Fast Simulation of a Linux-Capable 64-bit RISC-V SoC in Verilator

Max Wipfli

Supervisor: Paul Scheffler

Systems-on-Chip for Data Analytics and Machine Learning,
ETH Zürich, Spring 2025

Project Goal

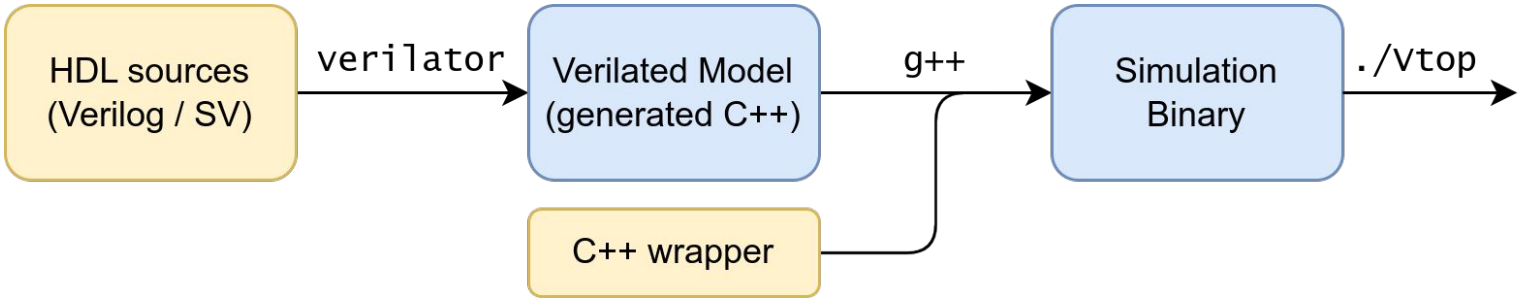
Cheshire: *“A minimal Linux-capable 64-bit RISC-V SoC built around CVA6”*

- **performance-oriented** RTL simulation
- maximize **simulation rate**

→ enable running **complex software** (e.g., Linux boot) in reasonable time

Verilator

- **open-source** Verilog/SystemVerilog simulator
- translates *synthesizable* HDL into **efficient C++ code**
- allows **multi-threaded simulation**



Overview and Methodology

- Initial Setup
- Step-by-Step Optimizations
  - Verilator
  - C++ Compiler
  - RTL
- Multithreading
- Final Results
- Booting Linux

System:

- standard IIS workstation
- AMD Ryzen 9 9900X, 64 GiB RAM

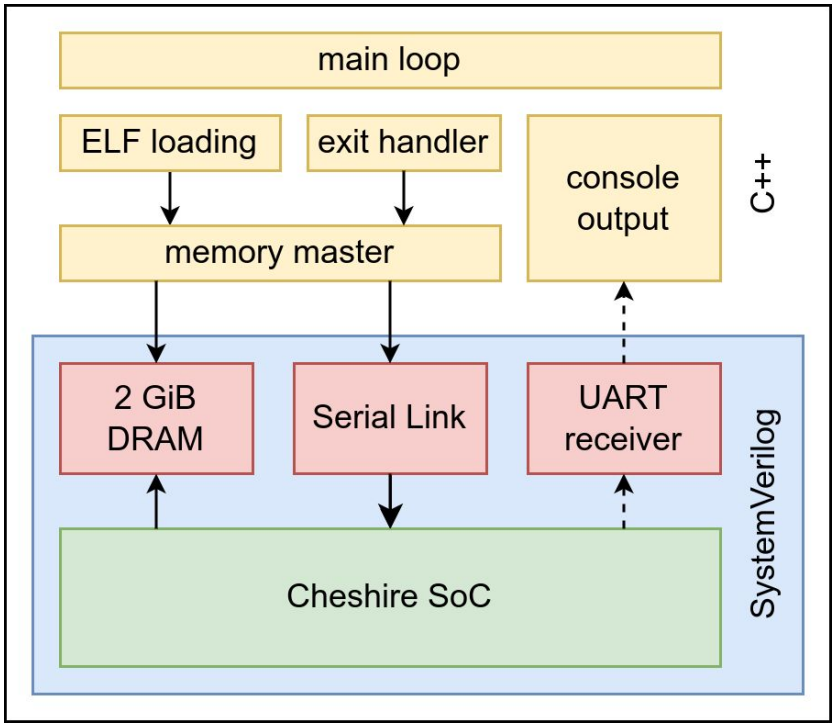
Workloads:

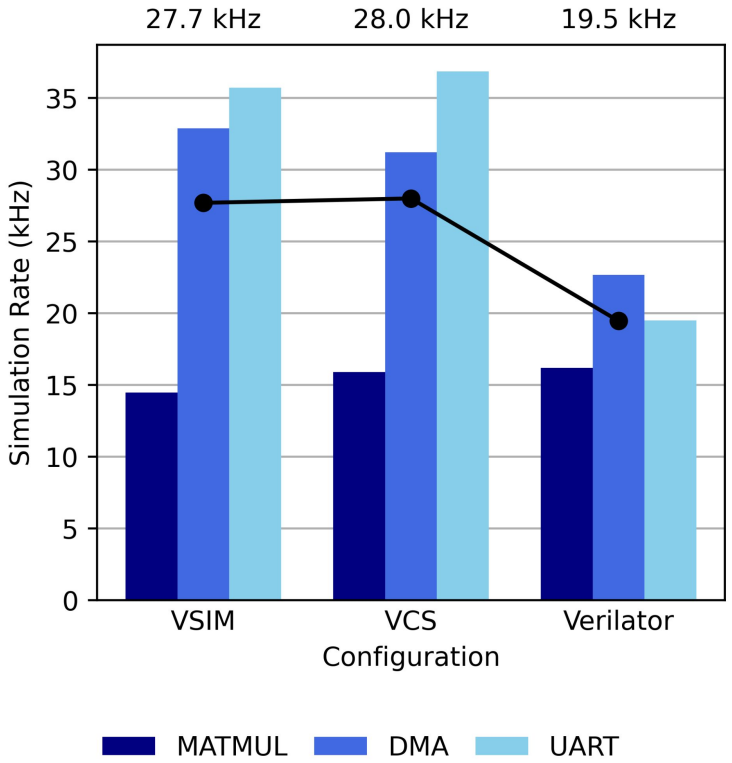
- MATMUL (core-bound)
- DMA (memory-bound)
- UART (I/O-bound)

Measurements:

- average of 5 runs per workload
- < 1.7% relative standard error

Step 1: Initial Setup





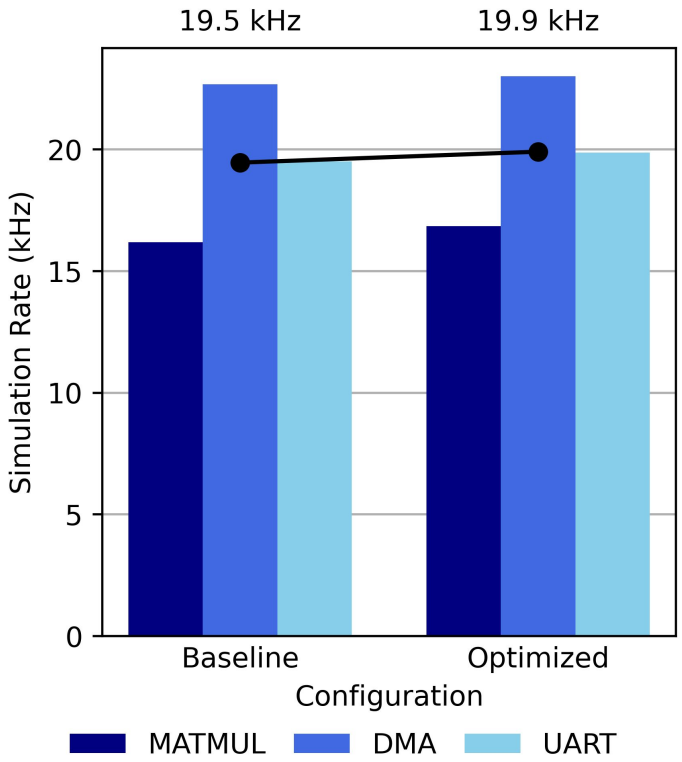
Step 2: Verilator Optimizations

Recommended Optimization Flags

- `-O3`
- `--x-assign fast --x-initial fast`

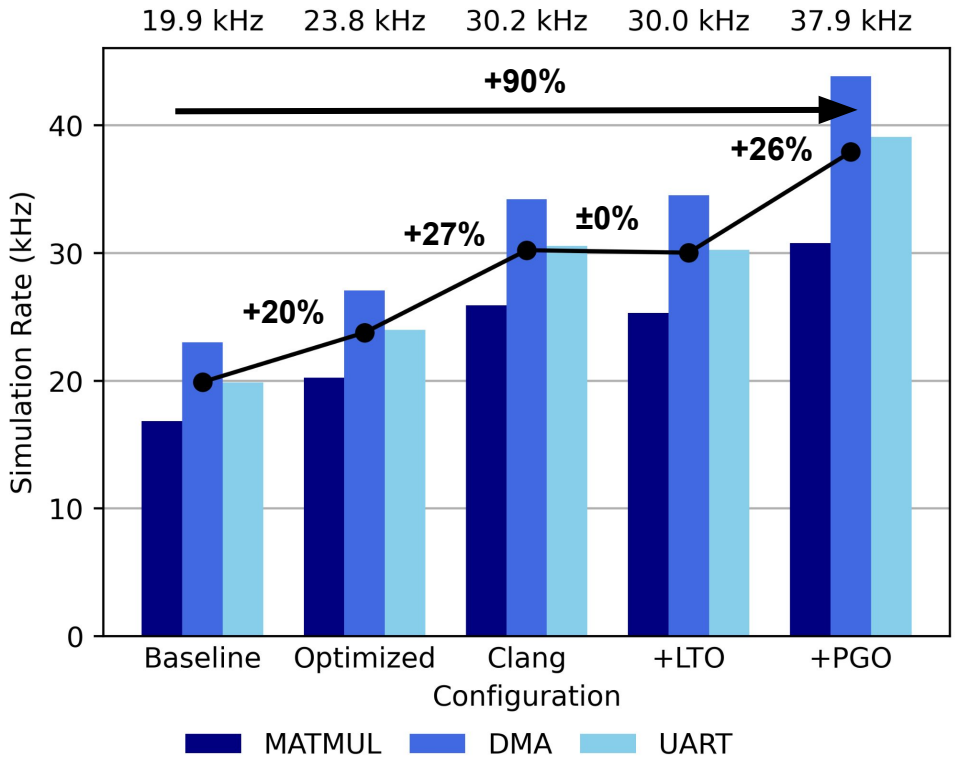
Tunables

- `--flatten`
- `--expand-limit <value>`
- `--inline-mult <value>`
- `--reloop-limit <value>`
- `--unroll-count <loops>`
- `--unroll-stmts <stmts>`



Step 3: C++ Compiler Optimizations

1. **Aggressive Opt. Flags**
   - `-O3` (instead of default `-Os`)
   - `-march=native`
     `-mtune=native`
2. **Use Clang instead of GCC**
3. **Link-Time Opt. (LTO)**
4. **Profile-Guided Opt. (PGO)**



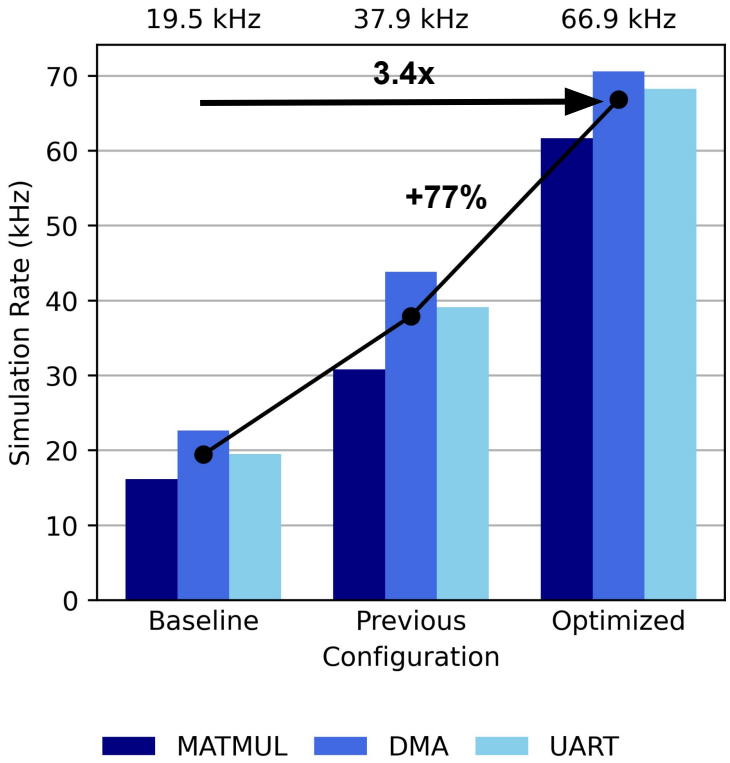
Step 4: RTL Optimizations

Profiling Simulation Binary

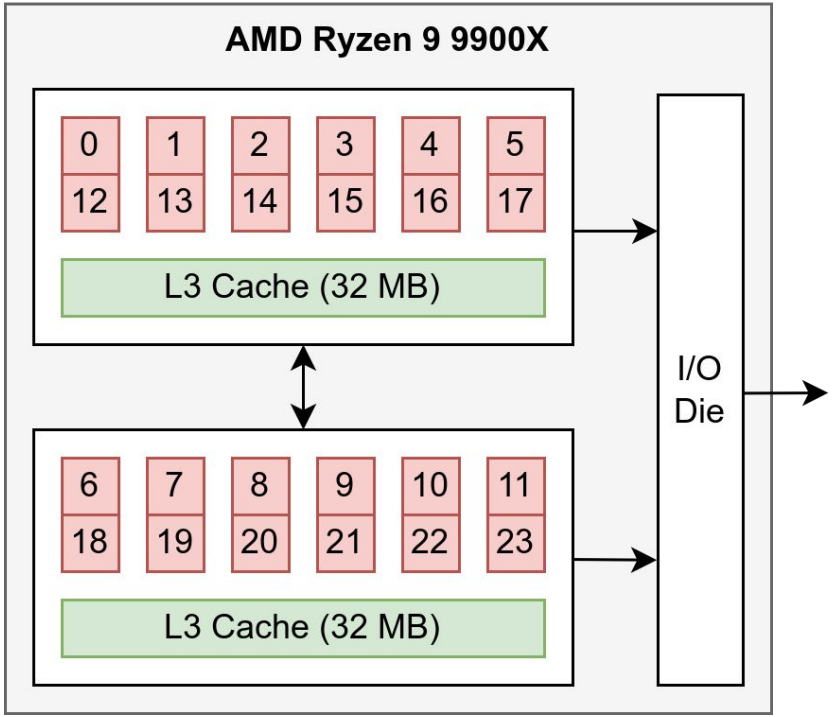
1. Compile and profile using `gprof`
2. Analyze profile to identify slow RTL constructs
3. Rewrite with more efficient equivalent code; or apply Verilator directives

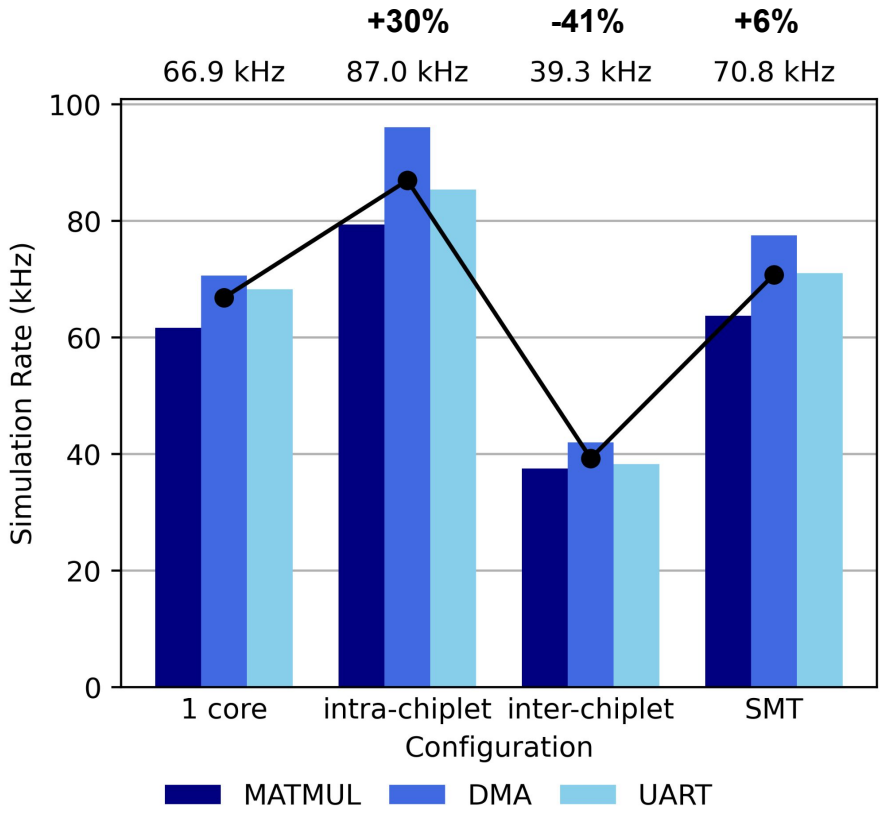
```
for (int i = 0; i < (2**HashWidth); i++) begin
  for (int j = 0; j < NoHashes; j++) begin
    indicator_o[i] = indicator_o[i] | hashes[j][i];
  end
end
for (int j = 0; j < NoHashes; j++) begin
  indicator_o = indicator_o | hashes[j];
end
```

pulp-platform/common_cells
pulp-platform/cva6
verilator/verilator



Step 5: Multi-Threading





Step 5: Multi-Threading

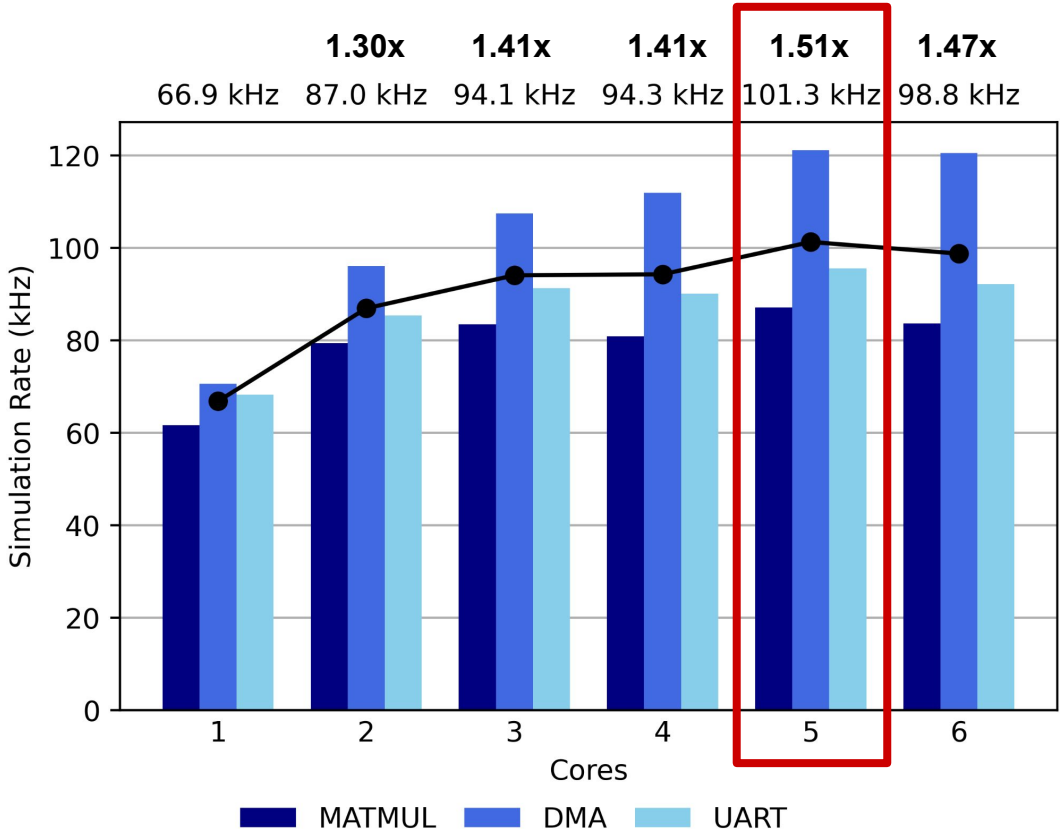
Optimize Thread Count

- pinned threads
- single chiplet
- no SMT

→ up to 6 threads

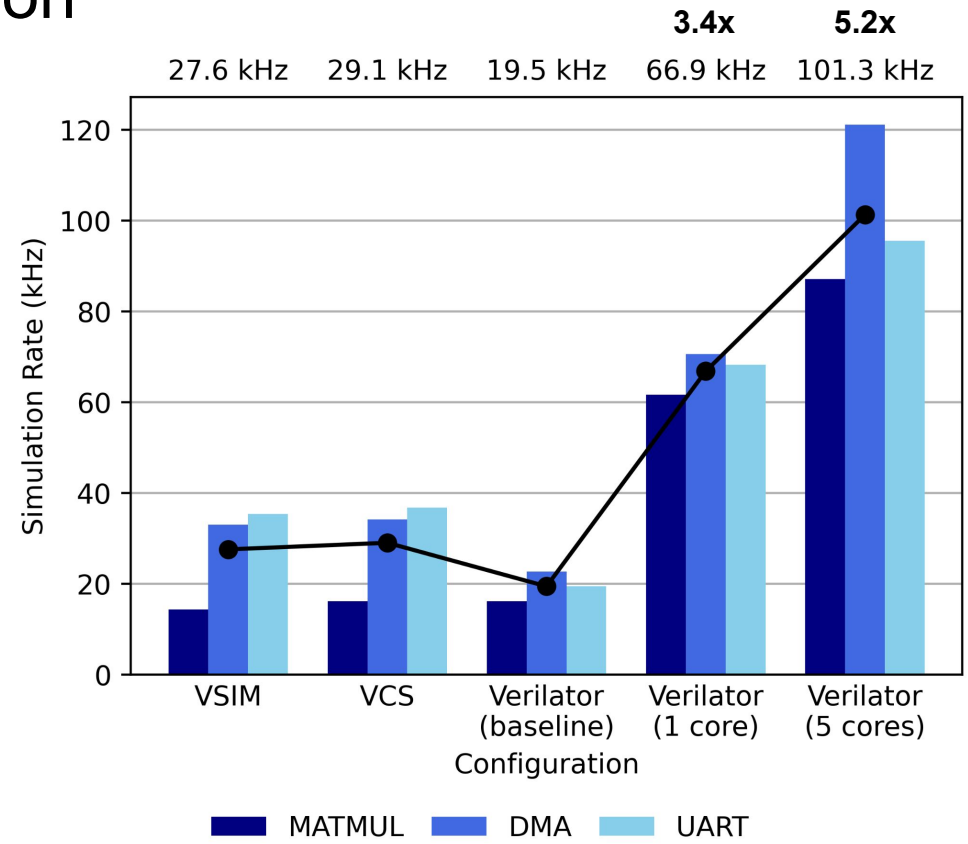
Verilator *Thread PGO*

- optimized work distribution across threads
- no improvement



Final Result and Comparison

- significant improvement over commercial simulators
- different workloads
- between **2.6x and 6.1x faster** than VCS



Linux Boot Attempt

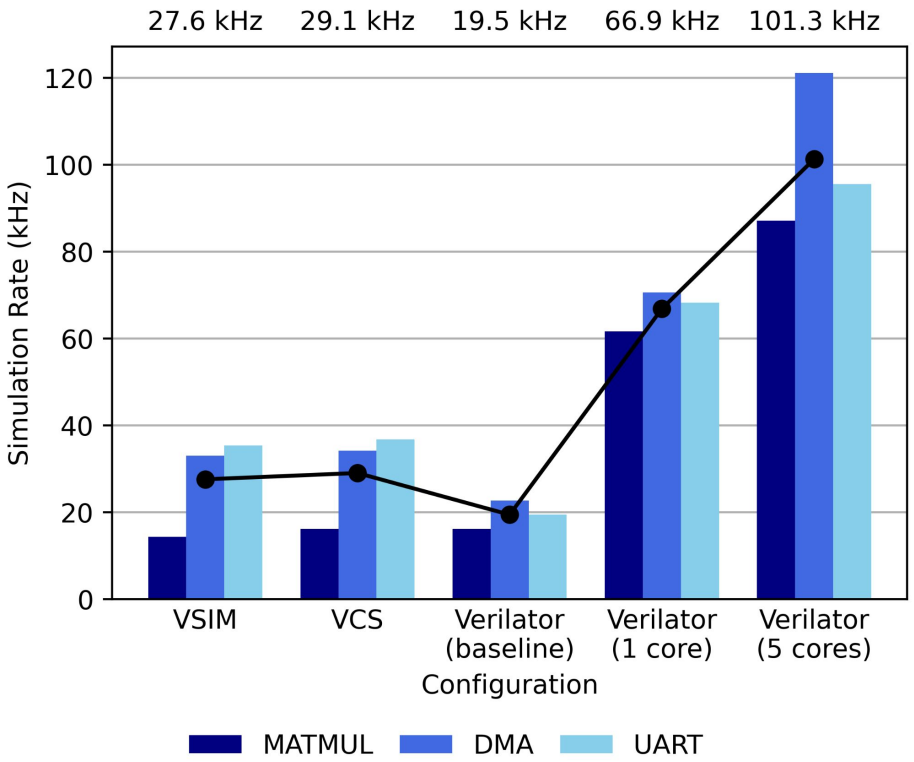
```
 /\___/\       Boot mode:       0
( o   o )      Real-time clock: 32768 Hz
(  =^=  )      System clock:    200078887 Hz
(        )     Read global ptr: 0x00000000
(    P    )    Read pointer:    0x10000000
(  U # L   )   Read argument:   0x00000000
(    P      )
(           ))))))))))

[ZSL] Launch firmware at 80000000 with device tree at 90000000

OpenSBI v0.9
   ____                    _____ ____ _____
  / __ \                  / ____|  _ \_   _|
 | |  | |_ __   ___ _ __ | (___ | |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | |_) |  __/ | | |____) | |_) || |_
  \____/| .__/ \___|_| |_|_____/|____/_____|
        | |
        |_|

[    0.000000] Linux version 5.10.7 (sem25f4@badile39.ee.ethz.ch) (riscv64-buildroot-linux-gnu-gcc
(Buildroot 2021.08) 10.3.0, GNU ld (GNU Binutils) 2.36.1) #18 SMP Tue Jul 22 16:10:32 CEST 2025
[    0.000000] OF: fdt: Ignoring memory range 0x80000000 - 0x80200000
[    0.000000] earlycon: ns16550a0 at MMIO32 0x0000000003002000 (options '961200')
[    0.000000] printk: bootconsole [ns16550a0] enabled
[    0.000000] efi: UEFI not found.
[    0.000000] Zone ranges:
[    0.000000]   DMA32    [mem 0x0000000080200000-0x00000000bfffffff]
[    0.000000]   Normal   empty
[    0.000000] Movable zone start for each node
[    0.000000] Early memory node ranges
[    0.000000]   node   0: [mem 0x0000000080200000-0x00000000bfffffff]
[    0.000000] Initmem setup node 0 [mem 0x0000000080200000-0x00000000bfffffff]
```

Crash due to Verilator misbehavior:

- after approx. 3 hours
- investigated with waveforms
- assignment to SV packed struct or array fields are sometimes skipped
- open GitHub issue: `verilator/verilator#5350`

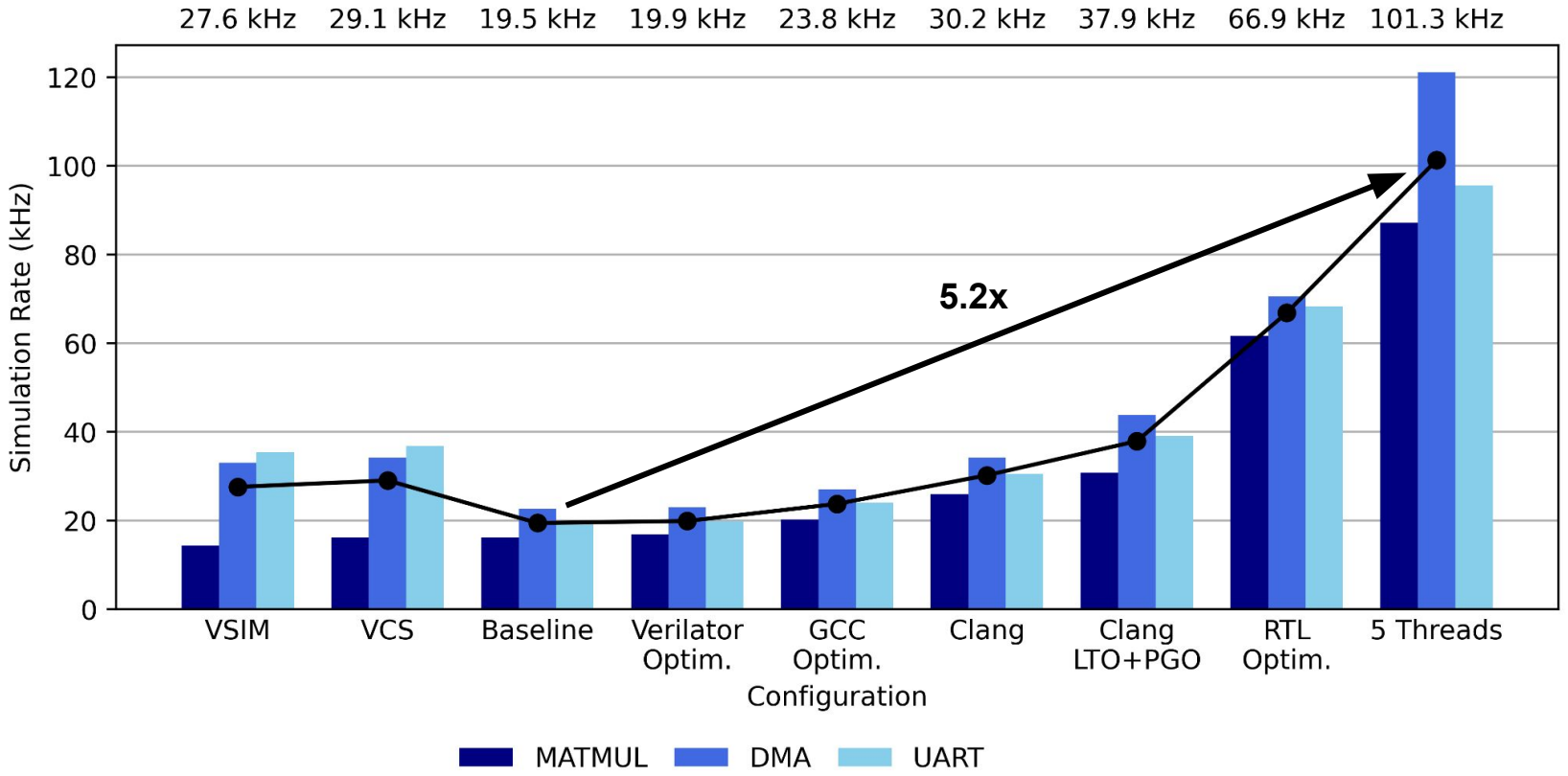
Conclusion

- performance-focused RTL simulation of Cheshire SoC
- open-source Verilator simulator
- optimizations and multi-threading
- upstream contributions
- between **2.6x and 6.1x faster** than state-of-the-art commercial simulators

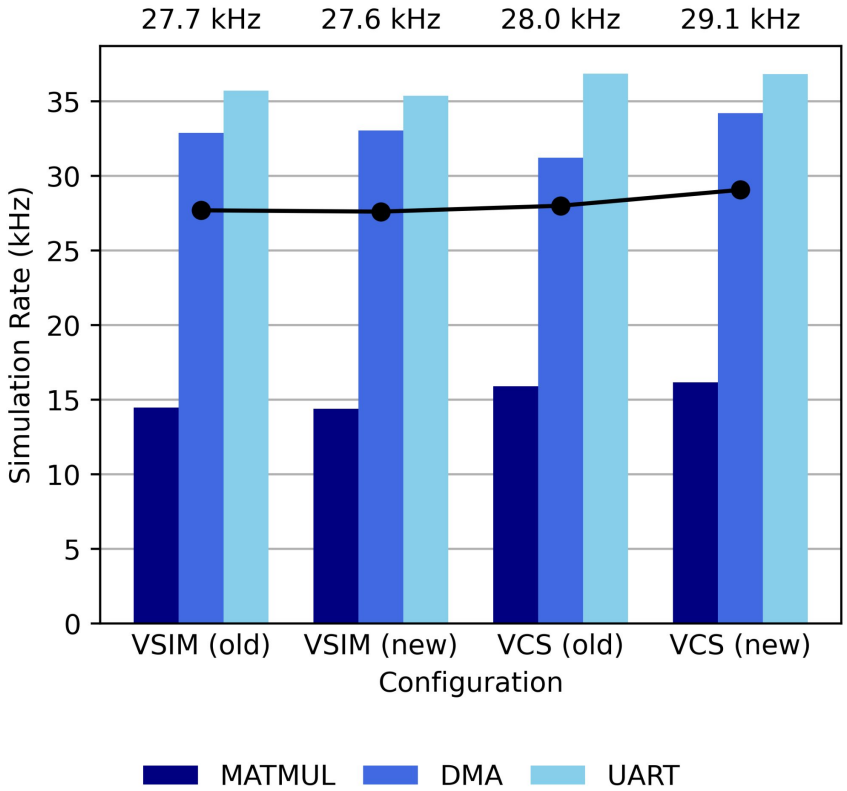


Additional Material

Step-by-Step Optimizations



VCS/VSIM Performance with RTL Changes



Open Source Contributions

pulp-platform/cheshire

- #230: Add Verilator fast simulation

pulp-platform/common_cells

- #258: id_queue: Simplify existence lookup for Verilator speed-up
- #259: treewide: Optimizations for faster Verilator simulation

pulp-platform/cva6

- #66: RTL optimizations for Verilator *(open)*

verilator/verilator

- #6203: Optimize 2 ** X to 1 << X even if base is signed

More RTL Optimization Examples

Leading Zero Counter (`lzc`)

```
always_comb begin : flip_vector
  for (int unsigned i = 0; i < WIDTH; i++) begin
    in_tmp[i] =
        (MODE) ? in_i[WIDTH-1-i] : in_i[i];
  end
end

if (MODE) begin : g_flip
  // Mode 1 (leading zero): flip input vector
  always_comb begin : flip_vector
    for (int unsigned i = 0; i < WIDTH; i++) begin
      in_tmp[i] = in_i[WIDTH-1-i];
    end
  end
end else begin
  // Mode 0 (trailing zero)
  assign in_tmp = in_i;
end
```

Round-Robin Arbitration Tree (`rr_arb_tree`)

```
idx_t    [2**NumLevels-2:0] index_nodes
    /* verilator split_var */;
DataType [2**NumLevels-2:0] data_nodes
    /* verilator split_var */;
logic    [2**NumLevels-2:0] gnt_nodes
    /* verilator split_var */;
logic    [2**NumLevels-2:0] req_nodes
    /* verilator split_var */;
```

Potential Further Improvements

- fix remaining **UNOPTFLAT warnings** (improper combinational loops)
- attempt to identify **“slow” components**
  - selectively disable optional peripherals (JTAG, VGA, USB, …)
- investigate **dependence on CPU** used for simulation

Linux Boot: Verilator Misbehavior

Exact Issue

- debugged using waveforms
- combinational assignment may be skipped

```
// cva.sv:1115
assign dcache_req_ports_cache_ex[1] = dcache_req_from_cache[1];
```

- highly dependent on exact RTL (butterfly effect)

Potential Solution

- annotate all packed structs/arrays assigned from multiple places with `/*verilator split_var*/`
- automation required

Linux Boot: Verilator Misbehavior

Assignments to structure elements from multiple locations can lead to assignment being skipped #5350

Open

matthew-humphrey opened on Aug 8, 2024

Verilator version: Verilator 5.027 devel rev v5.026-110-g18fc3e608
OS: Ubuntu 22.04 LTS
We are open to solving the issue ourself with some help.

When the RTL has an instance of a packed struct whose elements are assigned in multiple places, for example in multiple procedural blocks or procedural blocks and output ports of a module, the assignment of a structure member can be skipped or incorrect.

In the attached regression test, a module outputs both one-hot and binary versions of a value. Internal to the module, a packed struct was added to hold the values that drive the two output ports. One of the structure elements is assigned by an `assign` statement, and the other is assigned via an `always_comb` block. It appears that the conditional assignment in the always_comb block is skipped.

Assignees
No one assigned

Labels
status: ready

Type
No type

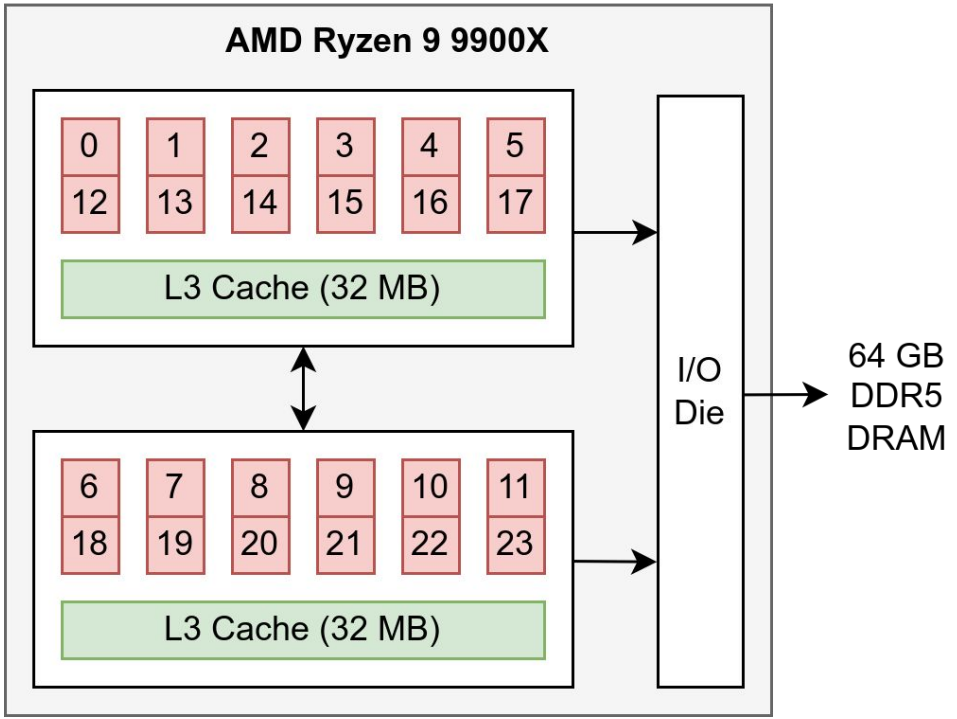
Projects
No projects

Milestone

Test System

`badile39`:

- new standard IIS workstation
- AMD Ryzen 9 9900X
  - 2 core complex dies + I/O die
  - 12 cores, SMT
  - 64 MB split L3 cache
- 64 GB DDR5 DRAM
- uniform memory access (UMA)



Test System

```
$ lscpu -e
CPU NODE SOCKET CORE L1d:L1i:L2:L3 ONLINE MAXMHZ    MINMHZ
0   0    0      0    0:0:0:0       yes    7025.0000 600.0000
1   0    0      1    1:1:1:0       yes    7192.0000 600.0000
2   0    0      2    2:2:2:0       yes    7364.0000 600.0000
3   0    0      3    3:3:3:0       yes    7364.0000 600.0000
4   0    0      4    4:4:4:0       yes    6681.0000 600.0000
5   0    0      5    5:5:5:0       yes    6853.0000 600.0000
6   0    0      6    6:6:6:1       yes    6002.0000 600.0000
7   0    0      7    7:7:7:1       yes    6514.0000 600.0000
8   0    0      8    8:8:8:1       yes    6170.0000 600.0000
9   0    0      9    9:9:9:1       yes    6342.0000 600.0000
10  0    0      10   10:10:10:1    yes    5830.0000 600.0000
11  0    0      11   11:11:11:1    yes    5658.0000 600.0000
12  0    0      0    0:0:0:0       yes    7025.0000 600.0000
13  0    0      1    1:1:1:0       yes    7192.0000 600.0000
14  0    0      2    2:2:2:0       yes    7364.0000 600.0000
15  0    0      3    3:3:3:0       yes    7364.0000 600.0000
16  0    0      4    4:4:4:0       yes    6681.0000 600.0000
17  0    0      5    5:5:5:0       yes    6853.0000 600.0000
18  0    0      6    6:6:6:1       yes    6002.0000 600.0000
19  0    0      7    7:7:7:1       yes    6514.0000 600.0000
20  0    0      8    8:8:8:1       yes    6170.0000 600.0000
21  0    0      9    9:9:9:1       yes    6342.0000 600.0000
22  0    0      10   10:10:10:1    yes    5830.0000 600.0000
23  0    0      11   11:11:11:1    yes    5658.0000 600.0000
```

Benchmark Build and Simulation Times

- building
  - running Verilator and C++ compilation
  - around 50 seconds (using 24 threads)

- simulating
  - UART: 7-35 seconds
  - DMA: 10-58 seconds
  - MATMUL: 21-115 seconds

Software Versions

- AlmaLinux 8.10
  - Linux Kernel 4.18.0
- VCS 2025.06
- QuestaSim (VSIM) 2025.1
- IIC-OSIC-TOOLS (oseda) 2025.03
  - Verilator 5.034
  - GCC 13.3.0
  - Clang 16.0.6