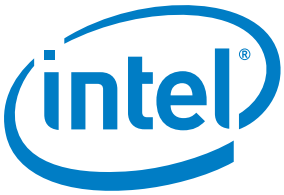


Enabling Hardware Accelerated VC-1 AP Interlace on Intel® Atom™ Processor E38XX Series

Application Note

May 2014

Document Number: 330630-001US

Contents

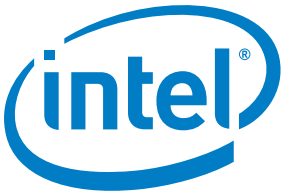
Figures

Tables

Revision History

Date	Revision	Description
May 2014	001	Initial release.

§



1 Introduction

VC-1 is a video codec standard released and maintained by the Society of Motion Picture and Television Engineers (SMPTE). The standard is also known as SMPTE 421M. VC-1 supports coding for progressive video content in Simple, Main, and Advanced Profiles, and interlaced video content in Advanced Profile only. VC-1 interlaced coding is especially attractive to the broadcast industry due to the bandwidth reduction via interlaced coding.

A hardware-accelerated media solution with EMGD for the Intel® Atom™ Processor E38XX Series is inherited from the Intel Open Source Technology Center (OTC) VA API stack. Even though general VC-1 SP/MP/AP decoding is supported in the original VA API stack, VC-1 support is not commonly used or tested due to the relative insignificance of the VC-1 codec in the Linux* space. VC-1 is most commonly found in codecs developed by Microsoft: WMV3, WMVA, WVC1A. However, some applications do require complete, hardware-accelerated, VC-1 decoding support, and this prompted Intel to develop a solution on top of the VA API stacks that decodes VC-1 AP Interlaced coded streams.

The objective of this solution is to ensure that the decoding of the VC-1 AP Interlaced coded stream is hardware-accelerated with the Intel® Atom™ Processor E38XX Series Multi-Format Decoding (MFD) engine. This paper identifies gaps in the current stack and provides an implementation for a solution on an Intel® Atom™ Processor E38XX Series system.

The Intel® Embedded Design Center provides qualified developers with web-based access to technical resources. Access Intel Confidential design materials, step-by-step guidance, application reference solutions, training, Intel's tool loaner program, and connect with an e-help desk and the embedded community. Design Fast. Design Smart. Get started today. www.intel.com/embedded/edc.

Table 1. Terminology

Term	Description
AP	Advanced Profile
API	Application Program Interface
EMGD	Intel Embedded Media and Graphics Driver
FCM	Frame Coding Mode
IOTG	Internet of Things Solution Group
MB	Macroblocks
MFD	Multi-Format Decoding
OTC	Intel Open Source Technology Center
SMPTE	Society of Motion Picture and Television Engineers
VA	Video Acceleration
WMV	Windows Media Video

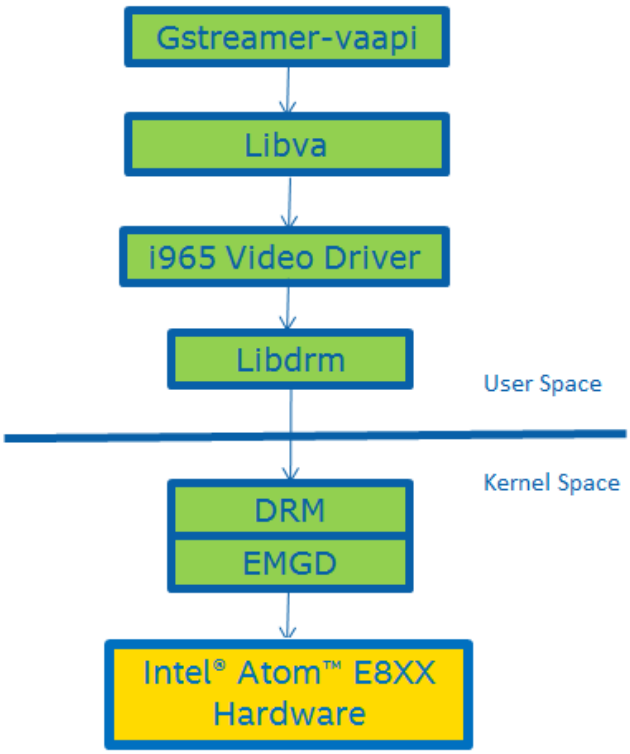
§

2 Background

EMGD for the Intel® Atom™ Processor E38XX Series uses the Intel Open Source Technology Center's VA API media solution for hardware-accelerated video decoding with different codec profiles such as H.264, MPEG-2, and VC-1. Even though VC-1 decoding is generally supported in the VA API stacks, decoding of VC-1 AP Interlaced coded streams is not supported in the current solution's stack. One convenient alternative is to switch to a software-accelerated solution that uses the system CPU to decode the decoded streams; however, this is not optimal in terms of performance and reducing CPU utilization of the system.

Figure 1 provides an architectural view of the VA API stacks to help illustrate what must be implemented in the OTC's VA API stack to decode VA-1 AP Interlaced coded streams.

Figure 1. VA API Software Architectural View



There are two areas to address when decoding the VC-1 AP Interlaced coded streams.

- The first area is the Gstreamer-vaapi component.

  Additional parsing logic is required so that the bitstream of a VC-1 AP Interlaced coded stream is parsed correctly according to the SMPTE standard.

- The second area is the i965 video driver.

  To get the compressed VC-1 AP Interlaced data to decode properly, a command buffer must be sent to the MFD hardware.

Based on these requirements, additional logic must be implemented in the gstreamer-vaapi decoder plugin and the i965 video driver to get the VA API stack to decode a VC-1 AP Interlace video. In this solution, the other components shown in Figure 1 do not require modification.

§

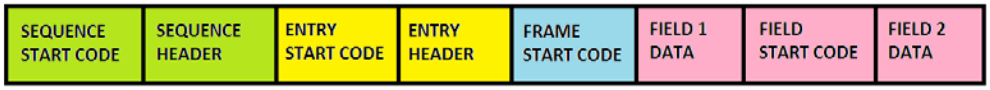
3 Solution

To implement the additional logic discussed in the Background section, the fundamental hierarchical layers that construct a VC-1 encoded stream, whether it is progressive or interlace video, must be understood. In general, a VC-1 AP stream is composed of the following hierarchical layers.

- Sequence
- Entry-point
- Picture
- Slices
- Macroblocks (MB)
- Blocks

Figure 2 illustrates the presence of some of these layers in the bitstream.

Figure 2. Hierarchical Layers of VC-1 Bitstream



Each layer can be detected by identifying a uniquely defined code (called “start code”) from the encoded bitstream. The start code is followed by a header section, which can contain some useful bits of information relevant to the solution. For example, the sequence header section contains a field called “Interlace” that must be parsed correctly by the codec for the decoder to determine if the stream is a progressive or an interlace video.

There are essentially two types of VC-1 interlace coding: Interlace frame and interlace field. Interlace frame coding is used when both fields of an interlace frame are coded together. Interlace field coding is used when the two fields of an interlace frame are coded separately. The difference between these two coding modes leads to different methods of decoding the stream. In order to identify the coding mode adopted by an interlace picture, the codec parser can look at the field called “Frame Coding Mode (FCM)” found in the picture header.

Table 2. Frame Coding Mode

Field Value	Frame Coding Mode
0b	Progressive
10b	Interlace Frame
11b	Interlace Field

Since the two fields of data for interlace field coding are coded separately, one of these two fields is meant to be displayed first. That field is called the first field, and it can be the top field or the bottom field of the frame. This leads to the necessity for the codec parser to extract the field called “Top Field First” found in the picture header. A value of 0 for TFF indicates that the first field in the frame is a bottom field and the second field in the frame is a top field. Alternately, a value of 1 for TFF indicates that the first field in the frame is a top field and the second field in the frame is a bottom field.

In an interlace field picture, each of the top and bottom fields can be decoded into different picture types such as I, P, B, or BI. From the picture header, a field called “Field Picture Type (FPTYPE)” can be extracted to identify the field picture type for each of the two fields. FPTYPE can be decoded by the codec parser according to Table 3.

Table 3. Field Picture Type

FPTYPE	First Field	Second Field
000b	I	I
001b	I	P
010b	P	I
011b	P	P
100b	B	B
101b	B	BI
110b	BI	B
111b	BI	BI

For the i965 video driver, a few fixes must be implemented so that the interlace frame and the interlace field coding are processed properly. First, the FCM value given by the codec needs to be extracted from the frame_coding_mode field defined in VAPictureParameterBufferVC1 structure. With the FCM value read, the driver can now, in the case of interlace field coding, determine the proper picture type for the first and second field. Consequently, the driver is also able to determine the correct picture height in macroblocks.

Additionally, the proper way to store the reference frame data for interlace field coding is missing in the i965 video driver. This has also been implemented in the patch provided.

§

4 Applying the Solution

As described in the Solution section, a couple of patch files have been created to enable the decoding of VC-1 AP Interlaced coded stream with Intel OTC's VA API stack. After applying the patches, both the VC-1 frame interlace and the field interlace should be decoded properly.

The following sections provide detailed instructions for patching the Gstreamer-vaapi and OTC's driver. Generally, these instructions should work as described if the specified software versions are installed on the target system. For different software versions, additional patching efforts may be required and the instructions should be used as a rough guideline.

4.1 Prerequisites

1. Development tools package
2. Git
3. Gstreamer base framework 1.0.7 or beyond
4. Gstreamer ugly plugin for asfdemux plugin
5. Patch tool

4.2 Patching the Gstreamer-vaapi Plugin

1. Clone the plugin by entering the following at a terminal prompt.

   ```
   $ git clone git://gitorious.org/vaapi/gstreamer-vaapi.git
   ```

2. Checkout the specific commit; this commit is used during the testing and enabling.

   ```
   $ git checkout 2e356b0f7efae33fb943ad11204020dcdbf1b04f
   ```

3. Go into the `gstreamer-vaapi` directory.

   ```
   $ cd gstreamer-vaapi
   ```

4. Retrieve the patch file Gst_Plugins_VAAPI_pre_0_5_8_VC1_Interlace_Patch and place it in the current directory.

   The patch file is located in the driver release package in patches\common\VA_Driver_i965\VC1_Interlace, which is in `IEMGD_HEAD_Linux.tgz`.

5. Patch the Gstreamer-vaapi project.

   ```
   $ patch –p1 < Gst_Plugins_VAAPI_pre_0_5_8_VC1_Interlace_Patch
   ```

6. Run the autogen to initialize the submodules in the project.

   ```
   $ ./autogen.sh
   ```

7. Go into the codec-parser directory.

   $ cd ext/codecparsers

8. Retrieve the patch file Gst_Codecparser_VAAPI_pre_0_5_8_VC1_Interlace_Patch from the path identified in Step 4 and place it in the current directory.

9. Patch the codec-parsers project.

   ```
   $ patch –p1 <
   Gst_Codecparser_VAAPI_pre_0_5_8_VC1_Interlace_Patch
   ```

10. Compile the Gstreamer-vaapi project.

    ```
    $ make
    $ make install
    ```

4.3 Patching OTC's intel-driver

1. Clone the driver.

   ```
   $ git clone git://anongit.freedesktop.org/vaapi/intel-driver
   ```

2. Checkout the specific version; this version is used during testing and enabling.

   ```
   $ git checkout 1.2.1
   ```

3. Go into the intel-driver directory.

   ```
   $ cd intel-driver
   ```

4. Retrieve the patch file Intel_VA_Driver_1_2_1_GEN7_VC1_Interlace_Patch from the path previously mentioned, and place it in the current directory.

5. Patch the intel-driver project.

   ```
   $ patch -p1 < Intel_VA_Driver_1_2_1_GEN7_VC1_Interlace_Patch
   ```

6. Compile the intel-driver project.

   ```
   $ ./autogen.sh
   $ make
   $ make install
   ```

4.4 Decoding a VC-1 AP Interlaced Stream

With the application of the patches, the VC-1 AP Interlaced coded stream can be decoded successfully. The gst-launch tool can be used to demonstrate the decoding capability of the VC-1 interlace. Using a WMV clip, the following gstreamer pipeline can be used.

```
$ gst-launch-1.0 -v filesrc location=<video clip path plus
filename> ! asfdemux ! vaapidecode ! vaapisink
```

§

5 Results and Test Coverage

Intel tested VC-1 AP interlace by decoding a list of VC-1 Field and Frame interlace clips in WMV contained format. In addition, tests have been carried out with the following conformance test bitstreams provided by SMPTE:

SA10210, SA10211, SA10212, SA10213, SA10214, SA10215, SA10216, SA10217

§

6 Conclusion

This paper provides guidelines and solutions for decoding VC-1 Interlace coded streams on the Intel® Atom™ Processor E38XX Series with EMGD Linux* driver.

The Intel® Embedded Design Center provides qualified developers with web-based access to technical resources. Access Intel Confidential design materials, step-by step guidance, application reference solutions, training, Intel's tool loaner program, and connect with an e-help desk and the embedded community. Design Fast. Design Smart. Get started today.

§