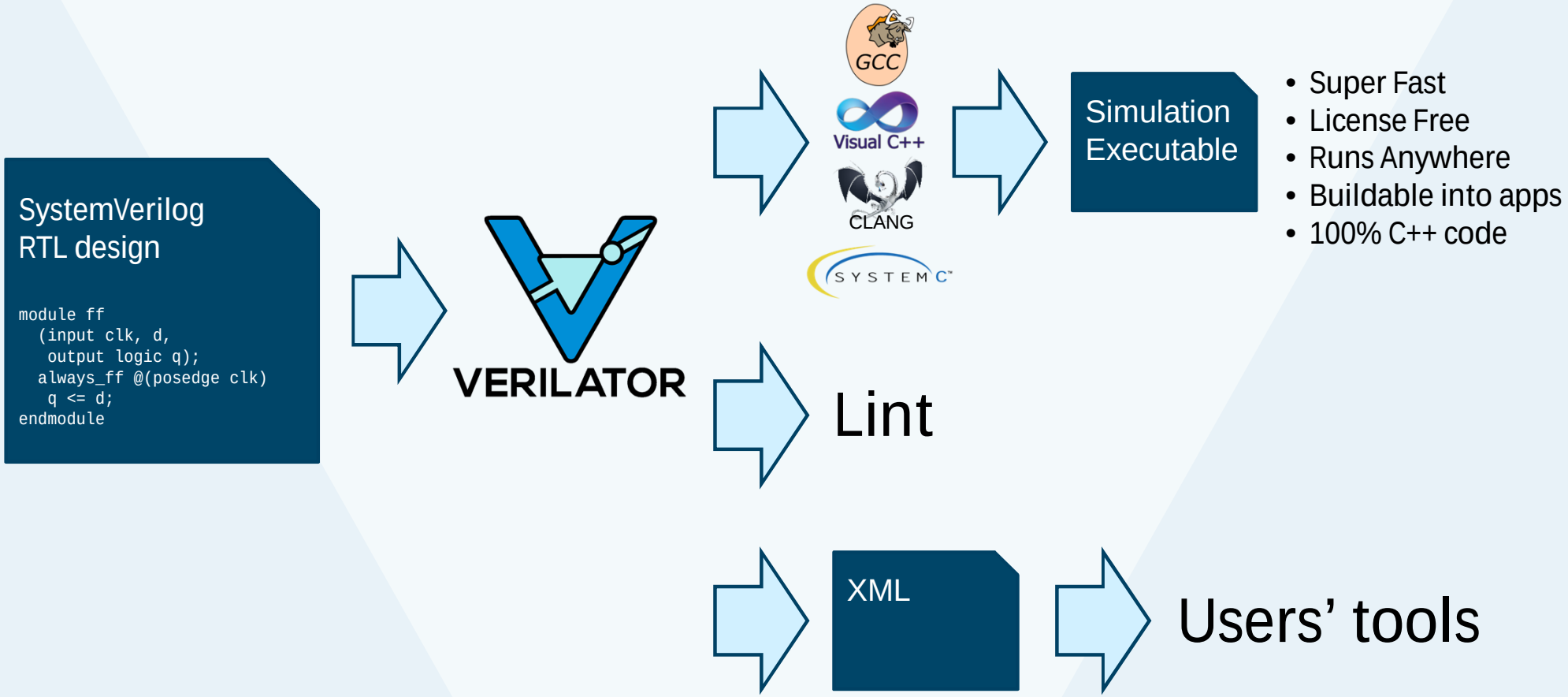
VERILATOR

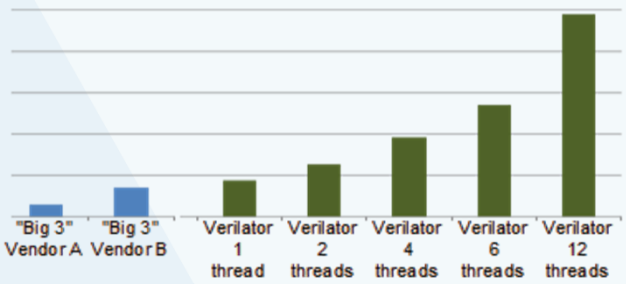
Ten Creative Uses for Verilator

Wilson Snyder, 2019

Verilator


SystemVerilog
RTL design
module ff
(input clk, d,
output logic q);
always_ff @(posedge clk)
q <= d;
endmodule
VERILATOR
GCC
Visual C++
CLANG
SYSTEMC
Simulation
Executable
Super Fast
License Free
Runs Anywhere
Buildable into apps
100% C++ code
Lint
XML
Users' tools

Why Verilator?



Fast

- Outperforms many commercial simulators
- Single- and multi-threaded output models

Widely Used

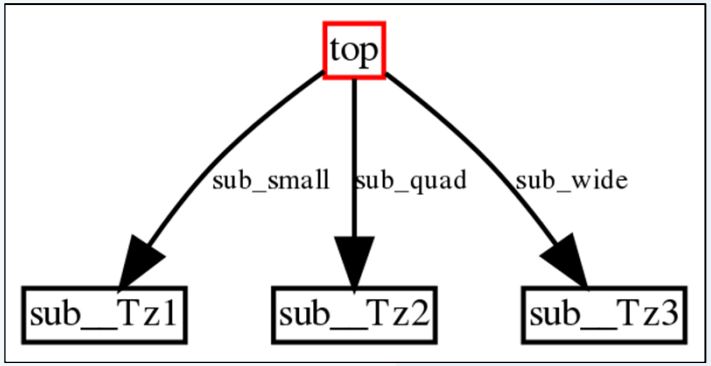
- Wide industry and academic deployment
- Out-of-the-box support from Arm, and RISC-V vendor IP

Community Driven & Openly Licensed

- Guided by the CHIPS Alliance and Linux Foundation
- Open, and free as in both speech and beer
- More simulation for your verification budget

#10 – Visualize the Design Hierarchy

- **Q:** How can I learn how modules & instances are associated?



- **A:** Use Verilator

  See NEW `examples/xml_py/vl_hier_graph`

  Run `vl_hier_graph` *`my_input.v`* `-o` *`graph.dot`*

  `dot -Tpdf -o` *`graph.pdf graph.dot`*

- **How:** Verilator XML, into Python, which builds a Graphvis image

Alternative 1: Use vhier, and write a Perl script. This supports 99% of SV2017, but without elaboration.
Alternative 2: Verilator –debug produces a similar .dot file as part of normal processing

#9 – Modify Code Without Modifying Code

- **Q:** How can I modify input to Verilator without changing primary sources?

  Example: Convert "`// lint-off MYCOMPANY_RULE_NAME`" to Verilator pragmas

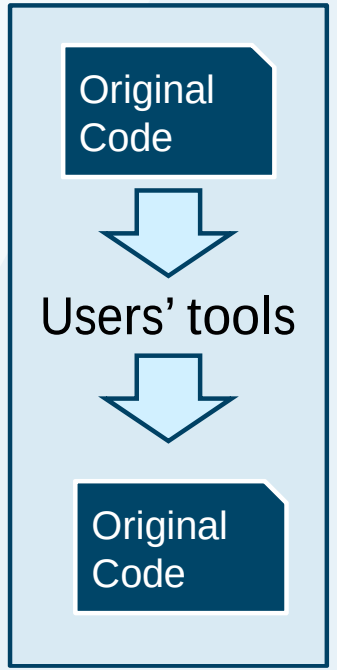
- **A:** Write a "pipe filter"

  Pass to "`verilator --pipe-filter` *`my_filter.py`*"

  This script finds the meta comments and prepends

```
`verilator_config
lint_off -msg RULE …
`verilog
```

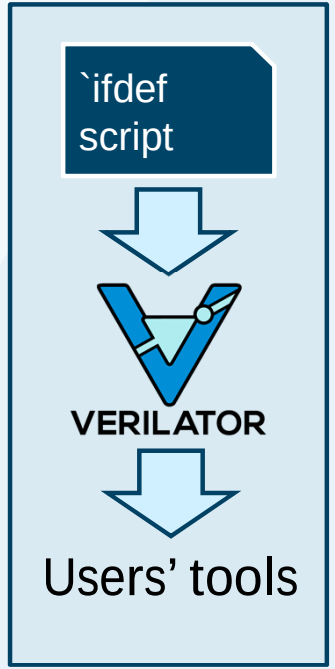
- **How:** Verilator passes all files read through the pipe



#8 – Preprocess a Script's Input

- **Q:** How can I conditionalize synthesis constraints based on IP version?
  - Or more generally, how do I conditionalize any input language?
- **A:** Use Verilator to Preprocess

  Have your script open(“`verilator -E` *`my_input.v`* `|`”)
  The input may then use `` `ifdef/`endif/`include `` etc, wrapping any arbitrary text (e.g. YAML)
  Enables the same `+defines` to feed all your toolage
- **How:** Verilator reads the input(s) and writes preprocessed output to STDOUT



Alternative: Use Verilog-Perl's (older) vppreproc.

#7 – Visualize Line Coverage

- **Q:** Is my code line (/statement) covered, and where in a test is it covered?

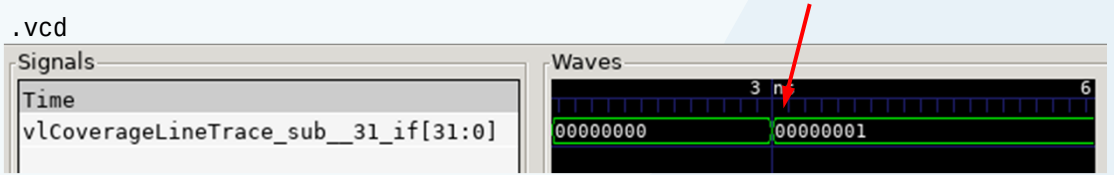
- **A:** Use Verilator

  `verilator --coverage-line --trace --trace-coverage` NEW

.log

```
                count_c <= count_c + 1;
%000001         if (count_c >= 3) begin
```

.vcd



- **How:** Verilator adds a waveform signal that increments when each line is executed

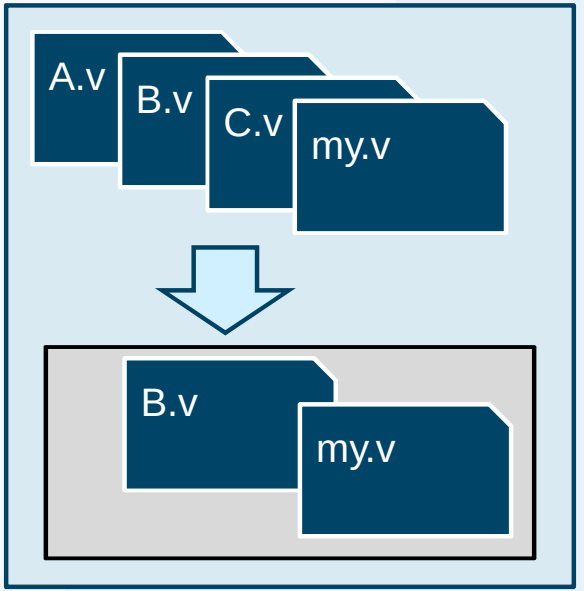
#6 – Tarball Verilog Inputs

- **Q:** How do I package up all the files needed to process just one .v file (e.g. to feed to a vendor)?

- **A:** Use Verilator

  See NEW `examples/xml_py/vl_file_copy`

  Run `vl_file_copy` *`my.v`* `-o` *`subdir/`*



- **How:** Verilator XML determines what files mentioned, fed into Python, which copies the files to *`subdir/`*

Alternative: Use vhier, and write a Perl script, but that does not support elaboration.

#5 – Find Reset/X Issues

- **Q:** Is my design broken due to missing resets?

```
always_ff @(posedge clk)
   reg <= 1'b0;
   if (reg) $halt_and_catch_fire;  // This will never fire, 1'bx is false
end
```

- **A:** Use Verilator!

  `--x-assign unique --x-initial unique`

  Call `Verilated::randReset(2)` in your test

  Run random regressions

  50% of your random tests will flag this broken code!

- **How:** Unknowns are randomized just like real physical flops

#4 – Timing-Accurate Customer Debugger

- **Q:** I need customers to have a timing-accurate model of my design within a development tool

- **A:** Use Verilator!

  Convert your Verilog to C++, then embed inside your tool, with your tool controlling your main loop

NEW Use –protect-ids to encrypt internal symbols

Example: Atmel Studio Designer
(for the chips that power Arduino!)

#3 – Find Improper Synthesis Pragmas

- **Q:** Are my "`// synthesis full_case/parallel_case`" (or "`priority case`") incorrect?

- **A:** Use Verilator

  `verilator --assert`

  Then run your (randomized) regression suite.

- **How:** Verilator adds automatic assertions on these constructs

Alternative: Use Yosys to formally prove assertions.

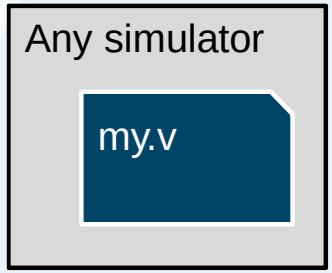
#2 – Attach to the Real World

- **Q:** I'm need real-world stimulus to test my design
  - Example: Real packets into a CPU/Ethernet switch
  - Example: HTTPS acceleration hardware

- **A:** Use Verilator!
  - Use the DPI to connect to C
  - The C file receives and sends packets in raw-socket mode
  - Your simulated system/switch now behaves as if "real" switch
  - Can network into it like any other "real" system
  - Lots of other cool applications for the DPI!

#1 – Embedding in Another Simulator

- **Q:** I want my Verilated code inside another Verilog simulator
  - Example: Faster simulation with mixed-signal models
  - Example: Protecting a library instead of needing encryption
  - Example: Speed up Verilation by separating libraries and other components



- **A:** Use Verilator!
  - NEW Verilator –protect-lib will create a Verilog wrapper for simulators
  - Some performance loss, but this is being improved…
- **How:** Creates a Verilog wrapper that calls C++ via DPI

#0 – Embedding in Python [Soon]

- **Q:** I want a Python package that represents my design
  - Example: Image compression

- **A:** Use Verilator!
  - 

[Soon] Verilator can create a Python library of your code!

- **How:** Uses pybind11 to link Verilated C++ into Python

Resources

- Verilator and open source design tools at http://www.veripool.org
  - Downloads
  - Bug Reporting
  - User Forums
  - News (add yourself as a watcher to see releases)
  - Presentations at http://www.veripool.org/papers/