HWComposer in Weston

Anand Balagopalakrishnan
Architect, Automotive Graphics
Texas Instruments
Outline

- Automotive Use Cases
- Graphics Stack – Linux vs Android
- HWComposer
  - Overview
  - Composition Policies
- Discussion Topics
- Reference
Automotive Use cases
Cars of Today / Tomorrow
Feature Sets

• ADAS
• Center Console
• Digital Instrument Cluster
• Heads Up Display
• Navigation
• Media Player
• Augmented Reality
• Connectivity
• Early Video – Rear View Camera
Graphics Requirements

- Window System => 10-12 layers per app
- Multiple displays => 2-3
- Streaming Video textures
- Efficient GFX buffer management
- DRM master “workaround”
- Compute => OpenCL / GPGPU
- Efficient Composition
- OpenGL ES 2.0 +
Graphics Stacks
Linux Graphics Stack

Applications
- Navigation
- Cluster
- RSE
- ADAS

Window System
- Wayland / Weston

HAL
- LIBGBM

Foundation SW
- LIBDRM
- OpenGLES / EGL
- G2D

Kernel Drivers
- DRM
- SGX KM
- GC320

HW IP
- DSS
- 3D GPU
- 2D GPU
Android Graphics Stack

Applications
- Navigation
- Cluster
- RSE
- ADAS

Layer Manager
- Surface Flinger

HAL
- GRALLOC
- HWComposer

Foundation SW
- LIBDRM
- OpenGLES / EGL
- G2D

Kernel Drivers
- DRM
- SGX KM
- GC320

HW IP
- DSS
- 3D GPU
- 2D GPU
Status Quo

• Good convergence at lower layers of Graphics stack
• Android
  – Switched to KMS for modesetting
  – GRALLOC still the interface but internally uses GEM instead of ION
  – HWComposer uses GRALLOC flags
• Linux
  – Weston integrates well with OpenGLES / DRM for embedded SoCs
  – No well defined interface for HW composition in Weston (by design)
Android Composition

HWComposer Overview

• Clean, well-documented Interface
• [https://github.com/android/platform_hardware_libhardware/blob/master/include/hardware/hwcomposer.h](https://github.com/android/platform_hardware_libhardware/blob/master/include/hardware/hwcomposer.h)

• Two main APIs
  – prepare
    • Identify which layers can be composited by HW
    • Rest of layers go through OpenGLES composition
    • Hints to specify behavior
  – set
    • Trigger composition of selected layers in HW
    • Display to screen
HWComposer Policies

- Decide based on DDR bandwidth, power, GPU offload
- Policy considerations
  - Pixel format is important – YUV => DSS better for most modern HW
  - Small layers => 2D GPU / blitter
  - Layer frame rate - Higher FPS => DSS
  - Layer size – Higher => DSS
  - Z-Order – Lower => 3D GPU (HSR beneficial)
  - Transparency – Per pixel Alpha => DSS
Discussion Topics
Areas

• New compositor backend implementation
  – compositor-hwcomposer.c

• Sync between OpenGLES composition and HW composition
  – Explicit vs Implicit sync?

• Modeset nodes
  – Multiple masters – one per CRTC?

• Graphics Buffer Management
  – GRALLOC vs GBM
  – Not a major concern – can be handled inside HWcomposer

• Relevant product space / Industry interest
  – Anyone with stakes in both Linux and Android?
  – With sufficient interest, can engage Linaro
Reference

- http://wayland.freedesktop.org/
Questions?