AMD Unveils First Superscalar 29K Core

Sophisticated Organization Doubles High-End Performance

by Brian Case

At the recent Microprocessor Forum, AMD's Mike Johnson revealed details of his company's forthcoming new high-end 29000-family chips. This technical disclosure comes hot on the heels of Intel's recent announcement of the 960 H-series embedded controllers *(see **081302.PDF**)*. Like the 960 H-series, the new 29K chips offer superscalar execution, large on-chip caches, and clock-doubled and clock-tripled operation. Neither AMD's nor Intel's new chip is available now; production is expected for both by the middle of 1995.

This design is AMD's first new 29K core since the introduction of the family in 1988. (The integer core of the 29050 introduced some minor improvements to boost performance by about 10%, but it was essentially the same design as the original 29000.)

Like the 960 family, AMD's first superscalar chip will have few on-chip peripherals, but it will have the same MOESI cache consistency, power-saving modes, MMU, and memory-control capabilities as the 29040. Versions aimed at specific markets will come later. AMD said that it will fabricate the new chips in its 0.5-micron CMOS process but did not reveal the die size.

Microarchitecture Details

The core of the superscalar 29K is a completely new design. The similarities between this core and the K5 core *(see **081401.PDF**)* are not coincidental: the superscalar 29K was an "intellectual predecessor" to the K5.

The microarchitecture uses a decoupled organization that separates instruction fetching and dispatching from instruction execution *(see **081102.PDF**)*. This decoupled organization facilitates speculative and out-of-order execution.

As shown in the block diagram in Figure 1, the major hardware blocks are the two caches, the instruction dispatcher, the register file, the reorder buffer, and the execution units. The operand buses and the logic to select and forward values are distributed throughout the machine. These buses plus the reorder buffer consume significant chip area: about the same as 4K of cache.

Up to four instructions can be dispatched simultaneously to the six execution units. There is no fetch queue, so all four instructions from one instruction-cache line must be dispatched before the next line can be fetched. Instructions are dispatched from the instruction stream in program order; if an instruction cannot be dispatched for some reason, no instructions beyond it will be dispatched.

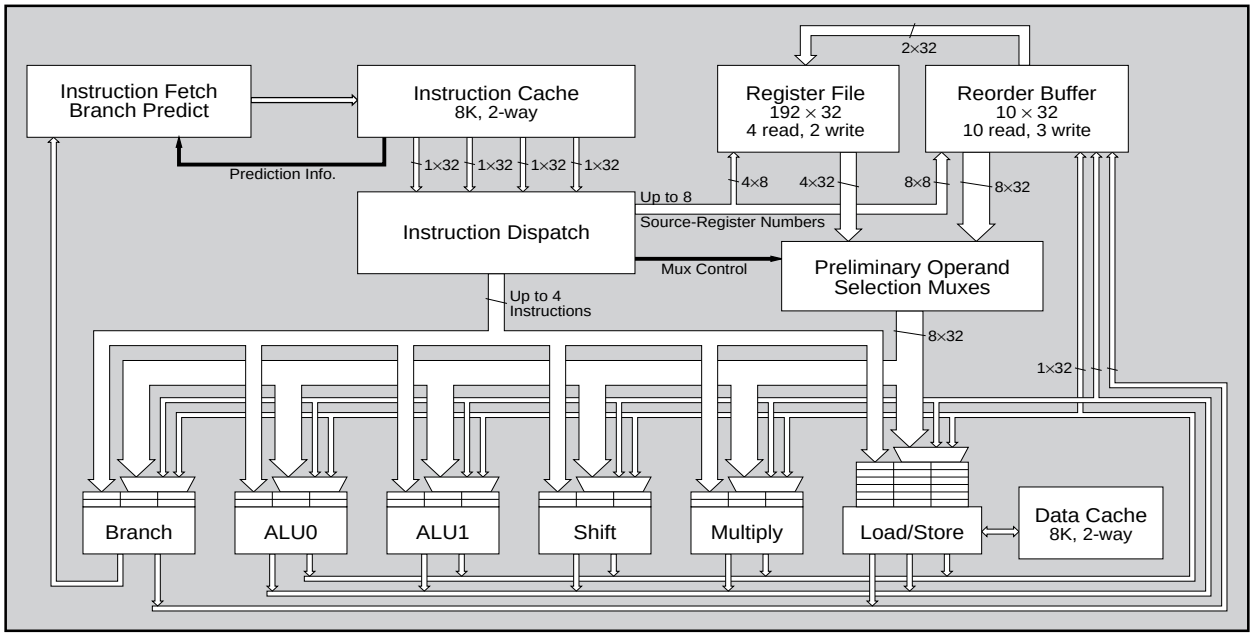

Figure 1. AMD has implemented a sophisticated, decoupled organization for the superscalar 29000. The decoder, register file, and reorder buffer have enough bandwidth to issue up to four instructions in a single cycle.

The execution units have reservation stations to buffer instructions that cannot begin executing immediately after they are dispatched. As Figure 1 shows, the load/store execution unit has six reservation stations; all others have two. Instructions within a given execution unit are executed in the same order that they are dispatched to that unit. Although instruction issue and execution within an execution unit are in order, instruction execution between units can be out of order.

There are eight source-operand buses, so it is possible to dispatch the maximum of four instructions even if each requires two source registers. For example, it would be possible to dispatch two ALU instructions, a store instruction, and a conditional indirect branch.

When an instruction is dispatched, its operands are read from either the register file or the reorder buffer, depending on where the most up-to-date copy of the needed operand resides. The reorder buffer has twice as many read ports as the register file because, during steady-state execution, the machine spends most of its time executing instructions speculatively, following the paths of predicted branches. Since the register file stores only results that are guaranteed to be correct, most operands are supplied by the reorder buffer, which stores the results of speculatively executed instructions. Any instructions that reference these values get their operands from the reorder buffer.

CLARENCE TOWERS

AMD's Mike Johnson, who led the development of the superscalar 29K and K5 designs.

If a register value is not yet available (because it is the result of an instruction that has not yet executed), the instruction that needs the value is dispatched to the appropriate execution unit with a tag (supplied by the reorder buffer) for the value instead of the value itself. When the value is available, it is forwarded directly from a result bus to the appropriate execution unit.

There are two result buses for delivering computed results to the 10-entry reorder buffer. The third is for branches and stores to return status information.

Comparing the 29K core with the K5 core makes clear their respective design targets. The K5, with a focus on highest performance, has more register-file read and write ports, a larger reorder buffer, more result buses, two load/store units, a dual-ported D-cache, and a larger I-cache. The 29K core, for which cost is the limiting factor, is tuned for an average execution rate of two instructions/cycle and has fewer expensive resources.

Exploiting Multiple Execution Units

The organization shown in Figure 1 has tremendous potential for instruction-execution concurrency, but exploiting the available concurrency requires finding independent instructions. To this end, this machine implements branch prediction and register renaming.

Register renaming is essentially a by-product of the reorder buffer *(see **081401.PDF**)*, which automatically provides a unique storage location for every result produced by an execution unit. When an instruction is dispatched, associative logic in the reorder buffer automatically supplies the most up-to-date values for the instruction's source registers (a tag is supplied if a result is pending). The reorder buffer helps eliminate unnecessary register dependencies that would stall execution *(see **081102.PDF**)*.

Branch prediction enables speculative issue and execution. As with register renaming, the reorder buffer naturally helps implement speculative execution by providing temporary storage for the results of speculatively executed instructions. If a speculation proves wrong, the results of the speculatively executed instructions can simply be invalidated in the reorder buffer, clearing the way for the machine to immediately start issuing instructions from the correct branch path.

Rather than use a separate hardware structure for branch-prediction information—as is done in Pentium and other superscalar implementations—the superscalar 29K stores branch-prediction information with each block of four instructions in the instruction cache. As each block is fetched from the cache, a "next-cache-block" pointer field, which is stored with each block, is used to predict the next block in the cache. This technique is used in other microprocessors, e.g., the forthcoming UltraSparc implementation *(see **081301.PDF**)*.

Implementation Complexity

To exploit the concurrency available in the superscalar 29K core, complex control logic and lots of buses are required to route operands to and from the execution units. The dispatch logic sends the source register numbers to the reorder buffer, which uses associative lookup to find any entries that have (or will have) the required values. If the reorder buffer does not have an entry for a needed register, the value in the register file is used. This relationship between the buffer and the register file creates a complex chain of logic that can create a slow timing path. Careful tuning in the implementation is required for high-speed operation.

Ideally, each execution unit would have a dedicated result bus for delivering results to the reorder buffer. In practice, rarely do all execution units generate results simultaneously. This fact—coupled with the high cost of

result buses, buffer write ports, and forwarding multiplexers in front of reservation stations—led the 29K designers to implement just two general result buses.

Fewer result buses reduce routing overhead but require arbitration logic to allocate access to the result buses. This logic must predict which execution units will produce results on the next cycle and decide which units will be granted use of the buses. If more results are produced than the buses can accommodate, the arbitration logic must stall one or more execution units. Simulations show this rarely happens. As with the operand-selection logic mentioned above, the arbitration logic can create critical timing paths that must be carefully designed.

The reservation stations at the front of each execution unit hold waiting operations and their operands. When an operand is represented by a tag instead of the actual value (because the value was unavailable when the operation was dispatched to the reservation station), the station must monitor the result buses to catch the needed value when it becomes available. Thus, the reservation stations have tag-checking comparators that compare the decisions of the result-bus arbitration logic with tags in the station entries. If a match is detected, the reservation station uses the forwarding multiplexers to gate the value into the correct station entry. The tags are four bits long.

Microarchitecture Bottlenecks

There are a couple of potential performance bottlenecks in the superscalar 29K core. These bottlenecks represent tradeoffs between hardware complexity and peak performance potential.

One bottleneck is that the register file alone cannot support the maximum issue rate of four two-operand instructions per cycle. With only four read ports, the dispatcher will be limited to two such instructions per cycle when the reorder buffer does not supply operands.

This situation typically occurs after a mispredicted branch, when all speculative results are flushed from the reorder buffer. When the dispatcher fetches instructions from the correct branch path, the reorder buffer has no valid renamed register values, and the four read ports in the register file can become a dispatch bottleneck.

Fortunately, the lack of read ports in the register file will not limit performance much because branch prediction is frequently (at least 80%) correct on most applications. When prediction is correct, most operands are supplied either by the reorder buffer or by result forwarding directly from the result buses. Also, when a branch is mispredicted, the correct path often begins in the middle of a block of four instructions, which means the full dispatch bandwidth of four instructions/cycle is impossible anyway. Taking all effects into account, AMD says the register file read-port bottleneck has less than a 1% impact on performance.

For More Information

AMD has not announced pricing or production schedules for the new 29K chips. For more information, contact John Wilkinson at 512.602.2292.

Another bottleneck is the fact that the two result buses cannot support the maximum completion rate of up to six instructions per cycle. The maximum completion rate is also less than the maximum issue rate, and the two result buses for ALU-type operations drop the completion rate to a maximum of two results per cycle for the ALU, shifter, and multiplier execution units.

The result-bus bottleneck is probably not significant, however, because the three buses provided will typically meet the average completion rate. Also, the few percent improvement in performance that could be gained by an extra bus probably did not justify the implementation cost of the routing, extra reorder-buffer write port, and tag-checking and result-forwarding logic in the reservation stations. This is, after all, supposed to be a relatively inexpensive embedded microprocessor.

Another possible bottleneck is the limited bandwidth for retiring validated results from the reorder buffer to the register file. With only two write ports into the register file, a maximum of two results per cycle can be retired under steady-state conditions.

It is important to note, however, that a typical instruction mix will contain branches and stores, which require entries in the reorder buffer but do not retire results to the register file. The reorder buffer can actually retire four instructions per cycle in the case when only two require writing a result to the register file.

Finally, the size of the reorder buffer and the sizes of the reservation stations could be bottlenecks under certain—but probably only pathologically extreme—circumstances. For example, a long string of ALU instructions with many true dependencies could force instruction dispatch to stall for lack of ALU reservation-station entries. If the long string of dependent ALU instructions were followed by some independent memory references and branches, performance potential could be wasted.

In the end, these potential bottlenecks merely create opportunities for future upgrades to the 29K superscalar core. In addition to bigger caches, obvious ways to increase potential performance without increasing the clock rate are more result buses, more register file ports, a deeper reorder buffer, more reservation stations, and wider dispatch capability.

Extending the 29K Performance Range

Figure 2 shows the performance of the entire 29K family on a page-description-language (e.g., PostScript)

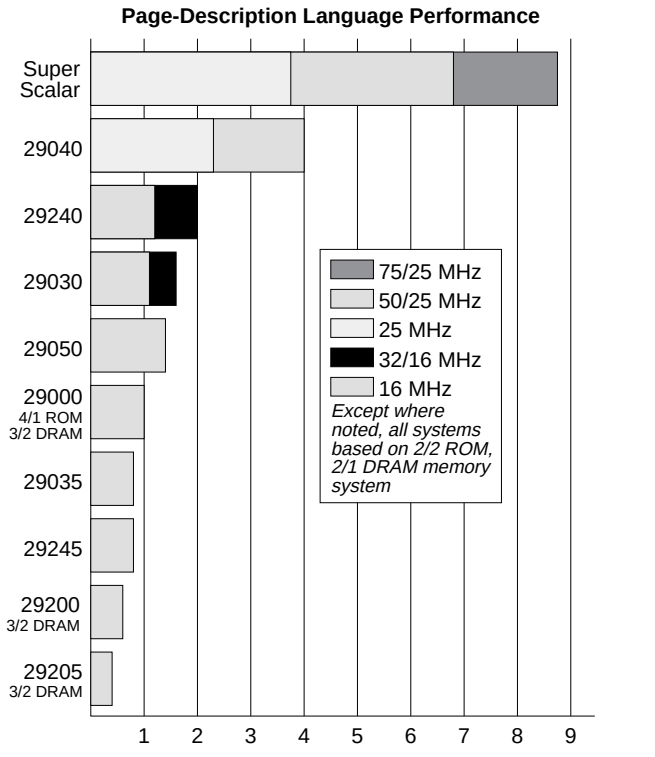

Figure 2. The superscalar implementation dramatically broadens the performance range of the 29000 family. The 25-MHz superscalar processor nearly equals the performance of the 50-MHz single-issue 29040. (Memory system description X/Y means X cycles for the first access, Y cycles for subsequent sequential burst accesses.) (Source: AMD)

benchmark. The performance shown for the superscalar chip is based on simulation. As shown, the performance available from the 50- and 75-MHz superscalar chips dwarfs all but the 29040.

The 25-MHz superscalar chip nearly equals that of the 50-MHz 20940, indicating that the overall boost in the rate of instruction execution for the superscalar organization is a little less than a factor of two.

New Core Competes with 960 H-series

The currently shipping high-end 29K and 960 chips deliver similar performance levels for many applications, despite the fact that the superscalar core used in the 960 C-series chips should be faster than the single-issue core of the current 29K chips. This is due mainly to the fact that the instruction cache in the early Intel superscalar chips is too small, which means the chip too often spends time fetching instructions—one per clock—from off-chip memory. With the 16K instruction cache in the H-series, the 960 superscalar core can achieve its potential.

The superscalar 29K core is much more general than the 960's and should be able to achieve higher performance. The 29K core can issue up to four ALU-type instructions at once, assuming the mix includes a shift and a multiply. The 960, on the other hand, can dispatch only two, and one of them must be an address-computation ALU instruction, which limits the dispatch possibilities. Further, the 29K superscalar core can execute instructions out of order and speculatively. The 960 does not allow out-of-order execution and can only issue instructions speculatively, not execute them.

Still, the performance levels of the 960 H-series and the superscalar 29K should be comparable at the same clock rate; they will both probably sustain an execution rate of just under two instructions per cycle. This is due mainly to the fact that the H-series chips have a larger instruction cache. For applications that lock critical code in the cache, the H-series chips can lock 4K, leaving a 12K instruction cache. Locking 4K of code in the 29K cache will leave only a 4K instruction cache. The 29K will reap a small relative gain over the 960 when its data cache is used in write-back mode.

Both of these chips are aimed at high-end embedded control, a market that requires some compromises to keep costs in line with market expectations. In the latest implementations, AMD chose to splurge on the superscalar core and scrimp a little on the cache, while Intel increased cache size to better exploit its existing—if somewhat limited—superscalar core.

Small Market Impact at First

AMD is not releasing many chip details now but has said it will fabricate the new 29K chips in 0.5-micron CMOS. AMD expects this process to be up and running in its new Fab 25 in Texas by the end of this year and in production by mid-95. This fab will also be producing the superscalar K5.

Because of the advanced process, the new 29K chips will operate from a 3.3-V supply. Although AMD is talking about only 25-, 50-, and 75-MHz parts now, 100-MHz chips will eventually be offered, perhaps after its 0.35-micron process (developed with HP) is available. Unlike the 960 H-series, where a different part must be purchased for each clock multiplier, AMD may decide to allow a programmable clock multiplier.

AMD justifies its high-end 29K chips with the same markets cited by Intel: printers, telecom, networking, and RAID disk controllers. Laser printers are getting faster, have higher resolution, and are moving to color. The ratio of color to monochrome printer designs has been about 1-to-10, but AMD says it is seeing the proportion of color printers increase. Faster networks are also creating a need for faster embedded controllers.

Nonetheless, the superscalar 29K, like the 960 H-series, will not make a significant market impact at first. For one thing, these chips will initially be too expensive to garner the really high-volume design wins. Gradually, though, AMD's superscalar core will move into the mainstream and ensure the continuing competitiveness of the 29K family. ♦

