Western Digital®

An Introduction to RISC-V Boot Flow

Atish Patra <Atish.Patra@wdc.com>
Anup Patel <Anup.Patel@wdc.com>

Outline

- Common embedded boot flow
- Current RISC-V boot flow
- OpenSBI Project
- Tutorials
- Current Status
- Future work
- Tutorials

Getting started..

- Create a working directory

```
mkdir summit_demo; cd summit_demo
```

- Download cross-compiling toolchain
  - https://toolchains.bootlin.com/releases_riscv64.html
- Download pre-built images
  - https://wdc.box.com/s/ihywc2xap5m4mflyngjtndf0sy62zha3
- Clone OpenSBI

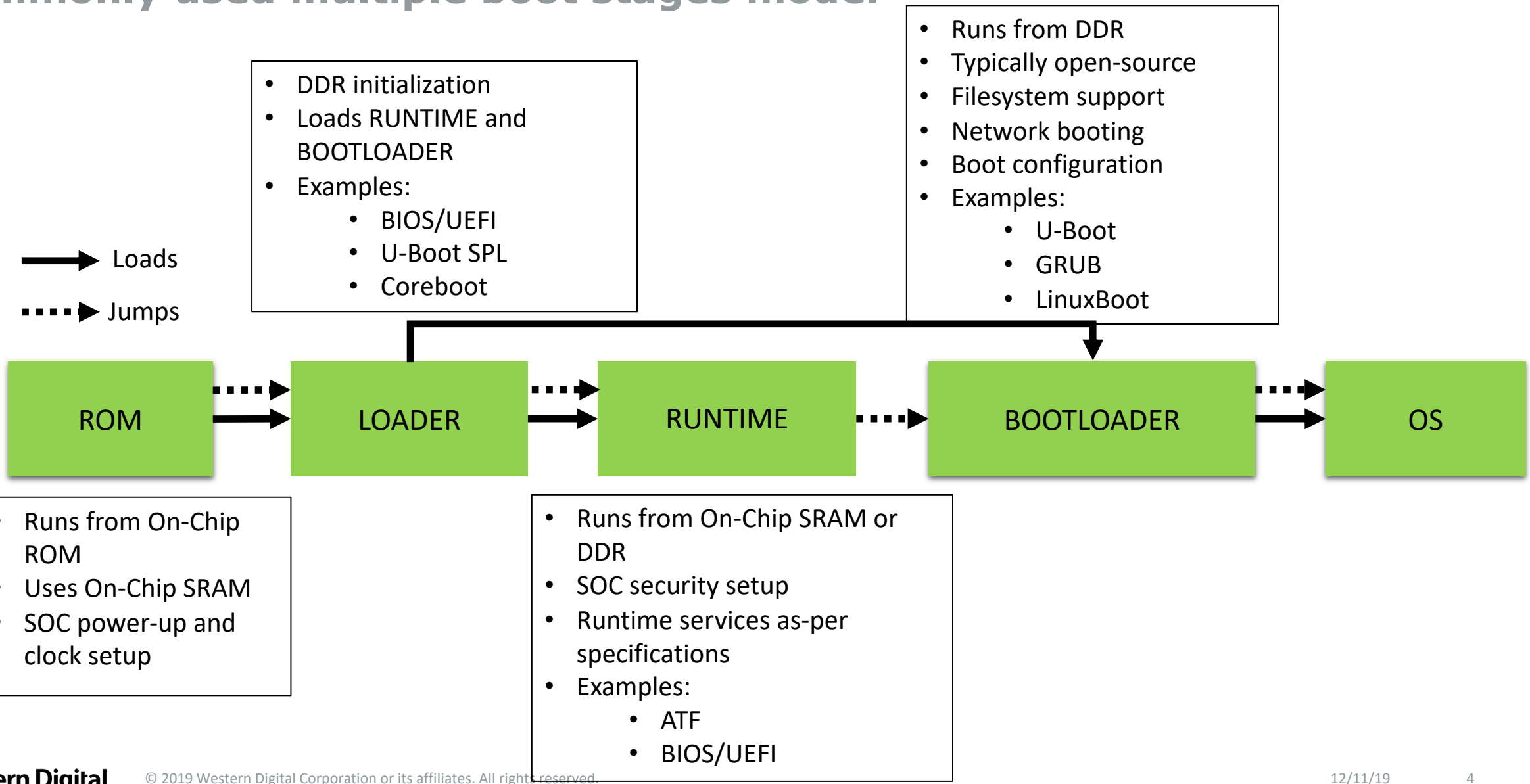
```
git clone https://github.com/riscv/opensbi.git
```

- Clone U-Boot

```
git clone https://github.com/u-boot/u-boot.git ; git checkout v2019.10
```

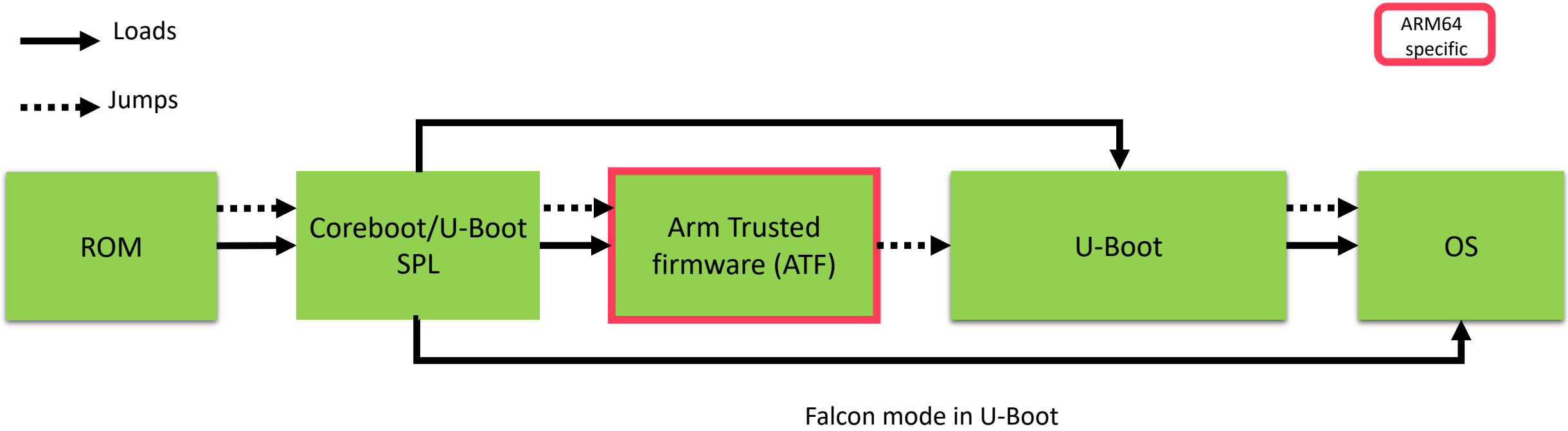
Common boot flow

Commonly used multiple boot stages model



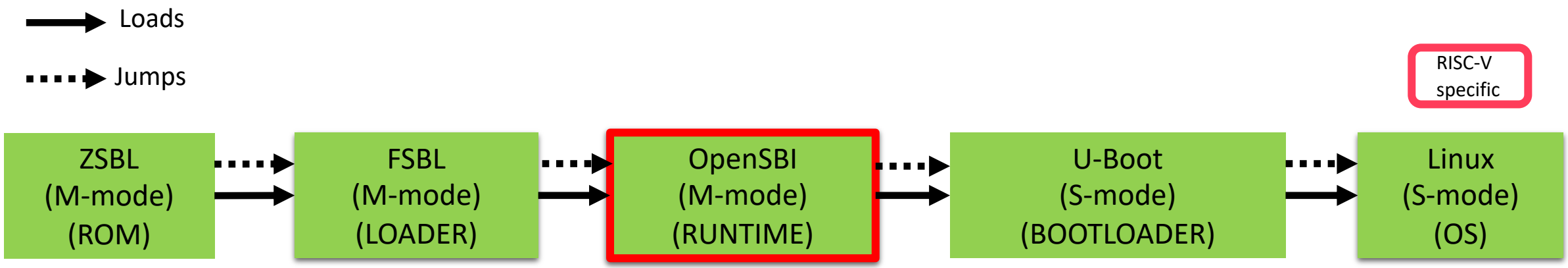
Common boot flow in ARM64

Commonly used multiple boot stages model



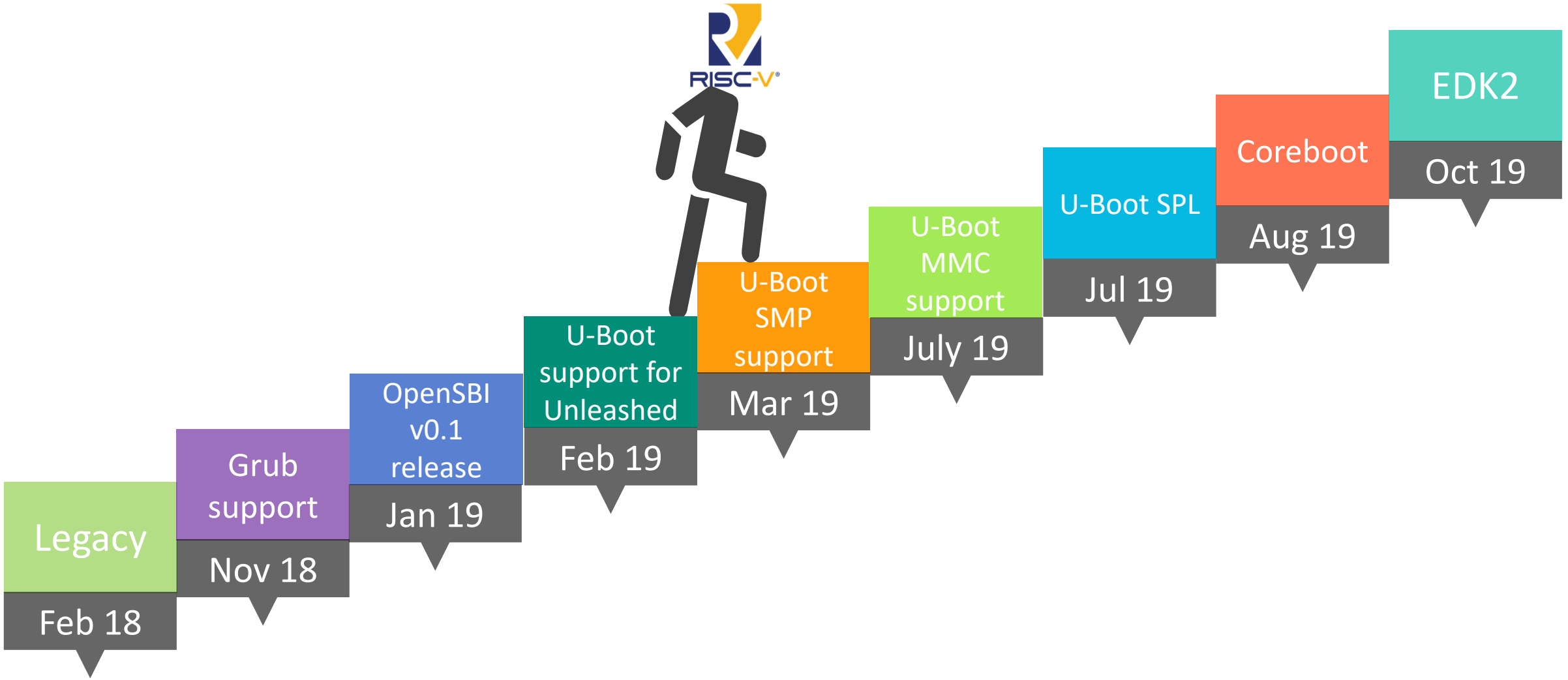
RISC-V Upstream Boot Flow

Follows commonly used multiple boot stages model



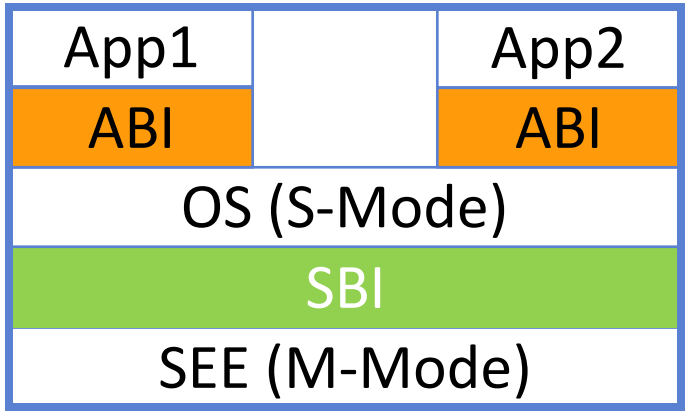
- For HiFive Unleashed hardware (only Linux capable RISC-V platform available)
- Follows a standard boot flow
- Uses U-Boot as the last stage boot loader
- U-Boot binary as the payload to OpenSBI
- FSBL is SiFive specific and will be replaced by Coreboot/U-Boot SPL
- OpenSBI is a RISC-V specific runtime service provider

RISC-V boot flow development timeline



What is SBI ?

- SBI is the RISC-V Supervisor Binary Interface
  - System call style calling convention between Supervisor (S-mode OS) and Supervisor Execution Environment (SEE)
- SEE can be:
  - A M-mode RUNTIME firmware for OS/Hypervisor running in HS-mode
  - A HS-mode Hypervisor for Guest OS running in VS-mode
- SBI calls help:
  - Reduce duplicate platform code across OSes (Linux, FreeBSD, etc)
  - Provide common drivers for an OS which can be shared by multiple platforms
  - Provide an interface for direct access to hardware resources (M-mode only resources)
- Specifications being drafted by the Unix Platform Specification Working group
  - Currently, SBI v0.1 in-use and SBI v0.2 in draft stage
  - https://github.com/riscv/riscv-sbi-doc

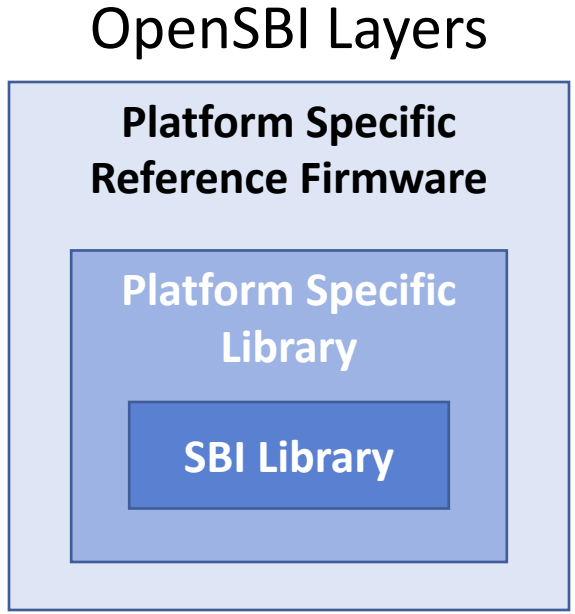


What is OpenSBI ?

- OpenSBI is an open-source implementation of the RISC-V Supervisor Binary Interface (SBI) specifications
  - Licensed under the terms of the BSD-2 clause license
  - Helps to avoid SBI implementation fragmentation
  - Maintained by the community
- Aimed at providing RUNTIME services in M-mode
  - Typically used in boot stage following ROM/LOADER
- Provides support for reference platforms
  - Generic simple drivers included for M-mode to operate
    - PLIC, CLINT, UART 8250
  - Other platforms can reuse the common code and add needed drivers
- Source
  - https://github.com/riscv/opensbi

Key Features

- Layered structure to accommodate various use cases
  - Generic SBI library with platform abstraction
    - Typically used with external firmware and bootloader
      - EDK2 (UEFI implementation), Secure boot working group
  - Platform specific library
    - Similar to core library but including platform specific drivers
  - Platform specific reference firmware
    - Three different types of RUNTIME firmware
- Wide range of hardware features supported
  - RV32 and RV64
  - Hypervisor support
  - Misaligned load/store handling
  - Missing CSR emulation
  - Protects firmware using PMP support



Platform support

- SiFive HiFive Unleashed
- Andes AE350
- Ariane FPGA SOC
- Kendryte K210
- QEMU virt machines (32-bit/64-bit)
- OmniXtend

Tutorial

Pre-requisites

- Working setup

```
atish@jedi-01:/scratch2/summit_demo$ ls -l
total 146484
-rwxrwxr-x  1 atish atish  9127676 Dec  4 23:38 linux_Image
-rw-r--r--  1 atish atish 62914560 Dec  5 11:16 linux_rootfs.img
drwxrwxr-x 10 atish atish     4096 Dec  4 23:35 opensbi
-rwxrwxr-x  1 atish atish 47665320 Dec  4 23:39 qemu-system-riscv64
drwxr-xr-x  9 atish atish     4096 Nov 25  2018 riscv64--glibc--bleeding-edge-2018.11-1
-rw-rw-r--  1 atish atish 63313585 Nov 25  2018 riscv64--glibc--bleeding-edge-2018.11-1.tar.bz2
drwxrwxr-x 26 atish atish     4096 Dec  4 23:49 u-boot
```

- Extract toolchain and add it to the environment path

```
tar –xvf riscv64--glibc--bleeding-edge-2018.11-1.tar.bz2
export PATH=$PATH:riscv64--glibc--bleeding-edge-2018.11-1/bin/
```

- Set environment variable ARCH

```
export ARCH=riscv
```

- Set environment variable CROSS_COMPILE

```
export CROSS_COMPILE=riscv64-linux-
```

Tutorial-I

Boot Linux in Qemu as a payload to OpenSBI

- Compile OpenSBI

```
cd opensbi; make PLATFORM=qemu/virt FW_PAYLOAD_PATH=../linux_Image; cd ..
```

Virt machine in Qemu

Linux kernel image

- Run it in Qemu

```
./qemu-system-riscv64 -M virt -m 256M -nographic \
-kernel opensbi/build/platform/qemu/virt/firmware/fw_payload.elf \
-drive file=linux_rootfs.img,format=raw,id=hd0 \
-device virtio-blk-device,drive=hd0 \
-append "root=/dev/vda rw console=ttyS0"
```

OpenSBI image

Linux rootfs

Adding Support for New Platforms

- To add support for a new *<xyz>* platform
  1. Create directory named *<xyz>* under */platform* directory
  2. Create platform configuration file *<xyz>/config.mk*
     - *config.mk* will provide compiler flags, select common drivers, and select firmware options
     - *platform/template/config.mk* can be used as reference for creating *config.mk*
  3. Create platform objects file *<xyz>/objects.mk* for listing platform-specific objects to be compiled
     - *platform/template/objects.mk* can be used as reference for creating *objects.mk*
  4. Create platform source file *<xyz>/platform.c* providing "*struct sbi_platform*" instance
     - *platform/template/platform.c* can be used as reference for creating *platform.c*
- The *<xyz>* platform support directory can also placed outside OpenSBI sources

Reference Firmwares

- OpenSBI provides several types of reference firmware, all platform-specific
  - **FW_PAYLOAD**
    - Firmware with the next booting stage as a payload
    - Default firmware being used in Linux capable RISC-V hardware
  - **FW_JUMP**
    - Firmware with fixed jump address to the next booting stage
    - Default method for QEMU
  - **FW_DYNAMIC**
    - Firmware with dynamic information on the next booting stage
    - U-Boot SPL/Coreboot is using FW_DYNAMIC
- SOC Vendors may choose:
  - Use one of OpenSBI reference firmwares as their M-mode RUNTIME firmware
  - Build M-mode RUNTIME firmware from scratch with OpenSBI as library
  - Extend existing M-mode firmwares (U-Boot_M_mode/EDK2) with OpenSBI as library

U-Boot

An universal boot loader

- Most commonly used in embedded systems
- Used as a last stage boot loader
- Support
  - many ISAs (x86, ARM, AARCH64, RISC-V, ARC..)
  - Peripherals (UART, SPI, I2C, ethernet, SD, USB..)
  - Multiple file systems
  - Various network protocols
- Can load images from network, file system, removable devices
- Easy command line interface for management
- Lot of customization
  - U-Boot SPL (a redacted version of U-Boot used as first stage boot loader)
  - Falcon mode (for fast booting)

Tutorial - II

Boot Linux in Qemu using U-Boot proper

- Compile U-Boot

```
cd u-boot; make qemu-riscv64_smode_defconfig; make; cd  ..
```

- Compile OpenSBI

```
cd opensbi; make PLATFORM=qemu/virt; cd ..
```

No payload!!

virt machine in Qemu

- Run it in Qemu

```
./qemu-system-riscv64 -M virt -smp 4 -m 256M -nographic \
-bios opensbi/build/platform/qemu/virt/firmware/fw_jump.elf \
-kernel u-boot/u-boot.bin -device loader,file=linux_Image,addr=0x84000000 \
-drive file=linux_rootfs.img,format=raw,id=hd0 \
-device virtio-blk-device,drive=hd0
```

OpenSBI firmware (fw_jump)

No payload!!

Kernel address expected by U-boot

Tutorial - II

Boot Linux in Qemu using U-Boot proper

- At U-Boot prompt

```
OpenSBI v0.5-13-g813f7f4c250a
   ____                    _____ ____ _____
  / __ \                  / ____|  _ \_   _|
 | |  | |_ __   ___ _ __ | (___ | |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | |_) |  __/ | | |____) | |_) || |_
  \____/| .__/ \___|_| |_|_____/|____/_____|
        | |
        |_|

Platform Name          : QEMU Virt Machine
Platform HART Features : RV64ACDFIMSU
Platform Max HARTs     : 8
Current Hart           : 2
Firmware Base          : 0x80000000
Firmware Size          : 116 KB
Runtime SBI Version    : 0.2

PMP0: 0x0000000080000000-0x000000008001ffff (A)
PMP1: 0x0000000000000000-0xffffffffffffffff (A,R,W,X)


U-Boot 2020.01-rc4-00066-g7e5ee346fc4c (Dec 04 2019 - 23:48:42 -0800)

CPU:   rv64imafdcsu
Model: riscv-virtio,qemu
DRAM:  256 MiB
In:    uart@10000000
Out:   uart@10000000
Err:   uart@10000000
Net:   No ethernet found.
Hit any key to stop autoboot:  0
=>
```

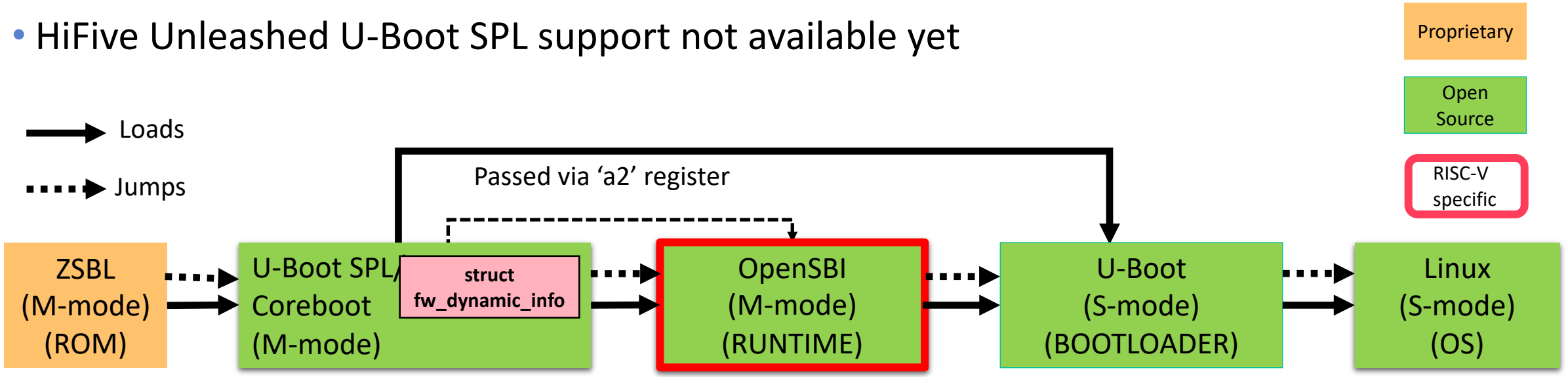
```
=> setenv bootargs "root=/dev/vda2 rw console=ttyS0 earlycon=sbi"
=> setenv bootargs "root=/dev/vda rw console=ttyS0 earlycon=sbi"
=> cp.l ${fdtcontroladdr} ${fdt_addr_r} 0x10000
=> booti ${kernel_addr_r} - ${fdt_addr_r}
```

=>setenv bootargs "root=/dev/vda rw console=ttyS0 earlycon=sbi"
=>cp.l ${fdtcontroladdr} ${fdt_addr_r} 0x10000
=>booti ${kernel_addr_r} - ${fdt_addr_r}

```
[    0.000000] printk: bootconsole [sbi0] enabled
[    0.000000] initrd not found or empty - disabling initrd
[    0.000000] Zone ranges:
[    0.000000]   DMA32    [mem 0x0000000080200000-0x00000000ffffffff]
[    0.000000]   Normal   empty
[    0.000000] Movable zone start for each node
[    0.000000] Early memory node ranges
[    0.000000]   node   0: [mem 0x0000000080200000-0x00000000ffffffff]
[    0.000000] Zeroed struct page in unavailable ranges: 512 pages
[    0.000000] Initmem setup node 0 [mem 0x0000000080200000-0x00000000ffffffff]
[    0.000000] software IO TLB: mapped [mem 0x8abfa000-0x8ebfa000] (64MB)
[    0.000000] elf_hwcap is 0x112d
[    0.000000] percpu: Embedded 17 pages/cpu s30680 r8192 d30760 u69632
[    0.000000] Built 1 zonelists, mobility grouping on.  Total pages: 64135
[    0.000000] Kernel command line: root=/dev/vda rw console=ttyS0 earlycon=sbi
[    0.000000] Dentry cache hash table entries: 32768 (order: 6, 262144 bytes, linear)
[    0.000000] Inode-cache hash table entries: 16384 (order: 5, 131072 bytes, linear)
[    0.000000] Sorting __ex_table...
[    0.000000] mem auto-init: stack:off, heap alloc:off, heap free:off
[    0.000000] Memory: 164580K/260096K available (6023K kernel code, 391K rwdata, 1955K r
[    0.000000] SLUB: HWalign=64, Order=0-3, MinObjects=0, CPUs=4, Nodes=1
[    0.000000] rcu: Hierarchical RCU implementation.
[    0.000000] rcu:     RCU restricting CPUs from NR_CPUS=32 to nr_cpu_ids=4.
[    0.000000] rcu: RCU calculated value of scheduler-enlistment delay is 25 jiffies.
```

Boot flow using OpenSBI dynamic firmware

- OpenSBI firmware with dynamic information about the next booting stage
- Coreboot support available on both hardware & QEMU
- U-Boot SPL available for QEMU and Andes AE350
- HiFive Unleashed U-Boot SPL support not available yet



Tutorial - III

Boot Linux in Qemu using U-Boot SPL

- Compile OpenSBI

```
cd opensbi; make PLATFORM=qemu/virt; cd ..
```

No payload!!

- Compile U-Boot SPL

```
cd u-boot; export OPENSBI=../opensbi/build/platform/qemu/virt/firmware/fw_dynamic.bin;
ARCH=riscv CROSS_COMPILE=riscv64-linux- make qemu-riscv64_spl_defconfig
ARCH=riscv CROSS_COMPILE=riscv64-linux- make; cd ..
```

Path to OpenSBI dynamic firmware

- Run it in Qemu

```
./qemu-system-riscv64 -nographic -machine virt -m 2G -bios u-boot/spl/u-boot-spl \
-kernel u-boot/u-boot.itb -device loader,file=linux_Image,addr=0x8600000
-drive file=linux_rootfs.img,format=raw,id=hd0 \
-device virtio-blk-device,drive=hd0
```

U-Boot SPL binary

FIT image (OpenSBI + U-Boot proper

Tutorial - III

Boot Linux in Qemu using U-Boot proper

- At U-Boot prompt

```
U-Boot SPL 2020.01-rc4-00066-g7e5ee346fc4c (Dec 05 2019 - 15:55:07 -0800)
Trying to boot from RAM


U-Boot 2020.01-rc4-00066-g7e5ee346fc4c (Dec 05 2019 - 15:55:07 -0800)

CPU:   rv64imafdcsu
Model: riscv-virtio,qemu
DRAM:  128 MiB
In:    uart@10000000
Out:   uart@10000000
Err:   uart@10000000
Net:   No ethernet found.
Hit any key to stop autoboot:  0

Device 0: QEMU VirtIO Block Device
           Type: Hard Disk
           Capacity: 60.0 MB = 0.0 GB (122880 x 512)
... is now current device
** No partition table - virtio 0 **
No ethernet found.
No ethernet found.
=>
```

```
=>cp.b 0x86000000 ${kernel_addr_r} 0x1000000
=>setenv bootargs "root=/dev/vda rw console=ttyS0 earlycon=sbi"
=>cp.l ${fdtcontroladdr} ${fdt_addr_r} 0x10000
=>booti ${kernel_addr_r} - ${fdt_addr_r}
```

```
=> cp.b 0x86000000 ${kernel_addr_r} 0x1000000
=> setenv bootargs "root=/dev/vda rw console=ttyS0 earlycon=sbi"
=> cp.l ${fdtcontroladdr} ${fdt_addr_r} 0x10000
=> booti ${kernel_addr_r} - ${fdt_addr_r}
## Flattened Device Tree blob at 88000000
   Booting using the fdt blob at 0x88000000
   Using Device Tree in place at 0000000088000000, end 0000000088006bb7

[    0.000000] Movable zone start for each node
[    0.000000] Early memory node ranges
[    0.000000]   node   0: [mem 0x0000000080200000-0x00000000bfffffff]
[    0.000000] Zeroed struct page in unavailable ranges: 512 pages
[    0.000000] Initmem setup node 0 [mem 0x0000000080200000-0x00000000bfffffff]
[    0.000000] software IO TLB: mapped [mem 0xb9ff5000-0xbdff5000] (64MB)
[    0.000000] elf_hwcap is 0x112d
[    0.000000] percpu: Embedded 17 pages/cpu s30680 r8192 d30760 u69632
[    0.000000] Built 1 zonelists, mobility grouping on.  Total pages: 258055
[    0.000000] Kernel command line: root=/dev/vda rw console=ttyS0 earlycon=sbi
```

Boot flow using OpenSBI as a library

- OpenSBI as a part of external firmware source code
- Must configure program stack and scratch space for OpenSBI
- Same GCC target option for external firmware and OpenSBI
- Currently EDK2 integration with OpenSBI
  - HPE leading this effort
  - Available in EDK2 mailing list for U540 on Xilinx VC707 FPGA
  - OpenSBI built with EDK2 build environment
  - OpenSBI used as library in Pre-EFI Initialization (PEI) phase

Constraints on using OpenSBI as a 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:
  - Program Stack
  - OpenSBI scratch space (i.e. *struct sbi_scratch* instance with extra space above)
- Two constraints in calling any OpenSBI functions from external firmware:
  - MSCRATCH CSR of calling HART must be set to its own OpenSBI scratch space
  - 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

Current status

Rapid progress: traditional full boot support expected by year end

- OpenSBI
  - Actively developed and maintained
  - Version 0.5 released
  - Default in Buildroot, Yocto/OpenEmbedded and the QEMU “BIOS”
  - Fedora/Debian provides images available with OpenSBI binary
- U-Boot
  - U-Boot-2019.10 release has full support HiFive Unleashed S-mode
  - Boot via network/MMC supported
  - FIT image support
  - EFI support for RISC-V available
- Grub
  - RISC-V support available upstream
- Linux Kernel
  - Upstream kernel boots in QEMU
  - Device tree hosted in Kernel
  - v5.3 kernel works with OpenSBI+U-Boot on HiFive Unleashed

Future boot support

Toward a stable and easy to use boot ecosystem

- EFI stub support in Linux kernel for full UEFI support in progress
  - Enterprise class boot environment
- U-Boot SPL support for HiFive Unleashed
- SMP support in coreboot
- EDK2 project upstreaming (in progress)
- Oreboot (Coreboot written in Rust)
- LinuxBoot ?
- Other bootloaders ?

Ongoing work

Toward a stable and easy to use boot ecosystem

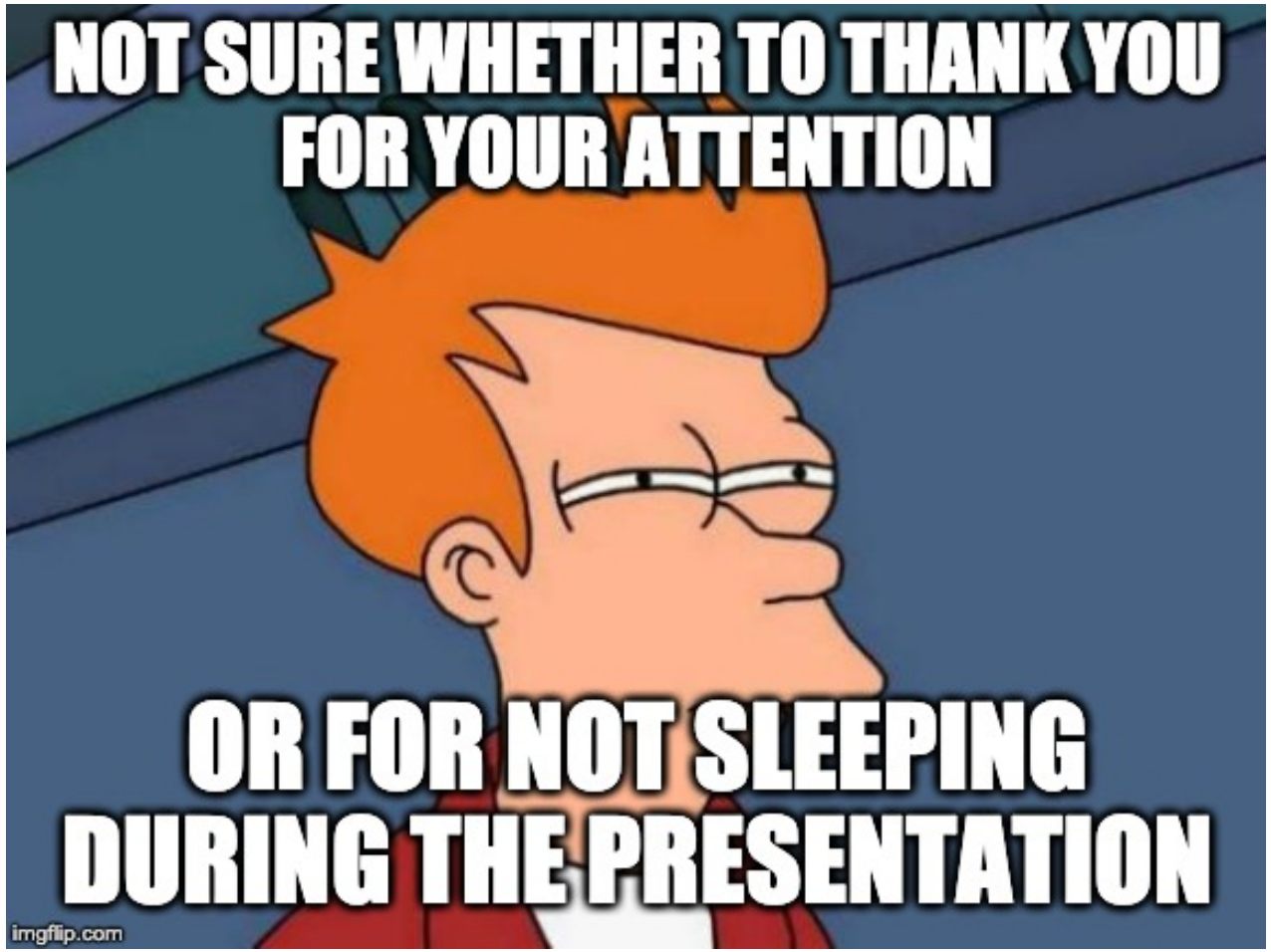
- SBI specifications
  - SBI v0.2 specification
  - Hart state management extension in SBI
  - Legacy SBI extension replacement in SBI
- OpenSBI
  - Sequential cpu boot via CPU hotplug
  - Support other M-mode boot loader such as Coreboot/U-Boot SPL
  - Hypervisor support when specification changes
  - More platforms support
    - Need hardware !
- Linux kernel
  - SBI v0.2 implementation (patches are under review)
  - EFI stub in Linux kernel (work in progress)
  - SBI legacy extension replacements
  - Sequential booting
  - CPU hotplug

Acknowledgements

- U-Boot
  - Lukas Auler
  - Bin Meng
  - Anup Patel
- Coreboot
  - Ron Minnich
  - Jonathan Neuschäfer
  - Patrick Rudolph
  - Philip Hug
- EDK2
  - Abner Chang
- And others

Thank you!!

- Q&A ?

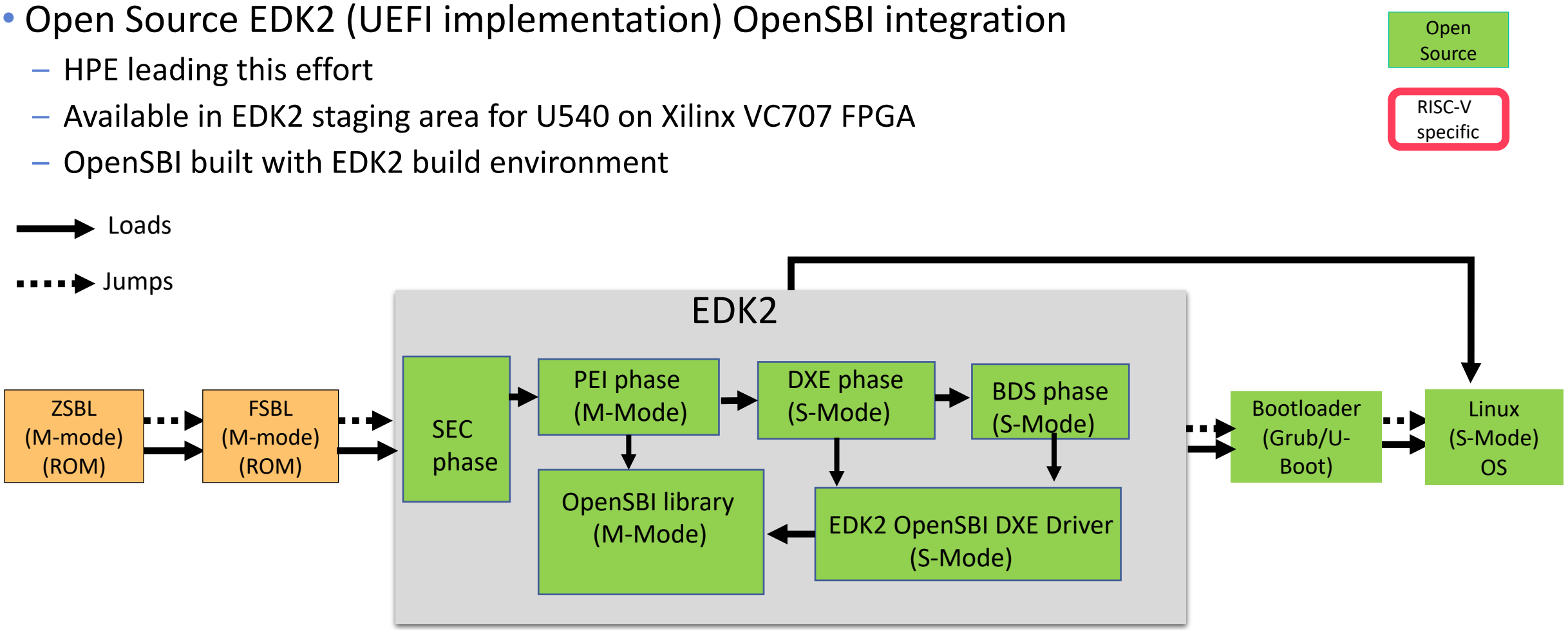


Western Digital®

Backup

EDK2 implementation details

- OpenSBI used as library
- Open Source EDK2 (UEFI implementation) OpenSBI integration
  - HPE leading this effort
  - Available in EDK2 staging area for U540 on Xilinx VC707 FPGA
  - OpenSBI built with EDK2 build environment



Reference

- 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