Vitis Installation on Windows 10 and Windows 11 Systems for Red Pitaya

Samuel LeRose

Department of Physics and Astronomy

University of Kentucky

Lexington, Kentucky 40508, USA

Contents

1 Introduction

The purpose of this manual is to establish a Windows 10 (Win10) or 11 (Win11) based development machine for working with my GitHub repository. Red Pitaya was developed on the Linux Ubuntu distribution, necessitating the use of a computer running a Linux OS for development and communication. Typically, this would require a virtual machine or creating a dual boot system. However, on Win10 and Win11, we can use the Windows Subsystem for Linux GUI (WSLg) to run our Linux applications. If you are using Win10, then you must have Build 19044 or later as previous builds do not include the GUI portion of WSLg.

What is WSLg? WSLg is an integrated Linux distribution that can be run within Windows. You can think of it as a very barebones virtual machine, which will make more sense once you begin interacting with WSLg. The latest version of Ubuntu should work with this project, and at the time of writing, I am using 22.04.1 LTS (Jammy Jellyfish).

How do you program an FPGA? Programming an FPGA is done using Vivado, an application used to synthesize, simulate, and analyze digital circuits using hardware design languages (HDLs) like Verilog, the standard “coding” language for FPGAs. The term “coding” is used loosely when working with FPGAs, as it is slightly different from working with software (Python, C++, etc.). Software code utilizes the pre-defined circuits and functionality of a CPU for execution while FPGA code is designed to restructure the hardware itself to allow the FPGA to perform specific functions. Throughout this manual, programming and coding will be used interchangeably when discussing FPGAs, but it is important to know the difference between the purpose of software languages and HDLs.

Are there system requirements? The short answer is no, however, there are some general recommendations. Firstly, use a desktop computer as they are generally more dependable, powerful, and expandable than laptops. Secondly, prepare your computer for heavy workloads. Vivado is memory and CPU intensive, so 8-16GB of RAM and an Intel CORE i5 or AMD Ryzen 5 no older than 5 years is preferable. Vivado will work on older systems, but it could lead to instability if your hardware is pushed to its limits. Lastly, have a dedicated storage option. Installing WSLg is like a virtual machine, so it will take up a fair bit of storage, as will your Vivado projects. As such, I recommend you have a separate storage disk (ideally an SSD for its speed and reliability) between 125 and 250GB for your installations. External storage options are NOT compatible.

Writing code is not easily done in MS Word, but I tried to make it as seamless as possible. For reference, any code written in gray highlight is performed in Windows PowerShell and any in orange highlight is performed in Ubuntu.

2 Prerequisites

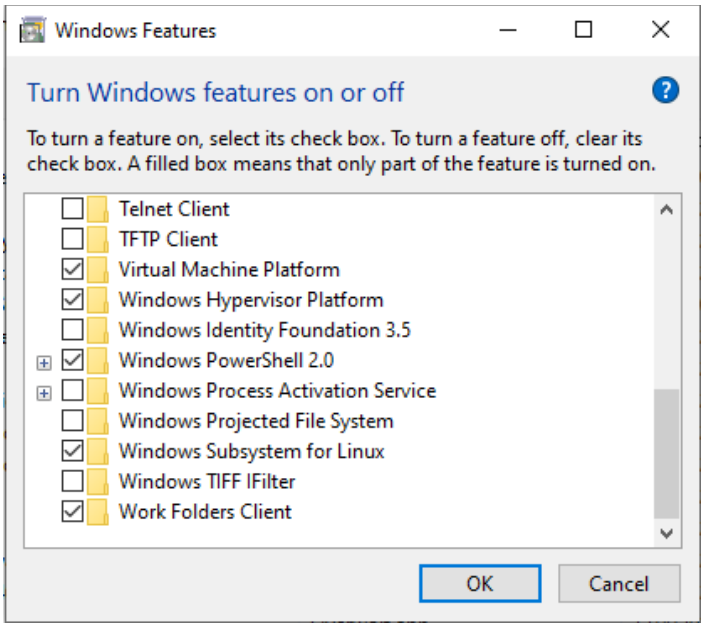
There are a few steps that must be performed on either Win10 or Win11 before beginning the installation.

2.1 Make sure your Windows graphics environment is up to date.

- WSLg runs alongside Windows using the same hardware, meaning your standard graphics environments must be up to date to allow for this integration.
  - Simply check your Windows Update to see if any updates need to be installed.
- If you are running a high-end system, you most likely have either an AMD or Nvidia graphics card, so make sure your drivers for those are up to date as well.

2.2 Build variables for running Virtual Machines

- Go to Settings → Apps → Apps & Features → Programs and Features (upper right) → Turn Windows Features on or off.
  - From here, enable "Virtual Machine Platform", "Windows Hypervisor Platform", and "Windows Subsystem for Linux".



  - Restart your computer when prompted.

3 Installing Ubuntu on Windows 10

3.1 Install WSL from the Microsoft Store.

- Search "WSL" and install the "Windows Subsystem for Linux" app with the penguin icon.
- This installs the latest release of WSLg.

3.2 Install Windows PowerShell 7 (WPS7).

3.3 Run WPS7 as administrator and perform the following:

- Make sure WSL is up to date (MS store should install the latest version so this step is performed for redundancy).
  - *wsl --update*
  - If WSL updates, then run *wsl --shutdown*, otherwise, no actions need to be taken.
- Install the Ubuntu distribution.
  - *wsl --install -d Ubuntu*
  - Ubuntu should launch once installed, and it will ask for a username and password for the distribution (make sure it is memorable and simple) which will be used in the future to run administrator commands.
- Check that the default WSL version is 2.
  - *wsl -l -v*
  - Make sure "2" is beside the previously installed Ubuntu distro.
- If the default WSL is NOT 2 (say you already have WSL and a distro installed), change the version.
  - *wsl --set-default-version 2*
  - Check the version to make sure the change is successful.

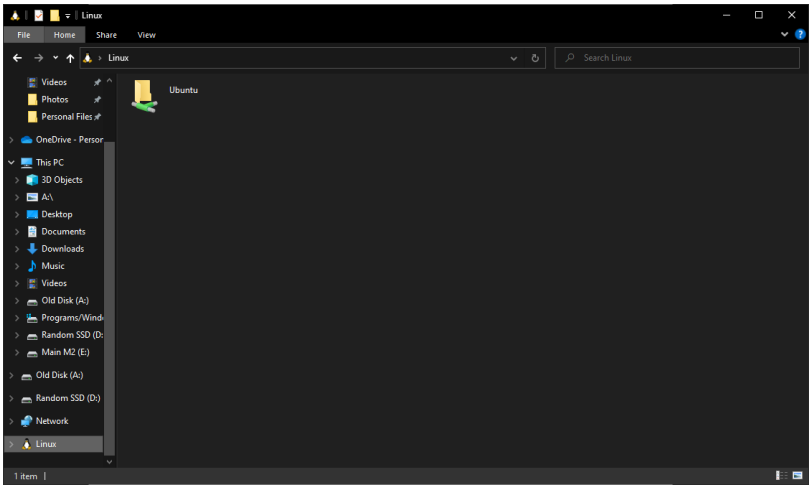
3.4 Once Ubuntu is installed, run it as admin and perform the following:

- *sudo apt-get update* AND *sudo apt-get upgrade*
- *sudo apt update && sudo apt upgrade -y*
- The above commands install all relevant updates and upgrades to the Ubuntu distribution. It is recommended you perform these commands when new updates are available (or roughly every other week).

3.5 Download X11 apps.

- *sudo apt install x11-apps -y*

3.6 You should now see a Linux folder at the bottom of your File Explorer directory, and within it, the folder holding the Ubuntu distribution.



3.7 (Optional) Download file viewer for WSLg.

- Allows you to view files accessible to WSL from a dedicated explorer.
- *sudo apt install nautilus -y*

4 Installing Ubuntu on Windows 11

4.1 Within Windows 11, install "Ubuntu" from the Microsoft Store (a version number is not associated with the installation).

- Win11 already has WSLg built into the build variables installed in the Prerequisite stage, so installing a distribution is all that is necessary.

4.2 Once Ubuntu is installed, run as admin, and perform the following:

- *sudo apt-get update* AND *sudo apt-get upgrade*
- *sudo apt update && sudo apt upgrade -y*

5 (Optional) Moving Ubuntu to Another Drive

This step is only necessary if you need to save space on your C drive (Note: I have only done this on Win10, but I assume it should work the same on Win11).

I followed the steps from DEV (link) and I am using D: as my new drive.

5.1 Within Ubuntu, double-check your username.

- Run *whoami* and note the username.

5.2 Within WPS7, shutdown all Linux consoles

- *wsl --shutdown*

5.3 Create a backup of Linux distribution (not very important unless you have already used Linux in the past, but still a good practice).

- *mkdir D:\backup*
- *wsl --export Ubuntu D:\backup\ubuntu.tar*
- This may take a few minutes, and there is no progress bar, so just wait until the command line refreshes.

5.4 Unregister Ubuntu on C: and import it to your new disk.

- *wsl --unregister Ubuntu*
- *mkdir D:\WSL*
- *wsl --import Ubuntu D:\WSL\ D:\backup\ubuntu.tar*

5.5 Switch to personal Linux account

- *cd $env:USERPROFILE\AppData\Local\Microsoft\WindowsApps*
- *ubuntu config --default-user <USER_NAME>*
- Replace <USER_NAME> with your Ubuntu username recorded earlier.
  - USERPROFILE is not a variable like <USER_NAME>, so leave it as is.

5.6 Run *wsl* to restart WSL and apply the changes.

5.7 The new file path will be in D:\WSL as an ext4 "external drive".

- The Ubuntu folder can be accessed just as before, all the data will just be stored on your new drive instead.

6 Installing Xilinx Vitis 2023.1

6.1 Install “AMD Unified Installer for FPGAs & Adaptive SoCs 2023.1: Linux Self Extracting Web Installer” from this link.

- Click 2023.l, scroll to “Full Product Installations”, Linux installer is the second option.

- You will be asked to create an AMD account and include what your intended use for Vivado is, just record it as “Personal Use” and your job description as “Student” or something similar.

6.2 To access the Vivado web installer you must access the folder it is downloaded in, likely the “Downloads” folder in your C: drive.

- *cd /mnt/c/Users/<USER_NAME>/Downloads* to access the file location in Ubuntu.
  - Your <USER_NAME> can be found by using File Explorer, entering C: drive, and opening the “Users” folder.

6.3 Install Vivado by executing the bin file.

- Change access permissions of files and directories.
  - *sudo chmod +x ./Xilinx_Unified_2023.1_0507_1903_Lin64.bin*
- Run the bin file.
  - *sudo ./Xilinx_Unified_2023.1_0507_1903_Lin64.bin*
- Once the install wizard is open, click next, insert ID and password, click “Download and Install Now”, accept all license agreements, click “Vivado”, check “Vivado HL WebPACK”, and select only Zynq-7000 under SoCs.
  - Change install path to /opt/Xilinx.
- If, after attempting the *sudo ./Xilinx_Unified_2023.1_0507_1903_Lin64.bin* command, you are thrown a Java Headless Exception error due to no X11 DISPLAY variable set, try this solution (only occurred on Win11).
  - Install Java Development Kit via *sudo apt-get install openjdk-default-jdk*, and an installation of Java Runtime Environment via *sudo apt-get install openjdk-default-jre*
  - Solutions are from Stack Overflow at Links 1 and 2
- Once finished, exit the installer and install additional libraries.
  - *sudo apt-get install libxft2 -y*

6.4 Fix LC_ALL locale error.

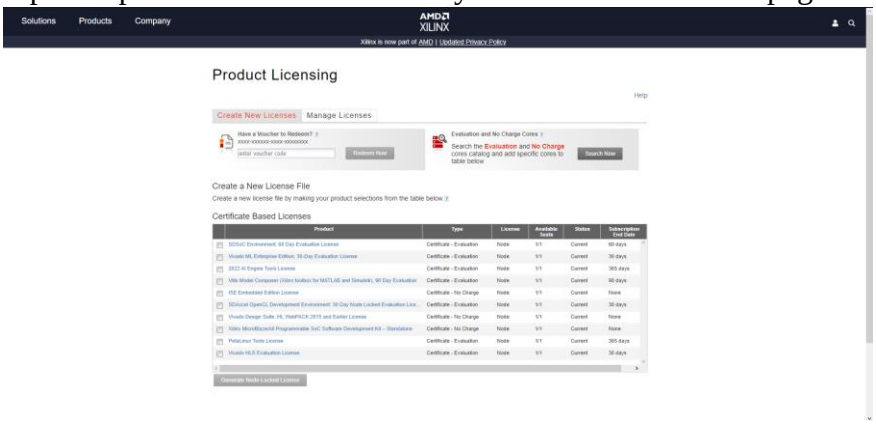
- *sudo locale-gen en_US.UTF-8*

6.5 Set source for Vivado to run when called (we will permanently implement this later, this step is for licensing the installer).

- *source /opt/Xilinx/Vivado/2023.1/settings64.sh*

6.6 Vivado installer did not launch the license manager, so licensing will be done manually.

- Navigate to the Xilinx licensing form in your browser (link).
- It should look familiar as it is the same form seen when installing the Unified web installer.
    - Input all pertinent information and you should then be on a page seen here.



- Select “ISE WebPACK License” and follow the prompts, no information needs to be entered in any boxes.
    - Should receive an email from Xilinx.notification with the license as a .lic file, save this file to the Downloads folder and do NOT delete the email.
- Return to Ubuntu in the ~ directory (*cd*), and run *vlm* for Vivado License Manager
    - Click “Load License” on the left nav bar and click “Copy License”.
    - In the navigation drop-down, return to / directory, navigate to the Downloads folder (mnt → C → Users → <USER_NAME> → Downloads), then select the “Xilinx.lic” file.
    - Exit VLM from the Files dropdown.

6.7 Set *sources* in *bashrc* for Ubuntu to load at the start (prevents you from having to run the source script on each startup).

- *sudo nano ~/.bashrc*
- Once in the console, navigate to the bottom (using arrow keys) and add the following for the Vivado sources:
    - *# set Vivado source for config of system variables*
    - *source /opt/Xilinx/Vivado/2023.1/settings64.sh*
- Save and exit.

7 Library Installations and Executable Fixes

7.1 (Nvidia users) Preemptively fix symbolic link errors with the CUDA environment.

- Source of fix: GitHub WSL Issue #5548.
- Add *ldconfig* variable to *wsl.conf*:
    - *echo -e "[automount]\nldconfig = false" | sudo tee -a /etc/wsl.conf*

- Make and use the symbolic link:
  - *sudo mkdir /usr/lib/wsl/lib2*
  - *sudo ln -s /usr/lib/wsl/lib/* /usr/lib/wsl/lib2*
  - *echo /usr/lib/wsl/lib2 | sudo tee /etc/ld.so.conf.d/ld.wsl.conf*

7.2 Install the various development packages and other dependencies.

- *sudo apt-get install unixodbc unixodbc-dev libncurses-dev libzmq3-dev libxext6 libasound2 libxml2 libx11-6 libxtst6 libedit-dev libxft-dev libxi6 -y*
- *sudo apt-get install make curl xz-utils libssl-dev device-tree-compiler u-boot-tools schroot qemu qemu-user qemu-user-static libtinfo-dev gcc-multilib g++-multilib libncurses5-dev lib32z1 bc zip -y*

7.3 Install the Meson Build system which is used for some new code.

- Run the following:
  - *sudo apt-get install python3 python3-pip*
  - *sudo pip3 install --upgrade pip*
  - *sudo pip3 install meson*
  - *sudo apt-get install ninja-build*
- You may receive some warnings about running pip as the root user, but I have not had any issues.

7.4 Fix the missing *gmake* and *libtinfo* executables.

- *sudo ln -s /usr/bin/make /usr/bin/gmake*
- *sudo ln -s /lib/x86_64-linux-gnu/libtinfo.so.6 /lib/x86_64-linux-gnu/libtinfo.so.5*

7.5 Install git.

- *sudo apt install git*

8 Unpack RedPitaya-DSP Repository

8.1 Close the RedPitaya-DSP repo from GitHub.

- From the ~ directory, run:
  - *git clone https://github.com/sslerose/RedPitaya-DSP.git*

8.2 Rename cloned repo.

- The repo will be cloned as a folder named “RedPitaya-DSP-master”, rename the folder to “RedPitaya-DSP”.
  - All the automated scripts rely on this folder name.