Bridging the Processor-Memory Performance Gap with 3D IC Technology

Christianto C. Liu, Ilya Ganusov, Martin Burtscher, and Sandip Tiwari
Cornell University

Editor's note:
Bringing the memory close to the processor is obviously a good idea for reducing latency, but the memory is not a monolith: It includes submodules, and their arrangement determines the effectiveness of using 3D technology. The authors look at various scenarios for arranging the CPU relative to the memory modules and provide a quantitative comparison.
—Sachin Sapatnekar, University of Minnesota

MICROPROCESSOR PERFORMANCE has been improving at roughly 60% per year. Memory access times, however, have improved by less than 10% per year.[1] The resulting gap between logic and memory performance has forced microprocessor designs toward complex and power-hungry architectures that support out-of-order and speculative execution. Moreover, processors have been designed with increasingly large cache hierarchies to hide main memory latency. This article examines how 3D IC technology can improve interactions between the processor and memory.

Three-dimensional ICs, as shown in Figure 1, consist of planar device layers stacked one atop another and interconnected by short, vertical wires.[2] With the end of conventional device scaling in sight, 3D ICs let scaling continue by shifting the focus from device scaling to circuit and system scaling. By stacking multiple planar device layers with short, vertical separations, designers can build systems that exhibit lower interconnect latencies; higher packing densities of logic, memory, and other circuits; and heterogeneous integration of circuits of different materials (for example, CMOS, SiGe, and III-V), signals (digital, analog, and RF), or technologies (microelectromechanical systems, optics, and so forth). In addition to lower wire latency through shorter paths, 3D IC technology offers unparalleled bandwidth. These short, vertical, on-chip connections are not pin limited, as in 2D designs that must go off chip; they can be anywhere on a 3D chip. For maximum efficiency, 3D designs must take advantage of the increased device density and the low-latency, high-bandwidth interconnections.

How 3D technology bridges the processor-memory performance gap

Unlike a conventional 2D chip, on which logic and memory units reside at opposite ends, a 3D chip can have logic and memory stacked together to shorten the critical path. More importantly, bringing main memory onto the chip can significantly reduce latency. Bandwidth between the CPU and DRAM improves dramatically with on-chip buses that can fetch hundreds to thousands of bits at once. Power consumption also drops because the large capacitive loads due to off-chip accesses are removed. Many commercial processors, including IBM's Blue Gene/L processor and ATI's graphics processor for the new Xbox 360, have exhibited the performance benefits of embedded DRAMs.

Considering the densities of current technology puts things in perspective. There are two types of embedded DRAM. Dense DRAM has 16 to 20 times the capacity of SRAM and is designed for high density, low cost, and low leakage.[3] Alternatively, designers can build DRAM using logic technology, which is more expensive but allows for higher performance. The density of logic-based DRAM ranges from four to eight times that of SRAM. *International Technology Roadmap for Semiconductors* data (http://public.itrs.net) for 90-nm technology shows

0740-7475/05/$20.00 © 2005 IEEE Copublished by the IEEE CS and the IEEE CASS

that one 2-cm^2 chip can accommodate about 256 Mbytes of dense DRAM. About the same number of Mbytes in logic-based DRAM requires five chips. For comparison, only about 64 Mbytes of SRAM can fit on four chips. With density and chip area in mind, we consider only cache sizes up to 64 Mbytes and embedded main memory sizes up to 256 Mbytes. None of the benchmark programs in this study has a memory footprint exceeding 200 Mbytes. For applications that use more memory, it's possible to envision a nonuniform memory access architecture in which the embedded main memory serves as local memory for the CPU.

Researchers have used textbook memory latency models to characterize the performance benefits of stacking caches and main memory atop the processor.[4] However, these results don't apply to today's sophisticated microprocessors. In this study, we conduct simulations on representative application programs to evaluate the performance of stacked memory for a superscalar, out-of-order processor.

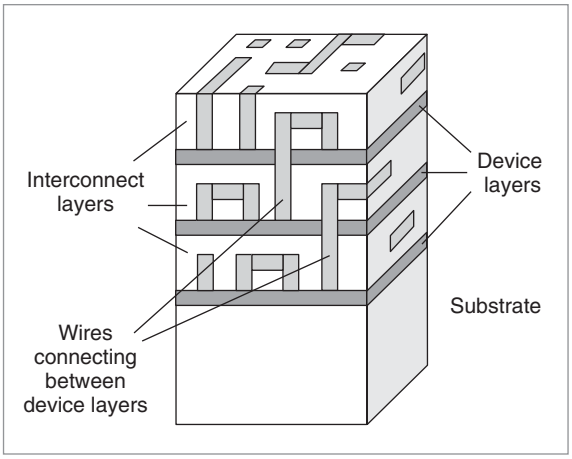

Figure 1. 3D ICs consist of planar device layers separated by silicon dioxide or other insulating materials and interconnected by short, vertical wires.

Methodology

We analyze the performance improvements of 3D IC technology using an extended version of the SimpleScalar Version 4.0 simulator.[5] The baseline 2D processor core represents current technology (3-GHz CPU; 750-MHz memory; and 64-Kbyte L1-instruction, 64-Kbyte L1-data, and 1-Mbyte L2 caches). We provide the CPU and memory frequencies only for reference; the simulator accepts inputs in units of CPU clock cycles and doesn't actually require frequency values. Table 1 shows details of the simulated processor. We use all 12 integer and 10 of the 14 floating-point programs from the SPEC CPU2000 benchmark suite (http://www.spec.org/osg/cpu2000), omitting the four Fortran 90 programs for lack of a compiler. The

Table 1. Baseline 2D processor configuration.

Feature	Specification
Baseline processor	
Fetch/dispatch/commit width	4/4/4 instructions/cycle
Instruction window/reorder buffer/load-store queue	64/128/64 entries
Integer/floating-point registers	208
Load-store/integer/floating-point units	2/4/2
Execution latencies	Similar to those of the Alpha 21264
Branch predictor	8-K entry bimodal/gshare hybrid
Return address stack	16 entries
Branch target buffer	2,048 sets, four-way set associativity
Minimum branch misprediction penalty	16 cycles
Baseline processor's memory subsystem	
Cache sizes	64-Kbyte L1 instruction, 64-Kbyte L1 data, 1-Mbyte L2
Cache associativity	Two-way L1, four-way L2
Cache (load-to-use) latencies	Three-cycle L1, 11-cycle L2
Cache line sizes	64-byte L1, 64-byte L2
Miss status holding registers	64-entry L1, 64-entry L2
Main memory latency	300 cycles + cache access time
Hardware stream prefetcher	Between L2 and main memory;16 streams; maximum prefetch distance: 16 strides

programs run with the SPEC-provided reference inputs. We use the SimPoint toolset to identify representative simulation points.[6] We simulate each program for 500 million instructions after fast-forwarding past the number of instructions determined by SimPoint.

We set out to investigate various scenarios:

- Examine the potential benefits of stacking an existing L2 cache above the CPU.
- Examine the benefits of stacking main memory atop the CPU. This study includes on-chip main memory fabricated using standard DRAM technology and that made from the faster logic technology.
- As an alternative to stacking main memory atop the CPU, expand the existing cache hierarchy. This includes increasing the L2 cache size and adding an L3 cache.
- Consider the impact of stream prefetching[7] on different cache organizations.
- Combine the optimized cache hierarchy with stream prefetching and on-chip main memory.

To quantify the performance benefits of on-chip main memory, we must examine the different components that contribute to the latency. In general, main memory latency includes the time to request data from the DRAM core (memory controller latency) and the time to retrieve the data from the DRAM core (core access latency). Other components of memory latency, such as the delay through the bus interface unit, are not covered here. Core access latency includes the time needed to decode the address signals, activate the word line, select and amplify the bit lines, and transfer the data out of the DRAM chip. Bringing the DRAM onto the CPU chip lets us significantly reduce the core access time.[3] With on-chip buses, data moves between the DRAM and the CPU at far higher frequencies than are possible over long, off-chip wires with large capacitances. Moreover, on-chip DRAM obviates address multiplexing and removes the off-chip drivers and receivers. Although it's still possible to fabricate the DRAM core using DRAM technology, designers can use the faster logic technology to build the peripheral circuitry, offering further speed improvements.

Memory controller latency includes the time needed to translate from physical addresses to memory addresses as well as the time to schedule the memory request, which means converting memory transactions to command sequences and queuing them. The DRAM core speed limits the memory controller's speed. In other words, the memory controller needn't schedule memory accesses much faster than the rate at which the DRAM core returns data. However, with the faster core speed that results from bringing the DRAM on chip, we can also run the memory controller faster. Based on the estimates from an IBM process, bringing main memory onto the CPU chip reduces core latency by roughly 60%; latency would further decrease with on-chip DRAM made using logic technology.[3] This should let us run the memory controller at twice its original speed.

With these ideas in mind, let's examine different scenarios for main memory. We assume that the baseline 2D processor has an average main memory latency of 300 cycles. Although this value can be higher or lower, we are concerned only with trends and not absolute values. We assume that half of the 300 cycles are due to the memory controller latency and half to the DRAM core access latency. Using the estimated access times of an IBM process, we first analyze the performance of an on-chip DRAM macro fabricated with standard DRAM technology. Then we assume that the on-chip DRAM is made using the faster logic technology. The next two cases examine on-chip DRAM with memory controller latency improved by 50% because of the faster core latency. The various memory latencies (in CPU cycles) appear in Table 2.

Cacti Version 3.2, an updated version of Cacti Version 3.0, determines the L2 cache access latencies.[8] Finding the optimum cache access times involves varying the number of cache banks, the word line and bit line configurations, and the cache line size. We fixed the associativity to four ways in all cases. Cache latencies appear in Table 2. We have assumed uniform cache access. In reality, this is a worst-case scenario because large caches are likely to require nonuniform access.[9] For instance, we can design an on-chip network to access different parts of a large cache with different latencies, depending on the number of data hops through the network.

Stacking existing L2 cache and main memory

There are two possibilities for improving the baseline processor. The first is to stack the L2 cache atop the CPU. Unfortunately, the benefit of stacking a standard L2 cache (with just a few megabytes) is small because proper floorplanning would already have placed the L2 strategically to minimize critical paths (for example, to the L1 cache). Also, even if 3D does reduce L2 cache latency, the reduction is at most 1 to 2 cycles out of the roughly

Table 2. Changes to the baseline configuration. The top portion shows the reduction in main memory latency (in CPU cycles). The bottom portion describes the different cache configurations.

Feature	Specification
Latency of on-chip main memory	
Standard memory controller	
Dense DRAM	208 cycles + cache access time
Logic-based DRAM	179 cycles + cache access time
Improved memory controller	50% of original controller latency
Dense DRAM	133 cycles + cache access time
Logic-based DRAM	104 cycles + cache access time
Cache configurations	
Cache sizes	2, 4, 8, 16, 32, and 64 Mbytes
Cache associativity	Four way (fixed)
Number of banks	8 banks for 2 to 32 Mbytes, 4 banks for 64 Mbytes
Cache (not load-to-use) latencies	Assume uniform access (worst case) 9, 13, 18, 31, 49, 86 cycles
Cache line sizes	128 bytes for 2 Mbytes, 256 bytes for 4 to 64 Mbytes

10-to-15-cycle load-to-use latency typical of today's L2 caches. Suppose that with better floorplanning in a 3D design, we reduce L2 latency (hypothetically) by 2 cycles. This reduction produces little performance improvement, as Figure 2 illustrates. In fact, an average speedup over the baseline processor of 0.7% and 0.4% for integer and floating-point programs, respectively, is negligible.

On the other hand, stacking main memory (instead of the existing L2 cache) atop the CPU provides a large boost in performance. This is especially true for certain benchmark programs like mcf (a combinatorial optimization program used for vehicle scheduling) that have numerous misses from the L2. In Figure 2, we consider four scenarios of on-chip main memory. First, we examine standard DRAM technology with a bus width of 8 bytes, which is consistent with a single channel of double-data-rate synchronous DRAM (SDRAM) in use today. The average speedup is 13% for integer and 25% for floating-point programs. We also consider the same DRAM with a much wider bus (64 bytes). This width is possible, given that buses now remain completely on chip. The relative improvement of a 64-byte bus over an 8-byte bus is only 1% to 2%. The primary reason for the small improvement is because the latency due to trailing-edge data (data sent from the memory after the initial burst of data is sent) is only 7 cycles, compared with the 219-cycle latency due to the rising edge (initial burst). This is only about a 3% difference in latency. However, a 3% increase in latency translates to less than a 3% decrease in performance, mainly because the processor is waiting for memory only for a fraction of the total execution time.

Thus far, we've assumed that despite the reductions in core latency, memory controller latency remains the same as that of standard off-chip DRAMs. However, as mentioned earlier, lower memory controller latency is likely to accompany lower core latency. Integrating the memory controller on chip and improving controller design can greatly reduce memory latency, as AMD's Opteron processor shows (http://www.amd.com/us-en/assets/content_type/DownloadableAssets/Hammer_in_Context_-_Fred_Weber.pdf). Assuming a 50% reduction in memory controller latency, the speedup of the dense DRAM macro over the same macro with no memory controller improvement is 12% and 27%, respectively, for integer and floating-point programs. This illustrates the importance of memory controller optimization.

Switching to logic technology provides an additional 6% and 15% boost in performance for integer and floating-point programs over the dense DRAM macro. This is because designers can use the faster logic technology for both the DRAM cells and the peripheral circuits, such as decoders and word line drivers. However, given the density issue of implementing main memory with logic technology (256 Mbytes implemented on five chips), pursuing this option is unlikely.

For both the integer and the floating-point programs, the mean performance gain is still far less than that of the perfect L2 cache case, which represents a hypothet-

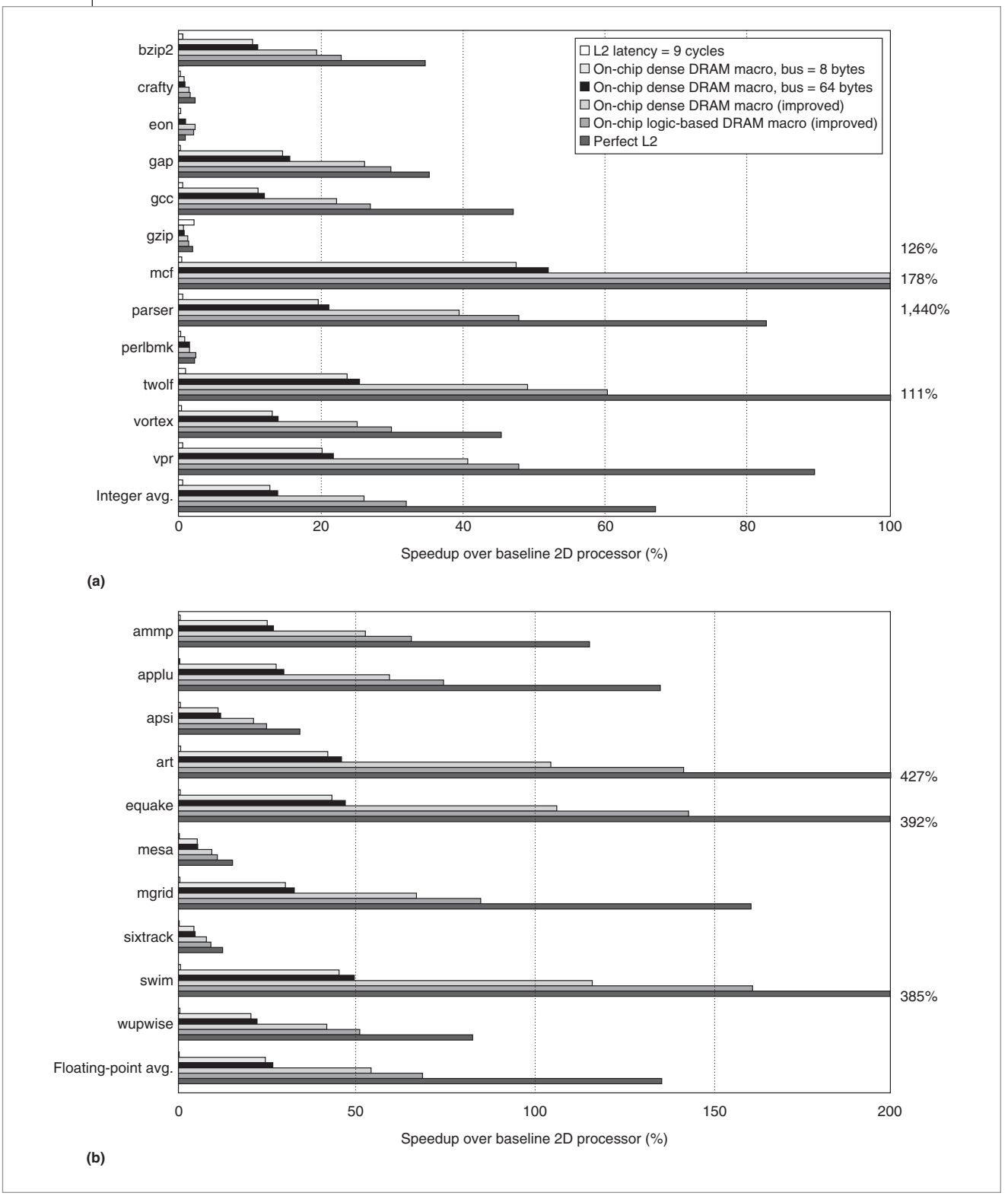

Figure 2. Benchmark performance of 3D processors with a hypothetical 2-cycle reduction in L2 hit latency: integer (a) and floating-point (b). The figure shows four possible scenarios of stacked on-chip main memory with different memory bus widths and compares both dense DRAM and logic-based DRAM macros. "Improved" indicates a memory controller latency improvement of 50%. We also plot the perfect L2 case, in which all accesses hit in the L2.

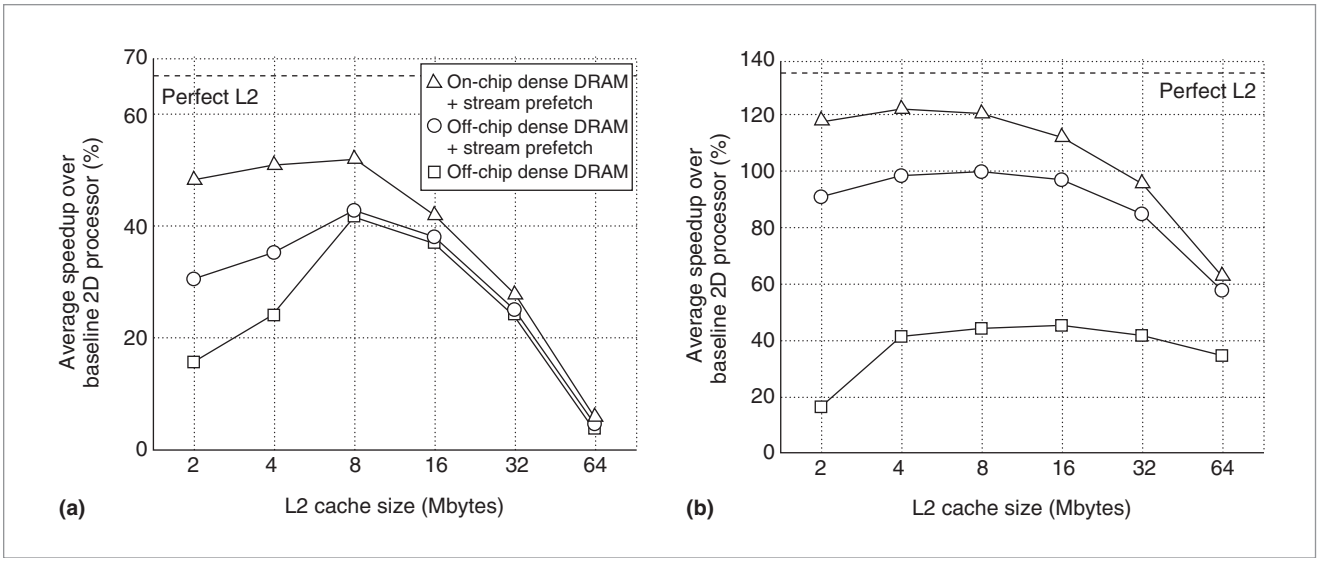

Figure 3. Average performance with varying L2 cache sizes: integer (a) and floating-point (b). The straight dashed line near the top illustrates the ideal case of a perfect L2 cache, in which all accesses hit in the L2.

ical configuration in which all accesses hit in the L2 cache (no accesses to main memory). This suggests that perhaps a different cache design would work better for 3D technology. In the rest of the article, on-chip main memory refers to the dense DRAM macro with improved memory controller (50% of the original latency). We choose this DRAM configuration because it offers the best trade-off between density, cost, and speed.

Expanding the L2 cache

With the larger circuit densities that 3D IC technology offers, designers can expand the cache hierarchy to migrate data closer to the processor. The baseline 2D processor has a 1-Mbyte L2 cache. With 3D, it's possible to enlarge the L2 cache, which might provide additional benefits despite the accompanying increase in access time. We investigated the processor performance for various L2 cache sizes up to 64 Mbytes, with latencies listed in Table 2.

The curves for the off-chip dense DRAM in Figure 3 show the average speedup over the SPEC benchmark suite. For the integer programs, the peak performance is clearly at 8 Mbytes. This illustrates the trade-off between fitting the working set into the cache (offering better performance) and the increased access latency for larger caches (offering poorer performance). With the integer programs, the average working set of 4 to 16 Mbytes works well with small caches. With floating-point programs, however, performance peaks at 16 Mbytes because of the larger working sets.

Implementing stream prefetching

It would be useful to take advantage of large-cache availability by observing program behavior and prefetching useful data into the cache. This is the role of the hardware stream prefetcher. We simulated an aggressive hardware stream prefetcher between the L2 cache and main memory.[7] (Later, when we add an L3 cache, the prefetcher prefetches data to both the L2 and L3 caches from main memory.) The stream prefetcher tracks the history of the last 16 miss addresses in a global history table, detects arbitrary-sized strides, and allocates a stream after a particular stride has been observed twice. It can simultaneously track 16 independent streams and prefetch up to 16 strides ahead of the processor's data consumption. Prefetches go directly into the L2 cache and are tagged. If the processor later requests the prefetched cache line, the prefetcher issues a request for the next element of the corresponding stream.

Adding stream prefetching significantly improves the performance of the integer programs with small caches. (Compare the curves for off-chip dense DRAM with and without stream prefetching in Figure 3.) With larger caches, prefetching provides little gain, because most of the active working set already fits in the cache. Floating-point programs, which have larger working sets and tend to exhibit more streaming-data access patterns, show a substantial performance gain for all cache sizes up to 64 Mbytes. However, performance begins to degrade at larger sizes because of the increased cache latency. With stream prefetching, a processor with a

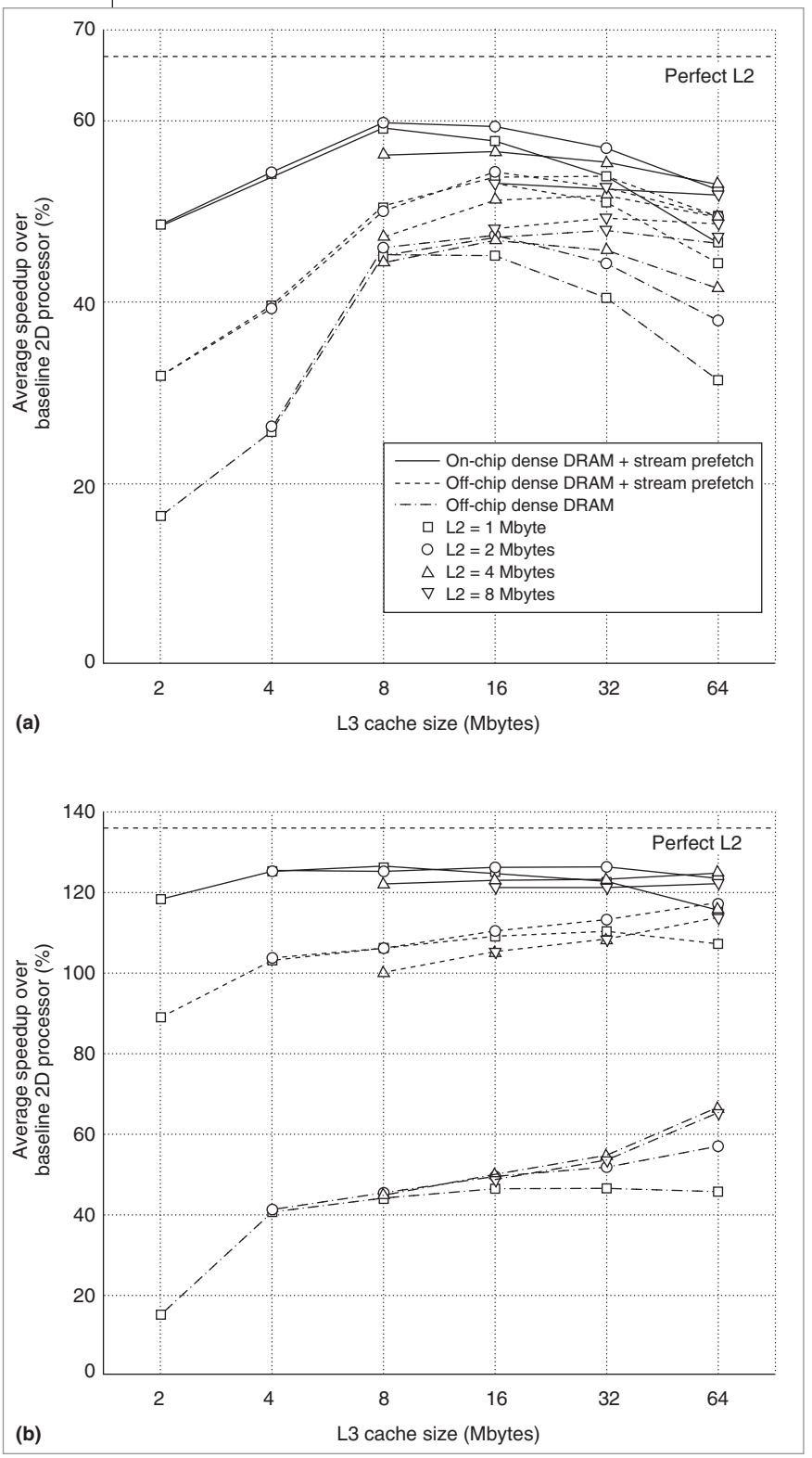

Figure 4. Performance with varying L2 and L3 cache sizes: integer (a) and floating-point (b). The straight dashed line near the top illustrates the performance of the ideal case of a perfect L2 cache, in which all accesses hit in the L2.

larger L2 cache and off-chip main memory (the middle curves in Figure 3) can perform better than the baseline processor with on-chip memory (Figure 2).

We can obtain an even greater performance gain by combining a larger L2 cache, stream prefetching, and on-chip main memory. Notice that with floating-point programs, the reduced memory latency provided by bringing memory on chip means that a smaller L2 cache (4 Mbytes) is needed to obtain peak performance.

Deepening the cache hierarchy

A larger L2 cache, though desirable, increases the access latency, which affects all data in a uniformly accessed L2 cache. One alternative is to use a nonuniform cache access scheme.[9] Another is to deepen the cache hierarchy by adding an L3 cache. Figure 4 examines different L2-L3 combinations and shows the performance relative to the baseline 2D processor. As in the previous figure, we compare the performance of off-chip dense DRAM (dot-dashed lines), off-chip dense DRAM with prefetching (dashed lines), and on-chip dense DRAM with prefetching (solid lines).

Combining a small but fast L2 cache with a larger but slower L3 cache generally produces much better performance than with a single large L2 cache. We observe the same trends as before: The use of stream prefetching and on-chip main memory reduces the cache sizes required to achieve peak performance. Note that obtaining optimum performance for integer and floating-point programs on the system with off-chip DRAM requires dramatically different cache organizations. For instance, a processor with off-chip DRAM and prefetching requires a 16-Mbyte L3 cache to work best with integer applications but a 64-Mbyte L3 cache for floating-point applications. With on-chip DRAM and prefetching (solid lines in Figure 4), a processor with

1-Mbyte L2 and 8-Mbyte L3 caches can provide nearly optimal performance for both integer and floating-point applications.

Combining better caches with faster DRAMs

Working in 3D makes it easy to combine different types of technology because each one can be fabricated on a separate substrate before wafer or die stacking. Therefore, it's possible to stack both caches and main memory on separate layers. Figures 3 and 4 show performance with a combination of large cache sizes and an on-chip DRAM macro (using dense DRAM technology) for main memory. On-chip DRAM as main memory delivers an additional improvement over off-chip DRAM. This is true for both the large L2 (Figure 3) and the L2 and L3 combinations (Figure 4).

It might appear at first that off-chip DRAM with stream prefetching can achieve the speedups of on-chip main memory with stream prefetching. However, the speedup with off-chip DRAM generally occurs with larger cache sizes. For example, on-chip main memory with prefetching delivers an average speedup of 126% for floating-point programs when L2 = 1 Mbyte and L3 = 8 Mbytes. Off-chip DRAM with prefetching provides a speedup of 117%, but here L2 = 2 Mbytes and L3 = 64 Mbytes. As mentioned earlier, a 64-Mbyte cache requires a silicon area of up to four chips. So even though off-chip DRAM with prefetching can achieve good speedups, it requires an unreasonably large cache. With L2 = 1 Mbyte and L3 = 8 Mbytes, off-chip DRAM with prefetching offers only a 106% speedup, 20% less than that of on-chip memory. The performance for integer programs is about 9% poorer than for the on-chip counterpart with L2 = 1 Mbyte and L3 = 8 Mbytes.

With on-chip main memory, stream prefetching, and a cache combination of L2 = 1 Mbyte and L3 = 8 Mbytes, the speedup over a baseline 2D processor is about 59% and 126% for the integer and floating-point programs, respectively. These speedup values are very close to the ideal values of 67% and 136% for a perfect L2 cache. In other words, the addition of on-chip memory provided the extra boost to achieve performance of only 8% and 10% below the perfect L2 cases. Small L2 and L3 cache combinations permit these large speedups because of the significant reduction in main memory latency. Designers can implement these small L2 and L3 caches within the CPU die, followed by wafer- or chip-stacking of a DRAM main memory. This simple two-layer processor-memory design can achieve near-ideal performance.

Our work examines the performance of a single-core, single-threaded processor under representative workloads. We have shown that reducing memory latency by bringing main memory on chip gives us near-perfect performance. As the industry moves toward multicore, multithreaded, and multimedia applications, there is an increasing drive not only for low memory latency but also for high memory bandwidth. Three-dimensional IC technology can provide the much needed bandwidth without the cost, design complexity, and power issues associated with a large number of off-chip pins. The principal challenge remains the demonstration of a highly manufacturable 3D IC technology with high yield and low cost. ■

Acknowledgments

This research project is supported by DARPA. Christianto C. Liu is also supported by an AMD/SRC graduate fellowship. We are indebted to the following people for many invaluable discussions: John Barth, Norman Jouppi, Rajit Manohar, and Richard Matick, with special thanks to Sally McKee and David Wang. We also thank the reviewers for their thorough comments.

■ References

1. D.A. Patterson and J.L. Hennessy, *Computer Architecture: A Quantitative Approach*, 2nd ed., Morgan Kaufmann, 1996.
2. K. Banerjee et al., "3-D ICs: A Novel Chip Design for Improving Deep-Submicrometer Interconnect Performance and Systems-on-Chip Integration," *Proc. IEEE*, vol. 89, no. 5, May 2001, pp. 602-633.
3. R.E. Matick and S.E. Schuster, "Logic-Based eDRAM: Origins and Rationale for Use," *IBM J. Research and Development*, vol. 49, no. 1, Jan. 2005, pp.145-165.
4. M.B. Kleiner et al., "Performance Improvement of the Memory Hierarchy of RISC-Systems by Application of 3-D Technology," *IEEE Trans. Components, Packaging, and Manufacturing Technology, Part B: Advanced Packaging*, vol. 19, no. 4, Nov. 1996, pp. 709-718.
5. E. Larson, S. Chatterjee, and T. Austin, "MASE: A Novel Infrastructure for Detailed Microarchitectural Modeling," *Proc. IEEE Int'l Symp. Performance Analysis of Systems and Software* (ISPASS 01), IEEE CS Press, 2001, pp. 1-9.
6. T. Sherwood et al., "Automatically Characterizing Large Scale Program Behavior," *Proc. 10th Int'l Conf. Architectural Support for Programming Languages and Operating Systems* (ASPLOS 02), ACM Press, 2002, pp. 45-57.

7. N.P. Jouppi, "Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers," *Proc. 17th Ann. Int'l Symp. Computer Architecture* (ISCA 90), IEEE CS Press, 1990, pp. 364-373.
8. P. Shivakumar and N.P. Jouppi, *CACTI 3.0: An Integrated Cache Timing, Power, and Area Model*, tech. report, Compaq Computer Corp., 2001.
9. C. Kim, D. Burger, and S.W. Keckler, "Nonuniform Cache Architectures for Wire-Delay Dominated On-Chip Caches," *IEEE Micro*, vol. 23, no. 6, Nov.-Dec. 2003, pp. 99-107.

Christianto C. Liu is a PhD student in electrical and computer engineering at Cornell University. His research interests include the design of novel circuits and architectures for 3D IC technology. He has a BS in electrical engineering from the California Institute of Technology and an MEng in electrical and computer engineering from Cornell University. Liu is a member of Tau Beta Pi and currently holds an AMD/SRC graduate fellowship. He is a student member of the IEEE.

Ilya Ganusov is a PhD student in electrical and computer engineering at Cornell University. His research interests include high-performance microprocessor architectures, prediction and speculation mechanisms, prefetching techniques, multithreading, and compiler technology. Ganusov has a BS in electronics engineering from Ivanovo State Power University, Russian Federation, and an MS in electrical and computer engineering from Cornell University. He is a student member of the IEEE and the ACM.

Martin Burtscher is an assistant professor in the School of Electrical and Computer Engineering at Cornell University. His research interests include high-performance microprocessor architecture, instruction-level parallelism, and compiler optimizations. Burtscher has a BS and an MS in computer science from the Swiss Federal Institute of Technology (ETH), Zurich, and a PhD in computer science from the University of Colorado at Boulder. He is a member of the IEEE Computer Society, the IEEE, and the ACM.

Sandip Tiwari is a professor of electrical and computer engineering at Cornell University, the Lester B. Knight Director of the Cornell NanoScale Facility, and director of the National Nanotechnology Infrastructure Network. His research interests include small structures, devices, circuits, systems, and related technologies. Tiwari has a BTech from the Indian Institute of Technology Kanpur, an MEng from Rensselaer Polytechnic Institute, and a PhD from Cornell University. He is a fellow of the IEEE and the American Physical Society.

■ Direct questions and comments about this article to Christianto C. Liu, 119 Phillips Hall, School of Electrical and Computer Engineering, Cornell University, Ithaca, NY 14853; ccL24@cornell.edu.