



U-Boot from Scratch

Jagan Teki

FOSDEM - 2019

Jagan Teki

- JagannadhaSutradharudu Teki
- Enthusiastic Linux kernel hacker
- U-Boot Maintainer for SPI, SPI-FLASH, Allwinner sunXi SoC
- Buildroot, Yocto contributor
- Heading to **Amarula Solutions India**

Agenda

- U-Boot, In a NutShell
- Image boot
- Features
- Port new hardware
- Future plans



U-Boot, In a NutShell

- History
- Community
- Hardware support
- U-Boot
- Source
- Source tree
- Build process
- Boot Sequence
- Loading sources
- Debug
- Tools
- Testing

History

- 8xxROM written by Magnus Damm
- October 1999 Wolfgang Denk moved to sourceForge.net and as PPCBoot
- PPCBoot-0.4.1 released in July 19, 2000
- PPCBoot-2.0.0 became U-Boot-0.1.0 with x86
- Since then added many architectures, boards, features etc.
- Current name is termed as Das U-Boot
- Flexible and opensource bootloader
- Wolfgang Denk as head custodian for over 10 years
- Tom Rini as head custodian since September 2012
- Recent release v2019.01

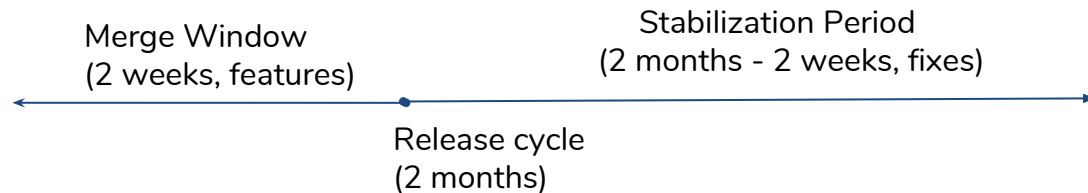
Community

- On average over 25+ employers and 120+ developers contributing every release
- Number of talks on various conferences

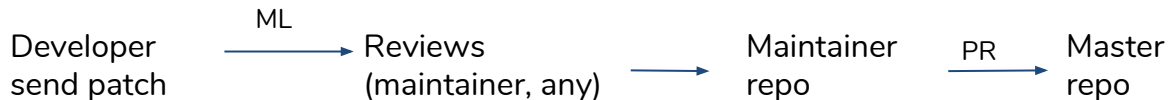
Development process:

- 35+ Custodian/Maintainer for various subsystems
- Release cycle

Note: RC now changed to 3 months, with 3 weeks MW



- Workflow



Hardware support

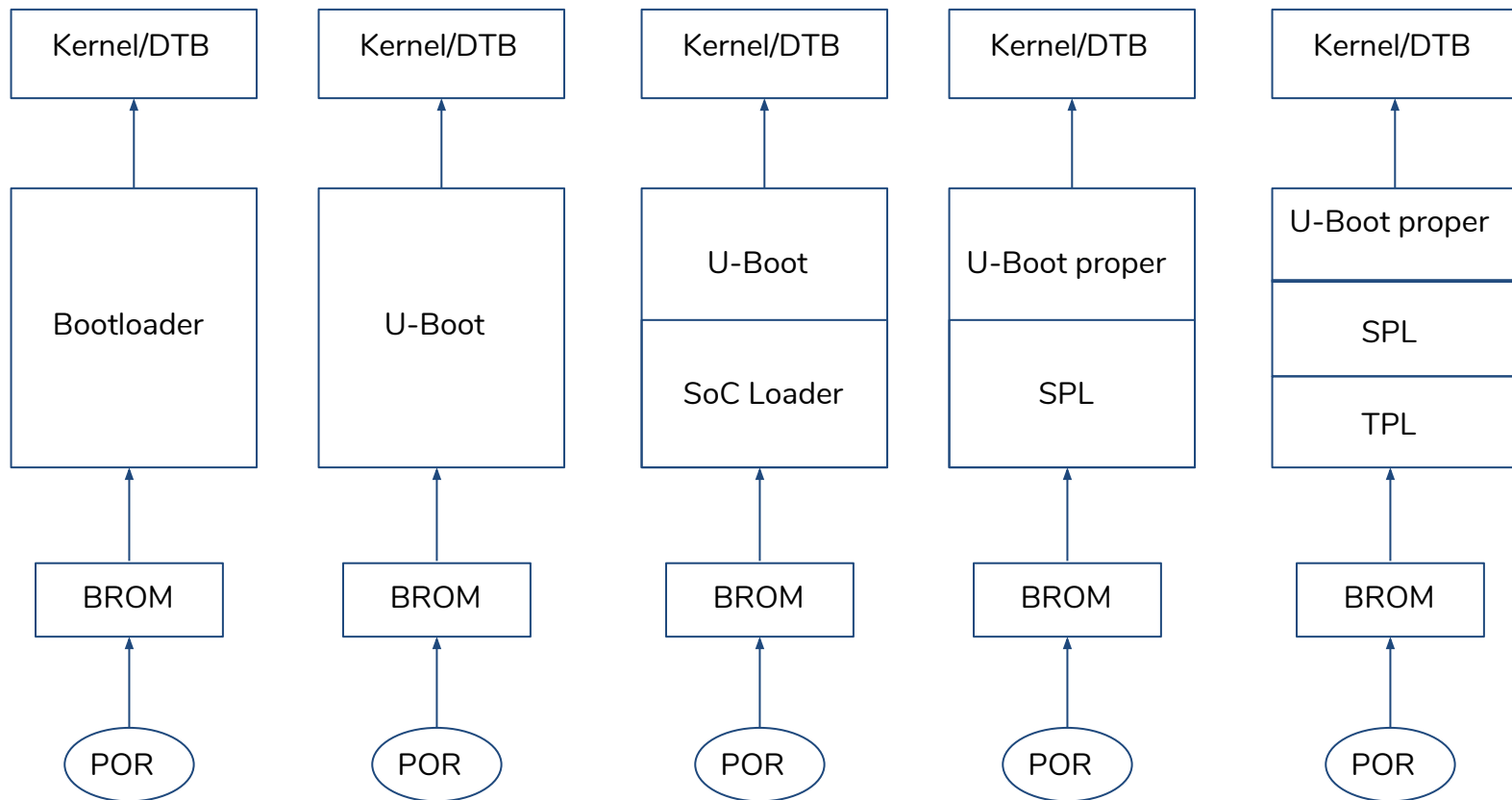
Architecture/SoC:

- ARM
 - ◆ 32-bit: Aspeed, Altera, Allwinner, Atmel, Broadcom, Qemu, Qualcomm, Marvell, NXP, Rockchip, STM32, Tegra, TI, UniPhier, Xilinx
 - ◆ 64-bit: Allwinner, Marvell, NXP, Rockchip, Tegra, UniPhier, Xilinx
- X86 (Baytrail, Broadwell, Quark, etc)
- ARC, M68K, MicroBlaze, MIPS, NDS32, NIOS2, PowerPC, RISCV, Sandbox, SuperH, Xtensa

Boards:

- 186+ different board vendors
- 1083+ different boards

U-Boot



U-Boot build

- Git master or dev tree at <http://git.denx.de/?p=u-boot.git;a=summary>
- Custodian's or Maintainers tree at <http://git.denx.de/?p=u-boot.git;a=forks>

Example of building vyasa RK3288 board, which is ARM platform with arm-linux-gnueabi

1. `$ git clone git://git.denx.de/u-boot.git`
2. `$ u-boot`
3. `$ export ARCH=arm`
4. `$ export CROSS_COMPILE=arm-linux-gnueabi-`
5. `$ make vyasa-rk3288_defconfig`
6. `$ make`

U-Boot tree

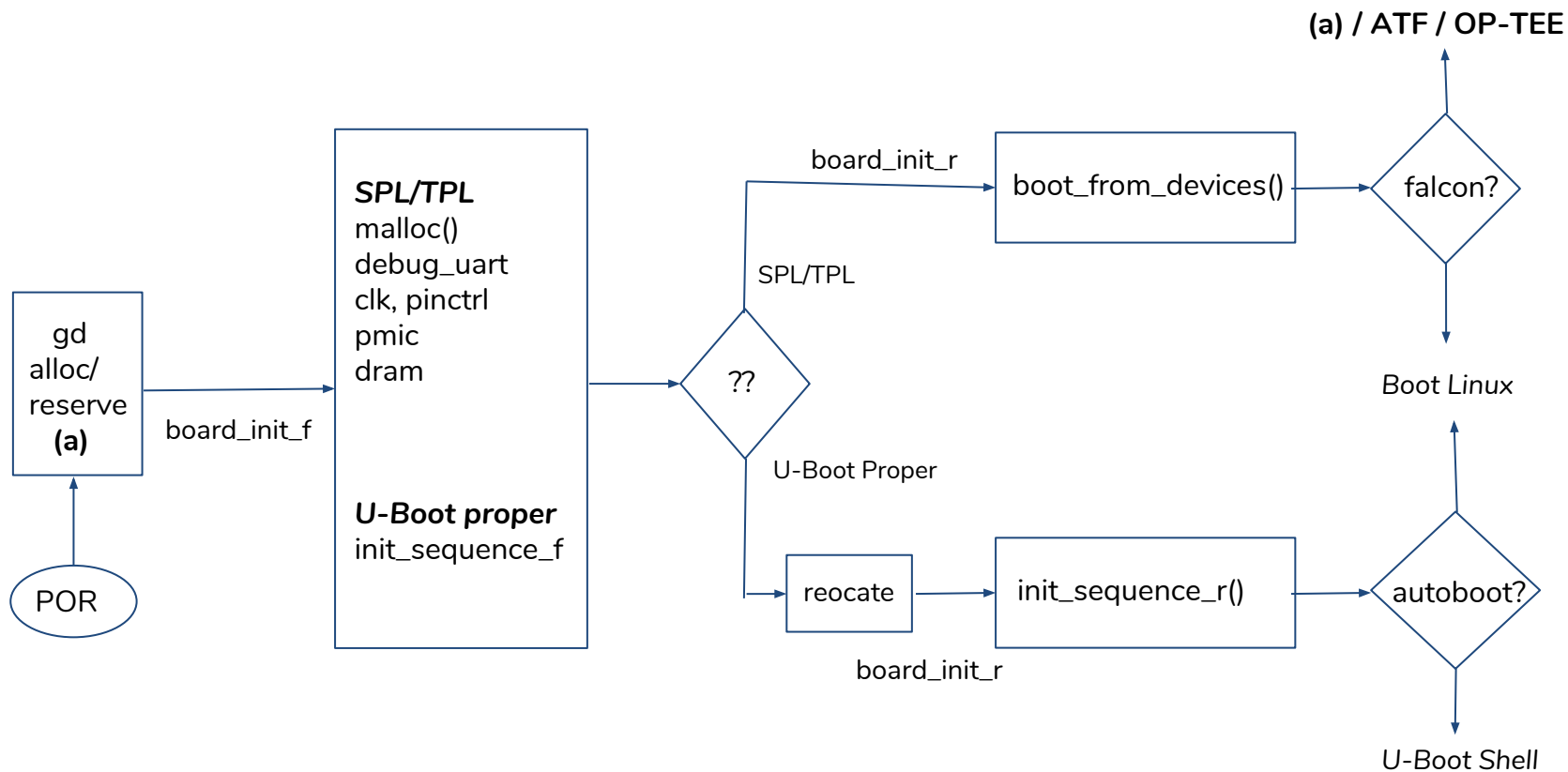
- Arch
 - ◆ arch/foo
 - Architecture specific code
 - ◆ arch/foo/mach-joo
 - foo CPU specific code
 - ◆ arch/foo/dts
 - joo MACHINE specific code
 - foo devicetree code
- board
 - board specific code
- cmd
 - commands
- common
 - common code
 - ◆ common/spl
 - SPL code
- configs
 - per board configuration
- disk
 - disk or partition
- doc
 - documentation
- drivers
 - drivers code
- env
 - environment
- fs
 - filesystems
- net
 - Networking
- Kbuild
 - Kbuild scripts
- Kconfig
 - Global kconfig file
- Makefile
 - Global makefile
- scripts, tools
 - build script, etc

Build process

```
1. scripts/kconfig/conf --syncconfig Kconfig
2. CHK include/config.h
3. UPD include/config.h
4. CFG u-boot.cfg
5. GEN include/autoconf.mk
6. GEN include/autoconf.mk.dep
7. CFG spl/u-boot.cfg
8. GEN spl/include/autoconf.mk
9. CFG tpl/u-boot.cfg
10. GEN tpl/include/autoconf.mk
11. CHK include/config/uboot.release
12. UPD include/config/uboot.release
13. CHK include/generated/version_autogenerated.h
14. UPD include/generated/version_autogenerated.h
15. CHK include/generated/timestamp_autogenerated.h
16.
17. LD arch/arm/cpu/built-in.o
18. CC arch/arm/cpu/armv7/cache_v7.o
19. AS arch/arm/cpu/armv7/cache_v7_asm.o
20. CC arch/arm/cpu/armv7/cpu.o
21. CC arch/arm/cpu/armv7/cp15.o
22. CC arch/arm/cpu/armv7/syslib.o
23. AS arch/arm/cpu/armv7/sctlr.o
24. AS arch/arm/cpu/armv7/lowlevel_init.o
25. LD arch/arm/cpu/armv7/built-in.o
26. AS arch/arm/cpu/armv7/start.o
27. CC arch/arm/lib/eabi_compat.o
28. AS arch/arm/lib/crt0_arm_efi.o
29. CC arch/arm/lib/reloc_arm_efi.o
30. CC arch/arm/mach-rockchip/boot_mode.o
31. CC arch/arm/mach-rockchip/rk3288-board.o
32. CC arch/arm/mach-rockchip/sdram_common.o
33. CC arch/arm/mach-rockchip/rk_timer.o
34. CC arch/arm/mach-rockchip/rk3288/clk_rk3288.o
35. CC arch/arm/mach-rockchip/rk3288/rk3288.o
36. CC arch/arm/mach-rockchip/rk3288/syscon_rk3288.o
37. CC board/amarula/vyasa-rk3288/vyasa-rk3288.o
```

```
1. CC spl/arch/arm/mach-rockchip/sdram_common.o
2. CC spl/arch/arm/mach-rockchip/rk_timer.o
3. CC spl/arch/arm/mach-rockchip/rk3288/clk_rk3288.o
4. CC spl/arch/arm/mach-rockchip/rk3288/rk3288.o
5. CC spl/arch/arm/mach-rockchip/rk3288/syscon_rk3288.o
6. CC spl/arch/arm/mach-rockchip/bootrom.o
7. CC spl/arch/arm/mach-rockchip/rk3288-board-spl.o
8. CC spl/arch/arm/cpu/armv7/cache_v7.o
9. CC spl/arch/arm/cpu/armv7/cpu.o
10. AS spl/arch/arm/cpu/armv7/lowlevel_init.o
11. AS spl/arch/arm/cpu/armv7/start.o
12. LD spl/u-boot-spl
13. OBJCOPY spl/u-boot-spl-nodtb.bin
14. COPY spl/u-boot-spl.dtb
15. CAT spl/u-boot-spl-dtb.bin
16. COPY spl/u-boot-spl.bin
17.
18. CC tpl/arch/arm/mach-rockchip/sdram_common.o
19. CC tpl/arch/arm/mach-rockchip/rk_timer.o
20. CC tpl/arch/arm/mach-rockchip/rk3288/clk_rk3288.o
21. CC tpl/arch/arm/mach-rockchip/rk3288/rk3288.o
22. CC tpl/arch/arm/mach-rockchip/rk3288/syscon_rk3288.o
23. CC tpl/arch/arm/mach-rockchip/bootrom.o
24. CC tpl/arch/arm/mach-rockchip/rk3288-board-tpl.o
25. CC tpl/arch/arm/cpu/armv7/cache_v7.o
26. AS tpl/arch/arm/cpu/armv7/cache_v7_asm.o
27. CC tpl/arch/arm/cpu/armv7/cpu.o
28. CC tpl/arch/arm/cpu/armv7/cp15.o
29. CC tpl/arch/arm/cpu/armv7/syslib.o
30. AS tpl/arch/arm/cpu/armv7/lowlevel_init.o
31. AS tpl/arch/arm/cpu/armv7/start.o
32. LDS tpl/u-boot-spl.lds
33. LD tpl/u-boot-tpl
34. OBJCOPY tpl/u-boot-tpl-nodtb.bin
35. COPY tpl/u-boot-tpl.bin
36. COPY u-boot.dtb
37. MKIMAGE u-boot-dtb.img
```

U-Boot Sequence



Loading sources

- U-Boot can support variety of loading sources
- RAM
- Network: tftp, DHCP
- Serial: Kermit
- SPI Flash
- MMC
- Parallel NOR
- NAND, UBI
- USB
- SATA
- AHCI
- NVME

Debug

- `printf`
- `CONFIG_DEBUG`
- `GDB`
- `Early UART (CONFIG_DEBUG_UART)`

```
#include <debug_uart.h>

static inline void _debug_uart_init(void)
{
    _mxc_serial_init(base);
}

static inline void _debug_uart_putc(int ch)
{
    while (!(readl(&base->ts) & UTS_TXEMPTY))
        WATCHDOG_RESET();

    writel(ch, &base->txd);
}

DEBUG_UART_FUNCS

debug_uart_init();
printch('T');
printch('P');
printch('L');
```

Tools

- Patman
 - ◆ Manual patch creation, cover-letter, adding maintainers etc can be error-prone.
 - ◆ Patman make above automated
 - ◆ Create patch, insert cover-letter, add maintainer (via ~/.git-mailrc), run checkpatch.pl etc
 - ◆ How to use? tools/patman/README
- Buildman
 - ◆ U-Boot builder for multiple commits, branches etc
 - ◆ Replaced by legacy MAKEALL
 - ◆ Understandable output summary
 - ◆ Checking image sizes
 - ◆ tools/buildman/README
- Binman
 - ◆ Packaging multiple image components

```
/ {  
    binman {  
        filename = "u-boot-sunxi-with-spl.bin";  
        pad-byte = <0xff>;  
        blob {  
            filename = "spl/sunxi-spl.bin";  
        };  
        u-boot-img {  
            offset = <CONFIG_SPL_PAD_TO>;  
        };  
    };  
};
```


Testing

- travis-ci.org
 - ◆ Automated build environment, with limited run-time, free to use
 - ◆ May take longer duration, if more jobs are initiated
 - ◆ .travis.yml, u-boot travis build plugin
- test.py
 - ◆ Pytest framework
 - ◆ Works for sandbox, qemu, some real hardware
 - ◆ Sanity tests for dm code
 - ◆ doc/README.trace
- trace
 - ◆ Kind of Linux ftrace
 - ◆ Collect execution and sent to host for analysis
- tbot
 - ◆ Execute test cases on boards
 - ◆ Heiko Schocher demo, <https://www.youtube.com/watch?v=zfpj3DLsx4>

Image Boot

- Legacy image
- FIT
- Verified image
- ARM64 ATF
- OP-TEE
- Secure boot
- Falcon mode
- EFI boot
- Distro boot

Legacy Image

- Fixed offset images - standalone, zImage binaries
- go addr [arg ...]
- u-boot Image format
- Single component ulmage
- Monolithic, combination of images
- bootm [addr [arg ...]]

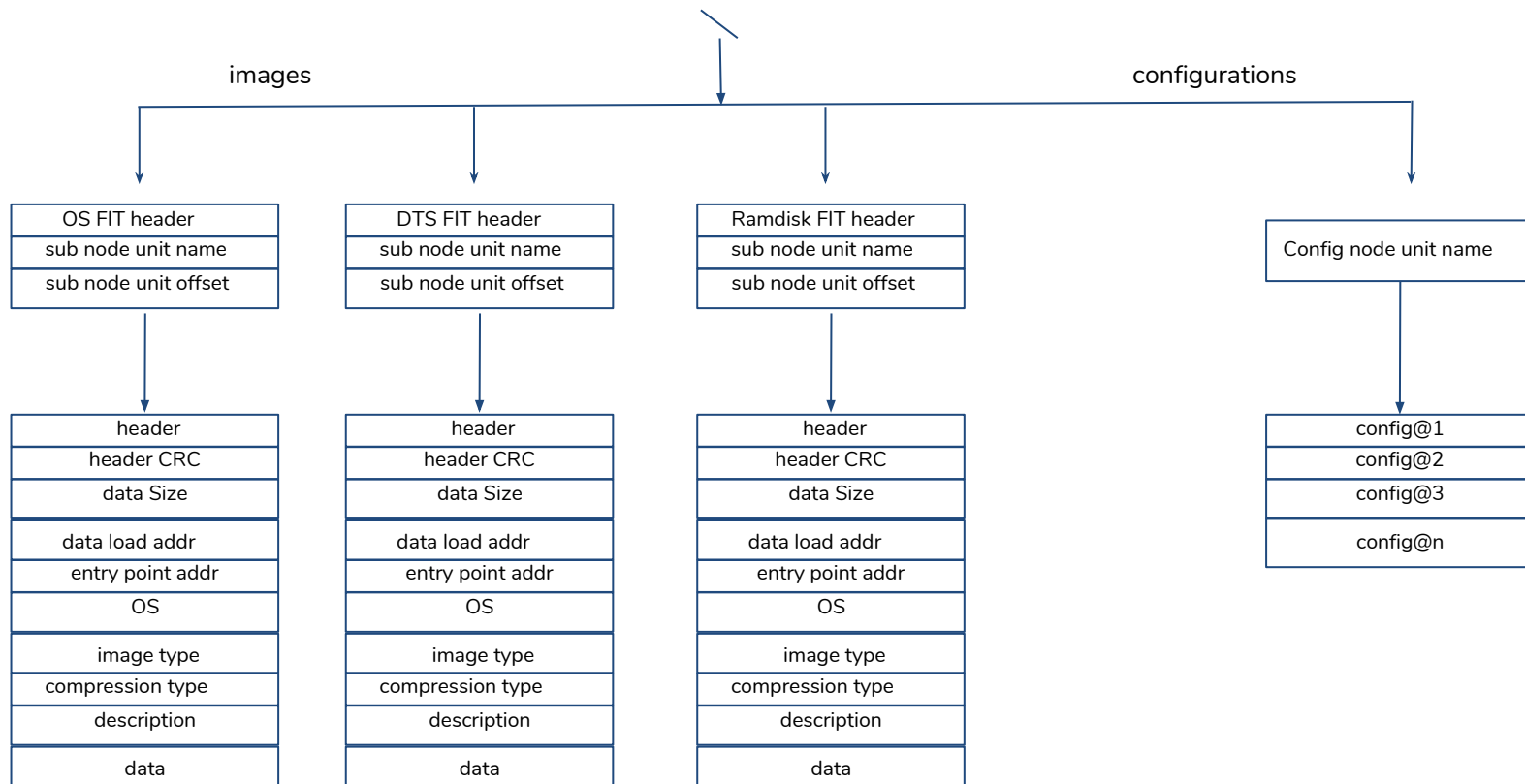
? Not flexible, indexing

? No hash integrity

? No scope of security addition

```
1 $ mkimage \  
2 > -A arm \  
3 > -O linux \  
4 > -T kernel \  
5 > -a 0x10008000 \  
6 > -e 0x10008000 \  
7 > -n "Linux-4.19.0-rc2" \  
8 > -d zImage uImage  
9 Image Name: Linux-4.19.0-rc2-next-20180905-0  
10 Created: Tue Sep 11 23:07:55 2018  
11 Image Type: ARM Linux Kernel Image (uncompressed)  
12 Data Size: 8372992 Bytes = 8176.75 kB = 7.99 MB  
13 Load Address: 10008000  
14 Entry Point: 10008000  
15  
16  
17 $ mkimage  
18 > -A arm \  
19 > -O linux \  
20 > -T multi \  
21 > -a 0x10008000 \  
22 > -e 0x10008000 \  
23 > -n 'Multi image' \  
24 > -d vmlinux.bin.gz:ramdisk.image.gz:imx6q-icore-rqs.dtb uMulti  
25 Image Name: Multi image  
26 Created: Tue Sep 11 23:07:55 2018  
27 Image Type: ARM Linux Multi-File Image (gzip compressed)  
28 Data Size: 82092755 Bytes = 80168.71 KiB = 78.29 MiB  
29 Load Address: 10008000  
30 Entry Point: 10008000  
31 Contents:  
32 Image 0: 7609333 Bytes = 7430.99 KiB = 7.26 MiB  
33 Image 1: 74445331 Bytes = 72700.52 KiB = 71.00 MiB  
34 Image 2: 38071 Bytes = 37.18 KiB = 0.04 MiB  
35
```

FIT (Flattened ulmage Tree)



FIT, contd

```
1 /dts-v1/;
2 / {
3     description = "FIT with single Linux kernel and FDT blob";
4     #address-cells = <1>;
5     images {
6         kernel@1 {
7             description = "i.MX6 Linux kernel";
8             data = /incbin(/". /vmlinux.bin.gz");
9             type = "kernel";
10            arch = "arm";
11            os = "linux";
12            compression = "gzip";
13            load = <0x10008000>;
14            entry = <0x10008000>;
15            hash@1 {
16                algo = "md5";
17            };
18            hash@2 {
19                algo = "sha1";
20            };
21        };
22        fdt@1 {
23            description = "Engicam i.CoreM6 Quad/Dual RQS Starter Kit Devicetree blob";
24            data = /incbin(/". /imx6q-icore-rqs.dtb");
25            type = "flat_dt";
26            arch = "arm";
27            compression = "none";
28            hash@1 {
29                algo = "md5";
30            };
31            hash@2 {
32                algo = "sha1";
33            };
34        };
35    };
36 };
37
38 configurations {
39     default = "conf@1";
40     conf@1 {
41         description = "Boot Linux kernel with FDT blob";
42         kernel = "kernel@1";
43         fdt = "fdt@1";
44     };
45 };
46 };
```

```
1 icore6qdl-rqs> fatload mmc 0:1 $loadaddr fit.itb
2 reading fit.itb
3 6167494 bytes read in 335 ms (17.6 MiB/s)
4 icore6qdl-rqs> bootm $loadaddr
5 Booting FIT image from mmc ...
6 ## Loading kernel from FIT Image at 12000000 ...
7 Using 'conf@1' configuration
8 Verifying Hash Integrity ... OK
9 Trying 'kernel@1' kernel subimage
10 Description: i.MX6 Linux kernel
11 Type: Kernel Image
12 Compression: gzip compressed
13 Data Start: 0x120000f0
14 Data Size: 6130148 Bytes = 5.8 MiB
15 Architecture: ARM
16 OS: Linux
17 Load Address: 0x10008000
18 Entry Point: 0x10008000
19 Hash algo: md5
20 Hash value: b975a202ea2963c53c53f527329930cd
21 Hash algo: sha1
22 Hash value: 78b93fe404b795de8c837af27d67f4df9b96083a
23 Verifying Hash Integrity ... md5+ sha1+ OK
24 ## Loading fdt from FIT Image at 12000000 ...
25 Using 'conf@1' configuration
26 Trying 'fdt@1' fdt subimage
27 Description: Engicam i.CoreM6 Quad/Dual RQS Starter Kit Devicetree blob
28 Type: Flat Device Tree
29 Compression: uncompressed
30 Data Start: 0x125d8dbc
31 Data Size: 35298 Bytes = 34.5 KiB
32 Architecture: ARM
33 Hash algo: md5
34 Hash value: 4371a4dfe55127c2fda8a9feb4d3b313
35 Hash algo: sha1
36 Hash value: e34a9326b5e7fd43557753ef980fe67326f82ea1
37 Verifying Hash Integrity ... md5+ sha1+ OK
38 Booting using the fdt blob at 0x125d8dbc
39 Uncompressing Kernel Image ... OK
40 Using Device Tree in place at 125d8dbc, end 125e479d
41
42 Starting kernel ...
```

FIT Complexity ?

Multiple kernel, dtb, ramdisk

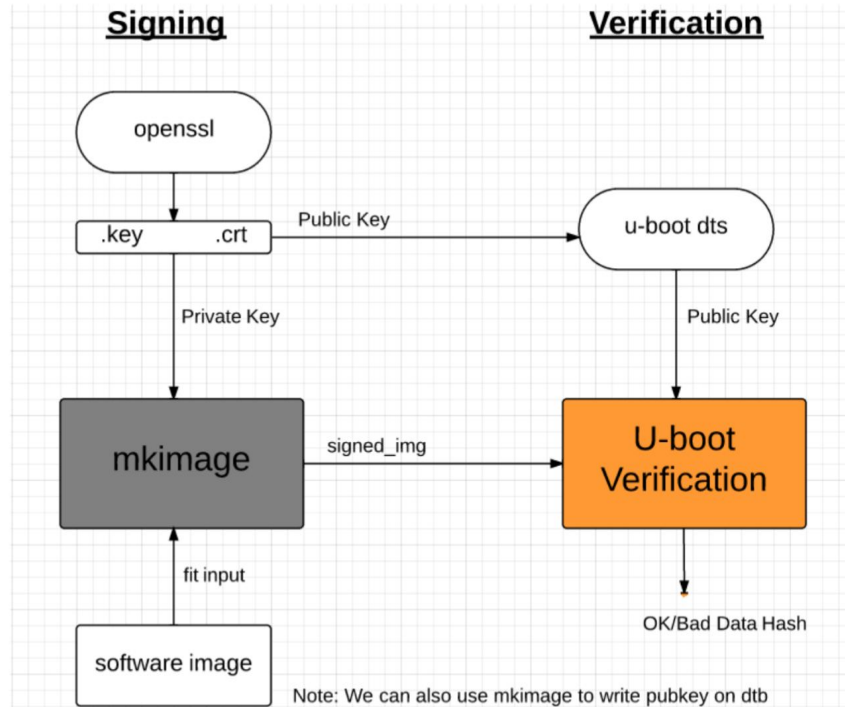
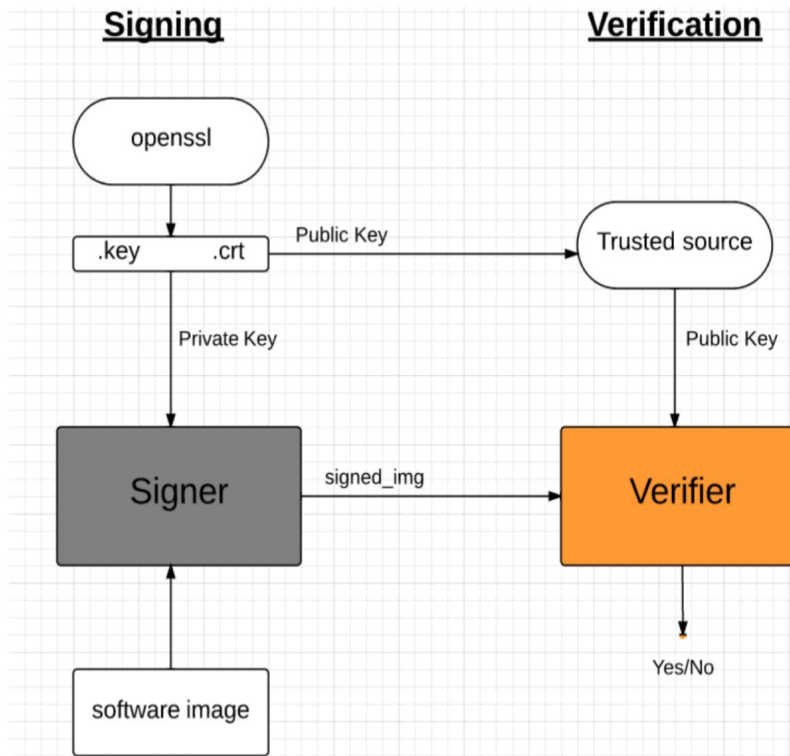
bit file, fpga

ATF

OPTEE

new image type?

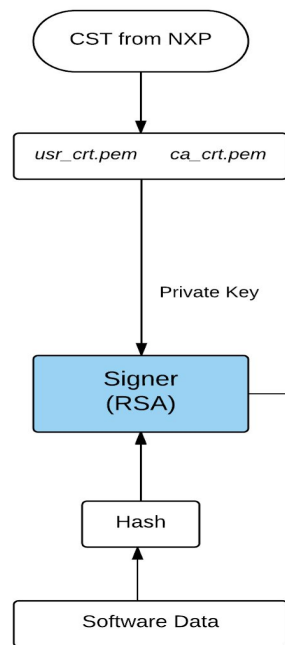
Verified boot



Secure boot, Signed

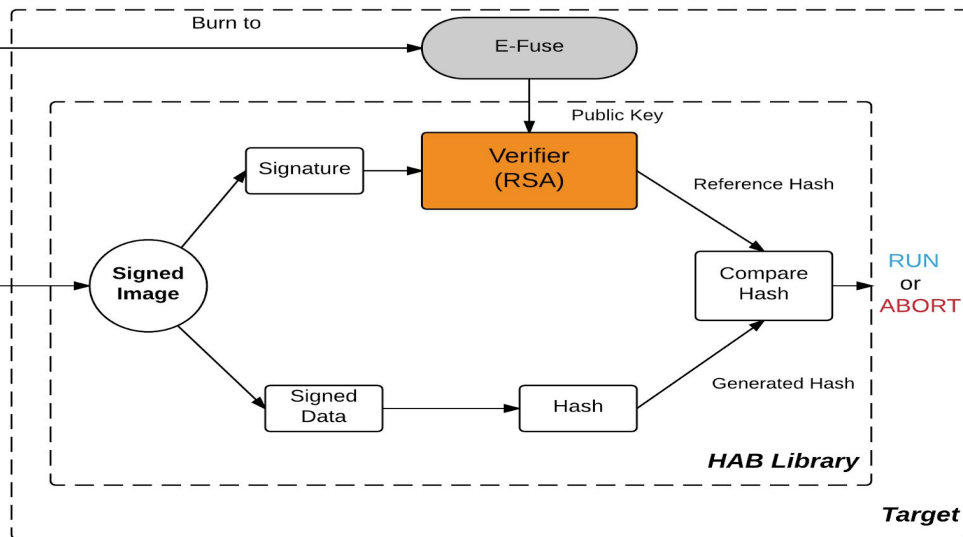
Signing

(Image Signing using Private key)



Verification

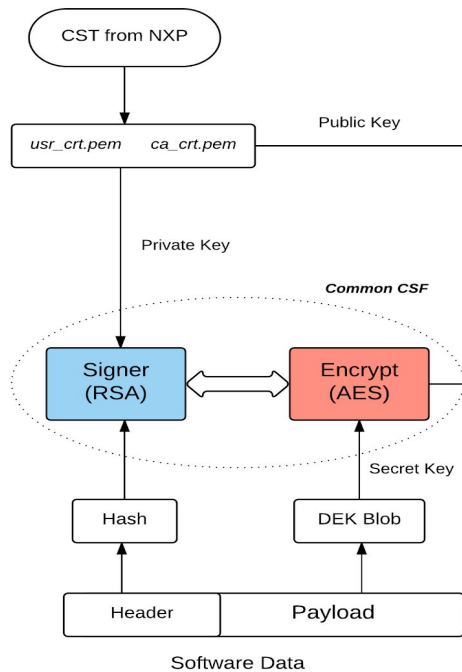
(Image Verify/authenticate using Public key)



Secure boot, Encrypted

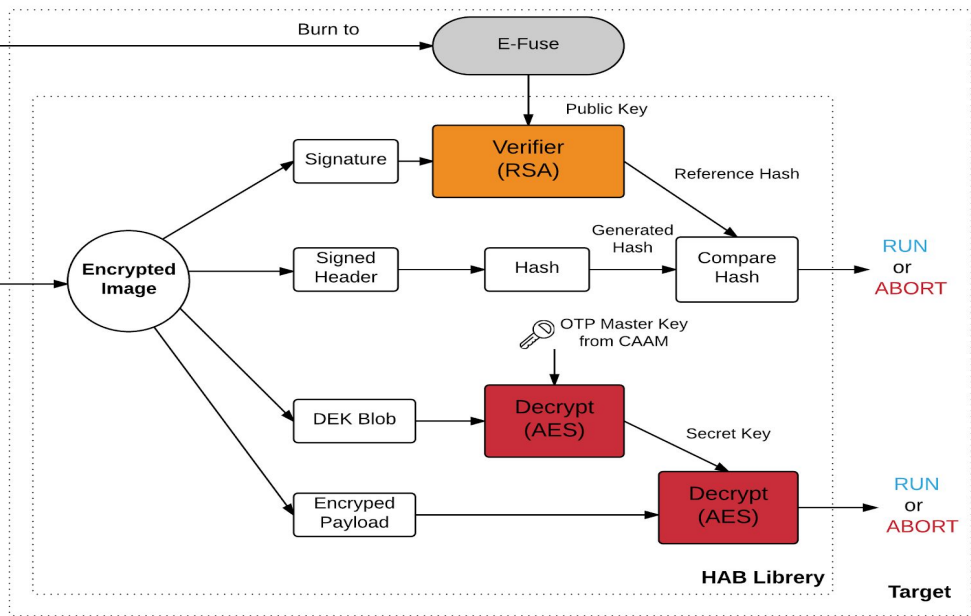
Encryption

(Image Encrypt using Secret key)

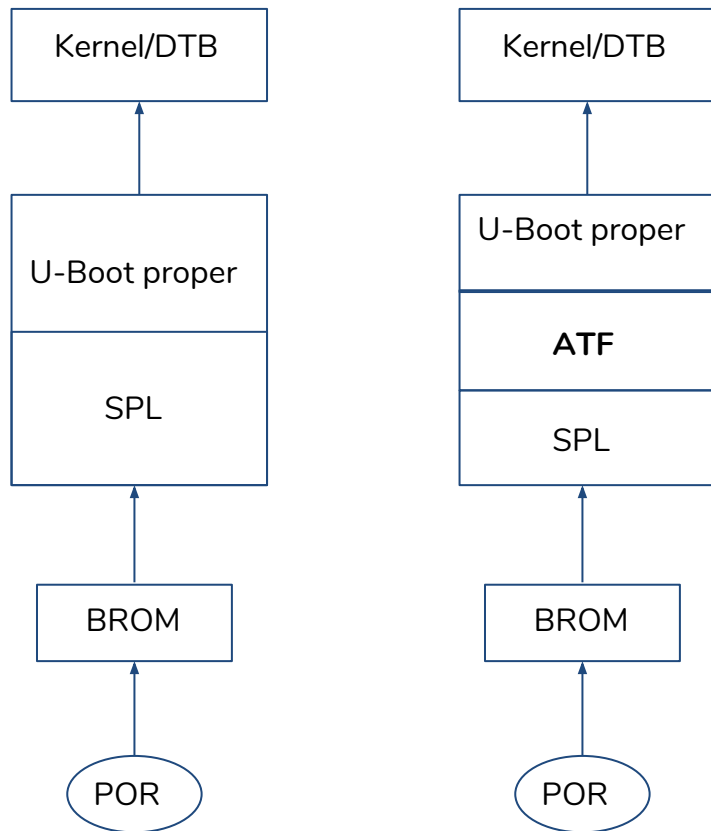


Decryption

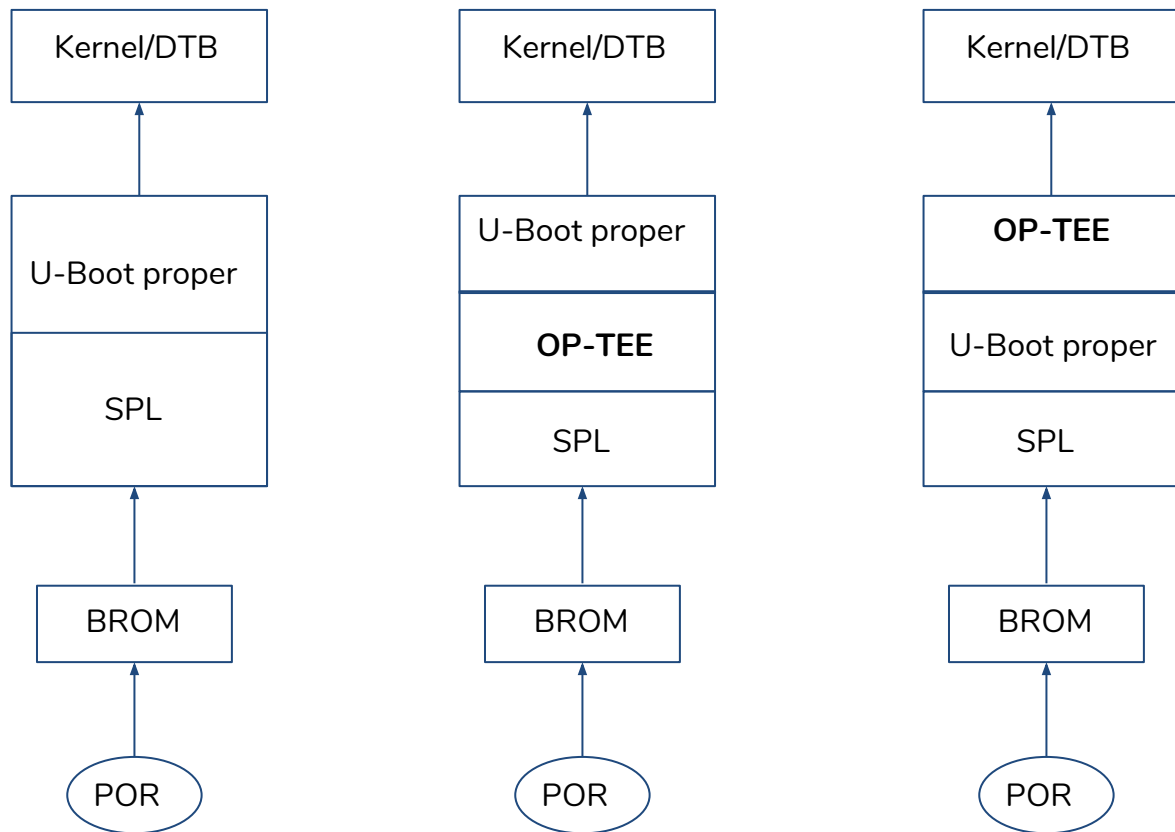
(Image Decrypt using Secret key)



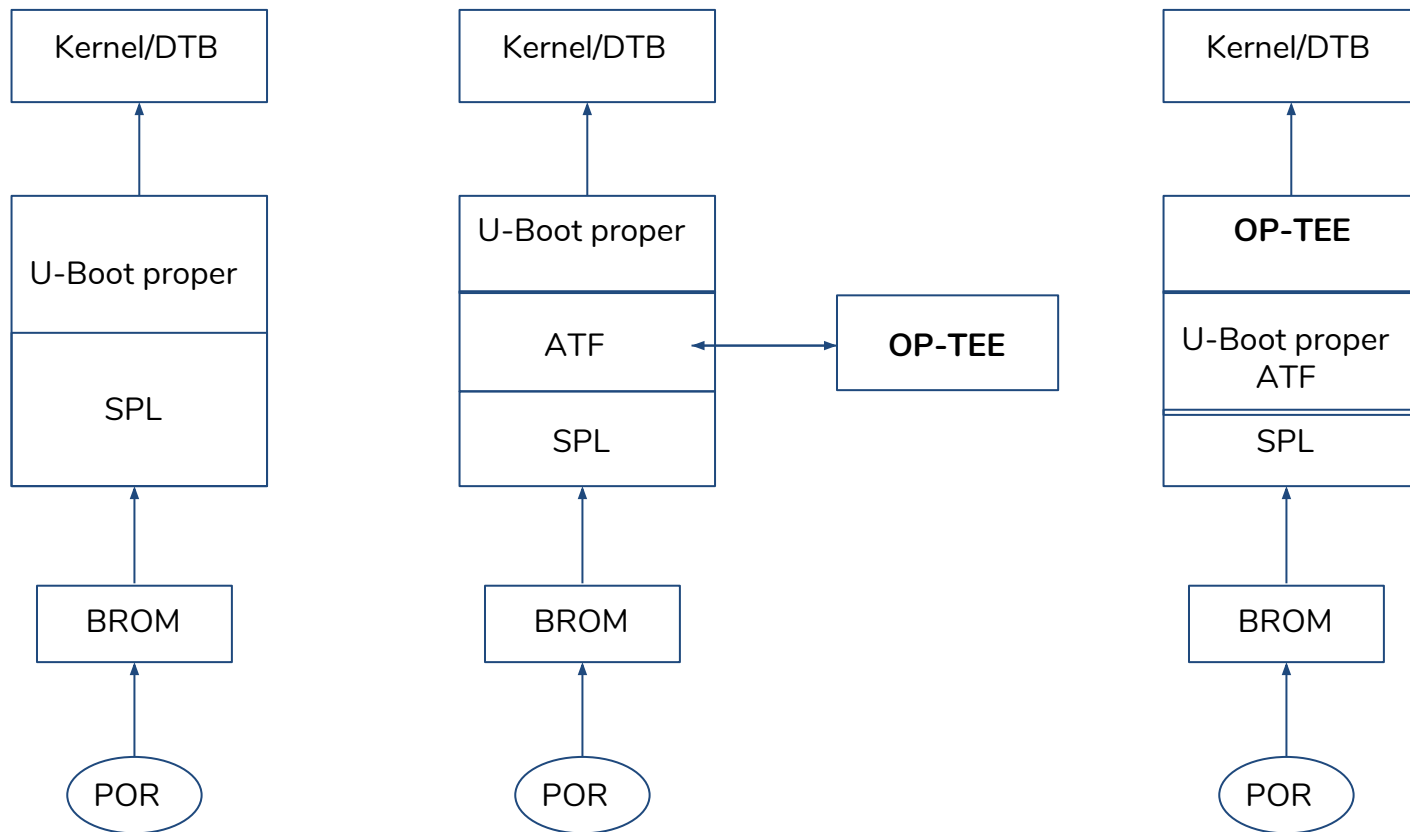
ATF



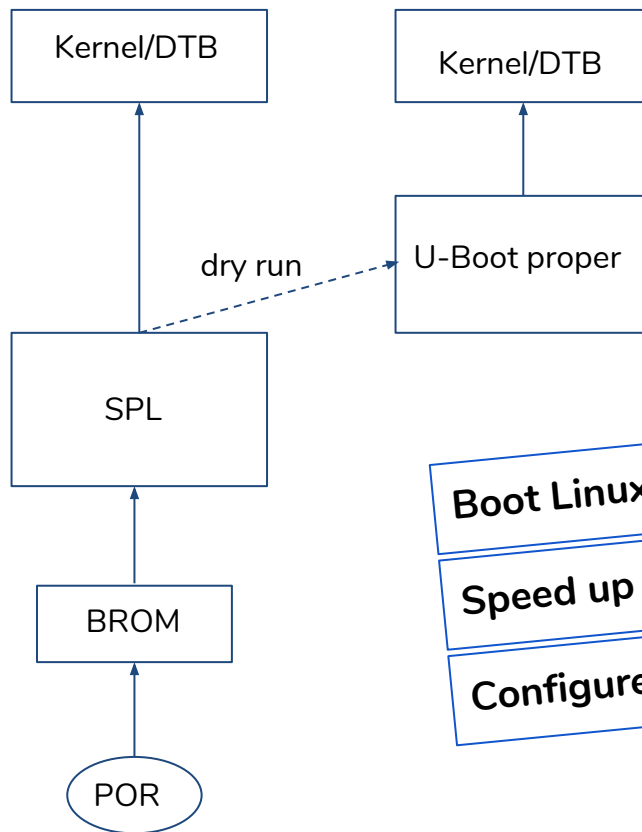
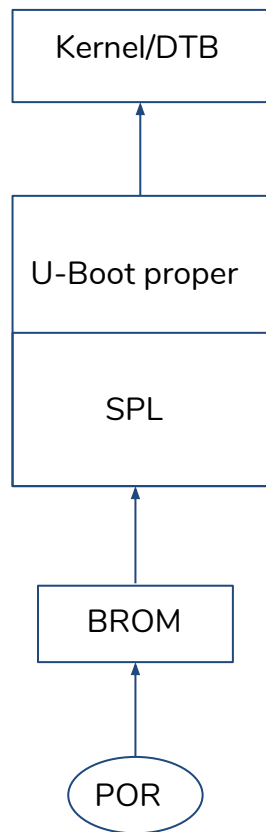
OP-TEE, ARMv7



OP-TEE, ARMv8



Falcon mode



Boot Linux from SPL

Speed up boot

Configure falcon, during dryrun

Falcon, is useful?

```
[0.001344 0.001342] U-Boot TPL 2017.09-rc2-13373-g2cffd0d-dirty (Aug 31 2017 - 20:41:14)
[0.005975 0.004631] Trying to boot from BOOTROM
[0.008394 0.002419] Returning to boot ROM...
[0.216735 0.208341]
[0.217195 0.000460] U-Boot SPL 2017.09-rc2-13373-g2cffd0d-dirty (Aug 31 2017 - 20:41:14)
[0.223097 0.005902] Trying to boot from MMC1[0.262093 0.038996] Expected Linux image is not found. Trying to start U-boot
[0.436129 0.174036]]
[0.436416 0.000287]
[0.436696 0.000280] U-Boot 2017.09-rc2-13373-g2cffd0d-dirty (Aug 31 2017 - 20:41:14 +0530)
[0.442273 0.005577] [0.442369 0.000096] Model: Amarula Vyasa-RK3288
[0.444854 0.002485] DRAM: 2 GiB
[0.479422 0.034568] MMC: dwmmc@ff0c0000: 1
[0.627295 0.147873] *** Warning - bad CRC, using default environment
[0.631527 0.004232]
[0.635980 0.004453] In: serial@ff690000
[0.637982 0.002002] Out: serial@ff690000
[0.640004 0.002022] Err: serial@ff690000
[0.642244 0.002240] Model: Amarula Vyasa-RK3288
[0.644783 0.002539] Net: Net Initialization Skipped
[0.647824 0.003041] No ethernet found.
[0.651954 0.004130] Hit any key to stop autoboot: 0
[0.772503 0.120549] switch to partitions #0, OK
[0.774802 0.002299] mmc1 is current device
[1.069762 0.294960] Scanning mmc 1:1...
[1.312386 0.242624] Found /boot/extlinux/extlinux.conf
[1.315209 0.002823] Retrieving file: /boot/extlinux/extlinux.conf
[1.353460 0.038251] 145 bytes read in 28 ms (4.9 KiB/s)
[1.356412 0.002952] 1: Vyasa Linux-4.13
[1.358237 0.001825] Retrieving file: /boot/uImage
[1.740753 0.382516] 7836344 bytes read in 375 ms (19.9 MiB/s)
[1.744302 0.003549] append: console=ttys2,115200n8 root=/dev/mmcblk0p1 rootwait quiet
[1.750086 0.005784] Retrieving file: /boot/rk3288-vyasa.dtb
[1.786952 0.036866] 36291 bytes read in 28 ms (1.2 MiB/s)
[1.789950 0.002998] ## Booting kernel from Legacy Image at 02000000 ...
[1.794524 0.004574] Image Name: linux-4.13.0-rc4-next-20170810-0
[1.799040 0.004516] Image Type: ARM Linux Kernel Image (uncompressed)
[1.803818 0.004778] Data Size: 7836280 Bytes = 7.5 MiB
[1.807426 0.003608] Load Address: 02000000
[1.809776 0.002350] Entry Point: 02000000
[1.812097 0.002321] Verifying Checksum ... OK
[1.945372 0.133275] ## Flattened Device Tree blob at 01f00000
[1.948106 0.002734] Booting using the fdt blob at 0x1f00000
[1.951715 0.003609] Loading Kernel Image ... OK
[1.968363 0.016648] Loading Device Tree to 0ffffdc2, end 0ffffdc2 ... OK
[1.974709 0.006346]
[1.974911 0.000202] Starting kernel ...
[0.002243 0.002243]
[1.273396 1.271153] [ 0.090111] dmi: Firmware registration failed.
[1.617881 0.344485] [ 0.581926] EXT4-fs (mmcblk0p1): couldn't mount as ext3 due to feature incompatibilities
[1.627592 0.009711] [ 0.592177] EXT4-fs (mmcblk0p1): couldn't mount as ext2 due to feature incompatibilities
[4.596400 2.968808] Starting logging: OK
[4.610216 0.013816] Initializing random number generator... done.
[4.622379 0.012163] Starting network: OK
[4.746329 0.123950]
[4.748163 0.001834] Welcome to VYASA RK3288!
[4.750725 0.002562] vyasa-rk3288 login:
```

```
[0.001135 0.001134] U-Boot TPL 2017.09-rc2-13373-g2cffd0d-dirty (Aug 31 2017 - 20:41:14)
[0.005690 0.004555] Trying to boot from BOOTROM
[0.008221 0.002531] Returning to boot ROM...
[0.196488 0.188267]
[0.196704 0.000216] U-Boot SPL 2017.09-rc2-13373-g2cffd0d-dirty (Aug 31 2017 - 20:41:14)
[0.202759 0.006055] Trying to boot from MMC1
[1.879613 1.676854] [ 0.090151] dmi: Firmware registration failed.
[2.287880 0.408267] [ 0.645755] EXT4-fs (mmcblk0p1): couldn't mount as ext3 due to feature incompatibilities
[2.301935 0.014055] [ 0.660209] EXT4-fs (mmcblk0p1): couldn't mount as ext2 due to feature incompatibilities
[2.425052 0.123117] Starting logging: OK
[2.440868 0.015816] Initializing random number generator... done.
[2.452302 0.011434] Starting network: OK
[2.580451 0.128149]
[2.584354 0.003903] Welcome to VYASA RK3288!
[2.586697 0.002343] vyasa-rk3288 login:
```

EFI boot

- Most of AArch64 and x86 UEFI is default booting system.
- U-Boot support via EFI_LOADER, CMD_BOOTEFI configs.

```
=> load mmc 0:2 ${fdt_addr_r} boot/dtb
29830 bytes read in 14 ms (2 MiB/s)
=> load mmc 0:1 ${kernel_addr_r} efi/debian/grubaa64.efi
reading efi/debian/grubaa64.efi
120832 bytes read in 7 ms (16.5 MiB/s)
=> bootefi ${kernel_addr_r} ${fdt_addr_r}
```

`doc/README.uefi`

ELCE 2017: Marrying U-Boot, UEFI and grub, Alexander Graf

Distro boot

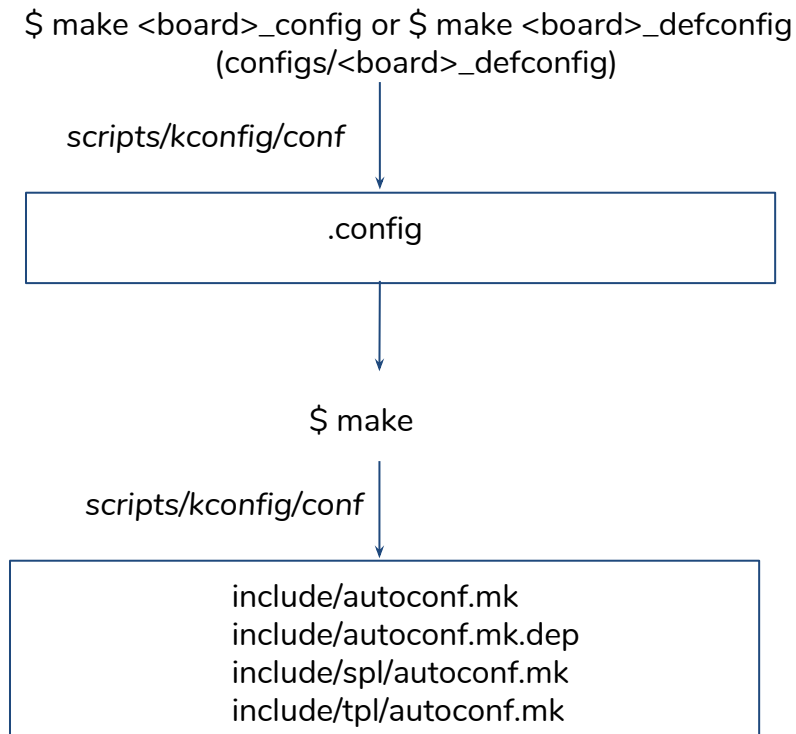
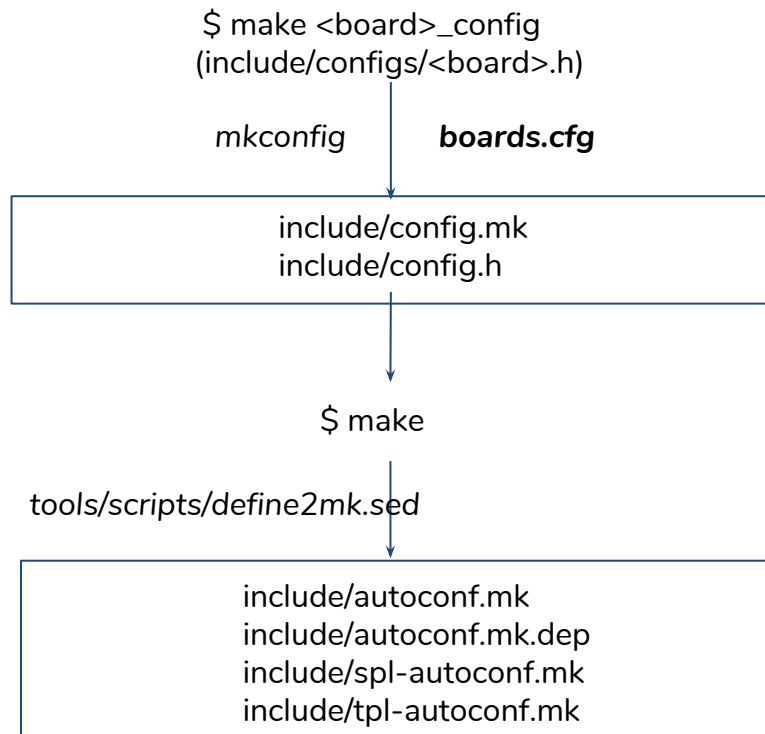
- bootargs - system boot environment commands
- board-specific logic with respect to storage mechanism
- DISTRO_DEFAULTS, generic distribution configuration

```
# u-boot, extlinux.conf
label linux-5.0.0-rc3
    kernel /Image
    devicetree /sun50i-a64-amarula-relic.dtb
    append console=ttyS0,115200 earlyprintk root=/dev/mmcblk0p2 rootwait rw
```


U-Boot Features

- Kconfig
- FDT
- OF livetree
- FDT Overlay
- Driver model
- OF platdata
- DFU

Kconfig



FDT (Flat Device Tree)

- Run-time hardware configuration
- Single U-Boot binary for multiple boards (with board controlled dts)
- Handle via libfdt
- Enabled via CONFIG_OF_CONTROL

- FDT supported U-Boot can build
 - ◆ with default dts, CONFIG_DEFAULT_DEVICE_TREE=<dts-file-name> in defconfig
 - \$ make
 - ◆ with user-specified dts
 - \$ make DEVICE_TREE=<dts-file-name>

- Sample, UniPhier Pro4 reference, Pro4 Ace, Pro4 Sanji boards
 - ◆ \$ make uniphier_v7_defconfig (single configuration)
 - ◆ \$ make DEVICE_TREE=uniphier-pro4-ref
 - ◆ \$ make DEVICE_TREE=uniphier-pro4-ace
 - ◆ \$ make DEVICE_TREE=uniphier-pro4-sanji

- DTB packing during build
 - ◆ CONFIG_OF_EMBED
 - ◆ CONFIG_OF_SEPARATE

FDT, u-boot

- Maintain U-Boot specific node definitions in separate file
- Useful for DT allocation in SPL
- u-boot,dm-pre-alloc, u-boot,dm-spl

```
/* arch/arm/dts/imx6qdl-u-boot.dtsi */  
/ {  
    soc {  
        u-boot,dm-spl;  
        aips-bus@02000000 {  
            u-boot,dm-spl;  
        };  
    };  
};  
  
&gpio1 {  
    u-boot,dm-spl;  
};
```

FDT, libfdt

```
1 int fdt_delprop(void *fdt, int nodeoffset, const char *name)
2 {
3     struct fdt_property *prop;
4     int len, proplen;
5
6     FDT_RW_CHECK_HEADER(fdt);
7
8     prop = fdt_get_property_w(fdt, nodeoffset, name, &len);
9     if (!prop)
10         return len;
11
12     proplen = sizeof(*prop) + FDT_TAGALIGN(len);
13     return fdt_splice_struct(fdt, prop, proplen, 0);
14 }
15
16 int ft_board_setup(void *blob, bd_t *bd)
17 {
18     int nodeoffset;
19
20     nodeoffset = fdt_path_offset(blob, "/soc/aips-bus@02100000/usdhc@02198000");
21
22     return fdt_delprop(blob, nodeoffset, "no-1-8-v");
23 }
```

? add/update, copy large amount

? tree need to rebuilt

? tree traversing is slow

Livetree (Live Device Tree)

- Pointer-based *hierarchical structures*
- Support after relocation
- ofnode , point to either flat tree or livetree
- Enabled via CONFIG_OF_LIVE

```
static int zynq_spi_ofdata_to_platdata(struct udevice *bus)
{
    struct zynq_spi_platdata *plat = bus->platdata;

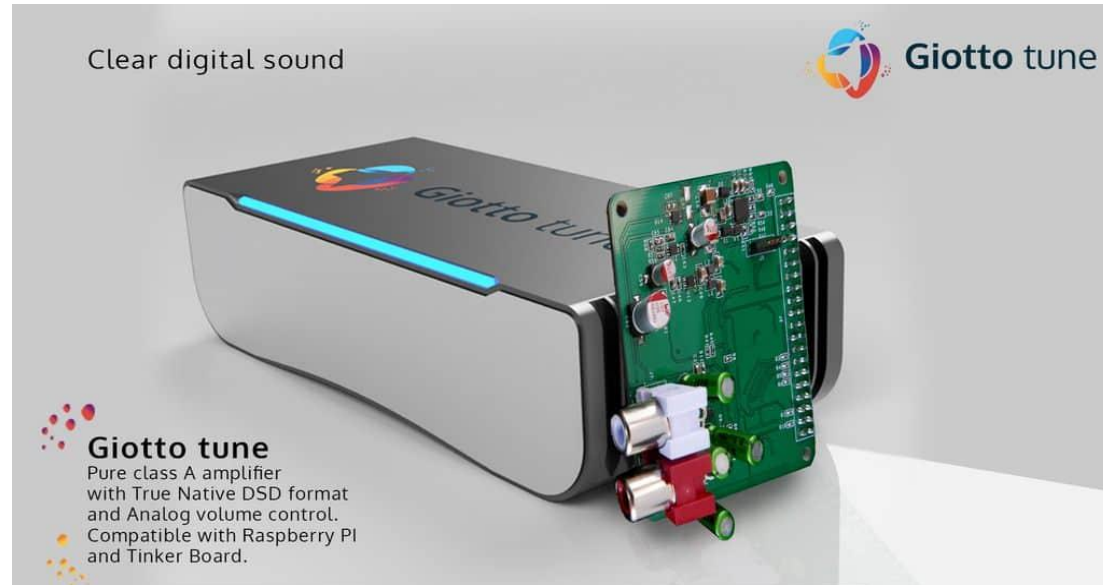
    /* old code */
    plat->regs = (struct zynq_spi_regs *)devfdt_get_addr(bus);
    plat->frequency = fdtdec_get_int(blob, node, "spi-max-frequency",
250000000);

    /* new code */
    plat->regs = (struct zynq_spi_regs *)dev_read_addr(bus);
    plat->frequency = dev_read_u32_default(bus, "spi-max-frequency", 250000000);

    return 0;
}
```

FDT Overlay

- DTO, enable centralize DTB to be overlaid on the device tree.
- Single image of multitude of similar boards and their expansion options
- **HAT** - Raspberry PI, Tinker board
- DTO can load U-Boot via
 - ◆ FIT image
 - ◆ Manual load



```

/dts-v1/;
/ {
    images {
        kernel {
            data = /incbin("./zImage");
            type = "kernel";
            load = <0x10080000>;
            entry = <0x10080000>;
        };
        fdt-1 {
            data = /incbin("./imx6q-icore.dtb");
            type = "flat_dt";
        };
        fdt-2 {
            data = /incbin("./imx6q-icore-mipi.dtb");
            type = "flat_dt";
        };
        configurations {
            default = "imx6q-icore.dtb";
            imx6q-icore.dtb {
                kernel = "kernel";
                fdt = "fdt-1";
            };
            imx6q-icore-mipi.dtb {
                kernel = "kernel";
                fdt = "fdt-2";
            };
        };
    };
};
};

```


FDT Overlay, loading

```
/* via FIT */  
=> bootm $loadaddr#imx6q-icore.dtb#imx6q-icore-mipi  
  
/* Manual load */  
=> setenv fdt_addr 0x18000000  
=> setenv fdt_ovaddr 0x180c0000  
  
=> load mmc 0:1 ${fdt_addr} ${bootdir}/base.dtb  
=> load mmc 0:1 ${fdt_ovaddr} ${bootdir}/overlay.dtb  
  
=> fdt resize 8192  
  
=> fdt apply $fdt_ovaddr  
  
=> bootm ${loadaddr} - ${fdt_addr}
```

U-Boot Driver model

- Driver model
- DM, CPU
- DM, Power
- DM, Core
- DM, Peripherals
- Block layer
- USB layer

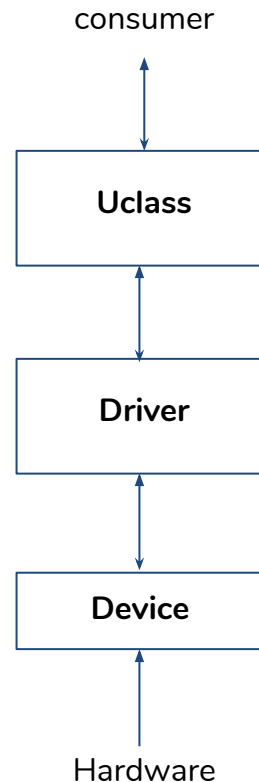
Driver model

- ? ad-hoc model, direct functions call
- ? multiple controllers can't fit same driver
- ? not scalable, difficult to maintain

Simple, scalable, modular, homogeneous

Lazy initialization, but bounded

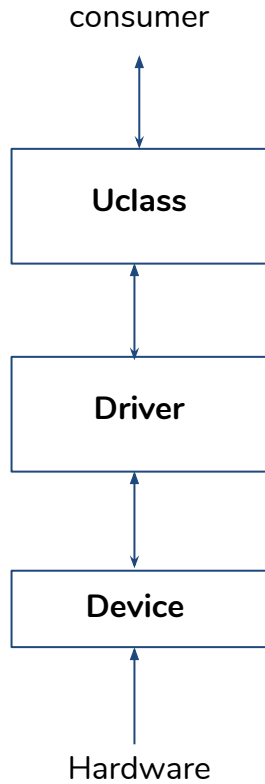
Small overhead, SPL



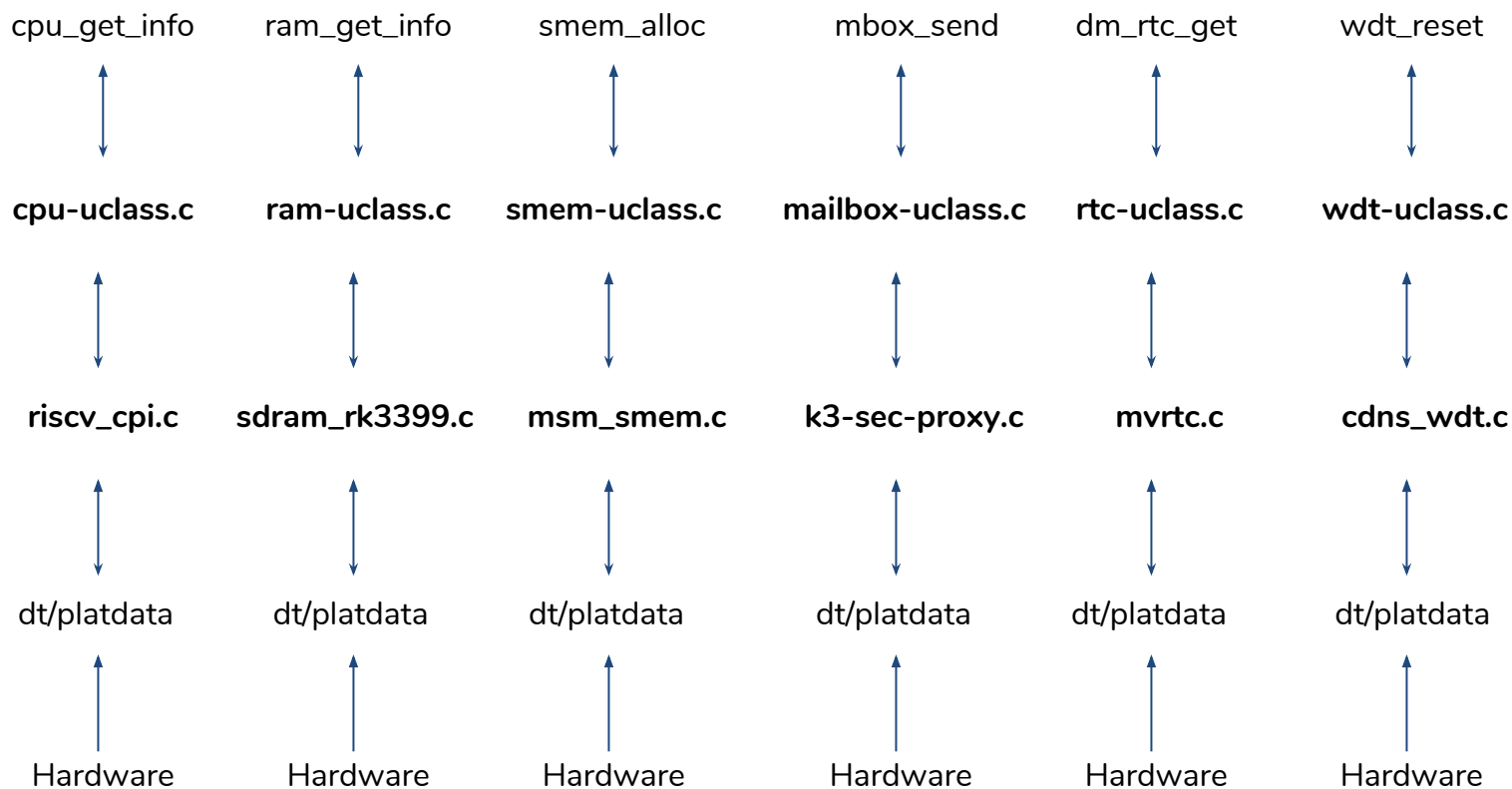
=> dm tree

Class	index	Probed	Driver	Name
root	0	[+]	root_drive	root_driver
clk	0	[]	fixed_rate	-- oscillator
mmc	0	[+]	rockchip_r	-- dwmmc@ff0c0000
blk	0	[+]	mmc_blk	`-- dwmmc@ff0c0000.blk
mmc	1	[+]	rockchip_r	-- dwmmc@ff0f0000
blk	1	[]	mmc_blk	`-- dwmmc@ff0f0000.blk
serial	0	[+]	ns16550_se	-- serial@ff690000
eth	0	[]	gmac_rockc	-- ethernet@ff290000
usb	0	[]	dwc2_usb	-- usb@ff540000
usb	1	[]	dwc2_usb	-- usb@ff580000
ram	0	[]	rockchip_r	-- dmc@ff610000
i2c	0	[+]	i2c_rockch	-- i2c@ff650000
pmic	0	[+]	rk8xx pmic	`-- pmic@1b
regulator	0	[]	rk8xx_buck	-- DCDC_REG1
regulator	1	[]	rk8xx_buck	-- DCDC_REG2
regulator	2	[]	rk8xx_buck	-- DCDC_REG3
regulator	3	[]	rk8xx_buck	-- DCDC_REG4
regulator	4	[]	rk8xx_ldo	-- LDO_REG1

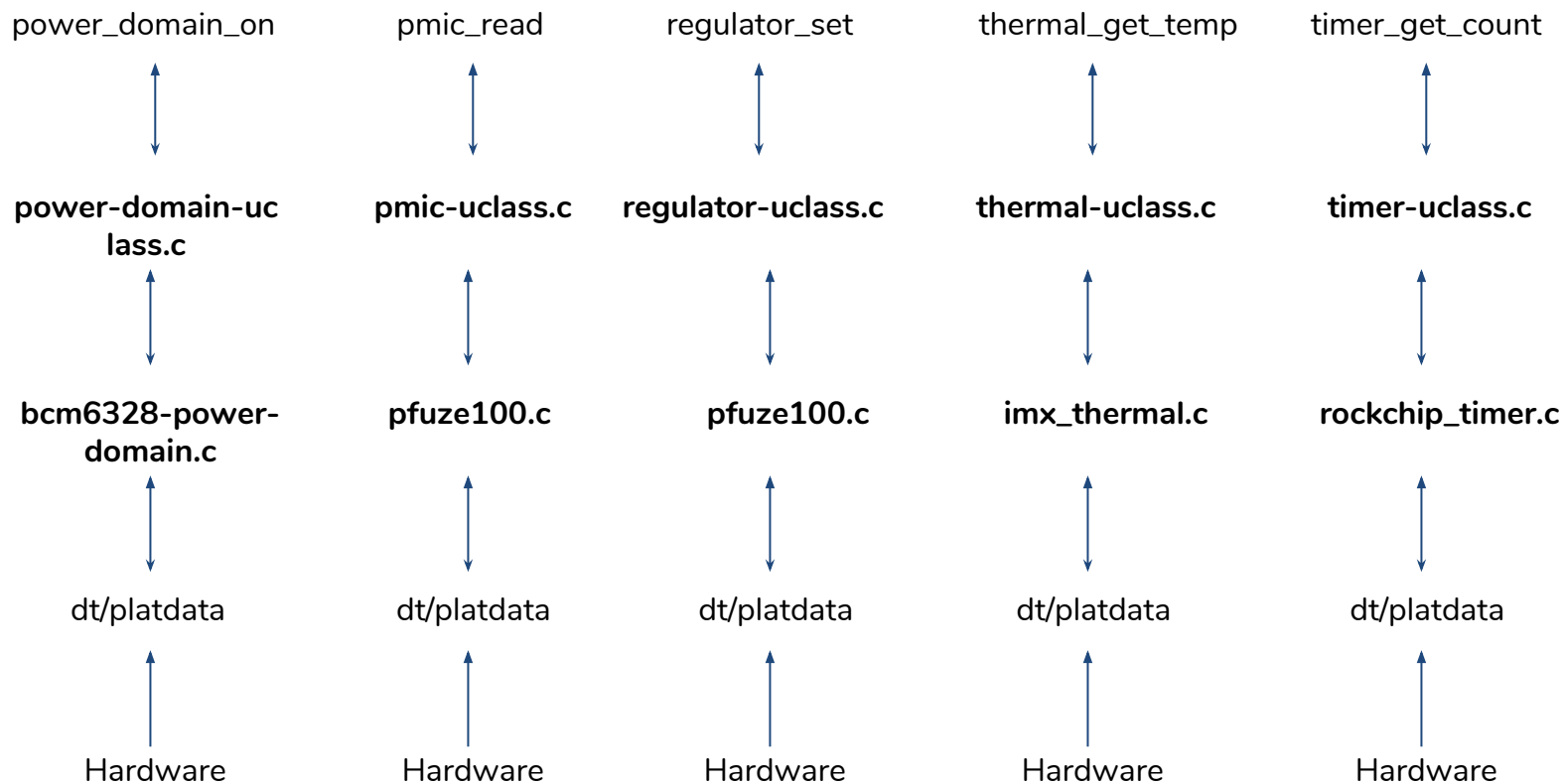
Driver model



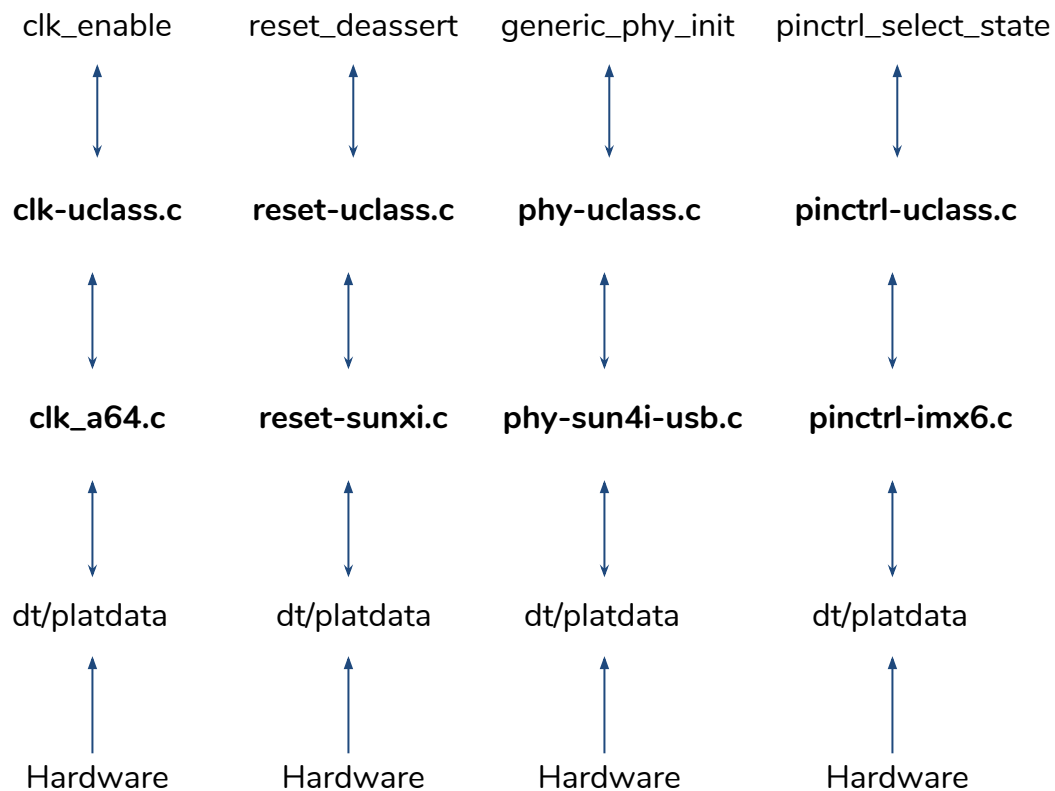
DM, CPU



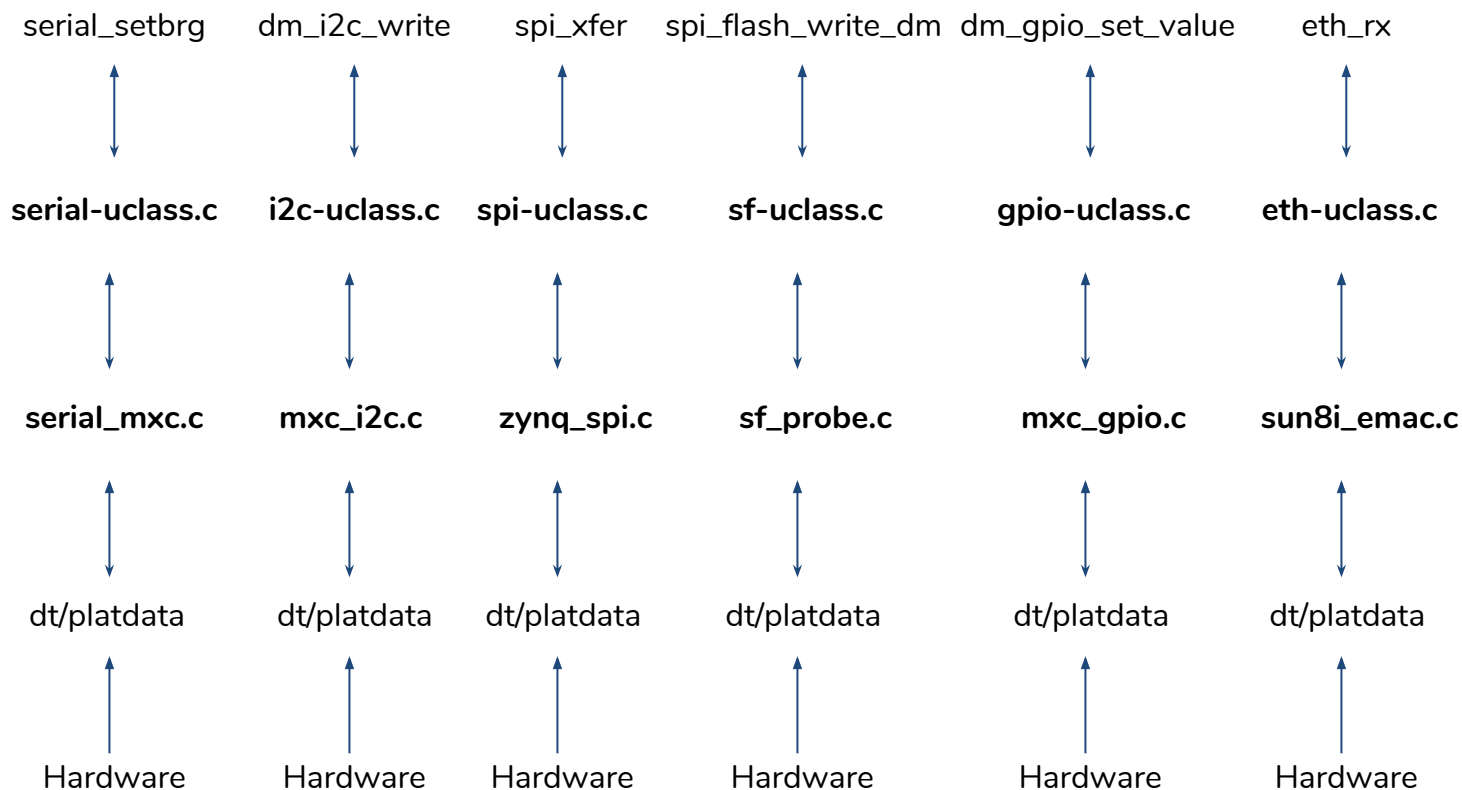
DM, Power



DM, Core



DM, Peripherals



```
UCLASS_DRIVER(spi) = {
    .id          = UCLASS_SPI,
    .name        = "spi",
    .flags       = DM_UC_FLAG_SEQ_ALIAS,
    .post_bind   = dm_scan_fdt_dev,
    .post_probe  = spi_post_probe,
    .child_pre_probe = spi_child_pre_probe,
    .per_device_auto_alloc_size = sizeof(struct dm_spi_bus),
    .per_child_auto_alloc_size = sizeof(struct spi_slave),
    .per_child_platdata_auto_alloc_size = sizeof(struct dm_spi_slave_platdata),
    .child_post_bind = spi_child_post_bind,
};

U_BOOT_DRIVER(zynq_qspi) = {
    .name      = "zynq_qspi",
    .id       = UCLASS_SPI,
    .of_match  = zynq_qspi_ids,
    .ops      = &zynq_qspi_ops,
    .ofdata_to_platdata = zynq_qspi_ofdata_to_platdata,
    .platdata_auto_alloc_size = sizeof(struct zynq_qspi_platdata),
    .priv_auto_alloc_size = sizeof(struct zynq_qspi_priv),
    .probe    = zynq_qspi_probe,
};
```

```

static int sun4i_usb_phy_exit(struct phy *phy)
{
    clk_disable(&usb_phy->clocks);
    reset_assert(&usb_phy->resets);
}

static int sun4i_usb_phy_init(struct phy *phy)
{
    clk_enable(&usb_phy->clocks);
    reset_deassert(&usb_phy->resets);
}

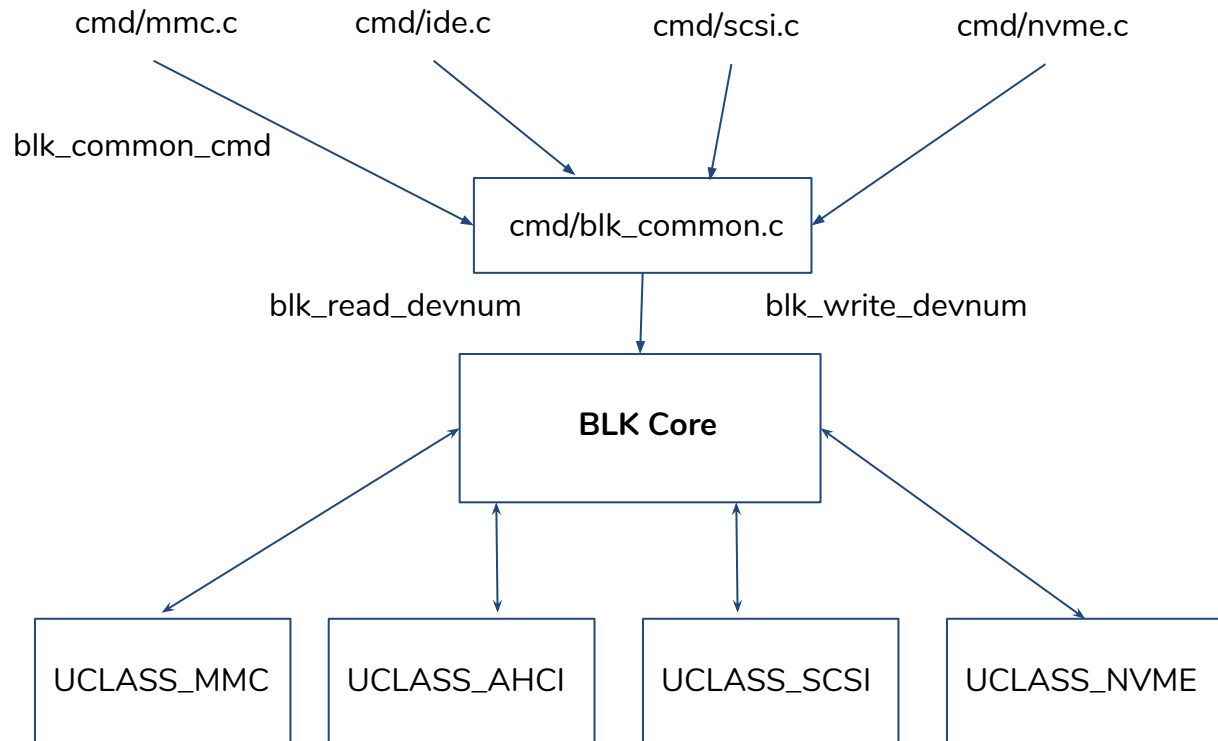
static struct phy_ops sun4i_usb_phy_ops = {
    .init = sun4i_usb_phy_init,
    .exit = sun4i_usb_phy_exit,
};

static int sun4i_usb_phy_probe(struct udevice *dev)
{
    clk_get_by_name(dev, "usb0_phy", &phy->clocks);
    reset_get_by_name(dev, "usb0_reset", &phy->resets);
}

U_BOOT_DRIVER(sun4i_usb_phy) = {
    .id      = UCLASS_PHY,
    .ops     = &sun4i_usb_phy_ops,
    .probe   = sun4i_usb_phy_probe,
};

```

Generic Block Layer



USB framework

- via UCLASS_USB
- uclass core: drivers/usb/host/usb-uclass.c
- platform specific uboot driver: drivers/usb/host/ehci-generic.c
- USB control, bulk, interrupt, create_int_queue etc via dm_usb_ops
- include/usb.h
- cmd/usb.c
- USB Gadgets can probe via Gadget UCLASS
- drivers/usb/gadget/ether.c, USB ETH Gadget
- MUSB can operate Host and Peripheral
- MUSB Host access via UCLASS_USB
- MUSB Peripheral access via
 - ◆ UCLASS_USB_DEV_GENERIC - host devices
 - ◆ UCLASS_USB_GADGET_GENERIC - gadget devices
- drivers/usb/musb-new/sunxi.c, SunXi MUSB driver

OF Platdata

- SPL size increases with FDT
- Enabled via CONFIG_SPL_OF_PLATDATA
- Explicitly define the device details Like legacy platform_device in Linux

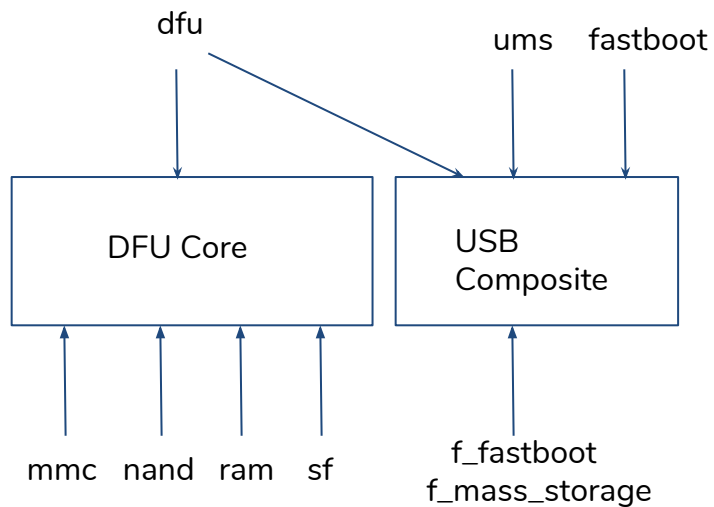
```
#include <dm/platform_data/spi_davinci.h>

static const struct davinci_spi_platdata davinci_spi_data = {
    .regs = (struct davinci_spi_regs *)0x01f0e000,
    .num_cs = 4,
};

U_BOOT_DEVICE(davinci_spi) = {
    .name = "davinci_spi",
    .platdata = &davinci_spi_data,
};
```

Firmware Upgrade

- Upgrade firmware images on running U-Boot
- DFU
- DFU via tftp
- UMS
- Fastboot



Port new hardware

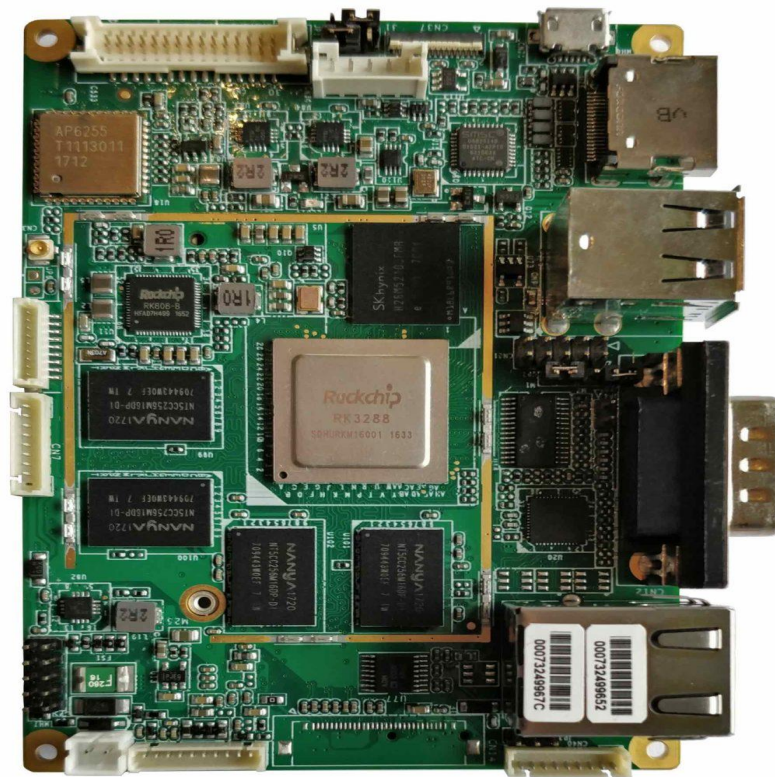
- Prerequisite
- Vyasa RK3288 port

Prerequisite

- SRAM size restrictions
- DDR configuration and timings
- Start with Serial port, can be debuggable further

Vyasa RK3288 port

- 3 stage bootloader
- TPL
 - ◆ CPU, dram init, clocks, debug uart
 - ◆ SPL BOOTROM
- SPL
 - ◆ SPL_OF_CONTROL, UART, falcon
 - ◆ SPL MMC
- U-Boot proper
 - ◆ OF_CONTROL, UART, MMC, I2C,
 - ◆ CLK, Reset, commands etc



- Add SoC support
 - ◆ arch/arm/mach-rockchip
 - ◆ arch/arm/mach-rockchip/rk3288
 - ◆ CPU clock, syscon, linker script etc
 - ◆ Add TARGET_VYASA_RK3288 in arch/arm/mach-rockchip/rk3288/Kconfig
- Add DTS support
 - ◆ arch/arm/dts/rk3288-vyasa.dts
 - ◆ arch/arm/dts/rk3288-vyasa-u-boot.dtsi
- Add Board support
 - ◆ board/amarula/vyasa-rk3288/
 - ◆ Board specific code
 - ◆ Board specific SPL code, if require
 - ◆ board/amarula/vyasa-rk3288/MAINTAINERS
- Add header file
 - ◆ include/configs/vyasa-rk3288.h
 - ◆ Include common useful headers
 - ◆ Distro CONFIG_ definitions
 - ◆ CONFIG_ items which doesn't support Kconfig yet
- Add defconfig file
 - ◆ CONFIG_ items which support Kconfig
- Finally run buildman or travis to make sure all build fine
- And use patman for sending patches to Mainline

U-Boot TPL 2018.09-00097-gd1e15041abf3 (Sep 13 2018 - 15:37:34 +0530)

Trying to boot from BOOTROM

Returning to boot ROM...

U-Boot SPL 2018.09-00097-gd1e15041abf3 (Sep 13 2018 - 15:37:34 +0530)

Trying to boot from MMC1

Expected Linux image is not found. Trying to start U-boot

U-Boot 2018.09-00097-gd1e15041abf3 (Sep 13 2018 - 15:37:34 +0530)

Model: Amarula Vyasa-RK3288

DRAM: 2 GiB

MMC: dwmmc@ff0c0000: 1, dwmmc@ff0f0000: 0

Loading Environment from MMC... *** Warning - bad CRC, using default environment

In: serial

Out: serial

Err: serial

Model: Amarula Vyasa-RK3288

Net: eth0: ethernet@ff290000

Hit any key to stop autoboot: 0

switch to partitions #0, OK

=>

Future plan

- Kconfig migration
- Driver model migrations
 - ◆ BLK, DM_MMC, DM_SCSI, DM_USB
 - ◆ DM_SPI, DM_SPI_FLASH, DM_VIDEO, DM_PCI
- MTD driver model
- Architecture CLK, RESET, Pinctrl Subsystems

Conclusion

- Use DT and DM for new ports
- Hands on with DM conversion
- ML: u-boot@lists.denx.de
- IRC: #u-boot
- Lin: <https://www.linkedin.com/in/jaganteki/>
- HABv4: <https://openedev.amarulasolutions.com/display/ODWIKI/i.MX6+HABv4>
- OP-TEE case study: <https://www.amarulasolutions.com/news/optee-case-study-on-imx6q/>
- Amarula Wiki: <https://openedev.amarulasolutions.com>

Questions??

Thank you

Jagan Teki <jagan@amarulasolutions.com>