Video4Linux:

Past

Present

Future

Hans Verkuil

Tandberg Telecom AS

Past: Statistics

Until kernel 2.6 the v4l subsystem was small: 1-2% of all drivers.

2.6 added dvb and many new v4l drivers and it has grown from 5% to >9%.

Third-largest subsystem after scsi and net.

TV capture devices, radio receivers, webcams, MPEG encoder and decoder devices, DVB devices, System-on-a-Chip video devices, FM transmitters, supporting i2c devices.

Kernel	v4l-dvb tree	% of all drivers
2.0	287953	1.00%
2.2.26	643187	1.30%
2.4.0	829547	1.50%
2.4.36	1477303	1.70%
2.6.0	4634875	5.80%
2.6.10	4232200	4.60%
2.6.16	5569219	5.40%
2.6.17	7189484	7.00%
2.6.18	7910318	7.70%
2.6.19	8081530	7.60%
2.6.20	8419446	7.80%
2.6.21	8556067	7.80%
2.6.22	9121077	8.10%
2.6.23	9334386	8.20%
2.6.24	9667264	8.10%
2.6.25	10121497	8.10%
2.6.26	11244052	8.90%
2.6.27	12322637	9.60%
2.6.28	12947608	9.30%
2.6.29	13800424	8.70%
2.6.30	14866804	8.90%
2.6.31	15457024	9.00%
2.6.32-rc3	16464660	9.28%

Present: state of V4L2

V4L2 public API: pretty good. Proven to be reasonably future proof except for SoCs.

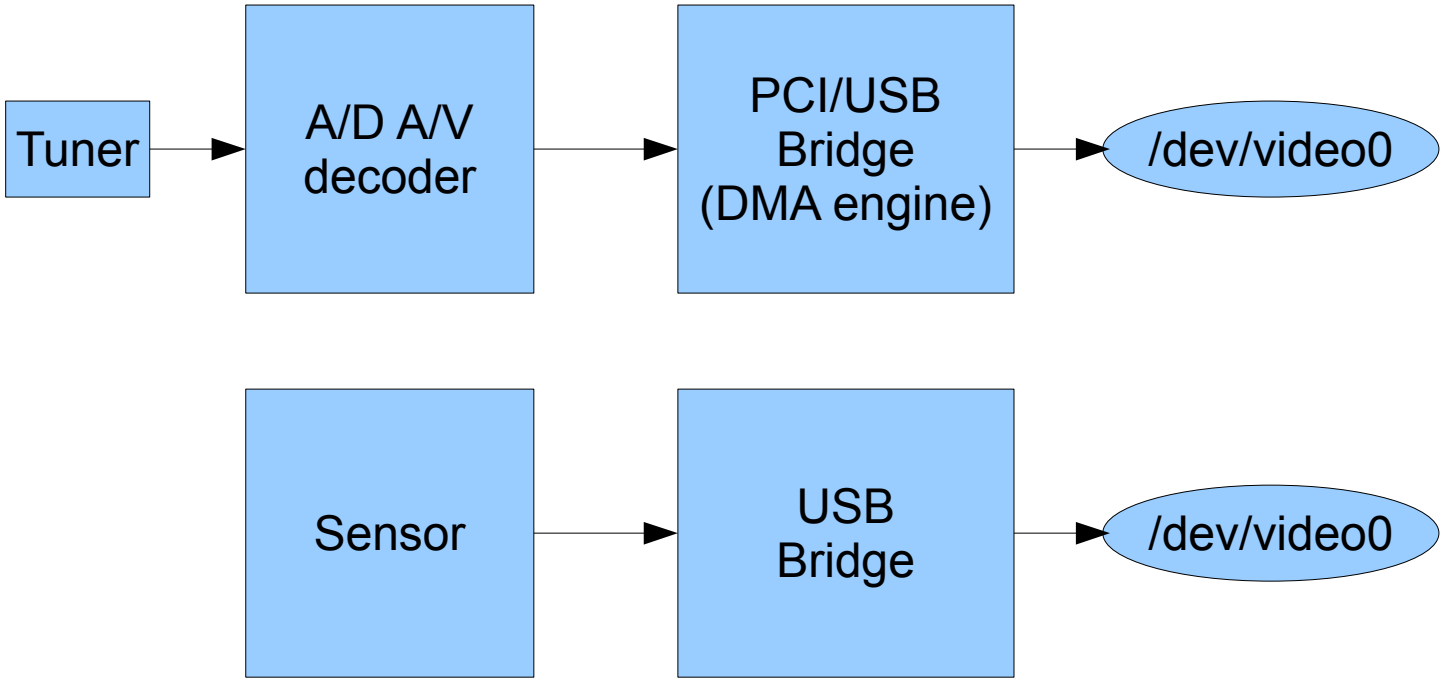
V4L drivers: anything from plain broken to excellent. Esp. the older, more obscure drivers are in poor condition, though.

Drivers suffer from reinventing the wheel due to lack of v4l core services.

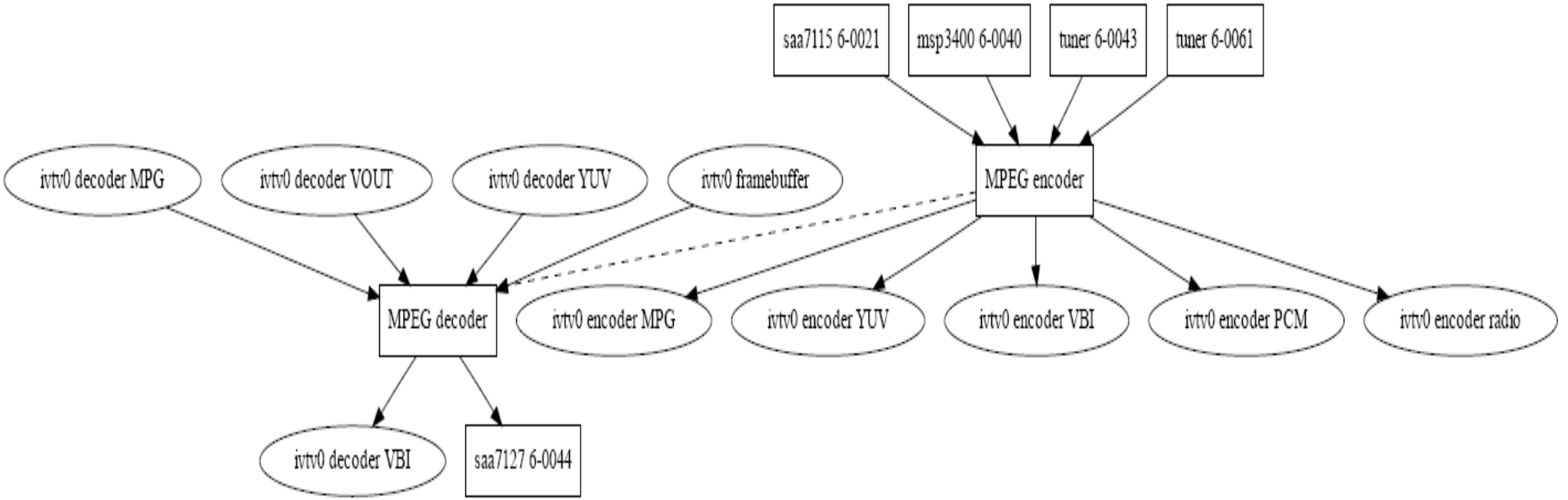
Often drivers are clearly copied from earlier drivers and so feature the same bugs.

Lack of driver compliance tests makes it hard to verify whether a driver works correctly.

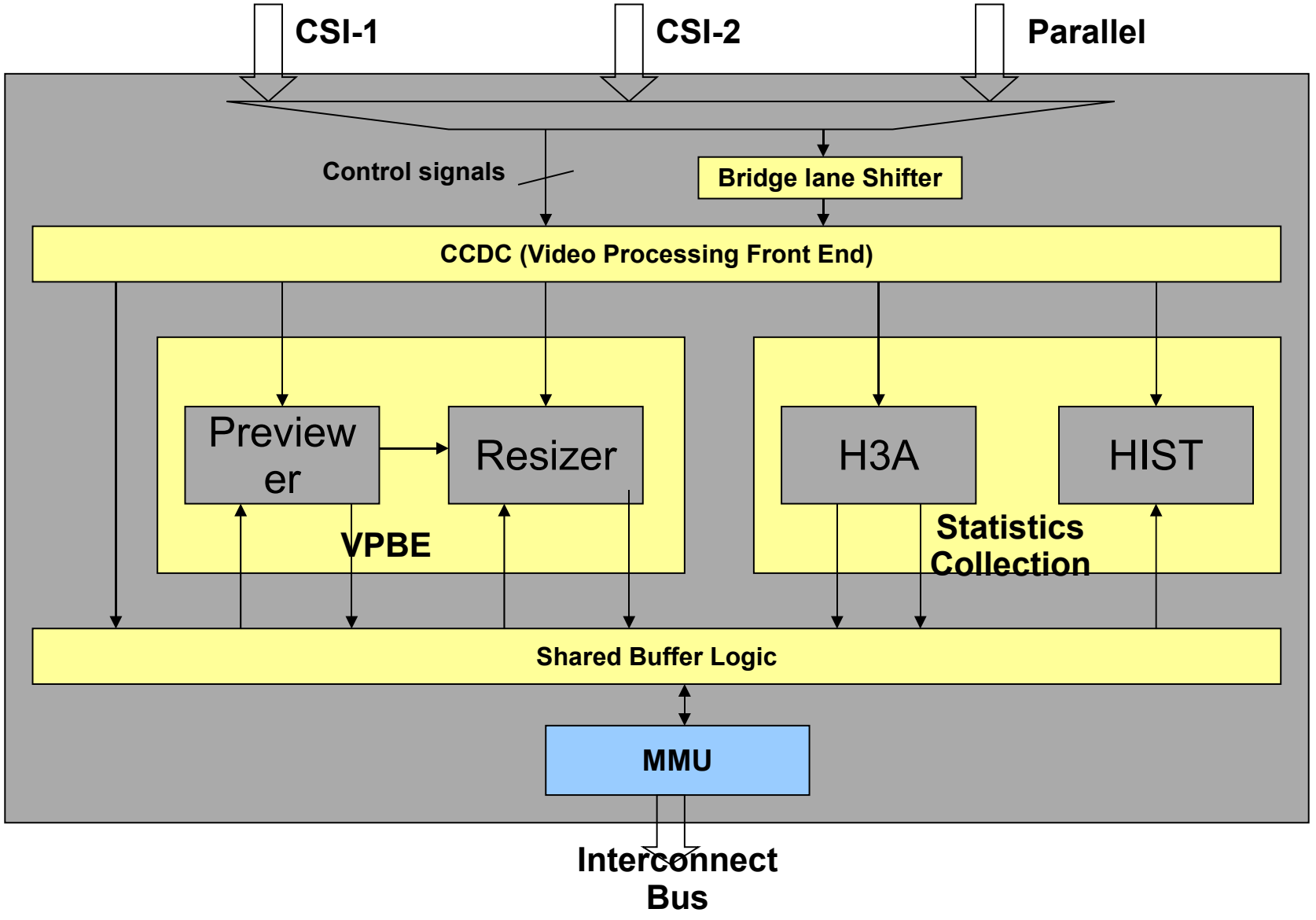
Present: typical hardware



Present: complex hardware

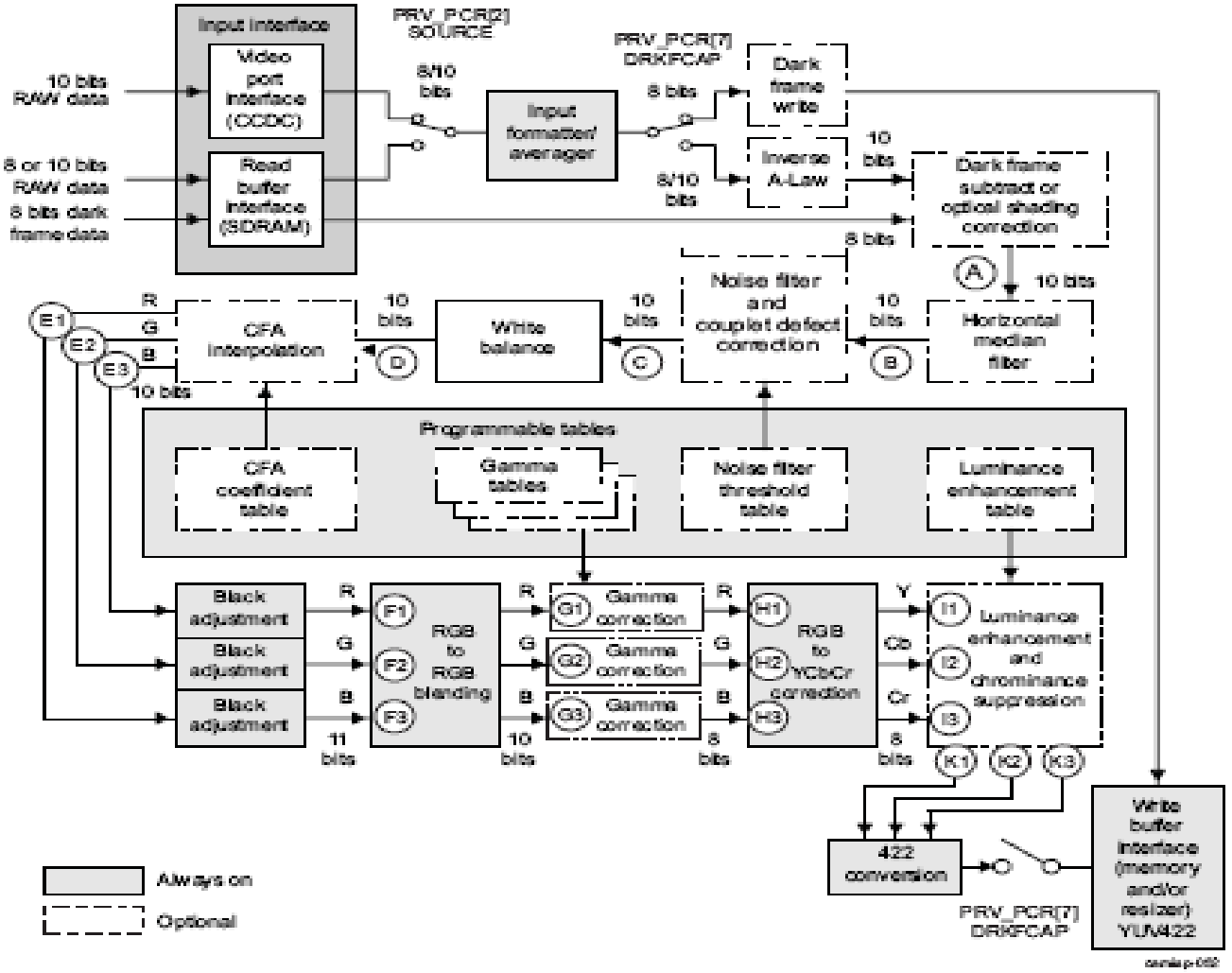


Future: TI OMAP3 ISP

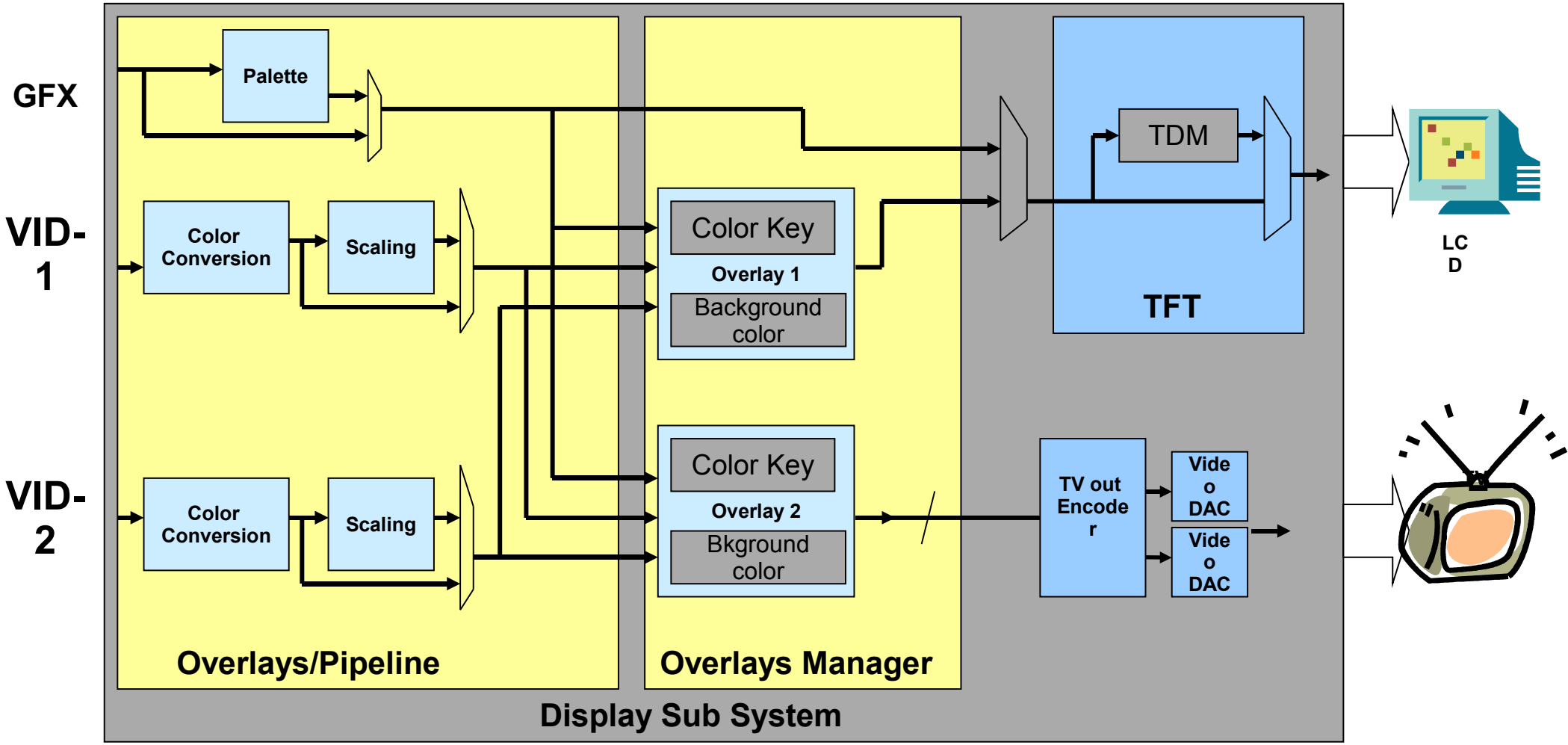


Future: TI OMAP3 Previewer

Figure 12-78. Preview Engine Block Diagram



Future: TI OMAP3 Display Controller



Future: SoC video devices

Very complex devices.

Multiple video and graphics streams.

Flexible video stream routing.

Applications require much more control.

Digital cameras, mobile phones, media players, TVs, surveillance applications, video conferencing, in-flight entertainment, etc., etc.

V4L2 API does not support these advanced devices, SoC manufacturers make their own custom drivers.

Past: SoC support developments

Initial talks with Texas Instruments Spring/Summer 2008.

First RFC July 2008.

Discussed during the Linux Plumbers Conference in August 2008.

Core framework required: mostly done by 2.6.30 (June 2009).

Organized a v4l-dvb mini-summit during the Linux Plumbers Conference in September 2009.

Participants from Texas Instruments, Intel, Nokia, Samsung and independent v4l-dvb developers.

Agreed on how to proceed with V4L SoC support.

Present: core framework

Created a struct v4l2_device for basic device-global data.

Created a struct v4l2_subdev to communicate with (usually i2c) sub-devices. Register them with v4l2_device. When v4l2_device is removed, unregister the sub-devices automatically.

The 'sub-device' concept is an abstract concept: it does not care on what (if any) bus the sub-device is located.

Ensures a unified API towards sub-devices to make it easy to swap one chip for another.

Ensures re-use of the same sub-device driver.

Can also be used to expose internals of the video subsystem of a SoC.

Future: results from mini-summit

New timings API for HDTV and other high-definition sources and sinks.

Standard event passing API.

New core control framework.

Buffer pool management API.

(Support for multi-planar framebuffers).

Media controller.

Future: Timings API

Has presets for the common standards (e.g. 720p30, 1080p60, etc.)

Can get/set/query the preset.

If the hardware allows it, then it must also be possible to set custom timings (front & backporch, sync widths, pixelclock).

Future: Events API

Standard API for V4L2 events.

Can be used with select() since it arrives as an exception.

Per-filehandle event queue and event subscription.

Future: control framework

Too much hard work for drivers.

A lot of code duplication.

A lot of buggy code.

Inconsistencies between drivers.

All controls go through the new framework.

A driver only needs to supply a s_ctrl function in most cases.

Future: Memory Pool

Prevent unnecessary cache and memory locking operations.

Allow easy transfer of buffers from one video device node to another.

We want this to be a global pool.

Lots of unresolved issues yet: needs more research.

Future: media controller

Modern v4l drivers often also support framebuffer, alsa, i2c, lirc and/or dvb devices, hard to keep track of by applications. Need some central authority to tell apps what is what.

Many SoCs can reroute the internal videostreams. E.g. capture from a sensor and do memory-to-memory resizing, or send the sensor output directly to the resizer.

Apps writing for SoCs want much more control about the various components of the device. A way is needed to provide that control without compromising the normal API which tends to hide complexity from the user.

Future: media controller

Create a /dev/v4l/mc0 device that can be used to enumerate the mesh-topology of the device.

Nodes in the mesh are sub-devices and device nodes. The general name for a mesh-node is entity.

The mc will also enumerate the possible and current links between entities.

The mc allows you to change the links.

Some sub-devices will have their own device node for advanced control (/dev/v4l/subdevX?).

Future: media controller

Current drivers for consumer products can use the mc to let apps discover which alsa device is associated with which v4l device.

For embedded devices (SoCs) the mc controls the internal data flow of the device.

For embedded devices (SoCs) the /dev/subdevX device nodes allow direct control of the advanced and hardware-specific features of sub-devices.

For each particular SoC a userspace library will be required to use the hardware optimally. This allows us to keep the kernel driver simple.

Future: Timeline

Timings and event API scheduled for 2.6.33.

Control framework for 2.6.33/34.

Memory pool and media controller for 2.6.34/35.

Questions?

e-mail:

hverkuil@xs4all.nl

linux-media mailinglist:

http://www.linuxtv.org/lists.php