SPECIAL GIANT ISSUE COVERING THE 1998 MICROPROCESSOR AND EMBEDDED PROCESSOR FORUMS

MICROPROCESSOR REPORT

VOLUME 12, NUMBER 14
OCTOBER 26, 1998

THE INSIDERS' GUIDE TO MICROPROCESSOR HARDWARE

K7 Challenges Intel

New AMD Processor Could Beat Intel's Katmai

by Keith Diefendorff

Tired of eating Intel's performance dust, AMD is preparing a new entry in the battle for the CPU socket in high-end PCs. At the Microprocessor Forum earlier this month, chief architect Dirk Meyer described AMD's next-generation K7 processor, on which the company will place its hopes for higher ASPs and improved profitability. The new processor was jointly developed by Meyer's team in Austin (Texas) and Fred Weber's team in Sunnyvale (Calif.).

The K7's three-issue superscalar design, which reorders instructions across a 72-instruction window, is the most aggressive of any x86 design yet announced. Despite this high degree of instruction-level parallelism (ILP), the K7's 10-stage pipeline should allow it to achieve high clock rates. With silicon in hand, AMD is confident the processor will exceed 500 MHz in its first incarnation, which the company expects to deliver in the first half of 1999 (an industry euphemism for June).

As Figure 1 shows, the new processor adopts the dual-bus architecture of its competitor, Pentium II, and its ancestor, NexGen's Nx686 (before it morphed into the Socket 7-based K6). The K7 employs large 64K L1 instruction and data caches, allowing it to function in low-cost systems without an L2, as well as in high-end systems with a backside L2.

To avoid the multiprocessor and memory-bandwidth limitations of the K6's Socket 7, the K7 adopts the Alpha 21264 bus. Although this bus is technically superior to Intel's P6 bus, AMD may have bitten off more than it can chew, as it must create a new bus infrastructure without Intel's help. AMD's success in building a 100-MHz Socket 7 infrastructure for the K6, however, indicates that its goal is possible.

Despite its 22-million-transistor complexity, the new processor is astonishingly small, occupying only 184 mm^2 in AMD's 0.25-micron process. In this mature process, AMD should experience none of the early yield and production problems that plagued K6. When the K7 is shrunk to 0.18 micron—which AMD will do in 2H99—there will be plenty of room to integrate additional features, such as L2 cache, a DRAM controller, or 3D-graphics rendering pipelines.

Symmetric Decoders Key to Throughput

Like Intel's P6, the K7 decodes three x86 instructions per cycle. Unlike the P6, and more like the K6, the K7's decoders are generalized and completely symmetric. Whereas the P6's decode pipeline will stall if any instruction other than the first in an issue packet is even mildly complex (e.g., not register to register), the K7 won't skip a beat. Furthermore, each of the K7's three pipelined decoders can handle moderately complex x86 instructions, including instructions such as load-operate-store and as long as 15 bytes.

The symmetry and generality of the K7's decoders may have limited benefit for code compiled for the P6, since such

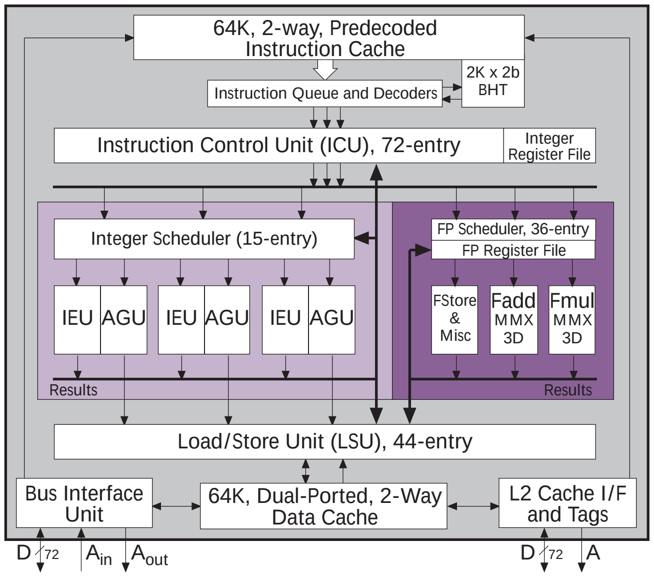

64K, 2-way, Predecoded Instruction Cache
2K x 2b BHT
Instruction Queue and Decoders
Instruction Control Unit (ICU), 72-entry
Integer Register File
Integer Scheduler (15-entry)
IEU
AGU
IEU
AGU
IEU
AGU
Results
FP Scheduler, 36-entry
FP Register File
FStore & Misc
Fadd MMX 3D
Fmul MMX 3D
Results
Load/Store Unit (LSU), 44-entry
Bus Interface Unit
64K, Dual-Ported, 2-Way Data Cache
L2 Cache I/F and Tags
D 72
A_in
A_out
D 72
A

Figure 1. Up to 72 instructions can be in execution in K7's out-of-order integer pipe (light purple), floating-point pipe (dark purple), and load/store unit.

code trys to avoid instructions that run afoul of the P6's decoders. But a lot of software is still compiled for 486s and Pentiums, which should be no problem for the K7. The K7's decoders will easily handle pathological code sequences that trip up the P6, and it should also have an advantage in multimedia processing, where memory-to-memory operations are frequent. Without improvements in Katmai's instruction decoders, the chip's KNI multimedia extension (see MPR 10/5/98, p. 1) may not outperform K7's implementation of 3DNow, despite KNI's more powerful architecture.

Long Pipeline, But Simple Branch Predictor

As lines are brought into the K7's instruction cache, which is 64K and two-way set-associative, they are inspected to locate instruction boundaries and branch instructions. This information is stored—three bits per instruction byte—in a predecode cache to help accelerate branch prediction and alignment of x86 instructions for the decoders.

As Figure 2 shows, x86 instruction decoding occupies the first half of the K7's 10-stage pipeline. In the *fetch* stage, 16 bytes are fetched from the I-cache. In the second or *scan* stage, these bytes are transferred into the direct-path alignment queue and scanned for complex instructions, like string ops, that require microcode sequencing and must be sent to the vector-path decoder. Because the direct-path decoders are powerful, few instructions require the vector-path decoder.

The K7 implements a surprisingly simple branch predictor for a machine with such a long pipeline. The K7 uses a 2,048-entry branch history table (BHT) with a simple two-bit Smith prediction algorithm (see MPR 3/27/95, p. 17). This predictor stands in sharp contrast to the K6's elaborate 8,192-entry BHT with its two-level GAs predictor—a feature that AMD now admits was overkill.

The K7's BHT is accessed in the *fetch* stage using the branch address, and the prediction is made in the *scan* stage. The prediction is fed back to direct instruction fetch on the next cycle. Branch target addresses are computed on the first misprediction and stored in a 2,048-entry branch target address cache (BTAC). A 12-entry return address stack is also provided. The short prediction-feedback loop inserts only a single bubble on a correctly predicted branch, and this bubble is normally squashed by the alignment queue.

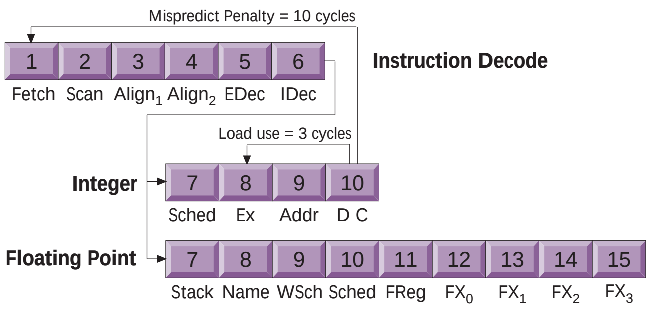


Figure 2. K7's deep 10-stage pipeline enables a high clock rate, but it pays a serious penalty on mispredicted branches.

The problem with the K7's branch strategy is that a misprediction incurs a minimum 10-cycle penalty, and the two-bit predictor is not very omniscient. The predictor does include a few simple but proprietary enhancements to improve its behavior in important special cases. AMD insists that a more accurate predictor would have been more complex, requiring additional prediction cycles or increasing cycle time and resulting in a net performance loss.

Decoders Deliver High Instruction Bandwidth

The K7's decoders convert variable-length x86 instructions into fixed-length "macro ops" (MOPs) and deliver them to the in-order instruction control unit (ICU). The ICU dispatches these MOPs to the instruction schedulers in the out-of-order core. The schedulers convert MOPs into RISC ops (ROPs), which they issue to the execution units. The execution units can execute up to nine ROPs per clock. It is the job of the K7's direct-path decoders to keep the ICU fed, so that it never stalls for want of MOPs to dispatch to the core.

As Figure 3 shows, at the head of the direct path is a queue of three 8-byte instruction registers. When empty, new entries fall straight through the queue to the next pipeline stage, but the goal is for fetch to stay ahead of decode and execution, keeping the queue full. In this way, the decoders always have 24 instruction bytes to consider, usually enough to extract three complete x86 instructions (which are about 2.5 bytes long on average) and to absorb branch bubbles.

In the *align*$_1$ and *align*$_2$ stages, an array of multiplexers extracts three x86 instructions from the queue, aligns them, and passes them to three identical parallel x86 instruction decoders. The multiplexers implement a full bytewise crossbar on the instruction queue, so as long as the instruction queue is sufficiently full, it appears as a sequential linear stream of instructions. Generally, the queue is sufficiently full as the average x86 instruction is shorter than one-third of the 16 bytes fetched from the I-cache every clock, and the execution pipelines rarely achieve 3-IPC throughput.

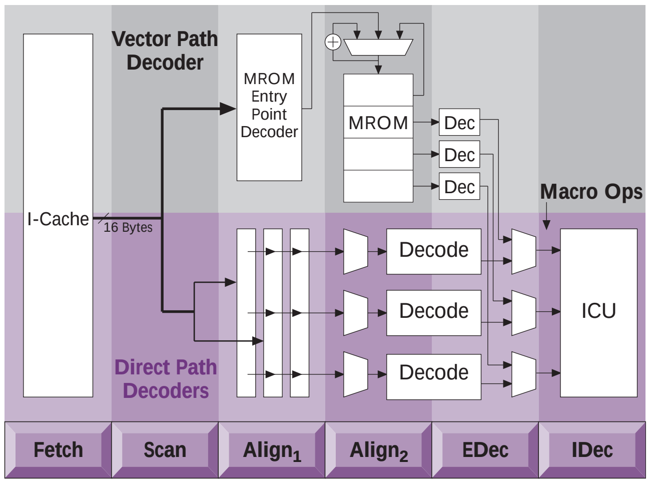


Figure 3. Symmetric decoders on the direct path (purple) can decode three x86 instructions into three macro ops every cycle.

At the end of the fifth, or early decode (*edec*) stage, each of the three x86 instructions has been transformed into a MOP. Most x86 instructions are representable as a MOP, but a few complex ones are not. These complex instructions are identified in the *scan* stage and passed to the vector path. Any MOPs produced from the vector path are merged, in program order, with direct-path MOPs and passed to the ICU. The ICU holds up to 72 MOPs in an in-order instruction queue.

RISC Core Eats MOPs

In stage six (*idec*) of the pipeline, the ICU prepares MOPs for dispatch to the integer and floating-point pipes. Due to the complexity of the x86 floating-point register file's stack-based addressing scheme, the ICU treats floating-point, MMX, and 3DNow MOPs differently than integer MOPs. (For brevity, we will refer to these collectively as FP MOPs.)

FP MOPs are passed directly into the floating-point pipe. Integer MOPs, on the other hand, get their destination registers renamed and their source operands read from the integer register file. AMD did not specify the number of physical registers but indicated that it was sufficient to rename the destination registers of all 72 instructions that could possibly be in flight at any given time, so the pipeline never stalls for lack of rename resources.

When the ICU dispatches a MOP to the integer scheduler, it produces a tag as a surrogate for any source operand that is not yet available (from a previous uncompleted instruction). The tag is used to match source operands to results returning from the execution units.

Out-of-Order Integer Pipe Issues Six ROPs/Cycle

The integer scheduler sets at the head of the integer pipe, occupying stage seven of the pipeline. This scheduler is an out-of-order 15-entry reservation station organized as three 5-MOP queues, as Figure 4 shows.

The basic unit of work issued to an execution unit is a ROP. ROPs specify either a load, a store, a load-store, an ALU operation, or a branch. A MOP is the semantic equivalent of one or two ROPs. A MOP, for example, represents a load-op-store x86 instruction as a load-store ROP plus an ALU ROP. The integer scheduler can issue up to six ROPs every cycle.

The integer pipe provides three integer execution units (IEUs) and three address generation units (AGUs). Each of the scheduler's three queues is physically associated with an IEU/AGU pair. Queue entries hold MOPs, from which individual ROPs can be issued, in any order, subject only to the availability of source operands and an execution unit.

A MOP representing a register-to-register ALU operation issues as a single ROP. IEUs execute ROPs in a single cycle, except for multiplies and divides, which iterate using IEU_0, IEU_1, and a shared multiplier. As IEUs complete execution, results are placed on one of three result buses and returned to the ICU. If necessary, the results are forwarded directly onto the source operand buses for use by ROPs waiting on them in the scheduler. The ICU stores the results in its in-order queue until the associated MOP is retired (in program order), at which time the results are written into the architectural register file. Up to three MOPs can be retired every cycle, matching the maximum decode rate.

When the ICU dispatches a store MOP to the scheduler, it immediately places a store request, along with store data if it's available, on the load/store unit's (LSU's) queue. The scheduler then issues the store ROP to an AGU, which computes the store's effective address and forwards it to the store request waiting in the LSU. If store data was not available when the store MOP was dispatched, the LSU snoops it from the result buses when it does become available.

Load MOPs issue a single ROP to an AGU, which places a load memory request onto the LSU's queue. After the request is serviced by the data cache, the LSU returns the load data on the result buses.

Load-op MOPs issue as two ROPs: first a load ROP is issued to an AGU; then, after load data is returned, an ALU ROP is issued to an IEU. While the ALU ROP is waiting to issue, other ROPs can be issued to the same IEU/AGU pair. Even though a MOP is physically aligned to an IEU/AGU pair, the IEU and AGU are independent, allowing an ALU ROP and a load/store ROP from different MOPs to be issued to the same IEU/AGU pair in the same cycle.

Load-op-store MOPs, like load-ops, issue as two ROPs. First a load-store ROP goes to an AGU and a load-store request (a load and a store request to the same address) is placed on the LSU's queue. The LSU executes the load half first, returning the result that enables the ALU ROP to issue to the IEU. When it finishes, the LSU snoops the store data from the result bus and executes the store half of the request.

FP/Multimedia Pipe Complicated by Stack

Because FP MOPs are simply passed to the FP pipe, four additional pipeline stages are required to get FP ROPs into execution, rather than the one required for integer ROPs. In the first stage of the FP pipe—stage 7 of the overall pipe-

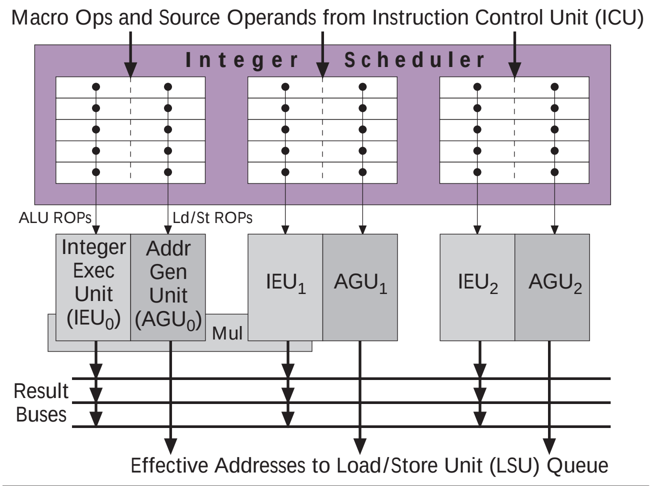

Macro Ops and Source Operands from Instruction Control Unit (ICU)
Integer Scheduler
ALU ROPs
Ld/St ROPs
Integer Exec Unit (IEU_0)
Addr Gen Unit (AGU_0)
Mul
IEU_1
AGU_1
IEU_2
AGU_2
Result Buses
Effective Addresses to Load/Store Unit (LSU) Queue

Figure 4. The integer scheduler converts MOPs to ROPs and issues up to six per cycle, out of order, to the execution units.

line—the floating-point stack is renamed to a flat register namespace. In the next stage, these registers are renamed again, this time for the traditional purpose of avoiding false dependencies. As in the integer case, enough physical registers are provided to rename the destination registers of all 72 of the FP MOPs that can be in flight at one time.

In stage 9, FP MOPs are dispatched to the 36-entry FP scheduler and scheduled onto the appropriate execution unit in stage 10. Scheduling is more difficult than for integers, because the execution units are not symmetric and because they have different and variable latencies. In stage 11, operands are read from the FP register file and ROPs issued to the execution units. Finally, in stage 12, execution begins.

As for integer ROPs, FP ROPs can be issued in any order from the scheduler, subject only to data dependencies and execution unit availability. In the case of FP ROPs, the reordering is more important, because the asymmetric execution units impose more structural hazards, and the longer execution latencies cause longer pipeline delays. The K7 accounts for this with a 36-entry reordering window, deeper than is used in the integer pipe.

Superscalar Floating Point, MMX

In the K7's FP pipe, three execution units—Fmul, Fadd, and Fstore—handle all floating-point, MMX, and 3DNow instructions. The scheduler processes FXCH (floating-point exchange) ROPs, up to three per cycle, by simply renaming registers and issuing a NOP to an execution unit.

The Fstore unit, among other things, transports store data from the FP registers to the LSU. For each FP load or store MOP, a MOP is also dispatched to the integer pipe to compute the memory address.

An FP ROP can issue to each unit on every cycle. The units are all fully pipelined with latencies given in Table 1. The K7 will have a peak x87 floating-point throughput of 1 GFLOPS at 500 MHz (twice P6's) and a peak 3DNow throughput of 2 GFLOPS. FP reordering capabilities appear adequate to sustain these rates, even with four cycles of latency between multiplies and dependent adds (a common sequence). The 3DNow throughput is the same as Katmai's KNI implementation, even though KNI provides twice the architectural parallelism, because Katmai's implementation is not fully pipelined to support KNI's full 128-bit architectural width.

	K7		Katmai†	
ROP Issue Rate	**Multiply**	**Add**	**Multiply**	**Add**
Floating Point	1 instr	1 instr	0.5 mul or 1 add	
MMX	1 instr	1 instr	1 instr	1 instr
3DNow/KNI	1 instr	1 instr	0.5 instr	0.5 instr
SIMD Width	2 ops	2 ops	4 ops	4 ops
Cycles	**Latency**	**Throughput**	**Latency**	**Throughput**
FP +	4 cycles	1 cycle	3 cycles	1 cycle
FP ×	4 cycles	1 cycle	5 cycles	2 cycles
MMX +	2 cycles	1 cycle	1 cycle	1 cycle
MMX ×+	3 cycles	1 cycle	3 cycles	1 cycle
3DNow/KNI +	4 cycles	1 cycle	4 cycles	2 cycles
3DNow/KNI ×	4 cycles	1 cycle	6 cycles	2 cycles

Table 1. The K7's superscalar fully pipelined FP unit should clean Katmai's clock. (Source: AMD for K7, †MDR projections)

No Scrimping on Memory Bandwidth

Having built a powerful execution engine, Meyer was determined to provide enough memory bandwidth to keep it fed. To this end, the K7 implements aggressive memory reordering, a large multiported L1 data cache, an associative backside L2 cache with on-chip tags, a multilevel TLB, and a memory interface with significantly more bandwidth than is provided by either K6's Socket 7 or Intel's Slot 1 or 2.

As Figure 5 shows, the heart of the K7's memory subsystem is the LSU with its large 44-entry memory-request queue. Each queue entry holds an address for either a load request, a store request, or a load-store request, along with data in the case of stores. Addresses are placed in the queue, up to three per cycle, by the AGUs. Store data is snooped from the result buses. Memory requests wait in the queue until granted access to the cache.

The 64K data cache is two-way set-associative and multibanked, making it appear effectively dual ported. The cache can service two 64-bit loads, a 64-bit load and a 64-bit store, or two 32-bit stores every cycle, providing up to 8 Gbytes/s of bandwidth to the core (at 500 MHz).

Requests are issued to the cache in the order that best suits advancement of the execution-unit pipelines, subject to the constraints of the x86 processor-consistency rules. Stores wait in the queue until their data is received and the ICU signals that no instruction preceding it in program order will take an exception. This is one reason for the deep request queue, another being the desire to schedule as many loads as possible to maximize off-chip bandwidth.

Usually, loads take precedence, and stores are deferred until the cache is not busy. Occasionally, store priority is boosted to unblock retirement from the ICU, but normally there are plenty of empty slots. Loads are allowed to bypass stores, except when the load would bypass a store to the same address. In this case, or whenever a load targets the same address as a preceding store still in the queue, the store's data is forwarded to the load as soon as it becomes available, thus allowing the load to complete without a cache access.

The cache is nonblocking, so when an L1 miss goes to the L2 or the bus for resolution, other loads behind it in the queue can access the cache. The data cache has three complete sets of tags, allowing simultaneous tag lookup by two requests from the queue and a snoop from the bus.

The data cache is physically tagged. Effective addresses are translated to physical addresses in parallel with D-cache tag lookup by a two-level translation lookaside buffer (TLB). The first level has 32 fully associative entries and is backed by a 256-entry, four-way set-associative second level. The memory mapper supports both the 4K and 4M page sizes of Intel's 36-bit physical-address-space extension and the newer extended server memory (IESM) architecture.

On-Chip Tags Support Large External L2

The K7 provides on-chip tags for up to 512K of two-way set-associative external L2 cache. The on-chip tags allow fast hit/miss detection and enable set associativity with standard commodity SRAMs. The P6 has a four-way L2 but requires a custom tag chip. The K7's L2 interface supports a variety of SRAMs, including inexpensive pipelined-burst SRAMs (like those used on Pentium II), late-write SRAMs, and the new double-data rate (DDR) SRAMs for ultimate performance.

Although the K7 has on-chip tags for only 512K of external L2, it can support up to 8M with external tags. In this mode, the on-chip tags implement L2 associativity and a unique early-miss-detection feature, which avoids accessing the external tags in most misses. Occasionally, the early-miss detector produces a false hit, requiring a normal access to the external tags, but by avoiding many external tag accesses altogether, the average L2 miss penalty is lowered.

K7 Leaves P6's Bus in the Dust

The K7's system bus, which AMD borrowed (licensed) from Digital's Alpha 21264 processor (see MPR 10/28/96, p. 11), is the first PC processor to depart from a traditional multidrop shared bus. Unlike other PC processors, the K7 uses a point-to-point interconnect, as Figure 6 shows.

The advantage is speed. Multidrop buses have lumped-capacitive loads and transmission-line stubs that limit their frequency. Socket 7, for example, tops out at 100 MHz, and Slot 1 is unlikely to exceed 133 MHz. Both could be increased somewhat if restricted to a single load, but the signal timing and clocking of these buses were not optimized for point-to-point operation and will still limit frequency. The K7's bus, however, uses source-synchronous clocking to minimize skew and latch-to-latch signaling to reach 200 MHz, with Slot 1–type printed-circuit module packaging. Up to 400 MHz is possible on impedance-controlled multilayer boards.

While the K7 itself could reach the higher speed, AMD will limit its initial chip sets and reference-board designs to 200 MHz, enough to saturate a 1.6-Gbyte/s Direct RDRAM channel (to which AMD recently become a licensee) or a 128-bit 100-MHz SDRAM port—about all we are likely to find in PCs over the next couple of years.

The K7's bus comprises three separate ports: address in, address out, and a 72-bit bidirectional data port. These ports are all decoupled to allow split transactions, and the K7 can support up to 20 outstanding transactions. Transactions are tagged for out-of-order operation. The separate address-in and -out ports allow simultaneous memory and snoop transactions, which is important in shared-memory symmetric multiprocessor (SMP) systems, where coherency traffic can be intense and address-bus bandwidth can limit performance.

The K7 uses a five-state MOESI cache-coherence protocol, which adds the "owned" (or shared-modified) state to the P6's four-state MESI protocol. The owned state reduces write-invalidation traffic, thus making the K7 more efficient in SMP systems that share data and use shared-memory synchronization and communication protocols.

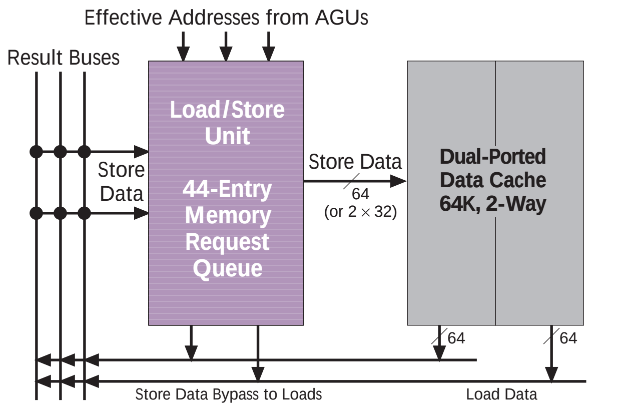

Effective Addresses from AGUs
Result Buses
Load/Store Unit
44-Entry Memory Request Queue
Store Data
Store Data
64
(or 2 × 32)
Dual-Ported Data Cache 64K, 2-Way
64
64
Store Data Bypass to Loads
Load Data

Figure 5. The load/store unit queues memory requests and issues them to the D-cache out of order. Store data is snooped from the result buses and forwarded to loads.

Although the K7's bus provides a major advance in performance over the Socket 7 and Slot 1 buses, AMD stopped short of the ultimate next-generation bus technology: a completely unidirectional point-to-point interconnect network with directory-based coherency, such as the Alpha 21364's design (see MPR 10/26/98, p. 12). AMD defends its choice, saying that the K7's bidirectional bus requires fewer pins. But the argument is unconvincing, as unidirectional pins avoid bus turnaround overhead and could easily have been clocked fast enough to deliver more bandwidth over the same number of pins. So, while the K7's bus clearly bests P6's, it does leave room for Intel to leapfrog AMD once again, as it did with Slot 1. Room is not something that one should leave for Intel.

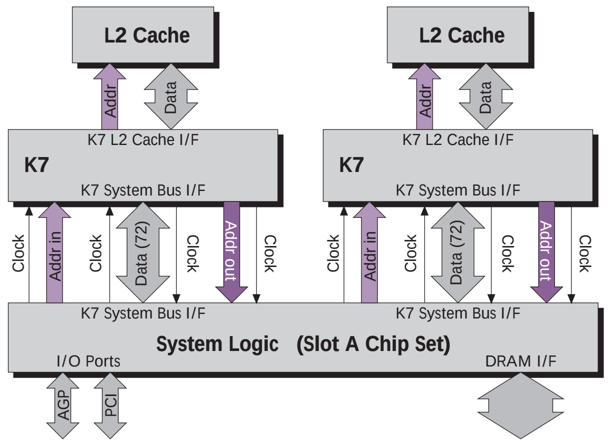


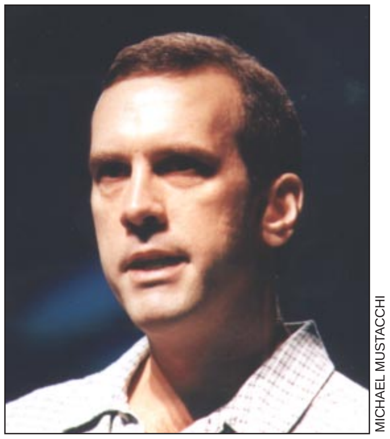
Figure 6. The K7 connects to the chip set via a point-to-point interconnect instead of a shared bus. This requires more pins in MP configurations but allows the bus to run at higher speed.

Better Bus, But Not a Slam Dunk

AMD had to do something. Socket 7 is clearly inadequate for future PCs and memory systems. Having negotiated away its ability to clone Slot 1, the company was forced to adopt a new bus for the K7. But the technical merits of the K7's bus could be moot. Unlike the K6's Socket 7—whose way was paved by Pentium—K7's bus has no ready-made chip set or motherboard infrastructure from which to get a jump start. Aside from an inconsequential contribution from Alpha 21264 systems, AMD alone must bear the burden of establishing an infrastructure to match Slot 1's. Although this is a formidable challenge, AMD's Socket 7 momentum may give it enough mindshare with chip-set vendors to get it over the hump.

To lower the barrier, AMD adopted the physical form factor of Pentium II's Slot 1, calling it Slot A. Because it is mechanically identical to a Slot 1 module, AMD can leverage existing card guides and PC chassis. Slot A also uses the same edge connector as Slot 1, although the connector is keyed to prevent insertion into Slot 1 sockets, or vice versa. Interestingly, AMD does this just as Intel's Celeron is moving from Slot 1 modules and back to a socket strategy.

AMD would not disclose the K7's power consumption, saying only that "it will fit within the Slot 1 thermal envelope." This sets an upper limit on the processor of about 40 W, which we expect the K7 to approach. Although flip-chip bonded in its 576-contact CBGA package, the K7 will use an aluminum heat spreader—unlike Intel's Pentium II, which is shedding its heat spreader in favor of directly attached heat sinks. AMD asserts the heat spreader gives OEMs a more convenient and universal heat-sink attachment point.



Dirk Meyer, chief architect of the K7, describes AMD's challenge to Intel's high-end microprocessors.

Although the K7 is, for the most part, mechanically compatible with Slot 1, it is electrically dissimilar and requires new chip sets and motherboards—a huge problem for AMD. But the company is actively designing standard-cell bus modules, chip sets, and reference motherboard designs that it intends to supply to the industry royalty-free. Five (unnamed) chip-set vendors have already signed up, and AMD is actively soliciting others. The company honestly believes, based on its K6 experience, that if it builds a compelling processor, the infrastructure problem will solve itself.

Although AMD can probably provide a few chip sets to get the K7 rolling, in the long run it will be difficult to match the breadth and depth of Slot 1 chip sets. The difficulty will increase over time. AMD must deliver—on the same schedule as Intel—chip sets with advanced memory systems, UMA graphics or AGP 4×, wide-and-fast PCI, and next-generation I/O like Gigabit Ethernet, Fibre Channel, and IEEE-1394.

While in theory the K7's bus allows compatibility with the Alpha 21264, in practice there are no specifications to cover this. It is questionable whether the thermal and frequency limitations of the K7's 200-MHz Slot A module are adequate for a high-speed 21264.

3DNow Faces Off Against KNI

Eventually, AMD must confront the issue of KNI. For now, the K7's implementation of 3DNow should perform as well as or better than Katmai's implementation of KNI, which is not fully pipelined. While KNI's new registers are an advantage, that advantage is diminished by the x86 architecture's two-operand format. AMD is relying on this fact, its current software lead, and its installed base (10 million units by the time Katmai arrives), to keep 3DNow alive.

In the long run, however, AMD cannot match Intel's investment in KNI compilers, libraries, development tools, and applications. As KNI proliferates, AMD will be forced to adopt it, at which point 3DNow becomes redundant and is sure to lose the attention of software developers. AMD recognizes the threat but fears that if it allows 3DNow to fail, Intel will regain total control of the x86 architecture, permanently relegating AMD to distant-follower status.

This is a valid concern, but not one within AMD's power to avoid. We believe AMD should drop the principle in favor of practical reality and adopt KNI. A bolder strategy would be to extend KNI with a radically more powerful multimedia architecture like AltiVec (see MPR 5/11/98, p. 1). The difficulty is that this would require architectural changes that Microsoft would have to agree to accommodate in Windows. It would also have to be sufficiently better than KNI to attract software developers.

K7 Will Outperform Pentium II

The K7 is, in many ways, an excellent mix of technology. It is evolutionary, not revolutionary, making it a low-risk design. The long pipeline should leave little frequency on the table. It exploits its large transistor budget wisely, with powerful symmetric decoders and a large complement of execution units. The deep instruction-reordering window desensitizes K7 to the compiler—an important attribute for AMD. The aggressive load/store unit, large dual-ported L1, backside L2, and high-speed bus reduce memory bandwidth as an issue.

The K7's superscalar FP pipe should give it impressive floating-point performance compared with that of the P6. But the K7's MMX and 3DNow implementations are less impressive. While providing a huge improvement over the K6, AMD missed a golden opportunity to skunk Katmai on multimedia applications. Although the K7's symmetric decoders give it some advantage, its single add unit and single multiply unit will limit throughput. To a large extent, however, floating-point, MMX, and 3DNow performance are hamstrung by the

	AMD		Intel	
Feature	**K7**	**K6-2**	**Deschutes**	**Katmai†**
x86 Decode	3 complex	2 complex	1 cplx + 2	1 cplx + 2
Issue Rate	9 ROPs	6 ROPs	5 ROPs	5 ROPs
Reorder Depth	72 x86	24 ROPs	40 ROPs	40 ROPs
Pipeline Stages	10 stages	7 stages	12–14	12–14
BHT Entries	2,048 × 2b	8,192	≥ 512	≥ 512
Return Stack	12 entries	16 entries	4 entries	4 entries
Int Multimedia	MMX	MMX	MMX	MMX
FP Multimedia	3DNow	3DNow	None	KNI
L1 Cache	64K/64K	32K/32K	16K/16K	16K/16K
L2 Cache I/F	64 bit	64 bit	64 bit	64 bit
Instr TLB	24+256	64 entries	32 entries	32 entries
Data TLB	32+256	128 entries	64 entries	64 entries
Clock Rate	> 500 MHz	400 MHz	450 MHz	500 MHz
Sys Bus B/W	1.6 GB/s	800 MB/s	800 MB/s	800 MB/s
Transistors	22 million	9.3 million	7.5 million	9-10 million
IC Process	0.25µ 6M	0.25µ 5M	0.25µ 4M	0.25µ 5M
Die Size	184 mm^2	81 mm^2	118 mm^2	140 mm^2
Production	1H99	Now	Now	1Q99
Est Mfg Cost*	$104	$47	$60	$66

Table 2. The K7 is the most complex of any current x86 processor, a title that should stand against Katmai. (Source: vendors, except *MDR estimates and †MDR projections)

x86 architecture; AMD will have to move beyond 3DNow and KNI if it wants to make significant advances.

A debatable choice was AMD's decision not to include an on-chip L2. With L2 on chip, a DRAM memory controller could have replaced the external L2 interface, giving the K7 a compelling advantage over Katmai. But AMD was understandably intent on keeping the die small for cost reasons, and it argues that K7 already has large L1s. Since these features can easily be added to the 0.18-micron K7 by the end of 1999, AMD may have made the right choice for now.

Although AMD declined to provide performance estimates, inspection of the K7's microarchitecture leaves little doubt it will outperform Deschutes, and probably Katmai, on an instructions-per-clock basis. Whether AMD can deliver on the clock-speed component remains to be seen.

From a pipeline perspective, we see no apparent obstacle. The question is semiconductor process. According to the MDR FET Performance Metric (see MPR 9/14/98, p. 1), Intel's 0.25-micron P856.5 is about 20% faster than AMD's 0.25-micron CS44E. AMD says it will use a slightly enhanced version of CS44E, which could close the gap. Even so, AMD must demonstrate an ability to manufacture high-speed parts with good yields—not historically a strong point for AMD.

The K7 looks good technically and, if it can cross the infrastructure hurdle, should put AMD in a far more competitive position than it is in now. Although Katmai will ship several months earlier at 500 MHz, K7 should outperform it handily if it ships midyear. Intel may pull ahead in 3Q99 with its 0.18-micron Coppermine, possibly at up to 700 MHz and probably with an on-chip L2, but AMD could recapture the performance lead in late 1999 with its own 0.18-micron version. An on-chip L2 would be a good addition to that chip.

The K7 could hold the performance lead until 2H00, when Intel delivers Willamette at speeds of up to 1 GHz (see MPR 10/26/98, p. 16). By that time the K7 may reach 1 GHz in the copper 0.18-micron process AMD is developing with Motorola, but it is still unlikely to match Willamette, which will have a far more aggressive microarchitecture than P6 and, probably, K7. AMD is working on the K8 to meet this challenge.

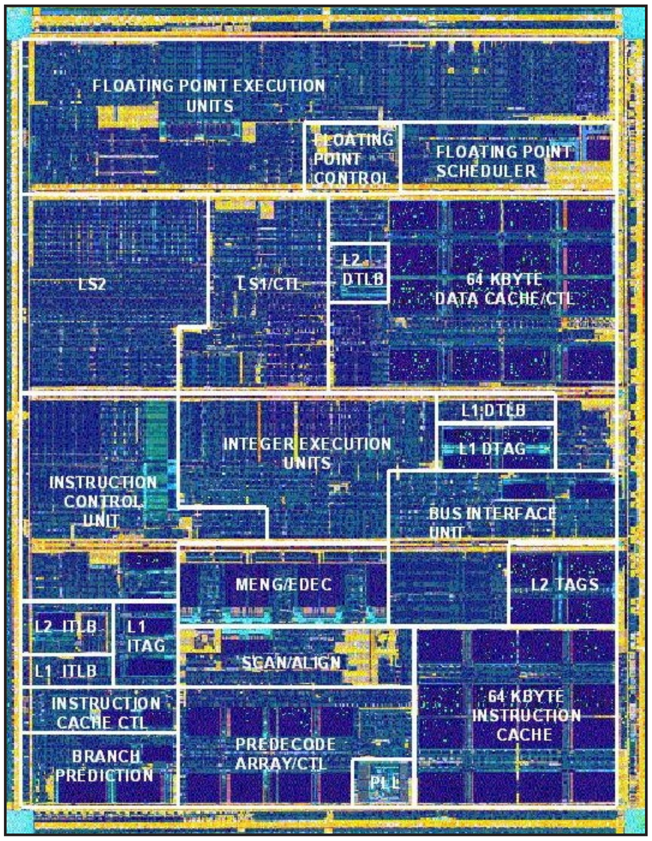


Figure 7. K7's 22 million transistors occupy 184 mm^2 in a slightly enhanced version of AMD's 0.25-micron 6-layer-metal CS44E process. (Source: AMD)

In this game of leapfrog, the K7 should give AMD long periods of performance leadership, a feat the K6 managed for only about a week. Intel's position being what it is, the K7 still may never achieve price parity with top-end Pentium IIs, but this is not necessary. At only 184 mm^2, as Figure 7 shows, it is only 30% larger than we project for Katmai and, as Table 2 shows, about 55% more expensive to manufacture. This situation should allow AMD to undercut Katmai's price by 10–20% and still maintain good margins.

Delivering its seventh-generation processor only 2.5 years after its sixth-generation device will be an impressive accomplishment for AMD, especially considering that Intel may take five years to do the same. The K7 will, for the first time, put AMD solidly in the high-end PC business and could enable a play in the midrange server/workstation games. Its fate hinges to a large extent on how aggressively Intel drives Katmai, KNI, and 0.18-micron technology across its product line. Regardless of Intel's efforts, however, the K7 should improve AMD's position, allowing it to increase both prices and profit margins—maybe significantly. M