Argomenti

- MEMORIA
  - Flip-flop
  - Registri
  - Organizzazione della Memoria
  - Buffer
  - Chip di memoria
- Random Access Memory
- Memorie Non-volatili
  - ROM
  - PROM
  - EPROM
  - EEPROM
- Chip di CPU

- BUS
  - Ampiezza del bus
  - clock del bus
  - Bus sincroni/asincroni
  - Arbitraggio del bus
  - Interrupt handling

MEMORIA

- La Memoria è un componente fondamentale in un computer perché è utilizzata per memorizzare i dati e le istruzioni.
- Per costruire un circuito che memorizza dati è necessario utilizzare i circuiti sequenziali in cui l'uscita (O) dipende non solo dagli ingressi (I) ma anche dallo stato del circuito (S).
- Un circuito sequenziale è in grado di mantenere memorizzati dei dati, quando è alimentato.

Latch

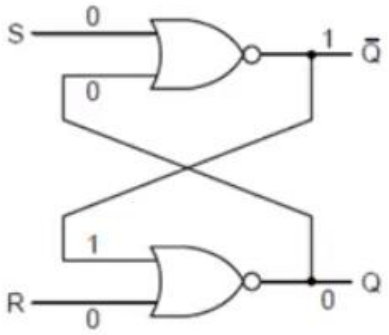
- Un circuito sequenziale semplice può essere realizzato con due porte logiche NOR (oppure NAND) che hanno le uscite retroazionate in ingresso (che dipendono dai valori dei precedenti ingressi).

- Il Latch Set-Reset (**SR latch**) ha due ingressi: S (Set) è utilizzato per impostare ad 1 il valore dell'uscita e R (Reset) usato per azzerarlo.

Latch

- Supponiamo che S ed R siano 0, quindi esistono due possibili stati:

a) Q=0 e $\overline{Q}$=1



b) Q=1 and $\overline{Q}$=0

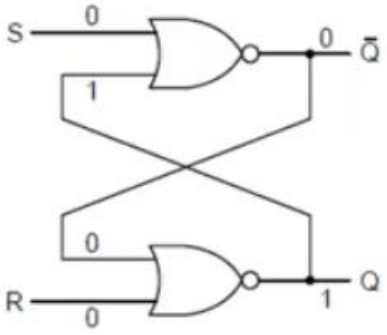


Tavola di verità

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

- Queste sono condizioni di stabilità quando gli ingressi non cambiano (e il circuito è alimentato).

SR Latch

- Se S diventa 1 mentre Q era 0 esso imposta Q ad 1

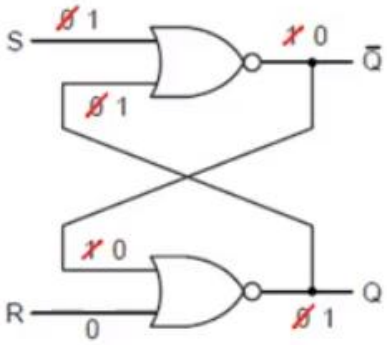


Tavola di verità del NOR

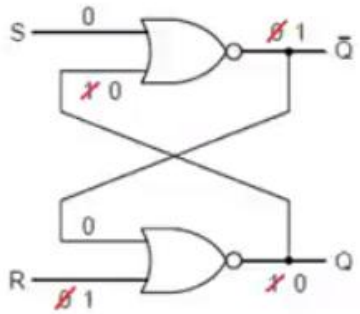
S	Q	NOR
0	0	1
0	1	0
1	0	0
1	1	0

R	$\bar{Q}$	NOR
0	0	1
0	1	0
1	0	0
1	1	0

- A questo punto se Q=1, S non potrà più cambiare lo stato del circuito che si trova in una **condizione di stabilità**.

SR Latch

- Supponiamo che ora R diventi 1 (Q è ad 1), esso resetta lo stato del latch poiché Q diventa 0:



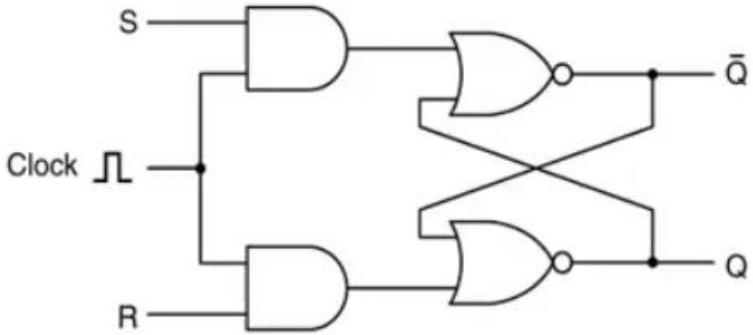
R	$\bar{Q}$	NOR
0	0	1
0	1	0
1	0	0
1	1	0

S	Q	NOR
0	0	1
0	1	0
1	0	0
1	1	0

- Non sono possibili altri cambiamenti di stato se cambia R quando Q=0 (condizione di stabilità).

SR Latch clocked

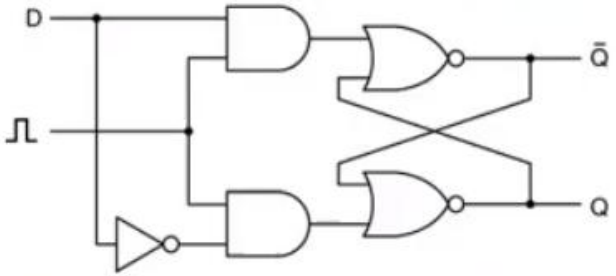
- È un latch in cui i segnali S ed R agiscono solo quando il segnale di clock è attivo.
- Può essere realizzato semplicemente aggiungendo al Latch SR due porte AND che mascherano i segnali originali S ed R.



- Il latch SR hanno uno stato instabile quando S=R=1, in questa circostanza il comportamento del circuito non è deterministico.
- Per ristabilire la condizione di equilibrio occorre impostare i valori S=R=0.
- Il latch SR è il primo circuito di memoria che ha la capacità di "ricordare" in Q e l'ultimo valore impostato per S o R.

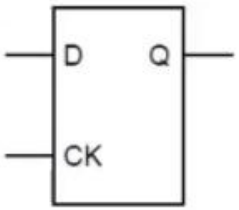
D Latch clocked

- Per risolvere l'ambiguità del latch SR (causata dalla condizione S=R=1) può essere utilizzato un solo ingresso dati (D).
- Quando D=1 e il clock è alto, il latch va nello stato Q=1.
- Quando D=0 e il clock è alto, il latch va nello stato Q=0.

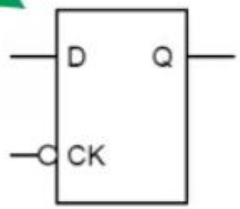


Questo circuito è detto **D latch clocked** e realizza una cella di memoria ad 1-bit.

Simbolo standard per un D latch clocked basato sulla logica positiva (lo stato cambia quando il clock è 1).

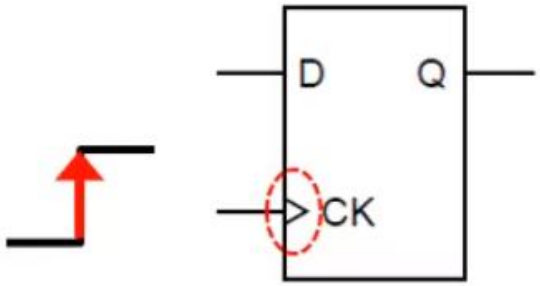


Simbolo standard per un D latch clocked basato sulla logica negativa (lo stato cambia quando il clock è 0).

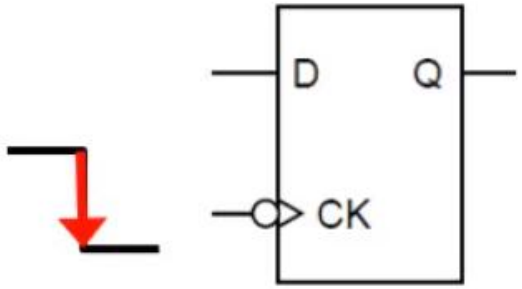


Flip-Flop

- Nei circuiti latch la transizione avviene quando il segnale di clock è alto (**level triggered**).
- Nel flip-flop lo stato cambia durante la transizione del segnale di clock da 0 a 1 (tronte in salita) oppure da 1 a 0 (fronte in discesa):



Simbolo standard per un flip-flop D che scatta sul fronte in salita

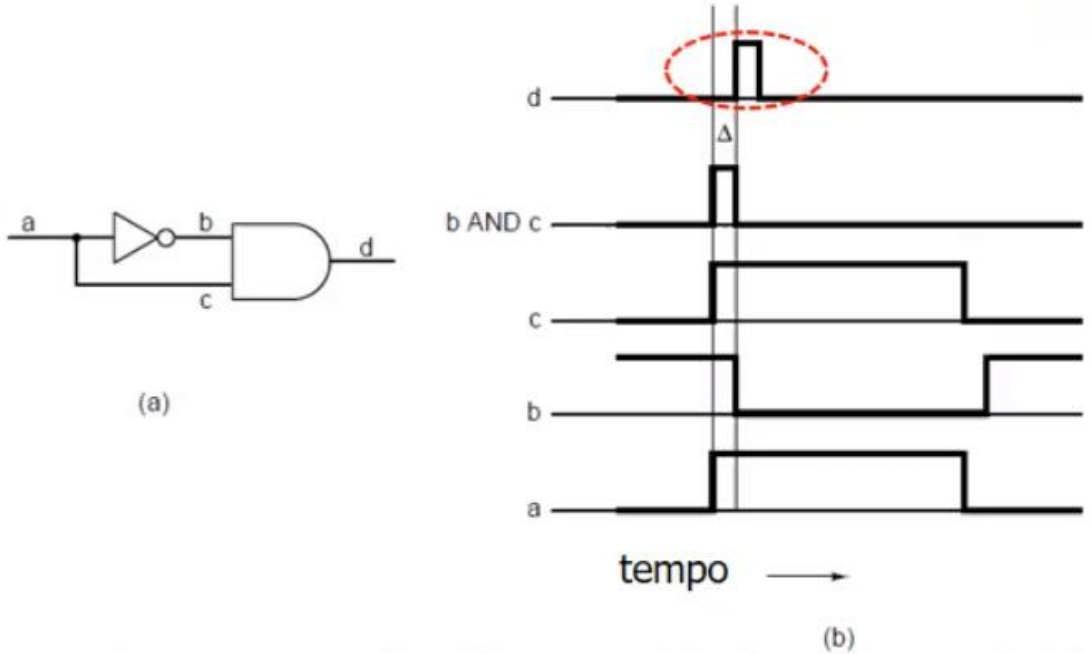


Simbolo standard per un flip-flop D che scatta sul fronte in discesa

- A causa di questo comportamento il flip-flop si dice **edge triggered** (scatta sul fronte).

Generatore di impulsi

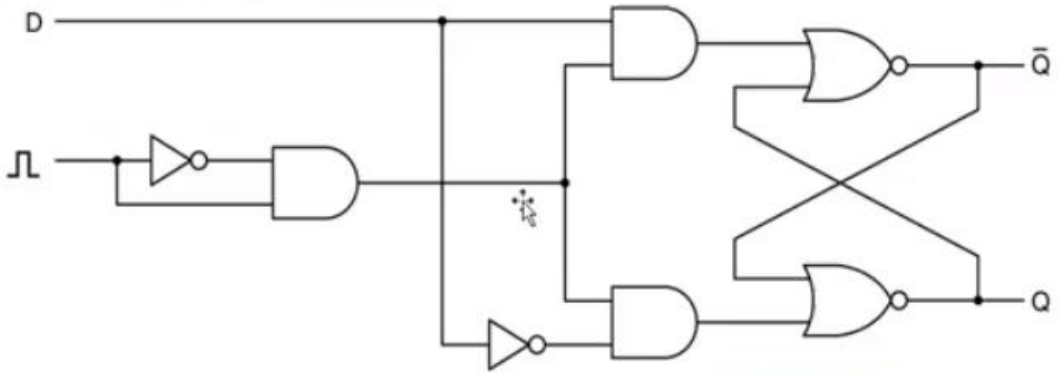
- Un generatore di impulsi si può realizzare semplicemente utilizzando una porta logica AND e un inverter che ritarda il segnale in uno dei due ingressi così in figura:



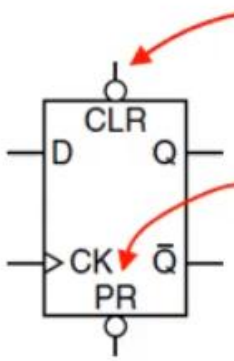
- Si osservi come anche il segnale d subisca il ritardo della porta AND.

Flip-Flop

- Il generatore di impulsi sostituisce il segnale di clock e permette di campionare il valore dell'ingresso D in un intervallo di tempo ridotto.



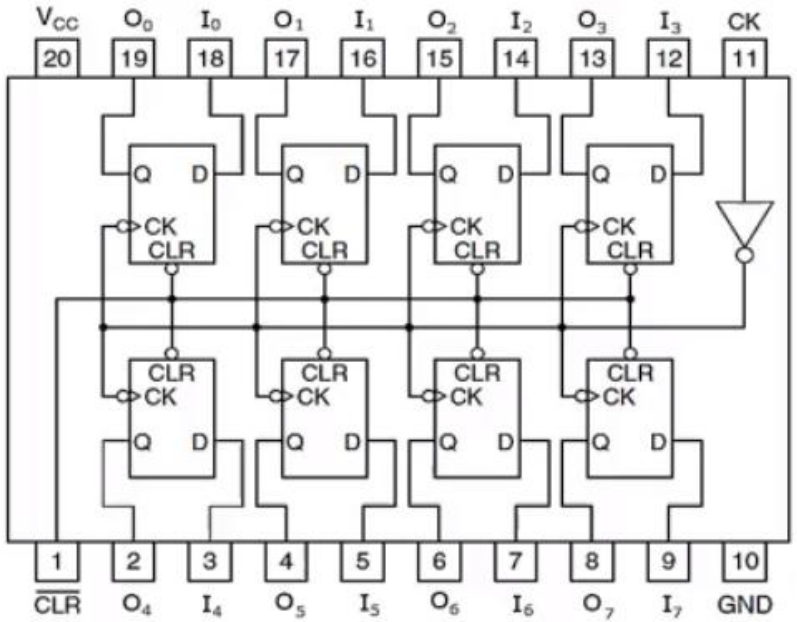
- Molti latch hanno due ingressi aggiuntivi Preset (per forzare lo stato Q=1) e Clear (per forzare lo stato Q=0).



I segnali clear ($\overline{CLR}$) e preset ($\overline{PR}$) spesso sono in logica negativa: sono attivi quando sono al livello logico basso.

Organizzazione della memoria

- Per realizzare un registo ad 1-byte possono essere utilizzati 8 flip-flop D così come mostratio nello schema del chip:



- Questo schema mostra un approccio generale per realizzare una memoria a n-byte.

Organizzazione della memoria

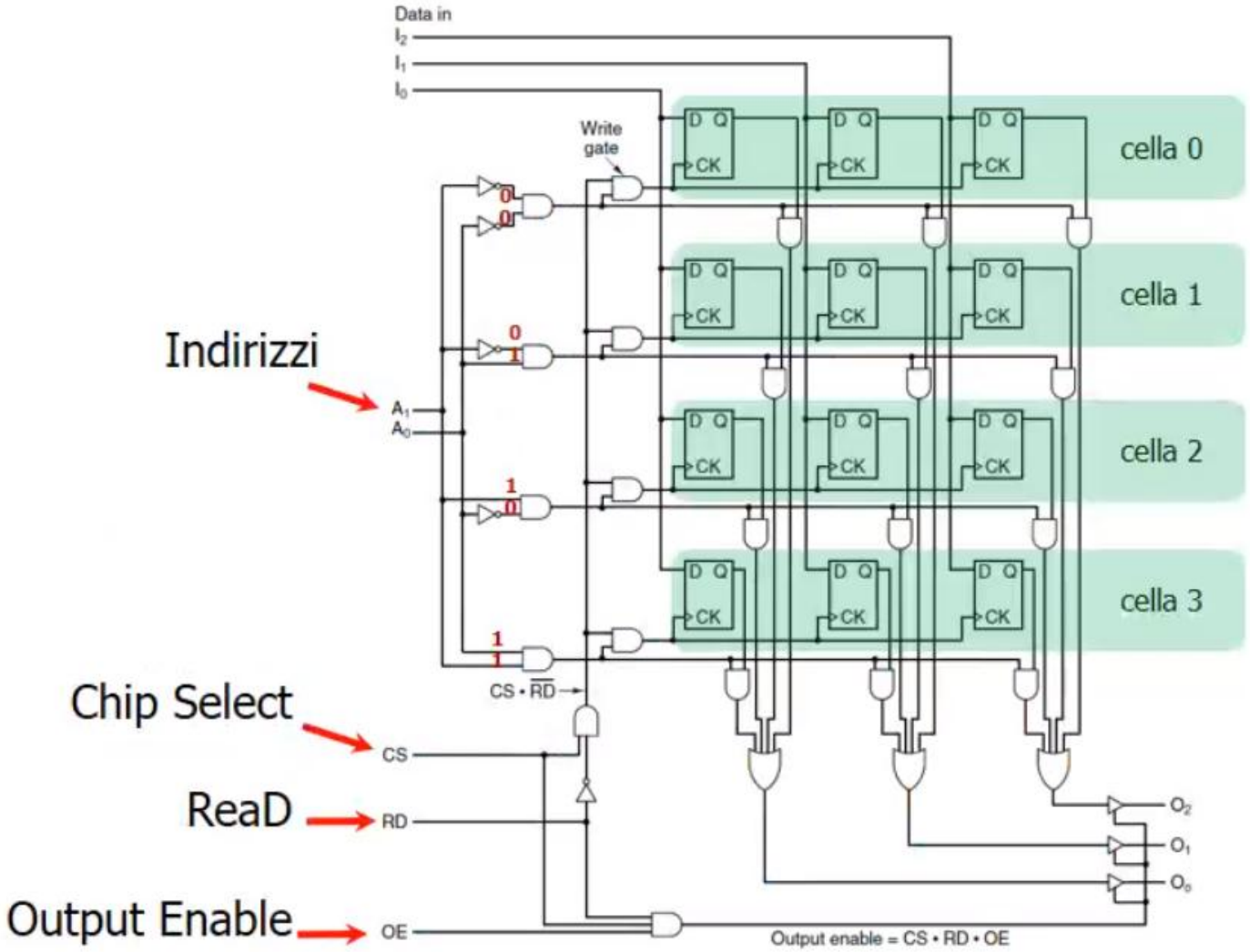
- Memorie di grandi dimensioni utilizzano un indirizzo per localizzare la cella sulla quale eseguire una operazione di lettura/scritttura, ogni cella ha lo stesson numero di bit per memorizzare dati.

- L'esempio che segue è una memoria a 12 bit costituita da 4 celle ciascuna delle quali memorizza una una parola di 3 bit.

Organizzazione della memoria

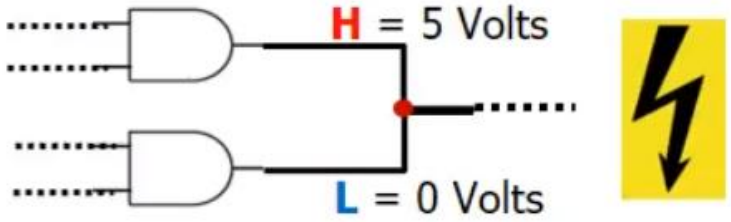
- Il circuito usa due bit per indirizzare le celle (A_0, A_1), tre bit per l'ingresso della parola (I_0, I_1, I_2), tre bit per l'uscita della parola (O_0, O_1, O_2) e tre segnali di controllo:
  - CS (**Chip Select**) usato per selezionare il circuito.
  - RD (**ReaD**) utilizzato per specificare il tipo di operazione (RD=1 lettura, RD=0 scrittura).
  - OE (**Output Enable**) utilizzato per abilitare le uscite.
  - La dimensione di una memoria generica con n linee di indirizzamento che memorizza m bit di informazione per ogni cella è pari a: 2^n x m bits.

Una memoria a 12 bit di esempio

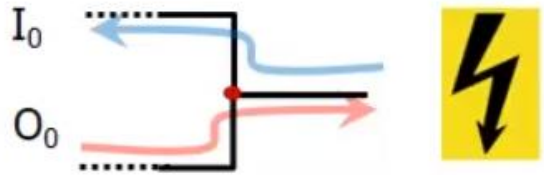


Come connettere più uscite insieme?

- Mentre si possono collegare insieme più ingressi (rispettando il Fan-out e Fan-in delle porte logiche), quando si collegano due o più uscite c'è il potenziale rischio del corto circuito:



- Nelle attuali memorie gli ingressi e le uscite occupano le stesse linee, quindi si deve far in modo di scollegare le uscite nelle operazioni di scrittura.



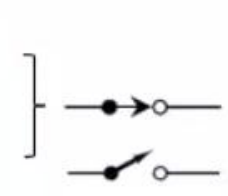
- Per risolvere questo problema esistono due differenti soluzioni: **open collector** e **buffer three state**.

Buffer tri-state

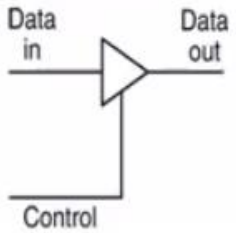
- L'uscita assume un terzo stato di alta impedenza (cioè come se fosse un circuito aperto).

- La porta logica si comporta come un interruttore che abilta o meno il passaggio del segnale dall'ingresso all'uscita.

Data in	Control	Data out
0	1	0
1	1	1
X	0	Alta impedenza



three-state o tri-state



Chip di memoria

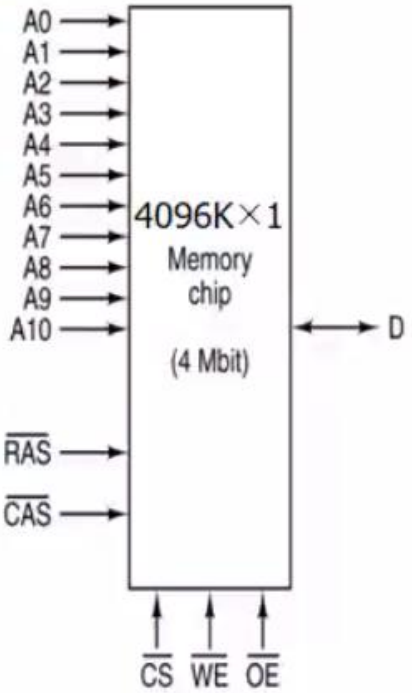
- I chip di memoria hanno schemi riusabili facilmente espandibili.
- Alcuni segnali sono in are in ***logica positiva*** (sono attivi al livello alto) altri in ***logica negativa*** (sono attivi al livello basso).
- Per evitare confusione si dirà che il segnale è **asserito** (piuttosto che dire che assume valore alto o basso) quando esso è attivo in base alla logica definita per quel segnale, così come negli esempi:
  - A è asserito quando A=1
  - $\overline{B}$ è asserito quando B=0
- Un segnale **negato** indicherà lo stato non attivo rispetto alla logica definita per quel segnale, cioè:
  - A è negato se A=0
  - $\overline{B}$ è negato se B=1

Chip di memoria

- Poiché un computer ha normalmente molti chip di memoria sono necessari alcuni segnali fondamentali:
  - **CS** (Chip Select) permette di selezionare il chip da attivare.
  - **WE** (Write Enable) determina il tipo di operazione: lettura o scrittura.
  - **OE** (Output Enable) consente di fondere insieme i segnali di uscita.

Chip di memoria

- Nella figura è mostrata una memoria da $2^{11} \times 2^{11}$ celle da un bit, per leggere o scrivere occorre:
  - Selezionare la riga in $A_0..A_{10}$ e asserire il segnale **RAS** (Row Address Strobe);
  - Selezionare la colonna in $A_0..A_{10}$ e asserire il segnale **CAS** (Column Address Strobe);
  - Leggere/scrivere **D** (Data).
- Questa organizzazione riduce il numero di pin ma rende il chip più lento rispetto ad uno equivalente con indirizzamento lineare (che richiederebbe un solo ciclo per la selezione della cella ma 11 pin in più per l'indirizzamento).



RAM statica e dinamica

- Le **RAM** (Random Access Memories) sono memorie che hanno lo stesso tempo di accesso indipendentemente dalla posizione della cella.
- Esistono due tipologie di RAM:
  - RAM statiche (**SRAM**) costruite utilizzando circuiti simili ai flip-flop D che sono in grado di mantenere l'informazione fintanto che sono alimentate.

    Sono molto veloci (nsec) e sono utilizzate per le cache di secondo livello.
  - RAM dinamiche (**DRAM**) non usano flip-flop ma usano un array di celle con un transistor ed un piccolo condensatore che può essere caricato (1) o scaricato (0) ma deve essere rinfrescato (msec).

    Le DRAM hanno maggiori capacità (1 transistor e condensstore per bit) delle SRAM (almeno 6 transistor per bit) ma richiedono circuiti più complessi per l'interfacciamento.

Chip di memoria non volatile

- Le **ROM** (Read-Only Memories) sono memorie a sola lettura il cui contenuto non cambia anche quando non sono alimentate.
- I dati in una ROM sono inseriti in fase di costruzione.
- Le **PROM** (Programmable ROM) è una ROM che può essere programmata una sola volta bruciando dei fusibili collocati nelle intersezioni della matrice.

Chip di memoria non volatile

- La **EPROM** (Erasable PROM) funziona come una PROM che può essere riprogrammata attraverso l'esposizione ai raggi ultravioletti per 15 minuti.
- La **EEPROM** (Electrically Erasable PROM) funziona come una PROM che per essere cancellata (byte a byte) non deve essere inserita in una camera speciale, per l'esposizione alla luce ultravioletta, ma è sufficiente applicare un impulso elettrico.

  La **memoria flash** è un tipo speciale di EEPROM che può essere cancellata a bocchi (e non a byte).

Tipi di memorie

Tipo	Categoria	Cancellaz.	Byte modif. ?	Volatile ?	Applicazione
SRAM	Read/Write	Elettrica	Sì	Sì	Cache II Livello
DRAM	Read/Write	Elettrica	Sì	Sì	Vecchie mem. centrali
SDRAM	Read/Write	Elettrica	Sì	Sì	Memoria centrale
ROM	Read-Only	Impossibile	No	No	Elettrodomestici (grandi volumi)
PROM	Read-Only	Impossibile	No	No	Dispositivi (piccoli volumi)
EPROM	Principalm. Read	Raggi UV	No	No	Prototipi di dispositivi
EEPROM	Principalm. Read	Elettrica	Sì	No	Prototipi di dispositivi
Flash	Read/Write	Elettrica	Sì	No	Memorie di massa

Chip della CPU

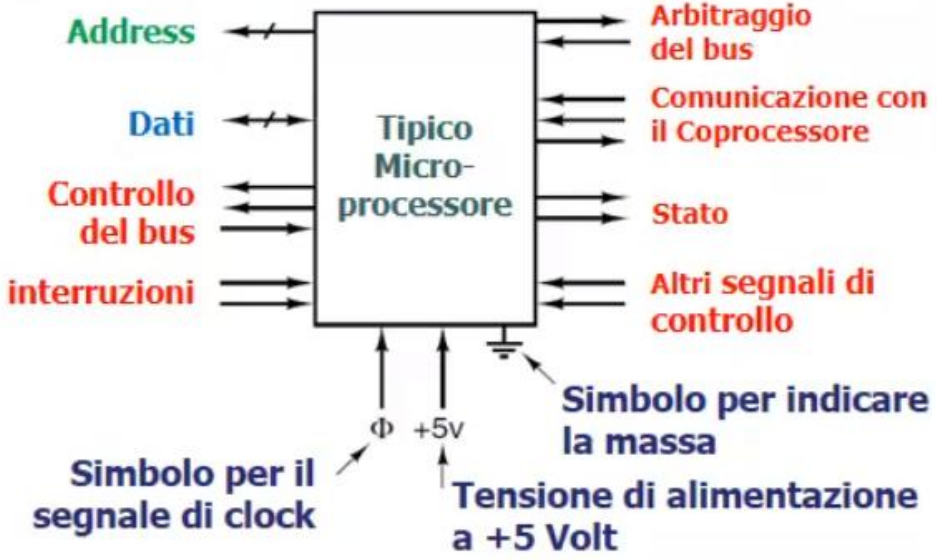
- Tutte le CPU moderne sono contenute in un unico chip che possiede un insieme di pin (e segnali) attraverso i quali la CPU comunica con il mondo esterno.
- I pin/segnali della CPU possono essere divisi in tre differenti categorie: **indirizzo** (o **address**), **dati** e **controllo**.
- I pin sono connessi agli altri componenti (memoria, dispositivi di I/O,...) attraverso il **bus**.

Chip della CPU

- Quando la CPU esegui un'istruzione:
  - inizialmente pone l'indirizzo di memoria dove si trova l'istruzione in memoria sul bus indirizzi;
  - Informa la memoria che vuole eseguire una operazione di lettura (utilizzando le linee di controllo del bus);
  - La memoria risponde ponendo la parola sul bus dati e asserendo un segnale che indica che l'operazione è stata svolta;
  - La CPU riceve questo segnale di controllo, legge i dati e si predispone per eseguire l'istruzione.

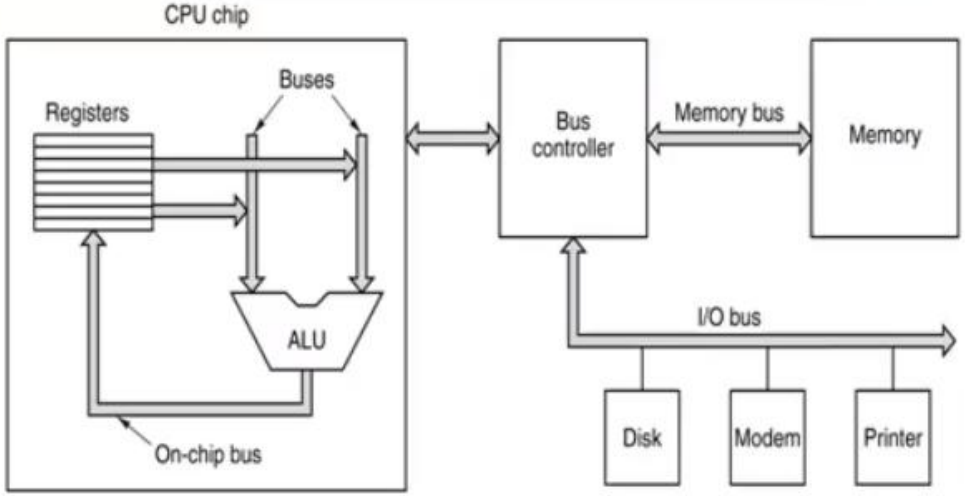
Pin della CPU

- I principali parametri che determinano le prestazioni di una CPU sono il **numero di pin** per l'**indirizzamento** e per i **dati**.
- Una CPU con m pin di address può indirizzare fino a 2^m locazioni di memoria, con n pin dati può leggere/scrivere una parola di n-bit in una singola operazione.
- I pin di **controllo** possono essere raggruppati nelle seguenti categorie:
  - Interruzioni.
  - Arbitraggio del bus
  - Controllo del bus.
  - Comunicazione cor il coprocessore.
  - Stato.
  - Vari altri segnali di controllo.



Bus del calcolatore

- Un **bus** è un collegamento elettrico **comune** che unisce vari dispositivi.
- I bus possono essere **interni** alla CPU (connessione registri-ALU) oppure **esterni** per connetterla alla memoria o alle periferiche di I/O.
- Mentre i progettisti di CPU possono utilizzare il tipo di bus che desiderano, per i bus esterni devono essere definite delle regole (**protocollo del bus**) rispettare.
- I primi PC avevano un solo bus di sistema, oggi esistono più bus per il collegamento **CPU-memoria** e **CPU-device di I/O**



Bus del calcolatore

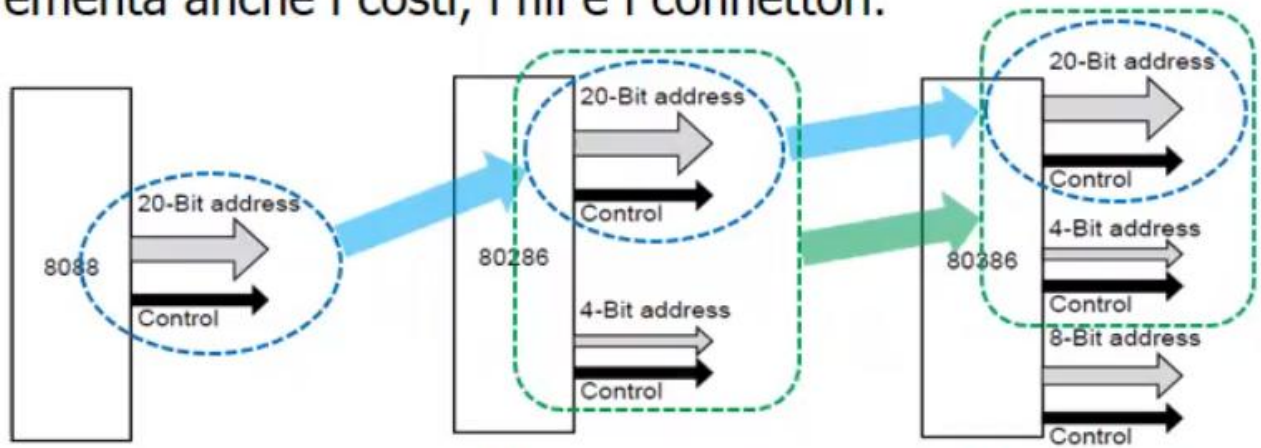
- Alcuni dispositivi nel calcolatore sono **attivi** (o **master**) perché possono iniziare un trasferimento dati sul bus, altri **passivi** (o **slave**) poiché restano in attesa di una richiesta di un master.

- Per amplificare il segnale dei master sul bus è utilizzato un chip chiamato **bus driver**, viceversa gli slave usufruiscono di un chip chiamato **bus receiver** per il collegamento con il bus.

- Questi chip di interfacciamento per evitare il problema di connettere più uscite insieme utilizzano delle porte logiche **buffer tri-state** oppure il collegamento in **wired-OR** di più porte open collector.

Ampiezza del bus

- Se un bus ha n linee di address, allora la CPU può indirizzare almeno 2^n diverse locazioni di memoria.
- L'incremento della dimensione del bus indirizzi (in termini di bit) permette di indirizzare maggiori spazi di indirizzamento ma incrementa anche i costi, i fili e i connettori.



2^{20} = max 1 MB 2^{24} = max 16 MB 2^{32} = max 1 GB

- L'evoluzione dei processori Intel mostra come l'incremento delle linee e la retrocompatibilità siano due caratteristiche contrapposte.

Larghezza di banda del bus

- Per incrementare la **larghezza di banda** del bus si può:

Ridurre il periodo del clock del bus (più trasferimenti al secondo) è possibile ma difficile perché di corre il rischio di:

- Avere più dispositivi che operano a velocità differenti (**problema del disallineamento del bus**).
- Perdere la retrocompatibilità.

Incrementare l'ampiezza del bus **dati** (più bit nell'unità di tempo).

Per non avere bus troppo ampi (ed effetti collaterali) si può utilizzare una tecnica di multiplexing delle linee sia per dati sia per indirizzi, ovviamente questo conduce ad un rallentamento del sistema.

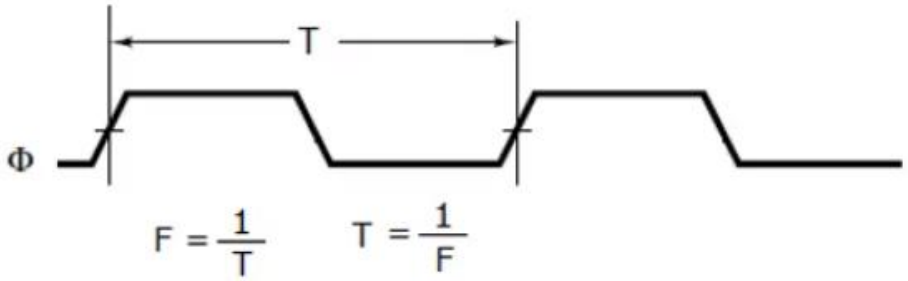
Temporizzazione del bus

- Dal punto di vista della temporizzazione esistono due differenti approcci: **bus sincroni** e **asincroni**.

- I bus sincroni utilizzano un clock che determina la temporizzazione delle attività sul bus (**ciclo di bus**): ogni operazione richiede un numero di periodi di clock per essere eseguita.

Temporizzazione del bus

- Per questioni di affidabilità e retrocompatibilità i bus moderni operano a velocità di clock non elevata (per esempio il bus PCI ha una frequenza di clock nell'intervallo 33 ÷ 66 MHz).



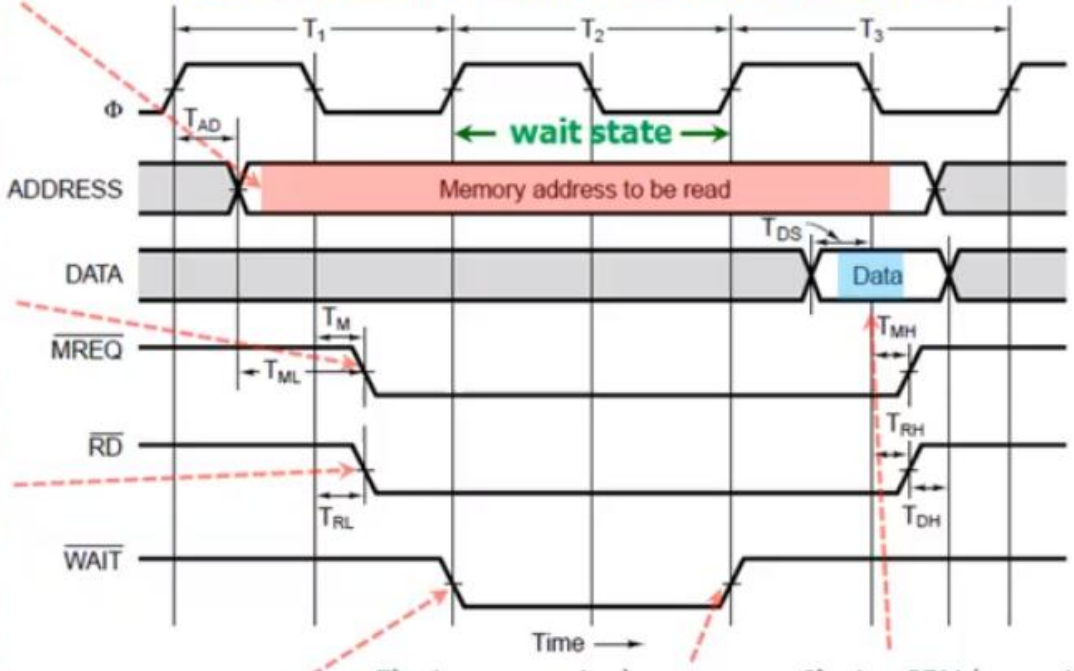
Se la frequenza di clock è 100 MHz il ciclo di bus ha un periodo di 10 ns

- Il bus asincrono invece il ciclo di bus può avere qualsiasi durata e quindi è più efficiente. Tuttavia è più difficile da costruire rispetto ad uno sincrono.

Bus sincrono

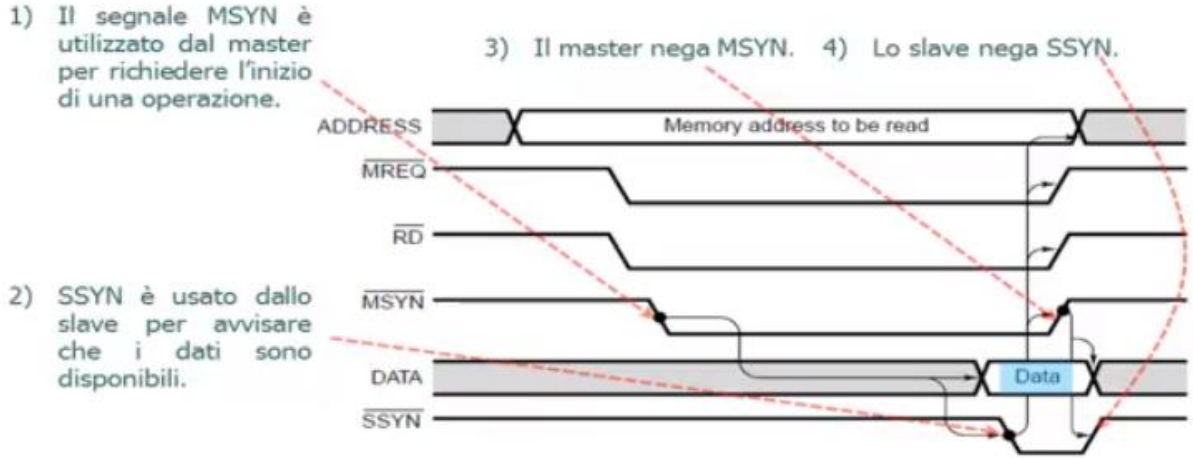
1) La CPU (**master**) pone l'indirizzo di memoria sullo address bus in modo che le linee si stabilizzino.
2) La CPU comunica al sistema che l'operazione che intende svolgere è con la memoria.
3) La CPU comunica che si tratta di una operazione in lettura, a questo punto la memoria (**slave**) deve fornire sul data bus il contenuto della cella indirizzata dall'address bus.
4) Poiché la memoria è meno veloce della CPU, le chiede uno stato di attesa aggiuntivo (**wait state**).

Ciclo di lettura in memoria con uno stato di wait



5) La memoria è pronta a fornire i dati allora nega il segnale di WAIT e la CPU potrà leggere i dati.
6) La CPU legge i dati sul fronte in discesa di T_3.

Bus asincrono



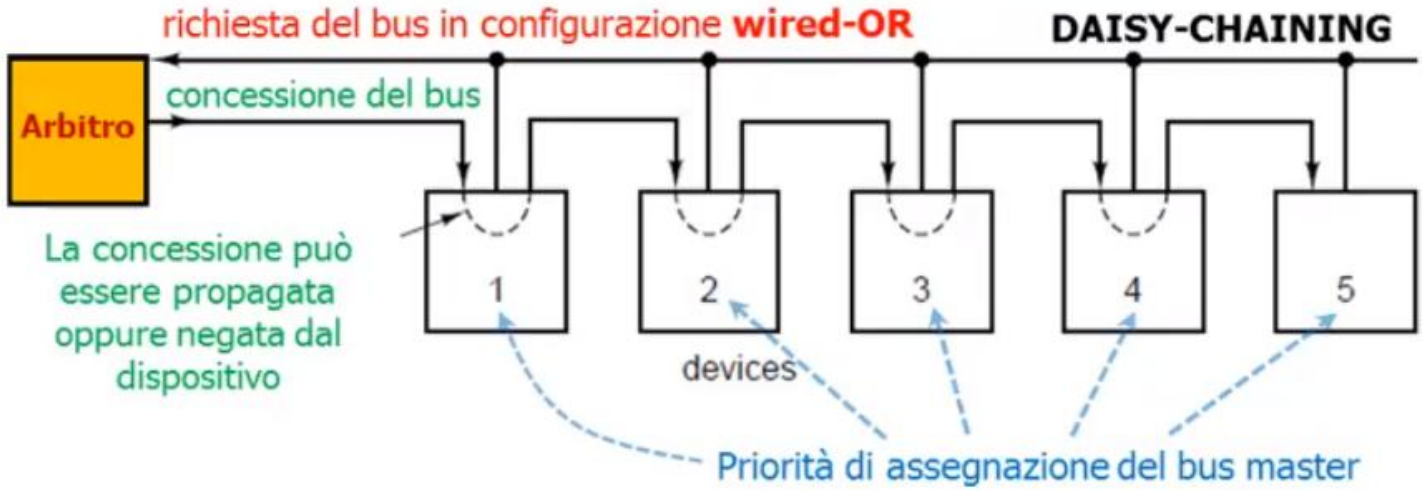
- I 4 eventi in figura mostrano un **handshake** completo (definito **full**) tra master e slave, il tutto avviene senza base dei tempi.
- In questo modo il bus si adatta alla velocità del dispositivo collegato ed un dispositivo lento non rallenta l'intero sistema!
- La maggior parte dei bus sono sincroni perché più semplice da realizzare e a causa degli enormi investimenti fatti fin ora.

Arbitraggio del bus

- Ciascun dispositivo "intelligente" del computer (CPU, coprocessor, I/O devices,...) può diventare, a turno, master del bus.
- L'arbitraggio del bus è utilizzato per prevenire situazioni di conflitto in cui due o più dispositivi tentano di diventare, nello stesso momento, master del bus.
- L'arbitraggio può essere **centralizzato** o **decentralizzato**.
- Il primo necessita della presenza di un **arbitro**, normalmente integrato nel chip della CPU, che può diventare anche un fattore di criticità in caso di rottura.

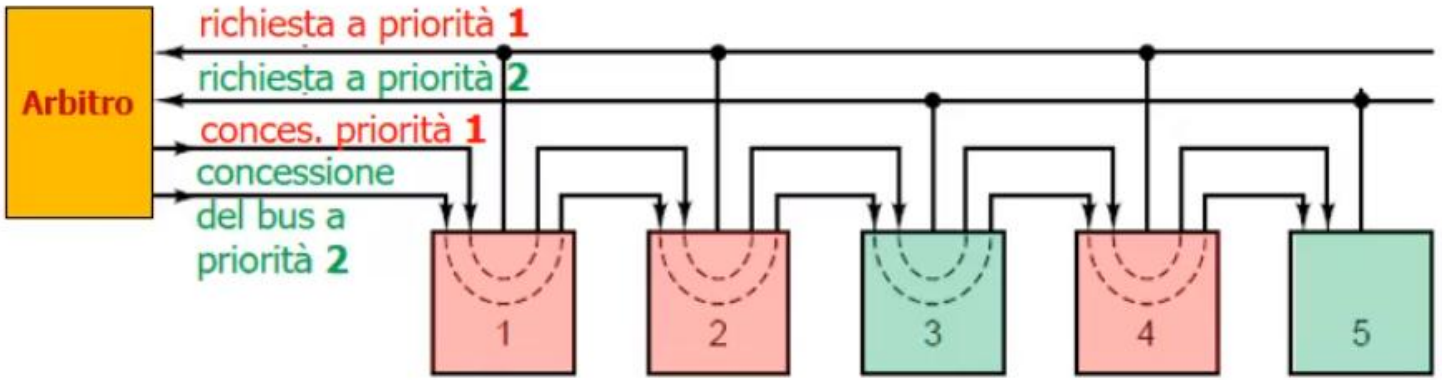
Arbitraggio centralizzato del bus

- Quando l'arbitro riceve una **richiesta**, concede la richiesta asserendo una linea di **concessione del bus**.
- Quando il dispositivo più vicino all'arbitro vede la concessione:
  - Se è lui che lo ha chiesto, blocca la line negandola a tutti i sui successori;
  - Altrimenti mantiene asserita la linea.
- Quando due, o più, dispositivi fanno richiesta vince il più vicino all'arbitro (la priorità è nel cablaggio della connessione).



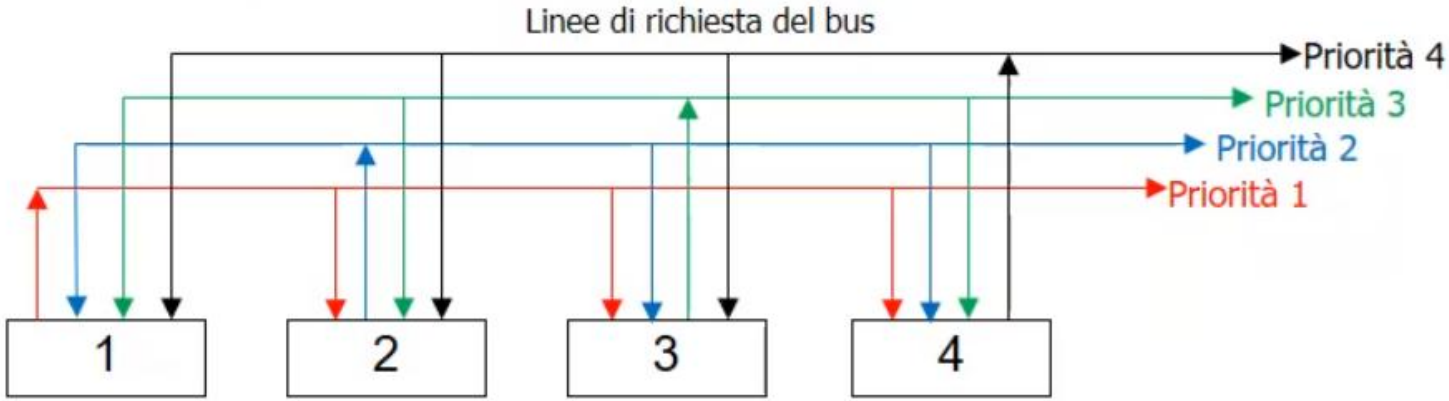
Arbitraggio con più linee di priorità

- Per superare il problema che la priorità è cablata nella connessione molti bus definiscono più livelli di priorità utilizzando differenti linee richiesta-concessione (4, 8 o 16).
- In questo caso l'arbitro concede il bus al dispositivo con la priorità più alta, a parità di priorità vince chi è più vicino all'arbitro nella catena (daisy-chain).



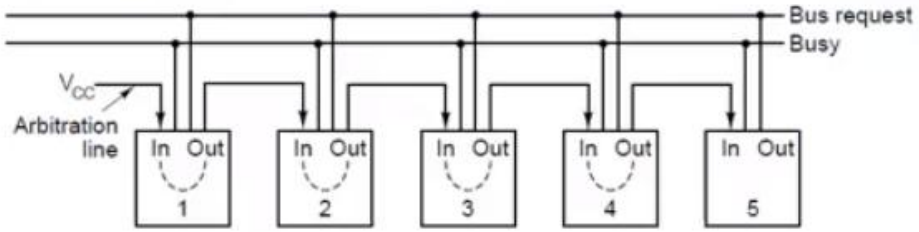
Arbitraggio decentralizzato del bus

- Ogni dispositivo ha una propria linea di richiesta ed una priorità.
- Prima di richiedere il bus ciascuno deve verificare che non ci sia già una richiesta con priorità più alta.
- Al termine dell'utilizzo del bus, la linea di richiesta deve essere negata.
- Svantaggi: troppi collegamenti, il numero di dispositivi non può superare il numero di linee.



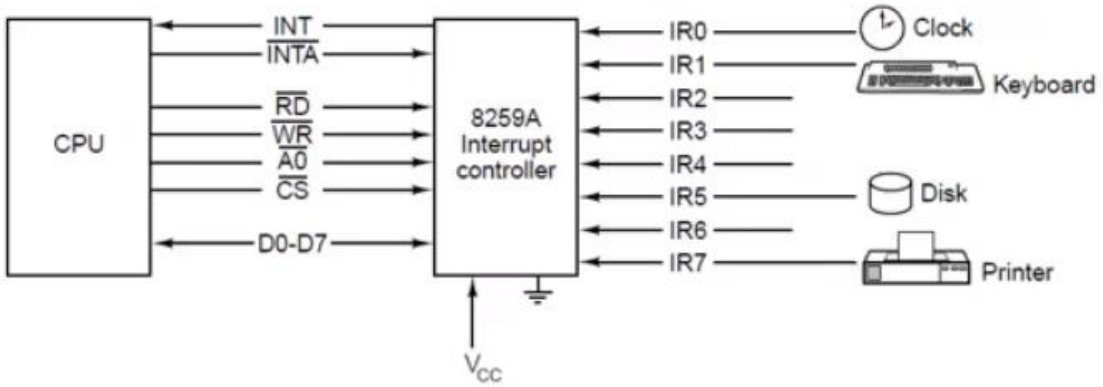
Arbitraggio decentralizzato del bus

- Un differente schema decentralizzato di arbitraggio utilizza tre linee:
  - La linea di **richiesta del bus** (in configurazione wired-OR).
  - La linea the **busy** asserita dal dispositivo bus master corrente.
  - La linea di **arbitraggio** propagata tra i dispositivi in cascata.
- Per ottenere il bus un dispositivo deve:
  - Controllare che busy sia negata e che l'ingresso **In** sia asserito.
  - In questo caso nega **Out**, asserisce la linea busy e diventa il master del bus.
- Al termine, sblocca **Out** asserendolo e libera busy negandolo.



Interrupt handling

- Il chip 8259A può ricevere fino ad 8 richieste di interrupt ($IR_0 \div IR_7$).
- Quando uno o più dispositivi richiede una interruzione l'8259A asserisce il segnale **INT** (INTerrupt) alla CPU.
- Se la CPU è in grado di gestire la richiesta di interruzione, risponde al controllore con il segnale **INTA** (Acknowledge signal). A questo punto l'8259A deve porre sul data bus il numero del dispositivo che ha generato l'interrupt.



Interrupt handling

- La CPU utilizza quel numero come indice per accedere al **vettore di interruzione** e trovare l'indirizzo della **ISR** (Interrupt Service Routine).
- Per gestire più di 8 dispositivi i controllori 8259A possono essere collegati in cascata.