The Pentium

M. Sonza Reorda

Politecnico di Torino
Dipartimento di Automatica e Informatica

Summary

- **Introduction**
- **The Pentium pipelines**
- **Branch prediction**
- **Caches**
- **Test and identification features**
- **The bus cycles.**

Introduction

The first Pentium devices (previously denoted as P5) were released in 1993.

The Pentium succeeded in combining two opposite characteristics:

- It is a superscalar processor based on RISC technology
- It is fully compatible with previous Intel architectures (e.g., 486).

After the first versions, significant changes were introduced with the P54C series.

Main characteristics

- Superscalar architecture, based on two integer pipelines (u- and v-) and one FP pipeline
- Dynamic branch prediction
- Separate data and instruction caches, each 8Kb large
- 64-bit data bus and 32-bit address bus.

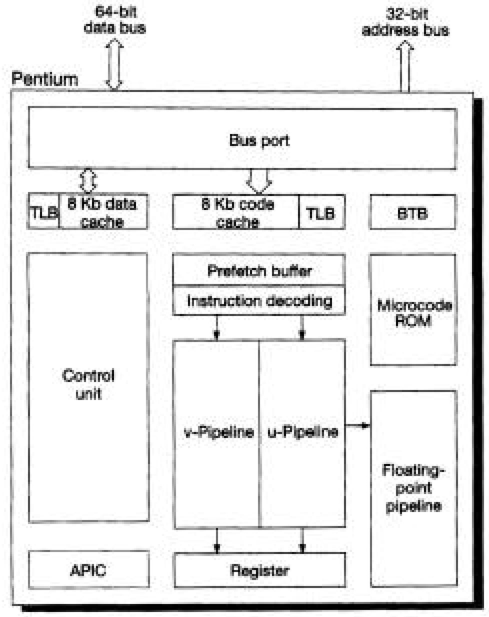
Electrical characteristics

- About 3 million transistors
- Staggered Pin Grid Array (SPGA) with 296 pins
- 53 GND and 53 Vcc pins
- 13 W consumption (need for effective heat sink).

Internal structure

- Internally, the Pentium is a 32-bit processor
- The external data bus is 64-bit wide to support fast exchange of data between memory and caches
- Internal busses are between 128 and 256 bits wide.

Block diagram



Integer pipelines

The two integer pipelines contain 5 stage each:

- **Instruction fetch (IF)**
- **Decoding 1 (D1)**
- **Decoding 2 (D2)**
- **Execution (EX)**
- **Register write-back (WB).**

The u-pipeline can execute all x86 instructions.

The v-pipeline can only execute "simple integer instructions".

Under certain conditions, the two pipelines can work in parallel, thus allowing two results to leave the pipeline in the same clock cycle.

IF stage

At each clock cycle it collects two instructions either from the code cache, or from the external memory.

If one of the fetched instructions is a jump, the Branch Target Buffer is activated, which

- predicts whether the branch is going to be taken or not
- In the former case, it gives the address where to fetch the following instruction.

If the prediction is later found to be false, the pipeline is then emptied.

ID1 stage

It decodes two instructions per clock cycle and determines whether the two can be paired (i.e., sent in parallel to the u- and v-pipeline).

In the case of prefixes, a further clock cycle is required.

ID2 stage

It determines operand addresses.

All addresses, even the more complex (e.g., base indexed) are computed in a single clock cycle.

EX stage

It performs the ALU operations and cache accesses.

If both a cache access and an ALU operation are required (e.g., `ADD eax, mem32`), more than one clock cycle is required.

It also checks for the correctness of some branch predictions performed in the IF stage.

WB stage

It writes back the instruction results in the destination registers.

It also checks for the correctness of the branch predictions for conditional branches.

Exceptions

If an exception, or a software interrupt is raised, it is recognized when the instruction is in the EX stage.

Similarly, external interrupt requests are monitored at every clock cycle.

When one of these conditions is detected, all the instructions in the EX stage or in the following pipeline stages of both pipelines are completed, while those in the IF, D1 and D2 stages are deleted.

Instruction pairing

The Pentium fetches two instructions at the same time and check whether they can be executed in parallel by the two pipelines.

For sake of compatibility, no special compiler intervention is required on the code accepted by the Pentium to support instruction pairing.

This condition is checked using 6 pairing rules.

There are several exceptions to the rules.

Rule 1

Both instructions in a pair must be *simple*.

Simple instructions include

```
MOV  reg, reg/mem/imm
MOV       mem, reg/imm
ALU       reg, reg/mem/imm
ALU       mem, reg/imm
INC       reg/mem
DEC       reg/mem
PUSH reg/mem
POP       reg/mem
NOP
```

Rule 2

Unconditional jumps (JMP), conditional jumps (Jcc) near, and function calls can be paired only if they occur as the second instruction in the pair.

In fact, they can only be carried out by the v-pipeline.

Rule 3

***Shift* and *rotate* instructions can be paired only if they occur as the first instruction in the pair (they can only be carried out by the u-pipeline).**

Rule 4

No register dependencies can occur between the instructions in an instruction pair.

If a direct or indirect register dependency occurs, instructions are executed in sequence (and not in parallel).

Rule 5

Instructions with a prefix (e.g., segment override) can be paired only if they occur as the first instruction in the pair (they can only be carried out by the u-pipeline).

Rule 6

The two instructions in the pair can not simultaneously refer to a displacement and an immediate operand (e.g., `ADD array[02h], 08h`).

Complex instructions

Complex instructions (e.g., string operations, task switching, LOOP, etc.) are microcoded, i.e., they correspond to a series of microinstructions coded in a ROM.

Pentium microcodes have been optimized, and use both pipelines in parallel as far as possible.

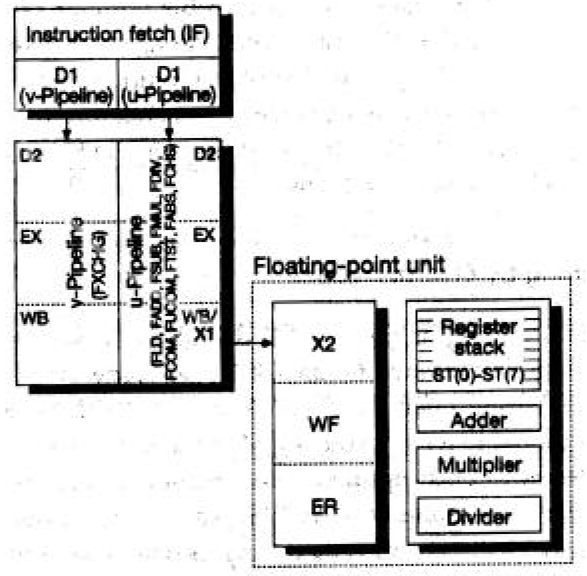
The FP pipeline

The Pentium includes a FP unit, and FP instructions can be executed in a pipelined way.

The FP pipeline contains 8 stages; the first 5 stages are in common with the u-pipeline.

The pairing rules prevent the parallel execution of integer and FP instructions.

FP pipeline



Details about the FP pipeline

For FP instructions the WB stage acts as the first execution stage (X1).

It converts an operand read from the data cache (or memory) in the temporary real format and writes it in the FP(0) to FP(7) registers.

X1 also identifies "*safe*" instructions, i.e., those that do not cause overflow, underflow or exceptions.

If an instruction is safe, the following FP one can enter the X1 stage as soon as the previous moved to X2. Otherwise it must wait until the previous instruction moved out of the ER stage.

FP instruction pairing

Only the FXCHG instruction can (under certain conditions) be executed in parallel (by the v-pipeline) with another FP instruction.

Branch prediction

The Pentium includes a branch prediction unit based on a Branch Target Buffer (BTB).

The BTB is a cache composed of 256 entries.

The cache is written each time a jump instruction is found. In this case

- The target address is stored in the cache line
- The address of the instruction preceding the jump one is stored in the tag field.

BTB access

When an instruction reaches the D1 stage, its address is sent to the BTB:

- In the case of a miss, nothing happens
- In the case of a hit, the BTB provides the target address (if the branch was predicted taken), which is immediately sent to the IF stage.

The correctness of the prediction is checked either in the EX, or in the WB stage; in case of misprediction, the whole pipeline is emptied.

On-chip caches

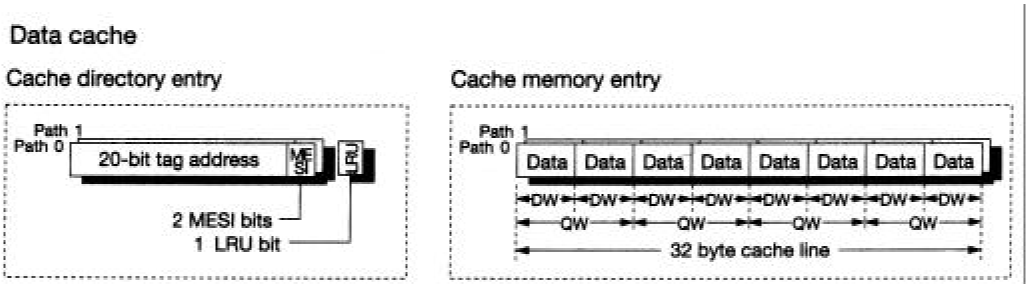
The Pentium includes two 8 Kb caches for data and code.

In this way no collisions are possible between instruction prefetching and data accesses performed by different instructions in the pipeline.

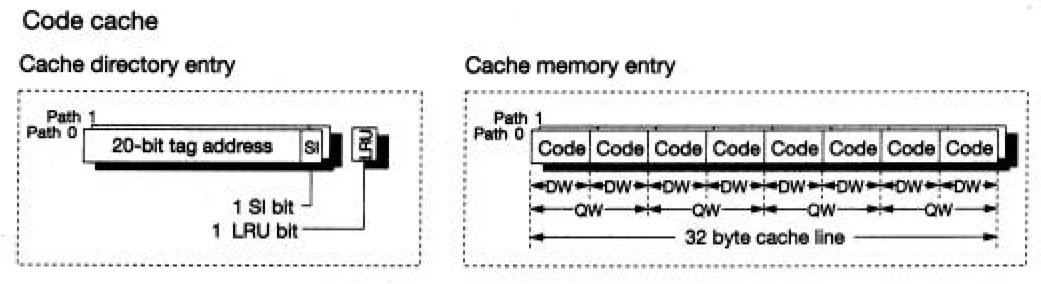
Each cache adopts a 2-way set-associative strategy with 128 sets and 32-byte lines.

Using burst mode, a cache line can be read or written from/to memory within 4 transfer cycles.

Data cache



Code cache

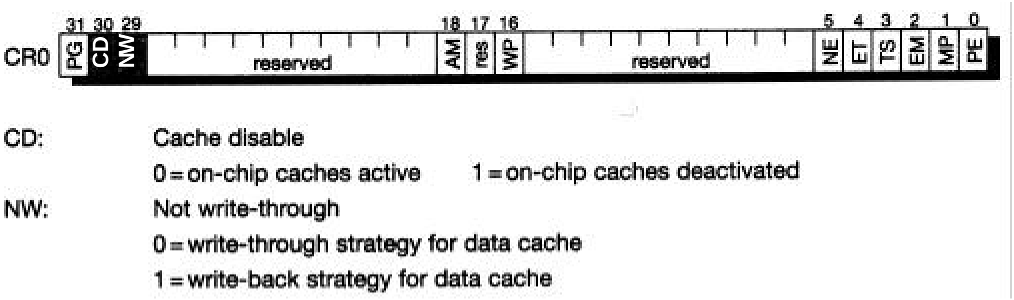


Cache usage customization

It can be performed by suitably setting

- **The CD and NW bits in the CR0 control register**
- **the PCD and PWT bits in the page directory or page table entries.**

CD and NW bits



CD: Cache disable

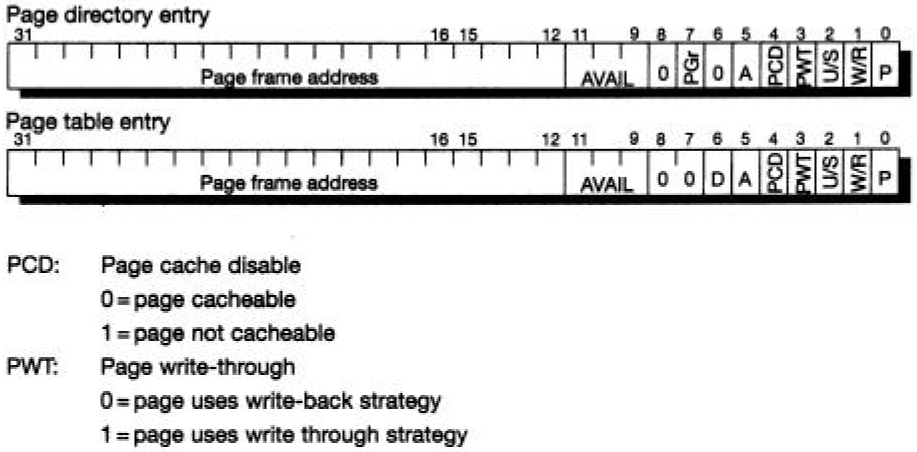
0 = on-chip caches active 1 = on-chip caches deactivated

NW: Not write-through

0 = write-through strategy for data cache

1 = write-back strategy for data cache

PCD and PWT bits



PCD: Page cache disable

0 = page cacheable

1 = page not cacheable

PWT: Page write-through

0 = page uses write-back strategy

1 = page uses write through strategy

KEN signal

When the Pentium addresses a memory word, two situations may arise:

- The KEN signal is activated by the memory subsystem, indicating that the block can be cached: the cycle is expanded to a cache line fill cycle
- The KEN signal is not activated, which means that the memory block can not be cached: the word is read, but not written in the cache.

WB/WT signal

This signal is properly driven by the memory subsystem during memory access cycles.

It indicates whether the accessed line should be managed according to a write-back or write-through strategy.

It is used for the implementation of the MESI protocol.

MESI protocol

It is adopted to guarantee cache coherency if other caches are present in the system.

The specific meaning assigned to each state is reported in the following.

M state

An M-state line is available in only one cache and it is also modified (i.e., different from main memory). An M-state line can be accessed (read/written to) without sending a cycle out on the bus.

E state

An E-state line is available in only one cache in the system, but the line is not modified (i.e., it is the same as in main memory).

An E-state line can be accessed (read/written to) without generating a bus cycle. A write to an E-state line causes the line state to change to Modified.

S state

This state indicates that the line is potentially shared with other caches (i.e., the same line may exist in more than one cache).

A read to an S-state line does not generate bus activity, while a write to a SHARED line generates a write-through cycle on the bus.

The write-through cycle may invalidate this line in other caches. A write to an S-state line updates the cache.

I state

This state indicates that the line is not available in the cache.

A read to this line will be a MISS and may cause the processor to execute a line fill.

A write to an INVALID line causes the processor to execute a write-through cycle on the bus.

Data and code caches

Each line of the data cache is associated one out of 4 states (M, E, S, I).

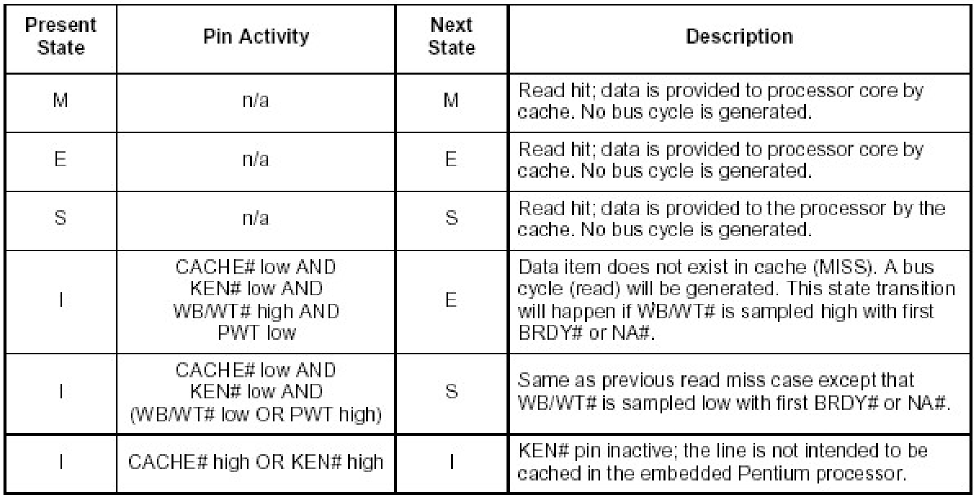
Each line of the code cache is associated one out of 2 states (S, I).

State transitions

Lines cached in the processor can change state because of processor-generated activity or as a result of activity on the processor bus generated by other bus masters (*snooping*).

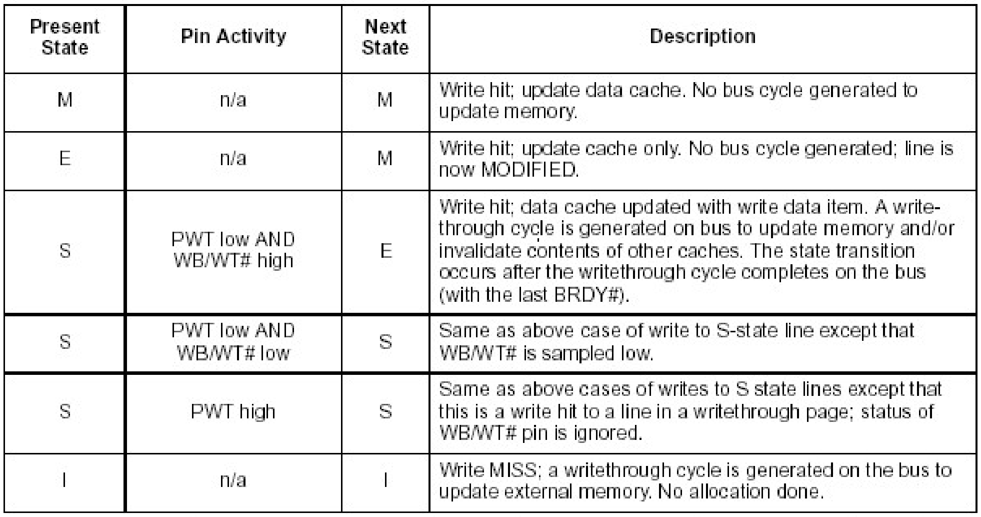
State transitions happen because of processor-generated transactions (memory reads/writes) and based on a set of external input signals and internally generated variables.

Transitions caused by read cycles



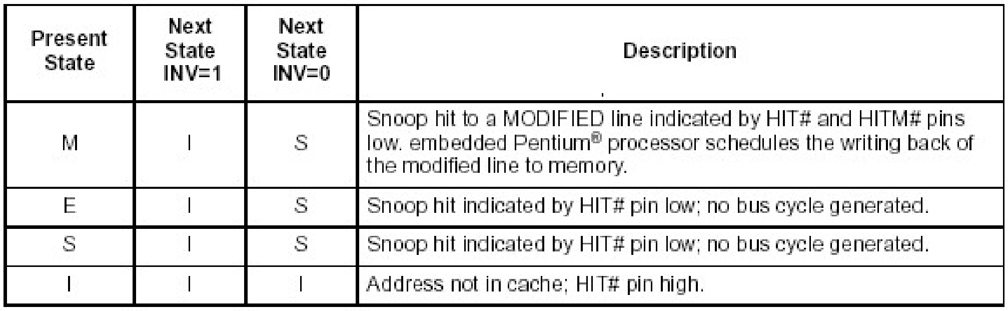
Present State	Pin Activity	Next State	Description
M	n/a	M	Read hit; data is provided to processor core by cache. No bus cycle is generated.
E	n/a	E	Read hit; data is provided to processor core by cache. No bus cycle is generated.
S	n/a	S	Read hit; data is provided to the processor by the cache. No bus cycle is generated.
I	CACHE# low AND KEN# low AND WB/WT# high AND PWT low	E	Data item does not exist in cache (MISS). A bus cycle (read) will be generated. This state transition will happen if WB/WT# is sampled high with first BRDY# or NA#.
I	CACHE# low AND KEN# low AND (WB/WT# low OR PWT high)	S	Same as previous read miss case except that WB/WT# is sampled low with first BRDY# or NA#.
I	CACHE# high OR KEN# high	I	KEN# pin inactive; the line is not intended to be cached in the embedded Pentium processor.

Transitions caused by write cycles



Present State	Pin Activity	Next State	Description
M	n/a	M	Write hit; update data cache. No bus cycle generated to update memory.
E	n/a	M	Write hit; update cache only. No bus cycle generated; line is now MODIFIED.
S	PWT low AND WB/WT# high	E	Write hit; data cache updated with write data item. A write-through cycle is generated on bus to update memory and/or invalidate contents of other caches. The state transition occurs after the writethrough cycle completes on the bus (with the last BRDY#).
S	PWT low AND WB/WT# low	S	Same as above case of write to S-state line except that WB/WT# is sampled low.
S	PWT high	S	Same as above cases of writes to S state lines except that this is a write hit to a line in a writethrough page; status of WB/WT# pin is ignored.
I	n/a	I	Write MISS; a writethrough cycle is generated on the bus to update external memory. No allocation done.

Transitions caused by inquiry cycles



Present State	Next State INV=1	Next State INV=0	Description
M	I	S	Snoop hit to a MODIFIED line indicated by HIT# and HITM# pins low. embedded Pentium® processor schedules the writing back of the modified line to memory.
E	I	S	Snoop hit indicated by HIT# pin low; no bus cycle generated.
S	I	S	Snoop hit indicated by HIT# pin low; no bus cycle generated.
I	I	I	Address not in cache; HIT# pin high.

Real, protected and virtual mode

Real and protected mode are the same as in the 486.

Virtual mode was expanded to better and more efficiently support the interrupt mechanism activated by original DOS programs.

Test features

The Pentium supports some features intended to allow the test of the processor and of the rest of the system:

- A *self-test procedure* can be activated at the reset, which test using BIST (Built-In Self-Test) the internal memories (PLAs, caches, TLBs, BTB)
- The Pentium can enter a *functional redundancy checking* mode (FRC) that allows it to verify the signals produced by a similar processor and activating an error signal if a difference is detected
- The Pentium can enter a *tristate* mode, so that it disconnects from the rest of the system, which can be more easily tested.

CPU identification

Thank you to the newly introduced `CPUID` instruction, the software can easily get information about the processor model and version.

The Pentium bus

Due to the efficiency of its internal pipelines, the memory access is a potential bottleneck of the Pentium.

To solve this

- The data bus has been extended to 64 bits
- An L2 cache is supposed to exist outside
- Bus cycles are further optimized.

The address bus includes A3 to A31 bits; bits A0 to A2 must be generated from the BE0 to BE7 signals.

Parity

To allow detection of possible errors during bus transmission, the Pentium supports additional parity bits to protect the data and address busses:

- AP is the parity bit generated by the Pentium to protect the address bus
- DP7 to DP0 are the parity bits (one for every data byte) generated by the Pentium or by the memory subsystem to protect the data bus.

Access to bytes, words and double words

Although the data bus is 64-byte wide, access to individual bytes is still possible thanks to the enable signals BE0 to BE7.

Accesses to memory objects which span the quad word boundary are split into two consecutive accesses.

Memory accesses are mainly performed for cache line fill or write-back (hence, in blocks of 32 consecutive bytes).

Access to smaller memory objects is mainly required when I/O ports are involved.

Single transfer cycle

It implements read and write operations on 8-, 16-, 32- or 64-bit data.

It requires at least two clock cycles; when 64-bit data are transferred, this corresponds to a maximum bandwidth of 264 Mb/s.

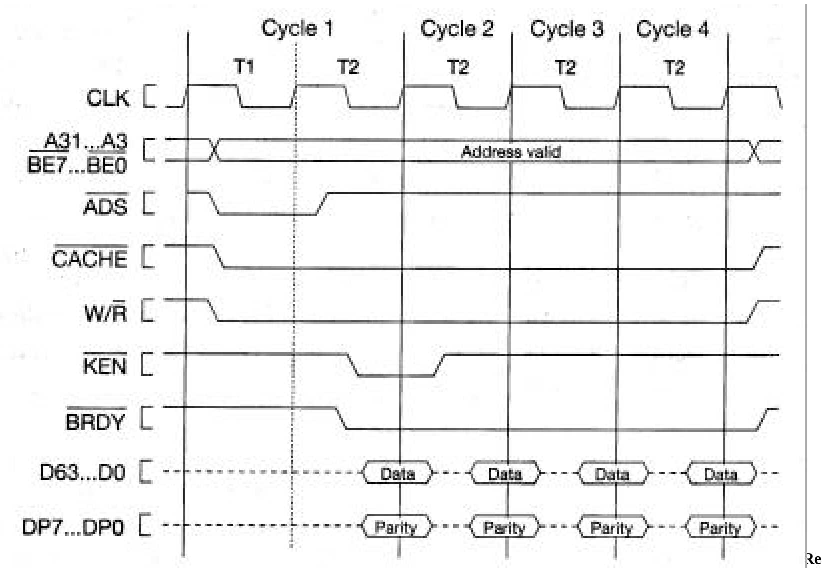
Burst mode

It is similar to that of the 486, but

- The BLAST signal and BRDY signals are substituted by the CACHE and KEN signals
- It supports the transfer of 4 quad-words (32 bytes) in 4 clock cycles
- It can be used for both read and write operations.

If the Pentium runs at 66 MHz, this corresponds to a maximum bandwidth of 403 Mb/s.

Burst read cycle



Address pipelining

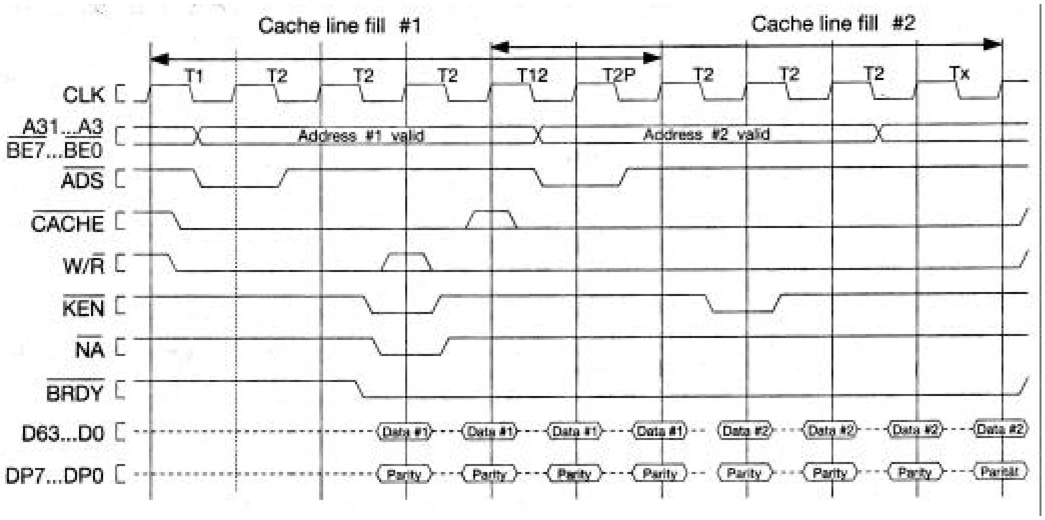
The Pentium also supports address pipelining, and the NA (Next Address) signal is used to manage this mode.

Address pipelining can be combined with burst mode: the first burst-cycle transfer cycle is overlapped by the last transfer cycle of the previous cycle.

In this way, the first bus cycle of the burst cycle reduces to a single clock cycle.

A maximum bandwidth of 504 Mb/s can be achieved in this way.

Address pipelining


Cache line fill #1
Cache line fill #2
T1
T2
T2
T2
T12
T2P
T2
T2
T2
Tx
CLK
A31...A3
BE7...BE0
ADS
CACHE
W/R
KEN
NA
BRDY
D63...D0
DP7...DP0
Address #1 valid
Address #2 valid
Data #1
Data #1
Data #1
Data #1
Data #2
Data #2
Data #2
Data #2
Parity
Parity
Parity
Parity
Parity
Parity
Parity
Parität