Enabling Hardware Accelerated Playback for Intel® Atom™/Intel® US15W Platform and IEGD

Case Study Using MPlayer on Moblin

March, 2010
Executive Summary

Media playback on the Intel® Atom™ platform is optimal when the video decoding is handled by the video engine instead of the CPU. Using a video engine can reduce the CPU workload by more than 50%.

Using a video engine to decode video reduces CPU workload by more than 50%. It requires the application, such as a media player, to use VA API to communicate with Intel® Embedded Graphics Driver (IEGD).

To use the video engine, the media player application, MPlayer, for example, needs to use the appropriate API to communicate with the video driver that controls this video engine. This case study provides instructions on how to enable MPlayer to interface with Intel Embedded Graphics Driver (IEGD) video driver through the VA API.

This case study was done on a platform with the following characteristics: Intel Atom with US15W, Moblin-IVI 2.1 or Ubuntu 8, IEGD 10.2. However, the findings are applicable to other Intel Atom platforms using IEGD 10.x.

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.


Acknowledgment: Thanks to many people whose hard work is the basis for this paper. Special thanks to my colleagues Phillipe Lecluse and David Verbieren for helping put together the original instructions. To Karthik Thangavel for validating the instructions on several distros. And of course, Gwenole Beauchesne of Splitted Desktop for the groundbreaking work on advancing libVA and patching MPlayer.
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware Accelerated Decode</td>
<td>5</td>
</tr>
<tr>
<td>Building for libVA 0.29</td>
<td>7</td>
</tr>
<tr>
<td>Step-by-Step Instructions</td>
<td>7</td>
</tr>
<tr>
<td>Results</td>
<td>9</td>
</tr>
</tbody>
</table>
Hardware Accelerated Decode

The Intel Atom processor is being adopted increasingly by system vendors looking for low-cost/small-footprint/low-power consumption designs. Atom platforms feature rich multimedia capabilities and leverage the huge IA ecosystem.

The users still expect an experience delivered by their much bigger/expensive IA platforms with several applications running at once. This situation is especially challenging for real-time sensitive applications such as video playback that could consume the entire CPU bandwidth. To meet the low power/high performance demands, the Atom platform provides dedicated graphics and video engines—a hardware assist for delivering ultra-fast video decode and graphics rendering.

Media playback on the Atom platform is optimal when the video decoding is handled by the video engine instead of the CPU. This frees up the CPU for other non-video applications for a delightful user experience—multiple applications running simultaneously on a low-power envelope platform.

To use the video engine, the application, a media player for example, needs to use the appropriate API to communicate with the video driver that controls this video engine. If the media player/codec is not able to use the video engine, then by default the media playback will revert to the CPU, affecting overall system performance.

This case study describes how to enable MPlayer to communicate with Intel Embedded Graphics Driver (IEGD) video driver through the VA API.

Enabling Hardware Accelerated Playback for Intel® Atom™/Intel® US15W Platform and IEGD

Figure 1. Hardware Accelerated Decode Overview

[Diagram of hardware accelerated playback process]

Video Decoder Application

VA API

VA HW Driver

DRMLib

DRILib

X Server

DRI

DRM Kernel Module

Hardware (video decode, 3D graphics, etc)
Building for libVA 0.29

There may be situations where the libVA version supported by IEGD may be different from the libVA library included with the host distribution. If there is such a discrepancy between these libraries, libVA needs to be patched. Similarly, the application may also need patching to ensure libVA compatibility.

The following step-by-step instructions cover how to update the libVA 0.29 (supported by IEGD 10.x) to work with MPlayer. The instructions explain how to patch libVA and MPlayer so it can take advantage of hardware video acceleration through the VA API version supported by IEGD.

The patch and instructions come from the work of Gwenole Beauchesne at Splitted-Desktop Systems: http://www.splitted-desktop.com/~gbeauchesne/

Note: The patched libVA must be installed on the system prior to building the patched MPlayer. The libVA installation must be complete; it is not sufficient to copy the libva.so and corresponding symbolic links because the updated VA header files are required during the build of the modified MPlayer. Complete installation requires “make install” of libVA.

Step-by-Step Instructions

1. Download and Install Moblin
   a. Download and install the standalone Moblin image with open source VESA driver. This image is available from Moblin.org at: http://moblin.org/projects/2.1-ivi-fcrelease.
   b. Follow the installation directions at moblin.org to create a live USB flash image and then install the image on a hard drive.
   c. Set the path to the correct repository in the /etc/yum.repo.d
      The link to the repository is available where you downloaded the Moblin image in step a. above.

2. Install Required Packages
   There are several dependencies for installing VA API and a hardware accelerated MPlayer. In addition, the system needs several packages to create a development environment for compiling the patched libVA and MPlayer.
   a. Log in as a superuser.
   b. Install the following packages:
      
      yum install gcc gcc-c++ make imake rpm-build
      yum install byacc flex xorg-x11-util-macros hal-devel openssl-devel
      yum install libX11-devel libGL-devel libXfixes-devel libXv-devel
      libXaw-devel libXpm-devel
      yum install libXi-devel libXrender-devel libdmx-devel libXdmcp-devel
      libXfont-devel libXext-devel
yum install libXres-devel libXau-devel libdrm-devel xorg-x11-xtrans-devel
yum install xorg-x11proto-xf86driproto xorg-x11proto-damageproto
yum install xorg-x11proto-bigreqsproto
yum install xorg-x11proto-xf86vidmodeproto xorg-x11proto-xcmiscproto
yum install xorg-x11proto-x11proto-glproto
yum install xorg-x11proto-dri2proto xorg-x11proto-evieext xorg-x11proto-glproto
yum install xorg-x11proto-randrproto xorg-x11proto-resourceproto
yum install xorg-x11proto-compositeproto
yum install xorg-x11proto-scrnsaverproto libxkbfile-devel
yum install libpciaccess-devel pixman-devel
yum install libXinerama-devel libXdamage-devel
yum install libXxf86vm-devel expat-devel
yum install glx-utils
yum install libtool autoconf

3. Build and Install modified libVA
   a. Download patched libVA from Splitted-Desktop:
      http://www.splitted-desktop.com/~gbeauchesne/libva/
      (for IEGD 10.2, it is 0.29-2+sds12).
   b. Follow the instructions below to compile and install the patched libVA:
      cp libva-0.29-2+sds12.tar.gz /
      cd /
      tar xvzf libva-0.29-2+sds12.tar.gz
      cd libva-0.29
      for p in debian/patches/*.patch; do patch -p1 < $p; done
      ./autogen.sh
      make
      sudo make install

4. Install IEGD 10.x

5. Patch and Build MPlayer
   MPlayer needs a patch to use hardware video acceleration instead of forcing the
   CPU to decode. Download the MPlayer patch for VA API, MPlayer source, and then
   build MPlayer. Use the following steps to patch and build MPlayer.
   a. Install the Subversion (SVN) client, needed to download the MPlayer source
      code, for example, yum install SVN
   b. cd /
      Download the libVA 0.29 patch for MPlayer from http://www.splitted-
      desktop.com/~gbeauchesne/mplayer-vaapi/mplayer-vaapi-latest.tar.bz2
      (e.g., use mplayer-vaapi-20090423.tar.bz2).
   c. tar -xvjf mplayer-vaapi-20090423.tar.bz2
   d. cd mplayer-vaapi-20090423
Enabling Hardware Accelerated Playback for Intel® Atom™/Intel® US15W Platform and IEGD

6. Point to the correct libraries
   
   ```bash
   export LD_LIBRARY_PATH=/usr/local/lib/
   export LIBVA_DRIVERS_PATH=/usr/lib/xorg/modules/drivers
   ```

   **Note:** You may want to add these to the user’s bash profile to store this setup permanently or add it to `/etc/ld.so.conf/local.conf` and `/etc/profile.d` so it is enabled for all users.

7. Disable compositing window manager
   
   If you are using qfce window manager (post-September Moblin 2.1 build uses qfce), then you need to disable compositing as it consumes too many CPU cycles:
   
   Go to menu | settings | windows manager tweaks | compositor tab and clear “enable”

8. Play the video with patched MPlayer
   
   ```bash
   ./mplayer -vo vaapi -va vaapi -ao alsa /<example_video_clip>.mp4
   ```

**Results**

Use the top utility to check the CPU utilization:

```bash
# top
```

The CPU utilization should be 30%-60%.

Authors

Ishu Verma is a Software Technical Marketing Engineer at Intel Corporation.

Acronyms

IA  Intel Architecture, for example, the specific architectures originally designed by Intel, such as IA-32 and IA-64.

IEGD  Intel® Embedded Graphics Driver is composed of a runtime graphics driver and a Video BIOS (VBIOS) firmware component. Both the driver and VBIOS control the south bridge to perform display and render operations.

MPlayer  Versatile, open source video player that uses FFmpeg media infrastructure.

SVN  Subversion*, a version-control system initiated in 2000 by CollabNet Inc. Developers use Subversion to maintain current and historical versions of files such as source code, web pages, and documentation.

VA API  Video Acceleration API (VA API) is an open source software library ("libVA") and API specification which enables and provides access to graphics hardware (GPU) acceleration for video processing on Linux and UNIX based operating systems. Accelerated processing includes video decoding, video encoding, sub-picture blending and rendering.