WHITE PAPER

Software Techniques for ARM big.LITTLE Systems

Robin Randhawa, Principal Engineer April 2013

Introduction

Mobile usage has changed significantly and today's consumers are increasingly using their smartphone for the majority of their connected lives. This includes high-performance tasks, such as web browsing, navigation and gaming, and less demanding 'always on, always connected' background tasks, such as voice calls, social networking and email services. As a result, the mobile phone has become an indispensible computing device for many consumers. At the same time, new mobile form factors, such as tablets, are redefining computing platforms in response to consumer demand. This is creating new ways for consumers to interact with content and brings what was once only possible on a tethered device to the mobile world. Truly smart next generation computing.

So where is Moore's Law going to take us with all this? History predicts a doubling every eighteen months until we get from thousands to billions of transistors, but if you actually look at the performance of a single processor it has all but stalled because the amount of power you can consume in the system has peaked.

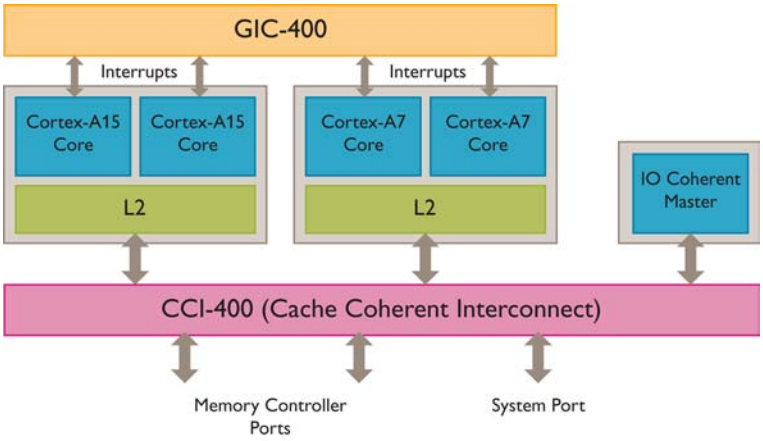
For any single processor in the future, heat dissipation will limit any significant increase in speed. Once you hit the thermal barrier of a device, it will simply melt, or in the case of a mobile phone, start to heat the device to the point where it is uncomfortable to the user. Apart from the physical aspects of heat dissipation it is also hugely power inefficient. If you take a processor and tweak the implementation so it goes faster and faster, the amount of power it takes becomes exponential, and the last little bit is really expensive. Whereas in the past double the size meant double the speed, now double the size equates to just a few percent faster, so the gains aren't there from complexity, that's one of the reasons why we have hit a limit for single core systems.

So if you can't make a single core go faster, the number of individual cores has to increase. This also brings the benefit of being able to match each core to the demands being placed on it, which is where the concept of ARM big.LITTLE™ processing helps.

Big.LITTLE processing addresses one of today's most significant challenges: extending consumers' always on, always connected mobile experience with both improved performance AND longer battery life. It achieves this by grouping a ‘big’ multicore processor with a ‘LITTLE’ multicore processor and seamlessly selecting the right processor for the right task, based on performance requirements. Importantly, this dynamic selection is transparent to the application software or middleware running on the processors.

The current generation of big.LITTLE designs now shipping in devices bring together a high performance Cortex™-A15 multi-processor cluster with an energy efficient Cortex-A7 multi-processor cluster. These processors are 100% architecturally compatible and have the same functionality (support for LPAE, Virtualization extensions and functional units such as NEON™ and VFP), allowing software applications compiled for one processor type to run on the other without modification.

Structure of a big.LITTLE system



Both multi-processor clusters are fully cache coherent both within their clusters and between the clusters. This inter-cluster coherency is enabled by ARM's CoreLink™ Cache Coherent Interconnect (CCI-400) which also enables I/O coherency with components such as a GPU system like ARM Mali™-T604. CPUs in both clusters can signal each other via a shared interrupt controller such as the CoreLink GIC-400.

Execution modes in a big.LITTLE system

Since the same application can be made to run on a Cortex-A7 or a Cortex-A15 without modification, this opens up the possibility to map the tasks of an application to the right processor on an opportunistic basis. This fact is the basis for a number of execution models, namely:

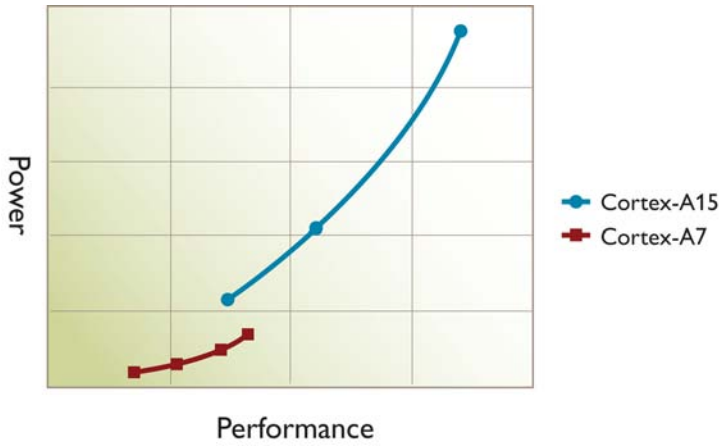
1. big.LITTLE migration model
2. big.LITTLE MP model

As the name suggests, the migration model enables the capture and restoration of software context from one processor type to another. In CPU migration, each CPU in a cluster is paired with its counterpart on the other cluster and the software context is migrated opportunistically between clusters on a per-CPU basis. If no CPU within a cluster is active, then the entire cluster and the associated L2 cache can be powered off.

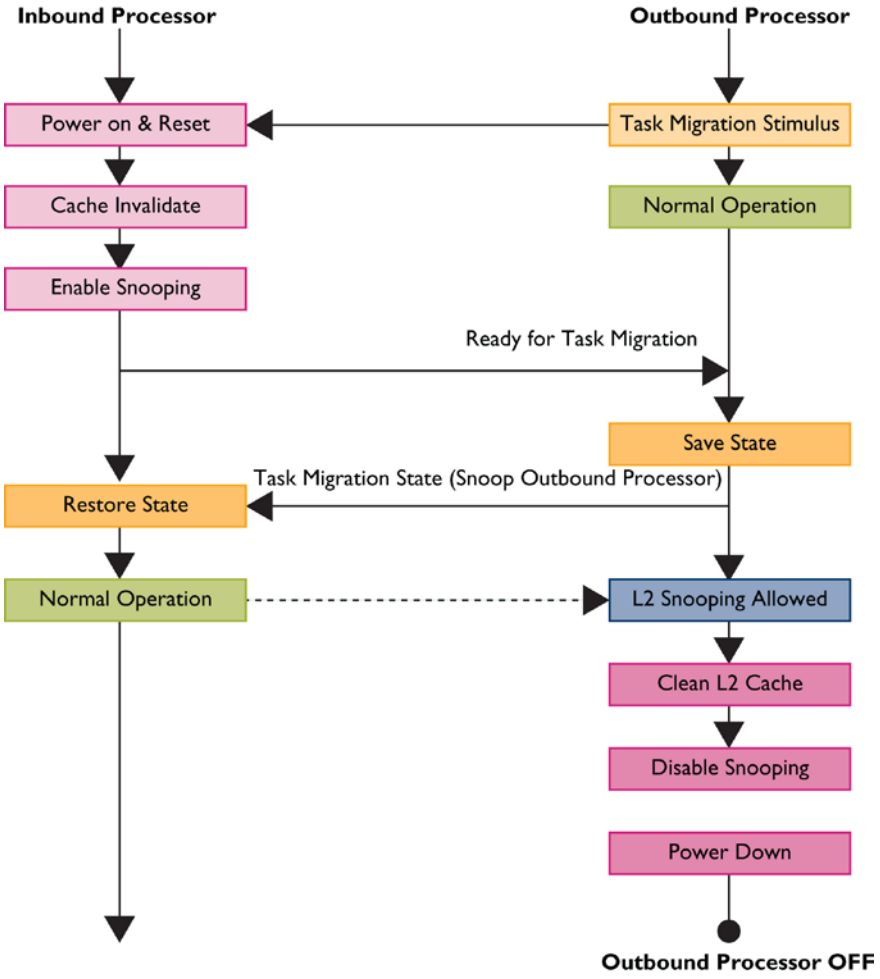
The MP model enables the software stack to be distributed across processors in both clusters. All CPUs may be in operation simultaneously offering peak system performance.

big.LITTLE migration model

The migration model is a natural extension to power-performance management techniques such as DVFS (Dynamic Voltage and Frequency Scaling). A migration action is akin to a DVFS operating point transition. Operating points on a processor's DVFS curve will be traversed in response to load variations. When the current processor (or cluster) has attained the highest operating point, if the software stack requires more performance, a processor (or cluster) migration action is affected. Execution then continues on the other processor (or cluster) with the operating points on this processor (or cluster) being traversed. When performance isn't needed, execution can revert back.



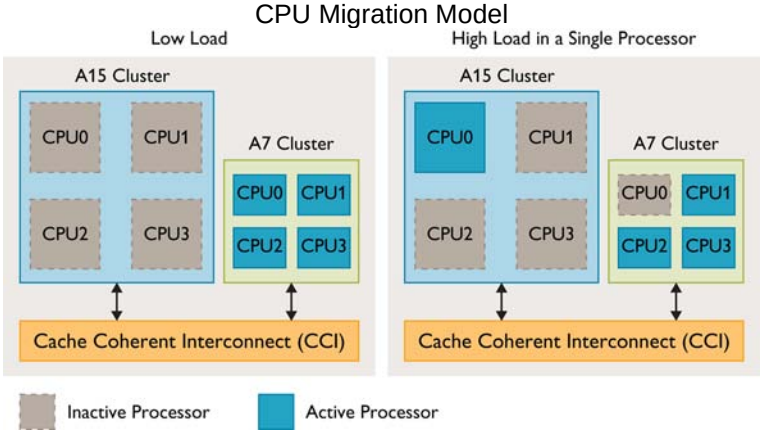
Coherency is clearly a critical enabler in achieving a fast migration time required as it allows the state that has been saved on the outbound processor to be snooped and restored on the inbound processor rather than going via main memory. Additionally, because the L2 cache of the outbound processor is coherent it can remain powered up after a task migration to improve the cache warming time of the inbound processor through snooping of data values. However, since the L2 cache of the outbound processor cannot be allocated too, it will eventually need to be cleaned and powered off to save leakage power. The sequence is depicted below.



big.LITTLE CPU migration model

In CPU migration, each processor on the LITTLE cluster is paired with a processor on the big cluster. CPUs are divided in CPU pairs (CPU0 on the Cortex-A15 and Cortex-A7 processors, CPU1 on the Cortex-A15 and Cortex-A7 processors and so on). When using CPU migration only one CPU per processor pair can be used at the same time.

CPU Migration Model



The system actively monitors the load on each processor. High load causes the execution context to be moved to the big core, and equally, when the load is low, the execution is moved to the LITTLE core. Only one processor in the pairing can be active at any time. When the load is moved from an outbound core to an inbound core the former is switched off. This model allows a mix of big and LITTLE cores to be active at any one time.

big.LITTLE MP operation

Since a big.LITTLE system is fully coherent through the CCI-400 another model is to allow both Cortex-A15 and Cortex-A7 processors to be powered on and simultaneously executing code. This is called big.LITTLE MP (which is essentially a Heterogeneous Multi-Processing paradigm). This is the most sophisticated and flexible mode for a big.LITTLE system, involving scaling a single execution environment across both clusters. In this use model a Cortex-A15 processor core is powered on and executing simultaneously with a Cortex-A7 processor core if there are active threads that need such a level of processing performance. If not, only the Cortex-A7 processor needs to be powered on. Since processor cores are not explicitly matched, asymmetric cluster topologies are simpler to support with MP operation.

With big.LITTLE MP the operating system's task scheduler is aware of the power-performance capabilities of the different processors and maps tasks to suitable processors. Here the operating system runs on all processors in all clusters which may be operating simultaneously. The OS attempts to map tasks to processors that are best suited to running those tasks, and will power off unused, or under-used processors.

ARM has produced a set of modifications to the Linux kernel known as the ARM big.LITTLE MP HMP patches. These modifications have been evaluated with Android on ARM's big.LITTLE test silicon and provide excellent power-performance.

Multi-processor support in the Linux kernel assumes that all processors have identical performance capabilities and therefore a task can be allocated to any available processor. Therefore supporting full big.LITTLE MP requires significant changes to the scheduling and power management components of Linux. This work is already proceeding well within the open community with the first silicon devices showing very promising results.

Mobile Usage Profile

One of the characteristics that makes big.LITTLE advantageous is the varied performance requirements of typical mobile workloads. The graph below shows the percentage of time spent in DVFS states, and in idle and full shutdown states, by two cores in a currently shipping Cortex-A9 based mobile device. In the diagram, the red color indicates the highest frequency operating point, while the green colored regions indicate the lowest frequency operating point, and colors in between represent intermediate frequencies. In addition to the DVFS states, the OS power management can idle a CPU. The light blue regions in the graph indicate this idle time. When a CPU has been idled for a long enough period, the system power control software may take a core to a full shutdown to save leakage power. This is show by the darkest color on the graph.

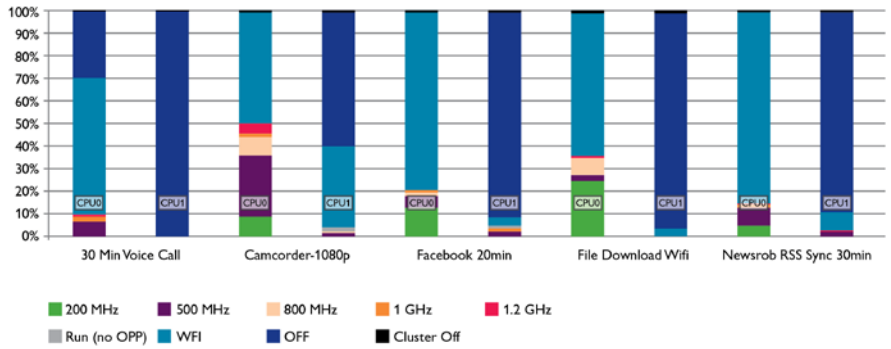


Figure 6: DVFS residency for low intensity use cases

It is clear from the graph above that the applications processors spend a considerable portion of time in lower frequency states across several common workloads. In a big.LITTLE system, the SoC would have the opportunity to run all but the dark red portions of the work on a lower power Cortex-A7 CPU. In the following graph, more intense workloads are analyzed in the same way, and even in these cases there is significant opportunity to map frequencies below 1GHz to a Cortex-A7 processor, which is known to provide performance per clock with 5~10% of the Cortex-A9 processor.

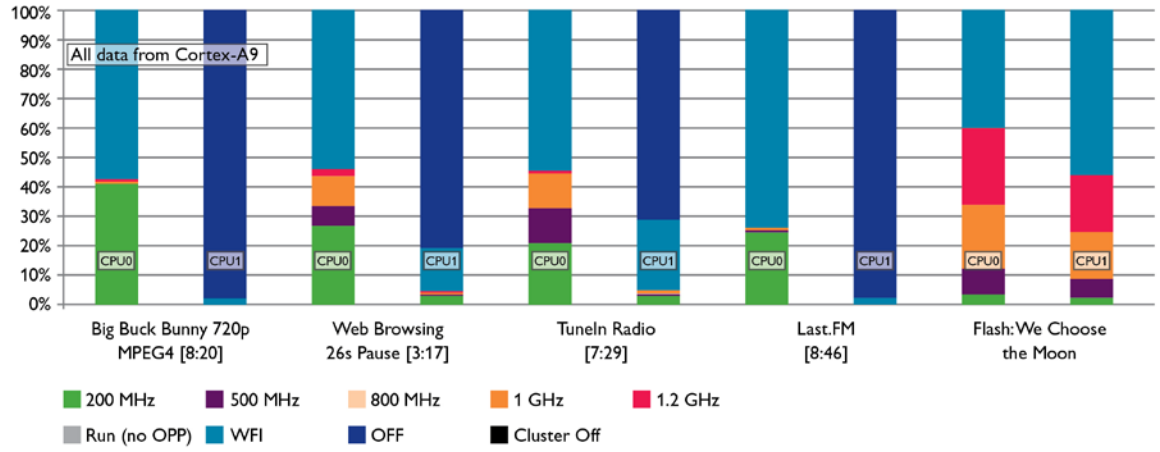


Figure 7: DVFS residency for low intensity use cases

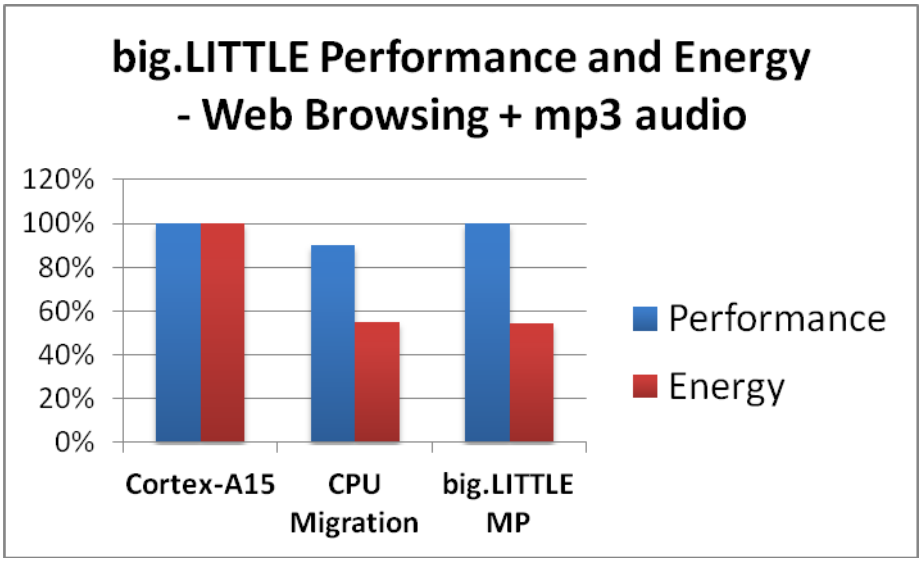
Performance and Power Analysis: big.LITTLE Test Chip

User level software has been running on top of big.LITTLE scheduling since 2011, however only on software models of the cores and interconnect. To fully evaluate the performance, power savings, and appropriate tuning of a big.LITTLE system it was necessary to build a test chip that could run user software at full speed. The ARM test chip came back from the manufacturing facility in early summer 2012, and within a few weeks the test silicon was running on a development board and running full

Linux and Android Ice Cream Sandwich. (Jelly Bean is running as well, but the results in this paper are from ICS). The test chip consists of a dual-core Cortex-A15 cluster, a tri-core Cortex-A7 cluster, and the CCI-400 cache coherent interconnect. The test chip does not include a GPU, which affects some of the user benchmarks, but the platform is able to run Linux and Android OSs and benchmark software.

The performance benchmarks in figure 3 were run on the Cortex-A15 and Cortex-A7 CPU clusters independently. The Cortex-A15 on the test chip has a maximum frequency of 1.2GHz, while the Cortex-A7 on the test chip has a maximum frequency of 1 GHz. The benchmarks showed the performance of the Cortex-A15 and Cortex-A7 CPUs was within the range of expected performance, although the memory system on the test-chip is a lower performance memory system than would be expected on a production big.LITTLE SoC. Based on the results from the cores running individually, we developed sufficient confidence that the platform would be accurate for measuring big.LITTLE performance. The software on the test chip platform consisted of the base Linux kernel, with CPU migration software and big.LITTLE MP patch sets applied, to allow the testing of either the CPU Migration or the big.LITTLE MP models.

The main workload used to test big.LITTLE performance was a web browser benchmark cycling through web pages, with audio playback happening in the background, on top of Android ICS. This use case allowed a fairly intense workload to be paired with a background activity with low performance requirements. The web browser cycled through web pages every 2 seconds and performed a 500-pixel page scroll action on each page, to present the system with a relatively high level of required performance. In measuring the performance and power while running this benchmark, it was first necessary to establish a performance and power baseline. That baseline was measured with the Cortex-A15 CPU cluster running standalone.



It bears mentioning that this set of results is fairly early; they come from an early version of the big.LITTLE MP patch set, which modifies the Linux scheduler away from a completely fair and balanced scheduling model, towards a big.LITTLE model. We expect performance and power improvements as the software is refined, and additional tuneable elements are explored. Another point worth mentioning, is the lack of a GPU in the test chip; this led to higher CPU loading than would exist in a system with a GPU for offload, and in situations where CPU loading is lower it is possible to make greater use of the LITTLE cores and thereby save more energy. It also has a rudimentary set of voltage and frequency operating points, and no ability to power gate individual cores, so production big.LITTLE SoCs are expected to deliver even better results. The performance of background tasks is exhibiting greater than 70% energy savings, for example.

Choosing a big.LITTLE software model
The question that is often asked is "which software model to choose?" The choice today is effectively between CPU migration and big.LITTLE MP, and there are pros and cons to each. In CPU migration, big and LITTLE cores are paired, so symmetric topologies work easily. An asymmetric topology, that is one with unequal number of big and LITTLE cores, requires additional work. Given the small size of Cortex-A7 CPU cores it may be attractive to use 4 LITTLE cores along with 1 or 2 big cores. On the positive side, CPU migration allows simpler power and performance tuning, and its reuse of existing OS power management code means that years of development and test underpin the implementation. With no modifications to the kernel scheduler, it was simpler in scope to implement than big.LITTLE MP and hence the software is more mature today. In sum, CPU migration is a great solution for products in 1st half 2013 and will remain a workable solution beyond that timeframe for systems that don't wish to upgrade to big.LITTLE MP.

big.LITTLE MP has several technical advantages, but is less mature today; it is currently in advanced stages of development, with very good test results as shown in this paper. big.LITTLE MP can make use of all cores in the system, as asymmetric topology support is standard with no software modifications. It offers greater opportunity for performance and power benefits. As an example, it can make use of all cores simultaneously for greater performance, or tune DVFS settings and scheduler settings differently on big and LITTLE for greater power savings. It allows fine-grained selection of cores by the scheduler, and generally offers more tuning parameters. This greater flexibility does come at a cost, as more tuning is required to extract the full performance and power benefits from a big.LITTLE MP platform. This is not so different than mobile SoCs of the last several years, where silicon vendors and OEMs have tuned the OS power management settings and DVFS parameters for the device – big.LITTLE MP extends that tuning process to include new parameters that enable greater power savings and increased responsiveness from the performance-optimized 'big' cores.

big.LITTLE MP is maturing quickly but not yet upstreamed into the mainline. It is available now for partner integration, and is expected to be upstreamed in the second half of 2013. Fortunately, no hardware changes are required to support big.LITTLE MP, so it is possible for a silicon vendor to deploy a platform with CPU migration and upgrade to big.LITTLE MP with a kernel update to deployed platforms, or to build a big.LITTLE SoC today to intercept big.LITTLE MP software in the 2nd half of 2013.

While big.LITTLE MP is not deployed in production yet, the software is running as demonstrated in the results shown in this paper and it has been demonstrated on silicon vendor development platforms. The big.LITTLE MP software ran on our test system "out of the box", and efforts are now being focused on hardening the software, and tuning the system performance for best results against a wide variety of use cases. Some of the tuneable elements being exercised include the load balancing policy of the scheduler, the up and down migration point, and thread priority. System tuning is ongoing at ARM and with silicon partners in each of these areas. ARM is making regular monthly releases of the big.LITTLE MP patchset in the open source, with the latest tuning in place for the test chip platform, test results, and documentation. The current patchset is available from Linaro at:
http://git.linaro.org/gitweb?p=arm/big.LITTLE/mp.git;a=summary
The CPU migration software is available for Linaro members now.

Next-generation big.LITTLE hardware
The Cortex-A15 and Cortex-A7 cores represent the first generation of big.LITTLE hardware. ARM has announced 2 new CPU cores that are also capable of big.LITTLE processing, the Cortex-A57 and the Cortex-A53 processors. The Cortex-A57 processor is a performance optimized CPU, bringing more than 25% more performance per clock cycle, higher frequency capability, and slightly higher efficiency than the Cortex-A15 processor. The Cortex-A53 processor is a LITTLE core with 40% more performance per clock cycle, at about the same power efficiency as Cortex-A7 processor.

Both these new processors are architecturally identical and introduce support for the ARMv8 architecture, which introduces improved NEON and floating point capability, cryptography acceleration, and 64-bit support. Both cores also support a next-generation coherent interconnect in addition to AMBA4 ACE, and they can run in AArch32 mode to run existing code in the same fashion as current ARMv7 CPU cores. The support for 64-bit and additional general-purpose registers is implemented in an elegant and efficient way with little added power. Microarchitectural enhancements are introduced to give each core more throughput per instruction clock cycle. These new cores will support big.LITTLE in the same way as the current Cortex-A15 and Cortex-A7 processors after minor updates to the software to support 64b addressing modes. Both processors will be available to lead silicon partners in 2013, with silicon production expected in 2014. In the meantime, the first big.LITTLE silicon based on Cortex-A15 and Cortex-A7 has been announced and demonstrated by Samsung and Renesas, with at least another five ARM partners planning big.LITTLE implementations during 2013.

Conclusion

This white paper has described the first big.LITTLE system from ARM. The combination of a fully coherent system with Cortex-A15 and Cortex-A7 opens up new processing possibilities beyond what is possible in current high-performance mobile platforms.

A big.LITTLE system opens the door to an extremely wide dynamic range of power and performance control points. This would otherwise not be possible in implementations comprising a single type of processor. This wide dynamic range provides the perfect execution environment for workloads that we see in devices today, which are often, composed of a mixture of high demand and low demand threads. This is complemented by the opportunity to create an extremely energy-efficient implementation of Cortex-A7 since it will be the workhorse of the platform.

Through these implementation techniques and the variety of use-models, big.LITTLE provides the opportunity to raise performance and extend battery life in the next generation of mobile platforms.
