RETROSPECTIVE: Single-ISA Heterogeneous Multi-Core Architectures for Multithreaded Workload Performance

Rakesh Kumar Dean M. Tullsen Parthasarathy Ranganathan Norman P. Jouppi Keith I. Farkas

University of Illinois, Urbana-Champaign University of California, San Diego Google Google VMWare

I. Technical Backdrop

This work began in 2001 when multicore processors had just started appearing. It was apparent to all a major technological shift was happening.

Technological shifts typically represent tremendous opportunities to rethink the assumptions that shaped past products. In our case (during a fruitful collaboration over a number of years) we chose to question two assumptions, (1) that all cores should be identical, and (2) that they should be distinct and separable with no shared functional blocks. The work we are discussing here is the result of questioning that first assumption.

The earliest multicores were not aggressive – two cores stamped onto the same die with no communication or common logic between them. They soon became more truly multicore, with communication between them, peripheral logic (and eventually caches) shared, etc. But even in the early 2000's, all roadmaps were clearly symmetric multiprocessing.

So why was this technological shift the right time to question the "all cores identical" assumption? In a multiprocessor composed of uniprocessor chips, particularly if you optimized the cores to coexist in a heterogeneous design, it would require two (or more) distinct fab runs to produce those chips, and all the fixed costs associated with the production of each chip. For a single-chip heterogeneous multicore, you could introduce all the heterogeneity you wanted, and still require only a single fab run to produce it. The economies of scale no longer favored homogeneity.

At that time, the primary mechanism being deployed (or even discussed in the literature) to save power for applications that did not utilize the whole processor was clock gating. If a 4-wide superscalar was getting an IPC of 0.5, we could clock gate most of the ALUs and some other structures. But the gains were limited to a small percentage of total power. Besides, this was a time when power was already a first class concern, voltage scaling had more or less stopped, and static power (which clock gating could not target) was already significant. It was clear that we needed a dramatically better power reduction technique.

II. Single-ISA Heterogeneous Multicores

In Micro 2003, we introduced the idea of heterogeneous multicores [1]. The primary gain was power efficiency. That work showed that performance difference between a "modern", heavyweight core and a much less aggressive core varied dramatically, sometimes as much as 20X, other times almost not at all. It varied by program, but also program phase. The latter point is critical.

We made two assumptions in that work to minimize the perceived barriers to adoption. One, we assumed we were constrained to use pre-existing cores that had already been designed and verified – this constrained us to a further limitation that the cores tended to be monotonically increasing in capability along all dimensions. The second assumption was that the cores all shared a single ISA – because different phases of execution of the same program often required dramatically different core choices, the ability to adapt dynamically and move quickly between cores was critical. We challenged the first assumption over the next few years, looking at sets of cores co-designed to work optimally together, and not necessarily providing monotonically increasing resources [2]. We did not challenge the second assumption until almost a decade later, and found that increased diversity (including now the ISA) only magnified the advantages [4].

The subject of this retrospective, however, is the second heterogeneous multicore paper. While the first had focused heavily on power, we wanted to make a more clear performance argument (given various caps on resources). Second, we imagined a different use case – in the first paper, we assumed one thread using one of the cores at a time and the others all powered down. In this paper, we considered the more common case of having many cores and many jobs to run, and a scheduler having to figure out what jobs to run on what cores (which now depends not only on the characteristics of that job, but also those it is co-resident with).

Our results showed that a single-ISA heterogeneous multicore architecture can provide significantly higher performance in the same area than a conventional multi-core processor by matching each application to a core that is just right for it. It can provide high single-thread performance when thread-level parallelism is low and high throughput when thread-level parallelism is high. This provided another incentive to build such processors and added to commercial and research impact. We also presented several practical core assignment policies that maximize efficiency.

Finally, we also chose to hit head-on an issue we had ignored in the first paper due to our single-thread assumption – that the largest cores were likely to include simultaneous multithreading (SMT). This created much more messy (i.e., interesting) tradeoffs for the scheduler. In fact, this introduced the first element of non-monotonicity that we would advocate for in later papers. In the simple design of the first paper (big, medium, and little cores), it was always clear which core would give the best performance and which would give the worst. But with big SMT cores and small non-SMT cores, it was not at all clear whether a thread should be scheduled as the second thread of the big core, or run alone on a small core (particularly given the impact on the first thread).

III. Impact

By 2001, when this work began, there was a dire need for a new way to save power. By that time, the technical and commercial feasibility of putting multiple cores on the same die had also been established. In that context, when single-ISA heterogeneous multi-core processors were proposed, the impact was near-immediate. Follow-on papers from industry and academia appeared quickly, validating the power reduction potential as well as energy-proportionality benefits of such processors. Soon thereafter, a large number of projects spawned on different aspects of heterogeneous multi-core processor design, implementation, task scheduling, and deployment. Within a few years, commercial processors started appearing - first in the embedded and mobile domain and then in the desktop/tablet domain - that were based on the architecture.

For example, ARM big.LITTLE, introduced in 2011 to support single-ISA heterogeneous multi-core architectures, powers a significant fraction of popular mobile computing devices (e.g., most Samsung Exynos-based devices). Apple A12, A13, A14, etc., that power IPhones support single-ISA heterogeneous multi-cores. Apple M1 and M2 that power Macs and IPads are also single-ISA heterogeneous multi-core processors. Intel's Alder Lake and Raptor Lake processors that power desktop and mobile devices use performance cores and efficient cores in single-ISA heterogeneous multi-core configuration. Several generations of nVidia's Tegra cores, starting with Tegra 3 introduced in 2001, adopted single-ISA heterogeneous architecture. The list of systems based on single-ISA heterogeneous multi-core architecture continues to grow.

Of note is that most of the earlier commercial heterogeneous designs followed our earliest examples, mixing and matching pre-existing ARM cores. However, more varied heterogeneous multi-core designs have started appearing. Of particular interest, relative to this paper, is Intel's Alder Lake and it's mixing of large SMT and smaller non-SMT cores – the same architecture we considered in the latter part of this paper. As a result, it is the first heterogeneous multi-core design with non-obvious scheduling decisions even in the most simple case – when scheduling purely for performance. In general, we are seeing more evidence that the industry is willing to custom design cores that complement each other, rather than mixing previous designs optimized to be the only core type.

IV. The future of heterogeneous architectures

We think that customization of cores will continue to increase. We have done a number of studies that showed that in a world of infinite design choices, and running all permutations of a large set of applications, the best multicore design was (of course) never homogeneous, but also rarely monotonic (ie, small, medium, big). Rather, the best designs were typically much more mixed – e.g., a wide in-order core (great for ILP intensive workloads) and a narrow, out-of-order core (better for memory-intensive workloads), etc. We expect non-monotonic heterogeneous multi-cores to appear at some point.

There are PPACT (power-performance-area-cost-time) tradeoffs between different kinds of heterogeneity - generational (i.e., integrating cores from different generation), market segment-based (i.e., integrating desktop and embedded CPU, etc.), and custom-designed (e.g., non-monotonic). Similarly, there are tradeoffs between single-ISA and multi-ISA heterogeneity. There are also tradeoffs between different levels of heterogeneity - e.g., two types of cores (which is what all current heterogeneous multicore products have) vs more core types. We expect different heterogeneity design points to be used in different scenarios.

Single-ISA heterogeneous multi-core architectures have not yet made their way to servers and datacenters. We expect the energy-proportionality benefits of these architectures to also push them into these domains.

Current heterogeneous multi-cores use fairly simple scheduling policies, even as the cores become more complex and varied. For example, Intel Thread Director uses only a small number of parameters to decide on the thread schedule. We think that OS scheduling [3] and corresponding hardware support will get much richer going forward.

The emergence and increased adoption of chiplets may give further impetus to heterogeneous multi-core architectures since it may become easier to mix-and-match cores to suit workload needs.

Overall, we would not be surprised if heterogeneous multi-core architectures become and remain the mainstay of all processors for at least the next decade.

References

[1] R. Kumar, K. Farkas, N. Jouppi, P. Ranganathan, and D. Tullsen, "Single-isa heterogeneous multi-core architectures: the potential for processor power reduction," in *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.*, 2003, pp. 81–92.

[2] R. Kumar, D. M. Tullsen, and N. P. Jouppi, "Core architecture optimization for heterogeneous chip multiprocessors," in *2006 International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2006, pp. 23–32.

[3] J. C. Mogul, J. Mudigonda, N. Binkert, P. Ranganathan, and V. Talwar, "Using asymmetric single-isa cmps to save energy on operating systems," *IEEE Micro*, vol. 28, no. 3, pp. 26–41, 2008.

[4] A. Venkat and D. M. Tullsen, "Harnessing isa diversity: Design of a heterogeneous-isa chip multiprocessor," in *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, 2014, pp. 121–132.