

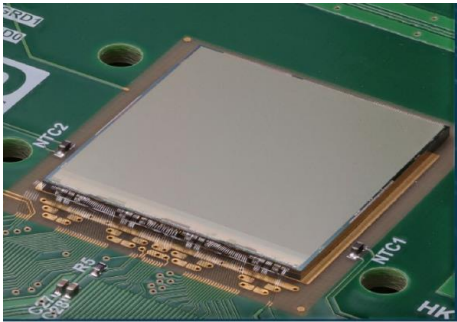
A reusable verification, emulation and validation flow for ASIC design

Tomasz Hemperek

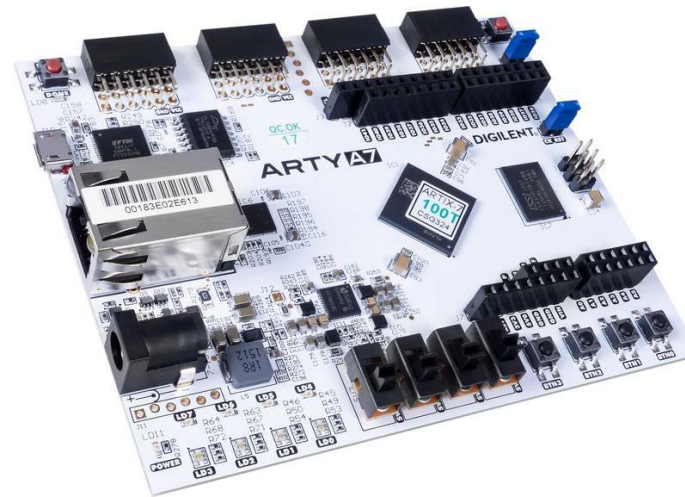
Typical setup

... in my sensor design experience

ASIC



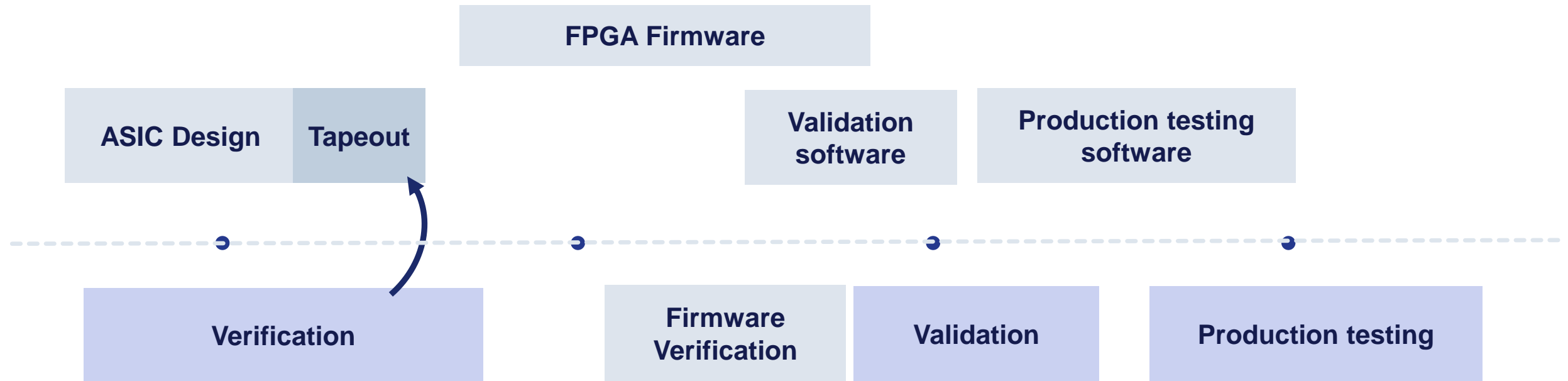
FPGA



PC



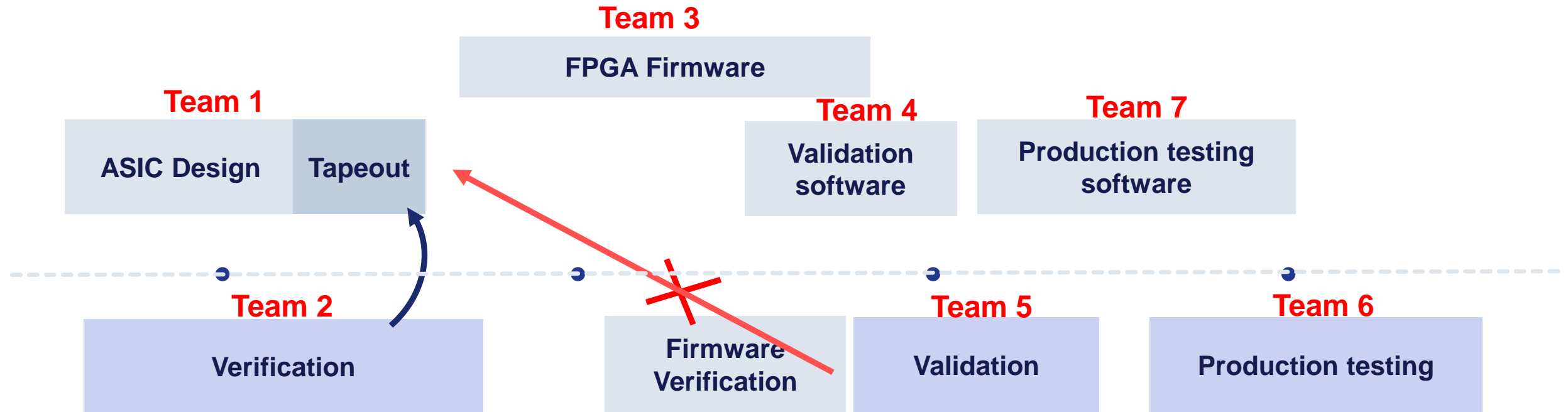
Development life cycle



Multiple teams → communication via specifications
→ non overlapping codebase

Different timelines → very costly late discovered specification issues and bugs

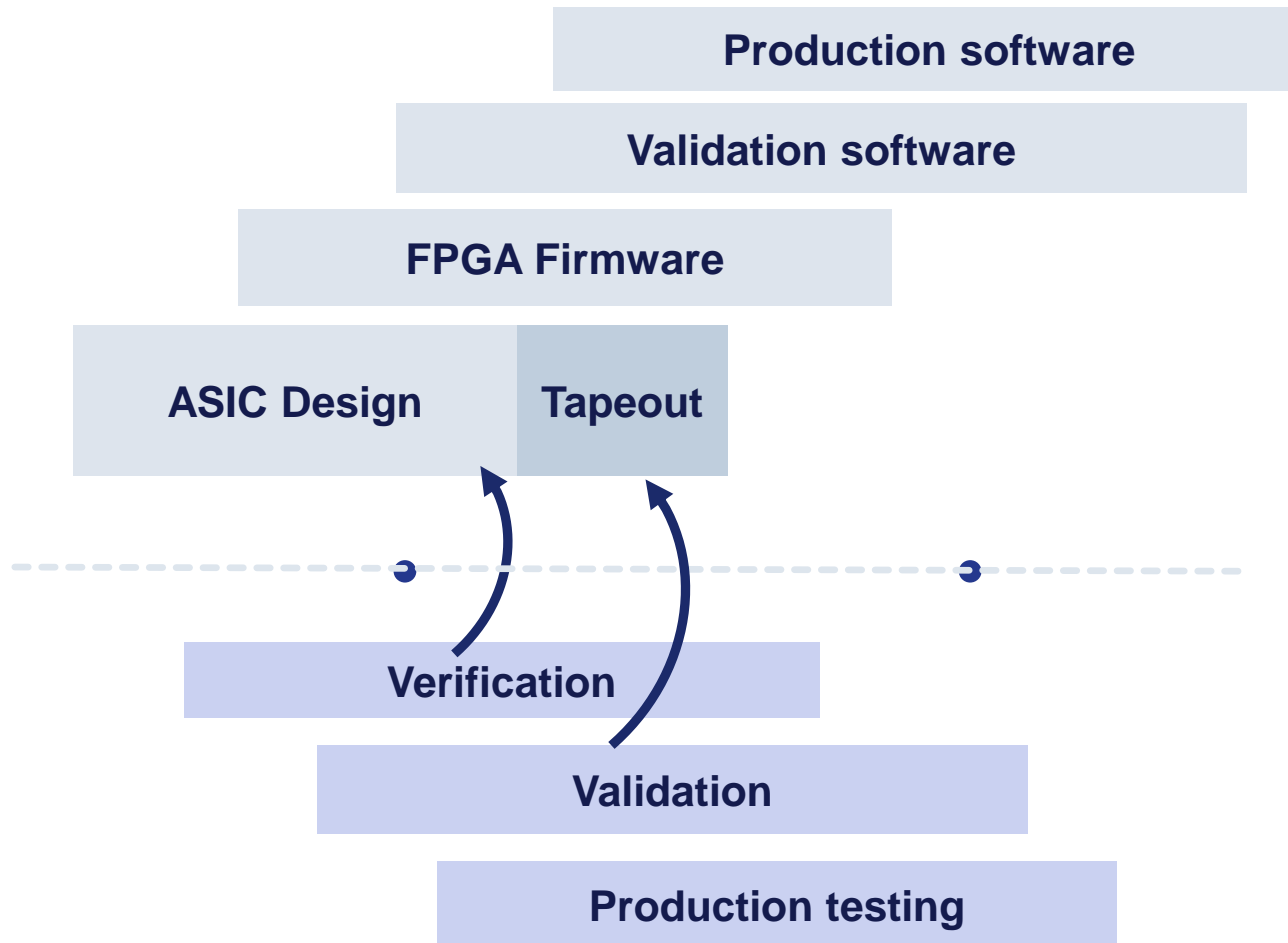
Development life cycle



Multiple teams → communication via specifications
→ non overlapping codebase

Different timelines → very costly late discovered specification issues and bugs

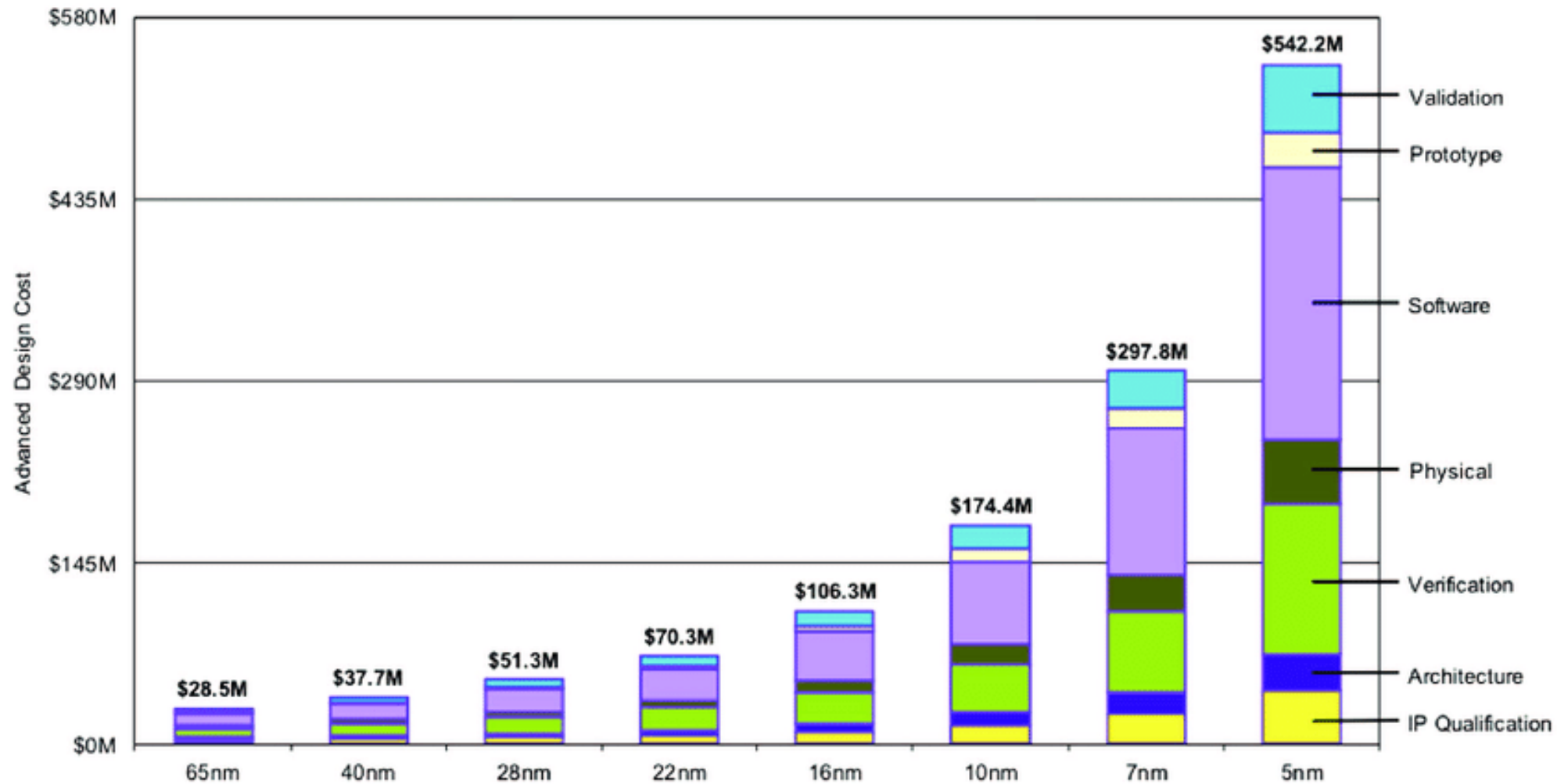
Development life cycle



- shorten development time
- improve communication
- early influence on ASIC functionality
- better coverage

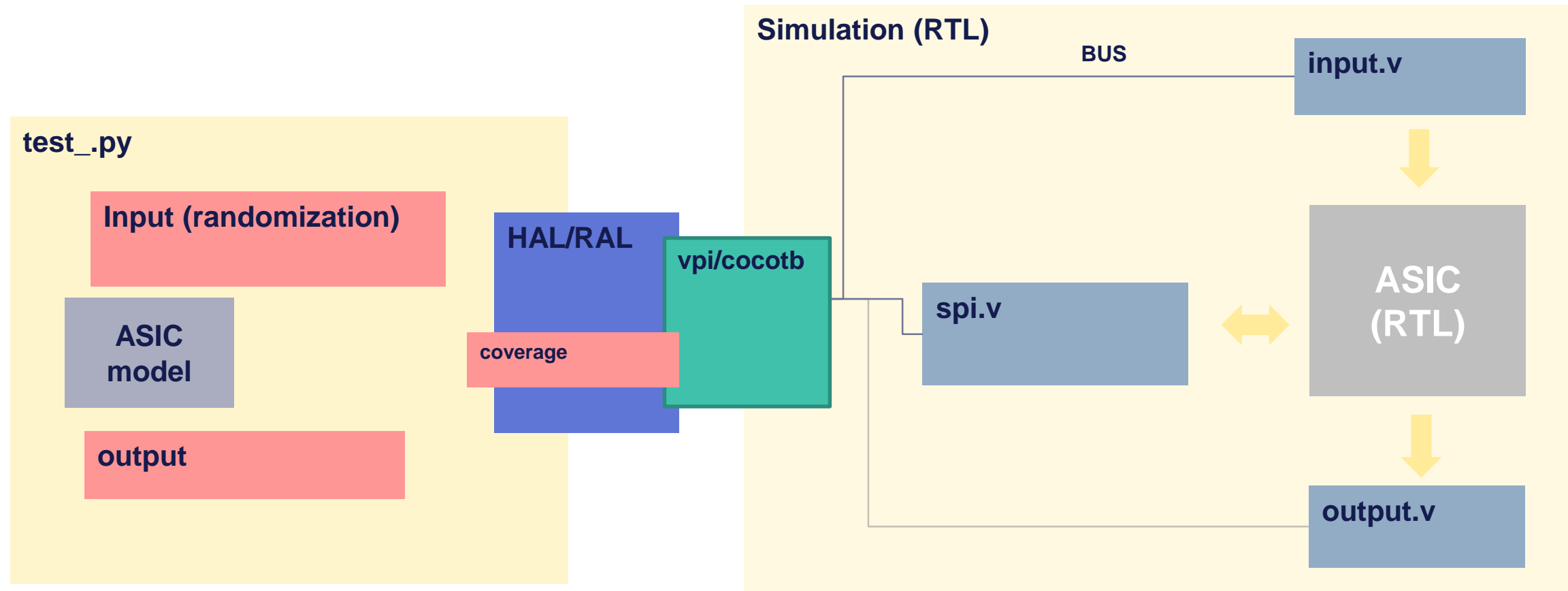
Verification is software.

Chip Design and Manufacturing Cost



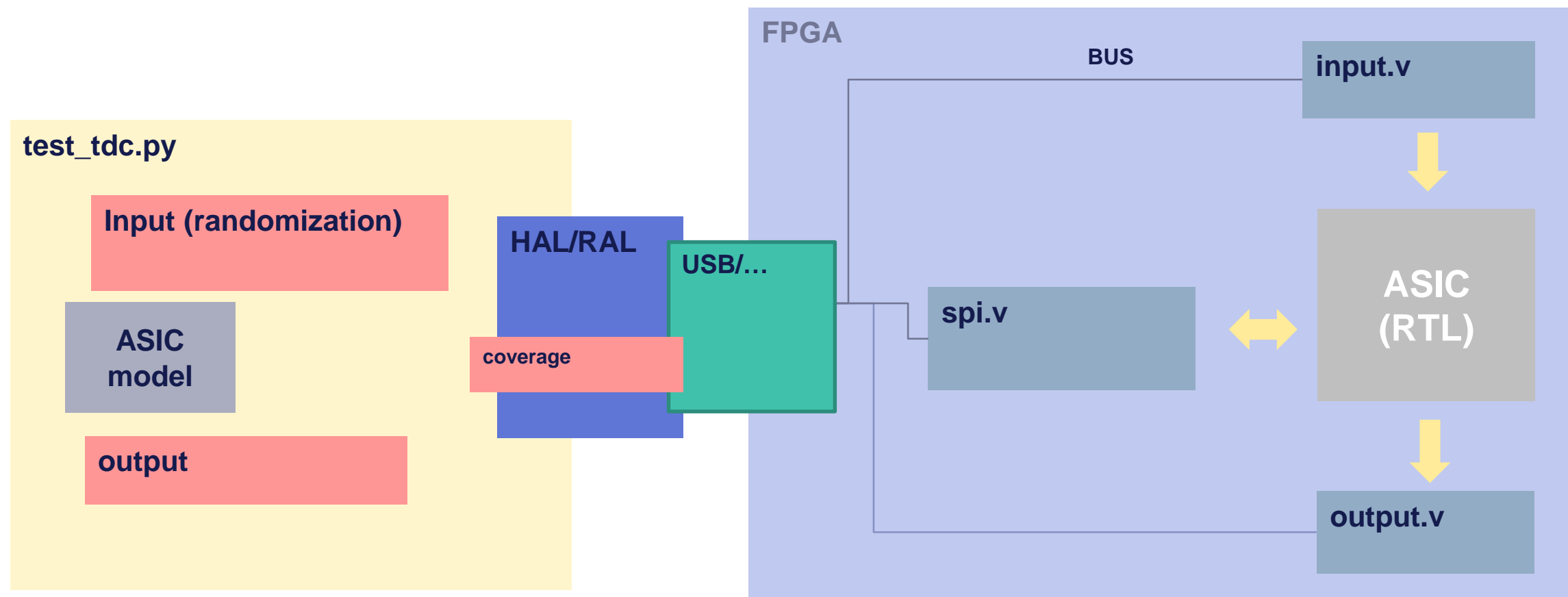
Source: IBS

Simulation/Verification

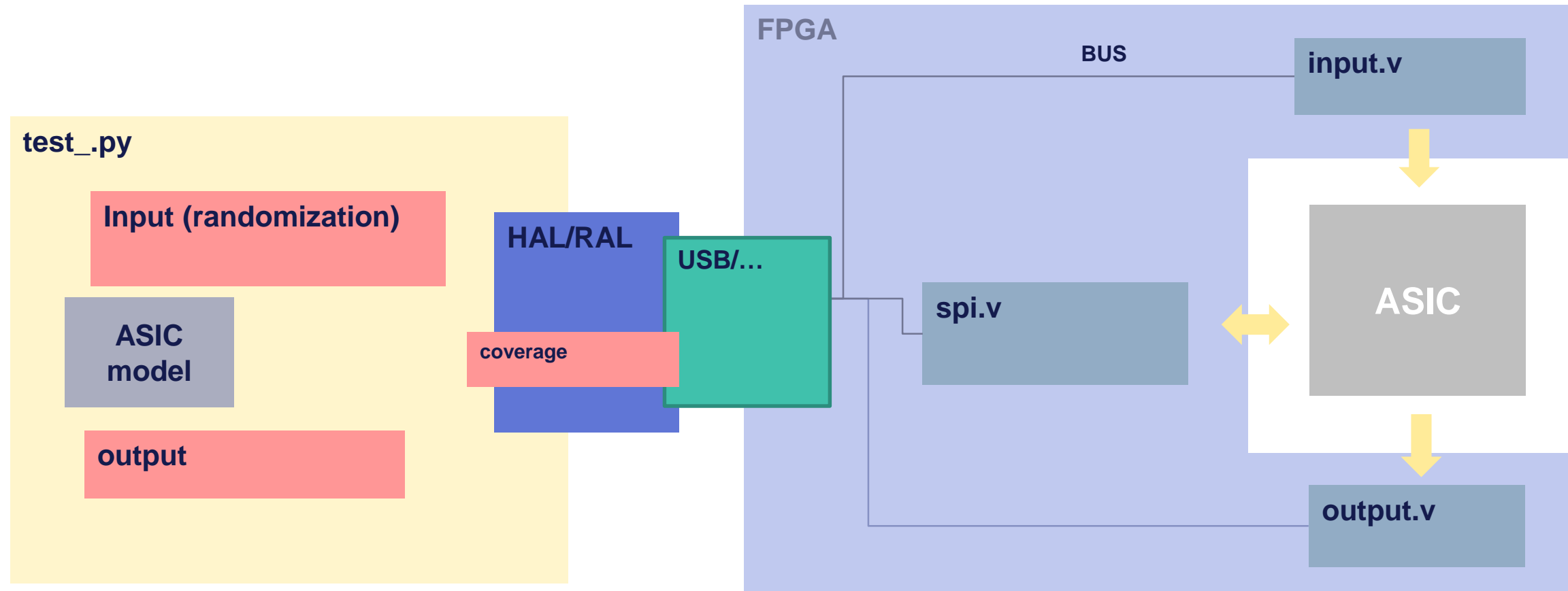


Move the verification part to the software.

Emulation



Validation



Example use case

<https://github.com/themperek/ORConf23>

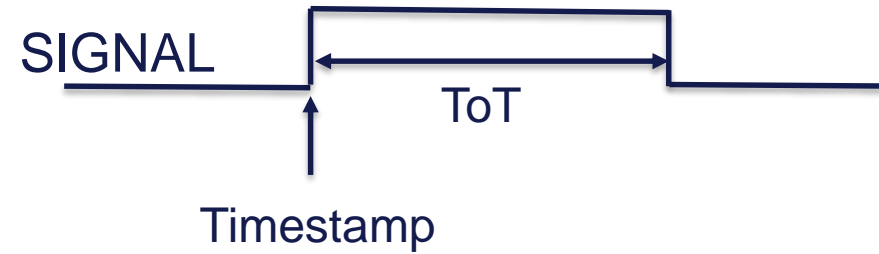
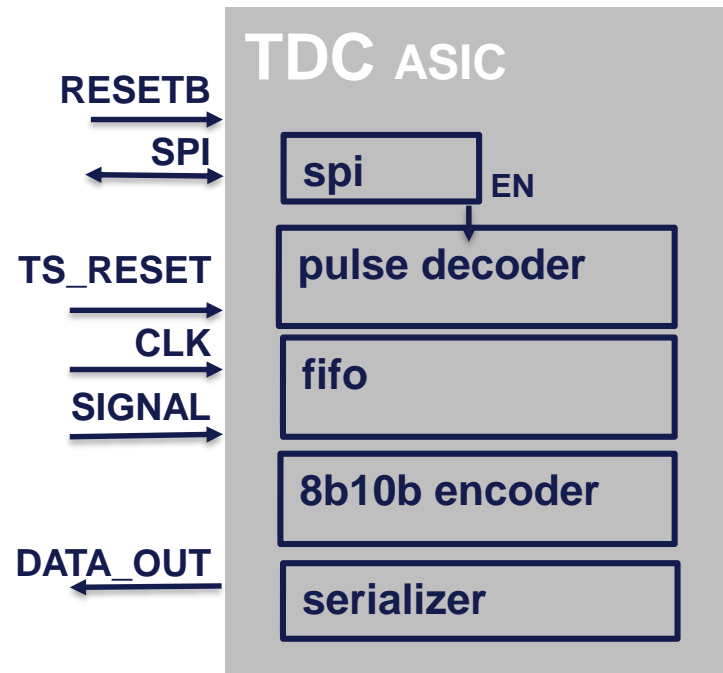
Disclaimer:

This is a very simple demo.

In large part, this work was done while at the University of Bonn (https://gitlab.cern.ch/silab/tdc_example/).

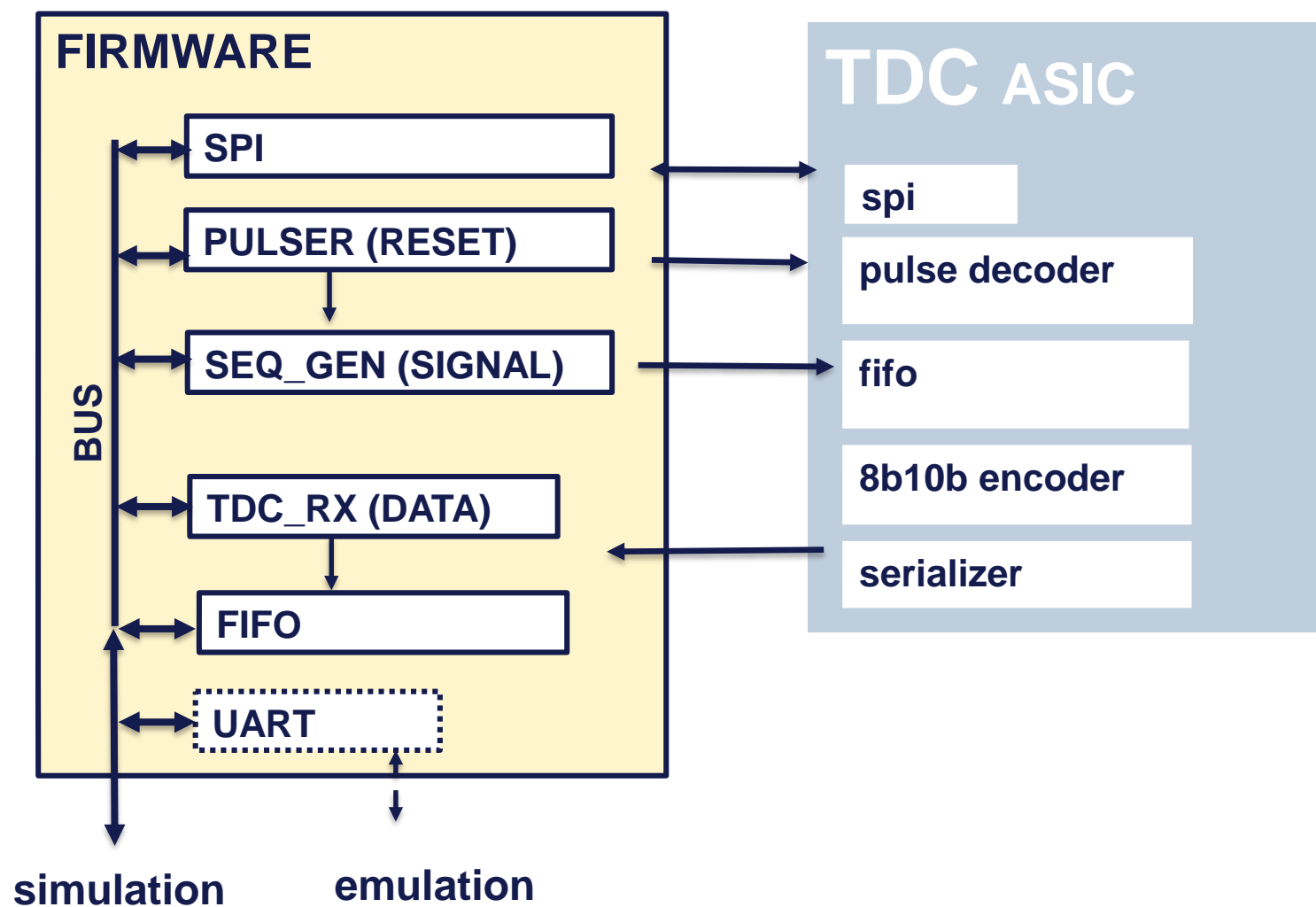
Some amount of “hackery” is required.

TDC / DUT



- Single channel TDC
- Configured/Enabled by SPI interface
- Input "SIGNAL" sampled with CLK
- Timestamp (16bit) and ToT (8bit) stored to FIFO
- Timestamp reset with "TS_RESET"
- Data are 8b10b encoded (simple protocol) and serialized

Firmware



Firmware/software modules based on Basil
<https://github.com/silab-bonn/basil>

In emulation use UART
https://github.com/ultraembedded/core_dbg_bridge

Example module

tdc_fw_core.sv:

```
spi_sbus #(
    .BASEADDR (SPI_BASEADDR),
    .HIGHADDR (SPI_HIGHADDR),
    .ABUSWIDTH(32),
    .MEM_BYTES(16)
) spi (
    .BUS_CLK(BUS_CLK),
    .BUS_RST(BUS_RST),
    .BUS_ADD(BUS_ADD),
    .BUS_DATA_IN(BUS_DATA_IN[7:0]),
    .BUS_DATA_OUT(SPI_DATA_OUT),
    .BUS_RD(BUS_RD),
    .BUS_WR(BUS_WR),

    .SPI_CLK (SPI_CLK),
    .EXT_START(1'b0),

    .SCLK(SCLK),
    .SDI (SDI),
    .SDO (SDO),
    .SEN (),
    .SLD (SLD)
);
```

Hardware Abstraction Layer (HAL)

Every firmware (Verilog) module has a corresponding driver

basil/HL/spi.py:

```
class spi(RegisterHardwareLayer):
    '''Implement serial programming interface (SPI) driver.
    ...

    _registers = {'RESET': {'descr': {'addr': 0, 'size': 8, 'properties': ['writeonly']}},
                  'VERSION': {'descr': {'addr': 0, 'size': 8, 'properties': ['ro']}},
                  'READY': {'descr': {'addr': 1, 'size': 1, 'properties': ['ro']}},
                  'START': {'descr': {'addr': 1, 'size': 8, 'properties': ['writeonly']}},
                  'SIZE': {'descr': {'addr': 3, 'size': 16}},
                  'WAIT': {'descr': {'addr': 5, 'size': 32}},
                  'REPEAT': {'descr': {'addr': 9, 'size': 32}},
                  'EN': {'descr': {'addr': 13, 'size': 1}},
                  'MEM_BYTES': {'descr': {'addr': 14, 'size': 16, 'properties': ['ro']}}}

    _require_version = "==2"
```

Top configuration

tdc.yaml

```
transfer_layer:
- name : intf
  type : uart_bridge
  init:
    port : "/dev/ttyUSB1"
    baudrate : 115200

hw_drivers:
- name : FIFO
  type : bram_fifo
  interface : intf
  base_addr : 0x8000
  base_data_addr: 0x80000000

- name : SPI
  type : spi
  interface : intf
  base_addr : 0x3000

- name : RX
  type : tdc_rx
  interface : intf
  base_addr : 0x4000
```

← communication

← hardware drivers/HAL

← data fifo

← spi module

Communication interfaces are separate.
Memory address mapping are defined.

Register Abstraction (RAL)

tdc.yaml

```
/ registers:  
/ - name      : CONFIG_REG  
/   type      : StdRegister  
/   driver     : None  
/   size       : 8  
/   fields:  
/     - name    : EN  
/       size    : 1  
/       offset  : 0
```

- Can define arbitrary size registers
- An arbitrary amount of fields
- Can specify bit order
- Can specify arrays (repeating fields)
- Can be initialized (also with configuration file)

User experience

Initialize:

```
from basil.dut import Dut
self.chip = Dut("tdc.yaml")
self.chip.init()
```

Same code for verification
and validation.

```
def test_readout(self):
    self.assertEqual(self.chip["RX"].READY, 1)

    self.chip["CONFIG_REG"]["EN"] = 1
    self.write_config()

    self.assertEqual(self.chip["RX"].READY, 1)

    self.chip["RX"].ENABLE_RX = 1

    self.chip["TS_RESET"].DELAY = 1
    self.chip["TS_RESET"].WIDTH = 1

    self.chip["SIGNAL_SEQ"].set_data([0x00, 0x07, 0x00, 0x00])
    self.chip["SIGNAL_SEQ"].REPEAT = 1
    self.chip["SIGNAL_SEQ"].SIZE = 8 * 8 + 8 * 16
    self.chip["SIGNAL_SEQ"].EN_EXT_START = 1

    self.chip["FIFO"].RESET
    self.chip["TS_RESET"].START

    while not self.chip["SIGNAL_SEQ"].READY:
        pass

    self.assertEqual(self.chip["FIFO"].FIFO_SIZE, 16)

    data = self.chip["FIFO"].get_data()
```

← Set configuration and send via SPI

← Check 8b10b sync

← Enable data receiving

← Timestamp reset

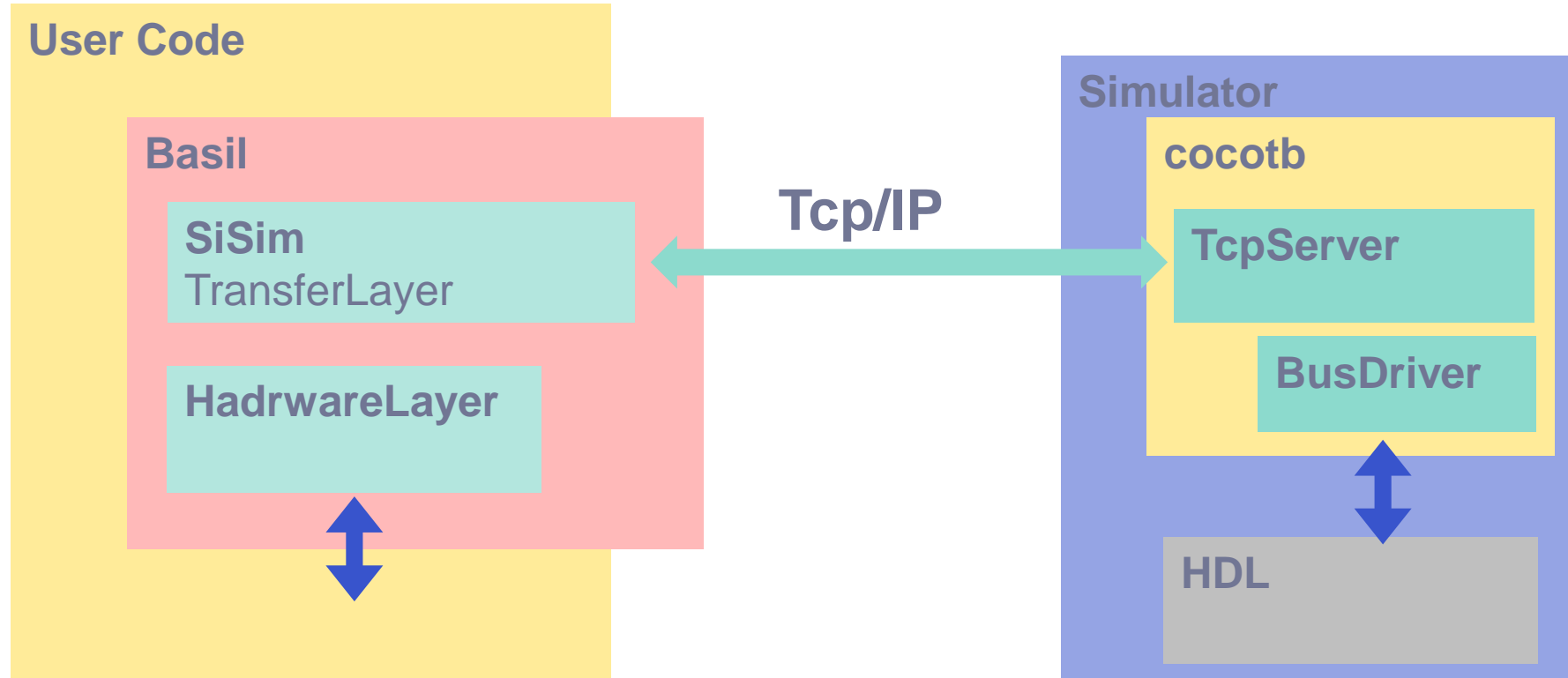
← Configure SIGNAL pattern

← Wait for sequence to finish

← Check FIFO fill level

← Get data from FIFO

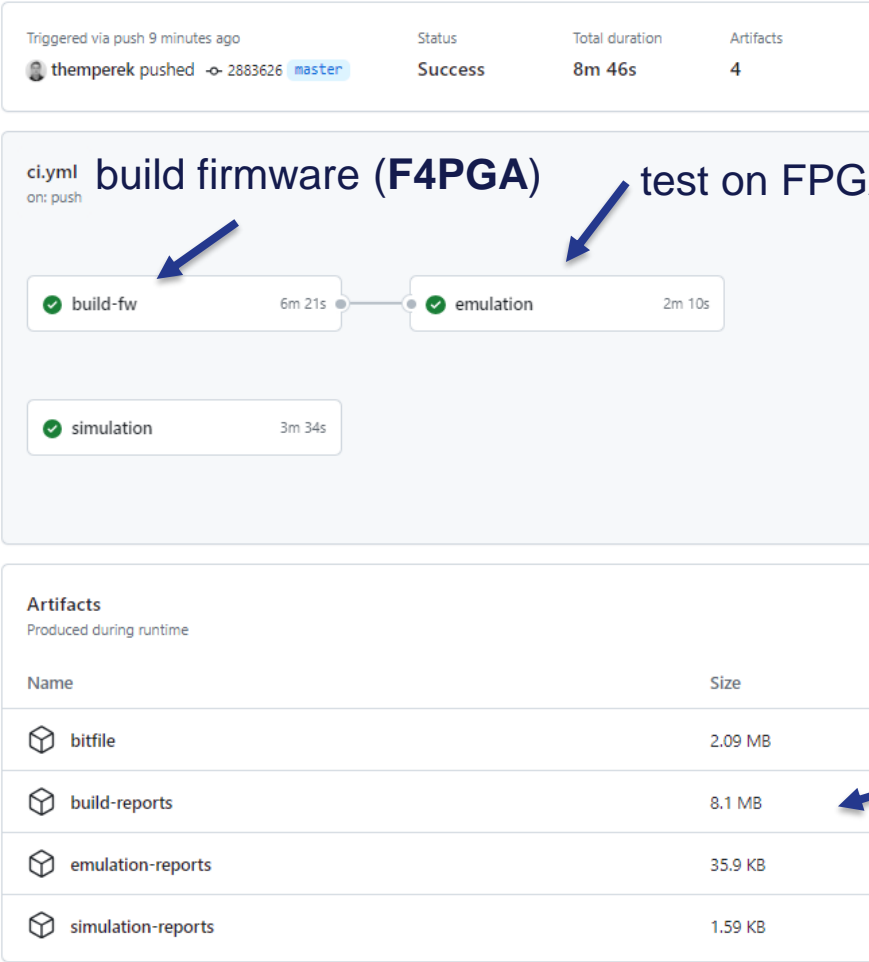
How simulation works



UART interface is replaced with the simulation interface.

Continuous integration

<https://github.com/themperek/ORConf23/actions>



Test reports

```
tomasz.hemperek> pytest -v test
===== test session starts =====
platform linux -- Python 3.8.15, pytest-7.4.0, pluggy-1.3.0 -- /home/tomasz.hemperek/m
iniconda3/bin/python
cachedir: .pytest_cache
metadata: {'Python': '3.8.15', 'Platform': 'Linux-4.18.0-477.13.1.el8_8.x86_64-x86_64-
with-glibc2.17', 'Packages': {'pytest': '7.4.0', 'pluggy': '1.3.0'}, 'Plugins': {'coco
tb-test': '0.2.3', 'metadata': '3.0.0', 'html': '4.0.1', 'xdist': '3.3.1'}}
rootdir: /home/tomasz.hemperek/git/ORConf23
configfile: pytest.ini
plugins: cocotb-test-0.2.3, metadata-3.0.0, html-4.0.1, xdist-3.3.1
collected 6 items

test/test_tdc.py::TestTDC::test_parameterized_0 PASSED [ 16%]
test/test_tdc.py::TestTDC::test_parameterized_1 PASSED [ 33%]
test/test_tdc.py::TestTDC::test_parameterized_2 PASSED [ 50%]
test/test_tdc.py::TestTDC::test_parameterized_3 PASSED [ 66%]
test/test_tdc.py::TestTDC::test_simple_readout PASSED [ 83%]
test/test_tdc.py::TestTDC::test_spi PASSED [100%]

===== 6 passed in 10.00s =====
```

report.html

Report generated on 12-Sep-2023 at 14:11:19 by [pytest-html](#) v4.0.1

Environment


Python	3.8.15
Platform	Linux-4.18.0-477.13.1.el8_8.x86_64-x86_64-with-glibc2.17
Packages	<ul style="list-style-type: none">• pytest: 7.4.0• pluggy: 1.3.0
Plugins	<ul style="list-style-type: none">• cocotb-test: 0.2.3• metadata: 3.0.0• html: 4.0.1• xdist: 3.3.1

Summary

6 tests took 00:00:09.

(Un)check the boxes to filter the results.

☒ 0 Failed, ☒ 6 Passed, ☒ 0 Skipped, ☒ 0 Expected failures, ☒ 0 Unexpected passes, ☒ 0 Errors, ☒ 0 Reruns [Show all details](#) / [Hide all details](#)

Result 	Test	Duration	Links
Passed	test/test_tdc.py::TestTDC::test_parameterized_0	00:00:02	
Passed	test/test_tdc.py::TestTDC::test_parameterized_1	00:00:02	
Passed	test/test_tdc.py::TestTDC::test_parameterized_2	00:00:02	
Passed	test/test_tdc.py::TestTDC::test_parameterized_3	00:00:02	
Passed	test/test_tdc.py::TestTDC::test_simple_readout	00:00:02	
Passed	test/test_tdc.py::TestTDC::test_spi	00:00:01	

Coverage and Randomization

Define covergroup:

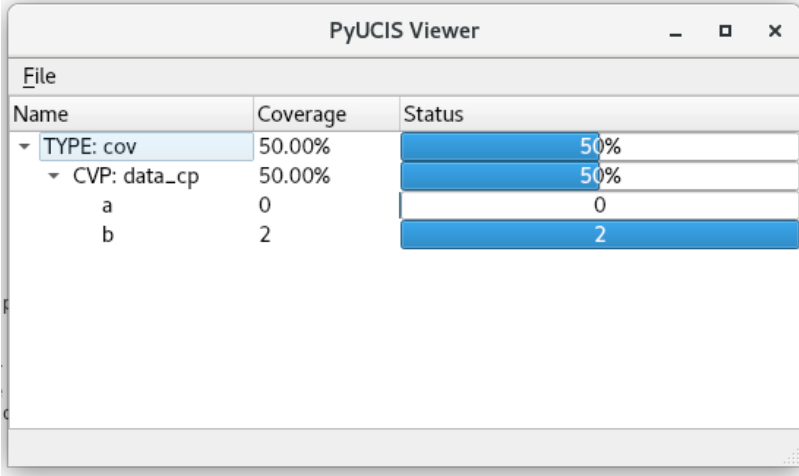
```
@vsc.covergroup
class cov(object):
    def __init__(self):
        self.with_sample(dict(data=vsc.bit_t(8)))
        self.data_cp = vsc.coverpoint(self.data, bins=dict(data=vsc.bin_array([8], [0, 255])))
```

Sample:

```
@parameterized.expand([[x] for x in range(4)])
def test_parameterized(self, data):
    self.assertEqual(self.chip["RX"].READY, 1)

    self.cover.sample(data * 32)

    ret = self.rw_dut_spi(0x91)
    self.assertEqual(ret, 0x91)
```



The PyUCIS Viewer window displays a table of coverage data. The table has three columns: Name, Coverage, and Status. The data is organized into a tree structure under the 'TYPE: cov' node. The 'CVP: data_cp' node is expanded, showing two bins: 'a' and 'b'. Bin 'a' has a coverage of 0 and a status of 0. Bin 'b' has a coverage of 2 and a status of 2. The status column is highlighted in blue.

Name	Coverage	Status
TYPE: cov	50.00%	50%
CVP: data_cp	50.00%	50%
a	0	0
b	2	2

PyVSC: <https://github.com/fvutils/pyvsc>

Testing with pytest

Tests parametrization:

```
@parameterized.expand([[x] for x in range(4)])
def test_parameterized(self, data):

    self.assertEqual(self.chip["RX"].READY, 1)
```

Parallel execution :

```
pip install pytest-xdist
```

```
pytest -n auto
```

The screenshot shows the output of a pytest run with parallel execution using pytest-xdist. It displays 56 numbered lines, each with a progress bar and a 100.0% completion status. At the bottom, it shows the total number of workers used (56) and the total time taken (1.75s).

```
1 [100.0%]
2 [100.0%]
3 [100.0%]
4 [100.0%]
5 [100.0%]
6 [100.0%]
7 [100.0%]
8 [100.0%]
9 [100.0%]
10 [100.0%]
11 [100.0%]
12 [100.0%]
13 [100.0%]
14 [100.0%]
15 [100.0%]
16 [100.0%]
17 [100.0%]
18 [100.0%]
19 [100.0%]
20 [100.0%]
21 [100.0%]
22 [100.0%]
23 [100.0%]
24 [100.0%]
25 [100.0%]
26 [100.0%]
27 [100.0%]
28 [100.0%]
29 [100.0%]
30 [100.0%]
31 [100.0%]
32 [100.0%]
33 [100.0%]
34 [100.0%]
35 [100.0%]
36 [100.0%]
37 [100.0%]
38 [100.0%]
39 [100.0%]
40 [100.0%]
41 [100.0%]
42 [100.0%]
43 [100.0%]
44 [100.0%]
45 [100.0%]
46 [100.0%]
47 [100.0%]
48 [100.0%]
49 [100.0%]
50 [100.0%]
51 [100.0%]
52 [100.0%]
53 [100.0%]
54 [100.0%]
55 [100.0%]
56 [100.0%]
Mem [100.0%]
Swift [100.0%]
```

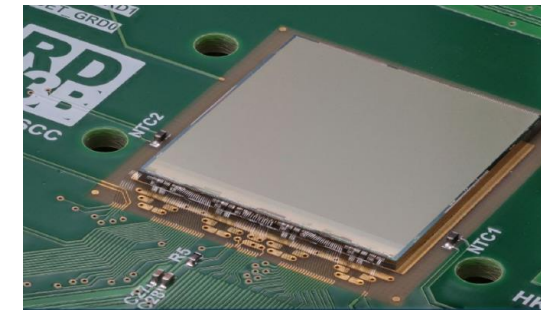
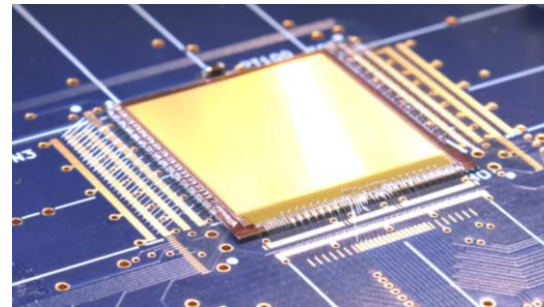
Conclusions

- Moving teams together for better collaboration
- Verification is software, let firmware/software contribute early
- Reused for validation and production testing
- Understand limitations
- Can be an organizational challenge
- Can be done with all open tools and modern infrastructure
- Successfully used to develop and test ASICs (example: <https://doi.org/10.1016/j.nima.2020.164721>)

Contact:

Mail: tomasz.hemperek@dectris.com

LinkedIn: <https://linkedin.com/in/hemperek>



Thank you for your attention!

