

Draft

Rebuttal of Worley - Huck Paper on IA-64

3/9/2000

I was somewhat disappointed in the Worley-Huck paper on IA-64 that appeared in the February Microprocessor Report. After a decade of work a few more facts would have been welcome. However close reading of the paper reveals many hidden messages.

After observing that all implementations require good memory systems they then state “Criticizing the IA-64 architecture on the basis of an initial memory-system design, which balanced cost and performance, is a bit unfair.” This is an amazing contention. I leave out the use of the word “unfair”. (Are Intel and HP now asking for exemption from criticism?) Balancing cost and performance is what engineers do. If the IA-64 drives one to uneconomical solutions then it’s a bad architecture. After all it’s not as if the importance of memory is some brand new fact. They then go on to speculate how they could in the future do a memory system that would compete with an out of order superscalar. “... one can find better approaches to use available memory technology if one is willing to accept additional cost.” In other words if we were willing to build a much more costly machine using some other set of techniques IA-64 would be a good deal.

After stating that initial use of IA-64 will be in the “high-end” market, they claim that, “Over time designs will broaden out to a wide range of markets.”. For evidence they state, “An HP Labs study for high-performance embedded controllers has confirmed that EPIC-like machines are exceptionally effective.” Notice they don’t claim that IA-64 will scale down to low power implementations. A new architecture based on “EPIC-like machines” is required. A reasonable conclusion is that IA-64 doesn’t scale.

One questionable comment is that “OOO engines ... are restricted to fixed hardware algorithms for correctly predicting the execution path and for triggering memory fetches.” The shoe is actually on the other foot. For example the branch prediction hardware used in OOO machines dynamically changes the branch prediction based on context and recent history. EPIC style machines make such decisions at compile time. It would be nice to know which strategy is best on a large class of programs.

“The IA-64 compiler has heuristics to control the degree of speculation, and the programmer has control over the compiler’s heuristics.” So, not only does the application development team have to develop training suites for profiling, they have to fine tune the compiler optimizations. This suggests a complete lack of understanding of the programming process. As I’ve said before, programming is hard; we should be doing all we can to ease the task. More compiler knobs don’t help.

An interesting insight into the architectural thinking that went into IA-64 can be seen in their discussion of the register stack engine (RSE). This is a feature that operates asynchronously to

save and restore state over procedure calls. They conclude “The RSE does not add to the critical path of the machine and is a relatively small part of the Itanium and McKinley designs”. I think most architects and designers would agree that there’s a lot of hidden complexity in such a feature. One is reminded of the old adage, if you get something for nothing, it’s probably because you previously got nothing for something. I have a feeling that this describes a lot of IA-64.

Quite a bit of their article is on what might be done rather than what has actually been accomplished. IA-64 is basically an in-order machine. The authors write, “An in-order machine has options beyond a simple stall when a cache miss occurs. Running ahead with rollback, predictive address buffers, implicit prefetch, buddy prefetch, and other dynamic approaches can all be effective.” The authors seem to be reluctant to say whether their implementations are stall on miss or use. What is clear is that, whatever they’re doing now, they will need out of order style mechanisms in the future.

The silence on X86 performance is deafening. Let me speculate. Itanium will be 800 MHz and X86 processors in the same time frame will be running at 1.2 GHz, a factor of 1.5:1. The X86 code will have to be interpreted sequentially or dynamically translated. The first technique means it can’t go out of order. The second says they will pay a high price for translation and the mechanisms to keep instruction caches consistent, etc. Either way there is another factor of 2:1. Even with hardware assists there is probably another 20% lost due to handling corner cases where the hardware doesn’t quite match X86. Papering over the architectural differences of MP and other memory and system aspects will cost another 10%. Give them 10% for cleverness that I’ve not considered and Itanium is still probably a factor of three off the best X86 in the same technology. Because McKinley was designed by HP, without benefit of the X86 expertise of Intel, it may well be worse.

The overall tone of the article seems to be excuses for now and high promises for the future based on data that isn’t provided to the reader. “Any initial implementation of a new architecture will be conservative and will concentrate on the most important architectural elements. Itanium and McKinley are not exceptions to these rules.” The hidden message here is that, not only is Itanium a development vehicle, but McKinley isn’t so great either. How long must we wait for the real thing? Their conclusion indicates a fundamental blindness to the broad range of factors that go into a good architecture and implementation. “IA-64 will deliver on its promise of expressing, enhancing, and exploiting instruction-level parallelism to improve performance.” We need to know a lot more about such issues as frequency, power, chip size, IA-32 performance, cache sizes and geometry, and ease of use by programmers. These are much more important than ILP. When will we learn about them?