AMD Tools and Win10 WSL

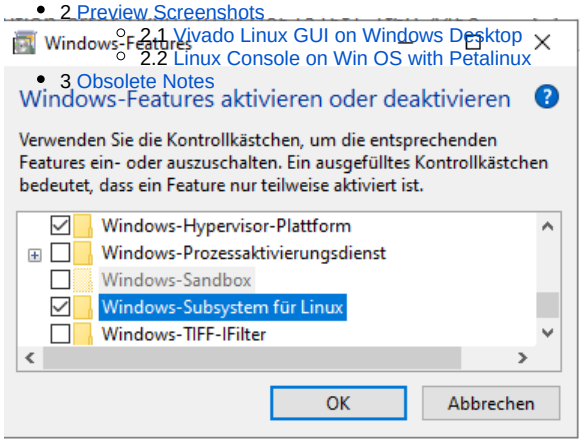
Documentation for 2023.2 can be changed permanently at the moment.
With the current status documentation, everything can be created and used under WSL but 2023.2 under WSL2 is currently being evaluated.

Instructions

Table of contents

Setup WSL and Install Linux

1. Enable CPU virtualisation in BIOS
2. Open Windows Features
   a. Enable Windows Subsystem for Linux



1. Powershell as Admin:
   a. **dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart**
   b. **dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart**
2. Reboot
3. Download and install: https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi
4. Powershell as Admin:
   a. **wsl --set-default-version 2**

The list is only a guide that has worked for us, there may still be complications that we cannot help with at this time.

- Vivado/Vitis/Petalinux 2023.2
  - with WSL Ubuntu 22.04 LTS
- Archive(was created with older version of this guideline)
  - Vivado/Vitis/Petalinux 2020.2
    - with WSL Ubuntu 18.04 LTS
  - Vivado/Vitis/Petalinux 2021.2/2022.2
    - with WSL Ubuntu 18.04 LTS
    - with WSL Ubuntu 20.04 LTS

Install Ubuntu Distribution

1. Open Powershell as Admin
   a. Show all available distributions **wsl.exe --list --online**

b. Install(Need for Multi Distributions): **wsl.exe --install -d <linux distribution>** or **wsl.exe --install <linux distribution>**

2. Start Linux console and follow install instruction
    a. Start Ubuntu App (Windows Start Button Ubuntu)
3. (optional) WSL Configuration on linux console:
    a. **sudo vim /etc/wsl.conf**
        i. [wsl2]
        memory=<size>GB
        processors=<cnt>
    b. Other options: https://docs.microsoft.com/en-us/windows/wsl/wsl-config
4. Other Install options from MS:https://docs.microsoft.com/de-de/windows/wsl/install-manual

Mount external ext4 drive (optional)

1. mount ext4 drive (optimal as working drive instead if vhs): https://learn.microsoft.com/en-us/windows/wsl/wsl2-mount-disk
    a. Mounting a partitioned disk
        i. Show drives on powershell: **GET-CimInstance -query "SELECT * from Win32_DiskDrive"**
        ii. Mount drive inside powershell**: wsl --mount <DiskPath> -p <partition number>**
        iii. Linux Console
            1. show available partitions: **lsblk -lf**
            2. Mount drive inside linux**:**
                a. **sudo mkdir /mnt/<name>**
                b. **sudo mount /dev/sdd<Number> /mnt/<name>/**
            3. Change Permission
                a. **sudo chown <owner>:<owner> /mnt/<name>/**
    b. todo automount (otherwise ex4 is unmounted after wsl shutdown)
        i. https://learn.microsoft.com/de-de/windows/wsl/wsl-config

1. Check that the DNS server is configured on the Linux console:
    a. **sudo vim /etc/resolv.conf**
2. In case DNS server is correct, skip this instruction otherwise follow instruction below
3. Linux console:
    a. **sudo vim /etc/wsl.conf**
        i. [network]
        generateResolvConf = false
    b. (in case file can't be wrote):
        i. **sudo rm /etc/.wsl.conf.swp**
4. Powershell console:
    a. **wsl --shutdown**
5. Get DNS Server IP over powershell:
    a. **ipconfig -all**
6. Restart Linux (over powershell or win start button Ubuntu)
7. Linux console:
    a. **sudo vim /etc/resolv.conf**
        - nameserver <DNS IP>
    b. (in case file can't be wrote, remove) and try again point a:
        i. **sudo rm /etc/.resolv.conf.swp**
        ii. **sudo rm etc/resolv.conf**

1. Open explore
2. Right click on PC icon --> Connect network drive
3. Select drive
4. Add to folder \\wsl$\Ubuntu-18.04
5. Click Finish

Prepare Linux for AMD Tools

1. Resize WSL VHD, see "Resize WSL VHD" on "Notes and Hints and other stuff" section
2. Install browser
    a. **sudo apt-get install firefox**
3. change console from dash to bash

    a. **sudo dpkg-reconfigure dash**

i. Press "No" to disable dash and activate default bash

4. Enable i386 architecture
    a. **sudo dpkg --add-architecture i386**
       sudo apt-get update
5. ~~Download and run plnx-env-setup.sh(optional, not longer recommended from AMD): https://www.xilinx.com/support/answers/73296.html~~
    a. ~~**chmod 777 ./plnx-env-setup.sh**~~
    b. ~~**sudo ./plnx-env-setup.sh**~~
6. Install packages(see https://support.xilinx.com/s/article/000035572?language=en_US PetaLinux_2023.2_OS_Package_List.xlsx)
    a. Note:
        i. From excel additional to python3 also python was recommended which is not longer possible to install
        ii. Additionally bc and was libtinfo5 added
        iii. Add subversion (need only in case svn is used)
        iv. Add u-boot-tools (need to generate own boot.scr file instead of petalinux version)
    b. **sudo apt-get install iproute2 gawk python3 build-essential gcc git make net-tools libncurses5-dev tftpd zlib1g-dev libssl-dev flex bison libselinux1 gnupg wget git diffstat chrpath socat xterm autoconf libtool tar unzip texinfo zlib1g-dev gcc-multilib automake zlib1g:i386 screen pax gzip cpio python3-pip python3-pexpect xz-utils debianutils iputils-ping python3-git python3-jinja2 libegl1-mesa libsdl1.2-dev pylint bc libtinfo5 subversion u-boot-tools -y**

1. Download AMD Vitis All OS installer Single-File(Includes Vitis, Vivado and Petalinux downloader):
    a. Downloads (xilinx.com)
2. Extracted Files on linux drive
    a. **tar -xvf FPGAs_AdaptiveSoCs_Unified_2023.2_1013_2256.tar.gz**
3. Create Folder /tools/Xilinx
    a. **sudo mkdir /tools**
    b. **sudo mkdir /tools/Xilinx**
4. Change owner to user
    a. **sudo chown <owner>:<owner> /tools/Xilinx**
5. Install Vitis
    a. run:$ **./xsetup**
    b. select Vitis on default installation path
6. Vivado License
    a. Open Xilinx License manager and chose Copy License and select your license file
    b. In case license file is node-locked and shared with WinOS:
        i. **sudo vim /etc/bash.bashrc**
            1. sudo ip link add bond0 type bond
                a. in case bond0 is not available
            2. sudo ip link set dev bond0 address xx:xx:xx:xx:xx:xx
                a. (use MAC from Win OS)
7. Install petalinux
    a. run:$ **./xsetup**
    b. select petalinux on default installation path
8. Add source environment to auto start of new console :
    a. **sudo vim /etc/bash.bashrc**
        i. source /tools/Xilinx/PetaLinux/2023.2/tool/settings.sh

Notes and Hints and other stuff

sudo vim /etc/bash.bashrc (complete example)

```
#...
#(original part)
#...
#all extensions to setup environment and
# load petalinux environment, can be overwrite via project scripts
export LM_LICENSE_FILE=<path to license file>
export GLOBAL_VIVADO=2023.2
source /tools/Xilinx/PetaLinux/${GLOBAL_VIVADO}/tool/settings.sh

# additional TE environment variables to change timeout and used jobs

# additional TE envirnmoent variables
export TE_TIMEOUT=200
export TE_RUNNING_JOBS=16
export TE_WSL_USAGE=1
export TE_EDITOR=notepad++.exe
#export TE_SERIAL_PS=<path>
export TE_COM=/mnt/<path to putty on winOS>
export TE_PLX_SSTATE_CACHE_DOWNLOAD=~/design/sstate-cache
/downloads_${GLOBAL_VIVADO}/downloads
export TE_PLX_SSTATE_CACHE_AARCH64=~/design/sstate-cache
/sstate_aarch64_${GLOBAL_VIVADO}/aarch64
export TE_PLX_SSTATE_CACHE_ARM=~/design/sstate-cache
/sstate_arm_${GLOBAL_VIVADO}/arm
export TE_PLX_SSTATE_CACHE_MB_FULL=~/design/sstate-cache/sstate_mb-
full_${GLOBAL_VIVADO}/mb-full
# finish
echo "TE Enviroment Variables"
echo "TE_TIMEOUT=${TE_TIMEOUT}"
echo "TE_RUNNING_JOBS=${TE_RUNNING_JOBS}"
echo "TE_WSL_USAGE=${TE_WSL_USAGE}"
echo "TE_EDITOR=${TE_EDITOR}"
echo "TE_SERIAL_PS=${TE_SERIAL_PS}"
echo "TE_COM=${TE_COM}"
echo "TE_PLX_SSTATE_CACHE_DOWNLOAD=${TE_PLX_SSTATE_CACHE_DOWNLOAD}"
echo "TE_PLX_SSTATE_CACHE_AARCH64=${TE_PLX_SSTATE_CACHE_AARCH64}"
echo "TE_PLX_SSTATE_CACHE_ARM=${TE_PLX_SSTATE_CACHE_ARM}"
echo "TE_PLX_SSTATE_CACHE_MB_FULL=${TE_PLX_SSTATE_CACHE_MB_FULL}" # finish

function x_run () {
  sudo ip link add bond0 type bond
  # replace xx:xx:xx:xx:xx:xx with win 10 MAC
  sudo ip link set dev bond0 address xx:xx:xx:xx:xx:xx
}

# xSetup only needed in case wsl is rebooted
echo "Refresh XSetup? y/N"
read xsetup
if [ "${xsetup}" == "y" ]; then x_run; fi

#show version
lsb_release -a
```

Release reserved memory from Vmmem

```
#Sometimes Vmmem did not release DDR memory
# ################
#Solution 1:
# #######
#shutdown (close all running application before)
wsl --shutdown
#start WSL again
wsl
#Solution 2:
#todo
```

Resize WSL VHD

```
# Resize VHD
# ###############
# see: https://docs.microsoft.com/en-us/windows/wsl/vhd-size
#
#1 open powershell as admin
#Terminate WSL
wsl --shutdown
# https://learn.microsoft.com/en-us/windows/wsl/disk-space#how-to-locate-
the-vhdx-file-and-disk-path-for-your-linux-distribution
#find your distribution installation.
#replace <distribution-name> with your Ubuntu Version, for example Ubuntu-
22.04
(Get-ChildItem -Path HKCU:\Software\Microsoft\Windows\CurrentVersion\Lxss
| Where-Object { $_.GetValue("DistributionName") -eq '<distribution-name>'
}).GetValue("BasePath") + "\ext4.vhdx"

#Copy the path to that file, it should look something like %LOCALAPPDATA%
\Packages\CanonicalGroupLimited.Ubuntu20.
04onWindows_79xxxx\LocalState\ext4.vhdx
# This path will be called "<pathToVHD>" below
# -------------------------#start diskpartdiskpart# ----------
diskpart
#inside diskpart console
#select disk
DISKPART> Select vdisk file="<pathToVHD>"
#check details
DISKPART> detail vdisk
#(optional) shrink size
DISKPART> compact vdisk
#expand for example to 512GB  <sizeInMegaBytes>=512000
#Note: WSL2 and Ubuntu 22.04 has default maximum size from harddisk
DISKPART> expand vdisk maximum=512000
#exit
DISKPART> exit
# ----------
#start WSL from powershell
sudo mount -t devtmpfs none /dev
#  "/dev: none already mounted on /dev." warning can be ignored
mount | grep ext4
#resize fs to 512GB <sizeInMegaBytes>=512000M
sudo resize2fs /dev/sdb 512000M
```

Warning "/bin/bash: warning: setlocale: LC_ALL: cannot change locale (en_US.UTF-8)

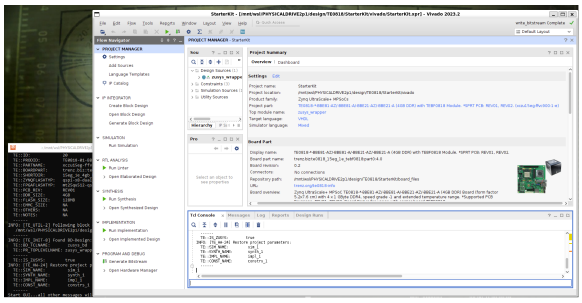
```
#change manually with
sudo dpkg-reconfigure locales
```

Other Trenz Documentation around AMD Tool

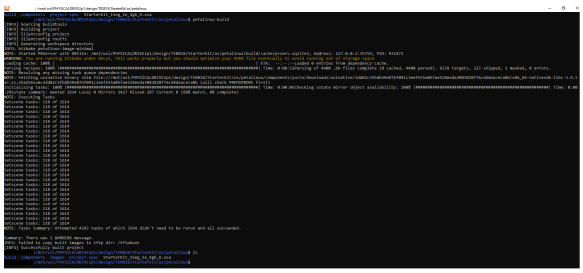
- AMD Development Tools

Preview Screenshots

Vivado Linux GUI on Windows Desktop



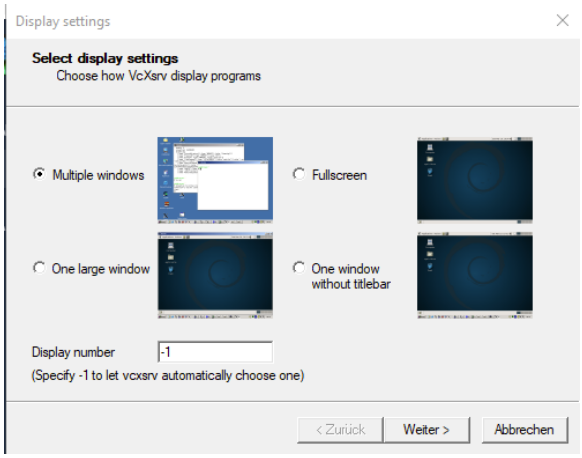
Linux Console on Win OS with Petalinux



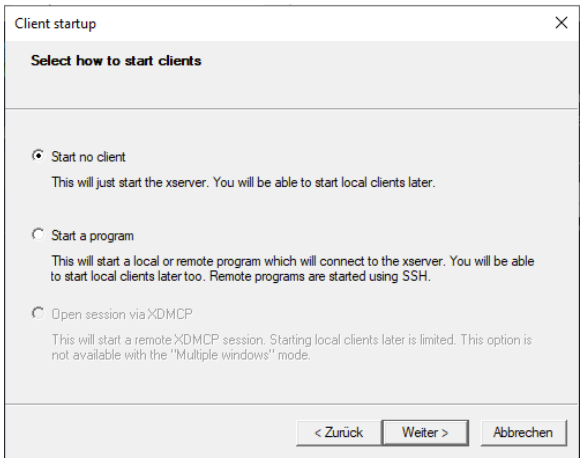
Obsolete Notes

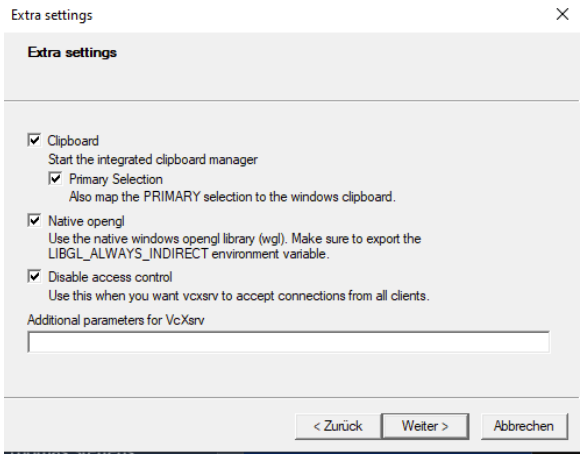
> ⚠ VCXsrv is not longer needed with newest WSL und Ubuntu 22.04

1. Install VcXsrv
2. Start VcXsrv (XLaunch)
    a.



b.





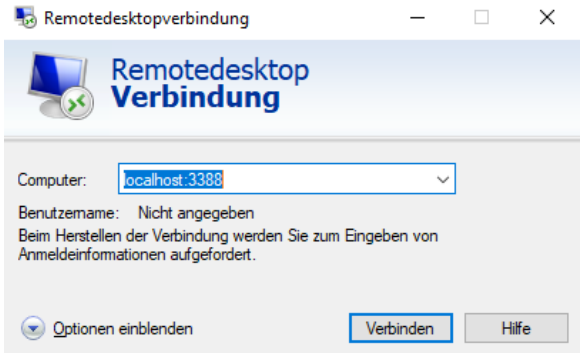
c. (optional) Save Configuration and run XLaunch directly with "XLaunch -run <config. xlaunch>"

3. Linux console:
    a. **sudo apt update && sudo apt upgrade**
    b. **sudo apt install xfce4**
    c. **sudo apt install build-essential**
    d. **sudo apt install net-tools**
    e. **sudo apt install xrdp -y && sudo systemctl enable xrdp**
4. Add environment to auto start of new console :
    a. **sudo vim /etc/bash.bashrc**
        i. export DISPLAY=<host IP>:0.0
        ii. export LIBGL_ALWAYS_INDIRECT=1
        iii. sudo /etc/init.d/dbus start
        iv. sudo /etc/init.d/xrdp start

⚠ Currently not tested with newest WSL und Ubuntu 22.04

1. Start Ubuntu App (Windows Start Button Ubuntu)
2. Add username and password
3. Update packet manager:
    a. **sudo apt update && sudo apt upgrade**

4. Install:
    a. **sudo apt install xfce4 xrdp**
5. Modify Remote Port
    a. **sudo sed -i 's/3389/3388/g' /etc/xrdp/xrdp.ini**
6. Start XRDP Server
    a. **sudo /etc/init.d/xrdp start**
7. (optional) XRDP autostart :
    a. **sudo vim /etc/bash.bashrc**
        i. **sudo /etc/init.d/xrdp start**
8. Connect via RDP (Win10 Remote Desktop Connection):
    a. localhost:3388



sudo vim /etc/bash.bashrc (complete example)

```
#...
#(original part)
#...
#all extensions to setup environment and
# load petalinux environment, can be overwrite via project scripts
export GLOBAL_VIVADO=2021.2
source /tools/Xilinx/PetaLinux/${GLOBAL_VIVADO}/tool/settings.sh

# additional TE environment variables to change timeout and used jobs

# additional TE envirnmoent variables
export TE_TIMEOUT=200
export TE_RUNNING_JOBS=16
export TE_WSL_USAGE=1
export TE_EDITOR=notepad++.exe
#export TE_SERIAL_PS=<path>
export TE_COM=/mnt/<path to putty on winOS>
export TE_PLX_SSTATE_CACHE_DOWNLOAD=~/design/sstate-cache
/downloads_${GLOBAL_VIVADO}/downloads
export TE_PLX_SSTATE_CACHE_AARCH64=~/design/sstate-cache
/sstate_aarch64_${GLOBAL_VIVADO}/aarch64
export TE_PLX_SSTATE_CACHE_ARM=~/design/sstate-cache
/sstate_arm_${GLOBAL_VIVADO}/arm
export TE_PLX_SSTATE_CACHE_MB_FULL=~/design/sstate-cache/sstate_mb-
full_${GLOBAL_VIVADO}/mb-full
# finish
echo "TE Enviroment Variables"
echo "TE_TIMEOUT=${TE_TIMEOUT}"
echo "TE_RUNNING_JOBS=${TE_RUNNING_JOBS}"
echo "TE_WSL_USAGE=${TE_WSL_USAGE}"
echo "TE_EDITOR=${TE_EDITOR}"
echo "TE_SERIAL_PS=${TE_SERIAL_PS}"
echo "TE_COM=${TE_COM}"
```

```
echo "TE_PLX_SSTATE_CACHE_DOWNLOAD=${TE_PLX_SSTATE_CACHE_DOWNLOAD}"
echo "TE_PLX_SSTATE_CACHE_AARCH64=${TE_PLX_SSTATE_CACHE_AARCH64}"
echo "TE_PLX_SSTATE_CACHE_ARM=${TE_PLX_SSTATE_CACHE_ARM}"
echo "TE_PLX_SSTATE_CACHE_MB_FULL=${TE_PLX_SSTATE_CACHE_MB_FULL}" # finish

function x_run () {
  #XServer Display
  export DISPLAY=xxx.xxx.xxx.xxx:0.0
  # replace xxx.xxx.xxx.xxx with your win 10 host IP
  export LIBGL_ALWAYS_INDIRECT=1

  sudo /etc/init.d/dbus start
  sudo /etc/init.d/xrdp start
  # replace xx:xx:xx:xx:xx:xx with win 10 MAC
  sudo ip link set dev bond0 address xx:xx:xx:xx:xx:xx
}

# x-server Setup only needed in case wsl is rebooted
echo "Refresh XSetup? y/N"
read xsetup
if [ "${xsetup}" == "y" ]; then x_run; fi

#show version
lsb_release -a
```