

CPUFreq开发指南

发布版本：1.0

作者邮箱：finley.xiao@rock-chips.com

日期：2018.12.04

文档密级：公开资料

前言

概述

主要描述CPUFreq的相关概念、配置方法和用户态接口。

产品版本

芯片名称	内核版本
所有芯片	Linux4.4

读者对象

软件开发工程师

技术支持工程师

修订记录

日期	版本	作者	修订说明
2018-12-04	V1.0	肖锋	初始版本

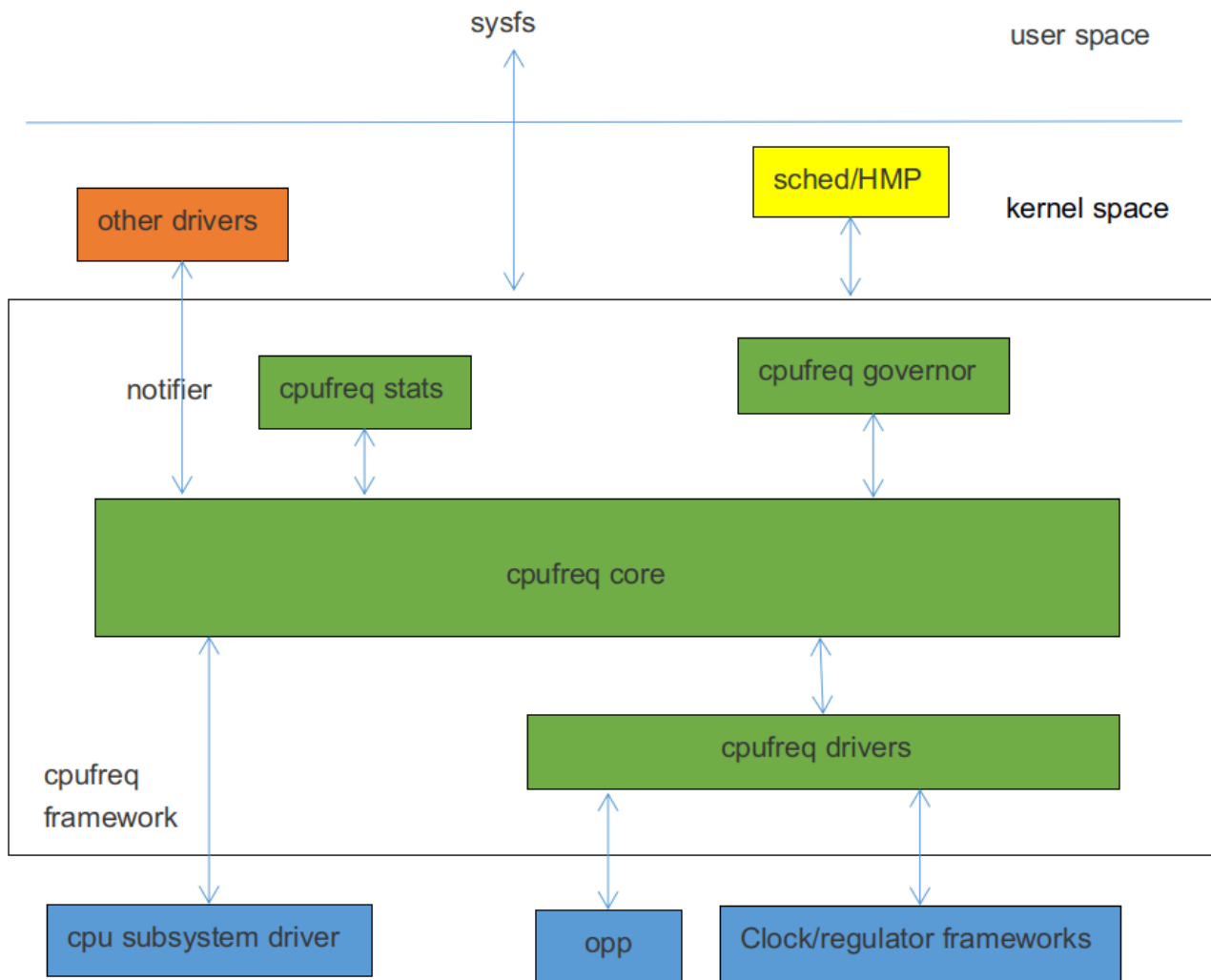
CPUFreq开发指南

- 1 概述
- 2 代码路径
- 3 配置方法
 - 3.1 Menuconfig配置
 - 3.2 Clock配置
 - 3.3 Regulator配置
 - 3.4 OPP Table配置
 - 3.4.1 增加OPP Table
 - 3.4.2 删除OPP
 - 3.5 根据leakage调整OPP Table
 - 3.5.1 根据leakage调整电压
 - 3.6 根据PVTM调整OPP Table

- 3.6.1 根据PVTM调整电压
 - 3.7 根据IR-Drop调整OPP Table
 - 3.8 宽温配置
- 4 用户态接口介绍
- 5 常见问题
 - 5.1 各平台CPU的最高
 - 5.2 如何查看频率电压表
 - 5.3 如何修改电压
 - 5.4 如何定频
 - 5.5 如何查看当前频率
 - 5.6 如何查看当前电压
 - 5.7 如何单独调频调压
 - 5.8 如何查看当前电压的档位
 - 5.9 如何查看leakage

1 概述

CPUFreq是内核开发者定义的一套支持根据指定的governor动态调整CPU频率和电压的框架模型，它能有效地降低CPU的功耗，同时兼顾CPU的性能。CPUFreq framework由governor、core、driver、stats组成，软件架构如下：



CPUFreq governor: 用于CPU升降频检测，根据系统负载，决定CPU频率。目前Linux4.4内核中包含了如下几种governor:

- conservative: 根据CPU负载动态调频，按一定的比例平滑的升高或降低频率。
- ondemand: 根据CPU负载动态调频，调频幅度比较大，可直接调到最高频或最低频。
- interactive: 根据CPU负载动态调频，相比ondemand，响应时间更快，可配置参数更多，更灵活。
- userspace: 提供相应接口供用户态应用程序调整频率。
- powersave: 功耗优先，始终将频率设置在最低值。
- performance: 性能优先，始终将频率设置为最高值。
- schedutil: EAS使用governor。EAS（Energy Aware Scheduling）是新一代的任务调度策略，结合CPUFreq和CPUIidle的策略，在为某个任务选择运行CPU时，同时考虑了性能和功耗，保证了系统能耗最低，并且不会对性能造成影响。Schedutil调度策略就是专门给EAS使用的CPU调频策略。

CPUFreq core: 对cpufreq governors和cpufreq driver进行了封装和抽象，并定义了清晰的接口。

CPUFreq driver: 用于初始化CPU的频率电压表，设置具体CPU的频率。

CPUFreq stats: 提供cpufreq有关的统计信息。

2 代码路径

Governor相关代码:

```
1 drivers/cpufreq/cpufreq_conservative.c      /* conservative调频策略 */
2 drivers/cpufreq/cpufreq_ondemand.c          /* ondemand调频策略 */
3 drivers/cpufreq/cpufreq_interactive.c       /* interactive调频策略 */
4 drivers/cpufreq/cpufreq_userspace.c         /* userspace调频策略 */
5 drivers/cpufreq/cpufreq_performance.c       /* performance调频策略 */
6 kernel/sched/cpufreq_schedutil.c           /* schedutil调频策略 */
```

Stats相关代码:

```
1 drivers/cpufreq/cpufreq_stats.c
```

Core相关代码:

```
1 drivers/cpufreq/cpufreq.c
```

Driver相关代码:

```
1 drivers/cpufreq/cpufreq-dt.c                /* platform driver */
2 drivers/cpufreq/rockchip-cpufreq.c          /* platform device */
3 drivers/soc/rockchip/rockchip_opp_select.c  /* 修改电压表相关接口 */
```

3 配置方法

3.1 Menuconfig配置

```
1 CPU Power Management --->
2   CPU Frequency scaling --->
3     [*] CPU Frequency scaling
4     <*> CPU frequency translation statistics /* cpufreq stats */
5     [ ] CPU frequency translation statistics details
6     [*] CPU frequency time-in-state statistics
7     Default CPUFreq governor (interactive) ---> /* cpufreq governor */
8     <*> 'performance' governor
9     <*> 'powersave' governor
10    <*> 'userspace' governor for userspace frequency scaling
11    <*> 'ondemand' cpufreq policy governor
12    -* 'interactive' cpufreq policy governor
13    <*> 'conservative' cpufreq governor
14    [ ] 'schedutil' cpufreq policy governor
15    *** CPU frequency scaling drivers ***
16    <*> Generic DT based cpufreq driver /* platform driver */
17    < > Generic ARM big LITTLE CPUfreq driver
18    <*> Rockchip CPUfreq driver /* platform device */
```

通过“Default CPUFreq governor”配置项，可以选择变频策略，开发者可以根据实际产品需求进行修改。

3.2 Clock配置

根据平台的实际情况，在CPU节点下增加“clock”属性，一般在DTSI文件中。Clock的详细配置说明，请参考clock相关的开发文档。

对于非大小核的平台，比如RK3326、RK3328等，在CPU0节点下增加“clocks = <&cru ARMCLK>;”，以RK3328为例：

```
1  cpu0: cpu@0 {
2      device_type = "cpu";
3      compatible = "arm,cortex-a53", "arm,armv8";
4      ...
5      clocks = <&cru ARMCLK>;
6  };
```

对于大小核的平台，如RK3368、RK3399等，在每个大核的CPU节点下增加“clocks = <&cru ARMCLKB>;”，在每个小核的CPU节点下增加“clocks = <&cru ARMCLKL>;”，以RK3399为例：

```
1  cpu_l10: cpu@0 {
2      device_type = "cpu";
3      compatible = "arm,cortex-a53", "arm,armv8";
4      ...
5      clocks = <&cru ARMCLKL>;
6  };
7
8  cpu_l11: cpu@1 {
9      device_type = "cpu";
10     compatible = "arm,cortex-a53", "arm,armv8";
11     ...
12     clocks = <&cru ARMCLKL>;
13 };
14
15 cpu_l12: cpu@2 {
16     device_type = "cpu";
17     compatible = "arm,cortex-a53", "arm,armv8";
18     ...
19     clocks = <&cru ARMCLKL>;
20 };
21
22 cpu_l13: cpu@3 {
23     device_type = "cpu";
24     compatible = "arm,cortex-a53", "arm,armv8";
25     ...
26     clocks = <&cru ARMCLKL>;
27 };
28
29 cpu_b0: cpu@100 {
30     device_type = "cpu";
31     compatible = "arm,cortex-a72", "arm,armv8";
```

```

32     ...
33     clocks = <&cru ARMCLKB>;
34 };
35
36 cpu_b1: cpu@101 {
37     device_type = "cpu";
38     compatible = "arm,cortex-a72", "arm,armv8";
39     ...
40     clocks = <&cru ARMCLKB>;
41 };

```

注意：如果clock没有配置，CPUFreq驱动加载失败，提示如下错误：

```

1  cpu cpu0: failed to get clock: -2
2  cpufreq-dt: probe of cpufreq-dt failed with error -2

```

3.3 Regulator配置

根据实际产品硬件使用的电源方案，在CPU节点下增加“cpu-supply”属性，一般在板级DTS文件中。

Regulator的详细配置说明，请参考Regulator和PMIC相关的开发文档。

对于非大小核的平台，在CPU0节点下增加“cpu-supply”属性，以RK3328为例：

```

1  &i2c1 {
2      status = "okay";
3      rk805: rk805@18 {
4          compatible = "rockchip,rk805";
5          status = "okay";
6          ...
7          regulators {
8              compatible = "rk805-regulator";
9              status = "okay";
10             ...
11             vdd_arm: RK805_DCDC2 {
12                 regulator-compatible = "RK805_DCDC2";
13                 regulator-name = "vdd_arm";
14                 regulator-init-microvolt = <1225000>;
15                 regulator-min-microvolt = <712500>;
16                 regulator-max-microvolt = <1450000>;
17                 regulator-initial-mode = <0x1>;
18                 regulator-ramp-delay = <12500>;
19                 regulator-boot-on;
20                 regulator-always-on;
21                 regulator-state-mem {
22                     regulator-mode = <0x2>;
23                     regulator-on-in-suspend;
24                     regulator-suspend-microvolt = <950000>;
25                 };
26             };
27             ...
28         };
29     };

```

```

30 };
31
32 &cpu0 {
33     cpu-supply = <&vdd_arm>;
34 };

```

对于大小核的平台，在每个CPU节点下增加“cpu-supply”属性，以RK3399为例：

```

1  &cpu_l0 {
2      cpu-supply = <&vdd_cpu_l>;
3  };
4
5  &cpu_l1 {
6      cpu-supply = <&vdd_cpu_l>;
7  };
8
9  &cpu_l2 {
10     cpu-supply = <&vdd_cpu_l>;
11 };
12
13 &cpu_l3 {
14     cpu-supply = <&vdd_cpu_l>;
15 };
16
17 &cpu_b0 {
18     cpu-supply = <&vdd_cpu_b>;
19 };
20
21 &cpu_b1 {
22     cpu-supply = <&vdd_cpu_b>;
23 };

```

注意：如果regulator没有配置，cpufreq驱动仍然可以加载成功，认为只调频不调压，频率比较高时，可能会因为电压偏低而出现死机的现象。

3.4 OPP Table配置

Linux4.4内核将频率、电压相关的配置放在了devicetree中，我们将这些配置信息组成的节点，称之为OPP Table。OPP Table节点包含描述频率和电压的OPP节点、leakage相关配置属性、PVTM相关配置属性等。OPP的详细配置说明，可以参考如下文档：

```

1 Documentation/devicetree/bindings/opp/opp.txt
2 Documentation/power/opp.txt

```

3.4.1 增加OPP Table

根据平台的实际情况，增加一个OPP Table节点，并在每个CPU节点下增加“operating-points-v2”属性，一般在DTSI文件中。以RK3328为例：

```

1  cpu0: cpu@0 {
2      device_type = "cpu";
3      compatible = "arm,cortex-a53", "arm,armv8";
4      ...
5      operating-points-v2 = <&cpu0_opp_table>;
6  };
7  cpu1: cpu@1 {
8      device_type = "cpu";
9      compatible = "arm,cortex-a53", "arm,armv8";
10     ...
11     operating-points-v2 = <&cpu0_opp_table>;
12 };
13 cpu2: cpu@2 {
14     device_type = "cpu";
15     compatible = "arm,cortex-a53", "arm,armv8";
16     ...
17     operating-points-v2 = <&cpu0_opp_table>;
18 };
19 cpu3: cpu@3 {
20     device_type = "cpu";
21     compatible = "arm,cortex-a53", "arm,armv8";
22     ...
23     operating-points-v2 = <&cpu0_opp_table>;
24 };
25
26 cpu0_opp_table: opp_table0 {
27     compatible = "operating-points-v2";
28     opp-shared;                                /* 表示该OPP Table是多个CPU共用的 */
29
30     /*
31     * 频转换因子，通过一定的算法转换成频率，表示该平台支持的最高频率，超过该频率的频点，会被删除。
32     * 比如13转换成频率后是1296MHz，那么OPP Table中超过1296MHz的频点都会被删除。
33     * 用于防止误填了该平台不支持的且较高的频率，一般不需要增加。
34     */
35     rockchip,avs-scale = <13>;
36
37     opp-408000000 {
38         opp-hz = /bits/ 64 <408000000>;        /* 单位Hz */
39         opp-microvolt = <950000 950000 1350000>; /* 单位uV，格式<target min max> */
40         clock-latency-ns = <40000>;            /* 完成变频需要的时间，单位ns */
41         /*
42         * 休眠，关闭整个大核的CPU或者关闭整个小核的CPU的时候，会将CPU频率设置为包含该属性的
43         * OPP所指定的频率。一个OPP Table中，只有一个OPP节点包含该属性。
44         */
45         opp-suspend;
46     };
47     ...
48     opp-1296000000 {
49         opp-hz = /bits/ 64 <1296000000>;
50         opp-microvolt = <1350000 1350000 1350000>;
51         clock-latency-ns = <40000>;
52     };
53 }

```


注意：如果operating-points-v2没有配置，cpufreq初始化失败，系统启动后无法进行调频调压，提示类似如下的错误：

```
1 | cpu cpu0: OPP-v2 not supported
2 | cpu cpu0: couldn't find opp table for cpu:0, -19
```

3.4.2 删除OPP

如果开发者需要删除某些频点，可以使用如下方法。

方法一：直接在对应该OPP节点下增加“status = "disabled";”，比如：

```
1 | cpu0_opp_table: opp_table0 {
2 |     compatible = "operating-points-v2";
3 |     opp-shared;
4 |
5 |     opp-408000000 {
6 |         opp-hz = /bits/ 64 <408000000>;
7 |         opp-microvolt = <950000 950000 1350000>;
8 |         clock-latency-ns = <40000>;
9 |     };
10 |     ...
11 |     opp-1296000000 {
12 |         opp-hz = /bits/ 64 <1296000000>;
13 |         opp-microvolt = <1350000 1350000 1350000>;
14 |         clock-latency-ns = <40000>;
15 |         status = "disabled";
16 |     };
17 | }
```

方法二：在板级DTS中重新引用OPP Table节点，并在对应OPP节点下增加“status = "disabled";”，比如：

```
1 | &cpu0_opp_table {
2 |     opp-1296000000 {
3 |         status = "disabled";
4 |     };
5 | };
```

3.5 根据leakage调整OPP Table

IDDQ(Integrated Circuit Quiescent Current)集成电路静止电流，指CMOS电路静态时从电源获取的电流，我们也称之为leakage。CPU的leakage指给CPU提供特定的电压，测得的静态电流值。在芯片生产过程中，会将leakage写到eFuse或者OTP中。

3.5.1 根据leakage调整电压

背景：通过测试芯片的Vmin，发现相同频率下，小leakage的芯片Vmin比较大，大leakage的芯片Vmin比较小，通过这一特性可以根据leakage值降低大leakage芯片的电压，以降低功耗和提高性能。

功能说明：从eFuse或OTP中获取该芯片的CPU leakage值，通过查表得到对应的档位，然后在每个OPP中选择对应档位的电压，作为该频点的电压。

配置方法：首先需要增加eFuse或者OTP的支持，具体方法请参考eFuse和OTP的相关文档。然后在OPP Table节点增加“rockchip,leakage-voltage-sel”、“nvmem-cells”和“nvmem-cell-names”三个属性，同时OPP节点根据实际情况增加“opp-microvolt-<name>”属性，这些配置一般都在DTSI文件中。以RK3328为例：

```
1  cpu0_opp_table: cpu0-opp-table {
2      compatible = "operating-points-v2";
3      opp-shared;
4
5      /*
6       * 从eFuse或OTP中获取CPU leakage值
7       */
8      nvmem-cells = <&cpu_leakage>;
9      nvmem-cell-names = "cpu_leakage";
10
11     /*
12     * leakage值为1mA-10mA的芯片，使用opp-microvolt-L0指定的电压
13     * leakage值为11mA-254mA的芯片，使用opp-microvolt-L1指定的电压
14     *
15     * 如果删除rockchip,leakage-voltage-sel属性或者leakage值不在该属性指定的范围内，
16     * 则使用opp-microvolt指定的电压。
17     */
18     rockchip,leakage-voltage-sel = <
19         1   10   0
20         11  254  1
21     >;
22
23     opp-408000000 {
24         opp-hz = /bits/ 64 <408000000>;
25         opp-microvolt = <950000 950000 1350000>;
26         opp-microvolt-L0 = <950000 950000 1350000>;
27         opp-microvolt-L1 = <950000 950000 1350000>;
28         clock-latency-ns = <40000>;
29         opp-suspend;
30     };
31     ...
32     opp-1296000000 {
33         opp-hz = /bits/ 64 <1296000000>;
34         opp-microvolt = <1350000 1350000 1350000>;
35         opp-microvolt-L0 = <1350000 1350000 1350000>;
36         opp-microvolt-L1 = <1300000 1300000 1350000>;
37         clock-latency-ns = <40000>;
38     };
39 };
```

如需关闭该项功能，可以删除“rockchip,leakage-voltage-sel”属性，这时使用opp-microvolt指定的电压。

3.6 根据PVTM调整OPP Table

CPU PVTM(Process-Voltage-Temperature Monitor)是一个位于CPU附近，能反应出不同芯片之间性能差异的模块，它受工艺、电压、温度的影响。

3.6.1 根据PVTM调整电压

背景：通过测试芯片的Vmin，发现相同频率和电压下，PVTM值小的芯片Vmin比较大，PVTM值大的芯片Vmin比较小，通过这一特性可以根据PVTM值降低大PVTM芯片的电压，以降低功耗和提高性能。

功能说明：在指定的电压和频率下获取PVTM值，并转换成参考温度下的PVTM值，然后查表得到对应的档位，最后在每个OPP中选择对应档位的电压，作为该频点的电压。

配置方法：首先需要先增加PVTM的支持，具体方法请参考PVTM的相关文档。然后在OPP Table节点增加“rockchip,pvtm-voltage-sel”、“rockchip,thermal-zone”和“rockchip,pvtm-<name>”属性，多种工艺的情况还需要增加“nvmem-cells”和“nvmem-cell-names”属性，OPP节点根据实际情况增加“opp-microvolt-<name>”属性。这些配置一般都在DTSI文件中。以RK3288为例：

```
1  cpu0_opp_table: opp_table0 {
2      compatible = "operating-points-v2";
3      opp-shared;
4
5      ...
6      /*
7       * 从eFuse或OTP中获取CPU工艺信息。
8       * 只有一种工艺的情况，可以不加；
9       * 包含多种工艺的情况，需要增加。
10     */
11     nvmem-cells = <&process_version>;
12     nvmem-cell-names = "process";
13
14     /*
15     * 只有一种工艺需要支持PVTM，需要增加rockchip,pvtm-voltage-sel属性，OPP节点也需要增加
16     * opp-microvolt-L0、opp-microvolt-L1等属性来区分电压；
17     *
18     * 多种工艺需要支持pvtm，比如有工艺0和工艺1，如果2种工艺配置不同，则需要增加
19     * rockchip,p0-pvtm-voltage-sel和rockchip,p1-pvtm-voltage-sel两个属性，
20     * 同时OPP节点也需要增加opp-microvolt-P0-L0、opp-microvolt-P1-L0等属性来区分电压；
21     * 如果2种工艺配置相同，也可以只增加rockchip,pvtm-voltage-sel属性。
22     *
23     * PVTM值为0-14300的芯片，使用opp-microvolt-L0指定的电压；
24     * PVTM值为14301-15000的芯片，使用opp-microvolt-L1指定的电压；
25     * PVTM值为15001-16000的芯片，使用opp-microvolt-L2指定的电压；
26     * PVTM值为16001-99999的芯片，使用opp-microvolt-L3指定的电压；
27     *
28     * 如果删除rockchip,pvtm-voltage-sel属性或者PVTM值不在该属性指定的范围内，
29     * 则使用opp-microvolt指定的电压。
30     */
31     rockchip,pvtm-voltage-sel = <
32         0          14300   0
33         14301     15000   1
34         15001     16000   2
```

```

35     16001     99999     3
36     >;
37     rockchip,pvtm-freq = <408000>;          /* 获取PVTM值前，需要先设置CPU频率，单位Khz */
38     rockchip,pvtm-volt = <1000000>;          /* 获取PVTM值前，需要先设置CPU电压，单位uV */
39     rockchip,pvtm-ch = <0 0>;                /* PVTM通道，格式<通道序号 sel的序号> */
40     rockchip,pvtm-sample-time = <1000>;      /* PVTM采样时间，单位us */
41     rockchip,pvtm-number = <10>;             /* PVTM采样个数 */
42     rockchip,pvtm-error = <1000>;            /* 允许采样数据之间的误差 */
43     rockchip,pvtm-ref-temp = <35>;           /* 参考温度 */
44     /* PVTM随温度变化的比例系数，格式 <小于参考温度的比例系数 大于参考温度的比例系数> */
45     rockchip,pvtm-temp-prop = <(-18) (-18)>;
46     rockchip,thermal-zone = "soc-thermal"; /* 通过哪个thermal-zone获取温度 */
47
48     opp-126000000 {
49         opp-hz = /bits/ 64 <126000000>;
50         opp-microvolt = <950000 950000 1350000>;
51         opp-microvolt-L0 = <950000 950000 1350000>;
52         opp-microvolt-L1 = <950000 950000 1350000>;
53         opp-microvolt-L2 = <950000 950000 1350000>;
54         opp-microvolt-L3 = <950000 950000 1350000>;
55         clock-latency-ns = <40000>;
56     };
57     ...
58     opp-1608000000 {
59         opp-hz = /bits/ 64 <1608000000>;
60         opp-microvolt = <1350000 1350000 1350000>;
61         opp-microvolt-L0 = <1350000 1350000 1350000>;
62         opp-microvolt-L1 = <1350000 1350000 1350000>;
63         opp-microvolt-L2 = <1300000 1300000 1350000>;
64         opp-microvolt-L3 = <1250000 1250000 1350000>;
65         clock-latency-ns = <40000>;
66     };
67 };

```

如需关闭该项功能，可以删除“rockchip,pvtm-voltage-sel”属性，这时使用opp-microvolt指定的电压。

3.7 根据IR-Drop调整OPP Table

IR-Drop是指出现在集成电路中电源和地网络上电压下降或升高的一种现象。在这里我们理解为由于电源纹、电路板布线等因素导致的压降。

背景：实测发现有些客户的板子电源纹波比较差，使用和EVB相同的电压表，某些频点的电压偏低，导致系统运行不稳定，这种情况需要根据IR-Drop调整调整OPP Table。

功能说明：将样机板每个频点的纹波减去EVB板的纹波，得到的差值就是该频点所需要增加的电压。

配置方法：需要在OPP Table节点增加“rockchip,max-volt”、“rockchip,evb-irdrop”和

“rockchip,board-irdrop”属性，其中“rockchip,board-irdrop”一般在板级DTS文件中配置，其他在DTSI文件中配置。以RK3326为例，DTSI中配置如下：

```

1  cpu0_opp_table: cpu0-opp-table {
2      compatible = "operating-points-v2";
3      opp-shared;
4
5      /* 允许设置的最高电压，单位uV */
6      rockchip,max-volt = <1350000>;
7      rockchip,evb-irdrop = <25000>; /* EVB板或者SDK板的电源纹波 */
8  }

```

板级DTS文件中配置如下：

```

1  &cpu0_opp_table {
2      /*
3       * max IR-drop values on different freq condition for this board!
4       */
5      /*
6       * 实际产品硬件，不同频率下的电源纹波情况：
7       * 0Mhz-815MHz，电源纹波为37500uV，最终电压会增加12500uV（37500-25000（evb板纹波））
8       * 816Mhz-1119MHz，电源纹波为50000uV，最终电压会增加25000uV（50000-25000（evb板纹波））
9       * 1200Mhz-1512MHz，电源纹波为75000uV，最终电压会增加50000uV（75000-25000（evb板纹波））
10     */
11     rockchip,board-irdrop = <
12     /*MHz    MHz    uV */
13         0      815    37500
14         816    1119    50000
15         1200   1512    75000
16     >;
17 };

```

如需关闭该项功能，可以删除“rockchip,board-irdrop”属性。

3.8 宽温配置

宽温通常指环境温度为-40~85℃。

背景：实测发现某些平台在低温环境下，运行不稳定，对某些频点抬压后可以稳定运行，这种情况需要根据温度调整电压表。实测也发现高温高压下芯片的寿命会缩短，也需要根据温度限制频率和电压。

功能说明：当系统检测到温度低于一定程度后，对各个频点进行抬压，如果某些频点的电压超过了系统允许设置的最高电压，这些频点将被限制，即运行过程中不会跑到这些频点。当温度恢复常温，电压表恢复成默认的状态。当系统检测到温度大于一定程度后，电压超过一定值的频点，将被限制。当温度恢复常温，解除频率限制。

配置方法：低温情况在OPP Table节点增加“rockchip,temp-hysteresis”、“rockchip,low-temp”、

“rockchip,low-temp-min-volt”、“rockchip,low-temp-adjust-volt”、“rockchip,max-volt”属性。高温情况在OPP Table节点增加“rockchip,temp-hysteresis”、“rockchip,high-temp”和“rockchip,high-temp-max-volt”属性。这些配置一般都在DTSI文件中。

```

1  cpu0_opp_table: opp_table0 {

```

```

2 compatible = "operating-points-v2";
3 opp-shared;
4
5 /*
6  * 迟滞参数, 单位millicelsius, 防止频繁进入低温或者高温
7  * 比如小于0度进入低温, 大于0+5度恢复常温, 大于85度进入高温, 低于85-5度恢复常温
8  */
9 rockchip,temp-hysteresis = <5000>;
10 rockchip,low-temp = <0>; /* 低温阈值, 单位millicelsius*/
11 rockchip,low-temp-min-volt = <900000>; /* 低温下最低电压, 单位uV */
12 rockchip,low-temp-adjust-volt = <
13     /* MHz      MHz      uV */
14     0          1800    25000 /* 低温下, 0-1800MHz内的频点, 电压增加25mV */
15 >;
16 /* 允许设置的最高电压, 单位uV */
17 rockchip,max-volt = <1250000>;
18
19 rockchip,high-temp = <85000>; /* 高温阈值, 单位millicelsius */
20 /* 高温下, 允许设置的最高电压, 单位uV, 超过该电压的频点, 会被限制 */
21 rockchip,high-temp-max-volt = <1200000>;
22 ...
23 }

```

4 用户态接口介绍

非大小核的平台, 如RK3288、RK3326、RK3328等, 所有CPU共用一个clock, 用户态接口也是相同的,

在/sys/devices/system/cpu/cpufreq/policy0/目录下。

大小核的平台, 如RK3368、RK3399等, 包含两个cluster, 每个cluster都有独立的clock和用户态接口, 比如cluster0是小核, 对应接口在/sys/devices/system/cpu/cpufreq/policy0/目录下, cluster1是大核, 对应的接口在/sys/devices/system/cpu/cpufreq/policy4/目录下。

通过用户态接口可以切换governor, 查看当前频率, 修改频率等, 具体如下:

```

1 related_cpus          /* 同个cluster下的所有CPU */
2 affected_cpus         /* 同个cluster下未关的CPU */
3 cpuinfo_transition_latency /* 两个不同频率之间切换所需要的时间, 单位ns */
4 cpuinfo_max_freq      /* CPU所支持的最高运行频率 */
5 cpuinfo_min_freq      /* CPU所支持的最低运行频率 */
6 cpuinfo_cur_freq      /* 硬件寄存器中读取CPU当前所处的运行频率 */
7 scaling_available_frequencies /* 系统支持的频率 */
8 scaling_available_governors /* 系统支持的变频策略 */
9 scaling_governor       /* 当前使用的变频策略 */
10 scaling_cur_freq       /* 软件上最后一次设置的频率 */
11 scaling_max_freq       /* 软件上限制的最高频率 */
12 scaling_min_freq       /* 软件上限制的最低频率 */
13 scaling_setspeed       /* 将governor切换为userspace才会出现, 可以通过该节点修改频率 */
14 stats/time_in_state    /* 记录CPU在各个频率下的运行时间, 单位: 10ms */
15 stats/total_trans      /* 记录CPU的变频次数 */
16 stats/trans_table      /* 记录CPU在每个频率上的变频次数 */

```

5 常见问题

5.1 各平台CPU的最高

产品名称	ARM核	最高主频
RK312x	4 * A7	1200MHz
RK322x	4 * A7	1464MHz
RK3288	4 * A17	1608MHz
RK3328	4 * A53	1296MHz
RK3368	4 * A53 + 4 * A53	1512MHz(big) + 1200MHz(little)
RK3399	2 * A72 + 4 * A53	1800MHz(big) + 1416MHz(little)

5.2 如何查看频率电压表

执行如下命令：

```
1 | cat /sys/kernel/debug/opp/opp_summary
```

以PX30为例：

1	device	rate(Hz)	target(uV)	min(uV)	max(uV)
2	-----				
3	cpu0				
4		408000000	950000	950000	1350000
5		600000000	950000	950000	1350000
6		816000000	1000000	1000000	1350000
7		1008000000	1125000	1125000	1350000
8		1200000000	1275000	1275000	1350000
9		1248000000	1300000	1300000	1350000
10		1296000000	1350000	1350000	1350000
11		1416000000	1350000	1350000	1350000
12		1512000000	1350000	1350000	1350000

5.3 如何修改电压

方法一：直接修改OPP节点中每个档位的电压。以CPU 816MHz抬压25000uV为例：

假设默认值如下：

```

1  opp-816000000 {
2      opp-hz = /bits/ 64 <816000000>;
3      opp-microvolt = <1075000 1075000 1350000>;
4      opp-microvolt-L0 = <1075000 1075000 1350000>;
5      opp-microvolt-L1 = <1050000 1050000 1350000>;
6      opp-microvolt-L2 = <1000000 1000000 1350000>;
7      opp-microvolt-L3 = <950000 950000 1350000>;
8      clock-latency-ns = <40000>;
9      opp-suspend;
10 };

```

修改后如下：

```

1  opp-816000000 {
2      opp-hz = /bits/ 64 <816000000>;
3      /* 单位uV, 格式<target min max>, 只需修改target和min, max为最高电压, 不需要修改 */
4      opp-microvolt = <1100000 1100000 1350000>;
5      opp-microvolt-L0 = <1100000 1100000 1350000>;
6      opp-microvolt-L1 = <107500 1075000 1350000>;
7      opp-microvolt-L2 = <1025000 1025000 1350000>;
8      opp-microvolt-L3 = <975000 975000 1350000>;
9      clock-latency-ns = <40000>;
10     opp-suspend;
11 };

```

方法二：通过修改IR-Drop的配置调整电压，具体参考3.7章节的介绍。以CPU 408MHz以下的频率全部抬压25000uV为例。

假设IR-Drop默认配置如下：

```

1  &cpu0_opp_table {
2      /*
3       * max IR-drop values on different freq condition for this board!
4       */
5      /*
6       * 实际产品硬件，不同频率下的电源纹波情况：
7       * 0Mhz-815MHz，电源纹波为37500uV，最终电压会增加12500uV（37500-25000（evb板纹波））
8       * 816Mhz-1119MHz，电源纹波为50000uV，最终电压会增加25000uV（50000-25000（evb板纹波））
9       * 1200Mhz-1512MHz，电源纹波为75000uV，最终电压会增加50000uV（75000-25000（evb板纹波））
10     */
11     rockchip,board-irdrop = <
12     /*MHZ    MHZ        uV */
13         0      815      37500
14         816    1119      50000
15         1200   1512      75000
16     >;
17 };

```

修改后如下：

```

1  &cpu0_opp_table {

```



```

2      /*
3      * max IR-drop values on different freq condition for this board!
4      */
5      /*
6      * 实际产品硬件，不同频率下的电源纹波情况：
7      * 0Mhz-408MHz，电源纹波为62500uV，最终电压会增加37500uV（62500-25000（evb板纹波））
8      * 409Mhz-815MHz，电源纹波为37500uV，最终电压会增加12500uV（37500-25000（evb板纹波））
9      * 816Mhz-1119MHz，电源纹波为50000uV，最终电压会增加25000uV（50000-25000（evb板纹波））
10     * 1200Mhz-1512MHz，电源纹波为75000uV，最终电压会增加50000uV（75000-25000（evb板纹波））
11     */
12     rockchip,board-irdrop = <
13     /*MHz    MHz    uV */
14         0      408    62500 /* 408MHz以下的频率，由原来的37500变为63500 */
15         409    815    37500
16         816    1119   50000
17         1200   1512   75000
18     >;
19 };

```

5.4 如何定频

方法一：在menuconfig中将governor设置为userspace。开机后CPU频率为CRU节点中设置频率。

方法二：将OPP Table中不想要的频率全部disable掉，只留一个想要的频率。以RK3308为例，CPU定频1008MHz的配置如下：

```

1  cpu0_opp_table: cpu0-opp-table {
2      compatible = "operating-points-v2";
3      opp-shared;
4
5      opp-408000000 {
6          opp-hz = /bits/ 64 <408000000>;
7          opp-microvolt = <950000 950000 1340000>;
8          clock-latency-ns = <40000>;
9          opp-suspend;
10         status = "disabled";
11     };
12     opp-600000000 {
13         opp-hz = /bits/ 64 <600000000>;
14         opp-microvolt = <950000 950000 1340000>;
15         clock-latency-ns = <40000>;
16         status = "disabled";
17     };
18     opp-816000000 {
19         opp-hz = /bits/ 64 <816000000>;
20         opp-microvolt = <1025000 1025000 1340000>;
21         clock-latency-ns = <40000>;
22         status = "disabled";
23     };
24     opp-1008000000 {
25         opp-hz = /bits/ 64 <1008000000>;
26         opp-microvolt = <1125000 1125000 1340000>;

```

```

27         clock-latency-ns = <40000>;
28     };
29     opp-1200000000 {
30         opp-hz = /bits/ 64 <1200000000>;
31         opp-microvolt = <1250000 1250000 1340000>;
32         clock-latency-ns = <40000>;
33         status = "disabled";
34     };
35     opp-1296000000 {
36         opp-hz = /bits/ 64 <1296000000>;
37         opp-microvolt = <1300000 1300000 1340000>;
38         clock-latency-ns = <40000>;
39         status = "disabled";
40     };
41 };

```

方法三：开机后通过命令定频。

非大小核平台，比如RK3288，执行如下命令：

```

1  /* 切换governor到userspace */
2  echo userspace > /sys/devices/system/cpu/cpufreq/policy0/scaling_governor
3  /* 设置216MHz */
4  echo 216000 > /sys/devices/system/cpu/cpufreq/policy0/scaling_setspeed

```

大小核平台，比如RK3399，执行如下命令：

```

1  /* 切换小核governor到userspace */
2  echo userspace > /sys/devices/system/cpu/cpufreq/policy0/scaling_governor
3  /* 设置小核216MHz */
4  echo 216000 > /sys/devices/system/cpu/cpufreq/policy0/scaling_setspeed
5
6  /* 切换大核governor到userspace */
7  echo userspace > /sys/devices/system/cpu/cpufreq/policy4/scaling_governor
8  /* 设置大核408MHz */
9  echo 408000 > /sys/devices/system/cpu/cpufreq/policy4/scaling_setspeed

```

注意：通过cpufreq节点设置CPU频率时，通常电压也会被改变，除非两个频点的电压相同。

5.5 如何查看当前频率

可以通过cpufreq的用户接口和clock的debug接口两种方法查看频率。

非大小核平台，执行如下命令：

```

1  /* 方法一：cpufreq的用户态接口 */
2  cat /sys/devices/system/cpu/cpufreq/policy0/scaling_cur_freq
3
4  /* 方法二：clock debug接口 */
5  cat /sys/kernel/debug/clk/armclk/clk_rate

```

大小核平台，执行如下命令：

```

1  /* 方法一: cpufreq的用户态接口 */
2  cat /sys/devices/system/cpu/cpufreq/policy0/scaling_cur_freq /* 小核频率 */
3  cat /sys/devices/system/cpu/cpufreq/policy4/scaling_cur_freq /* 大核频率 */
4
5  /* 方法二: clock debug接口 */
6  cat /sys/kernel/debug/clk/armclk1/clk_rate /* 小核频率 */
7  cat /sys/kernel/debug/clk/armclkb/clk_rate /* 大核频率 */

```

5.6 如何查看当前电压

非大小核平台，执行如下命令：

```

1  /* 不一定是vdd_core，根据实际的regulator配置修改 */
2  cat /sys/kernel/debug/regulator/vdd_core/voltage

```

大小核平台，执行如下命令：

```

1  /* 不一定是vdd_core_l和vdd_core_b，根据实际的regulator配置修改 */
2  cat /sys/kernel/debug/regulator/vdd_core_l/voltage /* 小核电压 */
3  cat /sys/kernel/debug/regulator/vdd_core_b/voltage /* 小核电压 */

```

5.7 如何单独调频调压

关闭CPU自动变频，参考5.3中的方法三。

调频，通过clock的debug接口设置频率，举例如下：

```

1  /* 非大小核平台，比如RK3288，设置216MHz */:
2  echo 216000000 > /sys/kernel/debug/clk/armclk/clk_rate /* 设置频率 */
3  cat /sys/kernel/debug/clk/armclk1/clk_rate /* 查看频率 */
4
5  /* 大小核平台，比如RK3399，小核设置216MHz，大核设置408Mhz */
6  echo 216000000 > /sys/kernel/debug/clk/armclk1/clk_rate /* 设置小核频率 */
7  cat /sys/kernel/debug/clk/armclk1/clk_rate /* 查看小核频率 */
8  echo 408000000 > /sys/kernel/debug/clk/armclkb/clk_rate /* 设置大核频率 */
9  cat /sys/kernel/debug/clk/armclkb/clk_rate /* 查看大核频率 */

```

调压，通过regulator的debug接口设置电压，举例如下：

```

1  /*
2   * 非大小核平台，比如RK3288，设置950mV，
3   * 不一定是vdd_core，根据实际的regulator配置修改
4   */
5  echo 950000 > /sys/kernel/debug/regulator/vdd_core/voltage /* 设置电压 */
6  cat /sys/kernel/debug/regulator/vdd_core/voltage /* 查看电压 */
7
8  /*
9   * 大小核平台，比如RK3399，小核设置950mV，大核设置1000mV，
10  * 不一定是vdd_core_l和vdd_core_b，根据实际的regulator配置修改
11  */

```

```

12 echo 950000 > /sys/kernel/debug/regulator/vdd_core_l/voltage /* 设置小核电压 */
13 cat /sys/kernel/debug/regulator/vdd_core_l/voltage /* 查看小核电压 */
14 echo 950000 > /sys/kernel/debug/regulator/vdd_core_b/voltage /* 设置大核电压 */
15 cat /sys/kernel/debug/regulator/vdd_core_b/voltage /* 查看小核电压 */

```

注意：升频时，先升压再升频，降频时，先降频再降压。

5.8 如何查看当前电压的档位

如果是通过PVTM调压，执行如下命令

```
1 dmesg | grep pvtm
```

以RK3399 CPU为例，会打印出如下信息：

```

1 [ 0.669456] cpu cpu0: temp=22222, pvtm=138792 (140977 + -2185)
2 /* pvtm-volt-sel=0, 说明当前芯片小核用的是opp-microvolt-L0对应的电压 */
3 [ 0.670601] cpu cpu0: pvtm-volt-sel=0
4 [ 0.683008] cpu cpu4: temp=22222, pvtm=148761 (150110 + -1349)
5 /* pvtm-volt-sel=1, 说明当前芯片大核用的是opp-microvolt-L1对应的电压 */
6 [ 0.683109] cpu cpu4: pvtm-volt-sel=1
7 [ 1.495247] rockchip-dmc dmc: Failed to get pvtm
8 [ 3.366028] mali ff9a0000.gpu: temp=22777, pvtm=120824 (121698 + -874)
9 [ 3.366915] mali ff9a0000.gpu: pvtm-volt-sel=0

```

同理如果是通过leakage调压，则执行如下命令，也有类似打印输出。

```
1 dmesg | grep leakage
```

5.9 如何查看leakage

执行如下命令

```
1 dmesg | grep leakage
```

以RK3399 CPU为例，会有如下打印：

```

1 [ 0.656175] cpu cpu0: leakage=10 /* leakage=10, 说明当前芯片小核的leakage是10mA */
2 [ 0.671092] cpu cpu4: leakage=20 /* leakage=20, 说明当前芯片大核的leakage是20mA */
3 [ 1.492769] rockchip-dmc dmc: Failed to get leakage
4 [ 3.341084] mali ff9a0000.gpu: leakage=15

```