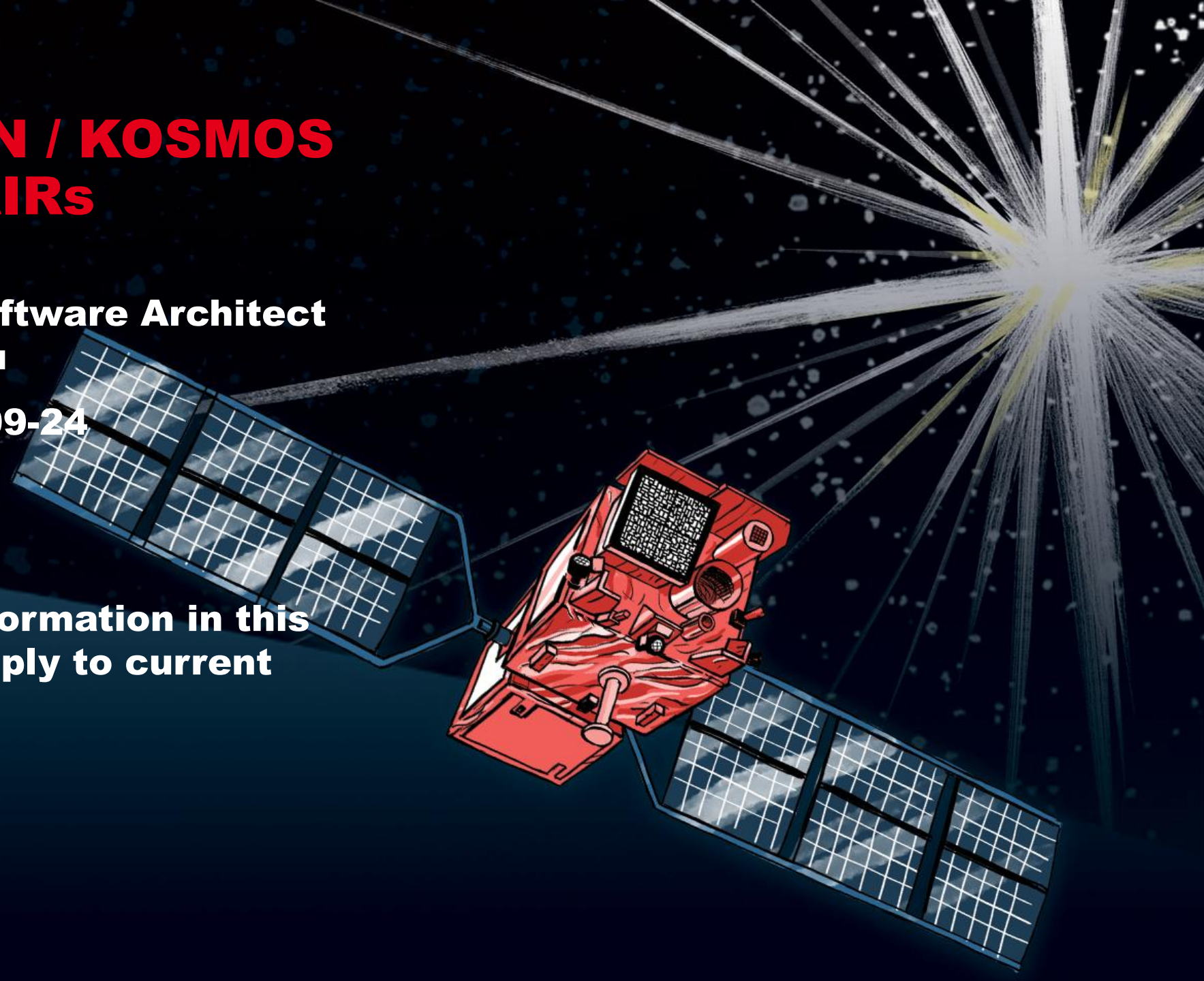
Using LVCUGEN / KOSMOS for SVOM/ECLAIRs

Frédéric Chateau – Software Architect
CEA Paris-Saclay / Irfu

KOSMOS Day – 2026-09-24



DISCLAMER: some information in this REX may no longer apply to current KOSMOS version !

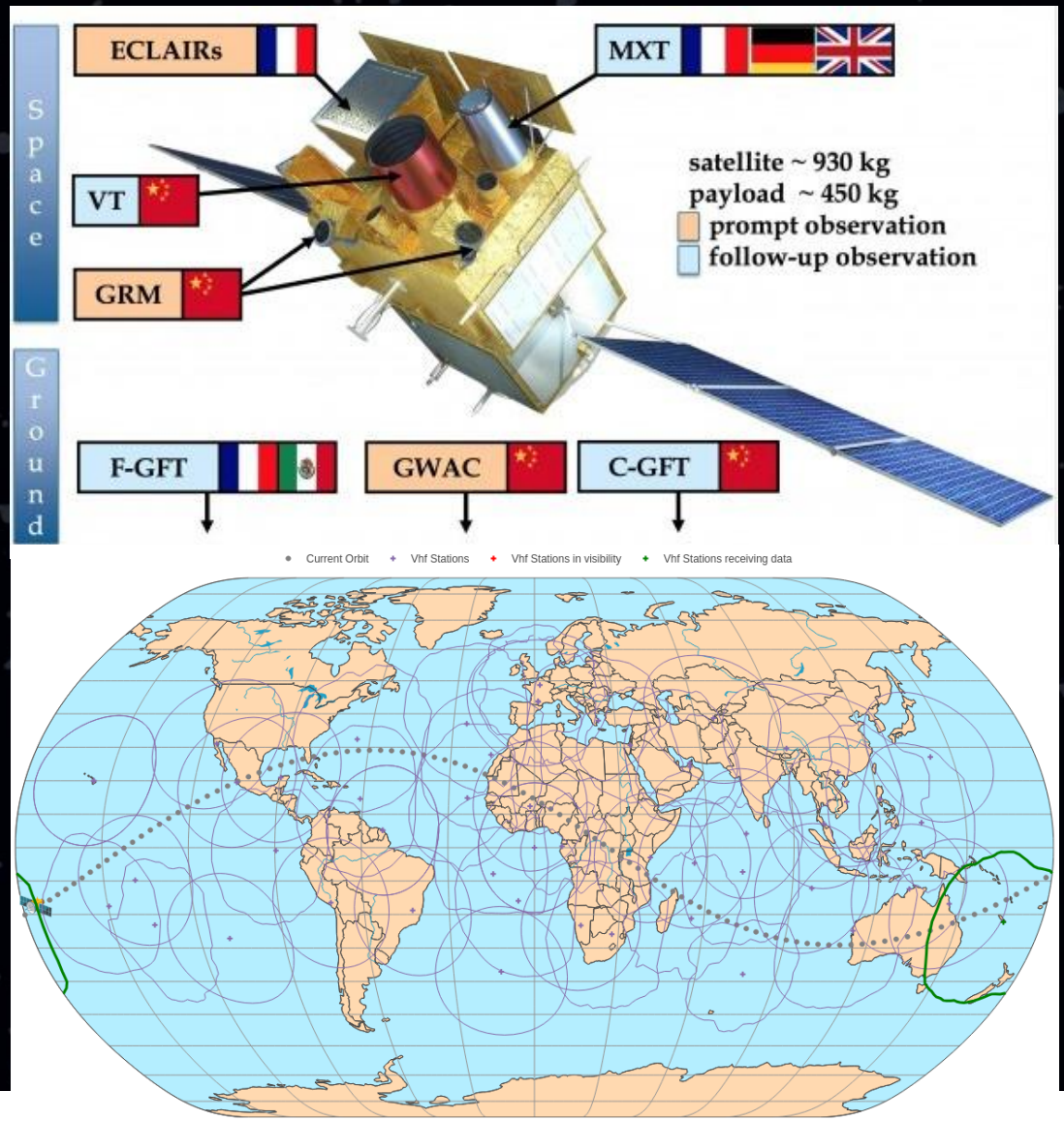
cea irfu

cnes
CENTRE NATIONAL D'ÉTUDES SPATIALES

Summary

- SVOM Mission
- ECLAIRs Instrument
- Software Requirements
- Software Projects
- Partitioning
- Storage
- Scheduling
- LVCUGEN Integration
- LVCUGEN Configuration
- LVCUGEN Modifications
- Advantages of LVCUGEN
- Limits of LVCUGEN
- Possible Improvements

SVOM Mission



China-France joint mission

- SVOM = Space Variable Objects Monitor
- Gamma-ray bursts (GRB) detection & analysis
  - Autonomous alerting and repointing
- 4 instruments → ECLAIRs, MXT, VT, GRM

Orbit

- ~650km altitude
- ~1h30 / revolution
- ± 30° latitude

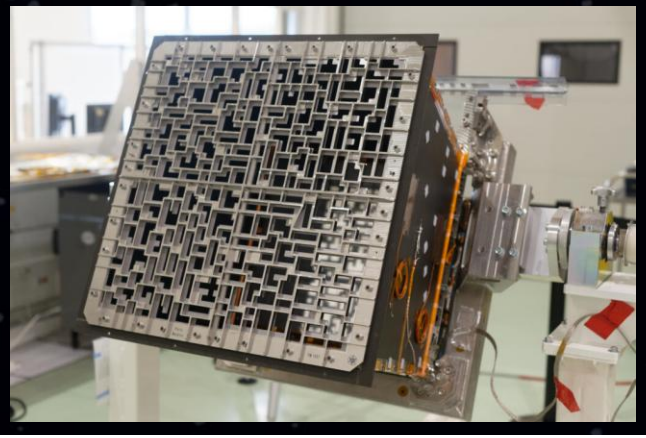
Ground network

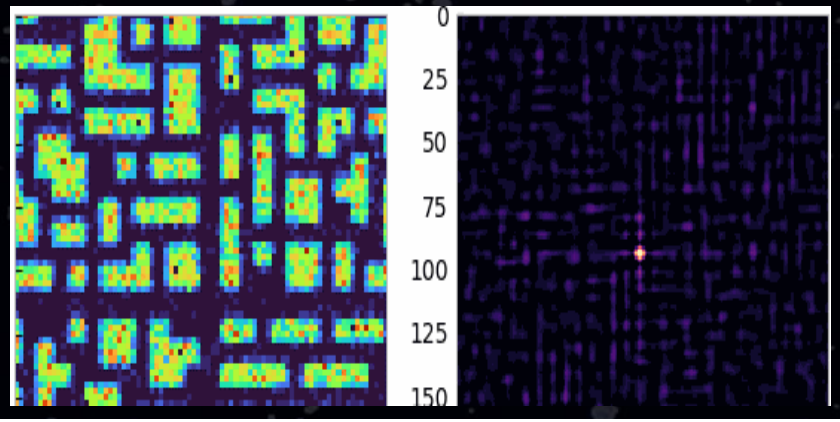
- 51 VHF stations
  - Receive real-time alerts (< 30s delay)
- 3 X-Band stations
- SVOM observatories → GWAC, C-GFT, F-GFT
- Partner observatories → VLT, GTC, CFHT, …

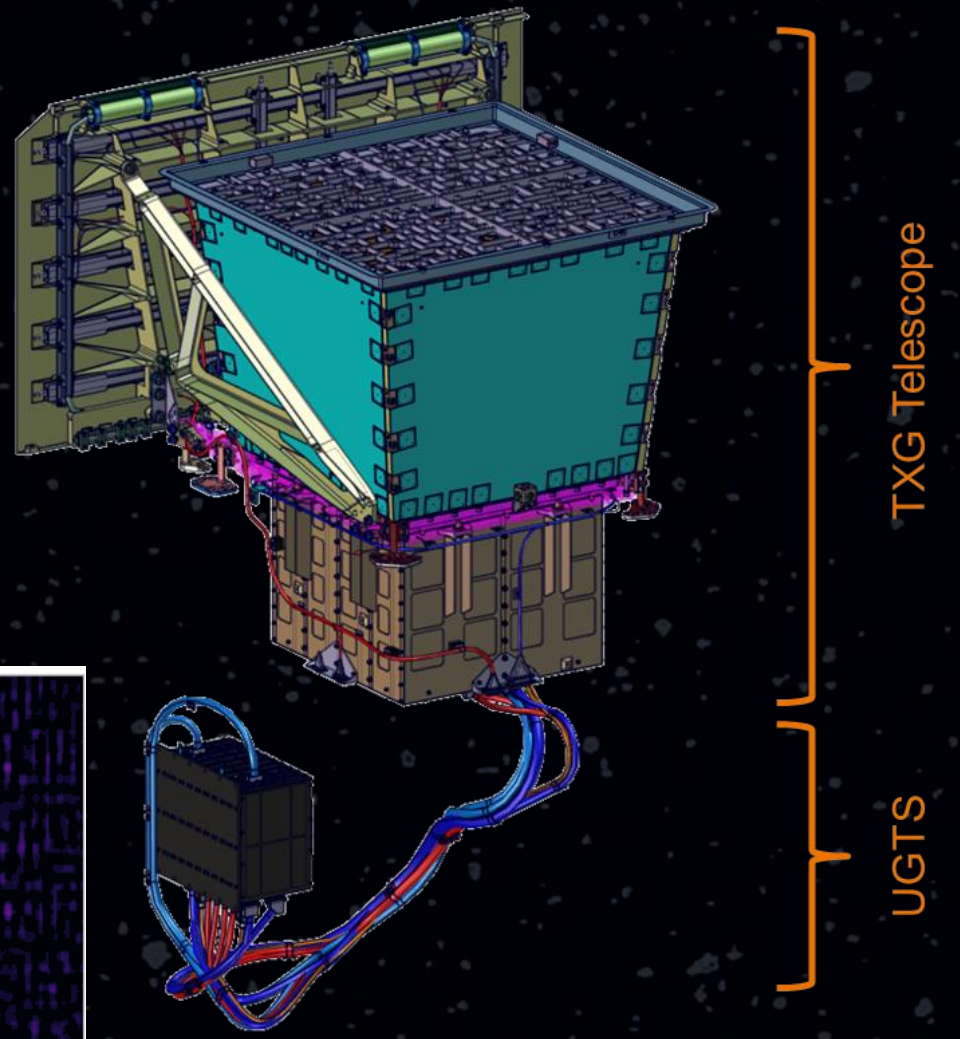
ECLAIRs Instrument

ECLAIRs: X/Gamma telescope

- Wide-field (89°×89° FoV) → 1/6th of the sky
  - Observation → detection → alert → repointing
    - For narrow-field instruments follow-up
- Coded-mask imaging
  - Source → pattern projected on camera pixels (*Shadowgram*)
    - Source position → pattern translation across X/Y axis
  - Deconvolution of mask pattern → Sky
    - Shows point-like sources
- Computing Unit → " UGTS"







Software Requirements

Key Attributes

- Onboard scientific code ⇒ heavy computing
- Onboard decision taking ⇒ GRB alerts
- Many features → single computing board
  - Originally 3 distinct DPU

Challenges

- Chinese satellite → EAR/ITAR regulations
  - If using U.S. components / software
- Limited computing : 2 x 80 MHz SPARC-V8
  - Gaisler GR712RC LEON 3
  - Small FPGA → moderate CPU offloading
- Limited storage : ASW < 2 MiB
  - Prototype was bigger
- Complex scientific code
  - mixed with safety-critical functions

(ECLAIRs) ASW main features

- Science
  - GRB detection (Triggers)
  - Alert sequence
  - Data acquisition (DAQ)
- Detector Management
  - Thermal regulation
  - Power supplies control
  - Detector configuration and control
  - Noisy pixels detection

ASW main features

- Usual ASW facilities
  - TM and TC handling
  - Observability (HKs, events)
  - Monitoring and FDIR
  - Modes
  - Configuration (tables)
  - ASW management (lifecycle, upgrade)
  - Devices management (drivers, redundancies)

Software Projects

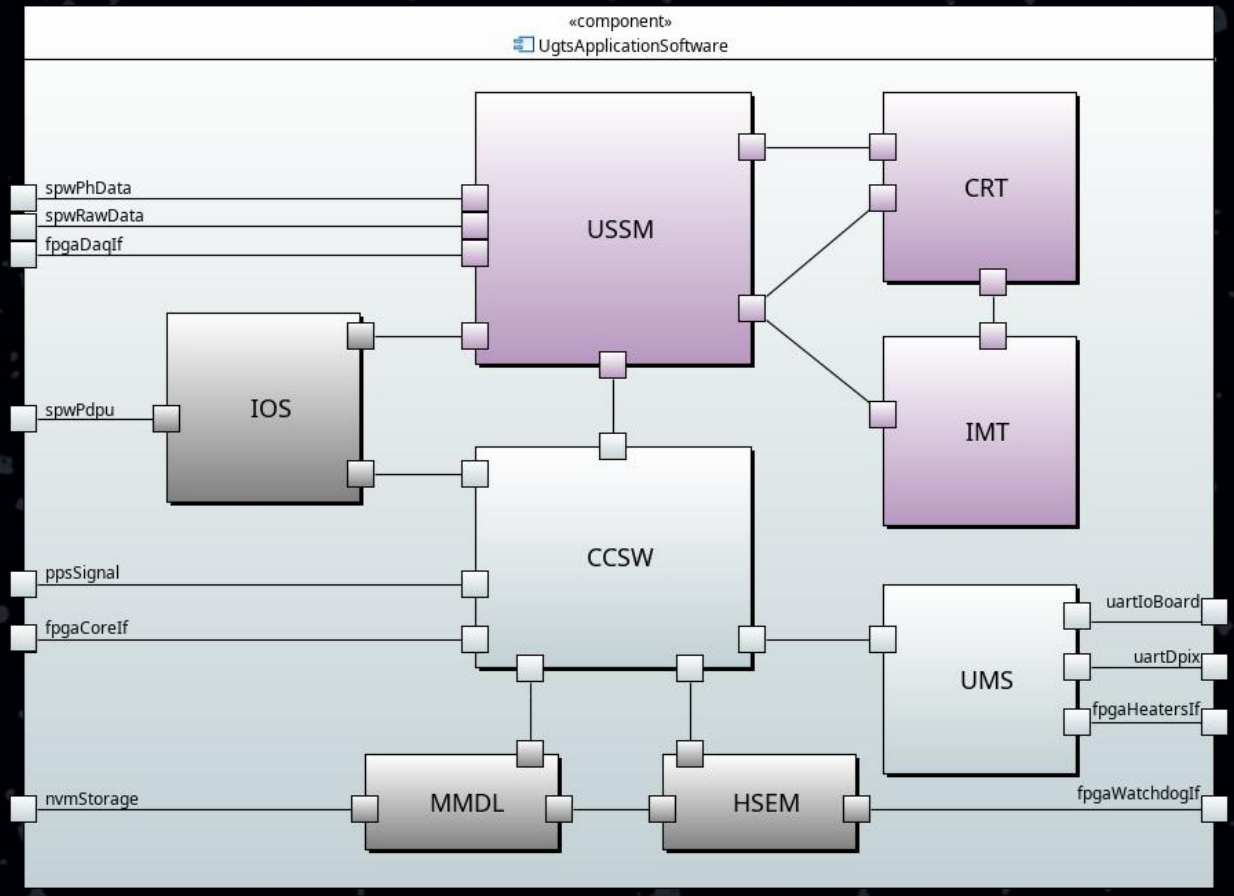
Solution → XtratuM + LVCUGEN + Boot JUICE

- Over 2016-2022
  - LVCUGEN, not yet KOSMOS
  - Not yet XNG, XtratuM v2.1.5
- European software
  - OK to export within instrument to China
- Run isolated programs → partitions
  - ASW can mix critical functions with less robust code
- Handle needed usual ASW + Boot facilities
  - Given some configuration / adaptations
- Available for our platform (GR712)
- Mature enough
- Small footprint

Development Teams

- ECLAIRs ASW → CEA Irfu + reuse
  - Dev. Phase → 4-10 people 2016-2022
    - Including for test-bench, validation, tools, quality
  - Pre-design study → ~2 people 2007-2015
  - Commissioning, operations → ~2 people 2023-…
- CNES + CS Group (+LAL)
  - Common modules + Boots (ECLAIRs + MXT)
    - MXT = SVOM second French instrument
  - MXT command-control
  - MXT scientific partition → LAL (IJCLab)

Partitioning



XtratuM hypervisor

- Dual-core mode
- Partitions: 7 bare-metal + 1 LithOS (MMDL)
- MMU and virtual memory addressing

Partitions

- New CEA developments (C++ 17)
  - USSM: Science management (DAQ, alerts…)
  - CRT & IMT: Scientific computing (Triggers)
  - UMS: Detector management
- LVCUGEN reuse (C 99)
  - CCSW: Coordination, usual ASW facilities
  - IOS: SpaceWire TM/TC link
  - MMDL: Storage/memory access, partitions lifecycle
  - HSEM: FDIR and scrubbing

Storage

Reused partitions → XM + LVCUGEN + CCSW

- 627 KiB → 832 reserved
  - Reasonable storage requirement
  - Could be improved
    - MMDL, HSEM, IOS, CCSW built unoptimized (-O0)
    - Regressions to fix with -O2

CEA partitions → USS + UMS

- USS binary → 592 KiB / 1024
  - Single binary for 3 science partitions
    - CRT, IMT, USSM
  - Avoid symbols duplication
  - Need virtual memory addressing
- UMS: 77 KiB / 192

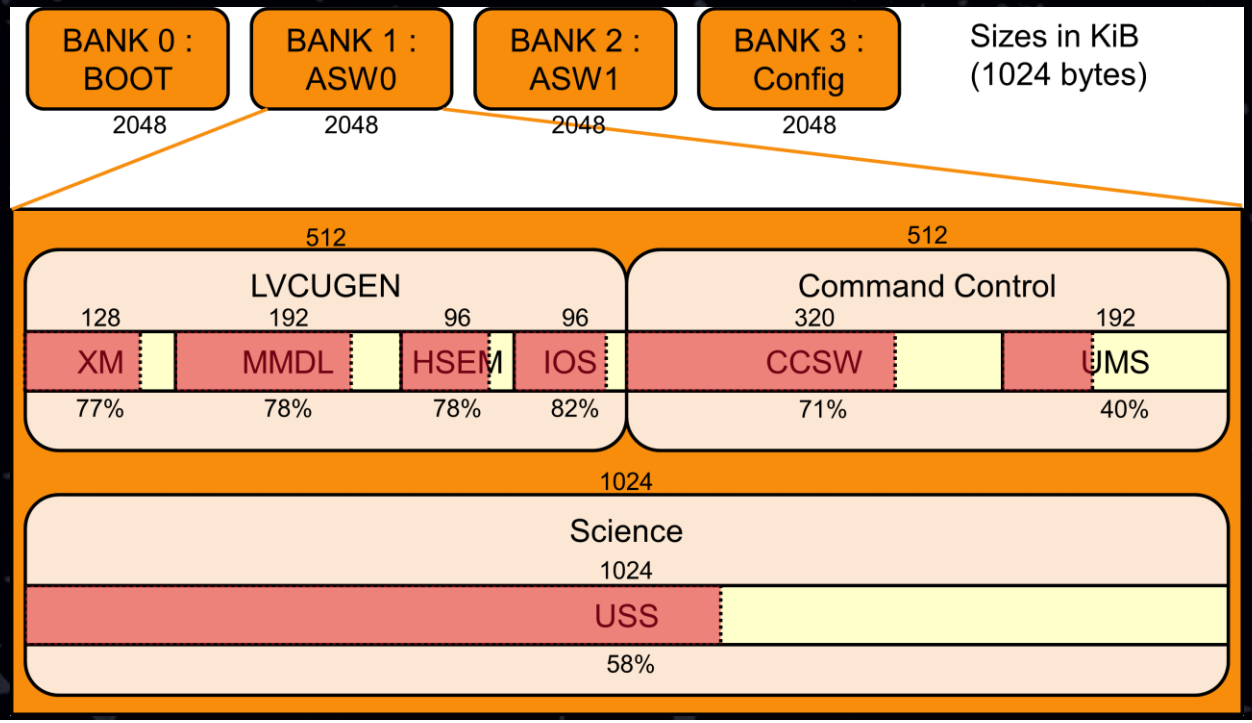
Total

- 8 partitions, 12 binaries, total size : ~1.3 MiB

In-flight update by CCSW

- PUS 222 → partitions
- PUS 220 → tables
  - Upload new ASW or tables during observation

MRAM storage subdivision



Scheduling

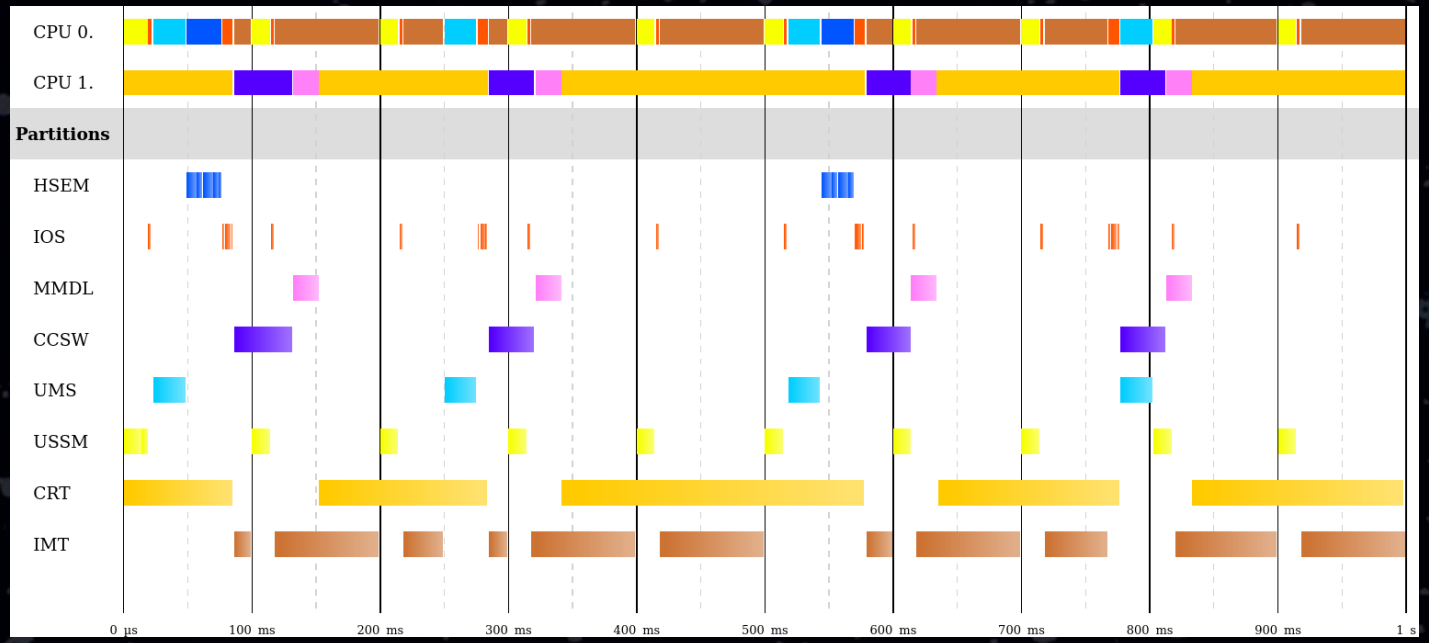
Bare-metal scheduling model in XM

- Alternative: virtualized OS… (LithOS/RTEMS)
- Each partition → N timeslices “slots”
  - Cycle with 1 second period
- Main thread: resume at next slot
  - Background tasks: long, event-loops…
  - Soft deadlines or manual scheduling
- Start-of-slot IT handler
  - Interrupt main thread
- “Task-slot model”
  - Each slot → few designated tasks
  - Hard deadlines → must finish before end-of-slot
    - Or “slot overrun”
  - LVCUGEN adheres to this model

ECLAIRs constraints

- Task-slot model rarely used
  - Long / variable-duration tasks
    - Scientific processing: can span over 20 seconds
    - Detector configuration (UART): up to 90 seconds / TC
- Max CPU time for science processing
  - Reduce GRB alert delay
  - Optimize UART (in UMS) → complex driver
    - UART IP-core of GR712 → unfit for XM
      - Tiny 8-byte buffer, no DMA → inefficient
  - Reduce context switches
    - 14x IOS → Merged activities, but design limitations
    - 10x USSM → Hard real-time requirements (SpW driver)
  - Adjust LVCUGEN slots
  - Dual-core balancing

Scheduling



Partition	Core 0	Core 1	Slot count
HSEM	5.4%		2
IOS	6.9%		14
MMDL		8.2%	4
CCSW		15.3%	4
UMS	10.3%		4
USSM	15.1%		10
CRT		76.5%	5
IMT	62.3%		11
Total	100.0%	100.0%	51

CPU-time consumed by reused partitions

- ~18% total (with 2x80 MHz, caches ON, -O0)
- WCET → Reserved but largely unused

LVCUGEN Integration

ECLAIRs ASW : forked from early MXT ASW

- SVOM → 2 French ASW → ECLAIRs + MXT
- Very similar hardware (CPU and IO boards)
- Development started first by CS Group
  - XM + LVCUGEN + CCSW for MXT
    - Reused, progressively adapted
  - Detector management + science partitions for MXT
    - Reimplemented

Reconfiguration

- IOS, MMDL, HSEM, XM, LibPUS → config system
  - static → header #defines, compilation-time
  - dynamic → distinct binary, data-only, runtime loading
  - No need to touch their code

LVCUGEN Configuration

- IOS: Adapt SpaceWire activities
  - High-speed science output
    - 5 Mbps: 60 TX packets @ 10 Hz
  - Many low-speed inputs / outputs
    - TC, HK, Science-HK, AAV-PVT, Dumps, VHF Alerts…
    - Periodic or random | limits per second
  - Solution: TX-RX-TX pattern on single slot @ 4 Hz → optimize number of slots
    - Dynamic routing → handle several packet types
    - Round-robin → 4 pre-defined priority sequences → prevent starvation, enforce limits

- HSEM + CCSW → New FDIR strategy
  - Recover from CCSW and UMS partition failure
    - Detect unresponsive
    - Partition warm restart
    - Other partitions keep running
    - Fresh partition instance re-integrates the flow

- MMDL, HSEM, XtratuM
  - Adapt memory layout, partitions list, PUS 5 events
  - Add ECLAIRs partitions and ports

- LibPUS
  - PUS services (3, 5, 12, 19, 21) configuration
    - Python code generator from Instrument DB

LVCUGEN Modifications

- MMDL
  - ECLAIRs-specific support for virtual memory
- CCSW
  - Not generic → no config or driver like IOS…
    - Requires code modifications
  - TC for new services
    - Routing, ACK and delegation to CEA partitions
  - Modes state-machine
    - Per-mode TC authorization
      - Generated from Instrument DB
  - Housekeeping
    - Datapool, marshalling lists, retrieval
  - PUS Service parameters
    - APIDs , Event-reports, HK, Actions, Tables, MemIDs, …
- Build system → reimplemented
  - Autoconf/automake instead of scripts/makefiles
    - Not mandatory, but more comfortable
  - Build all with a single command
  - Minimal rebuild (global dependency tracking)
  - Parallel builds (can use 10-20 CPU cores)
  - Out-of-source build
    - Source-dir ≠ Build-dir(s) ≠ Install-dir(s)
    - Keep multiple builds (≠ options) on same sources
  - Modifications
    - LVCUGEN build: simple → reimplemented
    - XM build: too complex → kept and integrated
    - Change source tree layout
    - Use more recent compiler (BCC 2), except for XM

Advantages of LVCUGEN

- Feature-rich
  - Quickly build-up an ASW
  - Handle most payload-generic tasks
    - Partitions load, start and stop
    - In-flight code upgrades
    - SpW link + TM/TC dispatch
    - Onboard Time
    - Modes
    - Observability
    - Actions and sequences
    - FDIR
    - Memory scrubbing
    - Boot Software (now in KOSMOS)
  - Help focus on instrument-specific code
    - Or enhance operability & standardization for similar effort
- Robust
  - Mature, flight-proven
  - Helps guaranteeing schedulability
    - XM / Xoncrete / task-slot model
  - Strong partitions isolation
    - MMU
    - Time determinism
  - Recover from partition failure
- Configurable
  - Static and dynamic configurations
  - No need to modify IOS, HSEM, MMDL, XM, libPUS
- Lightweight
  - Modest storage, memory and computing allocations

Limits of LVCUGEN

Aspects that were underestimated:

- Customization effort
  - « An » ASW is not « Our » ASW !
  - CCSW is the heart of the ASW and is NOT generic
    - Required a deep understanding of the code
  - Complex system
    - Many large and scattered configuration files, some duplication
      - Eg: duplicate partition-lists, event-reports lists
    - Low-abstraction C code
  - Learning & adaptation
    - Some unfamiliar tools, build, code style, designs, etc.
- Overheads
  - Partitioning + task-slot model
    - CPU underutilization
    - Higher context switching & cache miss
    - Additional copies
    - No real-time IRQ serving
  - ➢ Performance penalty
  - ➢ Higher engineering effort
- Validation effort
  - No help for documentation / validation
  - SRS / SVS : as if we wrote the code ourselves
    - Got data-pack but late and partition-isolated specs/tests
    - Hundreds of specifications and test scenarios to write
      - Just to cover reused code
  - No reusable & automated test scripts
    - BVS: arrived too late (and ECLAIRs-specific)
  - We developed a test-bench + validation suites
    - Huge effort: ~14 man.year
  - No tool for WCET analysis
    - LICE traces visualization & analysis tool developed
    - 1 part-time apprentice over 3 years

Possible Improvements

- CCSW / LibPUS
  - Generic code generators from Instrument DB
    - Packet structure mapping, serialization / deserialization
    - Lists: events, monitoring, event-actions, TC-sequences
    - Event-report factory
    - TC dispatcher
    - Modes state-machine
  - Plugin model, like drivers
    - Integrate generated code
    - Integrate custom code
    - ➢ Eliminate necessity to edit CCSW code
- XM
  - *hypervisor-space* driver architecture
    - XM plugin → like kernel-space drivers in Linux…
    - ➢ Benefit: real-time IRQ handling in the driver
- IOS: true asynchronous model
  - Enqueue several RX and TX operations
    - Without waiting for completion of previous one
    - Poll for results later
    - Send/receive many packets per slot
      - Parameterize limits per slot
  - Less ties to hardware limits
    - Eg: 60-TM batch because 64 TX descriptors on Leon3/4
  - ➢ Would reduce context switching
- MMDL
  - Generic support for virtual memory
  - Queue of asynchronous requests
    - ➢ Maximize throughput, less context switches

Possible Improvements

- Provide Execution & Validation package
  - With specifications & validation scenarios
    - For re-validation of KOSMOS code, at ASW-level
    - Easily importable in most ALM tools
  - With generic simulation environment & test-bench
    - Simulated test-bench, like BVS
      - Support for popular CPU-platforms for space
      - Simplified or complete instrument HW simulation
    - Test suites
      - Implementation of KOSMOS test scenarios
  - With tools for WCET and traces analysis
    - LICE or other solutions
    - Also code coverage…
- Higher software-engineering standards
  - Have a modern build system → CMake or Meson
  - Have a Git repository for each component
  - CCSW code: more encapsulation and factorization, no entities named in French
  - Fix issues with –O2, enable stricter warnings
  - Availability on recent compilers/platforms

Thank you for your attention !

Any questions ?