UNIVERSITÄT BONN

SI LAB
Silizium Labor Bonn

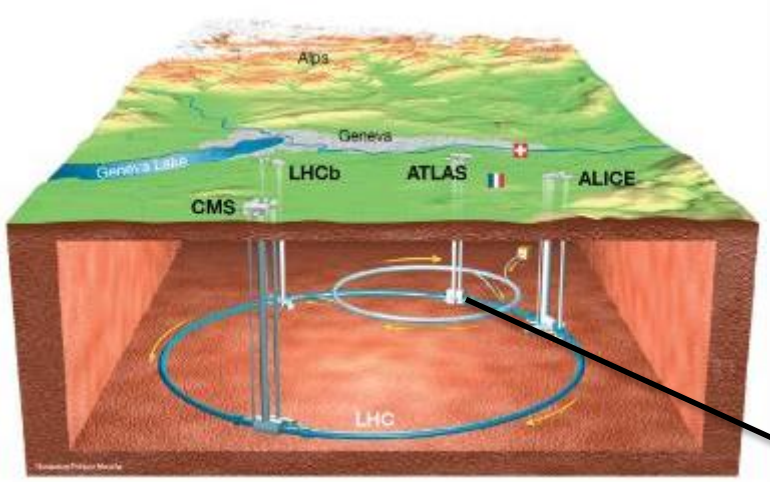
Testing "the Python way" with cocotb and pytest

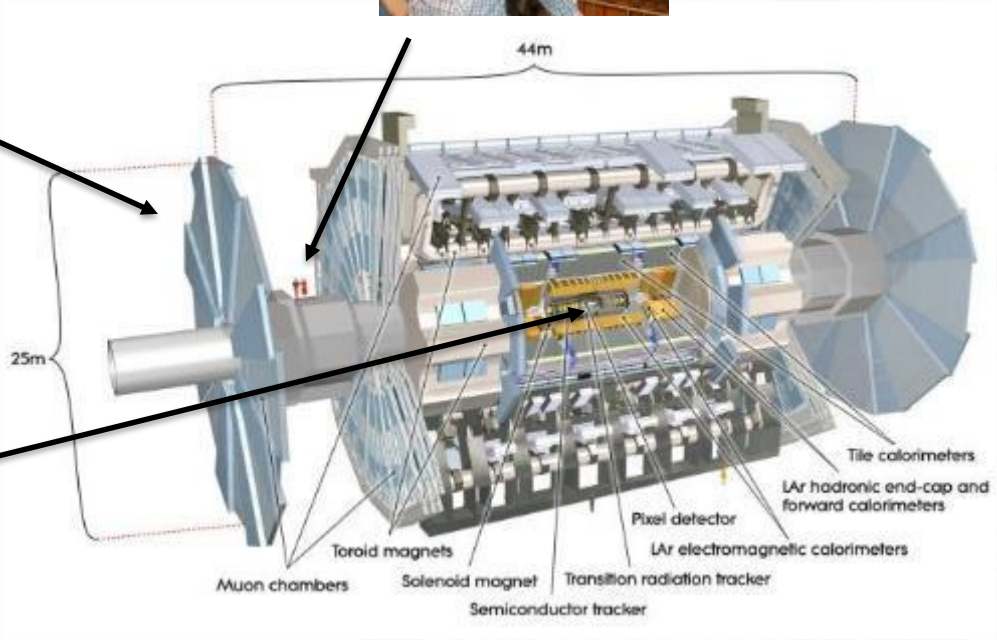
Tomasz Hemperek

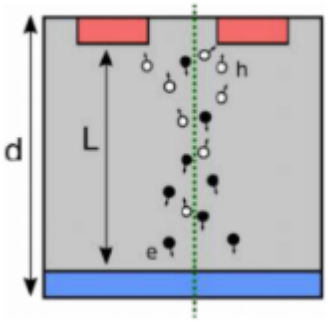
CHIPS ALLIANCE

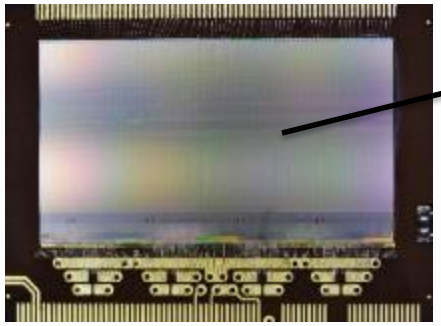
Workshop on Open Source Design Verification / 2019 / Munich

who am I









Neither software nor verification is my field of expertise …

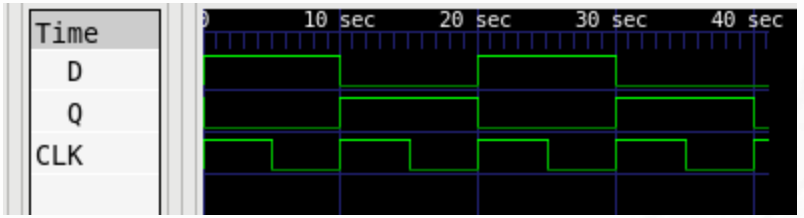
what is cocotb

UNIVERSITÄT BONN

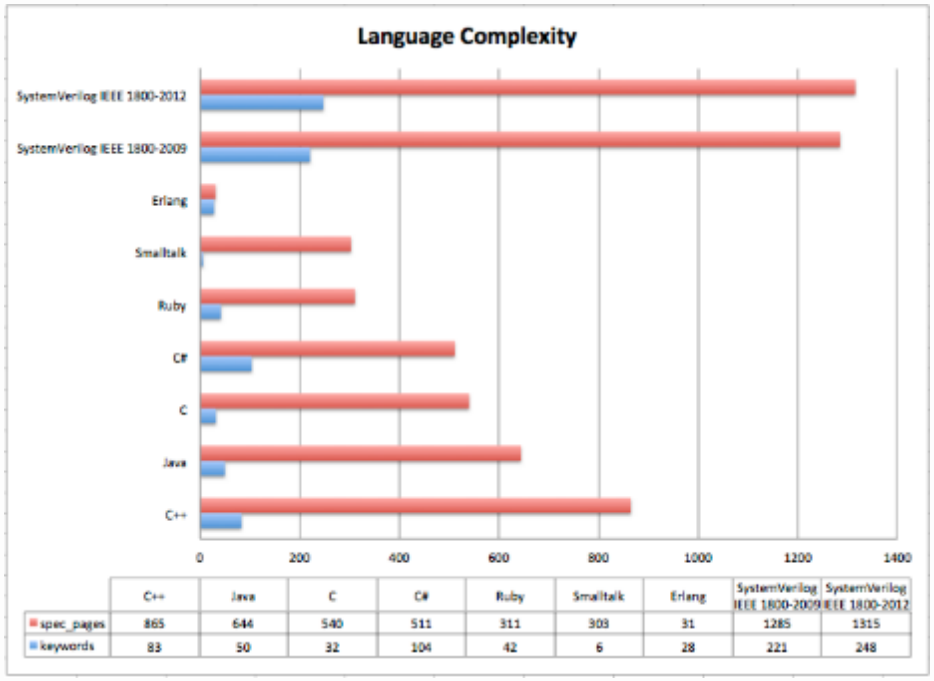
COroutinebased **CO**simulation**T**est**B**ench
allow writing verification testbenches in Python

```
// dff.v

module DFF (CLK, D, Q);
    input wire CLK, D;
    output reg Q;

    always @(posedge CLK)
    begin
        Q <= D;
    end

endmodule
```

```
# dff_test.py

import cocotb
from cocotb.clock import Clock
from cocotb.triggers import RisingEdge
import random

@cocotb.test()
async def dff_test(dut):

    cocotb.fork(Clock(dut.CLK, 10).start(start_high=False))

    for _ in range(5):
        dut.D <= random.choice([0,1])
        await RisingEdge(dut.CLK)
```



why cocotb



	C++	Java	C	C#	Ruby	Smalltalk	Erlang	SystemVerilog IEEE 1800-2009	SystemVerilog IEEE 1800-2012
spec_pages	865	644	540	511	311	303	31	1285	1315
keywords	83	50	32	104	42	6	28	221	248

www.fivecomputers.com

Language Rank	Types	Spectrum Ranking
1. Python		100.0
2. C++		99.7
3. Java		97.5
4. C		96.7
5. C#		89.4
6. PHP		84.9
7. R		82.9
8. JavaScript		82.6
9. Go		76.4
10. Assembly		74.1

2018 IEEE Spectrum

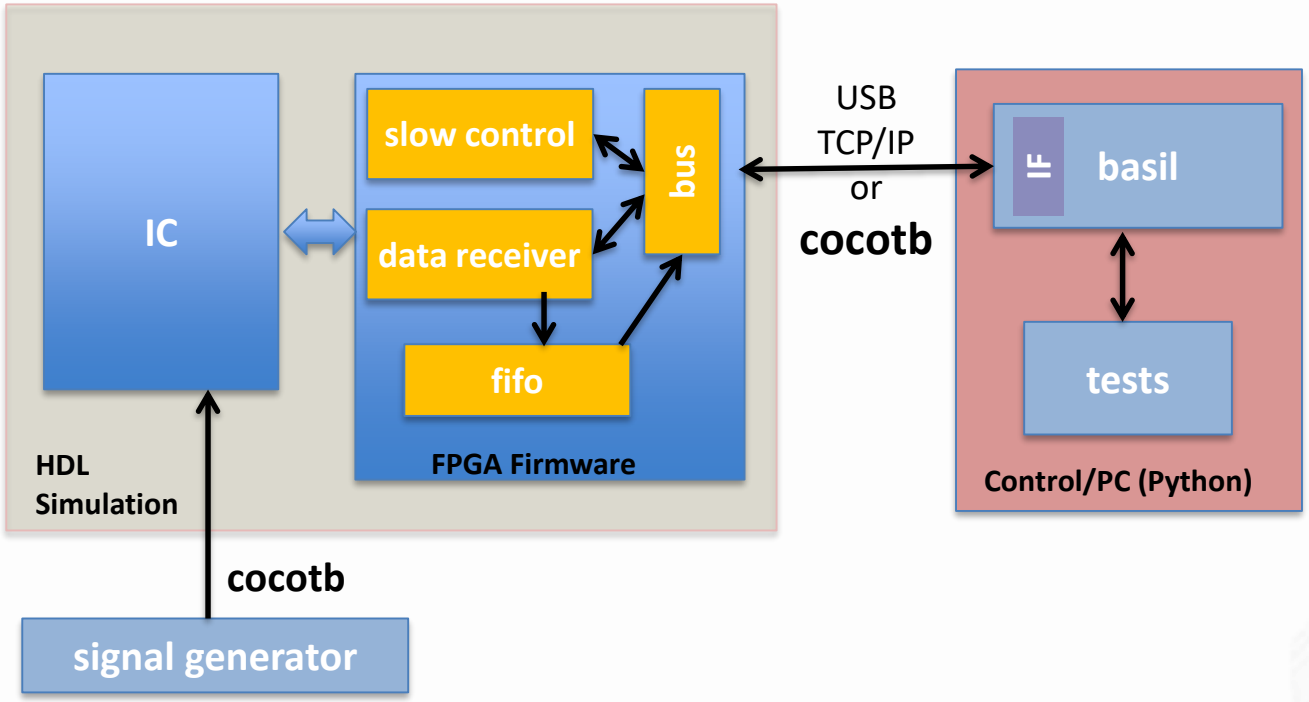
Python (3.7) keywords: **35**
“The Python Language Reference”: **122/160** pages

and UVM …

770 vs 1750 full Python standard library

-> Matlab

how we use cocotb



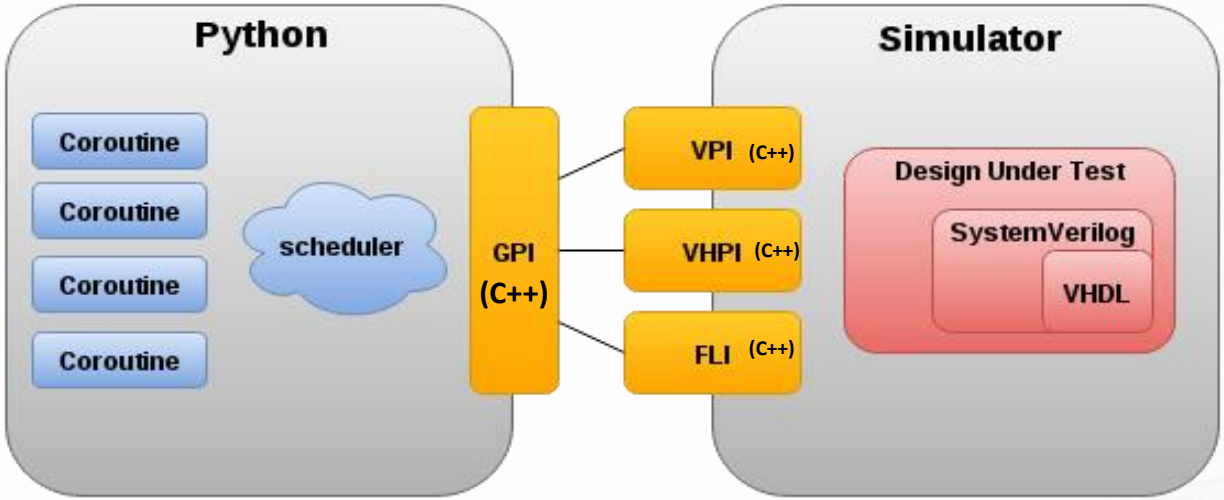
DAQ was a big issue

- Involve DAQ/validation (students of physics ;-) into chip verification
- Common software and firmware base for verification and daq

better chips -> faster measurements -> happy students

github.com/silab-Bonn/basil

how cocotb works



C. Higgs - Cocotb - ORCONF 2015

- compile C/C++ libraries
- start simulator
- load libraries
- load Python interpreter
- run simulations

makefiles

UNIVERSITÄT BONN

Control file:

```
# Makefile

CWD=$(shell pwd)

VERILOG_SOURCES =$(CWD)/dff.v

TOPLEVEL=dff
MODULE=dff_cocotb

include $(shell cocotb-config --makefiles)/Makefile.inc
include $(shell cocotb-config --makefiles)/Makefile.sim
```

compile libraries and start simulator

Running:

```
make
```

results.xml (xUnit format)

UNIVERSITÄT BONN

What is PyTest?

PyTest is a testing framework which allows us to write test codes using Python
You can write code to test anything like database , API, even UI if you want

Why use PyTest?

- Very easy to start with because of its simple and easy syntax
- Can run tests in parallel
- Can run a specific test or a subset of tests
- Automatically detect tests
- Skip tests
- Open source

Pytest:
Downloads last day: 453,858
Downloads last week: 2,747,330
Downloads last month: 12,598,277

https://www.guru99.com/

pytest – writing test

UNIVERSITÄT BONN

Installation:

```
pip install pytest
```

Automatic test discovery:

Pytest expects our tests to be located in files whose names begin with **test_** or end with **_test.py**
Pytest expects our test functions to be named prefix with **test_**

Example:

```
# test_capitalize.py

def capital_case(x):
    return x.capitalize()

def test_capital_case():
    assert capital_case('semaphore') == 'Semaphore'
```

pytest – running test

UNIVERSITÄT BONN

Running:

```
pytest
```

Result:

```
============================= test session starts ==============================
platform linux -- Python 3.7.4, pytest-5.2.1, py-1.8.0, pluggy-0.13.0
rootdir: /pytest
collected 1 item

test_capitalize.py .                                                    [100%]

============================== 1 passed in 0.72s ===============================

=================================== FAILURES ===================================
_____________________________ test_capital_case ______________________________

    def test_capital_case():
>       assert capital_case('semaphore') == 'semaphore'
E       AssertionError: assert 'Semaphore' == 'semaphore'
E         - Semaphore
E         ? ^
E         + semaphore
E         ? ^

test_capitalize.py:7: AssertionError
============================== 1 failed in 0.05s ===============================
```

cocotb-test

UNIVERSITÄT BONN

`github.com/themperek/cocotb-test`

cocotb-test provides standard Python unit testing capabilities for **cocotb**

- allow the look and feel of Python unit testing
- remove the need for Makefiles (includes Makefile compatibility mode)
- allow easy customization of simulation flow
- easy installation (especially on Windows)
- allow to use **pytest-xdist** or **pytest-parallel** for parallel runs
- supports: icarus, questa, ius, vcs, riviera, verilator
- allow to package binaries

Installation:

```
pip install -v cocotb-test
```

On Windows with conda:

```
conda install m2w64-gcc libpython
conda install -c conda-forge iverilog
```

setup / libraries

VPI/VHPI/FLI libraries are built during setup using **setuptools**.
Binaries can be distributed with the package (no compiler required on user side).

```
pip install -v cocotb-test
```

```
Installing collected packages: cocotb-test

…

 Installed /home/themperek/git/cocotb-test
    Compiling interface libraries for Icarus Verilog ...
    Compiling interface libraries for Questa ...
    Compiling interface libraries for GHDL ...
    Compiling interface libraries for IUS ...
    Compiling interface libraries for VCS ...
    Riviera executable not found. No VPI/VHPI interface will be available.
    Compiling interface libraries for Verilator ...
Successfully installed cocotb-test
```

execution

Files are compiled on demeand (change)
Simulators are started with Python subprocess

```
process = subprocess.check_call(cmd, cwd=self.sim_dir, env=self.env)
```

```
Skipping compilation:/cocotb-test/tests/sim_build/dff_test.vvp
vvp -M /cocotb-test/cocotb_test/libs/icarus -m libvpi /cocotb-test/tests/sim_build/dff_test.vvp
     -.--ns INFO     cocotb.gpi                  ../embed/gpi_embed.c:78   in set_program_name_in_venv       Did not detect Python virtual environment. Using
system-wide Python interpreter
     -.--ns INFO     cocotb.gpi                  ../gpi/GpiCommon.cpp:91   in gpi_print_registered_impl      VPI registered
     0.00ns INFO     cocotb.gpi                      gpi_embed.c:303  in embed_sim_init            Running on Icarus Verilog version 10.3 (stable)
     0.00ns INFO     cocotb.gpi                      gpi_embed.c:304  in embed_sim_init            Python interpreter initialized and cocotb loaded!
     0.00ns INFO     cocotb                        __init__.py:131  in _initialise_testbench       Running tests with Cocotb v1.2.0 from Unknown
     0.00ns INFO     cocotb                        __init__.py:148  in _initialise_testbench       Seeding Python random module with 1573221882
     0.00ns INFO     cocotb.regression               regression.py:189  in initialise              Found test dff_cocotb.run_test_001
     0.00ns INFO     cocotb.regression               regression.py:331  in execute                 Running test 1/1: run_test_001
     0.00ns INFO     ...test.run_test_001.0x7f9222c29d90      decorators.py:250  in _advance          Starting test: "run_test_001"
                                                                                                    Description: Automatically generated test
0.00ns INFO     cocotb.scoreboard.dff_test          scoreboard.py:216  in add_interface           Created with reorder_depth 0
100005001000000000.00ns INFO     cocotb.regression              regression.py:276  in handle_result          Test Passed: run_test_001
100005001000000000.00ns INFO     cocotb.regression              regression.py:212  in tear_down              Passed 1 tests (0 skipped)
100005001000000000.00ns INFO     cocotb.regression              regression.py:383  in _log_test_summary
*************************************************************************************
** TEST                PASS/FAIL  SIM TIME(NS)  REAL TIME(S)  RATIO(NS/S) **
*************************************************************************************
** dff_cocotb.run_test_001    PASS     100005001000000000.00        1.41   70875253831138168.00  **
*************************************************************************************
100005001000000000.00ns INFO     cocotb.regression              regression.py:400  in _log_sim_summary
*************************************************************************************
**                                      ERRORS : 0                                   **
*************************************************************************************
**                        SIM TIME : 100005001000000000.00 NS                        **
**                        REAL TIME : 1.52 S                                         **
**                        SIM / REAL TIME : 65601362995307280.00 NS/S                **
*************************************************************************************
```

example

UNIVERSITÄT BONN

Adding test:

```
# test_dff.py

from cocotb_test.run import run

def test_dff():
    run(
        verilog_sources=["dff.v"], # sources
        toplevel="dff",            # top level HDL
        module="dff_cocotb"        # name of cocotb test module
    )
```

Running:

```
pytest [-s]
```

simulator customization

UNIVERSITÄT BONN

```python
# test_dff_custom_icarus.py

from cocotb_test.run import run
from cocotb_test.simulator import Icarus

class IcarusCustom(Icarus):
    def run_command(self):
        return ["vvp", "-v", "-l", self.logfile, "-M", self.lib_dir, "-m", "libvpi", self.sim_file]

def test_dff_custom_icarus():
    IcarusCustom(
        verilog_sources=["dff.v"],
        toplevel="dff",
        module="dff_cocotb",
        logfile="custom_log.log",  # extra custom argument
    ).run()
```

parametrization

```python
# test_plus_args.py

@pytest.mark.parametrize('param1', [10,20,30,40])
def test_plus_args(param1):
    run(
        verilog_sources=["plus_args.v"],
        toplevel="plus_args",
        plus_args=["+PARAM1=" + str(param1)],
    )
```

```
============================= test session starts ==============================
platform linux -- Python 3.7.4, pytest-5.2.1, py-1.8.0, pluggy-0.13.0
rootdir: /cocotb-test
plugins: xdist-1.29.0, parallel-0.0.9, forked-1.1.1, cocotb-test-0.0.6
collected 4 items

test_plus_args.py ....                                                   [100%]

============================== 4 passed in 1.05s ===============================
```

test control

UNIVERSITÄT BONN

skip/skipif:

```
@pytest.mark.skipif(os.getenv("SIM") == "ghdl", reason="Verilog not supported")
def test_dff():
    run(
        verilog_sources=["dff.v"], # sources
        toplevel="dff",            # top level HDL
        module="dff_cocotb"        # name of cocotb test module
    )
```

xfail:

```
@pytest.mark.xfail
def test_dff():
    run(
        verilog_sources=["dff_fail.v"],
        toplevel="dff",
        module="dff_cocotb"
    )
```

selecting test to run

UNIVERSITÄT BONN

```
# test_set.py

import pytest

@pytest.mark.set1
def test_file1():
    pass

@pytest.mark.set2
def test_file2():
    pass
```

Substring matching

```
#pytest -k  <expression>

pytest -k  file1/file2
```

By markers

```
#pytest -m  <markexpr>

pytest -m  set1/set2
```

pre/post run

UNIVERSITÄT BONN

Module :

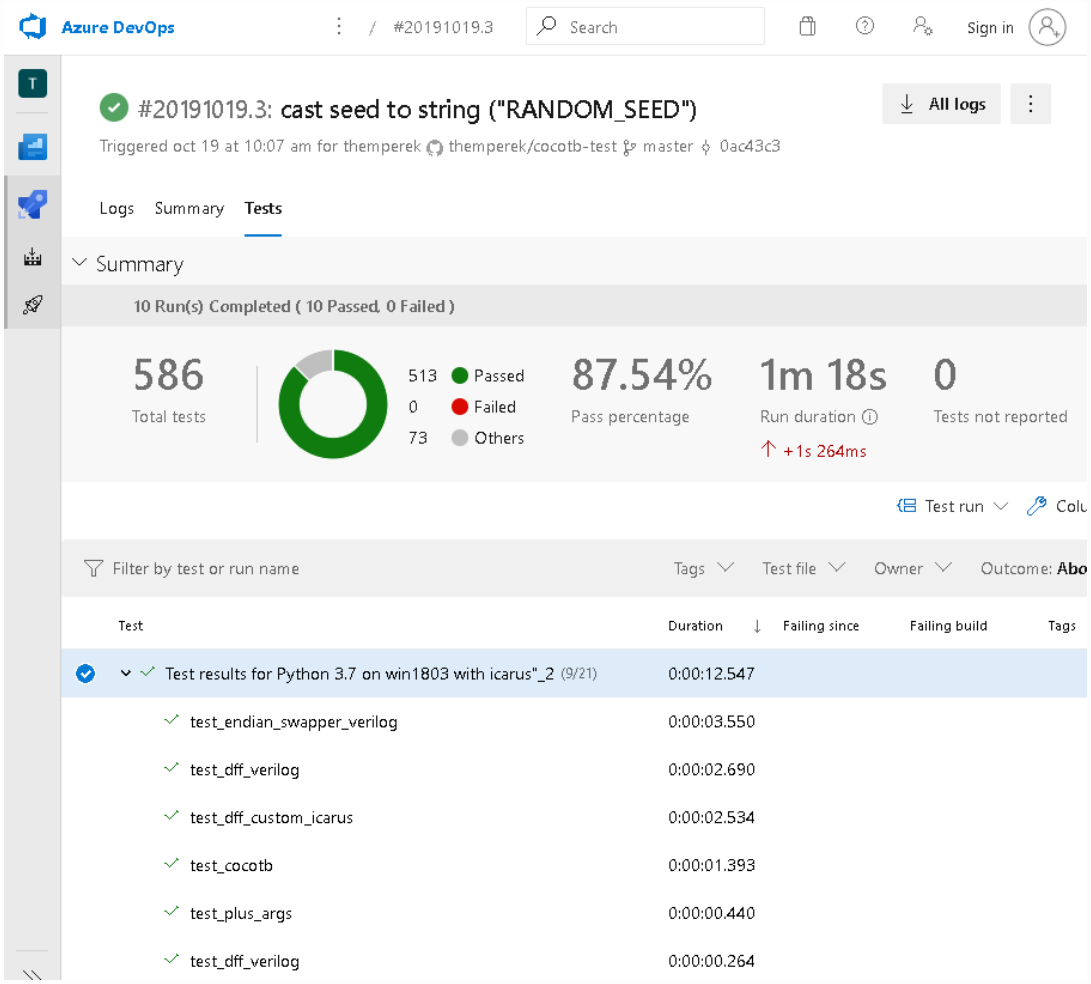
```
@pytest.fixture(scope="module", autouse=True)
def module_run_at_beginning(request):
    print("In module_run_at_beginning()")

    def module_run_at_end():
        print("In module_run_at_end()")

    request.addfinalizer(module_run_at_end)
```

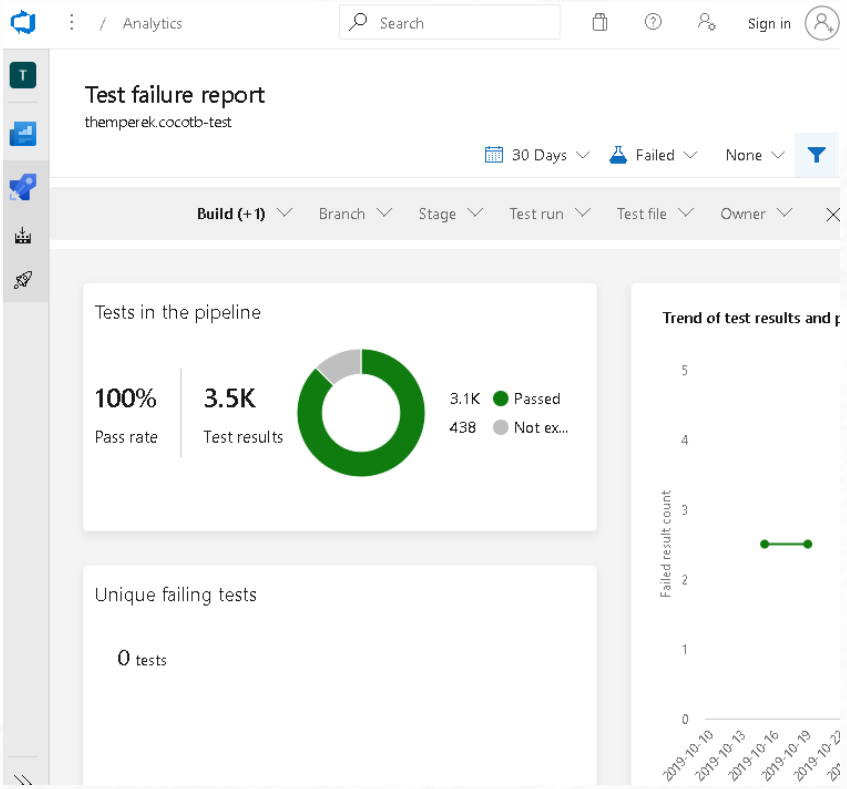
Session :

```
@pytest.fixture(scope="session", autouse=True)
def session_run_at_beginning(request):
    print("In session_run_at_beginning()")

    def sesion_run_at_end():
        print("In session_run_at_end()")

    request.addfinalizer(sesion_run_at_end)
```

reporting / ci

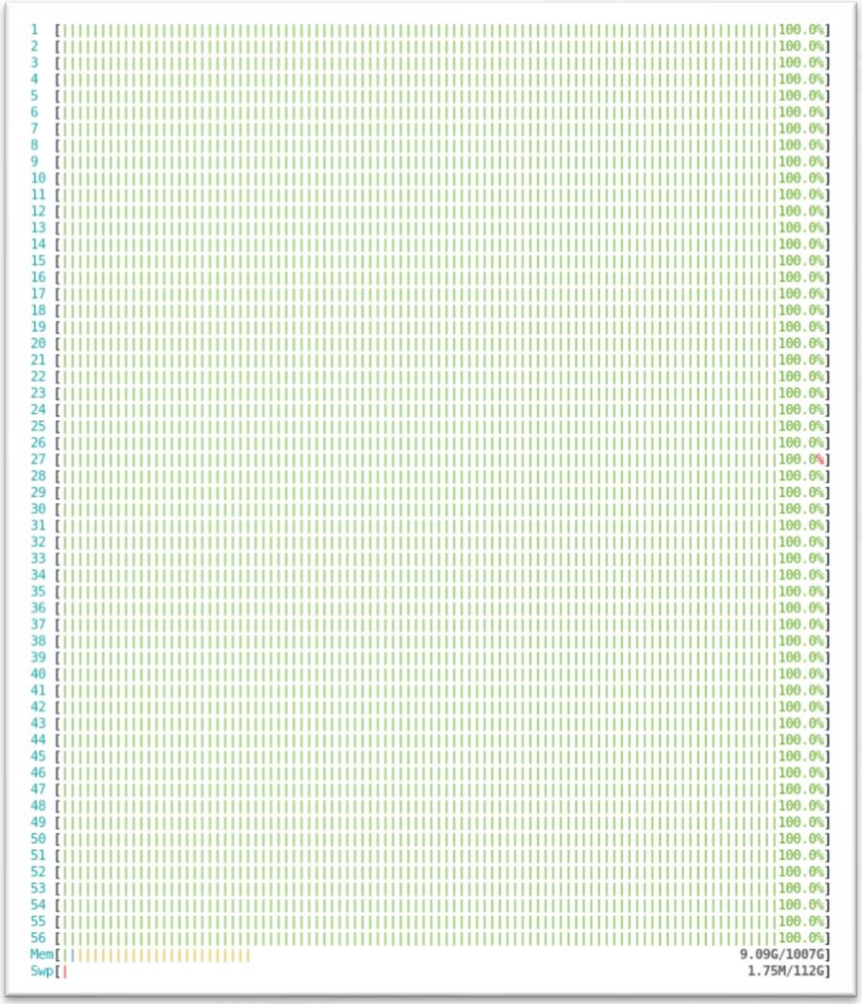
```
pytest -s --junitxml=test-results.xml --cocotbxml=test-cocotb.xml
```





parallel run

htop cpu usage:




```
pip install pytest-xdist
```

```
pytest -n auto
```

will work with any simulator
run on some 10k cluster …

cocotb-coverage

UNIVERSITÄT BONN

From: github.com/mciepluc/cocotb-coverage

```
class Point(crv.Randomized): # point represented by x and y coordinates in range (-10, 10)
    def __init__(self, x, y):
        crv.Randomized.__init__(self)
        self.x = x
        self.y = y
        self.add_rand("x", list(range(-10, 10)))
        self.add_rand("y", list(range(-10, 10)))
        self.add_constraint(lambda x, y: x < y) # constraining the space so that x < y
...
p = Point(0,0) # create an arbitrary point
for _ in range (10):
    # cover example arithmetic properties
    @CoverPoint("top.x_negative", xf = lambda point : point.x < 0, bins = [True, False])
    @CoverPoint("top.y_negative", xf = lambda point : point.y < 0, bins = [True, False])
    @CoverPoint("top.xy_equal", xf = lambda point : point.x == point.y, bins = [True, False])
    @CoverCross("top.cross", items = ["top.x_negative", "top.y_negative"])
    def plot_point(point):
        ...
    p.randomize()  # randomize object
    plot_point(p)  # call a function which will sample the coverage

#export coverage
coverage_db.export_to_yaml()
```

coverage.py : 478 lines
crv.py : 308 lines

multi-thread test scattering

UNIVERSITÄT BONN

```python
from cocotb_test.run import run
import pytest

@pytest.mark.parametrize('seed', range(0, 1000))
def test_basic(seed):
    run(
        verilog_sources=["test_basic.v"],
        toplevel="tb",
        module="test_basic",
        seed=seed
    )

@pytest.fixture(scope="session", autouse=True)
def session_run(request):

    def collect_coverege():
        …

    request.addfinalizer(collect_coverege)
```

```
pytest -n auto   /  pytest --workers 10 --tests-per-worker auto
```

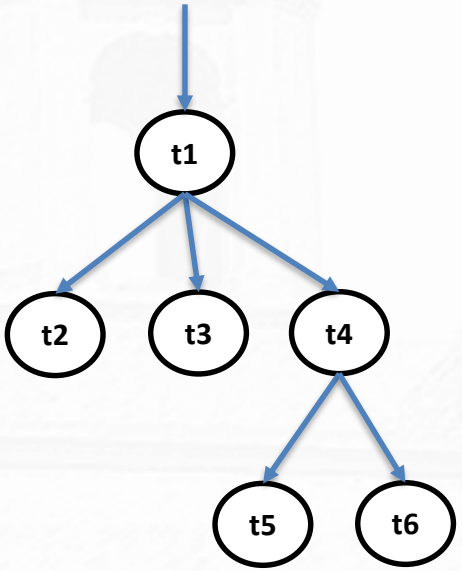
Future? : pytest -n auto --cocotbcov=**coverage.yaml**

smart coverage?

UNIVERSITÄT BONN

Use checkpoints and previous test results to control test parameters (something for ML?).

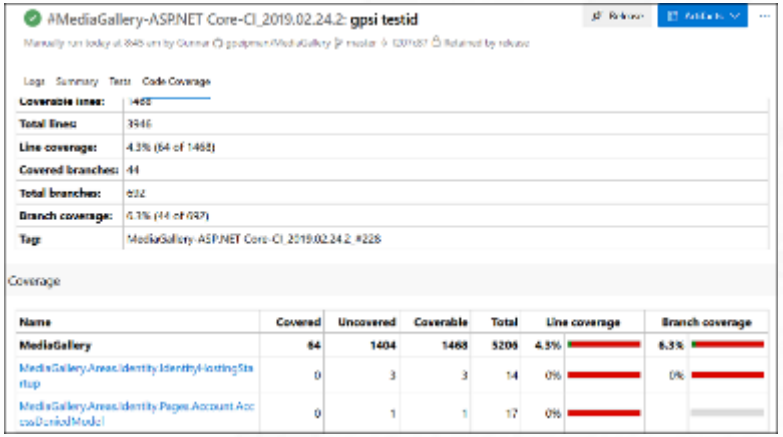
```
from mp_tests import MpTest, mp_run

t1 = MpTest(sim_args, seed=1, checkpoint=None, pram1=1, param2=3)
t2 = MpTest(sim_args, seed=2, checkpoint=None, pram1=2, param2=3)
t3 = MpTest(sim_args, seed=3, checkpoint=None, pram1=3, param2=3)
t4 = MpTest(sim_args, seed=4, checkpoint=None, pram1=4, param2=3)

mp_run([t1, t2, t3, t4])

t1_1 = MpTest(sim_args, seed=1, checkpoint=t1.checkpoint, pram1=1, param2=3)

mp_run([t1_1])

if t1.ret1 == 5:
    tests = []
    for i in range(2):
        tests.append(MpTest(sim_args, seed=i, checkpoint=t2.checkpoint))

    mp_run(tests)
```



contact **@mciepluc**

conclusions and future

- fully Python cocotb based testing
- using standard pytest
- parallel execution



Future:

- improve simulator support
- improve API
- top level cocotb parametrization
- coverage report
- integrate into cocotb?

something extra

```
pip install pytest-sugar
```