Device Trees for ARM

Vitaly Wool, Mentor Graphics

Embedded Linux Conference 2009
Grenoble, France

What is a device tree?

- What is it?
  - A tree-like data structure
    - Each node is named
    - Each node has a single parent node
    - Each node has properties
  - Standardized
    - Descriptions follow IEEE 1275
  - Plain text-based
    - Compiled into binary form by the special tool
    - Parsed by kernel code at boot time

What is a device tree for?

- What is it *for*?
  - Aims to describe a platform
    - Functional layout (CPU, Memory, ICs...)
    - Configuration (kernel parameters, consoles, etc.)
    - Device names
  - May be supplied by firmware
  - Requirement for arch/powerpc
    - Also used on Sparc
  - Not deployed on other architectures
    - Why oh ***why***?

Device tree example

```
/ {
    model = "MPC8548CDS";
    compatible = "MPC8548CDS", "MPC85xxCDS";

    cpus {
        #address-cells = <1>;
        #size-cells = <0>;

        PowerPC,8548@0 {
            device_type = "cpu";
            reg = <0x0>;
            d-cache-line-size = <32>; // 32 bytes
            i-cache-line-size = <32>; // 32 bytes
            d-cache-size = <0x8000>;       // L1, 32K
            i-cache-size = <0x8000>;       // L1, 32K
            timebase-frequency = <0>; //  33 MHz, from uboot
            bus-frequency = <0>;  // 166 MHz
            clock-frequency = <0>;    // 825 MHz, from uboot
            next-level-cache = <&L2>;
        };
    };

    memory {
        device_type = "memory";
        reg = <0x0 0x8000000>;    // 128M at 0x0
    ...
```

What is there for ARM now?

- arch/arm/tools/mach-types
  - Plaintext machines' description
    - Name
    - CONFIG_ option
    - MACH_TYPES_ subname
    - Machine ID (unique number)
- Machine ID
  - Passed to the kernel by firmware
  - Allows to determine in run-time
    - CPU type, memory size etc.
    - platform_devices to add
    - Initialization specifics

ARM “mach-types” drawbacks

- Adding new SoC support is overcomplicated
  - New machine description
  - New platform_devices list
    - Even if the number of specifics is very small
  - “versioned” Makefiles/Kconfigs
  - Requires kernel re-compilation
- Platform data bloat
  - Lengthy platform_device lists for each board/SoC
  - Duplication of data

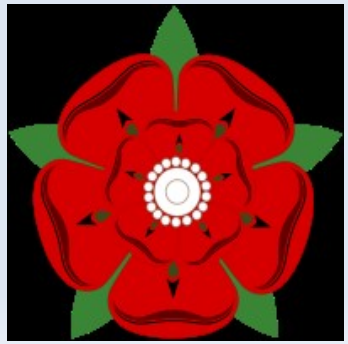
ARM “mach-types” drawbacks (continued)

- Too few flexibility
  - No way to tell the kernel it shouldn't re-init some devices
    - Splashscreen flicker unavoidable
    - Longer boot time
    - “handover” handling in kernel
- ARCH_ and MACH_ mess
  - Can't build a kernel supporting both i.MX31 and OMAP2430

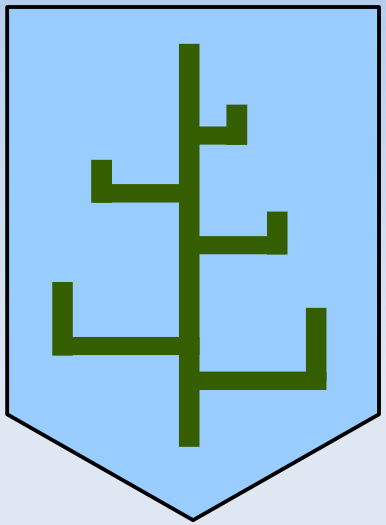
DTs and ARM: current status

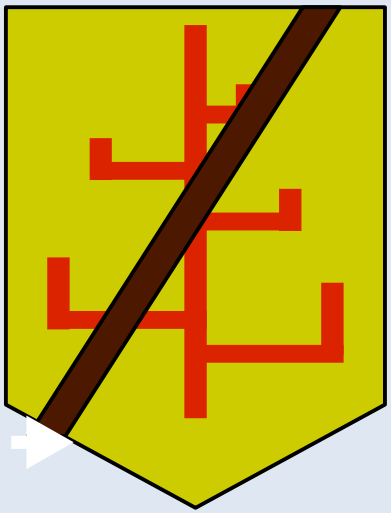
- Multiple attempts to implement and deploy
  - Each causing heated discussion
  - None hitting the mainline
  - Last attempt: May 2009
- Latest news
  - “Holy War” May-June 2009
  - Reminded of The War of The Roses

- Dynastic war, 15th century
- Yorks (white), Lancasters (red)



Wars of the trees





- Start date: Wed May 27, 2009
- Started with: Janboe Ye's LKML patch
- The Greens (Pro DT) commanders:
  - Grant Likely
  - David Miller
  - Benjamin Herrenschmidt
- The Reds (Contra DT) commanders:
  - Russell King
  - Sasha Hauer
  - Mark Brown

The Greens' armor

- Simplified new SoC support addition
  - Might be as simple as “define a new tree”
    - No re-compilation
- Flexibility
  - Different initialization options
    - Parallel initialization possible
    - Ability to clearly specify dependencies
  - Device tree validation options
    - If it's invalid, fall back to default
- **True** multiplaform kernel
  - CPU model based
    - ARCH_XXX could go away

The Reds' armor

- DT's are bloated
  - Additional code to parse the trees
- DT's slow down kernel bootup
  - Tree parsing takes CPU cycles
- DT's don't describe some things well enough
  - Complicated interconnections between devices
    - Audio codec/bluetooth/GSM
  - GPIO-based initializations
    - Can't express the code in plain text!

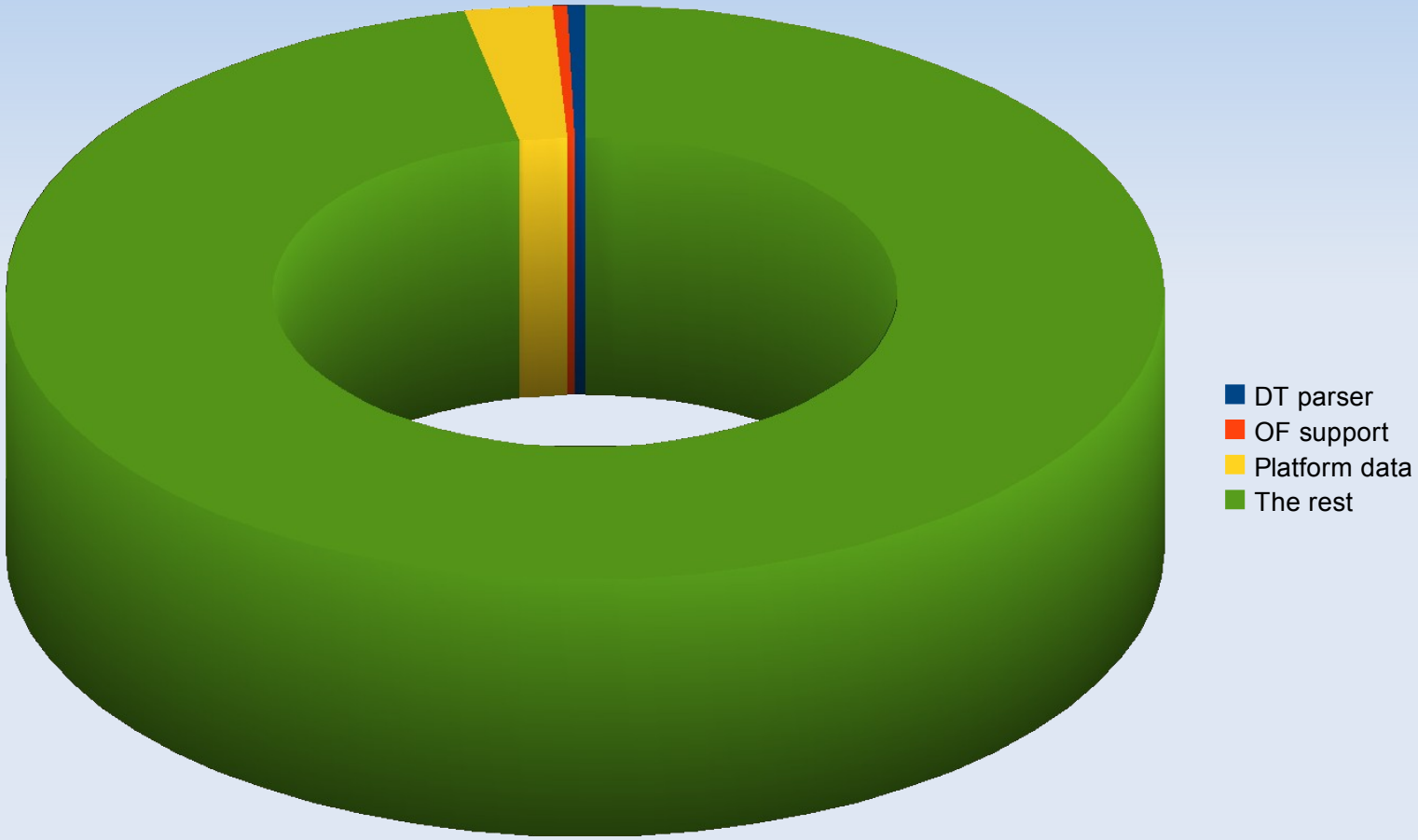
So...

And there was Fight!

Battle 1: “bloat”

- DT's add 5+k overall
  - ~3k drivers/of
  - ~4k ARM DT support
- DT parsing is complex
  - And so is Linux
  - written once used many
+ DT saves ~10k/platform
  - platform_devices/platform data for each platform
= Conclusion: this point is invalid.

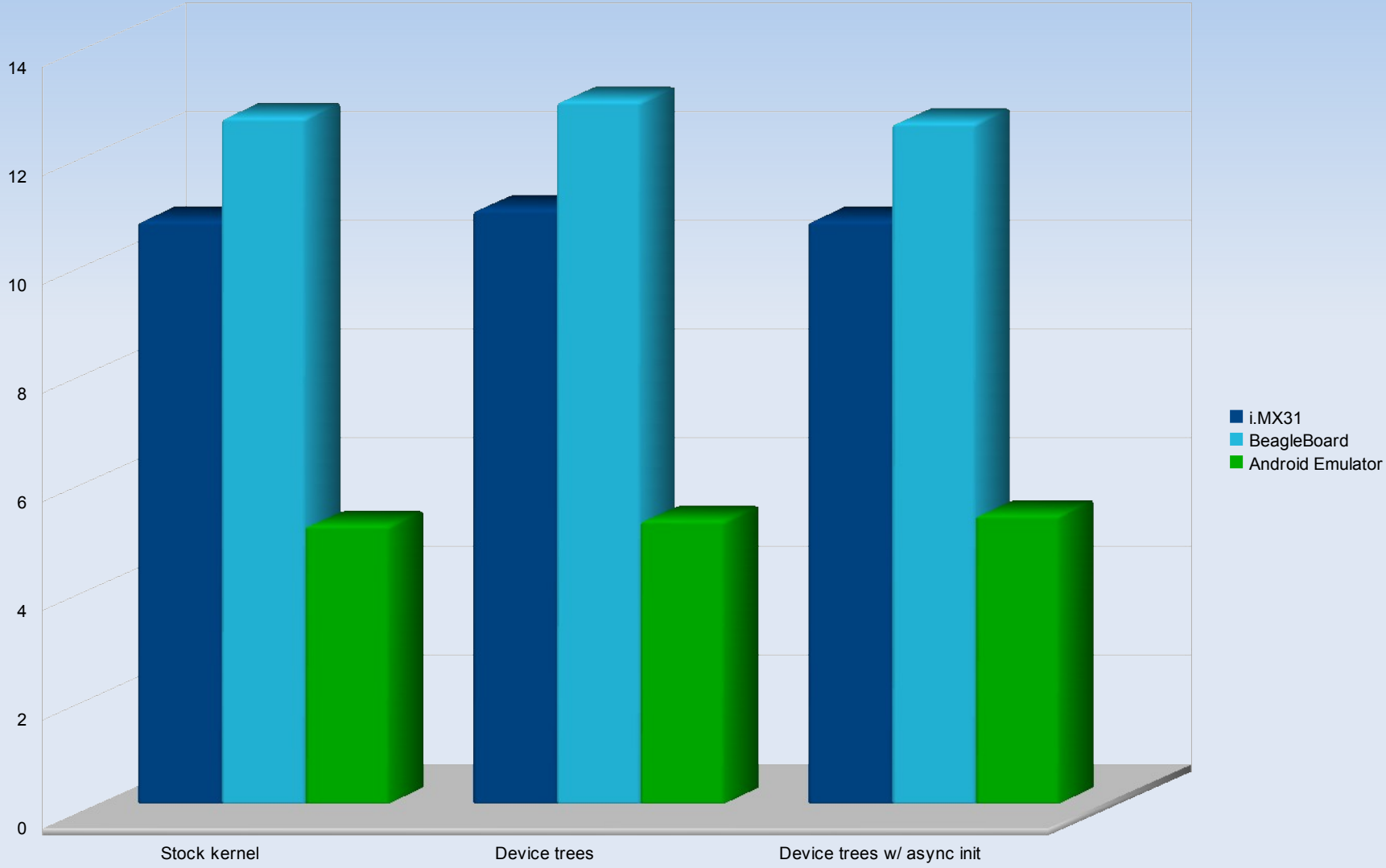
Kernel code and DTs



Battle 2: boot-up time

- DT parsing adds time to bootup
  - The time depends on CPU performance
  - It is really marginal for modern ARM CPUs
+ DT's may be used for parallel initialization
  - Easy to express depenencies
  - Easy to specify “weight”
= Conclusion: this point is also irrelevant

Boot-up time and DT's



Battle 3: flexibility

+ DT's add flexibility
  - Initialization
  - validation
- Too much flexibility is granted to firmware
  - With mach-id, things have settled up well wrt firmware/kernel border definition
- DT's are not flexible enough for some corner cases
  - Tighly coupled hardware (like BT+GSM+codec)
  - Complicated platform-specific device init (GPIO)
= Conclusion: DT's are not ready to handle that

“Corner cases” ?

- Rare thing on PowerPC
- Not that important for typical PowerPC-based systems
  - Openness
  - Flexibility to add clones
- e. g. PXA is a big fat corner case
- GPIO configuration for most of the devices (e. g. i.MX)
  - Closedness
  - Flexibility to reconfigure the same platform

Example:
platform_device and pin multiplexing

```
...
struct stmp3xxx_fb_platform_data {
    char name[16];
    u16 x_res;
    u16 y_res;
    u16 bpp;
    u32 cycle_time_ns;
    int lcd_type;
    int (*init_panel)(); /* pins multiplexing */
    void (*release_panel)(); /* pins release */
...
```

- How to express this using DT?
  - List of pins to configure as a property
  - Platform-wide function for pin configuration
    - Supplied if the property is present for a device
- Still no way to express e. g. dotclock init

Battle 4: proof of concept

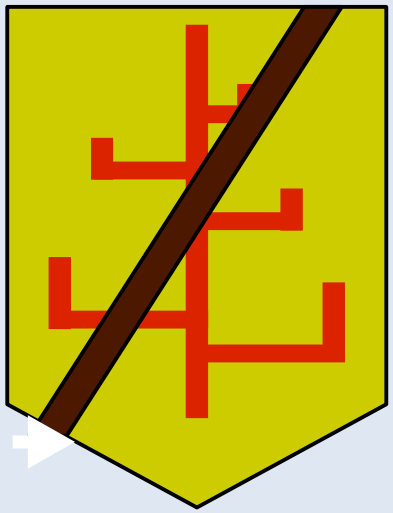
+ DT's are there for quite a while
  - PowerPC
  - OpenFirmware / OpenBIOS
- ARM is special
  - Variety of ARM firmware (not standardized)
  - No OpenBIOS, so no need for DT's
  - ARM is mobile, so it's closed architecture
  - No CompactPCI-like hotswap
- No working DT utilization example
= Conclusion: no **real** proof of concept for ARM

Battle 5: of_device

+ Used for PowerPC for ages
  - Simple wrapper over struct device
- Doesn't convey what ARM needs
  - No platform_data analog
  - No resource analog
- Reworking ARM platform part for of_device is lengthy and senseless
  - And it's better to have unified approach
= Conclusion: of_device is not providing what ARM needs

The War of Trees: results

- Local successes of the Greens
- But overall, the Reds take the victory
  - Device trees are not ready for deployment on ARM
- The Greens have to better prepare for the next battle :-)



Winning strategy for the Greens?

- Proof-of-concept for a really complicated multi-SoC platform
  - Work for PXA is ongoing
- Update the implementation
  - Add GPIO descriptions
    - A platform-wide function could be used as a callack
  - Get rid of of_device
  - A property for “trusted” bootloaders?
- Use vendors as a reinforcement :)
  - Many are interested in DT's adption for ARM

Good luck the Greens!

- With a true multiplatform kernel:
  - Less effort for kernel testing
    - More automation
    - Better quality
  - More concentration on middleware
    - We have to add value there, kernel's almost done
- With DT's adopted for ARM
  - Less duplication of code
    - Merge of_device/platform_device versions of the same thing
  - Better firmware/kernel interworking

Peace!

Thanks for your attention!