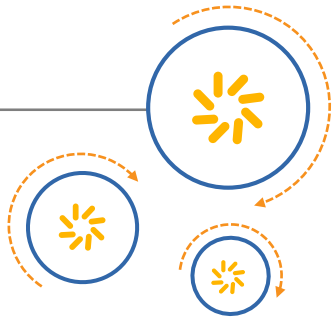
Qualcomm Technologies, Inc.



Core Control Feature

Application Note

80-P0106-1 A

March 25, 2015

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision history

Revision	Date	Description
A	Mar 2015	Initial release.

Contents

1 Introduction

1.1 Purpose

This document describes the operation of Core Control, including its logic implementation and tunable parameters.

1.2 Conventions

Function declarations, function names, type declarations, attributes, and code samples appear in a different font, for example, `#include`.

Code variables appear in angle brackets, for example, `<number>`.

Commands to be entered appear in a different font, for example, **`copy a:*.* b:`**.

Button and key names appear in bold font, for example, click **Save** or press **Enter**.

1.3 Technical assistance

For assistance or clarification on information in this document, submit a case to Qualcomm Technologies, Inc. (QTI) at https://support.cdmatech.com/.

If you do not have access to the CDMATech Support website, register for access or send email to support.cdmatech@qti.qualcomm.com.

2 Purpose of Secondary Boot Image

Core Control has been introduced to preemptively offline cores, and only online them when necessary. This resembles mpdecision, although implementation is not the same.

2.1 Benefits

The goal of Core Control is to improve thermal conditions and save power by keeping cores offline unless they are required. This also helps to improve the user experience performance.

2.2 Limitations

Load-based Core Control logic is compatible with either a scheduler-guided governor or legacy governor as long as the governor notification patch is included. Currently only the interactive governor has changes to support this load reporting; therefore, Core Control will not work with other governors.

3 Core Control tuning

This chapter describes the Core Control tuning parameters. Once the Core Control module is inserted, the "core_ctl" folder is visible under /sys/devices/system/cpu/cpuX/, where X is the CPU number of first CPU inside cluster. For example, on MSM8994, you will find this folder in "/sys/devices/system/cpu/cpu0/" and "/sys/devices/system/cpu/cpu4/".

3.1 Tuning parameters

Sl No.	Tuning Parameter	Description
1	min_cpus	Minimum number of CPUs to keep online. This overrides other heuristics, and can be used to set hard requirements from user space.
2	max_cpus	Maximum number of CPUs to keep online. This overrides other heuristics, and can be used to set hard requirements from user space.
3	busy_up_thres	The normalized load% threshold that the CPU load should exceed for the CPU to transit to busy state. It could be a single threshold for all CPUs in a group, or num_cpus thresholds separated by spaces to specify different thresholds based on the current number of online CPUs. For example, "60" means when load exceeds 60%, consider a core as busy. Use 50, 60, 70, or 80 for 8994 big cluster, which means when there is 1 core online, 50%+ is considered as busy. When there are two cores online, 60%+ load is busy, etc. Be aware that the last number has no effect because there are no more cores to online even if all cores are busy.
4	busy_down_thres	The normalized load% threshold that the CPU loads should be lower than for the CPU to transit to "not busy" state. It could be a single threshold for all CPUs in a group, or num_cpus thresholds separated by spaces to specify different thresholds based on the current number of online CPUs.
5	offline_delay_ms	The time to wait for before offlining cores when the number of needed CPUs goes down. The current default is 100ms.
6	is_big_cluster	This is used to tell Core Control which cluster is the big cluster when collecting running tasks information. This must be set for cluster based targets. There is no correct default value.
7	task_thres	This is the threshold where all cores will be onlined immediately. Core Control will reject any values that are lower than the number of cores in the cluster. Default is UINT_MAX which effectively removes this functionality. For 8994, 4 is used for big cluster. NOTE: UINT_MAX is defined as 0xFFFFFFFF
8	/sys/module/core_ctl/parameters/rq_avg_period_ms	Window size of calling into the scheduler API to get averaged task load. This is usually the same as the window size of governor.

NOTE: Both busy_up_thres and busy_down_thres are transition thresholds, which means they only affect the core's state when the threshold is met. For example, busy_up_thres is 60, and busy_down_thres is 30. The core's state is busy if load >= 60, and is not busy if load < 30. If the load is between 30-60, its state remains the same as the previous evaluation (which could be busy or not busy).

3.2 Tuning decisions

Major tuning would be busy_up_thres or busy_down_thres. The recommended busy_down_thres has been 30 for 8994; busy_up_thres is 60 for 8994, and 67 for 8939. The logic is to online cores when the cluster is close to 100% busy, running at hispeed_freq. Thus busy_up_thres is set to be slightly below hispeed_freq divided by max_freq.

It is possible to use variable busy_up/down_thres based on current online cores. This potentially makes the core 3/4 harder to online compared to core 2 to further save power.

task_thres needs to be carefully tuned (or just disabled by leaving it at UINT_MAX) for the little cluster/SMP, because multiple small tasks can be packed onto a single CPU. The task count could be quite different from the number of needed cores in this case.

4 Software implementation

Core Control is implemented in two phases. One is the load based Core Control and the other one is based on task information from scheduler.

4.1 Load-based Core Control

- Core Control shall turn the cores online or offline based on the load of current online cores.

This logic is applicable to per-cluster cores, i.e., each cluster only evaluates load and online/offline cores within the same cluster.

Core Control gets load information from the governor, and thus is compatible with either the scheduler guided governor or the legacy governor, as long as the governor notification patch is included.

NOTE: Only the interactive governor has changes to support this load reporting, therefore Core Control will not work with other governors.

Core Control defines “up threshold (busy_up_thres)” and “down threshold (busy_down_thres)” for the CPU load. A core is considered "busy" when its load exceeds up threshold. A core is considered "not busy" when its load drops below down threshold. If the load is between up and down threshold, its state remains same as the previous evaluation. The load is normalized to current policy->max of that CPU/cluster.

Each time the governor updates load for one CPU, the entire cluster's state is re-evaluated. Core Control counts the total number of "busy" cores inside the cluster, and adjusts the final needed CPUs after considering task count (refer to next section).

If “needed CPUs” increases since the previous evaluation, Core Control immediately wakes up the hotplug thread and tries to adjust total number of online cores to “needed CPUs”. It tries to do core rotation to keep the temperature low/distributed by preferring to online cores that have been offline for the longest time.

If “needed CPUs” decreases since the last evaluation, it waits for “offline delay ms” from the time the “needed CPUs” count starts dropping before it actually offlines any cores. It offlines cores that are currently considered "not busy" to match the number of online cores to needed CPUs. In case “needed CPUs” conflicts with min_cpus and max_cpus set by userspace, Core Control will always respect the userspace limit.

4.2 Core Control based on task information from scheduler

Core Control gets additional information from the scheduler so that cores are only onlined when there is indeed more work to run. Task information can also be used to online multiple cores simultaneously instead of one by one.

This implementation would use an averaged running task (nr_running) count from the scheduler.

The number of running tasks are saved to each cluster's per-cpu field. If this number changes, it re-evaluates the need of cluster core count. Evaluation first goes through the same logic as in load-based Core Control, determining the number of "busy" cores. Once that is determined, it adjusts the final number based on the number of running tasks.

If the number of running tasks equals to or exceeds a user-defined threshold (task_thres), all cores are onlined immediately. This effectively overrides the load-based metric to accommodate the performance requirement for MT benchmarks or a burst of sudden heavy load starting. All cores are onlined together for these workloads. If the number of running tasks is greater than the "busy" cores, but lower than the threshold, one additional core is kept online. Otherwise, no additional cores are kept online.

5 Enabling Core Control and verification

This section describe the methods used to enable Core Control in the build and to verify if this feature is integrated/enabled properly.

5.1 Enable Core Control

Follow these steps to enable Core Control in the build. The first two steps are already included in the patch, and are mentioned only for information purposes. Only the last step (3) should be followed to enable core Control.

1. Add the following CONFIG flag in your build “perf_defconfig” and “defconfig” files.

   ```
   CONFIG_MSM_CORE_CTL_HELPER=y
   ```

2. Add following lines in “device-vendor.mk” file to insert the core_ctl kernel module.

   ```
   CORECTL := core_ctl.ko
   ```

   and

   ```
   PRODUCT_PACKAGES += $(CORECTL)
   ```

3. Add the following in the “init.qcom.post_boot.sh” file to set up core_ctl parameters for MSM8994 devices.

   ```
   insmod /system/lib/modules/core_ctl.ko
   echo 1 > /sys/devices/system/cpu/cpu4/core_ctl/min_cpus
   echo 60 > /sys/devices/system/cpu/cpu4/core_ctl/busy_up_thres
   echo 30 > /sys/devices/system/cpu/cpu4/core_ctl/busy_down_thres
   echo 100 > /sys/devices/system/cpu/cpu4/core_ctl/offline_delay_ms
   echo 1 > /sys/devices/system/cpu/cpu4/core_ctl/is_big_cluster
   echo 4 > /sys/devices/system/cpu/cpu4/core_ctl/task_thres
   ```

5.2 Verify Core Control

The following steps verify Core Control logic on the build.

1. Check the Number of CPUs when the system is idle.

   It should match min_cpus that are set for each cluster.

2. Run a CPU intensive benchmark tool.

   Check the number of online cores either by executing cat command for online node or monitor kernel log. Without Core Control, there is no hotplug except on the occasion of thermal and BCL. If Core Control is working, cores come and go regularly when running a benchmark that combines single-threaded and multi-threaded tests.