RISC-V bootflow: What’s next?

Atish Kumar Patra
Western Digital System Software Research
Outline

• Current boot flow in RISC-V
• Upstream status
• Random booting
• Ordered booting
• SBI HSM extension
• UEFI boot in RISC-V
• Future work
Current RISC-V Boot Flow

Follows commonly used multiple boot stages model

- Follows an industry standard boot flow based on open-source components
- Can use U-Boot proper as the last stage boot loader
- FSBL is SiFive specific and will be replaced by Coreboot/U-Boot SPL
- OpenSBI is a RISC-V specific runtime service provider
- All harts jump to Linux at the same time and a lottery-based approach chooses the booting hart

ZSBL
(M-mode) (ROM)

FSBL/U-Boot SPL/ Coreboot
(M-mode) (LOADER)

OpenSBI
(M-mode) (RUNTIME)

U-Boot
(S-mode) (BOOTLOADER)

Linux
(S-mode) (OS)

Loads

Jumps

U-Boot SPL loads U-Boot proper as well

Coreboot loads Linux directly

RISC-V specific
Current upstream status

Very good support available in upstream

• U-Boot RISC-V support upstream
  – U-Boot-2019.10 release has SPI/MMC
  – EFI support for RISC-V available
  – U-Boot SPL support for Qemu and Andes AE350
    are in upstream
  – U-Boot SPL support patches for HiFive
    Unleashed is in the mailing list.
  – U-Boot SPL uses firmware dynamic

• Coreboot RISC-V support upstream
  – Both Qemu and HiFive Unleashed supported
  – Uses OpenSBI firmware dynamic

• Oreboot
  – “RISC-V firmware in Rust” by Ryan O’Leary

• OpenSBI
  – Currently at v0.5
  – v0.6 will be this week
  – SBI v0.2 is already available in v0.6
  – HSM extension and hotplug patches are in the
    mailing list and will be merged after v0.6

• LittleKernel (LK)
  – Bootloader for Android devices
  – RISC-V support already available
  – HiFive Unleashed supported patches are
    available

• Nothing on secure boot yet
Random Booting (Present approach)

How does HARTS enter Linux?

Boot loader

Linux Kernel

- Harted initialization
- __cpu_up
- start_kernel
- secondary_start
- smp_callin
- continue to boot
- waits

hart0
hart1
hart2
hart3
Random booting - Issues

Every stage in multi-stage booting need to implement!!
Ordered booting - Advantages

Simple and follows standard for every case

- All harts are booted orderly.
- Currently first hart is chosen based lottery in OpenSBI only
- Other boot stages doesn’t have to worry about SMP
- Follows a standard way of bringing up harts in every case

CPU hotplug:
- All harts are returned to OpenSBI upon cpu shutdown
- Kernel won’t have to manage the state of the harts
- Resets all harts every time cpu is brought up online
- Power management becomes easier in future

Kexec:
- Kexec can easily return the harts to OpenSBI
- No need to keep “purgatory” code or memory management
Ordered booting (Proposed)

Only boots a single cpu all the way to Linux!

stage1

Boot loaders

- hsm_hart_start()
- continue to boot

Lottery

- wait
- After IPI received

All other harts waits for IPI

stage2

- hart0
- hart1
- hart2
- continue to boot

stage3

- hart3
- continue to boot

Directly jumps to Kernel

Linux Kernel

- start_secondary__cpus
- continue to boot

- __cpu_up

Kernel initialization...

- start_kernel
- continue to boot

- smp_callin

- secondary_start_sbi (head.S)
- continue to boot
SBI Hart State Management (HSM) extension

Required to achieve ordered booting

• Already available in SBI specification

• Functions
  – struct sbiret sbi_hart_start(unsigned long hartid, unsigned long start_addr, unsigned long priv)
  – struct sbiret sbi_hart_stop()
  – struct sbiret sbi_hart_status(unsigned long hartid)

• Implemented in OpenSBI and Linux kernel

• Patches are already in the mailing list
## Backward compatibility

<table>
<thead>
<tr>
<th>OpenSBI</th>
<th>Linux kernel</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>master</td>
<td>with HSM patches</td>
<td>Ordered booting</td>
</tr>
<tr>
<td>v0.6 or older</td>
<td>with HSM patches</td>
<td>Random booting</td>
</tr>
<tr>
<td>v0.6 or older</td>
<td>without HSM patches</td>
<td>Random booting</td>
</tr>
<tr>
<td>v0.5 or older</td>
<td>with HSM patches</td>
<td>Random booting</td>
</tr>
<tr>
<td>(equivalent: BBL)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>v0.5 or older</td>
<td>without HSM patches</td>
<td>Random booting</td>
</tr>
<tr>
<td>(equivalent: BBL)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>master</td>
<td>without HSM patches</td>
<td>NO SMP booting! (single hart boots only)</td>
</tr>
</tbody>
</table>
UEFI boot in RISC-V

• Grub support
  – RISC-V support available upstream
  – Grub loader patches in the mailing list
  – Thanks to Alex Graf and Chester Lin

• U-Boot
  – RISC-V UEFI support already added
  – “bootefi” can be used to load UEFI executables
  – Thanks To Alex Graf

• EDK2/Tianocore
  – Already in the staging area
  – Thanks to Abner Chang
  – “Open source UEFI and TianoCore” by Leif Lindholm

• EFI Stub
  – Fakes Linux kernel image as a EFI executable
  – Possible because PE/COFF header in the kernel Image header
  – Enabled via CONFIG_EFI_STUB=y
  – Boot time services are implemented in Kernel
  – Boots SMP with bootefi command in U-Boot
  – Patches will be available soon
  – Demo at the end of the talk!!
RISC-V: EBBR Compliant?

- A specification that allows interoperability between different hardware platforms
- Currently maintained by Linaro/ARM

Technical requirement:
- UEFI Boot Services
  - Memory Map, Configuration Tables, UEFI secure boot (optional)
- UEFI Runtime services
- Startup protocol
- Firmware storage

Political requirement:
- Hosted in “ARM-software” Github repo
- Current copyright “Copyright © 2017-2019 Arm Limited and Contributors.”
Future work
To infinity and beyond!

• Ratify the SBI specification
• Upstream SBI HSM extension and CPU hotplug code
• Upstream EFI stub code
• Add runtime services in EFI stub
• Unify ARM64 and RISC-V efi stub code (if possible)
• Make RISC-V EBBR compliant (if possible)
• Implement Kexec and Kdump