

AM335x 平台在全彩 LED 显示墙异步控制卡的应用

秦耀明

TI 通用嵌入式处理器技术支持

摘要

全彩 LED 显示墙异步控制卡以成本低，集中管理等特点，逐渐成为全彩 LED 显示墙控制卡的主流。AM335x 具有丰富的硬件外设，基于 Linux 的软件方案，包含 GPU Composition 模块能提供完整的多图层叠加缩放等功能，十分适合全彩 LED 显示墙的异步控制卡应用。本文将从硬件和软件两个方面介绍基于 AM335x 提供的相应解决方案。

内容

摘要

1.1	同步控制卡	2
1.2	异步控制卡	2
2	异步控制卡系统分析	3
2.1	硬件部分	3
2.2	软件部分	3
2.2.1	操作系统	3
	全彩 LED 显示墙控制卡简介	
3	AM335x 的解决方案	4
3.1	硬件方案	4
3.2	软件方案	5
3.2.1	操作系统	5
3.2.2	软件模块	5
3.2.3	软件复杂度分析	6
3.2.4	基于 GPU 的优化方案 – GPU Composition	7
4	方案实验	9
4.1	代码及编译	10
4.2	代码运行	10
4.3	性能分析	10
5	总结	11

图

图 1	同步控制模块图	2
图 2	异步控制卡模块图	2
图 3	AM335x 异步控制卡硬件模块图	5
图 4	默认软件方案数据流程图	6
图 5	GPU Composition 软件模块图	7
图 6	示例运行效果图	10
图 7	单 Cortex-A8 软解视频流的系统负载	11

图 8 GPU Composition 方案视频播放的系统负载..... 11

1 全彩 LED 显示墙控制卡简介

全彩 LED 显示墙控制卡根据控制方式，可以分为两大类：同步控制卡和异步控制卡

1.1 同步控制卡

全彩 LED 同步显示墙主要由 PC，同步控制卡和 LED 显示模块组三部分组成，其连接方式如下：

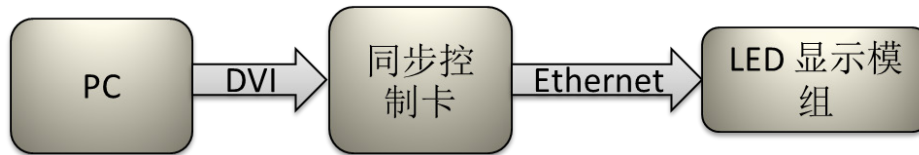


图 1 同步控制模块图

同步控制卡将 DVI 信号转成 LED 显示模组所需要的视频信号格式，而且用以太网的方式传输给 LED 显示模组。同步控制卡本身不做视频解码等处理，仅做格式转换。因此，一般采用 FPGA 实现该功能。

1.2 异步控制卡

全彩 LED 异步显示墙由异步控制卡和 LED 显示模组组成，其连接方式如下：

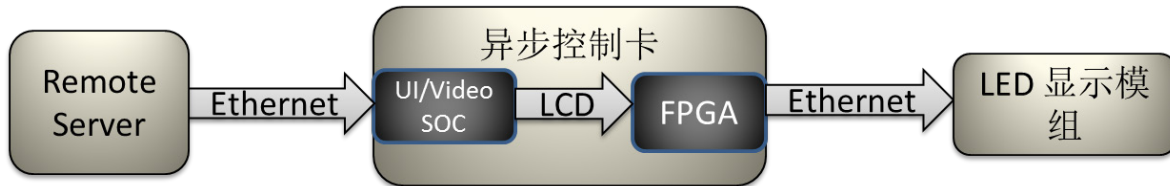


图 2 异步控制卡模块图

由上图，异步控制卡主要由两个大的部分组成：

- 视频处理模块。
在此模块中，SOC 从网口得到视频流以及 UI 的素材，进行视频解码和 UI 绘制，最后通过 LCD 接口传送给 FPGA。
- 视频信号转换模块。
在此模块中，FPGA 将视频信号转换成 LED 显示模组所需的信号，并通过网口输出，该功能和同步控制卡的功能一样。

对比两种方案，可见异步控制卡具有成本低，便于集中管理的特点。

2 异步控制卡系统分析

下面从硬件和软件两个方面分析其主芯片的系统需求。

2.1 硬件部分

从硬件上看，视频处理模块部分主要由最小系统和外围模块两大部分组成。

- 最小系统

最小系统由主芯片，电源系统，DDR 和存储四部分组成。

不同级别的全彩屏对 SOC 的处理能力有不同要求，具体的要求在软件部分有说明。

- 外围模块

- 音频接口，LCD 接口。即 LED 显示墙的基本需求。
- 网络接口。百兆甚至千兆网口可以有效保证显示内容更新的高效性。
- USB 接口。便于系统升级，以及扩展基于 USB 各种外设。
- SD 卡/TF 卡支持。便于系统升级以及内容的本地更新。

此外，异步卡一般和 LED 显示墙一起放置于室外，所以需要可工作在宽温度范围的工业级芯片。

2.2 软件部分

软件部分主要由操作系统和应用软件两大部分组成。

2.2.1 操作系统

在异步控制卡行业中，主流系统选择了 Linux。

2.2.2 应用软件

应用软件主要包含三个部分：

- 多媒体部分。

用于对音视频码流的解码。

全彩屏主要分为高端和中低端两个档次：

- 高端，视频分辨率以及显示分辨率要求在 720p 分辨率以上。
- 中低端，视频分辨率以及显示分辨率在 640x480 以内。

由于 LED 墙一般显示物理面积大，而且亮度高，所以对视频流的帧率要求较高，要求在每秒 25 帧以上。因此，对于高端产品，一般需带有视频硬解码模块的主芯片，其价格一般较高；对于低端产品，使用软解码可实现，所以需要运算性能较强的主芯片，成本优势较好。

- **UI 部分。**

用于显示字幕，图片等，并处理 UI 元素和视频层的叠加。叠加部分。由于涉及到透明度，尺寸变换等，运算需求也很大，所以需要主芯片具有相关的硬件加速模块。

- **远程控制部分。**

该部分主要实现上位机对各控制卡的远程控制，内容更新等功能。该部分一般通过网络应用层实现，各控制厂家有自己的协议。

3 AM335x 的解决方案

AM335x 是 TI 新近推出的基于 ARM Cortex-A8 的 SOC，外设丰富，主要针对工业应用领域。针对异步控制卡应用，TI 也提供了基于 Linux 的解决方案。下面将从硬件和软件两方面分别介绍该方案。

3.1 硬件方案

AM335x 具有一个强劲的核心 Cortex-A8，该核的运算能力可达 2.0DMIPS/MHz，而且 AM335x 的主频可到 1GHz，即运算总的的能力可达 2000 DMIPS，可流畅解码 640x480 的 MPEG4 视频流，而且有足够的运算余量绘制各种 UI。

此外，AM335x 还有一个 3D 图形加速核，SGX530，可支持 OpenGL ES2.0。TI 在 OpenGL ES2.0 之上提供了相应的软件方案，将 SGX530 用于视频帧的尺寸缩放以及实现对 UI 层和视频层的透明叠加的加速，后面软件部分会详细介绍该方案。

同时，AM335x 具有丰富的外设，如下图所示：

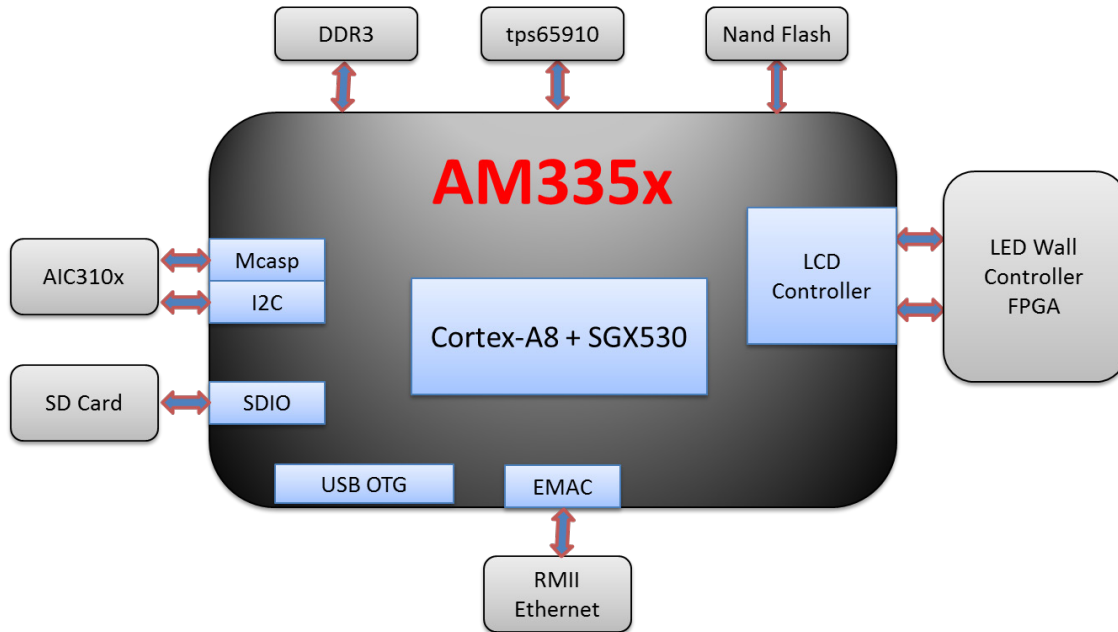


图 3 AM335x 异步控制卡硬件模块图

由上图可见 AM335x 可完全涵盖所有异步控制卡的外设需求，不需要其他扩展。因此，总体成本具有很强竞争力。

TI 的开发板 GP EVM（可查阅参考文档[1]）都可以很便利的进行 LED 应用的评估和开发，下文中的软件方案是以 GP EVM 为平台进行开发的。

3.2 软件方案

软件方案主要分为操作系统和应用软件两大块，具体介绍如下。

3.2.1 操作系统

如前所述，Linux 是异步控制卡的主流操作系统，因此，本方案也选择了 Linux 作为平台。AM335x EZSDK 提供了 Linux 的完整开发包，包括板级支持包，交叉编译器，文件系统等，可查阅参考文档[2]。

3.2.2 软件模块

- UI

在基于 Linux 的异步控制卡平台上，QT 以免费，开源，开发资料全以及在嵌入式系统上运行效率高等特点，已经成为异步控制卡厂商开发 UI 主要的平台。在 EZSDK 中已包含对 QT4 的移植，可查阅参考文档[3]。QT 在开源网站上也有很丰富的资源，可查阅参考文档[4]。

- 多媒体

在 EZSDK 中提供 Gstreamer+ffmpeg 的多媒体解决方案，可查阅参考文档[5] 和参考 Gstreamer 文档（参考文档[6]）。在多媒体中，由于格式比较多，各种编码的复杂度以及编码质量差异较大是一个难点。而在 LED 显示墙的应用场景中，多媒体码流可接受转码方式，所以可指定码流的格式。这里，推荐的多媒体格式 MP4（MPEG4+AAC），其中 MPEG4 选择 simple profile，对此种码流，若分辨率为 640x480，AM335x 可流畅解码每秒 25 帧以上。

- 显示后端

AM335x 只有一个功能简单的 LCD 控制器，该控制器只支持 RGB 格式，其在 Linux 中的驱动为 framebuffer，可查阅参考文档[7]。相应的上述两个模块的显示后端也以 framebuffer 为基础：

- Gstreamer 的后端显示插件采用 fbdevsink。由于视频解码后的格式为 YUV 格式，而 AM335x 自带的 LCD 控制器只支持 RGB 格式，因此此处可使用 Gstreamer 的插件 ffmpegcolourspace 进行色度空间的转换
- QT 默认以 framebuffer 为显示后端。

Framebuffer 会接收来自 QT 和 Gstreamer 的图像帧数据，然后进行 OSD 的叠加和缩放等操作，数据流如下图所示：

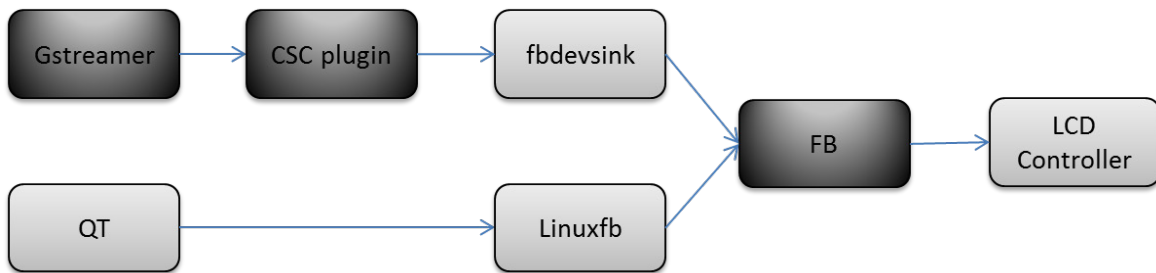


图 4 默认软件方案数据流程图

3.2.3 软件复杂度分析

在图 4 中，深色模块为运算较密集模块，具体分析如下：

- Gstreamer 的解码和 ffmpegcolourspace（CSC plugin）两个模块。

ARM 虽然有较强的运算能力，但对于较大分辨率的视频解码，视频解码的宏块运算等需较大运算量。另外，色度空间涉及浮点运算，而且为逐点运算，所以运算量需求也不小。

以 640x480 分辨率的 MP4(MPEG4 simple profile+AAC) 为例，若帧率为 30fps 时，ARM 核的 loading 在 91% 左右，其中 ffmpegcolourspace 模块约占运算量的 50%。

- Framebuffer 模块。

在该模块中的 OSD 叠加指的是 UI 图层和视频图层之间的叠加，而且是包含带透明度的叠加，而图层的缩放是指对原图等比例的缩放，因而需对每一帧数据的每个像素点进行浮点乘加运算，参考 ffmpegcolourspace 的运算量，该部分运算量也应较大。

可见，ARM 核无法独自胜任系统所需的全部运算。

3.2.4 基于GPU的优化方案 – GPU Composition

GPU Composition 软件模块，调用 SGX530 模块进行色彩空间转换，OSD 叠加，图层缩放功能，分担 A8 的运算负载使其专注于 QT，视频解码等应用，下面将具体介绍。

- GPU Composition 模块的编译和安装
在 TI Wiki 上有明确说明，可查阅参考文档[8]。
- GPU Composition 设计分析
 - A. 各功能模块

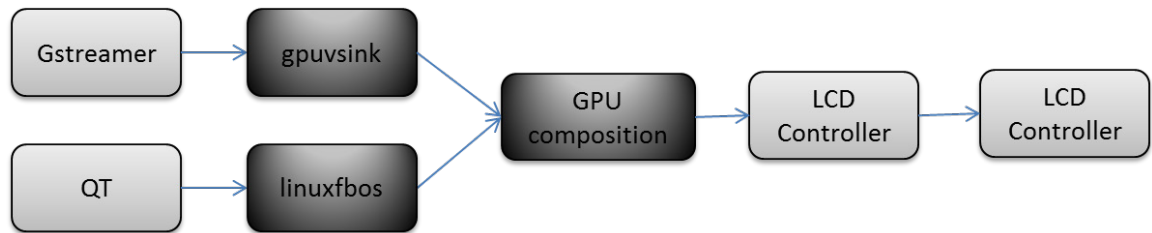


图 5 GPU Composition 软件模块图

SGX530 实现的功能模块标记为深色，具体功能如下：

a. gpusink

该模块设计为 Gstreamer 视频显示后端插件，将视频解码器解出的 YUV 数据帧，传送给 SGX530 模块。按照标准的 Gstreamer 视频显示后端插件设计，可采用标准的显示后端接口编程。对于视频输入的尺寸，要求其宽（width）为 4 个像素点的倍数。其输出视频帧数据这里可称为 Video Plane。

b. linuxfbos

该模块设计为 QT 架构中的显示后端，将 QT 的帧数据发送到 SGX530 模块中处理。linuxfbos 和 framebuffer 有同样的接口，对于 QT 应用开发是透明的。其输出界面帧数据为 Graphics Plane。

GPU Composition

该模块基于 Open GL ES 2.0 接口设计，对输入的 Video plane 和 Graphics Plane 进行色彩空间转换，图层缩放，OSD 叠加等操作，将最终的帧数据推送到 Framebuffer 中显示。

B. 模块间的数据流

模块间的数据以 Plane 的形式传递，具体介绍如下：

a. Plane 格式

Video Plane 可支持 YUV422, NV12, I420 和 YUV420 格式帧数据。

Graphics Plane 可支持 RGB565, RGB888 和 ARGB8888 帧数据。

GPU Composition 接收这些格式的帧数据，并将其转换为 RGB 格式，进行图层缩放，OSD 叠加等操作。

b. Plane 的内存分配

SGX530 输入内存 (Buffer)，只支持物理地址连续的 Buffer。因此，在 gpuvsink 和 linuxfbofs 中，使用 cmem (具体可查阅参考文档[9]) 据此要求分配内存 Pool 来存储帧数据，需在 Linux 启动时通过命令行参数 "mem=" 配置预留给 Kernel 的内存，而剩下的内存即是给 cmem 所准备，用于分配物理连续的内存。

其大小的计算公式如下：

```
Pool size for Graphics Plane = width * height * Bytes Per Pixel
Pool size for Video Plane = video frame width * height * 2 (Bytes
Per Pixel) * 8 (buffers)
```

对于一个 Video Plane 可能需要多个 Buffer，其具体个数定义在

```
gpu-compositing/gpuvsink/src/gst_render_bridge.h
#define PROP_DEF_QUEUE_SIZE 8
```

c. Pool 传递

Graphics Plane 和 Video Plane 以指针的形式将 Pool 传递给 GPU Composition。

C. 模块间的控制流

a. 配置信息数据结构

对于 Graphics Plane，通过命名管道"/opt/gpu-compositing/named_pipes/video_cfg_and_data_plane_X"其配置信息在下面数据结构中

```
typedef struct
{
    int enable; /* 1 - enable the gfx plane; 0 - disable */
    int input_params_valid; /* 1 - valid i/p parameters; 0 - invalid */
    struct in_g {
        unsigned long data_ph_addr; /* physical address of the gfx buffer */
        int width; /* gfx plane width in pixels */
        int height; /* gfx plane height in pixels */
        unsigned int pixel_format; /* fourcc pixel format */
        int enable_blending; /* 1 - blending enabled; 0 - disabled */
        int enable_global_alpha; /* 1 - global alpha; 0 - pixel alpha */
        float global_alpha; /* global alpha value [0.0 to 1.0] */
        float rotate; /* rotate angle in decimal degrees [-180.0 to
180.0] */
    } in_g;
    /* output window position and resolution in normalized device co-ordinates */
    int output_params_valid; /* 1 - valid o/p parameters; 0 - invalid */
    struct out_g {
        float xpos; /* x position [-1.0 to 1.0] */
        float ypos; /* y position [-1.0 to 1.0] */
        float width; /* width [0.0 to 2.0], 2.0 correspond to fullscreen width */
        float height; /* height [0.0 to 2.0], 2.0 correspond to fullscreen height */
    } out_g;
} gfxCfg_s;
```

关于此配置信息中，比较重要的有如下几点：

- 对于 QT 而言，输入的窗口参数来自 Linux 的 FB 窗口，即 LCD 屏的显示分辨率。
- 关于透明度 (Alpha)，Video plane 在底部，因此，Graphics Plane 决定 Video Plane 的可透明度。Alpha 可分：全局 Alpha，整个 Plane 使用同一的一个 Alpha；以像素点 (Pixel) 为单位的 Alpha，即像素的数据格式为 ARGB8888，可以在局部位置 Alpha。
- 可以通过窗口输出的结构体 out_g 窗口输出 Plane 的窗口。

对于 Video Plane, 配置信息如下：

```
#define MAX_VIDEO_BUFFERS_PER_CHANNEL 16
typedef struct
{
    int config_data; /* 1 - config 0 - data 2 - close the video plane (to close
the named pipe corresponding to video plane) */
    int buf_index; /* if data, buffer index */
    int enable; /* 1 - enable the video plane; 0 - disable */
    /* Video plane config structure */
    struct in {
        float rotate; /* rotate angle in decimal degrees [-180.0 to 180.0]*/
        int count; /* Number of video buffers */
        int width; /* video frame width in pixels */
        int height; /* video frame height in pixels */
        unsigned int fourcc; /* pixel format */
        unsigned long phyaddr[MAX_VIDEO_BUFFERS_PER_CHANNEL]; /* Physical addresses of
video buffers */
    } in;
    /* output video window position and resolution in normalized device co-ordinates */
    struct out {
        float xpos; /* x position [-1.0 to 1.0] */
        float ypos; /* y position [-1.0 to 1.0] */
        float width; /* width - [0.0 to 2.0], 2.0 correspond to fullscreen width */
        float height; /* height - [0.0 to 2.0], 2.0 correspond to fullscreen height */
    } out;
} videoConfig_s;
```

此配置结构体中的输入信息，会通过 Gstreamer 的标准接口，通过前级的 Gstreamer Plugin 进行配置。如前所述，输入视频帧的宽 (width) 的像素点数，需为 4 的倍数；对于输出信息，和 Graphics Plane 一样，可以通过配置输出数据结构 out，实现缩放功能。

b. 命名管道 (named pipe) 配置信息

上述配置信息，通过存放于文件系统中的命名管道，传递到 GPU Composition 模块。

对于 linuxfbfs，命名管道文件为 /opt/gpu-compositing/named_pipes/gfx_cfg_plane_X。

对于 gpuvsink，命名管道文件为 /opt/gpu-compositing/named_pipes/video_cfg_and_data_plane_X

4 方案实验

笔者基于 GPU Composition 方案，在 AM335x EVM 板上，开发了 Gstreamer 和 QT 应用程序，以验证整个异步 LED 显示墙方案的性能。

4.1 代码及编译

代码分成两个应用：

- Gstreamer 部分，可在参考文档[10]下载，为一个视频播放器，可循环播放 MP4 视频
- QT 部分，可以在参考文档[11]下载，包括一个时钟和滚动字幕。

可以根据参考文档[8]进行编译。

4.2 代码运行

在 AM335x EVM(AM3358 主频为 720MHz)上，运行命令行如下：

```
./playbin /usr/share/ti/video/HistoryOfTIAV-480p.mp4 &  
  
./clock -qws -display linuxfbofs:gfx_no=1:xpos=-  
1.0:ypos=1.0:width=2.0:height=2.0:blend_en=1:glob_alpha_en=1:global_alpha=0.5:rotate=0.  
0 &
```

这里播放的视频流为 mp4 格式，其包含有分辨率为 640x480 的 MPEG4 simple profile 码流以及 AAC 音频流。

运行效果图如下：



图 6 示例运行效果图

可以从截图中看到，OSD 层和视频层的透明度叠加很清楚。

4.3 性能分析

关于 GPU Composition 方案的性能提高，可以参考下面两个截图。

图 7 中，没有使用 GPU Composition 方案，CPU 除了要做解码，也需要做色彩空间转换，其 CPU 占用率达到 91%。

图 8 中，使用了 GPU 分担了视频叠加，色彩空间转换等运算，在整个系统的总运算量明显大于仅仅 Gstreamer 播放视频的情况下，ARM 核的 CPU 占用率仅仅只有 58%，仍给应用程序留下运行的空间。

更多的示例可查阅参考文献[8]。

```
Mem: 113544K used, 10012K free, 0K shrd, 32K buff, 85292K cached
CPU: 72% usr 27% sys 0% nic 0% idle 0% io 0% irq 0% sirq
Load average: 0.44 0.51 0.27 2/142 1592
  PID PPID USER STAT VSZ %MEM %CPU COMMAND
 1588 1526 root S 60976 49% 91% /usr/bin/gst-launch-0.10 filesrc locat
 1592 1526 root R 2668 2% 9% top
 1204 1 root S 16868 14% 0% /usr/sbin/console-kit-daemon
 1201 1 haldaemo S 12760 10% 0% /usr/sbin/hald
 1197 1 messageb S 3328 3% 0% /usr/bin/dbus-daemon --system
 1288 1269 root S 3316 3% 0% /usr/libexec/hald-addon-cpufreq
 1280 1269 root S 3304 3% 0% hald-addon-input: Listening on /dev/in
 1282 1269 root S 3300 3% 0% /usr/libexec/hald-addon-rfkill-killswi
 1283 1269 root S 3300 3% 0% /usr/libexec/hald-addon-generic-backli
 1284 1269 root S 3300 3% 0% /usr/libexec/hald-addon-generic-backli
 1269 1201 root S 3212 3% 0% hald-runner
 1385 1 root S 3032 2% 0% /usr/sbin/lighttpd -f /etc/lighttpd.co
 1526 1522 root S 2844 2% 0% -sh
 1318 1 root S 2732 2% 0% /sbin/syslogd -n -C64 -m 20
 1320 1 root S 2668 2% 0% /sbin/klogd -n
 1306 1 root S 2668 2% 0% /usr/sbin/telnetd
 1327 1 root S 2448 2% 0% /usr/sbin/tthttpd -d /srv/www -p 8080 -
 1522 1 root S 2400 2% 0% login -- root
 1313 1 root S 2200 2% 0% /usr/sbin/netserver
 939 1 root S < 2188 2% 0% /sbin/udevd -d
```

图 7 单 Cortex-A8 软解视频流的系统负载

```
Mem: 118764K used, 4792K free, 0K shrd, 32K buff, 85120K cached
CPU: 56% usr 3% sys 0% nic 39% idle 0% io 0% irq 0% sirq
Load average: 0.99 0.65 0.28 2/152 1572
  PID PPID USER STAT VSZ %MEM %CPU COMMAND
 1552 1526 root S 84532 68% 53% ./playbin2 ./ti4.mp4
 1533 1526 root S 88360 71% 5% ./composition
 417 2 root SW 0 0% 1% [kworker/0:1]
 8 2 root SW 0 0% 1% [kworker/u:1]
 1571 1526 root R 2844 2% 0% top
 5 2 root SW 0 0% 0% [kworker/u:0]
 1385 1 root S 3032 2% 0% /usr/sbin/lighttpd -f /etc/lighttpd.co
 1204 1 root S 16868 14% 0% /usr/sbin/console-kit-daemon
 1201 1 haldaemo S 12760 10% 0% /usr/sbin/hald
 1197 1 messageb S 3328 3% 0% /usr/bin/dbus-daemon --system
 1288 1269 root S 3316 3% 0% /usr/libexec/hald-addon-cpufreq
 1280 1269 root S 3304 3% 0% hald-addon-input: Listening on /dev/in
 1282 1269 root S 3300 3% 0% /usr/libexec/hald-addon-rfkill-killswi
 1283 1269 root S 3300 3% 0% /usr/libexec/hald-addon-generic-backli
 1284 1269 root S 3300 3% 0% /usr/libexec/hald-addon-generic-backli
 1269 1201 root S 3212 3% 0% hald-runner
 1526 1522 root S 2844 2% 0% -sh
 1318 1 root S 2732 2% 0% /sbin/syslogd -n -C64 -m 20
 1320 1 root S 2668 2% 0% /sbin/klogd -n
End of streamoot S 2668 2% 0% /usr/sbin/telnetd
```

图 8 GPU Composition 方案视频播放的系统负载

5 总结

本文主要介绍了基于 AM335x 的全彩 LED 显示墙异步控制卡方案，重点介绍了基于 GPU 的软件解决方案，在实现 LED 显示墙所需的视频层和 OSD 层叠加，缩放等功能的基础上，仍给客户定制的应用程序提供了足够的开发空间。希望该方案能加速客户进行异步控制卡的开发。

6 参考文献

1. <http://www.ti.com/tool/tmdxevm3358>
2. http://processors.wiki.ti.com/index.php/Sitara_Linux_Software_Developer%E2%80%99s_Guide
3. http://processors.wiki.ti.com/index.php/Building_Qt
4. <http://qt-project.org/>
5. http://processors.wiki.ti.com/index.php/ARM_Multimedia_Users_Guide
6. <http://gststreamer.freedesktop.org/>
7. http://processors.wiki.ti.com/index.php/AM335x_LCD_Controller_Driver%27s_Guide
8. http://processors.wiki.ti.com/index.php/GPU_Compositing#About_GPU_Compositing
9. http://processors.wiki.ti.com/index.php/CMEM_Overview
10. <http://processors.wiki.ti.com/index.php/File:Playbin2.zip>
11. <http://processors.wiki.ti.com/index.php/File:Clock.zip>
12. http://processors.wiki.ti.com/index.php/Graphics_SDK_Quick_installation_and_user_guide

重要声明

德州仪器(TI) 及其下属子公司有权根据 JESD46 最新标准, 对所提供的产品和服务进行更正、修改、增强、改进或其它更改, 并有权根据 JESD48 最新标准中止提供任何产品和服务。客户在下订单前应获取最新的相关信息, 并验证这些信息是否完整且是最新的。所有产品的销售都遵循在订单确认时所提供的TI 销售条款与条件。

TI 保证其所销售的组件的性能符合产品销售时 TI 半导体产品销售条件与条款的适用规范。仅在 TI 保证的范围内, 且 TI 认为 有必要时才会使用测试或其它质量控制技术。除非适用法律做出了硬性规定, 否则没有必要对每种组件的所有参数进行测试。

TI 对应用帮助或客户产品设计不承担任何义务。客户应对其使用 TI 组件的产品和应用自行负责。为尽量减小与客户产品和应用相关的风险, 客户应提供充分的设计与操作安全措施。

TI 不对任何 TI 专利权、版权、屏蔽作品权或其它与使用了 TI 组件或服务的组合设备、机器或流程相关的 TI 知识产权中授予 的直接或隐含权限作出任何保证或解释。TI 所发布的与第三方产品或服务有关的信息, 不能构成从 TI 获得使用这些产品或服务 的许可、授权、或认可。使用此类信息可能需要获得第三方的专利权或其它知识产权方面的许可, 或是 TI 的专利权或其它 知识产权方面的许可。

对于 TI 的产品手册或数据表中 TI 信息的重要部分, 仅在没有对内容进行任何篡改且带有相关授权、条件、限制和声明的情况 下才允许进行复制。TI 对此类篡改过的文件不承担任何责任或义务。复制第三方的信息可能需要服从额外的限制条件。

在转售 TI 组件或服务时, 如果对该组件或服务参数的陈述与 TI 标明的参数相比存在差异或虚假成分, 则会失去相关 TI 组件 或服务的所有明示或暗示授权, 且这是不正当的、欺诈性商业行为。TI 对任何此类虚假陈述均不承担任何责任或义务。

客户认可并同意, 尽管任何应用相关信息或支持仍可能由 TI 提供, 但他们将独力负责满足与其产品及其应用中使用的 TI 产品 相关的所有法律、法规和安全相关要求。客户声明并同意, 他们具备制定与实施安全措施所需的全部专业技术和知识, 可预见 故障的危险后果、监测故障及其后果、降低有可能造成人身伤害的故障的发生机率并采取适当的补救措施。客户将全额赔偿因 在此类安全关键应用中使用任何 TI 组件而对 TI 及其代理造成的任何损失。

在某些场合中, 为了推进安全相关应用有可能对 TI 组件进行特别的促销。TI 的目标是利用此类组件帮助客户设计和创立其特 有的可满足适用的功能安全性标准和要求的终端产品解决方案。尽管如此, 此类组件仍然服从这些条款。

TI 组件未获得用于 FDA Class III (或类似的生命攸关医疗设备) 的授权许可, 除非各方授权官员已经达成了专门管控此类使 用的特别协议。

只有那些 TI 特别注明属于军用等级或“增强型塑料”的 TI 组件才是设计或专门用于军事/航空应用或环境的。购买者认可并同 意, 对并非指定面向军事或航空航天用途的 TI 组件进行军事或航空航天方面的应用, 其风险由客户单独承担, 并且由客户独 力负责满足与此类使用相关的所有法律和法规要求。

TI 已明确指定符合 ISO/TS16949 要求的产品, 这些产品主要用于汽车。在任何情况下, 因使用非指定产品而无法达到 ISO/TS16949 要 求, TI 不承担任何责任。

	产品		应用
数字音频	www.ti.com.cn/audio	通信与电信	www.ti.com.cn/telecom
放大器和线性器件	www.ti.com.cn/amplifiers	计算机及周边	www.ti.com.cn/computer
数据转换器	www.ti.com.cn/dataconverters	消费电子	www.ti.com.cn/consumer-apps
DLP® 产品	www.dlp.com	能源	www.ti.com.cn/energy
DSP - 数字信号处理器	www.ti.com.cn/dsp	工业应用	www.ti.com.cn/industrial
时钟和计时器	www.ti.com.cn/clockandtimers	医疗电子	www.ti.com.cn/medical
接口	www.ti.com.cn/interface	安防应用	www.ti.com.cn/security
逻辑	www.ti.com.cn/logic	汽车电子	www.ti.com.cn/automotive
电源管理	www.ti.com.cn/power	视频和影像	www.ti.com.cn/video
微控制器 (MCU)	www.ti.com.cn/microcontrollers		
RFID 系统	www.ti.com.cn/rfidsys		
OMAP应用处理器	www.ti.com.cn/omap		
无线连通性	www.ti.com.cn/wirelessconnectivity	德州仪器在线技术支持社区	www.deyisupport.com

邮寄地址: 上海市浦东新区世纪大道 1568 号, 中建大厦 32 楼 邮政编码: 200122
Copyright © 2013 德州仪器 半导体技术(上海)有限公司