The Standardized Boot flow for RISC-V Platforms

- Jagan Teki, Amarula Solutions(India)

CRVA Technical Seminar (CRVS 2020)

Jagan Teki

- Co-Founder, CEO at **Amarula Solutions India**
  - Business development and customer handling.
  - Amarula Open Source portfolio creation, include RISC-V development.

- Embedded Linux Architect at **Amarula Solutions India**
  - *Bootloader*: BootROM, Bootloaders, U-Boot, Chip/Board bring ups, Devicetrees, Device drivers.
  - *Embedded Linux*: Linux BSP, Devicetrees, Device drivers, Multimedia, Optimizations, Integrations and etc.

- Mainline contributions
  - **Linux**
    - Contributor of Allwinner, Rockchip, i.MX platforms, BSP, device drivers.
    - Maintainer of few **DSI** LCD panels.
  - **U-Boot**
    - Contributor of Xilinx Zynq, Allwinner, Rockchip, i.MX platforms, Device drivers.
    - Maintainer of Allwinner **sunXi** SoCs.
    - Maintainer of **SPI/SPI-NOR** Subsystems.
  - Contributor of **Buildroot**, **Yocto.**

- Speaker at international Embedded, Open Source, and RISC-V conferences held on worldwide.

Agenda

Boot flow
- ➔ An Introduction to RISC-V Boot flow
- ➔ Processor modes
- ➔ OpenSBI

Standardized Boot flow
- ➔ Why Standard matter
- ➔ U-Boot Mainline

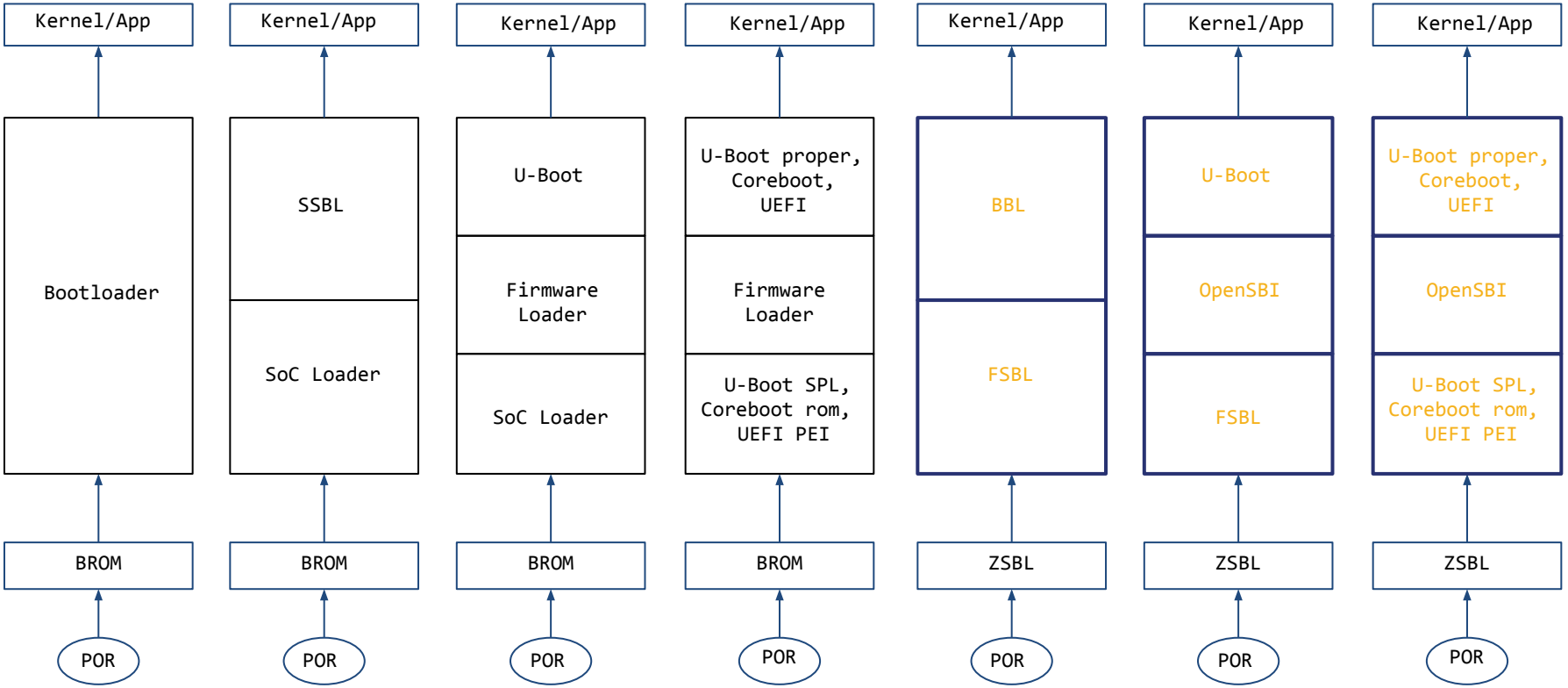
Summary
- ➔ Other projects status
- ➔ Booting from RAM
- ➔ Booting from SD
- ➔ Booting from SPI
- ➔ EFI Booting

Note: All Demos are run on SiFive HiFive-Unleashed A00 and Andes AX25-AE350 boards.

Boot flow

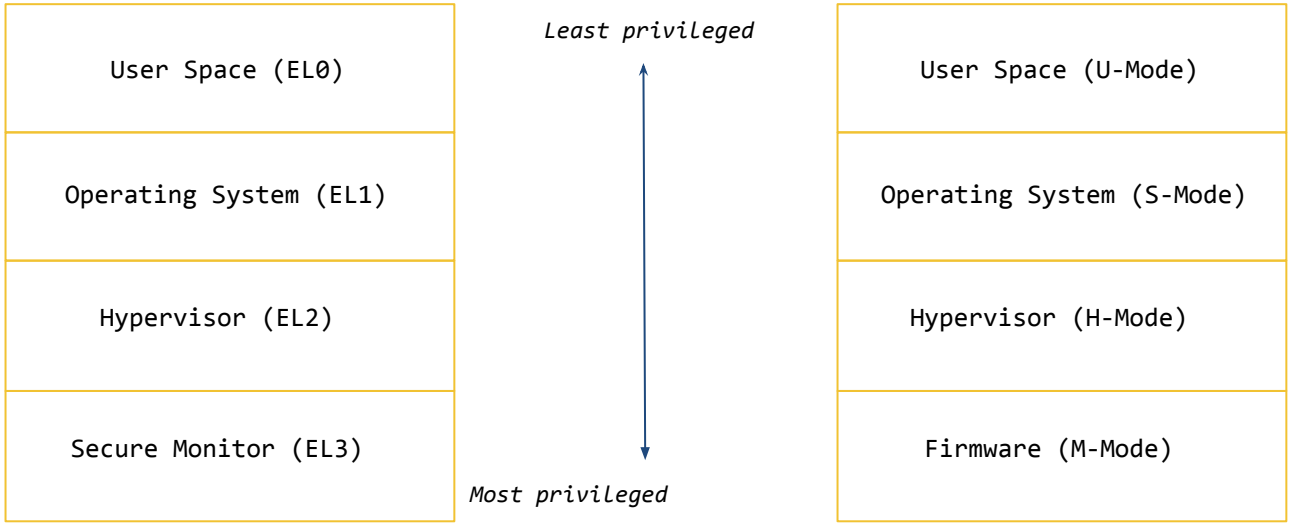
- ➔ An Introduction RISC-V Boot flow
- ➔ Processor modes
- ➔ OpenSBI

An Introduction to RISC-V Boot flow



BROM: BootROM **SoC Loader:** MLO, FSBL **Firmware Loader:** TF-A, OpenSBI **SSBL:** Second stage bootloader U-Boot, Coreboot, UEFI
ZSBL: Zero Stage Bootloader **FSBL:** First Stage Bootloader **OpenSBI:** RISC-V Open Source Supervisory Binary Interface **BBL:** Berkeley Bootloader

RISC Processor modes



ARM64 Exception Levels

- ➔ EL3 has platform specific runtime firmware.
- ➔ EL3 has secure privileges.
- ➔ ARM64 start from EL3, means in secure world.
- ➔ Bootloaders(non-secure) uses **ARM Trusted firmware (TF-A)** switch normal world EL2 since system boot from secure EL3.

RISC-V Privilege Modes

- ➔ M-Mode has platform specific runtime firmware(only).
- ➔ M-Mode does have secure privileges.
- ➔ RISC-V start from M-Mode, A bare metal machine mode.
- ➔ Bootloaders uses **OpenSBI** switch into S-Mode from M-Mode for non-hypervisor world.

Note: Comparing processor modes here is for the sake of understanding but the actual modes of operations are purely platform specific.

OpenSBI

SBI

- ➔ RISC-V Supervisor Binary Interface
- ➔ System call type interface layer between Firmware runtime, M-Mode to Operating system, S-Mode.
- ➔ Avoid fragmentation of various OEM silicon providers specific runtime firmware implementations.
- ➔ Standard, generic runtime firmware interface specification across all OSes, different cpu and silicon platforms.
- ➔ Specification in SBI v0.2 in usage (v0.2 in draft)

OpenSBI

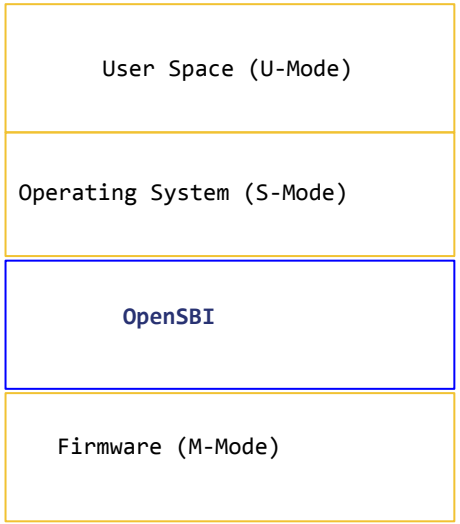
- ➔ RISC-V Open Source Supervisory Binary Interface
- ➔ An Open Source implementation of SBI specification, BSD-2 license
- ➔ Modular, Scalable and Extendable between all CPU and Silicon specific hardware configurations.
- ➔ Contains platform-independent and platform-dependent libraries like libsbi.a, libplatsbi.a
- ➔ Platforms supports like SiFive U540, Andes AE350, Ariane FPGA, Kendryte K210, Nuclei UX600, Openpiton FPGA, T-head C910, QEMU.

FW_PAYLOAD

- ➔ Pack the firmware with next level boot stage as payload, fw_payload.bin
- ➔ Can be packable with U-Boot, Kenel

FW_DYNAMIC

- ➔ Pack the firmware with runtime accessible to the next level boot stage, fw_dynamic.bin
- ➔ Can be packable in U-Boot SPL, Coreboot



RISC-V Privilege Modes, non-hyp

Source: *SBI; https://github.com/riscv/riscv-sbi-doc OpenSBI; https://github.com/riscv/opensbi*

Standardized Boot flow

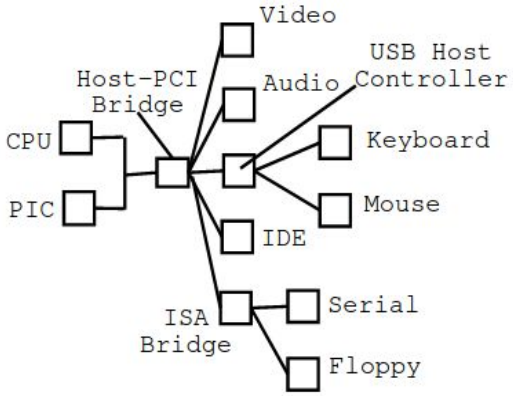
- ➔ Why Standard matter
- ➔ U-Boot Mainline status

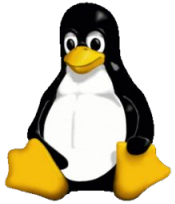
Why Standard matter, Hardware

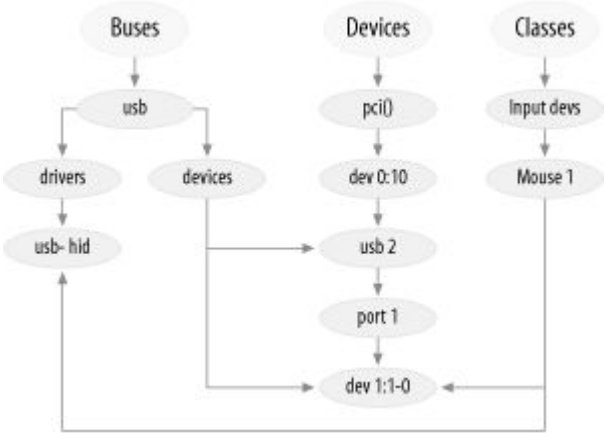
Apple 1
Presented as technology

Apple 2
Presented as product

Why Standard matter, Kernel (Device model)



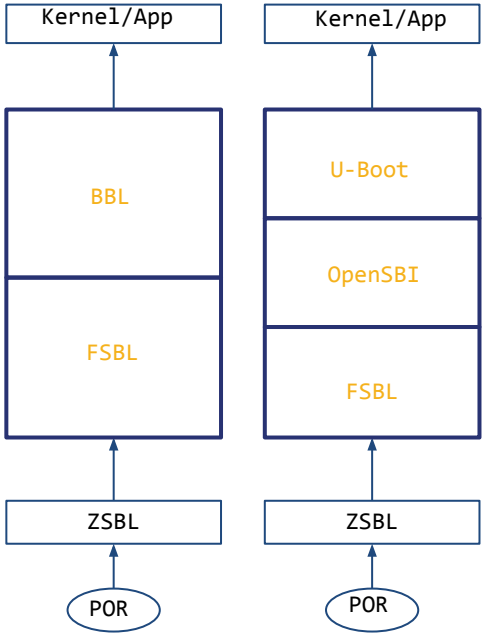




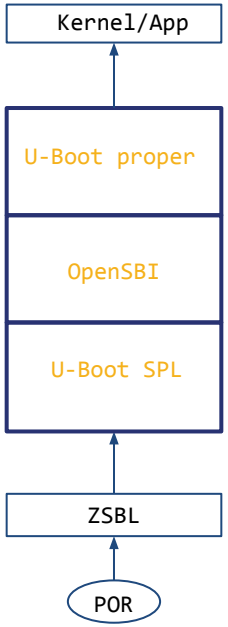
Till *v2.4*
- Unstructured device topology.
- Difficult to access and extend.

In *v2.5*
- Structured device topology.
- Meaningful way to access and extend.

Why Standard matter, Bootloader (RISC-V)



U-Boot



Till *v2020.04*
- Non standard booting stages
- Proprietary or SoC specific boot stages.

In *v2020.07*
- Standardized booting stages.
- Open Source accepted methodology.
- Generic for all RISC-V platforms.

U-Boot Mainline support

[U-Boot,v2,00/10] RISC-V AX25-AE350 support SPL

Message ID 20191114055230.20289-1-uboot@andestech.com
Headers show
Series RISC-V AX25-AE350 support SPL | expand

Message

Andes

From: Rick Chen <rick@andestech.com>

This series add support for SPL to AX25-AE350.

U-Boot SPL can boot from RAM or ROM and jump to
OPenSbi(FW_DYNAMIC firmware) and U-Boot proper from
RAM or MMC devices.

Also fix some bugs of andes plic driver and improve cache
configurations for SPL.

Changes in v2:
- Remove SYS_NS16550.
- Use CONFIG_IS_ENABLED(RISCV_MMODE).
- Add ALIGN(8) in ld for RV64.
- Add new [PATCH v2 09/10] riscv: dts: Add #address-cells and #size-cells in nor node.
- Add new [PATCH v2 10/10] doc: update AX25-AE350 RISC-V documentation.

Rick Chen (10):
  riscv: ax25: add SPL support
  riscv: ax25-ae350: add SPL configuration
  riscv: ax25-ae350: Use generic memory_size setup
  riscv: andes_plic: Fix some wrong configurations
  riscv: ax25: cache: Add SPL_RISCV_MMODE for SPL
  spl: cache: Allow cache drivers in SPL
  riscv: Fix clear bss loop in the start-up code
  riscv: dts: Support four cores SMP
  riscv: dts: Add #address-cells and #size-cells in nor node
  doc: update AX25-AE350 RISC-V documentation

[v5,0/6] riscv: sifive/fu540: SPI boot

Message ID 20200715100903.161363-1-jagan@amarulasolutions.com
Headers show
Series riscv: sifive/fu540: SPI boot | expand

Message

Jagan Teki

Updated series with boot device detection directly
on spl_boot_device function instead of having separate
board driver.

Changes for v5:
- rebase on master
Changes for v4:
- fix typo and unneeded configs.
Changes for v3:
- fixed env definitions build
- added boot device detection in board
Changes for v2:
- fu540 board driver
- runtime bootmode detection
- rebase on Pragnesh v11 series

Any inputs?
Jagan.

Jagan Teki (6):
  sifive: fu540: Add runtime boot mode detection
  sifive: fu540: Add Booting from SPI
  env: Enable SPI flash env for SiFive FU540
  sifive: fu540: Mark the default env as SPI flash
  sifive: fu540: Add boot flash script offset, size
  sifive: fu540: Enable SF distro bootcmd

[v13,00/19] RISC-V SiFive FU540 support SPL

Message ID 20200529060340.26708-1-pragnesh.patel@sifive.com
Headers show
Series RISC-V SiFive FU540 support SPL | expand

Message

Pragnesh Patel

This series add support for SPL to FU540. U-Boot SPL can boot from
L2 LIM (0x0800_0000) and jump to OpenSBI(FW_DYNAMIC firmware) and
U-Boot proper from MMC devices.

This series is also available here [1] for testing
[1] https://github.com/pragnesh26992/u-boot/tree/spl

How to test this patch:
1) Go to OpenSBI-dir : make PLATFORM=generic FW_DYNAMIC=y
2) export OPENSBI=<path to opensbi/build/platform/generic/firmware/fw_dynamic.bin>
3) Change to u-boot-dir
4) make sifive_fu540_defconfig
5) make all
6) Format the SD card (make sure the disk has GPT, otherwise use gdisk to switch)

```
        # sudo sgdisk --clear \
        > --set-alignment=2 \
        > --new=1:34:2081 --change-name=1:loader1 --typecode=1:5B193300-FC78-40CD-8002-E86C45580B47 \
        > --new=2:2082:10273 --change-name=2:loader2 --typecode=2:2E54B353-1271-4842-806F-E436D6AF6985 \
        > --new=3:10274: --change-name=3:rootfs --typecode=3:0FC63DAF-8483-4772-8E79-3D69D8477DE4 \
        > /dev/sda
```

Jagan Teki (2):
  sifive: fu540: Add sample SD gpt partition layout
  sifive: fu540: Add U-Boot proper sector start

Pragnesh Patel (17):
  misc: add driver for the SiFive otp controller
  riscv: sifive: fu540: Use OTP DM driver for serial environment
    variable
  riscv: Add _image_binary_end for SPL
  lib: Makefile: build crc7.c when CONFIG_MMC_SPI
  riscv: sifive: dts: fu540: Add board -u-boot.dtsi files
  sifive: fu540: add ddr driver
  sifive: dts: fu540: Add DDR controller and phy register settings
  riscv: sifive: dts: fu540: add U-Boot dmc node
  clk: sifive: fu540-prci: Add clock enable and disable ops
  clk: sifive: fu540-prci: Add ddr clock initialization
  clk: sifive: fu540-prci: Release ethernet clock reset
  riscv: sifive: dts: fu540: set ethernet clock rate
  riscv: dts: sifive: Sync hifive-unleashed-a00 dts from linux
  riscv: cpu: fu540: Add support for cpu fu540
  riscv: sifive: fu540: add SPL configuration
  configs: fu540: Add config options for U-Boot SPL
  doc: sifive: fu540: Add description for OpenSBI generic platform

Summary

- ➔ Other Projects
- ➔ Booting from RAM
- ➔ Booting from SD
- ➔ Booting from SPI
- ➔ EFI Booting

Other Projects

- Bootloaders:
  - U-Boot
  - Coreboot
  - Grub
  - EDK2
- Linux kernel
  - Works with v5.7-rc1
- Build Systems/distros
  - Buildroot (Mainline U-Boot, Linux support patches in Mailing-list[1])
  - Yocto
  - Fedorda

[1] https://patchwork.ozlabs.org/project/buildroot/patch/20200506100845.4356-2-jagan@amarulasolutions.com/

Booting from RAM

OpenSBI

```
$ git clone https://github.com/riscv/opensbi.git
$ cd opensbi
$ CROSS_COMPILE=riscv64-buildroot-linux-gnu- make PLATFORM=andes/ae350
(copy build/platform/andes/ae350/firmware/fw_dynamic.bin in U-Boot tree)
```

U-Boot

```
$ git clone https://gitlab.denx.de/u-boot/u-boot
$ cd u-boot
$ make ae350_rv64_spl_defconfig
$ CROSS_COMPILE=riscv64-buildroot-linux-gnu- make
```

Program

➔ U-Boot SPL will be loaded by gdb and runs in RAM in machine mode and then load FIT image from RAM device on AE350.

Boot Log

```
U-Boot SPL 2020.07 (Jul 16 2020 - 23:46:11 +0530)
Trying to boot from RAM

U-Boot 2020.07 (Jul 16 2020 - 23:46:11 +0530)

DRAM:  1 GiB
Flash: 64 MiB
MMC:   mmc@f0e00000: 0
Loading Environment from SPI Flash... SF: Detected mx25u1635e with page
size 256 Bytes, erase size 4 KiB, total 2 MiB
OK
In:    serial@f0300000
Out:   serial@f0300000
Err:   serial@f0300000
Net:   no alias for ethernet0
Warning: mac@e0100000 (eth0) using random MAC address - a2:ae:93:7b:cc:8f
eth0: mac@e0100000
Hit any key to stop autoboot:  0
```

Booting from SD

OpenSBI

```
$ git clone https://github.com/riscv/opensbi.git
$ cd opensbi
$ CROSS_COMPILE=riscv64-buildroot-linux-gnu- make PLATFORM=generic
$ export OPENSBI=/path/to/opensbi/build/platform/generic/firmware/fw_dynamic.bin
```

U-Boot

```
$ git clone https://gitlab.denx.de/u-boot/u-boot
$ cd u-boot
$ make sifive_fu540_defconfig
$ CROSS_COMPILE=riscv64-buildroot-linux-gnu- make
```

Format SD card

```
$ sudo sgdisk --clear --set-alignment=2 \
> --new=1:34:2081 --change-name=1:loader1 --typecode=1:5B193300-FC78-40CD-8002-E86C45580B47 \
> --new=2:2082:10273 --change-name=2:loader2 --typecode=2:2E54B353-1271-4842-806F-E436D6AF6985 \
> --new=3:10274: --change-name=3:rootfs --typecode=3:0FC63DAF-8483-4772-8E79-3D69D8477DE4 \
> /dev/sda
```

Program SD card

```
$ sudo dd if=spl/u-boot-spl.bin of=/dev/sda seek=34
$ sudo dd if=u-boot.itb of=/dev/sda seek=2082
```

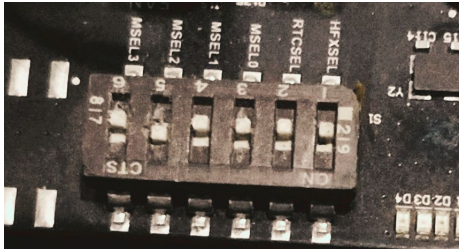
Boot Log

```
U-Boot SPL 2020.07 (July 16 2020 - 15:01:12 +0530)
Trying to boot from MMC1

U-Boot 2020.07 (July 16 2020 - 15:01:12 +0530)

CPU:   rv64imafdc
Model: SiFive HiFive Unleashed A00
DRAM:  8 GiB
MMC:   spi@10050000:mmc@0: 0
In:    serial@10010000
Out:   serial@10010000
Err:   serial@10010000
Net:   eth0: ethernet@10090000
Hit any key to stop autoboot:  0
```

SD Boot Jumber (Set MSEL[3:0] to 1011)



Booting from SPI

OpenSBI

```
$ git clone https://github.com/riscv/opensbi.git
$ cd opensbi
$ CROSS_COMPILE=riscv64-buildroot-linux-gnu- make PLATFORM=generic
$ export OPENSBI=/path/to/opensbi/build/platform/generic/firmware/fw_dynamic.bin
```

U-Boot

```
$ git clone https://gitlab.denx.de/u-boot/u-boot
$ cd u-boot
$ make sifive_fu540_defconfig
$ CROSS_COMPILE=riscv64-buildroot-linux-gnu- make
```

Partition SPI at Linux[1] Once SD Booted

```
$ sudo sgdisk --clear --set-alignment=2 \
> --new=1:34:2081 --change-name=1:loader1 --typecode=1:5B193300-FC78-40CD-8002-E86C45580B47 \
> --new=2:2082:10273 --change-name=2:loader2 --typecode=2:2E54B353-1271-4842-806F-E436D6AF6985 \
> --new=3:10274: --change-name=3:rootfs --typecode=3:0FC63DAF-8483-4772-8E79-3D69D8477DE4 \
> /dev/mtdblock0
```

Program SPI on U-Boot

```
# tftpboot $kernel_addr_r u-boot-spl.bin
# sf erase 0x5000 $filesize
# sf write $kernel_addr_r 0x5000 $filesize
# tftpboot $kernel_addr_r u-boot.itb
# sf erase 0x105000 $filesize
# sf write $kernel_addr_r 0x105000 $filesize
```

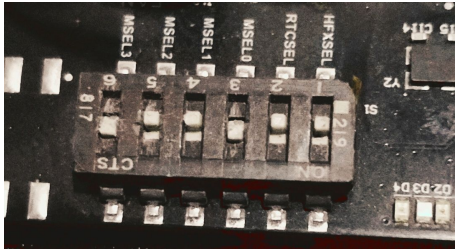
Boot Log

```
U-Boot SPL 2020.07 (July 16 2020 - 16:12:02 +0530)
Trying to boot from SPI

U-Boot 2020.07 (July 16 2020 - 16:12:02 +0530)

CPU:   rv64imafdc
Model: SiFive HiFive Unleashed A00
DRAM:  8 GiB
MMC:   spi@10050000:mmc@0: 0
In:    serial@10010000
Out:   serial@10010000
Err:   serial@10010000
Net:   eth0: ethernet@10090000
Hit any key to stop autoboot:  0
```

SPI Boot Jumber (Set MSEL[3:0] to 0110)



[1] https://github.com/amarula/bsp-sifive

Future plans

- ➔ RISC-V EFI runtime
- ➔ RISC-V Falcon Boot
- ➔ RISC-V Android Verified Boot
- ➔ RISC-V Android

References

- Working experience on RISC-V
- Wiki - https://wiki.amarulasolutions.com/bsp/riscv/hifive-unleashed.html
- OpenSBI - https://github.com/riscv/opensbi
- Jagan “An Introduction to RISC-V Boot flow”
  https://crvf2019.github.io/pdf/43.pdf
- Atish Patra “An Introduction to RISC-V Boot flow”
  https://content.riscv.org/wp-content/uploads/2019/12/Summit_bootflow.pdf

Questions??

Thank you

Jagan Teki <jagan@amarulasolutions.com>