Unifying Android and Mainline Kernel Graphics Stack

Gustavo Padovan
Principal Software Engineer @ Collabora Ltd
Agenda

- The Android World
- DRM Atomic Modesetting
- Mainline Explicit Synchronization
The Android World
Atomic Display Framework (ADF)

- Add atomic updates of planes
- Add custom Pixel Format
- Driver-specific blob
- Add explicit synchronization
Explicit Synchronization
Sync Framework

- Use fd for fence passing
- Sync Timeline to control ordering
- Sync Point to represent a fence
- Sync Fence for fd passing
Sync Timeline

- Monotonically increasing counter
- Usually one timeline per driver context
Sync Point

- It is the fence
- Represents a value on the timeline
- Three states: active, signaled and error
Sync Fence

- Wrap Sync Point into a file
- Also have active, signaled and error states
- Shared via fd-passing to/from userspace
Sync Fence

- Sync fences can be merged!
- It can contain many Sync Points
Sync Framework – userspace API

- `sync_wait(fd, timeout)`
- `fd3 = sync_merge(fd1, fd2)`
- `sync_fence_info(fd, info)`
ADF and Mainline

- Single update queue
- No atomic operation for modeset
- Inflexible mid-layer
- Driver specific blob
- New and not generic userspace API
DRM Atomic Modesetting

- Atomic plane updates
- Atomic modesets
- Single IOCTL using property infrastructure
- Check and commit phases
- DRM_ATOMIC_TEST_ONLY flag
- Easily extensible through helpers
Sync Framework de-staging

- Android Sync added to staging in 2013
- Mainly need for fd-passing
- Removed Sync Timeline
- Removed Sync Point
- Reworked Sync Fence
Sync File

- Renamed Sync Fence to Sync File
- Changed ioctl API
  - Provided patch to Android's libsync
- Removed internal kernel API
- Used strictly for fd-passing
  - \texttt{sync\_file = sync\_file\_create(fence)}
  - \texttt{fence = sync\_file\_get\_fence(fd)}
DRM/KMS Explicit Synchronization

- Entirely in DRM Core
- Only available for Atomic Modesetting
- Extended DRM Properties
- In-fences: fences to wait before scanout
- Out-fences: fences create and signaled by KMS
- It is on Linux v4.10
DRM/KMS: in-fences

- Fences to wait on before scanout
- IN_FENCE_FD property on each DRM Plane
- Receive sync_file fds carrying fences
- drm_atomic_helper_wait_for_fences()
DRM/KMS: out-fences

- Fences created and signaled by KMS
- OUT_FENCE_PTR property on each DRM CRTC
- It signals at CRTC scanout (VSYNC)
  - The previous buffer can be reused
DRM/renderer

• Similar to KMS side
• Extends execbuf ioctl on each driver
• Add sync_file/fences support on each driver
• Done on Freedreno
• WIP on i915, virgl and others
Android HWComposer API

- HAL to abstract the display driver
- HWC1.x supported speculative fences
- HWC2 supports non-speculative fences
drm_hwcomposer

- HWC that uses DRM Atomic Modesetting
- Ongoing work to support Explicit Fences
- git://git.collabora.com/git/user/robertfoss/drm_hwcomposer.git
Mesa

- EGL_ANDROID_native_fence_sync
  - Receive out-fence fd
- EGL_ANDROID_wait_sync
  - Wait for in-fence to signal
Thank you to everyone involved

Daniel Vetter, Rob Clark, Greg KH, Daniel Stone, Robert Foss, Sean Paul, Stéphane Marchesin, Greg Hackmann, Jesse Hall, Maarten Lankhorst, Chris Wilson, Christian König and others.
Thank you!

Gustavo Padovan

gustavo@padovan.org

www.padovan.org

www.collabora.com