Diversity: A Design Goal for Heterogeneous Processors

Citation for published version:
Tomusk, E, Dubach, C & O'Boyle, M 2016, 'Diversity: A Design Goal for Heterogeneous Processors', *IEEE Computer Architecture Letters*, vol. 15, no. 2, pp. 81 - 84. https://doi.org/10.1109/LCA.2015.2499739

Digital Object Identifier (DOI):
10.1109/LCA.2015.2499739

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
IEEE Computer Architecture Letters

Diversity: A Design Goal for Heterogeneous Processors

Erik Tomusk, Christophe Dubach, Michael O'Boyle
University of Edinburgh, United Kingdom

Abstract—A growing number of processors have CPU cores that implement the same instruction set architecture (ISA) using different microarchitectures. The underlying motivation for single-ISA heterogeneity is that a diverse set of cores can enable *runtime flexibility*. Modern processors are subject to strict power budgets, and heterogeneity provides the runtime scheduler with more latitude to decide the level of performance a program should have based on the amount of power that can be spent. We argue that selecting a diverse set of heterogeneous cores to enable flexible operation at runtime is a non-trivial problem due to diversity in program behavior. We further show that common evaluation methods lead to false conclusions about diversity. Finally, we suggest the KS statistical test as an evaluation metric. The KS test is the first step toward a heterogeneous design methodology that optimizes for runtime flexibility.

Index Terms—diversity, flexibility, heterogeneity, core selection, Kolmogorov-Smirnov test, metrics

1 Introduction

HETEROGENEOUS processors with two types of CPU cores have been available in mobile devices for some years [10], and a processor with three core types has recently been announced [8]. The benefits of these processors are obvious: Normal operation takes place on low-power cores to control temperature and prolong battery life. Demanding and time-critical programs can be run on higher-performance cores. If programs are scheduled correctly, the user will experience both good thermal behavior and good performance. All cores on these processors implement the same ISA (instruction set architecture), so no extra work is required from software developers to support heterogeneity.

Mobile devices must meet high user expectations for performance while operating under strict thermal limits and running off of small batteries. If a processor is *flexible*, then the operating system scheduler can run a given program at an appropriate power-performance point, taking into account the program's priority, thermal considerations, the charge left in the battery, etc. Traditionally, the flexibility to choose between high-performance and low-power operation has come from DVFS (dynamic voltage and frequency scaling; see, e.g., [3]). A heterogeneous processor with a diverse set of cores has the potential for greater flexibility and better energy efficiency [6]. Heterogeneity can be further augmented with DVFS. For simplicity, the power-performance points in our discussion come from heterogeneous cores without DVFS. However, the discussion remains valid if the points are derived from a combination of heterogeneous cores and DVFS levels.

Selecting CPU cores to enable flexibility is far from a simple problem, because each program interacts differently with each core's microarchitecture. We illustrate the intuition with figure 1 using two hypothetical benchmarks. For the examples in this paper, we assume that a processor designer has a benchmark suite that is representative of the programs that will be run on the processor. Each benchmark can be an entire program, a program phase, or a thread from a multi-threaded program. The examples use a trade-off between power and execution time. Power is a hard, physical limit, and time is important to the user. Figure 1 (top) shows eight cores, **A**-**H**, that the designer can select from. In this example, the designer has selected a diverse set of cores: **A**, **B**, **D**, and **H**. These cores are Pareto-optimal for benchmark 1—each core is at an optimal power-performance point, ranging from low power to high performance. The selection is flexible for benchmark 1, because the scheduler can run it at a wide range of performance levels.

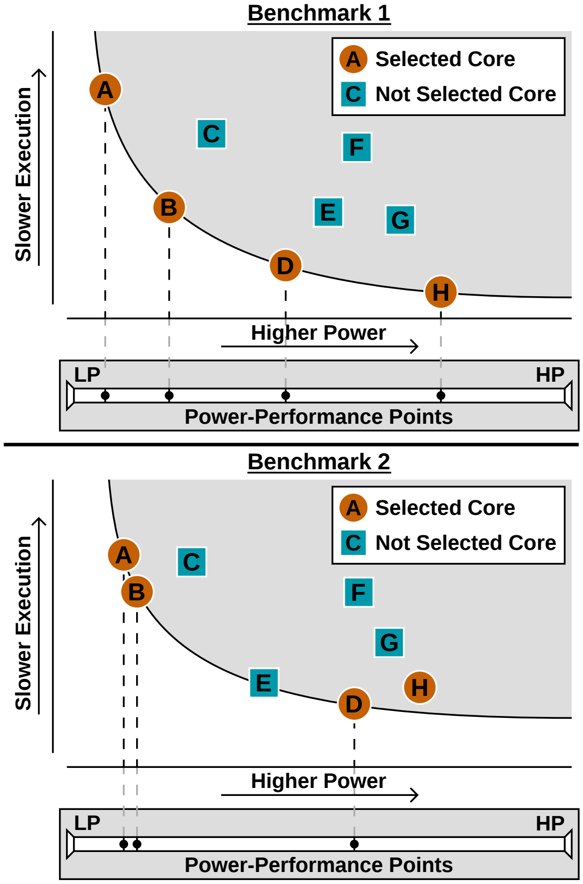

Fig. 1. (Top) Four selected cores, **A**, **B**, **D**, and **H** provide a range of power-performance points to benchmark 1, ranging from low power (**LP**) to high performance (**HP**). (Bottom) The same four cores are not as diverse for benchmark 2. **H** is not even Pareto-optimal. **E** is Pareto-optimal, but is not selected.

Figure 1 (bottom) shows the same cores for benchmark 2. For this benchmark, core **H** is no longer Pareto-optimal, since

Manuscript submitted: 10-Jul-2015. Manuscript accepted: 16-Oct-2015. Final manuscript received: 19-Oct-2015.

core **D** is both faster and consumes less power. Core **E** is Pareto-optimal, but has not been selected for the processor. Cores **A** and **B** have almost identical behavior. It can be seen that the same set of cores provides much less flexibility to the scheduler when running benchmark 2—benchmark 2 can be run at effectively two power-performance points (**A** or **B**, and **D**). In summary, a set of cores that is diverse for one benchmark is not necessarily diverse for another, because flexibility is dependent on the interaction between a core's microarchitecture and a program's behavior. To the best of our knowledge, we are the first to identify the variation in program behavior as a major roadblock to enabling runtime flexibility in heterogeneous processors.

In section 2, we will further clarify flexibility. Section 3 describes the difficulties of selecting a diverse set of cores. Section 4 demonstrates that quantifying diversity is a non-trivial problem. Section 5 suggests the KS statistical test as one method for evaluating diversity. The ability to evaluate diversity is the first step toward developing core selection methodologies that maximize runtime flexibility.

2 The Flexibility Goal

In the case of a homogeneous processor, the designer considers a number of different CPU core designs with different power-performance trade-offs, and chooses to implement one of them. In this case, the scheduler has access to only limited flexibility through DVFS. In contrast, in the case of a heterogeneous processor, the designer selects more than one power-performance point (more than one type of core) to implement. The designer leaves it to the scheduler to make the final choice about which CPU core is best, because the scheduler has more information about what is required of the CPU at a given time. This is the underlying goal of heterogeneity—to give the scheduler more choice (flexibility) at runtime using a diverse set of cores.

A crucial observation is that not all selections of Pareto-optimal cores are diverse. For example, assume a designer has access to six types of cores that are all power-performance Pareto-optimal, as shown in figure 2. Runtime flexibility is maximized if the designer implements all six, but this may not be possible. If the three "diverse" cores are implemented, then the selection is still flexible, as there are diverse power-performance points. If, instead, the "clustered" cores are implemented, then the selection is not flexible. The scheduler will only be able to choose a high-performance core; there is no low-power option.

This example shows that heterogeneity by itself does not guarantee flexibility; the selection of heterogeneous cores must be diverse. Section 3 will discuss why selecting a diverse set of cores is difficult, and section 4 will describe why a designer may be tempted to select a clustered set of cores.

3 The Selection Problem

The flexibility of a heterogeneous processor is maximized when it implements the most diverse set of cores possible—all Pareto-optimal cores. Since there can be hundreds of such core types, enabling flexibility requires the designer to select a small number of cores that preserve the diversity of the Pareto-optimal set. Intuitively, diversity means that the representative subset should have good spread, the cores should be uniformly spaced, and the selection should take diversity in program behavior into account. However, these three considerations are difficult to define objectively, and each poses non-trivial problems. We note that methods for sampling a Pareto frontier while preserving spread and uniformity exist for elitist genetic algorithms [11], but these methods use a larger number of points, can only select for one benchmark, and are therefore inappropriate for selecting cores.

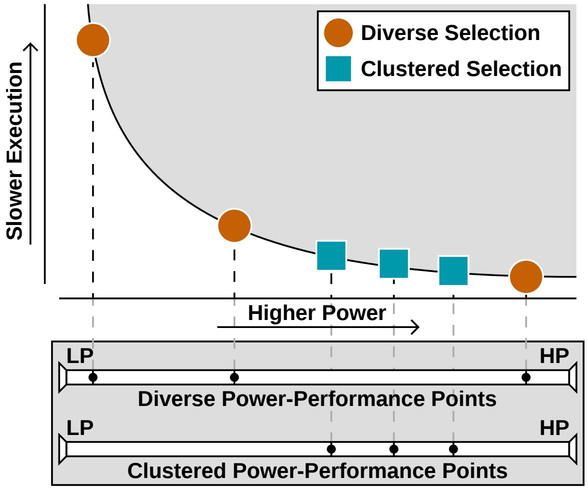

Fig. 2. Cores selected for flexibility (circles) implement diverse power-performance points. If cores have similar power-performance points (squares), then there is little flexibility available at runtime.

3.1 Spread

By *spread*, we simply mean how far the lowest-power and highest-performance cores are from each other (see the power-performance point scales in figures 1 and 2). To maximize diversity, the immediate intuition may be to maximize spread. However, this will not guarantee a good selection of cores. At the edges of the set of Pareto-optimal cores, tiny gains in one metric are made at the expense of large losses in the other. For example, re-architecting the fastest core to slow it down by 1% may lead to a 10% reduction in power, and re-architecting the lowest-power core to consume 1% more power may lead to a 10% increase in performance. This would reduce the spread of cores, but can make the cores more representative of the complete Pareto-optimal set. While having a broad spread of cores is necessary for diversity, a designer cannot rely on maximizing spread to maximize diversity.

3.2 Uniformity

A set of cores enables flexibility if the cores are uniformly distributed in the design space. Cores that have similar behavior, like the clustered set in figure 2, limit flexibility at runtime since the scheduler can only choose from a small range of power and performance levels. Similarly to spread, uniformity alone does not guarantee a diverse selection of cores. If spread is poor, then diversity will be poor regardless of uniformity. Even if spread is good, uniformity is affected by diversity in program behavior.

3.3 Program Diversity

The overarching difficulty of selecting heterogeneous cores is diversity in program behavior. Core selection is not simply about finding a representative sample for one Pareto frontier. Since each benchmark behaves differently, and since the set of Pareto-optimal cores is different for each benchmark, selecting cores requires sampling a Pareto frontier for every benchmark. The sampling must be done in such a way that a small number of cores capture the diversity seen across all Pareto frontiers.

Spread. Program diversity affects spread. For example, consider the problem of selecting a fast core for a heterogeneous processor using two benchmarks. One benchmark has a large working set, and the other has high ILP (instruction level parallelism). A large data cache will speed up the first benchmark, but will waste power for the second. A large instruction window will speed up the second benchmark, but will waste power for the first. It is not immediately obvious whether the designer should select a large data cache, a large instruction window, both, or if two different cores should be used. A core with a large data cache will not improve spread for the ILP benchmark, and a core with a large instruction window will not improve spread for the data-heavy benchmark. A core with both will waste power when running either benchmark, and will not be Pareto-optimal for either. Implementing two different fast cores may not be possible.

Uniformity. Achieving a uniform selection is similarly difficult, since cores that are evenly spaced for one benchmark may be clustered together for another. For example, if there are several cores with varying sizes of instruction windows, then for a benchmark with high ILP, the cores implement different power-performance points. In contrast, for a benchmark with low ILP, the cores have similar performance, but some will consume more power and will not be Pareto-optimal.

In summary, selecting a diverse set of cores is difficult, because the spread and uniformity of a the set vary from benchmark to benchmark. The problem is further compounded by the lack of methods for quantifying diversity, as the next section will show.

4 The Limitations of Summary Metrics

If microarchitectural diversity were easy to measure, then methods for improving diversity could be developed. In this section, we will discuss an evaluation methodology that uses a summary metric, and show why this seemingly obvious approach to achieving diversity cannot work in practice.

A recurring theme in microarchitecture research is that a CPU core should be energy efficient—regardless of its speed, the core should maximize the amount of useful work performed with the energy that is consumed (see, e.g., *energy-efficient architectures* [12]). Martin, et al. argue forcefully for the use of the ED^2 efficiency metric, as it captures the cubic relationship between power and speed in CMOS circuits [7]. ED^2, or energy-delay-squared product, takes into account the execution time and energy consumed by a program. For a compute-bound workload running on a core with DVFS, ED^2 will be constant across all DVFS levels. This can lead the designer of a heterogeneous processor to assume that ED^2 should also be constant for a diverse set of heterogeneous cores. One core might consume very little energy, but would be slow. Another core might be fast, but would consume large amounts of energy. Together, the cores would span a diverse set of power-performance points.

We argue that an ED^2 optimization approach is unreliable for a heterogeneous processor, as it is unlikely that different microarchitectures with different power-performance characteristics can actually be implemented with equally good ED^2. This has been shown quantitatively [1]; we offer an intuitive explanation with figure 3. Each black line in figure 3 is a constant ED^2 value. ED^2 is better toward the bottom and left of the graph. **A**, **B**, and **C** are cores. **B** corresponds to an in-order or simple out-of-order core. The high efficiency (work per unit of energy) of **B** makes it a good fit for battery-operated devices and throughput-oriented data centers. **C** corresponds to a wide, out-of-order core with aggressive prefetching and speculation. Cores like **C** are used for applications where latency is more important than efficiency (e.g., gaming). Finally, **A** is an extremely low-power core. Cores like **A** might be used when strict thermal limits are in place, and low-power operation is more important than speed or efficiency.

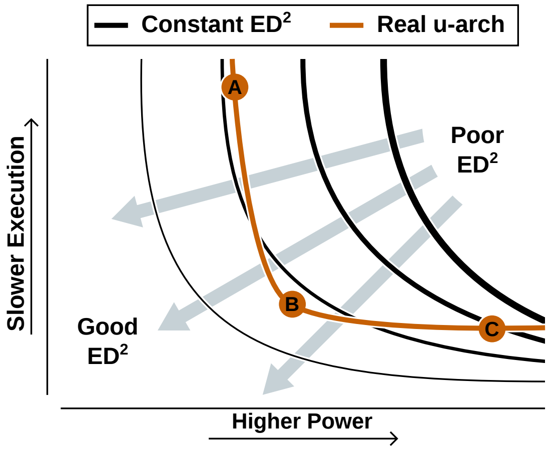

Fig. 3. The black lines show constant energy efficiency (ED^2) from low power to high performance. Ideally, low-power and high-performance cores would have equal efficiency, but this is difficult to do in practice.

If cores **A**, **B**, and **C** in figure 3 could be designed to have equal efficiency, then ED^2-optimization could be used to make at least a preliminary selection of diverse cores. However, the energy spent on aggressive speculation in **C** makes it inevitable that energy per instruction will be higher and ED^2 will be worse than for **B**. Similarly, the long execution time of **A** gives it a lower efficiency than **B**. As illustrated by figure 2, a processor with cores like **A**, **B**, and **C** is clearly more diverse than a processor with only cores like **B**. The average efficiency of a processor with only type **B** cores will, however, be better than that of a processor with **A**-, **B**-, and **C**-type cores. We can conclude that *efficiency cannot be used as a proxy for diversity*—if cores are selected only for efficiency, then all cores will be similar to **B**, and runtime flexibility will be minimal.

To summarize, cores selected for efficiency are not guaranteed to be diverse, and a diverse selection of cores will not necessarily have a good average efficiency. For this discussion, we have used the ubiquitous ED^2 metric, but the argument remains valid for all E^iD^j-type metrics. Adjusting the weights of the E and D terms simply shifts the point in the design space where the metric is minimized.

5 A Measure of Diversity: The KS Test

We have argued that a more diverse set of cores increases the flexibility that a scheduler has to control power and performance, and that flexibility is maximized when the scheduler can choose from all power-performance Pareto-optimal cores. Even if a perfect scheduler could be designed, implementing a processor with hundreds of different cores is currently infeasible. Maximizing flexibility is therefore a problem of finding a small subset of cores that capture the diversity of the complete Pareto-optimal set. We have argued that the major difficulty in selecting cores is ensuring that all relevant benchmarks have access to a broad spread of approximately uniformly distributed cores. One could develop search algorithms to maximize diversity, but this necessitates a quantitative metric of diversity. We have also argued that quantities like ED^2 do not provide

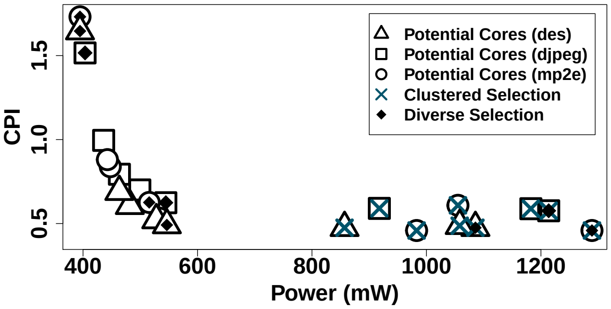

Fig. 4. Eight cores with increasing cache sizes and instruction windows, shown for the *des*, *djpeg* and *mp2e* benchmarks. Some selections of three cores enable flexibility; others do not.

TABLE 1
T_1 and ED^2 Values for Selections of Cores in Figure 4 (smaller is better)

Selection	Normalized Mean ED^2			T_1		
	des	*djpeg*	*mp2e*	*des*	*djpeg*	*mp2e*
All Cores	1.0	1.0	1.0	0.0	0.0	0.0
"Diverse"	1.9	1.6	1.6	0.24	0.21	0.33
"Clustered"	0.3	0.8	0.2	0.71	0.63	0.83

such a metric. Instead, we now propose measuring diversity with T_1, the two-sided, two-sample Kolmogorov-Smirnov test (KS test) [4, p.456]. T_1 is used to determine how likely it is that two sets of sampled data points come from different (unknown) distributions. If T_1 is close to 0.0, then it is very likely that the two sets of samples come from the same distribution.

We perform the KS test on a per-benchmark basis using power values. The first sample contains those cores from the complete set that are Pareto-optimal for the benchmark. The second sample contains the Pareto-optimal cores from the subset. Since the test is only applied to Pareto-optimal points, results are identical when using speed instead of power. We already know that the two samples come from the same underlying distribution—a distribution determined by the microarchitectural design space of cores. Calculating T_1 allows us to evaluate how much the subset "looks like" the complete set—how representative the selection is. A representative selection captures most of the diversity of the Pareto-optimal set, while a selection that is not representative is not diverse. The same methodology can be applied to points derived from a combination of heterogeneity and DVFS, and metrics other than power and speed can be used. An advantage of the KS test is that it favors subsets that cover the entire range of cores even if the complete set contains clusters of cores that skew the distribution. This application of the KS test assumes that cores that are so slow or so high-powered that they should never be considered have been pruned from the complete set.

We demonstrate T_1 with an example. The example shows how diversity can be evaluated; it is not intended to provide new insights into CPU design or workload characterization. We simulate eight out-of-order CPU cores with gem5 [2] and McPAT [5], varying the sizes of the caches and the amount of ILP that can be extracted through instruction reordering and register renaming. We run the *des* (encryption), *djpeg* (image decode), and *mp2e* (video encode) benchmarks from the EEMBC DENBench (digital entertainment) suite [9] on each core. We then hand-select three cores for high flexibility, and another three for low flexibility, as shown in figure 4. These are better and worse selections, but not necessarily the best and worst selections. Table 1 shows T_1 for each selection, and normalized average ED^2 for comparison. While ED^2 suggests that the diverse selection is worst and the clustered selection is best, T_1 correctly shows that diversity is worst on the clustered selection, improves with the diverse selection, and is maximized when all cores are included. This supports the argument in section 4 that optimizing for efficiency does not lead to diversity.

This example shows how T_1 can be used to choose the better of two selections of cores. We expect that in reality, a processor designer will have access to tens of benchmarks and hundreds of different cores. Since T_1 is a quantitative measure, the designer can implement a search algorithm that maximizes diversity for the given benchmarks with a small number of cores, while also taking into account benchmark priorities and any other design requirements. Such a design methodology that targets runtime flexibility is impossible without a way of evaluating the diversity of the selected cores.

6 Conclusion

We have argued that a heterogeneous processor must contain a diverse set of cores to provide flexibility at runtime. Achieving diversity—a uniform spread of cores—is difficult because of differences in how programs interact with cores' microarchitectures. By quantifying diversity, the KS test enables future work in selecting core types for flexibility. Future work also includes methods for determining how many of each type of core is required, given multiprogrammed and multithreaded workloads.

References

[1] O. Azizi, A. Mahesri, B. C. Lee, S. J. Patel, and M. Horowitz, "Energy-performance tradeoffs in processor architecture and circuit design: A marginal cost analysis," in *ISCA*, 2010.

[2] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comp. Arch. News*, vol. 39, no. 2, 2011.

[3] T. D. Burd and R. W. Brodersen, "Design issues for dynamic voltage scaling," in *ISLPED*, 2000.

[4] W. J. Conover, *Practical nonparametric statistics.*, 3rd ed., ser. Wiley series in probability and statistics: Applied probability and statistics section. John Wiley & Sons, Inc., 1999.

[5] S. Li, J. H. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *MICRO*, 2009.

[6] A. Lukefahr, S. Padmanabha, R. Das, R. Dreslinski Jr., T. F. Wenisch, and S. Mahlke, "Heterogeneous microarchitectures trump voltage scaling for low-power cores," in *PACT*, 2014.

[7] A. J. Martin, M. Nyström, and P. I. Pénzes, "Et^2: A metric for time and energy efficiency of computation," in *Power Aware Computing*, ser. Series in Computer Science, R. Graybill and R. Melhem, Eds. Springer US, 2002.

[8] Media Tek Inc. (2015, May) MediaTek launches the MediaTek Helio X20. Accessed 2015-05-18. [Online]. Available: http://www.mediatek.com/en/news-events/mediatek-news/mediatek-launches-the-mediatek-helio-x20-the-worlds-first-mobile-soc-featuring-tri-cluster-cpu-architecture/

[9] J. A. Poovey, M. Levy, S. Gal-On, and T. M. Conte, "A benchmark characterization of the EEMBC benchmark suite," *IEEE Micro*, vol. 29, no. 5, 2009.

[10] Samsung Electronics Co. Ltd. (n.d.) Samsung Exynos. Accessed 2014-02-17. [Online]. Available: http://www.samsung.com/global/business/semiconductor/minisite/Exynos/products5octa_5420.html

[11] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, 1999.

[12] V. Zyuban and P. M. Kogge, "Optimization of high-performance superscalar architectures for energy efficiency," in *ISLPED*, 2000.