Linux GPU
开发指南

版本号: 1.3
发布日期: 2022.4.26

版本历史

版本号	日期	制/修订人	内容描述
1.0	2020.7.2	AWA1639	初始化版本
1.1	2021.5.13	AWA1639	增加适配 Mali-G31
1.2	2022.4.25	AWA1831	增加适配 H618
1.3	2022.4.26	AWA1831	增加适配 Mali-400

目　录

1 前言

1.1 文档简介

介绍 Sunxi 平台上 GPU 驱动模块的一般使用方法及调试接口，为开发与调试提供参考。

1.2 目标读者

GPU 驱动开发人员及 GPU 应用开发和维护人员。

1.3 适用范围

表 1-1：适用产品列表

产品名称	内核版本	驱动文件	GPU 型号
T509	Linux-4.9	modules/gpu/img-rgx/*	GE8300
MR813	Linux-4.9	modules/gpu/img-rgx/*	GE8300
R818	Linux-4.9	modules/gpu/img-rgx/*	GE8300
A133	Linux-4.9&Linux-5.4	modules/gpu/img-rgx/*	GE8300
A33	Linux-4.9	modules/gpu/mali-utgard/*	Mali-400
H616	Linux-4.9	modules/gpu/mali-bifrost/*	Mali-G31
H618	Linux-5.4	modules/gpu/mali-bifrost/*	Mali-G31
T507	Linux-4.9	modules/gpu/mali-bifrost/*	Mali-G31
T507-H	Linux-4.9	modules/gpu/mali-bifrost/*	Mali-G31
T517-H	Linux-4.9	modules/gpu/mali-bifrost/*	Mali-G31

说明

mali-bifrost、mali-utgard、rgx 同时也代表着 GPU 的软件架构;

2 模块介绍

2.1 模块功能介绍

GPU 是图形加速引擎，能够提供 2D 和 3D 加速，能够绘制普通 UI、游戏，能够做缩放、全景拼接、畸变矫正等处理，GPU 还能提供并行运算算力。

在有硬件做支撑的前提下，软件的兼容性显得特别重要。目前使用最为广泛的图形加速 API 是 Khronos 组织提出来的 OpenGL，在移动端产品中对应地叫 OpenGL ES。OpenGL ES 是在 OpenGL 的基础上做一些裁剪，以更好地适应移动端产品对功耗和成本限制的较高要求。除了 OpenGL 之外，Khronos 还定义了 Vulkan 这一套新的图形渲染 API，旨在替代 OpenGL/OpenGL ES。此外，Khronos 组织还定义了一套并行运算 API，即 OpenCL，开发者可以通过 OpenCL 接口来使用 GPU 硬件进行算法加速。

不同的平台使用的 GPU 硬件型号一般都是有差异的，型号不同其驱动也有所不同，但同一型号的 GPU 的配置方法一般都是一样的，同时不同型号的 GPU 也有共性方面的使用及配置，下面的说明如未分型号说明，则表明是所有 GPU 通用的。

2.2 相关术语介绍

2.2.1 硬件术语

表 2-1: 模块硬件相关术语介绍

相关术语	解释说明
GPU	Graphics Processing Unit，图形处理单元，主要用于 2D 和 3D 加速

2.2.2 软件术语

表 2-2: 模块硬件相关术语介绍

相关术语	解释说明
DVFS	动态电压频率调整，用于根据不同需求动态调节频率及电压
GPU idle	根据 GPU 空闲与否自动开关电源及时钟

2.3 模块配置介绍

2.3.1 Device Tree 配置说明

Device Tree 主要用来配置模块相关的参数，例如中断、寄存器、时钟信息、vf 表等。
Device Tree 文件的路径为：
arch/arm64（32 位平台为 arm）/boot/dts/sunxi/{CHIP}.dtsi(CHIP 为研发代号，如 sun50iw10p1 等)
以及
device/config/chips/{IC}/configs/{BOARD}/board.dts(IC 为产品型号，如 T509，BOARD 为板型如 perf1)
其中，前者用于配置中断、寄存器、时钟信息等固定参数，后者则用于配置 vf 表、各种功能使能开关等可变参数。

2.3.1.1 GE8300

GE8300 的 Device Tree 配置信息如下：

```
    gpu: gpu@0x01800000 {
        device_type = "gpu";
        compatible = "img,gpu";
        reg = <0x0 0x01800000 0x0 0x80000>;//寄存器地址
        interrupts = <GIC_SPI 97 IRQ_TYPE_LEVEL_HIGH>;//中断
        interrupt-names = "IRQGPU";
        clocks = <&clk_pll_gpu>, <&clk_gpu>;//时钟
        clock-names = "clk_parent", "clk_mali";
        power-domains = <&pd_gpu>;
    };
```

```
    gpu: gpu@0x01800000 {
            gpu_idle = <0>;
            dvfs_status = <1>;
            pll_rate = <504000>;
            independent_power = <0>;
            operating-points = <
                 /* KHz   uV */
                504000 950000
                472500 950000
                441000 950000
                252000 950000
             >;
             gpu-supply = <&reg_dcdc4>;
    };
```

GE8300 在 board.dts 中的配置参数说明如下：

表 2-3: board.dts 配置说明

参数名	说明
independent_power	GPU 是否独立供电，0 表示非独立供电，1 表示独立供电
gpu_idle	是否打开 GPU idle，0 表示否，1 表示是
dvfs_status	是否打开 DVFS，0 表示关闭，1 表示打开
operating-points	GPU 的 vf 表
gpu-supply	GPU 独立供电时使用的 regulator 的 id

2.3.1.2 Mali-G31

Mali-G31 的 Device Tree 配置信息如下：

```
    gpu: gpu@0x01800000 {
        device_type = "gpu";
        compatible = "arm,mali-midgard";
        reg = <0x0 0x01800000 0x0 0x10000>;//寄存器地址
        interrupts = <GIC_SPI 95 IRQ_TYPE_LEVEL_HIGH>,//中断
                 <GIC_SPI 96 IRQ_TYPE_LEVEL_HIGH>,
                 <GIC_SPI 97 IRQ_TYPE_LEVEL_HIGH>;
        interrupt-names = "JOB", "MMU", "GPU";
        clocks = <&clk_pll_gpu>, <&clk_gpu0>, <&clk_gpu1>;//时钟
        clock-names = "clk_parent", "clk_mali", "clk_bak";
        #cooling-cells = <2>;
        ipa_dvfs:ipa_dvfs {//温控ipa相关参数
            compatible = "arm,mali-simple-power-model";
            static-coefficient = <17000>;
            dynamic-coefficient = <750>;
            ts = <254682 9576 0xffffff98 4>;
            thermal-zone = "gpu_thermal_zone";
            ss-coefficient = <36>;
            ff-coefficient = <291>;
        };
    };
```

```
    gpu: gpu@0x01800000 {
            gpu_idle = <1>;
            dvfs_status = <0>;
            operating-points = <
                 /* KHz   uV */
                600000 950000
                576000 950000
                540000 950000
                504000 950000
             >;
    };
```

Mali-G31 在 board.dts 中的配置参数及含义与 GE8300 一致，详情可参考上一章节。

2.3.1.3 Mali-400

Mali-400 的 Device Tree 配置信息如下：

```
    gpu_mali400_0: gpu@1c40000 {
        compatible = "arm,mali-400", "arm,mali-utgard";
        reg = <0x0 0x01c40000 0x0 0x10000>;
        interrupts = <GIC_SPI 97 4>, <GIC_SPI 98 4>, <GIC_SPI 99 4>, <GIC_SPI 100 4>, <
    GIC_SPI 102 4>, <GIC_SPI 103 4>;
        interrupt-names = "IRQGP", "IRQGPMMU", "IRQPP0", "IRQPPMMU0", "IRQPP1", "IRQPPMMU1
    ";
        clocks = <&clk_pll_gpu>, <&clk_gpu>;
    };
```

Mali-400 的 sys_config.fex 配置信息如下：

```
;-----------------------------------------------------------------------------------
;GPU parameters
;regulator_id     : the regulator id GPU used.
;dvfs_status      : dvfs status, if this is enabled, DVFS will work.
;temp_ctrl_status : temperature control status, if this is enabled, the gpu frequency
;                   will drop down if the temperature of gpu is too high.
;scene_ctrl_status: scene control status, if this is enabled, android layer can ask
;                   gpu driver to change frequency in certain scene.
;max_level        : maximum level, which is used when thermal system does not restrict
;                    GPU power consumption.
;begin_level      : the corresponding frequency and voltage will be used during GPU
;                   initialization.
;lv<x>_freq       : frequency in MHz of certain level.
;lv<x>_volt       : voltage in mV of certain level.
;-----------------------------------------------------------------------------------
[gpu]
dvfs_status      = 0       //是否打开dvfs
temp_ctrl_status  = 1       //是否打开温控
scene_ctrl_status = 0       //是否打开场景控制
```

说明

Mali-400** 对应的 **SDK** 版本较旧，部分配置信息在 **sys_config.fex** 中配置，**sys_config.fex** 文件的位置一般在 **device/config/chips/{IC}/configs/{BOARD}/sys_config.fex(IC** 为产品型号，如 **T509**，**BOARD** 为板型如 **perf1)

2.3.2 kernel menuconfig 配置说明

进入内核根目录，执行 make ARCH=arm64 menuconfig
GPU 配置路径如下：

Device Drivers —>
    Graphics support —>
        GPU support for sunxi —>
            (ge8300) The GPU type

配置界面如下图所示：

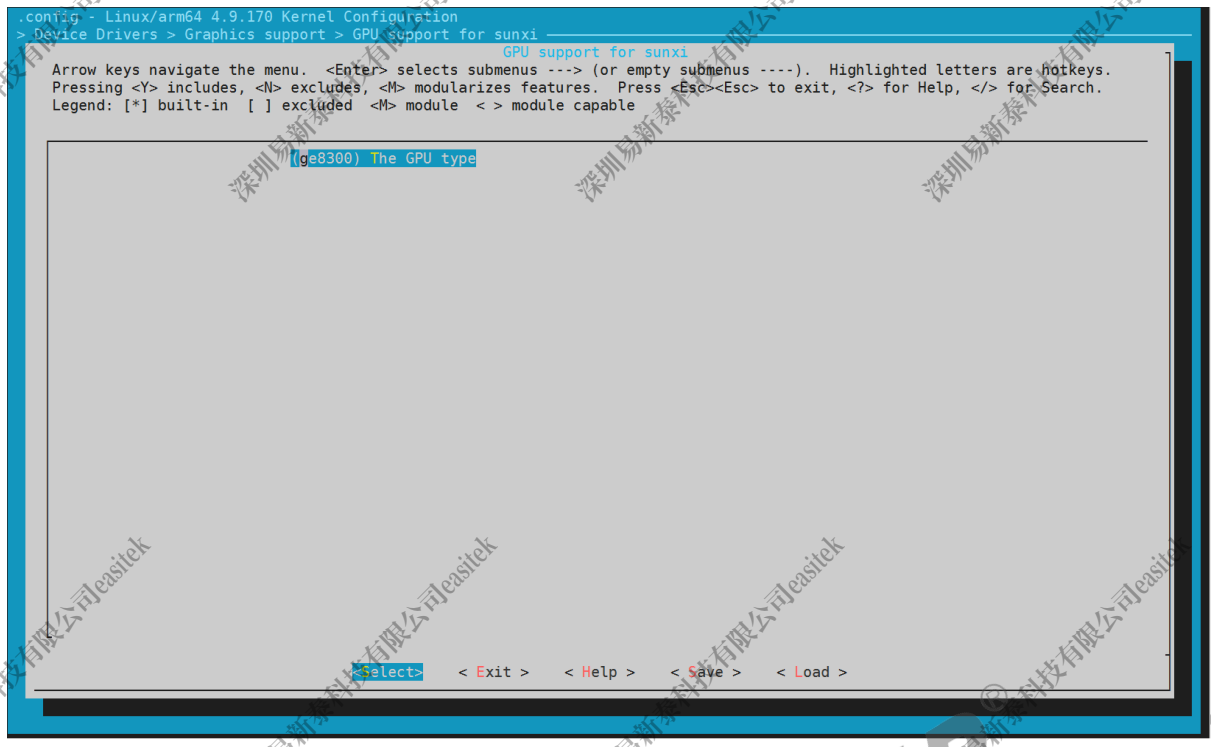

图 2-1: GPU 内核配置图

当需要编译 GPU 内核态驱动代码时，指定该项值为“ge8300”或“mali-g31”或“mali-400”即可使能相应的驱动编译，当不需要 GPU 的内核驱动时，将上述配置项的值修改为 None 即可。

2.4 驱动框架介绍

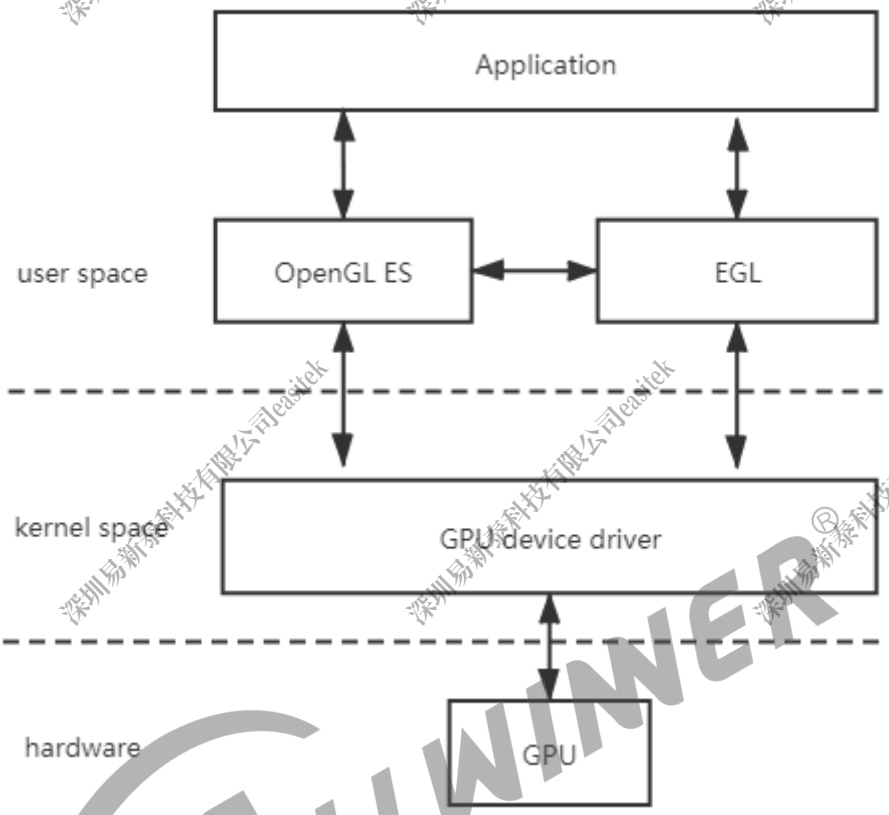

图 2-2: GPU 软件框架

针对 OpenGL ES 场景，GPU 软件框架如上图所示，应用通过调用 EGL 和 OpenGL ES 来使用 GPU 做硬件加速，而 EGL 和 OpenGL ES 则通过调用 GPU 内核态驱动（Device Driver）完成底层硬件操作。值得说明的是 EGL 和 OpenGL ES 代码闭源，以动态库 so 的形式提供。Vulkan、Open CL 等使用场景的代码框架与 OpenGL ES 场景基本一致，在此不赘述。

3 模块接口说明

应用使用 GPU 硬件加速主要是通过 OpenGL ES、Vulkan、Open CL 等接口来实现，有关这些接口的详细说明请参考以下链接：

https://www.khronos.org/registry/EGL/

https://www.khronos.org/registry/OpenGL/index_es.php

https://www.khronos.org/registry/OpenCL/

https://www.khronos.org/registry/vulkan/

4 模块使用范例

以下是一个使用 GPU 进行渲染的 OpenGL ES 程序（基于 fbdev）。

```
#include <stdio.h>
#include <EGL/egl.h>
#include <GLES2/gl2.h>
#include <GLES2/gl2ext.h>

int main(int argc, char **argv)
{
    EGLBoolean egl_ret;
    EGLContext context;
    EGLSurface window_surface;
    EGLConfig config;
    EGLint num_config;
    EGLDisplay dpy = eglGetDisplay(EGL_DEFAULT_DISPLAY);
    EGLint context_attribs[] = {
        EGL_CONTEXT_CLIENT_VERSION, 2,
        EGL_NONE
    };

    EGLint attrib_list[] = {
        EGL_RED_SIZE, 8,
        EGL_GREEN_SIZE, 8,
        EGL_BLUE_SIZE, 8,
        EGL_ALPHA_SIZE, 8,
        EGL_SURFACE_TYPE, EGL_WINDOW_BIT,
        EGL_RENDERABLE_TYPE, EGL_OPENGL_ES2_BIT,
        EGL_NONE
    };

    if (dpy == EGL_NO_DISPLAY) {
        printf("eglGetDisplay returned EGL_NO_DISPLAY\n");
        return -1;
    }

    egl_ret = eglInitialize(dpy, NULL, NULL);
    if (egl_ret != EGL_TRUE) {
        printf("eglInitialize failed\n");
        return -1;
    }

    egl_ret = eglChooseConfig(dpy, attrib_list, &config, 1, &num_config);
    if (egl_ret != EGL_TRUE) {
        printf("eglChooseConfig failed\n");
        return 0;
    }

    window_surface = eglCreateWindowSurface(dpy, config, (NativeWindowType)NULL, NULL);
    if (window_surface == EGL_NO_SURFACE) {
        printf("gelCreateWindowSurface failed\n");
        return 0;
```

```
    }

    context = eglCreateContext(dpy, config, EGL_NO_CONTEXT, context_attribs);
    if (context == EGL_NO_CONTEXT) {
        printf("eglCreateContext failed\n");
        return 0;
    }

    egl_ret = eglMakeCurrent(dpy, window_surface, window_surface, context);
    if (egl_ret != EGL_TRUE) {
        printf("eglMakeCurrent failed\n");
        return 0;
    }

    glClearColor(1.0f, 0.0f, 0.0f, 1.0f);

    glClear(GL_COLOR_BUFFER_BIT);

    eglSwapBuffers(dpy, window_surface);

    eglDestroySurface(dpy, window_surface);

    eglDestroyContext(dpy, context);

    eglTerminate(dpy);
}
```

5 FAQ

5.1 调试方法

5.1.1 调试工具

- Systrace
  Systrace 是 Android4.1 版本之后推出的，对系统性能分析的工具，实现原理是在系统的一些关键路径（比如 System Service, 虚拟机，Binder 驱动）插入一些信息收集，从而获取系统关键路径的运行时间信息，进而得到整个系统的运行性能信息。Systrace 的功能包括跟踪系统的 I/O 操作、内核工作队列、CPU 负载以及 Android 各个子系统的运行状况等。借助该工具能有效提升对显示绘制通路的分析调试效率，具体的使用说明可参照以下链接：
  https://developer.android.google.cn/studio/profile/systrace?hl=zh_cn
- PVRTrace
  PVRTrace 是 PowerVR 提供的用于性能分析、鉴定瓶颈、修改应用程序等功能的工具集 PowerVR Graphics Tools 中的一部分。PVRTrace 一般用于完成记录与分析操作，具体而言，PVRTrace 可以捕获 OpenGL ES 应用程序的 API 调用，方便溯源分析代码流程。通过 PVRTrace，可以抓取每个 OpenGL ES 调用接口及每一帧绘制的图像，并可将抓取的调用进行回放操作。具体的使用说明可参考以下链接：
  https://docs.imgtec.com/PVRTune_Manual/topics/pvrtune_introduction.html

5.1.2 调试节点

- /sys/kernel/debug/sunxi_gpu/dump
  该节点用于打印出当前 GPU 的状态信息，包括当前 GPU 的 idle 功能使能状态、DVFS 使能状态、是否独立供电、GPU 当前电压频率等。
  使用方法：

```
cat /sys/kernel/debug/sunxi_gpu/dump
```

著作权声明

商标声明

免责声明