Transistor Budgets Go Ballistic

The Vanishing Transistor Opens Vistas for Microprocessor Architects

by Keith Diefendorff

Microprocessor designers face a dilemma: how to make effective use of a rapidly growing transistor budget. Even though transistors will be plentiful, failure to use them wisely can have serious ramifications. Billions of dollars can easily be poured down a rat hole. RISC architecture, for instance, seemed like a great idea when transistor budgets were under a million transistors per chip, but it became nearly irrelevant when the budget grew to four million transistors per chip.

Today, in a 0.25-micron process, the transistor budget for a microprocessor is on the order of 10 million transistors. Getting here from one million transistors took nearly 10 years. Getting the next order of magnitude may take only another four or five years, or perhaps fewer, depending on how much of the die is used for memory. Figure 1 shows the range of transistors that will be available on an inexpensive 200-mm^2 die over the next few years.

The other component of technology growth is transistor speed. Intrinsic transistor speed has been increasing at a rate of about 20% per year. CPU clock rates, which include both transistor speedup and design improvements, have been increasing at closer to 40% per year, and processors are on schedule to cross the gigahertz mark in 2000. Speed multiplies the effectiveness of transistors, allowing fewer of them to perform the same function.

More and faster transistors create many opportunities. Will architects use these new transistors for more complex CPUs, more cache, more integration, or what? We expect to see new CPU architectures that are capable of exploiting parallelism beyond that of traditional instruction-level parallel (ILP) processors. External SRAM caches and memory controllers are likely to move onto the CPU, leaving SRAM vendors holding an empty bag. Many of the functions currently performed by separate graphics or media coprocessors may be absorbed into the CPU—an eventuality that keeps coprocessor vendors awake at night, or should.

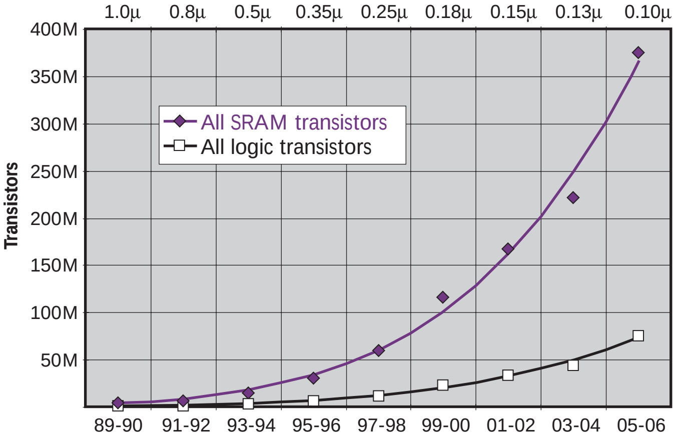

Figure 1. The transistor budget of a 200-mm^2 die, shown on a untraditional linear scale, lies between these two lines, depending on how much cache is on the processor. (Source: MDR)

Integration Doesn't Always Reduce Cost

An often cited, and often erroneous, argument for integration is that it reduces cost. But due to the exponential relationship between die cost and die area, one large chip is usually more expensive than two small ones of the same area.

This rule is not absolute. If the resulting chip is still small, less than about 80 mm^2, the nonlinear yield effects are insignificant and the integration reduces package, test, and system costs. Integration may also reduce cost if combining two functions onto one chip eliminates redundant circuits or eliminates pins. But for PC-microprocessor-sized die, integration itself seldom reduces cost. Simply grafting a 3D rendering chip onto a microprocessor, for example, would increase, not decrease, system cost. Furthermore, integration reduces flexibility, which can have indirect cost implications.

But peripheral-chip vendors can't relax just yet. Technology growth can endow a microprocessor with enough performance headroom to execute some peripheral functions in software, eliminating the need for another chip altogether. For example, a modem can now be implemented entirely in software using spare CPU cycles. Or, as in Cyrix's MXi, a small amount of rasterizing logic can eliminate an entire external 3D chip.

The best reasons for integrating a function onto the CPU are to improve bandwidth or performance. Integration onto one chip allows ultrawide, low-latency buses that cannot be provided across a chip boundary. Integration can also eliminate the overhead circuits and buffers associated with chip crossings. The resulting performance benefits are usually more significant than the cost savings, although performance can sometimes be translated into cost savings.

There is one other motivation for integration: putting a function on the CPU can set a standard. For example, MMX—which integrates the essential components of a video codec—has value mainly because it is more ubiquitous, if not more powerful, than an external coprocessor. This makes it more likely to be supported by, say, Microsoft.

Transistors Abound

The number of transistors on a die depends on several factors, chief among them being die area, process scaling, and the ratio of logic to memory. While a 400-mm^2 die fits within the reticle of modern steppers, manufacturers like to keep die sizes below about 200 mm^2 for high-volume manufacture. According to the MDR Cost Model, a 0.25-micron 200-mm^2 die has a manufacturing cost of $35. Adding a package and a 70% profit margin makes it a $200 processor.

The defect density of semiconductor processes improves gradually over time, which increases yield and allows a larger die to be built for the same cost. The Semiconductor Industry Association's (SIA's) *1997 National Technology Roadmap for Semiconductors* calls for an overall reduction of about 6% per year in defect densities, and about 12% per year within a given process generation.

On the other hand, fab equipment costs are increasing at a compound rate of about 20% per year, with little increase in wafer throughput. Depreciation of these assets is a major component of wafer cost. According to our projections, the net effect of better yield but higher wafer costs is a small (less than 2%) yearly reduction in the size of die that can be built and sold for $200. But the switch to 300-mm wafers, expected at 0.13 micron, should decrease the wafer-cost/cm^2, giving a one-time increase of about 25% in the constant-cost die size, resulting in a net 10-year increase of about 10%.

The prime driver of transistor count is process shrinks. Intel's 0.25-micron Deschutes implements logic transistors at a density of about 6 million transistors/cm^2 and cache SRAM arrays at 26 million transistors/cm^2. At the process generations predicted by the SIA, these densities could increase to 38 and 163 million transistors/cm^2 by the year 2006. (This ignores transistor-packing efficiency improvements from process refinements, additional metal layers, and better CAD tools. It should also be noted that the 1994 SIA roadmap predicted that 0.18-micron would not arrive until 2001, fully two years later than it will, and we suspect that the 1997 SIA roadmap may be similarly conservative.)

Most complex microprocessors today use only a small fraction of their die area for cache. Intel's Deschutes, for example, uses about 5% of its area for SRAM arrays. Mendocino, which will integrate 128K of L2 cache, will use about 20%. This is similar to the PowerPC 750, which is about 25% SRAM. Figure 2 shows the transistor budget that we project—including the effects of defect densities, wafer costs, and process shrinks—for various ratios of cache and logic.

CPUs Must Find New Approaches

RISC fell pitifully short in its bid to displace CISC in PCs, in part because it failed to deliver enough performance differential to overcome CISC's software lead. According to the game plan, RISC's architectural simplicity was supposed to free transistors that could then be spent on ILP-enhancing techniques, such as superscalar issue. The trick worked—but only while transistor budgets were a million transistors per chip. What RISC devotees failed to realize (or admit) was that CISCs could implement the same techniques once there were four million transistors on a chip, and that this would occur sooner than the software could be rewritten.

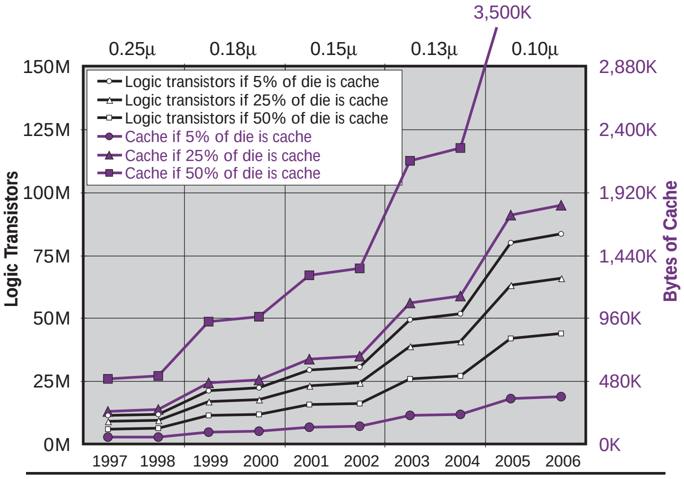

Figure 2. In 2001, a $200 microprocessor with 60% of its die area in cache will cross the 100M transistor mark. (Source: MDR)

CPU-core transistor counts have been steadily increasing to exploit more ILP. The core of the single-issue in-order 486DX4 was about 750,000 transistors. The dual-issue in-order Pentium/MMX core jumped to 2.8 million. The three-issue out-of-order Pentium II core is 5.7 million, and the six-issue out-of-order 21264 core takes about 8 million.

Intel and HP are continuing the hunt for ILP with their new VLIW-like IA-64 architecture. As it was for RISC, the goal is for the compiler to expose more ILP so the hardware can exploit more of it more easily. Also as with RISC, this is supposed to simplify the hardware at the cost of rewriting the software. We expect an eight-issue Merced (IA-64) core to come in at about 7.5 million transistors. (Another 2.5 million transistors may be provided for x86 compatibility, but those are not relevant to this discussion.)

It seems, however, that the industry may be reaching a point of diminished returns on transistors spent to mine ILP. There is scant evidence that going much beyond 10 million core transistors will uncover a lot more ILP. Techniques to exploit other levels of parallelism may have to be employed to scale CPU performance very far beyond this point.

One possibility is that CPU architectures may evolve to include more powerful vector or SIMD units to mine the low-level data parallelism in multimedia algorithms. Other approaches will be needed to harvest higher levels of parallelism: multiple program counters, multithreading, and chip multiprocessing (CMP) are all being investigated.

In selecting an approach, architects must be keenly aware of technology growth if they are to avoid the RISC syndrome. Any new approach must deliver a significant performance boost and must scale well, lest it be ambushed by technology growth before it achieves critical mass.

Cache: the Obvious Candidate for Integration

Whatever CPU architecture is used, the primary limiter of its performance is likely to be memory bandwidth or memory latency. Today's CPUs are already outrunning their memory systems—a trend that will continue. This will require more and more memory-system transistors to be spent per incremental logic transistor. This trend may, in fact, be what ultimately limits the number of deployable CPU logic transistors.

This fact makes cache the most obvious candidate for integration onto the CPU. The objective of any cache is to minimize the latency of the memory system, as seen by the CPU. Memory system latency is a function of the cache's access time on a hit, the average penalty on a miss, and the percentage of accesses that miss the cache (miss ratio).

Due to some pesky physics, the hit access time of a big cache is slower than that of a small one. For this reason, cache hierarchies are often employed, with small, fast level-one (L1) caches near the CPU backed by a larger, slower L2 cache, usually implemented in external SRAM. Although external caches, like Xeon's (see MPR 7/13/98, p. 1), can be made to run at full processor clock speed, their latency—the most critical parameter—will be worse than for on-chip caches that can have much wider and faster datapaths.

Integrating cache is a trade-off, because more of it can always be implemented externally than internally. All else being equal, large caches have lower miss ratios than small caches. Generalizing cache performance is inaccurate at best, because miss ratios are strongly dependent on the memory-access patterns of the software, but an often used rule of thumb is that miss ratios improve in proportion to the square-root of an increase in cache size.

If we assume that moving the L2 on chip cuts its latency in half, and that the effective miss rate of the combined cache is given by the square-root rule, Figure 3 shows the improvement in average access time for various on- and off-chip L2 cache sizes. (When moving an L2 onto the CPU, it is sometimes preferable to combine it into a larger L1, but this design decision does not affect the observations here.)

While this simple model ignores some important parameters and the situation will be different for any particular processor, it does illustrate a couple of salient points. For both internal and external caches, returns diminish above about 1M. While adding the first 1M of cache improves memory speed by 30% to 45%, adding another 1M improves it by only another 5%. Also notice that, due to the lower latency of the on-chip L2, a 128K on-chip cache gives nearly the same overall latency improvement as 512K of off-chip L2.

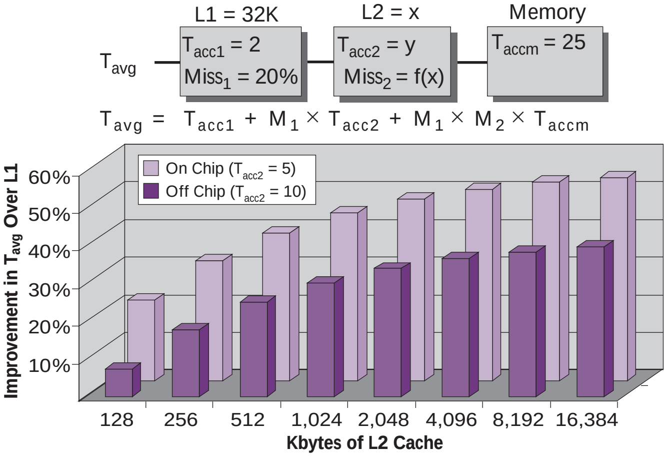

Figure 3. In this simple model, moving the L2 cache onto the CPU, thus cutting its access time in half, improves memory performance by about 15% for the same amount of cache. Notice the diminishing returns above 1M of cache. (Source: MDR)

The Alpha 21164 and 21264, the RM7000 (see MPR 8/3/98, p. 12), the PA-8500, and Intel's upcoming Mendocino are all examples of processors that implement on-chip caches of 128K or more. As technology permits, we expect to see this trend become ubiquitous and be extended to even larger caches.

North Bridge Integration Also Promising

Another component of PCs that is a candidate for integration is the north bridge, which interfaces the processor to DRAM and I/O. Even though a north bridge does not use many transistors, its integration is still hard to justify on a cost basis. It can, however, be justified on a performance basis.

With a north bridge, the DRAM latency on a cache miss is long. The CPU must first wait to sync up with the slow system bus, then use a high-overhead bus transaction to get the address to the north bridge, which must then perform the memory access and return the data to the CPU, again across the slow system bus. Most of this overhead would vanish with the memory controller on the processor. In the example of Figure 3 with an external 128K L2, reducing the memory access time by 10 clocks improves the average access time of the cache memory system by over 15%. Improving access time in this way has an effect similar to adding cache but it is even more effective because it does not have the statistical behavior of a cache.

Due to the performance advantage, we expect to see memory controllers move onto the CPU. The trend has already started with the MediaGX, MXi, and WinChip 2+NB, and it will be commonplace by the 0.18-micron generation, although Intel does not seem to be headed in this direction—yet. The trend will be aided by next-generation SLDRAMs and Direct RDRAMs that have narrower and faster channels than today's SDRAM channels, thus requiring fewer pins on the CPU while delivering higher performance.

In contrast, integrating the I/O functions found in traditional south-bridge chips or PCI devices results in little system speedup. This is true even for high-speed networking, SCSI, and IEEE-1394 devices. With little performance incentive and a potential loss of system flexibility, these functions are unlikely to migrate onto desktop PC processors soon, although we do expect to see them in low-end PC-on-a-chip products.

Integrating 3D Is Controversial

The most hotly debated candidate for integration is 3D graphics.The controversy stems from a characteristic unique to 3D graphics: an unbounded need for more performance. Regardless of how much horsepower or how many transistors are thrown at the problem, more will visibly improve the realism. Because we are so far from photorealistic animation today, the point of diminishing returns on 3D transistors seems far off. Indeed, real-time imaging of an HDTV animation with complex lighting and accurate model physics could occupy tens of billions of transistors running at a gigahertz.

So the debate rages: Can enough 3D graphics ever be put on the CPU to compete with external solutions? Today, 3D chips are by themselves as large as many microprocessors and have similar transistor counts. The Glint Gamma, which implements the geometry and lighting stages of a 3D pipeline at 3.3 Mpolygons/s, employs over five million transistors. The Riva TNT (see MPR 8/3/98, p. 1), which implements the setup and rendering stages at 250 Mpixels/s, uses seven million.

Graphics chips at these transistor counts are okay for today's games but not for tomorrow's, and they are light years away from photorealistic animation. It is a quirk of the market that most 3D chips sell for only a tenth of what Intel microprocessors sell for, making it extraordinarily difficult to reduce cost by integrating 3D onto the processor.

But things could change, and 3D may not escape the integration vacuum after all. One possibility is that the market could decide there is a threshold level of 3D image quality that is good enough. If this level turns out to be within the transistor budget of the processor, then poof!, 3D chips will be history. This will almost certainly happen in entry-level PC processors. The only question is how far up the performance scale this trend will extend.

Another possibility is that new 3D algorithms may be developed that require fewer transistors or are better suited to integration. Such algorithms might take advantage of tight coupling with the CPU in ways that conventional 3D-graphics pipelines cannot. Also, if the memory controller moves onto the CPU, an external 3D chip may have difficulty getting main-memory bandwidth, giving 3D logic on the processor an advantage in rendering to image buffers or to UMA (unified-memory architecture) frame buffers. Moving 3D onto the CPU would require the frame buffer to also be integrated, or that the CPU have a fast path to it—as graphics accelerators do today. But this can be arranged.

Technical arguments aside, the most compelling argument for integrating 3D is that it would be in the best interests of Intel. As we have pointed out several times, Intel is in desperate need of horsepower-hungry applications that can drive demand for expensive processors. 3D has potential in this role. Integration might even make 3D more ubiquitous, thereby promoting the emergence of new applications—beyond games—that would make 3D more broadly appealing. Such a scenario might spur demand for the high-performance (high-transistor-count) processors that Intel is uniquely positioned to supply.

Intel cannot afford to allow 3D-chip companies to extract profit from the PC platform. Considering what consumers are willing to pay for PCs these days, there is precious little profit to go around. Integrating 3D into the CPU could have strategic benefits, similar to Microsoft's integration of the browser into its operating system. We suspect that Intel's acquisition of 3D technology from Lockheed-Martin and its fielding of the 740 are preludes to integration of 3D, first onto the north bridge and ultimately onto the processor.

Conceivably, if the demand for 3D does broaden beyond games, and if 3D really can use an unbounded number of transistors, it could be the 3D chip that ends up swallowing the CPU. This would not necessarily be bad for Intel, as long as it could mentally adjust to the new paradigm.

Multimedia More Likely to Be Sucked In

Multimedia processing, although computationally intense, is a more bounded problem than 3D. Once enough transistors are available for a function, more add little benefit.

The most demanding multimedia task in use today is MPEG-2 encoding. The highest-quality, highest-compression encoding calls for a full-range motion search requiring many millions of transistors. But hierarchical search algorithms can get by with far fewer transistors yet produce near-identical image quality. The motion-estimation circuits for a real-time main-profile/main-level (MP@ML) encoder can be built with fewer than two million transistors running at 100 MHz. Using limited search algorithms, a 500-MHz processor with a vector or SIMD unit, like PowerPC's AltiVec (see MPR 5/11/98, p. 1), can also accomplish this task.

MPEG-2 decoding is far less demanding; hardware decoders can be built with under a half-million transistors. DVD decoding with full Dolby AC-3 sound can be processed in software by a 400-MHz MMX-equipped processor. Table 1 gives the transistor counts and die area for a number of processor, multimedia, and 3D units.

	Million	Die Area (mm^2)		
	Transistors	0.25µ	0.18µ	0.13µ
486 DX4 Core	0.7	11.7	6.0	3.2
P55C Core	2.8	46.5	24.1	12.6
Deschutes Core	5.7	95.0	49.2	25.7
PowerPC 750 Core	2.0	33.3	17.3	9.0
21164 Core	3.3	55.4	28.7	15.0
Merced Core (w/o x86)	7.5*	125.0	64.8	33.8
Merced x86 Compatibility Unit	2.5*	41.7	21.6	11.3
21264 Core	8.0	133.3	69.1	36.1
1 GFLOP DP Float Unit (@500 MHz)	0.3	4.4	2.3	1.2
DVD Decoder	0.5	8.3	4.3	2.3
MPEG-2 MP@ML Motion Estimator	1.5	25.0	13.0	6.8
AltiVec Unit	0.8*	13.3	6.9	3.6
3 Mpolys/s 3D Geometry/Lighting	5.0	83.3	43.2	22.5
250 Mpixel/s 3D Setup/Rendering	7.0	116.7	60.5	31.5
32K Cache	1.7	6.6	3.4	1.8
1M Cache	54.7	210.2	109.0	56.8

Table 1. Example transistor counts and the die area to implement them at densities typical of 0.25-, 0.18-, and 0.13-micron processes. At 0.18 micron, a 21264 core (or two 750 cores and two vector units), 128K of L1 cache, and 1M of L2 cache will all fit on a 200-mm^2 die. (Source: MDR estimates; *projected)

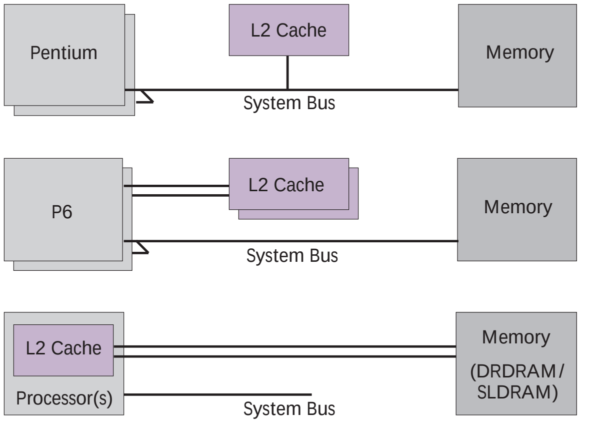

Figure 4. First-generation PCs (top) used the system bus for all memory traffic and I/O. Second-generation systems (middle) moved much of the memory traffic off the system bus with a private backside-cache bus. Next-generation systems (bottom) may move cache onto the CPU and use private channels directly to DRAM for all memory traffic.

C-Cube's DVx video codec (see MPR 12/8/97, p. 1) is capable of real-time MP@ML MPEG-2 encoding with simultaneous decoding of two DVD streams. The 100-MHz DVx chip includes a MicroSparc processor, 24K of cache and SRAM, and a DSP with motion-estimation hardware. The chip requires 5.4 million transistors, 3 million of which are memory. HDTV resolution (1,920 × 1,024) significantly increases the demands, but a 500-MHz version of the CPU should handle it without many additional transistors.

In the case of multimedia processing, integration can have cost, performance, and ease-of-use benefits. Cost savings would come from eliminating redundant circuits and data paths. For example, in the case of integrating DVx's functions, the MicroSparc processor, its cache, and some of the DSP would be superfluous. Performance increases would come from tighter coupling, giving software lower overhead, finer grained access to the hardware.

Embedded DRAM Looks Promising, But ...

An often-touted candidate for integration onto the CPU is DRAM. DRAM caches, for example, could have much higher capacity than SRAM caches in the same area. Integration of the graphics frame-buffer and even main memory are also interesting possibilities.

Embedded DRAM (see MPR 8/4/97, p. 19) is not a slam dunk. DRAM processes are tuned for low-leakage (slow) transistors with little interconnect. Logic processes are tuned for fast transistors and multilevel interconnect. A merged DRAM-logic process must either push both aspects of the process (and be expensive) or be a compromise. A compromise process has bigger DRAM cells, slower logic, or both.

This trade-off argues against integrating not only DRAM onto the CPU but also functions that would otherwise benefit from it. For example, if a graphics accelerator used an embedded-DRAM frame buffer, then implementing the accelerator as a separate chip may be preferable to integrating it onto the CPU, which would require sacrificing the frame buffer or penalizing the CPU speed.

Embedding DRAM can save significant amounts of power by eliminating chip crossings, making it attractive for microprocessors going into battery-powered applications. But, barring an unforeseen breakthrough in merged DRAM-logic processes, we do not expect to see DRAM on desktop PC processors for a few years.

No Obviously "Correct" Answer

With transistor budgets growing exponentially, technology constraints of the past will be relaxed. With so many transistors, the number of alternatives becomes bewildering—even frightening, considering the cost of a bad choice. More than ever before, we expect the definition of future processors to be driven by business concerns more than technical arguments.

Intel, for example, while possessing unparalleled technology prowess, may actually go slower than its competitors, which are desperately searching for ways to differentiate themselves. Intel may tread slowly, in part because it is not unhappy with the status quo, and in part because it is restrained by the burden of its own success. While its Merced may be a bold new approach in some ways, it is really just another high-ILP processor. This limited approach may leave an opening for Intel's competitors to exploit even more radical CPU architectures (like multiple processors on a chip) or more aggressive integration strategies (like PC-on-a-chip).

One thing is clear: future microprocessors will be different for different markets and will change with time and technology. The only universal trend seems to be the integration of cache onto the CPU. We expect this trend to take hold across the complete spectrum of microprocessors, possibly eliminating external SRAM caches completely within a couple of years. As if life weren't already miserable enough for SRAM vendors, this should make it unbearable.

In midrange-and-up PCs, the emphasis is likely to remain on CPU performance, including multimedia processing, for some time. The memory controller will move on chip, especially after DRDRAMs become pervasive. 3D geometry processing will probably be handled by the CPU, but rendering will remain external for a long time—unless Intel decides to force it onto the CPU for strategic reasons.

For entry-level PCs and below, there is likely to be more emphasis on integration, with designers opting to spend fewer transistors on the CPU and more on multimedia, 3D, UMA memory controllers, and, eventually, on high-speed I/O (gigabit Ethernet and IEEE-1394) and USB. In very-low-end markets and handheld applications, processors may integrate the complete memory system using embedded DRAM.

The ultimate question is whether Intel's competitors can find ways to put the enormous transistor budgets at their disposal to work carving out a defendable niche. The technology will provide a lot of opportunities to do so, but they will have to be aggressive and take risks. M