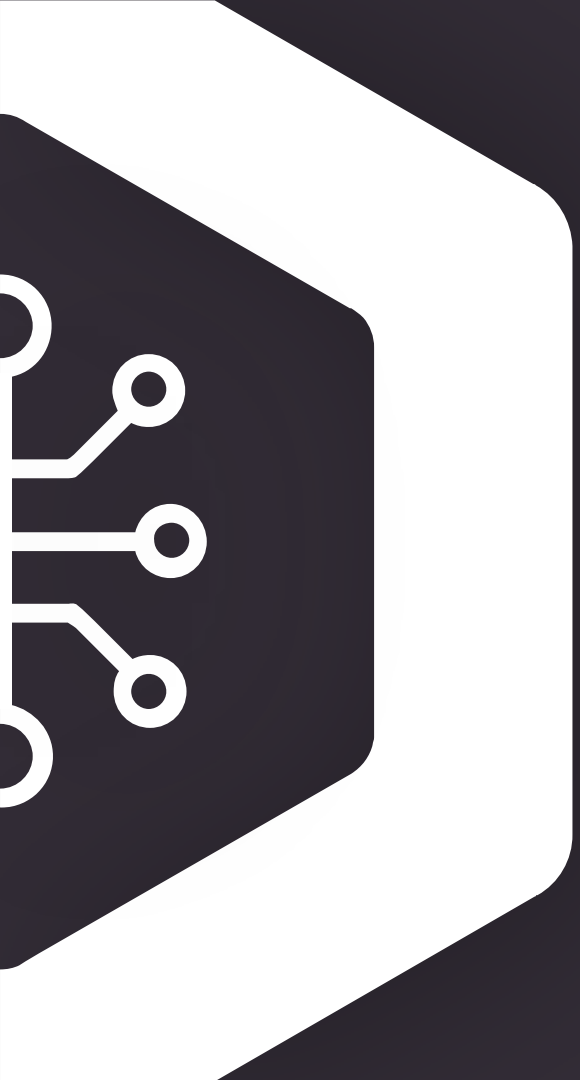


Unlocking open source RISC-V SoC verification

RISC-V Week, Paris, 2022-05-05

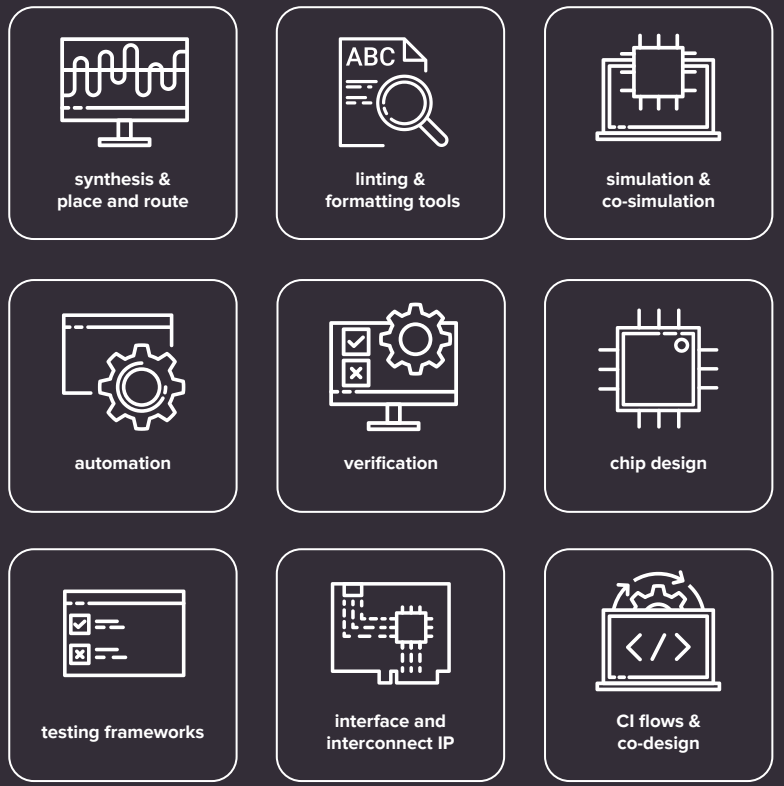
Michael Gielda, mgielda@antmicro.com

antmicro

antmicro

ANTMICRO: COMMERCIAL OPEN SOURCE SUPPORT & DEVELOPMENT SERVICES

- Much of Antmicro's work revolves around **open source FPGA and ASIC tooling** development where we work with partners such as Google, Western Digital, Microchip and SiFive
- We develop tools for synthesis, place and route, linting, formatting, simulation & co-simulation, hardware-software co-design, design verification & automation
- These tools allow us to effectively build **future proof systems, CI-driven flows** and enable **parallelization** of development for our customers



antmicro

ANTMICRO & CHIPS ALLIANCE

- Like Antmicro, CHIPS Alliance believes that on top of RISC-V, open source silicon requires un-core IP and a vibrant ecosystem of open source tools
- CHIPS coordinates with RISC-V closely since RISC-V is our de-facto primary ecosystem
- CHIPS believes open tools are possible and necessary for unconstrained collaboration, unlocking software-driven workflows & AI-assisted ASIC design
- We're hosting **OpenROAD/OpenLANE/OpenFASoC**
- Home to the open source **F4PGA toolchain**, the **FPGA Interchange Format**
- Home to **Chisel** and many other **HDL-related tools**
- Home to our **open source verification** efforts!

antmicro

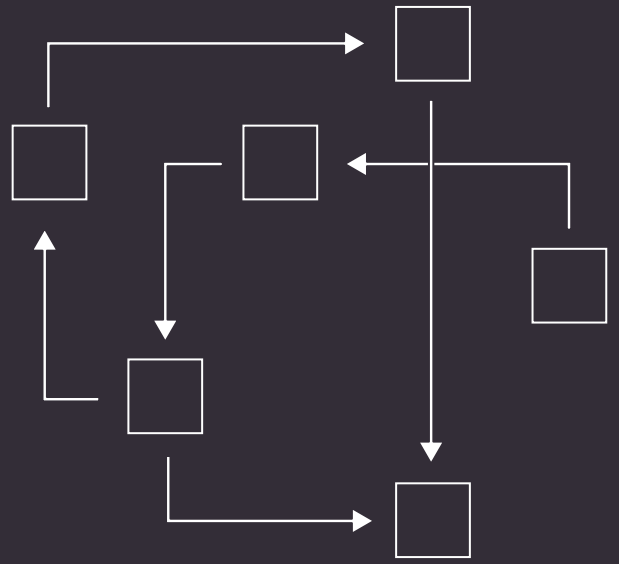
WHY DO WE NEED OPEN SOURCE UVM / SYSTEMVERILOG SUPPORT?

antmicro

BRIDGING NEW AND EXISTING METHODOLOGIES

- While RISC-V is spawning new approaches to HW design which offer tangible improvements, there is an immense body of pre-existing work
- Sustainable results require an inclusive approach towards existing pool of designs and designers
- We need to make it easy to combine and remix methodologies

Important progress has been made in verification space - and we're now analyzing how to deploy this to production for some limited use cases!
(this still means a lot of work ahead of us)



WHAT WOULD OPEN SOURCE UVM/SYSTEMVERILOG SUPPORT ENABLE?

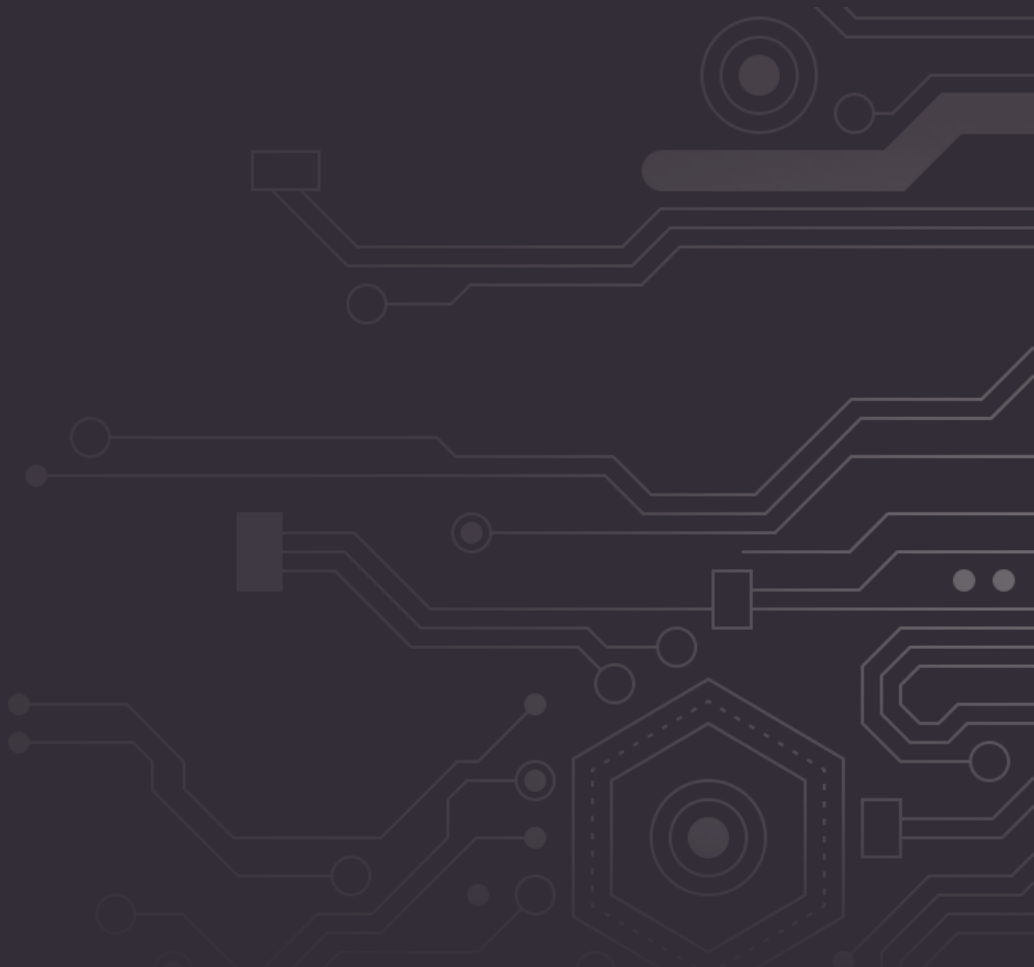
- Combining commercial ecosystem with open source tools and methodologies
- Chip-making companies can benefit from open source while keeping their existing UVM codebase
- Lack of licensing limitations would allow scalable, reproducible CIs
- Number of open source cores and lots of pre-existing IP implemented in SystemVerilog, e.g.
  - SweRV
  - Ibex
  - BlackParrot
  - Core-V
- Building a collaborative ecosystem around ASICs
- Proving open source tools are really possible

HOW TO GET THERE?

- Identify missing functionalities and features
- Reuse existing solutions
  - There are many existing projects which can be improved
- Create well documented and transparent flows
  - Include automated tests and status reporting in projects
- Cooperate with others
  - Gather information on what is needed
- **Provide incremental value**

antmicro

WHAT IS MISSING?



antmicro

CHECK THE SPEC: SV-TESTS

- We created a test suite to determine the SystemVerilog support level in various open source tools
- Aims to pinpoint all the supported and missing SystemVerilog features in various tools
- Generates report from last passing master build at github.com/chipsalliance/sv-tests
- Introduces three types of tests:
  - Testing individual features as per the SystemVerilog standard
  - Existing third party test suites
  - Selected open source IP cores, such as SweRV, Ibex and others

		icarus	moore	moore_parse	odin	slang	slang_parse	surelog	sv2v_zachjs	sv_parser	tree_sitter_verilog	uhdmverilator	uhdmyosys	verible
Keywords	5.6.2	248/248	248/248	248/248	248/248	248/248	248/248	243/248	248/248	248/248	2/248	243/248	243/248	246/248
System tasks and system functions	5.6.3	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	0/1	1/1	1/1	1/1
Compiler directives	5.6.4	64/66	76/90	60/66	1/90	90/90	90/90	90/90	90/90	90/90	63/66	90/90	90/90	63/66
Integer literal constants	5.7.1	64/64	63/64	62/64	3/64	64/64	64/64	64/64	64/64	64/64	64/64	64/64	64/64	64/64
Real literal constants	5.7.2	58/58	56/58	57/58	1/58	58/58	58/58	58/58	58/58	58/58	58/58	58/58	58/58	58/58
Time literals	5.8	1/1	1/1	1/1	0/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1
String literals	5.9	3/4	3/4	4/4	1/4	4/4	4/4	4/4	4/4	4/4	2/4	3/4	4/4	4/4
Special characters in strings	5.9.1	1/1	0/1	0/1	1/1	1/1	1/1	1/1	1/1	1/1	0/1	1/1	1/1	1/1
Structure literals	5.10	0/3	3/3	3/3	0/3	4/4	3/3	4/4	2/4	3/3	3/3	2/4	4/4	3/3
Array literals	5.11	0/3	3/3	3/3	0/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	2/3	3/3
Attributes	5.12	5/5	5/5	5/5	0/5	5/5	5/5	5/5	3/5	5/5	5/5	5/5	5/5	5/5
Built-in methods	5.13	1/1	0/1	1/1	0/1	1/1	1/1	1/1	1/1	1/1	0/1	1/1	0/1	1/1
Nets and variables	6.5	1/1	1/1	1/1	0/1	4/4	1/1	2/4	1/4	1/1	1/1	2/4	2/4	1/1
Wire and tri nets	6.6.1	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	1/2	2/2	2/2
Unresolved nets	6.6.2	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1
Wired nets	6.6.3	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	0/4	4/4	4/4
Trireg net	6.6.4	0/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	0/1	1/1	1/1
Tri0 and tri1 nets	6.6.5	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2
Supply nets	6.6.6	2/2	2/2	2/2	0/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2
User-defined nettypes	6.6.7	0/2	0/2	0/2	0/2	2/2	2/2	2/2	0/2	2/2	2/2	1/2	1/2	2/2
Generic interconnect	6.6.8	0/1	0/1	0/1	0/1	1/1	1/1	1/1	0/1	1/1	1/1	1/1	1/1	0/1
Specifying vectors	6.9.1	1/1	1/1	1/1	0/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1
Vector net accessibility	6.9.2	3/3	3/3	3/3	1/3	3/3	3/3	3/3	1/3	3/3	3/3	3/3	3/3	1/3

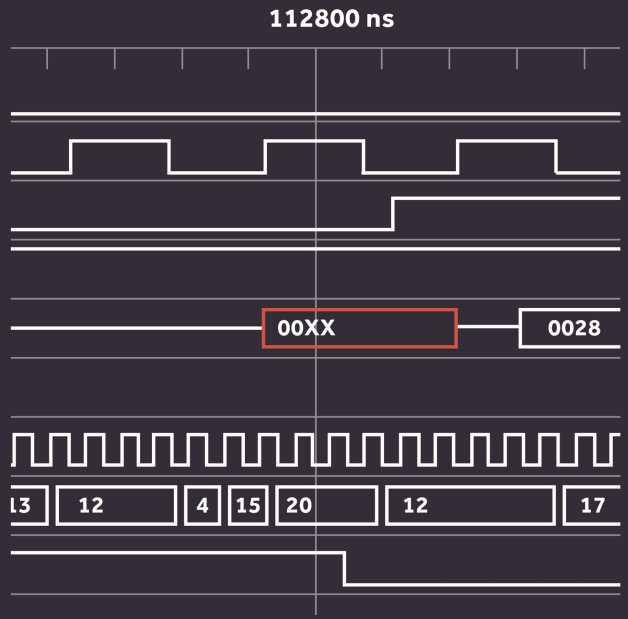
antmicro

Google

antmicro

GET AN EVENT DRIVEN SIMULATOR

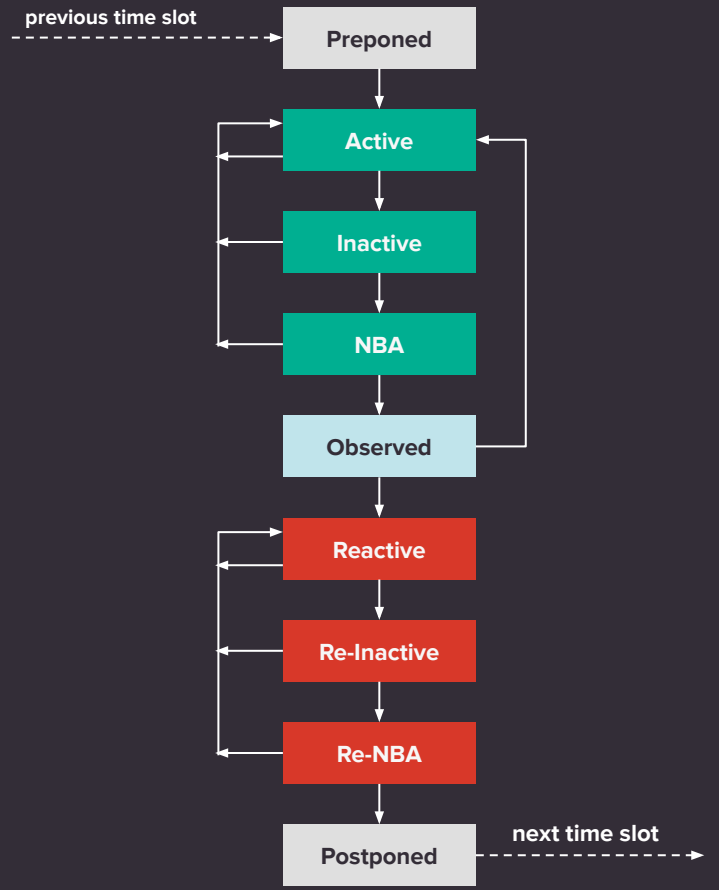
- One of the biggest missing pieces in open source open source UVM is a fast, time aware and event driver simulator
- Verilator was the obvious choice: it's the fastest simulator available on the market
  - no time awareness and events support though!
- To address this we decided to reimplement the way Verilator handles simulation scheduling



antmicro

STRATIFIED SCHEDULER

- SystemVerilog simulation time is divided into time slots which are further divided into regions
- Different types of statements fall into different regions
  - Normal assignments are in Active,
  - Non-blocking assignments are in NBA,
  - Concurrent assertions are in Observed, etc.
- These regions are not simple divisions of time; the simulation can go back to an earlier region within a time slot, e.g. if a non-blocking assignment triggers an active block
- Some events can get delayed to a following time slot



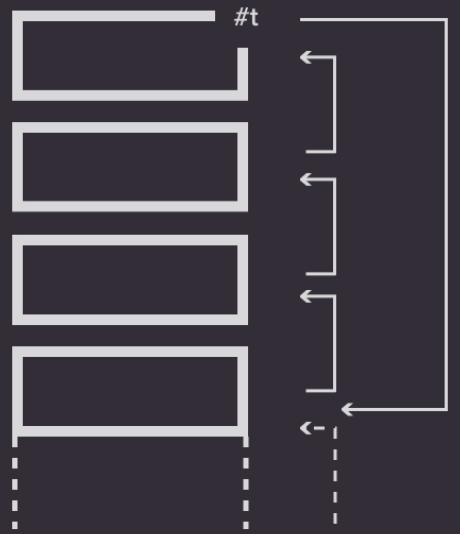
DYNAMIC SCHEDULING

- So far, Verilator only partially implemented a stratified scheduler, but only by statically ordering the generated code
- That is not sufficient in cases where we cannot predict (during compilation) what events get scheduled or delayed and when it happens (such as delays or forks in a deep call stack, or in a virtual function)
- UVM requires a more dynamic approach that allows scheduling events at runtime

antmicro

OUR APPROACH: COROUTINES

- Complete rewrite of the way Verilator schedules events (in several iterations)
- We use C++ coroutines (from C++20!) to handle the scheduling and synchronization of an asynchronous events in the simulation
  - Coroutines are a form of cooperative multitasking – and that's fine for this case!
  - With coroutines, we don't have to worry about multithreading (as much)
  - The overhead for coroutines is significantly smaller than threads
- Read more about the technical details in our blog note



antmicro

WHAT IS ALREADY POSSIBLE

- Delays
- Event variables
- Fork / join
  - `join` (wait for all children to finish)
  - `join_any` (wait for any one child)
  - `join_none` (do not wait, continue the main process immediately)
- Wait statement
- Constrained randomization
  - Not related to scheduling, but needed for UVM
  - Only basic constraints are supported so far; the next step is to add support for a real solver

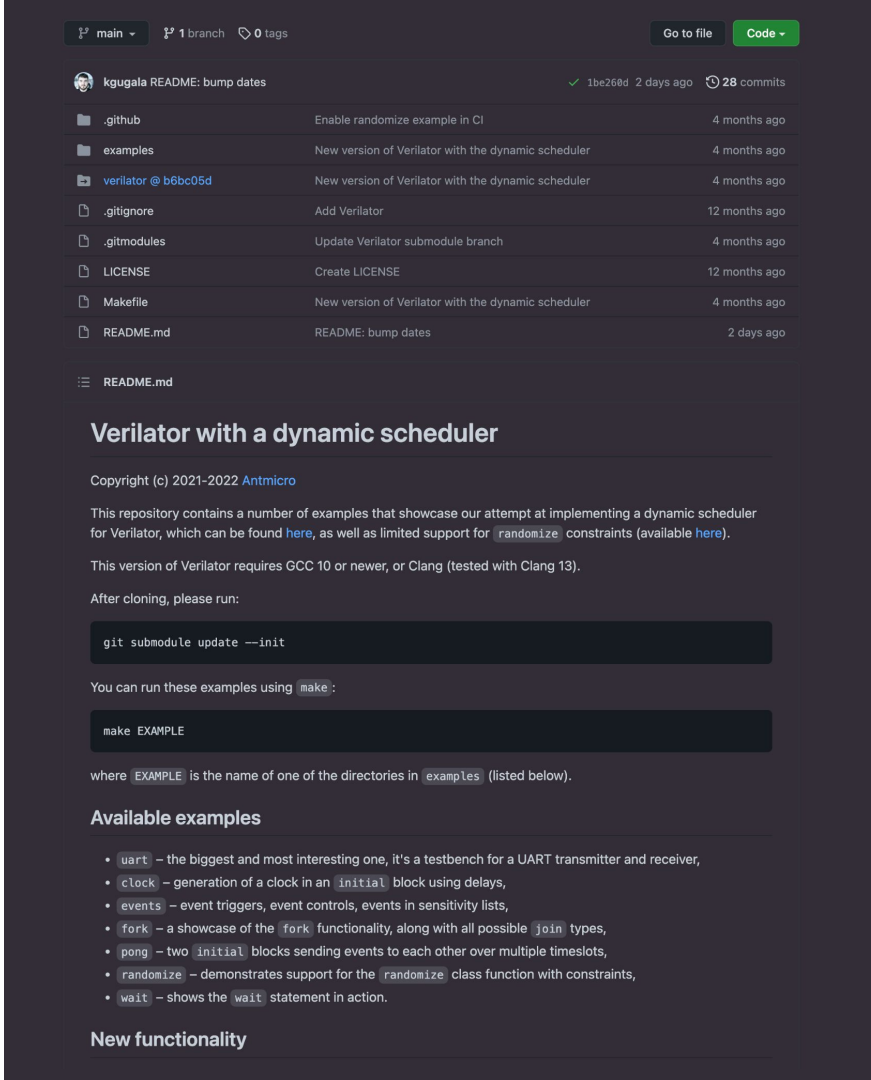
```
module t;
    event ping;
    event pong;

    initial forever begin
        @ping;
        #1;
        ->pong;
    end

    initial forever begin
        #1;
        ->ping;
        @pong;
    end
endmodule
```

DYNAMIC SCHEDULER EXAMPLES

- We've created a public GitHub repository showing example usage of the new features we added to Verilator: github.com/antmicro/verilator-dynamic-scheduler-examples
- The CI there runs simulations of all the examples in the repository



UHDM INTEGRATION

- Surelog is an open source SystemVerilog 2017 Pre-processor, Parser, Elaborator and UHDM Compiler
- Universal Hardware Data Model (UHDM) is used to exchange the information about elaborated SV design between the parser and other tools
- The Surelog->UHDM flow allows fully open source synthesis of SystemVerilog code thanks to the Yosys SystemVerilog plugin and fully open source SystemVerilog verification thanks to a custom Verilator frontend
- Already in late 2020 we could e.g. parse, synthesize and simulate OpenTitan's Ibex core directly from the SystemVerilog source

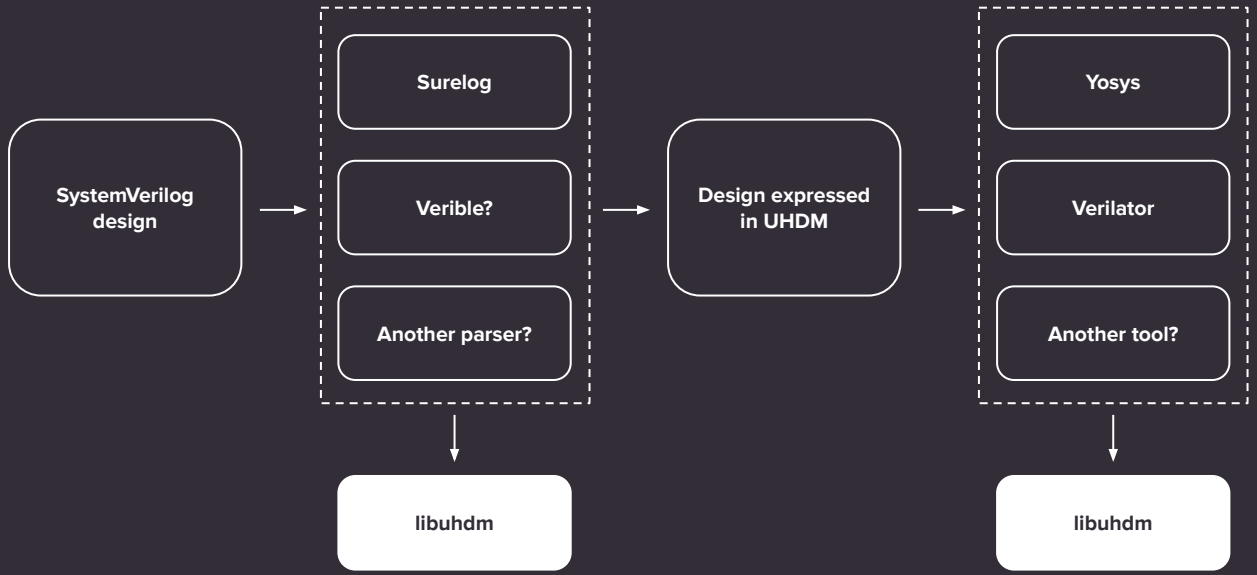
Google

lowRISC

opentitan

antmicro

OPEN SOURCE IBEX SYNTHESIS AND SIMULATION IN VERILATOR/YOSYS VIA UHDM/SURELOG

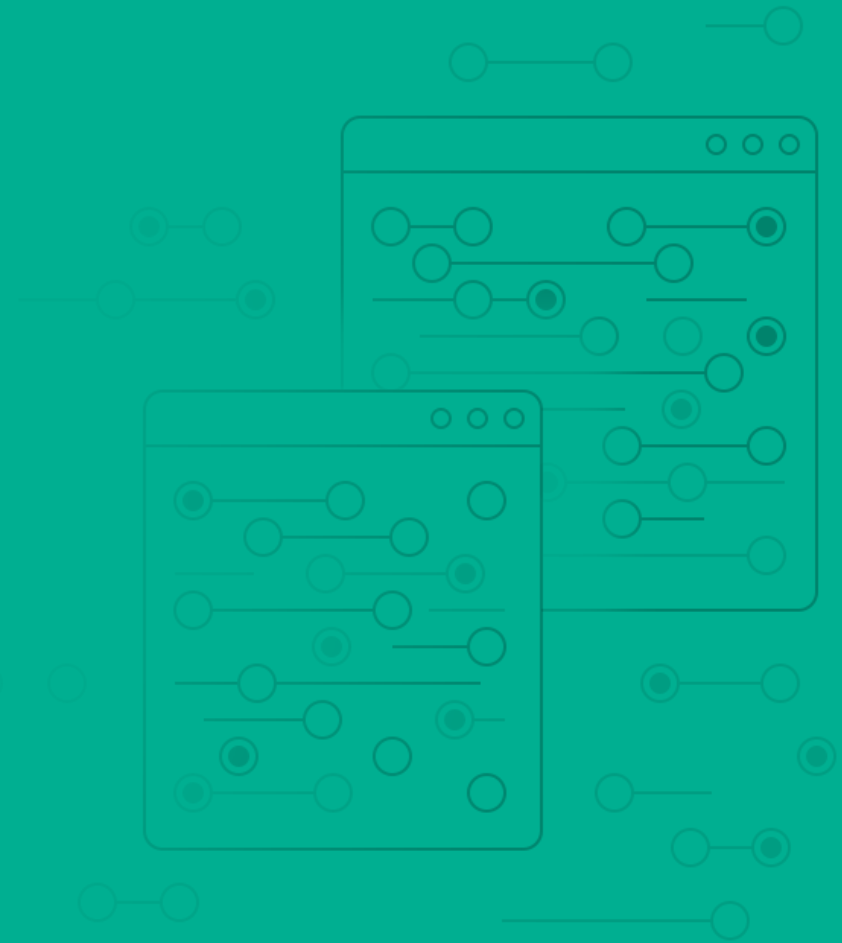


VERILATOR 5

- By introducing the concept of delayed execution to Verilator we have enabled:
  - Event driven simulations
  - Full Verilog/SystemVerilog testbenches (instead of C++ testbenches)
- Stratified scheduling now being integrated into mainline Verilator
  - Coming soon as Verilator 5!
  - Will be released together with major changes regarding the static scheduler (assigning statements to actual stratified scheduler regions and executing those regions in an order specified by the Language Reference Manual)

antmicro

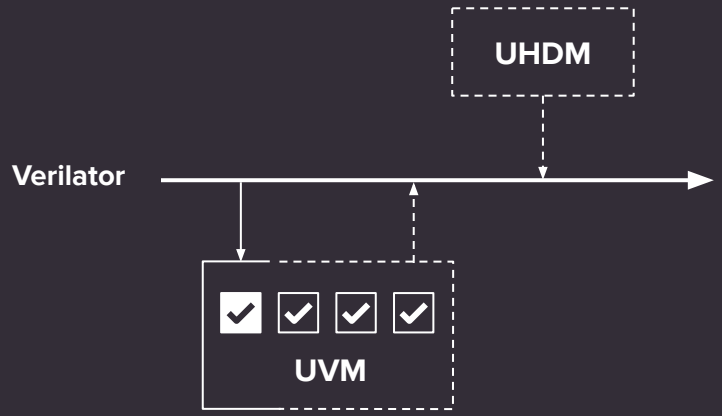
WHAT ELSE NEEDS TO BE DONE?



antmicro

NEXT STEPS

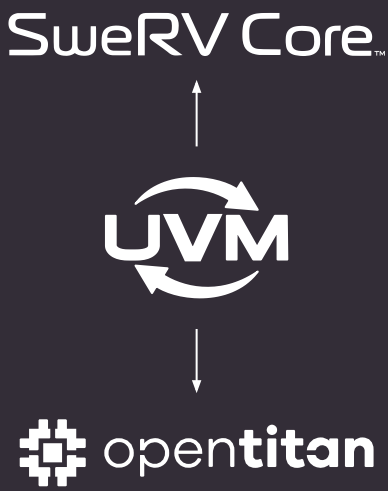
- Extending Verilator with support for more SystemVerilog features used in UVM such as:
  - Clocking blocks
  - Assertions
  - Class parameters
  - Cyclic randomization
  - Unpacked structs
  - Functional coverage
- Runtime optimization
- Integration with UHDM work
- Identifying issues preventing integration with verification test suites used by different projects like e.g. riscv-dv and addressing them



antmicro

FUTURE GOALS

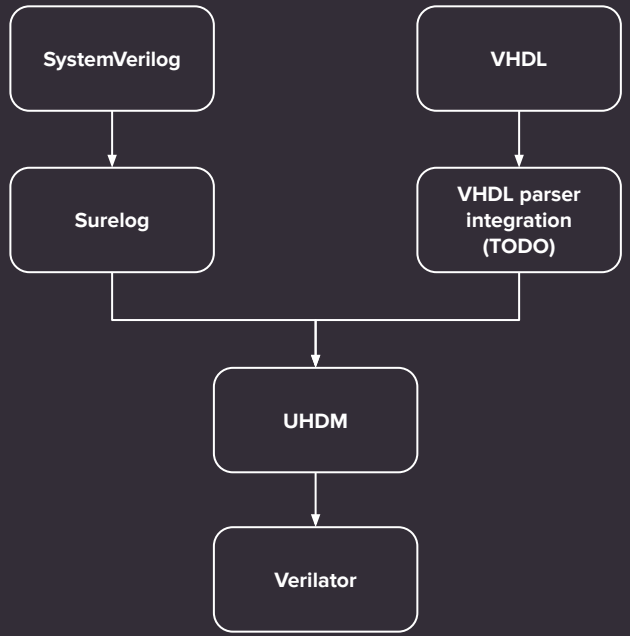
- Long-term goal is to enable full UVM support in Verilator
  - We are now looking at real life production grade designs to gauge what more is required to run their complete validation
  - If you want to get a study of coverage for your own use case, let us know
  - One other obvious focus are open source designs like SweRV or OpenTitan
- An open source flow for verification should lead to more open source verification testbenches and IP being shared!



antmicro

VHDL SUPPORT

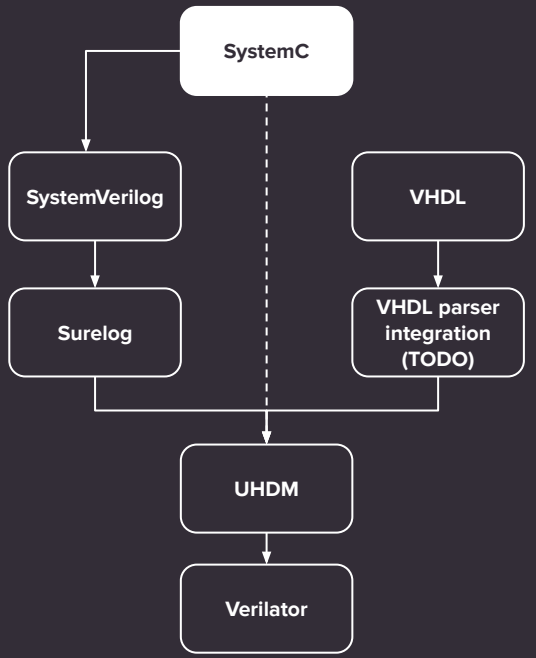
- Evaluating complexity of effort to extend the flow with full VHDL support
  - The plan is to reuse existing UHDM frontends in the tools
  - We would need to extend existing VHDL parser/elaborator with UHDM generation feature
- Considering a few parser/elaborator candidates for first implementation
- Would allow for interesting use cases for event-driven Verilator
- Looking for interested parties!



antmicro

SYSTEMC SUPPORT

- Intel, already a CHIPS member, reached out to contribute their SystemC compiler
- The tool can generate SystemVerilog, meaning that interoperability is possible without significant effort
- We could also go directly to UHDM, which might yield better processing speed/results, but the “easy” approach looks feasible
- This would provide us with pretty broad coverage of “traditional” methods of describing hardware
- Let us know if you are interested in this effort



THANK YOU
FOR YOUR ATTENTION!