Chapter 0 - Preparations

(Optional)

Course authors (Git file)

Section 1

Description

Description

About chapter 0 - Preparations:

- This chapter is not needed when you participate in an on-site course at IHP.
- The tools and computers at IHP are ready-to-use.

This chapter can help you if …:

- you want to do the course on your own.
- you want to give this course as a trainer.
- you Want to have everything running in your own environment.

Freshness:

- The development of the tools is moving fast.
- Expect the installation tutorials to break often.

Here comes a short description of the options, followed by their detailed guides:

Option A: OpenROAD Flow Scripts (ORFS) on your computer

- A plain installation of OpenROAD, Yosys, Klayout and some flow scripts into your system.
- This option puts everything directly under your control and only installs the minimum toolset neccessary for the course.
- It requires the permissions to install software on your computer.
- The guide makes use of Ubuntu Linux.

Option B: IIC-OSIC-TOOLS in a docker container on your computer

- This docker container is like a swiss knife for EDA tools. It can be configured in many ways and contains a lot of useful tools.
- All the tools for the course are in it.
- It requires the permissions to install software on your computer.
- The guide makes use of Ubuntu Linux.

Option C: ORFS docker images in a docker container on your computer

- This option was provided by a previous participant of the course:
  - **tucanae47**: https://github.com/tucanae47
- It links you to a docker that was build and uploaded by **tucanae47**
- The requirements are the same as with Options B

Section 2

Install Option A: OpenROAD Flow Scripts (ORFS) on your computer

Install Option A: OpenROAD Flow Scripts (ORFS) on your computer

- This guide is a list of shell commands with some short explanations and weblinks.
- This was tested on a freshly installed Ubuntu LTS 24.04.1.
- The order of the commands is crucial and must not be skipped.
- For more explanations look into the documentations and README files of the tools. The weblinks are given.

Prerequisites:

- Ubuntu LTS 24.04.1 (should work on other Linux too, see weblink)
- Permission to install software (sudo rights)
- Reliable internet connection
- git installed: `sudo apt install git`

Weblink for detailed information:

https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts/blob/master/docs/user/BuildLocally.md

Your install folder

Navigate to a folder where you want the installation to reside in. The install will need some Gigabytes space.

```
1 cd <INSERT PATH TO YOUR INSTALL FOLDER HERE>
```

Clone the ORFS repo

Clone the repository to your computer:

```
1 git clone --recursive https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts
```

Run the setup script

Run the setup script to install the dependecies:

```
cd OpenROAD-flow-scripts
```

```
sudo ./setup.sh
```

Build the tools

Build all tools. This will take a while, depending on the computer:

```
1 ./build_openroad.sh --local
```

Verify the builds

Verify that the tools are available. You should get version informations of the tools with the following commands:

```
source ./env.sh
```

```
klayout -v
```

```
yosys --version
```

```
openroad -version
```

Section 3

Install Option B: IIC-OSIC-TOOLS in a docker container on your computer

Install Option B: IIC-OSIC-TOOLS in a docker container on your computer

- This guide is a list of shell commands with some short explanations and weblinks.
- This was tested on a freshly installed Ubuntu LTS 24.04.1.
- The order of the commands is crucial and must not be skipped.
- For more explanations look into the documentations and README files of the tools. The weblinks are given.

Prerequisites:

- Ubuntu LTS 24.04.1
- Permission to install software (sudo rights)
- Reliable internet connection

The IIC-OSIC-TOOLS docker container:

With the following steps a preconfigured docker gets installed. The docker is created and maintained by: Institute for Integrated Circuits (IIC) at the Johannes Kepler University Linz (JKU)

and is avaiable in their Github with more detailed installation instructions:
https://github.com/iic-jku/IIC-OSIC-TOOLS

Step 1: Install docker with apt:

Weblink for detailed informations:
https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository

Add Docker's official GPG key:

```
1 sudo apt-get update
```

```
sudo apt-get install ca-certificates curl
```

```
sudo install -m 0755 -d /etc/apt/keyrings
```

```
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
```

```
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

Add the repository to apt sources:

```
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
        https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
sudo apt-get update
```

Install the latest version of docker:

```
1 sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

Step 2: Manage docker as a non-root user

Weblink for detailed informations:
https://docs.docker.com/engine/install/linux-postinstall/#manage-docker-as-a-non-root-user

```
sudo groupadd docker
```

```
sudo usermod -aG docker $USER
```

```
newgrp docker
```

Step 3: Run the hello-world docker:

Run the hello-world example docker (without the need of sudo user):

```
1 docker run hello-world
```

No errors should be displayed in running the hello-world example. The output in the shell should contain this message:

```
...
Hello from Docker!
This message shows that your installation appears to be
working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from
    the Docker Hub. (amd64)
 3. The Docker daemon created a new container from that
    image which runs the executable that produces the
    output you are currently reading.
 4. The Docker daemon streamed that output to the Docker
    client, which sent it to your terminal.
...
```

Step 4: Clone the IIC-OSIC-TOOLS git repository to your computer:

Weblink for detailed informations about the steps 4 - 5:
https://github.com/iic-jku/IIC-OSIC-TOOLS/blob/main/README.md

Install git:

```
sudo apt install git
```

Navigate to a folder where you want the repository to be in:

```
cd <INSERT PATH TO YOUR FOLDER HERE>
```

Clone the IIC-OSIC-TOOLS:

```
1 git clone --depth=1 https://github.com/iic-jku/iic-osic-tools.git
```

Step 5: Start the docker

```
./start_x.sh
```

A shell window pops up, in which the docker runs.

Step 6: Get the OpenROAD flow scripts

- To be written
- This should be matching to option C (IHP server)
- Waiting on IHP information about their docker / server install.

Section 4

Install Option C: ORFS docker images in a docker container on your computer

Foreword about Option C:

- This option was provided by a previous participant of the course:
  - **tucanae47**: https://github.com/tucanae47
- It links you to a docker that was build and uploaded by **tucanae47**

Prerequisites and Docker install

- same as option B

Build your own docker image

This is taken from the ORFS documentation here

https://openroad-flow-scripts.readthedocs.io/en/latest/user/BuildWithDocker.html#clone-and-build

and only tested with the designs from the IHP

```
git clone --recursive https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts
cd OpenROAD-flow-scripts
./build_openroad.sh
```

Use a pre-build docker image by tucanae47

- Here is a pre-build docker image by **tucanae47**
- This image can be used without building a docker image yourself.

Workaround with a pre-build image

- The dockerhub image provided by openroad after multiple tests was not working.
- As a workaround, one fully fresh docker image was built and uploaded to
  - https://hub.docker.com/r/tucanae47/orfs.
- Pull the image and tag it so that the docker_shell utility can use it.

```
1 docker pull tucanae47/orfs
2 docker tag tucanae47/orfs openroad/flow-ubuntu22.04-builder:latest
```

Verify the builds

Start docker using the provided scripts for starting openroad executing make from the flow directory using the docker_shell utility:

```
cd flow
./util/docker_shell make
```

The docker_shell script will execute comands from docker with correct config. The rest of the commands part of the make targets of ORFS will work in the same way:

```
./util/docker_shell make clean_all
./util/docker_shell make gui_final
```

Access docker and execute cmds with docker-compose (advanced):

docker compose allows the creation of complex docker setups inside a yaml file

Step 1

install docker-compose

```
sudo apt-get update
sudo apt-get install docker-compose-plugin
```

Step 2

Paste the following yaml snippet into a file named

docker−compose.yml

and place it on the flow in the ORFS directory

/path/to/OpenROAD−flow−scripts/flow

```
version: '3.8'

services:
  openroad:
    image: ${OR_IMAGE:-openroad/flow-ubuntu22.04-builder:latest}
    container_name: openroad_container
    network_mode: "host"
    environment:
      - LIBGL_ALWAYS_SOFTWARE=1
      - QT_X11_NO_MITSHM=1
      - XDG_RUNTIME_DIR=/tmp/xdg-run
      - DISPLAY=${DISPLAY}
      - QT_XKB_CONFIG_ROOT=/usr/share/X11/xkb
      - XAUTHORITY=/tmp/.docker.xauth
      - FLOW_HOME=/OpenROAD-flow-scripts/flow/
      - YOSYS_EXE=${YOSYS_EXE:-/OpenROAD-flow-scripts/tools/install/yosys/bin/yosys}
      - OPENROAD_EXE=${OPENROAD_EXE:-/OpenROAD-flow-scripts/tools/install/OpenROAD/bin/openroad}
      - KLAYOUT_CMD=${KLAYOUT_CMD:-/usr/bin/klayout}
    volumes:
      - /tmp/.X11-unix:/tmp/.X11-unix
      - /tmp/.docker.xauth:/tmp/.docker.xauth
      - .:/OpenROAD-flow-scripts/flow:Z
    stdin_open: true
    tty: true
```

This will allow a linux computer to execute gui commands too.

Step 3

Get inside the docker and continue the normal workflow for the course

```
# allow docker client to connect to your xserver
xhost +
docker-compose run openroad
# inside the docker run all cmds learned
root@userX:/OpenROAD-flow-scripts# source env.sh
root@userX:/OpenROAD-flow-scripts# cd flow
root@userX:/OpenROAD-flow-scripts# make clean_all
root@userX:/OpenROAD-flow-scripts# make
root@userX:/OpenROAD-flow-scripts# make gui_final
root@userX:/OpenROAD-flow-scripts# klayout
root@userX:/OpenROAD-flow-scripts# openroad -gui
```