Holden Sandlar

Advanced Computer Architecture – 0301 – 810

**Simultaneous Multithreading: Maximizing On-Chip Parallelism**

This paper mainly analyzed and argued for the advantages of simultaneous multithreading architecture models. Simultaneous multithreading (SM) is a technique in which multiple threads to issue to multiple functional units in any given cycle. Traditional multithreading architectures only issue instructions from a singular thread which can create both horizontal and vertical waste.

Four different multithreaded models were considered. Fine-Grain Multithreading allows only one thread to issue instructions each cycle, but it can use the entire issue width of the processor. Full Simultaneous Issue is the least realistic in terms of hardware complexity, allowing all threads to compete for open issue slots each cycle. This model gives insight into the potential for SM. Single-issue, dual-issue, and four-issue limit the number of instructions each thread can issue in a cycle. Limited connection directly connects each thread to a set of functional units (8 threads and 4 functional units – each FU could receive issues from 2 different threads). With these models defined, simulations were carried out to gather and analyze statistics about the performance of simultaneous multithreading.

After analysis of the collected data it was determined that cache sharing is the dominant cause of wasted cycles in simultaneous multithreading environments due to poor locality. One interesting and relatively cheap observation made was that increasing the TLB entries from 64 to 96 entries decreases wasted cycles from 6% to 1%. Several cache configurations were considered for the analysis to optimize the cache design. For all analysis L1 was configured, and L2,L3 were assumed to be shared cache among all threads. Based on the plot of the recorded data it is easy to see that as the number of threads increases, the instructions per cycle throughput decreases. It is however important to note that of all the configurations 64p64s (private instruction cache, shared data cache) is the most consistent and always performs higher than the baseline 64s64p (shared instruction cache, private data cache) configuration.

The next section of the paper compares SM to single chip multiprocessing (MP). Without going into all the detail, it is important to note that these metrics were biased toward MP. There are however two respects in which the SM results may be optimistic – “amount of time required to schedule instructions onto functional units, and the shared cache access time.” The result which I found to be most interesting is that the first few tests give both schemes an unlimited number of functional units. However, when that number is restricted, the SM scheme outperforms the MP scheme by nearly as much as the first test. Overall the results obtained show that SM outperforms MP in a variety of configurations because of the dynamic partitioning of functional units.

The final conclusions of the paper include that a SM architecture can: achieve 4 times the instruction throughput of a single-threaded wide superscalar with the same issue width, outperform fine-grain multithreading by a factor of 2, and outperform a MP with an equivalent configuration.