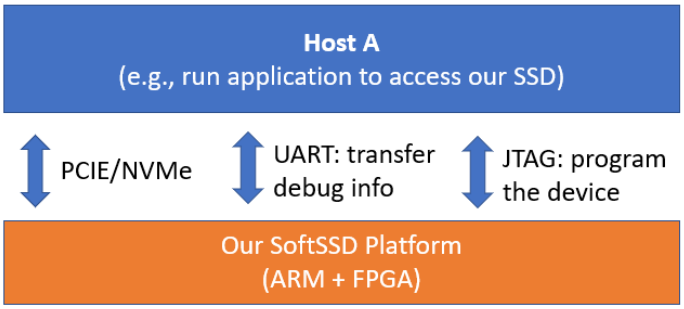
# SoftSSD Setup and Usage Guide

# CONTENTS

# 1. Overview

## 1.1 Hardware

1. Hardware platform: Use Zynq Ultrascale+ MPSoC device and implement it as an SSD with computation capacity (e.g., with ARM CPU or FPGA).




2. Function of ARM Cores (Software).
(1) 4x ARM Cortex A53. One for FTL, three for CSD tasks.
(2) 2x ARM Cortex R5. One for ECC, one for FIL.
3. Function of FPGA (Hardware offloading).
(1) NVMe protocol.
(2) PCIe protocol.

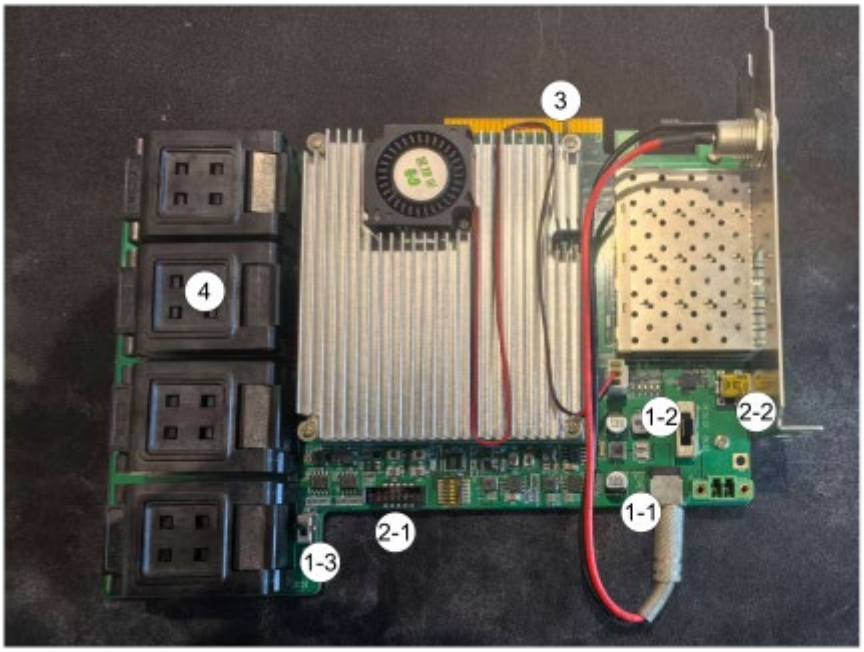
## 1.2 Software

1. Project github link (private): *https://github.com/duzelin/SoftSSD*.

## 1.3 Deployment

1. Prepare two host machines: one serves as the host PC for downloading firmware to and monitoring the output from the target board, and the other as the test machine.
2. Install flash chips.
3. (Optional) Configure the flash parameters.
4. Adjust the power switch.
5. Connect the board to the host PC
6. Insert the board into the test machine's PCIe slot.
7. Boot the board by downloading and launching the firmware.
8. Boot the **test machine's**
9. The storage device can be observed using *lspci* or *lsblk*.

### 1.4 CSD connection guide




1. Power delivery and regulation (1-1 12V DC power, 1-2 On/Off control, 1-3 Voltage control),
2. Debug interfaces (2-1 JTAG, 2-2 UART),
3. A PCIe interface for host connectivity (3), and
4. NAND flash interfaces to connect multiple flash packages (4).

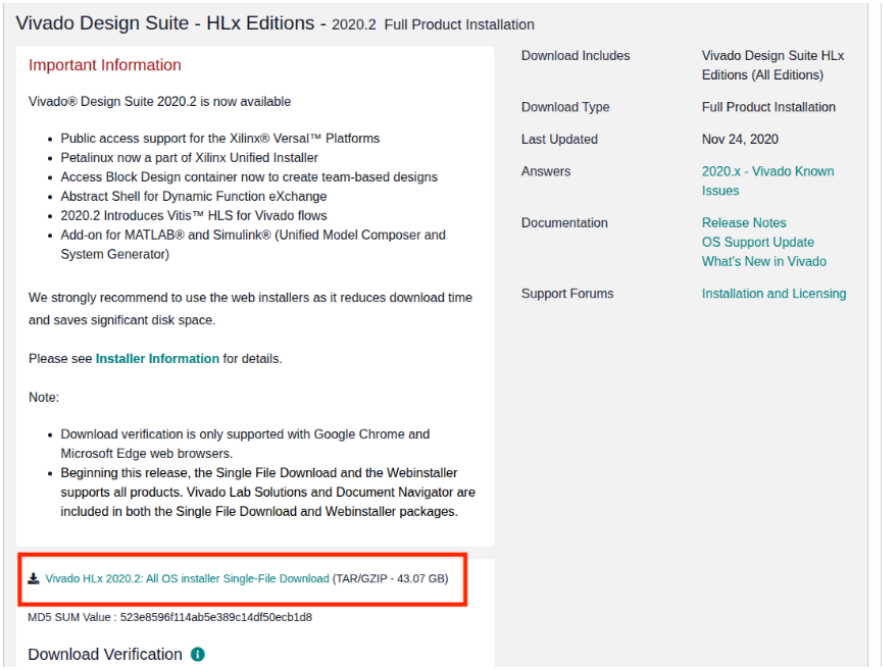
## 2. Software install

### 2.1 Prerequisites

1. OS: Ubuntu22.04
2. Vivado: Vivado 2020.2

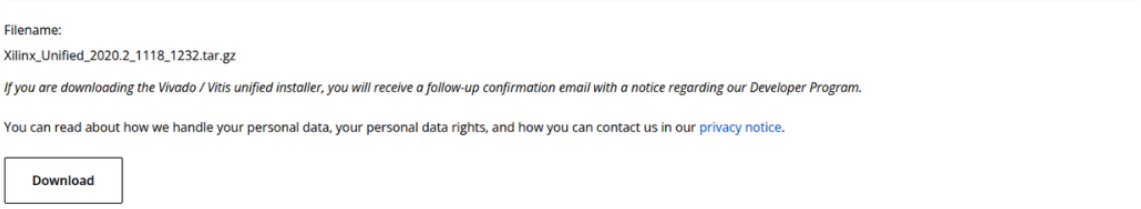
## 2.2 Install Vitis and Vivado 2020.2

1. Visit the download URL: https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/archive.html, then click the link below to download the installation package.



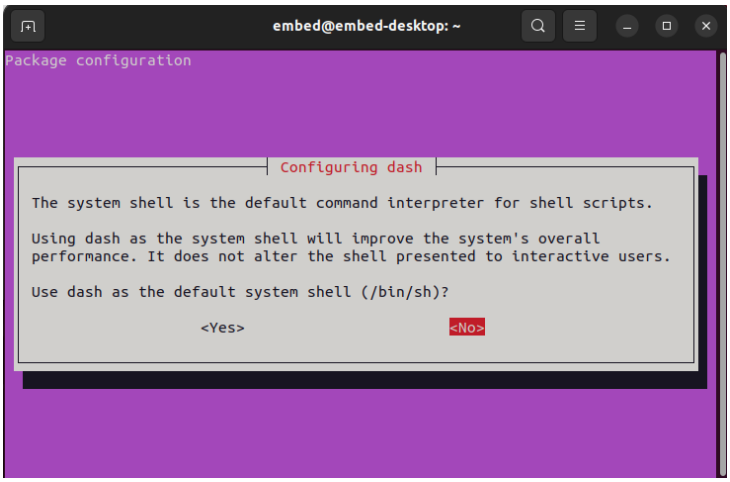

Please note that an AMD account is required to download this installation package. Please register and download it.


2. After logging in successfully, click Download. Once the download is complete, extract the compressed package and enter the directory.



3. Run the following command and select "No".

```
sudo dpkg-reconfigure dash
```

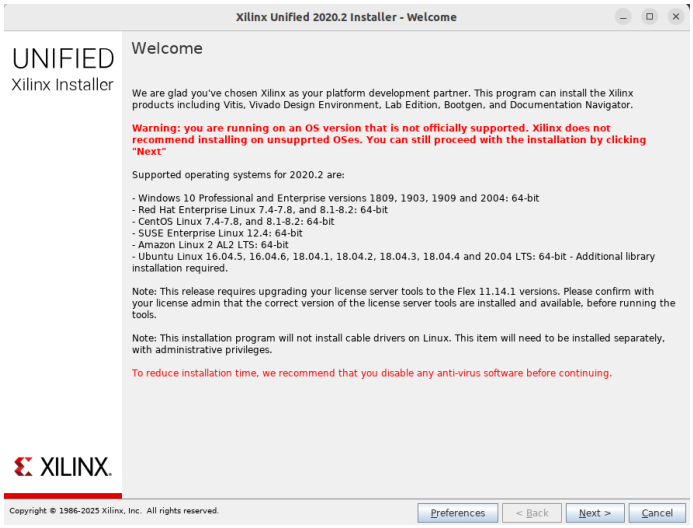



4. Execute the following commands to install Vivado.

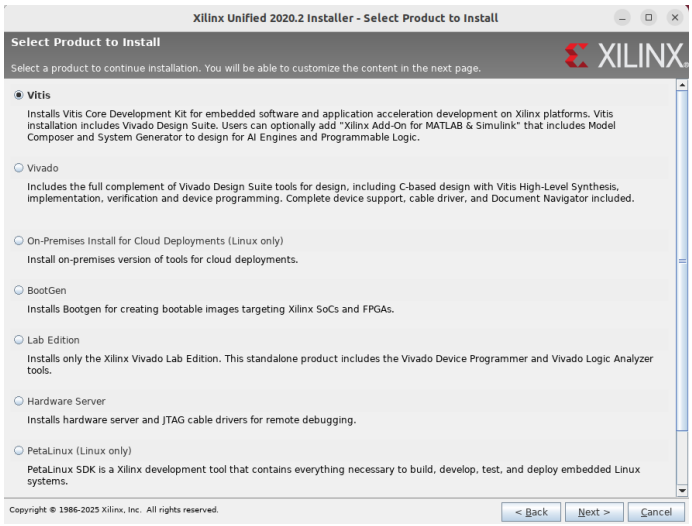
```
cd Xilinx_Unified_2020.2_1118_1232/
sudo ./xsetup
```

(1) We need to use Vivado 2020.2 version. Therefore, when the following prompt appears, select "Continue".

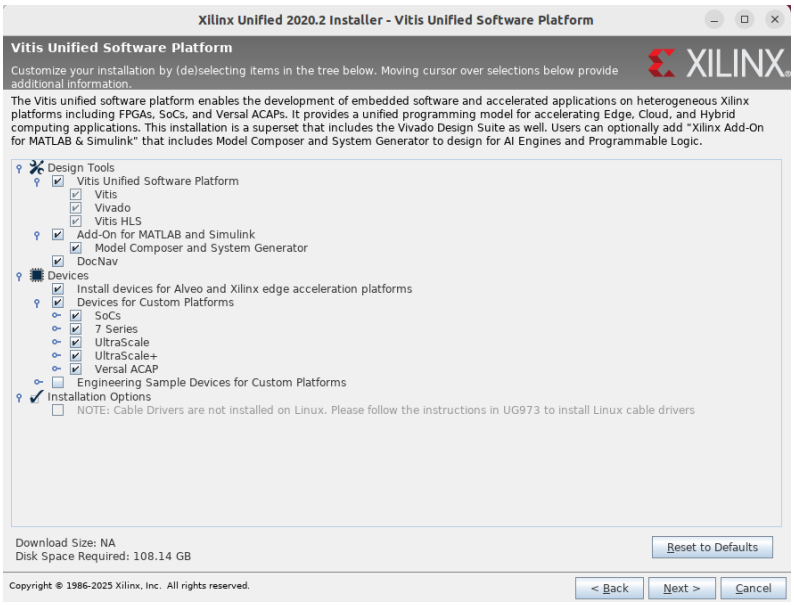

(2) Click "Next".



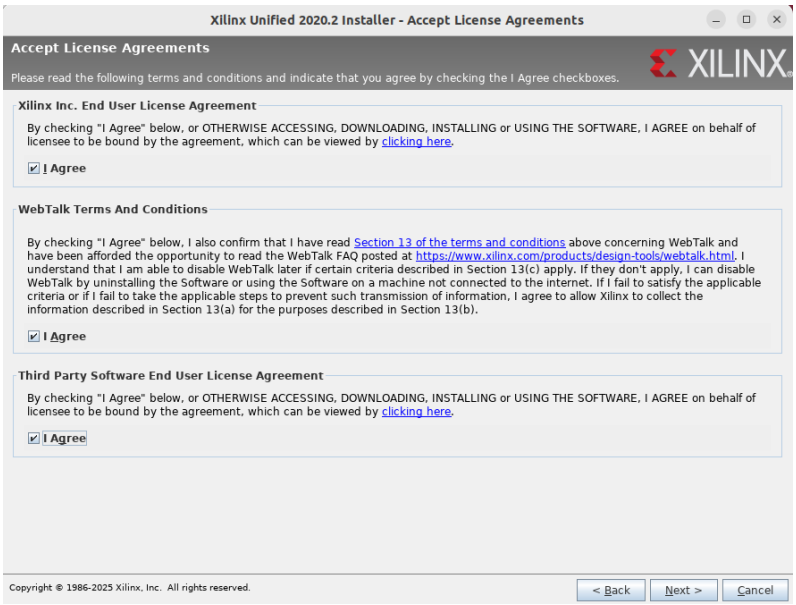

(3) Select "Vitis". Choosing this option will automatically install Vivado. Then, click "Next".



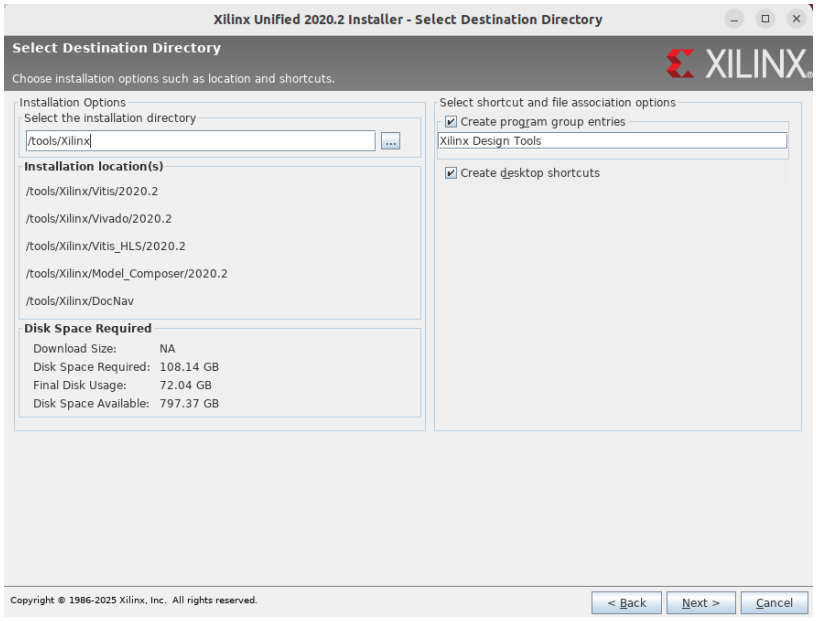

(4) Install the default components.



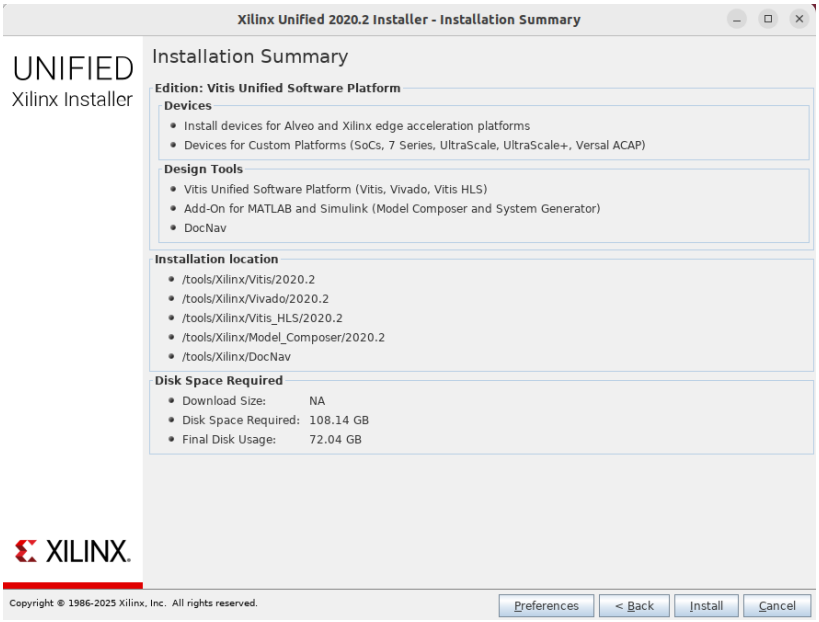

(5) Select all "I Agree" options, then click "Next".



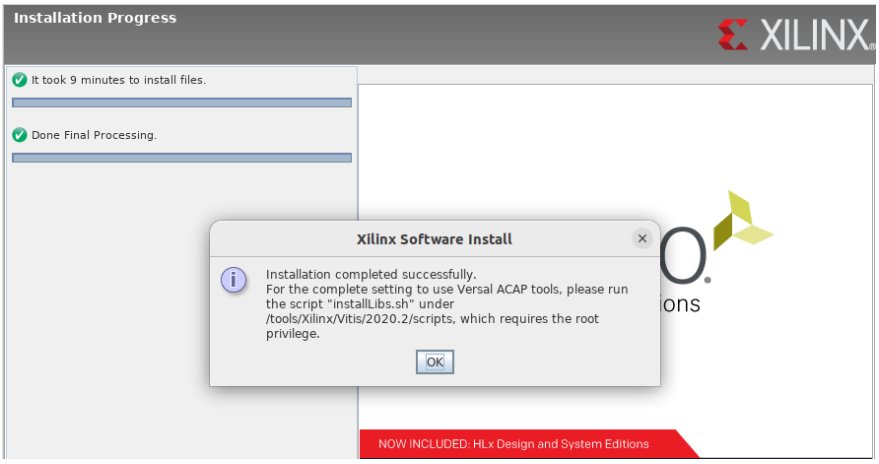
(6) Specify the installation location, then click "Next".




(7) Click "Install" to begin the installation.




(8) Wait for the installation to be completed.



5. Update the bashrc to enable launching Vivado/Vitis from the shell.

(1) Execute the following command to open the file ~/.bashrc.

```
vim ~/.bashrc
```

(2) Add the following content at the end of the file. The directory paths are on the installation location of the Vivado/Vitis software

```
source /tools/Xilinx/Vivado/2020.2/settings64.sh
source /tools/Xilinx/Vitis_HLS/2020.2/settings64.sh
export PATH=/usr/bin:$PATH
```

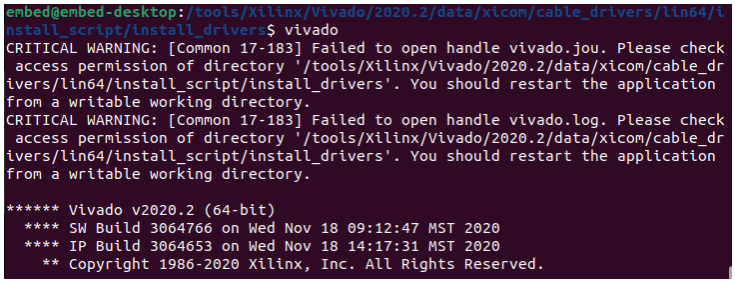
(3) Update the bashrc.

```
source ~/.bashrc
```

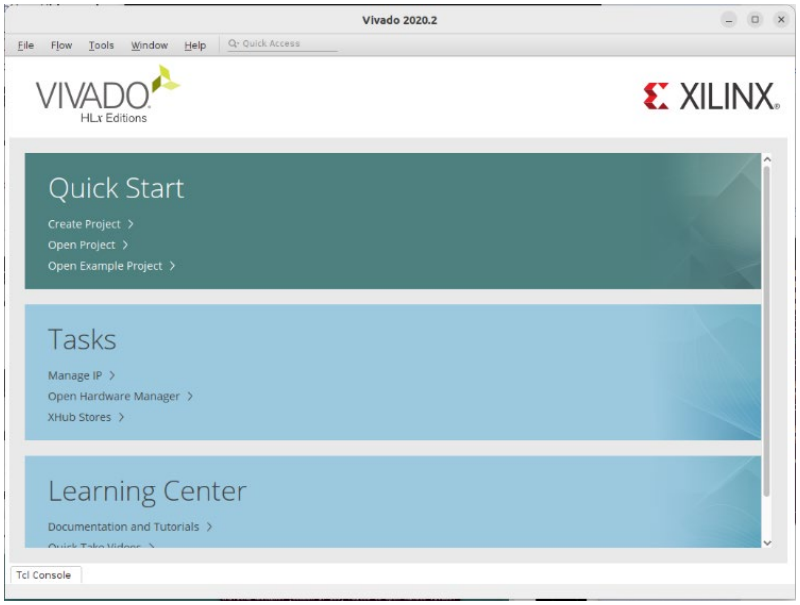
6. Install the USB driver. Open the terminal and enter the following two commands in sequence:

```
cd /tools/Xilinx/Vivado/2020.2/data/xicom/cable_drivers/lin64/install_script/install_drivers/
sudo ./install_drivers
```

7. Enter the command vivado in the terminal to launch the Vivado interface.

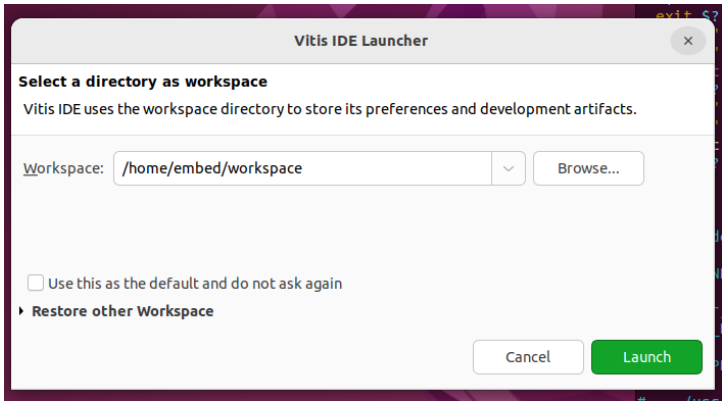



The displayed Vivado interface is as follows:




8. Enter the command vitis in the shell to open the Vitis interface.


The displayed interface is as follows:




# 3. Deploy the device-side runtime to the CSD

## 3.1 Overview of execution steps

1. First, insert the CSD into the PCIe slot on the motherboard of your PC, and connect the serial cable (for debug information output) and JTAG cable (for program downloading).
2. Execute steps 1-4 from "Section 3.2 Compilation and Execution Process".
3. Restart the PC.
4. Execute step 5 from "Section 3.2 Compilation and Execution Process".

## 3.2 Compilation and execution process

1. Download the source code from GitHub.

```
git clone https://github.com/duzelin/SoftSSD
```

2. Enter the soda directory and create a new workspace.

```
cd SoftSSD
mkdir workspace
```

3. Modify the content in the file SoftSSD/scripts/create_projects.tcl. Note that the XSA_FILE path on line 7 should be set to the directory path under the soda folder.

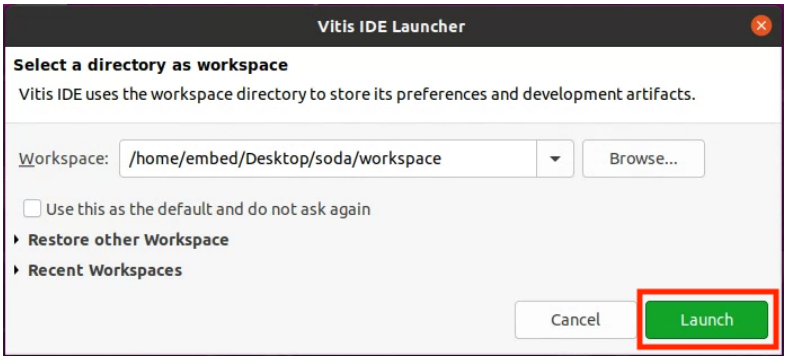
```
set PLATFORM_NAME hardnvme4core
set SYS_PROJ_NAME nvme_ftl_system
set FTL_PROJ_NAME nvme_ftl
set FIL_PROJ_NAME nvme_fil
set ECC_PROJ_NAME nvme_ecc

set XSA_FILE [Your XSA PATH]

catch {platform remove $PLATFORM_NAME}
```

```
catch {app remove $FTL_PROJ_NAME}
catch {app remove $FIL_PROJ_NAME}
catch {app remove $ECC_PROJ_NAME}

platform create -name $PLATFORM_NAME \
-hw $XSA_FILE \
-proc {psu_cortexa53_0} -os {standalone} -arch {64-bit} -fsbl-target {psu_cortexa53_0}

platform write
platform generate -domains
platform active {hardnvme4core}
bsp reload
bsp setlib -name xilffs
bsp write
bsp reload
catch {bsp regenerate}
platform active {hardnvme4core}
domain create -name {standalone_psu_cortexr5_0} -display-name {standalone_psu_cortexr5_0} -os
{standalone} -proc {psu_cortexr5_0} -runtime {cpp} -arch {32-bit} -support-app {empty_application}
domain create -name {standalone_psu_cortexr5_1} -display-name {standalone_psu_cortexr5_1} -os
{standalone} -proc {psu_cortexr5_1} -runtime {cpp} -arch {32-bit} -support-app {empty_application}
platform generate -domains
platform write
domain active {zynqmp_fsbl}
domain active {zynqmp_pmufw}
domain active {standalone_domain}
domain active {standalone_psu_cortexr5_0}
domain active {standalone_psu_cortexr5_1}
platform generate -quick
platform generate

# Create FTL project
app create -name $FTL_PROJ_NAME -sysproj $SYS_PROJ_NAME -platform $PLATFORM_NAME -domain
standalone_domain -template "Empty Application"
app config -name $FTL_PROJ_NAME -add include-path "\${workspace_loc:/$FTL_PROJ_NAME/include}"
app config -name $FTL_PROJ_NAME -add libraries m

# Create FIL project
app create -name $FIL_PROJ_NAME -sysproj $SYS_PROJ_NAME -platform $PLATFORM_NAME -domain
standalone_psu_cortexr5_0 -template "Empty Application"
app config -name $FIL_PROJ_NAME -add include-path "\${workspace_loc:/$FIL_PROJ_NAME/src}"
app config -name $FIL_PROJ_NAME -add include-path "\${workspace_loc:/$FTL_PROJ_NAME/include}"

# Create ECC project
app create -name $ECC_PROJ_NAME -sysproj $SYS_PROJ_NAME -platform $PLATFORM_NAME -domain
standalone_psu_cortexr5_1 -template "Empty Application"
app config -name $ECC_PROJ_NAME -add include-path "\${workspace_loc:/$ECC_PROJ_NAME/src}"
app config -name $ECC_PROJ_NAME -add include-path "\${workspace_loc:/$FTL_PROJ_NAME/include}"set
PLATFORM_NAME zu_onfi8
```

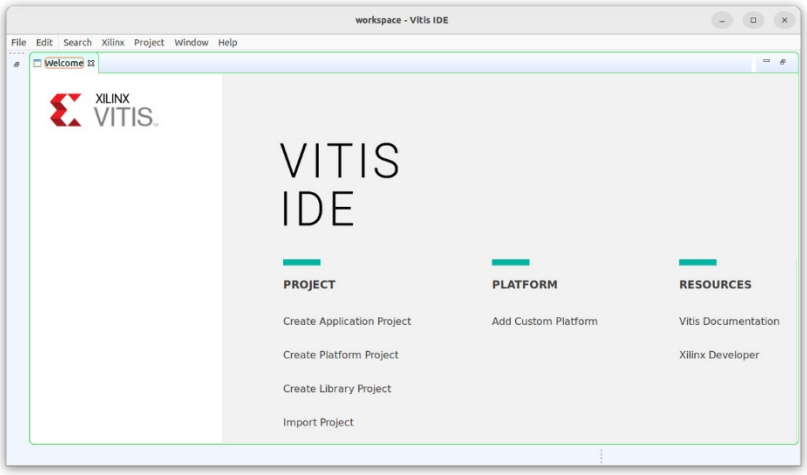
4. Download the Runtime to the CSD.

(1) Enter vitis in the Shell to launch the Vitis interface.

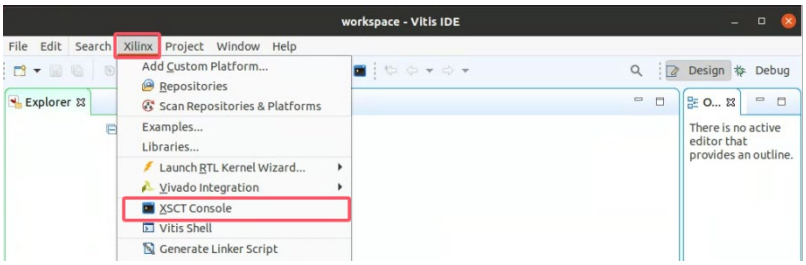

(2) Set the workspace to the newly created workspace directory under the soda folder, then click Launch.




After success, the following interface is displayed.

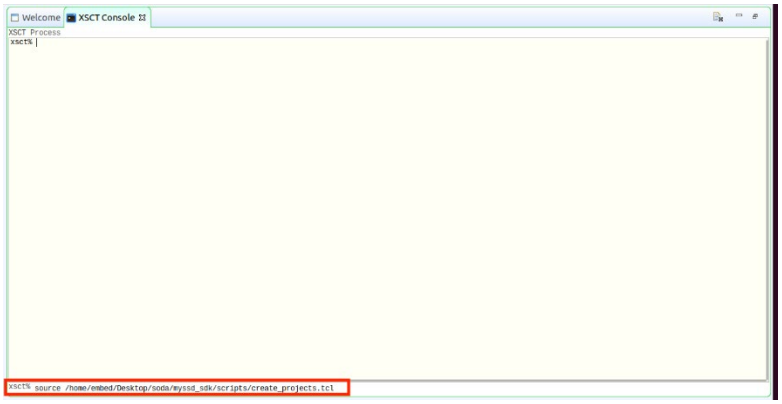



(3) Open the XSCT Console.




(4) In the XSCT shell, enter the following command, specifying the location of the create_projects.tcl file.

```
source /home/embed/Desktop/soda/myssd_sdk/scripts/create_projects.tcl
```

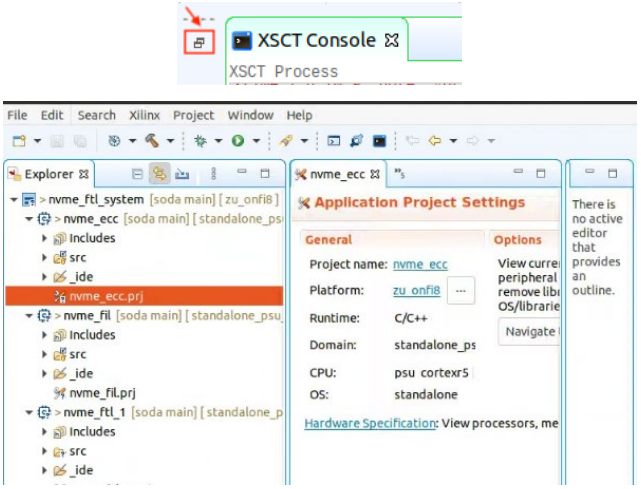



Run the command. Red warnings during execution can be ignored. Upon successful execution, the following will be displayed.

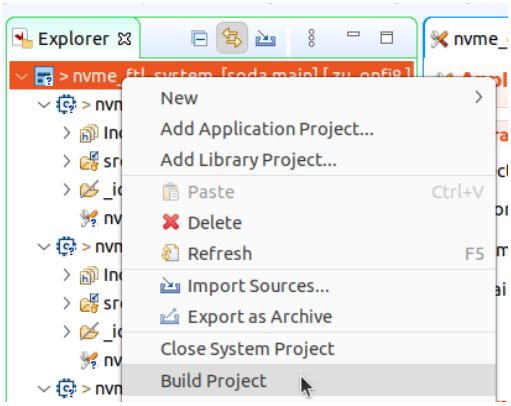

(5) Navigate to the directory SoftSSD and execute the following command.

```
cd ~/SoftSSD
./scripts/populate_workspace.sh ../workspace
```

Then click the leftmost interface expansion button. You will now see that all projects have been created.




(7) Compile the code: Right-click on nvme_ftl_system --> Build Project.




Compilation completed successfully.

(8) Right click the nvme_ftl_system project in Vitis, “Run As” -> “Run configurations…”, double click “System Project Debug” to add new a run configuration SystemDebugger_nvme_ftl_system. Under the “Target Setup” tab, deselect “Use FSBL flow for initialization" and select “Reset APU”, “Reset RPU”, “Enable RPU Split Mode”, “Run psu_init” and “PL Powerup”. Press “Run” to run the project.