Hybrid-Scheduling: A Compile-Time Approach for Energy-Efficient Superscalar Processors

Madhavi Valluri and **Lizy John**
Laboratory for Computer Architecture
The University of Texas at Austin, Austin, TX 78712
{valluri,ljohn}@ece.utexas.edu

Modern superscalar processors are increasingly relying on complex run-time techniques such as out-of-order issue, branch prediction, value prediction, etc to achieve high performance. These design trends are causing power dissipation and energy consumption on the chip to grow rapidly. Increased power dissipation directly affects circuit reliability and packaging costs. With the increasing use of high-performance general purpose processors in the portable devices such as laptops, where battery life is at a premium, reducing the total energy consumption is also becoming critical. Our current research investigates compiler solutions to these design bottlenecks.

Our work targets the power-hungry dynamic scheduling units in the processor. These units consume as much as a third to half of the processor energy budget. Dynamic issue hardware is useful in extracting parallelism in programs where the instruction scheduler in the compiler is unable to find parallelism statically. The compiler is often limited by lack of information of input data, irregular control flow in programs, library and system calls etc. However, for regular and well-structured programs such as media and scientific applications, the compiler can generate efficient (almost optimal) schedules for considerable portions of the code. In processors with dynamic issue logic however, the hardware searches for parallel instructions, *irrespective* of whether the compiler-generated schedule is perfect or not. Hence, in regular applications or regions, the dependence analysis and scheduling performed by the dynamic scheduling hardware is completely redundant.

We propose a technique that combines the advantages of both compile-time static scheduling and run-time dynamic scheduling to lower the energy consumption in a processor. In this **Hybrid-Scheduling** paradigm, regions of code containing large amounts of parallelism that can be identified and exploited at compile-time bypass the out-of-order issue logic and are issued and executed exactly in the order prescribed by the compiler. The processor runs in the superscalar mode with dynamic scheduling until a special instruction that indicates the beginning of a statically scheduled (S-Region) is detected, at which point, the out-of-order issue engine is shut down, and the processor switches to a VLIW-like *static mode* in which instruction packets scheduled by the compiler are issued sequentially in consecutive cycles with minimal hardware support. The hybrid-scheduling scheme is particularly suited for high-performance general-purpose systems which need to cater to diverse application domains such as integer, media and scientific applications. Further, the hybrid-scheduling scheme is particularly effective in regions where the recently proposed dynamic microprocessor resource adaptation schemes (eg: Folegnani et. al '01) have been less effective. Preliminary results using media and scientific programs show that this technique can save an average of 43% energy consumption in kernels and 25% in full applications with minimal performance degradation (ISLPED 2003).

Our ongoing research efforts are towards extending the scheme for irregular applications such as SPECint etc. Hard-to-predict branches, pointer-intensive memory accesses and extensive use of function and library calls make it difficult for the compiler to find regions it can statically schedule efficiently. While our preliminary work concentrated on loops as potential S-Regions, for integer programs, we are investigating program structures as hyperblocks, superblocks and traces as S-Regions. We are currently developing an out-of-order issue simulator for the Trimaran 2.0 compiler framework and experimenting with different types of S-Regions for SPEC CPU2000 applications.