I/O fundamentals

Computer System Architecture

By

M. Morris Mano

Prof.C.Navaneethan, SITE,VIT.

Peripheral Devices

- **I/O Subsystem (or I/O)**
  - Provides an efficient mode of communication between the central system and the outside environment
- **Peripheral (or I/O Device)**
  - Input or Output devices attached to the computer
    - » Monitor (*Visual Output Device*) : CRT, LCD
    - » KBD (*Input Device*) : light pen, mouse, touch screen, joy stick, digitizer
    - » Printer (Hard Copy Device) : Dot matrix (*impact*), thermal, ink jet, laser (*non-impact*)
    - » Storage Device : Magnetic tape, magnetic disk, optical disk

Input-Output Interface

- Provides a method for transferring information between internal storage and external i/o devices.
- Differences between the computer and peripheral devices
  1. A conversion of signal values may be required
     - Peripherals - Electromechanical Devices
     - CPU or Memory - Electronic Device
  2. Data Transfer Rate
     - Peripherals - slower
     - CPU or Memory - faster than peripherals
     - Some kinds of Synchronization mechanism may be needed
  3. Unit of Information
     - Peripherals - Byte
     - CPU or Memory - Word
  4. Operating Modes
     - Peripherals - Autonomous, Asynchronous
     - CPU or Memory - Synchronous

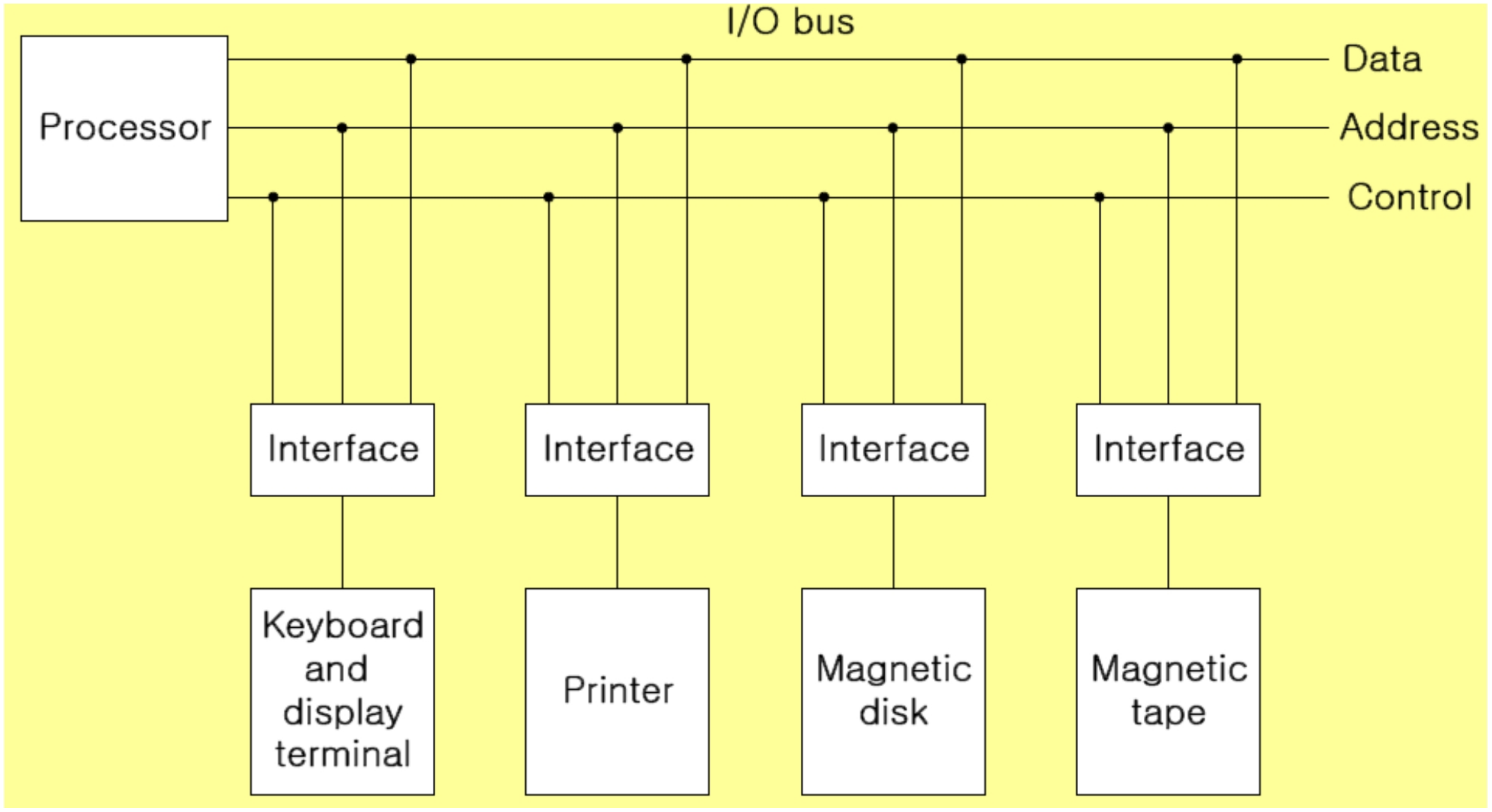
Interface units

- Special hardware components between the CPU and peripherals
- Supervise and Synchronize all input and output transfers
- Resolves the *differences* between the computer and peripheral devices

I/O Bus

- **I/O BUS**
  - Transfers information between CPU and I/O devices through their I/O interface
- I/O Bus consists of
  - » Data lines
  - » Address lines
  - » Control lines

Connection of I/O bus to input-output devices



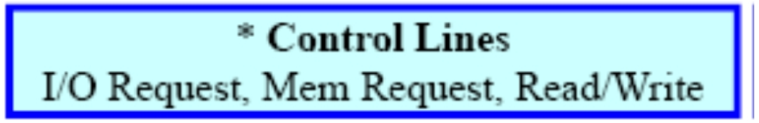
- I/O command : Function code from processor
  - » Control Command
    - Issued to activate the peripheral & to inform it what to do.
  - » Status Command
    - Test various status conditions in the interface and the peripheral.
  - » Input Command
    - Interface receives data from peripheral and places it in its buffer register
  - » Output Command
    - Interface transfer information from the

I/O Bus versus Memory Bus

- **I/O BUS**
  - Transfers information between CPU and I/O devices through their I/O interface
- **MEMORY BUS**
  - Transfer information between CPU and the Main Memory
  - Like I/O Bus, memory bus consists of
    - » Data lines
    - » Address lines
    - » Control lines

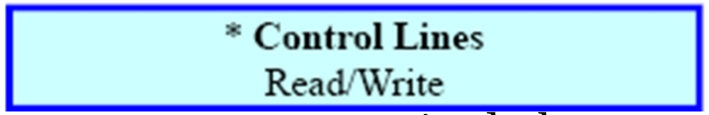
3 ways -computer buses can be used to communicate with memory and I/O

1) Use two separate buses, one for memory and the other for I/O.
2) Use one common bus for both memory and I/O but have separate control lines for each: ***Isolated I/O** or **I/O Mapped I/O***
    - **IN**, **OUT** : I/O Instruction
    - **MOV** or **LD** : Memory read/write Instruction



*** Control Lines**
I/O Request, Mem Request, Read/Write

3) Use one common bus for memory and I/O with common control lines:
   Memory - Mapped I/O
    - **MOV** or **LD** : I/O and Memory read/write Instruction



*** Control Lines**
Read/Write

- Many computers use a common single bus system for both memory and I/O interface units

Isolated vs Memory Mapped I/O

Isolated I/O

- Separate I/O read/write control lines in addition to memory read/write control lines
- Separate (isolated) memory and I/O address spaces

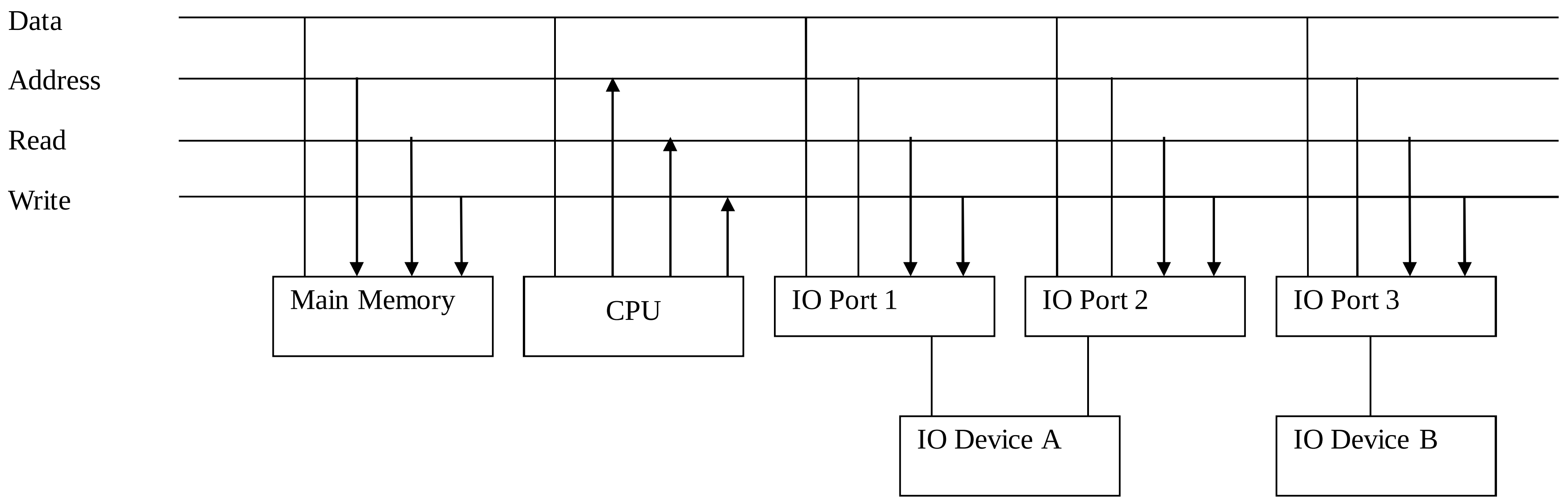
- Distinct input and output instructions
- When CPU fetches and decodes the opcode of an I/O instruction, it places the address into the common address lines. Also enables read/write control lines => the address in the address lines is for interface register and not for a memory word.

Memory mapped I/O

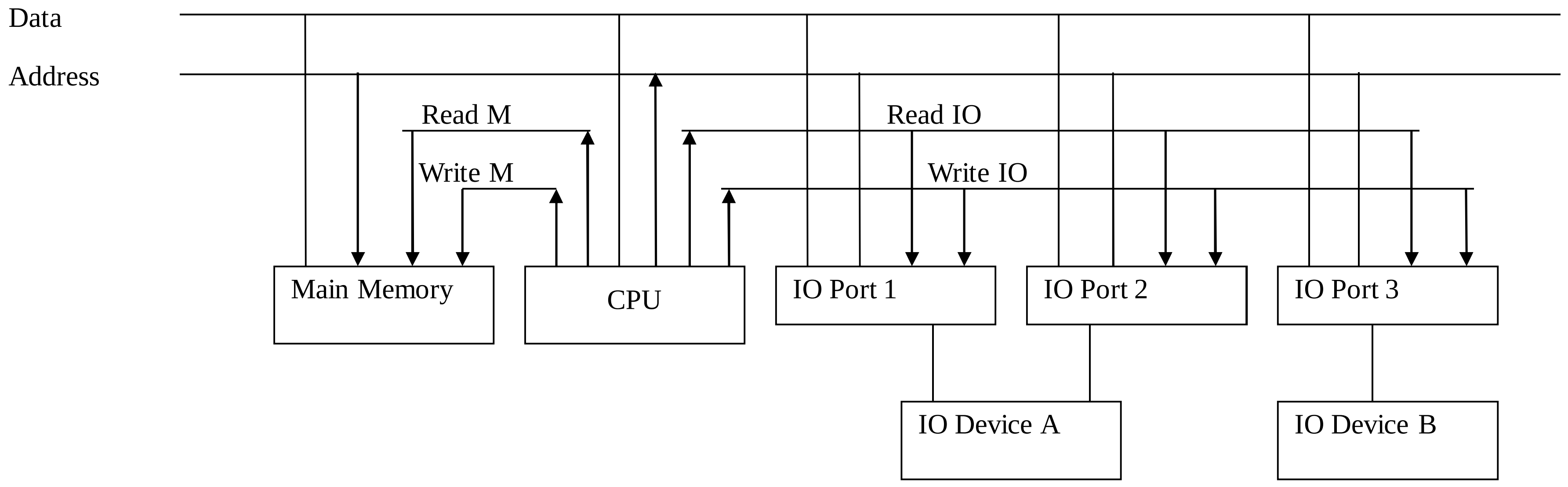
- A single set of read/write control lines (no distinction between memory and I/O transfer)
- Memory and I/O addresses share the common address space -> reduces memory address range available
- No specific input or output instruction -> The same memory reference instructions can be used for I/O transfers
- Considerable flexibility in handling I/O operations

I/O Mapping

- Memory mapped I/O
  - I/O devices and memory share an address space
  - I/O looks just like memory read/write
  - No special commands for I/O
    - Large selection of memory access commands available
- Isolated I/O
  - Separate address spaces
  - Need I/O or memory select lines
  - Special commands for I/O
    - Limited set



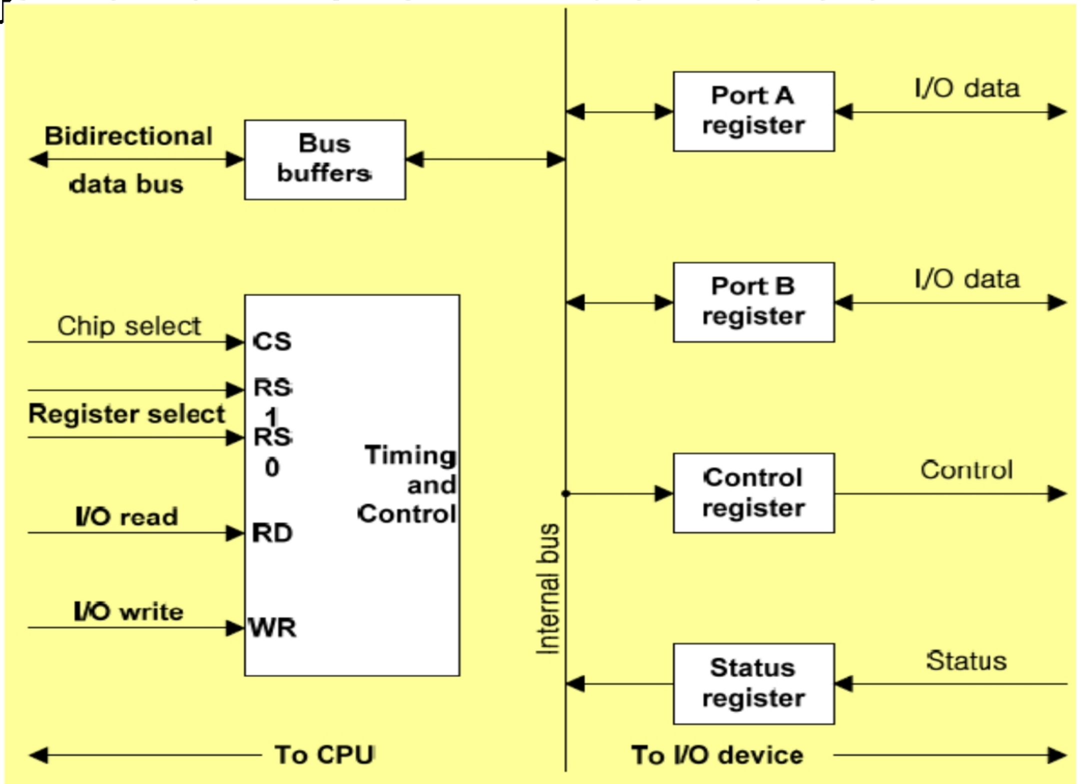
Memory-Mapped I/O



I/O Mapped I/O

Example of I/O Interface

- 4 I/O port:
    - Data port A,
    - Data port B,
    - Control,
    - Status
- Address Decode:
    - CS
    - RS1
    - RS0



CS	RS 1	RS 0	Register selected
0	×	×	None : data bus in high-impedance
1	0	0	Port A register
1	0	1	Port B register
1	1	0	Control register
1	1	1	Status register

Asynchronous Data Transfer

- Synchronous Data Transfer
  - All data transfers occur simultaneously during the occurrence of a clock pulse
  - Registers in the **interface** share a common clock with **CPU** registers
- Asynchronous Data Transfer
  - Internal timing in each unit (*CPU and Interface*) is independent
  - Each unit uses its own private clock for internal registers
  - No common clock

Asynchronous Data Transfer

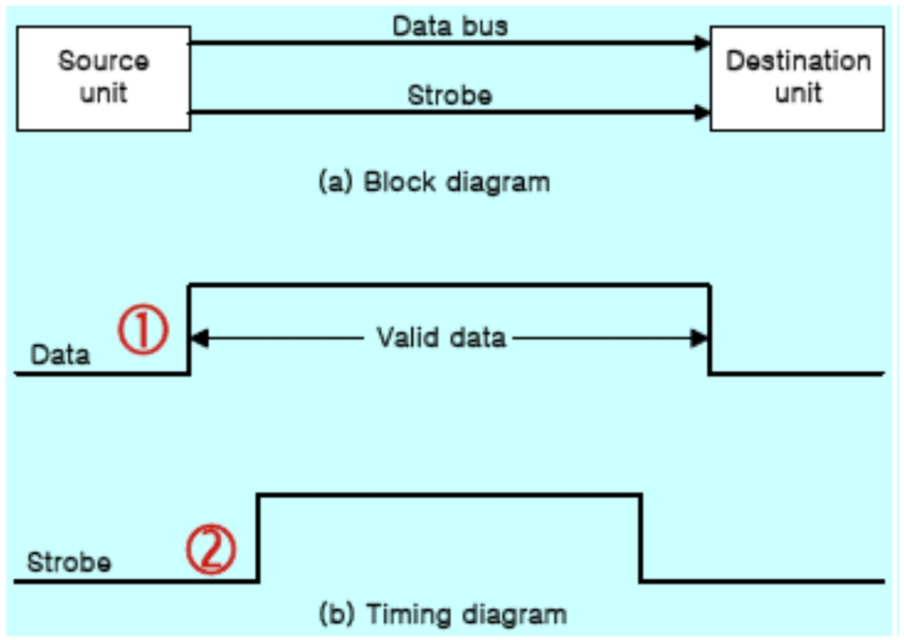
- Two Asynchronous Data Transfer Methods
  - **Strobe pulse**
    - A strobe pulse is supplied by one of the units to indicate when the transfer has to occur to the other unit.
  - **Handshaking**
    - A control signal is accompanied with each data being transmitted to indicate the presence of data
    - The receiving unit responds with another control signal to acknowledge receipt of the data

Strobe Control

- Employs a single control line to time each other.
- The strobe may be activated by either
  - the source or
  - the destination unit
- Data bus carries the binary information from source to destination

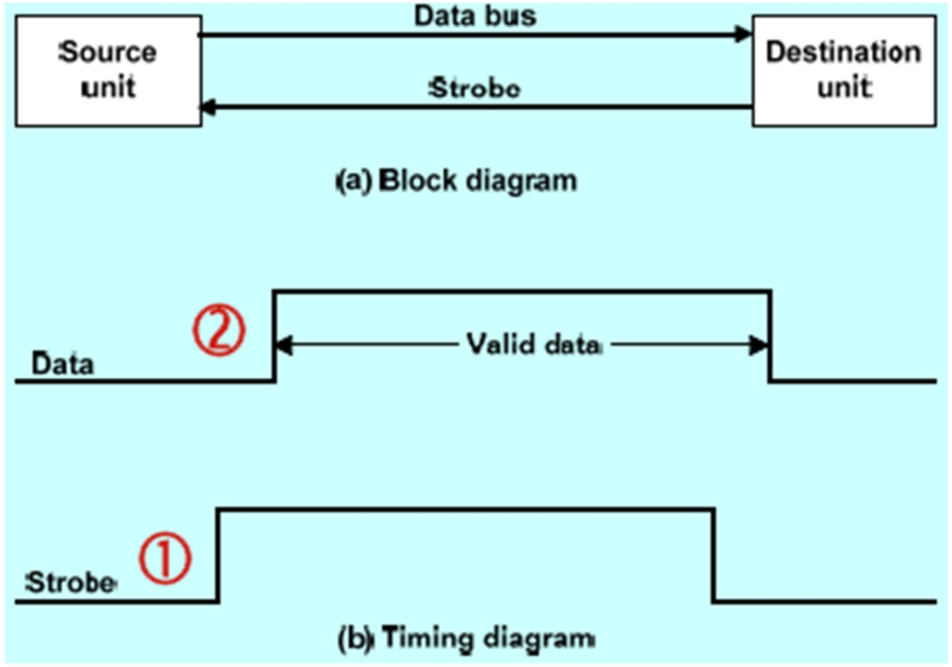
Source initiated strobe

- It could be a **memory write control signal** from the CPU to a memory unit.
- Source is the CPU, places a word on the data bus and informs the memory unit which is destination, that this is a **write operation**
- **Disadvantage** – no way of knowing whether the destination unit has actually received data



Destination initiated strobe

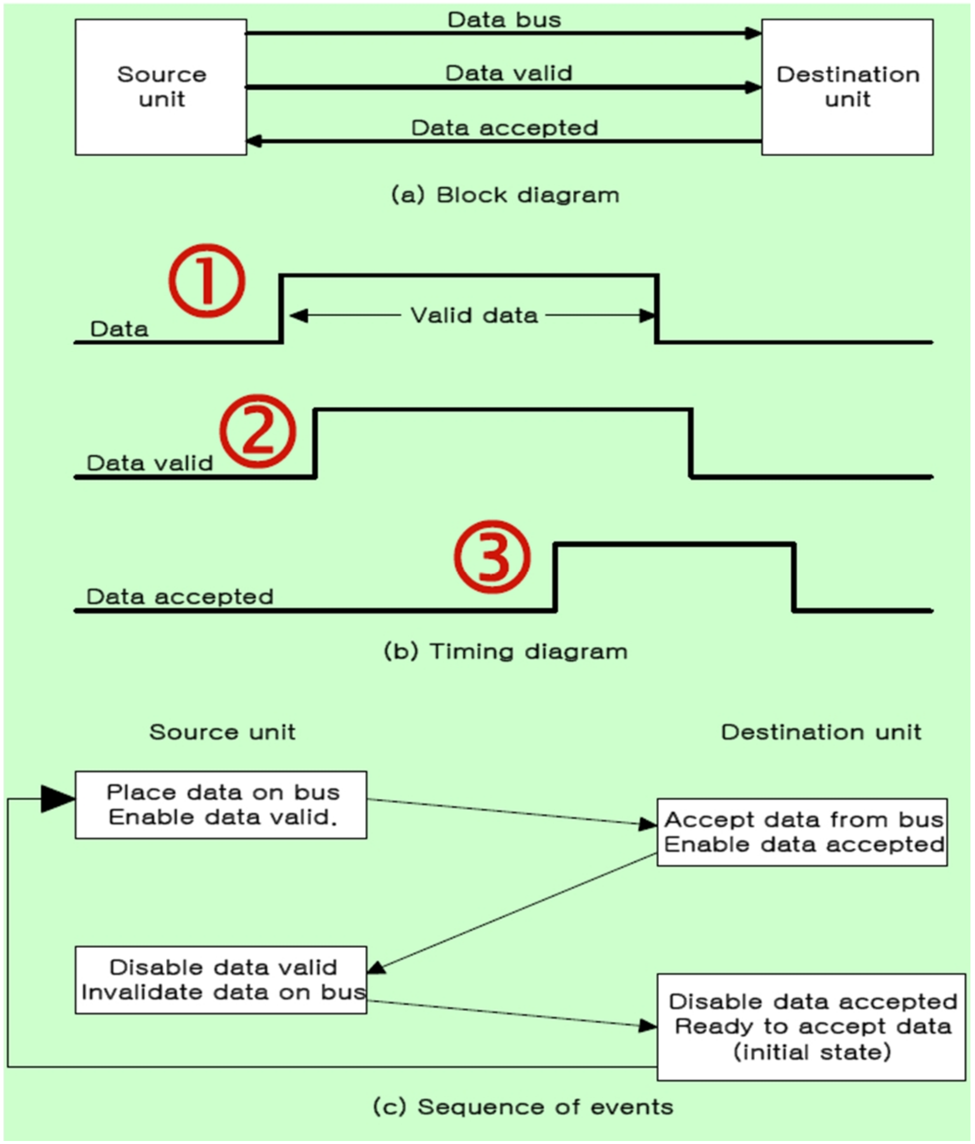
- It could be a **memory read control signal**.
- Destination is CPU, initiates the read operation to inform the memory which is the source to place a selected word into the data bus.
- **Disadvantage** - no way of knowing whether the source has actually placed the data on the bus



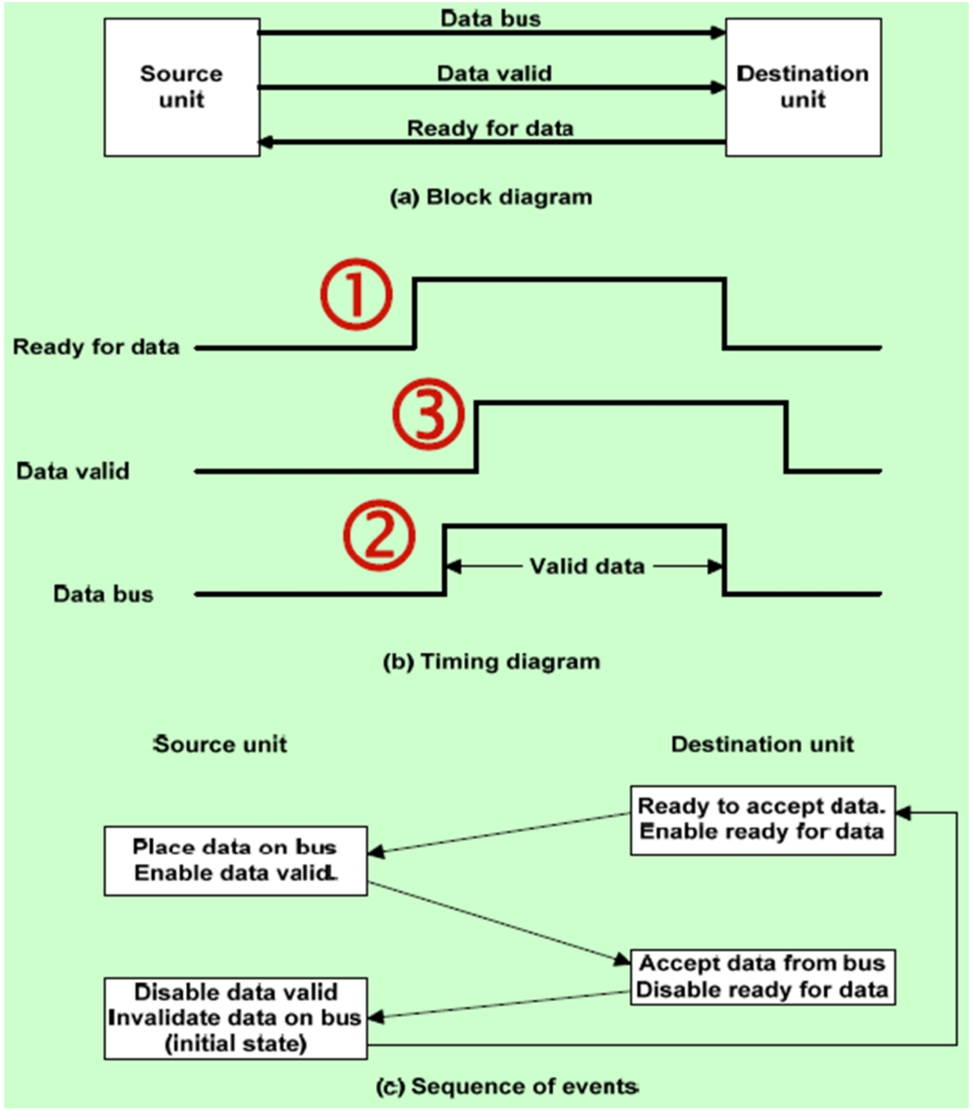
Hand Shaking

- To solve the disadvantages of strobe, the *HANDSHAKE* method introduces a second control signal to provide a *Reply* to the unit that initiates the transfer
- 2 types
  - Source initiated transfer
  - Destination initiated transfer
- Provides high degree of flexibility and reliability
- Incompletion of data transfer can be detected by means of a timeout mechanism.
- **Timeout** : If the return handshake signal does not respond within a given time period, the unit assumes that an error has occurred.
- Timeout signal can be used to interrupt the processor and hence execute a service routine that takes appropriate error recovery action.

Source-initiated handshake



Destination-initiated strobe :


Data bus
Source unit
Data valid
Destination unit
Ready for data
(a) Block diagram
①
Ready for data
③
Data valid
②
Valid data
Data bus
(b) Timing diagram
Source unit
Destination unit
Ready to accept data. Enable ready for data
Place data on bus Enable data valid.
Accept data from bus Disable ready for data
Disable data valid Invalidate data on bus (initial state)
(c) Sequence of events