Panel Session
Has Exploitable ILP Reached a Point of Diminishing Returns?

Moderator:
R. Govindarajan, Indian Institute of Science, Bangalore, India.

Panelists:
Anant Agarwal, Massachusetts Institute of Technology, Cambridge, MA, U.S.A.
Manoj Franklin, Clemson University, Clemson, SC, U.S.A.
K. Gopinath, Indian Institute of Science, Bangalore, India.
Vinod Kathail, Hewlett Packard Labs., Palo Alto, CA, U.S.A.
Krishna Palem, New York University, New York, NY, U.S.A.
Vivek Sarkar, IBM Academy of Technology, IBM Software Solutions Division, CA, U.S.A.
Mateo Valero, Universidad Politecnica de Catalunya, Barcelona, Spain.

Modern processor architectures are becoming increasingly sophisticated, with advanced architectural techniques for exploiting greater instruction-level parallelism (ILP). Most of today's superscalar processors support 4-way instruction issue supported by multiple execution units. Aggressive compilation techniques have been employed to expose more ILP in programs. Yet the return – the exploited instruction-level parallelism – is low, questioning the investments.
Why is exploitable ILP still low? Is there any hope it will improve? Where should we look?
The panelists' position statements are given below:

Anant Agarwal

Rapid advances in technology force a re-evaluation of current architectural approaches. Current architectures, such as hardware scheduled superscalars, are hitting performance and complexity limits and cannot be scaled indefinitely. I believe a much simpler software-supported architecture will replace existing methods. Such an architectural approach, called Raw, uses a simple, replicated, wire-efficient architecture that scales with increasing VLSI gate densities, and exposes completely its hardware details to the software system. Raw architectures allow the software system to discover and schedule instruction-level parallelism and stands to exploit current trends towards stream-based multimedia computations.

Manoj Franklin

I do not believe that exploitable ILP has reached the point of diminishing returns. The number of transistors integrated on a single chip has been steadily increasing, and computer architects have been finding ways to make use of the additional transistor budget for exploiting more ILP. The major hurdles that need to be overcome, to exploit further ILP, are: frequent control flow changes, complexity of scheduling hardware, and memory latencies. The hurdles due to control dependences are being overcome by inventing better prediction schemes. The scheduling hardware is being simplified by inventing good decentralization schemes. Cache memories, which have been used to reduce memory latencies, appear to be running out of steam. To handle the wide disparities between processor and memory speeds, computer architects are investigating schemes to integrate the two. Compiler capabilities have also been steadily improving.

Vinod Kathail

There have been significant advances in ILP architectures and compilers, but we are nowhere close to reaching the point of diminishing returns. A substantial amount of work remains to be done to fully realize the potential of the ILP technology.

ILP compilers need accurate and unambiguous information about program in areas such as memory dependences and branching behavior. Language support and more accurate analysis techniques will certainly help. In addition, there is a shift towards "statistical compilation" in which program statistics, such as memory dependence profile and branch frequencies, are used to guide the compilation process. Statistical compilation relies upon and makes use of novel architectural features such as predication, control speculation, data speculation and programmatic control of caches.

The performance of a program on ILP processor is often limited by control and data dependences. Such performance limits can be avoided using program transformations that reduce the height of critical paths. As the parallelism provided by the processor increases, critical path reduction techniques become increasingly important.

An ILP compiler's task is inherently complex. As a result, its speed and robustness are serious issues that needs to be addressed. Compiler complexity escalates with new hardware features and more complex algorithms for improving performance. Research is needed to improve the algorithms as well as the compiler architecture into which they are integrated. Region-based compiling provides an architectural framework to address some of these issues.

Novel architectural features, statistical compiling, critical path reduction techniques and the region-based framework have the potential to deliver practical systems that achieve high levels of ILP.

K. Gopinath

Many studies (based on data-flow architectures or based on trace-generated instruction streams) have shown that there exists considerable potential for ILP parallelism. Current architectures, however, are not in a position to effectively exploit ILP present due to lack of support for speculation, predicated execution, memory disambiguation and software pipelining; inability to be resilient to long latencies due to cache misses (as instruction scheduling is handled by a compiler), etc. In addition, compiler technology has not been sufficiently developed for predicated architectures, control and data critical path reduction, predicated data flow analysis (liveness analysis, alias analysis, $\cdots$), *etc.* It may be premature to say that exploitable ILP has reached the point of diminishing returns until we examine the combined potential of such newer architectural designs and compiler technology as they coevolve synergistically.

Krishna Palem

The evolution of RISC technology has enabled access to substantial computing power at relatively low cost by exploiting instruction level parallelism (ILP). The very attractive cost-performance ratio and scalable nature of ILP has the potential for enabling wide ranging applications in the embedded domain. Typical applications include high-performance video servers for multimedia applications, and controllers in hand-held video games. However, in order to harness the promised high performance of processors via ILP, several key research questions have to be addressed. These challenges arise from the fact that quite often, embedded applications have "timing-constraints" that have to hold between parts of the application program. Consequently, response-time and related metrics tend to dominate traditional metrics of performance such as throughput or utilization. Thus, the source programs have to be scheduled by the compiler, preserving the timing constraints between parts of the program; Also, the timing behavior of the programs as they execute on the targets ought to be predictable; compile-time management of the memory hierarchy, so that cache access latencies are predictable is again important.

We posit that attractive that the opportunity might be, embedded applications raise serious and exciting challenges that need to be addressed, before ILP technology can be used effectively. I will outline some of the emerging research questions to be tackled.

Vivek Sarkar

I believe exploitation of instruction-level parallelism started on the wrong foot and was at a point of diminished returns right from the beginning. Though modern optimizing compilers deliver a reliable performance improvement due to instruction scheduling for a wide range of programs, the magnitude of the improvement due to instruction scheduling is disappointingly modest (in the range of 20%). This is in contrast to parallel speedups of 2X or more obtained by exploiting medium-grain multithreaded parallelism on SMPs. I believe one reason for this discrepancy is that exploitation of instruction-level parallelism was initially conceived at the basic block level. Most global instruction scheduling algorithms such as trace scheduling and superblock scheduling strive to use a basic-block scheduling framework in a wider scope and hence retain many of its limitations (*e.g.,* preserving the order of branches and only allowing instructions to be moved "upwards"). There is an opportunity for better exploitation of instruction-level parallelism by taking a fresh look at global instruction scheduling in a way that encompasses the full range of permissible code motion transformations that can be performed across all regions of a program.

Mateo Valero

Current superescalars processors, that are able to issue four instructions for processor cycle, obtain a speed of not much more than one instruction per cycle in average. On the other hand, many papers indicate that programs contain much more high levels of parallelism and this is the reason why the ILP paradigm is important.

Collaboration between compilers and architectures is necessary in order to exploit the available ILP of programs. This seems to be harder for integer codes than for floating point codes. For integer codes, architectures such as Multiscalar or Trace Driven Multiscalar try to do the best. In the later case, the compiler can extract enough parallelism than can be exploited in multiprocessor, multithreaded or VLIW architectures. For vectorizable codes, the addition of a vector accelerator hardware to current superescalar processors is a very cost/efficient mechanism that permit to get more than one order of magnitude the speed of the current superescalar processors.