Pentium Extends 486 Bus to 64 Bits

Higher Frequencies, New Features Improve Performance

By Brian Case

*Last issue, (see **070402.PDF**) we looked at Pentium's microarchitechure. This article examines Pentium's system interface.*

Intel's new Pentium processor implements a sophisticated, high-speed bus that builds on the protocols of the 486. With pipelined, back-to-back cache line fills, the 66 MHz Pentium can achieve a 528 Mbytes/s burst transfer rate—more than twice the 200 Mbytes/s of the 50-MHz 486 and four times the 66-MHz 486DX2.

While the Pentium bus is deliberately similar to the 486 bus, there are many subtle changes and a few major ones. As with some of the major changes to the x86 architecture, some of the differences between the Pentium and 486 bus structures are documented in yet another unavailable appendix (Appendix A to the *Pentium Processor User's Manual: Volume 1*).

The standard Pentium package has a total of 273 pins, with 173 signal pins and 100 power and ground pins. Table 1 lists each signal pin, its direction, and a brief pin description.

Data and Address Buses

The most obvious change from the 486 is that Pentium's data bus is 64 bits wide. This allows the larger cache lines (32 bytes vs. 16 bytes) to be filled using the same number of transfers—four—used on the 486. There are also eight parity bits that are active for both input and output of data. Pentium uses even parity. Each byte has a separate byte-enable pin, and parity is checked or driven only for the bytes that are enabled.

Data parity checking is always enabled on input, and Pentium always generates parity for enabled bytes on output. The PCHK# output is asserted if a parity error is detected on input, which allows hardware to log parity errors or signal an interrupt.

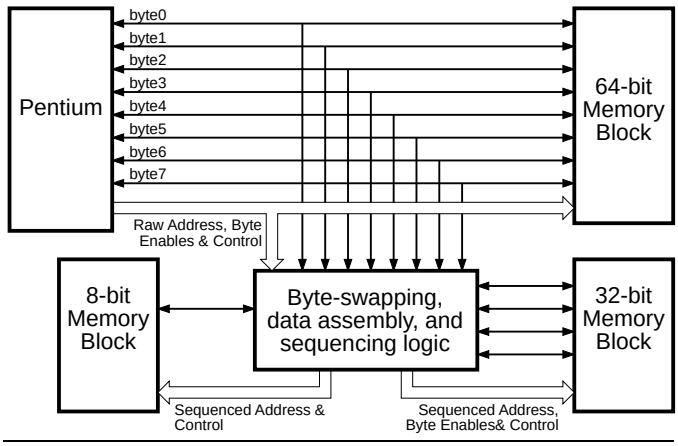

Figure 1. Memories narrower than 64 bits require external logic.

Pentium can also be configured to automatically cause an internal exception on parity errors. This exception can be blocked either by disabling the machine-check exception (via the MCE bit in CR4) or by deasserting PEN on a cycle-by-cycle basis. Thus, it is possible to have Pentium automatically take action on parity errors or have external hardware decide when to interrupt Pentium (or both).

The address bus consists of 29 address lines and the eight byte enables just mentioned. Parity is checked on the address lines, but only A31 through A5 participate; A4 and A3 are not checked. This is apparently due to the fact that only A31–A5 are used for cache snooping operations (described later). Address-bus parity errors are signaled by the APCHK# signal. Since it is not possible to cause an internal exception as a result of an address parity error, external hardware must be used to either deal with the problem or cause an interrupt.

Narrow Bus Issues

Pentium has a 64-bit bus, but some system implementations will probably prefer a 32-bit memory system. Unfortunately, Pentium requires that all of the enabled bytes for a given cycle be returned from memory to the processor simultaneously. Thus, for narrower memories, such as 32-bit RAM and byte-wide bootstrap PROMs, external logic is required to sequence addresses, swap bytes, and buffer data as shown in Figure 1.

Pentium does not provide the BS8# and BS16# inputs (8-bit and 16-bit bus-size indicators) that allow the 486 to work easily with narrow devices. To continue the tradition, Pentium would have had to add a third pin, BS32#. Presumably, simplifying Pentium's bus controller led the designers to eliminate bus sizing.

For most systems, this is probably not much of an issue since the required logic can, and therefore will, be incorporated into 32-bit Pentium chip sets. Also, Intel will eventually—though not in 1993—provide narrow-bus versions of Pentium for specific markets.

Bus Cycles and Timing

A Pentium bus cycle begins with the assertion of ADS#. At the same time ADS# is asserted, the address and status are driven. A bus cycle ends when the last BRDY# is returned. Each bus cycle may consist of one or

 © 1993 MicroDesign Resources

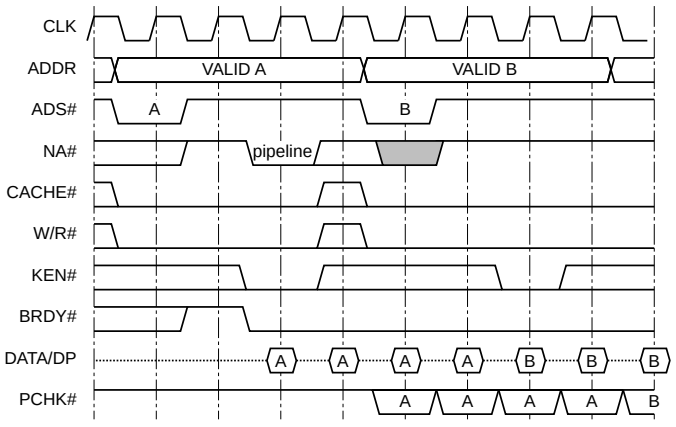

Figure 2. Pentium timing diagram for two pipelined cache line fills.

four data transfers. Bursts are always four transfers.

On the 486, the difference between a simple bus cycle and a burst cycle is determined by the acknowledgement: RDY# for simple or BRDY# for burst, with RDY# taking precedence. On Pentium, the difference between simple and burst cycles is determined by cacheability. A cacheable transaction is a burst of four 64-bit data transfers; all others are single, simple data transfers of 64 bits or less. Consequently, burst support is required in Pentium systems, while 486 systems can choose to implement burst transactions to improve performance or leave it out to simplify the system design. This requirement will affect Pentium system designers little since chip sets will provide the burst support.

Bus pipelining, missing on the 486, allows Pentium to begin a new external access while a previous access is still uncompleted. Pentium supports up to two pending bus cycles with the NA# signal.

Figure 2 shows a bus timing diagram for two back-to-back cache line fills that are pipelined. Each four-transfer cache-fill cycle is begun by simultaneously driving an address and asserting ADS#. Since CACHE# is asserted and KEN# is returned with the first BRDY#, the data is cacheable and the cycle will be a four-transfer line fill. NA# is asserted to pipeline the next line fill. Two cycles after NA#, the next address and ADS# are asserted. KEN# is asserted along with the first BRDY# of the second line fill. The result is two cache line fills that can proceed at the full bus speed of eight bytes every cycle.

In burst cycles—either cache-line fills or write-backs—Pentium supplies only the first address. For a cache line fill, Pentium supplies the address of the data requested by the program; for a write-back, the first address is the address of the first 64-bit word in the line. The other three addresses for the burst line fill or write-back are expected to be computed by external hardware according to Table 2, which shows the hex value of the low five address bits. For example, if a program requests a data word with the low five address bits equal to 0x08

Address		
A31..A3	I/O	Address bus
BE7#..BE0#	O	Byte enables
A20M#	I	Address bit 20 Mask
AP	I/O	Address bus parity (even)
APCHK#	O	Address parity error
Data		
D63..D0	I/O	Data bus
DP7..DP0	I/O	Byte parity bits (even)
PCHK#	O	Data bus parity error
PEN#	I	Data bus parity check enable
Bus Control		
M/IO#	O	Memory or I/O bus cycle
D/C#	O	Data or Code bus cycle
W/R#	O	Write or Read bus cycle
CACHE#	O	Read: will cache; Write: burst write-back
SCYC	O	Split cycles for LOCKed transfer
LOCK#	O	Locked (indivisible) bus cycle
ADS#	O	Start of new bus cycle
BRDY#	I	Burst ready
NA#	I	Next Address (for external bus pipelining)
BUSCHK#	I	Bus check, bus cycle complete unsuccessfully
BOFF#	I	Backoff, abort all outstanding bus cycles
BREQ	O	Bus request, internal Pentium bus request
HOLD	I	Bus hold request
HLDA	O	Bus hold acknowledge
Cache Control		
KEN#	I	Cache enable, current cycle cacheable
WB/WT#	I	Write-back/write-through, line-by-line control
AHOLD	I	Address hold, address bus floating next cycle
EADS#	I	Valid external address for Inquire cycle
HIT#	O	Hit, result of Inquire cycle
HITM#	O	Hit in modified line, result of Inquire cycle
INV	I	Invalidate if Inquire cycle hits in cache
FLUSH#	I	Flush and write-back cache contents
EWBE#	I	External write buffer empty
PCD	O	Page cache disable, for external cache
PWT	O	Page write through, for external cache
System Management Mode		
SMI#	I	System management interrupt request
SMIACT#	O	System management mode active
Execution Tracing		
BT3..BT0	O	Branch trace, 3 LSBs of target & special cycle
IU, IV	O, O	U-pipe, V-pipe instruction completion
IBT	O	Instruction branch taken
BP3..BP2	O	Breakpoint pins for debug registers 3, 2
PM1/BP1	O	Breakpoint/performance monitoring pin 1
PM0/BP0	O	Breakpoint/performance monitoring pin 0
Miscellaneous		
INTR, NMI	I, I	Maskable interrupt, non-maskable interrupt
IERR#	O	Internal error (parity or functional redundancy)
FERR#, IGNNE#	O, I	FP error (like ERROR# in '387), ignore FP error
FRCMC#	I	Functional redund. check master/checker mode
TCK, TMS, TDI,	I, I, I	JTAG clock, mode select, data input
TDO,TRST#	O, I	JTAG data output, reset
R/S#, PRDY	I, O	Intel debug port support (run/stop, stop ack.)
CLK, RESET	I, I	Processor clock, processor reset
INIT	I	Processor initialization (warm reset)

Table 1. Descriptions of Pentium's signal pins.

 © 1993 MicroDesign Resources

1st Address	2nd Address	3rd Address	4th Address
0x00	0x08	0x10	0x18
0x08	0x00	0x18	0x10
0x10	0x18	0x00	0x08
0x18	0x10	0x08	0x00

Table 2. Pentium burst address order (addresses are in hex).

and the data cache misses, Pentium supplies the first address, but external hardware must return the next three 64-bit words from addresses 0x00, 0x18, and 0x10 respectively. The patterns shown in Table 2 are the same as for the 486.

Table 3 lists the bus cycles that can be initiated by Pentium and how the bus signals encode them. Note that cycles consist of four transfers if and only if data is cacheable (CACHE# and KEN# asserted). Another type of bus cycle, the inquire cycle (described below), can be generated by the external system by asserting EADS#.

The special bus cycles, listed in Table 4, are provided to indicate that certain instructions have been executed or that certain conditions have occurred internally. As shown in Table 3, a special bus cycle is encoded as the impossible case of a write to code in I/O space (M/IO# and D/C# = 0, W/R# = 1). During special cycles, the data bus is undefined and address lines A31–A3 are driven to zero. Special bus cycles are acknowledged with BRDY#.

The *shutdown* special cycle can be generated if Pentium gets an exception while it is invoking the double-fault handler or if an internal parity error is detected. The *halt* special cycle is driven after a HLT instruction is executed. The halt state is like shutdown except that halt can be exited by maskable interrupts.

The *flush* special cycle is driven after the INVD (invalidate cache) or WBINVD (write-back and invalidate cache) instructions are executed. The *flush-acknowledge* special cycle indicates the completion of the cache flush operation in response to the assertion of the FLUSH# pin. This operation is implemented as an interrupt to a microcode routine.

The *write-back* special cycle is driven after the WBINVD instruction is executed to indicate that lines marked "modified" in the Pentium data cache were written back to memory or a second-level cache and that lines marked "modified" in any external caches should then be written back as well.

The *branch trace message* special cycle is driven every time a branch is taken if the execution-tracing enable bit in TR12 (test register 12) is set to one (IBT is asserted on taken branches regardless). This special cycle is the only one that does not drive zeros on the address bus; instead, the address bus and BT2–BT0 contain the branch target linear address.

M/IO#	D/C#	W/R#	CACHE#	KEN#	Cycle Description	# of Transfers
0	0	0	1	X	Interrupt acknowledge (2 locked cycles)	1 each cycle
0	0	1	1	X	Special cycle (see Table 4)	1
0	1	0	1	X	I/O read, 32 bits or less, non-cacheable	1
0	1	1	1	X	I/O write, 32 bits or less, non-cacheable	1
1	0	0	1	X	Code read, 64 bits, non-cacheable	1
1	0	0	0	1	Code read, 64 bits, non-cacheable	1
1	0	0	0	0	Code read, 256-bit burst line fill	4
1	0	1	X	X	Intel reserved (will not be driven by Pentium)	n/a
1	1	0	1	X	Memory read, 64 bits or less, non-cacheable	1
1	1	0	0	1	Memory read, 64 bits or less, non-cacheable	1
1	1	0	0	0	Memory read, 256-bit burst line fill	4
1	1	1	1	X	Memory write, 64 bits or less, non-cacheable	1
1	1	1	0	X	256-bit burst writeback	4

Table 3. Pentium-initiated bus cycles. Note that CACHE# will not be asserted for a cycle in which M/IO# is driven low or when PCD is asserted high.

Internal Cache Snooping

To maintain cache coherency in both single- and multiple-processor systems, Pentium implements a MESI cache consistency protocol (see µPR 6/20/90, p.12) with both internal and external cache snooping. Internal snooping occurs under three conditions.

First, an internal snoop is conducted if a miss is detected in the instruction cache. If the snoop hits in the data cache and the accessed line is in either the S (shared) or E (exclusive) state, the line is simply invalidated. If the accessed data cache line is in the M (modified) state, the line is first written back and then invalidated in the data cache. In all cases, the original instruction-cache miss is satisfied by a cache line fill from external memory (RAM or second-level cache).

Second, an internal snoop to the instruction cache occurs for internal data cache misses. If the snoop hits in the instruction cache, the line in the instruction cache is invalidated. These first two cases handle self-modifying code.

Third, an internal snoop to both caches occurs if there is a write to the "accessed" and/or "dirty" bits in the TLB entries. If the snoop hits in either or both caches, the accessed lines are invalidated. If the accessed line in the data cache is in the M state, it is written back first. This is done because the in-cache copies are stale after the change is made by the MMU to both the TLB entries and the page-table entries in memory.

External Cache Snooping

External cache snooping occurs when the system asserts EADS# to re-

 © 1993 MicroDesign Resources

BE7#	BE6#	BE5#	BE4#	BE3#	BE2#	BE1#	BE0#	Special Bus Cycle
1	1	1	1	1	1	1	0	Shutdown
1	1	1	1	1	1	0	1	Flush (INVD, WBINVD instr.)
1	1	1	1	1	0	1	1	Halt
1	1	1	1	0	1	1	1	Writeback (WBINVD instruction)
1	1	1	0	1	1	1	1	Flush acknowledge (FLUSH#)
1	1	0	1	1	1	1	1	Branch trace message

Table 4. Encoding of special bus cycles.

quest a cache consistency check called an "inquire" cycle. Inquire cycles could be used to keep caches and memory consistent during DMA transfers or during cache miss processing in multiprocessor systems. Since the external system must supply Pentium with a snoop address via the address bus, Pentium must first be told via AHOLD to float its address bus. AHOLD must be asserted a minimum of two cycles before EADS# is driven active.

The inquire cycle can have two goals: to simply discover if Pentium has an on-chip copy of data, or to cause Pentium to invalidate any on-chip copy. Asserting the INV pin will cause Pentium to invalidate on-chip copies if the snoop hits.

Driving the snoop address and asserting EADS# and INV are done simultaneously (two cycles after AHOLD) to start an inquire cycle. Since an entire cache line is affected by an inquire cycle, only address lines A31 through A5 are significant, but the others must be driven to a valid logic level for electrical reasons. The AHOLD/EADS# sequence can be performed even while Pentium is processing a data transfer (the data transfer in progress is not interrupted).

The external system is informed of a snoop hit in the on-chip caches through the HIT# and HITM# signals. These signals are valid two cycles after the assertion of EADS#. HIT# is always asserted if a hit occurred, while HITM# is asserted only if the snoop hits a data-cache line in the M state.

If an inquire cycle hits an M-state line in the data cache, the modified data in the accessed line will be written back immediately so that the line can be invalidated. Figure 3 shows a timing diagram for this case in which INV is asserted at the start of the snoop.

At the end of cycle 1, EADS# and INV are asserted to request an inquire cycle with invalidation. At the end of cycle 2, a previous data transaction is completed. At the end of cycle 3, HIT# and HITM# are asserted to indicate a hit, indicating that Pentium will start a write-back cycle with the next assertion of ADS#. (The only reason ADS# can be asserted during AHOLD is for an inquire-induced write- back.) At the end of cycle 5, ADS#, CACHE# and W/R# are all driven to signal the start of the write-back. The four write transfers follow. HITM# stays asserted until two cycles after the last BRDY# of the write-back.

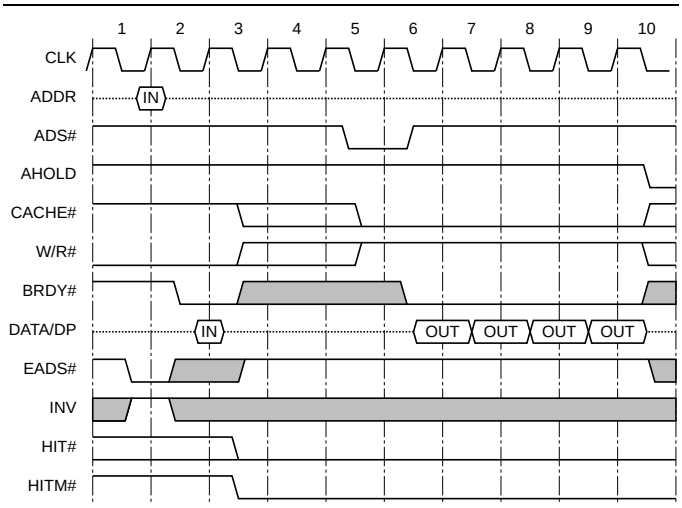

Figure 3. Pentium inquire cycle that invalidates and writes back an M-state cache line. ("IN" and "OUT" are with respect to Pentium.)

Since AHOLD is asserted during the entirety of this write-back transaction, Pentium is unable to drive addresses to the external system. Thus, in this case, the external system is required to drive and sequence all address bits for the write-back data transfers.

If desired, however, the external system can deassert AHOLD before Pentium begins the write-back cycle (before cycle 5 in Figure 3) to cause Pentium to drive the write-back address on the address bus. This can be done to simplify external hardware a little or to account for the possibility of an address parity error on an inquire cycle (see below). Even in this case, the external system is still responsible for sequencing addresses (as in all burst transactions).

Inquire cycles always snoop the internal instruction and data caches, but if the snoop is requested during a cache line fill, Pentium also snoops the line currently being filled (in a read buffer). If more than one cacheable cycle is outstanding because of address pipelining, Pentium snoops both transactions.

Similarly, if an M-state line is in a write buffer in the process of being written back, Pentium will snoop the write buffer on behalf of an inquire cycle. In this case, Pentium asserts HIT# and HITM# as usual, but there will not be a separate write-back of the M-state line since it was already in progress.

Address parity is checked for inquire cycles, but Pentium can do nothing about parity errors; if an address parity error occurs, the snoop cycle is not inhibited. If an inquire hits an M-state line and AHOLD remains asserted, it is not possible for Pentium to drive the address bus to tell the system what address was actually used for the snoop. Thus, it is possible that Pentium will start a write-back of incorrect data, and if the external system uses the address it supplied to Pentium for the inquire cycle, memory could be corrupted. In light of how much

 © 1993 MicroDesign Resources

Pentium and 486 Bus Differences

- Pentium has a 64-bit data bus; the 486 has a 32-bit data bus. Pentium has eight byte enables and eight parity pins. Pentium supports address parity.
- Pentium supports address pipelining via NA#.
- Pentium samples KEN# on the earlier of NA# or the first BRDY#. KEN# is sampled only once. The 486 samples KEN# both the clock before the first RDY#/BRDY# and the clock before the last RDY#/BRDY# of a cache-line fill cycle.
- Pentium implements system management mode via SMI# and SMIACT#.
- On Pentium, a bus cycle aborted via BOFF# is restarted from the beginning, and data returned previous to BOFF# is not saved. The 486 saves data returned before BOFF# is asserted and restarts the cycle from where it left off.
- Pentium signals burst length via the CACHE# pin together with the address (only bursts of four are supported). The 486 controls burst length via the BLAST# pin.
- Pentium does not have the 486's PLOCK# pin since eight-byte writes are performed as a single bus cycle.
- Pentium does not support any dynamic bus sizing; specifically, Pentium does not have the 486's BS8# and BS16# pins.
- Pentium does not change low-order address bits and byte enables during a burst.
- Pentium requires write-backs and cache line fills to be burst cycles, and since Pentium has no RDY# and BLAST# pins, the burst cannot be prematurely terminated.
- On Pentium, cacheable implies burstable. Pentium does not support non-cacheable burst cycles or cacheable non-burst cycles.
- Pentium supports a write-back cache protocol via the following new pins: CACHE#, HIT#, HIM#, INV, and WB/WT#.
- Pentium does not allow invalidations every clock or invalidations while Pentium is driving the address bus.
- Pentium guarantees an idle bus cycle between consecutive LOCKed cycles.
- Pentium supplies SCYC to indicate that a locked operation involves a split cycle.
- A non-cacheable code prefetch is eight bytes for Pentium vs. sixteen bytes for the 486.
- Pentium has INIT to perform a reset while maintaining internal cache and FP register state.
- Pentium supports strong store ordering via EWBE#.
- Pentium supports internal parity checking, selective data-bus parity checking on a cycle-by-cycle basis, and address-bus parity checking via the new pins APCHK#, BUSCHK#, PEN#, IERR#, and AP.
- Pentium implements boundary scan via TDI, TDO, TMS, TRST#, and TCK.
- Pentium supports external execution tracing via IU, IV, IBT, and a branch-trace message special cycle.
- Pentium supports functional redundancy checking via FRCMC# and IERR#.
- Pentium supports performance monitoring and external breakpoint indications via BP3, BP2, PM1/BP1, and PM0/BP0.
- On Pentium, FLUSH# is implemented as an interrupt, is edge-triggered, and is recognized once for every falling edge. Since it is an interrupt, it is recognized only at instruction boundaries.

Intel is making of Pentium's error-checking capabilities, it seems odd that address-parity errors are not handled more gracefully.

External Program Monitoring

As on all highly integrated processors, it is difficult to monitor program behavior on Pentium in detail because so much activity is occurring only between on-chip components. To address this problem, Pentium has many pins that expose internal operations and allow external program monitoring. These pins include: BP[3:0] (breakpoint) and PM[1:0] (performance monitoring), BT[3:0] (branch trace), IBT (instruction branch taken), and IU and IV (instruction completed in pipelines).

The four breakpoint pins correspond to internal debug registers, and the pins are asserted when a match is detected in the corresponding debug register. While BP3 and BP2 are dedicated, BP1 and BP0 are multiplexed with PM1 and PM0. Unfortunately, the PM pin functions are covered in the secret Appendix A.

IBT is asserted each time Pentium takes a branch. If enabled, each assertion of IBT is accompanied by a special bus cycle, the branch trace message special cycle (see Tables 3 and 4). On each of these cycles, the four BTx bits are also valid. They provide the low three bits of the branch target address (unavailable on the address bus) and tell whether the branch was a 16-bit or 32-bit instruction.

IU and IV simply indicate each instruction completion in the respective instruction pipeline. Note that IBT will be accompanied by either IU or IV and that IU and IV can be (and, it is hoped, often are) asserted simultaneously.

The INIT pin is a new "warm restart" pin that causes a reset-like action but does not cause the values in caches and FP registers to be lost. INIT can be used to switch via hardware from protected mode to real mode. Also, holding INIT high during reset invokes an automatic built-in self-test mode.

Conclusions

In some ways—elimination of RDY#, simpler burst determination, and no bus sizing—Pentium has a simpler bus than the 486. The addition of new features, however, such as pipelining, cache snooping, and forced burst support, mean that system hardware will need considerably more sophisticated bus controllers, especially if second-level caches are used in multiprocessor systems. As with earlier x86 generations, chip-set vendors will come to the rescue by sparing system makers the headache of designing bus-control state machines and interface logic.♦

 © 1993 MicroDesign Resources