Simultaneous Multithreading: Maximizing On-Chip Parallelism

The paper examines simultaneous multithreading (SM), a technique that improves the overall efficiency of superscalar processors by using hardware-level multithreading. SM exploits parallelism by allowing multiple independent threads to issue instructions to multiple functional units in a single cycle. Thus, instructions from more than one thread can be executed in any pipeline stage at a given time. The authors first provide a detailed experimental analysis identifying the bottlenecks of wide superscalar execution and claim that SM has the potential to decrease the waste associated with unused issue slots increasing the processor utilisation. They introduce four machine models for SM with various levels of hardware complexity and they evaluate their performance against single-threaded superscalar and fine-grain multithreading processors. The outcome shows that more complex SM models outperform both processor types when run on a wide superscalar while simpler SM implementations with limited capabilities per thread, can still gain high instruction throughput. However, regardless the SM model, the instruction throughput is capped by the cache sharing. The authors are simulating different cache designs and evaluate the tradeoffs in their try to determine the optimum configurations. They show that a proper tuning of cache organisation has beneficial results in performance and multi-thread contention. Finally, they make various experiments in order compare the performance of SM to small-scale on chip multiprocessors and demonstrate the superior performance advantages of the first while using fewer execution resources.

Strong points:
The authors are introducing an evolutionary processor architecture by combining multiple instruction issue from superscalar architectures and the latency-hiding ability of multi-threaded architectures. SM is able to overcome limitations imposed by low single thread instruction-level parallelism and increases the resource-efficiency of chip-level thread parallelism. It is a very interesting idea that effectively hides the impact of latency related issues. Running multiple threads has beneficial results in camouflaging the individual control hazards (e.g: branch mispredictions) and other penalties from long latency operations (e.g: main memory access latency due to cache misses). The topic is investigated in depth. The authors are very cautious in their simulations and they have modelled all sources of latency (cache, memory, TLB, branching, real instruction latencies) with attention to detail. The experimental process is very well structured and smoothly drives the reader from the results of one experiment to another, reasoning the outcome of them on each case. Even though the technology is not mature enough for production-ready systems, it is definitely in the right direction. The contribution of the work in the field is undoubtedly a reference point and an important step towards shaping the technology of the future.

Weak points:
SM is a technique that definitely provides significant performance improvements, however there are some parts that require certain attention and are discussed below. Firstly, SM increases the complexity of instruction scheduling (control unit) relative to superscalar and causes shared resource contention, particularly in the memory subsystem where additional stress is placed on the hierarchy. The authors mention that the size of the cache contributes significantly to the results of the various SM models and provide more resistant configurations to cache effects. However, the paper lacks results with simulations using caches closer to the current processors. Even though they claim that the analogous simulations has been performed and they discuss the

results, it would be more solid and clear to present them in the actual paper. Since SM requires a larger file for the various SM models discussed in the paper, more cycles are needed in order to read it. Consequently, it potentially increases the penalty for branch misprediction and raises the complexity for the bypass. Finally, increased performance possibly comes with other costs. Even though SM manages to increase the utilisation on processors, the authors do not discuss anything about the power consumption which is of a high importance. While SM seems like a great solution, the complexities inherent in its design can add some serious issues when it comes to a real implementation.