inventas

FDF 2025

HDLRegression – A reliable and efficient tool for FPGA regression testing

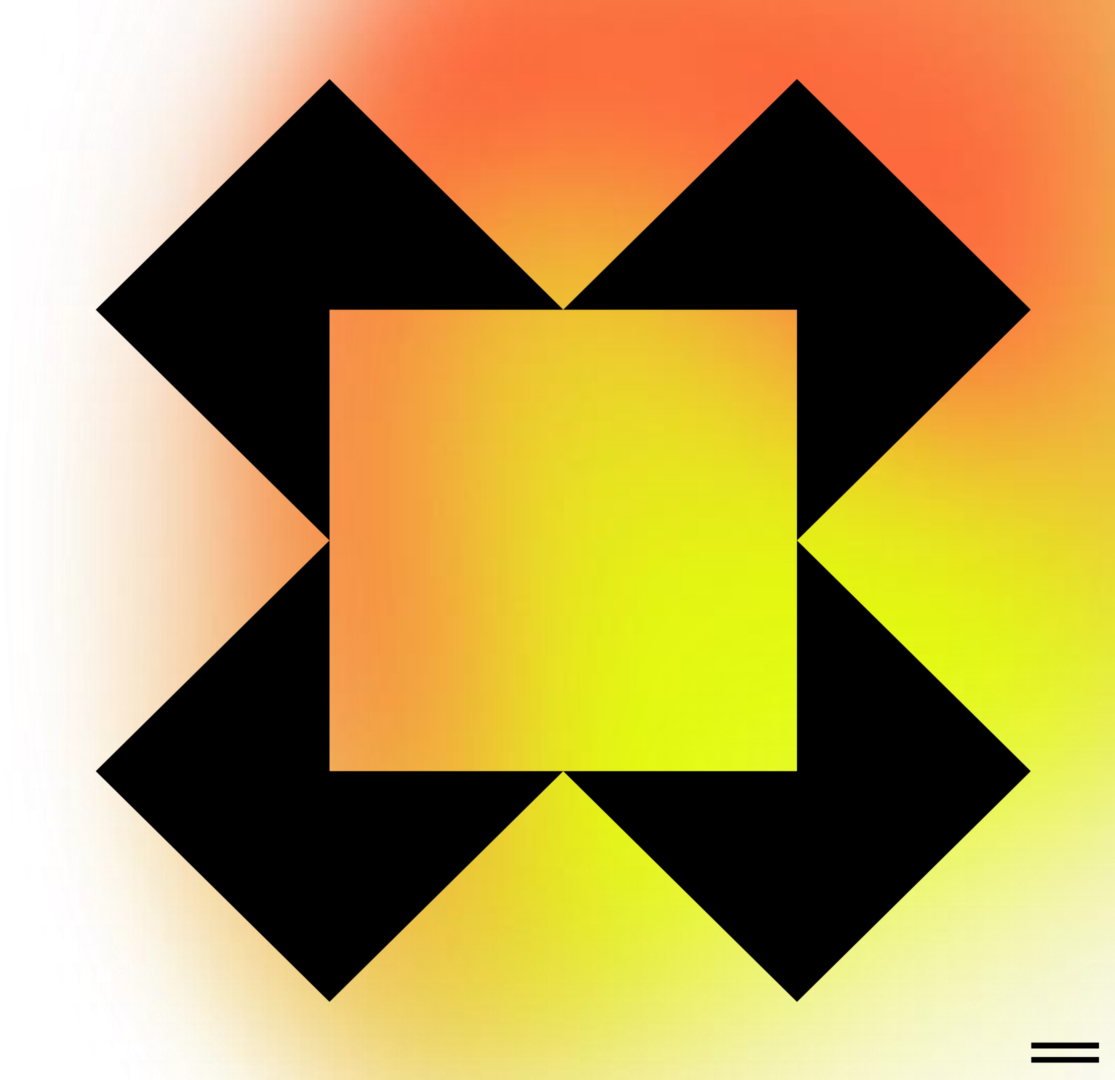
Inventas

- Norway’s largest independent design center and product development company
- Established in 1997
- 170 designers and engineers
- 2016 - Launched UVVM
- 2021 - Established UVVM Steering Group
- 2022 - Launched HDLRegression
- Provider of UVVM methodology and IP
- UVVM & HDLRegression webinars, training and presentations

ALDEC®
THE DESIGN VERIFICATION COMPANY

Topics

- **Motivation**
- HDLRegression
- Practical Example



Verification challenges

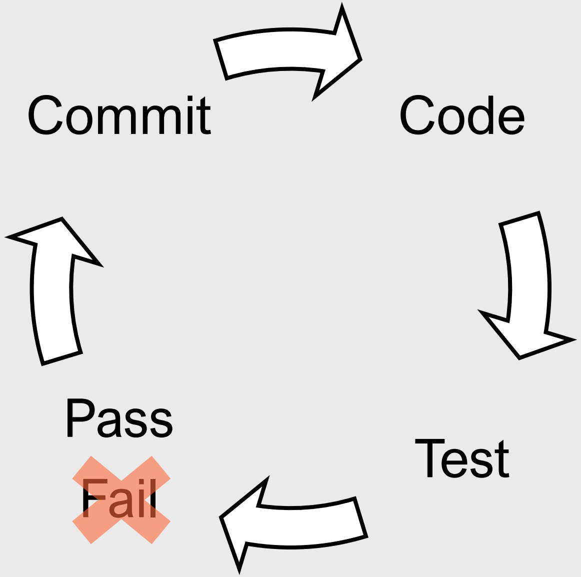
- 87 % of projects had at least one non-trivial bug escape into production.
- ≈ 67 % missed their original schedule.
- Verification can consume up to 50% of total project effort.
- Debugging is the single largest time sink for FPGA verification engineers.

*2024 Wilson Research Group FPGA functional verification trends

Why do these problems happen?

- Late discovery → many teams still only do a quick test before going to the lab; bugs appear late and cost more.
- Design complexity → SoC FPGAs, several clock domains, and embedded CPUs.
- Ad-hoc scripts → manually maintained compile orders and test lists get outdated fast.
- Unpredictable debug → one bad failing test can take days to fix.
- Change ripple → a fix in one block breaks logic that previously worked.

What is regression testing?

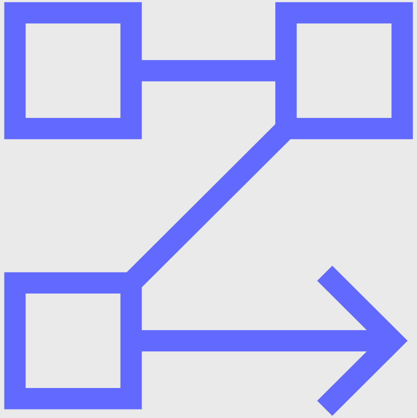


- Re-running the same functional and non-functional tests after each change.
- Confirms that behaviour we already validated still works.
- When something that passed yesterday fails today, that’s a regression.
- Must be fully automated and repeatable.
- Usual split: quick sanity pack and full suite overnight.

Benefits of regression testing

- Early fault detection – re-run tests after every code or configuration change.
- Foundation for CI pipelines – CI tools rely on healthy regression suite.
- Lower cost of defects – issues fixed at commit-time are orders of magnitude cheaper than those found in the lab – or in the field.
- Confidence to refactor and innovate – clean up code or add features without silently breaking existing functionality.

Continuous regression – what it is & why it helps



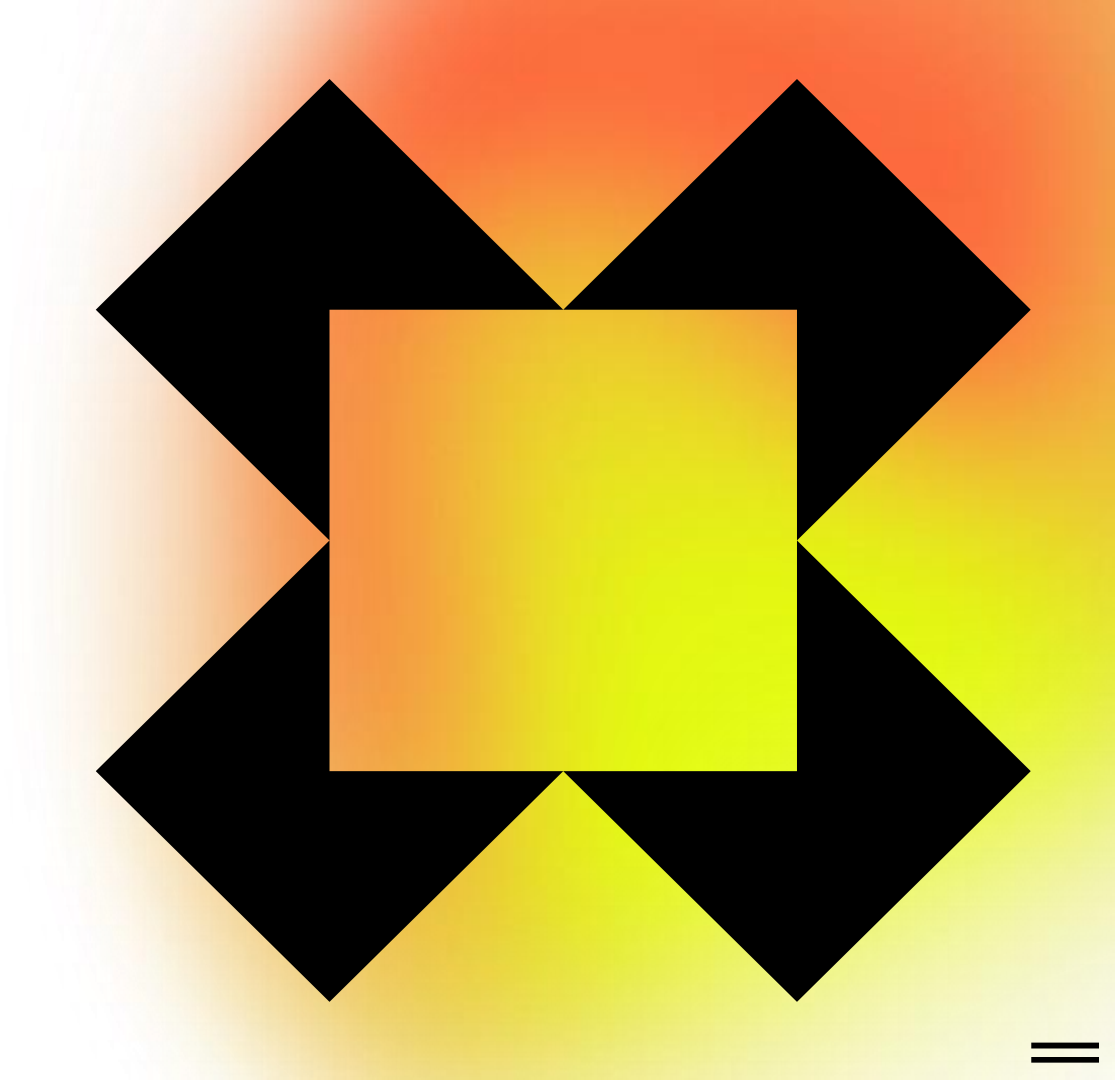
- Run tests each time we push code - we catch errors right away.
- Pipeline runs only tests impacted by the change.
- Small, quick set on every push; full set at night or before milestone.
- Keep logs from every run – easy to see when a bug appeared.
- Favours small and frequent commits – gives early feedback, easier to review, and easier to revert.

Topics

Motivation

HDLRegression

Practical Example

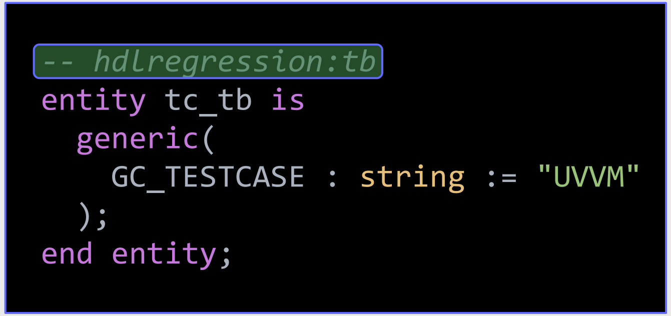


What is HDLRegression?

- An open-source regression runner for VHDL/Verilog TB – written in Python 3.
- Ultra-easy start-up – only a pragma comment in the TB; no refactor.
- Framework independent – works with UVVM, OSVVM, Vunit, or in-house code.
- Auto-detects ModelSim, Questa, Riviera-PRO, GHDL, NVC, and drives them for you.
- Scales from quick unit tests to large, multi-library projects.
- Goal: simplicity and efficiency → focus on writing good tests.

One-line testbench change

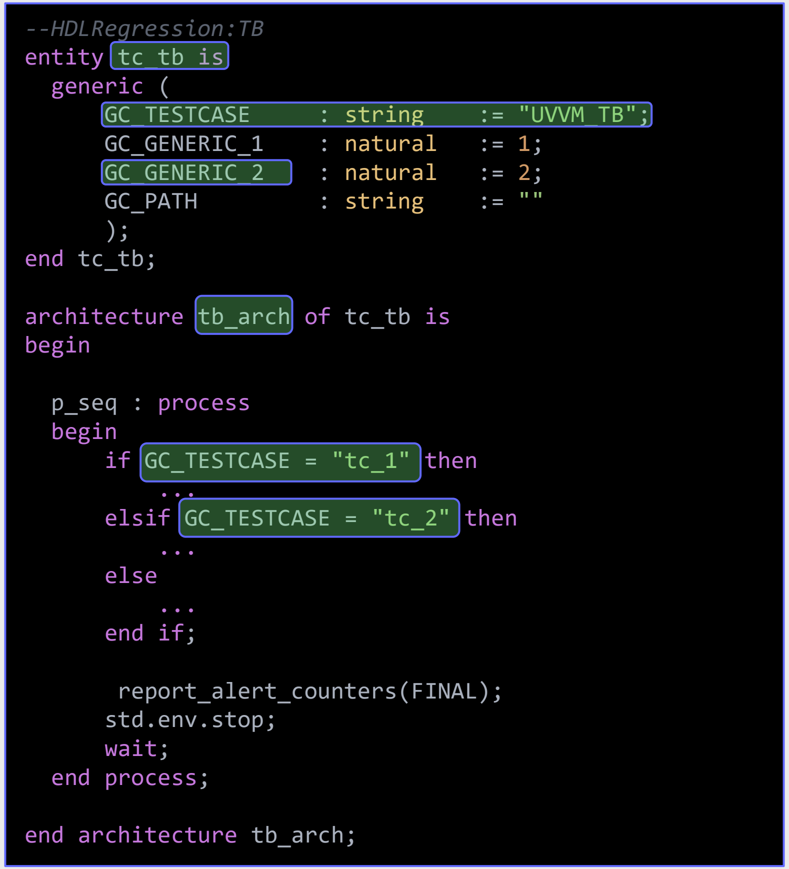
- Add one pragma comment at the top of the TB entity.
- No other edits, naming rules, or file-structure constraints.
- HDLRegression auto-discovers TB entities, architectures and built-in sequencer tests.



```vhdl
-- hdlregression:tb
entity tc_tb is
  generic(
    GC_TESTCASE : string := "UVVM"
  );
end entity;
```

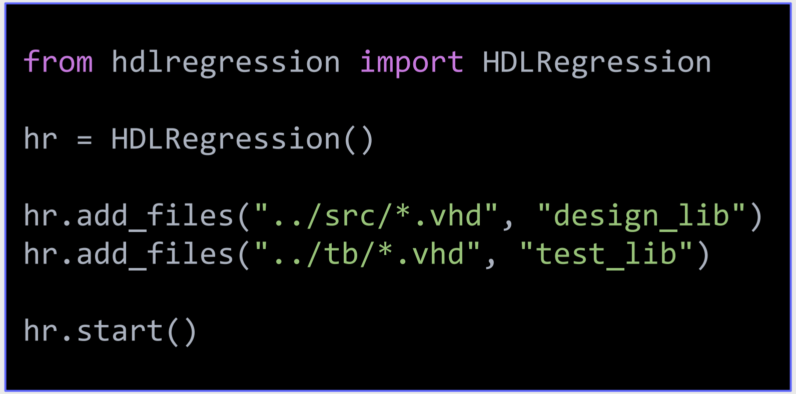
Test Case

- A sequence of actions controlled by the testbench sequencer.
- Can test one or multiple functions/requirements.
- Often test related functionality or a logical sequence of events.



```
--HDLRegression:TB
entity tc_tb is
  generic (
      GC_TESTCASE     : string    := "UVVM_TB";
      GC_GENERIC_1    : natural   := 1;
      GC_GENERIC_2    : natural   := 2;
      GC_PATH         : string    := ""
      );
end tc_tb;

architecture tb_arch of tc_tb is
begin

  p_seq : process
  begin
      if GC_TESTCASE = "tc_1" then
          ...
      elsif GC_TESTCASE = "tc_2" then
          ...
      else
          ...
      end if;

       report_alert_counters(FINAL);
      std.env.stop;
      wait;
  end process;

end architecture tb_arch;
```

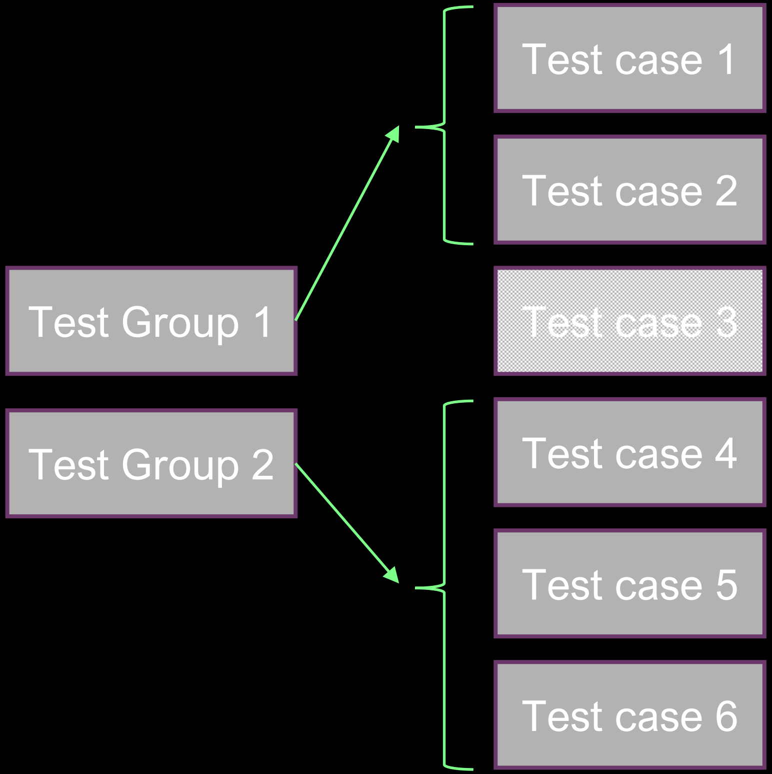
Simple Python script



```python
from hdlregression import HDLRegression

hr = HDLRegression()

hr.add_files("../src/*.vhd", "design_lib")
hr.add_files("../tb/*.vhd", "test_lib")

hr.start()
```

- 6-12 lines of plain Python.
- Import HDLRegression, list files & libraries, and call start().
- Same script runs RTL and netlist simulations – just add the netlist files.
- Use the same script to run locally and in CI pipelines.
- Full Python language available.

Test Group



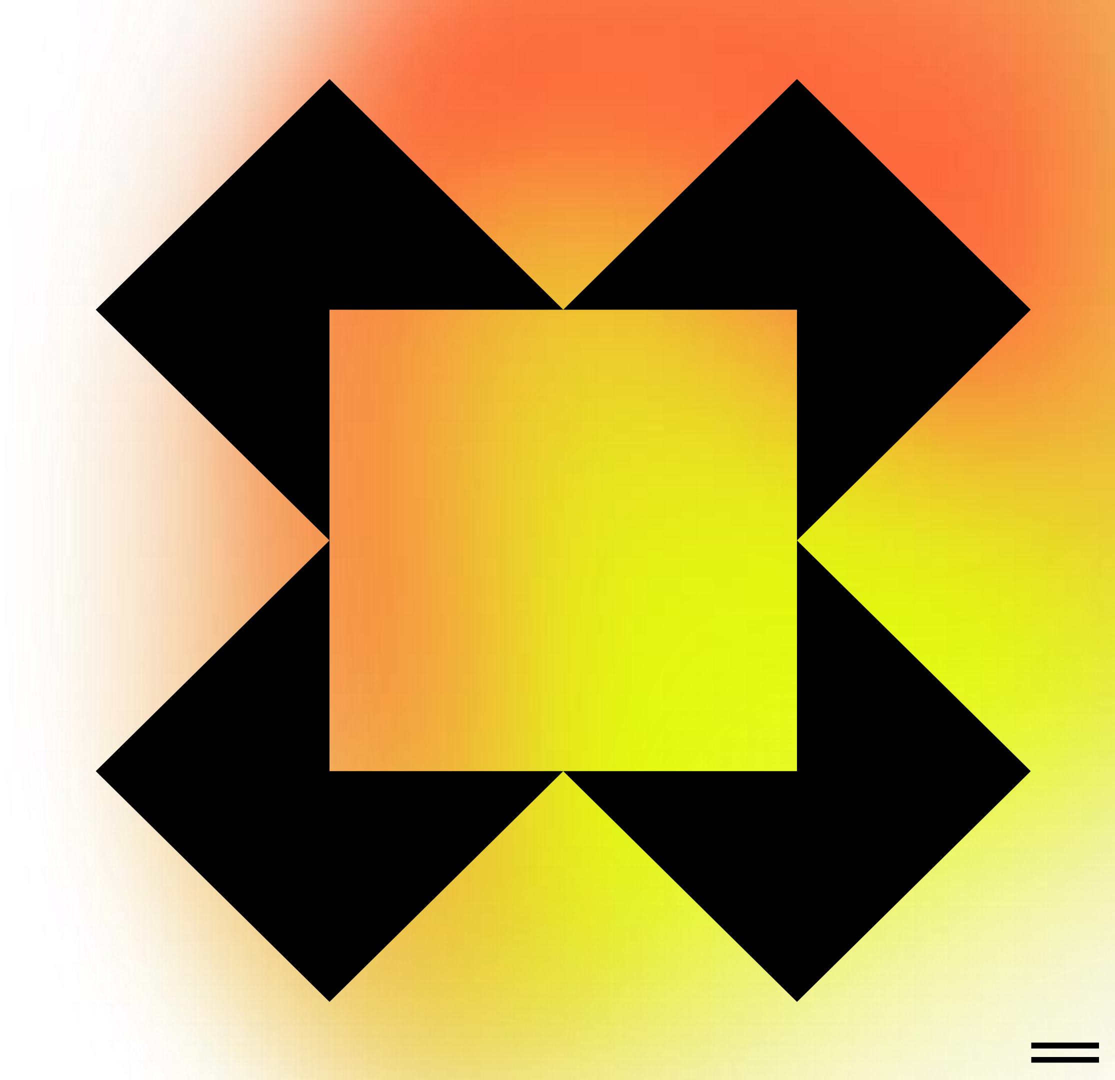
- A collection of test cases
- Test cases in a test group verify associated functionality or modules.

Command Line Interface

```
sim % python ../script/sim.py -h
usage: test.py [-h] [-v] [-d] [-g] [-fr] [-c] [-tc TESTCASE] [-tg TESTGROUP] [-ltc] [-etj EXPORTTESTCASEJSON] [-ltg] [-lco] [-fc] [-sof] [-s SIMULATOR]
              [-t [THREADING]] [-ns] [--waveFormat WAVEFORMAT] [--showWarnError] [--noColor] [--wlf]
HDLRegression CLI options
options:
  -h, --help            show this help message and exit
  -v, --verbose         enable full verbosity
  -d, --debug           enable debug mode
  -g, --gui             enable simulator GUI mode
  -fr, --fullRegression
                        run full regression
  -c, --clean           clean before regression run
  -tc TESTCASE, --testCase TESTCASE
                        run selected testbench[.architecture[.testcase]]
  -tg TESTGROUP, --testGroup TESTGROUP
                        run selected testgroup(s)
  -ltc, --listTestcase  list testcases
  -etj EXPORTTESTCASEJSON, --exportTestcaseJson EXPORTTESTCASEJSON
                        export testcases to JSON file with the given path. Example: --exportTestcaseJson testcases.json
  -ltg, --listTestgroup
                        list testgroups
  -lco, --listCompileOrder
                        list compile order
  -fc, --forceCompile   force recompile
  -sof, --stopOnFailure
                        stop simulations on testcase fail
  -s SIMULATOR, --simulator SIMULATOR
                        select simulator Modelsim/GHDL/NVC/Riviera_Pro/Vivado
  -t [THREADING], --threading [THREADING]
                        run tasks in parallel
  -ns, --no_sim         no simulation, only compilation
  --waveFormat WAVEFORMAT
                        wave file format [VCD (default) or FST]
  --showWarnError       Show error and warning messages during simulations.
  --noColor             Disable terminal output colors.
  --wlf                 Dumps wave file in WLF format (Questa/Modelsim).
```

Topics

- Motivation
- HDLRegression
- Practical Example



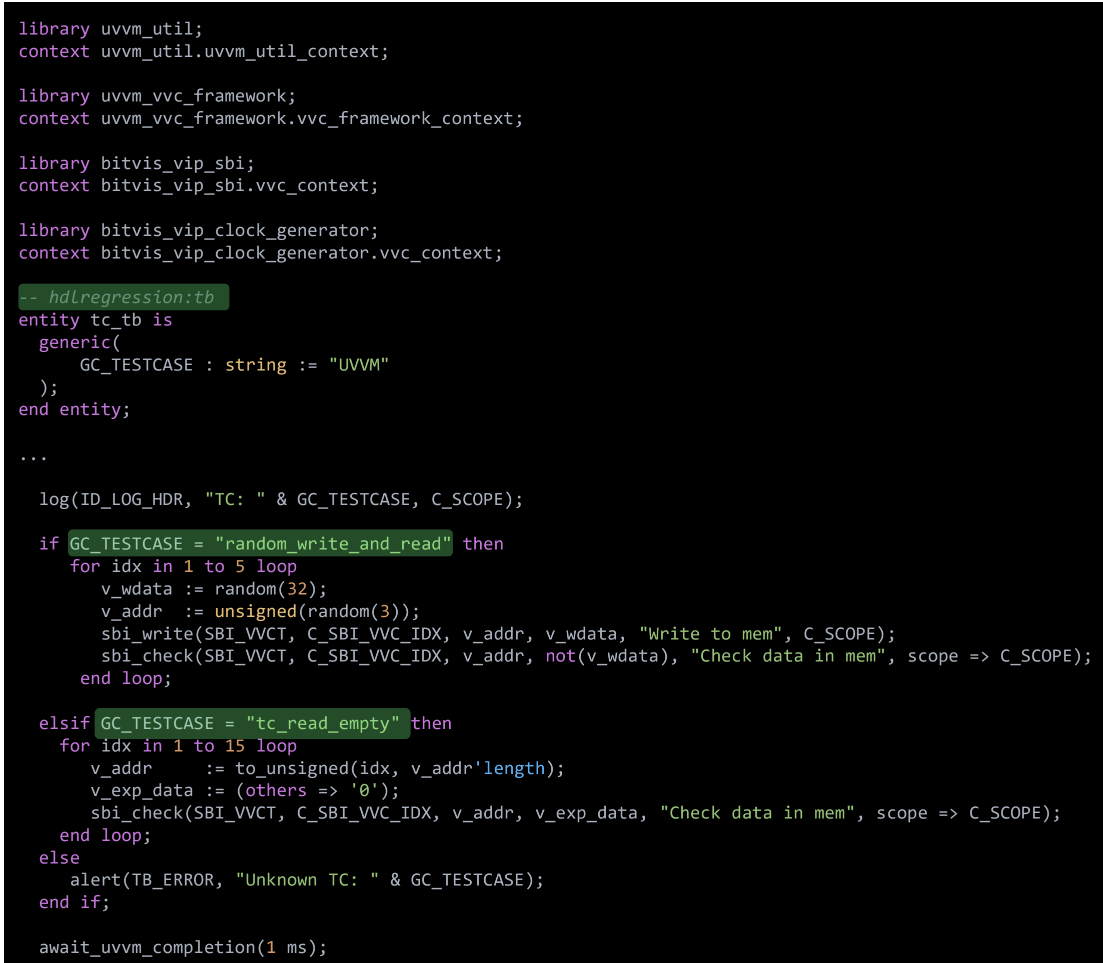
```
entity mem_block is
port (
  clk   : in  std_logic;
  rst   : in  std_logic;
  cs    : in  std_logic;
  wena  : in  std_logic;
  rena  : in  std_logic;
  addr  : in  unsigned(2 downto 0);
  wdata : in  std_logic_vector(31 downto 0);
  rdata : out std_logic_vector(31 downto 0);
  ready : out std_logic
);
end entity;

...

p_write : process(all) is
begin
  if rst = '1' then
    mem <= (others => (others => '0'));
  elsif rising_edge(clk) then
    if cs = '1' and wena = '1' then
        mem(to_integer(unsigned(addr))) <= wdata;
    end if;
  end if;
end process p_write;

p_read : process(all) is
begin
  if rst = '1' then
    rdata <= (others => '0');
  else
    if cs = '1' and rena = '1' then
        rdata <= mem(to_integer(unsigned(addr)));
    end if;
  end if;
end process p_read;
```

A simple memory block that allows writing data to and reading data from specific addresses.

The DUT has two processes:

- A **write process** (p_write) that updates the memory content on the rising clock edge when *cs* (chip select) and *wena* (write enable) are active.
- A **read process** (p_read) that provides data (rdata) when *cs* and *rena* (read enable) are active.

Uses a simple bus interface (SBI) approach, where the module is always “ready”, i.e. no wait cycles are needed for read or write operations.



```vhdl
library uvvm_util;
context uvvm_util.uvvm_util_context;

library uvvm_vvc_framework;
context uvvm_vvc_framework.vvc_framework_context;

library bitvis_vip_sbi;
context bitvis_vip_sbi.vvc_context;

library bitvis_vip_clock_generator;
context bitvis_vip_clock_generator.vvc_context;

-- hdlregression:tb
entity tc_tb is
  generic(
      GC_TESTCASE : string := "UVVM"
  );
end entity;

...

  log(ID_LOG_HDR, "TC: " & GC_TESTCASE, C_SCOPE);

  if GC_TESTCASE = "random_write_and_read" then
     for idx in 1 to 5 loop
        v_wdata := random(32);
        v_addr  := unsigned(random(3));
        sbi_write(SBI_VVCT, C_SBI_VVC_IDX, v_addr, v_wdata, "Write to mem", C_SCOPE);
        sbi_check(SBI_VVCT, C_SBI_VVC_IDX, v_addr, not(v_wdata), "Check data in mem", scope => C_SCOPE);
      end loop;

  elsif GC_TESTCASE = "tc_read_empty" then
    for idx in 1 to 15 loop
       v_addr     := to_unsigned(idx, v_addr'length);
       v_exp_data := (others => '0');
       sbi_check(SBI_VVCT, C_SBI_VVC_IDX, v_addr, v_exp_data, "Check data in mem", scope => C_SCOPE);
    end loop;
  else
     alert(TB_ERROR, "Unknown TC: " & GC_TESTCASE);
  end if;

  await_uvvm_completion(1 ms);
```

Testbench use UVVM for verification:
- Utility Library
- VVC Framework
- SBI VVC
- Clock Generator VVC

Marked for HDLRegression as testbench

Sequencer test cases. Each test is run isolated.

Manual simulation flow

```
nvc -L/proj/sim/hdlregression/library --work=uvvm_util:/proj/sim/hdlregression/library/uvvm_util --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_util/src/types_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=uvvm_util:/proj/sim/hdlregression/library/uvvm_util --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_util/src/adaptations_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=uvvm_util:/proj/sim/hdlregression/library/uvvm_util --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_util/src/string_methods_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=uvvm_util:/proj/sim/hdlregression/library/uvvm_util --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_util/src/protected_types_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=uvvm_util:/proj/sim/hdlregression/library/uvvm_util --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_util/src/license_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=uvvm_util:/proj/sim/hdlregression/library/uvvm_util --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_util/src/global_signals_and_shared_variables_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=uvvm_util:/proj/sim/hdlregression/library/uvvm_util --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_util/src/hierarchy_linked_list_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=uvvm_util:/proj/sim/hdlregression/library/uvvm_util --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_util/src/alert_hierarchy_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=uvvm_util:/proj/sim/hdlregression/library/uvvm_util --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_util/src/methods_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=uvvm_util:/proj/sim/hdlregression/library/uvvm_util --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_util/src/generic_queue_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=uvvm_util:/proj/sim/hdlregression/library/uvvm_util --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_util/src/bfm_common_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=uvvm_util:/proj/sim/hdlregression/library/uvvm_util --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_util/src/data_queue_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=uvvm_util:/proj/sim/hdlregression/library/uvvm_util --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_util/src/rand_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=uvvm_util:/proj/sim/hdlregression/library/uvvm_util --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_util/src/data_stack_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=uvvm_util:/proj/sim/hdlregression/library/uvvm_util --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_util/src/data_fifo_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=uvvm_util:/proj/sim/hdlregression/library/uvvm_util --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_util/src/func_cov_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=uvvm_util:/proj/sim/hdlregression/library/uvvm_util --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_util/src/uvvm_util_context.vhd
nvc -L/proj/sim/hdlregression/library --work=uvvm_vvc_framework:/proj/sim/hdlregression/library/uvvm_vvc_framework --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_vvc_framework/src/ti_data_queue_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=uvvm_vvc_framework:/proj/sim/hdlregression/library/uvvm_vvc_framework --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_vvc_framework/src/ti_protected_types_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=uvvm_vvc_framework:/proj/sim/hdlregression/library/uvvm_vvc_framework --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_vvc_framework/src/ti_generic_queue_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=uvvm_vvc_framework:/proj/sim/hdlregression/library/uvvm_vvc_framework --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_vvc_framework/src/ti_data_fifo_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=uvvm_vvc_framework:/proj/sim/hdlregression/library/uvvm_vvc_framework --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_vvc_framework/src/ti_data_stack_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=uvvm_vvc_framework:/proj/sim/hdlregression/library/uvvm_vvc_framework --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_vvc_framework/src/ti_vvc_framework_support_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=uvvm_vvc_framework:/proj/sim/hdlregression/library/uvvm_vvc_framework --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_vvc_framework/src/ti_uvvm_engine.vhd
nvc -L/proj/sim/hdlregression/library --work=uvvm_vvc_framework:/proj/sim/hdlregression/library/uvvm_vvc_framework --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_vvc_framework/src/ti_vvc_framework_context.vhd
nvc -L/proj/sim/hdlregression/library --work=bitvis_vip_scoreboard:/proj/sim/hdlregression/library/bitvis_vip_scoreboard --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/bitvis_vip_scoreboard/src/generic_sb_support_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=bitvis_vip_scoreboard:/proj/sim/hdlregression/library/bitvis_vip_scoreboard --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/bitvis_vip_scoreboard/src/generic_sb_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=bitvis_vip_scoreboard:/proj/sim/hdlregression/library/bitvis_vip_scoreboard --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/bitvis_vip_scoreboard/src/predefined_sb.vhd
nvc -L/proj/sim/hdlregression/library --work=bitvis_vip_sbi:/proj/sim/hdlregression/library/bitvis_vip_sbi --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/bitvis_vip_sbi/src/sbi_bfm_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=bitvis_vip_sbi:/proj/sim/hdlregression/library/bitvis_vip_sbi --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/bitvis_vip_sbi/src/transaction_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=bitvis_vip_sbi:/proj/sim/hdlregression/library/bitvis_vip_sbi --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/bitvis_vip_sbi/src/vvc_sb_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=bitvis_vip_sbi:/proj/sim/hdlregression/library/bitvis_vip_sbi --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/bitvis_vip_sbi/src/vvc_cmd_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=bitvis_vip_sbi:/proj/sim/hdlregression/library/bitvis_vip_sbi --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_vvc_framework/src_target_dependent/td_target_support_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=bitvis_vip_sbi:/proj/sim/hdlregression/library/bitvis_vip_sbi --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_vvc_framework/src_target_dependent/td_queue_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=bitvis_vip_sbi:/proj/sim/hdlregression/library/bitvis_vip_sbi --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/bitvis_vip_sbi/src/vvc_methods_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=bitvis_vip_sbi:/proj/sim/hdlregression/library/bitvis_vip_sbi --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_vvc_framework/src_target_dependent/td_vvc_framework_common_methods_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=bitvis_vip_sbi:/proj/sim/hdlregression/library/bitvis_vip_sbi --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/bitvis_vip_sbi/src/vvc_context.vhd
nvc -L/proj/sim/hdlregression/library --work=bitvis_vip_sbi:/proj/sim/hdlregression/library/bitvis_vip_sbi --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_vvc_framework/src_target_dependent/td_vvc_entity_support_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=bitvis_vip_sbi:/proj/sim/hdlregression/library/bitvis_vip_sbi --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/bitvis_vip_sbi/src/sbi_vvc.vhd
nvc -L/proj/sim/hdlregression/library --work=bitvis_vip_clock_generator:/proj/sim/hdlregression/library/bitvis_vip_clock_generator --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/bitvis_vip_clock_generator/src/vvc_cmd_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=bitvis_vip_clock_generator:/proj/sim/hdlregression/library/bitvis_vip_clock_generator --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_vvc_framework/src_target_dependent/td_target_support_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=bitvis_vip_clock_generator:/proj/sim/hdlregression/library/bitvis_vip_clock_generator --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_vvc_framework/src_target_dependent/td_queue_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=bitvis_vip_clock_generator:/proj/sim/hdlregression/library/bitvis_vip_clock_generator --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/bitvis_vip_clock_generator/src/vvc_methods_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=bitvis_vip_clock_generator:/proj/sim/hdlregression/library/bitvis_vip_clock_generator --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_vvc_framework/src_target_dependent/td_vvc_framework_common_methods_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=bitvis_vip_clock_generator:/proj/sim/hdlregression/library/bitvis_vip_clock_generator --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/bitvis_vip_clock_generator/src/vvc_context.vhd
nvc -L/proj/sim/hdlregression/library --work=bitvis_vip_clock_generator:/proj/sim/hdlregression/library/bitvis_vip_clock_generator --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/uvvm_vvc_framework/src_target_dependent/td_vvc_entity_support_pkg.vhd
nvc -L/proj/sim/hdlregression/library --work=bitvis_vip_clock_generator:/proj/sim/hdlregression/library/bitvis_vip_clock_generator --std=08 --stderr=error --messages=compact -M64m -H64m -a /UVVM/bitvis_vip_clock_generator/src/clock_generator_vvc.vhd
nvc -L/proj/sim/hdlregression/library --work=demo_lib:/proj/sim/hdlregression/library/demo_lib --std=08 --stderr=error --messages=compact -M64m -H64m -a /proj/src/mem_block.vhd
nvc -L/proj/sim/hdlregression/library --work=demo_lib:/proj/sim/hdlregression/library/demo_lib --std=08 --stderr=error --messages=compact -M64m -H64m -a /proj/tb/tc_tb.vhd
nvc -L/proj/sim/hdlregression/library --work=demo_lib:/proj/sim/hdlregression/library/demo_lib --std=08 --stderr=error --messages=compact -M64m -H64m -e --no-save --jit tc_tb-tb -gGC_TESTCASE=random_write_and_read -r
nvc -L/proj/sim/hdlregression/library --work=demo_lib:/proj/sim/hdlregression/library/demo_lib --std=08 --stderr=error --messages=compact -M64m -H64m -e --no-save --jit tc_tb-tb -gGC_TESTCASE=tc_read_empty -r
```

```python
from hdlregression import HDLRegression

hr = HDLRegression()

hr.add_files("../UVVM/uvvm_util/src/*.vhd", "uvvm_util")

hr.add_files("../UVVM/uvvm_vvc_framework/src/*.vhd", "uvvm_vvc_framework")

hr.add_files("../UVVM/bitvis_vip_sbi/src/*.vhd", "bitvis_vip_sbi")
hr.add_files("../UVVM/uvvm_vvc_framework/src_target_dependent/*.vhd", "bitvis_vip_sbi")

hr.add_files("../UVVM/bitvis_vip_scoreboard/src/*.vhd", "bitvis_vip_scoreboard")

hr.add_files("../UVVM/bitvis_vip_clock_generator/src/*.vhd",  "bitvis_vip_clock_generator")
hr.add_files("../UVVM/uvvm_vvc_framework/src_target_dependent/*.vhd",  "bitvis_vip_clock_generator")

hr.add_files("../src/*.vhd"), "demo_lib")
hr.add_files("../tb/*.vhd", "demo_lib")

hr.start()
```

Create a Python run script and import HDLRegression

Add all files and libraries used in TB and DUT

Call start() when finished adding files

Terminal

```
sim % python ../script/sim.py

===========================================================================
  HDLRegression version 0.61.2
  See /doc/hdlregression.pdf for documentation.
===========================================================================

Scanning files...
Building test suite structure...
```

Parsing files, detecting dependencies and building test suite structure

Terminal

```
sim % python ../script/sim.py

=====================================================================================
  HDLRegression version 0.61.2
  See /doc/hdlregression.pdf for documentation.
=====================================================================================

Scanning files...
Building test suite structure...
Simulator: NVC
Compiling library: uvvm_util  - OK -
Compiling library: uvvm_vvc_framework  - OK -
Compiling library: bitvis_vip_scoreboard  - OK -
Compiling library: bitvis_vip_sbi  - OK -
Compiling library: bitvis_vip_clock_generator  - OK -
Compiling library: demo_lib  - OK -
```

Parsing files, detecting dependencies and building test suite structure

Compiling libraries

Terminal

```
sim % python ../script/sim.py

========================================================================
  HDLRegression version 0.61.2
  See /doc/hdlregression.pdf for documentation.
========================================================================

Scanning files...
Building test suite structure...
Simulator: NVC
Compiling library: uvvm_util  - OK -
Compiling library: uvvm_vvc_framework  - OK -
Compiling library: bitvis_vip_scoreboard  - OK -
Compiling library: bitvis_vip_sbi  - OK -
Compiling library: bitvis_vip_clock_generator  - OK -
Compiling library: demo_lib  - OK -


Starting simulations...
Running 2 out of 2 test(s) using 1 thread(s).
Running: demo_lib.tc_tb.tb.random_write_and_read (test_id: 1)
Result: PASS (0h:0m:0s).

Running: demo_lib.tc_tb.tb.tc_read_empty (test_id: 2)
Result: PASS (0h:0m:0s).
```

Parsing files, detecting dependencies and building test suite structure

Compiling libraries

Number of tests in test-suite, current running test and test run result.

Terminal

```
sim % python ../script/sim.py

===========================================================================
 HDLRegression version 0.61.2
 See /doc/hdlregression.pdf for documentation.
===========================================================================

Scanning files...
Building test suite structure...
Simulator: NVC
Compiling library: uvvm_util  - OK -
Compiling library: uvvm_vvc_framework  - OK -
Compiling library: bitvis_vip_scoreboard  - OK -
Compiling library: bitvis_vip_sbi  - OK -
Compiling library: bitvis_vip_clock_generator  - OK -
Compiling library: demo_lib  - OK -


Starting simulations...
Running 2 out of 2 test(s) using 1 thread(s).
Running: demo_lib.tc_tb.tb.random_write_and_read (test_id: 1)
Result: PASS (0h:0m:0s).

Running: demo_lib.tc_tb.tb.tc_read_empty (test_id: 2)
Result: PASS (0h:0m:0s).

----------------------------------------------------------------------
Simulation run time: 0h:0m:0s.
SIMULATION SUCCESS: 2 passing test(s).
```

Parsing files, detecting dependencies and building test suite structure

Compiling libraries

Number of tests in test-suite, current running test and test run result.

Regression summary

Terminal

```
sim % python ../script/sim.py

==============================================================================
  HDLRegression version 0.61.2
  See /doc/hdlregression.pdf for documentation.
==============================================================================

Scanning files...
Building test suite structure...
Simulator: NVC

Starting simulations...
Test run not required. Use "-fr"/"--fullRegression" to re-run all tests.
```

No changes, no need to simulate

Regression Run

```
sim % touch ../src/mem_block.vhd

sim % python ../script/sim.py

==========================================================================
  HDLRegression version 0.61.2
  See /doc/hdlregression.pdf for documentation.
==========================================================================

Scanning files...
Building test suite structure...
Simulator: NVC
Compiling library: demo_lib  - OK -

Starting simulations...
Moving previous test run to: ./hdlregression/test_2025-04-30_11.46.58.379030.
Running 2 out of 2 test(s) using 1 thread(s).
Running: demo_lib.tc_tb.tb.random_write_and_read (test_id: 1)
Result: PASS (0h:0m:0s).

Running: demo_lib.tc_tb.tb.tc_read_empty (test_id: 2)
Result: PASS (0h:0m:0s).

----------------------------------------------------------------------
Simulation run time: 0h:0m:0s.
SIMULATION SUCCESS: 2 passing test(s).
```

Detecting changes, building test suite structure and compiling file(s).

Backup old test run and run affected tests

Failing Test Case

```
sim % python../script/sim.py

============================================================================
  HDLRegression version 0.61.2
  See /doc/hdlregression.pdf for documentation.
============================================================================

Scanning files...
Building test suite structure...
Simulator: NVC
Compiling library: demo_lib  - OK -


Starting simulations...
Moving previous test run to: ./hdlregression/test_2025-04-30_11.52.39.299812.
Running 2 out of 2 test(s) using 1 thread(s).
Running: demo_lib.tc_tb.tb.random_write_and_read (test_id: 1)
Result: FAIL (0h:0m:0s)
Test run: sim_errors=1, sim_warnings=0.
```

Failing test case is marked as FAIL – will be simulated in each run until marked as PASS.

Failing Test Case

```
Running 2 out of 2 test(s) using 1 thread(s).
Running: demo_lib.tc_tb.tb.random_write_and_read (test_id: 1)
Result: FAIL (0h:0m:0s)
Test run: sim_errors=1, sim_warnings=0.

================================================================================================================
UVVM: ID_UVVM_SEND_CMD          20.0 ns  tb_seq          ->sbi_check(SBI_VVC,1, x"6", x"F849F6DD"): 'Check data in mem'. [11]
UVVM: ID_CMD_INTERPRETER        20.0 ns  SBI_VVC,1        sbi_check(SBI_VVC,1, x"6", x"F849F6DD"). Command received  [11]
UVVM: ID_UVVM_CMD_ACK           20.0 ns  tb_seq            ACK received.   [11]
UVVM: ID_AWAIT_UVVM_COMPLETION   20.0 ns  TB seq.         await_uvvm_completion() => OK. All UVVM scoreboards are empty. Condition ...
UVVM: ID_CMD_INTERPRETER_WAIT    20.0 ns  SBI_VVC,1         ..Interpreter: Waiting for command
UVVM: ID_BFM                    32.5 ns  SBI_VVC,1        sbi_write(A:x"3", x"253B1447") completed. 'Write to mem'  [2]
UVVM: ID_CMD_EXECUTOR           32.5 ns  SBI_VVC,1        sbi_check(SBI_VVC,1, x"3", x"DAC4EBB8") - Will be executed  [3]
UVVM:
UVVM: ===========================================================================================================
UVVM: ***  ERROR #1  ***
UVVM:        42.5 ns   SBI_VVC,1
UVVM:               sbi_check(A:x"3", x"DAC4EBB8")=> Failed. Was x"253B1447". Expected x"DAC4EBB8".
UVVM:               'Check data in mem'  [3]
UVVM:
UVVM: Simulator has been paused as requested after 1 ERROR
UVVM: *** To find the root cause of this alert, step out the HDL calling stack in your simulator. ***
UVVM: *** For example, step out until you reach the call from the test sequencer. ***
UVVM: ===========================================================================================================
UVVM:
Running: demo_lib.tc_tb.tb.tc_read_empty (test_id: 2)
Result: PASS (0h:0m:0s).

---------------------------------------------------------------------------------------------------------------------------
Simulation run time: 0h:0m:0s.
SIMULATION FAIL: 2 tests run, 1 test(s) failed.
```

Log output from simulation

Run next test case and present final summary

Test Case

```
sim % python ../script/sim.py -ltc

================================================================================
  HDLRegression version 0.61.2
  See /doc/hdlregression.pdf for documentation.
================================================================================

Scanning files...
Building test suite structure...
TC:1 - tc_tb.tb.random_write_and_read
TC:2 - tc_tb.tb.tc_read_empty
```

List all test cases in test-suite

```
sim % py ../script/run.py -tc 2

================================================================================
  HDLRegression version 0.61.2
  See /doc/hdlregression.pdf for documentation.
================================================================================

Scanning files...
Building test suite structure...
Simulator: NVC

Starting simulations...
Running 1 out of 2 test(s) using 1 thread(s).
Running: demo_lib.tc_tb.tb.tc_read_empty (test_id: 2)
Result: PASS (0h:0m:0s).

--------------------------------------------------------------------------------
Simulation run time: 0h:0m:0s.
SIMULATION SUCCESS: 1 passing test(s).
```

Select test case to simulate

Test Group

```
...
hr.add_files(tb_path, "demo_lib")

hr.add_to_testgroup("long_tests", "tc_tb", "tb", "random_write_and_read")
hr.add_to_testgroup("short_tests", "tc_tb", "tb", "tc_read_empty")

hr.start()
```

Create test groups in regression script.

Any number of test groups, and any number of test cases in test group.

Format: <name>, <entity>, <architecture>, <tc>

```
sim % python ../script/sim.py -ltg
=================================================================
 HDLRegression version 0.61.2
 See /doc/hdlregression.pdf for documentation.
=================================================================

Scanning files...
Building test suite structure...
|---- long_tests
|    |--  tc_tb.tb.random_write_and_read
|---- short_tests
|    |--  tc_tb.tb.tc_read_empty
```

List test groups

Test Group

```
sim % py ../script/sim.py -tg short_tests

========================================================================
  HDLRegression version 0.61.2
  See /doc/hdlregression.pdf for documentation.
========================================================================

Scanning files...
Building test suite structure...
Simulator: NVC

Starting simulations...
Running 1 out of 2 test(s) using 1 thread(s).
Running: demo_lib.tc_tb.tb.tc_read_empty (test_id: 2)
Result: PASS (0h:0m:0s).

--------------------------------------------------------------------------
Simulation run time: 0h:0m:0s.
SIMULATION SUCCESS: 1 passing test(s).
```

Select test group to simulate

Verbosity

```
sim % py ../script/sim.py –tc 2 -v
...

Starting simulations...
Running 1 out of 2 test(s) using 1 thread(s).

UVVM: ID_UVVM_SEND_CMD          20.0 ns  tb_seq         ->sbi_check(SBI_VVC,1, x"1", x"00000000"): 'Check data in mem'. [2]
UVVM: ID_CMD_INTERPRETER        20.0 ns  SBI_VVC,1        sbi_check(SBI_VVC,1, x"1", x"00000000"). Command received  [2]
UVVM: ID_UVVM_CMD_ACK           20.0 ns  tb_seq           ACK received.   [2]
UVVM: ID_CMD_EXECUTOR           20.0 ns  SBI_VVC,1        sbi_check(SBI_VVC,1, x"1", x"00000000") - Will be executed  [2]
UVVM: ID_CMD_INTERPRETER_WAIT   20.0 ns  SBI_VVC,1        ..Interpreter: Waiting for command
UVVM: ID_UVVM_SEND_CMD          20.0 ns  tb_seq         ->sbi_check(SBI_VVC,1, x"2", x"00000000"): 'Check data in mem'. [3]
UVVM: ID_CMD_INTERPRETER        20.0 ns  SBI_VVC,1        sbi_check(SBI_VVC,1, x"2", x"00000000"). Command received  [3]
UVVM: ID_UVVM_CMD_ACK           20.0 ns  tb_seq           ACK received.   [3]
UVVM: ID_CMD_INTERPRETER_WAIT   20.0 ns  SBI_VVC,1        ..Interpreter: Waiting for command
UVVM: ID_UVVM_SEND_CMD          20.0 ns  tb_seq         ->sbi_check(SBI_VVC,1, x"3", x"00000000"): 'Check data in mem'. [4]
UVVM: ID_CMD_INTERPRETER        20.0 ns  SBI_VVC,1        sbi_check(SBI_VVC,1, x"3", x"
UVVM:
...

UVVM: ID_LOG_HDR                1172.5 ns  tb_seq                 SIMULATION COMPLETED
UVVM: ----------------------------------------------------------------------------------------------------------
UVVM:

Running: demo_lib.tc_tb.tb.tc_read_empty (test_id: 2)
Result: PASS (0h:0m:0s).
----------------------------------------------------------------------------------------------------------
Simulation run time: 0h:0m:0s.
SIMULATION SUCCESS: 1 passing test(s).
```

Select verbosity to get transcript in terminal

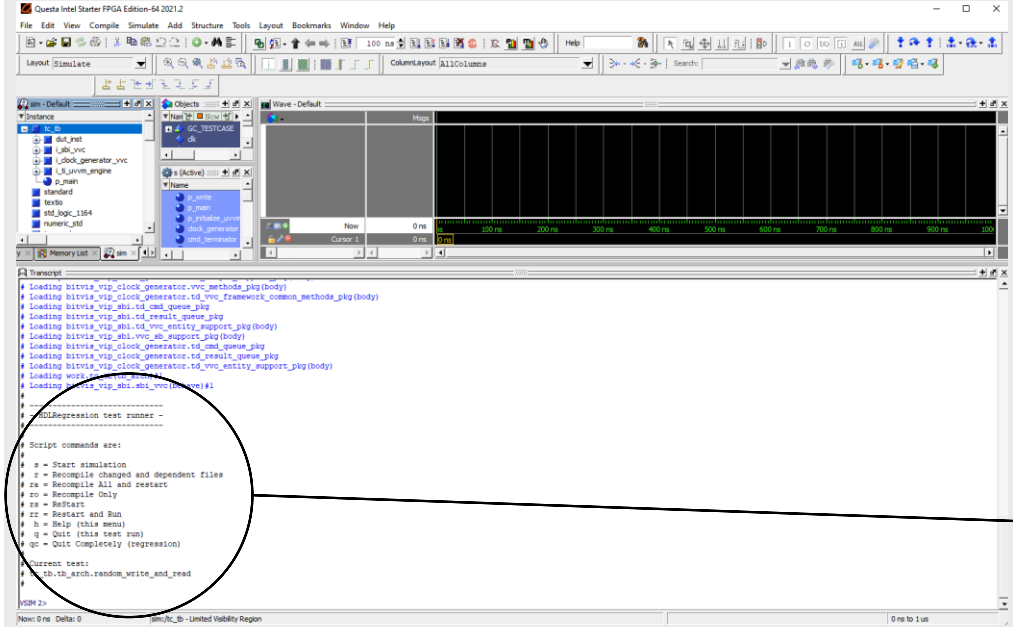
GUI

```
sim % python ../script/sim.py –tc 1 —-gui

=========================================================================
  HDLRegression version 0.61.2
  See /doc/hdlregression.pdf for documentation.
=========================================================================


Scanning files...
Building test suite structure...
Setting up: Y:/project/sim/hdlregression/library/modelsim.ini.
Compiling library: uvvm_util - OK -
Compiling library: uvvm_vvc_framework  - OK -
Compiling library: bitvis_vip_scoreboard  - OK -
Compiling library: bitvis_vip_sbi  - OK -
Compiling library: bitvis_vip_clock_generator  - OK -
Compiling library: demo_lib  - OK -
```

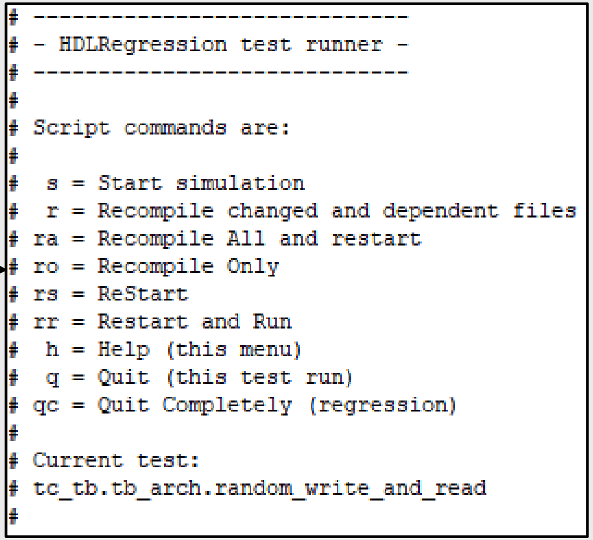
Run test case in GUI

GUI and waves

- `-g / --gui` CLI options opens ModelSim/Questa with project precompiled and test case loaded.
- Work interactively in GUI while debugging.
- GHDL / NVC dumps VCD/FST files.



```
# Loading bitvis_vip_clock_generator.vvc_methods_pkg(body)
# Loading bitvis_vip_clock_generator.td_vvc_framework_common_methods_pkg(body)
# Loading bitvis_vip_sbi.td_cmd_queue_pkg
# Loading bitvis_vip_sbi.td_result_queue_pkg
# Loading bitvis_vip_sbi.td_vvc_entity_support_pkg(body)
# Loading bitvis_vip_sbi.vvc_sb_support_pkg(body)
# Loading bitvis_vip_clock_generator.td_cmd_queue_pkg
# Loading bitvis_vip_clock_generator.td_result_queue_pkg
# Loading bitvis_vip_clock_generator.td_vvc_entity_support_pkg(body)
# Loading work.tc_tb(tb_arch)#1
# Loading bitvis_vip_sbi.sbi_vvc(behave)#1
#
# ------------------------------
# - HDLRegression test runner -
# ------------------------------
#
# Script commands are:
#
#  s = Start simulation
#  r = Recompile changed and dependent files
# ra = Recompile All and restart
# ro = Recompile Only
# rs = ReStart
# rr = Restart and Run
#  h = Help (this menu)
#  q = Quit (this test run)
# qc = Quit Completely (regression)
#
# Current test:
# tc_tb.tb_arch.random_write_and_read
#
VSIM 2>
```



```
# ------------------------------
# - HDLRegression test runner -
# ------------------------------
#
# Script commands are:
#
#  s = Start simulation
#  r = Recompile changed and dependent files
# ra = Recompile All and restart
# ro = Recompile Only
# rs = ReStart
# rr = Restart and Run
#  h = Help (this menu)
#  q = Quit (this test run)
# qc = Quit Completely (regression)
#
# Current test:
# tc_tb.tb_arch.random_write_and_read
#
```

```
cargo new --lib fibonacci_plugin
cd fibonacci_plugin
```

Create a Rust function - fibonacci

```
[package]
name = "fibonacci_plugin"
version = "0.1.0"
edition = "2021"

[lib]
crate-type = ["cdylib"]
```

Edit the Cargo.toml file to include crate-type for creating a shared library

```
pub extern "C" fn fibonacci(n: u32) -> u32 {
  if n < 2 {
      return n;
  }

  let mut f0: u32 = 0;
  let mut f1: u32 = 1;
  let mut f_next: u32;

  for _ in 2..=n {
      f_next = f0.wrapping_add(f1);
      f0 = f1;
      f1 = f_next;
  }
  f1
}
```

Write the Rust function
Exposed as extern "C" and compiled into a shared library (.so).

```
cd target/release
mv libfibonacci_plugin.dylib libfibonacci_plugin.so
```

Rename .dylib file to .so

```
cargo build --release
```

Build project

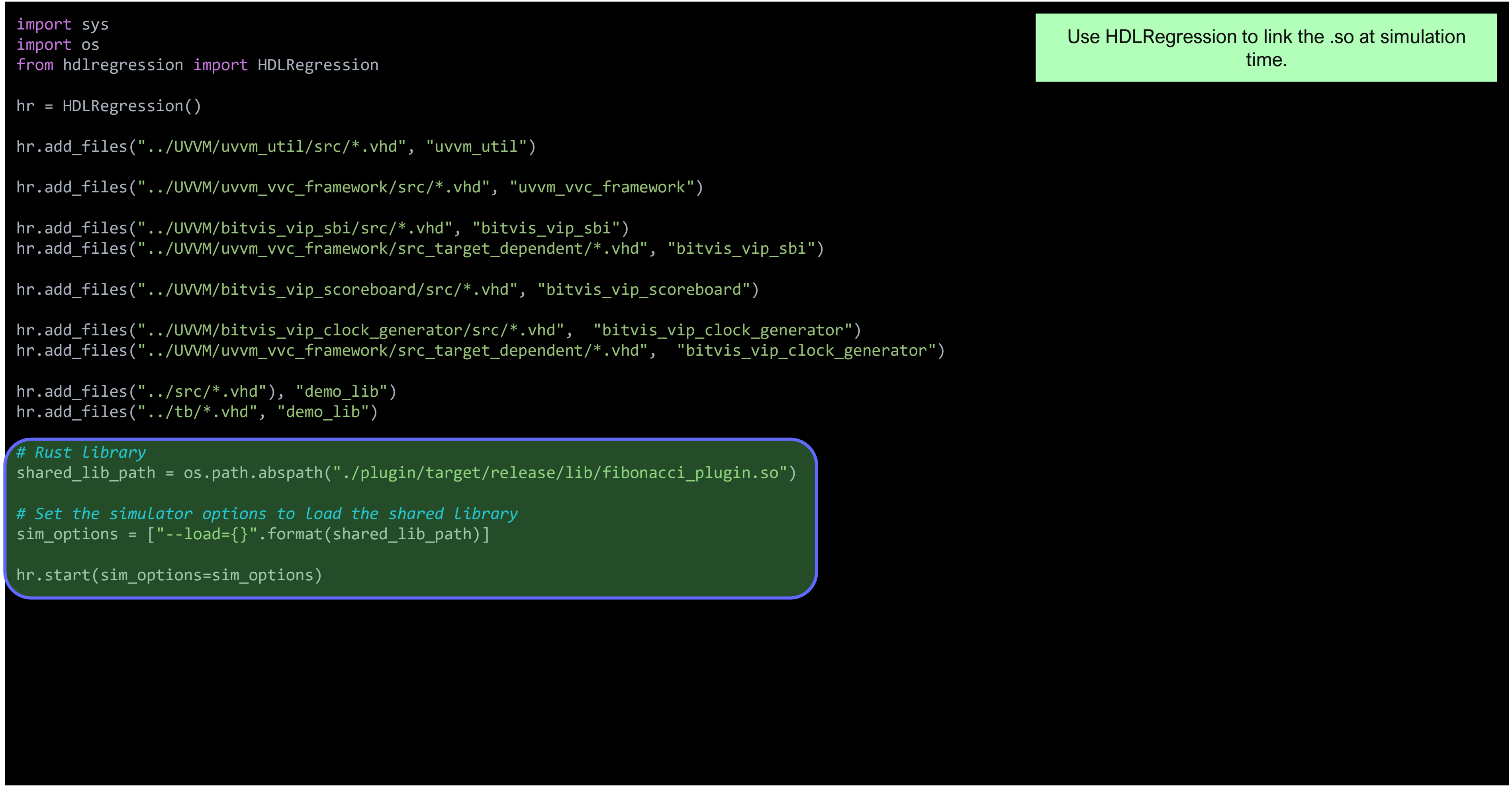
```
   function fibonacci(n : natural) return natural is
      begin
        report "This function should not be called in pure VHDL" severity failure;
      end function fibonacci;

   attribute foreign of fibonacci : function is "VHPIDIRECT fibonacci";
```

Write the VHDL function
- note the foreign attribute

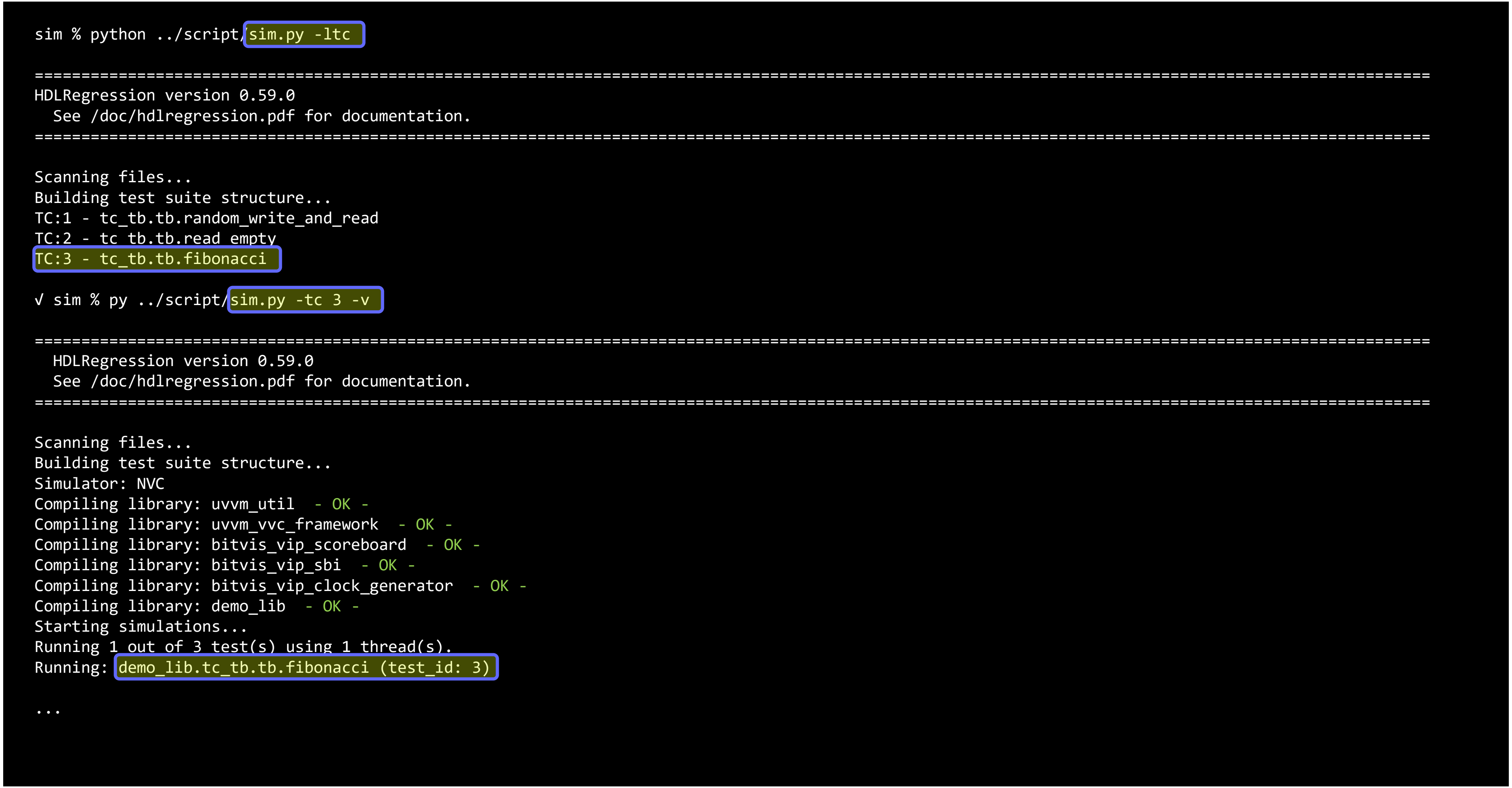
```
if GC_TESTCASE = "random_write_and_read" then
    for idx in 1 to 5 loop
        v_wdata := random(32);
        v_addr  := unsigned(random(3));
        sbi_write(SBI_VVCT, C_SBI_VVC_IDX, v_addr, v_wdata, "Write to mem", C_SCOPE);
        sbi_check(SBI_VVCT, C_SBI_VVC_IDX, v_addr, v_wdata, "Check data in mem", scope => C_SCOPE);
    end loop;

elsif GC_TESTCASE = "read_empty" then
    v_exp_data := (others => '0’);
    for idx in 1 to 15 loop
        v_addr     := to_unsigned(idx, v_addr'length);
        sbi_check(SBI_VVCT, C_SBI_VVC_IDX, v_addr, v_exp_data, "Check data in mem", scope => C_SCOPE);
    end loop;

elsif GC_TESTCASE = "fibonacci" then
    for idx in 1 to 15 loop
        v_wdata := std_logic_vector(to_unsigned(fibonacci(idx), v_wdata'length));
        v_addr  := unsigned(random(3));
        sbi_write(SBI_VVCT, C_SBI_VVC_IDX, v_addr, v_wdata, "Write to mem", C_SCOPE);
        sbi_check(SBI_VVCT, C_SBI_VVC_IDX, v_addr, v_wdata, "Check data in mem", scope => C_SCOPE);
    end loop;

else
    alert(TB_ERROR, "Unknowk TC: " & GC_TESTCASE);
end if;

await_uvvm_completion(1 ms);
```

Write a test case that use the fibonacci function

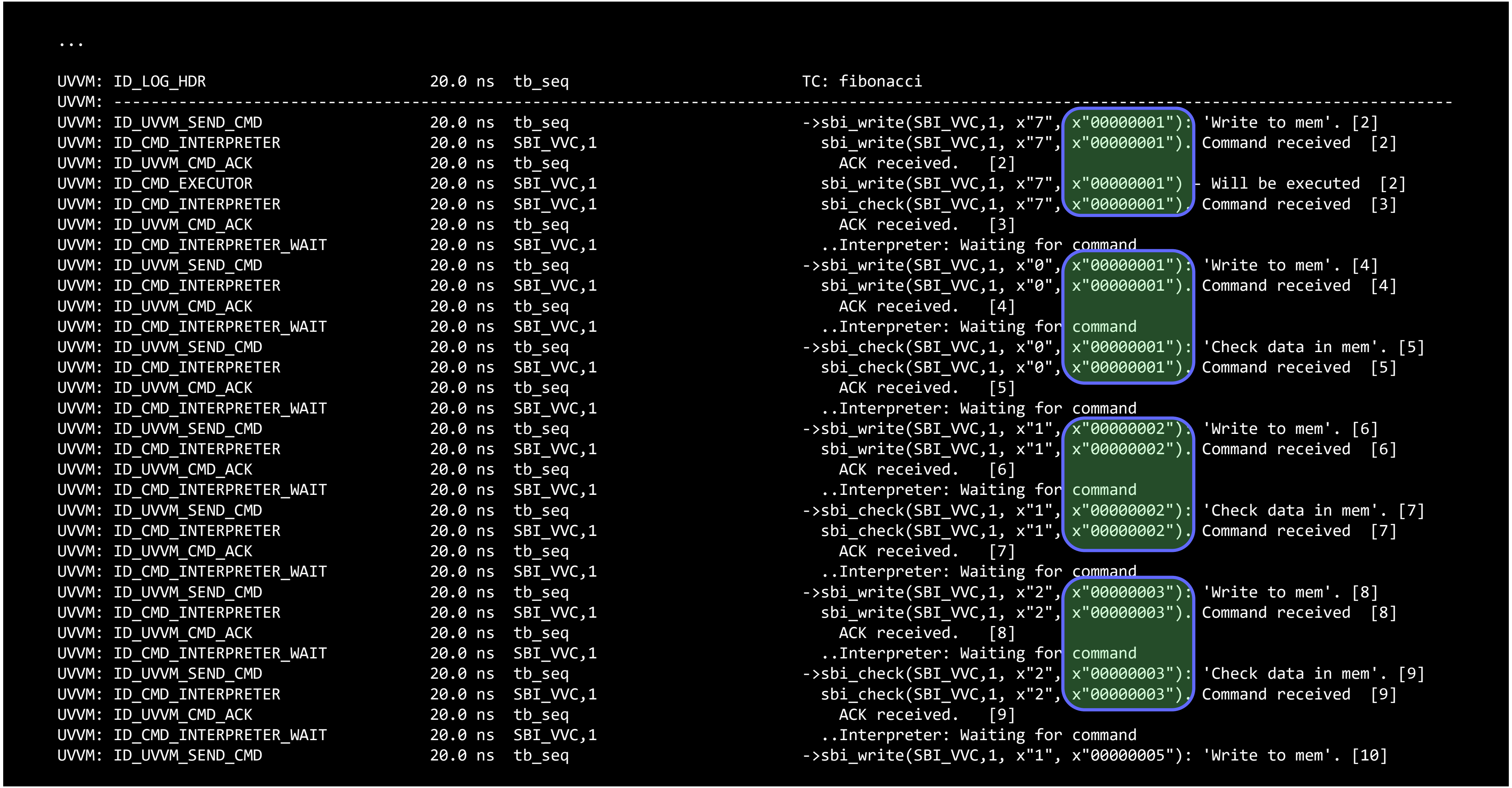
Use HDLRegression to link the .so at simulation time.



```
import sys
import os
from hdlregression import HDLRegression

hr = HDLRegression()

hr.add_files("../UVVM/uvvm_util/src/*.vhd", "uvvm_util")

hr.add_files("../UVVM/uvvm_vvc_framework/src/*.vhd", "uvvm_vvc_framework")

hr.add_files("../UVVM/bitvis_vip_sbi/src/*.vhd", "bitvis_vip_sbi")
hr.add_files("../UVVM/uvvm_vvc_framework/src_target_dependent/*.vhd", "bitvis_vip_sbi")

hr.add_files("../UVVM/bitvis_vip_scoreboard/src/*.vhd", "bitvis_vip_scoreboard")

hr.add_files("../UVVM/bitvis_vip_clock_generator/src/*.vhd",  "bitvis_vip_clock_generator")
hr.add_files("../UVVM/uvvm_vvc_framework/src_target_dependent/*.vhd",  "bitvis_vip_clock_generator")

hr.add_files("../src/*.vhd"), "demo_lib")
hr.add_files("../tb/*.vhd", "demo_lib")

# Rust library
shared_lib_path = os.path.abspath("./plugin/target/release/lib/fibonacci_plugin.so")

# Set the simulator options to load the shared library
sim_options = ["--load={}".format(shared_lib_path)]

hr.start(sim_options=sim_options)
```

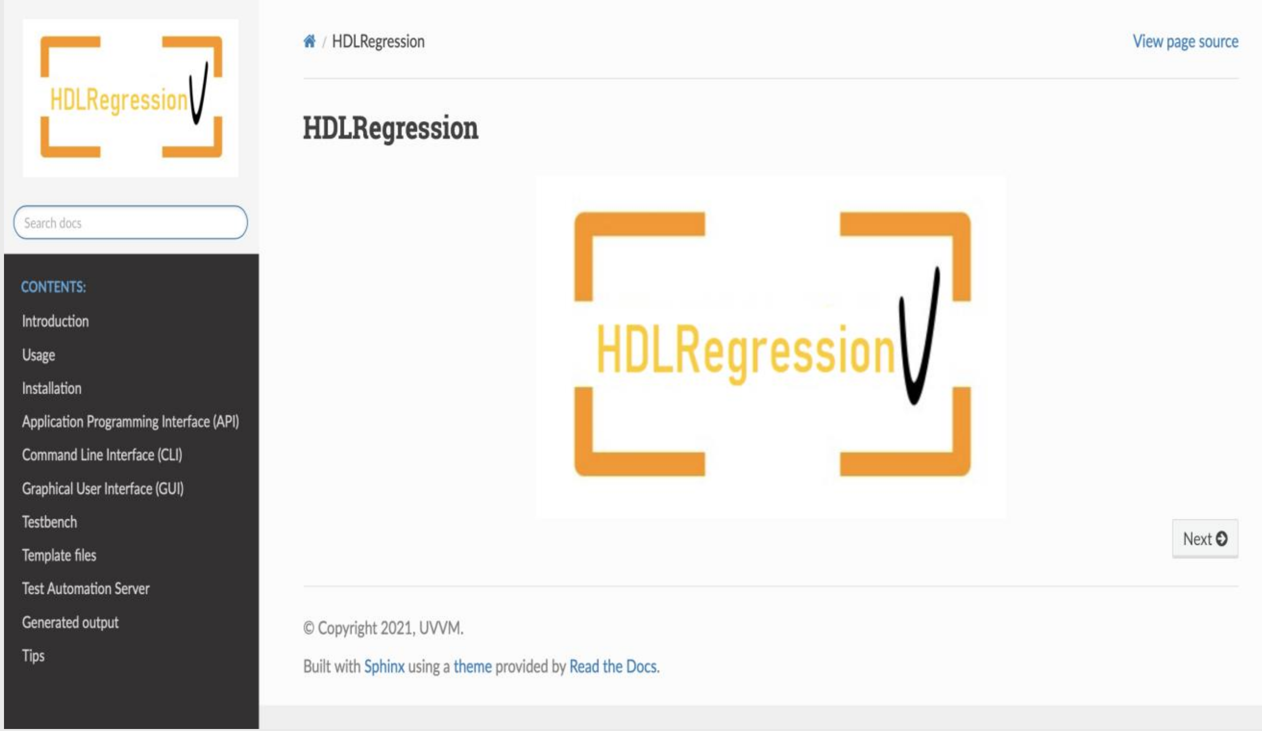


```
sim % python ../script/sim.py -ltc

======================================================================================================================================================
HDLRegression version 0.59.0
  See /doc/hdlregression.pdf for documentation.
======================================================================================================================================================

Scanning files...
Building test suite structure...
TC:1 - tc_tb.tb.random_write_and_read
TC:2 - tc_tb.tb.read_empty
TC:3 - tc_tb.tb.fibonacci

√ sim % py ../script/sim.py -tc 3 -v

======================================================================================================================================================
  HDLRegression version 0.59.0
  See /doc/hdlregression.pdf for documentation.
======================================================================================================================================================

Scanning files...
Building test suite structure...
Simulator: NVC
Compiling library: uvvm_util  - OK -
Compiling library: uvvm_vvc_framework  - OK -
Compiling library: bitvis_vip_scoreboard  - OK -
Compiling library: bitvis_vip_sbi  - OK -
Compiling library: bitvis_vip_clock_generator  - OK -
Compiling library: demo_lib  - OK -
Starting simulations...
Running 1 out of 3 test(s) using 1 thread(s).
Running: demo_lib.tc_tb.tb.fibonacci (test_id: 3)

...
```



```
...

UVVM: ID_LOG_HDR                        20.0 ns  tb_seq                         TC: fibonacci
UVVM: ------------------------------------------------------------------------------------------------------------------------------------------------
UVVM: ID_UVVM_SEND_CMD                  20.0 ns  tb_seq                         ->sbi_write(SBI_VVC,1, x"7", x"00000001"): 'Write to mem'. [2]
UVVM: ID_CMD_INTERPRETER                20.0 ns  SBI_VVC,1                        sbi_write(SBI_VVC,1, x"7", x"00000001"). Command received  [2]
UVVM: ID_UVVM_CMD_ACK                   20.0 ns  tb_seq                             ACK received.   [2]
UVVM: ID_CMD_EXECUTOR                   20.0 ns  SBI_VVC,1                        sbi_write(SBI_VVC,1, x"7", x"00000001") - Will be executed  [2]
UVVM: ID_CMD_INTERPRETER                20.0 ns  SBI_VVC,1                        sbi_check(SBI_VVC,1, x"7", x"00000001"). Command received  [3]
UVVM: ID_UVVM_CMD_ACK                   20.0 ns  tb_seq                             ACK received.   [3]
UVVM: ID_CMD_INTERPRETER_WAIT           20.0 ns  SBI_VVC,1                        ..Interpreter: Waiting for command
UVVM: ID_UVVM_SEND_CMD                  20.0 ns  tb_seq                         ->sbi_write(SBI_VVC,1, x"0", x"00000001"): 'Write to mem'. [4]
UVVM: ID_CMD_INTERPRETER                20.0 ns  SBI_VVC,1                        sbi_write(SBI_VVC,1, x"0", x"00000001"). Command received  [4]
UVVM: ID_UVVM_CMD_ACK                   20.0 ns  tb_seq                             ACK received.   [4]
UVVM: ID_CMD_INTERPRETER_WAIT           20.0 ns  SBI_VVC,1                        ..Interpreter: Waiting for command
UVVM: ID_UVVM_SEND_CMD                  20.0 ns  tb_seq                         ->sbi_check(SBI_VVC,1, x"0", x"00000001"): 'Check data in mem'. [5]
UVVM: ID_CMD_INTERPRETER                20.0 ns  SBI_VVC,1                        sbi_check(SBI_VVC,1, x"0", x"00000001"). Command received  [5]
UVVM: ID_UVVM_CMD_ACK                   20.0 ns  tb_seq                             ACK received.   [5]
UVVM: ID_CMD_INTERPRETER_WAIT           20.0 ns  SBI_VVC,1                        ..Interpreter: Waiting for command
UVVM: ID_UVVM_SEND_CMD                  20.0 ns  tb_seq                         ->sbi_write(SBI_VVC,1, x"1", x"00000002"): 'Write to mem'. [6]
UVVM: ID_CMD_INTERPRETER                20.0 ns  SBI_VVC,1                        sbi_write(SBI_VVC,1, x"1", x"00000002"). Command received  [6]
UVVM: ID_UVVM_CMD_ACK                   20.0 ns  tb_seq                             ACK received.   [6]
UVVM: ID_CMD_INTERPRETER_WAIT           20.0 ns  SBI_VVC,1                        ..Interpreter: Waiting for command
UVVM: ID_UVVM_SEND_CMD                  20.0 ns  tb_seq                         ->sbi_check(SBI_VVC,1, x"1", x"00000002"): 'Check data in mem'. [7]
UVVM: ID_CMD_INTERPRETER                20.0 ns  SBI_VVC,1                        sbi_check(SBI_VVC,1, x"1", x"00000002"). Command received  [7]
UVVM: ID_UVVM_CMD_ACK                   20.0 ns  tb_seq                             ACK received.   [7]
UVVM: ID_CMD_INTERPRETER_WAIT           20.0 ns  SBI_VVC,1                        ..Interpreter: Waiting for command
UVVM: ID_UVVM_SEND_CMD                  20.0 ns  tb_seq                         ->sbi_write(SBI_VVC,1, x"2", x"00000003"): 'Write to mem'. [8]
UVVM: ID_CMD_INTERPRETER                20.0 ns  SBI_VVC,1                        sbi_write(SBI_VVC,1, x"2", x"00000003"). Command received  [8]
UVVM: ID_UVVM_CMD_ACK                   20.0 ns  tb_seq                             ACK received.   [8]
UVVM: ID_CMD_INTERPRETER_WAIT           20.0 ns  SBI_VVC,1                        ..Interpreter: Waiting for command
UVVM: ID_UVVM_SEND_CMD                  20.0 ns  tb_seq                         ->sbi_check(SBI_VVC,1, x"2", x"00000003"): 'Check data in mem'. [9]
UVVM: ID_CMD_INTERPRETER                20.0 ns  SBI_VVC,1                        sbi_check(SBI_VVC,1, x"2", x"00000003"). Command received  [9]
UVVM: ID_UVVM_CMD_ACK                   20.0 ns  tb_seq                             ACK received.   [9]
UVVM: ID_CMD_INTERPRETER_WAIT           20.0 ns  SBI_VVC,1                        ..Interpreter: Waiting for command
UVVM: ID_UVVM_SEND_CMD                  20.0 ns  tb_seq                         ->sbi_write(SBI_VVC,1, x"1", x"00000005"): 'Write to mem'. [10]
```

Documentation



Search docs

CONTENTS:

- Introduction
- Usage
- Installation
- Application Programming Interface (API)
- Command Line Interface (CLI)
- Graphical User Interface (GUI)
- Testbench
- Template files
- Test Automation Server
- Generated output
- Tips

/ HDLRegression

View page source

HDLRegression

Next

© Copyright 2021, UVVM.

Built with Sphinx using a theme provided by Read the Docs.

Key take-aways

- Regression improves the verification flow.
- Use CI pipelines for running regression on each push.
- A reliable regression tool and self-checking TB is key.
- HDLRegression is easy to use
  - One pragma in the TB.
  - One small Python script.
  - Works with any verification FW.
  - Fast feedback, CI-ready, scales with your projects.
  - Open-source and free on GitHub

Thank you

Marius Elvegard
marius.elvegard@inventas.no
Inventas

inventas