Rockchip Gstreamer User Guide

ID: RK-YH-YF-921

Release Version: V1.1.1

Release Date: 2022-07-26

Security Level: □Top-Secret □Secret □Internal ■Public

Rockchip Electronics Co., Ltd.

No.18 Building, A District, No.89, software Boulevard Fuzhou, Fujian,PRC

Website: www.rock-chips.com

Customer service Tel: +86-4007-700-590

Customer service Fax: +86-591-83951833

Customer service e-Mail: fae@rock-chips.com

Preface

Overview

This document is going to introduce the ways to build and test Gstreamer and related plugins.

Product Version

Chipset	Version
RK356X	1.14.x
RK3588	1.18.x

Intended Audience

This document (this guide) is mainly intended for:

Technical support engineers

Software development engineers

Revision History

Date	Version	Author	修改说明
2022-01-06	V1.0.0	Jair Wu	Initial version
2022-02-24	V1.0.1	Jair Wu	Fix a wrong command option
2022-05-10	V1.1.0	Jair Wu	Add description of MPP plugins and environment variables, add new test examples
2022-07-26	V1.1.1	LGZ	Add introduction to Gstreamer and dump AFBC decoded data

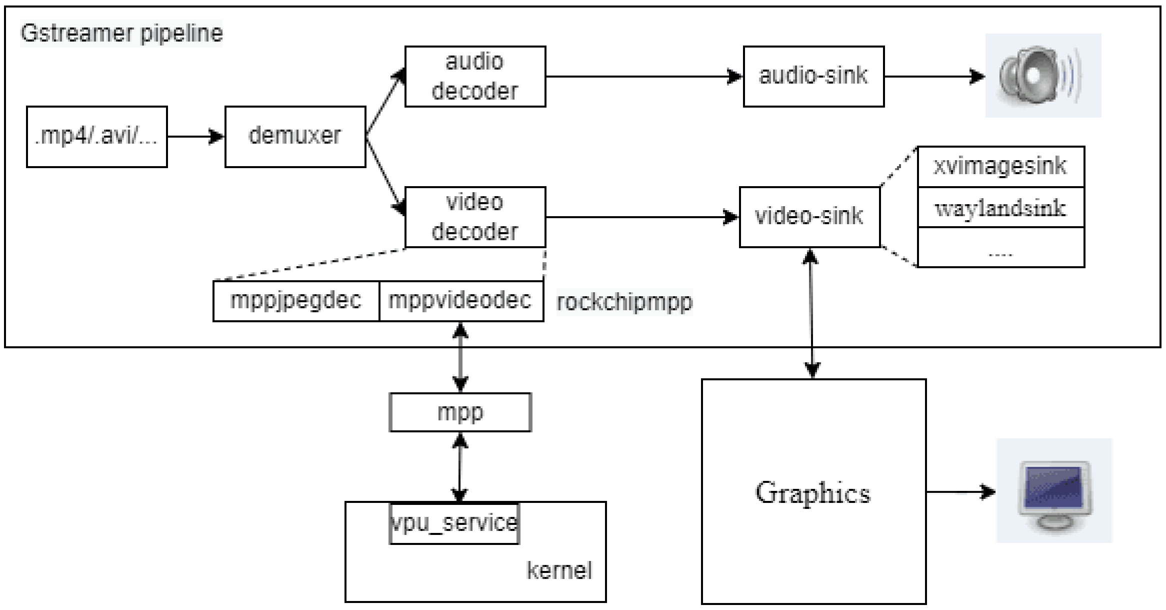
Contents

1. Introduction to Gstreamer

GStreamer is an open source multimedia framework. Currently, the multimedia of all Linux SDK (except IPC) mainly uses GStreamer to connect apps and codec components. Using the powerful features of the GStreamer plug-in, the written GStreamer plug-in is adapted to Rockchip hardware, so that the app can use hardware codec for acceleration.

1.1 GStreamer video codec adaptation scheme

As shown in the figure below, take video playback as an example to illustrate the basic process of video encoding, decoding and display on the Rockchip platform:



The video file (such as mp4) is first decapsulated into video (such as h264, h265 encoding) and audio stream by demuxer, and the video stream is decoded by video decoder (such as mppvideodec and mppjpegdec); the audio stream is decoded by audio decoder, and finally the decoded video data is sent to display through the display plug-in (such as xvimagesink, waylandsink, etc.), and the decoded audio data is sent to the sound card to play the sound through the audio playback plug-in (alsasink, etc.).

The Rockchip platform implements hardware acceleration for video encoding and decoding through the rockchipmpp plug-in, including decoding plug-ins: mppvideodec and mppjpegdec; encoding plug-ins: mpph264enc, mppvp8enc, mppjpegenc, etc. GStreamer will call the rockchipmpp plug-in first in the video decoding stage. The general process is as follows: plug-ins such as mppvideodec call the API provided by MPP, and MPP is the video codec middleware of the Rockchip platform and will call the vpu driver (vpu_service). The hardware codec function can also be tested directly through the test demo provided by MPP (such as mpi_dec_test\mpi_enc_test...).

The decoded video data is sent to the display device through display plug-ins (such as xvimagesink, waylandsink, etc.) for display. Different display plug-ins call different display architecture API to connect with different display architectures. For example, xvimagesink will call the API of X11 to connect to the X11 display architecture, and waylandsink calls the API of Wayland connects to the Wayland display architecture, etc.

Display related specific refer to `SDK Documentation Rockchip_Developer_Guide_Linux_Graphics_EN.pdf`

MPP source code refer to `<SDK>/external/mpp/`，MPP related documentation refer to `<SDK>/docs/Linux/Multimedia/Rockchip_Developer_Guide_MPP_EN.pdf`

test demo refer to: `<SDK>/external/mpp/test`

2. Source Code and Build

2.1 Path of the Source Code

Buildroot:

The source code of Gstreamer and related plug-ins can be downloaded from the network, and then apply the patches provided by RK. For details, please refer to `<SDK>/buildroot/package/gstreamer1/`.

Debian：

Search in Debian Repository to download the source code in Debian，and apply the patches provided by RK, the path of the patches is `<SDK>/buildroot/package/gstreamer1/<submodule>/<version>/*.patch`, now RK only provided two version in 1.18.5 and 1.20.0.

Gstreamer-rockchip:

The source code of the MPP plugin and rkximagesink is in the directory: `<SDK>/external/gstreamer-rockchip`, Buildroot and Debian share the same repository.

2.2 Build

Buildroot:

Enable related macros (which are enabled by default) and build them in the SDK root directory directly. All the macros are packaged in `<SDK>/buildroot/configs/rockchip/*_gst.config`, include them in target configuration. It supported to select building versions, such as `BR2_PACKAGE_GSTREAMER1_18` and `BR2_PACKAGE_GSTREAMER1_20`.

```
BR2_PACKAGE_MPP=y
BR2_PACKAGE_MPP_ALLOCATOR_DRM=y
BR2_PACKAGE_GSTREAMER1_ROCKCHIP=y
BR2_PACKAGE_LINUX_RGA=y
BR2_PACKAGE_CA_CERTIFICATES=y
BR2_PACKAGE_LIBSOUP_SSL=y
BR2_PACKAGE_GSTREAMER1=y
BR2_PACKAGE_GST1_PLUGINS_BASE=y
BR2_PACKAGE_GST1_PLUGINS_BASE_PLUGIN_ALSA=y
BR2_PACKAGE_GST1_PLUGINS_BASE_PLUGIN_VIDEOCONVERT=y
BR2_PACKAGE_GST1_PLUGINS_BASE_PLUGIN_VIDEOTESTSRC=y
BR2_PACKAGE_GST1_PLUGINS_GOOD=y
BR2_PACKAGE_GST1_PLUGINS_GOOD_PLUGIN_AUDIOPARSERS=y
BR2_PACKAGE_GST1_PLUGINS_GOOD_PLUGIN_AUTODETECT=y
BR2_PACKAGE_GST1_PLUGINS_GOOD_PLUGIN_DEINTERLACE=y
BR2_PACKAGE_GST1_PLUGINS_GOOD_PLUGIN_FLV=y
BR2_PACKAGE_GST1_PLUGINS_GOOD_PLUGIN_GDKPIXBUF=y
BR2_PACKAGE_GST1_PLUGINS_GOOD_PLUGIN_MATROSKA=y
```

```
BR2_PACKAGE_GST1_PLUGINS_GOOD_PLUGIN_MPG123=y
BR2_PACKAGE_GST1_PLUGINS_GOOD_PLUGIN_SOUPHTTPSRC=y
BR2_PACKAGE_GST1_PLUGINS_BAD=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_DVBSUBOVERLAY=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_DVDSPU=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_JPEGFORMAT=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_KMS=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_MPEGDEMUX=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_MPEG2ENC=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_VIDEOPARSERS=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_ADPCMDEC=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_ADPCMENC=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_FAAD=y
BR2_PACKAGE_GST1_PLUGINS_UGLY=y
BR2_PACKAGE_GST1_PLUGINS_UGLY_PLUGIN_ASFDEMUX=y
BR2_PACKAGE_GST1_PLUGINS_UGLY_PLUGIN_DVDLPCMDEC=y
BR2_PACKAGE_GST1_PLUGINS_UGLY_PLUGIN_DVDSUB=y
BR2_PACKAGE_GST1_PLUGINS_UGLY_PLUGIN_MPEG2DEC=y
...
```

The complete list of plugins can be found in menuconfig->Target packages->Audio and video applications->gstreamer 1.x.

Debian:

The source code should be placed on the board, and make sure that the `debian` directory exists in the root directory of the source code. Enter the source root directory and execute:

```
# 1 Update software sources
apt update
# 2 Install dependent libraries
apt build-dep .
# 3 Optional: start building the deb installation package
dpkg-buildpackage -b -d -uc -us
# After the building is completed, the deb installation package will be generated
in the upper directory, which can be installed by using dpkg -i xxx.deb.
# 3 Optional: build and install
meson build && ninja -C build install
```

It is generally recommended to use the first way to build the deb installation package, which can ensure that the options such as compilation and installation are unified.

Note: Some compilation options depend on the macro definitions in header files such as `video-format.h`, so you need to install the `libgstreamer-plugins-base1.0-dev` package first to ensure the headers such as `video-format.h` to the latest and ensure that certain features are turned on. If some plugins are missing, check the compilation script and log, install all the dependencies of target plugin, and make sure target plugin is included in debian/*.install, then rebuild.

3. Commonly Used Commands

- gst-launch-1.0

  Gstreamer launcher for quickly building pipelines, examples are as follows:

```
  # Generate a video by videotestsrc, and display it through xvimagesink
  gst-launch-1.0 videotestsrc ! xvimagesink
```

- gst-play-1.0

  Gstreamer player, used to play various streaming media, examples are as follows:

```
  # Play test.mp4 and display it through xvimagesink
  gst-play-1.0 test.mp4 --videosink=xvimagesink
  # Commonly used command options
--flags          # bit0: video, bit1: audio, bit2: subtitle, such as --flags=1
means only video is played
  --videosink    # specify videosink
  --audiosink    # specify audiosink
  --use-playbin3 # use playbin3, otherwise use playbin2
```

- gst-inspect-1.0

  A finder to list all plugins or detailed information of a plugin, for example:

```
  # Without any parameters, list all plugins
  gst-inspect-1.0
  # List all information about the xvimagesink plugin
  gst-inspect-1.0 xvimagesink
```

- Enable log function

```
  #Set environment variables
  export GST_DEBUG=2
  #Or specified before the command, and invalid after the end of the command
  GST_DEBUG=2 gst-play-1.0 ...

  #Specify different log levels for different modules, support wildcards,
fpsdisplaysink is specified as DEBUG (5), xvimage* is specified as FIXME (3),
others are specified as WARNING (2)
  GST_DEBUG=2,fpsdisplaysink:5,xvimage*:3
```

The log levels are divided into ERROR(1), WARNING(2), FIXME(3), INFO(4), DEBUG(5), LOG(6), TRACE(7) and so on.

4. Commonly Used Plugins

4.1 Source

Refers to plugins that can generate data but cannot receive data.

- filesrc

  Read data from a file, an example is as follows:

```
  gst-launch-1.0 filesrc location=/tmp/test ! filesink location=/tmp/test2
```

- videotestsrc

  Generate video data, an example is as follows:

```
  # Output video through default format
  gst-launch-1.0 videotestsrc ! xvimagesink
  # Output the video through the specified format
  gst-launch-1.0 videotestsrc ! "video/x-raw,width=1920,height=1080,format=
(string)NV12" !xvimagesink
```

- v4l2src

  Capture from camera, an example is as follows:

```
gst-launch-1.0 v4l2src ! video/x-raw,width=1920,height=1080,format=NV12 !
waylandsink
```

- rtspsrc

  Get stream from RTSP server, an exmaple is as follows:

```
gst-launch-1.0 rtspsrc location=rtsp://192.168.1.105:8554/ ! rtph264depay !
h264parse ! mppvideodec ! waylandsink
```

4.2 Sink

Refers to plugins that accept data but do not send data.

- filesink

  Save the received data as a file, an example is as follows:

```
  gst-launch-1.0 filesrc location=/tmp/test ! filesink location=/tmp/test2
```

- fakesink

  Discard all the received data, an example is as follows:

```
  gst-launch-1.0 filesrc location=/tmp/test ! fakesink
```

- xvimagesink

  Video Sink, receive video and display, which is implemented by the X11 interface, an example is as follows:

```
  gst-launch-1.0 videotestsrc ! xvimagesink
```

- kmssink

  Video Sink, receives video and displays it. It is implemented through the kms interface and requires exclusive hardware decoding layer. The example is as follows:

```
gst-launch-1.0 videotestsrc ! kmssink
# Common commands
connector-id        #specifies the screen
plane-id            #pecifies the hardware layer
render-rectangle    #specifies the rendering range
```

- waylandsink

  Video Sink, receives video and displays, it is implemented through the wayland interface, the example is as follows:

```
gst-launch-1.0 videotestsrc ! waylandsink
```

- rkximagesink

  Video Sink, receives video and displays it, zero-copy and other functions are implemented through the drm interface, and with better performance, but requires exclusive hard decoding layer. An example is as follows:

```
gst-launch-1.0 videotestsrc ! rkximagesink
```

- fpsdisplaysink

  Video Sink, receives the video and counts the frame rate, and at the same time transfers the video to the next level Sink for display, an example is as follows:

```
  # Set log level to TRACE(7) for real-time framerate, set log level to DEBUG(5)
for max/min framerate.
  GST_DEBUG=fpsdisplaysink:7 gst-play-1.0 --flags=3 --videosink="fpsdisplaysink
video-sink=xvimagesink signal-fps-measurements=true text-overlay=false
sync=false"
```

5. Rockchip MPP plugins

The decode/encode plugins is based on MPP, the base class of decode plugin is GstVideoDecoder class, the base class of encode plugin is GstVideoEncoder class. The path of source code is `<SDK>/external/gstreamer-rockchip/gst/rockchipmpp`.

The formats in support for decoder are JPEG, MPEG, VP8, VP9, H264, H265 [1].

The formats in support for encoder are JPEG, H264, H265, VP8.

5.1 gstmppdec

The path of source code is gstreamer-rockchip/gst/rockchipmpp, include mppvideodec, mppjpegdec, the following will take mppvideodec as an example for description.

```
gstreamer-rockchip/gst/rockchipmpp/
├── gstmppdec.c
├── gstmppdec.h
├── gstmppjpegdec.c
├── gstmppjpegdec.h
├── gstmppvideodec.c
├── gstmppvideodec.h
……
```

5.1.1 Description of Major Functions

gst_mpp_dec_start: Create MPP and Allocator。

gst_mpp_dec_set_format: Init the MPP, setup codec type and format, configure the properties such as Fast Mode, Ignore Error.

gst_mpp_dec_handle_frame: Get the mpp packet from MPP by get_mpp_packet, send to MPP by send_mpp_packet after filling all the data.

gst_mpp_dec_loop: Get the decoded frame by poll_mpp_frame, and push to next plugin.

gst_mpp_dec_rga_convert: If customers need to do some operation such as format convert, rotate, scale, crop, push to next plugin after all operations are completed by RGA [2].

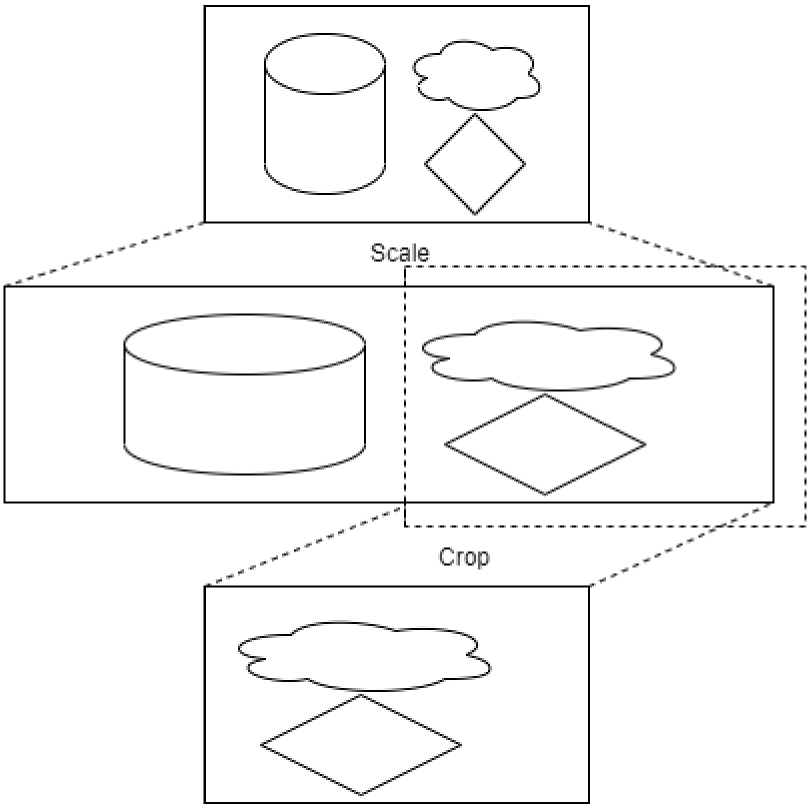
5.1.2 Description of Major Properties

rotation: Angle of rotation, 0°, 90°, 180°, 270° are avaliable.

width: Zero for no scaling.

height: Zero for no scaling.

crop-rectangle: Specified the crop range by <x, y, w, h>, which means start from the <x, y>, cropping a w*h image to next plugin. It should be noted that scaling has a higher priority than cropping, so cropping parameters should be calculated based on the scaled width and height, as shown in the figure for the processing logic when specifying `crop-rectangle=‘<1920,0,1920,1080>' width=3840 height=1080`:



arm-afbc: ARM Frame Buffer Compression, disabled by default, some platform such as RK3399 do not support AFBC. Enable it the DDR bandwidth occupation can be reduced, and the decoding efficiency of some chips will be significantly improved.

format: Output format. If it is not 0-"auto", the format will be converted.

fast-mode: Enable MPP fast mode. For example, on the RK3588 platform, part of the decoding process can be parallelized to improve decoding efficiency. Enbaled by default.

ignore-error: Ignore error of MPP decoder, force output the decoded frame. Enabled by default.

5.2 gstmppenc

The path of source code is gstreamer-rockchip/gst/rockchipmpp, include mpph264enc, mppvp8enc, mppjpegenc, etc. The following will take mpph264enc as an example for description.

```
gstreamer-rockchip/gst/rockchipmpp/
├── gstmppenc.c
├── gstmppenc.h
├── gstmppjpegenc.c
├── gstmppjpegenc.h
├── gstmpph264enc.c
├── gstmpph264enc.h
……
```

5.2.1 Description of Major Functions

gst_mpp_enc_start: Create MPP, setup codec type and format。

gst_mpp_enc_apply_properties: Configure the properties such as gop, bps.

gst_mpp_enc_handle_frame: Get the buffer from last plugin and store it.

gst_mpp_rga_convert: If any operation such as rotate, scale is in need, will complete it via RGA [3] before storing the buffer.

gst_mpp_enc_loop: Get the oldest frame in queue, send it to MPP by encode_put_frame, and get packet back by encode_get_packet, then push the packet to next plugin.

5.2.2 Description of Major Properties

width: Zero for no scaling.

height: Zero for no scaling.

rc-mode: Bit rate control mode, support VBR, CBR and Fixed QP.

bps: Target bit rate, ignored in Fixed QP mode.

bps-max: Max bit rate, ignored in Fixed QP mode.

bps-min: Min bit rate, ignored in Fixed QP mode.

gop: Group Of Picture, the interval of two I frames. 0 indicates that there is only one I frame, other frames are P frames, 1 means all I frames, 2 means the sequence is I P I P I P... . Gop is equal to framerate by default.

level: Indicates the level_idc parameter in SPS.

profile: Indicates the profile_idc parameter in SPS.

rotation: Angle of rotation, 0°, 90°, 180°, 270° are avaliable.

6. Environment Variables

Common environment variables are sorted into /etc/profile.d/gst.sh. For detailed instructions, you can directly view the comments in the script.

```
export GST_MPP_VIDEODEC_DEFAULT_ARM_AFBC=1: Try to use ARM AFBC to get better performance, but not work for all sinks.
export GST_MPP_VIDEODEC_DEFAULT_FORMAT=NV12: Convert to NV12(using RGA) when output format is not NV12.
export GST_V4L2_PREFERRED_FOURCC=NV12:YU12:NV16:YUY2: Preferred formats for V4L2.
export GST_VIDEO_CONVERT_PREFERRED_FORMAT=NV12:NV16:I420:YUY2: Preferred formats for videoconvert.
export GST_VIDEO_CONVERT_USE_RGA=1: Try RGA 2D accel in videoconvert and videoscale.
export GST_VIDEO_FLIP_USE_RGA=1: Try RGA 2D accel in videoflip.
export GST_MPP_DEC_DEFAULT_IGNORE_ERROR=0: Disable ignoring MPP error.
export GST_MPP_DEC_DEFAULT_FAST_MODE=0: Disbale fast mode。
...
```

7. Command Examples

7.1 Video Playback

```
gst-play-1.0 --flags=3 --videosink="fpsdisplaysink video-sink=xvimagesink signal-fps-measurements=true text-overlay=false sync=false" --audiosink="alsasink device=hw:0,0" test.mp4
```

7.2 Multiple Video Playback

```
# Use the render-rectangle of waylandsink for different rendering positions
gst-launch-1.0 filesrc location=/usr/local/test.mp4 ! parsebin ! mppvideodec ! waylandsink render-rectangle='<0,0,400,400>' &
gst-launch-1.0 filesrc location=/usr/local/test.mp4 ! parsebin ! mppvideodec ! waylandsink render-rectangle='<0,500,400,400>' &
gst-launch-1.0 filesrc location=/usr/local/test.mp4 ! parsebin ! mppvideodec ! waylandsink render-rectangle='<0,1000,400,400>' &
```

7.3 Encode and Preview

Use tee plugin, copy the data of camera capture, the first way send to mpph264enc for encoding, and then save to file by filesink, the second way send to autovideosink for display rendering. It shoud be noted that add the queue plugin after the tee plugin, which can buffering the data, avoid stream blocking.

```
gst-launch-1.0 v4l2src ! 'video/x-raw,format=NV12' ! tee name=tv ! queue ! mpph264enc ! 'video/x-h264' ! h264parse ! 'video/x-h264' ! filesink location=/data/out.h264 tv. ! queue ! autovideosink
```

7.4 Split Stream

Some plugins such as qtdemux, will generate not only one source pad, such as audio pads, video pads, subtitle pads. You can named the plugin, and then get the target stream. As the following example, named the qtdemux to qt, then qt.audio_0 is the first audio stream, qt.video_0 is the first video stream, save these two streams to different files. And the queue plugin is needed too. The different plugins have different name style for their pads, you can check it via gst-inspect command, or directly use like `qt. ! queue ! mppvideodec` in your pipeline, the gstreamer framework will negotiate the caps with next plugin.

```
gst-launch-1.0 filesrc location=test.mp4 ! qtdemux name=qt qt.audio_0 ! queue ! filesink location=audio.bin qt.video_0 ! queue ! filesink location=video.bin
```

8. AFBC

AFBC stands for ARM Frame Buffer Compression, which is a compression format used to save bandwidth. Currently, the encoding formats of AFBC supported by the mppvideodec plugin are: H264, H265, VP9, and the supported color formats are NV12, NV12 10bit, NV16. The way to open is as follows:

```
# Enable global AFBC, applicable to situations where mppvideodec cannot be
directly operated using some command like gst-play-1.0
export GST_MPP_VIDEODEC_DEFAULT_ARM_AFBC=1
# Enable afbc for current pipeline
gst-launch-1.0 filesrc location=/test.mp4 ! parsebin ! mppvideodec arm-afbc=true
! waylandsink
```

The waylandsink and xvimagesink support rendering AFBC format, or using kmssink/rkximagesink specufied Cluster plane for display, this method requires an exclusive plane. The examples are as follows：

```
# GST_DEBUG=*mpp*:4 enable the log of mpp plugin, you can use the log to check if
the AFBC is enabled successfully, if "AFBC" is not printed, it may be
unsuccessfully opened or the format does not in support
GST_DEBUG=*mpp*:4 gst-play-1.0 --flags=3 --videosink=waylandsink test.mp4
GST_DEBUG=*mpp*:4 gst-play-1.0 --flags=3 --videosink="kmssink plane-id=101"
...
0:00:00.256819945 29143   0x7f70008700 INFO                  mppdec
gstmppdec.c:465:gst_mpp_dec_apply_info_change:<mppvideodec0> applying NV12(AFBC)
1920x1080 (1920x1104)
...
```

8.1 AFBC dump the decoded data

If you wants to check whether the hardware decoded data is correct in GStreamer, can dump the decoded data (usually NV12 and other format images) in the following way:

1. Turn on the MPP log function

```
export mpp_debug=0x400
```

Then there will be decoded data of MPP's own dump in the /data directory. This is the dump debugging function that comes with MPP. Dump is not supported when AFBC is enabled; when AFBC is not enabled, the dumped data is generally in NV12 format, which can be viewed using rawplayer (or other raw data players).

2. Use GStreamer's plugin filesink dump to the decoded data. This method supports dump regardless of whether AFBC is turned on or not. The usage is as follows:

```
gst-launch-1.0 uridecodebin uri=file://xxx ! filesink location=xxx.yuv
```

The decoded AFBC data is in the xxx.yuv file. Because AFBC is turned on, the dumped image needs to be decompressed before it can be viewed using rawplayer (or other raw data players). The decompression command (the decompression software afbcDec should be obtained from the relevant person in charge):

```
./afbcDec filename w h format afbcmode
#eg: ./afbcDec 178_Surfa_id-26_1088x1824_z-0.bin 1088 1824 0 1
# 0=RGBA,1=NV12,2=RGB888, afbcmode 0=afbc, 1=afbc|YTR
```

The image format output by afbcDec is ARGB.

If you want to check whether each frame of the video is correct, you also need to divide the file dumped by filesink into frames: because the above decompression software can only transfer one frame of data, but all the frames of the dumped video are in the same file. An example of spliting the frames is as follows:

```
# GST_DEBUG view the size of each frame
GST_DEBUG=filesink:6 gst-launch-1.0 uridecodebin uri=file://xxx ! filesink
location=xxx.yuv

0:00:01.224149631 14266   0x7f7c00ab00 DEBUG  filesink
gstfilesink.c:769:gst_file_sink_flush_buffer:<filesink0> Flushing out buffer of
size 1390080
# Use the split command to split frames
split -b 1390080 -a 5 -d  xxx.yuv dump_frame
```

9. Subtitle

When subtitles are turned on, there will be lags. Usually, subtitle synthesis requires intercept some images from the video and convert them to RGB, and then synthesize subtitles and then convert them back to the source format before sending them for display. That is, the time-consuming of decoding also needs to consider the time-consuming of subtitle synthesis. , causing the overall frame rate to drop. Use the gst-play-1.0 command to test and subtitles can be turned off with `--flags=3`. Subtitles should be implemented independently of the video layer using frameworks such as QT.

10. Layers Assignment

When using rkximagesink or kmssink, it is required to have a exclusive hardware layer, and the plug-in will automatically find the layer to play, but the automatically found layer may not meet the requirements, so you have to manually specify the layer, the way is as follows:

```
gst-play-1.0 --flags=3 test.mp4 --videosink="kmssink plane-id=117"
```

The 117 is the ID of the target layer, which can be confirmed through the `/sys/kernel/debug/dri/0/state` node. You can use the following command to list all layers:

```
root@linaro-alip:/# cat /sys/kernel/debug/dri/0/state | grep "plane\["
plane[57]: Smart1-win0
plane[71]: Cluster1-win0
plane[87]: Smart0-win0
plane[101]: Cluster0-win0
plane[117]: Esmart1-win0
plane[131]: Esmart0-win0
# You can also use cat /sys/kernel/debug/dri/0/state directly to list complete
information
```

The plane[xx] is the plane-id. Usually, different layers support different formats. For example, Cluster supports AFBC, but Esmart does not support AFBC. Please refer to the datasheet or TRM for details. If the node is not exist, you can list it with modetest -p.

11. FAQ

1. There is no lagging when playing 4K 30FPS, but there is lagging when playing 4K 60FPS.

   Due to system load, DDR bandwidth and other issues, 4K 60FPS may not be achieved. You can try to enable AFBC, refer to the AFBC chapter. In addition, the synchronization function of subtitles and sink can be turned off, such as `gst-play-1.0 test.mp4 --flags=3 --videosink="waylandsink sync=false"`, when the frame rate cannot reach 60FPS, turning on sync will cause the video frame timestamps do not align with clocks resulting in obvious frame drop.

2. There is relatively lagging when playing some sources, and the CPU usage is very high.

   Currently hard decode supports H264, H265, VP8, VP9, MPEG. You can turn on DEBUG through `echo 0x100 > /sys/module/rk_vcodec/parameters/mpp_dev_debug` to see if the serial port or dmesg has decoded printing. If not, it may be a format not supported by the hard decode.

3. Some sources cannot be played, LOG is lagging and the progress is not printed or the progress is always 0

   You can try to use playbin3, like `gst-play-1.0 --flags=3 --use-playbin3 test.mp4`.

4. Flickering when playing 4K video after AFBC is turned on

   First make sure to turn on performance mode, `echo performance | tee $(find /sys/ -name *governor)`. Then, confirm whether there is obvious scaling in the vertical direction, such as using the vertical screen to play the horizontal video, in this case, the AFBC performance is not as good as the non-AFBC performance.

5. Play with pictures but no sound

   You can manually specify the audiosink, such as `gst-play-1.0 --flags=3 test.mp4 --audiosink="alsasink device=hw:0,0"`. It is recommended to make sure it can work using basic testing tools such as aplay and then use gstreamer to test.

6. When running the decompression command afbcDec, an error is reported: the library libgraphic_lsf.so is missing

   Find the relevant person in charge to obtain libgraphic_lsf.so, and push the missing libgraphic_lsf.so library to the /usr/lib/ directory.

1. Only the formats supported by the plug-in are listed here. Please check the relevant datasheet if the specific chip supports it. ↩
2. At present, some platforms such as RK3588, RGA function is abnormal, so it is not recommended to use it. ↩
3. At present, some platforms such as RK3588, RGA function is abnormal, so it is not recommended to use it. ↩