



ARM support in the Linux kernel

Thomas Petazzoni

Bootlin

thomas.petazzoni@bootlin.com





- ▶ CTO and Embedded Linux engineer at Bootlin
 - ▶ Embedded Linux **development**: kernel and driver development, system integration, boot time and power consumption optimization, consulting, etc.
 - ▶ Embedded Linux **training**, Linux driver development training and Android system development training, with materials freely available under a Creative Commons license.
 - ▶ **We're hiring!**
 - ▶ <http://bootlin.com>
- ▶ Contributing the **kernel support for the new Armada 370 and Armada XP** ARM SoCs from Marvell (widely used in NAS devices).
- ▶ Major contributor to **Buildroot**, an open-source, simple and fast embedded Linux build system
- ▶ Living in **Toulouse**, south west of France

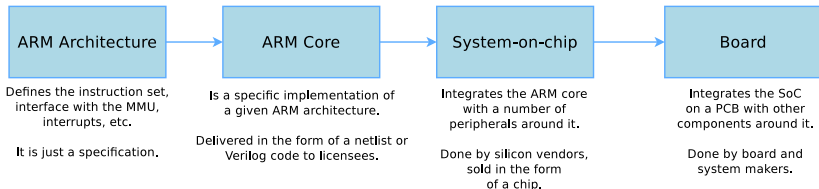


Agenda

- ▶ Background on the ARM architecture and Linux support
- ▶ The problems
- ▶ Changes in the ARM kernel support
- ▶ Getting the support for a SoC in mainline, story of Armada 370/XP

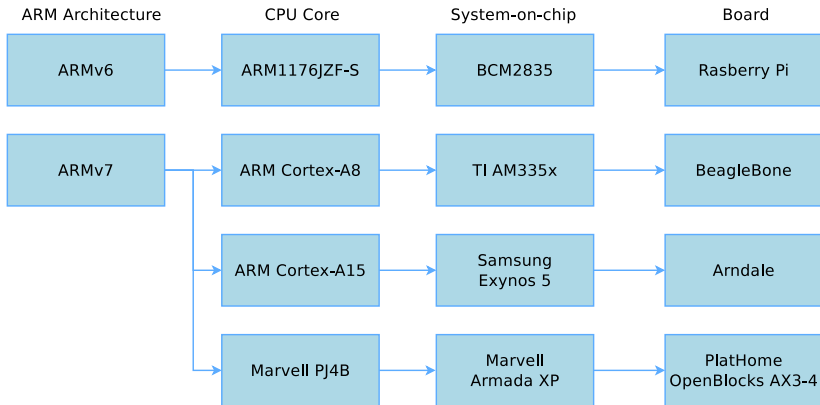


From the ARM architecture to a board



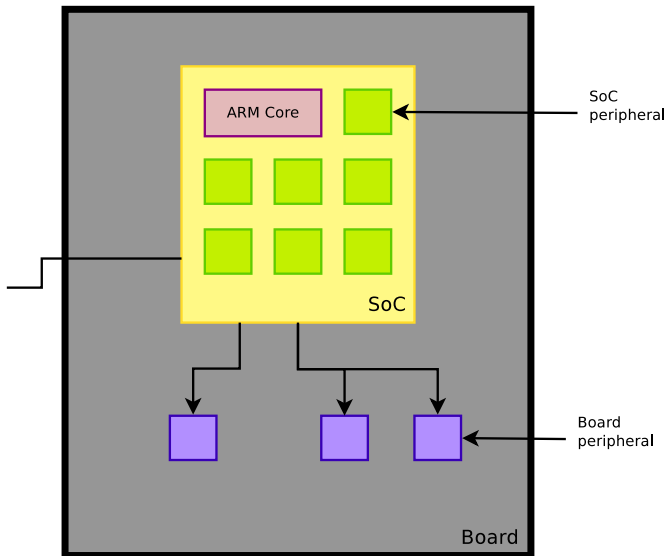


From the ARM architecture to a board, examples





Schematic view of a board





No standardization

- ▶ Beyond the ARM core itself, a lot of freedom is left to the SoC vendor.
- ▶ There is **no standard** for the devices, the management of clocks, pinmuxing, IRQ controllers, timers, etc.
 - ▶ Note: some things like IRQ controllers and timers are now standardized.
- ▶ There **is no mechanism** to enumerate the devices available inside the SoC. All devices have to be known by the kernel.



“Old” ARM code organization in the Linux kernel

- ▶ `arch/arm/`
 - ▶ `arch/arm/{kernel,mm,lib,boot}/`

The core ARM kernel. Contains the code related to the ARM core itself (MMU, interrupts, caches, etc.). Relatively small compared to the SoC-specific code.
 - ▶ `arch/arm/mach-<foo>/`

The SoC-specific code, and board-specific code, for a given SoC family.

 - ▶ `arch/arm/mach-<foo>/board-<bar>.c.`

The board-specific code.
- ▶ `drivers/`

The device drivers themselves.



Issue #1: too much code, lack of review

- ▶ **Exploding number of ARM SoC**, from different vendors
- ▶ The historical maintainer, Russell King, got **overflowed by the amount of code** to review.
- ▶ Code started to flow directly from sub-architecture maintainers directly to Linus Torvalds.
- ▶ Focus of each sub-architecture teams on **their own problems**, no vision of the other sub-architectures.
- ▶ Consequences: lot of code **duplication, missing common infrastructures**, maintainability problems, etc.
- ▶ Linus Torvalds, March 2011: *Gaah. Guys, this whole ARM thing is a f*cking pain in the ass.*



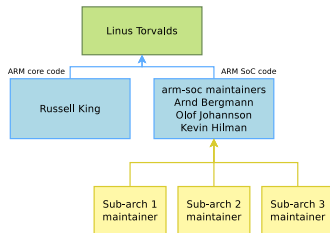
Issue #2: the need for multiplatform kernel

- ▶ On x86 PC, one can build a **single kernel image** (with many modules) that boots and work on all PCs
- ▶ Good for distributions: they can ship a single kernel image.
- ▶ On ARM, it was **impossible to build a single kernel** that would boot on systems using different SoCs.
- ▶ Issue for distributions: they have to build and maintain a kernel image almost for each ARM hardware platform they want to support.
- ▶ Need for **ARM multiplatform support** in the kernel.



Change #1: *arm-soc* and maintainers

- ▶ A new maintainer team for the ARM sub-architectures: **Arnd Bergmann** (Linaro) (currently replaced by Kevin Hilman) and **Olof Johansson** (Google)
- ▶ All the ARM SoC-specific code goes through them, in a tree called **arm-soc**
 - ▶ They send the changes accumulated in arm-soc to Linus Torvalds.
 - ▶ Those maintainers have a **cross-SoC view**: detection of things that should be factorized, consistency across SoC-specific code.
 - ▶ Core ARM changes continue to go through **Russell King**.
 - ▶ Role of the **Linaro** consortium





Change #2: Before the Device Tree... (1)

- ▶ Most devices inside an ARM SoC and on the board cannot be dynamically enumerated: they have to be **statically described**.
- ▶ The old way of doing this description was by using **C code**, registering `platform_device` structures for each hardware device. Each board was identified by a unique *machine ID* passed by the bootloader.
- ▶ This code represented a significant portion of the code in `arch/arm/mach-<foo>`.



Change #2: Before the Device Tree... (2)

From `arch/arm/mach-at91/at91sam9263_devices.c`

```
static struct resource udc_resources[] = {
    [0] = {
        .start = AT91SAM9263_BASE_UDP,
        .end   = AT91SAM9263_BASE_UDP + SZ_16K - 1,
        .flags = IORESOURCE_MEM,
    },
    [1] = {
        .start = NR_IRQS_LEGACY + AT91SAM9263_ID_UDP,
        .end   = NR_IRQS_LEGACY + AT91SAM9263_ID_UDP,
        .flags = IORESOURCE_IRQ,
    },
};

static struct platform_device at91_udc_device = {
    .name           = "at91_udc",
    .id             = -1,
    .dev            = {
        .platform_data = &udc_data,
    },
    .resource       = udc_resources,
    .num_resources  = ARRAY_SIZE(udc_resources),
};

some_init_code() {
    platform_device_register(&at91_udc_device);
}
```



Change #2: Device Tree

- ▶ This has been replaced by a hardware description done in structure separated from the kernel, called the **Device Tree**.
 - ▶ Also used on PowerPC, Microblaze, ARM64, Xtensa, OpenRisc, etc.
 - ▶ Not invented specifically for Linux: was part of the OpenFirmware standard used on PowerPC.
- ▶ The *Device Tree Source*, in text format, gets compiled into a *Device Tree Blob*, in binary format, thanks to the *Device Tree Compiler*.
 - ▶ Sources are stored in `arch/arm/boot/dts`
- ▶ At boot time, the kernel parses the *Device Tree* to instantiate the available devices.
- ▶ Can also be used by other platform software than Linux: the U-Boot and Barebox bootloaders have started using it as well.



Change #2: SoC Device Tree example

```
/include/ "skeleton.dtsi"
/ {
    compatible = "brcm,bcm2835";
    model = "BCM2835";
    interrupt-parent = <&intc>;

    chosen {
        bootargs = "earlyprintk console=ttyAMA0";
    };

    soc {
        compatible = "simple-bus";
        #address-cells = <1>;
        #size-cells = <1>;
        ranges = <0x7e000000 0x20000000 0x02000000>;

        [...]
        intc: interrupt-controller {
            compatible = "brcm,bcm2835-armctrl-ic";
            reg = <0x7e00b200 0x200>;
            interrupt-controller;
            #interrupt-cells = <2>;
        };
        uart@20201000 {
            compatible = "brcm,bcm2835-pl011", "arm,pl011", "arm,primecell";
            reg = <0x7e201000 0x1000>;
            interrupts = <2 25>;
            clock-frequency = <3000000>;
            status = "disabled";
        };
    };
};
```



Change #2: Board Device Tree example

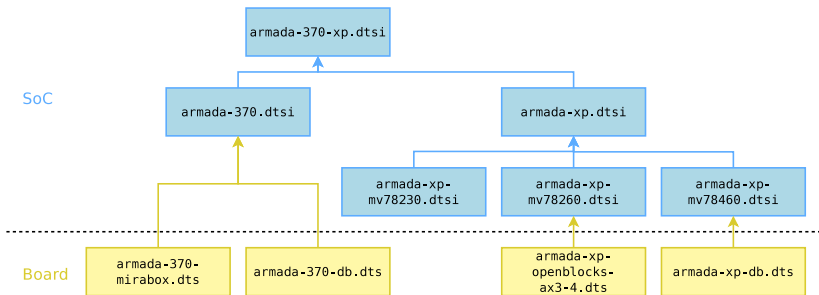
```
/dts-v1/;
/memreserve/ 0x0c000000 0x04000000;
/include/ "bcm2835.dtsi"

/ {
    compatible = "raspberrypi,model-b", "brcm,bcm2835";
    model = "Raspberry Pi Model B";

    memory {
        reg = <0 0x10000000>;
    };
    soc {
        uart@20201000 {
            status = "okay";
        };
    };
};
```



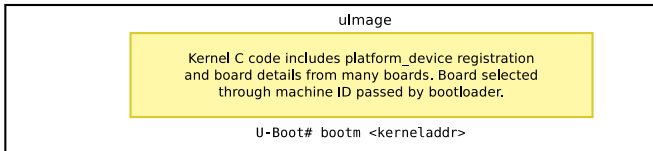

Change #2: Device Tree inheritance



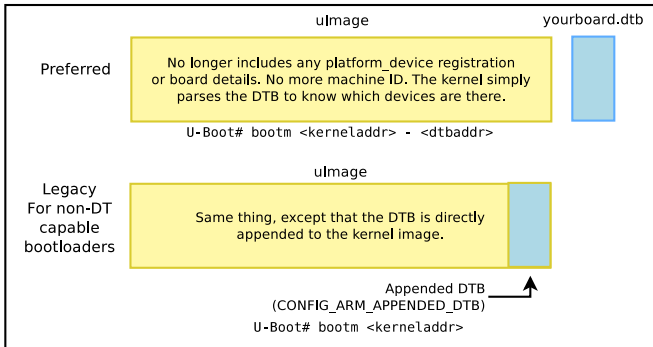


Change #2: Booting with a Device Tree

Without Device Tree



With Device Tree





Change #2: The notion of *DT binding*

- ▶ The idea of the *Device Tree* is that it should be a data structure that **represents the hardware**.
 - ▶ It should not be specific to Linux.
 - ▶ It should not contain *configuration*, but only *hardware description*
- ▶ The *Device Tree* becomes part of the kernel ABI. The kernel must remain capable of using old Device Trees.
- ▶ **Device Tree bindings** are the description of a particular entry of the Device Tree to represent a specific device (or set of devices).
 - ▶ Due the ABI stability requirement, they must be very carefully designed.
 - ▶ A specific mailing list, and a team of maintainers has been assigned to review those bindings.
 - ▶ `Documentation/devicetree/bindings/`
 - ▶ Increase in complexity?

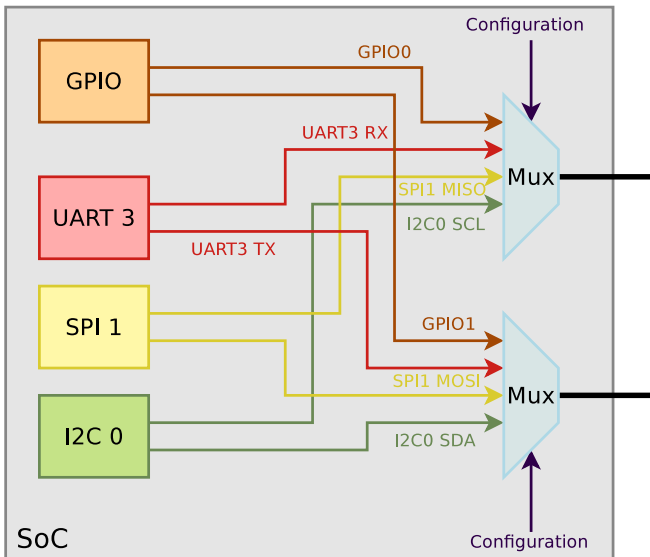


Change #3: Multiplatform kernel

- ▶ Fits the need of distributions willing to build a single kernel image that works on many ARM platforms.
- ▶ The SoC choice now contains a **Allow multiple platforms to be selected** option, and all the SoC families that are compatible with this can be compiled together in the same kernel.
 - ▶ There is still a split between ARMv4/ARMv5 on one side, and ARMv6/ARMv7 on the other side.
- ▶ A **lot of changes** have been done in the ARM kernel to make this possible: avoid two different platforms from defining the same symbol, from using the same header names, no more `#ifdef` but runtime detection instead.
- ▶ The support for all new SoCs **must use the multiplatform** mechanism.



Change #4: Pinctrl subsystem, introduction



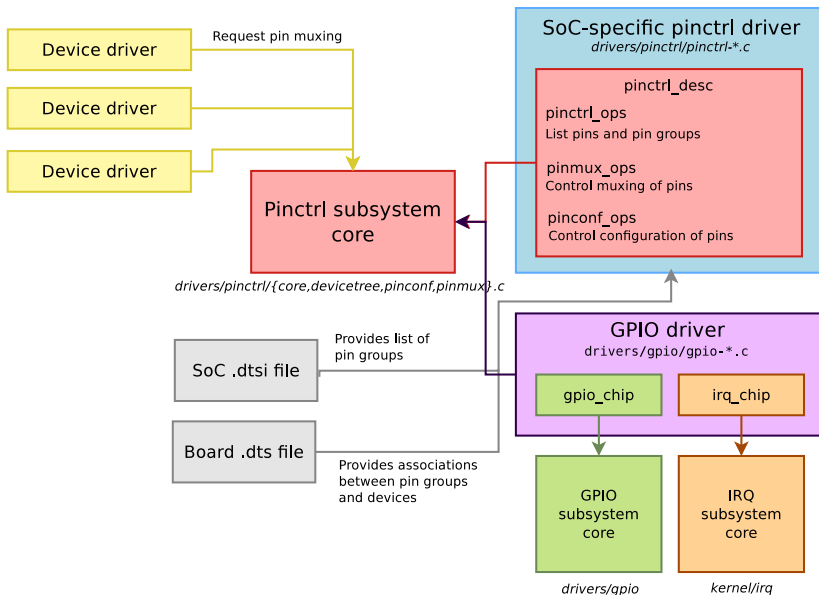


Change #4: Pinctrl subsystem, old code

- ▶ Each ARM sub-architecture had its own pin-muxing code
- ▶ The API was specific to each sub-architecture
- ▶ A lot of similar functionality implemented in different ways
- ▶ The pin-muxing had to be done at the SoC level, and couldn't be requested by device drivers



Change #4: Pinctrl subsystem, new subsystem





Change #5: Clocks

- ▶ In a System-on-Chip, all peripherals are driven by one or more **clocks**.
- ▶ Those clocks are organized in a **tree**, and often are **software configurable**.
- ▶ Since quite some time, the kernel had a simple API: `clk_get`, `clk_enable`, `clk_disable`, `clk_put` that were used by device drivers.
- ▶ Each ARM sub-architecture had its **own implementation** of this API.
- ▶ Does not work for **multiplatform** kernels.
- ▶ Does not allow **code sharing**, and common mechanisms.

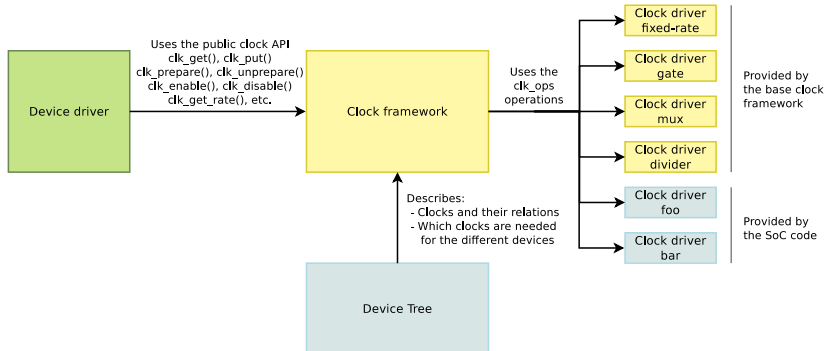


Change #5: Common clock framework

- ▶ A proper **common clock framework** has been added in kernel 3.4, released in May 2012
- ▶ This framework:
 - ▶ Implements the `clk_get`, `clk_put`, `clk_prepare`, `clk_unprepare`, `clk_enable`, `clk_disable`, `clk_get_rate`, etc. **API for usage by device drivers**
 - ▶ Implements **some basic clock drivers** (fixed rate, gatable, divider, fixed factor, etc.) and allows the implementation of **custom clock drivers** using `struct clk_hw` and `struct clk_ops`
 - ▶ Allows to declare the available clocks and their association to devices in the Device Tree (preferred) or statically in the source code (old method)
 - ▶ Provides a *debugfs* representation of the clock tree
 - ▶ Is implemented in `drivers/clock`



Change #5: Common clock framework architecture





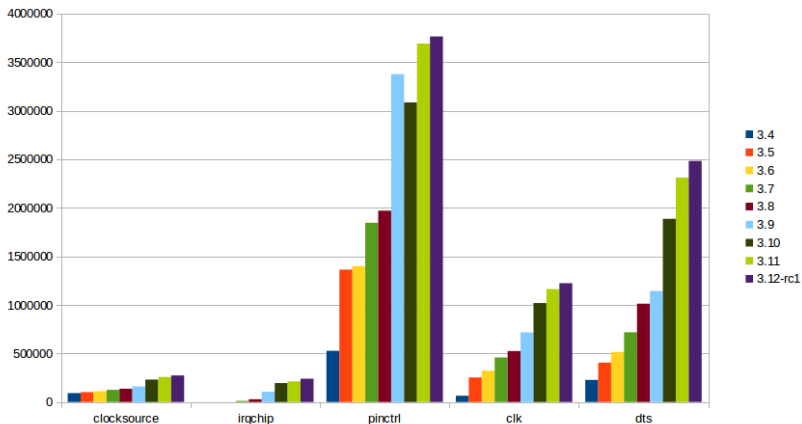
Change #6: More things in drivers/

- ▶ Another goal of the ARM cleanup is to have less code in `arch/arm` and create proper drivers and related infrastructures.
- ▶ For example

IRQ controller drivers	<code>drivers/irqchip/</code>
Timer drivers	<code>drivers/clocksource/</code>
PCI host controller drivers	<code>drivers/pci/host/</code>
Clock drivers	<code>drivers/clk/</code>
Pinmux drivers	<code>drivers/pinctrl/</code>
Memory drivers	<code>drivers/memory/</code>
Bus drivers	<code>drivers/bus/</code>



Adoption rate



Size in bytes of the source code, in the following directories:
drivers/clocksource, drivers/irqchip, drivers/pinctrl, drivers/clk,
arch/arm/boot/dts.



Armada 370/XP, Linux 3.6

Device Tree
(SoC and board)

arch/arm/boot/dts/

Basic
"initialization"
C file
+
basic header
files

arch/arm/mach-mvebu/

Timer
driver

drivers/clocksource/

IRQ controller
driver

drivers/irqchip/

earlyprintk
support

arch/arm/include/debug

Serial port
driver
(already existing
8250 driver)

drivers/tty/serial/

Total: 10 patches



Armada 370/XP, Linux 3.7

Device Tree
(SoC and board)

arch/arm/boot/dts/

Basic
"initialization"
C file
+
basic header
files

arch/arm/mach-mvebu/

Timer
driver

drivers/clocksource/

Pinctrl
driver

drivers/pinctrl/

IRQ controller
driver

drivers/irqchip/

earlyprintk
support

arch/arm/include/debug

Serial port
driver
(already existing
8250 driver)

drivers/tty/serial/

GPIO
driver

drivers/gpio/

Address
decoding
code

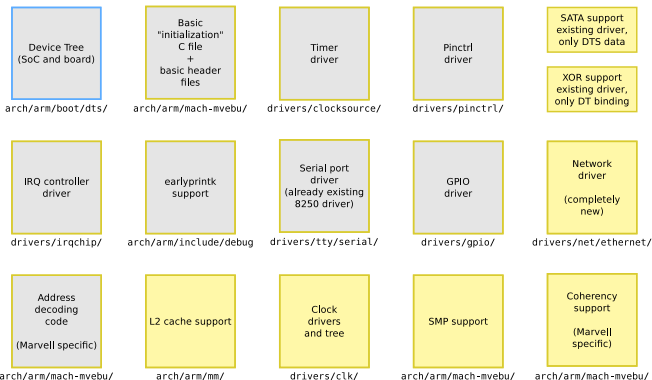
(Marvell specific)

arch/arm/mach-mvebu/

Total: 35 patches



Armada 370/XP, Linux 3.8



Total: 99 patches



Armada 370/XP, Linux 3.9

<p>Device Tree (SoC and board)</p>	<p>Basic "initialization" C file + basic header files</p>	<p>Timer driver</p> <p>Local timer support</p>	<p>Pinctrl driver</p>	<p>SATA support existing driver, only DTS data</p>	<p>SDIO support existing driver, only DT binding</p>
<p>arch/arm/boot/dts/</p>	<p>arch/arm/mach-mvebu/</p>	<p>drivers/clocksource/</p>	<p>drivers/pinctrl/</p>	<p>XOR support existing driver, only DT binding</p>	<p>USB support existing driver, only DTS data</p>
<p>IRQ controller driver</p> <p>IRQ affinity</p>	<p>earlyprintk support</p>	<p>Serial port driver (already existing 8250 driver)</p>	<p>GPIO driver</p>	<p>Network driver (completely new)</p>	<p>RTC support existing driver, only DTS data</p>
<p>drivers/irqchip/</p>	<p>arch/arm/include/debug</p>	<p>drivers/tty/serial/</p>	<p>drivers/gpio/</p>	<p>drivers/net/ethernet/</p>	<p>SPI support existing driver, only DTS data</p>
<p>Address decoding code (Marvell specific)</p>	<p>L2 cache support</p>	<p>Clock drivers and tree</p>	<p>SMP support</p>	<p>Coherency support (Marvell specific)</p>	
<p>arch/arm/mach-mvebu/</p>	<p>arch/arm/mm/</p>	<p>drivers/clk/</p>	<p>arch/arm/mach-mvebu/</p>	<p>arch/arm/mach-mvebu/</p>	

Total: 58 patches



Armada 370/XP, Linux 3.10



Total: 57 patches



Armada 370/XP, Linux 3.11

Device Tree (SoC and board)	Basic "initialization" C file + basic header files	Timer driver Local timer support	Pinctrl driver	SATA support existing driver, only DTS data	SDIO support existing driver, only DT binding
arch/arm/boot/dts/	arch/arm/mach-mvebu/	drivers/clocksource/	drivers/pinctrl/	XOR support existing driver, only DT binding	USB support existing driver, only DTS data
IRQ controller driver IRQ affinity	earlyprintk support	Serial port driver (already existing 8250 driver)	GPIO driver	Network driver (completely new)	RTC support existing driver, only DTS data
drivers/irqchip/	arch/arm/include/debug	drivers/tty/serial/	drivers/gpio/	drivers/net/ethernet/	SPI support existing driver, only DTS data
MBus driver	L2 cache support	Clock drivers and tree	SMP support	Coherency support (Marvell specific)	PCIe driver (new)
drivers/bus/	arch/arm/mm/	drivers/clk/	arch/arm/mach-mvebu/	arch/arm/mach-mvebu/	drivers/pci/host
LPAE support	Thermal driver (new)	Device bus/NOR driver (new)			
arch/arm/mach-mvebu	drivers/thermal	drivers/memory			

Total: 66 patches



Armada 370/XP, Linux 3.12-rc1

<p>Device Tree (SoC and board)</p> <p>arch/arm/boot/dts/</p>	<p>Basic "initialization" C file + basic header files</p> <p>arch/arm/mach-mvebu/</p>	<p>Timer driver</p> <p>Local timer support</p> <p>drivers/clocksource/</p>	<p>Pinctrl driver</p> <p>drivers/pinctrl/</p>	<p>SATA support existing driver, only DTS data</p> <p>XOR support existing driver, only DT binding</p>	<p>SDIO support existing driver, only DT binding</p> <p>USB support existing driver, only DTS data</p>
<p>IRQ controller driver</p> <p>IRQ affinity</p> <p>drivers/irqchip/</p>	<p>earlyprintk support</p> <p>arch/arm/include/debug</p>	<p>Serial port driver (already existing 8250 driver)</p> <p>drivers/tty/serial/</p>	<p>GPIO driver</p> <p>drivers/gpio/</p>	<p>Network driver (completely new)</p> <p>drivers/net/ethernet/</p>	<p>RTC support existing driver, only DTS data</p> <p>SPI support existing driver, only DTS data</p>
<p>MBus driver with DT support</p> <p>drivers/bus/</p>	<p>L2 cache support</p> <p>arch/arm/mm/</p>	<p>Clock drivers and tree</p> <p>drivers/clk/</p>	<p>SMP support</p> <p>arch/arm/mach-mvebu/</p>	<p>Coherency support (Marvell specific)</p> <p>arch/arm/mach-mvebu/</p>	<p>PCIe driver (new)</p> <p>drivers/pci/host</p>
<p>LPAA support</p> <p>arch/arm/mach-mvebu</p>	<p>Thermal driver (new)</p> <p>drivers/thermal</p>	<p>Device bus/NOR driver (new)</p> <p>drivers/memory</p>	<p>NAND support started</p> <p>drivers/mtd/nand</p>	<p>PCIe MSI support started</p> <p>drivers/pci</p>	<p>Big endian support</p>

Total: 92 patches



Getting an ARM SoC in mainline

- ▶ **Throw away the vendor BSP code.** Most likely it is completely crappy. You have to start from scratch.
- ▶ **Start small,** and send code piece by piece. Don't wait to have everything fully working.
- ▶ **Comply with the latest infrastructure changes:** Device Tree, clock framework, pinctrl subsystem. They are mandatory.
- ▶ **Read and post to the LAKML,** Linux ARM Kernel Mailing List
- ▶ **Listen to reviews and comments,** and repost updated versions regularly.
- ▶ **Look at recently merged sub-architectures:** highbank, mvebu, sunxi, bcm2835, socfpga, etc.



And now...

Over the last year, ARM has gone from a constant headache every merge window to an outstanding citizen in the Linux community

Linus Torvalds, August 2012

Questions?

Thomas Petazzoni

`thomas.petazzoni@bootlin.com`

Thanks to Gregory Clement, Ezequiel Garcia (Bootlin, working with me on Marvell mainlining), Lior Amsalem and Maen Suleiman (Marvell)

Slides under CC-BY-SA 3.0

<http://bootlin.com/pub/conferences/2013/kernel-recipes/arm-support-kernel/>