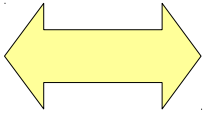
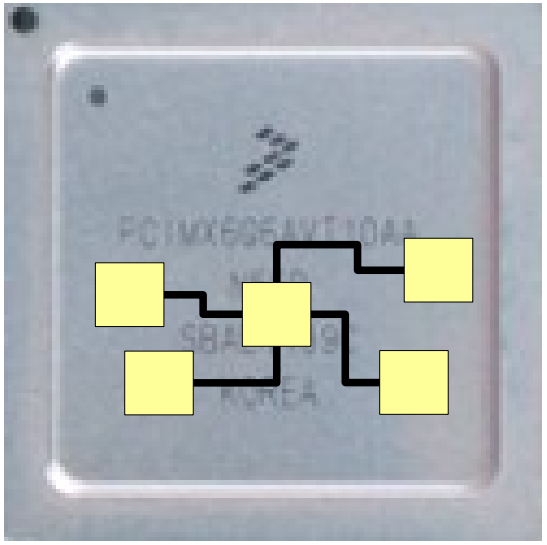
Modular Graphics on Embedded ARM





Embedded Linux Conference Europe
Barcelona, 2012-11-05
Philipp Zabel <p.zabel@pengutronix.de>

Pengutronix.

The Past: PC Graphics Drivers

- Historically, PC graphics cards are monolithic devices
- DRM drivers are implemented monolithically:
  - Driver has full control over instantiation of sub components
  - Sub components are no „linux devices“
  - Lots of details, but all private to driver

Pengutronix.

The Past: Embedded Systems

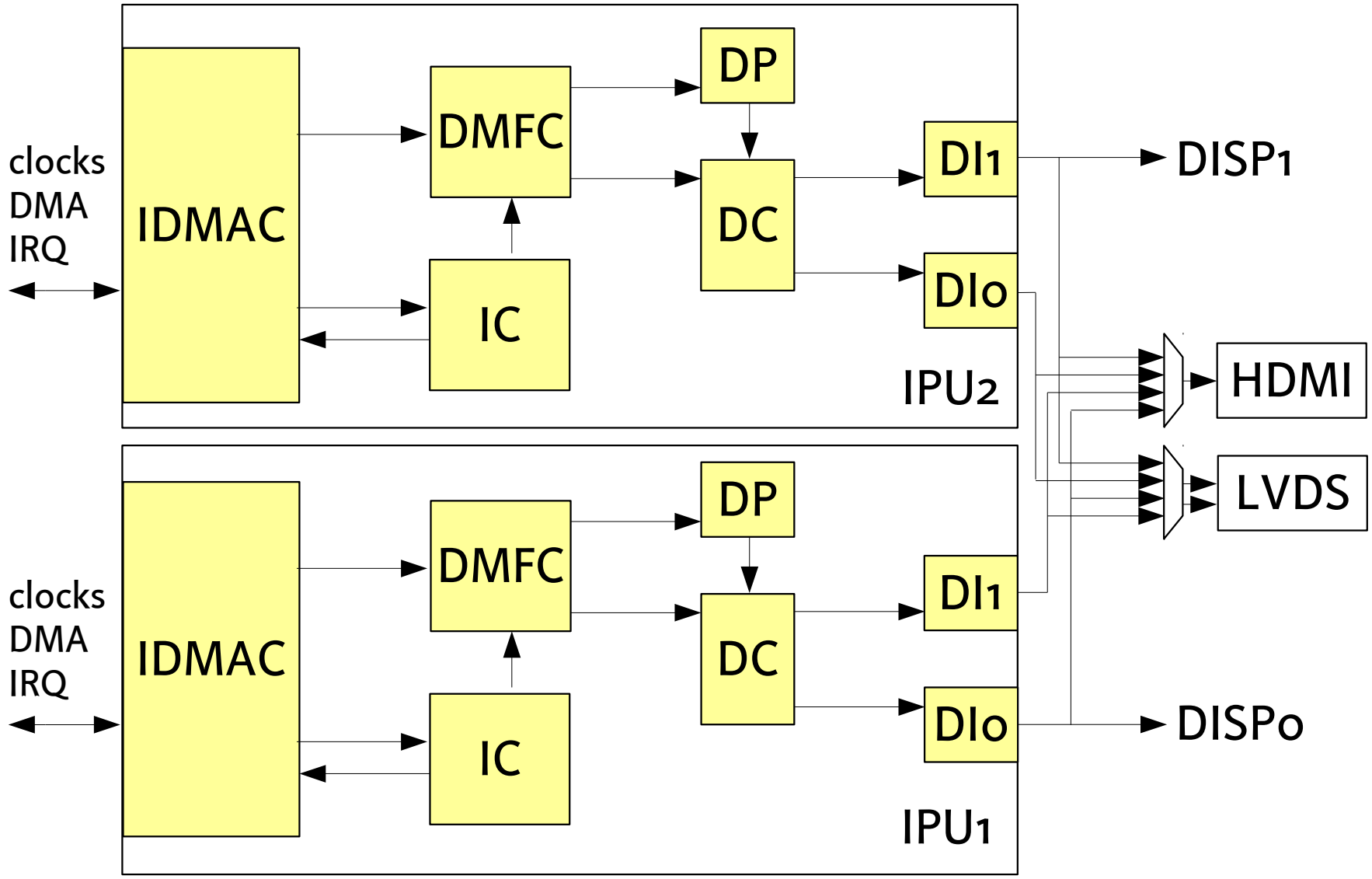
- Historically, embedded linux systems had fbdev drivers
- Simple model:
  - Rectangular memory <-> Display
  - Only one output

The Present: How to do Graphics on i.MX...?

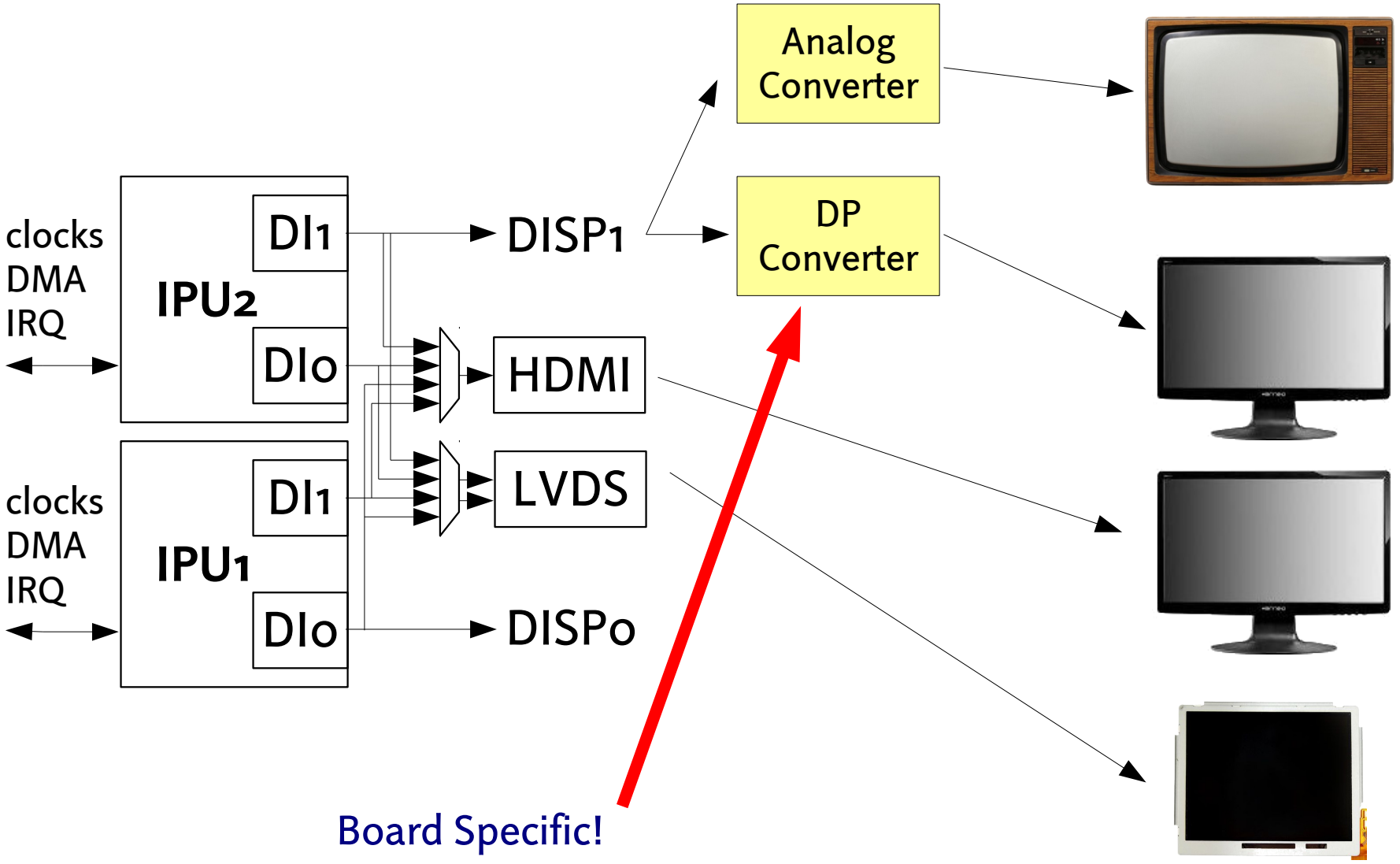
- Modern ARM SoC processors have more complex graphics these days
- Example: MX6 Project
- Camera input + processing
- LVDS, parallel RGB, smart panel output, HDMI, DP, ...
- Internal function blocks
- External function blocks

Pengutronix.

Display Path: Internal Function Blocks



Display Path: External Function Blocks

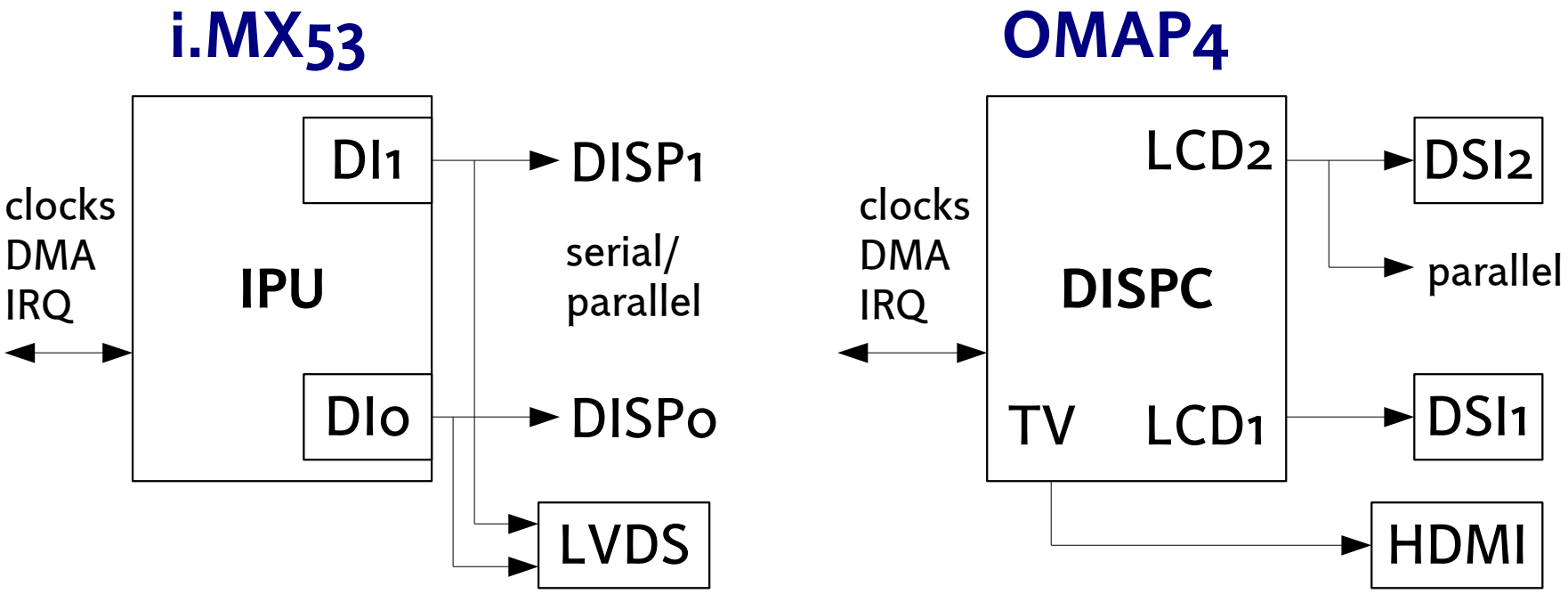


The SoC Perspective: Re-Usable Blocks

- SoC graphics consists of multiple blocks
- Sampling frame buffers from system memory into various video signals
- Optionally: Plane combining, color space format conversion
- On-SoC components may be grouped differently
  (example: MX53: IPUv3 -> MX6: 2x IPUv3)
- Off-SoC components may be grouped differently
  (example: external signal encoders)

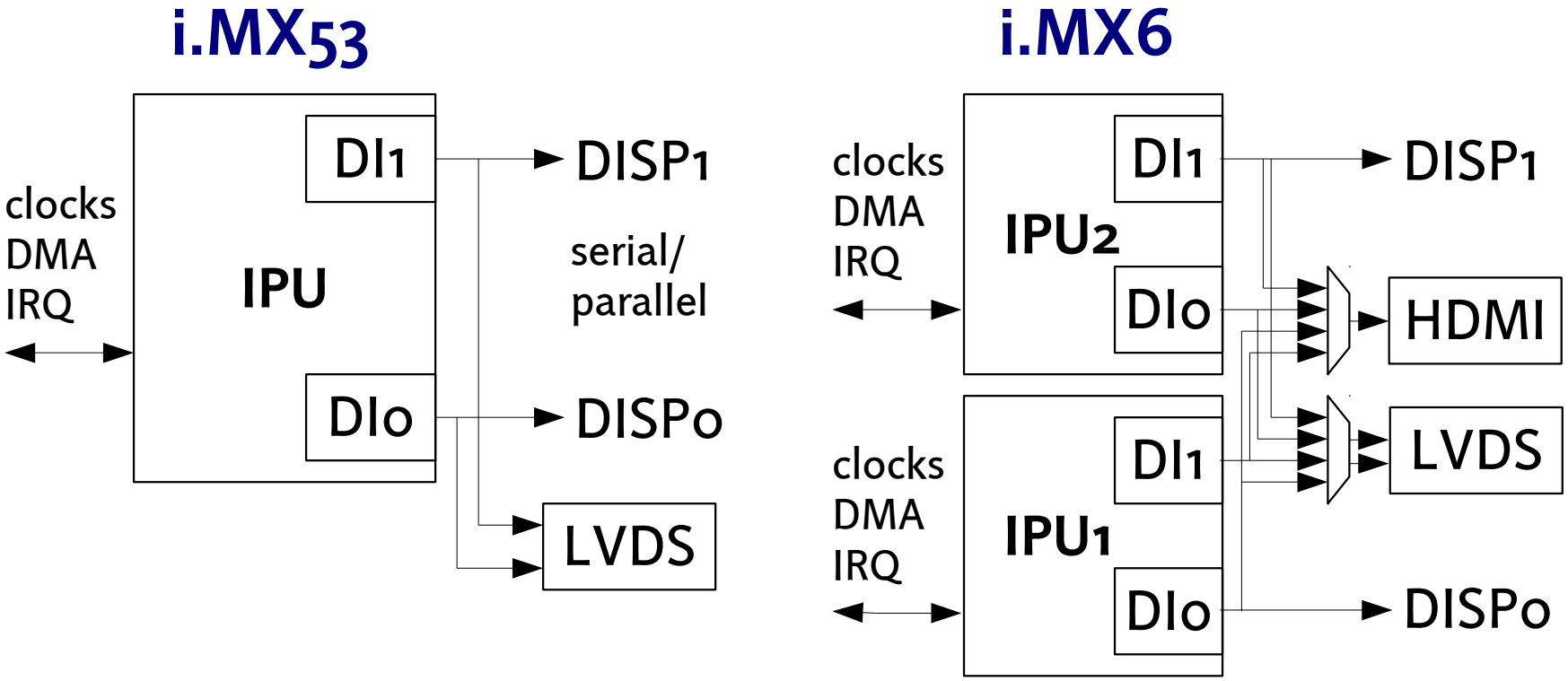
The SoC Perspective: Re-Usable Blocks

- Example: MX53 vs. OMAP4



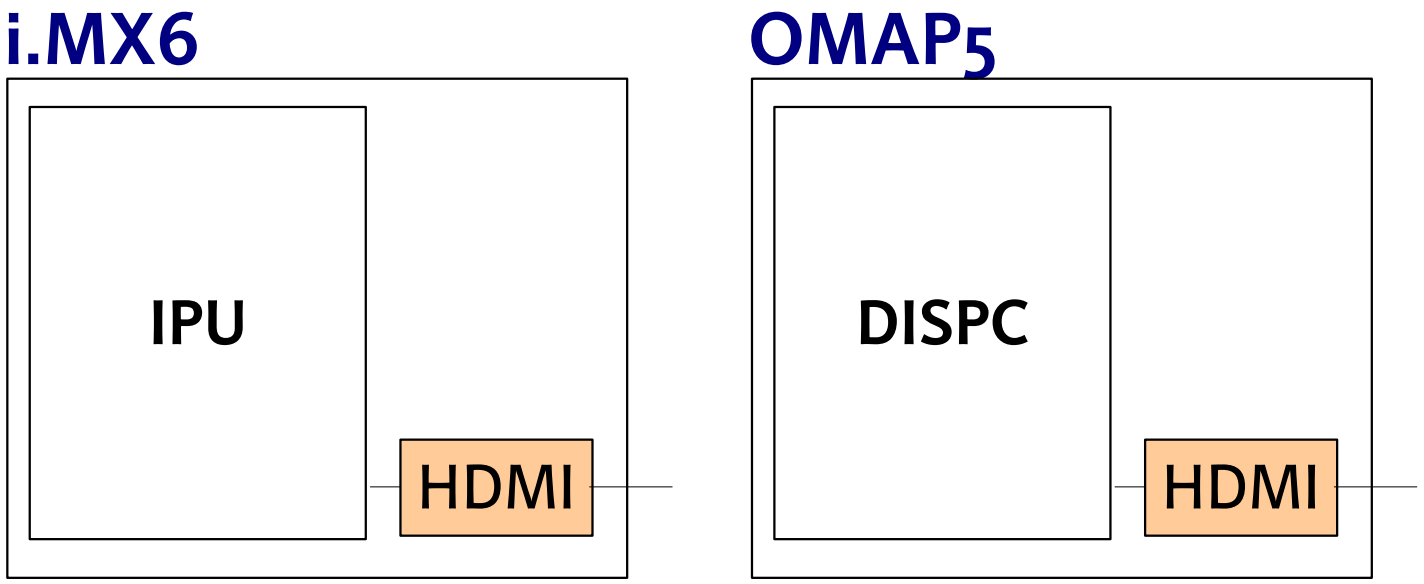
The SoC Perspective: Re-Usable Blocks

- Example: MX53 vs. MX6



The SoC Perspective: Re-Usable Blocks

- Example: „HDMI TX“ encoder on MX6 and OMAP5



Identical (3rd party) IP cores!

Re-Usable External Blocks

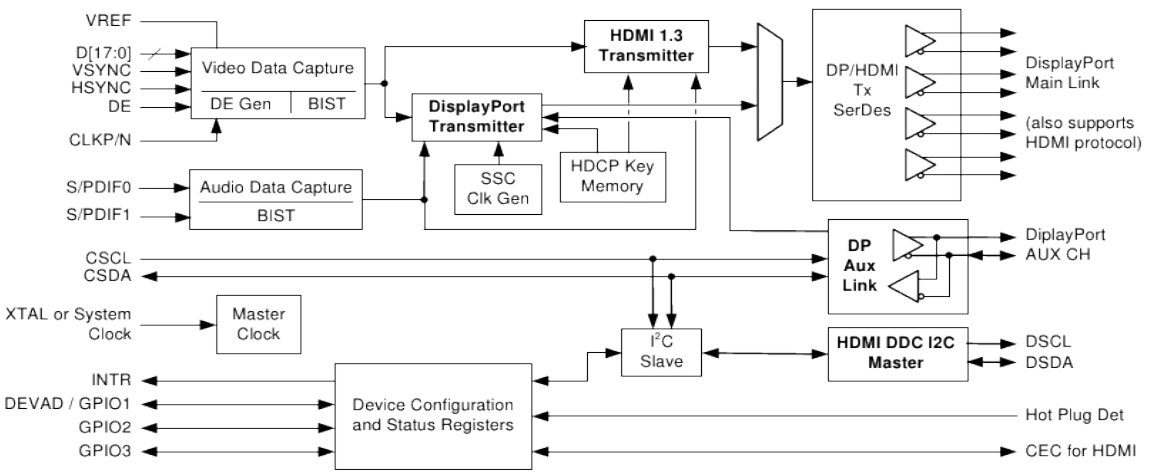
- Example: External DP/HDMI transmitter

RGB888, Sync

Audio

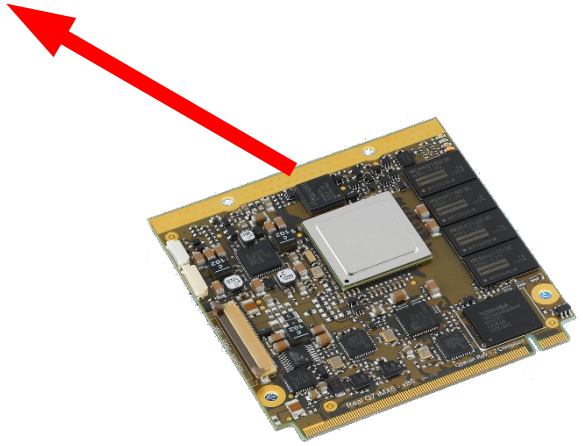
I2C

GPIO



- Converter chip family:
  Originally developed for AMD Fusion

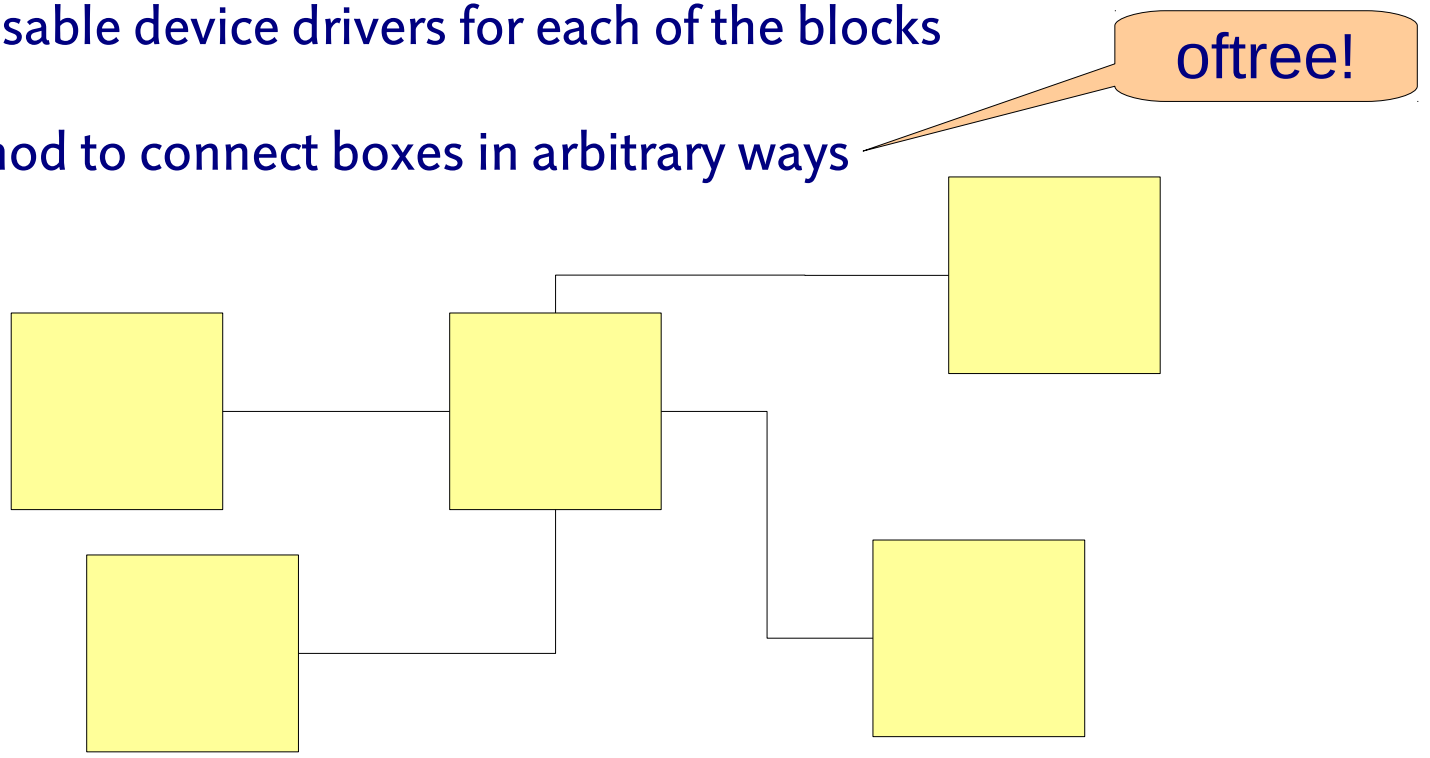
- Here: MX6



Pengutronix.

Wanted: Drivers for Connectable Blocks

- Re-Usable device drivers for each of the blocks
- Method to connect boxes in arbitrary ways

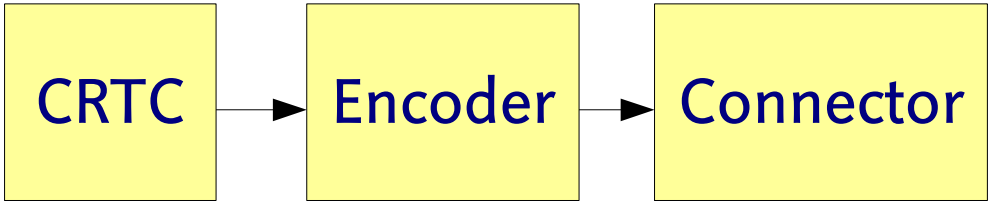


- Similar problems, one step ahead:
  ALSA → ASoC
  V4L2 → SoC-Camera, V4L2 subdevices, ...

Pengutronix.

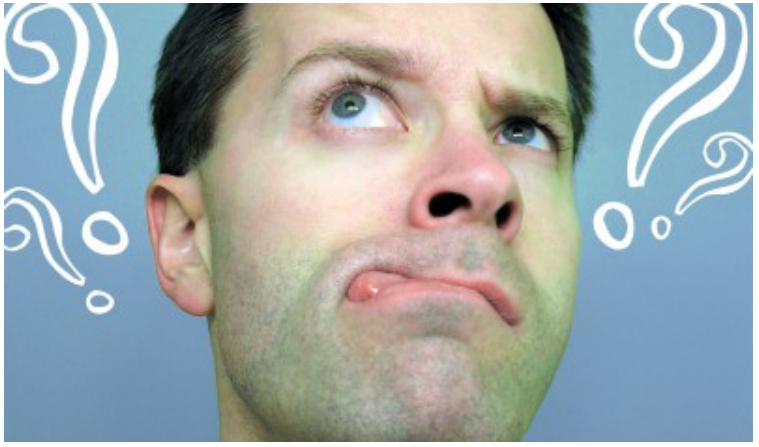
The Present: drivers/gpu/drm

- CRTC (Cathode Ray Tube Controller)
  - Samples frame buffers (or parts thereof)
  - Pushes pixels to an encoder
- Encoder
  - Encodes pixels into video signals for off-chip transmission
- Connector
  - Hotplug
  - Mode detection



The Present: drivers/gpu/drm

- How to implement a SoC with DRM?



1st Attempt: Generic SDRM Driver

- Tried to implement one DRM driver that registers components

```
The following changes since commit 0034102808e0dbbf3a2394b82b1bb40b5778de9e:

  Linux 3.4-rc2 (2012-04-07 18:30:41 -0700)

are available in the git repository at:

  git://git.pengutronix.de/git/imx/linux-2.6.git gpu/sdrm

for you to fetch changes up to fc3d0ff4825de998f1fd902184f7df040248d0de:

  DRM: add PXA kms simple driver (2012-04-11 17:10:46 +0200)

----------------------------------------------------------------
Philipp Zabel (1):
      DRM: add PXA kms simple driver

Sascha Hauer (6):
      drm: remove legacy mode_group handling
      drm: make gamma_set optional
      DRM: add sdrm layer for general embedded system support
      DRM: Add sdrm 1:1 encoder - connector helper
      DRM: add i.MX kms simple driver
      ARM i.MX27 pcm038: Add sdrm support
```

- Not accepted ...

2nd Attempt: drivers/staging/imx-drm

```
rsc@thebe:linux$ ls -l drivers/staging/imx-drm/
total 72
-rw-r--r-- 1 rsc ptx   960 Oct 28 11:26 Kconfig
-rw-r--r-- 1 rsc ptx   272 Oct 28 11:26 Makefile
-rw-r--r-- 1 rsc ptx   840 Oct 28 11:26 TODO
-rw-r--r-- 1 rsc ptx 20121 Oct 28 11:26 imx-drm-core.c
-rw-r--r-- 1 rsc ptx  1954 Oct 28 11:26 imx-drm.h
-rw-r--r-- 1 rsc ptx  1389 Oct 28 11:26 imx-fb.c
-rw-r--r-- 1 rsc ptx  1849 Oct 28 11:26 imx-fbdev.c
drwxr-sr-x 2 rsc ptx  4096 Oct 28 11:26 ipu-v3
-rw-r--r-- 1 rsc ptx 13753 Oct 28 11:26 ipuv3-crtc.c
-rw-r--r-- 1 rsc ptx  7071 Oct 28 11:26 parallel-display.c
```

Implements DRM Device

How to deal with modularity?

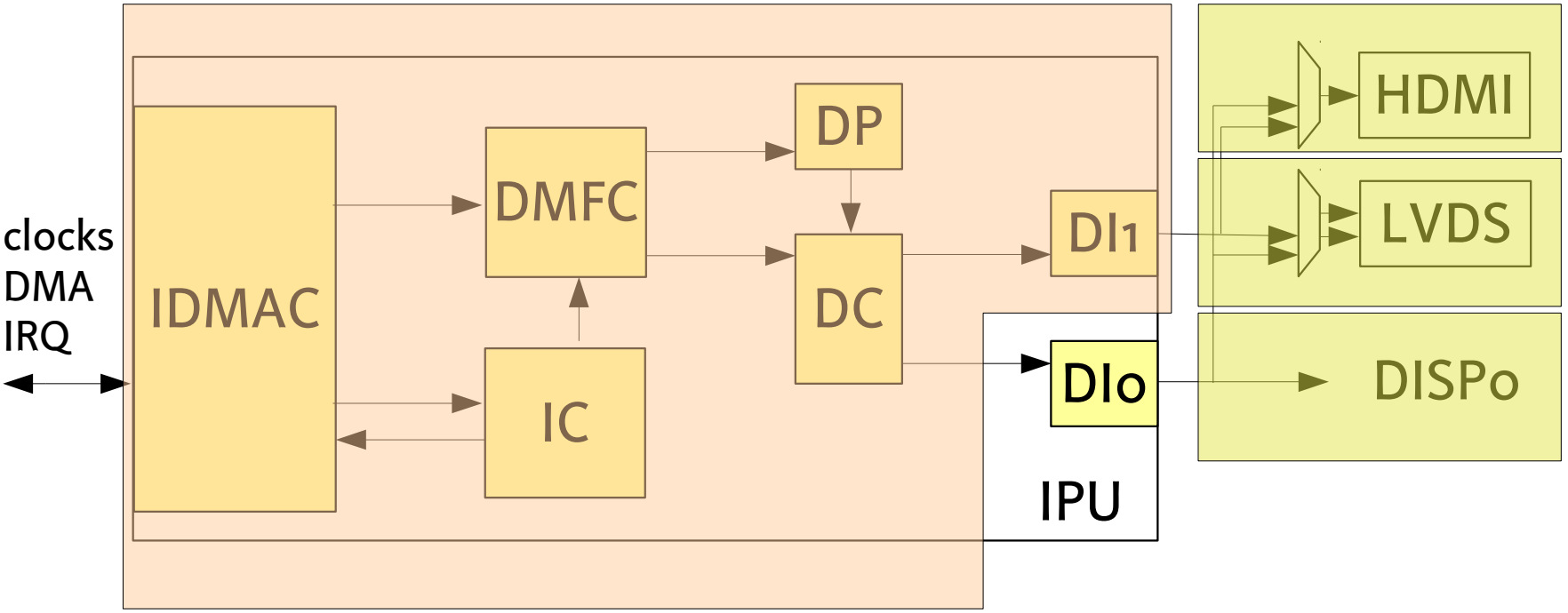
2nd Attempt: drivers/staging/imx-drm

```
static int __init imx_drm_init(void)
{
        int ret;

        imx_drm_device = kzalloc(sizeof(*imx_drm_device), GFP_KERNEL);
        if (!imx_drm_device)
                return -ENOMEM;

        mutex_init(&imx_drm_device->mutex);
        INIT_LIST_HEAD(&imx_drm_device->crtc_list);
        INIT_LIST_HEAD(&imx_drm_device->connector_list);
        INIT_LIST_HEAD(&imx_drm_device->encoder_list);

        imx_drm_pdev = platform_device_register_simple("imx-drm", -1, NULL, 0);
```

register lists

setup global DRM device

2nd Attempt: drivers/staging/imx-drm



CRTC

Encoder +
Connector

Pengutronix.

Issues Faced

- Encoder / connector / crtc matching with devices / device tree nodes
  - oftree describes hardware topology, but no „order“
  - DRM: encoder has bitfield („possible_crtcs“), corresponding to the CRTCs as ordered in the drm_device's internal crtc_list
  - CRTCs, Encoders, Connectors are no „devices“
    -> not „the Linux way“
    -> difficult to match with oftree components
  - We would like to load the modules in arbitrary order
    -> needs one instance that knows topology
  - Clock routing + multiplexing needs topology as well ...

Modeling Graphics Components in oftree

```
display@di0 {
        compatible = "fsl,imx-parallel-display";
        crtc = <&ipu 0>;
        interface-pix-fmt = "rgb24";
        /* timing description ... */
};

ipu: ipu@18000000 {
        #crtc-cells = <1>;
        compatible = "fsl,imx6q-ipu";
        reg = <0x18000000 0x080000000>;
        interrupts = <11 10>;
};
```

Modeling Graphics Components in oftree

```
hdmi {
        compatible = "fsl,imx-hdmi";
        crtc = <&ipu1 0 &ipu1 1 &ipu2 0 &ipu2 1>;
        /* timing from monitor EDID */
        ddc = <&i2c2>
};

ipu1: ipu@18000000 {
        #crtc-cells = <1>;
        compatible = "fsl,imx6q-ipu";
};
```

Conclusion

- DRM is the way-to-go for modern graphics
- Its monolithic approach doesn't fit the modular SoC hardware well
- Doesn't even fit modern PC hardware Embedded is NOT different!
- We need more modularity -> avoid current copy-paste attitude
- Need to look at other frameworks with similar challenges:
  -> ASoC, SoC-Camera, V4L2 Subdevices ...

Let's try to reuse and improve on existing concepts!

The Future: Current + Next Activities

- The imx-drm driver is in staging now
  -> We would like to get your feedback!
- Laurent Pinchart:
  „Generic Panel Framework“
- Steffen Trumtrar:
  „Display Timing Helpers“
- Guennadi Liakhovetski:
  „oftree Bindings for SoC Camera“

Join the discussion at dri-devel!