



MODULAR MODEL-BASED DESIGN AND
TESTING FOR APPLICATIONS IN SATELLITES

TASTE and AADL in the METASAT Model-Based Engineering Workflow

Leonidas Kosmidis (BSC)

ADEPT 2024



**Barcelona
Supercomputing
Center**
Centro Nacional de Supercomputación

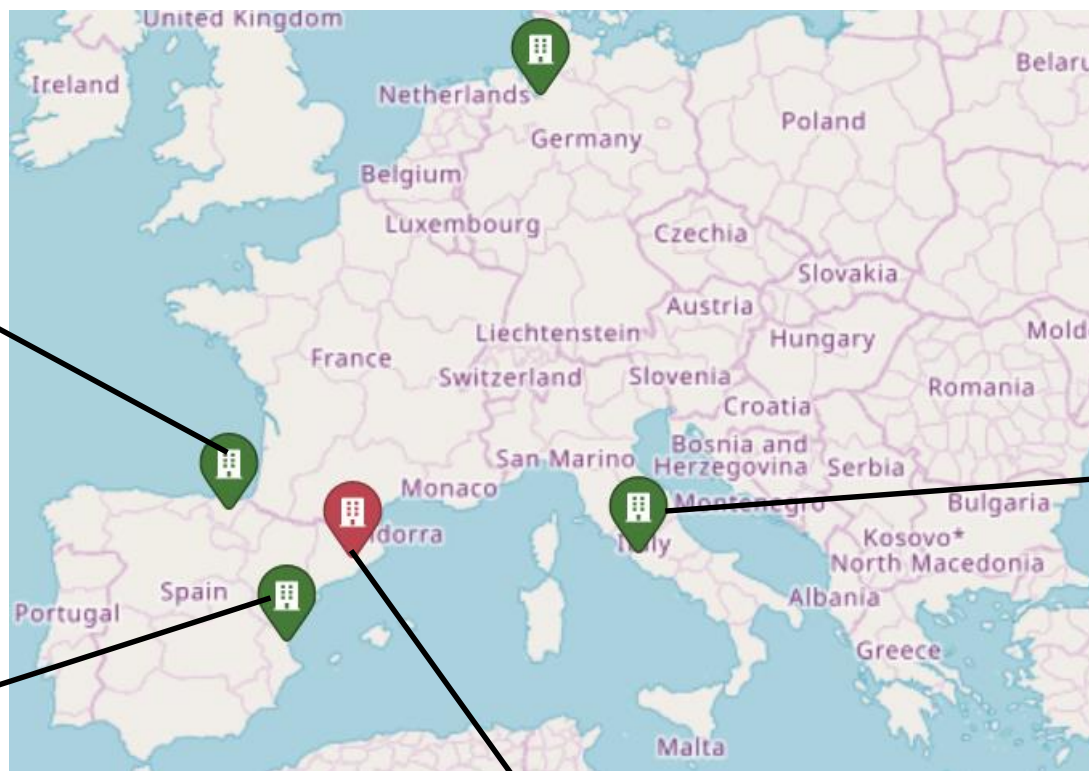


**UNIVERSITAT POLITÈCNICA
DE CATALUNYA**
BARCELONATECH

June 14, 2024

METASAT Consortium

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



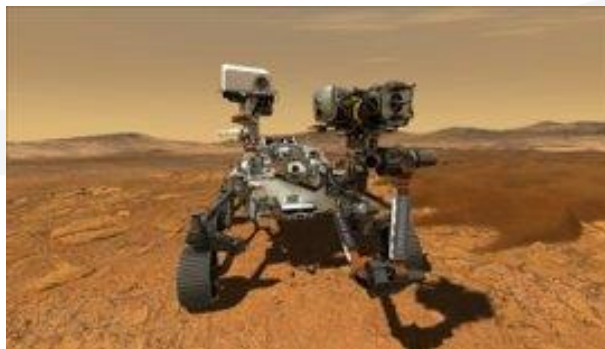
Collins Aerospace



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Introduction

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



METASAT Overview

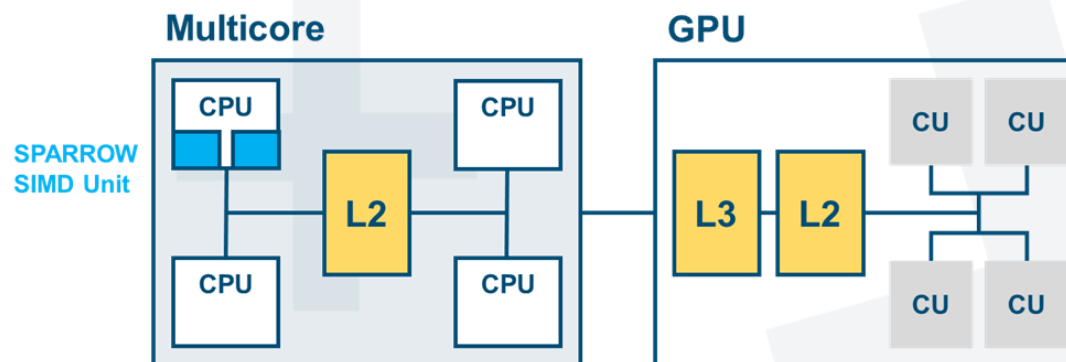
- Modern aerospace systems require new, advanced functionalities
 - Artificial Intelligence (AI)
 - High Resolution Sensors
 - Optical communications
 - Advanced Robotics...
- Advanced functionalities require complex **hardware** and **software** compared to the existing space technologies
- High Performance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

Model-Based Design

- Model-Based Design can reduce the development and verification time for these complex platforms
- Development can be assisted by high level design methods (models) from which code can be automatically generated
 - Correct-by-construction
 - Various levels of verification: model-in-the-loop, software-in-the-loop, processor-in-the-loop etc
 - Virtual platforms allow starting software development before the hardware is ready
 - Break the dependency between hardware and software development

Hardware Selection

- No hardware with such 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



Virtualisation

- Time and Space isolation provide benefits for faster and easier integration
- Components can be developed and tested in isolation
- Fault Detection, Isolation and Recovery (FDIR)
- XtratuM NG hypervisor by fentISS

METASAT



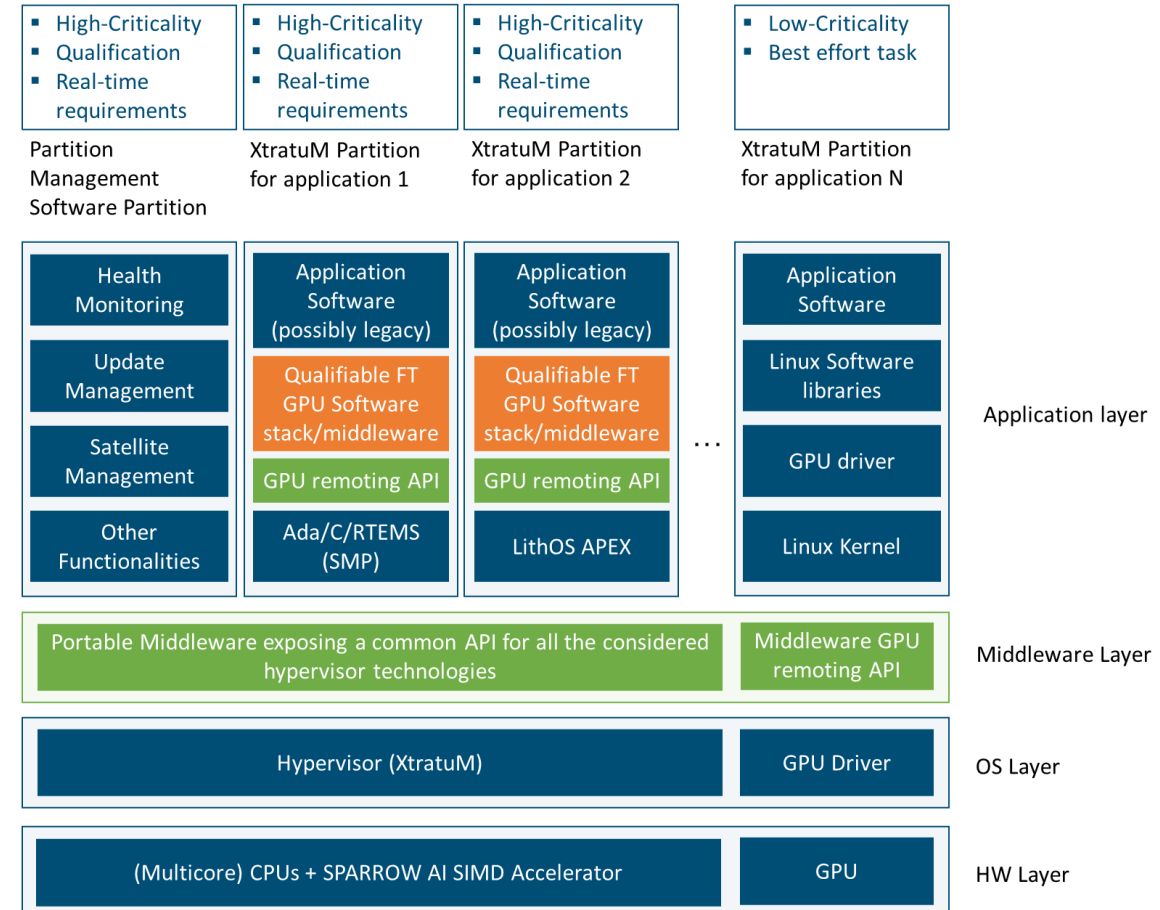
- METASAT will rely on open source and standardized technologies
 - Maximise interoperability and avoid vendor lock-in
 - Facilitate the development of a space ecosystem
- ESA's TASTE Open Source Model Based Design framework, enhanced with support for high performance platforms such as multicores and GPUs
- Open Source Processor technologies such as Gaisler's NOEL-V RISC-V processors
 - Enhancement with AI processing acceleration capabilities

taste

The METASAT RISC-V Platform

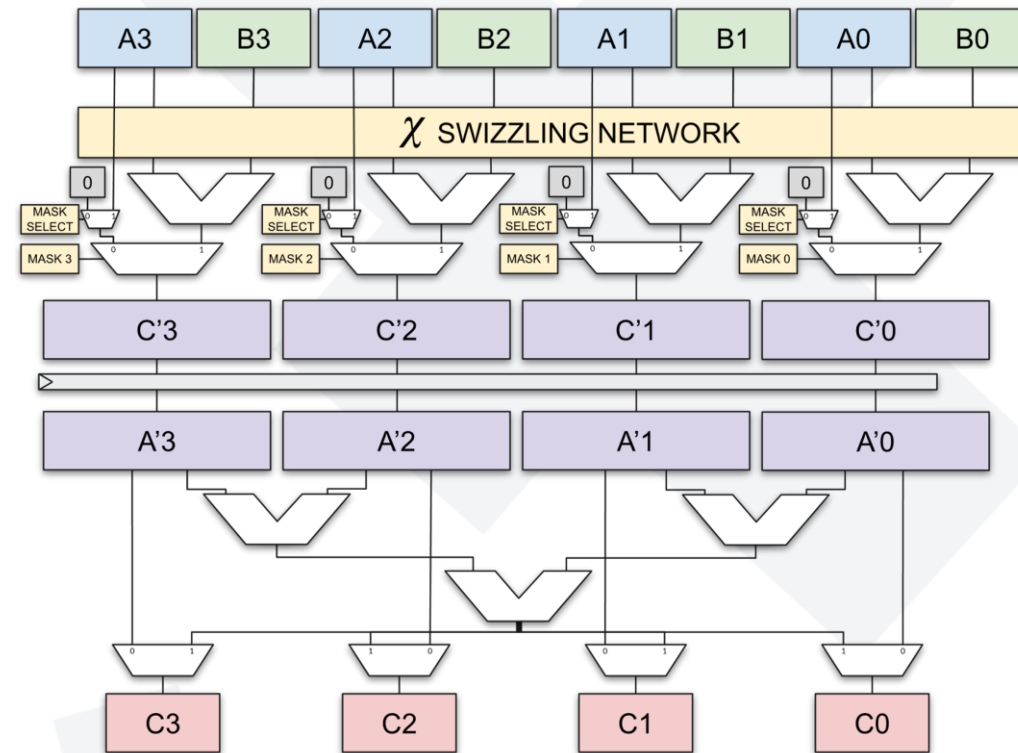


- Mixed Criticality Platform
- FPGA Prototype on a Xilinx VCU118
- Multicore CPU Based on NOEL-V + SPARROW AI SIMD Accelerator
 - Qualifiable software stack for high criticality software with moderate AI acceleration needs



The METASAT RISC-V Platform

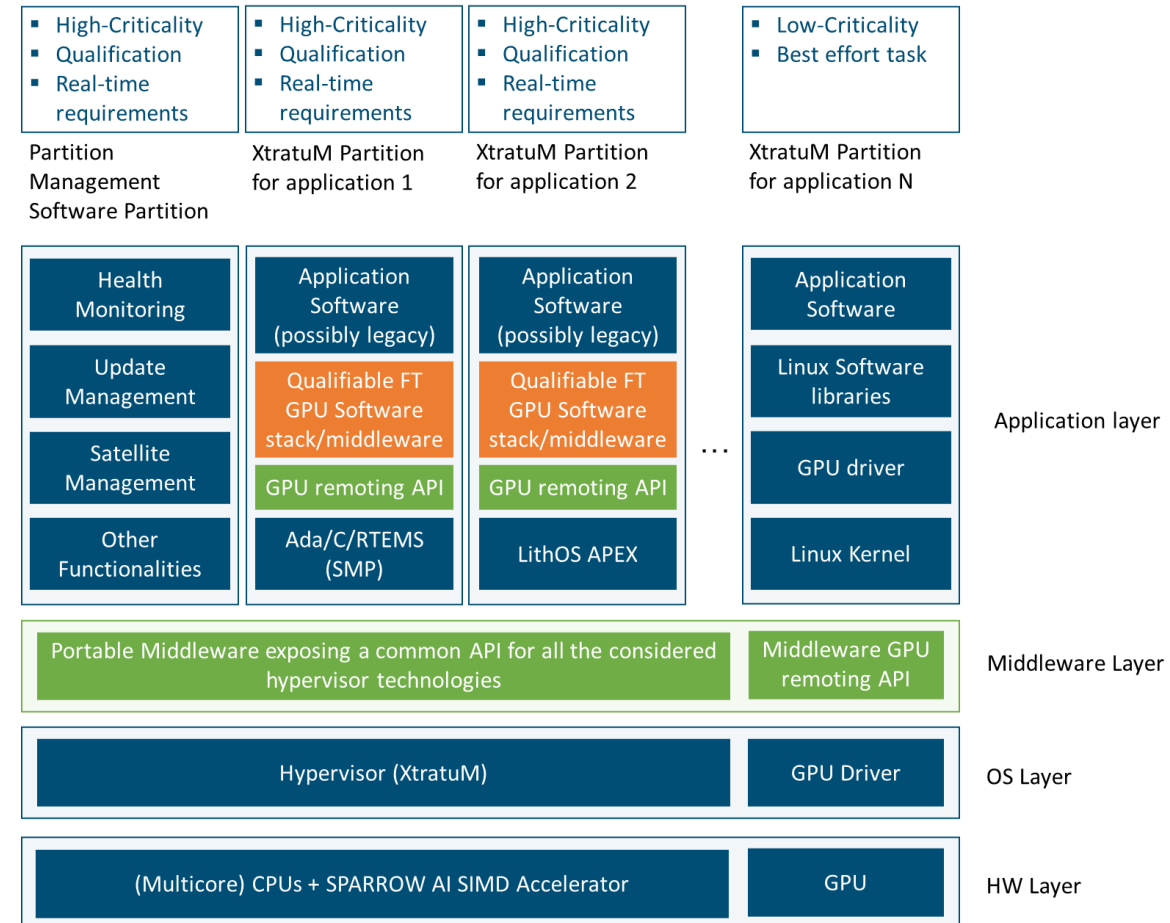
- SPARROW AI SIMD Accelerator
- High-performance, Low-cost at least 30% smaller than conventional vector processors with similar performance
- Minimal core modifications
 - incremental qualification
- Key features: reuse of integer register file, short SIMD unit (8-bit), swizzling, reductions
- Intrinsic-like software support similar to ARM's NEON



The METASAT RISC-V Platform



- Mixed Criticality Platform
- FPGA Prototype on a Xilinx VCU 118
- Configurable Vortex RISC-V GPU
 - Enhancements for real-time execution and reliability
 - Qualifiable software stack for tasks requiring very high performance
 - Enable the use of GPUs from bare metal, or RTOS
 - Share the GPU among partitions
- The hardware platform will be open sourced as well as much of its software



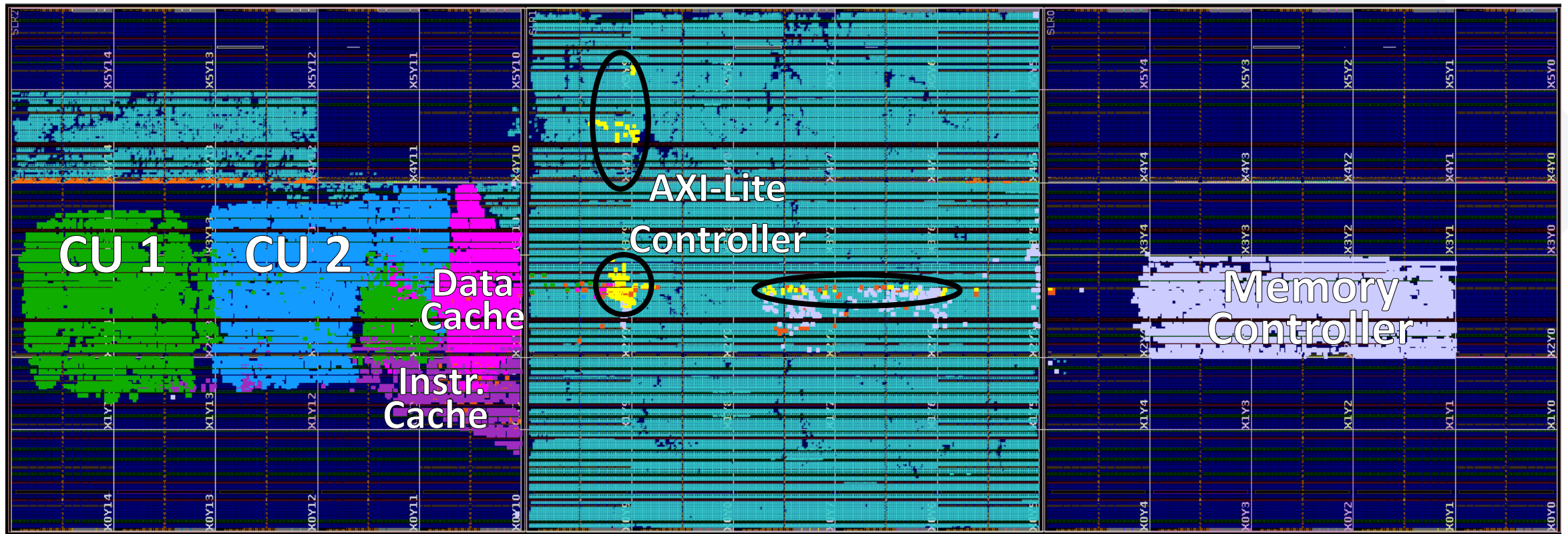
The METASAT RISC-V Platform: Current Status



- FPGA Resource utilisation
- Current configuration:
 - 4 NOEL-V high performance, Dual-Issue, FPLite, Hypervisor support + with 2 SPARROW accelerators each
 - L2 cache L2Lite
 - GPU: 64bit 2 CUs, 4 threads each
 - GRETH Ethernet
 - UART
- A design space exploration will be performed to find the best configuration for the project use cases

FPGA Utilization

Multicore GPU



The METASAT RISC-V Platform: Current Status



- Able to run OpenMP programs on both FPGA and QEMU under RTEMS, with and without XtratuM
- See [1] for performance results and comparison with other multicore architectures
- SPARROW support in RTEMS and XtratuM
 - RTEMS Compiler modifications
 - Support for SPARROW control register to added to RTEMS
 - TensorFlow Lite Support

[1] M. Solé, J. Wolf, I. Rodriguez, A. Jover, M. M. Trompouki, L Kosmidis, D. Steenari. Evaluation of the Multicore Performance Capabilities of the Next Generation Flight Computers. Digital Avionics Systems Conference (DASC) 2023

```
unet32alpha1_0_upsample -- root@a3de48b...
...ogin!::~ --zsh ...
...olo_v8 --zsh ...
...ogin!::~ --zsh ...
...jwolf@caos17 ...
...nloads --zsh ...
'build-riscv-rtems6-rv64imafdc' finished successfully (0.802s)
[root@a3de48bca934:/workspace/app/matrix_multiplication_bench# ./sim-qemu

OPENMP DISPLAY ENVIRONMENT BEGIN
  _OPENMP = '201511'
  OMP_DYNAMIC = 'FALSE'
  OMP_NUM_THREADS = '4'
  OMP_SCHEDULE = 'DYNAMIC'
  OMP_PROC_BIND = 'FALSE'
  OMP_PLACES = ''
  OMP_STACKSIZE = '0'
  OMP_WAIT_POLICY = 'PASSIVE'
  OMP_THREAD_LIMIT = '4294967295'
  OMP_MAX_ACTIVE_LEVELS = '1'
  OMP_NUM_TEAMS = '0'
  OMP_TEAMS_THREAD_LIMIT = '0'
  OMP_CANCELLATION = 'FALSE'
  OMP_DEFAULT_DEVICE = '0'
  OMP_MAX_TASK_PRIORITY = '0'
  OMP_DISPLAY_AFFINITY = 'FALSE'
  OMP_AFFINITY_FORMAT = 'level %L thread %i affinity %A'
  OMP_ALLOCATOR = 'omp_default_mem_alloc'
  OMP_TARGET_OFFLOAD = 'DEFAULT'
  GOMP_CPU_AFFINITY = ''
  GOMP_STACKSIZE = '0'
  GOMP_SPINCOUNT = '30000'
OPENMP DISPLAY ENVIRONMENT END
Creating and starting an application task
bla
Application task was invoked with argument and has id of 0x00000
4
4
s
1024
Using device: Generic device
Elapsed time Host->Device: 0.000000000 milliseconds
Elapsed time kernel: 84672.781250000 milliseconds
Elapsed time Device->Host: 0.000000000 milliseconds

[ RTEMS shutdown ]
CPU: 2
RTEMS version: 6.0.0.3612dc7d61d91e0bc121b2d226a1b3082ff9e333
RTEMS tools: 12.2.1 20230224 (RTEMS 6, RSB bfed51462eafcb6a5102a2d6d80b233f3c6ef635, Newlib 17ac400)
executing thread ID: 0x0a010002
executing thread name: TA1

*** FATAL ***
```

The METASAT RISC-V Platform: Current Status

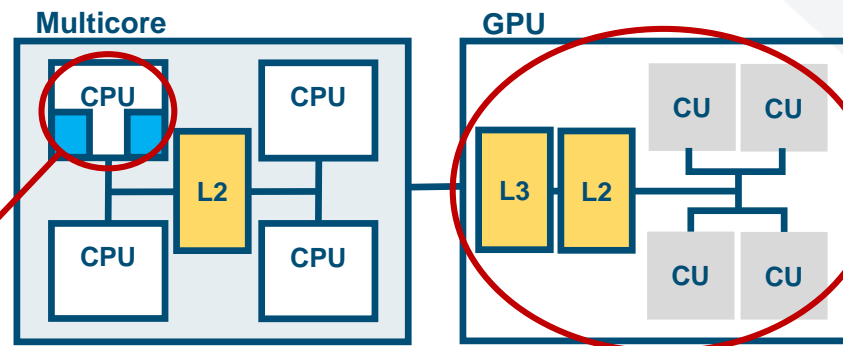


- NOEL-V Integration with Vortex GPU
 - AXI interface added to Vortex
 - Able to offload a GPU kernel in bare-metal, RTEMS and under the XtratuM hypervisor
 - Established programming methodology for using the GPU in the METASAT platform
 - Precompile the GPU kernel
 - Common practice in safety critical systems (OpenGL SC, Vulkan SC)
 - Embed the kernel binary in the program executable
 - No filesystem
 - Linker script modifications
- Supported GPU APIs:
 - Vortex GPU API
 - On going work on Brook Auto[1]/BRASIL[2], OpenCL and OpenGL SC 2.0
 - CUDA code generation from Matlab/Simulink and CUDA → Brook Auto translator

[1] Trompouki and Kosmidis, Brook Auto: High-Level Certification-Friendly Programming for GPU-powered Automotive Systems [DAC'18], <https://github.com/lkosmid/brook>

[2] Trompouki and Kosmidis, BRASIL: A High-Integrity GPGPU Toolchain for Automotive Systems [ICCD'19]

The METASAT Accelerators



```

March
Matrix multiplication
- 8.0561 s
Accelerated matrix multiplication
- 1.4688 s

Success!

Matrix multiplication
- 8.0561 s
Accelerated matrix multiplication
- 1.4688 s

Success!

Matrix multiplication
- 8.0561 s
Accelerated matrix multiplication
- 1.4688 s

Success!

Matrix multiplication
- 8.0561 s
Accelerated matrix multiplication
- 1.4688 s

Success!

Matrix multiplication
- 8.0561 s
Accelerated matrix multiplication
- 1.4688 s

Success!

```

```

March
GPU time: 87062 us

Vector size: 40 bytes
CPU time: 4 us
GPU time: 265 us

Vector size: 400 bytes
CPU time: 24 us
GPU time: 303 us

Vector size: 4000 bytes
CPU time: 168 us
GPU time: 380 us

Vector size: 40000 bytes
CPU time: 2409 us
GPU time: 1195 us

Vector size: 400000 bytes
CPU time: 28502 us
GPU time: 9979 us

Vector size: 4000000 bytes

```


Virtual The METASAT RISC-V Platform



- METASAT multicore CPU modeled in QEMU
 - On-going support for SPARROW
- Vortex GPU is simulated in Verilator
 - Cycle-accurate behavioural simulation
 - SystemVerilog to SystemC/C++
- Work started with GR740 and GR712RC models
 - Accurate modeling of the GR740 GRFPU
 - Cannot handle subnormal numbers as input, raise *unfinished_Fpop* exception
 - Handling denormalized numbers with the GRFPU
 - <https://www.gaisler.com/doc/antn/GRLIB-AN-0007.pdf>
 - Can boot unmodified Linux and RTEMS binaries

```
lkosmidi@Caos17:~/scratch/lkosmidi/rtems$ qemu_inst/bin/qemu-syst
em-sparc -machine gr740 -serial stdio -kernel ~/rcc-1.3.0-gcc/rc
c-1.3.0-gcc/src/samples/rtems-shell_gr740
VNC server running on 127.0.0.1:5900
--- RISC TOPOLOGY ---
-> DEV 0xd52b8 GAISLER_LEON4
-> DEV 0xd5320 GAISLER_APBMS1
-> DEV 0xd5388 GAISLER_IRQMP
-> DEV 0xd53f0 GAISLER_GPTIMER
-> DEV 0xd5458 GAISLER_APBUART

You can use the shell commands drvmgr and pci to find out
more about the system

Creating /etc/passwd and group with three useable accounts
root/pwd , test/pwd, rtems/NO PASSWORD

RTEMS Shell on dev/console. Use 'help' to list commands.
SHLL [/] #
```

```
Segment Routing with IPv6
sit: IPv6, IPv4 and MPLS over IPv4 tunneling driver
NET: Registered protocol family 17
Key type dns_resolver registered
Freeing unused kernel memory: 5760K
This architecture does not have kernel memory protection.
Run /init as init process
Starting syslogd: OK
Starting klogd: OK
Running sysctl: OK
Saving random seed: random: dd: uninitialized urandom read (32 bytes read)
OK
Starting network: OK
random: crng init done
ssh-keygen: generating new host keys: RSA DSA ECDSA ED25519
Starting sshd: OK

Welcome to Buildroot
buildroot login:
0$ bash 1$ bash 2$ bash 3-$ bash
```

Virtual The METASAT RISC-V Platform



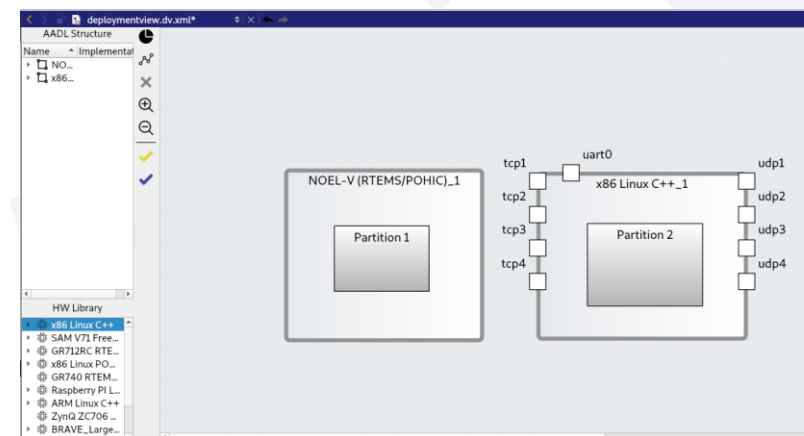
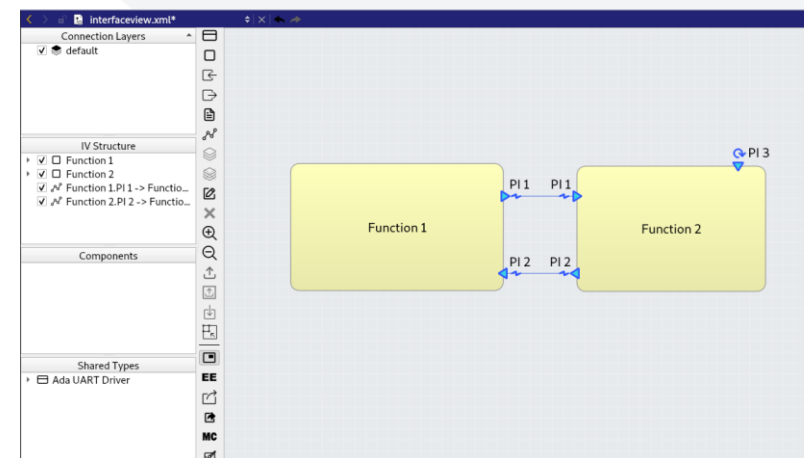
- NOEL-V model in QEMU
 - Can boot unmodified Linux and RTEMS binaries
- METASAT QEMU Model
 - Partial SPARROW support
 - L2-lite cache controller

METASAT MBD Toolchain

METASAT MBD Toolchain

TASTE

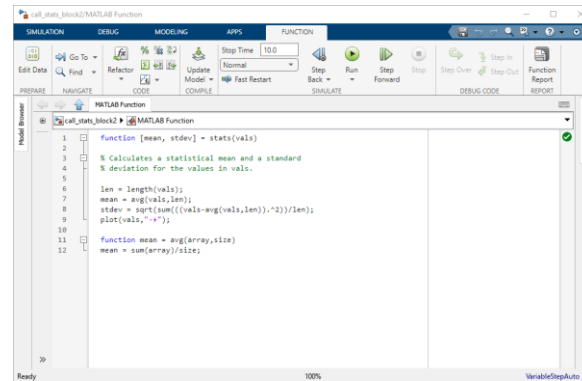
- Open-source toolchain for developing embedded real-time systems.
- Addresses modeling, automatic code generation and deployment of distributed systems.
- Compatibility with external tools
- Use of formal languages:
 - AADL (Architecture Analysis and Design Language)
 - ASN.1 (Abstract Syntax Notation One).
- Code generation support (Kazoo, Ocarina)



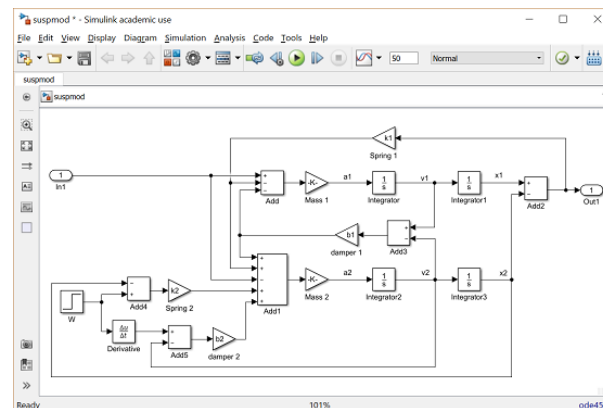
METASAT MBD Toolchain

MATHWORKS TOOLS

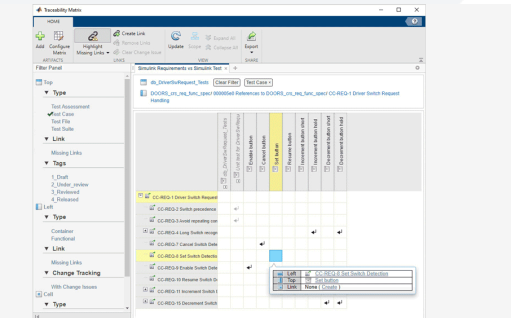
- Matlab



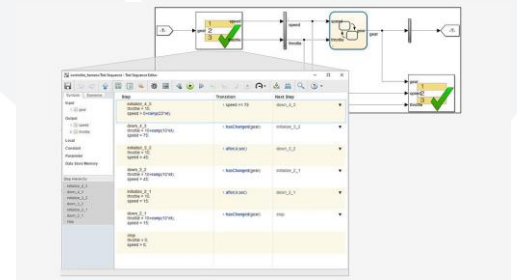
- Simulink



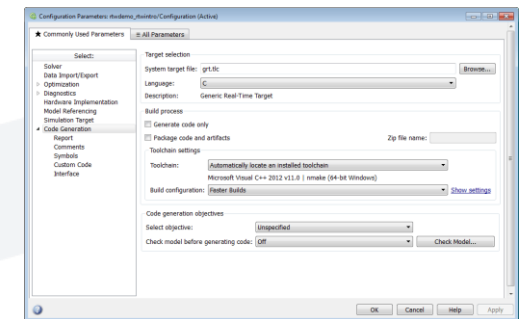
- Requirements Toolbox



- Simulink Test

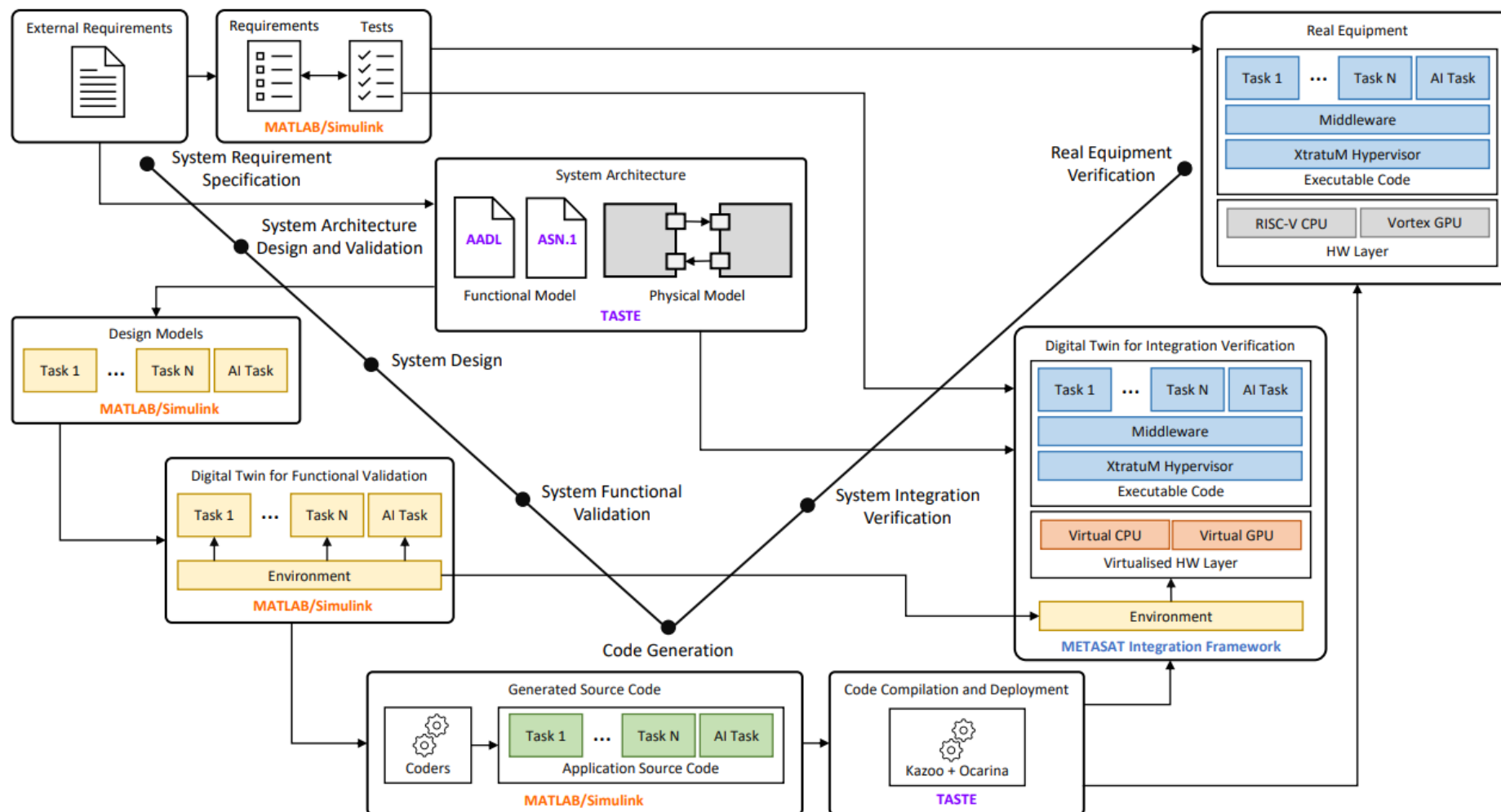


- Embedded Coder



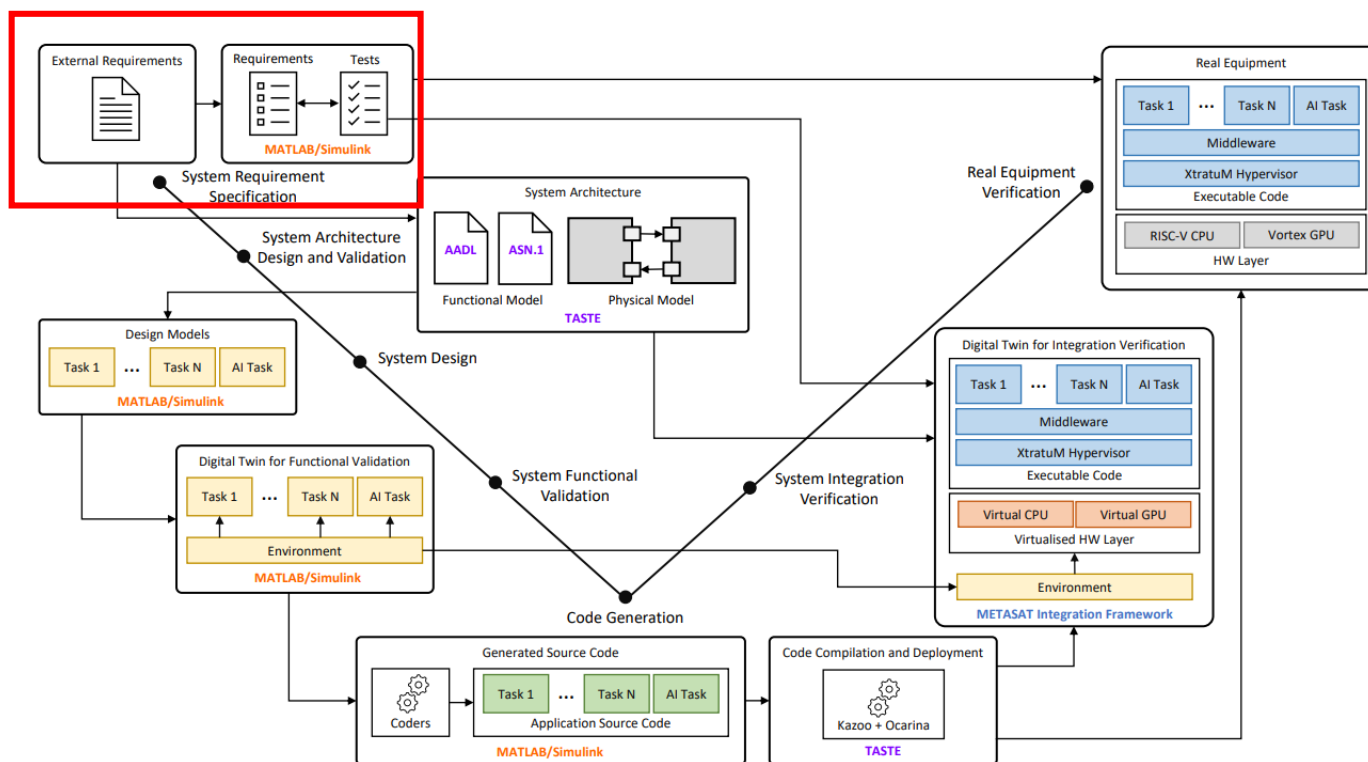
METASAT MBD Workflow

METASAT MBD Workflow



METASAT MBD Workflow

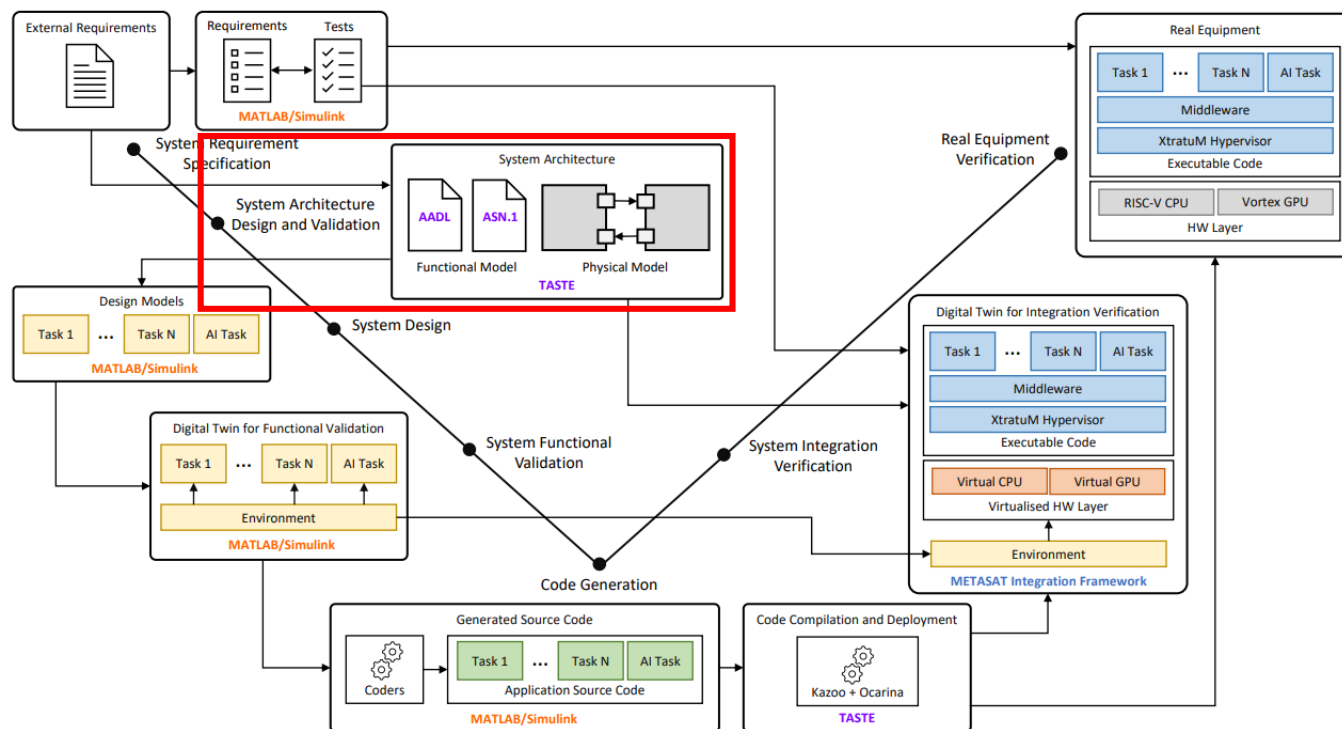
1. System Requirement Specification



- Collect, analyze, and formalize system requirements.
- Define test procedures
- Ensure traceability and coverage by linking collected system requirements to test cases.
- Tools: Matlab Requirements Toolbox

METASAT MBD Workflow

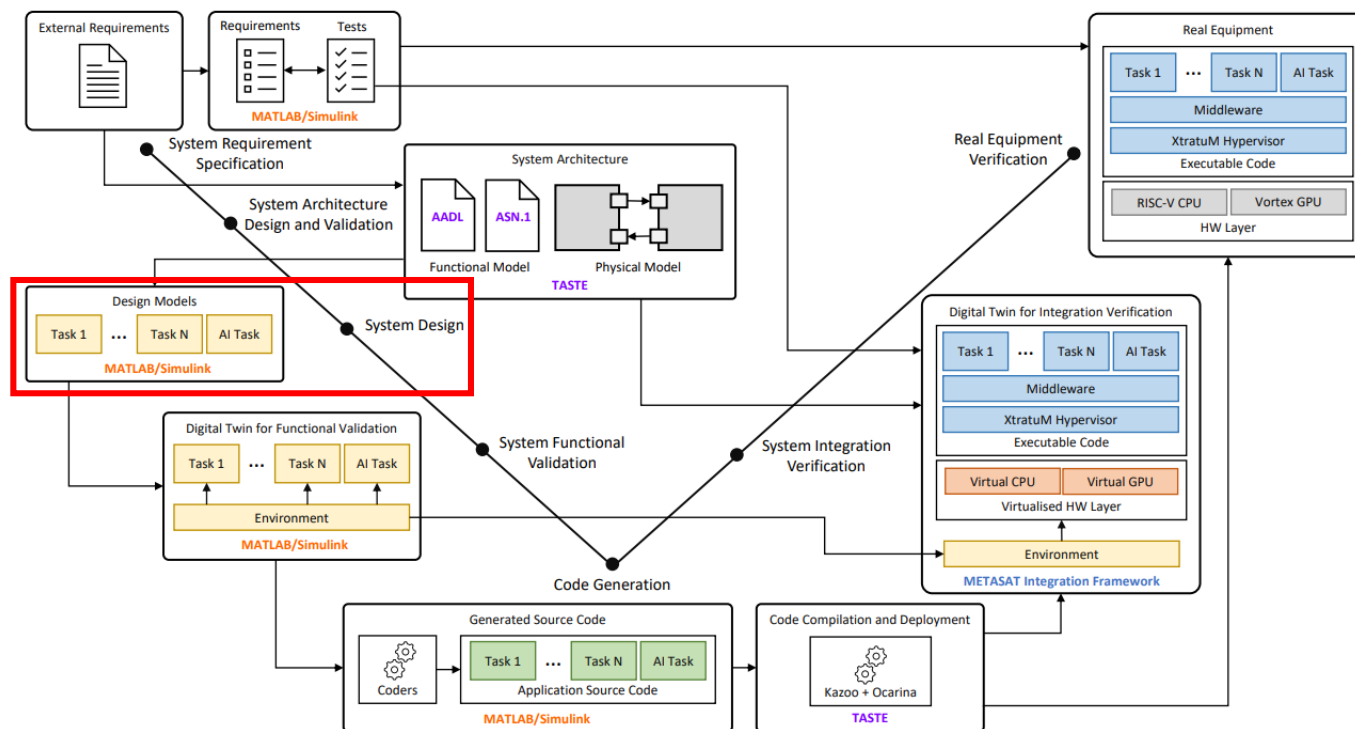
2. System Architecture Design and Validation



- Define and integrate:
 - A functional architecture model
 - Physical architecture model
- TASTE
 - **Interface View:** Functional components
 - **Data View:** Definition of data types
 - **Deployment View:** Definition of the physical architecture model
- AADL and ASN.1 languages
 - **AADL:** Functional and physical architecture of the system
 - **ASN.1:** Semantic information of the data and limitations of the values
- Validation at several points of the design phase
 - Requirement coverage (Simulink)
 - Architecture integration validation (TASTE)

METASAT MBD Workflow

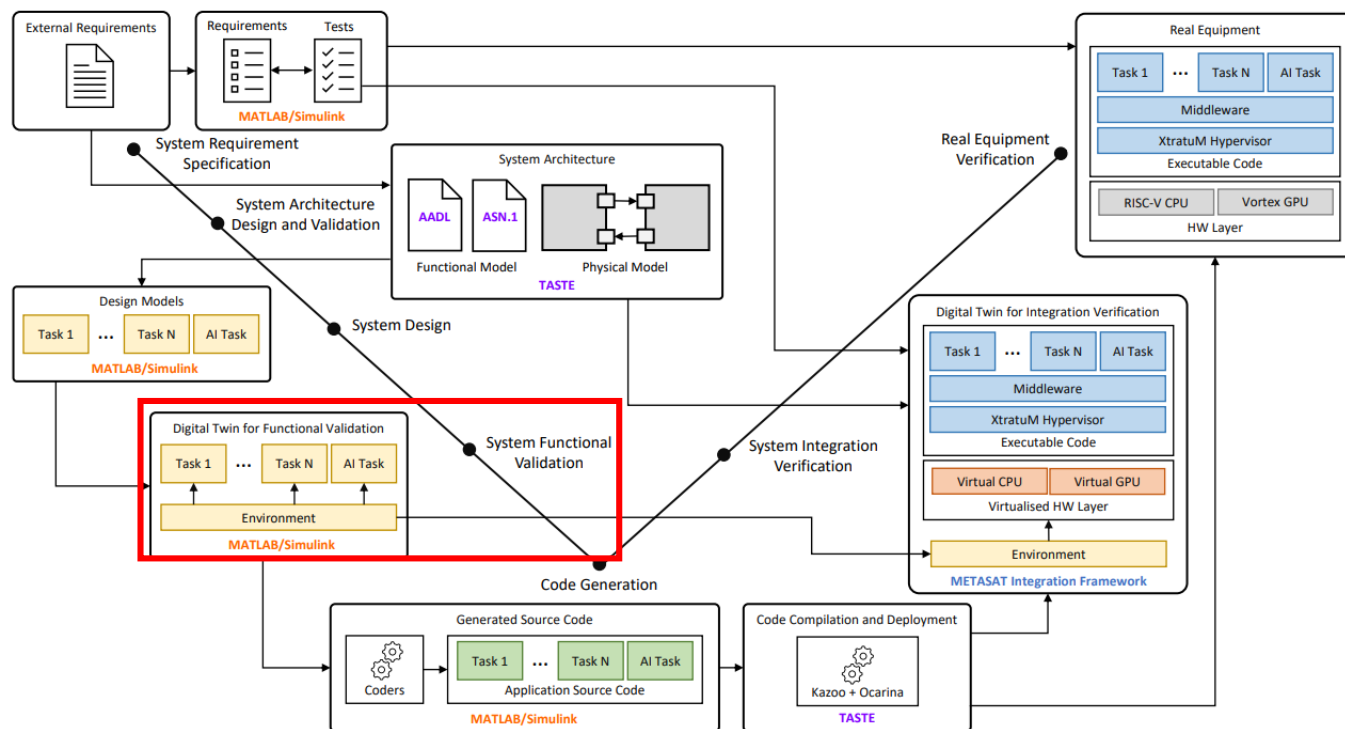
3. System Design



- Define the executable behavioural models for the functional elements
- TASTE: Automatic generation of design model skeletons as Simulink blocks
- Functionality implemented in Simulink
- Digital twin is created for V&V purposes

METASAT MBD Workflow

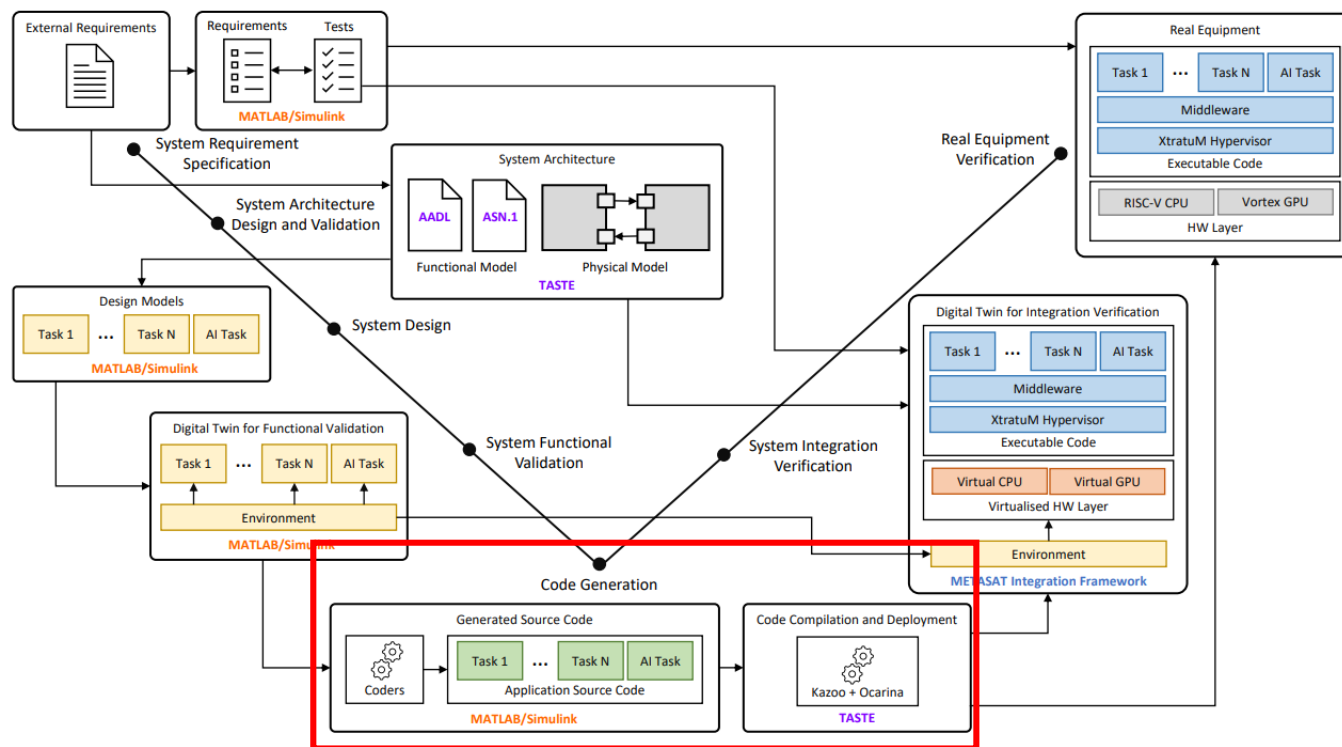
4. System Functional Validation



- Examines how well the behavioral design models from the previous phase fit together
- The obtained system model defines a functional Digital Twin of the system, used for functional validation analysis executing the test procedures from the first stage.
- Individual components are developed and tested independently, but also in a full system simulation.
- MiL validation strategy is used.
- Design traceability by using Simulink

METASAT MBD Workflow

5. Code Generation



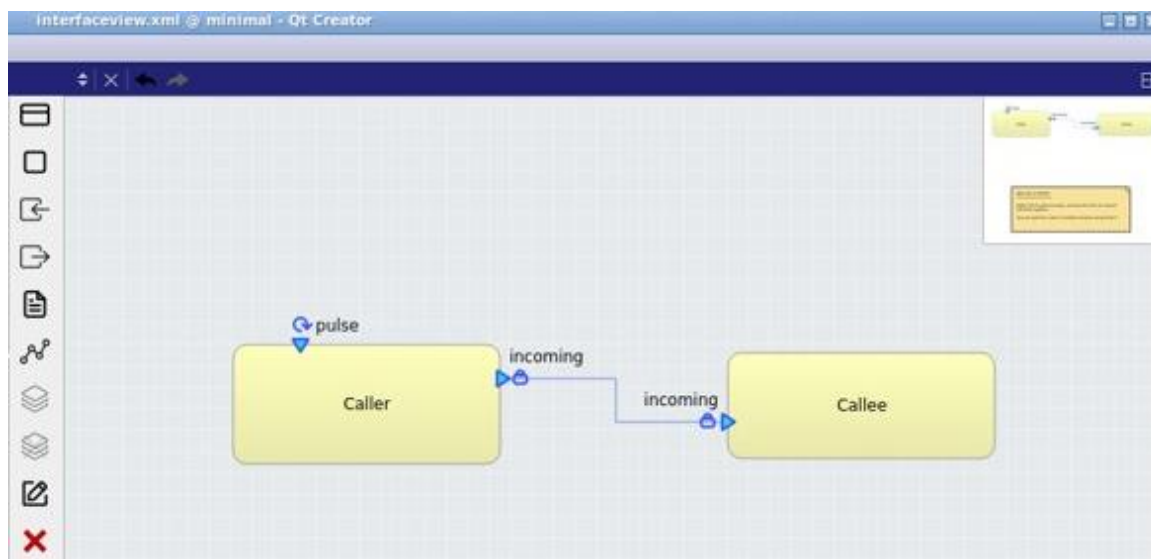
- Automatic SW code generation of the behavioural design models
- Decreases the effort of the SW verification stage.
- TASTE (Kazoo/Ocarina) and Matlab/Simulink (Embedded Coder/GPU Coder) provide code generation
- Working in code generation from ONNX and TensorFlow models and testing on a Jetson Xavier platform

RISC-V Support in TASTE

RISC-V support added in TASTE

- RTEMS
- RISC-V compiler
- QEMU RISC-V Simulator

Minimal communication example:

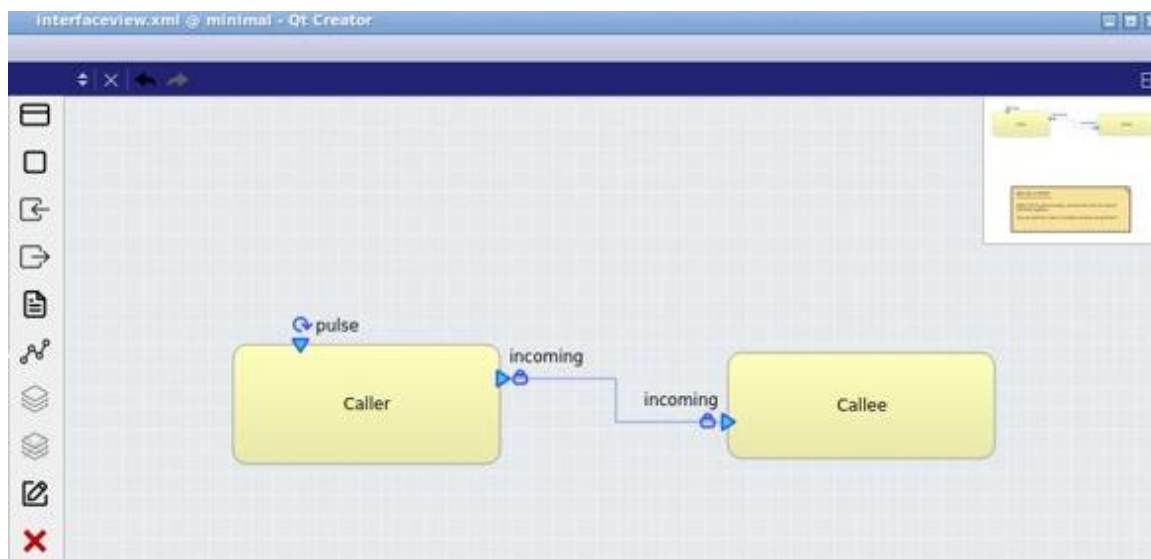


```
taste@taste10 ~/communication_example/work/binaries
$ taste-simulate-leon3 partition_1_leon3_rtems.exe
[Caller] Startup
[Callee] Startup
Send a=1
Received b=2
Send a=2
Received b=3
Send a=3
Received b=4
Send a=4
Received b=5
Send a=5
Received b=6
Send a=6
Received b=7
```

COTS GPU Support in TASTE

COTS GPU support added in TASTE

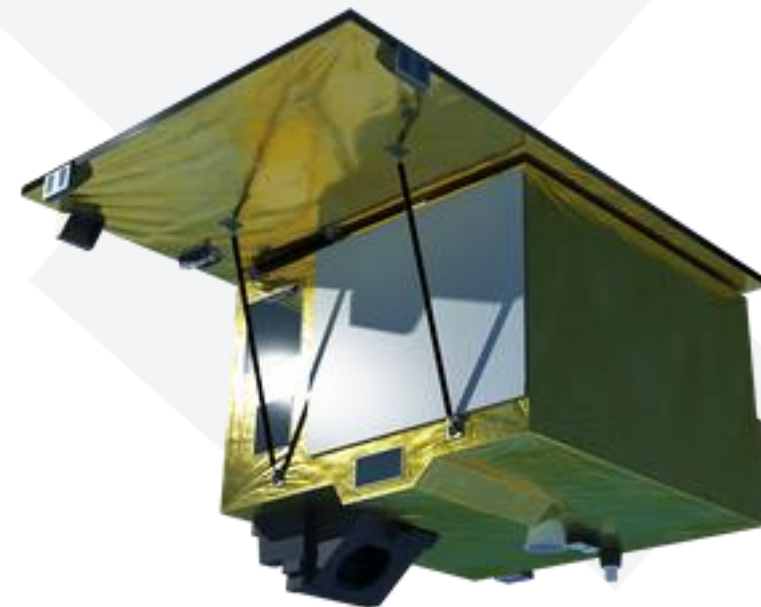
- Linux
- aarc64/nvcc compiler
- CUDA/TensorFlow



```
jannis — jetson@xavier-up2date: ~ — ssh jetson@192.168.10.45 — ...
[jetson@xavier-up2date:~]$ ./TASTE_RT_example_aarch64_xavier
[Callee] Startup
[Caller] Startup
[Pulse]
[Pulse]
[Pulse]
[Pulse]
[Pulse]
[Pulse]
[Pulse]
[Pulse]
[Pulse]
[Pulse]
[Pulse]
[Pulse]
[Pulse]
[Pulse]
[Pulse]
[Pulse]
[Pulse]
[Pulse]
[Pulse]
[Pulse]
[Pulse]
[Pulse]
```

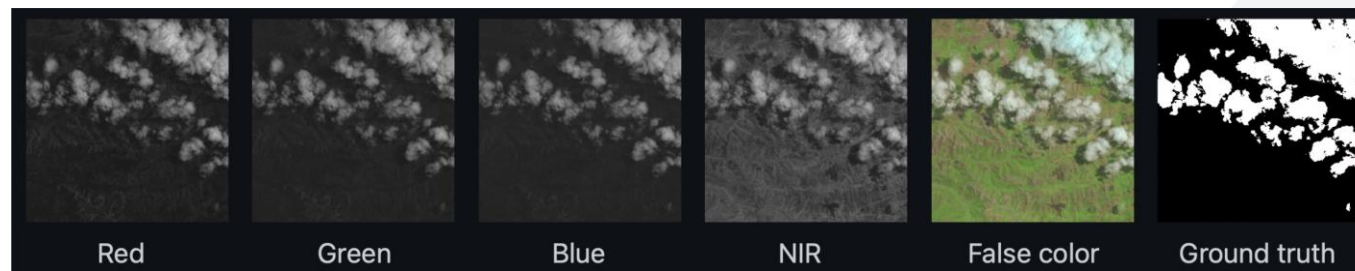
Project Use Cases

- 3 Project Use cases will be implemented
- OHB/DLR Use Case
 - Hardware interlocking
 - Protect against wrong software behaviour
 - Implement interlocks at software level instead of hardware
 - Reduce cost
 - Implement AI Based FDIR
 - To be accelerated on the CPU using the SPARROW AI accelerator
 - Housekeeping data from ENMAP



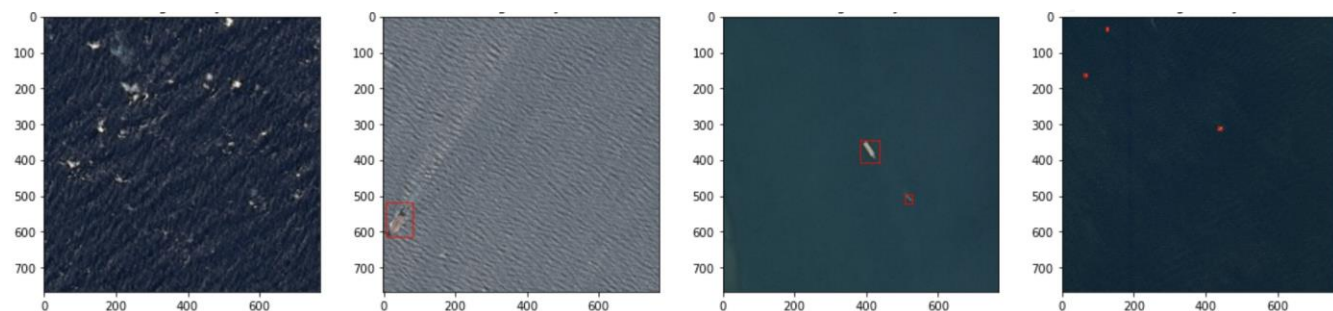
Project Use Cases

- 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)

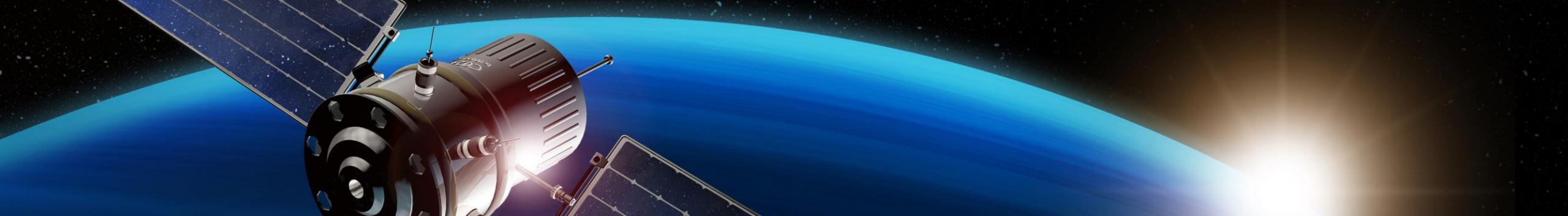
- Ship Detection



- To be executed on the GPU

Conclusion

- METASAT achieves a major milestone towards the use of GPUs and high performance platforms in space
- Provides an open source reference hardware platform
 - FPGA and virtual
 - Possible starting point for a future GPU tape out on a radiation tolerant/hardened process
- Solves key limitations preventing GPUs to be adopted today in institutional missions
 - Qualifiable software stack
- Model-Based design methodology supporting TASTE
- Evaluation to be performed with relevant space use cases



<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.