# Extending the Dynamic Power Range of Client Devices using Heterogeneous Processors

Vishal Gupta*, Paul Brett†, Scott Hahn†, David Koufaty†, Mishali Naik‡, Paolo Narvaez‡,
Abirami Prabhakaran‡, Dheeraj Reddy†, Karsten Schwan*, Ganapati Srinivasa‡
*Georgia Institute of Technology, Atlanta, GA
†Intel Labs, Hillsboro, OR
‡Intel Corporation, Hillsboro, OR

***Abstract*—The ubiquity of handhelds is causing an unprecedented increase in the range of performance demands imposed on mobile platforms, and at the same time, battery life and energy efficiency remain critical concerns. Yet modern processors are typically designed to meet only one, not both, of these two conflicting goals: to offer high performance vs. provide power savings. This work explores an approach in which heterogeneous processors, i.e., a mix of different cores, are used to extend the dynamic power/performance range of client devices. Unlike previous work addressing server systems, we focus on the client workloads typically seen in modern end-user devices. Further, we evaluate the importance of taking into account 'uncore' power in total SoC power consumption, with results that indicate the need for additional uncore power scalability when seeking to extend a platform's dynamic power range. Experimental evaluations based on characterization of several client applications and usage scenarios seen on mobile devices use a unique experimental testbed comprised of heterogeneous cores that strongly differ in power/performance and with a shared uncore component.**

## I. Introduction

Mobile devices have emerged as a dominant computing platform for end users. Since their battery capacities are severely restricted due to constraints on size and weight, energy efficiency is critical to their usability. Desired long battery life, however, is challenged by end user demands for high performance for compute-intensive tasks like gaming and media-rich interactions. Chip vendors' adoption of multi-core processor architectures to deliver increased levels of performance is further increasing the difficulties of providing long battery life.

Heterogeneous chip multi-processors (CMPs), consisting of cores with different power/performance characteristics (see Figure 1), have been proposed as an energy-efficient alternative to symmetric multicores (SMPs) [1], [3], [4], [6]. Heterogeneous processors enable different applications to be executed on the core that is most appropriate. For example, applications that are I/O heavy or that do not produce a result that is time critical to the user can be executed on low power small cores, while compute-intensive threads requiring significant amounts of processing or applications with their output visible to the user such as browsing can be allocated to high performance big cores. The promise, then, is that by judicious use of heterogeneity, one can extend the *dynamic power range* of client devices, to meet both the high-performance and the low-power demands of client devices.

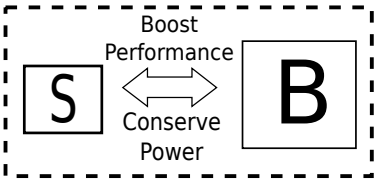



Fig. 1. A heterogeneous processor consisting of high-performance big and low-powered small cores can provide both high-performance and power-save modes.

This paper investigates the opportunities and limitations in using heterogeneous multicore processors to provide a wider dynamic power range for client devices, i.e., to enable both high-performance and power-savings modes while maximizing energy efficiency. Unlike previous work on heterogeneous processors that focused on server workloads, we investigate typical client workloads seen on modern end-user devices. This involves characterizing several client workloads and describing different ways in which they can exploit heterogeneity. It also includes an assessment of the *uncore* subsystem's contribution to total SoC (system-on-chip) power consumption and its implications on heterogeneous processor design.

Experimental evaluation uses a unique, experimental heterogeneous multicore platform comprised of both high and low power cores operating in a shared coherence domain. For this platform, we analyze the behavior of several client applications under two configurations: with a fixed vs. a scalable uncore, the latter allowing uncore power to scale along with core power. Results demonstrate that heterogeneous core architectures can provide significant performance improvements while also lowering energy consumption for a diverse set of applications when compared to homogeneous processor configurations. They also highlight the need for a scalable uncore in order to fully realize the potential gains obtained from the use of heterogeneity.

## II. Dynamic Power Range

Client applications exhibit highly diverse behavior in their processor usage and performance requirements. Compute-intensive, user-facing applications require a high performance mode to provide richer user experience. Background tasks or naturally low-performance applications should be run in ways that maximize battery life. To provide extended battery life

and at the same time, meet the rapidly increasing demands of high performance mobile use cases, a client device must offer a wide *dynamic power range* – it must be able to operate both in high-performance and in power-savings modes.

Modern processors are typically designed to satisfy only one of these two conflicting requirements. Current low-power cores (e.g., Intel's Atom processor) are energy efficient, but their performance is limited. On the other hand, more powerful big cores like Intel Core® processors provide high performance, but at the cost of higher levels of power consumption. The technological reasons for this is the fact that the power consumption of a processor core consists of static (leakage) power and dynamic (switching) power. During high activity periods, the total power consumption of the device is dominated by dynamic power consumption, while during low activity periods, leakage power becomes a significant component of the total power consumption. Current high performance cores are built from transistors on fast process technologies that have high leakage power and very fast switching times [1]. Such big cores, therefore, consume high leakage power under idle or near-idle conditions, but can provide high performance without significant increases in dynamic power, as shown in Figure 2. Conversely, low power small cores are built from low power process technologies with low leakage power but slower switching times [1]. Such processors consume small amounts of leakage power, but significantly increase dynamic power consumption to provide a high-performance mode (see Figure 2).

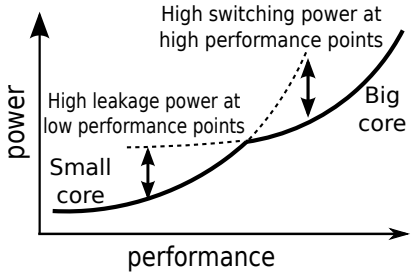



Fig. 2. Big cores are less efficient at low activity points, while small cores are less efficient at high activity points. Using a heterogeneous processor provides a wide dynamic power range.

The intuitive outcome is that by using both types of cores, a single platform can be optimized for both high performance and low power consumption. The objective of such a system would be to always use its most efficient cores for the tasks at hand (shown by the solid line in Figure 2). Such a heterogeneous platform exhibits a higher power-performance range than individual big or small cores. This paper explores whether and to what extent the hardware-based arguments for heterogeneity stated above lead to realistically achievable client devices with wide power ranges.

## III. Client Workloads

To assess the viability of using heterogeneity to widen client device power ranges, it is useful to refer to prior server-centric research on heterogeneous processors [2], [5], [7], [10], but such server-centric investigations do not directly address the needs and processor usage models seen on typical client devices. This section presents representative and typical client workloads, like browsing, gaming, video playback, etc., and it formulates metrics for the performance of client applications that differ from those used for server workloads. Table I provides a summary of the applications used in our analysis, along with relevant performance metrics. Concrete usage scenarios under which such workloads can exploit processor heterogeneity are described in Section V.

Our application suite consists of a diverse set of applications, which we briefly summarize below:

| Workload | Description | Metric |
|---|---|---|
| browse | Web-page rendering | Load time |
| palbum | Web photo-album | Load time |
| youtube | Web-based video playback | FPS |
| strike | Web-based 2D game | FPS |
| mplayer | Accelerated video playback | FPS |
| openarena | 3D first-person-shooter game | FPS |
| lightsmark | 3D graphics rendering | FPS |
| filescan | File system scan | Time |
| 7zip | File archiver | Time |
| gmagick | Image editor | Time |
| x264 | Media encoder | Time |
| eclipse | Development environment | Time |

TABLE I
Client Workload Summary

- browse: web-browsing is the most common usage of client devices. This workload fetches a set of web pages from a web server and renders them periodically. We introduce a sleep interval of 5s between every two page-loads to emulate user's think time.
- palbum: this is a web-based photo album application that flips through a set of photographs at a 0.5s interval.
- youtube: video playback plays a streaming video inside the browser for 120s.
- strike: online gaming is an increasingly dominant use case of client platforms. A demo of a web based 2D game is played for 120s to evaluate this usage scenario.
- mplayer: a H/W accelerated version of mplayer plays an HD movie clip.
- openarena: plays a benchmark demo from a 3D first-person-shooter game (OpenArena).
- lightsmark: this benchmark renders scenes from a 3D game and measures graphics performance.
- filescan: to evaluate the behavior of I/O intensive applications, this workload scans through the Linux source tree.
- 7zip: 7zip is a popular client application used for archive creation. We use a parallelized version of 7zip to compress a text file using LZMA compression.
- gmagick: an OpenMP version of the GraphicsMagick image editing application is used to resize a set of images.
- x264: x264 media encoder is used to encode a media file.
- eclipse: this Java based multi-threaded benchmark runs non-GUI performance tests for the Eclipse IDE.

## IV. Beyond Core: Uncore

The dynamic power range offered by a platform consisting of heterogeneous cores can be strongly affected by the *uncore* subsystem present on modern multicore processors. This subsystem consists of components like the last-level-cache (LLC), integrated memory controller (IMC) etc. With growing cache sizes to satisfy the needs of all of the cores in a multicore processor, increasing complexity of the interconnection network, various core power optimizations (e.g, idle states), and the integration of SoC components on CPU die, the uncore is increasingly becoming a major power component in total SoC power [8].

Figure 3 illustrates the contribution of uncore power to the energy consumption of an application executing on heterogeneous cores. A big core running an application finishes its execution faster and enters a low-power idle state. The same application when executed on a small core takes longer ($t_{small}$) to finish, which also keeps the uncore active for a longer period of time. If uncore power is substantial in comparison to core power, then the energy gains from running on a small core are strongly affected by the uncore power. For such a system, energy-efficiency gains from small core execution may be offset by the increase in uncore energy consumption due to longer execution time. This observation is in line with prior work that highlights the tradeoff between CPU and system-level power reduction in the context of frequency scaling [9].

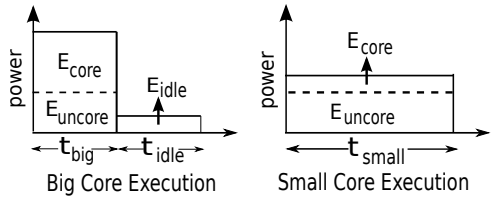



Fig. 3. Effect of uncore power on the dynamic power range of heterogeneous cores.

To account for uncore power and understand its impact on the dynamic power range of heterogeneous platforms, we consider two different uncore configurations in our analysis, namely, fixed uncore and scalable uncore.

*1) Fixed Uncore:* The fixed uncore configuration uses the same uncore subsystem when executing on either big or small cores. The big core and small core energy consumption for this scenario can be modeled using Equations 1 and 2, respectively. Here, $E$ refers to the energy consumed, $t$ denotes execution time, and $P_{core}$ and $P_{uncore}$ represent core and uncore power, respectively. $P_{idle}$ is the idle platform power and $t_{idle}$ is the corresponding time. We use a single value of $P_{uncore}$ in both of the equations for this case.

$$E_{big} = t_{big} * (P_{core}^{big} + P_{uncore}) + P_{idle} * t_{idle} \quad (1)$$

$$E_{small} = t_{small} * (P_{core}^{small} + P_{uncore}) \quad (2)$$

*2) Scalable Uncore:* The scalable uncore scenario models an uncore where certain uncore components are turned off or powered down as we move to the small core. For example, fewer memory channels, memory controllers, or a smaller cache can be used with a slow small core that imposes smaller resource requirement on the cache and memory subsystem. Hence, in this case, the uncore power scales down along with core power when a workload moves to a different core. Equations 3 and 4 model this scenario by using different uncore power values for big vs. small core execution ($P_{uncore}^{big}$ and $P_{uncore}^{small}$).

$$E_{big} = T_{big} * (P_{core}^{big} + P_{uncore}^{big}) + P_{idle} * T_{idle} \quad (3)$$

$$E_{small} = T_{small} * (P_{core}^{small} + P_{uncore}^{small}) \quad (4)$$

An analysis of these two uncore configurations provides a comparative perspective on the potential benefits of heterogeneous CMPs with a wholistic SoC-wide view and motivates a co-design of various SoC components.

## V. Usage Scenarios

Client device usage patterns along with the diverse nature of client workloads motivate the need for a wide dynamic power range.

### A. Platform Usage Scenarios

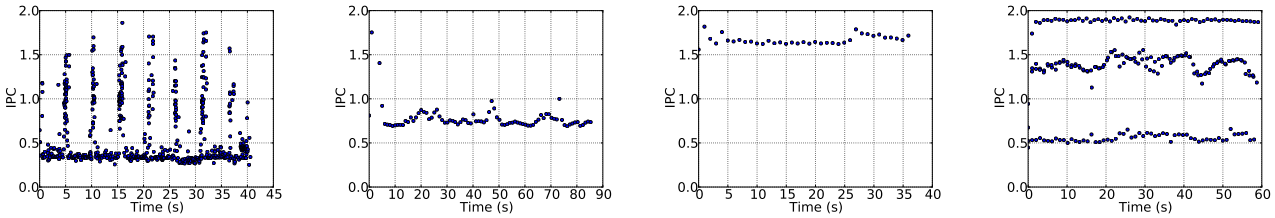
*1) Mobility Constraints:* Mobility is an essential part of client systems. Such devices may either be powered via wall-power or battery. Wall-power usage does not impose energy constraints, so that big cores can provide desired levels of maximum performance. During battery driven operation, however, a user may be willing to accept lower performance at the benefit of higher battery life. Low-powered energy-efficient small cores may be more suitable under such conditions.

*2) Thermal Constraints:* Client devices like cell phones and tablets rely on natural cooling. Therefore, these devices are quite sensitive to platform thermal constraints that impose limits on the extent to which it is possible to use power-hungry big cores for sustained periods of time. A small core can be used for moving the execution away from a big core when thermal constraints are violated.

### B. Workload Usage Scenarios

Users perform a wide variety of tasks on mobile devices which demands platforms that are able to meet their dynamic needs and maintain high levels of efficiency. This section categorizes client applications based on their behavior and then, discusses opportunities for exploiting heterogeneous cores.

*1) Intermittent Workloads:* Client devices like cellphones and tablets are typically powered-on for long periods of time, but often perform their heavy computing work in short bursts. Web-browsing is an example of such usage, and workloads browse and palbum in Table I belong to this category. A timeline trace of IPC (instructions-per-cycle) for the browse




(a) Intermittent (browse) (b) Sustained-low (openarena) (c) Sustained-high (x264) (d) Multi-threaded (youtube)

Fig. 4. Diverse Client Workload Profiles (IPC vs. Time)

workload is shown in Figure 4(a). Idle periods are marked by low IPC periods, while page loads correspond to spikes in the graph. Since page loads generate high IPC activity, a big core can be used for rendering the pages and improving page-load performance, while resorting to a small core during low activity periods to conserve power.

*2) Sustained Workloads:* Sustained workloads differ from intermittent workloads in that their behavior is uniform over a longer duration. They can be further classified into two sub-categories: sustained-high and sustained-low.

*Sustained-low:* client applications like gaming and media playback typically run for a long duration (a few minutes to hours). Moreover, the wide adoption of accelerators in all types of client devices allows these applications to offload significant portion of their computation to accelerators. Therefore, the performance of such applications is less dependent on the CPU. Figure 4(b) shows the IPC trace of the openarena gaming benchmark. As the observed IPC is low for the application, it can be run on a small core without significant degradation in performance and at lower power.

*Sustained-high:* mobile devices are also used for compute-intensive tasks, which include media encoding, photo/video editing, etc. These applications typically have a high IPC (e.g., see x264 encoder in Figure 4(c)), and their performance scales well on a big core. This makes big cores suitable for these applications when they require high performance, e.g., when they are user-facing, while a small core may provide higher energy-efficiency when they run in background mode (e.g, virus-scan).

*3) Multi-threaded Workloads:* With increasing number of cores on mobile devices, parallelization of client applications is key to further performance enhancement. Such multi-threaded applications also present opportunities for exploiting heterogeneity. 7zip, gmagick, x264, and eclipse are examples of parallel applications. The youtube workload also uses multiple threads for audio, video decoding, and rendering for instance which differ significantly in their IPC as shown in Figure 4(d). Due to the variation in the behavior and needs of these threads, their performance will be affected by how they are mapped to different heterogeneous cores, which can be leveraged by task mapping and scheduling methods.

In summary, the platform and workload usage scenarios described above create a case for using heterogeneity to create a wide dynamic power range for future mobile devices. The next section validates this point with experimental results obtained with the aforementioned client applications on a representative heterogeneous platform.

## VI. Evaluation

### A. Experimental Setup

Our experimental platform consists of a quad-core Intel i7-2600 client processor. To create heterogeneity, we use a proprietary Intel tool to emulate the performance of a low-power core for a subset of the CPU cores. A block diagram of the platform configuration is shown in Figure 5. The i7-2600 processor also contains an on-die graphics processor that is used to accelerate graphics workloads. All of the CPU cores operate at a frequency of 2.6GHz and share a last level cache (LLC) of size 8MB. All the workloads are run using Linux kernel 3.0 and automated using scripts.

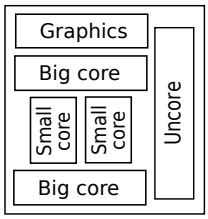
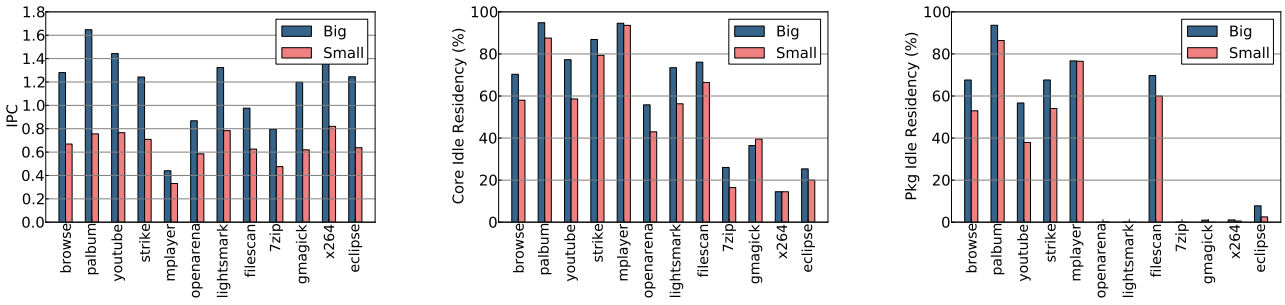



Fig. 5. Experimental Heterogeneous Platform

### B. Methodology

Experimental evaluation and analysis are carried out as the multiple steps summarized below.

- Each workload is first evaluated on a system configured to use only big cores. Multi-threaded applications are configured for a one to one mapping of threads to big cores.
- Next, the same workloads are run using only small cores.
- The metrics collected include: application performance, IPC, and various core and package C-state residencies.
- With the help of data collected in previous steps and the power models described in Section VI-C, we calculate the performance improvement provided by big over small cores, and the energy savings that can be obtained by using small vs. big cores.




(a) Instructions per cycle (b) Core Idle Residency (c) Package Idle Residency

Fig. 6. A comparison of the behavior of several client workloads on big vs. small cores

- The analysis is performed for two different uncore power configurations, fixed and scalable, using the models described in Section IV.

The analysis currently uses big or small cores for the entire execution of the application. In practice, an application can dynamically switch between different types of cores and achieve higher gains, but the implementation and evaluation of a dynamic scheduling algorithm for client devices remains part of our future work. The analysis also assumes that workload performance is not affected by uncore scaling.

### C. Power Model

The power models incorporate both core and uncore power to obtain total CPU power consumption.

$$P_{total} = P_{core} + P_{uncore} \tag{5}$$

The average power consumption of a CPU core core can be modeled using the following equations:

$$P_{core} = C_{active} * P^{core}_{active} + C_{idle} * P^{core}_{idle} \tag{6}$$

$$P^{core}_{active} = C_{dyn} * V^2 * F \tag{7}$$

Here, $C_{active}$ denote the percentage residency the core spends in active $C0$ state, while $C_{idle}$ is the residency in higher idle C states. $P^{core}_{active}$ is the power consumption for each of the big and small cores in the active state, while $P_{idle}$ is the idle core power. $C_{dyn}$ is the dynamic capacitance, V denotes the operating voltage, and F represents the switching frequency. Core $C_{dyn}$ is modeled as a function of IPC, as shown and validated by other researchers [11], in Equations 8 and 9.

$$C_{big} = 0.499 * ipc_{big} + 0.841 \tag{8}$$

$$C_{small} = 0.472 * ipc_{small} + 0.176 \tag{9}$$

Similarly, uncore power can be modeled as shown in Equation 10:

$$P_{uncore} = PC_{active} * P^{uncore}_{active} + PC_{idle} * P^{uncore}_{idle} \tag{10}$$

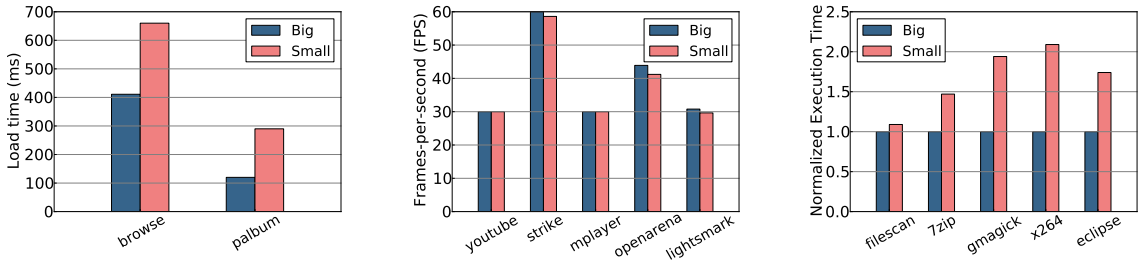
where $PC_{active}$ and $PC_{idle}$ are package-active and package-idle residencies, respectively, and $P^{uncore}_{active}$ and $P^{uncore}_{idle}$ are the corresponding power values. The current power model ignores the variation in uncore power due to LLC access activity.

The analysis uses a value of 0.9V for the voltage (V), and frequency (F) is kept at 2.6GHz. Core and uncore idle power are assumed to be 0.1W, while a value of 2W is used for the active uncore power ($P^{uncore}_{active}$) in case of a fixed uncore. This power is assumed to scale down to 1W for a scalable uncore. A 2W power component is added to workloads that exercise the on-die graphics processor.

### D. Experimental Results

The results shown in Figure 6 provide a comparison of application behavior on heterogeneous cores. Specifically, they compare average IPC (instructions-per-cycle), core-idle state residency, and package-idle state residency for all of the workloads in Table I for big and small core execution. As evident from Figure 6(a), most of the applications observe a significant decrease in their IPC when running on the small core as compared to the big core. This reduction in IPC results in the small core being active for longer durations, thereby causing a decrease in core and package idle residency (see Figures 6(b) and 6(c)). Further, many applications are seen to have almost negligible package idle residency. These applications either heavily use the graphics processor (e.g., openarena, lightsmark), or they always keep one of the CPU cores busy (e.g., 7zip, gmagick, x264), and thus do not allow the uncore to enter into an idle state.

The results shown in Figure 7 evaluate the impact on application performance of using heterogeneous processors. All of the applications in the figure are categorized into three different graphs, depending on the performance metric in Table I. Figure 7(a) compares the average load-time for the browse and palbum workloads. We see that the page-load latency is significantly decreased for these applications when using a big core. For example, the average page-load time for browse is decreased from 660ms to 441ms on the big core. Thus, a big core provides a notable performance boost for such bursty applications. Figure 7(b) shows the frames-per-second (FPS) metric for various graphics and media applications. These applications show only minor performance degradation on a small core, at levels not perceivable to end users. Therefore, they can be run on a small core, with only minor performance loss and a decrease in energy consumption (discussed further below). The last graph (see Figure 7(c)) compares the




(a) Load time (ms) (b) Frames-per-second (FPS) (c) Normalized Execution Time

Fig. 7. Application performance comparison on big and small cores

normalized execution time for various applications. filescan being an I/O intensive workload observes a small degradation (9.1%) in performance on the small core, while other CPU-bound applications show a significant increase in execution time with the small core (highest 109.04% for x264).

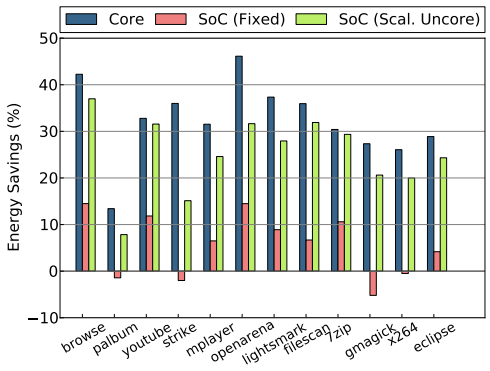



Fig. 8. Energy savings of small core execution over big cores for three configurations: core-only savings, SoC-wide savings with a fixed uncore, and SoC-wide savings with a scalable uncore.

Energy savings results based on our power models are shown in Figure 8. The figure shows energy savings for three scenarios, i.e., core-only savings, SoC-wide savings using a fixed uncore, and SoC-wide savings assuming a scalable uncore. As seen in the figure, all of the applications show significant savings on a small core when only considering the energy consumption of the CPU cores. The palbum application has the lowest savings of 13.4%, while openarena has the largest savings of 46.1%. However, these savings are strongly affected when the power consumption of the uncore is taken into account. Some applications even exhibit negative energy savings for the fixed uncore. On the other hand, when a scalable uncore is used, these savings increase and become comparable to core-only energy savings. These results motivate the need for a scalable uncore design when seeking to obtain large gains from heterogeneous multicore processors.

## VII. Conclusions & Future Work

In summary, this paper investigates the use of heterogeneous multicore processors in order to provide a wide dynamic power range to client devices. Using a diverse mix of client applications and an experimental heterogeneous platform, we show that heterogeneous CMPs can be used to provide a superior solution for these client devices by enabling both high-performance and power-savings modes while also being energy-efficient. We also highlight the growing importance of uncore power in total SoC power consumption and the need for a scalable uncore design along with heterogeneous cores to completely realize the intended gains.

As part of future work, we are investigating client-centric energy-aware scheduling algorithms and heuristics, to dynamically schedule tasks on heterogeneous cores. Another interesting venue for research would be to investigate the ideal ratios between the number of big and small cores for different client systems.

## VIII. Acknowledgements

The authors would like to thank Eugene Gorbatov, Andrew Herdrich, Alon Naveh from Intel and Karthik Gururaj from UCLA for their help on this work.

## References

[1] Variable SMP: A multi-core cpu architecture for low power and high performance. White paper, Nvidia Corporation, 2011.
[2] S. Balakrishnan, R. Rajwar, M. Upton, and K. Lai. The Impact of Performance Asymmetry in Emerging Multicore Architectures. In *Proceedings of the 32nd ISCA*, Washington, DC, USA, 2005.
[3] P. Greenhalgh. Big.LITTLE Processing with ARM CortexTM-A15 & Cortex-A7. White paper, ARM, Sept 2011.
[4] M. D. Hill and M. R. Marty. Amdahl's Law in the Multicore Era. *Computer*, 41(7):33–38, July 2008.
[5] D. Koufaty, D. Reddy, and S. Hahn. Bias scheduling in heterogeneous multi-core architectures. In *Proceedings of the 5th EuroSys*, pages 125–138, New York, NY, USA, 2010. ACM.
[6] 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 *Proceedings of the 36th MICRO*, pages 81—-, Washington, DC, USA, 2003.
[7] T. Li, P. Brett, R. Knauerhase, D. Koufaty, D. Reddy, and S. Hahn. Operating system support for overlapping-ISA heterogeneous multi-core architectures. In *IEEE 16th HPCA*, pages 1–12, 2010.
[8] G. H. Loh. The cost of uncore in throughput-oriented many-core processors. In *In Proc. of Workshop on Architectures and Languages for Throughput Applications (ALTA)*, 2008.
[9] A. Miyoshi, C. Lefurgy, E. Van Hensbergen, R. Rajamony, and R. Rajkumar. Critical power slope: understanding the runtime effects of frequency scaling. In *Proceedings of the 16th ICS*, New York, NY, USA, 2002.
[10] J. C. Saez, M. Prieto, A. Fedorova, and S. Blagodurov. A comprehensive scheduler for asymmetric multicore systems. In *Eurosys '10*, pages 139–152, New York, NY, USA, 2010. ACM.
[11] V. Spiliopoulos, S. Kaxiras, and G. Keramidas. Green governors: A framework for continuously adaptive dvfs. In *Green Computing Conference (IGCC)*, July 2011.