WHITE PAPER

Big.LITTLE Processing with ARM Cortex™-A15 & Cortex-A7

Improving Energy Efficiency in High-Performance Mobile Platforms

Peter Greenhalgh, ARM **September 2011**

This paper presents the rationale and design behind the first big.LITTLE system from ARM based on the high-performance Cortex-A15 processor, the energy efficient Cortex-A7 processor, the coherent CCI-400 interconnect and supporting IP.

Contents

Introduction

The range of performance being demanded from modern, high-performance, mobile platforms is unprecedented. Users require platforms to be accomplished at high processing intensity tasks such as gaming and web browsing while providing long battery life for low processing intensity tasks such as texting, e-mail and audio.

In the first big.LITTLE system from ARM a 'big' ARM® Cortex™-A15 processor is paired with a 'LITTLE' Cortex™-A7 processor to create a system that can accomplish both high intensity and low intensity tasks in the most energy efficient manner. By coherently connecting the Cortex-A15 and Cortex-A7 processors via the CCI-400 coherent interconnect the system is flexible enough to support a variety of big.LITTLE use models, which can be tailored to the processing requirements of the tasks.

The Processors

The central tenet of big.LITTLE is that the processors are architecturally identical. Both Cortex-A15 and Cortex-A7 implement the full ARM v7A architecture including Virtualization and Large Physical Address Extensions. Accordingly all instructions will execute in an architecturally consistent way on both Cortex-A15 and Cortex-A7, albeit with different performances.

The implementation defined feature set of Cortex-A15 and Cortex-A7 is also similar. Both processors can be configured to have between one and four cores and both integrate a level-2 cache inside the processing cluster. Additionally, each processor implements a single AMBA® 4 coherent interface that can be connected to a coherent interconnect such as CCI-400.

It is in the micro-architectures that the differences between Cortex-A15 and Cortex-A7 become clear. While Cortex-A7 (Figure 1) is an in-order, non-symmetric dual-issue processor with a pipeline length of between 8-stages and 10-stages, Cortex-A15 (Figure 2) is an out-of-order sustained triple-issue processor with a pipeline length of between 15-stages and 24-stages.

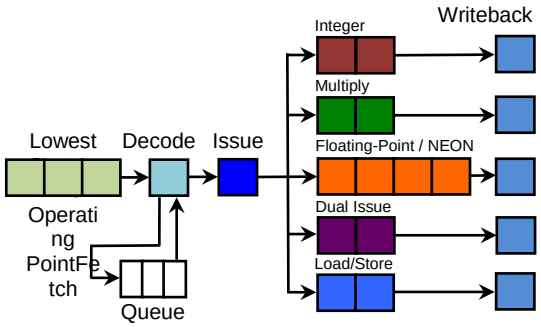

Figure 1 Cortex-A7 Pipeline

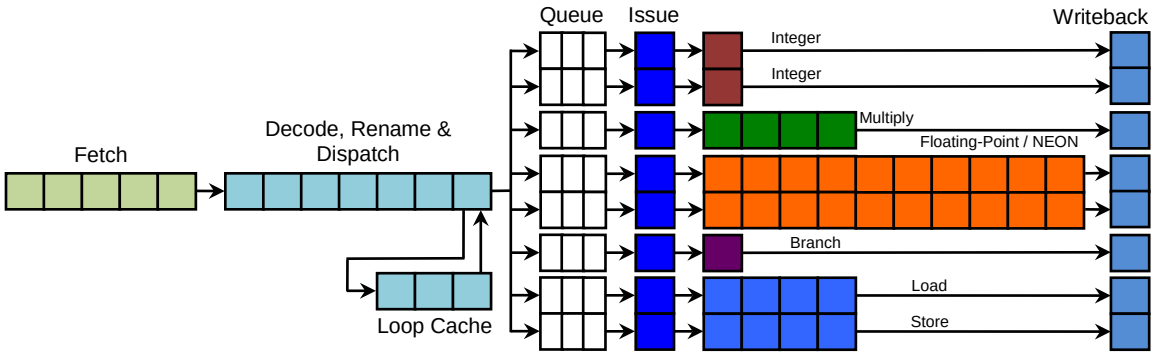

Figure 2 Cortex-A15 Pipeline

Since the energy consumed by the execution of an instruction is partially related to the number of pipeline stages it must traverse, a significant difference in energy between Cortex-A15 and Cortex-A7 comes from the different pipeline lengths.

In general, there is a different ethos taken in the Cortex-A15 micro-architecture than with the Cortex-A7 micro-architecture. When appropriate, Cortex-A15 trades off energy efficiency for performance, while Cortex-A7 will trade off performance for energy efficiency. A good example of these micro-architectural trade-offs is in the level-2 cache design. While a more area optimized approach would have been to share a single level-2 cache between Cortex-A15 and Cortex-A7 this part of the design can benefit from optimizations in favor of energy efficiency or performance. As such Cortex-A15 and Cortex-A7 have integrated level-2 caches.

Table 1 illustrates the difference in performance and energy between Cortex-A15 and Cortex-A7 across a variety of benchmarks and micro-benchmarks. The first column describes the uplift in performance from Cortex-A7 to Cortex-A15, while the second column considers both the performance and power difference to show the improvement in energy efficiency from Cortex-A15 to Cortex-A7. All measurements are on complete, frequency optimized layouts of Cortex-A15 and Cortex-A7 using the same cell and RAM libraries. All code that is executed on Cortex-A7 is compiled for Cortex-A15.

	Cortex-A15 vs Cortex-A7 Performance	Cortex-A7 vs Cortex-A15 Energy Efficiency
Dhrystone	1.9x	3.5x
FDCT	2.3x	3.8x
IMDCT	3.0x	3.0x
MemCopy L1	1.9x	2.3x
MemCopy L2	1.9x	3.4x

Table 1 Cortex-A15 & Cortex-A7 Performance & Energy Comparison

It should be observed from Table 1 that although Cortex-A7 is labeled the “LITTLE” processor its performance potential is considerable. In fact, due to micro-architecture advances Cortex-A7 provides higher performance than current Cortex-A8 based implementations for a fraction of the power. As such a significant amount of processing can remain on Cortex-A7 without resorting to Cortex-A15.

The System

To create a compelling big.LITTLE solution the system around the processors must also be considered.

A key part is the CCI-400 interconnect which facilitates full coherency between Cortex-A15 and Cortex-A7 as well as IO coherency for components such as a GPU. Through optimizations around the transaction characteristics of Cortex-A15 and Cortex-A7 as well as by considering the paths to main memory and the system, a single solution is offered with the highest possible performance.

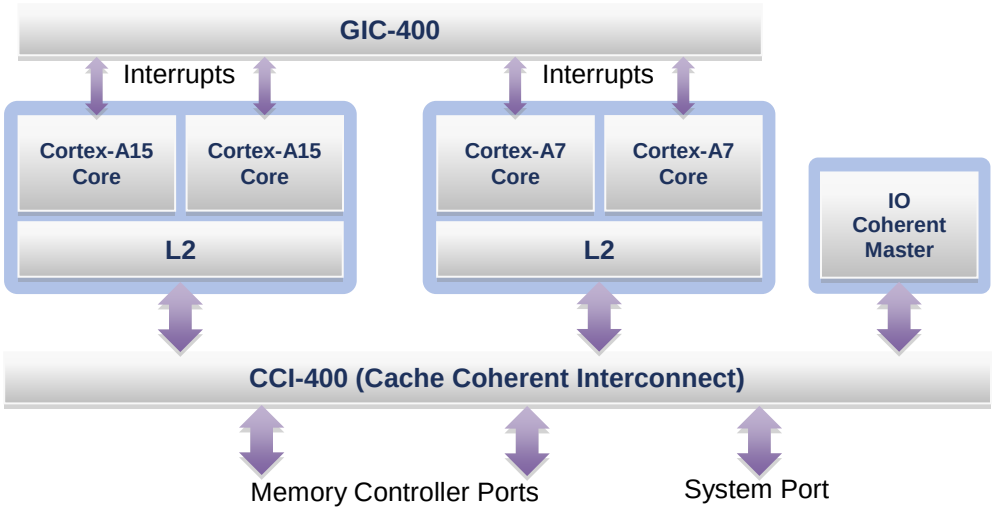

Figure 3 Cortex-A15 CCI Cortex-A7 System

Another element of the big.LITTLE system is a shared Generic Interrupt Controller (GIC-400). As well as being able to distribute up to 480-interrupts to Cortex-A15 and Cortex-A7, the programmable nature of the GIC-400 allows interrupts to be migrated between any cores in the Cortex-A15 or Cortex-A7 clusters.

From the perspective of trace and debug, both Cortex-A15 and Cortex-A7 offer trace solutions and are both compliant with the Debug v7.1 architecture. Full support for big.LITTLE debug and trace is provided through CoreSight™ SoC.

A final point to consider is that a big.LITTLE system incorporating Cortex-A15, Cortex-A7, CCI-400 and the GIC-400 is optimal for all big.LITTLE use-models. There is no configurable feature or optimization that favors a particular use-model. However, to reduce software complexity in the big.LITTLE task migration use model it is recommended that the same number of cores be implemented in the Cortex-A15 cluster and Cortex-A7 cluster.

big.LITTLE Task Migration Use Model

In the big.LITTLE task migration use model the OS and applications only ever execute on Cortex-A15 or Cortex-A7 and never both processors at the same time. This use-model is a natural extension to the Dynamic Voltage and Frequency Scaling (DVFS), operating points provided by current mobile platforms with a single application processor to allow the OS to match the performance of the platform to the performance required by the application.

However, in a Cortex-A15-Cortex-A7 platform these operating points are applied both to Cortex-A15 and Cortex-A7. When Cortex-A7 is executing the OS can tune the operating points as it would for an existing platform with a single applications processor. Once Cortex-A7 is at its highest operating point if more performance is required a task migration can be invoked that picks up the OS and applications and moves them to Cortex-A15.

This allows low and medium intensity applications to be executed on Cortex-A7 with better energy efficiency than Cortex-A15 can achieve while the high intensity applications that characterize today's smartphones can execute on Cortex-A15.

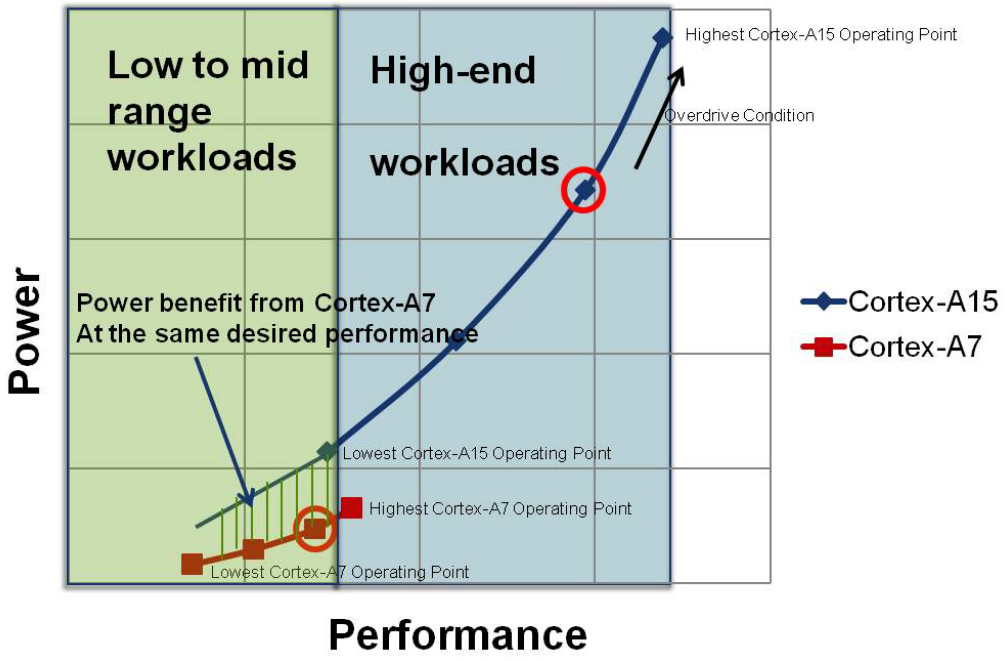

Figure 4 Cortex-A15-Cortex-A7 DVFS Curves

An important consideration of a big.LITTLE system is the time it takes to migrate a task between the Cortex-A15 cluster and the Cortex-A7 cluster. If it takes too long then it may become noticeable to the operating system and the system power may outweigh the benefit of task migration for some time. Therefore, the Cortex-A15-Cortex-A7 system is designed to migrate in less than 20,000-cycles, or 20-microSeconds with processors operating at 1GHz.

One of the reasons the task migration can be so fast is that the amount of processor state involved in the task migration is relatively small. The processor that is going to be turned off, which is termed the outbound processor, must have all of the integer and Advanced SIMD register files contents saved along with the entire CP15 configuration state. The processor that is going to resume execution, which is termed the inbound processor, must then restore all of the state saved from the outbound processor. Additionally, any active interrupts that are being controlled by the GIC-400 must be migrated. Less than 2,000 instructions are required to achieve save-restore and because the two processors are architecturally identical there is a one-to-one mapping between state registers in the inbound and outbound processors.

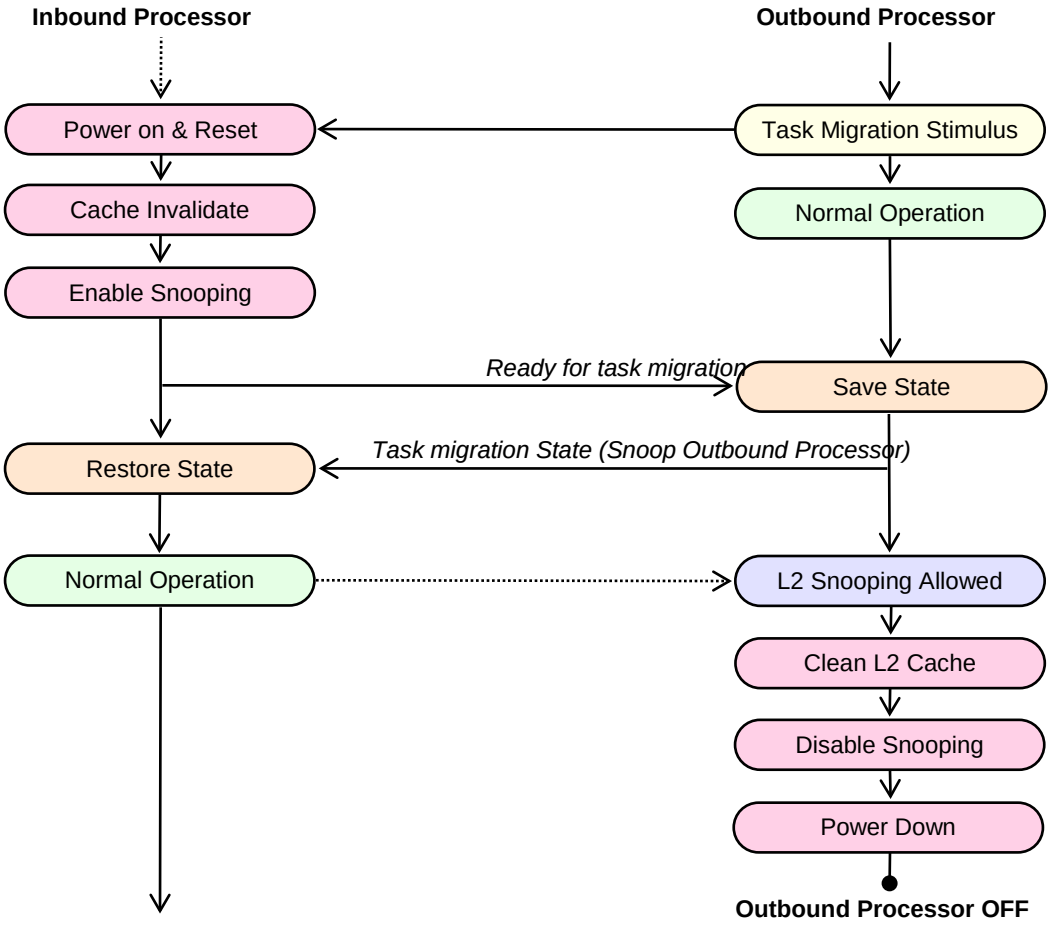

Figure 5 big-LITTLE Task Migration

Figure 5 describes the task migration process between inbound and outbound processors. Coherency is clearly a critical enabler in achieving a fast task migration time 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 level-2 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 level-2 cache of the outbound processor cannot be allocated too, it will eventually need to be cleaned and powered off to save leakage power.

It should also be observed that normal execution of the thread occurs during the task migration process. The only “black out” period is during the task migration when interrupts are disabled and state is transferred from the outbound to the inbound processor.

big.LITTLE MP Use Model

Since a big.LITTLE system containing Cortex-A15 and Cortex-A7 is fully coherent through CCI-400 another logical use-model is to allow both Cortex-A15 and Cortex-A7 to be powered on and simultaneously executing code. This is termed big.LITTLE MP, which is essentially Heterogeneous Multi-Processing. Note that in this use model Cortex-A15 only needs to be powered on and simultaneously executing next to Cortex-A7 if there are threads that need that level of processing performance. If not, only Cortex-A7 needs to be powered on.

big.LITTLE MP is compelling because it enables threads to be executed on the processing resource that is most appropriate. Compute intensive threads that require significant amounts of processing performance, as their output is user visible, can be allocated to Cortex-A15. Threads that are I/O heavy or that do not produce a result that is time critical to the user can be executed on Cortex-A7.

A simple example of a non-time critical thread is one associated with e-mail updates. While web browsing the user will want email updates to continue, but it does not matter if they are done at Cortex-A15 performance levels or Cortex-A7 performance levels. Since Cortex-A7 is a more energy efficient processor it makes more sense to take a LITTLE longer, but consume less battery life.

Finally, as a fully coherent system can create a significant volume of coherent transactions, Cortex-A15, Cortex-A7 and CCI-400 have been designed to cope with worst case snooping scenarios. This includes the case where a Mali™-T604 GPU is connected to one of the I/O coherent CCI-400 ports and every transaction is snooping Cortex-A15 and Cortex-A7 at the same time as Cortex-A15 and Cortex-A7 are snooping each other.

Software

As part of the big.LITTLE system, ARM provides a software switcher for use with Cortex-A15, Cortex-A7, CCI-400 and the GIC-400. The switcher serves two purposes:

The first purpose is to provide all of the mechanisms required for task migration between Cortex-A15 and Cortex-A7. As well as the processor state save-restore this also includes the code required to bring the processors in and out of coherency, control snooping in the interconnect and migrate interrupts. The switcher can be used as-is or the code can be used as a template for integration in to the operating system.

A second purpose is to hide the small number of programmer’s model differences between Cortex-A15 and Cortex-A7 from the Operating System. While Cortex-A15 and Cortex-A7 are architecturally identical and all registers are read and written in an architecturally consistent manner, the contents of the registers may not always be identical. So Cortex-A15 and Cortex-A7 are not totally programmers model identical in all cases.

For example, the contents of the Main ID register that identifies the processor will be different between Cortex-A15 and Cortex-A7 as will the contents of the CP15 registers that describe the level-1 and level-2 cache topologies. Fortunately, since both Cortex-A15 and Cortex-A7 implement the virtualization extensions OS accesses to these registers can be trapped to the hypervisor layer which is where the switcher can handle them.

The switcher enables a big.LITTLE system to be built today with current Operating Systems. However, as in the case of the state save-restore code, it may be that the small number of programmer's model differences between Cortex-A15 and Cortex-A7 may want to be handled by the OS rather than the switcher.

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.

Rather than having to compromise the implementation of a single applications processor to cope with high and low intensity tasks, the big.LITTLE system opens the door to an extremely high performance implementation of Cortex-A15 since it will only be powered on when that performance is needed. This is complimented 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.