ILP is Dead, Long Live IPC!

(A position paper.)

Augustus K. Uht
Microarchitecture Research Institute
Dept. of Electrical and Computer Engineering
University of Rhode Island
December 15, 2005

Overview: the State of the Art

We will discuss the current state of microprocessor architecture, where it is currently headed, and where it should be headed. Specifically, until recently processors consisted of one copy of a CPU, the latter exploiting as much Instruction-Level Parallelism as possible. This improved performance.

Unfortunately, two trends collided and caused a rapid shift in processor architecture: 1) The architecture community's ability to extract ILP from typical code asymptotically approached zero, so processor companies kept increasing the CPU clock rate to compensate and improve performance in a brute force way. Marketing also played a large role here: it's relatively easy to sell processors based on a single [higher] number. 2) Processor temperatures were becoming excessive. The newer processors from Intel (Prescott Pentium 4's) have power dissipations measurable in light bulb or toaster equivalents.

The net result was the almost-appearance of the ill-fated 4.0 GHz Pentium 4. It could not be reliably sold or used due to its large power dissipation; in short, it would burn up. The industry 'fix' to this problem is to put two or more CPU's on a chip or package ('multi-core' processors) and run them at lower speeds, thereby reducing power to acceptable levels while increasing performance. Is there a fallacy here? Is this the right way to go?

Looking at Figure 1, a chart from a recent Intel talk given by Benson Inkley [1], the two processors being compared are a single-core 3.73 GHz Pentium 4 vs. a dual-core Pentium 4 –based 3.2 GHz 840 processor. Both processors use dual-threading (Hyperthreading). The performance numbers are based on the execution of the SPEC 2000 benchmarks, both integer and floating point. We will focus on the left-most comparison: the processors running the SPECint2000 benchmarks without tuning.

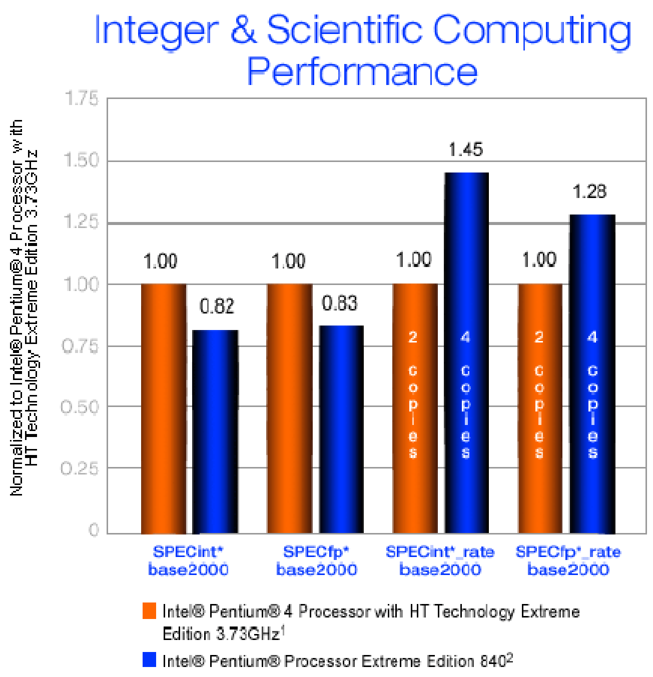

Figure 1. Intel single vs. multi-core performance.

We see that by doubling the number of cores, performance increases by only 18%. While the single-core processor is 'faster' than the individual cores of the 840 processor (3.73 vs. 3.2 GHz), on a cycle-by-cycle basis one would expect a performance gain of 72%.

Examining the power requirements of the two processors, from the datasheets [2, 3] we see that the total suggested design power (NOT the peak) for the single-core processor is: 115 W, and 125 W for the dual-core processor. However, the Intel-supplied fan runs at 24 W.

Therefore, for about the same power dissipation, and about twice the cost, we get a slight performance increase of about 18%.

I have been unable to determine the specifics of the experiments. For example, were the SPECInt benchmarks initially recoded to take advantage of the multithreading and dual core options? This seems likely. In a more realistic scenario, a shrink-wrapped program, unable to be recompiled, would likely have experienced a performance *decrease* when going to the dual-core processor, since each core runs at a lower speed than the 3.73 single-processor.

The conclusion is that for normal, nominally sequential programs, multi-core processing is a 'lose.' (Of course, if a program can be recompiled, and it contains lots of parallelism, like many media applications, then there will likely be a performance benefit from multi-core processors, but not necessarily proportionally; see the right-hand side of Figure 1.)

Parallelizing Compilers

In its current state, the hardware community can not make progress in improving the performance of the vast majority of existing or future programs. As currently envisioned, multi-core processors are not useful. However, industry has thrown the problem over the wall to the programmers, and said: 'You fix it. You MUST learn how to program in parallel.'

While dictating technological progress can sometimes have a positive effect, it is unlikely in this case. Look at the supercomputing literature: for the past thirty to forty years researchers have attempted to write auto-parallelizing compilers, with only modest success. The community has turned to such tools as OpenMP or assertions, both of which are very hard to use and port. Also, the use of assertions can easily lead to functionally wrong code.

Being realistic, we all have a hard time writing *sequential* programs, much less *parallel* programs. This is true for novice and expert programmers alike [4].

But we still want to improve performance. Is there a solution? I think so: ***'Long Live IPC.'***

Back to the Future

Whatever happened to ILP? Many, many limit studies have shown large amounts of ILP (potential parallelism) in typical programs, even gcc [5]. But architects have been unable to realize the potential performance in IPC (realized parallelism).

ILP exploitation is hard, no doubt about it. But computer architects have solved tougher problems.

For many years, the classic superscalar architecture has become a de facto standard. No serious deviations from its microarchitecture are allowed. Little changed, little gained.

We need to wipe the slate clean, and create dramatically new microarchitectures in order to make significant gains. This is generally frowned upon by industry, which doesn't like big changes. (But recall, the Intel P6 microarchitecture was a radical change, and it paid off big. Intel is even returning to it: the mobile Pentium M processor is based on the P6.) The results of this industry bias are a slew of incremental performance improvements.

Some of us are starting fresh, e.g., the TRIPS machine [6] and the Levo machine [7]. Both have yielded IPC's (not ILP) greater than three and five (resp.) with realistic simulation assumptions. (TRIPS requires compiler support, Levo does not.)

But this is just the start. Power is still an issue. We must get away from the frame-of-mind that a microprocessor must use as many transistors as possible. On the contrary, it should use as few transistors as possible. (Sounds obvious, but we seem to have forgotten this.)

We must also re-examine the multi-core model in even its most basic sense. Forget about duplicating entire processors. Remember, we can't program the end result. Think of using less complex and less costly computation units.

Conclusions

Everyone has fallen in behind the microprocessor manufacturers in the multi-core futility, even the major operating systems' and applications' programmers. Even the latter say they won't be able to do anything for years [4].

'Those who cannot remember the past are condemned to repeat it.'

Let's not waste another 30-40 years. If there was ever a time to think out-of-the-box, this is it. Let's do some real envelope-pushing. Multi-core machines (the same as multiprocessors) and parallelizing compilers are well known and are not likely to produce any meaningful new results.

We need IPC; we don't need PPC (processors per cycle).

References

[1] B. Inkley, "Intel Multi-core Architecture and Implementation." Hillsboro, OR: Intel Corp., 2005. Talk given at GSPx.
[2] Intel Staff, "Intel Pentium Processor Extreme Edition 840," Intel Corp., Datasheet 306831-002, October 2005.
[3] Intel Staff, "Thermal Management for Boxed Intel Pentium 4 Processor in the 775-land Package," Intel Corp., 2005. URL: http://www.intel.com/cd/channel/reseller/asmo-na/eng/products/box_processors/desktop/proc_dsk_p4/technical_reference/99346.htm.
[4] K. Krewell, "Software Grapples With Multicore," *Microprocessor Report*, December 12, 2005.
[5] M. S. Lam and R. P. Wilson, "Limits of Control Flow on Parallelism," in *Proceedings of the 19th Annual International Symposium on Computer Architecture*. Gold Coast, Australia: IEEE and ACM, May 1992, pp. 46-57.
[6] K. Sankaralingam, R. Nagarajan, H. Liu, C. Kim, J. Huh, D. Burger, S. W. Keckler, and C. R. Moore, "Exploiting ILP, TLP, and DLP with the Polymorphous TRIPS Architecture," in *Proceedings of the 30th Annual International Symposium on Computer Architecture*. San Diego, California, USA: ACM and IEEE, June 9-11 2003.
[7] A. K. Uht, D. Morano, A. Khalafi, and D. R. Kaeli, "Levo - A Scalable Processor With High IPC," *The Journal of Instruction-Level Parallelism*, vol. 5, August 2003. (http://www.jilp.org/vol5), URL: http://www.ele.uri.edu/~uht/papers/JILP2003FnlLevo.pdf.