Energy-Efficient Computing with Heterogeneous Multi-Cores

Tulika Mitra
School of Computing
National University of Singapore
Email: tulika@comp.nus.edu.sg

***Abstract*—Homogeneous multi-cores, while ubiquitous today, cannot provide the desired performance and energy-efficiency for various application domains. A promising alternative design is heterogeneous multi-core architecture where cores with different functional characteristics (CPU, GPU, DSP etc.) and/or performance-energy characteristics (simple versus complex micro-architecture) co-exist on the same die. Given an application, only the cores that best fit the application can be exploited leading to faster and energy-efficient computing. This paper describes heterogeneous multi-core architectures and the runtime management strategies to leverage the potential of such architectures for improved energy-efficiency.**

I. Introduction

The past decade has witnessed an unprecedented and exponential growth in the amount of data being produced, stored, transported, processed, and displayed. The journey of zettabyte of data from the myriad of end-user devices in the form of PCs, tablets, smart phones through the ubiquitous wired/wireless communication infrastructure to the enormous data centers is fueled by electricity. The Information-Communication-Technologies (ICT) ecosystem consumes an estimated 10% of the world electricity and accounts for roughly 2% of the global carbon emission. In an energy-constrained world, it is imperative to develop energy-efficient computing techniques that lead to responsible and sustainable energy consumption with minimal impact on the environment. Thus power and energy have become first class design constraints in all computing systems starting from smartphones to massive data centers.

Heterogeneous multi-cores offer a promising solution towards energy-efficient computing. Computing systems made an irreversible transition towards multi-core architectures in early 2000. As of now, homogeneous multi-cores are prevalent in all computing systems starting from smartphones to PCs to enterprise servers. Unfortunately, homogeneous multi-cores cannot provide the desired performance and energy-efficiency for diverse application domains. A promising alternative design is heterogeneous multi-core architecture where cores with different functional characteristics (CPU, GPU, DSP etc.) and/or performance-energy characteristics (simple versus complex micro-architecture) co-exist on the same die. Given an application, only the cores that best fit the application can be exploited leading to faster and power-efficient computing.

Another reason behind the emergence of the heterogeneous multi-cores is the thermal design power constraint. While the number of cores on die continues to increase due to Moore's Law [1], the failure of Dennard scaling [2] has led to rising power density that forces a significant fraction of the cores to be kept powered down at any point in time. This phenomenon, known as the Dark Silicon [3], provides opportunities for heterogeneous computing as only the appropriate cores need to switch on for efficient processing under thermal constraints.

978-1-4799-4833-8/14/$31.00 2014 IEEE

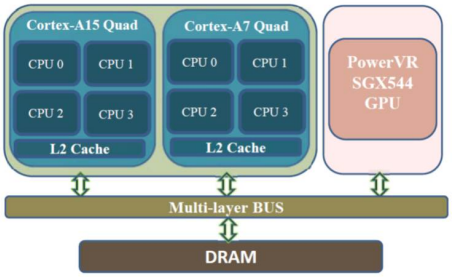

Fig. 1: Exynos 5 Octa SoC simplified block diagram

Heterogeneous multi-cores, however, present a number of unique challenges in terms of runtime management of the underlying platform so as to provide the best power-performance tradeoff. For heterogeneous multi-cores where the cores have the same instruction-set architecture (ISA) but different micro-architecture [4], the issue is to identify at runtime the core that best matches the computation in the current context. For heterogeneous multi-cores consisting of cores with different functionality, for example CPU and GPU, the runtime layer should perform coordinated power management of the different cores through voltage-frequency scaling and power-state management as opposed to the current practice of managing each core type individually.

II. Performance Heterogeneous Multi-Core

Heterogeneous computing architectures can be broadly classified into two categories: *performance heterogeneity* and *functional heterogeneity*. Performance heterogeneous multi-core architectures consist of cores with different power-performance characteristics but all sharing the same instruction-set architecture. The difference stems from distinct micro-architectural features such as in-order core versus out-of-order core. The complex cores can provide better performance at the cost of higher power consumption, while the simpler cores exhibit low-power behavior alongside lower performance. This is also known as single-ISA heterogeneous multi-core architecture [4] or asymmetric multi-core architecture. The advantage of this approach is that the same binary executable can run on all different core types depending on

the context and no additional programming effort is required. However, either the system designer or the runtime management layer has to identify the appropriate core type for each application or even for different phases within the same application. Examples of commercial performance heterogeneous multi-cores include ARM big.LITTLE [5] integrating high-performance out-of-order cores with low-power in-order cores, nVidia Kal-El (brand name Tegra3) [6] consisting of four high-performance cores with one low-power core, and more recently Wearable Processing Unit (WPU) from Ineda consisting of cores with varying power-performance characteristics [7]. An instance of the ARM big.LITTLE architecture integrating quad-core ARM Cortex-A15 (big core) and quad-core ARM Cortex-A7 (small core), as shown in Figure 1, appears in the Samsung Exynos 5 Octa SoC driving high-end Samsung Galaxy S4 smart-phones.

Performance heterogeneous multi-core architectures are a promising solution to two related but critical problems today: performance limits due to Amdahl's Law [8] and energy efficiency of multi-core computing in the dark silicon era. As Moore's Law [1] continues to improve the transition density and consequently the number of cores on a single chip, we are firmly moving towards many-core architectures with hundreds and perhaps thousands of simple homogeneous cores. However, all the contemporary and emerging applications are not perfectly parallelizable to take advantage of the abundant thread-level parallelism (TLP) offered by homogeneous multi-core architectures. Amdahl's Law [8] reminds us that even for an application with 99% parallel code, the remaining 1% sequential fraction limits the speedup to 100 even with an infinite number of cores. Thus most applications will have limited speedup with homogeneous multi-cores. Hill and Marty in a seminal paper [9] argue that heterogeneous multi-cores where the sequential code fragment can be mapped to a complex core — capable of exploiting instruction-level parallelism (ILP), for example, through out-of-order execution and hence accelerate the execution of the sequential fraction — improve the speedup of the application quite dramatically. This is because the parallel portion of the code can be accelerated through the array of simple cores offering TLP while the sequential portion can be expedited by exploiting ILP through the complex core.

The energy-efficiency advantage of heterogeneous computing is quite obvious. At any point, we simply need to turn on the core(s) that is most power-efficient for the current computing need without negatively impacting the performance. For example, in a smartphone, the low-power small core can take care of simple tasks such as email client, web browsing etc. saving energy, while the complex core has to be switched on for compute-intensive tasks such as 3D gaming, browsing flash-based websites etc. sacrificing energy. This model of computing fits in well in the dark silicon era where thermal constraints anyway restrict the fraction of cores that can be switched on at any point in time; so it is beneficial to switch on the appropriate cores for better energy efficiency. Note that apart from micro-architectural differences, these architectures offer additional design points in the power-performance trade-off curve through dynamic voltage-frequency scaling (DVFS) of the cores.

Figure 2 shows the power-performance heterogeneity of

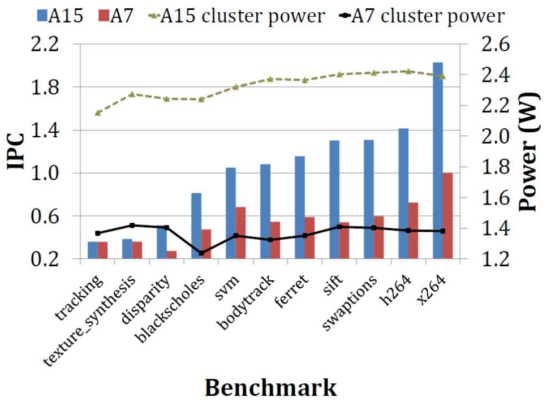

Fig. 2: Power-Performance characteristics of small, big cores

heterogeneous multi-core architecture for commercial ARM big.LITTLE architecture. The evaluation platform we use is the Versatile Express development platform comprising of a prototype chip with two Cortex-A15 cores and three Cortex-A7 cores at 45nm technology. All the cores implement ARM v7A ISA. While each core has private L1 instruction and data caches, the L2 cache is shared across all the cores within a cluster. The L2 caches across clusters are kept seamlessly coherent via the cache coherent interconnect so that an application can be easily migrated from one cluster to the other. The architecture provides DVFS feature per cluster. But all the cores within a cluster should run at the same frequency level. Moreover an idle cluster can be powered down if necessary. The chip is equipped with sensors to measure frequency, voltage, power, and energy consumption of each cluster as well as performance counters.

Figure 2 plots the Instructions Per Cycle (IPC) and the average power (Watt) for benchmark applications on Cortex-A7 and Cortex-A15 cluster, respectively. In this experiment, we set the the same voltage (1.05 Volt) and frequency (1 GHz) level for the two clusters and utilize only one core at a time to run the benchmark. Note that we can only measure the power at cluster level rather than individual core level. So the power reported in this figure corresponds to the power in a cluster even though only one core is running the benchmark application, while other cores are idle. Clearly, A15 has significantly better IPC compared to A7 (average speedup of 1.86) but far worse power behavior (1.7 times more power than A7 on an average).

Sophisticated runtime management techniques are required to leverage the unique opportunity offered by heterogeneous multi-cores towards energy-efficient computing. This includes (a) determining the core type that is most suitable for an application or the phase of an application, (b) moving the application to the appropriate core through task migration, and (c) setting the proper voltage-frequency level for the cores such that the performance requirements of the applications are satisfied at minimal energy while not exceeding the thermal design power (TDP) constraint.

The first step required in this process is an accurate power-performance estimation mechanism. As an application is running on one core type, we would like to predict its power, performance behavior on the other core types and at different voltage-frequency levels so as decide whether the application

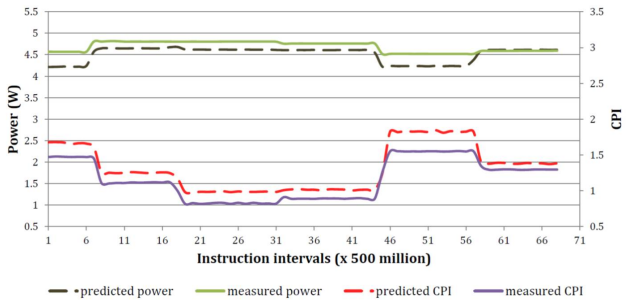

Fig. 3: Accuracy of power-performance prediction model

should be migrated and to where. We develop such a power-performance model for ARM big.LITTLE architecture [10]. This modeling is challenging for a real architecture for various reasons. First, the big core and the small core are dramatically different, not just in the pipeline organization, but also in terms of cache hierarchy and the branch predictor — a reality that is ignored in almost all works. Thus given the cache miss rate or branch misprediction on one core type, we have to estimate the same for the other core type. Second, we are constrained by the performance counters available on the cores and cannot assume additional profiling information that is simply not available such as inter-instruction dependency. We overcome these challenges through a combination of static program analysis (to identify inter-instruction dependency), mechanistic modeling that builds analytical model from an understanding of the underlying architecture (such as impact of pipeline stalls due to inter-instruction dependency and resource constraints versus miss events on performance), and empirical modeling that employs statistical inferencing techniques like regression to create an analytical model (for inter-core miss events estimation). This hybrid approach results in estimations that are fairly close to the actual values. For example, Figure 3 shows the estimated power, performance on Cortex-A15 continuously predicted from executing the benchmark *astar* on Cortex-A7 where both cores are assumed to run at 1GHz. For references, we also show the measure power, performance on Cortex-A15. It should be clear that the model can track the trends quite accurately.

These prediction are then employed to choose the suitable core type for an application or the phase of an application and its DVFS. We initially developed a control-theory based approach [11] that synergistically integrates multiple controllers (handling different constraints or optimization goals) to achieve energy-efficiency for multiple applications running on heterogeneous multi-core system. However, this approach suffers from scalability issues due to centralized decision making regarding task migration and power allocation among the cores under tight TDP constraint. Thus it is quite effective for mobile platforms with limited number of cores/clusters but is not quite practical in data center scenario with myriad of cores. We surmount this scalability issue through a distributed approach based on the solid foundations of price theory from economics [12]. The resource allocation, DVFS, task mapping, and migration are all controlled through the virtual market place, where the commodity being traded is processing unit using virtual money. The framework is realized as a collection of autonomous entities called agents, one for each task, core, cluster, and the entire chip. The performance requirement is

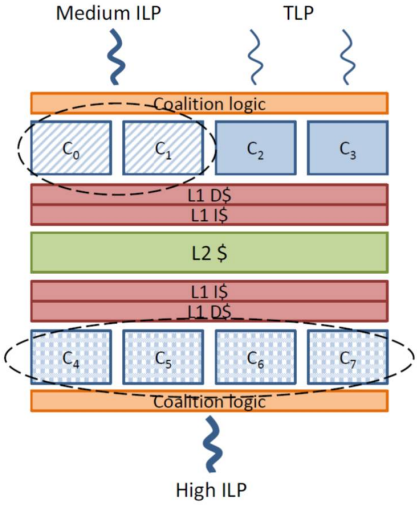

Fig. 4: Bahurupi dynamic heterogeneous multi-core

modeled as the demand while the processing capability is modeled as the supply (depends on core type and frequency). The principle of price theory states that the market is only stable at a price equilibrium, which is the price at which the supply is equal to the demand and hence corresponds to the minimal energy consumption. We implemented this framework by modifying the commodity Linux operating system on the prototype ARM big.LITTLE platform mentioned earlier. Across a range of workloads, the price theory based power management framework reduces average power consumption to 2.96W compared to 5.99W for Linux heterogeneity-aware scheduler (which makes naive task migration decision) plus on-demand governor (for DVFS) at the same or even better performance level. The framework is also highly scalable with overhead being only 12ms for 256 clusters in a datacenter compared to hundreds of ms for existing approaches based on linear solvers [13].

So far we have discussed static heterogeneous multi-cores and runtime mechanisms for power management of such systems. The difficulty with static heterogeneous multi-core is that the mix of simple and complex cores has to be freezed at design time. The next logical step forward is to design dynamic heterogeneous multi-core that can, at runtime, tailor itself according to the applications and can provide even better speedup [9]. Towards this end, we recently proposed an architecture, called Bahurupi [14] that is physically fabricated as a set of clusters, each containing four simple 2-way out-of-order cores. Figure 4 shows an example of 8-core Bahurupi architecture with two clusters (C0-C3) and (C4-C7). At runtime, two or more such simple cores within a cluster can form a coalition to create a more complex virtual core. Similarly, the simple cores participating in a complex virtual core, can be disjoined at any point of time. Thus we can create diverse range of heterogeneous multi-cores on-demand through simple reconfiguration. The highlighted cores in Figure 4 are involved in two coalitions of two (C0, C1) and four (C4-C7) cores. In this example one parallel application runs its two threads on cores C2 and C3, one medium-ILP sequential application is scheduled to coalition (C0–C1) and one high-ILP sequential application is scheduled to coalition (C4–C7). Careful task scheduling on Bahurupi architecture [15]

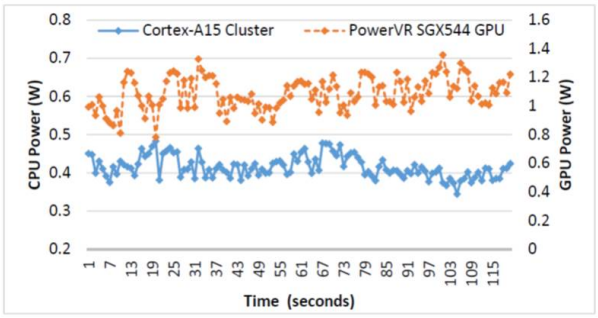

Fig. 5: CPU-GPU power behavior for mobile 3D games

yields speedup ranging from 10% to 62% compared to static homogeneous and heterogeneous multi-cores across a large range of task sets.

III. Functionally Heterogeneous Multi-Core

As mentioned earlier, a large class of heterogeneous multi-cores comprise of cores with different functionality. This is fairly common in the embedded space where a multiprocessor system-on-chip (MPSoC) consists of general-purpose processor cores, GPU, DSP, and various hardware accelerators (e.g., video encoder/decoder). The heterogeneity is introduced here to meet the performance demand under stringent power budget. For example, 3G mobile phone receiver requires 35–40 giga operations per second (GOPS) at 1W budget, which is impossible to achieve without custom designed ASIC accelerator [16]. Similarly, embedded GPUs are ubiquitous today in mobile platforms to enable not only mobile 3D gaming but also general-purpose computing on GPU for data-parallel (DLP) compute-intensive tasks such as voice recognition, speech processing, image processing, gesture recognition, and so on. Figure 1 shows a simplified block diagram of Samsung Exynos SoC consisting of ARM cores and PowerVR GPU core.

Traditionally, the power management of MPSoCs with mostly hardware accelerators apart from the programmable CPU is quite straightforward. The focus is only on managing power of the CPU while employing power/clock gating for the accelerators. As MPSoCs introduce more programmable cores (CPU, GPU, DSP), it is imperative to devise systematic runtime power management strategies for these cores. For example, Figure 5 shows the power behavior of the CPU (Cortex-A15 cluster only) and the GPU on the Exynos 5 Octa MPSoC running a popular Android game Asphalt 7: Heat over 2-minute lifetime. Clearly, both the CPU and the GPU contribute equally to the power consumption. Unfortunately, current power management strategies, for example for Linux in Android, are quite naive. Moreover, there is absolutely no co-ordination between the OS-level CPU power manager and device-driver level GPU power manager. This leads to wasted energy and poor performance specially when the chip power exceeds the TDP budget. We show that an integrated CPU-GPU power management strategy [17] can achieve similar to or better performance than current Andriod system while saving substantial energy.

IV. Conclusions

We have presented heterogeneous multi-core architectures as a promising approach towards energy-efficient computing. We have discussed the challenges and the opportunities offered by such architecture and the crucial runtime layer that is required to exploit heterogeneity.

Acknowledgment

This work was partially supported by CSR research funding and Singapore Ministry of Education Academic Research Fund Tier 2 MOE2012-T2-1-115.

References

[1] G. E. Moore *et al.*, "Cramming more components onto integrated circuits," 1965.

[2] R. H. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, and A. R. LeBlanc, "Design of ion-implanted MOSFET's with very small physical dimensions," *Solid-State Circuits, IEEE Journal of*, vol. 9, no. 5, pp. 256–268, 1974.

[3] H. Esmaeilzadeh, E. Blem, R. St Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Computer Architecture (ISCA), 2011 38th Annual International Symposium on*. IEEE, 2011, pp. 365–376.

[4] R. Kumar, K. I. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen, "Single-ISA heterogeneous multi-core architectures: The potential for processor power reduction," in *MICRO*, 2003, pp. 81–92.

[5] A. Peter Greenhalgh, "Big.LITTLE Processing with ARM Cortex-A15 & Cortex-A7," 2011.

[6] nVidia, "Variable SMP A Multi-Core CPU Architecture for Low Power and High Performance," 2011.

[7] Ineda Systems, "Hierarchical Computing," 2014. [Online]. Available: http://inedasystems.com/hierarchical-computing.html

[8] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the April 18-20, 1967, spring joint computer conference*. ACM, 1967, pp. 483–485.

[9] M. D. Hill and M. R. Marty, "Amdahl's law in the multicore era." *IEEE Computer*, vol. 41, no. 7, pp. 33–38, 2008.

[10] M. Pricopi, T. S. Muthukaruppan, V. Venkataramani, T. Mitra, and S. Vishin, "Power-performance modeling on asymmetric multi-cores," in *Compilers, Architecture and Synthesis for Embedded Systems (CASES), 2013 International Conference on*, 2013, pp. 1–10.

[11] T. S. Muthukaruppan, M. Pricopi, V. Venkataramani, T. Mitra, and S. Vishin, "Hierarchical power management for asymmetric multi-core in dark silicon era," in *Proceedings of the 50th Annual Design Automation Conference*. ACM, 2013, p. 174.

[12] T. Somu Muthukaruppan, A. Pathania, and T. Mitra, "Price theory based power management for heterogeneous multi-cores," in *Proceedings of the 19th international conference on Architectural support for programming languages and operating systems*. ACM, 2014, pp. 161–176.

[13] M. Guevara, B. Lubin, and B. C. Lee, "Navigating heterogeneous processors with market mechanisms," in *High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on*. IEEE, 2013, pp. 95–106.

[14] M. Pricopi and T. Mitra, "Bahurupi: A polymorphic heterogeneous multi-core architecture," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 8, no. 4, p. 22, 2012.

[15] ——, "Task scheduling on adaptive multi-core," *IEEE Transactions on Computers*, vol. 63, no. 10, pp. 2590–2603, 2014.

[16] W. J. Dally, J. D. Balfour, D. Black-Schaffer, J. Chen, R. C. Harting, V. Parikh, J. Park, and D. Sheffield, "Efficient embedded computing." *IEEE Computer*, vol. 41, no. 7, pp. 27–32, 2008.

[17] A. Pathania, Q. Jiao, A. Prakash, and T. Mitra, "Integrated CPU-GPU power management for 3D mobile games," in *Proceedings of the The 51st Annual Design Automation Conference on Design Automation Conference*. ACM, 2014, pp. 1–6.