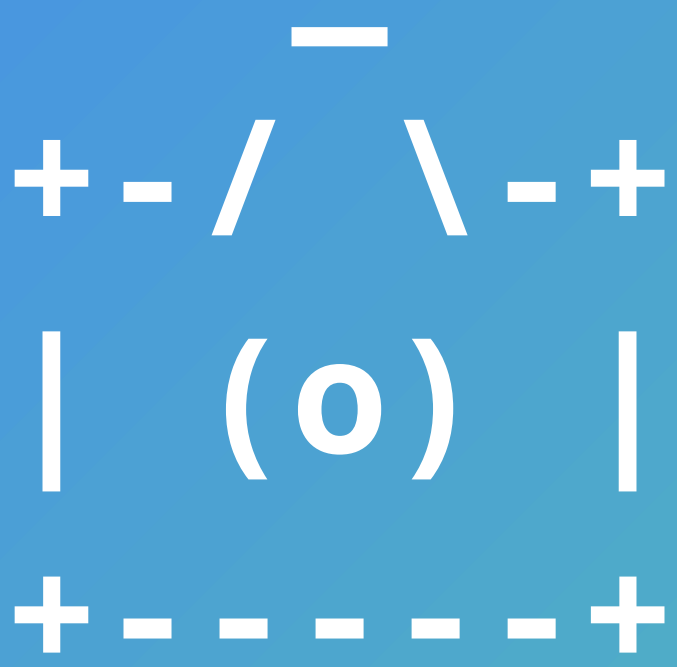


Multi-context support in V4L2

Media-summit 2025
Nice

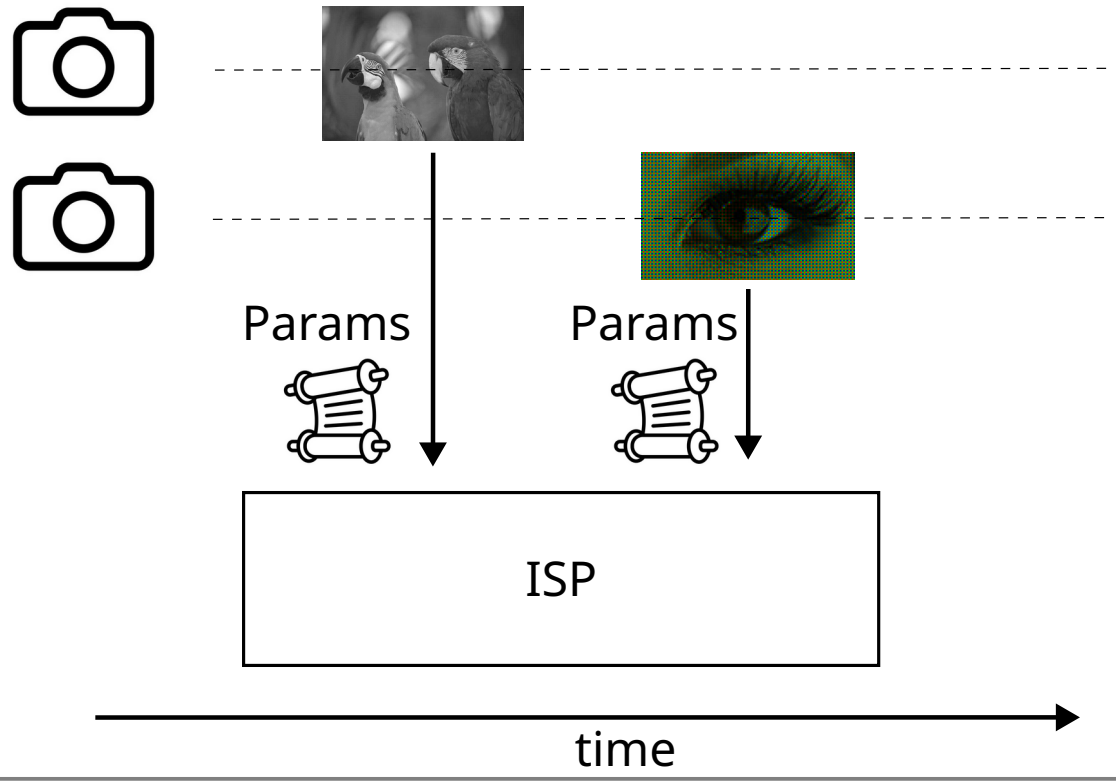
Jacopo Mondi

jacopo.mondi@ideasonboard.com

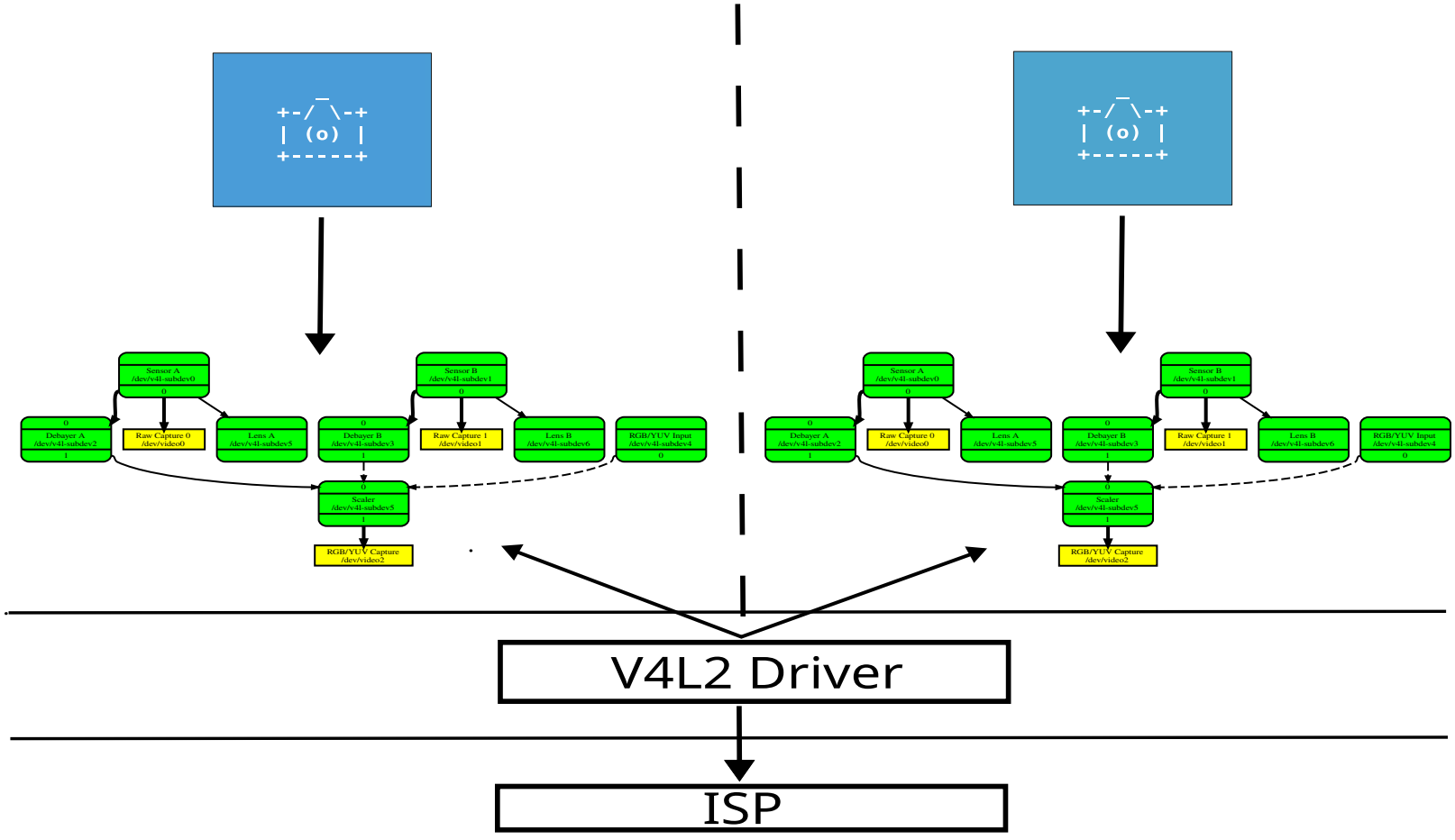
(Some) ISPs are time-multiplexed devices

- Resources are multiplexed at the HW or FW level and some ISP processes images in tiles
- ISPs can handle streams from different camera inputs by alternating ‘contexts’

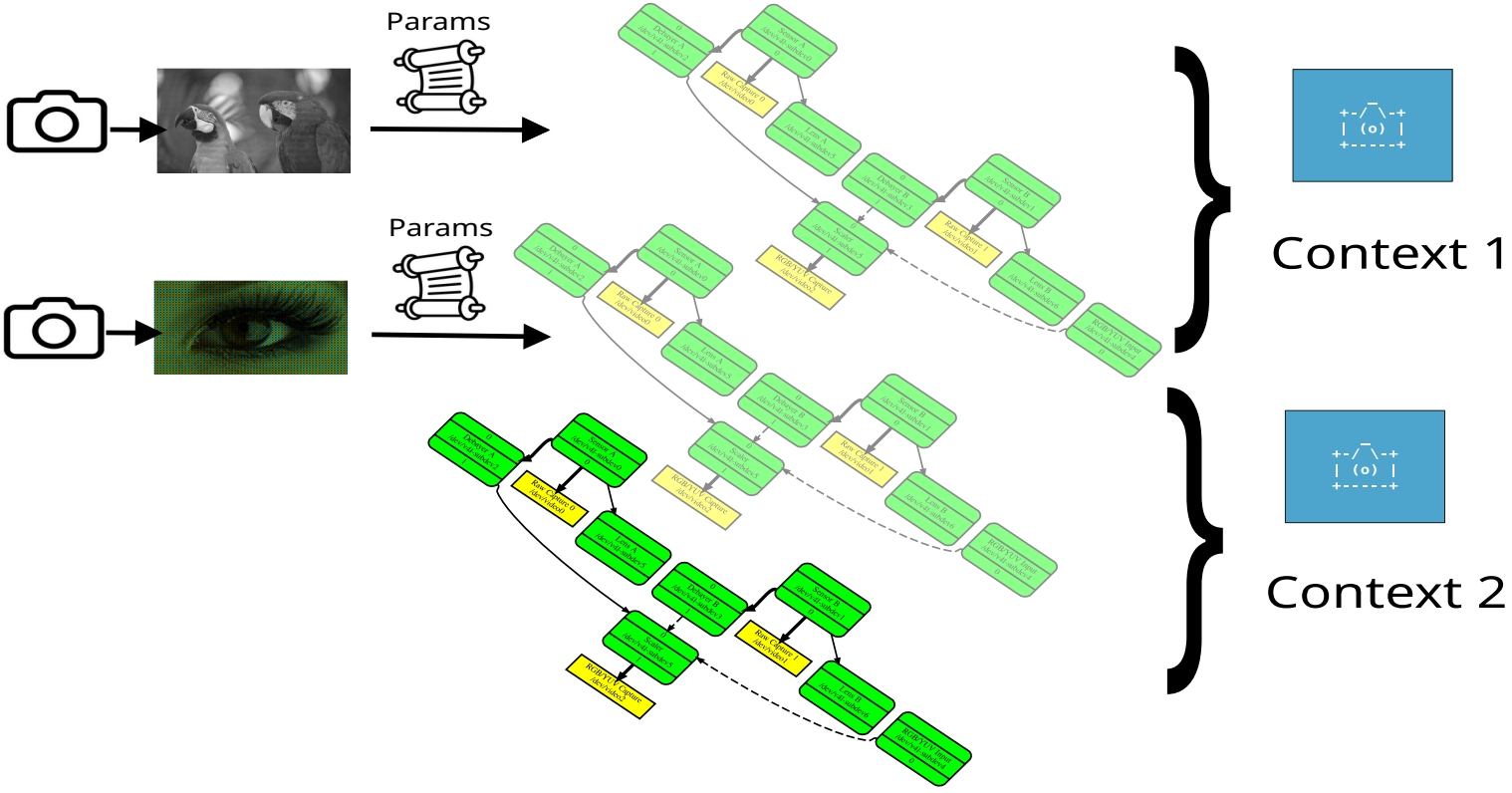
ISP time multiplexing



ISP time multiplexing



Media graph multiplexing



Introducing contexts

Contexts:

- Execution contexts stacked on a single instance of a media graph
- Isolated at the process level
- Associates in an isolated environment:
  - Video Devices
  - V4L2 Subdevices (todo)
  - Media device links state (todo)

Media contexts

- **Types:**
  - Media device context (MC)

  - Media entity context (MC)
    - Video device context (V4L2)
    - V4L2 subdevice context (V4L2)

- **IOCTLs**
  - VIDIOC_BIND_CONTEXT
  - VIDIOC_SUBDEV_BIND_CONTEXT (todo)

Media contexts: key design elements

Media entity context
- Refcounted
- Linked in the contexts list of *struct media_device_context*

Video device context
- Referenced by *struct v4l2-fh*
- Extends *media entity context*
- Stores *struct vb2_queue*

V4L2 subdevice context (todo)
- Referenced from *struct v4l2-subdev-fh*
- Extends *media entity context*
- Stores v4l2_subdev_state

Media entity context

VIDIOC_BIND_CONTEXT(media_fd)
VIDIOC_SUBDEV_BIND_CONTEXT(media_fd)

- Create a *video_device_context*
- Add it to the list of contexts in *media_device_context*
- Stores a reference to in the *struct v4l2-fh*

The uAPI: VIDIOC_BIND_CONTEXT

VIDIOC_BIND_CONTEXT(media_fd)

```
media_fd = open("/dev/mediaX", ...);
video_fd = open("/dev/videoX", ...);

struct video_device_context c = {
        .context_fd = media_fd;
};

ioctl(video_fd, VIDIOC_BIND_CONTEXT, &c);

/*
 * Operate the video device as usual.
 * 'video_fd' is bound to a specific context.
 */
ioctl(video_fd, VIDIOC_..., ...);
```

VIDIOC_BIND_CONTEXT(media_fd)

```
/* Create a different context. */
media_fd2 = open("/dev/mediaX", ...);
video_fd2 = open("/dev/videoX", ...);

struct video_device_context c2 = {
        .context_fd = media_fd2;
};

ioctl(video_fd2, VIDIOC_BIND_CONTEXT, &c2);

/*
 * 'video_fd' and 'video_fd2' operates on different
 * contexts.
 */
ioctl(video_fd, VIDIOC_..., ...);
ioctl(video_fd2, VIDIOC_..., ...);
```

Rebased:

- Still depends on Sakari's lifetime management series
  - Mostly for the introduction of "*struct media_device_fh*"

There is plan to continue working on this:

- Originally implemented on RPi5: still a target
- A second ISP driver will be targeted
  - Time allocated to work on this in the next months

V4l2 Controls (not for today)

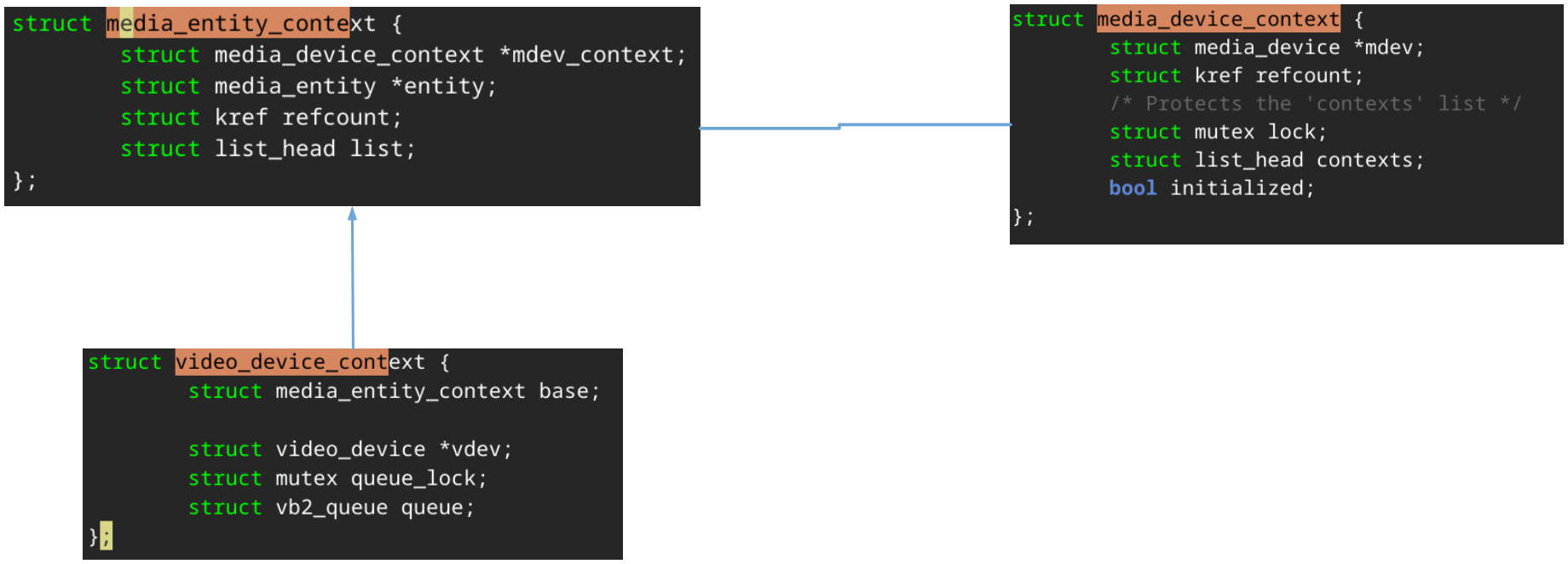
- I’m no expert there and I don’t know if this is possible or even desirable

m2m context

- Should the two be unified ?

Open questions (from last year)

Let’s have a recap of the types



Let’s have a recap of the types

```
struct media_device_context {
        struct media_device *mdev;
        struct kref refcount;
        /* Protects the 'contexts' list */
        struct mutex lock;
        struct list_head contexts;
        bool initialized;
};
```

```
struct media_entity_context {
        struct media_device_context *mdev_context;
        struct media_entity *entity;
        struct kref refcount;
        struct list_head list;
};
```

```
struct video_device_context {
        struct media_entity_context base;

        struct video_device *vdev;
        struct mutex queue_lock;
        struct vb2_queue queue;
};
```

```
struct v4l2_m2m_queue_ctx {
        struct vb2_queue        q;

        struct list_head        rdy_queue;
        spinlock_t              rdy_spinlock;
        u8                      num_rdy;
        bool                    buffered;
};
```

IDEAS ON BOARD

mem2mem context

```
struct media_entity_context {
        struct media_device_context *mdev_context;
        struct media_entity *entity;
        struct kref refcount;
        struct list_head list;
};
```

```
struct video_device_context {
        struct media_entity_context base;

        struct video_device *vdev;
        struct mutex queue_lock;
        struct vb2_queue queue;
};
```

```
struct v4l2_m2m_queue_ctx {
        struct vb2_queue        q;

        struct list_head        rdy_queue;
        spinlock_t              rdy_spinlock;
        u8                      num_rdy;
        bool                    buffered;
};
```

Both types have a vb2_queue

Inheriting from *struct media_entity_context* gives free reference counting

But what about the media device context and bounding ?

mem2mem context

mem2mem and media contexts

- mem2mem is about tying together an OUTPUT and a CAPTURE queue on a single video device
- Media context is about bounding together different video devices and subdevices and associate them with a *media device context*

Open questions