Linux CPUFREQ

开发指南

版本号: 2.1
发布日期: 2025.04.21

版本历史

版本号	日期	制/修订人	内容描述
1.0	2022.08.04	AWA0863	初始版本
1.1	2022.11.16	AWA2099	T3 系列 &A40i 系列的适用
1.2	2023.03.08	AWA1743	A523 系列适用
1.3	2023.04.13	AWA1743	MR527 系列适用
1.4	2023.05.23	AWA1743	AI985 系列适用
1.5	2023.08.01	AWA1743	T527 系列适用
1.6	2024.09.11	AWA2152	T536/MR536 系列适用
1.7	2024.11.13	AWA2152	A733 系列适用
1.8	2025.02.21	AWA2152	H135/H136 系列适用
1.9	2025.03.29	AWA0863	A537/A333 系列适用
2.0	2025.04.07	AWA2099	T736 系列适用
2.1	2025.04.21	AWA0863	H723/H726/H727/TV323 系列适用

目　录

1 前言

1.1 文档简介

介绍 CPUFREQ 使用方法。

1.2 目标读者

CPUFREQ 驱动及应用层的使用人员。

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
T507	Linux-5.10	bsp/drivers/cpufreq/*
T3/T3-C/T3-Pro	Linux-5.10	bsp/drivers/cpufreq/*
A40i/A40i-C/A40i-H	Linux-5.10	bsp/drivers/cpufreq/*
A523	Linux-5.15	bsp/drivers/cpufreq/*
MR527	Linux-5.15	bsp/drivers/cpufreq/*
AI985	Linux-5.15	bsp/drivers/cpufreq/*
T527	Linux-5.15	bsp/drivers/cpufreq/*
MR536	Linux-5.15-origin	bsp/drivers/cpufreq/*
T536	Linux-5.10-origin	bsp/drivers/cpufreq/*
A733	Linux-6.6	bsp/drivers/cpufreq/*
H135	Linux-6.6-xuantie	bsp/drivers/cpufreq/*
H136	Linux-6.6-xuantie	bsp/drivers/cpufreq/*
A537	Linux-6.6	bsp/drivers/cpufreq/*
A333	Linux-6.6	bsp/drivers/cpufreq/*
T736	Linux-5.15	bsp/drivers/cpufreq/*
H723	Linux-5.15	bsp/drivers/cpufreq/*
H726	Linux-5.15	bsp/drivers/cpufreq/*
H727	Linux-5.15	bsp/drivers/cpufreq/*
TV323	Linux-5.15	bsp/drivers/cpufreq/*

2 模块介绍

2.1 模块功能介绍

CPUFREQ 负责系统运行过程中 CPU 频率和电压的动态调整。

2.2 相关术语介绍

表 2-1: 术语介绍

术语	说明
Sunxi	指 Allwinner 的一系列 SOC 硬件平台。
DVFS	动态频率电压调整

2.3 模块配置介绍

2.3.1 Device Tree 配置说明

设备树中存在的是该类芯片所有平台的模块配置，设备树文件的路径为：bsp/configs/KERNEL_VER/CHIP.dtsi(KERNEL_VER 为内核版本，如 linux-5.10；CHIP 为研发代号，如 sun50iw10p1 等)。

- 对于 sun8iw19p1 等没有 cpu 分 bin 需求的平台，
  v-f 表：

```
cpu_opp_l_table: opp_l_table {
    compatible = "operating-points-v2";
    opp-shared;

    opp@720000000 {
        opp-hz = /bits/ 64 <720000000>;
        opp-microvolt = <820000>;
        clock-latency-ns = <244144>;
    };
    opp@936000000 {
        opp-hz = /bits/ 64 <936000000>;
```

```
        opp-microvolt = <820000>;
        clock-latency-ns = <244144>;
      };
      opp@1104000000 {
        opp-hz = /bits/ 64 <1104000000>;
        opp-microvolt = <900000>;
        clock-latency-ns = <244144>;
      };
      opp@1200000000 {
        opp-hz = /bits/ 64 <1200000000>;
        opp-microvolt = <950000>;
        clock-latency-ns = <244144>;
      };
      opp@1320000000 {
        opp-hz = /bits/ 64 <1320000000>;
        opp-microvolt = <1000000>;
        clock-latency-ns = <244144>;
      };
      opp@1416000000 {
        opp-hz = /bits/ 64 <1416000000>;
        opp-microvolt = <1050000>;
        clock-latency-ns = <244144>;
      };
      opp@1512000000 {
        opp-hz = /bits/ 64 <1512000000>;
        opp-microvolt = <1050000>;
        clock-latency-ns = <244144>;
      };
    };

compatible = "operating-points-v2";： 用于匹配驱动的属性。
opp-hz： 某个频点的频率。
opp-microvolt： 频率对应的电压。
```

cpu 节点：

```
      cpu0: cpu@0 {
        device_type = "cpu";
        compatible = "arm,cortex-a53","arm,armv8";
        reg = <0x0 0x0>;
        enable-method = "psci";
        clocks = <&clk_pll_cpu>;
        clock-latency = <2000000>;
        clock-frequency = <1320000000>;
        dynamic-power-coefficient = <190>;
        operating-points-v2 = <&cpu_opp_l_table>;
        cpu-idle-states = <&CPU_SLEEP_0 &CLUSTER_SLEEP_0>;
        #cooling-cells = <2>;
      };

      cpu@1 {
        device_type = "cpu";
        compatible = "arm,cortex-a53","arm,armv8";
        reg = <0x0 0x1>;
        enable-method = "psci";
        clocks = <&clk_pll_cpu>;
        clock-frequency = <1320000000>;
        operating-points-v2 = <&cpu_opp_l_table>;
        cpu-idle-states = <&CPU_SLEEP_0 &CLUSTER_SLEEP_0>;
        #cooling-cells = <2>;
```

```
    };

operating-points-v2 = <&cpu_opp_l_table>;：引用v-f表。
```

- 对于 sun50iw9p1、sun50iw10p1 等有 cpu 分 bin 需求的平台，v-f 表：

```
cpu_opp_l_table: opp_l_table {
  compatible = "allwinner,sun50i-operating-points";
  nvmem-cells = <&speedbin_efuse>;
  nvmem-cell-names = "speed";
  opp-shared;

  opp@480000000-0 {
    opp-hz = /bits/ 64 <480000000>;
    opp-microvolt = <820000>;
    clock-latency-ns = <244144>; /* 8 32k periods */
    opp-supported-hw = <0x3>;
  };
  opp@480000000-1 {
    opp-hz = /bits/ 64 <480000000>;
    opp-microvolt = <880000>;
    clock-latency-ns = <244144>; /* 8 32k periods */
    opp-supported-hw = <0x4>;
  };
  opp@600000000-0 {
    opp-hz = /bits/ 64 <600000000>;
    opp-microvolt = <820000>;
    clock-latency-ns = <244144>; /* 8 32k periods */
    opp-supported-hw = <0x3>;
  };
  opp@600000000-1 {
    opp-hz = /bits/ 64 <600000000>;
    opp-microvolt = <880000>;
    clock-latency-ns = <244144>; /* 8 32k periods */
    opp-supported-hw = <0x4>;
  };
  opp@792000000-0 {
    opp-hz = /bits/ 64 <792000000>;
    opp-microvolt = <860000>;
    clock-latency-ns = <244144>; /* 8 32k periods */
    opp-supported-hw = <0x3>;
  };
  opp@792000000-1 {
    opp-hz = /bits/ 64 <792000000>;
    opp-microvolt = <940000>;
    clock-latency-ns = <244144>; /* 8 32k periods */
    opp-supported-hw = <0x4>;
  };
  opp@1008000000-0 {
    opp-hz = /bits/ 64 <1008000000>;
    opp-microvolt = <900000>;
    clock-latency-ns = <244144>; /* 8 32k periods */
    opp-supported-hw = <0x3>;
  };
  opp@1008000000-1 {
    opp-hz = /bits/ 64 <1008000000>;
    opp-microvolt = <1020000>;
    clock-latency-ns = <244144>; /* 8 32k periods */
```

```
        opp-supported-hw = <0x4>;
    };
    opp@1200000000-0 {
        opp-hz = /bits/ 64 <1200000000>;
        opp-microvolt = <960000>;
        clock-latency-ns = <244144>; /* 8 32k periods */
        opp-supported-hw = <0x3>;
    };
    opp@1200000000-1 {
        opp-hz = /bits/ 64 <1200000000>;
        opp-microvolt = <1100000>;
        clock-latency-ns = <244144>; /* 8 32k periods */
        opp-supported-hw = <0x4>;
    };
    opp@1296000000 {
        opp-hz = /bits/ 64 <1296000000>;
        opp-microvolt = <1100000>;
        clock-latency-ns = <244144>; /* 8 32k periods */
        opp-supported-hw = <0x2>;
    };
    opp@1344000000 {
        opp-hz = /bits/ 64 <1344000000>;
        opp-microvolt = <1120000>;
        clock-latency-ns = <244144>; /* 8 32k periods */
        opp-supported-hw = <0x4>;
    };
    opp@1512000000 {
        opp-hz = /bits/ 64 <1512000000>;
        opp-microvolt = <1100000>;
        clock-latency-ns = <244144>; /* 8 32k periods */
        opp-supported-hw = <0x1>;
    };
};
```

compatible = "allwinner,sun50i-operating-points";： 用于匹配驱动的属性。
opp-hz： 某个频点的频率。
opp-microvolt： 频率对应的电压。
opp@480000000-0、opp@480000000-1： 后缀的0、1仅用于区分不同节点名字，以免报错。
opp-supported-hw： 选择该频点所支持的芯片版本。如“opp-supported-hw = <0x3>;”，表示该频点支持bit0、bit1所表示的芯片版本。详见内核文档Documentation/devicetree/bindings/opp/opp.txt关于opp-supported-hw的说明。
需要注意，因为不能存在多个相同频率的频点，所以要避免相同频率的频点都被选择的情况，如opp@480000000-0、opp@480000000-1不能被同一个芯片版本选择。

或 v-f 表：

```
    cpu_opp_l_table: opp_l_table {
        compatible = "allwinner,sun50i-operating-points";
        nvmem-cells = <&speedbin_efuse>, <&cpubin_efuse>;
        nvmem-cell-names = "speed", "bin";
        opp-shared;

        opp@408000000 {
            opp-hz = /bits/ 64 <408000000>;
            clock-latency-ns = <244144>; /* 8 32k periods */
            opp-microvolt-a0 = <900000>;
            opp-microvolt-a1 = <900000>;
            opp-microvolt-a2 = <900000>;
            opp-microvolt-b0 = <900000>;
            opp-microvolt-b1 = <900000>;
        };
```

```
opp@600000000 {
    opp-hz = /bits/ 64 <600000000>;
    clock-latency-ns = <244144>; /* 8 32k periods */
    opp-microvolt-a0 = <900000>;
    opp-microvolt-a1 = <900000>;
    opp-microvolt-a2 = <900000>;
    opp-microvolt-b0 = <900000>;
    opp-microvolt-b1 = <900000>;
};

opp@816000000 {
    opp-hz = /bits/ 64 <816000000>;
    clock-latency-ns = <244144>; /* 8 32k periods */
    opp-microvolt-a0 = <940000>;
    opp-microvolt-a1 = <900000>;
    opp-microvolt-a2 = <900000>;
    opp-microvolt-b0 = <900000>;
    opp-microvolt-b1 = <900000>;
};

opp@1008000000 {
    opp-hz = /bits/ 64 <1008000000>;
    clock-latency-ns = <244144>; /* 8 32k periods */
    opp-microvolt-a0 = <1020000>;
    opp-microvolt-a1 = <980000>;
    opp-microvolt-a2 = <950000>;
    opp-microvolt-b0 = <980000>;
    opp-microvolt-b1 = <950000>;
};

opp@1200000000 {
    opp-hz = /bits/ 64 <1200000000>;
    clock-latency-ns = <244144>; /* 8 32k periods */
    opp-microvolt-a0 = <1100000>;
    opp-microvolt-a1 = <1020000>;
    opp-microvolt-a2 = <1000000>;
    opp-microvolt-b0 = <1020000>;
    opp-microvolt-b1 = <1000000>;
};

opp@1320000000 {
    opp-hz = /bits/ 64 <1320000000>;
    clock-latency-ns = <244144>; /* 8 32k periods */
    opp-microvolt-a0 = <1160000>;
    opp-microvolt-a1 = <1060000>;
    opp-microvolt-a2 = <1030000>;
    opp-microvolt-b0 = <1060000>;
    opp-microvolt-b1 = <1030000>;
};

opp@1416000000 {
    opp-hz = /bits/ 64 <1416000000>;
    clock-latency-ns = <244144>; /* 8 32k periods */
    opp-microvolt-a0 = <1180000>;
    opp-microvolt-a1 = <1180000>;
    opp-microvolt-a2 = <1130000>;
    opp-microvolt-b0 = <1100000>;
    opp-microvolt-b1 = <1070000>;
};
```

```
            opp@1512000000 {
                opp-hz = /bits/ 64 <1512000000>;
                clock-latency-ns = <244144>; /* 8 32k periods */
                opp-microvolt-b0 = <1180000>;
                opp-microvolt-b1 = <1130000 1130000 1140000>;
            };

            opp@1608000000 {
                opp-hz = /bits/ 64 <1608000000>;
                clock-latency-ns = <244144>; /* 8 32k periods */
                opp-microvolt-b0 = <1180000>;
                opp-microvolt-b1 = <1130000 1130000 1140000>;
            };
    };

compatible = "allwinner,sun50i-operating-points";： 用于匹配驱动的属性。
opp-hz： 某个频点的频率。
opp-microvolt-x： 该频率下，x类型bin对应的电压。详见内核文档Documentation/devicetree/bindings/opp/opp.txt关于
      opp-microvolt-<name>的说明。
```

cpu 节点：

```
    cpu0: cpu@0 {
        device_type = "cpu";
        compatible = "arm,cortex-a53","arm,armv8";
        reg = <0x0 0x0>;
        enable-method = "psci";
        clocks = <&clk_pll_cpu>;
        operating-points-v2 = <&cpu_opp_l_table>;
        cpu-idle-states = <&CPU_SLEEP_0>;
        dynamic-power-coefficient = <100>;
        #cooling-cells = <2>;
    };

    cpu@1 {
        device_type = "cpu";
        compatible = "arm,cortex-a53","arm,armv8";
        reg = <0x0 0x1>;
        enable-method = "psci";
        clocks = <&clk_pll_cpu>;
        operating-points-v2 = <&cpu_opp_l_table>;
        cpu-idle-states = <&CPU_SLEEP_0>;
        #cooling-cells = <2>;
    };

operating-points-v2 = <&cpu_opp_l_table>;：引用v-f表。
```

对于 sun8iw18p1 这个没有使用标准 pwm regulator 驱动的平台，与没有 cpu 分 bin 需求的平台基本一样，不再赘述。

2.3.2 board.dts 配置说明

board.dts 用于保存每一个板级平台的设备信息（如 demo 板，perf1 板等），里面的配置信息会覆盖上面的 Device Tree 默认配置信息。

cpu 节点（具体参考相应的 dts 文件）：

```
&cpu0 {
   cpu-supply = <&reg_dcdc2>;
};
```

2.3.3 kernel menuconfig 配置说明

在命令行中进入 linux 目录，执行 make ARCH=arm64 menuconfig(32 位系统为 make ARCH=arm menuconfig) 进入配置主界面 (Linux-5.4 及以上内核版本执行：./build.sh menuconfig)，并按以下步骤操作。

对于 sun8iw19p1 等没有 cpu 分 bin 需求的平台：

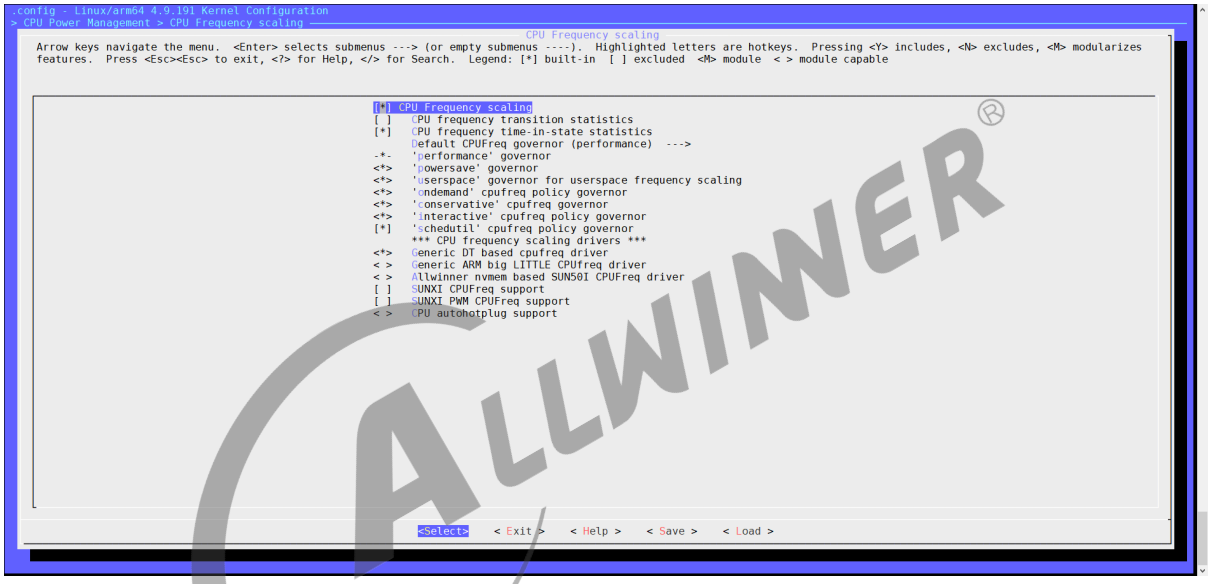

图 2-1: 配置图 1

对于 sun50iw9p1、sun50iw10p1 等有 cpu 分 bin 需求的平台：

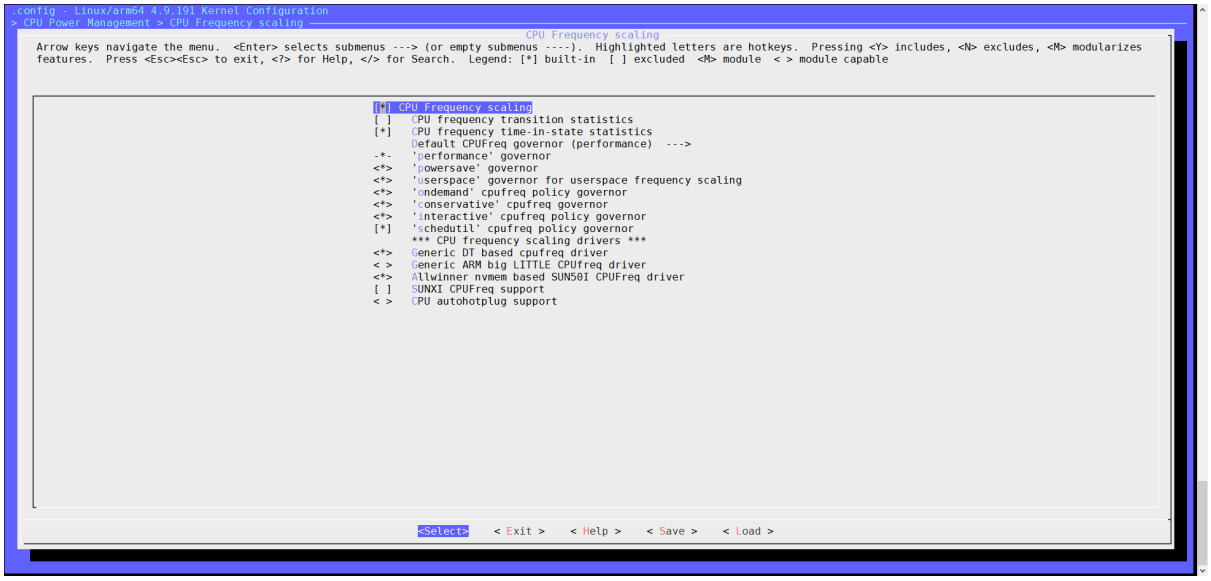

图 2-2: 配置图 2

对于 sun8iw18p1 这个没有使用标准 pwm regulator 驱动的平台：

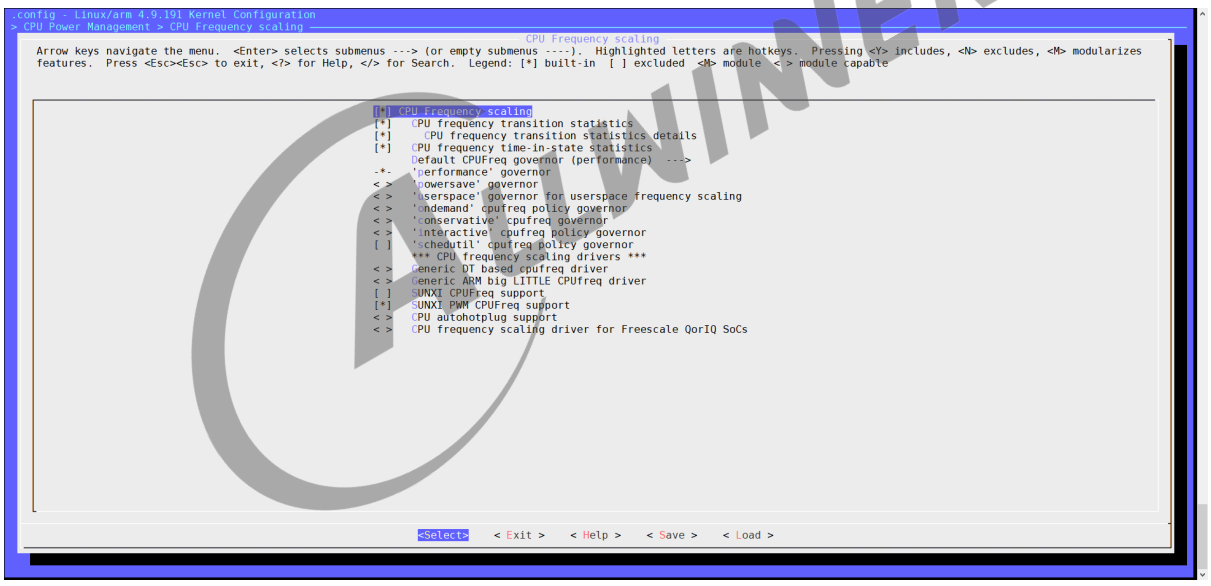

图 2-3: 配置图 3

对于 sun65iw1 这类支持 cpu 硬件调频的平台：

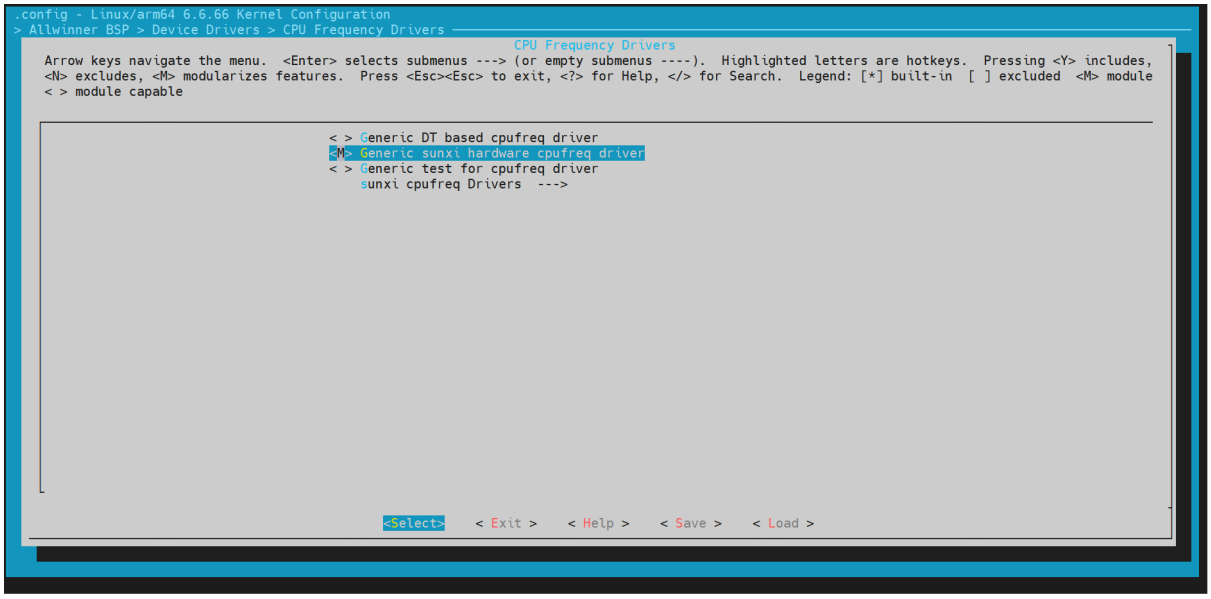

图 2-4: 配置图 4

图 2-5: 配置图 5

注：使用不同目录下的 cpufreq 驱动需要注意如下配置项：

1.使用longan/kernel/linux-xxx/drivers/cpufreq/目录下的cpufreq驱动配置项:
CONFIG_CPUFREQ_DT && CONFIG_CPUFREQ_DT_PLATDEV
2.使用longan/bsp/drivers/cpufreq/cpufreq-linux-xxx目录下的cpufreq驱动配置项:
CONFIG_AW_CPUFREQ_DT && CONFIG_AW_CPUFREQ_DT_PLATDEV
3.使用cpu硬件调频，需要关闭CONFIG_AW_CPUFREQ_DT，打开CONFIG_AW_HW_CPUFREQ &&
CONFIG_ARM_AW_SUN50I_CPUFREQ_NVMEM，最终生成sunxi-cpufreq-hw.ko文件

2.4 源码结构介绍

CPUFREQ 的源代码位于内核 drivers/cpufreq/目录下：

```
bsp/drivers/cpufreq/
├── cpufreq-linux-5.10
│   ├── cpufreq-dt.c
│   ├── cpufreq-dt.h
│   ├── cpufreq-dt-platdev.c
│   ├── Kconfig
│   ├── Kconfig.arm
│   └── Makefile
├── cpufreq-linux-5.15
│   ├── cpufreq-dt.c
│   ├── cpufreq-dt.h
│   ├── cpufreq-dt-platdev.c
│   ├── Kconfig
│   ├── Kconfig.arm
│   └── Makefile
├── cpufreq-linux-5.4
│   ├── Kconfig
│   └── Makefile
├── cpufreq-linux-6.1
│   ├── cpufreq-dt.c
│   ├── cpufreq-dt.h
│   ├── cpufreq-dt-platdev.c
│   ├── Kconfig
│   ├── Kconfig.arm
│   └── Makefile
├── cpufreq-linux-6.6
│   ├── cpufreq-dt.c
│   ├── cpufreq-dt.h
│   ├── cpufreq-dt-platdev.c
│   ├── Kconfig
│   ├── Kconfig.arm
│   ├── Makefile
│   └── sunxi-cpufreq-hw.c
├── Kconfig
├── Makefile
├── sun50i-cpufreq-nvmem.c
└── vf-test.c
```

cpufreq-dt.c 为调频调压功能实现代码。
cpufreq-dt-platdev.c 为匹配没有 cpu 分 bin 需求的平台的代码。
sun50i-cpufreq-nvmem.c 为匹配有 cpu 分 bin 需求的平台的代码，它依赖于 nvmem 模块驱动提供芯片版本信息。

2.5 驱动框架介绍

无。

3 FAQ

3.1 调试方法

3.1.1 调试节点

节点	权限	说明
scaling_setspeed	R/W	设置频率的接口，仅当调频策略为 userspace 时可用
scaling_governor	R/W	调频策略
scaling_max_freq	R/W	软件调频最大频率
scaling_min_freq	R/W	软件调频最小频率
affected_cpus	R	受到该调频策略影响且在线的 cpu
related_cpus	R	受到该调频策略影响的所有 cpu
scaling_cur_freq	R	当前频率
scaling_driver	R	调频驱动名称
scaling_available_governors	R	可用调频策略
cpuinfo_transition_latency	R	频率转换延迟
cpuinfo_max_freq	R	硬件最大频率
cpuinfo_min_freq	R	硬件最小频率
cpuinfo_cur_freq	R	硬件实际运行频率

节点位于/sys/devices/system/cpu/cpufreq/policyX (X = 0,4,6，根据方案是否支持大中核而定)下

3.2 常见问题

3.2.1 调频策略使用说明

governor	说明
powersace	节能
performance	性能
userspace	由用户控制

governor	说明
ondemand	按需
conservative	保守，适用于带电池设备
schedutil	利用调度器提供信息进行调频，与 EAS 调度器一起配合使用

选择合适的governor，并勾选对应的调频驱动即可。

如果需要手动更改频率，选择governor为userspace。
echo userspace > /sys/devices/system/cpu/cpufreq/policy0/scaling_governor

需要注意，CPU调频功能会受到温控功能影响。所以如果有自主调频而不受温控影响的需求，要关闭温控功能。详细参考《Linux_Thermal_开发指南》。

3.2.2 怎样获取当前使用的电压频率表

方法一：使用 sunxi 自定义节点 cpufreq_table。
注意，这个节点并不是每个平台都支持（Linux-5.4 及以上内核版本不支持），是需要 kernel 编译 drivers/soc/sunxi/vf-table.c 才有。

```
/ # mount -t debugfs none /sys/kernel/debug/
/ # cat /sys/kernel/debug/cpufreq_table
freq(kHz)      vol(mv)
-----------------------
408000        900
600000        900
816000        900
1008000       980
1200000       1020
1320000       1060
1416000       1180
```

方法二：使用内核原生的 opp 节点。

```
/ # mount -t debugfs none /sys/kernel/debug/

/* 对于Linux4.9 */
/* 获取所有频点的频率，单位是Hz */
/ # cat /sys/kernel/debug/opp/cpu0/opp*/rate_hz
1008000000
1200000000
1320000000
1416000000
1464000000
1512000000
1608000000
408000000
600000000
816000000

/* 获取所有频点的电压，单位是mV */
/ # cat /sys/kernel/debug/opp/cpu0/opp*/u_volt_target
1020000
1100000
```

```
1160000
0
1180000
0
0
900000
900000
940000

/* 对于Linux5.4及以上版本 */
/* 获取所有频点的频率，单位是Hz */
/ # cat /sys/kernel/debug/opp/cpu0/opp*/rate_hz
1008000000
1200000000
1320000000
1464000000
408000000
600000000
816000000

/* 获取所有频点的电压，单位是mV */
/ # cat /sys/kernel/debug/opp/cpu0/opp*/supply-0/u_volt_target
1020000
1100000
1160000
1180000
900000
900000
940000
```

3.2.3 怎样修改电压频率表

原则上不允许私自修改电压频率表，如下方法仅供调试使用。

方法一：直接修改 dts 文件中的 v-f 表，再重新编译固件。具体参考 Device Tree 配置说明。

方法二：在 uboot 修改 v-f 表，这样不需要重新编译固件。但是安全固件是不支持修改保存，所以机器重启后修改失效。

以某个平台为例，说明如何通过 uboot 修改 v-f 表。其他平台的节点路径和命名可能不同，具体参考 Device Tree 配置说明。

进入 uboot 命令行：

```
/* 获取v-f表 */
=> fdt list /opp_l_table
opp_l_table {
    compatible = "allwinner,sun50i-operating-points";
    nvmem-cells = <0x000000ff 0x00000100>;
    nvmem-cell-names = "speed", "bin";
    opp-shared;
    linux,phandle = <0x000000fc>;
    phandle = <0x000000fc>;
    opp@408000000 {
    };
```

```
        opp@600000000 {
        };
        opp@816000000 {
        };
        opp@1008000000 {
        };
        opp@1200000000 {
        };
        opp@1320000000 {
        };
        opp@1416000000 {
        };
        opp@1464000000 {
        };
        opp@1512000000 {
        };
        opp@1608000000 {
        };
};

/* 删除不想要的频点，如408MHz的频点 */
=> fdt rm /opp_l_table/opp@408000000

/* 保存修改，注意安全固件是不支持修改保存 */
=> fdt save

/* 从uboot启动到内核后，确认频点修改是否正确 */
/ # cat /sys/devices/system/cpu/cpufreq/policy0/scaling_available_frequencies
600000 816000 1008000 1200000 1320000 1464000
```

3.2.4 怎样验证 cpufreq 电压频率

先获取当前使用的电压频率表，再调节至某个频点，读出 CPU 时钟寄存器确认 CPU 频率正确，使用万用表实测 CPU 电压正确，验证时注意覆盖每个频点。

```
/ # echo userspace > /sys/devices/system/cpu/cpufreq/policy0/scaling_governor
/ # cat /sys/devices/system/cpu/cpufreq/policy0/scaling_available_frequencies
408000 600000 816000 1008000 1200000 1320000 1416000
/ # echo 1200000 > /sys/devices/system/cpu/cpufreq/policy0/scaling_setspeed
/ # cat /sys/devices/system/cpu/cpufreq/policy0/scaling_cur_freq
1200000
```

著作权声明

商标声明

免责声明