

I bus

M. Sonza Reorda

Politecnico di Torino
Dip. di Automatica e Informatica



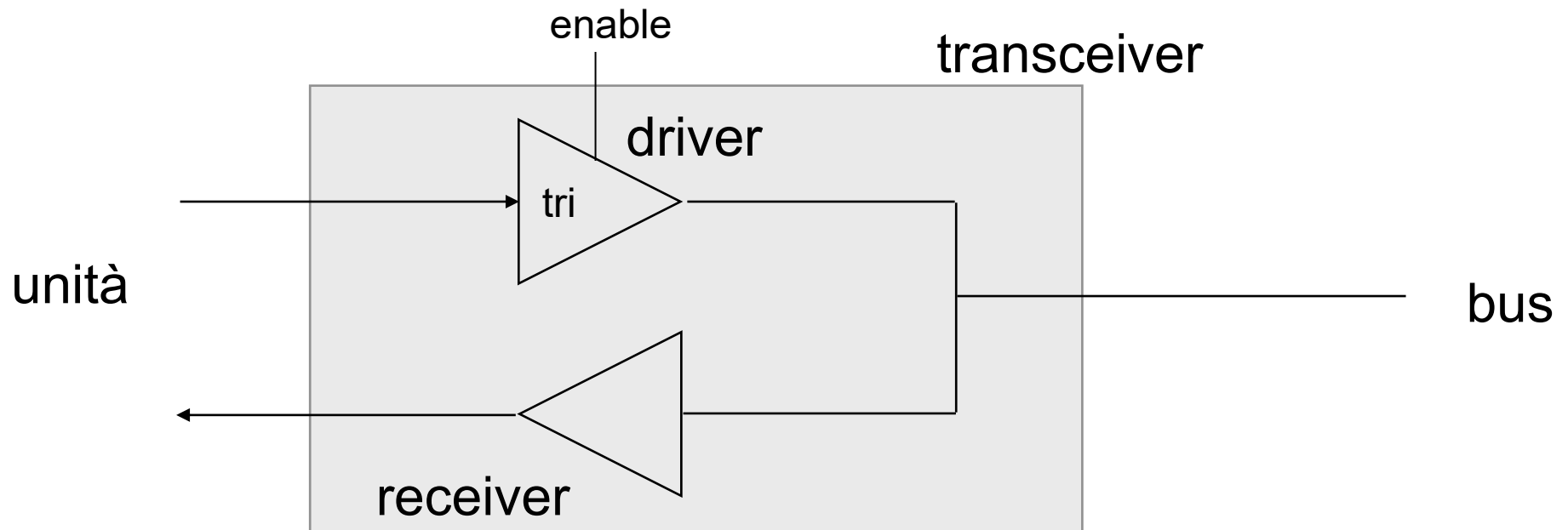
Introduzione

Un bus è una struttura che interconnette due o più dispositivi.

Un bus è una struttura condivisa: i valori che un dispositivo scrive sul bus sono accessibili a tutti gli altri dispositivi connessi.

Interfaccia al bus

- Le unità connesse al bus utilizzano 2 tipi di dispositivi:
 - *driver* per pilotare le linee del bus (driver tri-state)
 - *receiver* per leggere i valori sul bus
- I due dispositivi sono spesso raggruppati in un'unica entità denominata *transceiver*.



Implementazione del bus

Si possono avere bus:

- **interni ad un singolo circuito integrato**
- **per la connessione di più circuiti integrati su una scheda**
- **per la connessione di più schede in un sistema (*backplane*).**

Struttura di un bus

Un bus è composto da 3 gruppi di segnali:

- segnali di *dato*: normalmente sono in numero pari ad un multiplo di 8; possono essere bidirezionali o unidirezionali (in tal caso è necessario un numero doppio di linee);
- segnali di *indirizzo*: identificano lo slave con cui il master vuole comunicare (nonché quale parte dello slave è coinvolta);
- segnali di *controllo*: forniscono informazioni di stato, di temporizzazione, di tipo (dei dati sul bus).

Bus multiplexati

In taluni casi le stesse linee portano, in tempi diversi, segnali di tipo diverso.

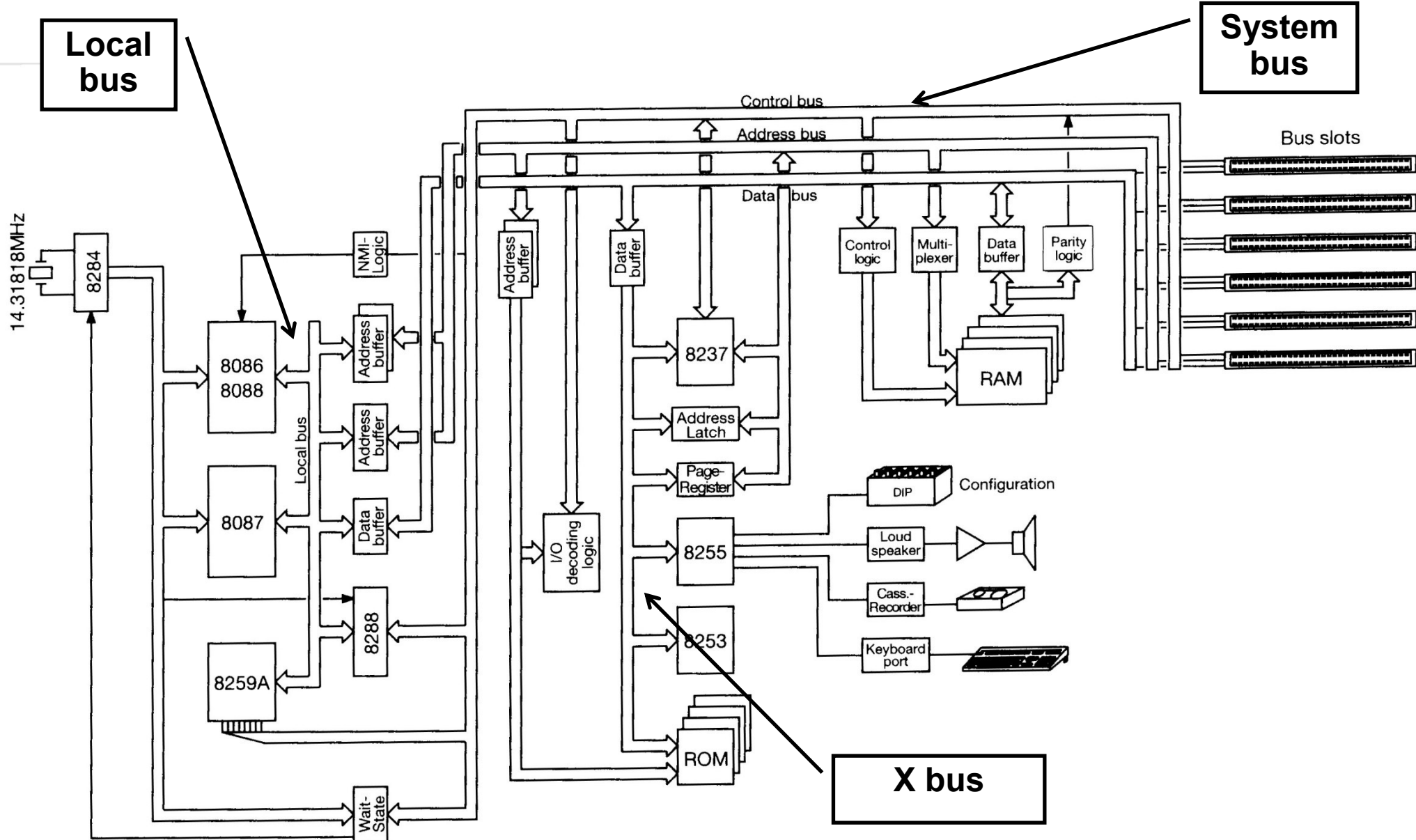
In tal modo

- **si riduce il numero di linee del bus**
- **è necessaria una circuiteria aggiuntiva.**

Esempio

In alcuni bus le linee di dato e quelle di indirizzo sono multiplexate, e a seconda dei momenti le stesse linee portano segnali di dato o di indirizzo.

Esempio: bus di un sistema 8086



Standard

Al fine di facilitare lo sviluppo di dispositivi compatibili a livello di bus, sono stati introdotti degli standard che ne descrivono le caratteristiche a livello

- ***meccanico*** (ad esempio per i bus di backplane il numero delle linee, il tipo dei connettori, le dimensioni delle schede)
- ***elettrico*** (ad esempio i valori delle tensioni e correnti di riferimento)
- ***logico/funzionale*** (ad esempio il significato dei vari segnali e la sequenza di valori che devono assumere per eseguire ciascuna operazione).

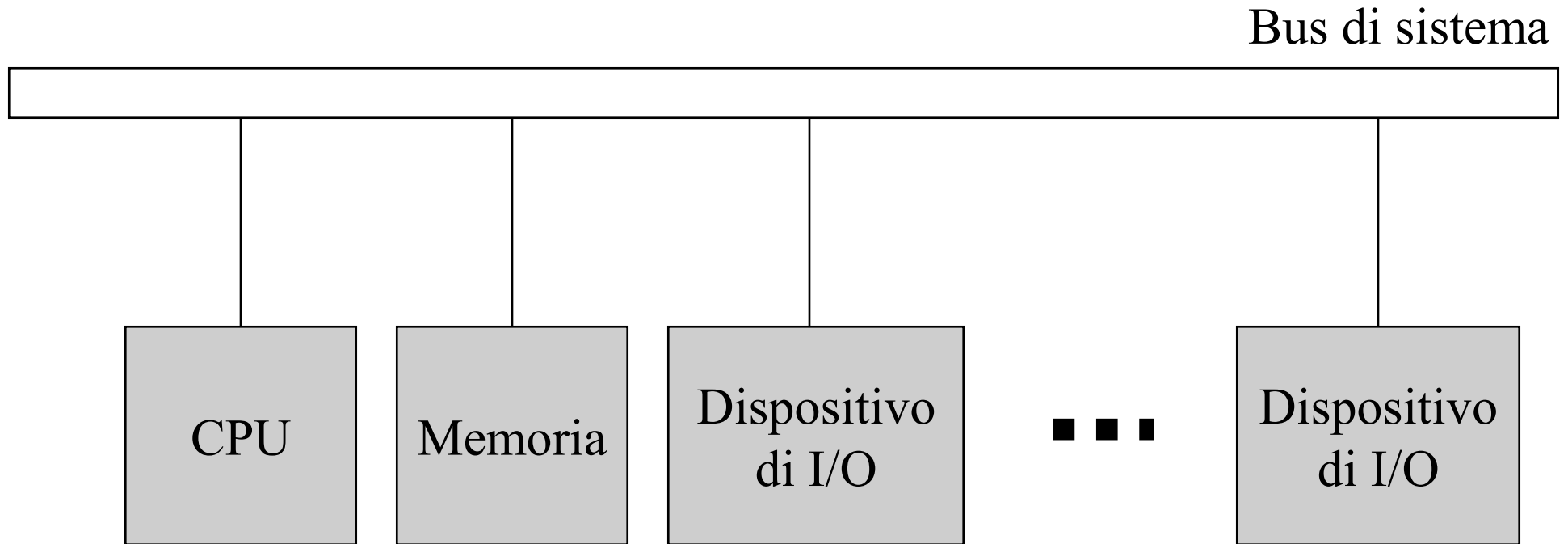
Esempi di standard sono *Multibus*, *PCI*, *VME*, *EISA*, *Futurebus+*, *SCSI*, *AMBA*.

Architetture di bus

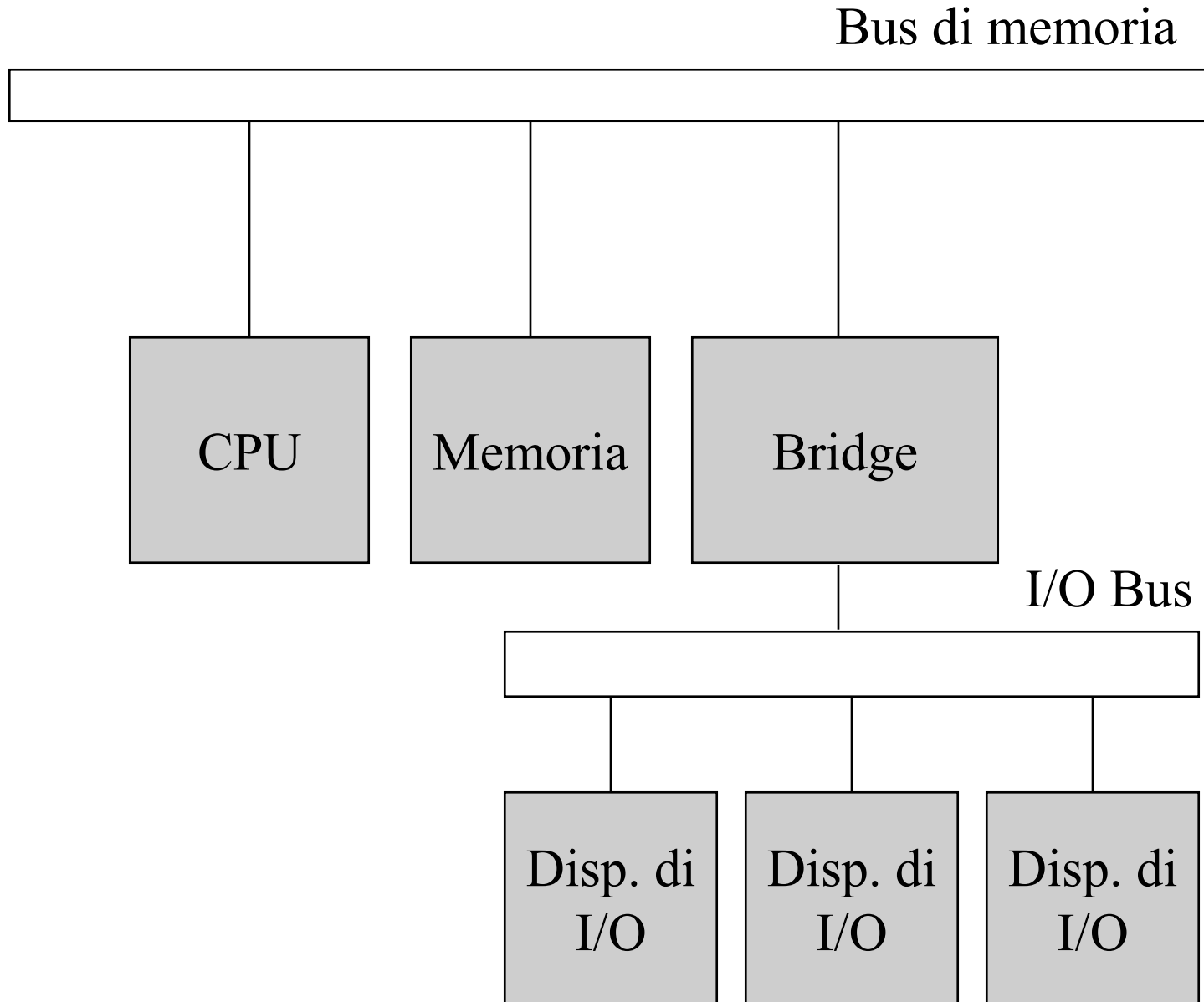
Si possono avere 2 tipi di architetture a bus:

- *bus singolo*: è la configurazione più semplice
- *bus multiplo*: è utile laddove si desiderano prestazioni elevate, oppure quando si devono connettere diverse classi di dispositivi, con caratteristiche tra loro diverse.

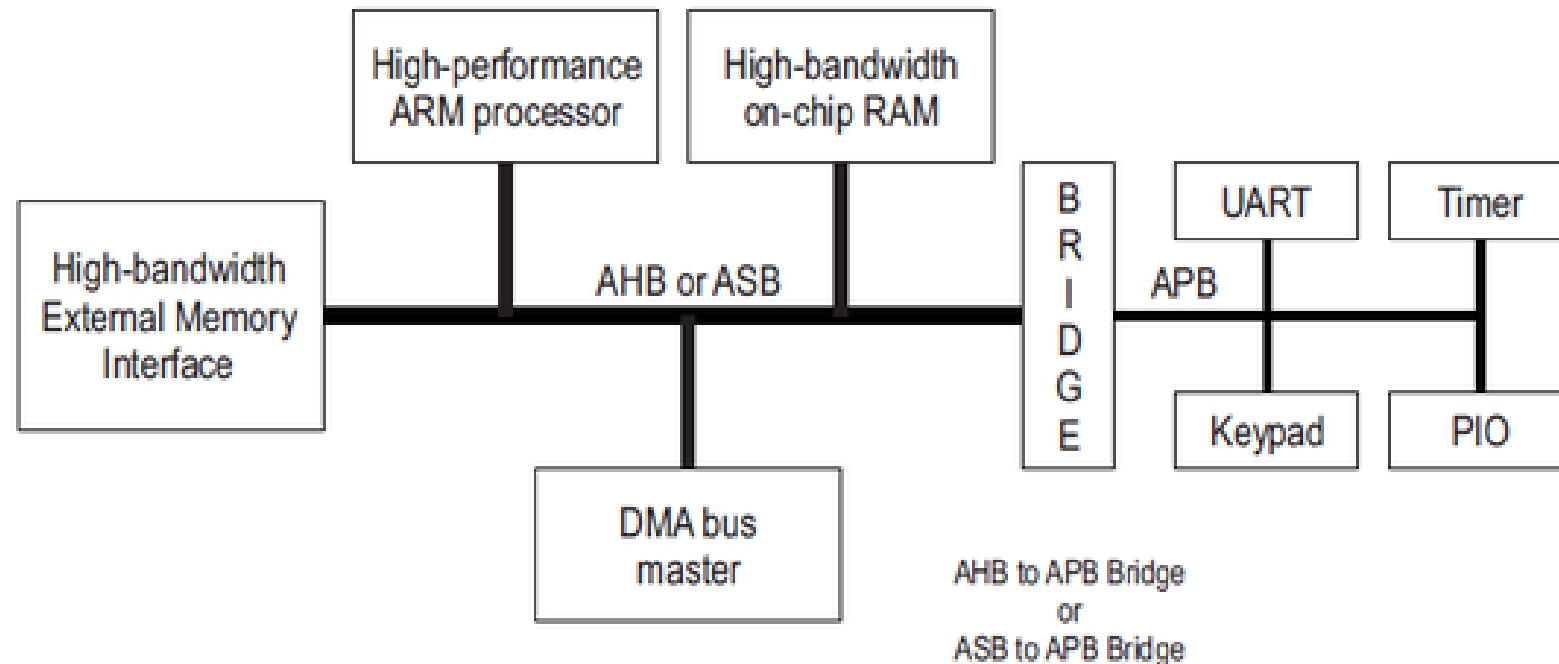
Bus singolo



Bus multiplo (esempio)



Esempio: architettura a più livelli



Master e slave

In un sistema a bus le unità connesse sono di 2 tipi:

- **unità *master*: inizia ogni procedura di trasferimento dati e sceglie lo slave con cui comunicare**
 - **nei sistemi più semplici esiste un'unica unità master, che coincide con la CPU**
 - **nei sistemi più complessi esistono più unità master, e l'unità master cambia a seconda dei momenti**
- **unità *slave*: risponde ai comandi dell'unità master**
 - **le memorie e le interfacce dei periferici sono unità slave.**

Problemi

L'utilizzo di un sistema a bus richiede la soluzione di 2 principali problemi:

- **la definizione delle tempistiche con cui si svolgono le operazioni sul bus**
- **l'introduzione di un meccanismo per la gestione dei conflitti nell'accesso al bus.**

Tempistiche

Come possono le varie unità connesse ad un bus sapere quando leggere o scrivere dal/sul bus?

Le soluzioni possibili sono riconducibili a due tipologie:

- **Bus *sincroni***
- **Bus *asincroni*.**

Le specifiche di un bus comprendono tra l'altro la descrizione del *protocollo* che i segnali devono seguire, nonché i limiti di tempo che devono essere rispettati.

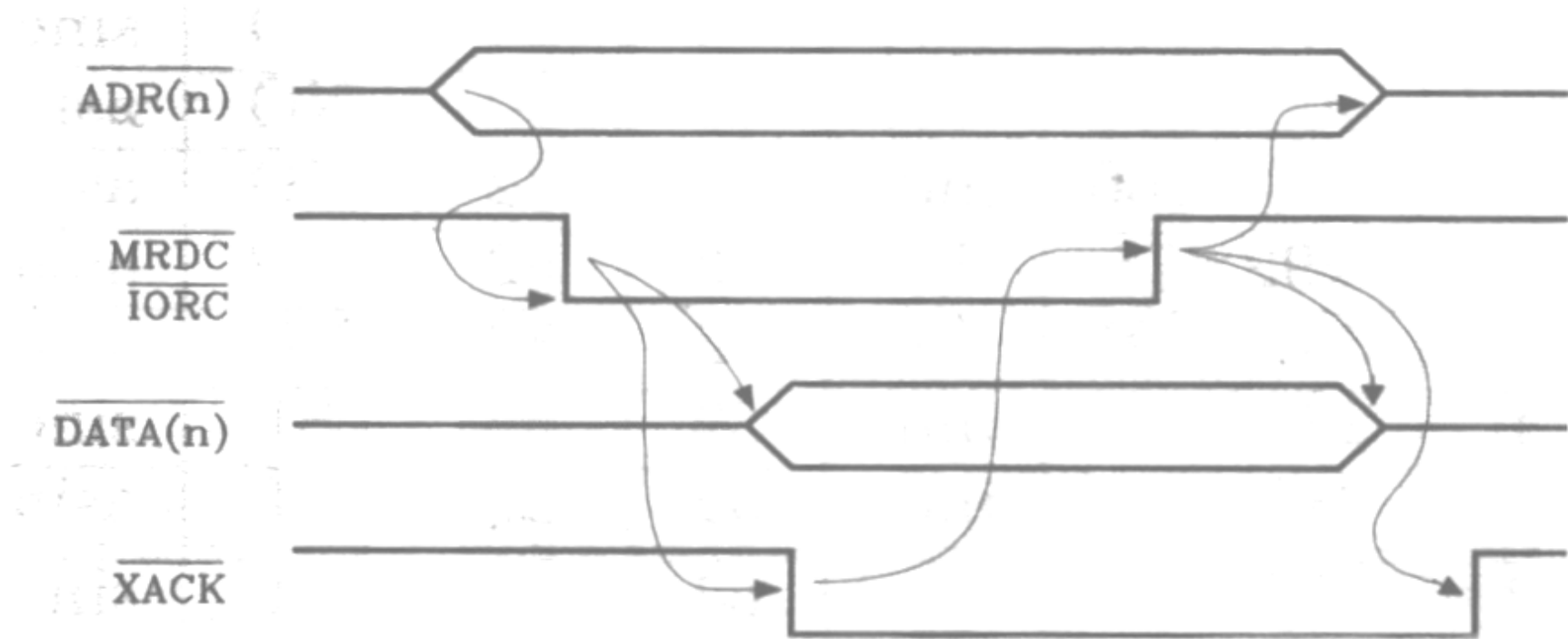
Bus sincroni

- Le unità sorgente e destinazione utilizzano lo stesso segnale di clock, che fa parte del bus stesso; alternativamente, le 2 unità possono avere clock separati, ma alla stessa frequenza, e scambiare periodicamente segnali di sincronizzazione
- la frequenza del clock è imposta dal dispositivo più lento
- ogni unità di dato è trasferita in un periodo di tempo prefissato (normalmente un periodo di clock)
- il meccanismo funziona bene su distanze ridotte.

Bus asincroni

- Ogni operazione di comunicazione può avere una sua velocità, determinata da appositi segnali di controllo che accompagnano i segnali di dato e implementano il cosiddetto *handshaking*
- si ottiene così la massima flessibilità, a spese di una maggiore complessità del protocollo.

Multibus I: ciclo di lettura



Multibus I: ciclo di lettura

1. Il master pone gli indirizzi su ABUS

3. Lo slave mette i dati su DBUS

5. Dopo la lettura, il master disattiva il segnale di controllo e libera ABUS

$\overline{\text{ADR}}(n)$

$\overline{\text{MRDC}}$
 $\overline{\text{IORC}}$

$\overline{\text{DATA}}(n)$

$\overline{\text{XACK}}$

2. Quando gli indirizzi sono stabili, il master attiva un segnale di controllo (memoria o I/O)

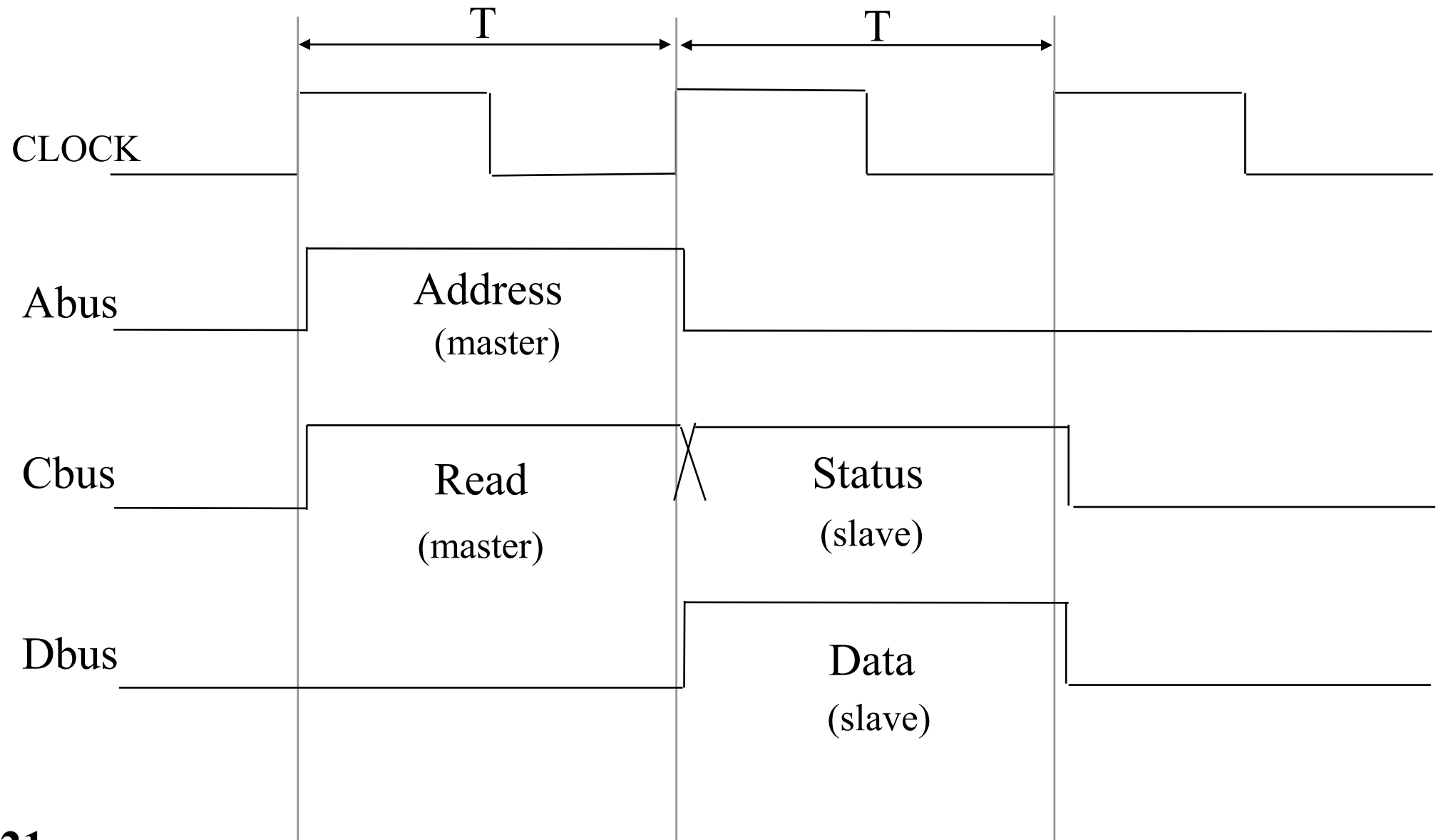
4. Slave attiva XACK

6. Lo slave disattiva XACK e libera DBUS

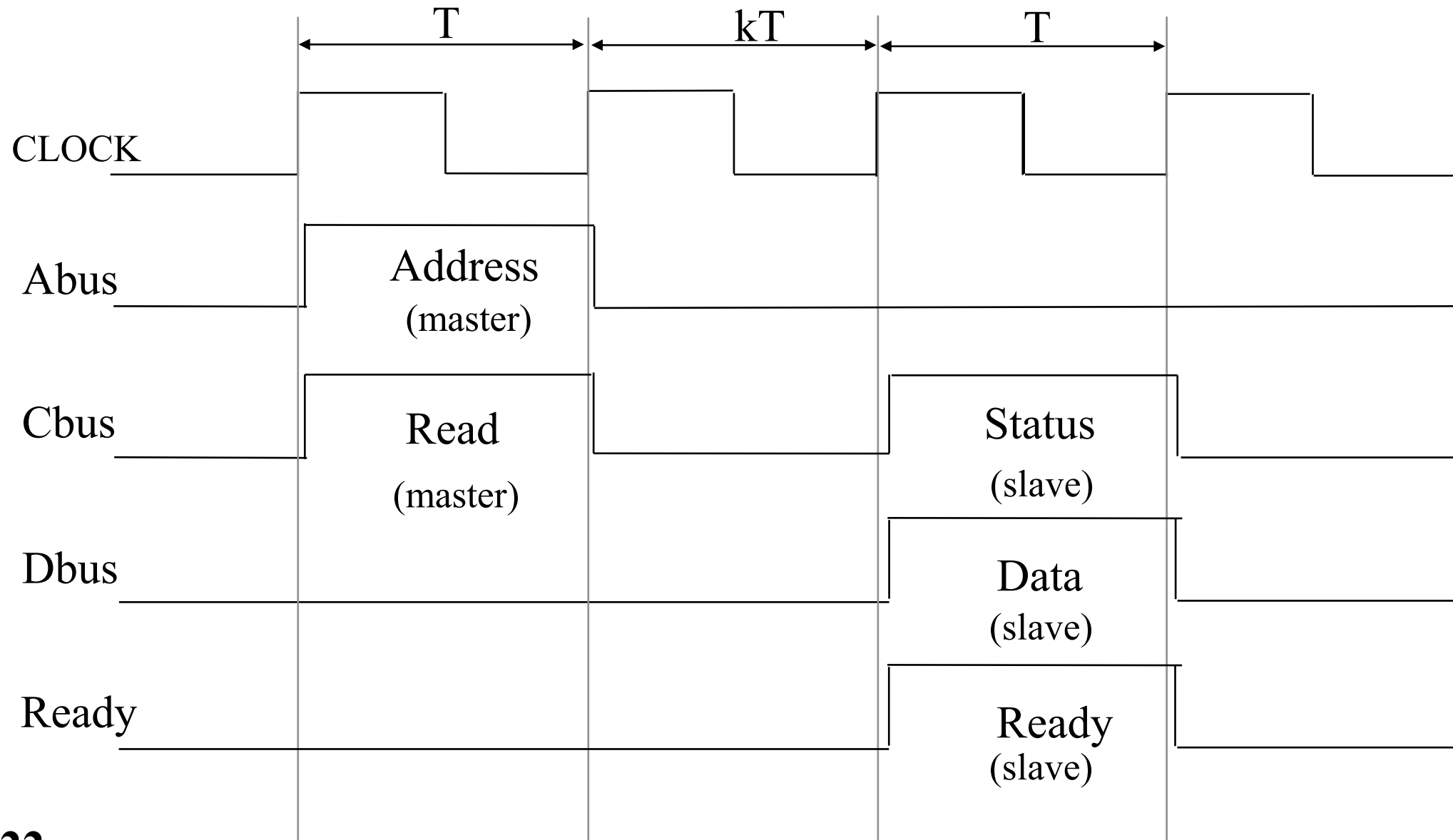
Cicli di wait

- Nel caso di bus sincroni può esistere un meccanismo per introdurre occasionalmente dei cicli aggiuntivi (detti di *wait*) che permettono allo slave di ottenere più tempo per eseguire l'operazione richiesta
- Il meccanismo utilizza un segnale apposito (READY) che segnala la necessità di aggiungere o meno i cicli aggiuntivi.

Lettura senza cicli di wait



Lettura con cicli di wait



Arbitraggio

- Ad ogni istante un solo dispositivo può funzionare da master del bus
- Il meccanismo di arbitraggio del bus entra in funzione quando 2 o più unità fanno contemporaneamente richiesta di diventare master del bus: il meccanismo deve allora designare il nuovo dispositivo master
- L'arbitraggio può avvenire in maniera
 - *centralizzata*: esiste un *arbitro*
 - *distribuita*: ogni modulo contiene la logica necessaria per implementare un meccanismo di arbitraggio che permette di definire il nuovo master.

Arbitraggio distribuito: il bus SCSI

- **Il bus SCSI possiede 8 linee DB (0), ..., DB (7) che vengono utilizzate sia per il trasferimento dati che per l'arbitraggio**
- **Durante l'arbitraggio, ogni linea è associata a un dispositivo: la linea DB (7) ha la priorità massima**
- **Quando la linea BSY diventa inattiva, tutti i dispositivi (al più 8) che desiderano fare accesso al bus alzano la rispettiva linea DB (i). Tutti i dispositivi osservano il valore sulle linee DB (0), ..., DB (7), e il dispositivo con priorità massima vince la contesa; gli altri attendono che BSY torni inattiva.**

Arbitraggio centralizzato

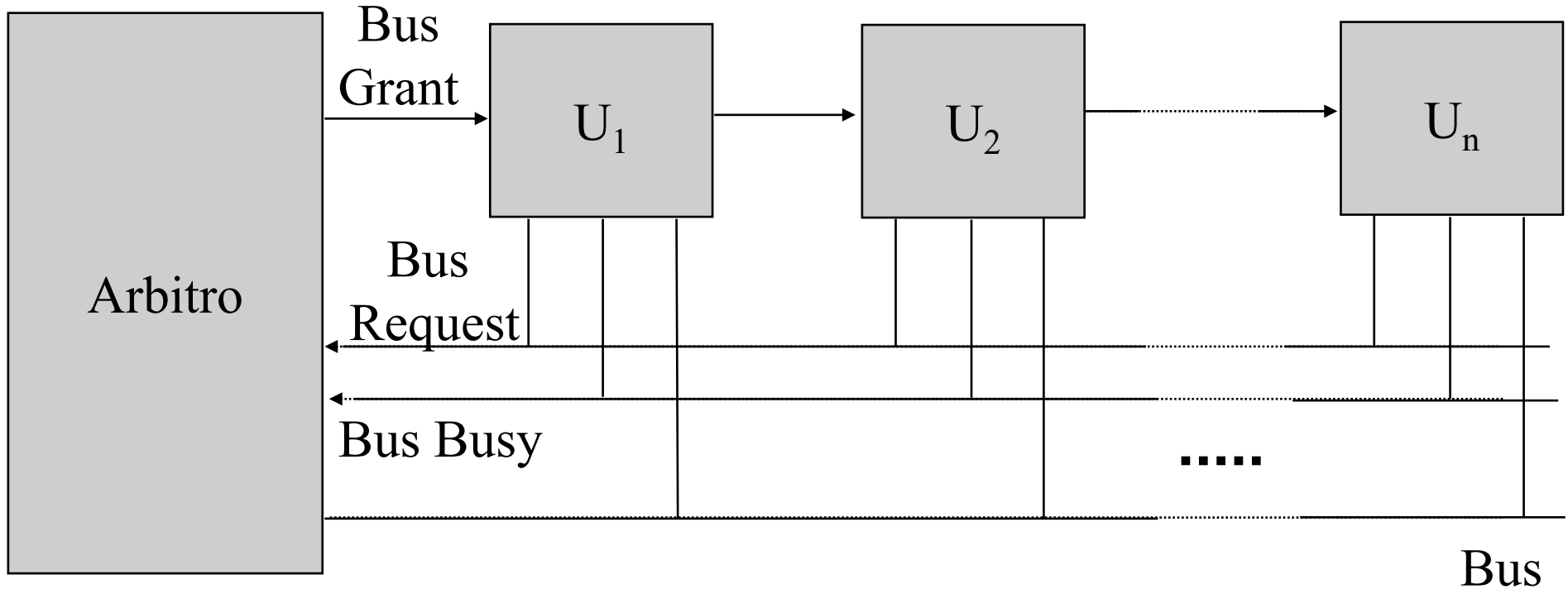
Esistono 3 meccanismi di arbitraggio centralizzato:

- *Daisy Chaining*
- *Polling*
- *Richieste indipendenti.*

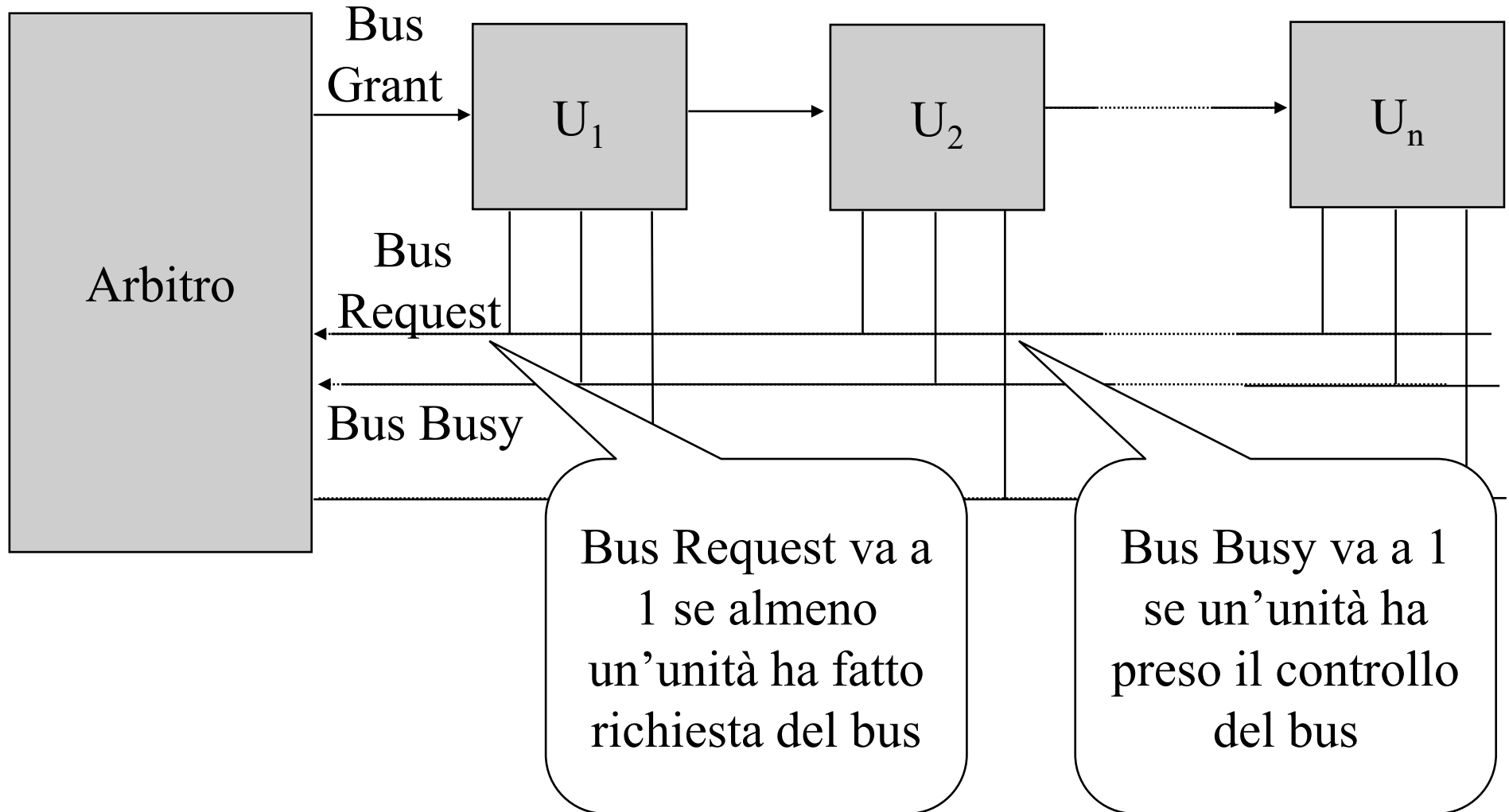
Essi differiscono per:

- numero di linee di controllo richieste
- velocità di risposta del bus controller
- flessibilità nella gestione delle priorità
- tolleranza ai guasti.

Daisy Chaining: struttura



Daisy Chaining: struttura



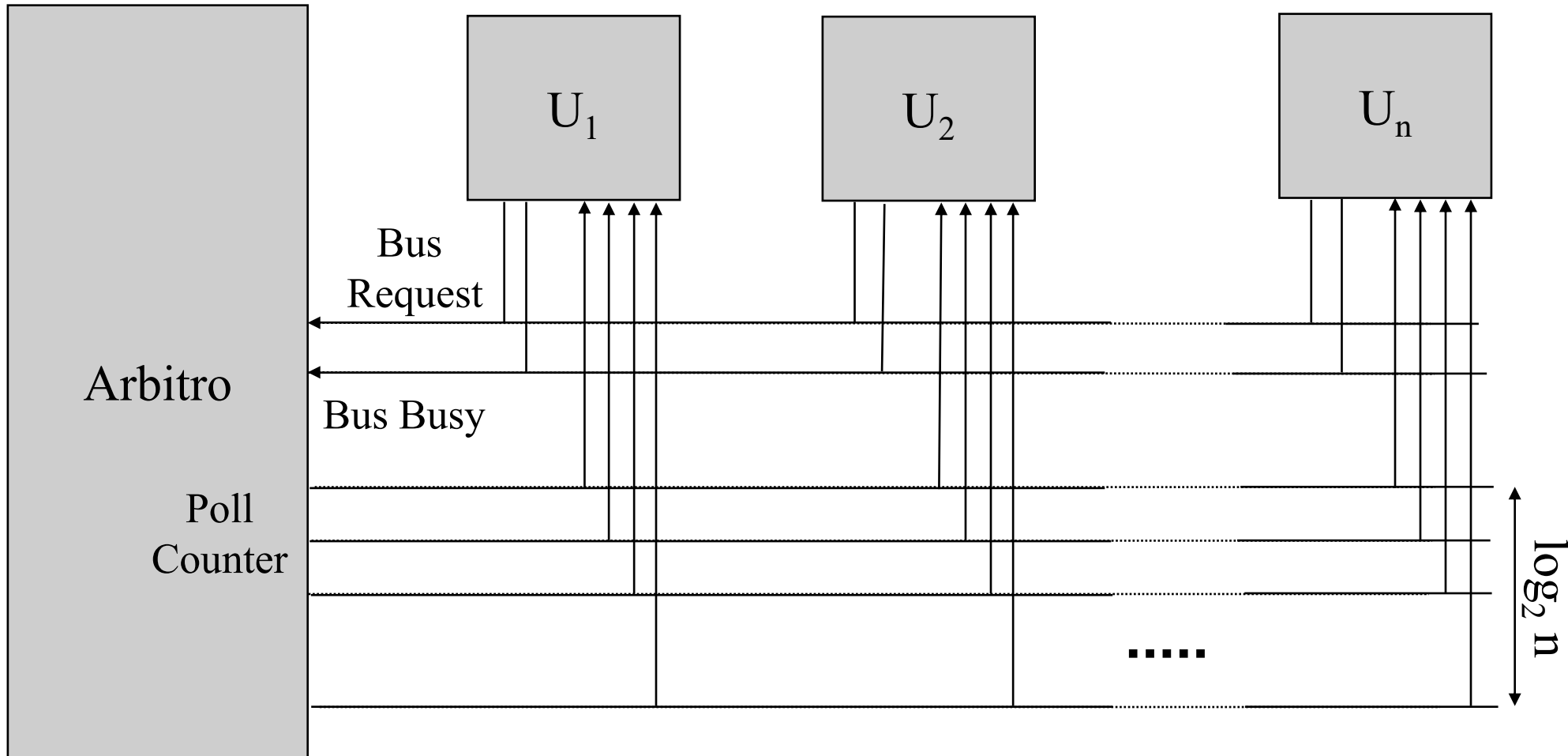
Daisy Chaining: funzionamento

- Un'unità fa richiesta del bus (BUS REQUEST) attendendo che il bus sia libero (BUS BUSY)
- L'arbitro attiva il segnale di BUS GRANT
- Ogni unità, quando riceve il BUS GRANT:
 - se ha richiesto il bus: attiva BUS BUSY e prende il controllo del bus
 - se non ha richiesto il bus: attiva BUS GRANT verso l'unità a valle.

Daisy Chaining: caratteristiche

- + richiede solo 3 segnali di controllo**
- non permette di modificare le priorità**
- non è adatta a numeri elevati di dispositivi connessi**
- non è tollerante ai guasti.**

Polling: struttura



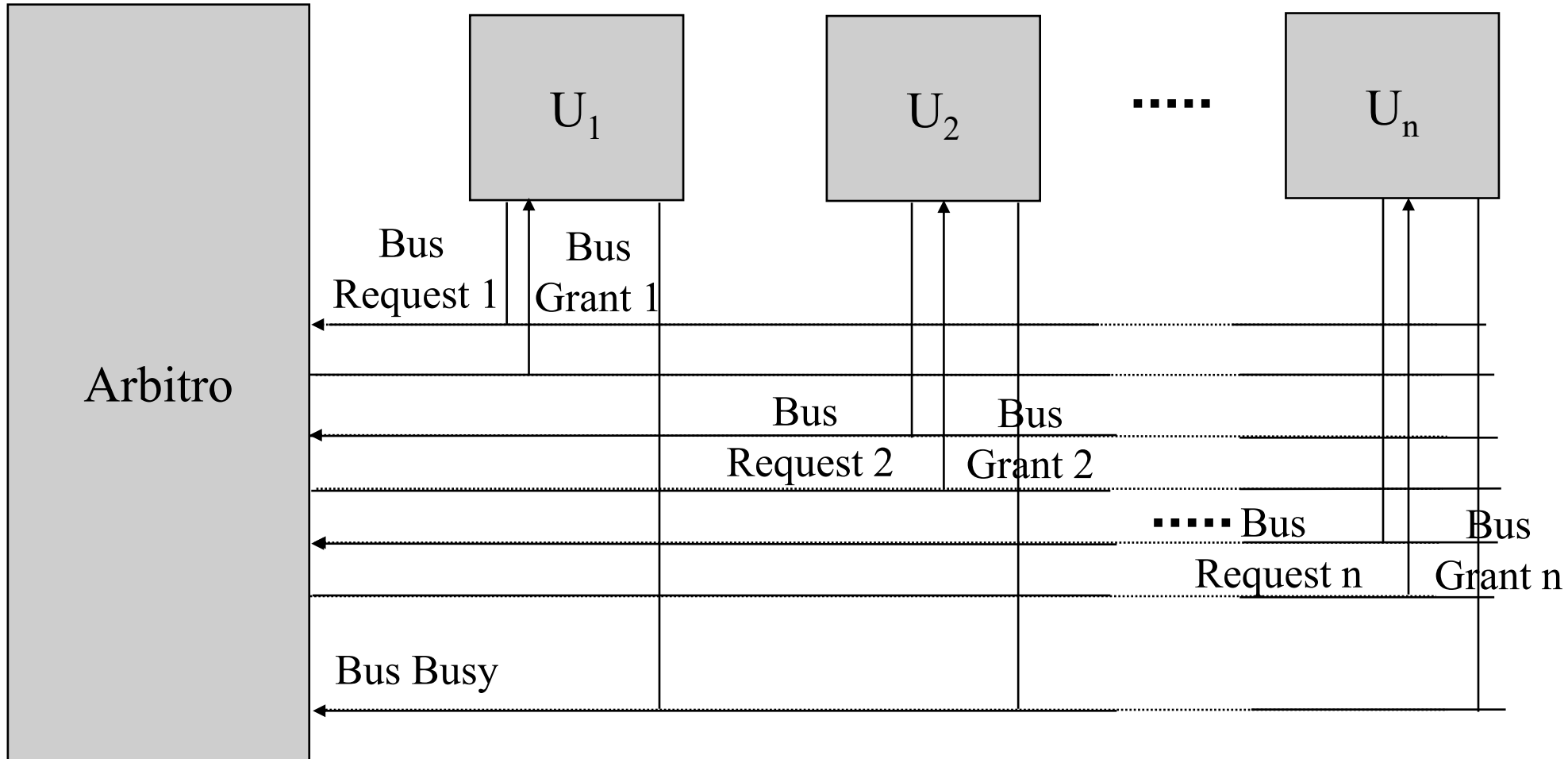
Polling: funzionamento

- Un'unità fa richiesta del bus (BUS REQUEST), attendendo che il bus sia libero (BUS BUSY)
- L'arbitro scandisce tutte le unità collegate, mettendo sulle linee POLL COUNTER l'identificativo di ciascuna, in sequenza
- Quando un'unità è indirizzata
 - se aveva fatto richiesta attiva il segnale BUS BUSY; a questo punto l'arbitro interrompe la scansione
 - se no, non fa nulla, e l'arbitro passa all'unità successiva
- L'unità prende il controllo del bus.

Polling: caratteristiche

- richiede $2 + \log(n)$ segnali di controllo per gestire n unità
- + la priorità delle unità può essere cambiata modificando la sequenza di scansione
- + il sistema è tollerante a un eventuale guasto in un'unità.

Richieste indipendenti: struttura



Richieste indipendenti: funzionamento

- **L'unità i-esima, quando vuole fare accesso al bus**
 - **attende che il bus sia libero (osservando BUS BUSY)**
 - **fa richiesta del bus (attivando BUS REQUEST i)**
- **L'arbitro gestisce tutte le richieste, e concede il bus all'unità con priorità massima (BUS GRANT j) tra quelle che hanno fatto richiesta**
- **L'unità j assume il controllo del bus (BUS BUSY).**

Richieste indipendenti: caratteristiche

- richiede $2*n+1$ segnali di controllo per gestire n unità**
- + le priorità dei dispositivi dipendono dai meccanismi implementati dall'arbitro**
- + il sistema può tollerare un eventuale guasto in una unità.**