Overview: Accellera SystemC Verification Working Group activities

Thilo Vörtler, COSEDA Technologies GmbH

Accellera SystemC VWG Chair

accellera SYSTEMS INITIATIVE™

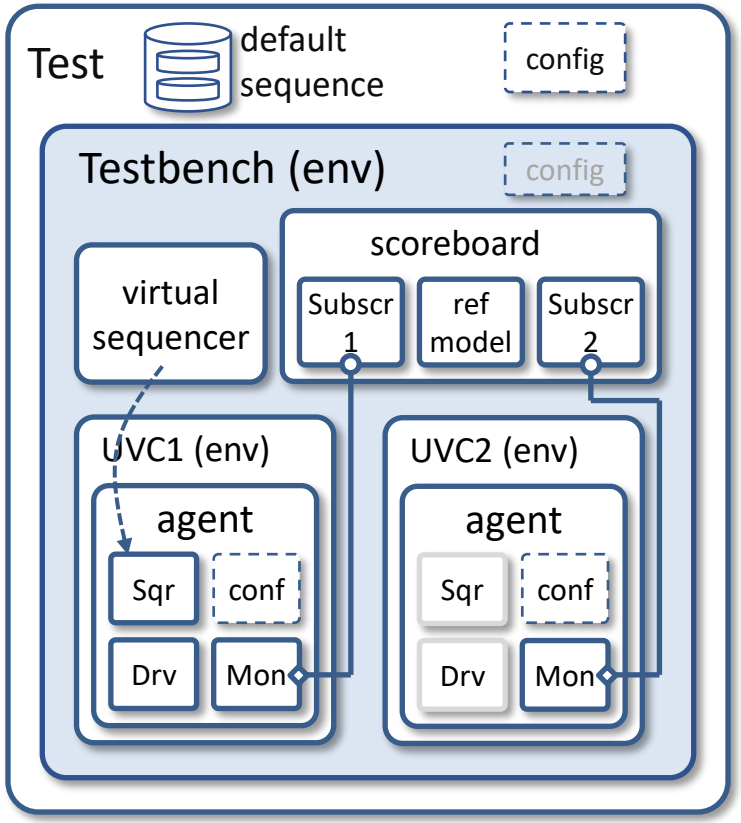
SYSTEMC™ EVOLUTION FIKA

Copyright Permission

- A non-exclusive, irrevocable, royalty-free copyright permission is granted by **Thilo Vörtler, COSEDA Technologies** to use this material in developing all future revisions and editions of the resulting draft and approved Accellera Systems Initiative **SystemC** standard, and in derivative works based on the standard.

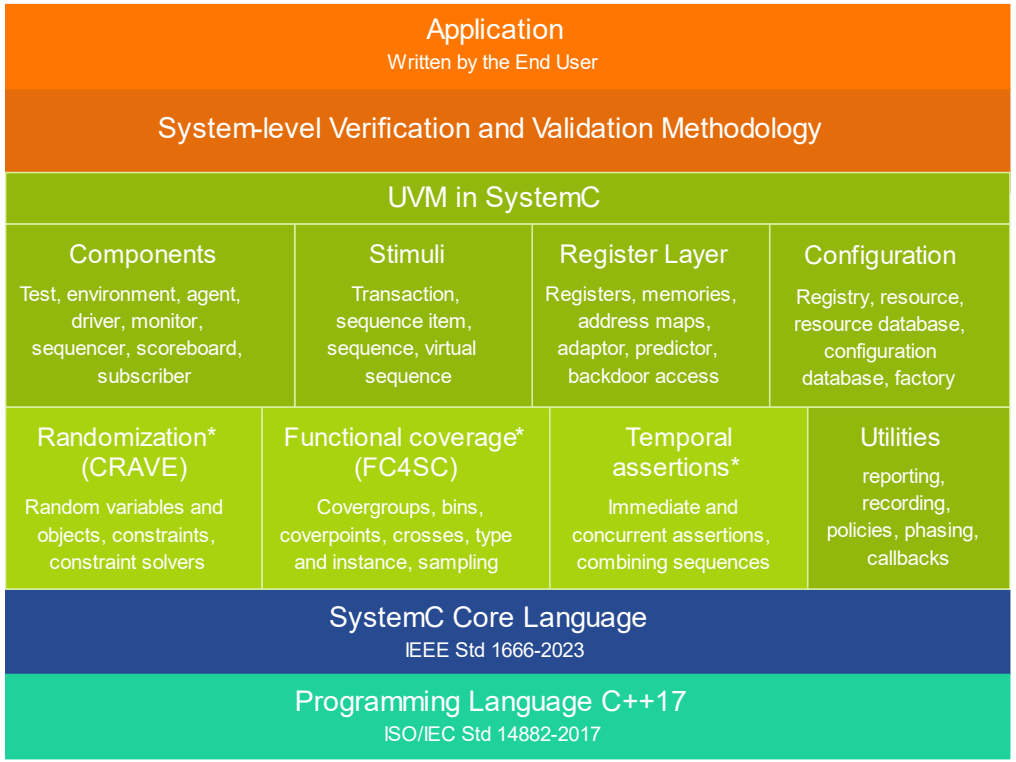
Introduction

- The VWG is responsible for defining verification extensions to the SystemC language standard
- Main focus is the development of UVM in SystemC
- VWG provides proof of concept implementation, creates the UVM SystemC standard Language Reference Manual
- Support for Constraint Randomization and functional Coverage by donated Libraries:
  - CRAVE - **C**onstrained **Ra**ndom **V**erification **E**nvironment
  - FC4SC - **F**unctional **C**overage **for** **S**ystem**C**



UVM in SystemC

- Methodology to create **modular, scalable, configurable and reusable System Level** testbenches
- Follows the UVM-SystemVerilog Standardized API
  - Similar class definitions, methods and other definitions in the LRM
  - Divergence where SystemC and C++ already offer solutions
  - Stricter with regards to non-LRM API
- Complies with SystemC IEEE 1666-2023 standard and SystemC 3.0.x reference implementation
  - Follow SystemC-defined TLM1 and TLM2 communication mechanism
  - SystemC modules capture testbench hierarchy, test sequences as transient objects



* Integration on Roadmap

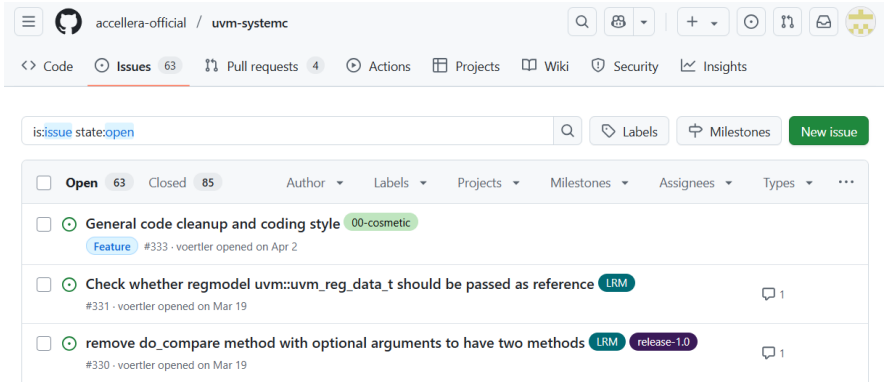
https://systemc.org/overview/systemc-verification/

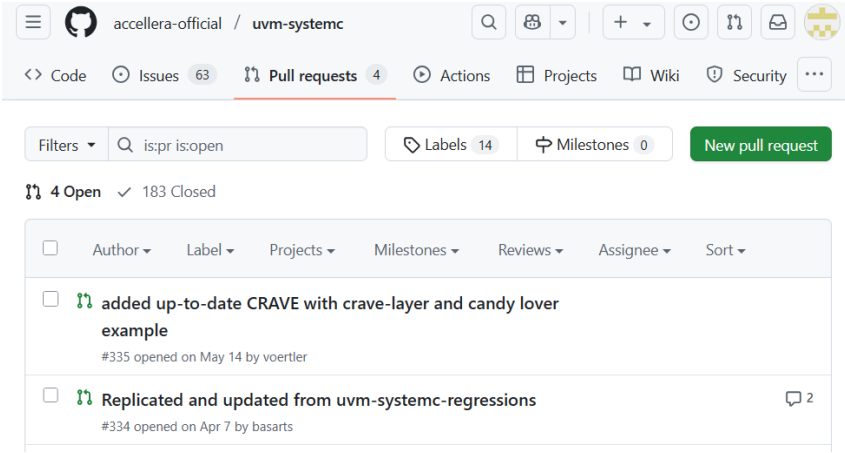
Current Activities of the WG

- Made UVM SystemC reference implementation available on GitHub since May 2025
  - https://github.com/accellera-official/uvm-systemc
  - Includes regression tests (based on UVM SystemVerilog tests)
- Creation of the Language Reference Manual for UVM SystemC 1.0
  - Goal: release the LRM by end of 2025
- Update reference implementation to be in sync with 1.0 release
- Switch compile flow to CMake
- Integrate github actions for CI flow
- Work on issues and feedback provided by users

Call for Participation

- How can you help the working group?
- Download and use Libraries
  - https://github.com/accellera-official/uvm-systemc
  - https://github.com/accellera-official/crave
  - https://github.com/accellera-official/fc4sc
- Provide Feedback using GitHub
  - Feature requests, bugs, usability issues, API problems
- Contribute Code through pull requests (Apache 2.0 license)
- Join the working group (Accellera Members)
  - Meetings twice a month on Wednesday





A SystemC-UVM testbench for a student lab exercise

Jens Schönherr, HTW Dresden

Thilo Vörtler, COSEDA Technologies GmbH

accellera SYSTEMS INITIATIVE™

HTW D

SYSTEMC™ EVOLUTION FIKA

Copyright Permission

- A non-exclusive, irrevocable, royalty-free copyright permission is granted by **Jens Schönherr, HTW Dresden** to use this material in developing all future revisions and editions of the resulting draft and approved Accellera Systems Initiative **SystemC** standard, and in derivative works based on the standard.

Outline

- Introduction – Motivation
- Structure of testbench
- Experiences
- Future work

Boundary conditions

- lecture “Test and Verification” (6th semester bachelor)
  - simulation techniques/testbenches/formal verification/IC test
  - with some lab exercises and a computer project
- lab exercise “Directed and constraint random simulation” (3 h)
  - directed test with VHDL
  - directed test with UVM (SystemC)
  - constraint random test with UVM (SystemC)
  - creation of UVM testbench takes too long
  - testbench given, students create directed test and parameterize constraint random test to achieve coverage goal

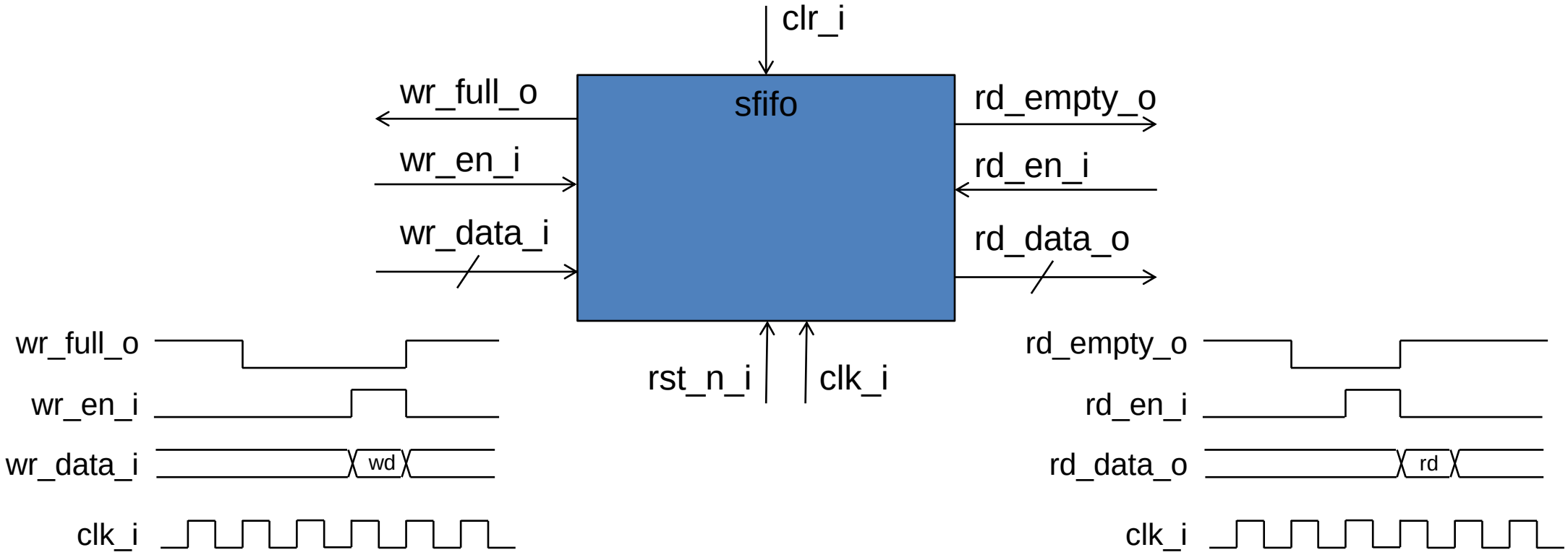
Motivation

- Why verification in lectures?
  - industry need
- Why SystemC-UVM?
  - UVM is industry standard (SystemVerilog/e/SystemC)
  - students: C/C++, Matlab, Python, VHDL, … and SystemVerilog or e?
  - SystemC well documented (online, books)
  - source code available
  - tool costs

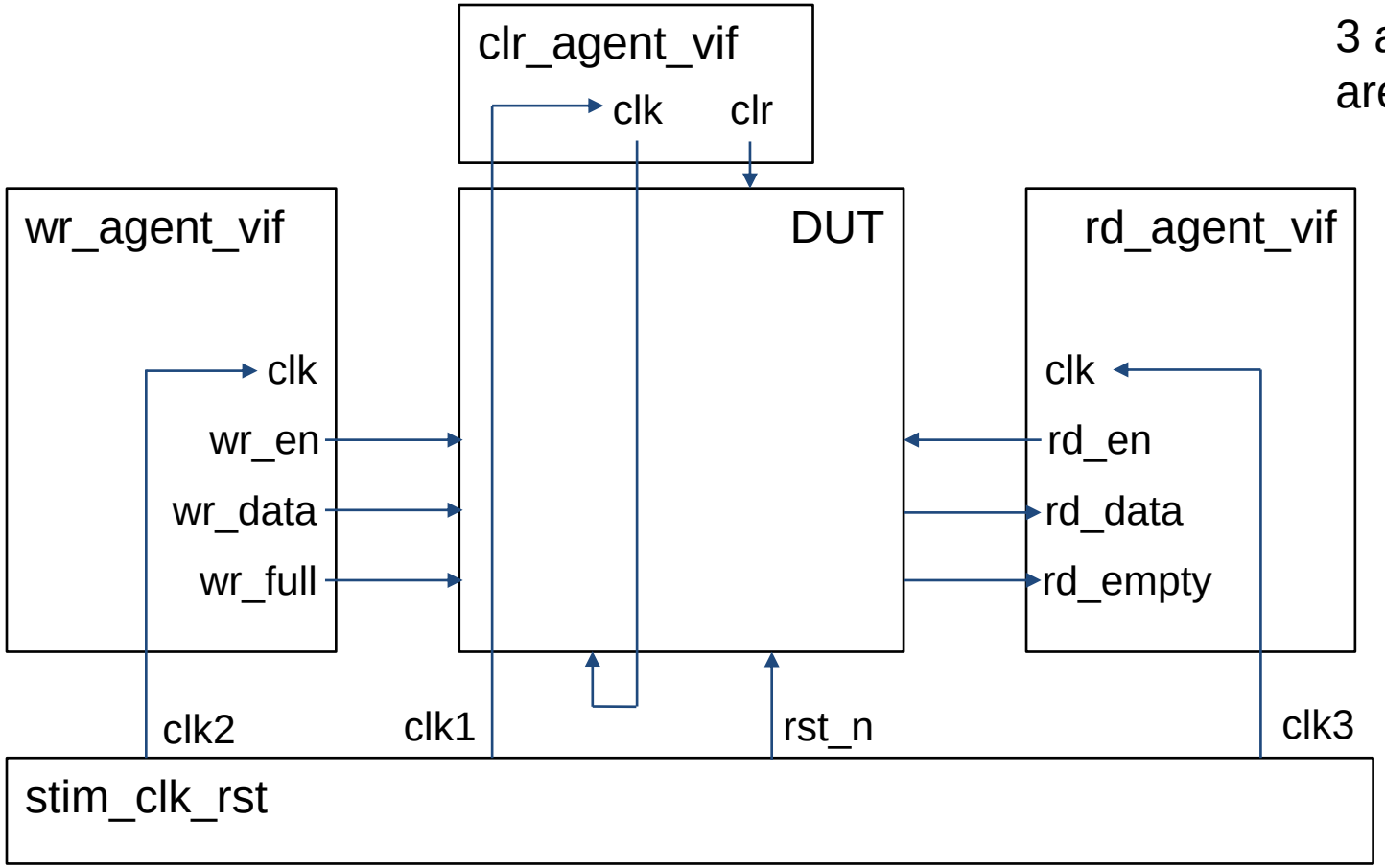
accellera SYSTEMS INITIATIVE™

SYSTEMC™ EVOLUTION FIKA

DUT – synchronous FIFO



sc_main()

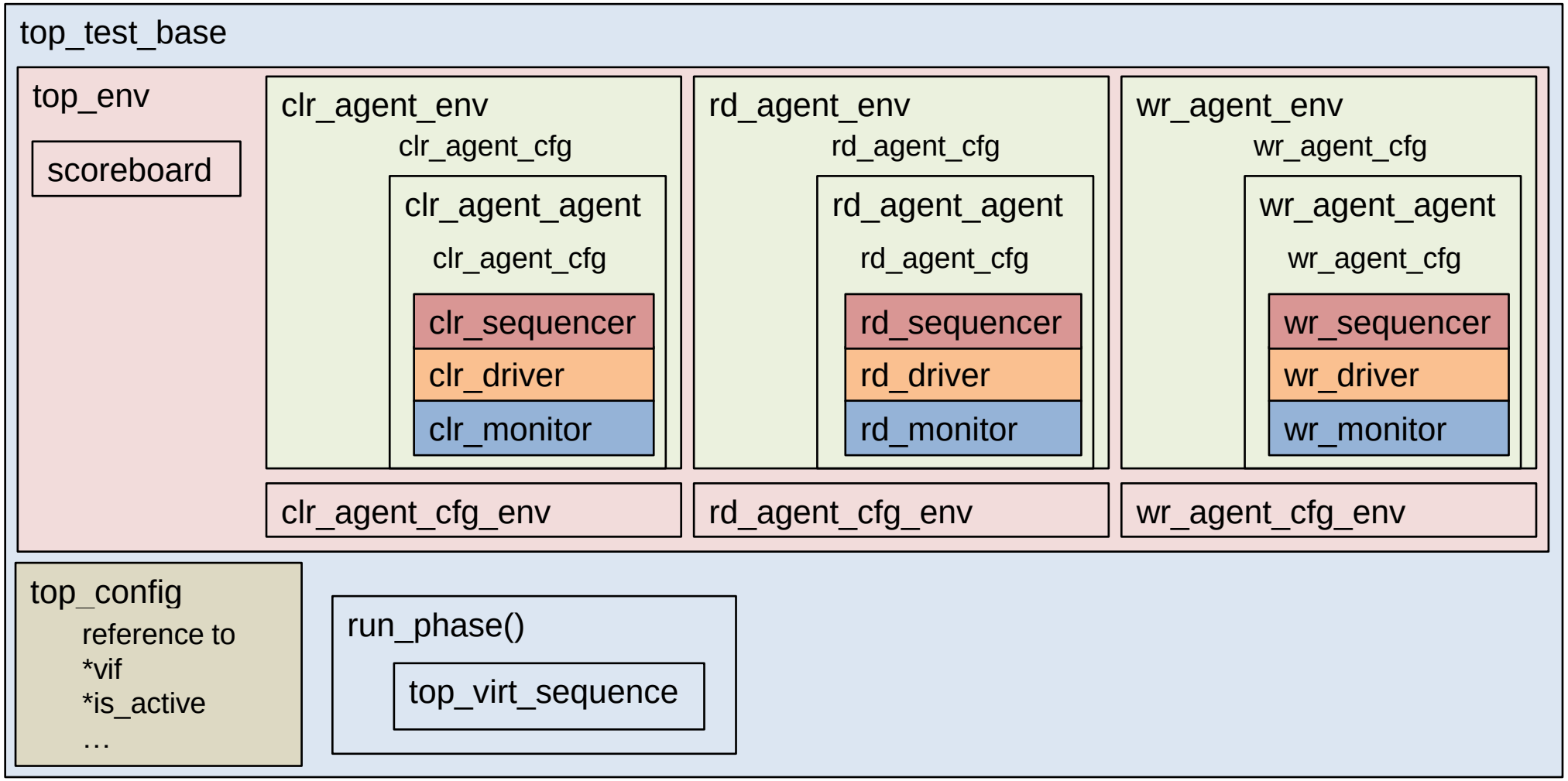


3 agents because random inputs are generated independently

signals in vif (virtual interfaces)

generates reset and synchronous clocks

Tests

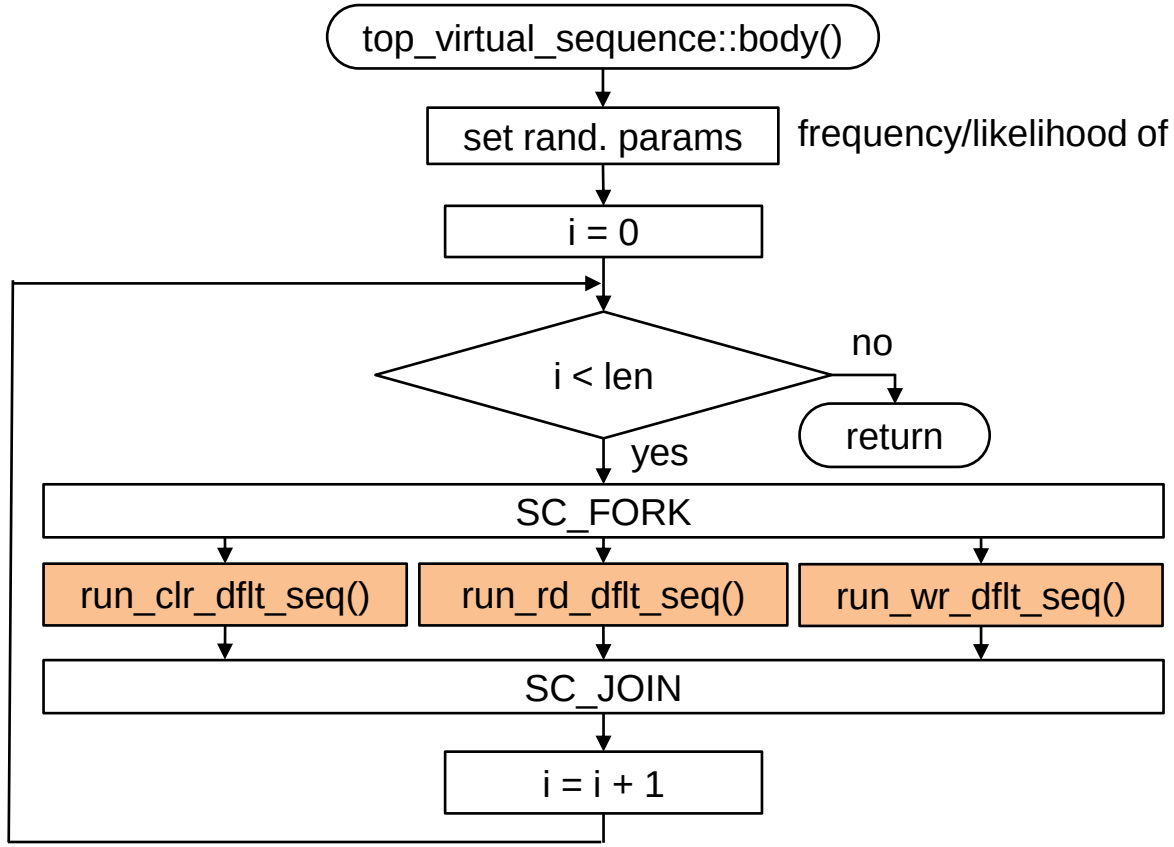


top_test_directed with same structure

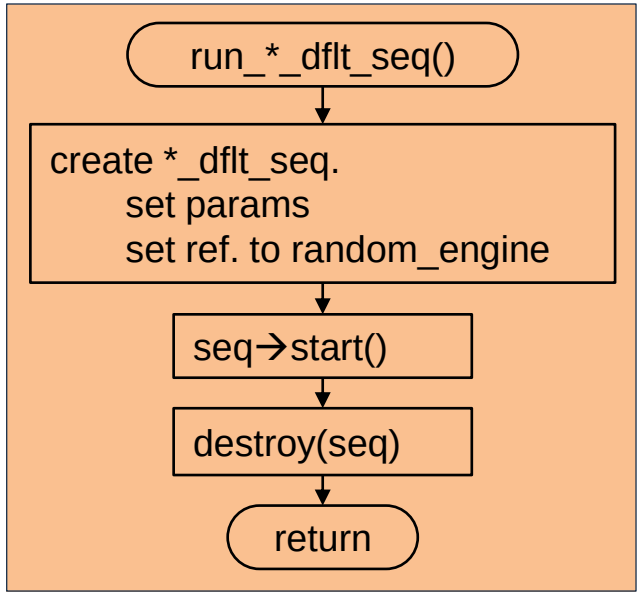
Random sequences

top_virt_sequence

clr_agent_env*
rd_agent_env*
wr_agent_env*
random_engine



random parameters are copied:
member of top_virt_sequence
→ sequence

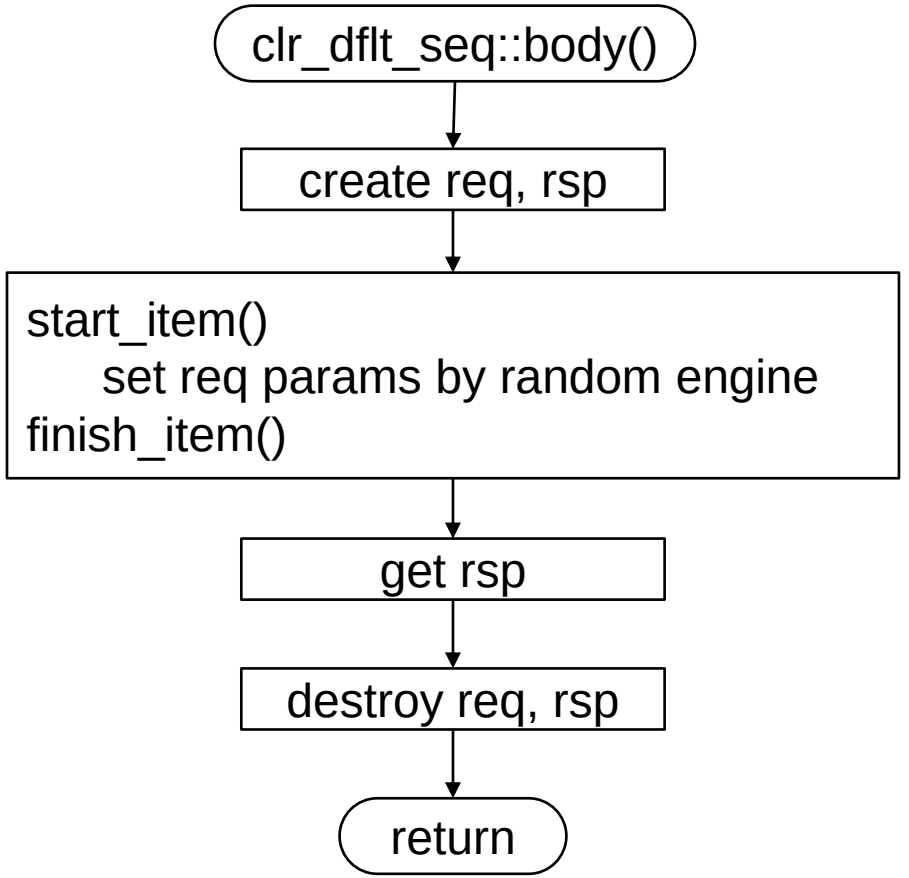


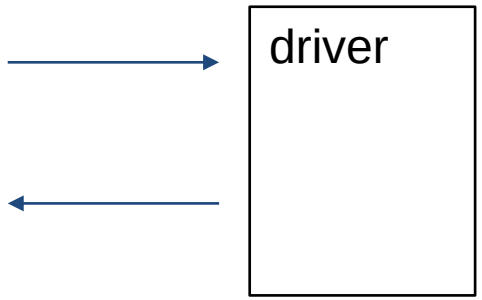

```
SC_FORK
  sc_spawn(sc_bind(&top_virt_sequence::run_clr_dflt_seq, this)),
  sc_spawn(sc_bind(&top_virt_sequence::run_wr_dflt_seq,  this)),
  sc_spawn(sc_bind(&top_virt_sequence::run_rd_dflt_seq,  this)),
SC_JOIN
```

Creating random transaction

clr_dflt_seq

params for random
random_engine

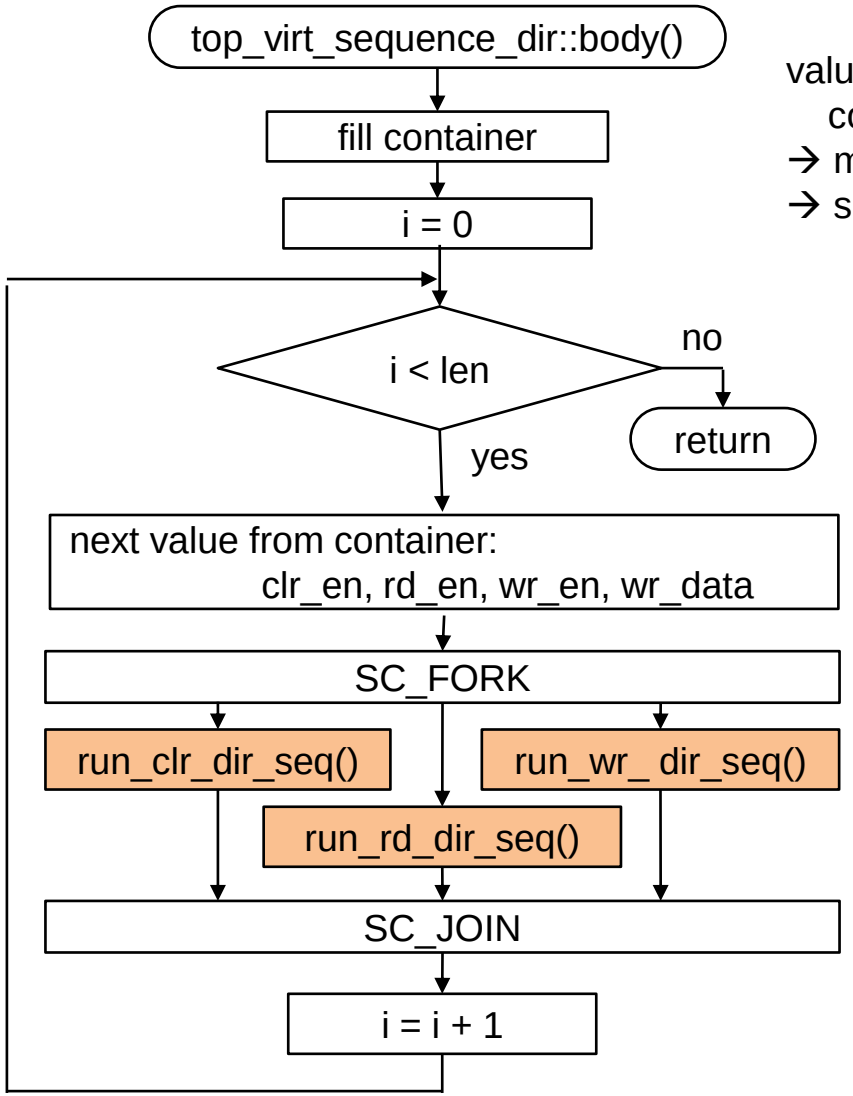




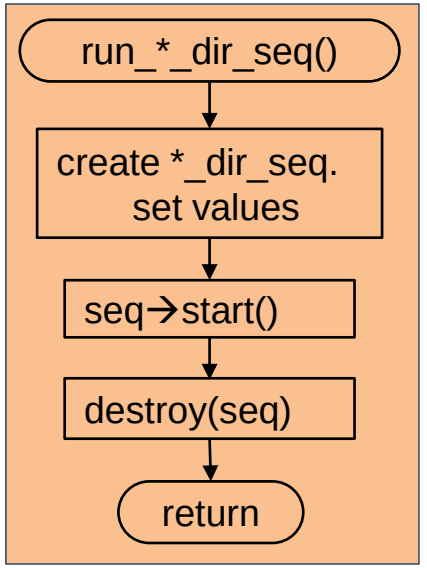
Directed sequences

top_virt_sequence_dir

clr_agent_env*
rd_agent_env*
wr_agent_env*



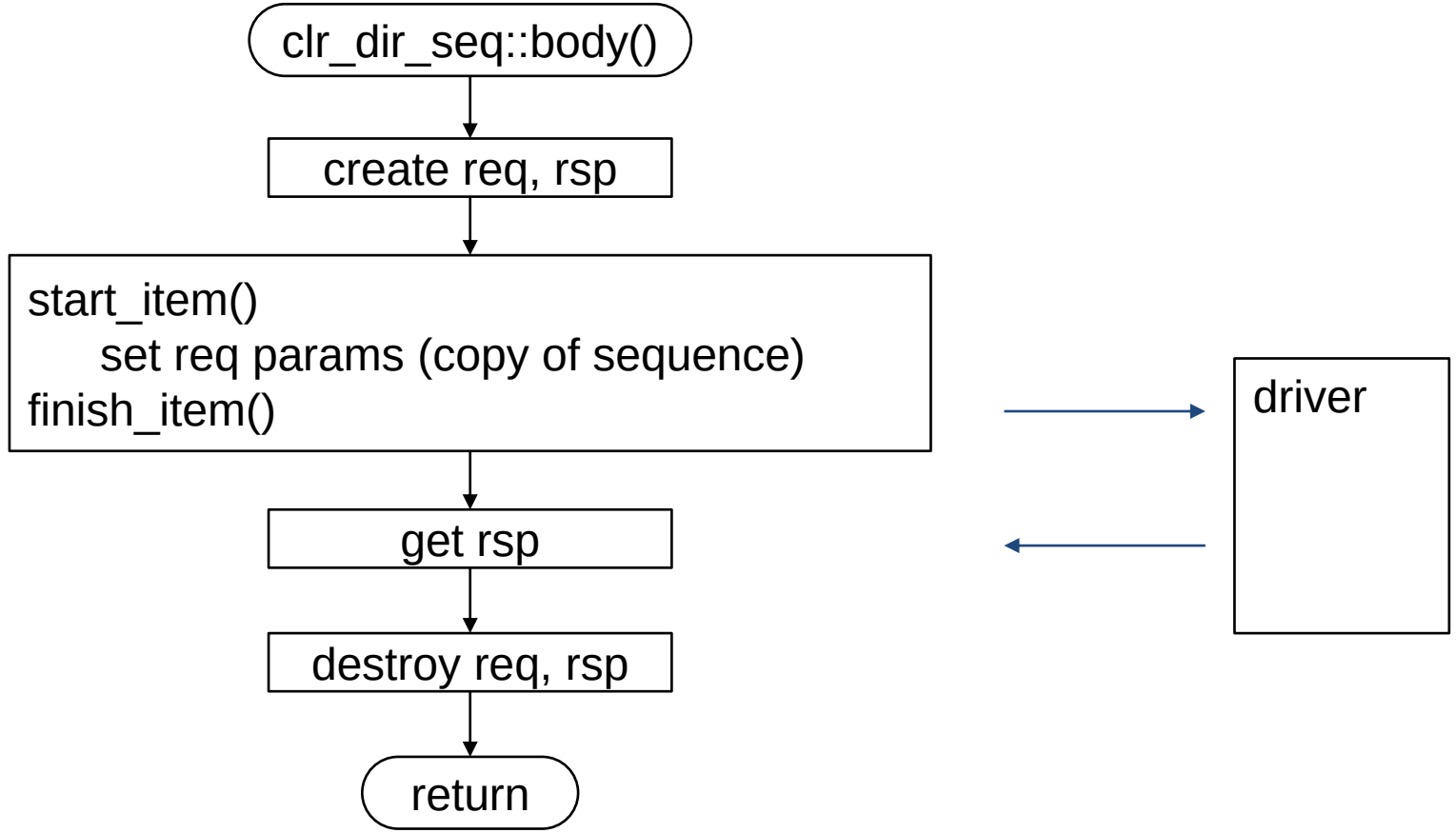
values are copied:
container
→ member of top_virt_sequence_dir
→ sequence



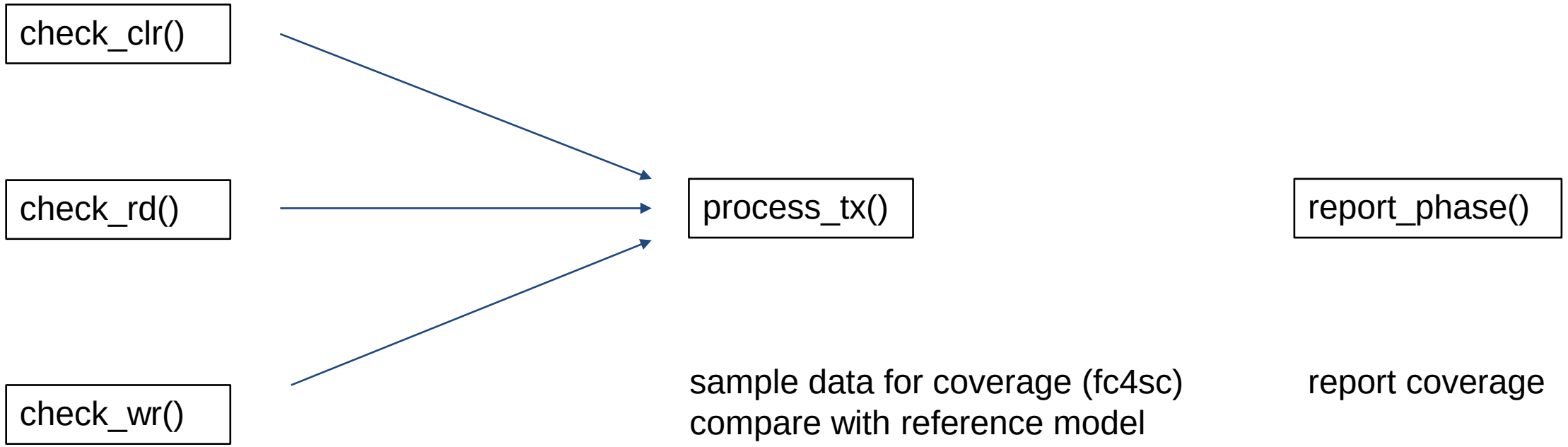
Creating directed transaction

clr_dir_seq

value



Scoreboard

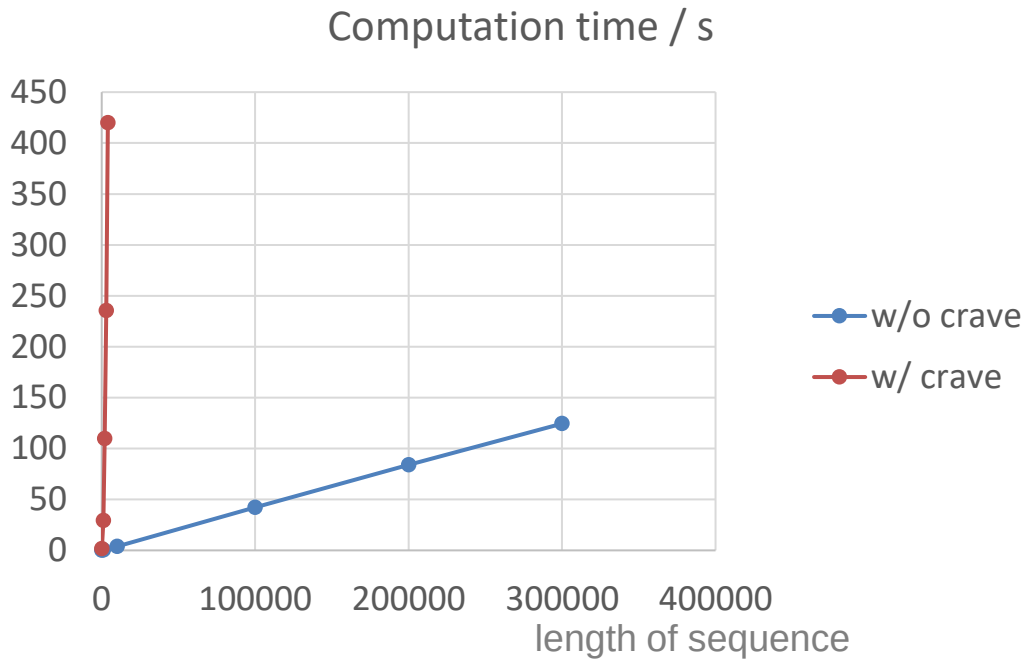


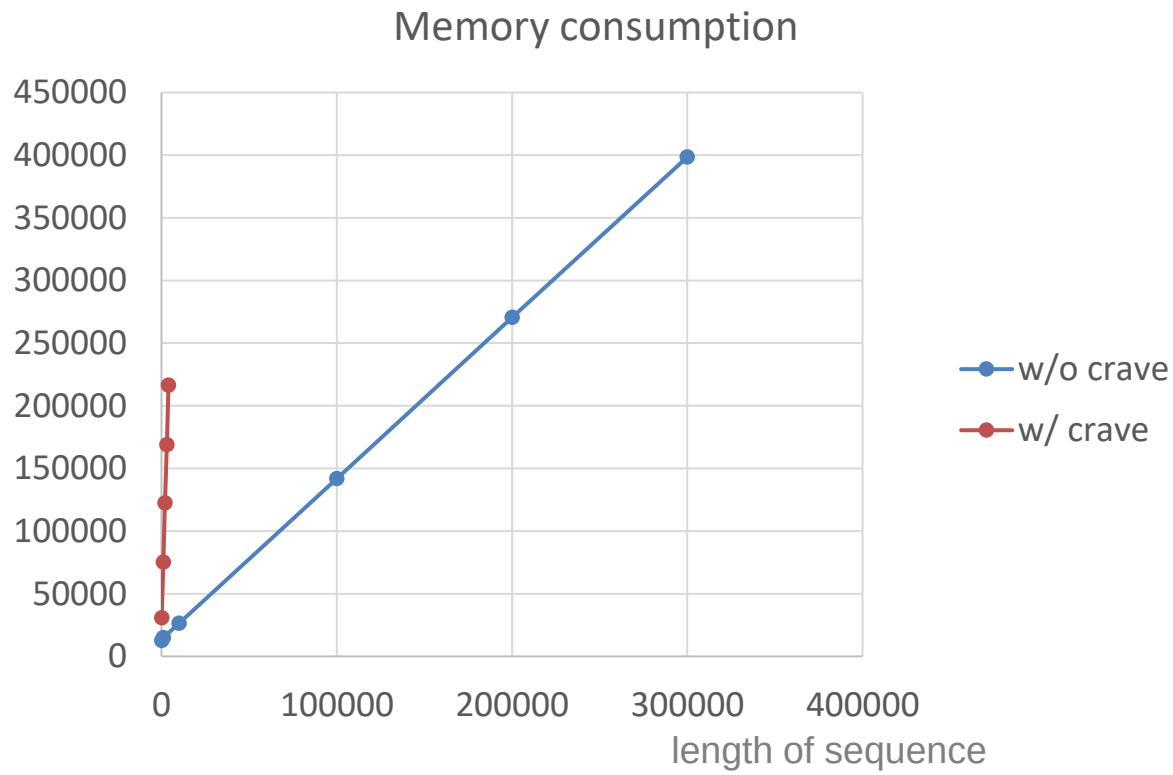
collect items in a class
→ one object per time
→ rsp contains time (set in monitor)
→ each monitor one rsp per clock cycle

sample data for coverage (fc4sc)
compare with reference model
if deviation
    report coverage
    UVM_FATAL()

report coverage

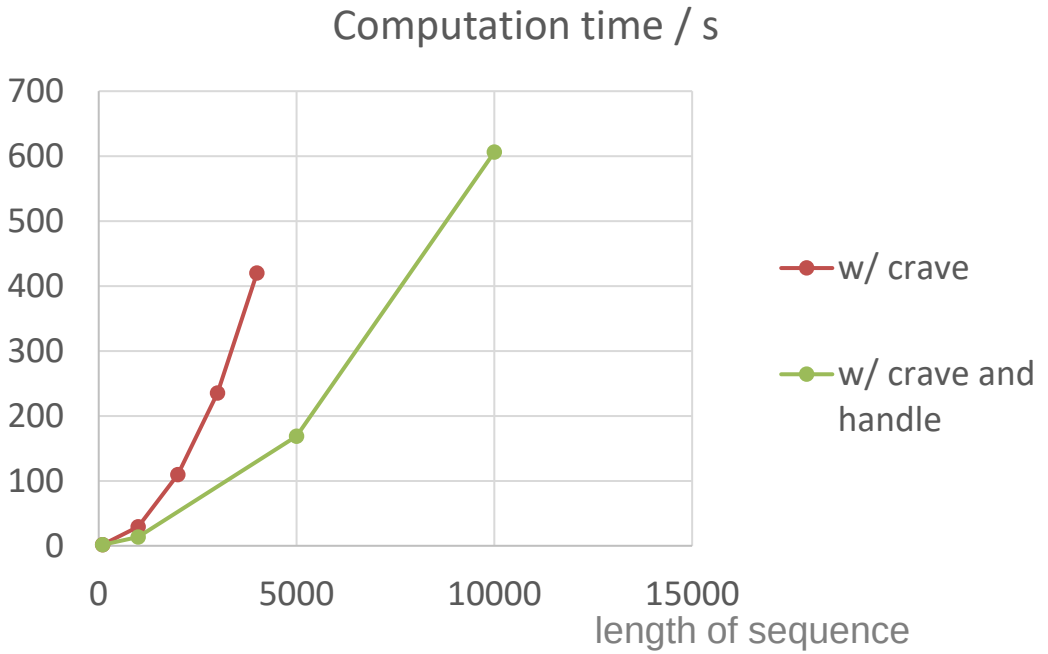
Performance

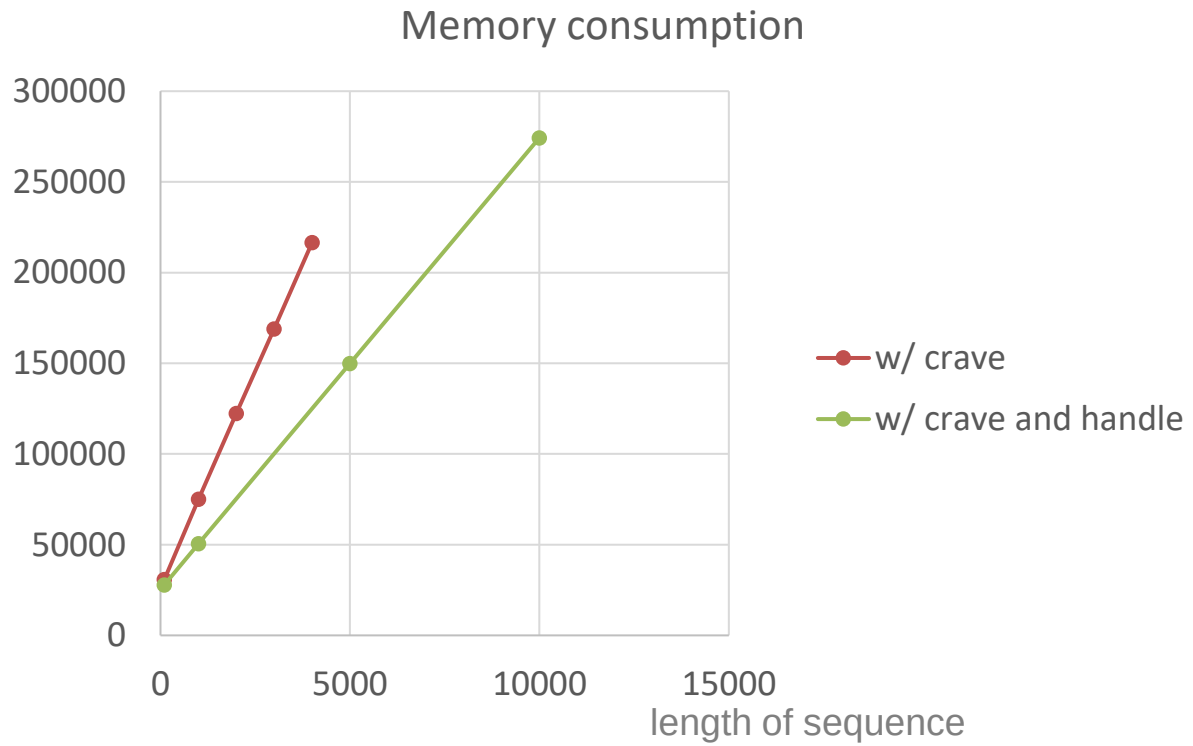




Why increasing? Leak?

Crave - enhancements





use handle object for random sequence items →
random item is allocated and deleted exactly one time [Thilo Voertler]

Crave

- parameters of random generation set in class definition by literal numbers
  - not in virtual sequence
  - use header file

top_virtual_sequence_cfg.h:

```
// likelihood of clear
#define clr_en_lh 5

// likelihood of read
#define rd_en_lh 60

// likelihood of write
#define wr_en_lh 65
```

fifo_clr_agent_fifo_clr_tx.h:

```
class fifo_clr_agent_fifo_clr_tx:
  public uvm_randomized_sequence_item
{
  …
  crave::crv_variable< sc_dt::sc_uint<1> > en;
  sc_core::sc_time   start_time;
  sc_core::sc_time   end_time;

  crave::crv_constraint c_en_arb{ crave::dist(en(),
    crave::make_distribution(
      crave::weighted_range<unsigned>(0, 0, 100-clr_en_lh),
      crave::weighted_range<unsigned>(1, 1, clr_en_lh) ))};
```

Conclusion

- first working version of the UVM SystemC testbench in field
- coverage measurement by fc4sc → needs patches

next steps

- development of a tutorial for students using Coside testbench generation for FIFO
- feedback from community for good practice
  - testbench structure (3 vs. 1 agent)
  - how to realize generation/scoreboard/coverage
  - crave: yes or no
- development of VIP for standard interfaces (UART/SPI/I2C etc.) for student projects

accellera SYSTEMS INITIATIVE™

SYSTEMC™ EVOLUTION FIKA