Metodo di arbitraggio

- Se un sistema ha solo un master (il processore) allora non c'è necessità di arbitraggio: l'accesso al bus è sempre garantito al processore che "pilota" lo slave con cui correntemente effettua la transazione, che ovviamente avviene senza conflitto (secondo il protocollo specificato dal bus)
- Problema di questo approccio: le comunicazioni passano tutte per il processore.

  P.es. lettura di un blocco da disco:

  - il processore (master) legge una parola dal disco (slave)
  - il processore (master) scrive la parola in memoria (slave)
  - e così per ogni parola del blocco

⇒ Introduzione di più dispositivi attivi in grado di agire da master (DMA)

⇒ Possibile conflitto per la risorsa bus: più dispositivi possono in uno stesso momento fare richiesta di accesso al bus

⇒ Necessità di un meccanismo di arbitraggio: risolve i conflitti decidendo chi ha accesso al bus, evitando accessi contemporanei

Parametri per giudicare un meccanismo di arbitraggio

Trade-off

- Priorità:
  possibilità di suddividere i dispositivi in categorie con priorità diversa, accordando il bus ai dispositivi con priorità più alta
- Equità (fairness):
  garantire a tutti i dispositivi la possibilità di accedere al bus
- Latenza:
  tempo dedicato all'arbitraggio più basso possibile (e possibilimente sovrapposto con il tempo di trasferimento sul bus)
- Complessità di realizzazione e costo

Due categorie di meccanismi di arbitraggio

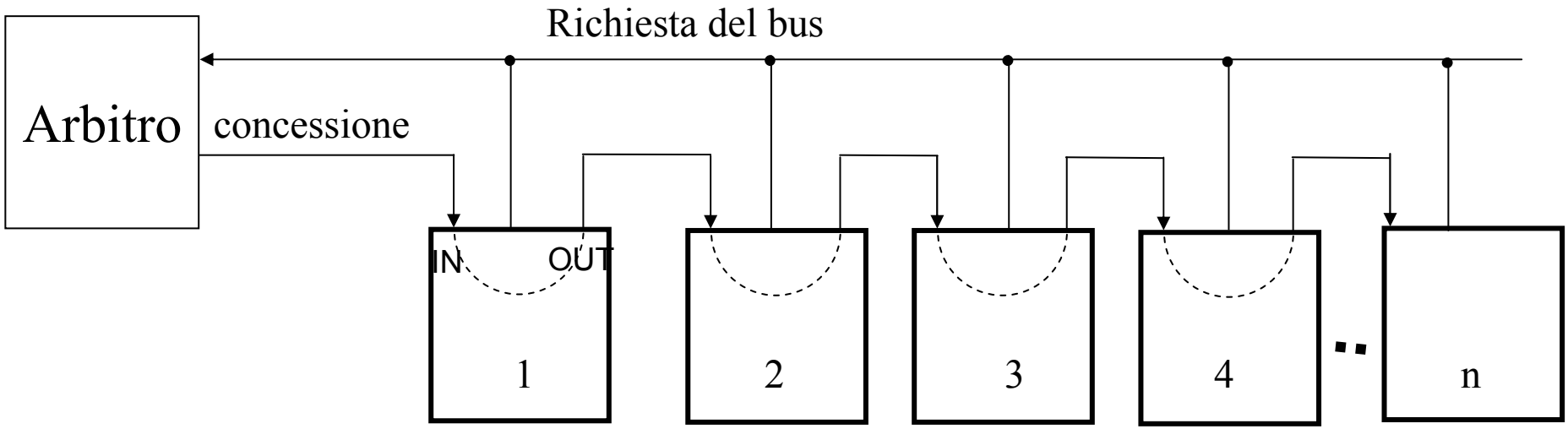
- Centralizzato:
  dispositivo hardware che riceve tutte le richieste e decide chi ha l'accesso
- Distribuito:
  dispositivi collegati al bus, esiste un protocollo per cui si scambiano informazioni e ciascuno decide se è il suo turno

Vediamo:

- Meccanismo centralizzato usando daisy chain
- Meccanismo centralizzato usando livelli + daisy chain
- Meccanismo completamente centralizzato parallelo
- Meccanismo decentralizzato

(e cenni a possibili varianti)

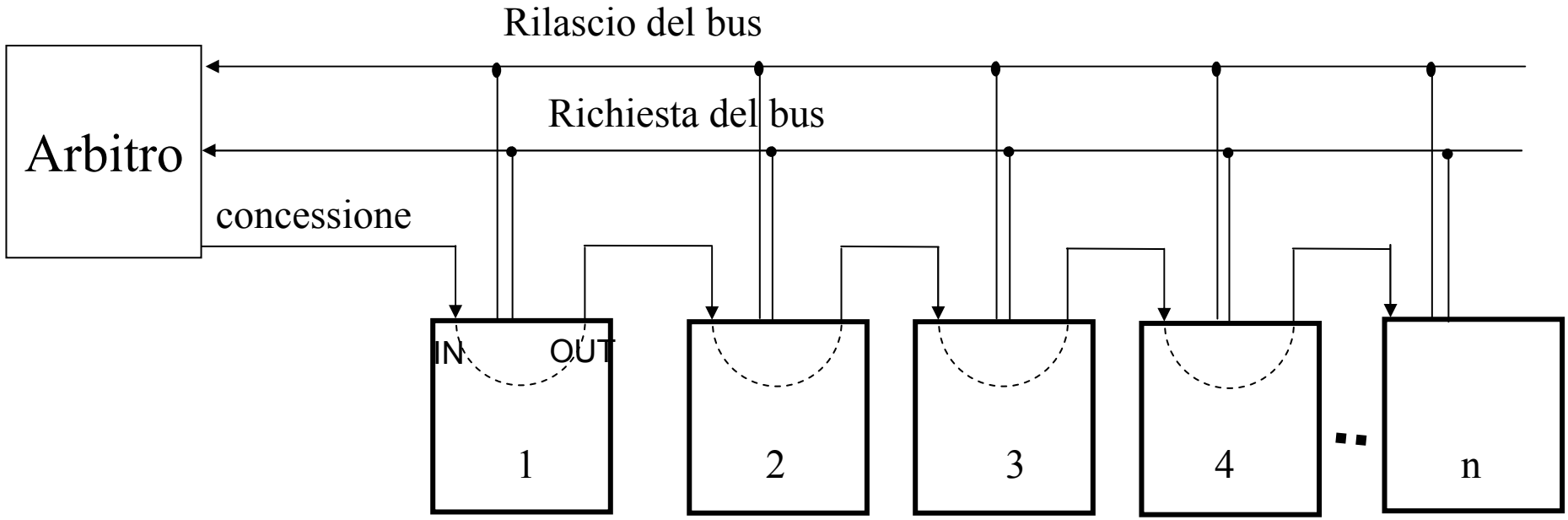
Meccanismo centralizzato usando daisy chain (già visto a proposito di interrupt)



Idea: concessione bloccata da un dispositivo più vicino all'arbitro

Se si vuole garantire che dispositivo a più alta priorità non interrompa l'accesso, occorre rendere dispositivi sensibili ai fronti…

Per effettuare arbitraggio durante il trasferimento per risparmiare tempo, si può usare una linea aggiuntiva che permette di “tenere occupato” il bus



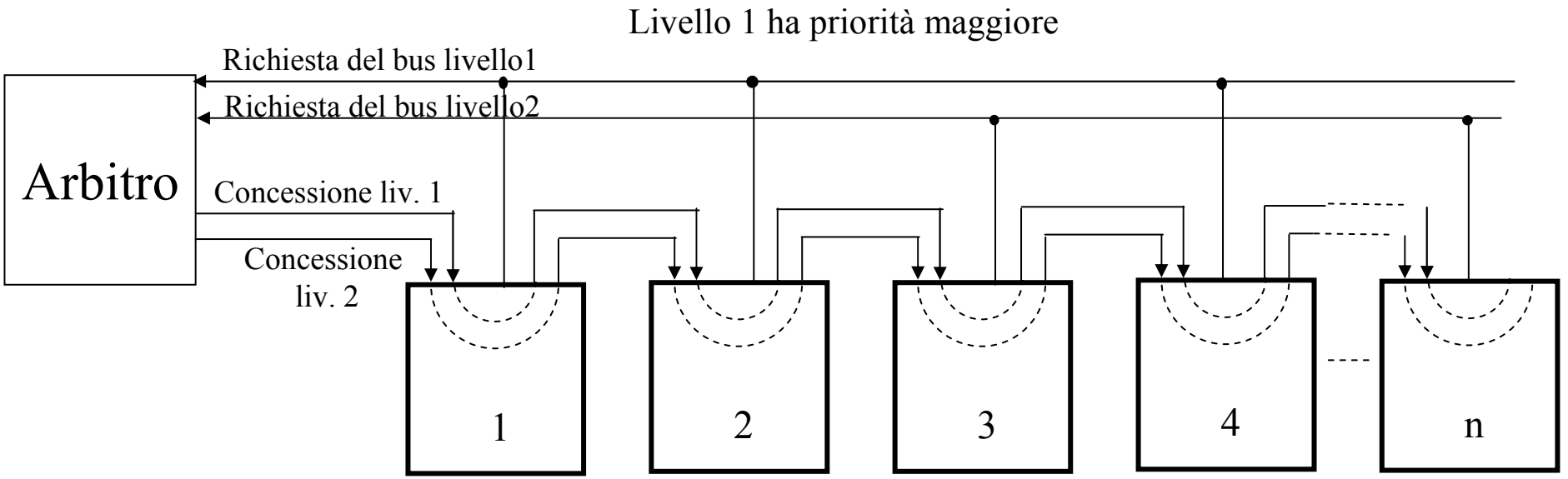
- L’arbitraggio si svolge già mentre un dispositivo ha accesso al bus, ma il dispositivo che ha la concessione del bus (e vi sta accedendo) non rilascia il bus finchè non ha finito.
- Appena rilasciato il bus, l’accesso avviene immediatamente da parte del dispositivo che lo ha conquistato

Valutazione meccanismo con daisy chain

- priorità: legata alla posizione fisica (vince il più vicino)
- equità: i dispositivi a bassa priorità possono non essere mai serviti se dispositivi a più alta priorità continuano a richiedere il bus
- latenza: il tempo dedicato all'arbitraggio (request-grant) dipende dalla posizione del dispositivo (tempo di propagazione segnali)
  $\Rightarrow$ limitazione velocità in funzione della lunghezza
- complessità & costo: semplice da realizzare, di basso costo

Meccanismo centralizzato usando daisy chain + livelli

(già visto a proposito di interrupt)

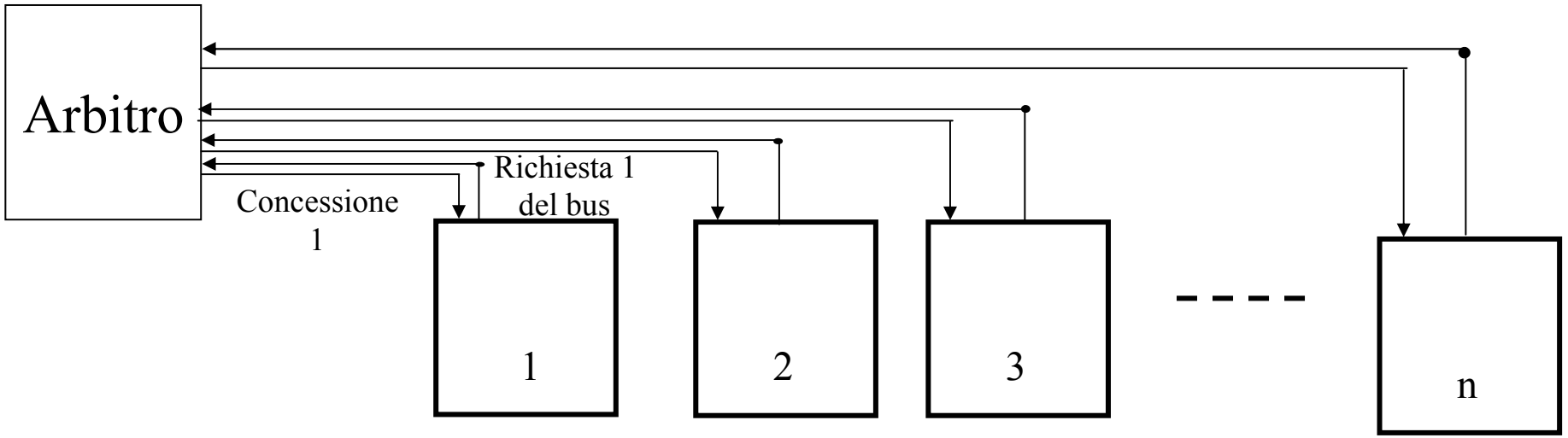


In questo caso la priorità è, nell'ordine: 1, 2, 4, 3, n

NB: una linea di concessione potrebbe attraversare solo i dispositivi collegati alla relativa linea di richiesta, ma è più semplice realizzare il collegamento come in figura [dal punto di vista logico, è come avere linee separate]

Esercizio possibile (banale): dato lo schema ordinare i dispositivi secondo la priorità

Meccanismo centralizzato parallelo



- I dispositivi richiedono il bus, ciascuno con la propria linea
- L'arbitro centralizzato sceglie uno dei richiedenti, concedendogli il bus mediante la relativa linea di concessione

VALUTAZIONE

Vantaggio:
- può utilizzare diversi algoritmi per la concessione del bus (es. priorità, round robin, ecc.) garantendo un buon compromesso tra priorità ed equità

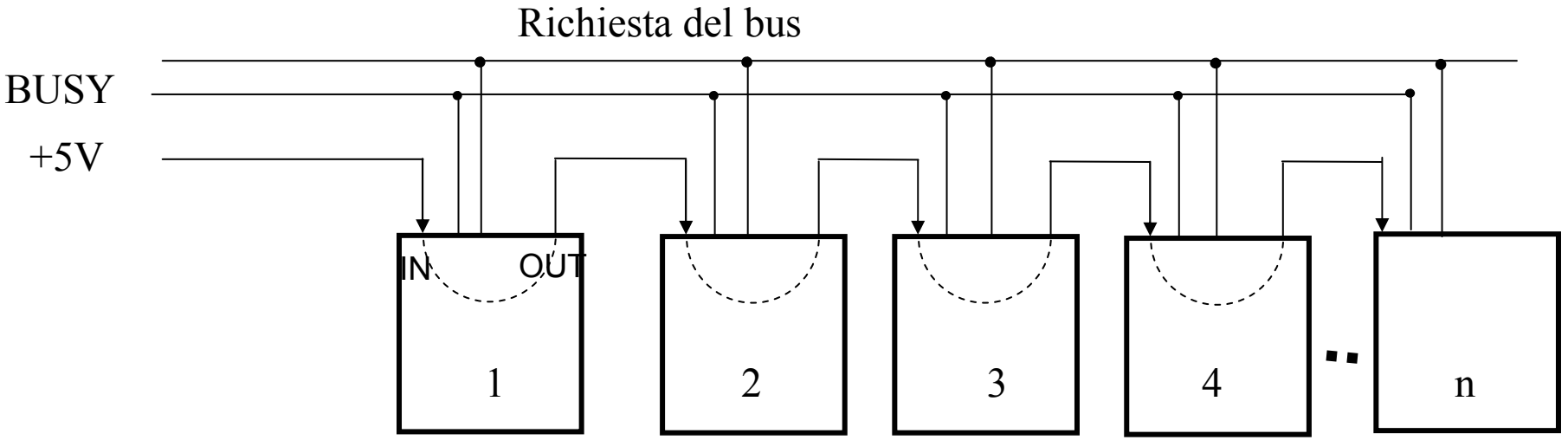
Svantaggio:
- più costoso rispetto a daisy chain
- può costituire il collo di bottiglia

NB: Il bus PCI utilizza un arbitraggio centralizzato:

ogni dispositivo ha una linea di richiesta e di concessione.

Il protocollo non specifica l'algoritmo secondo cui l'arbitro decide!

Un Meccanismo decentralizzato



Il pretendente master controlla IN e BUSY (Occupato):

se IN non asserito: bus occupato da dispositivo a maggior priorità:
il dispositivo non può diventare master, disasserisce OUT [cfr. daisy chain]

se IN asserito e BUSY asserito: bus occupato da unità con minor priorità.
Disasserisce OUT e questo provocherà a valle il disasserimento di IN-OUT

se IN asserito e BUSY non asserito (bus libero): disasserisce OUT,
asserisce BUSY e OUT , e inizia il trasferimento.

E' possibile vedere che alla fine si arriva ad uno stato stabile in cui un dispositivo accede al bus (quello che ha IN asserito e OUT non asserito)

Il meccanismo precedente è molto simile a daisy chain:

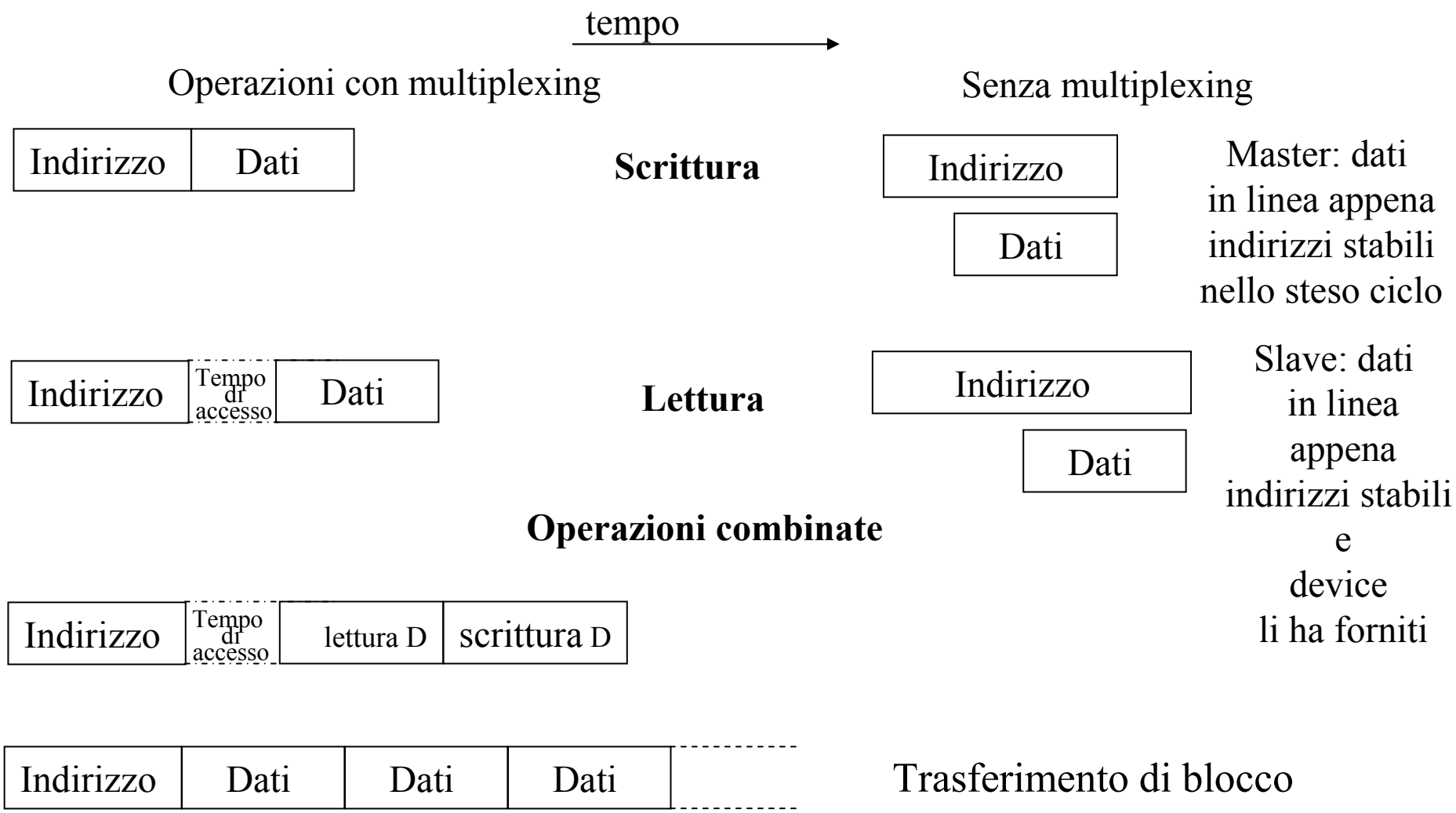
- non equo (il più vicino vince sempre)
- ma:
  senza arbitro: economico, più veloce, evita fallimenti dell'arbitro

Esistono anche altri meccanismi decentralizzati:

- Es: ciascun dispositivo scrive sul bus un codice; dalla combinazione risultante, letta da ciascun dispositivo, ciascuno è in grado di sapere se è il richiedente con priorità più elevata
- Es: arbitraggio distribuito con rilevamento delle collisioni (vedi rete Ethernet)

Tipi di operazioni dati

Oltre ai cicli di bus "ordinari" (trasferimento di parola) il protocollo può prevedere diverse altre possibilità



Ulteriori ritardi per arbitraggi

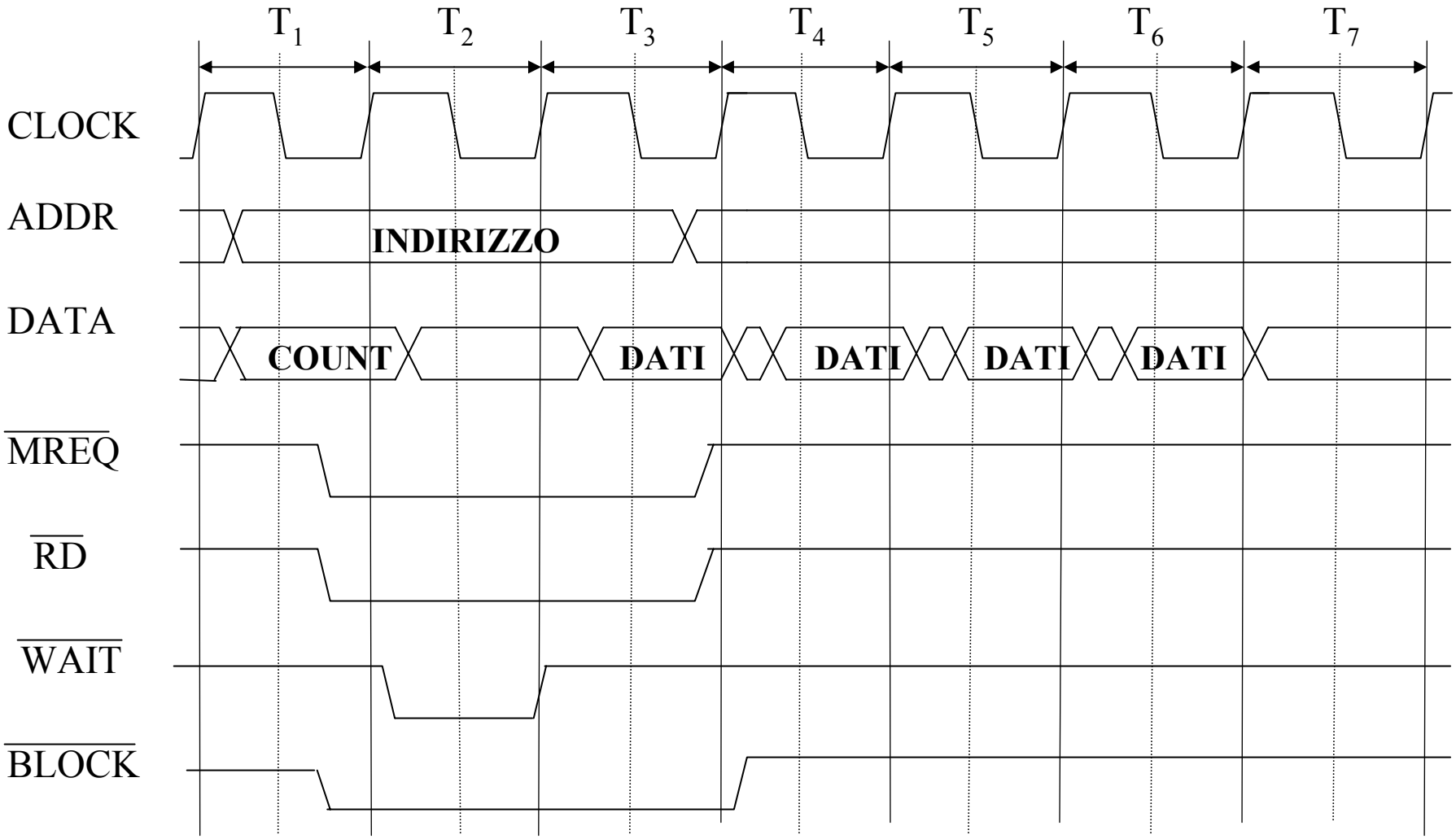
Vediamo un tipo di operazione

Trasferimento di un blocco di parole su bus sincrono

- Esempio: esiste la cache e c'è la necessità di prelevare un intero blocco di parole
- Il trasferimento riguarda 1 blocco piuttosto che una singola parola
- Quando inizia una lettura (o scrittura) del blocco il master del bus comunica allo slave del bus il numero di parole che devono essere trasferite
- Questo numero può essere ad esempio posto sulla linea dei dati
- Nel caso sincrono, lo slave trasferirà (o riceverà) una parola durante ogni ciclo di clock

Bus sincrono: lettura di un blocco di 4 parole

[impiega 6 cicli invece di 12]



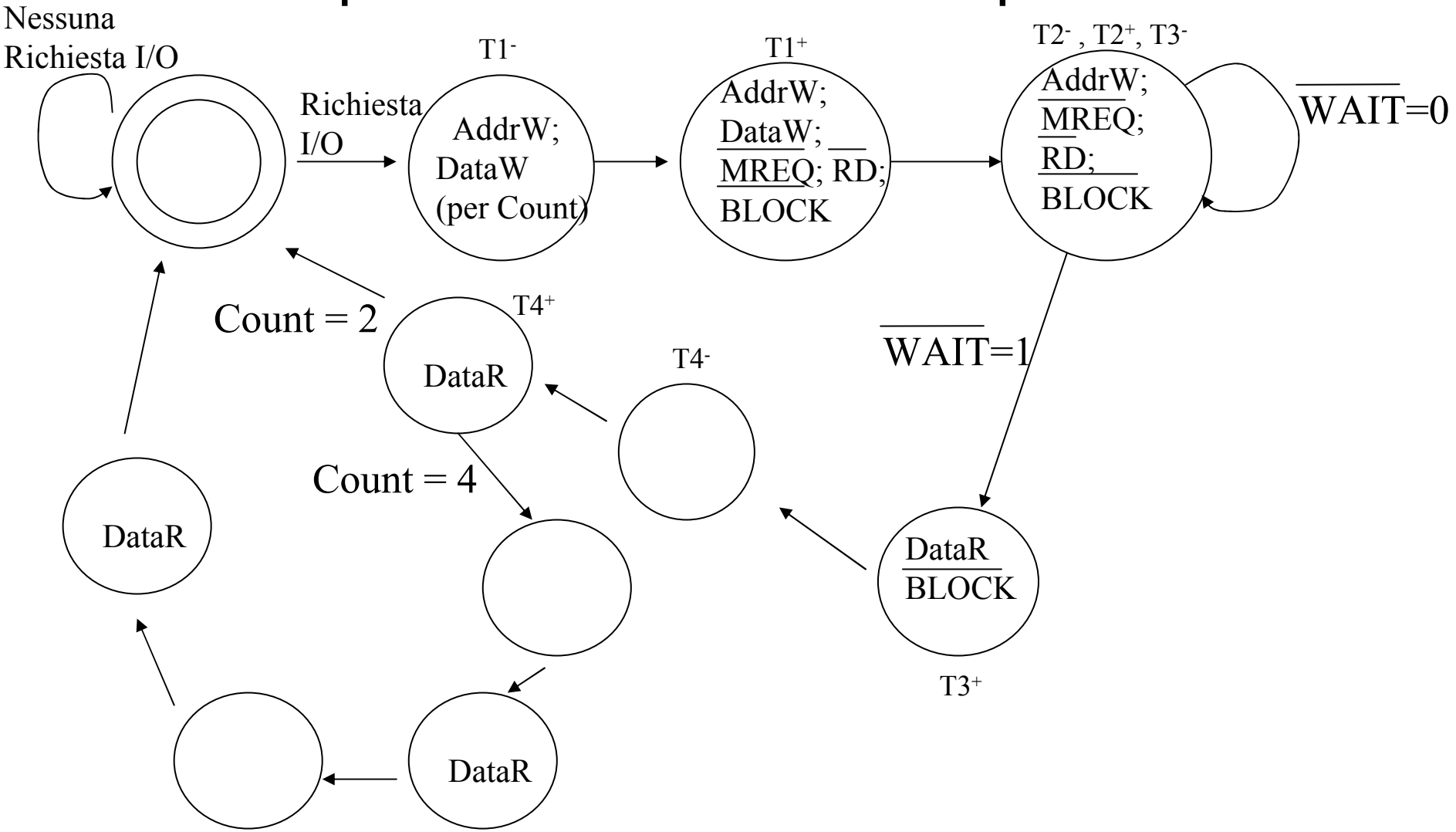
Note sul trasferimento di blocchi

- Contemporaneamente all'indirizzo viene inserito il dato "count" (numero di parole) sulle linee di dati
- Così lo slave, invece di ritornare una parola, ritorna una parola ad ogni ciclo di clock finchè non viene raggiunto il valore di count
- Il segnale BLOCK viene asserito per indicare che viene richiesto un trasferimento di blocco

ESERCIZIO TIPICO: DIAGRAMMA A STATI (P.ES. DEL MASTER)

Esempio di macchina a stati finiti (master)

per trasferimento blocchi di 2 o 4 parole



Prestazioni dei sistemi di I/O

- Le prestazioni dei sistemi di I/O dipendono dalla velocità di trasferimento dei dati
- Questa velocità è espressa in

  MB/sec (***ampiezza di banda***)
- Nei sistemi di I/O i MB sono misurati usando la base 10 (1 MB = 10^6 byte)
- Ricordare che quando si parla di memoria 1 MB = 2^{20} byte
- Ad esempio, avendo un bus a 100 MB/s, il tempo necessario per trasferire $100\ MB_2$ (100 MB di memoria) è:

  (100*1.048.576 bytes) / (100 *10^6 bytes/s) = 1,048576 s

  [differenza spesso trascurabile]

Promemoria

Prefissi

10^{12}	T	Tera	2^{40}	1.099,521.422.776
10^{9}	G	Giga	2^{30}	1.073.741.824
10^{6}	M	Mega	2^{20}	1.048.576
10^{3}	K	Kilo	2^{10}	1024
10^{2}	h	Etto		
10^{1}	da	Deca		
10^{0}		unità	2^{0}	1
10^{-1}	d	Deci		
10^{-2}	c	Centi		
10^{-3}	m	Milli		
10^{-6}	μ	Micro		
10^{-9}	n	Nano		
10^{-12}	p	Pico		

Varie

1 byte=8bit

$2^{11} = 2048$

16 K= 2^{14}

Esercizio

- Si consideri il bus sincrono illustrato in precedenza, assumendo di disporre di 32 bit per le linee dati e di un clock di 200 MHz
- Si assuma di poter effettuare (secondo le modalità viste) tre tipi di trasferimento:
  - singola parola
  - blocchi di 2 parole
  - blocchi di 4 parole

  Tra un trasferimento e il successivo, è necessario 1 ciclo di clock di attesa
- Si consideri il trasferimento di 256 parole (di 32 bit):

  Si calcoli, per ciascuna delle tre modalità di trasferimento, il tempo di trasferimento e l'ampiezza di banda (in bytes/s) raggiunta dal sistema memoria-bus
- Si assuma di poter disporre di una memoria arbitrariamente veloce.

  Trovare – nello stesso caso considerato – le massime prestazioni raggiungibili [in termini di tempo di trasferimento e ampiezza di banda]

Soluzione

- F_{bus} = 200 MHz $\Rightarrow$ T_{bus} = 5 ns $[1/(2*10^8) = 10/2 * 10^{-9} = 5 * 10^{-9}]$
- In ogni caso, vengono trasferiti 256*4 bytes = 1024 bytes

A questo punto, dal diagramma temporale si ricavano i cicli necessari per ciascun tipo di trasferimento, consentendo di ricavare i dati richiesti…

Trasferimento a parole:

1 parola (4 bytes) richiede 3 cicli di clock + 1 di attesa

Totale: 256* 4 = 1024 cicli

$\Rightarrow$ T_{trasf} = 1024*5 ns = 5120 ns

$\Rightarrow$ Larghezza di banda = $(1024/5120*10^{-9})$ = 200 MB/s

Trasferimento a blocchi di due parole:

E' necessario trasferire 256/2 = 128 blocchi

1 blocco (2 parole) richiede 4 cicli di clock + 1 di attesa

Totale: 128* 5 = 640 cicli

$\Rightarrow T_{trasf} = 640*5$ ns = 3200 ns

$\Rightarrow$ Larghezza di banda = $(1024/3200*10^{-9})$ = 320 MB/s

Trasferimento a blocchi di quattro parole:

E' necessario trasferire 256/4 = 64 blocchi

1 blocco (4 parole) richiede 6 cicli di clock + 1 di attesa

Totale: 64* 7 = 448 cicli

$\Rightarrow T_{trasf} = 448*5$ ns = 2240 ns

$\Rightarrow$ Larghezza di banda = $(1024/2240*10^{-9})$ = 457 MB/s

Se si dispone di una memoria arbitrariamente veloce, scompare nel diagramma temporale il ciclo di WAIT, quindi ciascun tipo di trasferimento richiede un ciclo in meno…

Si possono rifare tutti i calcoli con i nuovi numeri di cicli richiesti