METASAT

MODULAR MODEL-BASED DESIGN AND TESTING FOR APPLICATIONS IN SATELLITES

The METASAT Space Platform: High Performance On-Board Processing for Institutional Missions Using Multicores, GPU and AI Accelerator

Leonidas Kosmidis, Marc Solé, Jannis Wolf, Aridane Álvarez, Matina Maria Trompouki, Eckart Göhler, Alfred Hönle

RISC-V IN SPACE

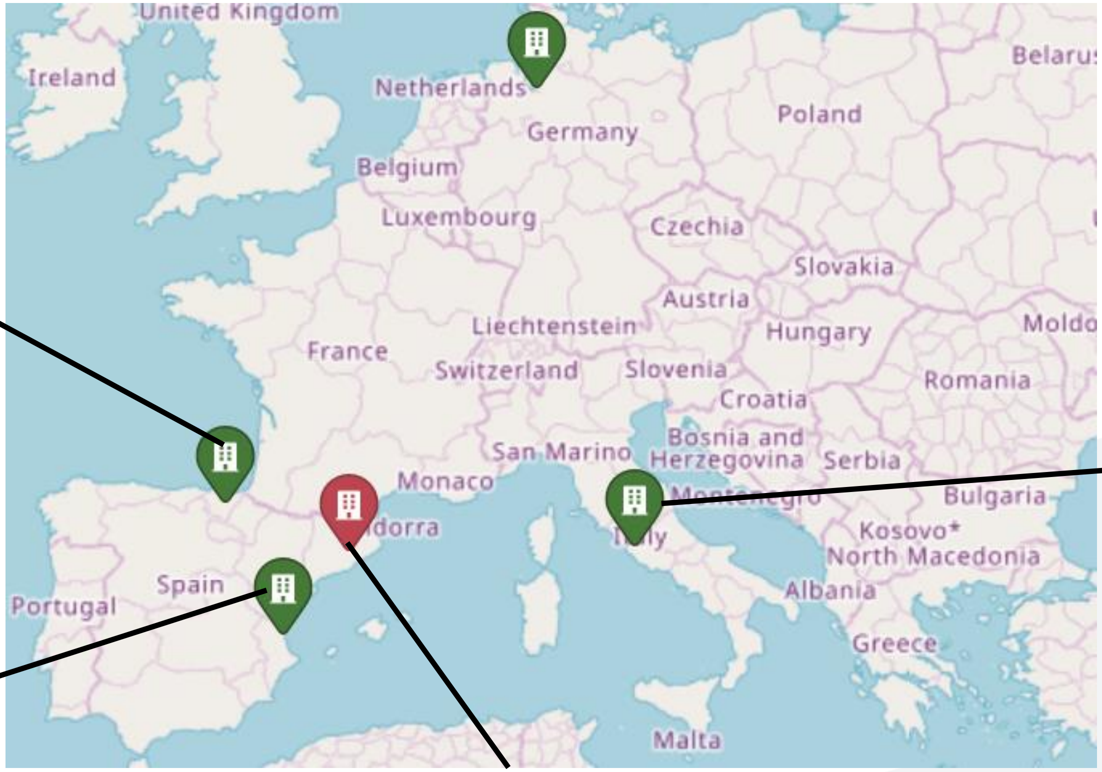
02/04/2025

METASAT Consortium

METASAT

- 2-year Horizon Europe project: January 2023-December 2024
- TRL 3-4

OHB



ikerlan
MEMBER OF BASQUE RESEARCH
& TECHNOLOGY ALLIANCE

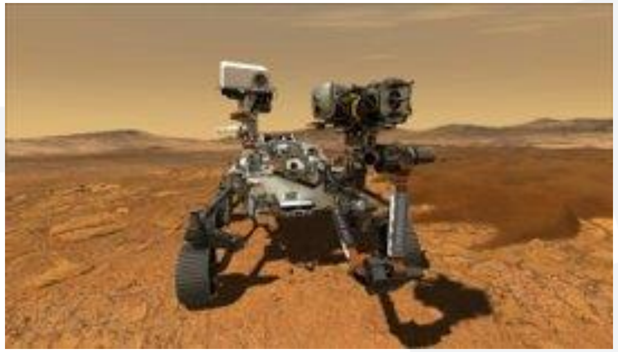
fentISS
FENT Innovative Software Solutions

Collins Aerospace

BSC
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Introduction

- Modern and upcoming space systems require increasing levels of computing power
- Traditional space processors cannot provide this performance level
- Need for higher performance hardware in space systems



METASAT Overview

- Modern aerospace systems require new, advanced functionalities
  - Artificial Intelligence (AI)
  - High Resolution Sensors
  - Optical communications
  - Advanced Robotics…
- Advanced functionalities require complex hardware and software compared to the existing space technologies
- High Performance Hardware technologies: Advanced Multi-cores, GPUs, AI accelerators
- Programming high performance hardware requires complex software: parallel and GPU programming

Traditional Space Systems for Institutional Missions

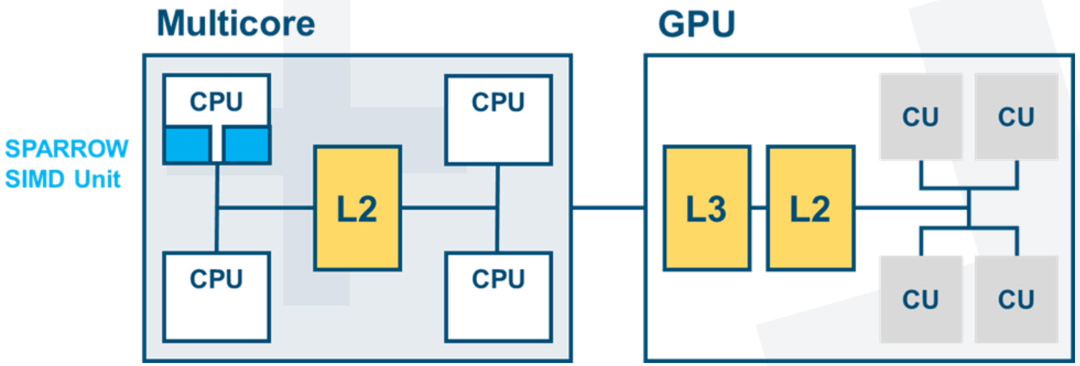
- Space-grade processors
- Space-grade FPGAs
- Each subsystem implemented in different hardware
- Software of each subsystem has different criticality
  - Bare metal or RTOS (e.g. RTEMS)

METASAT Approach

- Use a complex, highly capable space processor SoC
- Integrate multiple functionalities in a single platform
  - Similar to the Integrated Modular Avionics concept (IMA)
- Hardware cost reduction
- Mixed Criticality support through time and space partitioning
  - Software qualification cost reduction
- Use Model-Based Design to manage complexity

Hardware Selection

- No hardware with high-performance and architectural complexity exists for the space domain
- COTS Embedded Multicore and GPU devices provide these features but depend on non-qualifiable software stacks
  - GPU drivers available only for Linux
  - Blocking point for use in institutional missions
- Design a prototype hardware platform based on the RISC-V ISA

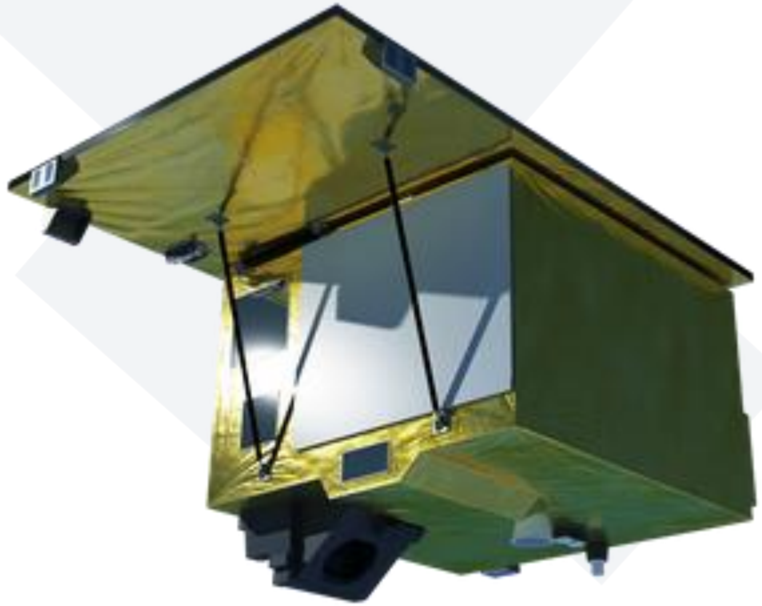


METASAT Use cases

- Several independent use cases
  - Different processing and acceleration requirements
- Representative of different flight software criticalities
- All use cases were integrated in a single platform
  - High degree of integration was achieved

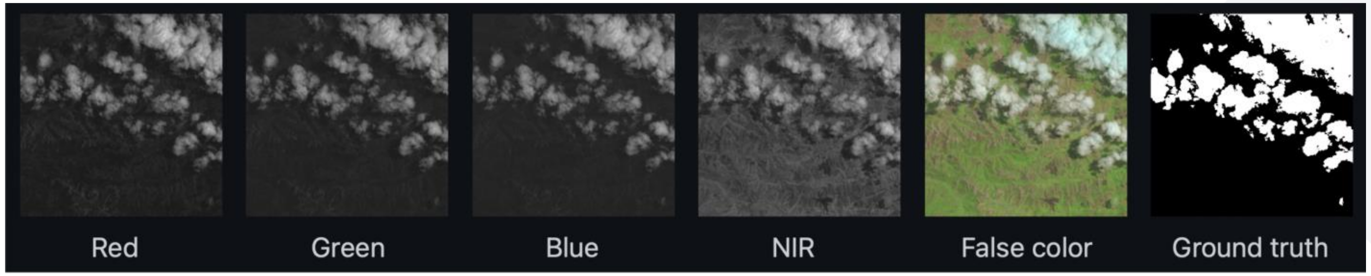
Project Use Cases

- 3 Project Use cases were implemented
  - High degree of integration
  - Distributed over 8 partitions executed together
- OHB/DLR Use Case - #UC1
  - Hardware interlocking – ILSWA, ILSWB
    - Protect against 2 types of wrong software behaviour
  - Implemented interlocks at software level instead of hardware
    - Reduced cost
  - Instrument Control Software
  - Implemented AI Based FDIR
  - Housekeeping data from ENMAP



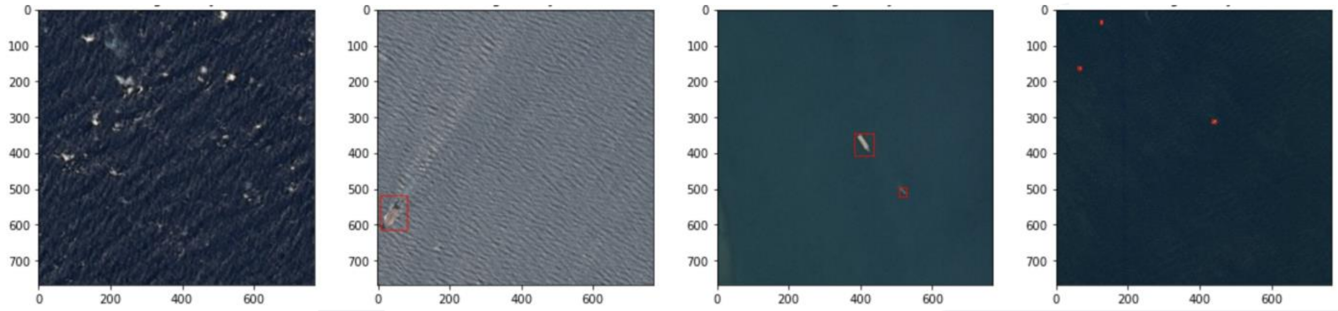
Project Use Cases

- 2 BSC provided use cases based on ESA's OBPMark-ML Open Source Benchmarking suite
- Cloud screening



4 Channels RGB/NIR mapped to binary mask (cloud/no cloud)

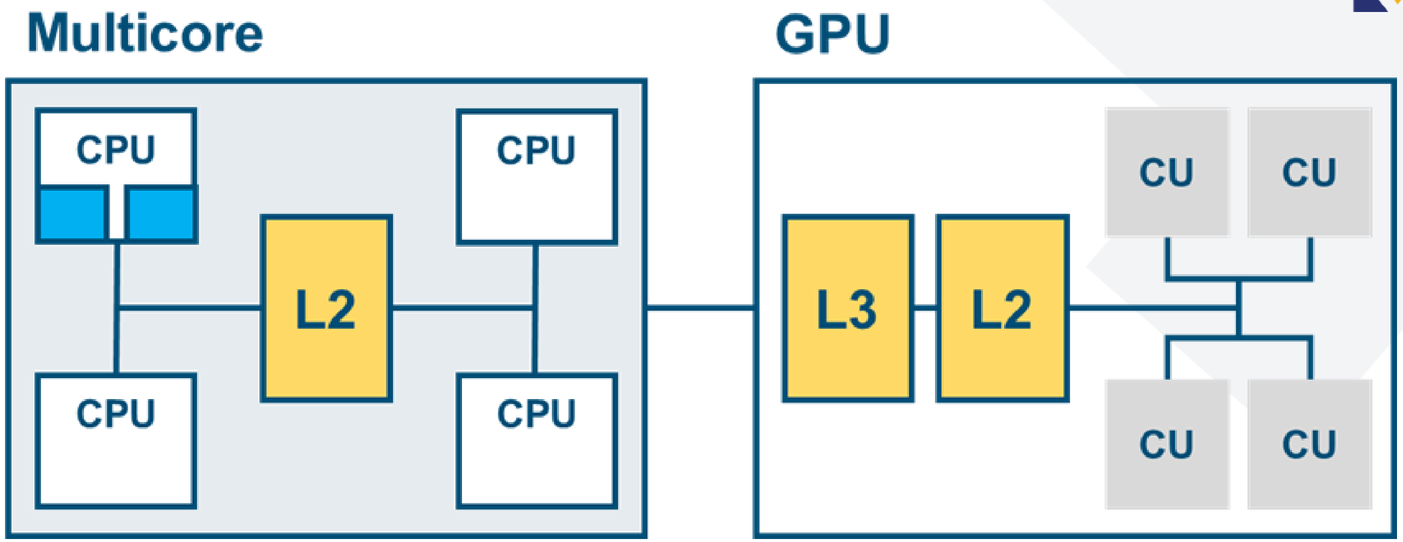
- Ship Detection



- Accelerated on the SPARROW and GPU

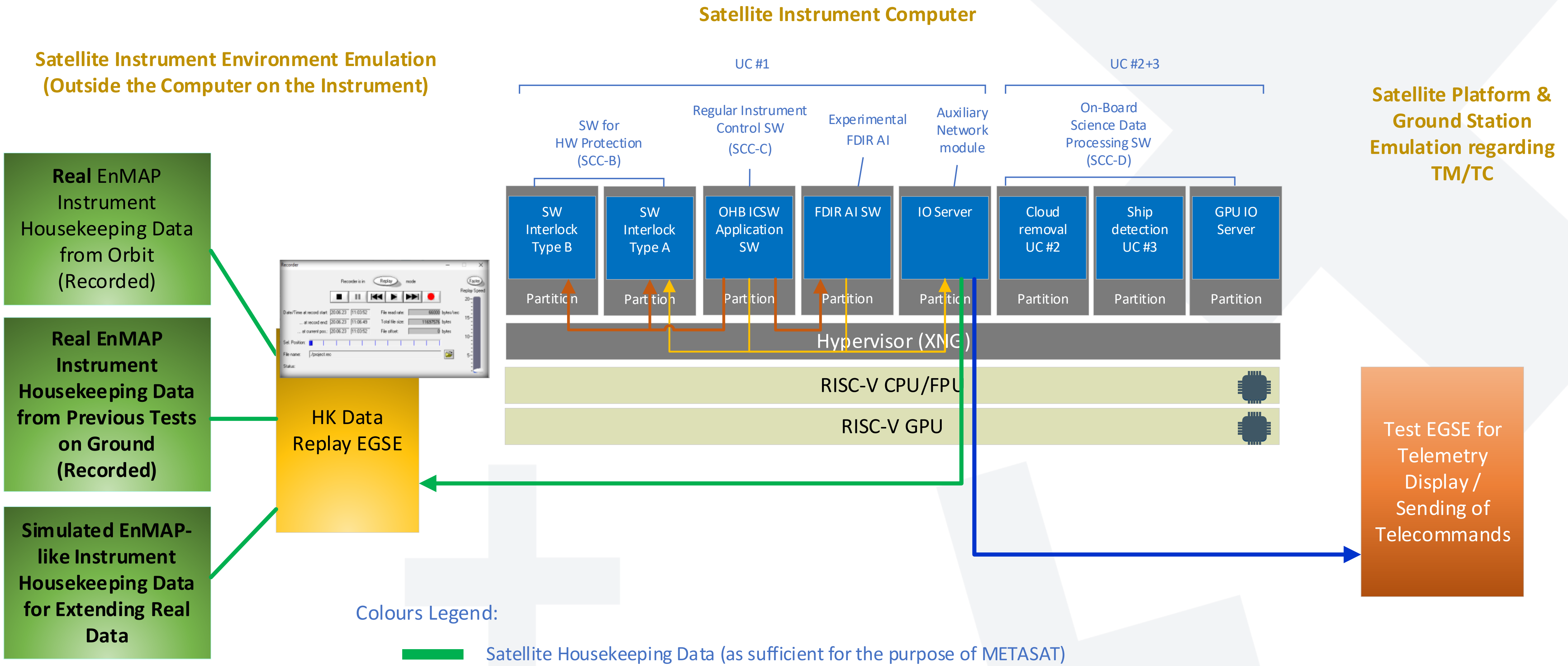
The METASAT Platform Overview

SPARROW SIMD Unit



- Qualifiable Software Stack: accelerators can be used from bare metal or RTEMS SMP
- Mixed-criticality: TSP support
- Added support in Model-based design tools
- Standard-based Digital Twin framework

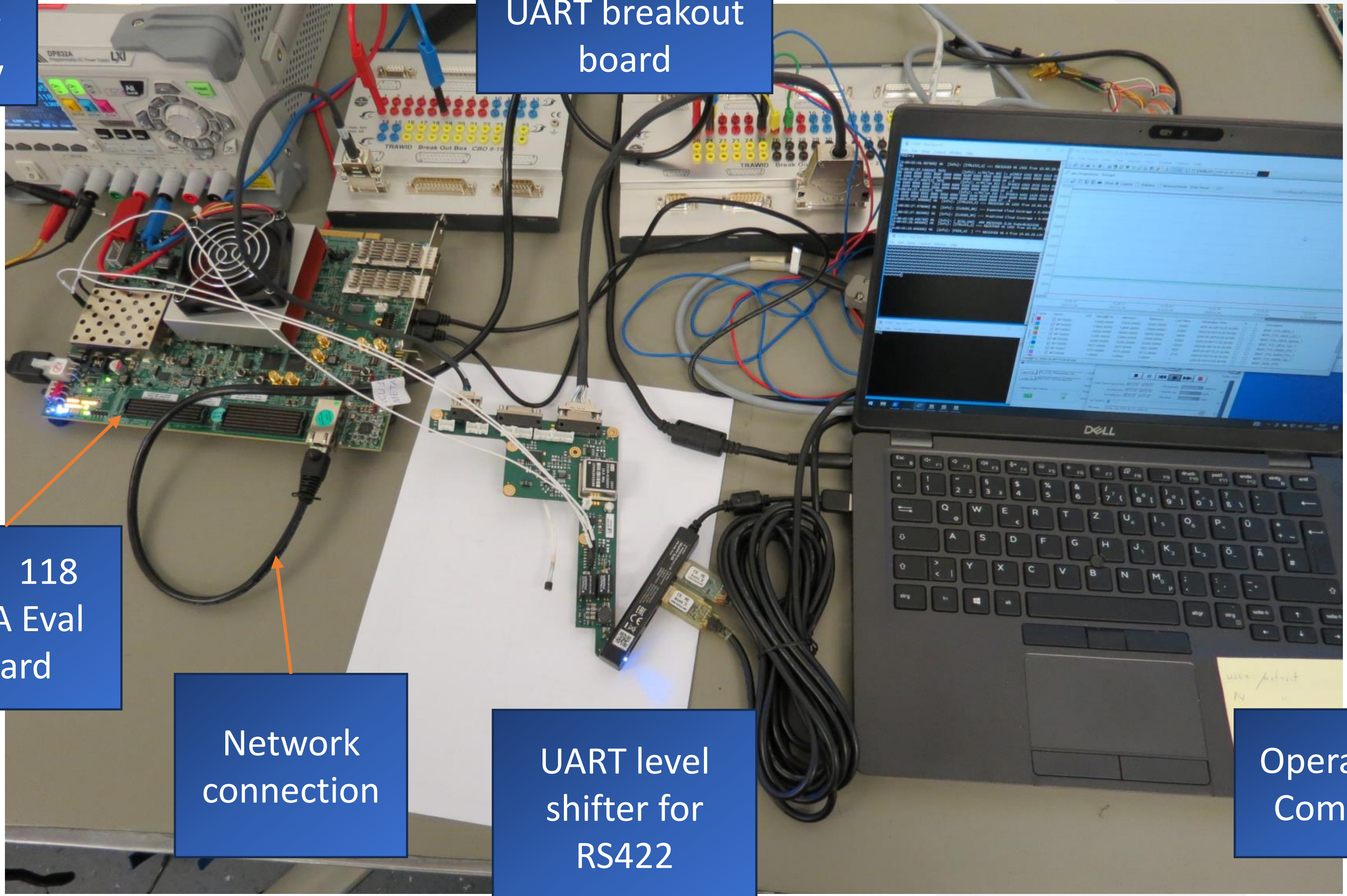
METASAT Use Cases Architecture



Colours Legend:

- Satellite Housekeeping Data (as sufficient for the purpose of METASAT)
- Telemetry and Telecommanding for controlling and testing the Use Case
- Inter-partition communication for commanding and status/data request
- Inter-partition communication for IP network

METASAT Platform Laboratory Setup

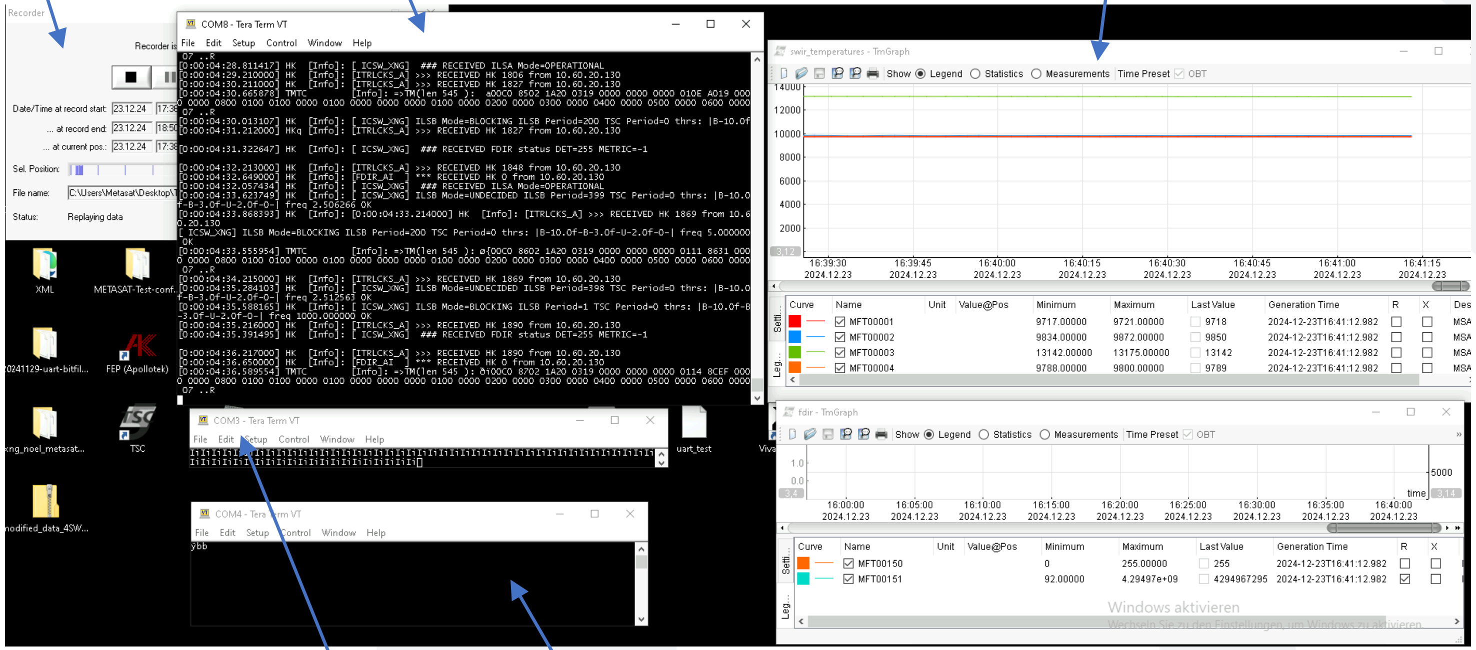


METASAT Integrated Prototype

Replayer

XNG console
For all partitions, including cloud removal and ship detection

TSC ground parameter plot



Integrated METASAT use cases running in parallel

ICSW hardware output

Interlock software B output

METASAT Demo Technologies

Hardware Platform

- Multicore NOEL-V
- SPARROW AI accelerator
- Vortex GPU
- GRETH ethernet controller
- 2 UARTS: emulation of controlled devices through I/O

- Fully functional FPGA prototype on AMD/Xilinx VCU118
- Fully functional Digital Twin

METASAT Demo Technologies

Hypervisor – XtratuM/NG

- Time and Space Partitioning
- Inter-partition communication
- Multicore Support
- METASAT Hardware and Virtual Platform support
- Resource Virtualisation
  - vCPU , including SPARROW
  - Ethernet – I/O Server
  - GPU I/O Server
- Resource Access Control
  - Memory area Write Access Control
  - New feature implemented in METASAT for the interlock type A use case

METASAT Demo Technologies

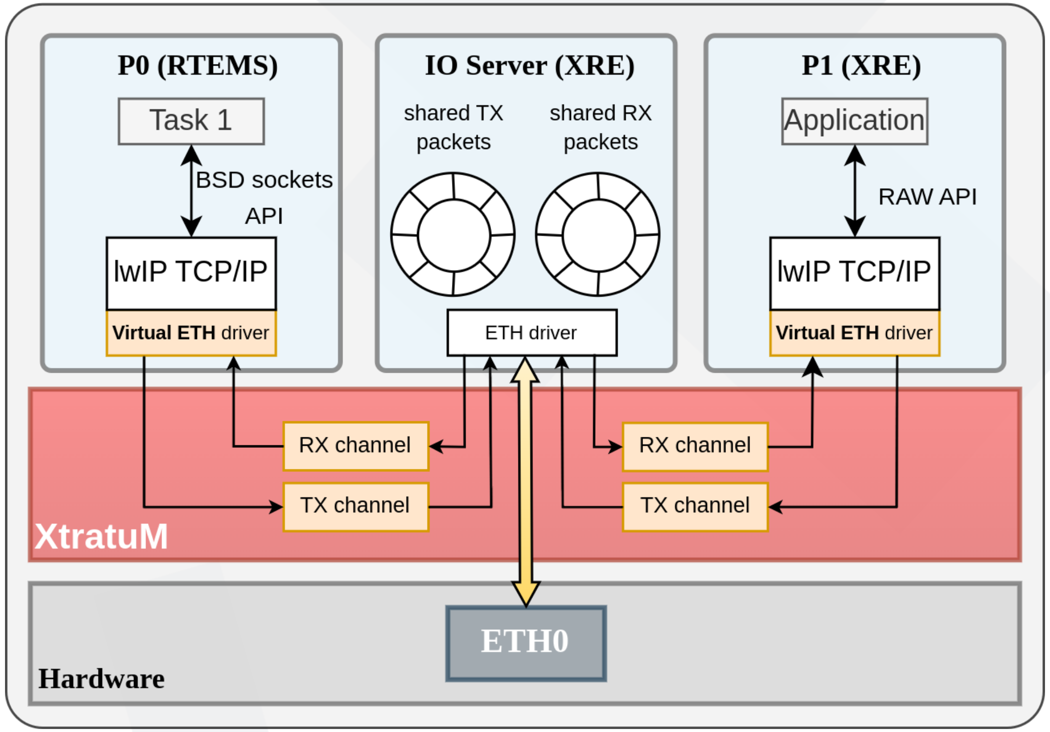
Memory area Write Access Control

- XtratuM/NG Hypervisor extension conceived and implemented in METASAT
- Provide partitions with the capability of controlling the state of a **memory area write access** permission.
- New **configuration attributes** in the XtratuM Configuration File for memory areas.
- New Hypervisor **services** for **enabling**, **disabling** and **retrieving** a memory area write access status.
- Used by the **Software Interlocks**.

METASAT Demo Technologies

Ethernet I/O Server

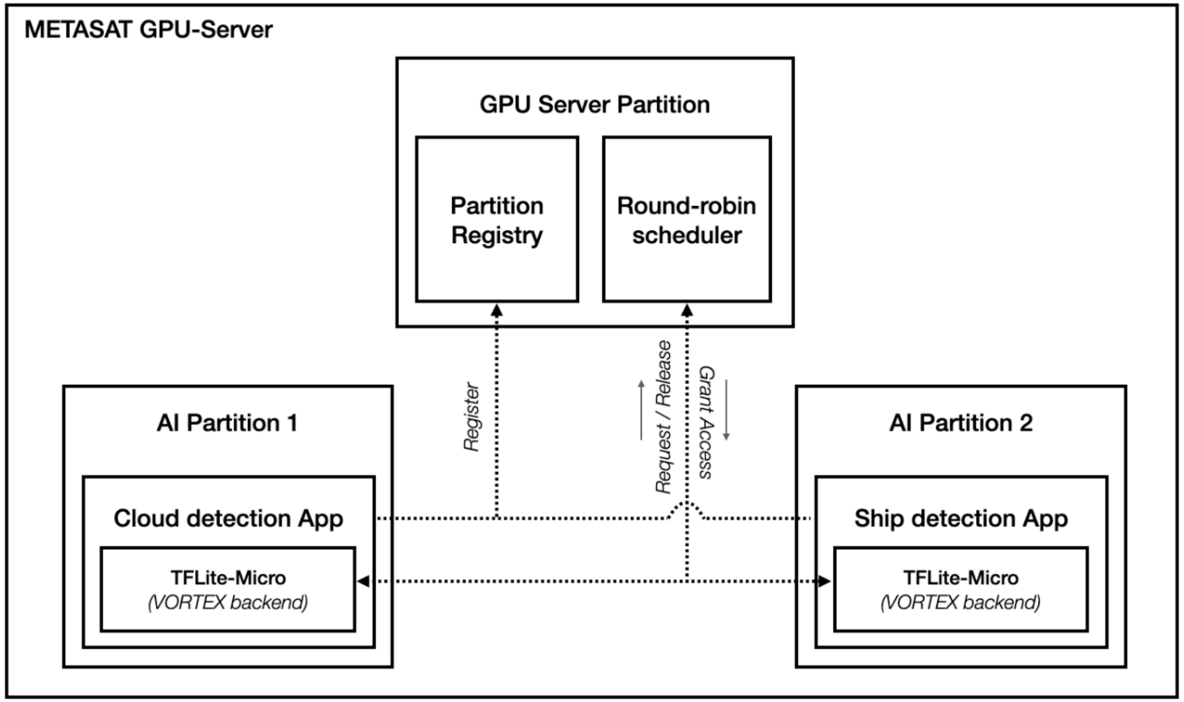
- Ethernet Virtualization at partition level.
- One additional partition for managing the Ethernet device, the I/O Server Partition.
- METASAT Platform Ethernet support (GRETH).
- XtratuM/NG Queuing ports for Ethernet frames transport.
- Virtual Ethernet Device Drivers for RTEMS and XRE partitions using the lwIP TCP/IP Stack.



METASAT Demo Technologies

GPU Server

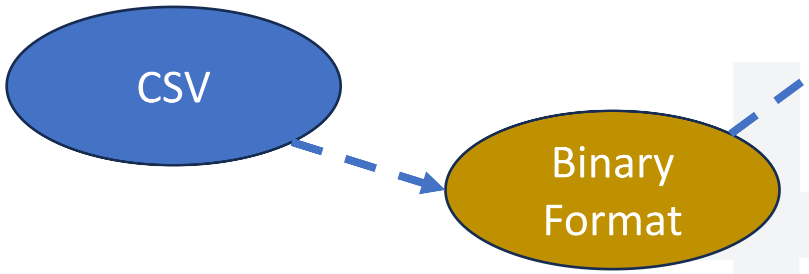
- Allows Time Sharing of the GPU
- Each Partition gets exclusive access to the GPU
- Permission request, get GPU lock, release

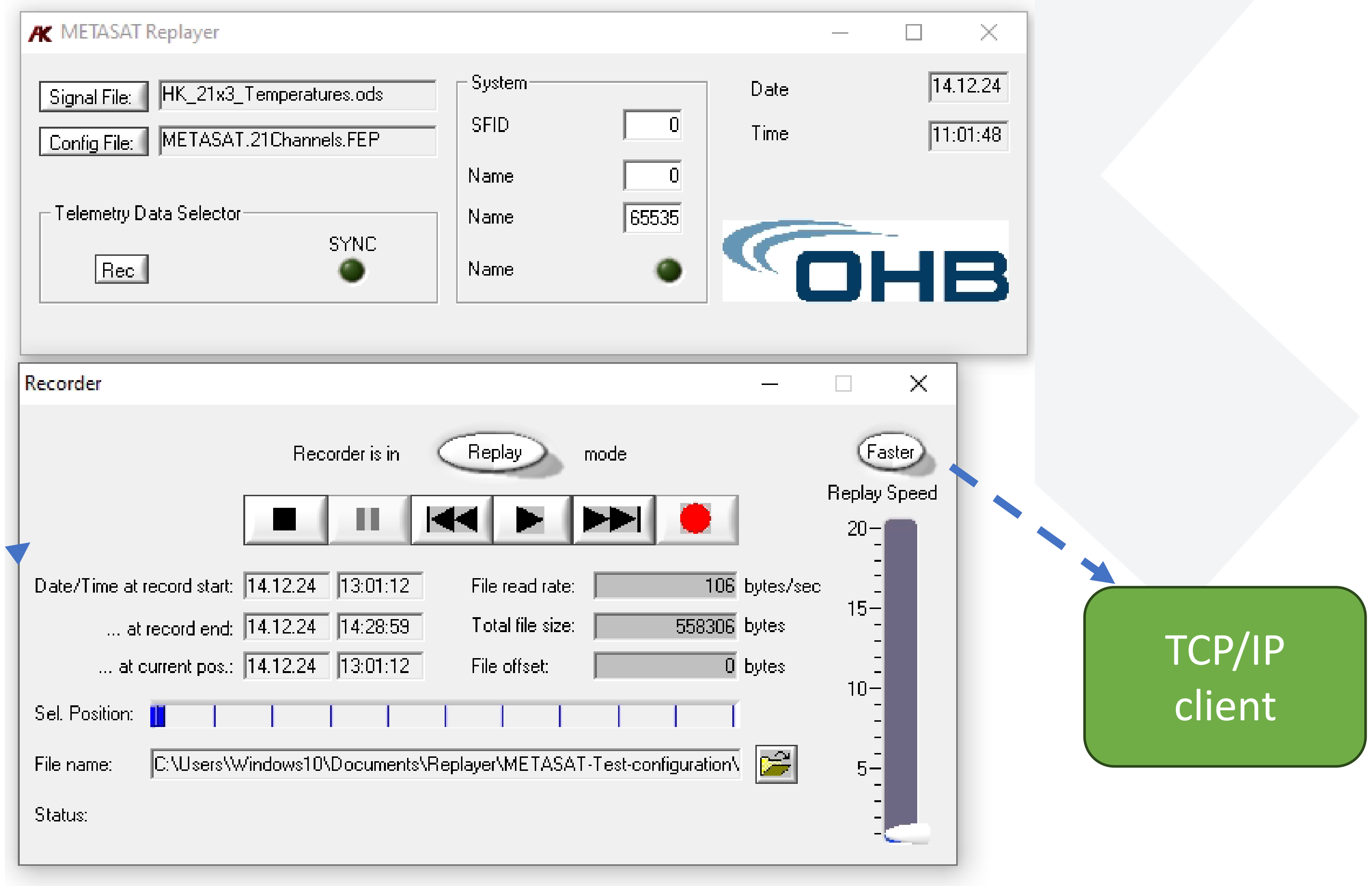


METASAT Demo Technologies

Replayer

- Canned housekeeping data
- TCP/IP server on different ports
- Re-use of test software
    - Customizable for ports and packet format
- Input in CSV format
- Manual speed settings

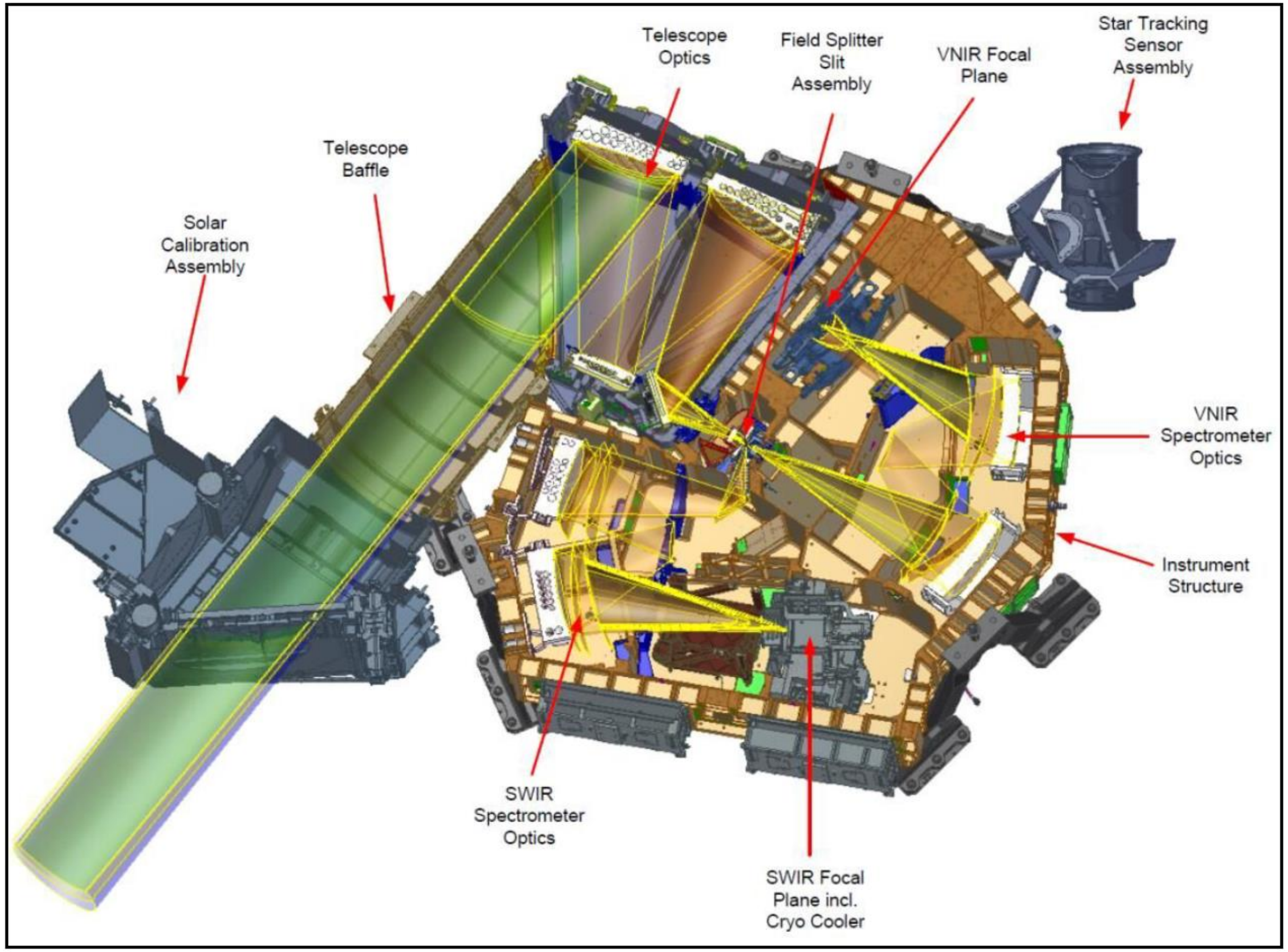




METASAT Demo Technologies

House Keeping Data

- EnMAP housekeeping data
  - Provided by DLR RFA
  - Data from real satellite operating in space
- Useful parameters: temperatures of
  - SWIR and VNIR camera
  - Calibration units
  - Cooling devices
- Using in total 21 parameters
  - 13 SWIR
  - 8 VNIR



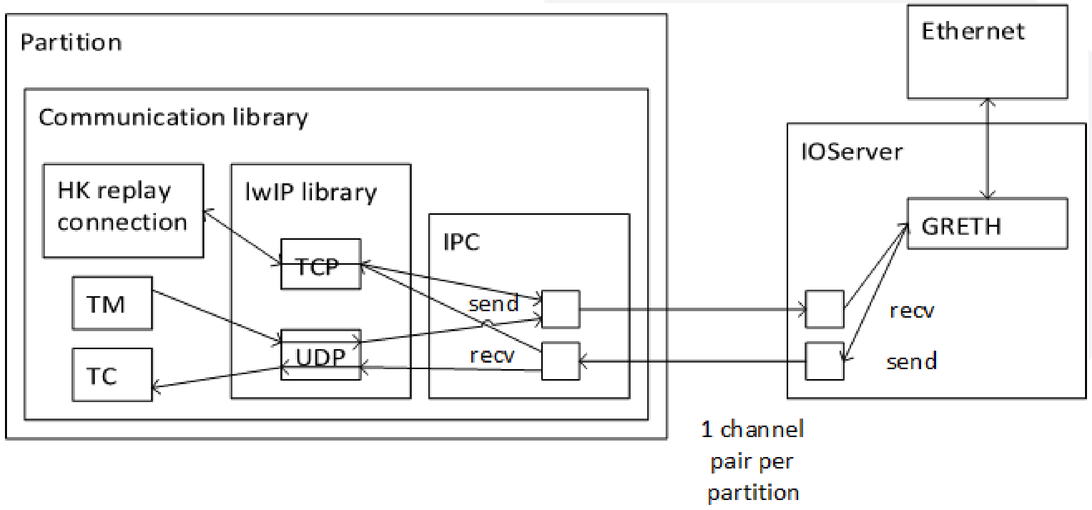
METASAT Demo Technologies

Communication Library

- Send/receive telecommands
- Send/receive telemetry

Underlying networking handled by library

- Inter-partition communication message channels to IOServer
- Pair of Send/Receive channels to IOServer for each partition that needs network access
- TCP/UDP IP networking layered on top of the lwIP library
- IOServer (fentISS) uses GRLIB GRETH driver to passthrough packets from IPC messages to Ethernet HW



METASAT Demo Technologies

taste

METASAT

Model Based Design for RISC-V and Multicores

- Added TASTE support for:
  - RISC-V/NOEL-V
  - SPARROW and OpenMP code generation configuration for RTEMS
  - XtratuM inter-partition communication modeling and multicore partition configuration generation

Changed to 2 cores by software engineer

Generated code

User code

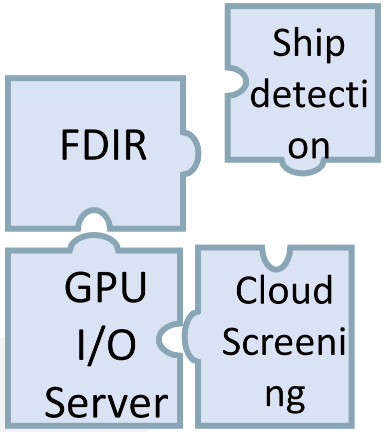
```c
#pragma once

#include "dataview-uniq.h"

#define USER_NUM_OPENMP_THREADS 2  // Here the designer set the number of threads for his application

#if defined( RTEMS_SMP )
    #include "omp.h"
    void __attribute__((__constructor__(1000))) config_libgomp(void)
    {
        setenv("OMP_DISPLAY_ENV", "VERBOSE", 1);
        setenv("GOMP_SPINCOUNT", "30000", 1);
        setenv("GOMP_DEBUG", "1", 1);
        setenv("OMP_NUM_THREADS", " USER_NUM_OPENMP_THREADS ", 1);
    }
#endif

#include "sparrow.h"  // include sparrow instruction support

#ifdef __cplusplus
extern "C" {
#endif


void function_1_startup(void){
  /* OpenMP test: Print from two different cores */

  #pragma omp parallel
  {
     printf("Hello World... from thread = %d on CPU %d\n",
                omp_get_thread_num(), _SMP_Get_current_processor());
  }

  /* SPARROW test: Dot product of two int8x8_t arrays */

  // 64-bit packed integers (8 * int8 = 64 bits)
  int64_t a = 0x0807060504030201;  // [1, 2, 3, 4, 5, 6, 7, 8] packed
  int64_t b = 0x0807060504FD02FF;  // [-1, 2, -3, 4, -5, 6, -7, 8] packed
  int32_t result = 0;

  // Use sparrows dot_s8 function to calculate the dot product for int8 arrays of 8 elements
  result = dot_s8(a, b); // Expected result: 36

  // Print the result
  printf("Test dot_s8: (Expected: 36) Result: %ld\n", result);

};

/* Required interfaces */

#ifdef __cplusplus
}
#endif
```

Generated code

METASAT Demo Technologies

Model-Based Design for Accelerators

- TensorFlow Micro Support for SPARROW and Vortex GPU
- TensorFlow Lite code generation from MATLAB/Simulink to TensorFlowMicro
- Accelerated layers in SPARROW intrinsics and Vortex
- Bare Metal, RTEMS and XRE
- Seamless integration with the GPU Server
  - No changes in the integrated partitions
  - Plug and Play



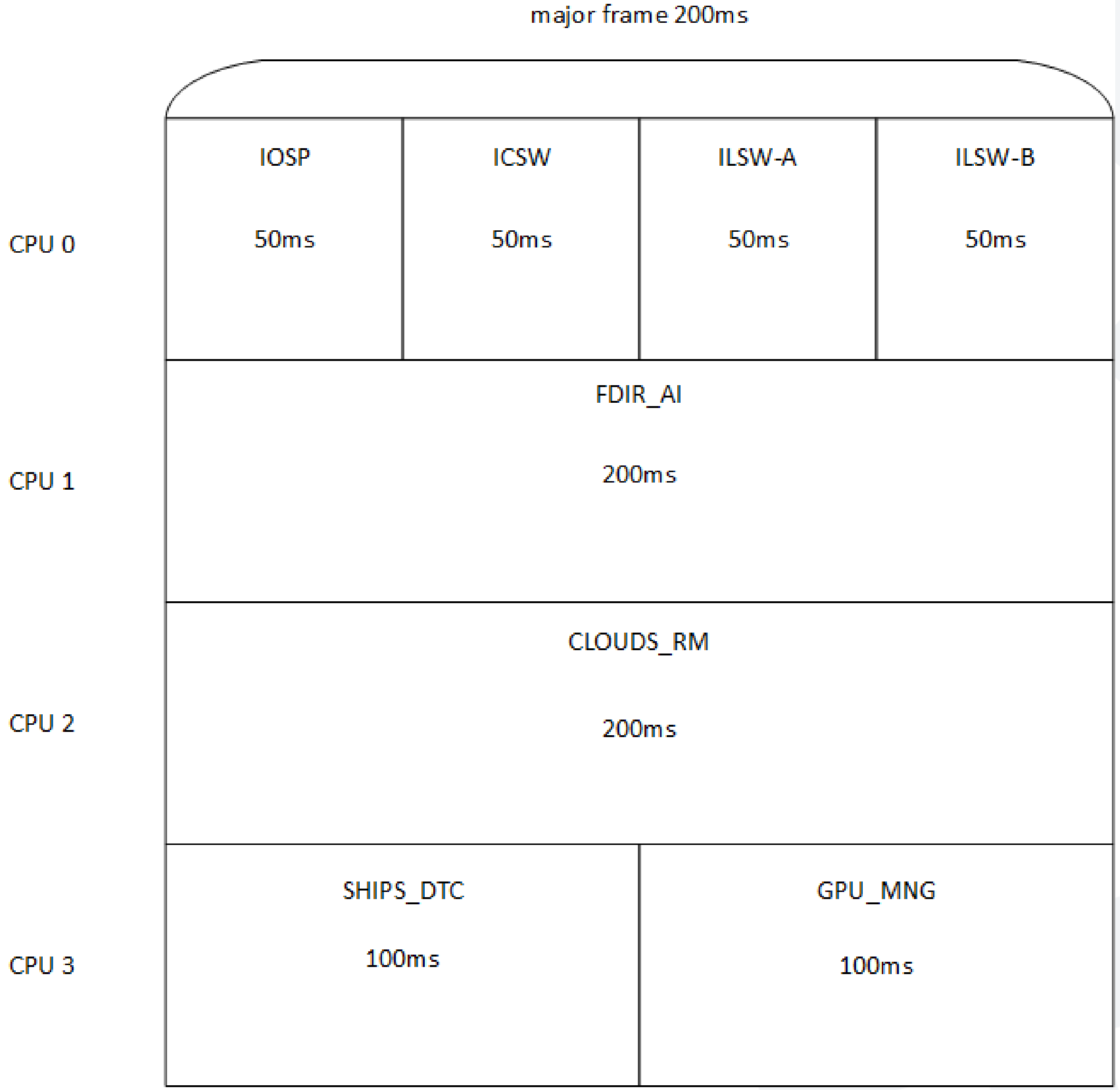
Integration Procedure Overview

Integration Procedure

METASAT achieved a high degree of integration with all use cases running in parallel on the FPGA at OHB premises

- Each use case was developed in isolation
  - Validated on QEMU, Digital Twin and FPGA
- Some use case parts were developed by different partners
  - Integration at partition level performed by the use case provider
- Partial integration tests performed with a mock-up of the OHB use case from different partners
- Final integration of all partitions performed by OHB
  - Modularity and Plug and Play functionality demonstrated
  - Both for QEMU and FPGA

Multicore Schedule of the METASAT Demo



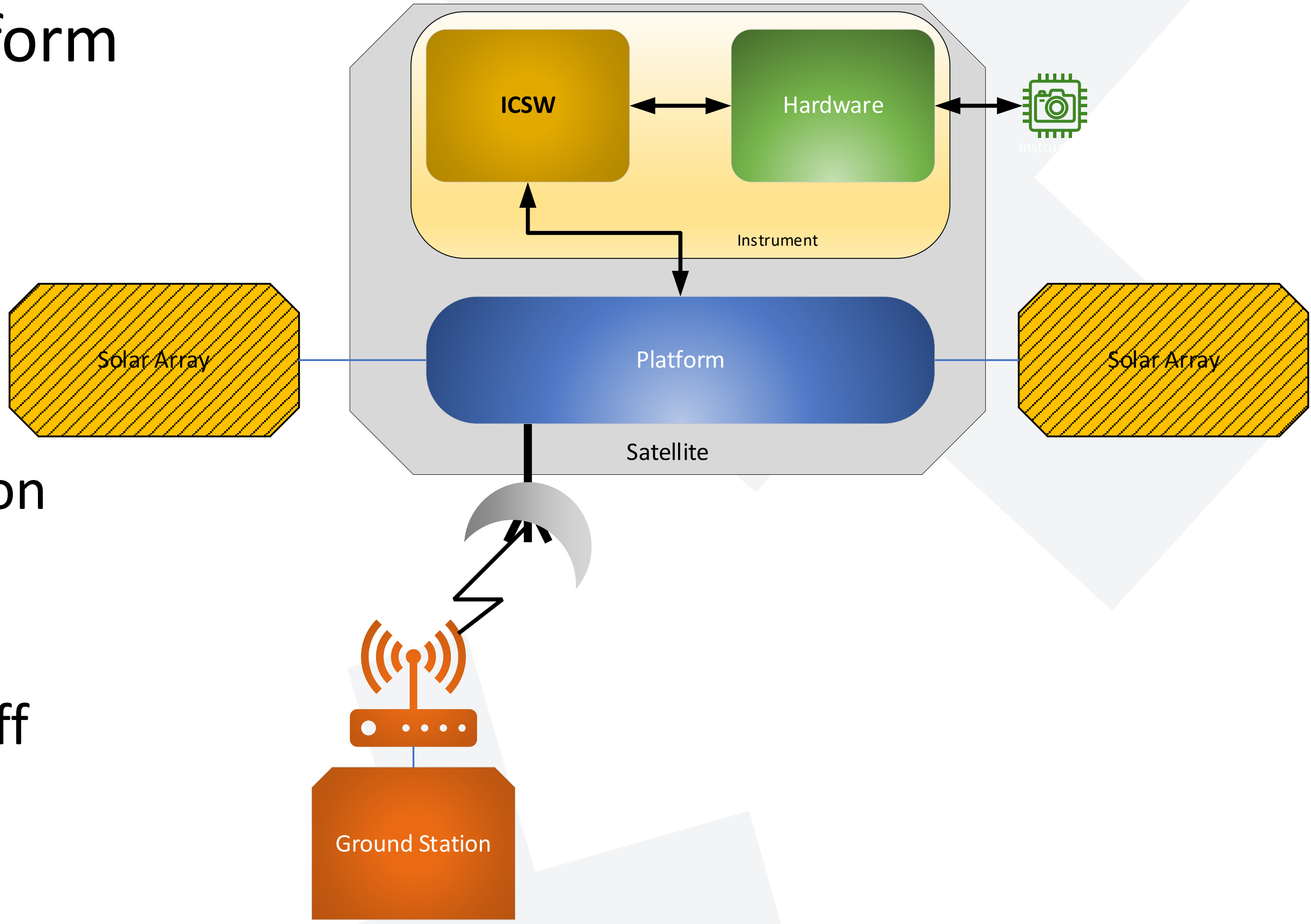
Resource Table of the METASAT Demo

SYSTEM	PARTITIONS	ENTRY POINT	SIZE	PART TYPE
XNG	HYPERVISOR	0x00100000		
SBI	HYPERVISOR	0x00000000		
XCF	HYPERVISOR	0x00500000		
NETWORK	IOSP_XRE	0x02000000	6MB	XRE
	IOSP_XRE			
	ICSW_METASAT	0x3000000	256M	RTEMS_XNG
	INTERLOCKS_A	0x13000000	16MB	XRE
	INTERLOCKS_B	0x14000000	16MB	XRE
	FDIR_AI	0x15000000	32MB	XRE
	SHIPS_DTC	0x17000000	256MB	XRE
	CLOUDS_RM	0x27000000	128MB	XRE
	GPU_MNG	0x30000000	16MB	XRE

Instrument Control Use Case

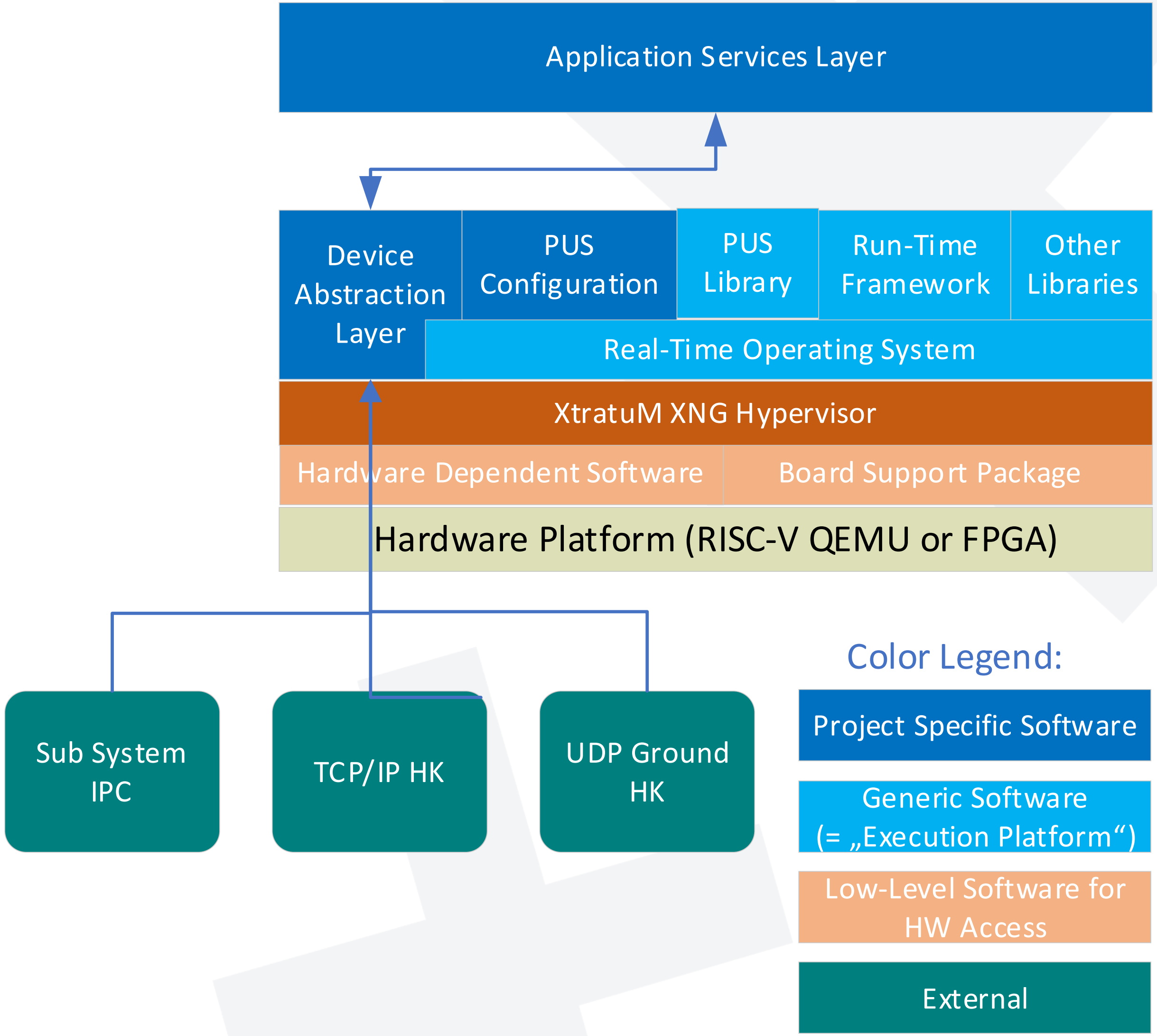
Demonstration of an Instrument Control Unit

- Satellite architecture with platform and instrument payload
- Instrument data processing:
  - Read housekeeping data
  - Provide to ground via telemetry
  - Steer the hardware
  - Monitor the operational condition
  - Commanding via telecommands
- Interlock and failure detection
  - Put instrument to safe state in off nominal case
  - Alert for potential off-nominal condition



ICSW Architecture

- Instrument Control Software (ICSW) fork from OHB satellite project
- Ported to RISC-V
- TC/TM in PUS (Packet Utilisation Services)
- IP network integration
- Inter partition communication to
  - interlock software
  - FDIR



Interlocks Use Case

Interlock Purpose

- In the space domain: high demands on the overall system dependability and safety
- One possible commonly agreed approach for handling this are architectures with redundancies
- Example for a dissimilar redundancy: the regular control and processing instance (e.g. the control SW of a satellite instrument) shall be monitored
  - by a strictly independent module that acts as an interlock …
  - … and steps in when a critical condition is met

- Such an interlock module being implemented in hardware (like in the past at OHB) has several limitations
  - A HW solution typically has a rather simple failure detection functionality
  - Adds extra resource budgets (extent of electronics and/or FPGA size grows)
  - Adaptations during the project may generate high costs and may pose a risk for the project schedule
  - The solution is rather inflexible when the operational conditions change in space
  - Adds high qualification effort to the hardware for dependability and safety

(refer ADCSS 2024)

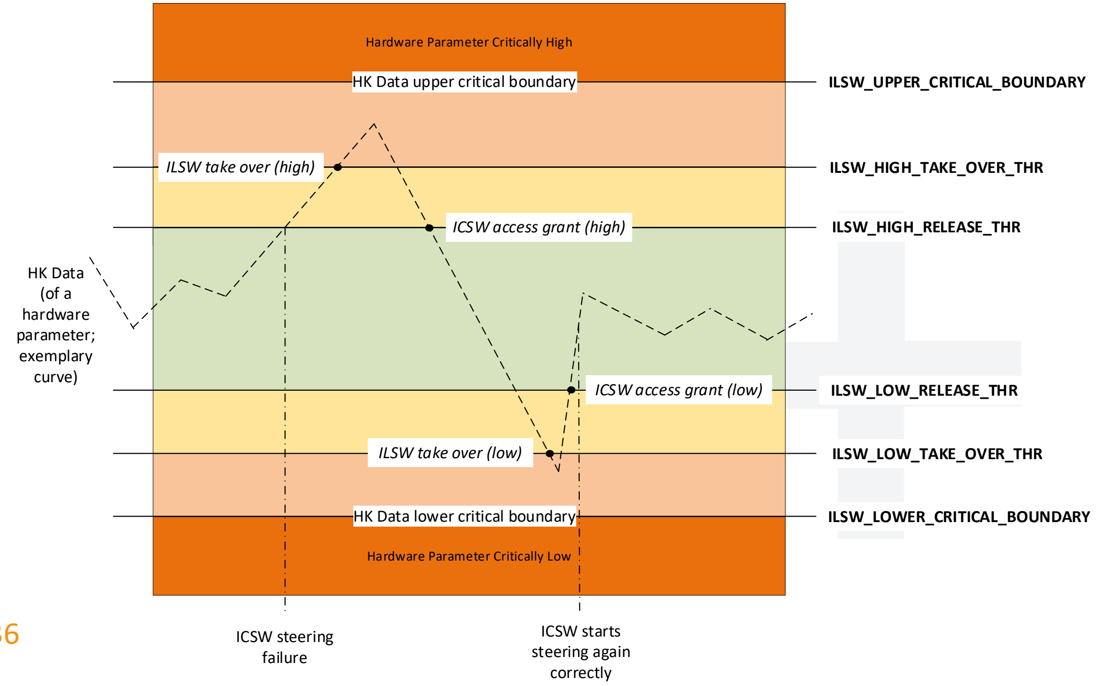
Software Interlocks

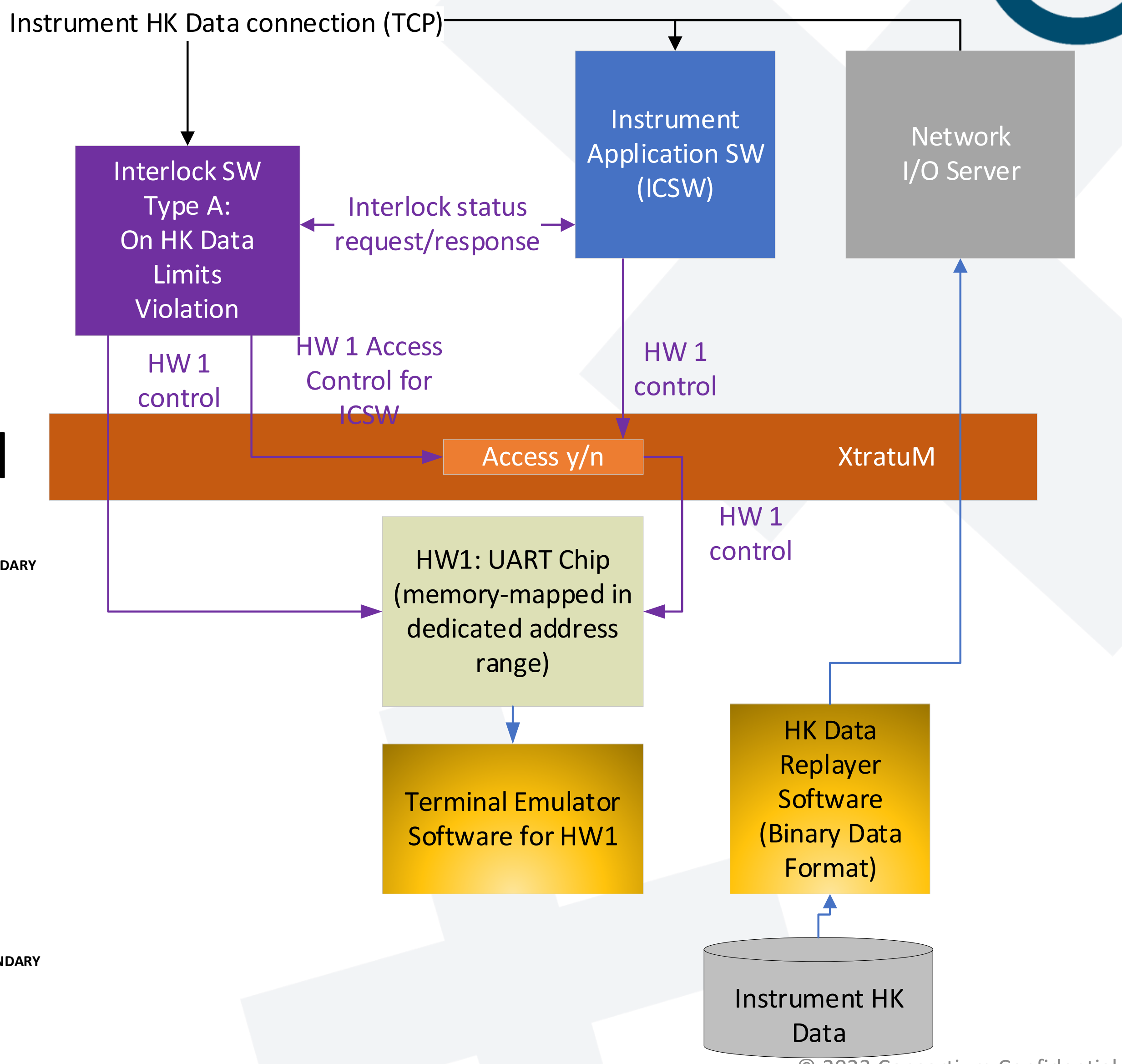
- A software solution for interlocks would not have these disadvantages. Idea / scenario:
- Main application software with sophisticated system functionality
  - Complex; multi-tasking; executing on a real-time operating system
  - Interest of OHB to qualify it only to a medium level criticality for cost reasons
  - E.g. ECSS software criticality category "C"
- A second software module ("Software Interlock") monitoring the application software and performing interlock functionality
  - Software Interlock: small, simple, bare-metal (i.e. no OS)
  - Be a compensating provision to the main application software
  - Has to be qualified one software criticality category higher than the ASW (i.e. SCC-B)
- Overall: more flexibility, and significantly lower V&V project effort assessed than with the complete application software being qualified as SCC-B

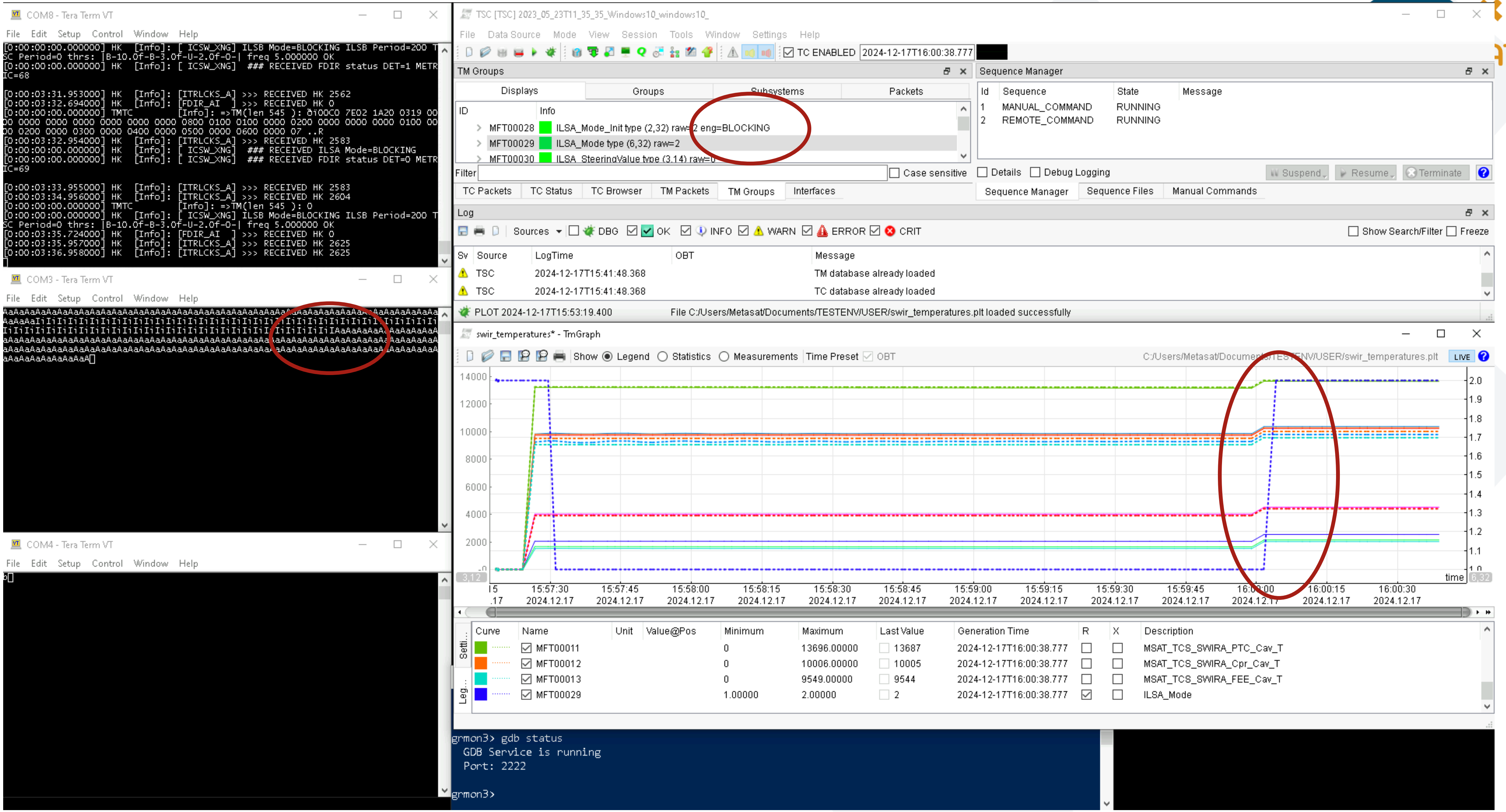
(refer ADCSS 2024)

Interlock A

- ICSW with direct HW access
- Parameters are monitored (temperature)
- Access can be blocked by hypervisor based on threshold

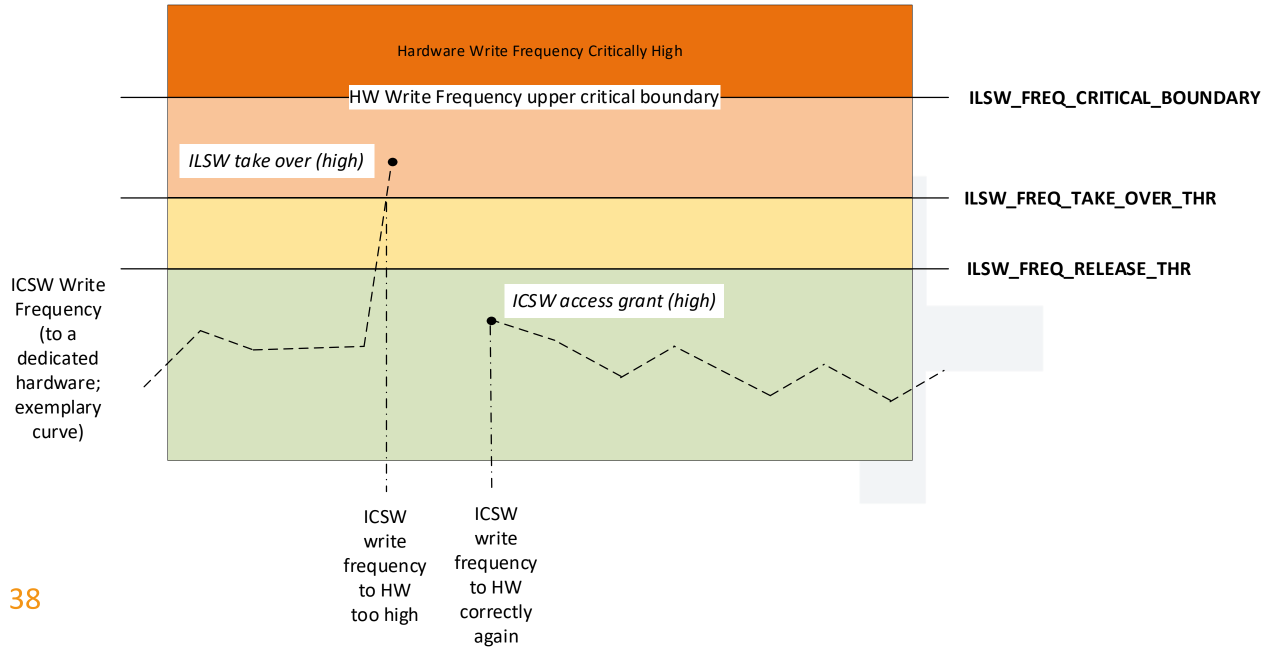


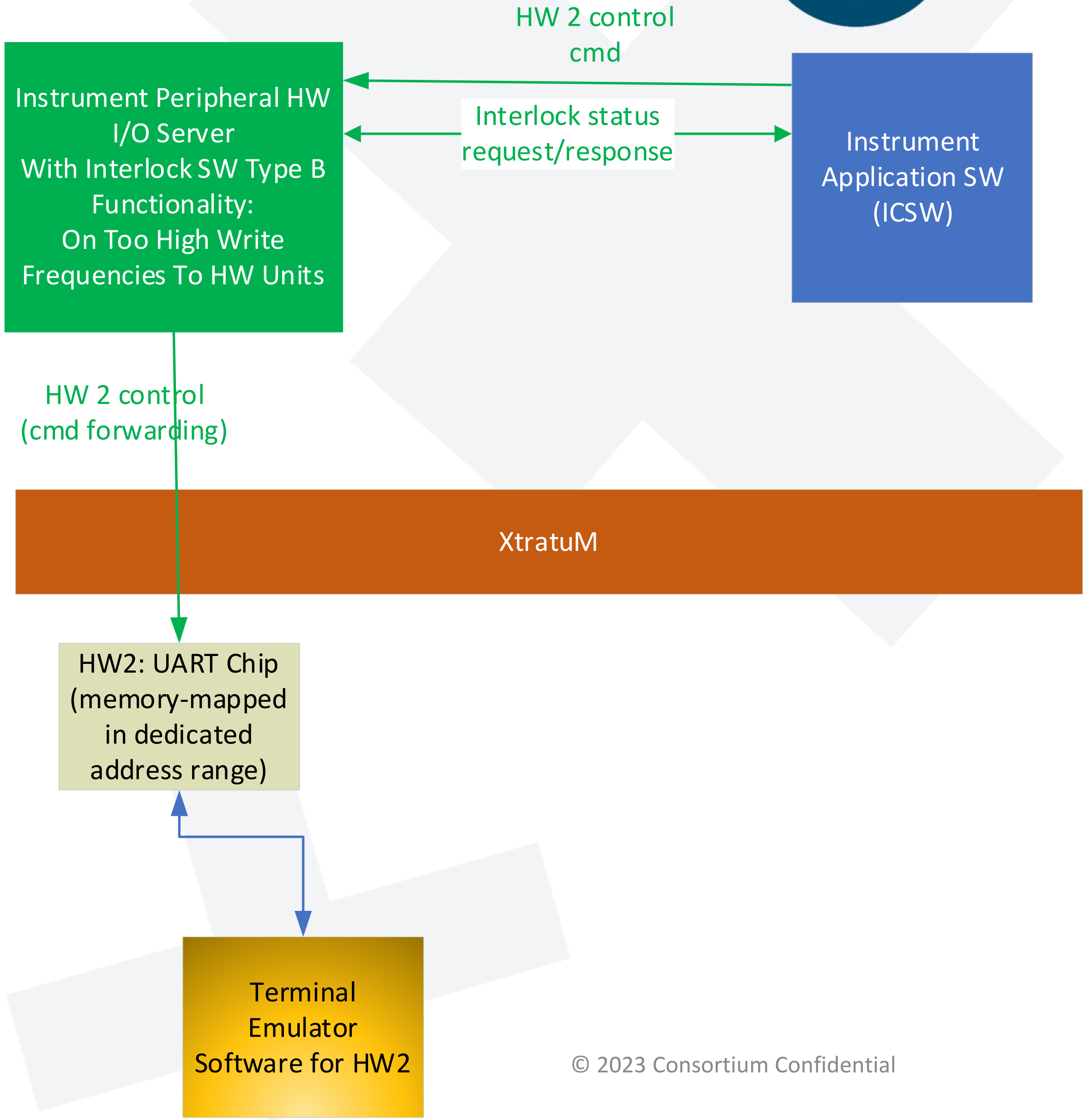



COM8 - Tera Term VT
File Edit Setup Control Window Help
[0:00:00:00.000000] HK [Info]: [ICSW_XNG] ILSB Mode=BLOCKING ILSB Period=200 T
SC Period=0 thrs: [B-10.0f-B-3.0f-U-2.0f-0-| freq 5.000000 OK
[0:00:00:00.000000] HK [Info]: [ICSW_XNG] ### RECEIVED FDIR status DET=1 METR
IC=68
[0:00:03:31.953000] HK [Info]: [ITRLCKS_A] >>> RECEIVED HK 2562
[0:00:03:32.694000] HK [Info]: [FDIR_AI] >>> RECEIVED HK 0
[0:00:03:32.954000] HK [Info]: [ITRLCKS_A] >>> RECEIVED HK 2583
[0:00:00:00.000000] HK [Info]: [ICSW_XNG] ### RECEIVED ILSA Mode=BLOCKING
[0:00:00:00.000000] HK [Info]: [ICSW_XNG] ### RECEIVED FDIR status DET=0 METR
IC=69
[0:00:03:33.955000] HK [Info]: [ITRLCKS_A] >>> RECEIVED HK 2583
[0:00:03:34.956000] HK [Info]: [ITRLCKS_A] >>> RECEIVED HK 2604
[0:00:03:35.724000] HK [Info]: [FDIR_AI] >>> RECEIVED HK 0
[0:00:03:35.957000] HK [Info]: [ITRLCKS_A] >>> RECEIVED HK 2625
[0:00:03:36.958000] HK [Info]: [ITRLCKS_A] >>> RECEIVED HK 2625
COM3 - Tera Term VT
COM4 - Tera Term VT
TSC [TSC] 2023_05_23T11_35_35_Windows10_windows10_
File Data Source Mode View Session Tools Window Settings Help
TC ENABLED 2024-12-17T16:00:38.777
TM Groups
Displays Groups Subsystems Packets
ID Info
MFT00028 ILSA_Mode_Init type (2,32) raw=2 eng=BLOCKING
MFT00029 ILSA_Mode type (6,32) raw=2
MFT00030 ILSA_SteeringValue type (3,14) raw=0
Filter Case sensitive
TC Packets TC Status TC Browser TM Packets TM Groups Interfaces
Sequence Manager
Id Sequence State Message
1 MANUAL_COMMAND RUNNING
2 REMOTE_COMMAND RUNNING
Details Debug Logging Suspend Resume Terminate
Sequence Manager Sequence Files Manual Commands
Log
Sources DBG OK INFO WARN ERROR CRIT Show Search/Filter Freeze
Sv Source LogTime OBT Message
TSC 2024-12-17T15:41:48.368 TM database already loaded
TSC 2024-12-17T15:41:48.368 TC database already loaded
PLOT 2024-12-17T15:53:19.400 File C:/Users/Metasat/Documents/TESTENV/USER/swir_temperatures.plt loaded successfully
swir_temperatures* - TmGraph
Show Legend Statistics Measurements Time Preset OBT
LIVE
time
Curve Name Unit Value@Pos Minimum Maximum Last Value Generation Time R X Description
MFT00011 0 13696.00000 13687 2024-12-17T16:00:38.777 MSAT_TCS_SWIRA_PTC_Cav_T
MFT00012 0 10006.00000 10005 2024-12-17T16:00:38.777 MSAT_TCS_SWIRA_Cpr_Cav_T
MFT00013 0 9549.00000 9544 2024-12-17T16:00:38.777 MSAT_TCS_SWIRA_FEE_Cav_T
MFT00029 1.00000 2.00000 2 2024-12-17T16:00:38.777 ILSA_Mode
grmon3> gdb status
GDB Service is running
Port: 2222
grmon3>

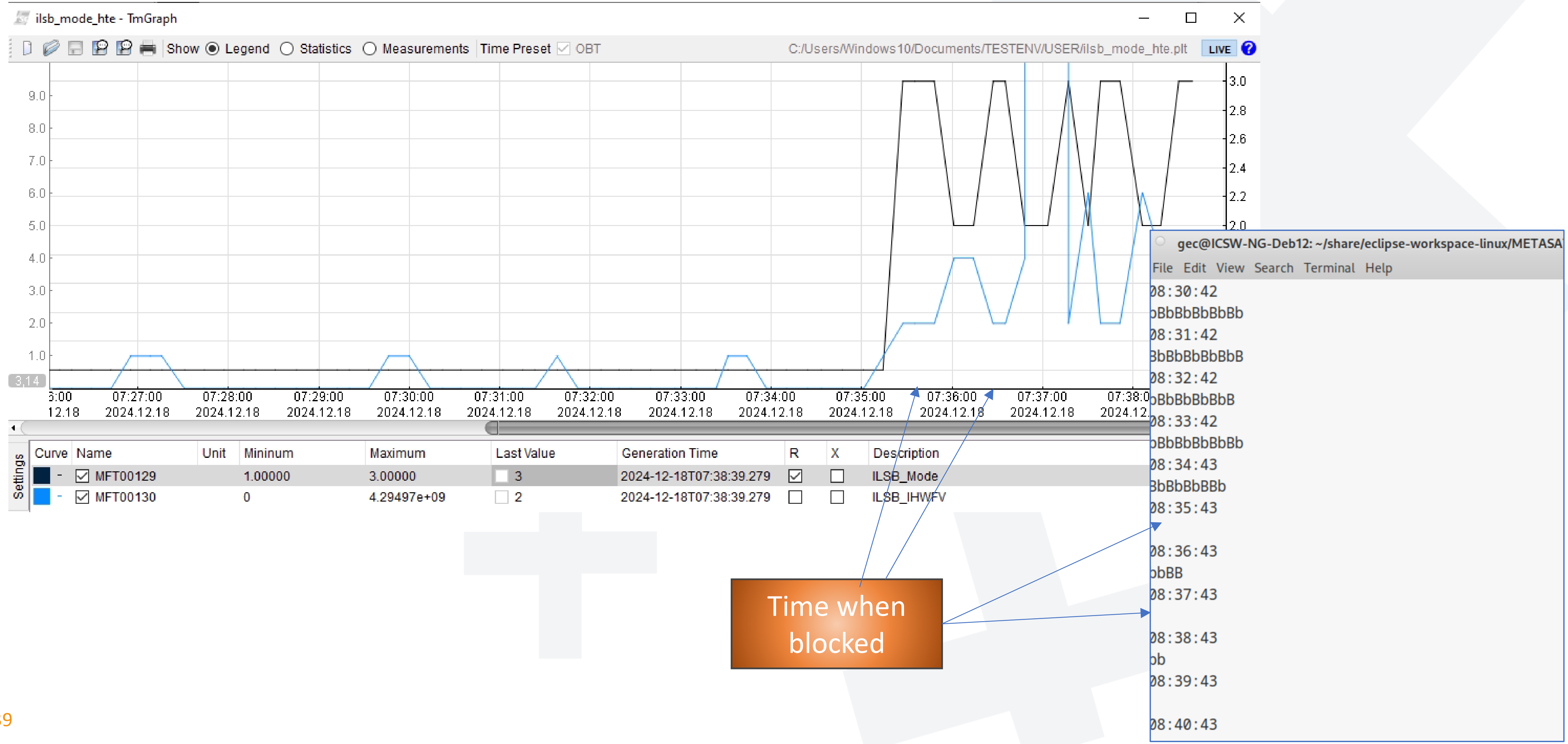
Interlock B

- Proxy by interlock software
- Full control over commands in interlock software
- Demonstrated for high write frequency block





Interlock B Test Result



FDIR AI Use Case

FDIR AI Purpose

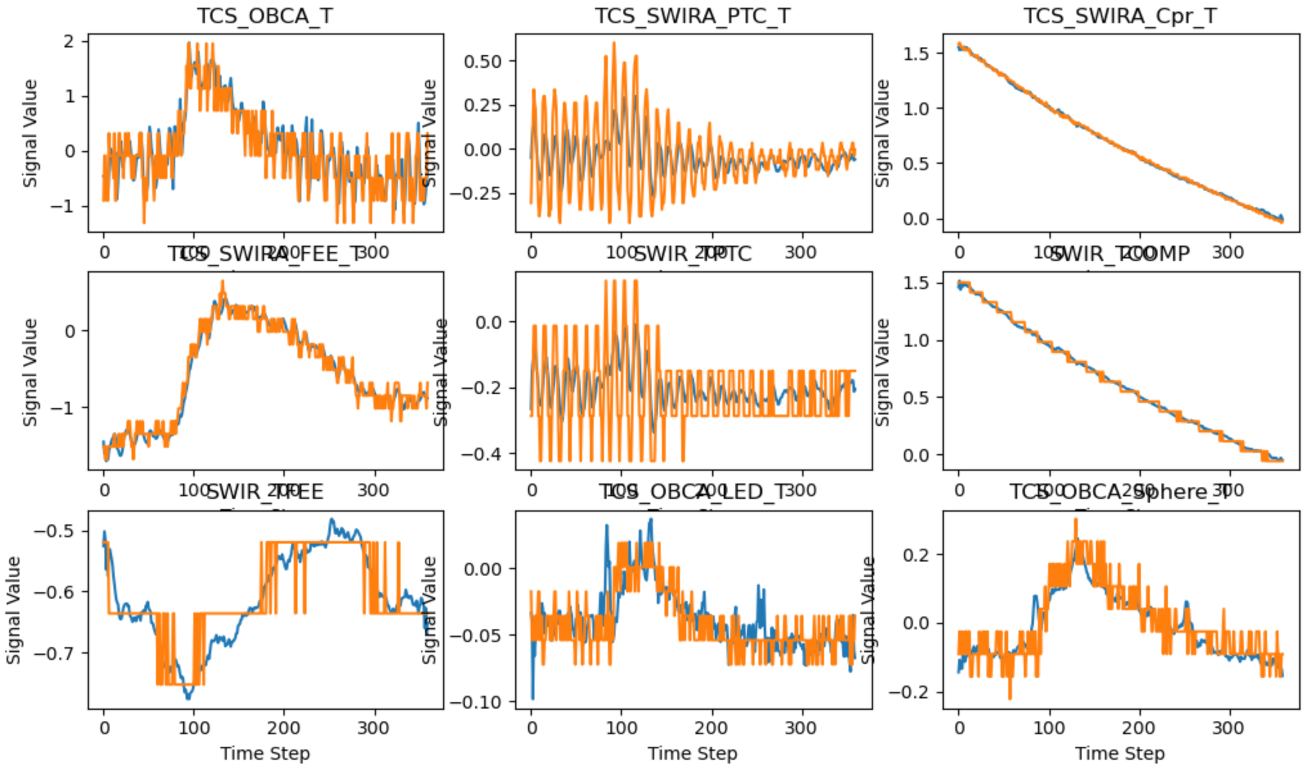
Conventional FDIR uses hard boundaries or human ground observation and intervention to classify anormal behavior of satellite systems

METASAT FDIR AI adds more sophisticated sensor observation to increase autonomy

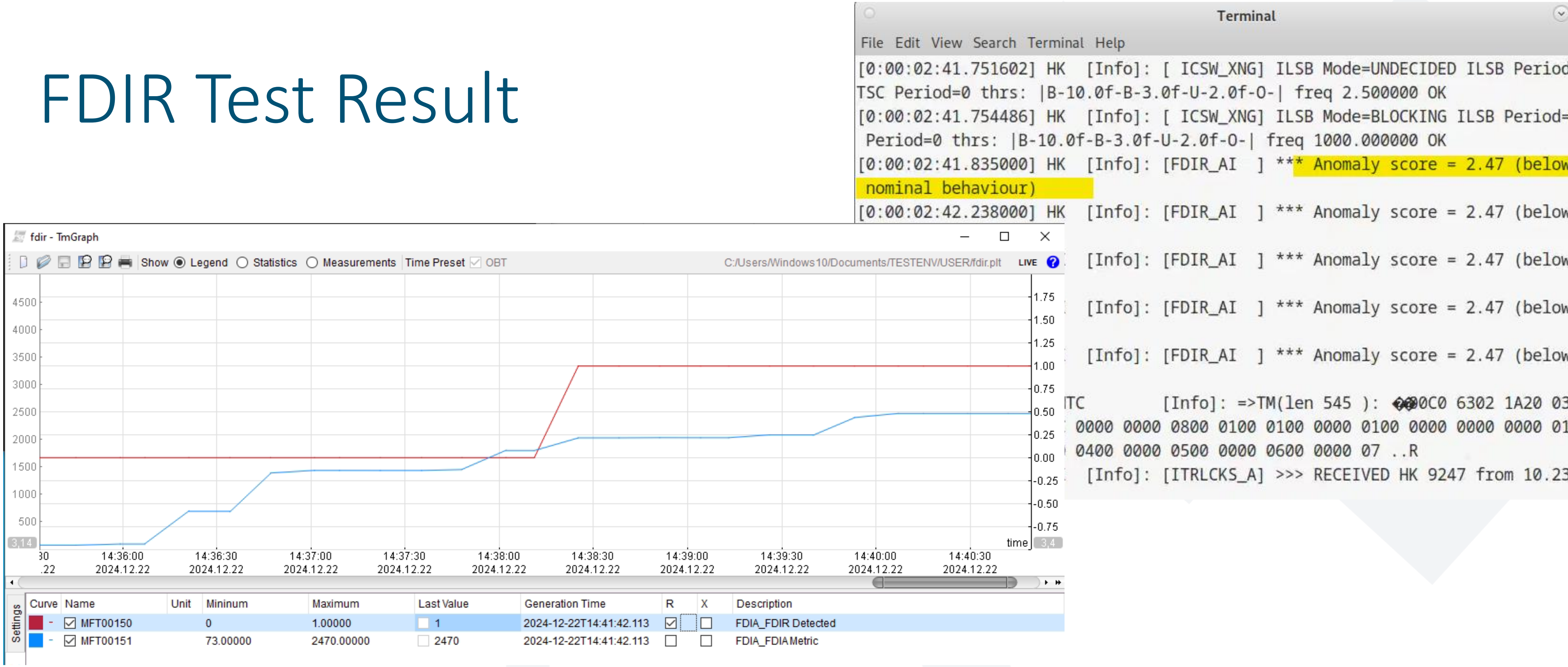
- Uses neural network based Autoencoder
- Predicts normal behavior over sensor readings of 1 hour
- Calculates Anomaly score to alert in case of failures

FDIR AI Design

Anomaly Score is the difference between measured (orange) and predicted (blue) curves



FDIR Test Result

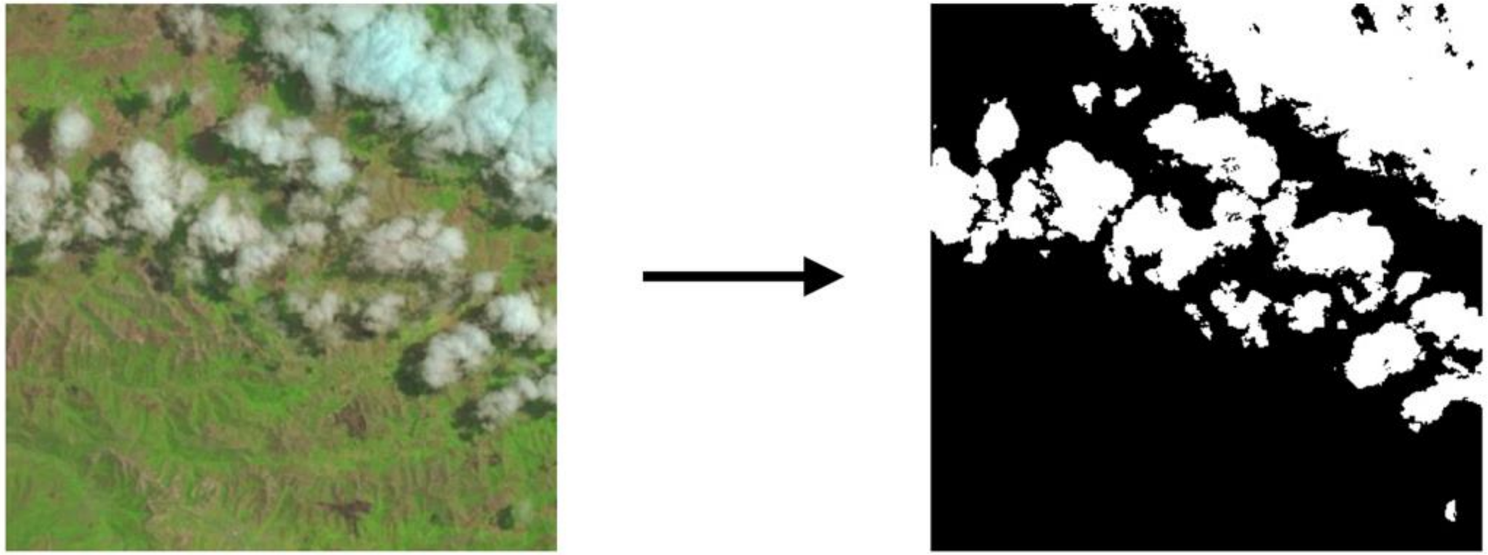


Cloud Screening Use Case

Cloud Screening

Identifying and classifying cloud-covered areas in satellite imagery

- UNet segmentation model trained on the Cloud95 dataset



4 Channel Input Image — Binary Segmentation Mask

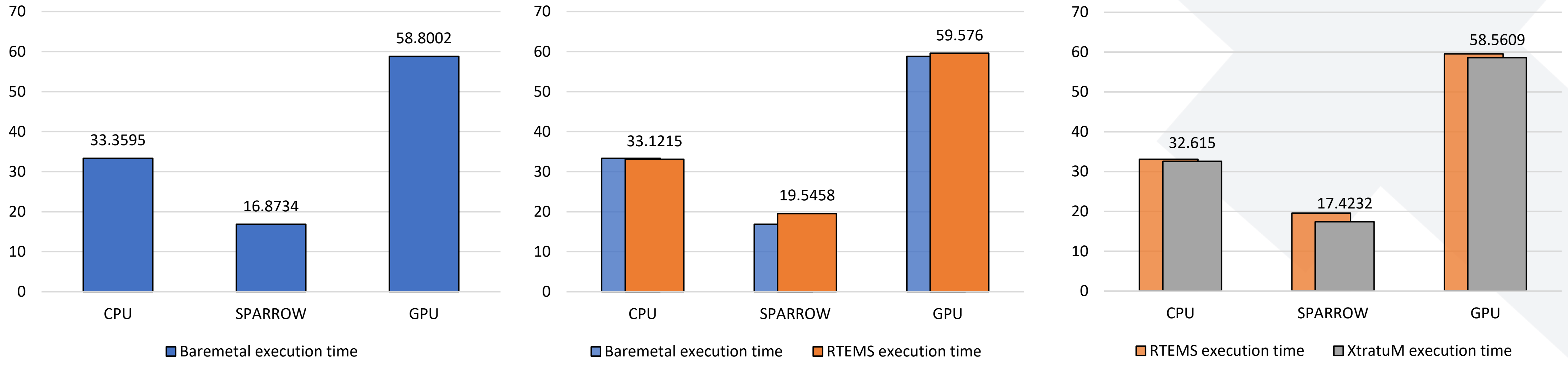
Cloud Screening

Integration with the rest of the system

- BSC provided 3 versions of the partition
  - CPU, SPARROW Accelerated, Vortex Accelerated
  - Each partition differ only on the TensorFlow Micro library backend
  - Integration at BSC with the mock-up version provided by OHB
  - Individual UC tests (or both UC#2 and UC#3) run at BSC
- Integration at OHB required only replacing cloud screening partition with the desired version of the partition
  - A GPU accelerated partition works seamlessly with GPU I/O server
  - No change required
  - Experiments with the fully integrated use cases run at OHB

Cloud Screening

- Evaluation of UC#2 in isolation – Baremetal, RTEMS and XtratuM



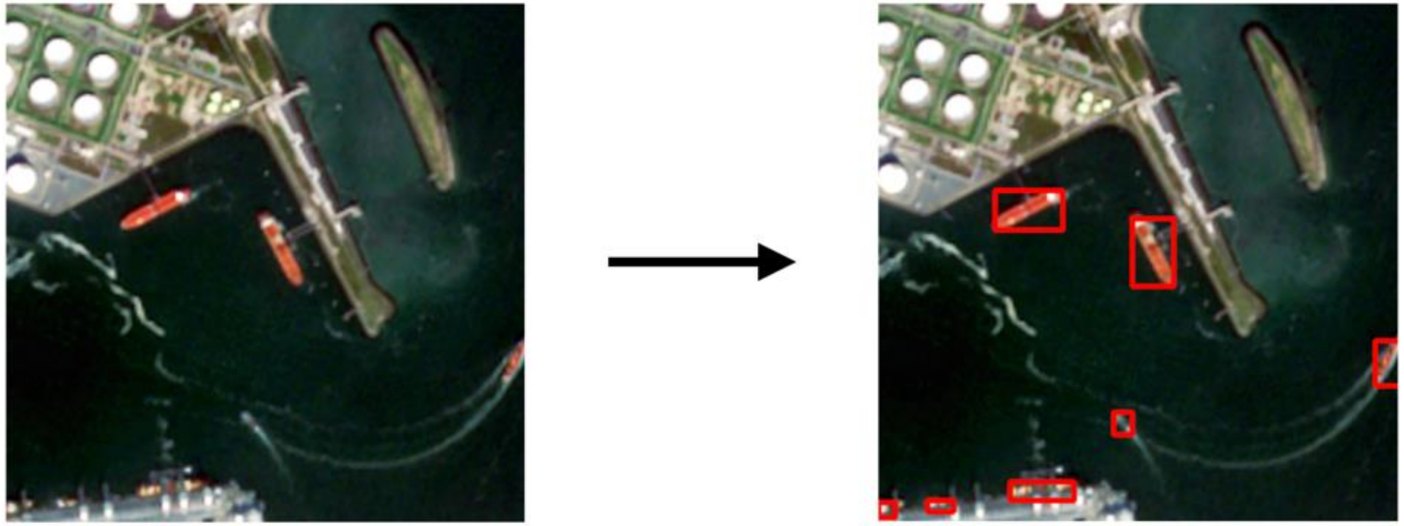
- The three versions provide effectively the same performance
- SPARROW provides 2x performance compared to the scalar version
- The GPU has low performance because of the minimal GPU configuration and non-optimized hw/sw
- With improvements in the near future we expect similar performance with COTS GPUs

Ship Detection Use Case

Ship Detection

Localizing ships in satellite imagery

- YOLOX-Tiny object detection model trained on Airbus Ship Detection Dataset

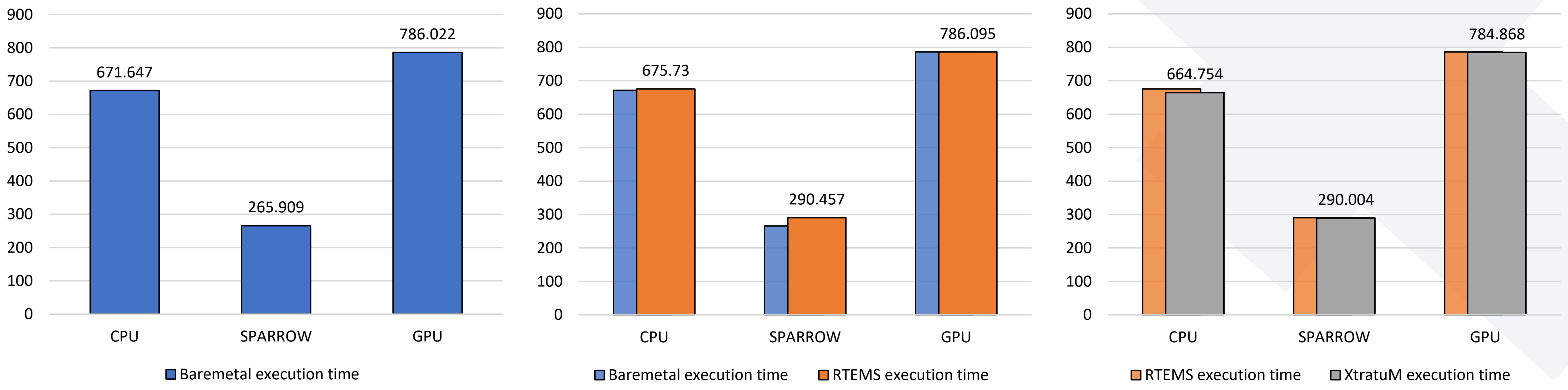


3 Channel Input Image

List of Bounding Boxes

Ship Detection

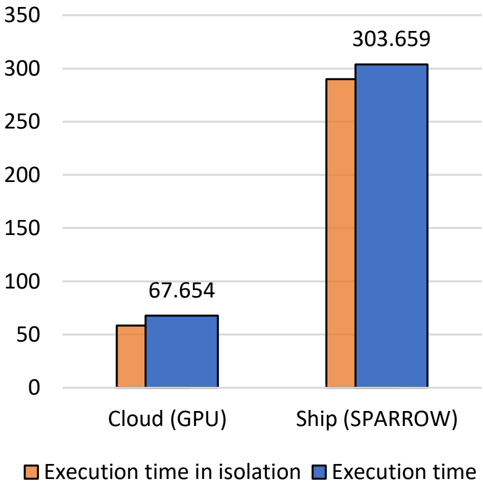
- Evaluation of UC#3 in isolation – Baremetal, RTEMS and XtratuM

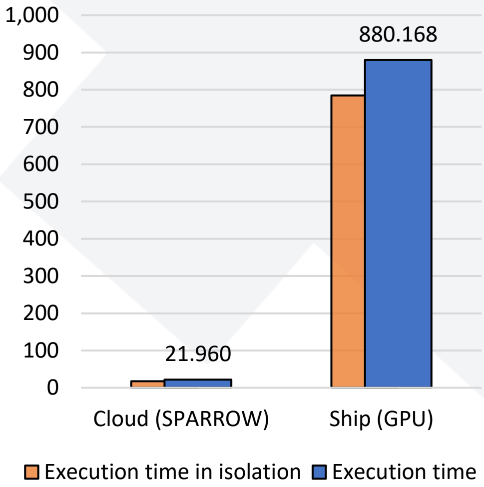


- The three versions provide effectively the same performance
- SPARROW provides 2.55x performance compared to the scalar version
- As in UC#2, the GPU has low performance because of the minimal GPU configuration and non optimised hw/sw
- With improvements in the near future we expect similar performance with COTS GPUs

UC#2 and UC#3 executed together

- Evaluation of UC#2 and UC#3 in XtratuM





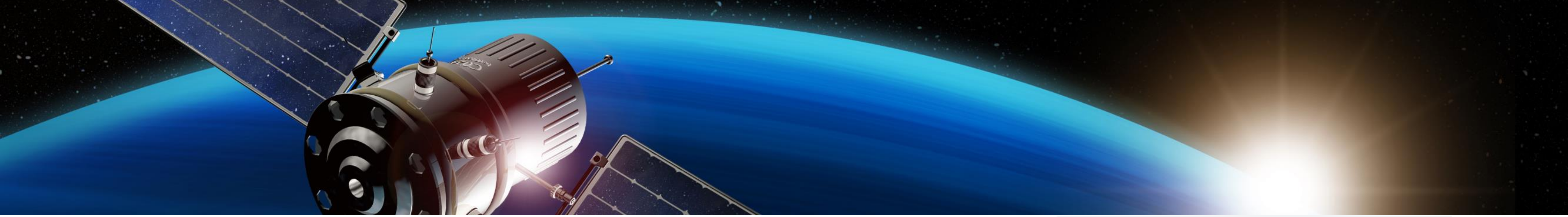
- Each partition is executed on a different core
- SPARROW has less interference than the GPU, first test has the largest slowdown due to time sharing the GPU in both partitions

Conclusions

- A high-performance RISC-V prototype platform was implemented on an FPGA with a qualifiable software stack
- Several use cases of different criticalities were executed in parallel
- Significant acceleration achieved
- Important hardware savings and software qualification cost reduction

Future work

- The platform and its software will be released soon as open source:
  - https://gitlab.bsc.es/metasat
- OBPMark-ML developments to be merged with ESA’s repository:
  - https://github.com/OBPMark



https://metasat-project.eu/
info@metasat-project.eu

https://twitter.com/MetasatProject

https://www.linkedin.com/company/metasat-project

METASAT has received funding from the European Union's Horizon Europe programme under grant agreement number 101082622.