Bus systems

M. Sonza Reorda

Politecnico di Torino
Dipartimento di Automatica e Informatica

Introduction

The bus system is the key element in a system, since it strongly influences

- **the system architecture**
- **the communications among system components**
- **the communications with external components.**

The PC/XT architecture

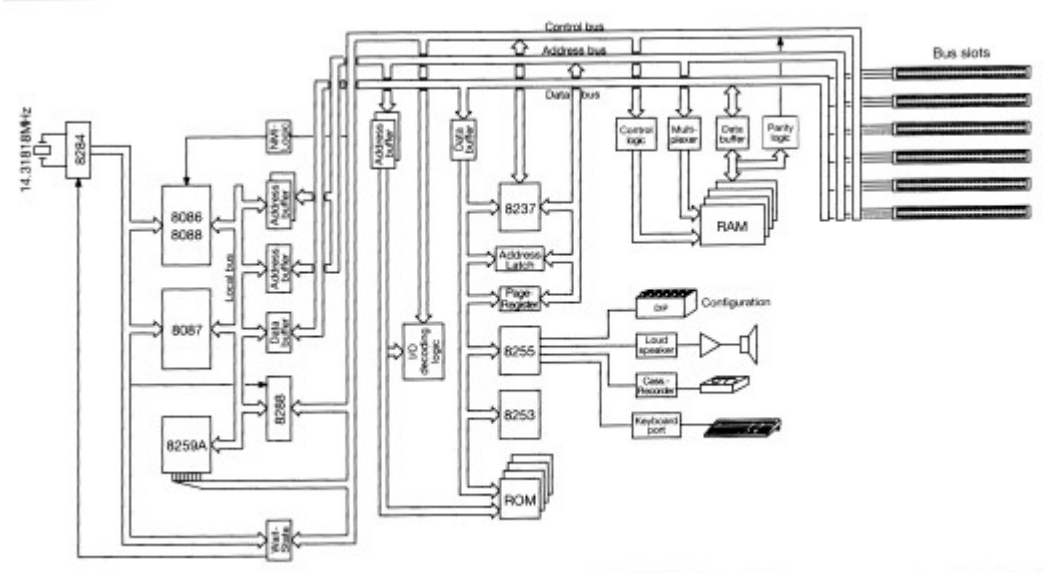
The first IBM PC was the PC/XT, which was equipped with the 8088; its data bus was an 8-bit wide one.

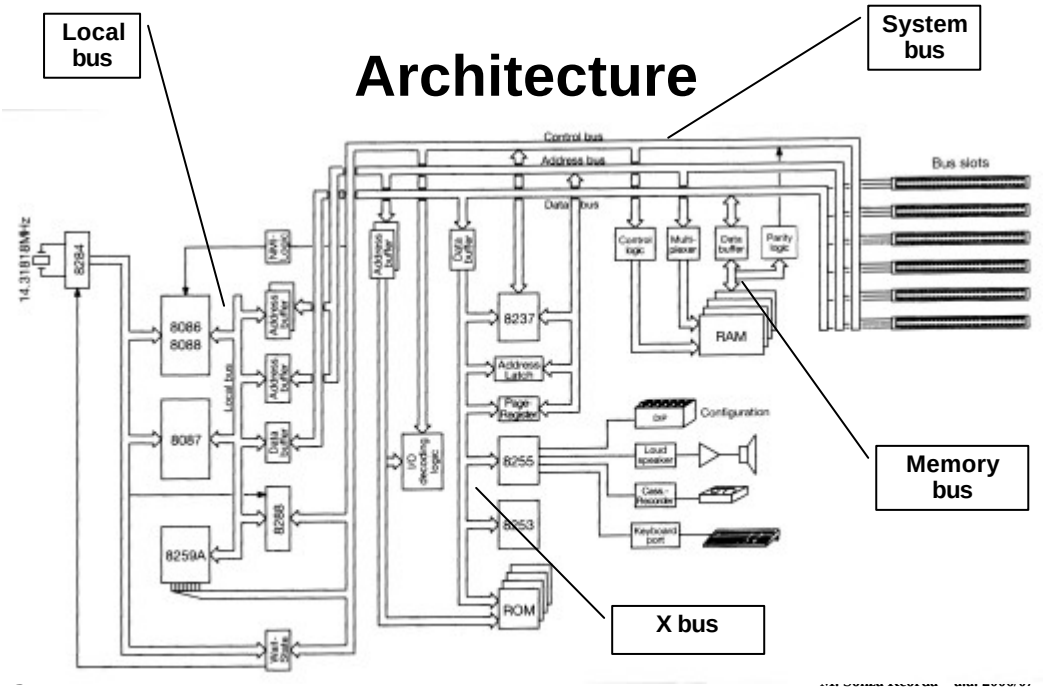
In the 8086 version, the data bus was 16-bit wide.

The system clock maximum frequency was 10 MHz.

Several components are still used in today PCs.

Architecture



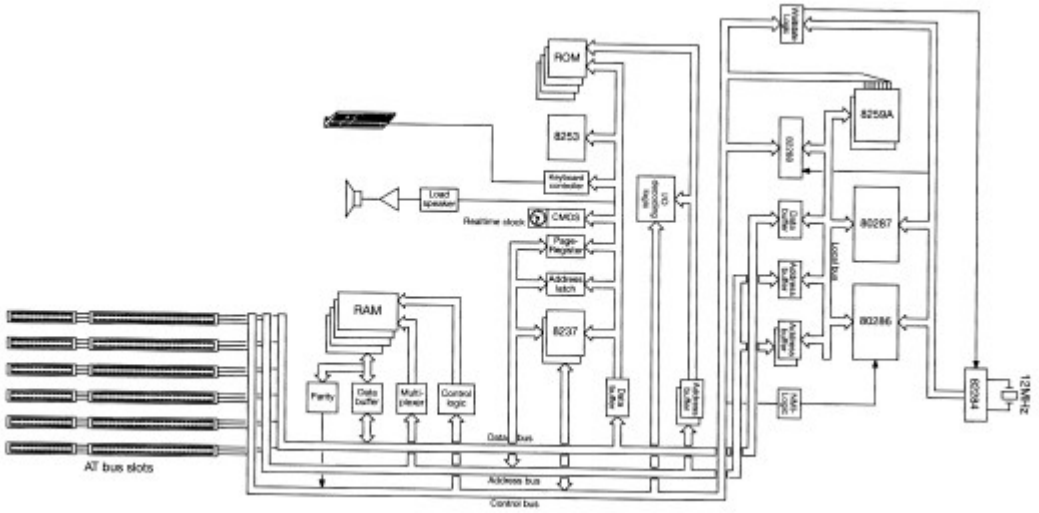


AT architecture

It was introduced with the first 80286 CPUs.

The frequency ranged from 6 to 8 MHz, even for motherboards equipped with 25 MHz CPUs.

Architecture



Bus slots

In the PC/XT they are 8-bit wide and include 62 contacts.

In the PC/AT they are 16-bit wide and include 100 contacts; old cards can still be inserted into the new slots, and the bus logic automatically recognizes the type of the card.

ISA

ISA (*Industrial Standard Architecture*) was the result of a standardization effort aimed at defining clear compatibility rules for AT systems.

ISA is 99% compatible with the original IBM AT bus.

The maximum operating frequency is 8.33 MHz.

After ISA

The introduction of the 386 and 486 32-bit microprocessors required an extension of the ISA bus.

Two solutions were proposed

- ***Microchannel* by IBM**
- ***EISA* (Extended ISA) by a group of manufacturers of IBM-compatible PCs.**

Microchannel was a completely new architecture, which was incompatible with previous cards.

EISA was the evolution of ISA, and was more complex, due to the constraint of supporting existing HW.

EISA

It is a synchronous bus.

Its maximum frequency is 8.33 MHz for accesses to all external units.

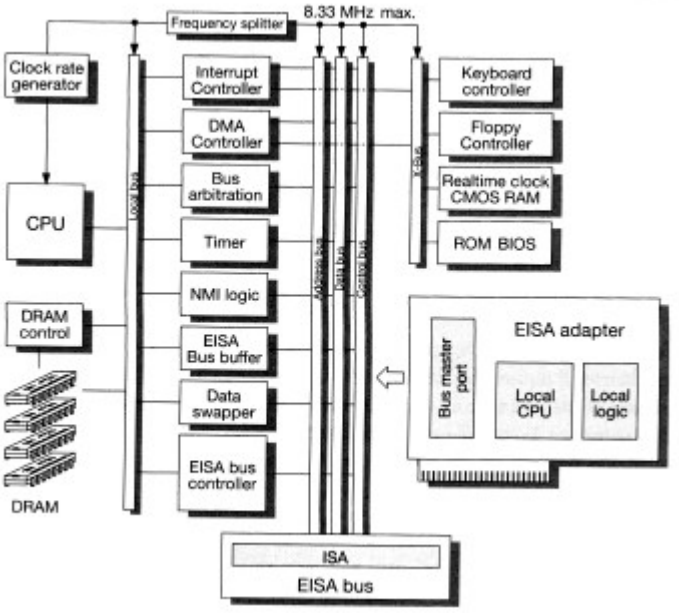
Its maximum bandwidth is 33 Mbyte/s.

The CPU can access the main memory at the full clock frequency through the local bus.

The EISA bus buffer allows data transfers between the local bus and the EISA bus.

CPUs hosted by EISA cards can control the bus (i.e., they can act as bus-masters). This is crucial to implement multiprocessor systems.

Architecture



Bus arbitration

In contrast with ISA, the EISA bus allows an external microprocessor or an EISA adapter card to control the bus.

EISA uses an arbitration model with three levels:

- **DMA/refresh (highest level)**
- **CPU/Master**
- **Other masters (lowest level).**

Within each level the bus is switched in a rotating order.

PCI

The main limitation of EISA (as well as of Microchannel) is the operation frequency (8 MHz with a 50 MHz 486), which becomes critical when certain devices (e.g., video controllers, hard disks) must be managed.

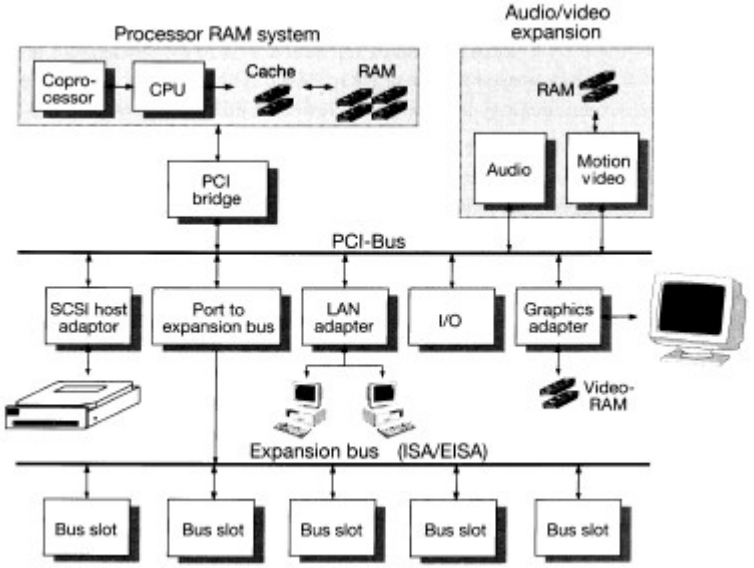
For this reason, in 1993 Intel introduced the PCI (*Peripheral Component Interconnect*) bus.

At the same time, a group of industries working in the VESA consortium independently proposed the *VL bus*.

Main characteristics

- **Decoupling of the processor and expansion bus by means of a bridge**
- **32-bit width, with possible expansion to 64 bits**
- **Maximum 133 Mbyte/s transfer rate**
- **Multimaster capability**
- **Operating frequency up to 33 MHz**
- **Temporal multiplexing of address and data bus**
- **Processor-independent specification.**

Architecture



PCI units

They can be either integrated on the motherboard, or implemented as adapters.

The expansion bus can be a ISA, EISA or Microchannel bus, and can be seen by the PCI bus as a further adapter.

Up to 10 PCI units can be supported by a single bus.

Bus multiplexing

To save on the number of lines, the PCI bus multiplexes data and address lines.

This means that a single transfer requires two to three clock cycles:

- **Address transfer**
- **Write data**
- **Read data.**

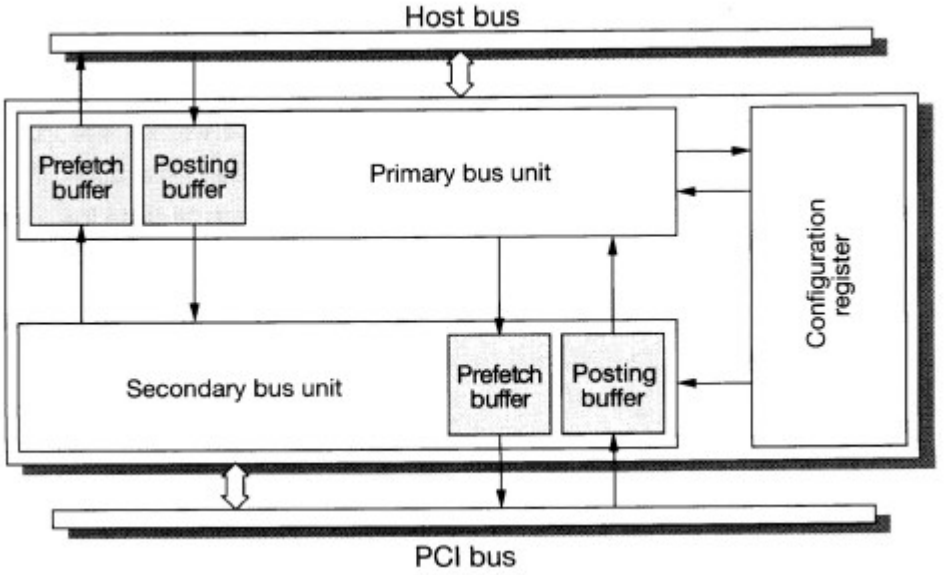
When operating at 33 MHz with 32-bit width, this means that the maximum transfer rate is 66 Mbyte/s for write operations, and 44 Mbyte/s for read operations.

Burst mode

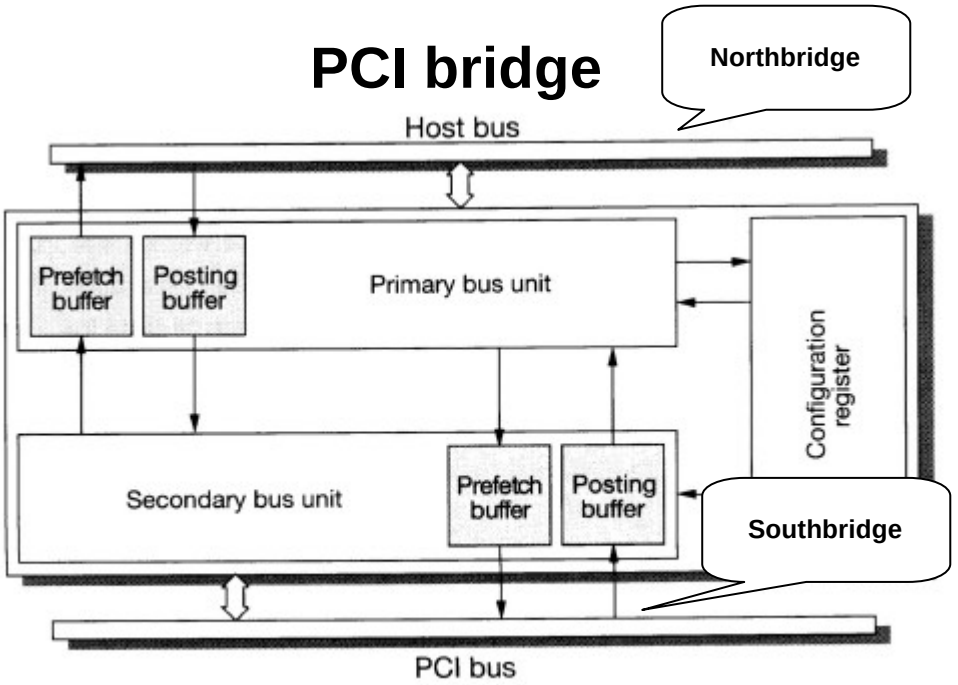
It requires only sending one address during the first cycle.

The maximum data transfer rate in burst mode increases to 133 Mbyte/s with a 32-bit data bus.

PCI bridge



PCI bridge



Read and write buffers

The PCI bridge is able to independently form burst accesses.

This means that the PCI bridge independently joins together incoming single transfer read and write operations to form burst accesses if their addresses are sequential.

For this reason, read and write buffers are included in the bridge.

Configuration space

The PCI bus has three address spaces:

- **Memory**
- **I/O**
- **Configuration.**

The last space includes the configuration registers of each unit (256 bytes).

Bus decoupling

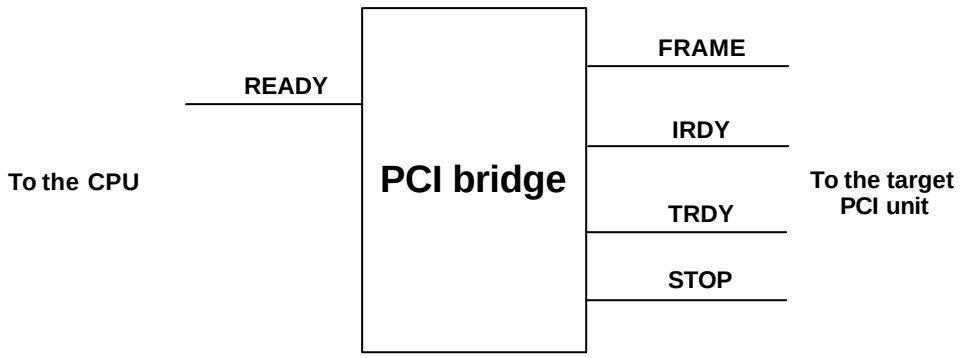
The PCI bridge completely decouples the primary from the secondary busses.

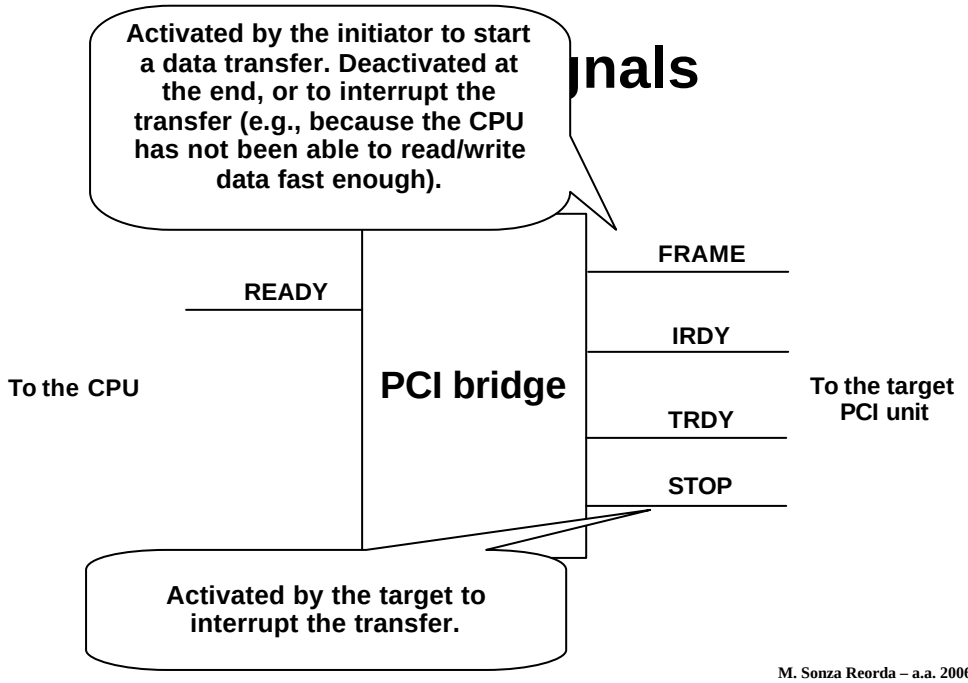
Therefore, two PCI units can communicate one with each other while the CPU is accessing its own RAM.

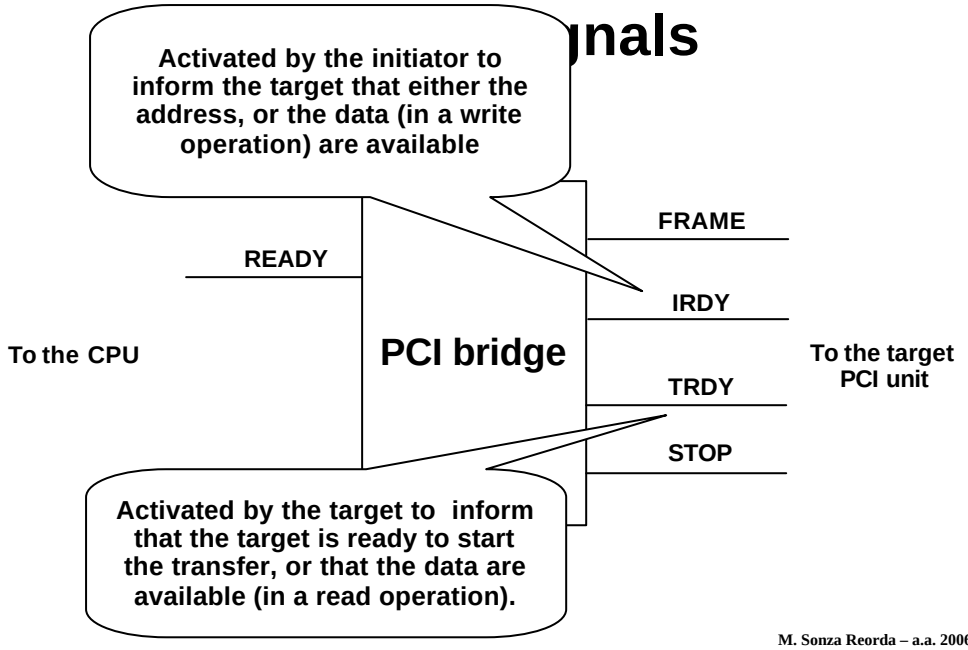
Initiator and target

In the PCI terminology, the PCI unit requesting a cycle is denoted as *initiator* (i.e., master), while the addressed PCI unit is called *target* (i.e., slave).

Control signals




Activated by the initiator to start a data transfer. Deactivated at the end, or to interrupt the transfer (e.g., because the CPU has not been able to read/write data fast enough).
FRAME
READY
IRDY
To the CPU
PCI bridge
To the target PCI unit
TRDY
STOP
Activated by the target to interrupt the transfer.


Activated by the initiator to inform the target that either the address, or the data (in a write operation) are available
FRAME
READY
IRDY
To the CPU
PCI bridge
To the target PCI unit
TRDY
STOP
Activated by the target to inform that the target is ready to start the transfer, or that the data are available (in a read operation).

Bus cycles

Each bus cycle type is defined by the value of the command signals C/BE3 to C/BE0).

The possible bus cycle types are:

- INTA sequence
- Special cycle
- I/O read and write
- Memory read and write
- Configuration read and write
- Memory multiple read
- Dual addressing
- Line memory read
- Memory write with invalidation.

Bus cy

Used to inform the PCI unit about the operating status of the processor (e.g., performing a shutdown, or a flush operation).

Each bus cycle type is def
command signals C/BE3 to

The possible bus cycle

- INTA sequenc
- Special cycle
- I/O read and write
- Memory read and write
- Configuration read and write
- Memory multiple read
- Dual addressing
- Line memory read
- Memory write with invalidation.

Bus cycles

Each bus cycle type is defined by the value of the command signals C/BE3 to C/B

The possible bus cycle types a

- **INTA sequence**
- **Special cycle**
- **I/O read and write**
- **Memory read and w**
- **Configuration read and write**
- **Memory multiple read**
- **Dual addressing**
- **Line memory read**
- **Memory write with invalidation.**

Used to inform the target unit that the initiator wants to perform a burst operation longer than the usual cache line (in that case the line memory read mode is used).

Bus cycles

Each bus cycle type is defined by the value of the command signals C/BE3 to C/BE0).

The possible bus cycle types are:

- **INTA sequence**
- **Special cycle**
- **I/O read and write**
- **Memory read and write**
- **Configuration read and write**
- **Memory multiple read**
- **Dual addressing**
- **Line memory read**
- **Memory write with invalidation.**

Used to support 64-bit addresses within systems with 32-bit address bus. The 64-bit address is split over two 32-bit address transfer operations.

Bus cycles

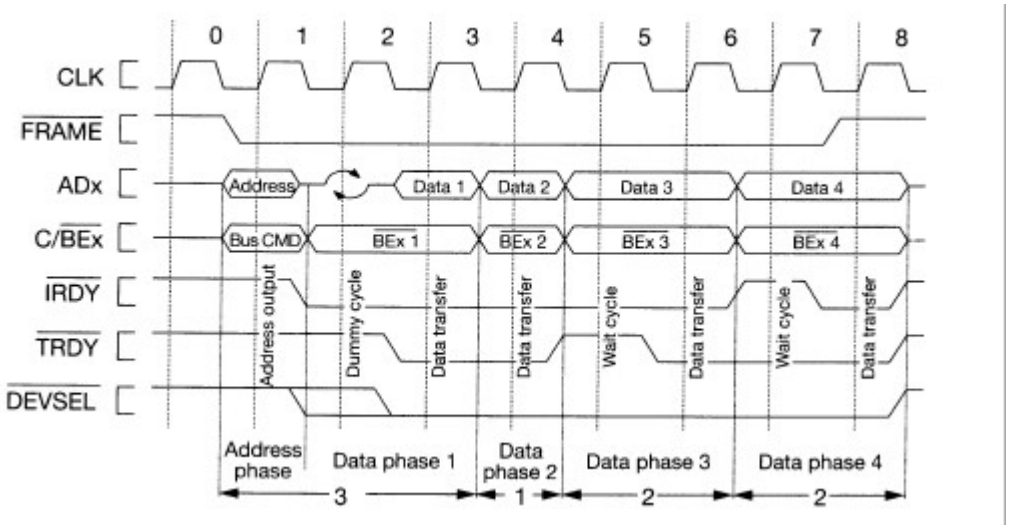
Each bus cycle type is defined by the value of the command signals C/BE3 to C/BE0).

The possible bus cycle types are:

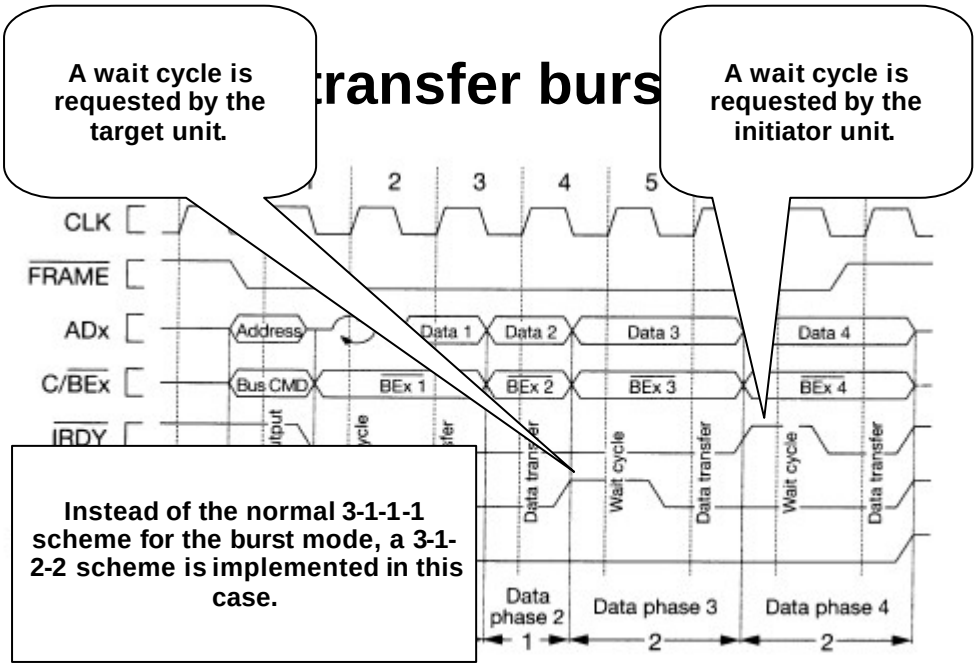
- **INTA sequence**
- **Special cycle**
- **I/O read and write**
- **Memory read and write**
- **Configuration read and writ**
- **Memory multiple read**
- **Dual addressing**
- **Line memory read**
- **Memory write with invalidation.**

Used to request a write operation to memory, in which the cache is also notified that the corresponding line should be invalidated.

Read transfer burst cycle



transfer burs



Bus arbitration

The target for each cycle is decided in parallel with normal bus cycles (*hidden arbitration*).

Arbitration happens at each bus cycle.

Each PCI unit has a REQ and GNT signal which allow it to interact with an arbitration logic.

The PCI standard does not define which strategy the arbitration logic should implement.