Module 2.2
Interconnection System and DMA

ec

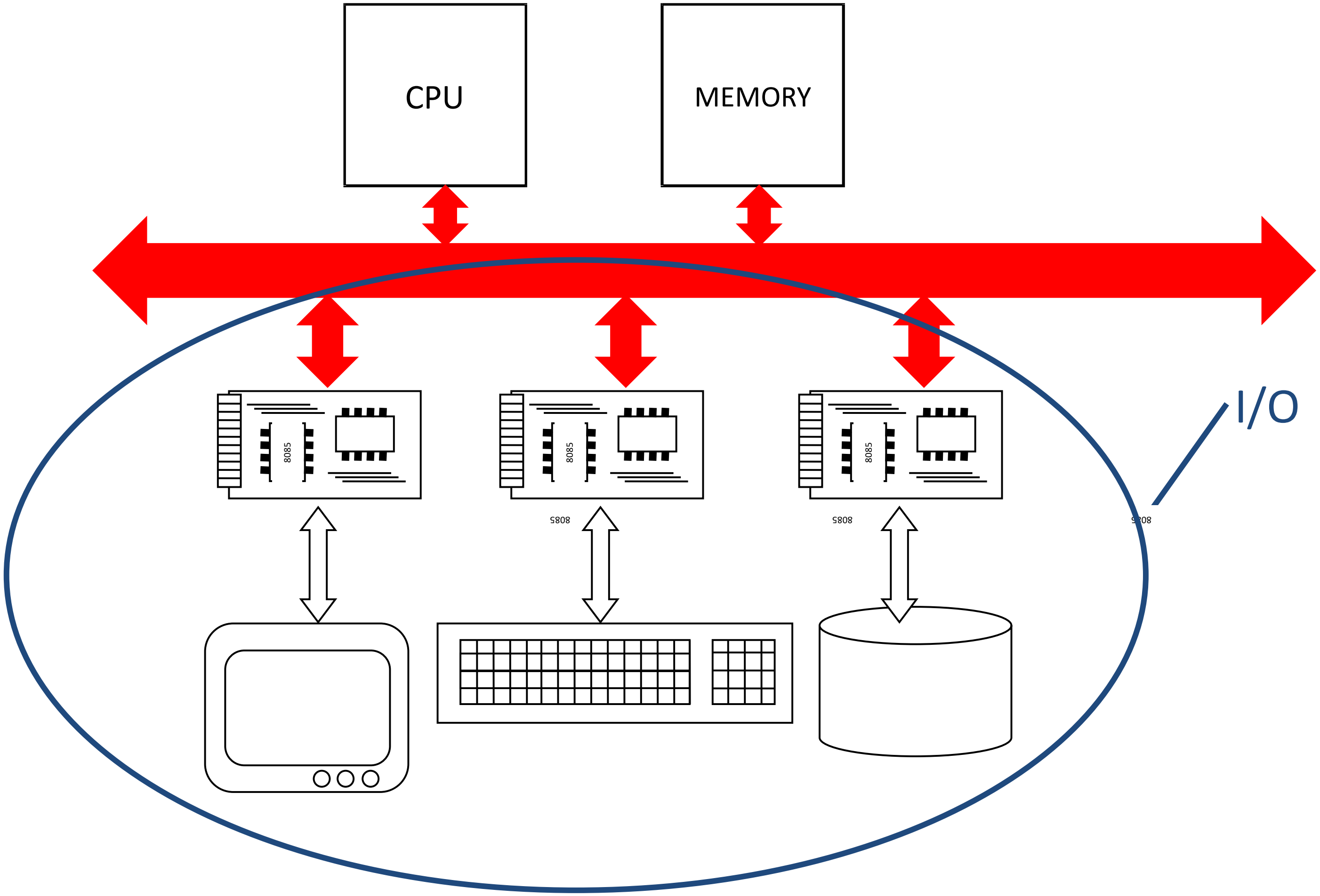
Outline

- Interconnection system
  - Taxonomy of interconection lines
  - Buses
  - Examples
- DMA
  - DMA controller
  - DMA transfer
  - DMA modes

- References:
  - D. A. Patterson & J. L. Hennessy; Computer Organization and Design. The hardware/software interface. Chapter 6, sections 5 (buses) and 6 (DMA).
  - W. Stallings; Computer Organization and Architecture.

Interconnection System

- A communication between the CPU, memory and the I/O devices is required.



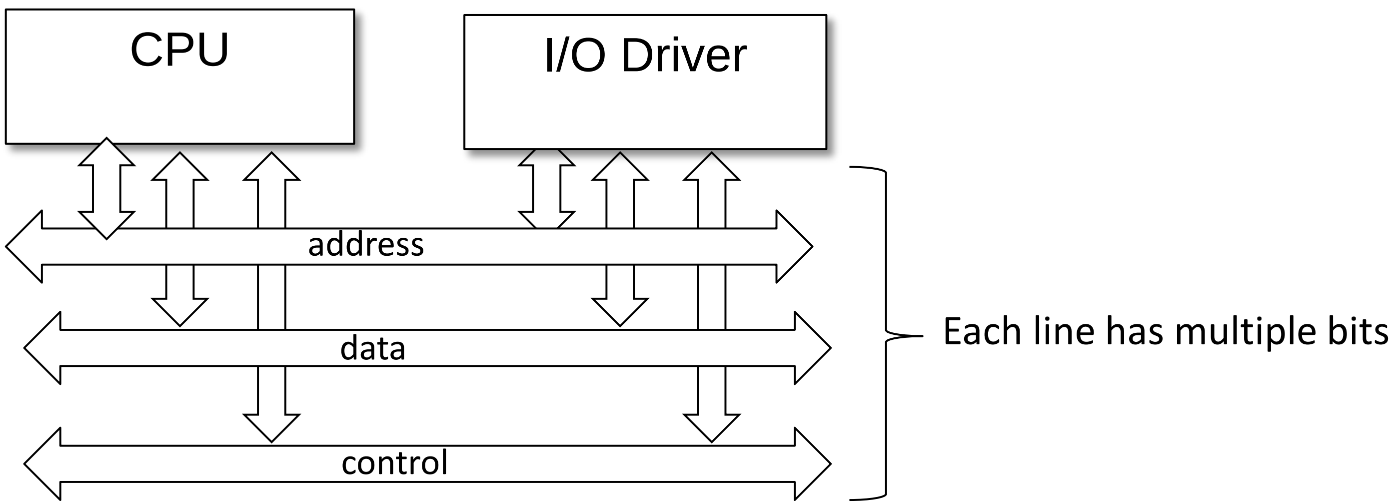
ec

Taxonomy of communication lines

- Depending on the "width" of the line
  - Serial
  - Parallel
- Depending on the synchronization mechanism
  - Synchronous
  - Asynchronous
- Depending on the topology
  - Shared
  - Point to Point

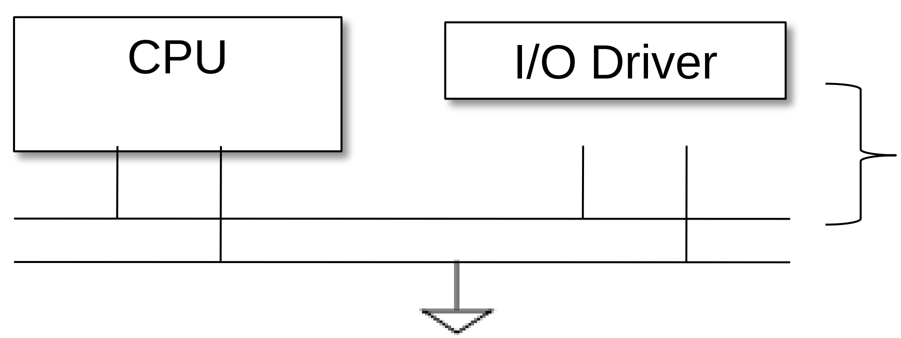
Width of the line: Parallel Transmission

- Based on the use of multiple communication lines (wires) through which **several information bits are sent simultaneously**
  - ✓ High bandwidth (theoretically)
  - ✗ It is very complex to connect devices from medium to long distances
  - ✗ Operating frequency is limited due to physical factors
  - ✗ The low-speed devices do not use the potential speed of the parallel transmission (mouse, modem ...)



Width of the line: Serial Transmission

- Single line of communication (2 wires: data and ground) through which the information bits are **sent sequentially**
  - ✓ It is less complex than the parallel I/O
  - ✓ It allows higher frequencies
  - ✗ Bandwidth, in the case of the same frequency, is smaller than when using the parallel transmission



All information (data, control ...) is transmitted sequentially

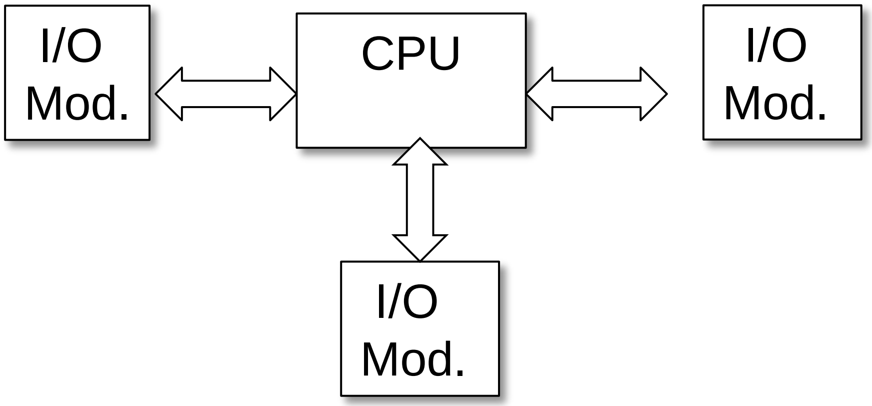
It is possible to increase the bandwidth between two devices using multiple serial connections

Synchronous vs. Asynchronous Communication

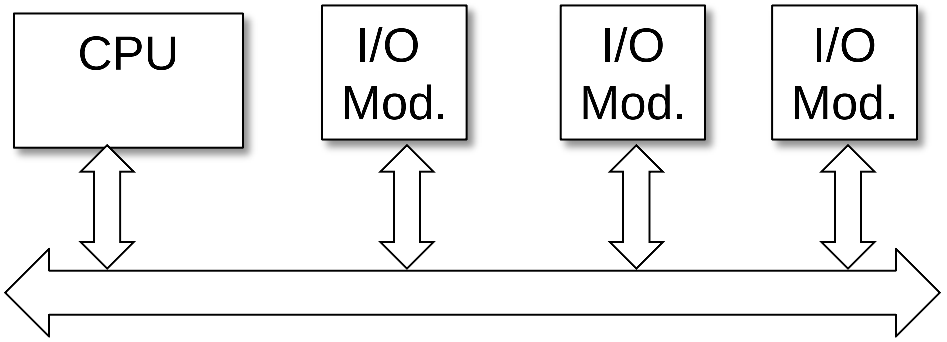
- When does the transmission of data start/finish?
- Asynchronous:
  - The clock signal is **NOT** sent through the communication line
  - The transmitter and the receiver use their own clock signals
  - It is necessary to establish a **synchronization protocol** among them
  - ✓ It is NOT essential that the sender and the receiver operate at the same frequency
- Synchronous
  - ✓ The clock signal is sent along with the other signals
  - ✓ Simpler and generally more efficient
  - ✗ It requires transmitter and receiver to operate at the same frequency
  - ✗ It should be short if it has to be fast, for the transmission not to be affected by the clock skew

Point to Point vs. shared bus

- Point to Point Communication
  - ✓ It can provide high performance
  - ✗ It requires a large number of interfaces

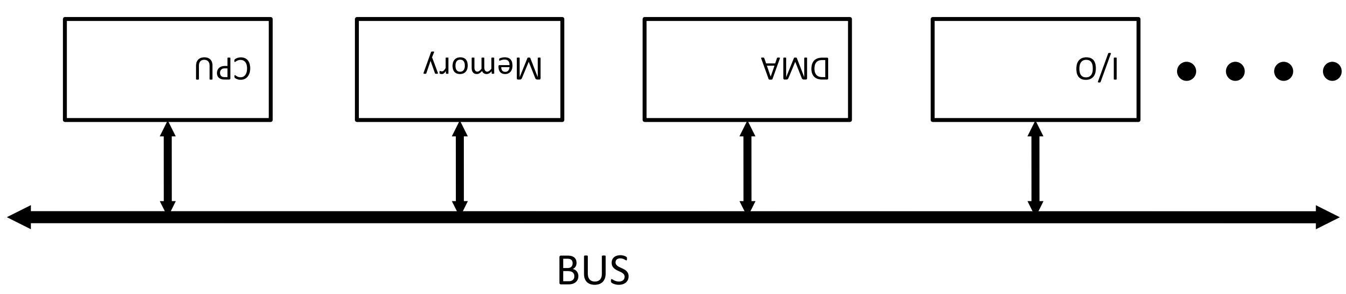


- Shared bus
  - ✓ More versatile and lower cost
  - ✗ It can create a communication bottleneck
  - ✗ It requires synchronization between devices to prevent concurrent writes



Buses

- Set of wires connecting multiple components:
  - Parallel
  - Shared
    - In this case, an **arbiter is needed**
  - There are synchronous and asynchronous options



General organization of a bus

- Control lines
  - Used to implement the communication protocol among components
- Data lines
  - They transmit information from sender to receiver
  - Width: number of bits that can be transmitted simultaneously
- Address lines
  - They contain routing/addressing information

Data transfer on a bus

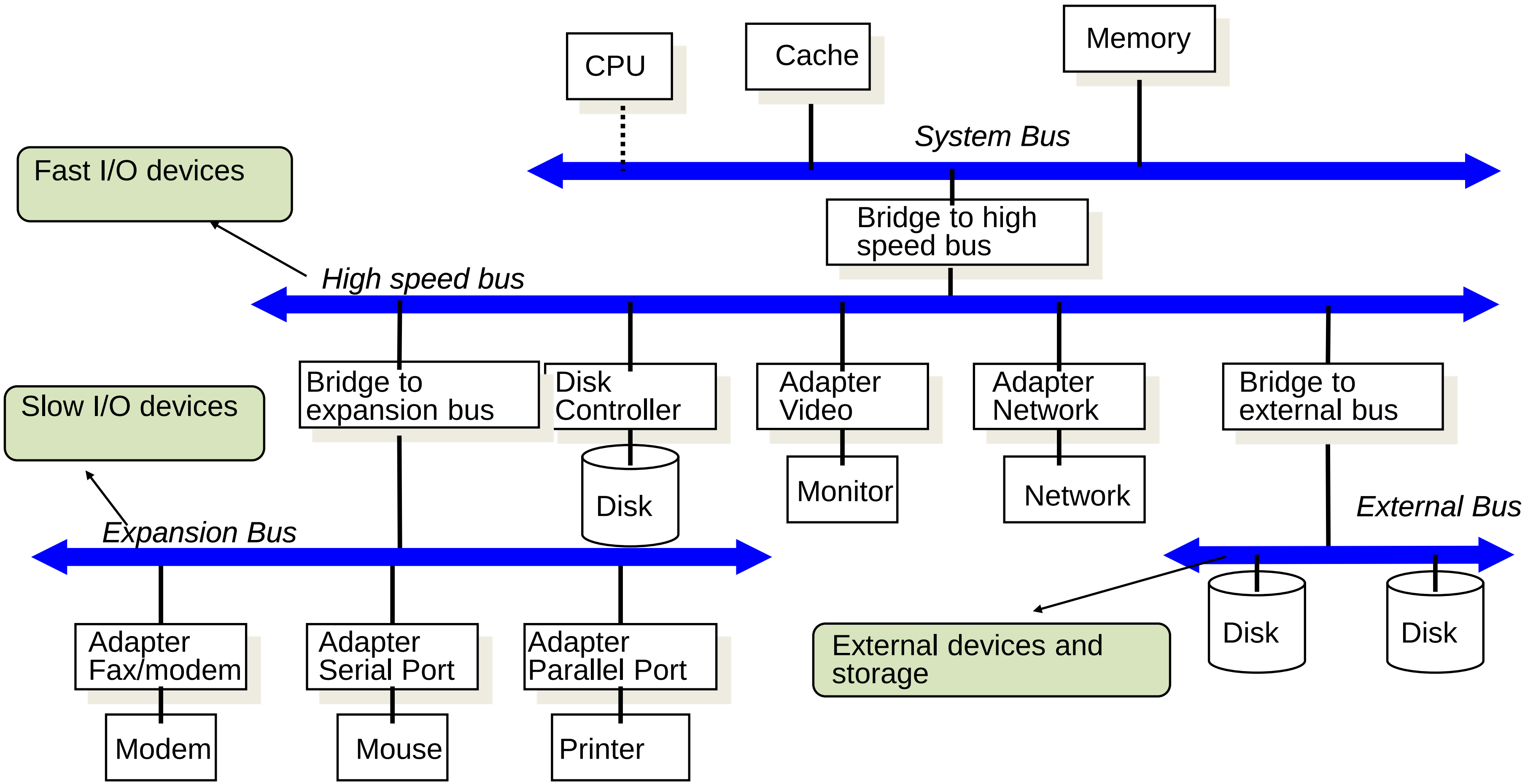
- Basic transfer types:
  - Writing: The Master sends data to the slave
  - Reading: The Slave sends data to the master

- Phases of a transfer
  1. Address (identificate) the slave
  2. Specify the type of operation (read or write)
  3. Transfer the data
  4. Completion of the bus cycle

- Control of the data transfer:
  - Synchronization: We must determine the beginning and end of each transfer
  - Arbitration: We must control access to the bus if there exist several masters

Hierarchy of buses

- In most systems (PCs, servers) there are many devices with very different operation frequency.
  – A single bus would be a severe bottleneck for the system
  – Different buses work at different speeds



ec

Example 1: Intel

Communication bridges. Chipset Intel 2004.

Modern PCs have two chips that manage the interconnection system: the Northbridge or Memory Controller Hub and the Southbridge or I/O Controller Hub.

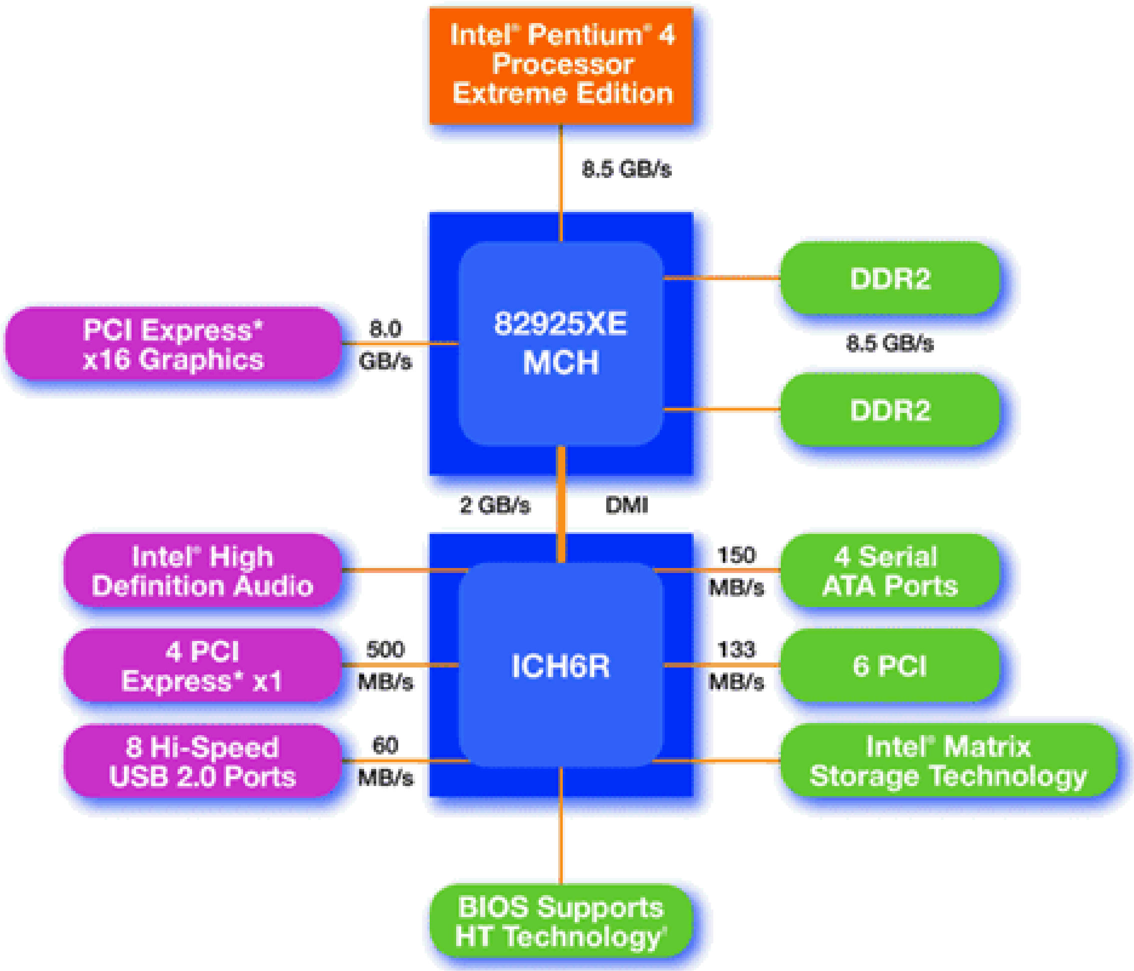
➢ **The Northbridge** controls the connection of the CPU and the high speed components:

- RAM memory
- High speed buses (i.e. PCI)
- Southbridge

➢ **The Southbridge** is in charge of coordinating the I/O devices and other low speed functions.
Its communication with the CPU is through the Northbridge.

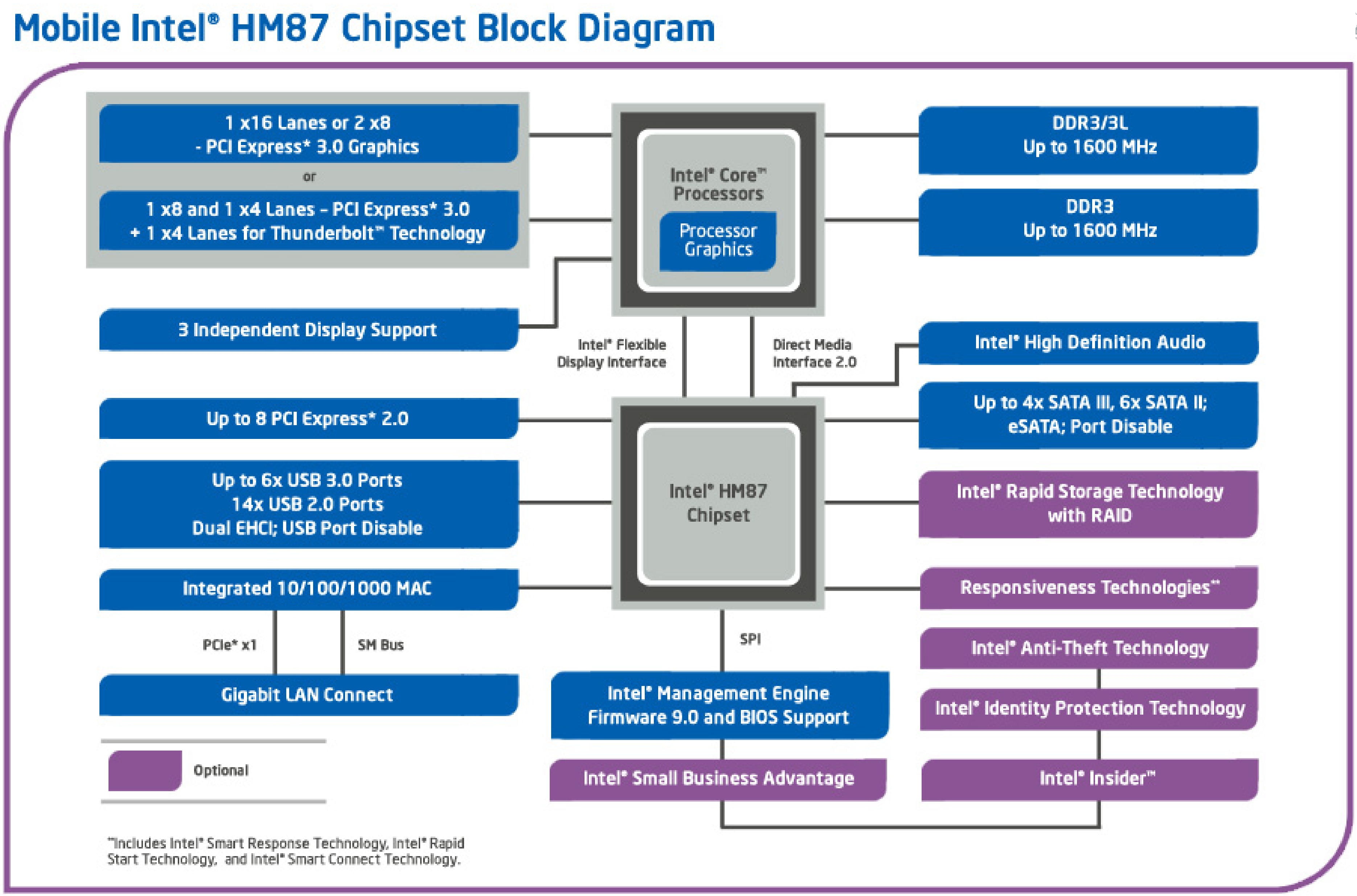
It can support:

- PCI bus
- ISA bus
- Interrupt controllers
- Ethernet
- RAID
- USB
- Audio Codec



Example 1: Intel

Actual systems are more complex, including point to point connections (QPI, PCIe) together with buses.



Example 2 – AMBA AHB-Lite

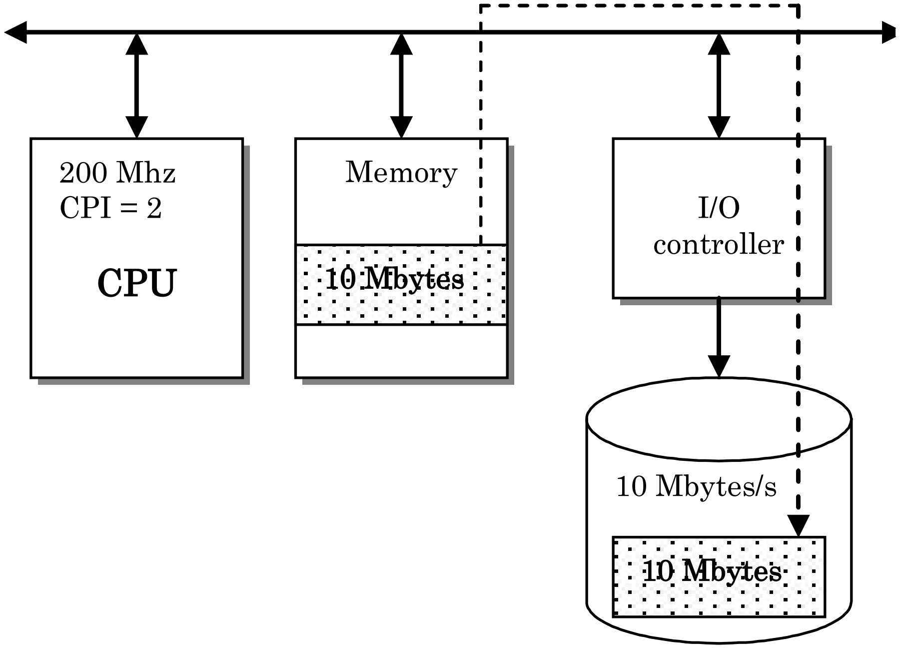
- Advanced High-performance Bus (AHB):
  - Bus included in 2nd Advanced Microcontroller Bus Architecture (AMBA) Specification from ARM.
- AHB-Lite:
  - Bus included in 3rd AMBA Specification from ARM.
  - Conceived for implementing much simpler peripherals.
  - Used in ARM and MIPS processors.
  - It simplifies the AHB specification by removing the protocol required for multiple bus masters, which includes the request/grant protocol to the arbiter and the split/retry responses from slaves.

DMA

- What is the Direct Memory Access (DMA)?
  - The DMA technique allows data to be transferred between a peripheral and memory without CPU intervention (except in the initialization phase to set the transfer parameters)
    - With interrupts, the waiting loop is avoided but the transfer is still handled by the processor
      - For an N bytes block, N interrupts are generated
    - Using DMA, the transfer is completely done by the DMA controller
      - Only one interruption when the transfer ends

I/O transfer for a high speed device

10 MBytes/s means 1 byte every 100 ns.
200MHz means clock cycle = 5ns.
If the I/O controller generates an interrupt every time there is a byte ready, there will be one interrupt every 20 cycles!

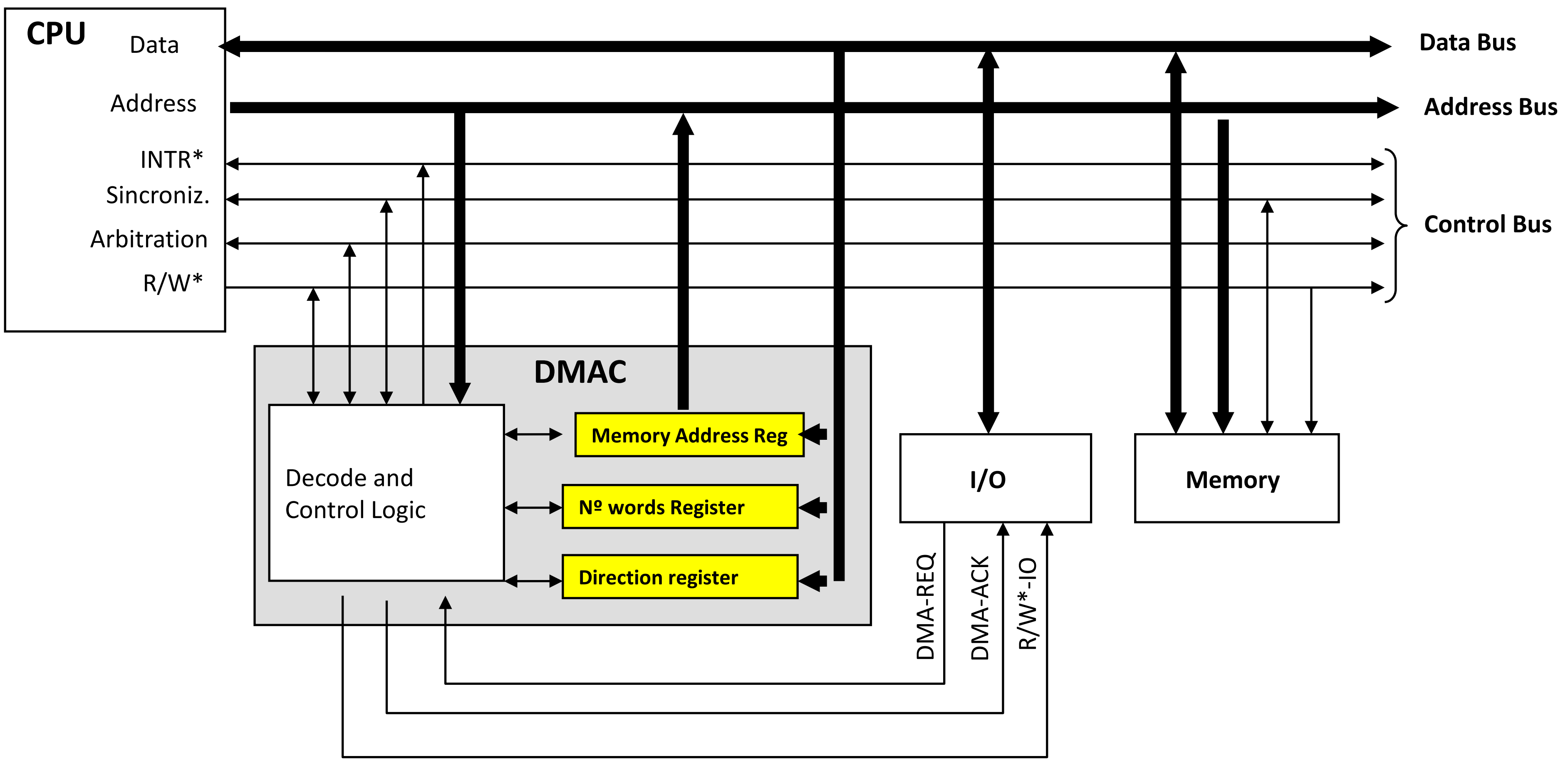


The DMA Controller (DMAC)

- The DMA controller (DMAC) is a device capable of controlling data transfers between a peripheral and memory without CPU intervention
- The DMAC acts as a master in the bus during a DMA transfer, and should be able to:
  - Request the use of the bus via appropriate signals and arbitration logic
  - Have a specific memory address on which to make the transfer
  - Generate the bus control signals
    - Type of operation (read/write)
    - Synchronization signals in the transfer

DMAC: registers

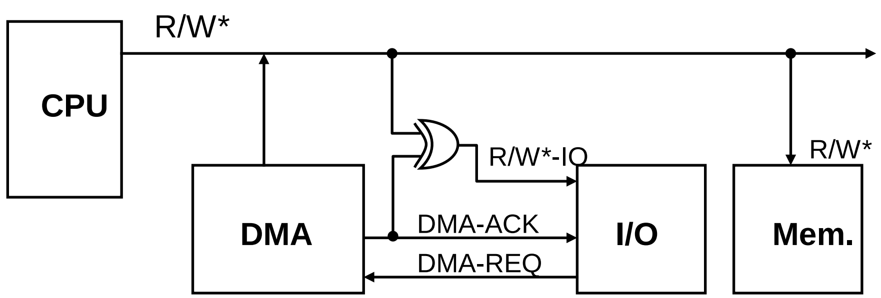
- **Register to store the memory address** : It stores the starting memory address and it is incremented/decremented after the transfer of each word
- **Register to store the # of words**: It stores the number of words to be transferred and it is decremented after transferring each word
- **Direction Register:** It stores the direction of the transfer (read or write)



DMAC: Signals

- **DMA-REQ:** Request DMA service
  - The peripheral/device tells the DMAC that it is ready to start transmitting/receiving
- **DMA-ACK:** Grant of the DMA service
  - DMAC tells the peripheral/device that it can start the transfer
  - Before setting this signal, the DMAC must acquire the bus
- **R/W*-IO**: Direction of the transfer to the peripheral/device

Generation of control signals R/W* and R/W*-IO



DMA-ACK	R/W*	R/W*-IO
0	0	0
0	1	1
1	0	1
1	1	0

Normal I/O operation (controlled by the CPU)
R/W*-IO = R/W*

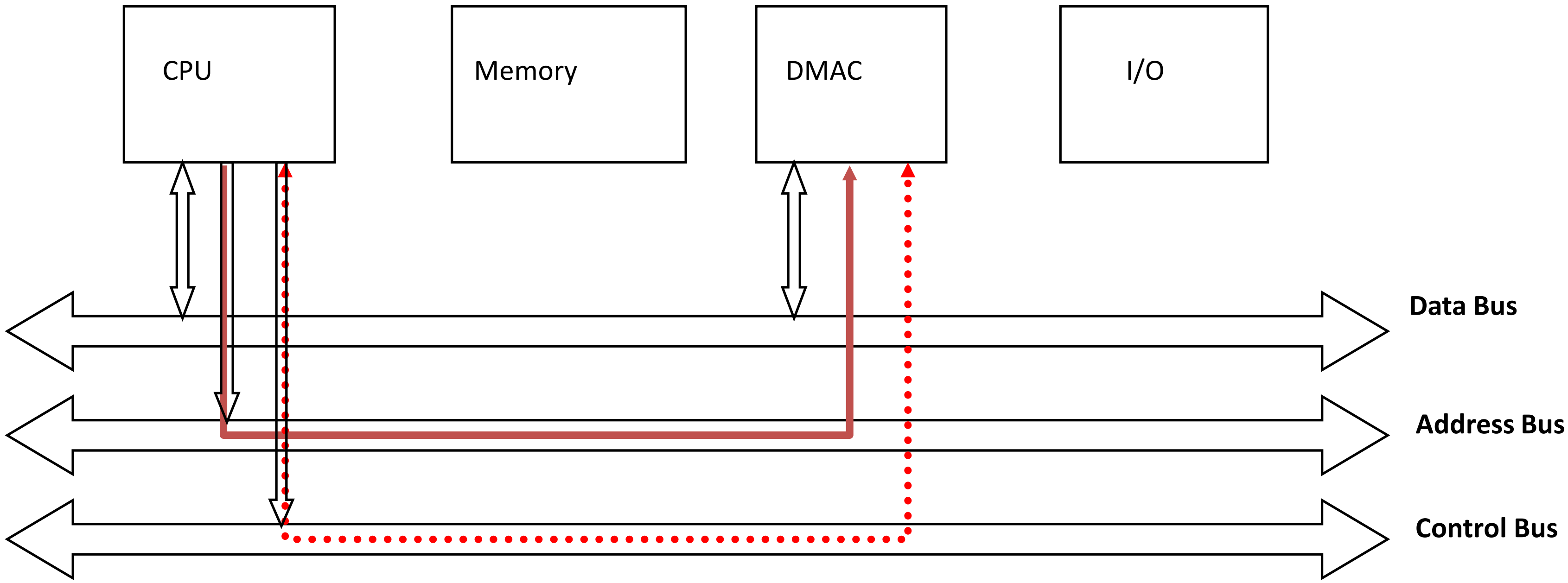
DMA controlled I/O operation
R/W*-IO = NOT (R/W*)

Stages of a DMA transfer

- Initialization of a transfer
  - The CPU must send the transfer parameters to the DMAC
  - Initialization of the DMA controller (master bus: CPU - slave bus: DMAC)
    - Number of bytes or words to transfer
    - Type of transfer (read/write)
    - Starting memory address of the transfer
    - Channel No. (only needed for multi-channel DMAs)
  - After initialization, the CPU can continue performing other tasks without paying attention to the evolution of the transfer
  - DMAC then sends the transfer parameters to the controller of the device
  - Device Driver Initialization (master bus: DMA – slave bus: Device Driver)
    - Number of bytes to transfer
    - Type of transfer (read/write)
    - Other control information (track, sector, etc.)

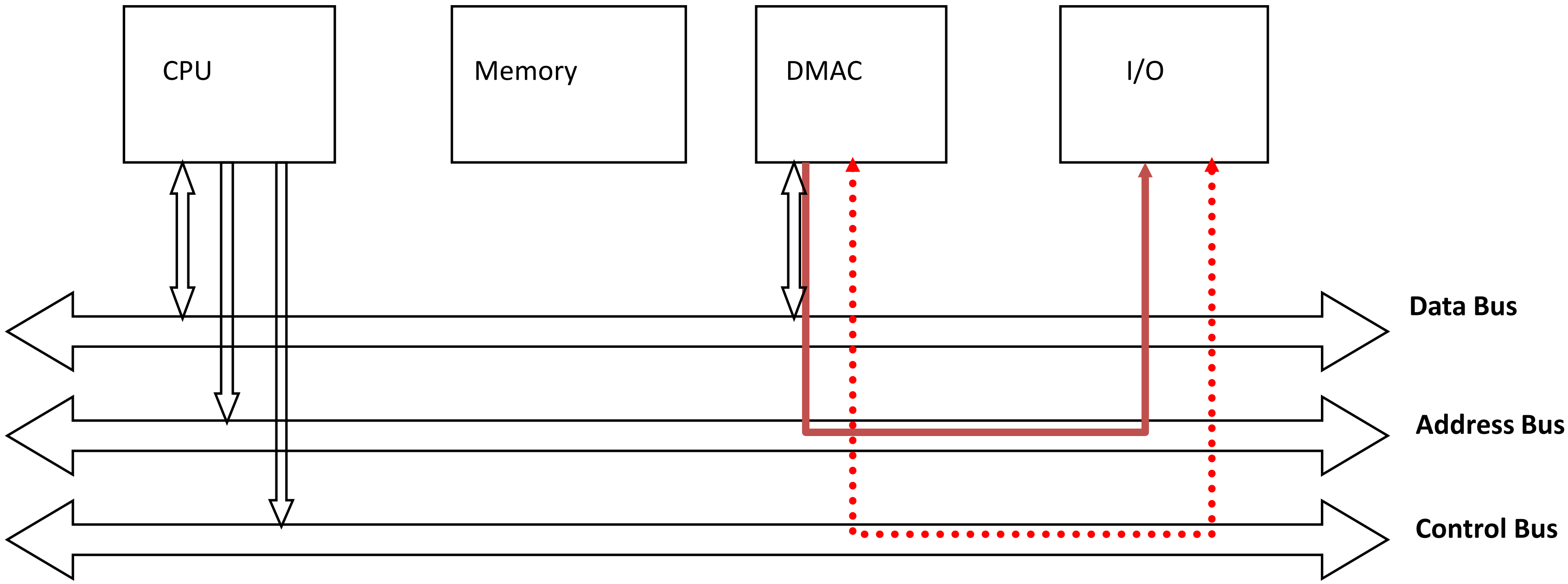
Initialization of a transfer

1. The CPU programs the DMA controller to execute the full I/O operation.
    - Starting memory address → Memory address Register
    - Number of words to be transferred → Register for counting transferred words
    - Direction of the transfer → Direction Register



Initialization of a transfer

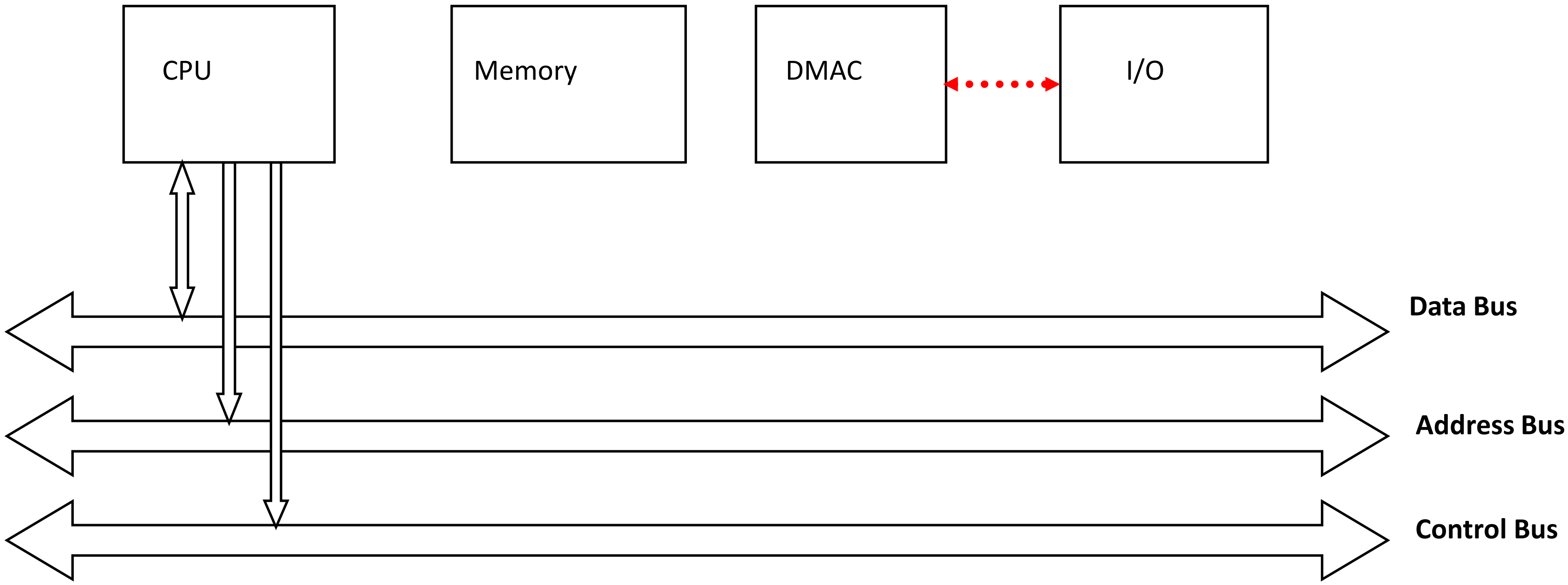
2. The DMA controller programs each single I/O operation as in basic program driven I/O.



ec

Transfer process

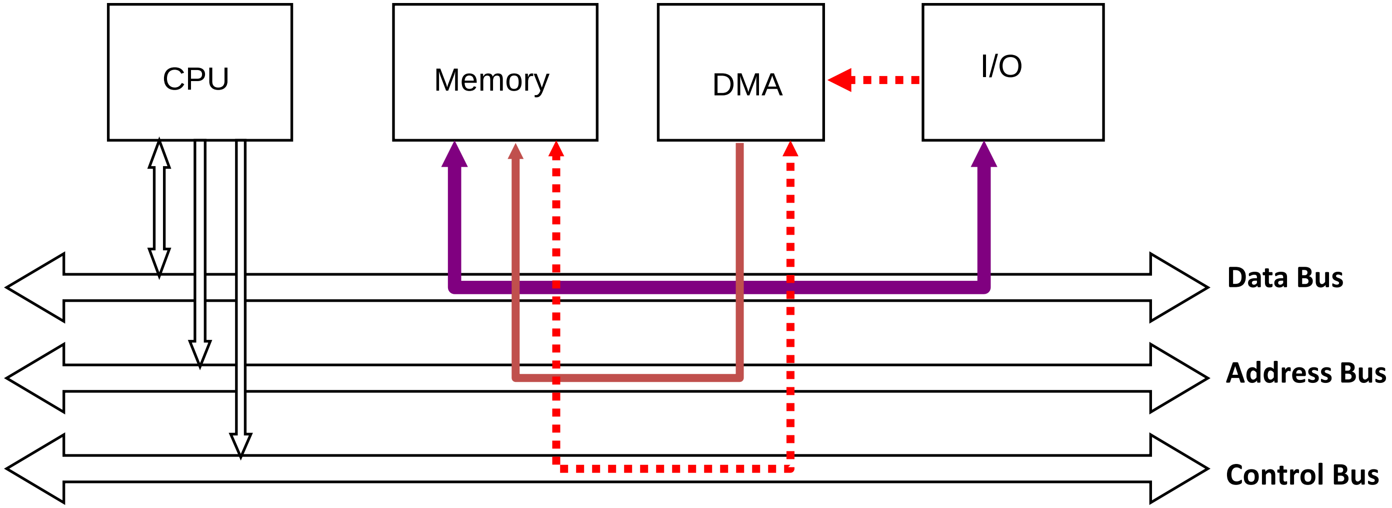
3. The device driver notifies that it is ready for starting the transfer.
   It sets the ***DMA-REQ*** signal



Transfer

4. The DMA acts as bus master (instead of the CPU) and controls the I/O operation.

After each transfer the DMA controller updates its registers and prepares the following I/O operation (if any) or interrupts the CPU if all operations are complete.



Possible Problems when using DMA

- Performance of the CPU can be degraded if the DMAC has an intensive use of the bus
  - The CPU cannot access memory (to read instructions/data) when the bus is busy in a DMA transfer
- This problem is reduced with the use of cache memory
  - When the CPU is hitting in the cache, it does not need to use the memory bus
  - The DMAC can use the intervals when the CPU is hitting in the cache (in which the CPU is not accessing the memory bus)
- In computers without cache memory
  - The processor does not use the bus in all stages of the pipeline
  - In these stages, the DMAC can use the bus

DMA transfer on burst mode

- The DMAC requests the bus control to the CPU
- When the CPU grants the use of the bus to the DMAC, the DMAC does not release it until finishing the transfer of the entire data block
- ADVANTAGES:
  - It achieves the fastest possible transfer speed
- DISADVANTAGES:
  - The CPU cannot use the memory bus during the DMAC transfer, which can degrade System/CPU performance

DMA transfer on cycle stealing mode

- The DMAC requests bus control to the CPU, which grants it after finishing the instruction currently in-flight
- When the CPU grants the bus to the DMAC, the DMAC transfers only one word and then releases the bus
- The DMAC requests again the control of the bus as many times as necessary until completing the transfer of the whole block
- ADVANTAGES:
  - System performance is negligibly degraded (in the worst case, only 1 cycle per instruction)
- DISADVANTAGES:
  - It takes longer to complete the transfer of a block

DMA transfer on transparent mode

- The DMAC requests control of the bus to the CPU, which grants it ONLY when it does not need it
- When the CPU grants the bus to the DMAC, the DMAC transfers only one word and then releases the bus
- The DMAC requests again the control of the bus as many times as necessary until completing the transfer of the whole block
- ADVANTAGES :
  - System performance is not degraded at all
- DISADVANTAGES :
  - It takes longer to complete the transfer of a block