FFplay –
RTK Hardware Transcode Tutorial

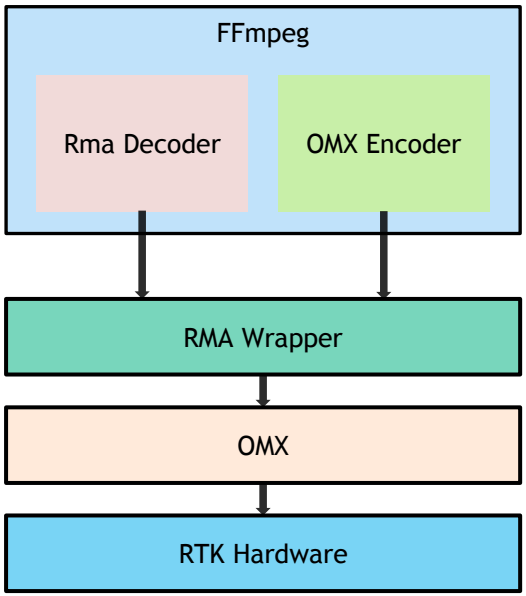
Driving IC Innovation

Agenda

- Introduction
- Build FFmpeg/FFplay
- Realtek Patch of Ffplay
- ALSA compress API
- Playback command line

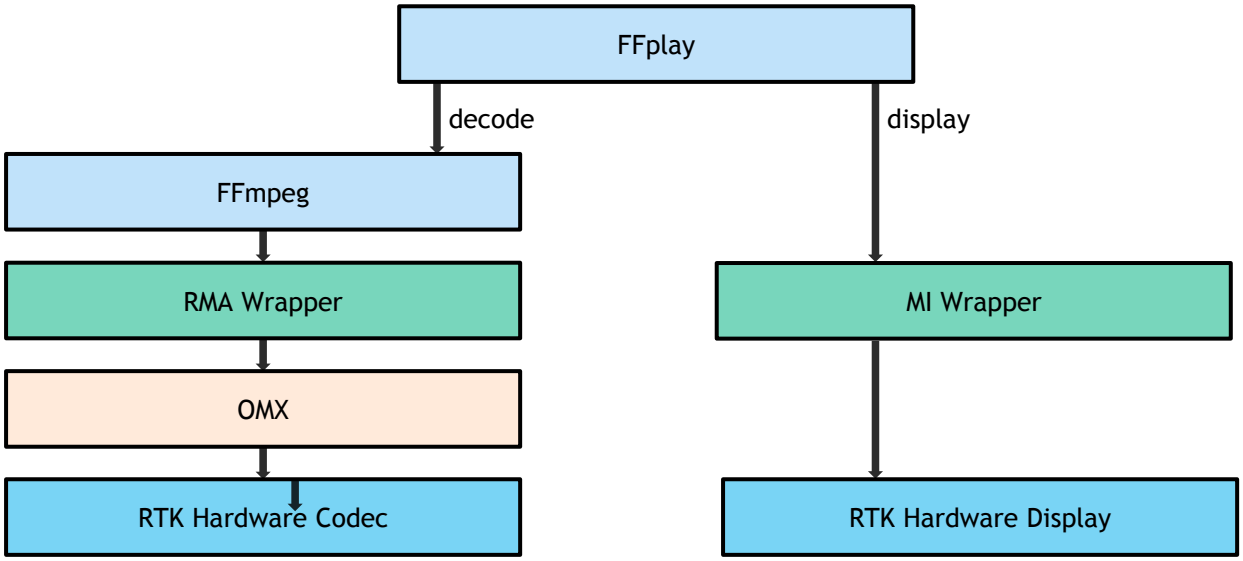
Introduction - FFmpeg

- Realtek uses RMA (RTK Media Accelerator) as wrapper layer between FFmpeg and Openmax Integration Layer
  - Decoder: libavcodec/rma_dec.c
  - Encoder: libavcodec/omx.c
  - Wrapper library: libRMA.so



Introduction - FFplay

- Realtek uses RTK MI (RTK Media Interface) as wrapper layer between FFplay and RTK Hardware Display
  - Wrapper library: libRTKMediaIf.so



Build FFmpeg/FFplay

- Build SDL2 (2.0.4) for FFplay
  - http://www.libsdl.org/release/SDL2-2.0.4.tar.gz
  - ./configure;make;make install
- Get FFmpeg 3.3.7 source code.
  - https://ffmpeg.org/releases/ffmpeg-3.3.7.tar.xz
- Patch RTK FFmpeg patch. Run
  - patch –p0 < 0001-FFmpeg3.3.7-RTK-HW-ACCEL.patch
- Copy RTK-NAS-Transcode-Libs to target. Run
  - cp -dR RTK-NAS-Transcode-Libs/usr /usr

Build FFmpeg/FFplay (cont.)

- Configure FFmpeg. You can reference build.sh to enable RMA(RTK Media Accelerator). Run
  - ./configure --enable-omx --enable-rma --enable-decoder=h264_rma --enable-decoder=mpeg4_rma --enable-decoder=hevc_rma --enable-decoder=mpeg1_rma --enable-decoder=mpeg2_rma --enable-decoder=vp8_rma --enable-decoder=vp9_rma --enable-decoder=vc1_rma --enable-decoder=wmv3_rma --enable-decoder=mjpeg_rma --enable-decoder=h263_rma --enable-decoder=avs_rma --enable-decoder=flv_rma --enable-encoder=h264_omx
- Build and install FFmpeg. Run
  - make ; make install

Realtek Patch of FFplay

- We use the define : REALTEK_PATCH to distinguish between official release and RTK patch.
- CONFIG_AVFILTER must be disabled. RTK Hardware can't display the buffers that have processed by avfilter.
- In order to display video by RTK Hardware, the SDL must be made:
  - Create a borderless and zero width/height video window
  - Disable SDL render functions: SDL_UpdateYUVTexture/ SDL_RenderCopyEx
- Passing renderFlg to avcodec_open2() to inform decoder uses render mode

```
int renderFlg = 1;
avctx->opaque = &renderFlg;
if ((ret = avcodec_open2(avctx, codec, &opts)) < 0) {
        goto fail;
}
```

Realtek Patch of FFplay (cont.)

- Use RTK MI to control RTK Hardware
  - rtk_init() : Load libRTKMediaIf.so and initialize RTK MI.
  - rtk_deinit() : Destroy MI
  - rtk_direct_render() : Render frames

ALSA Compress API

- To use ALSA compress APIs, FFplay can utilize tinycompress, a userspace library that provides the APIs to open a ALSA compressed device and read/write compressed data like AC3 etc. to it
- Install tinycompress
  - Get sources from alsa-project.org
  - Or from http://git.alsa-project.org/?p=tinycompress.git
- FFplay needs codec types and parameters for compressed data streaming interface
  - Copy from kernel's include/uapi/sound/compress_params.h

ALSA Compress: Data Structure

- Include necessary header files and compress structure

```
#include "compress_params.h"
#include "tinycompress/tinycompress.h"
struct compress *compress = NULL;
```

- Define variables for compressed mode
  - audio_compr is used as option
  - audcompr_finished is used to determine write completion

```
static int audio_compr = 0;
static int audcompr_finished = 0;
```

- Add -audio_compr option

```
static const OptionDef options[] = {
#include "cmdutils_common_opts.h"
    { "x", HAS_ARG, { .func_arg = opt_width }, "force displayed width", "width" },
    ...
    { "acompr", OPT_BOOL, {&audio_compr}, "compressed audio mode" },
    { NULL, },
};
```

ALSA Compress: Device Open

- FFplay opens audio or video stream in stream_component_open()
  - Need to open compressed device

```
static int stream_component_open(VideoState *is, int stream_index)
{
    switch (avctx->codec_type) {
    case AVMEDIA_TYPE_AUDIO:
        /* prepare audio output */
        if (audio_compr) {
            if ((ret = rtk_compress_open(avctx)) < 0)
                goto fail;
        }
        …
}
```

- To open compressed device, for example, in rtk_compress_open
  - Set codec structure
  - Call compress_open() to open a compressed device, e.g., /dev/snd/comprC0D0

ALSA Compress: Compress Write

- FFplay's audio_thread will decodes compressed data in decoder_decode_frame()
  - Need to write compressed data to the compressed device instead of decoding frames

```
static int decoder_decode_frame(...) {
        switch (d->avctx->codec_type) {
            case AVMEDIA_TYPE_AUDIO:
                if (audio_compr) {
                    got_frame = rtk_compr_write(&d->pkt_temp);
                    if (!got_frame) {
                        av_usleep(1000000);
                        audcompr_finished = 1;
                    }
                }
```

- To write compressed data, for example, in rtk_compr_write()
  - Call compress_start() if not running
  - Call compress_write() to write the data

ALSA Compress: Device Close

- FFplay calls do_exit() upon playback completion
- To close compressed device
  - Call compress_stop() and compress_close()

Playback command line

- Enable weston

```
weston --tty=1 &
```

- Run FFplay command

```
SDL_VIDEODRIVER=dummy SDL_RENDER_DRIVER=software ffplay –autoexit /VIDEO/FILE/PATH
```

- FFplay options for RTK
  - “res” : Setting HDMITx resolution

    [0] = 1080p@60, [1] = 1080p@50, [2] = 4k@60, [3] = 4k@50,

    [4] = 4k@30, [5] = 4k@25, [6] = 720p@60, [7] = 720p@50

```
SDL_VIDEODRIVER=dummy SDL_RENDER_DRIVER=software ffplay –autoexit –res 5 /VIDEO/FILE/PATH
```

  - “rtk_mi_version” : Show verion information about RTK mi

```
ffplay –autoexit –rtk_mi_version 1 /VIDEO/FILE/PATH
```