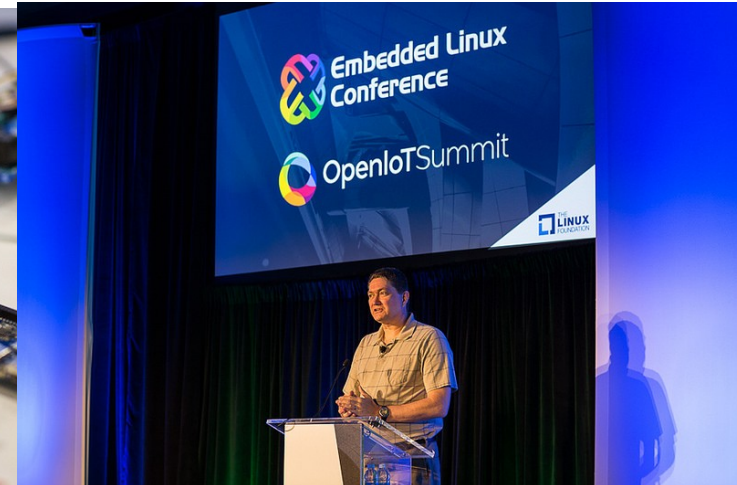
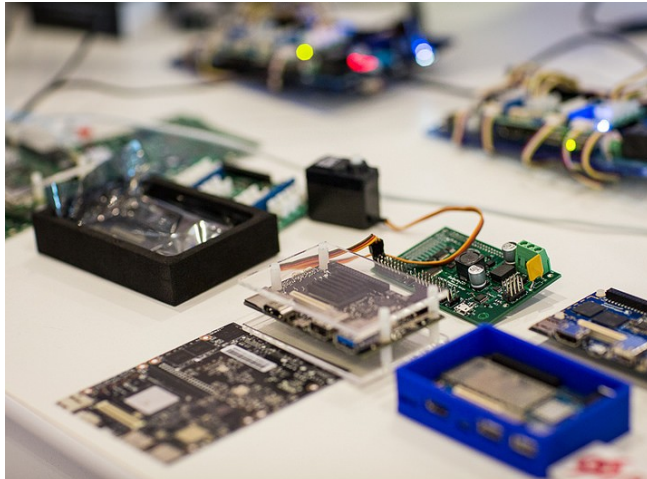


Embedded Linux Conf 2016



Drew Fustini (@pdp7)
drew@beagleboard.org

<http://events.linuxfoundation.org/events/embedded-linux-conference>



Free Electrons and C.H.I.P.

- C.H.I.P. Does Embedded Linux Conference Things with the Free Electrons Team



Free Electrons and C.H.I.P.



Context: the CHIP, the capes and us

- ▶ The **CHIP**: a 9\$ board by **NextThing Co.** built around the Allwinner R8 SoC (Cortex-A8).
- ▶ Funded thanks to a Kickstarter campaign in 2015.
- ▶ **Free Electrons** working on the CHIP kernel support.
- ▶ Was designed from the beginning to have adapters:
 - ▶ VGA adapter.
 - ▶ HDMI adapter.
 - ▶ Pocket CHIP.

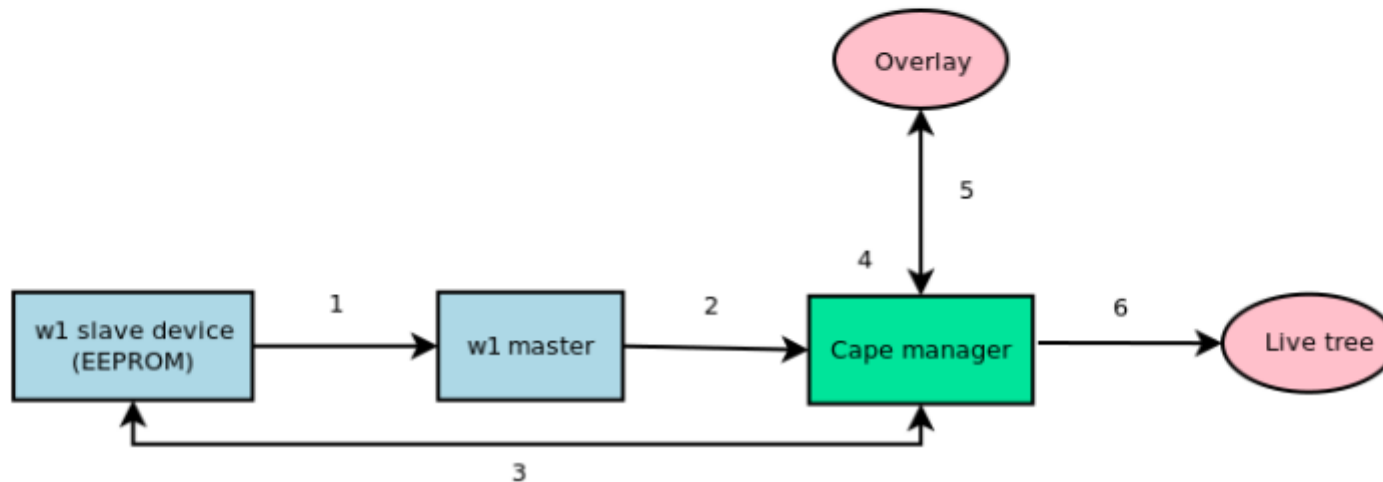


Free Electrons and C.H.I.P.

- Using DT Overlays to Support the C.H.I.P.'s Capes - Antoine Tenart, Free Electrons
 - PDF of Slides
 - how Device Tree overlays work and how they are generated, walk through the design of the cape manager used on the C.H.I.P and detail how a cape is detected and the Device Tree overlays applied at run-time.
 - CHIP Capes also called “Dips”, e.g. *“Chips and Dip”*

Using DT overlays to support the C.H.I.P.'s capes:

1. A new slave device is detected on the 1-wire bus.
2. If the new device family is recognized by the cape manager, a callback is called.
3. The cape manager reads the header stored on the EEPROM.
4. The cape manager parses the header and decides which cape to load.
5. A DT overlay is loaded from userspace.
6. The overlay is applied on the live tree.



Using DT overlays to support the C.H.I.P.'s capes: EEPROM header

- ▶ Each pin used to communicate to the EEPROM cannot be reused:
 - ▶ We wanted a bus with the lowest number of lines.
- ▶ We did not need a high speed bus:
 - ▶ Only used to read the cape's header.
- ▶ The bus must support enumeration, to connect more than one cape.
- ▶ We chose the 1-wire bus.

```
struct cape_chip_header {  
    u32    magic;        /* must be 0x43484950 "CHIP" */  
    u8     version;      /* spec version */  
    u32    vendor_id;  
    u16    product_id;  
    u8     product_version;  
    char   vendor_name[32];  
    char   product_name[32];  
    u8     rsvd[36];     /* rsvd for future versions */  
    u8     data[16];     /* per-cape specific */  
} __packed;
```


Using DT overlays to support the C.H.I.P.'s capes:

1 Wire Bus

- ▶ Single signal.
- ▶ Low-speed data and signaling.
- ▶ Only two wires needed:
 - ▶ Data.
 - ▶ Ground.
- ▶ Uses a capacitor to store charge and power the device when the data line is active.
 - ▶ The capacitor needs to be charged!
 - ▶ We had weird side effects because of this in U-Boot → the line needs to be pulled long enough firstly.
- ▶ Two speed modes: normal and overdrive (speed x10).
- ▶ Four operations: `read`, `write 0`, `write 1` and `reset`.
- ▶ Can be used over a GPIO.
 - ▶ `drivers/w1/master/w1-gpio.c`

Using DT overlays to support the C.H.I.P.'s capes:

1 Wire Bus & Cape Manager

- ▶ Responsible for detecting a cape, identifying it and applying the corresponding overlay.
- ▶ Uses all components described before:
 - ▶ The 1-wire bus.
 - ▶ The EEPROM in which the cape's header is stored.
 - ▶ The device tree overlay mechanism.
- ▶ Implemented in the kernel space.
- ▶ We patched the 1-wire framework to add callbacks when a new device is detected on the bus.
 - ▶ Allows to read the header stored on the cape's EEPROM as soon as the cape is detected.
- ▶ The EEPROM driver for the DS2431 was available in `drivers/w1/slaves/`
- ▶ Cannot be used outside of the 1-wire framework!
- ▶ We redefined its read function in the cape manager.

Using DT overlays to support the C.H.I.P.'s capes: Status

- ▶ Works fine for most uses.
- ▶ Our first test was with a LED and a PWM.
- ▶ This can't work when adding / enabling devices handled by subsystems without hotplug support.
 - ▶ Like DRM/KMS.
- ▶ Quick solution: add the overlay support in the bootloader.
 - ▶ Maxime Ripard patched U-Boot.
 - ▶ Not yet upstreamed.
- ▶ Would be better to patch directly DRM/KMS.

Free Electrons and C.H.I.P.

- Bringing Display and 3D to the C.H.I.P Computer -
Maxime Ripard, Free Electrons
 - **PDF of Slides**
 - walk through the DRM stack, the architecture of a DRM/KMS driver and the interaction between the display and GPU drivers. The presentation is based on the work we have done to develop a DRM driver for the Allwinner SoCs display controller, as part of enabling the C.H.I.P platform with the upstream Linux kernel. The work done to make the ARM Mali OpenGL driver work on top of a mainline DRM/KMS driver will also be detailed.

Bringing display and 3D to the C.H.I.P computer



Doing display things

- ▶ Different solutions, provided by different subsystems:
 - ▶ FBDEV: Framebuffer Device
 - ▶ DRM/KMS: Direct Rendering Manager / Kernel Mode Setting
 - ▶ More exotic ones: V4L2, auxdisplay
- ▶ How to choose one: it depends on your needs
 - ▶ Each subsystem provides its own set of features
 - ▶ Different levels of complexity
 - ▶ Different levels of activity

Bringing display and 3D to the C.H.I.P computer



Which one to choose?

- ▶ DRM
 - ▶ Actively maintained
 - ▶ Provides fine grained control on the display pipeline
 - ▶ Widely used by user-space graphic stacks
 - ▶ Provides a full set of advanced features
- ▶ FBDEV
 - ▶ Deprecated?
 - ▶ Does not provides all the features found in the modern display controllers (overlays, sprites, hw cursor, ...)

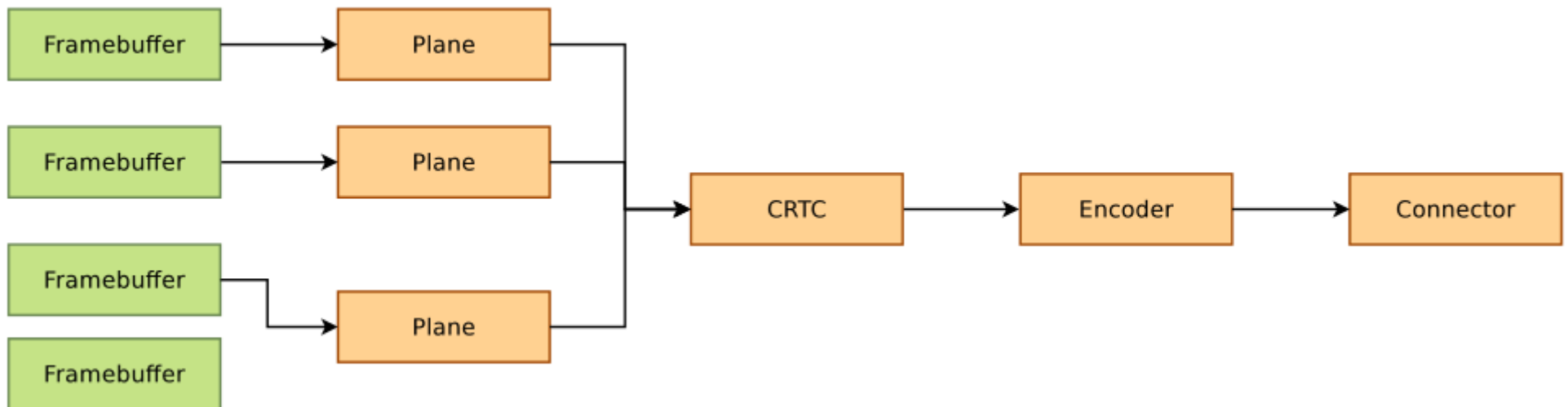
Bringing display and 3D to the C.H.I.P computer



DRM/KMS: Definition

- ▶ DRM stands for Direct Rendering Manager and was introduced to deal with graphic cards embedding GPUs
- ▶ KMS stands for Kernel Mode Setting and is a sub-part of the DRM API
- ▶ Though rendering and mode setting are now split in two different APIs (accessible through `/dev/dri/renderX` and `/dev/dri/controlDX`)
- ▶ KMS provide a way to configure the display pipeline of a graphic card (or an embedded system)
- ▶ KMS is what we're interested in when looking for an FBDEV alternative

Bringing display and 3D to the C.H.I.P computer



Bringing display and 3D to the C.H.I.P computer



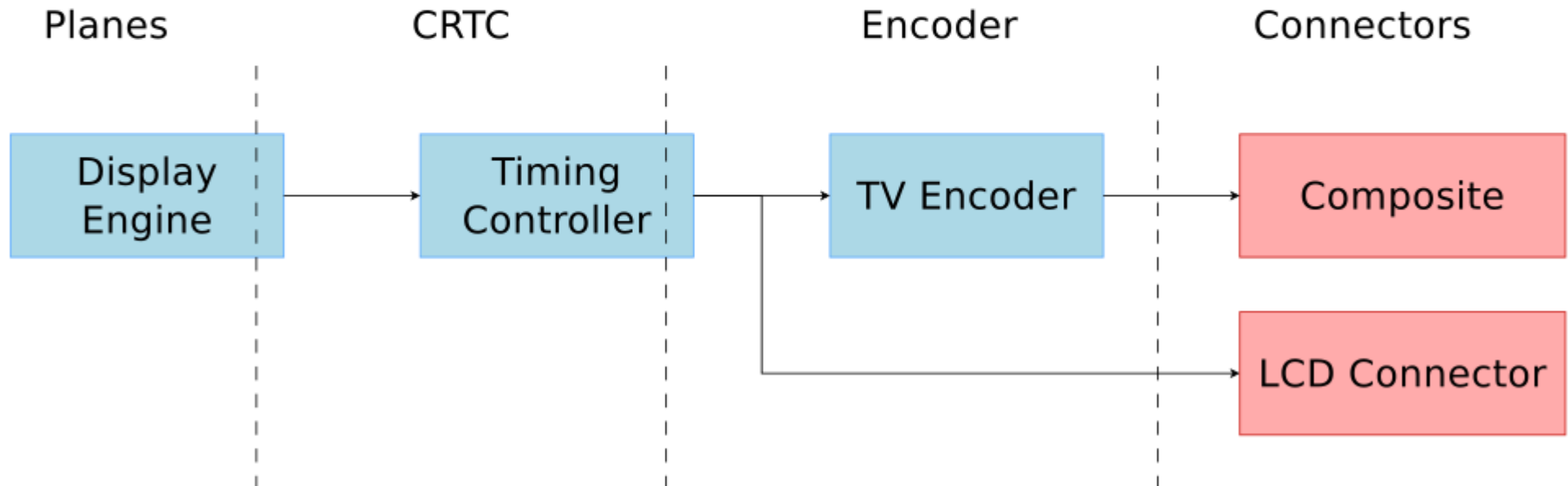
KMS components

- ▶ Planes
 - ▶ Image source
 - ▶ Associated with one (or more!) framebuffers
 - ▶ Holds a resized version of that framebuffer
- ▶ CRTC's
 - ▶ Take the planes, and does the composition
 - ▶ Contains the display mode and parameters
- ▶ Encoders
 - ▶ Take the raw data from the CRTC and convert it to a particular format
- ▶ Connectors
 - ▶ Outputs the encoded data to an external display
 - ▶ Handles hotplug events
 - ▶ Reads EDIDs

Bringing display and 3D to the C.H.I.P computer



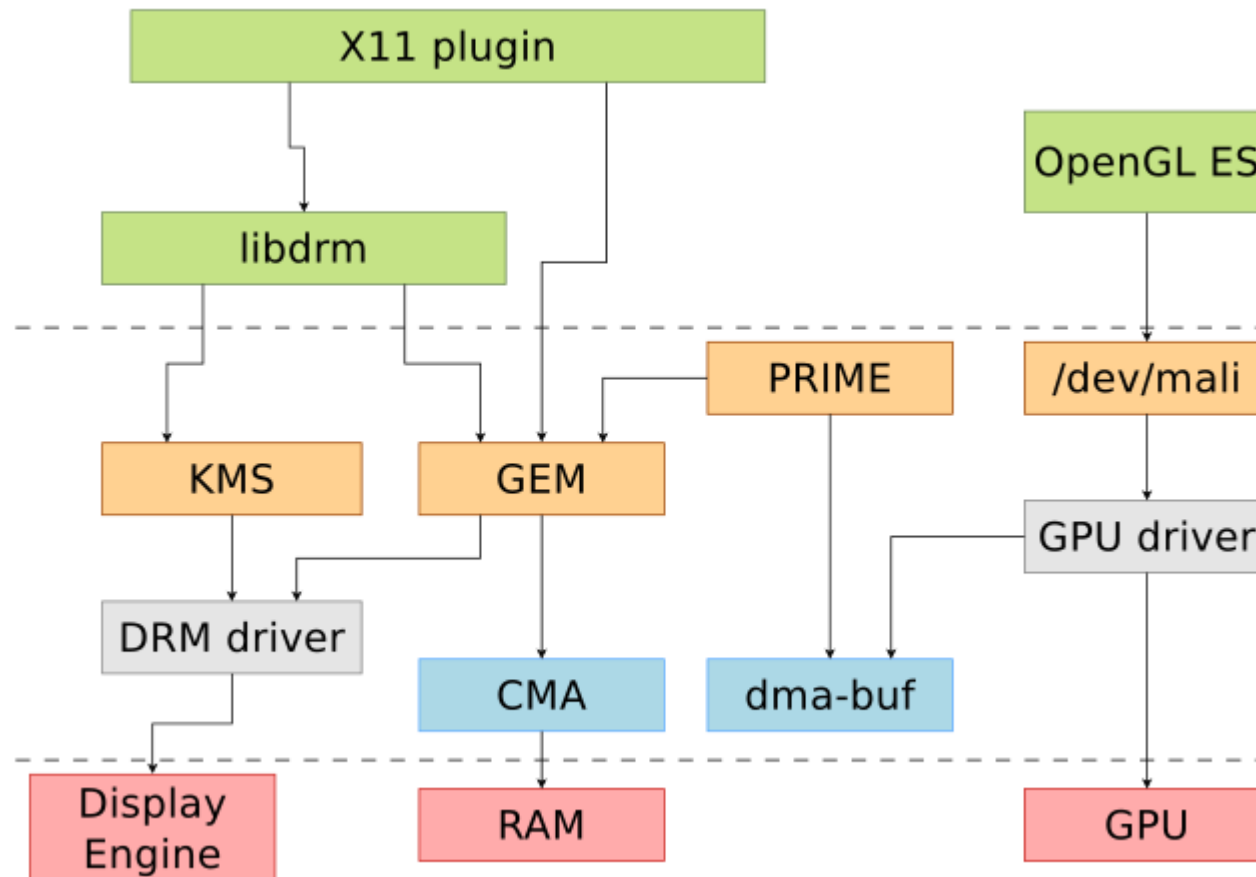
DRM vs SoC pipeline



Bringing display and 3D to the C.H.I.P computer

- ▶ The GPU found in most Allwinner SoCs is the Mali-400 from ARM (with a variable number of cores)
- ▶ There are two options to support that GPU:
 - ▶ Lima
 - ▶ Reversed engineered proof-of-concept
 - ▶ Triggered the reverse engineering effort of the GPUs (freedreno, etnaviv, etc.)
 - ▶ Development (closed to?) stopped two years ago
 - ▶ ARM-Provided support
 - ▶ Featureful
 - ▶ Two parts: GPL kernel driver and proprietary OpenGL ES implementation

Bringing display and 3D to the C.H.I.P computer



Bringing display and 3D to the C.H.I.P computer



X11 integration

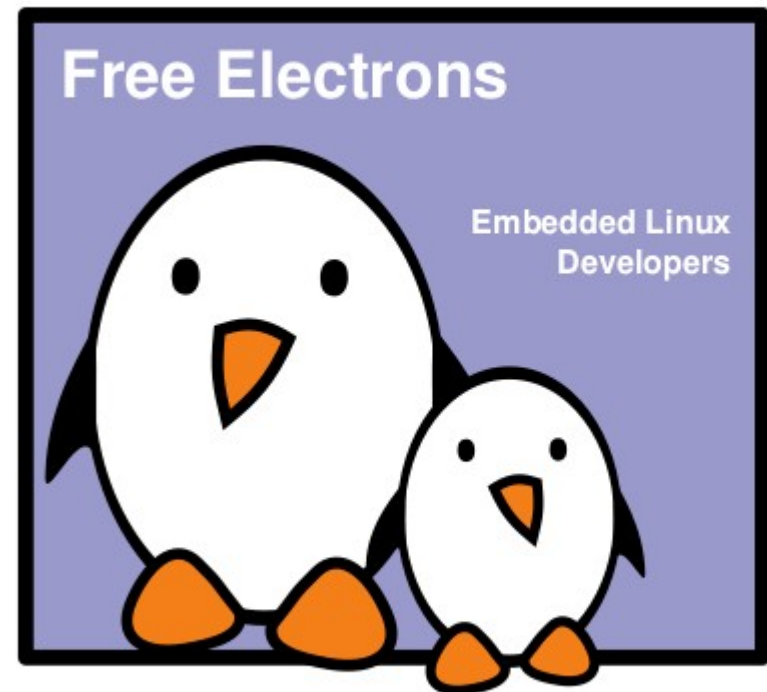
- ▶ We need a DDX (Device Dependent X) driver
- ▶ `xf86-video-modesetting` is working on top of KMS and MESA (Gallium3D)
- ▶ ARM developed `xf86-video-armsoc` for SoC using a 3rd party GPU (Mali, PowerVR, Vivante, etc.)
- ▶ Relies on KMS for the display configuration, driver-specific ioctl for buffer allocations and vendor-provided OpenGL ES implementation
- ▶ Just have to write a small glue to use your driver allocator, and give some hints to X about what your hardware support (hw cursor, vblank, etc.)

https://events.linuxfoundation.org/sites/events/files/slides/brezillon-nand-framework_0.pdf

Modernizing the NAND framework: The big picture

Boris Brezillon
Free Electrons
boris@free-electrons.com

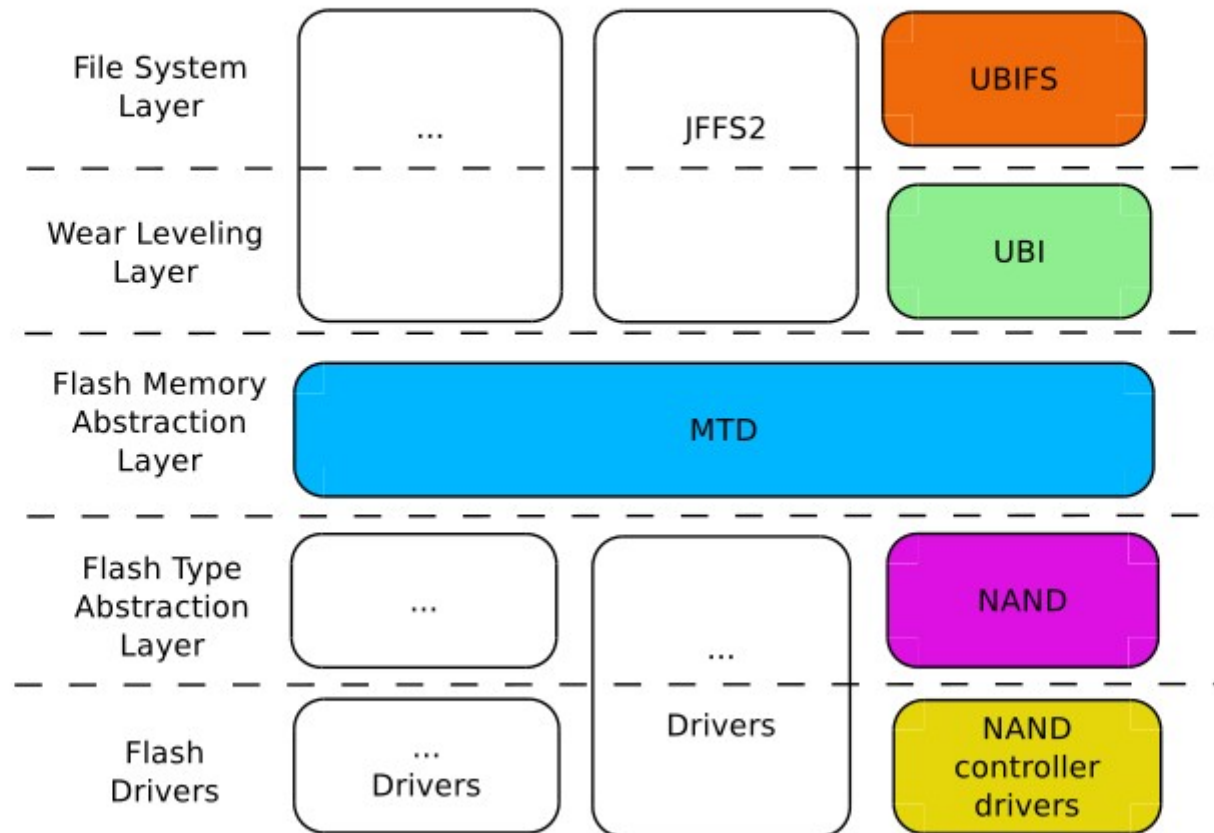
© Copyright 2004-2016, Free Electrons.
Creative Commons BY-SA 3.0 license.
Corrections, suggestions, contributions and translations are welcome!



https://events.linuxfoundation.org/sites/events/files/slides/brezillon-nand-framework_0.pdf



NAND framework position in the flash stack



<https://events.linuxfoundation.org/sites/events/files/slides/elc-understanding-rt-system-2016.pdf>

Understanding a Real-Time System

More than just a kernel

Steven Rostedt
rostedt@goodmis.org
srostedt@redhat.com

<https://events.linuxfoundation.org/sites/events/files/slides/elc-understanding-rt-system-2016.pdf>

What is Real-Time?

Deterministic results

Repeatable results

Doing what you expect when you expect it

No unbounded latency

Can calculate worst case scenarios

All environments are Real-Time.

<https://events.linuxfoundation.org/sites/events/files/slides/elc-understanding-rt-system-2016.pdf>

What is Real Fast?

Hot cache

Look ahead features

Paging

Translation Lookaside Buffer (TLB)

Least interruptions

Optimize the most likely case

Transactional Memory

https://events.linuxfoundation.org/sites/events/files/slides/libiio_0.pdf

Introduction to IIO

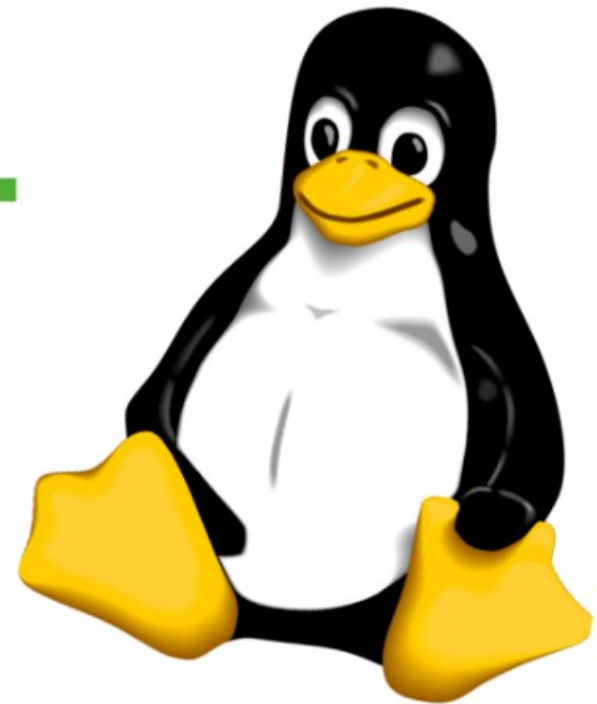
- Industrial Input/Output framework
 - Not really just for Industrial IO
 - All non-HID IO
 - ADC, DAC, light, accelerometer, gyro, magnetometer, humidity, temperature, rotation, angular momentum, lifestyle sensors ...
- Developed by Jonathan Cameron
- In the kernel since v2.6.32 (2009)
- Moved out of staging/ in v3.5 (2012)
- ~220 IIO device drivers (v4.6)
 - Many drivers support multiple devices

- https://events.linuxfoundation.org/sites/events/files/slides/elc_2016_mem.pdf

Virtual Memory and Linux



Alan Ott
Embedded Linux Conference
April 4-6, 2016



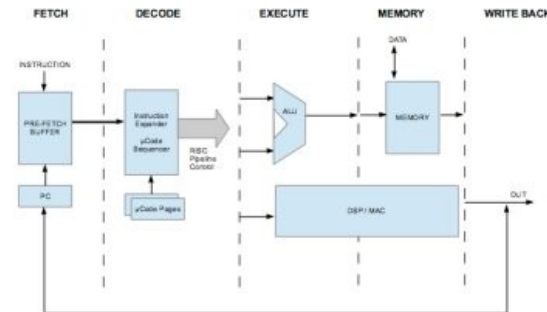
J-Core Open Source Processor

- PDF: <http://j-core.org/ELC-2016.pdf>
- *Google Drive:*
https://drive.google.com/open?id=0B_NI2VDamOOfc0RaNWJHQWNsOFk
- Website: <http://j-core.org/>
- This page describes the j-core processor, a clean-room open source processor and SOC design using the SuperH instruction set, implemented in VHDL and available royalty and patent free under a BSD license.

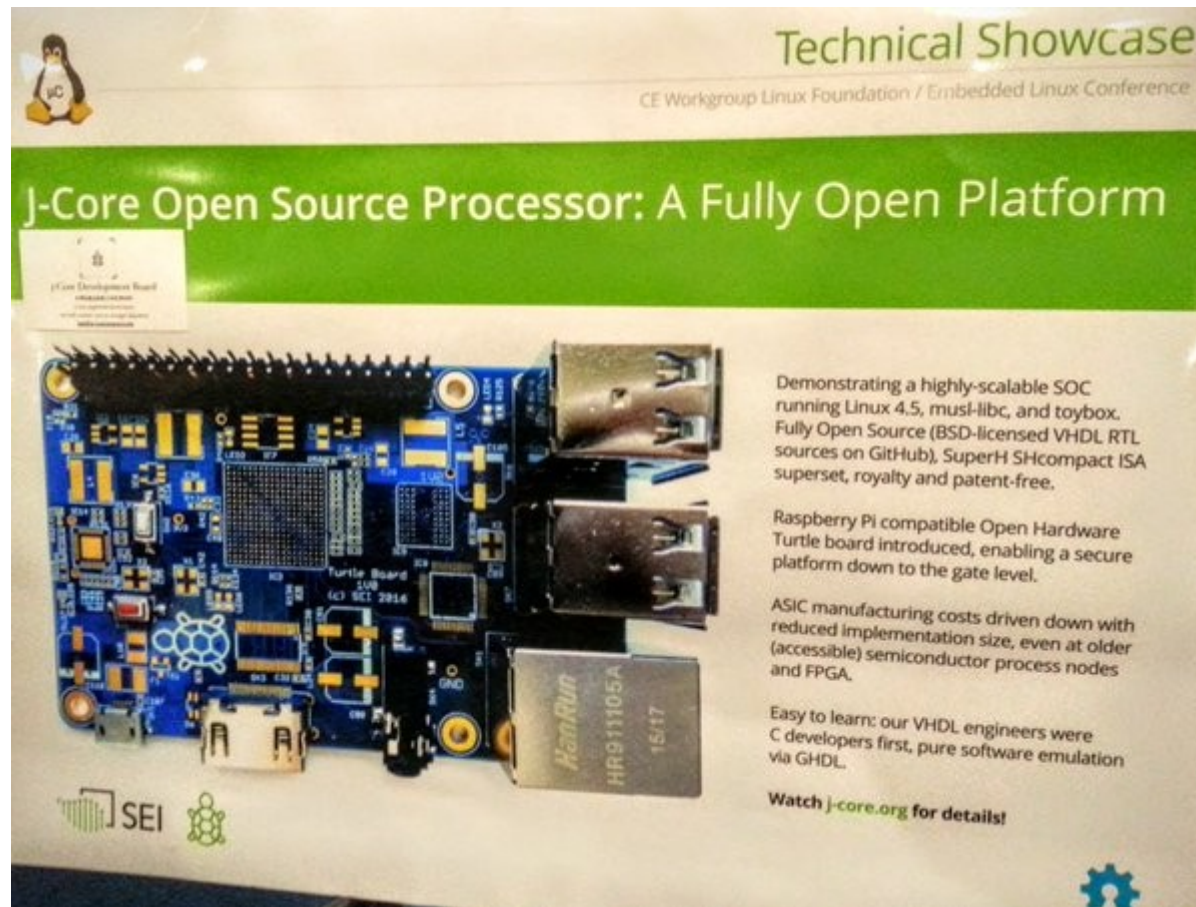
J-Core Open Source Processor




j-core Design Walkthrough




J-Core Open Source Processor



The poster features a photograph of a blue J-Core development board with various components like a microcontroller, capacitors, and connectors. A small white card is placed on the board. The background is a light green gradient with white text and logos.

 **Technical Showcase**
CE Workgroup Linux Foundation / Embedded Linux Conference

J-Core Open Source Processor: A Fully Open Platform

 J-Core Development Board
Fully Open Source
www.j-core.org

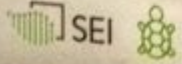

Demonstrating a highly-scalable SOC running Linux 4.5, musl-libc, and toybox. Fully Open Source (BSD-licensed VHDL RTL sources on GitHub), SuperH SHcompact ISA superset, royalty and patent-free.


Raspberry Pi compatible Open Hardware Turtle board introduced, enabling a secure platform down to the gate level.

ASIC manufacturing costs driven down with reduced implementation size, even at older (accessible) semiconductor process nodes and FPGA.

Easy to learn: our VHDL engineers were C developers first, pure software emulation via GHDL.

Watch j-core.org for details!

 SEI 



Demonstrating a highly-scalable SOC
running Linux 4.5, musl-libc, and toybox.
Fully Open Source (BSD-licensed VHDL RTL
sources on GitHub), SuperH SHcompact ISA
superset, royalty and patent-free.

Raspberry Pi compatible Open Hardware
Turtle board introduced, enabling a secure
platform down to the gate level.

ASIC manufacturing costs driven down with
reduced implementation size, even at older
(accessible) semiconductor process nodes
and FPGA.

Easy to learn: our VHDL engineers were
C developers first, pure software emulation
via GHDL.

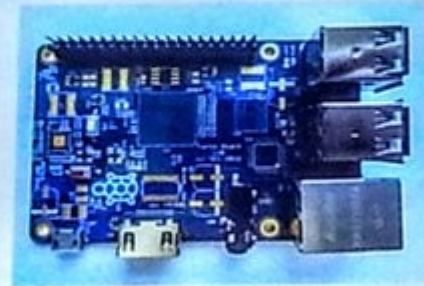
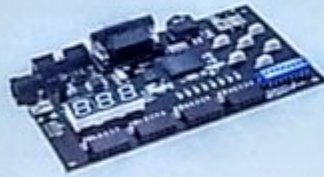
Watch j-core.org for details!



J-Core Open Source Processor

Demonstration Platforms

- Our j2 processor core can run linux on low-cost FPGA Spartan6 platforms such as Numato Labs' Mimas2



- We are launching a custom development board (Turtle Board) with the same form factor as Raspberry Pi
- (Please support out Kickstarter)





Video?

- Coming soon (~2 weeks):
<https://www.youtube.com/user/TheLinuxFoundation/videos>