Parallel Architectures
Exercise 1

Stephen McGruer, s0840449

1 Superspeculative Microarchitecture for Beyond AD 2000

In [1] Lipasti and Shen explain their vision for a billion transistor microprocessor architecture; a 'superspeculative' processor that is able to speculate both control flow and data flow. This section provides an overview of their paper, an examination of their results, and a discussion on how accurate their vision has turned out to be.

1.1 Overview

The design presented by Lipasti and Shen in [1] is based on the idea that dataflow speculation can be used to find large amounts of instruction level parallelism (ILP), surmounting the 'ILP wall'. Lipasti and Shen propose a 'superspeculative' architecture, Superflow, which has powerful speculative abilities. They evaluate Superflow against a conventional superscalar architecture on nine benchmarks to determine which approach is superior.

The Superflow architecture relies on using a *weak consistency model* (WCM). In a WCM data dependencies between instructions can be ignored temporarily, as committed results are correct with respect to the dependencies. Using a WCM allows Superflow to apply speculation to improve three properties: instruction flow, which is the rate at which instructions are issued; register dataflow, which is the rate at which values in registers are computed; and memory dataflow, which is the rate at which values are read and written to memory.

To improve instruction flow, Superflow applies known branch prediction techniques, combined with a trace cache [2]. Register dataflow is increased by using *value speculation*, where values stored in registers are speculated before the instruction to calculate them has been executed. In particular, Superflow utilises: *source operand value predication* [3], where instruction operands are predicted to allow early execution; *dependence prediction* [3], which allows dependencies between instructions to be predicted in order for other speculation to proceed; and *value stride prediction* [4], which extends source operand prediction to predict values that stride by constant amounts. Finally, the memory dataflow is increased by applying *load value prediction* [5] to predict the result of a load instruction. The load value prediction is augmented using the previously described dependence prediction to find linked store/load pairs.

Lipasti and Shen justify their chosen approach based on two claims: that parallelism is required to extract more performance from processors, and that the best way to find such parallelism is to circumvent the ILP wall using dataflow prediction. They argue that other approaches for extracting parallelism are not viable as they require programmers to transition to thread or process-level parallelism (TLP, PLP), which Lipasti and Shen argue is more effort than implementing value speculation.

1.2 Results

The results presented by Lipasti and Shen show impressive performance from their architecture. Superflow achieves an average of 9 instructions per cycle (IPC), over double the 4 IPC managed by the comparison superscalar architecture. However, the value of 9 IPC assumes an architecture with a perfect cache and unlimited cache ports: more realistic configurations achieve averages between 6 and 8 IPC. Lipasti and Shen also also present sub-divided results showing the impact of speculating each type of flow. These results indicate that speculating register dataflow is most important, as it provides at least 80% of

the speedup. However it is unclear if the results of register dataflow rely on previous speculations, as the speculation types are applied in only one order (instruction flow speculation only, then instruction flow and memory dataflow speculation, and finally speculation for all three flows).

1.3 Discussion

None of the value prediction ideas put forward by Lipasti and Shen have ever made it into commercial processors. Branch prediction is used in almost all modern processors, but was already common in processors when [1] was written. Interestingly the trace cache advocated by Lipasti and Shen has only been used in a single commercial processor, the Pentium 4 [6]. Later Intel processors use similar mechanisms, but only for caching loop instructions rather than general control flow.

There are two possible explanations for why 'superspeculative' architectures never became mainstream. The first is complexity. It was far easier for manufacturers to implement multiprocessor designs (see section 2) than the speculative design proposed by Lipasti and Shen. Academic interest in value prediction has continued sporadically (i.e. [7, 8, 9, 10]), but has yet to make its way into commercial space. The other possible reason is the importance of power usage. The value predictors used by Lipasti and Shen have been found to be very energy inefficient [11, 12] and so are unsuitable for modern architectures.

The central flaw in Lipasti and Shen's reasoning is their assumption that multiprocessor architectures would not work. They argued that programmers would not transition well to TLP and PLP and so multiprocessor architectures would be rejected. However while the transition to TLP-suitable programs has been slow [13], multiprocessors have still flourished (see section 2.3) and so there has been little reason to implement value speculation.

The future of value speculation in computer architecture is unclear. Programmers continue to find it difficult to create TLP-suitable programs. This has caused the number

of cores on consumer processors to effectively stagnate between the quad (4) and octa (8) level. Circumventing the IPC wall could therefore be, once more, a valid strategy. However, implementing value speculation would require a non-trivial amount of work, both to tackle the power issues mentioned before and because the *sequential consistency* model used by most multiprocessor architectures does not allow for value speculation [14]. The experiments presented by Lipasti and Shen would also have to be re-designed, to gauge the impact of value speculation in a multi-core environment.

2 A Single-Chip Multiprocessor

In [15] Hammond et al. explain their vision for a billion transistor microprocessor architecture; multiple single-threaded processor cores placed on a single chip. This section provides an overview of their paper, an examination of their results, and a discussion on how accurate their vision has turned out to be.

2.1 Overview

Two possible designs are suggested by Hammond et al. in [15]. *Chip multi-processor* (CMP) architectures utilise multiple single-threaded processor cores on a single silicon chip. Although the cores share an L2 cache and are able to communicate via an on-chip interconnect, each core generally executes independently. In contrast, *Simultaneous Multithreading* (SMT) architectures use a single processor core, which is able to issue and execute instructions from multiple threads simultaneously without context switching. Simulated versions of CMP and SMT are compared (alongside a traditional superscalar approach) on four benchmarks to determine which design is superior.

Hammond et al. justify their designs based primarily on two claims: that future archi-

tectures must avoid increasing complexity, and that the best way to use a billion transistors is to find and exploit parallelism in programs. They argue that the complexity-to-benefit ratio of extracting further single-thread parallelism (e.g. ILP) is too high to be feasible. Instead, Hammond et al. contend that supporting TLP and PLP is a suitable approach due to the (expected) impact of smarter compilers, increasing use of multimedia applications, and multiprocessor-aware operating systems. They justify software support for TLP and PLP (i.e. forcing the programmer to specify the parallelism) over hardware support by arguing that hardware support would be too complex and thus slow.

2.2 Results

The results presented by Hammond et al. show impressive performance from both of their designs, with the CMP architecture performing better overall. In three of the four benchmarks, where TLP or PLP was available, the CMP architecture outperformed SMT by 10-20% and outperformed the superscalar processor by 150-400%. In a fourth benchmark, where there was little available parallelism of any kind, both the SMT and superscalar architectures outperformed CMP by 43%.

The analysis carried out by Hammond et al. on their own results concluded that the performance difference was due to the increased overall number of issue slots in CMP (2 per processor x 8 processors = 16 issue slots, compared to the 12 in SMT) and due to the per-processor caches in the CMP architecture. The per-processor caches were noted as particularly important, as the unified cache in SMT performed poorly when dealing with many memory-heavy threads, resulting in many cache misses.

2.3 Discussion

In general, the CMP architecture put forward by Hammond et al. closely matches the current state of microprocessor architectures. Whether their reasons for supporting the

architecture were correct is not as definite.

The first non-embedded CMP architecture, the IBM Power4, was developed based on a desire to avoid complexity and the belief that surmounting the 'ILP wall' was cost-ineffective [16] - two of the core reasons given by Hammond et al. However, the Power4 had only 174 *million* transistors - far from a billion transistor architecture. By the time that the first billion transistor processor appeared (the Itanium 2 'Montecito' [17]) a third factor had emerged: power. This was not considered by Hammond et al. and is a major supporting factor for CMP. CMP architectures require less power than equivalent monolithic processors, and also provide a way around the limit on clock frequencies caused by the power wall.

Hammond et al. also expected that the cores in a CMP architecture would be simple. Instead, the complexity of CMP cores has continued to grow. Modern architectures such as the Intel 'Sandy Bridge' (1.1 billion transistors [18]) contain cores that are 4-issue rather than 2-issue as Hammond et al. predicted, and which are *each* 2-thread SMT cores. Indeed, the fact that each core in a CMP architecture could also be SMT[1] was not considered by Hammond et al.

A final flaw in the reasons given by Hammond et al. was their expectation that future programs would have high levels of TLP. It has proven difficult to educate programmers to create concurrent programs [13], and most compilers are still unable to automatically extract parallelism from a program. Despite this CMP architectures achieve good performance overall, likely due to the ability of operating systems to utilise multiple cores to execute different programs simultaneously (which Hammond et al. did predict!)

[1]This was first implemented in the IBM Power5 in 2004 [19], which had 276 million transistors - still far below one billion!

References

[1] M. Lipasti and J. Shen, "Superspeculative microarchitecture for beyond AD 2000," *Computer*, vol. 30, no. 9, pp. 59–66, 1997.

[2] E. Rotenberg, S. Bennett, and J. Smith, "Trace cache: a low latency approach to high bandwidth instruction fetching," in *Proceedings of the 29th annual ACM/IEEE international symposium on Microarchitecture*, pp. 24–35, IEEE Computer Society, 1996.

[3] M. H. Lipasti and J. P. Shen, "The performance potential of value and dependence prediction," in *in EUROPAR-97*, pp. 1043–1052, Springer-Verlag, 1997.

[4] F. Gabbay, "Speculative execution based on value prediction," tech. rep., EE Department TR 1080, Technion - Israel Institue of Technology, 1996.

[5] M. Lipasti, C. Wilkerson, and J. Shen, "Value locality and load value prediction," *ACM SIGOPS Operating Systems Review*, vol. 30, no. 5, pp. 138–147, 1996.

[6] G. Hinton, D. Sager, M. Upton, D. Boggs, *et al.*, "The microarchitecture of the Pentium® 4 processor," in *Intel Technology Journal*, Citeseer, 2001.

[7] B. Goeman, H. Vandierendonck, and K. De Bosschere, "Differential FCM: Increasing value prediction accuracy by improving table usage efficiency," in *High-Performance Computer Architecture, 2001. HPCA. The Seventh International Symposium on*, pp. 207–216, IEEE, 2001.

[8] J. Yang and R. Gupta, "Frequent value locality and its applications," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 1, no. 1, pp. 79–105, 2002.

[9] T. Sato and I. Arita, "Low-cost value predictors using frequent value locality," in *High Performance Computing*, pp. 309–312, Springer, 2006.

[10] P. Prabhu, G. Ramalingam, and K. Vaswani, "Safe programmable speculative parallelism," in *ACM Sigplan Notices*, vol. 45, pp. 50–61, ACM, 2010.

[11] R. Moreno, L. Pinuel, S. del Pino, and F. Tirado, "A power perspective of value speculation for superscalar microprocessors," in *Computer Design, 2000. Proceedings. 2000 International Conference on*, pp. 147–154, IEEE, 2000.

[12] N. Sam and M. Burtscher, "On the energy-efficiency of speculative hardware," in *Proceedings of the 2nd conference on Computing frontiers*, pp. 361–370, ACM, 2005.

[13] G. Blake, R. Dreslinski, T. Mudge, and K. Flautner, "Evolution of thread-level parallelism in desktop applications," *ACM SIGARCH Computer Architecture News*, vol. 38, no. 3, pp. 302–313, 2010.

[14] M. Martin, D. Sorin, H. Cain, M. Hill, and M. Lipasti, "Correctly implementing value prediction in microprocessors that support multithreading or multiprocessing," in *Proceedings of the 34th annual ACM/IEEE international symposium on Microarchitecture*, pp. 328–337, IEEE Computer Society, 2001.

[15] L. Hammond, B. Nayfeh, and K. Olukotun, "A single-chip multiprocessor," *Computer*, vol. 30, no. 9, pp. 79–85, 1997.

[16] K. Diefendorff, "Power4 focuses on memory bandwidth," *Microprocessor Report*, vol. 13, no. 13, pp. 1–8, 1999.

[17] C. McNairy and R. Bhatia, "Montecito: A dual-core, dual-thread Itanium processor," *Micro, IEEE*, vol. 25, no. 2, pp. 10–20, 2005.

[18] E. Rotem, A. Naveh, D. Rajwan, A. Ananthakrishnan, and E. Weissmann, "Power-management architecture of the Intel microarchitecture code-named Sandy Bridge," *Micro, IEEE*, vol. 32, no. 2, pp. 20–27, 2012.

[19] R. Kalla, B. Sinharoy, and J. Tendler, "IBM Power5 chip: A dual-core multithreaded processor," *Micro, IEEE*, vol. 24, no. 2, pp. 40–47, 2004.