Enabling Linux Usage in Space Applications ?

Airbus Defence and Space

Antoine Certain
Embedded Linux Conference

Agenda

- **Space Industry Context**
- Architecture exploration
- Target Selection
- Technical overview
- What’s next ?

AIRBUS

Space Industry Context

Environmental Constraints:

- Radiations
- Energy
- Mechanical and thermal

- **Radiations issues**
  - Destructive effects, transient effects, cumulated effects
  - Manage thanks to faults tolerant chips and systems
  - But poor electronics devices, lower processing performances and costly characterization and qualification

- **Energy issues**
  - Solar Energy only
  - Becomes rare when far from the Sun
  - Unpredictable on Planetary surfaces

- **Mechanical and Thermal constraints**
  - Vacuum and thermal variations
  - Mass limitations
  - Extreme and variable operational conditions
    - Assembly Integration and Tests
    - Ground, air and sea Transport
    - Launch
    - Orbital LEO short night/day cycles, GEO, Deep Space

Space Industry Context

Technical Constraints

Industrial Constraints

- **Technical**
  - Time and Synchronisation with specific hardware
  - Performances (agility, instruments data processing) with low power processors
  - Communication (bandwidth availability, various path, security)
  - On board storage with specific hardware
  - Maintainability for years
  - Safety and Security both matters
- **Industrial**
  - Variety of missions, difficult to promote a complete product reuse
  - Looking for European independence
  - System Testability very difficult and expensive
  - Rigorous standards for development and manufacturing processes (ECSS)
  - Component obsolescence
  - Safety and Security both matters

AIRBUS

Focus on ECSS criticity level

Category	Definition
A	Software that if not executed, or if not correctly executed, or whose anomalous behaviour can cause or contribute to a system failure resulting in: → Catastrophic consequences
B	Software that if not executed, or if not correctly executed, or whose anomalous behaviour can cause or contribute to a system failure resulting in: → Critical consequences
C	Software that if not executed, or if not correctly executed, or whose anomalous behaviour can cause or contribute to a system failure resulting in: → Major consequences
D	Software that if not executed, or if not correctly executed, or whose anomalous behaviour can cause or contribute to a system failure resulting in: → Minor or Negligible consequences

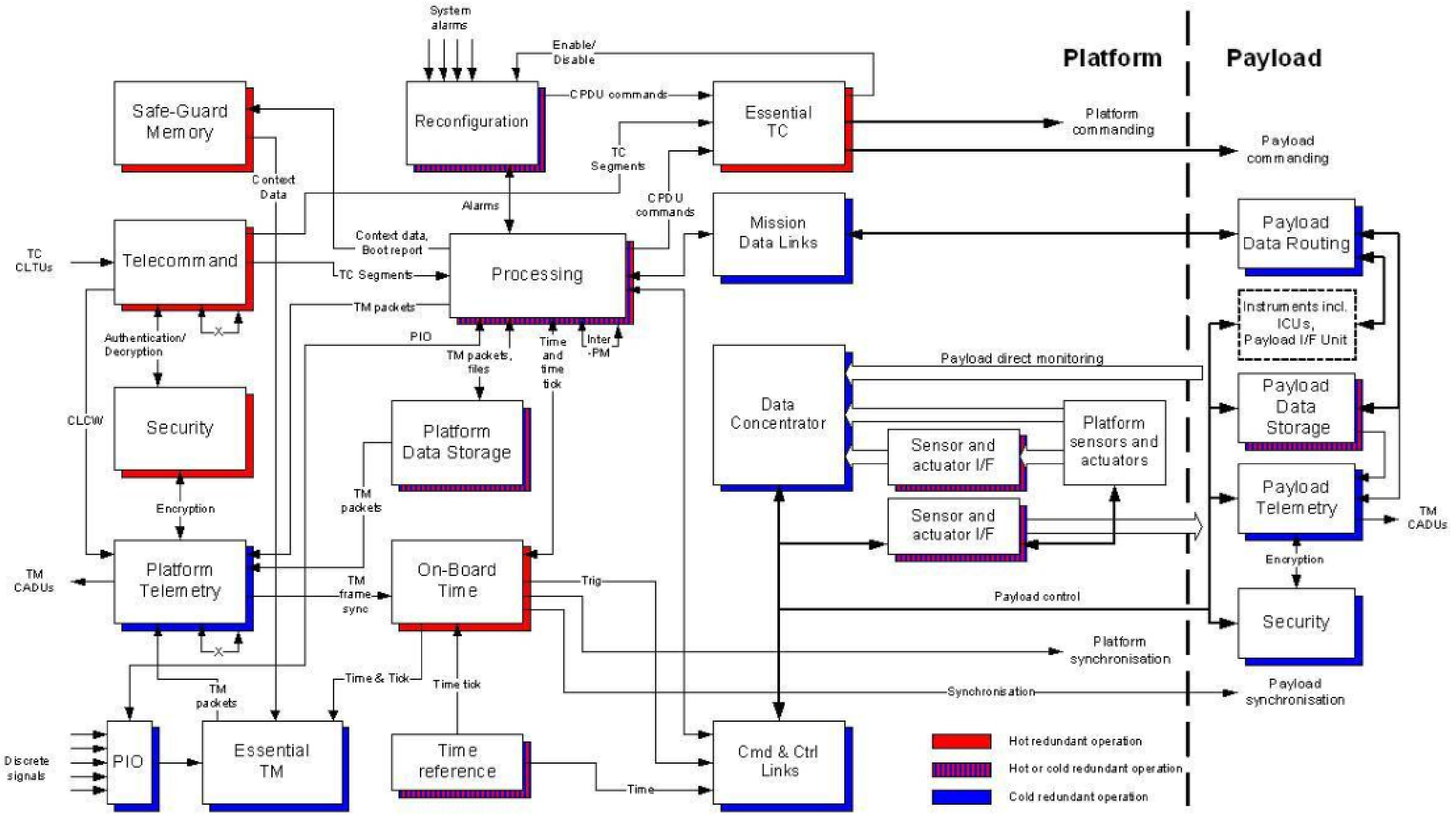
More or less as DO-178

Define validation and verification effort

For most satellites :

- Boot category B
- Applicative software category C

Functional overview of an On Board Computer



SAVOIR
space avionics open interface architecture

esa

Satellite Function

Processing Unit

Data storage

Reconfiguration unit (redundancy)

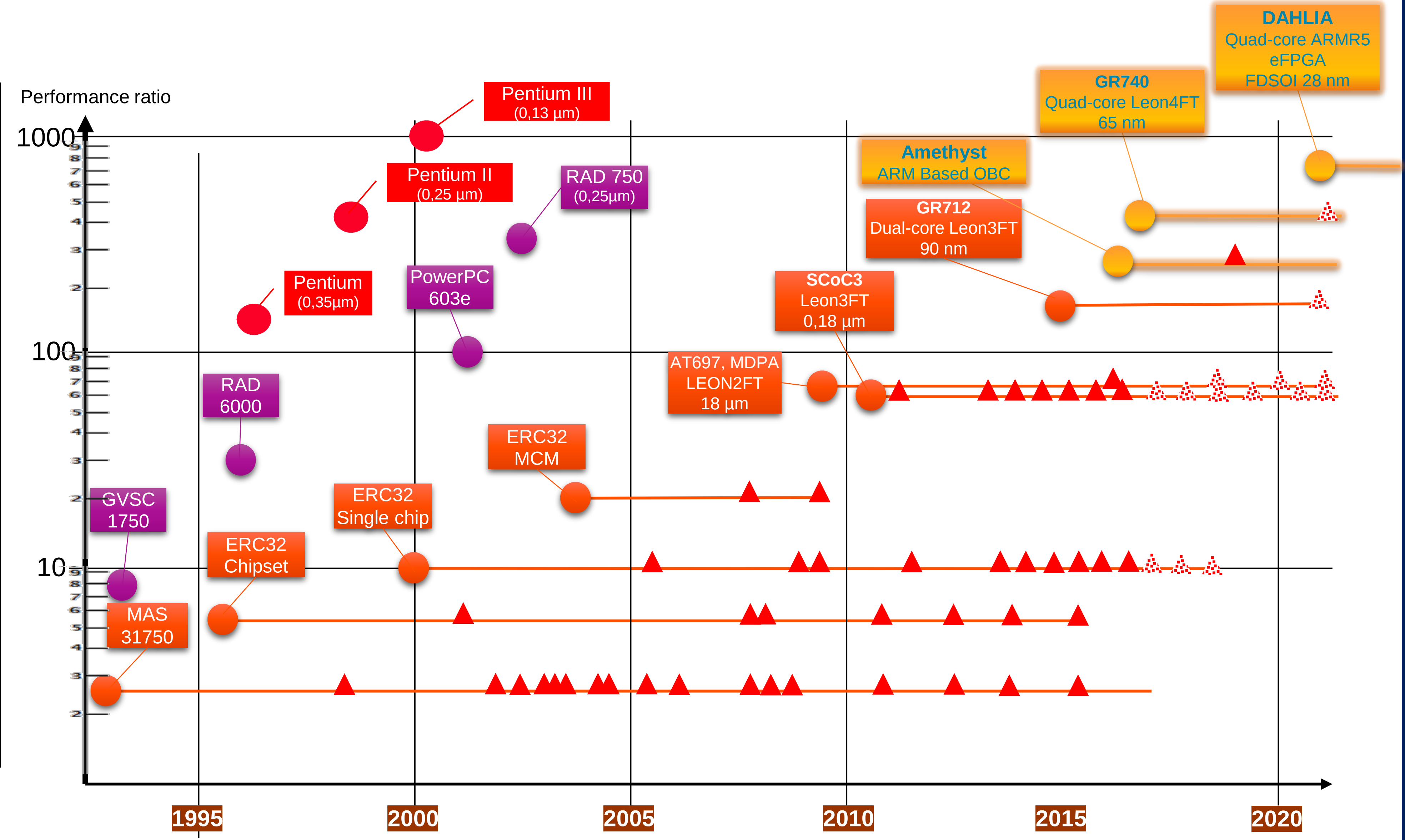
Security Unit

Time Unit

Data switch

....

AIRBUS



Processors in Space

LEON 3/4

ARM R5 (MPU)

ARM R52 (MPU)

AIRBUS

Sum up space context

- **Hardware is difficult to manage**
  - Environmental constraints
  - Always very specific (niche market)
  - Very expensive and hard to test
  - Software is the only leverage to correct late bugs
  - Not very powerful

- **Evolution of the needs**
  - In order to reduce costs, the functionality has to be gather into one electronic device
  - Integration of black box software
  - New requirements about autonomy, reduce downlink

- **Safety and security**
  - Safety is a crucial part of the development (ECSS)
  - Until now, security is mainly about downlink
  - But it is changing since the black box integration

AIRBUS

Why using Linux ?

- **State of the art framework (with or without HW support) :**
  - Artificial Intelligence
  - Software defined radio
  - Image/Video Processing

- **Linux Ecosystem**
  - Provides lot of powerful packages
  - Avoid to reinvent the wheel
  - Ease to develop (many powerful tools)

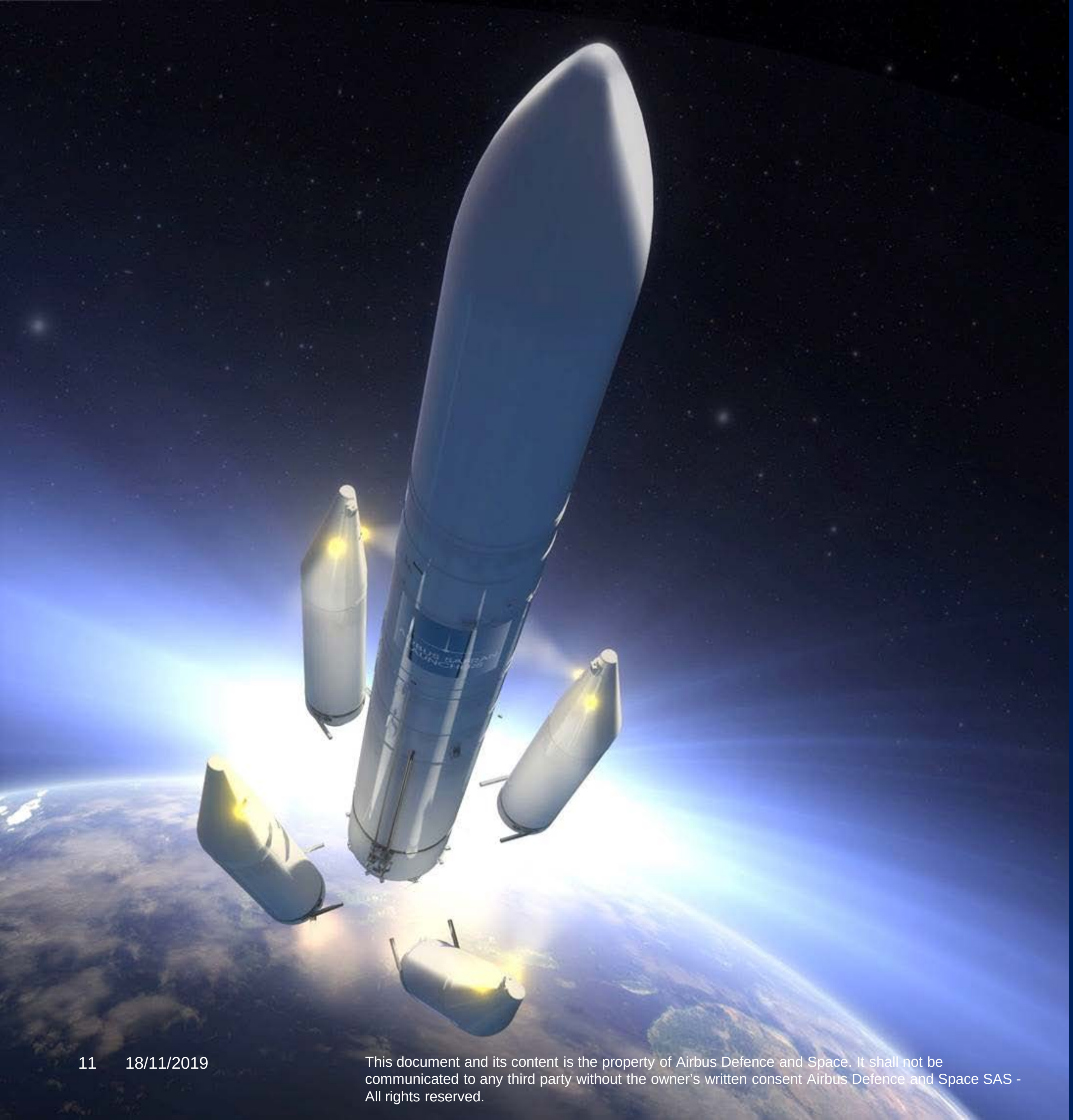
- **Portability**
  - ARM, LEON, RISC V Architecture
  - Easy image generation thanks to YOCTO, buildroot

- **Community**
  - World wide spread community supported by Linux Foundation
  - Many developers on market including fresh grad

Which are our issues with Linux usage ?

- **Technical issues**
  - Real time usage
  - Which packages to use ? On which criteria ?
  - Our systems have low capacity memories
  - Our ARM systems are MMUless

- **Industrial Issues**
  - ECSS and qualification
  - Criticality level
  - SDE update
  - License issues
  - Constant evolution



Agenda

- Space Industry Context
- **Architecture exploration**
- Target Selection
- Technical overview
- What’s next ?

Architecture exploration

Simple architecture

Allowing exploration of mixed criticality concept on HMP device

Without Hypervisor ? More interest in HW capabilities

Real time master application :
- Boot/reboot Linux software
- Monitor correct execution of Linux
- At least, tolerant to radiations
- TCM used as working memory

Messages exchanges
Power management

Linux slave application :
- Execute mission
- Watchdog
- Low criticity level

AIRBUS

Agenda

- Space Industry Context
- Architecture exploration
- **Target Selection**
- Technical overview
- What’s next ?

AIRBUS

What we need at HW level ?

Real time core(s)
Deterministic
Dedicated memory access
-> Dedicated to safety critical functionalities

Applicative cores
Rich OS enabled
Fast memory access
Memory Management Unit
L1, L2 caches
-> Dedicated to mission handling

SOC Management Unit
Power management
Time management
Debug support Unit with traces
DMA management
Security Management
Reconfiguration Management

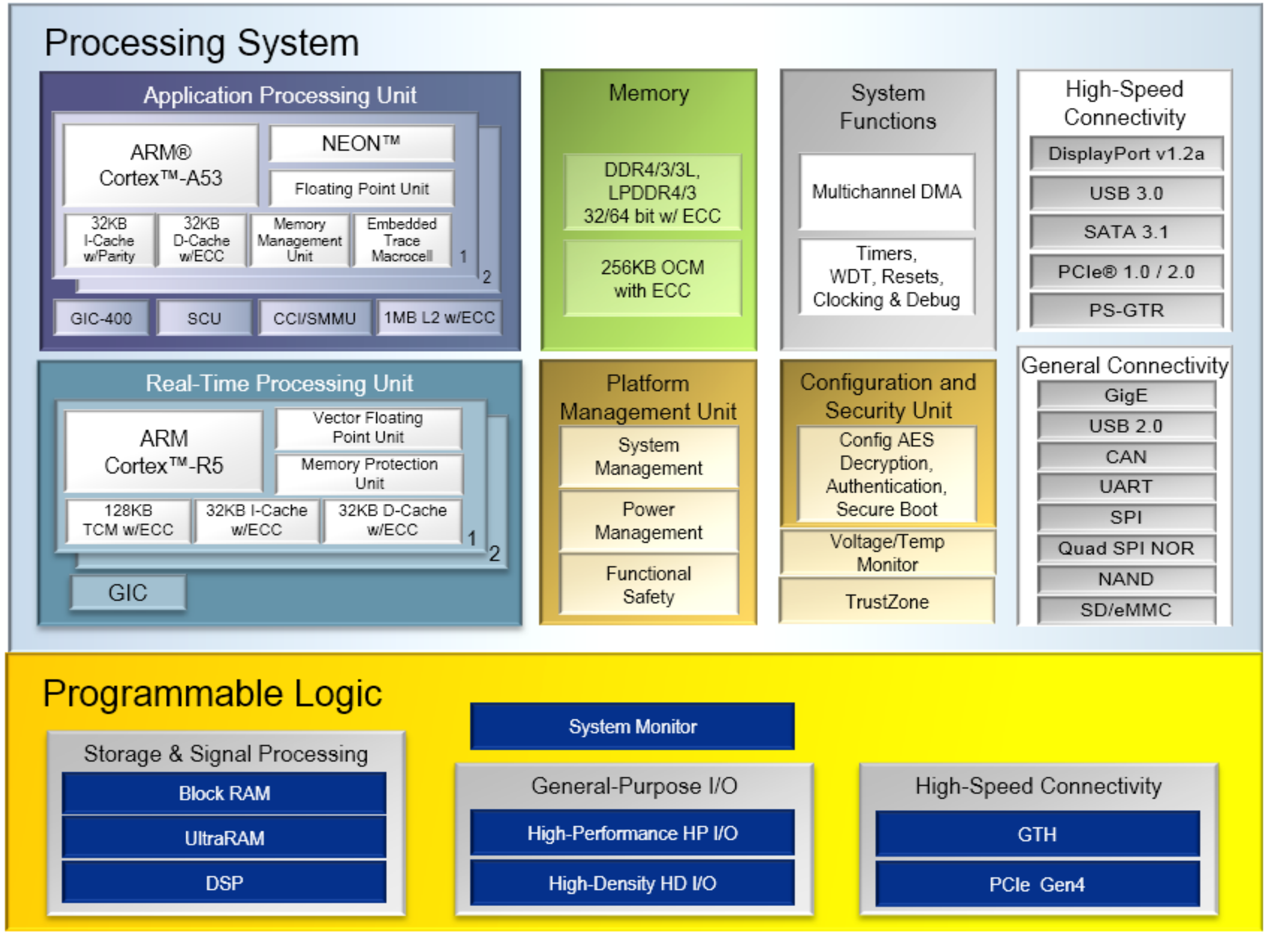
Network on Chip (aware of interferences)

Memories Controller
Nor Flash
Nand Flash
DDR, SRAM

Hardware accelerator
For complex algorithms
(GPU, FPGA, ManyCores)

IO and eFPGA
Ethernet (TSN), Spacewire, legacy interfaces

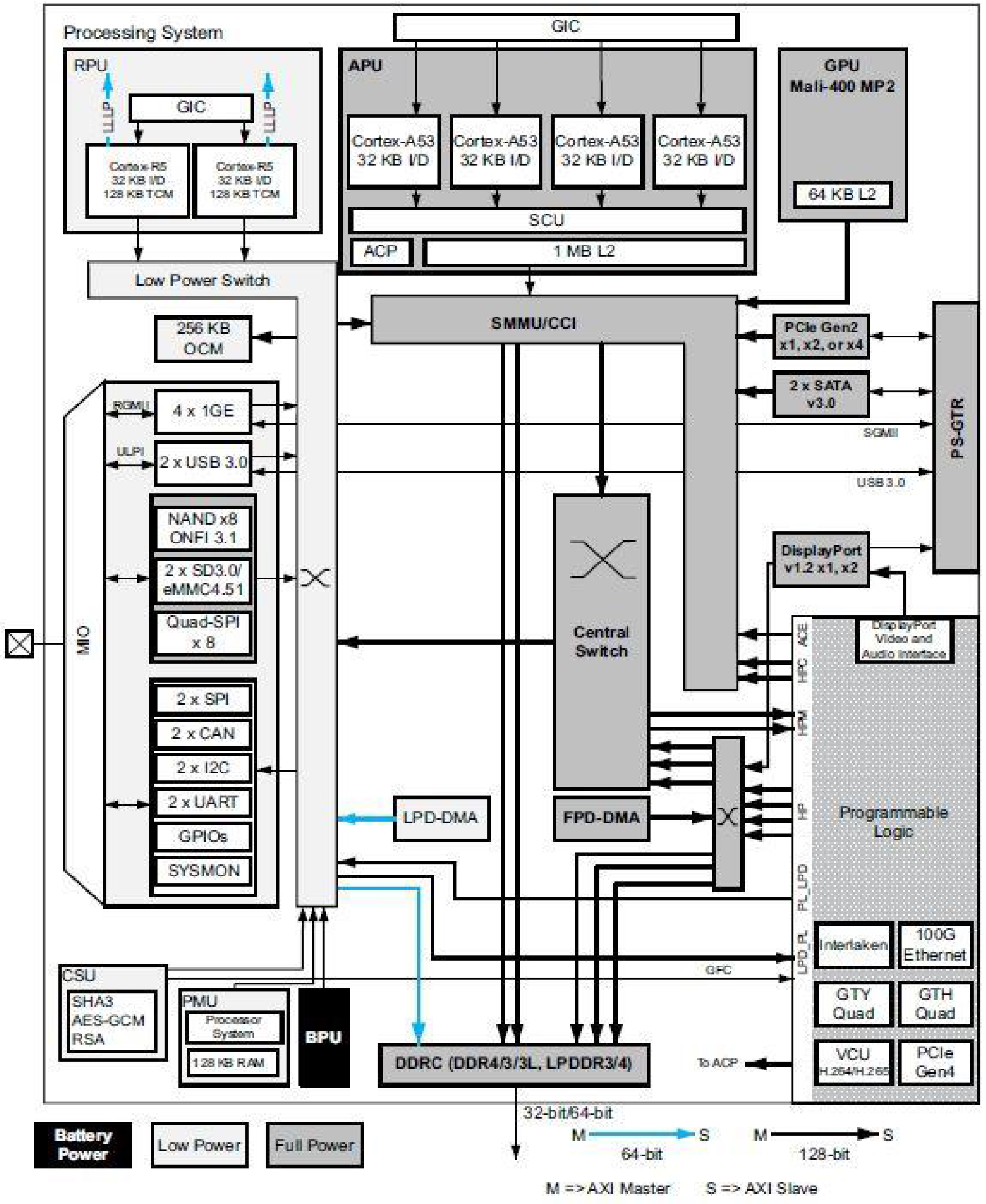
AIRBUS



Zynq Ultrascale plus

Global presentation :

- Heterogeneous Multi Processing
- Embedded FPGA
- Complete set of I/O



Zynq Ultrascale plus

More detail architecture :

- PMU : Power Management Unit
- CSU : Configuration Security Unit (secure boot)
- RPU : Real time Processing Unit (Cortex R5 lockstep)
- APU : Applicative Processing Unit (Cortex A53)

What about the SW architecture on ZUP+?

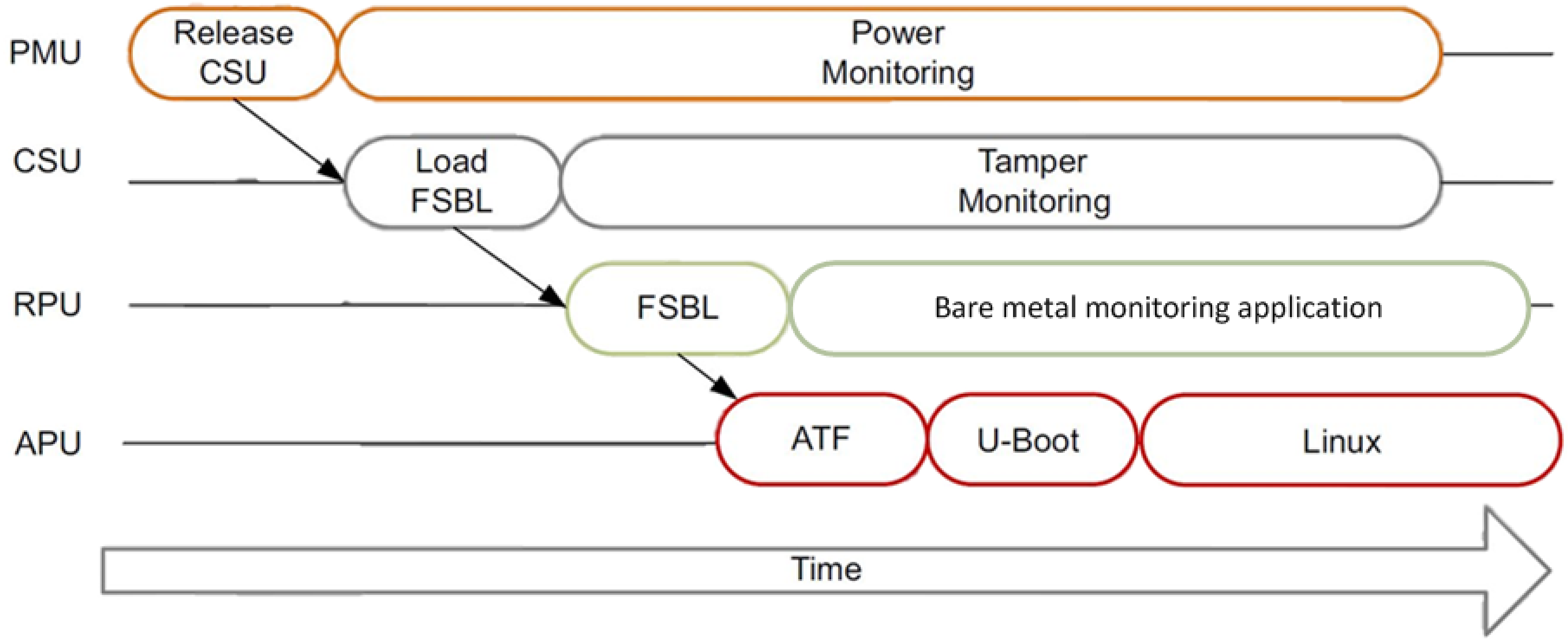
- **Advantages**
  - Easy to implement solution
  - First step in mixed criticality
  - Matches with the need to reduce development costs
  - R5 in lockstep mode and support transient radiations errors
  - Few bus interactions between R5 and A53 since TCM is used for code execution on R5

- **Drawbacks**
  - Difficulties to manage integration at system level
  - Do not ensure that all the Linux software correctly behave
  - Simple real time application, in real world certainly more complex
  - Based on hardware and more difficult to port

Agenda

- Space Industry Context
- Architecture exploration
- Target Selection
- **Technical overview**
- What’s next ?

AIRBUS



Boot sequence

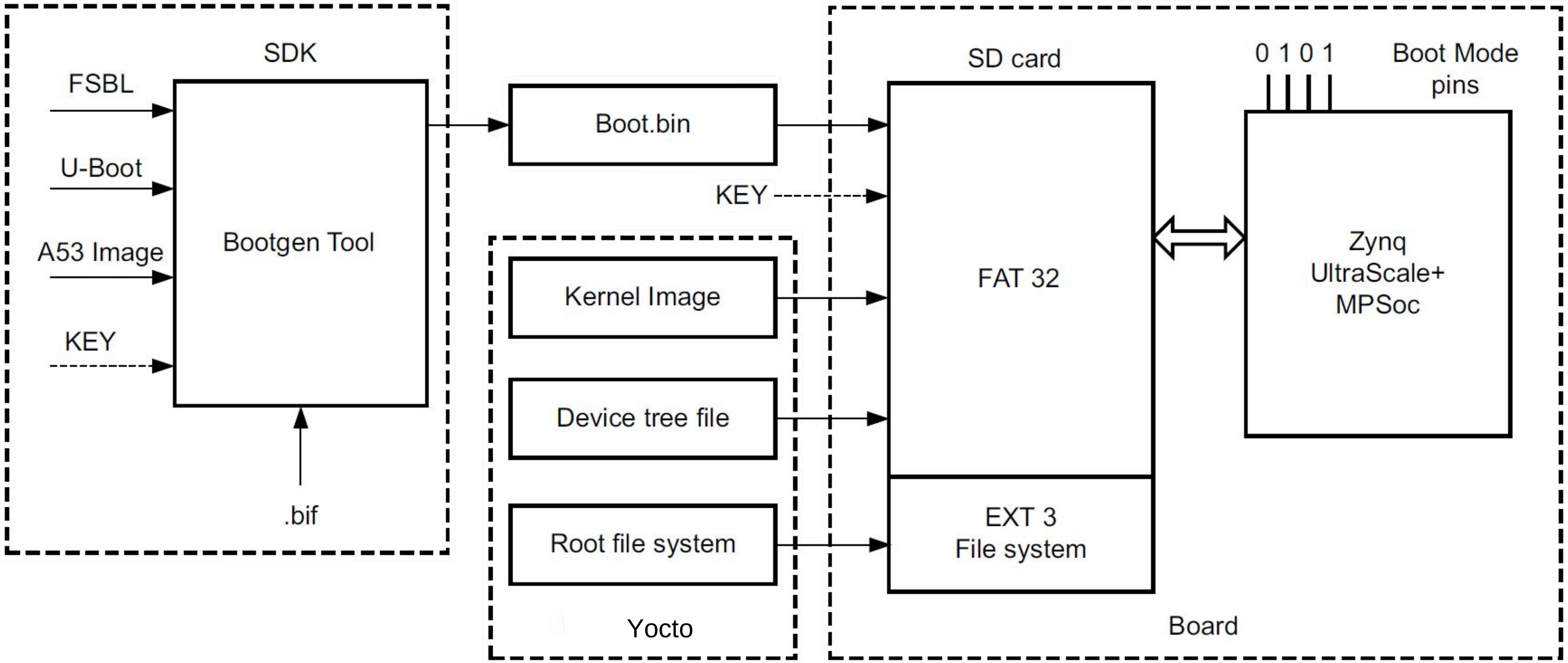
Several stages boot

FSBL : Loading the bitstream and configuring PS

ATF : Trusted Firmware in order to secure APU

AIRBUS

Boot build on SD card



Boot SD

Boot build thanks to xilinx SDK

Boot partition :

- FSBL
- U-Boot
- Kernel
- Device tree

Slave Name	Size	Start Address	End Address
CSU_RAM	32 KB	0x00FFC40000	0x00FFC47FFF
CSU_ROM	128 KB	0x00FFC00000	0x00FFC1FFFF
EFUSE	64 KB	0x00FFCC0000	0x00FFCCFFFF
PMU_ROM	256 KB	0x00FFD00000	0x00FFD3FFFF
PMU_RAM	128 KB	0x00FFDC0000	0x00FFDDFFFF
OCM_RAM	256 KB	0x00FFFC0000	0x00FFFFFFFF
R5_0_ATCM_SPLIT	64 KB	0x00FFE00000	0x00FFE0FFFF
R5_0_BTCM_SPLIT	64 KB	0x00FFE20000	0x00FFE2FFFF
R5_0_ICACHE	64 KB	0x00FFE40000	0x00FFE4FFFF
R5_0_DCACHE	64 KB	0x00FFE50000	0x00FFE5FFFF
R5_1_ATCM_SPLIT	64 KB	0x00FFE90000	0x00FFE9FFFF
R5_1_BTCM_SPLIT	64 KB	0x00FFEB0000	0x00FFEBFFFF
R5_1_ICACHE	64 KB	0x00FFEC0000	0x00FFECFFFF
R5_1_DCACHE	64 KB	0x00FFED0000	0x00FFEDFFFF
R5_0_ATCM_LSTEP	128 KB	0x00FFE00000	0x00FFE1FFFF
R5_0_BTCM_LSTEP	128 KB	0x00FFE20000	0x00FFE3FFFF

Shared memories

How to manage corruption due to radiations ?

How to manage standard shared memories issues ? (caches coherencies, synchronization,..)

On ZUP, two choices :

TCM : working memory of the R5 lockstep

OCM : Used by trusted firmware

AIRBUS

Watchdog

Linux shall refresh watchdog regularly

If not, real time application assumes that applicative software is stuck

Then reboot Cortex A53

1. Power off Cortex A53 thanks to inter-processors interruptions via PMU
2. Trusted firmware and U-boot loaded in RAM
3. Power on Cortex A53 thanks to inter-processors interruptions via PMU

AIRBUS

Agenda

- Space Industry Context
- Architecture exploration
- Target Selection
- Technical overview
- **What’s next ?**

What's next ?

- **First step on mixed criticality on Heterogeneous devices**
  - It is not a solution
  - Work done to oversee difficulties with this kind of device
  - Memories is one of the weakness of this solution on this target
- **PhD in collaboration with ONERA**
  - To study these safety problematics
  - Put these issues at System Level
  - To watch out and maybe contribute to ELISA
- **Running out of time**
  - Increase of functional requirement
  - Reduce costs
- **Change mindset regarding collaboration and License**

Thanks you for your attention