SYSTEMATICALLY SECURING THE RISC-V

MARKO MITIC, 12/8/21

“NVIDIA PROVIDES THE CORE TECHNOLOGY RESHAPING INDUSTRY AND SOCIETY”

DIGINOMICA—April 14, 2021

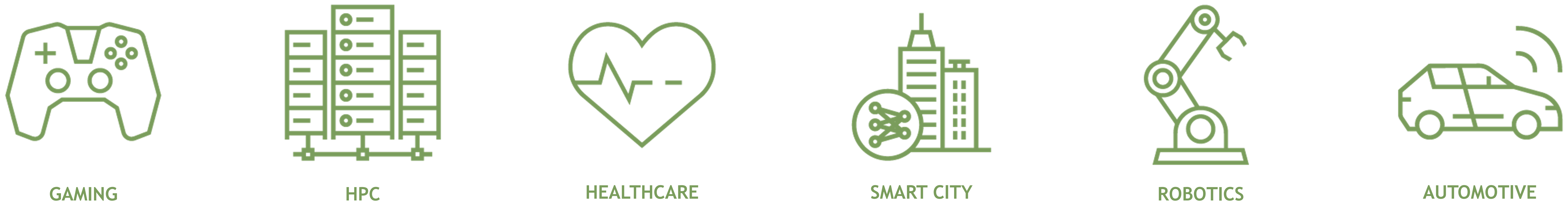
NVIDIA relies heavily on embedded controllers (24+) for most critical applications

RISC-V®

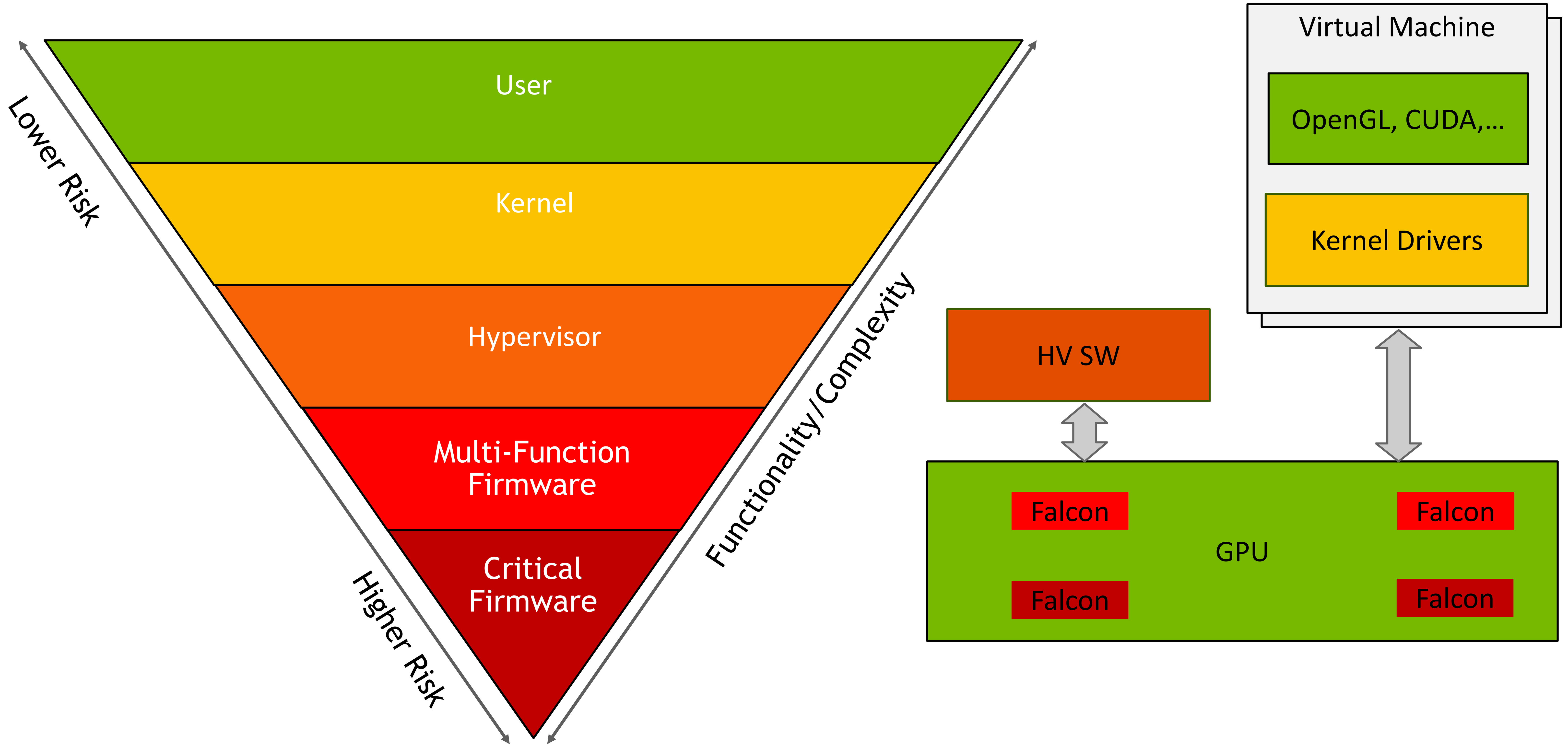
NVIDIA

CHALLENGES

Products demanding more Complexity and Security

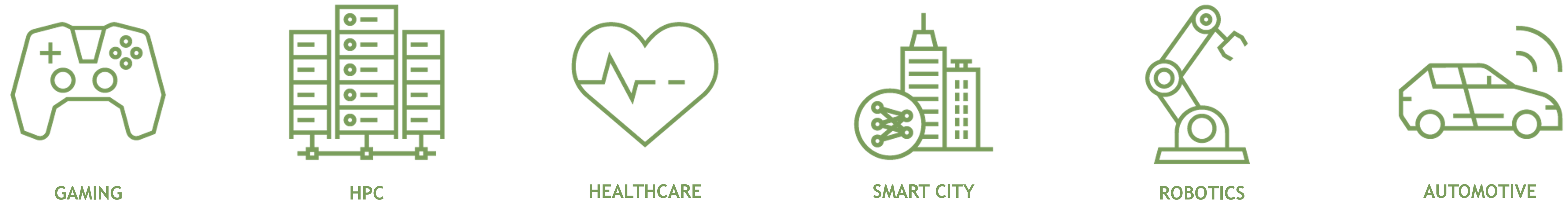


- NVIDIA demanding more flexible HW
  - Different products from same silicon
- Long lived products with ongoing development
- Growing complexity pushed into microcontrollers
  - Risk of a compromise is higher
- Expensive security incidents



CHALLENGES

Wide Security Vulnerability Spectrum

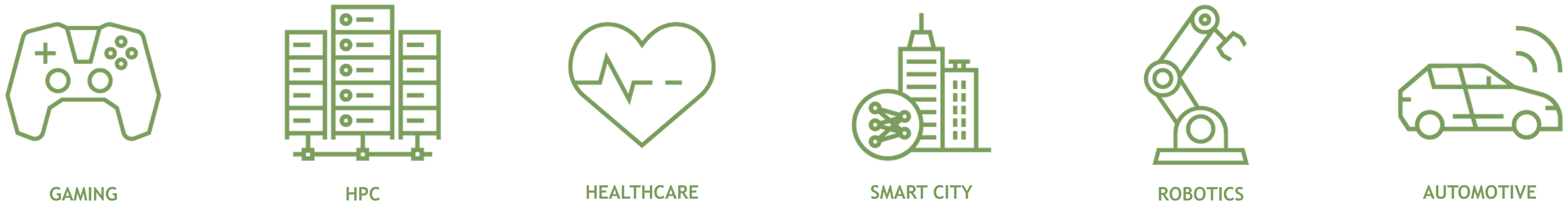


- Embedded GPU microcontrollers form security foundation for all GPU Products
- Falcon challenges
  - Modal secure/insecure world switch
  - Coarse HW isolation capabilities

NVIDIA

CHALLENGES

Wide Security Vulnerability Spectrum

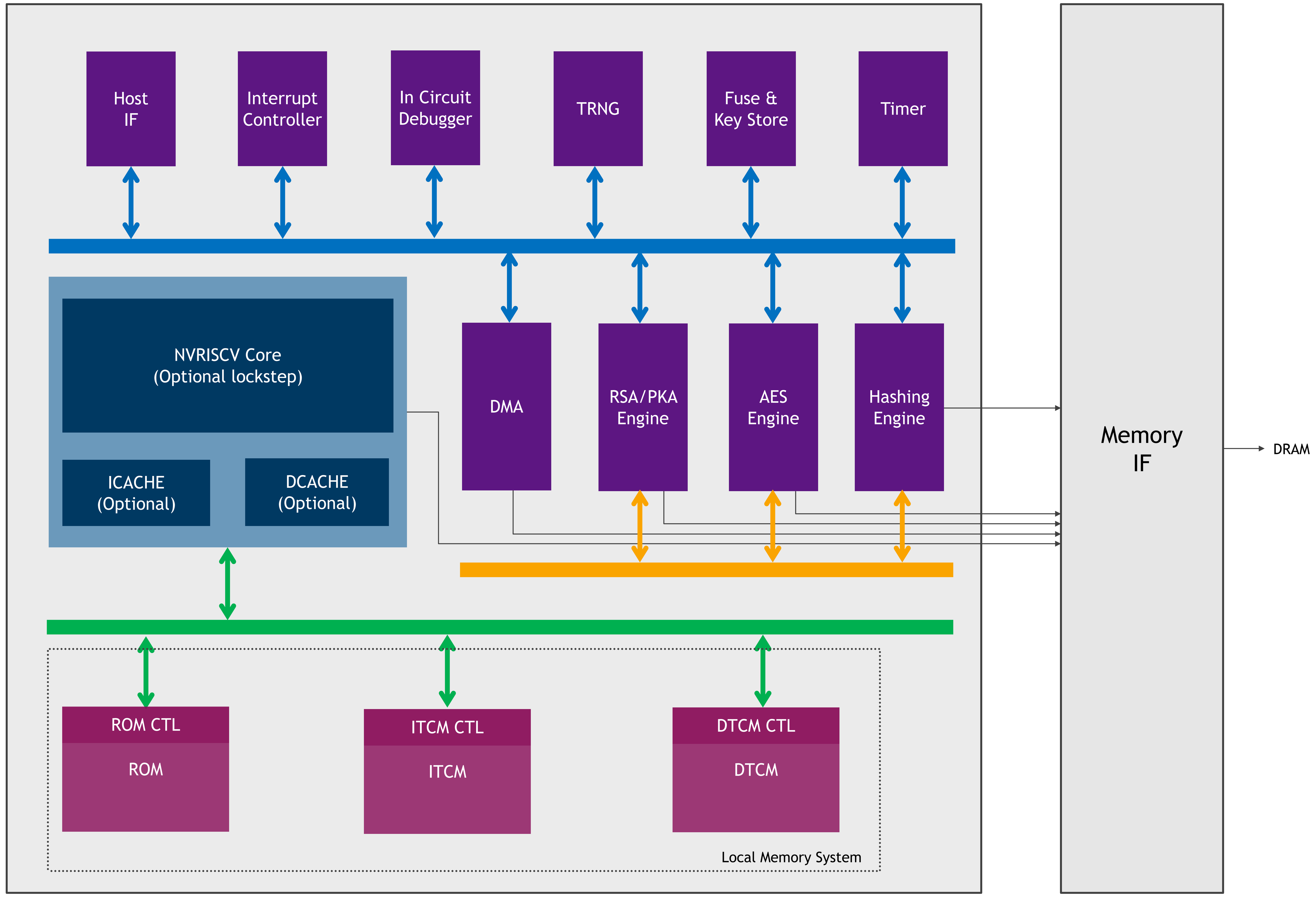


- Embedded GPU microcontrollers form security foundation for all GPU Products
- Falcon challenges
  - Modal secure/insecure world switch
  - Coarse HW isolation capabilities

NVRISCV & PEREGRINE

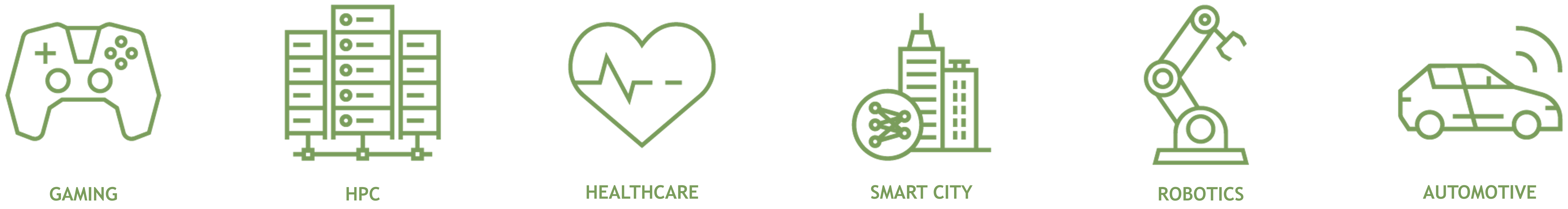
- RISC-V is free and open Instruction Set Architecture
- Standard maintained by RISC-V Foundation

- NVRISCV is NVIDIA's implementation of the RISC-V ISA
- Peregrine is the CPU + Peripherals

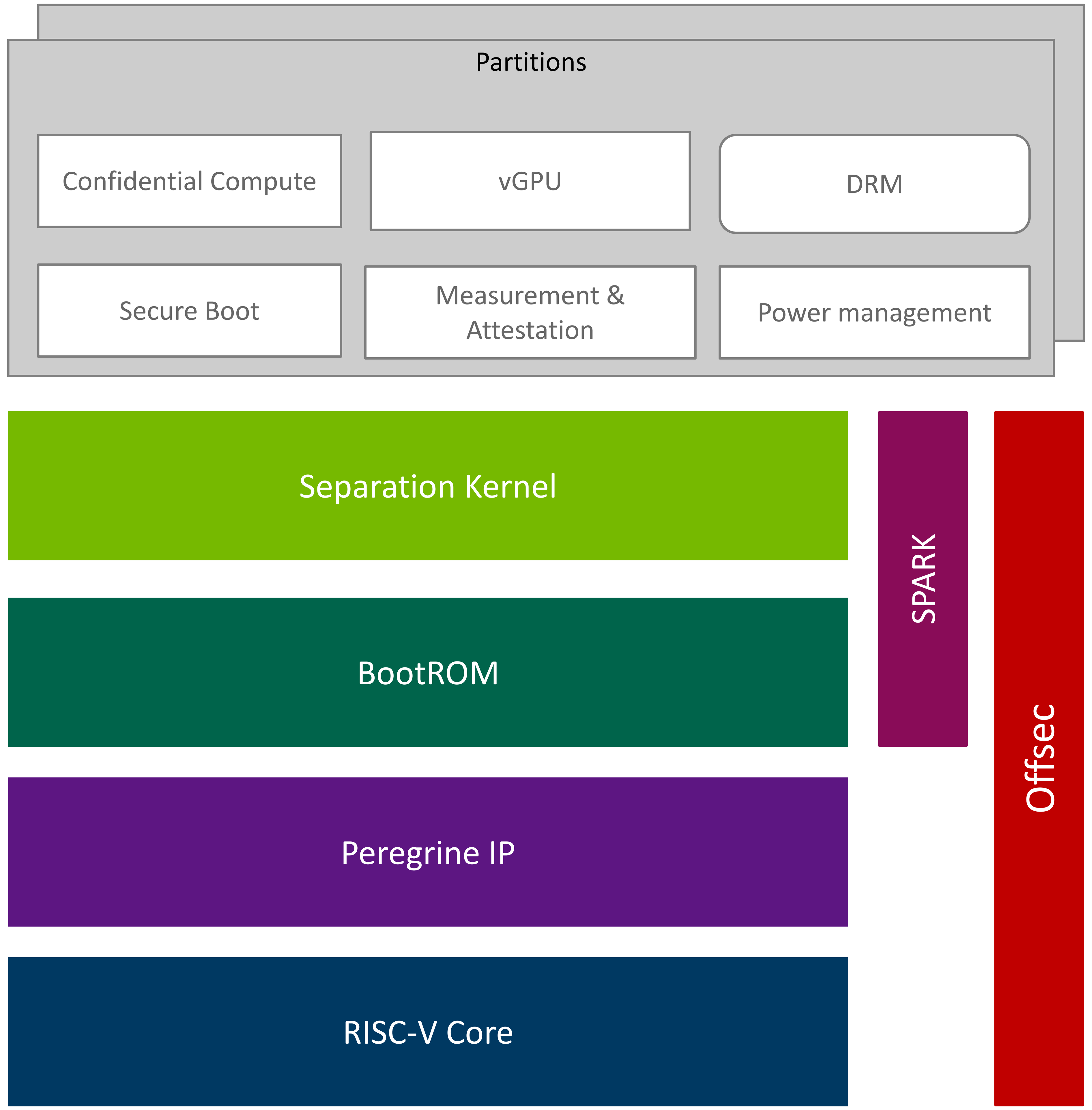


STRATEGY

Goal: Contain complexity and manage risk securely

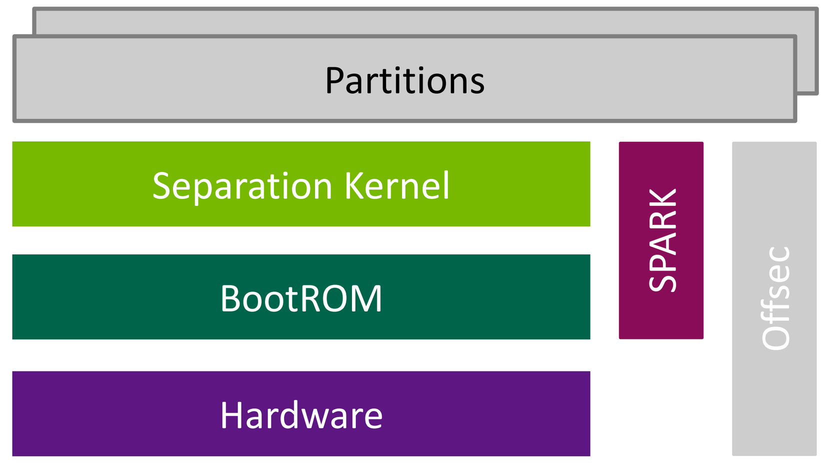


- Build secure foundation for complex and evolving features
- Leverage HW/SW codesign
- Layer and decompose software
- Invest in robust SW development tools and techniques
- Invest in in-depth offensive security efforts
- Leverage critical security functionalities everywhere

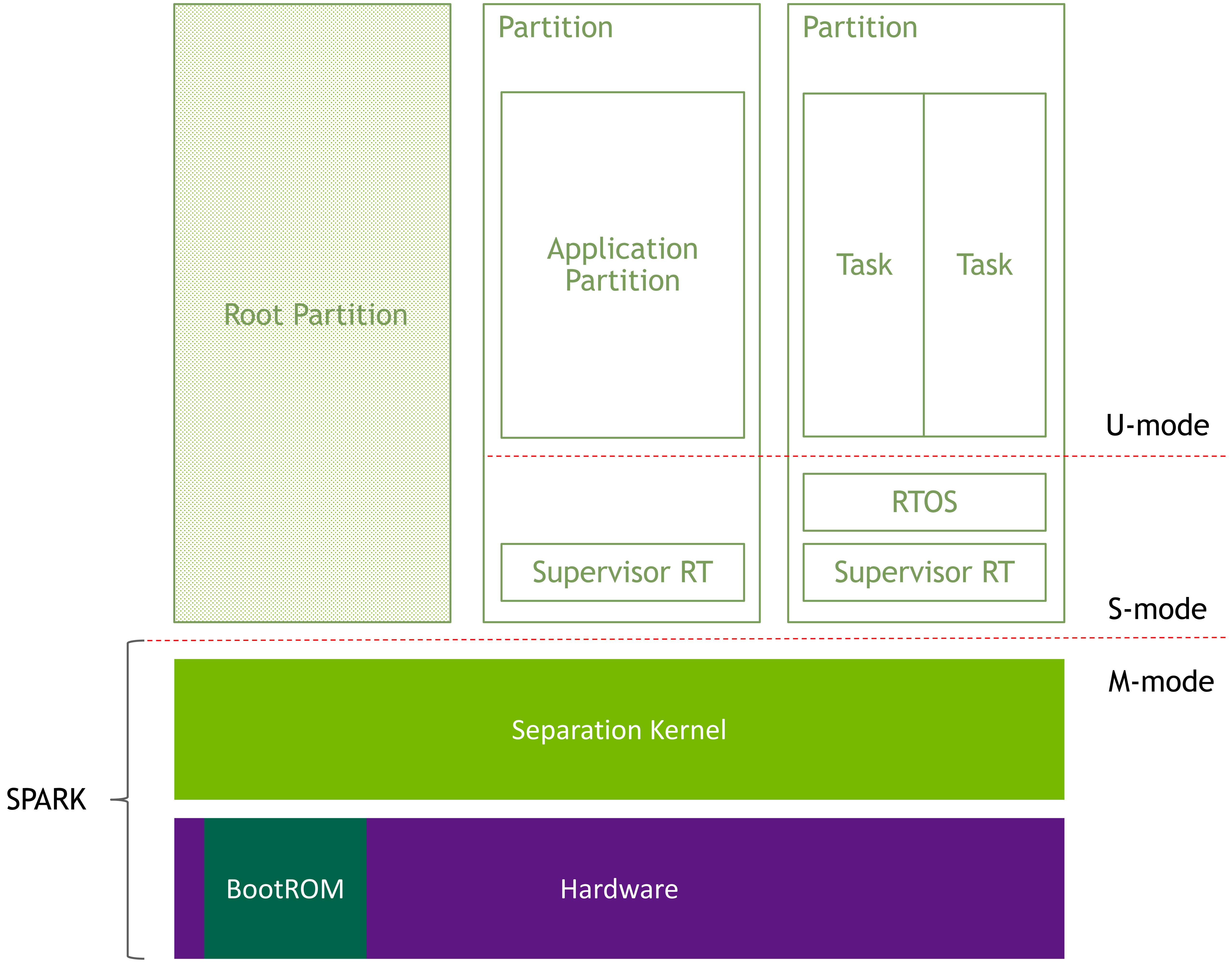


SECURE MONITOR ARCHITECTURE

Policies defined in software, enforced by hardware

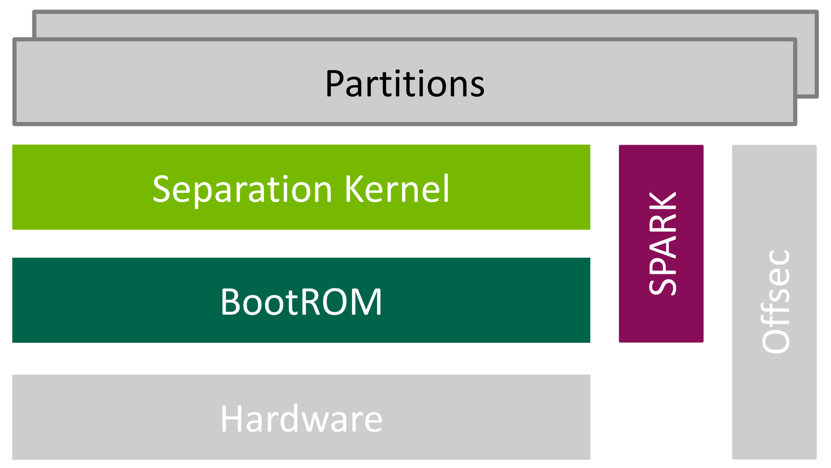


- Multiple Independent Levels of Security/Safety (MILS) architecture
  - Each component can be developed and analyzed separately
- Secure Boot starts with immutable BootROM in HW
- Separation Kernel creates and manages isolated execution environments (partitions)
- Applications run in isolated execution environments

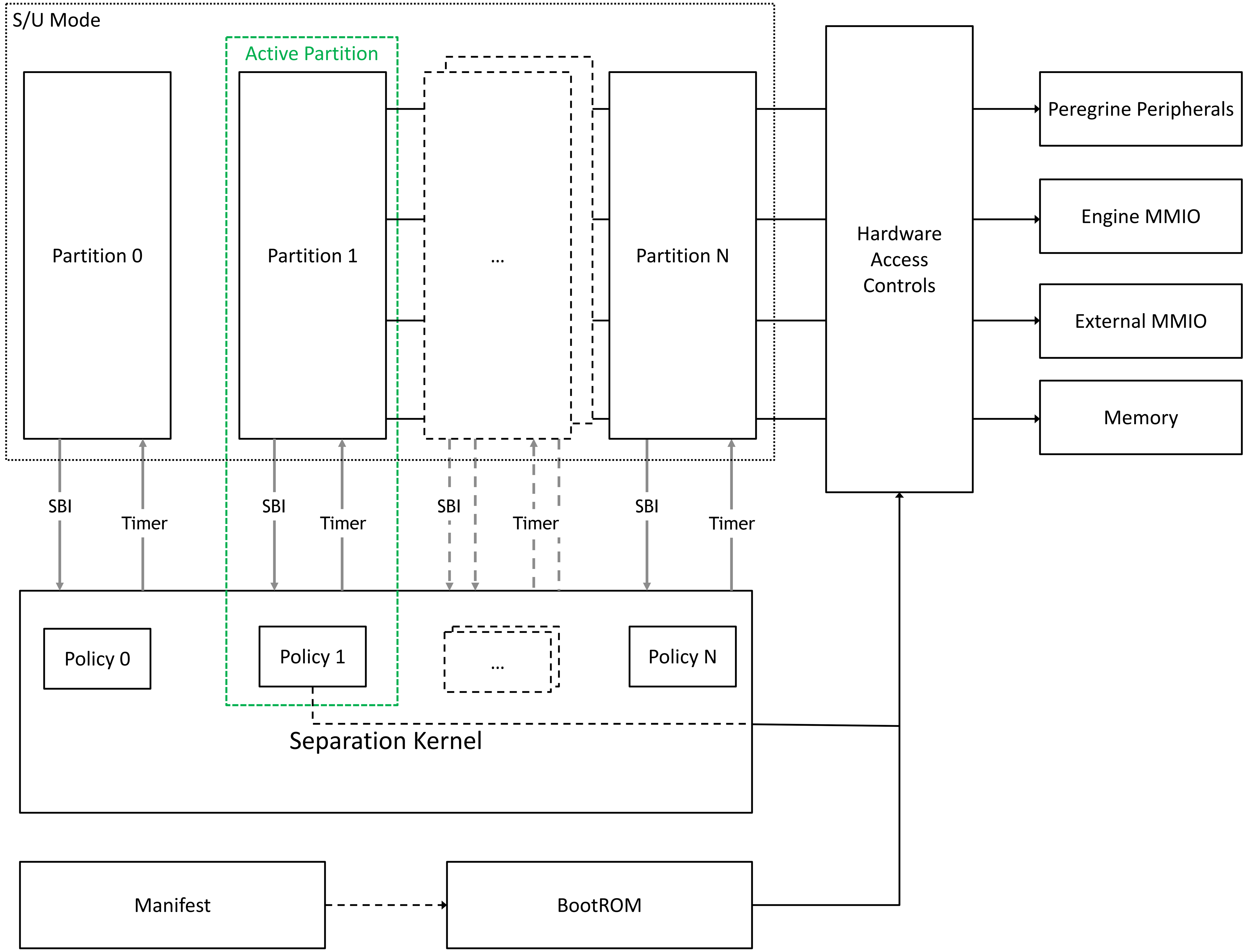


NVRISCV SEPARATION KERNEL

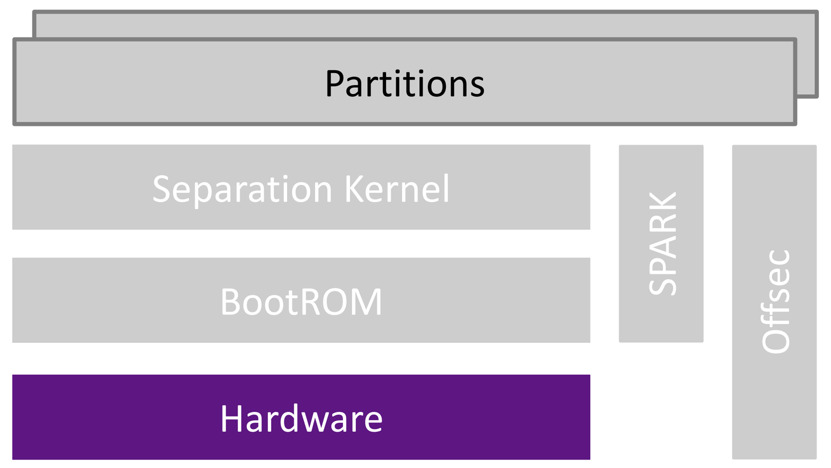
Foundation for running mixed-criticality applications on a single NVRISCV core



- Controls what HW is exposed to partition, does not provide interfaces to it
- Small and formally verified to be free of runtime errors
- Manifest is signed, static configuration set
- Maximum privilege set by BootROM, defined by the manifest
- Partition is defined by partition configurations – *partition policies*
- All information flow in/out partitions is access controlled
- Partitions are NVRISCV Trusted Execution Environments (TEE)



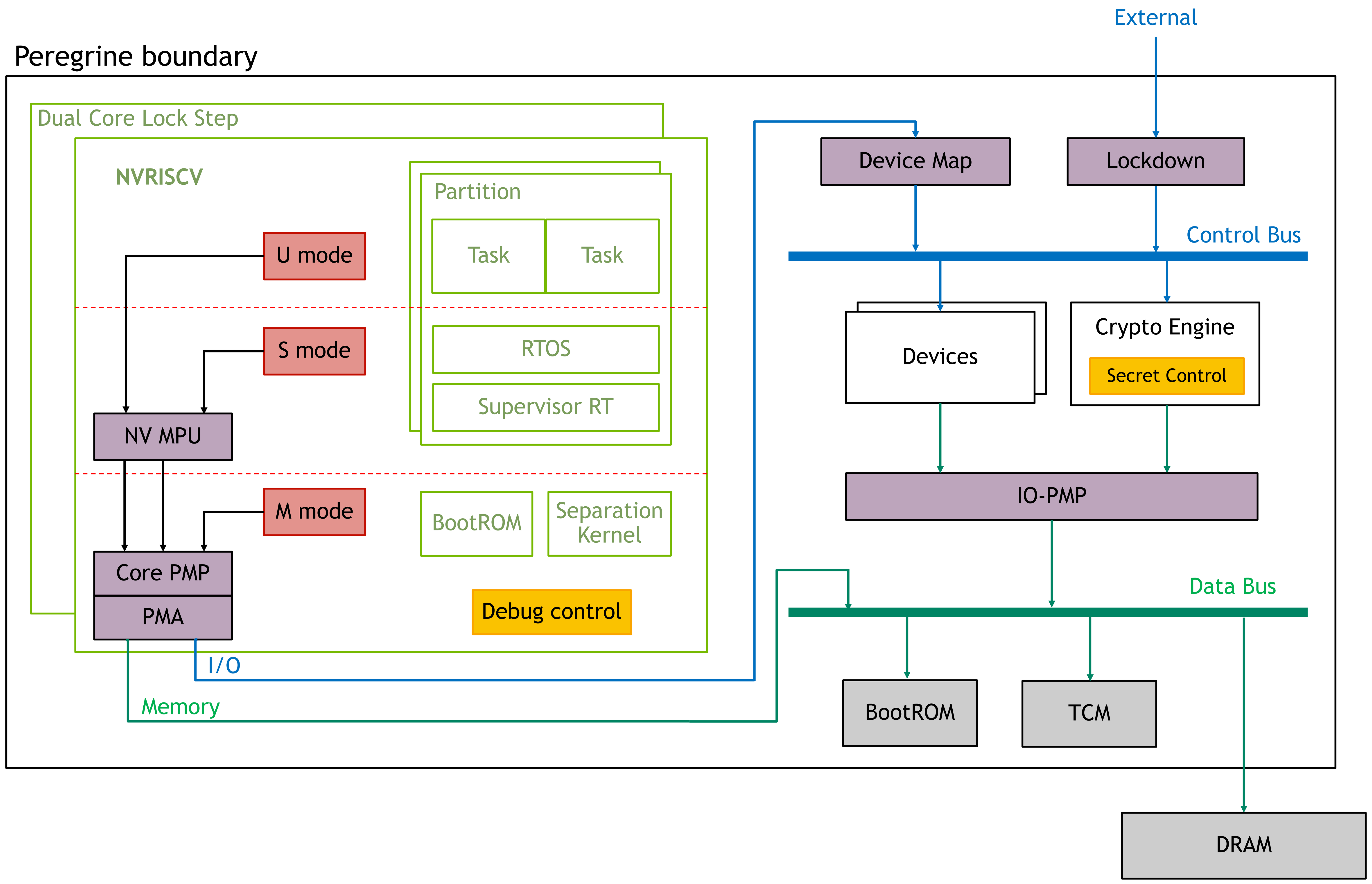
NVIDIA



LAYERED ACCESS CONTROLS

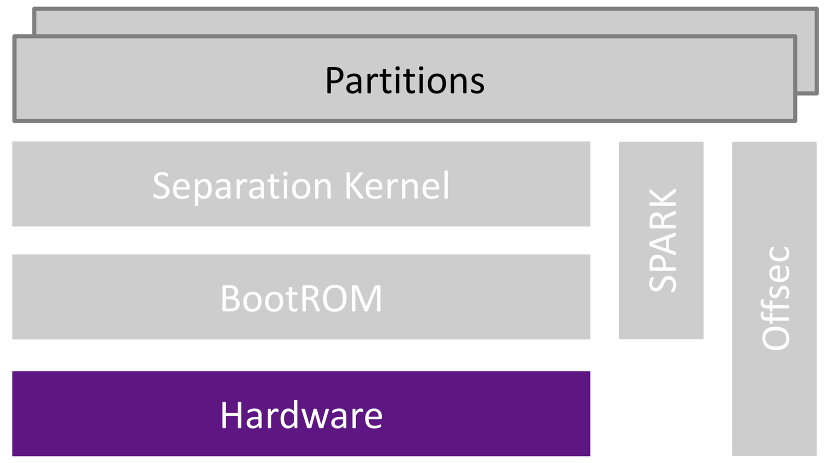
Overview

- Core privilege isolating on M/S/U
- Core access control
  - NV Memory Protection Unit, Core Physical Memory Protection, Physical Memory Attribute
- Fault-Injection countermeasure
  - Dual Core Lock Step
- Control bus access control
  - Device map, Lockdown
- Data bus access control
  - IO PMP
- Other access control
  - Debug/profiling controls, Secret HW Key control

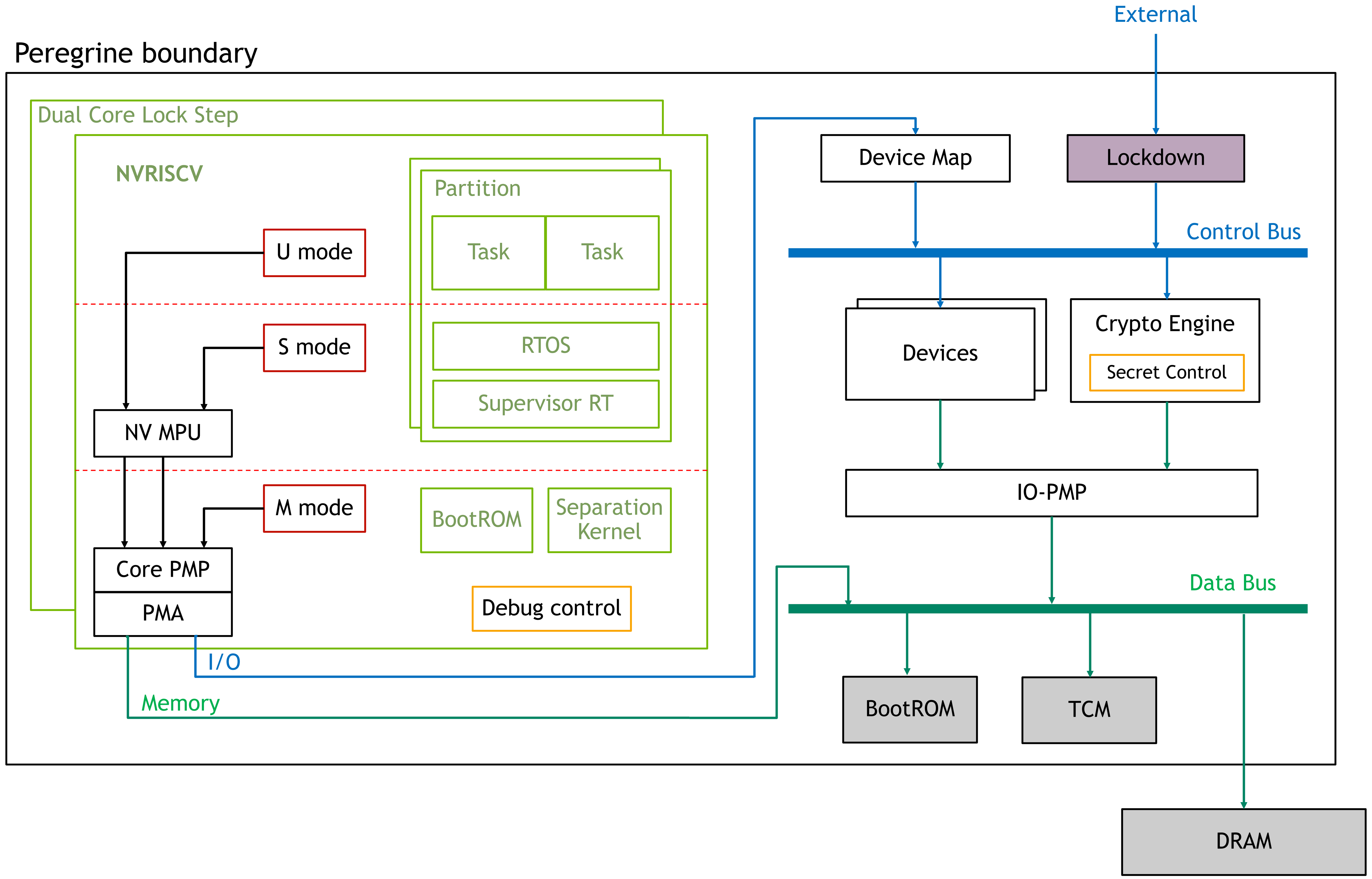


LAYERED ACCESS CONTROLS

Out of Reset

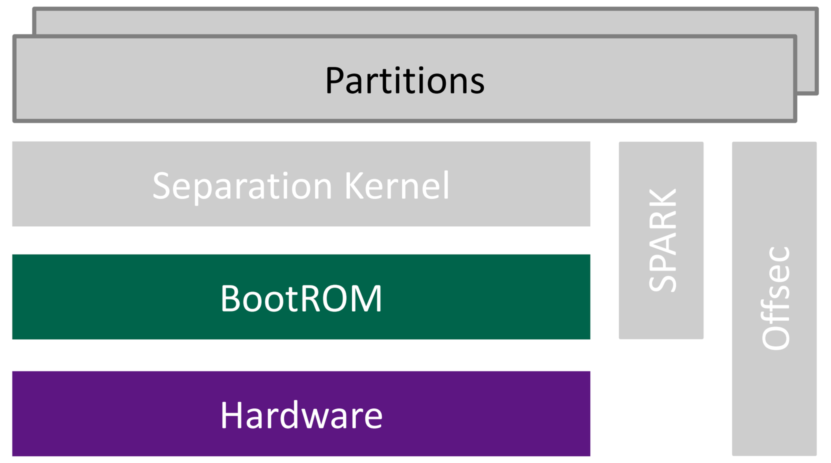


- External MMIO lockdown is enabled
- Peregrine opaque to the outside world

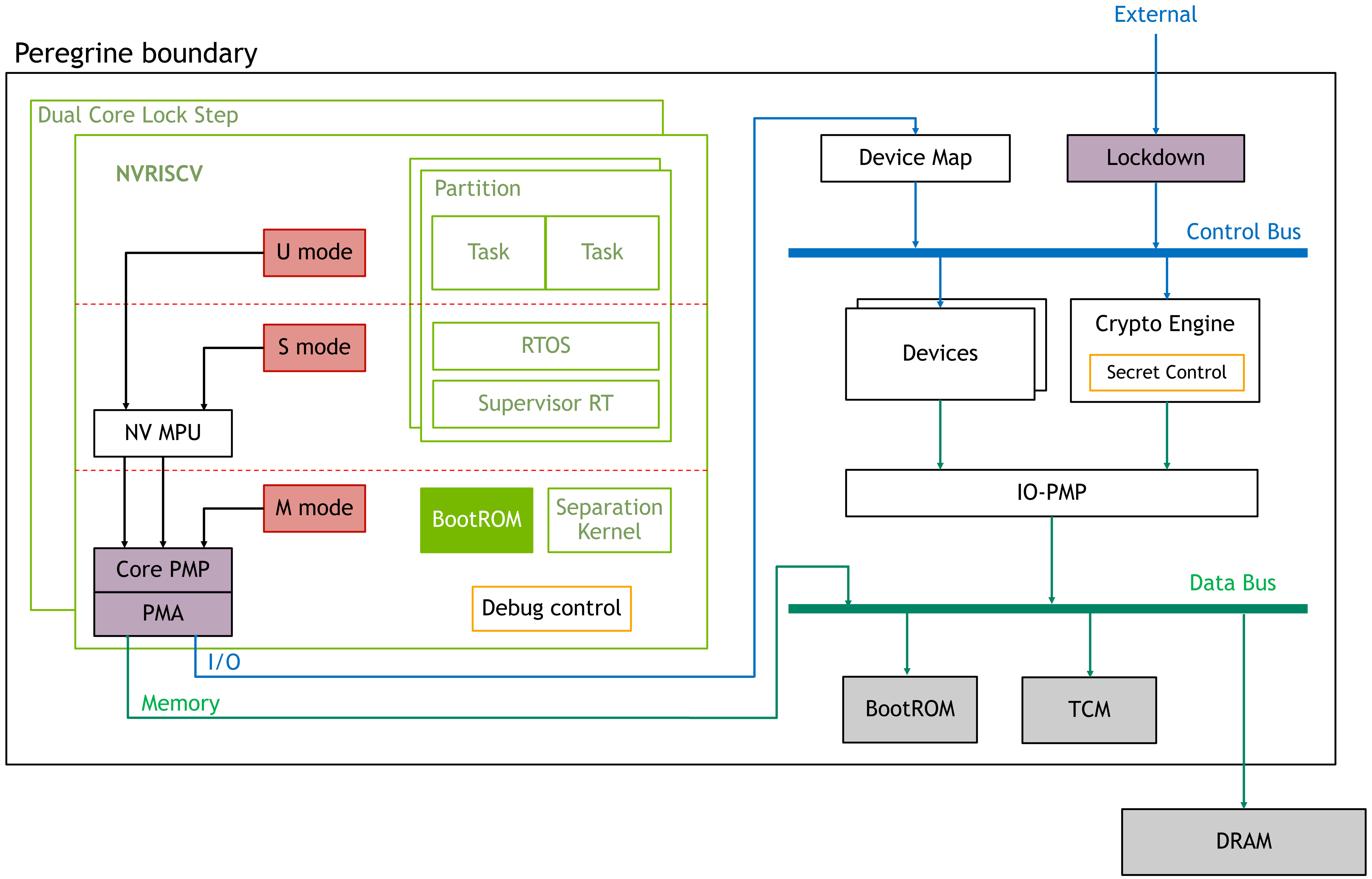


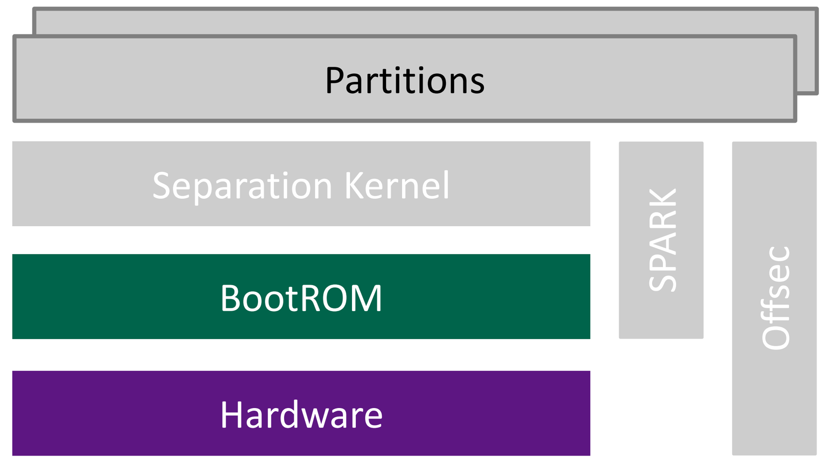
LAYERED ACCESS CONTROLS

1st instruction integrity



- PMA ensure BROM cannot be dumped
- Default Core-PMP ensure execute from BROM only
- DCLS enabled in default

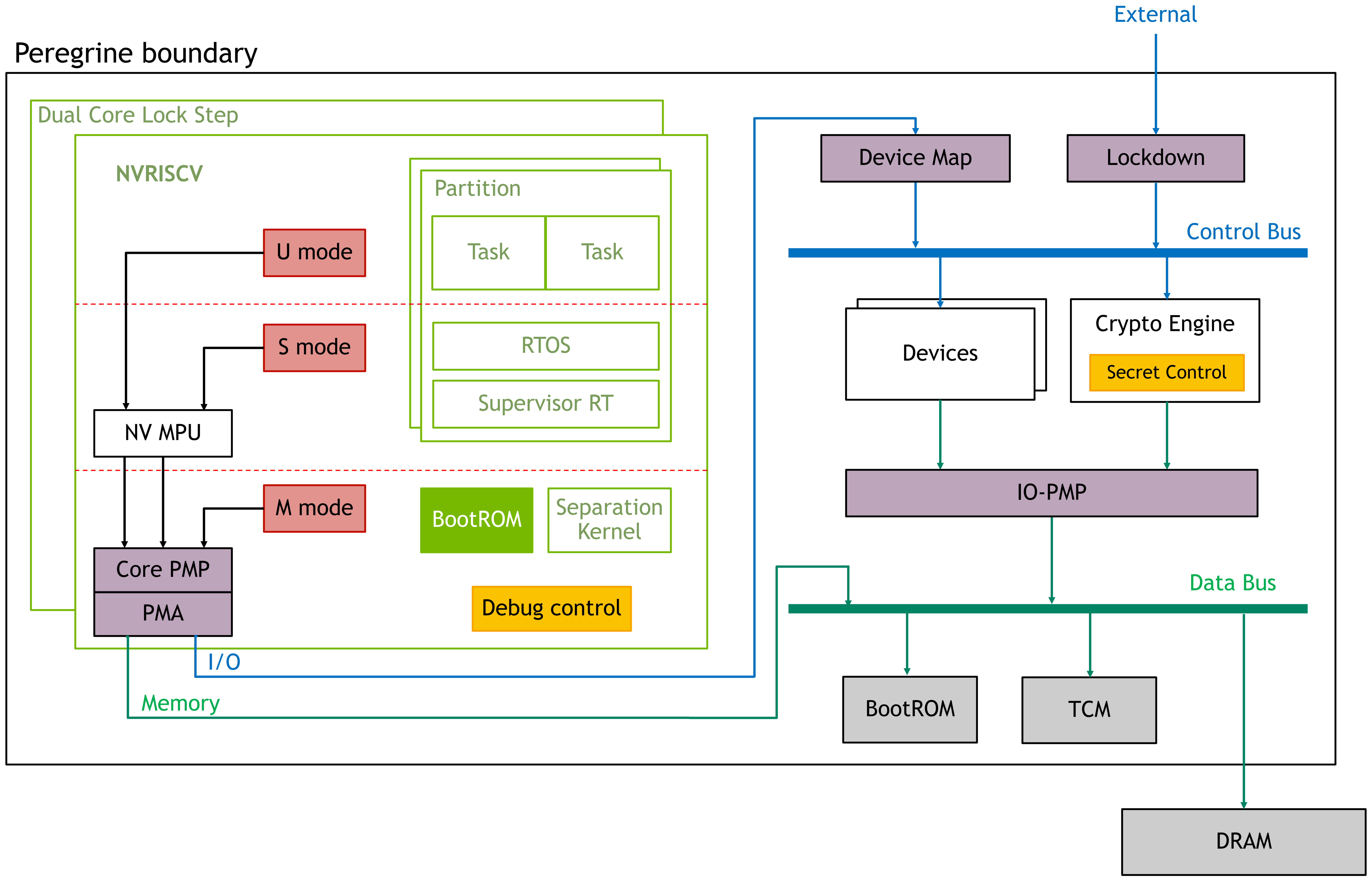


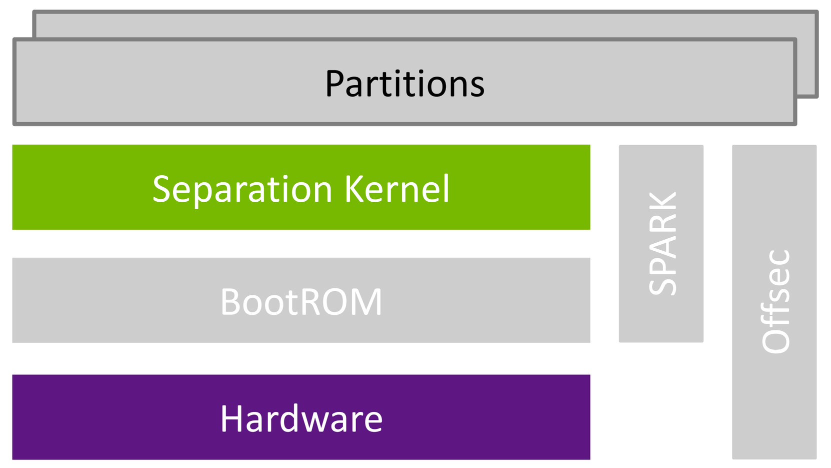


LAYERED ACCESS CONTROLS

BootROM runs

- Fetch manifest/FMC
- Verifies signature, decrypts FMC
- Sets up execution environment per manifest
- Cleans up after itself
- BootROM can never be re-entered

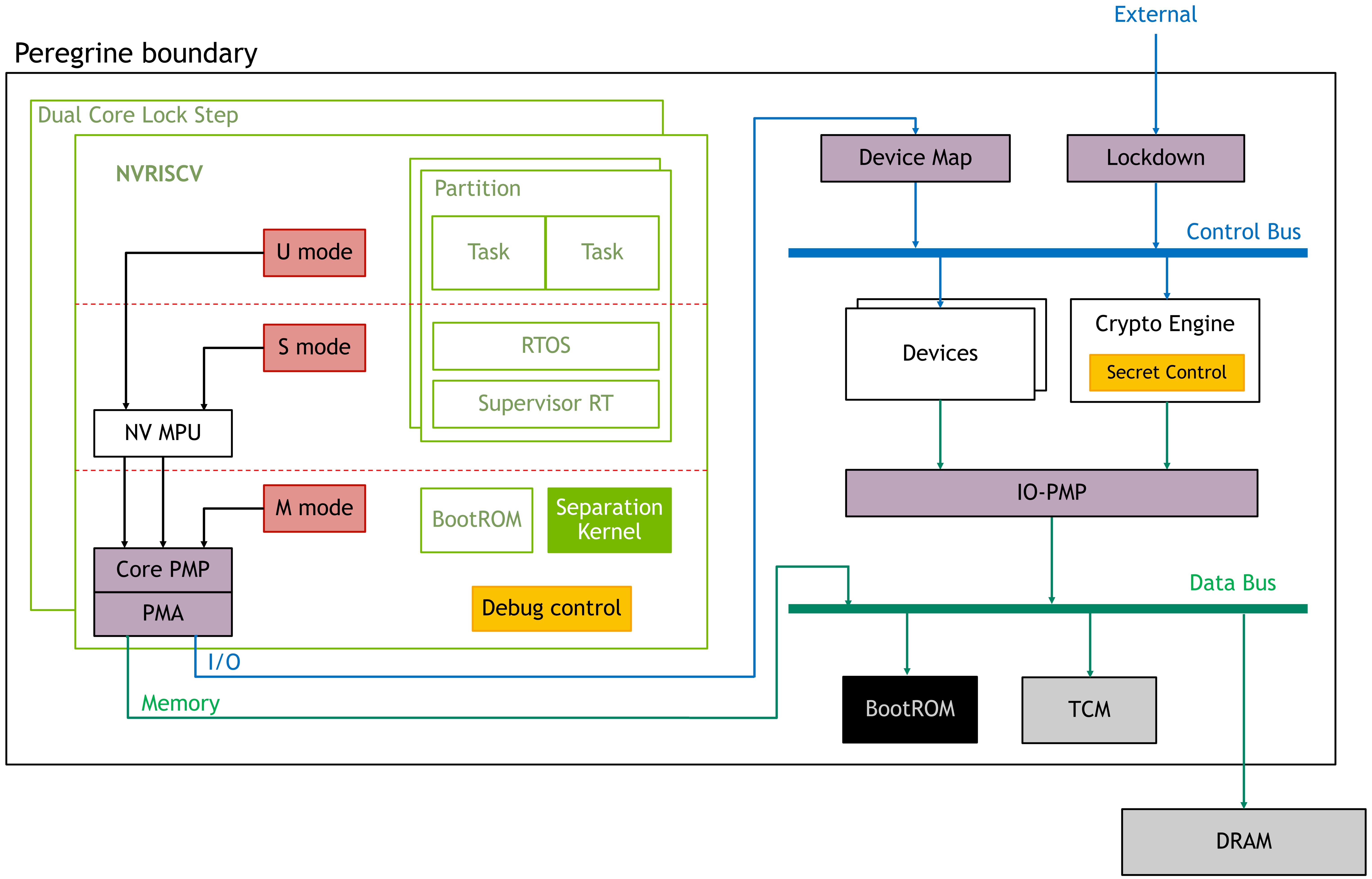


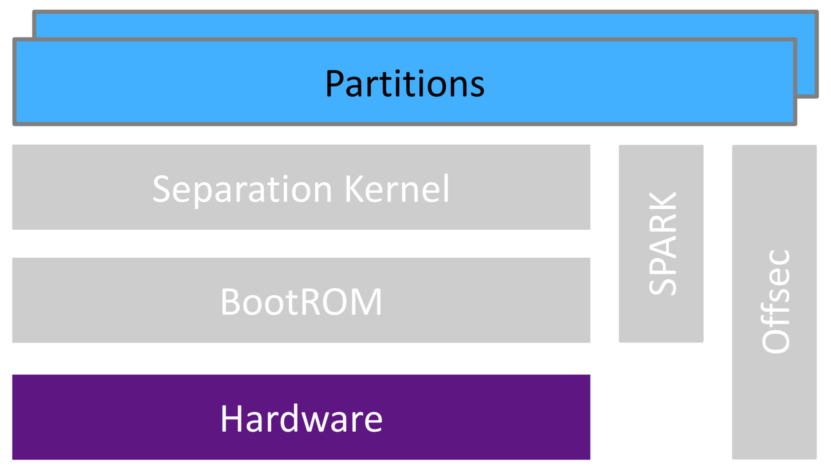


LAYERED ACCESS CONTROLS

Separation Kernel runs

- Verifies/Configures HW to protect itself
- Controls what HW is exposed to partitions per partition policies via:
  - Core-PMP
  - Device map
  - IO-PMP
  - Debug/profiling control
  - Secret control

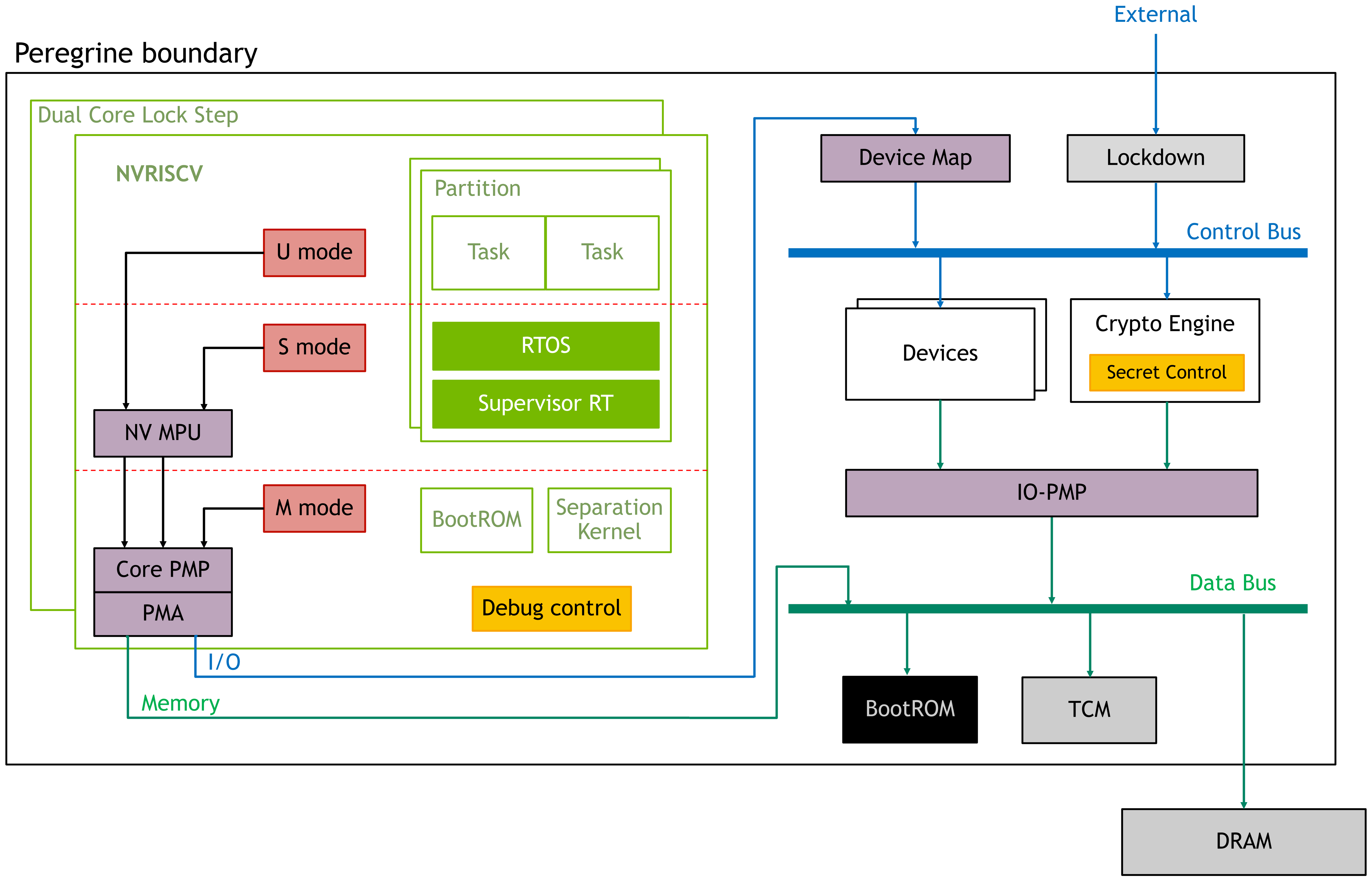




LAYERED ACCESS CONTROLS

RTOS runs

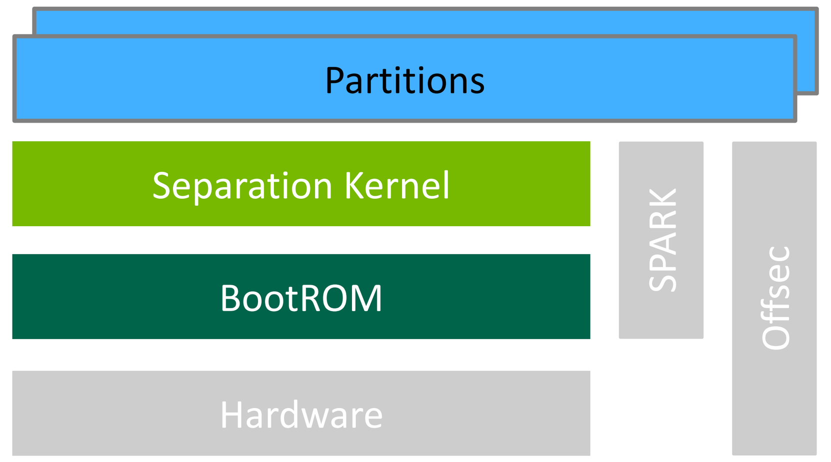
- RTOS can further isolate U-mode tasks through NV-MPU



FLEXIBLE ARCHITECTURE

Limitless partition configurations

NVIDIA



SOFTWARE SECURITY CHALLENGES


GAMING


HPC

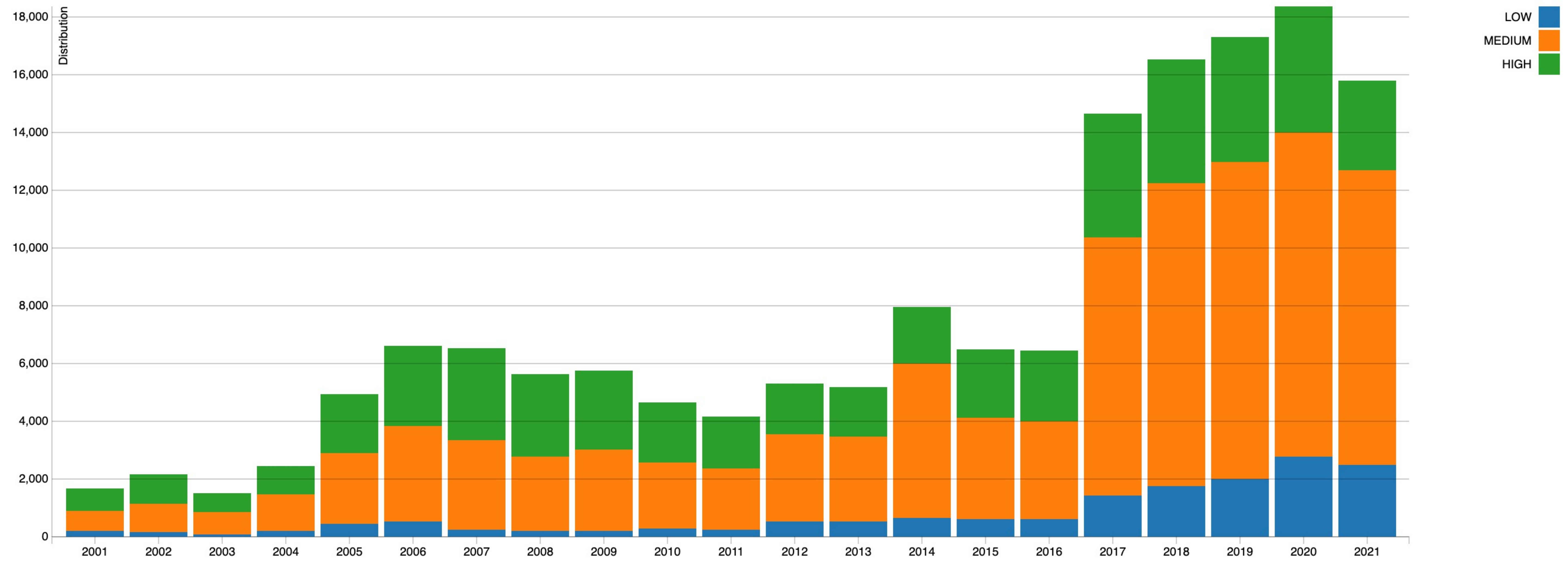

HEALTHCARE


SMART CITY

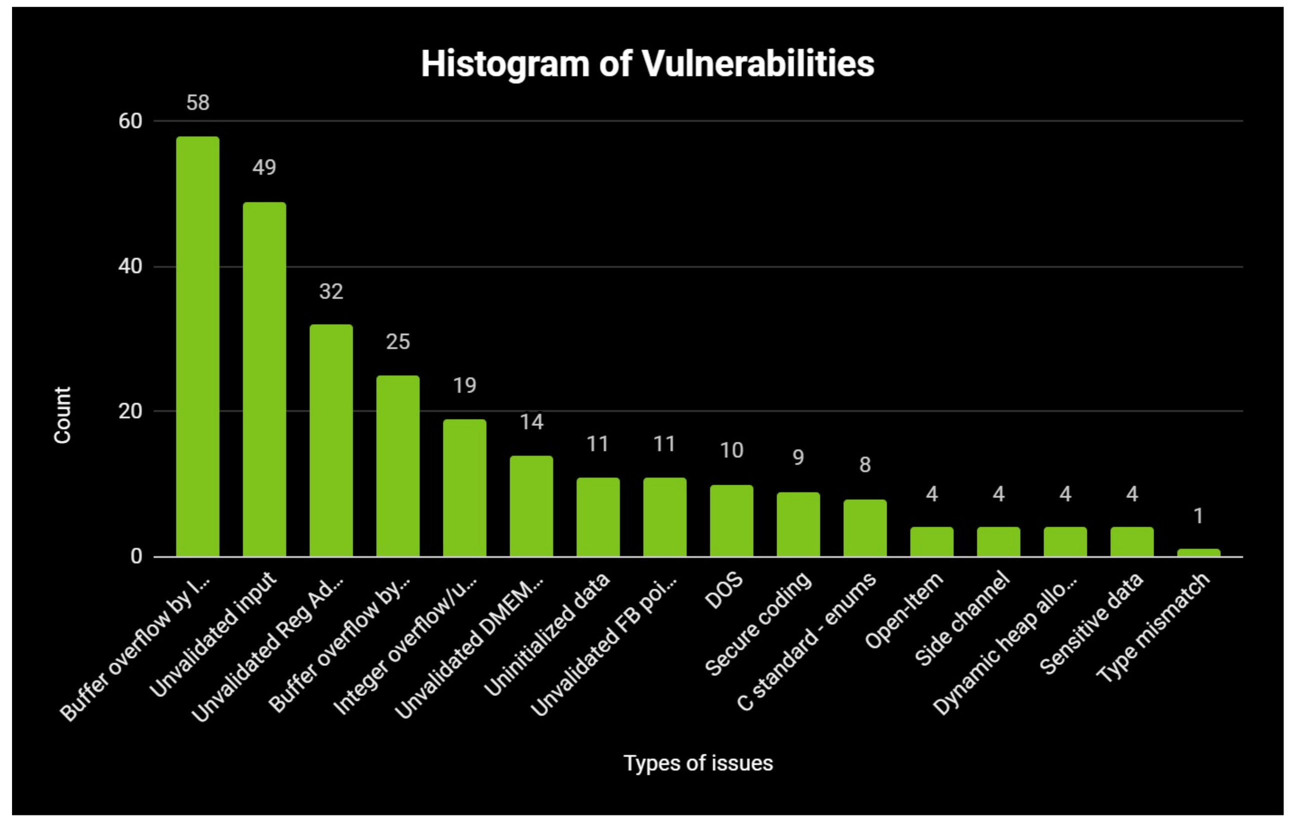

ROBOTICS

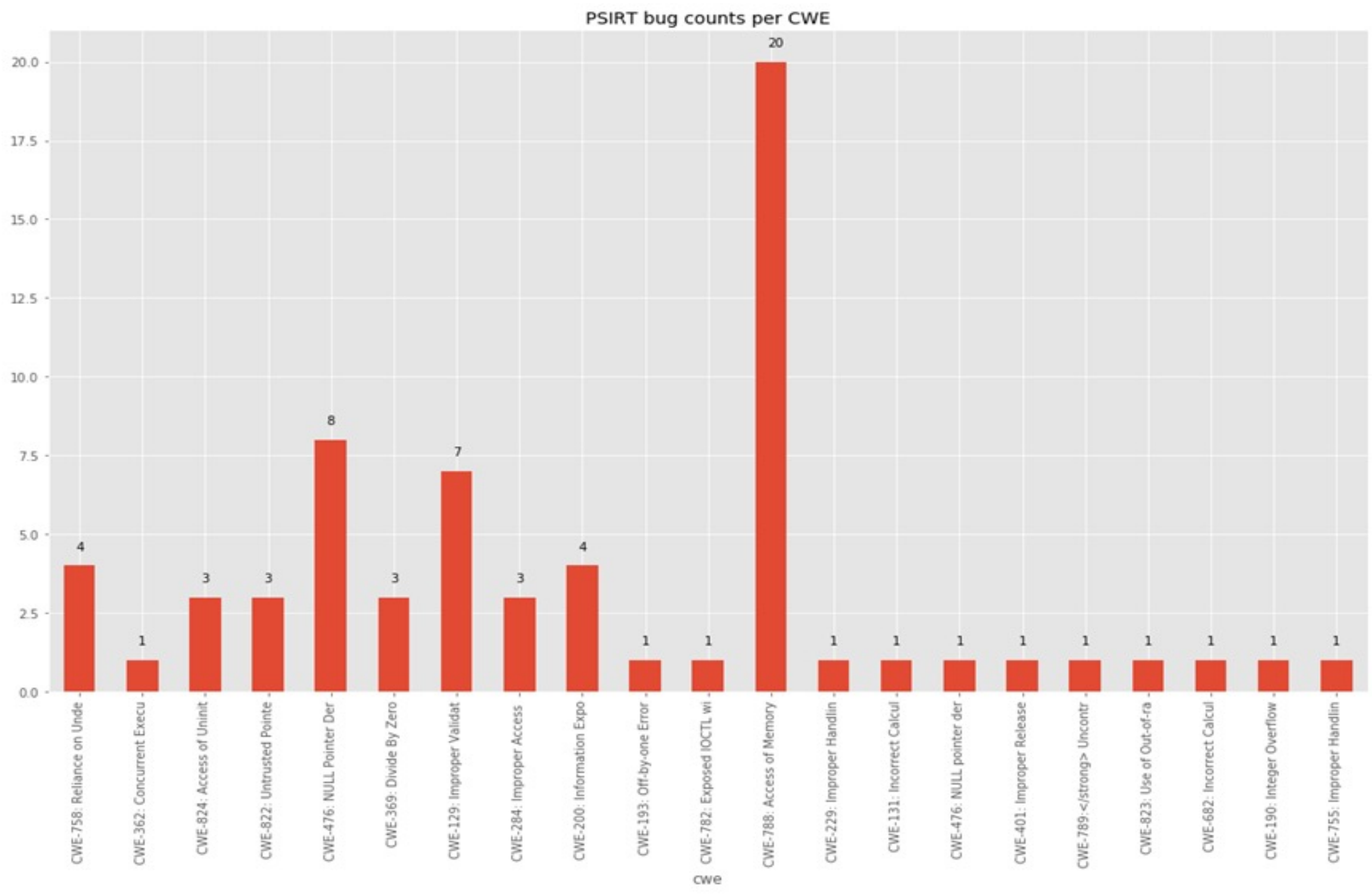

AUTOMOTIVE

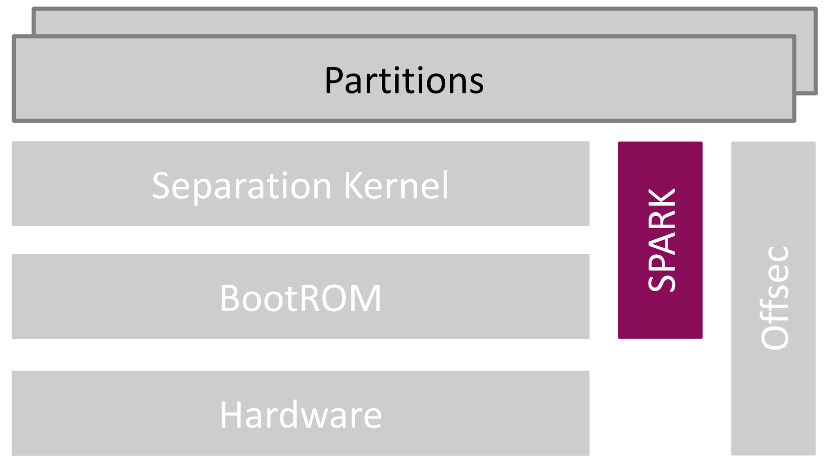
- Increasing criticality (IP risks, safety)
- Software complexity is increasing
- Limited prevention techniques
  - Penetration Testing
  - Fuzz testing
  - Static Analysis
  - Internal assessments
  - 3rd party audits
- Mitigation technique challenges
- Human factor
  - Scalability
- Automation
  - Limited tooling effectiveness
- “Don’t automate, obliterate”
  - Goal: Eliminate whole classes of vulnerabilities



NVD CVSS Severity Distribution Over Time



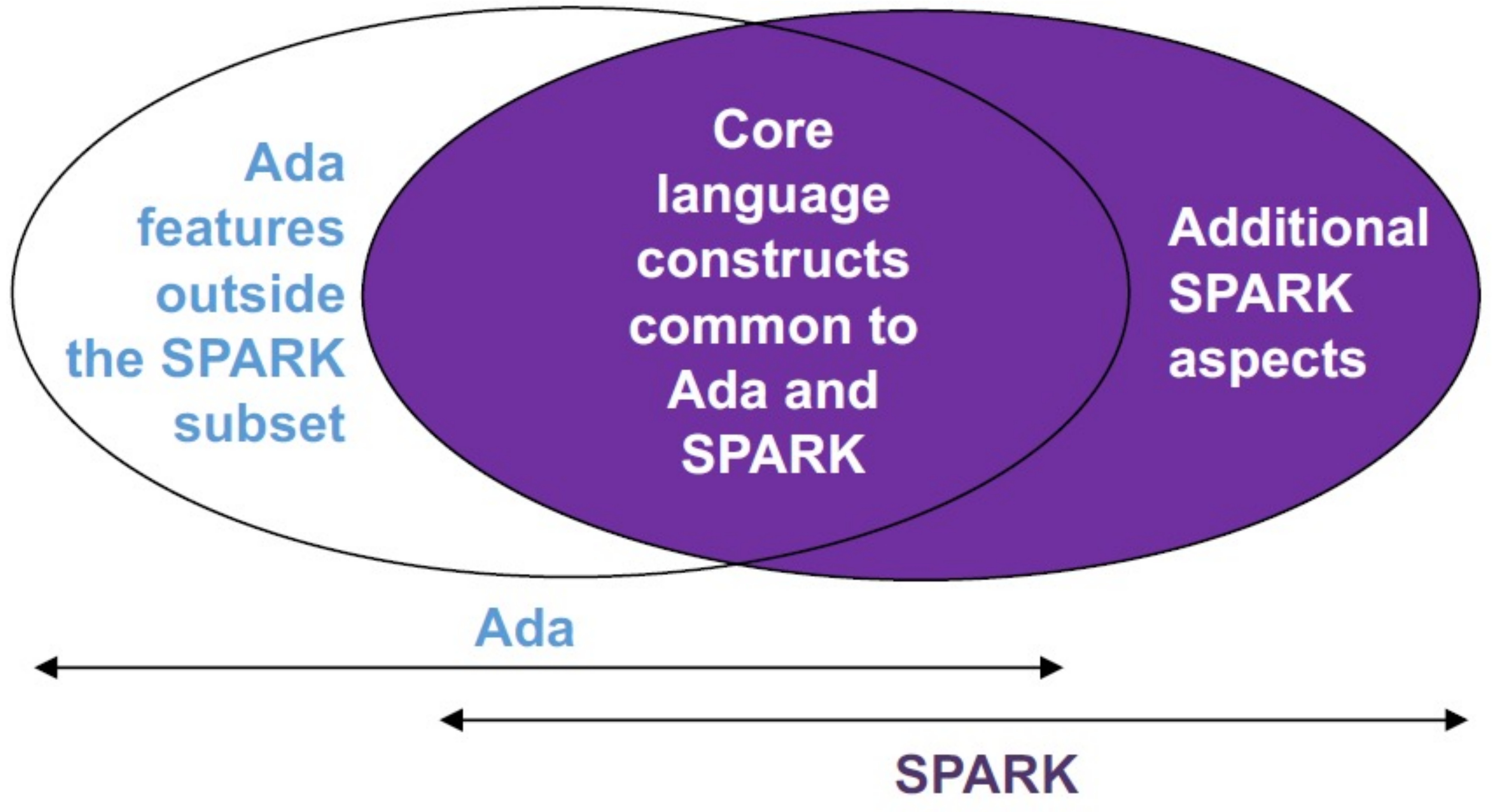


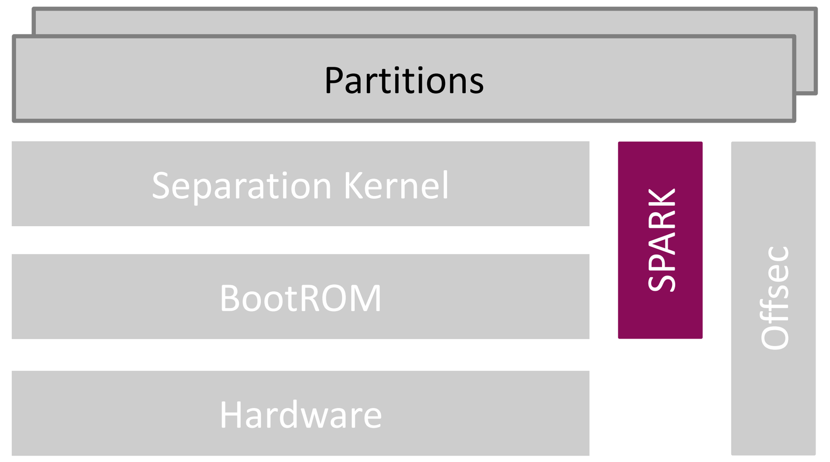


WHAT IS SPARK?

Programming language + set of analysis tools

- Statically provable
  - Proves that dynamic checks cannot fail
- Very strong typing system
- Supports programming by contracts, Ghost code
- RISCV64, GNATProve, GNATStack, GNATTest, GNATEmulator
  - RISCV compiler is ISO26262 certified
- Traditionally used in industries where code reliability is critical, such as
  - Avionics, Railways, Defense, Auto, IoT
- Most suitable for safety/security critical applications

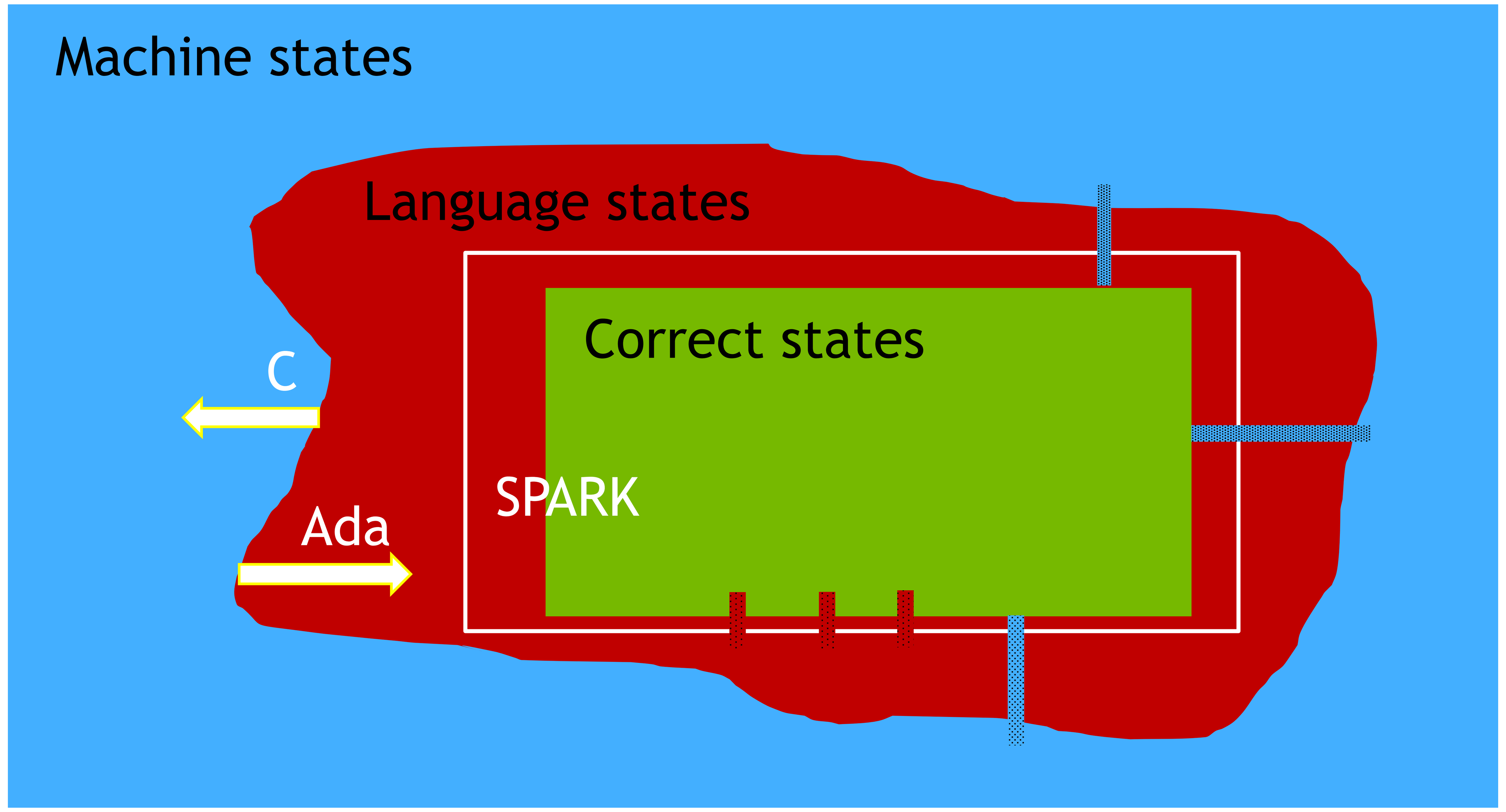




WHY SPARK?

AoRTE – Absence of Run-Time Errors

- Mature language
- Broad ecosystem and commercial support
  - AdaCore
- Upstream OSS presence
- ‘Proofs’ over ‘tests’
- Ability to **prove Absence of Run-Time Errors**



Tested Procedure

Proven Procedure

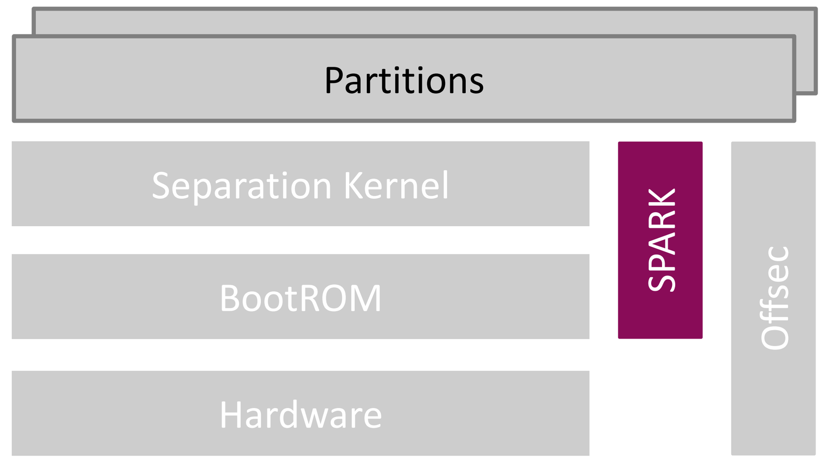
Preconditions
are tested
Postconditions
are proven

Proven Procedure

Tested Procedure

Preconditions
are proven
Postconditions
are tested

*[*What is Safety-Critical Software, and How Can Ada and SPARK Help?](#)*



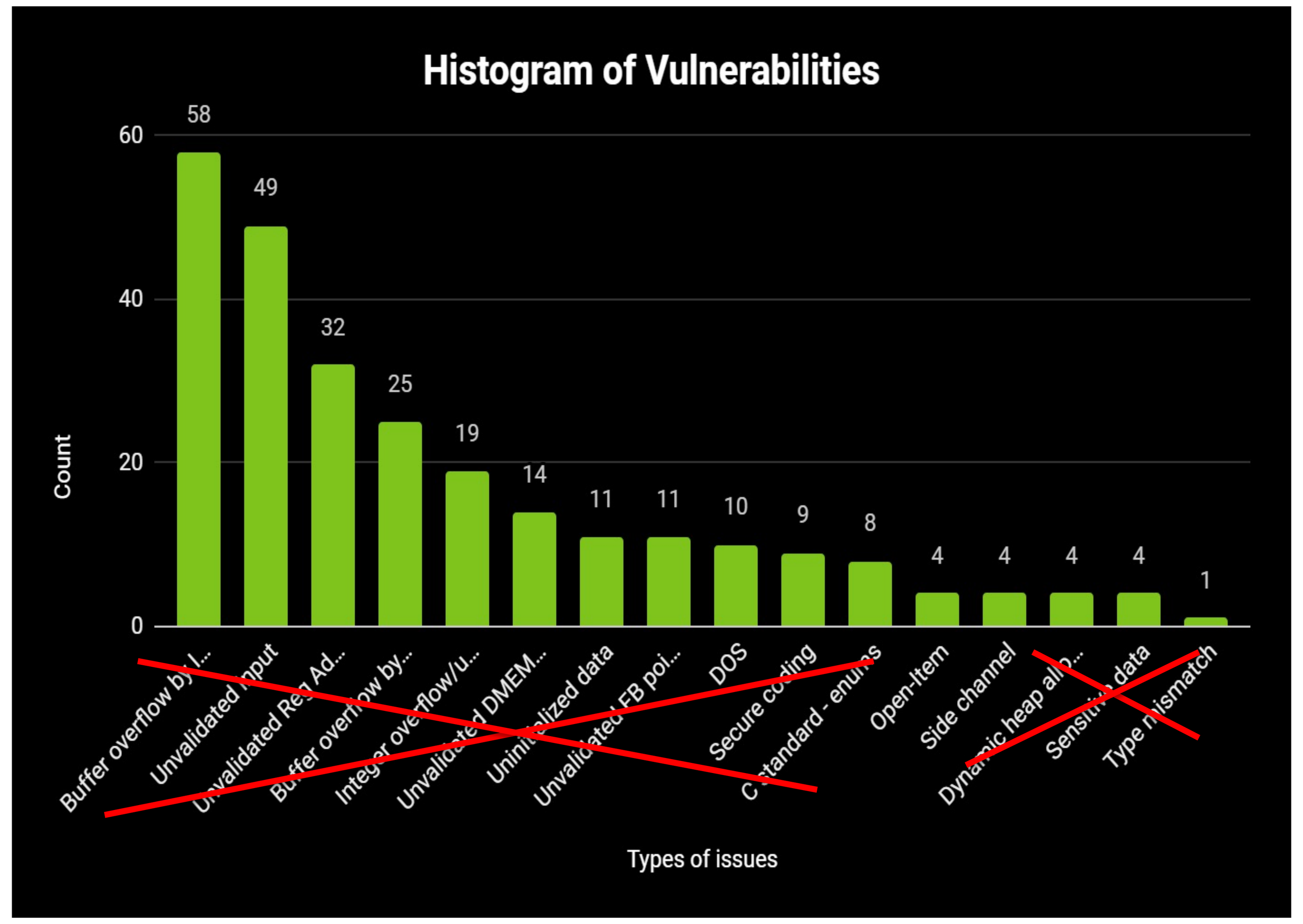
SPARK BENEFITS

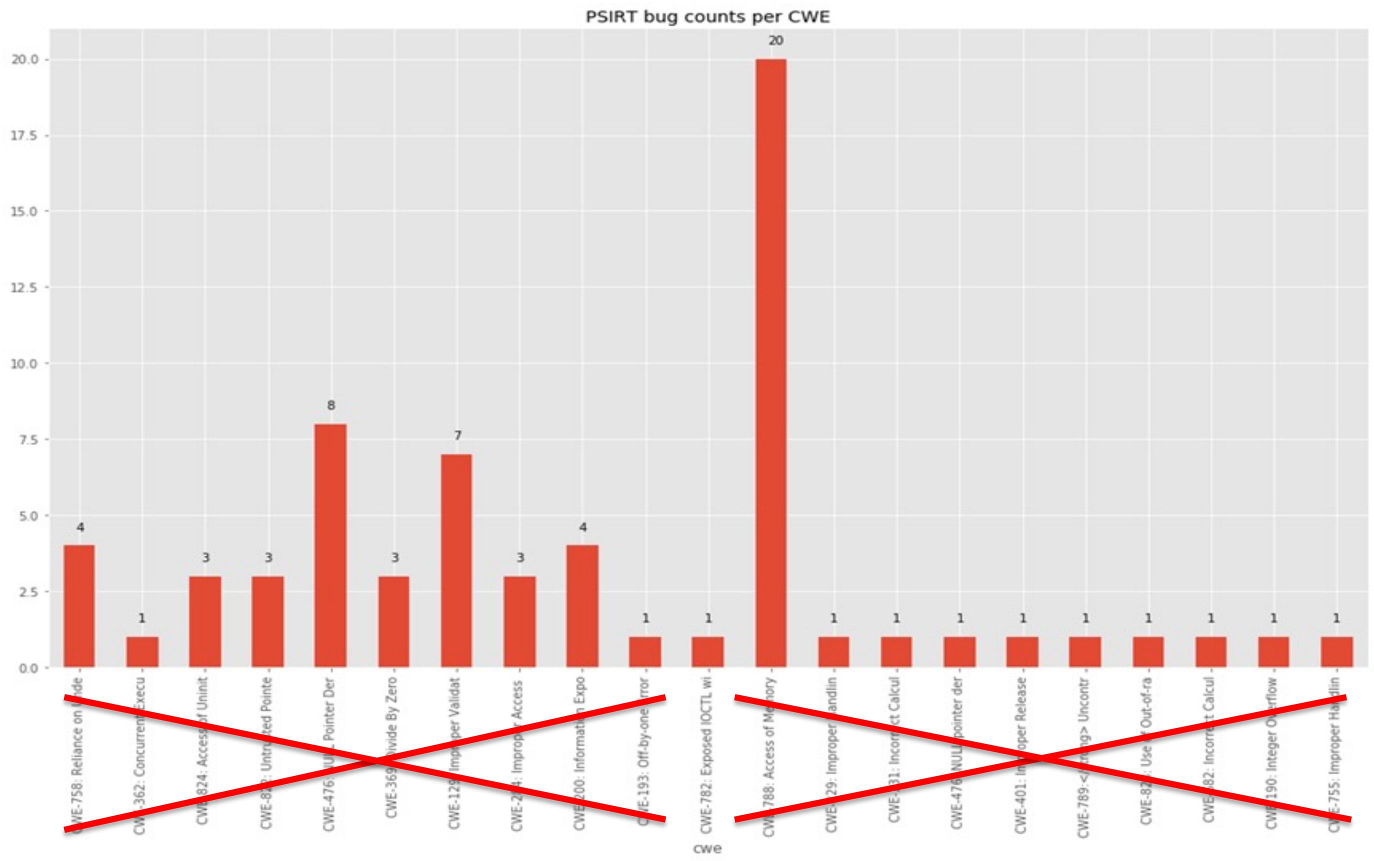
Security, reliability and cost

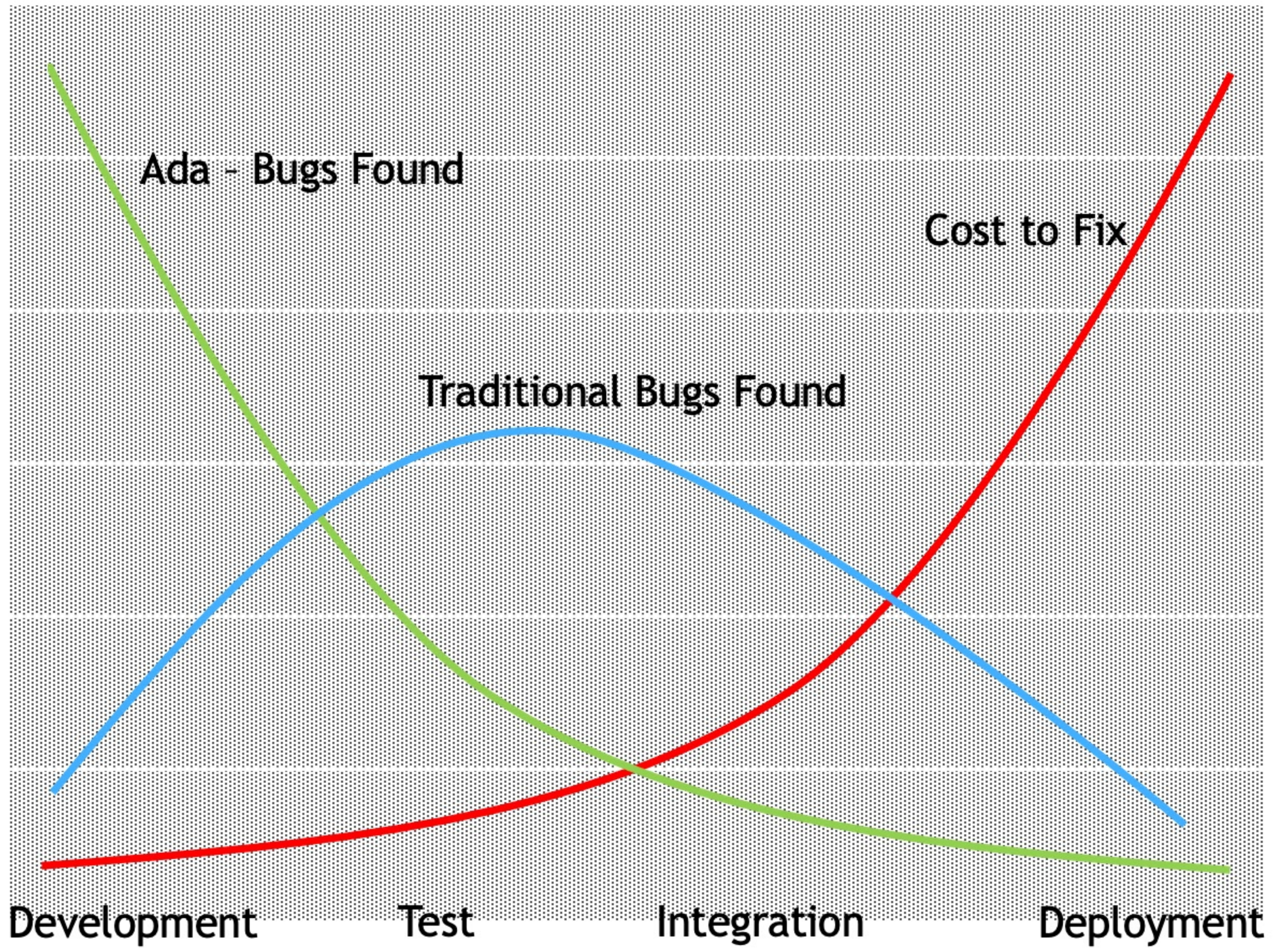
- AoRTE (Absence of Run-Time Errors)
- CWE prevention
- No MISRA-C, Cert-C overhead

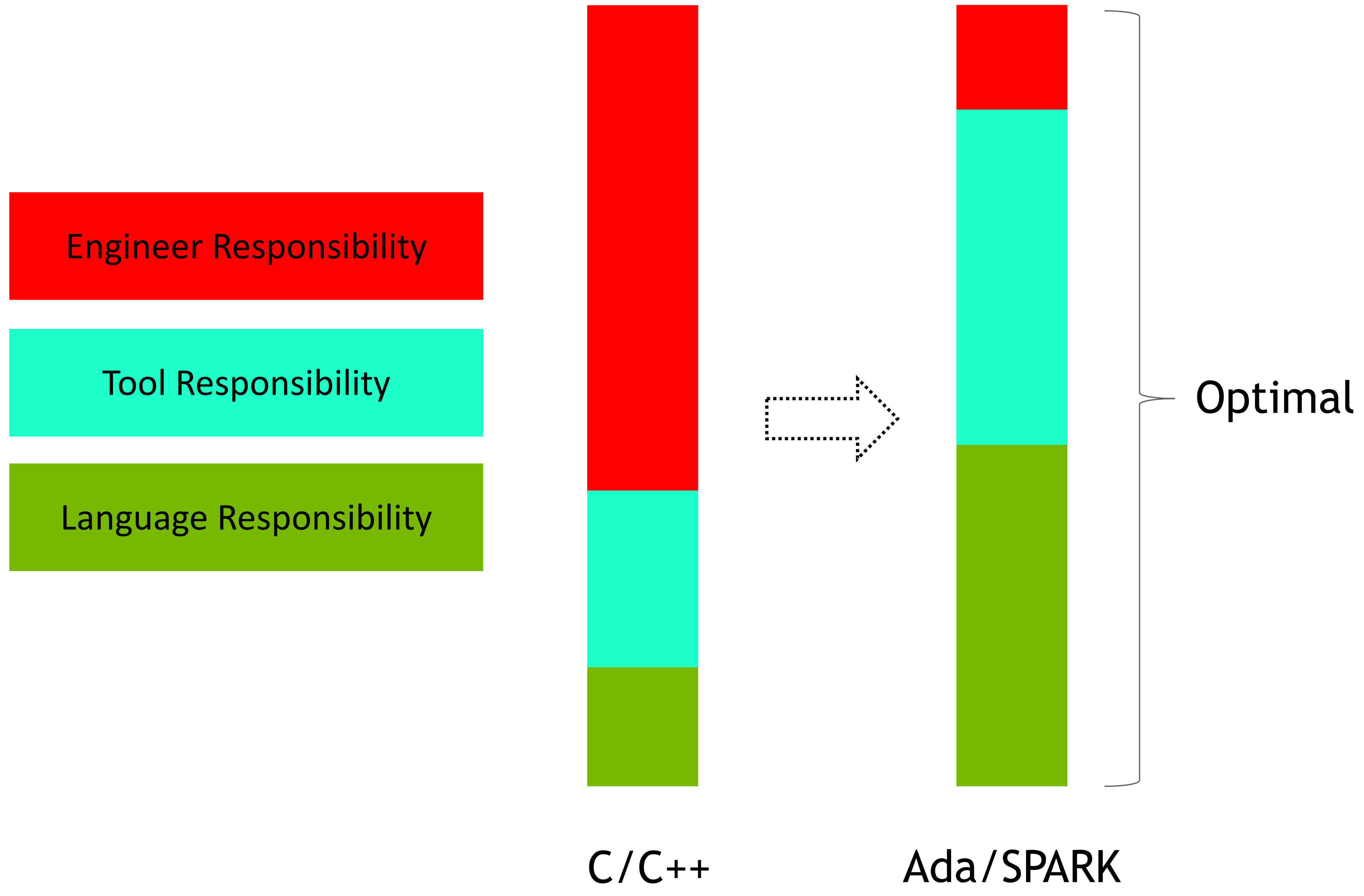
- Strong Typing
- Sound Static Verification
- Enforce Secure Practices

- Accurate worst-case stack usage prediction (GNATStack)
- Enables more efficient offensive security efforts





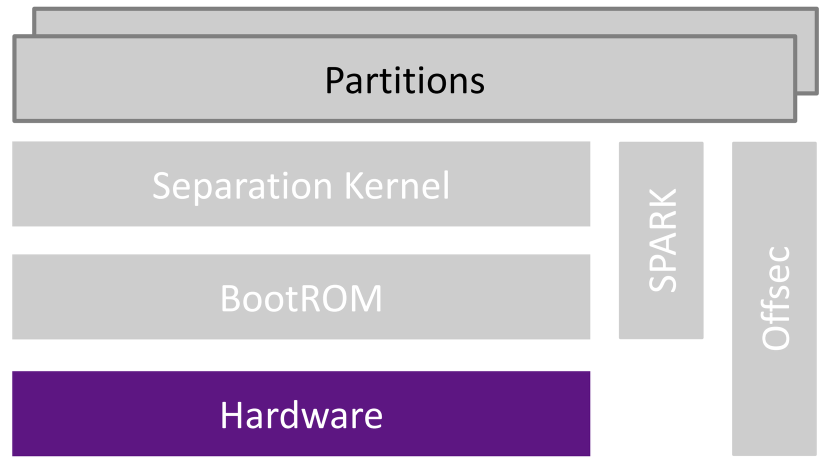




NVIDIA

OFFENSIVE SECURITY RESEARCH

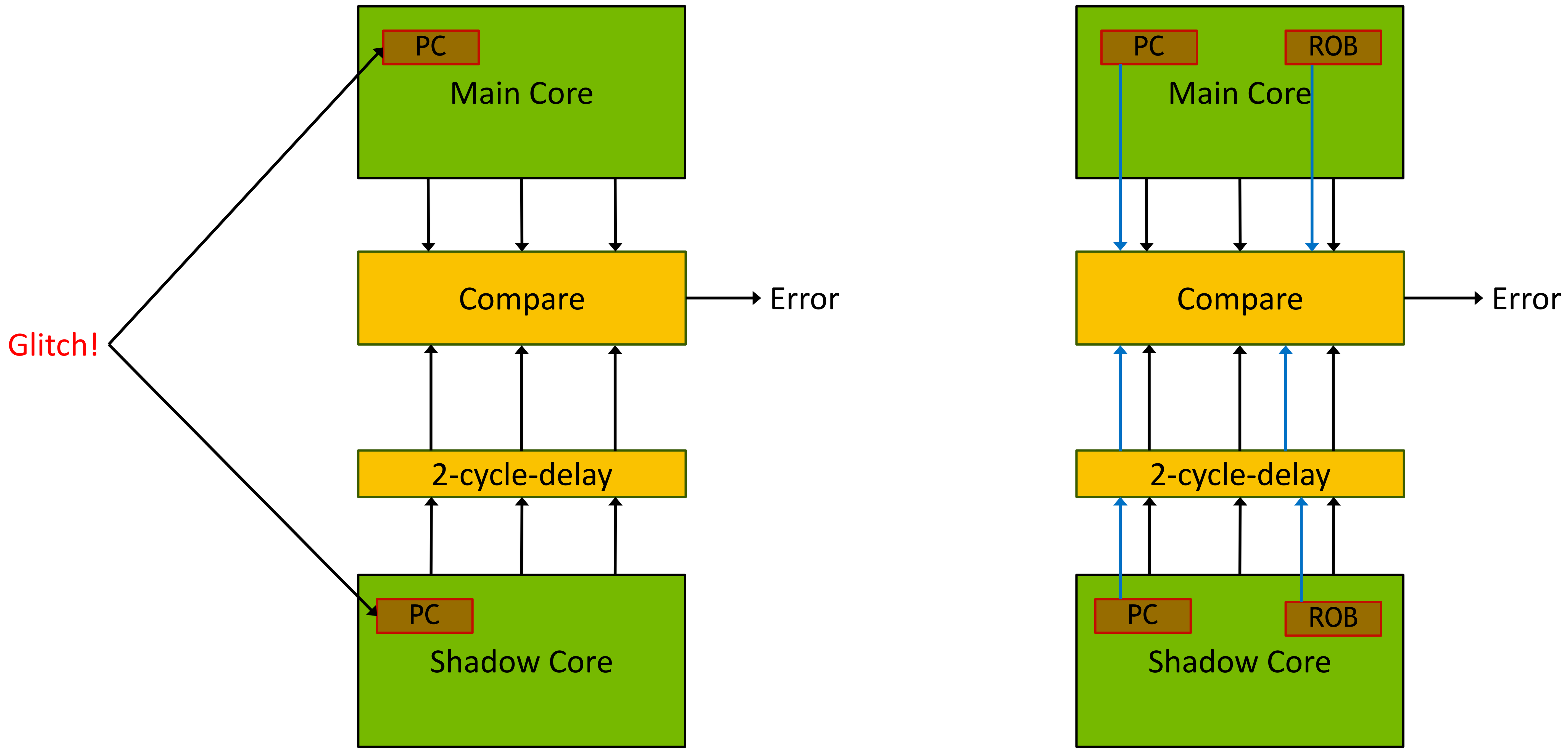
- Offensive security activities against our own products
- Internal Peregrine offensive security architectural evaluation
- Internal SPARK Evaluation
- DEF CON 29: "*Glitching RISC-V chips: MTVEC corruption for hardening ISA*"
  - *RISC-V MTVEC register specifications **does not define** the initial value (mostly 0)*
  - *In many implementations 0 is not a valid address (or not mapped) and any reference to it generates an exception*
  - *If there is any trap/exception generated before initialization of MTVEC register, RISC-V ends up in a very "stable" **infinite exception loop***
  - *Such state is ideal for a glitching attack - RISC-V is running at the highest privilege mode and constantly dereferencing glitchable register.*
  - **CVE-2021-1104**



NVRISCV/PEREGRINE SECURITY

2 cycle delayed Dual Core Lock Step

- ASIL D (*Automotive Safety Integrity Level D)* requirement
- Compares internal signals (PC, ROB, Debug signal, CSRs)
- The core halts itself on DCLS error
- “halt” instruction stop the core until next reset



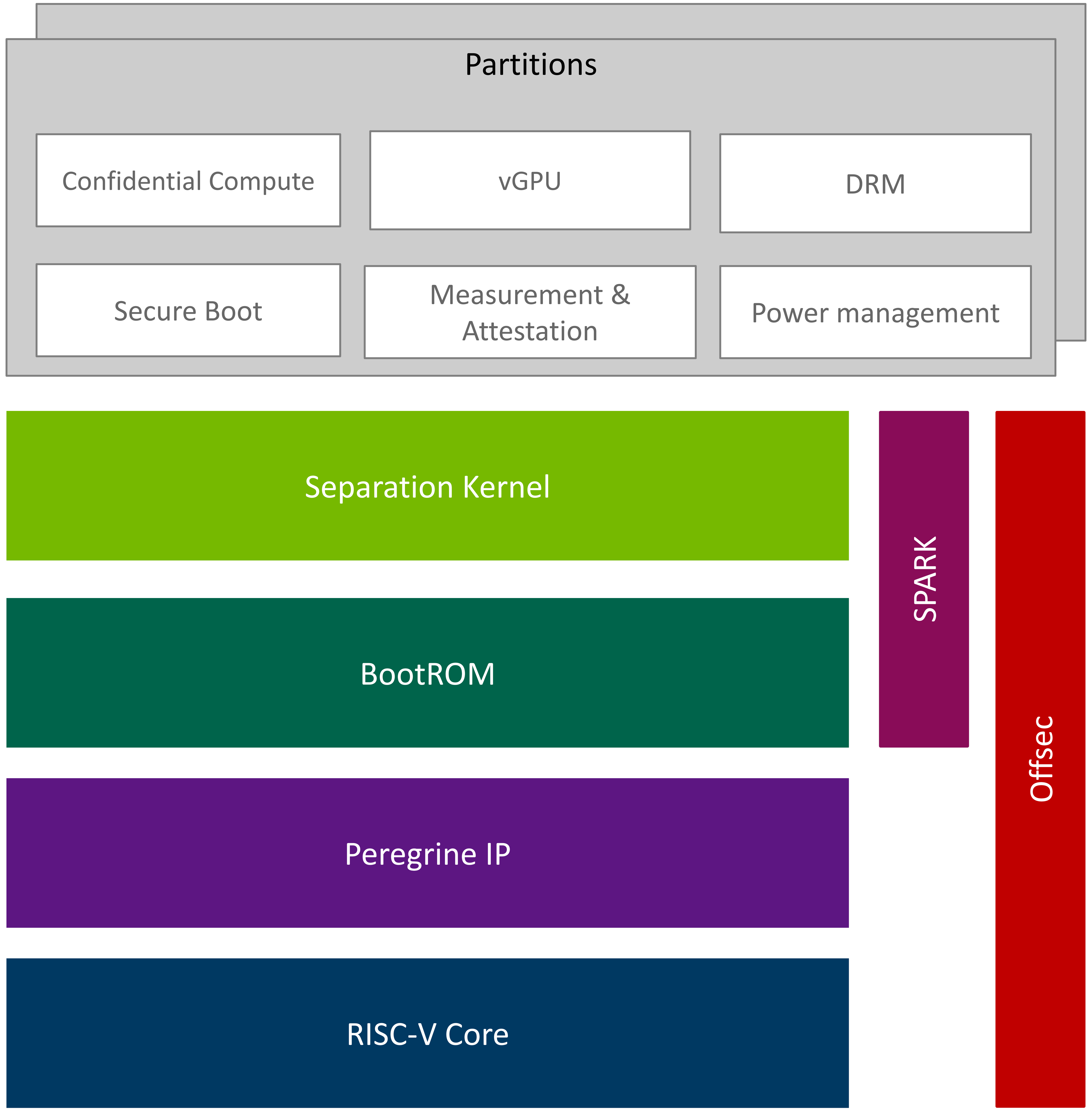
NVIDIA

SUMMARY

- TBI/Pointer Masking
- Static Analysis
- Fuzz testing

- Formal Verification
- Compiler mitigations
- SW mitigations

- Formal Verification
- Fault Injection Protections



BENEFITS

- Silicon area savings
- Engineering savings
- Security cost savings

THANK YOU

NVIDIA®