Calcolatori Elettronici
Il Bus

Ing. Gestionale e delle Telecomunicazioni

A.A. 2008/09

Gabriele Cecchetti

Sommario

- Il bus
- Il bus asincrono
- Il bus sincrono
- Il bus semisincrono
- Arbitraggio del bus
- Set di operazioni del bus

Il bus (1/2)

- Il BUS è l'insieme delle linee che collegano i moduli di un sistema di elaborazione
- Può avere dimensioni fisiche molto diverse:
  - il bus interno alla CPU mette in comunicazione i moduli che la compongono;
  - il bus di un calcolatore interconnette la CPU, le schede di I/O e la memoria;
  - il bus SCSI è utilizzato per connettere le periferiche a un calcolatore e può avere una estensione di qualche metro;
  - il portante fisico di una rete Ethernet può essere considerato un bus in grado di connettere calcolatori.
- Possono essere considerati tutti bus in quanto le funzionalità e le problematiche da affrontare sono le medesime.

Il bus (2/2)

- Un bus è costituito da un fascio di collegamenti elettrici per le funzioni di
  - controllo,
  - indirizzo,
  - dati.
- Affinché i moduli connessi dal bus siano in grado di comunicare è necessario che essi interagiscano con il bus secondo un insieme di regole ben definite.
- L'insieme delle regole viene definito come **il protocollo del bus.**

Tipi di bus commerciali

Esistono numerosi bus in commercio, ognuno con il suo protocollo e le sue caratteristiche fisiche ed elettriche (tensione, temporizzazione, connettori, …):

- PCI, PCI-X, PCI-Express
- SCSI,
- USB, FireWire
- Ethernet,
- ecc.

Linee del bus

Le linee del bus (ad esclusione di quelle relative alla alimentazione) possono essere distinte in due tipi:

- quelle che trasportano informazioni a livelli;
- quelle che trasportano segnali.

Nelle linee che trasportano segnali è importante l'istante in cui avviene la transizione da un livello ad un altro.

Linee di segnale del bus

Le linee di segnale sono usate per trasportare informazioni temporali:

- validare lo stato delle linee a livello
- trasportare segnali di clock
- segnalare l'avanzamento di un protocollo

Linee che trasportano informazioni a livelli sul bus

Le linee che trasportano informazioni a livelli sono utilizzate per trasferire:

- dati,
- indirizzi,
- comandi,
- informazioni di stato.

Linee del bus condivise

- Entrambi i tipi di linee possono essere condivise tra diversi moduli e quindi possibilmente pilotate in istanti diversi da moduli diversi.
- E' necessario organizzare il sistema in modo tale che non si verifichino conflitti nell'accesso alle linee condivise.

Moduli presenti sul bus

- I moduli presenti sul bus comunicano pilotando le linee con una temporizzazione ed una sequenza di azioni ben precise.
- Il protocollo del bus è costituito da una sequenza di azioni elementari quali:
  - *presentare una certa configurazione sulle linee dati*;
  - *generare un fronte in salita/discesa su una linea*;
  - *attendere il trascorrere di un certo intervallo di tempo*.
- Nel caso più semplice la comunicazione coinvolge due partecipanti:
  - il **MASTER**: il modulo che richiede la comunicazione;
  - lo **SLAVE**: il modulo che risponde alla richiesta.

Esempi di combinazioni master-slave

Master	Slave	Esempio
CPU	Memoria	Prelievo di istruzioni e dati
CPU	Dispositivo di I/O	Inizio trasferimento dati
CPU	Coprocessore	Passaggio operandi
I/O	Memoria	DMA

Organizzazione di un bus

I principali problemi che un bus deve risolvere sono:

- chi può eseguire una operazione
- quale operazione deve essere eseguita
- quando una operazione deve essere eseguita

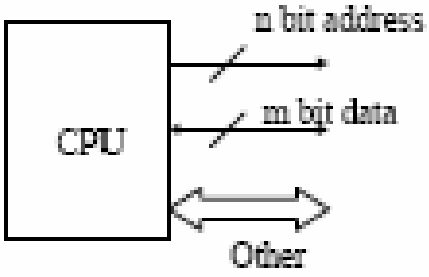
Parametri di un bus

I principali parametri di progettazione di un bus sono:

- Larghezza
- Temporizzazione
- Arbitraggio
- Set di operazioni

Linee di un bus

- Il numero delle linee utilizzate per trasferire gli indirizzi determinano la massima quantità di memoria indirizzabile
- Il numero delle linee utilizzate per il trasferimento dei dati determinano la quantità di informazione che è possibile trasferire con una singola operazione
- Esempio: è possibile
  - indirizzare 2^n locazioni di memoria
  - trasferire m bit alla volta



Larghezza di un bus

Per **aumentare la banda di un bus** è possibile incrementare:

1. il numero di trasferimenti per unità di tempo;
2. il numero di dati per trasferimento.

Aumentare 1 è difficile in quanto nella realtà:

- le linee introducono un ritardo nella propagazione dei segnali elettrici e possono distorcerne la forma;
- all'aumentare della frequenza di lavoro aumentano anche gli effetti negativi derivanti dagli accoppiamenti capacitivi ed induttivi tra le linee del bus.

Aumentare 2 significa aumentare **m** e quindi aumentare la larghezza del bus.

Bus multiplexato

- Nel caso che la larghezza del bus sia eccessiva è possibile optare per un **bus multiplexato.**
- In questo caso le linee utilizzate per il trasferimento dei dati e degli indirizzi sono le stesse.
- All'inizio di un'operazione le linee sono utilizzate per il trasferimento degli indirizzi, successivamente per il trasferimento dei dati.
- Poiché dati ed indirizzi non possono essere posti sul bus nello stesso istante (come ad esempio viene fatto dal bus non multiplexato durante una scrittura in memoria), il bus multiplexato è più lento.

Linee di sincronizzazione

In precedenza abbiamo separato le linee che portano segnali di sincronizzazione da quelle che portano informazioni a livelli quali ad es. i dati, gli indirizzi, i comandi.

Nel caso di un processore con due spazi di indirizzamento (memoria ed I/O),

- due linee possono essere utilizzate per definire quale operazione effettuare e su quale spazio (**R/W** e **MEM/IO**),
- una terza linea **/REQ** può essere utilizzata come segnale di sincronizzazione.

In alcuni sistemi commerciali esistono invece linee di sincronizzazione separate per le varie operazioni ad es:

/IOW /IOR /MR /MW.

In questo caso il numero di linee utilizzate è maggiore, ma gli slave risultano semplificati in quanto in molti casi è possibile fornire le informazioni di sincronizzazione direttamente agli slave (eliminando una barriera di logica).

Linee di sincronizzazione a livello

- Questo approccio è possibile solo se il numero di comandi è limitato.
- Le stesse informazioni di sincronizzazione possono essere trasportate mediante linee a livello, a condizione di disporre di un segnale di clock nel sistema.
- Le informazioni a livelli sono campionate nell'intorno di uno dei fronti del segnale del clock, che viene trasportato sul bus mediante una linea apposita .

Il bus asincrono

Nel **bus** a funzionamento **asincrono** possiamo distinguere le seguenti **4 fasi**:

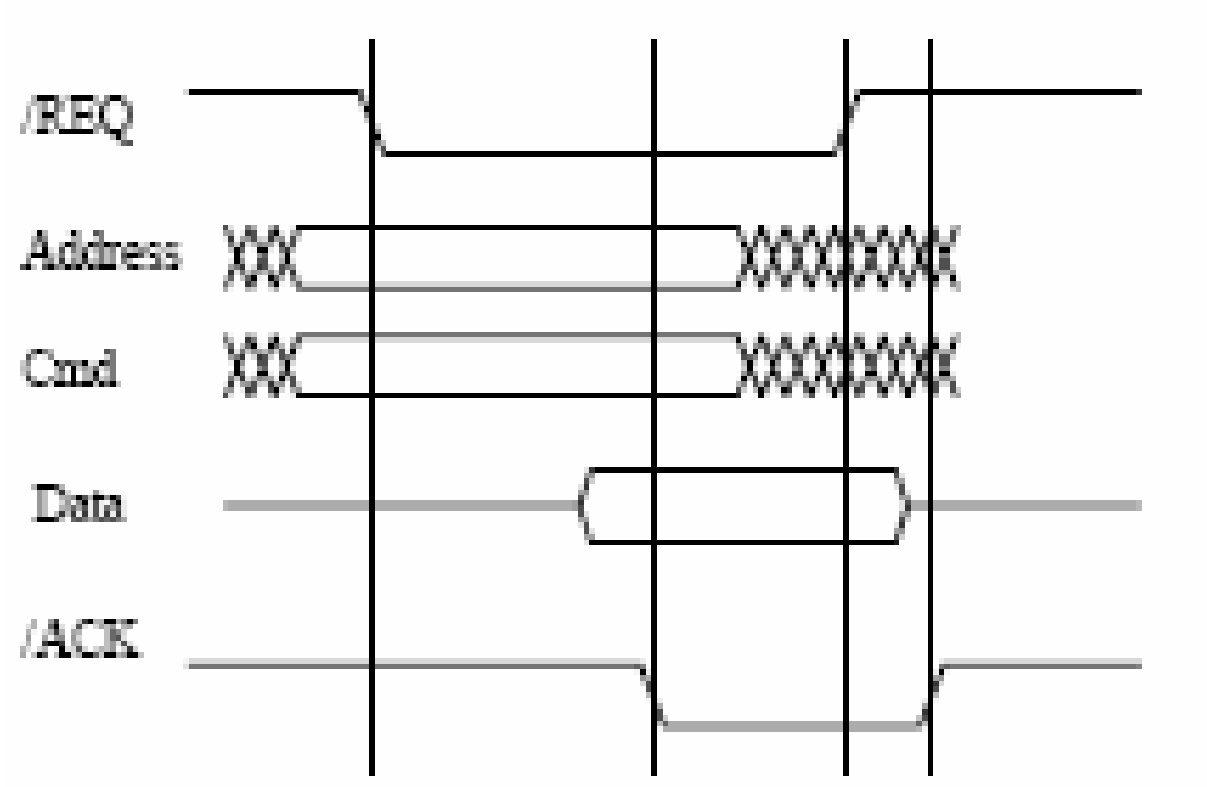
1) **il master richiede un trasferimento dati;**
2) **lo slave accetta il trasferimento e segnala di essere pronto ad eseguire l'operazione;**
3) **il master comunica di aver ricevuto il segnale dallo slave e di aver completato le sue operazioni;**
4) **lo slave comunica al master di aver ricevuto il segnale e completato il trasferimento.**

Bus asincrono: ciclo di lettura (1/2)

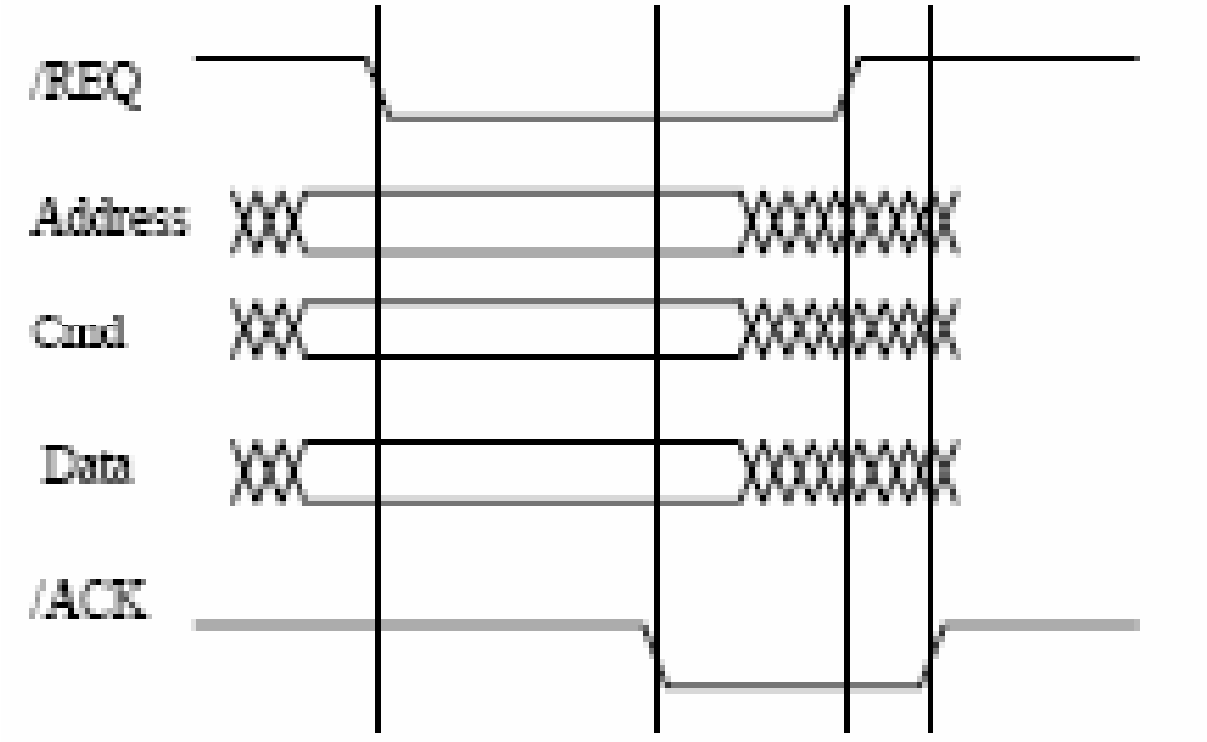
- Il **master** pilota le linee **/REQ**, **Address**, **Cmd**.
- Le linee **cmd** indicano il tipo di operazione (lettura/scrittura).
- Lo **slave** pilota le linee **/ACK** e **Data**.

1) **Il master mette sul bus l'indirizzo della locazione coinvolta nel trasferimento, specifica il tipo di operazione, e mette /REQ a zero.** Quando lo slave vede /REQ transire effettua l'operazione nel minor tempo possibile.
2) **Quando lo slave ha portato a termine l'operazione mette /ACK a zero.** Il master capisce che i dati sono disponibili e li memorizza.
3) **Il master nega /REQ ad indicare che adesso lo slave può rimuovere i dati dal bus.**
4) **Lo slave nega /ACK. Il ciclo è terminato.**

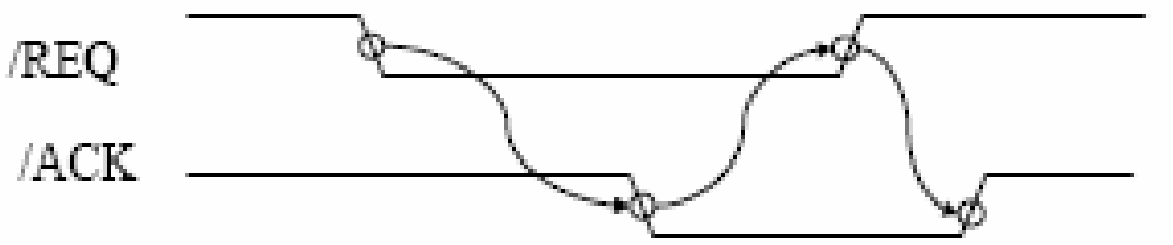
Bus asincrono: il ciclo di lettura (2/2)



Bus asincrono: il ciclo di scrittura



Bus asincrono



1. **/REQ viene attivato.**
2. **/ACK viene attivato in risposta a /REQ.**
3. **/REQ viene negato in risposta a /ACK.**
4. **/ACK viene negato in risposta alla negazione di /REQ.**

- Una segnalazione di questo tipo è detta ***full handshake***.
- Ogni evento è causato dall'evento precedente ed è indipendente dal tempo.

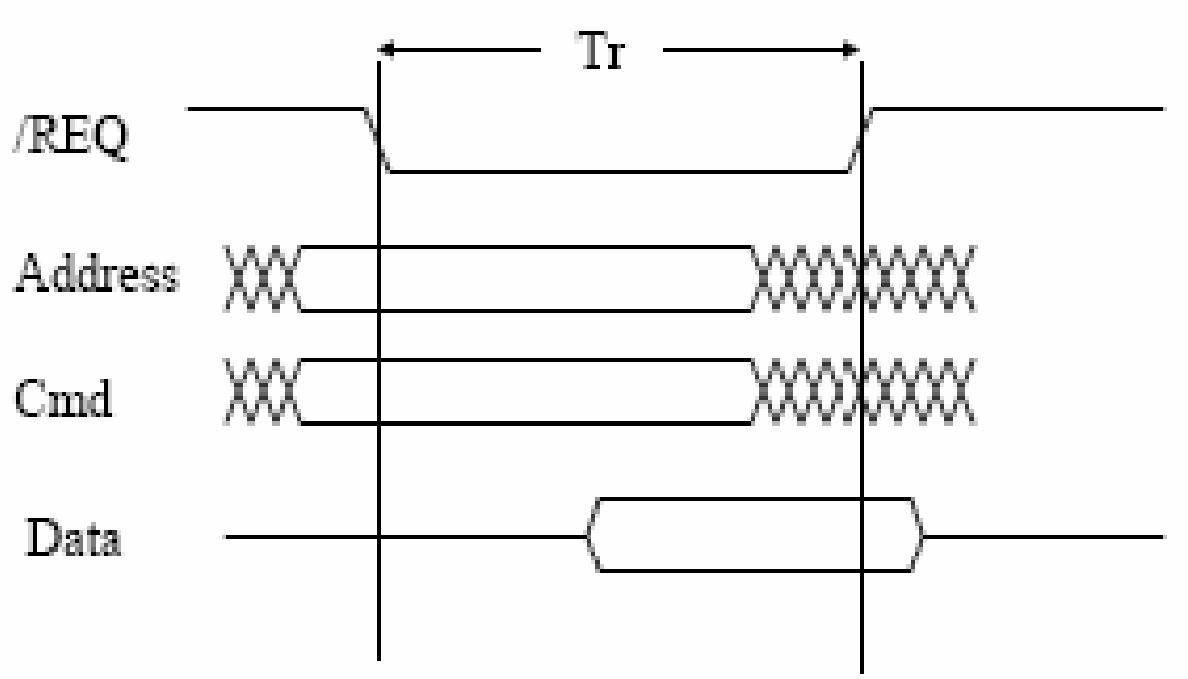
Il bus asincrono: vantaggi e svantaggi

- **Vantaggi:**
  - Flessibilità: la durata di una operazione è unicamente determinata dalla velocità della coppia master-slave.
- **Svantaggi:**
  - E' necessario inserire negli slave i circuiti necessari a rispondere opportunamente al protocollo.
  - Per completare una comunicazione sono sempre necessarie 4 azioni.

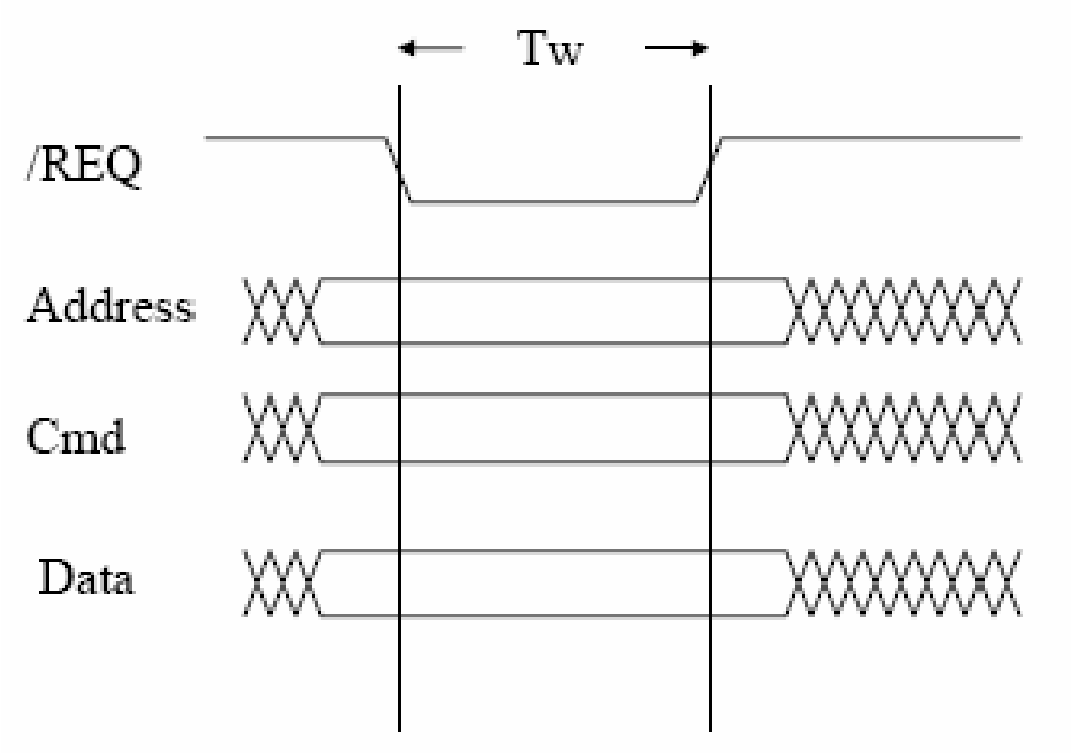
Il bus sincrono

- La durata delle fasi di una comunicazione è nota esattamente ad entrambi i partecipanti, e l'unica incognita è l'istante di inizio di una comunicazione.
- Una linea **/REQ** indica l'inizio di una comunicazione.

Il bus sincrono: ciclo di lettura



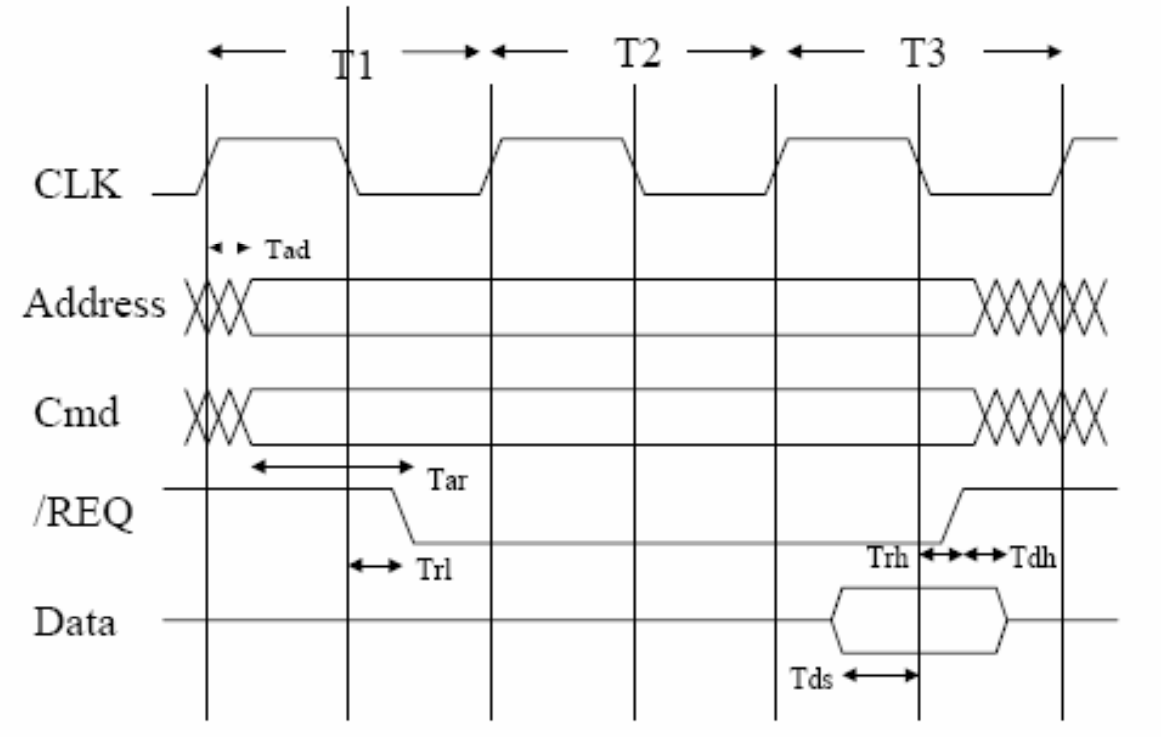
Il bus sincrono: ciclo di scrittura



Esempio di bus sincrono

- Una linea **CLK**, pilotata da un oscillatore, può essere utilizzata per sincronizzare i dispositivi sul bus.
- Il segnale presente su tale linea è una onda quadra, avente una frequenza generalmente compresa tra 5 Mhz e 100 Mhz.
- Supponiamo di avere un bus operante a 40 Mhz e quindi con un periodo pari a 25 nsec.
- Supponiamo inoltre che la memoria impieghi 40 nsec a fornire i dati in uscita (a partire dall'istante in cui vengono presentati sul bus gli indirizzi).

Temporizzazione del bus sincrono



Parametri della temporizzazione del bus sincrono

- **Tad** Intervallo di tempo tra il fronte in salita di **CLK** e l'istante in cui sono valide le linee degli indirizzi (e dei comandi).
- **Tar** Indica da quanto tempo sono stabili le linee degli indirizzi e dei comandi prima del fronte in discesa di **/REQ**.
- **Trl** Ritardo che intercorre tra il fronte in discesa di **CLK** e il fronte in discesa di **/REQ**.
- **Trh** Ritardo che intercorre tra il fronte in discesa di **CLK** e il fronte in salita di **/REQ**.
- **Tds** Tempo di setup per le linee dei dati prima del fronte in discesa di **CLK**.
- **Tdh** Indica l'intervallo di tempo che intercorre tra il fronte in salita di **/REQ** e la rimozione da parte dello slave dei dati dal bus.

Esempio di temporizzazione del bus sincrono

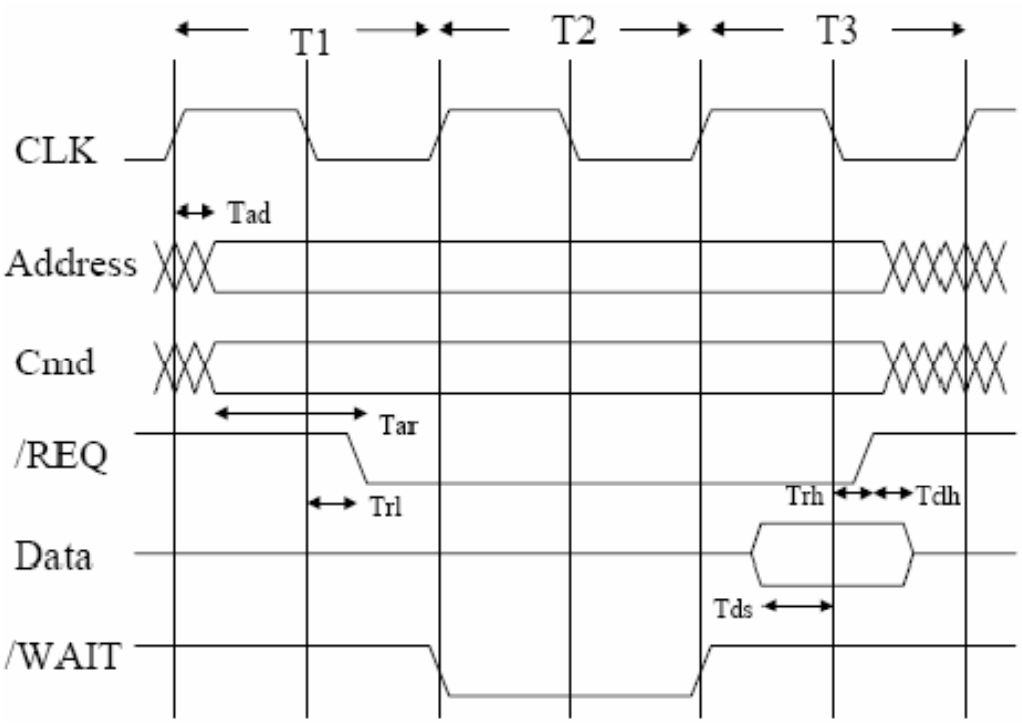
- In questo caso la memoria ha avuto a disposizione un tempo pari a

  2,5T - Tad - Tds

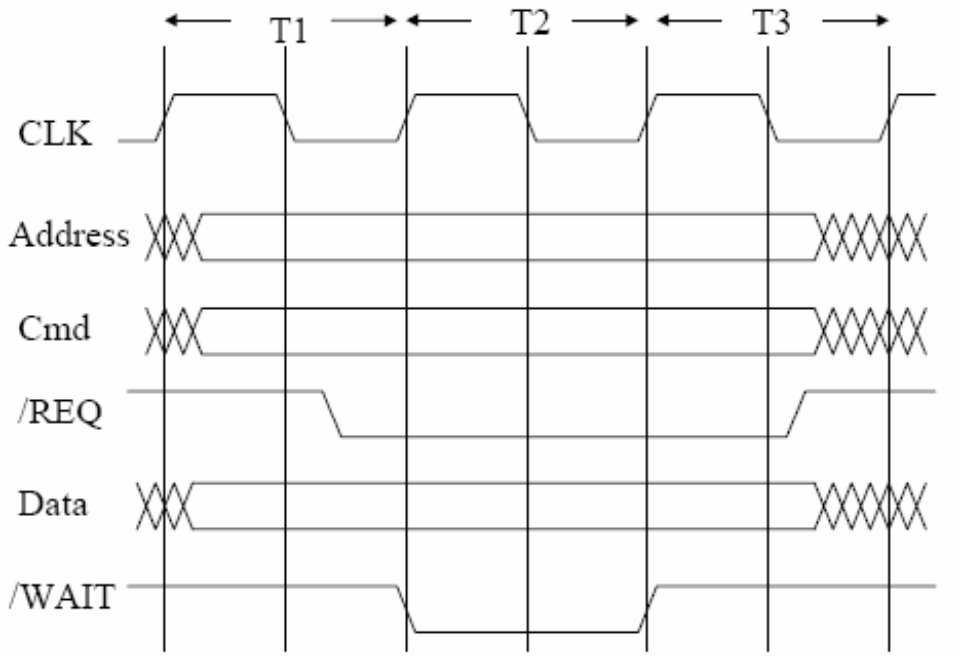
- Le specifiche del bus impongono dei vincoli sul valore di alcune costanti di tempo. Supponiamo ad es:
  - **Tad** <11 nsec **Tar** >6 nsec **Trl** <8 nsec,
  - **Trh** <8 nsec **Tds** >5 nsec **Tdh** > 3nsec.
- Tempo a disposizione della memoria:

  46,5 nsec

Protocollo semisincrono: il ciclo di lettura



Protocollo semisincrono: il ciclo di scrittura



Il bus sincrono: vantaggi e svantaggi

- **Vantaggi:**
  - La realizzazione degli slave può risultare semplificata
  - E' particolarmente vantaggioso quando la durata di una operazione è fissa e nota a priori (posso eliminare la linea wait).
- **Svantaggi:**
  - La durata di una operazione di comunicazione deve necessariamente avere una durata pari ad un numero intero di cicli.

Arbitraggio del bus (1/2)

- La presenza su un bus di più unità master richiede la presenza di un meccanismo in grado di regolare l'accesso al bus stesso.
- Tale meccanismo può essere implementato da una unità detta ***arbitro***, che accetta su linee dedicate le richieste di accesso al bus da parte dei master e che realizza la mutua esclusione tra gli stessi.
- **La funzione dell'arbitro è di scegliere, tra uno o più moduli che effettuano una richiesta, a quale deve essere concesso l'accesso al bus.**

Arbitraggio del bus (2/2)

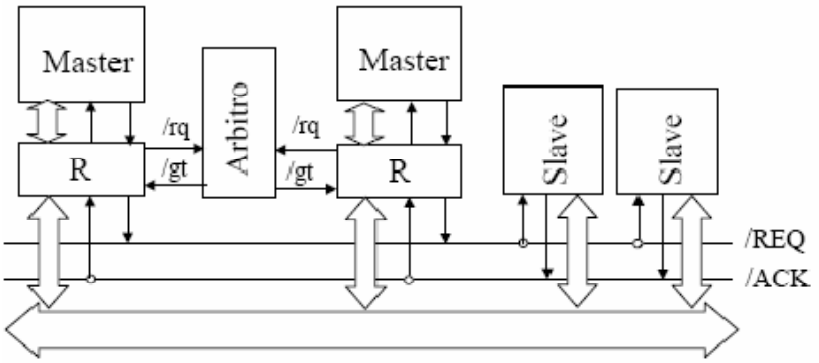
Le due politiche di scelta più comuni sono:

- a priorità fissa: alcuni moduli sono sempre privilegiati rispetto ad altri;
- a priorità variabile: nessun modulo è staticamente privilegiato.

L'arbitro può essere realizzato come

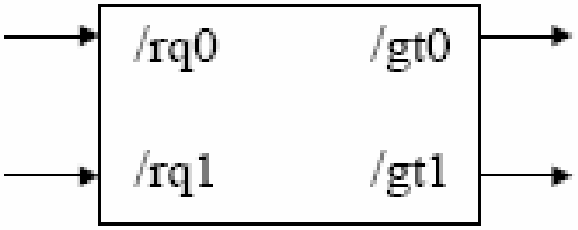
- una unità monolitica alla quale arrivano tutte le richieste / un insieme di blocchi elementari distribuiti tra i vari moduli del sistema;
- un sistema asincrono / un sistema sincronizzato.

Schema arbitro/bus



- E' necessario introdurre dei moduli addizionali tra i master ed il bus.
- La rete R gestisce l'interazione con l'arbitro e separa le linee del master da quelle del bus quando non lo sta utilizzando:
  - **/rq** Richiesta di utilizzo del bus,
  - **/gt** Autorizzazione all'utilizzo del bus (*grant*).

Arbitro asincrono elementare



- I due ingressi ricevono le richieste dei (due) master.
- Le uscite notificano ai master se hanno il diritto di utilizzare il bus.
- **L'arbitro è per sua natura pilotato in modo non corretto** (possono arrivare richieste contemporanee).

Tabella di stato dell'arbitro asincrono elementare

/rq0 /rq1	00	01	11	10	/gt0	/gt1
W0	W1	G0	(W0)	G1	1	1
G0	(G0)	(G0)	W0	W0	0	1
G1	(G1)	W0	W0	(G1)	1	0
W1	W0	-	-	-	0	0

Utilizziamo /gt0 e /gt1 sia come variabili di uscita che come variabili di stato

Funzionamento arbitro asincrono element.

1) Se ***nessuna delle richieste*** è attiva l'arbitro resta nello stato W0.
2) Se l'arbitro riceve ***una sola richiesta*** si salta nello stato di uscita corrispondente (G0 o G1).
3) Se arrivano ***due richieste contemporanee*** si fa in modo di instaurare una situazione di oscillazione tra gli stati W0 e W1. Tale oscillazione si risolverà in favore di uno degli stati G0 o G1 dopo breve tempo a causa dei diversi tempi di propagazione.

/rq0 /rq1	00	01	11	10	/gt0	/gt1
W0	W1	G0	(W0)	G1	1	1
G0	(G0)	(G0)	W0	W0	0	1
G1	(G1)	W0	W0	(G1)	1	0
W1	W0	-	-	-	0	0

4) Al rilascio della richiesta *da parte di una unità* si rientra nello stato W0, anche se nel frattempo è giunta una altra richiesta (la tabella di flusso non è più normale ma si semplifica la realizzazione della rete).

Sintesi arbitro asincrono elementare

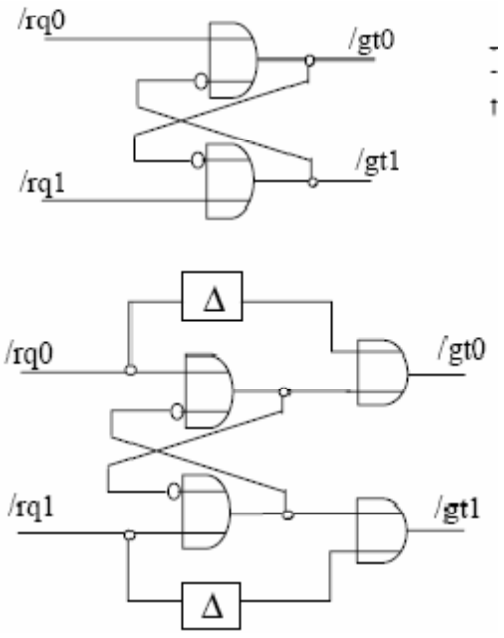
/gt0 /gt1 \ /rq0 /rq1	00	01	11	10
00	1	-	-	-
01	0	0	1	1
11	0	0	1	1
10	1	1	1	1

$$/gt0 = \overline{/gt1} + /rq0$$

/gt0 /gt1 \ /rq0 /rq1	00	01	11	10
00	1	-	-	-
01	1	1	1	1
11	0	1	1	0
10	0	1	1	0

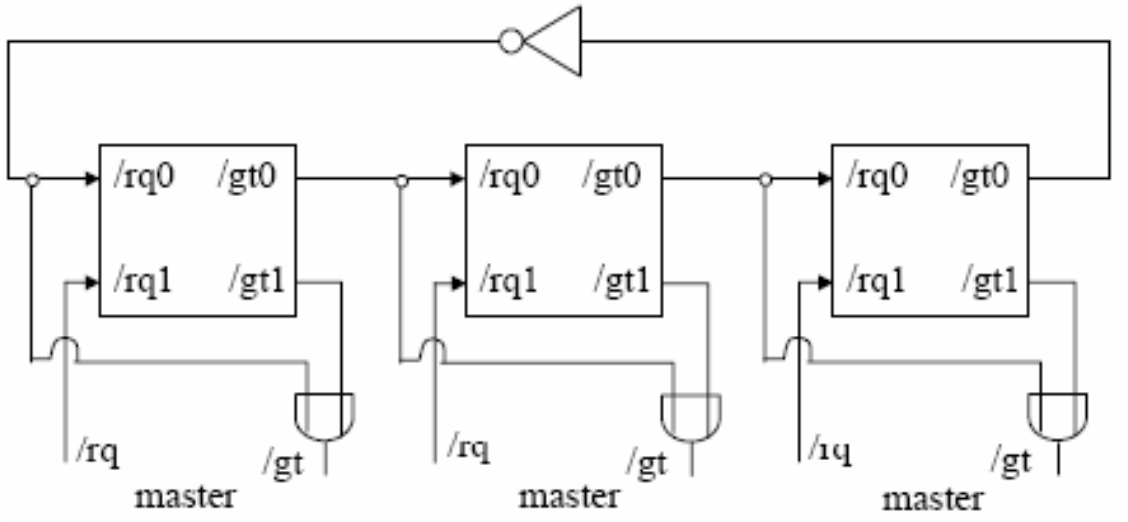
$$/gt1 = \overline{/gt0} + /rq1$$

Sintesi circuitale arbitro asincrono elementare



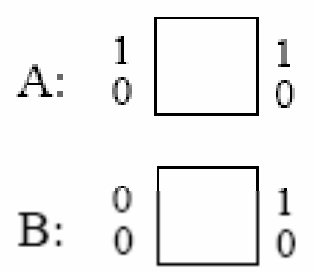
- Nel caso di richieste concorrenti i transitori si propagano in uscita
- Le uscite vengono filtrate mediante una ulteriore porta che diviene attiva solo dopo un tempo **Δ** dall'arrivo delle richieste.
- Variando **Δ** è possibile ridurre a piacere la probabilità che i transitori siano ancora attivi

Arbitro a priorità rotante (1/2)



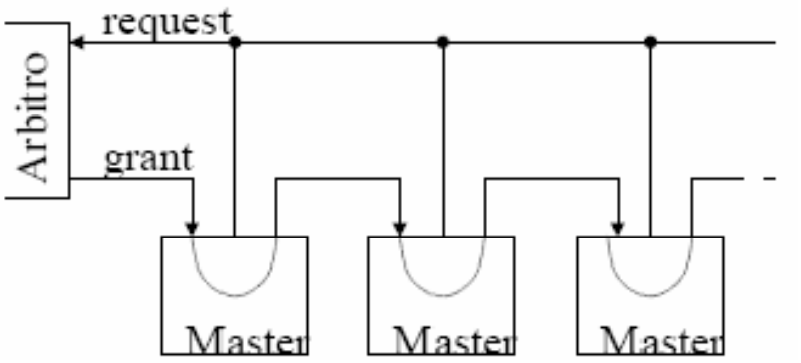
Mediante l'interconnessione di arbitri asincroni elementari è possibile realizzare un arbitro ad n ingressi (utile soprattutto quando il numero di master non è noto a priori).

Arbitro a priorità rotante (2/2)



- Solo uno dei **/gt** può essere al livello basso, e ciò accade quando il singolo modulo di arbitraggio passa da una configurazione A ad una configurazione B.
- In questa situazione infatti l'OR che fornisce il segnale di *grant* al master avrà in ingresso due zeri, mentre l'1 passato al modulo a destra eviterà che altri master ricevano il segnale di *grant*.
- L'anello superiore è instabile (invertitore + ritardo) ed oscilla fino a quando non viene concessa la risorsa ad uno dei master.

Arbitro a priorità fissa



- Quando l'arbitro vede arrivare una richiesta di utilizzo del bus, attiva la linea di grant.
- Quando un master vede attivarsi la linea di grant:
  - se aveva fatto lui la richiesta, non propaga il segnale di grant al master alla sua destra ed utilizza il bus;
  - se non aveva fatto lui la richiesta propaga il segnale di grant al master alla sua destra.
- Il master più vicino all'arbitro ha la priorità più alta.

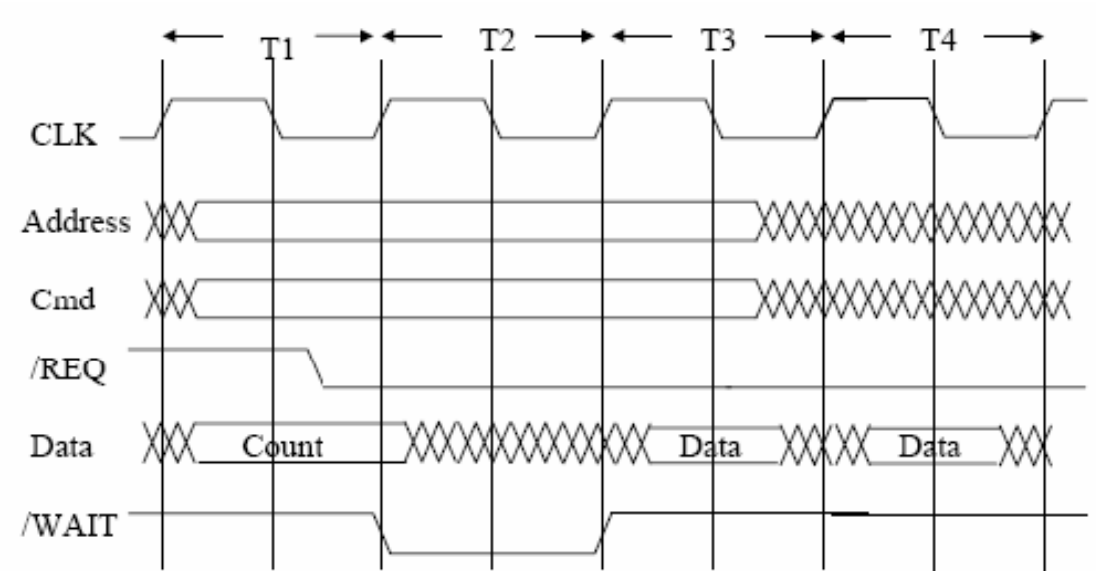
Set di operazioni

- Esistono altri cicli di bus oltre a quelli elementari di lettura/scrittura ad es:
  - trasferimento di blocco,
  - read-modify-write.

Trasferimento di blocco (1/2)

- Consente di trasferire il contenuto di più locazioni memoria con un singolo ciclo di bus
- Ad esempio nel caso di una lettura di blocco il master :
  - indica che si tratta di una operazione di trasferimento di blocco (lettura/scrittura) utilizzando le linee **`Cmd`**;
  - dice allo slave quanti dati devono essere trasferiti utilizzando le linee dati;
  - pone l'indirizzo di partenza sulle linee **`Address`**.
- Lo slave risponde fornendo i dati ad ogni ciclo (se ne è in grado).

Trasferimento di blocco (2/2)



Il trasferimento di un blocco è vantaggioso rispetto al trasferimento della stessa quantità di dati mediante più trasferimenti elementari.

Ready-modify-write

- Le seguenti operazioni vengono eseguite in un singolo ciclo di bus:
  - un dato viene trasferito all'interno della CPU (*read*);
  - la CPU esegue un test sul dato ed eventualmente lo modifica;
  - il dato, eventualmente modificato, viene riscritto in memoria.
- Questo ciclo viene utilizzato nei sistemi con più CPU per garantire che una sola di esse possa accedere ad una risorsa condivisa.

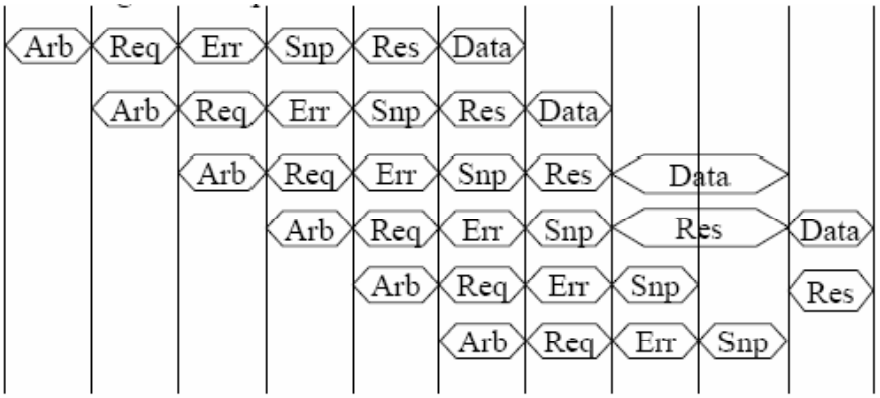
Pipelined bus (1/2)

In una operazione di comunicazione possiamo distinguere le seguenti fasi:

- *arbitraggio*: determina il prossimo utilizzatore del bus;
- *richiesta*: l'indirizzo ed il comando vengono posti sul bus e viene effettuata la richiesta;
- *segnalazione di errori*: lo slave riporta eventuali errori;
- *snoop phase*: utilizzata nei sistemi multiprocessore (coerenza delle informazioni);
- *risposta*: lo slave segnala che i dati sono pronti;
- *dati*: i dati vengono trasferiti.

Pipelined bus (2/2)

Nei bus con architettura a pipeline più operazioni possono avere luogo contemporaneamente:



E' necessario che ogni fase utilizzi linee del bus differenti, in modo tale che ogni fase (arb, req, …) sia indipendente dalle altre.