The Ultimate[1] Guide for Xilinx Vitis Installation

Introduction

This guide will assist through the setup of a functional synthesis and emulation framework for a Zynq UltraScale+ (ZCU104) platform. The following tools are required:

- **Xilinx Vitis 2020.2**
- **Embedded Base Platform:** platform files specific to the embedded platform we're using
- **ZynqMP Common Image Package:** a bootable Linux system for the ZCU104, along with cross-compilation tools

We will consider that installation will be performed on a system running Ubuntu 20.04 LTS. At least 100GB of free space is required for the whole installation process (a significant portion is occupied by download files, which are deleted after installation). Although this guide is specific to Vitis version 2020.2 and the ZCU104, instructions might be similar for other platforms. For more information about the installation process and some first steps, refer to our repository (https://github.com/comododragon/fccm2021-tutorial) or also the reference documentation from Xilinx, for example:

- **Vitis 2020.2 Getting Started Tutorials:** https://github.com/Xilinx/Vitis-Tutorials/tree/master/Getting_Started
- **Xilinx Downloads Section:** https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vitis/2020-2.html
- **Vitis Installation Guide:** https://www.xilinx.com/html_docs/xilinx2020_2/vitis_doc/acceleration_installation.html#vhc1571429852245

Requirements

We assume that the reader has the essential knowledge of Linux, including dealing with files through command line, the filesystem hierarchy, using a package manager (apt in this case), etc. This tutorial was performed on a clean install of Ubuntu 20.04 LTS. There are some libraries that should be installed on your system before proceeding to installation:

```
$ sudo apt install libtinfo5 libtinfo6 libtinfo-dev
```

Please do install these libraries, or your installation will hang at the very end after downloading tons of GBs (not nice!)!

Also download build-essential if you don't have it. Very important for your programming needs:

```
$ sudo apt install build-essential
```

1 Maybe not.

Then, download the required files[2]:

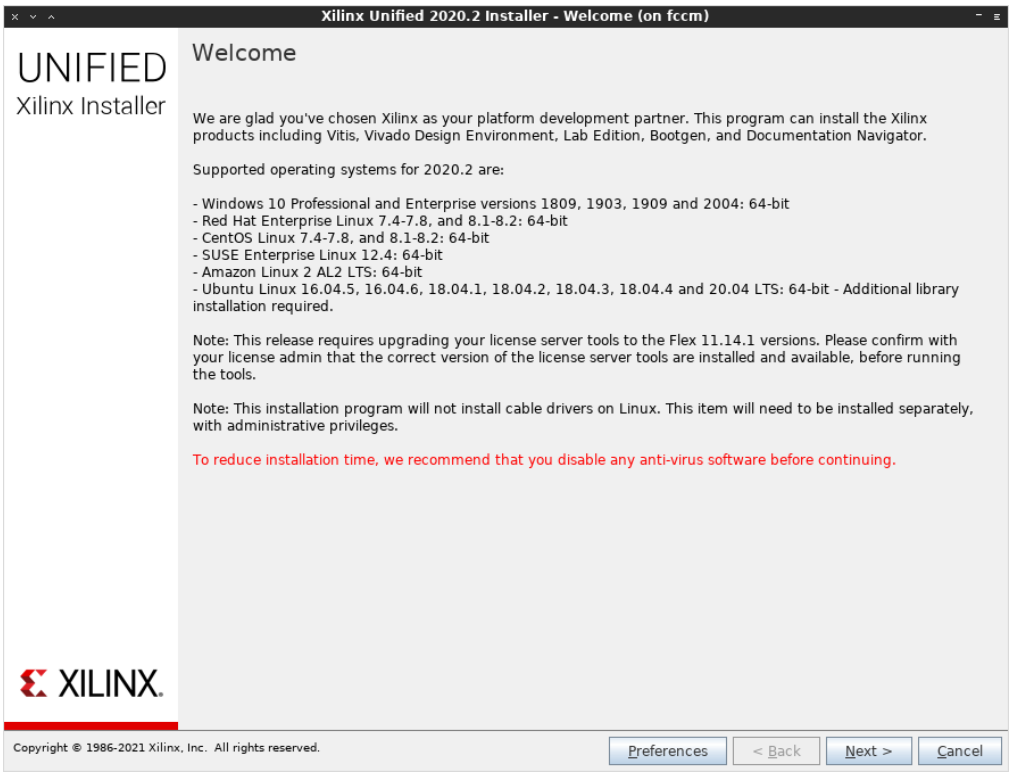
- **Xilinx Vitis 2020.2 Linux Web Installer (`Xilinx_Unified_2020.2_1118_1232_Lin64.bin`):** https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vitis/2020-2.html
- **ZCU104 Embedded Base Platform (`xilinx_zcu104_base_202020_1.zip`):** https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/embedded-platforms/2020-2.html
- **ZynqMP Common Image Package (`xilinx-zynqmp-common-v2020.2.tar.gz`):** Same URL as before, look for the "Common images for Embedded Vitis platforms" section.

Vitis Installation

Make the setup file (`Xilinx_Unified_2020.2_1118_1232_Lin64.bin`) executable and run it:

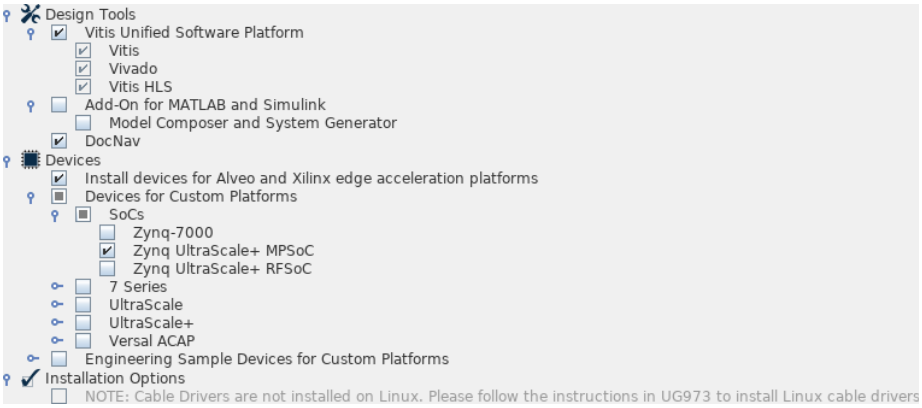
```
$ chmod +x Xilinx_Unified_2020.2_1118_1232_Lin64.bin
$ ./Xilinx_Unified_2020.2_1118_1232_Lin64.bin
```

A graphical setup will appear:



- Click **Next**
- Insert your Xilinx credentials, and click **Next**
- Leave Vitis option selected, click **Next**
- Select at least the following options to install and click **Next**:

2 You must be registered on the Xilinx website to download the files. The registration is free.



- Select “I agree” to all three options and click **Next**
- Select **`/opt/xilinx`** as the installation directory, click **Next**
- Press **Install** (go drink a coffee, a beer, play some games…)
- Press **Finish** to conclude Vitis installation
- It is recommended to run the installLibs.sh script to automatically install missing libraries if any:

```
$ sudo /opt/xilinx/Vitis/2020.2/scripts/installLibs.sh
```

- Installation of Vitis is finished! You can now proceed to installing the ZCU104 platform.

ZCU104 Embedded Base Platform Installation

Extract the downloaded platform to /opt/xilinx/platforms:

```
$ mkdir /opt/xilinx/platforms
$ unzip xilinx_zcu104_base_202020_1.zip -d /opt/xilinx/platforms/
```

ZynqMP Common Image Package Installation

Extract the downloaded common image package to /opt/xilinx/rootfs:

```
$ mkdir /opt/xilinx/rootfs
$ tar xvzf xilinx-zynqmp-common-v2020.2.tar.gz -C /opt/xilinx/rootfs/
```

Then run the **sdk.sh** script to generate the cross-compilation environment. Use **/opt/xilinx/petalinux/2020.2** as the destination path:

```
$ /opt/xilinx/rootfs/xilinx-zynqmp-common-v2020.2/sdk.sh
PetaLinux SDK installer version 2020.2
======================================
Enter target directory for SDK: /opt/xilinx/petalinux/2020.2
You are about to install the SDK. Proceed [Y/n]? y
Extracting
SDK.....................................................................................
.............................................done
Setting it up...done
/bin/bash: warning: setlocale: LC_ALL: cannot change locale (en_US.UTF-8)
SDK has been successfully set up and is ready to be used.
Each time you wish to use the SDK in a new shell session, you need to source
the environment setup script e.g.
 $ . /opt/xilinx/petalinux/2020.2/environment-setup-aarch64-xilinx-linux
```

Your system is now ready to build and run Vitis designs!

Testing your Setup

You can test your setup by using our hello world project:

1. GIT clone https://github.com/comododragon/fccm2021-tutorial
   ```
   $ git clone https://github.com/comododragon/fccm2021-tutorial.git
   ```
   - You can alternatively download the repo as a zip
2. cd to the repo:
   ```
   $ cd fccm2021-tutorial
   ```
3. Setup your environment by sourcing the setup.sh script:
   ```
   $ source ./setup.sh
   ```
   Note: this script sets your system up according to the installation paths as presented in this tutorial. If you installed in different paths from here, you should either modify the script or perform the following steps manually once per shell session:
   - `source <PATH_TO_VITIS_INSTALL>/Vitis/2020.2/settings64.sh`
     - **<PATH_TO_VITIS_INSTALL>** is the installation directory selected in the Vitis Installation (in this tutorial: **/opt/xilinx**)
   - `source <PATH_TO_PETALINUX>/environment-setup-aarch64-xilinx-linux`
     - **<PATH_TO_PETALINUX>** is the folder where the cross-compilation environment was set using **sdk.sh** (in this tutorial: **/opt/xilinx/petalinux/2020.2**)
   - `export PLATFORM_REPO_PATHS=<PLATFORMS_FOLDER>/xilinx_zcu104_base_202020_1`
     - **<PLATFORMS_FOLDER>** is the folder where the **ZCU104 Embedded Base Platform** was extracted to (in this tutorial: **/opt/xilinx/platforms**)
   - `export EDGE_COMMON_SW=<ROOTFS_FOLDER>/xilinx-zynqmp-common-v2020.2`
     - **<ROOTFS_FOLDER>** is the folder where the **ZynqMP Common Image Package** was extracted to (in this tutorial: **/opt/xilinx/rootfs**)
   - `export SYSROOT=<PATH_TO_PETALINUX>/sysroots/aarch64-xilinx-linux`
4. cd to the **01-hello-world** project:
   ```
   $ cd 01-hello-world
   ```
5. Run make to build the project and a bootable SD image:
   ```
   $ make build TARGET=<TARGET>
   ```
   where **<TARGET>** may be **sw_emu** (software emulation), **hw_emu** (hardware emulation) or **hw** (hardware generation). The two first may be executed on the QEMU emulator as shown in the next step. For **hw**, the generated **project/package.hw/sd_card.img** must be cloned to a microSD card and booted on a ZCU104 platform. In this case you can skip the next step.
6. Run make to open QEMU. This will start an emulated embedded Linux on your computer:
   ```
   $ make run TARGET=<TARGET>
   ```
7. Wait for the embedded Linux to boot. When the command prompt shows, run the following commands to access the SD card partition, set up the platform and run the hello world example:
   ```
   $ cd /mnt/sd-mmcblk0p1
   $ source ./init.sh
   $ ./run_app.sh
   ```
8. The example should run and show the message **TEST_PASSED** (output might slightly vary):
   ```
   Found Platform
   Platform Name: Xilinx
   INFO: Reading vadd.xclbin
   Loading: 'vadd.xclbin'
   [ 1066.514435] [drm] Pid 1363 opened device
   ```

```
Trying to program device[0]: xilinx_zcu104_base_202020_1
[ 1069.457044] [drm] get section DEBUG_IP_LAYOUT err: -22
[ 1069.468640] [drm] get section AIE_METADATA err: -22
[ 1069.489221] [drm] zocl_xclbin_read_axlf fe0e3f0b-deb4-4f35-8936-
10d0ed851c98 ret: 0
[ 1069.584959] [drm] bitstream fe0e3f0b-deb4-4f35-8936-10d0ed851c98 locked,
ref=1
[ 1069.591993] [drm] No ERT scheduler on MPSoC, using KDS
[ 1069.707133] [drm] scheduler config ert(0)
[ 1069.707177] [drm]   cus(1)
[ 1069.709142] [drm]   slots(16)
[ 1069.711052] [drm]   num_cu_masks(1)
[ 1069.726736] [drm]   cu_shift(16)
[ 1069.728954] [drm]   cu_base(0x80000000)
[ 1069.730214] [drm]   polling(0)
[ 1069.735498] [drm] bitstream fe0e3f0b-deb4-4f35-8936-10d0ed851c98
unlocked, ref=0
Device[0]: program successful!
[ 1069.882318] [drm] bitstream fe0e3f0b-deb4-4f35-8936-10d0ed851c98 locked,
ref=1
[ 1070.096975] [drm] User buffer is not physical contiguous
[ 1070.173217] [drm] User buffer is not physical contiguous
[ 1070.193661] [drm] User buffer is not physical contiguous
TEST PASSED
[ 1493.986319] [drm] bitstream fe0e3f0b-deb4-4f35-8936-10d0ed851c98
unlocked, ref=0
[ 1494.794881] [drm] Pid 1363 closed device
INFO: host run completed.
```

9. You can now power off the system:

```
$ poweroff
```

10. To end emulation, press **CTRL+A** and **X**