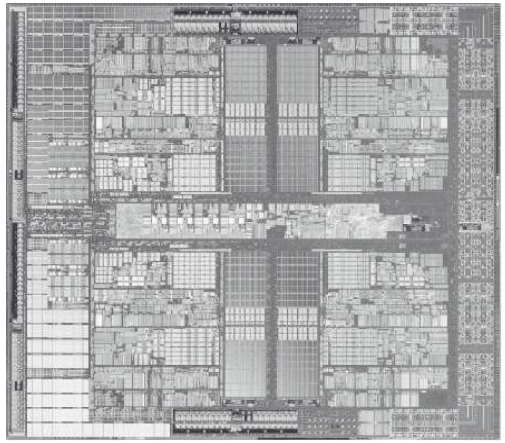
Università degli Studi di Cassino e del Lazio Meridionale



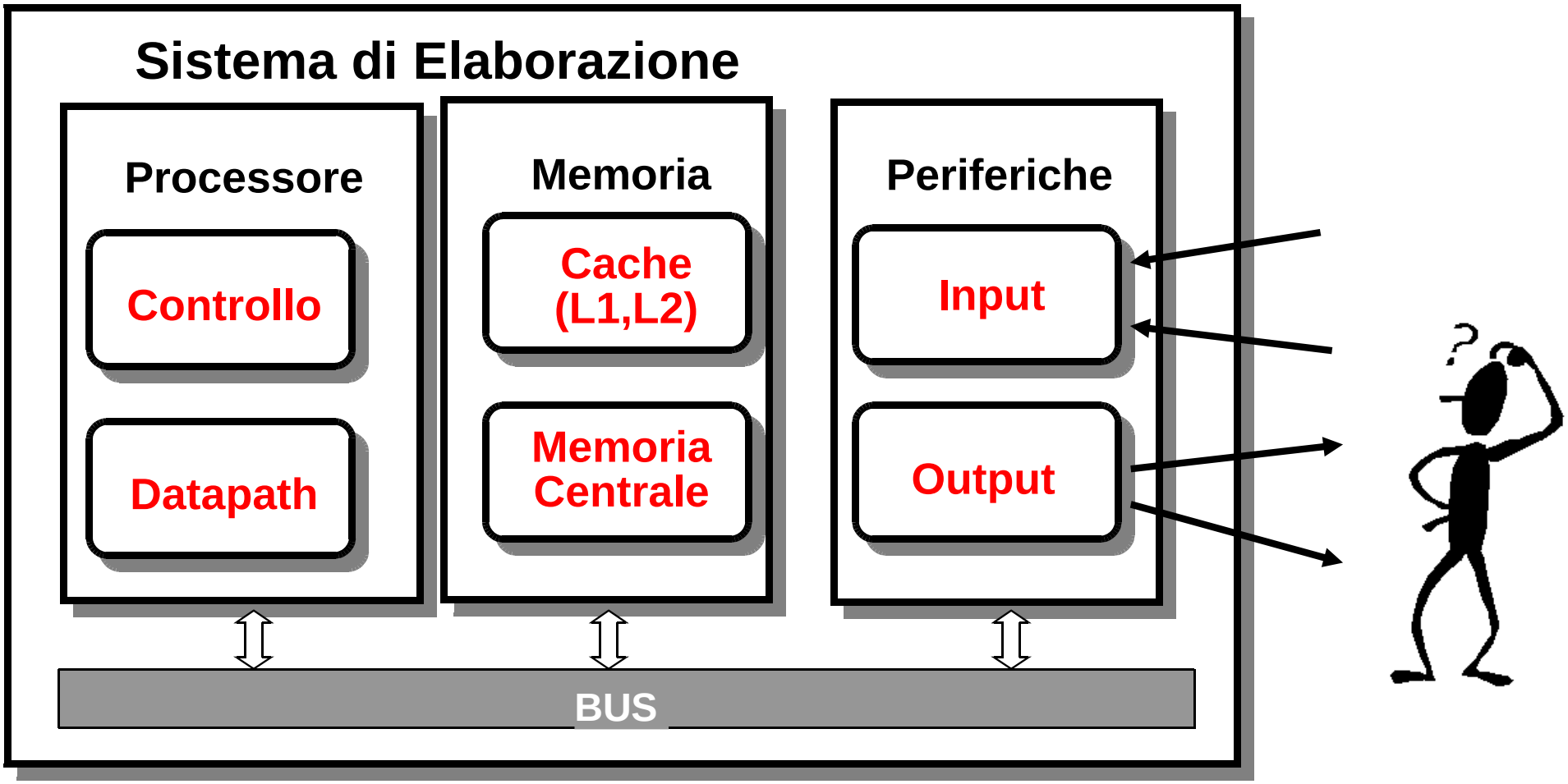
Corso di Calcolatori Elettronici

Gestione delle periferiche

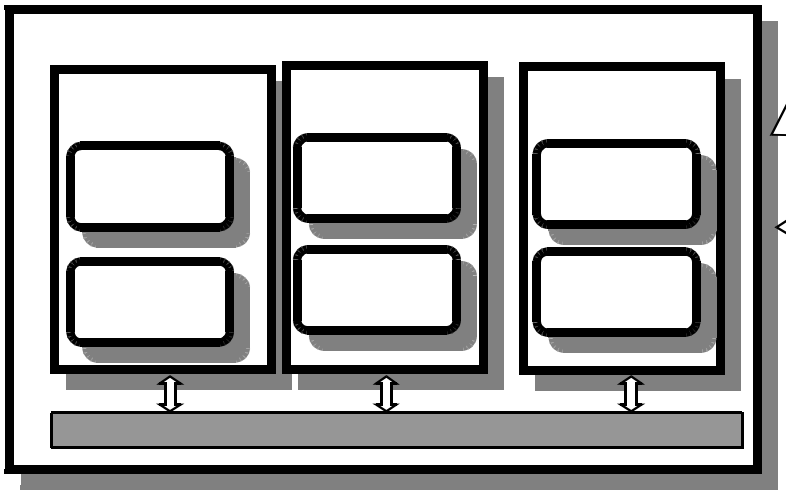
Anno Accademico 2011/2012

Francesco Tortorella

Input/Output:
da un sistema di elaborazione a … ?



...a ?



Motivazioni per l' Input/Output

- Il sistema di I/O definisce l'interazione tra calcolatori ed esseri umani e permette operazioni notevoli:

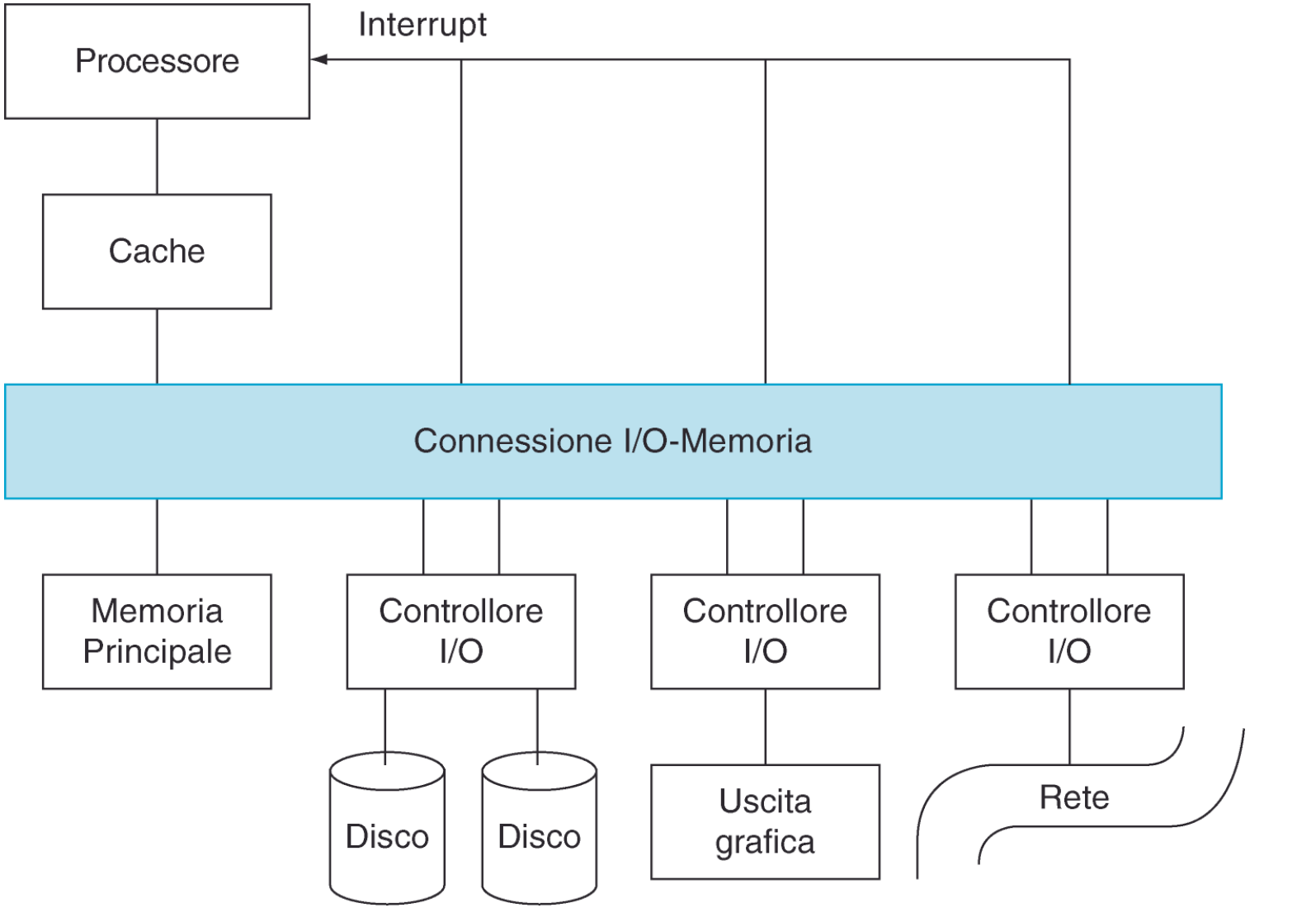
  - Acquisisce dati sensoriali (pressione, temperatura) dalla mano sintetica del vigile del fuoco Ken Whitten e li inoltra sul sistema nervoso.

  - Controlla la propulsione, l'assetto, gestisce la comunicazione in BOB (Breathable Observable Bubble)

  - Legge I codici a barre degli articoli nel frigorifero (e prepara la lista della spesa)

Computer without I/O like a car without wheels; great technology, but won't get you anywhere

Organizzazione generale della periferia

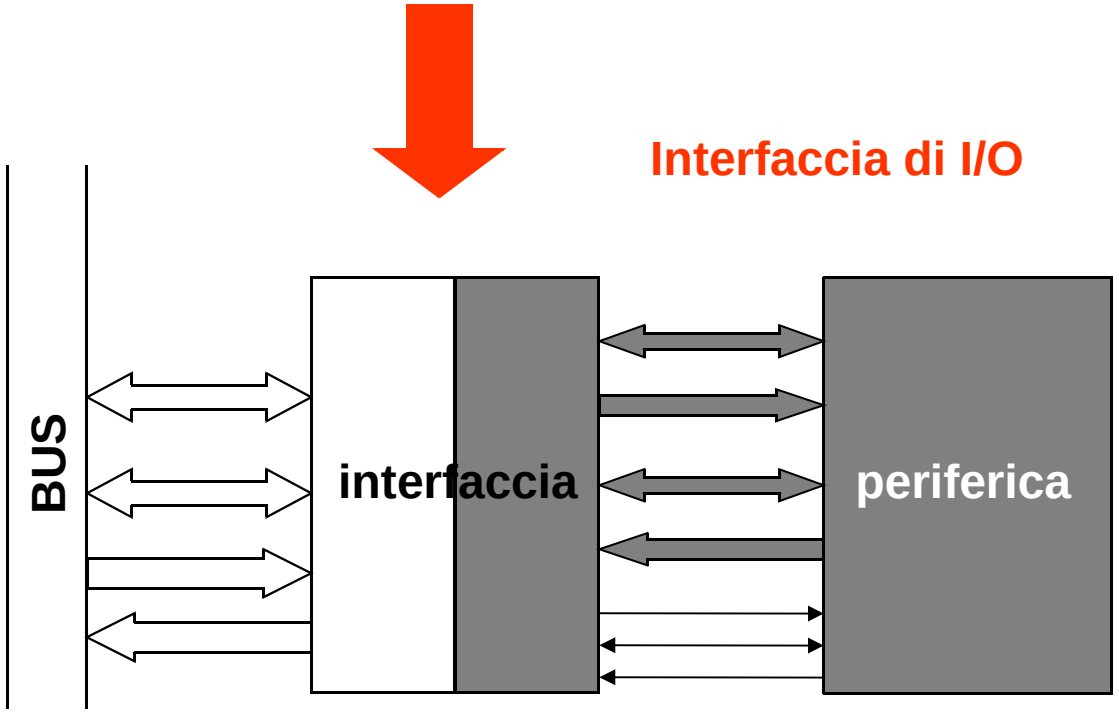


Dispositivi: esempi e velocità

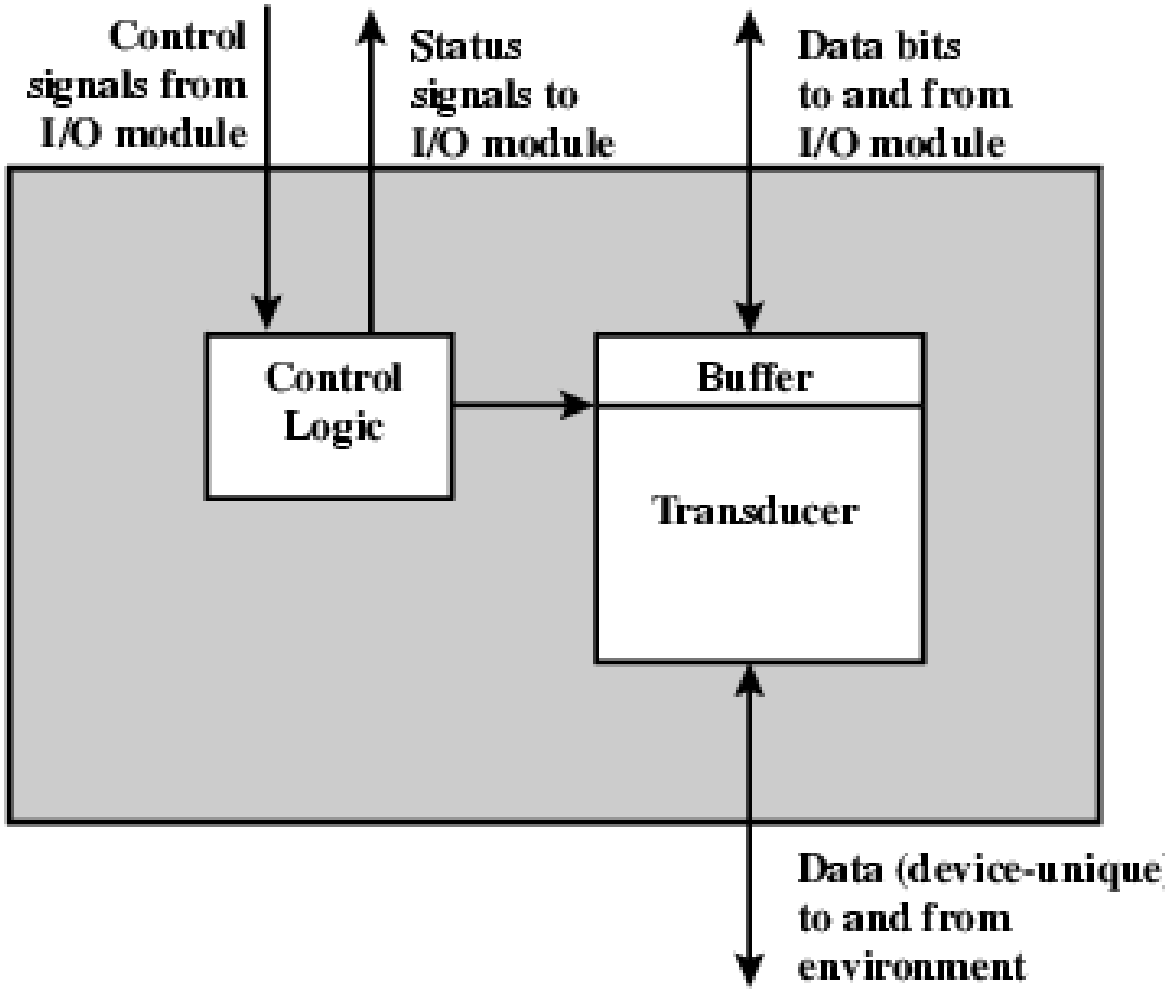
Dispositivo	Comportamento	Partner	Frequenza dati (Mbit/s)
Tastiera	Input (ingresso)	Uomo	0,0001
Mouse	Input (ingresso)	Uomo	0,0038
Input vocale	Input (ingresso)	Uomo	0,2640
Input audio	Input (ingresso)	Macchina	3,0000
Scanner	Input (ingresso)	Uomo	3,2000
Output vocale	Output (uscita)	Uomo	0,2640
Output audio	Output (uscita)	Uomo	8,0000
Stampante laser	Output (uscita)	Uomo	3,2000
Display grafico	Output (uscita)	Uomo	800,0000-8000,0000
Modem via cavo	Input o output	Macchina	0,1280-6,0000
Rete/LAN	Input o output	Macchina	100,000-10000,0000
Rete/LAN wireless	Input o output	Macchina	11,0000-54,0000
Disco ottico	Memoria	Macchina	80,0000-220,0000
Nastro magnetico	Memoria	Macchina	5,0000-120,0000
Memoria flash	Memoria	Macchina	32,0000-200,0000
Disco magnetico	Memoria	Macchina	800,0000-3000,0000

Interfaccia di I/O

- **Molti tipi diversi di periferiche**
- **Grossa variabilità all'interno di una stessa classe di periferiche**
- **Necessità di uniformare la connessione tra la periferica ed il resto del sistema**



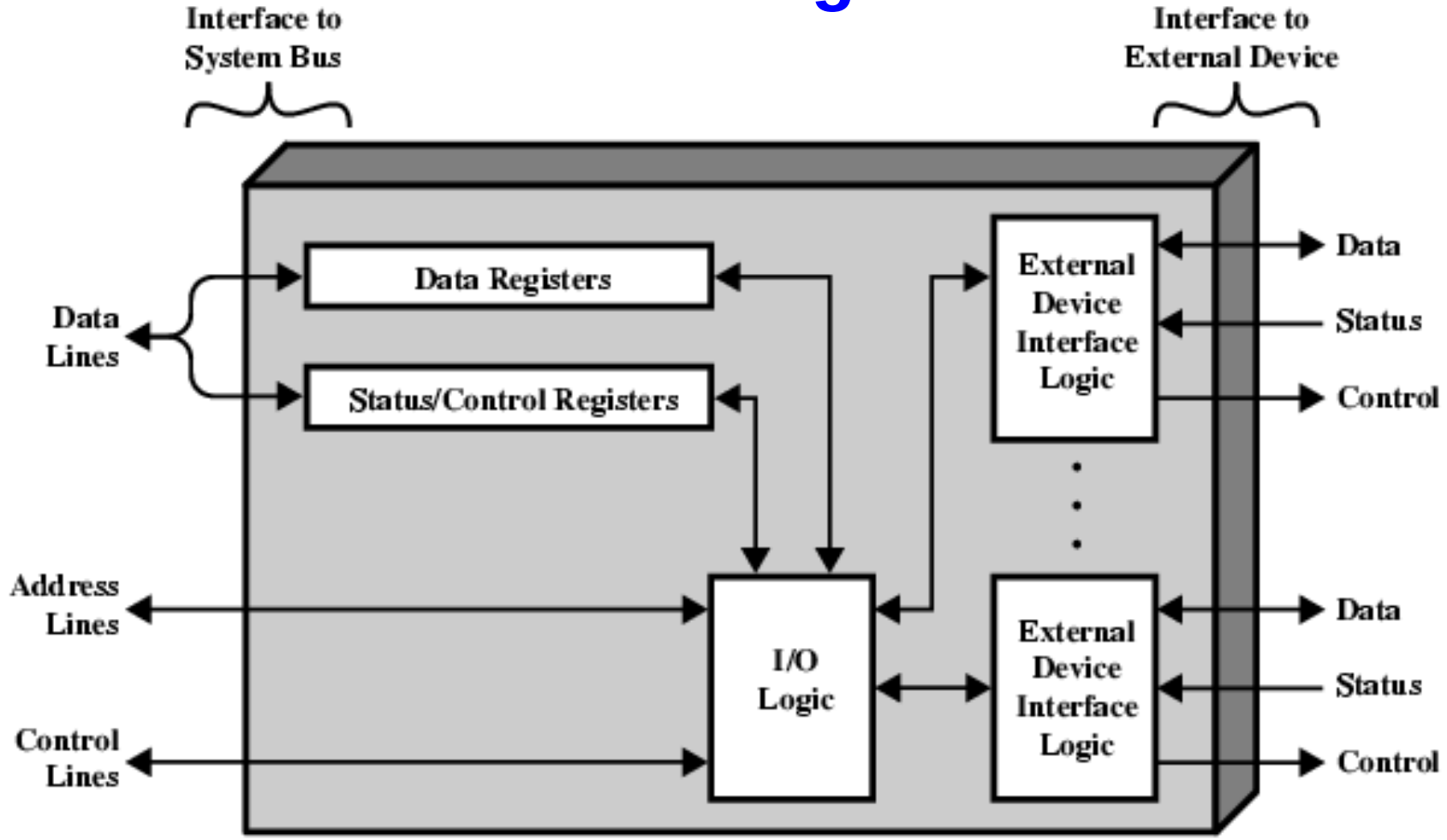
Interfaccia di I/O: schema della periferica



Funzioni dell'interfaccia di I/O

- Controllo e tempificazione
- Comunicazione con la CPU
- Comunicazione con la periferica
- Buffering dei dati
- Error Detection

Interfaccia di I/O: schema generico



Interfaccia di I/O

- L'interfaccia è accessibile tramite bus in maniera analoga ad un modulo di memoria (viene selezionata dal processore tramite un indirizzo)
- Le comunicazioni avvengono tramite operazioni di lettura/scrittura su un insieme di registri (*I/O ports*) appartenenti all'interfaccia:
  - Data IN: registro dati in ingresso
  - Data OUT: registro dati in uscita
  - Control Register (in): riceve istruzioni per le operazioni della periferica
  - Status Register (out): contiene informazioni sullo stato della periferica
- L'accesso ai registri è multiplexato:
  - interfaccia mappata su due indirizzi, multiplexati in funzione di un segnale di controllo READ/WRITE

    IND1: Data IN / Data OUT

    IND2: Control / Status

Connettiamo tutto !

- Sistema di elaborazione:
  struttura formata da unità diverse (CPU, moduli di memoria, moduli di I/O) collegate e comunicanti
- Ogni unità presenta tipi di connessioni diverse
- Esaminiamo
  - Un modulo di memoria
  - Un modulo di Input/Output
  - La CPU

Connessione del modulo di memoria

Riceve ed invia dati

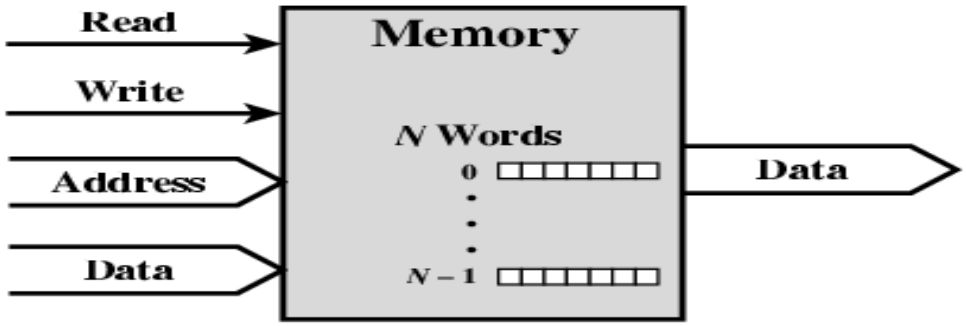
Riceve indirizzi di registri

Riceve segnali di controllo

Read

Write

Tempificazione



Connessione del modulo di I/O (1)

Simile al modulo di memoria

Output

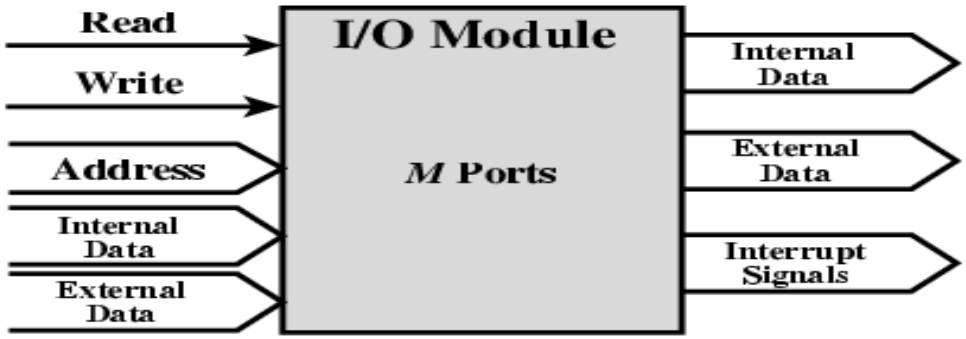
Riceve dati dal sistema

Invia dati alla periferica

Input

Riceve dati dalla periferica

Invia dati al sistema



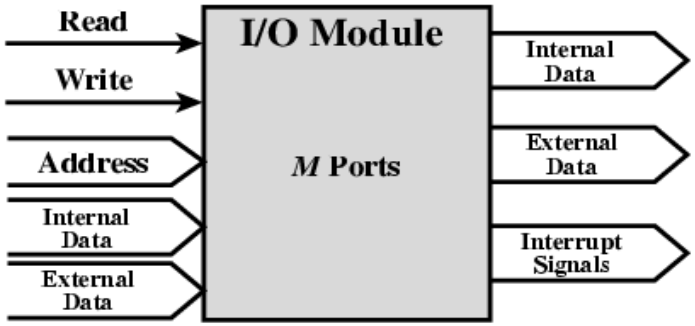
Connessione del modulo di I/O (2)

Riceve segnali di controllo dal sistema

Invia segnali di controllo alla periferica

Riceve indirizzi dal sistema (p.es. per identificare la periferica)

Invia al sistema segnali di richiesta di interruzione



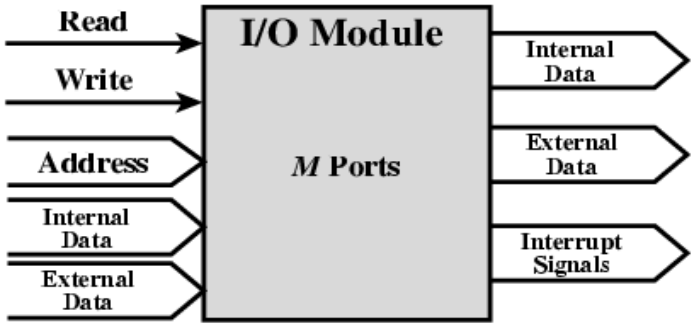
Connessione del modulo di I/O (2)

Riceve segnali di controllo dal sistema

Invia segnali di controllo alla periferica

Riceve indirizzi dal sistema (p.es. per identificare la periferica)

Invia al sistema segnali di richiesta di interruzione



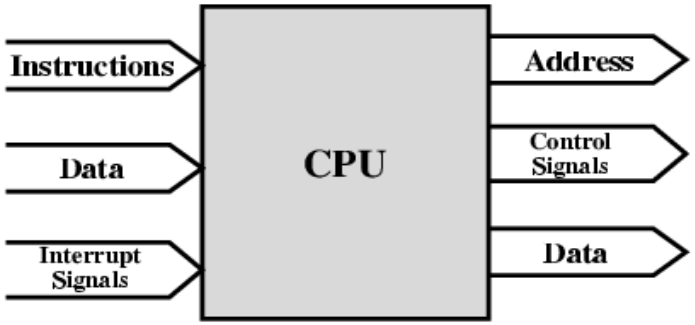
Connessione della CPU

Legge istruzioni e dati

Scrive dati

Invia segnali di controllo alle altre unità

Riceve (e gestisce) richieste di interruzione



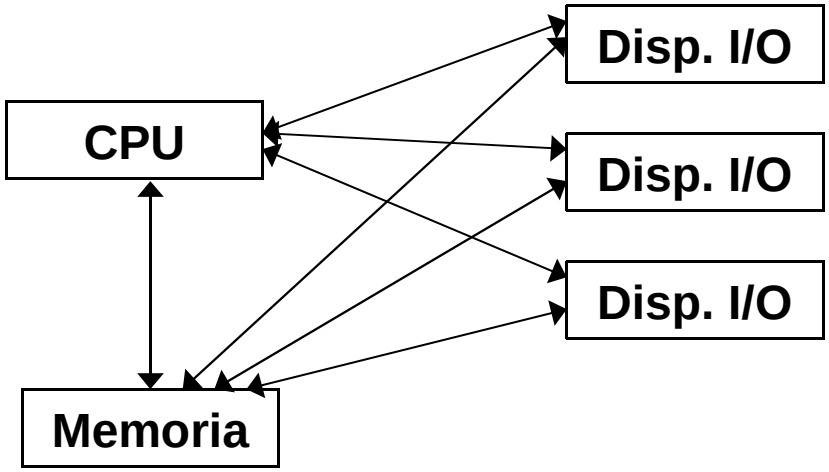
Come si realizza la connessione ?

Una prima soluzione potrebbe essere quella di collegare i vari componenti con connessioni punto-punto, ma ciò comporta:

complessità dell'hardware

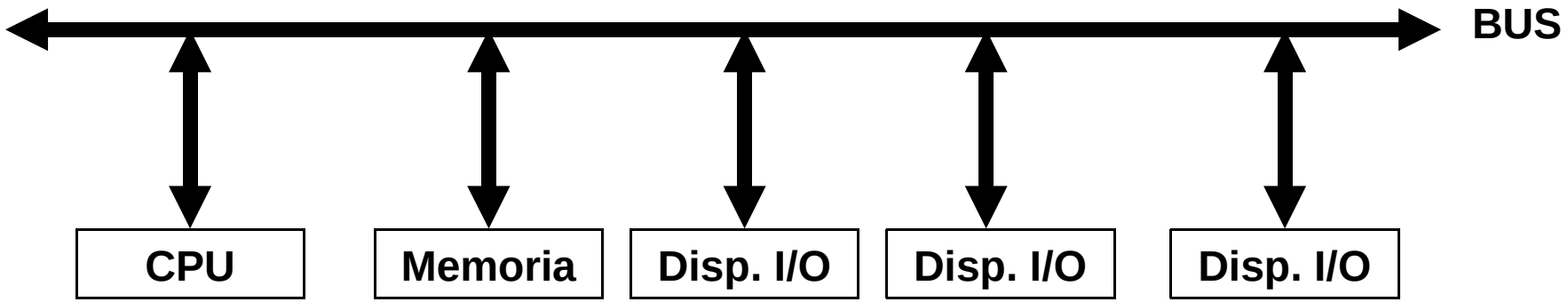
costo realizzativo

struttura estremamente rigida



Una soluzione efficiente: il bus

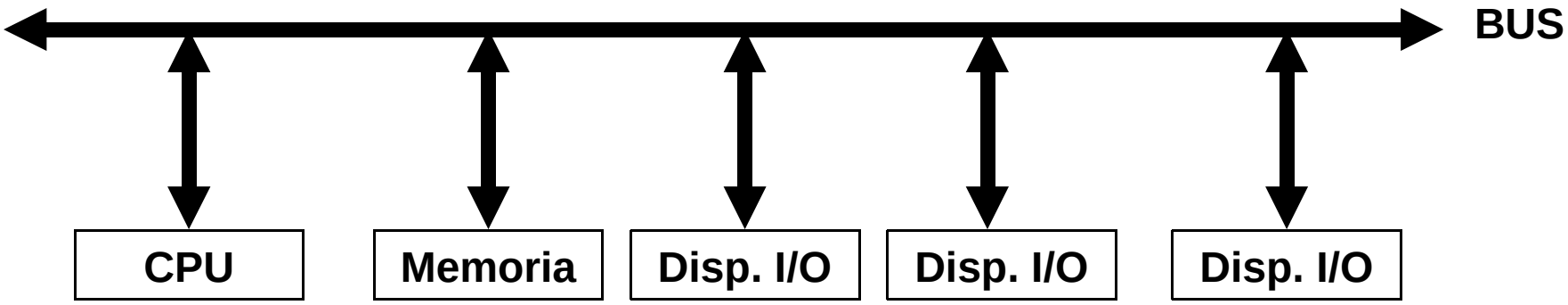
Alternativa: utilizzare un canale di comunicazione condiviso dai vari componenti



Vantaggi:

- **Basso costo:**
  lo stesso canale di comunicazione è utilizzato in modi diversi per diverse esigenze
- **Versatilità:**
  facile aggiungere o rimuovere dispositivi e periferiche
- **Semplicità di progetto:**
  è possibile operare una decomposizione sul progetto dell'architettura

Svantaggi



- **velocità limitata:**
  limita il throughput dell'I/O e della memoria
- **La velocità sul bus è limitata da fattori fisici**:
  - la lunghezza del bus
  - il numero di dispositivi presenti
  - la necessità di supportare un insieme di dispositivi con latenze e velocità di trasferimento che variano notevolmente

Tipi di bus

- **Bus CPU-Memoria** (specifici)

corti ed ad alta velocità

progettati solo per il memory system (massimizzano la banda passante tra memoria e processore)

sono connessi direttamente al processore

ottimizzati per il trasferimento a blocchi della cache

- **I/O Bus** (industry standard)

di solito lunghi e più lenti

devono adattarsi a un vasto insieme di dispositivi I/O

si connettono al processor-memory bus o al backplane bus

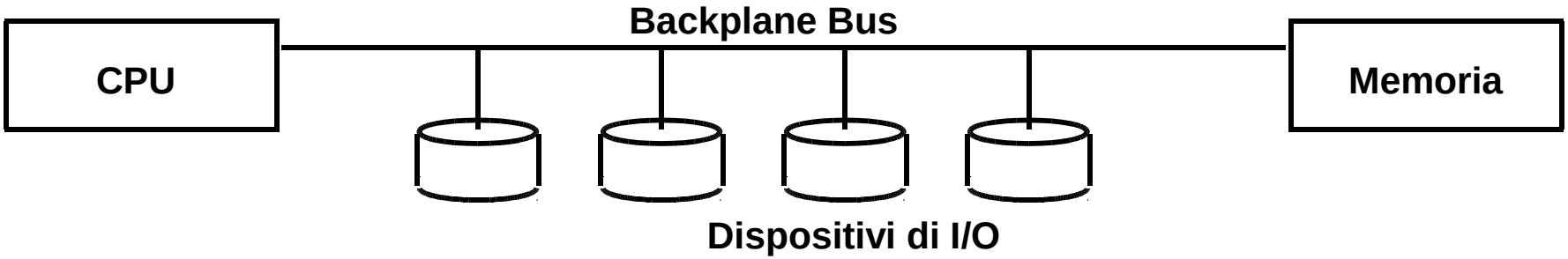
- **Backplane Bus** (standard o proprietario)

backplane: una struttura di interconnessione nello chassis

coesistono processori, memorie, e dispositivi di I/O

vantaggio di costo: un bus per tutti i componenti

Sistema a bus singolo



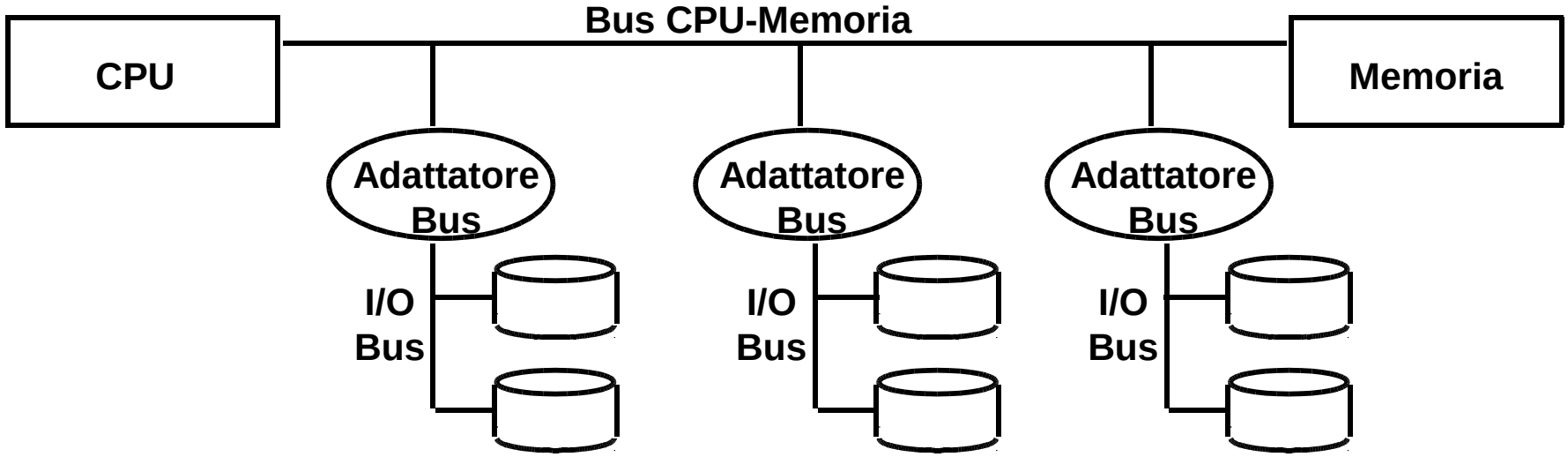
Un solo bus (il backplane bus) è usato per la comunicazione tra processore, memoria e dispositivi di I/O

Vantaggi: semplice ed economico

Svantaggi: lento, il bus può diventare il principale collo di bottiglia del sistema

Esempio: IBM PC - AT

Sistema a due bus



I bus di I/O sono connessi al bus CPU-memoria tramite “adattatori” :

Bus CPU-memoria: usato principalmente per il traffico di memoria

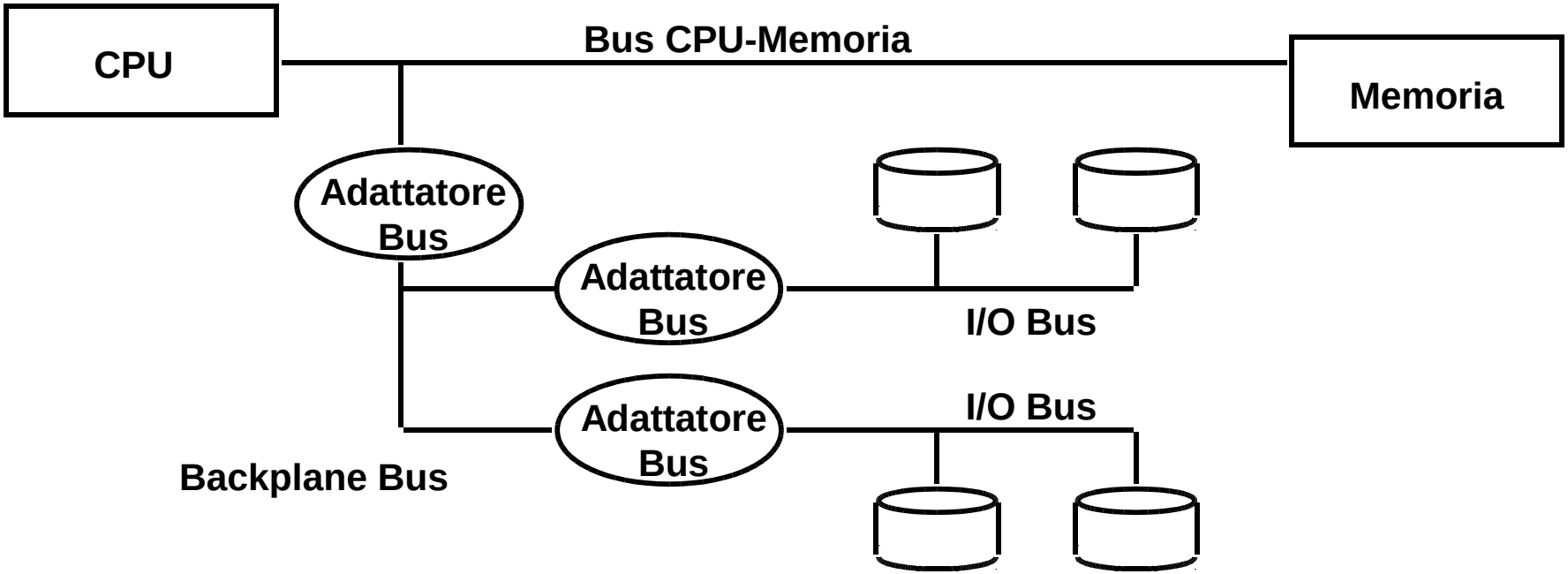
Bus I/O: forniscono slot di espansione per i dispositivi di I/O

Esempio: Apple Macintosh-II

NuBus: Processore, memoria, e pochi, ben selezionati dispositivi

SCSI Bus: il resto dei dispositivi di I/O

Sistema a tre bus



Un backplane bus è connesso al bus CPU-memoria

il bus CPU-memoria è usato per il traffico di memoria

i bus di I/O sono connessi al backplane bus

Vantaggio: il bus del processore è molto più scarico (c'è un'unica connessione)

Organizzazione di un bus

- Linee di Controllo:

trasportano segnali di richiesta e di acknowledgment

indicano che tipo di transazione si sta svolgendo

- Linee Indirizzi:

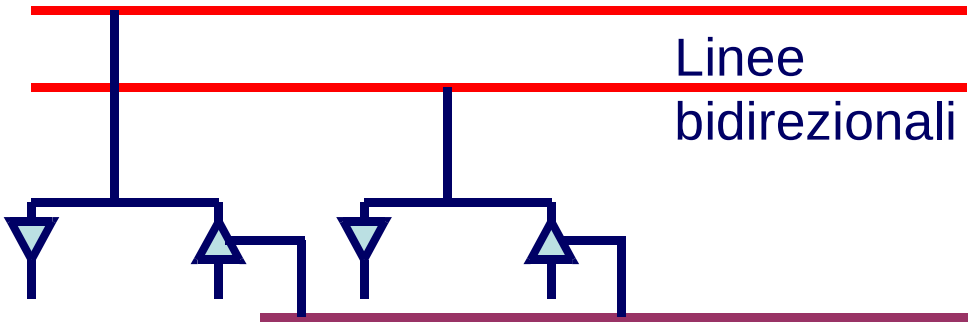
trasportano gli indirizzi dei dati da trasferire

- Linee Dati:

trasportano i dati da trasferire (anche comandi complessi per alcune periferiche)

Collegamento al bus

- Il collegamento delle periferiche al bus avviene di solito tramite un dispositivo intermedio (*driver* del bus) che:
  - fornisce potenza sufficiente per pilotare le linee del bus
  - isola l'uscita del dispositivo quando questa è inattiva
- Sugli ingressi dei dispositivi si pone un *latch*, che realizza il buffering dei dati.
- Il collegamento a linee bidirezionali del bus avviene tramite un *transceiver*



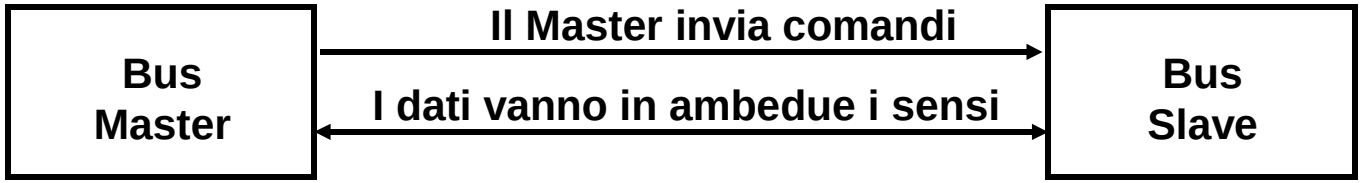
Organizzazione della comunicazione sul bus

La comunicazione sul bus non è paritetica.

Bus Master: ha la capacità di controllare il bus, inizia la transazione

Bus Slave: modulo attivato in seguito alla transazione

Un dispositivo sul bus può fare da solo master (CPU), da solo slave (memoria) o da master/slave (DMA controller)



La comunicazione tra master e slave avviene in base ad una sequenza precisa di azioni realizzate dai partner (transazione).

Il **Protocollo di Comunicazione del Bus** precisa la sequenza di eventi e le specifiche di tempificazione con cui avviene la transazione.

Transazione di bus

- Una transazione di bus include due parti:
  - invio del comando (e dell'indirizzo) – request
  - trasferimento dei dati – action
- Il master inizia la transazione:
  - inviando il comando (e l'indirizzo)
- Lo slave (identificato dall'indirizzo) risponde:
  - inviando i dati al master se richiesti
  - ricevendo i dati dal Master se richiesto
- L'operazione può essere realizzata con due diversi tipi di tempificazione:
  - tempificazione sincrona
  - tempificazione asincrona

Bus sincrono

- Le operazioni sul bus sono tempificate da un segnale di sincronizzazione (clock)
- Gli istanti in cui sono validi i segnali sulle linee sono fissati dal clock (tipicamente sul fronte di salita o di discesa)
- In relazione al clock, si definiscono anche la durata dei segnali e gli intervalli temporali di separazione dei segnali (tipicamente in termini di periodi o semiperiodi del clock)

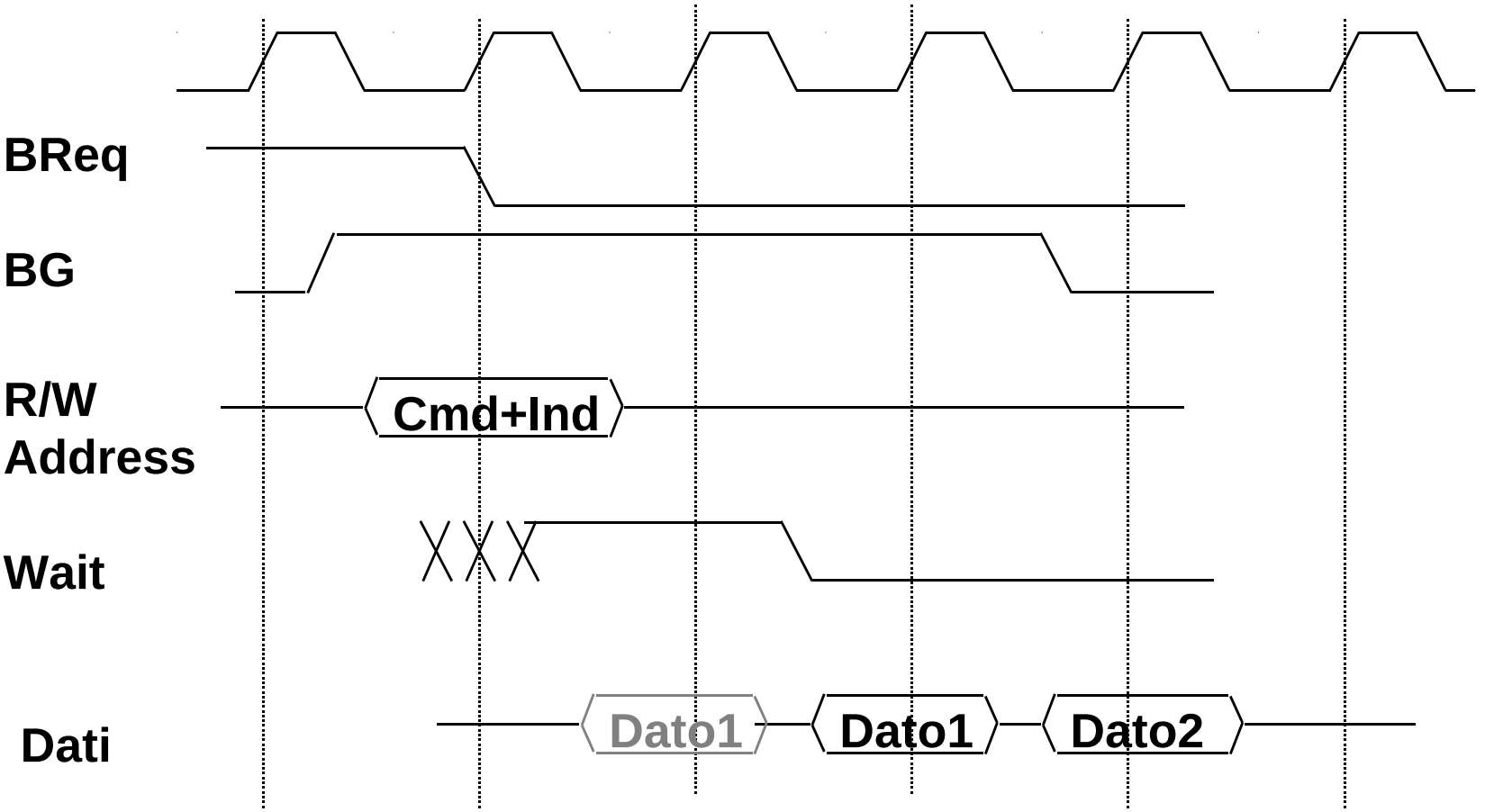
Vantaggi:

semplicità di progettazione del sistema (circuiti sincroni più facili da progettare degli asincroni)

Svantaggi:

la velocità del bus deve adeguarsi ai dispositivi più lenti

Tempificazione di un bus sincrono



Bus asincrono

Nei bus asincroni manca un segnale di clock principale.

Le operazioni avvengono sulla base di un colloquio (handshake) tra master e slave basato su apposite linee di sincronizzazione (DATA READY, DATA REQUEST, ACK, ecc.)

Il tempo impiegato dalle singole operazioni è legato esclusivamente alle velocità esibite dai dispositivi coinvolti.

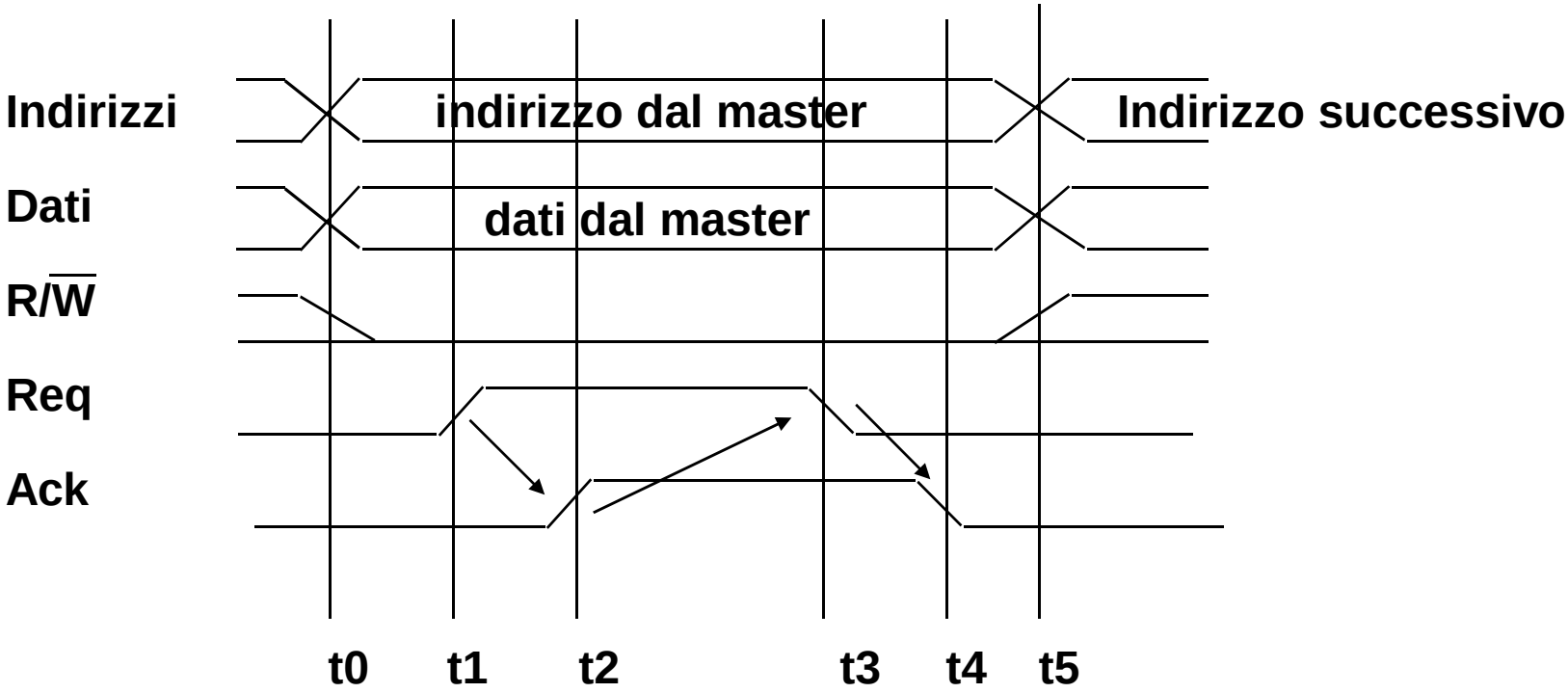
Vantaggi:

possibilità di connettere periferiche con velocità diverse

Svantaggi:

al tempo necessario per una transazione bisogna aggiungere il tempo dell'handshake

Tempificazione asincrona: scrittura



t0 : Il master predispone indirizzo, direzione del trasferimento, dati.
Attende un certo intervallo di tempo (necessario perché gli slaves leggano l'indirizzo)

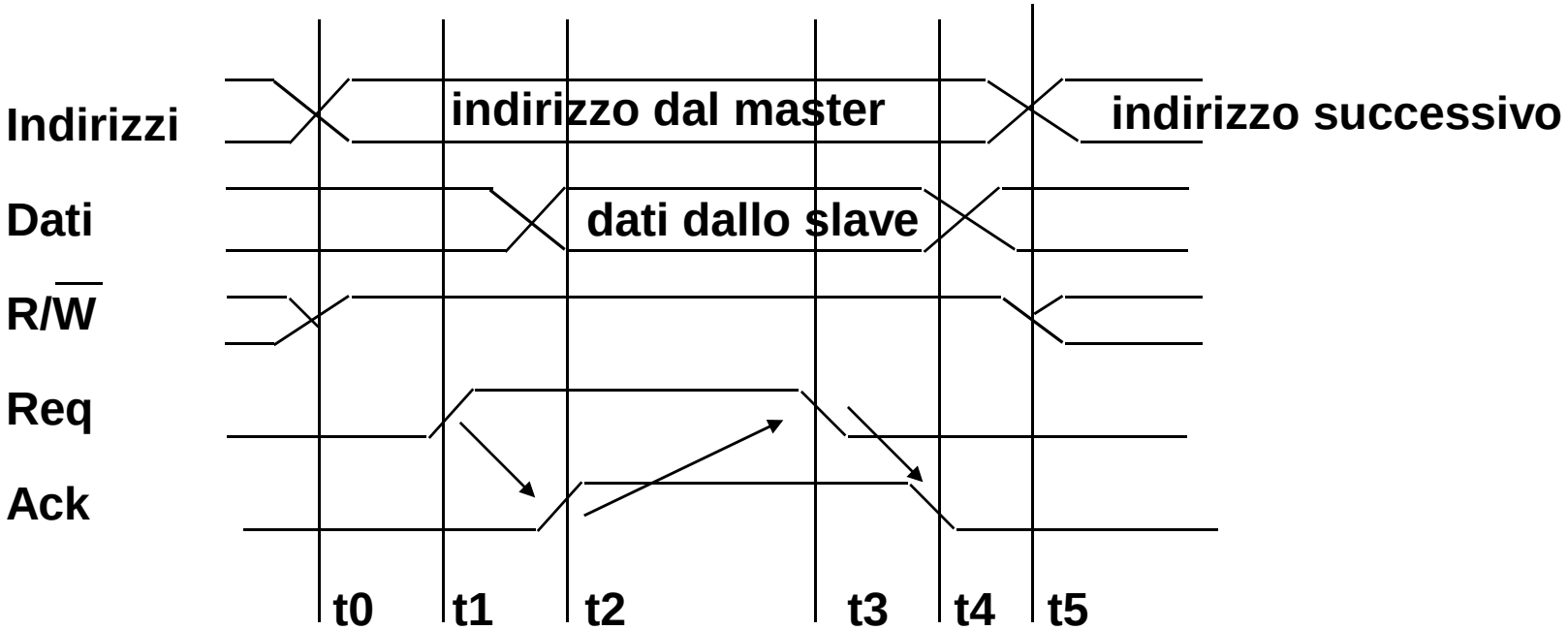
t1: Il master asserisce la linea di richiesta (REQ)

t2: Lo slave asserisce ack (dato ricevuto)

t3: Il master abbassa la linea di richiesta

t4: Lo slave abbassa l'ack

Tempificazione asincrona: lettura



t0 : Il master predispone indirizzo e direzione del trasferimento
Attende un certo intervallo di tempo (necessario perché gli slaves leggano l'indirizzo)

t1: Il master asserisce la linea di richiesta (REQ)

t2: Lo slave asserisce ack (dato disponibile)

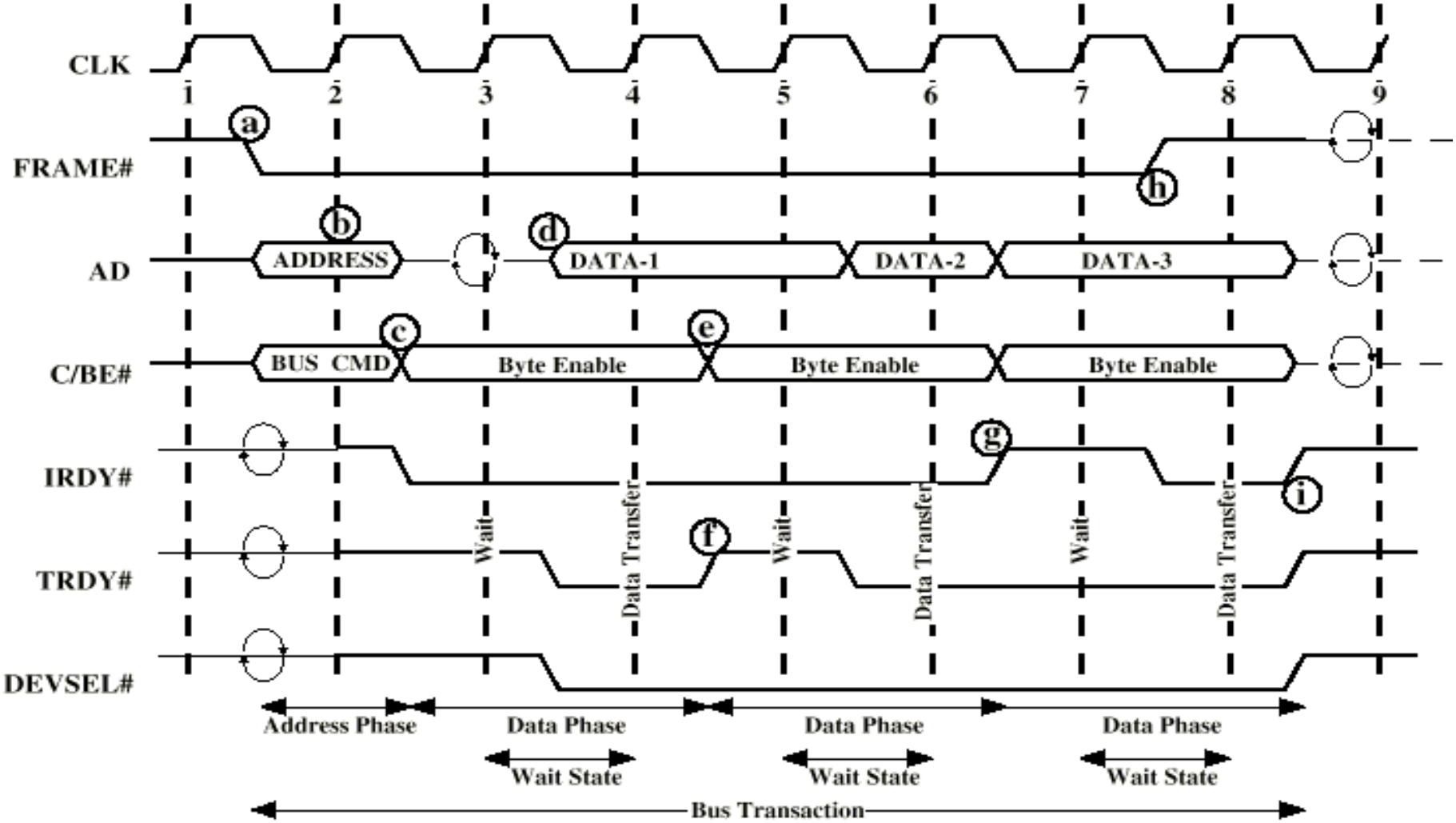
t3: Il master abbassa la linea di richiesta (dato ricevuto)

t4: Lo slave abbassa l'ack

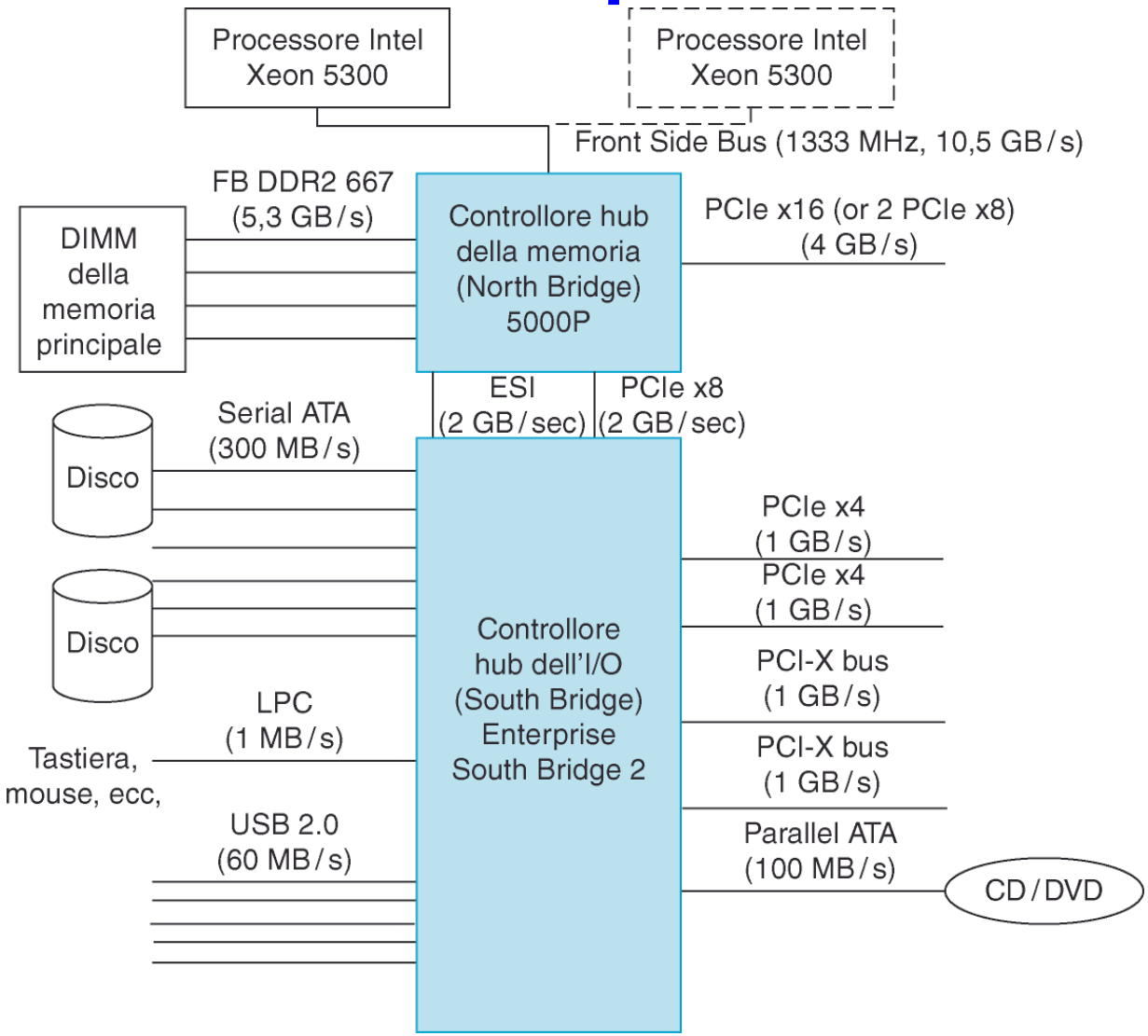
Esempi di standard di bus

Caratteristica	Firewire (1394)	USB 2.0	PCI Express	Serial ATA	Serial Attached SCSI
Utilizzo previsto	Esterno	Esterno	Interno	Interno	Esterno
Numero dispositivi per canale	63	127	1	1	4
Larghezza base dei dati (numero di segnali)	4	2	2 per linea	4	4
Larghezza di banda di picco teorica	50 MB/s (Firewire 400) o 100 MB/s (Firewire 800)	0,2 MB/s (low speed), 1,5 MB/s (full speed), o 60 MB/s (high speed)	250 MB/s per linea (1x); le schede PCIe sono disponibili in versione 1x, 2x, 4x, 8x, 16x o 32x	300 MB/s	300 MB/s
Collegamento a caldo	Sì	Sì	Dipende dalle dimensioni	Sì	Sì
Lunghezza massima del bus (piste in rame)	4,5 metri	5 metri	0,5 metri	1 metro	8 metri
Nome dello standard	IEEE 1394, 1394b	Forum degli implementatori USB	SIG PCI	SATA-IO	Comitato T10

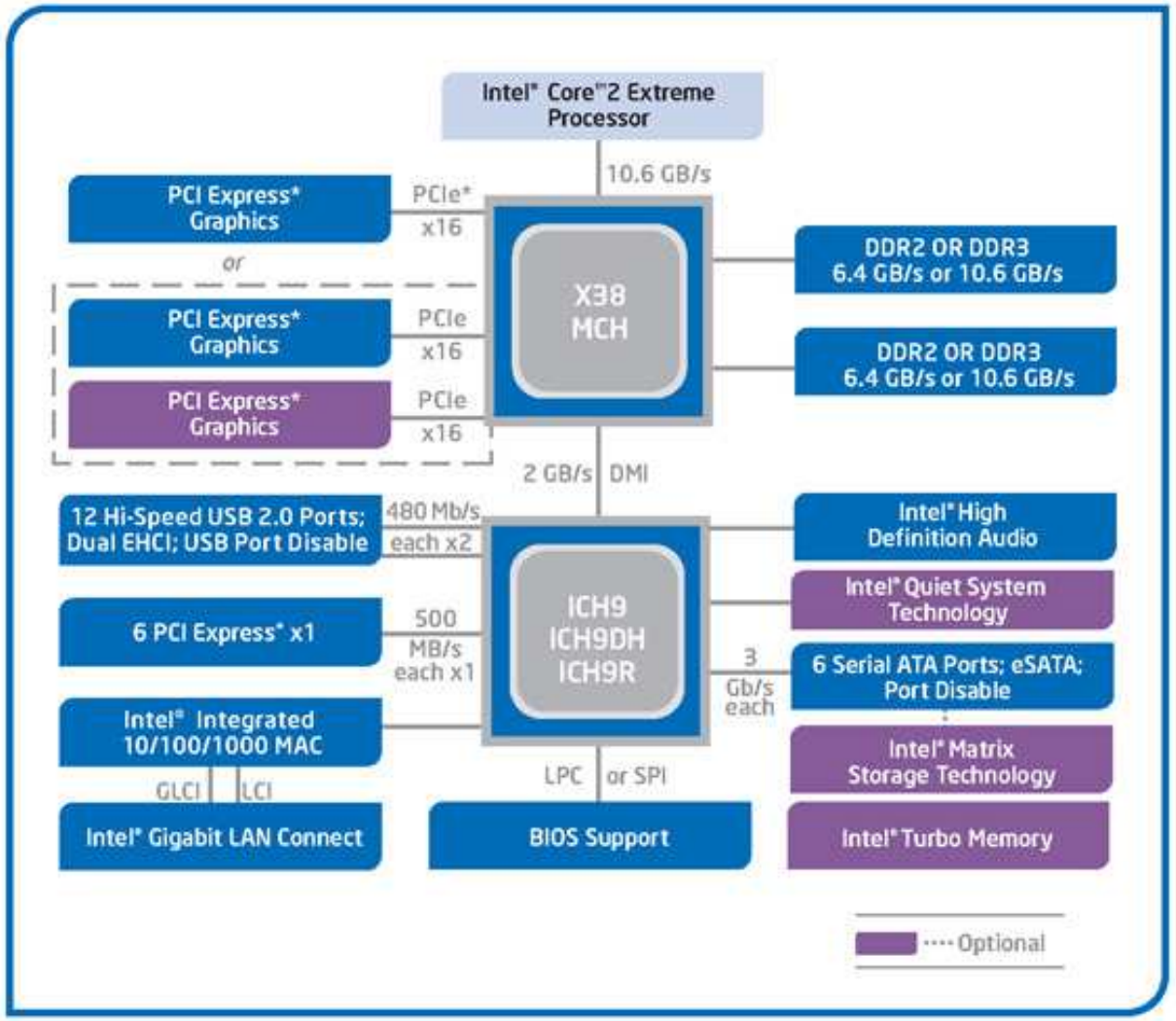
Esempio di tempificazione: PCI read timing



Interconnessioni nei processori Intel

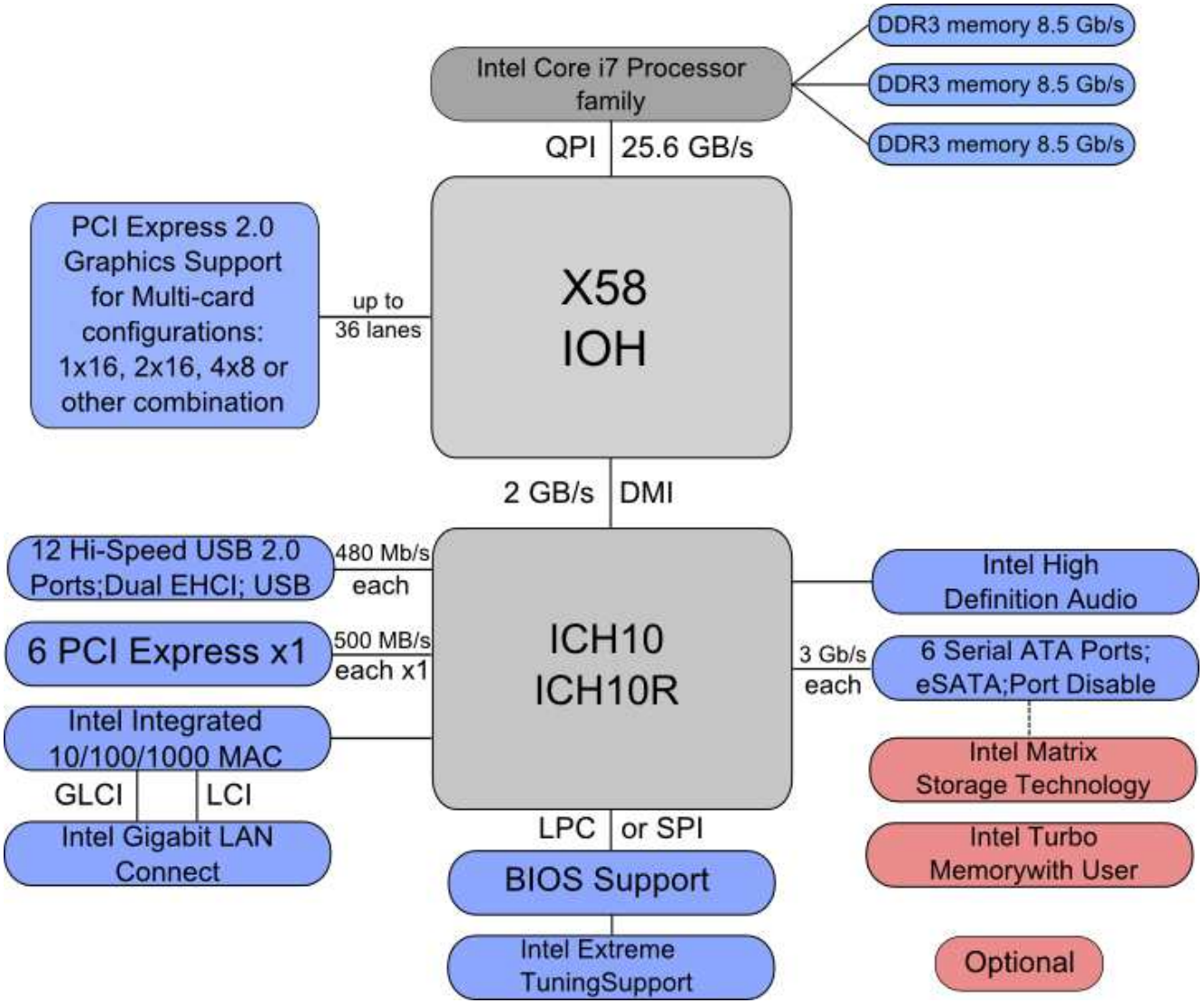


Intel X38 Express Chipset

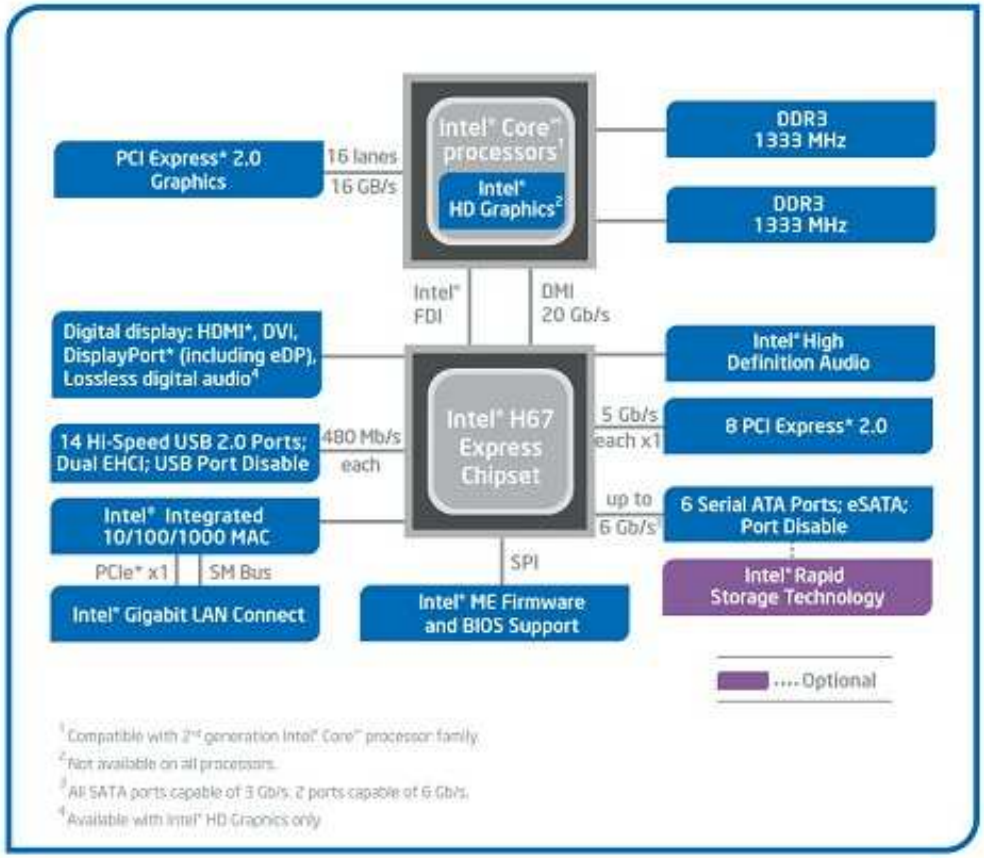


Intel® X38 Express Chipset Block Diagram

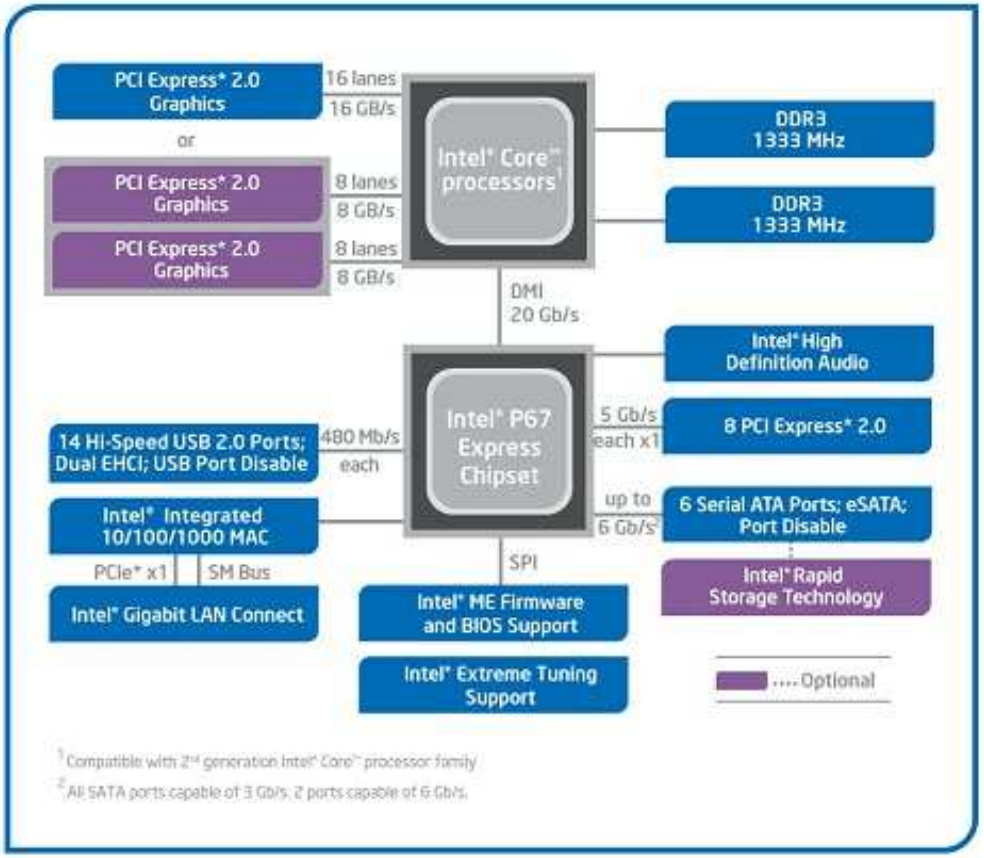
Intel X58


DDR3 memory 8.5 Gb/s
DDR3 memory 8.5 Gb/s
DDR3 memory 8.5 Gb/s
Intel Core i7 Processor family
QPI
25.6 GB/s
PCI Express 2.0 Graphics Support for Multi-card configurations: 1x16, 2x16, 4x8 or other combination
up to 36 lanes
X58 IOH
2 GB/s
DMI
12 Hi-Speed USB 2.0 Ports;Dual EHCI; USB
480 Mb/s each
6 PCI Express x1
500 MB/s each x1
Intel Integrated 10/100/1000 MAC
GLCI
LCI
Intel Gigabit LAN Connect
ICH10 ICH10R
Intel High Definition Audio
3 Gb/s each
6 Serial ATA Ports; eSATA;Port Disable
Intel Matrix Storage Technology
Intel Turbo Memorywith User
LPC
or SPI
BIOS Support
Intel Extreme TuningSupport
Optional

Intel Chipset Serie 6



Intel® H67 Express Chipset Platform Block Diagram



Intel® P67 Express Chipset Platform Block Diagram

Fasi di un'operazione di I/O

- La CPU verifica lo stato dell'interfaccia di I/O
- L'interfaccia di I/O fornisce lo stato
- Se lo stato è ready, la CPU richiede un trasferimento di dati
- L'interfaccia di I/O acquisisce i dati dalla periferica
- L'interfaccia di I/O trasferisce i dati alla CPU

- **ACHTUNG**: descrizione generica – possibili molte variazioni sul tema !!

I/O Addressing

Ogni interfaccia viene selezionata dal processore tramite un indirizzo univoco.

Questo può appartenere:

- allo spazio di indirizzamento della CPU ⇨ **Memory mapped I/O**
  - operazioni di I/O realizzate tramite istruzioni di MOVE
  - numero di I/O ports teoricamente illimitato
  - struttura del bus più semplice
- ad uno spazio di indirizzamento separato ⇨ **Independent I/O**
  - presenza di istruzioni specifiche per l'I/O (IN, OUT)
  - non si sacrificano indirizzi di memoria per l'I/O
  - struttura del bus più complessa

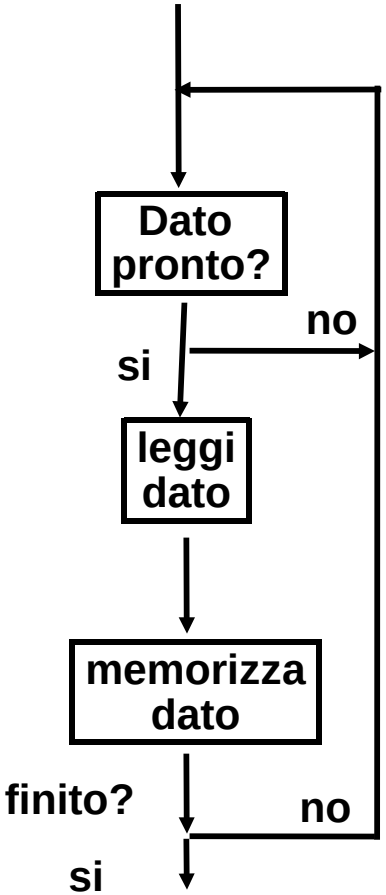
Possibili soluzioni miste

Tempificazione delle operazioni di I/O

- Enorme differenza tra la velocità della CPU e delle periferiche
- Un processore funzionante a 500 MHz può (teoricamente) gestire un traffico di dati ad una velocità di 200000 KB/s, mentre le periferiche più veloci arrivano a 30000 KB/s
- **Input**: la periferica può non essere in grado di inviare dati alla CPU alla stessa velocità con cui il processore riesce a leggerli. Inoltre, la periferica potrebbe interagire con un utente umano, rallentando ulteriormente l'attività
- **Output**: la periferica può non essere in grado di accettare dati dalla CPU alla stessa velocità con cui il processore li produce
- Non è realizzabile una gestione sincrona delle operazioni di I/O.

Gestione delle operazioni di I/O: Polling

- Gestione asincrona dell'operazione
- La CPU inizia, dirige e termina l'operazione di I/O, rimanendo in attesa del completamento (*Programmed I/O*)
- Esempio: Realizzazione della lettura di un dato
  - Inizio: la CPU accede al Control Register dell'interfaccia, scrivendo una word che inizializza l'interfaccia e la predispone per l'operazione richiesta
  - Attesa: la CPU accede in lettura allo Status Register dell'interfaccia, rimanendo in loop finchè il dato non è disponibile (*busy waiting*)
  - Terminazione: la CPU accede in lettura al Data OUT, prelevando il dato ed inizializzando lo Status Register



E’ conveniente il polling ?

- Con la tecnica del polling, la CPU spende del tempo in attesa della periferica:
  - quanto tempo viene perso in ogni interrogazione ?
  - con che frequenza avvengono le interrogazioni ?
- Consideriamo un processore con un clock da 500 MHz; per ogni interrogazione vengono impiegati 400 cicli di clock (chiamata alla procedura di polling, accesso all’interfaccia, ritorno). Valutiamo il carico dovuto al polling per periferiche diverse
  - Mouse: interrogato 30 volte al secondo per non perdere movimenti dell’utente
  - Floppy disk: trasferimento dati in unità da 2 byte, con una velocità di trasferimento di 50 KB/sec. E’ necessario non perdere nessun dato.
  - Hard disk: trasferimento dati in blocchi da 16 byte, con una velocità di trasferimento di 8 MB/sec. Anche qui è necessario non perdere alcun dato.

Un po' di conti … (1/2)

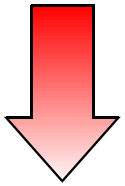
- Mouse:
  - cicli al secondo necessari per l'interrogazione:
    30 * 400 = 12000 cicli/sec
  - overhead sul processore:
    12*103/500*106 = 0.002%
- $\Rightarrow$ Polling con un impatto trascurabile sul processore
- Floppy:
  - interrogazioni/sec = 50 KB/s /2B = 25K poll/sec
  - cicli al secondo necessari per l'interrogazione:
  - 25K * 400 = 107 cicli/sec
  - overhead sul processore :
  - 10*106/500*106 = 2%
- $\Rightarrow$ ancora accettabile se non ci sono molte periferiche

Un po' di conti … (2/2)

- Hard Disk:
  - interrogazioni/sec = 8 MB/s /16B = 500K poll/sec
  - cicli al secondo necessari per l'interrogazione:
  - 500K * 400 = 2*108 cicli/sec
  - overhead sul processore :
  - 200*106/500*106 = 40%
- $\Rightarrow$ inaccettabile

Alternative al polling - Interrupt I/O

- E' inutile che il processore sprechi tempo nell'attesa che la periferica sia pronta.
- Sarebbe sufficiente che la procedura di lettura fosse eseguita solo quando la periferica è pronta.

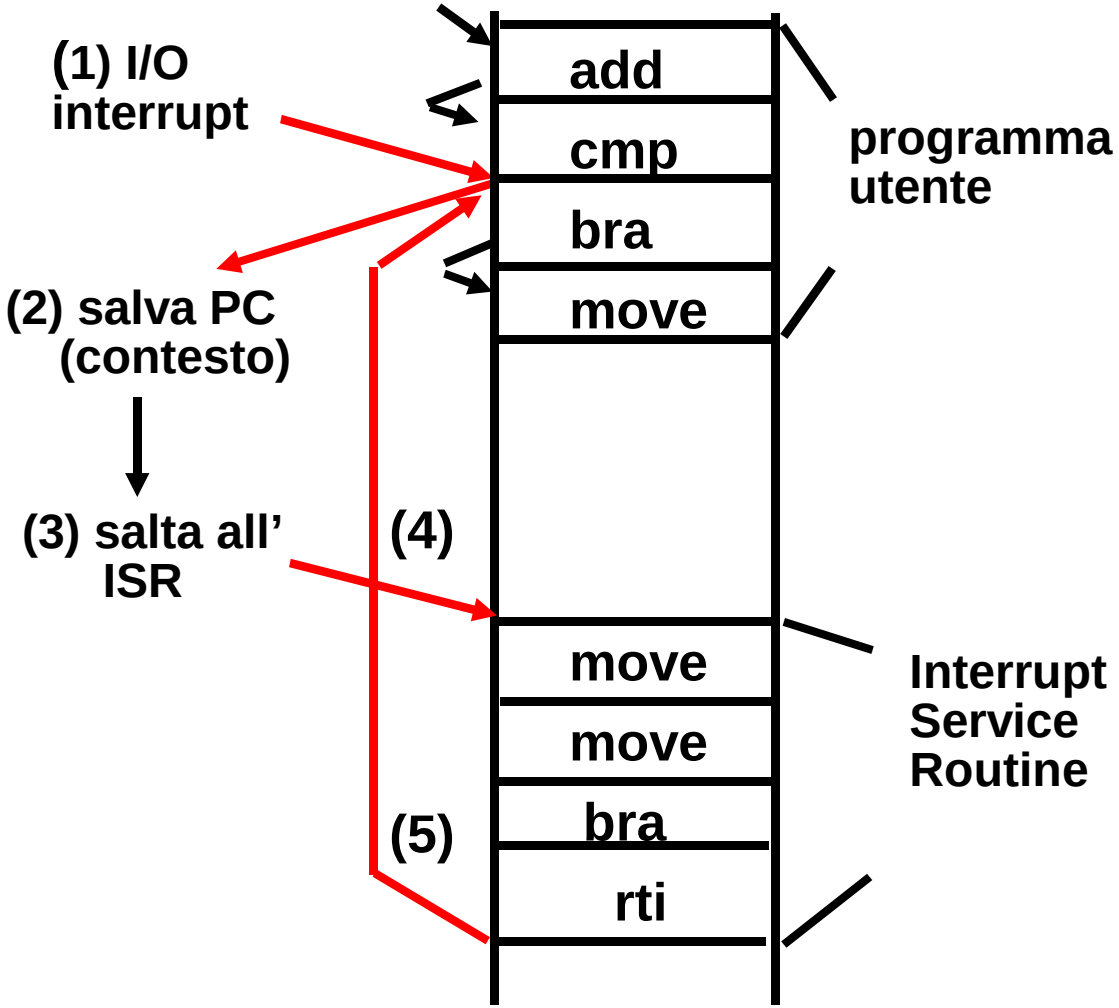


Uso delle interruzioni

- Possibile una gestione asincrona dell'operazione di I/O
- L'interfaccia deve essere in grado di generare un segnale di interrupt
- La CPU non deve attendere il completamento dell'operazione, ma risponde alla richiesta di interruzione

Gestione di un'operazione via interrupt I/O

- La CPU inizia l'operazione, scrivendo il contenuto del C.R. dell'interfaccia.
- Non rimane in attesa del completamento, ma riprende il programma, interrompendolo su richiesta dell'interfaccia



Valutazione dell'Interrupt I/O

- Supponiamo un overhead di 500 cicli di clock per ogni trasferimento, interrupt inclusa. Supponiamo che l'hard disk sia attivo il 5% del tempo.
- Interrupt rate
  - Disk Int/sec = 8 MB/s /16B
    = 500K int/sec
  - Disk Polling cicli/sec = 500K * 500
    = 250,000,000 cicli/sec
  - Overhead sul processore durante i trasferimenti:
    250*106/500*106= 50%
- Disco attivo 5% $\Rightarrow$ 5% * 50% = 2.5% overhead totale

Gestione dell' I/O: questioni aperte

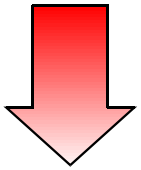
- L'organizzazione e la gestione delle operazioni di I/O ha una complessità molto elevata che non può essere gestita dall'utente normale, il quale
  - Non ha conoscenza del sistema di I/O
  - Non ha le competenze necessarie

- E' necessario che questa gestione avvenga a livello di sistema

- Chi mantiene traccia dello stato di tutti i device, gestisce gli errori, sa come è organizzato il sistema di I/O ?

Il ruolo del Sistema Operativo

- **Il sistema di I/O è condiviso tra molti processi che si contendono l'uso del processore**
  - **possibili violazioni di spazi riservati**
  - **possibile sbilanciamento tra gli accessi alle risorse di I/O**
- **Il controllo delle operazioni di I/O è complesso**
- **Le periferiche impiegano spesso le interruzioni per comunicare i risultati delle operazioni di I/O**



Il Sistema Operativo

→ **garantisce che il programma utente accede solo quella parte del sistema di I/O su cui ha diritto**

→ **stabilisce una politica di accesso alle risorse di I/O che assicuri la distribuzione equa degli accessi e l'efficienza complessiva del sistema**

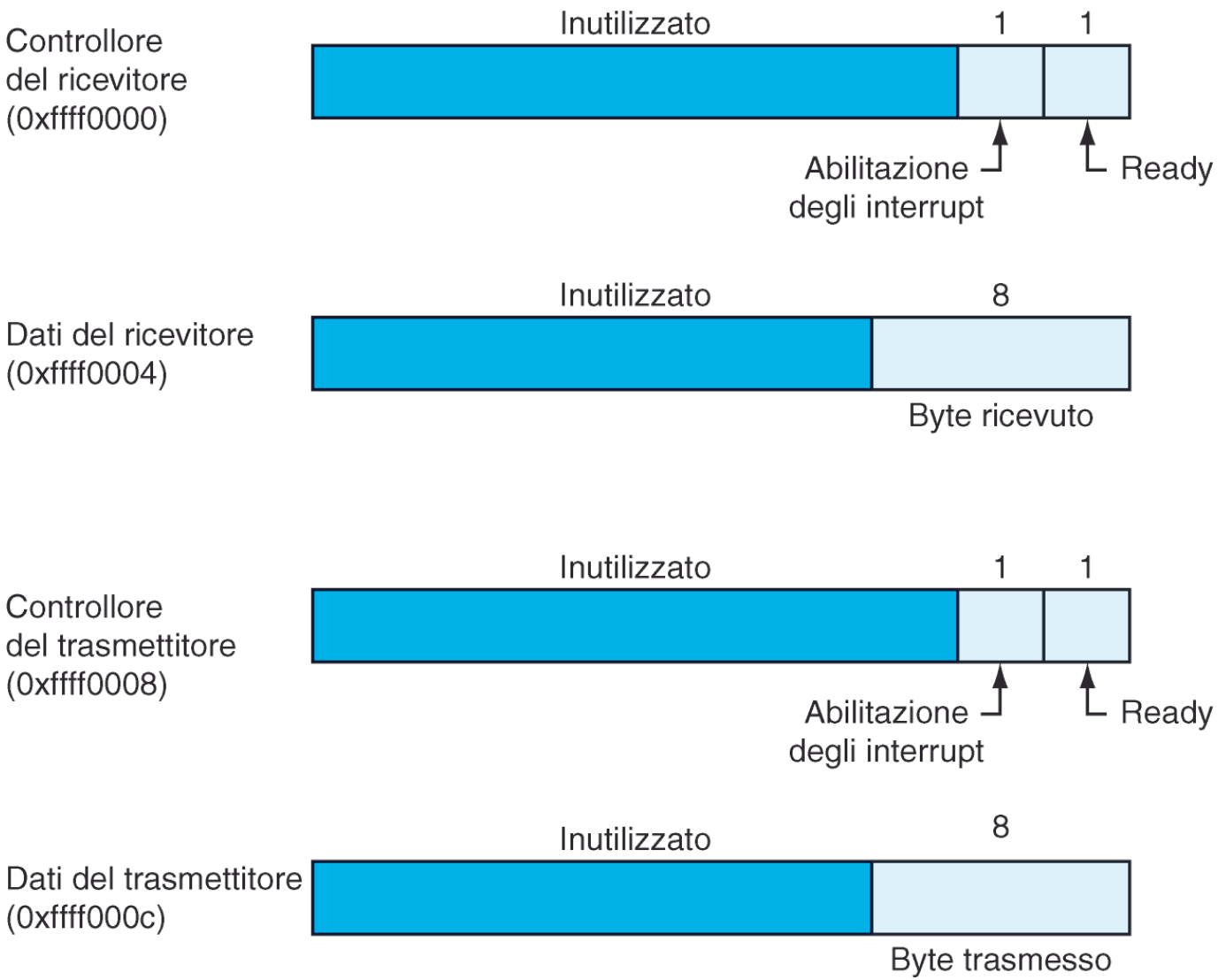
→ **virtualizza l'accesso alle periferiche fornendo procedure che gestiscono le operazioni a basso livello sul device (drivers)**

→ **gestisce le interruzioni generate dalle periferiche**

Esempio: l'I/O in MARS (SPIM)

- MARS simula una console memory-mapped su cui un programma in grado di leggere e scrivere caratteri.
- Il dispositivo è costituito da due unità indipendenti:
  - Un ***ricevitore*** che legge i caratteri digitati sulla tastiera
  - Un ***trasmettitore*** che mostra i caratteri sul terminale.
- Le due unità sono completamente indipendenti: i caratteri digitati sulla tastiera non sono automaticamente inviati al terminale.
- Ogni unità contiene due registri:
  - Un *Control Register*
  - Un *Data Register*
-

Esempio: l'I/O in MARS (SPIM)



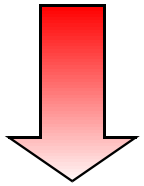
Esempio: l'I/O in MARS (SPIM)

```
                .data
message:        .asciiz "Pippo"

        .text
main:   la $a0,message          # Indirizzo della stringa
        li      $t3,0xFFFF0008  # Carica in t3 l'ind. del contr. reg. del trasm.
        li      $t4,0xFFFF000C  # Carica in t4 l'ind. del data reg. del trasm.
        move    $t0,$a0         # Carica in t0 l'indirizzo della stringa
# Lettura di un carattere
lchar:  lbu     $t1,($t0)       # Carica in t1 il carattere corrente
        beqz    $t1,fine        # Carattere di fine stringa -> termina
# Output di un carattere
busyw:  lw      $t2,0($t3)      # Carica in t2 il cont. del contr. reg.
        andi    $t5,$t2,1       # Controlla il bit ready
        beqz    $t5,busyw       # Torna in ciclo se ==0
        sb      $t1,0($t4)      # Scrive il carattere nel data reg. del trasm.
        addi    $t0,$t0,1       # Incrementa il puntatore alla stringa
        b       lchar           # Continua il ciclo di scrittura dei caratteri
fine:   li $v0,10               # Codice di chiamata al sistema per la
        syscall                 # fine del programma
```

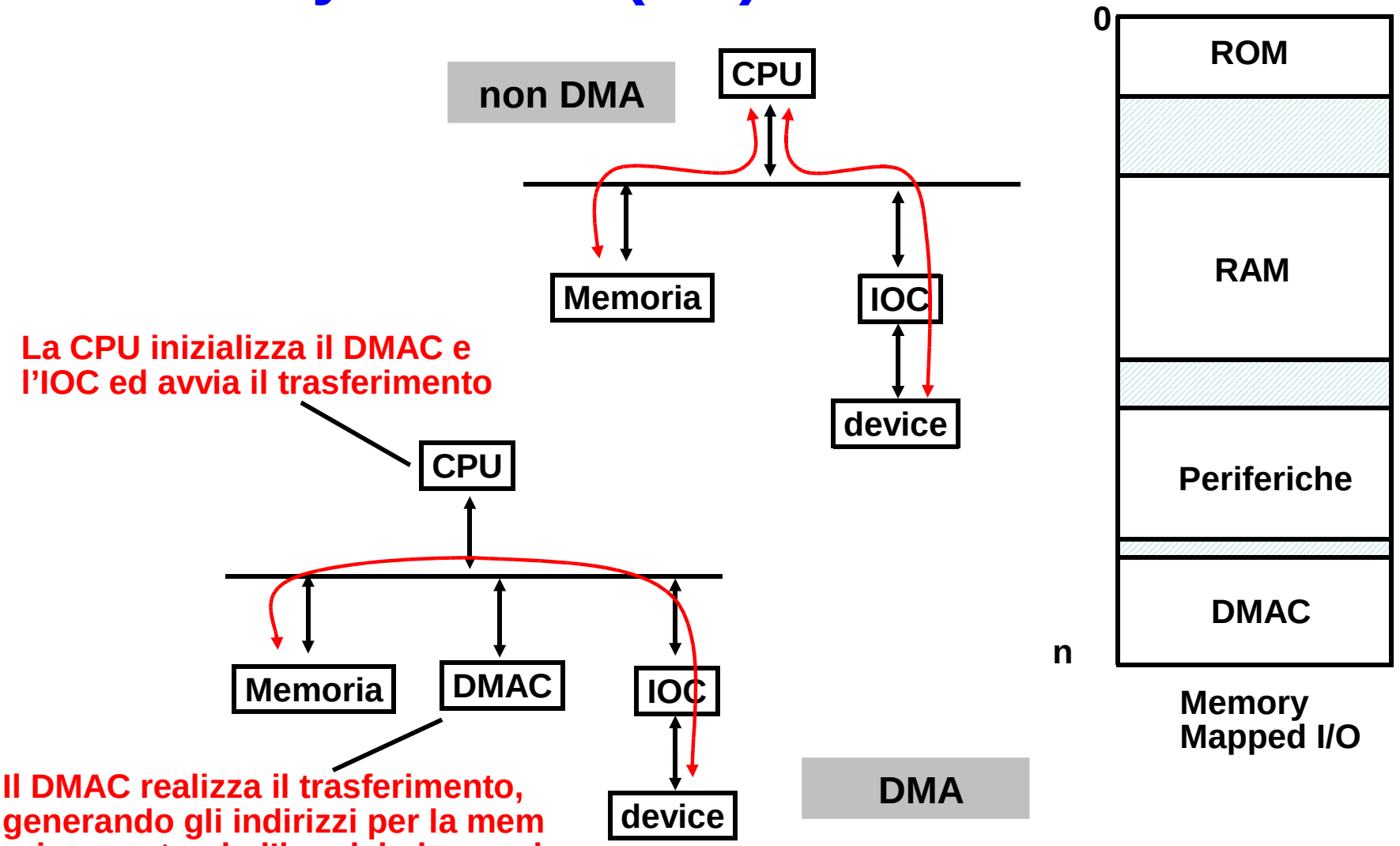
Direct Memory Access (1/3)

- Nel trasferimento di blocchi di dati l'interrupt I/O può risultare comunque pesante
- Durante il trasferimento di un blocco di dati, la CPU è occupata nell'inoltro di dati tra memoria e periferica
- Possibile liberare il processore dalla realizzazione del trasferimento, lasciandogli solo la supervisione dell'operazione (specificare il blocco da trasferire, gestire l'avvio e la chiusura) ?



Direct Memory Access (DMA):
l'operazione è realizzata tramite un dispositivo (DMA controller) capace di trasferire un blocco di dati tra la memoria ed una periferica e di generare un interrupt al termine

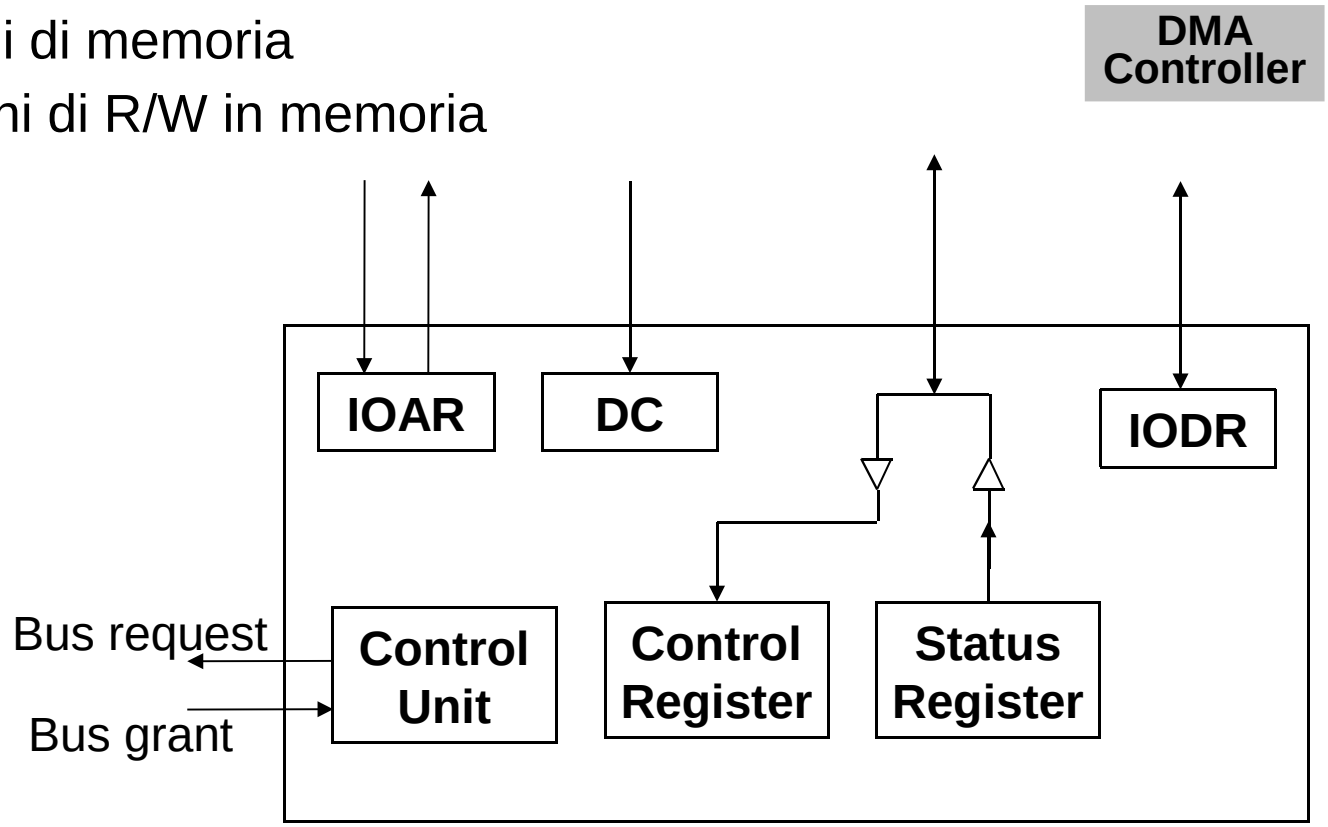
Direct Memory Access (2/3)



Direct Memory Access (3/3)

Per realizzare l'accesso diretto in memoria, il DMA controller deve essere capace di:

- gestire la richiesta per l'uso del BUS
- generare gli indirizzi di memoria
- realizzare operazioni di R/W in memoria



Registri del DMAC:

IOAR **(I/O Addr. Reg.)**
DC **(Data Count)**
IODR **(I/O Data Reg.)**

Sequenza di operazioni di un trasferimento DMA

- 1. La CPU fornisce le informazioni relative al blocco da trasferire ed inizializza il DMAC:
  - 1.1 base address → IOAR
  - 1.2 data count → DC
  - 1.3 controllo → Control Register
- 2. DMAC asserisce BUS REQUEST
- 3. La CPU rilascia il bus e asserisce BUS GRANT
- 4. Acquisito il bus, il DMAC trasferisce i dati da/verso la memoria incrementando IOAR e decrementando DC
- 5. Se DC ≠ 0 e il device esterno non è pronto, il DMAC rilascia il bus che ritorna alla CPU
- 6. Se DC = 0, il controllo ritorna alla CPU. Eventualmente, viene generata un'interruzione

DMA - Modalità di trasferimento

- **Trasferimento a blocco** (block transfer)
  - L'intero blocco viene trasferito mentre il DMA Controller è master del bus
  - Necessario per periferiche (p. e. dischi) in cui il trasferimento non può essere interrotto o rallentato

- **Furto di ciclo** (cycle stealing)
  - Il blocco viene trasferito attraverso una sequenza di trasferimenti di blocchi di lunghezza predefinita, alternandosi con la CPU sul bus

- **Trasferimento trasparente** (trasparent DMA)
  - Il trasferimento ha luogo solo quando la CPU non è master del bus

Valutazione dell'overhead relativo al DMA

- Frequenza processore: 500 MHz
- Setup del DMAC: 1000 cicli
- Servizio dell'interruzione: 500 cicli
- Hard Disk connesso via DMA
- Velocità di trasferimento: 8 MB/s
- Dimensione media del blocco trasferito: 8 KB

Tempo per trasferimento di un blocco: 8 KB / 8 MB/s = 10^{-3} sec

Disco sempre attivo $\Rightarrow$ 10^3 trasf/sec

Overhead del processore $\Rightarrow$ (1000+500) * 10^3 cicli/sec

% Overhead $\Rightarrow$ $1.5*10^6 / 500*10^6$ = 0.3 %

I processori di I/O

programmed I/O

interrupt I/O

DMA

Dispositivi di I/O dalle più ampie capacità liberano la CPU dal peso della gestione delle operazioni di I/O.

Il costo da pagare è la complessità del dispositivo di I/O.

Soluzione estrema: **un processore dedicato all'I/O**

I processori di I/O

I processori di I/O (o *canali*) sono vere e proprie CPU dedicate esclusivamente alla gestione delle operazioni di I/O.

Il codice da eseguire (programma di I/O) è contenuto in una memoria riservata e realizza tutte le fasi delle operazioni di I/O (inizializzazione, trasferimento, chiusura, gestione di errori).

La CPU avvia l'operazione di I/O semplicemente fornendo al processore di I/O l'indirizzo del programma di I/O da eseguire.

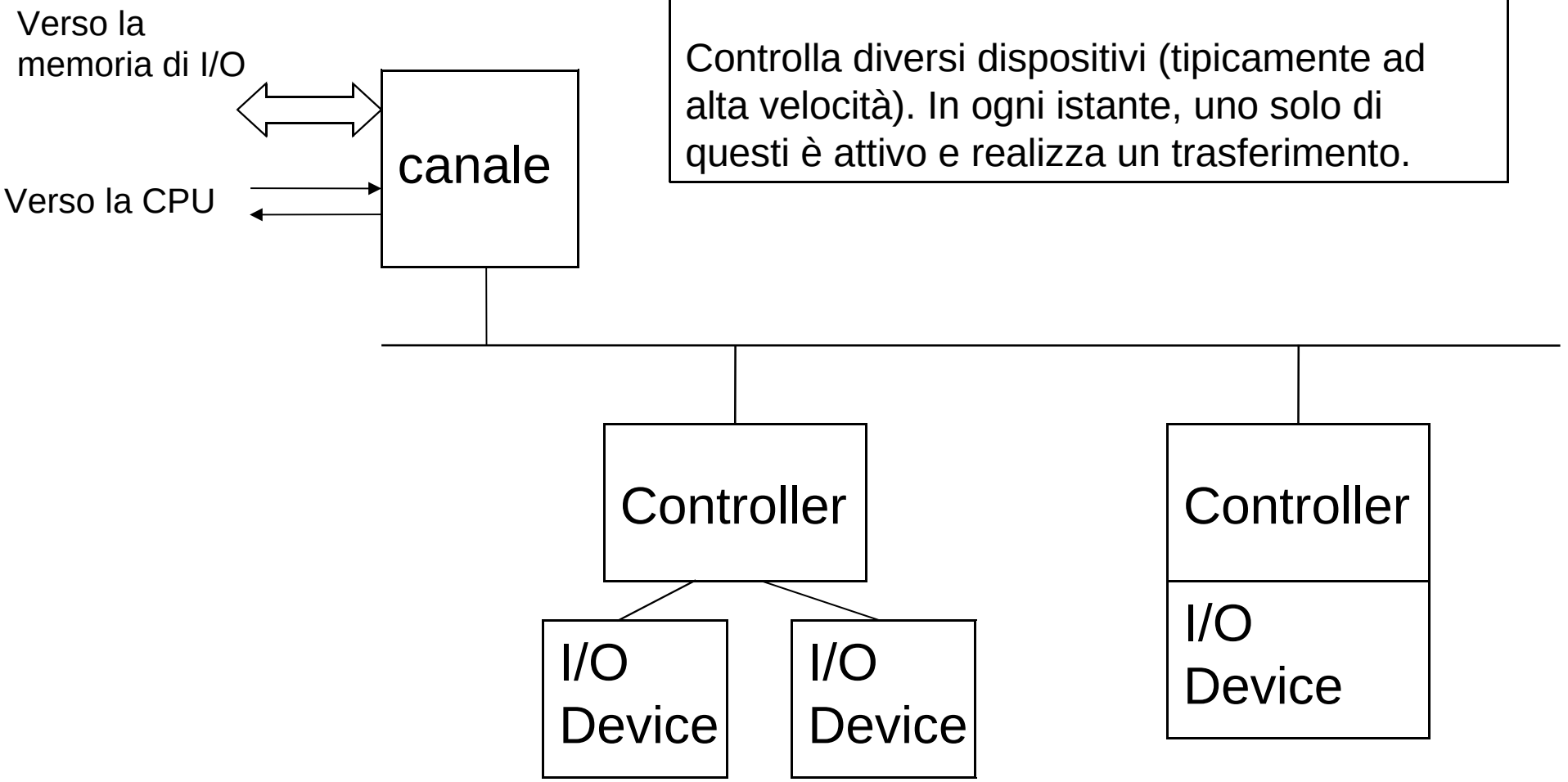
Al termine dell'operazione, il processore di I/O produce un'interruzione per avvertire la CPU.

In questo modo, la CPU non ha più diretto controllo sul sistema di I/O.

Configurazione dei canali (1)

Canale selettore

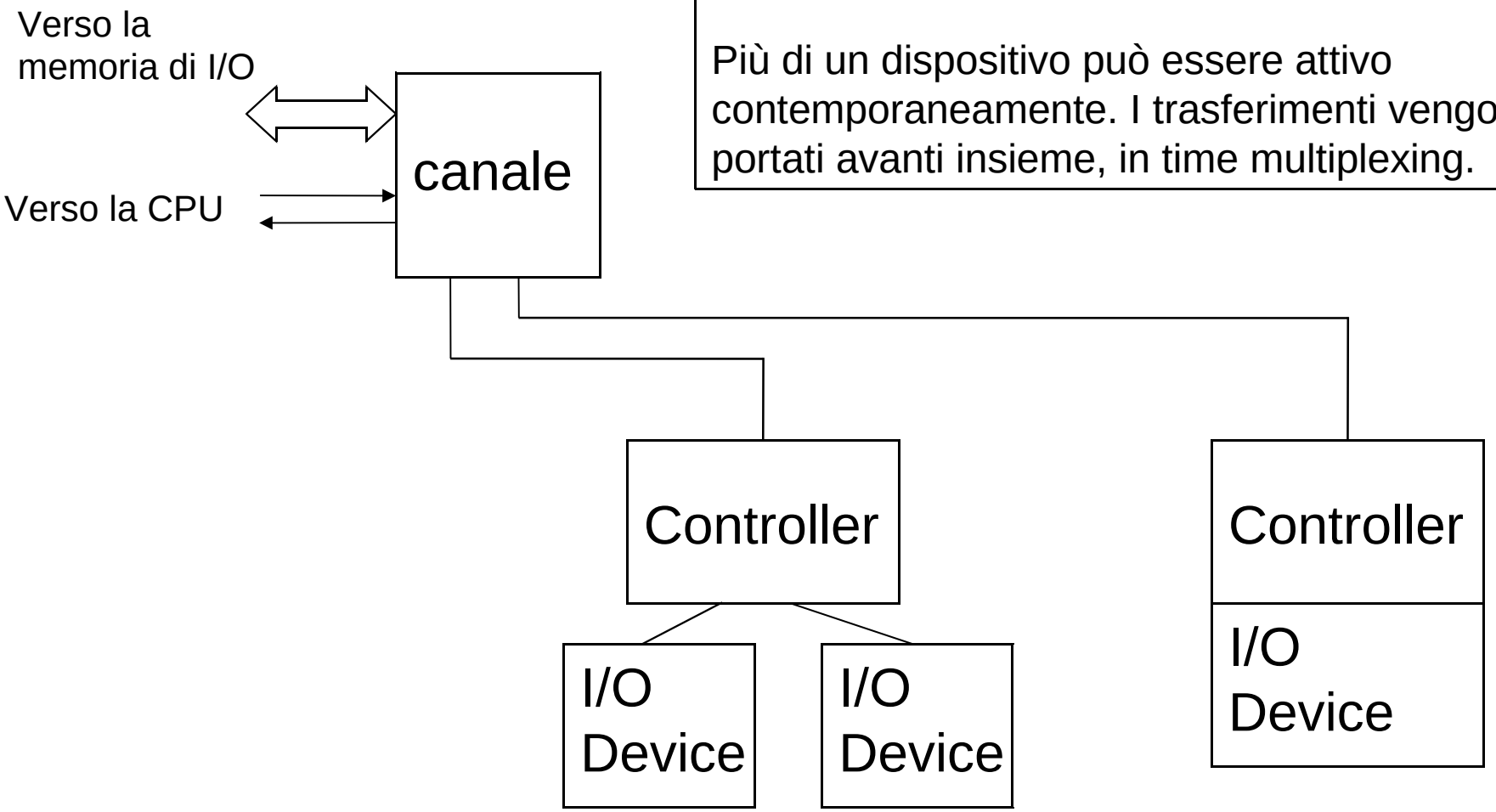
Controlla diversi dispositivi (tipicamente ad alta velocità). In ogni istante, uno solo di questi è attivo e realizza un trasferimento.



Configurazione dei canali (2)

Canale multiplexer

Più di un dispositivo può essere attivo contemporaneamente. I trasferimenti vengono portati avanti insieme, in time multiplexing.



E adesso ??

Prossimi appuntamenti:

* **Lunedì 18 giugno, ore 9.30 (aula da stabilire)**
* **Lunedì 2 luglio, ore 9.30 (aula da stabilire)**
* **Lunedì 23 luglio, ore 9.30 (aula da stabilire)**
* **Giovedì 13 settembre, ore 9.30 (aula da stabilire)**