Fixing the boot process in RISC-V

Atish Patra <Atish.Patra@wdc.com>
Objective

• Make the boot process as standard and boring as possible
• Bootloader/Firmware support
  • Follow all the upstream boot stages
  • Linux kernel loading via network/sd card
  • Easy porting of mainstream firmware/boot loaders to RISC-V
    • U-Boot Proper and U-Boot SPL
    • Coreboot
    • EDK2
    • Linuxboot ?
• Follow sequential booting protocol
• Secure boot
What is OpenSBI?

- OpenSBI is an open-source implementation of the RISC-V Supervisor Binary Interface (SBI) specifications
- Aimed at providing RUNTIME services in M-mode
- Helps avoid fragmentations in SBI implementations
- Provides support for reference platforms
- Can be used as a separate firmware or included as a library
  - Next stage can be loaded as a payload or dynamic address passing
  - U-Boot SPL/Coreboot using dynamic firmware
  - EDK2 is using it as a library
RISC-V Upstream Boot Flow

Follows commonly used multiple boot stages model

- Supported only on HiFive Unleashed
- Follows a standard boot flow
- U-Boot binary as the payload to OpenSBI
- ZSBL/FSBL is SiFive specific and can be replaced by Coreboot/U-Boot SPL
- OpenSBI is a RISC-V specific runtime service provider
Upstream status

Rapid progress: traditional full boot support expected by year end

• OpenSBI
  – Default in Buildroot, Yocto/OpenEmbedded and the QEMU “BIOS”
  – Fedora/Debian provides images available with OpenSBI binary

• U-Boot
  – U-Boot-2019.07 release has HiFive Unleashed S-mode support with SMP
  – Boot via network supported
  – MMC boot support coming in 2019.10
  – EFI support for RISC-V available
  – U-Boot SPL support only for QEMU

• Coreboot
  – Upstream can boot HiFive Unleashed and Qemu
  – No SMP support

• Grub
  – RISC-V support available upstream

• Linux Kernel
  – Upstream kernel boots in QEMU
  – 5.3 kernel works with OpenSBI+U-Boot on HiFive Unleashed
Proposed hart hotplug extension

- struct sbiret sbi_hart_add(unsigned long hartid, unsigned long start_addr, unsigned long priv)
  - Asynchronous
  - Caller should confirm if hart is really up or not
- struct sbiret sbi_hart_remove()
  - Synchronous call, doesn’t expect to return
  - Only called by self hart
- struct sbiret sbi_hart_status(unsigned long hartid)
  - Query the status of the hart
  - Caller should be aware of the fact that state may change during the call

* https://github.com/riscv/riscv-sbi-doc/pull/23
Future work

Toward a stable and easy to use boot ecosystem

• SBI v0.2 specification
• Hart hotplug extension in SBI
• Sequential cpu bringup instead of random boot in Linux
• EFI stub support in Linux kernel full UEFI support
• U-Boot SPL support for hardware
• Coreboot SMP support
• Oreboot(Coreboot in “Rust”) support
• EDK2 project upstreaming complete
• Secure boot
• Anything else ?
Constraints on using OpenSBI Library

• Same GCC target options (i.e. -march, -mabi, and -mcmodel) need to be used for the external firmware and OpenSBI sources

• External firmware must create per-HART non-overlapping:
  1. Program Stack
  2. OpenSBI scratch space (i.e. struct sbi_scratch instance with extra space above)

• Two constraints in calling any OpenSBI functions from external firmware:
  1. MSCRATCH CSR of calling HART must be set to its own OpenSBI scratch space
  2. SP register (i.e. the stack pointer) of calling HART must be set to its own stack

• External firmware must also ensure that:
  − Interrupts are disabled in the MSTATUS and MIE CSRs when calling sbi_init()
  − sbi_init() is called for each HART that is powered-up at boot-time or in response to a CPU hotplug event
  − sbi_trap_handler() is called for M-mode interrupts and M-mode traps
Reference

• OpenSBI
  – https://github.com/riscv/opensbi

• SBI
  – https://github.com/riscv/riscv-sbi-doc

• EDK2
  – https://edk2.groups.io/g/devel/message/46479?p=,,,20,0,0,0::Created,,riscv,20,2,0,33047245