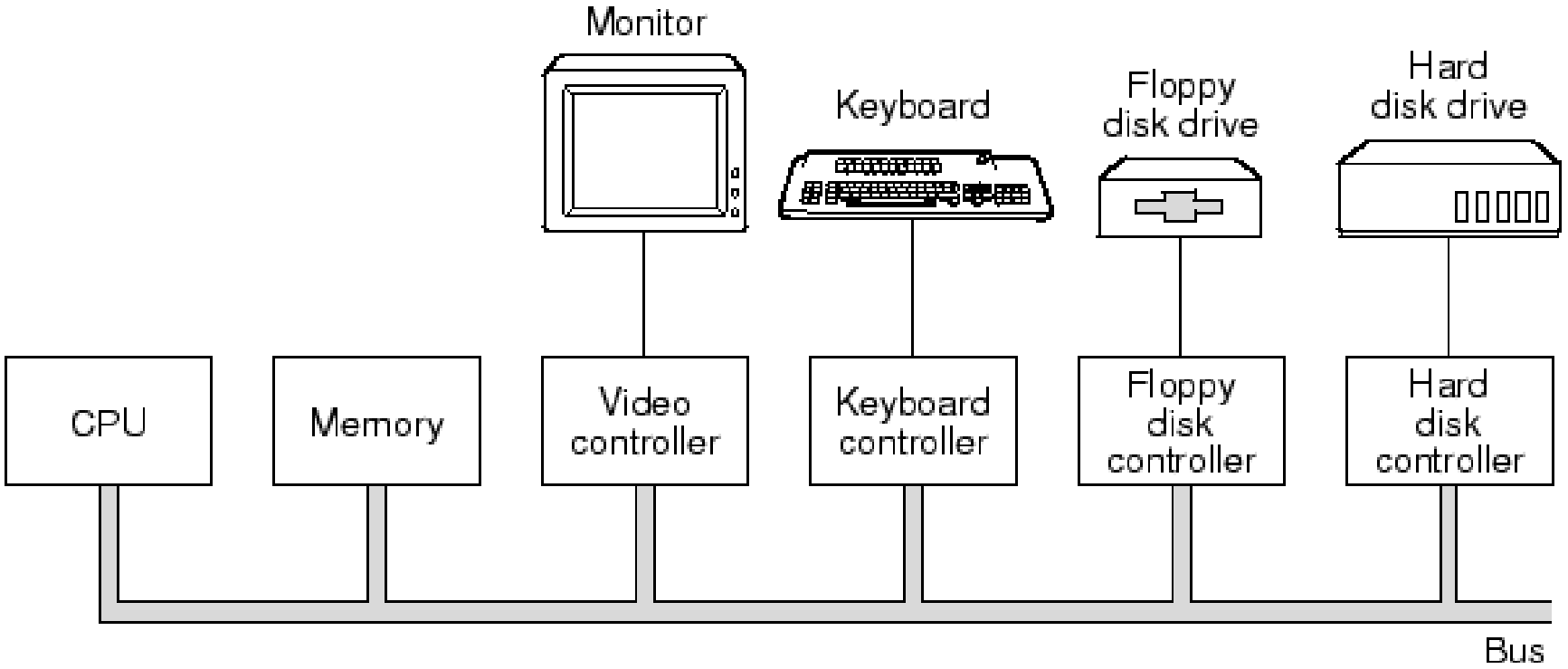
Buses

(Never there when you want them, tnen three come along together0

- The bus is the main data highway inside the computer system



Bus Terminology

- **Width**

  There are generally three types of lines on a bus:

  1. **Address lines**

     Values on these lines specify addresses: main memory adresses, I/O device address

  2. **Data lines**

     These lines carry data between CPU, memory and I/O.

  3. **Control lines**

     All other lines; these have various functions mainly concerned with synchronising operations on the bus

  The number of data lines on the bus determines the bus's **width**. The greater the width, the more data can be transferred at once.

(Politically incorrect0

Bus Terminology

- **Master** vs. **Slave**
  - Some devices on the bus must be capable of taking control of the bus, these devices are referred to as **bus masters**. The CPU is the obvious example, although many I/O controllers are bus masters too.
  - This is because I/O typically involves transferring data between the I/O device (e.g. hard disk) and memory. **DMA** (**Direct Memory Access**) I/O is the name given to such transfers, which can continue with the intervention of the CPU. Typically the CPU will request the I/O to begin and the I/O controller will signal (via an interrupt) when the I/O has completed.
  - When a bus master takes control of the bus it will always want to communicate with another device on the bus, that device is known as the **slave**. Memory is always the slave, I/O controllers can be both master and slave (although not simultaneously)

Bus Terminology

- Bus arbitration

  When two masters attempt to take control of the bus at the same time there must be a method for resolving the conflict, this is called **bus arbitration**.

- Cycle Stealing

  Although the CPU can always wait to access memory, I/O devices, although much slower, cannot (it is hard to stop a disk spinning).

  Whenever there is a conflict between CPU and I/O controller, the controller is given priority, this is called cycle stealing.

Bus Standards

The original IBM PC had an 8-bit bus. When the PC-AT (based on the 80286 CPU) was introduced, this was extended to a 16-bit bus, clocked at 8MHz.

This was adopted by the clone manufacturers (mainly because, in order to encourage high availability of devices and adapters, IBM licensed the bus architecture very cheaply) and become known as the **ISA** (**Industry Standard Architecture**) bus.

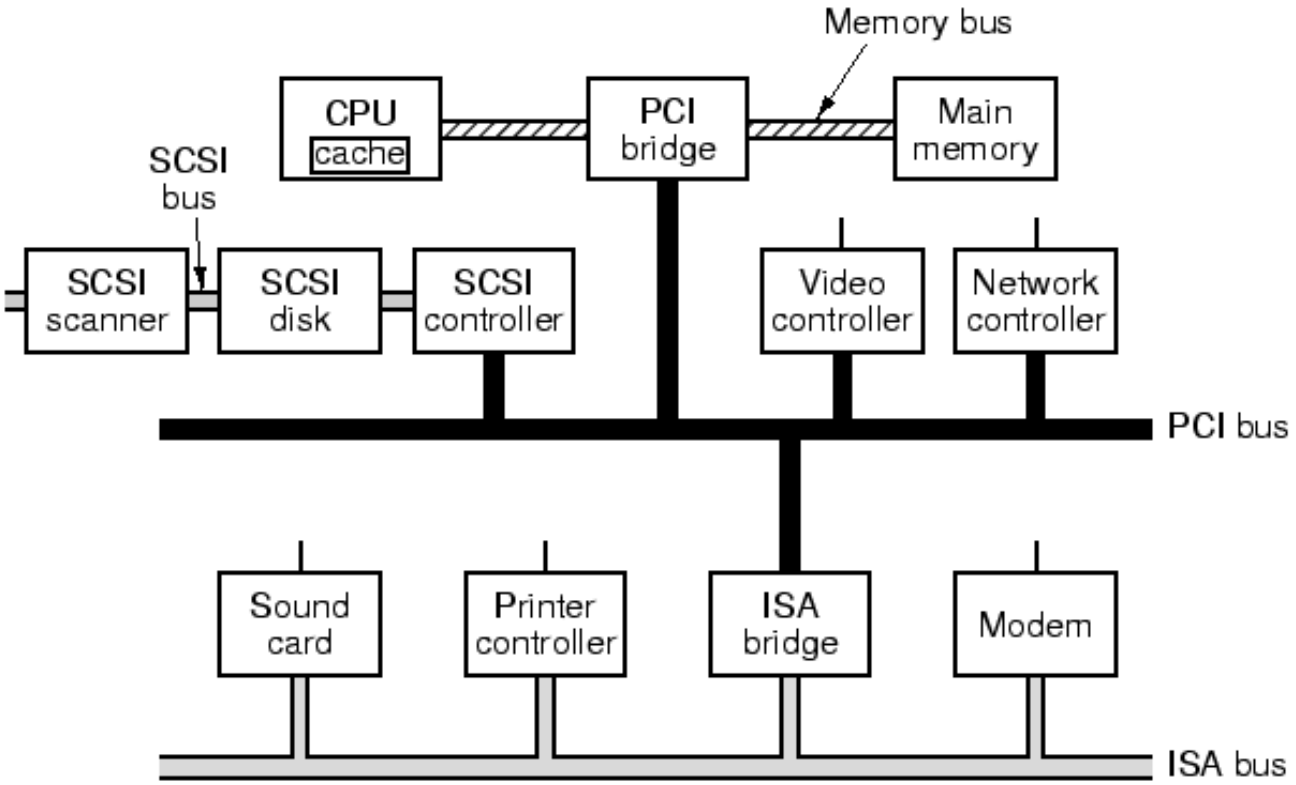
IBM attempted to introduced a new, proprietary bus with the PS/2 range, called the **MCA** (**MicroChannel Architecture**) bus, but this was not a success, nor was the clone manufacturers' attempt to upgrade the ISA bus to 32-bits with the **EISA** (**Extended ISA**) bus.

The current bus standard in the Intel world is the **PCI** (**Peripheral Component Interconnect**) bus, designed by Intel for use with the Pentium CPU and placed into the Public Domain so that all manufacturers would adopt it.

The PCI is a 32-bit (optionally, 64-bit) bus originally clocked at 33MHz; the current specification (PCI2.2) allows for 66MHz, although these are still rare in practice

Multiple Bus Architectures

Modern computers have many buses in many places: inside the CPU, connecting the CPU to memory, connecting the CPU to I/O controllers etc.



Bus Design

- Type
  - Dedicated
    - Each bus line is permanently assigned a single function.
  - Multiplexed
    - Lines may be used for multiple purposes at different times.

      For example, data and addresses can be sent over the same lines provided that we add a control line called **Address Valid**.

Bus Design

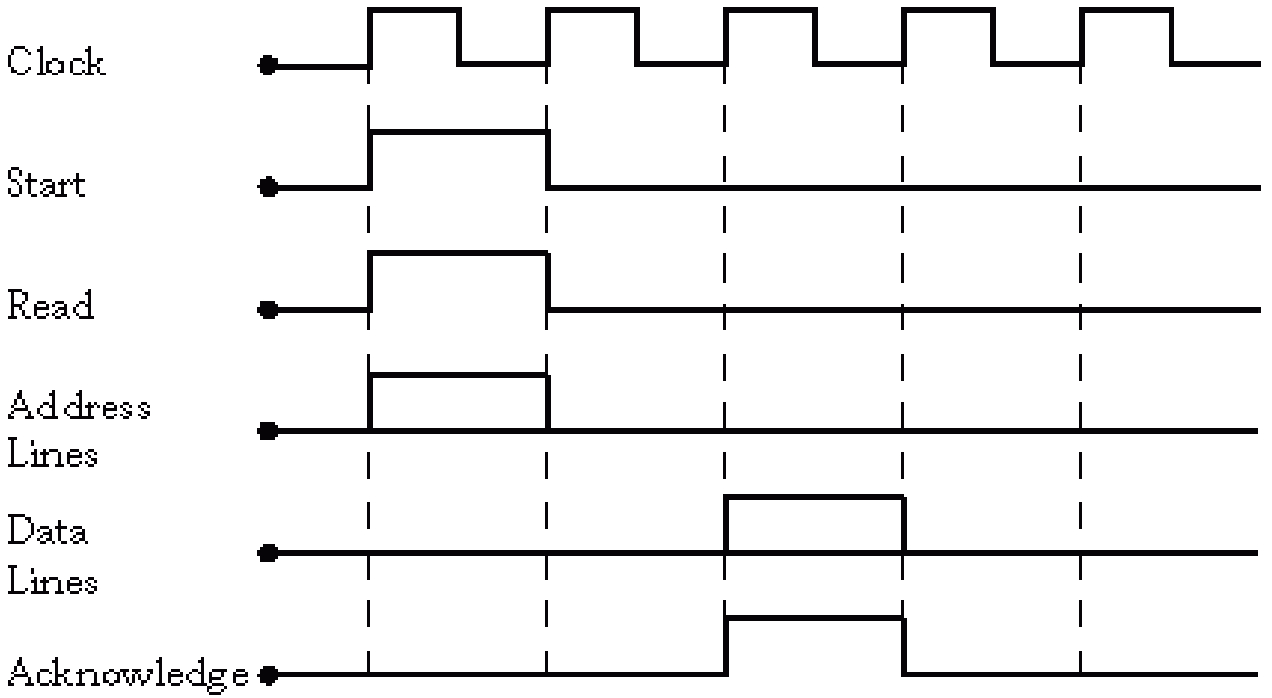
- Arbitration Method
  - Centralized
    - A single hardware device, known as the bus controller or arbiter is responsible for allotting time on the bus. This may be a separate device or integrated into the CPU.
  - Distributed
    - In a distributed arbitration scheme each module contains access control logic and the modules cooperate.

  With both types of arbitration the end result is the same: one device is designated the bus master; it is then able to initiate a data transfer to or from another device, which is the slave for this transaction.

  Note that at different times the same module may be master or slave.

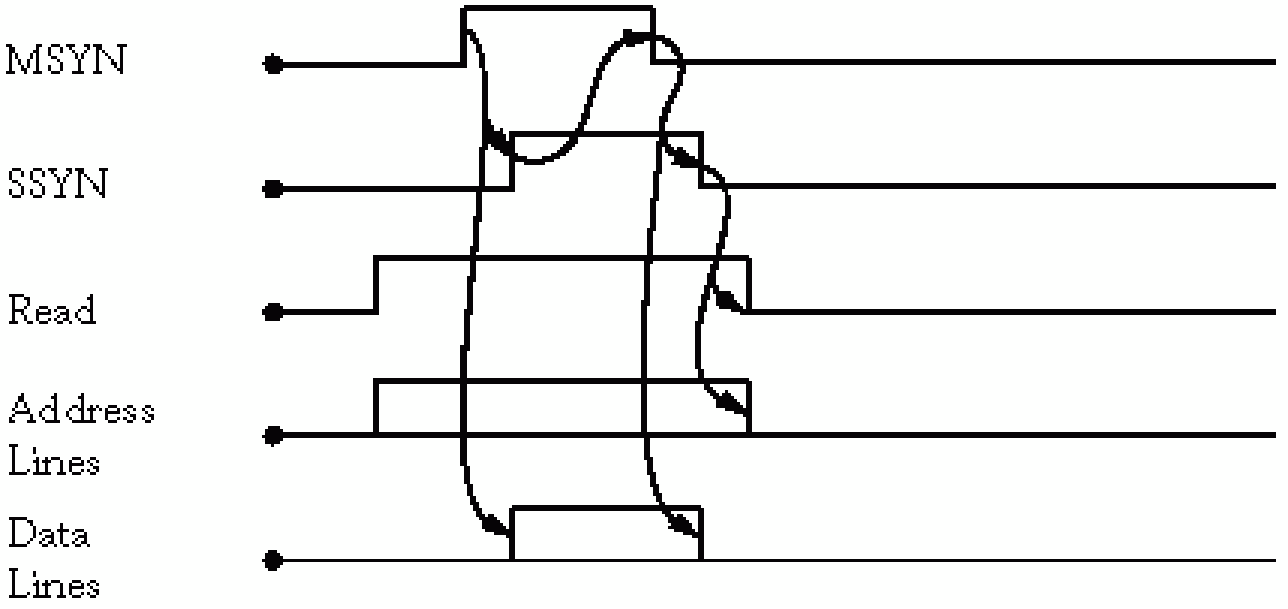
Bus Timing

- How are events on the bus coordinated?
  - Synchronous
    - The occurrence of events is determined by a clock which provides a regular pulse of alternating 0s and 1s of equal length.
    - Each 0-1 transmission is called a **clock cycle**.
    - Devices on the bus all see the clock line and all events start at the beginning of a clock cycle.



Bus Timing

- Asynchronous
  - There is no central clock. All devices may proceed at their own optimal rate. Events on the bus are triggered by the occurrence of other events.

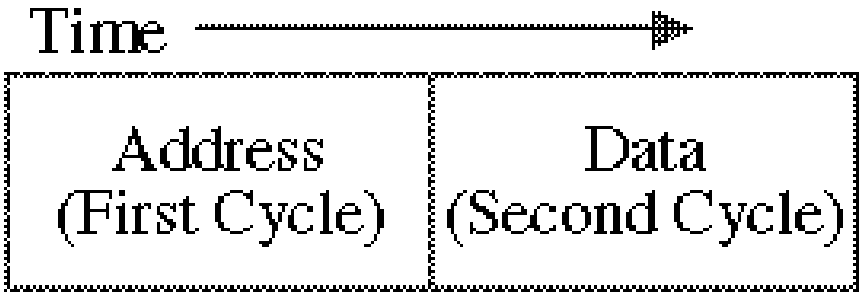


Bus Width

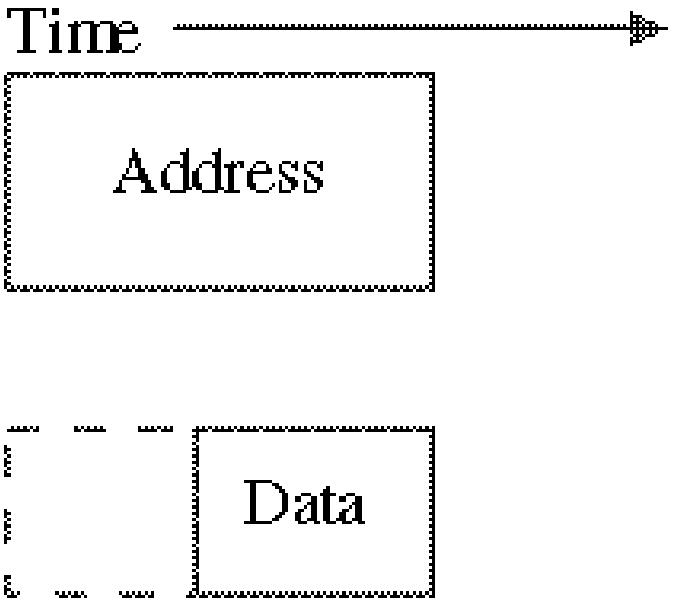
- Data
  - The wider the data bus the more bits can be transferred in parallel, hence the higher the data transfer rate.
- Address
  - The wider the address bus the more cells (words) of memory can be addressed.

Data Transfer Type

- Write



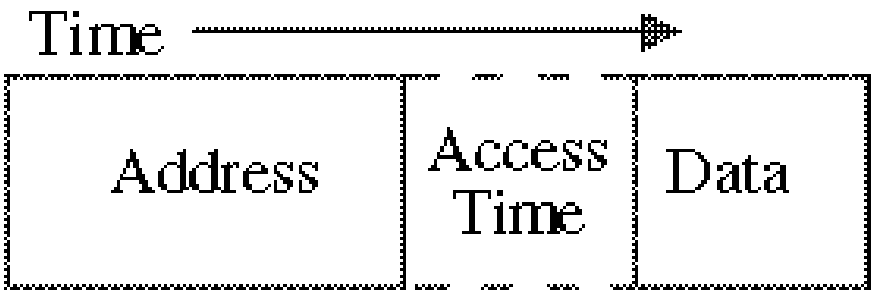
Multiplexed Write Operation



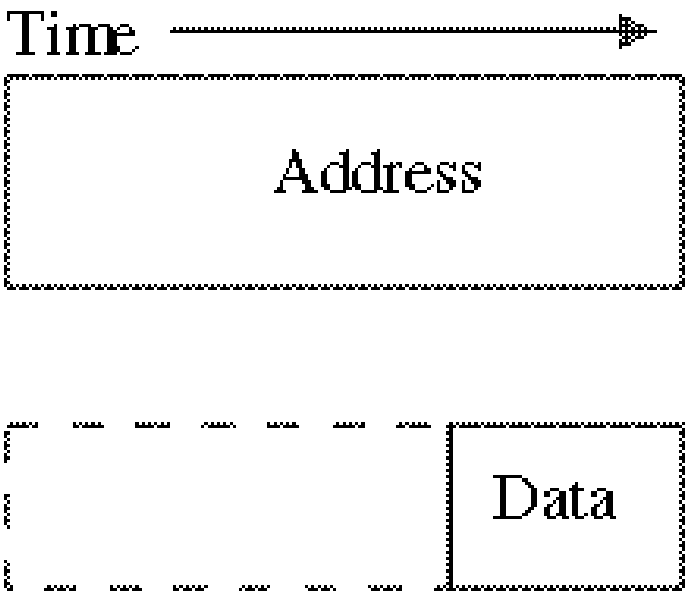
Non-multiplexed Write

Data Transfer Type

- Read



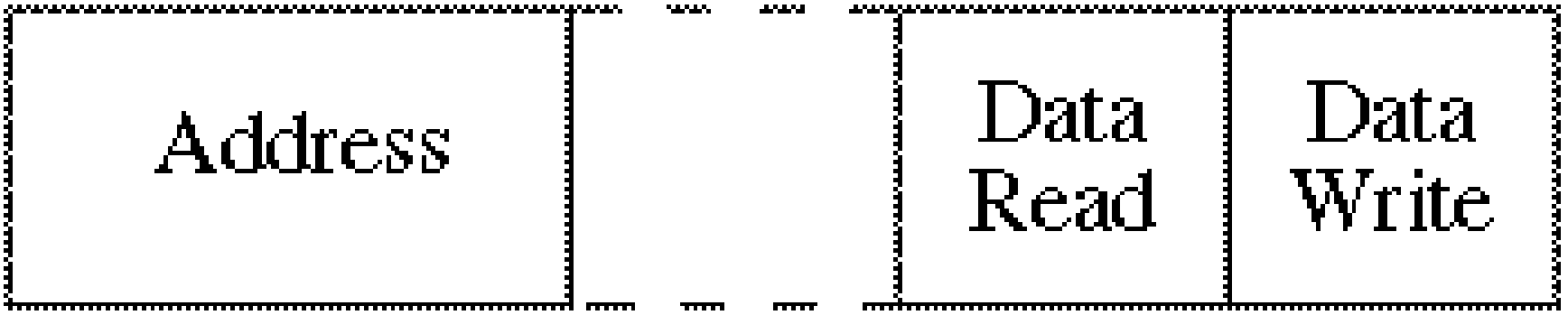
Multiplexed Read Operation



Non-multiplexed Read

Data Transfer Type

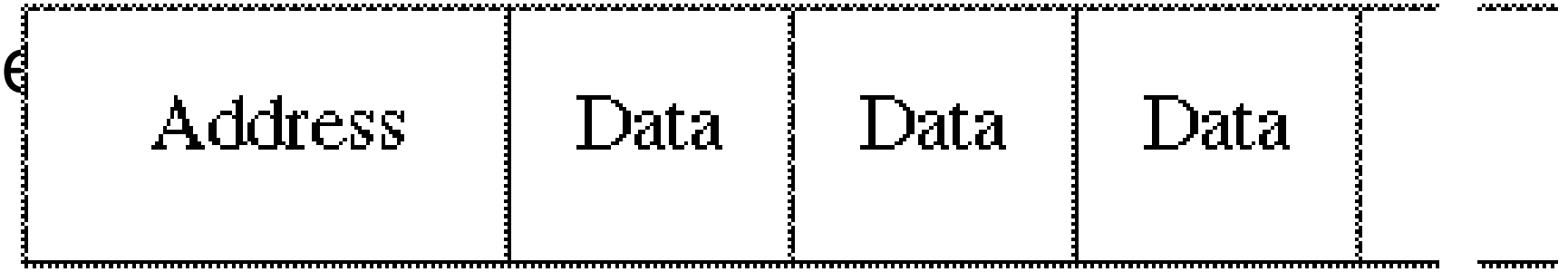
- Read-Modify-Write
  - This is an indivisible combination operation: a read followed immediately by a write
  - is often used in multi-tasking systems to support synchronisation primitives.



Read-Modify-Write Operation

Block Data Transfer

- One address cycle is followed by n data cycles; transfers after the first are to/from subsequent me



Block Data Transfer

The PCI Bus

Intel began designing the PCI specification in 1990 for its P5 (subsequently Pentium) line. They later released the spec into the public domain

PCI2.2, latest standard, although three new standards (PCI2.3, PCI-X and PCI-Express) are all on the horizon

Features:

- High-Bandwidth, CPU-independent, Mezzanine or Peripheral Bus
- 32 (optionally 64) Data Lines

The PCI Bus Structure

PCI may be configured as a 32- or 64-bit bus.

There are four groups of **pins** (or lines)

Group	Function
System Pins	Includes clock and reset
Address and Data Pins	32 lines are multiplexed between address and data. Other lines interpret and validate address/data lines.
Interface Control Pins	Control timing of transaction and provide coordination.
Arbitration Pins	Not shared lines; each PCI master has its own pair of arbitration lines.
Error Reporting Pins	Used to report parity and other errors.

The PCI Bus Structure

The PCI spec defines 50 mandatory signal lines which must be implemented.

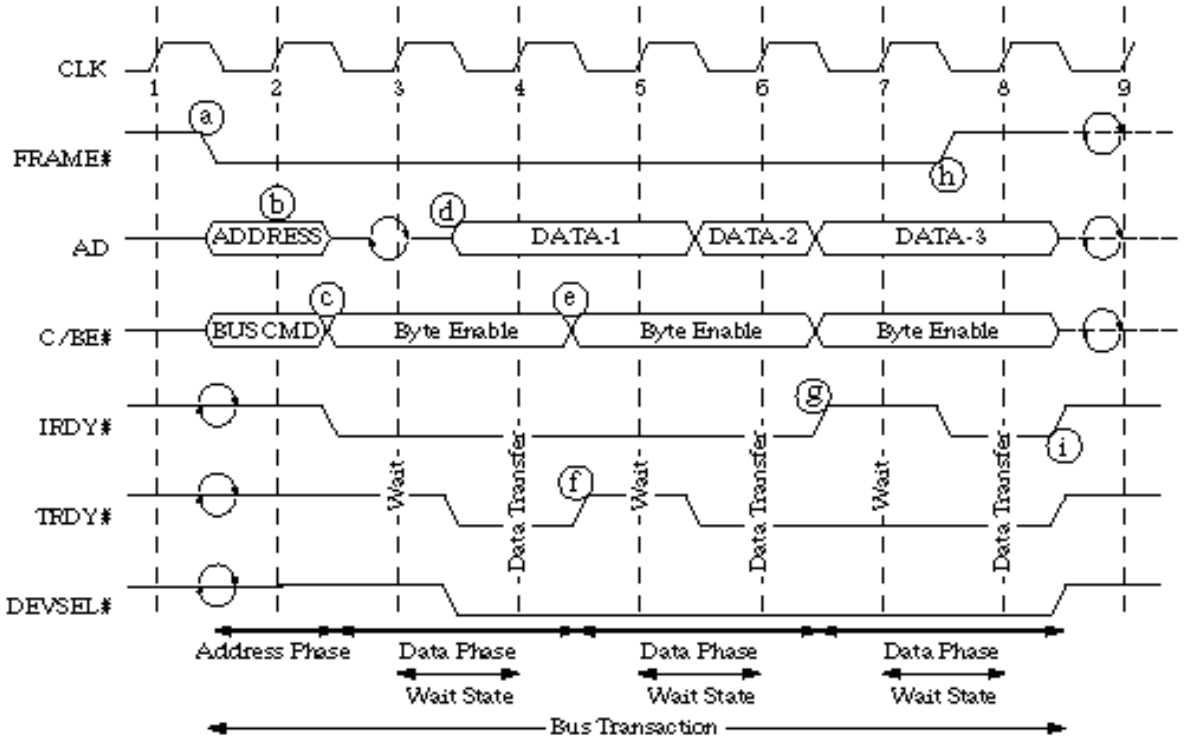
Designation	Type	Description
CLK	in	Provides timing for all transactions and is sampled by all inputs on the rising edge. Clock rates up to 33 MHz (66Mhz in PCI2) are supported.
FRAME#	s/t/s	Driven by current master to indicate the start and duration of a transaction. It is asserted at the start and deasserted when the initiator is ready to begin the final data phase.
AD[31::0]	t/s	Multiplexed lines used for address and data
C/BE[3::0]#	t/s	Multiplexed bus command and byte enable signals. During the data phase, the lines indicate which of the four byte lanes car meaningful data.
IRDY#	s/t/s	Initiator Ready. Driven by current bus master (initiator of transaction). During a read, indicates that the master is prepared to accept data; during a write, indicates that valid data is present on AD.
TRDY#	s/t/s	Target Ready. Driven by the target (selected device). During a read, that valid data is present on AD; during a write, indicates that target is ready to accept data.
DEVSEL#	in	Device Select. Asserted by target when it has recognized its address. Indicates to current initiator whether any device has been selected.
REQ#	t/s	Indicates to the arbiter that this device requires use of the bus. This is a device-specific point-to-point line.
GNT#	t/s	Indicates to the device that the arbiter has granted bus access. This is a device-specific point-to-point line.

PCI Commands

- Bus activity consists of transactions between an initiator (or master) and a target (or slave).
- The master gains control of the bus and then determines the type of transaction. During the address phase of the transaction, the C/BE lines are used to signal the transaction type.
- Transactions include:

Interrupt Acknowledge	Special Cycle
I/O Read	I/O Write
Memory Read	Memory Read Line
Memory Read Multiple	Memory Write
Memory Write and Invalidate	Configuration Read
Configuration Write	Dual Address Cycle

A Sample PCI Transaction



(a) Bus master begins transaction by asserting FRAME; also puts start address on address bus and read command on C/BE lines.

(b) Target device recognises its address on the AD lines

(c) Initiator ceases driving AD bus; initiator changes info on C/BE lines to designate which AD lines are used for data transfer. Also assert IRDY.

(d) Target asserts DEVSEL to indicate it will respond. Places data onto AD lines and asserts TRDY to indicate the presence of valid data.

(e) Initiator reads data and changes B/CE lines for next read.

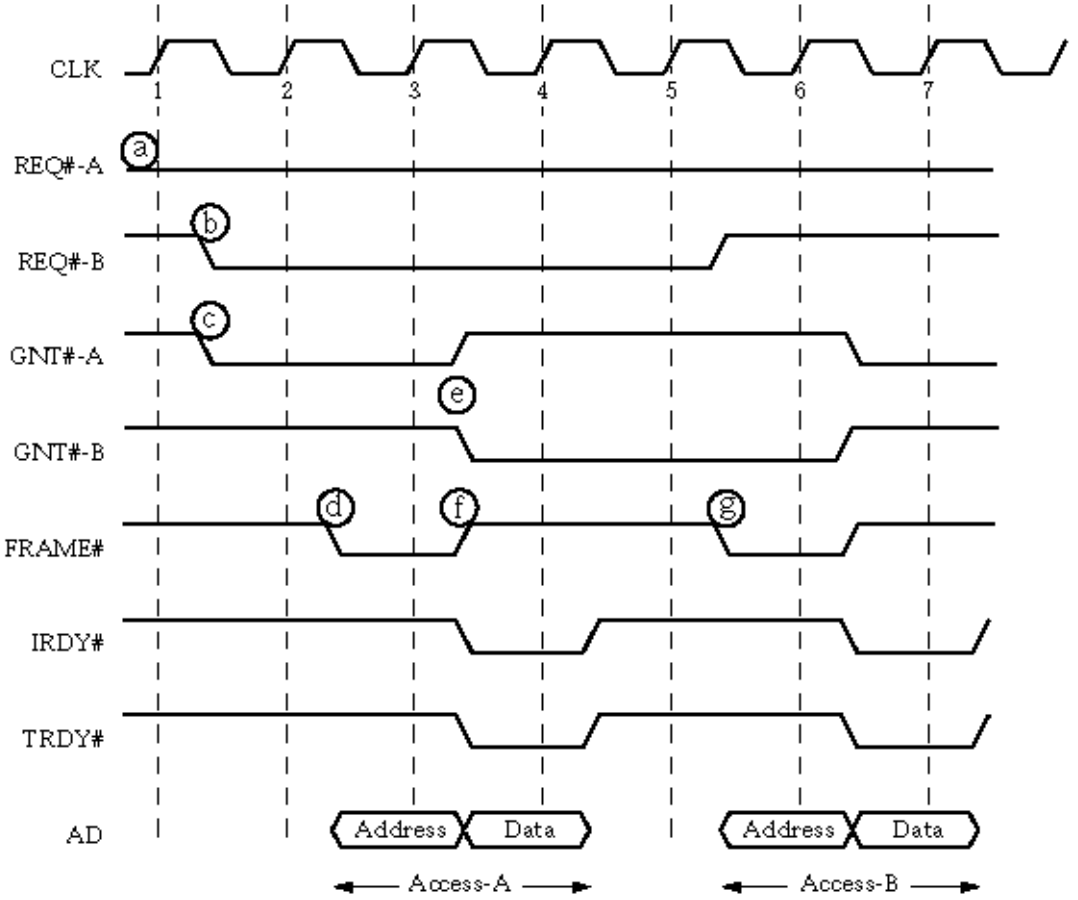
(f) Target needs time for second item, so it deasserts TRDY to indicate that there will be no new data in the next cycle.

(g) Target places third data item onto bus; but initiator is not ready yet, so it deasserts IRDY. target maintains data item for further cycle.

(h) Initiator knows third item is last so it deasserts FRAME to signal target that this is last. Also asserts IRDY to indicate it is ready.

(i) Initiator deasserts IRDY, returning the bus to idle; target deasserts TRDY and DEVSEL.

An Arbitration Example



(a) Arbiter samples all REQ# lines at the beginning of clock cycle 1.

(b) During clock cycle 1 B requests use of the bus.

(c) Arbiter grants bus to A by asserting A's GNT# line.

(d) A samples GNT# and finds it has the bus. IRDY and TRDY deasserted means bus idle. A asserts FRAME, puts address onto bus and command on CB/E, continues to assert REQ# because it has a second transaction.

(e) Arbiter samples all REQ# lines. Grants bus to B for next cycle and deasserts GNT# for device A. B must wait for the bus to be idle.

(f) A deasserts FRAME (last transfer). Puts data onto data bus and signals target with IRDY. Target reads data at next clock cycle.

(g) IRDY and FRAME deasserted so B can take control of bus by asserting FRAME; deasserts REQA# because it only has one transaction.