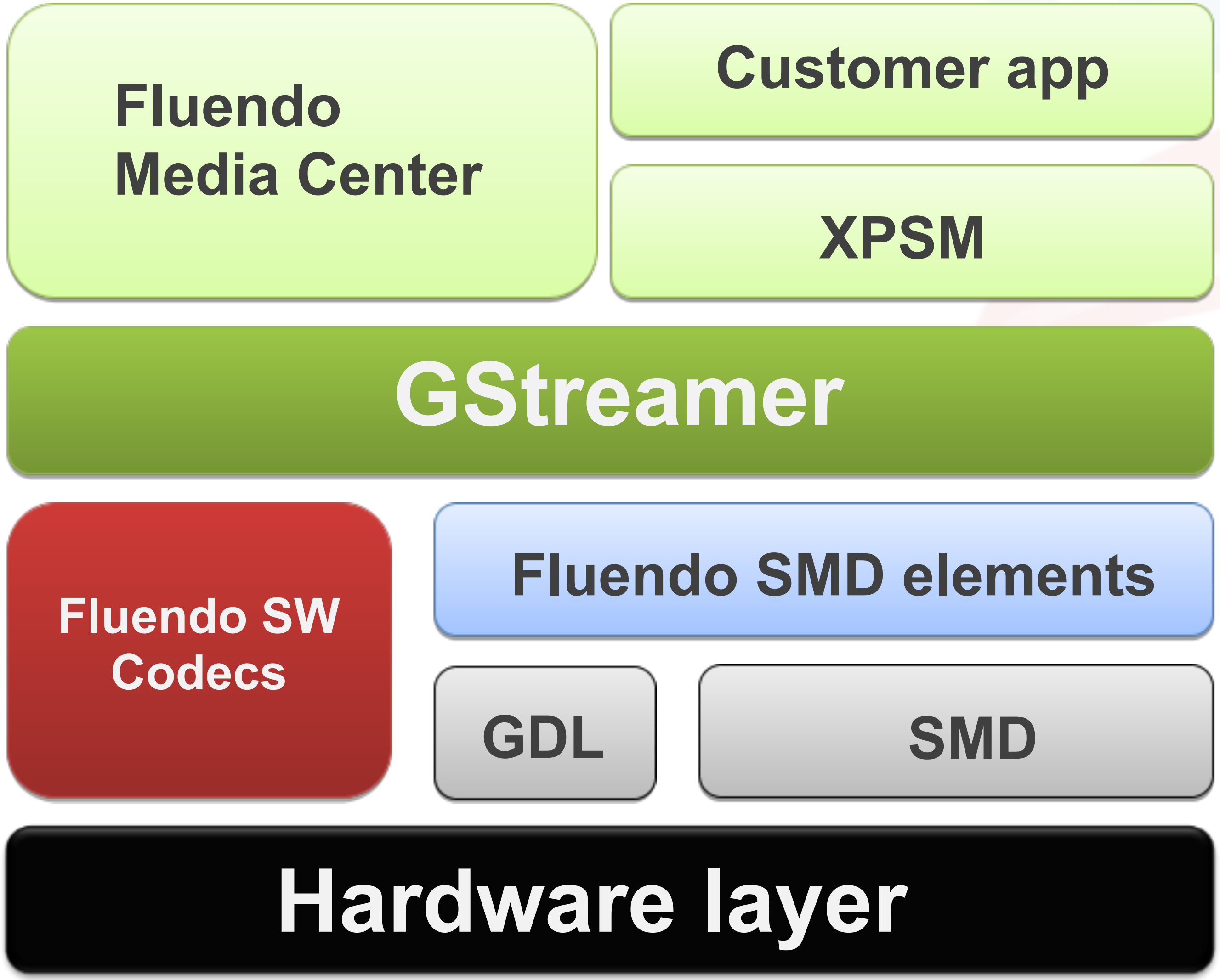
FLUENDO GStreamer Elements for CanMore and SodaVille Systems.

Julien Moutte (2010)
julien@fluendo.com

CanMore / SodaVille block diagram



Fluendo Media Center

Customer app

XPSM

GStreamer

Fluendo SW Codecs

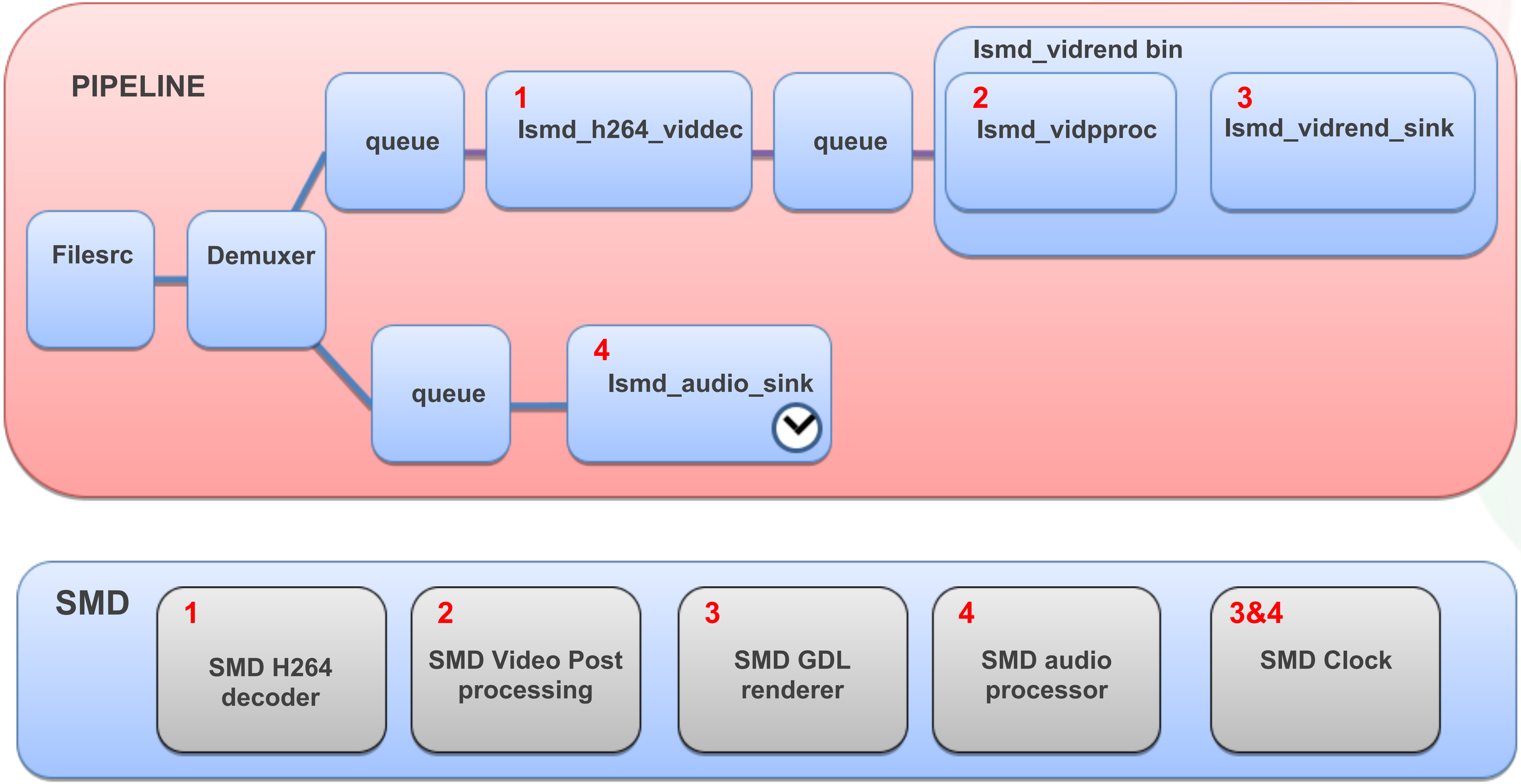
Fluendo SMD elements

GDL

SMD

Hardware layer

Typical GStreamer pipeline on CanMore/SodaVille (fig 1)



Pipeline explanation (see fig 1)

- Encoded video and audio buffers are pushed by the demuxer to Fluendo SMD elements. These elements will convert these software buffers into SMD buffers and push them to the SMD component they are wrapping.

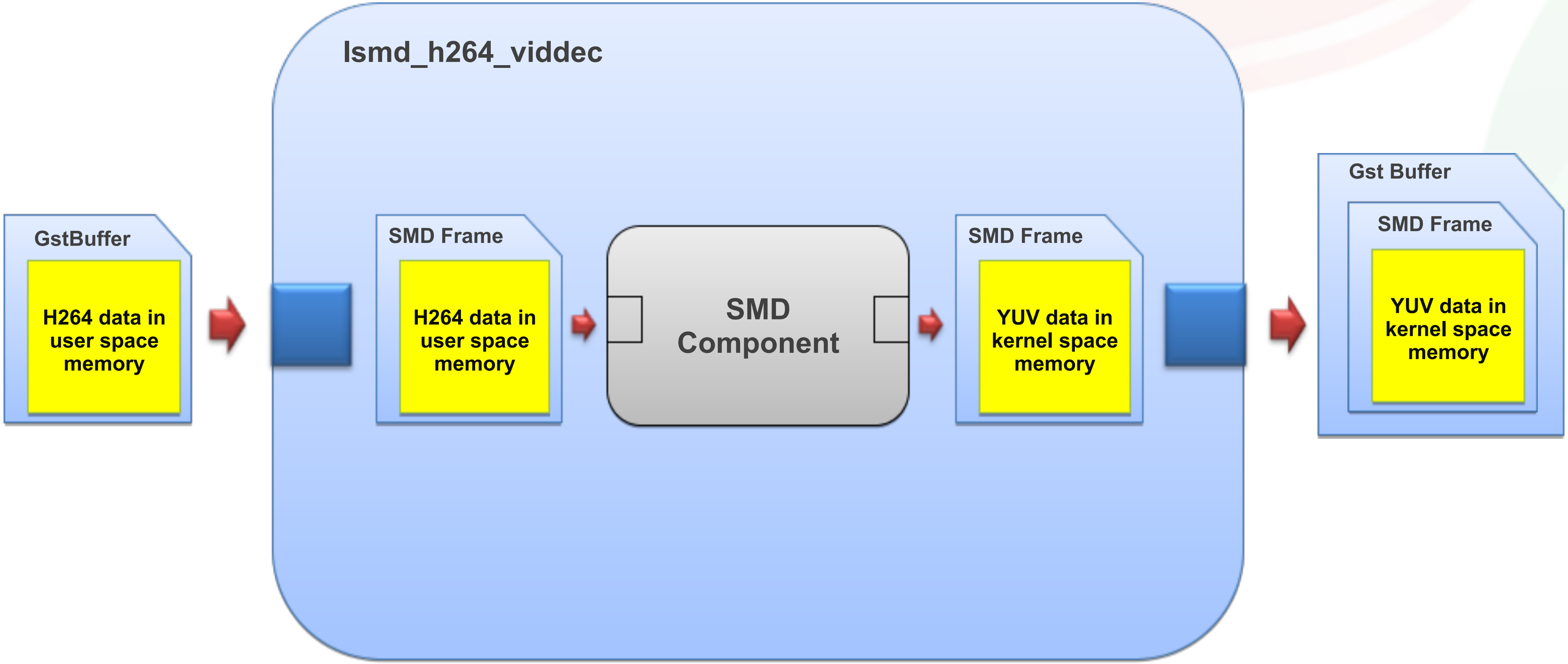
- The Fluendo SMD elements will manage and monitor the SMD component to catch any processed output. When something comes out the Fluendo SMD element will relay the output to the next element.

- The video and audio rendering elements can be a clock provider. That means that they expose the SMD clock in the GStreamer pipeline automatically so that synchronization and trick modes can happen correctly using the SMD components for rendering.

- SMD components are not connected directly. The application layer does not need to do anything related to the SMD API, everything is handled by the GStreamer graph and the elements involved inside.

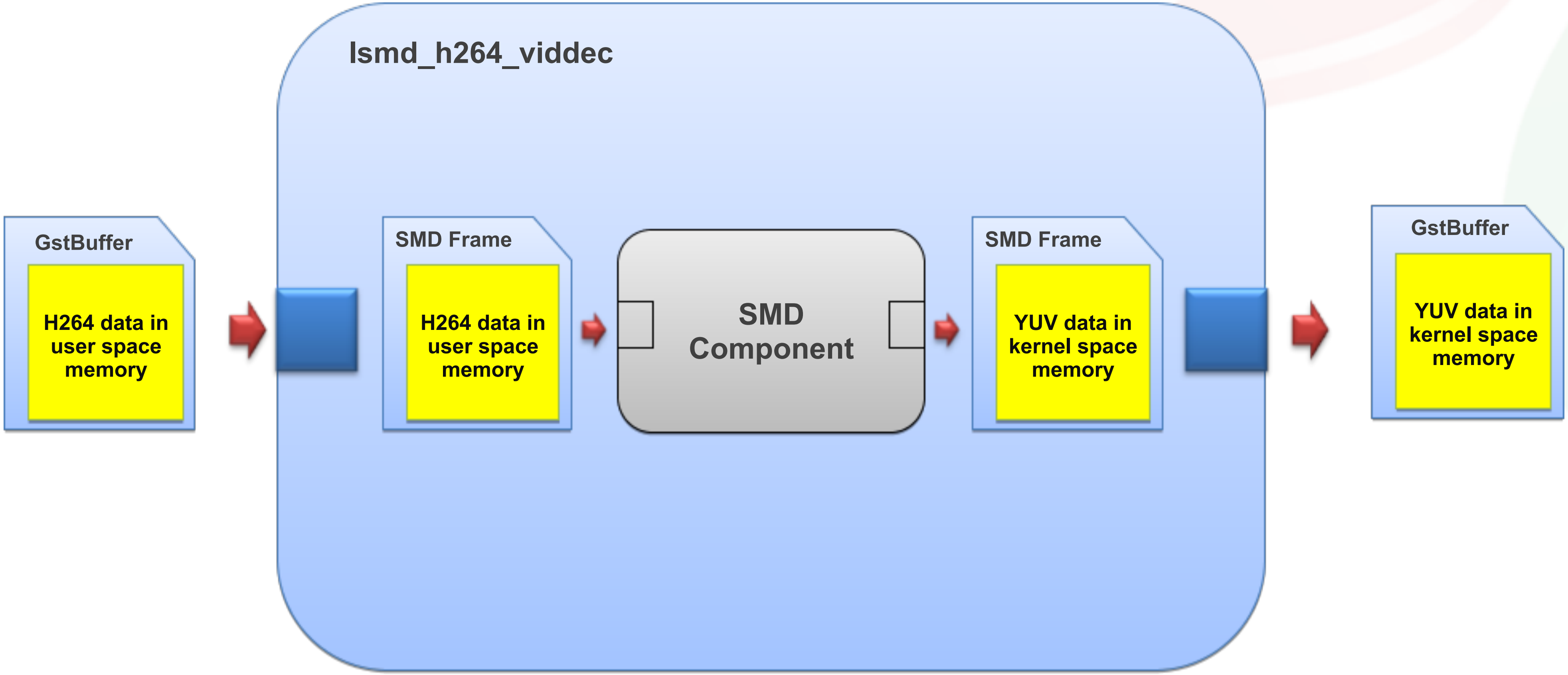
Description of a Fluendo SMD element (fig 2a)

Simple decoder receiving user space memory encoded buffers, decoding in hardware and transmitting to next element keeping data in hardware space


GstBuffer
H264 data in user space memory
Ismd_h264_viddec
SMD Frame
H264 data in user space memory
SMD Component
SMD Frame
YUV data in kernel space memory
Gst Buffer
SMD Frame
YUV data in kernel space memory

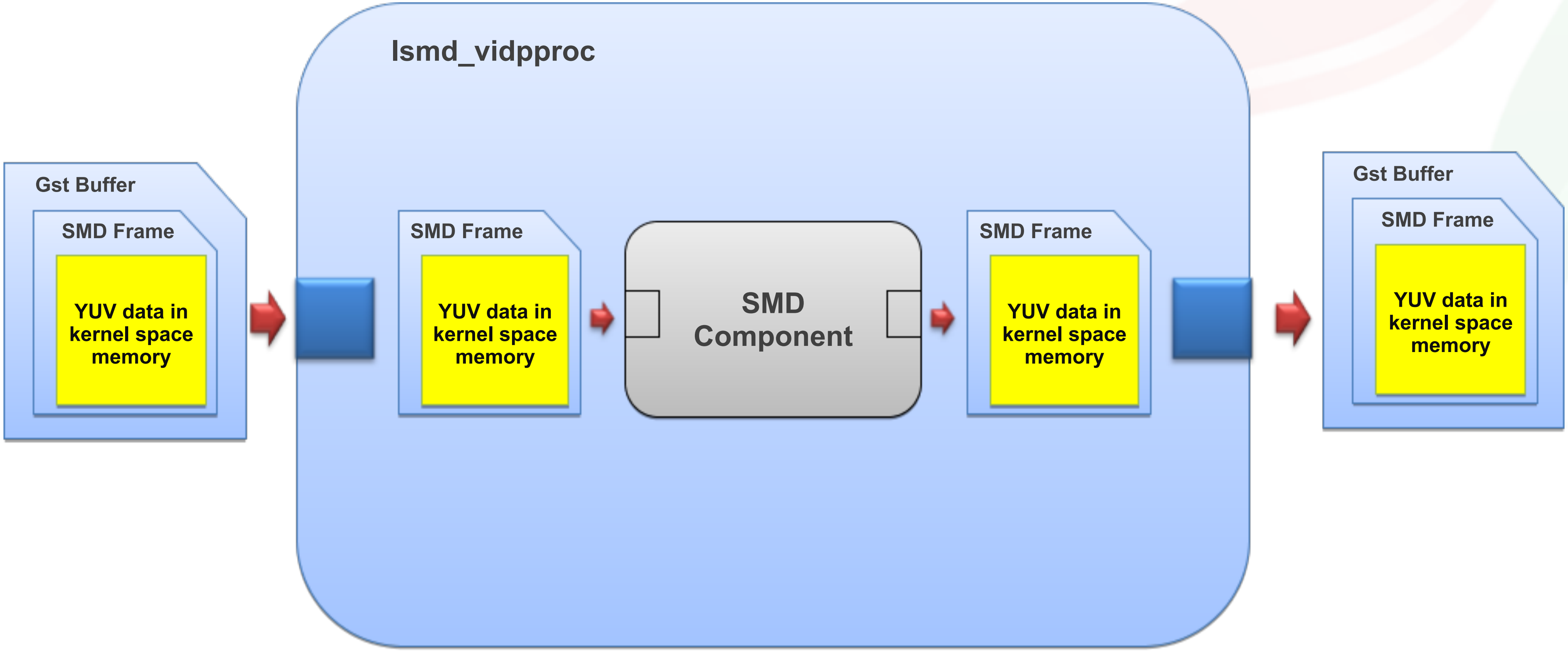
Description of a Fluendo SMD element (fig 2b)

Simple decoder receiving user space memory encoded buffers, decoding in hardware and transmitting to next element in user space memory. (for demo purpose)


Ismd_h264_viddec
GstBuffer
H264 data in user space memory
SMD Frame
H264 data in user space memory
SMD Component
SMD Frame
YUV data in kernel space memory
GstBuffer
YUV data in kernel space memory

Description of a Fluendo SMD element (fig 2c)

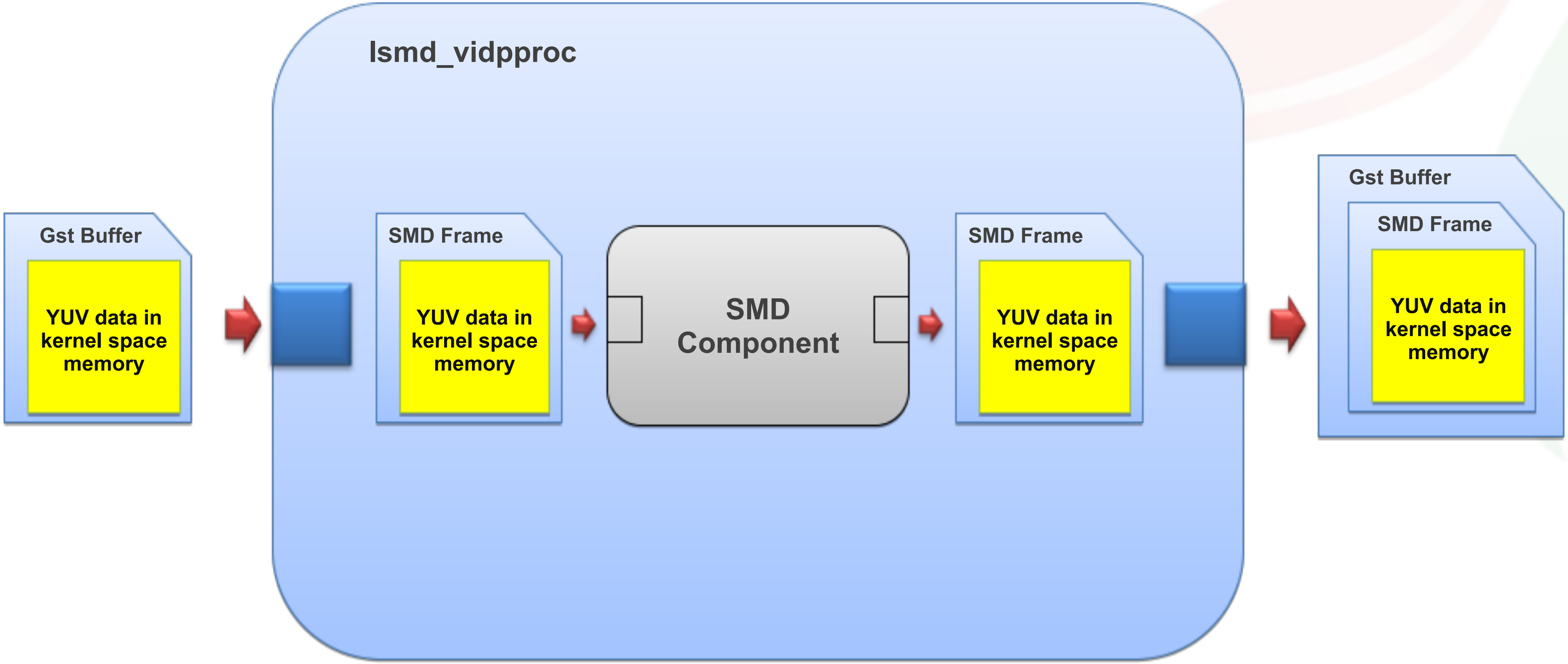
Simple filter receiving kernel space memory decoded buffers, processing in hardware and transmitting to next element keeping data in hardware space



Ismd_vidpproc
Gst Buffer
SMD Frame
YUV data in kernel space memory
SMD Frame
YUV data in kernel space memory
SMD Component
SMD Frame
YUV data in kernel space memory
Gst Buffer
SMD Frame
YUV data in kernel space memory

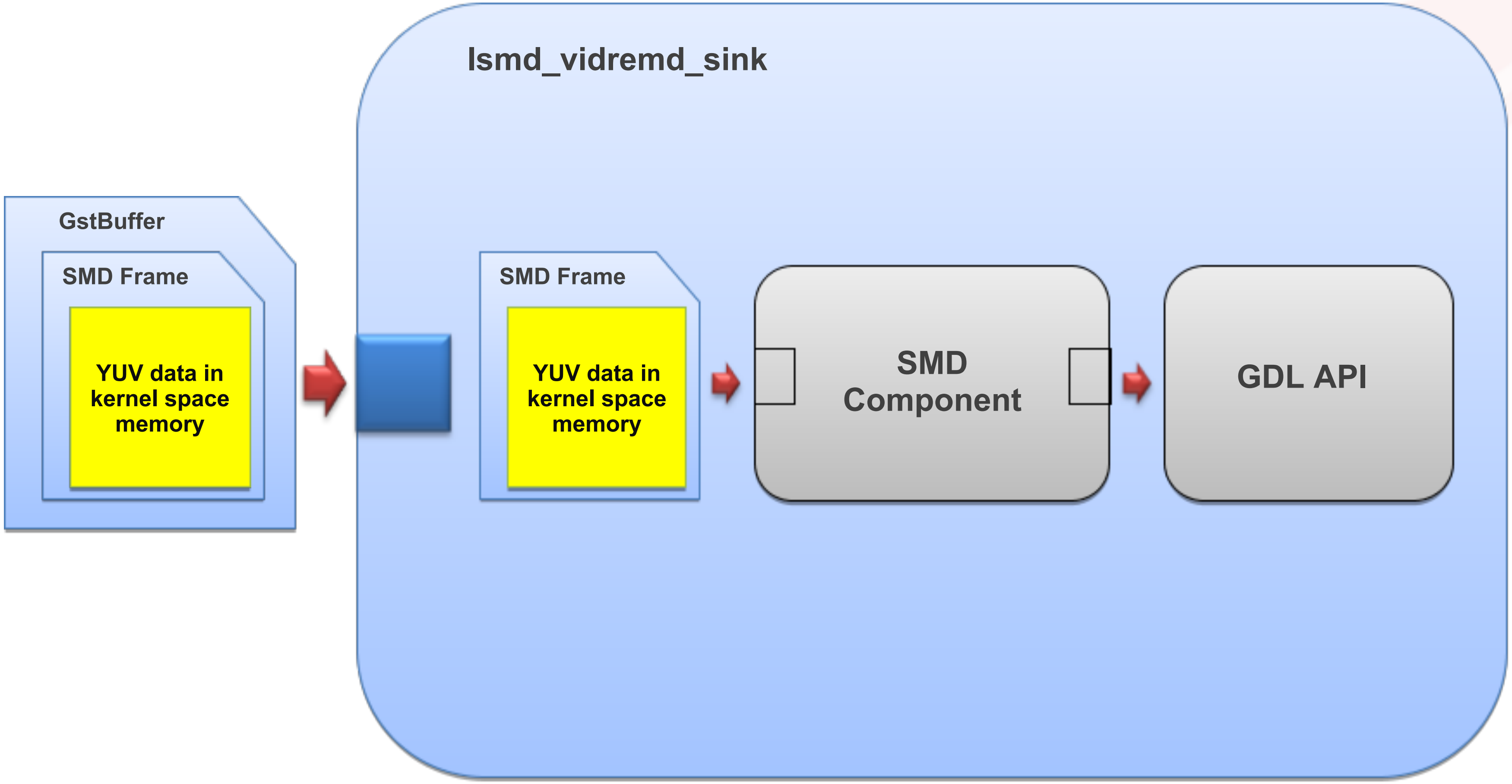
Description of a Fluendo SMD element (fig 2d)

Simple filter receiving user space memory decoded buffers (with proper stride), processing in hardware and transmitting to next element uploading data in hardware space


Ismd_vidpproc
Gst Buffer
YUV data in kernel space memory
SMD Frame
YUV data in kernel space memory
SMD Component
SMD Frame
YUV data in kernel space memory
Gst Buffer
SMD Frame
YUV data in kernel space memory

Description of a Fluendo SMD element (fig 2e)

Rendering element receiving kernel space memory decoded buffers.



More explanations

- Each Fluendo SMD element is able to detect if it is receiving or needs to emit user space data (normal GstBuffers) or kernel space data (GstBuffers wrapping SMD frames). It will then do the appropriate conversion to and from the wrapped SMD component.
- SMD components are treated as completely autonomous processing units, they don't talk to each other directly. Every data exchange happens by passing around SMD frames.
- This allows to keep data flow happening in GStreamer controlling prerolling, queueing and buffering of data and proper application development control over the pipeline.

Example : It is possible to connect hardware H264 decoder to a queue and then to the video post processing element. The queue will fill up with wrapped buffers allowing to control the amount of bytes/time in the queue. In the case of audio rendering element this allows to manage proper prerolling of the hardware queues before commiting state change to the pipeline for proper synchronization.

Fluendo SMD elements list, roles and properties

- **ismd_audio_sink:** Intel Streaming Media Driver (ISMD) Hardware Audio Renderer Sink
  Role : Decodes and render compressed or raw audio frames to different audio output routes.
  Input formats : MPEG 1 and 2 audio, AAC, AC3, EAC3, DTS, WMA, LPCM, PCM

  Properties :

  - global-processor : boolean, use global audio processor
  - audio-output-hdmi : enumeration, defines whether and how audio is sent on HDMI out.
  - audio-output-spdif : enumeration, defines whether and how audio is sent on SPDIF out.
  - audio-output-i2s0 : enumeration, defines whether and how audio is sent on i2s0 out.
  - audio-output-i2s1 : enumeration, defines whether and how audio is sent on i2s1 out.
  - render-delay : integer, additional render delay of the sink in milliseconds.
  - volume : float, volume scaling factor.

- **ismd_vidrend_bin:** Intel Streaming Media Driver (ISMD) Hardware Video Renderer bin
  Role : Render raw video frames (takes care of post processing) to a GDL plane.
  Input formats : SMD Frames or Raw YUV buffers with proper stride.

  Properties :

  - tvmode, bgcolor, gdl-plane, rectangle, scale-mode: wrapped properties of ismd_vidpproc and ismd_vidrend_sink described in next slide.

Fluendo SMD elements list, roles and properties (cont)

IMPORTANT: ismd_vidrend_bin wraps ismd_vidpproc and ismd_vidrend_sink. It is recommended to use the bin and not the separate elements. That' s what playbin2 will do automatically.

- **ismd_vidrend_sink**: Intel Streaming Media Driver (ISMD) Hardware Video Renderer Sink
  Role : Render raw video frames (post processing applied) to a GDL plane.
  Input formats : SMD Frames from post processor .

  Properties :

  - tvmode : enumeration, defines different resolutions for screen initialization.
  - bgcolor : integer, defines background color of the plane.
  - gdl-plane : enumeration, defines which GDL plane to use for video rendering.
  - rectangle: string "<x>,<y>,<width>,<height>" specifies the rendering area. Default "0,0,0,0" meaning full screen.

- **ismd_vidpproc:** Intel Streaming Media Driver (ISMD) Hardware Video Post-Processor
  Role : Video post processing, deinterlacing, scaling, stride management.
  Input formats : RAW YUV video and SMD frames.

  Properties :

  - rectangle: string "<x>,<y>,<width>,<height>" specifies the rendering area. Default "0,0,0,0" meaning full screen.
  - scale-mode: enumeration, defines the scaling mode.

Fluendo SMD elements list, roles and properties (cont)

- **ismd_vc1_viddec:** Intel Streaming Media Driver (ISMD) Hardware VC1 Video Decoder
  Role : VC1 video decoder.
  Input formats : Compressed VC1 or WMV9 bitstream.

- **ismd_h264_viddec:** Intel Streaming Media Driver (ISMD) Hardware H.264 Video Decoder
  Role : H264/AVC video decoder.
  Input formats : Compressed H.264/AVC bitstream.

- **ismd_mpeg2_viddec:** Intel Streaming Media Driver (ISMD) Hardware MPEG-2 Video Decoder
  Role : MPEG2 video decoder.
  Input formats : Compressed MPEG2 bitstream.

- **ismd_mpeg4_viddec:** Intel Streaming Media Driver (ISMD) Hardware MPEG-4 Video Decoder
  Role : MPEG4 Part 2 video decoder.
  Input formats : Compressed MPEG4 Part 2 bitstream (aka DivX).
  NOTE : Only available on SodaVille boards.

- **flugdlsink:** Fluendo GDL image rendering element
  Role : Renders ARGB video frames on a GDL plane, mostly used for subtitles.
  Input formats : ARGB video frames.

Fluendo SMD elements list, roles and properties (cont)

- **ismd_dvb_src:** Intel Streaming Media Driver (ISMD) Hardware DVB Source
  Role : Push the data produced in the DVB tuner to the pipeline.
  Output formats : MPEG Transport Stream.

  Properties :

  - tuner : specifies the TSI ID connected to the DVB tuner.
  - program : service ID to filter a MPTS into a SPTS.
  - psi-timeout : permit change the timout of the PSI filter.
  - extra-pids : permit add PIDs to the main output filter.
  - pid-filter : filter PIDs by category.
  - check-crc : discard incoming PCI packets with wrong CRC.
  - clock-recovery : enables clock recovery monitoring the PCR samples.

- **ismd_clock_recovery_provider:** Intel Streaming Media Driver (ISMD) Hardware Clock Recovery/Provider
  Role : Performs clock recoveries reading the PCR samples in the MPEG transport stream and provides a SMD clock.
  Input formats : MPEG Transport Stream.
  Properties :

  - pcr : enables clock recovery monitoring the PCR samples in the specified PID.

Sample pipelines with gst-launch-0.10

Thanks to Fluendo's SMD elements compliance with GStreamer design rules it is possible to run media pipelines with the gst-launch-0.10 command line tool. Here are some examples :

- **Simple video test pattern on GDL :**
  gst-launch-0.10 videotestsrc ! queue ! flugdlsink
- ***Simple video test pattern on SMD renderer :***
  gst-launch-0.10 videotestsrc ! queue ! ismd_vidrend_bin
- **Simple audio test signal :**
  gst-launch-0.10 audiotestsrc ! queue ! ismd_audio_sink

Sample pipelines with gst-launch-0.10 (cont)

- **Simple A/V decoding from Quicktime :**
  gst-launch-0.10 filesrc location=file.mov ! qtdemux name=d ! queue ! ismd_h264_viddec ! queue max-size-buffers=3 ! ismd_vidrend_bin d. ! queue ! ismd_audio_sink

- **Playbin2 example :**
  gst-launch-0.10 playbin2 flags=0x43 uri=file:///file.mov

- ***UDP src with clock recovery example :***
  gst-launch-0.10 udpsrc port=<xxxx> ! typefind ! queue max-size-buffers=0 max-size-time=0 ! ismd_clock_recovery_provider pcr=<pid> ! flutsdemux name=d ! queue ! ismd_mpeg2_viddec ! queue max-size-buffers=3 ! ismd_vidrend_bin d. ! queue ! ismd_audio_sink

Trick modes support

In GStreamer trick modes are supported using the SEEK event with a specific rate differing from 1.0 where :

- rate > 1.0 means fast forward playback
- 0 < rate < 1.0 means slow motion
- rate < 0 means reverse playback at different speeds (-1.0 for normal speed rewind, -2.0 for x2 rewind, etc...)

Fluendo SMD elements support trick modes with that **API**. Obviously the hardware decoding is not able to cope with any decoding rate so some Quality Of Service management will happen automatically when a non supported rate is requested. That means that when the requested rate goes beyond the hardware capabilities the **Fluendo SMD** element will switch to keyframe only decoding.

SMD BufMon support

When the ismd_dvb_src or ismd_clock_recovery_provider elements are used in a pipeline is enabled BufMon.

In PLAYING state the SMD BufMon object is used to change the clock values when a buffer underrun is detected in the renderer elements.

This is intended to react when a signal loose in fixed rate data flow scenarios like DVB or IPTV.

It behaves like pausing the clock until the underrun situation is solved.