Working Open

Design and Verification of ASIC / FPGA with free tools.

How close to Silicon can You get by paying only for pizza and coffee?

Ilia Greenblat

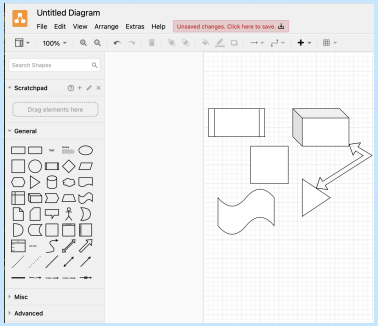
TOPICS for today.

* Documentation of the design intent and verification aspects
* Writing the design and verification components
* Keeping versions and sharing with collaborators
* Creating and running tests and regressions.
* Viewing test results
* Synthesis and Static timing
* Place and route
* CDC and Lint for poor people (frugal!).

and …Python as the glue logic of all flows

“Before the design” tools

Diagrams: draw.io

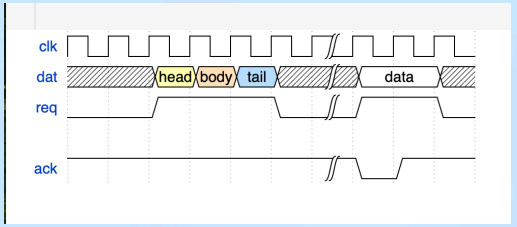


Text files:

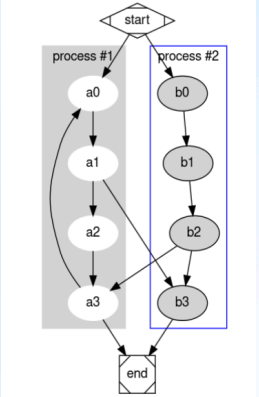
Any You fancy

(as long as it is VIM)

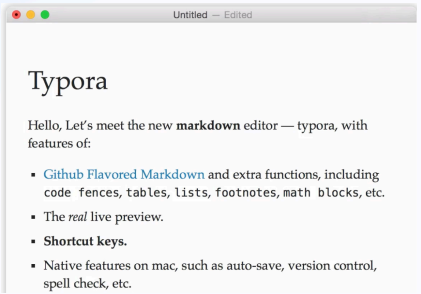
timing diags: WaveDrom



Graphs: Dot /Graphviz



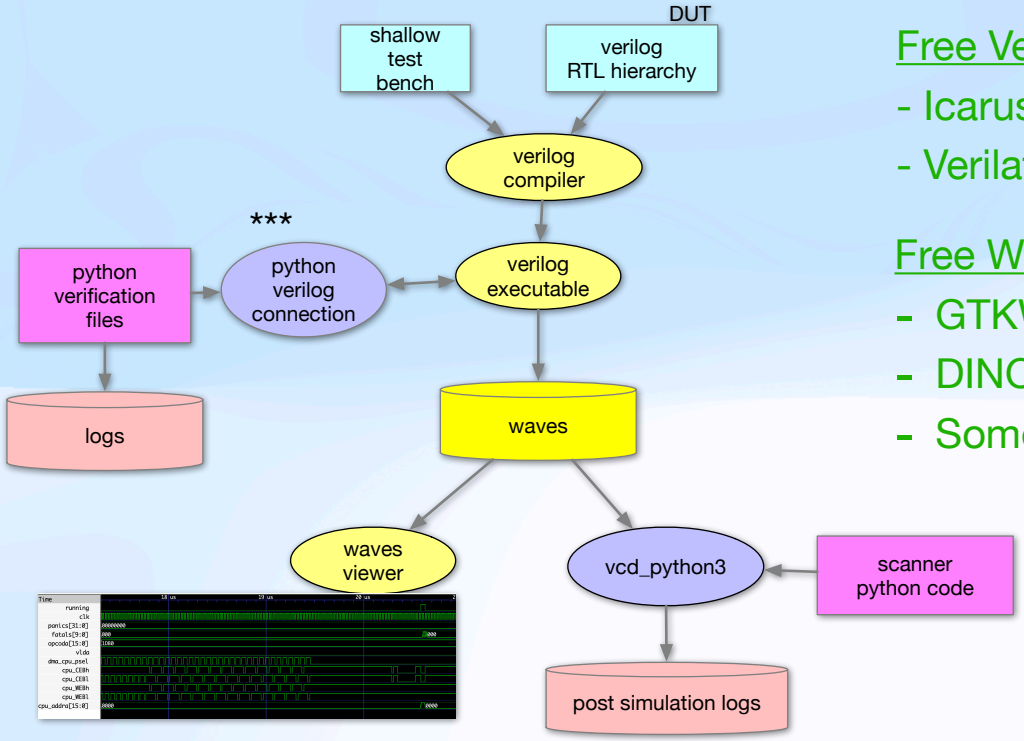
Documents: Markdown files



My favorite is

Typora, but any markdown presenter /editor is fine. The file is pure text file, editable by VIM.

Flow : the fun begins



Basics:

Free Verilog Compilers:

- Icarus (Iverilog)
- Verilator

Free Waves Viewers

- GTKWAVE
- DINOTRACE
- Some new ones

*** Same python-verilog-connection works for all commercials!

Why free tools tramp commercials?

- I am writing this presentation on my MACBOOK-PRO M1 Pro laptop.
- it has 32 Gigabytes ram.
- 8 high speed cores (and 2 efficiency cores)

With free tools i can run 8 simulations concurrently without the laptop sweating a brow. Regressions are a breeze. And synthesis and P&R on a side.

(my other laptop has 16 cores and 64 Gigabytes. But it is other company's property, so it is limited in use.

No need for VPN. No need for internet most of the time. When licenses expire i am the only one who keeps pounding on the keys. Electricity failure - no problem. No coffee for Intelites - so what? I have my own. Working on the train full speed ahead (trying not to eavesdrop trivial conversations, and get annoyed when i miss the ending - did She dumped him or not?

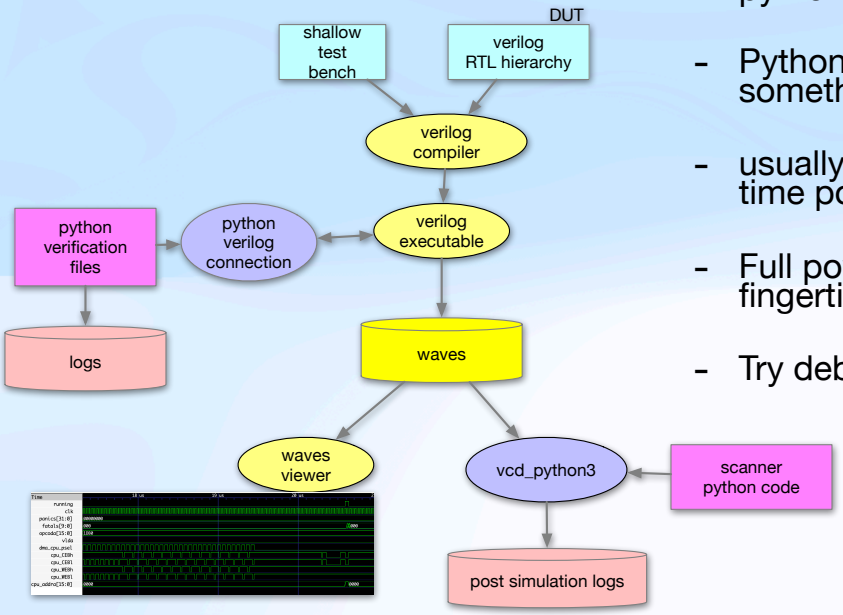
Addons:

PythonDrivenVerification

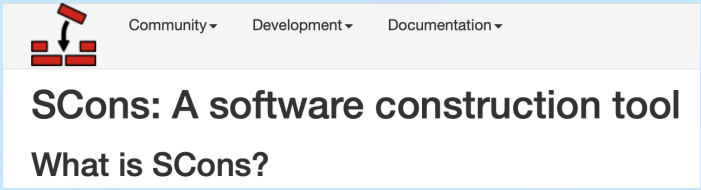
- replaces SV+UVM and Specimen
- easy to write checkers and drivers.
- Python code is globally visible
- Python has it all. (Libraries, VIPs, Random, even constraints if You really need them).
- Popular is CoCoTb, personally i like mine better.
- more info in vlsistuff repository

Waves scanner: vcd python3

- the core of this app, is compiled “C” program
- simple API to peek into waves from python.
- Python code is called every time something interesting occurs
- usually near edges of clock or any new time point.
- Full power of python at Your wave fingertips.
- Try debugging AXI protocol without it.



Database management and Builds



Makefile for old school.

Bazel for advanced users.

not ideal for hardware, but works.

Hosting data on GitHub or similar prevents data loss and eases collaboration.

Hierarchy build tools

More free tools to consider:

- Yosys: ASIC synthesis. For me, main use to iron out pre-synthesis problems. Like good linter. Gives idea on timing and area.

- openSTA: Static timing for gatelvel net lists. Not sure it is tape-out worthy, but catches crude violations.

- openRoad: Place and Route suite. actually works. Once again - not for super fast, super big, super anything. But gives pretty good idea on what is ahead. So when You hit real tools - less surprises.

- My own tools. look in vlsistuff gitHub repository - too many to simply list.

Final story:

my RTL is getting verified by team in Serbia. They do a good job and share waves of simulations that caught bugs.

Looking and the waves puts me to sleep. So first i scan the waves with vcd_python3. Once i know what to fix, i fix the RTL and need to rerun.

One way to do it: i push fixed rtl, cross fingers and wait for rerun. It takes time. And multiple reruns.

Colleague of mine, asks them to send snapshot of the simulation and can rerun it to get live set of waves.

But what happens when there is a candidate for fix?

He waits. I don’t.

I translate the waves into tester. This tester drives all inputs (of just my module!) and compares the outputs.

Without any RTL modification i expect the “test” to pass. Now i modify the RTL - if the test “fails” i probably closer to solution. I can rerun doses fixes in an hour. Remember that i dont run the whole chip, but just my problematic area.

I can also, prior to real fix, add interesting expressions to the rtl - to facilitate easier debug.

True - this flow doesn’t make me coffee, but keeps my barber relevant.