EMBEDDED SYSTEMS – PERFORMANCE EVALUATION, EMBEDDED MULTIPROCESSORS

Asst. Prof. M.Sc. Slavyanov K.[1], Asst. Prof. M.Sc. Kulev N. [2]

Faculty of Artillery, Air defense and CIS – Shumen, National Military University, Bulgaria
k.o.slavyanov@gmail.com[1]
nz_kulev@abv.bg[2]

Abstract: *All embedded systems need high performance and high energy efficiency for their faster real-time task execution. These can be achieved only with the implementation of new innovative technology on higher level than the processor, power battery and standardized test by the EDN Embedded Microprocessor Benchmark Consortium. The big.LITTLE technology is the reason so many users to enjoy the big performance and high power efficiency in their pockets nowadays.*

Keywords: REAL-TIME EXECUTION, GLOBAL TASK SCHEDULING, GLOBAL INTERRUPT CONTROL,

1. Introduction

The trends in development of smartphones as an architecture class often make the real-time requirements very important. The real-time importance is even the reason of maximum execution time constraints for significant part of the application.

For example each frame rendering time is constrained by the processor time needed to receive and execute the frame before the next one arrives. This is characteristic of the so-called hard real-time systems. Some applications have even more sophisticated requirements. The average time for concrete task can be constrained by the limits in score when some maximum time is exceeded – typically named soft real-time. According to these approaches some incident miss are accepted only if the misses are not so frequent events.

It is common for the real-time performance to be very application dependable. Thus it can be measured by specific or application based program kernels or standardized benchmarks. The hard-real time system realization can be organized by tree variables. First of all it is the frequency each task uses.

Tightly connected to this are the particular hardware and software means to realize the frequency without any problem. Often it can be very difficult the new improvements for desktop products to be evaluated by real-time execution analyses. For example the speculative branch execution, cache memory and other technologies can put some indetermination in the classical code creating.

2. Preconditions and means for resolving the problem

Specific part of the code can be executed very efficiently or very inefficiently in strong dependence to the ability of hardware branch predicting techniques and cache memory performance expected. Designers have to analyze the code precisely, accepting the worst-case execution time – WCET. The traditional microprocessor approach can be very pessimistic if there is assumption that all the branches are mistaken and all cash searches end with cache misses. Hence in system creation period achievement of concrete WCET have to be proved, even if not so high-end system can be satisfied. [1]

In order to answer the new challenges in hard real-time systems in combination with common branch control architecture technologies and access locality it is possible entire processor design to be reconstructed. Even if the branch predicting technique perform very well the system reaction is more predictable by usage of static "hint bits" or flags attached to the instructions. By other point of view, although the cache use is better than the software managed memories on the chip, the last ones always have the latencies predicted.

In some embedded processors the cache can be turned very well to software managed memory on the chip by line locking. In that way some cache line can be locked in place and replaced only if it is unlocked.

Performance evaluation

For those embedded systems witch can be characterized by kernel performance for their applications, one of the best benchmarking is that of the EDN Embedded Microprocessor Benchmark Consortium (EEMBC). EEMBC is divided to six subclasses: games, telecommunication, network devices, automotive/industrial, office and customer. Although many applications for embedded systems are sensitive to small core's performance, often the performance of the entire application (it may consist few thousand lines of code) is also critical. Hence many benchmarks for embedded systems are used only for parts from overall application performance.

The cost and power in the embedded market very often are more important than the performance. Besides the processor cost with all the interface circuits needed, next to it in price tag usually is the memory of the embedded system. In contrast to desktop or servers, common embedded systems do not use secondary memory. The variant full application to be in flash or DRAM is chosen.

The mere fact that most of the systems of this class, like PDAs and mobile phones are constrained by cost and form factor, causes the memory capacity needed for each application to be critical. Because of that power consumption more frequently is the main factor in processor choice, especially for battery powered systems. EEMBC EnergyBench is a product, making possible the energy consumption profile during EEMBC benchmarking process. Other application in that range is Energymark score for developer's ability to test and indicate their processors with standardized and certified performance and power energy efficiency results. For general purpose testing and customer popularization EEMBC standardize the National Instruments product LabVIEW with proper GUI and toolboxes. [2]

3. Results and discussion

Embedded multiprocessors

The Multi processors are commonly used in server and some desktop configurations like the multiprocessors of vendors like Sun, Compaq and Apple. In embedded systems space, many special-purpose models use specific multiprocessor configurations like Sony PlayStation in games for example. Many specialized embedded designs use programmable general purpose processor or special-purpose DSP, finite-state units for stream-oriented Input/Output. From computer graphic and media to telecommunication products the use of such multiprocessors is traditional.

In those systems the internal processor interactions are perfectly organized and comparatively simple, mainly because the use of a simple communication channel (placed on silicon), but the base and heavy task is the communication protocols cooperation of the IO

communication units, built on few general-purpose processors. This type of multiprocessing is intelligible oriented in telecommunication and network field, where the scalability is critical.

Design of that type is the MXP processor of empowerTel Networks for use in VOIP systems. It has four base components: serial voice stream interface, full Ethernet interface with MAC layer, fast packet distribution support and channel look up and four MIPS 32 class R4000 processors with 12 KB cache each. MIPS processors are used for code execution of the VOIP channels, in this case with quality control echo correction, simple compression and packet encryption. If the goal is running of more independent voice steams, the multiprocessor is the perfect solution. [2]

Comparatively small size of the MIPS core means not so much transistors on the chip and the future opportunity for more channels realization, next to more sophisticated echo correction, voice activity detection and compression. Multiprocessors are widely spread because of two reasons.

The first one is that the problems with binary software compatibility, typical for desktop and servers are not so significant in embedded systems. Commonly the software for embedded application is written especially for the individual application or is well modified. Hence the VLIW is more suitable than the superscalar in high-end embedded instruction level parallelism.

Second place is for the fact that the applications often has natural parallelism, typical for game consoles, network switches and sell phones. The small constraints in thread level parallelism, complemented with more efficient usage of silicone field lead to widespread use of the multiprocessors in embedded systems in response to the more performance demanded.

The performance growing speed for the tablets and smartphones is far higher than the improvement rate in battery capacity or in semiconductor space. In the same time the customers expect much longer battery life in time of small technology improvements. These conflicting facts make the door for new ideas and approaches wide open for the mobile systems on a chip (SoC), approaches different from whatever improvements in processor technology and the power management.

Big.LITTLE is one of the technologies for power management used to safe energy in mobile SoC. In combination with Dynamic Voltage and Frequency Scaling (DVFS), clock gating, temperature control, core power gating, retention nodes etc. it provides full set of power control tools in SoC.

Big.LITTLE works well on the fact that the usage profile of tablets and smartphones is very dynamic. Hard task periods such as web page initial loading and game graphic processing alternate with comparatively long time with low activity on tasks like text reading, user response waiting in game processing or common operations like text, e-mail or audio running.

The multiprocessor ARM big.LITTLE system has heterogeneous architecture based on two processor types - “LITTLE” processor, designed for maximum power efficiency and “big” processor capable to achieve best computing performance. The big performance cores tend to be used in burst working mode with small duration to act on peak frequency while more than 80 % of the working time the job is handled by the smaller cores on moderate frequency. The first widespread big.LITTLE realization is used in one system of the ARM Cortex – A5 and the ARM Cortex-A7 processors (fig.1) [3, 4]

Multiprocessors and accelerating systems in their work together need to share information. The additional processing cores improve the system performance and produce better efficiency, but to achieve that goal the shared data need to be perfectly managed to be the correct one where it have to be used. High-performance and high-efficiency clusters are connected with cache coherent interconnection like ARM CoreLink CCI-400 Cache Coherent Interconnect. That high-performance, energy efficient cache coherent system, is developed to be connection interface for all the processors and the dynamic memory access controller like CoreLink DMC-400. The technology finally provides the virtual memory management and hardware coherence, vital for the software simplicity and scalability.

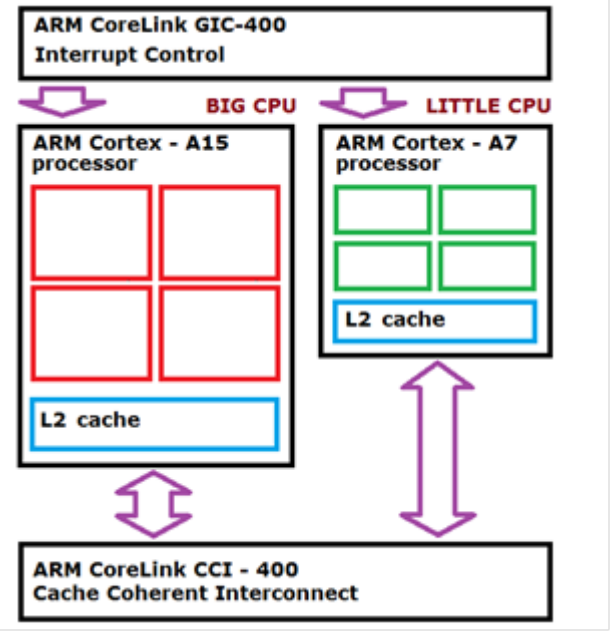

Fig. 1 *Big.LITTLE system using ARM Cortex-A15 and ARM Cortex-A7*

The operational system uses all processors like if it is one in their place. The user software in big.LITTLE SoC environment is identical to standard SMP processor. The main task here is correct forwarding of every task to the right processor, answer provided by the ARM CoreLink GIC-400 Interrupt Control and shown on figure 2. That system provides OS awareness for the big and the LITTLE processor status with the ability to navigate each execution thread to the best chosen processor based on dynamic attendance to each core. [3]

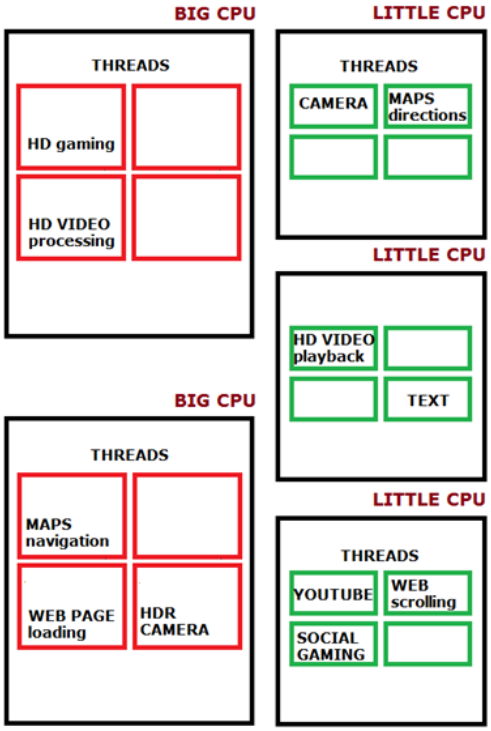

Fig. 2 *ARM Global task scheduling*

The software keeps every loaded thread activation history in order to be aware of the next execution requirements. The Global Task Scheduling enables task distribution between all processor

cores with 75% less power consumption for the same or higher performance.

To be the big.LITTLE processor invisible for the software, the processor subsystems should use fully coherent cache, the big and the LITTLE processors must be completely architecturally compatible. It means to use the same instruction set, as well as to be able to use the same extensions like virtualization, long physical addressing etc. The ARM Cortex-A series are designed to meet these requirements in recommended combination like the examples on Figure 3.

	1st generation ARMv7 – 32-bit,40 bit physical address	**2nd generation** ARMv8 – 32-bit/64 bit
High-preformance CPU	Cortex-A15	Cortex-A57
High-efficiency CPU	Cortex-A7	Cortex-A53

Fig. 3 *ARM Cortex-A series big.LITTLE recommended combination*

In each combination mentioned above high-performance and high-efficient cluster consists of no more than four cores. Smartphone applications mostly use one or two high-performance cores to handle performance expectation. The high-end smartphones and tablets on the other hand use the advantages of four big and four little cores together for their software. By Global Task Scheduling software all processors can be active in working, in order to provide hard load acceleration ability in combination with power efficiency. System with these features can be designed with cache coherent interconnect, Global Interrupt control and other components in addition like on the figure 4. [3]

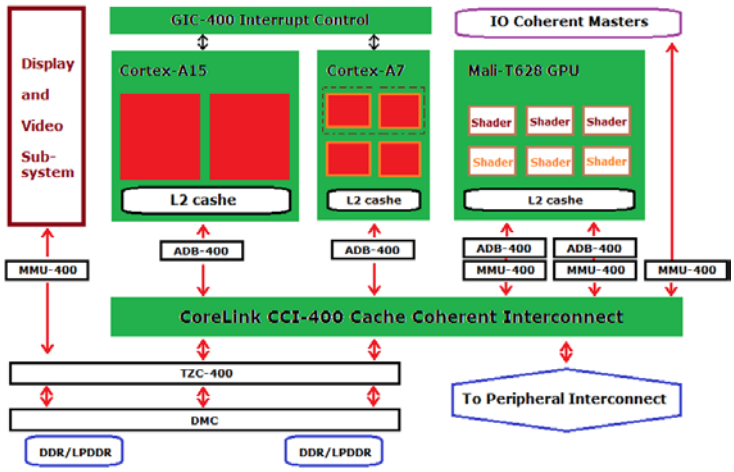

Fig. 4 *Big.LITTLE hardware requirements*

4. Conclusion

Basic characteristics of the big.LITTLE architecture are two.

The first one is high performance with some significant advantages: The big cores lead to extremely little respond time and very fast processing of more complex web content. The usage of big.LITTLE reduces processor power consumption, supports faster and more detailed graphics in the same SoC cost; The big.LITTLE technology makes possible high-end tablet performance to fit in a pocket size devices.

Second is the longer battery life. Big.LITTLE technology uses the most appropriate cores for the processor power demand. The power efficient LITTLE cores cannot be underrated too – they handle the common simple tasks like text and e-mail but take the workload about 95% of the time and in that way the power profile looks like such on the low power devices. Significant advantage is more than 40% energy safe than in standard SoC loads like web browsing. It is possible for the cores in the same time to achieve effective task distribution, allowing overall performance to be 40% higher for big multithread load in contrast to ARM Cortex A15 processor alone. [5]

The next in a row ARMv8 processor architectures like Cortex-A53 and ARM cortex-A57 fully support the big.LITTLE technology.

5. Literature

1. Hennessy, John L. Patterson David A., Computer Architecture - Fifth edition, 2012,
2. Hennessy, John L. Patterson David A., Computer Architecture - Fifth edition, Appendix E - , 2012
3. http://www.arm.com/products/processors/technologies/biglittleprocessing.php (13.04.2014).
4. big.LITTLE Technology: The Future of Mobile, http://www.arm.com/files/pdf/big_LITTLE_Technology_the_Futue_of_Mobile.pdf (11.04.2014)
5. http://www.thinkbiglittle.com/ (12.04.2014)