Making RISC-V EBBR Compatible

Atish Patra
Western Digital
Linux Plumbers 2020
Why EBBR for RISC-V?

Make the boot process “boring”

• Embedded Base Boot Requirement (EBBR) is a community driven booting specification
  – “Provides an interface between platform firmware and an operating system that is suitable for embedded platforms.”

• Currently supports only ARM64 but aimed to be generic

• Well defined and proven with many ARM64 platforms

• Following this specification for RISC-V would make the RISC-V boot process standard
  – Generic implementation for firmware/boot loaders
  – Easy to support various Linux distributions that already supports EBBR
EBBR Specification

Requirements

• Logistics requirements
  – The specification is hosted under “ARM-software” on Github
  – The discussion happens at “boot-architecture@lists.linaro.org”
  – Licensed under Creative Commons
  – Copyright "Arm Limited and Contributors"

• Requires some modification to improve architecture independence

• Technical requirements
  – A subset of UEFI boot/runtime services support
  – Device Tree/ACPI support
  – Multiprocessor Startup Protocol
  – Firmware Storage

• No major problems for RSIC-V
  – But platform reset handling raises some issues
RISC-V EBBR Compliance

Logistic requirements

• Ongoing discussion

• Logistic requirements:
  – Open to community contribution
  – RISC-V section/copyright can be added (Agreed)
  – Proposals to move the specification to UEFI.org (Not agreed yet)
RISC-V EBBR Compliance

Technical Requirements

• UEFI support
  – Full UEFI support for RISC-V Linux is available
    • [https://lkml.org/lkml/2020/8/19/1252](https://lkml.org/lkml/2020/8/19/1252)
  – Upstream U-Boot already supports UEFI for RISC-V
  – EDK2 working [port](https://github.com/edk2) is already available
  – Reset System as a boot service is missing

• Multi-process startup protocol
  – The OS should boot only in Supervisor mode
  – a0 should contain hartid
  – a1 should contain the device tree location in memory
  – SBI v0.2 required
  – Ordered booting with SBI HSM extension
  – “boot-hartid” property under /chosen node is mandatory

• Firmware storage requires GPT partitioning
  – already supported in U-Boot for RISC-V
  – EDK2 support in progress
RISC-V EBBR Compliance Issues

What’s missing?

- EBBR mandates reset system only during boot service
- Current known users:
  - GRUB (command reboot)
  - iPXE (commands reboot, poweroff)
  - EFI shell (command reset)
  - OpenBSD boot prompt (commands reboot and poweroff)
- No standard reset system interface in RISC-V privileged specifications and RISC-V SBI specifications
- No standard MMIO device for system reset across architectures
- Does the reset system have to be mandatory during boot service?
RISC-V EBBR Compliance issues

Possible solutions

• **Solution1**: Platform and Hypervisor specific RESET mechanism
  – Every hypervisors will have to emulate their own MMIO device (or SBI call) for system reset
  – U-Boot has CONFIG_SYSRESET that can be overridden for individual platforms (and hypervisors)
  – All UEFI implementations (EDK2) have to carry platform (and hypervisor) specific reset code
  – No way to authenticate system reset from non-secure OS
RISC-V EBBR Compliance issues

Possible solutions

• Solution 2: SBI RESET extension
  – Already under review ([https://lists.riscv.org/g/tech-unixplatformspec/message/49](https://lists.riscv.org/g/tech-unixplatformspec/message/49))
  – System reset via standard SBI interface
  – Standard mechanism for all hypervisors to provide system reset
  – Allows supervisors to use the same interface for system reset
  – System reset request from a non-secure OS authenticated via secure monitor
  – SBI RESET extension is optional
    • It allows the platforms to define non-standard platform specific function if required
  – EFI_RESET_SYSTEM implementation can fallback to platform RESET if SBI RESET is not available