Graphic Testing without hardware: discovering the power of VKMS!

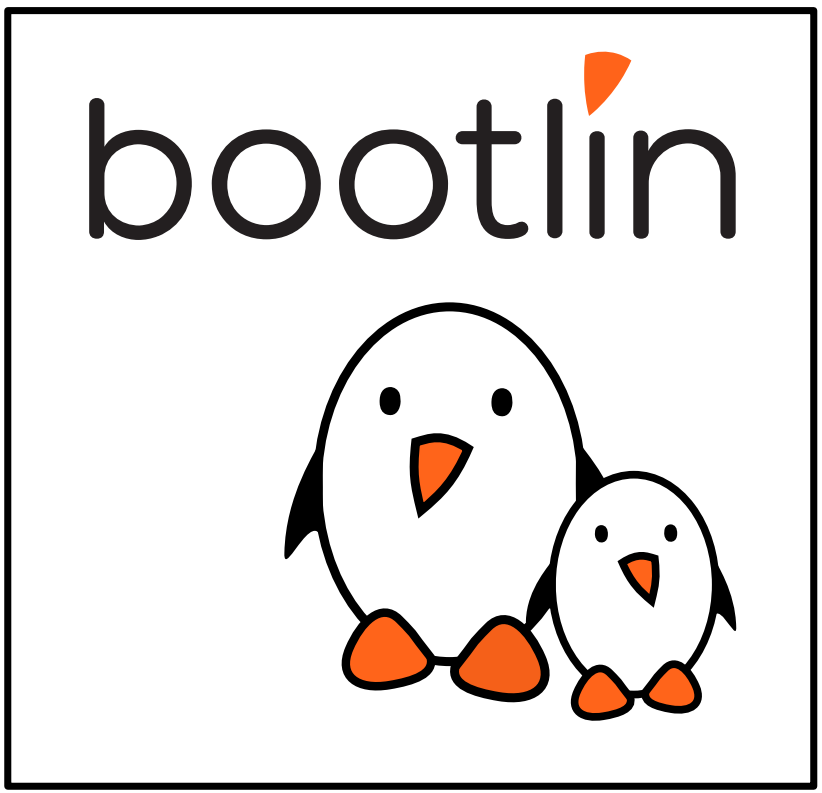
Louis Chauvet
louis.chauvet@bootlin.com

Louis Chauvet

- Embedded Linux engineer at Bootlin
  - Embedded Linux expertise
  - Development, consulting and training
  - Strong open-source focus
- VKMS Maintainer
- Living in **Toulouse**

Display testing

- What do we want to test?
- How to do it?



What do we want to test?

- Anything with a screen
- App which don't know exactly what is the hardware it will be ran on

How to do it?

Without VKMS	With VKMS

How to do it?

Without VKMS	With VKMS
1. Buy boards 2. Setup board 3. Debug 4. Go to step 1.	

How to do it?

Without VKMS	With VKMS
1. Buy boards 2. Setup board 3. Debug 4. Go to step 1.	1. Run VM 2. Set VKMS config 3. Debug 4. Go to step 1.

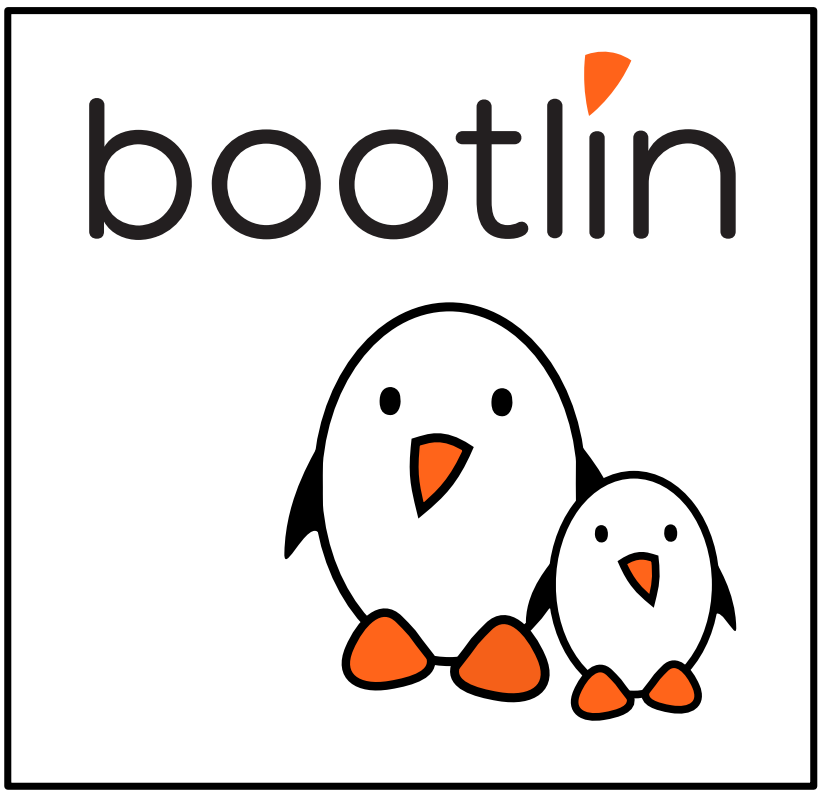
How to do it?

Without VKMS	With VKMS
1. Buy boards 2. Setup board 3. Debug 4. Go to step 1.	1. Run VM 2. Set VKMS config 3. Debug 4. Go to step 1.
Pros: Real hardware **Cons**: Expensive, physical manipulations, hard to integrate in CI	

How to do it?

Without VKMS	With VKMS
1. Buy boards 2. Setup board 3. Debug 4. Go to step 1.	1. Run VM 2. Set VKMS config 3. Debug 4. Go to step 1.
Pros: Real hardware **Cons**: Expensive, physical manipulations, hard to integrate in CI	**Pros**: Free, easy to automate **Cons**: Not the real hardware

VKMS, what it is?



VKMS, what it is?

VKMS: **V**irtual **K**ernel **M**ode **S**etting

> VKMS is a software-only model of a KMS driver that is useful for testing and for running X (or similar) on headless machines. VKMS aims to enable a virtual display with no need of a hardware display capability, releasing the GPU in DRM API tests.
>
> — Documentation/gpu/vkms.rst

VKMS, what it is?

Virtual - No hardware required, can be used in virtual machine

VKMS, what it is?

Virtual - No hardware required, can be used in virtual machine

Kernel Modesetting - Real kernel interface, no hack on your application

VKMS, what it is?

Virtual - No hardware required, can be used in virtual machine

Kernel Modesetting - Real kernel interface, no hack on your application

Main goal: Testing userspace without hardware

Quick overview of display pipeline

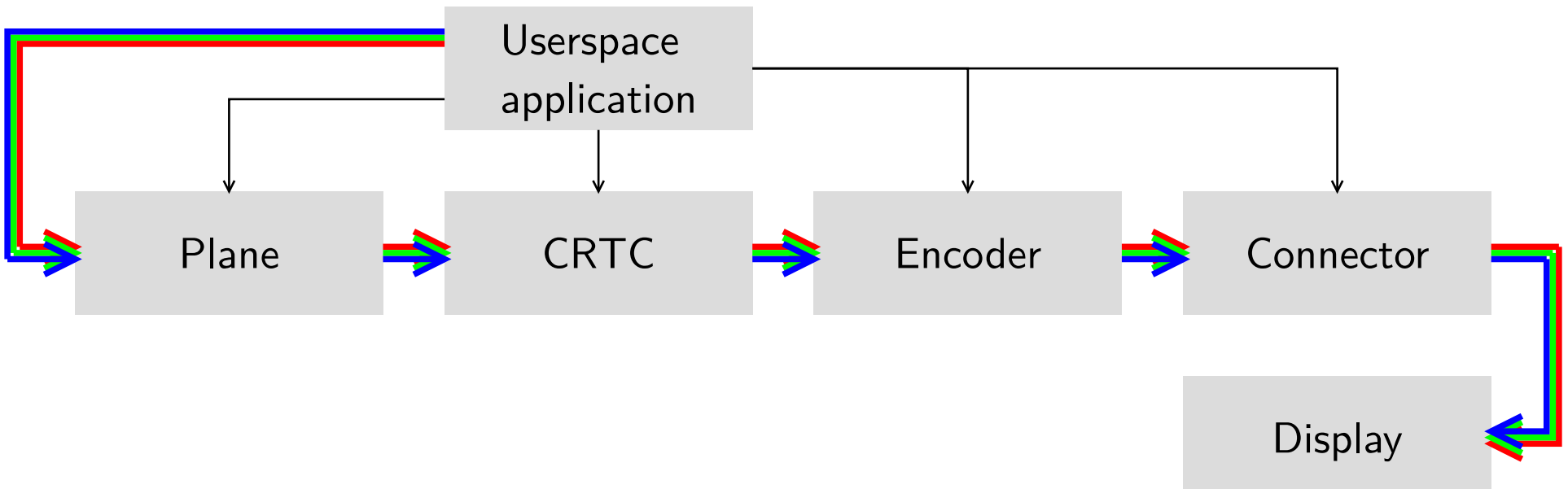
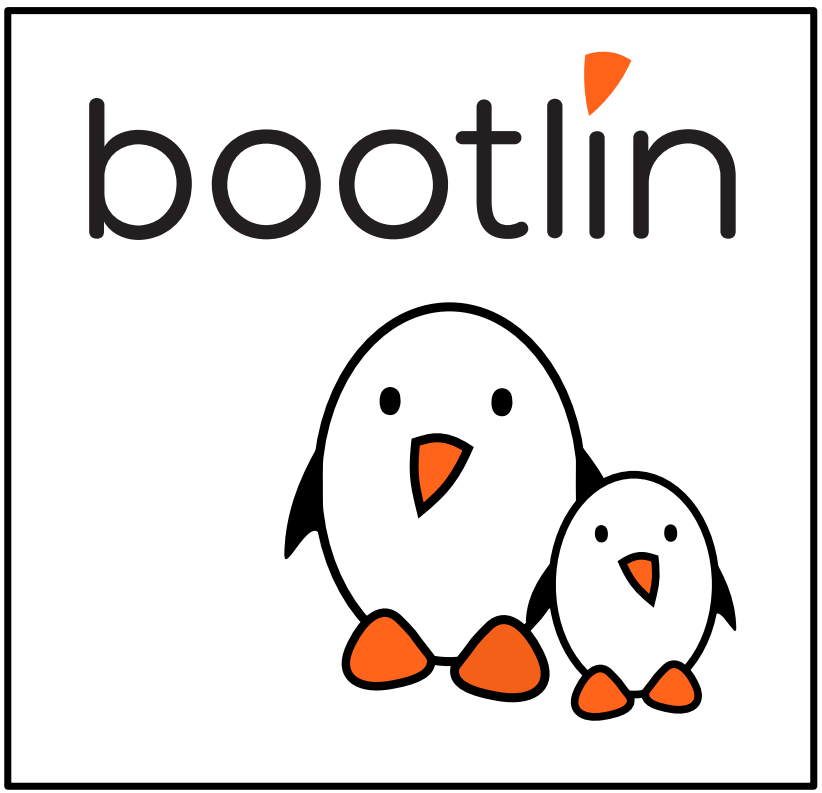


Figure 1: Overview of a display pipeline

Current VKMS Capabilities

- Plane composition
- Color conversion
- Writeback
- Kernel arguments
- Minimal configuration
- Maximal configuration



Plane composition

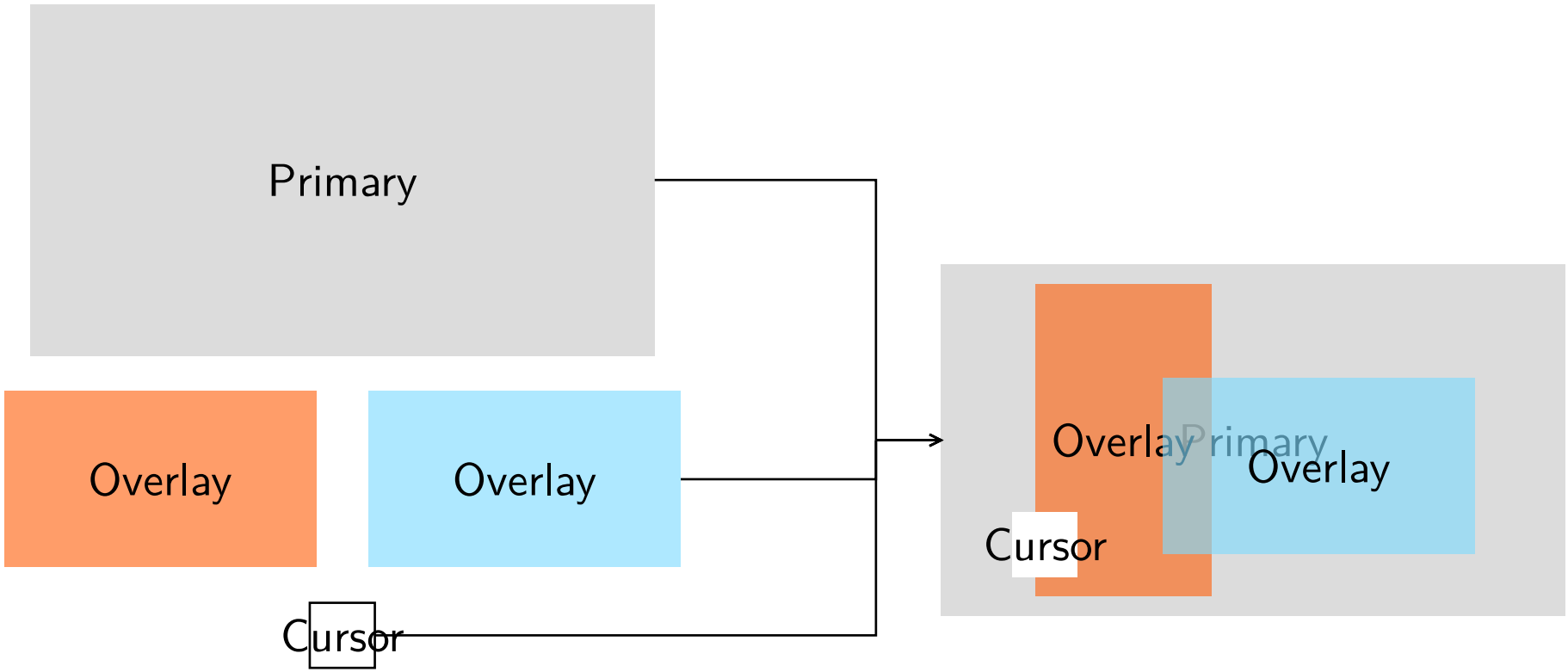


Figure 2: Plane composition

Color conversion

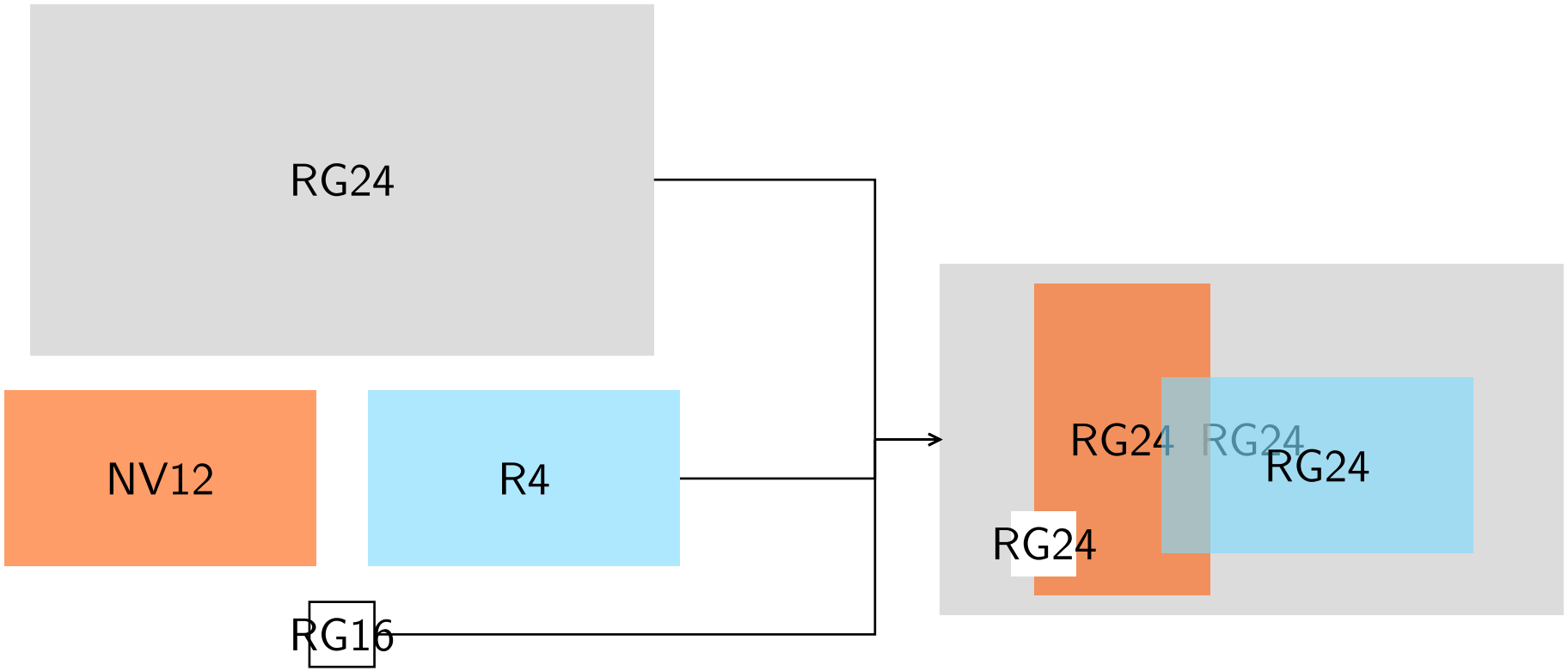


Figure 3: Plane composition

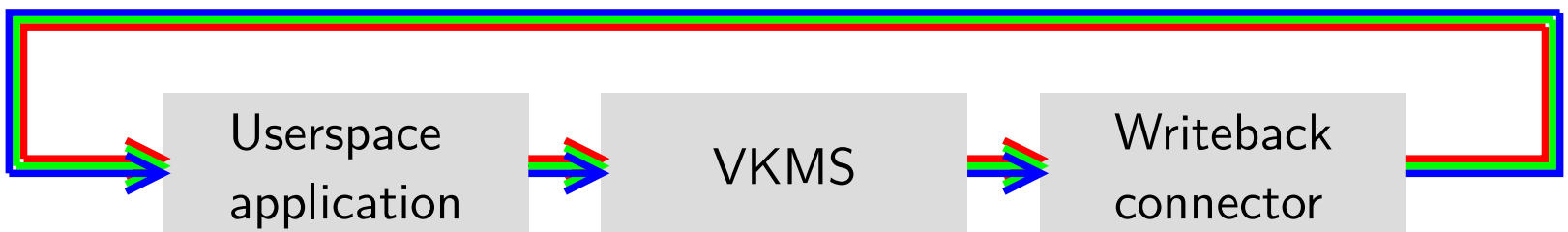


Figure 4: Writeback connector

Kernel arguments

- `enable_cursor` - `true`

  Add a cursor plane
- `enable_writeback` - `true`

  Add a writeback connector
- `enable_overlay` - `false`

  Add multiple planes

Minimal configuration

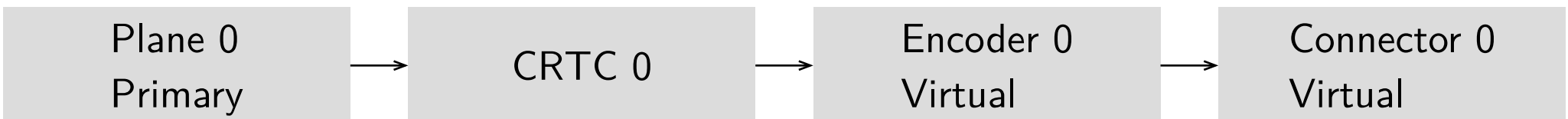


Figure 5: `enable_cursor=false`, `enable_writeback=false`, `enable_overlay=false`

Maximal configuration

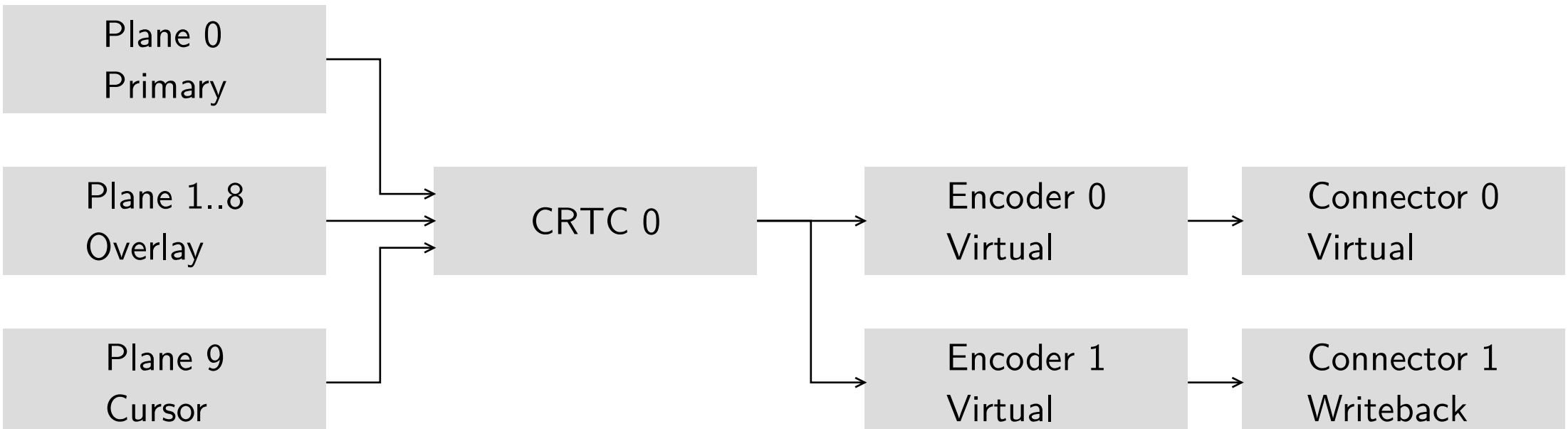
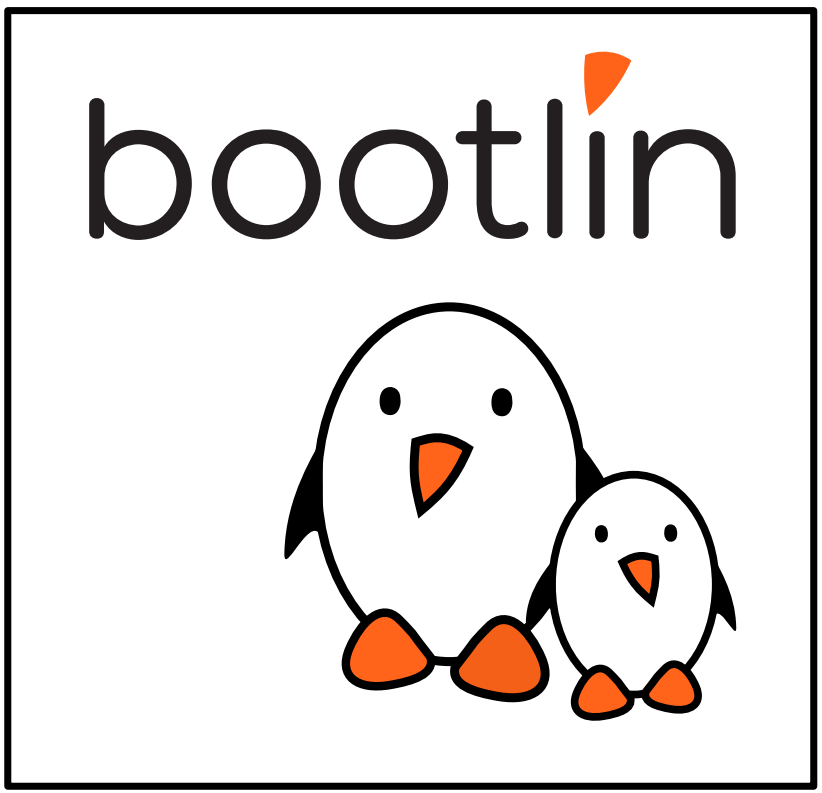


Figure 6: `enable_cursor=true`, `enable_writeback=true`, `enable_overlay=true`

Real life examples

- Texas Instrument - AM6232
- Framework 13 - intel



Real life examples - AM6232

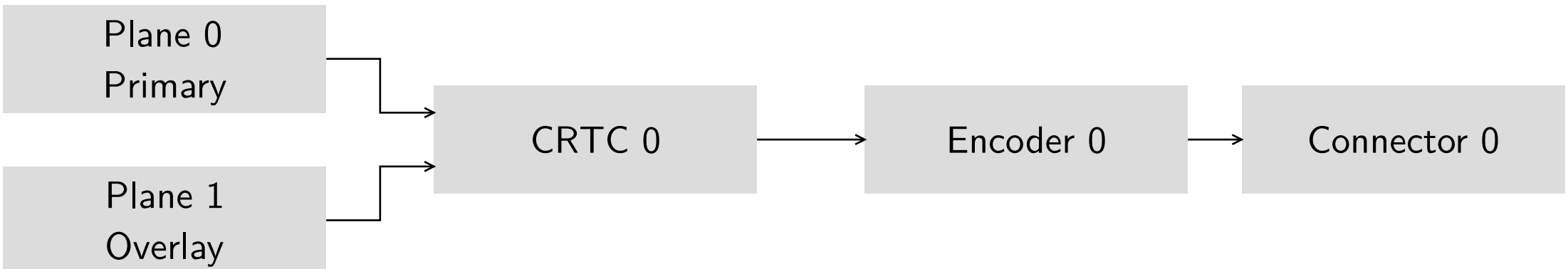


Figure 7: AM6232

Real life examples - Framework 13 - intel

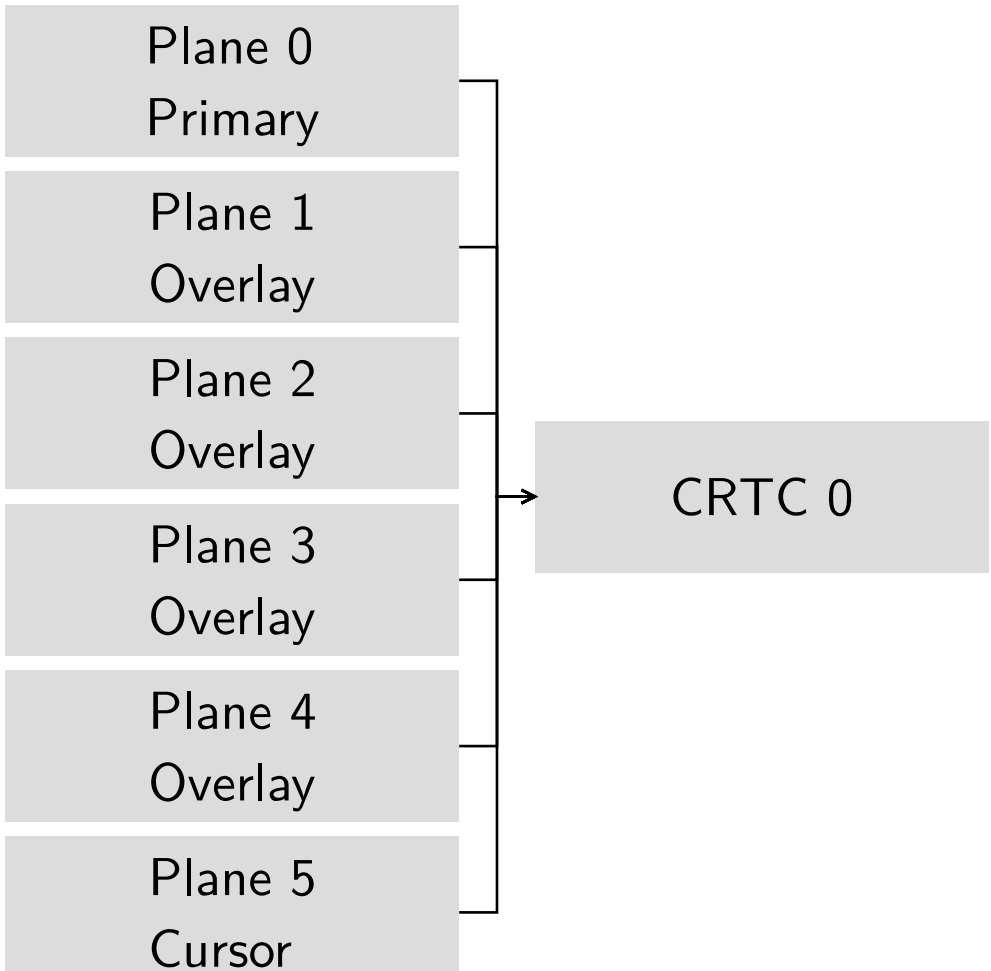


Figure 8: Framework 13

Real life examples - Framework 13 - intel

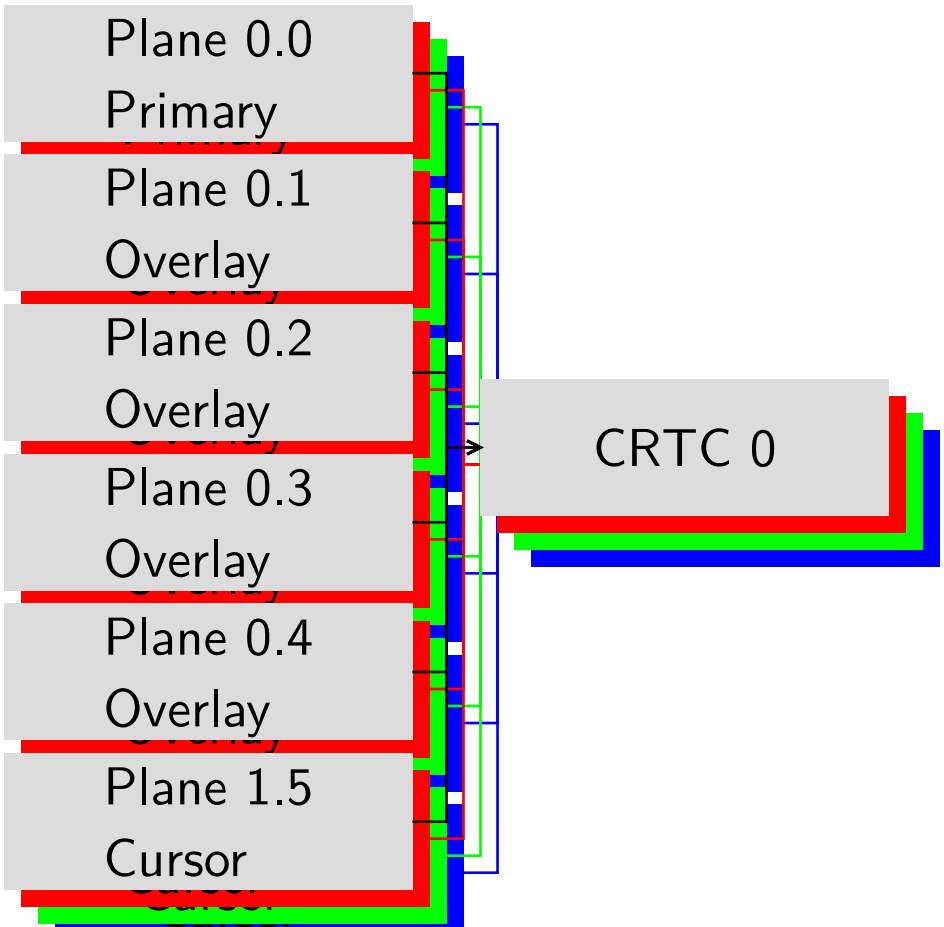


Figure 9: Framework 13

Real life examples - Framework 13 - intel

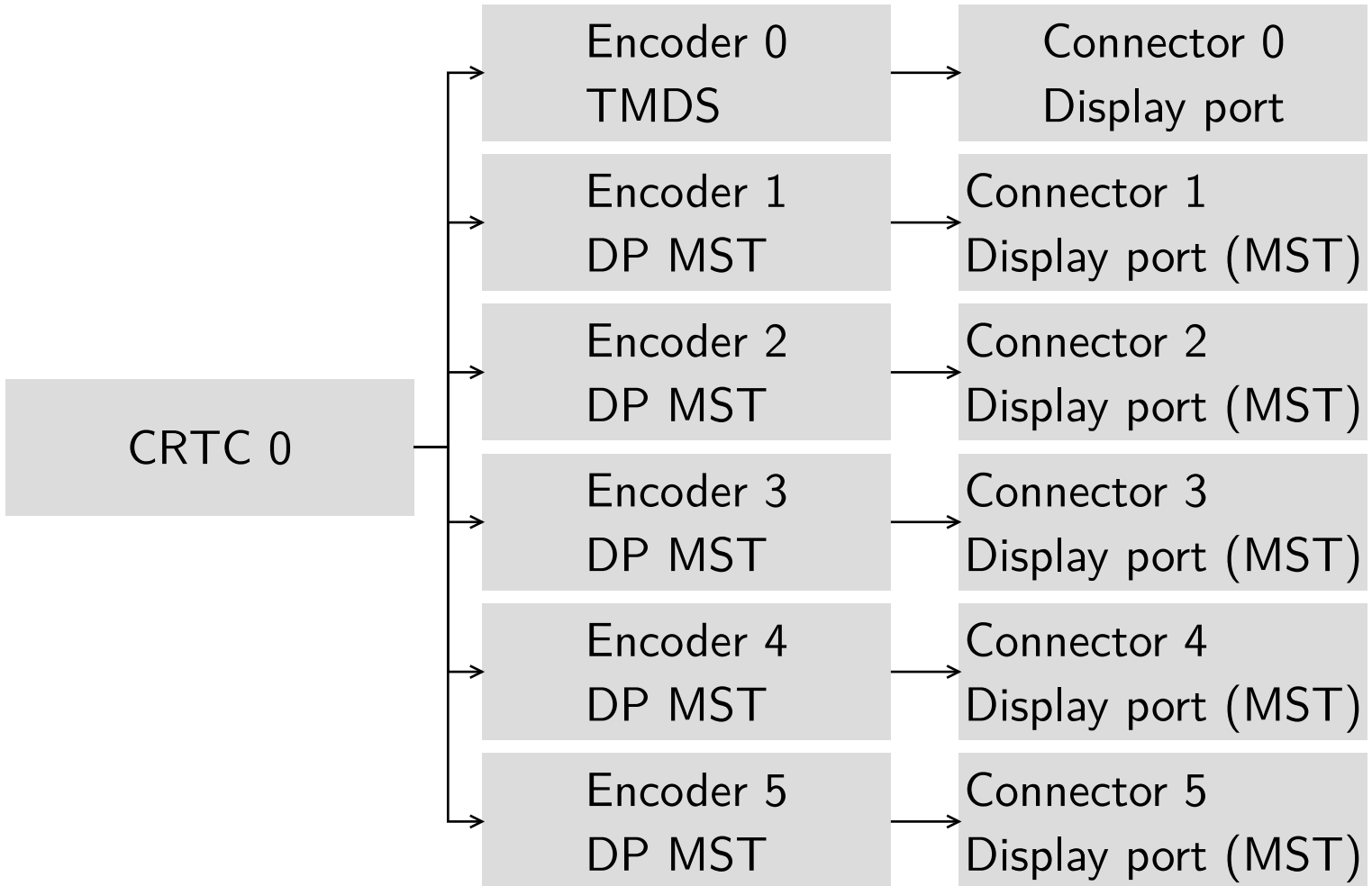


Figure 10: Framework 13

Real life examples - Framework 13 - intel

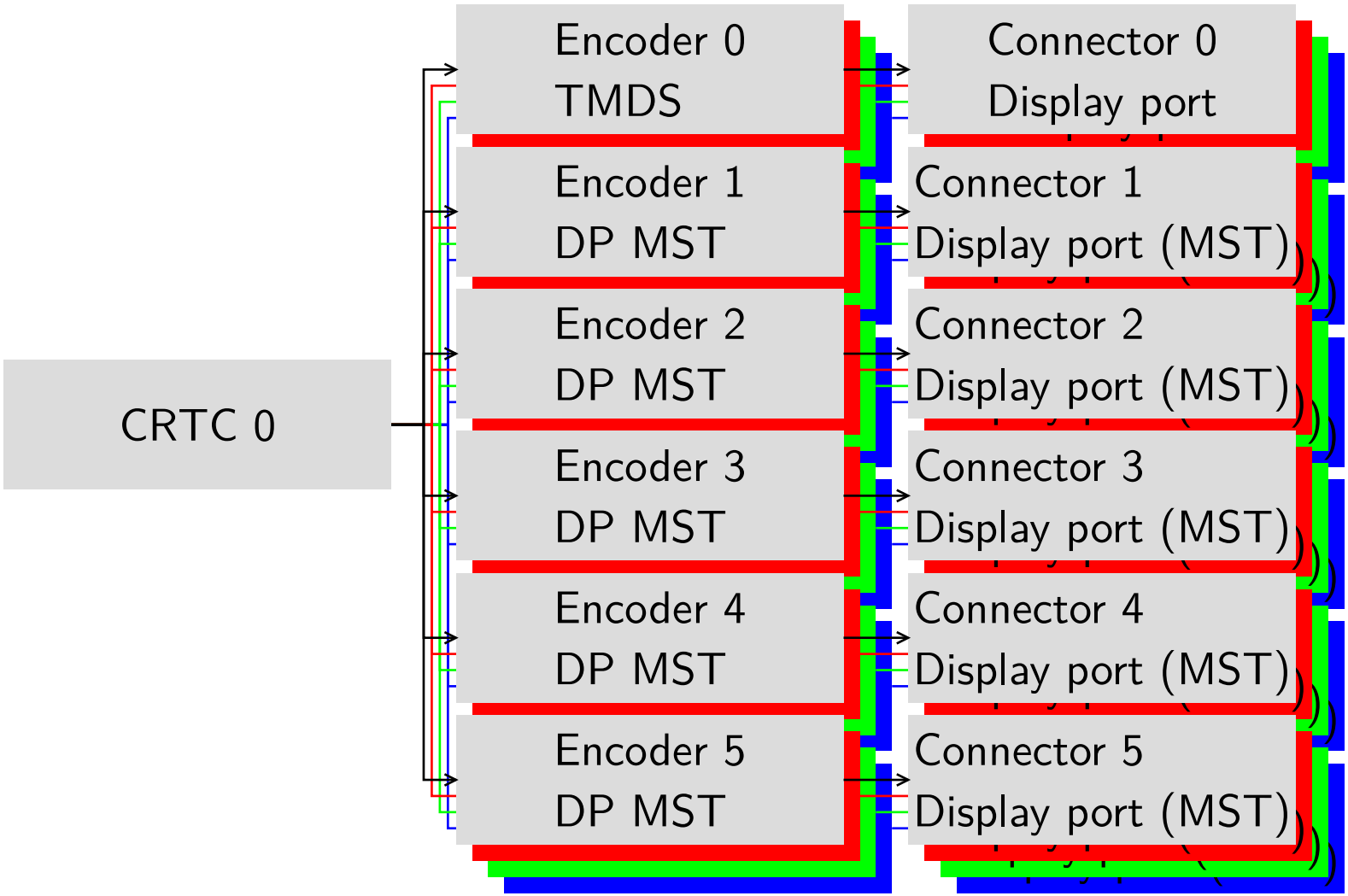


Figure 11: Framework 13

ConfigFS interface

- Upstreaming status

Upstreaming status

- Internal VKMS modifications done

Upstreaming status

- Internal VKMS modifications done
- First set of configurations proposed by José Exposito and me

Upstreaming status

- Internal VKMS modifications done
- First set of configurations proposed by José Exposito and me

Last iteration: 20250507135431.53907-1-jose.exposito89@gmail.com

Repository: github.com/Fomys/linux/tree/review/20250507135431.53907-1-jose.exposito89@gmail.com

Demo time - probing

- New kernel argument: `create_default_dev`, disable or enable the creation of the default device to ensure backwards compatibility

```
$ modprobe vkms create_default_dev=0
```

Demo time - Creating a device

- Creating a device is as simple as creating a folder in ConfigFS:

```
$ mkdir /sys/kernel/config/vkms/MY_DEV
```

Demo time - Creating a device

- Creating a device is as simple as creating a folder in ConfigFS:

```
$ mkdir /sys/kernel/config/vkms/MY_DEV
```

```
$ tree /sys/kernel/config/vkms/MY_DEV
/sys/kernel/config/vkms/DEV_1/
├── connectors/
├── crtcs/
├── enabled
├── encoders/
└── planes/
```

Demo time - Enabling a device

- Enabling a device is done using the enabled file:

Demo time - Enabling a device

- Enabling a device is done using the enabled file:

```
$ echo 1 > /sys/kernel/config/vkms/MY_DEV/enabled
```

Demo time - Enabling a device

- Enabling a device is done using the enabled file:

```
$ echo 1 > /sys/kernel/config/vkms/MY_DEV/enabled
(NULL device *): [drm] The number of planes must be between 1 and 31
```

But currently we did not configure it yet, DRM is complaining about missing planes

Demo time - Creating a plane

- Again, creating a plane only requires to create a folder:

```
$ mkdir /sys/kernel/config/vkms/MY_DEV/planes/PLA_1
```

Demo time - Creating a plane

- Again, creating a plane only requires to create a folder:

```
$ mkdir /sys/kernel/config/vkms/MY_DEV/planes/PLA_1

$ tree /sys/kernel/config/vkms/MY_DEV/planes
/sys/kernel/config/vkms/MY_DEV/planes
└── PLA_1
    ├── possible_crtcs
    └── type
```

Demo time - Creating a plane

- Again, creating a plane only requires to create a folder:

```
$ mkdir /sys/kernel/config/vkms/MY_DEV/planes/PLA_1
```

```
$ tree /sys/kernel/config/vkms/MY_DEV/planes
/sys/kernel/config/vkms/MY_DEV/planes
└── PLA_1
    ├── possible_crtcs
    └── type
```

- **possible_crtcs** - Folder containing the connected CRTC
- **type** - Plane type configuration: 0 = Overlay, 1 = Primary, 2 = Cursor

Demo time - Creating a plane

- Again, creating a plane only requires to create a folder:

```
$ mkdir /sys/kernel/config/vkms/MY_DEV/planes/PLA_1
```

```
$ tree /sys/kernel/config/vkms/MY_DEV/planes
/sys/kernel/config/vkms/MY_DEV/planes
└── PLA_1
    ├── possible_crtcs
    └── type
```

- **possible_crtcs** - Folder containing the connected CRTC
- **type** - Plane type configuration: 0 = Overlay, 1 = Primary, 2 = Cursor

We want it to be a primary plane:

```
$ echo 1 > /sys/kernel/config/vkms/MY_DEV/planes/PLA_1/type
```

Demo time - Creating a plane

- Again, creating a plane only requires to create a folder:

```
$ mkdir /sys/kernel/config/vkms/MY_DEV/planes/PLA_1
```

```
$ tree /sys/kernel/config/vkms/MY_DEV/planes
/sys/kernel/config/vkms/MY_DEV/planes
└── PLA_1
    ├── possible_crtcs
    └── type
```

- **possible_crtcs** - Folder containing the connected CRTC
- **type** - Plane type configuration: 0 = Overlay, 1 = Primary, 2 = Cursor

We want it to be a primary plane:

```
$ echo 1 > /sys/kernel/config/vkms/MY_DEV/planes/PLA_1/type
```

```
$ echo 1 > /sys/kernel/config/vkms/MY_DEV/enabled
(NULL device *): [drm] The number of CRTCs must be between 1 and 31
```

Demo time - Creating a CRTC

- And again, just create a folder

```
$ mkdir /sys/kernel/config/vkms/MY_DEV/crtcs/CRTC_1
```

Demo time - Creating a CRTC

- And again, just create a folder

```
$ mkdir /sys/kernel/config/vkms/MY_DEV/crtcs/CRTC_1
```

```
$ tree /sys/kernel/config/vkms/MY_DEV/crtcs
/sys/kernel/config/vkms/MY_DEV/crtcs
└── CRTC_1
    └── writeback
        └── writeback
```

Demo time - Creating a CRTC

- And again, just create a folder

```
$ mkdir /sys/kernel/config/vkms/MY_DEV/crtcs/CRTC_1
```

```
$ tree /sys/kernel/config/vkms/MY_DEV/crtcs
/sys/kernel/config/vkms/MY_DEV/crtcs
└── CRTC_1
    └── writeback
        └── writeback
```

- **writeback** - create a writeback connector for this CRTC

Demo time - Creating a CRTC

- And again, just create a folder

```
$ mkdir /sys/kernel/config/vkms/MY_DEV/crtcs/CRTC_1
```

```
$ tree /sys/kernel/config/vkms/MY_DEV/crtcs
/sys/kernel/config/vkms/MY_DEV/crtcs
└── CRTC_1
    └── writeback
        └── writeback
```

- **writeback** - create a writeback connector for this CRTC

```
$ echo 1 > /sys/kernel/config/vkms/MY_DEV/enabled
(NULL device *): [drm] The number of encoders must be between 1 and 31
```

This time we need an encoder

Demo time - Creating an encoder

- You guessed it, just a new folder!

```
$ mkdir /sys/kernel/config/vkms/MY_DEV/encoders/ENC_1
```

```
$ tree /sys/kernel/config/vkms/MY_DEV/encoders
/sys/kernel/config/vkms/MY_DEV/encoders
└── ENC_1
    └── possible_crtcs
```

Demo time - Creating an encoder

- You guessed it, just a new folder!

```
$ mkdir /sys/kernel/config/vkms/MY_DEV/encoders/ENC_1
```

```
$ tree /sys/kernel/config/vkms/MY_DEV/encoders
/sys/kernel/config/vkms/MY_DEV/encoders
└── ENC_1
    └── possible_crtcs
```

- **possible_crtcs** - Folder containing the connected CRTC

Demo time - Creating an encoder

- You guessed it, just a new folder!

```
$ mkdir /sys/kernel/config/vkms/MY_DEV/encoders/ENC_1
```

```
$ tree /sys/kernel/config/vkms/MY_DEV/encoders
/sys/kernel/config/vkms/MY_DEV/encoders
└── ENC_1
    └── possible_crtcs
```

- **possible_crtcs** - Folder containing the connected CRTC

```
$ echo 1 > /sys/kernel/config/vkms/MY_DEV/enabled
(NULL device *): [drm] The number of connectors must be between 1 and 31
```

And last, a connector!

Demo time - Creating a connector

The last folder to create!

```
$ mkdir /sys/kernel/config/vkms/MY_DEV/connectors/CON_1
```

Demo time - Creating a connector

The last folder to create!

```
$ mkdir /sys/kernel/config/vkms/MY_DEV/connectors/CON_1
```

```
$ tree /sys/kernel/config/vkms/MY_DEV/connectors
/sys/kernel/config/vkms/MY_DEV/connectors
└── CON_1
    ├── possible_encoders
    └── status
```

Demo time - Creating a connector

The last folder to create!

```
$ mkdir /sys/kernel/config/vkms/MY_DEV/connectors/CON_1
```

```
$ tree /sys/kernel/config/vkms/MY_DEV/connectors
/sys/kernel/config/vkms/MY_DEV/connectors
└── CON_1
    ├── possible_encoders
    └── status
```

- **possible_encoders** - Folder containing the connected encoders
- **status** - Set the connection status of the connector: 1 = connected, 2 = disconnected, 2 = unknown

Demo time - Creating a connector

The last folder to create!

```
$ mkdir /sys/kernel/config/vkms/MY_DEV/connectors/CON_1
```

```
$ tree /sys/kernel/config/vkms/MY_DEV/connectors
/sys/kernel/config/vkms/MY_DEV/connectors
└── CON_1
    ├── possible_encoders
    └── status
```

- **possible_encoders** - Folder containing the connected encoders
- **status** - Set the connection status of the connector: 1 = connected, 2 = disconnected, 2 = unknown

```
$ echo 1 > /sys/kernel/config/vkms/MY_DEV/enabled
(NULL device *): [drm] All planes must have at least one possible CRTC
```

It is time to wire all those items together

Demo time - connect everything

Item in the graphic pipeline are connected together using symlinks. In our case:

```
$ ln -s /sys/kernel/config/vkms/MY_DEV/crtcs/CRTC_1/ /sys/kernel/config/vkms/
MY_DEV/planes/PLA_1/possible_crtcs/
$ ln -s /sys/kernel/config/vkms/MY_DEV/crtcs/CRTC_1/ /sys/kernel/config/vkms/
MY_DEV/encoders/ENC_1/possible_crtcs/
$ ln -s /sys/kernel/config/vkms/MY_DEV/encoders/ENC_1/ /sys/kernel/config/vkms/
MY_DEV/connectors/CON_1/possible_encoders/
```

Demo time - connect everything

Item in the graphic pipeline are connected together using symlinks. In our case:

```
$ ln -s /sys/kernel/config/vkms/MY_DEV/crtcs/CRTC_1/ /sys/kernel/config/vkms/
MY_DEV/planes/PLA_1/possible_crtcs/
$ ln -s /sys/kernel/config/vkms/MY_DEV/crtcs/CRTC_1/ /sys/kernel/config/vkms/
MY_DEV/encoders/ENC_1/possible_crtcs/
$ ln -s /sys/kernel/config/vkms/MY_DEV/encoders/ENC_1/ /sys/kernel/config/vkms/
MY_DEV/connectors/CON_1/possible_encoders/
```

```
$ echo 1 > /sys/kernel/config/vkms/MY_DEV/enabled
```

Demo time - Some experimentations

- `$ modetest -M vkms`

  `[...]`

Demo time - Some experimentations

- `$ modetest -M vkms`
  `[...]`
- `$ drm_info /dev/dri/card0`
  `[...]`

Demo time - Some experimentations

- `$ modetest -M vkms`
  `[...]`
- `$ drm_info /dev/dri/card0`
  `[...]`
- `$ modetest -M vkms -s 40:#0`

```
opened device `Virtual Kernel Mode Setting` on driver `vkms` (version
1.0.0 at 0)
setting mode 1024x768-60.00Hz on connectors 40, crtc 38
```

Demo time - Some experimentations

- `$ modetest -M vkms`
  `[...]`
- `$ drm_info /dev/dri/card0`
  `[...]`
- `$ modetest -M vkms -s 40:#0`
  `opened device `Virtual Kernel Mode Setting` on driver `vkms` (version 1.0.0 at 0)`
  `setting mode 1024x768-60.00Hz on connectors 40, crtc 38`
- Disconnection of the connector:

```
$ echo 2 > /sys/kernel/config/vkms/MY_DEV/connectors/CON_1/status
$ modetest -M vkms -s 40:#0
opened device `Virtual Kernel Mode Setting` on driver `vkms` (version 1.0.0 at 0)
failed to find mode "#0" for connector 40
failed to create dumb buffer: Invalid argument
```

Pending patches

A lot of work is ready to go

- New color formats for planes and writeback
- More configuration for planes, connector, encoder

Repository: https://github.com/Fomys/linux/tree/configfs-everything

Demo time 2

For readers:

```
$ modprobe vkms create_default_dev=0
$ mkdir /sys/kernel/config/vkms/MY_DEV
$ mkdir /sys/kernel/config/vkms/MY_DEV/planes/PLA_1
$ mkdir /sys/kernel/config/vkms/MY_DEV/planes/PLA_2
$ mkdir /sys/kernel/config/vkms/MY_DEV/crtcs/CRTC_1
$ mkdir /sys/kernel/config/vkms/MY_DEV/encoders/ENC_1
$ mkdir /sys/kernel/config/vkms/MY_DEV/connectors/CON_1
$ mkdir /sys/kernel/config/vkms/MY_DEV/connectors/CON_2
```

Demo time 2 - continuation


```
$ ln -s /sys/kernel/config/vkms/MY_DEV/crtcs/CRTC_1/ /sys/kernel/config/vkms/
MY_DEV/planes/PLA_1/possible_crtcs/
$ ln -s /sys/kernel/config/vkms/MY_DEV/crtcs/CRTC_1/ /sys/kernel/config/vkms/
MY_DEV/planes/PLA_2/possible_crtcs/
$ ln -s /sys/kernel/config/vkms/MY_DEV/crtcs/CRTC_1/ /sys/kernel/config/vkms/
MY_DEV/encoders/ENC_1/possible_crtcs/
$ ln -s /sys/kernel/config/vkms/MY_DEV/encoders/ENC_1/ /sys/kernel/config/vkms/
MY_DEV/connectors/CON_1/possible_encoders/
$ ln -s /sys/kernel/config/vkms/MY_DEV/encoders/ENC_1/ /sys/kernel/config/vkms/
MY_DEV/connectors/CON_2/possible_encoders/
$ echo 1 > /sys/kernel/config/vkms/MY_DEV/planes/PLA_1/type
$ echo 0 > /sys/kernel/config/vkms/MY_DEV/planes/PLA_2/type
```

Demo time 2 - plane configuration


```
$ tree /sys/kernel/config/vkms/MY_DEV/planes/
/sys/kernel/config/vkms/MY_DEV/planes/
└── PLA_1
    ├── default_color_encoding      # default for color encoding
    ├── default_color_range         # default for color range
    ├── default_rotation            # default for rotation
    ├── possible_crtcs
    │   └── CRTC_1 -> ../../../../../vkms/MY_DEV/crtcs/CRTC_1
    ├── supported_color_encoding    # color encoding bitmask
    ├── supported_color_range       # color range bitmask
    ├── supported_formats           # +RG24 to enable RGB888, -RG24 to disable
    ├── supported_rotations         # rotation bitmask
    └── type                        # 0 = Overlay, 1 = Primary, 2 = Cursor
```

Demo time 2 - plane color format configuration

- Only support RGB888 for primary plane

```
$ echo '-*' > /sys/kernel/config/vkms/MY_DEV/planes/PLA_1/supported_formats
$ echo '+RG24' > /sys/kernel/config/vkms/MY_DEV/planes/PLA_1/supported_formats
```

- Support everything except RGB888 for overlay plane

```
$ echo '+*' > /sys/kernel/config/vkms/MY_DEV/planes/PLA_2/supported_formats
$ echo '-RG24' > /sys/kernel/config/vkms/MY_DEV/planes/PLA_2/supported_formats
```

Demo time 2 - encoder configuration


```
$ tree /sys/kernel/config/vkms/MY_DEV/encoders/
/sys/kernel/config/vkms/MY_DEV/encoders/
└── ENC_1
    ├── possible_crtcs
    │   └── CRTC_1 -> ../../../../../vkms/MY_DEV/crtcs/CRTC_1
    └── type            # 5 = virtual, 3 = LVDS, ...
```

Demo time 2 - connector configuration


```
$ tree /sys/kernel/config/vkms/MY_DEV/connectors/
/sys/kernel/config/vkms/MY_DEV/connectors/
└── CON_1
    ├── dynamic                 # 1 = dynamic connector (MST), 0 = static
    ├── edid                    # raw edid content
    ├── edid_enabled            # 1 = use the provided edid, 0 = use default
mode list
    ├── enabled                 # if dynamic, create/destroy this connector on
the fly
    ├── possible_encoders
    │   └── ENC_1 -> ../../../../../vkms/MY_DEV/encoders/ENC_1
    ├── status                  # 1 = connected, 2 = disconnected, 2 = unknown
    ├── supported_colorspace    # for HDR support (only for type=eDP, DP or
HDMI)
    └── type                    # 15 = virtual, 10 = DP, ...
```

Demo time 2 - dynamic connector


```
$ echo 1 > /sys/kernel/config/vkms/MY_DEV/connectors/CON_2/dynamic
$ echo 0 > /sys/kernel/config/vkms/MY_DEV/connectors/CON_2/enabled
$ echo 1 > /sys/kernel/config/vkms/MY_DEV/enabled
$ modetest -M vkms | grep connected
45  0 connected Virtual-1       0x0   34  44
$ echo 1 > /sys/kernel/config/vkms/MY_DEV/connectors/CON_2/enabled
$ modetest -M vkms | grep connected
45  0 connected Virtual-1       0x0   34  44
46  0 connected Virtual-2       0x0   34  44
```

Demo time 2 - Plane formats


```
$ modetest -M vkms -p
[...]
```

Visual demonstrations

Questions? Suggestions? Comments?

Louis Chauvet
louis.chauvet@bootlin.com

https://bootlin.com/pub/conferences/2025/elce/chauvet-graphic-testing