presented by

arm

Evolving ACPI Standards for Arm Systems: Advancements in Specification and Implementation

UEFI 2025 Developers Conference & Plugfest

October 10, 2025

Samer El-Haj-Mahmoud and Jose Marinho

Meet the Presenters

Samer El-Haj-Mahmoud

Distinguished Engineer and System Architect, Arm Ltd.

Samer El-Haj-Mahmoud is a Distinguished Engineer and System Architect at Arm Architecture and Technology Group (ATG). He has 25 years of experience specializing in server system architecture, firmware, remote management, and industry standards. His current work focuses on Arm system architecture for servers and PCs. Samer has a long history of contribution to industry standards and related open-source projects, including the UEFI Forum, DMTF, OCP, CXL, UCIe, OpenBMC, TianoCore, and the Arm System Architecture Advisory Council (SystemArchAC).

Meet the Presenters

Jose Marinho

Principal System Architect, Arm Ltd.

Jose Marinho is a System Architect within the Arm Architecture and Technology Group. Jose has been working with firmware, for the Arm ecosystem, for the past 8 years, contributing to firmware interface definitions, collaborating with various open-source firmware communities to promote and progress firmware component interoperability.

Agenda

- Intro on ACPI Model
- Enablement Status in Tianocore
- The Path Ahead

ACPI on Arm

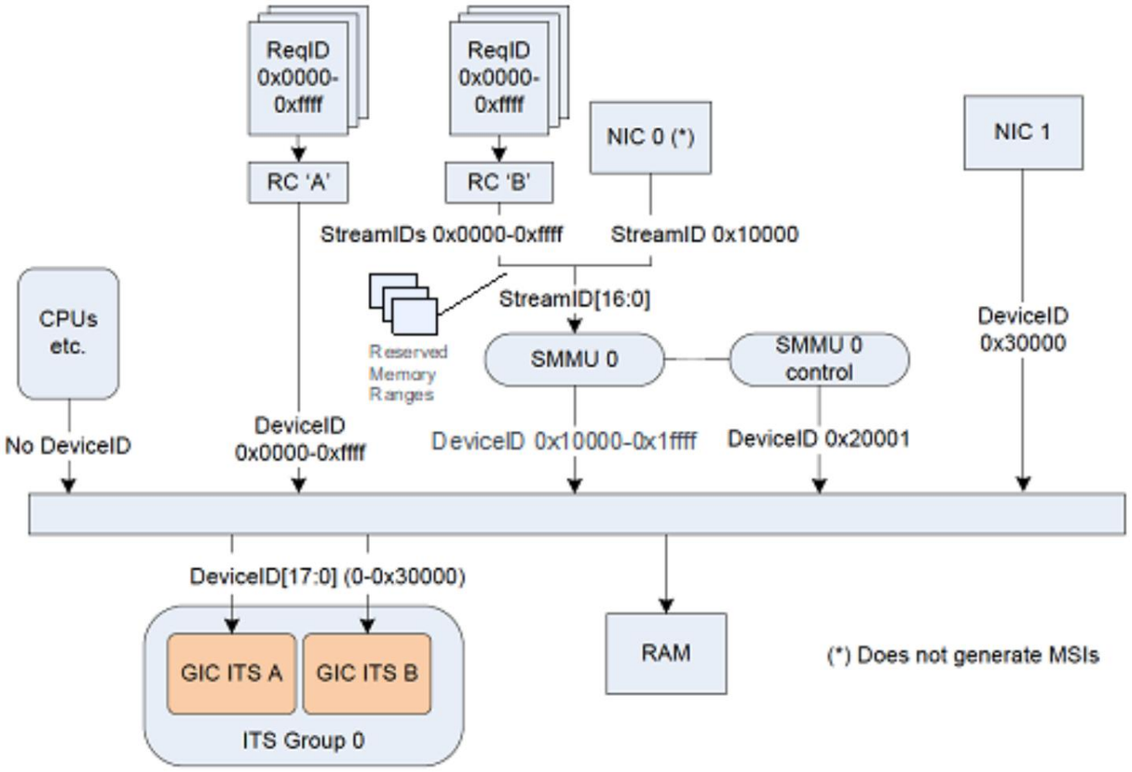
- ACPI describes hardware in a mostly architecture-agnostic manner
- Complements self-describing mechanisms in hardware
- Arm vs other architectures require arch-specific descriptions
  - Some Arm-specifics are in the main ACPI spec
  - Others are in external documented (maintained by Arm)
- Examples of Arm-specifics
  - Performance Monitoring Unit (PMU) or Trace IP
  - I/O Remapping Table (IORT)
    - System Memory Management Unit (SMMU)
    - Generic Interrupt Controller Interrupt Translation Service (GIC ITS)
  - Arm Error Source Table (AEST), used in OS-first RAS.
  - Memory Partitioning and Monitoring (MPAM), used for QoS.
- For Base Boot Requirements (BBR) compliant systems, the ACPI tables are obtained via the UEFI configuration tables.

Multiple APIC Description Table (MADT)

- Common table representing interrupt controllers.
  - Arch-specific controllers have structures in the MADT.
- For Arm
  - Generic Interrupt Controller CPU interface (GICC)
  - Generic Interrupt Controller Distributor (GICD)
  - Generic Interrupt Controller Redistributor (GICR)
  - Generic Interrupt Controller Interrupt Translation Service (ITS)
  - Generic Interrupt Controller version 5 (GICv5) description - WIP
- Arm has supported several versions of the GIC, notably
  - GICv2, that lacked architectural support for MSI, those were added with v2-M
  - GICv3, with architected MSI (needed for PCIe support)
  - GICv4, with enhanced virtualisation capabilities (backwards compatible with v3)
  - GICv5, new architecture, MADT descriptions is currently WIP (https://github.com/tianocore/edk2/issues/11148)
- For GICv2—GICv4 codebases are mature on both EDK2 and Linux

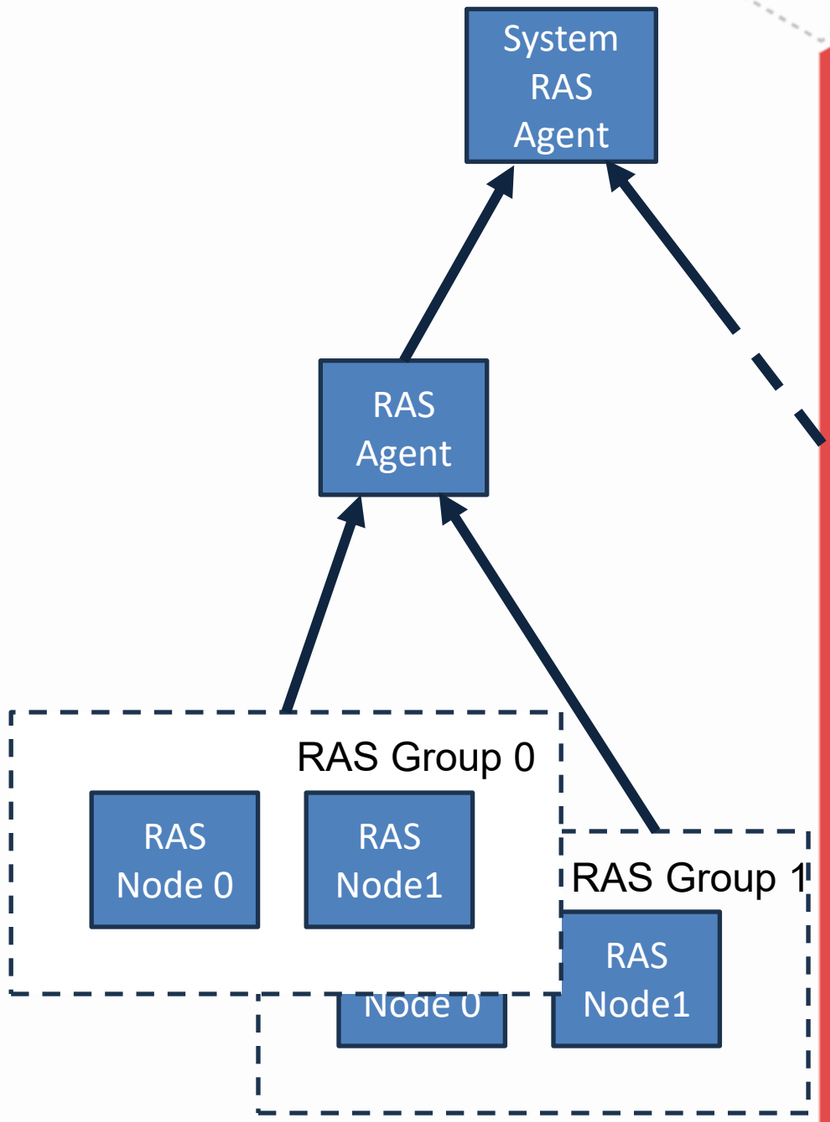
Input/Output Remapping Table (IORT)

- Defined in Arm IO Remapping Table specification (DEN0049)
  - https://developer.arm.com/documentation/den0049/latest/
- Describe relationship between IO-related subsystems
  - Root Ports
  - System Memory Management Unit (SMMU - similar to IOMMU)
  - Interrupt controller ITS
  - Generic devices (with a namespace entry)
  - Performance Monitoring Unit (PMU)
- Describe Memory Ranges
  - Associated devices and the SMMU identifier to the OS
  - The expectation is OS establishes corresponding SMMU mappings when its driver initializes
- Table creation done per-platform in EDK2, framework is enabled upstream
- Fully enabled in Linux



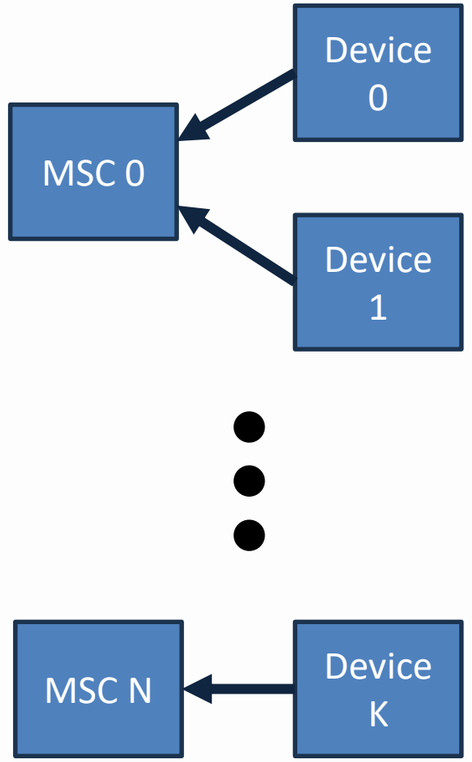
Arm Error Source Table (AEST)

- Defined in Arm ACPI for RAS specification (DEN0085)
  - https://developer.arm.com/documentation/den0085/latest/
- Describe topology of Arm Architecture-compliant RAS nodes and RAS Agents
  - Standard Error information sources, Architected by Arm
- Abstraction focused on OS-first RAS handling
- There can be more RAS nodes in the system hidden from the OS
- AEST complements the standard ACPI Platform Error Interfaces (APEI)
- Table creation done per-platform in EDK2, framework is enabled upstream
- Ongoing consumption enablement in Linux: https://lore.kernel.org/all/20250115084228.107573-1-tianruidong@linux.alibaba.com/



Memory System Resource Partitioning and Monitoring (MPAM)

- Defined in Arm ACPI for MPAM Specification (DEN0065)
  - https://developer.arm.com/documentation/den0065/latest/
- Describe collection of Memory System Components (MSC) that implement the MPAM QoS controls
- Does not describe topological relationship between MSC
- Describes any associated devices (having a {_HID, _UID} association)
  - Purely for power management purposes
- MPAM Firmware-backed specification (DEN0144)
  - https://developer.arm.com/documentation/den0144/latest/
  - PCC-based interface for QoS programming.
  - Enables platform firmware (e.g. auxiliary microcontroller) to present MPAM controls without implementing the Architected interface in HW
- Table creation done per-platform in EDK2, framework is enabled upstream
- Ongoing enablement in Linux: https://lore.kernel.org/all/20250711183648.30766-1-james.morse@arm.com/



Software Delegated Exception Interface (SDEI)

- Defined in Arm SDEI specification (DEN0054)
  - https://developer.arm.com/documentation/den0054/latest/
- SDEI is a software generated non-maskable interrupt framework
- The SDEI framework is made up an SMC interface and an ACPI table to facilitate interface discovery
  - Privileged platform firmware can used the SDEI mechanism to execute OS-installed event handlers
  - Used extensively on Arm to implement EL3 to OS notification for RAS use-cases
  - The SDEI ACPI table advertises the presence of the SDEI SMC interface, so that the OS can safely start interacting with it
- Framework widely supported in TF-A, mature consumption in Linux
- EDK2 is responsible for creating the ACPI SDEI table

Arm IP

- Described over three different specifications:
  - ACPI for Arm Components (DEN0093)
  - ACPI for CoreSight (DEN0097)
  - ACPI for CoreSight PMU (DEN0117)

- Collection of:
  - DSDT Device descriptions (_HID and _CRS requirements)
  - Static ACPI tables
    - AGDI – request Diagnostic Dumps
    - APMT – Performance monitoring table
- Devices described
  - System PMUs
  - Interconnects
  - UART PL011 serial port controller
  - CoreSight Debug/Trace

Other Tables of Relevance

- Fixed ACPI Description Table (FADT)*
  - ARM_BOOT_ARCH
    - Specifies if Arm Power State Coordination Interface (PSCI) is implemented
    - Describes conduit {SMC/HVC} for PSCI or any other supervisory calls.
- Processor Properties Topology Table (PPTT)*
  - Describes processor and cache topology
    - Captures heterogeneity of cache hierarchy relevant for big.LITTLE systems.
- Generic Timer Description Table (GTDT)*
  - Describes architected timers
    - System register based
    - Memory mapped
  - Describe watchdogs

(*) table in the main ACPI specification

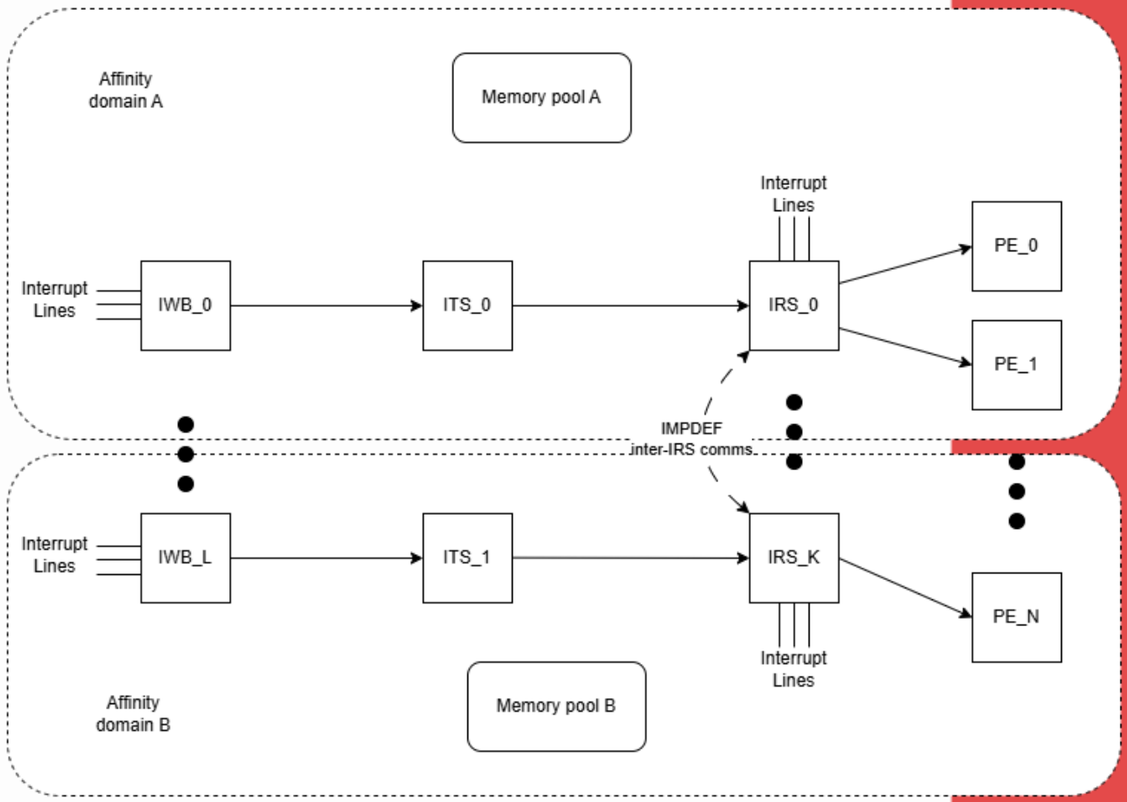
Fixed Function Hardware (FFH)

- Arm platforms are hardware-reduced
- This means, e.g., General Purpose Event blocks (GPE) do not exist, and Generic Event Devices (GEDs) must be used instead
- Additional HW-reduced Arm aspects are defined in the Arm Functional Fixed Hardware (FFH) specification (DEN0048)
  - Define Collaborative Performance Processor Control on Arm
    - Relies on Arm architected performance counters to create an abstract representation of CPU performance
  - FFH OpRegion
    - Enabled TrustZone calls from AML (using SMC or FF-A)

The Path Ahead

Generic Interrupt Controller v5 ACPI

- GICv5: new Arm interrupt controller Architecture
- Composed of:
  - Interrupt Routing Service (IRS)
  - Interrupt Translation Service (ITS): turns MSIs onto processor events/interrupts.
  - Interrupt Wire Bridge (IWB): turns wired events into MSIs
- Specification changes
  - ACPI Code-first ECR can be reviewed https://github.com/tianocore/edk2/issues/11148
  - IORT changes still under internal discussion - expected to define a new IWB IORT node
- EDK2 enablement ongoing.
  - EDK2 GIC driver upstreamed
  - ACPI table creation for reference platform is WIP
  - ACPI Component Architecture (ACPICA) changes WIP

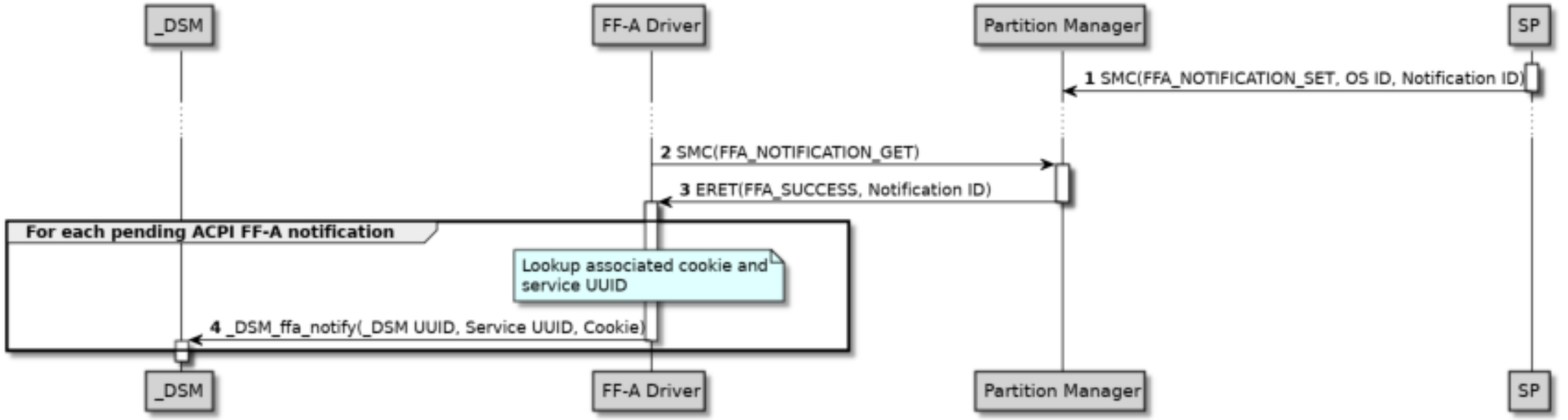


SMMU Driver

- Implementation of the IOMMU_PROTOCOL for Arm SMMU
- Enable sandboxing UEFI drivers from accessing unwanted areas of memory at boot
- Can populate the RMR nodes in the IORT table – allow boot-time drivers to continue executing at runtime
- EDK2 enables statically defined RMR regions
- Ongoing work to support dynamic RMR regions based on the drivers that persist across handoff to the OS

ACPI FF-A

- Enable FF-A FFH OpRegions
  - Enable OS support for SMC/FF-A method calling from AML code.
  - Requires FF-A driver support to register an OpRegion handler.
  - WIP for the Linux FF-A driver.
- Enable FF-A notifications.
  - Akin to a software-triggered interrupts



Example ACPI FF-A notification signalling

Questions?