The Tools

Carsten Wulff, carsten@wulff.no

Keywords: WSL, Git, AICEX, ngspice, xschem, Magic, cicconf, cicsim, cicpy

Tools

I would strongly recommend that you install all tools locally on your system. There is a video that describes the install procedure. It's a few years old, but should still be able to guide you https://youtu.be/DRppsdjo2Rc?si=x8cJsa1lpncvSFmu.

For the analog toolchain we need some tools, and a process design kit (PDK).

- Skywater 130nm PDK. I use open_pdks to install the PDK
- Magic VLSI for layout (Version 8.3 revision 541)
- ngspice for simulation (version 45.2)
- netgen for LVS (1.5.295)
- xschem (3.4.8RC)
- verilator (5.034)
- python > 3.10

The tools are not that big, but the PDK is huge, so you need to have about 50 GB disk space available.

Setup WSL (Applicable for Windows users)

Install a Linux distribution such as Ubuntu 24.04 LTS by running the following command in PowerShell on Windows and follow the instructions.

```
wsl --install -d Ubuntu-24.04
```

When you have installed the Linux distribution and signed into it, install make

```
sudo apt install make
```

Setup public key towards github

Do

```
ssh-keygen -t rsa
```

And press "enter" on most things, or if you're paranoid, add a passphrase

Then

```
cat ~/.ssh/id_rsa.pub
```

And add the public key to your github account. Settings - SSH and GPG keys

Provide git with author identity

There are interactions with git that require an author identity. You are supposed to use one of these interactions a lot during the project, namely, `git commit`. What you need to provide is an email address and a name. If you would like to keep your real email address private/secret, read what it says on GitHub at your user settings page under emails. Use the below commands to provide the author identity information to git.

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

Get AICEX and setup your shell

You don't have to put aicex in `$HOME/pro`, but if you don't know where to put it, choose that directory.

```
cd
mkdir pro
cd pro
git clone --recursive https://github.com/wulffern/aicex.git
```

You need to add the following to your `~/.bashrc` (note that `~` refers to your home directory `$HOME/.bashrc` also works, or `$HOME/.bash_profile` on some newer macs)

```
export PDK_ROOT=/opt/pdk/share/pdk
export LD_LIBRARY_PATH=/opt/eda/lib
export PATH=/opt/eda/bin:$HOME/.local/bin:$PATH
```

On systems with python3 > 3.12

On newer systems it's not trivial to install python packages because python is externally managed. As such, we need to install a python environment.

```
#- Find a package similar to name below
sudo apt-get update
sudo apt install python3.12-venv
sudo mkdir /opt
sudo mkdir /opt/eda
sudo mkdir /opt/eda/python3
sudo chown -R $USER:$USER /opt/eda/python3/
python3 -m venv /opt/eda/python3
```

Modify the `~/.bashrc` to include the python environment

```
export PATH=/opt/eda/bin:/opt/eda/python3/bin:$HOME/.local/bin:$PATH
```

Install Tools

Make sure you load the settings before you proceed

```
source ~/.bashrc
```

Hopefully the commands below work, if not, then try again, or try to understand what fails. There is no point in continuing if one command fails.

```
cd aicex/tests/
make requirements
make tt
```

On a mac, you probably need to add bison to the path

```
export PATH="/opt/homebrew/opt/bison/bin:$PATH"
```

I've split the install of each of the tools. It's possible to run the commented out lines instead, but they often fail

```
#make eda_compile
#sudo make eda_install
make magic_compile  magic_install
make netgen_compile netgen_install
make xschem_compile xschem_install
make iverilog_compile iverilog_install
make ngspice_compile # Sometimes fails
make ngspice_compile ngspice_install
```

For later versions of ngspice, they have modified the build system, so you may have to run

```
make ngspice2_compile
make ngspice2_install
```

On Mac, do

```
brew install yosys verilator
```

On Linux, do

```
make yosys_compile yosys_install
```

On all, do

```
python3 -m ensurepip --default-pip

python3 -m pip install matplotlib numpy click svgwrite \
    pyyaml pandas tabulate wheel setuptools tikzplotlib
source install_open_pdk.sh
```

Install cicconf

cIcConf is used for configuration. How the IPs are connected, and what version of IPs to get.

```
cd
cd pro/aicex/ip/cicconf
git checkout main
git pull
python3 -m pip install -e .
cd ../
```

Update IPs

```
cicconf clone --https
cd ../..
```

Install cicsim

cIcSim is used for simulation orchestration.

```
cd aicex/ip/cicsim
git checkout main
git pull
python3 -m pip install -e .
cd ../..
```

Install cicpy

CicPy is used to generate layout

```
cd aicex/ip/cicpy
git checkout master
git pull
python3 -m pip install -e .
cd ../..
cd aicex/ip/cicspi
git checkout main
git pull
python3 -m pip install -e .
cd ../..
```

Setup your ngspice settings

Edit `~/.spiceinit` and add

```
set ngbehavior=hsa      ; set compatibility for PDK libs
set ng_nomodcheck       ; don't check the model parameters
set num_threads=8       ; CPU hardware threads available
set skywaterpdk
option noinit           ; don't print operating point data
option klu
optran 0 0 0 100p 2n 0 ; don't use dc operating point,
option opts
```

CHECK THAT MAGIC AND XSCHEM WORK

To check that magic and xschem work

```
cd ~/pro/aicex/ip/sun_sar9b_sky130nm/work
magic ../design/SUN_SAR9B_SKY130NM/SUNSAR_SAR9B_CV.mag &
xschem -b ../design/SUN_SAR9B_SKY130NM/SUNSAR_SAR9B_CV.sch &
```

SUMMARY

The one-page version of this chapter:

- The whole flow is open source: xschem, ngspice, magic, netgen, and the cic tools on top
- Everything installs from scripts, and the docs and video walk the setup end to end
- If the tools do not run, nothing else in this course happens - do this first, and ask early when stuck

WOULD YOU LIKE TO KNOW MORE?

Each tool documents itself well: xschem, ngspice, magic, netgen and the SKY130 PDK

The tools chapter later in this book covers the cic layer built on top of them

Carsten Wulff received the M.Sc. and Ph.D. degrees in electrical engineering from the Department of Electronics and Telecommunication, Norwegian University of Science and Technology (NTNU), in 2002 and 2008, respectively. During his Ph.D. work at NTNU, he worked on open-loop sigma-delta modulators and analog-to-digital converters in nanoscale CMOS technologies. In 2006-2007, he was a Visiting Researcher with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada. Since 2008 he's been with Nordic Semiconductor in various roles, from analog designer, to Wireless Group Manager, to currently Principle IC Scientist. From 2014-2017 he did a part time Post.Doc focusing on compiled, ultra low power, SAR ADCs in nanoscale technologies. He's also an Adjunct Associate Professor at NTNU. His present research interests includes analog and mixed-signal CMOS design, design of high-efficiency analog-to-digital converters and low-power wireless transceivers. He is the developer of Custom IC Compiler, a general purpose integrated circuit compiler, and makes the occational video on analog integrated circuits at https://www.youtube.com/@analogicus. For full CV see https://analogicus.com/markdown-cv/.