低功耗方案

使用指南

文档版本　06

发布日期　2016-11-25

商标声明

 、HISILICON、海思和其他海思商标均为深圳市海思半导体有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

深圳市海思半导体有限公司

地址：深圳市龙岗区坂田华为基地华为总部 邮编：518129

网址：http://www.hisilicon.com

客户服务邮箱：support@hisilicon.com

前　言

概述

本文档主要基于“Hi3516A/Hi3516D/Hi3519V100/Hi3519V101 core 电源合并方案和功耗收益”文档中的推荐电源方案，介绍各个方案的降功耗策略、实现方法和软件 SDK 调试工具。

说明

本文以 Hi3516A 描述为例，未有特殊说明，Hi3516D 与 Hi3516A 一致，其他芯片与 Hi3516A 类似。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3516A	V100
Hi3516D	V100
Hi3519	V100
Hi3519	V101

读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

文档版本 06 (2016-11-25)

新增 Hi3519V101 内容。

文档版本 05 (2016-06-15)

2.2.2 小节，表 2-1 涉及修改。

文档版本 04 (2016-02-25)

第 1 章涉及修改。

第 2 章 SDK 调试

2.1 小节和 2.2.2 小节，涉及修改。

文档版本 03 (2015-05-29)

修改 2 路和 3 路电源域的相关内容；删除表 1-2，2.2.2 小节涉及修改；新增附录章节。

文档版本 02 (2015-02-10)

新增表 1-1 和 1-2，2.1.3 和 2.1.4 小节涉及修改。

文档版本 01 (2014-12-18)

第 1 次正式发布。

目 录

插图目录

表格目录

1 概述

电源域是芯片将对电压需求相近的一些逻辑模块采用同一电源来源，由此提供一个电压，使得芯片各个逻辑模块拥有专有的电压来源，各个逻辑模块电压频率按照定义好的算法独立进行合理的动态调整，能将芯片的功耗大幅度的降低。
Hi3516A/Hi3519V100/Hi3519V101 将芯片划分为 4 个区域，分别为 CORE 域，DDR 域，MEDIA 域，CPU 域，其中 Hi3519V100/Hi3519V101 的 CPU 域指的是 A17 核的供电域，用户可以根据自己的需求选择芯片的电源供压方案。

DVFS（Dynamic Voltage and Frequency Scaling）是根据运行场景的不同，动态设置不同的频率和电压水平来满足当前的电路时序和性能要求。DVFS 最大的特点是能根据运行不同业务 CPU 占用率的多少，快速切换到不同的电压和主频，保证最经济功耗运行。

AVS（Adaptive Voltage Scaling）是在 DVFS 的基础上，根据芯片工艺，温度，电路时序等情况，实时动态调整电压来进一步降低芯片的功耗。

SVB（Selective Voltage Binning）是在芯片上电的过程中，根据芯片工艺的不同，对满足场景的初始电压值进行微调，实现对芯片的降功耗的目的。

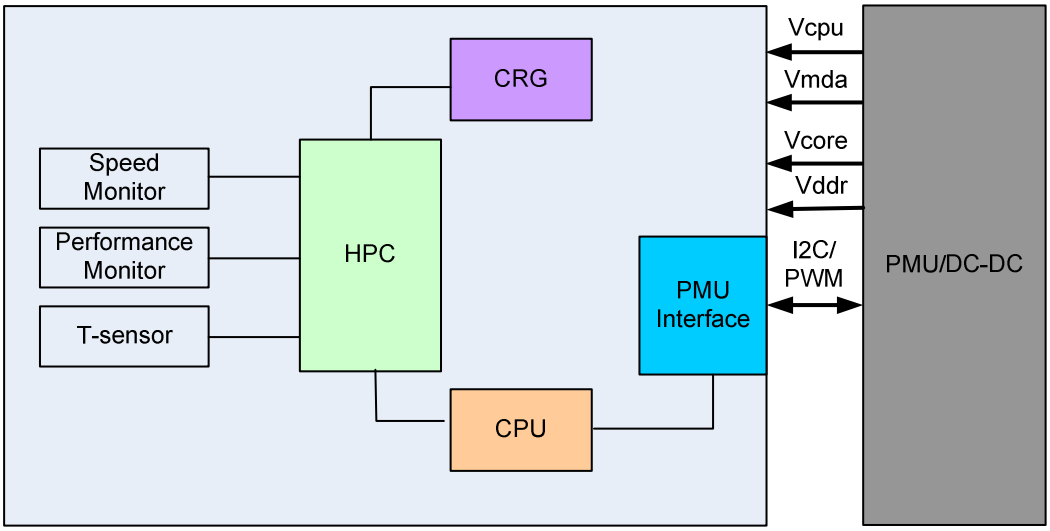
DVFS/AVS 能够保证在芯片性能不变的情况下，降低芯片运行时的功耗，如图 1-2 所示，HPC（Hardware Power Controller）根据预先设定的 AVS、DVFS 算法，依据 Speed Monitor、Performance Monitor 和 T-Sensor 的反馈数据，通过 PMU（Power Management Unit）Interface 控制外部 PMU/DC-DC，动态调节 CPU、DDR、MEDIA、CORE 的电压，从而达到降低整芯片平均功耗的效果。

hi35xxx_pm.ko 是低功耗调节内核模块，包含两个方面的功耗调节，一是 Media，二是 CPU。用户可以根据电源域的划分以及自身对功耗的调节来选择编译选项编译所需部分。加载模块时，如果编译了 Media 部分，用户可以设置模块参数 media_avs_en，media_avs_profile，media_avs_inter 来设置是否使能 Media 的 AVS，AVS 的 profile，以及 Media 的 AVS 检测间隔。如果编译了 CPU 部分，则 CPU 的 DVFS 自动开启，用户可以设置模块参数 cpu_avs_en，cpu_avs_inter 来设置是否能使 CPU 的 AVS，AVS 的调节间隔，各模块参数说明见表 1-1。

表1-1 低功耗模块参数说明

模块参数	media_avs_en	media_avs_profile	media_avs_inter	cpu_avs_en	cpu_avs_inter
Hi3516A	媒体域 AVS 使能，取值范围[0, 1]，默认为 1。	媒体域 profile。取值范围[0, 3]，默认为 3。 • 0：1080p@30fps • 1：3M@30fps • 2：1080p@60fps • 3：5M@30fps	媒体域 AVS 检测的时间间隔。默认 20ms。	CPU 域 AVS 使能，取值范围[0, 1] ，默认为 1。	CPU 域 AVS 检测的时间间隔。默认 20ms。
Hi3519V100/Hi3519V101	媒体域 AVS 使能，取值范围[0, 1] ，默认为 1。	不支持	媒体域 AVS 检测的时间间隔。默认 20ms。	CPU 域 AVS 使能，取值范围[0, 1] ，默认为 1。	CPU 域 AVS 检测的时间间隔。默认 20ms。

图1-2 DVFS、AVS 示意图



各芯片上电时，CPU 域独立供电时默认的工作电压与频率见表 1-2。

表1-2 CPU 域默认的工作电压与频率

默认值	电压	频率
Hi3516A	1.1V	600M

默认值	电压	频率
Hi3519V100	0.94V（A17 频率不大于 880M） 1.0V（A17 频率大于 880M）	880M
Hi3519V101	0.94V（A17 频率不大于 800M） 1.0V（A17 频率不大于 1000M） 1.07V（A17 频率大于 1000M）	930M

各芯片支持的电源域划分，如表 1-3 所示。

表1-3 各芯片支持的电源域

电源域 芯片	单路电源域	2 路电源域(media 独立)	2 路电源域(cpu 独立)	3 路电源域	4 路电源域
Hi3516A	支持	支持	支持	支持	支持
Hi3519V100	不支持	支持	不支持	支持	支持
Hi3519V101	不支持	不支持	不支持	支持	不支持

1. 单路电源域

   单路电源域的低功耗方案主要通过对该芯片实现 SVB 处理进行操作。给定芯片默认的最大处理能力即可在芯片的启动过程中完成该降功耗处理过程。

   - Hi3516A 寄存器配置如下，详见表 1-4。
     1) 配置 uboot 表格寄存器[**0x2005015C**]的[3:0]值为 0x0，将电源域模式设置为单路电源模式；
     2) 配置 uboot 表格寄存器[**0x2005015C**]的[7:4]值为 0x0，配置芯片默认的最高工作电压 1.1V；
     3) 配置 uboot 表格寄存器[0x20030000]和[0x20030004]分别为 0x12000000 和 0x01501032，设置 CPU 的默认工作频率 600M。
   - Hi3519V100/ Hi3519V101 不支持单路电源域方案。

2. 2 路电源域

   2 路电源域即对芯片提供两个电源来源。该方案下，对合并供电的电源采取 SVB 的低功耗手段，对独立电源域（如 CPU 或者 MEDIA）采用 AVS、DVFS 或者其他的降功耗手段。Hi3519V100 只支持 MEDIA 单独供电的方案。

   两路电源域合并方案包含两种方案，如下：

   - CPU、VDD 和 DDR 合并供电，MEDIA 单独供电。

     Hi3516A 的寄存器配置，如表 1-4 所示：

1) 配置 uboot 表格寄存器[**0x2005015C**]的[3:0]值为 0x1，将电源域模式设置为 2 路电源模式（MEDIA 域单独供电）；
2) 配置 uboot 表格寄存器[**0x2005015C**]的[7:4]值为 0x0， 配置芯片默认的最高工作电压 1.1V；
3) 配置 uboot 表格寄存器**[0x20030000]**和**[0x20030004]**分别为 0x12000000 和 0x01501032，设置 CPU 的默认工作频率 600M。

完成以上步骤，Hi3516A 的两个电源域均可以在启动过程中进行 SVB 的降功耗处理。

Hi3519V100 的寄存器配置如下：

1) 配置 uboot 表格寄存器[0x1202015C]的[1:0]值为 0x2，将电源域模式设置为 2 路电源模式（MEDIA 域单独供电）
2) 配置 uboot 表格寄存器[0x120A000C]的值为 0x4800c7， 配置芯片 MEDIA 域的工作电压为 1.0V；
3) 配置 uboot 表格寄存器[0x12010004]的值为 0x0b118370，配置 uboot 表格寄存器[0x12010034]的[6:4]的值为 0x1，设置 A17 核的默认工作频率为 880M。

Hi3519V100 在 2 路电源域下对合并电源域不做 SVB。

- MEDIA、VDD 和 DDR 合并供电，CPU 单独供电。

Hi3516A 的寄存器配置如下，详见表 1-4。

1) 配置 uboot 表格寄存器[**0x2005015C**]的[3:0]值为 0x2，将电源域模式设置为 2 路电源模式（CPU 域单独供电）；
2) 配置 uboot 表格寄存器[**0x2005015C**]的[7:4]值为 0x0， 配置芯片默认的最高工作电压 1.1V；
3) 配置 uboot 表格寄存器[0x20030000]和[0x20030004]分别为 0x12000000 和 0x01501032，设置 CPU 的默认工作频率 600M；

如果需要将 CPU 的频率设置为其他频点，则需要同步的将最高工作电压设置为频点对应的电压，设置后 CPU 的默认电压会调节到设置电压，而 MEDIA、VDD 和 DDR 合并供电的电压仍旧保持 1.1V 的默认处理，具体详见表 1-4。

Hi3519V100 不支持该电源域方案。

- Hi3519V101 不支持 2 路电源域方案

3. 3 路电源域

3 路电源域分别为 VDD 和 DDR 合并供电、MEDIA 独立供电、CPU 独立供电三个电源域。这种方案下，合并供电电源域只能进行 uboot 下的 SVB 的降功耗策略，而两个独立供电区能分别进一步进行降功耗调节。

- Hi3516A 寄存器配置如下，详见表 1-4。

1) 配置 uboot 表格寄存器[**0x2005015C**]的[3:0]值为 0x3，将电源域模式设置为 3 路电源模式；
2) 配置 uboot 表格寄存器[**0x2005015C**]的[7:4]值为 0x0， 配置芯片默认的最高工作电压 1.1V；
3) 配置 uboot 表格寄存器**[0x20030000]**和**[0x20030004]**分别为 0x12000000 和 0x01501032，设置 CPU 的默认工作频率 600M；

- Hi3519V100 寄存器配置如下：
  1) 配置 uboot 表格寄存器[0x1202015C]的[1:0]值为 0x1，将电源域模式设置为 3 路电源模式
  2) 配置 uboot 表格寄存器[0x120A000C]的值为 0x4800c7， 配置芯片 MEDIA 域的工作电压为 1.0V；
  3) 配置 uboot 表格寄存器[0x120A0004]的值为 0x6500c7， 配置芯片 CPU（A17 核）域的工作电压为 0.92V，若需要配置为 1.0V 则配置寄存器为 0x4800c7；
  4) 配置 uboot 表格寄存器[0x12010004]的值为 0x0b118370，配置 uboot 表格寄存器[0x12010034]的[6:4]的值为 0x1，设置 A17 核的默认工作频率为 880M。

     完成以上步骤，芯片的三个电源域均可以在启动过程中进行 SVB 的降功耗处理。
- Hi3519V101 寄存器配置如下：
  1) 配置 uboot 表格寄存器[0x1202015C]的[1:0]值为 0x1 或 0x0，将电源域模式设置为 3 路电源模式
  2) 配置 uboot 表格寄存器[0x120A000C]的值为 0x4800c7，配置芯片 MEDIA 域的工作电压为 1.0V；
  3) 配置 uboot 表格寄存器[0x120A0004]的值为 0x6500c7， 配置芯片 CPU（A17 核）域的工作电压为 0.92V，若需要配置为 1.0V 则配置寄存器为 0x4800c7；
  4) 配置 uboot 表格寄存器[0x12010004]的值为 0x0b1183a2，配置 uboot 表格寄存器[0x12010034]的[6:4]的值为 0x1，设置 A17 核的默认工作频率为 930M。

     完成以上步骤，芯片的三个电源域均可以在启动过程中进行 SVB 的降功耗处理。

4. 4 路电源域

   4 路电源域下每个电源域都是单独供电的，但是 VDD 和 DDR 域仅支持 uboot 下的 SVB 降功耗策略。

   - Hi3516A 寄存器配置如下，详见表 1-4。
     1) 配置 uboot 表格寄存器[**0x2005015C**]的[3:0]值为 0x4，将电源域模式设置为 4 路电源模式；
     2) 配置 uboot 表格寄存器[**0x2005015C**]的[7:4]值为 0x0， 配置芯片默认的最高工作电压 1.1V；
     3) 配置 uboot 表格寄存器[0x20030000]和[0x20030004]分别为 0x12000000 和 0x01501032，设置 CPU 的默认工作频率 600M；
   - Hi3519V100 寄存器配置如下：
     1) 配置 uboot 表格寄存器[0x1202015C]的[1:0]值为 0x0，将电源域模式设置为 4 路电源模式
     2) 配置 uboot 表格寄存器[0x120A000C]的值为 0x4800c7， 配置芯片 MEDIA 域的工作电压为 1.0V；

3) 配置 uboot 表格寄存器[0x120A0004]的值为 0x6500c7， 配置芯片 CPU 域（A17 核）的工作电压为 0.92V，若需要配置为 1.0V 则配置寄存器为 0x4800c7；

4) 配置 uboot 表格寄存器[0x12010004]的值为 0x0b118370，配置 uboot 表格寄存器[0x12010034]的[6:4]的值为 0x1，设置 A17 核的默认工作频率为 880M。

- Hi3519V101 不支持 4 路电源域方案。

对于 CPU 和 MEDIA 为独立电源域的情况，我们可以加载降功耗 KO（hi35xx_pm.ko）进一步来降低功耗，通过设置支持的模块参数（见表 1-1）来调整降功耗模块的行为。CPU 的 DVFS 默认的调节方案为 ondemand，该方案根据 CPU 负载动态调频调压，若需要修改，请参见 2.2 “设置低功耗参数”。

注意

- Hi3519V100 A17 核默认频率是 880M，用户可以自行设置，当频率大于 880M 时对应的 CPU（A17 核）域电压需要设置为 1.0V，否则设置为 0.92V。
- Hi3519V101 A17 核默认频率是 930M，用户可以自行设置，相应的频率对应的电压请参考表 1-2。

表1-4 Hi3516A 电源域，电压 CPU 频率寄存器配置

频率	寄存器	单路电源域	2 路电源域(media 独立)	2 路电源域(cpu 独立)	3 路电源域	4 路电源域
600M (1.1V)	SYSBOOT11 **0x2005015C**	[3:0]=0x0 [7:4]=0x0	[3:0]=0x1 [7:4]=0x0	[3:0]=0x2 [7:4]=0x0	[3:0]=0x3 [7:4]=0x0	[3:0]=0x4 [7:4]=0x0
	PERI_CFG0 **0x20030000**	0x12000000				
	PERI_CFG1 **0x20030004**	0x01501032				
732M (1.2V)	SYSBOOT11 **0x2005015C**	[3:0]=0x0 [7:4]=0x1	[3:0]=0x1 [7:4]=0x1	[3:0]=0x2 [7:4]=0x1	[3:0]=0x3 [7:4]=0x1	[3:0]=0x4 [7:4]=0x1
	PERI_CFG0 **0x20030000**	0x11000000				
	PERI_CFG1 **0x20030004**	0x015182dc				
850M (1.3V)	SYSBOOT11 **0x2005015C**	[3:0]=0x0 [7:4]=0x2	[3:0]=0x1 [7:4]=0x2	[3:0]=0x2 [7:4]=0x2	[3:0]=0x3 [7:4]=0x2	[3:0]=0x4 [7:4]=0x2
	PERI_CFG0	0x11000000				

频率	寄存器	单路电源域	2 路电源域(media 独立)	2 路电源域(cpu 独立)	3 路电源域	4 路电源域
	0x20030000					
	PERI_CFG1 **0x20030004**	0x01518352				

2 SDK 调试

hi35xxx_pm.ko 加载后，可使用本章的方法进行低功耗信息的查询或者调试。

2.1 查看低功耗信息

Hi3516A 的 cpu 是单核，在查看相关信息的时候以 cpu0 为准，Hi3519V100/Hi3519V101 是 big-little 架构的，其中只有大核进行低功耗调节，在查看相关信息时以 cpu1 为准。

2.1.1 查看 CPU 频率电压信息

在串口下输入如下命令：

```
cat /sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_cur_freq
```

如果串口打印：

```
600
```

那么，代表当前 CPU 频率为 600MHz。

在串口下输入如下命令（该命令的 regulator.1 中的数字会根据加载次数不同而变化）：

```
cat /sys/class/regulator/regulator.1/name
```

如果串口打印：

```
regulator_cpu
```

同时输入命令：

```
cat /sys/class/regulator/regulator.1/microvolts
```

- 对于 Hi3516A，如果串口打印：

```
1029
```

那么，代表当前 CPU 电压为 `1029MV`。

- 对于 Hi3519V100/Hi3519V101，如果串口打印：

```
878159
```

那么，代表当前 CPU 电压为 `0.878189V`。

2.1.2 查看 MEDIA 电压信息

在串口输入如下命令：

```
cat /sys/class/regulator/regulator.2/name
```

如果串口打印：

```
regulator_media
```

同时输入命令：

```
cat /sys/class/regulator/regulator.2/microvolts
```

- 对于 Hi3516A，如果串口打印：

```
983
```

那么，代表当前 MEDIA 电压为 983`MV`。

- 对于 Hi3519V100/Hi3519V101，如果串口打印：

```
880448
```

那么，代表当前 MEDIA 电压为 0.880448`V`。

2.1.3 查看 CPU 支持的低功耗策略

在串口输入如下命令：

```
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_governors
```

串口可能打印：

```
conservative ondemand userspace powersave interactive performance
```

以上策略基本说明如下：

- conservative：保守策略，逐级调整频率和电压；
- ondemand：当前默认使用的策略，根据 CPU 负载动态调频调压，比 interactive 策略反应慢；
- userspace：用户自己设置电压和频率，SDK 不会自动调整；
- powersave：功耗优先，始终将频率设置在最低值；
- interactive：根据 CPU 负载动态调频调压；
- performance：性能优先，始终将频率设置为最高值。

2.1.4 查看 CPU 使用的低功耗策略

在串口输入如下命令：

```
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

如果串口打印：

```
ondemand
```

那么表示当前使用的是 ondemand 的低功耗调整策略。

2.2 设置低功耗参数

2.2.1 设置 CPU 低功耗策略

- 在串口中输入如下命令(xxx 为策略名称)：
  `echo xxx > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor`
- 如果要关闭 CPU 动态调频调压，需要将低功耗策略设置为 userspace：
  `echo userspace > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor`
- 如果需要重新打开 CPU 动态调频调压，则需要将低功耗策略设置为 ondemand：
  `echo ondemand > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor`

2.2.2 设置 CPU 频率

目前各芯片 CPU 支持的频点如表 2-1 所示：

表2-1 各芯片 CPU 支持频点

芯片	频点单位：
Hi3516A/Hi3516D	单位：MHz 频率：400，500，600，732，850。
Hi3519V100（大核）	单位：KHz 频率：594000，792000，880000，1000000，1150000。
Hi3519V101（大核）	单位：KHz 频率：594000，792000，930000，1000000，1150000，1250000。

说明

因 Linux 版本差异，Hi3519V100/ Hi3519V101 与 Hi3516A/Hi3516D 频率单位不一样。

如果想手动设置 CPU 频率到指定频点，需要先关闭 CPU 动态调频调压：

```
echo userspace > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

然后设置命令：

```
echo 600 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
```

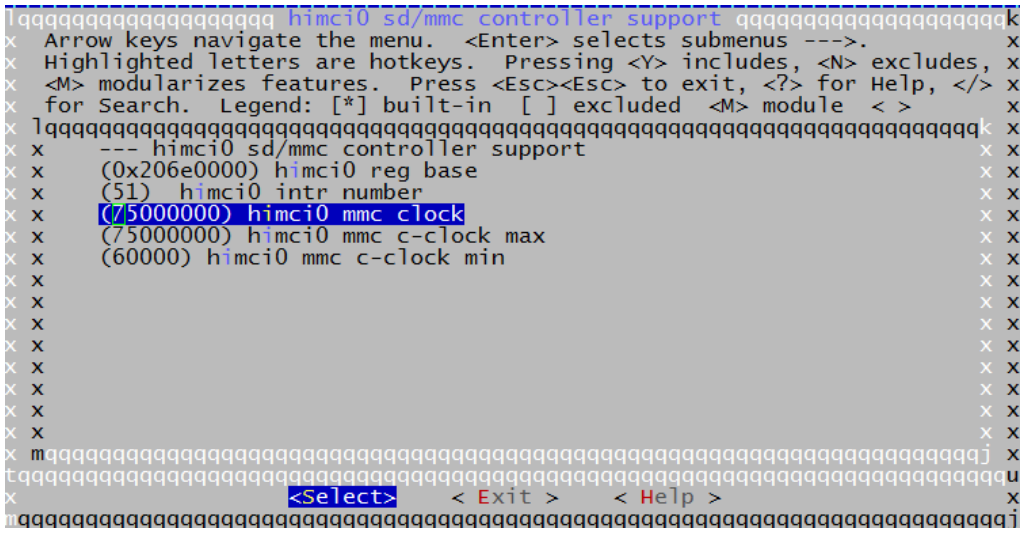
其中参数 600 单位为 MHz，即设置 CPU 频率为 600MHz。请注意，此时 CPU 的电压也会同步调整为预先设置好的支持此频率的稳定电压。

3 附录

3.1 注意事项

Hi3516A 芯片的 SDIO 控制器处在 Media 域，在对 Media 域进行动态调频调压时，SDIO 在 100MHz 下工作不稳定，故在打开 DVFS/AVS 情况下，必须将控制器参考时钟频率降为 75MHz。具体修改方法如图 3-1 所示。

图3-1 SDIO 控制器时钟配置



另外，若控制器默认时钟选择了 75M，则在卡接口时钟适配过程中，当卡支持时钟小于 75M 时需要分频已获得更小的时钟频率。例如，当卡支持时钟处在[50M, 75M)区间时，接口时钟由 75M 经过一次分频变成了 37.5M，直接跨过了 50M 这一档。为了提高读写性能，在这种情况下可不做分频处理，而是直接将控制器时钟配置成 50M。对应的代码修改方法如图 3-2 所示。

在函数 hi_mci_set_cclk()中，判断当卡时钟 cclk 设置到[50M, 75M)区间时，重新配置控制器时钟频率到 50M，并将分频系数 reg_value 配成 0，如图中红色部分。

图3-2 50MHz 时钟配置

```
    /*
     * set card clk divider value,
     * clk_divider = Fmmcclk/(Fmmc_cclk * 2)
     */
    if (0 == host->id) {
#ifdef CONFIG_HIMCI0
        if (CONFIG_MMC0_CLK <= cclk)
            reg_value = 0;
        else {
            reg_value = CONFIG_MMC0_CLK / (cclk * 2);
            if (CONFIG_MMC0_CLK % (cclk * 2))
                reg_value++;
            if ((50000000 <= cclk) && (75000000 > cclk)) {
                crg_value = himci_readl(PERI_CRG49);
                crg_value &= ~(SDIO0_CLK_BIT_HIGH);
                crg_value &= ~(SDIO0_CLK_BIT_LOW);
                crg_value |= SDIO0_CLK_SEL_50M;
                crg_value |= SDIO0_CKEN;
                himci_writel(crg_value, PERI_CRG49);
                reg_value = 0;
            }
        }
#endif
    } else if (1 == host->id) {
#ifdef CONFIG_HIMCI1
        if (CONFIG_MMC1_CLK <= cclk)
            reg_value = 0;
        else {
            reg_value = CONFIG_MMC1_CLK / (cclk * 2);
            if (CONFIG_MMC1_CLK % (cclk * 2))
                reg_value++;
            if ((50000000 <= cclk) && (75000000 > cclk)) {
                crg_value = himci_readl(PERI_CRG49);
                crg_value &= ~(SDIO1_CLK_BIT_HIGH);
                crg_value &= ~(SDIO1_CLK_BIT_LOW);
                crg_value |= SDIO1_CLK_SEL_50M;
                crg_value |= SDIO1_CKEN;
                himci_writel(crg_value, PERI_CRG49);
                reg_value = 0;
            }
        }
#endif
```

```
    } else {
        himci_error("himci host id error!");
        return;
    }
```