DRM bridge hotplug

Luca Ceresoli

luca.ceresoli@bootlin.com

Display Next Hackfest 2026

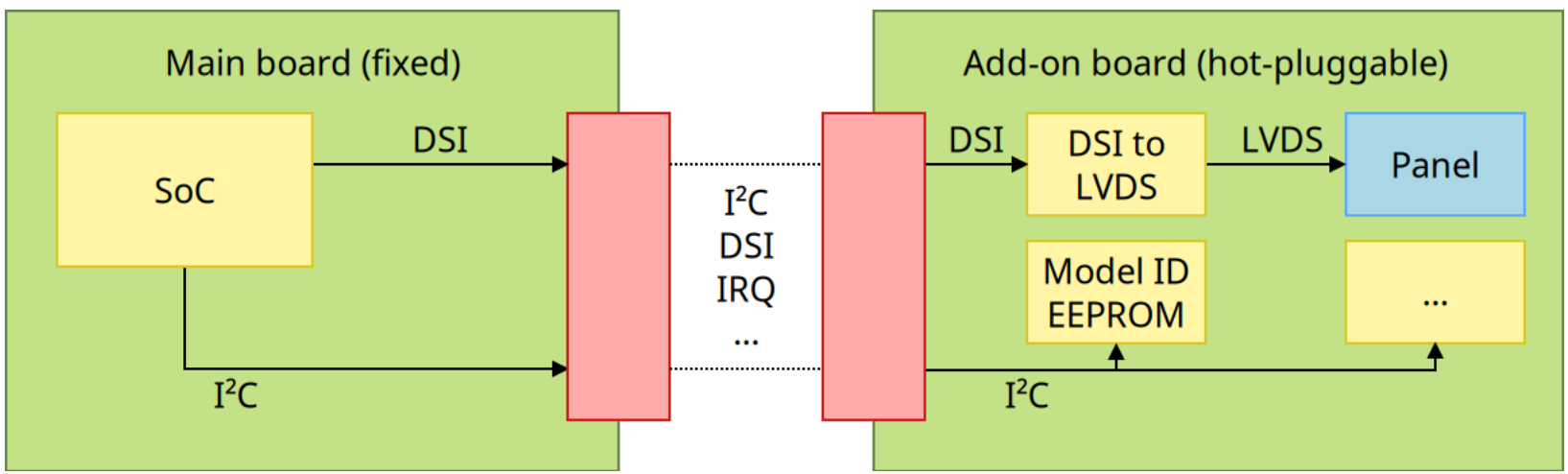
Luca Ceresoli

- Embedded Linux engineer at **Bootlin**
  - Development, consulting and training about **embedded Linux**
  - Open-source focus
- **Linux kernel** device driver developer
- Bootloaders, Buildroot and Yocto integration
- Open-source contributor
- Living in **Bergamo**, Italy

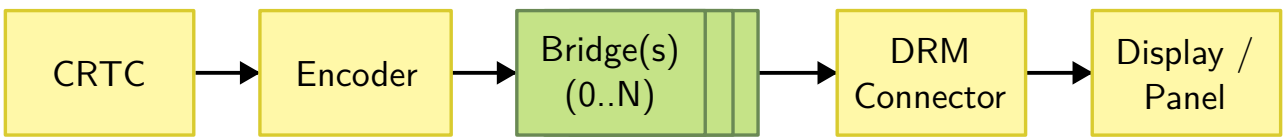
Use case and approach

Use case

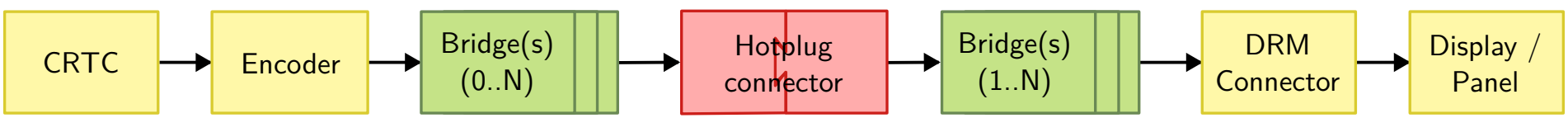
- Hot(un)pluggable addon with the tail of the video pipeline
  - Including 1+ bridge(s)
- Also covers coldplug with addon known at runtime (e.g. via DT overlay)



Hotplug DRM pipeline



A classic DRM pipeline



A DRM pipeline with hot-pluggable bridges

Approach

- Old approach: add a hotplug-bridge driver in the middle
  - Hotplug of Non-discoverable Hardware: Status and Future Directions, ELCE 2025 (`slides`, `video`)
  - Runtime hotplug on non-discoverable busses with device tree overlays, LPC 2024 (`https://lpc.events/event/18/contributions/1696/`)
- New approach: no hotplug-bridge driver, the DRM code must handle hotplug

Completed and ongoing work

Dynamic `drm_bridge` lifetime — Done ✓

- Pointers to `struct bridge` are taken by various parts of the DRM stack
- Added refcounting, converted the internal APIs
- Done (v6.16..v7.2)
- Exception: the `panel_bridge`
  - `drm_of_find_panel_or_bridge()` is problematic
  - `*_of_get_bridge()` are more problematic
  - Maxime Ripard had mentioned a plan to address the `panel_bridge`.
    - Perhaps every panel should have a wrapping bridge (or, better, an embedded bridge)
    - Any progress?

Handle bridge removal gracefully — Done ✓

- There are multiple *concurrent* entry points to a DRM bridge driver code
  - Atomic updates, hot-unplug
  - Add `drm_bridge_enter()` / `drm_bridge_exit()` to protect device resources (✓ v7.0)
- Add `drm_bridge_clear_and_put()` to avoid use-after-free more easily (✓ v7.1)
  - Example in `fbef867cf661 ("drm/bridge: samsung-dsim: use drm_bridge_clear_and_put() to put the next bridge")`

Handle bridge removal gracefully — Pending

- Protect `private_obj` removal from list
  - Patch on mailing list:

    `[PATCH v3] drm/atomic: protect bridge private_obj during bridge removal`
    - No reviews so far
    - Could be made more granular? (v4)

Current proposal and challenges

Current proposal

2026-05-19

[PATCH 00/37] drm bridge hotplug

22 files changed, 1078 insertions(+), 396 deletions(-)

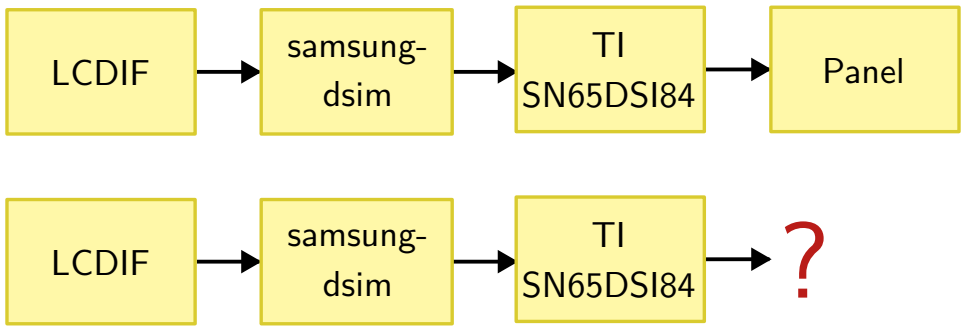
https://lore.kernel.org/r/20260519-drm-bridge-hotplug-v1-0-45e2bdb3dfb4@bootlin.com

Where?

- Which part of the DRM code should handle hotplug?
  - Old approach: dedicated `hotplug-bridge` device driver
- Chose to augment the **`drm_bridge_connector`** to handle hotplug
  - Currently the recommended component to implement the `drm_connector` for a bridge-based pipeline
  - `drm_bridge_connector` already handles `drm_connector` creation
    - and processes the whole bridge chain to do that
  - Adding/removing the `drm_connector` is central to DRM hotplug
- `Patch 36` (plus many preparatory patches)

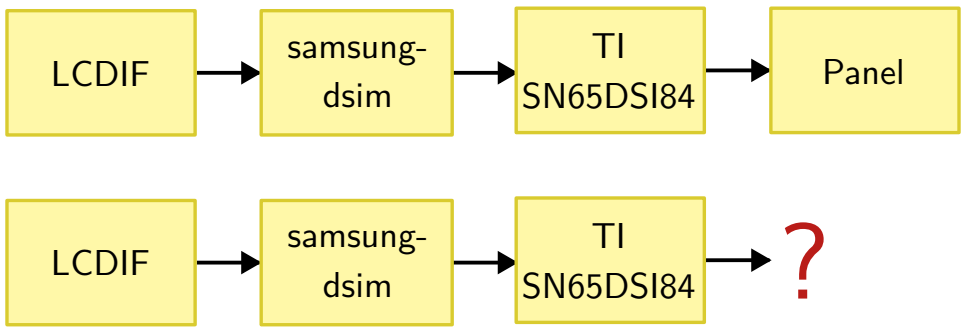
Implementation

Incomplete pipelines: problem



- Currently DRM always assumes a bridge chain is complete
  - *Complete pipeline* = it has all bridges up to a panel/connector
  - Incomplete pipelines won't even probe
- We need DRM to
  - be able to know whether a pipeline is complete or not (last bridge(s) still missing)
  - probe the card anyway (but with no `drm_connector`)
  - be ready to add the `drm_connector` on hotplug events

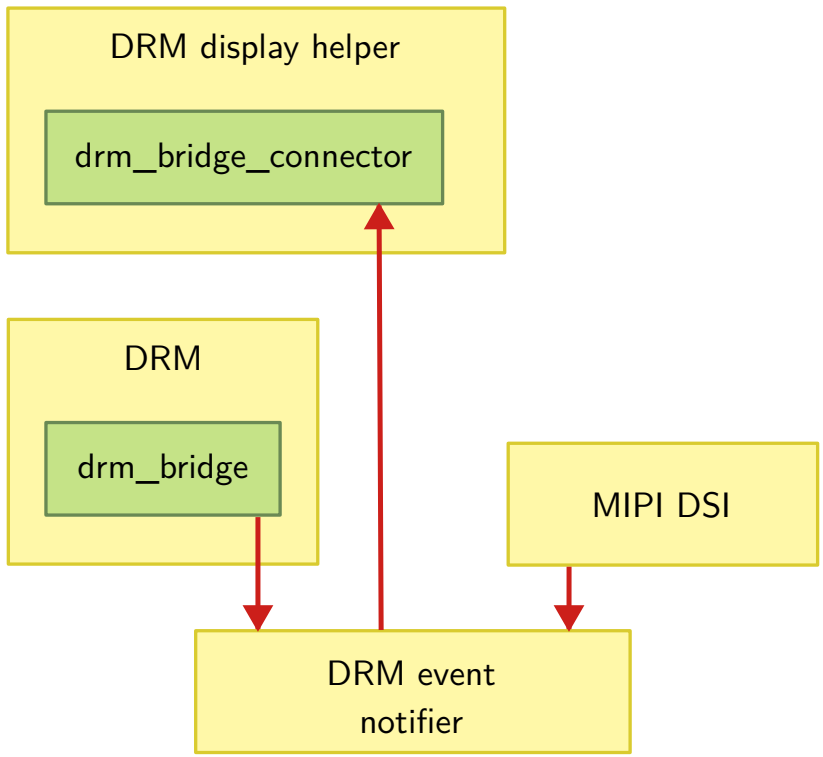
Incomplete pipelines: proposal



- Introduce *tail bridge* concept
  - *Last bridge* = the last bridge *currently* in the chain
  - *Tail bridge* 🆕 = a bridge not needing a next bridge (e.g. panel, hdmi-connector…)
  - `bool drm_bridge_is_tail(struct drm_bridge *bridge)` (`Patch 30`)
  - Based on the new `.is_tail` bridge func
- Bridge `.attach` func uses `-EPROBE_DEFER` to report "next bridge is missing"
  - Bridge drivers must return `-EPROBE_DEFER` when next bridge not (yet) available (`Patch 20`)
  - `drm_bridge_attach()` does not consider `-EPROBE_DEFER` a fatal error (`Patch 35`)

Event notification

- Need to react on hot(un)plug events
- Current events (for the hardware I'm on)
  - From different modules
  - `drm_bridge_remove()` / `drm_bridge_detach()`
  - `mipi_dsi_attach()`
- New: add `drm_event_notifier` module
  - Based on `struct notifier_block`
  - `Patch 26`
- Open points
  - For hotplug only? Other uses?
  - Event payload?



drm_bridge_connector: reacting to hotplug

- On a hotplug event (Patch 36)
- Propagate the attach call chain from the previously-last bridge

```
/* Propagate the attach call chain to newly hotplugged bridge(s) */
struct drm_bridge *last_bridge __free(drm_bridge_put) =
  drm_bridge_chain_get_last_bridge(bridge_connector->encoder);

/* Encoder chain empty? */
if (!last_bridge)
  return;

err = last_bridge->funcs->attach(last_bridge, bridge_connector->encoder,
                                 DRM_BRIDGE_ATTACH_NO_CONNECTOR);
```

- Add the connector

```
    if (drm_bridge_connector_pipeline_is_complete(bridge_connector))
        drm_bridge_connector_add_connector(bridge_connector);
```

drm_bridge_connector: reacting to hot-unplug

- On hot-unplug event
- Stop the pipeline and remove tail bridge(s) (Patch 25)

```
 void drm_bridge_remove(struct drm_bridge *bridge)
 {
+ if (bridge->encoder) {
+   drm_atomic_shutdown(bridge->dev);
+   drm_encoder_cleanup_from(bridge->encoder, bridge);
+ }
```

- It's a direct call, no notifier necessary here

Usage

Usage 1/2: adaptations needed on bridge drivers

On the last “fixed” bridge (Patch 20)

```
 static int foo_attach(...)
 {
 ...
+   if (!dsi->bridge.next_bridge)
+       return -EPROBE_DEFER;
 ...
 }
```

Usage 2/2: adaptations needed on encoder drivers

On the encoder: use the new `drm_bridge_connector` API (`Patch 37`)

```
-   struct drm_connector        *connector;
+   struct drm_bridge_connector *bridge_connector;
...
-   connector        = drm_bridge_connector_init(lcdif->drm, encoder);
+   bridge_connector = drmm_bridge_connector_init(lcdif->drm, encoder);
```

- Enabled the hotplg features
- Returns a `drm_bridge_connector` (drmm lifetime)
  - The `drm_connector` will be created/destroyed dynamically

Discussion!

Thank you!

Luca Ceresoli

luca.ceresoli@bootlin.com

https://bootlin.com/pub/conferences/