Multi format streams

Sakari Ailus
<sakari.ailus@linux.intel.com>
2013-10-22

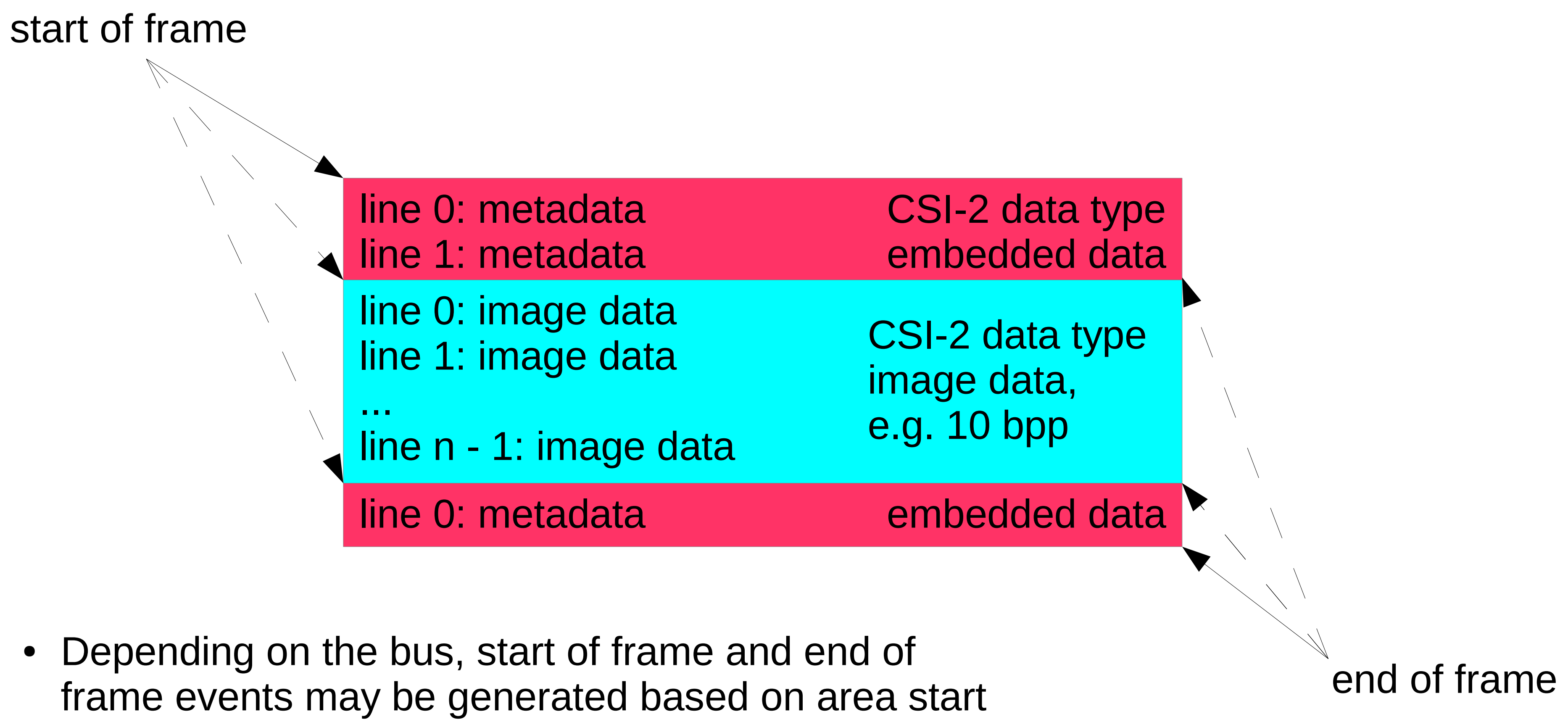
Use cases

- Camera sensors that transmit multiple streams simultaneously
- It's important to be able to separate these as they are independent of each other
  - Written to memory for software processing
  - Processing in other hardware entities
- Raw bayer
  - Metadata
  - Statistics
- SoC cameras
  - Statistics
  - YUV and JPEG
  - Interleaved YUV and JPEG for reduced memory requirements

Raw bayer

- Metadata is typically a few lines in the beginning of the frame
  - Sometimes uses a different data type so that the receiver can easily separate it from the image, depending on the bus
- ISPs must not process the metadata
  - Scaling or noise filtering, huh?

Raw bayer example (CSI-2)

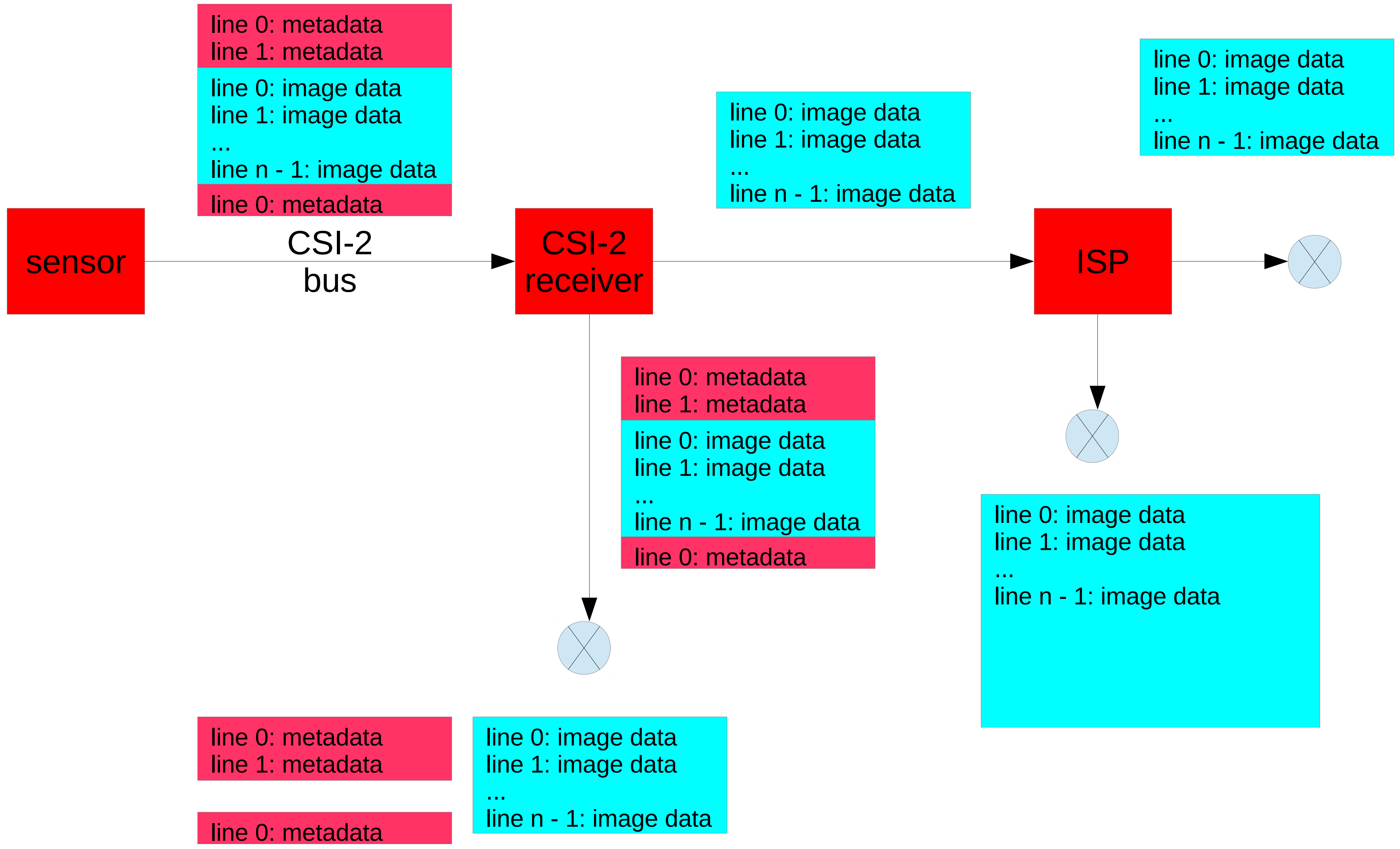


- Depending on the bus, start of frame and end of frame events may be generated based on area start or end
  - The above frame would thus have three start of frame and end of frame events

Raw bayer, continued

- The sensors could use a separate virtual channel to separate different areas of the frame (metadata, image data etc.)
  - Are there any?
  - This could make sense in the future
- Not all receivers support separation by data type

Raw bayer, continued



What do we need, then?

- Interface to tell the what's being transmitted on the bus
- V4L2 sub-device API changes in media bus format access
- V4L2 API changes to provide access to multiple streams
  - Multiple video nodes are not an option: video nodes would need to be created and destroyed based on the sensor configuration

Frame format descriptors

- Describe what an image source transmits
  - More details than struct v4l2_mbus_framefmt has
- Set by the image source driver
- Read-only
  - Relatively complex data structure
  - Changes should be made through a different interface

Frame format descriptors, continued

```
struct v4l2_mbus_frame_desc {
        struct v4l2_mbus_frame_desc_entry \
                entry[V4L2_MBUS_FRAME_DESC_ENTRY_MAX];
        unsigned short num_entries;
};

#define V4L2_MBUS_FRAME_DESC_ENTRY_FLAG_BLOB            (1 << 0)
#define V4L2_MBUS_FRAME_DESC_ENTRY_FLAG_LEN_IS_MAX      (1 << 1)

enum {
        V4L2_MBUS_FRAME_DESC_TYPE_CSI2,
        V4L2_MBUS_FRAME_DESC_TYPE_CCP2,
        V4L2_MBUS_FRAME_DESC_TYPE_PARALLEL,
};

struct v4l2_mbus_frame_desc_entry {
        u8 bpp;
        u16 flags;
        u32 pixelcode;
        union {
                struct {
                        u16 width;
                        u16 height;
                        u16 start_line;
                };
                u32 length; /* if BLOB flag is set */
        };
        unsigned int type;
        union {
                struct v4l2_mbus_frame_desc_entry_csi2 csi2;
                struct v4l2_mbus_frame_desc_entry_ccp2 ccp2;
                struct v4l2_mbus_frame_desc_entry_parallel par;
        };
};

struct v4l2_mbus_frame_desc_entry_csi2 {
        u8 channel;
};
```

- A new pad op is needed for obtaining the frame descriptor
- The main image must always come first for backward compatibility on the user space interface

V4L2 sub-device: media bus formats

- The current V4L2 sub-device interface for media bus format assumes a single format per pad
  - But we'd need many
  - Number of independent parts of the image could depend on image source configuration
  - Links model physical connections
    - Adding more links is thus not an option

V4L2 sub-device: media bus formats, continued

- A new field, format_index, could be added to the relevant IOCTL argument structs such as
  - struct v4l2_subdev_format,
  - struct v4l2_subdev_mbus_code_enum,
  - struct v4l2_subdev_frame_size_enum and
  - struct v4l2_subdev_selection
- This provides a way to access the additional formats

V4L2: access to multiple streams

- Formats are bound to video buffer queues
- If one wants to capture multiple, independent streams handled by the same DMA engine, an independent v4l2_format is required
- Two approaches
  - Extend multi-plane buffers to multi-format buffers
  - Multiplex buffer queues by index in addition to type

V4L2: from multi-plane buffers to multi-format buffers

- Extend multi-plane buffers to multi-format buffers
- Make format information specific to plane instead of the entire set of planes
- Clean and pretty, isn't it?

But...

- 14 bytes of reserved fields per plane
  - Hardly enough for all that's currently in struct v4l2_pix_format
- Forces capturing of all streams, all the time
  - The image source might not even transmit them
- Same queue length for every stream
  - Video recording might need, say, four, but still capture during the recording could survive with just two
- Buffers from different stream will finish at different periods of time
  - Especially the metadata is important for the 3A control loop

V4L2: multiple video buffer queues per video node

- Previously multiple buffer queues were possible but they always involved a different buffer type
- Add another field that allows multiplexing the same video node
- As the streams are independent, this avoids the issues that using multi-planar buffers had

V4L2: multiple video buffer queues per video node

- Previously multiple buffer queues were possible but they always involved a different buffer type
- Add another field that allows multiplexing the same video node
- As the streams are independent, this avoids the issues that using multi-planar buffers had

V4L2: multiple video buffer queues per video node, continued

- Most IOCTL argument structs have free reserved fields
  - struct v4l2_format has none, but we could steal up to 8 bytes from the union --- the largest struct consumes 192 bytes of 200
- We could also split the type field
  - 16 bits for buffer types and streams ought to be enough for everybody
  - Requires recognising the programs that can use the feature
    - Well that's easy: they enumerate the streams, but this is still hackish
- The first stream must be the main image one for backward compatibility

V4L2: metadata buffer type

- A new buffer type for metadata is much neater and cleaner, but only provides a partial solution
  - Several buffer queues of the same type may not exist on a single video node
  - V4L2_BUF_TYPE_VIDEO_CAPTURE2
    - Argh!