Arm Framebuffer Compression (AFBC)

AFBC is a proprietary lossless image compression protocol and format. It provides fine-grained random access and minimizes the amount of data transferred between IP blocks.

AFBC can be enabled on drivers which support it via use of the AFBC format modifiers defined in drm_fourcc.h. See DRM_FORMAT_MOD_ARM_AFBC(*).

All users of the AFBC modifiers must follow the usage guidelines laid out in this document, to ensure compatibility across different AFBC producers and consumers.

Components and Ordering

AFBC streams can contain several components - where a component corresponds to a color channel (i.e. R, G, B, X, A, Y, Cb, Cr). The assignment of input/output color channels must be consistent between the encoder and the decoder for correct operation, otherwise the consumer will interpret the decoded data incorrectly.

Furthermore, when the lossless colorspace transform is used (AFBC_FORMAT_MOD_YTR, which should be enabled for RGB buffers for maximum compression efficiency), the component order must be:

- Component 0: R
- Component 1: G
- Component 2: B

The component ordering is communicated via the fourcc code in the fourcc:modifier pair. In general, component '0' is considered to reside in the least-significant bits of the corresponding linear format. For example, COMP(bits):

- DRM_FORMAT_ABGR8888
  - Component 0: R(8)
  - Component 1: G(8)
  - Component 2: B(8)
  - Component 3: A(8)
- DRM_FORMAT_BGR888
  - Component 0: R(8)
  - Component 1: G(8)
  - Component 2: B(8)
- DRM_FORMAT_YUYV
  - Component 0: Y(8)
  - Component 1: Cb(8, 2x1 subsampled)
  - Component 2: Cr(8, 2x1 subsampled)

In AFBC, 'X' components are not treated any differently from any other component. Therefore, an AFBC buffer with fourcc DRM_FORMAT_XBGR8888 encodes with 4 components, like so:

- DRM_FORMAT_XBGR8888
  - Component 0: R(8)
  - Component 1: G(8)
  - Component 2: B(8)
  - Component 3: X(8)

Please note, however, that the inclusion of a "wasted" 'X' channel is bad for compression efficiency, and so it's recommended to avoid formats containing 'X' bits. If a fourth component is required/expected by the encoder/decoder, then it is recommended to instead use an equivalent format with alpha, setting all alpha bits to '1'. If there is no requirement for a fourth component, then a format which doesn't include alpha can be used, e.g. DRM_FORMAT_BGR888.

Number of Planes

Formats which are typically multi-planar in linear layouts (e.g. YUV 420), can be encoded into one, or multiple, AFBC planes. As with component order, the encoder and decoder must agree about the number of planes in order to correctly decode the buffer. The fourcc code is used to determine the number of encoded planes in an AFBC buffer, matching the number of planes for the linear (unmodified) format. Within each plane, the component ordering also follows the fourcc code:

For example:

- DRM_FORMAT_YUYV: nplanes = 1
  - Plane 0:
    - Component 0: Y(8)

        - Component 1: Cb(8, 2x1 subsampled)
        - Component 2: Cr(8, 2x1 subsampled)
- DRM_FORMAT_NV12: nplanes = 2
    - Plane 0:
        - Component 0: Y(8)
    - Plane 1:
        - Component 0: Cb(8, 2x1 subsampled)
        - Component 1: Cr(8, 2x1 subsampled)

Cross-device interoperability

For maximum compatibility across devices, the table below defines canonical formats for use between AFBC-enabled devices. Formats which are listed here must be used exactly as specified when using the AFBC modifiers. Formats which are not listed should be avoided.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)afbc.rst, line 119)

Unknown directive type "flat-table".

```
.. flat-table:: AFBC formats

   * - Fourcc code
     - Description
     - Planes/Components

   * - DRM_FORMAT_ABGR2101010
     - 10-bit per component RGB, with 2-bit alpha
     - Plane 0: 4 components
              * Component 0: R(10)
              * Component 1: G(10)
              * Component 2: B(10)
              * Component 3: A(2)

   * - DRM_FORMAT_ABGR8888
     - 8-bit per component RGB, with 8-bit alpha
     - Plane 0: 4 components
              * Component 0: R(8)
              * Component 1: G(8)
              * Component 2: B(8)
              * Component 3: A(8)

   * - DRM_FORMAT_BGR888
     - 8-bit per component RGB
     - Plane 0: 3 components
              * Component 0: R(8)
              * Component 1: G(8)
              * Component 2: B(8)

   * - DRM_FORMAT_BGR565
     - 5/6-bit per component RGB
     - Plane 0: 3 components
              * Component 0: R(5)
              * Component 1: G(6)
              * Component 2: B(5)

   * - DRM_FORMAT_ABGR1555
     - 5-bit per component RGB, with 1-bit alpha
     - Plane 0: 4 components
              * Component 0: R(5)
              * Component 1: G(5)
              * Component 2: B(5)
              * Component 3: A(1)

   * - DRM_FORMAT_VUY888
     - 8-bit per component YCbCr 444, single plane
     - Plane 0: 3 components
              * Component 0: Y(8)
              * Component 1: Cb(8)
              * Component 2: Cr(8)

   * - DRM_FORMAT_VUY101010
     - 10-bit per component YCbCr 444, single plane
     - Plane 0: 3 components
              * Component 0: Y(10)
              * Component 1: Cb(10)
              * Component 2: Cr(10)
```

```
 * - DRM_FORMAT_YUYV
   - 8-bit per component YCbCr 422, single plane
   - Plane 0: 3 components
            * Component 0: Y(8)
            * Component 1: Cb(8, 2x1 subsampled)
            * Component 2: Cr(8, 2x1 subsampled)

 * - DRM_FORMAT_NV16
   - 8-bit per component YCbCr 422, two plane
   - Plane 0: 1 component
            * Component 0: Y(8)
     Plane 1: 2 components
            * Component 0: Cb(8, 2x1 subsampled)
            * Component 1: Cr(8, 2x1 subsampled)

 * - DRM_FORMAT_Y210
   - 10-bit per component YCbCr 422, single plane
   - Plane 0: 3 components
            * Component 0: Y(10)
            * Component 1: Cb(10, 2x1 subsampled)
            * Component 2: Cr(10, 2x1 subsampled)

 * - DRM_FORMAT_P210
   - 10-bit per component YCbCr 422, two plane
   - Plane 0: 1 component
            * Component 0: Y(10)
     Plane 1: 2 components
            * Component 0: Cb(10, 2x1 subsampled)
            * Component 1: Cr(10, 2x1 subsampled)

 * - DRM_FORMAT_YUV420_8BIT
   - 8-bit per component YCbCr 420, single plane
   - Plane 0: 3 components
            * Component 0: Y(8)
            * Component 1: Cb(8, 2x2 subsampled)
            * Component 2: Cr(8, 2x2 subsampled)

 * - DRM_FORMAT_YUV420_10BIT
   - 10-bit per component YCbCr 420, single plane
   - Plane 0: 3 components
            * Component 0: Y(10)
            * Component 1: Cb(10, 2x2 subsampled)
            * Component 2: Cr(10, 2x2 subsampled)

 * - DRM_FORMAT_NV12
   - 8-bit per component YCbCr 420, two plane
   - Plane 0: 1 component
            * Component 0: Y(8)
     Plane 1: 2 components
            * Component 0: Cb(8, 2x2 subsampled)
            * Component 1: Cr(8, 2x2 subsampled)

 * - DRM_FORMAT_P010
   - 10-bit per component YCbCr 420, two plane
   - Plane 0: 1 component
            * Component 0: Y(10)
     Plane 1: 2 components
            * Component 0: Cb(10, 2x2 subsampled)
            * Component 1: Cr(10, 2x2 subsampled)
```