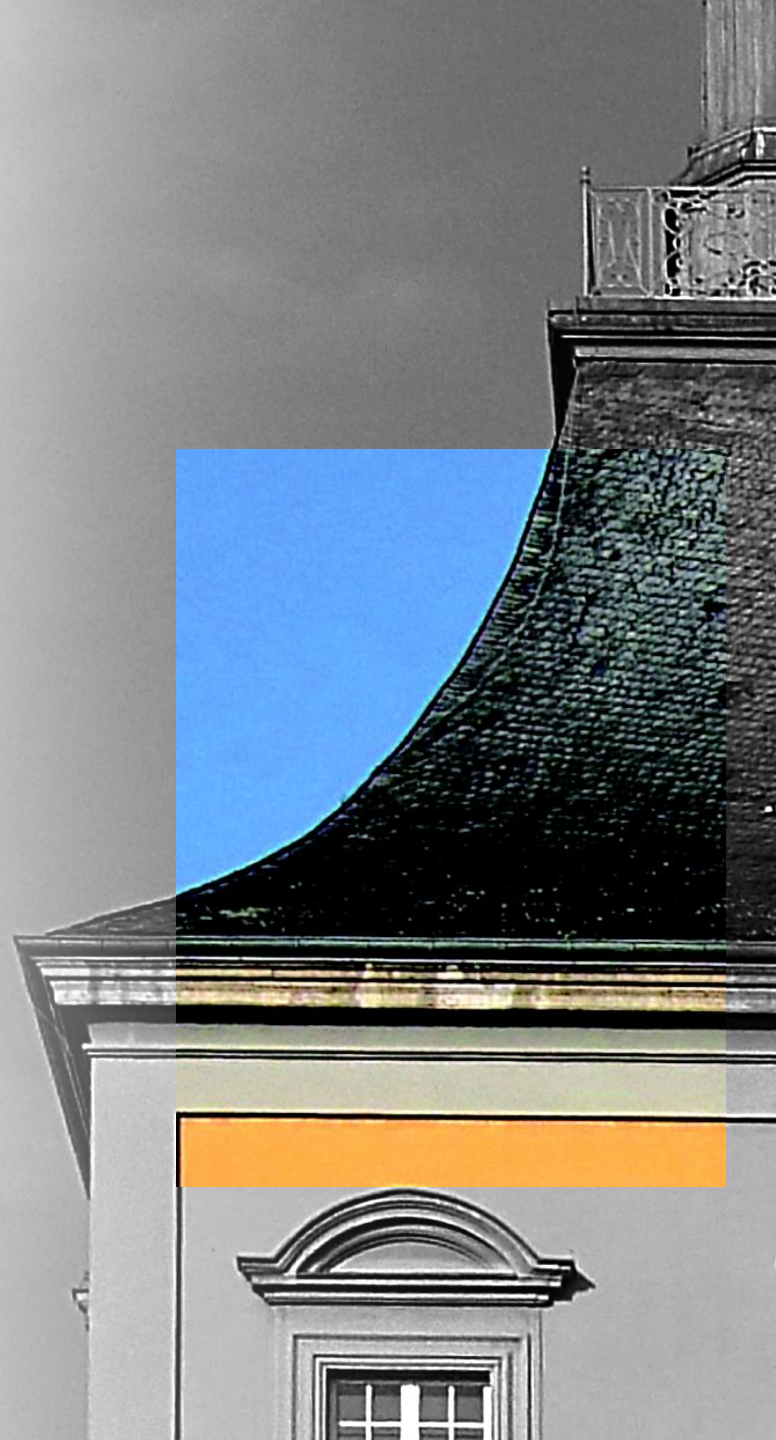
UNIVERSITÄT BONN

SI LAB
Silizium Labor Bonn

DESIGN FLOW

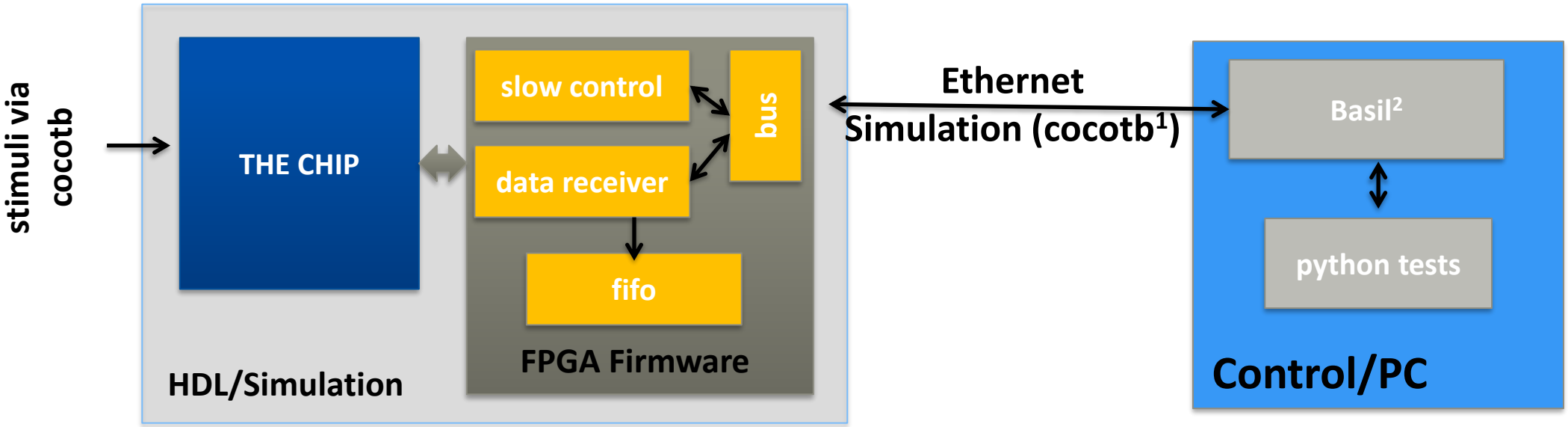
WE USE IN BONN

Tomasz Hemperek



hemperek@uni-bonn.de

Digital design and Verification




```
self.chip['global_conf']['InjEnLd'] = 0
self.chip['global_conf']['TDacLd'] = 0
self.chip['global_conf']['PixConfLd'] = 0
self.chip['global_conf'].set_wait(200)
self.chip.write_global() #send some test hit

while not self.chip['trigger'].is_done():
    pass

print 'fifo_size', self.chip['fifo'].get_fifo_size()
data = self.chip['fifo'].get_data()
```

Exactly the same python code can be used for testing physical chip. Same firmware for FPGA.

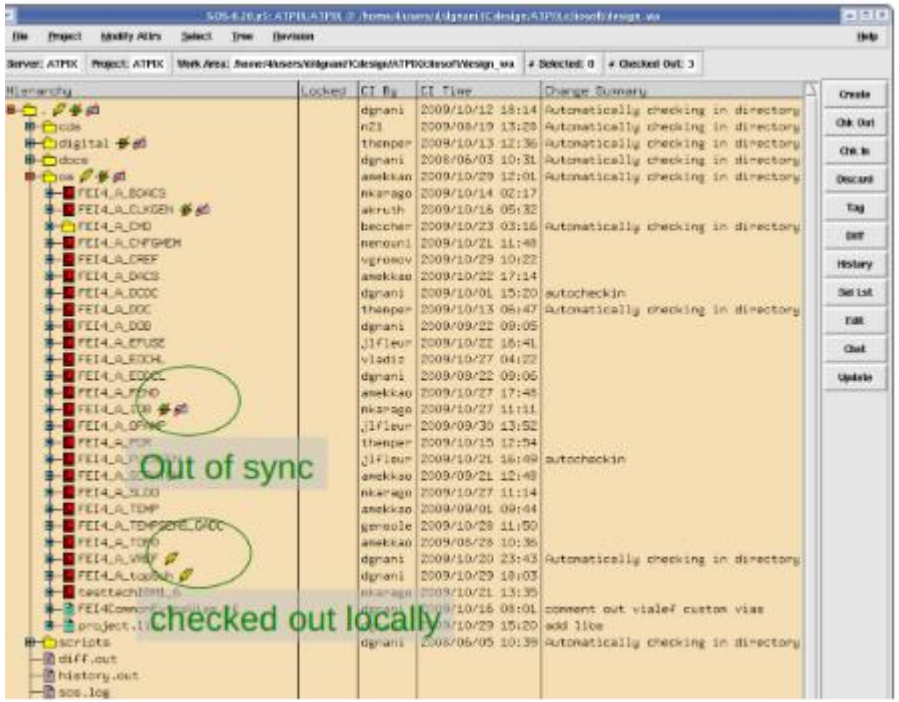
1 https://github.com/potentialventures/cocotb

2 https://github.com/SiLab-Bonn/basil

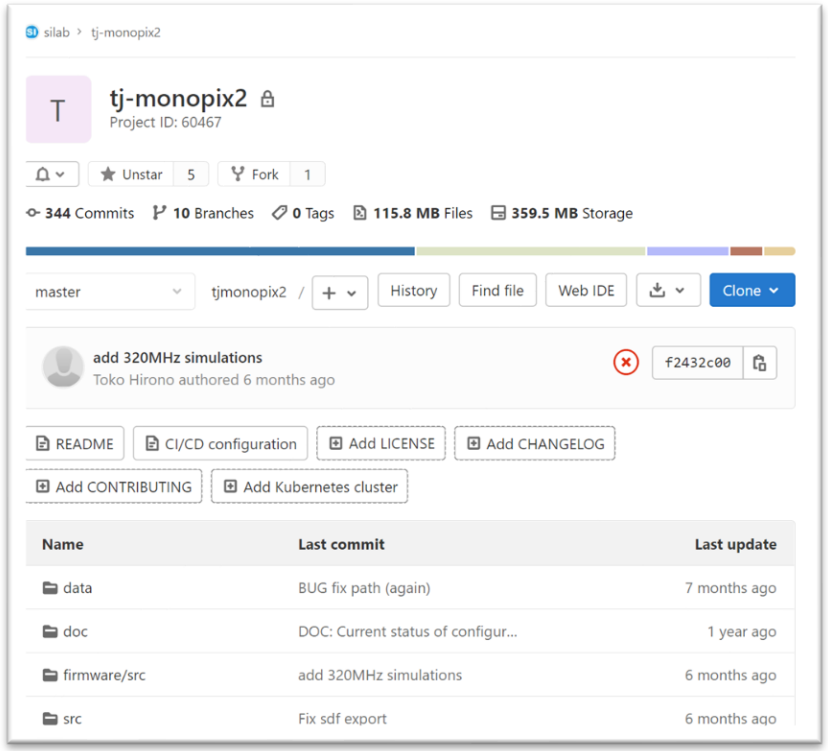
Project organization

Use content management system for everything

SoS for OA



all the test GIT



Continuous Integration

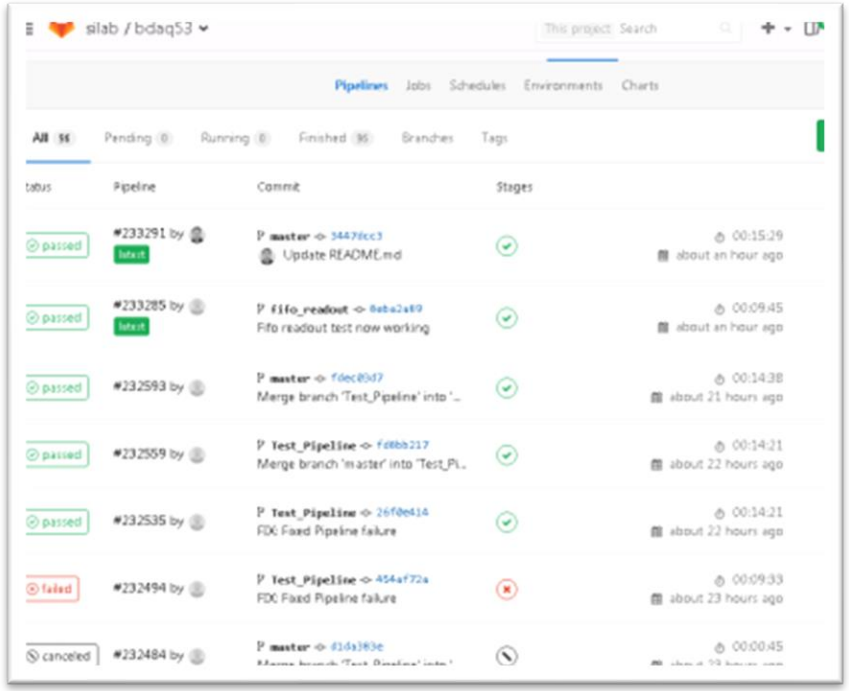
Use pytest* as testing framework

```
def test02_all0(self):
    self.dut["GCR"]["EnROCnfg"].setall(0)
    self.dut["GCR"].write("EnROCnfg", chip_id=self.dut.chip_id, write=True)
    self.dut.get_GCR("EnROCnfg")
    self.assertFalse(self.dut["GCR"]["EnROCnfg"].any())

    hit_csv = utils.startHIT(self.dut, self.hit_file.pop())
    size = self.dut['fifo'].get_FIFO_SIZE()
    for _ in range(10):
        size = self.dut['fifo'].get_FIFO_SIZE()
    self.assertEqual(size, 0)

#@unittest.skip("SKIP")
```

Use GitLab CI* to run test on every code change

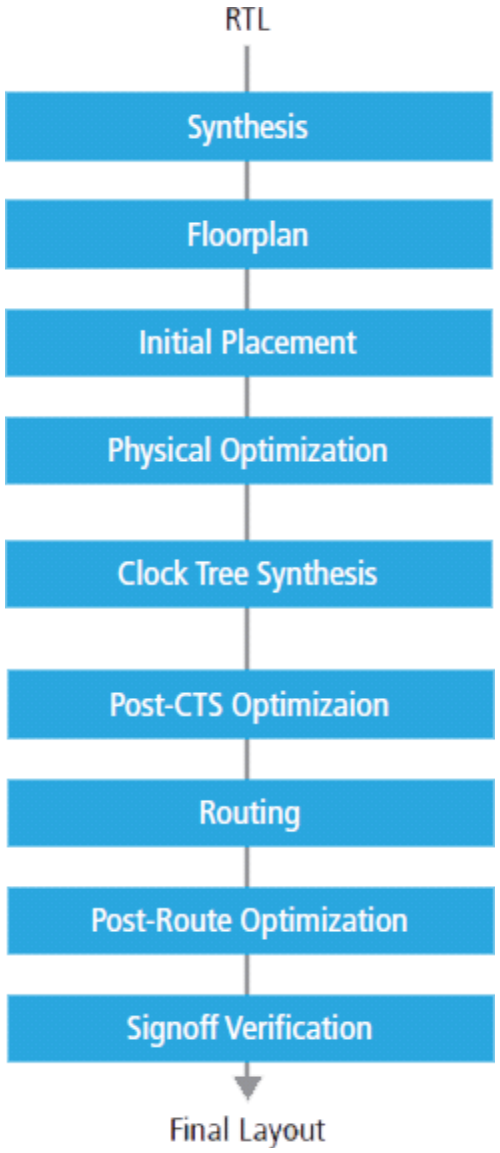


*** or similar**

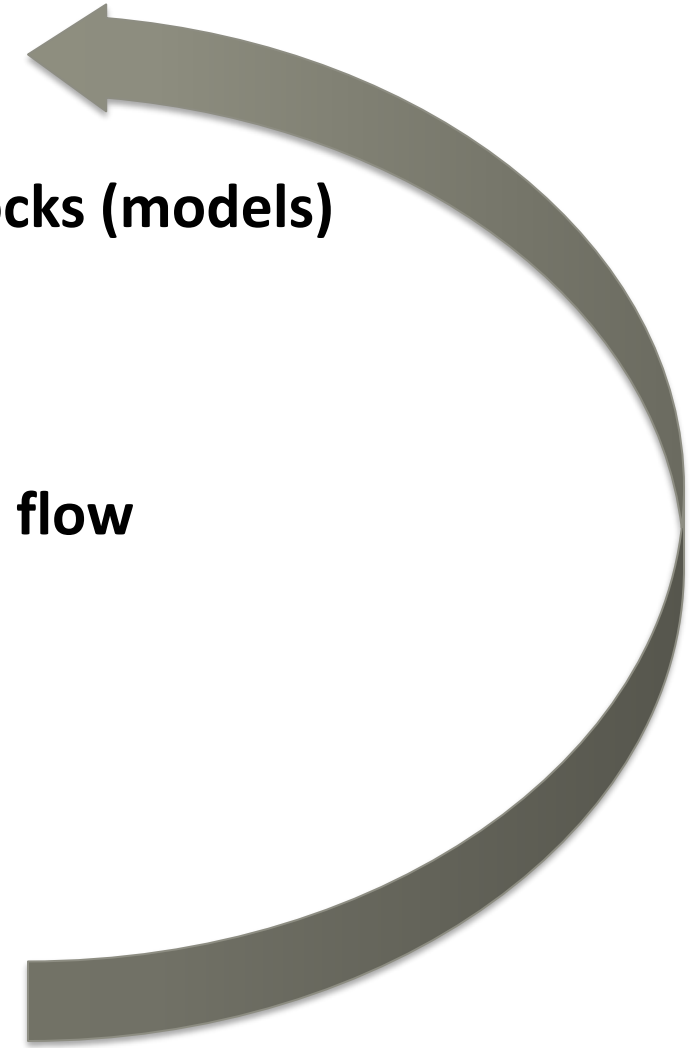
Verification Plan

	Task	test name to launch	Notes	Status
3	TRG new feature	switcher_test		done
4	Gated mode	switcher_test		done
5				
6	Behavior during switching of trigger (read-out) modes ? Needs detailed verification	switcher_test	veto triggered several times and the correct behavior was observed	done
7	row data record and send	extended_test		done
8	pedestal update	extended_test		done
9	hit recognition	vszs_test_vsim		done
10	Test on random data	vszs_test_vsim		done
11	include random pedestals	extended_test		done
12	trigger test	extended_test		done
13	check hammming correction on if working	ecc_test		done
14	read_back_erros	ecc_test		done
15	common mode algorythm(+ overflow)	cm_overflow_test		done

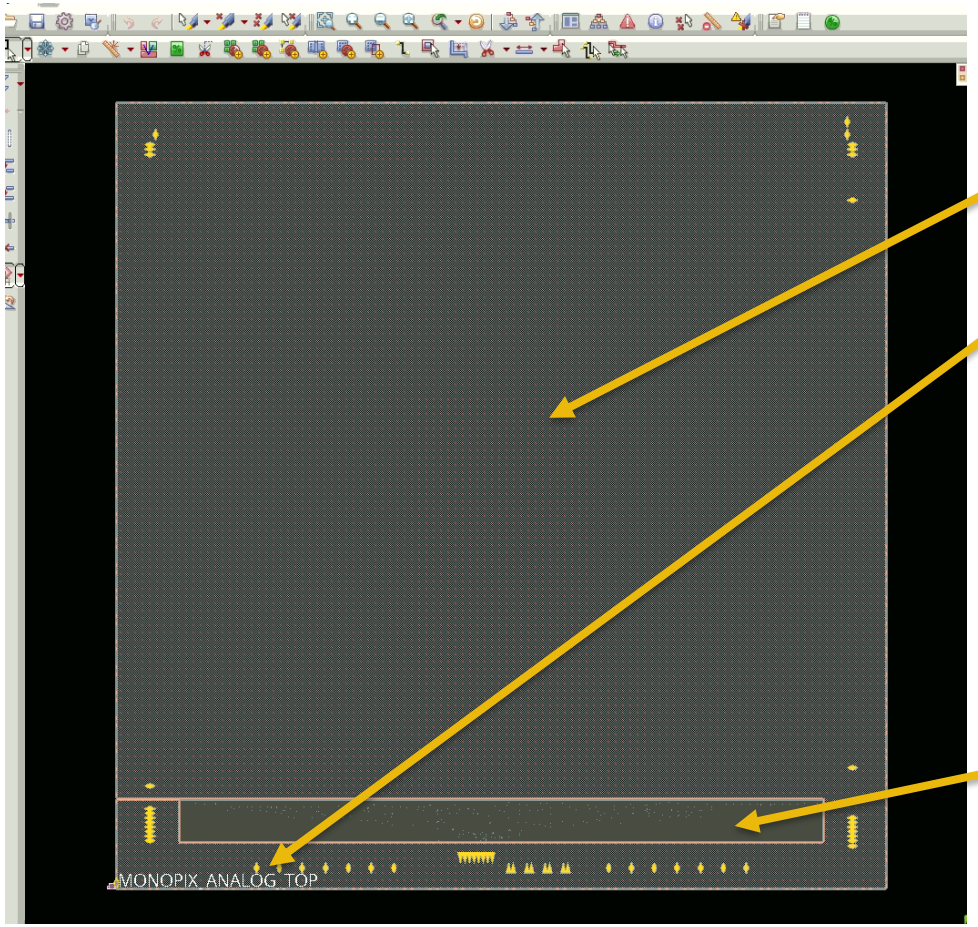
Flow (Digital on Top)



1. **Prepare analog blocks (models)**
2. **Use them in digital flow**
3. **Extract and verify**



Partition



Minimum amount of analog macros (array + periphery) including top power -> simulated and verified with spice

Space for digital logic (see of gates)

IP Design and Modeling

- For all Analog macro
  - Verilog model
  - Liberty file
  - Power grid model (we did not use this yet)
  - Proper netlist for LVS
  - Verified in spice

Signoff

- Timing
- Power
- IR/EM
- Gate level simulation
- FastSpice (with limited pixel array size)

Flow (Digital on Top)