Is your Pi “ServerReady”?  
Embracing UEFI and ACPI at the Edge

UEFI 2020 Virtual Plugfest  
Wednesday, April 29

Presented by Andrei Warkentin (VMware) and Samer El-Haj-Mahmoud (Arm)
Meet the Presenters

Samer El-Haj-Mahmoud
Senior Principal Architect
Member Company: Arm

Andrei Warkentin
Arm Enablement Architect
Member Company: VMware
Agenda

- Intro to Arm ServerReady
- What about SBCs?
- Making Pi “Boring”
- Challenges
- Demo
- Call to Action
Arm ServerReady

- Makes Arm servers a multi-player in horizontally-integrated ecosystem.
- Blueprint for “just works” Arm-based infrastructure
  - Industry standards (PCIe, UEFI, ACPI, SMBIOS, etc...)
  - Server Base System Architecture (SBSA).
  - Server Base Boot Requirement (SSBR).
  - Arm’s Server Architectural Compliance Suite (ACS).
- Just as “boring” to consume as x86 counterparts.
  - Not weird to OEM/ODMs.
  - Not weird to IT.
  - Not weird for software developers.
  - Easily support Commercial Off-the-Shelf (COTS) and proprietary system software.
  - ...a horizontally-integrated ecosystem.

A single binary OS distribution to boot on any system — enough to install missing drivers.
Server Base System Architecture (SBSA)

- Hardware requirements for OS vendors.
- Multiple levels, each corresponding to a set of increasing conformity:
  - Processor / memory features.
  - UART, Interrupt Controller, SMMU.
  - PCIe® technology integration.
  - Common I/O peripherals like USB/SATA.
  - …
- New levels added as architecture evolves, including/refining previous levels.
- Currently at Level 6.

**Example: SBSA L6**

**Processor Element (processors cores)**
- Up to $2^{28}$ PEs
- AArch64 at all Exception Levels
- Must to provide EL2 and EL3
- Advanced SIMD extensions.
- Instruction Caches VIPT or PIP
- 16-bit ASID support.
- 4KB and 64KB translation granules at stage 1 and stage 2.
- All PEs are coherent and in the same Inner Shareable domain.
- Where export restrictions allow, cryptography extensions.
- Little-endian support.
- 6 PMU counters, 6 breakpoints, 4 watchpoints
- CRC32, cond. Scalar vector ext.

**System level requirements**
- Interrupts
  - GIv3 (generic MSI support)
  - Standard interrupt numbers
  - No non-standard extensions
- IO virtualization
- SMUv3 (ATSI)
  - Access/Dirty Flags mandated
- Easier page sharing PE and SMMU
- PCIe for all assignable devices
- PCIe enhancements (Appendix E)
  - ECAM
  - On-chip Peripherals

**Memory**
- No deadlocks when accessed by processor or device
- MMIO Peripherals 64Kb apart

**System counter scaling**
- Nanosecond generic counter

**Other**
- Power
- Arm standard wake up timer
- Peripherals
- Arm standard UART
- Arm standard WDog
- EHCI v1.0 or later
- XHCI v1.0 or later
- AHCI v1.3 or later
Server Base Boot Requirements (SBBR)

OSV firmware requirements: adopt industry standards where they exist, complement with Arm-specific interfaces where mandatory.

Beyond Servers?

- Non-Server Arm devices are traditionally non-traditional.
- Even higher-end devices are treated as embedded.
  - Vertically-integrated ecosystem.
  - Very little off the shelf software.
  - Custom firmware (sometimes not even U-Boot).
  - Custom system software (non-upstreamed kernels, custom distros).
  - Poor support after just a few years.
  - Vendor lock-in.
- This class of systems is positioned for IoT/Edge, but there’s no common ecosystem. This is getting in the way of adoption and proliferation of Edge/IoT deployments.
The Light at the End of the Tunnel

- Standards aren’t just for servers.
- At TechCon’19 Arm extended “ServerReady” to other segments, where COTS “general purposes” system software needs to run. (see Arm whitepaper)
- “Server” requirements → “System” requirements (for all “A class” Arm systems)?
  - Server devices (e.g. datacenter, infra edge)
  - Client devices (e.g. laptops)
  - Edge devices (e.g. IoT gateways)
  - Facilitates creating a horizontally-integrated ecosystem.
  - Based on SBSA, SBBR, SBSG (security)
- What about EBBR?
  - Embedded Base Boot Requirements stop at describing firmware/OS hand-off.
  - Don’t standardize on describing or interfacing hardware.
  - Examples:
    - Raspberry Pi with U-Boot + FDT, loading UEFI OS loader
    - HP Envy x2 WoA laptop: UEFI + ACPI, but can’t boot a non-Snapdragon OS image.

...will it blend-boot???
Let’s Do Something!

- **Why Raspberry Pi 4?**
  - High-volume, low-cost ($50) device for everyone.
  - People may have not heard of Arm, but they have heard of the Raspberry Pi.
  - Pi will be around for a while (unlike a few other boards I can think of).

- **Is it feasible?**
  - Pi 4 finally has GICv2. XHCI for USB
  - Can safely ignore PCIe (no physical slot) – Level 2/3 SBSA compliance should be possible.

- **How should it be done?**
  - Out in the open community (true OSS project).
  - Set an example and show others how it’s done.

I’m a server buff - and why should I care?
- No good widely available client platforms today to build a good mass of developers.
- A $50 device that “just works” almost like a “server” is a great enabler for the Arm server ecosystem.

What could you boot with this?
- SBBR/SBSA OSes (ESXi, RHEL, Windows)
- EBBR (Device Tree) OSes: SUSE, Ubuntu, Debian, NetBSD, FreeBSD, OpenBSD
- EBBR (ACPI) OSes?
Pi 4 Firmware Task Force

- 10/2016 – Microsoft publishes 32-bit UEFI for Pi 2/3, as part of Windows IoT Core.
- 11/2016 – Ard Biesheuvel publishes minimal 64-bit UEFI for Pi 3 with TF-A.
- 11/2017 – Andrei Warkentin begins work on RaspberryPiPkg for Pi 3, building on Ard’s initial work and ms-iot.
- 11/2017 - Integration of a DWC2 UEFI USB host driver (originating with Linaro)
- 12/2017 – Booting Pi 3 SUSE Linux with Device Tree
- 04/2018 – First Pi 3 Windows on Arm boot!
- 10/2018 – First Pi 3 ESXi-Arm boot!
- 02/2019 – RaspberryPiPkg upstreamed to TianoCore edk2-platforms by Pete Batard.
- 09/2019 – VMware + Arm start community project to add Pi 4 support to Pi 3 code base..
- 10/2019 – ESXi-Arm demoed booting on Pi 4 at the Arm TechCon.
- 12/2019 – Jeremy Linton contributes UEFI PCIe RC driver for XHCI support.
- 12/2019 – First ever SBBR boots for Linux and NetBSD.
- 01/2020 – GENET ACPI support, Windows 10/WinPE on Pi 4 (with limitations)

Staging and releases https://github.com/pftf
Official blog https://rpi4-uefi.dev/

Andrei Warkentin (VMware), Samer El-Haj-Mahmoud (Arm), Pete Batard (Akeo), Ard Biesheuvel (Arm), Jeremy Linton (Arm), Jared McNeill (NetBSD)
PFTF Github

- [https://github.com/pftf](https://github.com/pftf)
- Forks from TianoCore ([edk2](https://github.com/pftf/edk2), [edk2-platform](https://github.com/pftf/edk2-platform), [edk2-non-osi](https://github.com/pftf/edk2-non-osi))
  - Kept relatively up-to-date with upstream
  - Used as staging branches before submitting patches to TianoCore
- Releases for **RPi4** and **RPi3**, built directly from TianoCore upstream
- Arm Enterprise ACS test reports: [https://github.com/pftf/acs-reports](https://github.com/pftf/acs-reports)
Pi ServerReady Blog

- [https://rpi4-uefi.dev/](https://rpi4-uefi.dev/)
- Background, History, FAQ, Links
- Detailed status kept at: [https://rpi4-uefi.dev/status/](https://rpi4-uefi.dev/status/)
Pi 4 Firmware Status

**Trusted Firmware**
- Using upstream Pi 4 TF-A
- PSCI for CPU boot and power-off/reset.
- PL011 support.
- TBD: SDEI

**Devices**
- RNG
- miniUART and PL011 serial.
- File-backed NVRAM.
- UEFI framebuffer via Pi mailbox (supporting a range of virtual resolutions).
- DWC2 (USB2) controller on Type-C.
- xHCI (USB3) via custom PCIe controller.
- Arasan SDHCI for SD card.
- GENET (SoC Ethernet) SNP driver. (upstreaming in progress)
- TBD: EMMC2 (new Pi 4 SDHCI controller)

**Miscellaneous**
- PXE booting
- iSCSI
- RAM disk
- HTTP(s) boot

**System Description**
- SMBIOS
- EBBR via Device Tree (including overlays)
- Early ACPI support (xHCI, GENET)

**Challenges**
- PCIe (and thus xHCI) have 3GB DMA limit
- Legacy Pi devices have 1GB DMA limit and translation.
- This doesn’t violate SBBR/SBSA, but not done before.

---

*Upstreaming and development in progress.*

www.uefi.org
Exposing PCIe® USB XHCI via ACPI

1. Straight-up ACPI MMIO device
   1. Configure PCI in UEFI.
   2. Patch DSDT with USB base address.
   3. Deal with PCIe bus mastering being disabled on UEFI exit.

2. Expose PCIe
   1. "Almost ECAM" (wrong alignment)
   2. Expose just the USB device config space as ECAM.

...in all cases it’s still non-cache coherent.

(2) Violates SBSA Levels > 0: because it’s non-cache coherent, ECAM range has wrong alignment and can’t scan for B:D:F > 0:0:0.
XHCI and legacy DMA constraints.

- A major headache.
- The legit way to support is through the _DMA object defined for the bus controller enumerating affected devices.
- This can deal with both bus translations and limits.
- Full support in NetBSD thanks to Jared.
- Have 5 lines of code patch for Linux.
- TBD for Windows.

```
Device (SCB0) {
  Name (_HID, "ACPI0004")
  Name (_CRS, ResourceTemplate() { ... })
  Name (_DMA, ResourceTemplate() { ... })
  Device (XHC0) { ... }
}
```

We might need to introduce an ACPI boot profile that limits useful RAM to 3GB and hides all legacy devices, at least to deal with OS versions that don’t support _DMA.
Call to Action

Raspberry Pi 4
• Always looking for more help!
• #rpi4-uefi-dev on Arm Developer-Ecosystem Discord server.
• https://rpi4-uefi.dev for Discord server link
• Contact us: awarkentin@vmware.com and samer.el-haj-mahmoud@arm.com.

Other platforms
• nVidia Tegra-based platforms (Jetson Nano, Xavier)
• Rockchip RK3399 platforms (Pinebook Pro, Orange Pi, Rock64, Rock960, Nano Pi-M4, etc)

Other firmware
• UEFI does not have to be TianoCore
• U-Boot’s UEFI implementation adding ACPI tables in addition to DT
• …small steps towards both EBBR and SBBR-compliant systems built with U-Boot.

Join Arm standards committee!
• Arm Server Advisory Council (ServerAC)
• Shaping the standards and requirements for servers (and future edge devices)
Demo!
Questions?
Thanks for attending the UEFI 2020 Virtual Plugfest

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