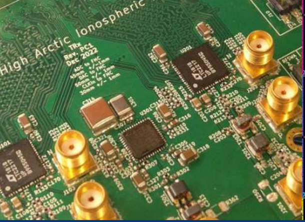
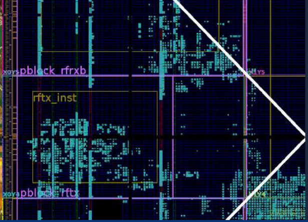
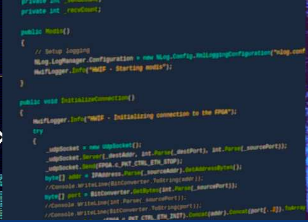
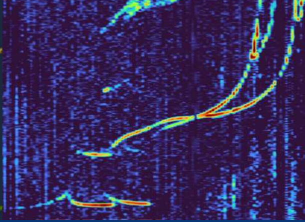






```
private int _sendCount;
private int _recvCount;

public Modis()
{
    // Setup logging
    NLog.LogManager.Configuration = new NLog.Config.XmlLoggingConfiguration("nlog.conf
    HwifLogger.Info("HWIF - Starting modis");
}

public void InitializeConnection()
{
    HwifLogger.Info("HWIF - Initializing connection to the FPGA");
    try
    {
        _udpSocket = new UdpSocket();
        _udpSocket.Server(_destAddr, int.Parse(_destPort), int.Parse(_sourcePort));
        _udpSocket.Send(FPGA.c_PKT_CTRL_ETH_STOP);
        byte[] addr = IPAddress.Parse(_sourceAddr).GetAddressBytes();
        //Console.WriteLine(BitConverter.ToString(addr));
        byte[] port = BitConverter.GetBytes(int.Parse(_sourcePort));
        //Console.WriteLine(int.Parse(_sourcePort));
        //Console.WriteLine(BitConverter.ToString(port));
                   (FPGA.c_PKT_CTRL_ETH_INIT).Concat(addr).Concat(port[..2]).ToArray
```



CAVEAT – a Framework for Context-Aware Verification, Emulation, and Training

Torsten Reuschel, M. Ferris, P. Trottier

FPGA Conference Europe 2025

FPGA Conference EUROPE

EST. 1785 UNB UNIVERSITY OF NEW BRUNSWICK

Electronics in Remote and Adverse Surroundings @ UNB

Emerging questions and ties at Department of Physics

- Quantum Sensing & Ultracold Matter Lab
- Radio and Space Physics Laboratory

New **ERAS** lab as of 2025

- Highly reliable, high accuracy measurement instrumentation in unconventional environments
- Infrastructure for assembly, integration, testing, calibration

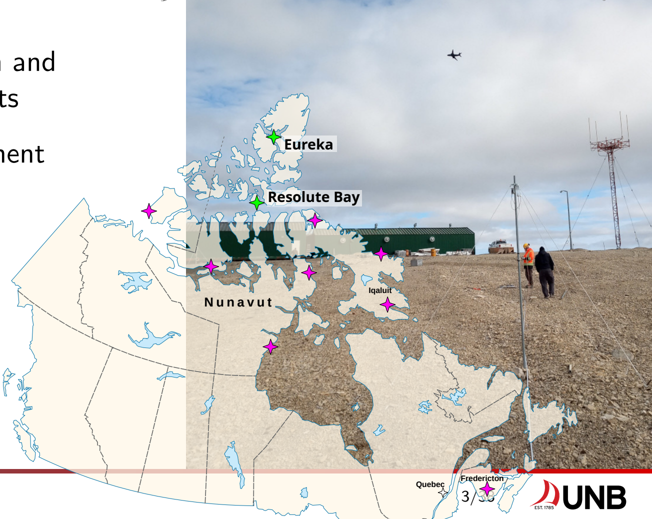
Campus image: UNB media services; map adapted from Hogweard, Public domain, Wikimedia Commons

Motivation

- Scientific remote sensing platform *sanimut* (currently ionospheric radar)
- Facilitate maintenance, reconfiguration and operating modes specific to experiments
- Operation subject to physical environment
  - radio environment
  - state of ionosphere

  impacts includes configuration of
  - analog frontend
  - signal processing

  ...and diverse interests of physicists.



Map adapted from Hogweard, Public domain, via Wikimedia Commons

Bird’s-Eye View of CAVEAT

A digital twin …

… built on and with existing tools

… contributing a physical verification framework to the community

Context-Aware Verification

- Create virtual environment for concept studies
- Insight into interdependency of components and their surrounding
- Operational risk mitigation

Emulation

- Facilitate root-cause analysis and predictive study
  beyond simulating a particular process
- Model-based core development and design of novel experiments
- Reduce time-to-field

Training

- Sandbox for student and research training

python powered

pytest

Forastero

cocotb

ICARUS

On Today’s Agenda

- Introduction to pytest
- Beyond pytest: static verification of design and operation specification
- Step-by-step introduction to cocotb
- Beyond cocotb: context-aware dynamic verification
- More examples from our lab: caveat in emulation and training
  (we’re scientists, technologists, and engineers with a knack for building tools)

pytest – Scalable Testing of Applications and Libraries

- Automate testing
  - Increase confidence in design – in an uncertain environment
  - Raise acceptance of changes – iterative improvement
  - Offer documentation and traceability – students come and go
  - Reduce worries and distraction
- Scope of testing: unit, integration, system – **software, gateware, and hardware!**
- Failing test ≠ faulty code ...it might be lack of understanding of DUT

pytest offers ...

- Automatic test discovery
- Parametrization
- Ability to combine with packages from Python ecosystem

Example Code and Test Definition

pytest

tests/test_adder.py

```python
import pytest

def adder(a: int, b: int) -> int:
  """Add two integers"""
  return a + b

@pytest.mark.parametrize(
  "invals, expect",
  [([3, 5], 8), ([2, 4], 6), ([6, 9], 42)])
def test_adder(invals, expect):
    assert adder(invals[0], invals[1]) == expect, \
      "Unexpected result"
```

1. Package import(s)
2. Code subject to testing
3. Test definition

   optional: parametrization

Initiating Tests

Command line

```
$ pytest tests/ -k "adder and not multiplier"
```

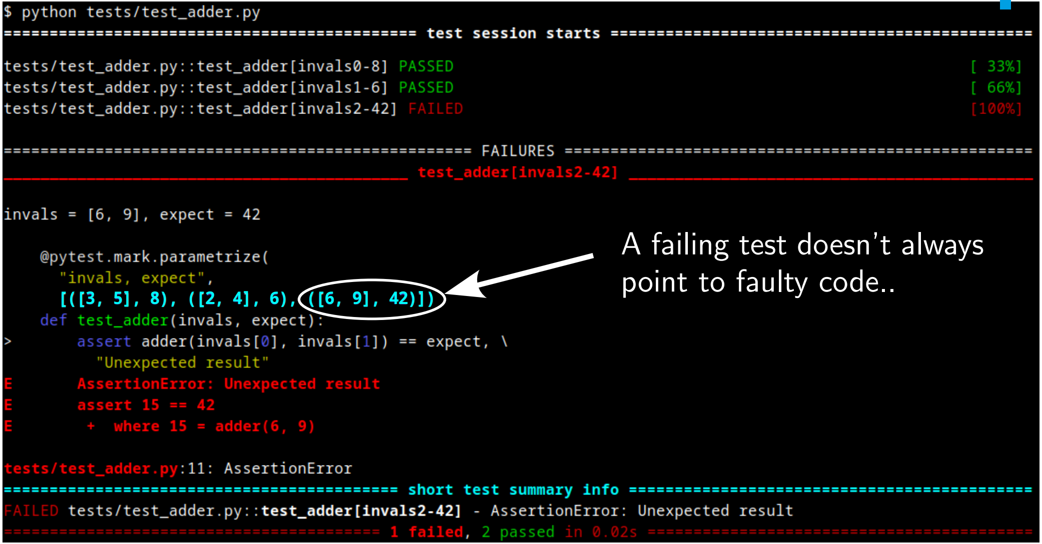
Programmatic call

```
pytest.main([
    'tests/',
    '-k adder and not multiplier',
  ])
```

A matter of preference and workflow..

- Interchangeable and simultaneous integration into workflows
- Options to include/exclude certain tests during automatic collection, ...
- Functions and corresponding test definition in same or nearby file
  OR separate directories for application and test

Example Test Output... Ooops!



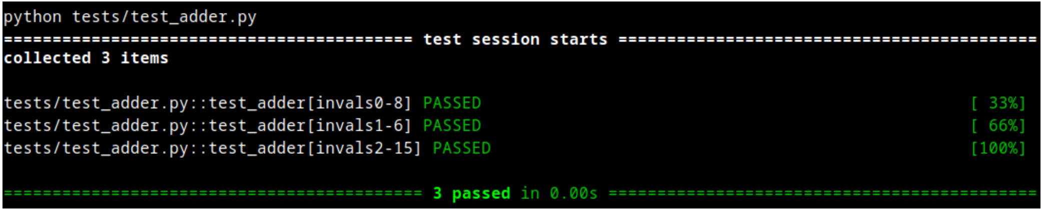
```
$ python tests/test_adder.py
============================================= test session starts =============================================

tests/test_adder.py::test_adder[invals0-8] PASSED                                                        [ 33%]
tests/test_adder.py::test_adder[invals1-6] PASSED                                                        [ 66%]
tests/test_adder.py::test_adder[invals2-42] FAILED                                                       [100%]

=================================================== FAILURES ==================================================
____________________________________________ test_adder[invals2-42] ___________________________________________

invals = [6, 9], expect = 42

    @pytest.mark.parametrize(
      "invals, expect",
      [([3, 5], 8), ([2, 4], 6), ([6, 9], 42)])
    def test_adder(invals, expect):
>       assert adder(invals[0], invals[1]) == expect, \
          "Unexpected result"
E       AssertionError: Unexpected result
E       assert 15 == 42
E        +  where 15 = adder(6, 9)

tests/test_adder.py:11: AssertionError
=========================================== short test summary info ===========================================
FAILED tests/test_adder.py::test_adder[invals2-42] - AssertionError: Unexpected result
========================================= 1 failed, 2 passed in 0.02s =========================================
```

A failing test doesn't always point to faulty code..

Example Fixed: All Pass

tests/test_adder.py

```
[...]

@pytest.mark.parametrize(
  "invals, expect",
  [([3, 5], 8), ([2, 4], 6), ([6, 9], ~~42~~ 15 )])
def test_adder(invals, expect):
    assert adder(invals[0], invals[1]) == expect, \
      "Unexpected result"
```

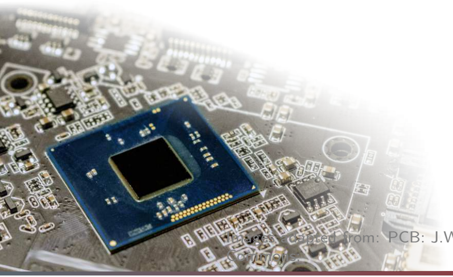


```
python tests/test_adder.py
========================================= test session starts ==========================================
collected 3 items

tests/test_adder.py::test_adder[invals0-8] PASSED                                                  [ 33%]
tests/test_adder.py::test_adder[invals1-6] PASSED                                                  [ 66%]
tests/test_adder.py::test_adder[invals2-15] PASSED                                                 [100%]

========================================== 3 passed in 0.00s ===========================================
```

CAVEAT Static Verification

Circuit design specification
(general definitions or project specific)

- Pin constraints
- Schematic or netlist
- Ohm's law



Images adapted from: PCB: J.Waterhouse/pexels.com #3665442; 'magnifier': http://www.simpleicon.com/, CC BY 3.0 via Wikimedia Commons.

CAVEAT Static Verification

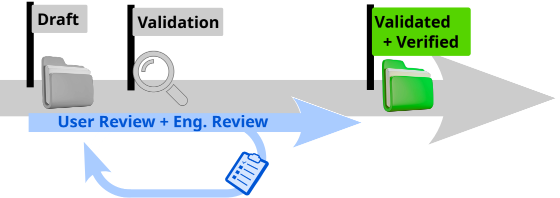
Circuit design specification
(general definitions or project specific)

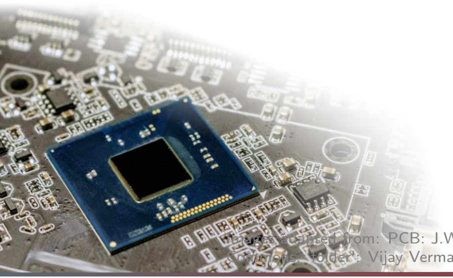
- Pin constraints
- Schematic or netlist
- Ohm's law

User input checking
(project specific)

- Syntax
- Valid operation sequence
- Radio license compatibility
- Concurrence of events

  e.g. power amplifier data + gate





Images adapted from: PCB: J.Waterhouse/pexels.com #3665442; 'magnifier': http://www.simpleicon.com/, CC BY 3.0 via Wikimedia Commons, 'folder': Vijay Verma, CC0, via Wikimedia Commons, 'clipboard': Emoji One, CC BY-SA 4.0 via Wikimedia Commons

CAVEAT Static Verification

Circuit design specification

(general definitions or project specific)

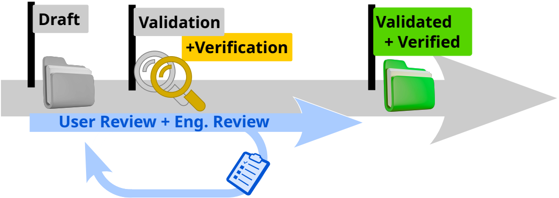
- Pin constraints
- Schematic or netlist
- Ohm's law

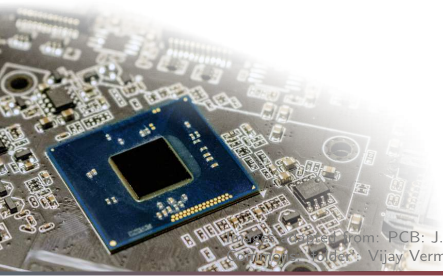
User input checking

(project specific)

- Syntax
- Valid operation sequence
- Radio license compatibility
- Concurrence of events

  e.g. power amplifier data + gate





Images adapted from: PCB: J.Waterhouse/pexels.com #3665442; 'magnifier': http://www.simpleicon.com/, CC BY 3.0 via Wikimedia Commons, 'folder': Vijay Verma, CC0, via Wikimedia Commons, 'clipboard': Emoji One, CC BY-SA 4.0 via Wikimedia Commons

CAVEAT Static Verification: Application Example

…with fixtures, docstring, and assertions.

caveat/tests/static/test_io_pin_all_constrained.py

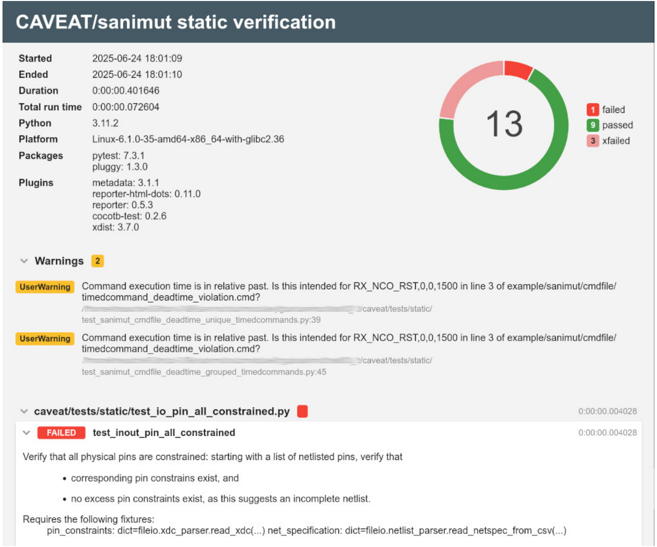
```python
def test_inout_pin_all_constrained(pin_constraints, net_specification):
    """Verify that all physical pins are constrained. Check that:
       - corresponding pin constrains exist, and
       - no excess pin constraints exist, as this suggests an incomplete netlist.
    """
    pins_constrained = set([pin['name'] for pin in pin_constraints])
    pins_netlisted = set(net_specification.keys())

    excess_constrained = pins_constrained - pins_netlisted
    assert excess_constrained == set([]), "constrained pin(s) unconnected or missing in netlist: {:}".format(' '.join(excess_constrained))

    excess_netlisted = pins_netlisted - pins_constrained
    assert excess_netlisted == set([]), "netlist contains unconstrained pins: {:}".format(' '.join(excess_netlisted))
```

CAVEAT Static Verification: Example Report

- **Failed** pin constraints check due to incomplete netlist
  report: test description and failed assertion
- **Warning** due to implicit operation sequence *

Start with predefined tests from framework… and add project-specific tests for past and future mishaps.

* Mismatching pin events of power amplifier data and gate results in error.



CAVEAT/sanimut static verification

Started	2025-06-24 18:01:09
Ended	2025-06-24 18:01:10
Duration	0:00:00.401646
Total run time	0:00:00.072604
Python	3.11.2
Platform	Linux-6.1.0-35-amd64-x86_64-with-glibc2.36
Packages	pytest: 7.3.1 pluggy: 1.3.0
Plugins	metadata: 3.1.1 reporter-html-dots: 0.11.0 reporter: 0.5.3 cocotb-test: 0.2.6 xdist: 3.7.0

Warnings 2

UserWarning Command execution time is in relative past. Is this intended for RX_NCO_RST,0,0,1500 in line 3 of example/sanimut/cmdfile/timedcommand_deadtime_violation.cmd?
/[redacted]/caveat/tests/static/test_sanimut_cmdfile_deadtime_unique_timedcommands.py:39

UserWarning Command execution time is in relative past. Is this intended for RX_NCO_RST,0,0,1500 in line 3 of example/sanimut/cmdfile/timedcommand_deadtime_violation.cmd?
[redacted]/caveat/tests/static/test_sanimut_cmdfile_deadtime_grouped_timedcommands.py:45

caveat/tests/static/test_io_pin_all_constrained.py 🟥 0:00:00.004028

FAILED test_inout_pin_all_constrained 0:00:00.004028

Verify that all physical pins are constrained: starting with a list of netlisted pins, verify that

- corresponding pin constrains exist, and
- no excess pin constraints exist, as this suggests an incomplete netlist.

Requires the following fixtures:
pin_constraints: dict=fileio.xdc_parser.read_xdc(...) net_specification: dict=fileio.netlist_parser.read_netspec_from_csv(...)

COroutine based COsimulation TestBench

COroutine cooperative scheduling, interacts with a simulator's event scheduler

CO-simulation cocotb's scheduler integrates with simulator's event scheduler via *triggers*

TestBench test environment and user interfacing via simulation *handles*

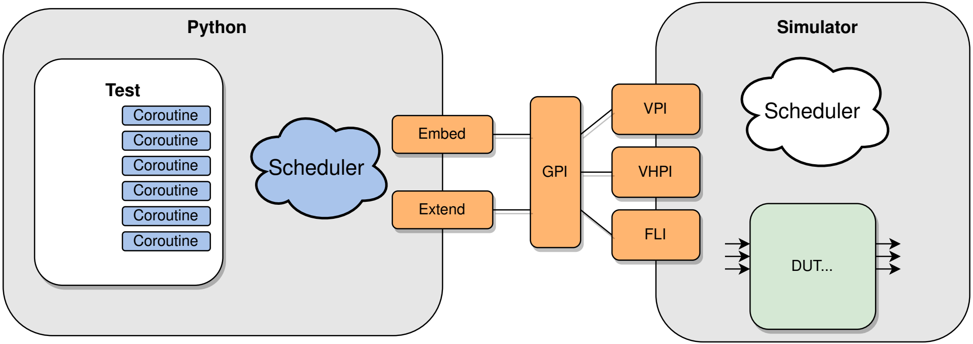


Image source: `https://docs.cocotb.org/en/stable/`

COroutine based COsimulation TestBench (cont'd)

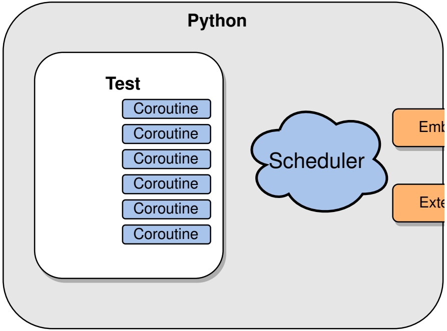


Image source: `https://docs.cocotb.org/en/stable/`

- Designed to reduce overhead when creating test
  Design re-use and randomized testing like UVM, implemented in Python rather than SystemVerilog
- With cocotb, VHDL/Verilog/SystemVerilog normally used for design, not the testbench...this is where Python comes in, e.g. for automatic test discovery
- It's free
  We're still a small group with limited in budget, faced with vertical integration
- And yet: tremendous ecosystem for building testbench and models (coroutines)
  - General: math, plotting, science, ..
  - cocotb-specific: bus drivers/monitors, e.g. cocotbext

Example Code and Test Definition

Let's define and test an adder in three distinct parts:

reference model .. cocotb 'glue' .. RTL code

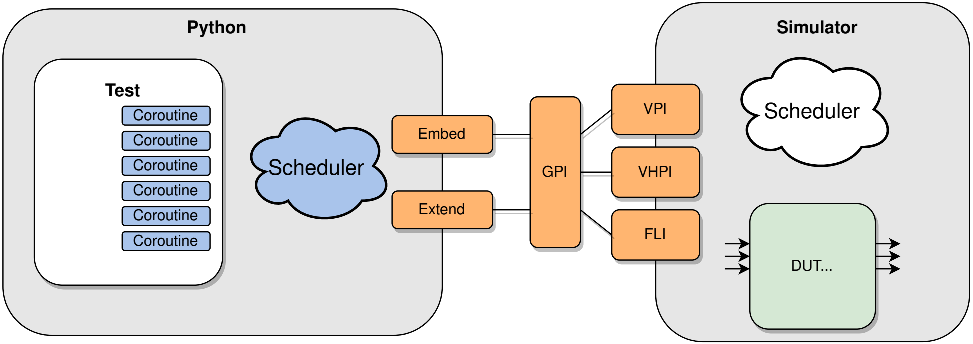
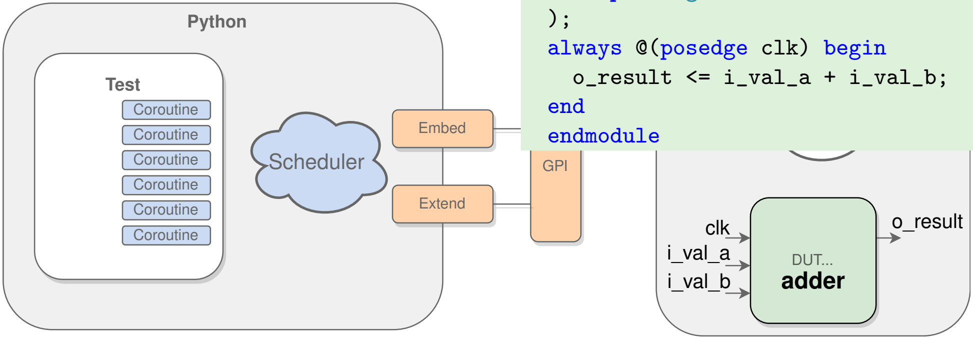


Image source: `https://docs.cocotb.org/en/stable/`

Example Code and Test Definition (cont'd)



```verilog
module adder #(parameter c_WIDTH = 4) (
  input wire clk,
  input wire [c_WIDTH-1:0] i_val_a,
  input wire [c_WIDTH-1:0] i_val_b,
  output reg [c_WIDTH-1:0] o_result
);
always @(posedge clk) begin
  o_result <= i_val_a + i_val_b;
end
endmodule
```

Image adapted from `https://docs.cocotb.org/en/stable/`

Example Code and Test Definition (cont'd)

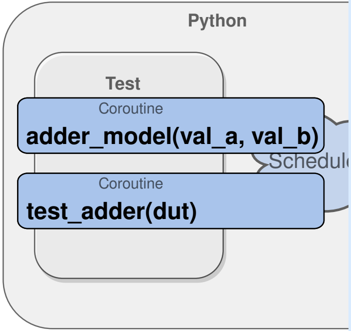


Image adapted from `https://docs.cocotb.org/en/stable/`

```python
import cocotb
from cocotb.clock import Clock
from cocotb.triggers import RisingEdge

def adder_model(val_a: int, val_b: int) -> int:
    """Adder reference model"""
    return val_a + val_b

@cocotb.test()
async def test_adder(dut):
    """Test coroutine"""
    # generate clock and start clock as coroutine
    cocotb.start_soon(Clock(dut.clk, 10, 'ns').start())
    # set input signals
    dut.i_val_a.value = 2
    dut.i_val_b.value = 3
    # compare output after two cycles with model
    expect = adder_model(2, 3)
    for _ in range(2):
        await RisingEdge(dut.clk)
    assert dut.o_result.value == expect, "Unexpected result"
```

Alternative Ways of Running a Test

Makefile

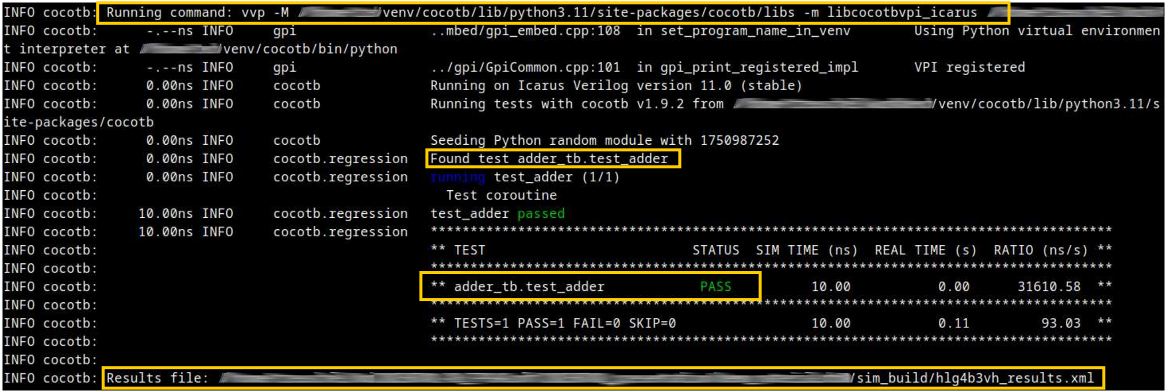
```
# source file(s)
SOURCES = rtl/adder.v
# name of the toplevel module
TOPLEVEL = adder
# basename of the Python test file
MODULE = adder_tb
# cocotb's make rules take care of simulator
↪  setup
include $(shell cocotb-config
↪  --makefiles)/Makefile.sim
```

Programmatic call

```
from cocotb_test.simulator import run
run(
    verilog_sources = ['rtl/adder.v',],
    toplevel = 'adder',
    module = 'adder_tb',
  )
```

As with pytest: interchangeable and concurrent integration into workflows

Example Single Test Output: Pass



```
INFO cocotb: Running command: vvp -M [redacted]/venv/cocotb/lib/python3.11/site-packages/cocotb/libs -m libcocotbvpi_icarus [redacted]
INFO cocotb:      -.--ns INFO     gpi                 ..mbed/gpi_embed.cpp:108  in set_program_name_in_venv        Using Python virtual environmen
t interpreter at [redacted]/venv/cocotb/bin/python
INFO cocotb:      -.--ns INFO     gpi                 ../gpi/GpiCommon.cpp:101  in gpi_print_registered_impl       VPI registered
INFO cocotb:      0.00ns INFO     cocotb              Running on Icarus Verilog version 11.0 (stable)
INFO cocotb:      0.00ns INFO     cocotb              Running tests with cocotb v1.9.2 from [redacted]/venv/cocotb/lib/python3.11/s
ite-packages/cocotb
INFO cocotb:      0.00ns INFO     cocotb              Seeding Python random module with 1750987252
INFO cocotb:      0.00ns INFO     cocotb.regression   Found test adder_tb.test_adder
INFO cocotb:      0.00ns INFO     cocotb.regression   running test_adder (1/1)
INFO cocotb:                                            Test coroutine
INFO cocotb:     10.00ns INFO     cocotb.regression   test_adder passed
INFO cocotb:     10.00ns INFO     cocotb.regression   **************************************************************************************
INFO cocotb:                                          ** TEST                          STATUS  SIM TIME (ns)  REAL TIME (s)  RATIO (ns/s) **
INFO cocotb:                                          **************************************************************************************
INFO cocotb:                                          ** adder_tb.test_adder            PASS          10.00           0.00      31610.58  **
INFO cocotb:                                          **************************************************************************************
INFO cocotb:                                          ** TESTS=1 PASS=1 FAIL=0 SKIP=0                 10.00           0.11         93.03  **
INFO cocotb:                                          **************************************************************************************
INFO cocotb:
INFO cocotb: Results file: [redacted]/sim_build/hlg4b3vh_results.xml
```

- *vvp* is iverilog's simulation engine
- Outputs: terminal, xml-file

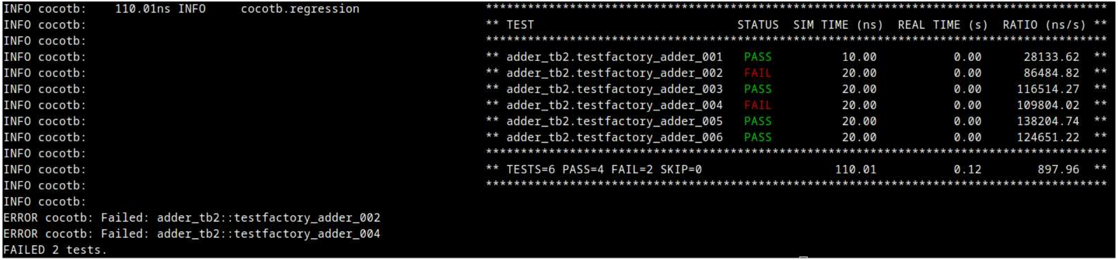
cocotb Parametrized Test

- *TestFactory* generates parametrized tests
- cocotb 2.0 introduces decorator (pytest look and feel)

```
@cocotb.test
@cocotb.parametrize(
  value_a=[2,1,0],
  value_b=[3,15],
)
async def test_adder(
    dut,
    value_a,
    value_b):
  ...
```

```
from cocotb.regression import TestFactory

async def testfactory_adder(dut, value_a, value_b):
    """Testfactory for adder"""
    # generate clock and start clock as coroutine
    cocotb.start_soon(Clock(dut.clk, 10, 'ns').start())
    # update input signals
    dut.i_val_a.value = value_a
    dut.i_val_b.value = value_b
    #obtain reference result
    expect = adder_model(value_a, value_b)
    # expect correct output after two cycles
    for _ in range(2):
        await RisingEdge(dut.clk)
    assert dut.o_result.value == expect, "Unexpected result"

tf = TestFactory(testfactory_adder)
tf.add_option('value_a', [2, 1, 0])
tf.add_option('value_b', [3, 15])
tf.generate_tests()
```

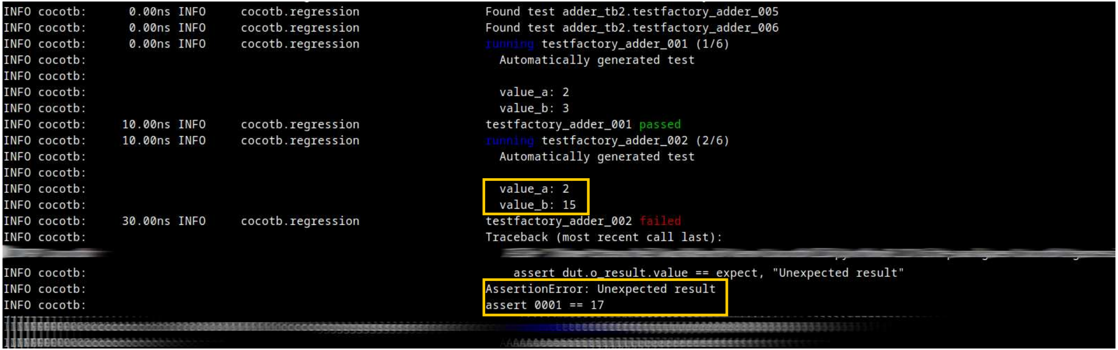
cocotb Parametrized Test Output.... Ooops!



```
INFO cocotb:    110.01ns INFO     cocotb.regression                    ******************************************************************************************
INFO cocotb:                                                           ** TEST                                  STATUS  SIM TIME (ns)  REAL TIME (s)  RATIO (ns/s) **
INFO cocotb:                                                           ******************************************************************************************
INFO cocotb:                                                           ** adder_tb2.testfactory_adder_001        PASS          10.00           0.00      28133.62  **
INFO cocotb:                                                           ** adder_tb2.testfactory_adder_002        FAIL          20.00           0.00      86484.82  **
INFO cocotb:                                                           ** adder_tb2.testfactory_adder_003        PASS          20.00           0.00     116514.27  **
INFO cocotb:                                                           ** adder_tb2.testfactory_adder_004        FAIL          20.00           0.00     109804.02  **
INFO cocotb:                                                           ** adder_tb2.testfactory_adder_005        PASS          20.00           0.00     138204.74  **
INFO cocotb:                                                           ** adder_tb2.testfactory_adder_006        PASS          20.00           0.00     124651.22  **
INFO cocotb:                                                           ******************************************************************************************
INFO cocotb:                                                           ** TESTS=6 PASS=4 FAIL=2 SKIP=0                        110.01           0.12        897.96  **
INFO cocotb:                                                           ******************************************************************************************
INFO cocotb:
ERROR cocotb: Failed: adder_tb2::testfactory_adder_002
ERROR cocotb: Failed: adder_tb2::testfactory_adder_004
FAILED 2 tests.
```

- TestFactory combines all parameters:
  $\text{value\_a} \times \text{value\_b} = \{2, 1, 0\} \times \{4, 15\} \mapsto \mathbf{(2, 4)}, \mathbf{(2, 15)}, \mathbf{(1, 4)}, \mathbf{(1, 15)}, \mathbf{(0, 4)}, \mathbf{(0, 15)}$
- What's wrong with cases 2 and 4?

cocotb Parametrized Test Output.... Ooops!

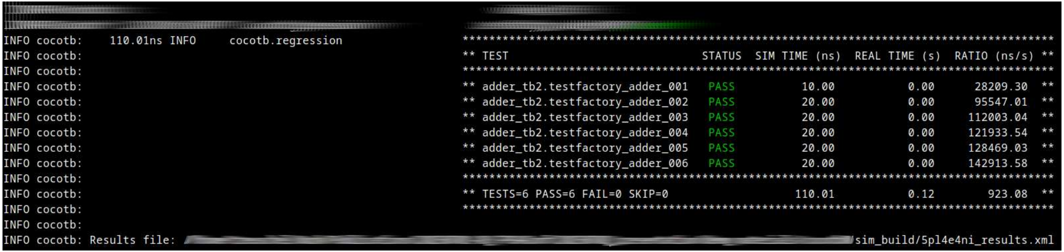


```
INFO cocotb:      0.00ns INFO     cocotb.regression                  Found test adder_tb2.testfactory_adder_005
INFO cocotb:      0.00ns INFO     cocotb.regression                  Found test adder_tb2.testfactory_adder_006
INFO cocotb:      0.00ns INFO     cocotb.regression                  running testfactory_adder_001 (1/6)
INFO cocotb:                                                           Automatically generated test
INFO cocotb:
INFO cocotb:                                                           value_a: 2
INFO cocotb:                                                           value_b: 3
INFO cocotb:     10.00ns INFO     cocotb.regression                  testfactory_adder_001 passed
INFO cocotb:     10.00ns INFO     cocotb.regression                  running testfactory_adder_002 (2/6)
INFO cocotb:                                                           Automatically generated test
INFO cocotb:
INFO cocotb:                                                           value_a: 2
INFO cocotb:                                                           value_b: 15
INFO cocotb:     30.00ns INFO     cocotb.regression                  testfactory_adder_002 failed
INFO cocotb:                                                         Traceback (most recent call last):

INFO cocotb:                                                             assert dut.o_result.value == expect, "Unexpected result"
INFO cocotb:                                                         AssertionError: Unexpected result
INFO cocotb:                                                         assert 0001 == 17
```

- TestFactory combines all parameters:
  $\text{value\_a} \times \text{value\_b} = \{2, 1, 0\} \times \{4, 15\} \mapsto \mathbf{(2, 4)}, \mathbf{(2, 15)}, \mathbf{(1, 4)}, \mathbf{(1, 15)}, \mathbf{(0, 4)}, \mathbf{(0, 15)}$
- What's wrong with cases 2 and 4?

Example Fixed: All Pass

```
module adder #(parameter c_WIDTH = 4) (
  input wire clk,
  input wire [c_WIDTH-1:0] i_val_a,
  input wire [c_WIDTH-1:0] i_val_b,
  output reg [c_WIDTH~~-1~~:0] o_result
);
```

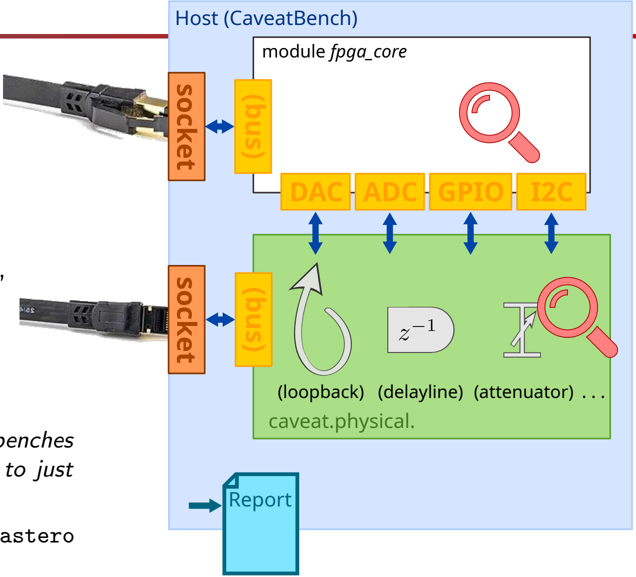


```
[redacted]
INFO cocotb:    110.01ns INFO     cocotb.regression                         **************************************************************************************
INFO cocotb:                                                                ** TEST                               STATUS  SIM TIME (ns)  REAL TIME (s)  RATIO (ns/s) **
INFO cocotb:                                                                **************************************************************************************
INFO cocotb:                                                                ** adder_tb2.testfactory_adder_001   PASS          10.00           0.00      28209.30  **
INFO cocotb:                                                                ** adder_tb2.testfactory_adder_002   PASS          20.00           0.00      95547.01  **
INFO cocotb:                                                                ** adder_tb2.testfactory_adder_003   PASS          20.00           0.00     112003.04  **
INFO cocotb:                                                                ** adder_tb2.testfactory_adder_004   PASS          20.00           0.00     121933.54  **
INFO cocotb:                                                                ** adder_tb2.testfactory_adder_005   PASS          20.00           0.00     128469.03  **
INFO cocotb:                                                                ** adder_tb2.testfactory_adder_006   PASS          20.00           0.00     142913.58  **
INFO cocotb:                                                                **************************************************************************************
INFO cocotb:                                                                ** TESTS=6 PASS=6 FAIL=0 SKIP=0                   110.01           0.12        923.08  **
INFO cocotb:                                                                **************************************************************************************
INFO cocotb:
INFO cocotb: Results file: [redacted]/sim_build/5pl4e4ni_results.xml
```

CAVEAT

- Foundation: pytest, cocotb, Forastero
- Attach physical environment (simulated or real)
- Interfacing: testbench, network socket, serial, ...
- Reporting: interactive HTML

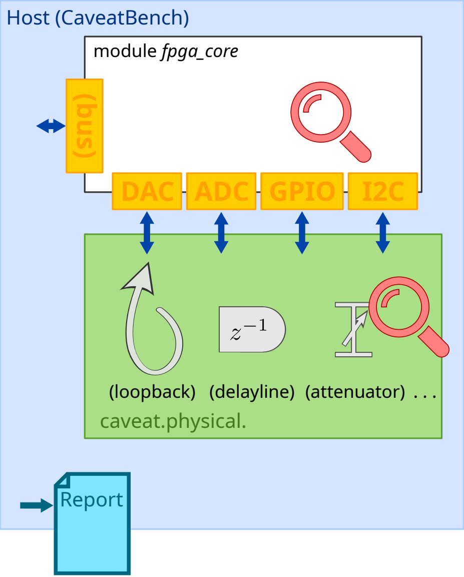
> "*Forastero is a library for writing better testbenches with cocotb, inspired by UVM but distilling it to just the most useful parts.*"
>
> —`https://github.com/Intuity/forastero`



CAVEAT

- Foundation: pytest, cocotb, Forastero
- Attach physical environment (simulated or real)
- Interfacing: testbench, network socket, serial, ...
- Reporting: interactive HTML

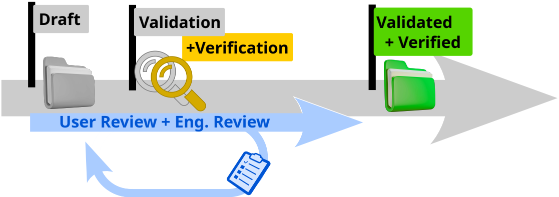
> *"Forastero is a library for writing better testbenches with cocotb, inspired by UVM but distilling it to just the most useful parts."*
>
> —`https://github.com/Intuity/forastero`



CAVEAT Dynamic Verification

Functional specification
(RTL, general or project specific)

- Interfaces
- State machines
- ...



User-/soft-defined operation
(project specific, some overlap w/ static)

- Interaction with peripherals
  - Practicability of sequences
  - Concurrence of events
  - Over-/underflow recovery
- Meaningfulness of input/output signals
- Radio license compatibility
- Compatibility with deployed system
  bitstream, circuit boards, peripherals

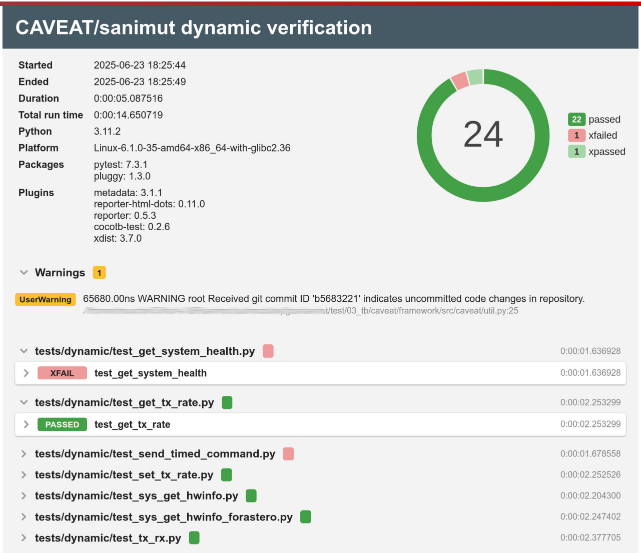
Images adapted from: 'magnifier': http://www.simpleicon.com/, CC BY 3.0 via Wikimedia Commons; 'folder': Vijay Verma, CC0, via Wikimedia Commons, 'clipboard': Emoji One, CC BY-SA 4.0 via Wikimedia Commons

CAVEAT Dynamic Verification: Example Report

- Warning
  - Truncated input: 16 bit → 12 bit
  - Code change, incl. user-defined command files, code files
- XFail ...because some things should not be viable, e.g. receive signal before physical propagation delay has passed

Feedback to user

- Operational constraints
  e.g. resource availability, concurrence, dead-times
- Physical limitations
  e.g. voltage rating, delay, tx power, ..
- Illustrate sequence for ease of review



CAVEAT/sanimut dynamic verification

Started	2025-06-23 18:25:44
Ended	2025-06-23 18:25:49
Duration	0:00:05.087516
Total run time	0:00:14.650719
Python	3.11.2
Platform	Linux-6.1.0-35-amd64-x86_64-with-glibc2.36
Packages	pytest: 7.3.1 pluggy: 1.3.0
Plugins	metadata: 3.1.1 reporter-html-dots: 0.11.0 reporter: 0.5.3 cocotb-test: 0.2.6 xdist: 3.7.0

22	passed
1	xfailed
1	xpassed

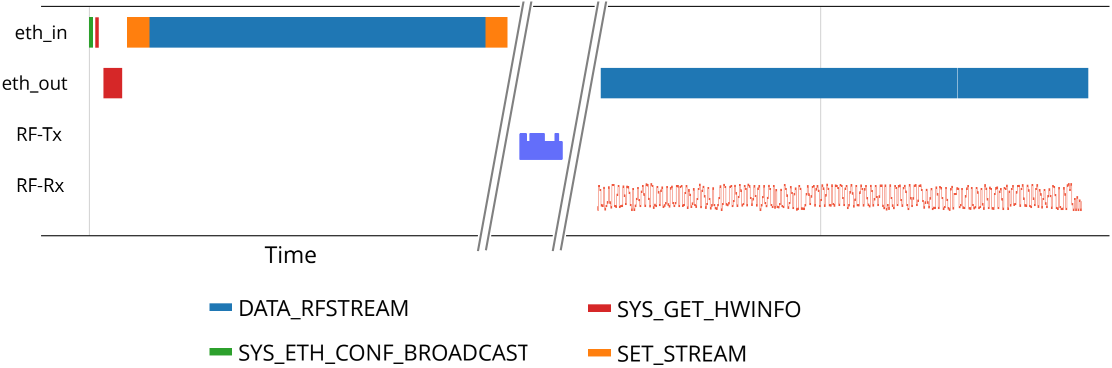
Warnings 1

UserWarning 65680.00ns WARNING root Received git commit ID 'b5683221' indicates uncommitted code changes in repository.
[redacted]t/test/03_tb/caveat/framework/src/caveat/util.py:25

tests/dynamic/test_get_system_health.py	🟥	0:00:01.636928
XFAIL	test_get_system_health	0:00:01.636928
tests/dynamic/test_get_tx_rate.py	🟩	0:00:02.253299
PASSED	test_get_tx_rate	0:00:02.253299
tests/dynamic/test_send_timed_command.py	🟥	0:00:01.678558
tests/dynamic/test_set_tx_rate.py	🟩	0:00:02.252526
tests/dynamic/test_sys_get_hwinfo.py	🟩	0:00:02.204300
tests/dynamic/test_sys_get_hwinfo_forastero.py	🟩	0:00:02.247402
tests/dynamic/test_tx_rx.py	🟩	0:00:02.377705

CAVEAT Dynamic Verification: Example Report (cont’d)

- **Integrate validation** (user) and verification (dev)
- Example: review Ethernet I/O, timed commands, experiment sequence



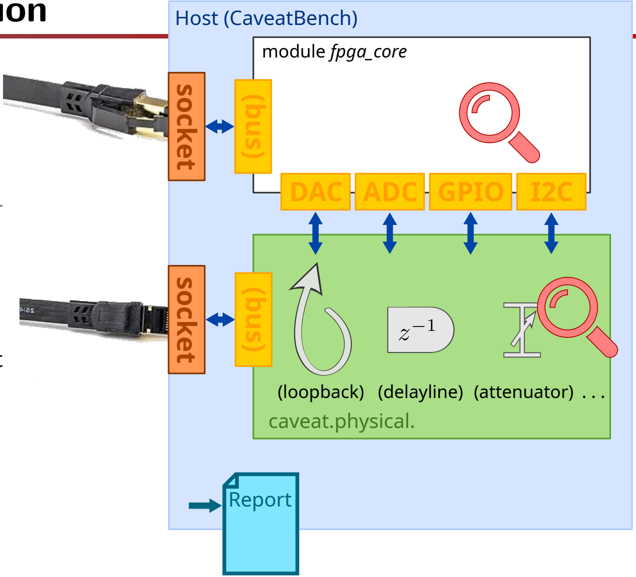
(Time scales of RF-Tx/RF-RX were compressed to fit this screen; relative sequencing unchanged.)

CAVEAT Functional Emulation

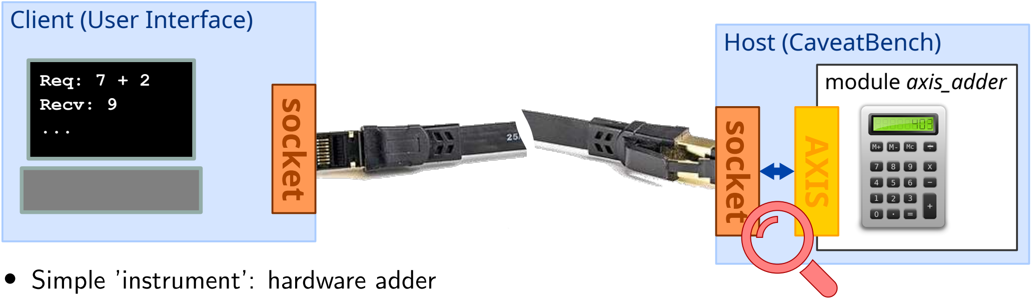
- Leverage developer tools for a (permanent) virtual instance
- Open-source framework compatible with closed-source code, e.g. ionospheric radar
- Simulate **RTL** + surrogate **environment**

Use cases

- Concurrent, yet independent development of hardware and software (API, controls)
- Developing new experiments, operational modes
- Study design parameter including impact of original user level software



Emulation Example: Adder with Network Interface



```
Req: 7 + 2
Recv: 9
...
```

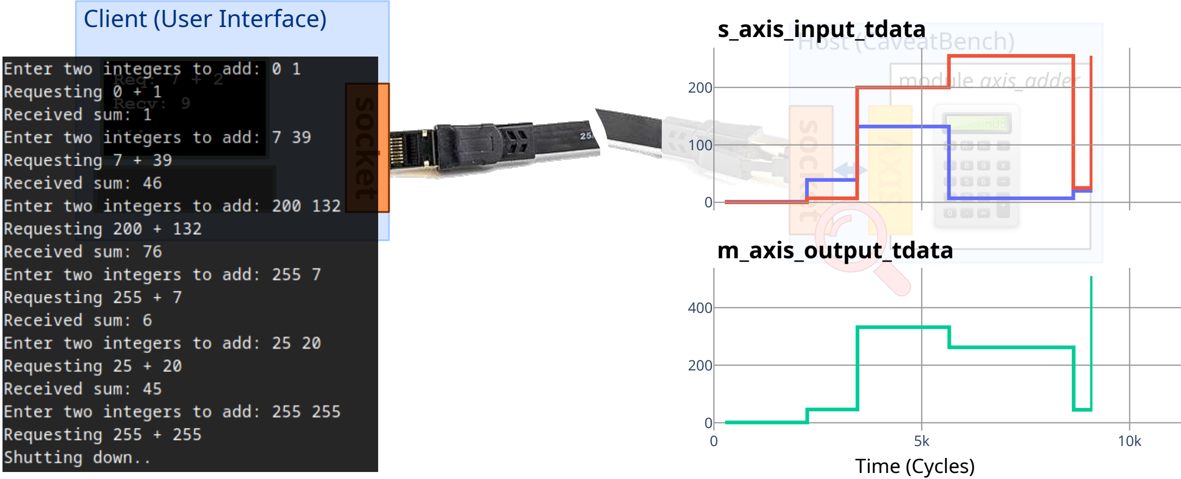
- Simple 'instrument': hardware adder
  Code is available in caveat repo as application example
- Custom, project specific parts:
  - User interfacing client
  - Module *axis_adder*
  - CaveatBench object: connect parts, monitor interfaces and operation

Images adapted from: 'calculator': Breathe Icon Team, CC BY-SA 3.0 via Wikimedia Commons; 'cable': R.Spekking CC BY-SA 4.0 via Wikimedia Commons; 'magnifier': http://www.simpleicon.com/, CC BY 3.0 via Wikimedia Commons

Emulation Example: Adder with Network Interface (cont'd)

```python
from cocotb.clock import Clock
from cocotb.triggers import RisingEdge
from caveat import augmented_handle
from caveat.caveatbench import CaveatBench

@cocotb.test()
async def run_network_adder(dut):
    tb_env = CaveatBench(dut)
    cocotb.start_soon(
        Clock(dut.clk, 8, units='ns').start())
    await tb_env.init_monitor('s_axis_tdata',
    ↪   'clk', callback=parse_input_frame)
    await tb_env.init_monitor('m_axis_tdata',
    ↪   'clk')
    dut.create_interface_socket_to_axis(
        remote_address = '127.0.0.1',
        remote_port = 20000,
        local_port = 20002,
        axis_bus_module_input = 's_axis',
        axis_bus_module_output = 'm_axis')
    #start simulation
    try:
        while True:
            await RisingEdge(dut.clk)
            #exit on magic number
            if dut.s_axis_tdata.value ==
            ↪   65535:
                break
    finally:
        tb_env.generate_plot()


if __name__ == "__main__":
    run(module = 'run_adder_host',
        verilog_sources =
        ↪   ['rtl/axis_adder.v'],
        toplevel = "axis_adder",
        timescale  = "1ns/1ps",
    )
```

Emulation Example: Adder with Network Interface (cont'd)



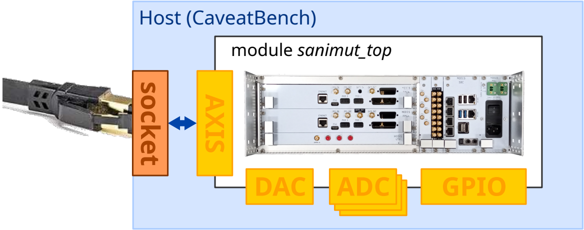
```
Enter two integers to add: 0 1
Requesting 0 + 1
Received sum: 1
Enter two integers to add: 7 39
Requesting 7 + 39
Received sum: 46
Enter two integers to add: 200 132
Requesting 200 + 132
Received sum: 76
Enter two integers to add: 255 7
Requesting 255 + 7
Received sum: 6
Enter two integers to add: 25 20
Requesting 25 + 20
Received sum: 45
Enter two integers to add: 255 255
Requesting 255 + 255
Shutting down..
```

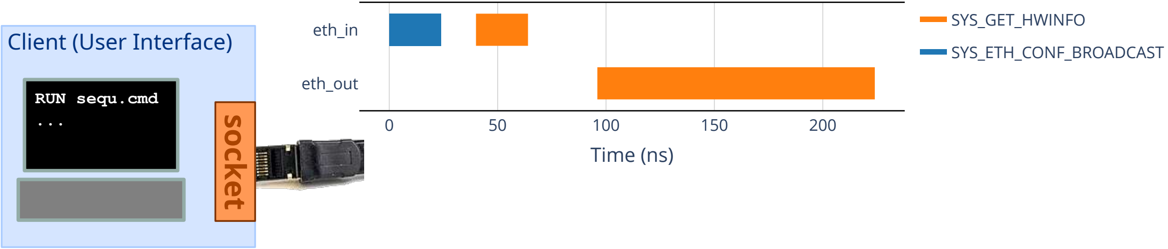
Images adapted from: 'calculator': Breathe Icon Team, CC BY-SA 3.0 via Wikimedia Commons; 'cable': R.Spekking CC BY-SA 4.0 via Wikimedia Commons; 'magnifier': http://www.simpleicon.com/, CC BY 3.0 via Wikimedia Commons

Emulation: Instrument Interaction

Read out hardware revision via user software

- Original software with command file
  (or interactive mode)
- Emulated design delivers code revision
  (actual information, adjusted to scope)

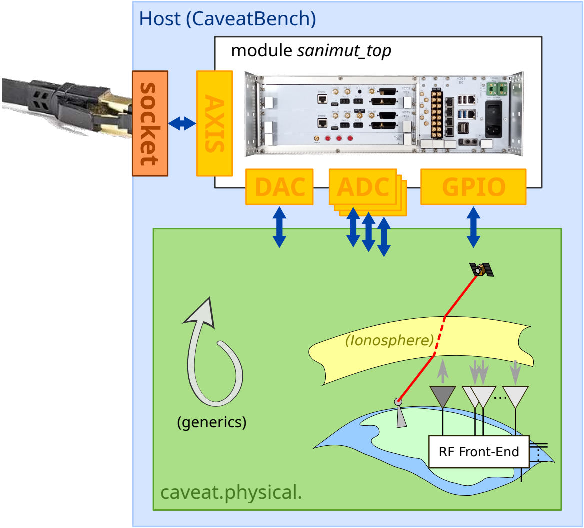


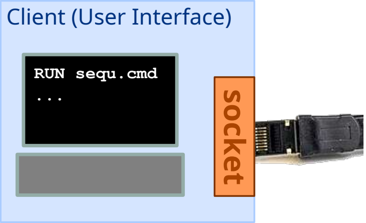


```
RUN sequ.cmd
...
```

Emulation: Instrument in Context

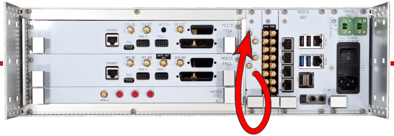
- Let's add some physically meaningful *context*..
- Simple example: tx → rx loopback





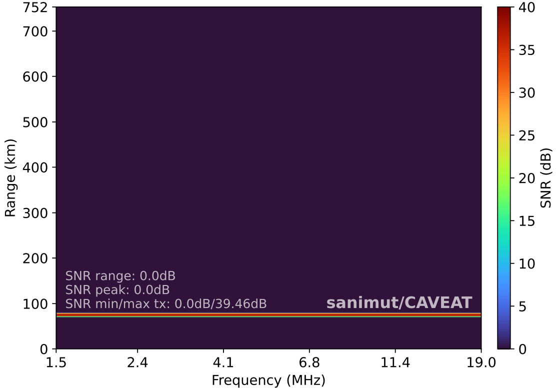
```
RUN sequ.cmd
...
```

Emulation: Instrument in Context



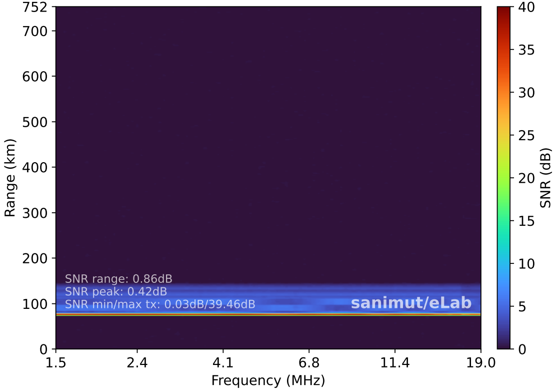
Emulated Design + Ideal Front-End/Cable

(constant range across frequencies, peak SNR)



Real Hardware + Real Cable

(artifacts in freq-range diagram)

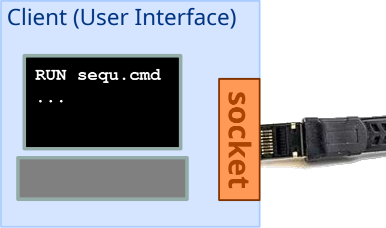


CAVEAT in Training

- Communicating technical design and function to disciplinary experts can be tough
- Regular onboarding of students
  project needs ≠ need to match individual experience and background

Our Approach

- Focus on experience, learning how to use an instrument in context
  e.g. investigating parameters of an experiment
- Adopt dev-tools in support of disciplinary research
  need to be mindful of overhead
- Bonus: guidance and tools for early project phase
  informing requirements engineering et al.



```
RUN sequ.cmd
...
```

Wrapping Up

- Address emerging entry barrier for research environments
  before: **availability** of technology, now: **accessibility** of technology

- Expand on variety of Python packages, added value through combination
  e.g. asserting and logging physical/mixed signal peripherals
  - Focus on building working tools with intend to give back to community
  - Anticipate to leverage (and support) development of **Forastero**

- Current use: ionospheric radar, learning/teaching framework
  anticipate expansion to: steered timing reference, environmental monitoring, laser control (QSUM)

Thank you for your attention

Get the sources, get started, get in touch:
`https://github.com/ERAS-Research/`

Electronics in Remote and Adverse Surroundings
`https://ERAS.ca`

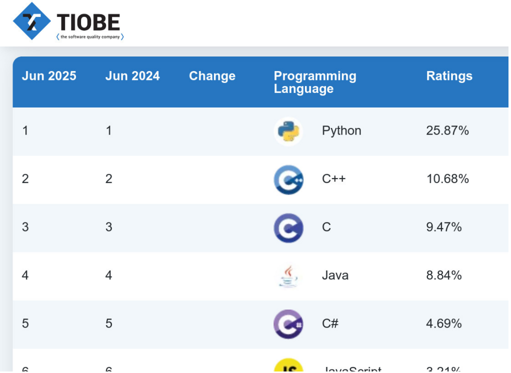
Prof. Dr.-Ing. Torsten Reuschel
✉ torsten.reuschel@unb.ca
🌍 https://eras.ca
☎ +1-506-452-6031

University of New Brunswick, Department of Physics
8 Bailey Dr. (Room 220), Fredericton, NB E3B 5A3, Canada

Appendix

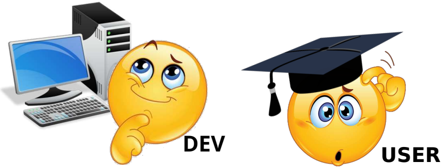
Beyond Verified Function: Involved People and Skills

- Verification is software
- Improve communication (via 'tangibles')
- Python is a popular language (ease of onboarding staff/students)
- Tremendous ecosystem (we are still a small group with limited budget)
- Cross-platform support (we opt for Linux)



Jun 2025	Jun 2024	Change	Programming Language	Ratings
1	1		Python	25.87%
2	2		C++	10.68%
3	3		C	9.47%
4	4		Java	8.84%
5	5		C#	4.69%
6	6		JavaScript	3.21%

Source: tiobe.com, accessed on June 25, 2025



Images: 'emoticon confused' (#2847), 'emoticon think' (#2982), 'hat' (#53868), 'computer' (#50654) via clipartix.com

Community Extensions

Efficient verification thanks to abstracted interfaces:

- Common interfaces available as *cocotbext*
- AXI(S), Wishbone, UART, I2C, .. integrate based on `cocotb.queue.Queue`

```python
from cocotbext.axi import AxiStreamBus, AxiStreamSource, AxiStreamMonitor

@cocotb.test
async def test_interface(dut)
  cocotb.start_soon(Clock(dut.clk, 5, units='ns').start())
  src = AxiStreamSource(AxiStreamBus.from_prefix(dut, 'axis'), dut.clk, dut.rst)
  mon = AxiStreamMonitor(AxiStreamBus.from_prefix(dut, 'axis'), dut.clk, dut.rst)
  #initialize transfer and probe monitor
  await src.send(bytes(test_data))
  mon_data = await mon.recv()
  #check transfer
  assert mon_rx.tdata == test_data, "Invalid AXIS transfer detected"
```