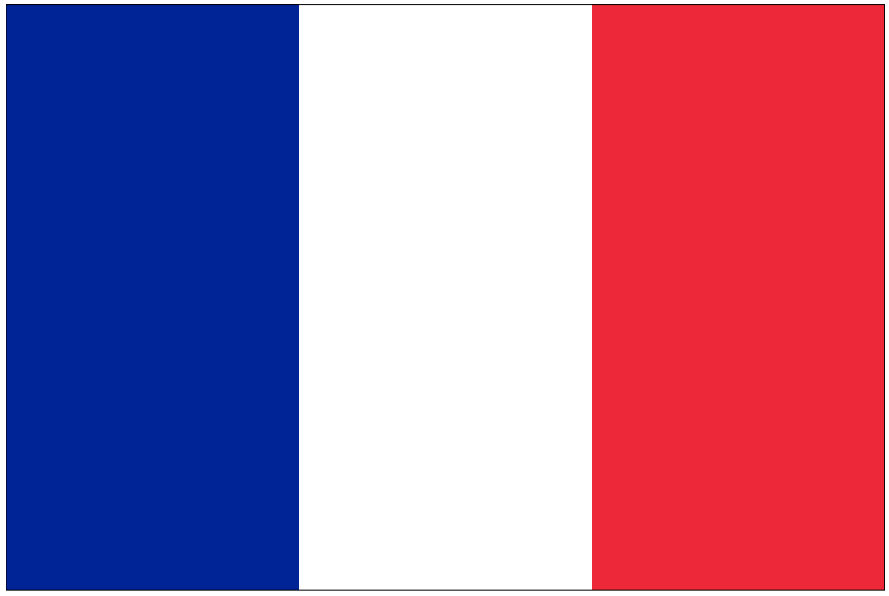
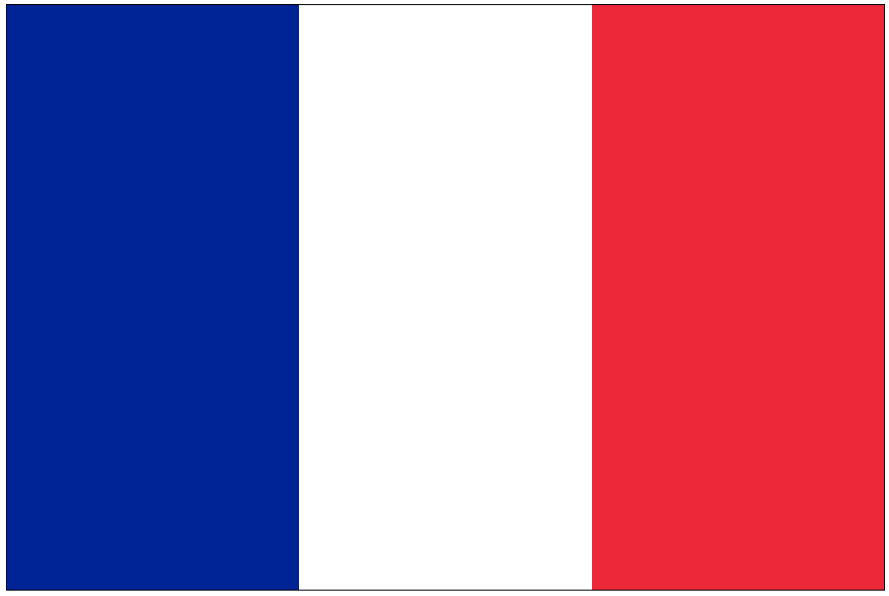
... like real computers - Making distributions work on single board computers

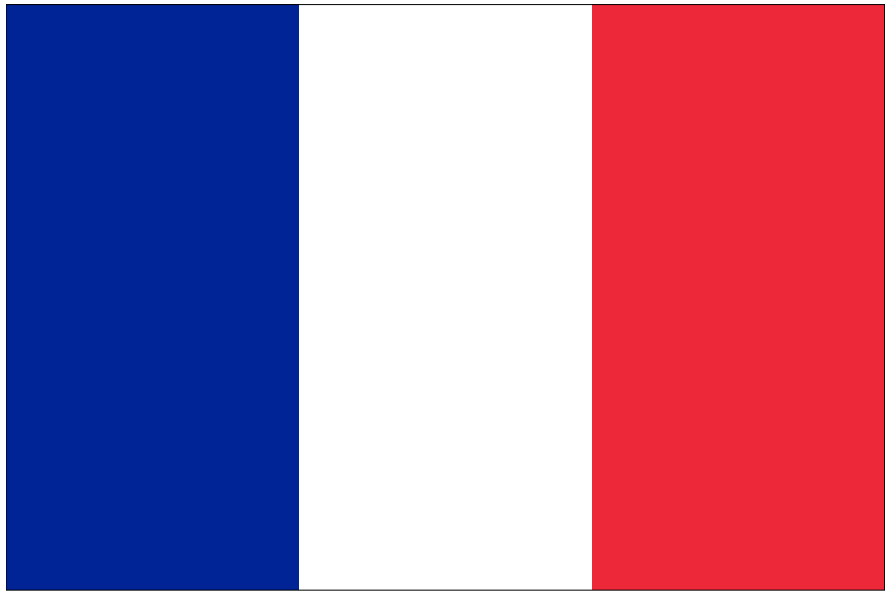
André Przywara

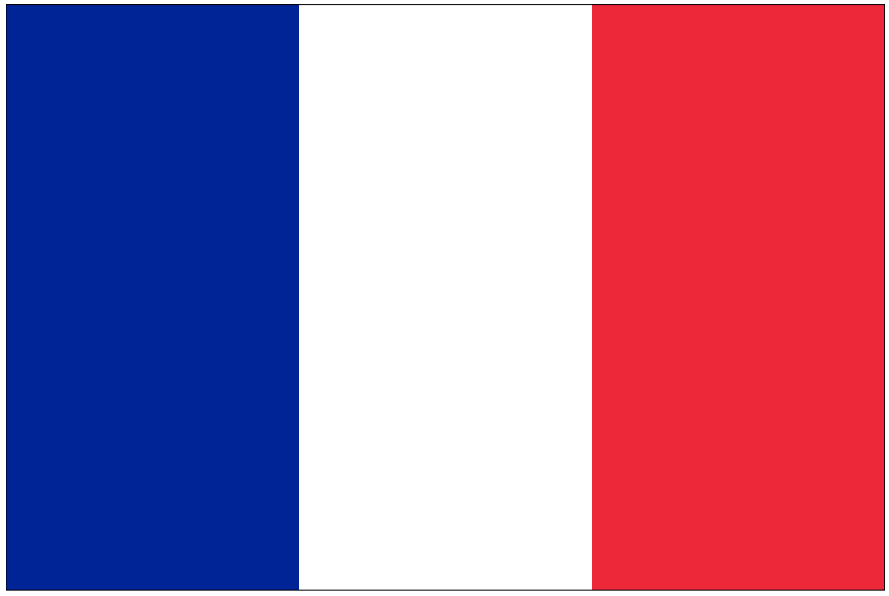
04/02/2018

apritzel@Freenode









Agenda

- Booting
  - Current firmware / boot situation
  - Problems ...
  - ... and how to solve them
- Linux kernel support
  - New SoC in the kernel - why does it take so long?
  - What can we do about it?

Demo?

Glossary / scope

Disclaimer: Not an Arm Ltd. story.

- SBC: single board computer with ARM core, "Fruit-Pis"
  - Not servers!
- SoCs from Allwinner, Rockchip, Amlogic, Marvell, Realtek, ...
- DT: device tree, hardware description, for generic OS support
  - Not ACPI!
- firmware: board-specific low-level software, including boot loader
- Mainline, not BSP.

Current situation

Board	Ubuntu	Debian	SuSE	Fedora	Armbian
Pine64	?	?	✓	✓	✓
BananaPi M64	?	?		✓	✓
NanoPi A64	?	?		✓	✓
Rock64	?	?			✓

Table: Board distribution support

Current situation

Board	Ubuntu	Debian	SuSE	Fedora	Armbian
Pine64	?	?	✓	✓	✓
BananaPi M64	?	?		✓	✓
NanoPi A64	?	?		✓	✓
Rock64	?	?			✓

Table: Board distribution support

Actual technical dependency: kernel support for SoC

What are the main problems?

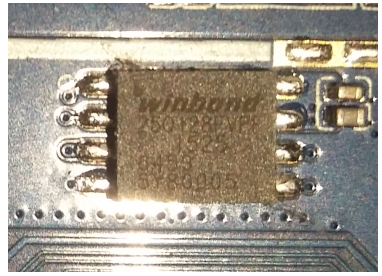
- Traditionally no well recognised standard way of booting
- Many boards come without on-board storage - no firmware!
- Distribution has to ship board DT - explicit board support

How to find and boot the kernel

Could be some U-Boot magic, but better:

- Using the UEFI standard!
- U-Boot implements (parts of) it now (no need for EDK2!)
- Widely recognised and supported
- Most distributions support it anyway (to cover servers)
- Mostly using grub-efi to actually load kernels (and initrds)
- Actually works already with a recent U-Boot!
- Can boot the default arm64 UEFI installer image

SPI flash



- Many SoCs can boot from SPI flash
- 2MB - 16MB chips have small footprint and are cheap
- Allows to keep firmware separate from mass storage (more secure!)
- Allows to boot via network (disk/card-less, via TFTP/PXE)
- Allows to ship firmware with the device - including the DT
- U-Boot supports booting and loading already - via same firmware image

Small chip, but makes a whole difference!

I thought *you* would bring the DT, honey ...

Who provides the DT?

- Mostly comes from the particular (Linux) kernel repository
- Shipped with the kernel
- Gets reviewed and matches the driver support

Who provides the DT?

- Mostly comes from the particular (Linux) kernel repository
- Shipped with the kernel
- Gets reviewed and matches the driver support

But...

- Prevents support for new boards (despite SoC support!)
- Requires upstreaming of board .dts files (latency!)
- Requires every OS to copy those files

Who provides the DT?

- Mostly comes from the particular (Linux) kernel repository
- Shipped with the kernel
- Gets reviewed and matches the driver support

But...

- Prevents support for new boards (despite SoC support!)
- Requires upstreaming of board .dts files (latency!)
- Requires every OS to copy those files
- *Actually...* DT describes the hardware

Who provides the DT?

- Mostly comes from the particular (Linux) kernel repository
- Shipped with the kernel
- Gets reviewed and matches the driver support

But...

- Prevents support for new boards (despite SoC support!)
- Requires upstreaming of board .dts files (latency!)
- Requires every OS to copy those files
- *Actually...* DT describes the hardware
- ... so should come with the hardware!

How it was actually conceived!

Stable Device Trees

- DTB sits in the actual firmware image
- Could be part of U-Boot ($fdtcontroladdr)
- or provided as a separate file in some container (FIP, FIT)
- Pros:
  - Immediate support for new boards (given SoC support)
  - Scales much better: Done *once* for each board.
  - Immediate support for other kernels
- Cons:
  - Requires stable DT bindings!
  - Forward- and backward compatible!
  - Lack of review?

Proposal

Steps (towards world domination)

- Distributions stop shipping board specific images
- Efforts get combined into generic firmware images for boards
- Distribution / OS agnostic! Should boot FreeBSD as well!
- Firmware images ship device tree(s)
- Implement UEFI boot services (possibly using U-Boot)
- Update mechanism to keep components up-to-date (TBD)
- ... ideally can update DTs independently
- Board vendors add SPI flash to their boards (and preload it)

Example: Allwinner A64 boards

Firmware image:

- Two image files: one for LPDDR3 DRAM, one for DDR3 DRAM
- Actual board (.dtb file name stub) is stored in the SPL header (once)
- FIT image contains many .dtbs
- SPL picks proper .dtb by looking at the SPL header
- U-Boot passes .dtb on to the EFI application ($fdtcontroladdr)

Example: Allwinner A64 boards

Firmware image:

- Two image files: one for LPDDR3 DRAM, one for DDR3 DRAM
- Actual board (.dtb file name stub) is stored in the SPL header (once)
- FIT image contains many .dtbs
- SPL picks proper .dtb by looking at the SPL header
- U-Boot passes .dtb on to the EFI application ($fdtcontroladdr)

Firmware update tool:

- `sunxi-fw info -v -i sun50i-a64-ddr3-fw.img`
- `sunxi-fw info -v /dev/sdc`
- `sunxi-fw list-dt-names`
- `sunxi-fw dt-name -n sun50i-a64-bananapi-m64 /dev/sdc`

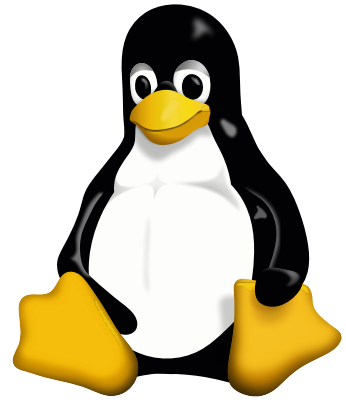
Improve Linux support

Current mainlining approach

- Board reaches developer - ideally early
- Developer mostly copy&pastes some drivers
  - clock driver
  - pinctrl driver
- Adds initial SoC .dtsi and board's .dts to the kernel
- Testing and discussion ...
- Eventually gets merged (into platform tree)
- Eventually gets merged in Linus' tree
- Eventually gets released in vanilla kernel
- Eventually gets picked up by distribution

Takes a decent amount of time: at least 20 weeks to reach mainline

Ways to accelerate kernel support



- Make boards/SoCs/documentation available earlier - ideally one year
- Exploit similar IP blocks in SoCs - more flexible device tree bindings
- Abstract some IP block via firmware interfaces

More flexible device tree bindings

- Avoid deriving too much from compatible string
- Try to design forward-looking bindings
- Describe generic features as properties
  - e.g. number of DMA channels:

```
compatible = "allwinner,sun50i-a64-dma",
                        "allwinner,sun8i-h3-dma";
dma-channels = <8>;
```

- Sent proposal for pincontroller

Use abstracting firmware interfaces

- Some less performance-critical devices might be driven by firmware
- Requires only one generic kernel driver (upstreamed once)
- Hides SoC details in firmware
- Firmware can be developed and deployed much faster
- Examples: ARMs SCPI / SCMI provide:
  - clock support
  - regulator support
  - device power planes (switch on/off devices)
  - DVFS (cpufreq)
  - Sensors (temperature, voltage, current, power, ...)

Proof-of-concept SCPI implementation available in ATF for Allwinner A64

Please help out on ...

- Testing!
- Spread the word!
- Engage in mailing list discussions!
- Testing

Please help out on ...

- Testing!
- Spread the word!
- Engage in mailing list discussions!
- Testing

Help to make SBCs behave more ... like real computers!

Thank You!

References

- http://linux-sunxi.org/
- https://github.com/apritzel/pine64
- https://github.com/apritzel/arm-trusted-firmware
- Freenode: #linux-sunxi