Mainline Explicit Fencing

A new era for graphics

Gustavo Padovan
Agenda

• Introduction
• Android Sync Framework
• Mainline Explicit Fencing
• Current Status
Fencing

- Ensure ordering between operations
- Synchronize buffer sharing
  - e.g.: Between GPU and Display drivers
- Allow async operations
- Implicit fencing: userspace not aware
- Explicit fencing: userspace aware
Implicit Fencing

• No userspace knowledge/interference
• Simple/Dumb compositors
  - No buffer state information
• But it can freeze the whole desktop!
Implicit Fencing

- Buffer C will be composed of A and B
- Buffers A and B can render in Parallel
- Compositor notified only when both finishes
Implicit Fencing

- A is fast and B takes too long
- C is blocked waiting for both to render
- The entire desktop freezes!
Explicit Fencing

- Fences goes to userspace
- Userspace can control synchronization
- Smart decisions on compositors
- Avoid blocking the entire desktop
Explicit Fencing

- Consumer/Producer without blocking
Explicit Fencing

- Better for traceability/debuggability
- Vulkan needs it
  - Part of the API
  - Efficient subbuffer processing
Android Sync Framework

- Android's Explicit Fencing implementation
- 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 and signaled states
- Shared via fd-passing to/from userspace
Sync Fence

- Sync fences can be merged!
- It can contain many Sync Points
Android Sync Framework - ioctlS

- sync_wait(fd, timeout)
- fd3 = sync_merge(fd1, fd2)
- sync_fence_info(fd, info)
Mainline Explicit Fencing

- Started with the fence synchronization mechanism by Maarten Lankhorst
- Buffer synchronization between drivers
struct fence

• fence->context
• fence_signal()
• fence_wait()
• fence_add_callback()
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
  – `sync_file = sync_file_create(fence)`
  – `fence = sync_file_get_fence(fd)`
struct fence_array

- Subclass of struct fence
- Store multiple fences
- Useful for merged Sync File
- Hide complexity from the drivers
- fence_is_array(fence)
DRM/KMS

- Only available for Atomic Modesetting
- Entirely in DRM Core
- Extended Atomic IOCTL
- in-fences: received from userspace
- out-fences: sent to userspace
DRM/KMS: in-fences

- in-fences: fences received from userspace
- FENCE_FD property on each DRM Plane
- Receives sync_file fds carrying fences
- drm_atomic_helper_wait_for_fences() helper
DRM/KMS: out-fences

• One out-fence per DRM CRTC
• `get_unused_fd() + sync_file_create() + fd_install()`
• It signals at CRTC scanout
  – It means the `previous` buffer can be reused.
• Userspace need to ask for out-fence
  – `DRM_MODE_ATOMIC_OUT_FENCE` flag
  – `libdrm: drmModeAtomicAddOutFences()`
DRM/renderer

- Similar to KMS side
- Extends execbuffer ioctl args on each driver
- Every driver needs sync_file/fences support
- WIP on freedreno, i915 and virgl
Mesa

- EGL_ANDROID_native_fence_sync
  - Create fence fd
- EGL_ANDROID_wait_sync
  - Make the GPU wait for fence to signal
- WIP by Rob Clark
- kmscube running on explicit fencing
drm_hwcomposer2

- Already support DRM fences semantic
- Use it as example userspace for upstream
- WIP by Sean Paul and Robert Foss
Current Status Summary

- Sync File synchronization de-stage: DONE
- SW_SYNC validation de-stage: DONE
- fence_array: DONE
- DRM/KMS: WIP – 4.10?
- DRM/renderer: WIP – 4.10?
- MESA: WIP
- igt-gpu-tools: WIP
- drm_hwcomposer2: WIP
- Wayland: TODO
- V4L explicit fences: TODO
Thank you to everyone involved

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

Gustavo Padovan

gustavo@padovan.org

www.padovan.org

www.collabora.com