

A decorative graphic on the left side of the slide, consisting of various geometric shapes (circles, squares, pentagons, hexagons) connected by thin lines, forming a network-like structure. The colors are shades of orange and red.

SIESTA compilation options

Yann Pouillon, Simune Atomistics

To build or not to build?

- Choice 1: build SIESTA yourself
- Choice 2: have someone build SIESTA for you

This video: big-picture overview of choice 1 on Linux

Detailed instructions for all platforms:

<https://docs.siesta-project.org/>

Yann Pouillon: [ORCID: 0000-0001-9850-2129](https://orcid.org/0000-0001-9850-2129)

- M. Eng. + PhD in condensed matter physics
- Contributions to various codes & frameworks, emphasis on build systems & packaging
- Co-founder of the Electronic Structure Library (ESL): <https://esl.cecam.org/>
- Currently: Innovation Architect at Simune Atomistics

Simune: <https://www.simuneatomistics.com/>

- Software & services around atomistic simulations
- Industrial & academic users
- Direct contributions to open-source software packages, e.g. [SIESTA](#), [ASE](#), [ABINIT](#), [ESL Bundle](#)
- Providing pre-compiled SIESTA binaries, in particular for Windows

Common traits of software builds



Building software \neq transforming source code into binary code

5 steps to consider:

1. Preparing the environment
2. Configuring the build
3. Producing the executables
4. Installing the executables
5. Running the executables in the correct environment

Building SIESTA 4.x manually

1. Preparing the environment

- Not specific to the software
- Valid for (almost) all DFT codes
- Critical for successful builds
- List of requirements
- Revise/update once or twice a year

- For SIESTA 4.x
 - C compiler
 - Fortran compiler
 - GNU Make
 - MPI distribution
 - Linear algebra libraries
 - HDF5 + NetCDF

- Package managers / App stores
 - Linux (Debian-based or RedHat-based)
 - MacOS (Intel-based or ARM-based)
 - Windows

- Build frameworks
 - Console (terminal): manual builds
 - Scripts & recipes: streamlined builds
 - Docker: generic containers
 - Singularity: HPC containers
 - EasyBuild: reproducible builds, HPC
 - Spack: highly-tuned build toolchains
 - Virtual machines: testing & learning

Details: <https://docs.siesta-project.org/>

2. Configuring the build of SIESTA 4.x

1. Which kind of executable to build?
 - a. Serial or parallel?
 - b. Which of the available compilers?
 - c. Which of the MPI distributions?
 - d. Which set of linear algebra libraries?
 - e. With or without platform-independent I/O?
2. Translate choices into an *arch.make* file in the *Obj/* subdirectory (see <https://docs.siesta-project.org/>)
3. Type:

```
sh ../Src/obj_setup.sh
```

Pro Tip

Type “make clean” before building SIESTA to check if the configuration is valid.

3.1. Building SIESTA 4.x

- For SIESTA 4.0 or to limit use of resources: `make`
- Multiple cores allow parallel builds (faster): `make -j N`, e.g. `make -j 4`
- Wait until the build finishes
- Type: `ls -ltr`
The last file should be the SIESTA executable

Pro Tip

If the build fails after typing “`make -j N`”, type “`make`” (without “`-j N`”) to find out more easily where the error occurred.

3.2. Building the SIESTA utilities

- After building SIESTA: got to the *Util/* subdirectory to build utilities
- Example: building *gnubands* to plot band structures with [Gnuplot](#)

```
cd Util/Bands  
make  
ls -ltr
```

- The last file displayed by the *ls* command should be named *gnubands* and be executable
- Rinse & repeat for any other utility you want to build

4. Installing SIESTA

- Good Practice: copy executables out of the source package
- Example with SIESTA 4.1 and *gnubands* (assuming the current directory is *Obj*)

```
mkdir -p $HOME/siesta/4.1/bin
cp siesta $HOME/siesta/4.1/bin
cp Util/Bands/gnubands $HOME/siesta/4.1/bin
```

- Create a configuration script: `$EDITOR $HOME/siesta/4.1/bin/siesta-vars.sh`

```
#!/bin/sh
PATH="$HOME/siesta/4.1/bin:$PATH"
export PATH
```

5. Running SIESTA with the correct environment



- Use siesta-vars.sh to select the desired SIESTA version

```
source $HOME/siesta/4.1/bin/siesta-vars.sh
```

- To be done only once per terminal session
- Advantage: keep the default environment clean

Detailed build instructions: <https://docs.siesta-project.org/>

Building SIESTA 5.x with the ESL Bundle

Getting started with the ESL Bundle




- ESL Bundle = curated collection of libraries used to build DFT codes
- Only provides components related to electronic structure
- Explicitly excluded: linear algebra & math libraries, HDF5, NetCDF
- Advantage: use the same up-to-date libraries to build several DFT codes
- 1 or 2 releases per year
- Download: <https://gitlab.com/ElectronicStructureLibrary/esl-bundle>

Installing SIESTA 5.x dependencies

- Example: compiling on Ubuntu with GCC and OpenMPI

```
git clone \  
    https://gitlab.com/ElectronicStructureLibrary/esl-bundle.git  
cd esl-bundle  
mkdir my_build_dir  
cd my_build_dir  
../install-bundle -s ubuntu -c gcc -f openmpi
```

- Wait until the build finishes (may take a while)
- Components will be installed in a subdirectory named *install/*
-  Recent LibXC versions may be very slow to compile

- Before compiling SIESTA:

```
LD_LIBRARY_PATH="/path/to/esl-bundle/my_build_dir/install/lib:$LD_LIBRARY_PATH"  
export LD_LIBRARY_PATH  
PATH="/path/to/esl-bundle/my_build_dir/install/bin:$PATH"  
export PATH
```

- Then: follow the same steps as for the manual SIESTA 4.x build

Building any SIESTA version with EasyBuild

Getting started with EasyBuild



- Choose a location where to install software packages, e.g. `/opt/hpc/` (need 10-20 Gb free)
- Prerequisites: GCC, Lmod, Python 3.x (see <https://docs.siesta-project.org/>)
- Installing EasyBuild:

```
pip install easybuild
```

- Configuring EasyBuild:

```
mkdir -p $HOME/.config/easybuild  
$EDITOR $HOME/.config/easybuild/config.cfg
```

EasyBuild configuration (config.cfg)



```
[basic]
repositorypath = /opt/hpc/EB_Archives
robot-paths = %(repositorypath)s:%(DEFAULT_ROBOT_PATHS)s


[config]
modules-tool = Lmod
prefix = /opt/hpc

[override]
download-timeout = 1200
```

Pro Tip

Before starting, make sure you can write files to the `/opt/hpc/` directory.

Selecting & installing a toolchain

- EasyBuild provides several toolchains (consistent compilers & libraries)
- Package availability: takes time \Rightarrow avoid selecting latest toolchains
- Example: `gompi/2020a` = GCC + OpenMPI, versions available at the beginning of 2020
- Making the toolchain ready: `eb gompi-2020a.eb -r`
-  This might take between 3 and 7 hours!
- Good news: you do it only once per toolchain

Installing common SIESTA dependencies

- Searching for an EasyBuild recipe:

```
eb -S KEYWORD
```

- Linear algebra libraries: search for scalapack

```
eb ScaLAPACK-2.1.0-gompi-2020a.eb -r
```

- NetCDF: search for netcdf-fortran

```
eb netCDF-Fortran-4.5.2-gompi-2020a.eb -r
```

- LibXC: search for libxc

```
eb libxc-4.3.4-GCC-9.3.0.eb -r
```

Installing SIESTA 5.x dependencies



- ESL EasyConfigs repository: <https://gitlab.com/ElectronicStructureLibrary/esl-easyconfigs>
- Build libraries one after the other

```
git clone
https://gitlab.com/ElectronicStructureLibrary/esl-easyconfigs.git
cd esl-easyconfigs/easyconfigs
ls -R
cd x/xmlf90
eb xmlf90-1.5.4-gompi-2020a.eb
cd ../../1/libpsml
eb libpsml-1.1.7-gompi-2020a.eb
cd ../libgridxc
eb libgridxc-0.8.5.1-gompi-2020a.eb
```

- EasyBuild builds “modules”
- Listing the available modules: `module avail`
- Loading a module and all its dependencies:

```
module load MODULE
```

or

```
module load MODULE/VERSION
```

- Then: follow the same steps as for SIESTA 4.x builds

- 5 steps to any build option:
 1. Prepare the environment
 2. Configure the build
 3. Build the executables
 4. Install the executables
 5. Run in the correct environment
- Detailed instructions: <https://docs.siesta-project.org/> (including an additional test step)
- Exercise: make scripts that perform the workflow of your choice