

HLS4ML Workflow Summary

Oleh Kondratyuk

July 19th, 2022

1 Assumptions

- Installing on Ubuntu 18.04.5 (no reason other distros/versions shouldn't work, but the steps may differ slightly)
- If you're using a virtual machine (VM), you have dedicated enough disk space to hold Vivado and some additional programs. Not sure how much exactly is required but 100 GB should be safe, though significantly smaller should work.
- This is your own machine on which you have administrative privileges.


2 Installing Vivado

- Navigate to the [Vivado download archive](#) and download the v2020.1 Linux Self Extracting Web Installer
- You will need to enter your account details (make an account if you don't already have one) and some additional information (Name, address, function), though you can put random nonsense in those fields.
- Once the .bin file has been downloaded navigate to the folder where it is stored
- Make the file executable by running:

```
chmod u+x Xilinx_Unified_2020.1_0602_1208_Lin64.bin
```

- Run the bin file (./Xilinx_Unified_2020.1_0602_1208_Lin64.bin)
- In the event of the following error:

```
olehkond@ubuntu:~/Downloads$ ./Xilinx_Unified_2020.1_0602_1208_Lin64.bin
Verifying archive integrity... All good.
Uncompressing Xilinx Installer.....
.....
Exception in thread "SPLASH_LOAD_MESSAGE" java.lang.IllegalStateException: no splash screen available
    at java.desktop/java.awt.SplashScreen.checkVisible(Unknown Source)
    at java.desktop/java.awt.SplashScreen.getBounds(Unknown Source)
    at java.desktop/java.awt.SplashScreen.getSize(Unknown Source)
    at com.xilinx.installer.gui.H.run(Unknown Source)
```



- (a) Cntrl+C Out of the installer, and change directories to /etc
- (b) Using your preferred editor open the os-release file with a sudo prefix since this is a protected file. (e.g. sudo vim os-release)
- (c) Change the version ID to "18.04.4" as below:

```
NAME="Ubuntu"
VERSION="18.04 LTS (Bionic Beaver)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 18.04 LTS"
VERSION_ID="18.04.4"
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
VERSION_CODENAME=bionic
UBUNTU_CODENAME=bionic
```

- (d) Save your changes and relaunch the installer (error should be resolved)
- Installer will prompt you to enter your account username and password (same as credentials used to download installer)

Xilinx Unified 2020.1 Installer - Select Install Type

Select Install Type

Please select install type and provide your Xilinx.com user ID and password for authentication.

User Authentication

Please provide your Xilinx user account credentials to download the required files.
If you don't have an account, [please create one](#). If you forgot your password, you can [reset it here](#).

User ID:

Password:

☒ Download and Install Now

Select your desired device and tool installation options and the installer will download and install just what is required.

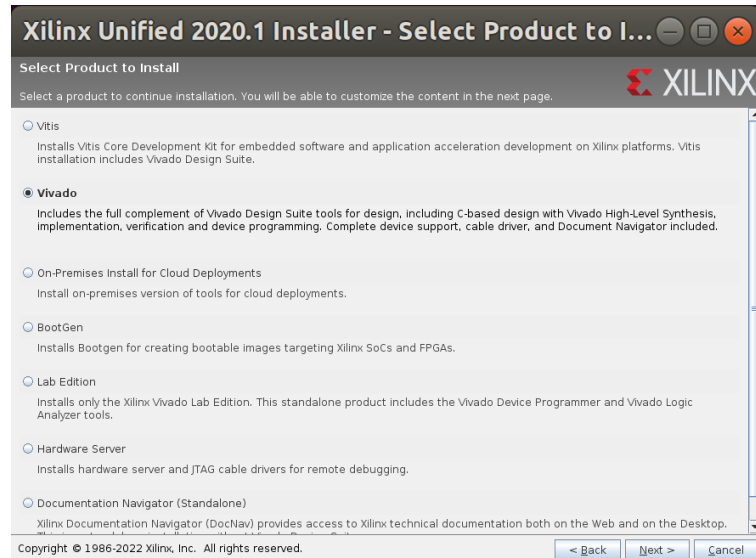
☐ Download Image (Install Separately)

The installer will download an image containing all devices and tool options for later installation. Use this option if you wish to install a full image on a network drive or allow different users maximum flexibility when installing.

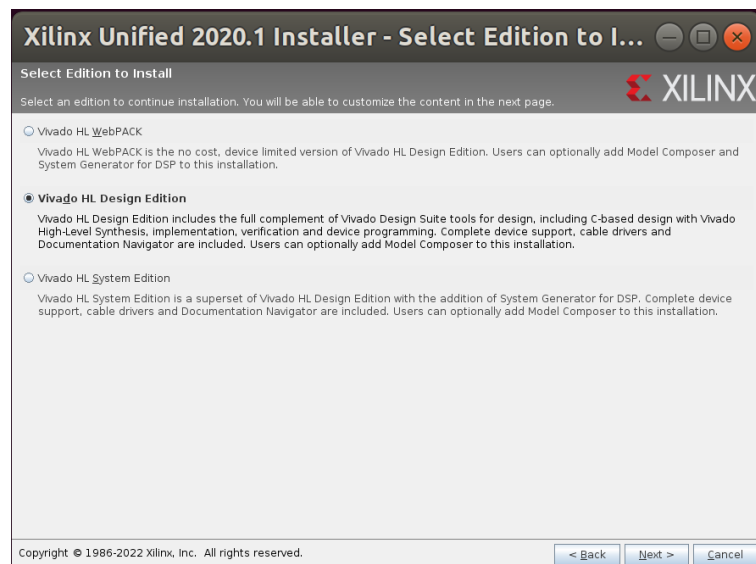
Copyright © 1986-2022 Xilinx, Inc. All rights reserved.

< Back Next > Cancel

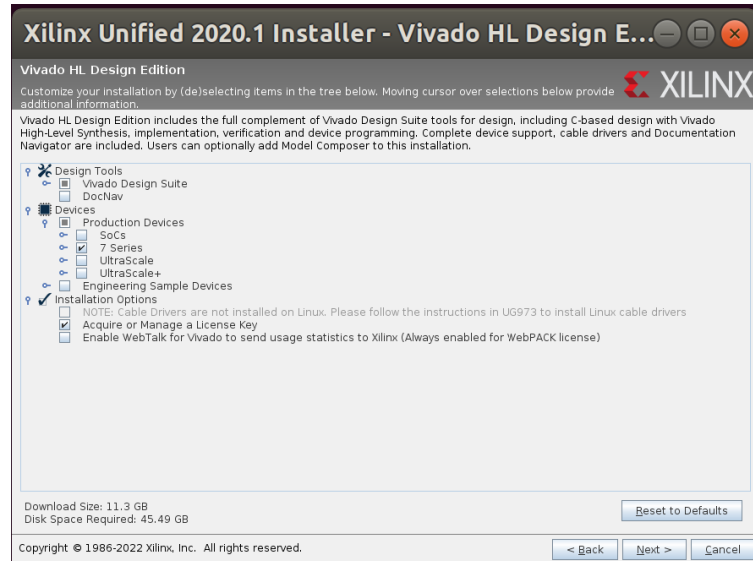
- Agree to all license agreements and select Vivado when prompted about which product to install:



- Since we need Vivado High Level Synthesis (HLS), select “Vivado HL Design Edition” when asked about the edition to install.



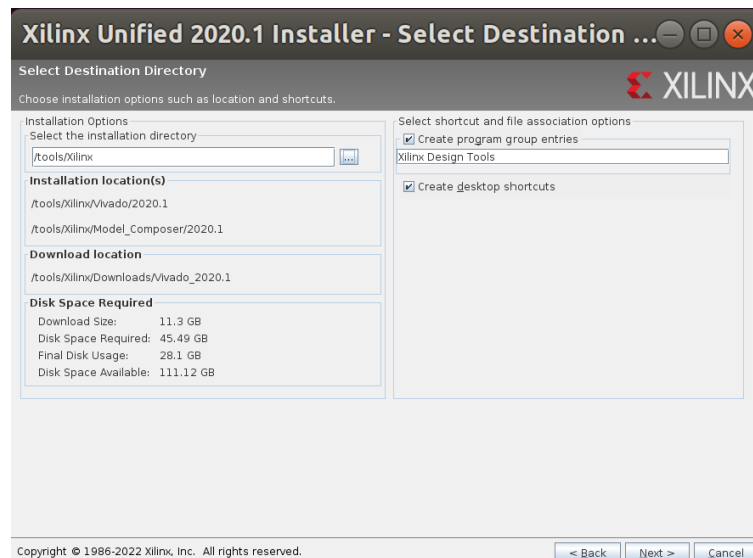
- To save disk space, dissect all the devices except one when prompted. However, if you plan on using this installation of Vivado to synthesize and place & route verilog designs (which is not what this document is about), you will want to leave the options relevant to you selected.



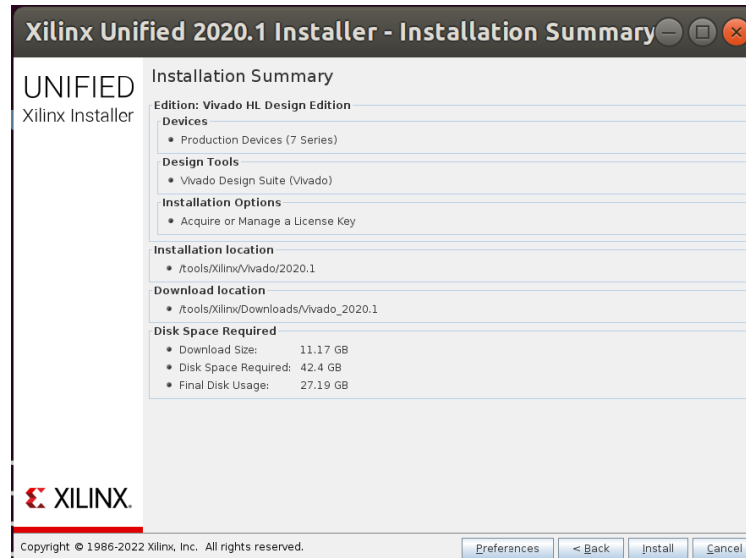
- Now choose an installation directory, you may either choose a custom location or create the default location (/tools/Xilinx) via:

```
cd /
sudo mkdir tools
sudo mkdir tools/Xilinx
sudo chmod -R a+rwX tools
```

After this, press the three dots next to the installation directory, then “Open” and the installer should stop complaining:



- The next page should give you an installation summary and you should be good to go on the installation.



Note: Installation times can easily be several hours depending on your internet speeds.

- Once installation finishes, you should be good to go and close the installer
- Once last step is to add the Vivado binaries to your PATH environment variable. To do this add the following line to the end of your .bashrc file:

```
export PATH="/tools/Xilinx/Vivado/2020.1/bin:$PATH"
```

3 Installing HLS4ML

It's a lot easier to use the anaconda virtual environment than to set up all the dependencies yourself, so before installing hls4ml, we need to install [anaconda3](#).

- After downloading the anaconda bash script, run it via:
"bash Anaconda3-2022.05-Linux-x86_64.sh"
- Go through the install script (it's fairly straightforward) and restart your terminal once the script finishes.
- Install hls4ml by simply running "pip install hls4ml", which will install all the necessary dependencies and hls4ml itself.
- A dependency not covered by anaconda is "make" and can be installed via:

```
sudo apt install make
```

- Done.

4 Work Flow for Lab

Now to actually generate Verilog models from hls4ml, we will need a couple files from the [hls4ml github](#). I will show the process for KERAS_1Layer, but the same idea applies to KERAS_conv2d.

- Create a directory to house your configuration and weight files
- The relevant example model files are found in [this folder](#), and for one layer we want to grab the 1 layer .json (config) file and .h5 (weights) file:

```
wget https://raw.githubusercontent.com/fastmachinelearning/example-models/ff74f73dbc253d1aa7de1603ee10ede551919548/keras/KERAS\_1layer.json
```

```
wget https://github.com/fastmachinelearning/example-models/raw/ff74f73dbc253d1aa7de1603ee10ede551919548/keras/KERAS_1layer_weights.h5
```

- Next grab, the keras config yaml file:

```
wget https://raw.githubusercontent.com/fastmachinelearning/example-models/ff74f73dbc253d1aa7de1603ee10ede551919548/keras-config.yaml
```

- At this point your directory should have the following files:

```
(base) oলেখকond@ubuntu:~/hls4ml$ ls
KERAS_1layer.json  KERAS_1layer_weights.h5  keras-config.yaml
```

This is all you need for the hls4ml tools, but before that we want to adjust it from it's default settings.

- Open the .yaml configuration file and make the following adjustments:
 - 1) Adjust the JSON and H5 paths (see example below)
 - 2) Change the Output directory and Project name to something more descriptive (optional)
 - 3) Change the Xilinx Part to xc7vx690tffg1761-2 (Virtex-7 VC709 Evaluation Board)
 - 4) Adjust the precision to whatever you need
 - a) Note that precision is specified as <Total Bits, Integer Bits>
 - 5) The default is fine for 1Layer, but for other models you may want to specify the IOType as io_parallel, io_serial, io_stream, similarly for reuse factor, 1 is fine for one-lay but you may need to adjust it for other models.

```

KerasJson: KERAS_1layer.json
KerasH5:   KERAS_1layer_weights.h5
#InputData: keras/KERAS_3layer_input_features.dat
#OutputPredictions: keras/KERAS_3layer_predictions.dat
OutputDir: oneLayer20_10
ProjectName: oneLayer20_10
XilinxPart: xc7vx690tffg1761-2
ClockPeriod: 5
Backend: Vivado

IOType: io_parallel # options: io_serial/io_parallel
HLSConfig:
  Model:
    Precision: ap_fixed<20,10>
    ReuseFactor: 1
# LayerType:
#   Dense:
#     ReuseFactor: 2
#     Strategy: Resource
#     Compression: True

```

4.1 hls4ml Commands

Now all that is left is to actually generate the verilog:

- Convert the config file into C++ code.

```
hls4ml convert -c keras-config.yml
```

You will find a new folder with C++ files detailing the function of the NN model.

- Now we want to build that C++ code into verilog using Vivado HLS.

```
hls4ml build -p oneLayer20_10 -a
```

- The resulting generated Verilog files would be found in:

```
oneLayer20_10/oneLayer20_10_prj/solution1/impl/verilog/
<Output Dir>/<Project Name>_prg/solution1/impl/verilog/ (general)
```

Now you can take the Verilog and create a Vivado project to synthesize and place and route the verilog to an actual FPGA. Email me at kondroleh@gmail.com if you have any questions or run into other issues.