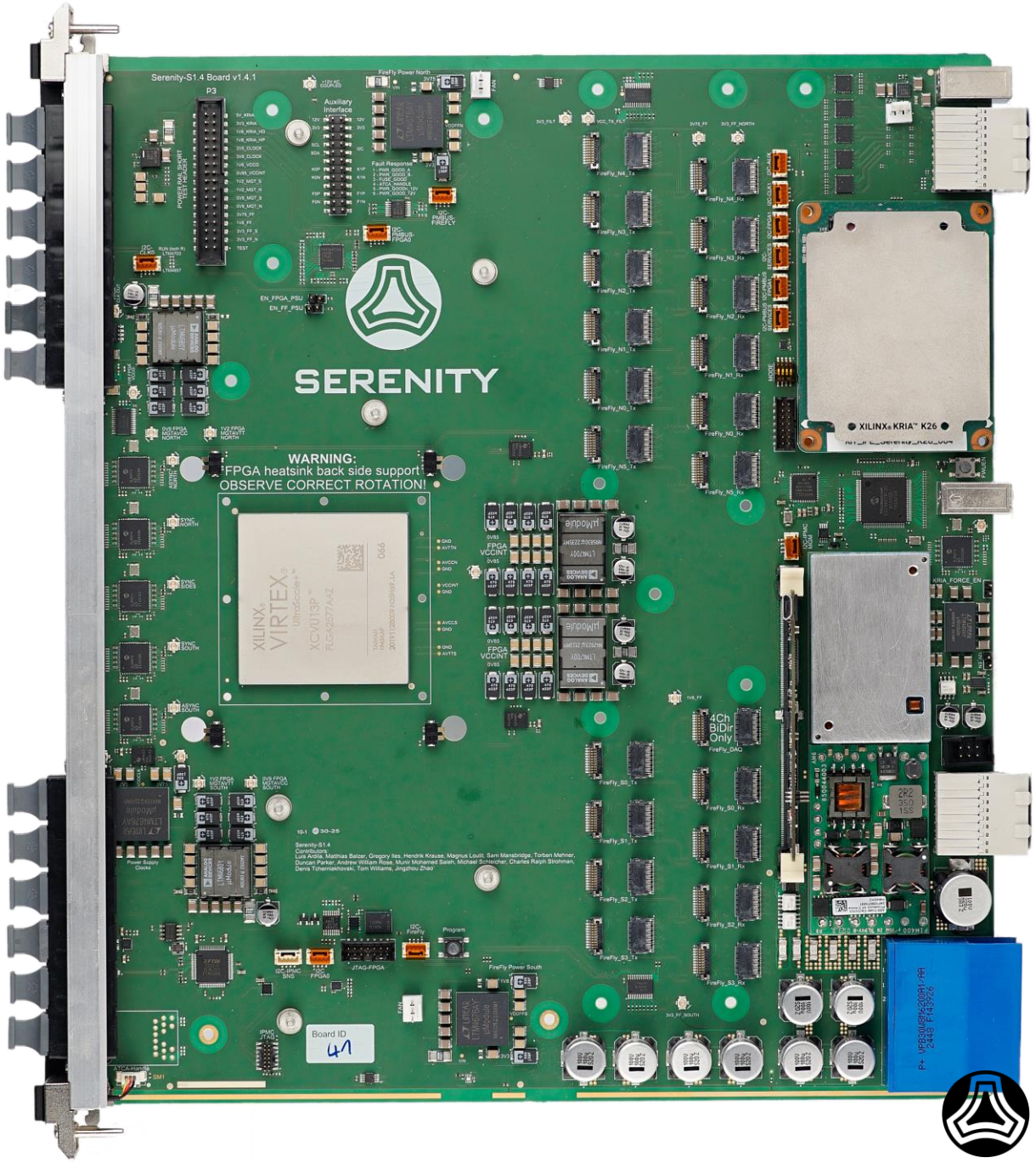
Factory Acceptance Test for Serenity-S1, a CMS Phase-2 Back-End Card

Hendrik Krause on behalf of the Serenity consortium

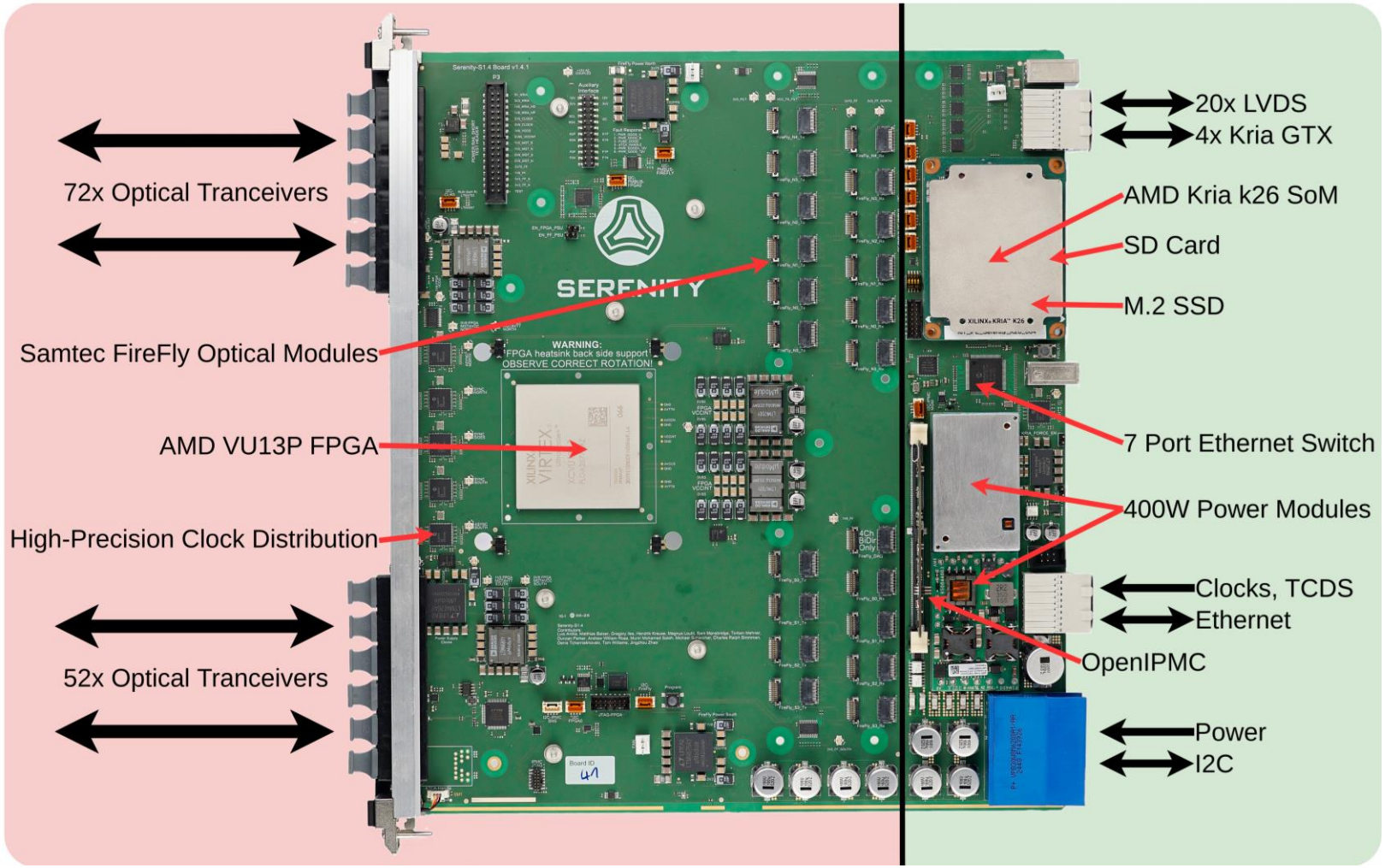
IPE-Seminar, 2026-03-12

What Is Serenity-S1 and Why Do We Need It?

- HL-LHC upgrade requires replacing and updating several sub-detector systems within CMS
- Upgrade covers both frontend and backend electronics
- Serenity-S1: common ATCA-based back-end board
  - Generic FPGA processing card
  - Meets requirements of multiple sub-detectors
- Adopted by 7 sub-detector systems
  - More then 700 boards are being produced and require testing



TWEPP 2023 [Talk] T. Mehner: Lessons learned while developing the Serenity-S1 ATCA card



Architecture

Tech Rxiv	[Paper] T. Mehner et al.:Serenity-S1 - A Versatile ATCA Processing Card for the CMS Phase-2 Upgrade

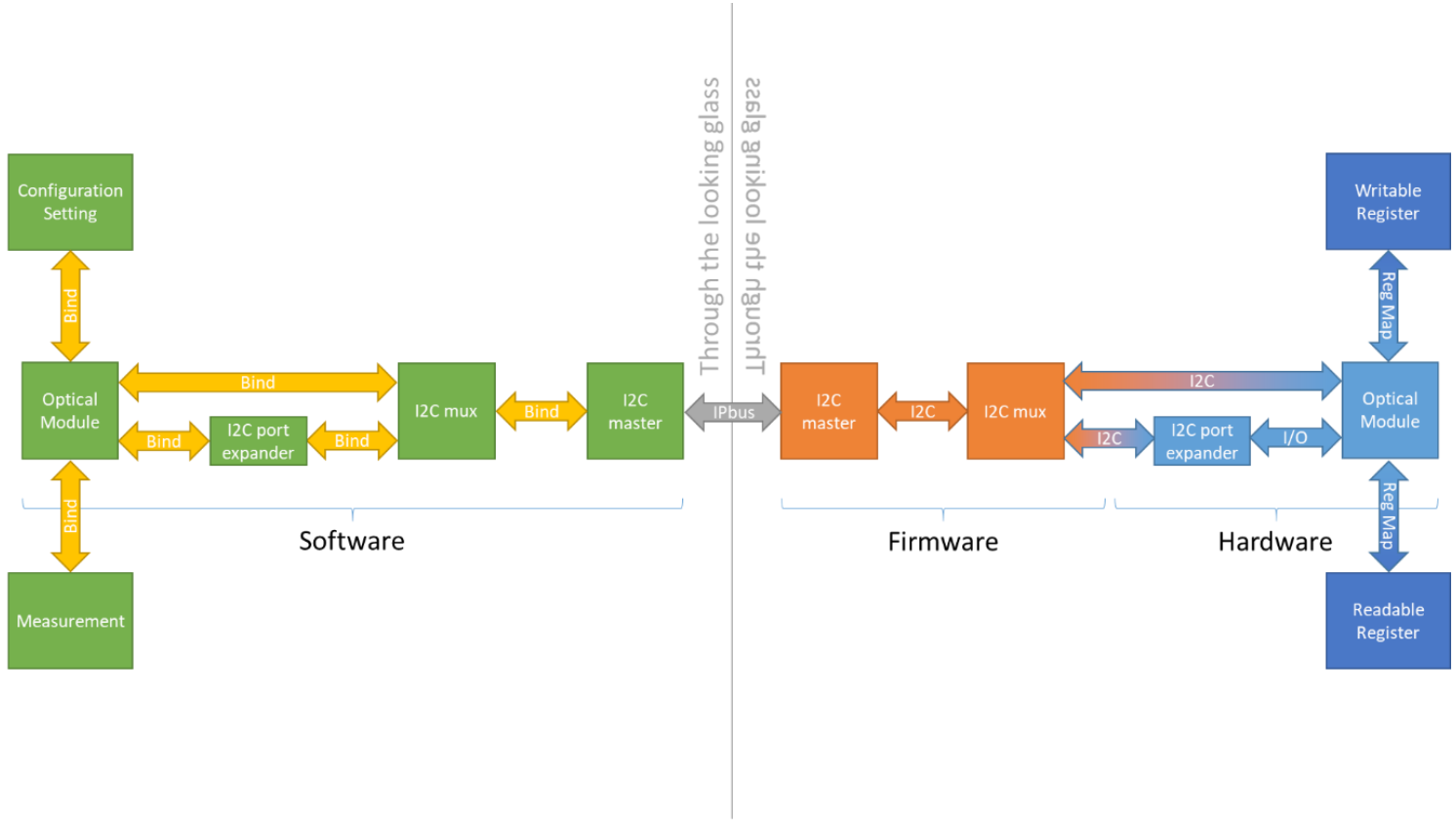
The Serenity-S1 is a versatile data processing card with...

- ...3.1 Tbps digital bandwidth using FireFly optical modules
- ...a large VU13P FPGA for data processing

SMASH

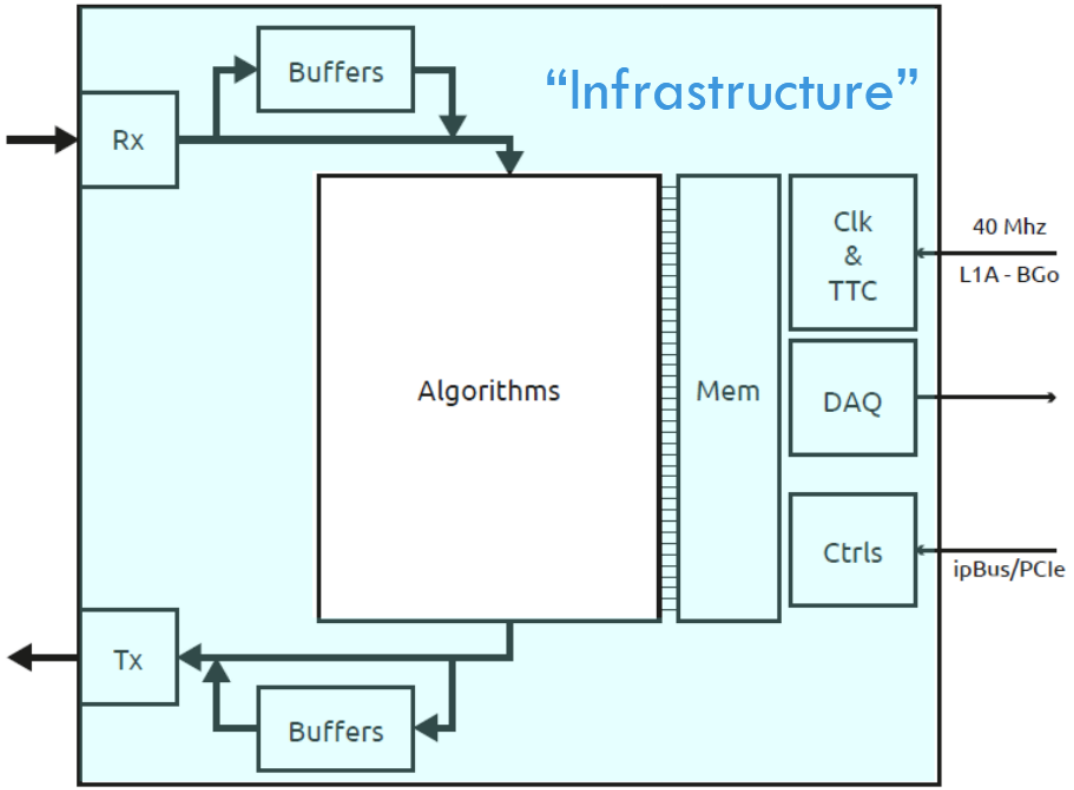
Serenity MAnagement SHell (SMASH)

- Software providing hardware abstraction for slow control
  - Comparable to Linux device tree
  - IPBus based
- Different usage options
  - Interactive shell (see below)
  - SMASH scripts
  - (Limited) Python interface




```
[root@mgmt-kki ~]# smash.exe -i
  #######  Welcome to the interactive SMASH shell.                    #######
  #######  Write SMASH commands and hit <enter> to run commands  #######
  #######  Hit the 'esc' key to quit                                 #######
>> Power:Sequencer Measure VOUT00
Power:Sequencer Measure VOUT00
 VOUT00 : +968.000mV
>>
```

EMP Framework



Extensible, Modular data Processor (EMP)

- Framework providing hardware abstraction for the FPGA
  - Algorithm can use provided buffers for high speed access
  - Control through IPBus registers (over AXI Chip2Chip)
- EMPbutler software can access the EMP framework
  - Interactive CLI or Python API

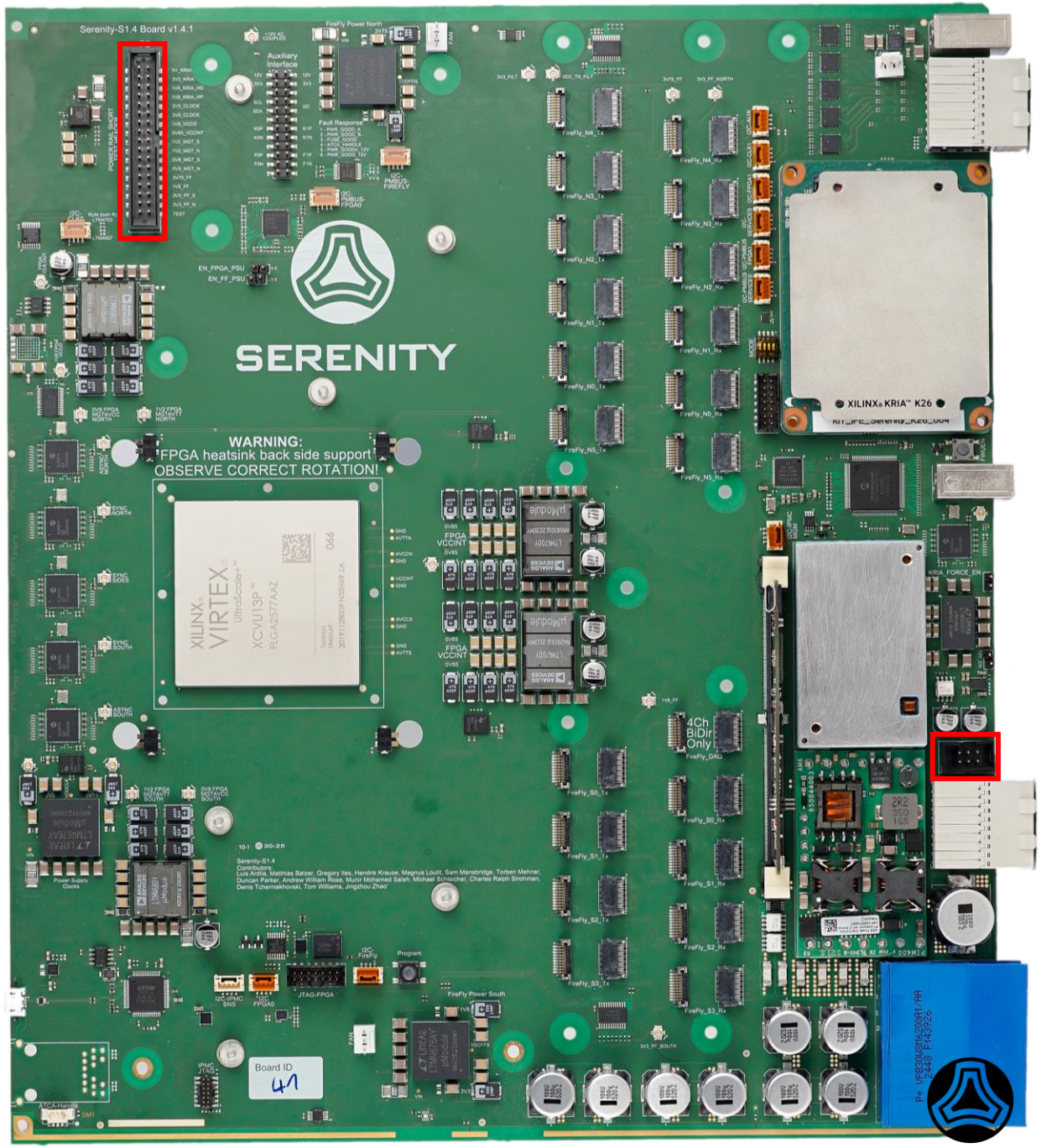
Example to reset FPGA and play and receive data:

```
$ empbutler –c CONNECTIONS.xml do DEVICE_ID reset internal

$ empbutler -c CONNECTIONS_FILE.xml do DEVICE_ID buffers tx PlayOnce -c CHANNEL_LIST --inject generate://pattern

$ empbutler -c CONNECTIONS_FILE.xml do BOARD_ID buffers tx Capture -c TX_CHANNEL_LIST
$ empbutler -c CONNECTIONS_FILE.xml do BOARD_ID capture --rx RX_CHANNEL_LIST --tx TX
```

JINST [Paper] T. Williams et al.: EMP: a common infrastructure firmware framework for the CMS phase-2 upgrades

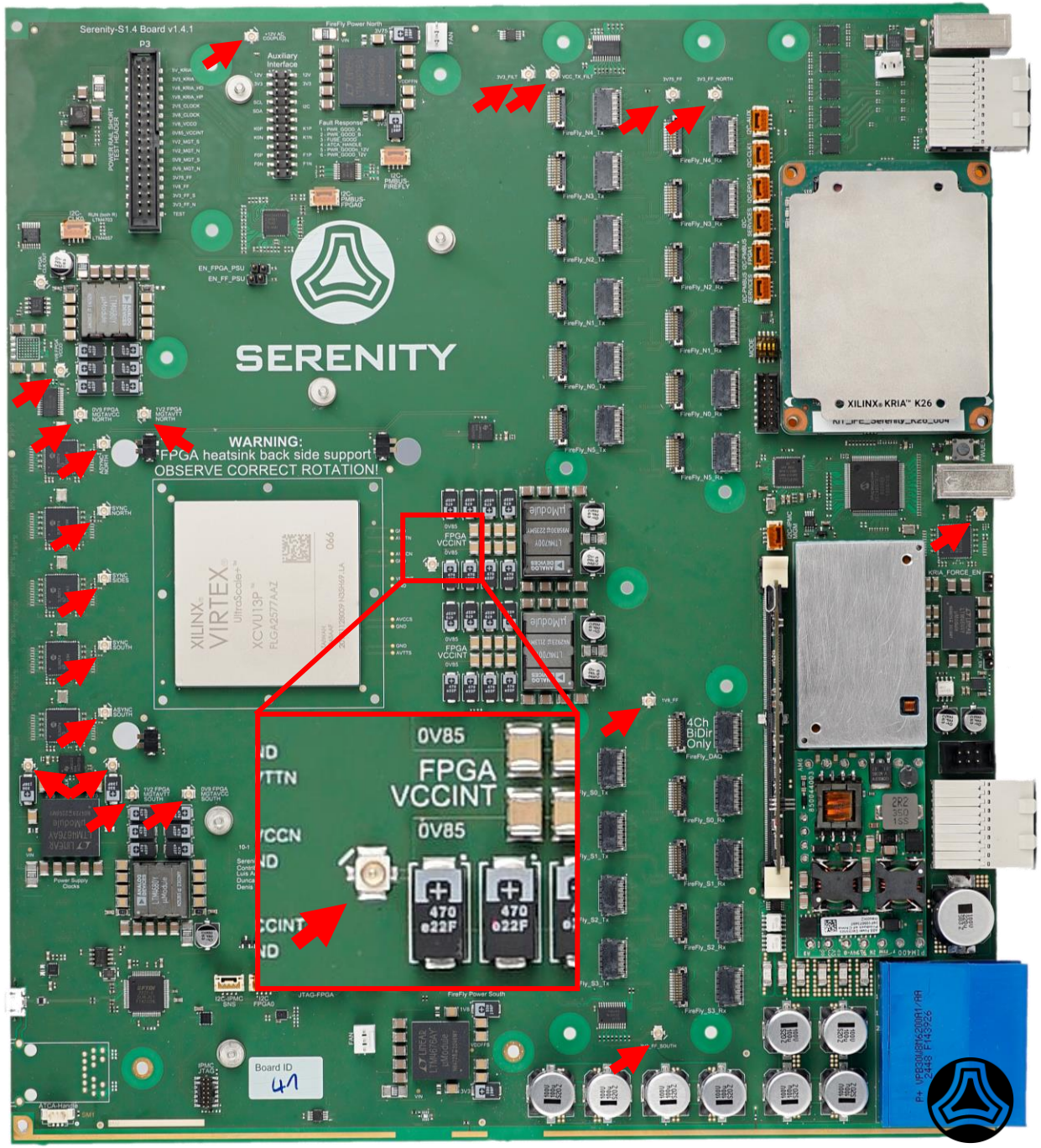
Design for Debug

- Dedicated test header for short circuit detection
  - Access to the main power rails



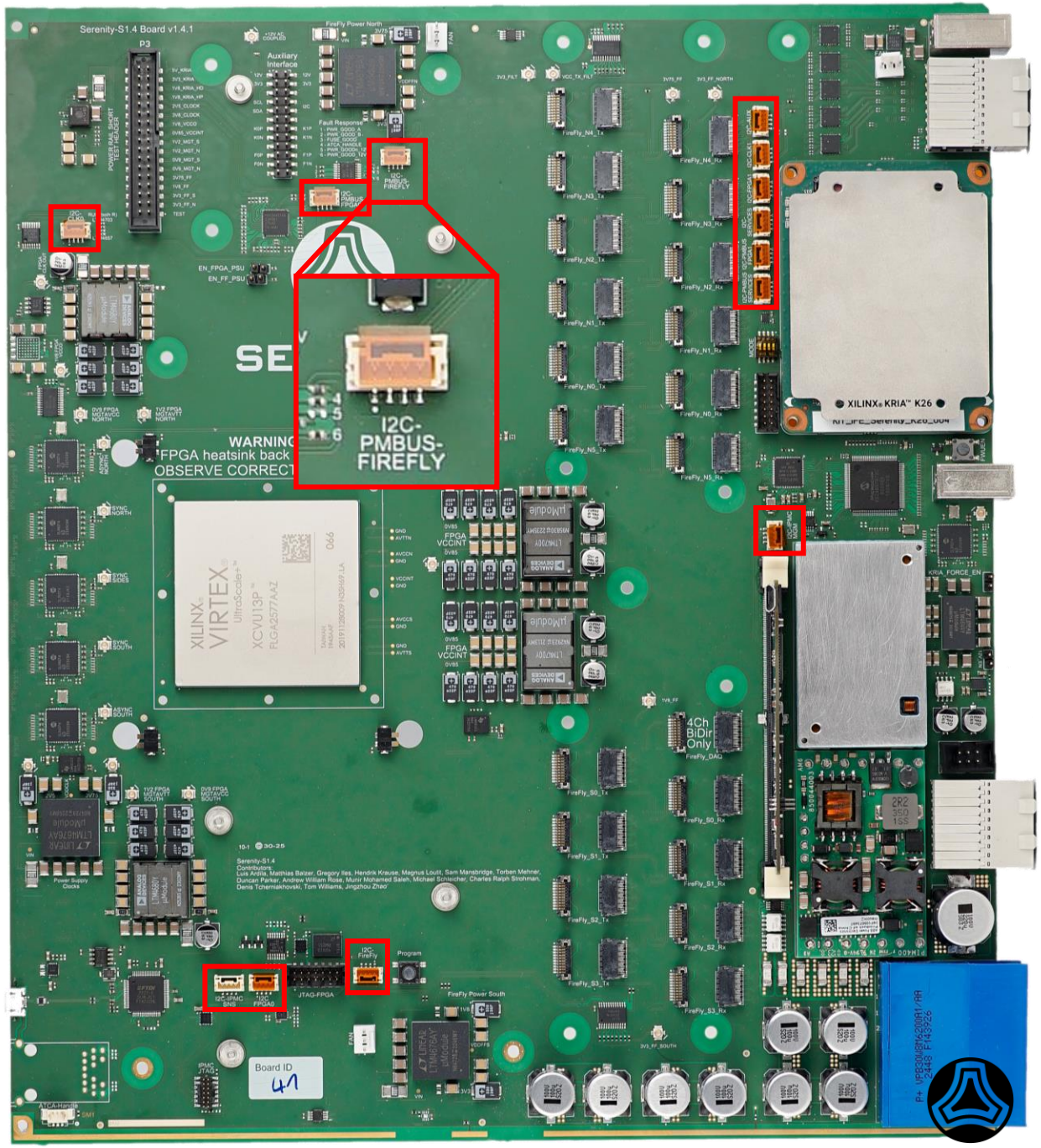
Design for Debug

- Dedicated test header for short circuit detection
  - Access to the main power rails
- Test points for the most important nets (power and clocks)
  - uFl connectors allow the easy connection to scopes



Design for Debug

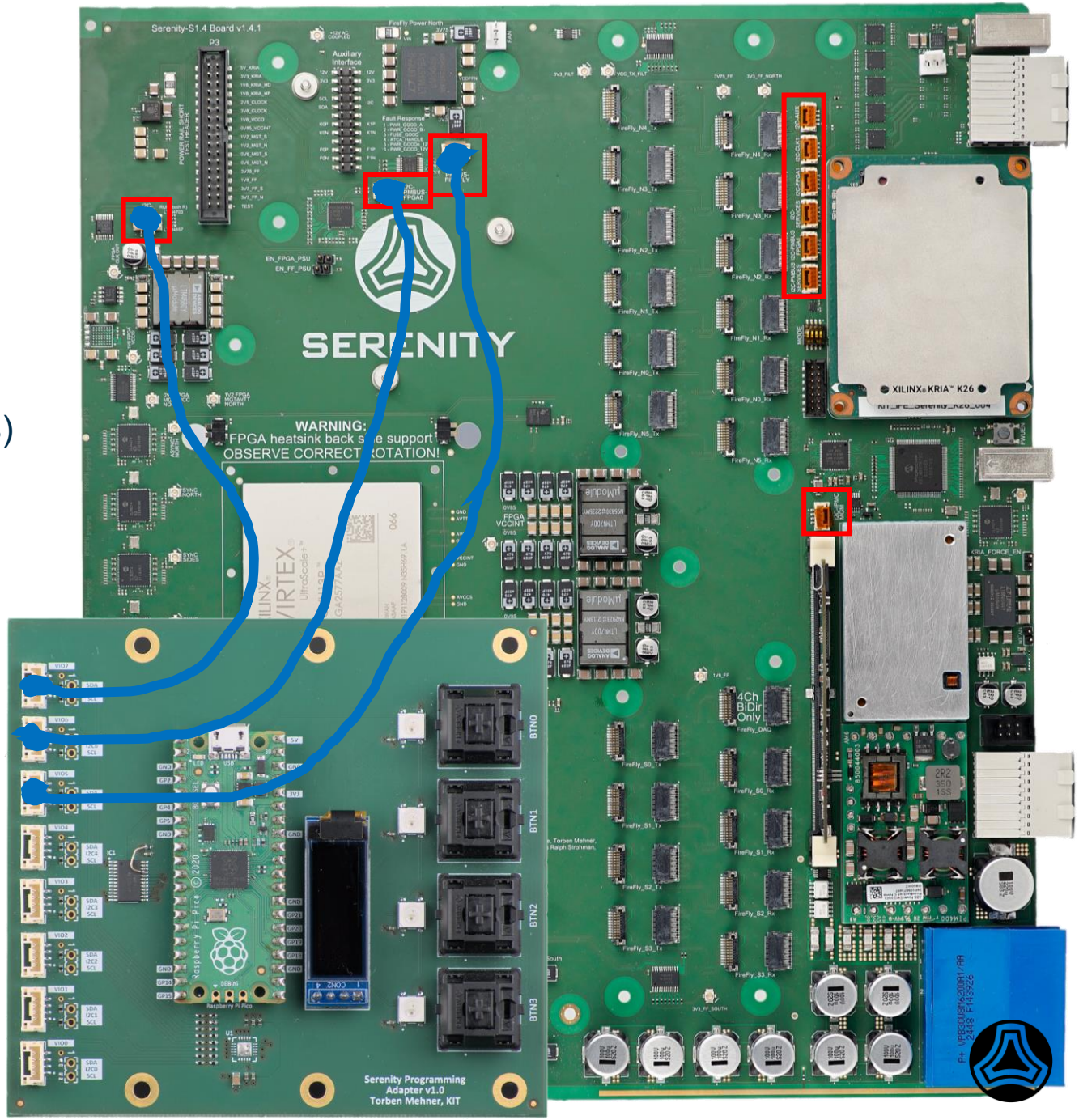
- Dedicated test header for short circuit detection
  - Access to the main power rails
- Test points for the most important nets (power and clocks)
  - uFl connectors allow the easy connection to scopes
- Split the slow control (I2C, PMBus) into individual nets
  - Use dedicated test header for each net



Design for Debug

- Dedicated test header for short circuit detection
  - Access to the main power rails
- Test points for the most important nets (power and clocks)
  - uFl connectors allow the easy connection to scopes
- Split the slow control (I2C, PMBus) into individual nets
  - Use dedicated test header for each net
- Developed the “Serenity Programming Adapter”
  - Raspbery Pi Pico + MicroPython
  - Easy external debugging of I2C busses and components

MST Group Meeting [Talk] T. Mehner: Improving Testing and Commissioning



Multi-Stage QA Strategy

PCB Manufacturer

Manufacturer QC

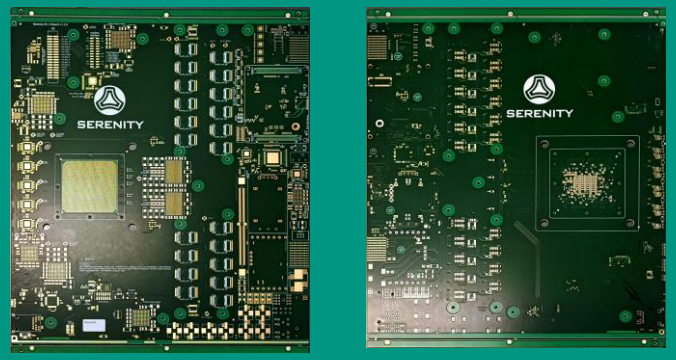
- Micro section analysis
- Surface finish
- Impedance measurement
- Solderability
- Ionic contamination
- …

Multi-Stage QA Strategy

PCB Manufacturer

Manufacturer QC

- Micro section analysis
- Surface finish
- Impedance measurement
- Solderability
- Ionic contamination
- …



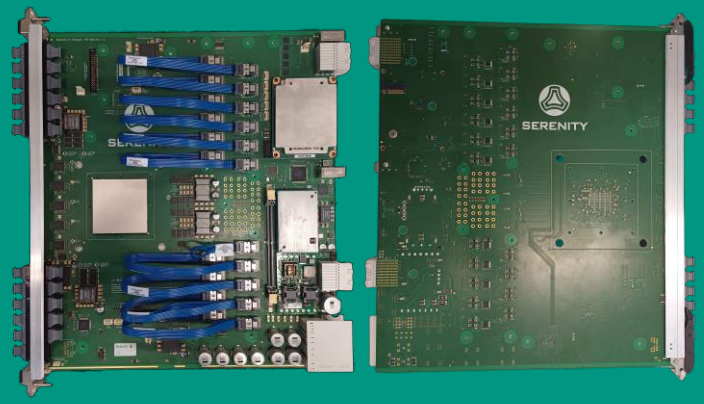
PCB Assembler

Assembler QC

- Visual Inspection
- Automated Optical Inspection (AOI)
- X-Ray

Functional Testing

- Factory Acceptance Test (FAT)

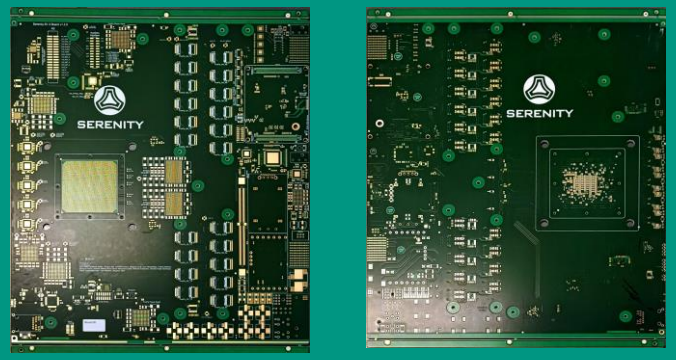


Multi-Stage QA Strategy

PCB Manufacturer

Manufacturer QC

- Micro section analysis
- Surface finish
- Impedance measurement
- Solderability
- Ionic contamination
- …



PCB Assembler

Assembler QC

- Visual Inspection
- Automated Optical Inspection (AOI)
- X-Ray

Functional Testing

- Factory Acceptance Test (FAT)

CERN

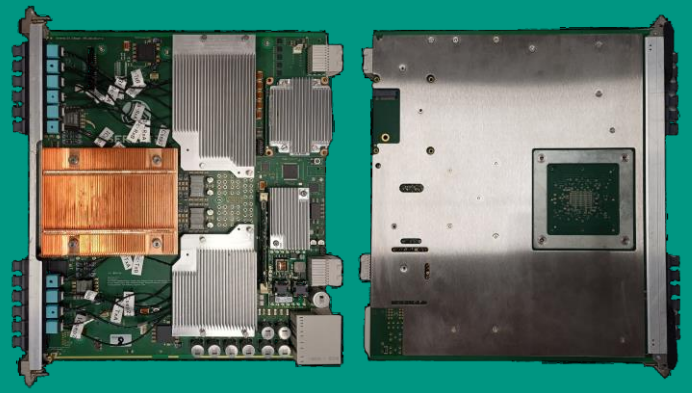
User Acceptance Test

- Repetition of the FAT with more focus on SerDes links

Add heatsink, optics, back cover, OpenIPMC, Kria,…

Sub-detector specific testing

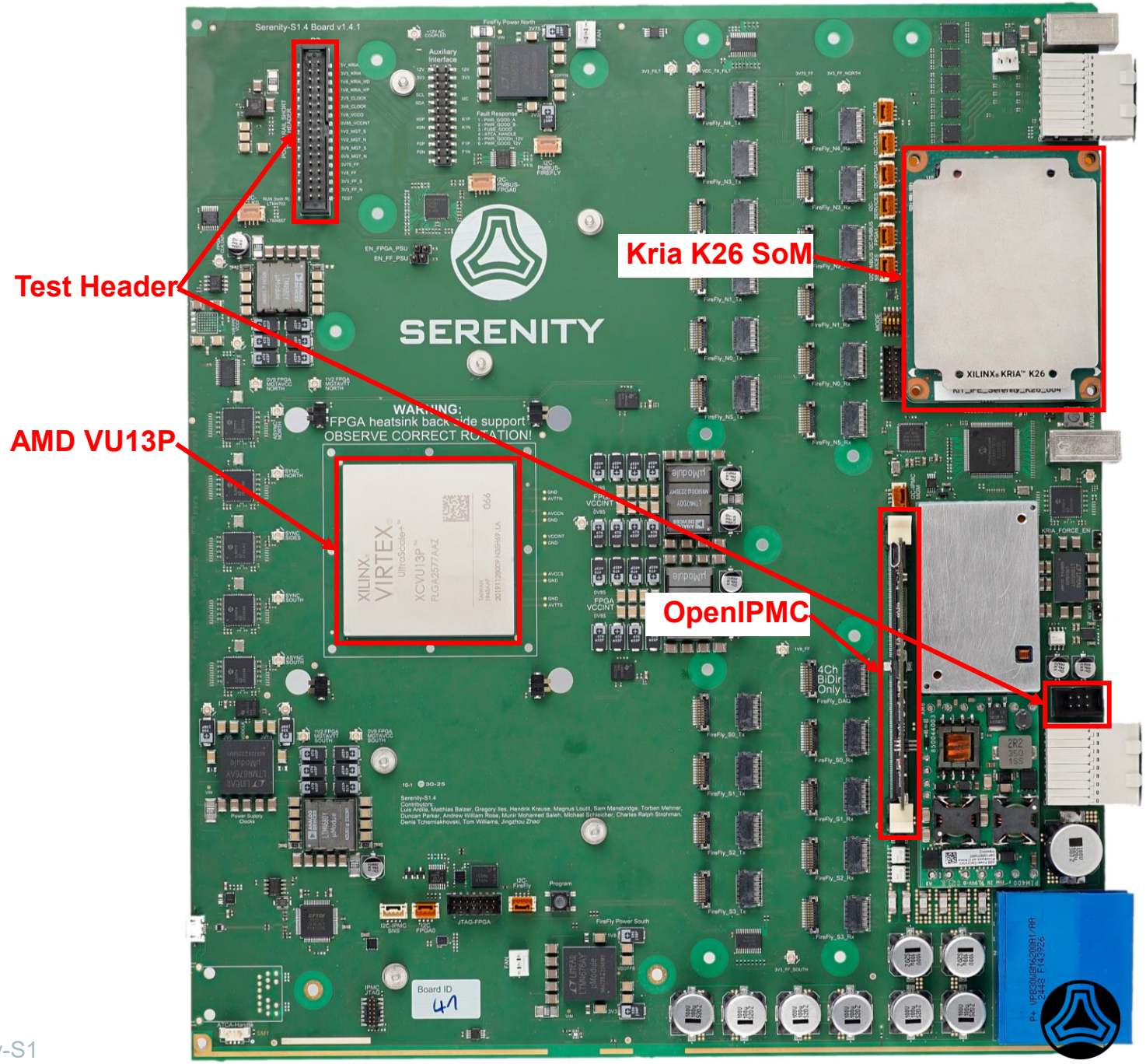
- Burn-in test



FAT Scope

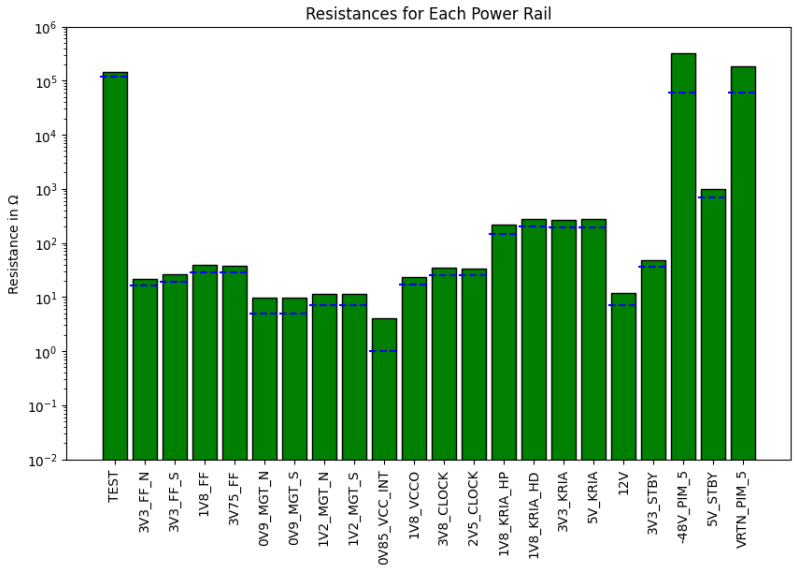
Goals

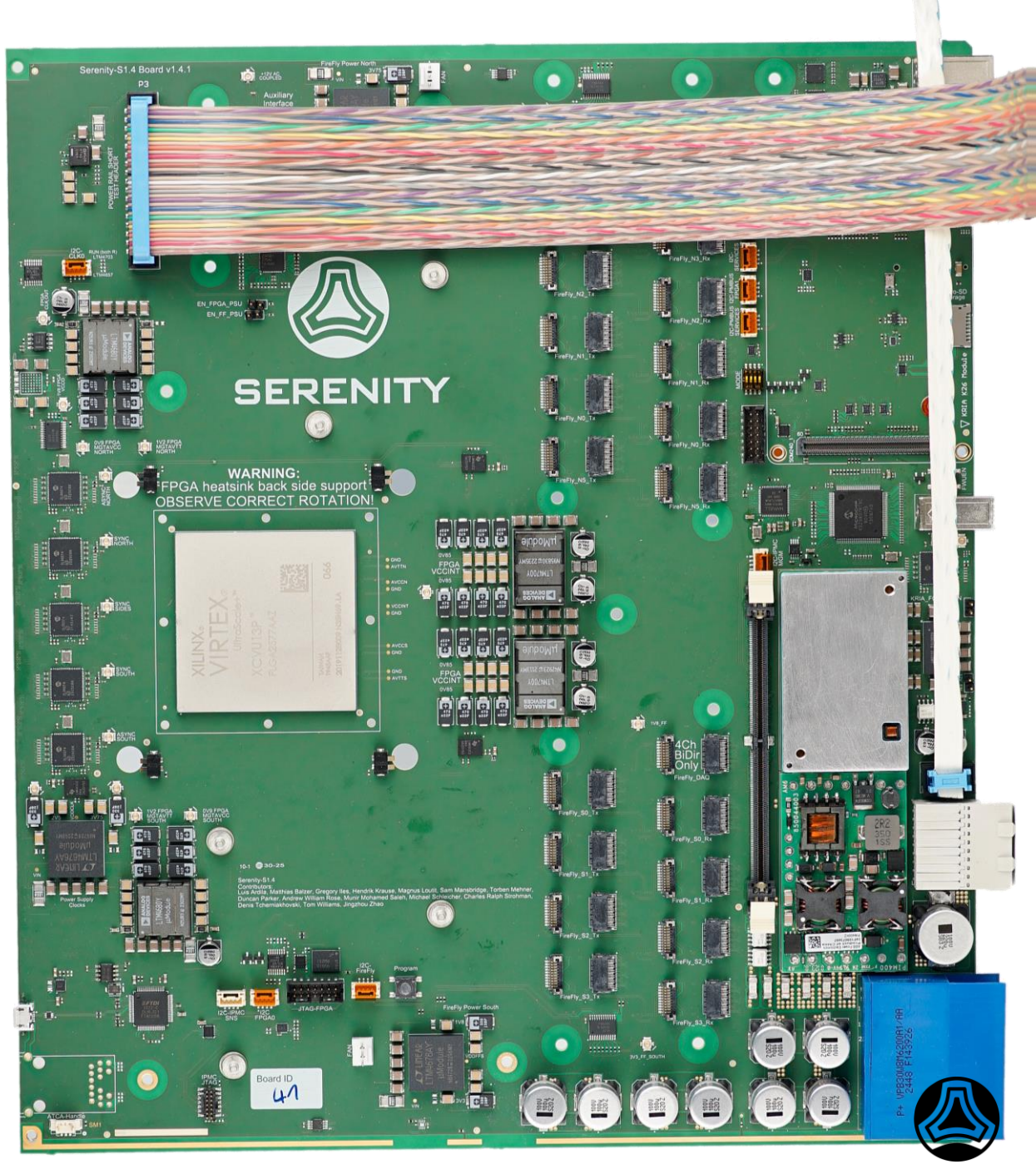
- Functional testing directly at the manufacturer
  - Short repair cycles
- End-of-line test, automated <10 min
- Run by non-experts
- Onboard self-test (OpenIPMC, Kria SoM, FPGA)
- Minimal external equipment
- Reusable for maintenance/diagnostic
- Focus on most important functionality
  - Power, slow control, SerDes links



FAT Procedure – Step 1: Short-Circuit Test

- Use off-the-shelf test equipment
- Measure the resistance of every power rail
- Comparison against threshold
- Controlled via test PC





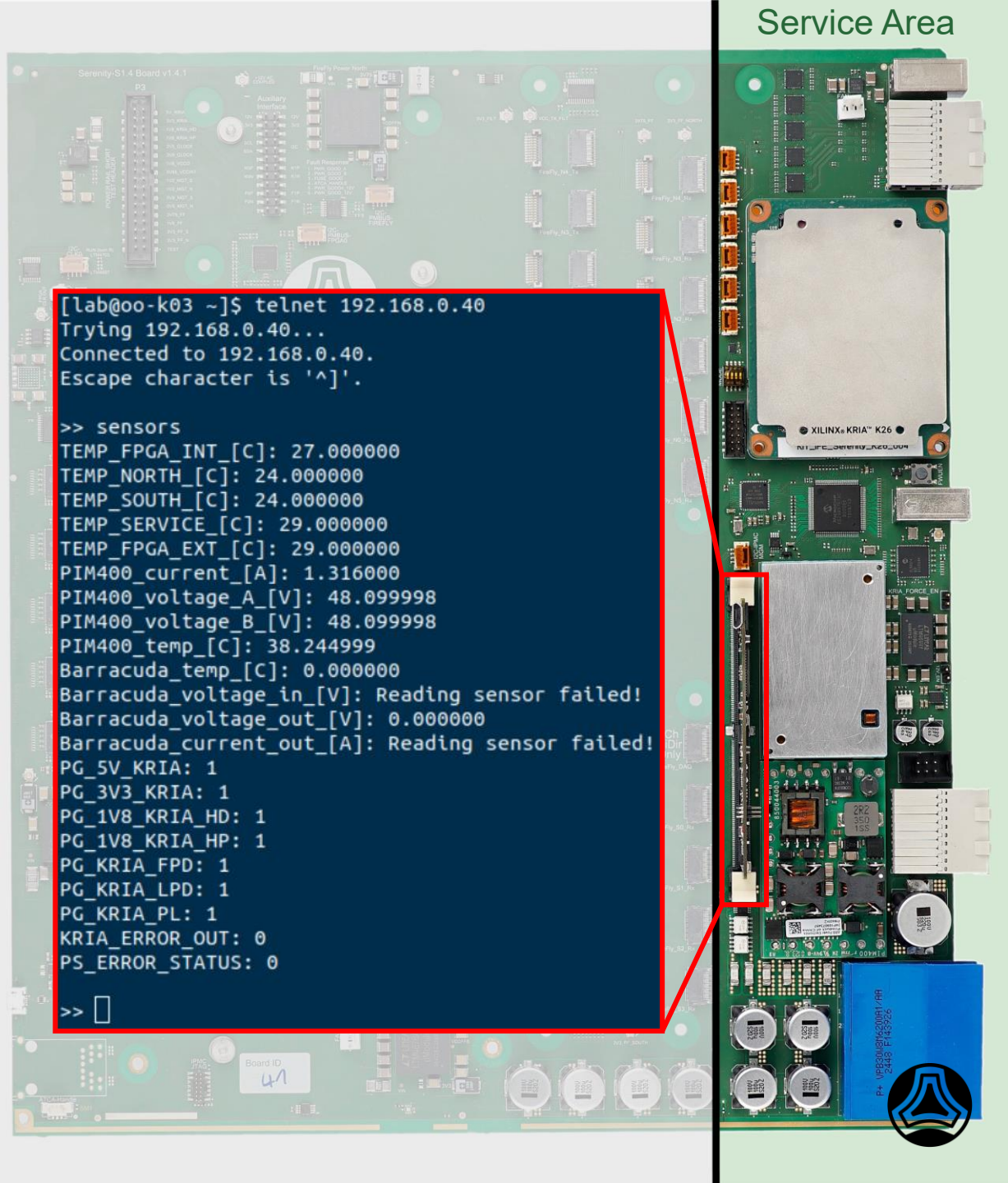
FAT Procedure – Step 2: Service Area Tests

OpenIPMC

- Open-source ATCA IPMC implementation
- Monitors temperature and power
- Reports to the Shelf Manager

Test Approach

- Expose sensor readings of the OpenIPMC to a telnet interface
- Python access via telnetlib from a test PC
- Offline check that the reading was successful and the values are plausible



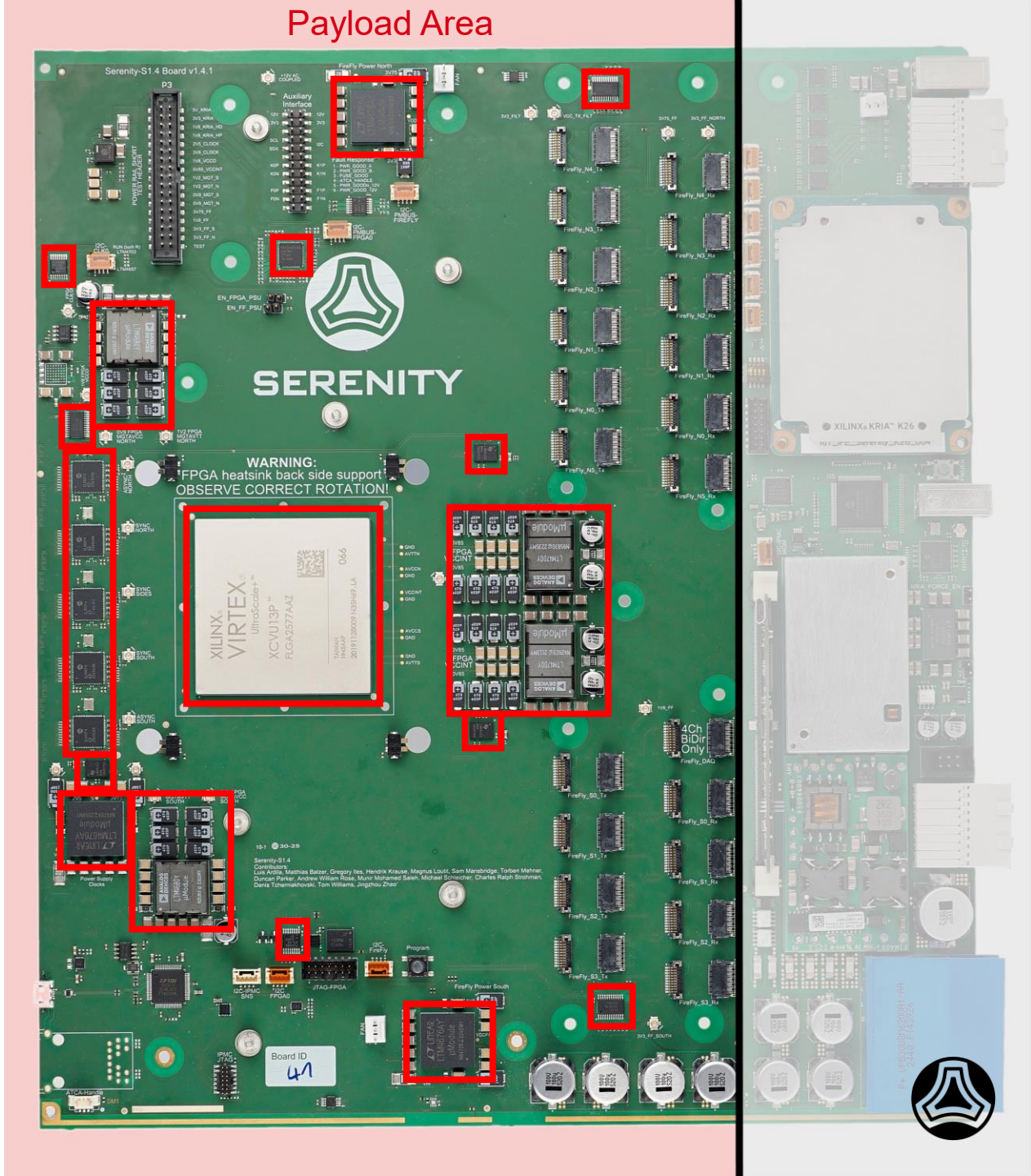
FAT Procedure – Step 3: Payload Area Tests

Challenge

- Verify that all components in the slow control tree work
- Must configure power supplies & sequencer
- Must configure clock chips
- Store board ID in EEPROM

Advanced board management: SMASH

- Framework on the Kria SoM
- Provide access to all I2C and PMBus devices + JTAG
- Build in mechanisms for measure, configure and verify
- Python API → use software test frameworks for hardware



FAT Procedure – Step 3: Structured Testing With Pytest

- SMASH Python API
  - Wrapped into Pytest functions
- Pytest features
  - Parameterization
  - pytest-json → results for reporting
  - pytest-dependency → enforce safe sequence
- Guarantees: if PSU fails → stop powering the payload area

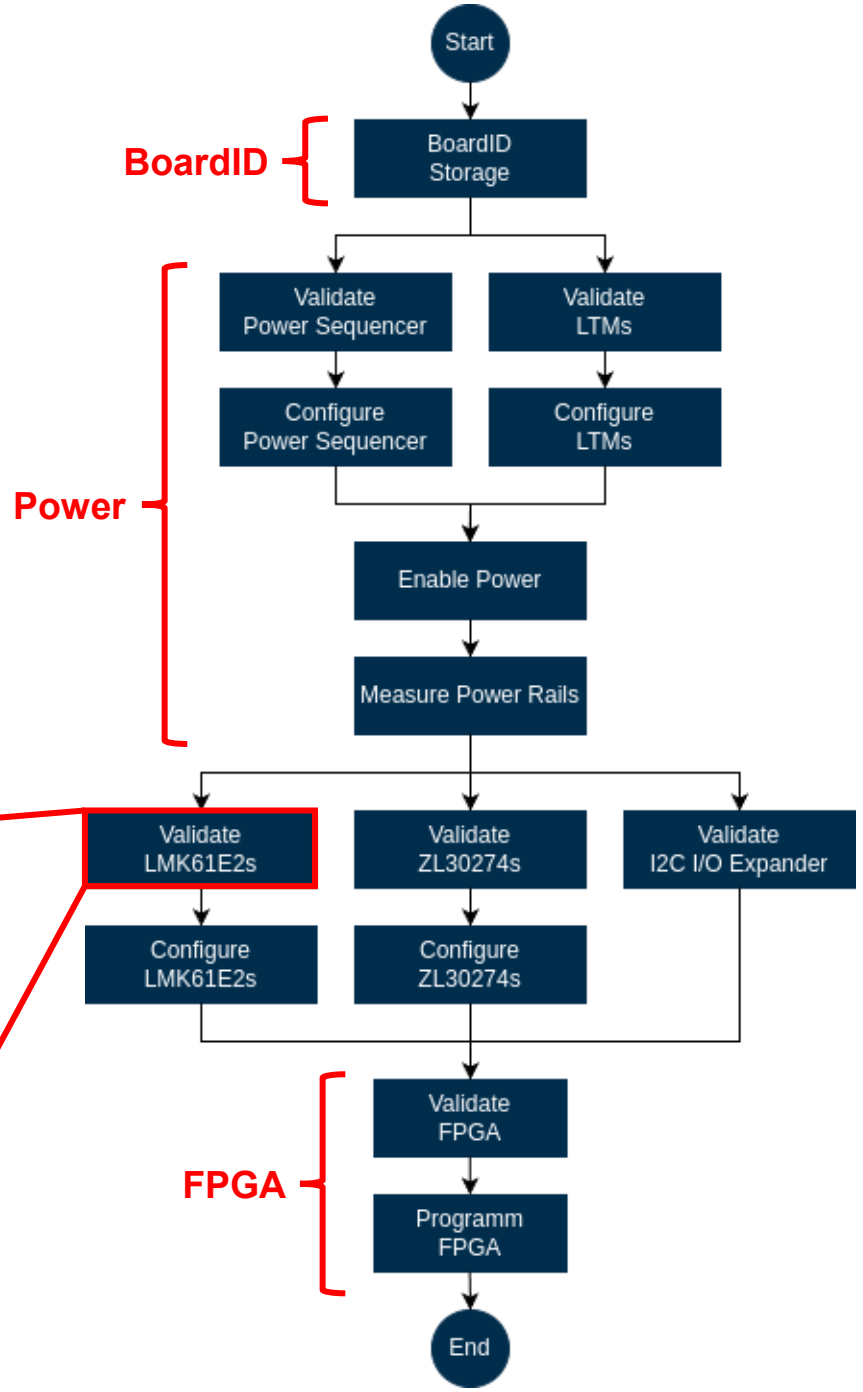
```python
@pytest.mark.dependency(depends=
    instances("test_measure_power_rails", param_nominal_rail_voltage))
@pytest.mark.parametrize("name, designator", param_LMK61E2)
def test_validate_lmk61e2(smash_, name, designator):
    """
    Validate LMK61E2.

    Use the SMASH validate function to test if the
    LMK61E2 respond with the correct device ID.
    """
    answer = smash_.getElement(designator).validate()

    assert answer == True, "Validate failed for %s (%s)" % (name, designator)
```

Dependencies

Parameter

SMASH API call

Evaluation

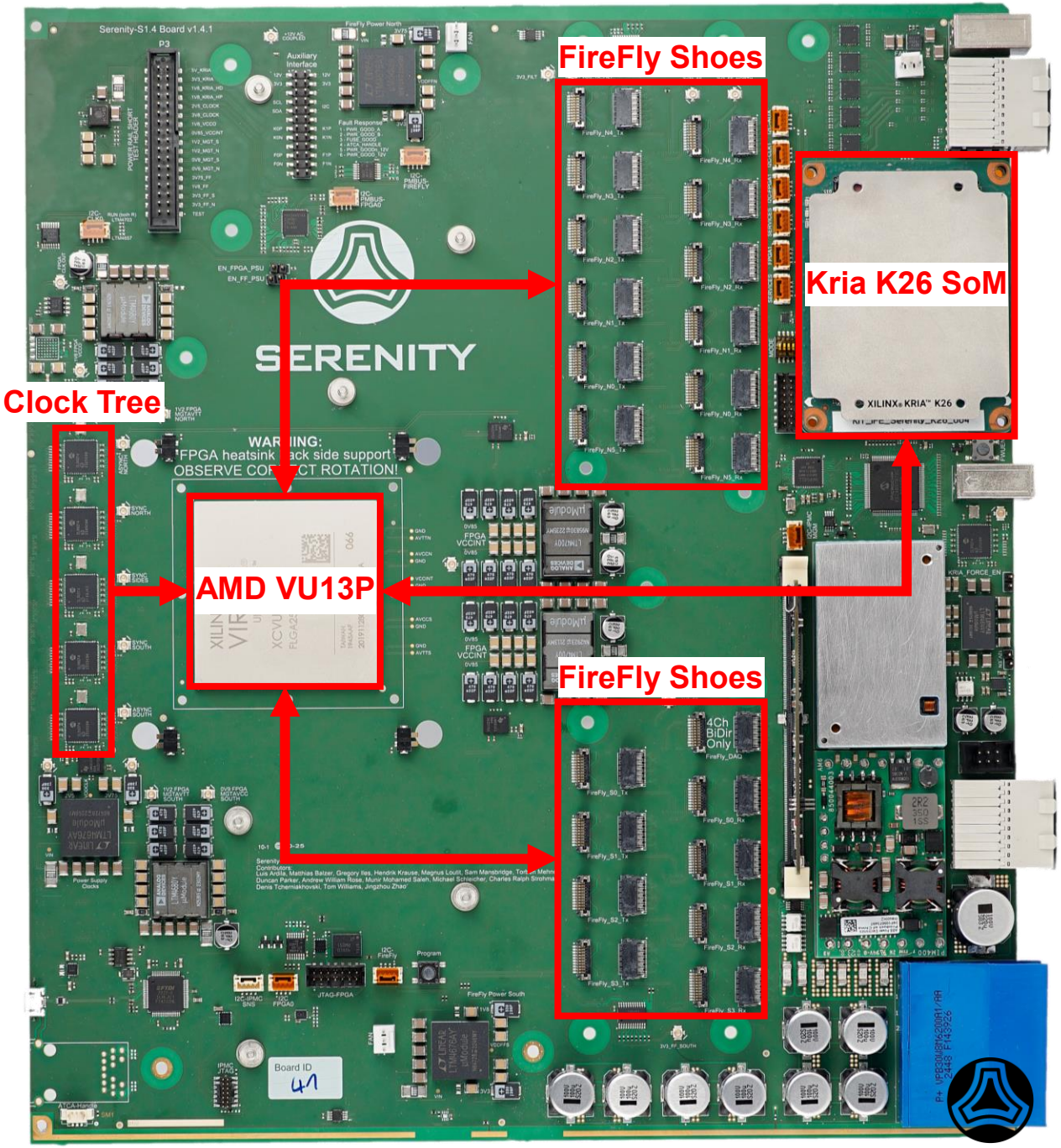


FAT Procedure – Step 4: FPGA Tests

- Use EMP framework + EMP toolbox
  - Bitfile with built-in test functions
- Controlled via EMP Python bindings
  - Integrated with Pytest

Test sequence after programming FPGA

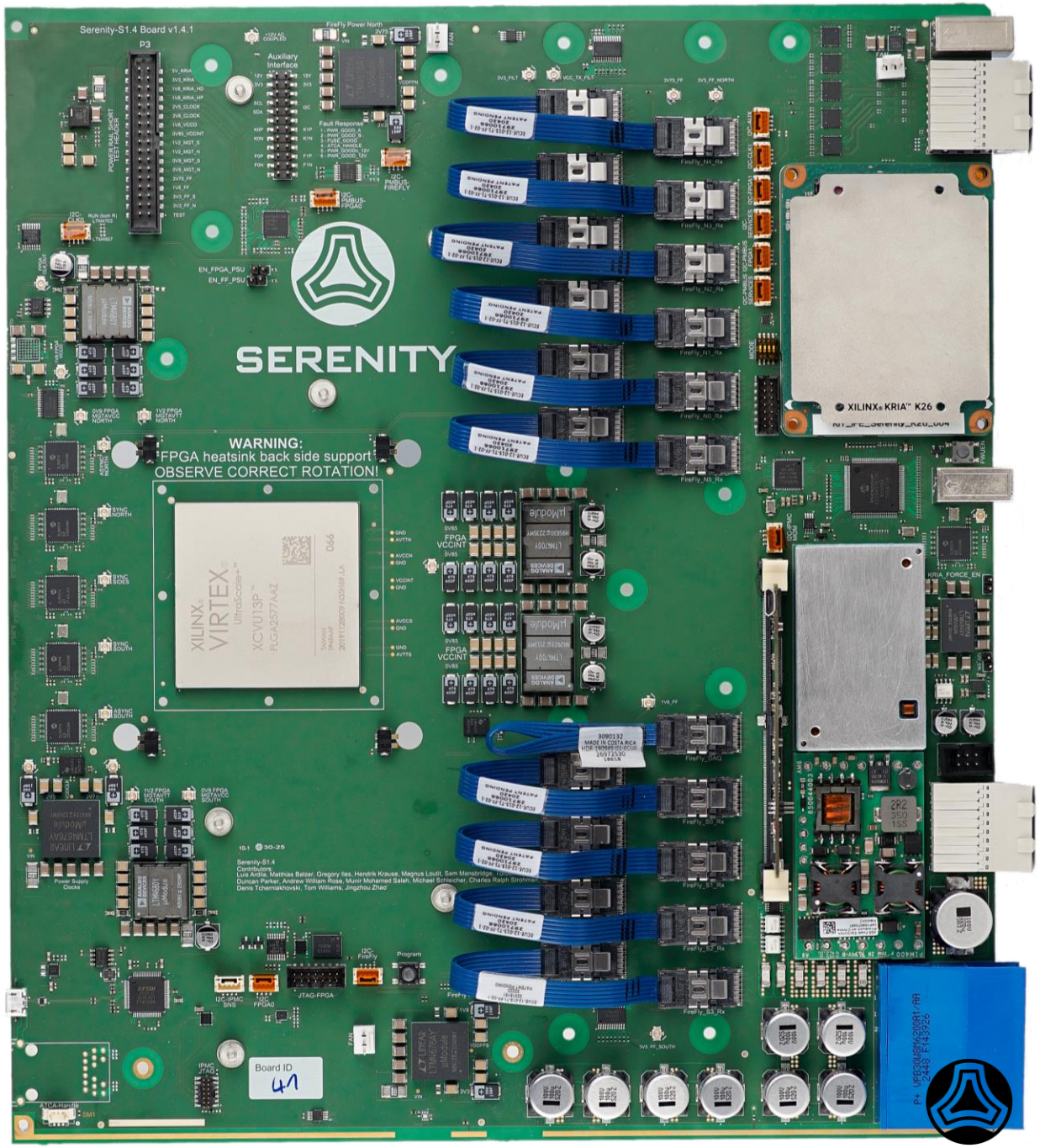
- Verify Kria FPGA Chip2Chip connection
- Measure MGT refclocks (async & sync)
- SerDes link test



JINST	[Paper] T. Williams et al.: EMP: a common infrastructure firmware framework for the CMS phase-2 upgrades

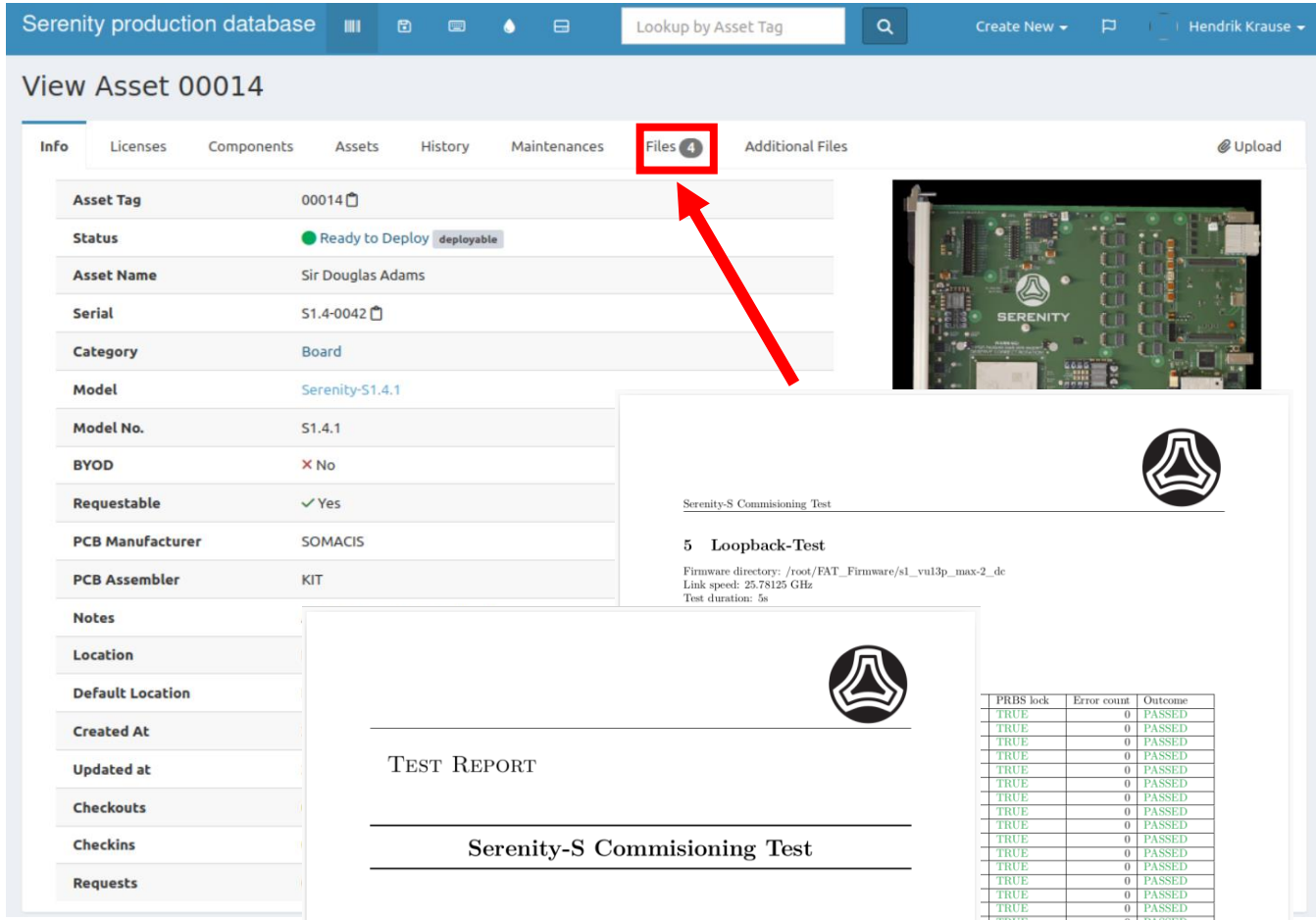
FAT Procedure – Step 5: SerDes Tests

- Based on existing EMP link test functionality (PRBS31)
- Copper loopback cables instead of optical FireFly modules
  - Faster setup & avoids risk of damaging optics
- Operated without heatsink → thermal precautions
  - Enable only 16 channels at a time
  - Run test for 5 seconds per link



FAT Procedure – Step 6: Report Creation & Database Upload

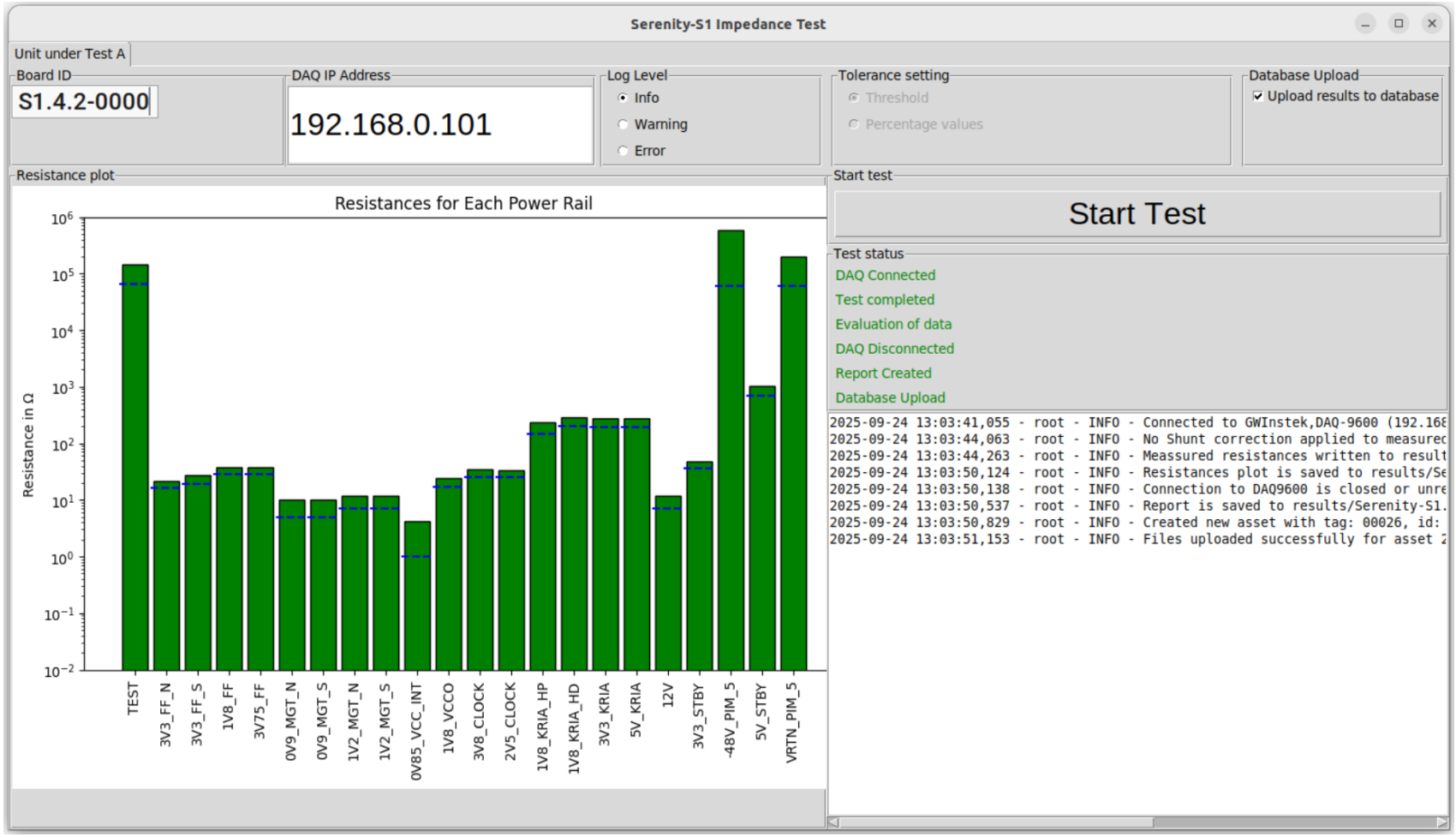
- Raw outputs: logfiles + pytest reports
- Automated report generation
  - pylatex + TeX Live
- Two reports per board:
  - Short-circuit test report
  - Functional test report
- Logs archived (zip) for full traceability
- Upload into the Serenity production database
  - Easy access & board history tracking



How to Control the FAT?

Python based GUI

- Simple interface
- Safe to operate
- Automated test flow
- Feedback on test status



How to Control the FAT?

Python based GUI

- Simple interface
- Safe to operate
- Automated test flow
- Feedback on test status

Conclusions & Lessons Learned

Conclusion (What we achieved):

- Extended QA beyond structural testing
  - Functional tests directly at the manufacturing companies
- Built on on-board controllers and existing board management tools
- Proven usage during extended pilot and pre-series production
  - FAT usage at CERN, KIT, TIFR and three Assembly companies
  - Successfully tested 60 Serenity-S1 boards

Lessons learned:

- Adding debugging features to a board is very useful
- Having good and flexible board management tools pays off
- Build-In self test is a good approach for functional testing
- Pytest is suitable for testing hardware
- If you are dealing with non experts having a GUI is essential
- Having a clear test specification in the beginning helps a lot to simplify the development

SERENITY