

Aaron Kobayashi

11/13/2003

Future Microarchitectural Problems

Assumptions:

One assumption made is that power will continue to be an issue. With a revolution in portable power sources, the power argument could be reduced and performance could become the only true concern in the future

Summary:

The microprocessors of today are very advanced machines capable of doing operations at amazing speeds. Even though this power is available, there is always a need for more computational power done in smaller, cheaper, faster processors. While the processors of today are amazing in their own rights, there are a few key roadblocks that keep us from realizing even more powerful implementations. Engineering is often a game of tradeoffs. Microarchitectural engineering is no different. Often times the tradeoff for performance is power and die size.

While total execution time of a program is the ultimate measure of a processor, IPC is a key factor in computing the total execution time. All other things being equal, increasing the IPC will result in a decrease in execution time of a processor. As a result, many techniques currently being researched focus on increasing IPC.

Instruction supply is the first piece of the IPC puzzle. Probably the most common technique to increase the supply of instructions is the use of branch prediction. Currently more sophisticated adaptive or hybrid predictors are being developed to increase the prediction accuracy thereby increasing the flow of instructions into the processor. Research shows that even a small increase in prediction can result in a significant increase in IPC. Prediction is just that, a guess. Consequently, some branches will always be mispredicted. To reduce misprediction penalties, one proposed system would maintain a cache of decoded instructions. This has the effect of reducing the pipeline length, greatly reducing the penalty incurred when stalling. A second idea known as eager execution, suggests that executing both paths of a difficult to predict branch will help reduce the prediction problem. Eager execution uses a confidence mechanism to determine the accuracy of a branch prediction. The major tradeoff here is IPC for power and complexity as executing both paths ensures that you throw out half of your work.

Another technique that increases instruction supply is to increase the number of fetched instructions per cycle. This approach comes at the price of multiporting and complex steering logic which increase power requirements. Trace caches and instruction fusion were proposed as a better alternative that provides high bandwidth with low latency. Traces are blocks of logically sequential instructions that are stored together in cache. Building traces does require a bit of overhead on the front end, but once built, they exhibit very fast fetching and execution. Taking this idea in a different direction, instruction fusion increases the instructions fetched per cycle by keeping the instructions together for as long as possible in the pipeline, much like carpooling.

Probably the most straightforward way to increase the performance of a processor is to increase the rate at which the execution engine can finish instructions. One way to reduce the execution time is to eliminate unnecessary instructions. Unified renaming attempts to do just this by using a value identity detector to determine if two future results will be equivalent. If so, all references to the later result are converted to use the earlier result, removing the need to execute the latter instructions. Another variation on this is instruction reuse. Instruction reuse is in my opinion one of the most interesting approaches to this problem. Instruction reuse attempts to trade off computation with

table lookup. The idea is to keep a history of executed instructions and their results. When the computer sees that segment of code again, it can skip the requested instructions by looking up the appropriate outputs based on the given inputs. While this is certainly very interesting, a lot of resources have already been placed in the super-scalar out of order processors we are familiar with today. With the current out of order execution, the main limiting factor is not the lack of resources but the data dependencies imposed by data flow among other instructions. Value prediction attempts to circumvent this problem by predicting the result of an incoming instruction based on previously computed results. While this seems like a very difficult goal to achieve, studies have shown that over 40% of the results can be predicted.

The third way to increase the performance of a processor is to increase its ability to access memory. The gap between microprocessor speeds and memory speeds is ever increasing. This widening gap causes more and more processor cycles to be wasted. One way to minimize this loss is to overlap the costs by using MLP. Memory Level Parallelism (MLP) overlaps memory requests by attempting to guess which portions of memory will be needed and attempts to prefetch them effectively overlapping the penalties of memory access. Another way to reduce this penalty is to put the DRAM right on the processor die. This memory is faster and has the added benefit of being more energy efficient than additional logic gates in a processor die.

While all of these solutions are intriguing, microarchitects must also consider the technological feasibility of their designs. With cell phones being the next computing platform, power requirements will become much more stringent. One way to save power is to dynamically reduce the voltage/frequency used by a processor. By turning down the clock speed and voltage during off peak usage, energy is saved. When a burst of processing power is necessary, the voltage is increased and peak performance is preserved. Another more aggressive idea involves shutting down circuits when they are not in use. This will save leakage current and keep power loss to a minimum.

Another technological problem is wire skew. As processor clocks increase to amazing speeds, electrical delay becomes an issue. As wires become longer, the maximum frequency possible decreases. One proposed way to combat this problem is to have a globally asynchronous locally synchronous clock. This would enable modularity in design, simplified interfacing, and a reduction of clock limitation. Related to clock speeds is the soft error problem. As technology is scaled down, hardware designs become increasingly susceptible to flipped bits. A soft error occurs when the amount of electrons collected by alpha particles and cosmic rays is enough to upset stored data. As processor technology scales down, care will have to be taken to insure that flipped bits will not adversely effect processor execution. One way to insure this is to employ ECC or Hamming codes to correct single-bit errors. A more general proposal is to build a less complex but highly parallel checker that will verify a group of results at a time.

While there are many techniques out there to increase performance and reduce power requirements, tradeoffs still have to be made. It is hard to say where microprocessors will be even 10 years from now, but one thing is sure; they will employ very different mechanisms than the processors of today.

References:

- R. Ronen, A. Mendelson, K. Lai, S.-L. Lu, F. Pollack, and J.P. Shen, "Coming Challenges in Microarchitecture and Architecture," *Proceedings of the IEEE*, Vol 89, No. 3 (March 2001), pp. 325-340.