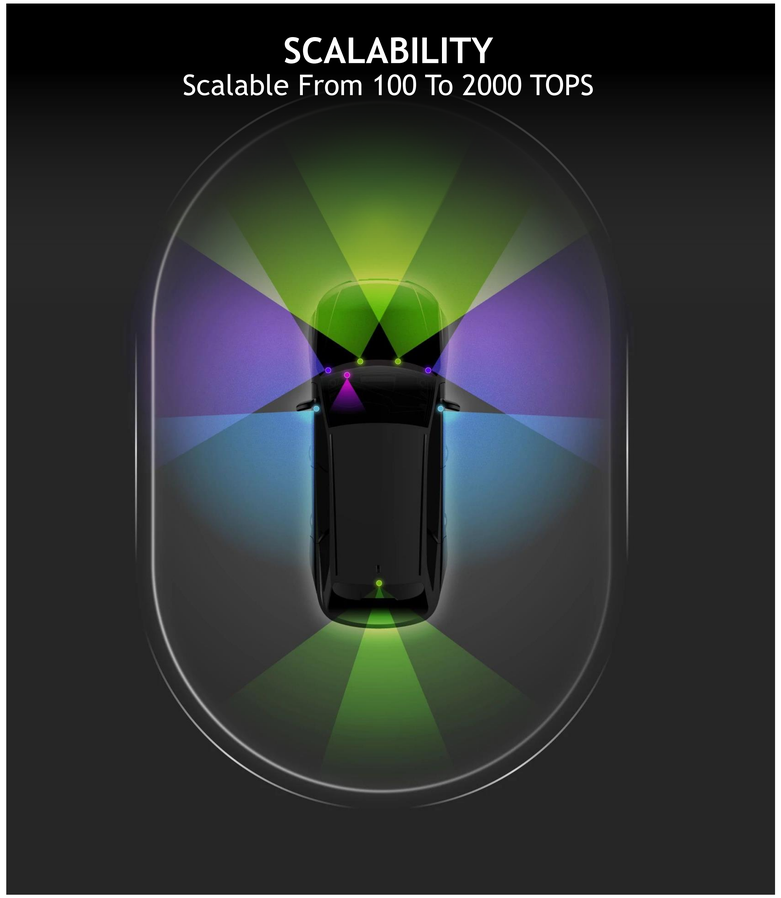
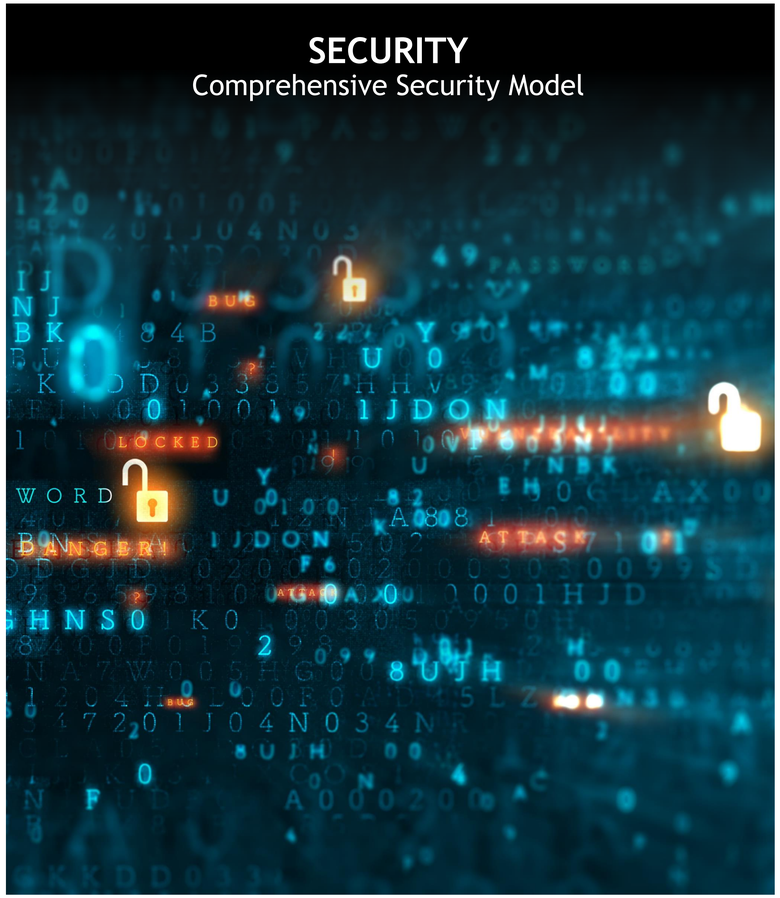
SAFE AND SECURE AUTONOMOUS VEHICLE DEVELOPMENT WITH NVIDIA DRIVE OS

GTC MARCH 2022

DRIVE OS

For Automated Driving Systems & Cockpit AI





NVIDIA

NVIDIA AUTOMOTIVE SAFETY-CRITICAL SYSTEMS

Key Pillars of a Safety-Critical System

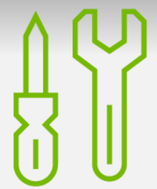
AVAILABILITY
Quality-of-Service

RELIABILITY
Correctness-of-Service

RESILIENCE
Error Tolerance, Detection & Correction

SECURITY
Comprehensive Security Model

SAFETY EXTENSIONS
Error Correction & Diagnostics



SAFETY CERTIFICATION
Automotive Industry Safety Standards

NVIDIA AUTOMOTIVE SAFETY-CRITICAL SYSTEMS

For Autonomous Vehicles & Cockpit AI

DRIVE OS QNX FOR SAFETY

Safety Certifiable System*

- Quality-of-Service, Real-Time Operating System
- Isolation, Freedom from Interference
- Error Tolerance, Detection & Correction
- Comprehensive Security Model
- ISO 26262 Safety Certifiable**

DRIVE OS LINUX WITH SAFETY EXTENSIONS

Safety-Critical System Foundation

- NVIDIA DRIVE Hypervisor
  - Quality-of-Service, Isolation, Freedom from Interference
- Error Tolerance, Detection & Correction
- Comprehensive Security Model
- Freedom and Flexibility of Open-Source Software

** DRIVE OS QNX for Safety is an ISO26262 Safety Certified SEooC*
*** Blackberry QNX OS for Safety (QOS) is ISO 26262 ASIL-D Safety Certifiable*

NVIDIA AUTOMOTIVE SAFETY-CRITICAL SYSTEMS

For Autonomous Vehicles & Cockpit AI

	DRIVE OS QNX for Safety Safety Certifiable System	DRIVE OS Linux with Safety Extensions Safety-Critical System Foundation
AVAILABILITY		
Guaranteed Quality-of-Service (NVIDIA Hypervisor)	✓	✓
Real-time Operating System (RTOS)	✓	✗ **
RELIABILITY & RESILIANCE		
Strict Resource Isolation (NVIDIA Hypervisor)	✓	✓
Freedom from Interference (NVIDIA Hypervisor)	✓	✓
Redundancy (via Hardware Configuration)	✓	✓
SECURITY		
Secure Boot	✓	✓
Security Services	✓	✓
SAFETY EXTENSIONS		
Safety Services	✓	✓
DRAM & GPU ECC (Error-Correcting Code)	✓	✓
Hardware Diagnostic Services (Runtime IST)	✓	✓
Functional Safety Island (FSI) (ASIL D capable)	✓	✓
Classic AUTOSAR (for FSI)	✓	✓
Error Propagation & Handling	✓	✓
Microcontroller (MCU) (if included in production Hardware)	✓	✓
Classic AUTOSAR (for MCU)	✓	✓
Power Management	✓	✓
Temperature & Voltage Monitoring	✓	✓
In-System-Test (IST)	✓	✓
Fail Over Handler	✓	✓
NVIDIA Orin Safety Manual(s)	✓	✓
DRIVE OS Linux with Safety Extensions Manual	✗	✓
SAFETY CERTIFICATION		
DRIVE OS QNX Safety Manual	✓	✗
ISO 26262 Safety Certifiable	✓	✗

*** Real-time Ubuntu kernel patch available*

NVIDIA

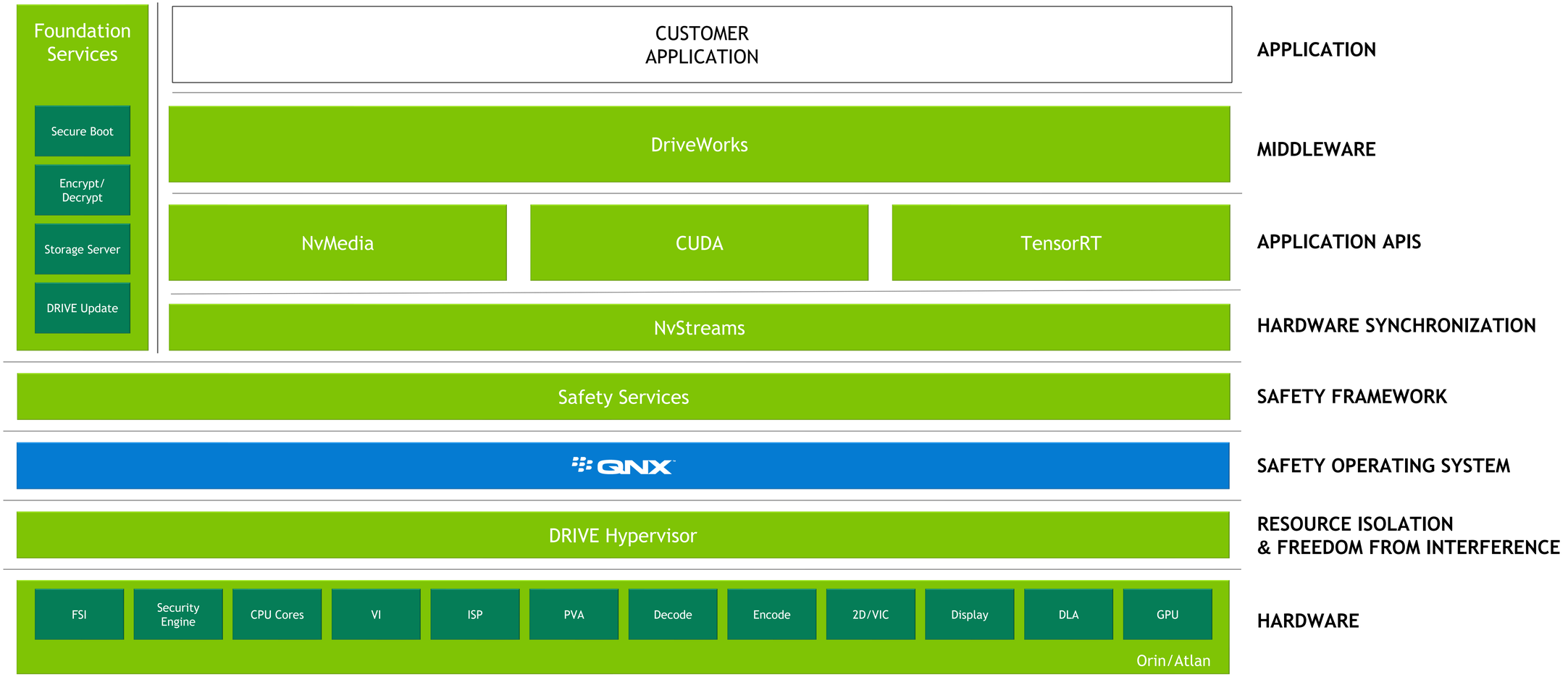
WHY QNX FOR SAFETY?

Safety OS Key Selection Criteria

- ISO 26262
  - ASIL D certified RTOS
  - TCL3 qualified toolchain
- POSIX PSE52 standards certification
  - Requirement for CUDA support
- Common Unix heritage with Linux
  - Rich dependent library support

DRIVE OS QNX FOR SAFETY

Architecture Overview



IS LINUX SAFE?

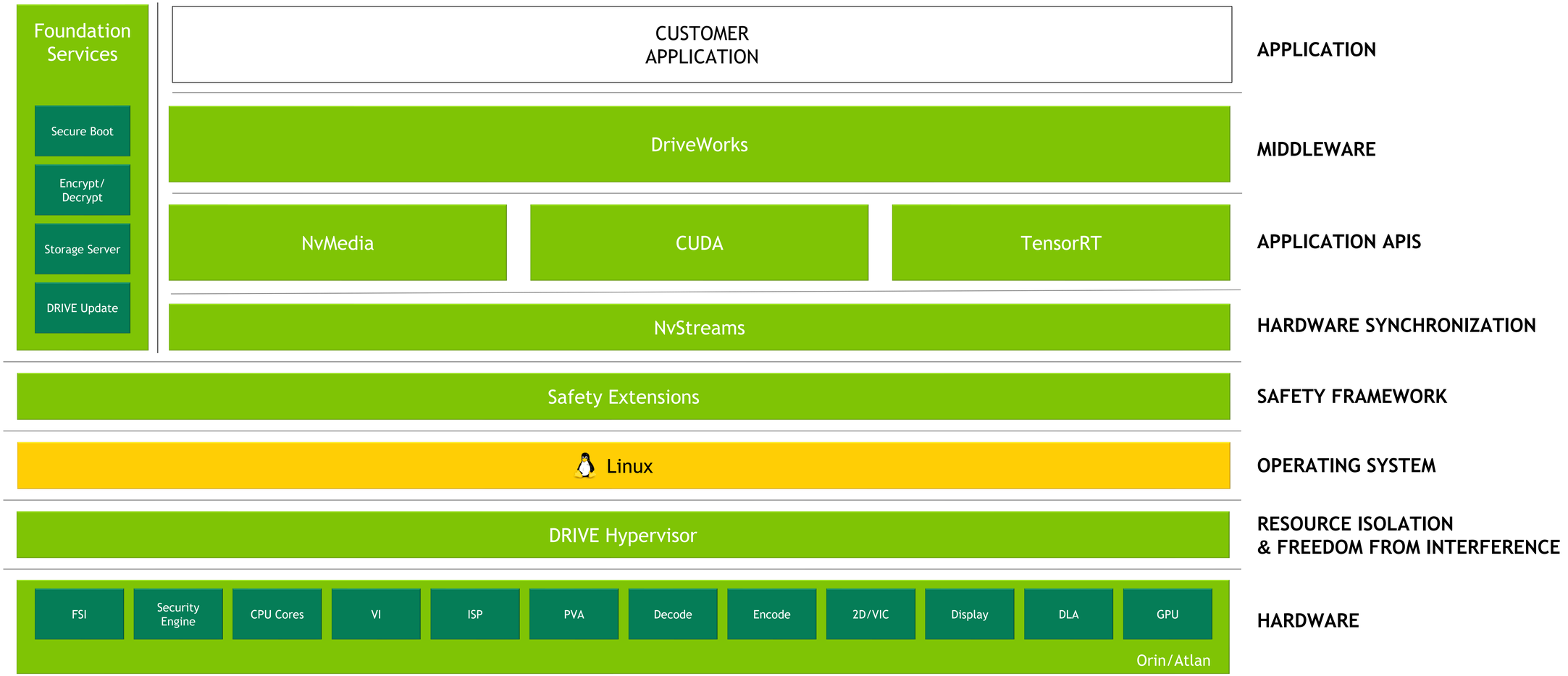
It Depends on Safety Decomposition and Criteria

- DRIVE OS QNX for Safety follows the ISO 26262 international standard for functional safety up to ASIL D
- DRIVE OS Linux by nature utilizes FOSS component that are not developed to this standard
- The Linux Foundation's ELISA project focuses on Linux in Safety-Critical Systems with one of the goals being to work with certification authorities and standardization bodies to establish how Linux can be used as a component in safety-critical systems
- In September of 2021, ISO/NP PAS 8926 – "Road vehicles – Functional safety – Qualification of pre-existing software" project was approved, and is targeting publication in Q3 2023
- This will then generate changes to the 3rd edition of ISO 26262-6 and ISO 26262-8 anticipated to appear around 2025
- General industry consensus is that ASIL B will likely be the highest target for Linux

- DRIVE OS Linux with Safety Extensions provides many of the same safety features that are provided by DRIVE OS QNX Safety Services

DRIVE OS LINUX WITH SAFETY EXTENSION

Architecture Overview



WHAT'S NEW?

In DRIVE OS 6

Platform APIs

- CUDA, TRT, and cuDNN support for Orin's Ampere GPU
- NvMedia support for the Orin's new Optical Flow Accelerator (OFA), DLA, and AV1 encode and decode
- NvStreams support over PCIe & Mellanox Accelerated support across inter-ECU boundaries
- Introducing Vulkan SC in 6.1

QNX Safety Services

- DRAM ECC Patrol Scrubbing
- Integration of Orin's integrated Functional Safety Island (FSI)
- New streamlined Safety Services Architecture (replacing NvGuard 3LSS)

Linux Safety Extensions

- Introduction of some of the safety features deployed on QNX

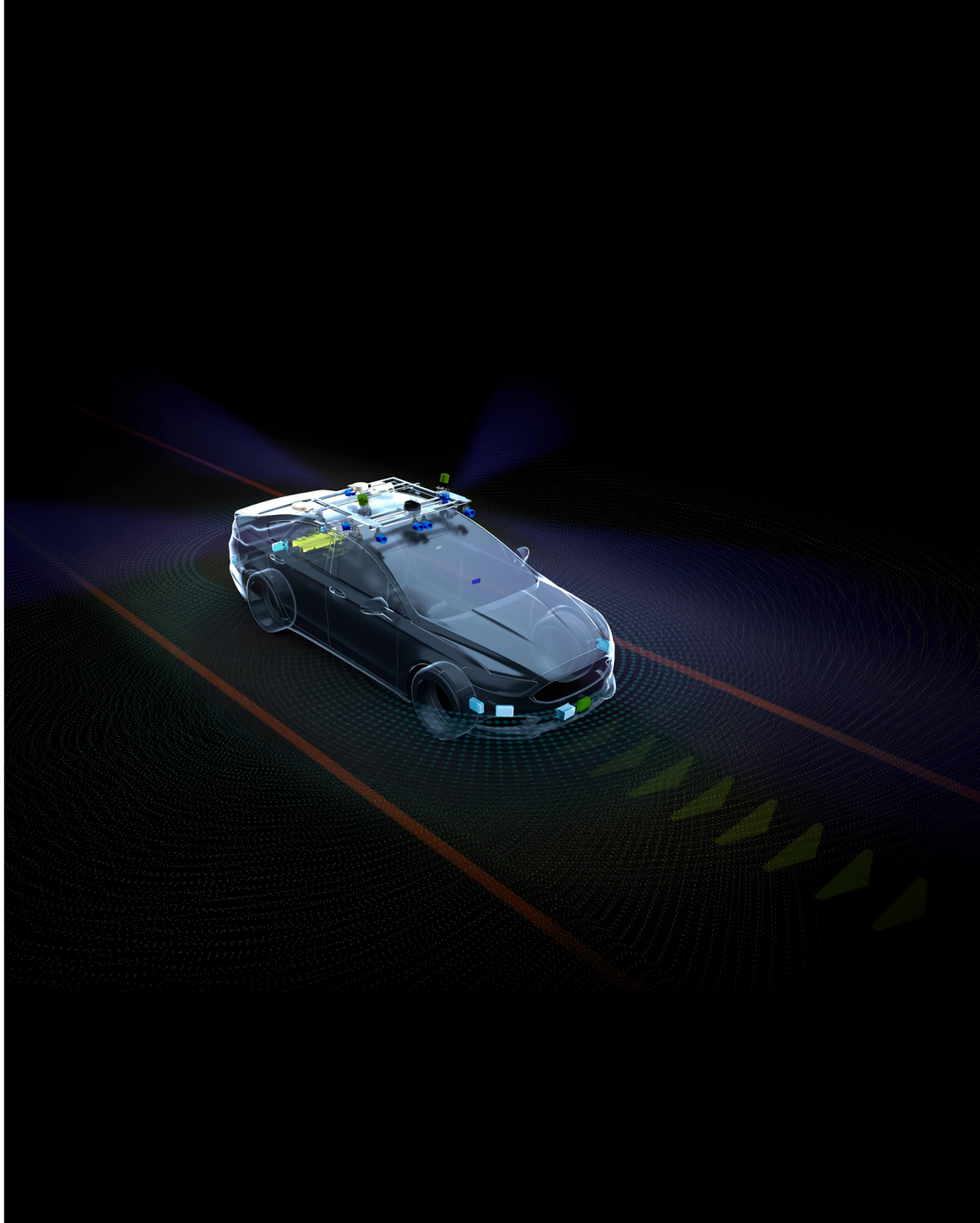
Developer Tools

- Nsight Graphics support for Vulkan & Vulkan SC
- Nsight Systems support for NvStreams, Vulkan, & Vulkan SC
- Nsight Visual Studio Code Edition

Security Services & Secure Boot

- Integration of Orin's Platform Security Controller (PSC)
- Secure boot & PKCS #11 support for Orin's additional Crypto Acceleration features

New items and key changes highlighted in blue in the following slides



WHAT'S NEW?

In DRIVE OS 6

Drive Components	DRIVE OS 5.2	DRIVE OS 6.0	DRIVE OS 6.1
Ubuntu Host Development Environment	18.04	20.04	
Ubuntu Target Root File System[1]			
Linux Kernel[1]	4.14	5.10	
Blackberry QNX SDP[2]	7.0.4	7.1.1	
Blackberry QNX QOS[2]	2.1	2.2	
QCC Toolchain	5.3	8.3	
GCC Toolchain	5.4	9.3	
C++ Feature Set	14	17	
DriveWorks	4.0	5.x	
CUDA Toolkit	10.2	11.4	
NVIDIA UDA CUDA Driver[1] (x86)	r450	r470	
TensorRT	6.4	8.x[3]	
cuDNN	7.6	8.x[3]	
Vulkan	1.2		
Wayland[1]	1.18		
Vulkan SC	n/a		1.0
QNX Screen[2]		n/a	7.1.1
PKCS#11	✓[4]	✓	

Hardware	DRIVE OS 5.2	DRIVE OS 6.0	DRIVE OS 6.1
Xavier/Xavier DevKit	✓	X	
Orin/Orin DevKit	X	✓	

Sensors			
OnSemi AR0231	✓	-	
OnSemi AR0820		✓[4]	-
Sony IMX390		-	
Sony IMX728	X	✓	
Sony IMX623			
OV OV2311			

Other		
Packaging	.run files	Debian/Docker
Distribution	NVONLINE	NGC

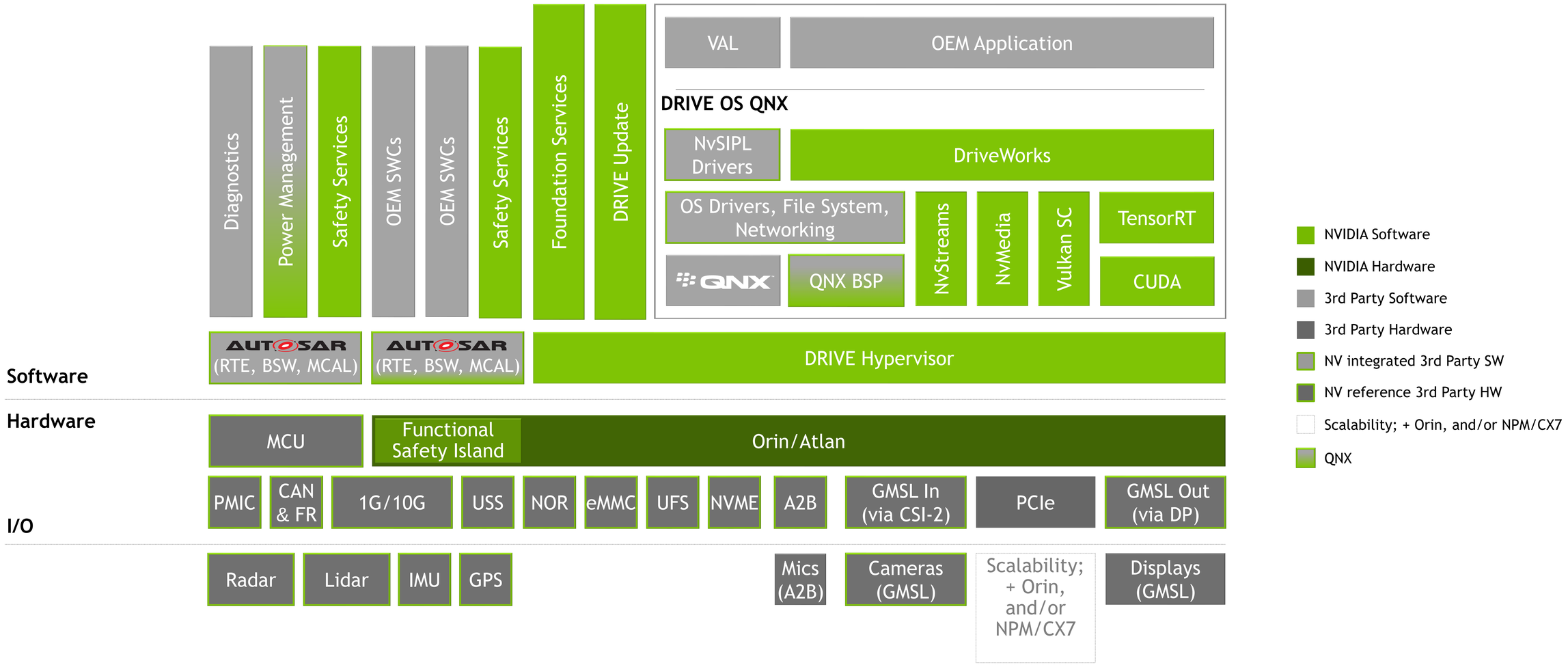
Notes:
1) Linux only, not available on QNX
2) QNX only, not available on Linux
3) Final version number TBD
4) For development only

DRIVE OS SAFETY FEATURES COMPARISON

Supported HW Metrics	Embedded Target (Tegra)	
	QNX OS for Safety (QOS)	Linux with Safety Extensions
SoC Error Detection and Reporting : Direct Reporting HSM	✓	6.0
SoC Error Detection and Reporting : Direct Reporting From SW	✓	6.0
Software Diagnostics Library and AOU	✓	6.1
DRAM ECC	✓	6.0
DRAM ECC Patrol Scrubbing	6.0	6.1
Key On/Off IST	✓	6.0
Online IST	✓	7.0
Predictive IST	-	-
Voltage and Temperature Monitoring	✓	6.0
ISO 26262 Safety Certification	✓	TBD

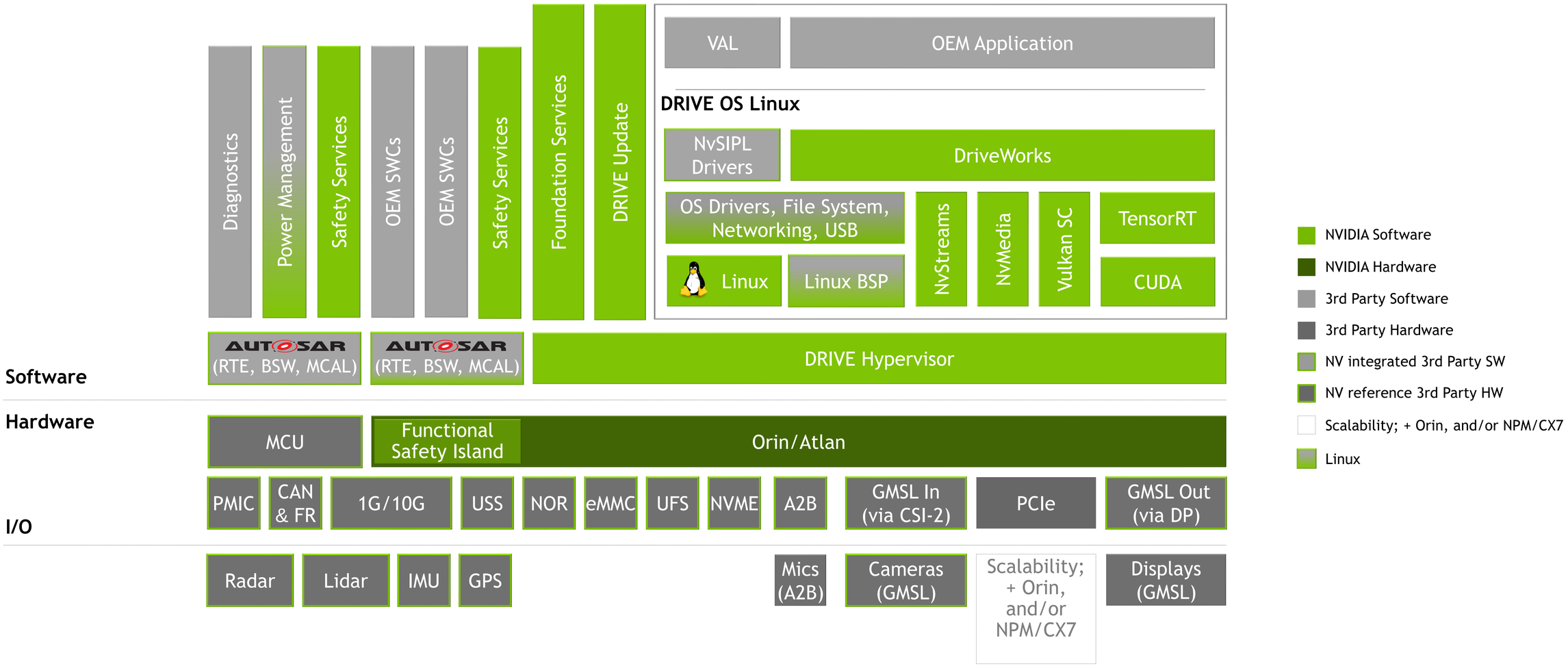
DRIVE OS QNX

AV Software Platform for Safety



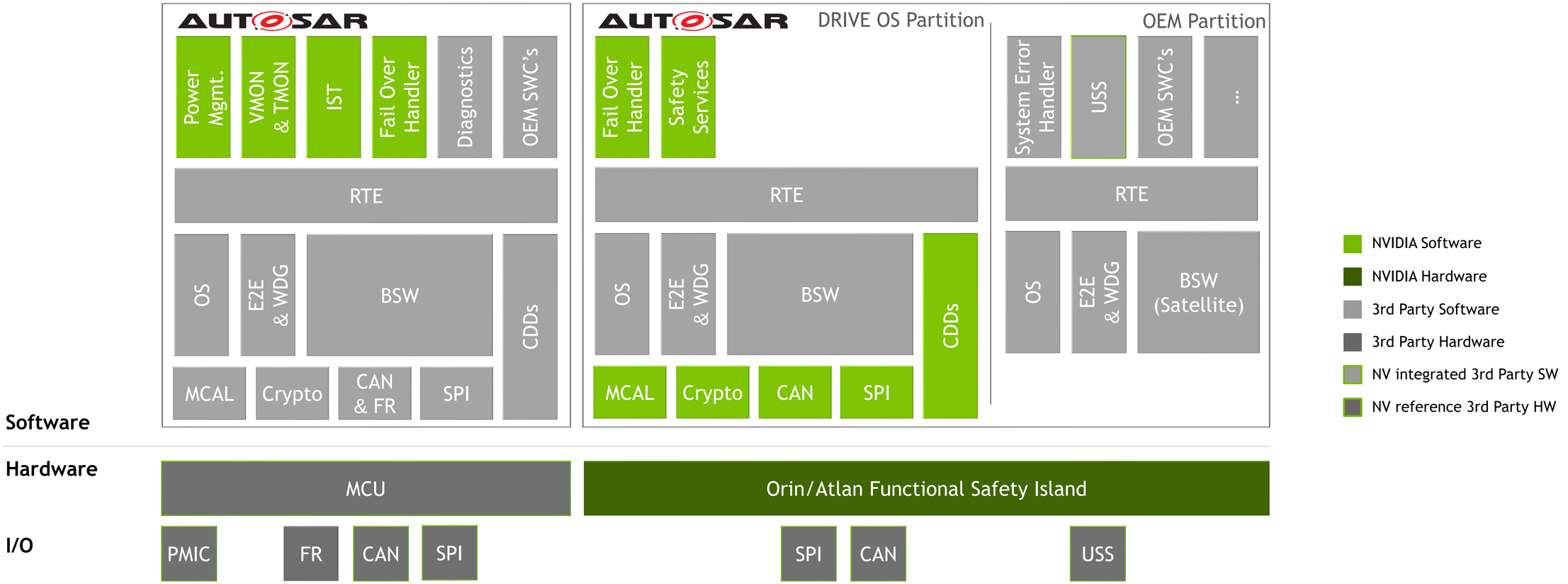
DRIVE OS LINUX

AV Software Platform with Safety Extensions



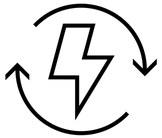
DRIVE OS

MCU & FSI Software for Safety



VIRTUALIZATION GUIDING PRINCIPLES

Tight Hardware and Software Integration for Isolation and QoS



FREEDOM FROM INTERFERENCE

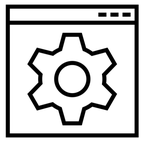
Absence of Cascading Failures that Could Lead to Violation of a Safety Requirement

STRICT RESOURCE ISOLATION

Protect Hardware and Memory Resources between Virtual Machines and Processes

GUARANTEED QUALITY OF SERVICE

Meet Real-time Requirements



MINIMIZE AND BOUND HYPERVISOR OVERHEAD

Maintain Native Performance in a Virtualized Environment

VIRTUALIZATION

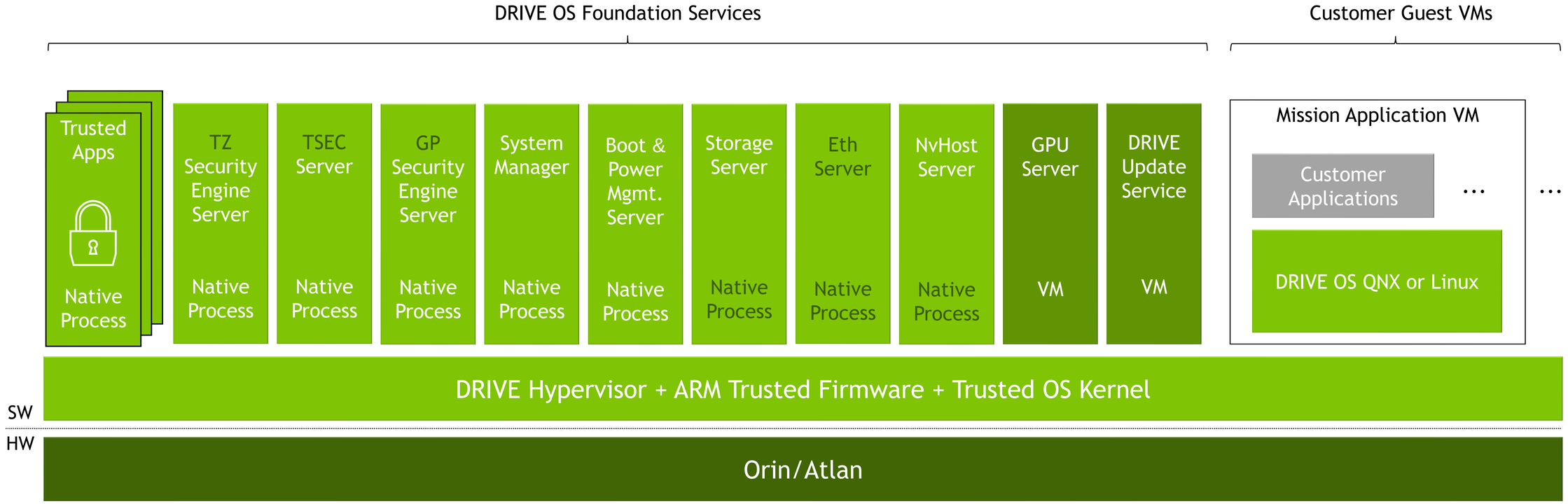
Industry’s Most Advanced

Built on NVIDIA’s own Type 1 Hypervisor

- Specifically designed to take advantage of Orin/Atlan hardware
- CPU virtualization with virtual CPUs support
- Static resource allocation (CPU, Memory, …)
- Strict resource isolation to protect hardware and memory
- Low overhead with near native performance
- Shipped in over 6M vehicles to date

VIRTUALIZATION ARCHITECTURE

Hypervisor Enables Safe, Secure and Real-Time Stack



Notes: Refer to DRIVE OS Safety Manual for safety context definitions.

Hypervisor Native Process

Virtual Machine

HYPERVISOR ARCHITECTURE

Hypervisor Kernel

HV RTOS provides core OS functionality and execution environment for Native Processes

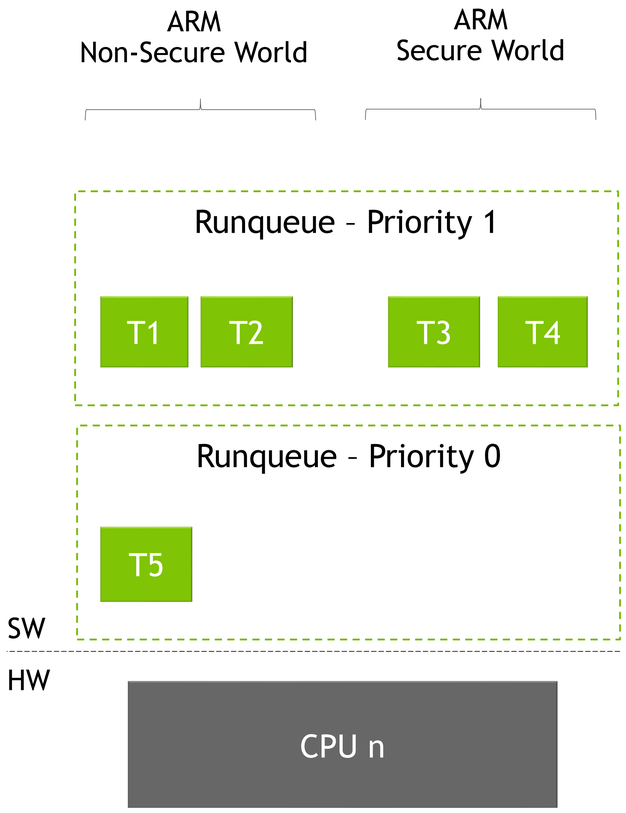
Hypervisor Kernel extends core HV RTOS functionality to support VM execution environment, manage hardware state that is shared across multiple VMs and provide infrastructure to handle per-VM state management by a Sidekick associated with each VM

- Handles CPU, Memory, and latency-sensitive Interrupt Controller emulation
- VM context management and VM CPU scheduling
- System initialization as specified by the Partition Configuration Table (PCT)
- Infrastructure to signals events between VMs, Native Processes and Sidekicks
- Shared memory setup for IVC communication between VMs, Native Processes and Sidekicks
- Handles SMMU Virtualization, enabling of some HW-assisted virtualization and resource partitioning, implementation of MMIO emulation, and error handling of global errors
- Fast-path optimization for high-frequency emulation (e.g., sending IPIs)

HYPERVISOR ARCHITECTURE

Scheduler Overview

- A Task is a fundamental execution unit and corresponds to either a Thread in a Native Process or a VCPU
- Scheduler implements preemptive, priority-based scheduling
  - Each Task runs on a dedicated CPU and never migrates to another CPU
- Scheduler exists in secure and non-secure worlds and shares the same number of priority levels
- Scheduling between the two worlds is co-operative within a priority level
  - Shared memory between the two worlds is used to convey event and scheduling state of the other world
- Interrupts assigned to lower-priority threads/VCPUs are prioritized below higher-priority threads/VCPUs
  - Leverages GIC HW for interrupt prioritization



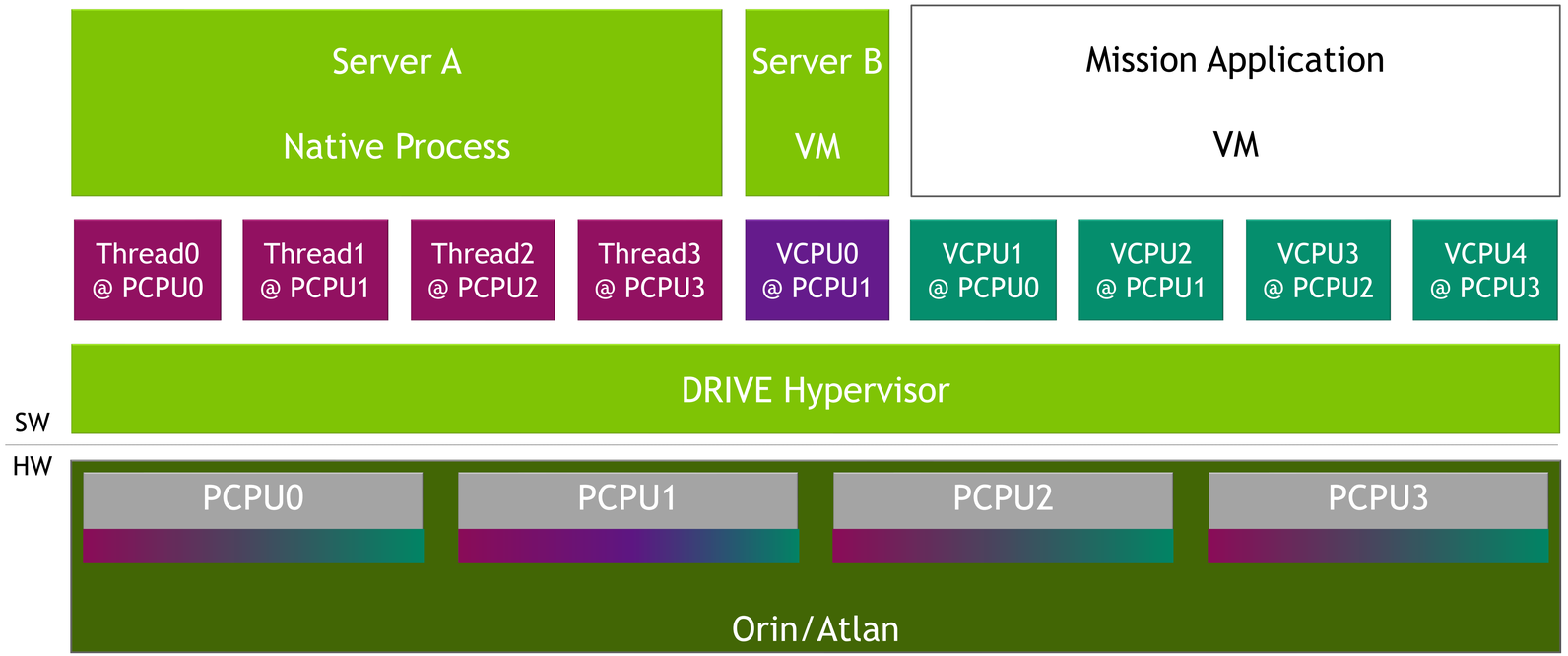
Execution Order

1. T1
2. T2
3. World Switch from NS-to-S
4. T3
5. T4
6. World Switch from S-to-NS
7. T5

CPU ALLOCATION & SCHEDULING

Illustration: 4-core SoC Allocation

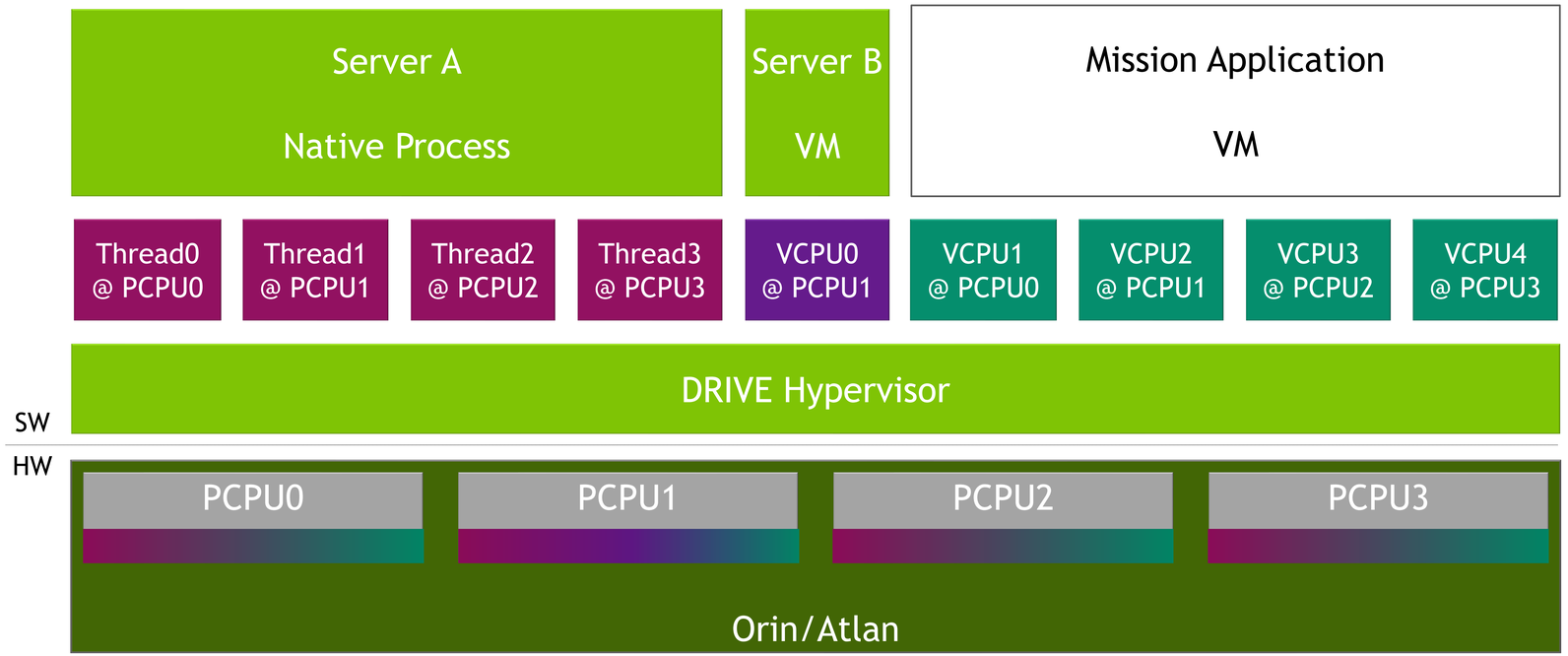
- Virtual Machines run on Virtual CPUs
- VCPUs or Native Process threads are statically assigned to physical CPU cores
  - N-to-1 mapping (VCPU/Thread to PCPU)
- The mapping is defined in the Partition Configuration Table (**PCT**)
- Hypervisor uses fixed-priority scheduling to time-share VCPUs/Threads on a single PCPU



CPU ALLOCATION & SCHEDULING CONT'D

Illustration: 4-core SoC Allocation

- Performance-critical Servers are multi-threaded and run on all PCPUs to provide better QoS
  - A client request then goes to the local PCPU thread, respectively
- Servers that are primarily dormant during runtime because they only handle boot-time init and/or error handling, remain single-threaded
- Virtualization Host Extensions will be used to enable efficient switching between VMs and Native Processes

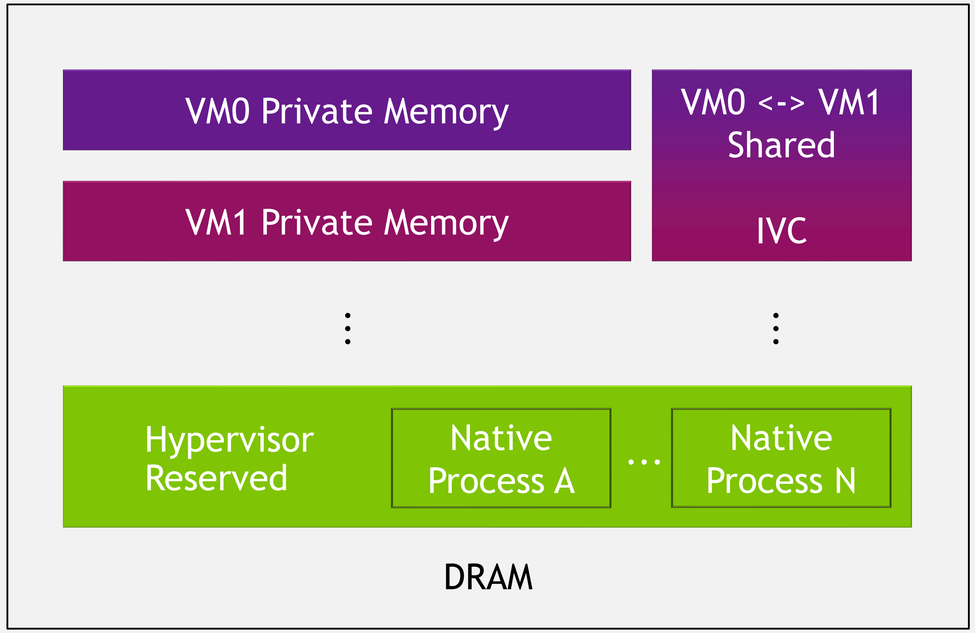


MEMORY ALLOCATION

Static Partitioning

DRAM is statically partitioned in the PCT

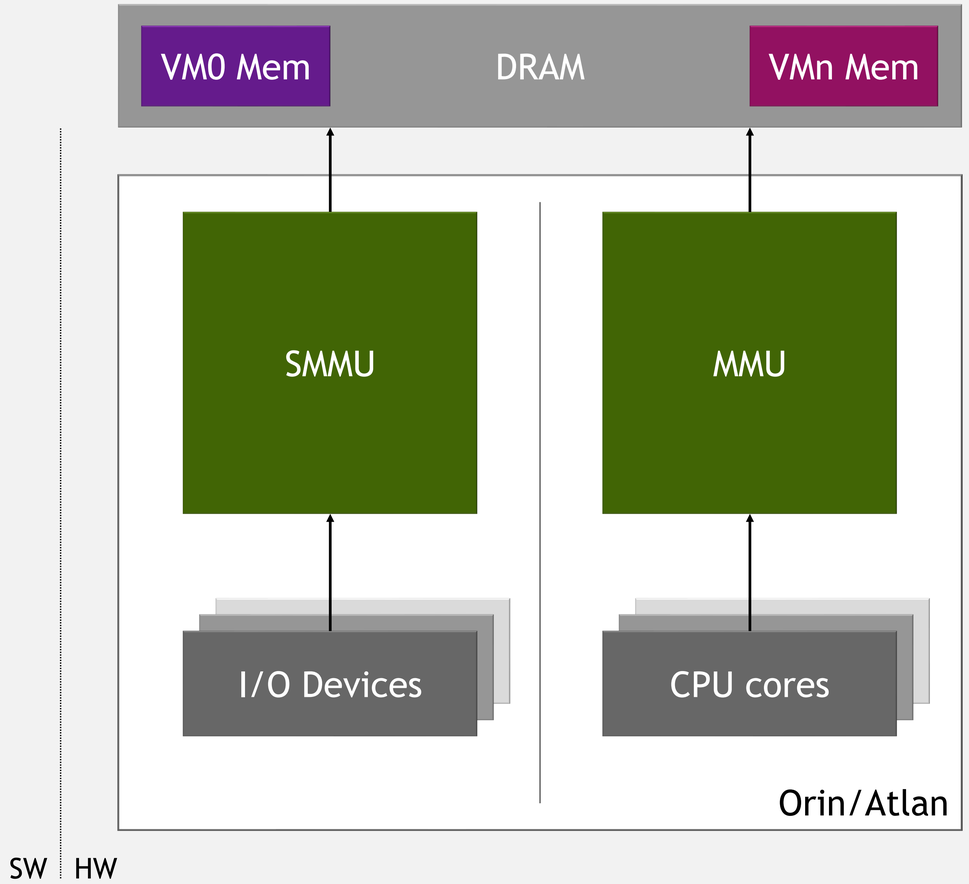
- Hypervisor reserved memory
  - Used to allocate Native Process memory and its internal usage
- VM private memory
- IVC memory (shared memory):
  - Queues
  - Mempools



MEMORY ISOLATION

Hardware-Assisted through MMU and SMMU

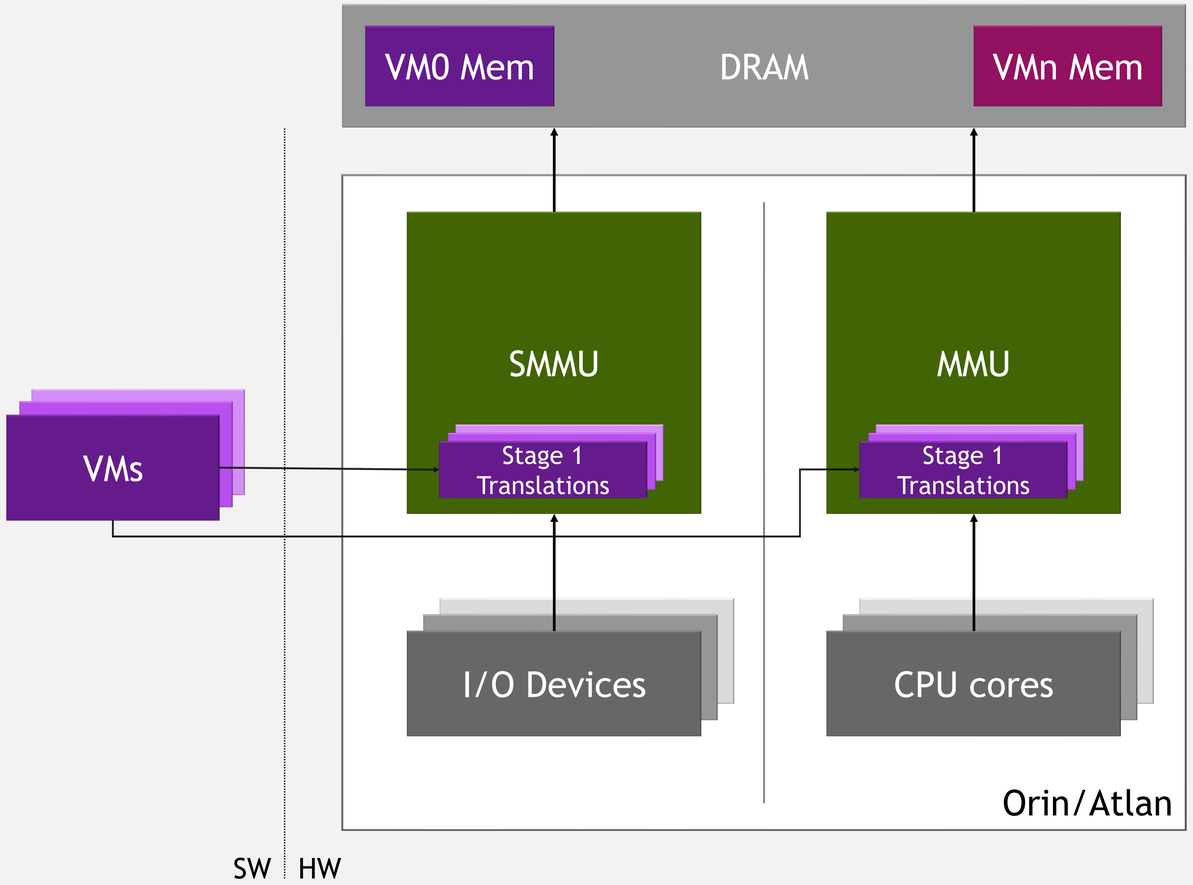
- Memory protection is enforced through virtualization-extensions in the SoC's **MMU** and **SMMU**



MEMORY ISOLATION

Hardware-Assisted through MMU and SMMU

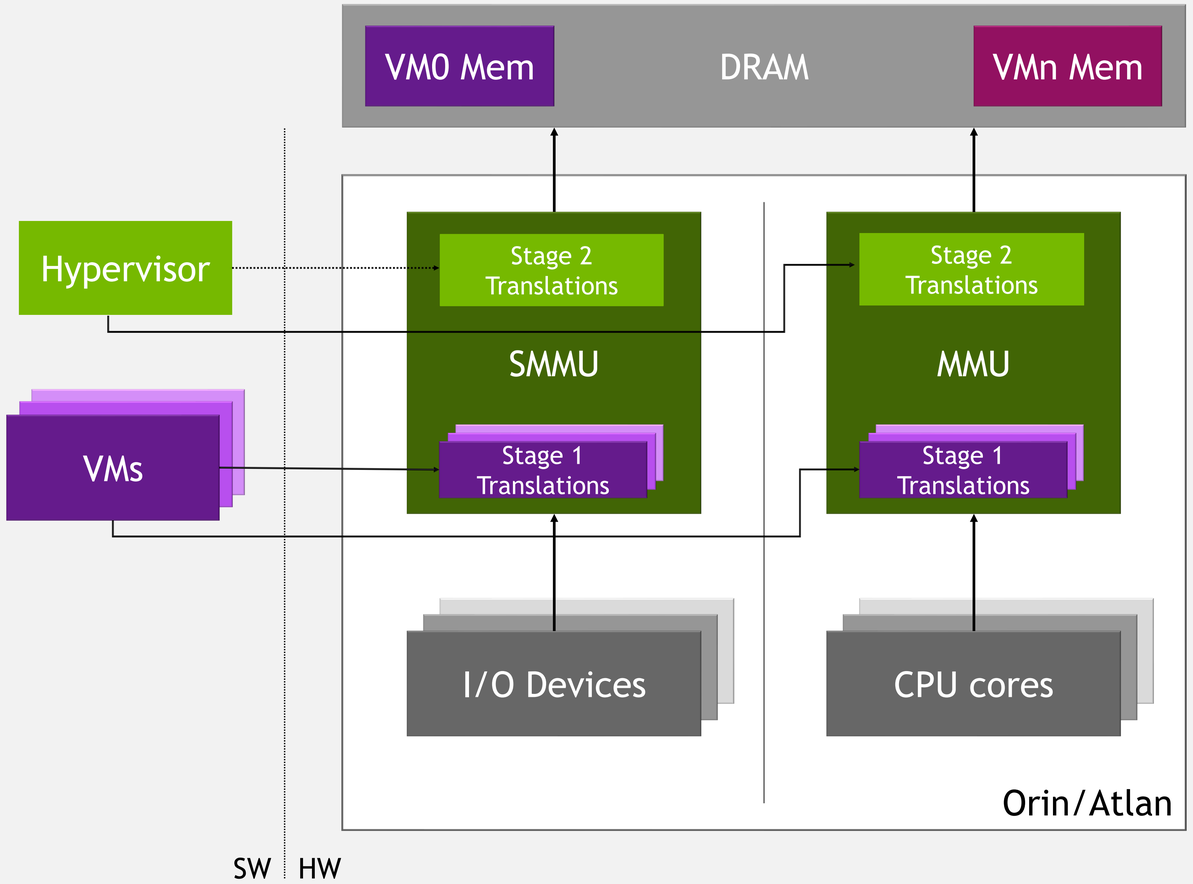
- Memory protection is enforced through virtualization-extensions in the SoC‘s **MMU** and **SMMU**
- Operating Systems inside VMs isolate their process memory through **Stage1** translation tables (like in a native system)



MEMORY ISOLATION

Hardware-Assisted through MMU and SMMU

- Memory protection is enforced through virtualization-extensions in the SoC‘s **MMU** and **SMMU**
- Operating Systems inside VMs isolate their process memory through **Stage1** translation tables (like in a native system)
- The Hypervisor isolates the VM‘s memory through **Stage2** translation tables



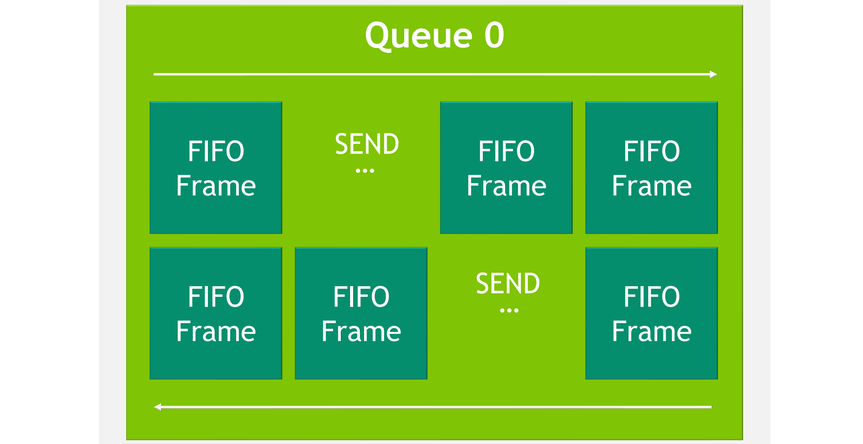
INTER-VM COMMUNICATION

Communication Between Two Virtual Machines

Inter VM Communication (IVC) is used to implement service requests to Virtualization Servers and Trusted Applications

ivclib API provides facilities for Inter-VM communication (IVC):

- Inter-VM channels are bidirectional and exist in shared memory region between 2 partitions
  - Each channel consists of 2 unidirectional FIFOs, one in each direction
  - Virtual interrupts for asynchronous notification of data availability and/or consumption
- Hypervisor is responsible for Memory management and signaling of Inter-VM channels
  - Trap-based signaling from EL0 to EL2 to optimize notification
- Each VM OS implements the Hypervisor Manager Driver responsible for discovering Inter-VM channels
- VM to Native Process notification is on the order of a few uS
- VM to VM notification performance is OS dependent



DEVICE ALLOCATION & SHARING

Support for all Industry-Standard Techniques

1. Static assignment of whole devices to a single VM or Native Process (**passthrough**)
2. Sharing of devices between multiple VMs and/or Native Processes
   - The following table gives an overview about the available techniques and typical device choices for each:

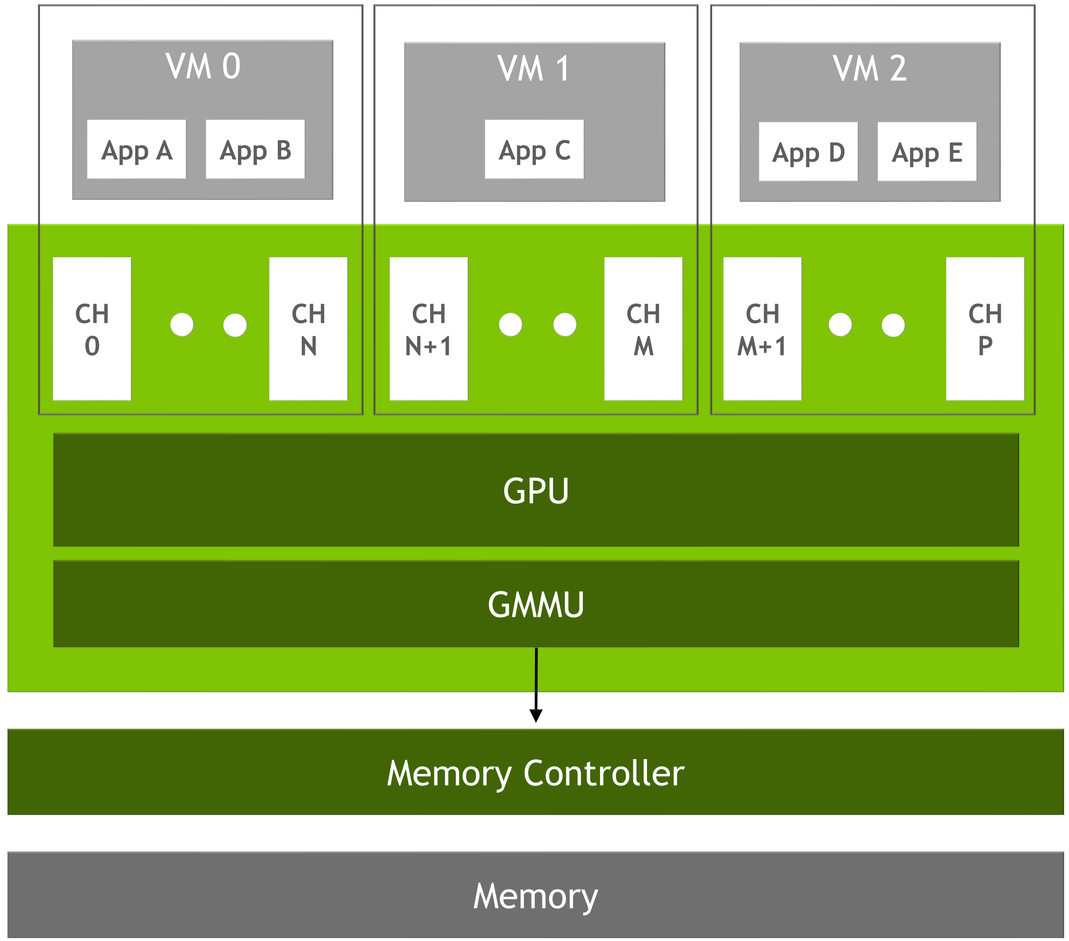
Device Performance	Shared		
	Trap & Emulate	HW-assisted, virtualization SW in critical path	HW-assisted sharing, no virtualization SW in critical path
Low	e.g. UART		e.g. GPIO
Medium		e.g. Storage via Storage Server (HW assistance for UFS only)	
High			e.g. GPU, capture, image encode/decode/processing, DL, Ethernet, USB

Note: Refer to DRIVE OS Safety Manual for safety context definitions.

NVIDIA GPU VIRTUALIZATION

Ideal Platform to Meet Isolation, Preemption and Realtime Scheduling

- Per Process and Per VM Isolation
- GPU partitioned amongst multiple VMs through channel assignment
- Applications on VMs assigned their own channels to minimize SW management of contexts and memory protection
- Channel Reset prevents rogue/malicious applications from hogging and crashing the GPU



AN ONSLAUGHT OF CYBERCRIME

The Crimes of the 21st Century

"Cybercrime will cost companies worldwide an estimated $10.5 trillion annually by 2025"

- https://www.embroker.com/blog/cyber-attack-statistics/

"FBI Warning: Hackers Now Targeting US Automotive Industry"

- https://www.cpomagazine.com/cyber-security/fbi-warning-hackers-now-targeting-us-automotive-industry/

"US truck and military vehicle manufacturer Navistar just disclosed it was targeted by a cyberattack"

- https://www.businessinsider.com/vehicle-manufacturer-navistar-cyberattack-sec-announcement-2021-6

"Car hackers demonstrate wireless attack on Tesla Model S"

- https://www.theverge.com/2016/9/19/12985120/tesla-model-s-hack-vulnerability-keen-labs

"Tesla Model X gets hacked through new relay attack"

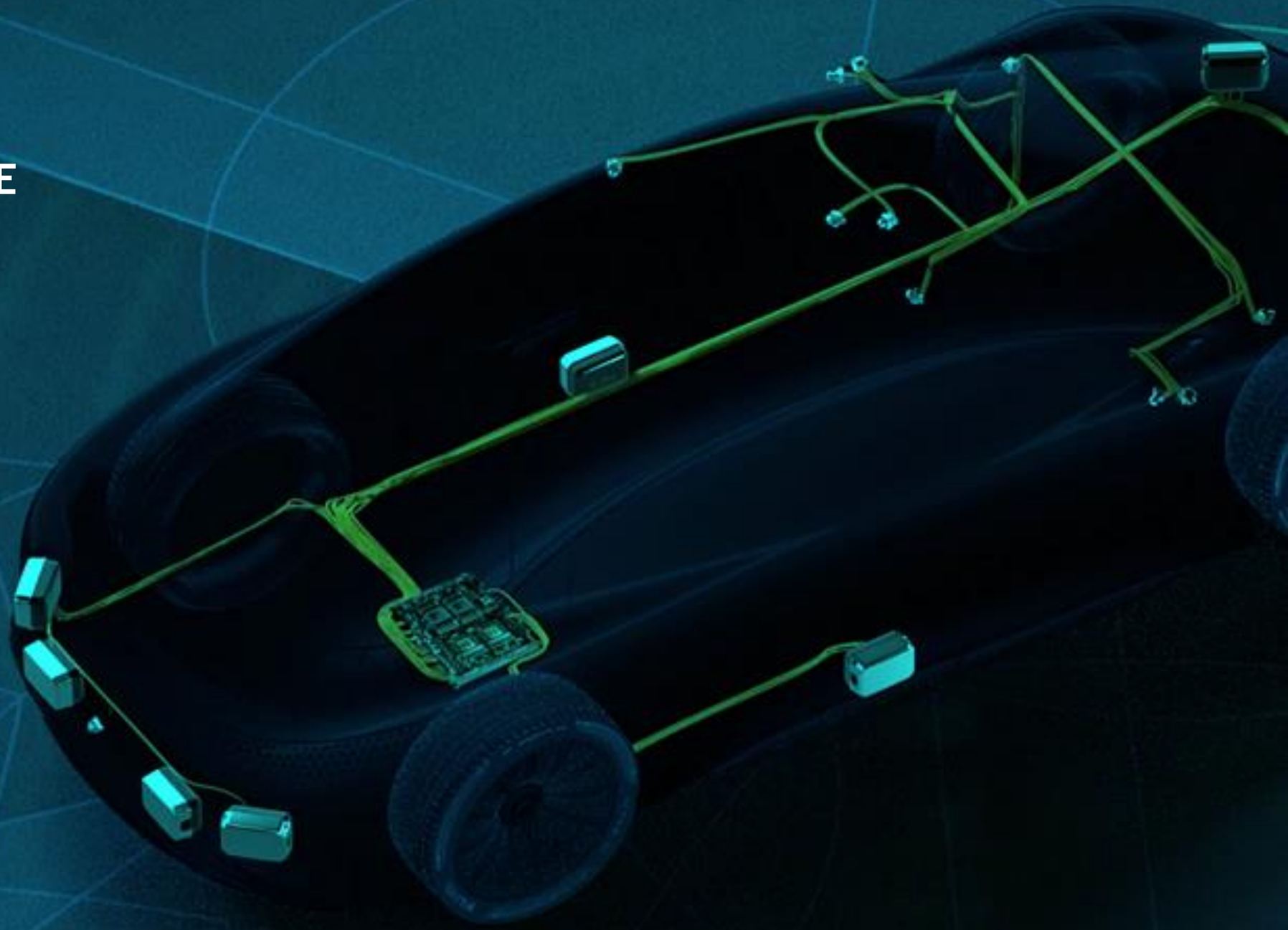
- https://electrek.co/2020/11/23/tesla-hacked-new-relay-attack-pushing-patch/

"Hacker claims Honda and Acura vehicles vulnerable to simple replay attack"

- https://hackaday.com/2021/08/30/hacker-claims-honda-and-acura-vehicles-vulnerable-to-simple-replay-attack/

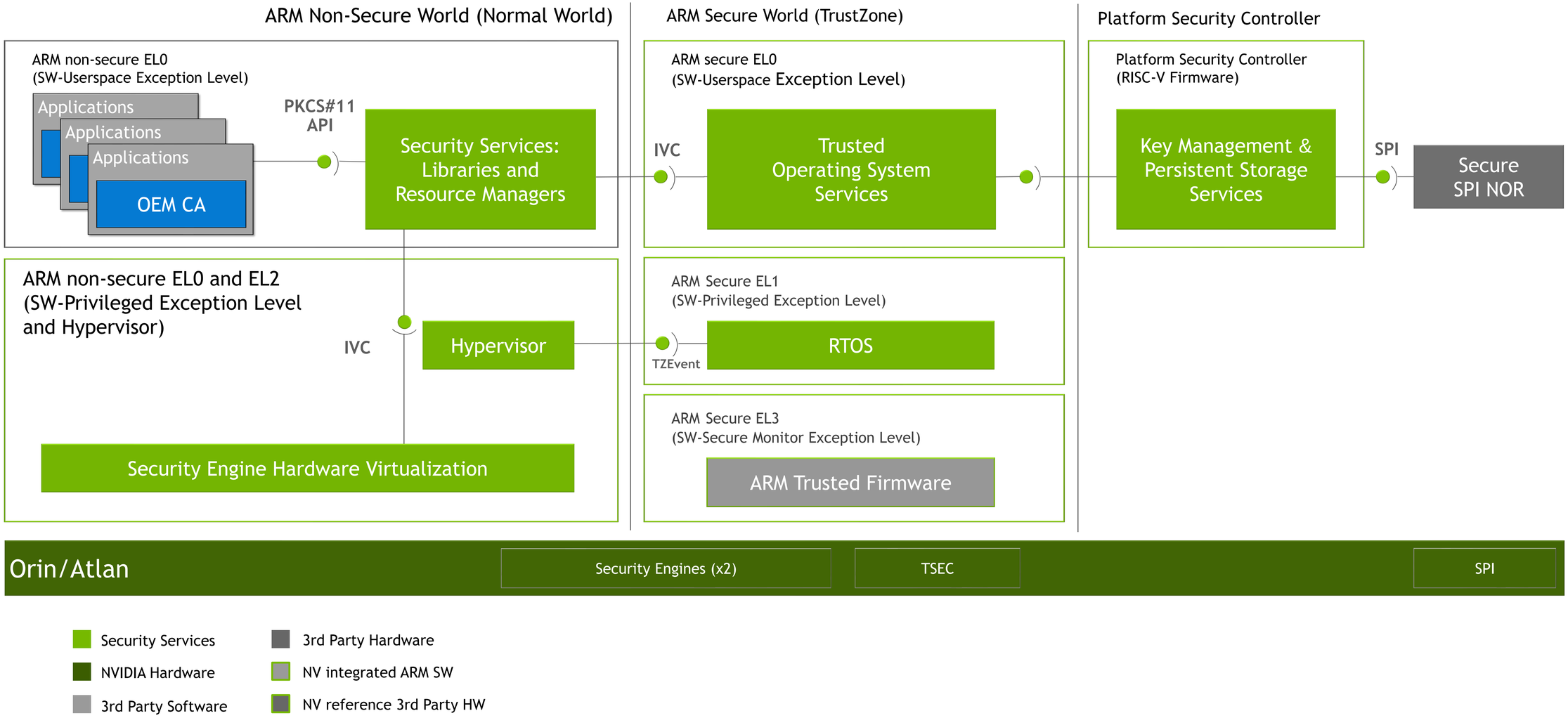
COMMON AUTOMOTIVE SECURITY THREATS

- Spoofing of Messages or Data
- Session Hijacking Replay Attack
- Denial of Service
- Malicious Messages
- Elevation of Privilege
- Compromised OTA Updates
- Manipulation of Data and Code
- Physical Attack



SECURITY SERVICES

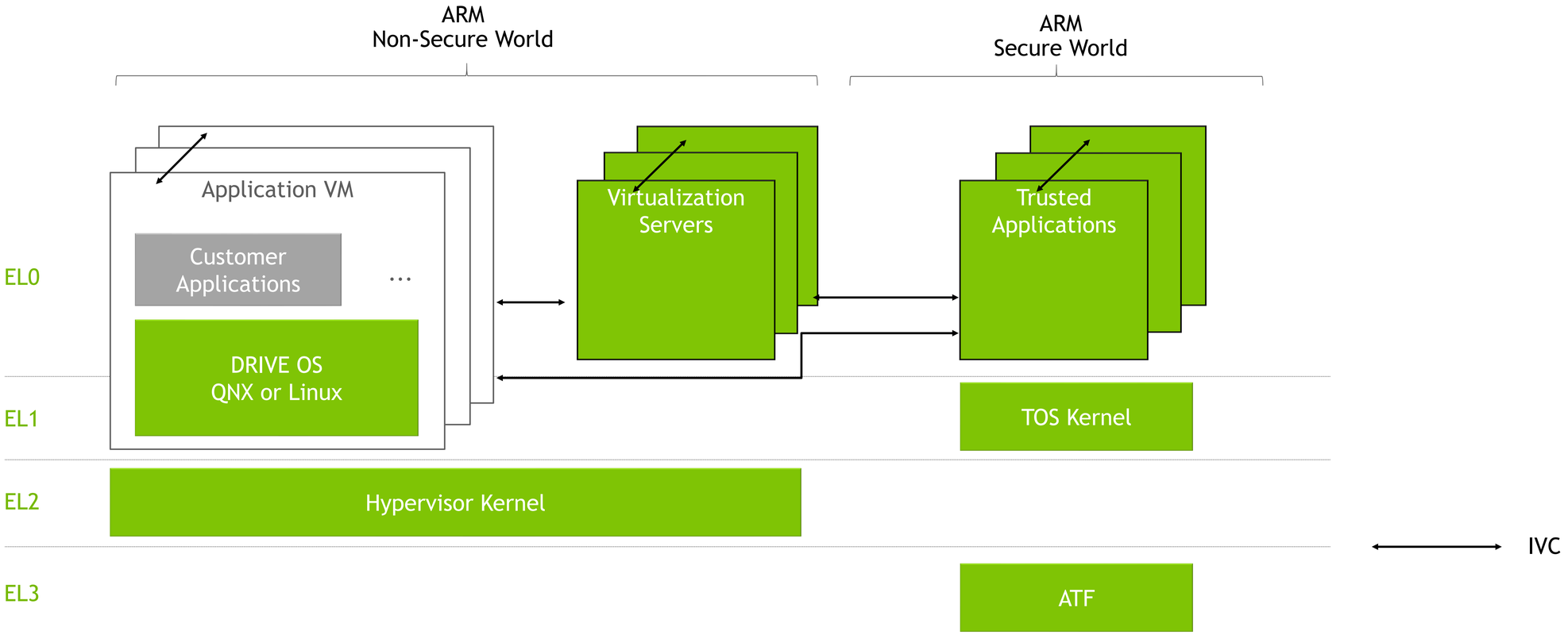
Overview



TOS ARCHITECTURE

Trusted OS Kernel

- The Trusted OS (TOS) Kernel extends core HV RTOS functionality to support Trusted Applications (TA)
- Performs TA initialization as specified by the TOS Configuration Table (TCT)



PKCS #11

Public Key Cryptography API

DRIVE OS implements the PKCS #11 3.0 API (Linux, QNX)

- Intends to be compliant with the extended provider profile
- See DRIVE OS PDK documentation for a list of APIs supported
- See docs.oasis-open.org for more information on PKCS #11 3.0

PKCS #11 Mechanism Support

- See DRIVE OS PDK documentation for a list of current mechanism supported
- Additions for DRIVE 6 include XMSS (verify only), AES-GCM, AES-GMAC, ECDH, SP800-108 derivation (HMAC), and SHA-3, EC signing

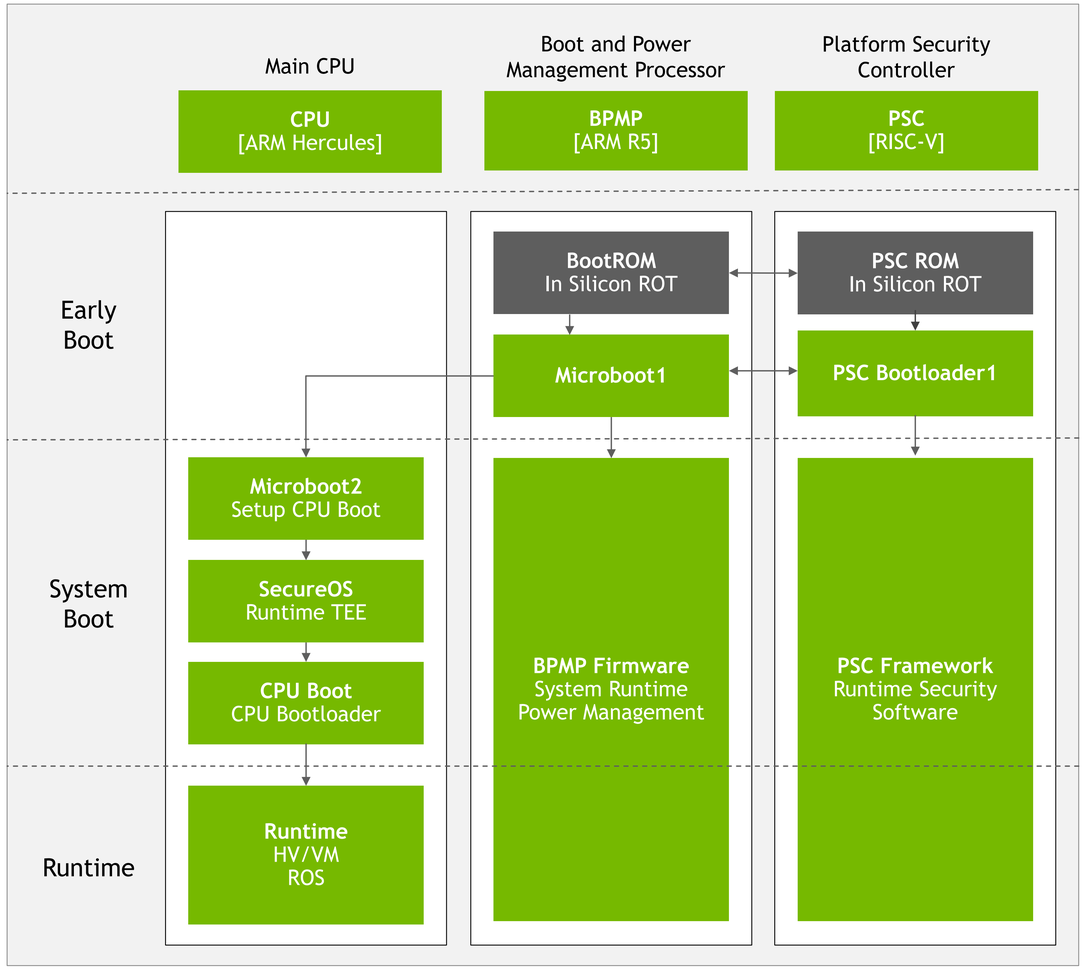
SECURE BOOT

Authentication Features

- Public-Key-Cryptography(PKC) based authentication of the images
  - Supported PKC algorithms:
    - RSA 3K, ECDSA (P-256 NIST), EdDSA (Ed25519), ECDSA NIST P-521, XMSS
  - Images can optionally be encrypted using 256-bit OEM AES key (SBK)
- Binaries in chain of trust are signed with OEM-Private-Key
- Various components in chain of trust authenticate binaries with OEM-Public-Key (and if required decrypt using OEM SBK) during image load from boot storage
- PSC is the root-of-trust and it validates OEM public key with hash value stored in the fuse
  - Fuse is write-protected once “security mode” is enabled

SECURE BOOT

Overview

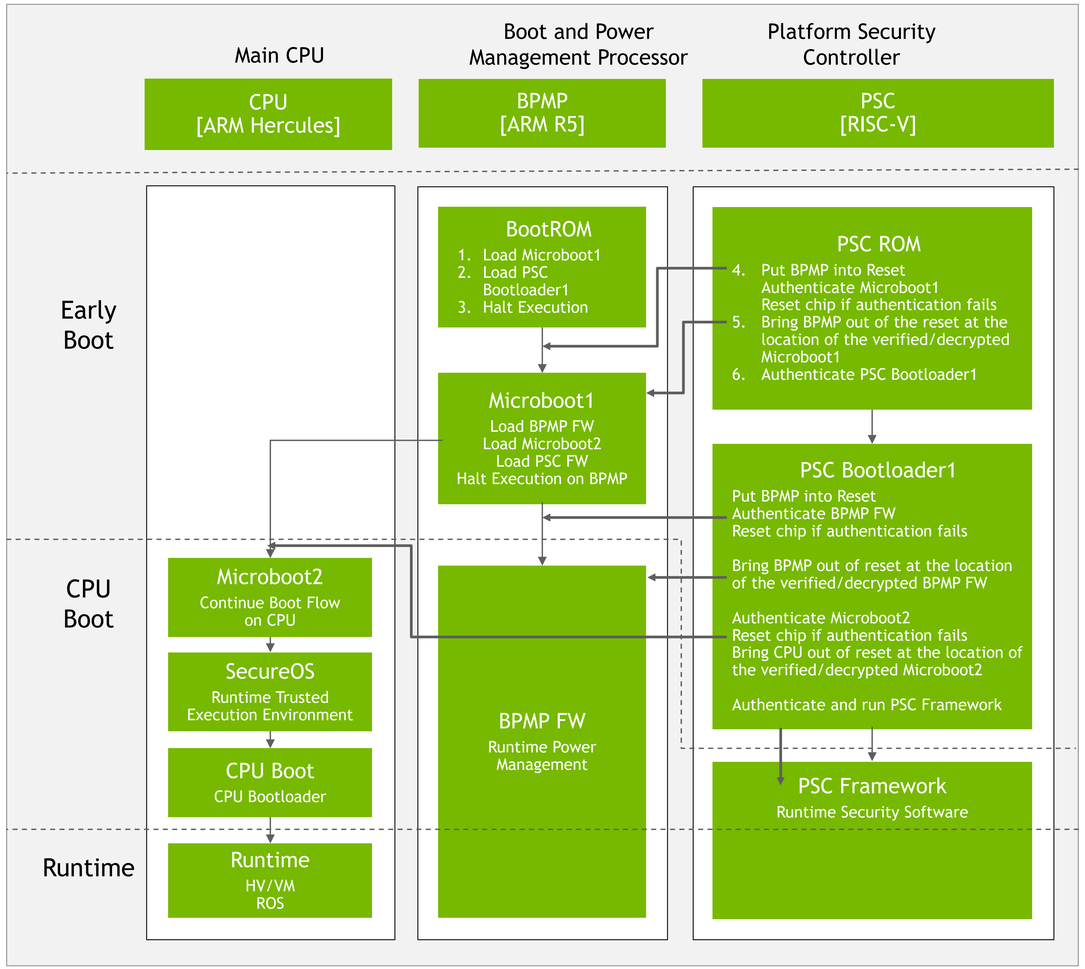


- BPMP and PSC roots of trust, based on on-chip boot ROMs
- Ability to hide the boot ROM in 4KB page level as the boot progresses
- Internal RAMs for secure execution
- Native support for encrypted boot loaders and firmware
- Software fault injection countermeasures
- Supports multiple authentication methods for chain of trust

Authentication	Encryption	Target Auth Perf
RSA-3K	AES-256	<1ms
ECDSA NIST P-256	AES-256	<1ms
EdDSA (Ed25519)	AES-256	<1ms
ECDSA NIST P-521	AES-256	<3ms
XMSS	AES-256	<40ms

SECURE BOOT

PSC Enforced Secure Boot



Purpose

Enforce an audited and authenticated boot flow for BPMP/CPU firmware up until the first Main CPU firmware component is loaded

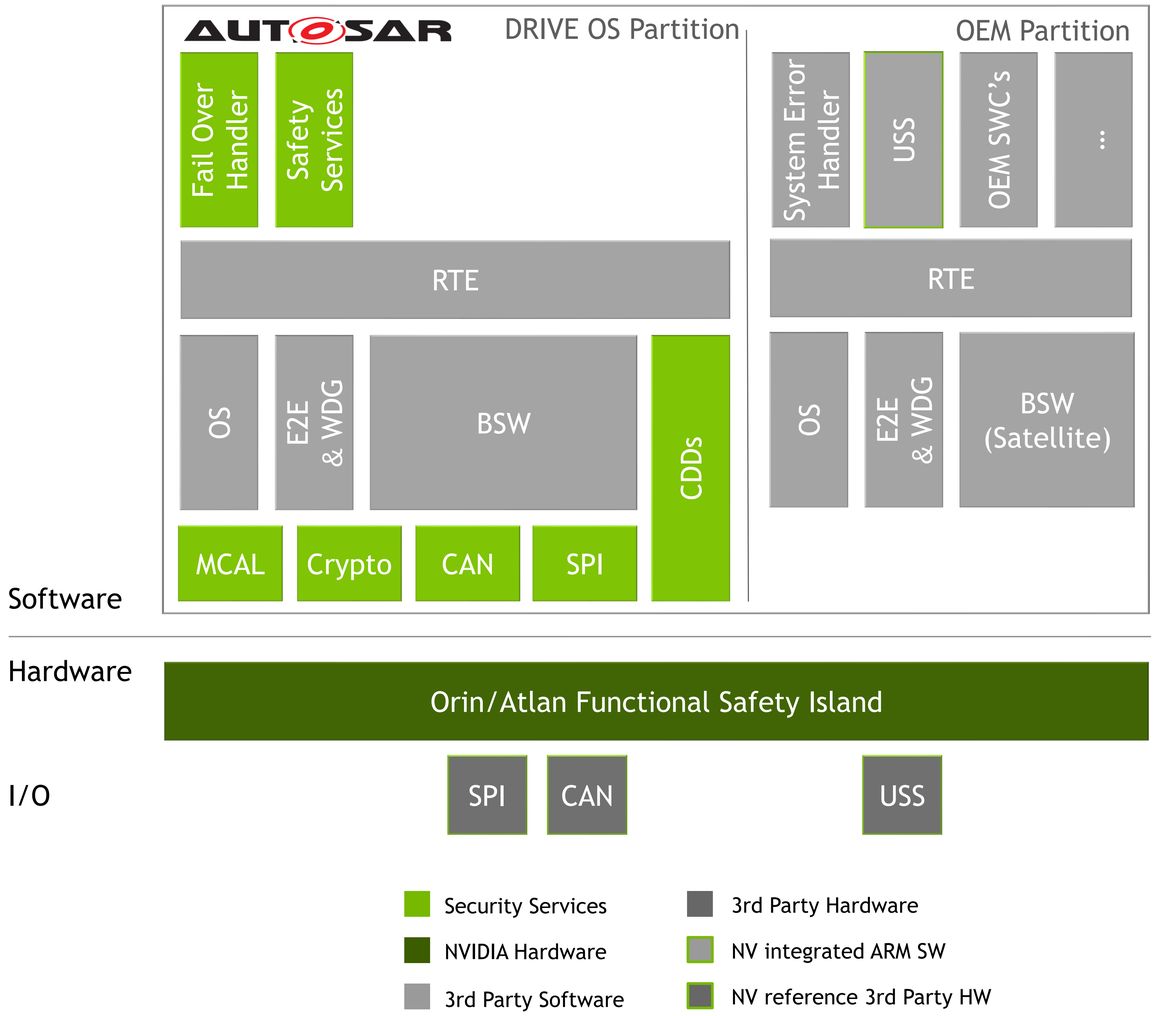
Details

- PSC handles authentication and decryption of all early-stage firmware. It has the smallest attack surface
- PSC places BPMP into reset while authenticating Microboot1 and takes BPMP out of reset at the location of Microboot1 ONLY if authentication succeeds
- PSC does the same for BPMP FW and CPU Microboot2
- Main CPU is in a non-functional state before PSC ROM finishes the enforced secure boot path
- Non-PSC masters cannot bring the Main CPU out of reset if BPMP is compromised during BootROM or Microboot1 stage

FSI SOFTWARE

for Partner AV

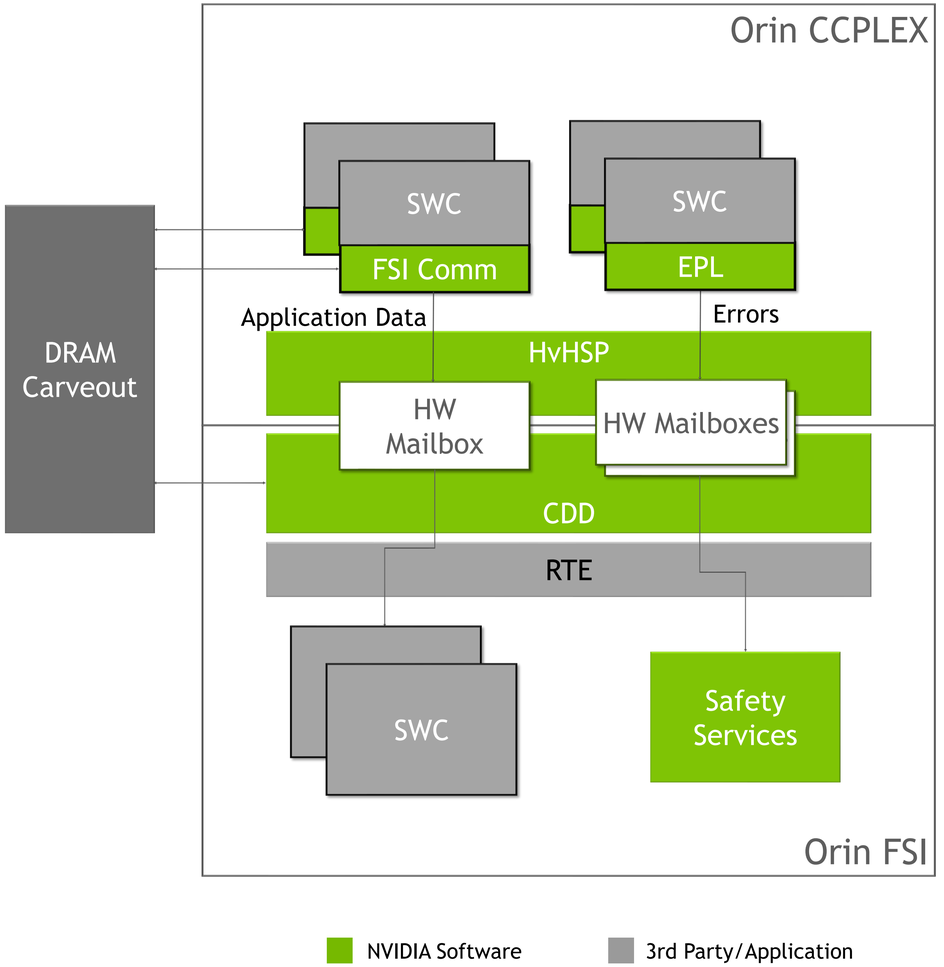
- Compliant with AUTOSAR Classic Platform Version 4.3
  - Safety Services
  - MCAL including drivers for Crypto, CAN, SPI
  - Complex Device Drivers
- All components are provided in source for integration into the customer's preferred AUTOSAR vendor
- DRIVE OS reference implementation is developed on Vector
- Binary image for provided for development on the DRIVE AGX development platforms
- Required for safety, optional for non-safety related



FSI COMM

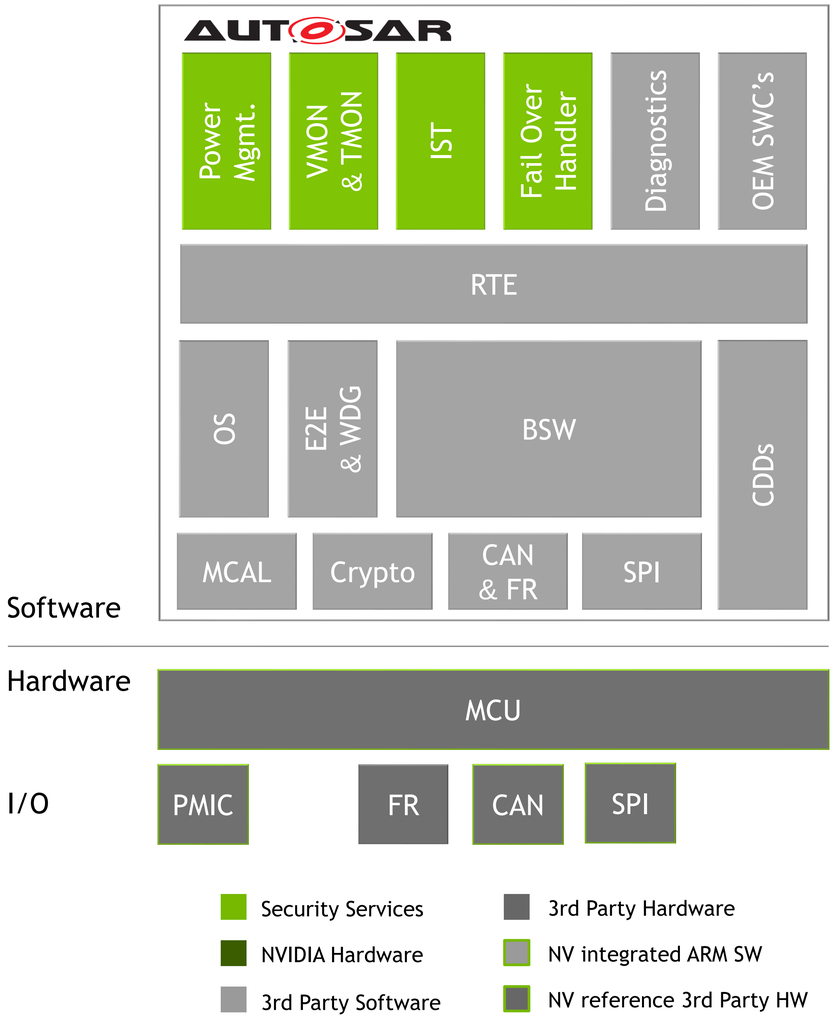
CCPLEX FSI Data Exchange

- Low latency communication utilizes HW mailboxes for fast synchronization between the CCPLEX and the FSI
- The Error Propagation Library (EPL) utilizes a dedicated HW mailboxes for safety-related SW error reporting from the CCPLEX to the FSI
- The FSI Comms library uses a HW mailbox along with a dedicated secure shared DRAM carveout for peer-to-peer communication between the CCPLEX and the FSI
  - The FSI Comms CDD provides RTE interfaces for send/receive ports to SWCs running on the FSI
  - Vendor-specific AUTOSAR Interop between Adaptive on the CCPLEX and Classic on the FSI can be realized on top of these interfaces



MCU SOFTWARE

- Compliant with AUTOSAR Classic Platform Version 4.3
  - Power Management
  - Voltage & Temperature Monitoring
  - IST*
  - Safety Services* (Fail Over Handler)
- All components are provided in source for integration into the customer's preferred AUTOSAR vendor
- DRIVE OS reference implementation is developed on Vector
- Binary image for Infineon Aurix provided for development on the DRIVE AGX development platforms
  - *Required for safety, optional for non-safety related



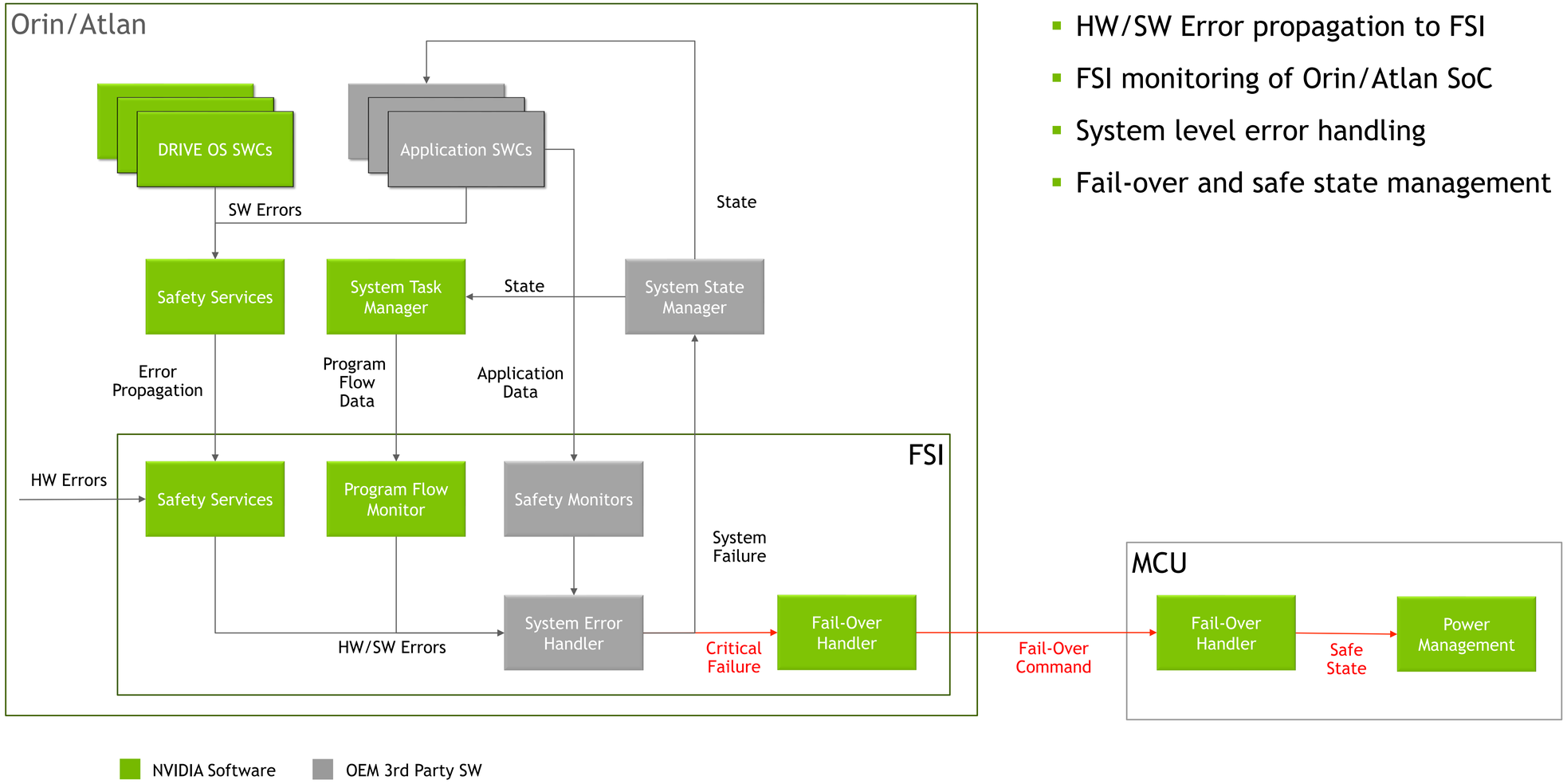
QNX SAFETY SERVICES & LINUX SAFETY EXTENSIONS

Overview

- Anchored by ASIL-D capable FSI on the DRIVE platform
- Streamlined mechanism for collecting and propagating HW and SW errors from Orin to FSI for error handling
  - Supports framework for system level error handling on the FSI
- Utilizes redundant HW mechanisms to report critical errors to external Safety MCU
  - Orin safe state management
  - Fail-over to external ECU

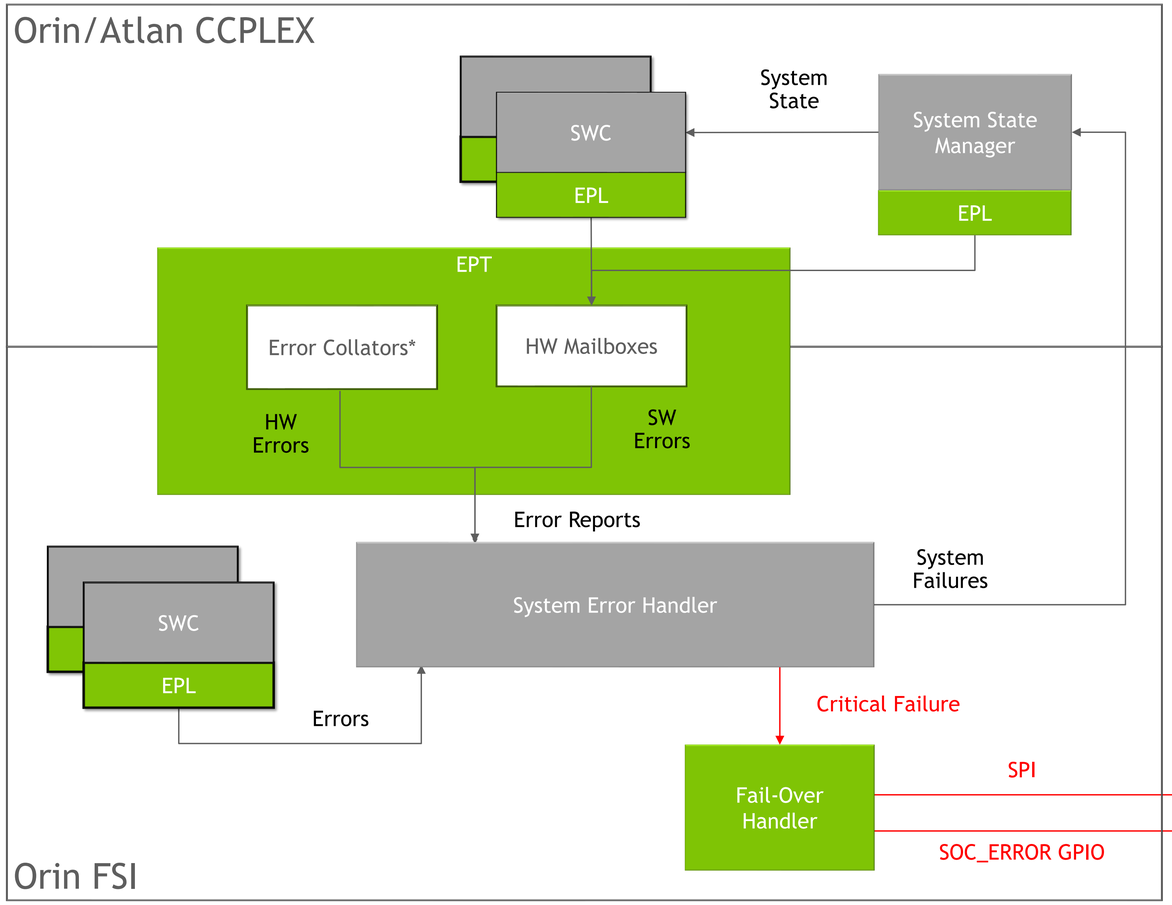
DRIVE OS SAFETY CONCEPT FOR AV

Key Safety Functions Supported in the DRIVE OS Safety Concept for Partner AV



- HW/SW Error propagation to FSI
- FSI monitoring of Orin/Atlan SoC
- System level error handling
- Fail-over and safe state management

ERROR PROPAGATION & HANDLING



Error Propagation:

- EPL Client used by all SWCs (DRIVE OS and application)
- Use of high-speed HW propagation mechanisms (EPT)
- Errors managed in central DB
  - Unique error IDs

Fail-Over Handler:

- Provides the HW shut-off mechanism used when Orin has been determined to be unreliable

System Error Handler:

- Application specific error handler
- Example place holder included in FSI AUTOSAR porting kit

HARDWARE DIAGNOSTIC SERVICES

Overview

- Hardware Diagnostic Services (HDS) support monitoring the health of Orin/Atlan SoC
- Provides supplemental HW diagnostic coverage
- Enables and confirms integrity of HW safety mechanisms
- Diagnoses HW errors through
  - Diagnostics directly embedded in drivers
  - Periodic diagnostics scheduled by application (provided as part of DRIVE OS)

HARDWARE DIAGNOSTIC SERVICES

Software Diagnostics

Startup — Diagnostics that must be executed and verified before ECU can enter mission mode

- e.g., Post boot checks

Periodic — Diagnostics that must be executed on a regular basis (frequency determined to meet Fault Detection Time Interval)

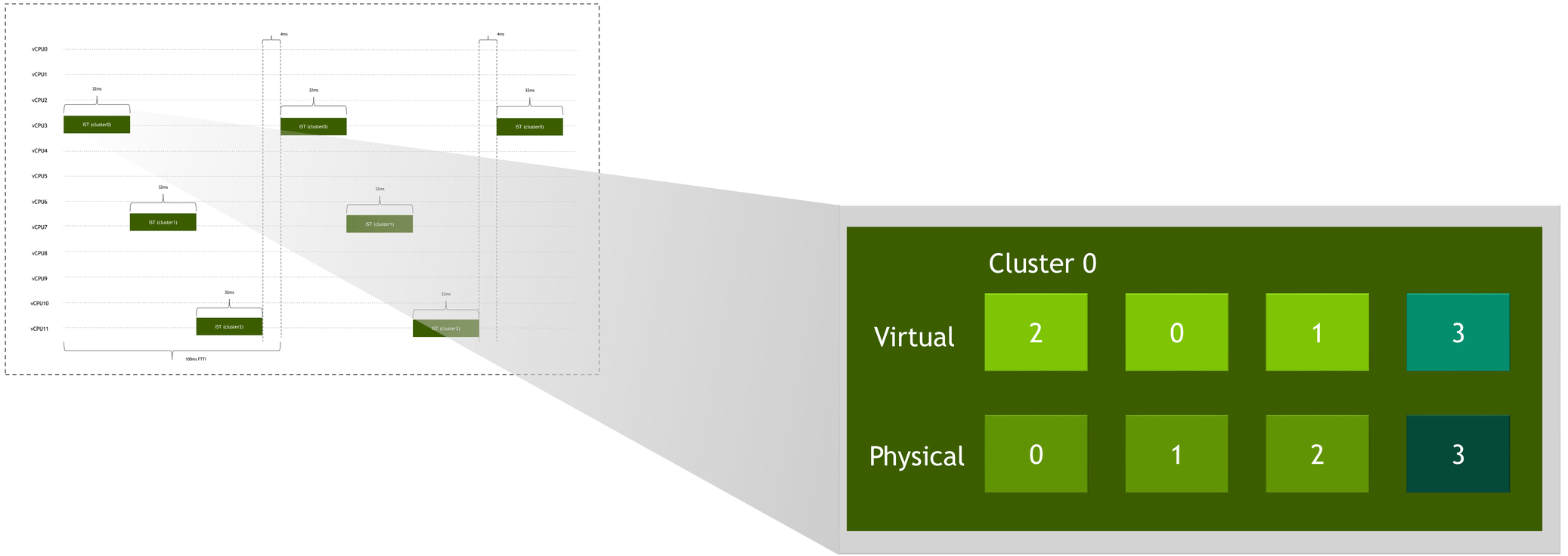
- e.g., Permanent fault diagnostics for hardware engines
- Scheduled by the Application

Shutdown — Diagnostics that will be executed at shutdown; Results are persistent so they are available upon next boot

- e.g., IST, SDRAM full latent fault test

HARDWARE DIAGNOSTIC SERVICES

Runtime IST



- Orin/Atlan's run-time IST can be SW triggered periodically to do a structural test of the CPU for permanent fault coverage
- Scheduler initiates the diagnostic sequence on each cluster so it knows when not to schedule threads to the offline vCPU
- DRIVE OS will internally migrate the diagnostics across the 4 physical cores of the cluster in order to test each core
- Run-time IST impacts overall performance available for the application. Executes on each CPU core and the diagnostic is not preemptable while running

DRIVE QNX FOR SAFETY

Components ASIL & Tools Confidence Levels

DRIVE Runtime Components	DRIVE OS 5.2	DRIVE OS 6.0	DRIVE OS 6.1
DriveWorks	B	B(D)	B(D)
TensorRT	B	B(D)[2]	B(D)
CUDA Runtime	B	B(D)[2]	B(D)
Vulkan SC	-	-	QM
NvMedia	B	B(D)	B(D)
NvStreams	B	D[2]	D
NvDisplay	QM	A	A
NvSocket	QM	B(D)	B(D)
QNX File System, Network stack	QM(B)	QM(D)	QM(D)
QNX BSP for Tegra	B	D	D
QNX Microkernel	B	D	D
DRIVE Hypervisor	B	D	D
Foundation Services	B	D	D
DRIVE Update	QM	QM(B)	QM(B)
Security Services	B	D	D
Safety Services	B	D	D
Bootloader	B	D	D

Safety MCU Runtime Components	DRIVE OS 5.2	DRIVE OS 6.0	DRIVE OS 6.1
Power Management	B	D	D
Safety Services	B	D	D
Voltage/Temperature monitor	B	D	D

Toolchains	DRIVE OS 5.2	DRIVE OS 6.0	DRIVE OS 6.1
QNX® Momentics® Tool Suite[3] (C compiler, linker, and assembler)	3	3	3
NVCC – CUDA compiler	3	3	3
TensorRT Builder	1	1	1

NVIDIA Developer Tools[1]			
CUDA-gdb – CUDA debugger	1	1	1
CUDA memcheck	1	1	1
Nsight Compute™	1	1	1
NVPROF	1	1	1
CUPTI – CUDA Profiling Tools Interface	1	1	1
Nsight™ Eclipse Edition	1	1	1
Nsight Systems™	1	1	1

Notes:
1) For development only, not for use in production in a safety context.
2) PG199/A100 for development only, not for use in production in a safety context.
3) BlackBerry QNX® Momentics® Tool Suite provided separately from QNX.
4) Refer to the QNX Safety Manual and the DRIVE OS Safety Manual for safety context definitions.

DRIVE OS ECOSYSTEM

Just a Few of Our Partners

SOFTWARE

HARDWARE & SENSOR

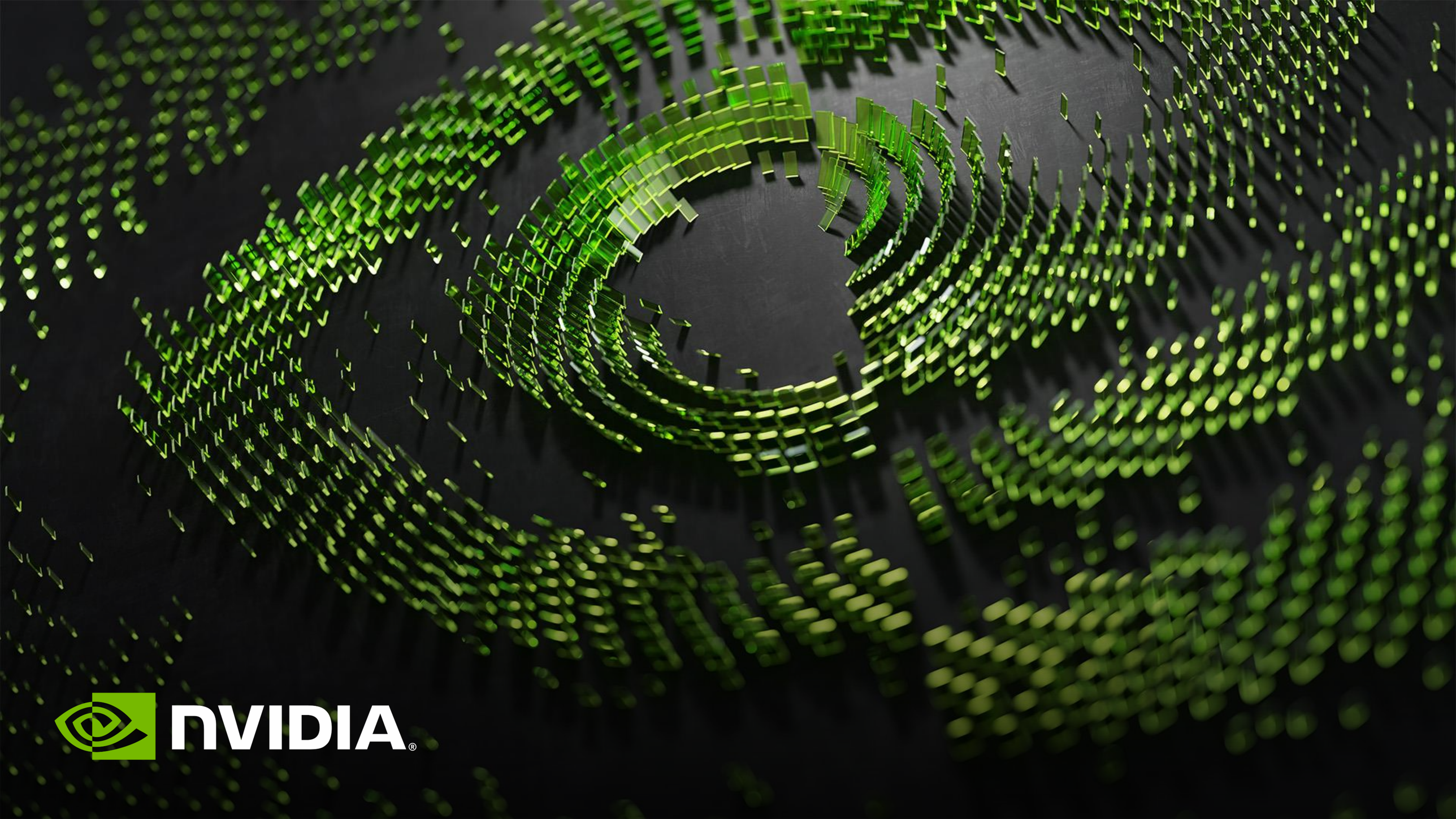
See https://www.nvidia.com/en-us/self-driving-cars/partners/ for full list of NVIDIA DRIVE Partners

DRIVE OS Distribution Model

- NVIDIA and BlackBerry QNX closely collaborate to produce DRIVE OS QNX
- NVIDIA distributes DRIVE OS QNX to Customers (via partners.nvidia.com) and is the first line of Support
- Customer must acquire the following Development licenses from BlackBerry QNX
  - SDP 7 (User or Floating Licenses)
  - QOS 2 (Multi-User Project License(s))
- Customer must acquire production runtime licenses from BlackBerry QNX
  - QOS 2
  - Libc++ (ASIL D)

DRIVE OS Distribution Model

- NVIDIA AUTOSAR Classic Platform components are compliant with AUTOSAR Classic Platform 4.3
  - Developed with Vector MICROSAR, however may be used with any vendor’s compliant AUTOSAR Classic Platform
- Binary images are provided for both the NVIDIA Orin FSI and the Infineon Aurix MCU for development on the DRIVE AGX development platforms
- Customer must acquire AUTOSAR development tools and licenses to modify and adapt these for binaries for their hardware, application, and vehicle integration
- Customer must acquire production licenses for
  - AUTOSAR Classic for the FSI
  - AUTOSAR Classic and MCAL for the MCU


NVIDIA