Yosys Installation Guide

October 4, 2025

1 Installing OpenROAD

1. Install Git, Yosys & Klayout (if not already installed)

Open a terminal and run:

```
sudo apt update
sudo apt install git -y

sudo apt install klayout
```

2. Clone the OpenROAD Repository

Download the OpenROAD-flow-scripts project:

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

Important: The `--recursive` flag ensures all necessary submodules are downloaded.

3. Run the Setup Script

This installs all required software and dependencies. It can take 10–30 minutes.

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

Note: You may be prompted for your password.

4. Build OpenROAD

Now build the OpenROAD tools:

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

This generates a file called `build_openroad.log`. If something goes wrong, check this file.

5. Check That Everything Works

First, set up the environment:

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

Then check the tools:

```
yosys -help
yosys -m slang -p "slang_version"
klayout
openroad -help
```

You should see version info or help output if everything is working.

6. Run a Test Flow (GCD Design)

Navigate to OpenROAD Flow Directory

Run a complete flow on a test design:

```
cd flow
make
```

This uses the `gcd` example with the `nangate45` PDK.

7. View Final Layout in GUI

After the flow completes, view the final layout in the GUI:

```
make gui_final
```

This will launch the OpenROAD GUI showing the layout.

Note: Every Time You Open a New Terminal

Before using OpenROAD tools, always run:

```
cd OpenROAD-flow-scripts
source ./env.sh
```

2 Adding New Design to OpenROAD

Step 1: Create and Define Your Verilog Source File

1. Navigate to the source code directory and create the folder for your design.

```
cd designs/src
mkdir <folder-name>
```

2. Create the Verilog file (`<file_name>.v`) using a text editor like `gedit` or `vi`.

```
gedit spm.v
```

3. **Copy and paste** your Verilog code into `<file_name>.v`. Make sure the top module name in your code is same as file name.

—

Step 2: Create the Design Configuration Directory and File

Next, create a specific folder for the **gf180** platform configuration files and the main configuration file, `config.mk`.

You can choose any other platform available in list

1. Navigate to the platform-specific directory and create the folder for your design (**spm**).

```
cd designs/gf180
mkdir <folder-name>
cd <folder-name>
```

Make sure the folder name remains same as folder in src folder

2. Create the main configuration file (`config.mk`).

```
gedit config.mk
```

—

Step 3: Define Key Design Parameters in `config.mk`

open `<folder-name>/config.mk` and **copy this entire block** into the file. These lines define critical settings for the automated design flow.

```
export PLATFORM          = <platform-name>

export DESIGN_NAME       = <top-module-name>

export DESIGN_NICKNAME   = <file-name>

# NOTE: In this flow, $(DESIGN_NICKNAME) is usually the same as $(DESIGN_NAME), 'spm'.
# This helps the flow find the correct files.

export VERILOG_FILES     = $(sort $(wildcard ./designs/src/<folder-name>/*.v))
export SDC_FILE          = ./designs/$(PLATFORM)/<folder-name>/constraint.sdc

# Physical Design Parameters (Layout)
export CORE_UTILIZATION = 40  # Target \% of the chip core area used by logic.
export PLACE_DENSITY    = 0.60 # Target density of logic cells during placement.

# SWITCH TO ABSOLUTE SIZING MODE
export FP_SIZING = absolute

# DEFINE DIE AND CORE AREAS (in microns)
export DIE_AREA  = "0 0 1000 1000"
export CORE_AREA = "50 50 950 950"

# Note: CORE_UTILIZATION and PLACE_DENSITY are ignored when FP_SIZING is 'absolute'

# However, many flows use PLACE_DENSITY (or PL_TARGET_DENSITY) even in absolute mode
# to control placement spread *within* the CORE_AREA, so keeping it is safe.

export PLACE_DENSITY = 0.60

# Timing Constraints
export TNS_END_PERCENT  = 100 # Target timing slack (keep at 100 for now).
```

Note: For this guide, `$(DESIGN_NICKNAME)` is assumed to be `spm`. We explicitly use `spm` in the paths above to avoid confusion.

—

Step 4: Define SDC (Timing) Constraints

The Synopsys Design Constraints (SDC) file tells the tools about your clock speed and how much time signals have to enter and exit your chip.

1. Navigate to your design's configuration directory and create the constraints file.

   ```
   cd designs/gf180/<folder-name>
   gedit constraint.sdc
   ```

2. **Copy this entire block** into `constraint.sdc`. This sets up a clock named `core_clock` with a period of **10 units** (10ns or 100MHz).

   ```
   current_design <module-name>

   set clk_name  core_clock
   set clk_port_name clk
   set clk_period 10
   set clk_io_pct 0.2
   ```

```
set clk_port [get_ports $clk_port_name]

create_clock -name $clk_name -period $clk_period
    $clk_port

set non_clock_inputs [lsearch -inline -all -not -exact [
    all_inputs] $clk_port]

set_input_delay  [expr $clk_period * $clk_io_pct] -clock
    $clk_name $non_clock_inputs
set_output_delay [expr $clk_period * $clk_io_pct] -clock
    $clk_name [all_outputs]
```

3. **Important:** You only need to check and potentially change three values in the code above:
   - `current_design <module-name>` (Must match your top module name).
   - `set clk_port_name clk` (Must match the name of the clock input port in your `<file-name>.v` file).
   - `set clk_period 10` (The required time, in nanoseconds (ns), for one clock cycle. A smaller number means a faster clock).

—

Step 5: Run the Design Flow

1. Go back to the main flow directory (or any directory where the main `Makefile` is located).
2. Run the following command, which tells the `make` tool to use your specific configuration file:

   ```
   make DESIGN_CONFIG=./designs/gf180/spm/config.mk
   ```

3. **Alternative (Permanent) Method:** You can edit the main `Makefile` to always run your design by default.
   - Open the main `Makefile`.

     ```
     gedit Makefile
     ```

   - Find any existing line that looks like `DESIGN_CONFIG=...` and **add a hash tag (#) at the beginning** to comment it out.
   - Add the following two lines to the `Makefile` and save it:

```
# New design configuration: spm
DESIGN_CONFIG=./designs/gf180/spm/config.mk
```

- Now, you can simply run:

```
make
```

Step 6: GDSII Visualization

To open the KLayout GDSII File

```
make DESIGN_CONFIG=./designs/gf180/spm/config.mk gui_final
```

```
make gui_final
```

3 Printing Flow Report

- Print the chip area

```
report_design_area
```

- Print Timing Report

```
report_worst_slack
report_tns
report_wns
```

- Power Report

```
report_power
```

Troubleshooting Tips

- **Setup or build failed?**
  → Check the `build_openroad.log` file.
- **Command not found?**
  → You probably forgot to run `source ./env.sh`.
- **Stuck at a wrong build**
  → Clean the previous build files

```
make DESIGN_CONFIG=./designs/gf180/spm/config.mk clean_all
```

- **Still stuck?**
  → Open an issue at:
  `https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts/issues`
  and upload `build_openroad.log`.

Key Design Configuration Variables

Explanation:

- `PLATFORM`: Specifies the **Process Design Kit** (PDK) or technology node (e.g., `gf180`).
- `DESIGN_NAME`: The name of the **top-level module** of the design.
- `VERILOG_FILES`: The path to the design `.v` **Verilog source files** or JSON files providing a description of modules (check `yosys -h write_json` for more details).
- `SDC_FILE`: The path to the design's **Synopsys Design Constraints** (`.sdc`) file.
- `CORE_UTILIZATION`: The **core utilization percentage**. This is the target percentage of the core area that will be occupied by standard cells.
- `PLACE_DENSITY`: The desired **placement density of cells**. It reflects how spread the cells would be on the core area. A value of **1** means closely dense, while **0** means widely spread.