Devfreq开发指南

发布版本：1.0

作者邮箱：finley.xiao@rock-chips.com

日期：2018.09.14

文档密级：公开资料

前言

概述

主要描述devfreq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
2018-09-14	V1.0	肖锋	初始版本

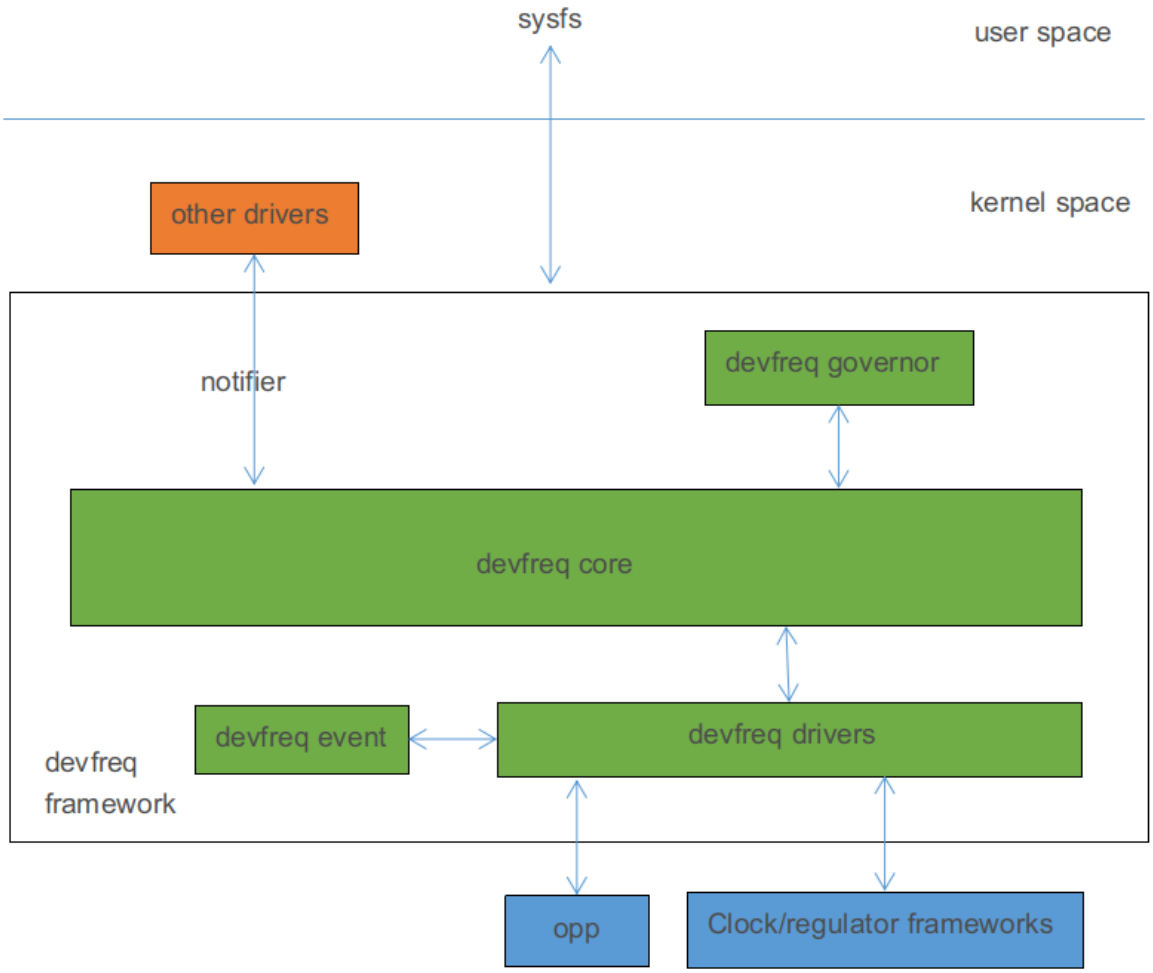
---

1 概述

Devfreq是内核开发者定义的一套支持根据指定的governor动态调整频率和电压的框架模型，它能有效地降

低的功耗，同时兼顾性能。Devfreq类似CPUFreq，不过CPUFreq只适用于CPU，devfreq用于除了CPU外，也需

要动态调频调压的模块。Devfreq framework由governor、core、driver、event组成，软件框架如下：



Devfreq governor：用于升降频检测，决定频率。目前Linux4.4内核中包含了如下几种governor：

- simple ondemand：根据负载动态调频。
- 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。
- dmc ondemand：simple ondemand的基础上，增加场景变频的支持，DDR变频专用。

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

Devfreq driver：用于初始化设备的频率电压表，设置具体设备的频率。

Devfreq event：用于监控设备的负载信息。

2 代码路径

Governor相关代码：

```
drivers/devfreq/governor_simpleondemand.c        /* simple ondemand调频策略 */
drivers/devfreq/governor_performance.c           /* performance调频策略 */
drivers/devfreq/governor_powersave.c             /* powersave调频策略 */
drivers/devfreq/governor_userspace.c             /* userspace调频策略 */
```

Event相关代码：

```
drivers/devfreq/devfreq-event.c
drivers/devfreq/event/rockchip-dfi.c              /* 用于监控DDR的读写cycle数 */
drivers/devfreq/event/rockchip-nocp.c             /* 用于监控各个模块访问DDR的字节数 */
```

Core相关代码：

```
drivers/devfreq/devfreq.c
```

Driver相关代码：

```
drivers/devfreq/rockchip_dmc.c                  /* dmc ondemand调频策略和DMC driver */
drivers/gpu/arm/midgard/backend/gpu/mali_kbase_devfreq.c            /* GPU driver */
drivers/gpu/arm/bifrost_for_linux/backend/gpu/mali_kbase_devfreq.c /* GPU driver */
drivers/gpu/arm/bifrost/backend/gpu/mali_kbase_devfreq.c            /* GPU driver */
drivers/gpu/arm/mali400/mali/linux/mali_devfreq.c                   /* GPU driver */
drivers/devfreq/rockchip_bus.c                                      /* bus driver */
drivers/soc/rockchip/rockchip_opp_select.c                        /* 修改电压表相关接口 */
```

3 Menuconfig配置

```
Device Drivers  --->
    [*] Generic Dynamic Voltage and Frequency Scaling (DVFS) support  --->
        --- Generic Dynamic Voltage and Frequency Scaling (DVFS) support
            *** DEVFREQ Governors ***           /* devfreq governor */
        -*-   Simple Ondemand
        <*>   Performance
        <*>   Powersave
            *** DEVFREQ Drivers ***
        <*>   ARM ROCKCHIP BUS DEVFREQ Driver   /* bus devfreq driver */
        <*>   ARM ROCKCHIP DMC DEVFREQ Driver   /* dmc devfreq driver */
        [*]   DEVFREQ-Event device Support  --->
            --- DEVFREQ-Event device Support
            -*-   ROCKCHIP DFI DEVFREQ event Driver  /* dfi event driver */
            /* nocp event driver */
            <*>   ROCKCHIP NoC (Network On Chip) Probe DEVFREQ event Driver
```

不同的平台可根据实际情况修改配置。

4 Device Tree配置方法

4.1 GPU DVFS配置方法

4.1.1 Clock配置

根据平台的实际情况，在GPU节点下增加“clock”和“clock-names”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。以RK3399为例：

```
gpu: gpu@ff9a0000 {
    compatible = "arm,malit860",
                 "arm,malit86x",
                 "arm,malit8xx",
                 "arm,mali-midgard";
    ...
    clocks = <&cru ACLK_GPU>;
    clock-names = "clk_mali";
    ...
};
```

4.1.2 Regulator配置

根据实际产品硬件使用的电源方案，在GPU节点下增加“mali-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。以RK3399为例：

```
&i2c0 {
    ...
    vdd_gpu: syr828@41 {
        compatible = "silergy,syr828";
        reg = <0x41>;
        vin-supply = <&vcc5v0_sys>;
        regulator-compatible = "fan53555-reg";
        pinctrl-0 = <&vsel2_gpio>;
        vsel-gpios = <&gpio1 14 GPIO_ACTIVE_HIGH>;
        regulator-name = "vdd_gpu";
        regulator-min-microvolt = <712500>;
        regulator-max-microvolt = <1500000>;
        regulator-ramp-delay = <1000>;
        fcs,suspend-voltage-selector = <1>;
        regulator-always-on;
        regulator-boot-on;
        regulator-initial-state = <3>;
            regulator-state-mem {
            regulator-off-in-suspend;
        };
    };
};

&gpu {
    status = "okay";
    mali-supply = <&vdd_gpu>;
};
```

4.1.3 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、leaakge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：

```
Documentation/devicetree/bindings/opp/opp.txt
Documentation/power/opp.txt
```

4.1.3.1 增加OPP Table

根据平台的实际情况，增加一个OPP Table节点，并在GPU节点下增加“operating-points-v2”属性，一般在

DTSI文件中。以RK3399为例：

```
&gpu {
    operating-points-v2 = <&gpu_opp_table>;
};

gpu_opp_table: opp-table2 {
    compatible = "operating-points-v2";

    opp-200000000 {
        opp-hz = /bits/ 64 <200000000>;          /* 单位Hz */
        opp-microvolt = <800000>;                /* 单位uV */
    };
    ...
    opp-800000000 {
        opp-hz = /bits/ 64 <800000000>;
        opp-microvolt = <1100000>;
    };
}
```

4.1.3.2 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OPP节点下增加“status = "disabeld";”，比如：

```
gpu_opp_table: opp-table2 {
    compatible = "operating-points-v2";

    opp-200000000 {
        opp-hz = /bits/ 64 <200000000>;          /* 单位Hz */
        opp-microvolt = <800000>;                /* 单位uV */
    };
    ...
    opp-800000000 {
        opp-hz = /bits/ 64 <800000000>;
        opp-microvolt = <1100000>;
        status = "disabled";
    };
}
```

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

```
&gpu_opp_table {
    opp-800000000 {
        status = "disabled";
    };
};
```

4.1.4 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。GPU的leakage指给GPU提供特定的电压，测得的静态电流值，如果GPU在VD logic

下，GPU的leakage等同于logic的leakage，即给logic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。

4.1.4.1 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GPU 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：

```
gpu_opp_table: gpu-opp-table {
    compatible = "operating-points-v2";

    /*
     * 从eFuse或OTP中获取GPU leakage值
     */
    nvmem-cells = <&gpu_leakage>;
    nvmem-cell-names = "gpu_leakage";

    /*
     * leakage值为1mA-10mA的芯片，使用opp-microvolt-L0指定的电压
     * leakage值为11mA-254mA的芯片，使用opp-microvolt-L1指定的电压
     *
     * 如果删除rockchip,leakage-voltage-sell属性或者leakage值不在该属性指定的范围内，
     * 则使用opp-microvolt指定的电压。
     */
    rockchip,leakage-voltage-sel = <
        1   10    0
        11  254   1
    >;

```

```
    opp-200000000 {
        opp-hz = /bits/ 64 <200000000>;
        opp-microvolt = <950000>;
        opp-microvolt-L0 = <950000>;
        opp-microvolt-L1 = <950000>;
    };
    ...
    opp-500000000 {
        opp-hz = /bits/ 64 <500000000>;
        opp-microvolt = <1150000>;
        opp-microvolt-L0 = <1150000>;
        opp-microvolt-L1 = <1100000>;
    };
};
```

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。

4.1.5 根据PVTM调整OPP Table

GPU PVTM(Process-Voltage-Temperature Monitor)是一个位于GPU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。

4.1.5.1 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RK3399为例：

```
gpu_opp_table: opp-table2 {
    compatible = "operating-points-v2";

    /*
     * PVTM值为0-121000的芯片，使用opp-microvolt-L0指定的电压；
     * PVTM值为121001-125500的芯片，使用opp-microvolt-L1指定的电压；
     * PVTM值为125501-128500的芯片，使用opp-microvolt-L2指定的电压；
     * PVTM值为128501-999999的芯片，使用opp-microvolt-L3指定的电压；
     *
     * 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，
     * 则使用opp-microvolt指定的电压。
```

```
     */
    rockchip,pvtm-voltage-sel = <
        0        121000   0
        121001   125500   1
        125501   128500   2
        128501   999999   3
    >;
    rockchip,pvtm-freq = <200000>;          /* 获取PVTM值前，需要先设置GPU频率，单位Khz */
    rockchip,pvtm-volt = <900000>;          /* 获取PVTM值前，需要先设置GPU电压，单位uV */
    rockchip,pvtm-ch = <3 0>;               /* PVTM通道，格式<通道序号 sel的序号> */
    rockchip,pvtm-sample-time = <1000>;     /* PVTM采样时间，单位us */
    rockchip,pvtm-number = <10>;            /* PVTM采样个数 */
    rockchip,pvtm-error = <1000>;           /* 允许采样数据之间的误差 */
    rockchip,pvtm-ref-temp = <41>;          /* 参考温度 */
    /* 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> */
    rockchip,pvtm-temp-prop = <46 12>;
    rockchip,thermal-zone = "gpu-thermal";  /* 通过哪个thermal-zone获取温度 */

    opp-200000000 {
        opp-hz = /bits/ 64 <200000000>;
        opp-microvolt = <800000>;
        opp-microvolt-L0 = <800000>;
        opp-microvolt-L1 = <800000>;
        opp-microvolt-L2 = <800000>;
        opp-microvolt-L3 = <800000>;
    };
    ...
    opp-800000000 {
        opp-hz = /bits/ 64 <800000000>;
        opp-microvolt = <1100000>;
        opp-microvolt-L0 = <1100000>;
        opp-microvolt-L1 = <1075000>;
        opp-microvolt-L2 = <1050000>;
        opp-microvolt-L3 = <1025000>;
    };
};
```

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。

4.1.6 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板相同的电压表，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 Ttable。

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，如果最终

电压超过了允许设置的最高电压，该频点将会被删除。

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：

```
gpu_opp_table: gpu-opp-table {
    compatible = "operating-points-v2";

    /* 允许设置的最高电压，单位uV */
    rockchip,max-volt = <1175000>;
    rockchip,evb-irdrop = <25000>;/* EVB板或者SDK板的电源纹波 */
    ...
}
```

板级DTS文件中配置如下：

```
&gpu_opp_table {
    /*
     * max IR-drop values on different freq condition for this board!
     */
    /*
     * 实际产品硬件，不同频率下的电源纹波情况:
     * 200Mhz-520MHz，电源纹波为50000uV，最终电压会增加25000uV（50000-25000（EVB板纹波））
     */
    rockchip,board-irdrop = <
        /* MHz  MHz uV */
           200  520 50000
    >;
};
```

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

4.1.7 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。

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。

配置方法：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。这些配置一般都在

DTSI文件中，以RK3399为例：

```
gpu_opp_table: opp-table2 {
    compatible = "operating-points-v2";

    /*
     * 迟滞参数，单位millicelsius，防止频繁进入低温或者高温
```

```
     * 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
     */
    rockchip,temp-hysteresis = <5000>;
    rockchip,low-temp = <0>;                    /* 低温阀值，单位millicelsius*/
    rockchip,low-temp-min-volt = <900000>;  /* 低温下最低电压，单位uV */
    rockchip,low-temp-adjust-volt = <
        /* MHz     MHz     uV */
           0       800     25000               /* 低温下，0-800MHz内的频点，电压增加25mV */
    >;
    rockchip,max-volt = <1150000>;          /* 最高电压不超过该值 */
    ...
}
```

4.1.8 升降频负载配置

背景：Simple ondemand调频策略有两个参数可以配置upthreshold和downdifferential，默认值分别是90

和5。当负载超过90%时，调到最高频，当负载小于90%且大于90%-5%是维持当前频率，当负载小于90%-5%，会

调到一个频率，使得负载差不多为90%-5%/2。使用默认的配置，某些平台在某些场景下会出现GPU提频不及时或

不提频，导致丢帧，所以需要支持修改配置。

配置方法：在GPU节点增加“upthreshold”、downdifferential“属性，这些配置一般都在DTSI文件中，以

RK3288为例：

```
gpu: gpu@ffa30000 {
    compatible = "arm,malit764",
                 "arm,malit76x",
                 "arm,malit7xx",
                 "arm,mali-midgard";
    reg = <0x0 0xffa30000 0x0 0x10000>;

    upthreshold = <75>;
    downdifferential = <10>;
    ...
}
```

4.2 DMC DVFS配置方法

DMC（Dynamic Memory Controller）DVFS，即DDR变频。

4.2.1 Clock配置

根据平台的实际情况，在DMC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。以RK3399为例：

```
dmc: dmc {
    compatible = "rockchip,rk3399-dmc";
    ...
    clocks = <&cru SCLK_DDRCLK>;
    clock-names = "dmc_clk";
    ...
};
```

4.2.2 Regulator配置

根据实际产品硬件使用的电源方案，在DMC节点下增加“center-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。以RK3399为例：

```
&i2c0 {
    ...
    rk808: pmic@1b {
        ...
        regulators {
            vdd_center: DCDC_REG1 {
                regulator-always-on;
                regulator-boot-on;
                regulator-min-microvolt = <750000>;
                regulator-max-microvolt = <1350000>;
                regulator-ramp-delay = <6001>;
                regulator-name = "vdd_center";
                regulator-state-mem {
                    regulator-off-in-suspend;
                };
            };
        };
    };
};

&dmc {
    status = "okay";
    center-supply = <&vdd_center>;
};
```

4.2.3 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、leaakge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：

```
Documentation/devicetree/bindings/opp/opp.txt
Documentation/power/opp.txt
```

4.2.3.1 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DMC节点下增加“operating-points-v2”属性，

一般在DTSI文件中。以RK3399为例：

```
&dmc {
    operating-points-v2 = <&dmc_opp_table>;
};

dmc_opp_table: opp-table3 {
    compatible = "operating-points-v2";

    opp-200000000 {
        opp-hz = /bits/ 64 <200000000>;          /* 单位Hz */
        opp-microvolt = <900000>;                /* 单位uV */
    };
    ...
    opp-800000000 {
        opp-hz = /bits/ 64 <800000000>;
        opp-microvolt = <900000>;
    };
};
```

4.2.3.2 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OPP节点下增加“status = "disabeld";”，比如：

```
dmc_opp_table: opp-table3 {
    compatible = "operating-points-v2";

    opp-200000000 {
        opp-hz = /bits/ 64 <200000000>;          /* 单位Hz */
        opp-microvolt = <800000>;                /* 单位uV */
    };
    ...
    opp-800000000 {
        opp-hz = /bits/ 64 <800000000>;
        opp-microvolt = <900000>;
        status = "disabled";
    };
}
```

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 = "disabeld";”，比如：

```
&dmc_opp_table {
    opp-800000000 {
        status = "disabled";
    };
};
```

4.2.4 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。DDR的leakage指给ddr提供特定的电压，测得的静态电流值，如果DDR在VD logic

下，DDR的leakage等同于logic的leakage，即给logic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。

4.2.4.1 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DDR 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：

```
dmc_opp_table: dmc-opp-table {
    compatible = "operating-points-v2";

    /*
     * 从eFuse或OTP中获取DDR leakage值
     */
    nvmem-cells = <&logic_leakage>;
    nvmem-cell-names = "ddr_leakage";

    /*
     * leakage值为1mA-10mA的芯片，使用opp-microvolt-L0指定的电压
     * leakage值为11mA-254mA的芯片，使用opp-microvolt-L1指定的电压
     *
     * 如果删除rockchip,leakage-voltage-sell属性或者leakage值不在该属性指定的范围内，
     * 则使用opp-microvolt指定的电压。
     */
    rockchip,leakage-voltage-sel = <
        1   10    0
        11  254   1
    >;

    opp-400000000 {
        opp-hz = /bits/ 64 <400000000>;
        opp-microvolt = <950000>;
        opp-microvolt-L0 = <950000>;
        opp-microvolt-L1 = <950000>;
    };
    ...
    opp-1066000000 {
        opp-hz = /bits/ 64 <1066000000>;
        opp-microvolt = <1175000>;
        opp-microvolt-L0 = <1175000>;
        opp-microvolt-L1 = <1150000>;
    };
```

```
};
```

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。

4.2.5 根据PVTM调整OPP Table

4.2.5.1 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PX30为例：

```
dmc_opp_table: dmc-opp-table {
    compatible = "operating-points-v2";

    /*
     * PVTM值为0-50000的芯片，使用opp-microvolt-L0指定的电压；
     * PVTM值为50001-54000的芯片，使用opp-microvolt-L1指定的电压；
     * PVTM值为54001-60000的芯片，使用opp-microvolt-L2指定的电压；
     * PVTM值为60001-99999的芯片，使用opp-microvolt-L3指定的电压；
     *
     * 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，
     * 则使用opp-microvolt指定的电压。
     */
    rockchip,pvtm-voltage-sel = <
        0        50000   0
        50001    54000   1
        54001    60000   2
        60001    99999   3
    >;
    rockchip,pvtm-ch = <0 0>; /* 延用CPU的PVTM值 */

    opp-194000000 {
        opp-hz = /bits/ 64 <194000000>;
        opp-microvolt = <950000>;
        opp-microvolt-L0 = <950000>;
        opp-microvolt-L1 = <950000>;
        opp-microvolt-L2 = <950000>;
        opp-microvolt-L3 = <950000>;
    };
    ...
```

```
    opp-786000000 {
        opp-hz = /bits/ 64 <786000000>;
        opp-microvolt = <1100000>;
        opp-microvolt-L0 = <1100000>;
        opp-microvolt-L1 = <1050000>;
        opp-microvolt-L2 = <1025000>;
        opp-microvolt-L3 = <1000000>;
        status = "disabled";
    };
};
```

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。

4.2.6 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板相同的电压表，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，如果最终

电压超过了允许设置的最高电压，该频点将会被删除。

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：

```
dmc_opp_table: dmc-opp-table {
    compatible = "operating-points-v2";

    /* 允许设置的最高电压，单位uV */
    rockchip,max-volt = <1150000>;
    rockchip,evb-irdrop = <25000>;/* EVB板或者SDK板的电源纹波 */
    ...
}
```

板级DTS文件中配置如下：

```
&dmc_opp_table {
    /*
     * max IR-drop values on different freq condition for this board!
     */
    /*
     * 实际产品硬件，不同频率下的电源纹波情况：
     * 451Mhz-800MHz，电源纹波为75000uV，最终电压会增加50000uV（75000-25000（EVB板纹波））
     */
    rockchip,board-irdrop = <
        /* MHz  MHz uV */
           451  800 75000
    >;
};
```

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

4.2.7 场景变频配置

背景：如果DDR固定频率，频率高了，功耗大，频率低了，性能差，很难满足产品需求。针对某些对DDR的

需求比较明确的场景，比如跑分，视频，待机等，动态提高或者降低DDR频率，可以满足他们对性能或者功耗的

不同需求。

功能说明：当系统进入某些特殊的场景时，将DDR频率调整到该场景指定的频率，如果同时进入多个场景，

最终频率取最大值，需要注意的是在SYS_STATUS_DUALVIEW和SYS_STATUS_DUALVIEW场景下，不支持DDR变

频，所以进入这两个场景后，即使再进入更高DDR频率的场景，DDR频率依然不变，直达退出这两个场景。

配置方法：在DMC节点增加“system-status-freq ”属性，以RK3399为例：

```
&dmc {
    status = "okay";
    ...
    system-status-freq = <
        /* system status         freq(KHz) */
        SYS_STATUS_NORMAL        800000  /* 除了以下定义的场景，其他场景都用该频率 */
        SYS_STATUS_REBOOT        528000  /* reboot场景，在reboot前设置DDR频率 */
        SYS_STATUS_SUSPEND       200000  /* 一级待机场景，灭屏后设置DDR频率 */
        SYS_STATUS_VIDEO_1080P   200000  /* 1080视频场景，播放视频前设置DDR频率 */
        SYS_STATUS_VIDEO_4K      600000  /* 4k视频场景，播放视频前设置DDR频率 */
        SYS_STATUS_VIDEO_4K_10B  800000  /* 4k 10bit视频场景，播放视频前设置DDR频率 */
        SYS_STATUS_PERFORMANCE   800000  /* 跑分场景，启动软件时前设置DDR频率 */
        SYS_STATUS_BOOST         400000  /* 触屏场景，需开启负载变频，触屏后修改DDR频率最小值 */
        SYS_STATUS_DUALVIEW      600000  /* 双屏显示场景，第二个屏显示前固定DDR频率 */
```

```
        SYS_STATUS_ISP          600000  /* 拍照场景，启动ISP前固定DDR频率 */
    >;
}
```

4.2.8 负载变频配置

背景：场景变频只能覆盖很少一部分场景，除此之外的场景需要根据DDR的利用率动态调整DDR频率，以优

化性能和功耗。

功能说明：定时检测DDR的利用率，根据simple ondeman的算法选择一个目标频率，并考虑特定场景对

DDR带宽的需求，最终选择一个最大值。需要注意的是，和场景变频一样，SYS_STATUS_DUALVIEW和

SYS_STATUS_ISP场景下DDR频率是固定的。

配置方法：在DMC节点增加“devfreq-events ”，“upthreshold”，“downdifferential”，

“system-status-freq”，“auto-min-freq”和“auto-freq-en”属性，以RK3399为例：

```
&dmc {
    status = "okay";
    ...
    devfreq-events = <&dfi>;            /* 通过dfi监控DDR的利用率 */
    /*
     * 调频阀值：
     * 当利用率超过40%时，调到最高频，
     * 当负载小于40%且大于40%-20%是维持当前频率
     * 当负载小于40%-20%，会调到一个频率，使得负载差不多为40%-2%/2。
     */
    upthreshold = <40>;
    downdifferential = <20>;
    system-status-freq = <
        /* system status        freq(KHz) */
        SYS_STATUS_NORMAL       800000  /* 启动负载变频后，该场景无效 */
        SYS_STATUS_REBOOT       528000  /* reboot场景，在reboot前修改DDR频率的
最低值 */
        SYS_STATUS_SUSPEND      200000  /* 一级待机场景，灭屏后修改DDR频率的最低
值 */
        SYS_STATUS_VIDEO_1080P  200000  /* 1080视频场景，播放视频前修改DDR频率的
最低值 */
        SYS_STATUS_VIDEO_4K     600000  /* 4k视频场景，播放视频前修改DDR频率的最
低值 */
        SYS_STATUS_VIDEO_4K_10B 800000  /* 4k 10bit视频场景，播放视频前修改DDR频
率的最低值 */
        SYS_STATUS_PERFORMANCE  800000  /* 跑分场景，启动软件时前修改DDR频率的最
低值 */
        SYS_STATUS_BOOST        400000  /* 触屏场景，需开启负载变频，触屏后修改
DDR频率最低值 */
        SYS_STATUS_DUALVIEW     600000  /* 双屏显示场景，第二个屏显示前固定DDR的
频率 */
        SYS_STATUS_ISP          600000  /* 拍照场景，启动ISP前固定DDR的频率 */
    >;
    /* 除了以上定义的场景，其他场景下DDR频率的最低值，防止提频不及时导致闪屏 */
    auto-min-freq = <400000>;
    auto-freq-en = <1>;                 /* 负载变频开关，1为开启，0为关闭 */
};
```

4.2.9 根据VOP带宽变频

背景：开启负载变频后，需要增加“auto-min-freq”属性限制最低频率，防止某些场景下提频不及导致闪屏，

所以这些场景的功耗仍然有优化的空间，因此引入根据VOP带宽调整DDR频率。

功能说明：每一帧显示之前，VOP驱动先计算出这一帧的DDR带宽需求，然后根据带宽需求修改DDR频率的

最低值。

配置方法：在DMC节点增加"vop-bw-dmc-freq"属性，以RK3399为例：

```
&dmc {
    status = "okay";
    ...
    /*
     * VOP带宽需求为0-577MB/s，DDR频率最低值为200MHz，
     * VOP带宽需求为578-1701MB/s，DDR频率最低值为300MHz，
     * VOP带宽需求为1702-99999MB/s，DDR频率最低值为400MHz，
     */
    vop-bw-dmc-freq = <
    /* min_bw(MB/s) max_bw(MB/s) freq(KHz) */
        0       577       200000
        578     1701      300000
        1702    99999     400000
    >;
    /*
     * 除了定义的场景，其他场景下DDR频率的最低值
     * 加入VOP带宽统计后，可将该值改成比较低的频率。
     */
    auto-min-freq = <200000>;
};
```

4.3 BUS DVFS配置方法

除了GPU、DMC外，还有一些模块也需要动态调频调压，比如PLL、CCI等，我们将他们统一归类到BUS

DVFS。

4.3.1 PLL DVFS配置

背景：在某些平台发现PLL的频率超过一定值后，PLL所在的电压域需要提高电压，因此需要根据PLL的频率调

整电压。

功能说明：通过注册clock notifier，监控PLL频率的变化，如果PLL是升频，先抬压再提频，如果PLL是降频，

先降频再降压。

配置方法：需要增加”rockchip,busfreq-policy“、”clocks“、”clock-names“、”operating-points-v2“和

“bus-supply”属性。

以PX30为例，DTSI文件配置如下：

```
bus_apll: bus-apll {
    compatible = "rockchip,px30-bus";
    /*
     * 使用clkfreq调频调压策略，通过注册clock notifier，监控PLL频率的变化，
     * 如果PLL是升频，先抬压再提频，如果PLL是降频，先降频再降压.
     */
    rockchip,busfreq-policy = "clkfreq";
    clocks = <&cru PLL_APLL>;                    /* 时钟配置 */
    clock-names = "bus";
    operating-points-v2 = <&bus_apll_opp_table>; /* OPP Table配置 */
    status = "disabled";
};

bus_apll_opp_table: bus-apll-opp-table {
    compatible = "operating-points-v2";
    opp-shared;
    /* PLL频率小于等于1008MHz，电压950mV，大于1008MHz，电压1000mV */
    opp-1512000000 {
        opp-hz = /bits/ 64 <1512000000>;
        opp-microvolt = <1000000>;
    opp-1008000000 {
        opp-hz = /bits/ 64 <1008000000>;
        opp-microvolt = <950000>;
    };
};
```

板级配置如下：

```
&i2c0 {
    status = "okay";
    rk809: pmic@20 {
        compatible = "rockchip,rk809";
        reg = <0x20>;
        ...
        regulators {
            vdd_logic: DCDC_REG1 {
                regulator-always-on;
                regulator-boot-on;
                regulator-min-microvolt = <950000>;
                regulator-max-microvolt = <1350000>;
                regulator-ramp-delay = <6001>;
                regulator-initial-mode = <0x2>;
                regulator-name = "vdd_logic";
                regulator-state-mem {
                    regulator-on-in-suspend;
                    regulator-suspend-microvolt = <950000>;
                };
            };
        }
    }
}

&bus_apll {
    bus-supply = <&vdd_logic>; /* regulator配置，根据实际产品硬件使用的电源方案修改 */
    status = "okay";
```

```
};
```

5 用户态接口介绍

设备成功注册devfreq后，会在/sys/class/devfreq/目录下生成一个包含用户态接口的子目录，比如ff9a0000.gpu，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：

```
available_frequencies        /* 系统支持的频率 */
available_governors          /* 系统支持的变频策略 */
cur_freq                     /* 当前频率 */
governor                     /* 当前使用的变频策略 */
load                         /* 当前负载 */
max_freq                     /* 软件上限制的最高频率 */
min_freq                     /* 软件上限制的最低频率 */
polling_interval             /* 检测负载的间隔时间 */
target_freq                  /* 软件上最后一次设置的频率 */
trans_stat                   /* 每个频率上的变频次数和运行时间 */
```

6 常见问题

6.1 如何查看频率电压表

执行如下命令：

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

以PX30为例：

```
 device                rate(Hz)    target(uV)    min(uV)    max(uV)
-------------------------------------------------------------------
 platform-dmc
                      194000000        950000     950000     950000
                      328000000        950000     950000     950000
                      450000000        950000     950000     950000
                      528000000        975000     975000     975000
                      666000000       1000000    1000000    1000000
 platform-ff400000.gpu
                      200000000        950000     950000     950000
                      300000000        950000     950000     950000
                      400000000       1025000    1025000    1025000
                      480000000       1100000    1100000    1100000
 platform-bus-apll
                     1008000000        950000     950000     950000
                     1512000000       1000000    1000000    1000000
```

6.2 如何定频

方法一：将OPP Table中不想要的频率全部disable掉，只留一个想要的频率即可。以PX30为例，GPU定频

400MHz的配置如下：

```
gpu_opp_table: gpu-opp-table {
```

```
    compatible = "operating-points-v2";
    ...
    opp-200000000 {
        opp-hz = /bits/ 64 <200000000>;
        opp-microvolt = <950000>;
        opp-microvolt-L0 = <950000>;
        opp-microvolt-L1 = <950000>;
        opp-microvolt-L2 = <950000>;
        opp-microvolt-L3 = <950000>;
        status = "disabled";
    };
    opp-300000000 {
        opp-hz = /bits/ 64 <300000000>;
        opp-microvolt = <975000>;
        opp-microvolt-L0 = <975000>;
        opp-microvolt-L1 = <950000>;
        opp-microvolt-L2 = <950000>;
        opp-microvolt-L3 = <950000>;
        status = "disabled";
    };
    opp-400000000 {
        opp-hz = /bits/ 64 <400000000>;
        opp-microvolt = <1050000>;
        opp-microvolt-L0 = <1050000>;
        opp-microvolt-L1 = <1025000>;
        opp-microvolt-L2 = <975000>;
        opp-microvolt-L3 = <950000>;
    };
    opp-480000000 {
        opp-hz = /bits/ 64 <480000000>;
        opp-microvolt = <1125000>;
        opp-microvolt-L0 = <1125000>;
        opp-microvolt-L1 = <1100000>;
        opp-microvolt-L2 = <1050000>;
        opp-microvolt-L3 = <1000000>;
        status = "disabled";
    };
};
```

方法二：开机后通过命令定频。以PX30为例，GPU定频400MHz的命令如下：

```
/* 切换到userspace，不一定是ff400000.gpu，根据不同的平台修改 */
echo userspace > /sys/class/devfreq/ff400000.gpu/governor
/* 设置400MHz */
echo 400000000 > /sys/class/devfreq/ff400000.gpu/userspace/set_freq
/* 查看当前频率 */
cat /sys/class/devfreq/ff400000.gpu/cur_freq
```

6.3 如何查看当前频率

可以通过devfreq的用户接口和clock的debug接口两种方法查看频率。以PX30为例，查看GPU的频率，命

令如下：

```
/* 方法一：devfreq的用户态接口，不一定是ff400000.gpu，根据不同的平台修改 */
cat /sys/class/devfreq/ff400000.gpu/cur_freq

/* 方法二：clock debug接口，不一定是aclk_gpu，根据实际的clock配置修改 */
cat /sys/kernel/debug/clk/aclk_gpu/clk_rate
```

6.4 如何查看当前电压

可以通过regulator的debug接口查看电压。以PX30为例，查看GPU的电压，命令如下：

```
/* 不一定是vdd_logic，根据实际的regulator配置修改 */
cat /sys/kernel/debug/regulator/vdd_logic/voltage
```

6.5 如何单独调频调压

以PX30 GPU为例，设置频率为400MHz，电压1000mV。

```
/* 关闭自动变频，不一定是ff400000.gpu，根据不同的平台修改 */
echo userspace > /sys/class/devfreq/ff400000.gpu/governor

/* 调频，不一定是aclk_gpu，根据实际的clock配置修改 */
echo 400000000 > /sys/kernel/debug/clk/aclk_gpu/clk_rate
cat /sys/kernel/debug/clk/aclk_gpu/clk_rate

/* 调压，不一定是vdd_logic，根据实际的regulator配置修改 */
echo 1000000 > /sys/kernel/debug/regulator/vdd_logic/voltage
cat /sys/kernel/debug/regulator/vdd_logic/voltage
```

注意：升频的时候，先升压再升频；降频的时候，先降频再降压。

6.6 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

```
dmesg | grep pvtm
```

以RK3399 GPU为例，会打印出如下信息：

```
[    0.669456] cpu cpu0: temp=22222, pvtm=138792 (140977 + -2185)
[    0.670601] cpu cpu0: pvtm-volt-sel=0
[    0.683008] cpu cpu4: temp=22222, pvtm=148761 (150110 + -1349)
[    0.683109] cpu cpu4: pvtm-volt-sel=0
[    1.495247] rockchip-dmc dmc: Failed to get pvtm
[    3.366028] mali ff9a0000.gpu: temp=22777, pvtm=120824 (121698 + -874)
/* pvtm-volt-sel=0，说明当前芯片GPU用的是opp-microvolt-L0对应的电压 */
[    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。

```
dmesg | grep leakage
```

6.7 如何查看leakage

执行如下命令

```
dmesg | grep leakage
```

以RK3399 GPU为例，会有如下打印：

```
[    0.656175] cpu cpu0: leakage=10
[    0.671092] cpu cpu4: leakage=20
[    1.492769] rockchip-dmc dmc: Failed to get leakage
/* leakage=15，说明当前芯片GPU的leakage是15mA */
[    3.341084] mali ff9a0000.gpu: leakage=15
```

6.8 如何修改电压

方法一：直接修改电压表，以GPU 200MHz抬压25000uV为例。

假设默认200MHz的OPP节点如下：

```
opp-200000000 {
    opp-hz = /bits/ 64 <200000000>;
    opp-microvolt = <800000>;
    opp-microvolt-L0 = <800000>;
    opp-microvolt-L1 = <800000>;
    opp-microvolt-L2 = <800000>;
    opp-microvolt-L3 = <800000>;
};
```

修改后：

```
opp-200000000 {
    opp-hz = /bits/ 64 <200000000>;
    /* 每个档位都要加25000uV */
    opp-microvolt = <825000>;
    opp-microvolt-L0 = <825000>;
    opp-microvolt-L1 = <825000>;
    opp-microvolt-L2 = <825000>;
    opp-microvolt-L3 = <825000>;
};
```

方法二：通过修改IR-Drop的配置，调整电压。以GPU 200MHz抬压25000uV为例。

假设IR-Drop默认配置如下：

```
&gpu_opp_table {
    /*
     * max IR-drop values on different freq condition for this board!
     */
    /*
     * 实际产品硬件，不同频率下的电源纹波情况：
     * 200Mhz-520MHz，电源纹波为50000uV，最终电压会增加25000uV（50000-25000（EVB板纹波））
     */
    rockchip,board-irdrop = <
        /* MHz  MHz uV */
           200  520 50000
    >;
};
```

修改后如下：

```
&gpu_opp_table {
    /*
     * max IR-drop values on different freq condition for this board!
     */
    /*
     * 实际产品硬件，不同频率下的电源纹波情况：
     * 200Mhz-299MHz，电源纹波为75000uV，最终电压会增加50000uV（75000-25000（EVB板纹波））
     * 300Mhz-520MHz，电源纹波为50000uV，最终电压会增加25000uV（50000-25000（EVB板纹波））
     */
    rockchip,board-irdrop = <
        /* MHz  MHz uV */
           200  299 75000  /* 200MHz-299MHz从之前的50000改成了75000 */
           300  520 50000
    >;
};
```