



Journey of a Massive Internal Kernel Rework

Luca Ceresoli

luca.ceresoli@bootlin.com

ELCE 2026

© Copyright 2004-2026, Bootlin.

Creative Commons BY-SA 3.0 license.

Corrections, suggestions, contributions and translations are welcome!



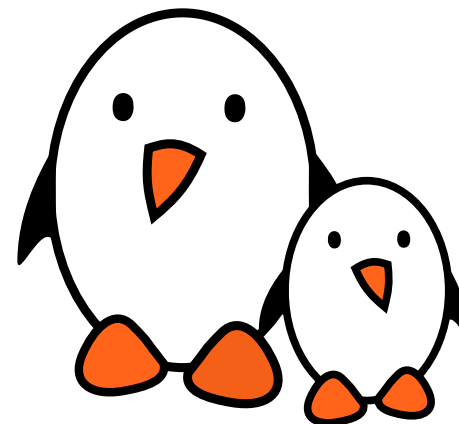


- ▶ Embedded Linux engineer at **Bootlin**
 - Embedded Linux **expertise**
 - Development, consulting and training
 - Strong Open-source focus
- ▶ **Linux kernel** device driver developer
- ▶ Bootloaders, Buildroot and Yocto integration
- ▶ Open-source contributor
- ▶ Living in **Bergamo**, Italy



Background

bootlin

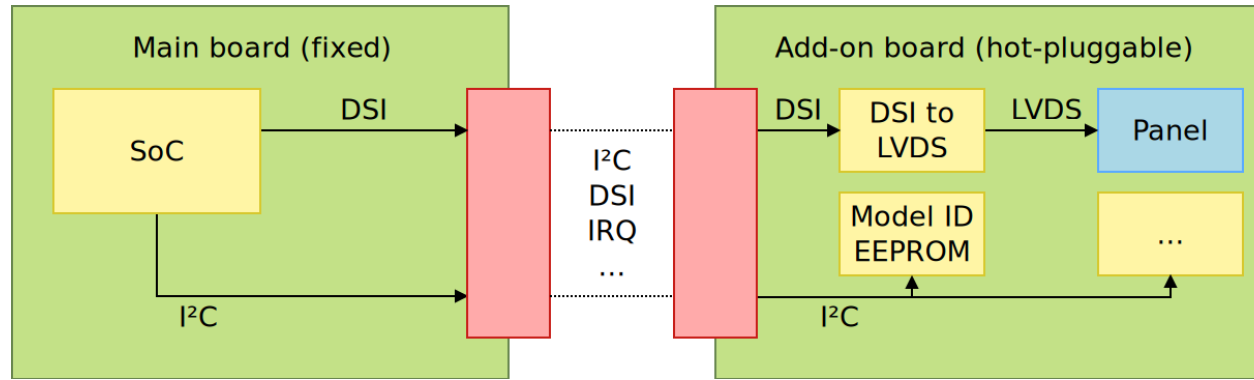




Use case: DRM meets bridge hotplug



Use case: embedded product with addons

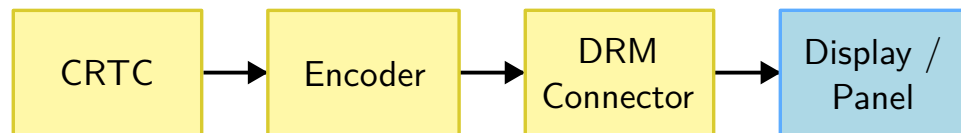


- ▶ A fairly normal ARM64 embedded system, using device tree...
- ▶ ...except it has a connector for an addon to extend features
 - Hotpluggable: can be inserted and removed by the user at any moment
 - With a MIPI-DSI video bus, converted to LVDS on the addon
- ▶ Well supported by DRM *without hotplug connector in the middle*

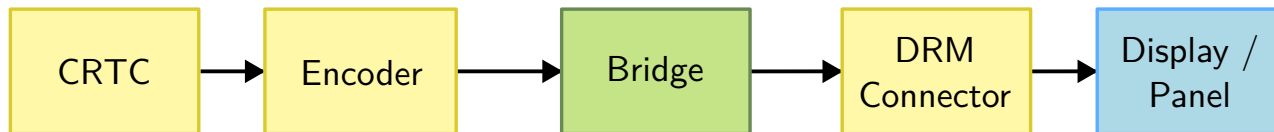


DRM 101

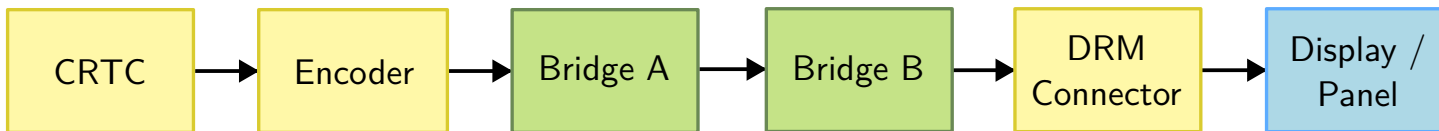
One card, multiple components



A simple DRM card



A DRM card with a bridge

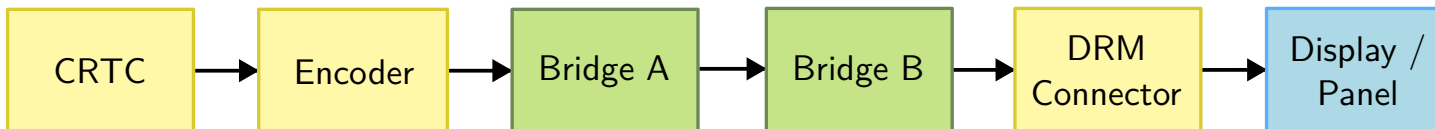


A DRM card with multiple bridges

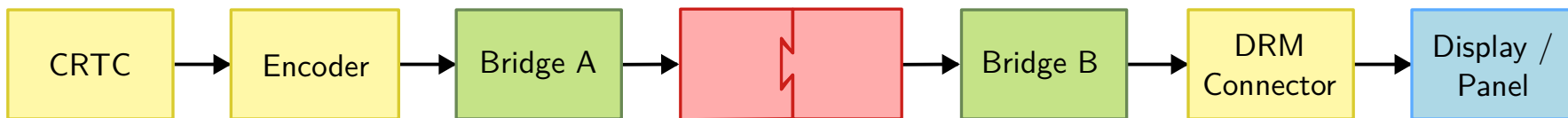


DRM + hotplug

- ▶ DRM supports hotplugging displays after a connector, not other components
- ▶ All the card components are assumed to be removed at once, on card teardown
- ▶ But the hotplug addon has a bridge!



A classic DRM card



A DRM card with hotpluggable bridges



Initial steps

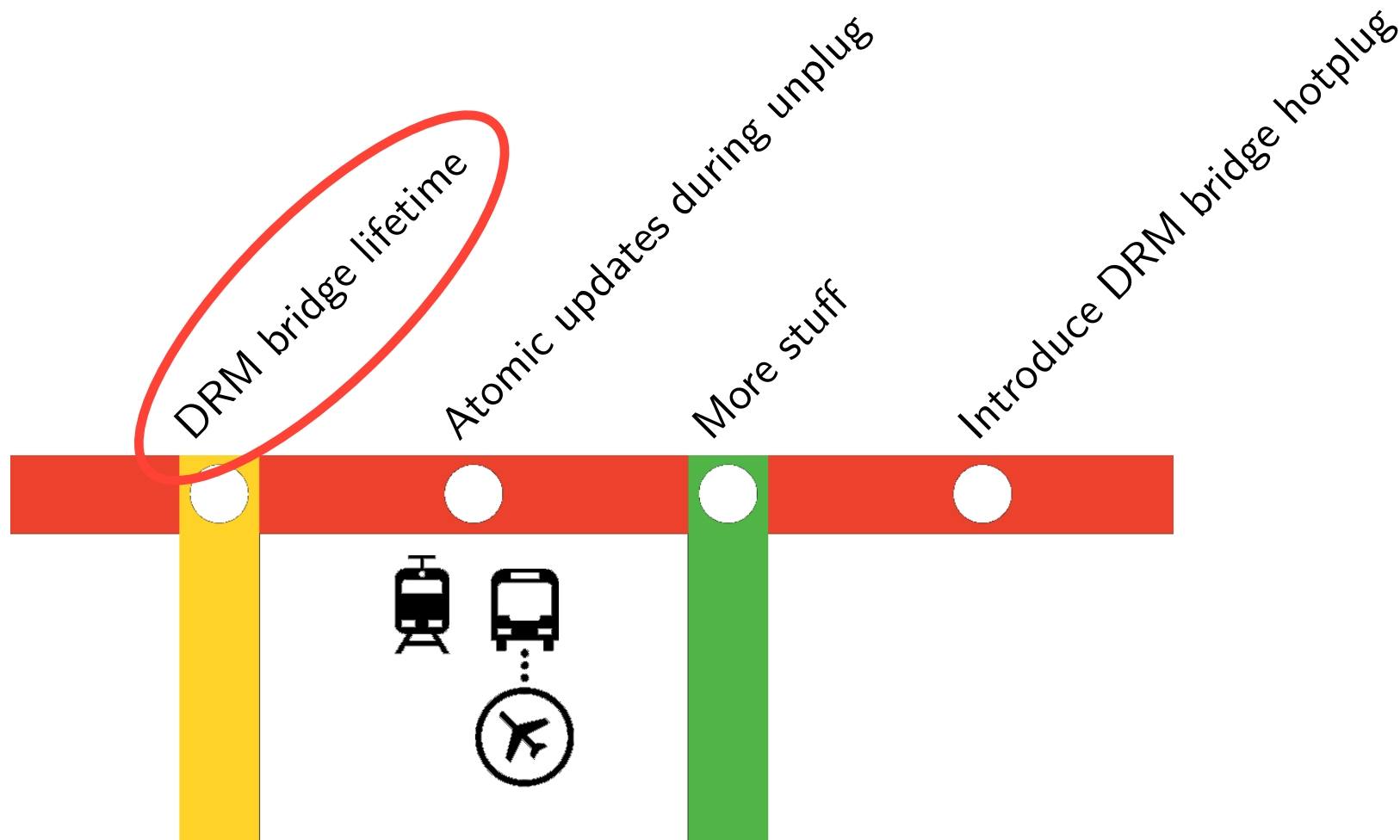


Initial attempts

- ▶ v1-v4 series with hotplug connector + DT overlay + DRM bridge hotplug
 - Trying to touch DRM core minimally
- ▶ v4 discussed at Linux Plumbers Conference 2024
 - Conclusion: need to rework DRM bridge lifetime significantly
 - Takeaway 1: don't be shy, don't assume the existing code is OK (especially if you have a new use case)
 - Led to sketching the *Grand Plan*TM



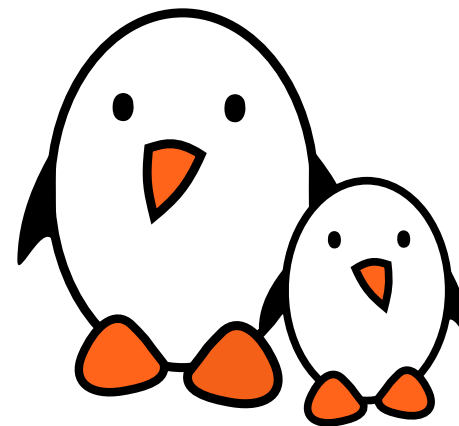
The Grand Plan





Dynamic bridge lifetime

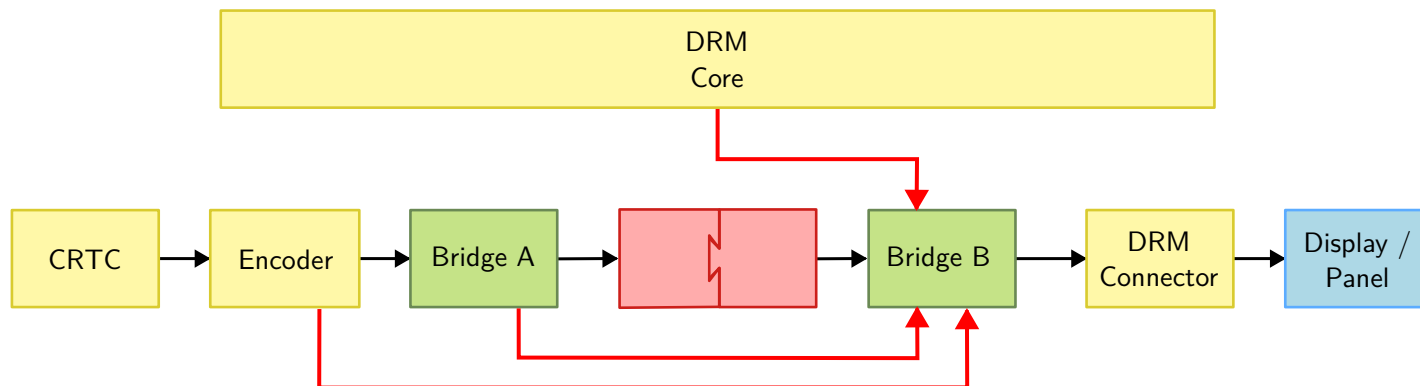
bootlin





The problem

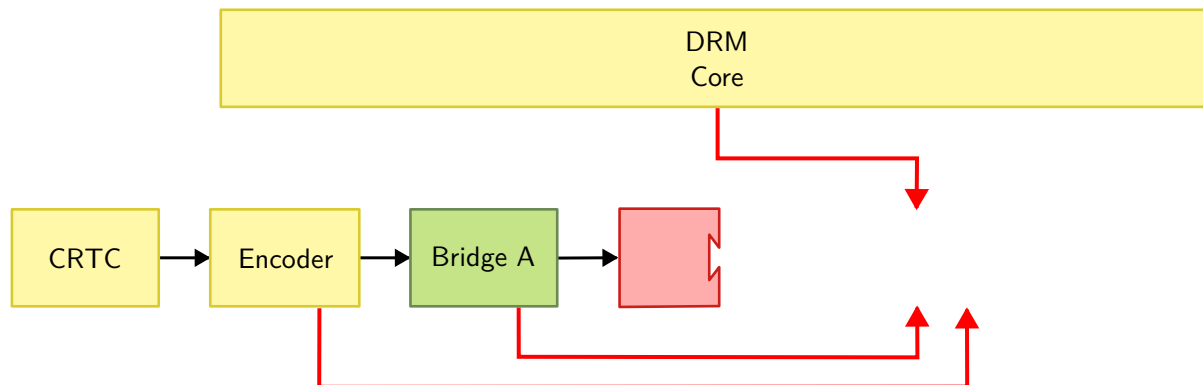
1. Several components take pointers to a bridge





The problem

2. When the bridge is removed, they become dangling pointers 💣

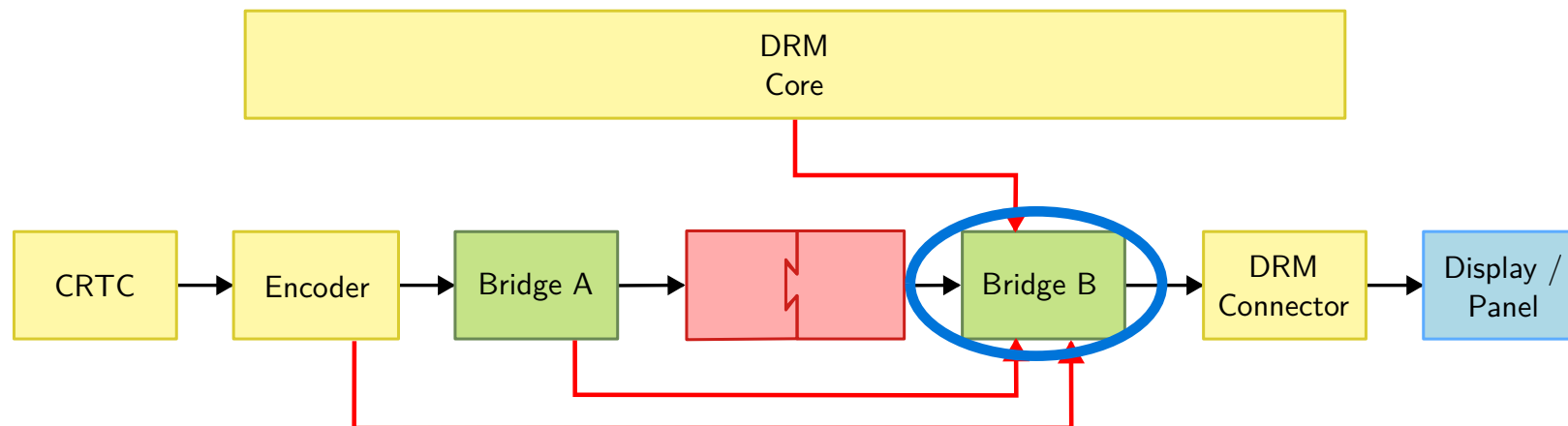




The provider side



The provider side: bridge drivers





Before: typical bridge driver pattern

```
struct foo {
    struct drm_bridge bridge;
    // driver-specific data
};

int foo_probe(struct platform_device *pdev)
{
    struct device *dev = &pdev->dev;
    struct foo *foo;

    foo = devm_kzalloc(dev, sizeof(*foo), GFP_KERNEL);
    foo->bridge.funcs = foo_bridge_funcs;
    // do something with foo
    drm_bridge_add(&foo->bridge); // Bridge pointer can be "found" by others
}
```

- ▶ Problem: foo is freed when dev is removed
 - The driver has no idea about who still has a pointer



Solution: bridge recounting + new allocation API

- ▶ recount added to struct `drm_bridge`
- ▶ New allocation API:

```
- foo = devm_kzalloc(dev, sizeof(*foo), GFP_KERNEL);  
+ foo = devm_drm_bridge_alloc(dev, struct foo, bridge, foo_bridge_funcs);
```

- ▶ `devm_drm_bridge_alloc`:
 - Allocates struct `foo`, which embeds a struct `drm_bridge`
 - Sets bridge recount to 1 (`dev` holds a reference)
 - Registers a `devm` action to put (= decrement) the recount
- ▶ When recount reaches 0 → `foo` is freed
- ▶ Added in v6.16



Converting bridge drivers to the new API

- ▶ All drivers had to be converted to the new API
 - Otherwise others cannot rely on refcounting
- ▶ Used a Coccinelle semantic patch to convert many in a single commit
 - `drm: convert many bridge drivers from devm_kzalloc() to devm_drm_bridge_alloc()`
API, v6.17
- ▶ Many drivers not caught by the semantic patch
 - Used `grep` to find them, converted manually
 - Not easy, there is no “old function” to find them easily



Robustness

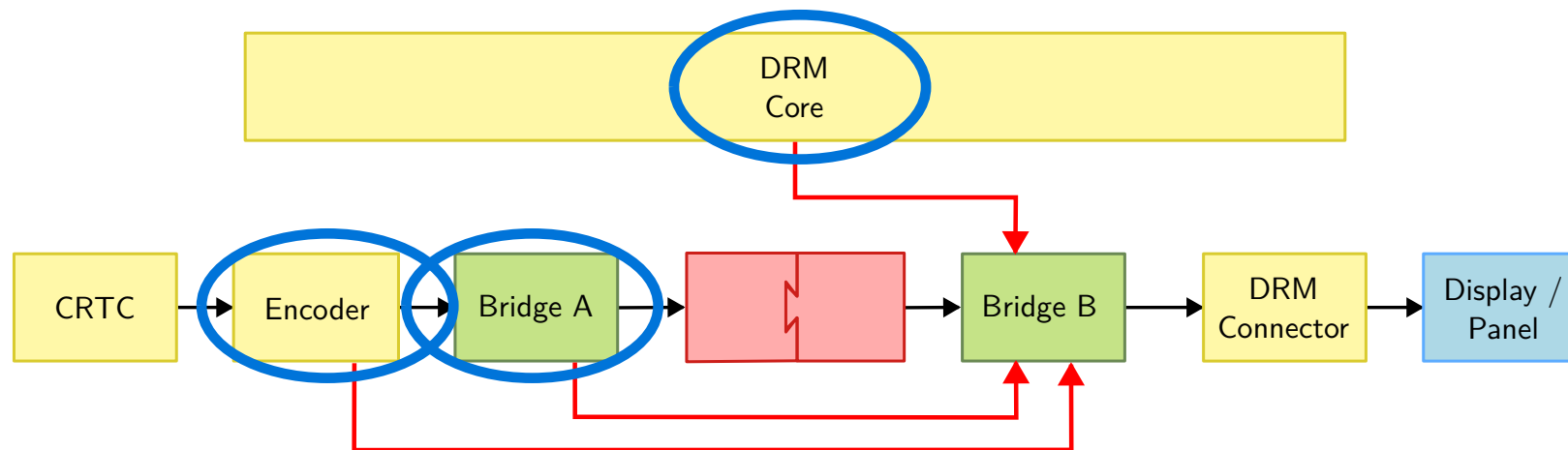
- ▶ Added warnings when the old pattern is used
 - Proved very useful!
 - Unveiled a few missed ones: reported by users of that hardware
 - Reacted quickly: sent a patch converting them, or reviewed their patches
- ▶ Added KUnit tests
 - To ensure the API keeps working as expected in the future
- ▶ Added debugfs improvements
 - Show bridge refcount and more
- ▶ Great ideas, mostly suggested by the community



The consumer side



The consumer side: bridge users





Before: typical bridge consumer pattern

1. Use **getters** to get a pointer to a bridge: `of_drm_find_bridge()`, `drm_bridge_chain_get_first_bridge()`, `drm_for_each_bridge_in_chain()`, etc
2. Often store it for later
 - ▶ Example:

```
struct bar {  
    struct drm_bridge *some_bridge;  
};  
  
int bar_something()  
{  
    bar->some_bridge = drm_bridge_chain_get_first_bridge(encoder);  
    // Often kept stored until bar removal  
    // Nobody keeps track of this reference  
}
```



Solution: implement refcounting in getters 1/2

- ▶ Increment refcount (=get a reference) in every getter when returning a bridge pointer

```
@@ -1337,6 +1337,9 @@ drm_bridge_get_prev_bridge(struct drm_bridge *bridge)
+ * The refcount of the returned bridge is incremented. Use drm_bridge_put()
+ * when done with it.
@@ -1344,8 +1347,8 @@ drm_bridge_get_prev_bridge(struct drm_bridge *bridge)
static inline struct drm_bridge *
drm_bridge_chain_get_first_bridge(struct drm_encoder *encoder)
{
-     return list_first_entry_or_null(&encoder->bridge_chain,
-                                     struct drm_bridge, chain_node);
+     return drm_bridge_get(list_first_entry_or_null(&encoder->bridge_chain,
+                                                     struct drm_bridge, chain_node));
}
```



Solution: implement refcounting in getters 2/2

- ▶ Update all callers of the getter to put the reference when done
 - Some easy: pointer used immediately and not stored

```
@@ -118,8 +118,9 @@ drm_mode_validate_pipeline(struct drm_display_mode *mode,  
    bridge = drm_bridge_chain_get_first_bridge(encoder);  
    *status = drm_bridge_chain_mode_valid(bridge,  
                                           &connector->display_info,  
                                           mode);  
+    drm_bridge_put(bridge);
```

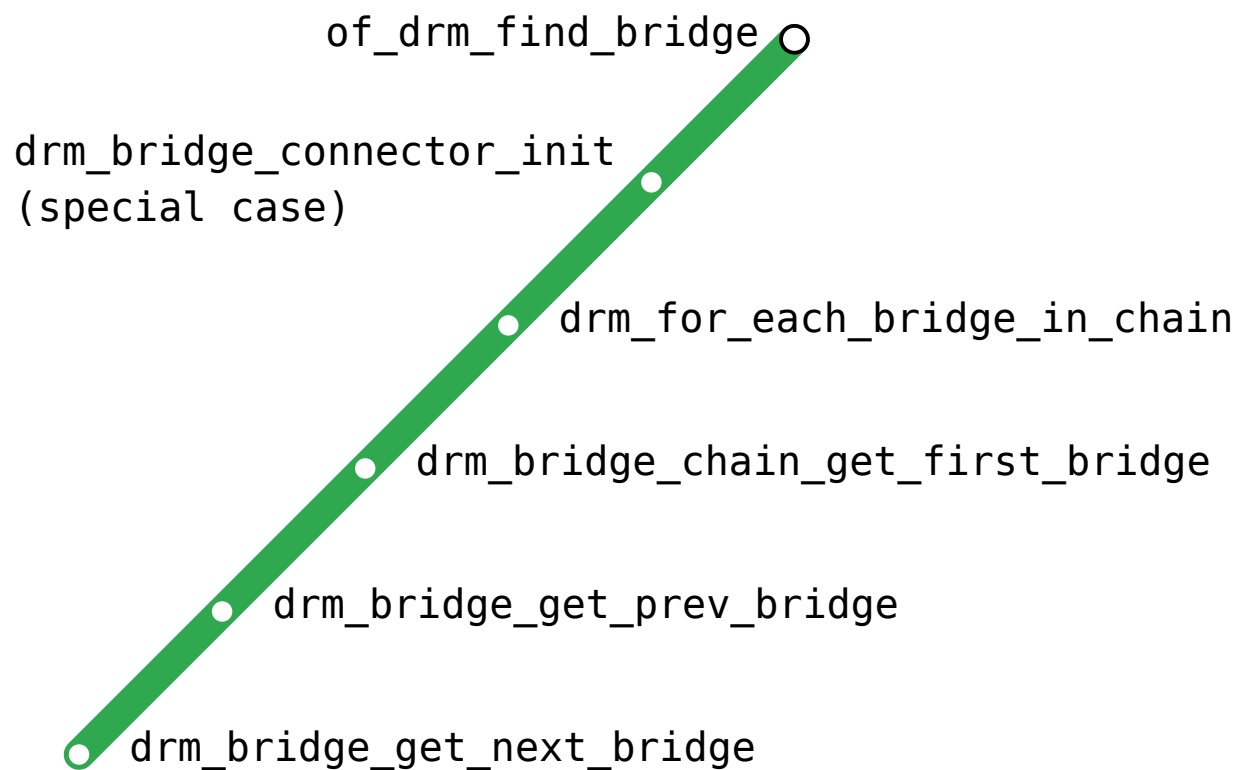
- ▶ Some hard: pointer stored, used in complex drivers with multiple files...



- ▶ Added some utilities to simplify putting the reference
- ▶ A **cleanup action** for local pointers : `__free(drm_bridge_put)` (v6.18)
 - **Example:**

```
struct drm_bridge *bridge __free(drm_bridge_put) =  
    drm_bridge_chain_get_first_bridge(encoder);
```

- ▶ A **next_bridge pointer** in `struct drm_bridge` (v7.0)
 - Automatically put when the bridge is freed
 - **Avoids use-after-free** which would be tricky to handle by each driver





Getters 1/3: simple, few users

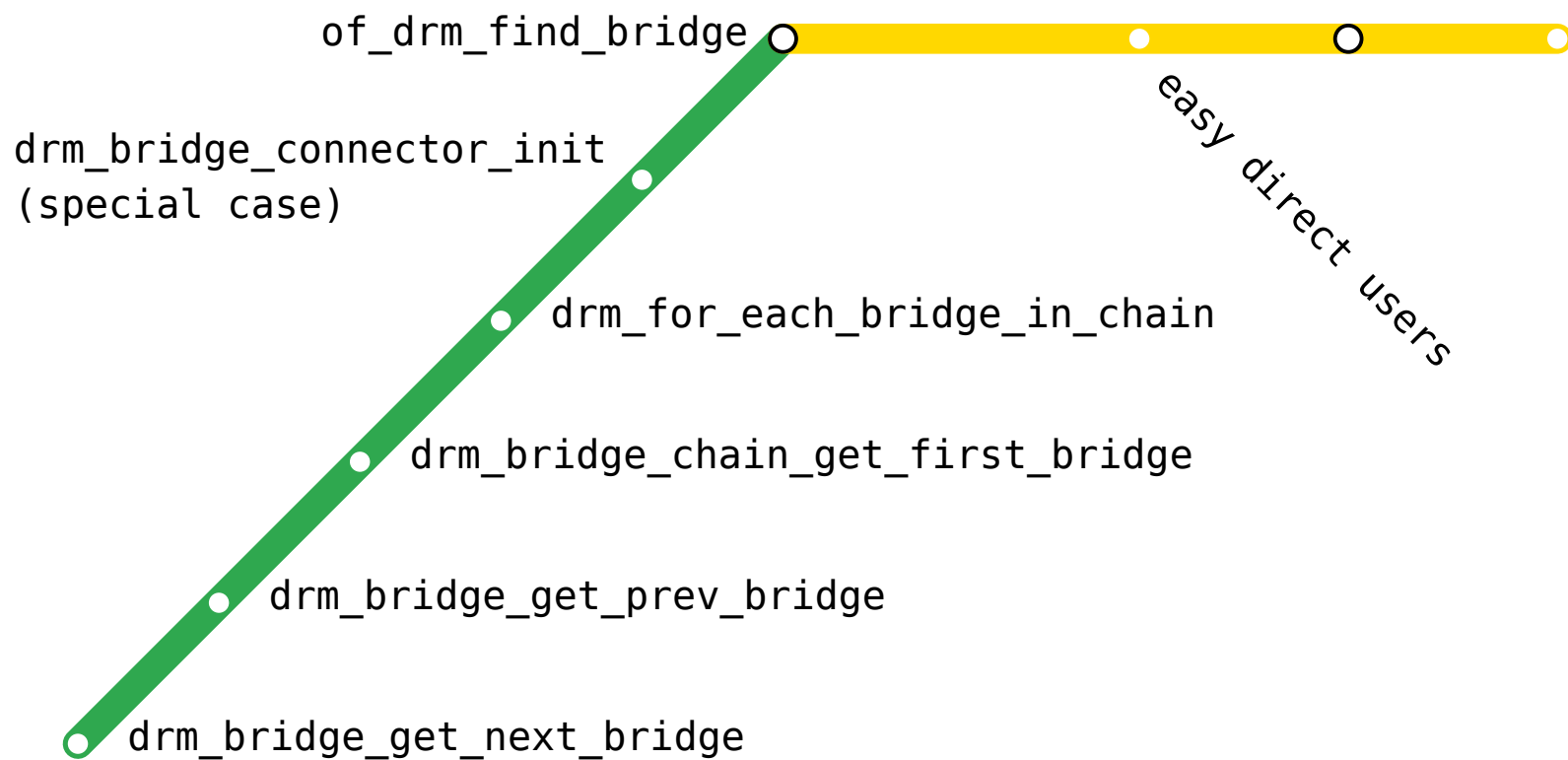
- ▶ Getters with few users and simple semantics: locate a given bridge, return its pointer

```
* drm_bridge_get_next_bridge: 6 users
* drm_bridge_get_prev_bridge: 3 users
* drm_bridge_chain_get_first_bridge: 7 users
* drm_for_each_bridge_in_chain: 4 users
```

- ▶ One patch series per getter:
 - One commit to `drm_bridge_get()` the returned bridge in the getter
 - One commit per user to add `drm_bridge_put()`
- ▶ Not bisectable
 - OK for DRM maintainers provided the series is applied at once
 - Other maintainers might disagree
 - Takeaway: understand the subsystem conventions



Getters, part II



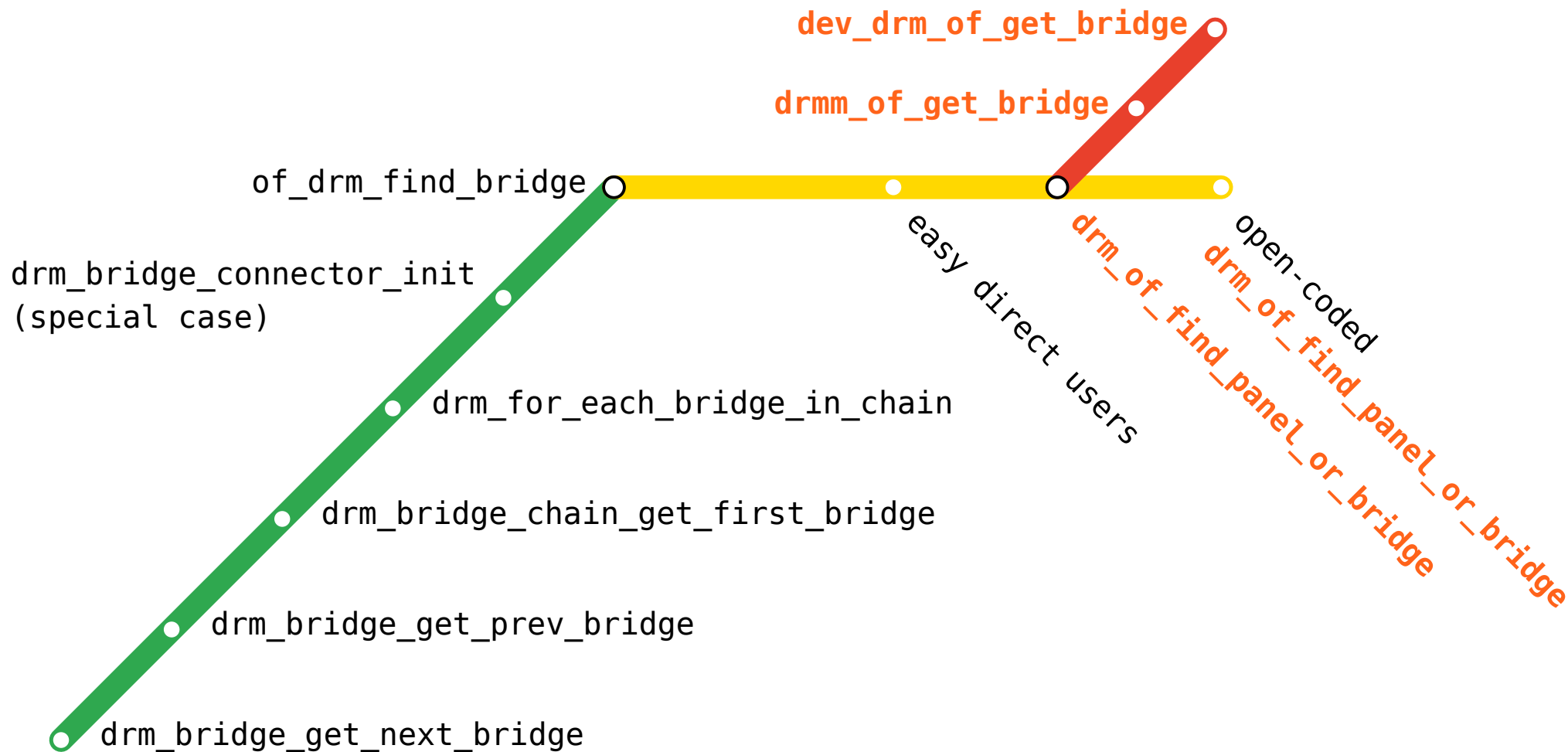


Getters, part II: `of_drm_find_bridge()` direct users

- ▶ Many users: about 100, some with extra complexity (see next slide)
- ▶ Converting all in a series was not realistic
- ▶ Add new replacement API: `of_drm_find_and_get_bridge()`
 - Same as `of_drm_find_bridge()` plus `drm_bridge_get()`
- ▶ Deprecate `of_drm_find_bridge()`
- ▶ Convert *direct* users one by one



Getters, part III





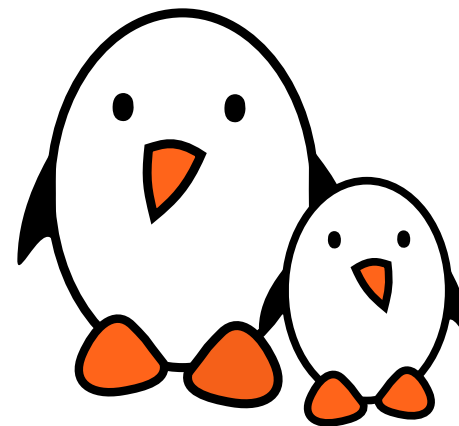
Getters III: `drm_of_find_panel_or_bridge()` and the panel bridge

- ▶ `drm_of_find_panel_or_bridge()` is very complex
 - It returns either a panel or a bridge, depending on what follows
- ▶ And it is used by `*_of_get_bridge()`
 - which wrap a panel in a bridge
 - inherently incompatible with bridge hotplug
- ▶ Got solved only recently:
 - `drm/panel`: embed a `drm_bridge` into every `drm_panel`, v7.4
 - Based on work by others to add refcounting to `struct drm_panel`, similarly to `struct drm_bridge`
 - Now all panels have a wrapping bridge
- ▶ Result: these getters are now deprecated and being removed



Conclusions

bootlin





Approach used

▶ Methodology:

- Study the problem space, estimate the *amount* and *complexity* of updates needed
- Sketch a plan, split it into smaller pieces, each doable as a self-contained series
- build-test everything, runtime-test what you can (you cannot have all the hardware)

▶ Tools:

- Your brain: understand the big picture
- `grep` and shell scripts to find places to convert
- Coccinelle semantic patches
- AI: not very useful when I started, but now:
 - useful for investigation and analysis of complex cases
 - use with care when editing code, ensure you understand the result
 - these changes are usually few-liners, ensure you understand the big picture first



Main takeaways

- ▶ Be open: discuss with the community, accept different ideas
- ▶ Be responsible
 - You'll break something
 - Don't worry, be happy (and reactive!): fix breakages as they get reported
- ▶ Don't be shy: don't assume the existing code is OK (especially if you have a new use case)
- ▶ Be humble: your proposal can be perfect for you but break other use cases, or some design pattern that still makes sense



Where we are now 1/2 (just in case you wonder)

Look at my **ridiculously broken-down cover letters**:

1. ... add refcounting to DRM bridges struct `drm_bridge`,
based on `devm_drm_bridge_alloc()`
 - A. ✓ add new alloc API and refcounting (v6.16)
 - B. ✓ convert all bridge drivers to new API (v6.17)
 - C. ✓ kunit tests (v6.17)
 - D. ✓ add get/put to `drm_bridge_add/remove()` + `attach/detach()`
and warn on old allocation pattern (v6.17)
 - E. ... add get/put on `drm_bridge` accessors
 1. ✓ `drm_bridge_chain_get_first_bridge()`, add cleanup action (v6.18)
 2. ✓ `drm_bridge_get_prev_bridge()` (v6.18)
 3. ✓ `drm_bridge_get_next_bridge()` (v6.19)
 4. ✓ `drm_for_each_bridge_in_chain()` (v6.19)
 5. ✓ `drm_bridge_connector_init` (v6.19)
 6. ✓ protect encoder bridge chain with a mutex (v7.2)
 7. ✓ `of_drm_find_bridge`
 - a. ✓ add `of_drm_get_bridge()` (v7.0),
convert basic direct users (v7.0-v7.1)
 - b. ✓ convert direct `of_drm_get_bridge()` users, part 2 (v7.0)
 - c. ✓ convert direct `of_drm_get_bridge()` users, part 3 (v7.0)
 - d. ✓ convert direct `of_drm_get_bridge()` users, part 4 (v7.1-v7.2)
 - e. ✓ bridge-only `drm_of_find_panel_or_bridge()` users (v7.2)

---->



Where we are now 2/2 (just in case you wonder)

<---

- 8. ... panel_bridge lifetime
 - a. ✓ cleanup DRM_PANEL in bridge drivers (v7.4)
 - b. ✓ embed a drm_bridge in every drm_panel (v7.4)
 - c. ... remove deprecated *_of_get_bridge(): non-OF drivers
 - d. ... remove deprecated *_of_get_bridge(): OF drivers
- 9. ✓ enforce drm_bridge_add before drm_bridge_attach (v6.19)
- F. ✓ debugfs improvements
 - 1. ✓ add top-level 'bridges' file (v6.16)
 - 2. ✓ show refcount and list lingering bridges (v6.19)
- 2. ✓ handle gracefully atomic updates during bridge removal
 - A. ✓ Add drm_bridge_enter/exit() to protect device resources (v7.0)
 - B. ✓ Add drm_bridge_clear_and_put() (v7.1)
- 3. ... DSI host-device driver interaction
- 4. ✓ removing the need for the "always-disconnected" connector
- 5. ✓ Migrate i.MX LCDIF driver to bridge-connector (v7.2)
- 6. ... DRM bridge hotplug
 - A. ... Bridge hotplug management in the DRM core
 - 1. ✓ bridge-connector: attach encoder to the connector (v7.2)
 - 2. ... drm bridge hotplug
 - B. Device tree description

Thank you!

Questions?

Luca Ceresoli

luca.ceresoli@bootlin.com

Slides under CC-BY-SA 3.0

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