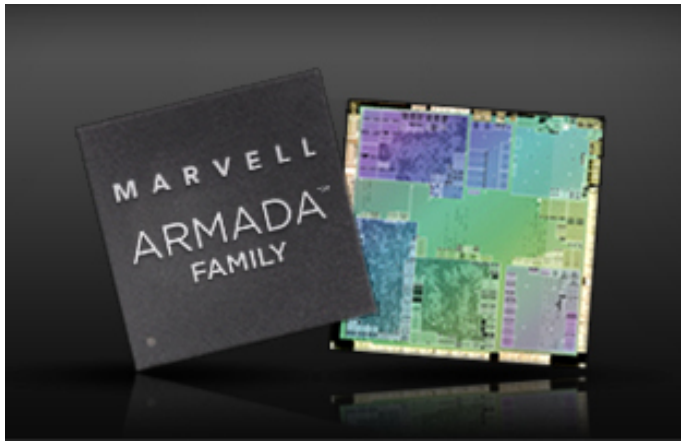
Two years of ARM SoC support mainlining: lessons learned



Thomas Petazzoni
Free Electrons
thomas.petazzoni@free-electrons.com

Thomas Petazzoni

- CTO and Embedded Linux engineer at Free Electrons
  - 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.
  - `http://free-electrons.com`
- Contributions
  - **Kernel support for the Marvell Armada** ARM SoCs from Marvell
  - Major contributor to **Buildroot**, an open-source, simple and fast embedded Linux build system
- Living in **Toulouse**, south west of France

Context

Already supported in mainline when we started

First step

Second step

Orion5x

Feroceon
ARMv5

`arch/arm/mach-orion5x`

Discovery

Feroceon
ARMv5

`arch/arm/mach-mv78xx0`

Armada 370

PJ4B
ARMv7

Armada 375

Cortex-A9
ARMv7 SMP

Kirkwood

Feroceon
ARMv5

`arch/arm/mach-kirkwood`

Dove

PJ4
ARMv7

`arch/arm/mach-dove`

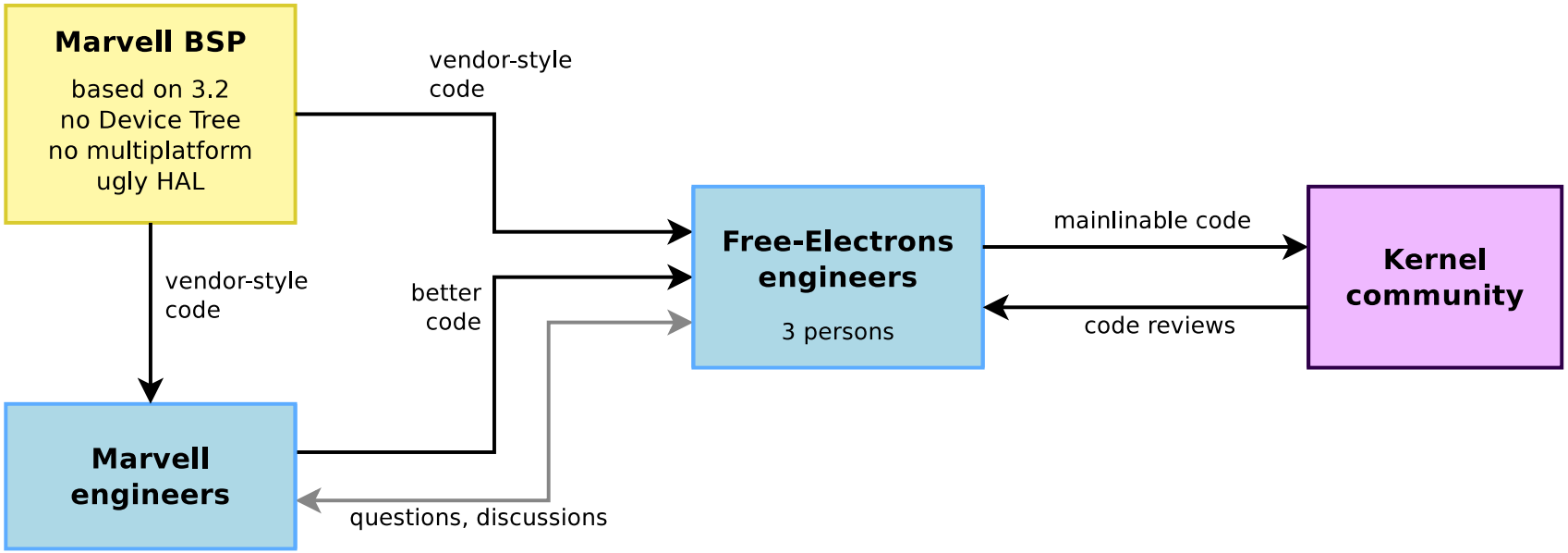
Armada XP

PJ4B-MP
ARMv7 SMP

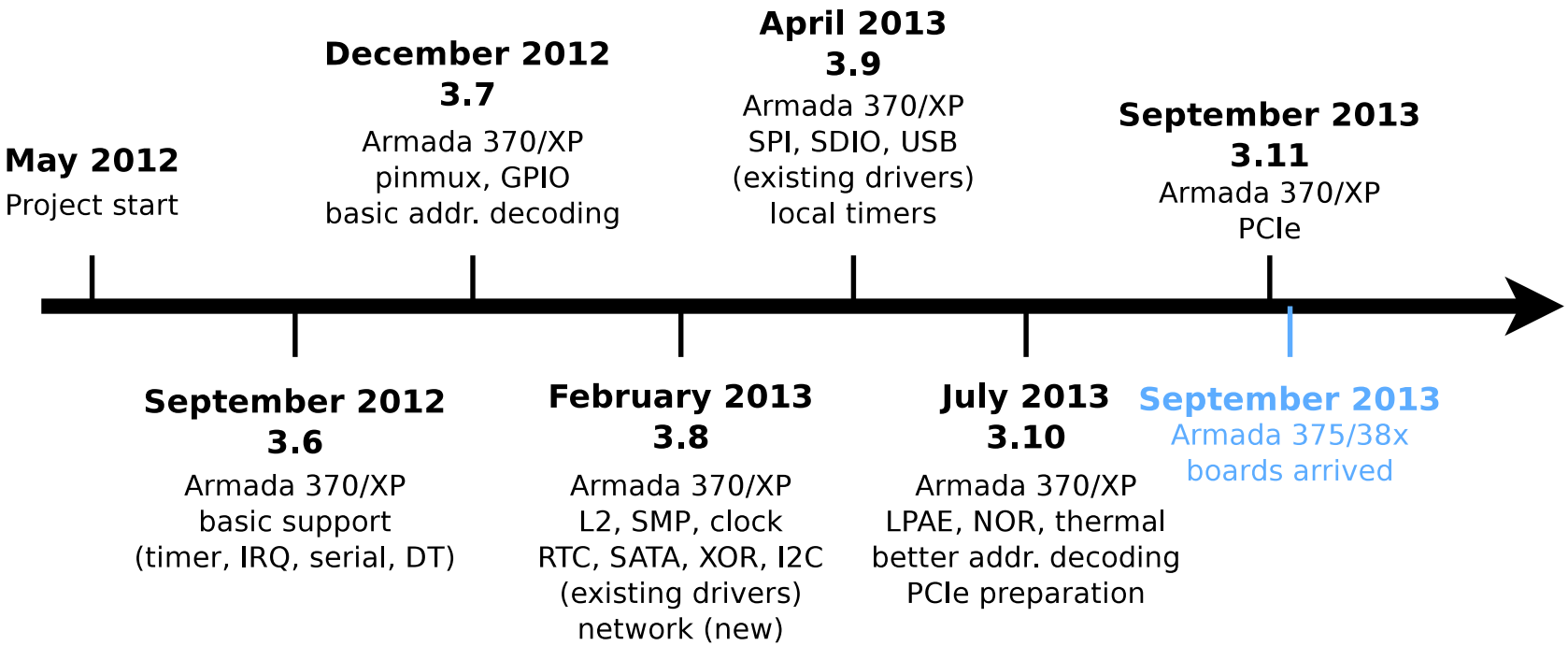
Armada 38x

Cortex-A9
ARMv7 SMP

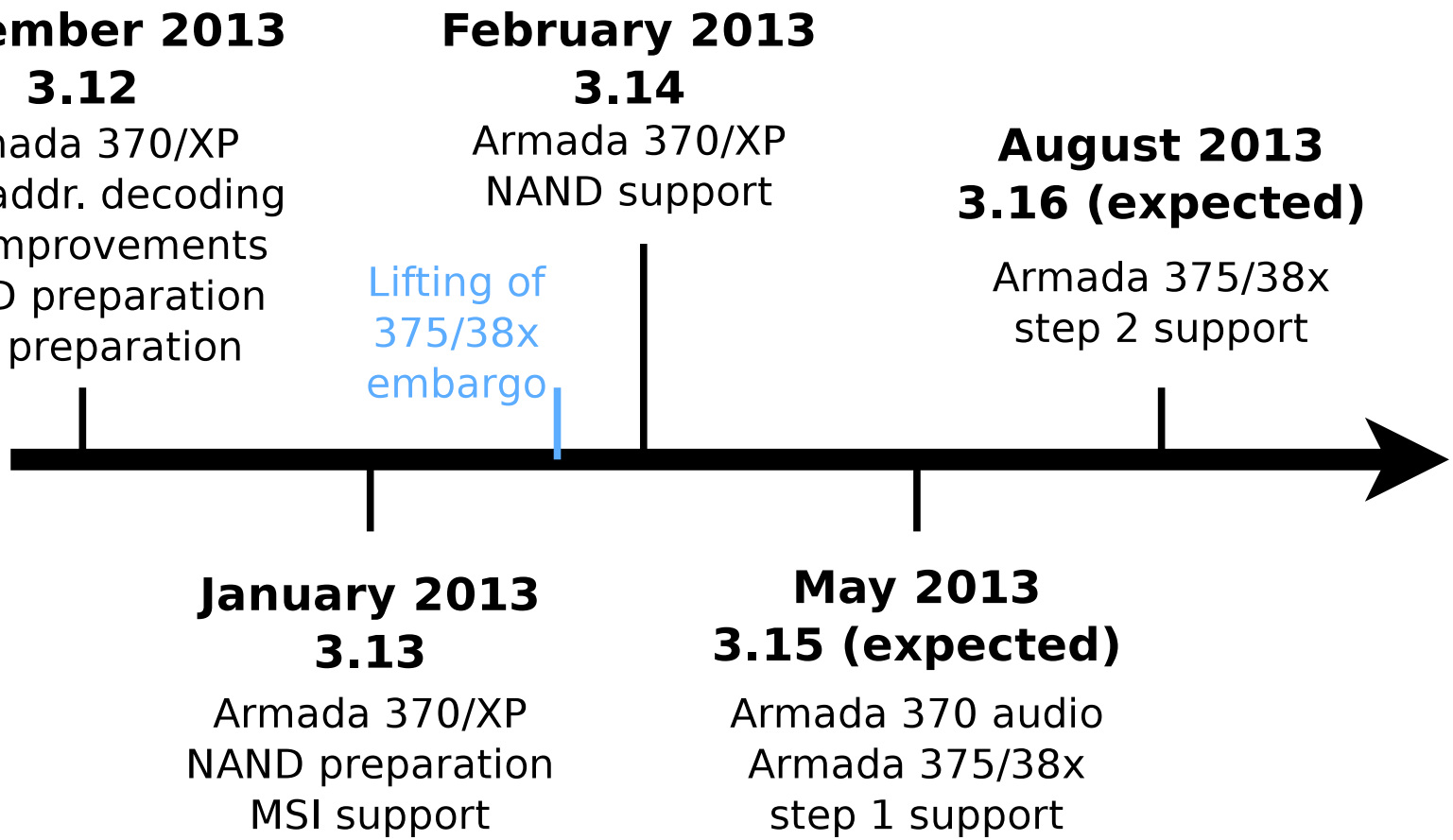
Process


Marvell BSP
based on 3.2
no Device Tree
no multiplatform
ugly HAL
vendor-style code
vendor-style code
better code
Free-Electrons engineers
3 persons
mainlinable code
code reviews
Kernel community
Marvell engineers
questions, discussions

Timeline (1)


May 2012
Project start
December 2012
3.7
Armada 370/XP
pinmux, GPIO
basic addr. decoding
April 2013
3.9
Armada 370/XP
SPI, SDIO, USB
(existing drivers)
local timers
September 2013
3.11
Armada 370/XP
PCIe
September 2012
3.6
Armada 370/XP
basic support
(timer, IRQ, serial, DT)
February 2013
3.8
Armada 370/XP
L2, SMP, clock
RTC, SATA, XOR, I2C
(existing drivers)
network (new)
July 2013
3.10
Armada 370/XP
LPAE, NOR, thermal
better addr. decoding
PCIe preparation
September 2013
Armada 375/38x
boards arrived

Timeline (2)


November 2013
3.12
Armada 370/XP
Final addr. decoding
I2C improvements
NAND preparation
MSI preparation
January 2013
3.13
Armada 370/XP
NAND preparation
MSI support
Lifting of
375/38x
embargo
February 2013
3.14
Armada 370/XP
NAND support
May 2013
3.15 (expected)
Armada 370 audio
Armada 375/38x
step 1 support
August 2013
3.16 (expected)
Armada 375/38x
step 2 support

Lesson #0

Submit early

Submit early

- *Release early, release often*
- Translated into the kernel contributor position: **submit early**
- You **will** make incorrect choices in your patches
- The only solution to know it is to post them and get reviews and comments
- On several occasions, we had too much internal discussions and reviews before posting, and we wasted too much time.

Lesson #1

Engage with the community

Community

- Really **embrace** the power of the community
- Test `linux-next` and `-rc`
  - Allows to find regressions early, and make sure your platform support is in a minimally working state at all times.
  - Much better than big jumps every 2/3 years!
- Provide boards for the **ARM board farm**
  - Talk with Kevin Hilman and Olof Johansson
  - Gets the latest mainline and linux-next built and booted on your platform every day!
- Create a **good relationship** with your sub-architecture maintainer, and possibly other maintainers.
  - They are the key to getting your patches merged.
  - Attend conferences, and plan a beer/drink budget.

Lesson #2

Encourage community contributions

Community contributions

- While the community may not necessarily work on “big” features, people can **help dramatically** in areas like: debugging, bug fixing, performance optimizations, etc.
- Provide **boards and datasheets** to a good selection of people, and they will solve problems
  - Unfortunately Marvell still doesn't provide public datasheets for Armada 370/XP
- Provide those contributors support/assistance.
- **Examples**:
  - Willy Tarreau debugged and fixed a major performance issue in the Armada 370/XP network driver.
  - Neil Greatorex, Jason Gunthorpe and Willy Tarreau investigated and fixed a number of PCIe related issues.
  - Help solving communication issues: special PCI terminology
  - Quick build fixes

Lesson #3

Review patches from others

Review patches from others

- Review patch from other developers. Not the ones in your company, people from other companies.
- Doing this helps the maintainers
- Shows that you care about the community and understand how it works
- Of course, do it wisely, and don't do stupid reviews!
- Ezequiel's evil plan
  - *Statistically speaking, you can speed-up your own reviewing process by reviewing other patches on the same queue*

Lesson #4

Assign dedicated engineering resources

Engineering resources

- Mainlining is a **time consuming process**
- Have a small team of engineers fully dedicated to mainlining.
  - Keep the **team small and efficient**, throwing more people will not necessarily make things go faster.
  - Half of the work is technical, half of the work is social.
- Make sure this small team has easy access to other engineers with deep knowledge of the hardware.
  - The datasheet often isn't enough.
- The engineers need to be **able to reply quickly** to community comments and requests.
  - Otherwise the community won't trust that you will fix issue and maintain the code moving forward.

Lesson #5

Take into account older SOCs

Take into account existing code

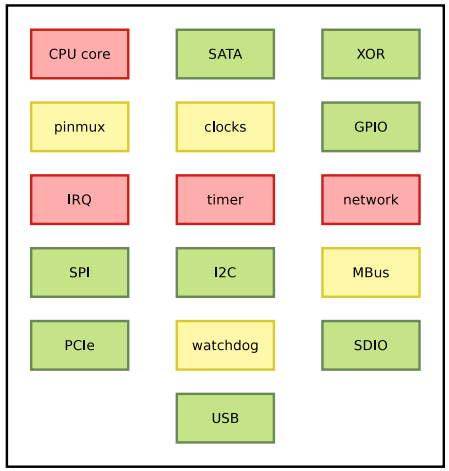
- Yes, you want to push the support for your new SoC / hardware *right now*
- But nothing **upsets more the community than neglecting existing hardware support** in the kernel.
- In our case:
  - We wanted to push the support for Armada 370/XP, sharing a lot of HW blocks with previous SOCs
  - We had to take into account Kirkwood, Discovery, Orion5x and Dove when doing changes to core drivers (pinctrl, clock, mbus, PCIe, etc.)
  - Help of the community is key here!
- If you care today for older code, you will care tomorrow for code you're submitting today.
- Also avoids carrying legacy code in your platform.

Lesson #6

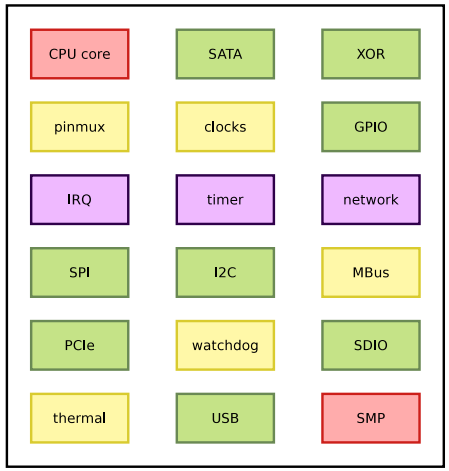
Code re-use actually work

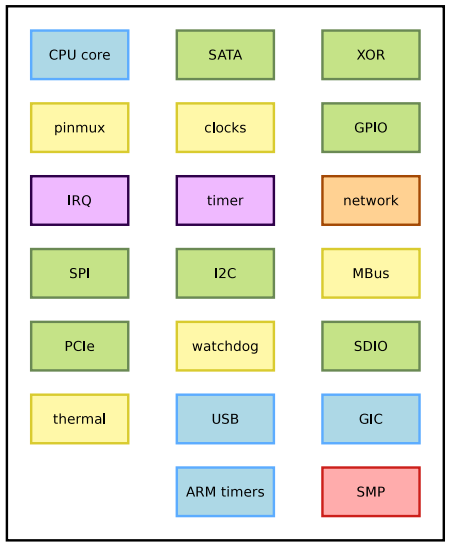
Code re-use actually work

Kirkwood

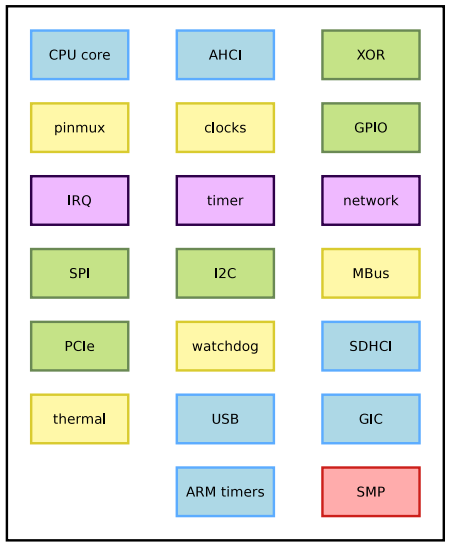


Armada XP





Armada 375



Armada 38x

Code re-use actually work

- Two successive groups of SOCs:
  1. Armada 370/XP
  2. Armada 375/38x
- Many drivers pre-existed from the Kirkwood/Orion era, and we could re-use them with just a DT binding addition.
- Lot of work done Armada 370/XP: writing (hopefully) good Device Tree bindings, proper pinctrl, clock, irqchip drivers, etc.
- Paid off when Armada 375/38x were introduced: all the identical HW blocks were enabled really quickly.
- Mainlining is a long term investment, but it pays off, especially if you have related SoCs.

Lesson #7

Adopt the new code

Adopt the new code

- The SoC vendor should really adopt the code that has been mainlined.
- There is a big risk of a split between:
  - the SoC vendor custom BSP: ugly code, but lots of QA
  - mainline: beautiful code, but poor QA
- In our case, mainlining effort from 3.6 to 3.12, Marvell adopted 3.10 + several backports early January 2014
- It was already too late: when they started testing, we discovered several issues thanks to their stronger QA effort.
- Also, a split means that there is a duplicated debugging effort.

Lesson #8

Realize there is a culture difference

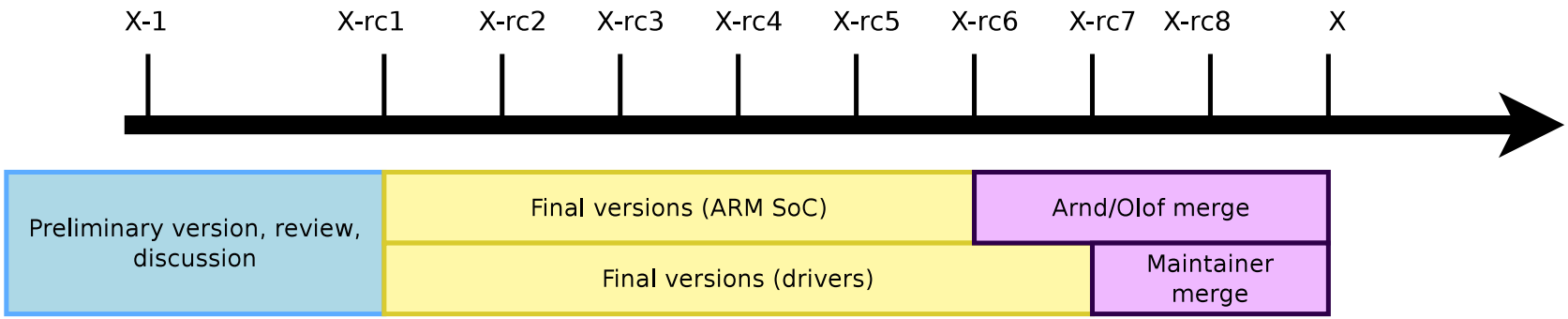
Culture difference

- Initially, surprised by the need of a third-party to mainline Marvell SOCs: Marvell has highly skilled engineers
- However, the **mainlining process is not (only?) about making code work**
- It's about
  - **Making code pretty**: use the appropriate subsystems, don't hack core code in a non-generic way
  - Caring about **other platforms**
  - Understanding how the **social interactions** with maintainers and the community work
- As said earlier: **half technical, half social**
- Need people having both an understanding of the community, and technical expertise. They may not be the deepest technical experts, but they know how to create the interface with the community.

Lesson #9

Know how to schedule patch submission

Patch submission timeline



Patch submission scheduling

- The maintainers/community has a bounded **absorption rate**
- **Don't send too many patches/features** during the same cycle
- **Complex things posted way before the cycle starts**, and be almost in the final stages when the `-rc1` of the previous cycle is released.
- You're **always two releases ahead**: version `X` has just been released, your stuff for release `X+1` should be ready since some time, and your active development is for `X+2`
- During a cycle, you typically
  - Post the final versions of your patches for `X+1`, do the remaining polishing and quick iterations to review/comments.
  - Develop the features for `X+2`, post `RFC`-level patch series.
- **Accelerate** patch submission as you approach merging.

Lesson #10

Merging special DT bindings is hard

Special DT bindings

- Bindings for simple, normal devices, are usually relatively easy to get merged.
- Binding for more complex devices, or busses, require **much more** discussion
- Armada 370/XP was the first ARM platform to merge a PCI host controller driver with a DT binding.
  - Initial patch proposed on December 7th, 2012.
  - 10 iterations until May 16th, 2013.
  - Finally released as part of 3.11, September 2013.
- Similar story for `mvebu-mbus`, the driver for the flexible MBus Marvell bus.
- Be patient, take comments into consideration, explain how your hardware works.

Special DT bindings

```
pcie-controller {
        compatible = "marvell,armada-xp-pcie";
        #address-cells = <3>;
        #size-cells = <2>;
        msi-parent = <&mpic>;
        bus-range = <0x00 0xff>;
        ranges =
               <0x82000000 0 0x40000 MBUS_ID(0xf0, 0x01) 0x40000 0 0x00002000   /* Port 0.0 registers */
                0x82000000 0 0x42000 MBUS_ID(0xf0, 0x01) 0x42000 0 0x00002000   /* Port 2.0 registers */
                0x82000000 0 0x44000 MBUS_ID(0xf0, 0x01) 0x44000 0 0x00002000   /* Port 0.1 registers */
                0x82000000 0 0x48000 MBUS_ID(0xf0, 0x01) 0x48000 0 0x00002000   /* Port 0.2 registers */
                0x82000000 0 0x4c000 MBUS_ID(0xf0, 0x01) 0x4c000 0 0x00002000   /* Port 0.3 registers */
                0x82000000 0 0x80000 MBUS_ID(0xf0, 0x01) 0x80000 0 0x00002000   /* Port 1.0 registers */
                0x82000000 0 0x82000 MBUS_ID(0xf0, 0x01) 0x82000 0 0x00002000   /* Port 3.0 registers */
                [...]
                0x82000000 0x1 0     MBUS_ID(0x04, 0xe8) 0 1 0 /* Port 0.0 MEM */
                0x81000000 0x1 0     MBUS_ID(0x04, 0xe0) 0 1 0 /* Port 0.0 IO  */
                0x82000000 0x2 0     MBUS_ID(0x04, 0xd8) 0 1 0 /* Port 0.1 MEM */
                0x81000000 0x2 0     MBUS_ID(0x04, 0xd0) 0 1 0 /* Port 0.1 IO  */
                0x82000000 0x3 0     MBUS_ID(0x04, 0xb8) 0 1 0 /* Port 0.2 MEM */
                0x81000000 0x3 0     MBUS_ID(0x04, 0xb0) 0 1 0 /* Port 0.2 IO  */
                0x82000000 0x4 0     MBUS_ID(0x04, 0x78) 0 1 0 /* Port 0.3 MEM */
                0x81000000 0x4 0     MBUS_ID(0x04, 0x70) 0 1 0 /* Port 0.3 IO  */
                [...]
        pcie@1,0 {
                device_type = "pci";
                assigned-addresses = <0x82000800 0 0x40000 0 0x2000>;
                reg = <0x0800 0 0 0 0>;
                #address-cells = <3>;
                #size-cells = <2>;
                #interrupt-cells = <1>;
                ranges = <0x82000000 0 0 0x82000000 0x1 0 1 0
                          0x81000000 0 0 0x81000000 0x1 0 1 0>;
                interrupt-map-mask = <0 0 0 0>;
                interrupt-map = <0 0 0 0 &mpic 58>;
                marvell,pcie-port = <0>;
                marvell,pcie-lane = <0>;
                clocks = <&gateclk 5>;
                status = "disabled";
        };
```

Lesson #11

Keeping DT stability is hard

Keeping DT stability is hard

- Some hardware blocks have well-defined functions and boundaries, generally for devices
- But some hardware blocks have less well-defined boundaries, generally core components (clocks, power management, etc.)
- The need for stable DT bindings pretty much requires a complete understanding of how all the hardware works...
- ... which goes a bit against the principle of iterative development.

Keeping DT stability is hard: example (1)

- For SMP enabling on Armada XP, we needed to fiddle with an unit called the PMSU (Power Management Service Unit) and some CPU reset registers.
- So, we did a DT binding like this:

```
armada-370-xp-pmsu@22000 {
        compatible = "marvell,armada-370-xp-pmsu";
        reg = <0x22100 0x400>, <0x20800 0x20>;
};
```

- First register region: PMSU
- Second register region: CPU reset registers

Keeping DT stability is hard: example (1)

A year later, we realized that:

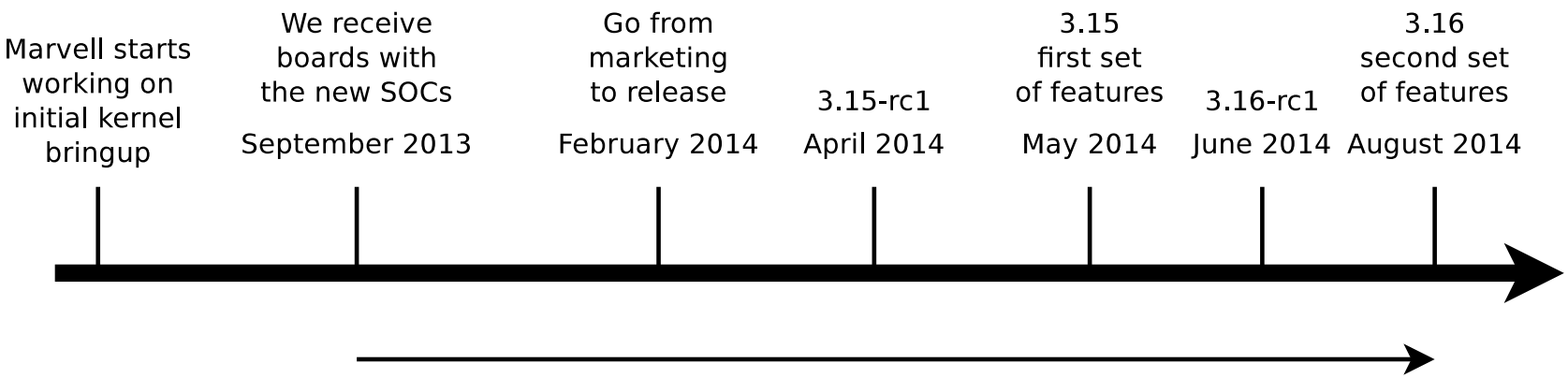
- To implement *cpuidle* on Armada XP, we need to access PMSU registers between 0x22000 to 0x22100: need to change the base address of the first register region.
- Armada 375 has CPU reset registers for SMP, but no PMSU: need to split in two DT nodes.
- And continue support the old DT binding.
- Not able to use the new *reset* framework
- Our experience: be very careful about all these system registers, and think twice.

Lesson #12

Technical vs. marketing

Technical vs. marketing

- A new SoC is going to be released and announced.
- Technical people: we want the SoC to be supported in mainline as soon as it starts shipping.
  - Would require to start and post patches early
- Marketing people: you're not allowed to disclose any details about the SoC before its official release.
  - Prevents from posting patches early, and goes against the limited *absorption rate* problem



Lesson #13

Making estimates is impossible

Making estimates is impossible

- Manager: *please tell me how much time you need to get this merged, and when it will be merged?*
- Making precise estimates for kernel mainlining work is very difficult, close to impossible.
- You can estimate how much work is needed to get to the point where you send the *first* version.
- But then, the review may point out issues, or require refactoring of some subsystem: will require time!
  - For MSI support on Armada 370/XP, had to touch PowerPC, x86, SPARC, Tile, S390!
- With the experience, you will progressively get a better feeling of how difficult it will be for a given change to be merged.

Questions?

thomas.petazzoni@free-electrons.com

Thanks to: Tawfik Bayouk, Lior Amsalem, Ezequiel Garcia, Grégory Clement and Marvell.

Slides under CC-BY-SA 3.0

http://free-electrons.com/pub/conferences/2014/elc/petazzoni-soc-mainlining-lessons-learned/