DRM Driver Development For Embedded Systems

Inki Dae

Software Platform Lab.

Embedded Linux Conference, 26-28.10.2011

Contents

- What is DRM?
- PC vs Embedded Systems
- Advantages with DRM
- DRM KMS Framework
- Considerations for Embedded Systems
- References

DRM consists of

- KMS (Kernel Mode Setting)
  - Change resolution and depth.
- DRI (Direct Rendering Infrastructure)
  - Interfaces to access hardware directly.
- GEM (Graphics Execution Manager)
  - Buffer management
- DRM Driver in kernel side
  - Access hardware.

DRM Features

- Device-independent kernel-level device driver for XFree86 DRI.
- Kernel module that gives direct hardware access to DRI clients.
- Contains code intended to support the needs of complex graphics devices.
- Deals with DMA, AGP memory management, resource locking, and secure hardware access.

PC vs Embedded Systems

- PC usually has graphics card with its own video memory.
  - Graphics card includes display, hdmi, 2D/3D accelerators, and so on.
- Embedded system without such things.
  - no dedicated video memory.
- DRM framework is designed for PC.

Which were used by Embedded Systems without DRM?

- Linux Framebuffer for Display.
- V4L2 based device drivers for multimedia.
  - Rotator and Scaler
  - Video codec
  - HDMI
- Buffer managers such as UMP, HWMEM, CMEM, PMEM, ION, and so on.

What is the Advantages with DRM?

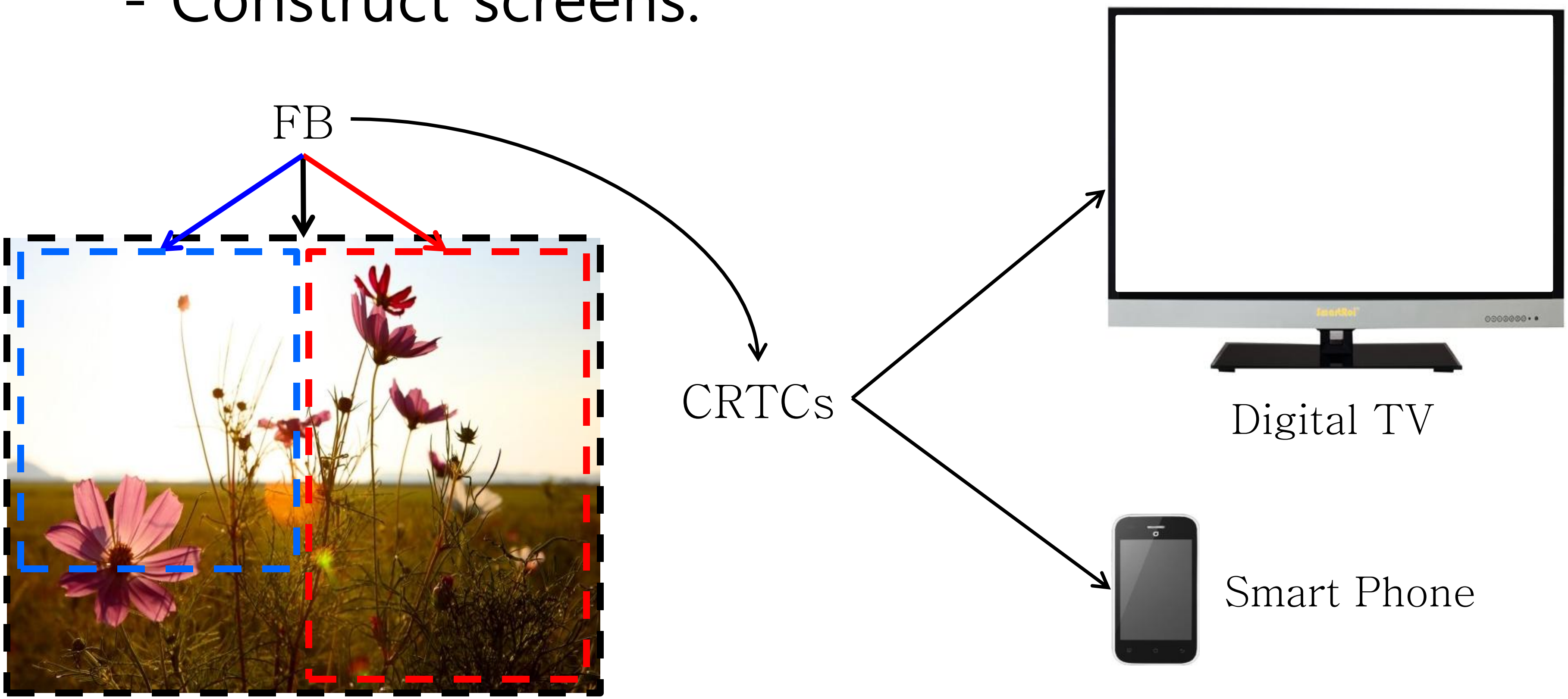
From interface point of view

- Control all HW thru a single device node.
- Common interfaces for hardware access
- Common interfaces for buffer management.

What is the Advantages with DRM?

From mechanism point of view

- Flexible Framebuffer and CRTC.
  - Construct screens.



How DRM Manages Buffer?

- GEM (Graphics Execution Manager)
  - Developed by Intel to manage graphics memory.
  - Framework for buffer management.
    - Buffer allocation and sharing.
  - Wrapper to buffer
  - Provide only interfaces and framework.

DRM KMS Framework

Consists of

- Framebuffer
- CRTC
- Encoder
- Connector

DRM KMS Framework

- Framebuffer
  - Memory information such as width, height, depth, bpp, pixel format, and so on.

DRM KMS Framework

- CRTC
  - Mode information.
    - resolution, depth, polarity, porch, refresh rate, and so on.
  - Information of the buffer region displayed.
  - Change current framebuffer to new one.

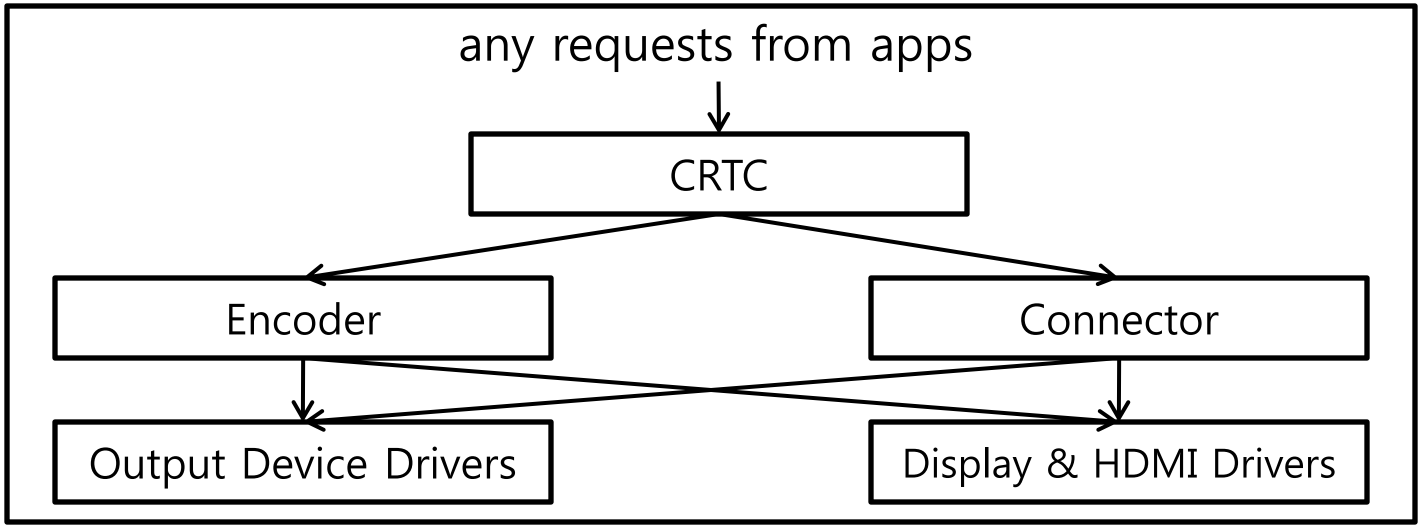
DRM KMS Framework

- Encoder
  - Take the digital bit-stream from the CRTC
  - Convert to the appropriate analog levels
    - for transmission across the connector to the monitor.
- Connector
  - Provide the appropriate physical plugs such as HDMI, DVI-D, VGA, S-Video, and so on.
- They are suitable to PC.

What is The Problem for Encoder and Connector?

- Display and HDMI controllers including all registers for setting.
  - Mode setting registers
  - Buffer setting registers
  - Power control registers
  - DMA control registers
- Cannot clearly split encoder and connector.

How To Mitigate?



- CRTC used commonly
  - call common callbacks of encoder or connector.
- Encoder and Connector with hardware specific callbacks.
  - call hardware specific callbacks of device drivers.

How to Implement Encoder?

Provide callbacks to

- Control *hardware overlays*
  - Setup, enable and disable them.
- Control the *power* to Display and HDMI controllers.

How to Implement Connector?

Provide callbacks to

- Control the *timing* to output devices.
- Control the *power* to output devices.
- Control the *connection* to output devices.

PC vs Embedded Systems For Buffer Management?

- PC graphics cards usually have VRAM.
  - Need pin/unpin for sync between system memory and video memory.
- Most Embedded Systems without VRAM
  - Don’t need pin/unpin.

PC vs Embedded Systems For Buffer Management?

- Most graphics cards use IOMMU.
  - Use non-continuous memory.
  - Map user space to physical pages at page fault handler.
- Most Embedded Systems without IOMMU.
  - Need physically continuous memory allocation from system memory.
  - Need direct mapping feature.

How To Implement GEM Framework For Embedded Systems?

- Physically continuous allocation
  - CMA (Continuous Memory Allocator)
- Direct mapping feature
  - Add a GEM ioctl command.
    - Map user space and physical memory once user requests directly.
    - Similar to mmap system call.

Issue on Reusing Existing Codes

- Same device drivers already exist to mainline.
  - Linux framebuffer and v4l2 based drivers.
  - Reuse them!
- Driver probing order.
  - Separated Platform/I2C Devices.
    - Display and HDMI controller
    - I2C based DDC(Display Data Channel) and HDMIPHY controller.
  - Some driver should be probed prior to another one.

DRM Features being Added for Embedded Systems

- Multiple Overlays
  - Have its own pixel format, size, and so on.
  - Overlays controlled respectively.
- What is the problem?
  - Supported one overlay.
    - use *plane feature* introduced by Jesse Barnes to control multiple overlays.

DRM Features being Added for Embedded Systems

- Multiple IRQ
  - Have hardware IRQ more than one.
    - Display and HDMI controller.
- What is the problem?
  - Supported one hardware IRQ.
    - request_irq() by each hardware specific driver.

DRM Features being Added for Embedded Systems

- Multi Planer
  - Support various pixel formats.
    - RGB, YUV, NV12, NV16, and so on.
  - NV12M, NV12MT and YUV420M
    - Need non-contiguous two or three buffers.
- What is the problem?
  - Not support NV12M, NV12MT and YUV420M format types.
  - Have one plane per buffer.

References

- DRM driver for Samsung Exynos SoC
  - http://git.infradead.org/users/kmpark/linux-2.6-samsung
  - branch name: samsung-drm
- Deferred Driver Probing introduced by Grant Likely
  - http://lwn.net/Articles/450178/
- DRM plane feature
  - http://lists.freedesktop.org/archives/dri-devel/2011-June/011855.html

Thank You

Contact Information:
inki.dae@samsung.com
daeinki@gmail.com