RENESAS

R-CAR 3D GPU VIRTUALIZATION

AUTOMOTIVE SOLUTION BUSINESS UNIT
AUTOMOTIVE DIGITAL PRODUCTS MARKETING DIVISION

30-NOV-2020

BIG IDEAS
FOR EVERY SPACE

RENESAS

Introduction

- Renesas started to provide ***3D GPU hardware virtualization*** for R-Car Gen3
  - Prototyping was already done for R-Car Gen2 and evaluated with customers & partners

- The Renesas solution is focusing on the requirements of the automotive industry
  - Renesas is a leading supplier for instrument cluster & IVI systems

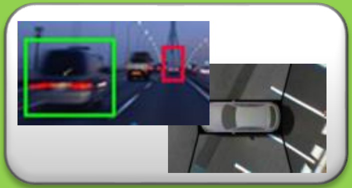
- Renesas is pleased to give an overview of the R-CarH3 3D GPU hardware virtualization within this GENIVI forum

Example application use case scenario

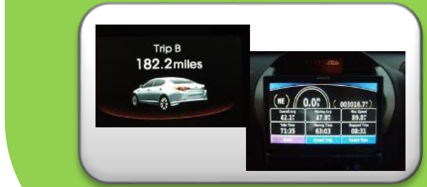
RTOS

Cluster

Camera Overlay



Trip & HMI

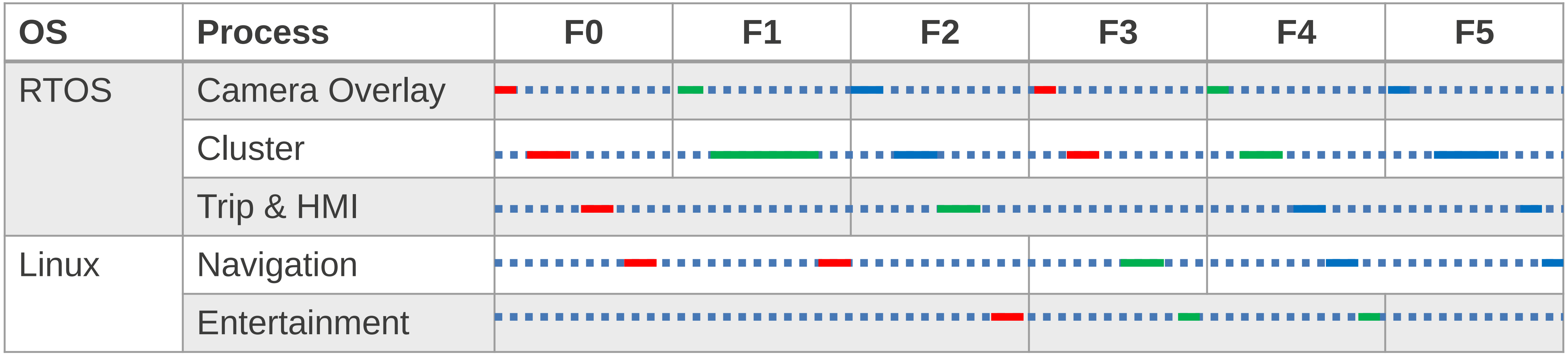


Complex OS (e.g. Linux)

Navigation

Entertainment

Simplified desired GPU usage



OS	Process	F0	F1	F2	F3	F4	F5
RTOS	Camera Overlay	🟥	🟩	🟦	🟥	🟩	🟦
	Cluster	🟥	🟩	🟦	🟥	🟩	🟦
	Trip & HMI	🟥		🟩		🟦 🟦	
Linux	Navigation	🟥 🟥			🟩	🟦 🟦	
	Entertainment	🟥			🟩 🟩		

Application constraints

- ***Camera Overlay*** framerate defined by VIN – must not miss a frame
- ***Cluster*** must achieve 60Hz (16ms) – must not miss a frame
- ***Trip & HMI*** shall reach 30Hz – should not miss a frame
- ***Navigation*** shall reach 30Hz – tolerant for missed frames
- ***Entertainment*** shall reach 30Hz – tolerant for missed frames

Fx : Display output frame

🟥 : GPU drawing (app_frame % 3)

🟩 : GPU drawing (app_frame % 3) +1

🟦 : GPU drawing (app_frame % 3) +2

Derived requirements

- Time scheduling shall work between processes (e.g. IMG context priority)
- Time scheduling shall work between OS
- It shall be possible to ensure GPU bandwidth at least for an OS (better for a process)
- It shall be possible to define & supervise "*deadlines*" for applications
- An application shall be able to decide what happens in case "*deadlines*" are exceeded; e.g. ignore, restart application, stop other processes

- Worst case context switch time needs to be "*defined"* and must not exceed 3ms (should be in 1ms range)
- Resources of each OS must be protected from access of other OS; desirable also for processes

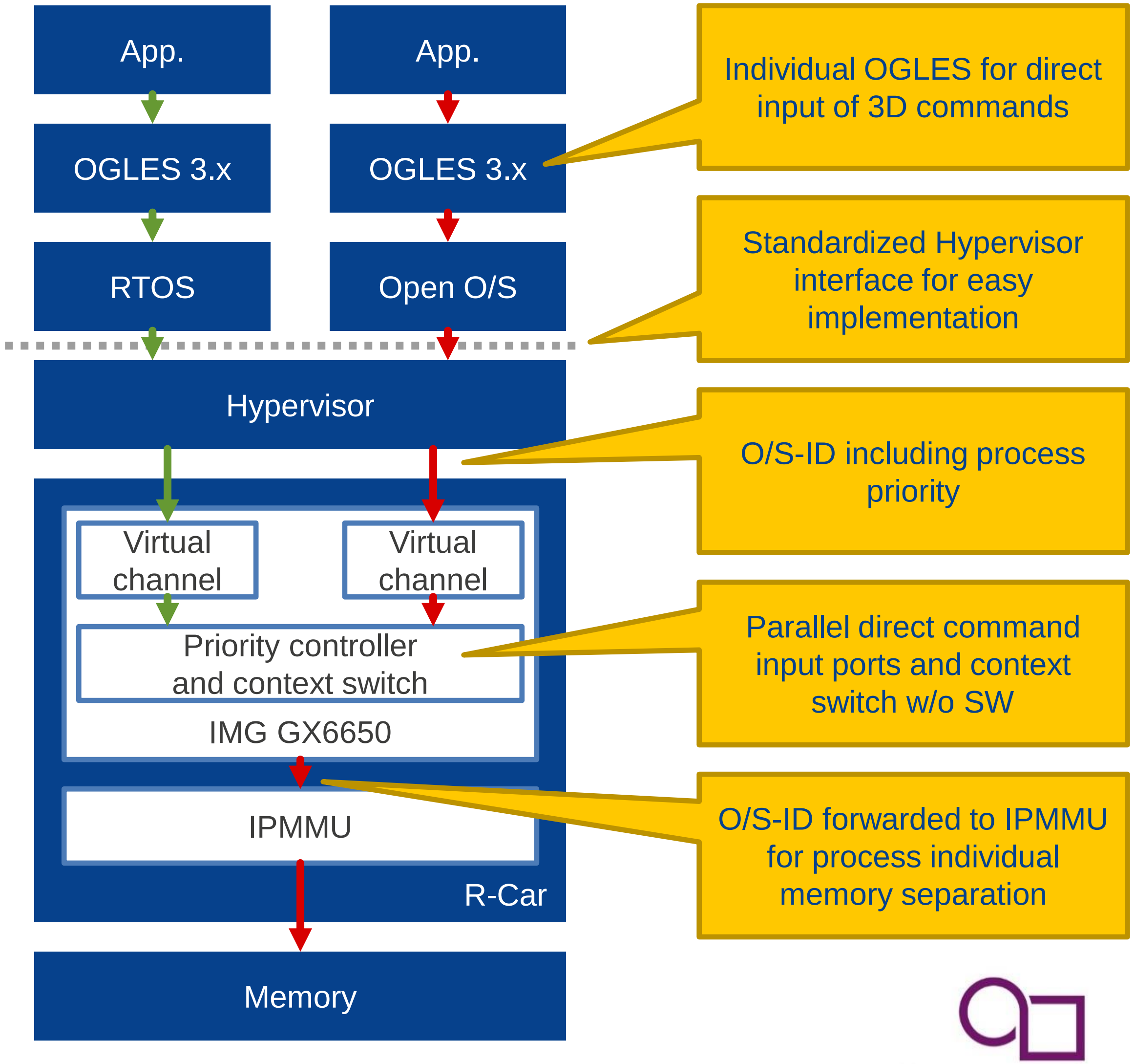
R-Car 3D GPU virtualization

3D GPU includes an internal **h/w control logic**

- To distribute the tasks to the different shader clusters
- To manage the 3D GPU graphics resources like memory

This control logic generates and maintains **several drawing contexts in parallel**

- Fine preemption and fine-grain task switching based on hypervisor priorities
- Direct link to IPMMU



Individual OGLES for direct input of 3D commands

Standardized Hypervisor interface for easy implementation

O/S-ID including process priority

Parallel direct command input ports and context switch w/o SW

O/S-ID forwarded to IPMMU for process individual memory separation

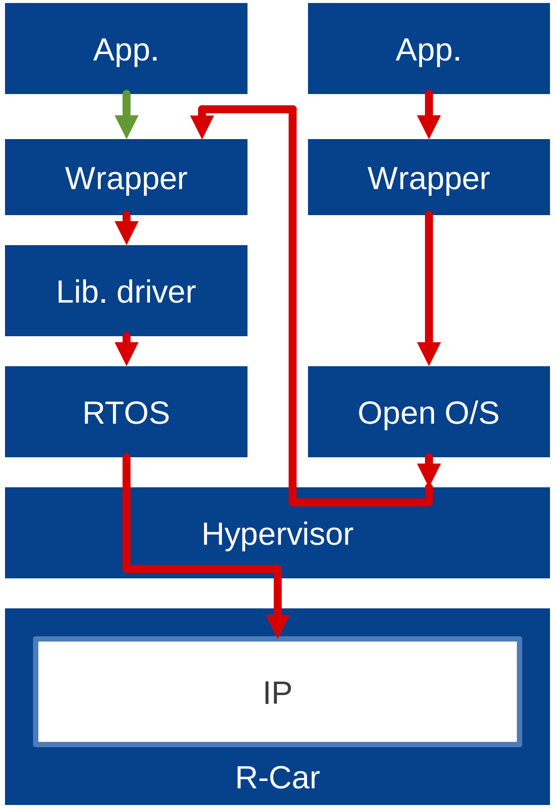
3D GPU virtualization SW and HW (R-CarH3)

SW:

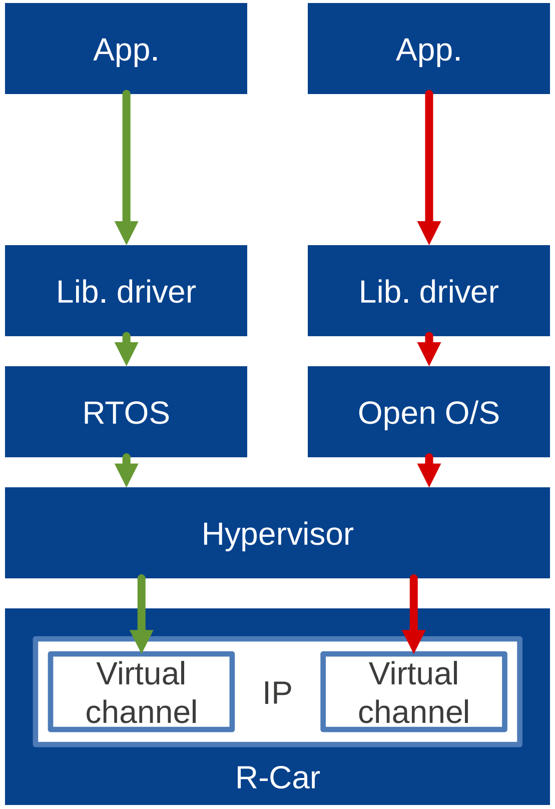
- No OSID information from GPU to IPMMU
  - You could consider to provide this via the bridge
- Guest driver forwarding calls via bridge
  - You could place the bridge at different levels, e.g. let parts of the "real" driver run in any guest, to make it efficient

HW (R-CarH3) :

- GPU provides OSID to tell the system on behalf of which OS it is working
- IPMMU is connected to GPU to enable OSID based protection
- Guest OS can run almost the standard native driver and no forwarding is done; each guest has even own kick register for accessing GPU



SW



R-Car Gen3

Resource separation by R-Car IPMMU

Virtual memory addresses are a fundamental concept for sandboxes and to ensure freedom from interference

- Private address space for every virtualized application
- Access right control for every application

However usually such virtual memory concept is limited to the CPU only

R-Car extends the virtual memory concept to all hardware IP's in the system

This **unique concept** allows applications to use even **hardware accelerators w/o corrupting the memory** resources of other applications

All R-Car include an **IPMMU** extending the ARM CPU MMU to the whole system

- Can be used to translate any memory request by a h/w IP from the process specific virtual address into a physical address
- Access rights of the applications can be applied also to the used h/w IP's

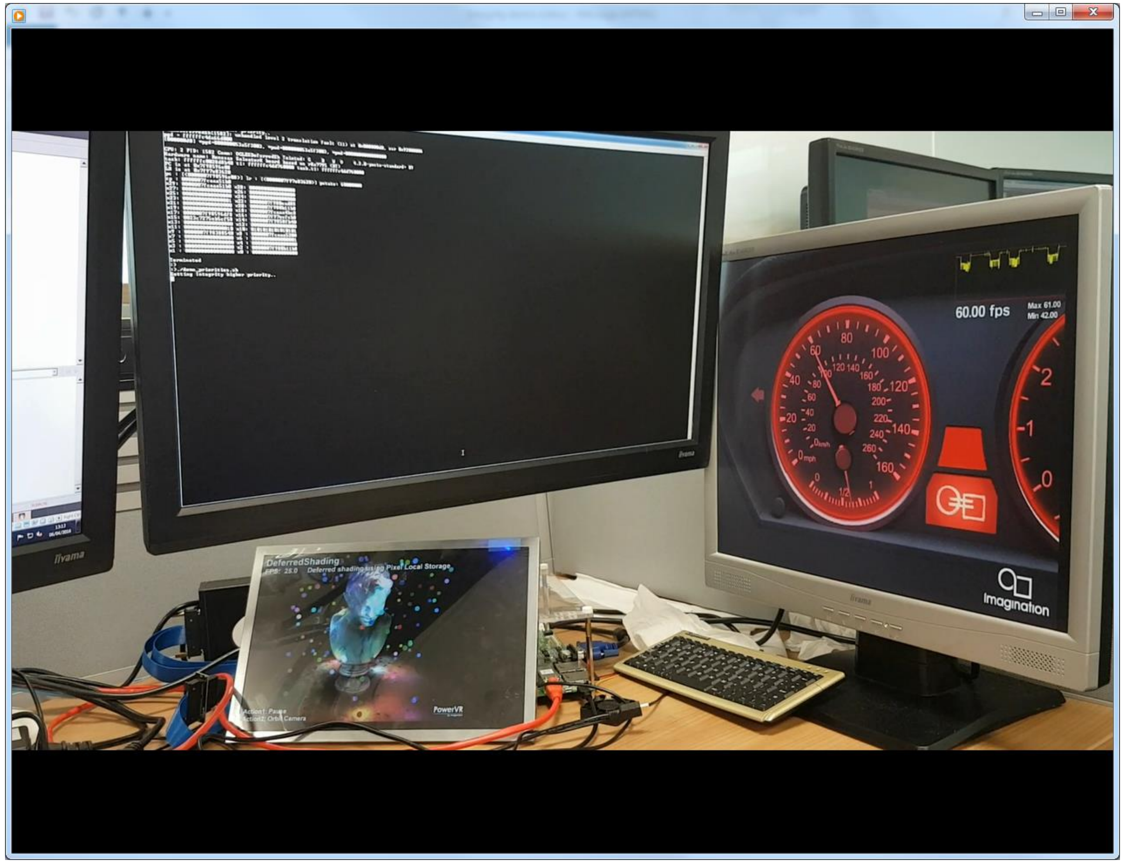
Key advantages

- Physical own GPU access channel for each OS
  - Each OS has its own almost “normal” driver
  - In case of any problem under one OS, only the driver for that OS needs to restart, no influence on the other OS driver
  - No “API proxy” approach required
- Optional OS isolation feature to guarantee bandwidth for the isolated OS and to ensure faulty applications have no influence on the isolated OS
- Scheduling is managed by the GPU FW itself, not by the CPU driver
- IPMMU and OSID awareness of GPU ensure, that the GPU can only access memory of the OS on which’s behalf it is working
- Small scheduling granularity of usually less than 1ms to ensure fast switch to higher priority drawing tasks
- Deadline scheduling for each OS
  - Deal with bad workload & blocking applications
  - No OS can block the GPU

Checking the HW virtualization

- Hypervisor: GHS
- ***Cluster*** running on INTEGRITY
- ***Deferred shading*** running on Linux
- Priority switch between INTEGRITY & Linux
- When INTEGRITY priority is higher, 60fps for the cluster is ensured
- In RR scenario the cluster drops to ~45fps

→ **Gen3 GPU virtualization works as expected**



Information & demos

POWERVR VIRTUALISATION

a critical feature for automotive GPUs

May 2019

- IMG white-paper: https://www.imgtec.com/whitepapers/gpu-virtualization/

Robustness

Thanks to our hardware virtualization, if one application crashes for any reason, the other, contained in its own isolated domain, remains unaffected.

- IMG demo video: https://www.youtube.com/watch?v=qliQGm4B3C8&list=ULa1S21DitU34&index=65

- ADAC demo: https://www.renesas.com/cn/en/solutions/event/devcon2017/sp-15.html

BIG IDEAS FOR EVERY SPACE

www.renesas.com

THANK YOU!