5 - Livello logico digitale

Componenti

Multiplexer

è un circuito logico che riceve 2^n valori di input, n valori di controllo e trasferisce in uscita solo uno degli 2^n input
Le tre linee di controllo A,B,C codificano un numero a 3 bit che specifica quale delle 8 linee di input deve essere instradata verso la porta OR (o verso l'output)

In funzione del valore sulle linee A,B e C solo una delle otto linee di entrata viene trasmessa in uscita.

2^n valore input
1 valore output
n valore controllo

A	B	C	D_i	F
0	0	0	D_0	D_0
0	0	1	D_1	D_1
0	1	0	D_2	D_2
0	1	1	D_3	D_3
1	0	0	D_4	D_4
1	0	1	D_5	D_5
1	1	0	D_6	D_6
1	1	1	D_7	D_7

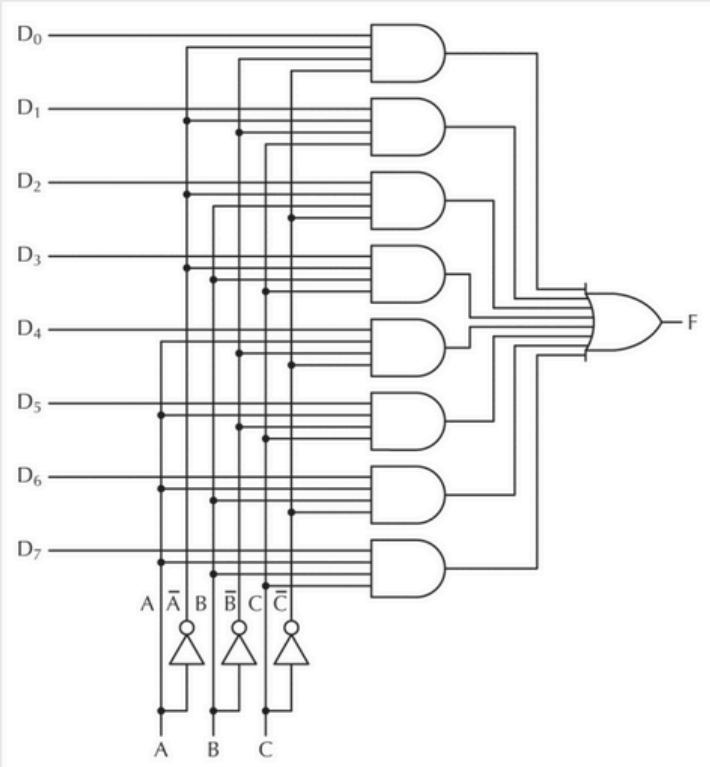


Figura 3.11 Multiplexer a otto vie.

Un multiplexer può essere utilizzato per trasformare il byte corrispondente al carattere di un tasto pigiato in una sequenza di bit che attraverso echo può essere inviato ad uno schermo.
Il contrario del multiplexer è il demultiplexer che indirizza un segnale in ingresso verso una delle otto uscite selezionata in base al segnale di controllo sulle linee A,B,C. Viene utilizzato per trasformare un segnale da seriale a parallelo

Decodificatore

domanda tipica da esame
Un decodificatore riceve n valori di input e produce 2^n valori di output (senza linee di controllo), il circuito decodificatore abilita una sola delle 2^n uscite quando eccitato dalle n entrate, esattamente quella corrispondente all'indirizzo di entrata.
È un circuito utilizzato per il processing delle istruzioni dell'ISA durante il ciclo di fetch

3 valore input (controllo)
8=2^3 valore output

A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

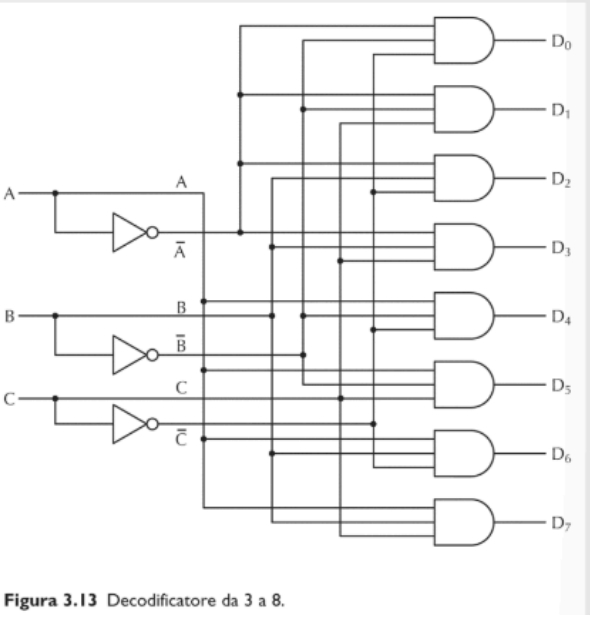


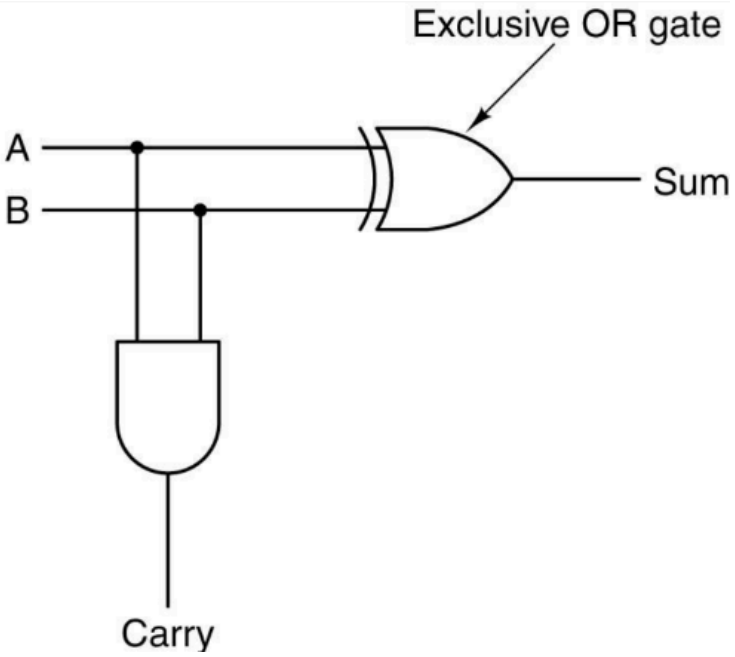
Figura 3.13 Decodificatore da 3 a 8.

In CPU cosa arriva? I 2^n bit

Addizione ad un bit (Half Adder)

Consideriamo due parole costituite da un bit
Un *half adder* (o semisommatore) è un circuito combinatorio usato per sommare due bit singoli, generando una somma e un riporto. È uno dei circuiti logici fondamentali utilizzati nei sistemi di calcolo binario.

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

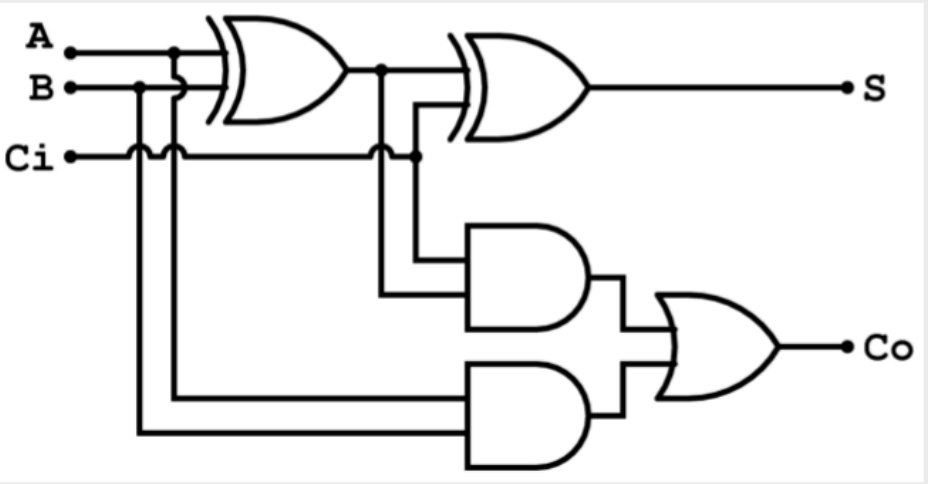


Vengono utilizzate due porte
Porta XOR: Utilizzata per calcolare la somma tra i bit in ingresso A e B.
Porta AND: Utilizzata per calcolare il riporto tra i bit in ingresso A e B.

Sommatore completo (Full Adder)

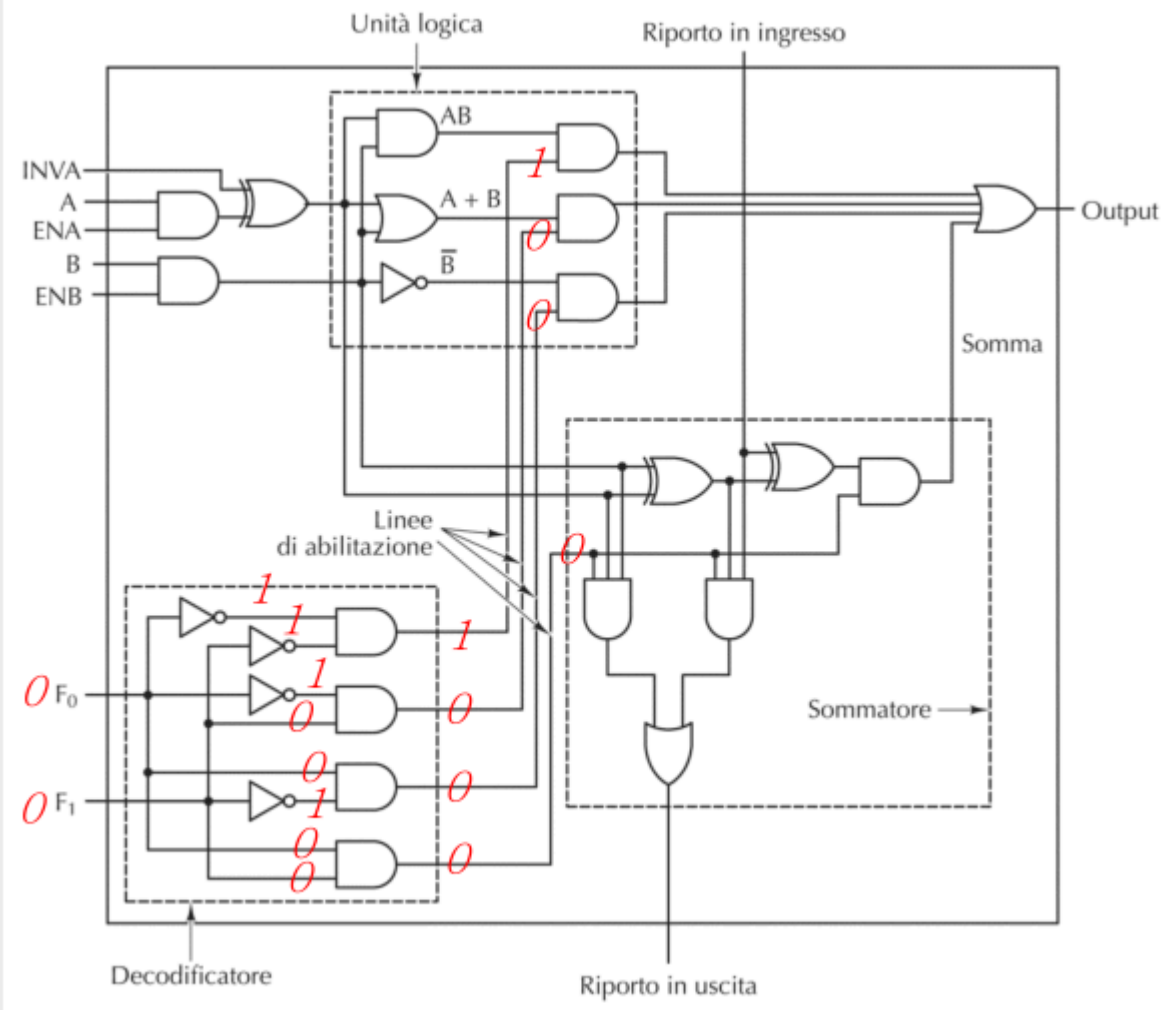
Un full adder è un circuito combinatorio usato per eseguire l'addizione tra tre singoli bit,due di input e uno di riporto.
Fa in sostanza quello che l'half adder non riesce a fare,poichè mantiene il bit di riporto dalle precedenti operazioni.

Inputs			Outputs	
A	B	C_i	C_o	S
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
1	1	0	1	0
0	0	1	0	1
1	0	1	1	0
0	1	1	1	0
1	1	1	1	1



ALU (Unità aritmetico logiche)

La maggior parte dei calcolatori contiene un unico circuito in grado di effettuare le operazioni AND,OR e somma di due parole, chiamato unità aritmetico logica.



Questo è un circuito ALU ad 1 bit

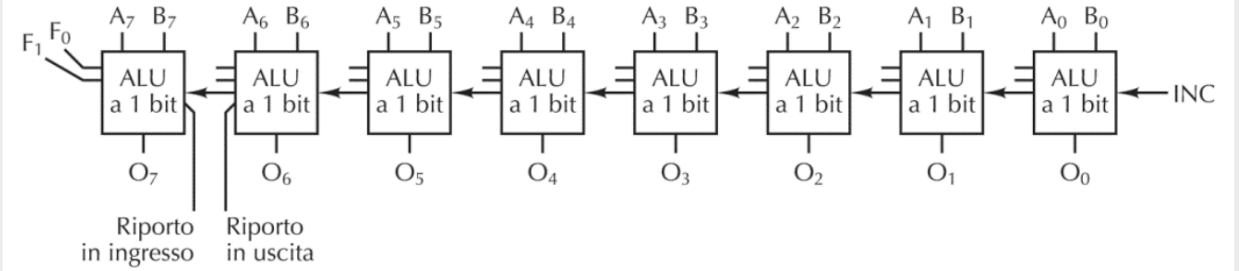
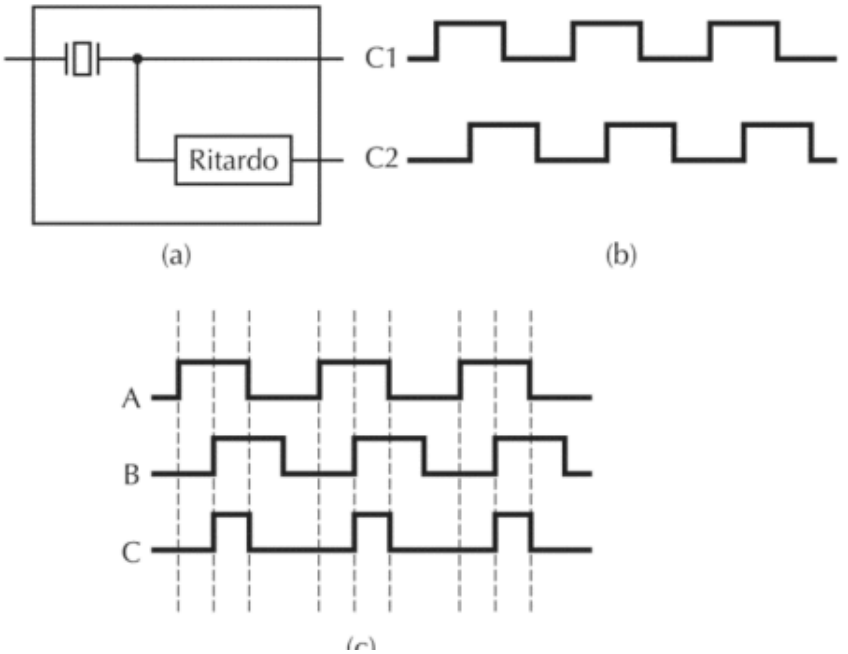


Figura 3.19 Otto ALU a 1 bit connesse per comporre una ALU a 8 bit. Per semplificare non sono mostrati segnali di abilitazione e segnali di inversione.

Questo è l'ALU a 8 bit

Clock

Molti circuiti usano i clock per gestire la sincronizzazione e permettere di ottenere relazioni temporali desiderate.
Il clock è un circuito che emette una serie di impulsi di larghezza definita e a intervalli temporali costanti, l'intervallo temporale compreso tra le due estermità di due impulsi consecutivi è detto **ciclo di clock**.
In un calcolatore possono verificarsi più eventi durante lo stesso ciclo di clock, ma se fosse necessario che si verifichino in un ordine preciso occorre dividere il ciclo in sottocicli;
Una tecnica molto utilizzata per questo fine è quella di intercettare la linea di clock e aggiungere un ritardo, generando così un secondo segnale dove si trova una fase traslata rispetto a quella principale.



In alcuni circuiti si è interessati maggiormente agli intervalli temporali rispetto agli istanti di

tempo, per questo si possono creare più linee di clock (figura in basso) oppure si può fare in modo che gli intervalli si sovrappongano parzialmente.

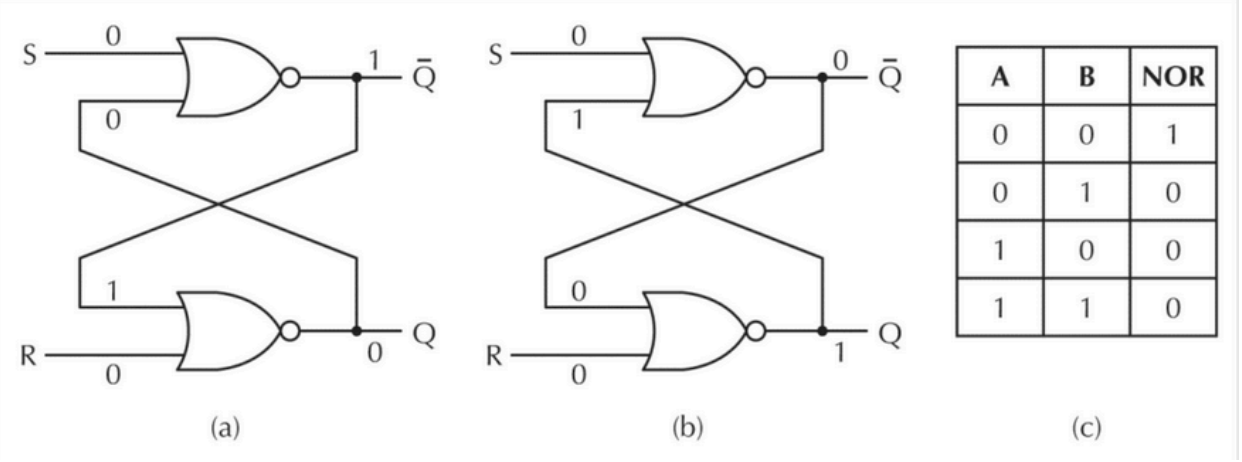
Latch SR

Un latch SR è un tipo di circuito sequenziale elementare utilizzato per memorizzare un singolo bit di informazione. È composto da porte NOR e presenta ingressi e uscite:

1. **Ingressi**: Ha due ingressi, S (Set) e R (Reset).
2. **Uscite**: Ha due uscite, Q e Q' (Q negato).

Funzionamento

- **Set (S=1, R=0)**: Quando l'ingresso S è alto (1) e R è basso (0), l'uscita Q viene impostata a 1 (stato di set).
- **Reset (S=0, R=1)**: Quando l'ingresso R è alto (1) e S è basso (0), l'uscita Q viene impostata a 0 (stato di reset).
- **Memoria (S=0, R=0)**: Quando entrambi gli ingressi sono bassi (0), il latch mantiene il suo stato precedente (memoria).
- **Condizione proibita (S=1, R=1)**: Questa combinazione è generalmente evitata perché può portare a uno stato indefinito, Q e Q' sarebbero uguali a 0.



A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

Quando un latch SR è definito come **bistabile**, significa che il circuito ha due stati stabili tra i quali può commutare: uno stato in cui l'uscita Q è 1 e uno stato in cui l'uscita Q è 0. Questi due stati stabili permettono al latch di memorizzare un singolo bit di informazione, rendendolo utile come cella di memoria elementare.
In pratica, il latch SR può mantenere il suo stato attuale fino a quando non riceve un segnale di ingresso che lo forza a cambiare stato.

Latch SR Temporizzato

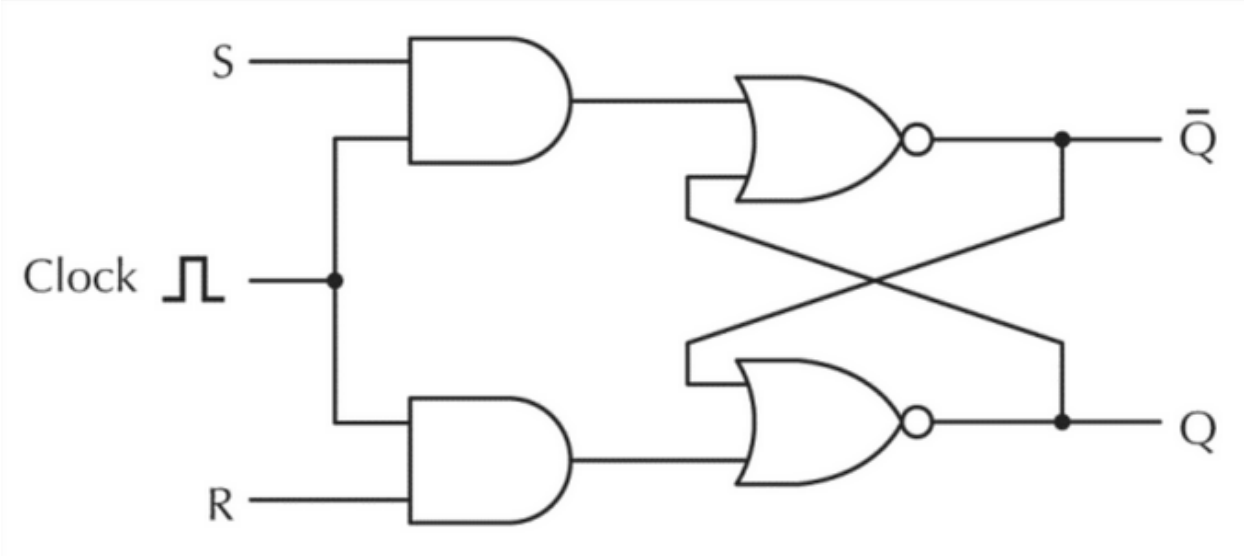
Se vogliamo che il cambio di valore avvenga in determinati istanti, bisogna aggiungere un clock per poterlo gestire. Questo ingresso aggiuntivo controlla quando il latch può cambiare stato, rendendolo più utile in applicazioni sincrone dove è necessario coordinare il cambiamento di stato con un segnale di clock.

Funzionamento

1. **Ingressi**: Ha tre ingressi principali: S (Set), R (Reset) e E (Enable o Clock).
2. **Uscite**: Ha due uscite, Q e Q' (Q negato).

Condizioni di Funzionamento

- **E=0**: Quando l'ingresso di abilitazione E è basso (0), il latch ignora gli ingressi S e R e mantiene il suo stato attuale.
- **E=1**: Quando l'ingresso di abilitazione E è alto (1), il latch si comporta come un normale latch SR:
  - **S=1, R=0**: L'uscita Q viene impostata a 1 (stato di set).
  - **S=0, R=1**: L'uscita Q viene impostata a 0 (stato di reset).
  - **S=0, R=0**: Mantiene il suo stato precedente (memoria).
  - **S=1, R=1**: Condizione proibita, porta a uno stato indefinito.



Latch D

Invece di avere due ingressi S e R in questo caso l'ingresso è unico, negando il valore di D, senza possibilità che due valori vadano ad 1, togliendo la condizione proibita

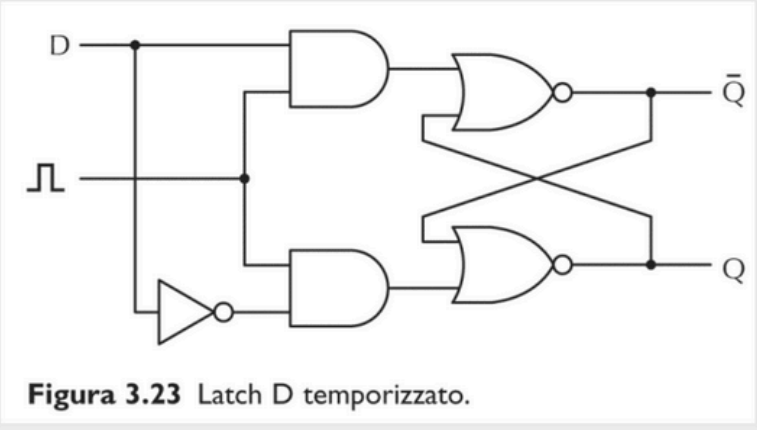


Figura 3.23 Latch D temporizzato.

E	D	Q	$\bar{Q}$
1	0	0	1
1	1	1	0
0	-	Q	$\bar{Q}$

Pro

Mantiene l'uscita e per questo è adatto all'impiego per l'interfaccia di memoria nel comma di tastiere o visualizzatori

Contro

Lo stato di ingresso si porta all'uscita nell'istante (e per tutto il tempo in cui) l'ingresso E vale 1, questo può essere causa di comportamenti indesiderati nel caso il componente fosse montato in contesti in cui l'uscita è riportata negata all'ingresso, in questo caso essa comincerebbe a oscillare e, non appena E torna a 0, si avrebbe un valore del tutto casuale4

Flip Flop (Flip Flop D-Edge-Triggered)

In molti circuiti è necessario campionare il valore di una certa linea in un particolare istante e memorizzarlo, si usano i flip-flop, la transizione di stato non si verifica quando il clock vale 1 ma durante la transizione di del clock da 0 a 1 (fronte di salita) oppure da 1 a 0 (fronte di discesa).
L'invertitore induce un piccolo ritardo di propagazione che permette al circuito di funzionare in modo corretto.

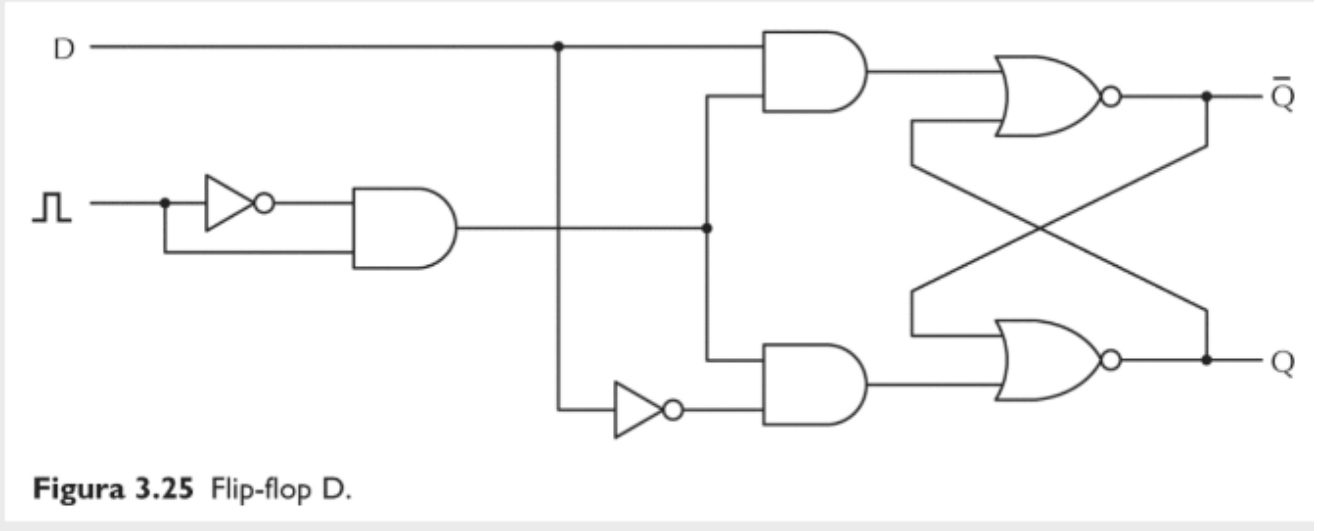
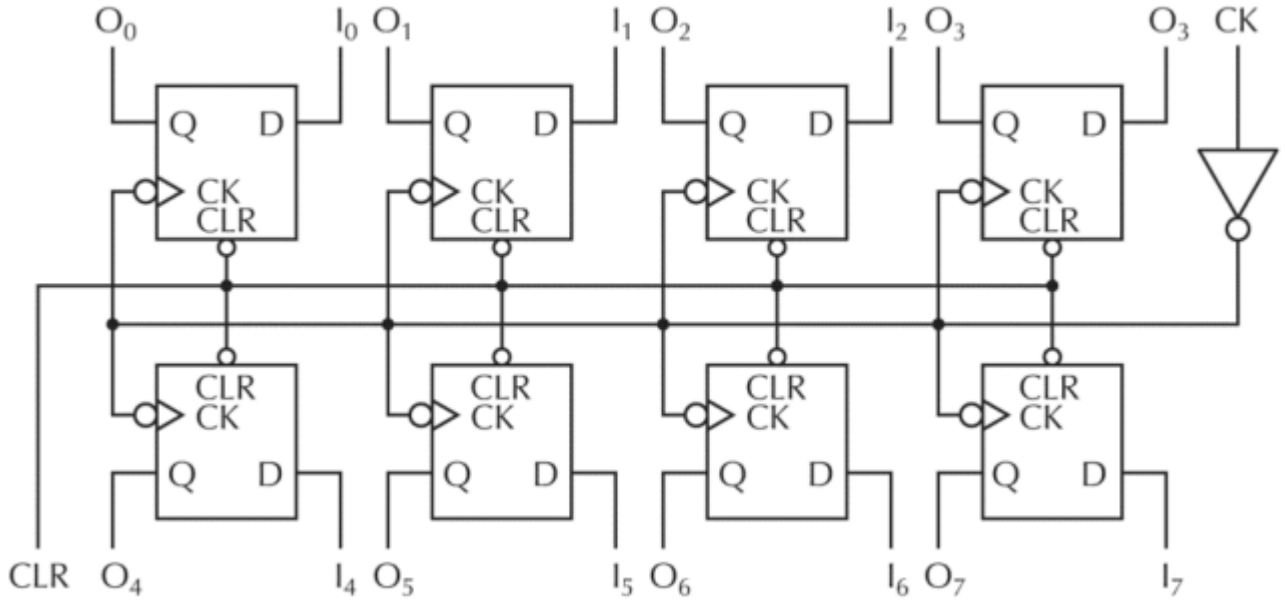


Figura 3.25 Flip-flop D.

Registro

I registri non sono altro che Flip-Flop combinati per creare questi registri che memorizzano dati composti da più bit



Questo è un registro a 8 bit, composta da 8 Flip-Flop, con tutte le linee di clock collegate allo stesso clock. Il valore di ogni bit di registro si modifica solo nell'istante del clock (CK)

Memoria

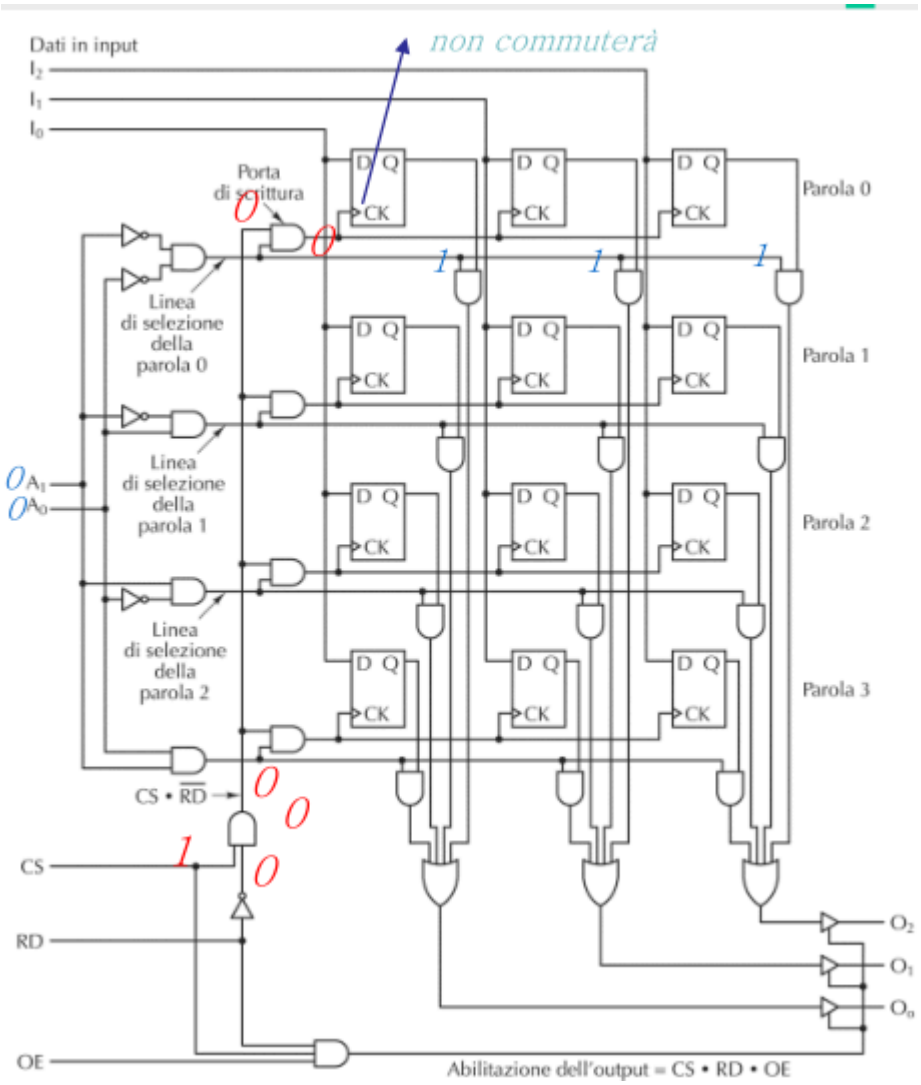
Per poter realizzare memorie di dimensione maggiore è necessaria un'organizzazione di tipo diverso, nella quale sia possibile indirizzare singole parole.
In un esempio di memoria a 4 parole di 3 bit l'una vediamo:

1. **Ingressi**:
   - **I0, I1, I2**: Questi sono i bit di input per ogni parola.
   - **A0, A1**: Questi sono i bit di indirizzo che selezionano quale delle 4 parole accedere.
   - **CS (Chip Select)**: Questo segnale abilita il chip della memoria.
   - **RD (Read/Write)**: Questo segnale determina se l'operazione è di lettura (1) o scrittura (0).
   - **OE (Output Enable)**: Questo segnale abilita l'output dei dati.

Funzionamento del Circuito

1. **Selezione della Parola**:
   - Gli indirizzi A0 e A1 determinano quale delle 4 parole è selezionata. Con 2 bit di indirizzo, possiamo avere 4 combinazioni (00, 01, 10, 11), ciascuna corrispondente a una delle 4 parole.
2. **Operazione di Lettura**:
   - Quando **RD** è impostato a 1 e **OE** è attivo, il circuito legge i dati dalla parola selezionata e li invia agli output.
   - Ad esempio, se A0=0 e A1=1, viene selezionata la seconda parola. I bit di questa

parola vengono inviati agli output.



BUS

Quando si progetta un bus si tiene presente:

- L'ampiezza
- La temporizzazione
- L'arbitraggio
- Le operazioni che consente
  Da queste scelte dipende la velocità sul bus

Master and slave

Le relazioni tra dispositivi sul bus sono regolate da una relazione di tipo Master/Slave

- **Master**: componente che ha il controllo del bus e avvia le comunicazioni

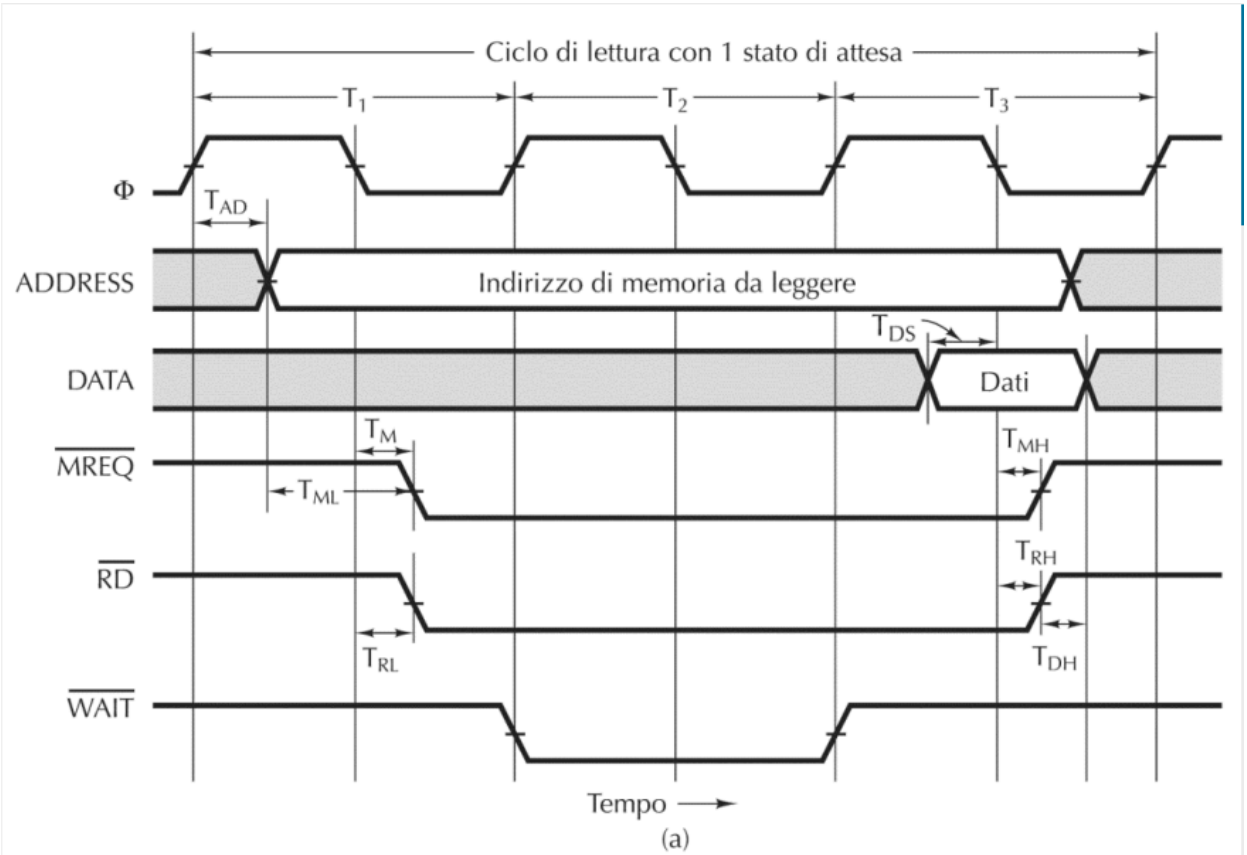
- **Slave**: chi risponde alle richieste del master

Master	Slave	Esempio
CPU	Memoria	Prelievo delle istruzioni e dei dati
CPU	Dispositivo di I/O	Inizio del trasferimento dei dati
CPU	Coprocessore	Passaggio dell'istruzione al coprocessore da parte della CPU
I/O	Memoria	DMA (Direct Memory Access)
Coprocessore	CPU	Prelievo degli operandi dalla CPU da parte del coprocessore

Temporizzazione

I bus si possono distinguere in due grandi categorie:

- Bus **sincroni**: tutte le operazioni vengono scandite dagli impulsi del clock
- Bus **asincroni**: funzionano con il meccanismo dell'invio della domanda e attesa della risposta, non viene utilizzato il clock
  Un esempio di lettura dati dalla memoria con bus sincrono è la seguente



Simbolo	Parametro	Min	Max	Unità
T_{AD}	Ritardo dell'output dell'indirizzo		4	nsec
T_{ML}	Indirizzo stabile prima di $\overline{MREQ}$	2		nsec
T_M	Ritardo di $\overline{MREQ}$ rispetto al fronte di discesa di Φ in T_1		3	nsec
T_{RL}	Ritardo di RD rispetto al fronte di discesa di Φ in T_1		3	nsec
T_{DS}	Tempo di impostaz. dei dati prima del fronte di discesa di Φ	2		nsec
T_{MH}	Ritardo di $\overline{MREQ}$ rispetto al fronte di discesa di Φ in T_3		3	nsec
T_{RH}	Ritardo di $\overline{RD}$ rispetto al fronte di discesa di Φ in T_3		3	nsec
T_{DH}	Tempo di mantenimento dei dati dopo la negazione di $\overline{RD}$	0		nsec

(b)

Esempio di lettura:

- Durante il primo ciclo di clock (T1), viene inviato l'indirizzo della locazione di memoria da accedere.
- Il segnale **MREQ** indica la richiesta di accesso alla memoria, mentre **RD** specifica se l'operazione è di lettura o scrittura.
- La memoria può segnalare un'attesa inviando il segnale **WAIT** alla CPU.
- Quando la memoria è pronta (alla fine del ciclo T2 o all'inizio di T3), nega il segnale **WAIT**, permettendo alla CPU di proseguire.
- I dati vengono trasferiti al terzo ciclo di clock.

Invece nel bus asincrono la procedura avviene in questa maniera

- **Invio dell'indirizzo**: Quando la CPU desidera accedere a una locazione di memoria, invia l'indirizzo attraverso le linee del bus. Questo segnale si propaga verso la

memoria.

- **Richiesta di accesso**: La CPU invia il segnale **MREQ** (Memory Request) per indicare la richiesta di accesso, e **RD** per specificare che l'operazione è di lettura.
- **Sincronizzazione master**: Viene inviato un segnale di sincronizzazione chiamato **MSYN** (Master Synchronism), che avvia formalmente l'operazione di trasferimento dei dati.
- **Risposta dello slave**: Una volta pronti, i dati sono inviati dalla memoria al bus. Al termine, la memoria invia il segnale di sincronizzazione **SSYN** (Slave Synchronism) per confermare la conclusione del trasferimento.
- **Chiusura della comunicazione**: Dopo la risposta dello slave, il trasferimento si considera completato e il bus diventa disponibile per una nuova comunicazione.

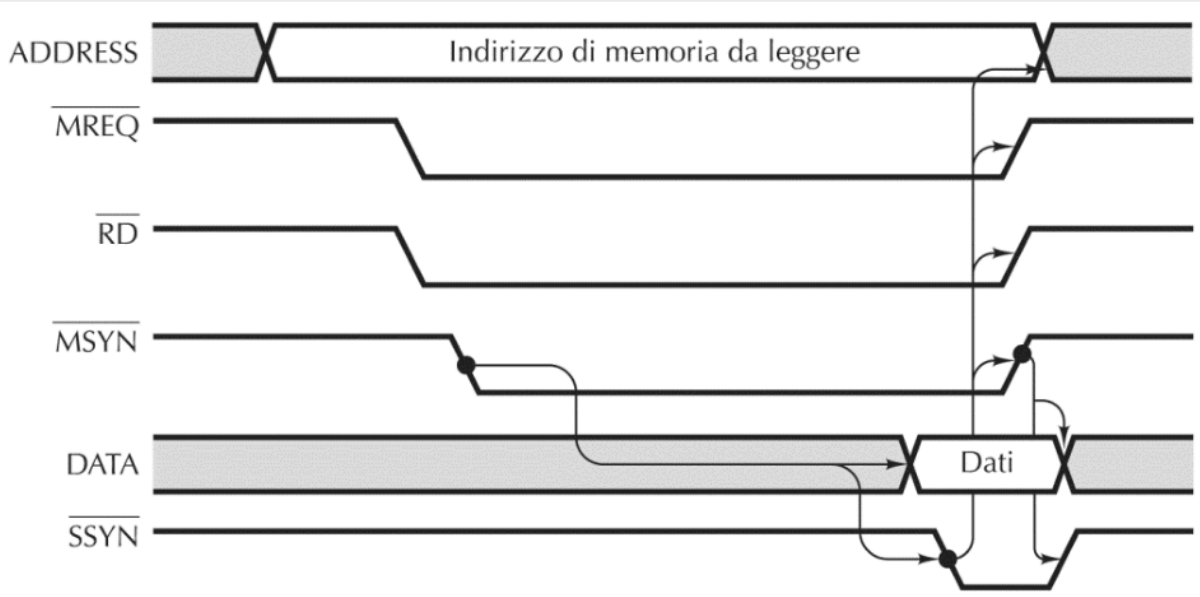


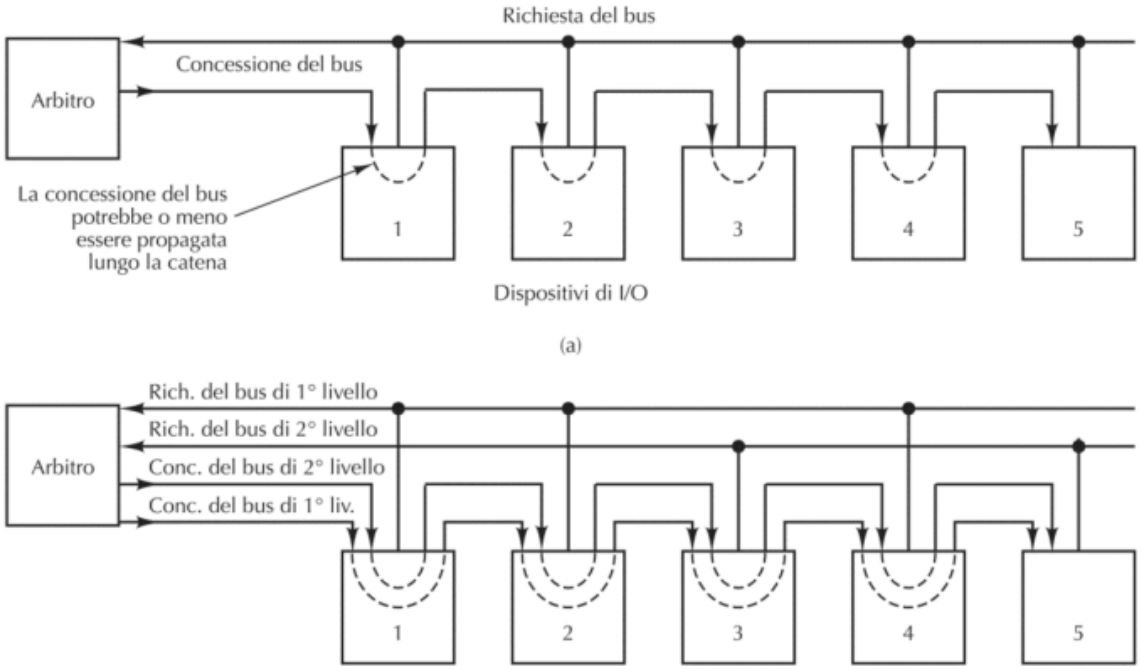
Figura 3.39 Funzionamento di un bus asincrono.

Arbitraggio

Su un bus sono collegati più dispositivi e porta sempre una richiesta, ma come si fa a capire quale dispositivo ha fatto la richiesta?
Una possibile risoluzione è il **daisy chain**:

- L'arbitro è integrato nella chip della CPU (quasi sempre, alcune volte può essere al di fuori)
- Tutti i dispositivi sono collegati ad un unica linea nell'ordine che si desidera dare loro (l'ordine corrisponde alla distanza del dispositivo dall'arbitro per livello di priorità, per esempio cache, poi RAM etc...)
- L'arbitro vede solo se c'è una richiesta, se questa è presenta concede il bus e se ne serve il dispositivo più vicino all'arbitro lungo la chain, che è quello che ha fatto la richiesta, se non l'ha fatta si passa al successivo, la successione passa lungo tutta la

chain di dispositivo in dispositivo fino a quello che ne ha fatto richiesto



Operazioni

Ma nelle macchine odierne multiprocessore, cosa accade se più dispositivi richiedono l'uso del bus contemporaneamente?
Si dota il bus di un ulteriore linea:

- 0 significa che il bus può essere concesso
- 1 significa che il bus è occupato, chi fa richiesta deve aspettare
  Ma quando è 0 e sono in due a fare richiesta temporanea?
  Bisogna eseguire le operazioni di lettura, modifica e scrittura in maniera atomica, cioè si disabilita l'interrupt solito

Tipologie di BUS

Due tipologie di bus che si usano sono:

- **USB**(Universal Serial Bus), utilizzato per connettere dispositivi lenti come tastiere, mouse, stampanti etc.
- **PCI**(Peripheral Component Interconnected):
  - Bus sincrono con relazione master-slave tra trasmittente e ricevente
  - Stesse 64 linee del bus, utilizzate sia per indirizzi che dati con il meccanismo di multiplexing. Un ciclo viene eseguito in questo modo:
    1. Viene immesso l'indirizzo sul bus
    2. Viene rimosso l'indirizzo ed il bus viene invertito in maniera che lo slave possa utilizzarlo
    3. Si generano in output i dati richiesti

PCI

Nel bus PCI il **Bridge**è un componente fondamentale che coordina i dispositivi connessi al bus PCI, permettendo l'integrazione di dispositivi a diverse velocità.
Un **commutatore** è un componente hardware utilizzato per gestire il flusso dei segnali tra diversi dispositivi o componenti di un sistema. La sua funzione principale è stabilire una connessione tra una sorgente (ad esempio, un dispositivo master) e una destinazione (ad esempio, un dispositivo slave) per consentire il trasferimento dei dati.
Il commutatore può stare nel bridge oppure nello stesso chip della CPU, le connessioni PCI sono punto-punto e seriali e consistono di due fili, uno di massa e l'altro di dati.
Questa connessione prevede la possibilità di inserimento e soppressione a caldo dei dispositivi

USB

Il sistema USB è composto da un Hub principale che si collega al bus di sistema e ai cavi dei dispositivi organizzati ad albero per dare la possibilità di connessioni multiple(la radice dell'albero si trova nell'hub principale)
Il cavo di connessione consiste di 4 fili

- 2 per i dati
- 1 per la terra
- 1 per la tensione a 5V

Per l'USB esistono 4 tipologie di frame:

- **Controllo**, utilizzati per configurare i dispositivi, assegnare loro comandi e interrogarli
- **Isocroni** (microfoni, altoparlanti etc.), che spediscono a precisi intervalli di tempo ma non possono richiedere a trasmissione in caso di errore
- **Bulk**, usati per la trasmissioni di grandi dimensioni di dati
- **Interrupt**