WHITE PAPER

big.LITTLE Technology Moves Towards Fully Heterogeneous Global Task Scheduling

Improving Energy Efficiency and Performance in Mobile Devices

Brian Jeff

November, 2013

Abstract

ARM® big.LITTLE™ processing is a power-optimization technology from ARM, where high-performance CPUs and efficiency-tuned CPUs are combined in a cache coherent compute cluster, and software dynamically transitions to the appropriate CPU. The software control mechanisms have been evolving toward a fully heterogeneous scheduling model, Global Task Scheduling, which is now ready for deployment. This software model runs on the same big.LITTLE technology, but it brings new capabilities including the ability to run on all cores simultaneously, finer grained control of thread placement and the opportunity to increase performance in addition to saving energy. This paper will discuss the fundamental operation of big.LITTLE technology and the current results, in performance and power, on mobile use cases as measured on silicon platforms from ARM partners.

Introduction

The performance demanded from users of current smartphones and tablets is increasing at a much faster rate than the capacity of batteries or power savings from advances in semiconductor process. At the same time, users are demanding longer battery life within roughly the same form factor. This conflicting set of demands requires innovations in mobile system-on-chip (SoC) design beyond what process technology and traditional power management techniques can deliver.

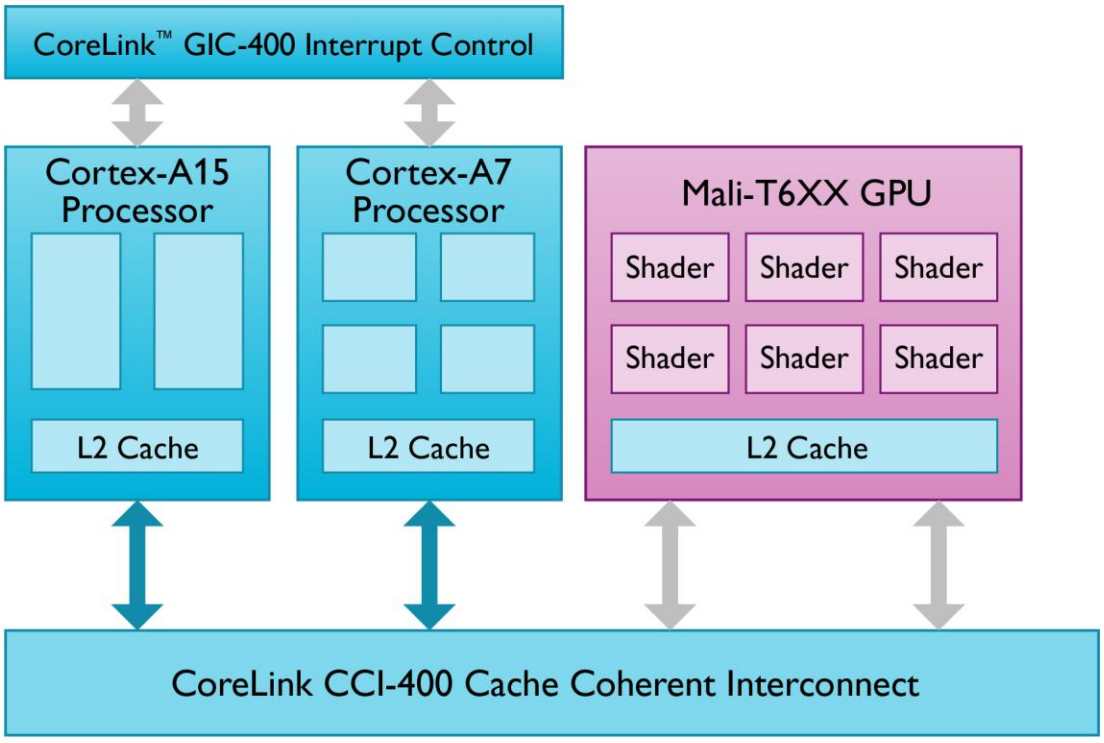
The usage pattern for smartphones and tablets is quite dynamic. Periods of high-processing intensity, such as those seen in mobile gaming and web browsing, alternate with typically longer periods of low-processing intensity tasks such as texting, e-mail and audio, and quiescent periods during complex apps. ARM big.LITTLE processing takes advantage of this variation in required performance by combining two very different processors together in a single SoC. The big processor is designed for maximum performance within the mobile power budget. The smaller processor is designed for maximum efficiency and is capable of addressing all but the most intense periods of work.

In the first big.LITTLE technology-based SoCs, the 'big' processors are ARM Cortex®-A15 cores, and the 'LITTLE' processors are Cortex-A7 cores. Together, they create a system that can run both high-intensity and low-intensity software threads in the most energy-efficient manner. By connecting the Cortex-A15 and Cortex-A7 processors via the ARM CoreLink™ CCI-400 cache coherent interconnect, the system is flexible enough to support a variety of big.LITTLE usage models, including the Global Task Scheduling model in which the operating system (OS) kernel scheduler directly manages thread allocation, enabling a wide range of tuning and optimization possibilities.

Since the introduction of big.LITTLE processing in 2011, ARM has been working together with partners to optimize and tune the software. The fruits of this collaboration are now becoming evident as the first big.LITTLE technology-based devices and platforms are in the market, and a new wave of devices is set to arrive, each introducing new levels of optimization. This paper will discuss the measurements ARM has taken from one of the more recent big.LITTLE implementation platforms from our silicon partners.

big.LITTLE Processing Technology

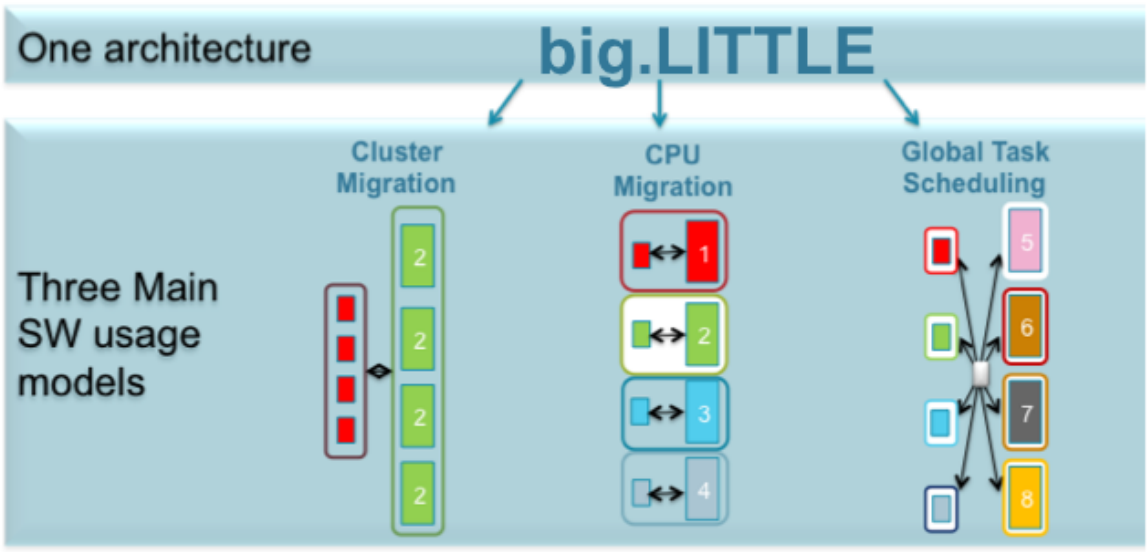
In big.LITTLE technology, a fundamental point is that the 'big' and 'LITTLE' processors are architecturally identical – they look the same from the perspective of software. Forming the first big.LITTLE processor configuration, the Cortex-A15 and Cortex-A7 cores implement the full ARMv7-A architecture. The system block diagram below shows a big.LITTLE implementation based on Cortex-A15 and Cortex-A7 processors. Each processor implements an ARM AMBA® 4 ACE coherent interface connected to the CoreLink CCI-400 cache coherent interconnect. The GIC-400 generic interrupt controller serves as the global interrupt distributor to the CPU cores in the system. A GPU such as the ARM Mali™-T600 series can be connected through the cache coherent interconnect as well, allowing IO coherence with the CPU cluster for improved GPU compute performance. The system shown is a generic big.LITTLE CPU+GPU subsystem; silicon vendors can tailor their SoC in many ways, including the implementation of the various blocks, the GPU and system components, as well as the number and type of big and LITTLE cores. With the launch of the Cortex-A57 and Cortex-A53 processors in 2012, silicon vendors can develop high-end 64-bit capable big.LITTLE processing systems. With future versions of the Cortex-A12 processor, SoC designers can develop big.LITTLE CPU subsystems targeting mid-range performance and power budgets.



Software for big.LITTLE Implementations

In software for big.LITTLE implementations, the fundamental idea is that executing code should be dynamically moved to the right-size processor for the performance needs of the task. There are different levels of granularity for addressing this partitioning of work to the big and LITTLE core clusters. The first two modes are migration modes: they use modifications to existing dynamic voltage and frequency scaling mechanisms to decide when to switch an entire CPU cluster or an individual CPU worth of work to the big or LITTLE side of the compute subsystem. The third mode, Global Task Scheduling, modifies the kernel schedule to be aware of the performance requirements of individual threads, and allocates threads to an appropriately sized core. Global Task Scheduling provides the most flexibility for tuning the performance and power balance in an SoC, but it is also the most complex method to implement. That complexity, however, is addressed in kernel patchsets and a stable Linux Kernel Tree version that incorporates the patches. Therefore an SoC developer's or OEM's work in system bringup is no more complex than for a standard DVFS-based system. It does potentially require more tuning of parameters related to the software for big.LITTLE implementations.

All three modes run in kernel space and require no modifications to user space code or middleware. Existing power management mechanisms for shutting down unused cores are active in all three cases. Additionally, all three modes can run on the same hardware, although only Global Task Scheduling can support different numbers of big and LITTLE cores.

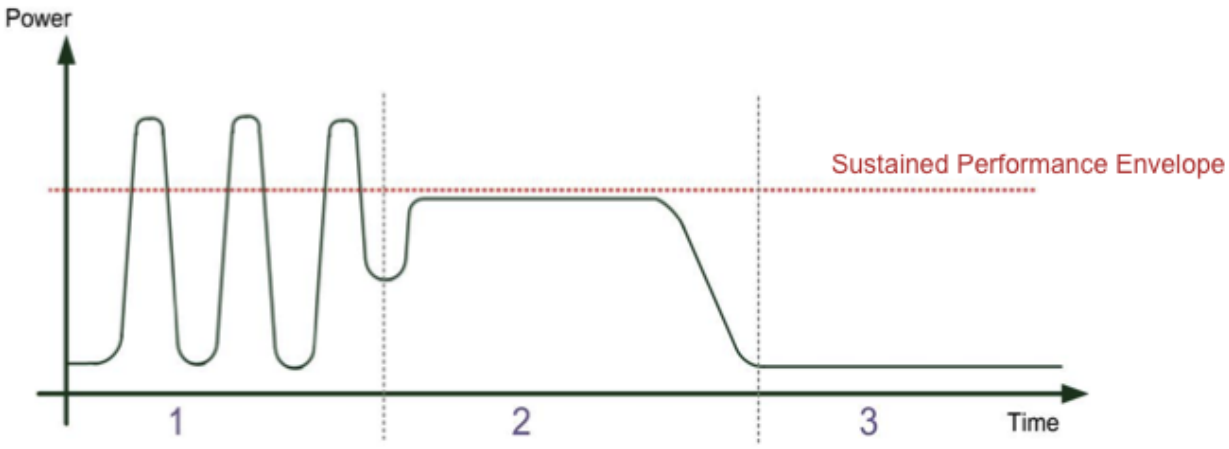


Other papers discuss the mechanics of Global Task Scheduling and CPU migration. The focus of this paper is on measurements from a partner development platform and production platforms using the Global Task Scheduling software model.

big.LITTLE Processing in Operation

There are three categories of run-time behavior that are particularly interesting for exploring the benefits and optimization potential of big.LITTLE processing:

1. High-intensity (bursty) workloads
2. Sustained workloads (with power and thermal limits)
3. Long-use, low-intensity workloads



In the high-intensity cases, software for big.LITTLE implementations is particularly well suited for responding to the bursts of peak performance as well as the troughs of lower required performance. Performance peaks can be allocated to Cortex-A15 cores or “big” CPUs, and the troughs can be addressed with Cortex-A7 cores or “LITTLE” CPUs. Because of the hardware cache coherency and the architecture of the Global Task Scheduling software, work can be migrated very quickly to big processors, and high-performance threads are identified by their load history and automatically started on big processors when they run. In these cases, the effectiveness of the software is determined by ability to react quickly to peak-performance requirements (so as not to slow things down relative to a big core only system) and in the ability to effectively use LITTLE cores with big cores shut down during the troughs to save power.

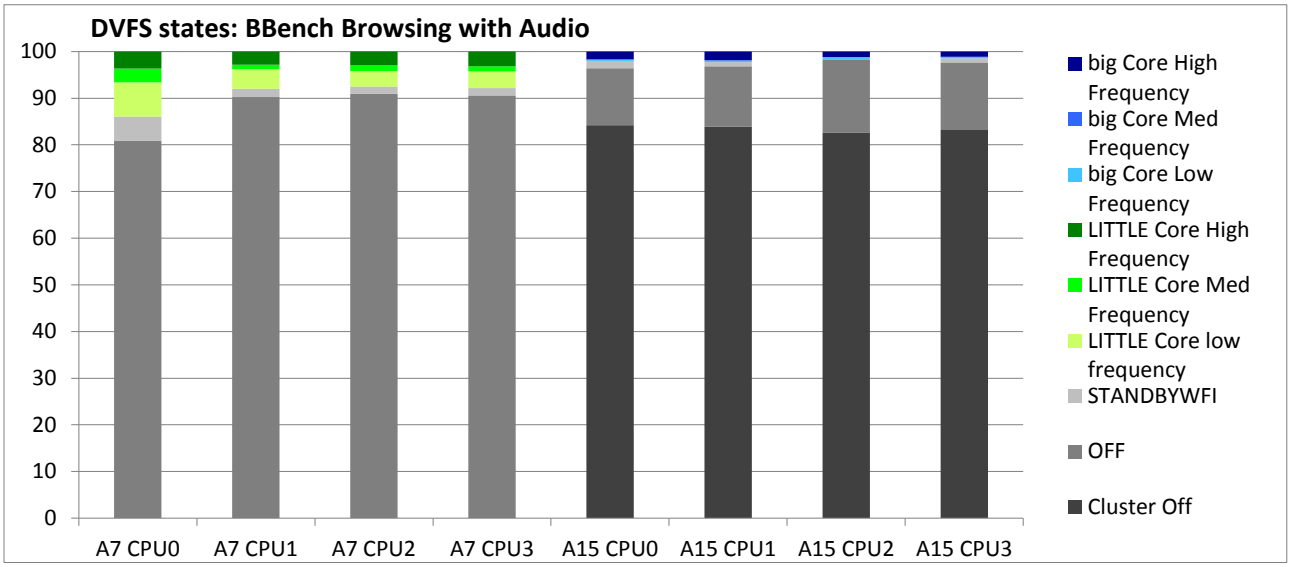
In the sustained performance cases, the benefits of big.LITTLE processing are just becoming evident as real systems reach the market and are tuned to real-world workloads. One example here is mobile gaming: in mobile games, the graphics processor is operating at near-peak capacity for much of the time, potentially consuming 80% of the SoC power budget. In a constrained thermal envelope, with a reduced power budget for the CPU subsystem, big.LITTLE processing enables an important reduction in power for these use cases that enables the GPU to run faster and deliver a better mobile gaming experience under the same SoC power budget. It is also possible for big.LITTLE technology to enable a more optimal mix of compute resource for a significantly constrained power and thermal budget. Libraries are in development to exploit the optimal balance of thermal capacity between CPU and GPU. This paper will highlight two use cases that are already exhibiting improved performance through big.LITTLE technology with power savings at the CPU subsystem level.

In the low-intensity cases, big.LITTLE technology has the most obvious benefits, as these workloads can run entirely on LITTLE processors at lower operating voltages. Here, the effectiveness of the software is determined by ability to remain running on the LITTLE processors without waking up the big cores.

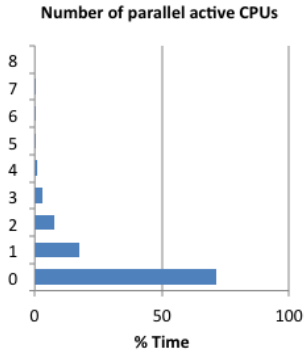
High Intensity (Bursty) Workloads

The graph below shows the activity of the big and LITTLE CPUs in a development board that ARM has been testing using global task scheduling software for big.LITTLE implementations. It is an Android™ based system capable of running mobile applications; the use case shown below is web browsing with audio running in the background, a known bursty load.

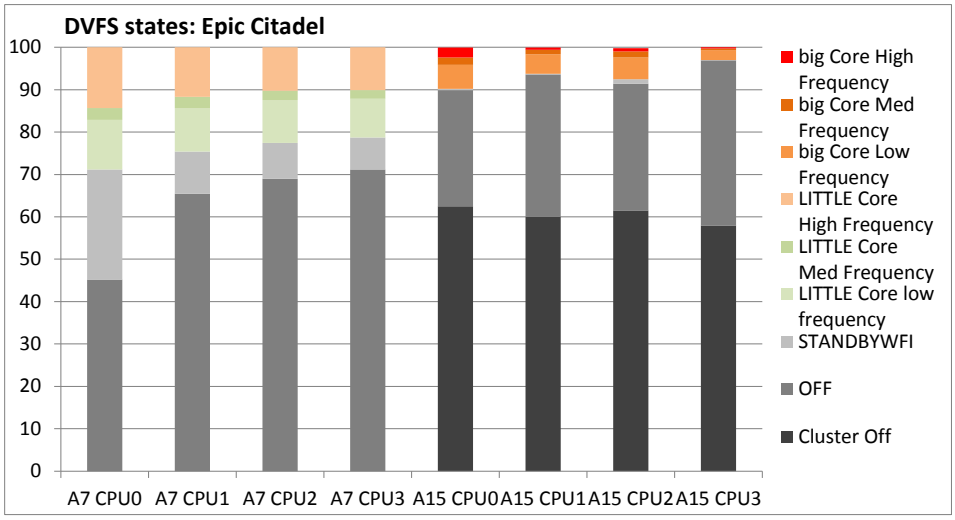
The graph shows the amount of time spent, as a percentage during the runtime of the test, in different DVFS and sleep states for each CPU in the big and LITTLE clusters. The darker the color, the higher the frequency, and the lighter the color, the lower the frequency and associated power. The LITTLE and big CPUs are represented as green and blue respectively. The gray colors represent wait for interrupt (WFI) idle, CPU power gated states, and full cluster power gated states. The darker the color the lower the leakage power – dynamic power is effectively zero for all three sleep states.



In the web browsing plus audio workload, the Cortex-A15 cores are cluster gated or power gated for a total of almost 95% of the runtime. When the big cores are running, they run at the maximum frequency.



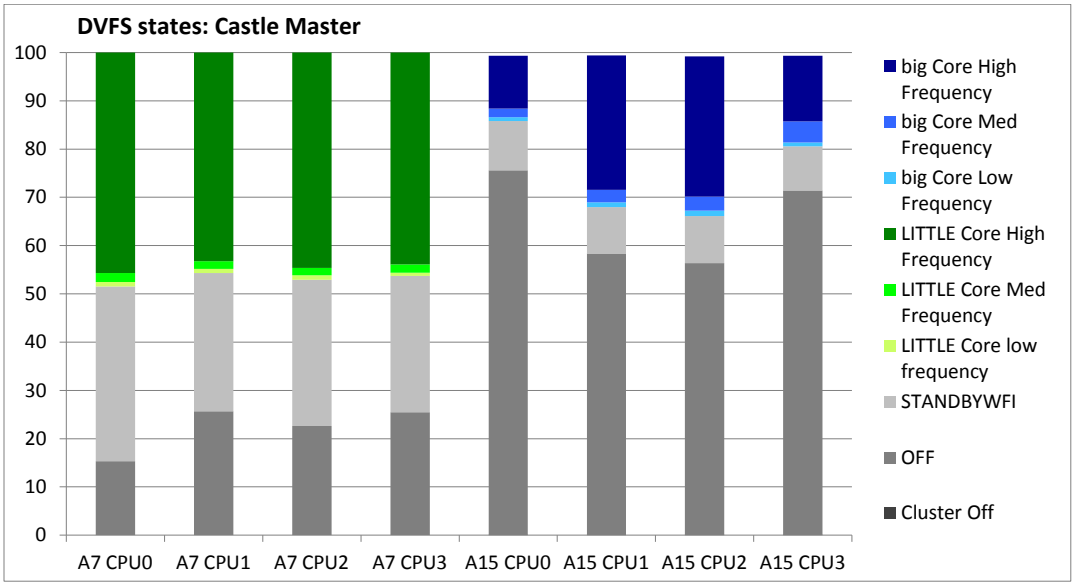
The Cortex-A7 CPU cluster runs at a mix of frequencies and is on between 10 and 20% of the time. The benchmark is configured to wait two seconds between subsequent page loads and to run audio decoding in the background, so there are large portions of the time when the CPUs are shut off. Not shown in the DVFS state diagram is how often multiple cores are running at the same time; in this workload, a single core or two cores are typically running. The percentage of time when more than four cores are running simultaneously is extremely small. This is consistent with the single-thread dominance typical of current web browsers.



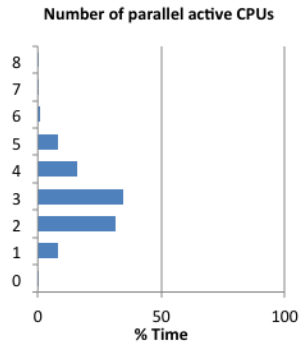
In the Epic Citadel game, the activity is similarly bursty, with the Cortex-A15 cores being used sparingly, and when they are running, they run at the moderately high frequencies or the maximum frequency. This example showcases the fact that during most of the runtime, the Cortex-A7 processor cluster is sufficient to handle the performance requirements of the game, and the software automatically adapts to the performance requirements. Analysis is still being done on the power savings and performance; when compared to the frame rate of the game in a Cortex-A15 core only mode of operation, the Global Task Scheduling software delivers the same performance as measured by the frame rate.

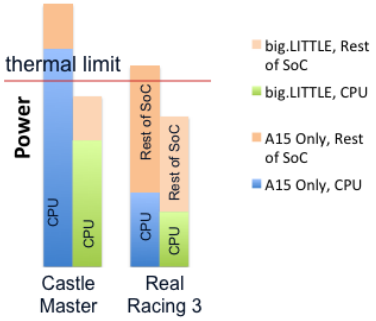
Sustained Workloads

The DVFS graph below shows the measured CPU activity when running a high-performance CPU intensive game, Castle Master. In this game, the Cortex-A7 cores run at peak frequency for almost 50% of the game time, and the Cortex-A15 cores run at peak frequency for significant portions of time as well.



In the sustained performance case, the Cortex-A7 cores are run at the maximum frequency, and the Cortex-A15 cores are run as needed. The allocation of work to big and LITTLE cores enables a significant reduction in power, as shown in the next graph. The Castle Master game and a similarly high-performance Real Racing game, both experience significant power savings relative to the base case where a Cortex-A15 processor runs by itself under DVFS control. In the case of the Real Racing game, the savings are enough to lower the system power below the thermal limit of power consumption. In the Castle Master game, the power savings can be used to increase the performance of the graphics processor or save SoC-level power and prolong battery life.



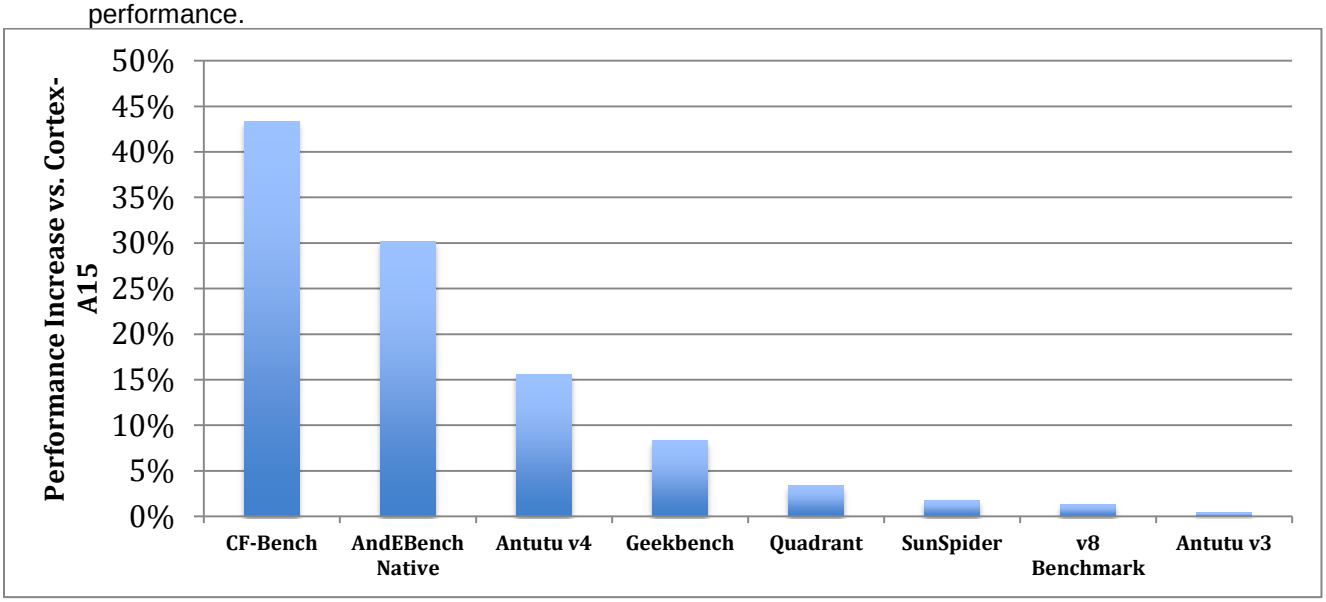


In the Castle Master game, the CPU load is much more threaded and is thus able to take advantage of parallel operation to achieve performance. This maps well to LITTLE cores and allows the big cores to be reserved for accelerating bursts of required performance. As software evolves to take more advantage of parallel threads, the benefit of big.LITTLE technology will increase. Even in the most-threaded gaming case in 2013, two big cores and four LITTLE cores would be sufficient. The amount of time the workload spends exercising all eight cores at the same time is extremely small.

Increased Performance on Threaded Workloads

When running benchmark software, the goal is to run the CPU or the GPU as fast as possible to measure a score. The effectiveness of software for big.LITTLE implementations in benchmark scenarios is measured against the case of a system with only big cores – if the software is doing its job properly, there should be no degradation in performance when running single-threaded performance benchmark tests. This is evident in the SunSpider, V8, and AndEBench cases. For the other workloads shown, additional

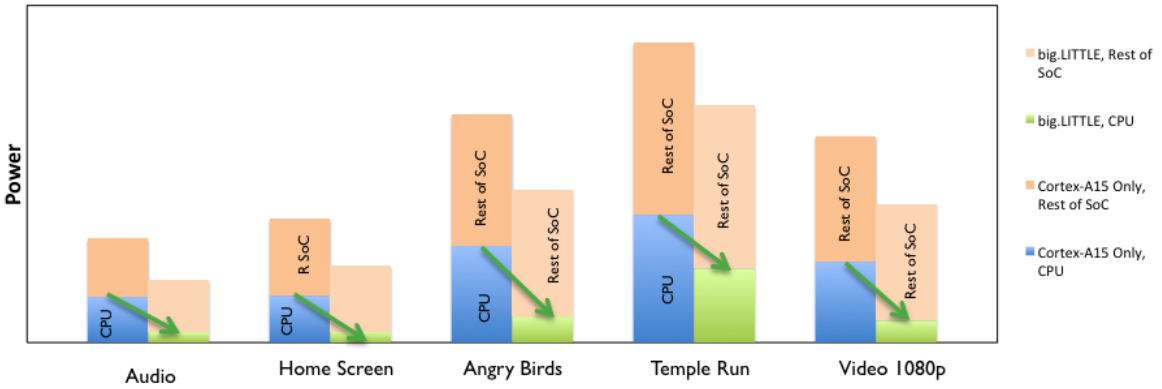
threads can make use of LITTLE cores at the same time as the big cores, enabling an increase in overall performance.



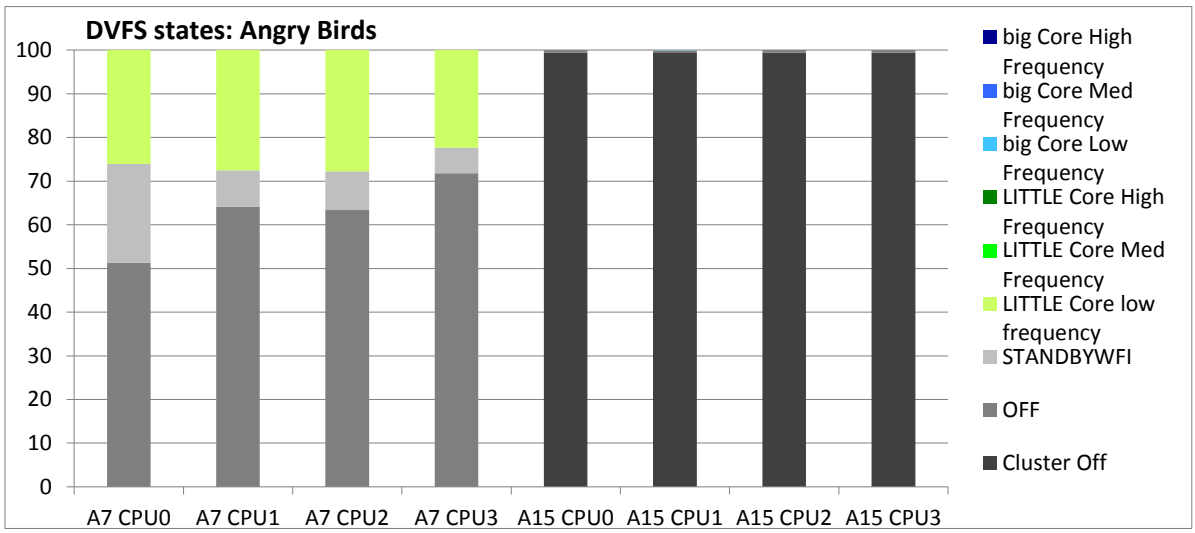
The performance increase on synthetic workloads is not necessarily indicative of the performance in real applications running on mobile devices. As real application software evolves to use more threads, the potential benefits of running more than four cores, as is possible in the Global Task Scheduling model, should become evident on real-application code.

Low-Intensity Use Cases

On workloads where the CPU intensity is low, the effectiveness of big.LITTLE processing is evident in the power savings at the CPU and SoC levels. The cases below can all predominantly or entirely run on the LITTLE cores, and therefore exhibit power savings in the range of 50 to 80% at the CPU level, and 30 to 50% at the SoC level.



The DVFS state diagram for the Angry Birds game is representative of the type of workload that can fit almost entirely on the Cortex-A7 cores; the GPU may be quite active, but for the CPU, it is a long-running, low-intensity workload. The Cortex-A7 cores spend their time in low and middle frequencies, and the Cortex-A15 cores are used quite sparingly, but they do wake up during initial screen render operation.

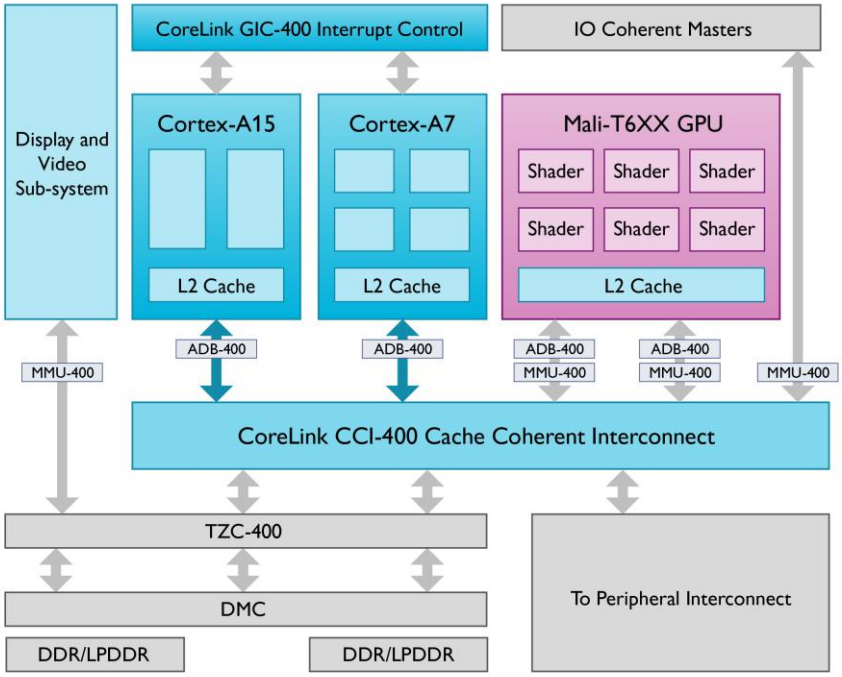


It should be stated that the frame-rate performance in the low-intensity game cases shown here is the same as measured during the Cortex-A15 core only case. The use of Cortex-A7 cores here doesn't impact game performance, but it does reduce the power quite substantially.

Integrating an Efficient System

Building a big.LITTLE processing system from a hardware standpoint includes more than the two CPU clusters and the coherent interconnect. As described in the initial system diagram, the global interrupt controller distributes interrupts to the various cores. The CPU clusters are connected to the cache coherent interconnect through an asynchronous bridge to enable each CPU cluster to be scaled independently in frequency and voltage. The independent DVFS operation of the CPU clusters allows the system to more closely track the performance requirements in loads where there is a mix of high-performance and background activity.

The display and video subsystem is connected directly through the memory controller, and the CCI block is connected through to the DMC via a TrustZone® security architecture controller that allows the secure state information from the CPU address lines to carry on through the system and enable hardware security. Finally, the CoreLink MMU-400 provides system memory management functionality to the design, enabling the GPU and other blocks to use the same virtual address map as the CPU.



The big.LITTLE processing system shown includes two big cores and four LITTLE cores – this asymmetric topology is supported by the Global Task Scheduling software natively. The OS is aware of the big and LITTLE cores and can allocate work to the number of high-performance and high-efficiency cores in the system.

Scalable Technologies for SoCs

This paper has discussed the operation and measurements from partner silicon platforms employing big.LITTLE technology and the Global Task Scheduling software model. The scalable performance afforded by the close coupling of architecturally identical big and LITTLE cores is enabling power savings, added performance, and increased optimization in thermally constrained use cases.

Another dimension of scalability is possible with the combination of an ARM Cortex-A series CPU subsystem and a Mali GPU. The Mali-T600 series GPUs are capable of snooping the CPU caches through the CCI-400 interconnect, providing IO coherence between the GPU and CPU and enabling GPU compute with more efficient drivers that require fewer cache flushes as is typical in software coherent systems. The combination of big, LITTLE, and GPU enables scalable performance for Renderscript compute and OpenCL applications that take advantage of the graphics processor to process and analyze video and data, in addition to rendering data for viewing. Applications

like face detection, augmented reality, and video analysis can take advantage of this scalable combination of processors, and no doubt new applications will emerge that take further advantage of this scalable capacity.

Conclusion

In the latest round of performance benchmarking and power analysis, big.LITTLE technology is shown to deliver on key benefits. The tested platforms, running Global Task Scheduling software, bring significantly higher peak performance within a tighter power budget, and in many use cases, showed improved SoC performance under thermal constraints.

The software and hardware for big.LITTLE technology saved power at the CPU and SoC level across a range of workloads and use scenarios, and demonstrated increased performance and efficiency increase on threaded workloads.

From a market deployment standpoint, big.LITTLE technology is shipping in products today, and a wider range of differentiated solutions from silicon partners is coming. These devices are transitioning to more advanced Global task scheduling software for big.LITTLE implementations, and are being tuned for a more optimal balance of performance and power in mobile devices.

References

Advances in big.LITTLE Technology for Power and Energy Savings - Improving Energy Efficiency in High-Performance Mobile Platforms – Brian Jeff, 2012.

big.LITTLE Processing with ARM Cortex-A15 & Cortex-A7 - Improving Energy Efficiency in High-Performance Mobile Platforms – Peter Greenhalgh, ARM, 2011.

System software for ARM big.LITTLE systems – A software technique to maximize energy efficiency and performance on ARM Big.LITTLE systems – Robin Randhawa, ARM, 2011.