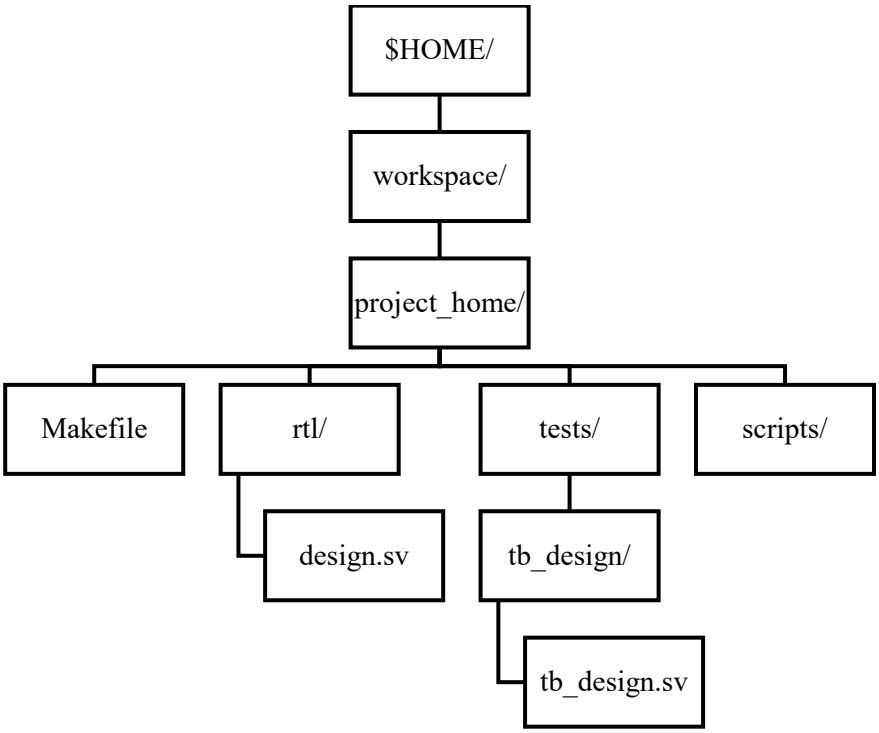
ASIC-Tools

Design and Simulation Quickstart Guide

Author: Ryan Cramer

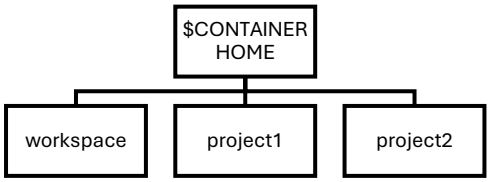
Basic Project Directory Structure:



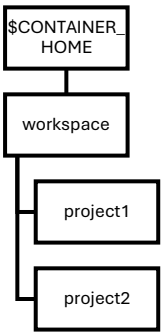
Once you've installed the tools, make sure Docker is running before running "run.sh".

** NOTE: Anything created outside the "workspace" directory will be deleted and unrecoverable after exiting the container. Only folders and files created within the "workspace" directory will available after exiting the container.

UNSAFE:



SAFE:



Basic Flow Steps:

Creating a Project:

1. **Navigate to the asic-tools directory and run the startup script**

```
user@host:~$ cd path/to/asic-tools && ./run.sh
```

**NOTE: you'll know you're in the container when the name changes from your `user@host` to `ubuntu@asic` (or `root@asic` depending on sudo usage)

2. **Create a project directory or clone your repository**

```
ubuntu@asic:~/workspace$ mkdir <project_name>\
```

or

```
ubuntu@asic:~workspace$ git clone git@github.com:you/your_proj.git
```

3. **Copy or add the Makefile to your project home (Latest Makefile)**

4. **Create an rtl folder and a tests folder**

```
~/workspace/project_home$ mkdir rtl tests
```

Designing and Testing RTL:

1. **Write an HDL module**

**NOTE: Use Verilog or SystemVerilog, just be sure to place design files in your projects rtl/ directory. The Makefile looks for design HDL files here.

```
~/workspace/project_home$ touch rtl/design.sv
```

2. **Once your module is written, use the make or CLI to lint**
   - CLI: verilator --lint-only --timing <your_verilog_file.sv>
   - Make: make lint

```
~/workspace/project_home$ make lint
```

You will see:

```
----- Linting All Modules -----
Linting rtl/design.sv . . . PASSED
```

-or-

```
----- Linting All Modules -----
Linting rtl/top/EEL_no_haz.sv . . . FAILED
%Error: <sometimes helpful error message>
```

- If output is green and says PASSED: design.sv, you've passed lint
- If output is red and says FAILED: design.sv, you've failed lint

3. **Write a testbench**
   - Make a sub-directory in tests/ with your design file name and the prefix "tb_"

```
~/workspace/project_home$ mkdir tests/tb_design
```

   - Create and write your testbench file with the same name formatting (SystemVerilog recommended)

```
~/workspace/project_home$ touch tests/tb_design/tb_design.sv
```

**NOTE: Please be sure to include the following or else you won't be able to view your results:

```
initial begin
        $dumpfile("tb_design.vcd");
        $dumpvars(0, tb_design);
end
```

**NOTE: You will get unstable results if you assign signals using = in your testbench. To get safe results, only use = assignments in initial blocksand <= for any other signal assignment.

Use SystemVerilog and Verilog Language Features:

Assertions: similar to C, use assertions when you expect specific values

assert (condition) else $error("error messages: %d equals %d", A, B)

Task: a function that can drive inputs and have delays

- Define within the scope of the module, so it can drive signals

Function: to compute values or generic programming

For Loop: compare functions and task outputs using assert (test many inputs)

Simulate with Verilator:

Verilator is a two state simulator. It will only show signals as being a 0 or 1.

CLI: `$ verilator --timing --trace --binary tests/<testname>/<testname>.sv -I"rtl"`

`$ ./obj_dir/V<testname>`

- if you type `./obj_dir/` and hit tab, it should autocomplete)
- Running the binary should create a waveform dump file (.vcd)

**NOTE:

--timing lets us simulate # delays

--binary makes a runnable test file

--trace tells Verilator to save simulation results to a waveform

-I"rtl" tells Verilator to include the rtl/ directory when looking for RTL files

Make: `$ make tests`

- automatically runs all tests inside the tests folder
  - every tb_<design_file_name> folder is its own test
- automatically generates your tests .vcd waveform dump file
- test can be specified with `make tests/<testname>`

To view the waveform:

- install surfer or gtkwave and use:
  - `surfer <name>.vcd`
  - `gtkwave <name>.vcd`

Simulate with Icarus Verilog:

Manual: `$ iverilog -g2012 rtl/<rtl_name>.sv tests/<test_dir>/<test_name>.sv`

`$ ./a.out`

**NOTE: -g2012 enables SystemVerilog and not just Verilog , per the IEEE 2012 spec

Make: `make itests`

- Automatically runs all tests inside tests/
- to run a specific test use: `ICARUS=1 make tests/<test_name>`
- Leaves a .vcd and a results.txt log file in the directory of the test

To view the waveform:

- install surfer or gtkwave and use:
  - `surfer <name>.vcd`
  - `gtkwave <name>.vcd`

Sources

Cal-Poly-Open-Source-ASIC-Class. *Open-Source ASIC Class*, GitHub. https://github.com/Cal-Poly-Open-Source-ASIC-Class/