[HV] Next Generation Multi-OS System Design Whitepaper

Based among other things on the idea presented at GENIVI AMM in Munich regarding VIRTIO use both with and without hypervisor, to communicate between multiple OSes, the need has been identified to describe the complexity of system design on modern heterogeneous multi-core SoCs, running several different Operating System kernel instances.

Proposed Document Outline

Focus: "**What HV technology can do for future automotive systems.**"

1 Motivation: Why to use HV:

1. Use of legacy systems with minor modifications,
    a. address what kind of modifications we expect,
    b. address problem of open source firmware and driver (MCAL) qualification when running virtualized drivers (see also section 3). HV helps with this by

1. isolating critical devices from non-critical ones, allows one to build systems with mixed-criticality w.r.t. safety-relevance.
2. qualified driver needs to come with the driver host or even the HV provider

1. Use of special purpose Guest/OS for isolating a specific functionality, i.e., building safety and security island
2. Sharing of HW in the presence of parallel system executions
3. Isolation properties in the presence of parallel systems executions:
    a. Spatial isolation, hiding of secrets
    b. Temporal isolation, implicit and explicit shared resources…

2 What does the HW need to fulfill to support unmodified execution of complete SW stacks.

This is directed towards the HV vendor to avoid the problems we have seen in the past.

3 Isolation and Partitioning

multiOS vs. HV

Timing and Spatial isolation

Different types of HV to support partitioning (see Gunnars list)

4 Inter-core communication

VM2VM

HV2VM

And HV-off partition to VMs

5 Sharing of physical devices

What does a HV need to do, to make its usage agnostic to Guest/OS:

Here we place the story on Virtio and generic interfaces

Title candidates

(depend on actual content)

Next Generation Multi-OS System Design

Multi-OS System Design on Multicore SoCs

What HV technology can do for future automotive systems.

Content Brainstorm

Intro. Why ? Motivation
-- consolidation of systems, "*mixed criticality*" requirements, e.g. security, safety, real-timedness.
This background is well know... Previous AGL white paper intro should cover this quite well, for example.
Generally try reuse, don't redo, and complement with what is missing.

Overall scope: **Multi-OS System design on heterogeneous multi-cores ?**
With and without hypervisor. Dedicated cores. Isolation possibilities for memory, peripherals, etc. Light-weight solutions.
Dedicated cores or Linux Containers.
Understanding what is what... must be done right. search and reuse LXC description.

Include MCU-style hypervisors. (*what does the word hypervisor mean here?*)

> Traditionally MCU classes were targeted for their use and position in the car such as body, powertrain and chassis and safety.
> To address the needs of zone-based electrical architectures these classes have begun to be consolidated. The requirement to isolate functionality with different safety or security requirements has meant these new MCUs may have support for h/w virtualisation. Examples: RH850/U2A series.

When hardware has support for isolation. TrustZone, but also other.
ARM input on this? Lots of ARM updates in the pipe... timing (can become obsolete quickly)
Describe at least High-level understanding.

- OS isolation
  - GPU OS isolation
    - Separate input ports
    - GPU scheduler ensuring pipeline for each OS is serviced
    - Memory separation
  - IOMMU (IPMMU) OS isolation
- Drawing isolation
  - Dedicated functional safety drawing paths, e.g. separate 2D rendering path for cluster telltales
- Bus Master / IP / Memory access isolation
  - Independent security and safety groups control access of IP on bus and memory protection
- Multi-level security isolation outside of common IP such as TrustZone
  - e.g. Implementation in real time R7 CPU
- Lifecycle Management
  - e.g. control of security at different stages of its life

System-level quality of service. ~~Future~~ Some architectures can guarantee QoS on internal interconnects and caches, etc. (what else?) controlled by VM configuration or h/w controls.

> Example of current architectures is support for QoS on AXI bus masters and GPU on R-Car.
>
> (*work needed to separate future from current architectures. MPEM future for example? Presumably other examples of current architectures be it SoC specific or IP*)
> – ref armv8 manual. MPEM Memory system resource performance....

Inter-*partition* protocols for example leveraging VIRTIO. (*Partitions* = VMs, to/from HV, but also between OSes running bare metal on dedicated core)

Hardware Device sharing - main purpose of Hypervisors. However device sharing can be set up with other means?

> Yes, examples:

- h/w display layers
- GPU virtualisaton with OS ID support

– Review "Adam's wish list" for silicon vendors... might be included in white paper as a result.
Risk of being outdated quickly since hardware is changing quickly
- Mostly HW features but also may include firmware requirements.

System design is unique – Not 100$ common hardware feature across all of them. But a small core of features are common.
Other hardware features can be mapped to functions of the system. E.g. if you need function X, then require hardware feature Y. Include optional and mandatory requirements.

The wishlist is likely to be more of principles than hard detailed requirements. In some areas analyze the more detailed requirements e.g. "X number of DMA channels"

Experiences that lead to requirements 64 bit support on all I/O masters...
.... Is this risking scope creep for the white paper?
Wish list might be better as independent.

Key topics: Partitioning and Spatial/Timing isolation of parts.

Goals

- Establishing common language/terms/understanding
- Design guidance

Focus on using established and traditional terms (some were defined in 1970s...)

Target audience

- System Architects - system design guidance
- Guiding purchasing of ECU systems... requirements on the hardware and software stacks
- HW / SoC designers (for the needed HW features)
- IP-vendors (ARM, Synopsys, Imagination Technologies, Cadence)

EARLIER/OTHER outline proposal

and some other details ...

Not agreed to be in scope?

1. Introduction

Intro: Need for multi-OS design. Possibilities given by modern hardware, hypervisors and other techniques

2. Isolation and Partitioning

Partitioning Technologies

1. Full-scale Hypervisors
2. MCU Hypervisors
3. Dedicated Core partitioning
4. Operating System process partitioning / containers
5. *Peripheral partitioning (or is this in later HW chapter)*

4. Designing for Functional Safety

Describe specific features (hardware and software) supporting Functional Safety Goals

Design process

5. Designing for Cyber-security

Describe specific features (hardware and software) supporting Functional Safety Goals

TrustZone but also others

Design process

6. Hardware sharing

Hardware Device sharing - is the main purpose of Hypervisors.
However device sharing can be set up with other means?
Yes, examples:

- h/w display layers
- GPU virtualisation with OS ID support

Specific Hardware Support

- GPU virtualization features
- GPU OS isolation
    - Separate input ports
    - GPU scheduler ensuring pipeline for each OS is serviced
    - Memory separation
    - Drawing isolation
        - Dedicated functional safety drawing paths, e.g. separate 2D rendering path for cluster telltales
- Other HW features
    - Bus Master / IP / Memory access isolation
        - Independent security and safety groups control access of IP on bus and memory protection
    - Multi-level security isolation outside of common IP such as TrustZone
        - e.g. Implementation in real time R7 CPU
    - Lifecycle Management
        - e.g. control of security at different stages of its life ★ which chapter does this fit in?
- *--> Some architectures can guarantee QoS on internal interconnects and caches, etc. (what else?) controlled by VM configuration or h/w controls.*

7. Selecting a Design

- Describe typical constraints and input that drives design decisions.
- Walk through the main characteristics of each method, and any other consequences of choosing one method or other.
- Guide the reader through the selection process, (comparing constraints to consequences of each choice)
  (Look at the "How to choose graphics sharing technology" presentation for inspiration)

8. Future outlook (?)

--> build something out of the "wish list"?

To be sorted

System-level quality of service. What is this and where to fit in?