VKMS: Virtual Kernel Modesetting

Haneen Mohammed, Rodrigo Siqueira
About Us and Our Mentors

Haneen Mohammed

Rodrigo Siqueira

OUTREACHY

Google Summer of Code
About Us and Our Mentors

Mentors:

- Daniel Vetter
- Gustavo Padovan
- Sean Paul
- All the dri-devel community (❤️)
Kernel Mode-Setting

User Space

Kernel Space

Direct Rendering Manager

Video RAM

Command Queue

Buffer 1

Buffer 2

Buffer N

GPU
Why do we need VKMS?

- Increase DRM test coverage
- It may work as a tool to help graphics developers
Development Cycle
Development Cycle

Watch IGT

Test fail
Development Cycle

Watch IGT test fail

Write code to pass the test
Development Cycle

Watch IGT test fail

Write code to pass the test

Improve the code
Development Cycle

Watch IGT test fail

Improve the code

Write code to pass the test
Basic Features
Basic Features

Diagram showing the relationship between Framebuffer, Userspace, Primary Plane, and Cursor Plane.
Basic Features

Diagram:
- Userspace
  - Framebuffer
  - Primary Plane
  - CRTC
  - Encoder Virtual
  - Cursor Plane

Connections:
1. From Framebuffer to Primary Plane
2. From Primary Plane to CRTC
3. From CRTC to Encoder Virtual
4. From Encoder Virtual to Cursor Plane
Basic Features
Page Flip and Vblank

- Draw
- Back Buffer
- Primary Buffer
- Pointer
Page Flip and Vblank

**Draw**
- Back Buffer

**FLIP**
- Primary Buffer
  - Pointer

**Draw**
- Primary Buffer
  - Pointer
- Back Buffer
Vblank on VKMS

Simulating Vblank with Hrtimers
Vblank on VKMS

Simulating Vblank with Hrtimers

Period

16.6666
Vblank on VKMS

Simulating Vblank with Hrtimers

16.6666 + delay
Vblank on VKMS

Simulating Vblank with Hrtimers

16.6666 - delay
Vblank on VKMS

vkms_enable_vblank() → vkms_vblank_simulate() → _vblank_handle() → drm_crtc_handle_vblank() → vkms_get_vblank_timestamp()
VKMS without VBlank (Patch)

VBlank signaling is faked by

- `drm_send_vblank_event()`
- `vkms_enable_vblank()` → `vkms_vblank_simulate()` → `_vblank_handle()` → `vkms_get_vblank_timestamp()`
- `drm_crtc_handle_vblank()`
Vblank on VKMS

[siqueira@atma igt-host]$ sudo ./tests/kms_flip --run-subtest basic-plain-flip
IGT-Version: 1.22-ge29bd428 (x86_64) (Linux: 4.18.0-rc3-VKMS-RULES+ x86_64)
Using monotonic timestamps
DRM_IOCTL_I915_GEM_APERTURE failed: Invalid argument
Assuming 131072kB available aperture size.
May lead to reduced performance or incorrect rendering.
get chip id failed: -1 [22]
param: 4, val: 0
Beginning basic-plain-flip on pipe A, connector Virtual-1
  1024x768 60 1024 1048 1184 1344 768 777 806 0xa 0x48 65000

基本-plain-flip on pipe A, connector Virtual-1: PASSED

Subtest basic-plain-flip: SUCCESS (10.185s)
[siqueira@atma igt-host]$
Vblank on VKMS

```
[siqueira@atma igt-host]$ sudo ./tests/kms_flip --run-subtest wf_vblank-ts-check
IGT-Version: 1.22-ge29bd428 (x86_64) (Linux: 4.18.0-rc3-VKMS-RULES+ x86_64)
Using monotonic timestamps
DRM_IOCTL_I915_GEM_APERTURE failed: Invalid argument
Assuming 131072kB available aperture size.
May lead to reduced performance or incorrect rendering.
get chip id failed: -1 [22]
param: 4, val: 0
Beginning wf_vblank-ts-check on pipe A, connector Virtual-1
  1024x768 60 1024 1048 1184 1344 768 771 777 806 0xa 0x48 65000
Expected frametime: 16666us; measured 16665.6us +- 0.500us accuracy 0.01%

.............
Subtest wf_vblank-ts-check: SUCCESS (30.582s)
[siqueira@atma igt-host]$
```
CRC API Support With VKMS

- value = \texttt{crc}(displayed frame)
- Goal: Pass the following tests from the IGT test suite
  - \texttt{kms\_pipe\_crc\_basic}
  - \texttt{kms\_cursor\_crc}

commit framebuffer to the display
\[\downarrow\]
start collecting crcs
\[\downarrow\]
wait for N vblanks to collect N crcs
\[\downarrow\]
collect reported crcs
\[\downarrow\]
verify len(crcs) == N and all are equal to each other
CRC API Support in DRM

- Add the following to the drm_crtc vfuns table:
  - `verify_crc_source()`
  - `set_crc_source()
- `drm_crtc_add_crc_entry()``
- CRC API exposed at `/sys/kernel/debug/dri/0/crtc-N/crc` -> `control` and `data` files
for each pixel visible in screen {
    // DRM_FORMAT_XRGB8888
    pixel = \texttt{clear\_alpha\_channel}(pixel);
    crc = \texttt{compute\_crc}(crc, pixel);
}
Computing CRC

for each pixel visible in screen {
    // DRM_FORMAT_XRGB8888
    pixel = clear_alpha_channel(pixel);
    crc = crc32_le(crc, vaddr(pixel), sizeof(pixel));
}
for (i = src_y; i < src_y + src_h; ++i) {
    for (j = src_x; j < src_x + src_w; ++j) {
        v_offset = i * pitch;
        h_offset = j * cpp /* bytes per pixel */;
        src_offset = offset + v_offset + h_offset;
        memset(vaddr, src_offset + 24, 0, 8);
        crc = crc32_le(crc, vaddr + src_offset, sizeof(u32));
    }
}
blend(primary, cursor);
for each pixel visible in screen {
    // DRM_FORMAT_XRGB8888
    pixel = clear_alpha_channel(pixel);
    crc = compute_crc(crc, pixel);
}
10000 Foot View

- How to synchronize framebuffer update with crc computations and flip event?

```c
struct vkms_crc_data {
    ...
    struct drm_framebuffer fb;
    uint32_t src_x, src_y;
    uint32_t src_w, src_h;
    ...
};
```
CRC API Support With VKMS (Challenges)

- `vkms_plane_atomic_check()`
- `vkms_prepare_fb()`
- `vkms_crtc_atomic_begin()`
- `vkms_plane_atomic_update()`
- `vkms_crtc_atomic_flush()`

**atomic commit**

- `vkms_vblank_simulate()`
- `drm_handle_vblank()`
- `frame_num = vblank_count`
- `queue_work(crc_workq, &crc_work)`

**vblank interval**

- `send_vblank_event`

**hrtimer callback**

```c
struct vkms_crc_data {
    ...
    struct drm_framebuffer fb;
    uint32_t src_x, src_y;
    uint32_t src_w, src_h;
    ...
};
```
CRC API Support With VKMS (Solution)

```c
struct vkms_crc_data {
    ...
    struct drm_framebuffer fb;
    uint32_t src_x, src_y;
    uint32_t src_w, src_h;
    ...
};
```
CRC API Support With VKMS (results)

<table>
<thead>
<tr>
<th>kms_pipe_crc_basic</th>
<th>kms_cursor_crc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.  bad-source</td>
<td>1.  cursor-size-change</td>
</tr>
<tr>
<td>2.  read-crc-pipe-A</td>
<td>2.  cursor-64x64-onscreen</td>
</tr>
<tr>
<td>3.  read-crc-A-frame-sequence</td>
<td>3.  cursor-64x64-offscreen</td>
</tr>
<tr>
<td>4.  nonblocking-crc-pipe-A</td>
<td>4.  cursor-64x64-sliding</td>
</tr>
<tr>
<td>5.  nonblocking-crc-pipe-A-frame-sequence</td>
<td>5.  cursor-64x64-random</td>
</tr>
<tr>
<td></td>
<td>6.  cursor-64x64-dpms</td>
</tr>
</tbody>
</table>
Conclusion and Future Works

VKMS is a working in progress project. We still have to improve:

- There are some tests related to kms_flip that fails
- There are some improvements to make at the CRC part
- Make Wayland run on top of VKMS
- Many other features

Future works:

- Probably I will get 5 extra months of work in VKMS
THANKS
Recab:
1. Development workflow
2. VBlank Support
3. CRC API Support

Questions?