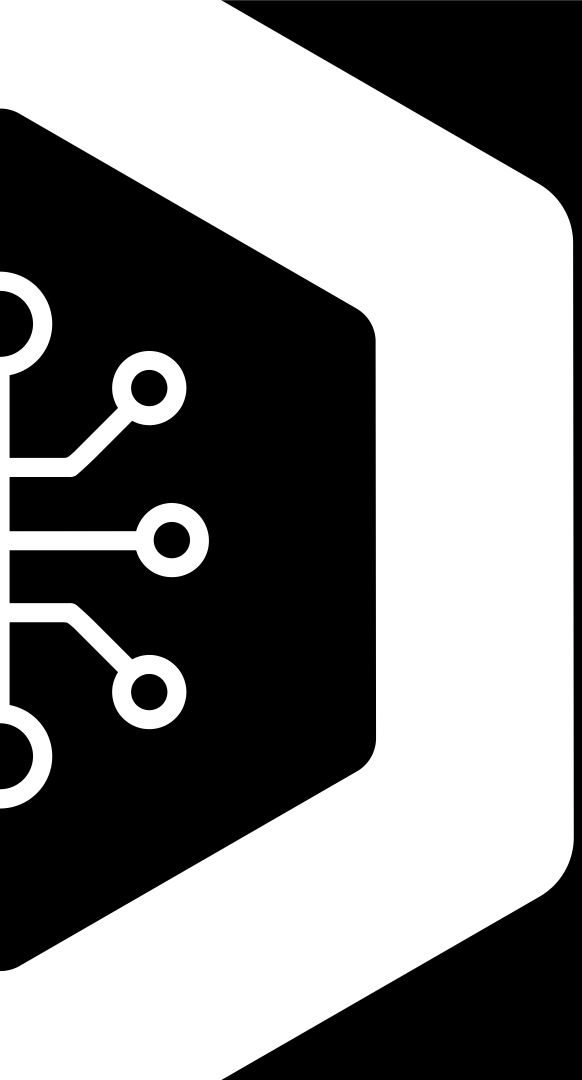


Recent updates in and around Verilator for architectural exploration, testing, verification and coverage reporting

DAC 2025, San Francisco, 2025-06-24

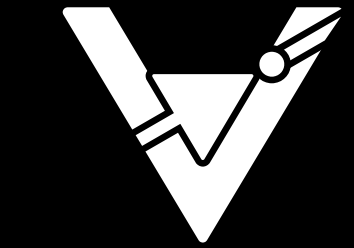
Michael Gielda, mgielda@antmicro.com

antmicro

antmicro

Introduction

- Expanding Verilator's use cases and making it better for existing ones, stemming from Antmicro's own needs and commercial engineering support for a variety of customer projects
- Also creating complementing tools for collaborative RTL design / test development (see CHIPS Alliance session 10:30 tomorrow)
- This presentation outlines some latest updates
- Also see earlier articles on optimizing for very large designs, dynamic scheduling, UVM support, coverage reporting, hierarchical verilation



VERILATOR

antmicro

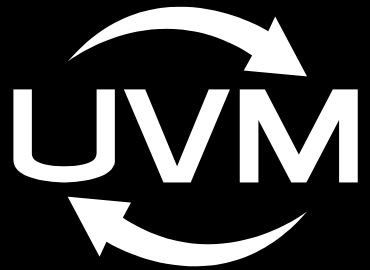
Plan

- Extending use cases
  - UVM verification
  - Coverage reporting
  - Power estimation
- Performance analysis and improvements

antmicro

Continuous UVM support improvements

- Ongoing development to enable more complex UVM testbenches support in Verilator
- Currently working on adding support for the features needed to run samples from the UVM Cookbook (15/33 passing as of now, but many of the failures are small corner cases)
  - biggest missing feature is functional coverage support - current approach will be to turn it off in the cookbook tests (before it's implemented)
- Progress trackable on GitHub antmicro.github.io/verilator-verification-features-tests/log.html (cookbook tests to be merged soon)



antmicro

Power analysis with OpenSTA/OpenROAD and Verilator

- Fast, fully open source flow for static, peak and glitch power analysis using parts of the OpenROAD flow
- Example: github.com/antmicro/verilog-power-analysis-workflows
- This required adding UDP and SAIF support
  - UDP - SystemVerilog feature allowing defining complex primitive blocks modelling combinational or sequential logic
  - PDKs often use this feature in the cells simulation models
  - Started in PR#5807, extended & merged in PR#5936
  - SAIF - Switching Activity Interchange Format
  - Contains information about changes of signals over time
  - Alternative to VCD files, which tend to grow in size quickly

Static power analysis workflow

```
Annotated 212 pin activities.
Group                  Internal  Switching    Leakage      Total
                          Power      Power      Power      Power
(Watts)
----------------------------------------------------------------
Sequential             4.06e+00   9.64e-02   9.49e-09   4.16e+00  43.9%
Combinational          8.64e-01   7.45e-01   3.43e-08   1.61e+00  17.0%
Clock                  2.65e+00   1.05e+00   1.92e-09   3.70e+00  39.1%
Macro                  0.00e+00   0.00e+00   0.00e+00   0.00e+00   0.0%
Pad                    0.00e+00   0.00e+00   0.00e+00   0.00e+00   0.0%
----------------------------------------------------------------
Total                  7.58e+00   1.89e+00   4.57e-08   9.47e+00 100.0%
                          80.1%      19.9%       0.0%
```

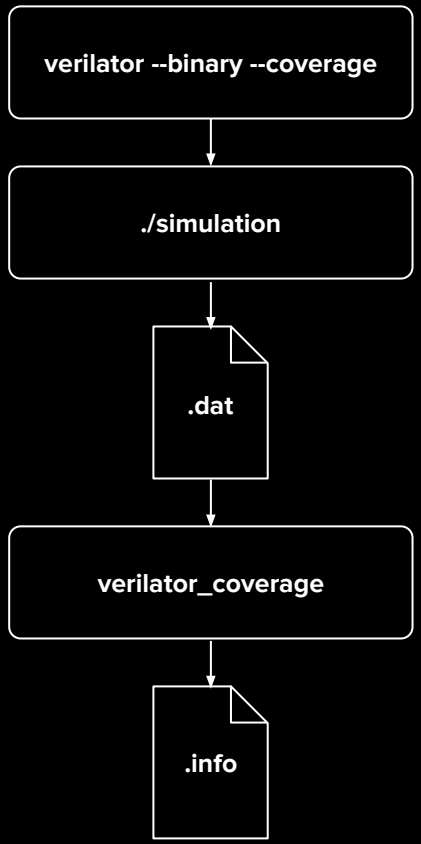
Peak and glitch power analysis workflow

```
...
Processing clock cycle #220
Processing clock cycle #221
Processing clock cycle #222
Processing clock cycle #223
Processing clock cycle #224
Processing clock cycle #225
Processing clock cycle #226
Processing clock cycle #227
Processing clock cycle #228
Maximum power consumption of a single clock cycle is 9.210000047600001
Watts and occurred in clock cycle #180
```

antmicro

Multiple ‘type’ coverage reports

- Previously it was not possible to generate separate reports for multiple coverage types, you needed to run Verilator a few times to get those
- We added an option to select **type** in the verilator_coverage post-processing tool, which allows user to generate multiple separate reports running verilation and simulation only once
- verilator_coverage takes much less time than the simulation itself so it’s a significant time saver



antmicro

Toggle coverage in genblocks enabled

```
for (genvar i = 0; i < P; i++) begin
   logic x;
   always @ (posedge clk) begin
      x <= toggle;
   end
   for (genvar j = 0; j < 3; j++) begin
      logic [2:0] y;
      always @ (negedge clk) begin
         y <= {toggle, ~toggle, 1'b1};
      end
   end
end
if (P > 1) begin : gen_1
   assign z = 1;
end
```

antmicro

Include ternary operator in branch coverage

```
assign a = (cyc == 1) ? 0 : clk;
assign b = (cyc % 3 == 1) ? (clk ? 1 : 0) : 1;
```

antmicro

Accurate toggle coverage

- We are working on having separate toggle coverage counters for changes 0 -> 1 and 1 -> 0
- This feature is under review in PR#6086

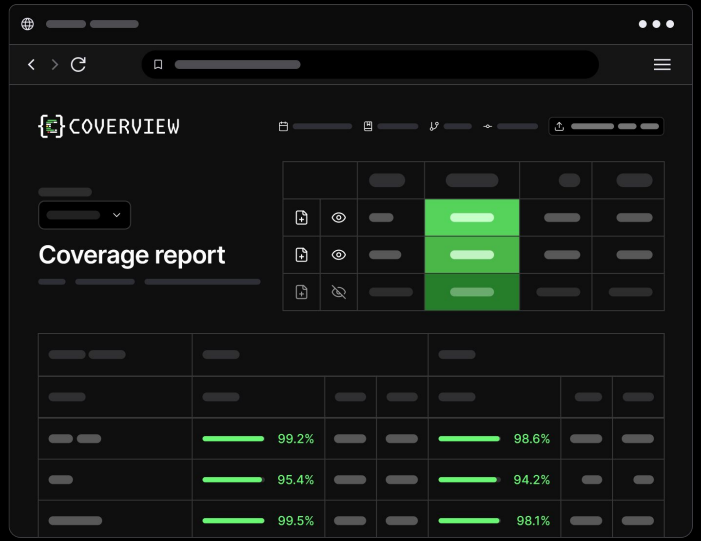
```
⊖ 2/2                :  input       rd_en,      // 1 bit  Read Enable from JTAG
  1  1                   rd_en[0]_0→1  rd_en[0]_1→0
⊖ 2/2                :  input       wr_en,      // 1 bit  Write enable from JTAG
  1  1                   wr_en[0]_0→1  wr_en[0]_1→0
                     :
                     :  // Processor Signals
⊖ 2/2                :  input       rst_n,      // Core reset
  1  1                   rst_n[0]_0→1  rst_n[0]_1→0
⊖ 2/2                :  input       clk,        // Core clock
  1  1                   clk[0]_0→1  clk[0]_1→0
```

antmicro

Coverview

Coverview is an open source tool we created last year that lets you generate coverage dashboards

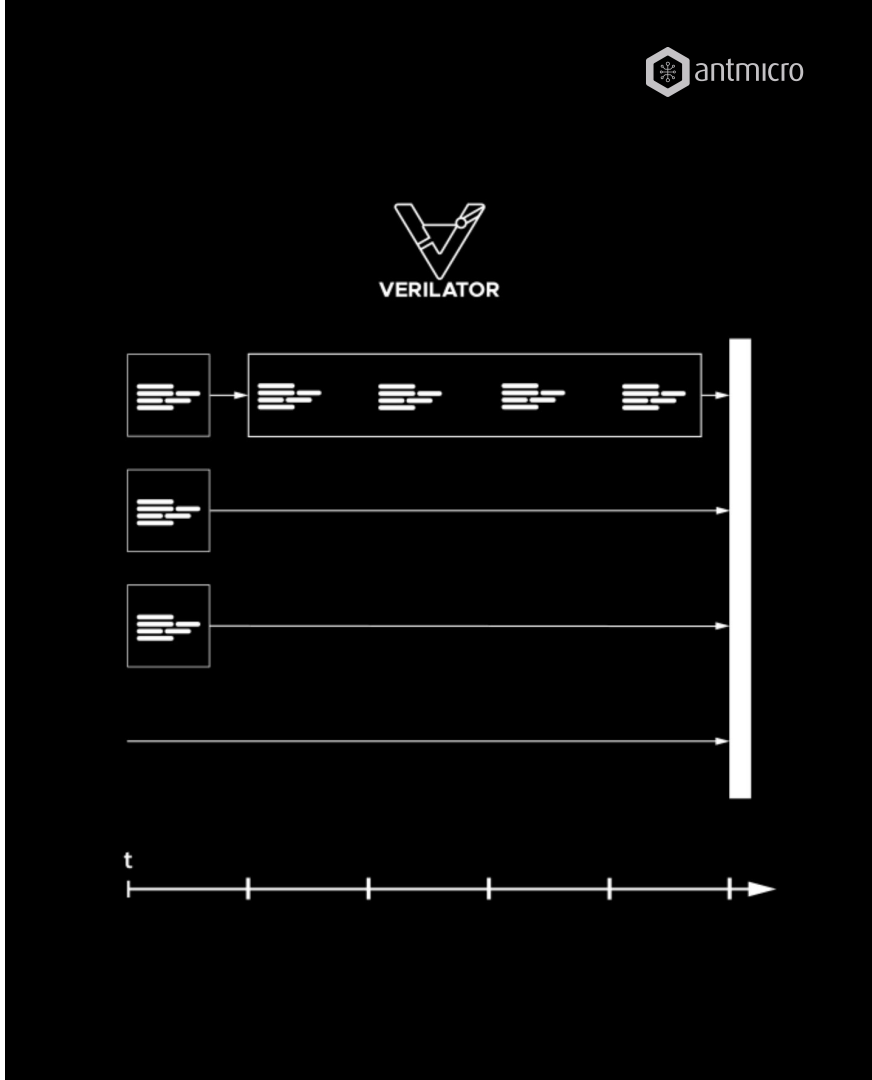
- allows to incrementally load coverage results from different sources
- aggregates multiple coverage types into a single-page, unified dashboard
- a standalone version is deployed on GitHub pages, which you can use by simply uploading data
- github.com/antmicro/coverview



antmicro

Verilation and simulation time optimizations

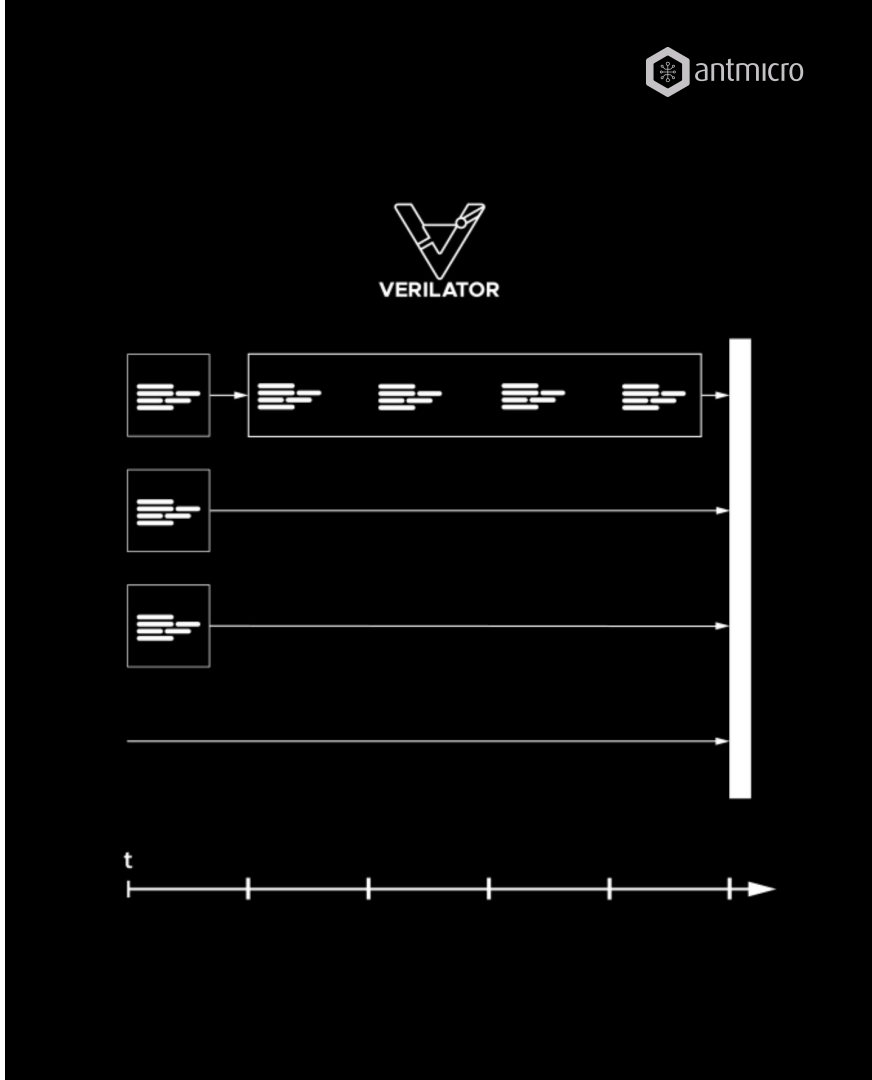
- Recently we have been working on improving Verilator runtime (both Verilation = translation from SV to C++, compilation of the C++ as well as simulation time = execution of the C++)
- Focusing on hierarchical, multithreaded flows



antmicro

Hierarchical verilation

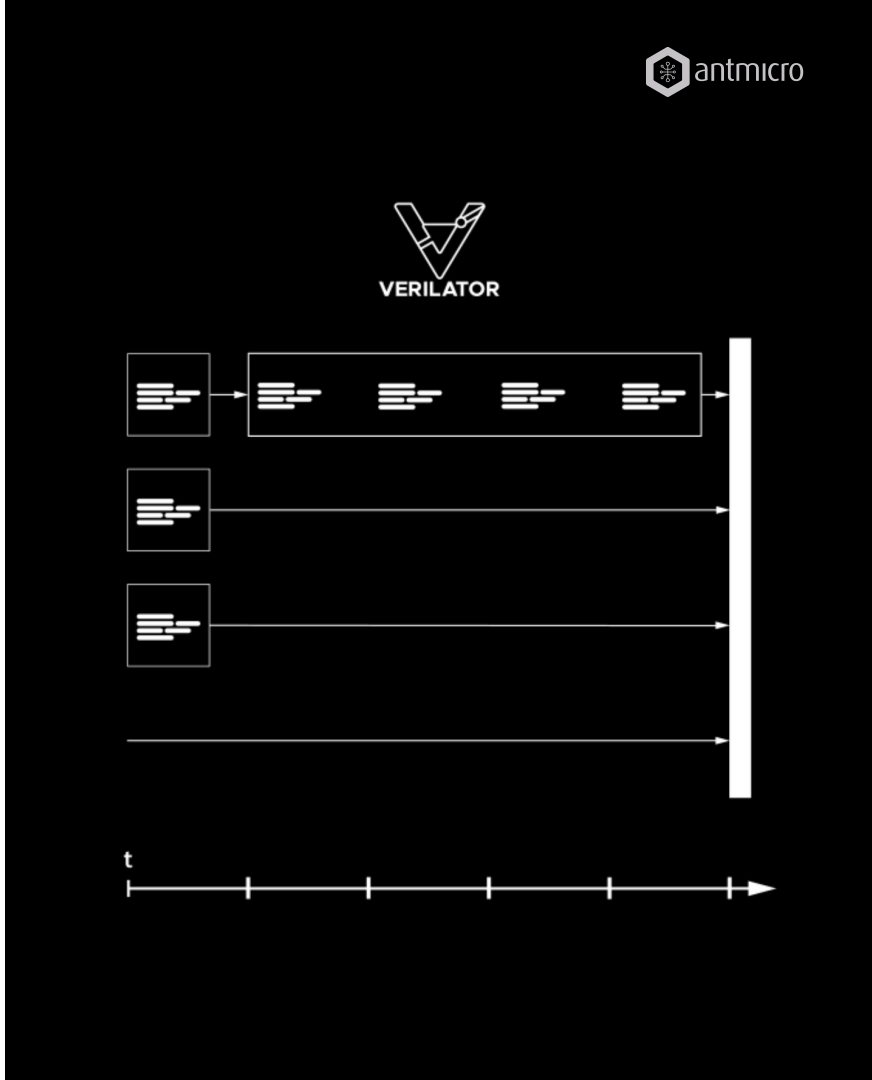
- Normally, when verilating the original HDL as a whole, all modules are coupled with each other, comprising a single verilation unit
- Any change in the code triggers (re-)verilation for the entire design
- Hierarchical verilation lets the user divide their design into parts and each of them acts as a separate verilation unit



antmicro

Allow scheduling hier block on multiple threads

- Previously, a hier block could be scheduled only on 1 thread
- Now possible to schedule a hier block on more threads which can increase the performance of the simulation as a whole
- The number of threads (workers) is specified by the user
- hier_workers -module "<module_name>" -workers "<worker_count>" in your .vlt config file

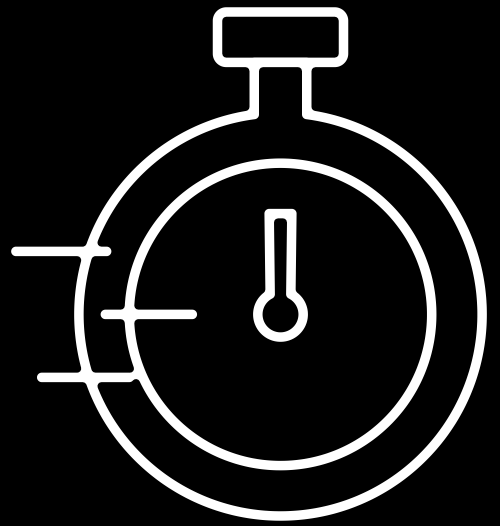


antmicro

Improved scheduling

Verilator schedules tasks based on their estimated costs

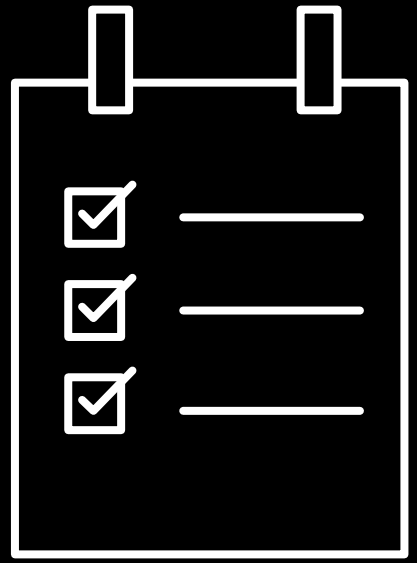
- Added support for Profile Guided Optimization of threads in **hier_blocks**
- Enabled evaluation of costs inside **hier_blocks**



antmicro

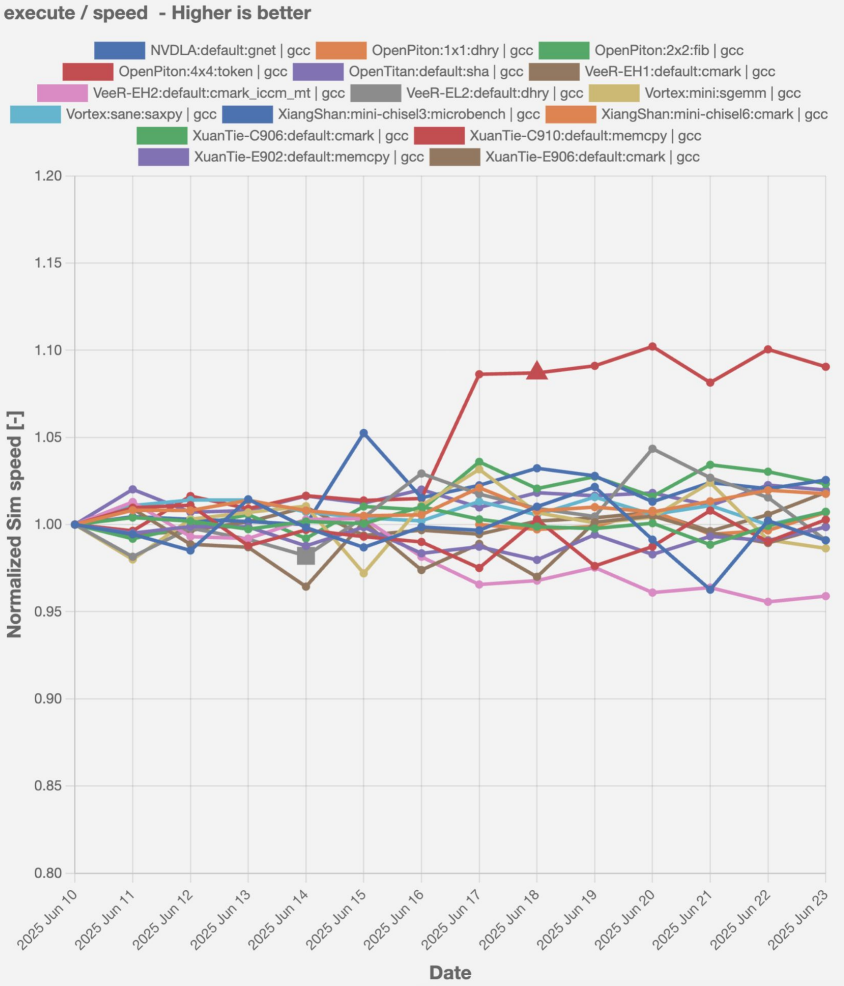
Relaxed hier_block requirements and enhanced usability

- Added support for marking module with type parameters as **hier_block**
- Added support for integer atom types and signed primitives
- Fixing hierarchical verilation for projects with **dot-f** dependency lists
- Fixing the **-j** option in hierarchical verilation
- Improved **verilator_gantt**, which is a tool that visualizes scheduling tasks on threads



Performance benchmarks

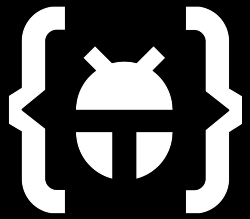
- RTLMeter developed and released by Geza Lore runs performance checks on Verilator using a number of real designs (including VeeR cores we're maintaining)
- It collects and presents historical data allowing tracking performance regressions
- Dashboard available at verilator.github.io/verilator-rtlmeter-results
- Currently supports Verilator, but should be usable with other tools
- We're planning to add more designs e.g. our open source I3C core



antmicro

Simplifying reporting issues

- sv-bugpoint is a tool for minimizing SystemVerilog code while preserving a user-defined behavior of that code, e.g:
  - Verilation errors
  - Compilation errors
  - Infinite loops in simulation (i.e. hanging)
  - Incorrect simulation results
- Saves a ton of time when debugging
- Recommended for use when reporting issues to Verilator
- Can also be used with tools other than Verilator



sv-bugpoint

THANK YOU
FOR YOUR ATTENTION!