presented by

arm

UEFI in Arm Platform Architecture

Fall 2017 UEFI Seminar and Plugfest
October 30 – November 3, 2017
Presented by Dong Wei (Arm Limited)

Agenda

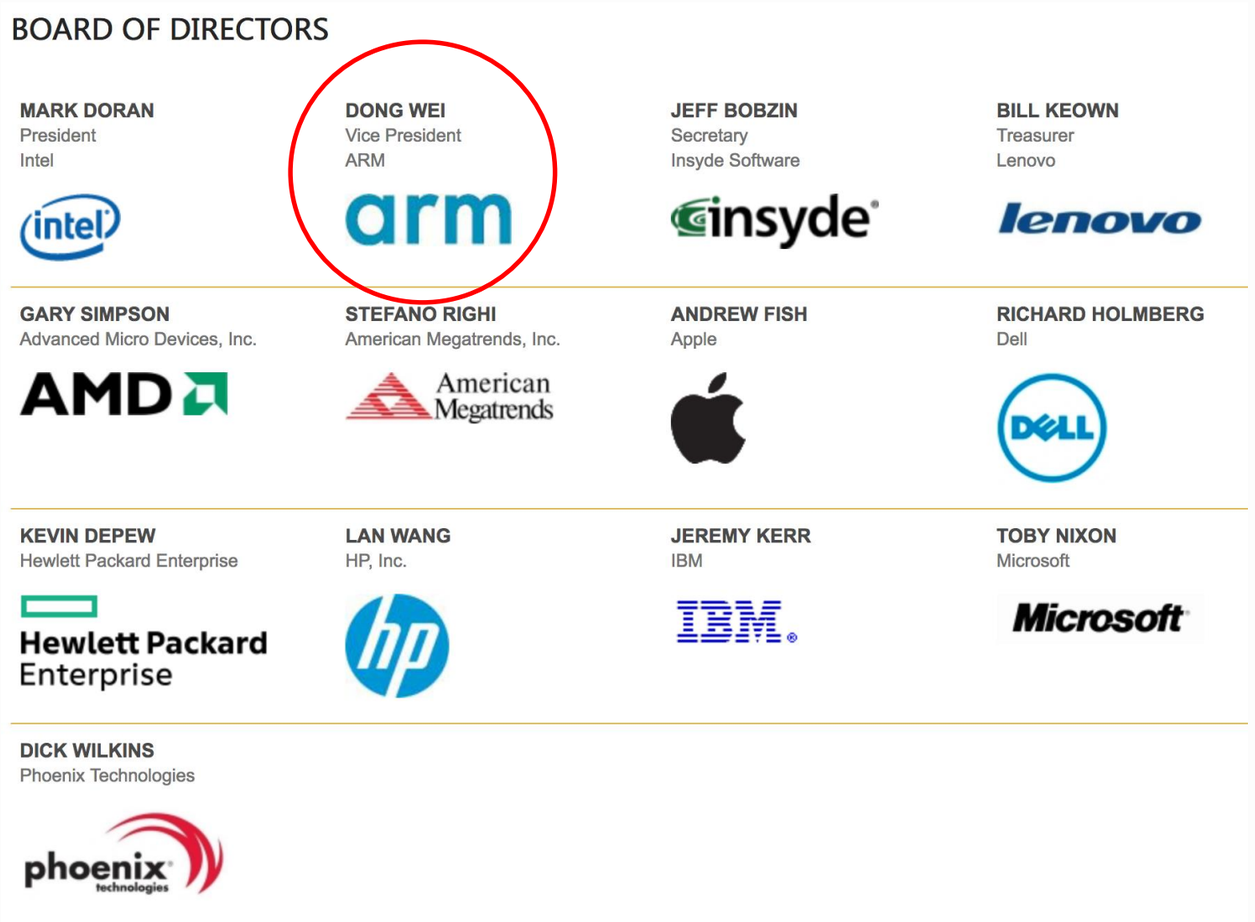
- Arm @ UEFI Forum
- UEFI in Servers
- UEFI in Embedded Systems
- Questions

Arm @ UEFI Forum

UEFI Forum

Arm Limited is now a Promoter of the UEFI Forum

- Board member
- Vice President (Chief Executive)
- Chair of the UEFI Test WG
- Co-Chair of the ACPI WG



ACPI

ACPI 6.2 was released in May

- Better support for cache topology discovery
- Improved PCC channels
- Alignment with Software Delegated Exceptions
- IORT updated in May which improved SMMUv3 support

Current work

- PCC operating regions: better ways for ASL to talk to platform controllers
- CoreSight
- SMMU and RAS
- MPAM

Anything else we should be looking at?

UEFI

UEFI 2.7 was released in May, no major updates affecting Arm bindings

SCT

- UEFI v2.6A SCT is accelerated (Final Draft under Membership Review until Oct 27)
- Investigating new development model

UEFI in Servers

Server Architecture

Base System Architecture (BSA)

- Defines hardware requirements

Base Boot Requirements (BBR)

- Defines firmware requirements

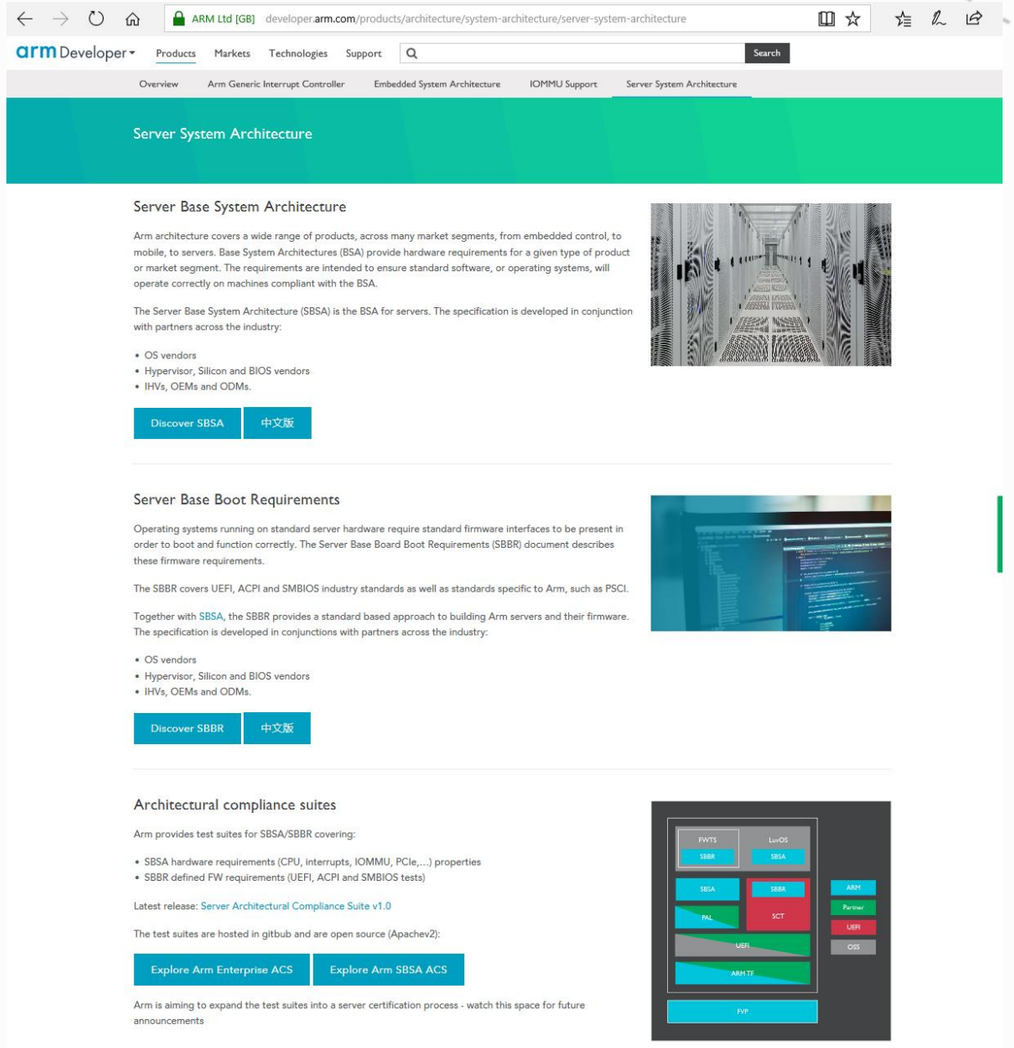
These specifications require a minimum set of hardware and firmware implementations that will ensure OS and firmware will interoperate

SBSA/SBBR are the BSA/BBR for the server systems

- Developed using feedback from vendors across the industry (Silicon vendors, OSVs, Hypervisor vendors, BIOS vendors, OEMs and ODMs)
- SBBR defines the required, recommended and optional UEFI, ACPI and SMBIOS interfaces

SBSA are SBBR are now available at https://developer.arm.com/

- Current versions are SBSA v3.1 and SBBR v1.0. No click through license required.



SBSA and SBBR Architectural Compliance Suites

SBSA test covers

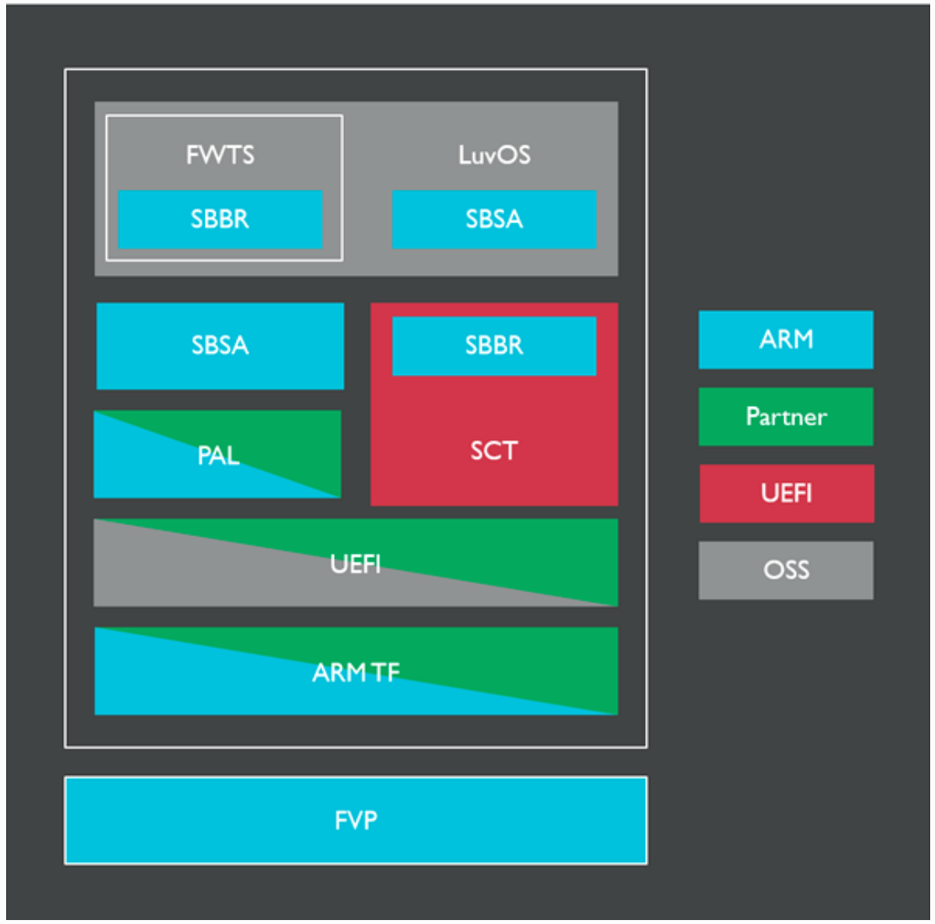
- SBSA CPU properties
- SBSA defined system components
- SBSA rules for PCIe integration
  - Based on the PCIe specification
  - Based on standard OS drivers with no quirks enabled

SBBR test covers

- UEFI testing based on the UEFI SCT
- ACPI testing based on FWTS
- SMBIOS testing

V1.0 released!

- https://github.com/ARM-software/sbsa-acs
- https://github.com/ARM-software/arm-enterprise-acs



SBBR Status and Plans

Drafting SBBR 1.1

Requires newer FW revisions

- ACPI6.2, UEFI2.7, SMBIOS 3.1.1, PSCI 1.1

Require PSCI as the only secondary core boot method

Require AArch64 native UEFI Drivers and Applications

Newer features

- Generic Event Devices and interrupt-signalled Events
- Software Delegated Exception
- Heterogeneous Memory Attribute
- Redfish Host Interface
- TCG TPM Trusted Boot and relationship with UEFI Secure Boot and Arm TF secure boot

arm ServerReady

- Benefits
- Trust
- Quality
- Confidence

UEFI in Embedded Systems

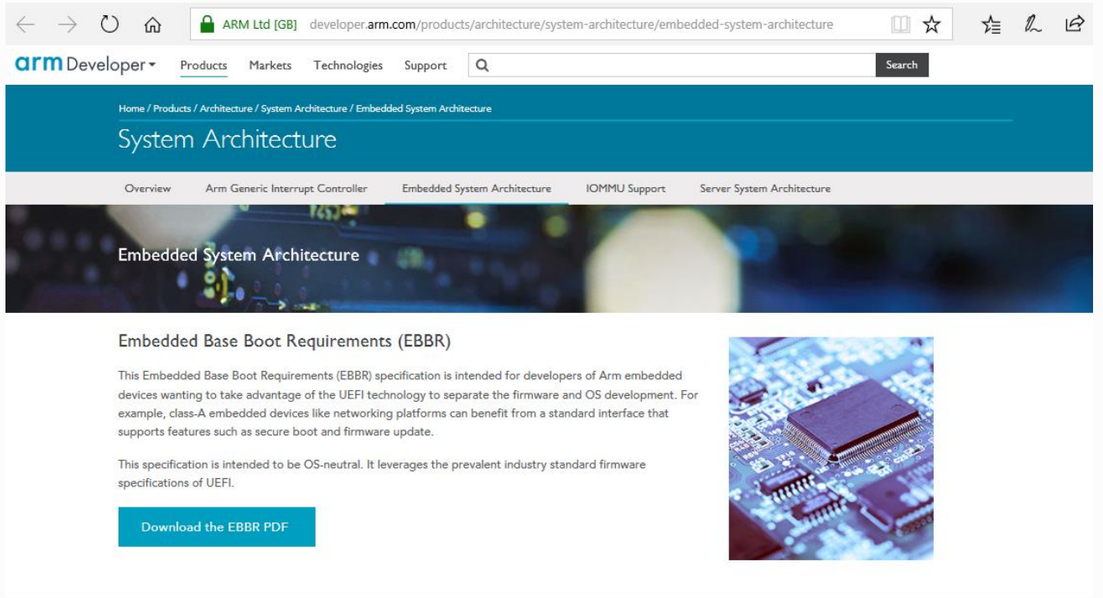
Embedded Architecture

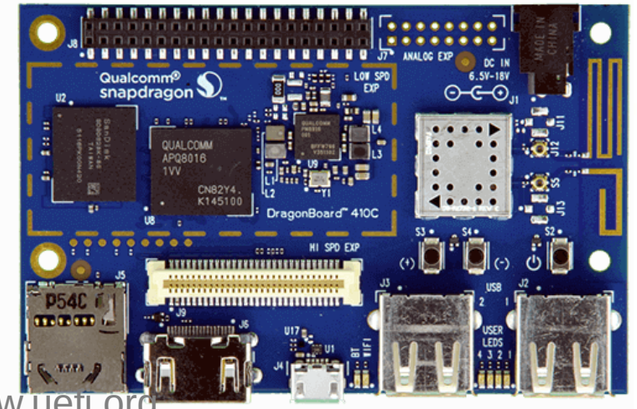
Base System Architecture (BSA)
- Defines hardware requirements

Base Boot Requirements (BBR)
- Defines firmware requirements

These specifications require a minimum set of hardware and firmware implementations that will ensure OS and firmware will interoperate

EBBR is the BBR for the embedded systems
- Under development





Rationale and Scope

To define the firmware and bootloader requirements for Arm embedded devices, which require a standardized interface between the platform (firmware) and the OS (system software)

Differs in scope from Server Base Boot Requirements (SBBR)

- SBBR is targeted at general purpose compute
  - Strictly requires UEFI and ACPI
  - Driven by requirements from enterprise OS vendors
- EBBR targets embedded when firmware/OS separation is required
  - For example, A-class embedded devices like networking platforms
  - Friendly to U-Boot and Devicetree implementations
  - Intended to support embedded OS vendors and community distributions

Rationale and Scope

This specification is intended to be both practical and aspirational at the same time

- Practical in that it seeks to solve the problems OS vendors face today with regard to controlling the boot path of the platform
- Practical in that it codifies what is feasible now or in the near future (ex. UEFI interfaces in U-Boot)
- Aspirational in that it specifies important features which may still require development to achieve (ex. U-Boot support for UEFI Runtime services)
- Aspirational in that it seeks to *move the bar* on bare minimum expectations on how Arm platforms should behave

Overview

This specification defines a set of base firmware requirements for embedded devices

- Compliant platforms must confirm to all the mandatory interfaces specified in EBBR
- Firmware must implement a subset of the UEFI specification
  - Must implement boot services
  - Must implement runtime services for changing boot variables
- May supply either Devicetree or ACPI system description data, but must conform to the specifications for the selected mechanism

This specification is intended to be OS-neutral. It leverages the prevalent industry standard firmware specifications of UEFI and can be implemented using U-Boot and Devicetree

Details – UEFI

Compliant with v2.7 of UEFI spec or later

UEFI running at EL2 if hypervisors are supported, EL1 otherwise

If system boots from local block storage, storage must be GPT formatted and use a FAT formatted system partition

Support running UEFI 64-bit PE/COFF loaded images containing only A64 code

Optionally may implement Secure Boot

- But if implemented, it must conform with UEFI specification for Secure Boot

Details – UEFI Runtime Services

UEFI Runtime services must be implemented

- Must support running at either EL1 or EL2 (if implemented)

UEFI Memory Map minimum page size is 4K

To support 64K pages in the OS, all 4k pages within the same 64k block must use identical page attributes

SetTime()/GetTime() must be implemented for interacting with real time clock

Details – UEFI Runtime Services – System Reset

EfiResetCold()/EfiResetShutdown() must be implemented, and shutdown must not reset the system

EfiWarmReset() must be implemented if used to support capsule updates

System software shall use EFI runtime services for all system resets

- Must not call PSCI directly (UEFI may have work to do)

Details – UEFI Runtime Services – Variables

Non-Volatile variables must be implemented and may not be emulated with RAM

Non-Volatile storage by runtime services must not require OS intervention (ie. to store to eMMC)

Questions

- Do we have the correct scope?
- Are the requirements too tight, or not tight enough?
- Are runtime services something useful for this market segment?
  - Runtime services are used to manipulate variables after booting an OS
  - The alternative is to do all variable manipulation from a UEFI application before ExitBootServices() is called.

EBBR

Specification Development:

Specification Location: https://developer.arm.com/products/architecture/system-architecture/embedded-system-architecture

Mailing list: arm.ebbr-discuss@arm.com.

Thanks for attending the Fall 2017
UEFI Plugfest

For more information on the UEFI Forum and UEFI Specifications, visit
http://www.uefi.org

presented by

arm