Enabling Collaborative Chip Design in the RISC-V VeeR Core and Caliptra RoT Project with CHIPS Alliance Tools

RISC-V Summit Europe, Barcelona, 2023-06-07

Karol Gugala, kgugala@antmicro.com
Matt Cockrell, mcockrell@google.com

antmicro

CHIPS ALLIANCE

- Organization which develops and hosts:
  - Open source hardware code & specs -> open source CPUs, I/O IP, interconnects, ML
  - Open source ASIC & FPGA development tools -> design, verification, simulation, workflows
- A barrier-free environment for collaboration:
  - Standards organization framework for collaboration and development
  - Under governance of Linux Foundation
- Legal framework – Apache v2 license / OWFa
- Shared resources ($, code, practices, infrastructure and time) which lower the cost of hardware development

CHIPS ALLIANCE

THE LINUX FOUNDATION

VEER

- Family of open source 32 bit RISC-V CPU cores
- Three options
  - EL2 - tiny and low power core
  - EH1 - high performance core
  - EH2 - dual threaded successor to EH1
- All the variants are implemented with SystemVerilog
- RTL is heavily ASIC proven

CALIPTRA

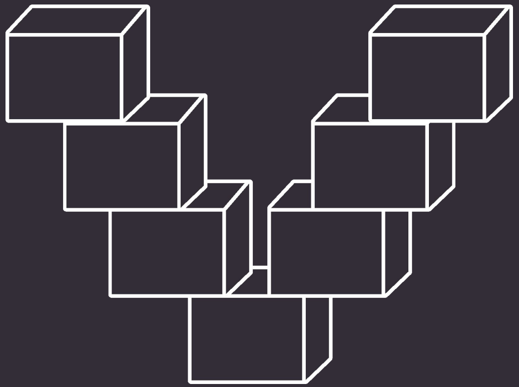
- Open source integrated Root of Trust block
- Boot Media Integrated/Dependent variants
- Collaboration between Google, Nvidia, Microsoft and AMD within CHIPS Alliance (spec lives in OCP, implementation is developed in CHIPS)
- Aims to build a component easy to integrate with bigger SoC
- Uses VeeR EL2 CPU core
- Reuses a number of OpenTitan peripheral cores

CALIPTRA AND ANTMICRO

- Focus on enabling open source collaboration on the project for both current project partners and external adoption
- Took over maintenance of VeeR cores family
- Main goal: extended the Caliptra/VeeR ecosystem to cover not just RTL but also tools and test flows to enable open source collaboration:
  - Tools for working with code
  - System-level testing and integration
  - Public-facing CI

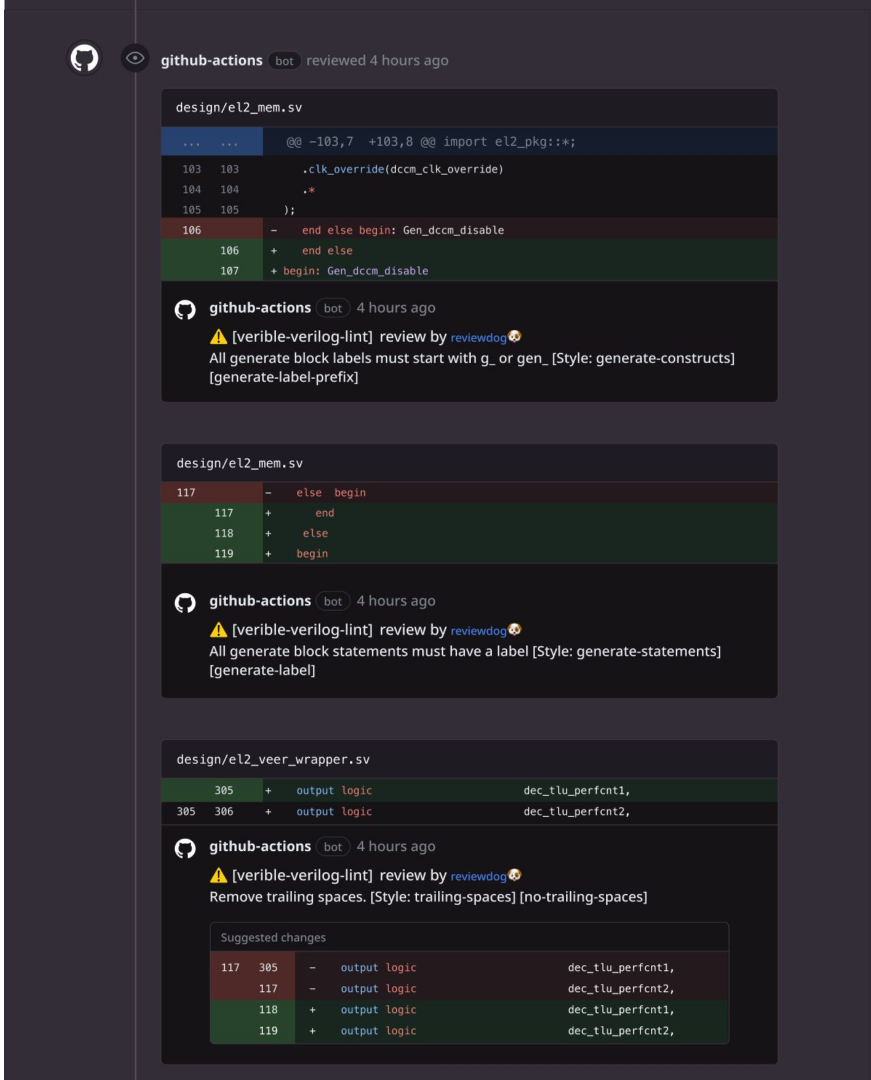
VERIBLE

- Developed by Google and donated to CHIPS Alliance
- Open Source suite of SystemVerilog development tools
  - Linter,
  - Formatter,
  - Lexical diff, obfuscator, indexer...
  - Language Server



VERIBLE GH ACTIONS

- Linter and formatter actions
  - chipsalliance/verible-formatter-action
  - chipsalliance/verible-linter-action
- Super easy to integrate with any Verilog/SystemVerilog repository
  - Actions provide automatic feedback in PRs
  - Formatter provides code changes suggestions applicable directly from a PR



github-actions bot reviewed 4 hours ago

design/el2_mem.sv

```
@@ -103,7 +103,8 @@ import el2_pkg::*;
      .clk_override(dccm_clk_override)
      .*
   );
-     end else begin: Gen_dccm_disable
+     end else
+ begin: Gen_dccm_disable
```

github-actions bot 4 hours ago

⚠ [verible-verilog-lint] review by reviewdog
All generate block labels must start with g_ or gen_ [Style: generate-constructs] [generate-label-prefix]

design/el2_mem.sv

```
-    else  begin
+       end
+     else
+    begin
```

github-actions bot 4 hours ago

⚠ [verible-verilog-lint] review by reviewdog
All generate block statements must have a label [Style: generate-statements] [generate-label]

design/el2_veer_wrapper.sv

```
+    output logic                                    dec_tlu_perfcnt1,
+    output logic                                    dec_tlu_perfcnt2,
```

github-actions bot 4 hours ago

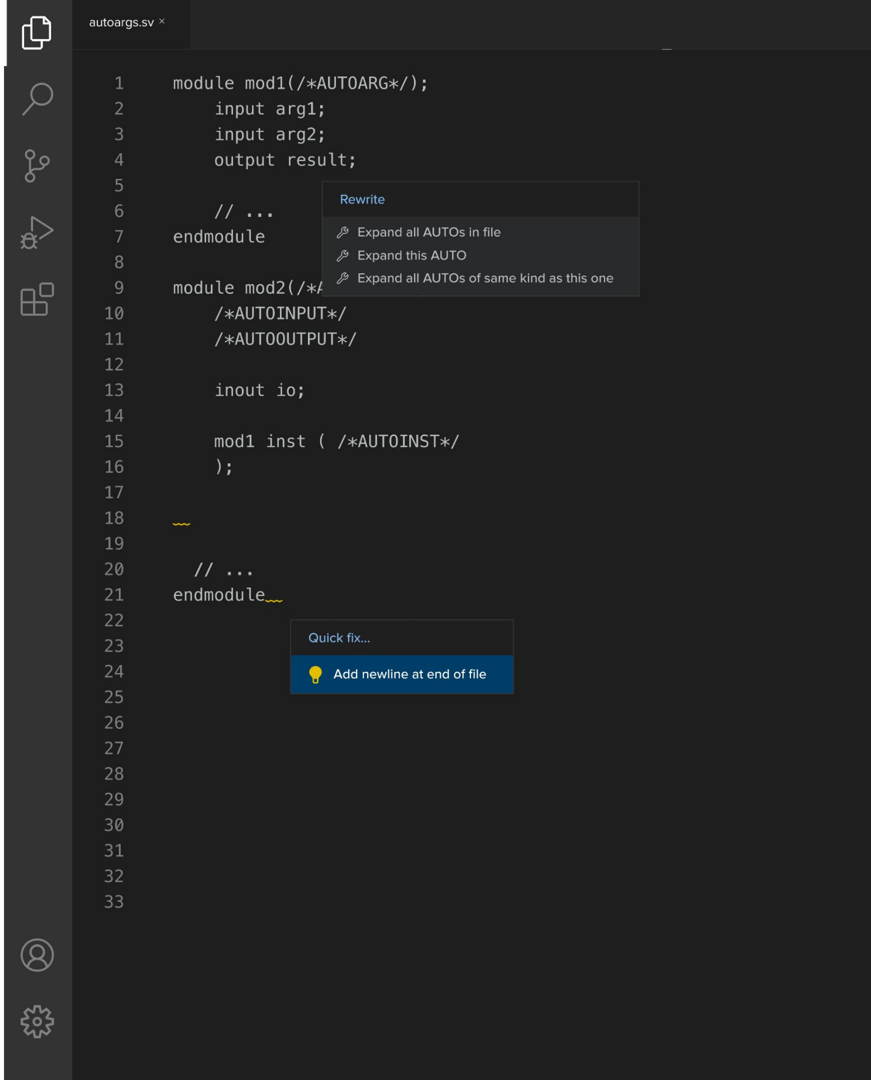
⚠ [verible-verilog-lint] review by reviewdog
Remove trailing spaces. [Style: trailing-spaces] [no-trailing-spaces]

Suggested changes

```
-    output logic                                    dec_tlu_perfcnt1,
-    output logic                                    dec_tlu_perfcnt2,
+    output logic                                    dec_tlu_perfcnt1,
+    output logic                                    dec_tlu_perfcnt2,
```

SYSTEM VERILOG LANGUAGE SERVER

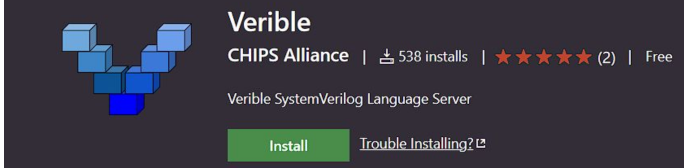
- Verible provides LSP functionality
    - Documentation provides information how to enable it for various editors
    - We handle most of standard LFS features
    - We also support AUTO-expansion (similar to Emacs Verilog-mode)



```
module mod1(/*AUTOARG*/);
    input arg1;
    input arg2;
    output result;

    // ...
endmodule

module mod2(/*A
    /*AUTOINPUT*/
    /*AUTOOUTPUT*/

    inout io;

    mod1 inst ( /*AUTOINST*/
    );



  // ...
endmodule
```

VERIBLE VS CODE PLUGIN

- For VSCode we have a plugin enabling this functionality
  - The plugin comes with Verilbe binaries for Linux and Windows



Verible

CHIPS Alliance | 538 installs | ★★★★★ (2) | Free

Verible SystemVerilog Language Server

VEER TESTING

- VeeR cores family originally only included system level tests
  - More complex test suite was internal to the original authors and was not released as open source
- One of the most important goals here was to extend VeeR's publicly available test suite with tests for specific core components
- Added public CI to the VeeR repo running those tests for every commit
- The suite includes code coverage reporting - the CI generates a summary webpage

OPEN SOURCE VERIFICATION

- Caliptra's primary use case is integration into bigger SoCs
- The primary testing methodology is UVM based based on widespread use in founding organizations
- UVM simulations are not yet possible with open source tools only
- There is ongoing effort on enabling UVM simulation in Verilator
- For the time being, to enable open source collaboration for improving testing coverage, we added Cocotb and pyuvm testing which can be run completely open source in CI

SystemVerilog

UVM

VERILATOR

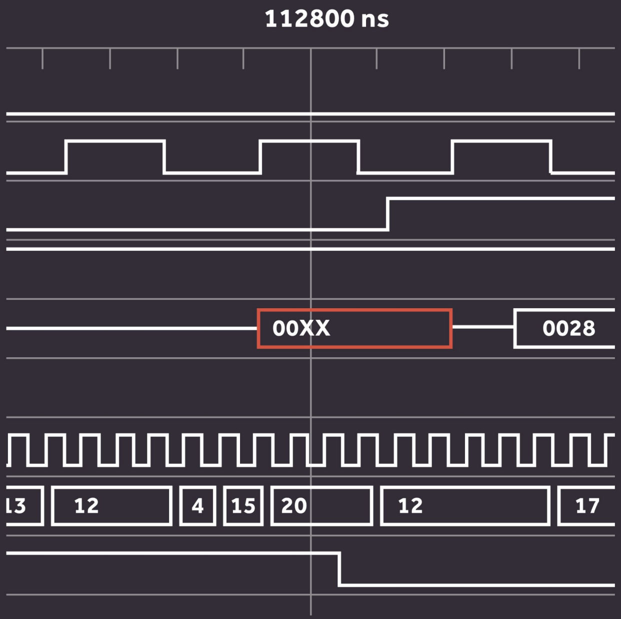
RISCV-DV

- riscv-dv is an open source instruction generator for RISC-V processor verification
  - Originally developed by Google, donated to CHIPS Alliance
- To make it work with the current EL2 codebase, VeeR's out of order div/rem block required special handling in the generator
  - We improved trace analyzer and code generator to handle the out-of-order division logic
- riscv-dv tests are now part of VeeR's public CI

```
0x8000bfa8:   01010113  addi sp, sp, 16
0x8000bfac:   00008067  ret
0x8000c0b8:   fff00793  li a5, -1
0x8000c0bc:   00f50e63  beq a0, a5, 28
0x8000c0c0:   00a4dc63  bge s1, a0, 24
0x8000c0d8:   00847413  andi s0, s0, 8
0x8000c0dc:   30042473  csrrs s0, mstatus, s0
0x8000c0e0:   01c12083  lw ra, 28(sp)
0x8000c0e4:   01812403  lw s0, 24(sp)
0x8000c0e8:   01412483  lw s1, 20(sp)
0x8000c0ec:   02010113  addi sp, sp, 32
0x8000c0f0:   00008067  ret
0x8000cac4:   de1f50ef  jal -41504
0x800028a4:   00800793  li a5, 8
0x800028a8:   3007a7f3  csrrs a5, mstatus, a5
0x800028ac:   10500073  wfi
0x80000010:   fb010113  addi sp, sp, -80
```

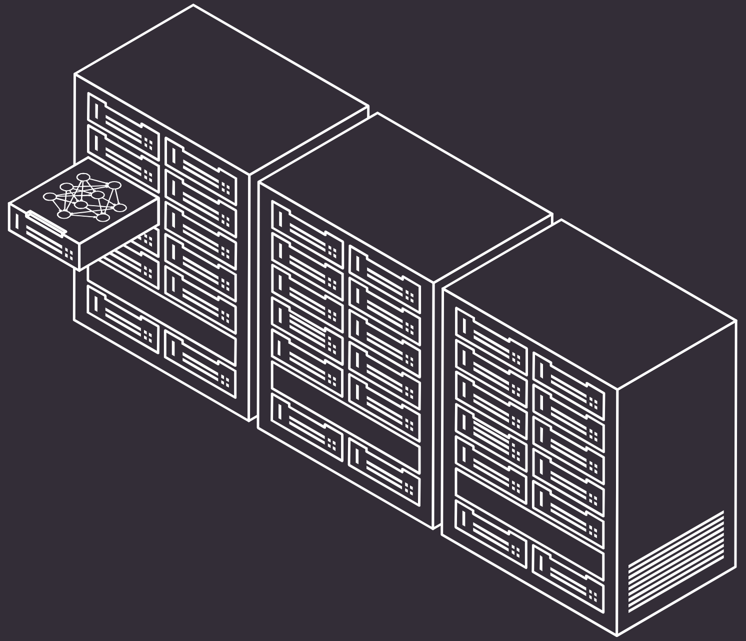
JTAG INTEGRATION

- For debugging the core, Caliptra added a JTAG interface, which needed its own end-to-end tests
  - The test connects to a simulated SoC with OpenOCD and runs typical debug scenarios
  - Run in public CI on GitHub with open source tools
- Verification showed an issue with accessing a peripheral from debug interface
  - The issue has been resolved and the case is now covered with tests



CUSTOM GH RUNNERS

- Verification flows require non-trival amount of computing resources and runtime
- Standard runners available on GH cannot provide required resources
- Some verification tasks need to be run using proprietary tools
- Antmicro’s open source custom GH runners can spawn a number of required machines and delegate build/test tasks to them
- We support third party tools and licenses
  - Job results can be filtered to not leak proprietary information
- Open source and available on GitHub



LEARN MORE

- ORConf 2023
  - Munich, September 15-17
  - Register at orconf.org
- CHIPS Alliance Technology Update colocated with DAC
  - San Francisco, July 13th
  - events.linuxfoundation.org/chips-biannual-technology-update

antmicro

THANK YOU
FOR YOUR ATTENTION!