Hewlett Packard Enterprise

EDK2 UEFI on RISC-V

Open Source Firmware, BMC and Bootloader Devroom

Daniel Schaefer
February 6, FOSDEM 2021
Agenda

- About Us
- Introduction (EDK2, RISC-V)
- History of Booting on RISC-V
- Timeline of This Implementation
- Details of Bootflow
- Demo Booting to Linux
- Status, Goals, Vision
- How to Help
About us

– UEFI Firmware Engineers for ProLiant Servers at HPE
– Learned a lot through this project:
  Changes required in entire UEFI/EDK2 boot flow

Abner Chang

Daniel Schaefer

– Senior UEFI Engineer
– Lead the project

– Graduated last year
– First real UEFI project
Disclaimer

- Work was done on company time
- But we can’t speak of the strategic direction of HPE
UEFI and EDK2

- UEFI is only an interface specification
- EDK2 is the reference implementation of UEFI
- Initially developed for Itanium
- Now mainstream for x86_64
- Starting to get adopted on ARM
RISC-V

- "Free and Open RISC Instruction Set Architecture"
- Tries to be simple and legacy-free
- Three privilege modes (Machine, Supervisor, User)
- Similar to x86, boot starts without MMU in Machine mode
- FW can stay resident after boot and be called by higher layers
  Like x86’s SMM but with well defined interface like Itanium’s SAL
  → Supervisor Binary Interface (SBI)
## History of booting on RISC-V

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015</td>
<td>BBL</td>
</tr>
<tr>
<td>2016</td>
<td>EDK2 Prototype with QEMU RISC-V PC/AT board[^5]</td>
</tr>
<tr>
<td>2017</td>
<td>U-Boot</td>
</tr>
<tr>
<td>2018</td>
<td>U-Boot with UEFI interface</td>
</tr>
<tr>
<td>2018</td>
<td>Coreboot</td>
</tr>
<tr>
<td>2019</td>
<td>Oreboot</td>
</tr>
<tr>
<td>2020</td>
<td>EDK2+OpenSBI Upstream</td>
</tr>
</tbody>
</table>
Timeline of this implementation

2015    Started at HPE
2016    Prototype presented at UEFI Forum Plugfest
2020    Initial upstreaming to EDK2
        Booting to UEFI shell on Hifive Unleashed
2021    Upstream Linux EFISTUB support (WIP)
2021    Port more boards, e.g. BeagleV (WIP)
UEFI Phases

Architecture Execution Flow

- Pre Verifier
- CPU Init
- Chipset Init
- Board Init
- Device, Bus, or Service Driver
- EFI Driver Dispatcher
- Intrinsic Services
- Boot Manager
- OS-Absent App
- Transient OS Environment
- Transient OS Boot Loader
- Final OS Boot Loader
- OS-Present App
- Final OS Environment
- Security (SEC)
- Pre EFI Initialization (PEI)
- Driver Execution Environment (DXE)
- Boot Dev Select (BDS)
- Transient System Load (TSL)
- Run Time (RT)
- After Life (AL)

Power on ➔ [... Platform initialization ...] ➔ [... OS boot ...] ➔ Shutdown

Hewlett Packard Enterprise
Reset at ZSBL

Walkthrough of Hifive Unleashed Bootflow

- Starts running in M-Mode without MMU
- ZSBL (zero stage bootloader)
- Embedded in mask ROM or hardcoded in QEMU source
- Jumps to predefined address
SEC - First Phase of UEFI (ASM)

- Fully custom for RISC-V
- Starts in assembly

1. Set up scratch register (mscratch)
2. Set up stack in temporary RAM
3. Set up trap handler to preserve registers and call OpenSBI
4. Calls SEC core C function with hartid and scratch pointer
SEC - First Phase of UEFI (C)

1. Add UEFI private region in scratch space
   – Machine information (march, mimpid, ...)

2. Initialize OpenSBI (sbi_init)
   – Stall non-booting harts
SEC - Initialize OpenSBI

1. Pass scratch pointer with 
   - Device Tree 
   - Next mode (M-Mode still) 
   - Platform specific functions

2. Register custom SBI calls (avoid linking PEI to OpenSBI)
SEC - Switch to PEI

1. Find PEI entrypoint in firmware volume
2. Switch to S-Mode
3. Enable identity mapped MMU
4. Jump to PEI and pass it information about
   - Boot firmware volume
   - Temporary ram
   - Stack
From here on most of the code is arch-agnostic in EDK2.

1. Discover RAM and migrate there
2. Dispatch PEIMs
   - Take device tree from scratch space and store in HOB
   - Discover processor features and store in HOB (For SMBIOS)
   - Others ...
3. Build new stack
4. Switch stack and execute DxeIpl
DXE - Dispatch DXEs

- Install timer interrupt handler and DXE Protocol
- Install RuntimeServices (WIP)
- Install SMBIOS tables using information from PEI
  - Type 4 (CPU)
  - Type 7 (Caches)
  - **Type 44** (Additional CPU information)
- Install device tree
  - Extract from HOB
  - Insert boot hartid (required by Linux)
  - Insert into EFI System Configuration Table
BDS - UEFI Shell

Not upstream yet.
Prep: Embed EFISTUB and initrd disk image in flash image

1. Load disk into memory and turn into ramdisk with shell command
2. Install initrd on handle of fixed device path (‘initrd’) 
3. Execute EFISTUB
EFISTUB - Since Linux 5.10

Implemented by Atish Patra, tested and finalized in cooperation with us

1. Take device tree from EFI System Configuration Table
2. Call LoadFile2 on device path to extract initrd
3. Execute kernel proper
   - Disable MMU
   - jump_kernel(hartid, fdt);
Demo Booting to Linux

https://asciinema.org/a/KPDSvhXNVTbsQ45oRUVEu81nY

Firmware image at https://github.com/riscv/riscv-uefi-edk2-docs/releases
Status - EDK2

- UEFI Shell
- UEFI Applications (e.g. bootloader)
- Booting Linux via EFISTUB
## Status - Platforms

<table>
<thead>
<tr>
<th>Platform Name</th>
<th>UEFI Shell</th>
<th>Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>HiFive Unleashed</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>QEMU sifive_u</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Freedom U500 FPGA</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>QEMU virt</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Andes AE350</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>BeagleV</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Status - Overall

- UEFI specification amended
- SMBIOS specification amended
- EDK2 port merged upstream
- Linux EFISTUB ported by Atish, merged in 5.10
- UEFI Self Certification Test ported, patches need cleanup
Goals

- Implement *ResetSystem* Runtime Service with SBI (WIP)[1]
- Upstream changes for booting Linux (FDT fixup and storing)
- SD card driver for Hifive Unleashed
- Implement new relocation types by newer GNU toolchains[2]
- Build "OVMF" for QEMU’s virt platform with VirtIO drivers[3]
  → Add boot tests to EDK2 CI
  → Boot with actual disk!
- Port to BeagleV when it arrives[4]
- SecureBoot?
How to Help

- Port to more boards, talk to us
- Check out the issues on the repo
- Spread the word
Vision for the Future

- Make RISC-V boot like rest of industry
  U-boot for embedded, UEFI for consumer and server
- Follow in ARM’s footsteps to *make booting boring*
- Encourage discussion about desktops and servers
Thanks

Thanks for listening!

Check us out on GitHub:
Development          Upstream
riscv/riscv-uefi-edk2-docs
riscv/riscv-edk2      tianocore/edk2
riscv/edk2-platforms  tianocore/edk2-platforms

Daniel Schaefer      <daniel.schaefer@hpe.com>
Abner Chang          <abner.chang@hpe.com>
References

UEFI and PI Spec  https://www.uefi.org/specifications
SBI Spec  https://github.com/riscv/riscv-sbi-doc


BBL  https://www.lowrisc.org/docs/build-berkeley-boot-loader/
OpenSBI  https://github.com/riscv/opensbi
U-Boot  https://www.denx.de/wiki/U-Boot
Coreboot  https://coreboot.org/
Oreboot  https://github.com/oreboot/oreboot
QEMU  https://www.qemu.org/
Itanium SAL/PAL  https://www.csee.umbc.edu/portal/help/architecture/24535901.pdf
### Glossary

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>UEFI</td>
<td>Unified Extensible Firmware Interface</td>
</tr>
<tr>
<td>EDK2</td>
<td>UEFI's reference implementation</td>
</tr>
<tr>
<td>Tianocore</td>
<td>Umbrella name of EDK2 and related projects</td>
</tr>
<tr>
<td>MMU</td>
<td>Memory Management Unit</td>
</tr>
<tr>
<td>SBI</td>
<td>RISC-V interface between S and M-mode</td>
</tr>
<tr>
<td>Itanium</td>
<td>First 64 bit processor by Intel</td>
</tr>
<tr>
<td>SAL</td>
<td>Itanium’s System Abstraction Layer</td>
</tr>
<tr>
<td>RISC</td>
<td>Reduced Instruction Set Computer (vs CISC)</td>
</tr>
<tr>
<td>SMM</td>
<td>System Management Mode</td>
</tr>
<tr>
<td>HPE</td>
<td>Hewlett Packard Enterprise</td>
</tr>
</tbody>
</table>