Factory Acceptance Test for the Serenity-S1

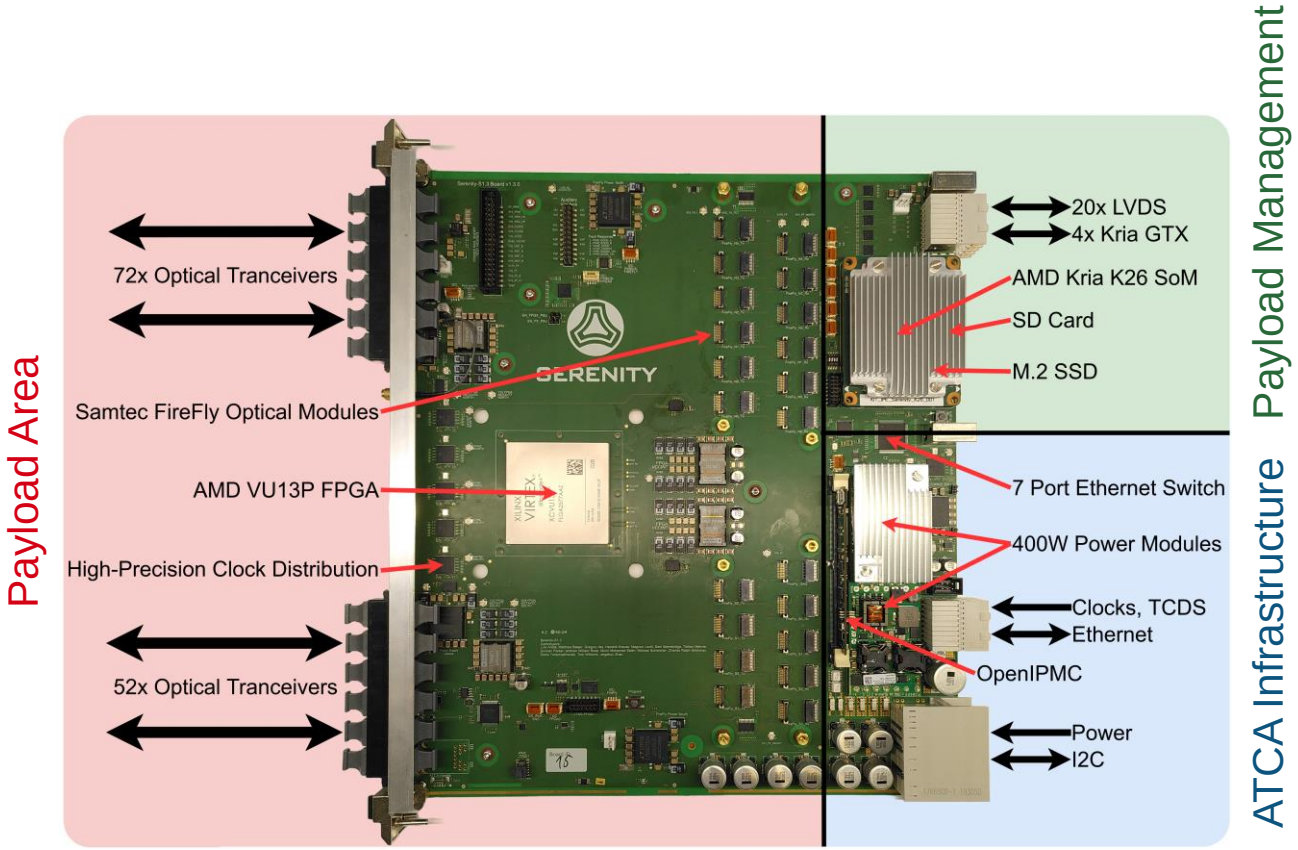
Hendrik Krause on behalf of the Serenity consortium

SEI-Tagung 2025

Motivation

The Serenity-S1 board (see Torben's talk) is very complex

- 2473 components in total
- 106 are (complex) integrated circuits
  - Some require initial configuration (power supplies,…)



Motivation

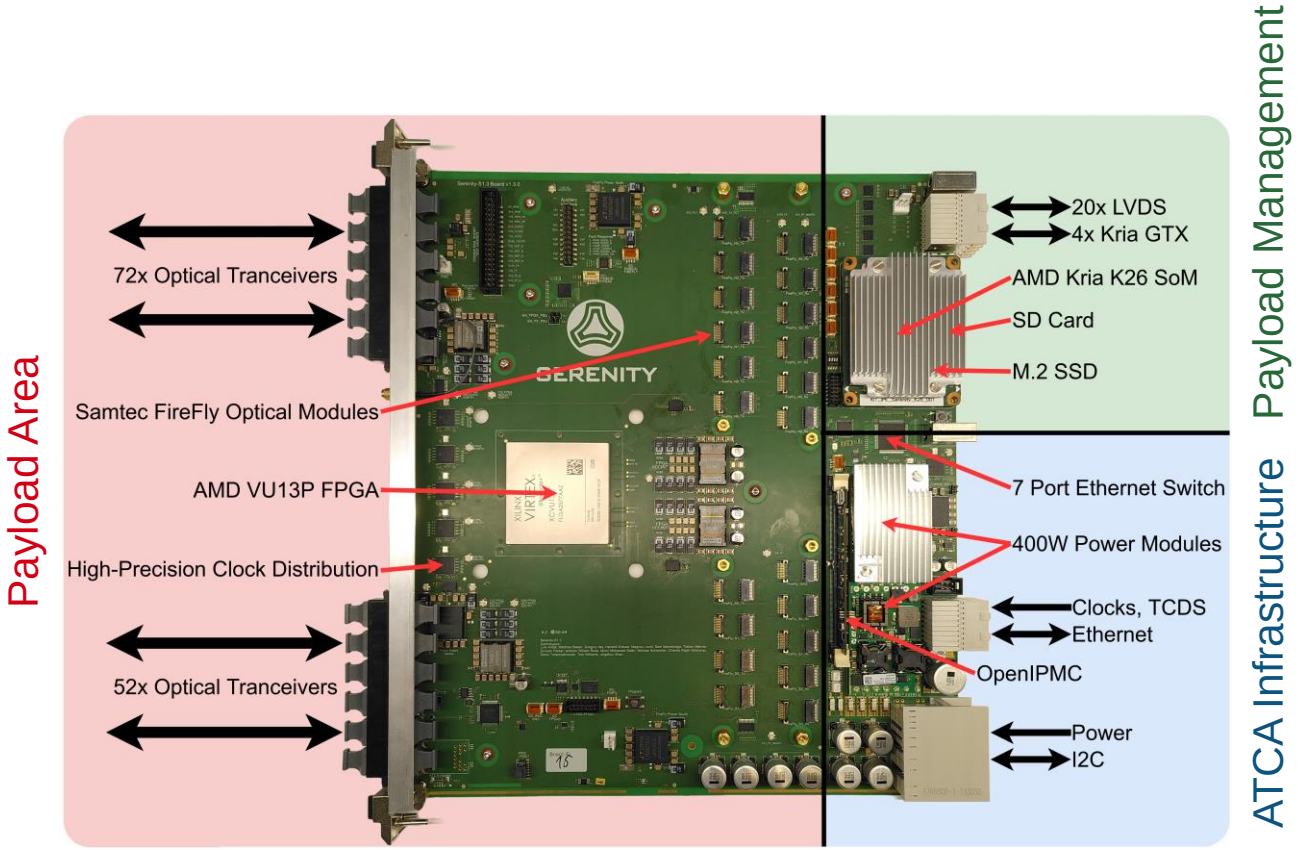
The Serenity-S1 board (see Torben's talk) is very complex

- 2473 components in total
- 106 are (complex) integrated circuits
  - Some require initial configuration (power supplies,…)

Initial commissioning has taken us weeks and requires detailed expert knowledge

- Manual commissioning revealed issues in many pilot- and extended pilot-production board
- Manual commissioning not feasible for a planned production of **725** boards

We need a reliable way to test the boards and automate the commissioning process



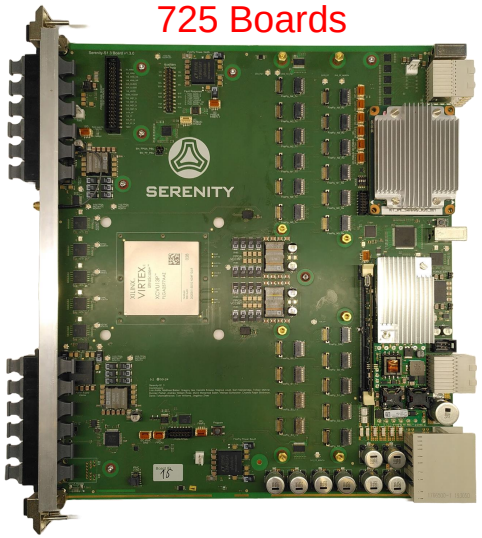
Scope of the Test

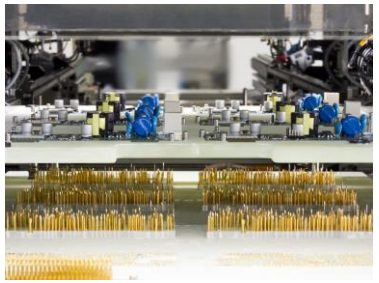
Single-Item Production	Small Batch Production	Large Batch Production	Mass Production
1-10 Boards	10-100 Boards	100-10,000 Boards	10,000+ Boards

Manual commissioning

- Multimeter
- Scope
- …

725 Boards





Highly automated and specialized testing

- In-Circuit Test (bed of nails)
- Flying-Probe Test
- …

Scope of the Test

Single-Item Production

1-10 Boards

Manual commissioning

- Multimeter
- Scope
- …

Small Batch

Large Batch

Factory Acceptance Test (FAT)

- Addition to the optical inspection done by the manufacturer
- End-of-line test directly at the factory (allow fast repair cycles)
- Mostly automated with a time scope of 10 minutes
- Can be done by non experts
- Focus on the active components
- Reuse existing tools to minimize development effort
- ➢ **Goal: Every board that arrives at CERN works!**

Mass Production

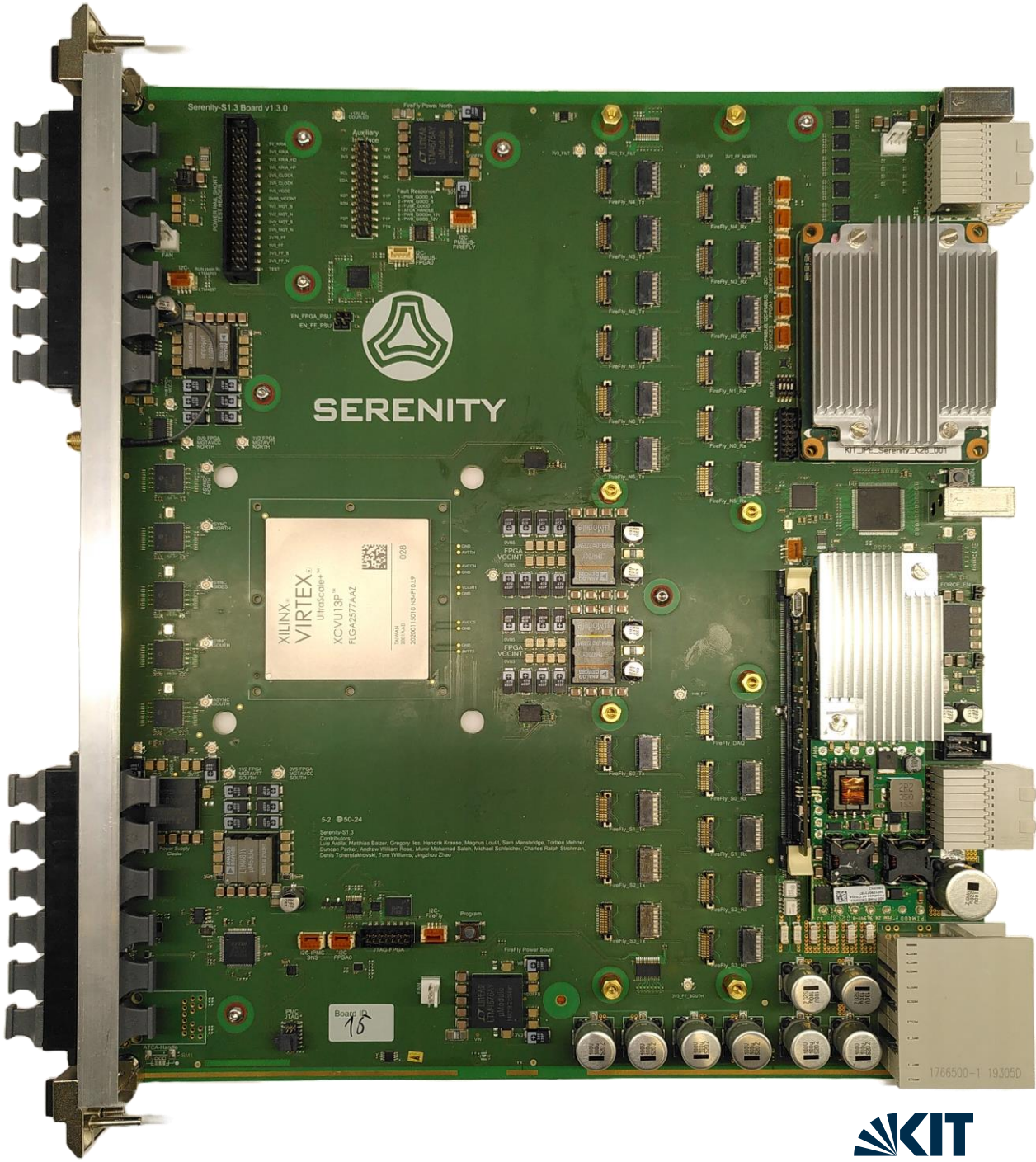
10,000+ Boards

Highly automated and specialized testing

- In-Circuit Test (bed of nails)
- Flying-Probe Test
- …

What does the Board already provide for testing?

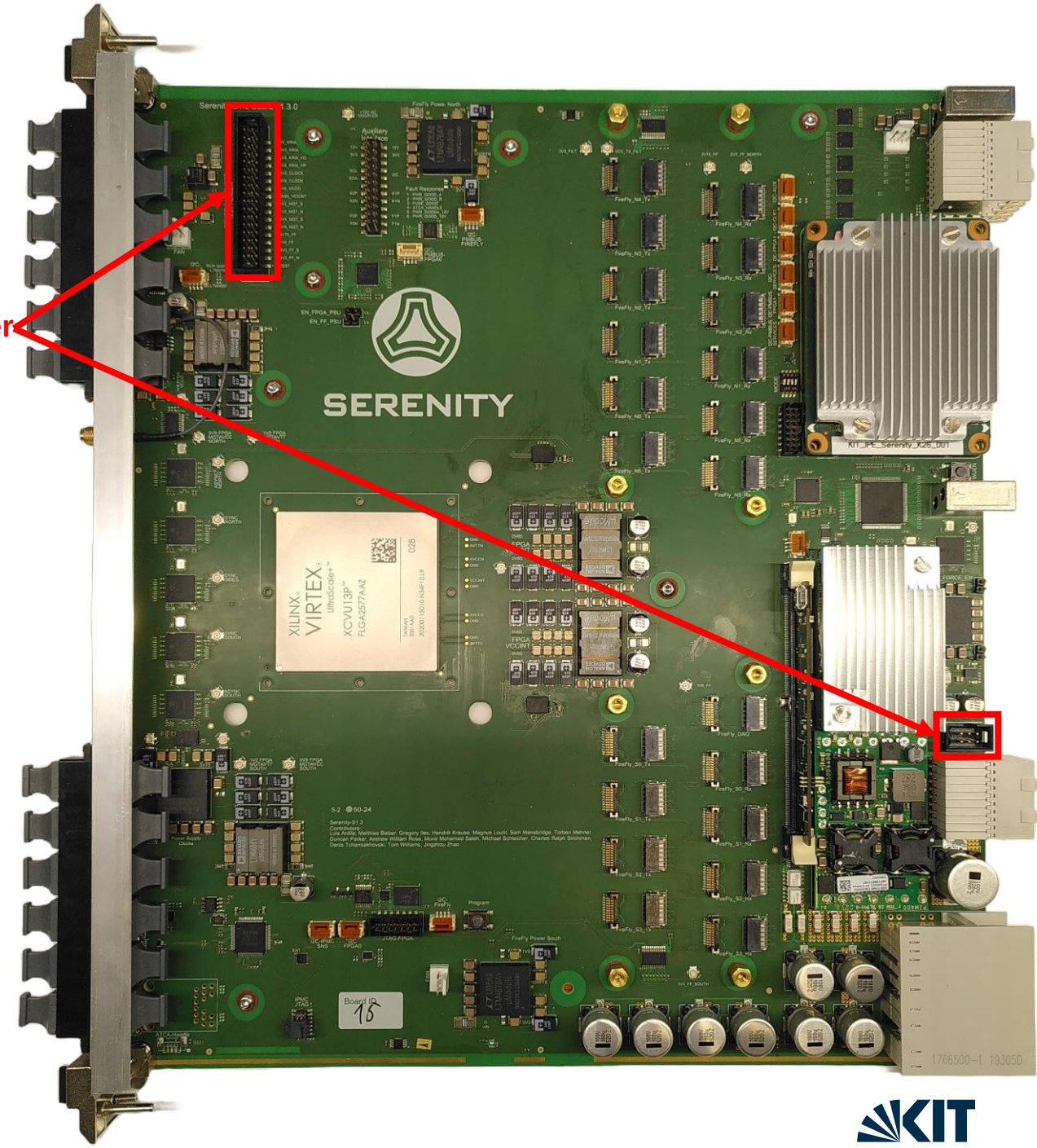
Idea: Reuse existing resources on the board



What does the Board already provide for testing?

Idea: Reuse existing resources on the board

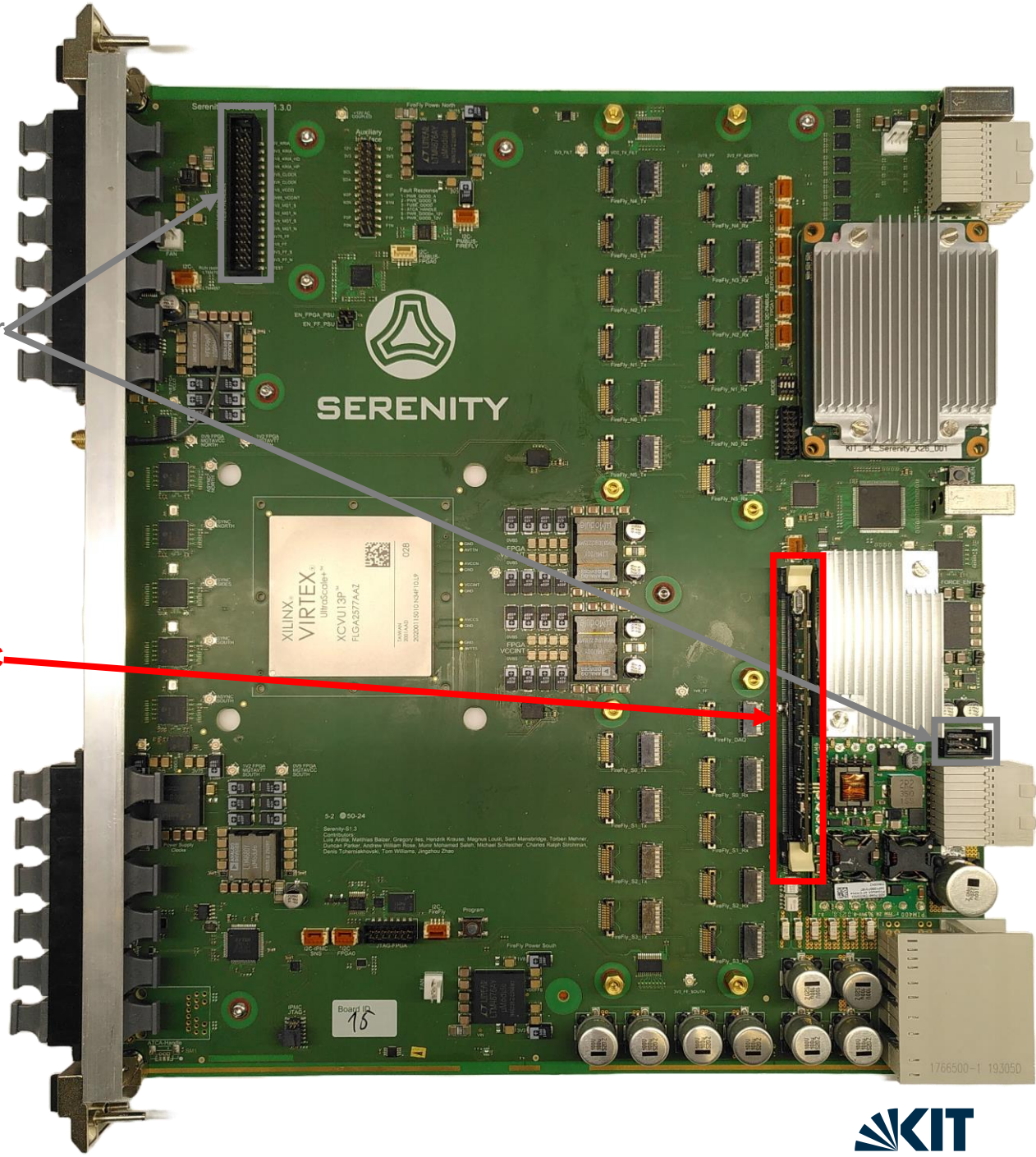
- Two Test Headers
  - Provide access to the power rails



What does the Board already provide for testing?

Idea: Reuse existing resources on the board

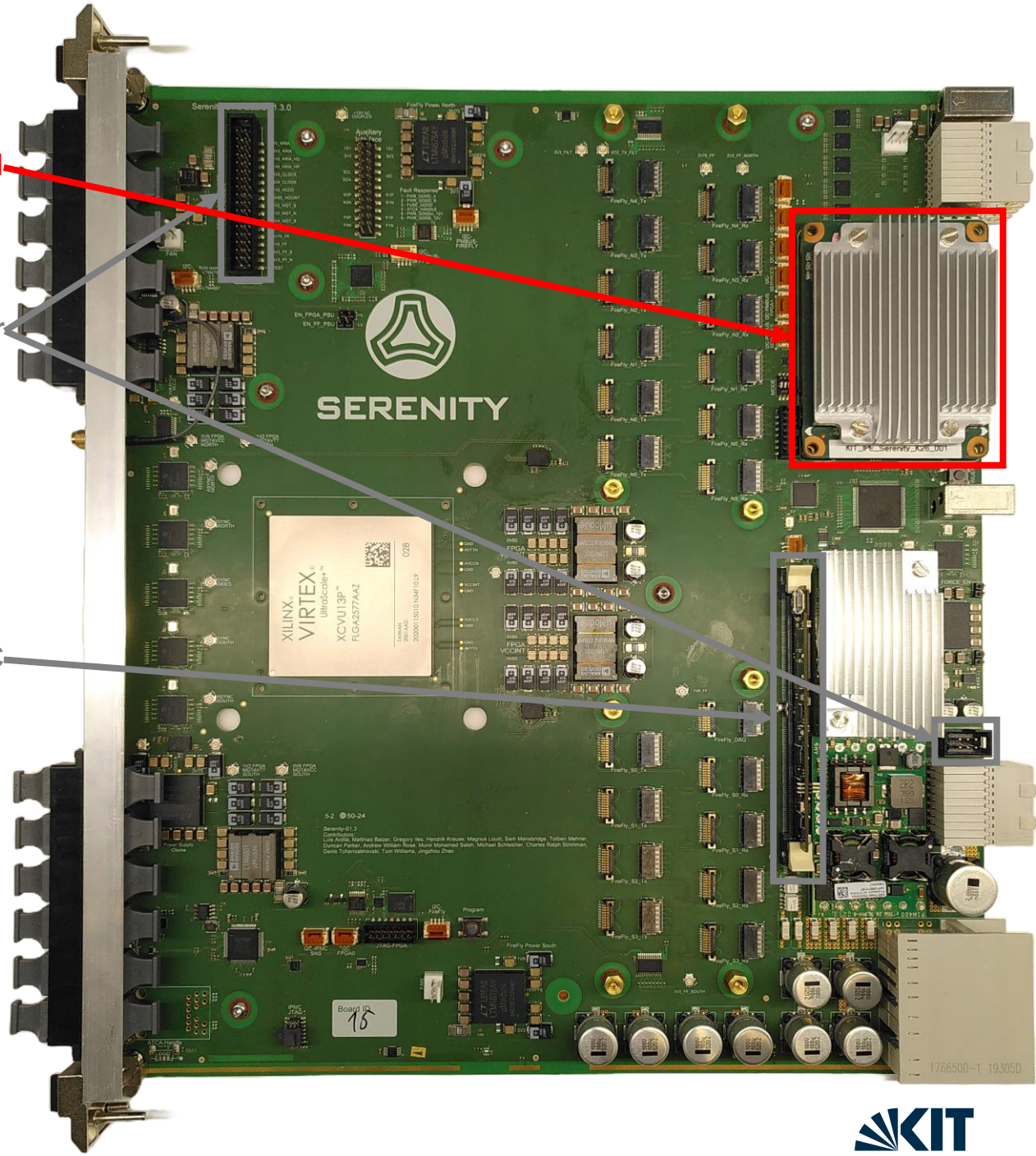
- Two Test Headers
  - Provide access to the power rails
- OpenIPMC
  - STM32 µController
  - Accessible via telnet



What does the Board already provide for testing?

Idea: Reuse existing resources on the board

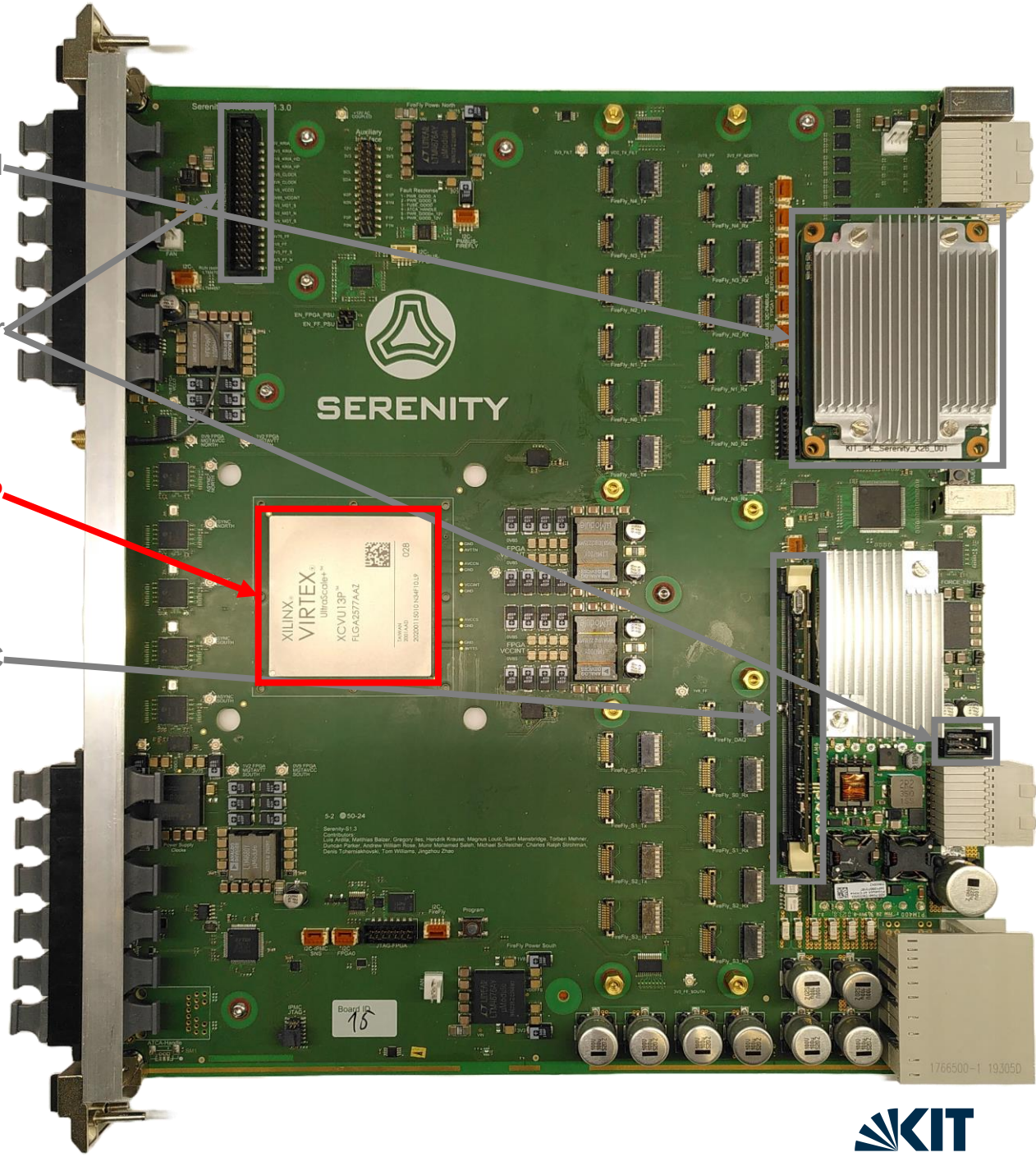
- Two Test Headers
  - Provide access to the power rails
- OpenIPMC
  - STM32 µController
  - Accessible via telnet
- Kria K26 SoM
  - Processing System with Linux
  - Advanced board management tools (SMASH & EMP)
  - Accessible via SSH



What does the Board already provide for testing?

Idea: Reuse existing resources on the board

- Two Test Headers
  - Provide access to the power rails
- OpenIPMC
  - STM32 µController
  - Accessible via telnet
- Kria K26 SoM
  - Processing System with Linux
  - Advanced board management tools (SMASH & EMP)
  - Accessible via SSH
- FPGA
  - Accessible via AXI C2C



How to structure the FAT?

Idea: Reuse existing resources on the board

- Two Test Headers
  - Provide access to the power rails
- OpenIPMC
  - STM32 µController
  - Accessible via telnet
- Kria K26 SoM
  - Processing System with Linux
  - Advanced board management tools (SMASH & EMP)
  - Accessible via SSH
- FPGA
  - Accessible via AXI C2C

Test Flow

How to structure the FAT?

Idea: Reuse existing resources on the board

- Two Test Headers
  - Provide access to the power rails
- OpenIPMC
  - STM32 µController
  - Accessible via telnet
- Kria K26 SoM
  - Processing System with Linux
  - Advanced board management tools (SMASH & EMP)
  - Accessible via SSH
- FPGA
  - Accessible via AXI C2C

Test Flow

1. Short-Circuit & Impedance Test
   - Measure the impedance of every power rail

How to structure the FAT?

Idea: Reuse existing resources on the board

- Two Test Headers
  - Provide access to the power rails
- OpenIPMC
  - STM32 µController
  - Accessible via telnet
- Kria K26 SoM
  - Processing System with Linux
  - Advanced board management tools (SMASH & EMP)
  - Accessible via SSH
- FPGA
  - Accessible via AXI C2C

Test Flow

1. Short-Circuit & Impedance Test
   - Measure the impedance of every power rail
2. OpenIPMC Test
   - Standby Power Test
   - Commission the service area

How to structure the FAT?

Idea: Reuse existing resources on the board

- Two Test Headers
  - Provide access to the power rails
- OpenIPMC
  - STM32 µController
  - Accessible via telnet
- Kria K26 SoM
  - Processing System with Linux
  - Advanced board management tools (SMASH & EMP)
  - Accessible via SSH
- FPGA
  - Accessible via AXI C2C

Test Flow

1. Short-Circuit & Impedance Test
   - Measure the impedance of every power rail
2. OpenIPMC Test
   - Standby Power Test
   - Commission the service area
3. Kria Test without FPGA
   - Using SMASH and the Serenity Toolbox to commission the payload area

How to structure the FAT?

Idea: Reuse existing resources on the board

- Two Test Headers
  - Provide access to the power rails
- OpenIPMC
  - STM32 µController
  - Accessible via telnet
- Kria K26 SoM
  - Processing System with Linux
  - Advanced board management tools (SMASH & EMP)
  - Accessible via SSH
- FPGA
  - Accessible via AXI C2C

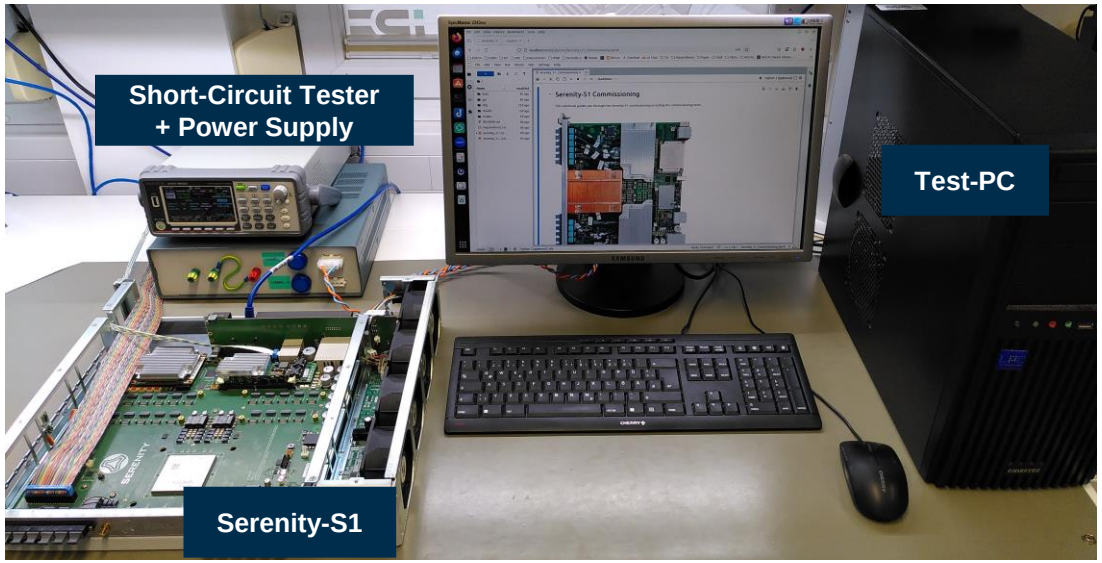
Test Flow

1. Short-Circuit & Impedance Test
   - Measure the impedance of every power rail
2. OpenIPMC Test
   - Standby Power Test
   - Commission the service area
3. Kria Test without FPGA
   - Using SMASH and the Serenity Toolbox to commission the payload area
4. Kria Tests with FPGA
   - Using a test firmware based on EMP to further extend the commissioning of the payload area

How to control the FAT?

Jupiter(lab) Notebooks combines code with documentation

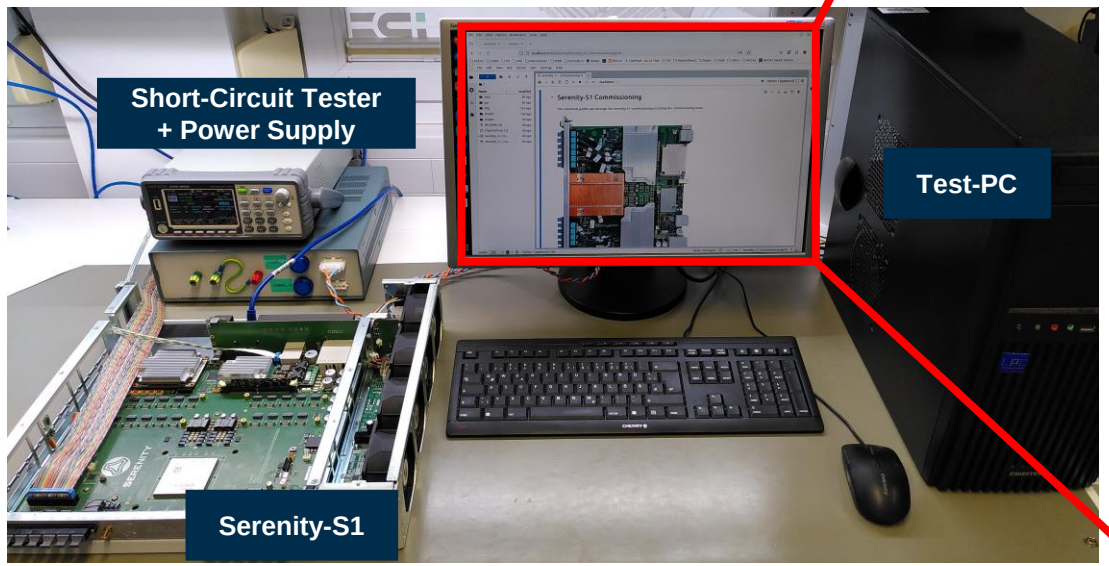
- Markdown to provide instructions to the user
- Python to control the test flow
  - Access the Serenity-S1 (OpenIPMC & Kria)
  - Access the power supply and short-circuit tester
  - Show test results

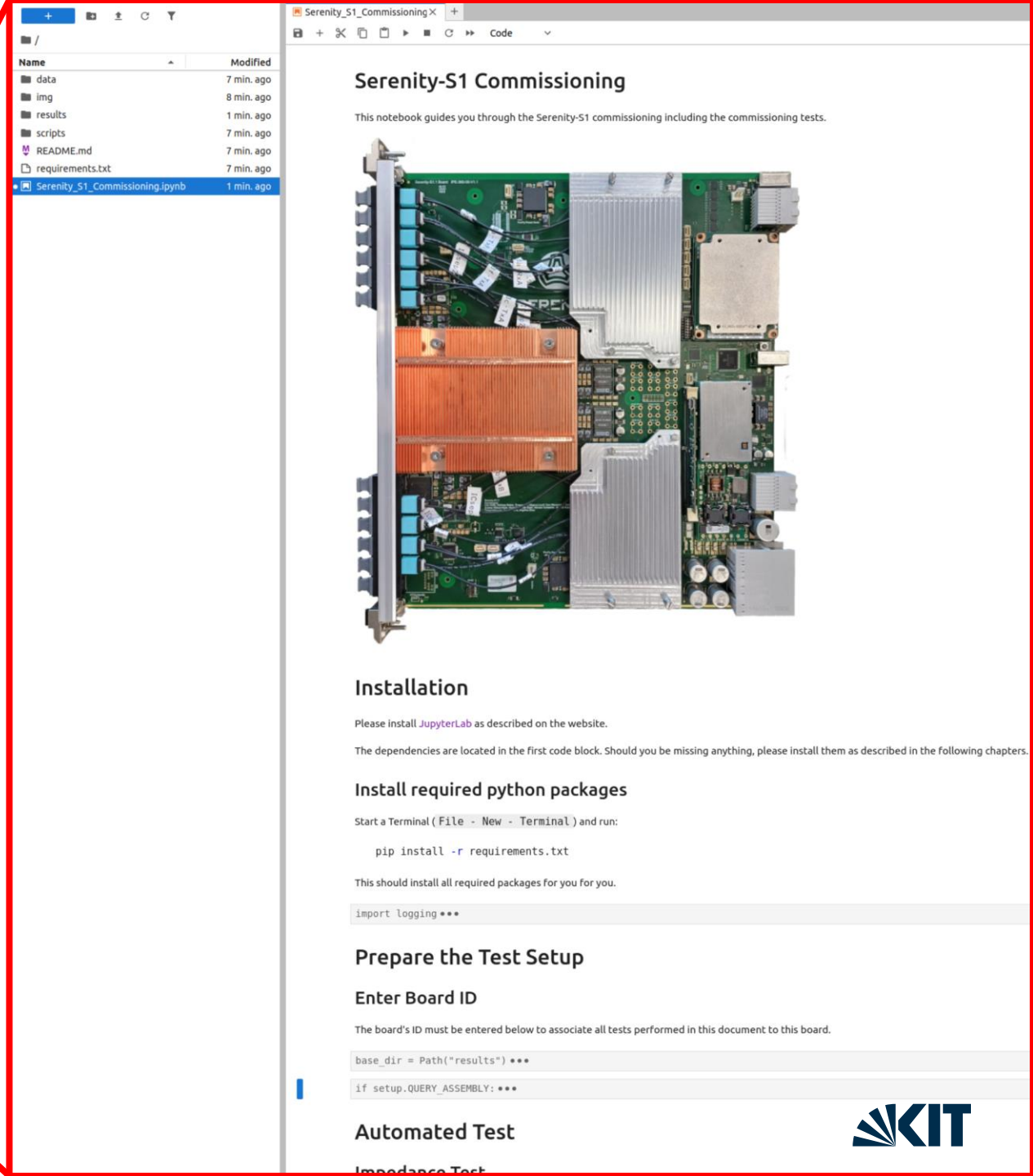


How to control the FAT?

Jupiter(lab) Notebooks combines code with documentation

- Markdown to provide instructions to the user
- Python to control the test flow
  - Access the Serenity-S1 (OpenIPMC & Kria)
  - Access the power supply and short-circuit tester
  - Show test results

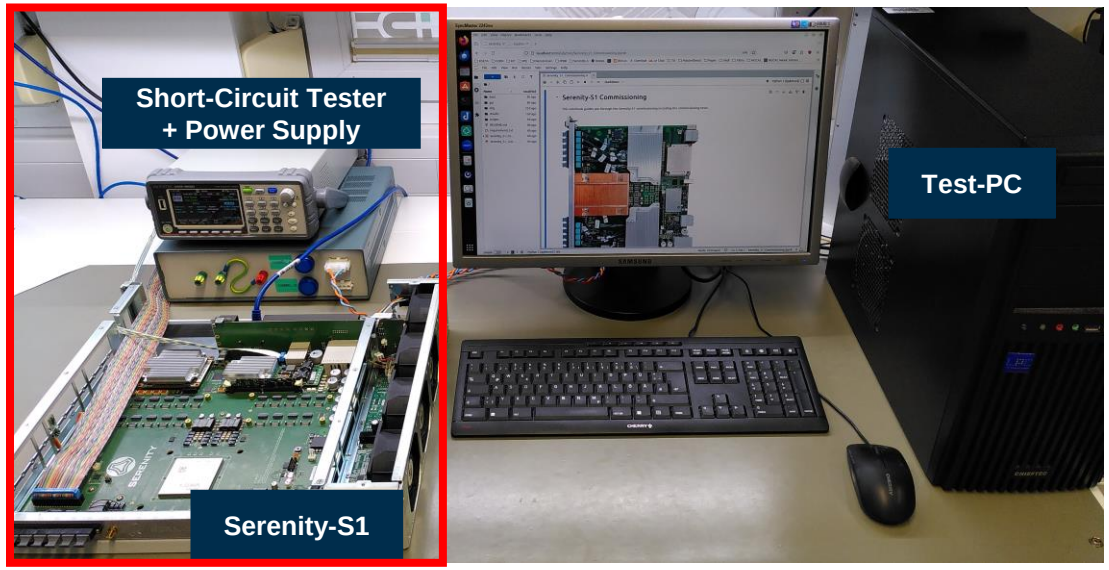




1. Short-Circuit & Impedance Test

Using of the Shelf Test Equipment

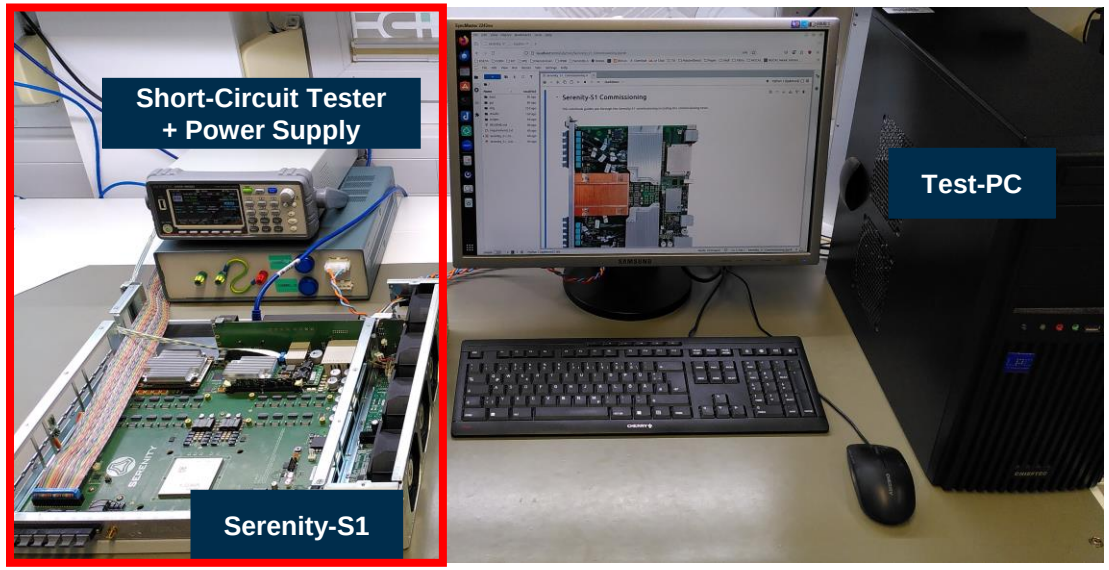
- Impedance tester based on constant current source
- Measure the impedance of every power rail
- Comparison against nominal values

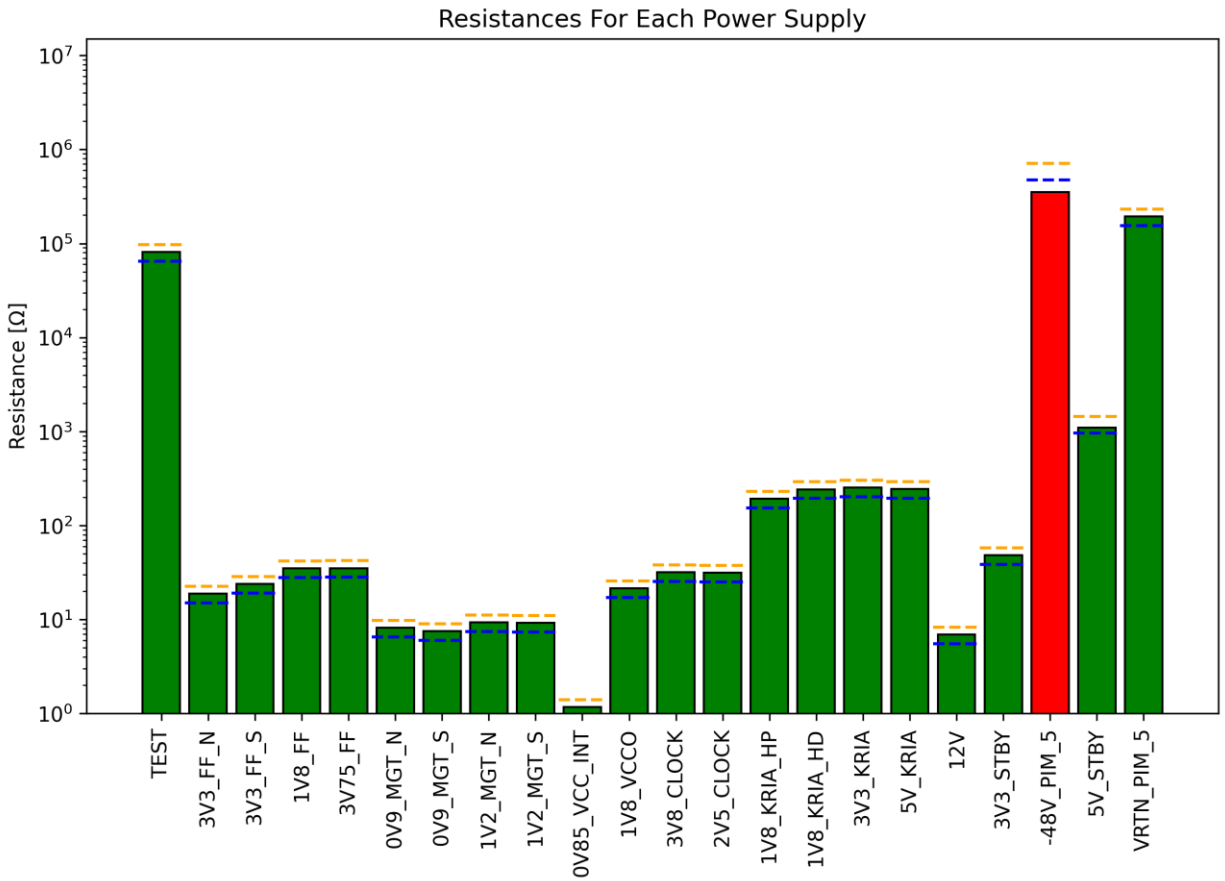


1. Short-Circuit & Impedance Test

Using of the Shelf Test Equipment

- Impedance tester based on constant current source
- Measure the impedance of every power rail
- Comparison against nominal values



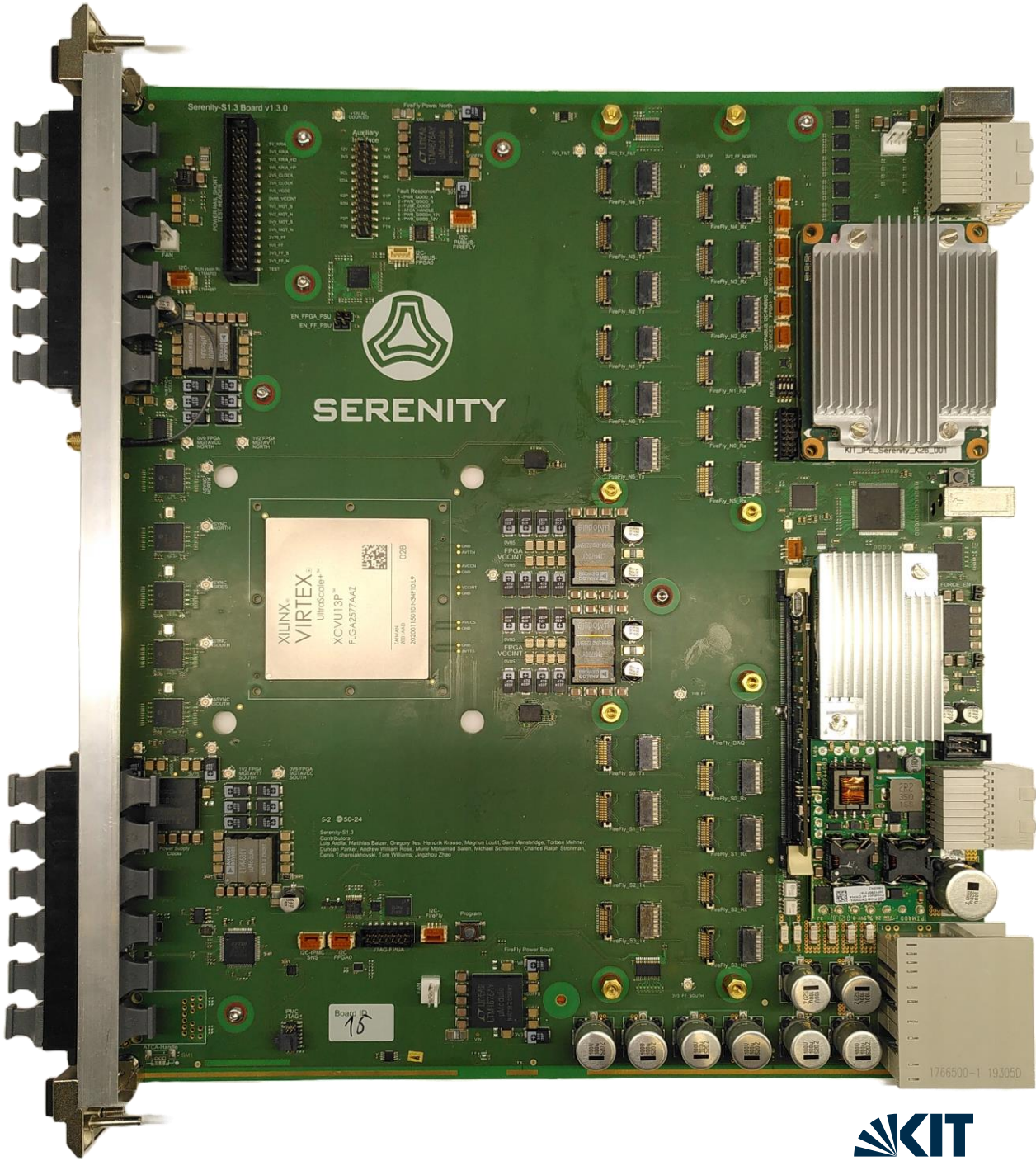


2. OpenIPMC Tests

OpenIPMC is an open-source implementation of the IPMC

- IPMC already includes power and temperature monitoring per ATCA standard

➢ This functionality can be reused for the standby power test

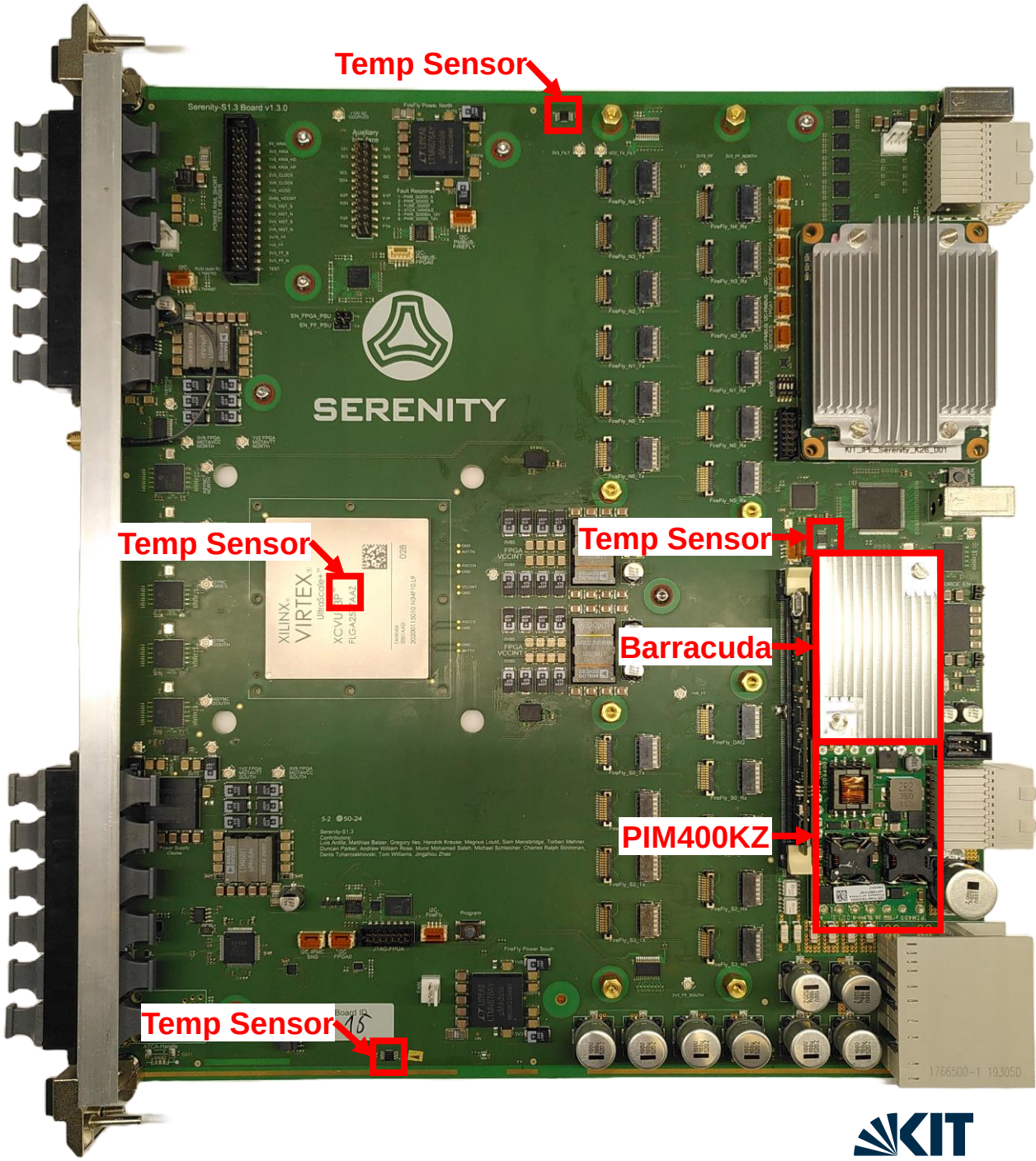


2. OpenIPMC Tests

OpenIPMC is an open-source implementation of the IPMC

- IPMC already includes power and temperature monitoring per ATCA standard

➢ This functionality can be reused for the standby power test



2. OpenIPMC Tests

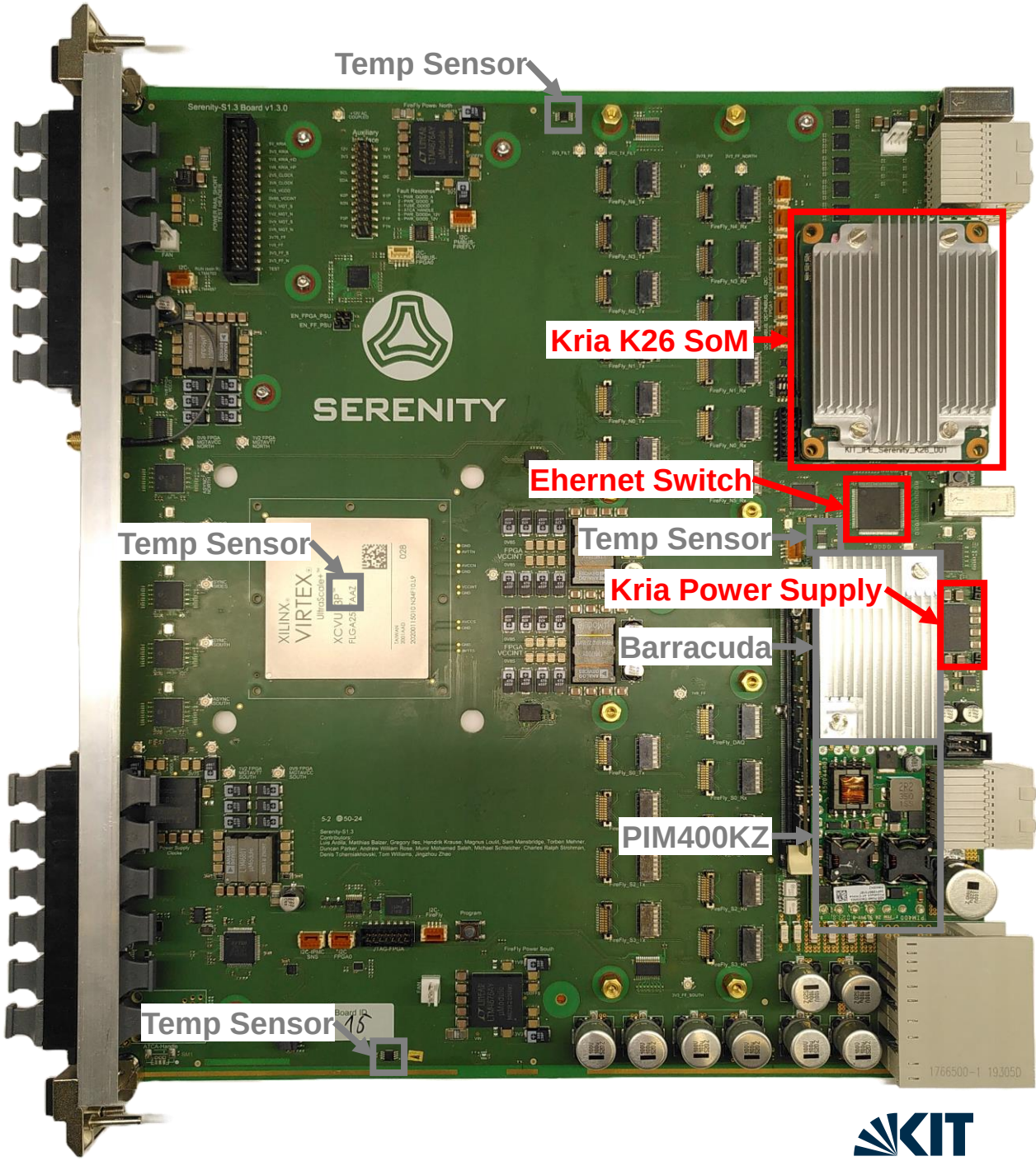
OpenIPMC is an open-source implementation of the IPMC

- IPMC already includes power and temperature monitoring per ATCA standard
- ➢ This functionality can be reused for the standby power test

Additional inputs are connected to various service area components

- Kria power supply
- Kria status signals
- UART to Kria
- Ethernet switch
- ➢ This functionality can be used to commission the service area components

Access via telnet



2. OpenIPMC Tests

OpenIPMC is an open-source implementation of the IPMC

- IPMC already includes power and temperature monitoring per ATCA standard
- ➢ This functionality can be reused for the standby power test

Additional inputs are connected to various service area components

- Kria power supply
- Kria status signals
- UART to Kria
- Ethernet switch
- ➢ This functionality can be used to commission the service area components

Access via telnet

```
[lab@oo-k03 ~]$ telnet 192.168.0.40
Trying 192.168.0.40...
Connected to 192.168.0.40.
Escape character is '^]'.

>> sensors
TEMP_FPGA_INT_[C]: 27.000000
TEMP_NORTH_[C]: 24.000000
TEMP_SOUTH_[C]: 24.000000
TEMP_SERVICE_[C]: 29.000000
TEMP_FPGA_EXT_[C]: 29.000000
PIM400_current_[A]: 1.316000
PIM400_voltage_A_[V]: 48.099998
PIM400_voltage_B_[V]: 48.099998
PIM400_temp_[C]: 38.244999
Barracuda_temp_[C]: 0.000000
Barracuda_voltage_in_[V]: Reading sensor failed!
Barracuda_voltage_out_[V]: 0.000000
Barracuda_current_out_[A]: Reading sensor failed!
PG_5V_KRIA: 1
PG_3V3_KRIA: 1
PG_1V8_KRIA_HD: 1
PG_1V8_KRIA_HP: 1
PG_KRIA_FPD: 1
PG_KRIA_LPD: 1
PG_KRIA_PL: 1
KRIA_ERROR_OUT: 0
PS_ERROR_STATUS: 0

>>
```

2. OpenIPMC Tests

OpenIPMC is an open-source implementation of the IPMC

- IPMC already includes power and temperature monitoring per ATCA standard
- ➢ This functionality can be reused for the standby power test

Additional inputs are connected to various service area components

- Kria power supply
- Kria status signals
- UART to Kria
- Ethernet switch
- ➢ This functionality can be used to commission the service area components

Access via telnet

```
[lab@oo-k03 ~]$ telnet 192.168.0.40
Trying 192.168.0.40...
Connected to 192.168.0.40.
Escape character is '^]'.

>> sensors
TEMP_FPGA_INT_[C]: 27.000000
TEMP_NORTH_[C]: 24.000000
TEMP_SOUTH_[C]: 24.000000
TEMP_SERVICE_[C]: 29.000000
TEMP_FPGA_EXT_[C]: 29.000000
PIM400_current_[A]: 1.316000
PIM400_voltage_A_[V]: 48.099998
PIM400_voltage_B_[V]: 48.099998
PIM400_temp_[C]: 38.244999
Barracuda_temp_[C]: 0.000000
Barracuda_voltage_in_[V]: Reading sensor failed!
Barracuda_voltage_out_[V]: 0.000000
Barracuda_current_out_[A]: Reading sensor failed!
PG_5V_KRIA: 1
PG_3V3_KRIA: 1
PG_1V8_KRIA_HD: 1
PG_1V8_KRIA_HP: 1
PG_KRIA_FPD: 1
PG_KRIA_LPD: 1
PG_KRIA_PL: 1
KRIA_ERROR_OUT: 0
PS_ERROR_STATUS: 0

>>
```

Temperature Sensors

2. OpenIPMC Tests

OpenIPMC is an open-source implementation of the IPMC

- IPMC already includes power and temperature monitoring per ATCA standard

➢ This functionality can be reused for the standby power test

Additional inputs are connected to various service area components

- Kria power supply
- Kria status signals
- UART to Kria
- Ethernet switch

➢ This functionality can be used to commission the service area components

Access via telnet

```
[lab@oo-k03 ~]$ telnet 192.168.0.40
Trying 192.168.0.40...
Connected to 192.168.0.40.
Escape character is '^]'.

>> sensors
TEMP_FPGA_INT_[C]: 27.000000
TEMP_NORTH_[C]: 24.000000
TEMP_SOUTH_[C]: 24.000000
TEMP_SERVICE_[C]: 29.000000
TEMP_FPGA_EXT_[C]: 29.000000
PIM400_current_[A]: 1.316000
PIM400_voltage_A_[V]: 48.099998
PIM400_voltage_B_[V]: 48.099998
PIM400_temp_[C]: 38.244999
Barracuda_temp_[C]: 0.000000
Barracuda_voltage_in_[V]: Reading sensor failed!
Barracuda_voltage_out_[V]: 0.000000
Barracuda_current_out_[A]: Reading sensor failed!
PG_5V_KRIA: 1
PG_3V3_KRIA: 1
PG_1V8_KRIA_HD: 1
PG_1V8_KRIA_HP: 1
PG_KRIA_FPD: 1
PG_KRIA_LPD: 1
PG_KRIA_PL: 1
KRIA_ERROR_OUT: 0
PS_ERROR_STATUS: 0

>>
```

Temperature Sensors

PIM400KZ

2. OpenIPMC Tests

OpenIPMC is an open-source implementation of the IPMC

- IPMC already includes power and temperature monitoring per ATCA standard

➢ This functionality can be reused for the standby power test

Additional inputs are connected to various service area components

- Kria power supply
- Kria status signals
- UART to Kria
- Ethernet switch

➢ This functionality can be used to commission the service area components

Access via telnet

```
[lab@oo-k03 ~]$ telnet 192.168.0.40
Trying 192.168.0.40...
Connected to 192.168.0.40.
Escape character is '^]'.

>> sensors
TEMP_FPGA_INT_[C]: 27.000000
TEMP_NORTH_[C]: 24.000000
TEMP_SOUTH_[C]: 24.000000
TEMP_SERVICE_[C]: 29.000000
TEMP_FPGA_EXT_[C]: 29.000000
PIM400_current_[A]: 1.316000
PIM400_voltage_A_[V]: 48.099998
PIM400_voltage_B_[V]: 48.099998
PIM400_temp_[C]: 38.244999
Barracuda_temp_[C]: 0.000000
Barracuda_voltage_in_[V]: Reading sensor failed!
Barracuda_voltage_out_[V]: 0.000000
Barracuda_current_out_[A]: Reading sensor failed!
PG_5V_KRIA: 1
PG_3V3_KRIA: 1
PG_1V8_KRIA_HD: 1
PG_1V8_KRIA_HP: 1
PG_KRIA_FPD: 1
PG_KRIA_LPD: 1
PG_KRIA_PL: 1
KRIA_ERROR_OUT: 0
PS_ERROR_STATUS: 0

>>
```

Temperature Sensors

PIM400KZ

Barracuda

2. OpenIPMC Tests

OpenIPMC is an open-source implementation of the IPMC

- IPMC already includes power and temperature monitoring per ATCA standard

➢ This functionality can be reused for the standby power test

Additional inputs are connected to various service area components

- Kria power supply
- Kria status signals
- UART to Kria
- Ethernet switch

➢ This functionality can be used to commission the service area components

Access via telnet

```
[lab@oo-k03 ~]$ telnet 192.168.0.40
Trying 192.168.0.40...
Connected to 192.168.0.40.
Escape character is '^]'.

>> sensors
TEMP_FPGA_INT_[C]: 27.000000
TEMP_NORTH_[C]: 24.000000
TEMP_SOUTH_[C]: 24.000000
TEMP_SERVICE_[C]: 29.000000
TEMP_FPGA_EXT_[C]: 29.000000
PIM400_current_[A]: 1.316000
PIM400_voltage_A_[V]: 48.099998
PIM400_voltage_B_[V]: 48.099998
PIM400_temp_[C]: 38.244999
Barracuda_temp_[C]: 0.000000
Barracuda_voltage_in_[V]: Reading sensor failed!
Barracuda_voltage_out_[V]: 0.000000
Barracuda_current_out_[A]: Reading sensor failed!
PG_5V_KRIA: 1
PG_3V3_KRIA: 1
PG_1V8_KRIA_HD: 1
PG_1V8_KRIA_HP: 1
PG_KRIA_FPD: 1
PG_KRIA_LPD: 1
PG_KRIA_PL: 1
KRIA_ERROR_OUT: 0
PS_ERROR_STATUS: 0

>>
```

Temperature Sensors

PIM400KZ

Barracuda

Kria Power Supplies

2. OpenIPMC Tests

OpenIPMC is an open-source implementation of the IPMC

- IPMC already includes power and temperature monitoring per ATCA standard

➢ This functionality can be reused for the standby power test

Additional inputs are connected to various service area components

- Kria power supply
- Kria status signals
- UART to Kria
- Ethernet switch

➢ This functionality can be used to commission the service area components

Access via telnet

```
[lab@oo-k03 ~]$ telnet 192.168.0.40
Trying 192.168.0.40...
Connected to 192.168.0.40.
Escape character is '^]'.

>> sensors
TEMP_FPGA_INT_[C]: 27.000000
TEMP_NORTH_[C]: 24.000000
TEMP_SOUTH_[C]: 24.000000
TEMP_SERVICE_[C]: 29.000000
TEMP_FPGA_EXT_[C]: 29.000000
PIM400_current_[A]: 1.316000
PIM400_voltage_A_[V]: 48.099998
PIM400_voltage_B_[V]: 48.099998
PIM400_temp_[C]: 38.244999
Barracuda_temp_[C]: 0.000000
Barracuda_voltage_in_[V]: Reading sensor failed!
Barracuda_voltage_out_[V]: 0.000000
Barracuda_current_out_[A]: Reading sensor failed!
PG_5V_KRIA: 1
PG_3V3_KRIA: 1
PG_1V8_KRIA_HD: 1
PG_1V8_KRIA_HP: 1
PG_KRIA_FPD: 1
PG_KRIA_LPD: 1
PG_KRIA_PL: 1
KRIA_ERROR_OUT: 0
PS_ERROR_STATUS: 0

>>
```

Temperature Sensors

PIM400KZ

Barracuda

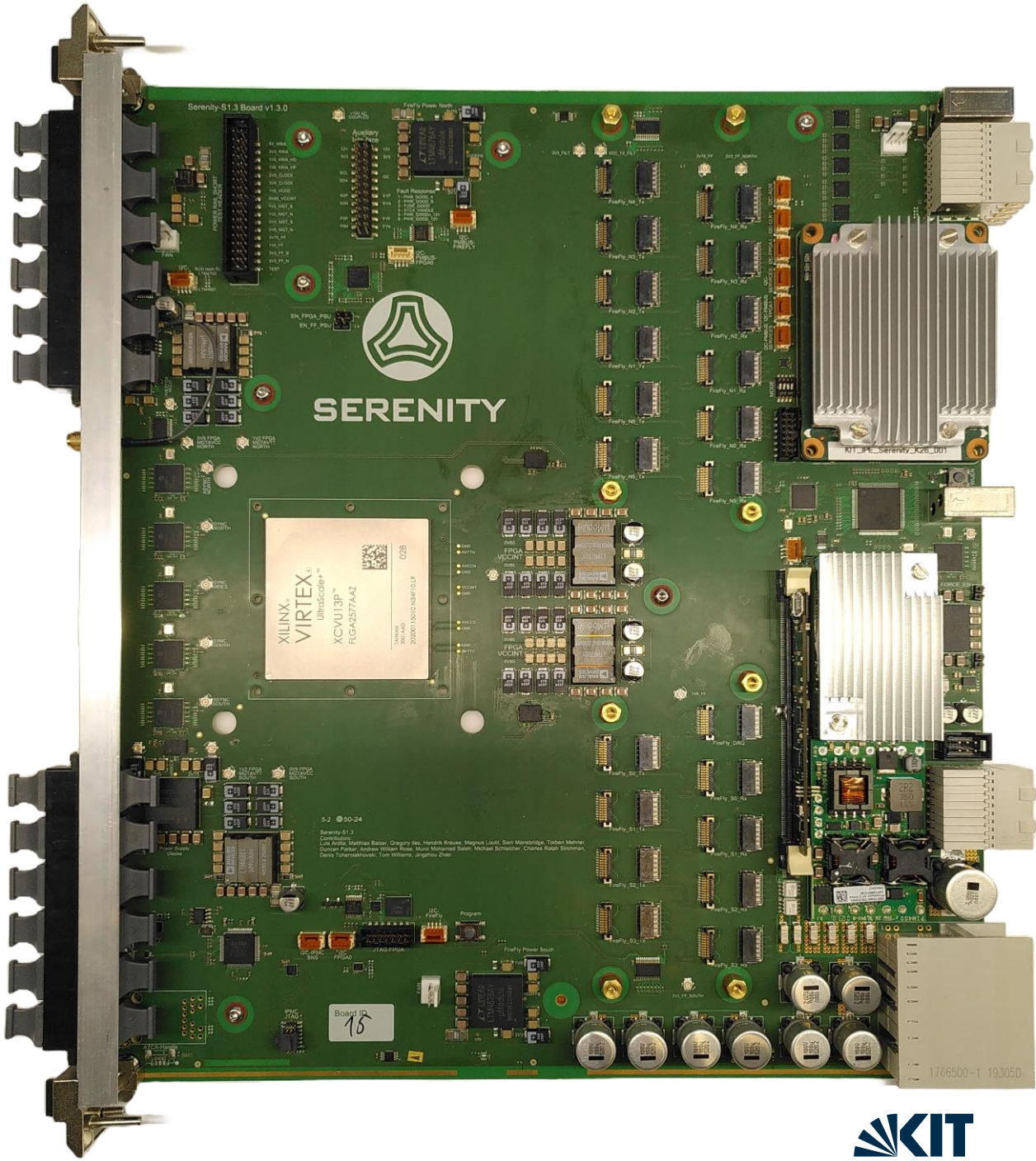
Kria Power Supplies

Kria Status Signals

3. Kria Test without the FPGA

Advanced board management: SMASH & Serenity Toolbox

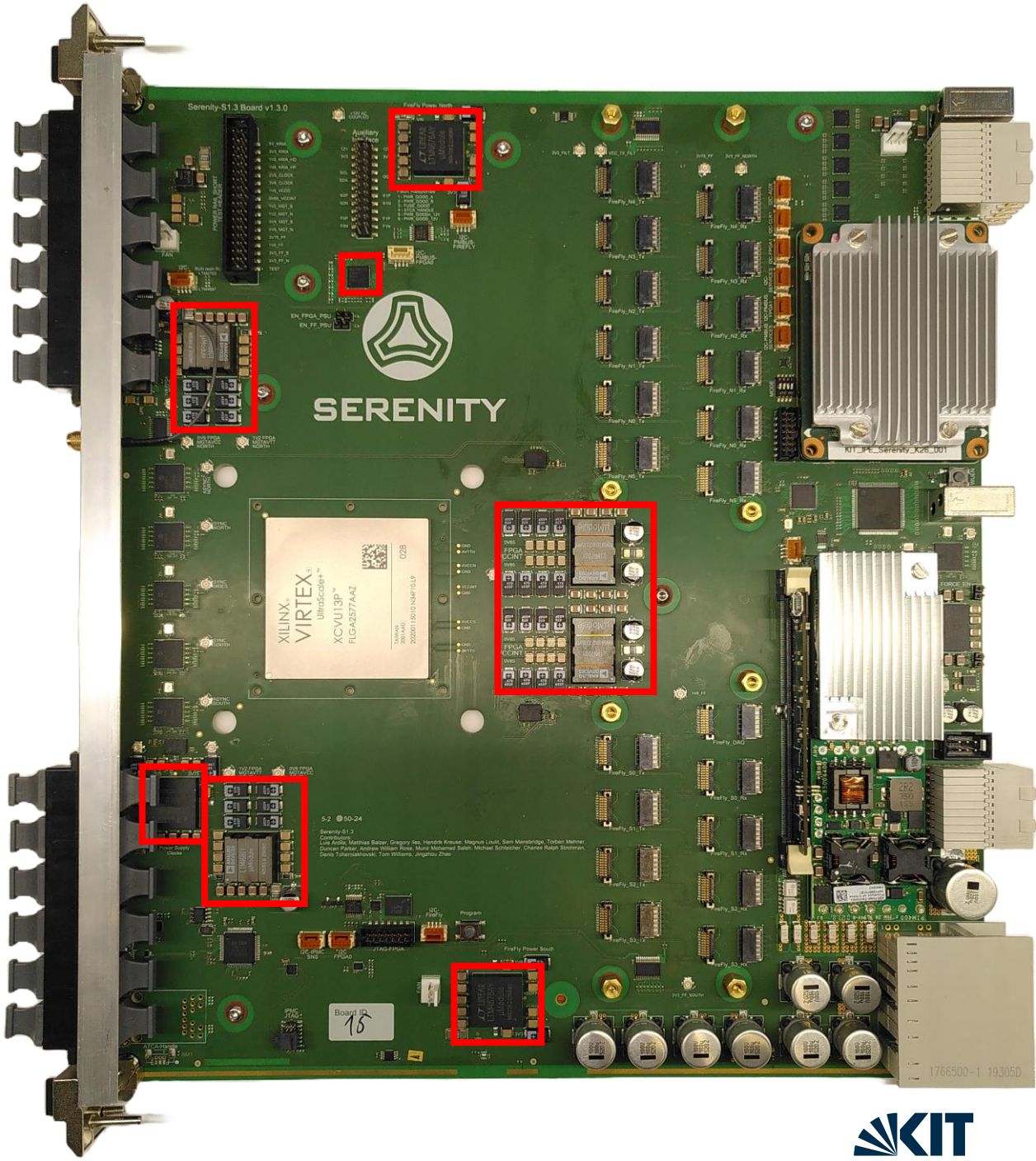
- Provide access to all I2C and PMBus devices + JTAG



3. Kria Test without the FPGA

Advanced board management: SMASH & Serenity Toolbox

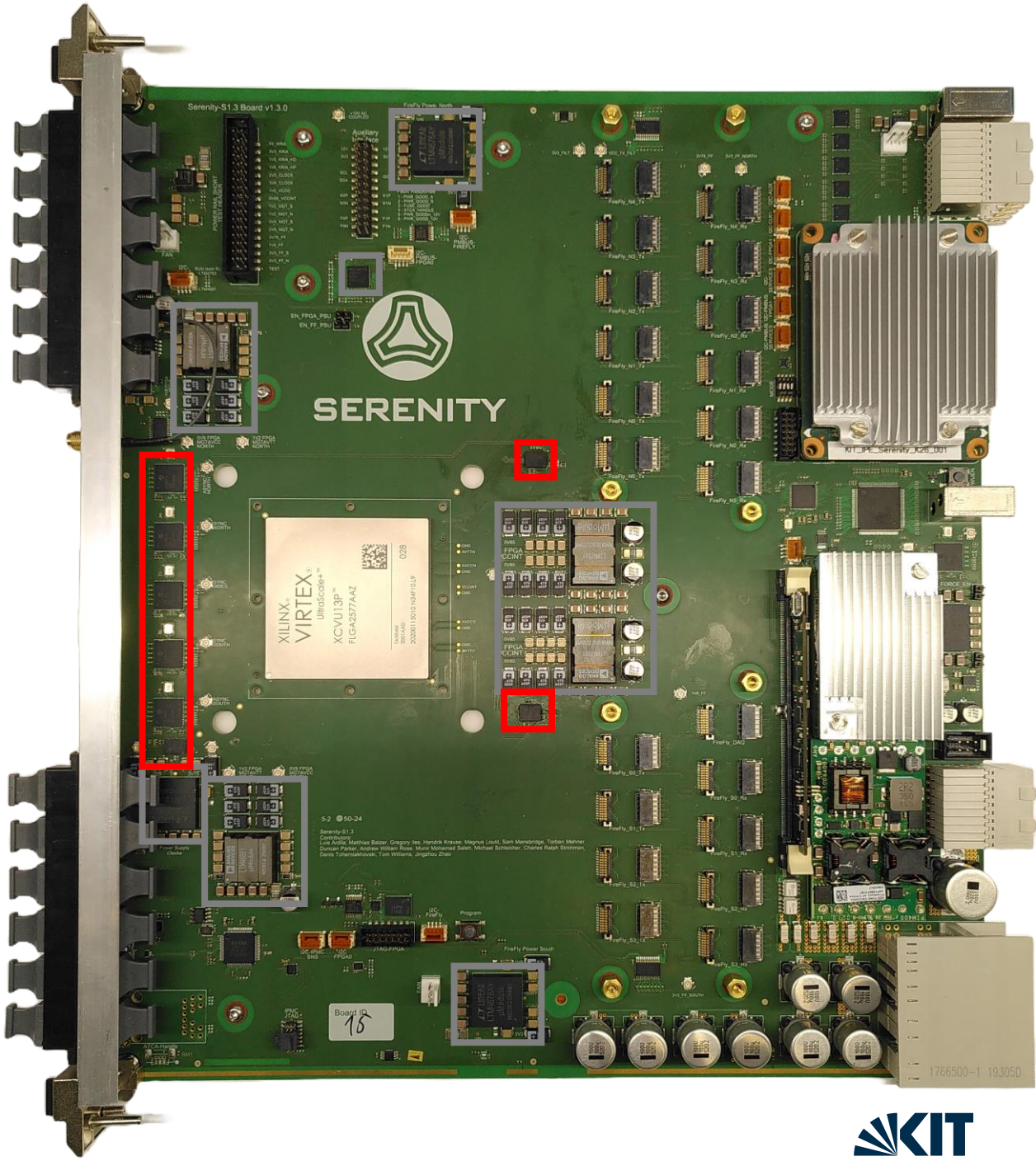
- Provide access to all I2C and PMBus devices + JTAG
  - **Power supplies** + **power sequencer**



3. Kria Test without the FPGA

Advanced board management: SMASH & Serenity Toolbox

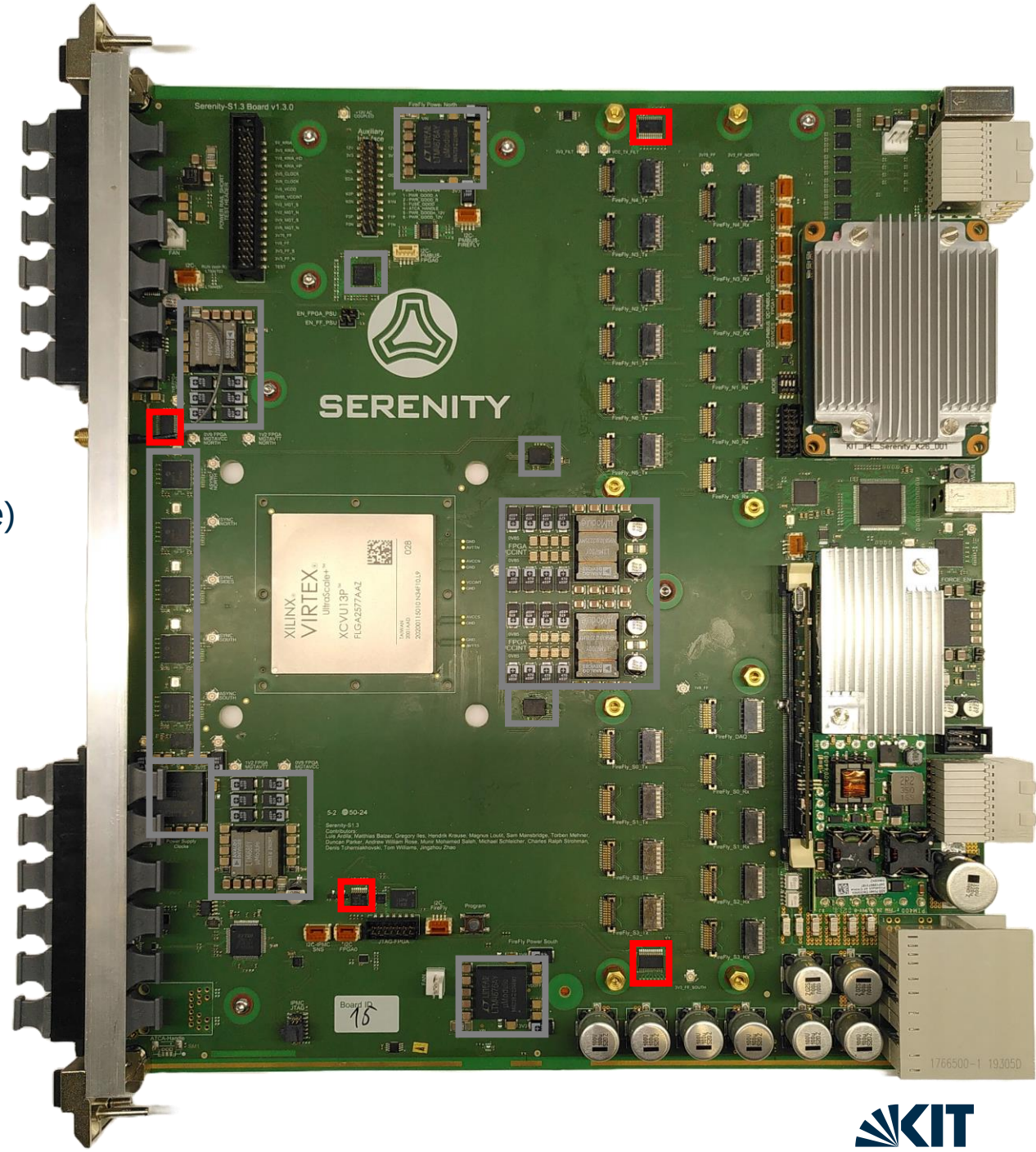
- Provide access to all I2C and PMBus devices + JTAG
  - Power supplies + power sequencer
  - **Clock chips**



3. Kria Test without the FPGA

Advanced board management: SMASH & Serenity Toolbox

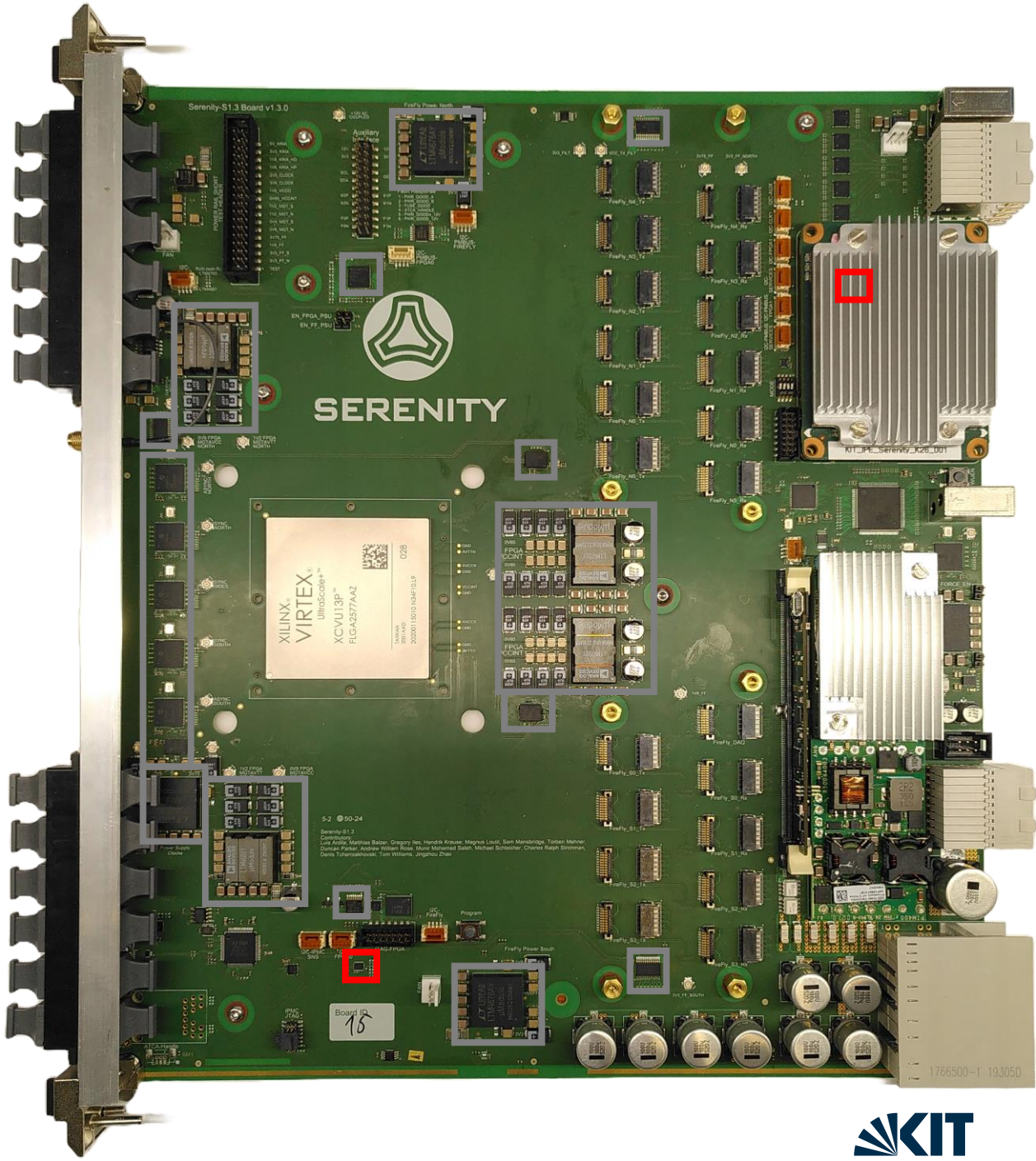
- Provide access to all I2C and PMBus devices + JTAG
  - Power supplies + power sequencer
  - Clock chips
  - **I2C I/O expander** & **multiplexer** (more on the bottom side)



3. Kria Test without the FPGA

Advanced board management: SMASH & Serenity Toolbox

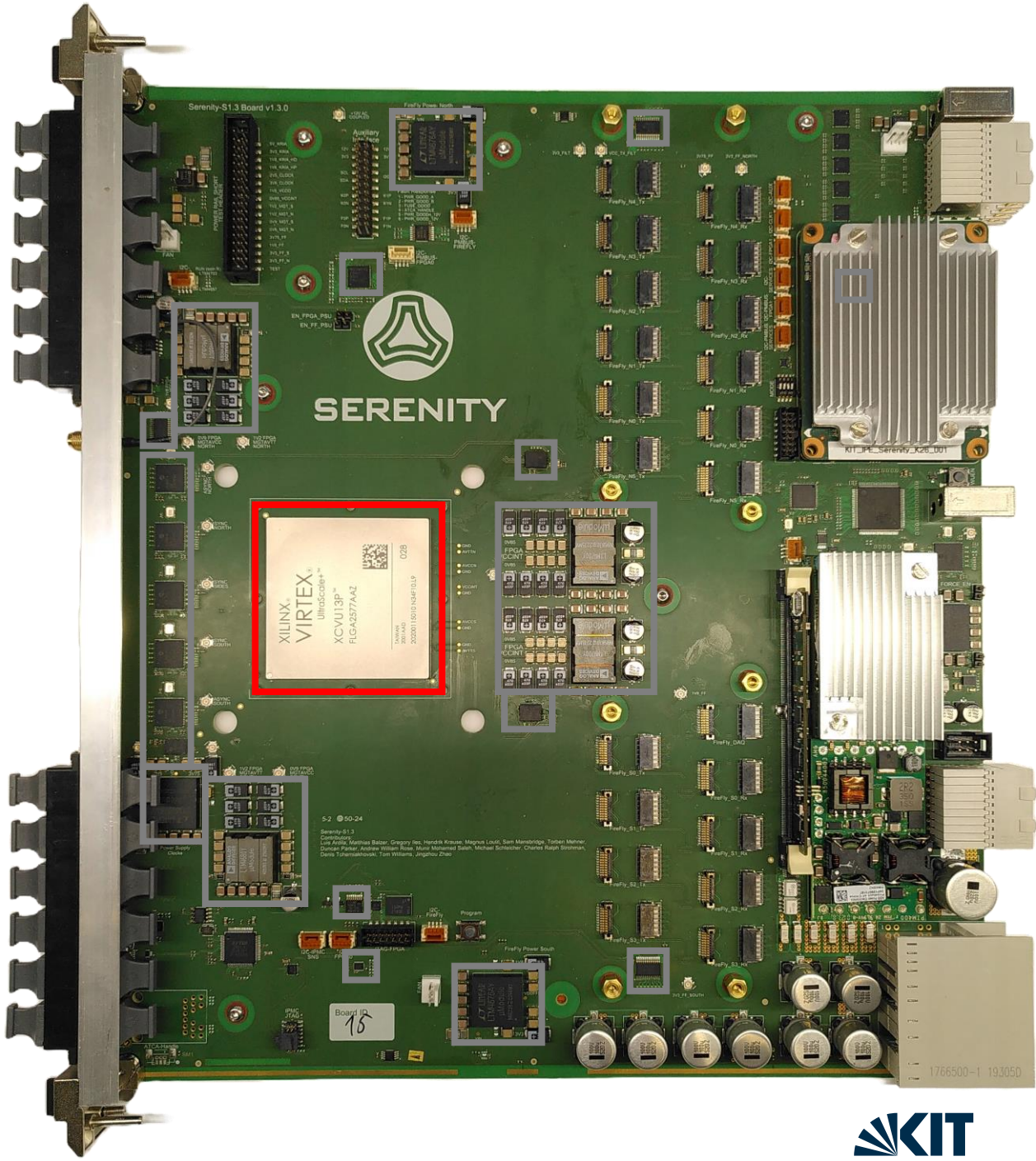
- Provide access to all I2C and PMBus devices + JTAG
  - Power supplies + power sequencer
  - Clock chips
  - I2C I/O expander & multiplexer (more on the bottom side)
  - **EEPROMs**



3. Kria Test without the FPGA

Advanced board management: SMASH & Serenity Toolbox

- Provide access to all I2C and PMBus devices + JTAG
  - Power supplies + power sequencer
  - Clock chips
  - I2C I/O expander & multiplexer (more on the bottom side)
  - EEPROMs
  - **FPGA**



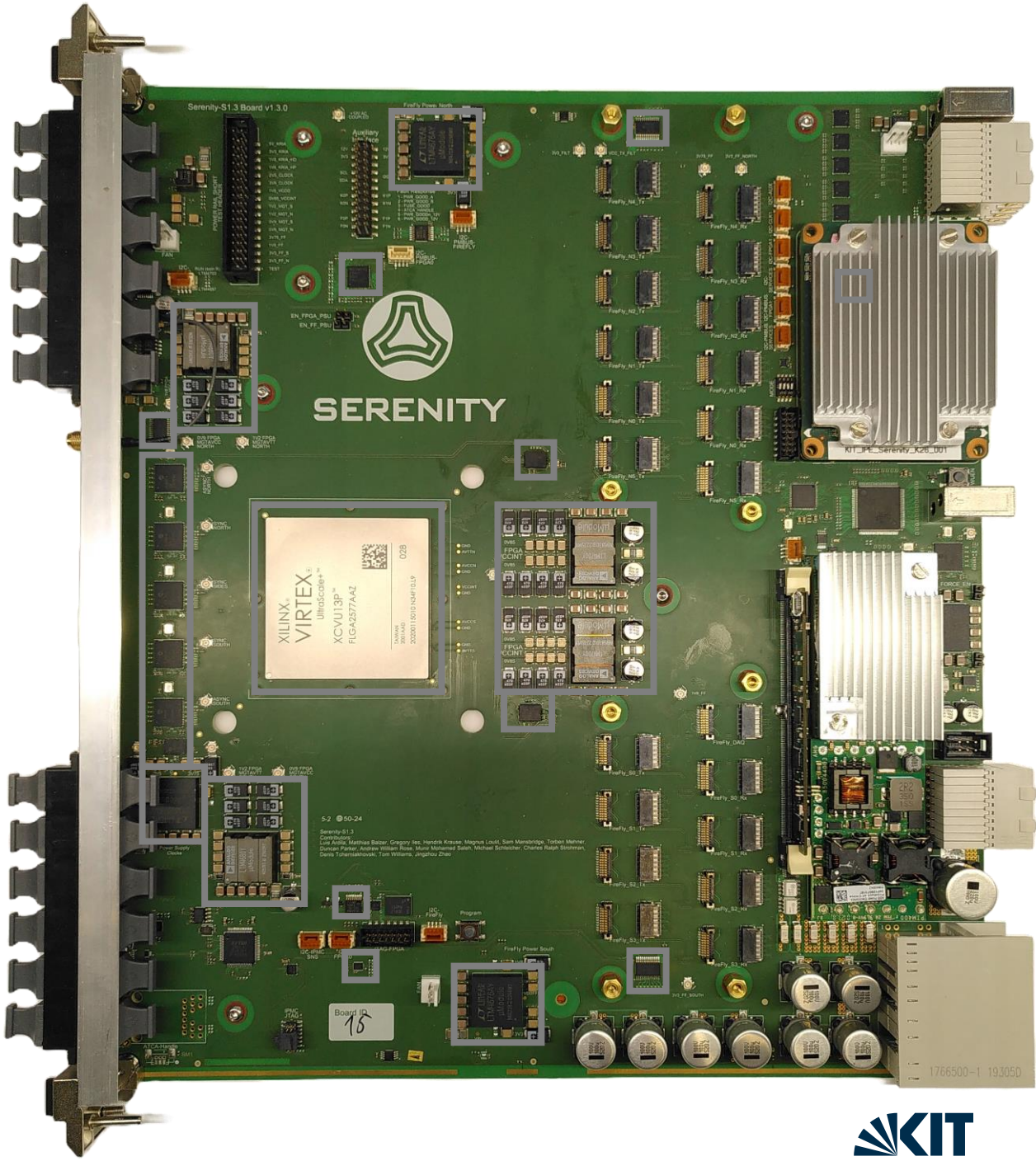
3. Kria Test without the FPGA

Advanced board management: SMASH & Serenity Toolbox

- Provide access to all I2C and PMBus devices + JTAG
  - Power supplies + power sequencer
  - Clock chips
  - I2C I/O expander & multiplexer (more on the bottom side)
  - EEPROMs
  - FPGA

Idea: Pytest can use the python API to provide a test suite to automate the configuration and commissioning process

- Well established test framework
- Test functions can be wrapped around existing functions for control, measure, configure, …



3. Kria Test without the FPGA

Advanced board management: SMASH & Serenity Toolbox

- Provide access to all I2C and PMBus devices + JTAG
  - Power supplies + power sequencer
  - Clock chips
  - I2C I/O expander & multiplexer (more on the bottom side)
  - EEPROMs
  - FPGA

Idea: Pytest can use the python API to provide a test suite to automate the configuration and commissioning process

- Well established test framework
- Test functions can be wrapped around existing functions for control, measure, configure, …

```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)
```

3. Kria Test without the FPGA

Advanced board management: SMASH & Serenity Toolbox

- Provide access to all I2C and PMBus devices + JTAG
  - Power supplies + power sequencer
  - Clock chips
  - I2C I/O expander & multiplexer (more on the bottom side)
  - EEPROMs
  - FPGA

Idea: Pytest can use the python API to provide a test suite to automate the configuration and commissioning process

- Well established test framework
- Test functions can be wrapped around existing functions for control, measure, configure, …

Dependencies

```
@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)
```

3. Kria Test without the FPGA

Advanced board management: SMASH & Serenity Toolbox

- Provide access to all I2C and PMBus devices + JTAG
  - Power supplies + power sequencer
  - Clock chips
  - I2C I/O expander & multiplexer (more on the bottom side)
  - EEPROMs
  - FPGA

Idea: Pytest can use the python API to provide a test suite to automate the configuration and commissioning process

- Well established test framework
- Test functions can be wrapped around existing functions for control, measure, configure, …

Dependencies

```
@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)
```

Parameter

3. Kria Test without the FPGA

Advanced board management: SMASH & Serenity Toolbox

- Provide access to all I2C and PMBus devices + JTAG
  - Power supplies + power sequencer
  - Clock chips
  - I2C I/O expander & multiplexer (more on the bottom side)
  - EEPROMs
  - FPGA

Idea: Pytest can use the python API to provide a test suite to automate the configuration and commissioning process

- Well established test framework
- Test functions can be wrapped around existing functions for control, measure, configure, …

Dependencies

```
@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)
```

Parameter

SMASH API call

3. Kria Test without the FPGA

Advanced board management: SMASH & Serenity Toolbox

- Provide access to all I2C and PMBus devices + JTAG
  - Power supplies + power sequencer
  - Clock chips
  - I2C I/O expander & multiplexer (more on the bottom side)
  - EEPROMs
  - FPGA

Idea: Pytest can use the python API to provide a test suite to automate the configuration and commissioning process

- Well established test framework
- Test functions can be wrapped around existing functions for control, measure, configure, …

Dependencies

```
@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)
```

Parameter

SMASH API call

Evaluation

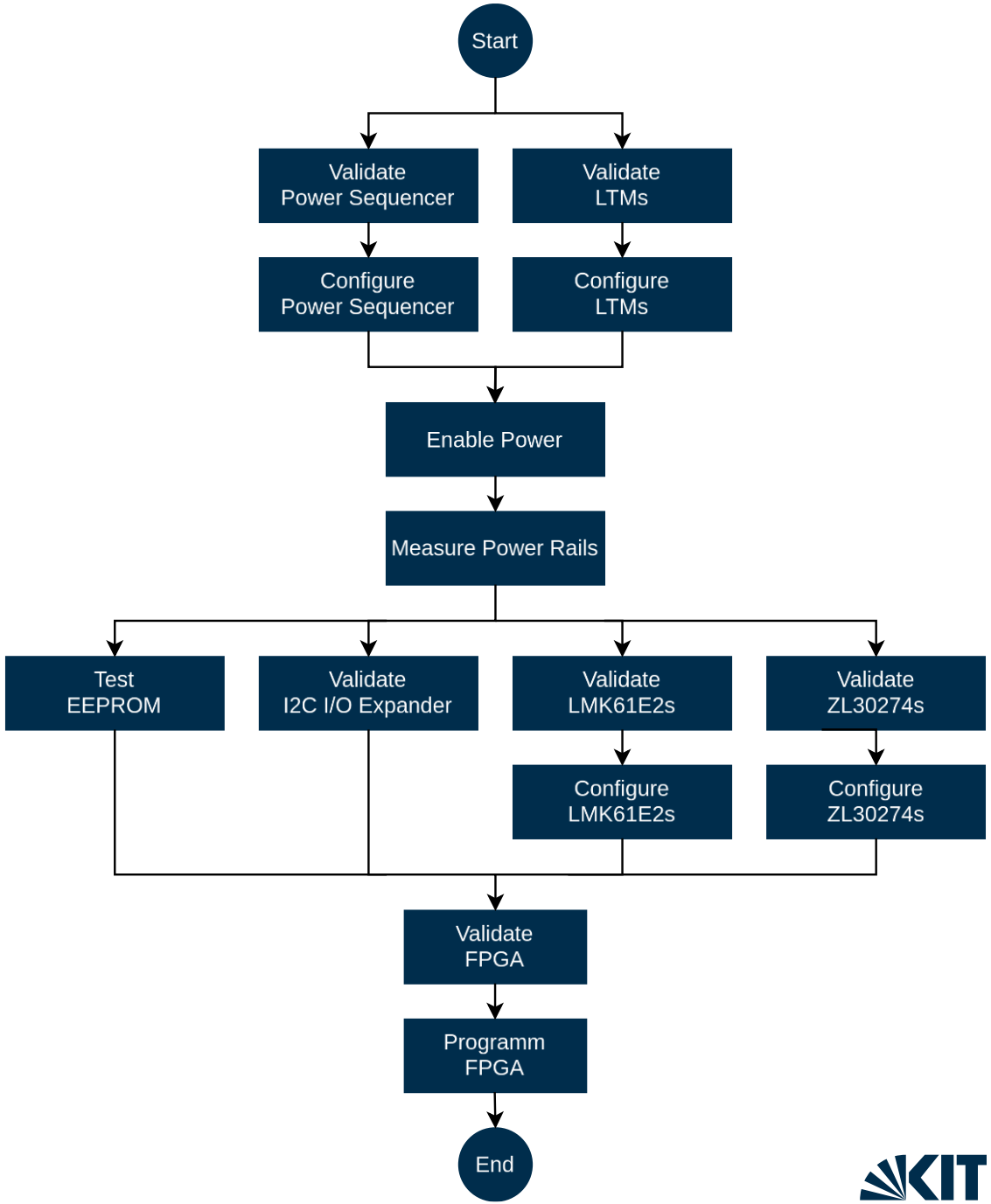
3. Kria Test without the FPGA

Advanced board management: SMASH & Serenity Toolbox

- Provide access to all I2C and PMBus devices + JTAG
  - Power supplies + power sequencer
  - Clock chips
  - I2C I/O expander & multiplexer (more on the bottom side)
  - EEPROMs
  - FPGA

Idea: Pytest can use the python API to provide a test suite to automate the configuration and commissioning process

- Well established test framework
- Test functions can be wrapped around existing functions for control, measure, configure, …



3. Kria Test without the FPGA

Advanced board management: SMASH & Serenity Toolbox

- Provide access to all I2C and PMBus devices + JTAG
  - Power supplies + power sequencer
  - Clock chips
  - I2C I/O expander & multiplexer (more on the bottom side)
  - EEPROMs
  - FPGA

Idea: Pytest can use the python API to provide a test suite to automate the configuration and commissioning process

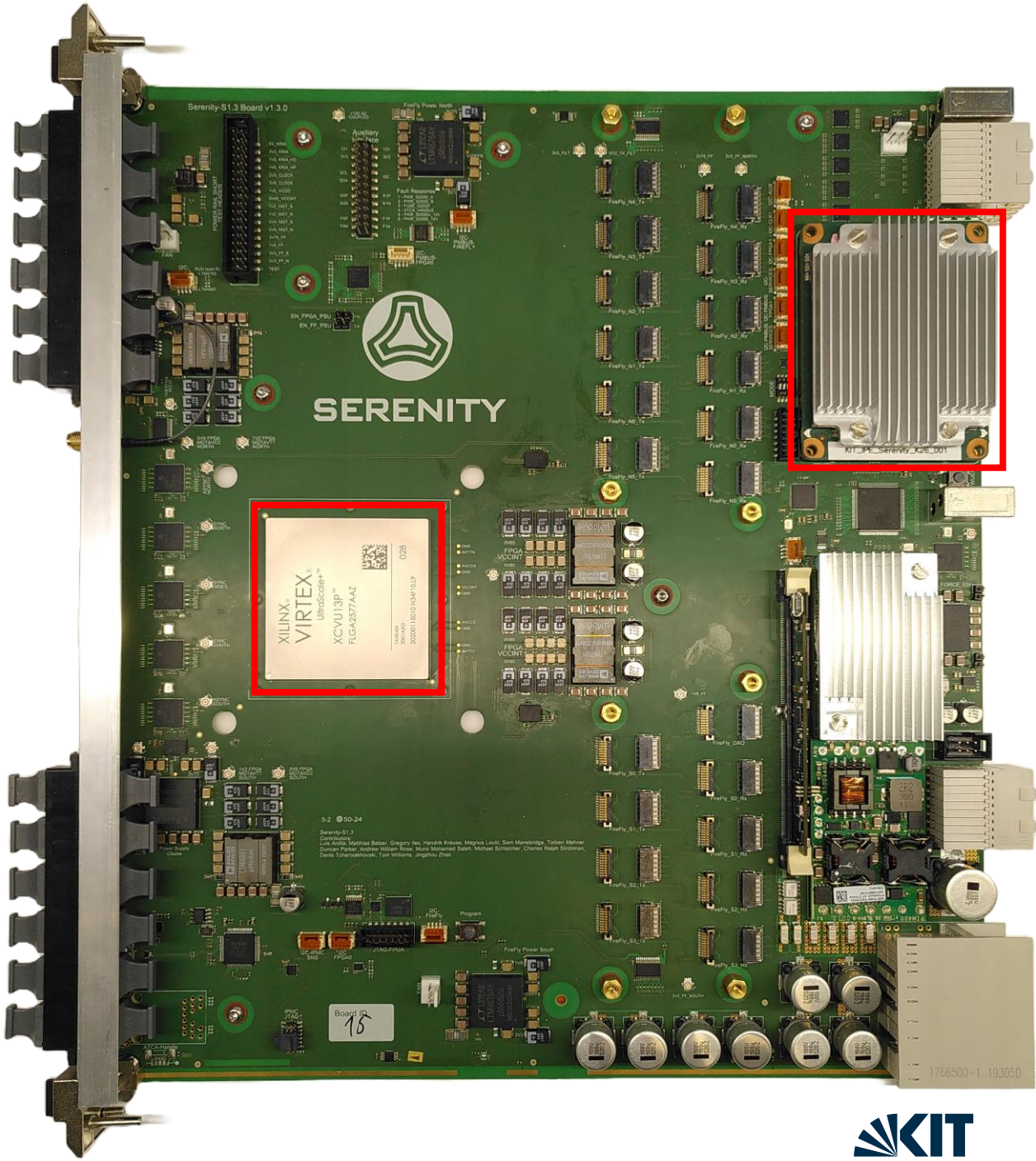
- Well established test framework
- Test functions can be wrapped around existing functions for control, measure, configure, …

```
[root@mgmt-kki ~]# python -m serenity.tests -k 'test_commissioning' -v --tb=no --config /root/serenity-toolbox/examples/serenity-s.yaml --bitff
le /root/firmware/s1_vu13p-2_max.bit
========================================================== test session starts ==========================================================
platform linux -- Python 3.11.10, pytest-7.0.1, pluggy-1.5.0 -- /usr/bin/python3.11
cachedir: .pytest_cache
rootdir: /root
plugins: dependency-0.6.0, logger-1.1.1
collected 78 items / 12 deselected / 66 selected

serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_validate_max34451[MAX34451-U:17] PASSED                        [  1%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_configure_max34451[MAX34451-U:17] PASSED                      [  3%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_validate_LTM4xx[LTM4676A_Clocks-U:3] PASSED                   [  4%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_validate_LTM4xx[LTM4680_MGT_NORTH-U:12] PASSED                [  6%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_validate_LTM4xx[LTM4680_MGT_SOUTH-U:15] PASSED                [  7%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_validate_LTM4xx[LTM4700_VCCINT_1-U:29] PASSED                [  9%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_validate_LTM4xx[LTM4700_VCCINT_0-U:30] PASSED                [ 10%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_validate_LTM4xx[LTM4676A FireFly North-U:23] PASSED            [ 12%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_validate_LTM4xx[LTM4676A FireFly South-U:24] PASSED            [ 13%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_configure_LTM4xxx[LTM4676A_Clocks-U:3] PASSED                 [ 15%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_configure_LTM4xxx[LTM4680_MGT_NORTH-U:12] PASSED              [ 16%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_configure_LTM4xxx[LTM4680_MGT_SOUTH-U:15] PASSED              [ 18%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_configure_LTM4xxx[LTM4700_VCCINT_1-U:29] PASSED              [ 19%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_configure_LTM4xxx[LTM4700_VCCINT_0-U:30] PASSED              [ 21%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_configure_LTM4xxx[LTM4676A FireFly North-U:23] PASSED          [ 22%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_configure_LTM4xxx[LTM4676A FireFly South-U:24] PASSED          [ 24%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_turn_power_on[MAX34451-U:17] PASSED                          [ 25%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_power_rails[3V3_FF_NORTH-VOUT00-3300] PASSED           [ 27%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_power_rails[3V3_FF_SOUTH-VOUT01-3300] PASSED           [ 28%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_power_rails[1V8_FF-VOUT02-1800] PASSED                [ 30%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_power_rails[3V75_FF-VOUT03-3750] PASSED               [ 31%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_power_rails[0V9_FPGA0_MGTAVCC_NORTH-VOUT04-900] PASSED  [ 33%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_power_rails[0V9_FPGA0_MGTAVCC_SOUTH-VOUT05-900] PASSED  [ 34%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_power_rails[1V2_FPGA0_MGTAVTT_NORTH-VOUT06-1200] PASSED [ 36%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_power_rails[1V2_FPGA0_MGTAVTT_SOUTH-VOUT07-1200] PASSED [ 37%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_power_rails[0V85_FPGA0_VCCINT-VOUT08-850] PASSED      [ 39%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_power_rails[1V8_FPGA0_VCCO-VOUT09-1800] PASSED        [ 40%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_power_rails[3V3_3V8_CLOCK-VOUT10-3750] PASSED         [ 42%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_power_rails[1V8_2V5_CLOCK-VOUT11-2500] PASSED         [ 43%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_power_rails[1V8_KRIA_HP-VOUT12-1800] PASSED           [ 45%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_power_rails[1V8_KRIA_HD-VOUT13-1800] PASSED           [ 46%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_power_rails[3V3_KRIA-VOUT14-3300] PASSED             [ 48%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_power_rails[5V_KRIA-VOUT15-5000] PASSED              [ 50%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_turn_power_off[MAX34451-U:17] SKIPPED (test_turn_power_off[M...) [ 51%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_validate_lmk61e2[Osc_TCDS-U:10] PASSED                        [ 53%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_validate_lmk61e2[Osc_FREE-U:27] PASSED                        [ 54%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_validate_lmk61e2[Osc_C2C-U:26] PASSED                        [ 56%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_configure_lmk61e2[Osc_TCDS-U:10] PASSED                      [ 57%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_configure_lmk61e2[Osc_FREE-U:27] PASSED                      [ 59%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_configure_lmk61e2[Osc_C2C-U:26] PASSED                      [ 60%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_validate_zl30274[ASYNC_SOUTH-U:4] PASSED                    [ 62%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_validate_zl30274[SYNC_SOUTH-U:5] PASSED                      [ 63%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_validate_zl30274[SYNC_SIDES-U:6] PASSED                      [ 65%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_validate_zl30274[SYNC_NORTH-U:7] PASSED                      [ 66%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_validate_zl30274[ASYNC_NORTH-U:8] PASSED                    [ 68%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_configure_zl30274[ASYNC_SOUTH-U:4] PASSED                    [ 69%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_configure_zl30274[SYNC_SOUTH-U:5] PASSED                      [ 71%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_configure_zl30274[SYNC_SIDES-U:6] PASSED                      [ 72%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_configure_zl30274[SYNC_NORTH-U:7] PASSED                      [ 74%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_configure_zl30274[ASYNC_NORTH-U:8] PASSED                    [ 75%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_read_write_EEPROMs[None-name0-designator0] SKIPPED (got empt...) [ 77%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_pcal6408[I/O Exp. Zone 3 Buffer-IC:15] PASSED          [ 78%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_pcal6408[I/O Exp. Clock Reset-IC:1] PASSED             [ 80%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_pcal6408[I/O Exp. FPGA-IC:4] PASSED                   [ 81%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_pcal6408[I/O Exp. FF S3-IC:16] PASSED                [ 83%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_pcal6408[I/O Exp. FF S2-IC:17] PASSED                [ 84%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_pcal6408[I/O Exp. FF S1-IC:18] PASSED                [ 86%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_pcal6408[I/O Exp. FF S0-IC:19] PASSED                [ 87%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_pcal6408[I/O Exp. FF DAQ-IC:20] PASSED               [ 89%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_pcal6408[I/O Exp. FF N5-IC:21] PASSED                [ 90%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_pcal6408[I/O Exp. FF N0-IC:22] PASSED                [ 92%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_pcal6408[I/O Exp. FF N1-IC:23] PASSED                [ 93%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_pcal6408[I/O Exp. FF N2-IC:24] PASSED                [ 95%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_pcal6408[I/O Exp. FF N3-IC:25] PASSED                [ 96%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_measure_pcal6408[I/O Exp. FF N4-IC:26] PASSED                [ 98%]
serenity-toolbox/python/pkg/serenity/tests/test_commissioning.py::test_programm_fpga[/root/firmware/s1_vu13p-2_max.bit] PASSED      [100%]

============================================ 64 passed, 2 skipped, 12 deselected in 39.36s ==============================================
[root@mgmt-kki ~]#
[root@mgmt-kki ~]#
[root@mgmt-kki ~]#
[root@mgmt-kki ~]#
[root@mgmt-kki ~]#
```

4. Kria Test with the FPGA

Advanced board management: EMP Toolbox + EMP Framework

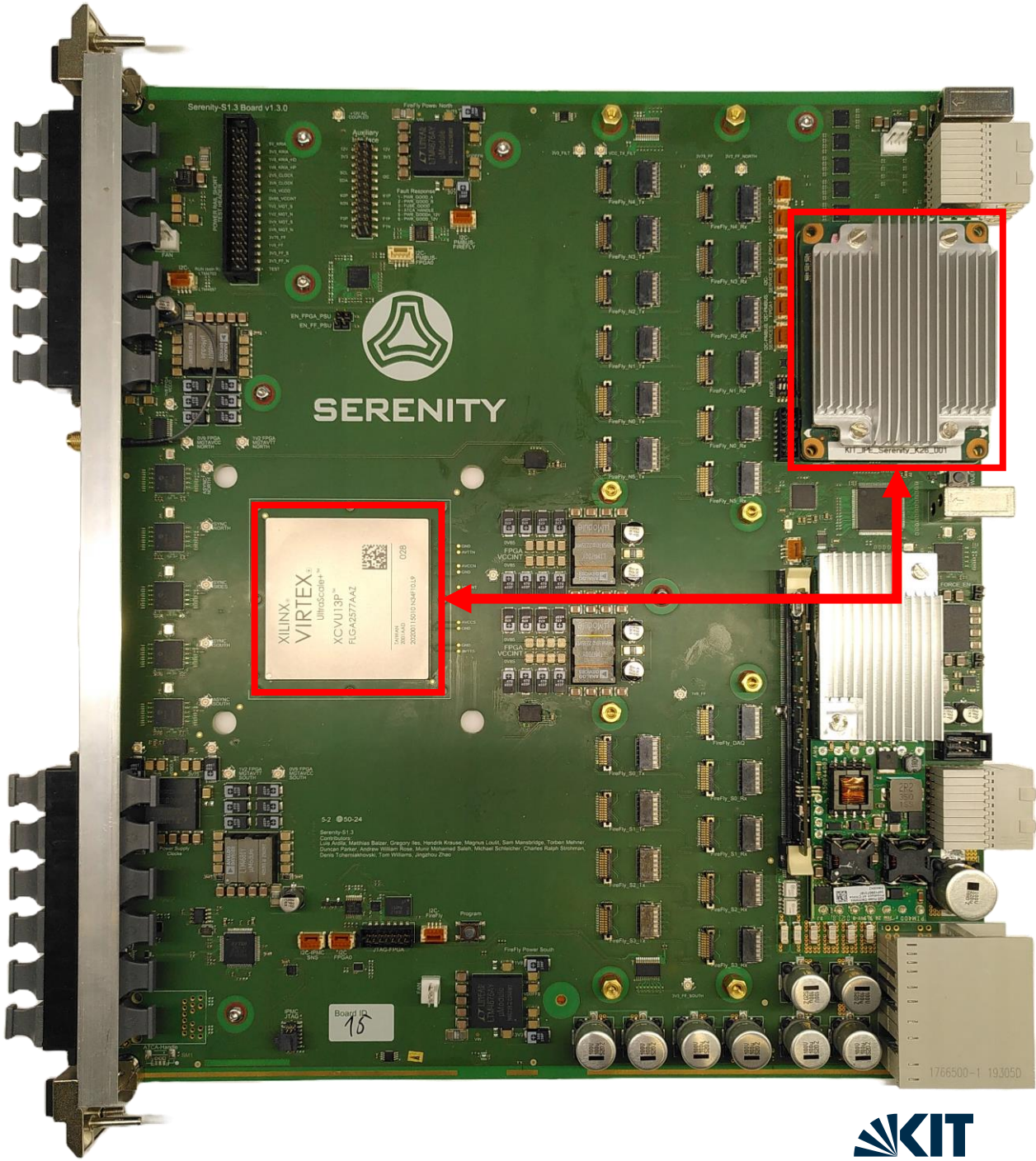
- Provide access to IPBus register (over AXI C2C)
- Framework already provides feature usable for testing



4. Kria Test with the FPGA

Advanced board management: EMP Toolbox + EMP Framework

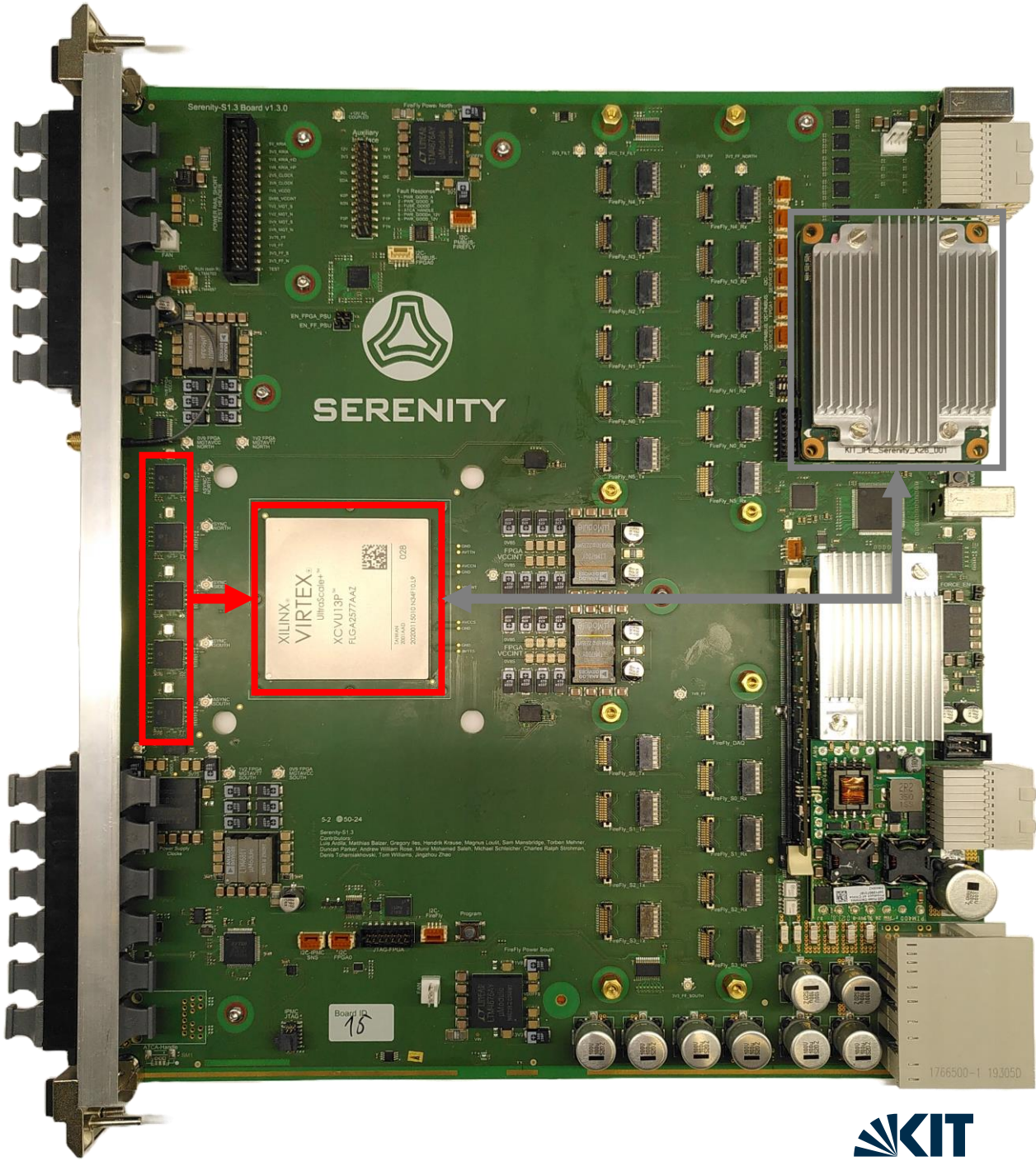
- Provide access to IPBus register (over AXI C2C)
- Framework already provides feature usable for testing
  - **AXI C2C connection**



4. Kria Test with the FPGA

Advanced board management: EMP Toolbox + EMP Framework

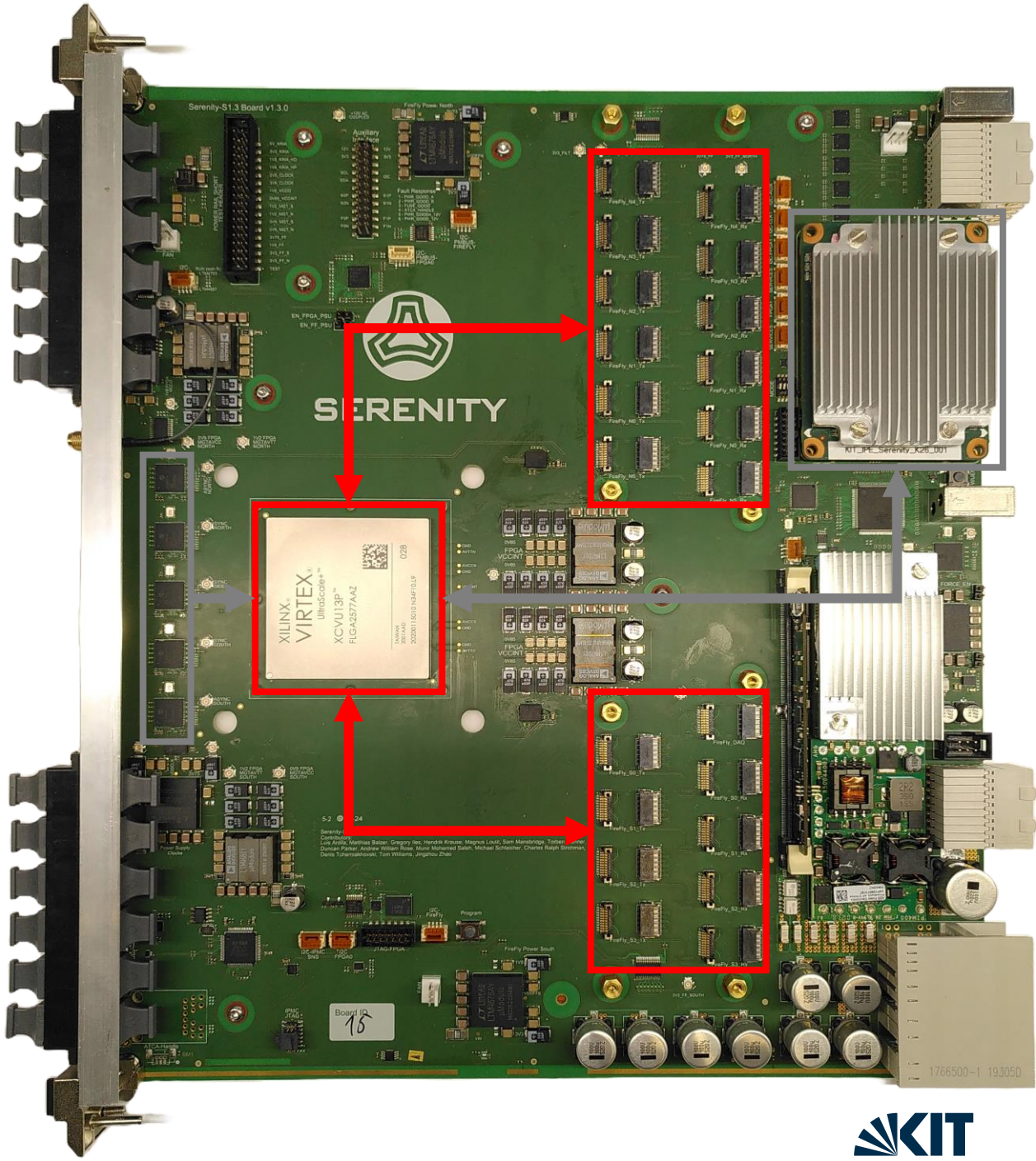
- Provide access to IPBus register (over AXI C2C)
- Framework already provides feature usable for testing
  - AXI C2C connection
  - **Frequency measurements**



4. Kria Test with the FPGA

Advanced board management: EMP Toolbox + EMP Framework

- Provide access to IPBus register (over AXI C2C)
- Framework already provides feature usable for testing
  - AXI C2C connection
  - Frequency measurements
  - **Bit error rate testing**



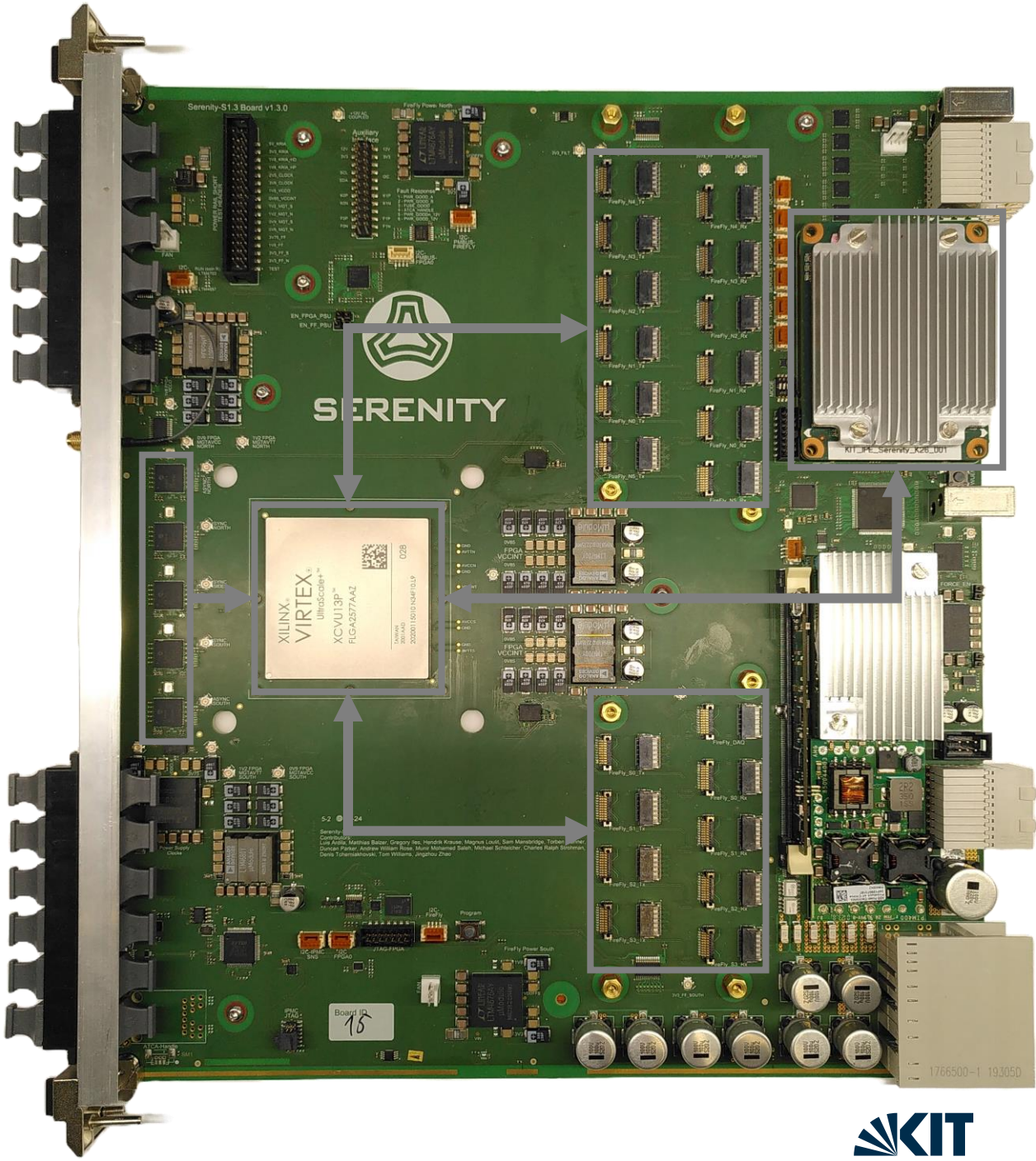
4. Kria Test with the FPGA

Advanced board management: EMP Toolbox + EMP Framework

- Provide access to IPBus register (over AXI C2C)
- Framework already provides feature usable for testing
  - AXI C2C connection
  - Frequency measurements
  - Bit error rate testing

Idea: Pytest can use the python API to extend the test suite to the FPGA to further commission the board

- Well established test framework
- Test functions can be wrapped around existing functions for control, measure, configure, …



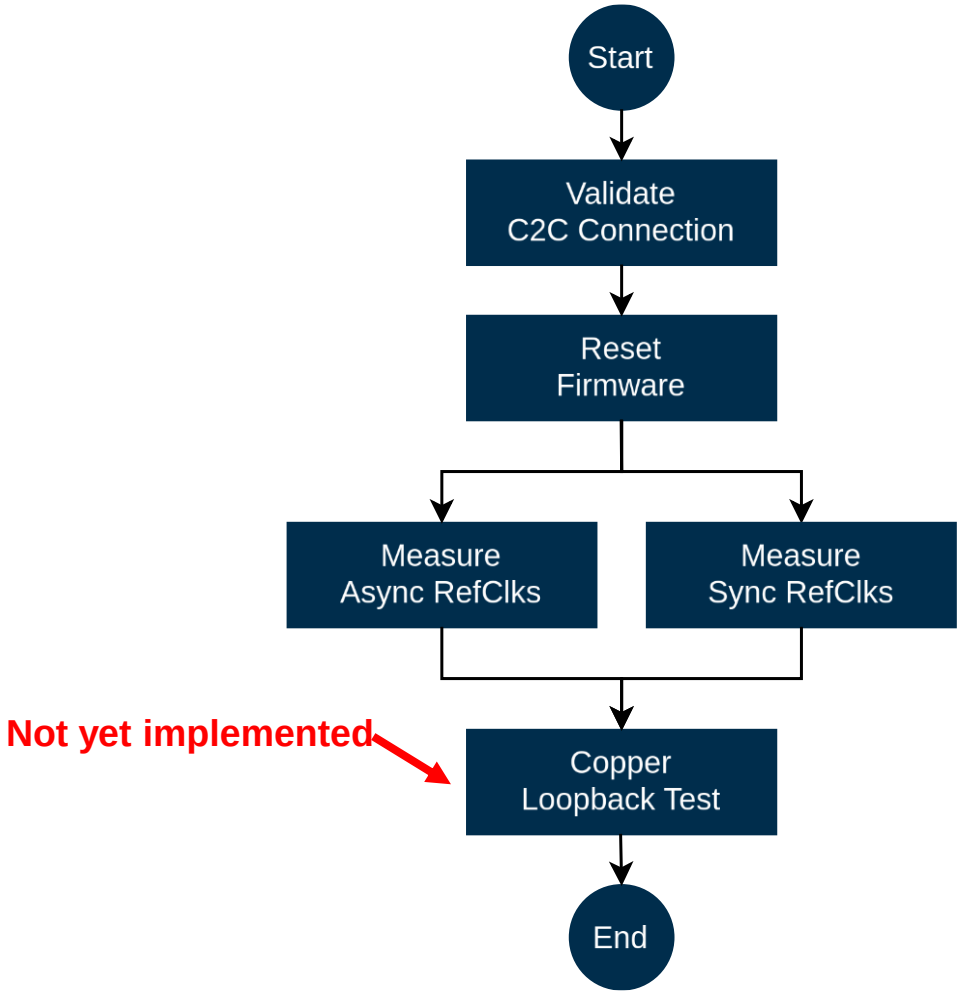
4. Kria Test with the FPGA

Advanced board management: EMP Toolbox + EMP Framework

- Provide access to IPBus register (over AXI C2C)
- Framework already provides feature usable for testing
  - AXI C2C connection
  - Frequency measurements
  - Bit error rate testing

Idea: Pytest can use the python API to extend the test suite to the FPGA to further commission the board

- Well established test framework
- Test functions can be wrapped around existing functions for control, measure, configure, …



4. Kria Test with the FPGA

Advanced board management: EMP Toolbox + EMP Framework

- Provide access to IPBus register (over AXI C2C)
- Framework already provides feature usable for testing
  - AXI C2C connection
  - Frequency measurements
  - Bit error rate testing

Idea: Pytest can use the python API to extend the test suite to the FPGA to further commission the board

- Well established test framework
- Test functions can be wrapped around existing functions for control, measure, configure, …

```
[root@mgmt-kki ~]# python -m emp.tests -k 'test_mgtrefclks' -v --tb=no --conn connections.xml --device VU13P
========================================================== test session starts ==========================================================
platform linux -- Python 3.11.10, pytest-7.0.1, pluggy-1.5.0 -- /usr/bin/python3.11
cachedir: .pytest_cache
rootdir: /root
plugins: dependency-0.6.0, logger-1.1.1
collected 2193 items / 2160 deselected / 33 selected

emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_reset[VU13P] PASSED                                                    [  3%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_async_refclks[VU13P-17-320.0-ASYNC_NORTH out0] PASSED          [  6%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_async_refclks[VU13P-19-320.0-ASYNC_NORTH out1] PASSED          [  9%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_async_refclks[VU13P-27-320.0-ASYNC_NORTH out2] PASSED          [ 12%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_async_refclks[VU13P-29-320.0-ASYNC_NORTH out3] PASSED          [ 15%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_async_refclks[VU13P-31-320.0-ASYNC_NORTH out4] PASSED          [ 18%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_async_refclks[VU13P-23-320.0-ASYNC_NORTH out5] PASSED          [ 21%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_async_refclks[VU13P-25-320.0-ASYNC_NORTH out6] PASSED          [ 24%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_async_refclks[VU13P-21-320.0-ASYNC_NORTH out7] PASSED          [ 27%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_async_refclks[VU13P-14-320.0-ASYNC_SOUTH out0] PASSED          [ 30%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_async_refclks[VU13P-12-320.0-ASYNC_SOUTH out1] PASSED          [ 33%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_async_refclks[VU13P-10-320.0-ASYNC_SOUTH out2] PASSED          [ 36%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_async_refclks[VU13P-4-320.0-ASYNC_SOUTH out3] PASSED           [ 39%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_async_refclks[VU13P-2-320.0-ASYNC_SOUTH out4] PASSED           [ 42%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_async_refclks[VU13P-6-320.0-ASYNC_SOUTH out5] PASSED           [ 45%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_async_refclks[VU13P-8-320.0-ASYNC_SOUTH out6] PASSED           [ 48%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_async_refclks[VU13P-1-320.0-ASYNC_SOUTH out7] PASSED           [ 51%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_sync_refclks[VU13P-25-320.0-SYNC_NORTH out0] PASSED           [ 54%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_sync_refclks[VU13P-27-320.0-SYNC_NORTH out1] PASSED           [ 57%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_sync_refclks[VU13P-29-320.0-SYNC_NORTH out2] PASSED           [ 60%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_sync_refclks[VU13P-31-320.0-SYNC_NORTH out3] PASSED           [ 63%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_sync_refclks[VU13P-23-320.0-SYNC_NORTH out4] PASSED           [ 66%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_sync_refclks[VU13P-21-320.0-SYNC_NORTH out5] PASSED           [ 69%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_sync_refclks[VU13P-19-320.0-SYNC_NORTH out6] PASSED           [ 72%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_sync_refclks[VU13P-17-320.0-SYNC_NORTH out7] PASSED           [ 75%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_sync_refclks[VU13P-1-320.0-SYNC_SOUTH out0] PASSED            [ 78%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_sync_refclks[VU13P-2-320.0-SYNC_SOUTH out1] PASSED            [ 81%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_sync_refclks[VU13P-4-320.0-SYNC_SOUTH out2] PASSED            [ 84%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_sync_refclks[VU13P-12-320.0-SYNC_SOUTH out3] PASSED           [ 87%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_sync_refclks[VU13P-14-320.0-SYNC_SOUTH out4] PASSED           [ 90%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_sync_refclks[VU13P-10-320.0-SYNC_SOUTH out5] PASSED           [ 93%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_sync_refclks[VU13P-8-320.0-SYNC_SOUTH out6] PASSED            [ 96%]
emp-toolbox/python/pkg/emp/tests/test_mgtrefclks.py::test_measure_sync_refclks[VU13P-6-320.0-SYNC_SOUTH out7] PASSED            [100%]

========================================================= 33 passed, 2160 deselected in 38.89s =========================================
[root@mgmt-kki ~]#
```

First Results

FAT usage at KIT

- 3 Serenity-S1.3 boards where tested
- Identification of 5 damaged LTM4676As
  (likely due to none ideal storage and humidity)
- No errors with new parts

First Results

FAT usage at KIT

- 3 Serenity-S1.3 boards where tested
- Identification of 5 damaged LTM4676As
  (likely due to none ideal storage and humidity)
- No errors with new parts

FAT usage at the Company

- 2 Serenity-S1.3 boards tested (more in the next weeks)
- Successful usage of the FAT (after initial problems) with no board errors detected

First Results

FAT usage at KIT

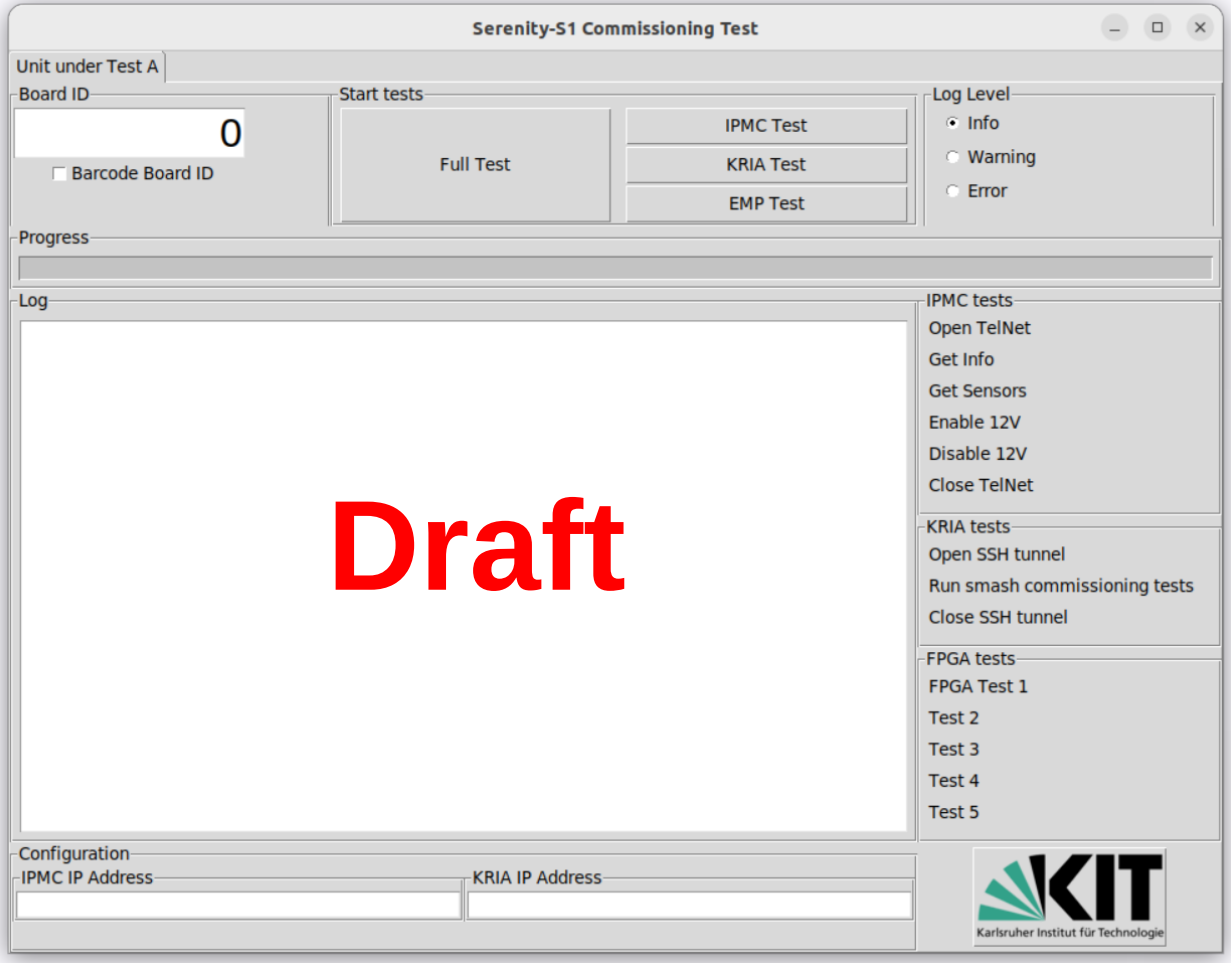
- 3 Serenity-S1.3 boards where tested
- Identification of 5 damaged LTM4676As (likely due to none ideal storage and humidity)
- No errors with new parts

FAT usage at the Company

- 2 Serenity-S1.3 boards tested (more in the next weeks)
- Successful usage of the FAT (after initial problems) with no board errors detected

Feedback from the Manufacturer

- Short-Circuit test should be separated from the other tests
- The code being editable is very risky and should be avoided
- A simple, buttoned user-interface is preferred



Draft

Conclusion

- A Factory Acceptance Test was developed for Serenity-S1 to allow an automated commissioning directly at the factory
- The test leverages already existing board infrastructure and tools (SMASH and EMP) minimizing additional development effort
- Pytest provides a sophisticated way to structure and automate the commissioning process
- First real-world usage during the extended pilot-production was successful and provided valuable feedback pathing the way toward pre-production and main-production
- Further development to increase robustness, extend the test cases and a button-based GUI is ongoing