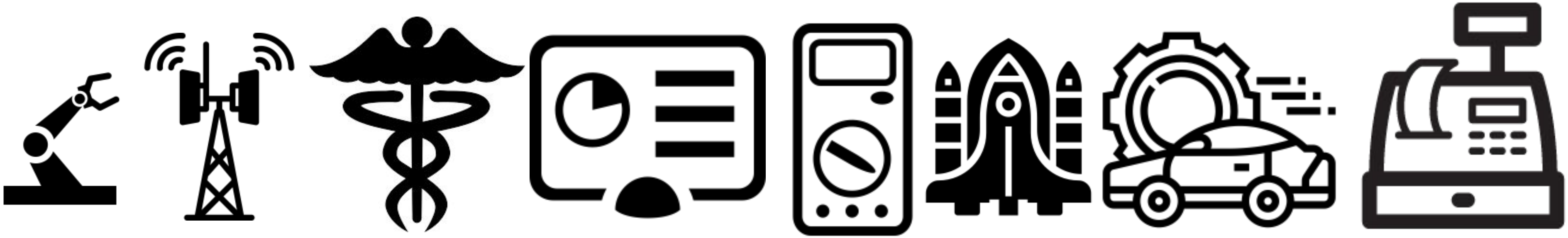
Architectural Foundations for Embedded Systems

Ved Shanbhogue, Rivos Inc.

Use Cases



- Wide array of embedded usages - industrial, telecom, medical, instrumentation, test and measurement, aerospace, automotive, retail, etc.
- Varying power and performance requirements
- Enabling these usages need some/all of these Foundational technologies
    - Security, Reliability, Deterministic performance, Isolation, confidential computing, telemetry

Foundational Technologies

Confidential Computing

Virtualization

Quality Of Service

Reliability, Availability, Serviceability

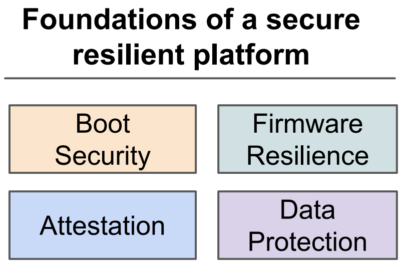
Secure Software

Secure and Resilient Platforms

- Detect/Prevent/Contain exploitation of SW/FW bugs
- Remote attacks, Physical attacks, persistent threats, supply chain attacks
- Maximize uptime and ensure correct operation of the system
- Error containment, ease of system integration, TCO
- Determinism, real time deadlines, bounded worst case execution time

Secure Resilient Platforms

- **Strong roots of trust for - ROT/COT - for update (RTU), detection of corruption (RTD), and recovery (RTRec)**
- **Boot Security** - Measured Boot
  - Protect system compromise during startup
  - Measure each layer of firmware & SW in chain of trust
  - Reliably record measurements for verification
  - Secure debug authorization
  - Attestation: evidence of identity, state, configurations, FW/SW images, etc.
  - Sealing: Encrypted data can only be decrypted by same device in same state
  - Standards and references
    - TCG Device Identifier Composition Engine (DICE)
    - Open Titan - Secure Boot
    - TPM based



- **Firmware resilience** - of platform firmware and critical data such as configurations
  - Protect: f/w Integrity; update authentication, resilient storage, anti-rollback
  - Detect: Corruption of f/w and critical data
  - Recover: From corruptions
  - Guidelines
    - NIST SP 800-193 - Platform Firmware Resiliency Guidelines
    - DMTF Redfish

Secure Resilient Platforms

- **Platform Attestation - secure reporting of platform ID and state to remote verifiers**
  - Provisioning, Job scheduling, Fleet Auditing
  - Supply chain security
  - Support many roots of trusts in the platform
  - Identity of devices and measurement of device firmware and configurations
  - Published attestation policies - Expected measurements, latest versions
  - Standards and references
    - DMTF security protocol and data model (SPDM)
    - PCI-SIG Component Measurement and Authentication (CMA)
    - USB type-C authentication specification
    - IETF Remote Attestation Procedures (RATS)
    - IETF Concise Reference Integrity Manifest (CoRIM, CoMID, CoSWID)

- **Data protection**
  - Sealing: Encrypted data can only be decrypted by same device in same state
  - Non-volatile storage encryption and integrity
  - Memory encryption and integrity
  - PCIe and CXL link Integrity and Data Encryption (IDE)

Secure Software

- Harden software against common and emerging attack techniques

- Available ISA capabilities
  - Privilege separation - M/S/U
  - Page Based memory protection
  - PMP/ePMP
  - Virtualization
  - Defend privilege escalations - executing from user-mode pages, accidentally consuming user-mode data

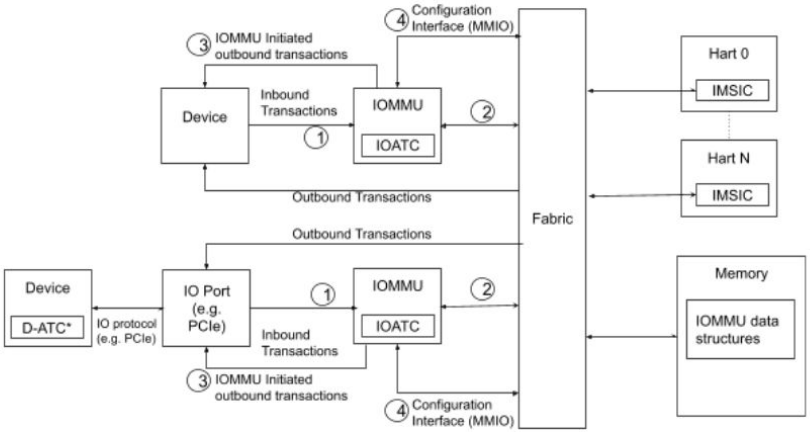
- Control-flow integrity (CFI) - **gap**
  - Commonly known as code-reuse attacks and one of most common exploit techniques
    - e.g., Return-oriented programming (ROP), Jump-oriented programming (JOP)
  - Divert control flow by overwriting control flow variables (return addresses, function pointers)
  - CFI-SIG formed to develop strategy for mitigation or prevention of these techniques

- Memory Safety Enforcement - **gap**
  - Memory safety bugs such as buffer overflows, heap-use-after-free, and stack-use-after-return.
  - Commonly exploited to cause remote code execution, data leakage, privilege escalation, etc.

- Transient execution
  - Transient instructions may perform unauthorized/unintended computations
  - Reveal unauthorized results - secrets - through side channels (Specter, Meltdown, etc.)
  - Mostly a function of secure hardware design
  - Need ISA constructs to control speculation (e.g. FENCE.T) - **gap**

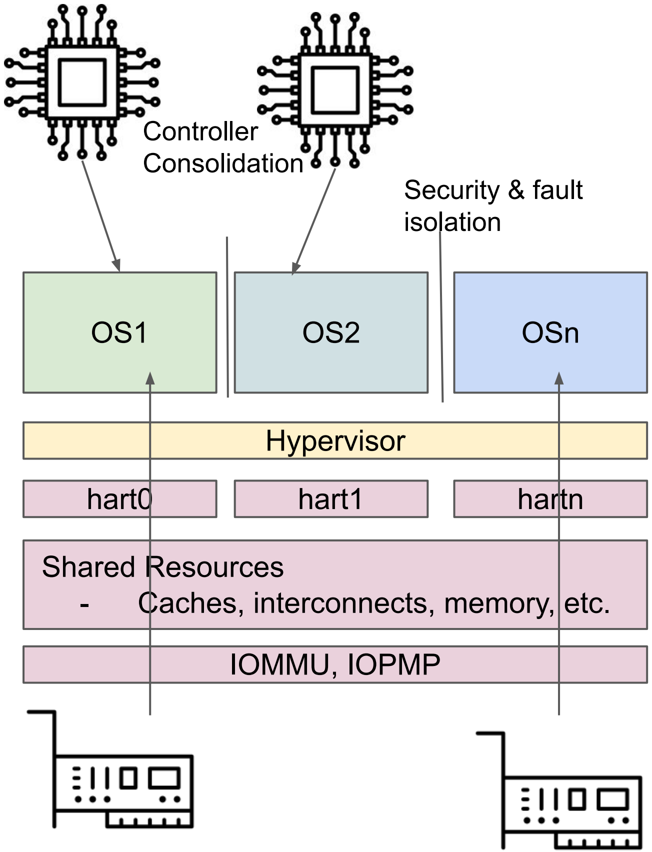
Secure Software

- IOMMU and IOPMP
  - Required even when virtualization not enabled
  - Defend against malicious DMA attacks
  - Defend against driver bugs
  - Protect cross VM attacks and attacks on hypervisor
    - Malicious/buggy VMs using direct assigned devices
  - Functions
    - IO virtual addresses => physical addresses
    - Page based protections to device DMA
    - Address Translation Services
    - Shared virtual memory with devices
    - Quality of Service Enforcement
    - Direct interrupt routing to guests



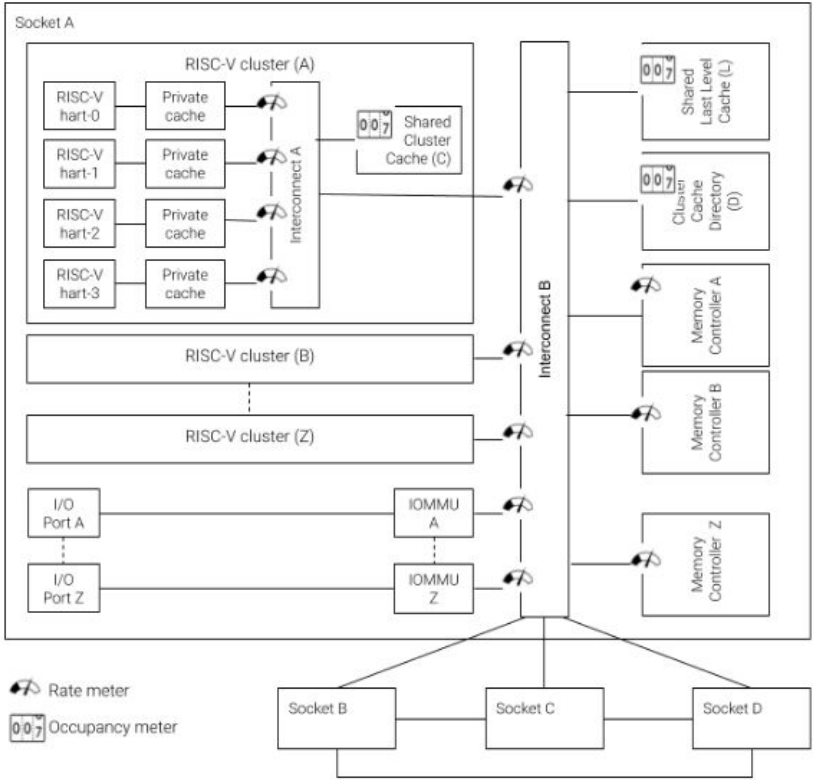
Virtualization

- Consolidate software on multiple embedded controllers to a single platform
- Fault isolation by executing critical applications in separate VMs
- Mixed criticality usages
  - Logically independent software (e.g. IVI, motor control) stacks in their own VMs
- Enhanced security through VM based isolation
- Ease of integration, updates, management of software stacks
- DMA protection and direct device assignment - needs IOMMU
- Challenges
  - Sharing of resources between multiple virtual machines
    - Caches, interconnects, memory controller, power, etc.
  - Lead to non-determinism
  - Need hardware mechanisms for controlling resource usage



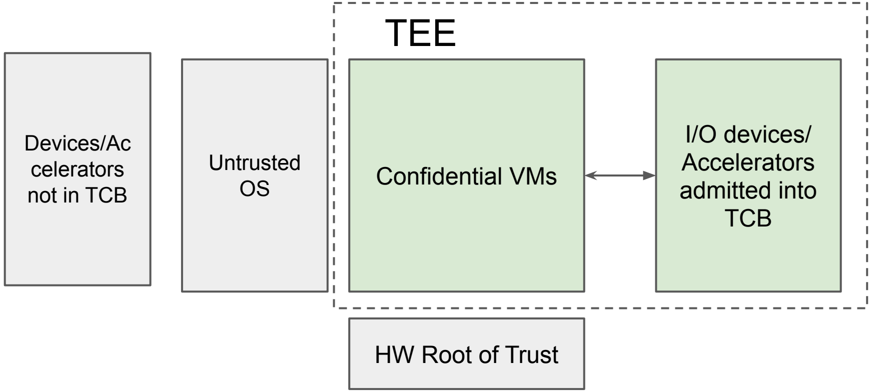
Quality Of Service

- Performance guaranteed in advance by a service level agreement (SLA) to an application.
    - Measured by metrics such as instructions per cycle (IPC), latency of servicing work, etc.
- Interference from unrelated workloads due to
    - Sharing of memory, caches, fabric, etc.
    - Arbitration for shared resources
- Quality of service mechanism needed - **gap**
    - Tag requests by originating workloads
    - Configure resource allocations based on tags
        - Cache capacity
        - Interconnect B/W
        - Memory B/W
    - Monitor resource usage
        - Measure and tune resource requirements
    - Enables techniques such as pseudo-locking critical code/data in caches to meet real time deadlines



Confidential Computing

- Hardware-based trusted execution environment
  - Data-in-use protection
  - Prevent unauthorized access or modification of data
- Devices in untrustworthy/unguarded locations/geographies
- Insider threats
- IP protection
- Address regulations around data processing
  - GDPR, CCPA, HIPAA, PCI, etc.
- TEE capabilities
  - Authenticated launch
  - Confidentiality, integrity, replay protection
  - Attestation
  - Sealing
  - Multi-party computation
  - Confidential computing on CPU as well as accelerators
- Virtual machines as TEE provide most flexible unit of protection
  - Cyclic executive, to RTOS, to general OS



- Standards and references
  - DMTF security protocol and data model (SPDM)
  - IETF Remote Attestation Procedures (RATS)
  - IETF Concise Reference Integrity Manifest (CoRIM, CoMID, CoSWID)
  - DMTF security protocol and data model (SPDM)
  - PCI-SIG Component Measurement and Authentication (CMA)
  - PCIe and CXL link Integrity and Data Encryption (IDE)

Other capabilities

- Cryptography
    - Encryption and hashing - AES, SHA, SM3, SM4, CLMUL
    - Entropy sources
    - Systems needing high performance hardware accelerations such as vector cryptography ISA - gap
    - Data encryption and protection with low overheads
- Low latency and Energy Efficient synchronization primitives
    - Work submission and completion signaling
        - Between applications
        - Between applications and devices/accelerators
- High resolution and synchronized Time
    - Nanosecond granularity
    - Time sensitive networking
- Telemetry - power, performance, health, etc.

Conclusions

- RISC-V provides a strong base for building secure, resilient, performant embedded systems
- Few gaps identified to complete the portfolio of foundational technologies
  - IOMMU, QoS, CFI, RAS, vector cryptography
- Calling on embedded SIG to help define the strategy to address these gaps