Verification of Digital Systems (02207)

Contents

1 Introduction

This document describes the steps for:

1. Set up the Linux environment on a non-Linux machine, using *WSL* (only for Windows users) or *Docker-Desktop*, section 2.
2. Set up the project environment on the Linux environment, section 3.

The project requires a Linux environment and the following tools will be used:

- GTKWave: Wave viewer.
- Icarus Verilog: HDL compiler and simulator.
- Python version 3.10.0

For Windows users, a Linux environment can be created using one of the following platforms:

- **Windows Subsystem for Linux (*WSL*)**: it is a feature of Windows that allows you to run a Linux environment on your Windows machine, without the need for a separate virtual machine or dual booting. [1]
- ***Docker***: It is a platform that allows to create containers to run on the host operating system, keeping all the installed applications or configuration in the container independent of the ones available in the host. [2]
  - ***Docker Desktop***: It is a one-click-install application for Mac, Linux, or Windows environment that allows the usage of the *Docker* platform. It provides a straightforward graphical user interface (GUI), facilitating the management of containers, applications, and images directly from the host machine. [3] [4]
  - ***Docker* container**: It is an isolated environment where an application runs without affecting the rest of the host system and without the host system impacting the application.
  - ***Docker* image**: It is read-only template containing instructions for creating a container. A *Docker* image creates containers to run on the *Docker* platform.

HINT

Some commands and file paths in the following sections contain values in angle brackets (`<...>`). These are placeholders for values that will vary for each machine.

For example

- `C:\Users\<WINDOWS-USERNAME>` denotes the specific windows file path to the users home directory, e.g. `C:\Users\syosil`
- `<ROOT>` specifies the location of the project folder, e.g. `C:\Users\syosil\Documents\syosil-project`.

2 Linux Setup

This section describes how to set up the Linux environment and installing all the tools required for the project. Based on the host operative system, one of the following methods must be followed:

- For ***Windows*** users, there are 2 methods to set up the Linux environment.
  - Installing through *WSL*, see section 2.1.
  - Installing through *Docker*, using *Docker-Desktop* application, see Section 2.2.
- For ***Linux*** users, skip this section and go to Section 3.

2.1 *WSL* v2

The following instructions will setup the *Ubuntu 22.04* on *WSL*.

IMPORTANT

- *WSL* version 2 is required.
- If *WSL* is already installed in the machine run, `wsl --version`, to check which version is installed.
  - Output should contain, `WSL version: 2.*.*.*` .
  - If this is not the case the command, `wsl --set-version`, can be used to update the version.

1. Open **Terminal** app (or **PowerShell**) on Windows.
2. Install ***WSL* 'Ubuntu-22.04'** distribution (choose one **a** or **b**):

   IMPORTANT

   - During the installation (or after the reboot) you will be asked to set a **username** and **password**.
   - These will be your Linux credentials in *WSL*, referred now on as <**WSL-USERNAME**> and <**WSL-PASSWORD**>
   - **!! Make sure to save these credentials !!**

   (a) If the ***WSL* environment** is not installed in the host, run:

   ```
   C:\Users\<WINDOWS-USERNAME> > wsl --install -d Ubuntu-22.04
   C:\Users\<WINDOWS-USERNAME> > wsl --update

   # A reboot might be required, if the following message shows up:
   # "The requested operation is successful. Changes will not be effective until the system is rebooted."
   ```

   i. After reboot the terminal might open automatically continuing the installation of the ***WSL* environment**. After that, you will be already in the 'home' directory of your WSL environment.

   ```
   # !!! Notice that your prompt has changed !!
   # You are now inside the WSL environment in the 'home' folder of your WSL system
   <WSL-USERNAME>@wsl-server:~/$
   ```

(b) If ***WSL*** is already installed in the host, run:

```
# Install the WSL environment
C:\Users\<WINDOWS-USERNAME> > wsl --install -d Ubuntu-22.04
C:\Users\<WINDOWS-USERNAME> > wsl --update

# Start WSL by running
C:\Users\<WINDOWS-USERNAME> > wsl
# (OR) If you have other Linux distributions installed, use the following command
C:\Users\<WINDOWS-USERNAME> > wsl --distribution Ubuntu-22.04

# !!! Notice that your prompt has changed !!
# You are now inside the WSL environment
<WSL-USERNAME>@wsl-server:/mnt/c/Users/<WINDOWS-USERNAME>$

# The default directory is the 'home' folder of your host.
# To move to the 'home' folder of your WSL system, run:
<WSL-USERNAME>@wsl-server:/mnt/c/Users/<WINDOWS-USERNAME>$ cd
<WSL-USERNAME>@wsl-server:~/$
```

3. Inside the *WSL* environment, run the following commands to complete the Ubuntu installation:

```
<WSL-USERNAME>@wsl-server:~$ sudo apt -y update
<WSL-USERNAME>@wsl-server:~$ sudo apt -y upgrade
<WSL-USERNAME>@wsl-server:~$ sudo apt install -y --no-install-recommends
```

4. Go to Section 3, and install the required software inside the *WSL* environment.

2.2 *Docker Desktop* setup

This section describes the steps for:

- installing *Docker Desktop*;
- installing the requirements to access to the GUI of an application running in the *Docker* container;
- creating and running the *Docker* container.

2.2.1 *Docker* Desktop: Installation

1. Download *Docker Desktop* from the official website, `https://docs.docker.com/desktop/install/windows-install/`.
2. Open the downloaded file and run the installer.
3. After successful installation, open **Powershell** and get the *Docker* image, `Ubuntu v.22:04`, by running:

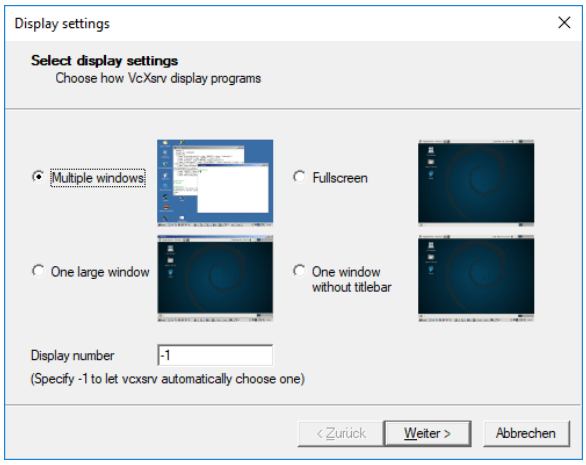
```
C:\Users\<WINDOWS-USERNAME> > docker pull ubuntu:22.04
```

Once the image is pulled you can check it by using the following command, which will list all the *Docker* images available on your machine. You can also check the same in *Docker-Desktop* application, under the *Images* tab.

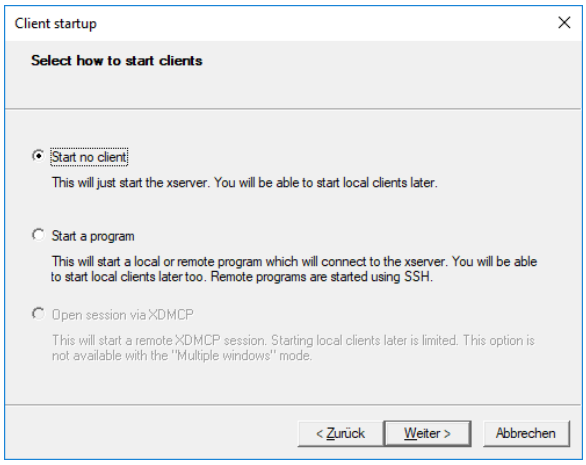
```
C:\Users\<WINDOWS-USERNAME> > docker images
```

2.2.2 *Docker* Desktop: Enable applications interface inside the *Docker* container

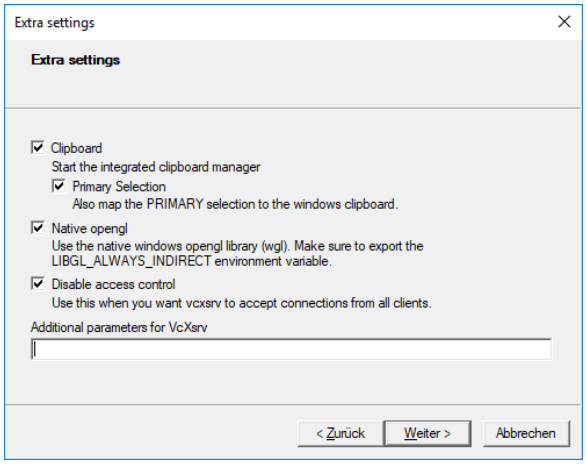
1. Download ***VcXsrv Windows X Server*** from, `https://sourceforge.net/projects/vcxsrv/`.
2. Open the file downloaded and run the installer.
3. After successful installation, open **Xlaunch** from the start menu, and follow the configuration in Figure 2.1.
    - Make sure to *Disable access control* (see Figure 2.13).
    - **Save the configuration file before you click finish!** (see Figure 2.14)



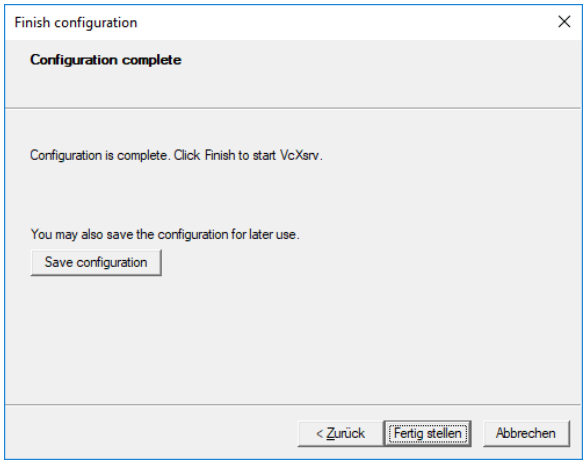
(1) Select "Multiple windows" and click "Next".



(2) Select "Start no client" and click "Next".



(3) Select all options and click "Next".



(4) Click on "Save configuration" and then click "Next".

Fig. 2.1: Configuration steps for **XLaunch**

2.2.3 *Docker* Desktop: Create and run *Docker* container

1. Create a folder on your machine that will be shared with the *Docker* container and note the file path e.g.: `C:\Users\<WINDOWS-USERNAME>\Documents\syosil_project` .
2. Locate the ***Dockerfile*** in `<ROOT>\client_setup\scripts`. [1]
3. Open **Powershell** in the `scripts`-folder and run the following command to build the *Docker container*. If you have problems copy-pasting the command into **Powershell**, it can be copied from the file `setup_command_build.sh`.

```
docker build `
-f Dockerfile `
--target builder-icarus `
--tag syosil-icarus-img `
--target syosil-base `
--build-arg USERNAME=$env:username-docker `
--tag syosil-ubuntu-container `
. ; `
docker image prune -f
```

[1] `<ROOT>` specifies for the file path of the project folder (e.g. `C:\Users\syosil\Documents\syosil-project`).

4. Then, run the following command to create the *Docker container*. If you have problems copy-pasting the command into **Powershell**, it can be copied from `setup_command_run.sh`.

IMPORTANT

- Replace `<C:\Users\<WINDOWS-USERNAME>\Documents\syosil-project>` by the full path of the folder created in step 1.
- Replace `<syosil-project>` the name of the created folder.

```
docker run -it `
-v ("<C:\Users\WINDOWS-USERNAME\Documents\syosil-project>" + `
    ":/home/" + $env:username + "-docker/<syosil-project>") `
-e DISPLAY=host.docker.internal:0.0 `
--name syosil-ubuntu-container `
syosil-ubuntu-container
```

5. To stop the container run `exit` in the *Docker* shell. To start the container again run:

```
C:\Users\<WINDOWS-USERNAME> > docker start syosil-ubuntu-container
C:\Users\<WINDOWS-USERNAME> > docker attach syosil-ubuntu-container
```

6. The container is set up and all the software installed. Section 3 can be disregarded.
7. It is possible to access the docker container in Visual Studio Code. See `https://code.visualstudio.com/docs/devcontainers/attach-container` for more information.

3 Software Setup

Here will be described all steps needed to install all the required tools in the Linux environment.

3.1 Icarus Verilog

Installation steps for Icarus Verilog.

```
# Install icarus requirements
<UNIX-USERNAME>@unix-server:~/$ sudo apt install -y make automake autoconf \
bison flex gperf build-essential g++ \
libreadline-dev git zlib1g zlib1g-dev

# Create temporary folder for icarus
<UNIX-USERNAME>@unix-server:~/$ mkdir -p /tmp/iverilog

# Get icarus
<UNIX-USERNAME>@unix-server:~/$ git clone --branch=v12_0 \
https://github.com/steveicarus/iverilog.git \
--depth 1 --single-branch /tmp/iverilog

# Build icarus
<UNIX-USERNAME>@unix-server:~/$ cd /tmp/iverilog
<UNIX-USERNAME>@unix-server:/tmp/iverilog$ autoconf && ./configure && make check \
&& sudo make install
<UNIX-USERNAME>@unix-server:/tmp/iverilog$ cd
```

3.2 Python v3.10

Installation steps for Python v3.10.

```
# Install Python 3.10
<UNIX-USERNAME>@unix-server:~/$ sudo apt install -y python3.10-minimal \
python3.10-venv python3.10-dev python3-pip

# For non native Linux users this is required by pyucis-viewer
<UNIX-USERNAME>@unix-server:~/$ sudo apt install libxcb-xinerama0 \
libqt5x11extras5
```

3.3 GTKWave

Installation steps for GTKWave.

```
|\UnixCommand{~}| sudo apt install -y gtkwave
```

3.4 Recommended tools

- VSCode, can be downloaded from `https://code.visualstudio.com/download`.

References

[1] Microsoft, "What is the Windows Subsystem for Linux?" [Online]. Available: https://learn.microsoft.com/en-us/windows/wsl/about

[2] Docker, "Docker overview." [Online]. Available: https://docs.docker.com/get-started/overview/

[3] ——, "Overview of Docker Desktop." [Online]. Available: https://docs.docker.com/desktop/

[4] ——, "Install Docker Desktop on Windows." [Online]. Available: https://docs.docker.com/desktop/install/windows-install/

A Appendices

A.1 Troubleshooting

A.1.1 GUI is not showing in docker

If no GUI shows up when starting *gtkwave* or the *pyucis-viewer* from the docker container, consider the following troubleshooting steps:

- Check if **XLaunch** is running.
- Check if **XLaunch** has been configured according to subsubsection 2.2.2. Especially if access control has been disable in the **XLaunch** setting.
- Check if `echo $DISPLAY` returns your IP address from step one in subsubsection 2.2.3.

A.2 Useful *WSL* commands

```
C:\Users\<WINDOWS-USERNAME> > wsl --install -d Ubuntu-22.04     # Install Ubuntu-22.04 on WSL
C:\Users\<WINDOWS-USERNAME> > wsl --distribution Ubuntu-22.04   # Start WSL
C:\Users\<WINDOWS-USERNAME> > wsl.exe --system -d Ubuntu-22.04 df -h /mnt/wslg/distro   # See disk
usage of WSL
C:\Users\<WINDOWS-USERNAME> > wsl --unregister Ubuntu-22.04     # Remove distribution

<WSL-USERNAME>@wsl-server:~/$ exit          # Exit from WSL environment by running 'exit'
```

A.3 Useful *Docker* commands

```
<WSL-USERNAME>-docker@docker-container:~/$ docker --help       # Docker help

<WSL-USERNAME>-docker@docker-container:~/$ sudo docker images  # List Docker images
<WSL-USERNAME>-docker@docker-container:~/$ sudo docker ps -a   # List all containers

<WSL-USERNAME>-docker@docker-container:~/$ docker start --attach <container_name>  # Start a container
<WSL-USERNAME>-docker@docker-container:~/$ docker attach <container_name>          # Attach to the
current terminal a running container
<WSL-USERNAME>-docker@docker-container:~/$ docker stop <container_name>            # Stop a running
container

<WSL-USERNAME>-docker@docker-container:~/$ docker rmi <IMAGE ID>       # Remove image
<WSL-USERNAME>-docker@docker-container:~/$ docker rm <CONTAINER ID>    # Remove container
```