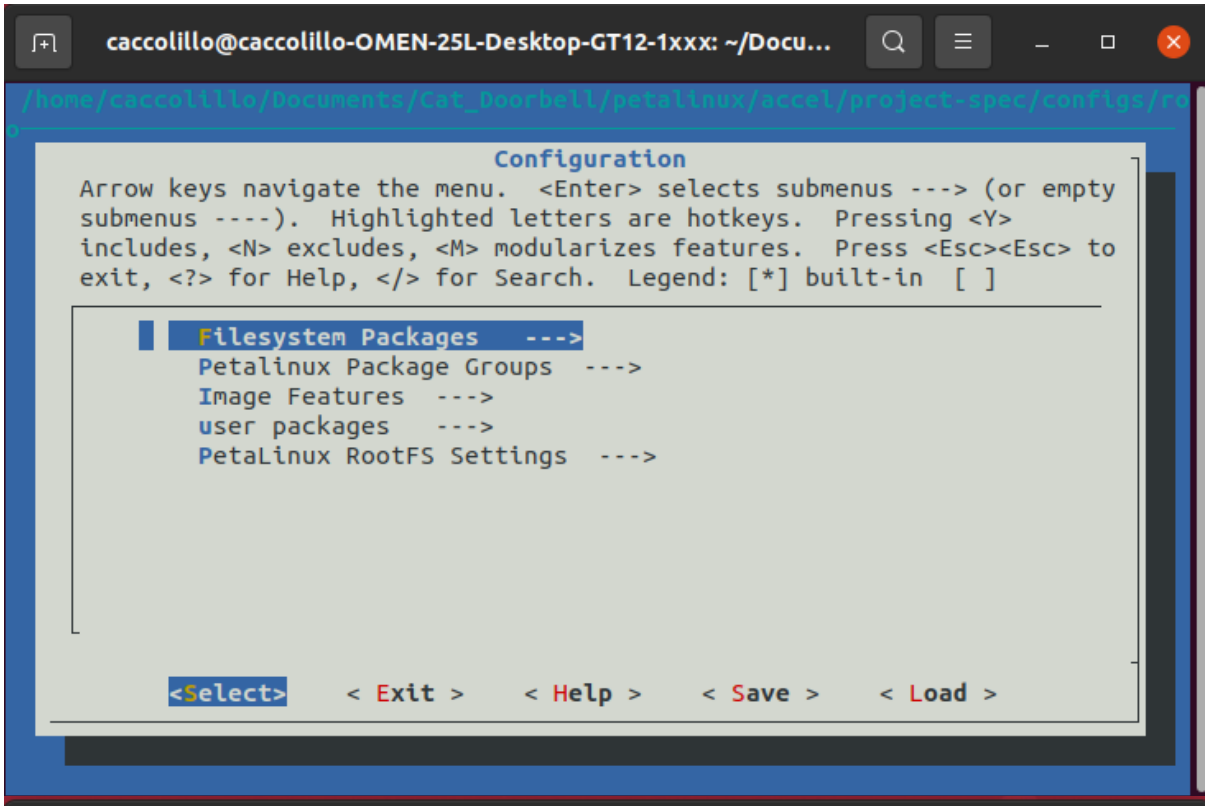

caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx: ~/Docu...
/home/caccolillo/Documents/Cat_Doorbell/petalinux/accel/project-spec/configs/co
misc/config System Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]
-*- ZYNQMP Configuration
Linux Components Selection --->
Auto Config Settings --->
-*- Subsystem AUTO Hardware Settings --->
DTG Settings --->
PMUFW Configuration --->
FSBL Configuration --->
ARM Trusted Firmware Configuration --->
FPGA Manager --->
u-boot Configuration --->
<Select> < Exit > < Help > < Save > < Load >

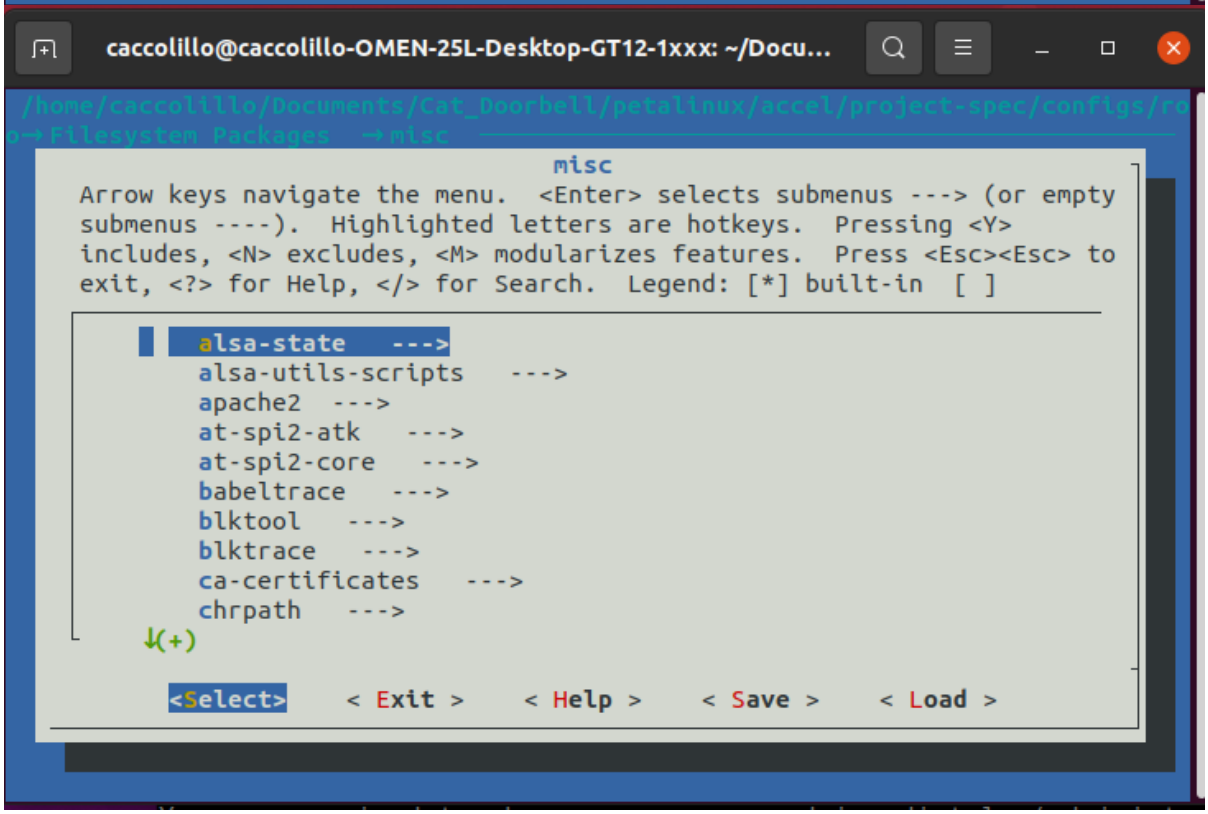

caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx: ~/Docu...
/home/caccolillo/Documents/Cat_Doorbell/petalinux/accel/project-spec/configs/co
n→Subsystem AUTO Hardware Settings
Subsystem AUTO Hardware Settings
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]
--- Subsystem AUTO Hardware Settings
System Processor (psu_cortexa53_0) --->
Memory Settings --->
Serial Settings --->
Ethernet Settings --->
Flash Settings --->
SD/SDIO Settings --->
RTC Settings --->
<Select> < Exit > < Help > < Save > < Load >

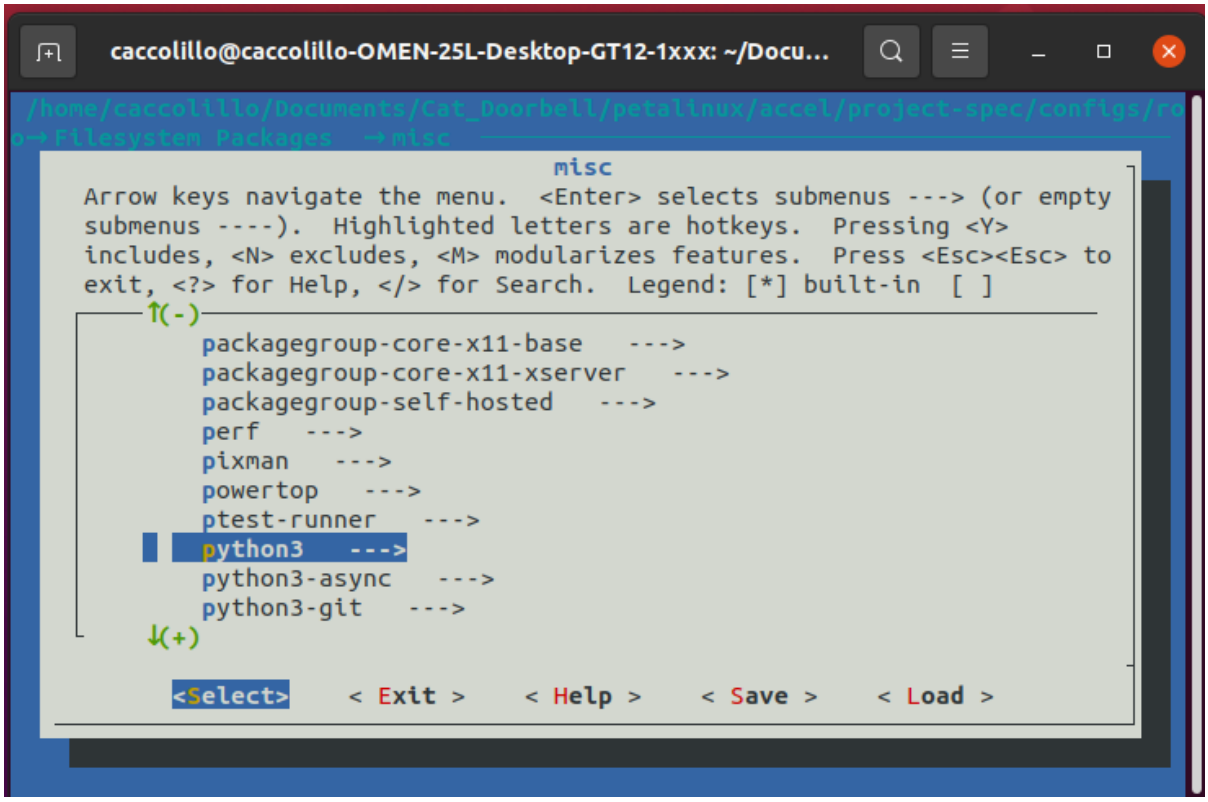


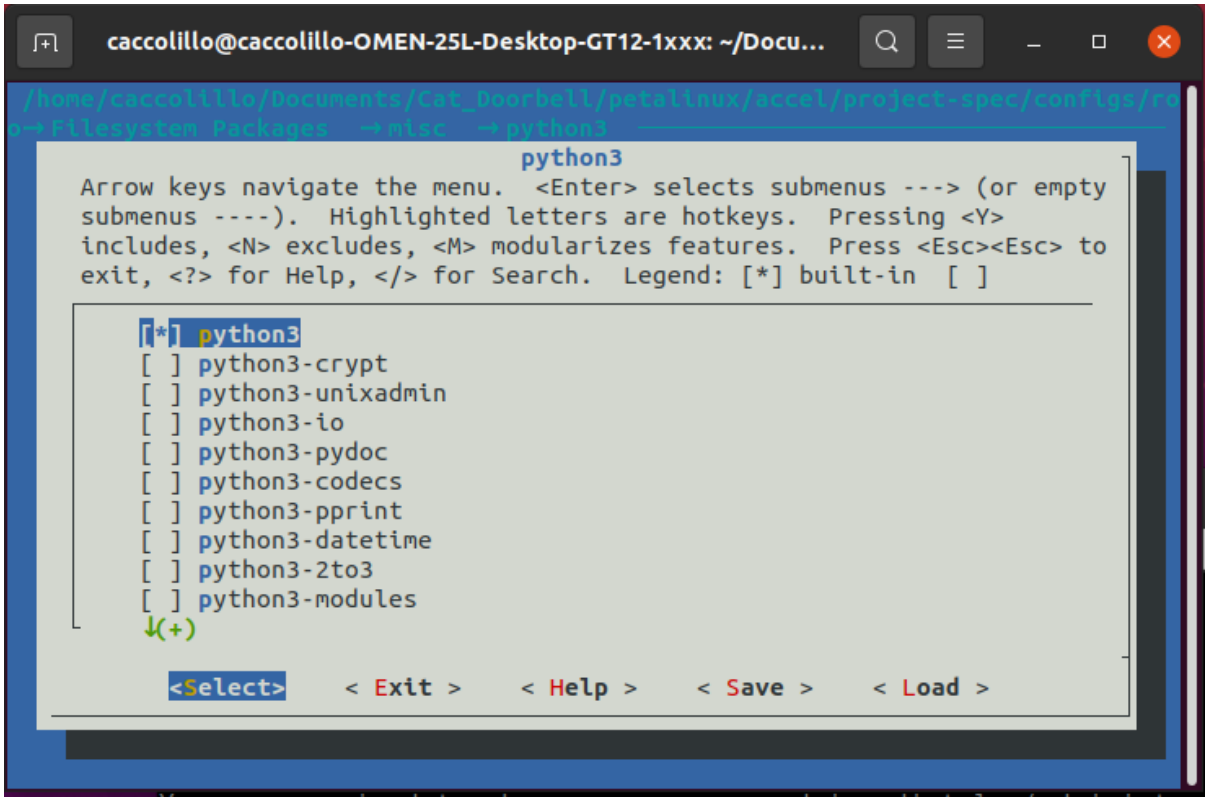
petalinux-config -c rootfs

Enable python

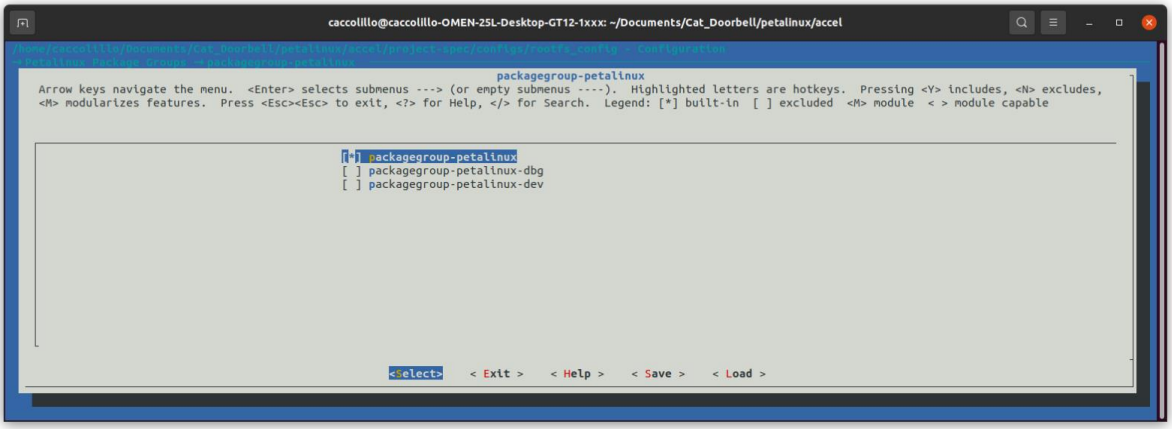

caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx: ~/Docu...
/home/caccolillo/Documents/Cat_Doorbell/petalinux/accel/project-spec/configs/ro
Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]
Filesystem Packages --->
Petalinux Package Groups --->
Image Features --->
user packages --->
PetaLinux RootFS Settings --->
<Select> < Exit > < Help > < Save > < Load >

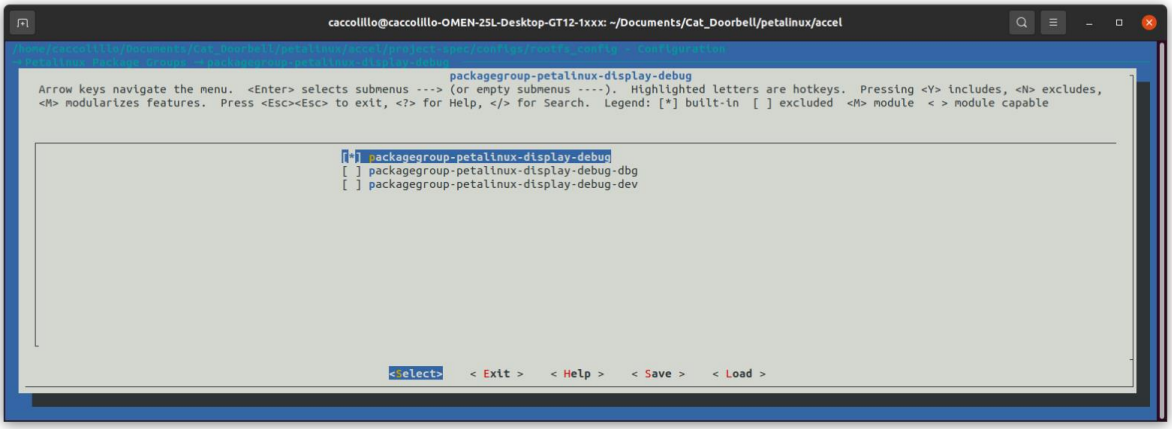

caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx: ~/Docu...
/home/caccolillo/Documents/Cat_Doorbell/petalinux/accel/project-spec/configs/ro
→Filesystem Packages →misc
misc
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]
alsa-state --->
alsa-utils-scripts --->
apache2 --->
at-spi2-atk --->
at-spi2-core --->
babeltrace --->
blktool --->
blktrace --->
ca-certificates --->
chrpath --->
↓(+)
<Select> < Exit > < Help > < Save > < Load >

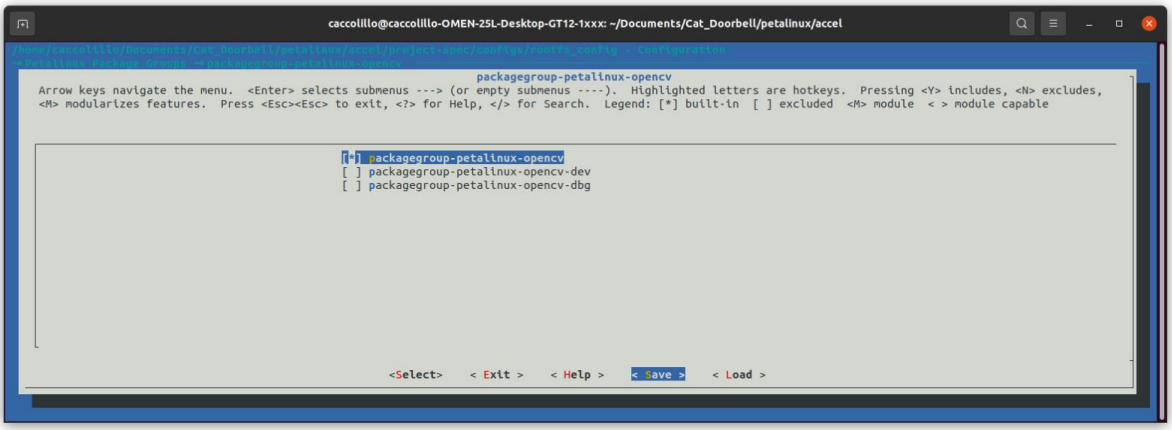


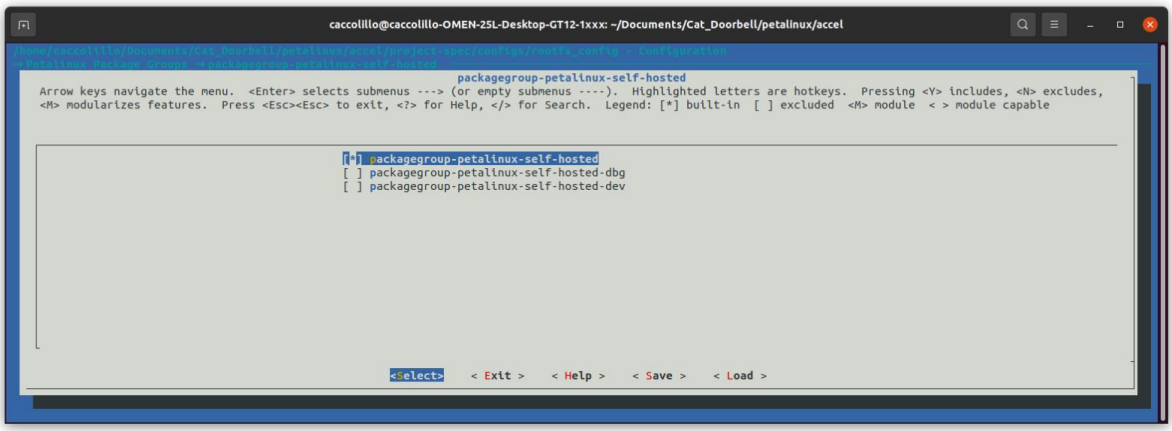


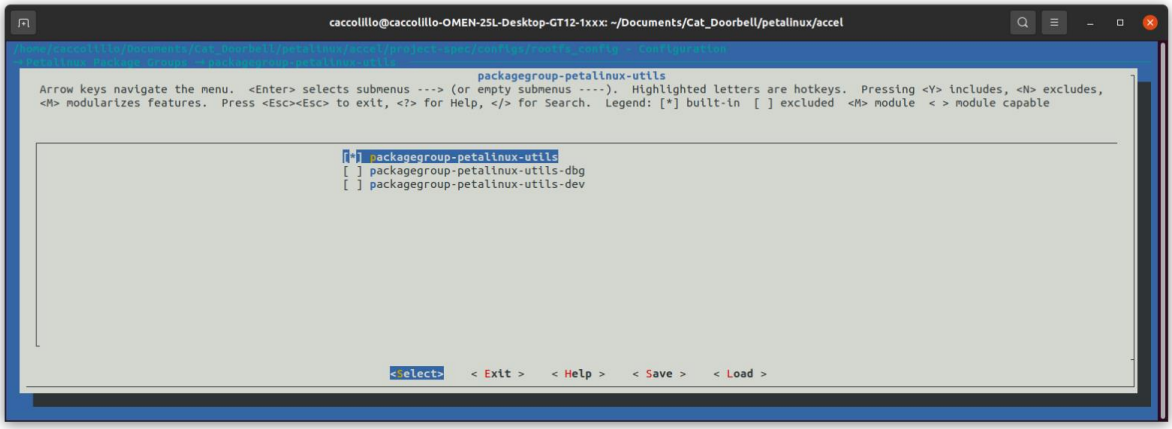
Enable webcam and OpenCV

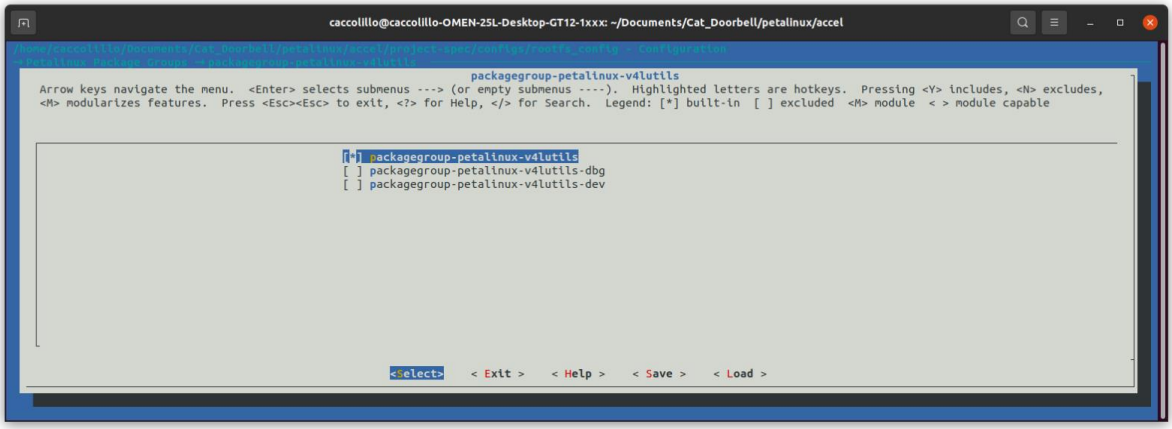

caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx: ~/Documents/Cat_Doorbell/petalinux/accel
/home/caccolillo/Documents/Cat_Doorbell/petalinux/accel/project-spec/configs/rootfs_config - Configuration
→ PetaLinux Package Groups → packagegroup-petalinux
packagegroup-petalinux
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
[*] packagegroup-petalinux
[ ] packagegroup-petalinux-dbg
[ ] packagegroup-petalinux-dev
<Select> < Exit > < Help > < Save > < Load >

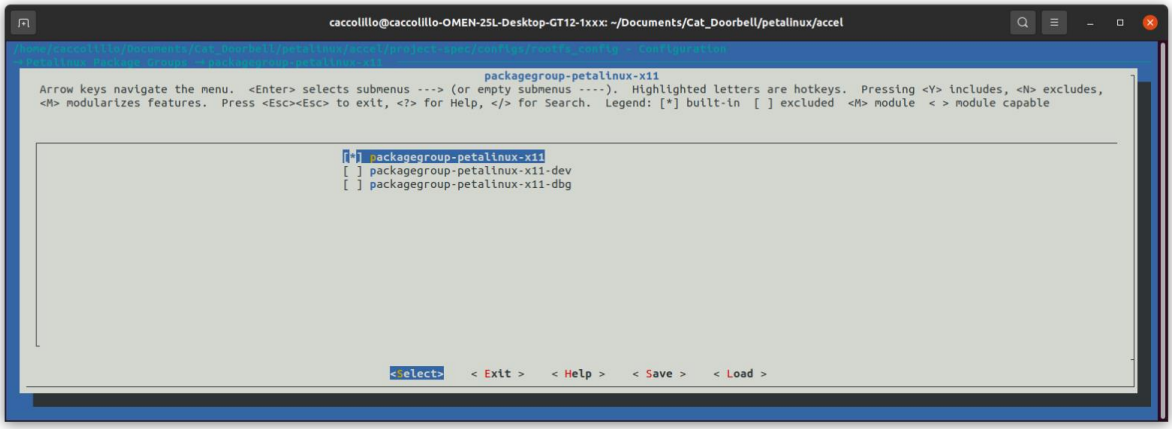

caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx: ~/Documents/Cat_Doorbell/petalinux/accel
/home/caccolillo/Documents/Cat_Doorbell/petalinux/accel/project-spec/configs/rootfs_config - Configuration
→ PetaLinux Package Groups → packagegroup-petalinux-display-debug
packagegroup-petalinux-display-debug
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
[*] packagegroup-petalinux-display-debug
[ ] packagegroup-petalinux-display-debug-dbg
[ ] packagegroup-petalinux-display-debug-dev
<Select> < Exit > < Help > < Save > < Load >

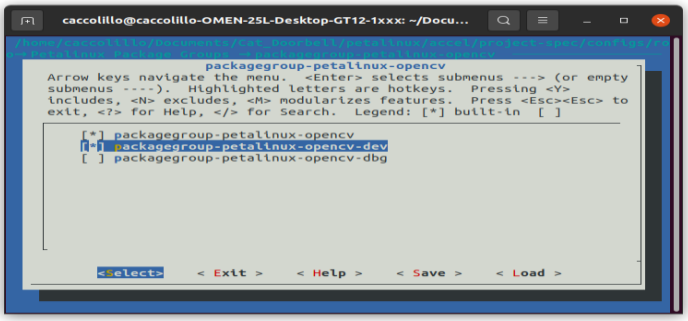

caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx: ~/Documents/Cat_Doorbell/petalinux/accel
/home/caccolillo/Documents/Cat_Doorbell/petalinux/accel/project-spec/configs/rootfs_config - Configuration
→ PetaLinux Package Groups → packagegroup-petalinux-opencv
packagegroup-petalinux-opencv
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
[*] packagegroup-petalinux-opencv
[ ] packagegroup-petalinux-opencv-dev
[ ] packagegroup-petalinux-opencv-dbg
<Select> < Exit > < Help > < Save > < Load >

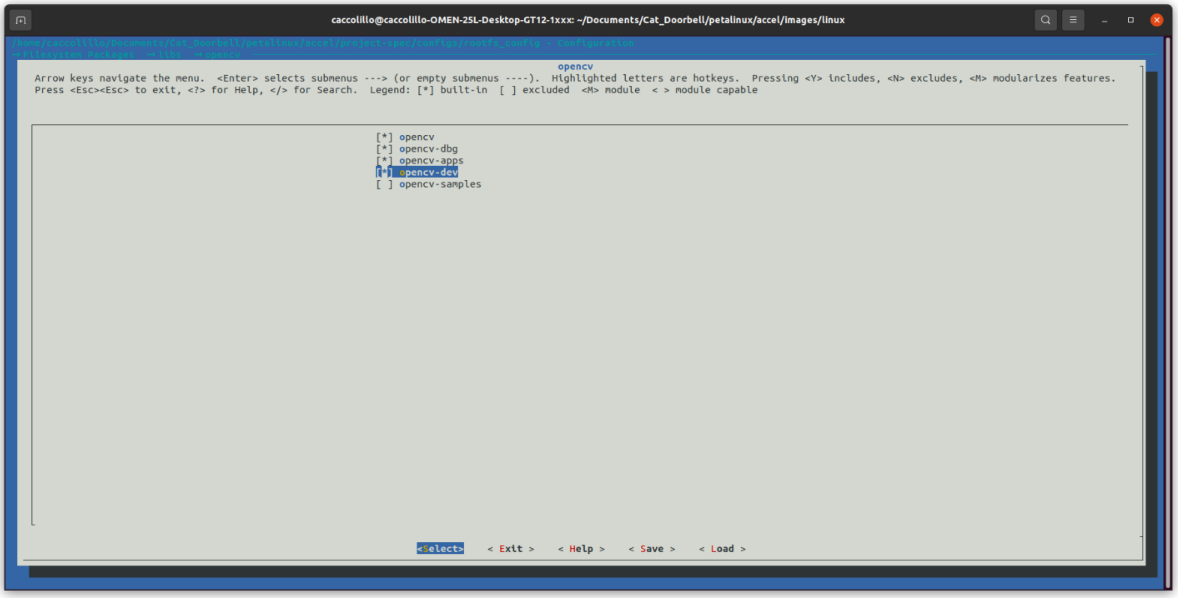

caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx: ~/Documents/Cat_Doorbell/petalinux/accel
/home/caccolillo/Documents/Cat_Doorbell/petalinux/accel/project-spec/configs/rootfs_config - Configuration
→ PetaLinux Package Groups → packagegroup-petalinux-self-hosted
packagegroup-petalinux-self-hosted
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
[*] packagegroup-petalinux-self-hosted
[ ] packagegroup-petalinux-self-hosted-dbg
[ ] packagegroup-petalinux-self-hosted-dev
<Select> < Exit > < Help > < Save > < Load >


caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx: ~/Documents/Cat_Doorbell/petalinux/accel
packagegroup-petalinux-utils
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
[*] packagegroup-petalinux-utils
[ ] packagegroup-petalinux-utils-dbg
[ ] packagegroup-petalinux-utils-dev
<Select> < Exit > < Help > < Save > < Load >


caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx: ~/Documents/Cat_Doorbell/petalinux/accel
packagegroup-petalinux-v4lutils
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
[*] packagegroup-petalinux-v4lutils
[ ] packagegroup-petalinux-v4lutils-dbg
[ ] packagegroup-petalinux-v4lutils-dev
<Select> < Exit > < Help > < Save > < Load >

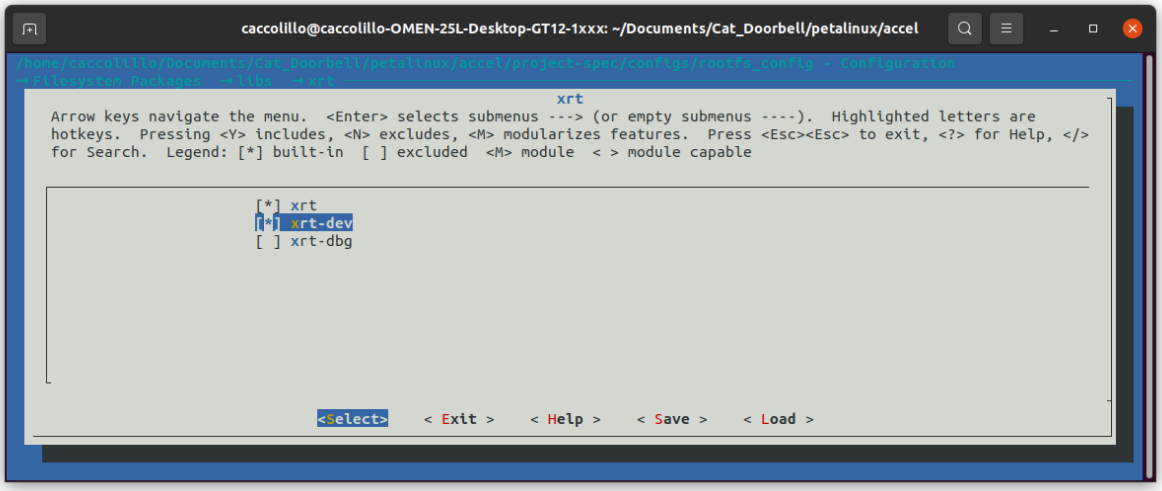

caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx: ~/Documents/Cat_Doorbell/petalinux/accel
packagegroup-petalinux-x11
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
[*] packagegroup-petalinux-x11
[ ] packagegroup-petalinux-x11-dev
[ ] packagegroup-petalinux-x11-dbg
<Select> < Exit > < Help > < Save > < Load >

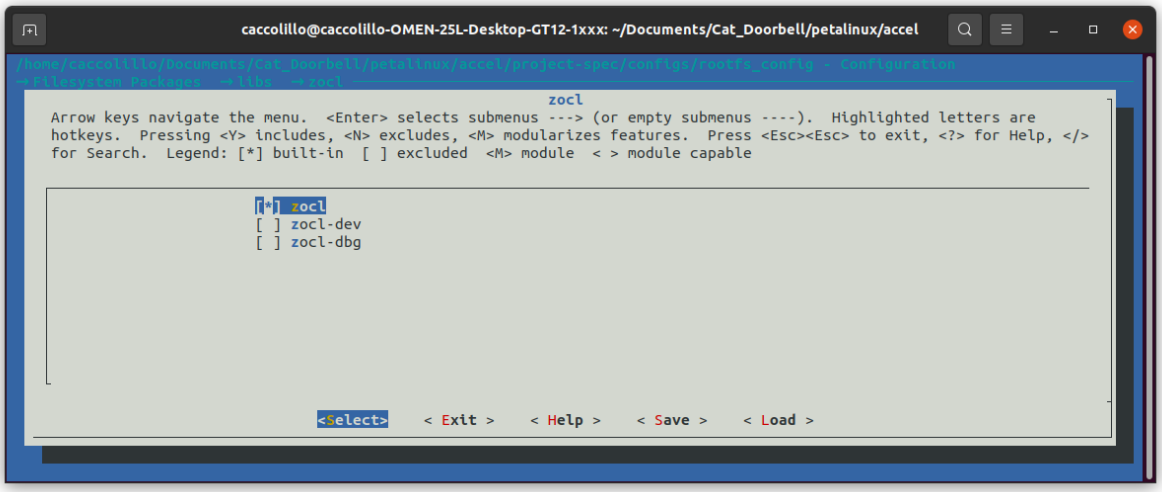

caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx: ~/Docu...
packagegroup-petalinux-opencv
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]
[*] packagegroup-petalinux-opencv
[*] packagegroup-petalinux-opencv-dev
[ ] packagegroup-petalinux-opencv-dbg
<Select> < Exit > < Help > < Save > < Load >

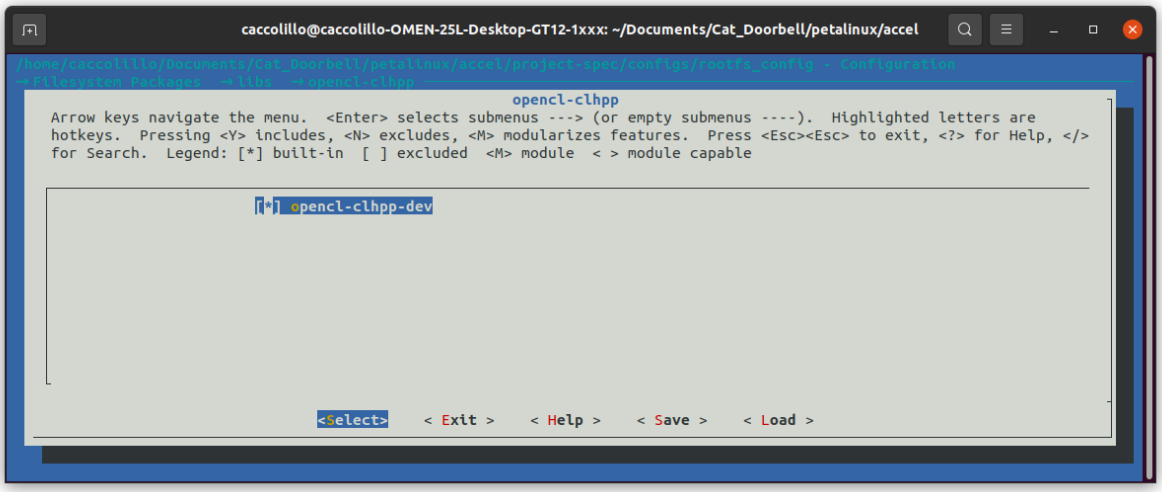


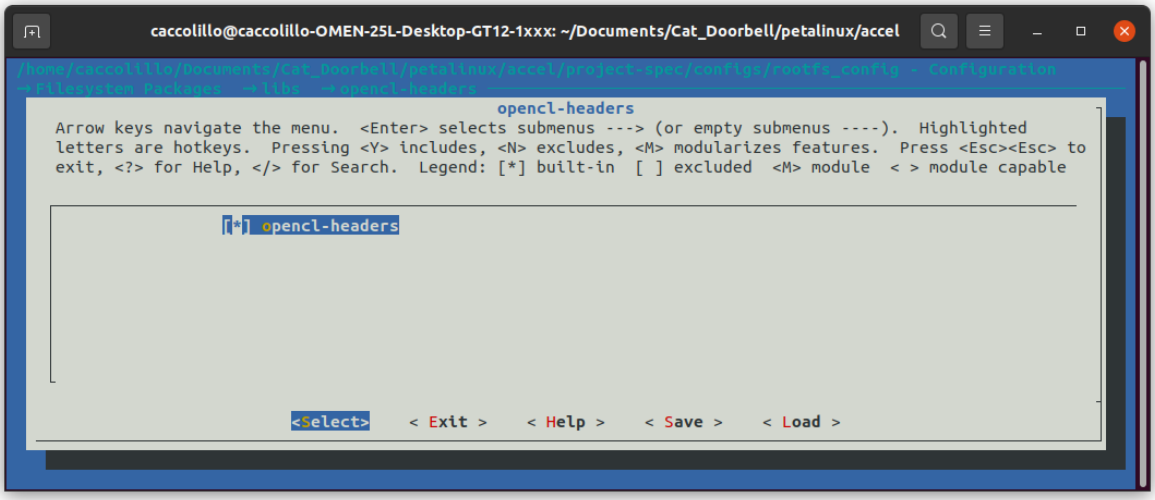
Enable XRT and OpenCL

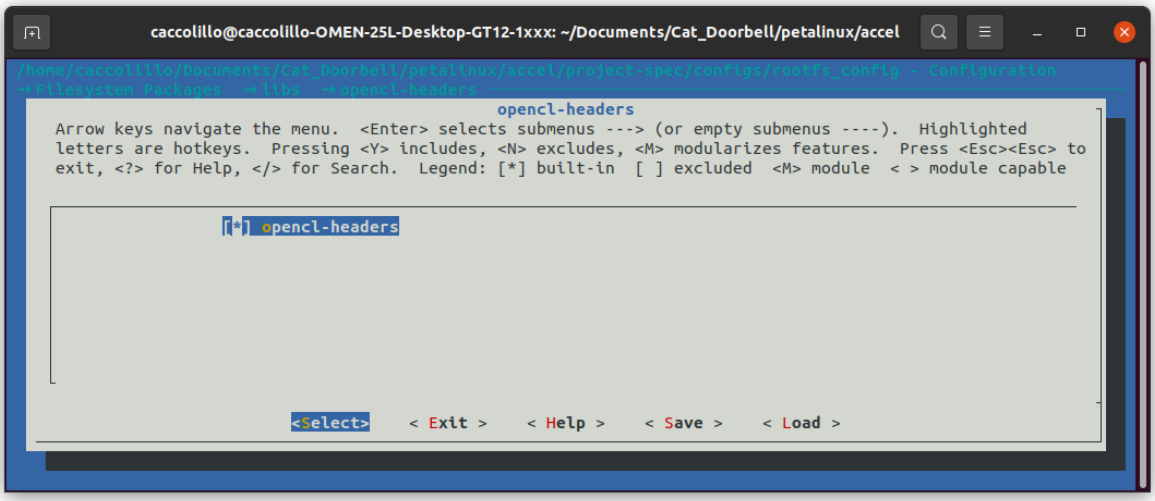
petalinux-config -c rootfs




caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx: ~/Documents/Cat_Doorbell/petalinux/accel
/home/caccolillo/Documents/Cat_Doorbell/petalinux/accel/project-spec/configs/rootfs_config - Configuration
→ Filesystem Packages → libs → zocl
zocl
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are
hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
[*] zocl
[ ] zocl-dev
[ ] zocl-dbg
<Select> < Exit > < Help > < Save > < Load >


caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx: ~/Documents/Cat_Doorbell/petalinux/accel
/home/caccolillo/Documents/Cat_Doorbell/petalinux/accel/project-spec/configs/rootfs_config - Configuration
→ Filesystem Packages → libs → opencl-clhpp
opencl-clhpp
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are
hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
[*] opencl-clhpp-dev
<Select> < Exit > < Help > < Save > < Load >

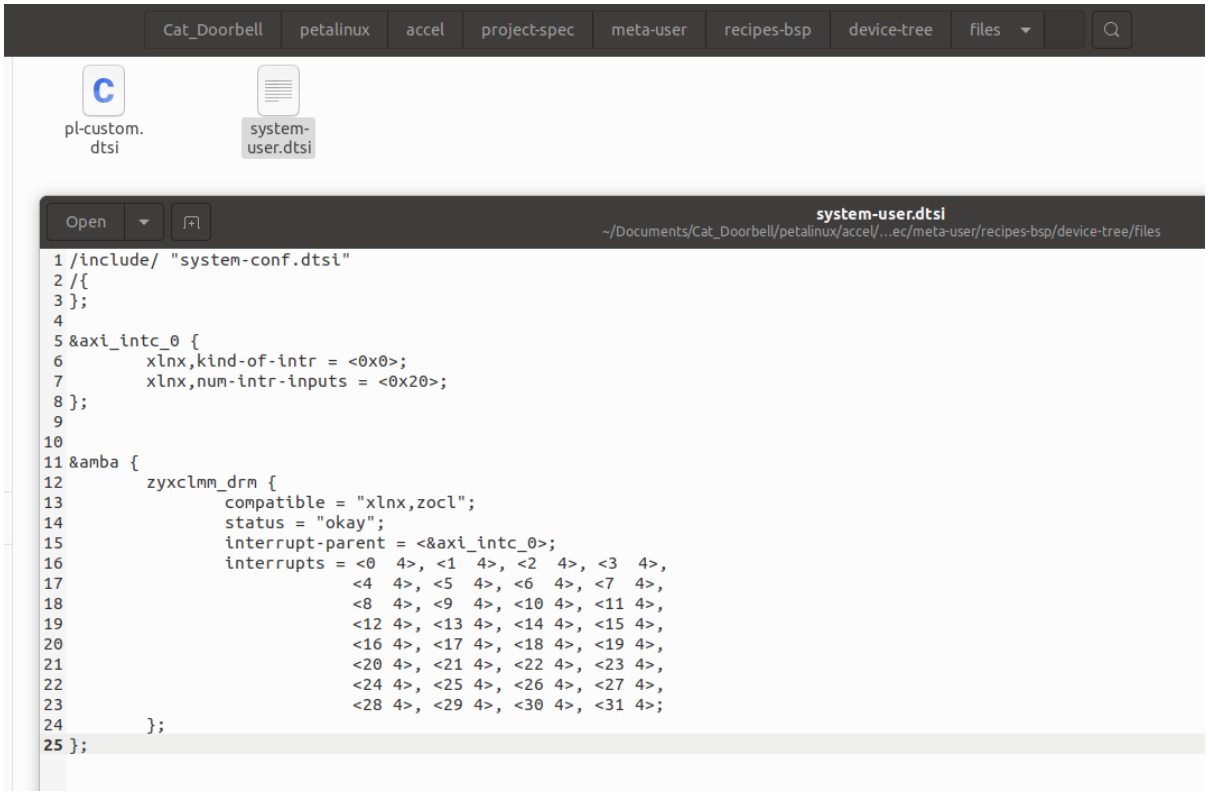




Change the *system-user.dtsi* as below:

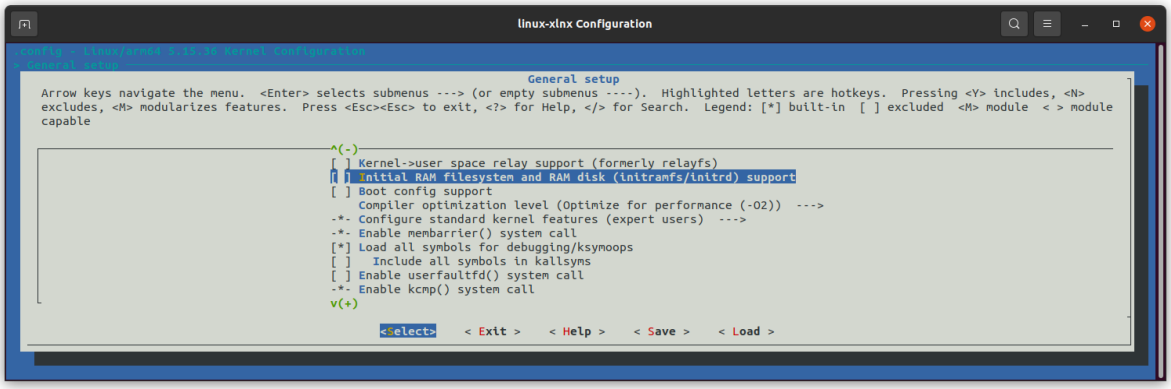
```
/include/ "system-conf.dtsi"
/{
 chosen {
      bootargs = "console=ttyPS0,115200 root=/dev/mmcblk0p2 rw earlyprintk
rootfstype=ext4 rootwait uio_pdrv_genirq.of_id=generic-uio devtmpfs.mount=1
earlycon";
   };
};

&axi_intc_0 {
   xlnx,kind-of-intr = <0x0>;
   xlnx,num-intr-inputs = <0x20>;
};


&amba {
   zyxclmm_drm {
      compatible = "xlnx,zocl";
```

```
        status = "okay";
        interrupt-parent = <&axi_intc_0>;
        interrupts = <0  4>, <1  4>, <2  4>, <3  4>,
                     <4  4>, <5  4>, <6  4>, <7  4>,
                     <8  4>, <9  4>, <10 4>, <11 4>,
                     <12 4>, <13 4>, <14 4>, <15 4>,
                     <16 4>, <17 4>, <18 4>, <19 4>,
                     <20 4>, <21 4>, <22 4>, <23 4>,
                     <24 4>, <25 4>, <26 4>, <27 4>,
                     <28 4>, <29 4>, <30 4>, <31 4>;
    };
};
```

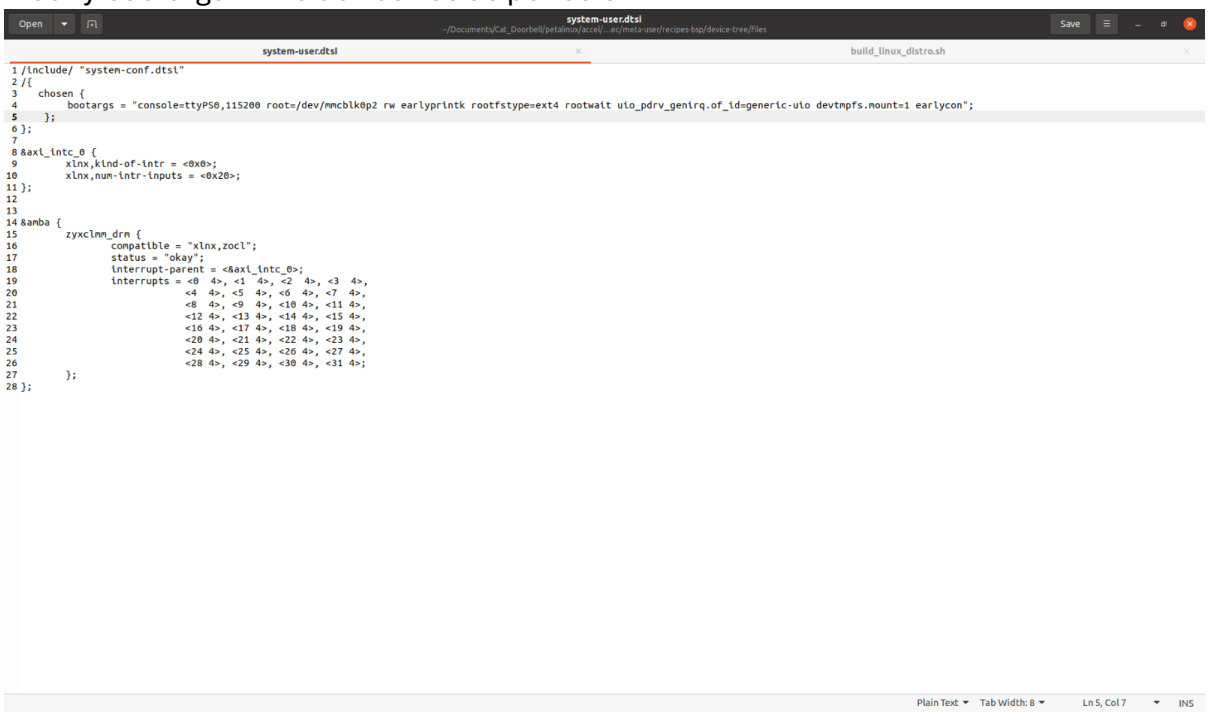


Disable ramdisk

petalinux-config -c kernel



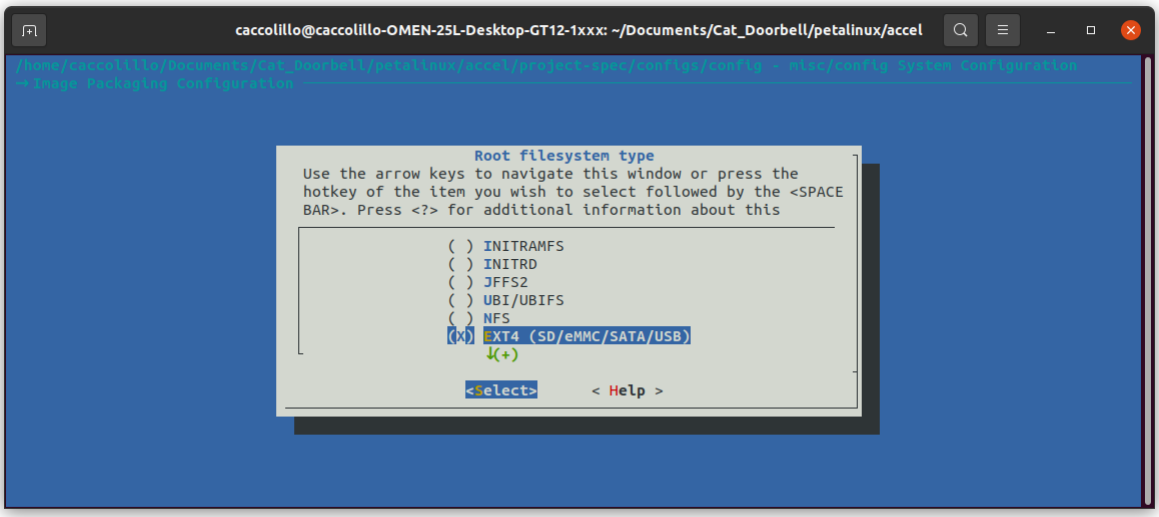
Modify bootargs in the devicetree as per below:



1) copy boot.scr, BOOT.BIN and image.ub to BOOT partition

2) run tar -xvf ./images/linux/rootfs.tar.gz -C /media/camelo/ROOT

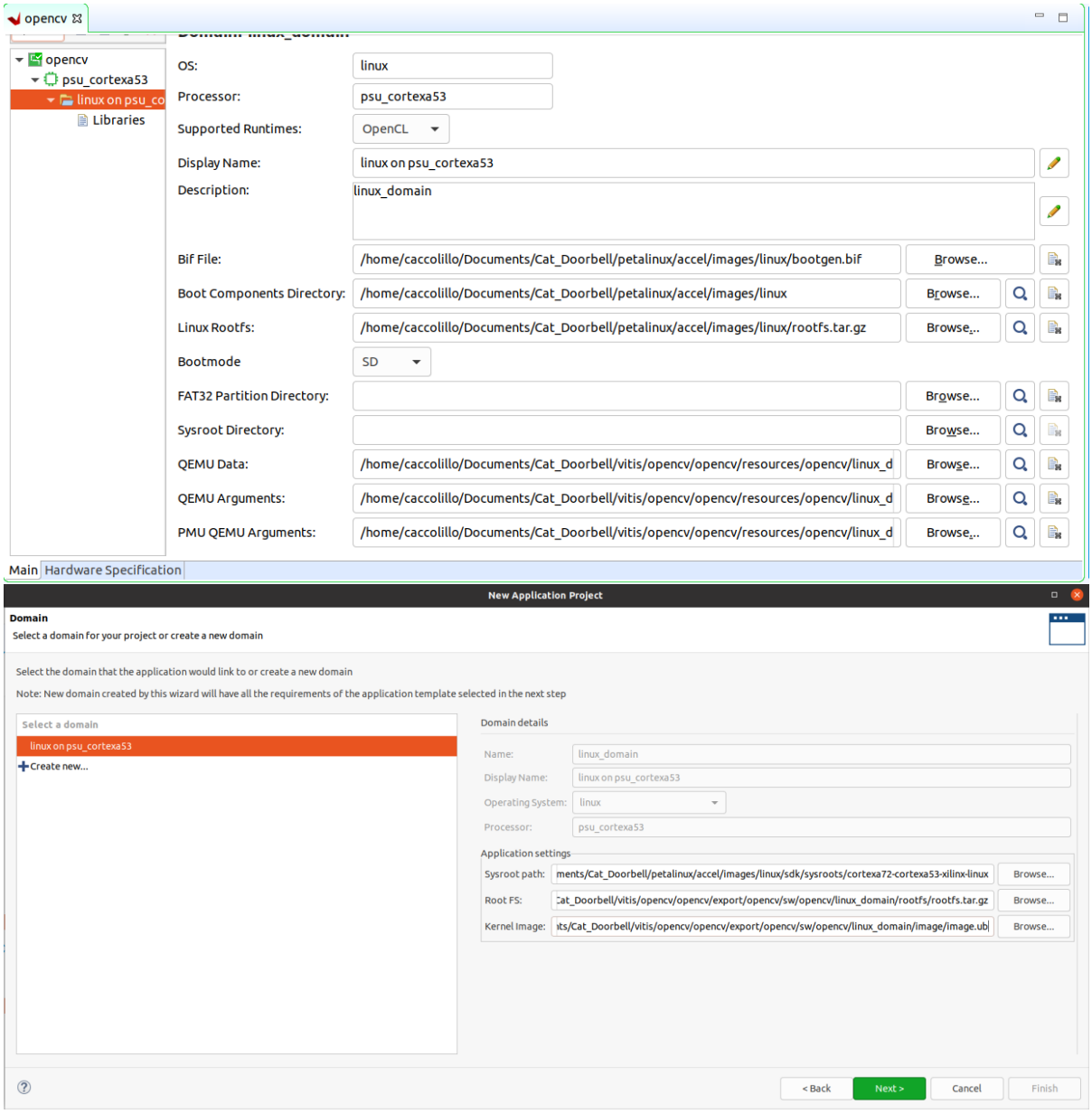
petalinux-config -c kernel

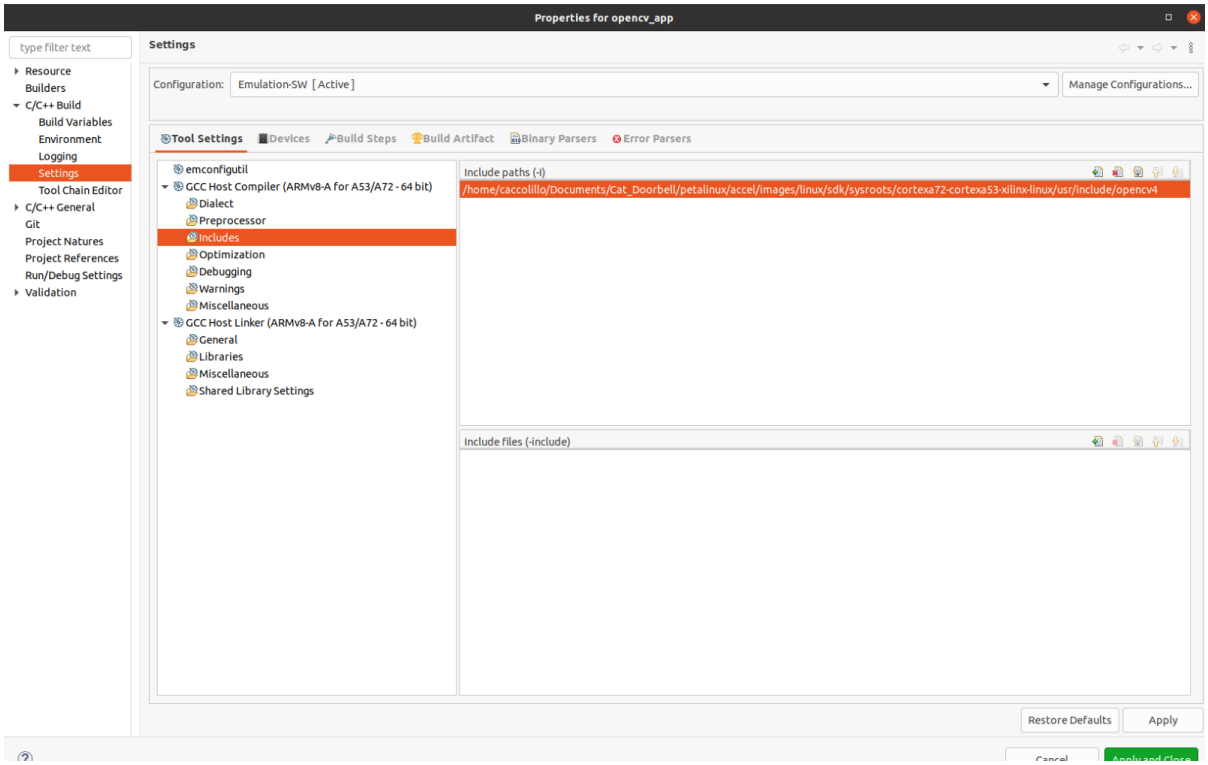


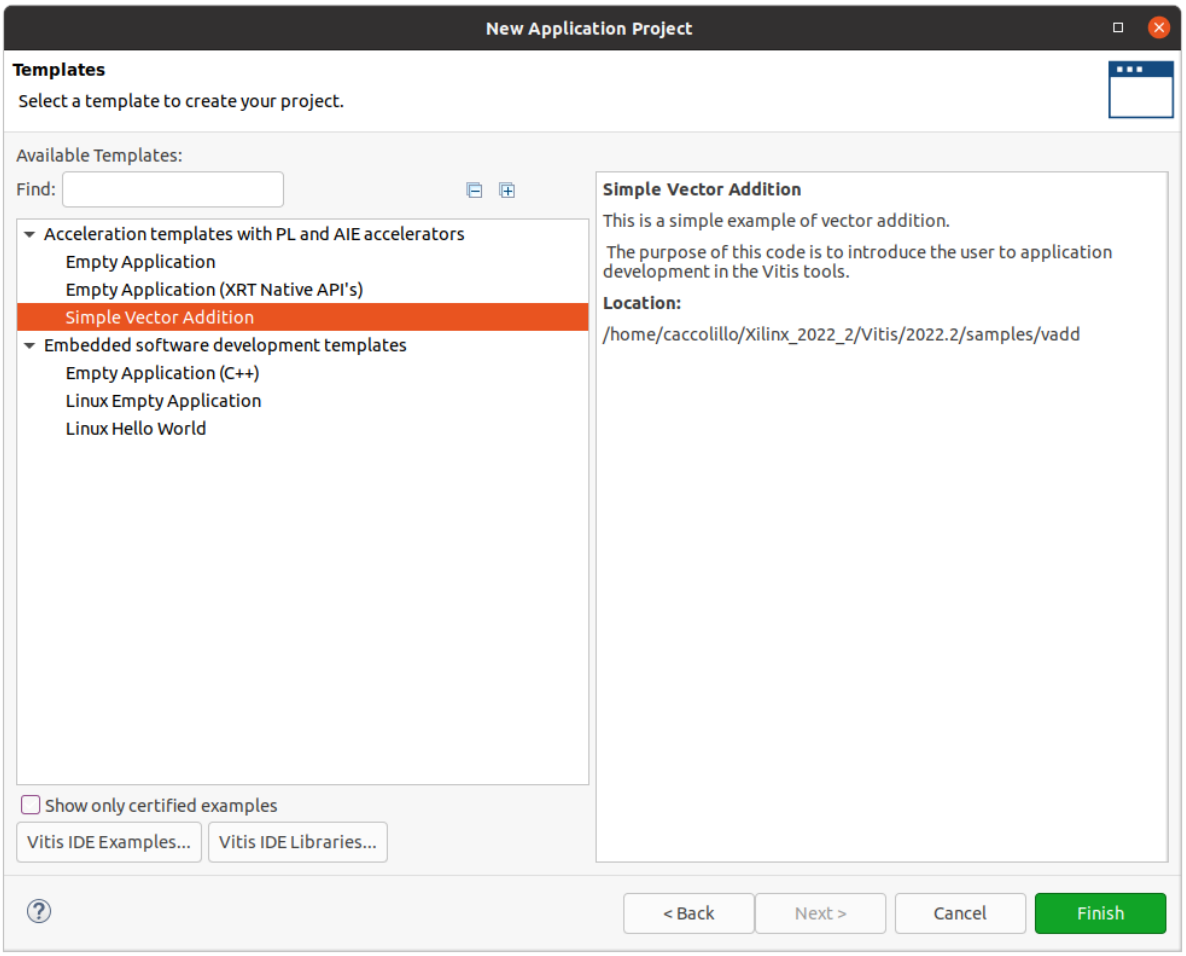

```
/include/ "system-conf.dtsi"
/{
  chosen {
     bootargs = "console=ttyPS0,115200 root=/dev/mmcblk0p2 rw earlyprintk
rootfstype=ext4 rw rootwait uio_pdrv_genirq.of_id=generic-uio devtmpfs.mount=1
earlycon";
   };
};

&sdhci0 {
no-1-8-v;
disable-wp;
};

&axi_intc_0 {
   xlnx,kind-of-intr = <0x0>;
   xlnx,num-intr-inputs = <0x20>;
};


&amba {
   zyxclmm_drm {
      compatible = "xlnx,zocl";
      status = "okay";
      interrupt-parent = <&axi_intc_0>;
      interrupts = <0  4>, <1  4>, <2  4>, <3  4>,
            <4  4>, <5  4>, <6  4>, <7  4>,
            <8  4>, <9  4>, <10 4>, <11 4>,
            <12 4>, <13 4>, <14 4>, <15 4>,
            <16 4>, <17 4>, <18 4>, <19 4>,
            <20 4>, <21 4>, <22 4>, <23 4>,
            <24 4>, <25 4>, <26 4>, <27 4>,
            <28 4>, <29 4>, <30 4>, <31 4>;
```

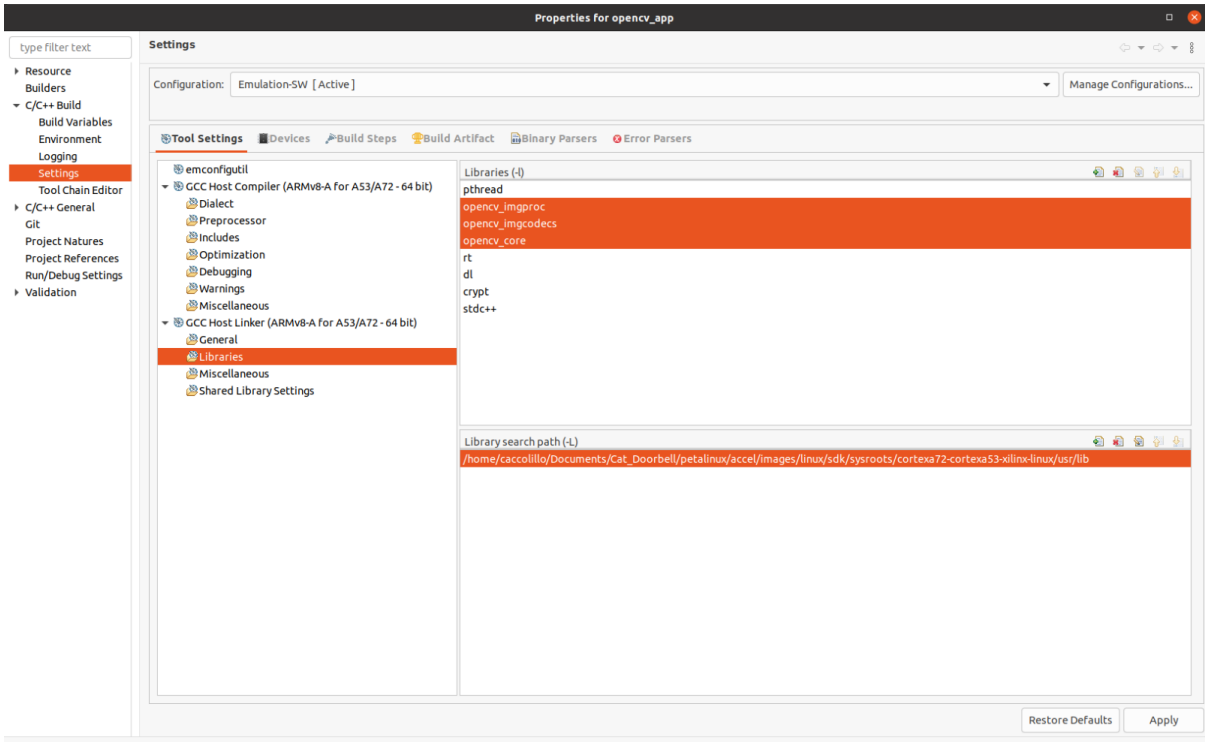
```
    };
};
```

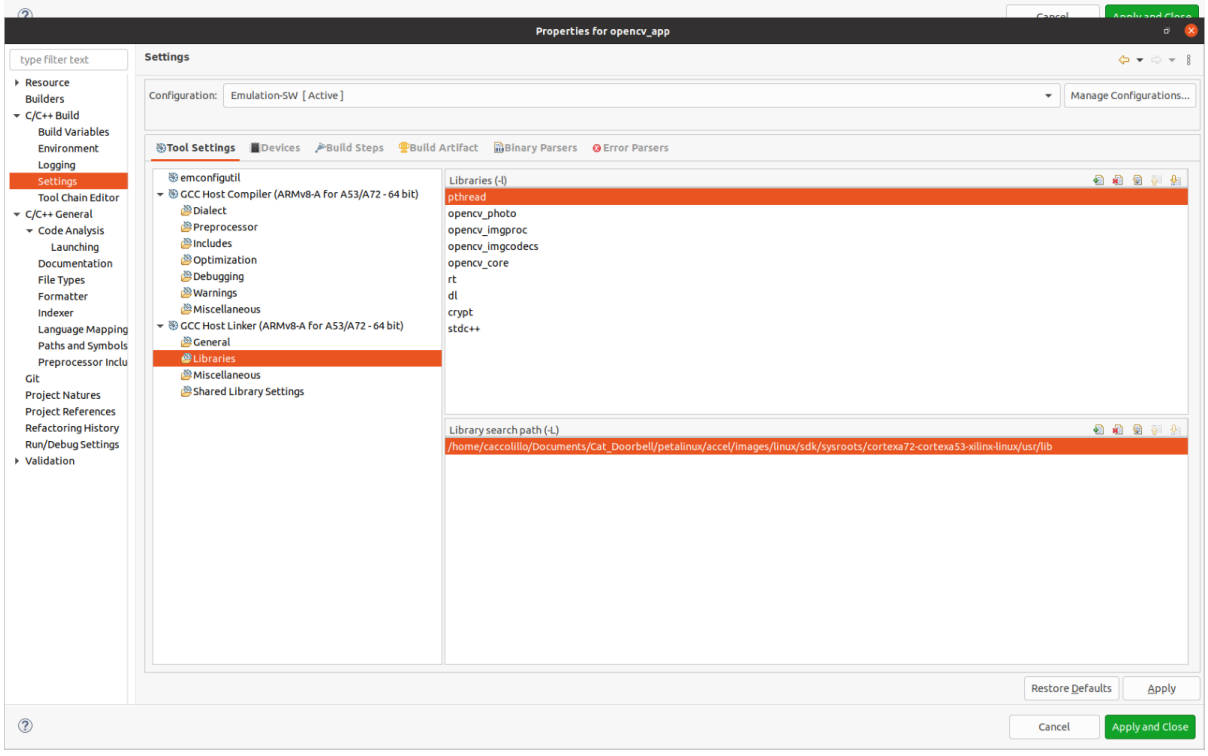
How to compile opencv apps in vitis and where to find the executable

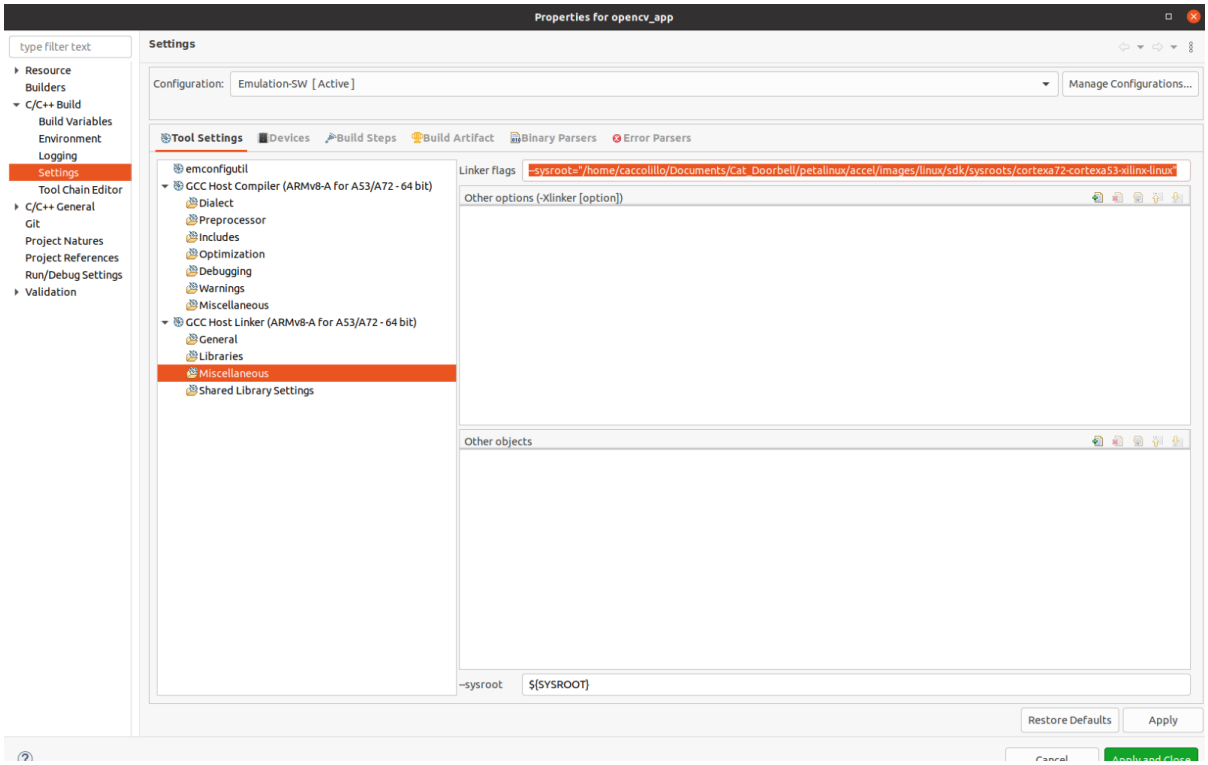


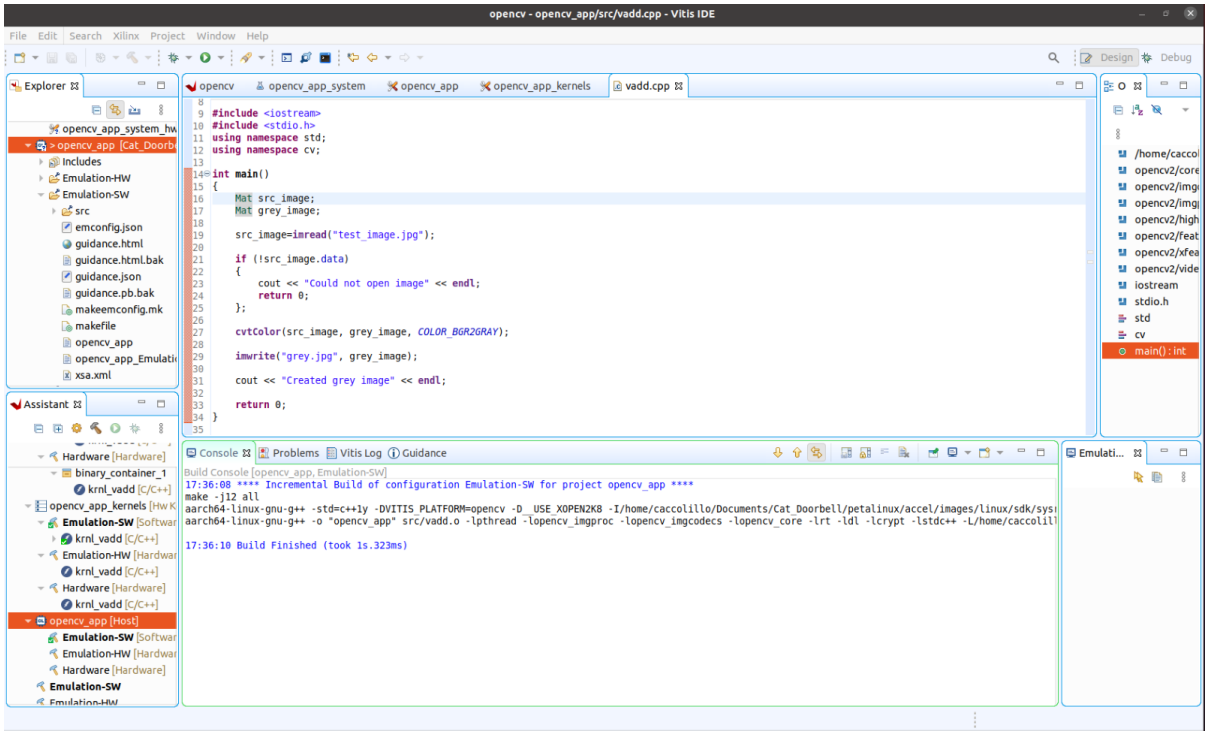

Properties for opencv_app
type filter text
Resource
Builders
C/C++ Build
Build Variables
Environment
Logging
Settings
Tool Chain Editor
C/C++ General
Git
Project Natures
Project References
Run/Debug Settings
Validation
Settings
Configuration: Emulation-SW [Active]
Manage Configurations...
Tool Settings
Devices
Build Steps
Build Artifact
Binary Parsers
Error Parsers
emconfigutil
GCC Host Compiler (ARMv8-A for A53/A72 - 64 bit)
Dialect
Preprocessor
Includes
Optimization
Debugging
Warnings
Miscellaneous
GCC Host Linker (ARMv8-A for A53/A72 - 64 bit)
General
Libraries
Miscellaneous
Shared Library Settings
Include paths (-I)
/home/caccolillo/Documents/Cat_Doorbell/petalinux/accel/images/linux/sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/include/opencv4
Include files (-include)
Restore Defaults
Apply

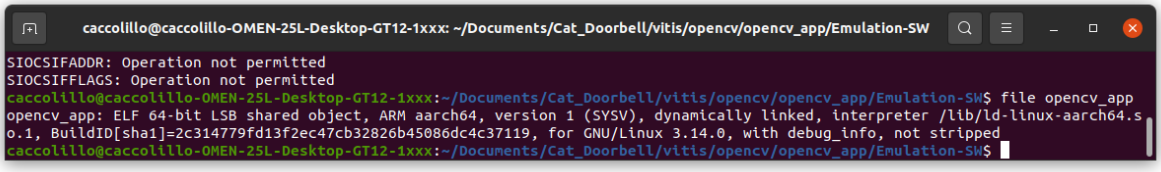

New Application Project
Templates
Select a template to create your project.
Available Templates:
Find:
Acceleration templates with PL and AIE accelerators
Empty Application
Empty Application (XRT Native API's)
Simple Vector Addition
Embedded software development templates
Empty Application (C++)
Linux Empty Application
Linux Hello World
Simple Vector Addition
This is a simple example of vector addition.
The purpose of this code is to introduce the user to application development in the Vitis tools.
Location:
/home/caccolillo/Xilinx_2022_2/Vitis/2022.2/samples/vadd
Show only certified examples
Vitis IDE Examples...
Vitis IDE Libraries...
< Back
Next >
Cancel
Finish


Properties for opencv_app
type filter text
Resource
Builders
C/C++ Build
Build Variables
Environment
Logging
Settings
Tool Chain Editor
C/C++ General
Git
Project Natures
Project References
Run/Debug Settings
Validation
Settings
Configuration: Emulation-SW [Active]
Manage Configurations...
Tool Settings
Devices
Build Steps
Build Artifact
Binary Parsers
Error Parsers
emconfigutil
GCC Host Compiler (ARMv8-A for A53/A72 - 64 bit)
Dialect
Preprocessor
Includes
Optimization
Debugging
Warnings
Miscellaneous
GCC Host Linker (ARMv8-A for A53/A72 - 64 bit)
General
Libraries
Miscellaneous
Shared Library Settings
Libraries (-l)
pthread
opencv_imgproc
opencv_imgcodecs
opencv_core
rt
dl
crypt
stdc++
Library search path (-L)
/home/caccolillo/Documents/Cat_Doorbell/petalinux/accel/images/linux/sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib
Restore Defaults
Apply


Properties for opencv_app
type filter text
Resource
Builders
C/C++ Build
Build Variables
Environment
Logging
Settings
Tool Chain Editor
C/C++ General
Code Analysis
Launching
Documentation
File Types
Formatter
Indexer
Language Mapping
Paths and Symbols
Preprocessor Inclu
Git
Project Natures
Project References
Refactoring History
Run/Debug Settings
Validation
Settings
Configuration: Emulation-SW [Active]
Manage Configurations...
Tool Settings
Devices
Build Steps
Build Artifact
Binary Parsers
Error Parsers
emconfigutil
GCC Host Compiler (ARMv8-A for A53/A72 - 64 bit)
Dialect
Preprocessor
Includes
Optimization
Debugging
Warnings
Miscellaneous
GCC Host Linker (ARMv8-A for A53/A72 - 64 bit)
General
Libraries
Miscellaneous
Shared Library Settings
Libraries (-l)
pthread
opencv_photo
opencv_imgproc
opencv_imgcodecs
opencv_core
rt
dl
crypt
stdc++
Library search path (-L)
/home/caccolillo/Documents/Cat_Doorbell/petalinux/accel/images/linux/sdk/sysroots/cortexa72-cortexa53-xilinx-linux/usr/lib
Restore Defaults
Apply
Cancel
Apply and Close

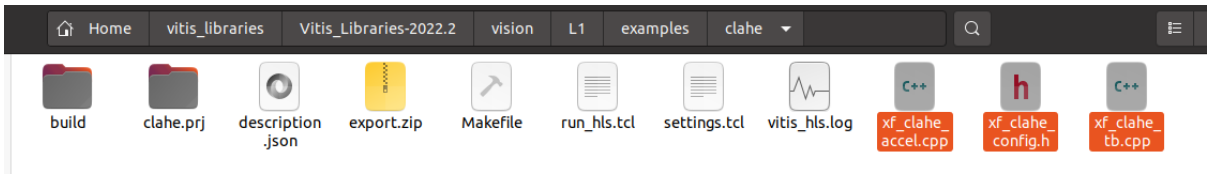

Properties for opencv_app
type filter text
Resource
Builders
C/C++ Build
Build Variables
Environment
Logging
Settings
Tool Chain Editor
C/C++ General
Git
Project Natures
Project References
Run/Debug Settings
Validation
Settings
Configuration: Emulation-SW [Active]
Manage Configurations...
Tool Settings
Devices
Build Steps
Build Artifact
Binary Parsers
Error Parsers
emconfigutil
GCC Host Compiler (ARMv8-A for A53/A72 - 64 bit)
Dialect
Preprocessor
Includes
Optimization
Debugging
Warnings
Miscellaneous
GCC Host Linker (ARMv8-A for A53/A72 - 64 bit)
General
Libraries
Miscellaneous
Shared Library Settings
Linker flags
--sysroot="/home/caccolillo/Documents/Cat_Doorbell/petalinux/accel/images/linux/sdk/sysroots/cortexa72-cortexa53-xilinx-linux"
Other options (-Xlinker [option])
Other objects
--sysroot
${SYSROOT}
Restore Defaults
Apply
Cancel
Apply and Close


opencv - opencv_app/src/vadd.cpp - Vitis IDE
File Edit Search Xilinx Project Window Help
Design
Debug
Explorer
opencv_app_system_hw
opencv_app [Cat_Doorb
Includes
Emulation-HW
Emulation-SW
src
emconfig.json
guidance.html
guidance.html.bak
guidance.json
guidance.pb.bak
makeemconfig.mk
makefile
opencv_app
opencv_app_Emulati
xsa.xml
Assistant
Hardware [Hardware]
binary_container_1
krnl_vadd [C/C++]
opencv_app_kernels [Hw K
Emulation-SW [Softwar
krnl_vadd [C/C++]
Emulation-HW [Hardwar
krnl_vadd [C/C++]
Hardware [Hardware]
krnl_vadd [C/C++]
opencv_app [Host]
Emulation-SW [Softwar
Emulation-HW [Hardwar
Hardware [Hardware]
Emulation-SW
opencv
opencv_app_system
opencv_app
opencv_app_kernels
vadd.cpp
#include <iostream>
#include <stdio.h>
using namespace std;
using namespace cv;
int main()
{
Mat src_image;
Mat grey_image;
src_image=imread("test_image.jpg");
if (!src_image.data)
{
cout << "Could not open image" << endl;
return 0;
};
cvtColor(src_image, grey_image, COLOR_BGR2GRAY);
imwrite("grey.jpg", grey_image);
cout << "Created grey image" << endl;
return 0;
}
/home/caccol
opencv2/core
opencv2/imgc
opencv2/imgc
opencv2/high
opencv2/feat
opencv2/xfea
opencv2/vide
iostream
stdio.h
std
cv
main() : int
Console
Problems
Vitis Log
Guidance
Build Console [opencv_app, Emulation-SW]
17:36:08 **** Incremental Build of configuration Emulation-SW for project opencv_app ****
make -j12 all
aarch64-linux-gnu-g++ -std=c++1y -DVITIS_PLATFORM=opencv -D__USE_XOPEN2K8 -I/home/caccolillo/Documents/Cat_Doorbell/petalinux/accel/images/linux/sdk/sysr
aarch64-linux-gnu-g++ -o "opencv_app" src/vadd.o -lpthread -lopencv_imgproc -lopencv_imgcodecs -lopencv_core -lrt -ldl -lcrypt -lstdc++ -L/home/caccolill
17:36:10 Build Finished (took 1s.323ms)
Emulati...


caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx: ~/Documents/Cat_Doorbell/vitis/opencv/opencv_app/Emulation-SW
SIOCSIFADDR: Operation not permitted
SIOCSIFFLAGS: Operation not permitted
caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx:~/Documents/Cat_Doorbell/vitis/opencv/opencv_app/Emulation-SW$ file opencv_app
opencv_app: ELF 64-bit LSB shared object, ARM aarch64, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-aarch64.so.1, BuildID[sha1]=2c314779fd13f2ec47cb32826b45086dc4c37119, for GNU/Linux 3.14.0, with debug_info, not stripped
caccolillo@caccolillo-OMEN-25L-Desktop-GT12-1xxx:~/Documents/Cat_Doorbell/vitis/opencv/opencv_app/Emulation-SW$

CLAHE FROM VITIS VISION LIBRARIES L1 USAGE UNDER OPENCL

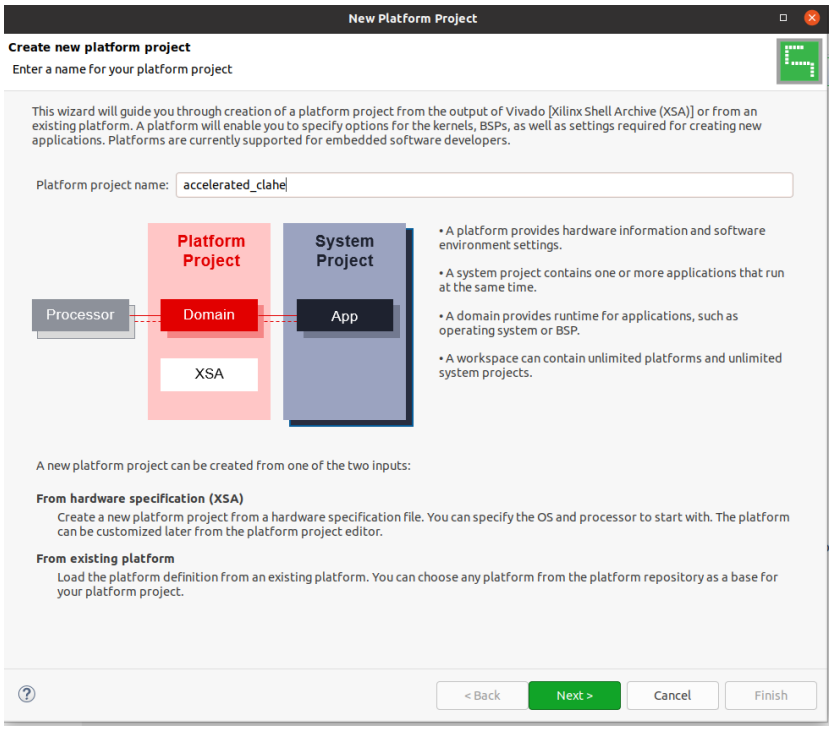
After having cloned the vitis vision libaries, in the example folder there is the CLAHE example:

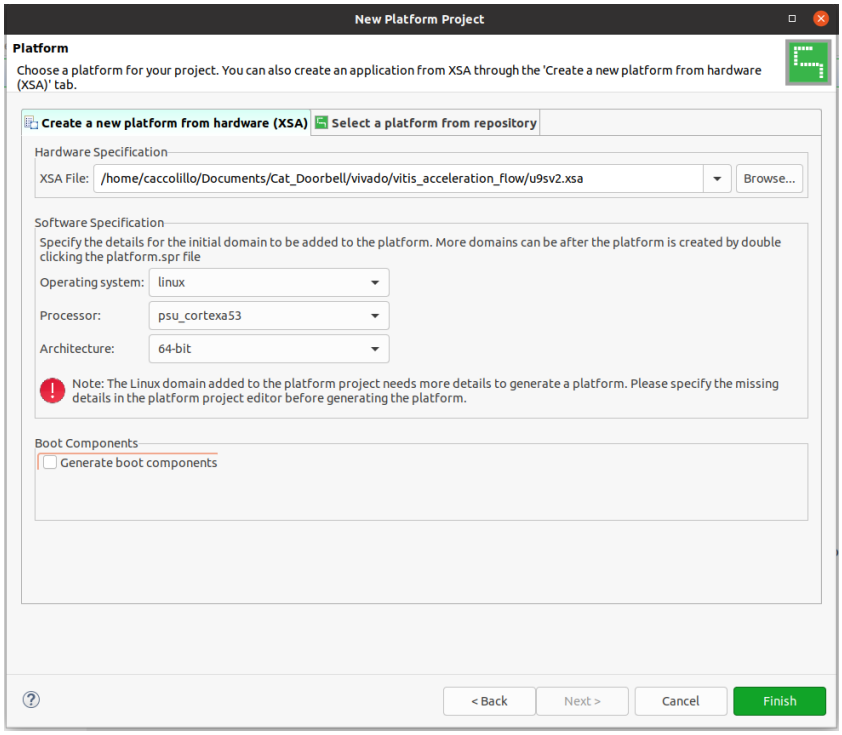


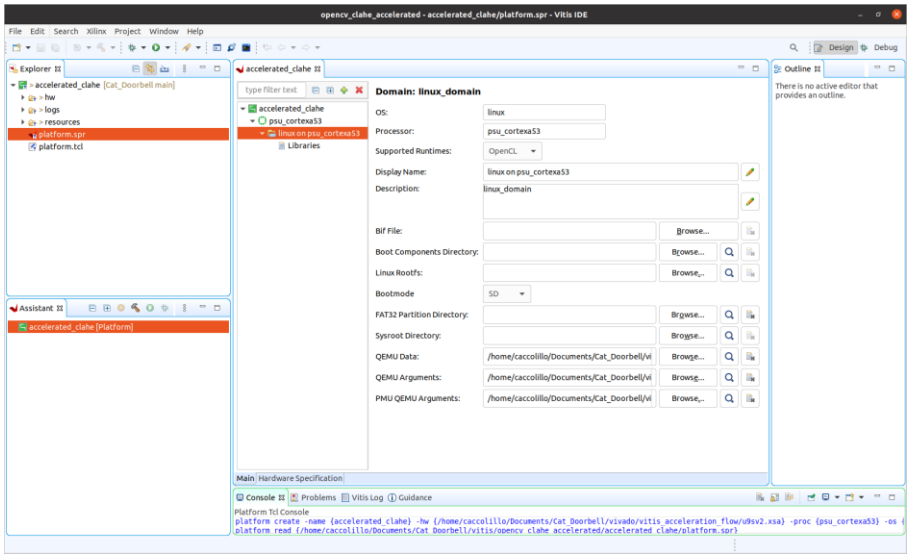
The testbench can be used in vivado to exercise the accelerated kernel in vitis HLS.

It would be nice using the code from within Vitis, so a Vitis project based upon the Petalinux distribution described insofar has been created, using the vector addition as a starting point.

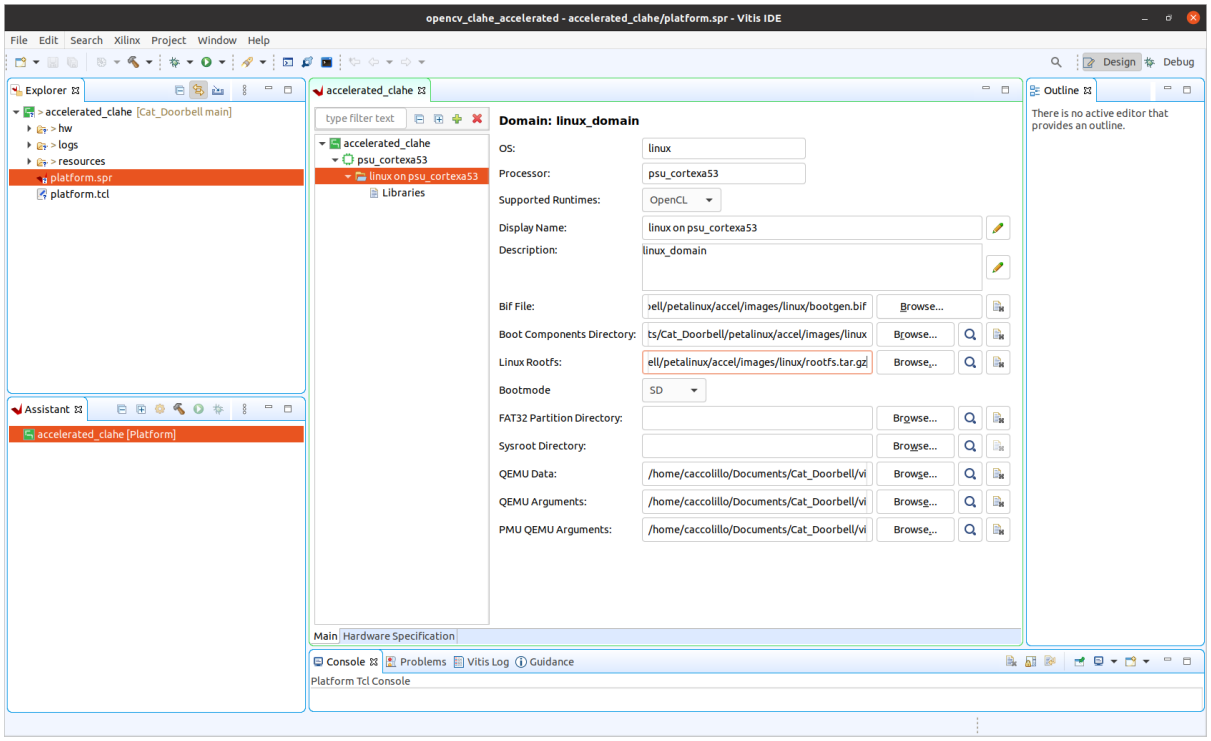
Create a new platform project, specifying the xsa file:



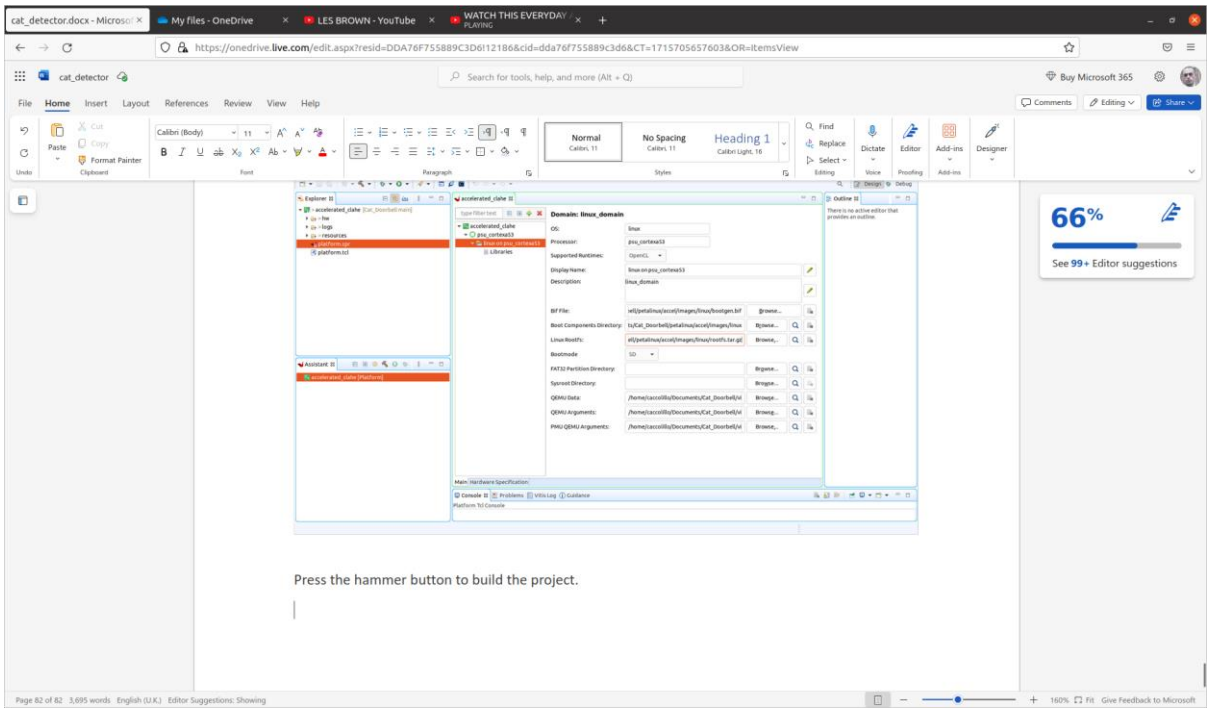




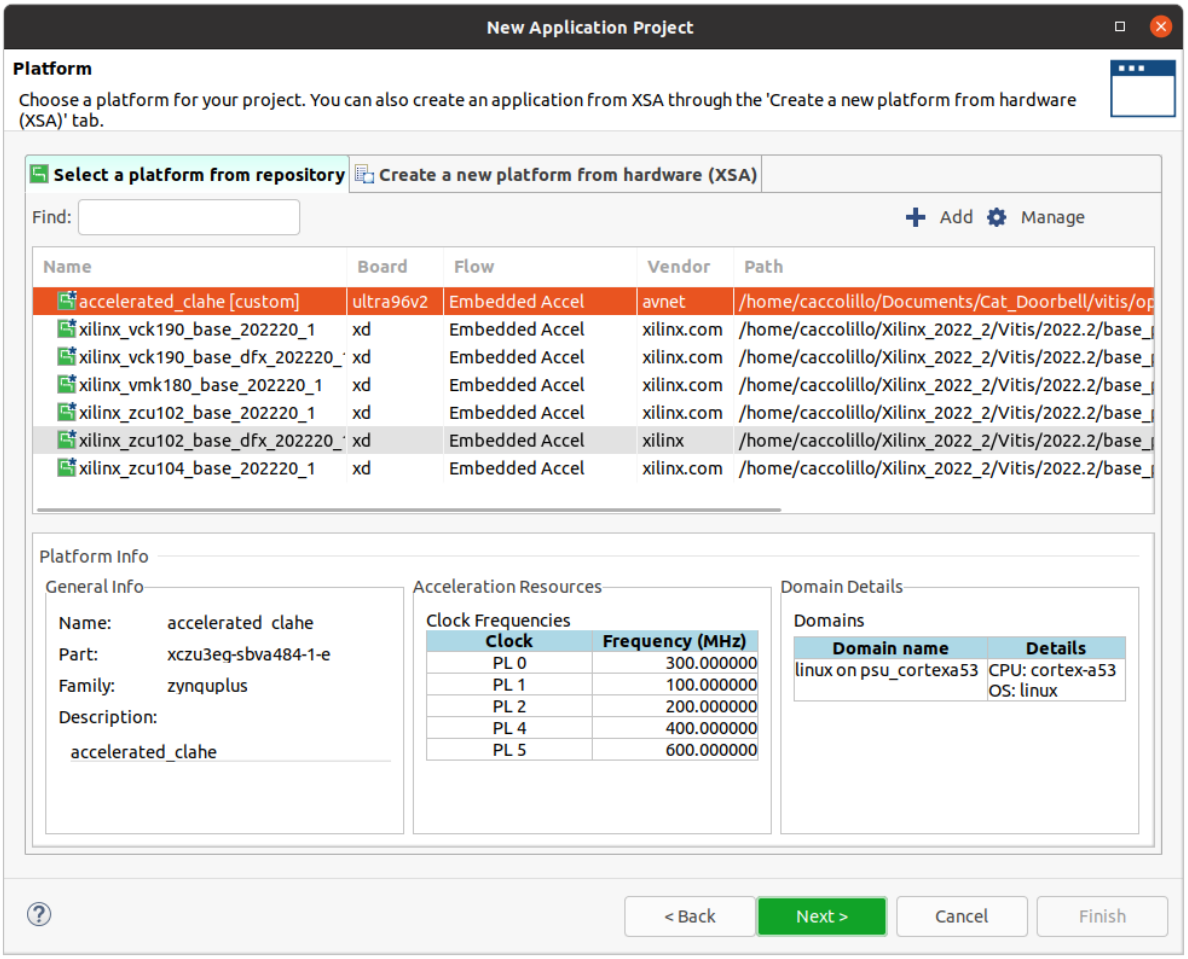
Point the bif file, boot components directory, and linux rootfs a s specified below:

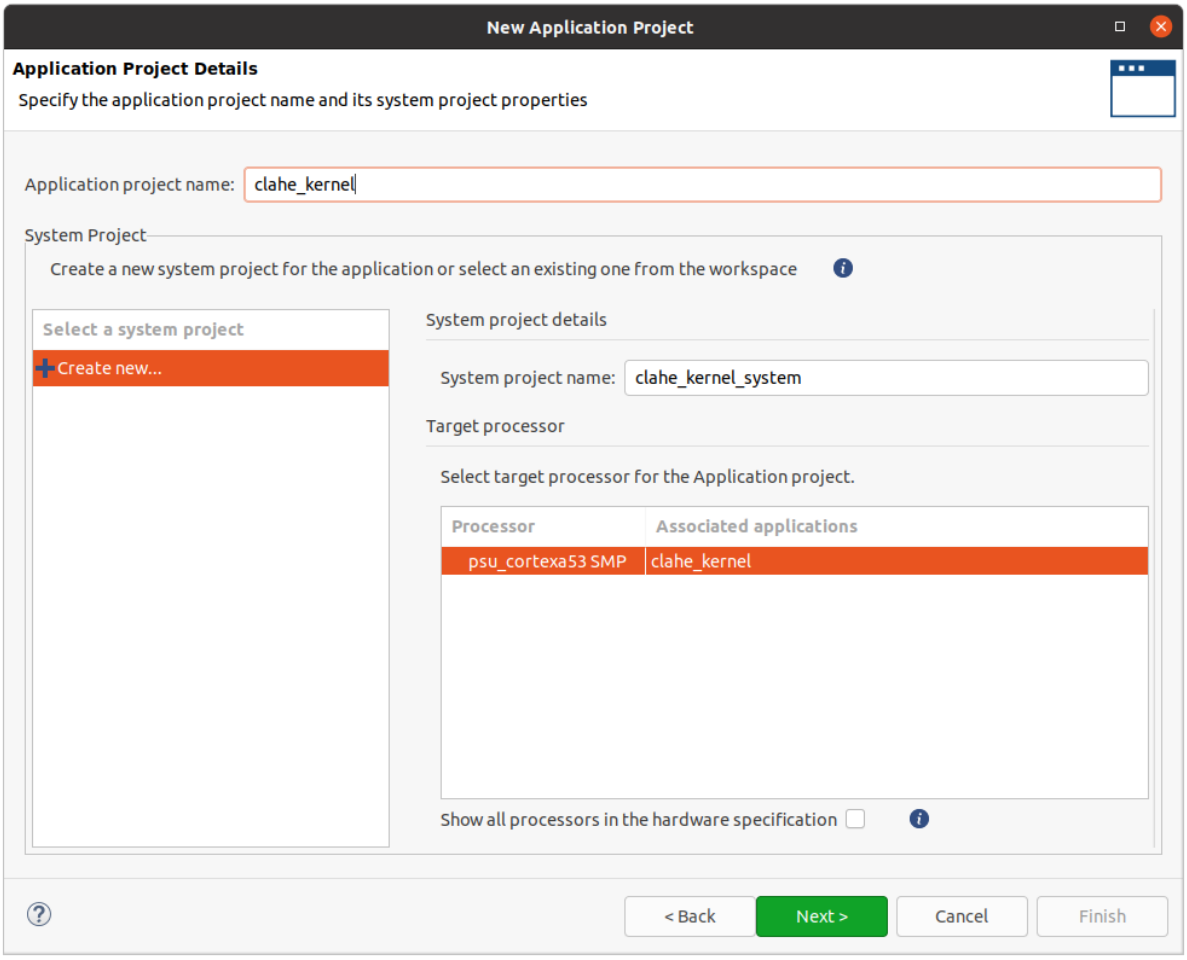


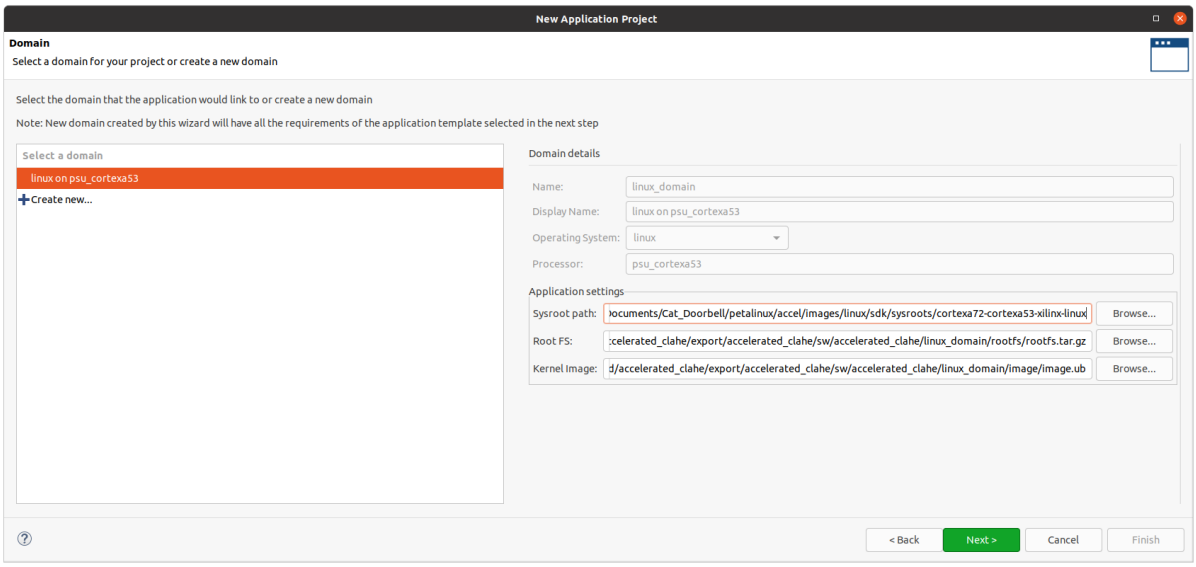
Press the hammer button to build the project.



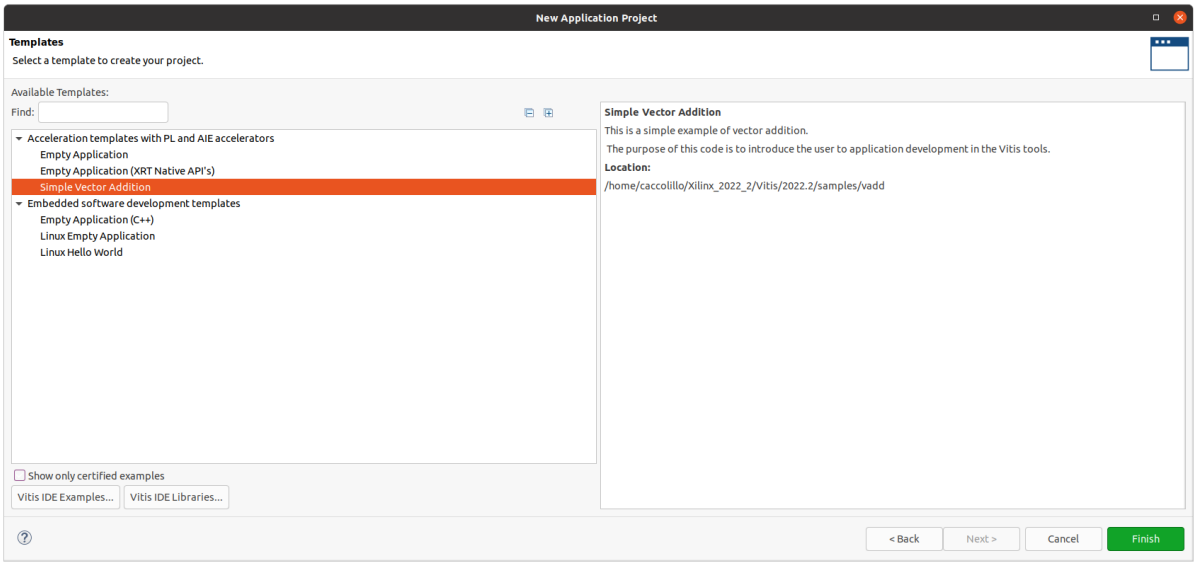
Then create a new application project, and select as platform repository the one just created:

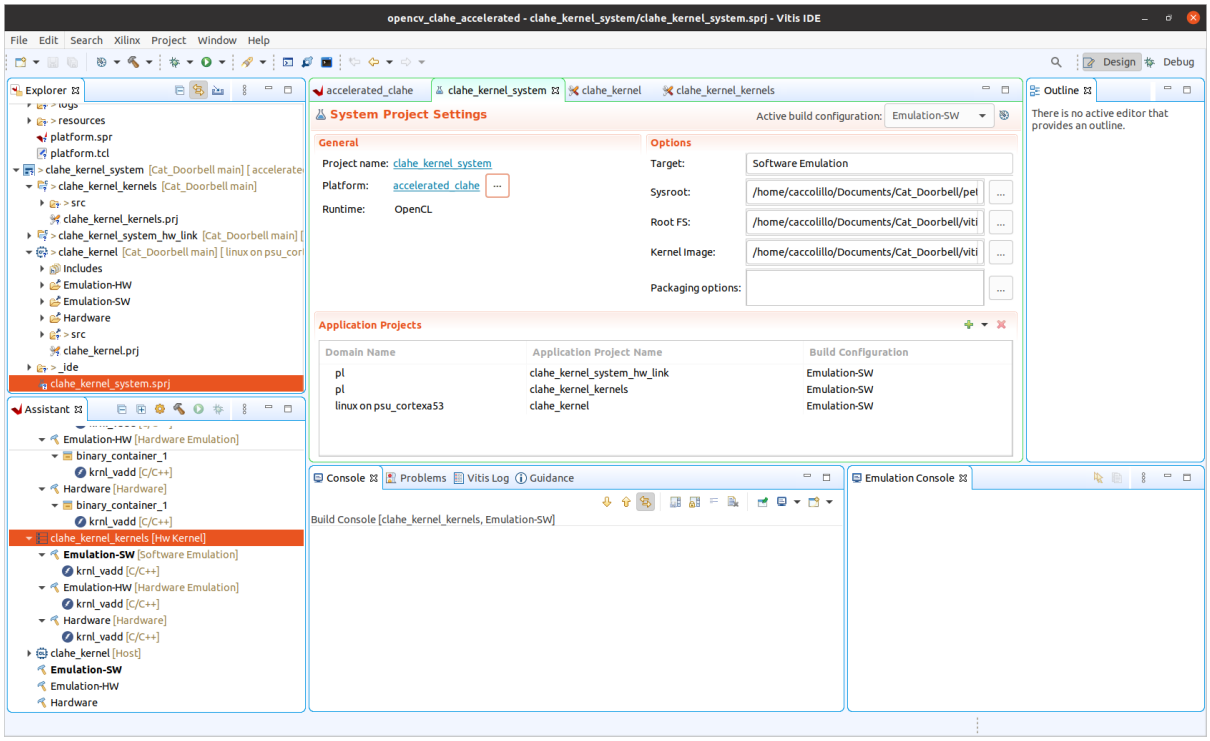

New Application Project
Platform
Choose a platform for your project. You can also create an application from XSA through the 'Create a new platform from hardware (XSA)' tab.
Select a platform from repository
Create a new platform from hardware (XSA)
Find:
Add
Manage
Name
Board
Flow
Vendor
Path
accelerated_clahe [custom]
ultra96v2
Embedded Accel
avnet
/home/caccolillo/Documents/Cat_Doorbell/vitis/op
xilinx_vck190_base_202220_1
xd
Embedded Accel
xilinx.com
/home/caccolillo/Xilinx_2022_2/Vitis/2022.2/base_
xilinx_vck190_base_dfx_202220_
xd
Embedded Accel
xilinx.com
/home/caccolillo/Xilinx_2022_2/Vitis/2022.2/base_
xilinx_vmk180_base_202220_1
xd
Embedded Accel
xilinx.com
/home/caccolillo/Xilinx_2022_2/Vitis/2022.2/base_
xilinx_zcu102_base_202220_1
xd
Embedded Accel
xilinx.com
/home/caccolillo/Xilinx_2022_2/Vitis/2022.2/base_
xilinx_zcu102_base_dfx_202220_
xd
Embedded Accel
xilinx
/home/caccolillo/Xilinx_2022_2/Vitis/2022.2/base_
xilinx_zcu104_base_202220_1
xd
Embedded Accel
xilinx.com
/home/caccolillo/Xilinx_2022_2/Vitis/2022.2/base_
Platform Info
General Info
Name: accelerated_clahe
Part: xczu3eg-sbva484-1-e
Family: zynquplus
Description:
accelerated_clahe
Acceleration Resources
Clock Frequencies
Clock
Frequency (MHz)
PL 0
300.000000
PL 1
100.000000
PL 2
200.000000
PL 4
400.000000
PL 5
600.000000
Domain Details
Domains
Domain name
Details
linux on psu_cortexa53
CPU: cortex-a53
OS: linux
< Back
Next >
Cancel
Finish





Select the simple vector addition template:





As a safety check press the hammer to build the project: if everything has been set right, it should build successfully.

Bear in mind that good tutorials on the subject are available at:

https://www.hackster.io/whitney-knitter/vitis-acceleration-flow-on-kv260-extensible-vitis-platform-c3f947

https://www.hackster.io/whitney-knitter/vitis-acceleration-flow-on-kv260-software-in-petalinux-b2456c

https://www.hackster.io/whitney-knitter/vitis-acceleration-flow-on-kv260-vitis-platform-c3537e

Other excellent tuorials from Xilinx are available at:

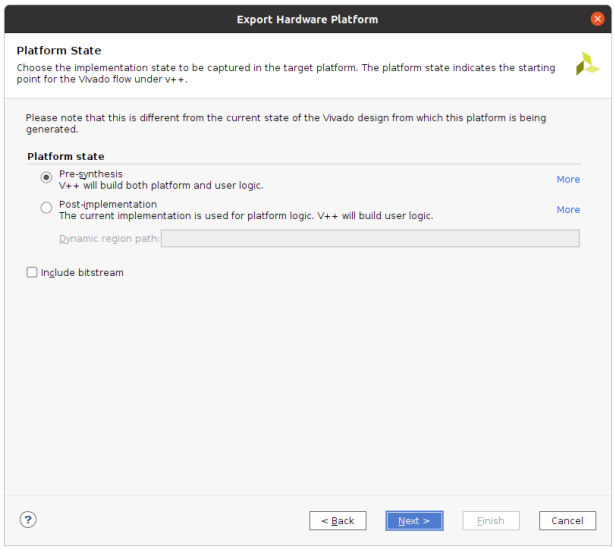
https://support.xilinx.com/s/article/1138208?language=en_US

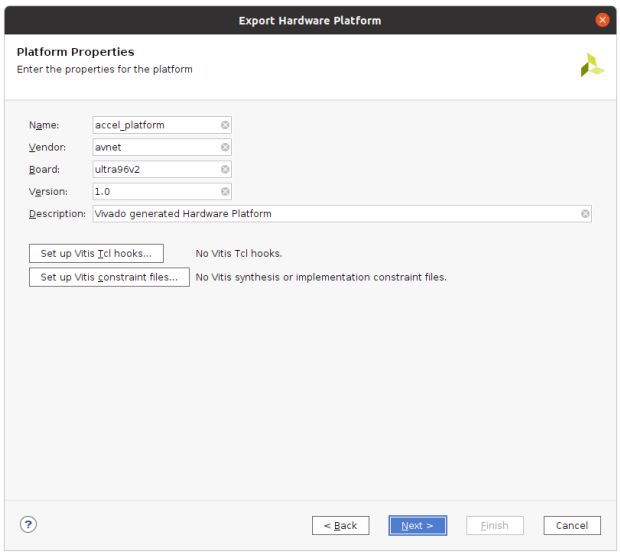
https://support.xilinx.com/s/question/0D52E00007IPgTmSAL/creating-an-acceleration-platform-for-vitis-part-two-creating-the-software-project-for-the-acceleration-platform-in-petalinux?language=en_US

https://support.xilinx.com/s/article/1138750?language=en_US

https://support.xilinx.com/s/article/1138865?language=en_US

Be careful when exporting the hardware (in “export platform” -> “export hardware platform”-> “hardware”-> “pre-synthesis” with “include bitstream deselected”) of adding a valid name in the gui:





Otherwise the Vitis IDE will error out.

A tutorial about how to use the debugger is at:

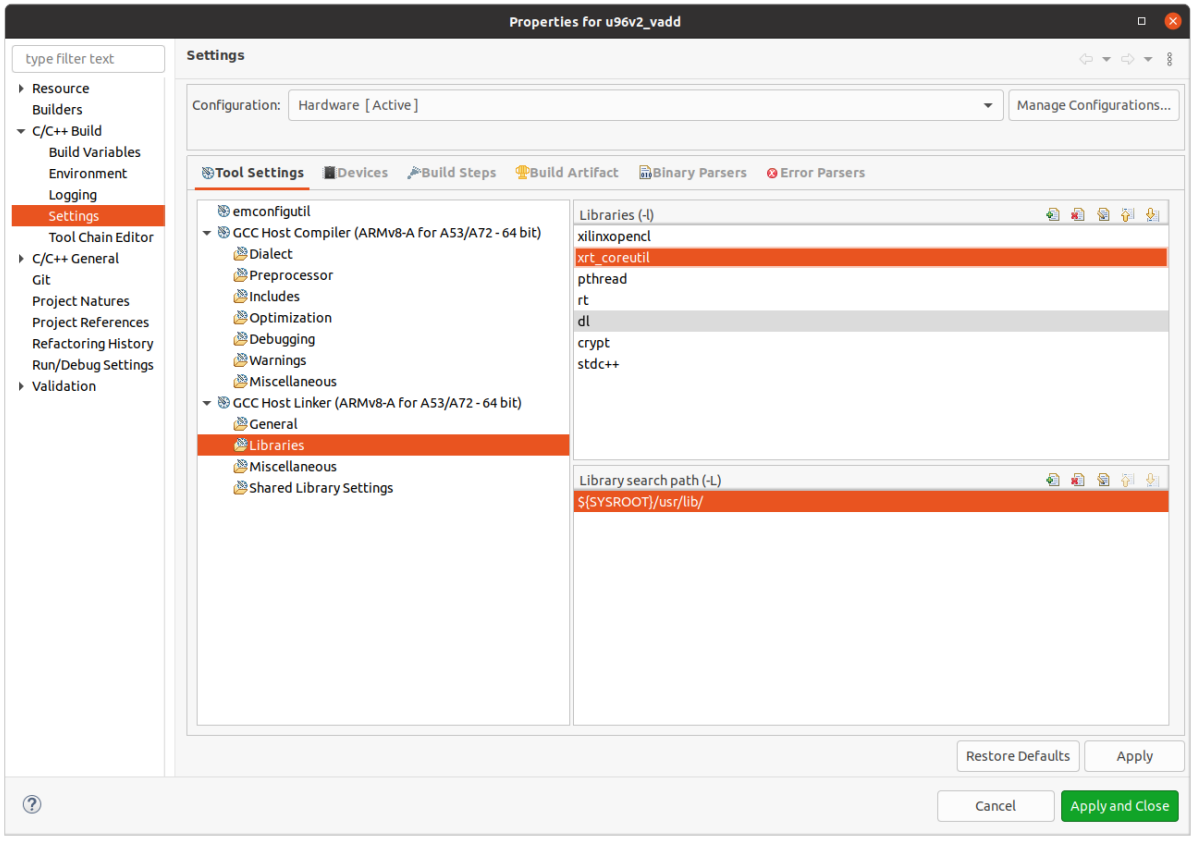
https://www.hackster.io/whitney-knitter/debugging-accelerated-apps-using-emulation-in-vitis-2021-2-749cd4

In the end, the tutorial to follow to get all sorted out in Vitis is:

https://www.hackster.io/whitney-knitter/accelerated-design-development-on-kria-kr260-in-vitis-2022-1-883799

At least for the Vitis 2022.2 version.

In order to use xrt, add xrt_coreutil library in vitis for the host code:



In the end this DOES NOT WORK.

To get it working, the tutorial design below has been followed:

https://www.hackster.io/AlbertaBeef/ultra96-v2-building-the-foundational-designs-e43155

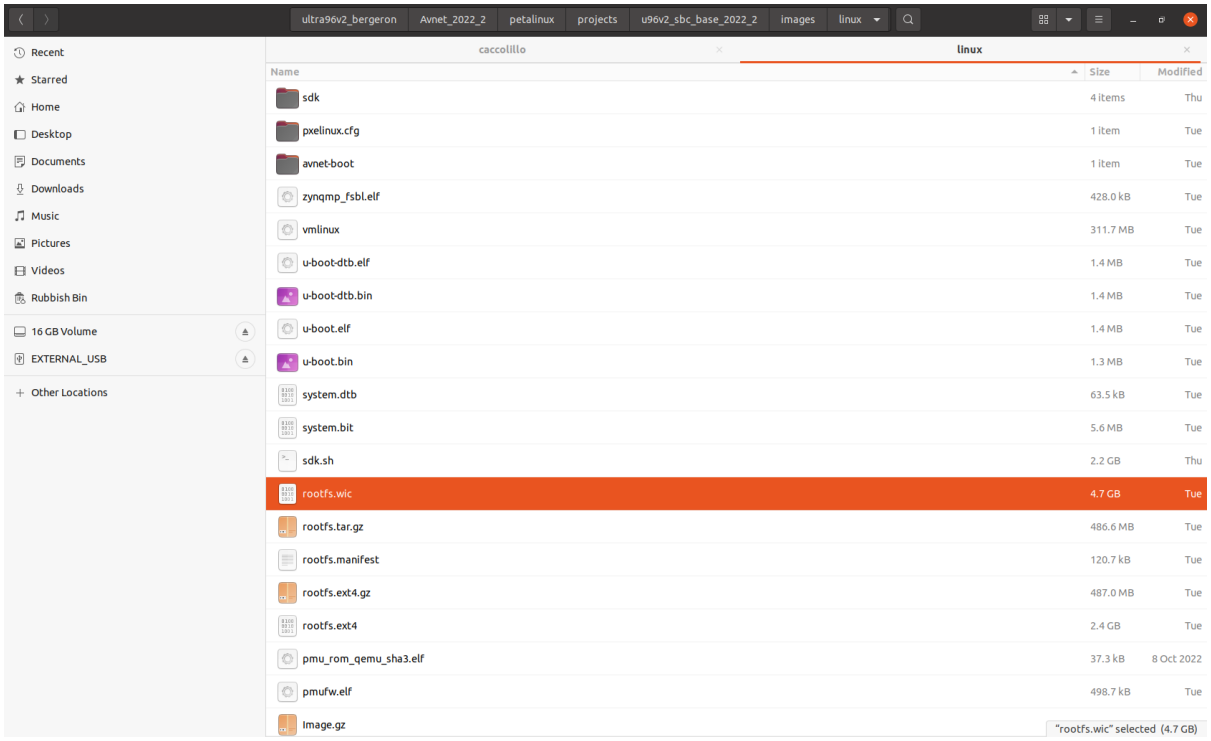
Creating the u96v2_sbc_base design.

Then the following tutorial:

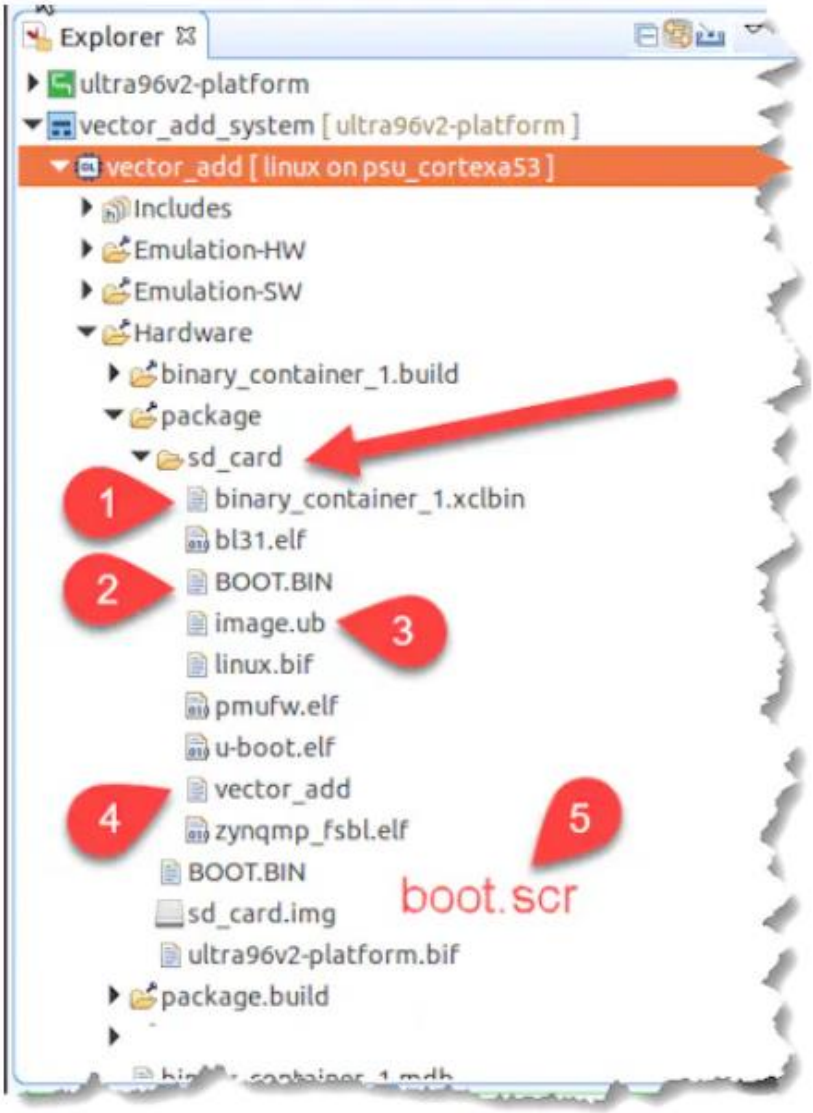
https://highlevel-synthesis.com/2021/08/14/how-to-create-ultra96v2-linux-based-platform-in-xilinx-vitis-2020-2/

Has been used to get the Vitis project, but the procedure for the bif was wrong, so the option in Vitis to let the tool to create the bif has been used.

The micro SD has been created from the WIC file generated by Bergeron tutorial and burned in Windows with Rufus:

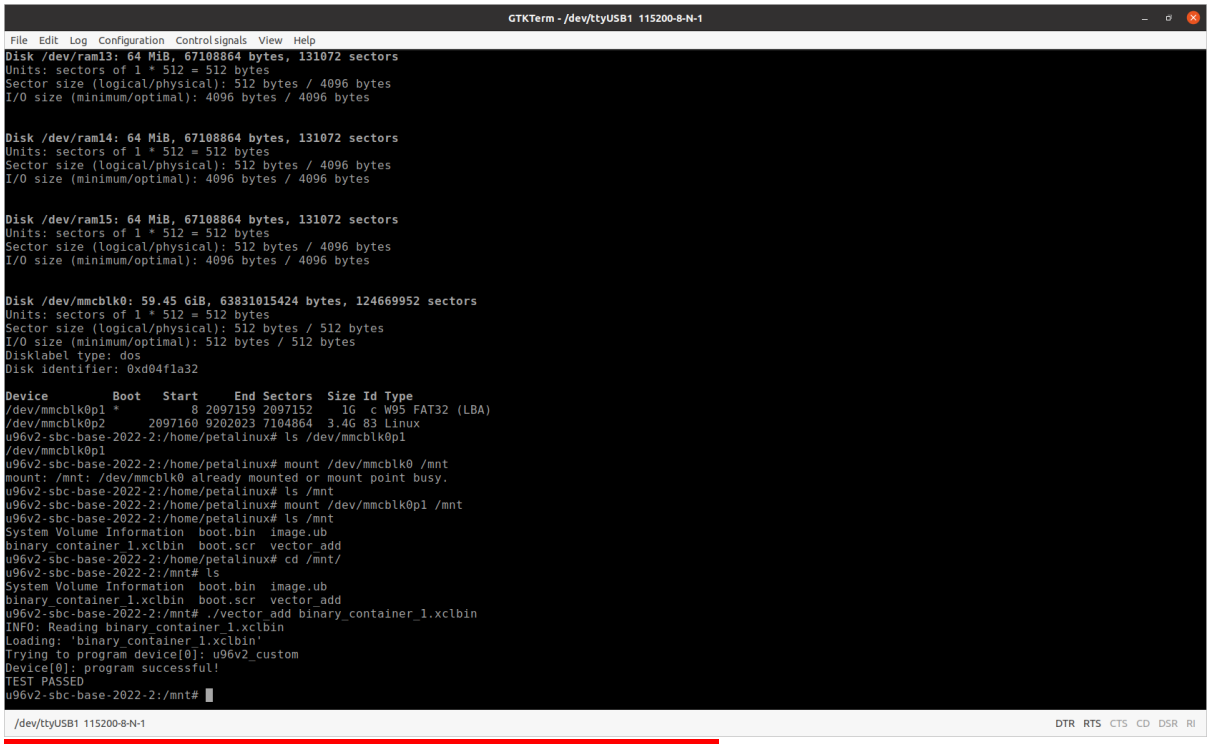


Then the files specified in the tutorial have been copied in the BOOT partition ():



Boot.scr was the one already in the SD-card, as it hasn't been generated.

Once the board has booted, giving sudo su, then fdisk –l:

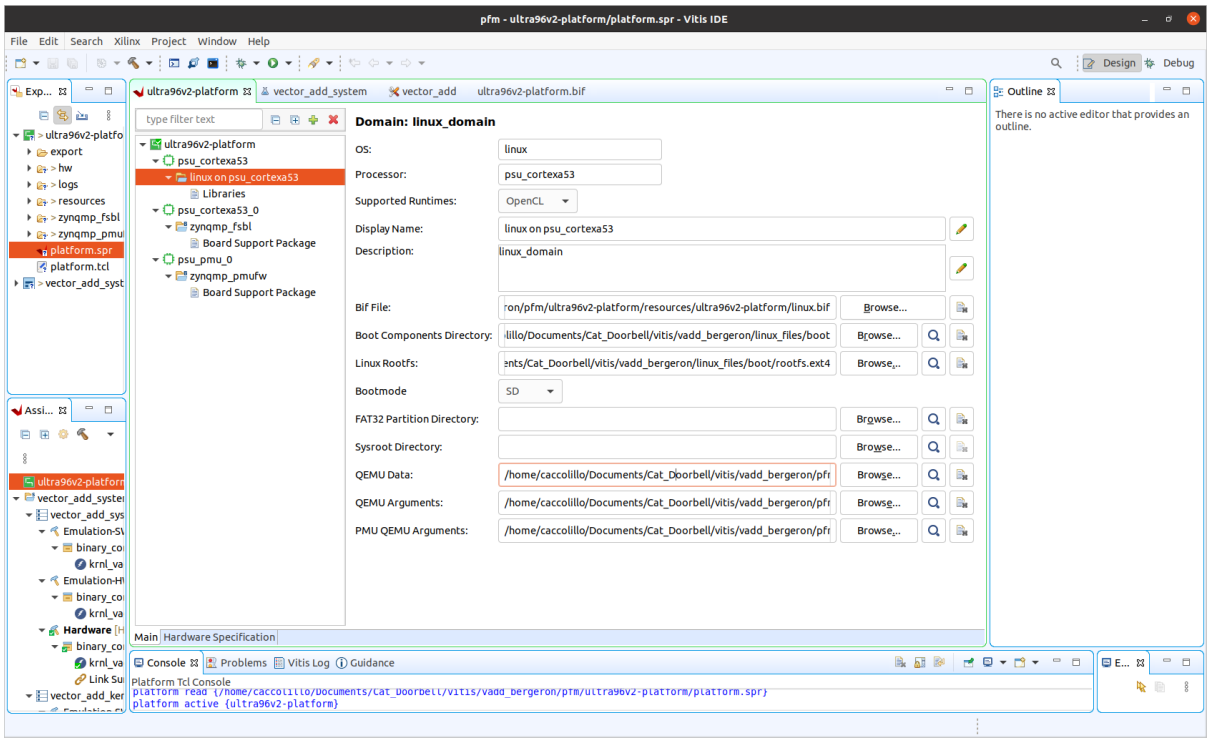


Run the following commands to define the Xilinx XRT library

export XILINX_XRT=/usr

And the program works!

In the following screenshots about the configuration in Vitis:



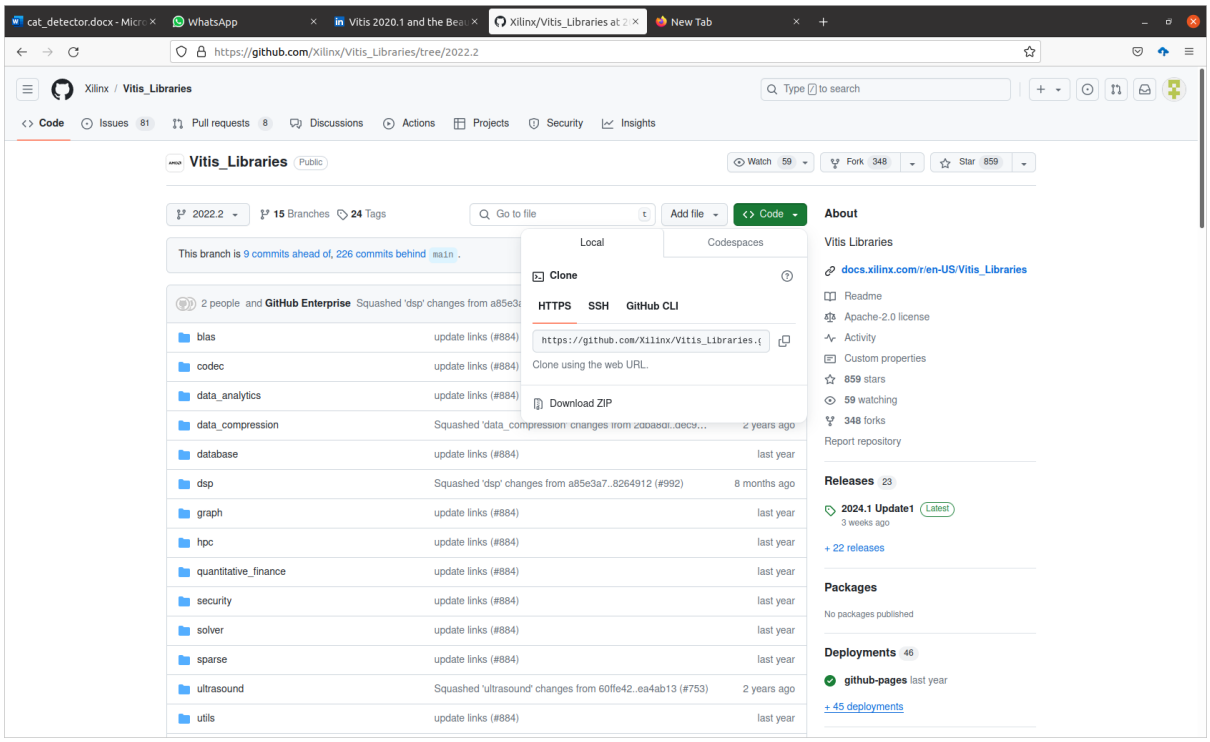
AUTOWHITE BALANCE FROM VITIS VISION LIBRARIES L2 USAGE UNDER OPENCL

As per tutorial below, an attempt to install Vitis Vision libraries in Vitis has ben done:

https://www.linkedin.com/pulse/vitis-20201-beauty-acceleration-martin-kellermann/

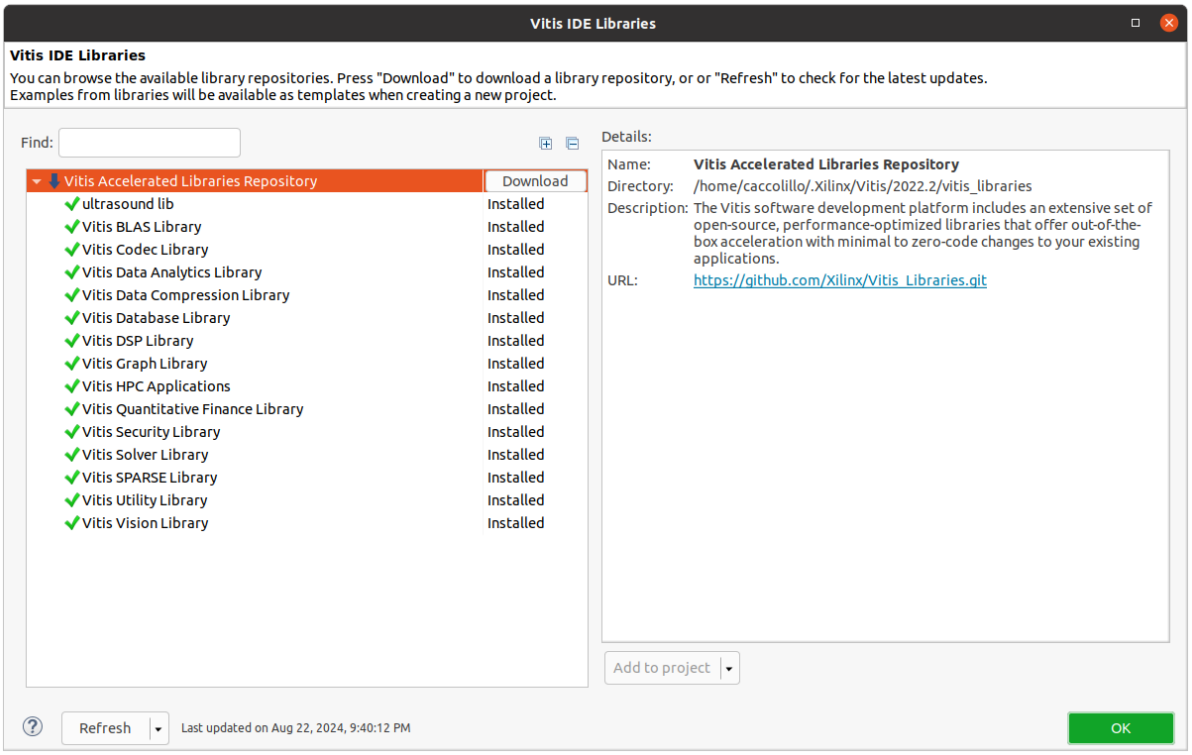
But due to internet connections, teh git cloning wasn't working.

So an archive of the libraries has been downloaded from git:



The archive gets extracted in the folder below:

Once done, they appear in Vitis:



The CLAHE example has been attempted but it wasn't working.

The auto white balance has worked and has been implemented and tested on hardware.

VITIS AI

The compatibility list of Vitis AI releases follows:

https://xilinx.github.io/Vitis-AI/3.0/html/docs/reference/version_compatibility.html

Current Release

Vitis AI v3.0 and the DPU IP released with the v3.0 branch of this repository are verified as compatible with Vitis, Vivado™, and PetaLinux version 2022.2. If you are using a previous release Vitis AI, please refer to the table below release.

Previous Releases

Software Tools Version	DPUCZDX8G IP Version	Vitis AI Release Version
Vivado / Vitis / PetaLinux 2022.2	4.1	v3.0
Vivado / Vitis / PetaLinux 2022.1	4.0	v2.5
Vivado / Vitis / PetaLinux 2021.2	3.4	v2.0
Vivado / Vitis / PetaLinux 2021.1	3.3	v1.4
Vivado / Vitis / PetaLinux 2020.2	3.3	v1.3
Vivado / Vitis / PetaLinux 2020.1	3.2	v1.2
Vivado / Vitis / PetaLinux 2019.2	3.2	v1.1
Vivado / Vitis / PetaLinux 2019.1	3.1	v1.0
Vivado / PetaLinux 2019.1	3.0	N/A
Vivado / PetaLinux 2018.2	2.0	N/A
Vivado / PetaLinux 2018.1	1.0	First Release

Software Tools Version	DPUCVDX8G IP Version	Vitis AI Release Version
Vitis / PetaLinux 2022.2	1.3	v3.0
Vitis / PetaLinux 2022.1	1.2	v2.5
Vitis / PetaLinux 2021.2	1.1	v2.0
Vitis / PetaLinux 2021.1	1.0	v1.4
Vitis / PetaLinux 2020.2	Early Access	v1.3

Software Tools Version	DPUCV2DX8G IP Version	Vitis AI Release Version
Vitis / PetaLinux 2022.2	Early Access	v3.0

Docker has to be installed as per:

https://docs.docker.com/engine/install/ubuntu/

Instructions to use a readymade micro SD from Avnet are at:

https://www.hackster.io/AlbertaBeef/ultra96-v2-adding-support-for-vitis-ai-3-0-704799

Instead step by step tutorials are at:

https://octavosystems.com/app_notes/vitis-ai-tutorial-part-1/?highlight=vitis%20ai

https://octavosystems.com/app_notes/vitis-ai-tutorial-part-2/?highlight=vitis%20ai

https://octavosystems.com/app_notes/vitis-ai-tutorial-part-3/?highlight=vitis%20ai

https://octavosystems.com/app_notes/vitis-ai-tutorial-part-4/?highlight=vitis%20ai

The tutorials from Octavo systems look promising as they use pre-built libraries downloaded from Xilinx repositories.

Another source of inspiration is the webinar from ADIUVO:

https://www.adiuvoengineering.com/vitis-training

Yet another tutorial comes from Beetlebox:

https://beetlebox.org/vitis-ai-using-tensorflow-and-keras-tutorial-part-1/

https://beetlebox.org/vitis-ai-using-tensorflow-and-keras-tutorial-part-2/

https://beetlebox.org/vitis-ai-using-tensorflow-and-keras-tutorial-part-3/

https://beetlebox.org/vitis-ai-using-tensorflow-and-keras-tutorial-part-4/

https://beetlebox.org/vitis-ai-using-tensorflow-and-keras-tutorial-part-5/

https://beetlebox.org/vitis-ai-using-tensorflow-and-keras-tutorial-part-6/

https://beetlebox.org/vitis-ai-using-tensorflow-and-keras-tutorial-part-7/

https://beetlebox.org/vitis-ai-using-tensorflow-and-keras-tutorial-part-8/

https://beetlebox.org/vitis-ai-using-tensorflow-and-keras-tutorial-part-9/

https://beetlebox.org/improving-convolutional-neural-networks-the-weaknesses-of-the-mnist-based-datasets-and-tips-for-improving-poor-datasets/

https://beetlebox.org/sign-language-recognition-hand-object-detection-using-r-cnn-and-yolo/

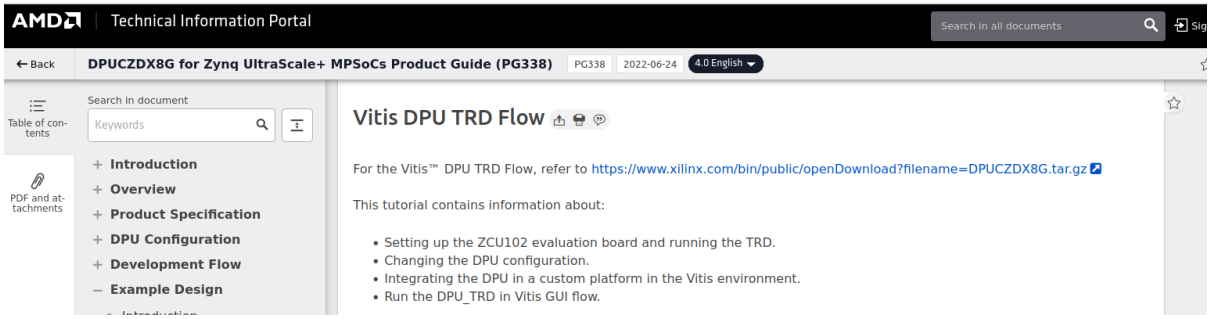
Another interesting tutorial is:

https://www.hackster.io/LogicTronix/zcu102-vitis-dpu-trd-vitis-ai-3-0-c51609

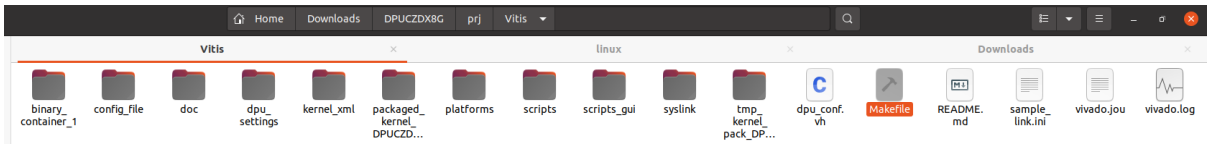
The product guide below has been used as a reference:

https://docs.amd.com/r/4.0-English/pg338-dpu/Vitis-DPU-TRD-Flow

And it refers to the following DPU archive, for the Vitis DPU TRD flow:



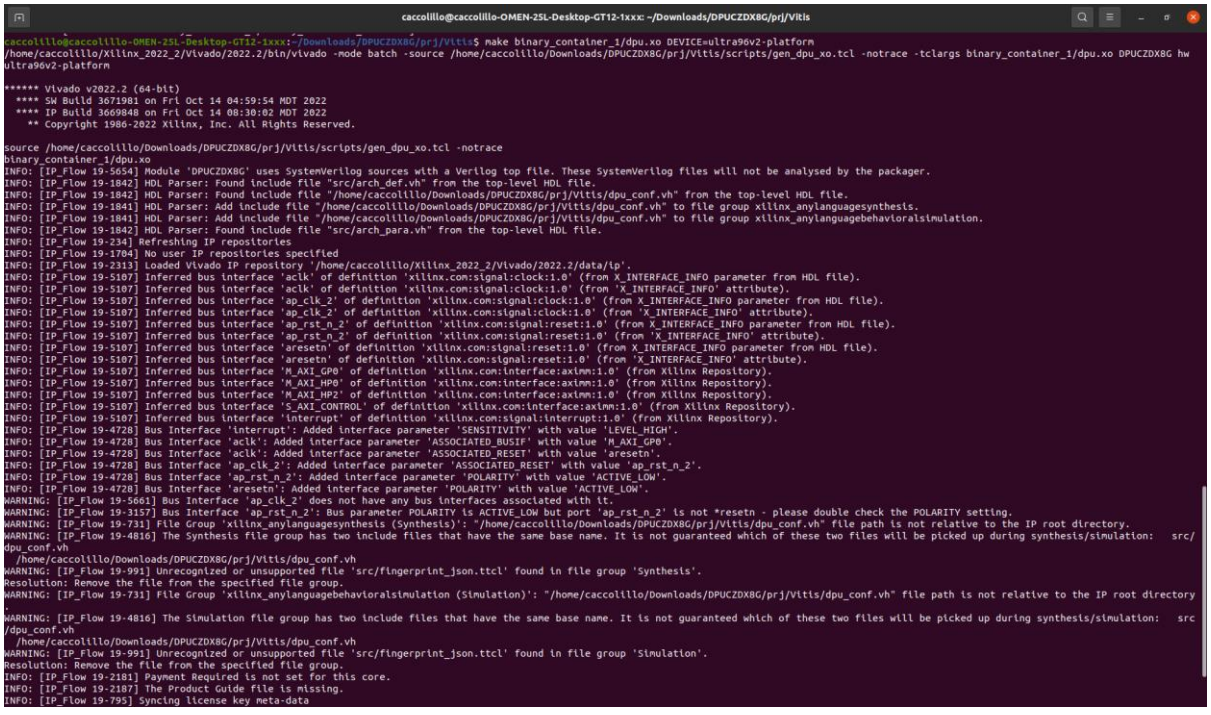
The archive gets downloaded and extracted, and the makefile gets modified:



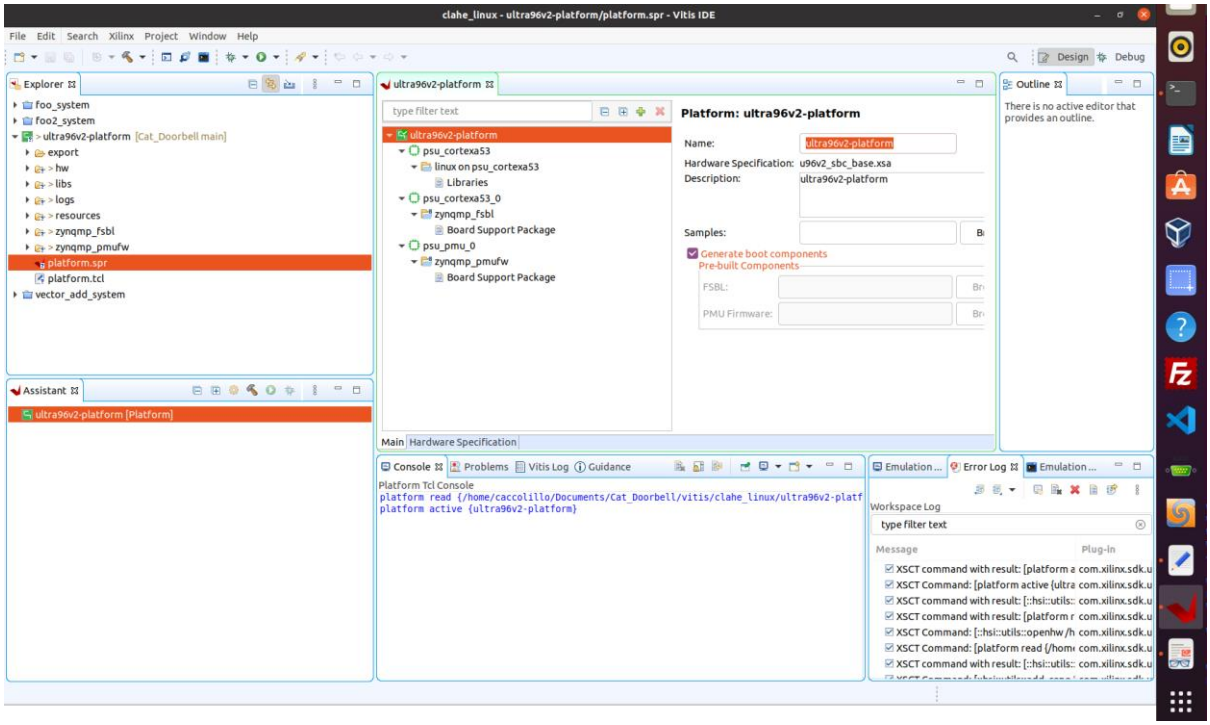
By adding the location of the linux image and of the xpfm file:

In the archive, there's also a readme file in MD. It can be opened as HTML by using "mdview README.md" on the command line.

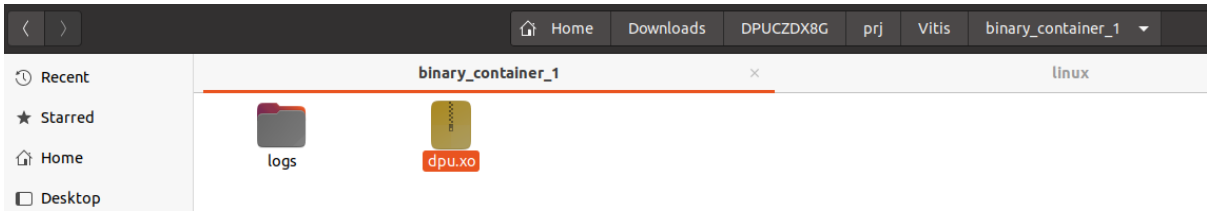
To generate the xo file, use "make binary_container_1/dpu.xo DEVICE=ultra96v2-platform":



Where "ultra96v2-platform" is the platform name in Vitis:



In the end, the xo file gets generated:



To install the DPU kernel template, follow:

https://adaptivesupport.amd.com/s/question/0D52E00007H6KQsSAN/why-application-template-of-dpu-kernel-rtl-kernel-is-not-exist-on-vitis-20221?language=en_US

It allows to create a project in Vitis easily.

A tutorial about how to use it in Vivado is at:

https://github.com/Xilinx/Vitis-AI-Tutorials/tree/2.0/Tutorials/Vitis-AI-Vivado-TRD

https://xilinx.github.io/Vitis-AI/3.5/html/docs/workflow-system-integration.html

https://github.com/Xilinx/Vitis-AI-Tutorials/blob/1.4/Introduction/README.md

https://github.com/Xilinx/Vitis-AI/blob/v3.0/dpu/ref_design_docs/README_DPUCZ_Vitis.md

A tutorial about how to use the DPU in Vitis, is available at:

https://octavosystems.com/app_notes/vitis-ai-tutorial-part-4/