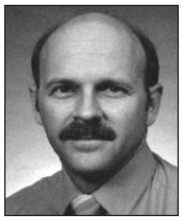
Microarchitecture in the Ditch

The Hunt for ILP Goes On, But Maybe It's Time for a Different Approach



After more than a decade of efforts to extract instruction-level parallelism, ILP doesn't appear to be the panacea that was once envisioned. Companies have poured many millions of transistors into wide instruction-issue mechanisms and large instruction-reordering windows. What do they have to show for all this complexity? Precious little.

Following the initial jump to dual-issue and simple reordering—which picked the low-hanging ILP fruit—more complex mechanisms have delivered little additional ILP. Although CPU performance has held on to its 55% annual growth rate, little of this increase is attributable to issue width or reorder depth. The three-issue out-of-order Mendocino, for example, executes only 15% more instructions per cycle on CPU benchmarks than its two-issue in-order predecessor (P55C), despite the fact that Mendocino has twice as many CPU-core transistors. For Mendocino, as for other processors, higher frequencies, faster caches, and, in the multimedia domain, SIMD units, have contributed far more to performance than has ILP.

Architects have often been misled by simulations, which have a nasty habit of predicting better ILP than what shows up in the final product. Unfazed by such practical realities, academics remain optimistic; visions linger of 32-issue processors with astronomical ILP. Intel and HP hope to tap into this ILP with their EPIC microarchitecture. Although this approach, or some other high-ILP mechanism, may eventually succeed, one finds little encouragement in the progress made to date.

The problem could be ILP itself. Maybe there isn't as much in real software as we think. Long-promised compiler technology to expose more ILP hasn't materialized, and memory latency continues to be a governor on ILP, throttling back complex processors. Regardless of the reasons, architects are tiring of the ILP hunt. Recent chips have focused more on clock rates and caches than on issue width. Perhaps there is a good reason that Intel hasn't taken the obvious step of beefing up the P6 with wider issue and a larger reorder buffer.

Architects have become loath to sacrifice any frequency for issue complexity. This is partially because frequency sells, but it's mostly because frequency is more likely to yield tangible performance gains. ILP often extracts a high price in frequency, reducing its net value. Although generous application of transistors (e.g., replicating logic) can mitigate the frequency loss, there are fundamental effects that—all else being equal—reduce the clock rate of complex wide-issue machines compared with simple narrow-issue ones.

To boost frequency, processors have steadily increased their pipeline lengths. But this technique has limits: beyond 12–15 pipeline stages, latch overheads and branch stalls offset frequency gains. Many processors have already reached this limit; beyond it they must rely on process technology for further frequency gains. Unfortunately, transistor speed increases at only 20% per year, and interconnect physics will make even this rate difficult to sustain for long.

Adding on-chip cache remains a viable route to higher performance. But this avenue may also be short. Once on-chip caches reach 2–4M—easily achieved in just a couple years with 0.13-micron technology—diminishing returns kick in. Even very large software systems have finite working sets that are satisfied by caches of this size.

Maybe it's time to take a completely different approach. One interesting approach enabled by 0.18-micron technology is chip multiprocessors. CMPs go after process-level parallelism using multiple processors on a single die, typically sharing a common on-chip cache. Enormous bandwidths and low interprocessor communication latencies make the technique fundamentally different from discrete multiprocessing. CMPs are well suited to media processing; with SIMD-enhanced cores, a CMP can exploit low-level data parallelism as well as high-level process parallelism, leaving little need for ILP. Thus, CMP cores can be simple, small, and fast.

An enormous benefit to CMP construction is that a chip with a defective core can still be functional. This fact improves yield and significantly lowers manufacturing costs compared with those of a uniprocessor of similar size—the economic equivalent of free transistors.

The drawback to CMP is software. Because CMP parallelism is explicit, taking full advantage requires that applications be written expressly for it. Compilers to automatically exploit the latent parallelism of closely coupled cores are far off, although maybe no farther than high-ILP compilers. But even with its software difficulties, CMP has compelling attributes that make it worth pursuing. RISCs, with their smaller cores, will be the first to test the concept.

There are other promising techniques in research that, like CMP, go after other levels of parallelism. Simultaneous multithreading, multiscalar, and single-program speculative multithreading, for example, exploit parallelism between process threads. Maybe EPIC and other ILP mechanisms aren't the only—or even the best—roads to performance. M

[illegible]