Agenda

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

Environmental Constraints:

- Radiations
- Energy
- Mechanical and thermal

Radiation 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
### Focus on ECSS criticality level

More or less as DO-178

Define validation and verification effort

For most satellites:

- Boot category B
- Applicative software category C

<table>
<thead>
<tr>
<th>Category</th>
<th>Definition</th>
</tr>
</thead>
</table>
| 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 |
Functional overview of an On Board Computer
Processors in Space

LEON 3/4
ARM R5 (MPU)
ARM R52 (MPU)
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 gathered 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
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

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

Messages exchanges
Power management
Agenda

• Space Industry Context
• Architecture exploration
• Target Selection
• Technical overview
• What’s next?
### What we need at HW level?

<table>
<thead>
<tr>
<th>Category</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Real time core(s)</strong></td>
<td>Deterministic&lt;br&gt;Dedicated memory access&lt;br&gt;-&gt; Dedicated to safety critical functionalities</td>
</tr>
<tr>
<td><strong>Applicative cores</strong></td>
<td>Rich OS enabled&lt;br&gt;Fast memory access&lt;br&gt;Memory Management Unit&lt;br&gt;L1, L2 caches&lt;br&gt;-&gt; Dedicated to mission handling</td>
</tr>
<tr>
<td><strong>SOC Management Unit</strong></td>
<td>Power management&lt;br&gt;Time management&lt;br&gt;Debug support Unit with traces&lt;br&gt;DMA management&lt;br&gt;Security Management&lt;br&gt;Reconfiguration Management</td>
</tr>
<tr>
<td><strong>Network on Chip (aware of interferences)</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Memories Controller</strong></td>
<td>Nor Flash&lt;br&gt;Nand Flash&lt;br&gt;DDR, SRAM</td>
</tr>
<tr>
<td><strong>Hardware accelerator</strong></td>
<td>For complex algorithms&lt;br&gt;(GPU, FPGA, ManyCores)</td>
</tr>
<tr>
<td><strong>IO and eFPGA</strong></td>
<td>Ethernet (TSN), Spacewire, legacy interfaces</td>
</tr>
</tbody>
</table>
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?
Boot sequence

Several stages boot

FSBL : Loading the bitstream and configuring PS

ATF : Trusted Firmware in order to secure APU
Boot build on SD card

Boot SD

Boot build thanks to xilinx SDK

Boot partition:
- FSBL
- U-Boot
- Kernel
- Device tree
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

<table>
<thead>
<tr>
<th>Slave Name</th>
<th>Size</th>
<th>Start Address</th>
<th>End Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSU_RAM</td>
<td>32 KB</td>
<td>0x00FFC400000</td>
<td>0x00FFC47FFF</td>
</tr>
<tr>
<td>CSU_ROM</td>
<td>128 KB</td>
<td>0x00FFC000000</td>
<td>0x00FFC1FFF</td>
</tr>
<tr>
<td>EFUSE</td>
<td>64 KB</td>
<td>0x00FFCC00000</td>
<td>0x00FFC0FFF</td>
</tr>
<tr>
<td>PMU_ROM</td>
<td>256 KB</td>
<td>0x00FFD000000</td>
<td>0x00FFD3FFF</td>
</tr>
<tr>
<td>PMU_RAM</td>
<td>128 KB</td>
<td>0x00FFD000000</td>
<td>0x00FFDFFF</td>
</tr>
<tr>
<td>OCM_RAM</td>
<td>256 KB</td>
<td>0x00FFC000000</td>
<td>0x00FFDFFF</td>
</tr>
<tr>
<td>R5_0_ATCM_SPLIT</td>
<td>64 KB</td>
<td>0x00FFE000000</td>
<td>0x00FFE0FFF</td>
</tr>
<tr>
<td>R5_0_BTCM_SPLIT</td>
<td>64 KB</td>
<td>0x00FFE200000</td>
<td>0x00FFE2FFF</td>
</tr>
<tr>
<td>R5_0_ICACHE</td>
<td>64 KB</td>
<td>0x00FFE400000</td>
<td>0x00FFE4FFF</td>
</tr>
<tr>
<td>R5_0_DCACHE</td>
<td>64 KB</td>
<td>0x00FFE500000</td>
<td>0x00FFE5FFF</td>
</tr>
<tr>
<td>R5_1_ATCM_SPLIT</td>
<td>64 KB</td>
<td>0x00FFE900000</td>
<td>0x00FFE9FFF</td>
</tr>
<tr>
<td>R5_1_BTCM_SPLIT</td>
<td>64 KB</td>
<td>0x00FFE000000</td>
<td>0x00FFE0FFF</td>
</tr>
<tr>
<td>R5_1_ICACHE</td>
<td>64 KB</td>
<td>0x00FFE100000</td>
<td>0x00FFE1FFF</td>
</tr>
<tr>
<td>R5_1_DCACHE</td>
<td>64 KB</td>
<td>0x00FFE300000</td>
<td>0x00FFE3FFF</td>
</tr>
<tr>
<td>R5_0_ATCM_LSTEP</td>
<td>128 KB</td>
<td>0x00FFE000000</td>
<td>0x00FFE1FFF</td>
</tr>
<tr>
<td>R5_0_BTCM_LSTEP</td>
<td>128 KB</td>
<td>0x00FFE000000</td>
<td>0x00FFE1FFF</td>
</tr>
</tbody>
</table>
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
Agenda

- Space Industry Context
- Architecture exploration
- Target Selection
- Technical overview
- 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