A Generic Linux CPUFreq Driver for ARM SoCs

http://dx.doi.org/10.3991/ijoe.v9iS6.2797

Lei Zhou[1], Qiang Lv[2] and Shengchao Guo[3]
[1] Changshu Institute of Technology, Changshu, China
[2] Soochow University, Suzhou, China
[3] Freescale Semiconductor, Inc., Shanghai, China

***Abstract*—Linux CPUFreq subsystem provides a framework for implementing Dynamic Voltage and Frequency Scaling (DVFS) to prolong batter life of mobile devices. Instead of creating hardware specific CPUFreq driver for every single ARM System on Chip (SoC) from different vendors, this paper presents the design and implementation of a generic CPUFreq driver. Managing the hardware specific clock and voltage details via Linux Common Clock Framework and Regulator subsystem, the driver can scale CPU frequency and voltage in a generic way, and thus should work for the majority of the ARM SoCs today. Freescale i.MX6 Quad was taken as the target hardware to develop and test the driver. A measurement on the hardware reports 37% CPU power saving in a typical video playback application. The feedback from Linux community tells that the driver works for OMAP and Calxeda processors as well, and hence the driver was merged into Linux 3.7 release as a generic CPUFreq driver for ARM SoCs.**

***Index Terms*—ARM, CPUFreq, DVFS, i.MX6 Quad.**

I. INTRODUCTION

With the popularity of mobile computing, people increasingly want their devices run a longer battery life without any user experience downgrading. That's why power efficiency becomes a very important fact of system design and implementation. Besides the traditional static power management [1], which manages power for particular system states like Standby, Suspend and Hibernate, system design starts adopting Dynamic Power Management (DPM) [2] to aggressively manage power consumption at runtime. Being one of the most typical DPM technologies, Dynamic Voltage and Frequency Scaling (DVFS) is widely adopted by both hardware and software design, and proved to be an effective way to reduce system power consumption without noticeable performance loss [3, 4, 5].

As the most successful processor architecture in mobile market, ARM becomes the dominator on devices like phones and tablets, primarily because of its excellent low power performance. Being one of the well known ARM System-on-Chip (SoC) vendors, Freescale offers i.MX 6Quad processor for mobile applications [6]. This SoC integrates ARM Cortex-A9 quad cores to provide high performance, and meanwhile it's designed with many DPM considerations to provide the lowest power consumption. For DVFS example, the clock frequency and power supply to CPU core can be scaled on the fly simply with a few register accesses. On the software side, Linux becomes the most popular operation system used on mobile devices, not only because it's free and open but also it offers great power management support. Linux CPUFreq subsystem provides a framework for different CPU to implement DVFS function based on low level hardware support [7, 8]. What authors initially try to do is only to implement a CPUFreq driver for i.MX6 Quad SoC. After studying a few other ARM SoC vendors' CPUFreq drivers, authors found that there are a lot of commonalities among these drivers, and decided to design and implement a generic CPUFreq driver rather than creating yet another vendor specific one.

This paper is organized as follows. Section II introduces the Linux CPUFreq subsystem, including how CPUFreq governors work and the registration of CPUFreq drivers. Section III elaborates the design principles and implementation details of the generic cpufreq-cpu0 driver. It introduces the prerequisites of cpufreq-cpu0 driver, CLK framework, Regulator subsystem, OPP library, and then illustrates how they work for cpufreq-cpu0 driver in the cpu0_set_target() procedure. Section IV gives the testing result of running cpufreq-cpu0 driver on i.MX6 Quad in a couple of typical use cases, web browsing and video playback. Finally, Section V shares the feedback from Linux community on this generic driver.

II. LINUX CPUFREQ SUBSYSTEM

To support DVFS implementation on different CPU architectures, CPUFreq subsystem was introduced on Linux 2.6 with a layered design illustrated in Fig. 1.

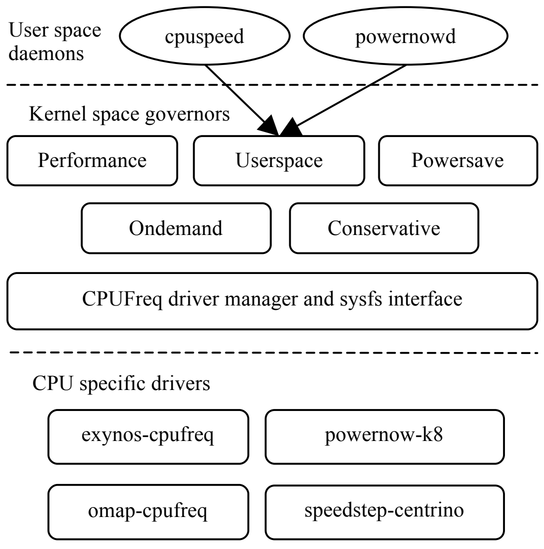

Figure 1. CPUFreq framework architecture

Designed in such a hierarchical architecture, CPUFreq subsystem can effectively separate two aspects of DVFS, policy and mechanism. Policy means when to scale voltage and frequency, which is a decision made by CPUFreq governors. And mechanism means how to scale, that is an implementation provided by individual CPUFreq driver. Policy is CPU independent, while mechanism is CPU specific. And user space will see a unified interface no matter which CPU the kernel is running on, since all the CPU specific details is handled by underneath CPUFreq driver.

A. CPUFreq Governors

There are totally five governors provided by Linux CPUFreq framework as shown in Fig. 1. They are designed for different use cases and can be switched at run-time using sysfs interface. The governor Performance sets the CPU statically to the highest frequency *scaling_max_freq*, while Powersave sets CPU to the lowest frequency *scaling_min_freq*. The governor Userspace sets the CPU to the frequency specified by the user space program, typically like daemon cpuspeed or powernowd as shown in Fig. 1, through sysfs entry *scaling_setspeed*. Ondemand checks the CPU load regularly, and sets the CPU to run at the highest frequency when the load rises above *up_threshold*. When the load falls below the same threshold, it sets the CPU to run at the next lowest frequency. Similar to Ondemand, Conservative also checks the CPU load. But when the load rises above *up_threshold*, it sets the CPU to run at the next highest frequency, and when the load falls below *down_threshold*, it sets the CPU to run at the next lowest frequency.

B. CPUFreq Governors

In CPUFreq framework, CPUFreq driver is the one who actually accesses hardware to have CPU frequency and voltage scaled. A CPUFreq driver is added to the framework by registering a *struct cpufreq_driver* to the CPUFreq core using *cpufreq_register_driver()*. The most important members of the structure are three function hooks: *init*, *verify* and *target*.

The *init* is a per-CPU initialization hook, which will be called whenever a new CPU is registered with the device model. It takes a *struct cpufreq_policy *policy* as argument and should set *policy->cur* as the current CPU operating frequency, and set up a few other members in *policy* and *policy->cpuinfo* as well. The *verify* hook is called to validate the frequency table when the user decides on a policy. The *target* function hook takes the new frequency to be set on the CPU as argument, and accesses hardware to scale the CPU frequency to the one requested.

III. GENERIC CPUFREQ DRIVER

As a typical ARM Symmetric Multiprocessing (SMP) multi-core SoC, i.MX6 Quad has all four Cortex-A9 cores in one cluster and share the same clock and voltage source. In terms of DVFS support, that means all the CPUs will be scaled together. With some coordination at software level, the DVFS support on such SMP system can be effectively implemented by managing the CPU0 frequency and voltage scaling. Such hardware design is commonly found on most of ARM multi-core SoCs, OMAP4 from TI, Exynos from Samsung, etc. So if there are generic interfaces for CPUFreq driver to operate on clock and voltage in spite of the differences at hardware level, the CPUFreq driver can also be generic on all this type of ARM SoCs. Fortunately, Linux kernel has APIs for clock and voltage management: CLK framework, Regulator subsystem and Operating Performance Points (OPP) library. They can act as the interface between the generic cpufreq-cpu0 driver and the actual hardware as shown in Fig. 2.

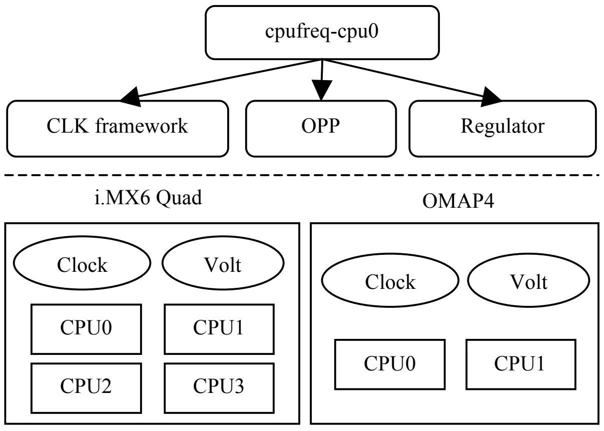

Figure 2. Generic cpufreq-cpu0 driver architecture

The reason cpufreq-cpu0 driver can be generic is that the low level hardware differences are all handled by the underneath support infrastructures, CLK framework, Regulator subsystem and OPP library. The Common CLK Framework (CCF) was introduced on Linux 3.4 and provides a suite of APIs defined in *include/linux/clk.h* for device drivers to manage clock. The cpufreq-cpu0 driver can use *clk_get_rate()*, *clk_round_rate()* and *clk_set_rate()* to control CPU frequency. Similar to CCF, the Regulator subsystem is designed to provide a standard kernel interface to control voltage regulators. On i.MX6 Quad the CPU voltage is controlled by module Power Management Unit (PMU). And PMU should be implemented as a regulator driver and registering the CPU regulator to the core, so that cpufreq-cpu0 driver can call regulator APIs *regulator_set_voltage_tol()* to change CPU voltage and *regulator_set_voltage_time()* to query the voltage ramping up time which should be a factor of frequency transition latency. OPP library provides a set of useful helper functions that can be found in *include/linux/opp.h* to read the operating point data from device tree [9], organize it in OPP, and help CPUFreq driver to query and retrieve the data conveniently.

As described in Section II, a CUPFreq driver is essentially a piece of code that implements a struct *cpufreq_driver* and calls *cpufreq_register_driver()* to register the structure to CPUFreq framework. And the most important part of *struct cpufreq_driver* is the functions hook *target*, which is *cpu0_set_target()* in cpufreq-cpu0 case. Whenever CPUFreq governor requests to transit to a new target frequency, the function will call CLK and Regulator APIs to change CPU frequency and voltage as requested. The flow chart in Fig. 3 illustrates the primary steps of the procedure.

It's very dangerous to have hardware run at a frequency with a voltage that is lower than required, even for a very short period. That's why the sequence of scaling voltage and frequency is different between frequency increasing and decreasing case.

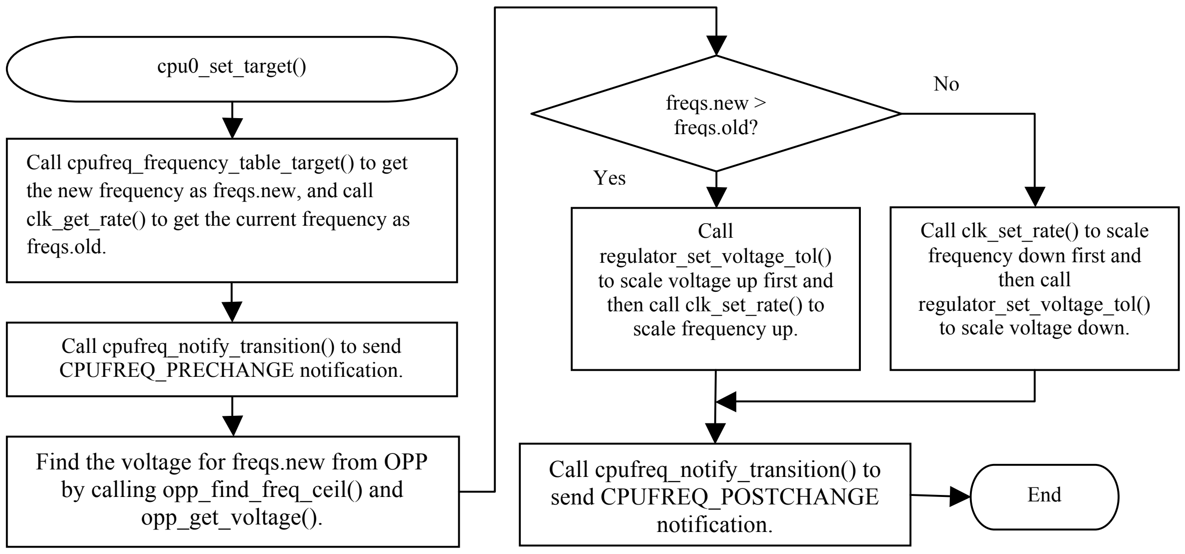
cpu0_set_target()
Call cpufreq_frequency_table_target() to get the new frequency as freqs.new, and call clk_get_rate() to get the current frequency as freqs.old.
Call cpufreq_notify_transition() to send CPUFREQ_PRECHANGE notification.
Find the voltage for freqs.new from OPP by calling opp_find_freq_ceil() and opp_get_voltage().
freqs.new > freqs.old?
Yes
No
Call regulator_set_voltage_tol() to scale voltage up first and then call clk_set_rate() to scale frequency up.
Call clk_set_rate() to scale frequency down first and then call regulator_set_voltage_tol() to scale voltage down.
Call cpufreq_notify_transition() to send CPUFREQ_POSTCHANGE notification.
End

Figure 3. cpu0_set_target() flow chart

The CPU frequency is concerned by other components in Linux kernel, like the global variable *loops_per_jiffy* which is used to implement miscellaneous delay functions *udelay()*, and *mdelay()*. The variable is calculated based on CPU frequency, so has to be updated accordingly whenever CPU frequency changes. The notifications *CPUFREQ_PRECHANGE* and *CPUFREQ_POSTCHANGE* shown in Fig. 3 are used to inform the change of CPU frequency, so that others can have a mechanism to react to the CPU frequency changes.

IV. MEASUREMENT AND DISCUSSION

Authors added a CPUFreq support for Freescale i.MX6 Quad SoC based on the generic cpufreq-cpu0 driver. It uses the operating points settings recommended by Freescale, 198/396/792 MHz CPU frequencies with the coupled voltage at 0.85/0.95/1.1 V. Measurement on i.MX6 Quad SABRE Smart Device (SabreSD) reference design board demonstrates that with Ondemand governor the driver can effectively reduce the power consumption of CPU cores without noticeable performance loss.

A. Measurement Setup

The i.MX6 Quad SabreSD board is designed as the reference for tablets. It's equipped with eMMC as the storage and LVDS as the display device to support running those popular desktop systems. Ubuntu 12.04 LST is chosen as the desktop environment for the measurement, as it provides a plenty of handy applications for measuring power in typical use cases like web browsing, video playback. Furthermore, the board has a design consideration to ease the power measurement. There is a 0.02 ohm current sensing resistor on VDDCORE power supply, as illustrated in Fig. 4.

From the identity P = V * I, the power of ARM cores will be calculated out if the current can be measured, as VDDCORE is known as 1.425 V. With that 0.02 ohm current sensing resistor R on board, authors can measure the voltage drop VΔ across the resister with a voltmeter, and then calculate the current from identity I = VΔ / R. Fluke 117C Digital Multimeter is used in the measuring, as it provides 0.1 mV precision and supports averaging function.

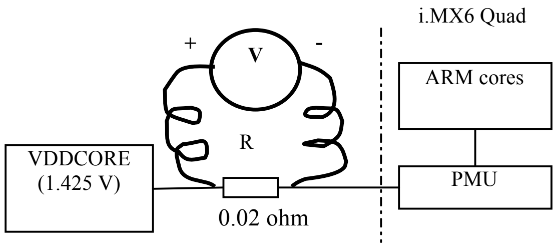

Figure 4. Power measurement setup

B. Web Browsing Test

Web browsing is a typical application where user experience is very important. People browsing web pages are generally sensible to the page load time. Authors attempt to see if CPUFreq support can gain some power saving without noticeable user experience impact. In this test, browser Firefox is used to load the page http://www.mozilla.org/en-US/ with 3 different CPUFreq governors. The test iteration consists of the following steps.

1. Go to Firefox menu Tools, Clear Recent History… to clean the browser cache.
2. Open page http://www.mozilla.org/en-US/ and start measuring voltage drop VΔ using at the same time.
3. Keep looking at the page load percent shown on Firefox status bar (Extended Statusbar Add-on installation is required), and stop voltage measuring as soon as the page load completes.
4. Read the page load time (T) from Firefox status bar and read average voltage (VΔ) from voltmeter.

The test results are collected in Table 1. In case of CPUFreq governor being Performance, CPU will always run at 792 MHz, which is equivalent to that CPUFreq is not enabled. So the last row can be treated as the result of no CPUFreq support.

As shown Table 1, the power (P) of Powersave case is %77 less than Performance. But since Powersave always forces CPU to run at the lowest frequency 198 MHz, it will take much longer time to finish a job. In the test, it takes almost triple time than Performance to complete the page load. Even though it eventually saves 29% power consumption (P * T) than Performance, it's not an ideal policy choice, because it takes so much time to do the work and will impact user experience badly. In comparison, Ondemand governor plays well in the balance between power saving and performance impact. It takes a little longer time (0.365 s) that people do not feel about to load the page, while saves %6 power consumption than Performance.

C. Video Playback Test

Video playback is another typical application on mobile devices, and it's used by many benchmarks to evaluate the battery life of devices. In this test, a 1080p H.264 video clip stored on eMMC card is played to see the effect of CPUFreq support on i.MX6 Quad. Table 2 gives the power data for Performance and Ondemand governors.

As the time of playing a given video clip is fixed, the P(mW) can essentially stand for the power consumption in that case. So the data tells that CPUFreq with Ondemand governor can approximately save 37% power consumption than no CPUFreq support. It also tells that video playback on i.MX6 Quad system does not necessarily need to keep CPU cores running at 792 MHz all the time. It can be reasonably explained by the fact that i.MX6 Quad has integrated Video Processing Unit (VPU) for video decoding and built DMA support for all those high speed peripherals. Using a script to launch the video player and check the sysfs entry */sys/devices/system/cpu/cpu0/cpufreq/stats/time_in_state* at the beginning and ending of the playback respectively, we can calculate the time spent in each state during the playback. As shown in Fig 5, there is 34% time in total that CPU cores haven't been running on 792 MHz, and it explains the power saving that's measured above.

TABLE I.
WEB BROWSING POWER DATA

Governor	V∆ (mV)	P (mW) = 1.425 * V∆ / 0.02	T (s)	P * T
Powersave	1.5	106.875	14.603	1560.696
Ondemand	5.7	406.125	5.078	2062.303
Performance	6.5	463.125	4.713	2182.708

TABLE II.
VIDEO PLAYBACK POWER DATA

Governor	V∆ (mV)	P (mW) = 1.425 * V∆ / 0.02
Ondemand	2.4	171.00
Performance	3.8	270.75

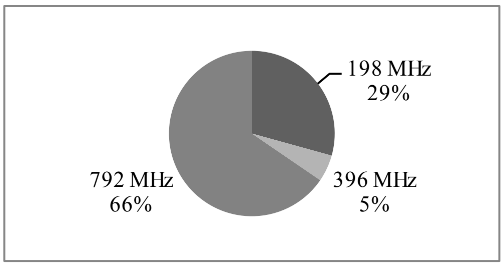

Figure 5. Time percentage in different states

I. LINUX COMMUNITY FEEDBACK

After verifying that the generic cpufreq-cpu0 driver can effectively reduce system runtime power consumption, authors submitted it to Linux community for mainline inclusion. The feedback from community people tells that the driver works well on other ARM SoCs too both single core and SMP systems, like TI OMAP, Calxeda Highbank processors. While achieving the goal of power saving, this generic driver avoids a lot of code duplication and eases the long term maintenance of CPUFreq drivers. As the result, it ' s accepted by Linux CPUFreq subsystem maintainer and finally merged into Linux 3.7 release.

REFERENCES

[1] Patrick Mochel, "Linux Kernel Power Management," *Proceedings of the Linux Symposium, Ottawa, Canada,* 23-26, July, 2003, pp. 326-339.

[2] L. Benini, A. Bogliolo, and G. D. Micheli, "A Survey of Design Techniques for System-Level Dynamic Power Management," *IEEE Transactions on VLSI Systems,* Vol. 8, Issue 3, 2000, pp. 299-316. http://dx.doi.org/10.1109/92.845896

[3] Zhang YuHua, Qian LongHua, Lv Qiang, "A Practical Dynamic Frequency Scaling Solution to DPM in Embedded Linux Systems," *Journal of Computers,* Vol. 4, Issue 8, 2009, pp. 787-793.

[4] Lu Chunpeng, "Function of Dynamic Voltage and Frequency Scaling in Power Reduction," *Microcontrollers & Embedded Systems,* Issue 5, 2007, pp. 12-15.

[5] HUANG Jianke, ZHOU Yun, "Research of Low Power SoC Technology Based on Self-adaptive DVFS," *Modern Electronics Technique,* Vol. 32, Issue 7, 2009, pp. 120-122.

[6] Freescale Semiconductor, Inc., *i.MX 6Dual/6Quad Applications Processor Reference Manual,* 2012.

[7] Jenifer Hopper, IBM, *Reduce Linux Power Consumption,* 2009. http://www.ibm.com/developerworks/linux/library/l-cpufreq-1/

[8] GU Li-hong, LIN Zhi-qiang, WU Shao-gang, "Research and Implementation of Software Layer Dynamic Frequency Scaling Based on Loongson," *Computer Engineering,* Vol.37, Issue 9, 2011, pp. 41-43.

[9] Grant Likely, Josh Boyer, "A Symphony of Flavours: Using the device tree to describe embedded hardware," *Proceedings of the Linux Symposium, Ottawa, Canada,* 23-26, July, 2008, pp. 27-38.

AUTHORS

Lei Zhou is with School of Computer Science and Engineering, Changshu Institute of Technology, No. 99, South Third Ring Road, Changshu, Jiangsu Province, 215500, China (e-mail: zhoulei@cslg.edu.cn).

Qiang Lv is with School of Computer Science and Engineering, Changshu Institute of Technology, No. 99, South Third Ring Road, Changshu, Jiangsu Province, 215500, China (e-mail: qiang@suda.edu.cn).

Shengchao Guo is with Freescale Semiconductor (Shanghai Branch), Inc. No. 192 Liangjing Road, Pudong New Area, Shanghai, 201203, China (e-mail: shawn.guo@freescale.com).

This work was supported in part by the National Natural Science Foundation of China under Grant 60970055. It is an extended and modified version of a paper presented at the 2012 International Conference on Artificial Intelligence and Its Application in Industry Production (AIAIP 2012), held in Wuhan, China in December 2012. Manuscript received 14 May 2013. Published as submitted by the authors 26 June 2013.