Murray Crichton - s1232200
Parallel Architectures (PA)
Coursework 1 (Submission 2)
29/1/2016

Murray Crichton
Parallel Architectures (PA) - Coursework 1
Scalable Processors in the Billion-Transistor Era: IRAM

Description - Paper C:

In *Scalable Processors in the Billion-Transistor Era: IRAM*, the authors outline a theoretical CPU design with DRAM integrated on-die, and a heavy focus on vector calculations. Such a design would theoretically improve memory performance when compared to traditional cache-based systems in some applications, reduce power usage (removing power-intensive main memory accesses through long busses), and beat competing designs on a transistor to performance-increase ratio. The authors assume that memory access latency will continue to increase at a poor rate compared to CPU performance; that other processor designs memory requirements will have an increasing bottleneck effect; that designs will need to keep power dissipation to a minimum; that the entire memory of a computer could fit on the CPU die; that the vector programming model will become more mainstream; and that fabrication techniques will improve to allow DRAM to be added to a CPU die with relative ease.

Results - Paper C:

While the paper does not show any experimental results, it still shows a theoretical design for an IRAM processor (the “Berkley V-IRAM System”) and some likely performance and memory statistics to go along with it. The performance statistics proposed seem reasonable, and the high memory bandwidth (more comparable to a modern GPU than a CPU) seem excellent (as one would expect, from the intention of the design). The memory capacity proposed, however, is extremely low, at a mere 96 Mbytes.

Discussion - Paper C:

The design outlined in the paper was not a successful one. While I have no industry insider knowledge as to exactly why this would be the case, there are several problems with the proposed design and assumptions made that would prevent such an architecture from achieving mainstream adoption.

One assumption is that the 96 Mbytes of DRAM proposed would be enough for mobile applications (with future scaling allowing enough capacity for desktop and workstation applications). This is clearly not the case: due to various reasons modern computers (be they desktop workstations, mobile alternatives, or even hand-held devices) have significantly more memory than that proposed, as required by the applications and operating systems used. This could simply be a mis-prediction on the future of DRAM density, but in either case, the memory amount proposed is wholly insufficient, and the solution offered - having more memory in traditional off-chip DRAM banks - erases any advantages presented by the IRAM design in the first place. The modern solution to this problem is to have a system of layered caches, capable of providing low latency memory access, due to the high hit-rate of the cache (often over 95%) coupled with techniques including, for example, the exploitation of memory access locality.

Another issue here is the requirement for software to be written in a vector-processor compatible manner, which would have required, for example, a large shift in the software development industry, and research into new compilers. The authors mention this issue, and that techniques were already in development to create vector-based software, but it appears that these have not been adopted, at least for mass-market software. One reason for this could be the integration of GPU units within the

CPU die, capable of efficiently dealing with the same tasks that a vector-processor would handle, while having other benefits (not needing a discrete GPU, for when heavy graphics processing power is not warranted, or when lower power requirements are preferred).

It would appear, then, that the industry chose to combine GPU and CPU into a single die, instead of CPU and memory. This could be for a number of reasons: I would imagine the power dissipation (and therefore cooling requirements) of a GPU unit would be higher, and so placing both the CPU and GPU in one spot would allow for more efficient cooling in devices (generally mobile computers) with a single cooling fan, leaving the memory to be passively cooled, possible due to the relatively low power dissipation of conventional memory. Another reason could be the ease of fabrication - the fabrication techniques used to create memory and CPU/GPUs are incompatible, requiring an extra set of steps to add memory to a CPU die, as opposed to adding a GPU, which could be done during the fabrication of the CPU. With the cache technology described earlier, the combined CPU-GPU die would allow for efficient processing of vector (GPU) and non-vector (CPU) tasks, while maintaining low memory access latencies. This provides a well-rounded set of capabilities, with a relatively simple manufacturing process, perfect for mass-market consumption.

A final issue with placing DRAM memory close to the CPU, and having it as the first (or close to first) layer of the memory hierarchy, relates to the nature of DRAM as opposed to SRAM caches. DRAM, over time, needs to be refreshed, to retain the values stored. Ideally, one would want to clock the memory at the same rate as the CPU, to allow for low-latency memory access, similar to that found

when using an SRAM cache hierarchy. This, however, greatly increases the power dissipated, when a high CPU clock is used, potentially bottlenecking performance.

Description - Paper B:

In *A Single-Chip Multiprocessor* the authors outline architectures for a trio of billion-transistor processor designs. A “superscalar” design, a “simultaneous multithreading design” (SMT; similar to the superscalar design, but with more threading capabilities), and a “chip multiprocessor” (CMP) design. They outline some vital statistics for the designs - issue width; number of threads; and so forth, making assumptions of reasonable values for these properties with reference to then-modern chips.

The authors then describe an experiment using the three designs, and discuss the results. This experiment was carried out using a simulation, created with the goal of testing some predicted use-cases (by proxy via some selected programs intended to represent these cases, while being constant and therefore providing measurable speedup results), and benchmarking the speedup provided by the proposed architectures in each case. The assumptions made were that these programs truly represented the software of “tomorrow” in which the billion-transistor chips could exist, and that the simulation created would be a reasonable representation of the performance of such chips.

Results - Paper B:

The results provided in the paper, produced by running a series of programs on the three simulated architectures, demonstrated that the CMP design achieved the best overall performance, only worse at tasks with “moderate memory behaviour” and “no thread-level parallelism,” one category out of four. The authors predicted that, moving forward, these tasks would be less common use cases in general, with more generally parallel workloads tasks being exposed by general market interest (multimedia or running multiple programs at once, for example) or with advancements in compiling techniques. This, combined with the relatively minimal gain the other designs showed in this specific area further cemented CMT as the superior architecture.

Discussion - Paper B:

The architectures proposed in *A Single-Chip Multiprocessor* appear to be altogether reasonable, being theoretical continuations and expansions on designs of that era. With the billion-transistor chip now being a reality, we can see that the “chip multiprocessor” (CMP) design shown as being a “promising candidate” has indeed been the most successful of those outlined. In this, the prediction of the paper appears to have been correct. The assumptions made about the simplicity and scalability of the design, as well as the use-cases (multimedia; multi-program execution) were correct, and are part of the overwhelming commercial success of modern processors.

The results presented in the paper, I believe, are still valid; the example programs chosen are still a solid representation of various types of use-cases. The experimental simulation itself appears to have produced accurate results, and the arguments for favouring the CMP design are still compelling - the advantages and disadvantages discussed are still relevant to this day.