ExynosAuto V9 Hypervisor Solution

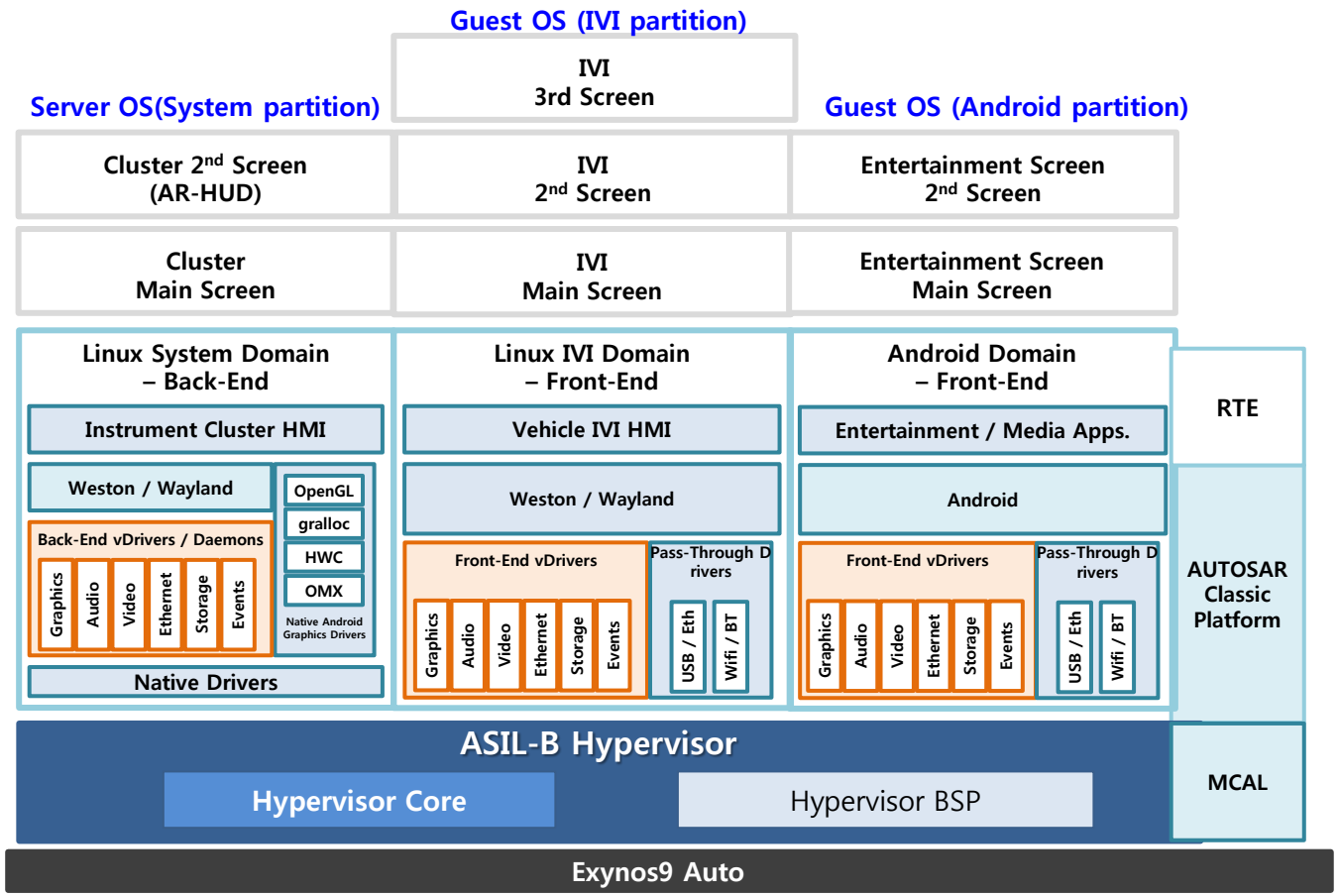
2018.07.10

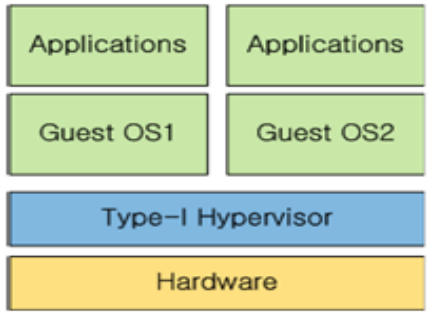
SW R&D Center

Virtualization on ExynosAuto9 for HCP3

- Type-I "bare-metal" hypervisor (Redbend hypervisor)
- Para-Virtualization(PV) and Pass-Through(PT)/HPT drivers
- CPU and memory pinning supported

HPT : H/W assisted PT
RTE : Runtime Environment

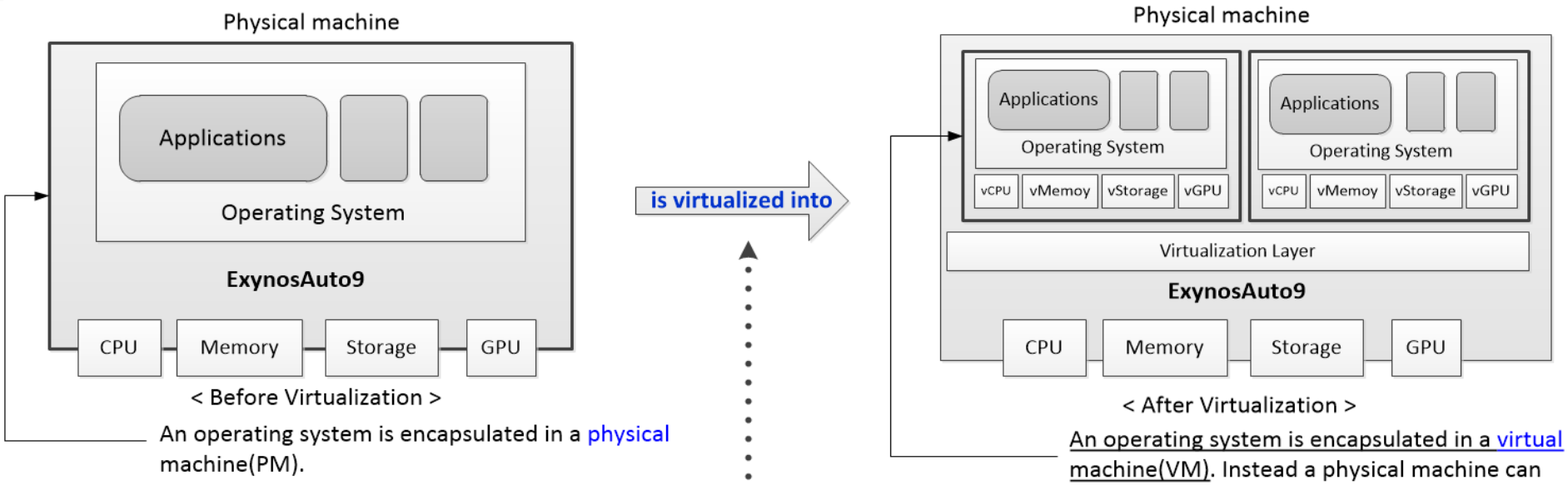




Type-1(baremetal) Hypervisor

Xen	Redbend
Dom0	Server OS (I/O Domain)
DomU	Guest OS
Dom0S (System partition)	Server OS (System partition)
DomI (IVI partition)	IVI Guest OS (IVI partition)
DomA (Android partition)	Android Guest OS (Android partition)

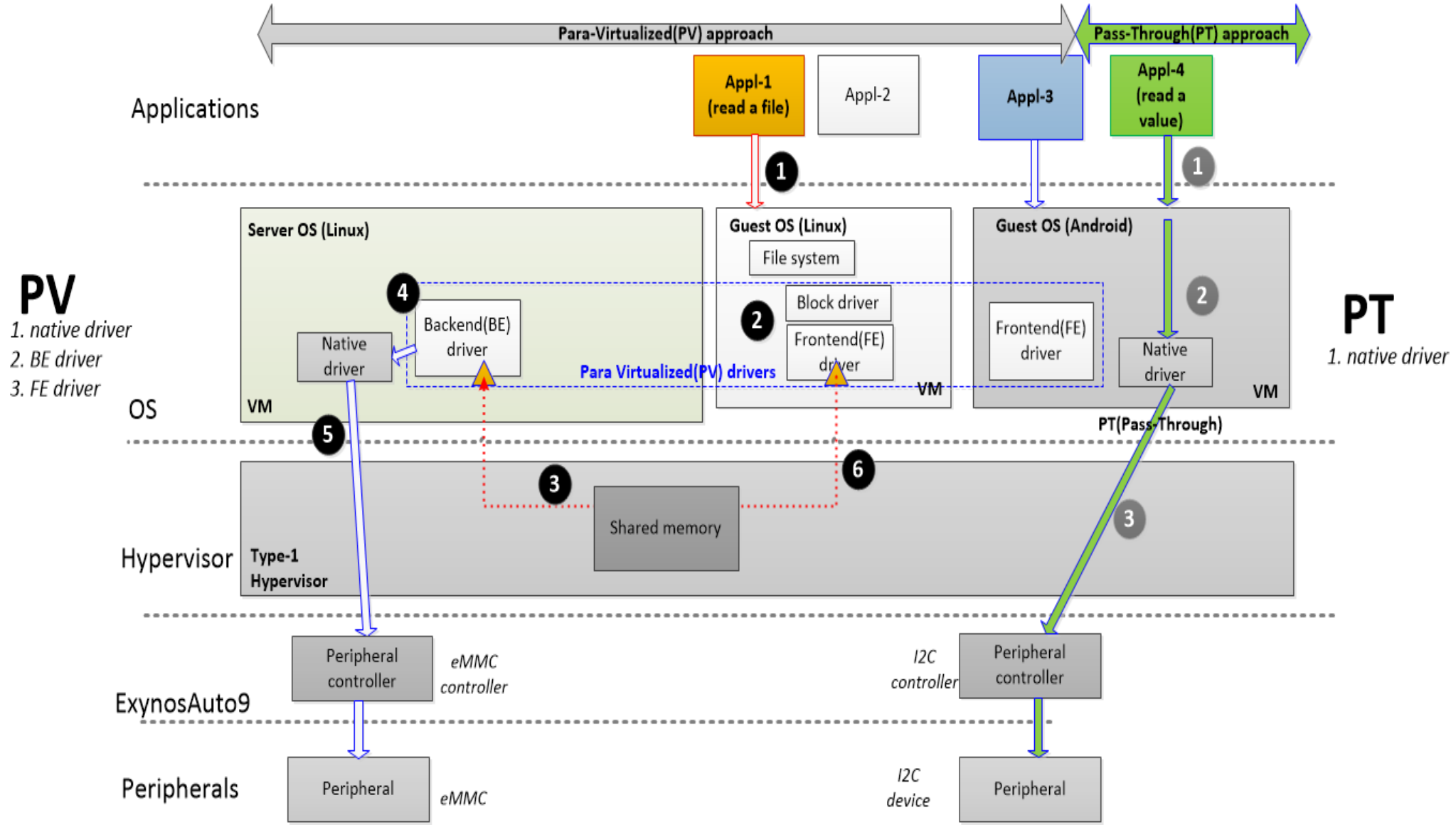
Virtualization on ExynosAuto9



Category	*H/W IP*	*Descriptions*		
Virtualization Support of ARM cores	*ARM CA-76(Enyo) cores*	*Hypervisor Mode (Exception Level2, EL2)*	*By HyperVisor Call(HVC)*	CPU virtualization
	MMU	*2-Stage Page Translation*	*VA <-> IPA <->PA translation*	Memory virtualization
	Interrupt	*Virtual interrupt Injection*	*ARM GIC400*	
Para-Virtualization (PV)	*Shared Peripherals*	*Split Device Driver Model across Guest OSes*	*FE(Front End) drivers in DomU VM <-> BE(BackEnd) drivers in Server OS*	IO virtualization
Device Passthrough (PT)	*Dedicated Peripherals (PT)*	*DomU can access directly to SFR of ExynosAuto9.*	*Some devices are dedicated to a certain VM. (GPU)*	
	***IOV Peripherals(HPT)*		*Some H/W logic for IOV is embedded into the peripheral to make more than two VMs to access the same peripheral at the same time. (e.g UFS/PCIe SR-IOV).*	

** MMU: Memory Management Unit* ** GIC : Generic Interrupt Controller* ** VA : Virtual Address* ** IPA : Intermediate Physical Address*
** GPU: Graphic Processing Unit* ** VM : Virtual Machine* ** PA : Physical Address* ** HPT : H/W assisted PT*
** SFR : Special Function Register.* **IOV : IO Virtualized*

How PV & PT drivers work



PT and PV drivers From a Guest OS point of view

	PV driver	PT driver		
		Non-DMAable	DMAable	
			w/o H/W support for virtualization	w/ H/W support for virtualization
	Frontend driver	PT driver	PT drivers	HPT drivers
e.g.	Network frontend driver	I2C PT driver	USB device PT driver	*UFS-IOV VF driver **PCIe-IOV VF driver
How physical memory access is controled	By backend driver of Server OS	By Hypervisor using the 2nd-stage address translation of ARM MMU	By Hypervisor using the 2-stage memory protection unit(S2MPU)	

* : RX/TM DMA is done by a DMA engine of UFS Host Controller.

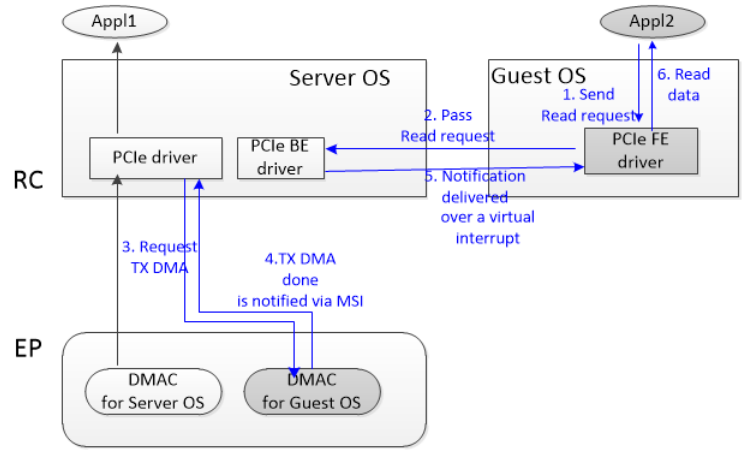
** : RX/TM DMA is done by a DMA engine of a PCIe EP(End Point).

HW-assisted Pass Through Driver (HPT)

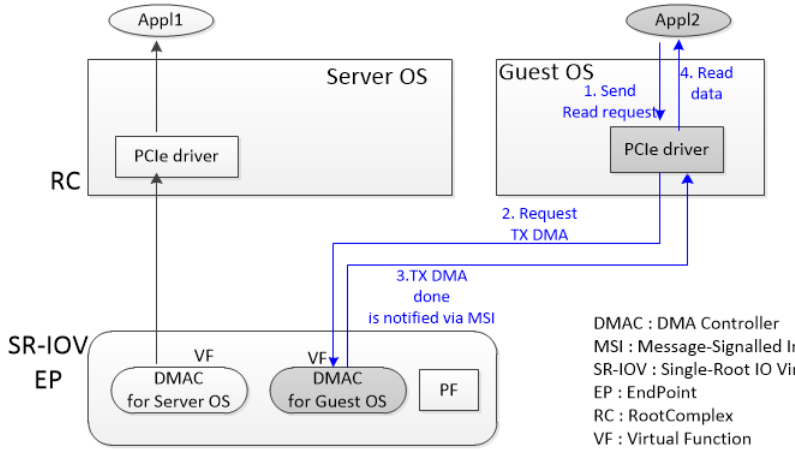
PCIe (SR-IOV)

(e.g) Appl2 in Guest OS reads from a device.

Without SR-IOV



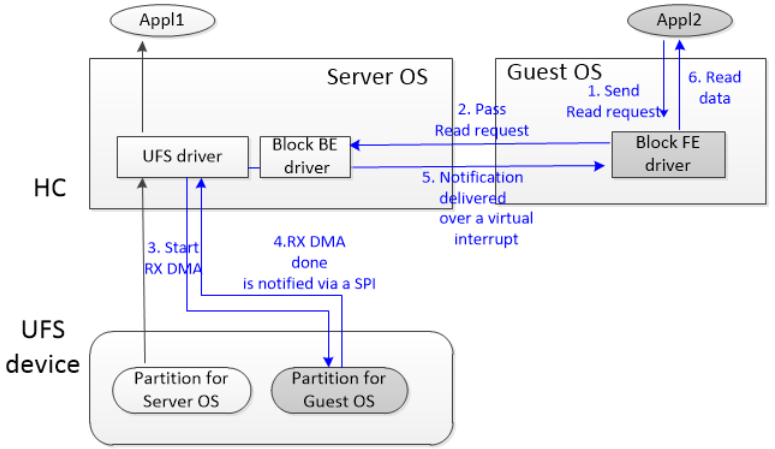
With SR-IOV



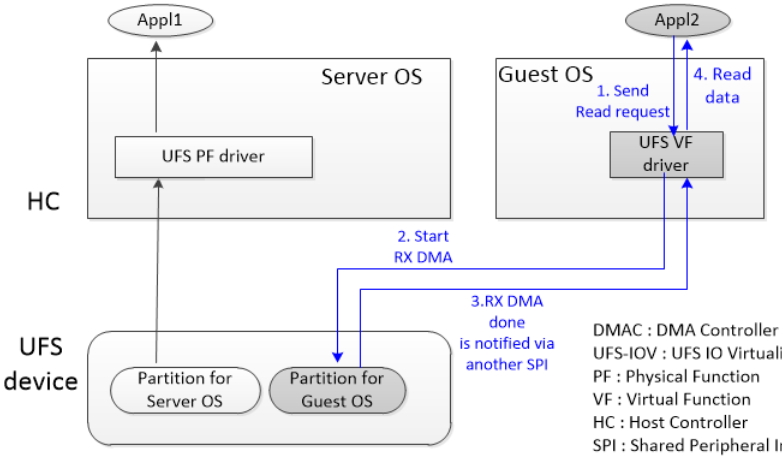
DMAC : DMA Controller
MSI : Message-Signalled Interrupt
SR-IOV : Single-Root IO Virtualization
EP : EndPoint
RC : RootComplex
VF : Virtual Function
PF : Physical Function

UFS (UFS-IOV)

Without UFS-IOV



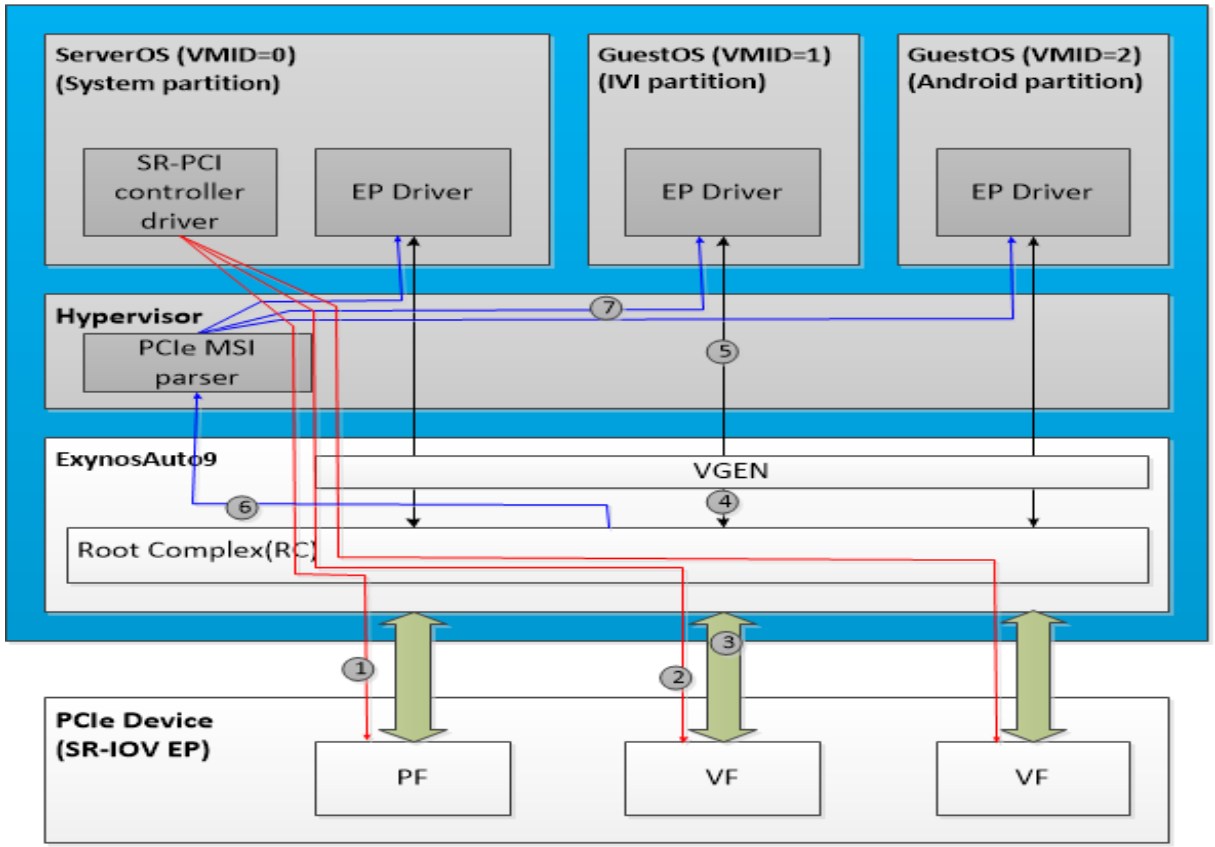
With UFS-IOV



DMAC : DMA Controller
UFS-IOV : UFS IO Virtualization
PF : Physical Function
VF : Virtual Function
HC : Host Controller
SPI : Shared Peripheral Interrupt

[HPT in detail] S/W Block Diagram for PCIe SR-IOV

(e.g.) IVI partition reads from one of VFs of the SR-IOVable EP.



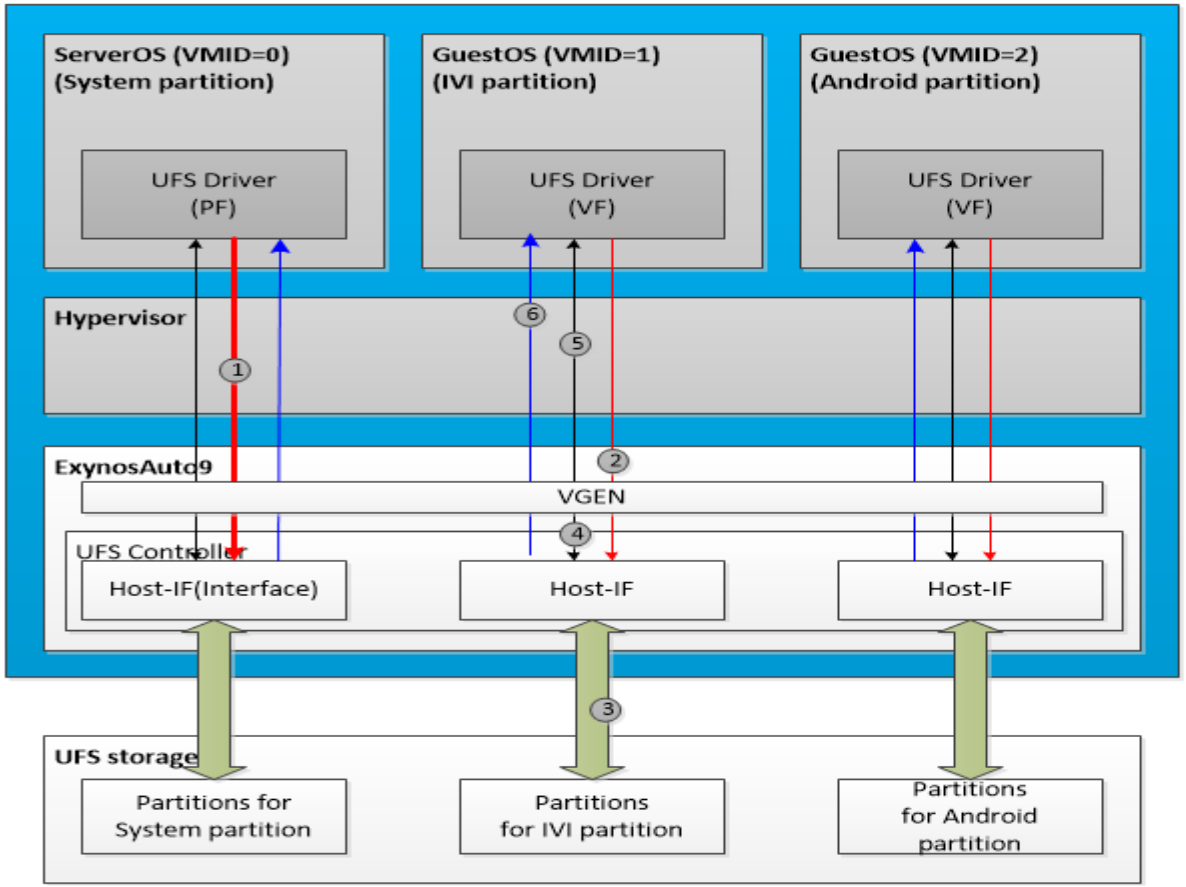
S2MPU: Stage-2 Memory Protection Unit
GIC: Generic Interrupt Controller
SR: Single Root
MSI : Message Signalled Interrupt
PF: Physical Function (for serve OS)
VF: Virtual Function (for guest OS)
VGEN : VID Generator/Detector
VID : ID of a virtual machine who is using the device

Data
Interrupt
Configuration
Data/Interrupt

① A SR-PCI controller driver sets common configurations for 3 Functions – PF and 2 VFs.
② The SR-PCI controller driver configures a VF assigned to the IVI partition.
③ A EP driver of IVI partition requests Read DMA from the VF, and the VF starts TX DMA which carrys the VMID(2) information of IVI partition.
④ A PCIe RC of ExynosAuto9 identifies the VMID from the DMAed data through VGEN.
⑤ It is checked by S2MPU whether the DRAM area for TX DMA is valid for IVI partition by walking a page table which the hypervisor has made up for the relevant VF.
⑥ When the DMA is done, the VF sends a MSI to RC. The MSI is routed by GIC into the hypervisor.
⑦ The hypervisor parses the MSI to find out the EP relevant driver to the interrupt, then inject a Virtual interrupt to the EP driver of the Guest OS(IVI partition).

[HPT in detail] S/W Block Diagram for UFS-IOV

(e.g.) IVI partition reads from a UFS storage.



S2MPU: Stage-2 Memory Protection Unit
PF: Physical Function (for server OS)
VF: Virtual Function (for guest OS)
VGEN : VID Generator/Detector
VID : ID of a virtual machine
who is using the device

Data
Interrupt
Configuration (Full)
Configuration (Partial)
Data

1. A PF driver sets common configurations for 3 Host-Ifs of a UFS controller.
2. The UFS VF driver of IVI partition configure its own Host-IF.
3. The UFS VF driver issues a command to Host-IF to read data from a IVI storage partition.
4. The UFS VF driver starts Read DMA tagged with a VMID(1) from a FIFO to a part of DRAM.
5. It is checked by S2MPU whether the DRAM area is eligible for Read DMA by walking a page table which the hypervisor has made up for the relevant Host-IF of the UFS controller.
6. If the DMA request is valid, the DMA starts up and ends up triggering a SPI(Shared Peripheral interrupt) dedicated to the Host-IF for the IVI partition to notify the DMA is done to the UFS VF driver.

To-be-confirmed points

- Please fill in driver distribution table in the next page.
- SR-IOVable EP will be used in HCP3 project ? If not, don't we need to develop PCIe RC driver supporting SR-IOV ?
- NPU is not under control of hypervisor ?

Device Driver Distribution Across Domains

**Native and PVB drivers exist in the same domain.
(E) : Driver needed to support early functions

Device		Cluster partition	IVI partition	Android partition	Remarks
OS		Linux	Linux	Android	
Display	MIPI-DSI/DP				
PCIe	PCIe #0 (4 Lane)				Inter-SoC Communication *In case of SR-IOV EP, HPT is available.
	PCIe #1 (2 Lane)				
	PCIe #2 (2 Lane)				
Ethernet	Eth #0				
	Eth #1				
I2C	I2C #0 to #4	PT	PT	PT	
SPI	SPI #0 to #1	PT	PT		
UART	UART #1 to #5	PT(E)	PT	PT	
UART_console	UART #0	PVF	PVF	PVF	Native and PVB driver exist in HV
I2S	I2S				
ADC					
GPIO	GPIO				
USB	USB 2.0/3.1				
UFS	UFS HS-G3	HPT	HPT	HPT	H/W assisted virtual function for DomU
Clock	Clock control	PVB	PVF	PVF	
Power	Power domain	PVB	PVF	PVF	
Regulator	Buck control	PVB	PVF	PVF	
Audio	Audio decoder				
Video Codec	Multi-Format Codec				
3D Graphics	GPU0 #0 (MP3)	PT			Native Linux/Android support are required.
	GPU0 #1 (MP3)			PT	
	GPU0 #2 (MP12)		PT		
Thermal	TMU	PT			