Heterogeneous Multi-core Architectures: Optimizing Power and Performance

Naman Jain, Prashanth Suresh
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh PA 15213
{namanj, psuresh}@andrew.cmu.edu
Project URL: http://www.contrib.andrew.cmu.edu/~namanj/18-743/

***Abstract*—In order to satisfy the high-performance and low-power requirements for advanced embedded systems with greater flexibility, it is necessary to develop chip multiprocessors. However, homogeneous multi-core systems do not achieve higher potential per watt when compared to heterogeneous architectures, as serialized code sections can be accelerated without programmer effort. Therefore, our work aims to determine the optimal number of cores of different types to provision the CMP with, such that the area and power budgets are met and the application performance is maximized. We consider general-purpose multi-threaded applications to evaluate single ISA heterogeneous multi-core architectures as a way to reduce processor power dissipation by switching cores to achieve target performance level. We used Sniper multi-core simulator for performance modeling and McPAT for power modeling. For the 4-core system we modeled, 1 big-core and 3 small-cores provided the best power-performance trade-off for most of the benchmarks.**

I. Introduction

The increase in need for machines with higher performance, limited computational power and increase in complexity in the design of uniprocessor has been the driving force for increase in interest in design of multi-core architecture. A multi-core processor has multiple cores integrated on a single chip. They are mainly of two types, (i) a multi-core architecture where every core is just an image of the other, called homogeneous multicore, and (ii) when a set of cores may differ in area, performance, power dissipated etc, it is called heterogeneous multicore.

However, multi-core processor introduces a number of new challenges and metrics of interest. First, accurate models for the execution time of multi-threaded benchmark applications running on heterogeneous cores are not readily available, although these are essential for optimization. This is in contrast to the application specific-domain where the performance models are well defined, often using formal models of computation such as data-flow graphs. Second, typical multi-threaded applications, for example the applications from the SPLASH-2 [19] and PARSEC 2.1 [18] benchmark suites, can be executed with a variable degree of parallelism, i.e., a varying number of parallel threads, at run-time. The architectural synthesis algorithm must be aware of the optimal degree of parallelism and optimal run-time scheduling of threads to cores for each application. Also, from an empirical perspective, an important metric of interest is the performance benefit of heterogeneous versus homogeneous CMP architectures with increasing chip area.

The various design issues in multi-core architecture also include resource sharing, power consumption, performance, area of the cache, cache coherence etc. In order to harness the resources provided by a multicore architecture the application must show a certain level of parallelism. With additional cores, power consumption and heat dissipation become a concern and must be simulated before layout to determine the best floorplan which distributes heat across the chip, while being careful not to form any hot spots [8]. Distributed and shared caches on the chip must adhere to coherence protocols to make sure that when a core reads from memory it is reading the current piece of data and not a value that has been updated by a different core. This adds to the total power consumption as well.

To satisfy the high-performance and low-power requirements for advanced embedded systems with greater flexibility, it is necessary to develop parallel processing on chips by taking advantage of the advances being made in semiconductor integration [12]. Heterogeneous cores provide different power/performance tradeoff [3]. In order to benefit from heterogeneous multicore architectures, the scheduler needs to consider the power/performance asymmetry of heterogeneous multicore architectures when making a scheduling decision.

A program's hardware demands are governed by its inherent characteristics. For example, consider the instruction-dependency-distance distribution of a program (the number of instructions between the producer of a data and its consumer). As shown in Fig. 1 (a) reported by Chen et al [15], the SPEC benchmark *apsi* has a large percentage of instructions with long dependency distance, while *mcf* has a high percentage of instructions with short dependency distance [15]. These two opposite trends in dependency distance distribution indicate different amounts of instruction level parallelism (ILP) in these two programs, and hence different requirements of instruction issue width on the processor core. As shown in Fig. 1 (b), *apsi* demonstrates a near constant reduction rate in execution time as the instruction issue width is changed from 1 to 8. This is because the program has sufficient ILP, as indicated in the distribution, to keep up with the issue width scaling, and hence favors processor core with large issue width.

On the other hand, *mcf* has a significantly lower reduction rate in execution time, and the amplitude of the rate sharply goes down as the instruction issue width gets larger, which means the program is more suitable to run on a core with small issue width. Therefore, a program's inherent characteristics determine its hardware resource demands, and this is used to find optimal number of cores of different types.

The rest of the paper is organized as follows, Section

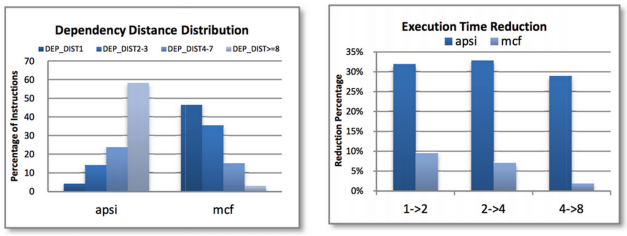


Fig. 1: Instruction dependency distance distribution and execution time reduction of *apsi* and *mcf*

II discusses about the related work, Section III describes the technical details, explaining why we need heterogeneous processors, our design and the scheduling policy we used. In Section IV we present our experimentation methodology while in Section V we present results of our experimentation and its analysis. In Section VI we tell about our future work, milestones, and Section VII concludes the work.

II. Related Work

Considerable amount of work has been done on power-related optimizations for processor design. These can be broadly classified into two categories: (1) work that uses gating for power management, and (2) work that uses voltage and frequency scaling of the processor core to reduce power [2].

Gating-based power optimizations [20], [21] provide the option to turn off (gate) portions of the processor core that are not useful to a workload. However, for all these techniques, gating benefits are limited by the granularity of structures that can be gated, the inability to change the overall size and complexity of the processor. Chip-wide voltage and frequency scaling reduces the parameters of the entire core [22]. While this reduces power significantly, the power reductions are uniform across the portions of the core that are performance critical for this workload, and the portions of the core that are not. Furthermore, voltage and frequency scaling is fundamentally limited by the process technology in which the processor is built. Heterogeneous multi-core designs address both these deficiencies.

Fine-grained voltage/frequency scaling techniques using multiple clock domains have been proposed recently which obviate some of the disadvantages of conventional scaling-based techniques. However, similar to gating-based approaches, the benefits are likely to be limited by static leakage inefficiencies as well as the number of voltage domains that can be supported on a chip.

Core switching to reduce power was introduced previously in [21]. Recently, it has also been used for reducing power density in a homogeneous multiple-core architecture through the use of frequent core switches to idle processors.

General-purpose, heterogeneous CMPs were first proposed by Kumar et al. [1] in the context of single-threaded applications running on CMPs in a multi-programmed fashion. In [1], the authors propose hill-climbing based solutions for heterogeneous CMP design, but only for multi-programmed workloads. Gupta et al. [16] take uncore which is a collection of components of a processor not in the core but essential for core performance such as last level cache (LLC), integrated memory controllers (IMC), on-chip interconnect (OCI), power control logic (PWR), etc.. They take into account the dynamic thread mapping for heterogeneous multicore systems. Flicker [17], proposed by Petrica et al., takes a different approach by dynamically scaling core resources to create adaptive and configurable heterogeneity in hardware.

Turakhia et al. [14] propose 'HaDeS' in the domain of synthesis for heterogeneous multicore systems by formulating a non-linear optimization problem and developing a heuristic to transform the problem into integer linear program in polynomial time. Their heterogeneous architecture explains how it addresses issues by (i) efficiently exploring the entire design space of heterogeneous CMP architectures using a novel iterative optimization strategy, (ii) using a more realistic performance model that is validated against detailed simulations on both homogeneous and heterogeneous CMPs, and (iii) optimizing over a set of benchmark applications.

Overall, having heterogeneous processor cores provides potentially greater power savings compared to previous approaches and greater flexibility and scalability of architecture design. As of now, there has not been much work on finding optimal number of cores of different types which this paper tries to solve such that performance needs as well as energy constraints are satisfied.

III. Technical Details

To attain high-performance and low-power requirements for advanced embedded systems it is necessary to develop parallel processing on chips by taking advantage of the advances being made in semiconductor integration. The subsequent sections explain the reason for choosing heterogeneous processors, proposes a design to get optimum value for power/performance and then describes the scheduling policy we used.

A. Why Heterogenous?

In an ideal homogeneous architecture, processing capability and power consumption are evenly distributed among its cores. Thus the decrease of computation time and the increase of system power are linear with the number of active cores. So the total energy efficiency (the Energy-Delay^2 product) is constant. However, these assumptions are not valid. First, as per Amdahl's law, performance improvement suffers from the law of diminishing return such that speedup may not scale linearly with the number of cores. Second, not all energy consumed by the system contributes to useful computation. Power consumption overhead is not linear with the number of active cores. Third, the achievable performance is limited by architectural factors such as I/O bandwidth and cache size.

In a heterogeneous multicore architecture a core may differ from the other cores in many ways. First, we focus on the performance aspect of multicores. Fig. 2 shows an example of a heterogeneous multi-core architecture on a single chip. The parallelizable portion of the code can be executed on smaller cores C2, C4, C5, whereas large critical sections can be executed on the large core C1. Building a performance oriented heterogeneous core is desired because many simple cores together provide high parallel performance while complex

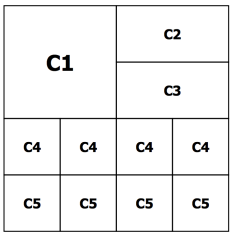


Fig. 2: Example Heterogeneous Design

cores help in providing high serial performance. Amdahl's Law is still relevant as we enter a heterogeneous multi-core computing era [9]. Amdahl's Law is a simple analytical model that helps developers to evaluate the actual speedup that can be achieved using a parallel program. If p is the number of processors and α is the parallelizable fraction of a program, then Amdahl's law states that speedup is given by

$$Speedup = \frac{1}{\frac{\alpha}{p} + (1 - \alpha)}$$

B. Design

A single processor will contain many small simple cores, and a few larger complex cores. Simple cores require less area as they will be scalar, in-order and might have a smaller cache and lower clock frequency. Complex cores will have superscalar design, higher issue width, larger ROB size and may be equipped with high-performance features such as out-of-order instruction scheduling, aggressive prefetching or a vector unit. However, they will be larger and will consume significantly more power. Heterogeneous architectures are motivated by their potential to achieve a higher performance per watt as compared to homogeneous systems because each application can run on a core that best suits its state. For example, a gaming app can run on a simple core when user is setting configurations or have paused a game, but the application should switch to complex core once the actual intensive gaming starts.

Our work primarily focuses upon optimizing power and performance with respect to area and number of cores of different types. A new performance model is designed for multi-threaded applications from the SPLASH-2 [19] and PARSEC-2.1 [18] benchmark suites. The performance model for applications is scheduled on both homogeneous and heterogeneous CMPs, and across a wide range of chip sizes and number of chips. We calculate the optimal number of cores of each type based on evaluation of multiple libraries of configuration within the area budget. We then feed the performance modeling parameters to McPAT which calculates the energy and power efficiency.

In a heterogeneous CMP, the execution time of each parallel thread is different. In the parallel phase of execution, threads typically synchronize on a barrier, i.e., all threads must finish execution before the application can proceed to the next phase. Therefore, the latency of a parallel phase is determined by the worst case execution latency across all parallel threads.

C. Scheduling

To take full advantage of heterogeneous CMPs, the system software must use the execution characteristics of each application to predict its future processing needs and then schedule it to a core that matches those needs if one is available [1], [2]. The predictions can minimize the performance loss to the system as a whole rather than that of a single application. Recent work has shown that effective schedulers [6] for heterogeneous architectures can be implemented and integrated with current commercial operating systems.

Lets assume the hardware consists of B big cores and S small cores. The scheduler we used selects B threads that have affinity to (all) big cores, while the other threads have affinity to the small cores. Periodically, or when threads on a big core stall/end, new threads are promoted to run on the big core(s). Each thread is pinned to a specific core and cores are handed out to new threads in round-robin fashion. If multiple threads share a core, they are time-shared with a configurable quantum.

Since we are interested in the highest performance that a processor can offer, we assume good static scheduling of threads to cores. Thus, the performance of four particular threads on four particular cores is the performance of the best static mapping. However, this actually represents, in some sense, a lower bound on performance.

IV. Experiment Setup

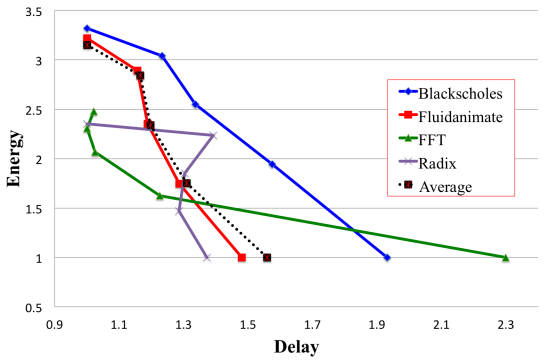
We used Sniper [10] for performance modeling. It is a modular platform for computer system architecture research, encompassing system-level architecture as well as processor microarchitecture. It also has easier interface for multiple heterogeneous core modeling. We initially wanted to use gem5 [4] for performance modeling, because of its precision, but it currently does not provide much flexibility for heterogeneous core modeling. McPAT [11] was used for power modeling as it has integrated power, area, and timing modeling framework for multicore architectures. All analysis is done for 45nm technology node and nominal 1.2V supply voltage.

The architectural design was tested using multi-threaded PARSEC-2.1 and SPLASH-2 [18], [19] benchmark suites. These benchmark suites focus on emerging workloads and has been designed to be representative of next-generation shared-memory programs for chip-multiprocessors.

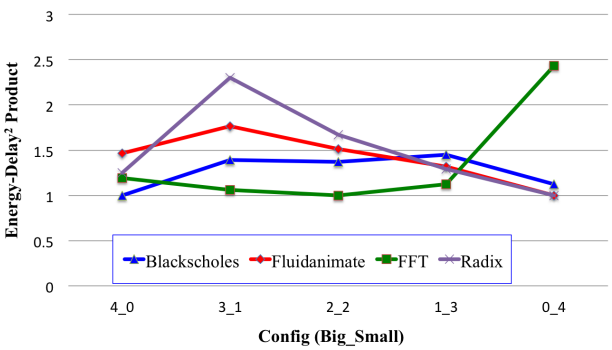
For the first phase of the work we used two types of cores, i.e. 'big' and 'small'. Table I describes the configuration we used for each of them. The area budget used was $180mm^2$. We also consider multi-threaded workloads and assume that each core runs at most one thread at a time - in other words, we experiment for single threaded execution, without simultaneous multithreading (SMT). Without loss of generality, the total number of threads is assumed to be n, identical to the total number of cores.

V. Experimental Results

We modeled all possible heterogeneous combinations for 4-core system and calculated the time-delay and energy consumption for each benchmark. The results are tabulated in Table II. Please note that we have denoted processor configuration as {num_bigs}_{num_smalls}, for example, a system



(a) Energy v/s Delay Plot



(b) Energy-Delay^2 Product

Fig. 3: Power-Performance Analysis

-

TABLE I: Configuration

Core Parameter	Big	Small
Dispatch width	4	1
L1-I/D Cache size	64 KB	32 KB
L2 Cache (private)	256 KB	128 KB
Frequency	2.66 GHz	2.66 GHz
ROB Window	256	32

with 3 big cores and 1 small core is referenced as 3_1 in the rest of the paper. The time delay reported is the maximum of all the processors as latency of a parallel phase is determined by the worst case execution latency across all parallel threads. Normalized Energy-Delay^2 is also reported and plotted in Fig. 3b to better analyze performance - energy trade-off.

Fig. 3a shows a plot between normalized-energy and normalized-delay, varying the configuration for each benchmark. We see that as the number of 'smalls' increase, energy reduces but delay increases. The most optimal design would reduce energy as well as delay.

Analysis

As expected, we see larger delays for systems with more number of 'small' cores and also a much higher energy for systems with more number of 'big' cores in Table II.

From Fig. 3b we observe that 4_0 does fairly well in terms of Energy-Delay^2 product, but at the cost of almost 3 times energy when compared to 0_4. Secondly for *fft* benchmark, we see that the product value is beyond the acceptable limits for 0_4. It is also interesting to note the characteristics of *radix* in Fig. 3a, that the configurations 2_2 and 3_1 lead to worse performance than 1_3. Thirdly, Fig. 3 suggest that config 3_1 fairs the worst and 1_3 does comparatively the best for these benchmarks, which is understandable, as generally only a single thread handles the critical section, or bottleneck phases.

Obstacle

We expected Energy-Delay^2 to be lower for heterogeneous architectures for most of the benchmarks but it was not the case. We attribute this to the absence of aggressive scheduling mechanism which is required to better reflect the advantages of heterogeneous designs. Therefore, in the later phase of the project we plan to implement an improved version of the scheduling policy.

TABLE II: Energy-Delay Values

Benchmark	Config	Delay (μs)	Energy (J)	Normalized Energy * Delay^2
blackscholes	**4_0**	20013435	4.15	1.00
	3_1	24681970	3.8	1.39
	2_2	26753118	3.19	1.37
	1_3	31517919	2.43	1.45
	0_4	38647773	1.25	1.12
fuildanimate	**4_0**	173002752	25.47	1.47
	3_1	200116222	22.86	1.76
	2_2	205544909	18.62	1.52
	1_3	222647873	13.8	1.32
	0_4	256194434	7.91	1.00
fft	**4_0**	11520095	1.86	1.19
	3_1	11277218	1.73	1.06
	2_2	11560415	1.55	1.00
	1_3	13826642	1.22	1.13
	0_4	25928783	0.75	2.43
radix	**4_0**	9856729	0.8	1.25
	3_1	13719557	0.76	2.30
	2_2	12844164	0.63	1.67
	1_3	12670812	0.5	1.29
	0_4	13522996	0.34	1.00

VI. Future Work

In the next phase of the project we plan to (i) increase number of cores, satisfying the area budget with different 'types' of cores, (ii) adopt aggressive scheduling, (iii) support larger configuration library. We also plan to decide configuration based on the benchmark, i.e. if benchmark A is best suited with 1_3 configuration and benchmark B is best suited with 2_2 configuration, the system would change configuration and provide best performance/energy depending on individual benchmarks, rather than a configuration based on average across all the benchmarks. We plan to do this by storing meta-data in the memory, instead of dynamically computing the configuration. Table III provides the milestone with the timeline for our project.

TABLE III: Milestone

Stage	Status/ Date	Progress
Initial	Done	Setup Snipersim and McPAT
Midway	Done	Complete heterogeneous design
75%	Done	Measure power and performance for single heterogeneous configuration
Final	Nov 16	Obtain optimal power/performance sweet-spot for user specified area and energy budget
Sky	Dec 2	Aggressive scheduling policy

VII. Conclusion

In this paper we focus on determining the optimal number of cores of different types, such that the area and power budgets are met and the application performance is maximized. We consider general-purpose multi-threaded applications to evaluate single ISA heterogeneous multicore architectures. For an architecture with 4-cores, we observed that configuration 1_3 provides the best performance-energy trade-off. Later, we also plan to propose a way to dynamically configure the system depending upon the benchmark. The configuration is decided offline so that no delays are faced during run-time and optimal performance/ energy is achieved.

VIII. Work Distribution

Naman: Setup Sniper, making modifications to the simulator, python scripting to run multiple simulations, writing report.
Prashanth: Analyzing related work, scripting to capture performance and power, plotting graphs, writing report.

References

[1] Kumar, Rakesh, Dean M. Tullsen, and Norman P. Jouppi. "Core architecture optimization for heterogeneous chip multiprocessors." Proceedings of the 15th international conference on Parallel architectures and compilation techniques. ACM, 2006.

[2] Kumar, Rakesh, et al. "Single-ISA heterogeneous multi-core architectures: The potential for processor power reduction." Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on. IEEE, 2003.

[3] Lukefahr, Andrew, et al. "Composite cores: Pushing heterogeneity into a core." Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture. IEEE Computer Society, 2012.

[4] Binkert, Nathan, et al. "The gem5 simulator." ACM SIGARCH Computer Architecture News 39.2 (2011): 1-7.

[5] Mutlu, Onur, "Asymmetry" 18-740 Lecture 2.1 Fall 2013

[6] Ghiasi, Soraya, Tom Keller, and Freeman Rawson. "Scheduling for heterogeneous processors in server systems." Proceedings of the 2nd conference on Computing frontiers. ACM, 2005.

[7] Tsoi, Kuen Hung, and Wayne Luk. "Power profiling and optimization for heterogeneous multi-core systems." ACM SIGARCH Computer Architecture News 39.4 (2011): 8-13.

[8] Hyari, Abeer. "A comparative study on heterogeneous and homogeneous multiprocessors." University of Jordan (2009).

[9] Marowka, Ami. "Extending Amdahl's Law for Heterogeneous Computing." Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on. IEEE, 2012.

[10] Carlson, Trevor E., Wim Heirman, and Lieven Eeckhout. "Sniper: exploring the level of abstraction for scalable and accurate parallel multi-core simulation." Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. ACM, 2011.

[11] Li, Sheng, et al. "McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures." Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on. IEEE, 2009.

[12] Uchiyama, Kunio, et al. Heterogeneous Multicore Processor Technologies for Embedded Systems. Springer, 2012.

[13] Liu, Guangshuo, Jinpyo Park, and Diana Marculescu. "Dynamic thread mapping for high-performance, power-efficient heterogeneous many-core systems."Computer Design (ICCD), 2013 IEEE 31st International Conference on. IEEE, 2013.

[14] Turakhia, Yatish, et al. "HaDeS: architectural synthesis for heterogeneous dark silicon chip multi-processors." Proceedings of the 50th Annual Design Automation Conference. ACM, 2013.

[15] Chen, Jian, and Lizy Kurian John. "Energy-aware application scheduling on a heterogeneous multi-core system." Workload Characterization, 2008. IISWC 2008. IEEE International Symposium on. IEEE, 2008.

[16] V. Gupta, P. Brett, D. Koufaty, and D. Reddy, The forgotten uncore: On the energy-efficiency of heterogeneous cores, in Proc. USENIX Annual Technical Conf, 2012.

[17] P. Petrica, A. Izraelevitz, D. Albonesi, and C. Shoemaker, Flicker: A Dynamically Adaptive Architecture for Power Limited Multicore Systems, in Proc. ISCA, 2013.

[18] Bienia, Christian, et al. "The PARSEC benchmark suite: Characterization and architectural implications." Proceedings of the 17th international conference on Parallel architectures and compilation techniques. ACM, 2008.

[19] Woo, Steven Cameron, et al. "The SPLASH-2 programs: Characterization and methodological considerations." ACM SIGARCH Computer Architecture News. Vol. 23. No. 2. ACM, 1995.

[20] Folegnani, Daniele, and Antonio Gonzlez. "Reducing power consumption of the issue logic." In Workshop on Complexity-Effective Design. 2000.

[21] Ghiasi, Soraya et al. Using IPC Variation in Workloads with Externally Specified Rates to Reduce Power Consumption. 2000.

[22] Govil, Kinshuk, Edwin Chan, and Hal Wasserman. "Comparing algorithm for dynamic speed-setting of a low-power CPU." Proceedings of the 1st annual international conference on Mobile computing and networking. ACM, 1995.

[23] Heo, Seongmoo, Kenneth Barr, and Krste Asanovic. "Reducing power density through activity migration." Low Power Electronics and Design, 2003. ISLPED'03. Proceedings of the 2003 International Symposium on. IEEE, 2003.

[24] Spiliopoulos, Vasileios, et al. "Introducing DVFS-management in a full-system simulator." Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS). IEEE, 2013.