Reservoir Computing in FPGA Setup Guide

By Nima Ghaffarzadeh
09/02/2026

In this guide you will find tutorials on the following topics:

- How to setup the PYNQ-Z1 board and access it
- How to install Vivado
- How to use hls4ml

System Information

I have tried the guides in this document on my laptop, and they have worked. Most importantly, you must have a **Linux** system with **x86-64** architecture. If you have another Operating System, you should look for other guides online. If it helps, here you have my PC's information:

```
Shell
Static hostname: debian
Icon name: computer-laptop
Chassis: laptop 💻
Operating System: Debian GNU/Linux forky/sid
Kernel: Linux 6.17.8+deb14-amd64
Architecture: x86-64
Hardware Model: VivoBook_ASUSLaptop X515JP_X515JP
Total Memory: 7.5Gi
```

Moreover, this guide assumes that you have at your disposal a PYNQ-Z1 board and its accessories.

How to setup the PYNQ-Z1 board and access it

* This guide is written based on the official PYNQ-Z1 setup guide.

1 - Connect the laptop to the PYNQ board, with both the ethernet cable and the USB cable. Then turn on the PYNQ board using the power button. After a minute you should see two Blue LD4 & LD5 LEDs and four Yellow/Green LD0-LD3 LEDs flash

simultaneously. The Blue LD4-LD5 LEDs will then turn on and off while the Yellow/Green LD0-LD3 LEDs remain on. The system is now booted and ready for use.

2 - To use jupyter homepage, which is how you can access PYNQ from your PC:

1. When you connect the ethernet cable to the PC, upon typing “ip link”, you should see a new interface for the ethernet cable.

2. Assign a static IP to the ethernet interface(If you use an Ubuntu-based system, you probably use NetworkManager, which means you can do it running the commands below):

```shell
nmcli con show          # find the connection name for the ethernet
nmcli con mod "interface_name" ipv4.method manual ipv4.addresses 
192.168.2.1/24
nmcli con up "interface_name"
```

3. Restart the network interface of your PC in order to apply the changes and turn on the ethernet interface:

```shell
systemctl restart networking
```

4. Ping the board:

```shell
ping 192.168.2.99
```

5. Browse to http://192.168.2.99:9090 to find the Jupyter homepage.
    - username: xilinx
    - password: xilinx
6. If you want to access the command interface, click on “New” -> “Terminal”.

3 - Give internet connection to the PYNQ board:

1. Enable IP forwarding on your PC. Edit /etc/sysctl.conf and ensure:

```
Shell
net.ipv4.ip_forward=1
```

2. Setup NAT(masquerading) on your PC. If you use iptables as a firewall, and if your PC is online via wlan0, and uses eth0 as the ethernet interface(if not, replace them with the correct interfaces. Also, 192.168.2.0/24 is the network of the PYNQ board):

```
Shell
sudo iptables -t nat -A POSTROUTING -o wlo1 -s 192.168.2.0/24 -j
MASQUERADE
```

```
Shell
sudo iptables -I FORWARD 1 -i PYNQ_IF -o INET_IF -j ACCEPT
```

```
Shell
sudo iptables -I FORWARD 2 -i INET_IF -o PYNQ_IF -m state --state
RELATED,ESTABLISHED -j ACCEPT
```

To view NAT rules(you should check if the counter is increasing over time):

```
Shell
sudo iptables -t nat -L -n -v
```

And the same for the FORWARD rules:

```
Shell
      sudo iptables -L FORWARD -v -n
```

To save rules, and make it persistent:

```shell
sudo apt install iptables-persistent
```

```shell
sudo netfilter-persistent save
```

3. Configure the PYNQ board. On the board, open the terminal, and go to /etc/network/interfaces.d/eth0 file:

```shell
auto eth0
iface eth0 inet static
address 192.168.2.99
netmask 255.255.255.0
gateway 192.168.2.1
dns-nameservers 8.8.8.8
```

   Restart networking:

```shell
sudo systemctl restart networking
```

   If it doesn't work, you could check if the route and gateway are set via:

```shell
ifconfig
route -n
cat /etc/resolv.conf
```

   If gateway isn't set:

```shell
sudo route add default gw 192.168.2.1
```

4. Test connectivity:

```Shell
ping 192.168.2.1
```

If it works, try:

```Shell
ping 8.8.8.8
```

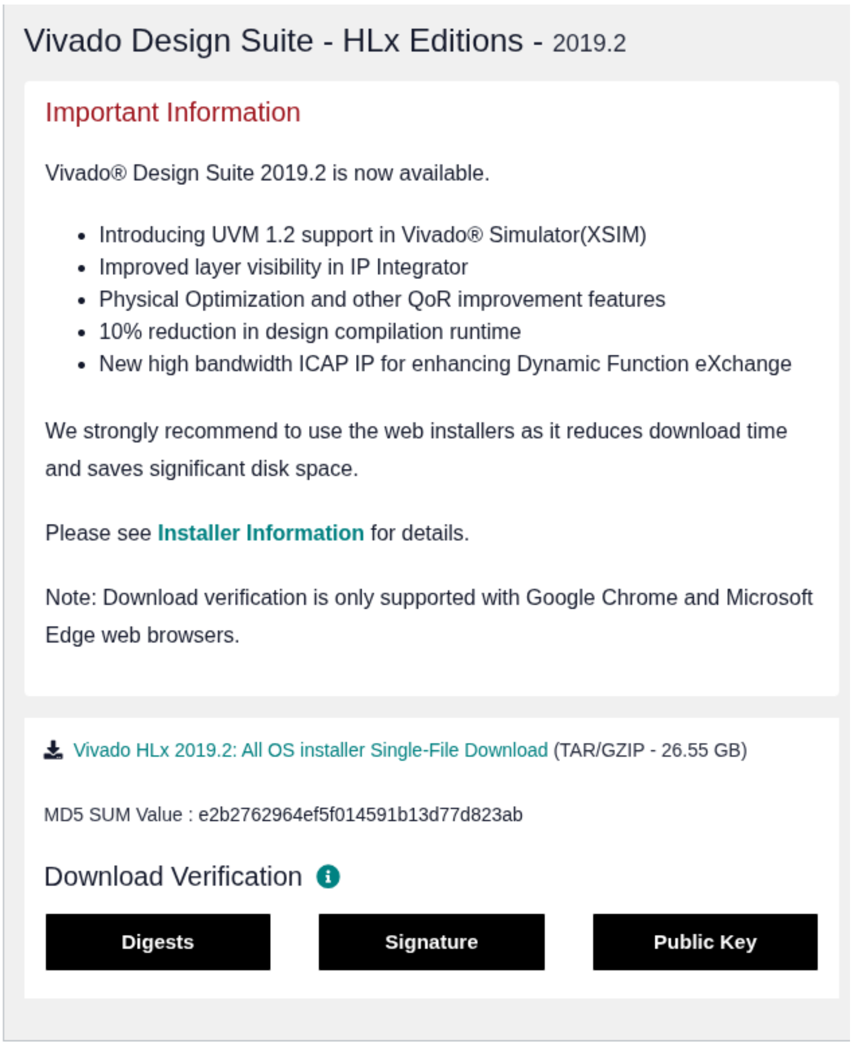
And to check DNS:

```Shell
ping google.com
```

How to install Vivado

1 - Go to the Vivado Archive, and if you can, download Vivado HLS 2020.1, because it is the Vivado version recommended by the hls4ml official website. I tried downloading this version but I faced installation errors. If you also face installation errors, then try version 2019.2, as I did. More information here.

2 - In the Vivado archive, download the "Vivado HLx 2019.2: All OS installer Single-File Download" (26.55 GB TAR/GZIP).


Vivado Design Suite - HLx Editions - 2019.2

Important Information

Vivado® Design Suite 2019.2 is now available.

- Introducing UVM 1.2 support in Vivado® Simulator(XSIM)
- Improved layer visibility in IP Integrator
- Physical Optimization and other QoR improvement features
- 10% reduction in design compilation runtime
- New high bandwidth ICAP IP for enhancing Dynamic Function eXchange

We strongly recommend to use the web installers as it reduces download time and saves significant disk space.

Please see Installer Information for details.

Note: Download verification is only supported with Google Chrome and Microsoft Edge web browsers.

Vivado HLx 2019.2: All OS installer Single-File Download (TAR/GZIP - 26.55 GB)

MD5 SUM Value : e2b2762964ef5f014591b13d77d823ab

Download Verification

3 - Once inside your Downloads directory, run these commands:

1. Extract the tar.gz file

```
Shell
tar -xzf Xilinx_Vivado_2019.2_1106_2127.tar.gz
```

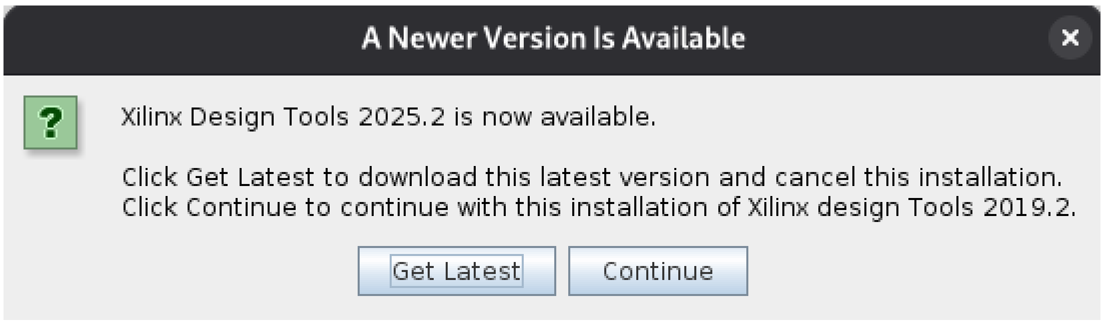
2. Navigate to the extracted directory

```
Shell
cd Xilinx_Vivado_2019.2_1106_2127
```

3. Run the installer

```
Shell
sudo ./xsetup
```

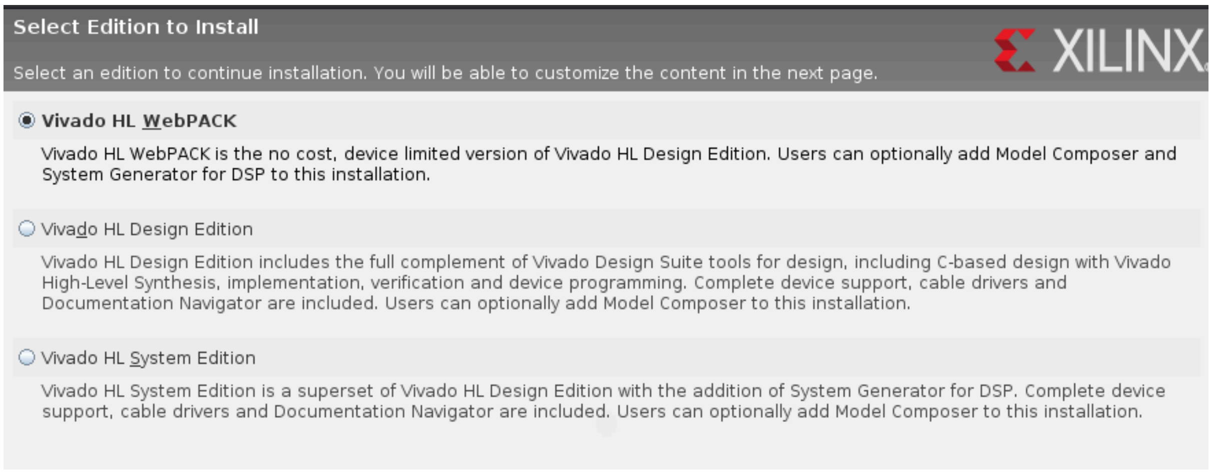
4 - When the wizard opens, it will suggest that you install a newer Vivado version. Make sure that you click on “Continue”.


A Newer Version Is Available

Xilinx Design Tools 2025.2 is now available.

Click Get Latest to download this latest version and cancel this installation.
Click Continue to continue with this installation of Xilinx design Tools 2019.2.

5 - In the Wizard, select the first option:


Select Edition to Install

Select an edition to continue installation. You will be able to customize the content in the next page.

[✓ Vivado HL WebPACK]

Vivado HL WebPACK is the no cost, device limited version of Vivado HL Design Edition. Users can optionally add Model Composer and System Generator for DSP to this installation.

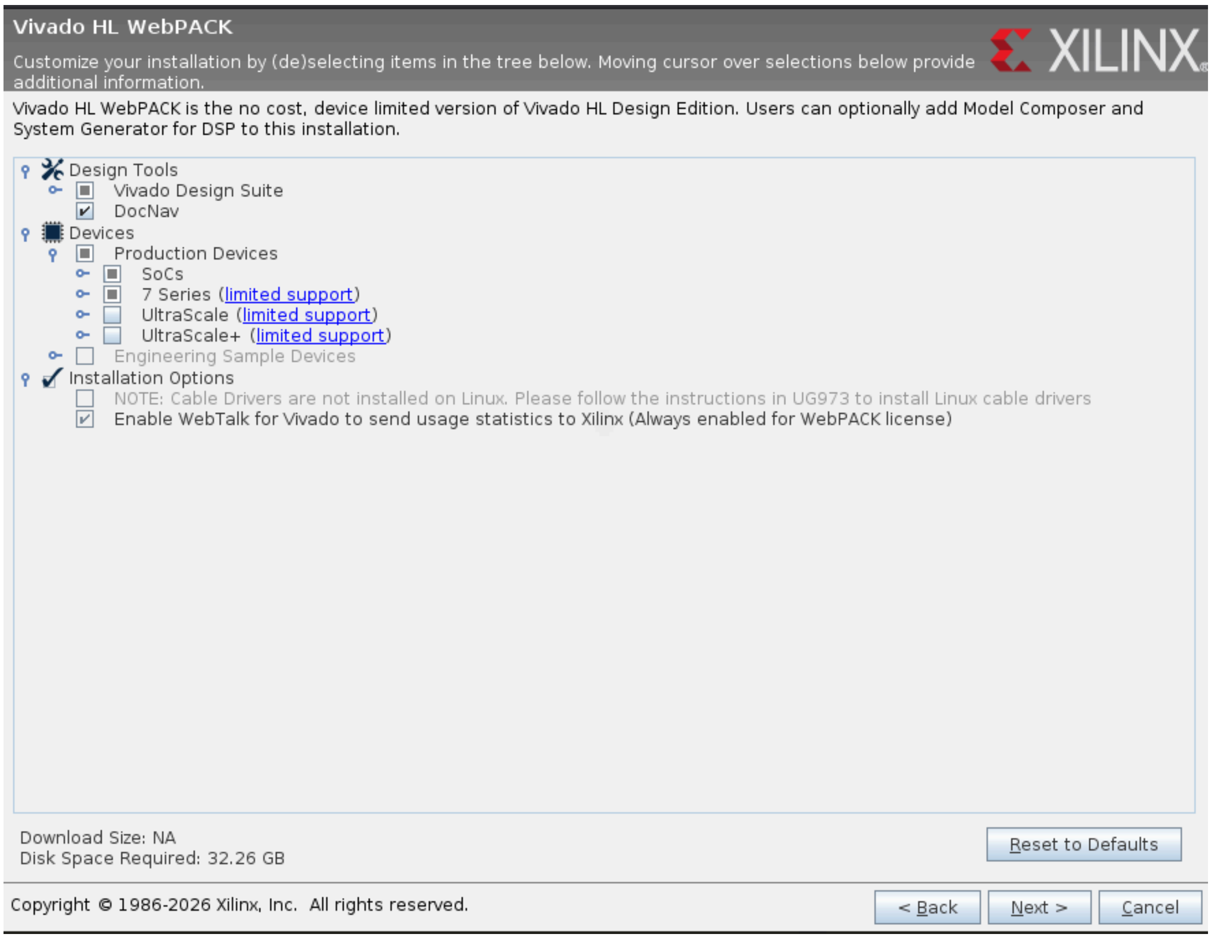
Vivado HL Design Edition

Vivado HL Design Edition includes the full complement of Vivado Design Suite tools for design, including C-based design with Vivado High-Level Synthesis, implementation, verification and device programming. Complete device support, cable drivers and Documentation Navigator are included. Users can optionally add Model Composer to this installation.

Vivado HL System Edition

Vivado HL System Edition is a superset of Vivado HL Design Edition with the addition of System Generator for DSP. Complete device support, cable drivers and Documentation Navigator are included. Users can optionally add Model Composer to this installation.

6 - Select the options below for a minimal installation:



Vivado HL WebPACK

Customize your installation by (de)selecting items in the tree below. Moving cursor over selections below provide additional information.

Vivado HL WebPACK is the no cost, device limited version of Vivado HL Design Edition. Users can optionally add Model Composer and System Generator for DSP to this installation.

- Design Tools
  - [■] Vivado Design Suite
  - [✓] DocNav
- Devices
  - [■] Production Devices
    - [■] SoCs
    - [■] 7 Series (limited support)
    - [ ] UltraScale (limited support)
    - [ ] UltraScale+ (limited support)
  - [ ] Engineering Sample Devices
- Installation Options
  - [ ] NOTE: Cable Drivers are not installed on Linux. Please follow the instructions in UG973 to install Linux cable drivers
  - [✓] Enable WebTalk for Vivado to send usage statistics to Xilinx (Always enabled for WebPACK license)

Download Size: NA
Disk Space Required: 32.26 GB

Copyright © 1986-2026 Xilinx, Inc. All rights reserved.

7 - After installation, you should have the file in /tools/Xilinx/Vivado/2019.2/, or in ~/Xilinx/Vivado/2019.2/. Navigate to the installation file, and:

1 . Source the settings file first. Open:

```Shell
nano ~/.bashrc
```

2. Add this line at the end (adjust path based on where you found it):

```Shell
source /tools/Xilinx/Vivado/2019.2/settings64.sh
```

3. Save and reload:

```Shell
source ~/.bashrc
```

4. Then launch Vivado GUI

```Shell
vivado
```

* If you get an error similar to this one:
application-specific initialization failed: couldn't load file "librdi_commontasks.so": libtinfo.so.5: cannot open shared object file: No such file or directory
%

You might need to install **libtinfo5** and **libncurses5** libraries:

```Shell
sudo apt-get install libtinfo5 libncurses5
```

How to use hls4ml

1 - On the PC side

First, let's install hls4ml. Hls4ml should be installed **on your PC** and not on the PYNQ board. In your project, using pip:

```Shell
pip install hls4ml
```

Then, you could write the training and inference code on your PC, and move the created model(bitstream) to the PYNQ board. You can follow the tutorials in this github repository:
https://github.com/fastmachinelearning/hls4ml-tutorial

You could get a minimal neural network to work on the FPGA by just following these tutorials:

- part1_getting_started.ipynb

- part7a_bitstream.ipynb
- part7b_deployment.ipynb

The rest of the tutorials are mostly for optimization. I am not sure how useful they are for Reservoir Computing.

I attach for you the modified versions of the first tutorial(part1_getting_started.ipynb), along with tutorials 7a and 7b(part7a_bitstream.ipynb and part7b_deployment.ipynb). The tutorials of the repository are written in **Keras**, so I have converted them to **Pytorch**.

The other change that I made in tutorial 7a, is that I deleted the **softmax function**, because it was very heavy for FPGA build, and took a ridiculous amount of time. I made some other small changes to make the build faster in tutorial 7. These changes are marked in the code with the comment:

```Python
# Modified to optimize
```

These are the names of the modified files that I attach for you:

- part1_getting_started.py
- part7_bitstreamMatteo.py (Includes both 7a and 7b)

Pay attention that the key line, which creates the bitstream for the FPGA, and is also the line that takes some time to run, is this one:

```Python
hls_model.build(csim=False, export=True, bitfile=True)
```

2 - On the PYNQ-Z1 board

After deployment to the board following tutorial part7b_deployment.ipynb, you will probably face some errors. These errors are probably due to the file axi_stream_driver.py. Change the “allocate” module with the “Xlnk” module in this file. I also attach the modified axi_stream_driver.py. This will probably work fine.

The file I mentioned(axi_stream_driver.py) is the driver that inputs and outputs data and results to/from the bitstream file. This driver file is a key file, and if you face other errors, it is very probable that the error is from this file.