International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 06 | June 2020 www.irjet.net p-ISSN: 2395-0072

A Review on Power State Coordination in Application Processors using ARM Specified PSCI

Raghuvarya S[1], Deepika P[2]

[1]*Raghuvarya S, Dept. of Electronics and Communication Engineering, RV College of Engineering, Bengaluru, Karnataka, India*
[2]*Deepika P, Assistant Professor, Dept. of Electronics and Communication Engineering, RV College of Engineering, Bengaluru, Karnataka, India*

Abstract - *Technology is growing fast, and there is an increased need for improvement in portable smart devices that are more capable of handling multiple tasks in parallel. The term improvement calls for the most obvious performance factor but at the same time power efficiency takes its stand. Handling of multiple apps on a smartphone requires multiple cores in operation. However, all cores need not be turned on at the same time causing wastage of power. The cores that are not required for computation need to turn themselves into power saving mode. This calls for an intelligence that can balance the power states between various cores and clusters on a chip, which is called the power state coordination interface (PSCI). The power state coordination interface is a specification proprietary to ARM Limited which can be implemented in two modes, namely OS initiated mode and the platform coordinated mode. In this paper a review is made on power state coordination in application processors using PSCI as specified by ARM Limited. A logical analysis of the same is done in terms of the performance and power factor.*

***Key Words*: ARM, Benchmark, Cluster, Core, OS, OS initiated mode, Performance, Power, PSCI, Platform, Platform coordinated mode, SoC**

1. INTRODUCTION

Power is a factor that is quite vital in reality. New chip designs come up with performance increment as the main motto. But when this comes at the cost of power efficiency there is a hit on the whole basic purpose. Hence the chipsets must be made power efficient without a significant compromise on the performance factor.

Today's smartphones deal with high end computations, artificial intelligence and handling multiple applications at the same time. The requirements vary from device to device and user to user. These processes are handled by multiple processing units or cores on a chipset. The cores need to distribute the processes accordingly and perform the tasks efficiently. All cores need not stay active all the time. Each core has a set of supported power states, at which they need to run in order to save power. Cores that are not required to perform any task at a particular instant must power down or enter low power mode. In order to accomplish the requirement, an intelligence is required to balance the power states among various cores and clusters.

ARM Limited provides a specification called PSCI (Power State Coordination Interface) that deals with balancing of the power states between various cores and clusters. The device will be running on a high level operating system that will have an idea about the apps that run and their processing requirements. The OS will hence vote for core and cluster power states for each core and the corresponding cluster. PSCI will provide an interface to coordinate with the power states of various cores and clusters and implement it on the hardware.

The paper is structured as described: in Section 2, the application processor topology is briefed considering an assumed six core system as an example. Section 3 discusses the two modes in which the PSCI can be implemented as described by ARM Limited. In Section 4, the performance and power measurements are discussed. Section 5 and 6 discusses the logically reviewed results and concludes the review with an analysis made between the two modes of ARM defined PSCI.

2. APPLICATION PROCESSOR TOPOLOGY

A modern day smartphone is designed to process multiple tasks and handle multiple applications at a time. In order to ensure handling multiple applications on a device smoothly keeping lags as minimal as possible, multiple cores are used. However all cores used need not be of the same computation capability. The clock frequency at which the core runs is one of the major factors for core power consumption. Lower the frequency, lesser will be power consumed per instruction for a given operation. Internal to a core, this is done using DVFS (Dynamic Voltage and Frequency Scaling).

For a core to keep running, a minimum clock frequency must be maintained. This threshold clock frequency depends on the maximum clock frequency for which the core is designed. In order to keep the processor power efficient while not compromising with the performance, a combination of cores of different clock frequencies is used in an application processor.

In this review, a six core configuration is considered. Here out of the six cores 2 cores are considered to be the performance cores and four cores are considered to be power saving cores. Fig 1 describes the topology of the application processor considered.

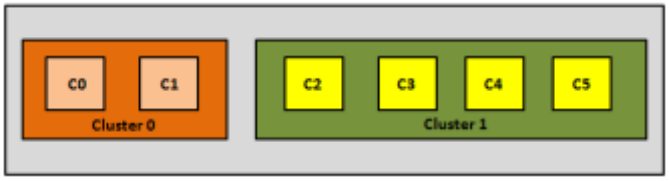

Fig -1: Application Processor Topology

The performance cores are the ones with high computation capability. They run at a relatively higher operating clock frequency and hence consume more power. Cores C0 and C1 are the performance cores. The power saving cores are those that run at a relatively lower frequency and consume less power. Cores C2, C3, C4 and C5 are the power saving cores.

Some applications such as camera require performance cores to process them. Some applications such as high graphics oriented games require multiple performance cores to run. Some simple applications such as call log or notes can run smoothly on the power saving cores itself.

A group of cores with similar specifications form a cluster. A set of hardware support components such as L2/L3 cache, logic and memory power rails will remain specific to a cluster depending on the requirement of each core pertaining to the cluster. Here, Cluster 0 will be the performance cluster and Cluster 1 will be the power saving cluster. All together they constitute a system

3. PSCI MODES

PSCI acts an interface between the supervisory software and the hardware implementation in managing the power states between the cores and clusters of a system. As per the guidelines of ARM Limited, PSCI can be implemented in two modes as described in the following sub sections: [2]

3.1 OS Initiated Mode

In OS initiated mode, OSPM only demands an idle power state for the implemented hierarchy node when the last core of a cluster powers down. The core selects an idle state for itself and when the core going down is the last core it acts as the master for putting down the higher levels into low power mode. The last core to go down checks the power states of other cores and decides the cluster power state. The platform takes the decision by only considering the most recent request made for a specific node for deciding on it's low power mode.

OS initiated mode takes the following advantages:

- Avoids OSPM overhead for evaluating selection for low power modes at higher levels, which won't be used as other cores in the cluster are awake.
- The higher level low power mode selections are taken by considering the most recent request for a particular node.

However, the platform will still hold the responsibility to ensure the functional correctness. In case of any incompatibilities of the requested modes, the request made will be rejected. To ensure that unnecessary rejections of low power mode requests don't happen, the ordering of requests of different cores is extremely critical. The race condition that may occur between two cores requesting for low power mode turn out to be a key issue. The platform performs all the dependency checks before executing the requested power state. The OS level view of the power state of a particular core might sometime differ from the platform level view.

The last core to go down in a cluster acting as the master may create an issue here. If the last core going down requests a low power state for its higher level which is deeper than the requirements of some other core in the same cluster, the request made is rejected.

3.2 Platform Coordinated Mode

In platform coordinated mode, the power state coordination is done at the platform level. So, the power states considered for coordination will be the platform level view. The platform computes the consolidated power state. Every time a core goes down, it will send a low power mode ID that specifies to which state of low power mode the core wants to enter and what states of low power mode it wants it higher levels to be at. Based on that the core will enter to the corresponding low power mode. For all the higher nodes it will update the desired low power mode as its vote for that node. When the last core of a cluster goes down, the core collapses and an aggregation of votes is done for the higher levels. The state to which the cluster collapses depends on the aggregated value. In particular the votes made take care of the following:

- Higher levels cannot enter a deeper low power state than what is requested by its children.
- Higher levels cannot enter a local power state that has higher wakeup latency as compared to the requested one.
- Higher level low power state does not violate any dependencies of powered down resources for its children.

At platform, the votes made by all the cores under a particular node is checked at every level. The platform tries to implement the deepest possible state without violating the requirements as per each vote. Hence, the aggregation results in the deepest possible state that does not violate the requirement of any of the nodes at the lower affinity levels. Logically, this means that the aggregated low power state for a node is the shallowest state amongst all the votes made for that particular node. It will be having the least sleep and wake up latencies among all the votes made. Although this fact that a shallower state having lesser entry and exit latencies always need not be true, practically it turns out to be valid for most of the use cases.

As a thumb rule for the sake of efficiency, the platform should not enter into a state with higher minimum residency than what is requested. However, there are instances where the rule can be broken if higher efficiency can be achieved.

The platform supports auto-promotion mechanism. Here the higher level votes are made implicit instead of having them to be explicit. The platform has availed OSPM to request certain power states to the lower affinity levels. These votes imply corresponding states as the votes for all higher levels. This can be observed during booting or hot plugging the CPU. Logically it can be said that the platform has a better synced view of the power states compared to the OS.

4. PERFORMANCE AND POWER MEASUREMENT

A device can be made to undergo performance and power validation to analyse between the two modes of PSCI.

Performance measurement is done using various industry standard benchmark use cases. Each use case delivers a benchmark score upon running them successfully. By using the benchmark scores drawn for both the modes of PSCI, an analysis can be made in terms of performance. The following are the performance benchmarks that can be used to validate performance:

i. Antutu, a Chinese benchmarking tool owned by Chinese company Cheetah Mobile is a commonly used to benchmark performance for smartphones and various devices. In this benchmark, higher scores indicate better performance.

ii. Geekbench, an industry standard cross platform processor benchmark is used to validate the performance of multi core processors by delivering scores for single core performance and multi core combinational performance for the processor. Since PSCI involves balancing power states between various cores and clusters, this turns to be an important tool that can analyse the performance factor between the two modes of PSCI.

iii. Androbench, an industry standard benchmarking tool is commonly used to validate the storage performance of an android device delivers information about SQLite benchmark and micro-benchmark.

iv. Jetstream, a commonly used benchmarking tool to analyse browsing performance gives insights about the browsing speed in terms of the scores. Higher scores indicate better performance.

v. PC Mark, a cross platform software benchmarking tool used to get insights on performance delivers scores corresponding to performance at a component level. Again higher score indicate a better performance.

Apart from the above, there are various GFX benchmarks used to analyse performance in terms of graphics. The performance is also analysed by timing the launch latencies of various commonly used games and apps.

Power can be analysed using a fundamental measuring process with RCM. The process of measuring power with RCM is based on differential current principle. This requires that all phases be guided through a residual current transformer at the measuring point (outlet to be protected), with the exception of the protective earth.

5. RESULTS

A review was made on power state coordination in application processors using PSCI as specified by ARM Limited. Upon analysis it can be logically said that PSCI implemented using platform coordinated mode may implement more accurate power states for the cores and clusters in the system as aggregation happens at the platform level which is closer to the actual hardware implementation. Hence, an improvement in the power factor can be expected when PSCI is implemented using platform coordinated mode in certain use cases. This is because the platform may have a better view of the core power states than what the OS will have. However, with certain performance use cases a regression can be expected in PSCI implemented using platform coordinated mode because of the extra aggregation overhead at the implementation level. These results may not be universally true and can vary between use cases.

6. CONCLUSION

PSCI is an ARM Limited specification which is an interface for coordinating power states between multiple cores and clusters in an SoC. PSCI can be implemented in two modes, namely OS initiated mode and platform coordinated mode.

There can be a race observed between the OS view and the platform view of the power states. However, power and performance factors must be considered before choosing the right mode to implement PSCI for a system. The performance factor depends on the high level operating system that will be used on the SoC. With some operating systems, the OS initiated mode and platform coordinate mode may not differ much in terms of the performance yield but may significantly save power. In such cases PSCI can be preferred to be implemented using platform coordinated mode if an overall better power factor is delivered. However, with some operating systems, OS initiated mode will yield significantly better performance than platform coordinated mode. In such cases it is wise to implement PSCI in OS initiated mode.

REFERENCES

[1] Limited, "Power State Coordination Interface - Platform Design Document" , ARM DEN 0022D, PSCI 1.1 release, 21 April 2017
Available online (as on 30 May, 2020) : http://infocenter.arm.com/help/topic/com.arm.doc.den0022d/Power_State_Coordination_Interface_PDD_v1_1_DEN0022D.pdf

[2] The Unified EFI Forum, "Advanced Configuration and Power Interface Specification" , Version 6.1, January 2016
Available online (as on 30 May, 2020) : https://uefi.org/sites/default/files/resources/ACPI_6_1.pdf

[3] Lin I, Jeff B, & Rickard I, "ARM platform for performance and power efficiency - Hardware and software perspectives", 2016 International Symposium on VLSI Design, Automation and Test (VLSI-DAT).

[4] ARM Limited, "big.LITTLE Technology: The Future of Mobile", ARM white paper, 2014

[5] H. Wei, Z. Huang, Q. Yu, M. Liu, Y. Guan, and J. Tan, "RGMP-ROS: A Real-time ROS Architecture of Hybrid RTOS and GPOS on Multi-core Processor", Proceedings of the 2014 IEEE International Conference on Robotics and Automation, 2014, pp. 2482–2487.

[6] A. Madhavapeddy, R. Mortier, C. Rotsos, D. Scott, B. Singh, T. Gazagnaire, S. Smith, S.Hand, and J. Crowcroft, "Unikernels: Library Operating Systems for the Cloud", Proceedings of the 18th International Conference on Architectural Support for Programming Languages and Operating Systems, 2013, pp. 461–472.

[7] S. Boyd-Wickizer, M. F. Kaashoek, R. Morris, and N. Zeldovich, "Nonscalable locks are dangerous", 2012 Proceedings of the Linux Symposium

[8] A. Bastoni, B. B. Brandenburg, and J. H. Anderson, "Cache-Related Preemption and Migration Delays: Empirical Approximation and Impact on Schedulability", Proceedings of the 6th International Workshop on Operating Systems Platforms for Embedded Real-Time

[9] A. Baumann, P. Barham, P.-E. Dagand, T. Harris, R. Isaacs, S. Peter, T. Roscoe, A. Schupbach, and A. Singhania, "The Multikernel: A new OS architecture for scalable multicore systems", Proceedings of the 22nd ACM Symposium on Operating Systems Principles, 2009, pp. 29–44.

[10] S. Kato and N. Yamasaki, "Scheduling Aperiodic Tasks using Total Bandwidth Server on Multiprocessors", Proceedings of the 5th International Conference on Embedded and Ubiquitous Computing, 2008, pp. 82–89.