Ampiezza e temporizzazione del BUS

Aniello Minutolo

Ampiezza del bus

Sommario

1. Ampiezza del bus
2. Bus sincroni
3. Bus asincroni

Relazione tra linee d'indirizzo e memoria indirizzabile

Nella progettazione dei bus il parametro più scontato da considerare è la **Ampiezza del bus.**

Maggiore è il numero di linee d’indirizzo di un bus, maggiore sarà la quantità di memoria che la CPU potrà indirizzare direttamente

- se un bus ha n linee d’indirizzi, una CPU può indirizzare 2^n diverse locazioni di memoria.

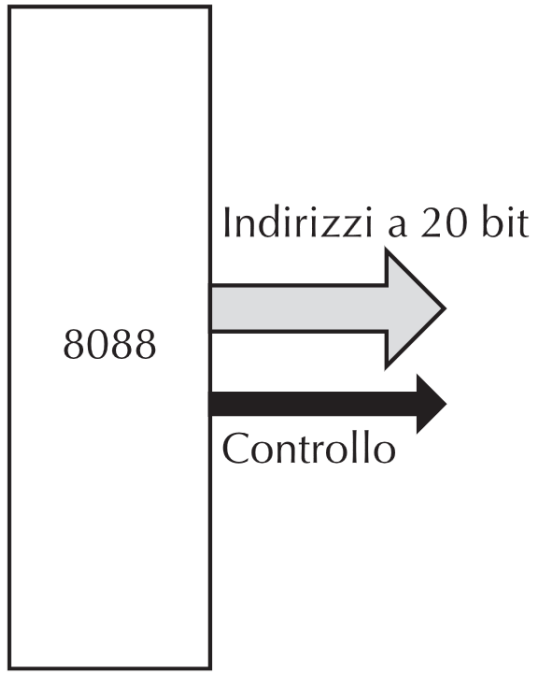
Un incremento delle linee d'indirizzo amplia lo spazio di memoria ma bus più larghi aumentano complessità e ingombro

- richiedono un numero maggiore di fili, connettori più grandi e maggior superficie sulla scheda madre.

Evoluzione storica e problemi di retrocompatibilità

Il PC IBM originario conteneva una CPU 8088 e un bus con indirizzi a 20 bit

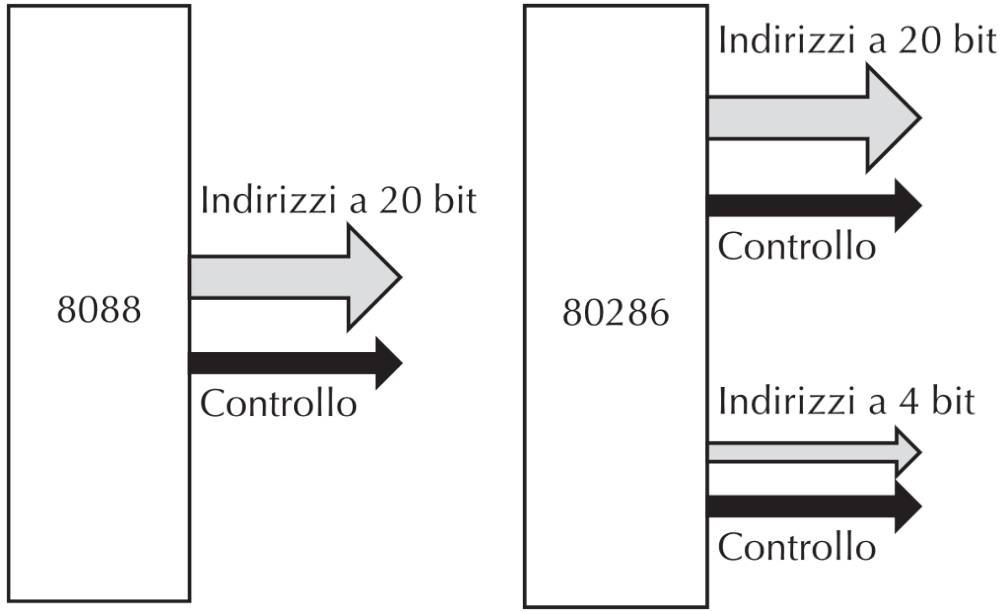
- poteva indirizzare soltanto 1 MB di memoria.



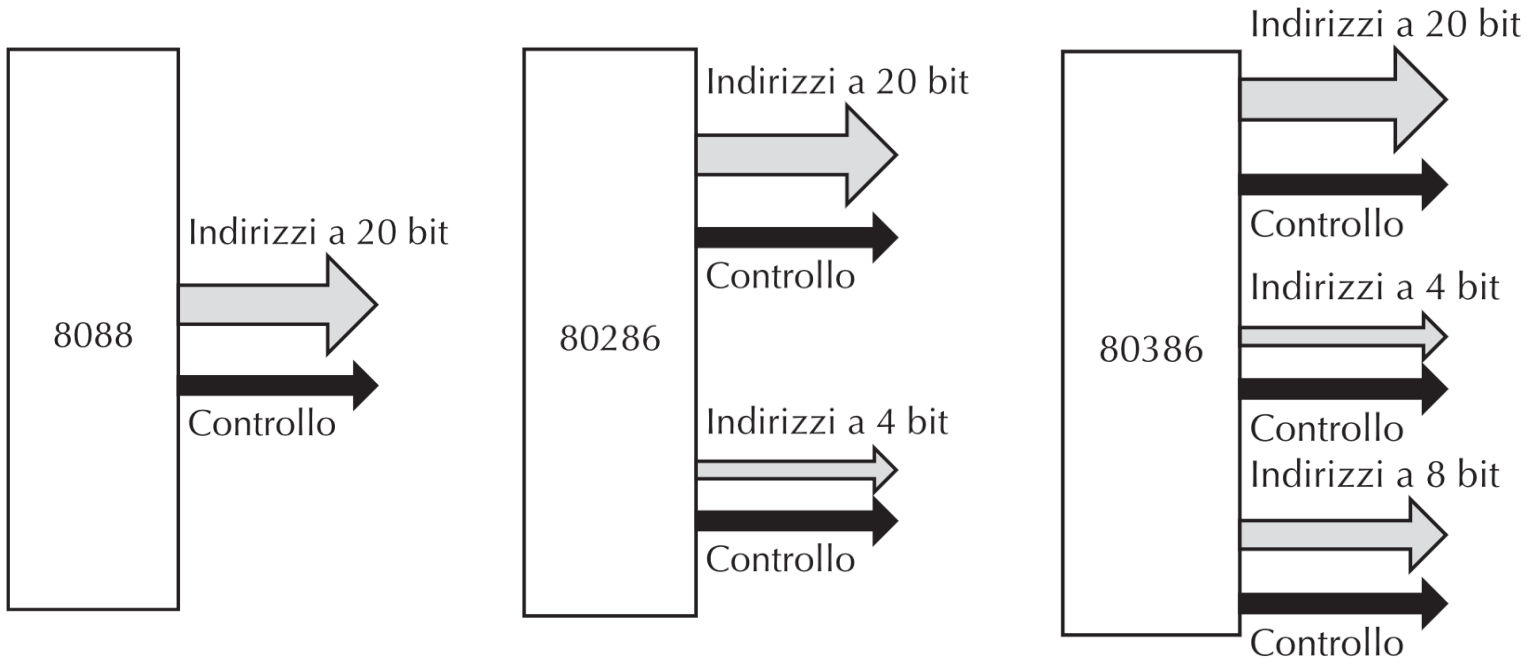
Evoluzione storica e problemi di retrocompatibilità

Con la CPU successiva, l'80286, Intel decise di aumentare lo spazio degli indirizzi a 16 MB

- questo aumento richiese l'aggiunta di quattro linee di bus, senza modificare le 20 precedenti per ragioni di retrocompatibilità;
- purtroppo fu necessario aggiungere anche alcune nuove linee di controllo per gestire le nuove linee d'indirizzo.



Evoluzione storica e problemi di retrocompatibilità



- Quando apparve l'80386 furono aggiunte altre otto linee d'indirizzo oltre alle linee di controllo aggiuntive;
- il progetto risultante (il bus EISA) è molto meno ordinato di come sarebbe potuto essere se si fosse partiti da zero avendo a disposizione fin dall'inizio tutte e 32 le linee.

Impatto sui costi

Ogni linea aggiuntiva incrementa i **costi di produzione** del sistema e riduce la compatibilità con componenti esistenti.

Ottimizzare il rapporto costo-prestazioni è cruciale, come nel caso dei PC IBM originari con aggiornamenti incrementali

- ad esempio, un sistema con 64 linee d'indirizzi e 2^{32} byte di memoria costerà più di uno con 32 linee d'indirizzi e 2^{32} byte di memoria, ma la memoria del secondo sistema non sarà ulteriormente espandibile;
- molti progettisti di sistemi tendono a essere poco lungimiranti.

Strategie per aumentare la larghezza di banda

Esistono due modi per aumentare la **larghezza di banda** dei dati su un bus

- diminuire il periodo di clock del bus (più trasferimenti/s);
- aumentare la larghezza dei dati del bus.

Strategie per aumentare la larghezza di banda

- Un’altra possibilità è quella di aumentare la velocità del bus, anche se ciò pone alcune difficoltà, in quanto i segnali su linee distinte viaggiano infatti a velocità leggermente diverse.
- Questo problema è conosciuto come **disallineamento del bus**, e più il bus è veloce più questo problema diventa marcato.

Strategie per aumentare la larghezza di banda

Un altro problema che sorge nel caso in cui si intenda velocizzare il bus è la **perdita della retrocompatibilità**

- schede progettate per bus più lenti non funzioneranno con il nuovo bus.

Per questa ragione l'approccio più utilizzato è quello di aggiungere nuove linee di dati.

BUS multiplexato

Per aggirare il problema di bus troppo ampi a volte i progettisti optano per un **bus multiplexato**

- invece di tenere separate le linee d'indirizzo e quelle dei dati, si utilizza un certo numero di linee, per esempio 32, per entrambi;
- all'inizio di un'operazione sul bus le linee sono utilizzate per gli indirizzi, mentre in seguito vengono impiegate per i dati.

L'uso del multiplexing riduce l'ampiezza del bus (e quindi il costo), ma rende il **sistema più lento**

Bus sincroni

Definizione e caratteristiche principali

Un **Bus sincrono** utilizza una linea pilotata da un oscillatore a cristalli per coordinare tutte le operazioni, con cicli di durata fissa multipli del periodo di clock

- su questa linea un segnale consiste in un'onda quadra con frequenza generalmente compresa tra 5 e 133 MHz.

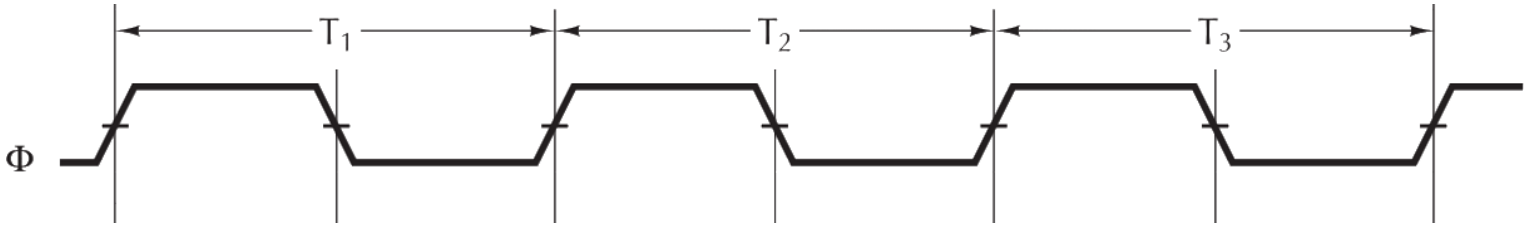
Il clock impone dei **vincoli temporali** a tutte le operazioni sul bus

- ogni **transazione sul bus** (es. lettura/scrittura di dati tra CPU e memoria o periferiche) richiede un numero intero di questi cicli, detti cicli di bus, per essere completata;
- i **cicli di bus** sono sempre un multiplo esatto del **periodo di clock**, garantendo che tutte le operazioni sul bus avvengano in modo sincronizzato e prevedibile;
    - ad esempio, se per leggere una parola dalla memoria sono richiesti tre cicli di bus, vuol dire che l'operazione impiega tre **periodi** del clock.

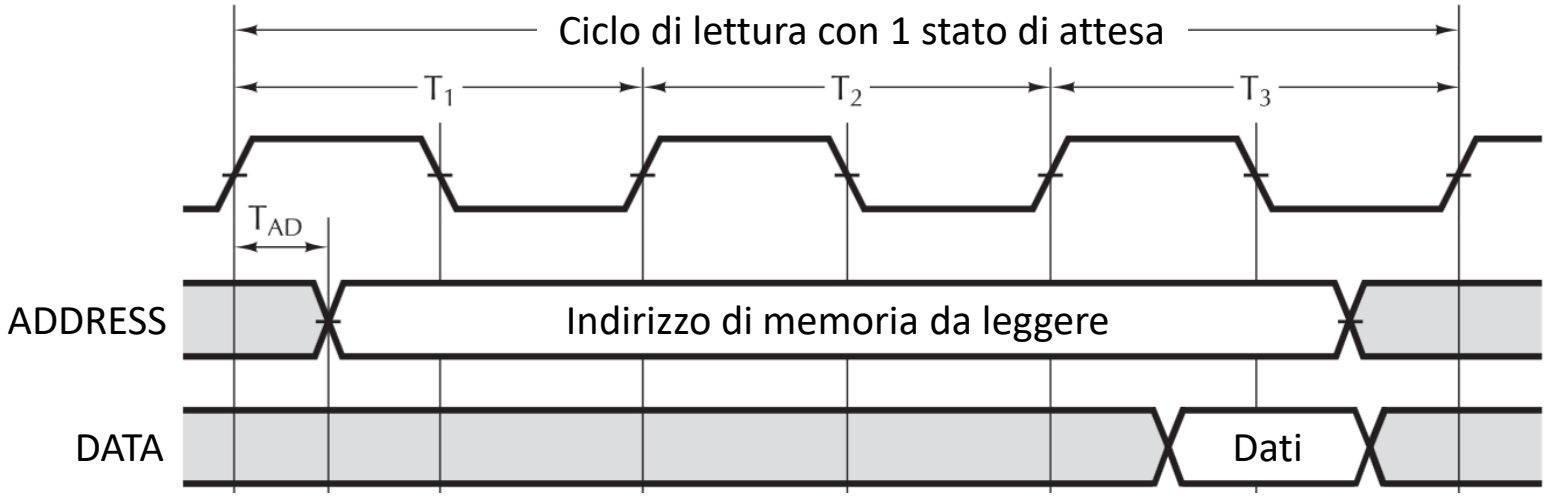
Esempio di temporizzazione di un bus sincrono

Come esempio del funzionamento di un bus sincrono, consideriamo un clock a 100 MHz, che fornisce un ciclo di bus di 10 ns

- i fronti di salita e di discesa non sono disegnati verticalmente in quanto nessun segnale elettrico può cambiare il proprio valore in un tempo nullo;
- assumiamo ad esempio che il segnale impieghi 1 ns per cambiare valore.



Esempio di temporizzazione di un bus sincrono



T_{AD} (Address Delay) è il ritardo dell'output dell'indirizzo, cioè il tempo tra il fronte di salita del ciclo di clock T_1 e il momento in cui vengono impostate le linee d'indirizzo

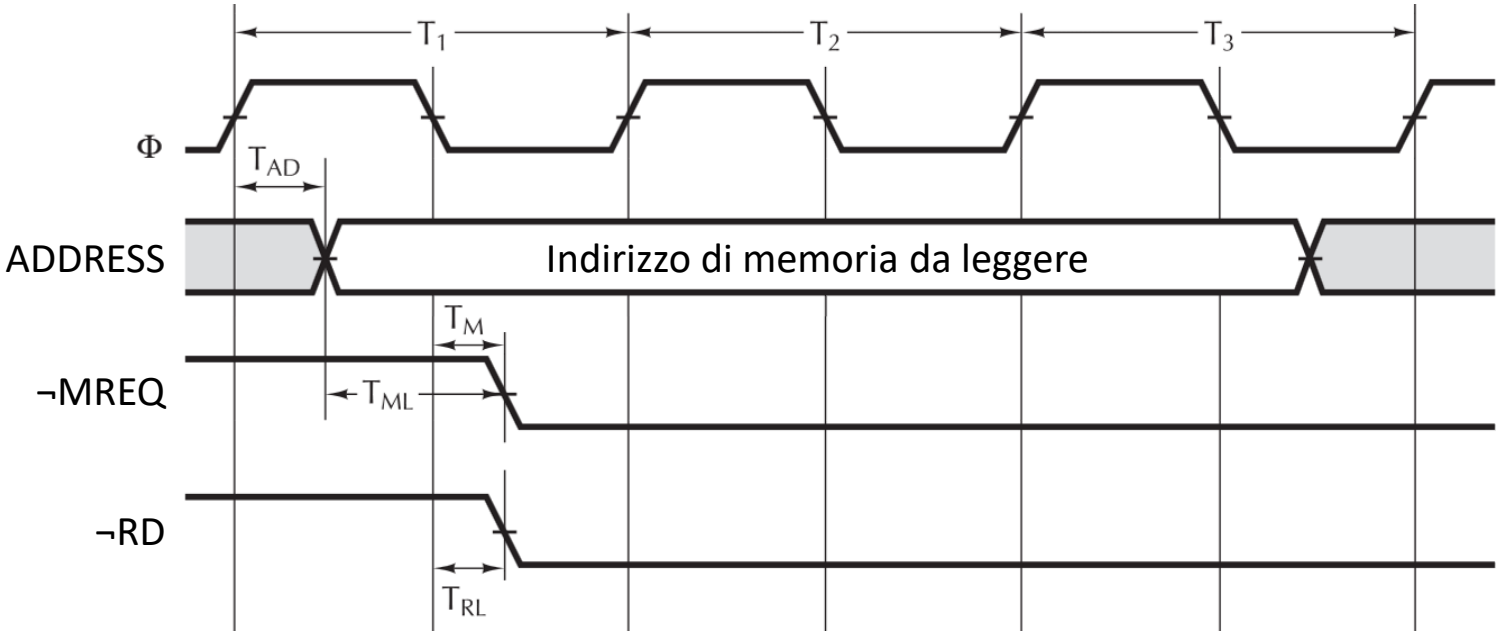
Ipotizziamo di voler effettuare un'operazione di lettura dalla memoria

- l'operazione inizia sul fronte di salita del ciclo T_1, in corrispondenza del quale la CPU fornisce l'indirizzo della parola sulle linee d'indirizzo;
- la disponibilità sulle linee dell'indirizzo da leggere viene rappresentata in figura tramite due linee che si incrociano.

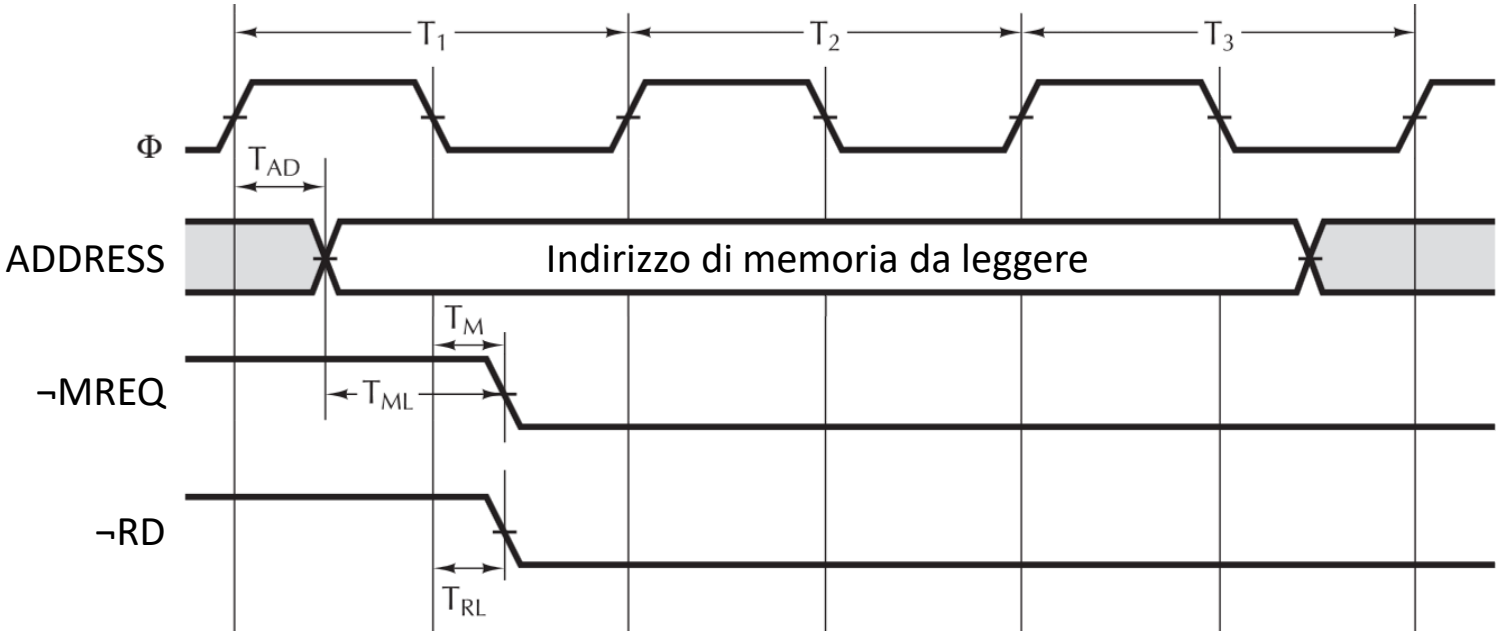
Esempio di temporizzazione di un bus sincrono

Dopo che le linee d'indirizzo si sono stabilizzate sui nuovi valori, vengono asseriti i segnali **MREQ** e **RD**

- **MREQ** viene asserito per indicare che si sta per accedere alla memoria (invece che a un dispositivo di I/O).
- **RD** viene asserito per le operazioni di lettura e negato per la scrittura.



Esempio di temporizzazione di un bus sincrono

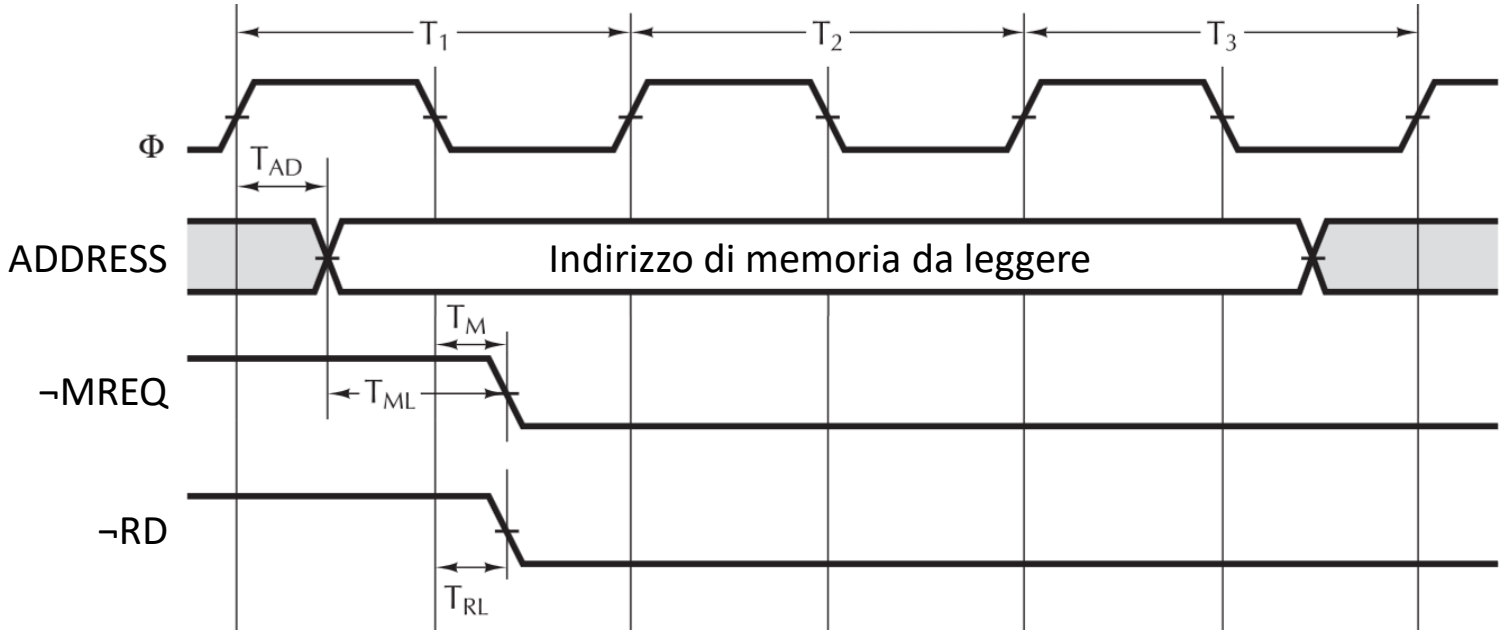


T_{ML} è il tempo minimo che la CPU attende, dopo aver impostato l'indirizzo, prima di asserire il segnale MREQ

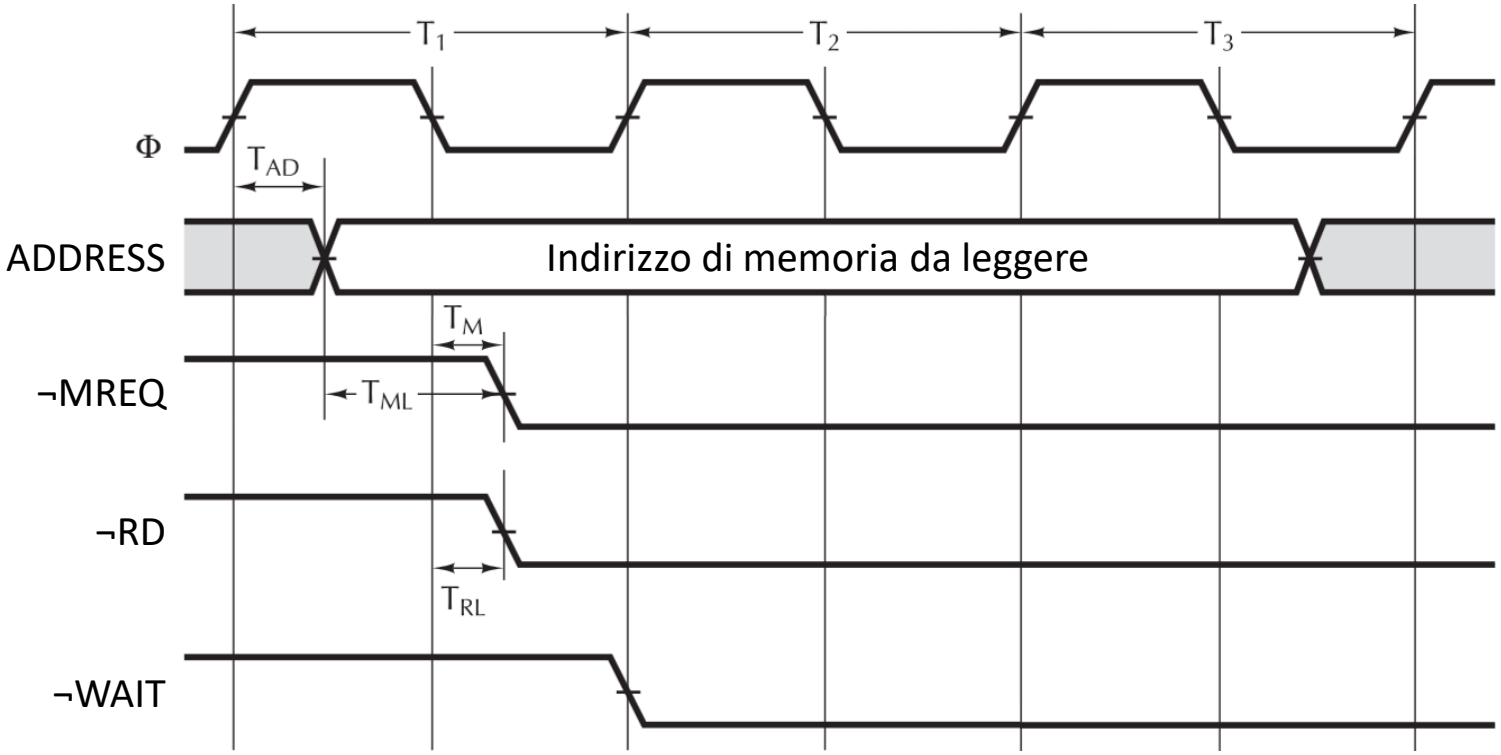
- la memoria può sfruttare questo intervallo temporale per svolgere eventuali sue operazioni di setup, ad esempio per la selezione del particolare chip sulla memoria.

Esempio di temporizzazione di un bus sincrono

I vincoli su $\mathbf{T_M}$ e $\mathbf{T_{RL}}$ definiscono il massimo ritardo dei segnali MREQ e RD rispetto al fronte di discesa del clock in T_1



Esempio di temporizzazione di un bus sincrono

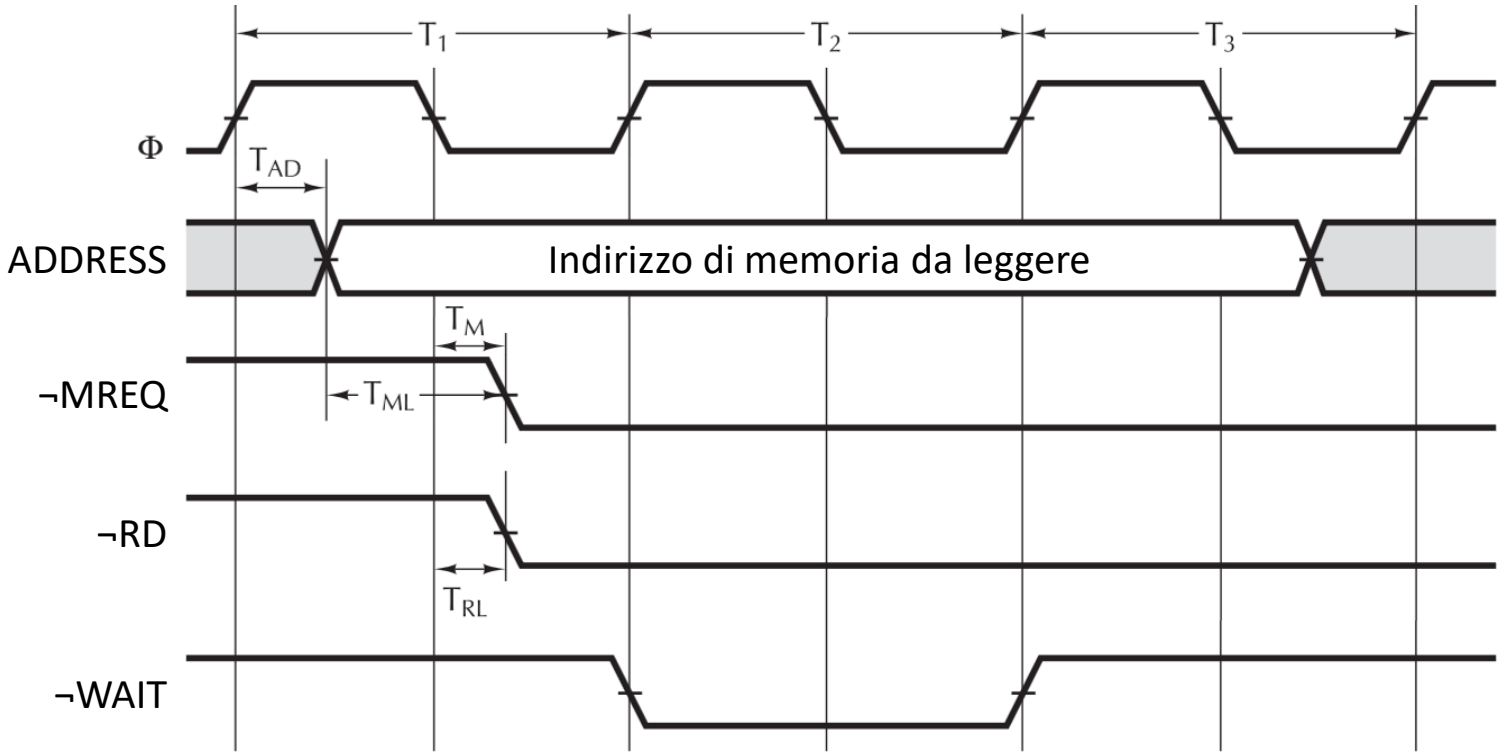


- La memoria asserisce la linea **WAIT** al fronte di salita del clock in T_2 per segnalare alla CPU di non aspettarla perché non può fornire i dati.
- Nell'ipotesi che la memoria impieghi 15 ns (dopo che l'indirizzo è stabile) per effettuare la lettura, non può fornire i dati richiesti durante T_2.

Esempio di temporizzazione di un bus sincrono

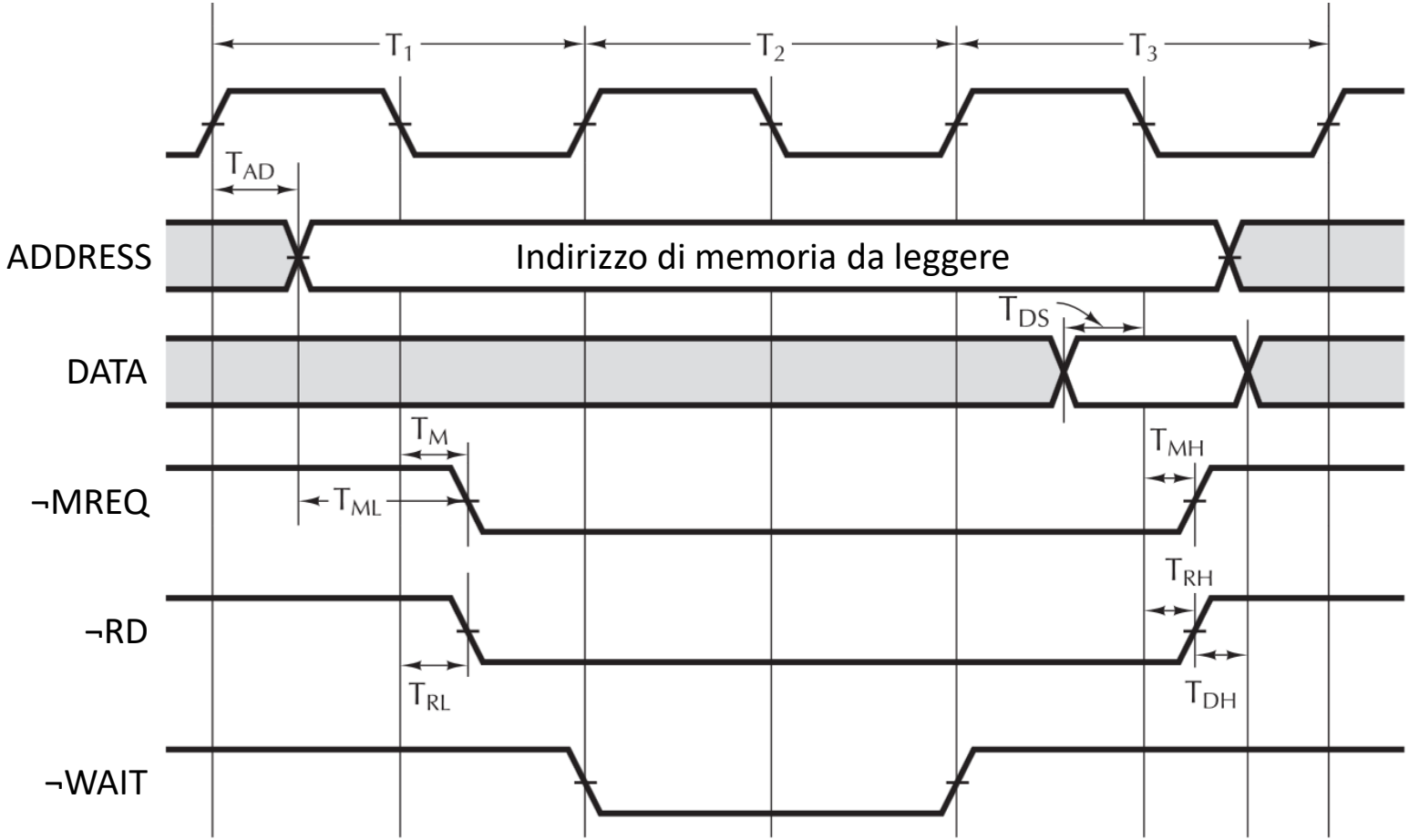
Asserendo il segnale **WAIT** si inseriscono alcuni stati di attesa finché la memoria non completi l'operazione e neghi il segnale **WAIT**

- nell'ipotesi che la memoria impieghi 15 ns (dopo che l'indirizzo è stabile) per effettuare la lettura, i dati saranno disponibili nel corso del ciclo T_3.



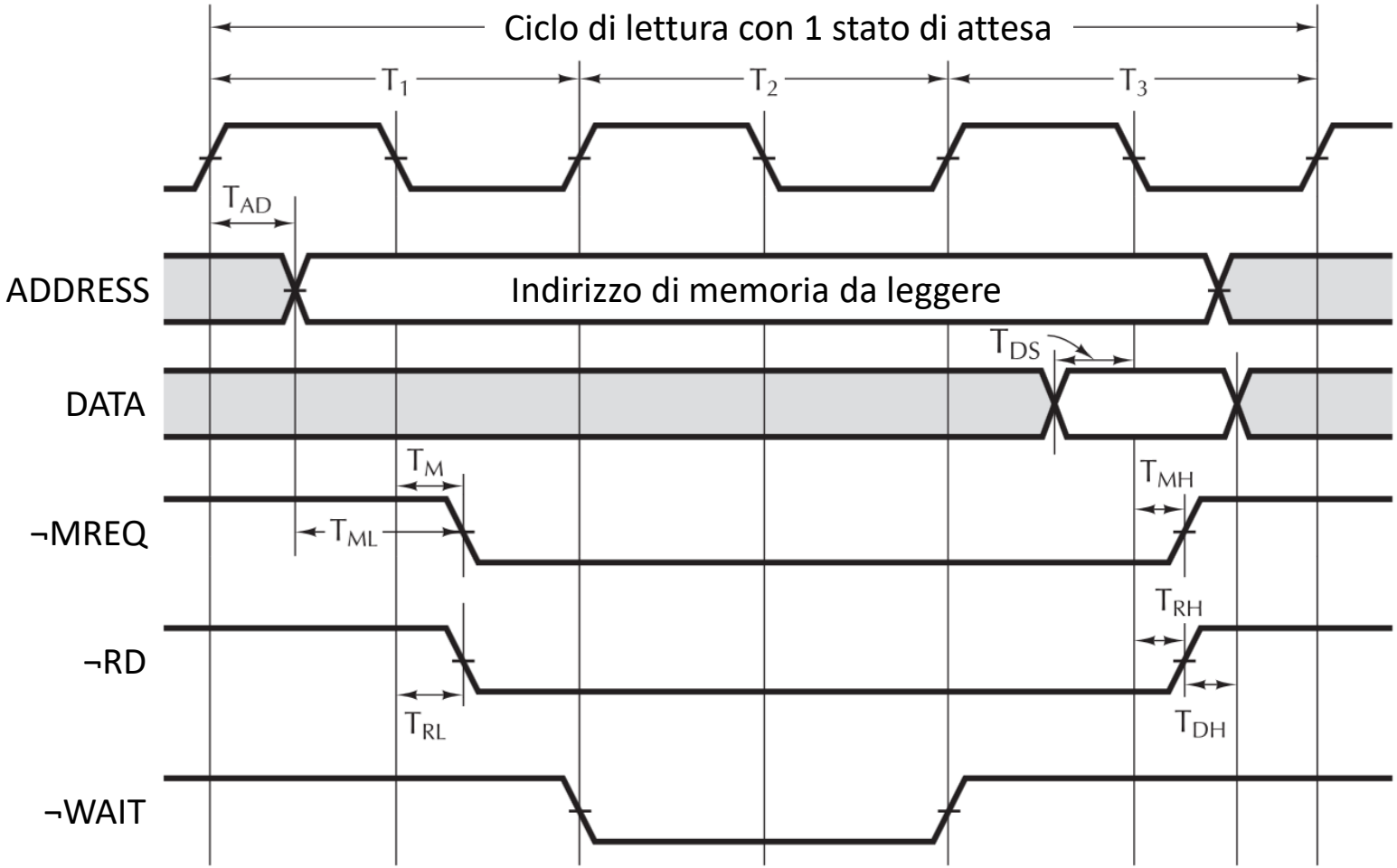
Esempio di temporizzazione di un bus sincrono

- Durante la prima metà di T_3, la memoria mette i dati sulle linee apposite

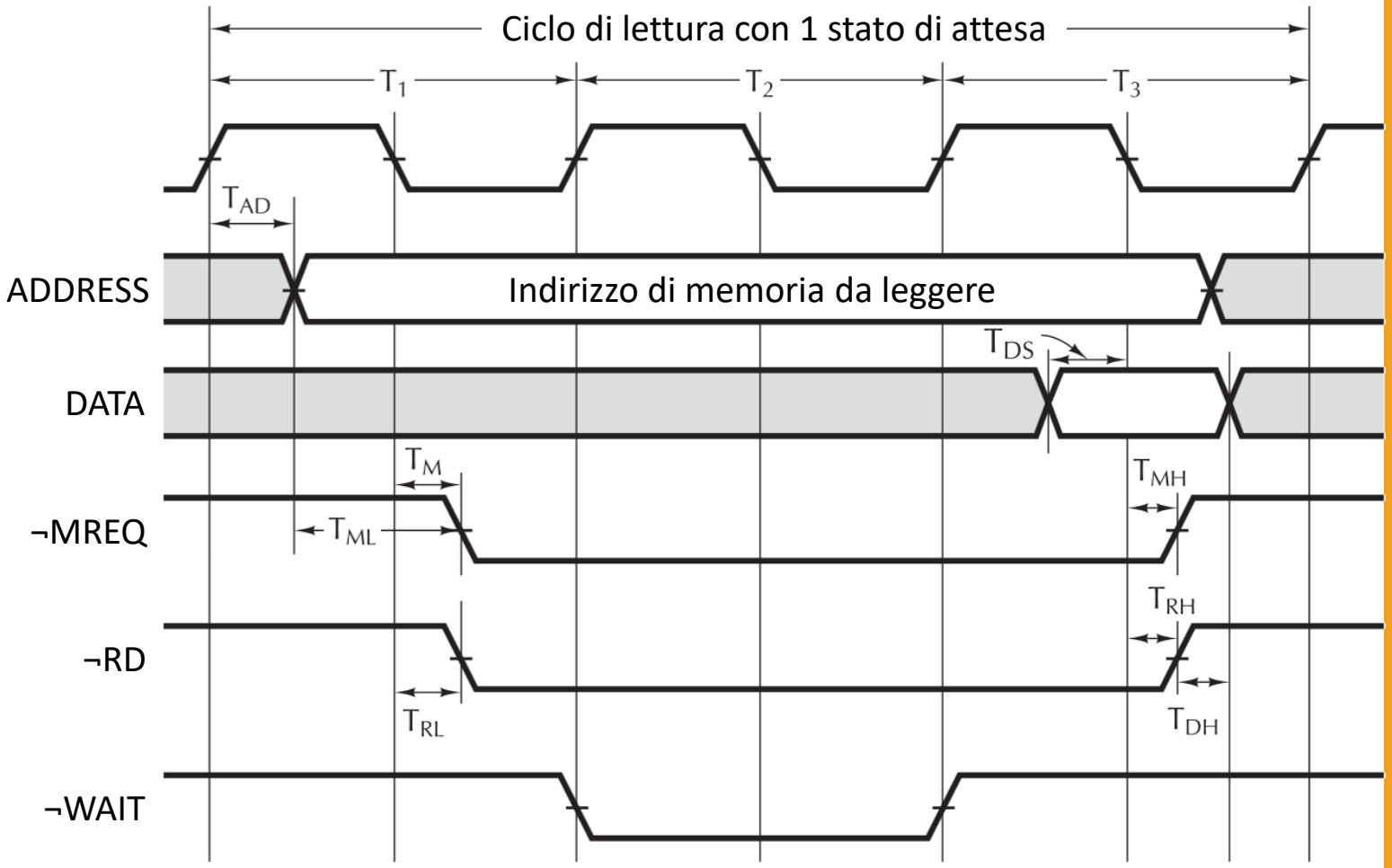


Esempio di temporizzazione di un bus sincrono

- $\mathbf{T_{DS}}$ è il tempo minimo di presenza dei dati prima della discesa di T_3
- serve per dare il tempo alle linee di stabilizzarsi prima che la CPU legga



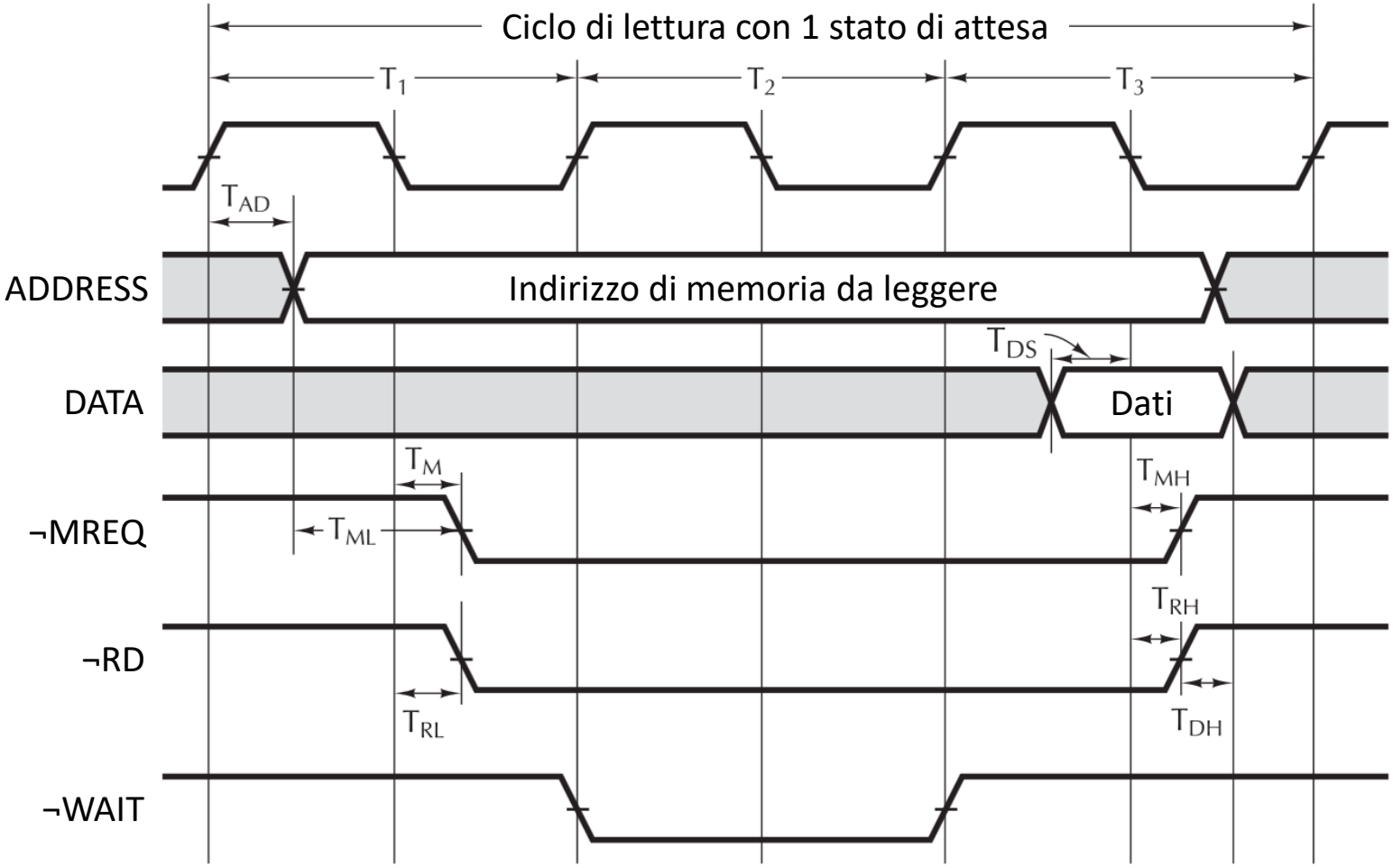
Esempio di temporizzazione di un bus sincrono



Nel fronte di discesa di T_3 la CPU consulta (cioè legge) le linee dei dati dovendo leggere i dati la CPU nega MREQ e RD.

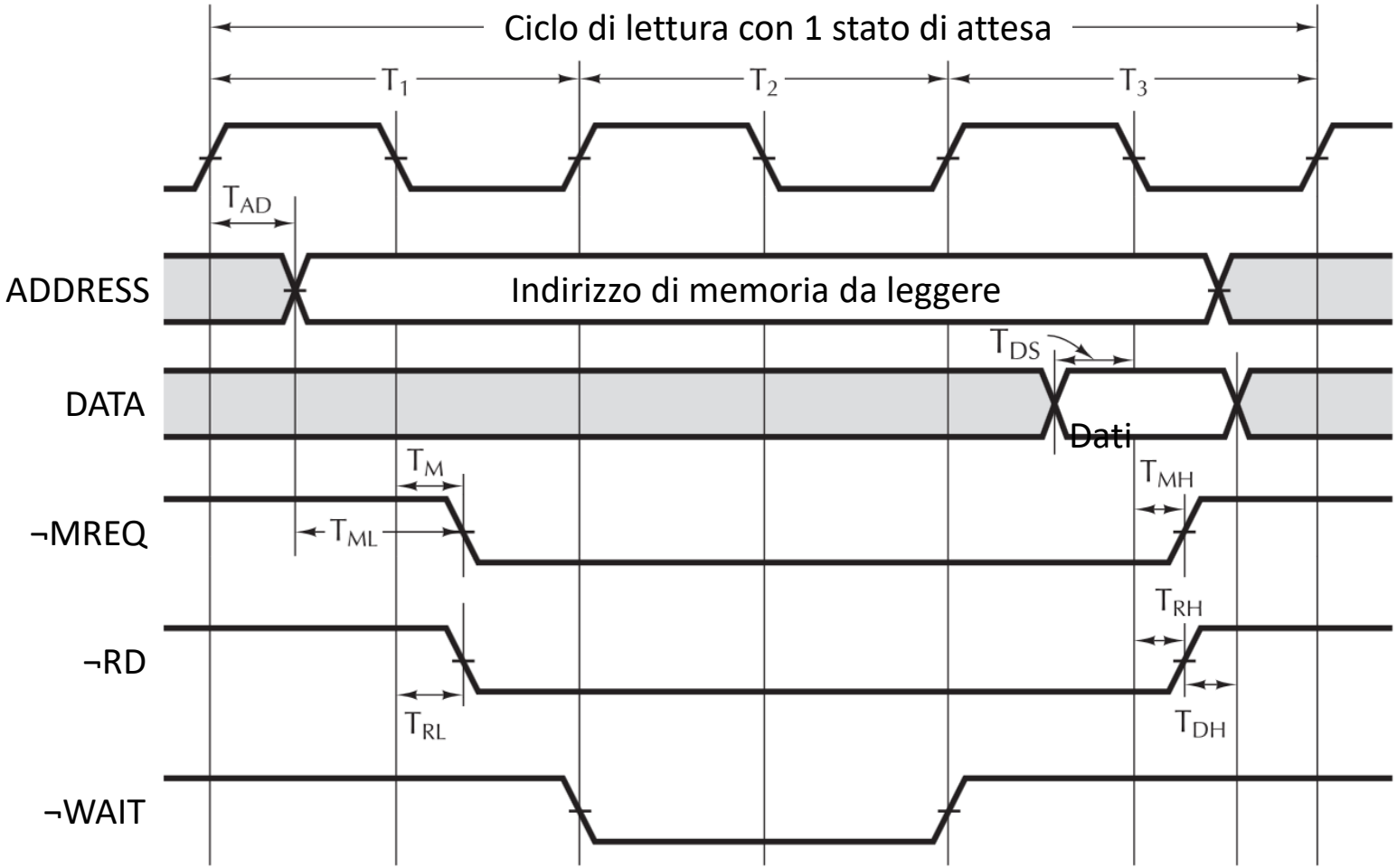
Esempio di temporizzazione di un bus sincrono

T_{MH} e **T_{RH}** indicano quanto tempo impiegano i segnali MREQ e RD per essere negati dopo che i dati sono stati generati



Esempio di temporizzazione di un bus sincrono

T_{DH} specifica per quanto tempo la memoria deve mantenere i dati sul bus dopo che il segnale RD è stato negato



Asserzione dei segnali di controllo

- I segnali di controllo possono essere asseriti con **valore alto** o con **valore basso.**
- Tocca al **progettista del bus** determinare quale dei due sia più conveniente, e la scelta è essenzialmente arbitraria.

Bus asincroni

Definizione e motivazioni d'uso

- **Bus asincrono**: assenza di clock centrale, con temporizzazione basata su handshake tra dispositivi.
- **Maggior flessibilità**: adatta a sistemi eterogenei con componenti a velocità variabili, eliminando l'attesa del dispositivo più lento e del clock (tipica dei bus sincroni).
- **Caso d'uso**: ideale per integrare nuove tecnologie senza modificare i parametri temporali globali del sistema.

Meccanismo del full handshake

- Il master di un bus, invece di legare ogni operazione al clock, dopo aver asserito l'indirizzo, MREQ, RD e tutto ciò che gli necessita, asserisce il segnale **MSYN** (Master SYNchronization).
- Quando lo slave vede **MSYN** asserito esegue il lavoro richiesto alla massima velocità possibile e, dopo aver terminato il proprio compito, asserisce **SSYN** (Slave SYNchronization).

Meccanismo del full handshake

Quando il master vede che **SSYN** è stato asserito, sa che i dati sono disponibili e può quindi memorizzarli

- successivamente il master nega le linee d'indirizzo, MREQ, RD e MSYN.

Quando lo slave vede la negazione di **MSYN**, sa che il ciclo è stato completato e quindi nega il segnale **SSYN**;

- a questo punto si è tornati alla situazione iniziale, nella quale tutti i segnali sono negati e si attende il master successivo.

Meccanismo del full handshake

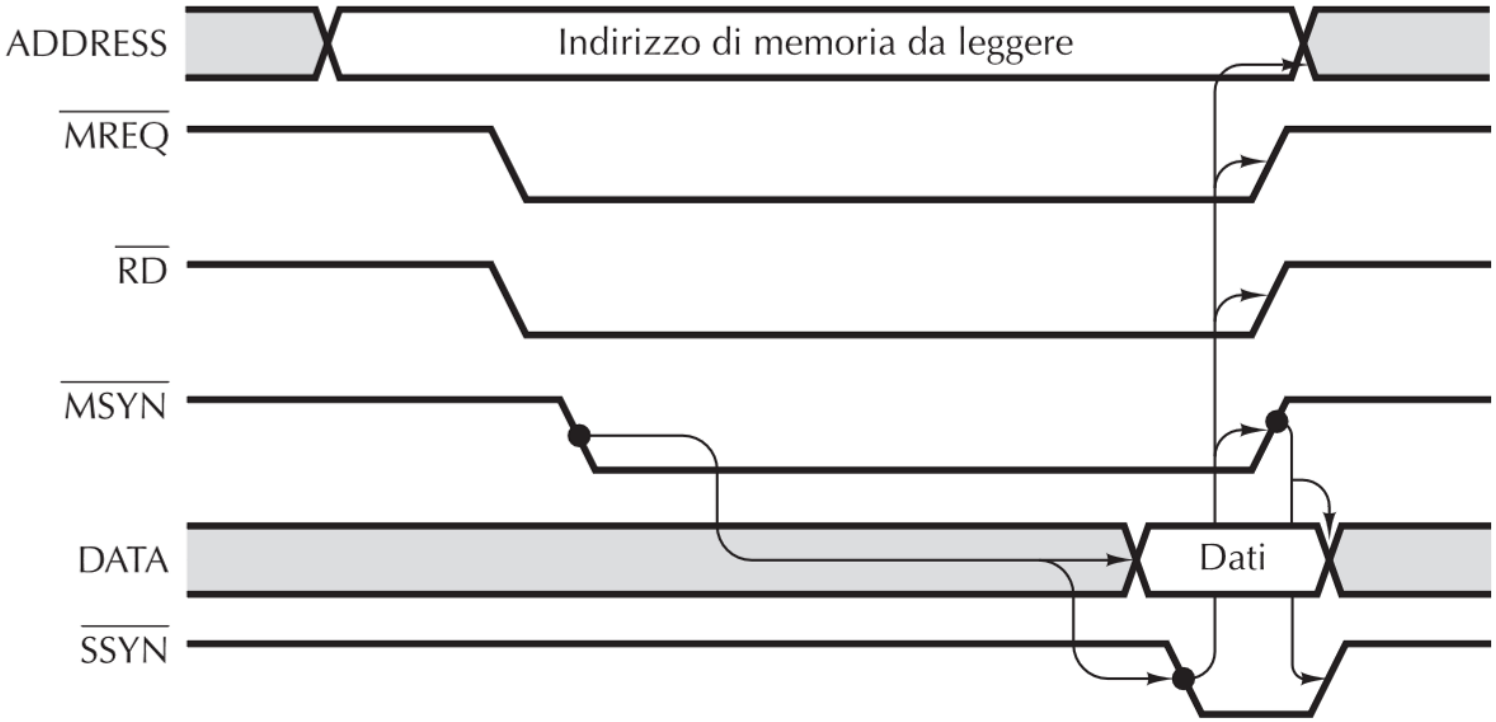
Un insieme di segnali che coordina in questo modo due dispositivi con lo scopo di non farli interferire è chiamato **full handshake**, e consiste essenzialmente di quattro eventi:

1. il segnale **MSYN** è asserito da un master;
2. in risposta al segnale **MSYN**, uno slave esegue il compito richiesto e dopo aver teminato asserisce il segnale **SSYN;**
3. in risposta al segnale **SSYN**, il master memorizza i dati e successivamente nega il segnale **MSYN;**
4. in risposta alla negazione del segnale **MSYN**, lo slave nega **SSYN.**

Esempio di temporizzazione asincrona

I diagrammi di temporizzazione dei bus asincroni (e talvolta anche dei bus sincroni) usano frecce per indicare cause ed effetti

- ad esempio, l'asserzione di MSYN ha come effetto l'asserzione delle linee dei dati e il fatto che lo slave asserisce SSYN.



Meccanismo del full handshake

Il full handshake non dipende dalla temporizzazione

- ogni evento è causato da un evento precedente e non da un impulso del clock;
- se una particolare coppia master-slave è lenta essa non influisce in alcun modo su una successiva coppia master-slave la cui velocità potrebbe essere molto più elevata.

Vantaggi rispetto ai bus sincroni

- **Efficienza**: elimina gli stati d'attesa inutili, sfruttando appieno la velocità dei componenti più recenti.
- **Scalabilità**: facilita l'aggiornamento del sistema con dispositivi più veloci senza ridefinire i tempi del bus.
- **Robustezza**: tollera variazioni temporali tra segnali, adattandosi a ritardi imprevedibili.

Sfide e limiti dei bus asincroni

Complessità progettuale

- implementare handshake richiede logica aggiuntiva e testing accurato per evitare deadlock.
- **La maggior parte dei bus sono sincroni.**
- sono più facili da realizzare.
- se i componenti siano stati scelti n modo appropriato, tutto funzionerà correttamente senza dover ricorrere all'handshake.
- enormi investimenti effettuati sulla tecnologia dei bus sincroni.

Bibliografia

Libro di testo

- Andrew S. Tanenbaum e Todd Austin. Architettura dei calcolatori. Un approccio strutturale. 6/ED. Anno 2013. Pearson Italia spa. (Disponibile nella sezione “Biblioteca”)

Fonte argomenti e immagini

- Capitolo 3, Paragrafi da 3.4.3 a 3.4.4, pp. 196-202