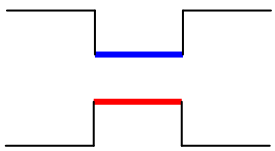
# Il livello logico digitale

I calcolatori odierni sono costituiti da **circuiti digitali** (Hardware); ogni circuito di base, se preso singolarmente, è straordinariamente semplice; d'altro canto, grazie all'aggregazione di circuiti di base possono essere realizzati sistemi arbitrariamente potenti e complessi.

Un sistema digitale opera con segnali "**discretizzati**". Nel caso dei calcolatori e, in generale, nella maggior parte dei circuiti elettronici digitali un segnale può assumere solo 2 stati:

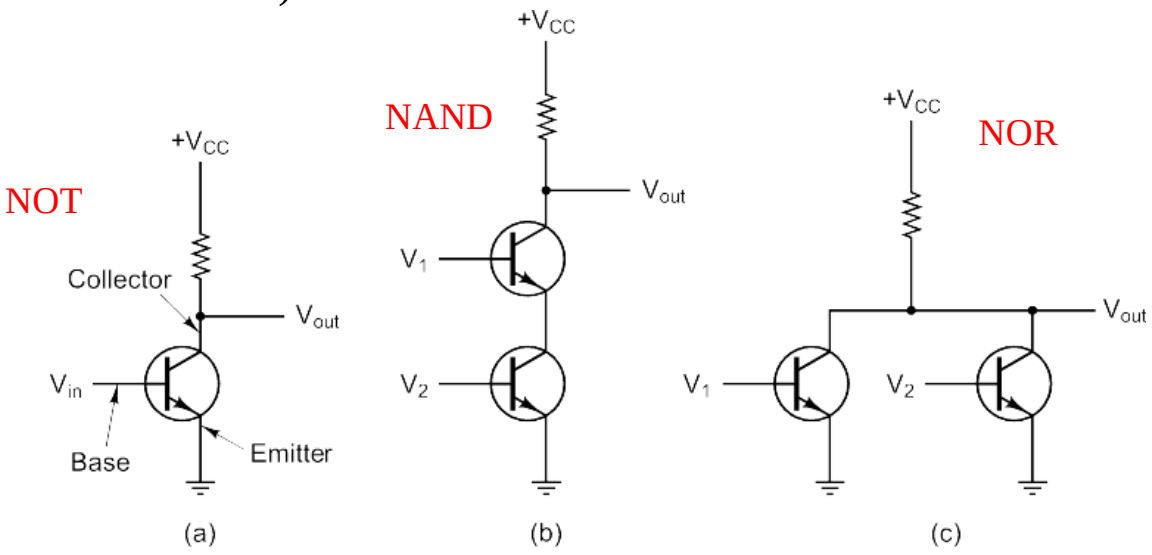
0 / FALSO / [0..1] Volt

1 / VERO / [2..5] Volt

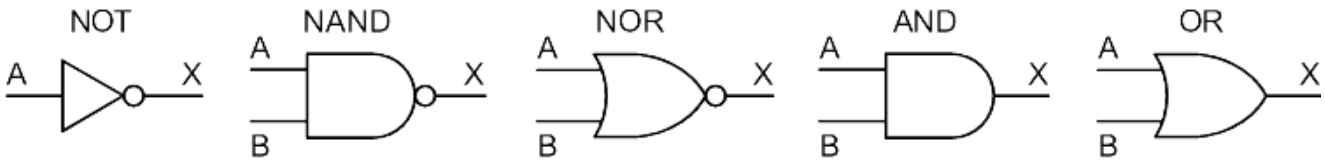


I mattoncini di base dei circuiti digitali sono le cosiddette **porte logiche**, ovvero semplici circuiti in grado di calcolare le principali operazioni dell'algebra Booleana.

Dal punto di vista elettronico, le porte logiche sono realizzate con elementi attivi chiamati "**transistor**" che operando come interruttori automatici. Un segnale sulla base ha l'effetto di mettere in comunicazione diretta emettitore e collettore (che in stato di riposo sono elettricamente isolati).



# Le 5 porte logiche di base




| A | X |
|---|---|
| 0 | 1 |
| 1 | 0 |

(a)

| A | B | X |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

(b)

| A | B | X |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

(c)

| A | B | X |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

(d)

| A | B | X |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

(e)

Una **tabella della verità**, è una tabella dove viene indicato l'output di un particolare circuito per ogni possibile configurazione di input.

*Quante sono le configurazioni di input di un circuito con N ingressi ?*
*Come enumerare tutte le possibili configurazioni di input ?*

Le tabelle della verità delle 5 porte logiche di base sono riportate in figura:

- La porta **NOT** inverte (o nega) il segnale di ingresso
- La porta **AND** ha output vero se e solo se entrambi gli input sono veri
- La porta **OR** ha output vero se almeno uno dei due input è vero

- La porta **NAND** equivale a un AND la cui uscita è negata (l'output è falso solo se entrambi gli input sono veri)
- La porta **NOR** equivale a un OR la cui uscita è negata (l'output vero solo se entrambi gli input sono falsi).

Sebbene le porte NAND e NOR sembrino più complicate rispettivamente di AND e OR, la realizzazione in termini di transistor di tali porte è più semplice (vedi lucido precedente).

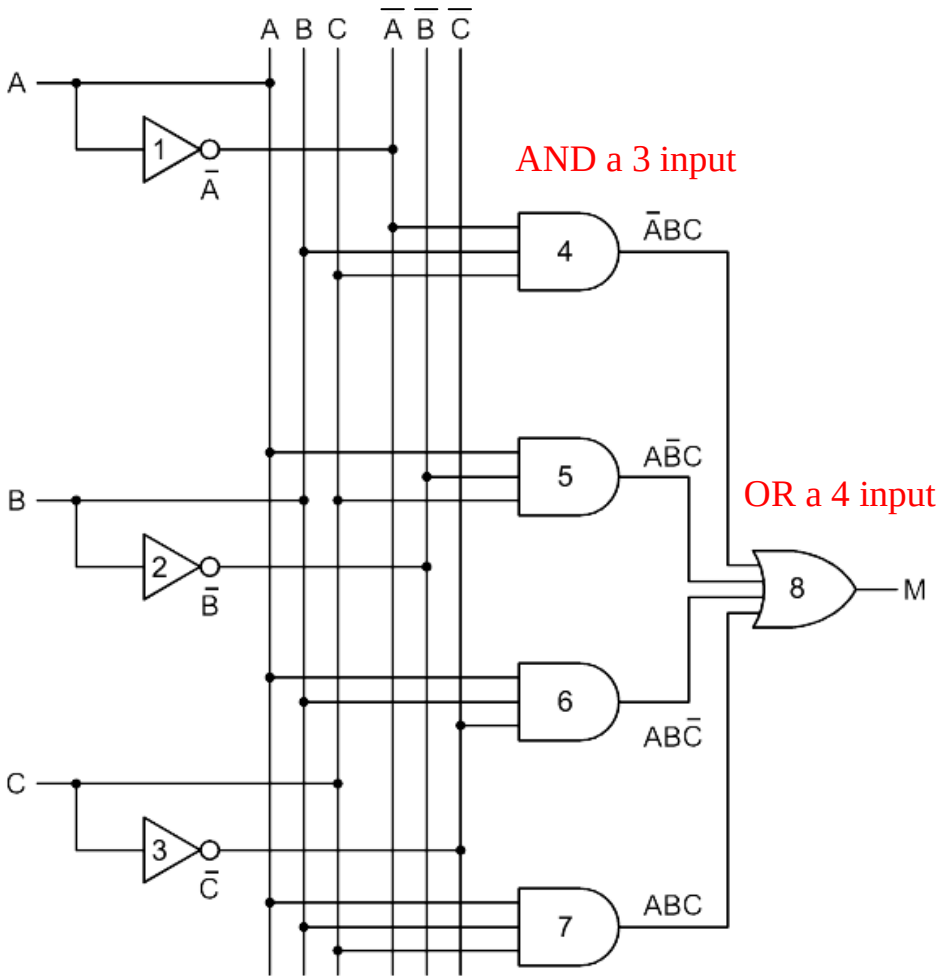
# Circuiti digitali e algebra Booleana

L'output di un circuito digitale può essere descritto, oltre che da una tabella della verità, anche come **funzione booleana** dei suoi input:

$$M = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

| A | B | C | M |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

(a)




(b)

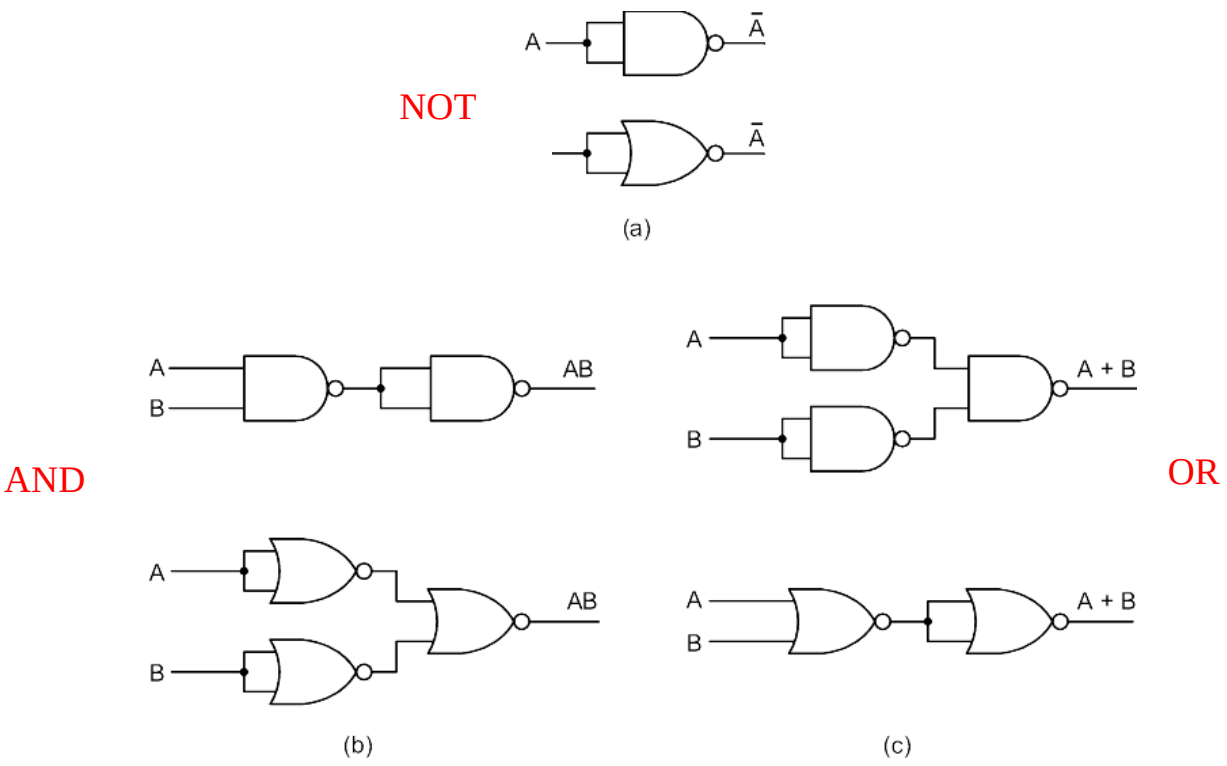
In generale, si può descrivere una funzione di **N** variabili dando come output la "somma" (OR) di al massimo $\mathbf{2^N}$ termini prodotto (AND) di **N** variabili.

# Circuiti digitali e algebra Booleana (2)

L'esempio del lucido precedente ci fornisce un modo semplice per **realizzare circuiti digitali** in grado di calcolare qualsiasi funzione Booleana:

1. Scrivere la tabella di verità per la funzione
2. Disporre di invertitori (NOT) per generare il complemento di ogni input
3. Introdurre una porta AND per ogni termine con un 1 nella colonna dell'output
4. Collegare le porte AND agli input appropriati
5. Inviare l'output di tutte le porte AND in una porta OR

Dal punto di **vista costruttivo** (Hardware) è preferibile realizzare un circuito utilizzando un solo tipo di porta logica (preferibilmente NAND o NOR). La figura sottostante mostra come realizzare le tre porte fondamentali (NOT, AND e OR) utilizzando solo porte NAND oppure solo porte NOR:




*Come dimostrare **l'equivalenza** di questi circuiti ?*

# Equivalenza dei circuiti

Al fine di ridurre i costi, lo spazio occupato e il consumo di corrente, i progettisti di circuiti digitali cercano di ridurre al minimo il numero di porte logiche necessarie per realizzare un determinato circuito.

Le leggi **dell'algebra Booleana** consentono spesso di **semplificare** espressioni Booleane complesse ottenendo espressioni equivalenti.

| Name | AND form | OR form |
|---|---|---|
| Identity law | $1A = A$ | $0 + A = A$ |
| Null law | $0A = 0$ | $1 + A = 1$ |
| Idempotent law | $AA = A$ | $A + A = A$ |
| Inverse law | $A\overline{A} = 0$ | $A + \overline{A} = 1$ |
| Commutative law | $AB = BA$ | $A + B = B + A$ |
| Associative law | $(AB)C = A(BC)$ | $(A + B) + C = A + (B + C)$ |
| Distributive law | $A + BC = (A + B)(A + C)$ | $A(B + C) = AB + AC$ |
| Absorption law | $A(A + B) = A$ | $A + AB = A$ |
| De Morgan's law | $\overline{AB} = \overline{A} + \overline{B}$ | $\overline{A + B} = \overline{A}\,\overline{B}$ |

**Attenzione:** $\overline{A}\,\overline{B} \neq \overline{AB}$ **!**

Le **mappe di Karnough** costituiscono un ulteriore strumento per l'analisi e la semplificazione di circuiti digitali.

# Equivalenza dei circuiti: esempio

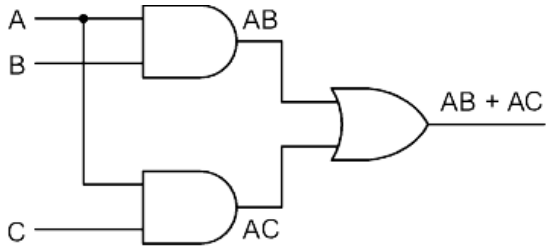
Data l'espressione Booleana:

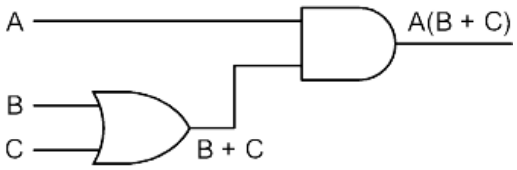
$$M = AB + AC$$

può essere applicata la proprietà distributiva ottenendo:

$$M = A(B + C)$$

che può essere implementata con 2 porte invece che con 3. *L'analisi delle rispettive tabelle della verità è una* "prova del 9" *dell'equivalenza*.




| A | B | C | AB | AC | AB + AC |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

(a)

| A | B | C | A | B + C | A(B + C) |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

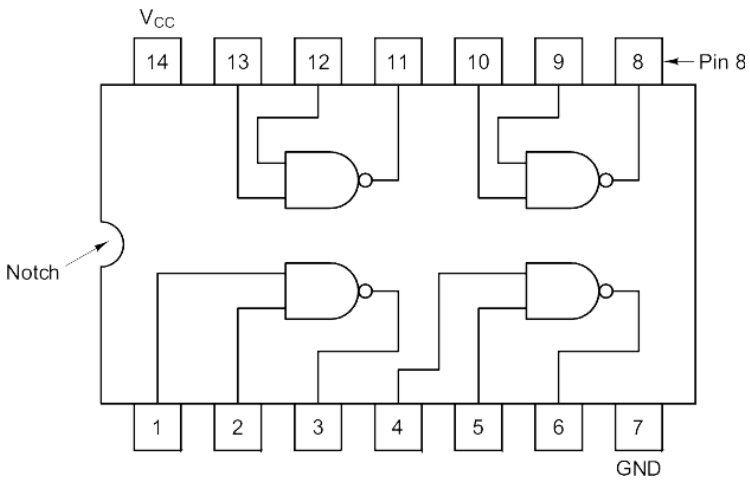
(b)

# Circuiti Integrati

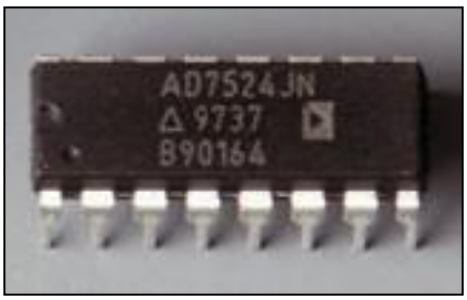
Le porte logiche non vengono prodotte o vendute individualmente, ma in unità chiamate **circuiti integrati** (**IC**) o più genericamente chip. Un circuito integrato è un pezzetto rettangolare di silicio di circa 5 mm x 5 mm su cui vengono realizzate le porte. I circuiti integrati vengono poi montati in contenitori (package) ceramici larghi da 5 a 15 mm e lunghi da 20 a 50 mm. Nella maggior parte dei circuiti integrati i contatti di uscita sono realizzati in due file parallele (DIP = Dual Inline Packages) poste ai lati del package.

I chip possono essere classificati grossolanamente sulla base del numero di porte:

- Circuito **SSI** (Small Scale Integrated): da 1 a 10 porte
- Circuito **MSI** (Medium Scale Integrated): da 10 a 100 porte
- Circuito **LSI** (Large Scale Integrated): da 100 a 100.000 porte
- Circuito **VLSI** (Very Large Scale Integrated): > 100.000 porte




*Un semplice circuito SSI con Quattro porte NAND*

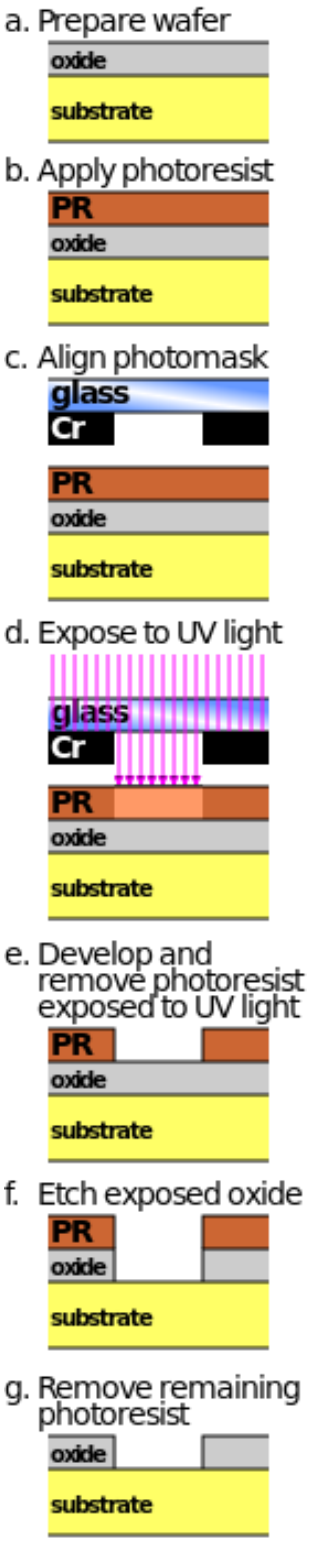



*Package di un Chip SSI*

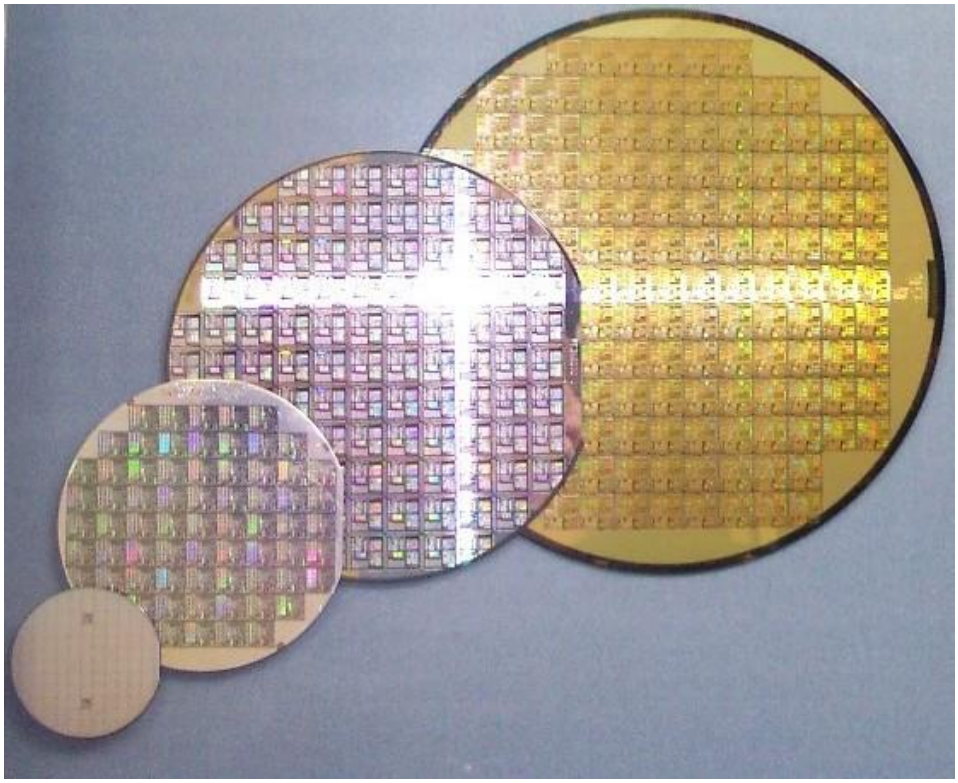
# Fabbricazione Circuiti Integrati

Il processo di fabbricazione prevede i seguenti passi:

- Fusione di sabbie silicee e realizzazione dei wafer (dischi). Su uno stesso wafer sono realizzati molti chip.
- Sul wafer sono depositati (crescita per epitassia) sottili strati di materiali diversi: semiconduttori, ossidi, isolanti, metalli.
- Il deposito dei suddetti materiali non è uniforme sull'intera superficie del wafer, ma avviene selettivamente grazie a un processo di **fotolitografia**:
  - ricopertura dell'intero wafer con un layer fotosensibile (fotoresist)
  - il layer fotosensibile viene esposto a una luce ultravioletta che passa attraverso le zone chiare di una maschera (vetro + cromo) ad altissima definizione.
  - le zone illuminate divengono solubili e il fotoresist può essere facilmente asportato lasciando scoperte area sottostanti che possono essere trattate selettivamente.
  - Attraverso lavaggio chimico (etching) si possono rimuovere layer sottostanti.
- Il processo di depositito/fotolitografia può essere ripetuto decine/centinaia di volte per chip complessi.
- Separazione dei chip dello stesso wafer e montaggio in contenitore.



# Fabbricazione Circuiti Integrati (2)



Al 2017 la tecnologia di punta è in grado di integrare 21 miliardi di transistor in un singolo chip (rif. GPU GV100 Volta).

Rif: https://en.wikipedia.org/wiki/Transistor_count

Al 2017 la dimensione minima dei transistor in chip commerciali è di 10 nanometri.

Rif: https://en.wikipedia.org/wiki/10_nanometer

# Tipologie di circuiti digitali

Circuito combinatorio: il valore istantaneo delle uscite è determinato univocamente da valori degli ingressi il circuito.

- Assenza di connessioni di feedback (all'indietro)
- Analisi circuitale semplice: si propagano semplicemente i valori degli ingressi verso le uscite.

Circuito sequenziale: il valore istantaneo delle uscite dipende, oltre che dai valori di ingresso, anche dalla storia passata del circuito.

- Connessioni all'indietro che riportano uscite verso gli ingressi
- Richiedono temporizzazioni
- Possono essere sincroni e asincroni
- Analisi circuitale più complessa
- Attraverso circuiti sequenziali si realizzano macchine a stati finiti (implementazione in hardware di automi a stati finiti)

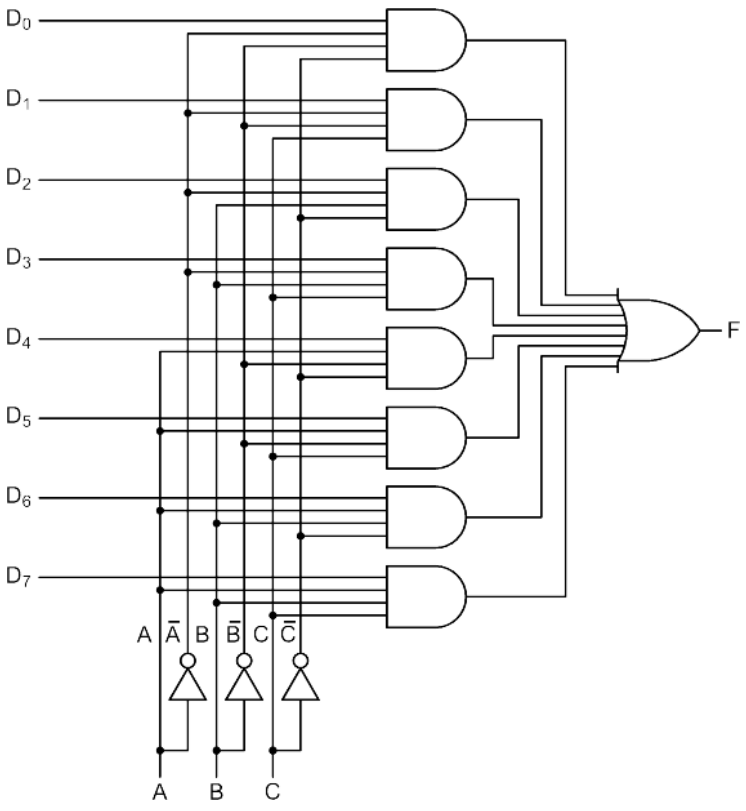
# Alcuni circuiti MSI (combinatori)

Partendo dall'osservazione che con semplici porte NAND è possibile realizzare qualsivoglia circuito, sembrerebbe logico creare circuiti integrati con una grande quantità di porte NAND indipendenti. D'altro canto un chip con **5 milioni** di porte **NAND** indipendenti, richiederebbe **15.000.002** piedini (2 per massa e alimentazione) e quindi, tenendo conto di una spaziatura DIP standard tra i piedini, sarebbe lungo circa **18 Km** !

Per questo motivo, allo scopo di aumentare il rapporto porte/piedini, i chip MSI, LSI e VLSI vengono realizzati implementando internamente circuiti più complessi e collegando in uscita solo i piedini "rilevanti".

## Multiplexer

Un multiplexer è un circuito con $2^n$ INPUT DATI, 1 OUTPUT DATI e n input di CONTROLLO. Gli input di controllo selezionano quale tra i $2^n$ input dati deve essere "portato" in uscita. In figura, i tre input di controllo A, B e C (interpretati come numero binario di 3 cifre) indicano quale degli input $D_0$...$D_7$ deve essere portato su F.

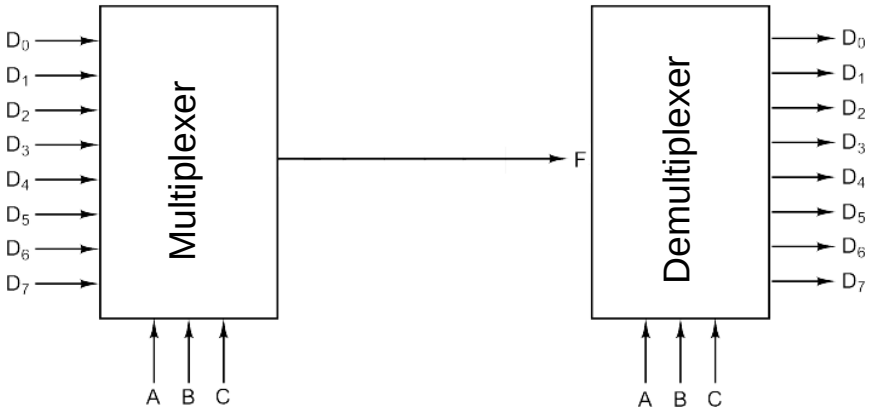


# Alcuni circuiti MSI (2)

## Demultiplexer

Si tratta del circuito INVERSO del multiplexer. In questo caso 1 INPUT DATI viene "smistato" su una delle $2^n$ LINEE DI OUTPUT (in base alla selezione degli n input di CONTROLLO).

Un circuito costituito da un **multiplexer** + **1 linea dati** + un **demultiplexer** può essere usato per metter in comunicazione più utenti su una stessa linea dati (es. combinatori telefonici).




*Esercizio: disegnare il circuito logico di un demultiplexer con 8 linee dati*

## Decoder (decodificatore)

Un decoder è un circuito che riceve in INPUT n segnali/bit (che vengono trattati come numero binario di n cifre); Il decoder ha $2^n$ linee di OUTPUT, di cui una sola viene impostata a 1; la linea impostata a 1 è quella selezionata dal numero binario in input.

Il decoder può esser visto come un traduttore dal sistema binario posizionale (INPUT) ad un sistema (non posizionale) dove ogni bit di uscita rappresenta un carattere (digit).

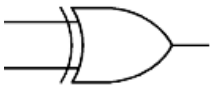
*Esercizio: disegnare il circuito logico di un decoder con 8 linee dati*

Ovviamente esiste il circuito inverso anche del decoder (chiamato **encoder**).

# Alcuni circuiti MSI (3)

## Comparatore

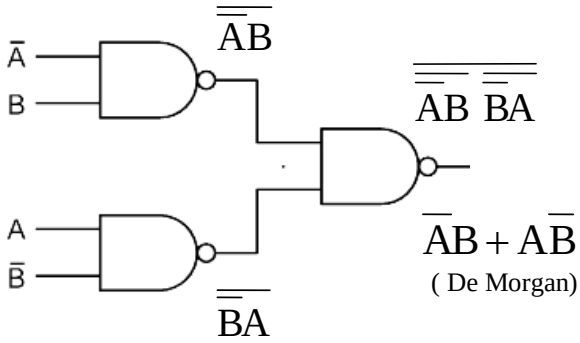
Un comparatore, come dice il nome, confronta due parole di input e produce un output che indica l'uguaglianza/ineguaglianza degli input. Il circuito può essere realizzato utilizzando **porte XOR**. Il simbolo della porta XOR, la sua tabella della verità e un semplice circuito che lo realizza a partire dalle 3 porte NAND, sono qui riportati.



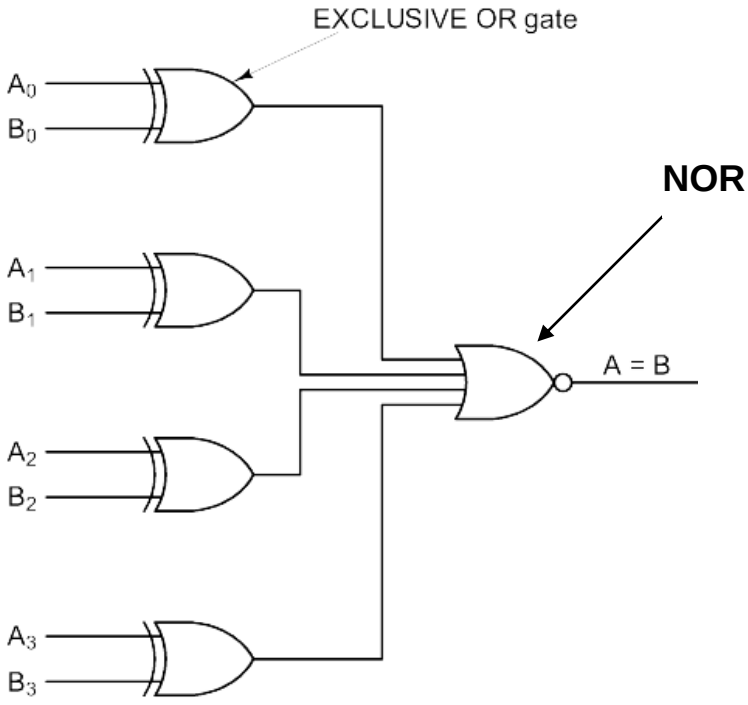
**XOR** (OR Esclusivo)

Simbolo ⊕

| A | B | XOR |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |




La figura mostra un comparatore di due parole A e B di 4 bit ciascuna. Ogni porta XOR confronta una coppia di bit; se almeno una delle coppie è diversa uno degli input della porta NOR vale 1 e pertanto l'output del comparatore vale 0. Se invece tute le coppie sono uguali tutti gli input del NOR valgono 0 e quindi l'output 1.



# Array di porte logiche programmabili

Un array di logiche programmabili detto **PLA** (Programmable Logic Array) è un chip costituito da un numero (anche molto elevato) di porte collegate internamente a seconda dei desiderata del progettista; un PLA è in grado quindi di calcolare funzioni Booleane arbitrarie.

In un circuito con n INPUT, m OUTPUT e k UNITA' INTERNE:

- Di ciascuno degli n input viene internamente generato il complemento (2×n linee di input interno).
- Il chip contiene un array di k porte AND a 2×n input (ciascun porta AND è inizialmente collegata a tutte le 2×n linee di input).
- Il chip contiene m porte OR a k input (ciascun porta OR è inizialmente collegata a tutte le k uscite delle porte AND).

Utilizzando la strategia vista in precedenza per implementare funzioni arbitrarie come OR di AND degli input, è possibile determinare quali collegamenti preservare al fine di realizzare il circuito desiderato.

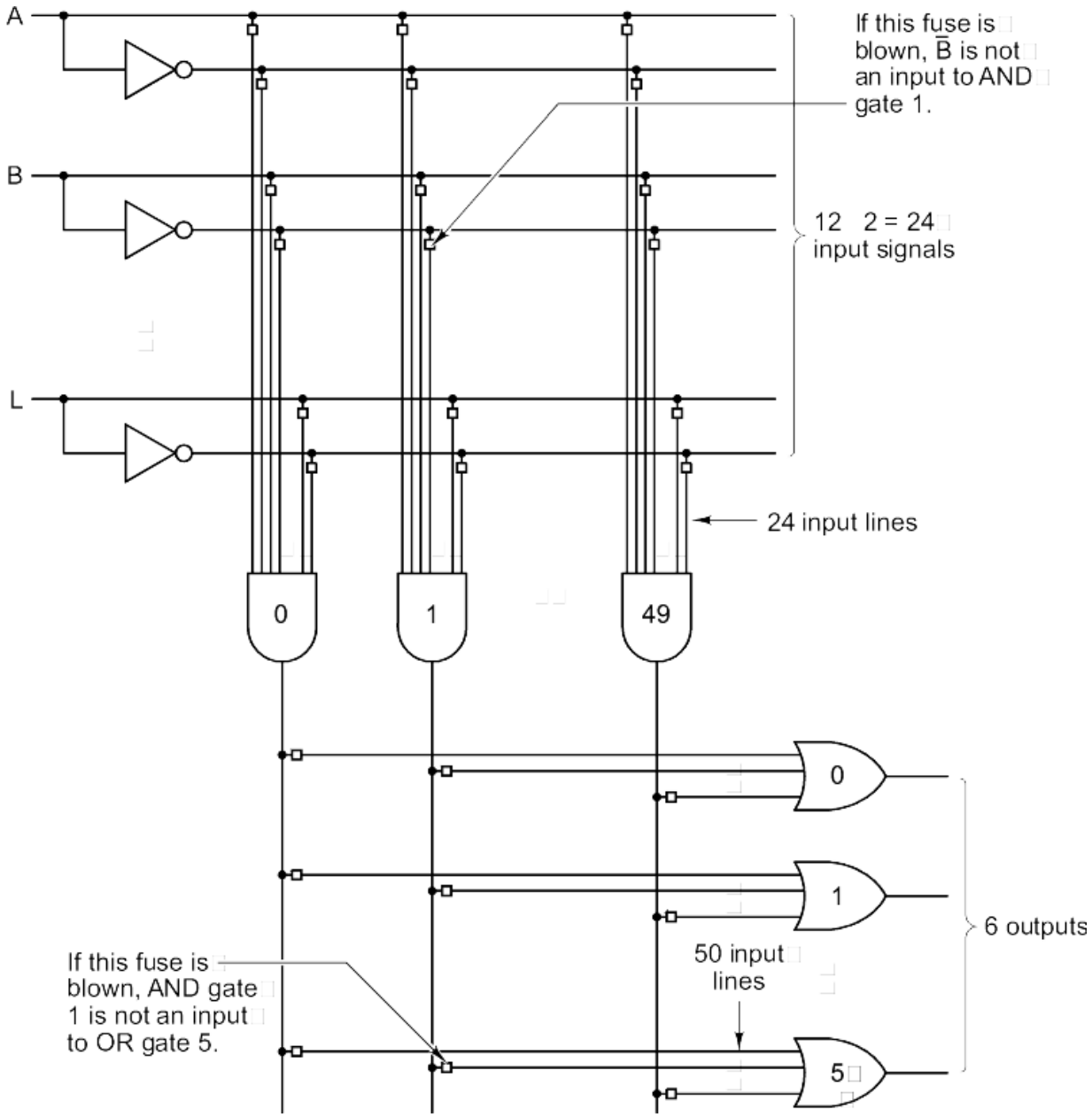
I primi modelli di PLA furono prodotto a livello industriale verso la fine degli anni '70. La loro evoluzione portò rapidamente all'introduzione delle prime FPGA.

Un **FPGA** (Field Programmable Gate Array) è un PLA complesso, che può contenere anche milioni di porte logiche, in cui i collegamenti tra le porte vengono definiti dallo sviluppatore via software.

I maggiori produttori di FPGA (Xilinx, Altera) mettono a disposizione potenti sistemi di sviluppo che a partire da linguaggi di programmazione o schemi simbolici, sono in grado di programmare automaticamente il Chip.

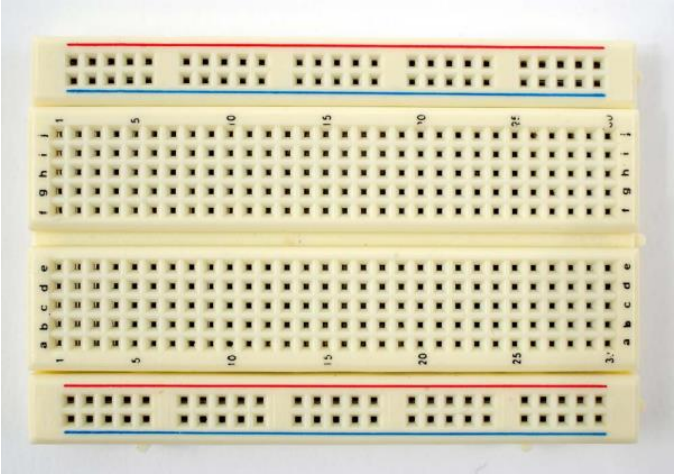
# Un esempio di PLA

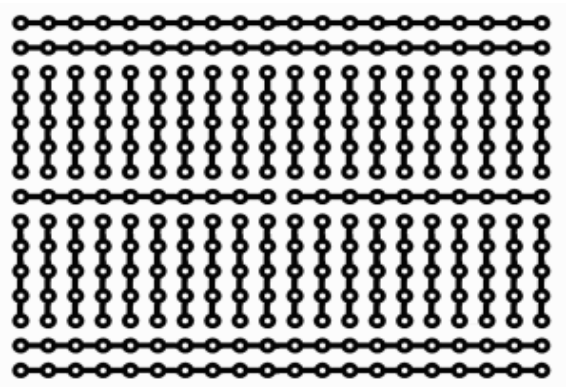
n = 12, m = 6, k = 50

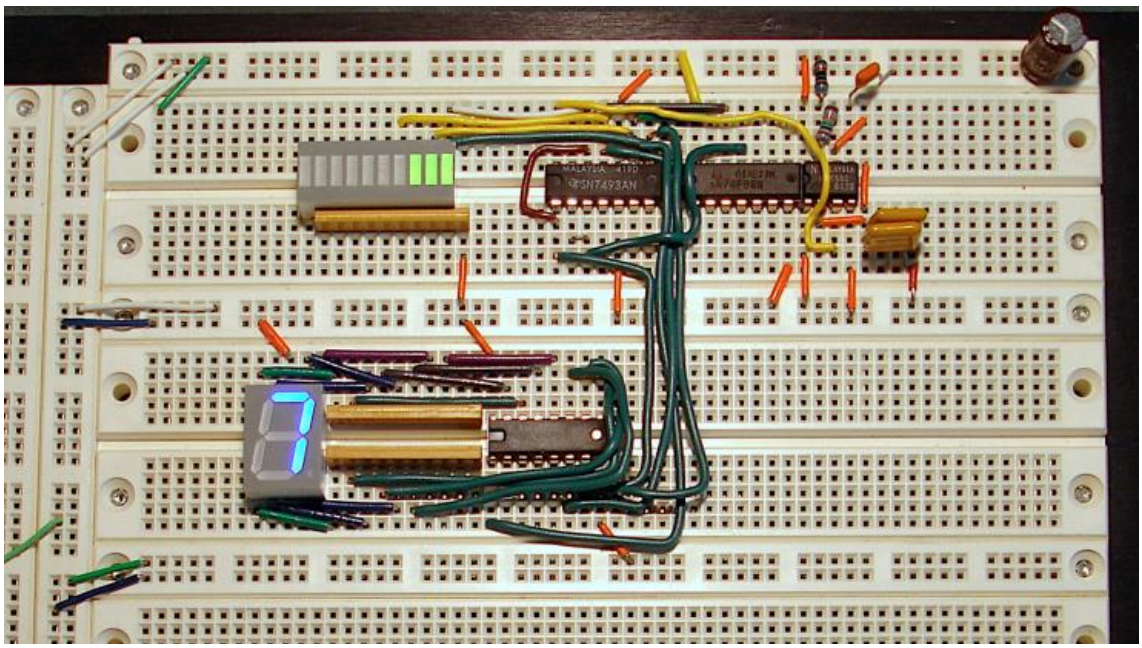


# Prototipazione di circuiti

Per la prototipazione di semplici circuiti (digitali ma anche analogici) si utilizzano solitamente le cosiddette breadboard: ovvero basette sperimentali precablate dove possono essere inseriti agevolmente componenti con passo standard (DIP).







# Printed Circuit Board

Un PCB (o circuito stampato) è una scheda utilizzata per il supporto meccanico e per il collegamento di componenti elettronici. È costituito da un substrato isolante (es. vetroresina) e da uno o più strati conduttivi in rame (layer) che consentono di collegare i componenti tramite piste (o vie).

Il montaggio tradizionale prevede piccole piazzole (pad) con fori passanti nelle quali sono saldati i piedini dei componenti.

Il montaggio SMT (Surface Mounting Technology), che ha preso piede negli anni 80, prevede il montaggio superficiale (senza fori) dei componenti e consente di ridurre il package dei componenti e automatizzare il montaggio (macchine pick & place).

# Circuiti aritmetici

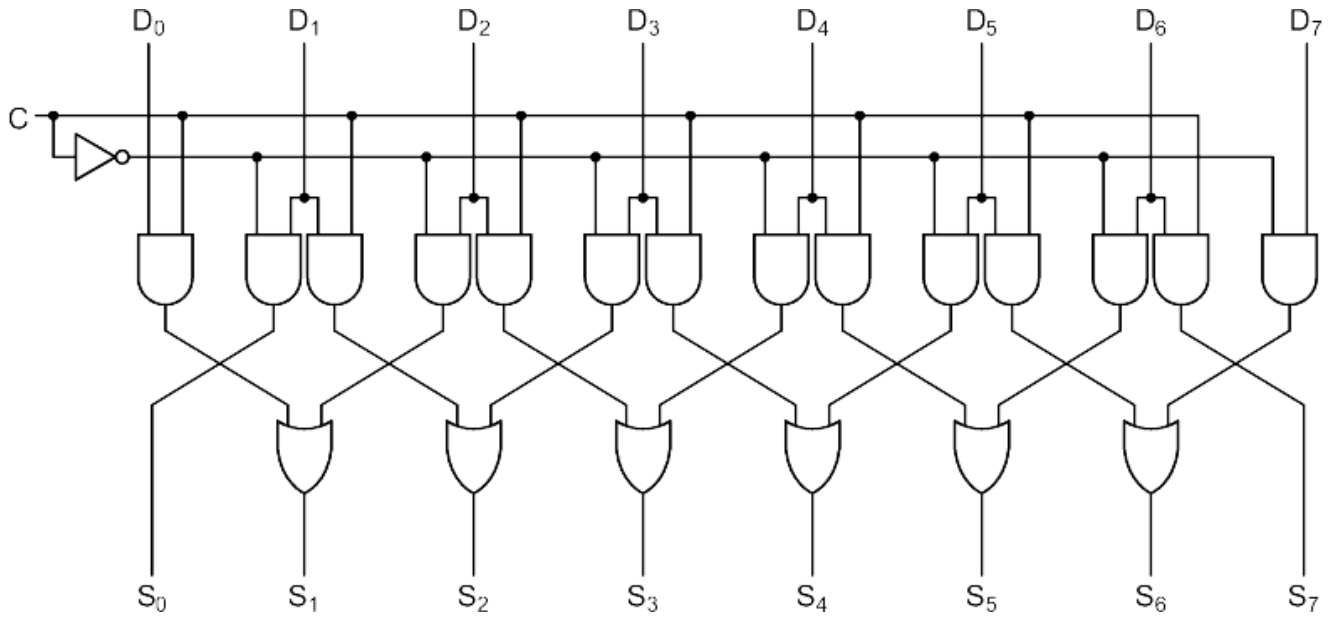
Sono circuiti specializzati nel calcolo di semplici operazioni aritmetiche dei loro input. Nel seguito vediamo alcuni semplici circuiti combinatori.

## Shifter (traslatore)

Uno shifter è un circuito con n INPUT ($D_0$..$D_{n-1}$) e n OUTPUT ($S_0$..$S_{n-1}$). I bit in output sono esattamente la copia di quelli in input traslati tutti di una posizione a destra o a sinistra. La direzione (destra/sinistra) è impostata da un bit di controllo C.

A seconda della direzione di traslazione il bit più significativo o meno significativo dell'output non riceverà valore da un bit di input. Tale bit dell'output per convenzione viene impostato a 0.

In figura è mostrato lo schema di uno shifter a 8 bit:




Per ogni bit di input (tranne il primo e l'ultimo) sono presenti due porte AND delle quali però solo una viene abilitata a seconda del valore di C. Tale porte, che copiano in avanti il valore del bit corrispondente, sono collegate in modo sfasato alle porte OR del livello successivo e quindi eseguono la traslazione desiderata.

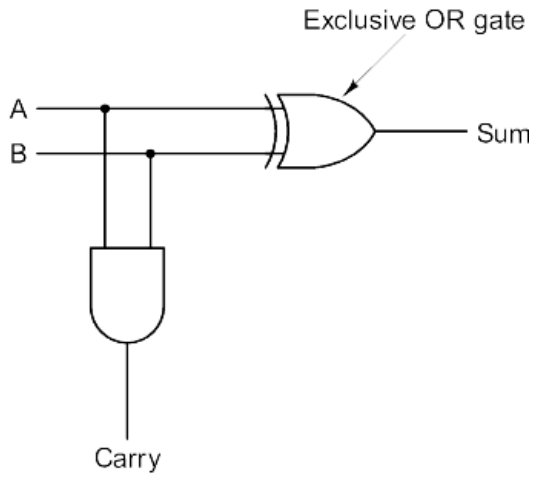
# Circuiti aritmetici (2)

## Adder (sommatore)

La somma tra numeri binari è un'operazione fondamentale per qualsiasi calcolatore. Per la realizzazione di un sommatore a n bit, vengono utilizzati n "mattoncini" elementari denominati full-adder a 1 bit (sommatore a 1 bit) realizzati a loto volta a partire da half-adder a 1 bit (semi sommatore a 1 bit).

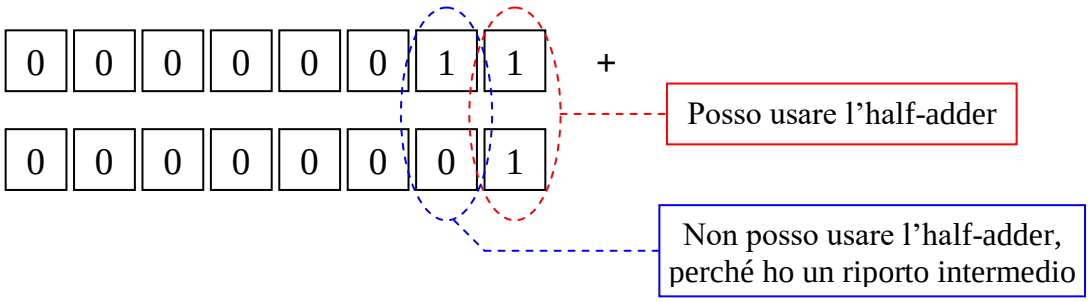
La tabella della verità e il circuito dell'**half-adder a 1 bit** sono qui riportati:

| A | B | Sum | Carry |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |




- La **somma** (Sum) vale 1 solo se i 2 bit di input sono diversi (XOR).
- Il **riporto** (Carry) vale 1 solo se entrambi gli input sono 1 (AND).

L'half-adder a 1 bit funziona in realtà solo per i bit meno significativi di una parola, ma NON per i bit "centrali" per i quali ci può essere un riporto pendente in input.

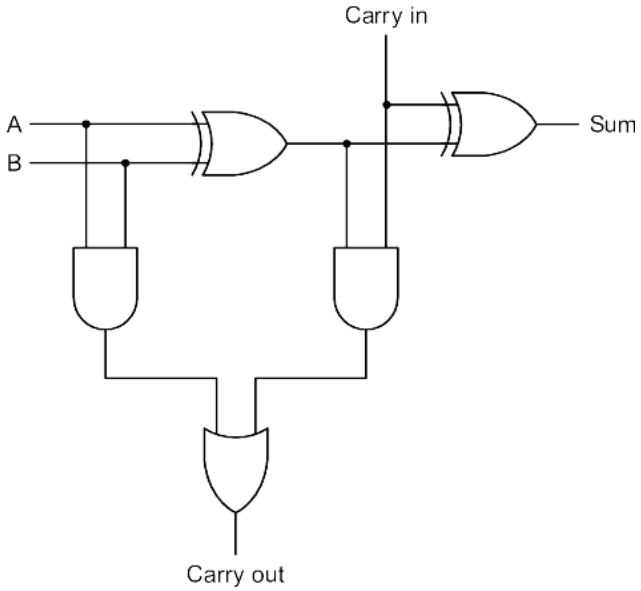



A tale scopo è necessario realizzare un **full-adder a 1 bit** che prevede un ulteriore input per il riporto intermedio:

# Circuiti aritmetici (3)

## Full-adder 1 bit

| A | B | Carry in | Sum | Carry out |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |




- La **somma** (Sum) vale 1 quando un numero dispari (1 o 3) di input (compreso il riporto in ingresso) vale 1.
- Il **riporto** in uscita (Carry out) vale 1 se il numero di input a 1 (compreso il riporto in ingresso) è maggiore o uguale a 2.

Il circuito può essere derivato dalla tabella della verità a seguito di alcune semplificazioni; infatti, indicando con C il riporto intermedio Carry in:

$$\text{Sum} = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + ABC$$

$$\text{Sum} = \overline{C}\left(\overline{A}B + A\overline{B}\right) + C\left(AB + \overline{A}\,\overline{B}\right)$$

$$\text{Sum} = \overline{C}\left(\overline{A}B + A\overline{B}\right) + C\left(\overline{\overline{A}B + A\overline{B}}\right)$$

$$\text{Sum} = C \oplus (A \oplus B)$$

Ricorda che : $\overline{A}B + A\overline{B} = A \oplus B$

Essendo $\overline{\overline{A}B + A\overline{B}} = AB + \overline{A}\,\overline{B}$

Dimostrarlo !

$$\text{Carry out} = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

$$\text{Carry out} = C\left(\overline{A}B + A\overline{B}\right) + AB\left(C + \overline{C}\right)$$

$$\text{Carry out} = C(A \oplus B) + AB$$

Ricorda che : $C + \overline{C} = 1$

# Circuiti aritmetici (4)

## Full-adder n bit

La realizzazione di un full-adder a n bit consiste semplicemente nell'utilizzo di n full-adder a 1 bit in parallelo. Ovviamente i segnali di riporto devono essere collegati opportunamente tra loro:

- il Carry out del bit i-esimo deve essere collegato al Carry in del bit i+1-esimo, per ogni i = 0 .. n-2

## ALU (Arithmetic Logic Unit)

Un'unità aritmetico logica è un circuito capace di eseguire operazioni aritmetiche e logiche elementari su due parole di input A e B (di lunghezza **n** bit). La maggior parte delle ALU esegue:

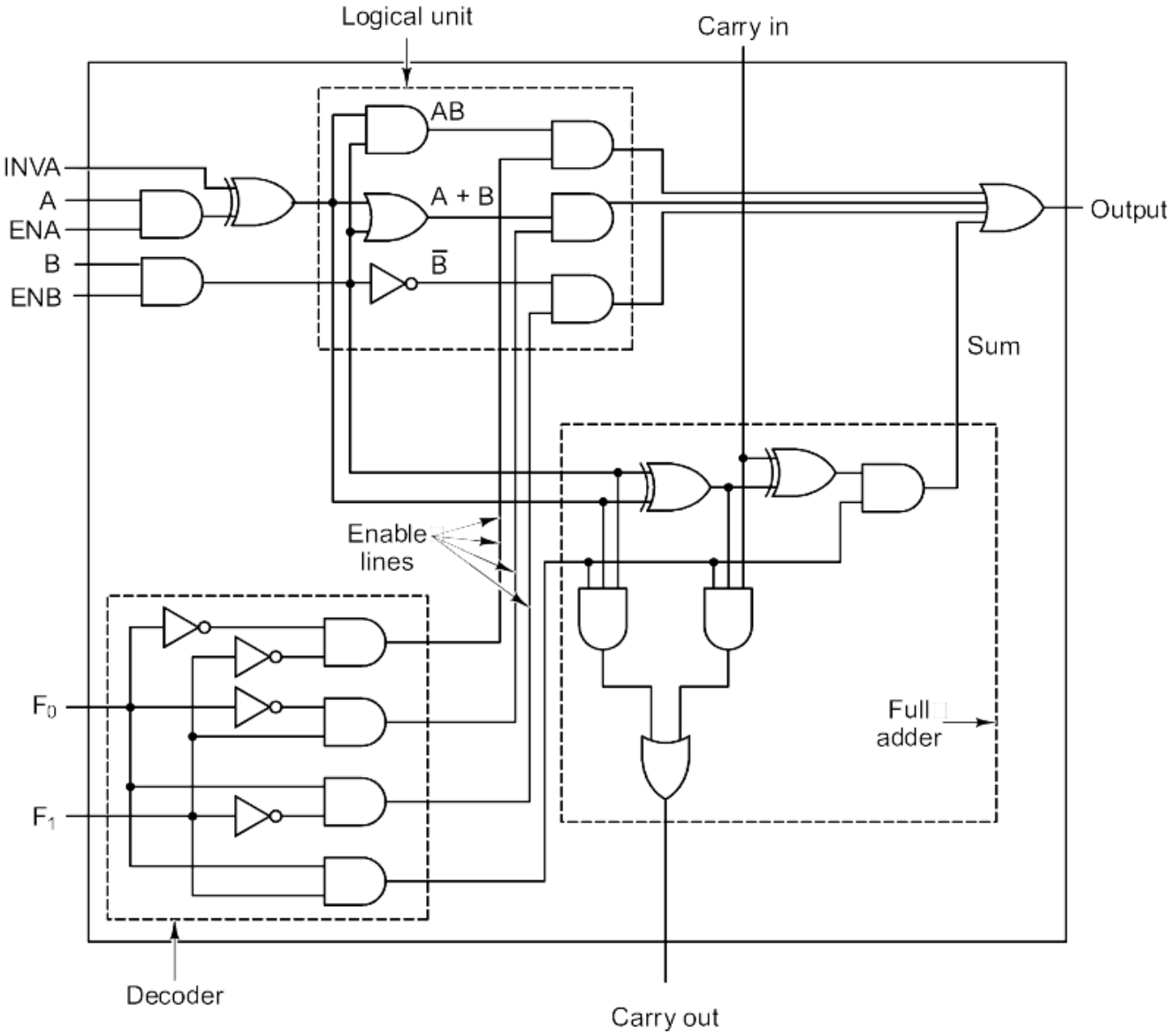
1. A **AND** B (AND logico bit a bit)
2. A **OR** B ( OR logico bit a bit)
3. $\overline{B}$ (complemento o negazione bit a bit)
4. A + B (somma aritmetica)

La figura (nel lucido successivo) mostra una ALU a 1 bit. Nel circuito sono evidenziati i diversi blocchi strutturali:

- Nell'angolo inferiore a sinistra, troviamo un **decodificatore** (decoder) a 2 bit (input $F_0$ ed $F_1$) le cui 4 linee di output vengono utilizzate per l'attivazione delle 4 diverse operazioni.
- In alto troviamo l'**unità logica** in grado di calcolare le 3 operazioni logiche. 3 porte AND sono necessarie per dirigere 1 solo output verso l'uscita (in base alla funzione selezionata).
- In basso a destra troviamo un **full-adder a 1 bit** che si differenzia da quello visto in precedenza per l'aggiunta di 3 porte AND che hanno il compito di abilitare o meno le uscite nel caso in cui la funzione "+" sia quella selezionata.

# Circuiti aritmetici (5)

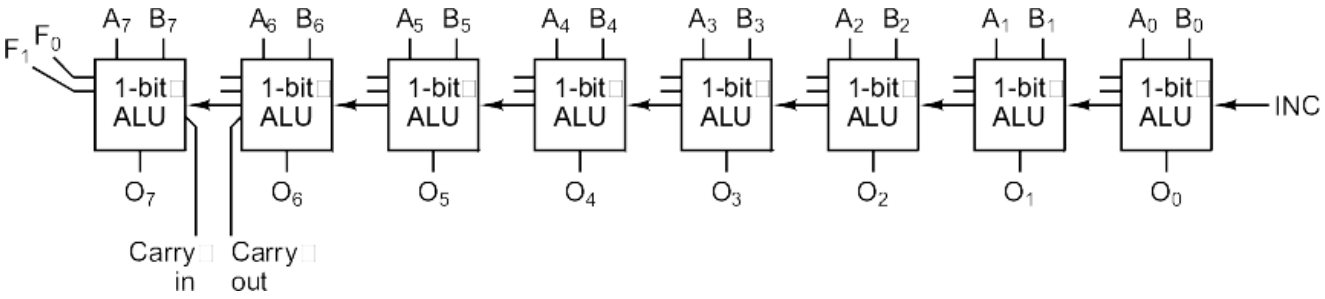
## Un circuito per una ALU a 1 bit




- Gli input ENA e ENB possono essere utilizzati per forzare a **0** gli input A e B.
- E' inoltre possibile avere come input $\overline{A}$ al posto di A semplicemente impostando a 1 l'input **INVA** (inverti A)
- Una sola delle quattro funzioni viene portata in output e fatta passare dalla porta **OR** a quattro ingressi.

# Circuiti aritmetici (6)

## Realizzazione di una ALU a n bit (nell'esempio n = 8)




Questo circuito consente di eseguire le operazioni aritmetiche e logiche di base su due parole **A** e **B** di lunghezza 8 bit e di produrre il risultato nella parola **O** (anch'essa di 8 bit).

Il circuito è realizzato concatenando 8 ALU a 1-bit:

- Gli input di selezione ($F_0$ ed $F_1$) sono collegati ai corrispondenti input di tutte le 1-bit ALU.
- Il riporto intermedio viene propagato da una 1-bit ALU alla successiva nella catena. ATTENZIONE: la propagazione non è istantanea.
- L'ulteriore ingresso INC (collegato alla 1-bit ALU relativa al bit meno significativo) consente **incrementare** il risultato di 1 unità nel caso di addizioni.
  - Tale input non richiede nessuna circuiteria aggiuntiva, in quanto coincide semplicemente con l'input Carry in dell'ultima 1-bit ALU.
  - D'altro canto la possibilità di calcolare A+1, A+B+1 può essere molto utile nella pratica (es. istruzione assembler INC).

*Quante porte logiche sono necessarie per realizzare tale circuito ?*

*Utilizzando solo porte NAND a 2 input, quante porte sono necessarie ?*

# Circuiti aritmetici (6)

## Moltiplicazione binaria

```
      1011   (this is 11 in decimal)
    × 1110   (this is 14 in decimal)
    ======
      0000   (this is 1011 × 0)
     1011    (this is 1011 × 1, shifted left one position)
    1011     (this is 1011 × 1, shifted left two positions)
+ 1011       (this is 1011 × 1, shifted left three positions)
=========
 10011010   (this is 154 in binary)
```

Sono necessarie semplici operazioni di somma e shift. In teoria non occorre un circuito specifico e la moltiplicazione può essere implementata con microprogramma (o macchina a stati) usando semplice ALU.

In questo modo però per moltiplicare due numeri a 64 bit occorrono 64 (shift + somme).

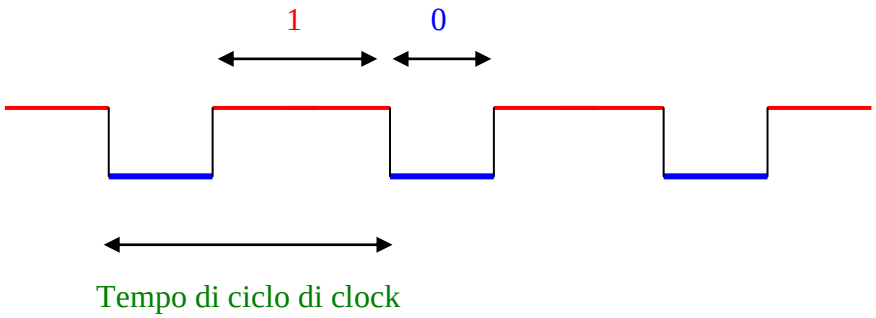
Moltiplicatori (hardware/software) ottimizzati richiedono un minor numero di somme che possono essere parallelizzate.

# Circuiti Sequenziali e Memorie

## Temporizzazioni e clock

Nella maggior parte dei circuiti digitali complessi è necessario stabilire l'ordine con cui si verificano gli eventi. E' quindi necessaria una base dei tempi in grado di **sincronizzare** il susseguirsi delle diverse operazioni.

Un **clock** (orologio) è un circuito, basato su un componente denominato quarzo, che emette impulsi digitali (onda quadra formata da stati 0 e 1 alternati) con una lunghezza specifica e con un preciso intervallo di tempo tra impulsi successivi.




- Il **tempo di ciclo di clock** specifica l'intervallo di tempo tra i fronti corrispondenti (es. discesa) in due impulsi successivi. Il tempo si misura nei calcolatori in sottomultipli di secondo:
  **1 ms** (millisecondo) = $1 \cdot 10^{-3}$ secondi
  **1 μs** (micro secondo) = $1 \cdot 10^{-6}$ secondi
  **1 ns** (nano secondo) = $1 \cdot 10^{-9}$ secondi

- La **frequenza di clock** specifica il numero di periodi di clock per unità di tempo (ovvero per secondo). La frequenza si calcola come inverso del tempo di ciclo di clock. L'unità di misura è l'**Hertz**, di cui si utilizzano per i calcolatori, i multipli:

$$\text{frequenza di clock} = \frac{1}{\text{tempo di ciclo di clock}}$$

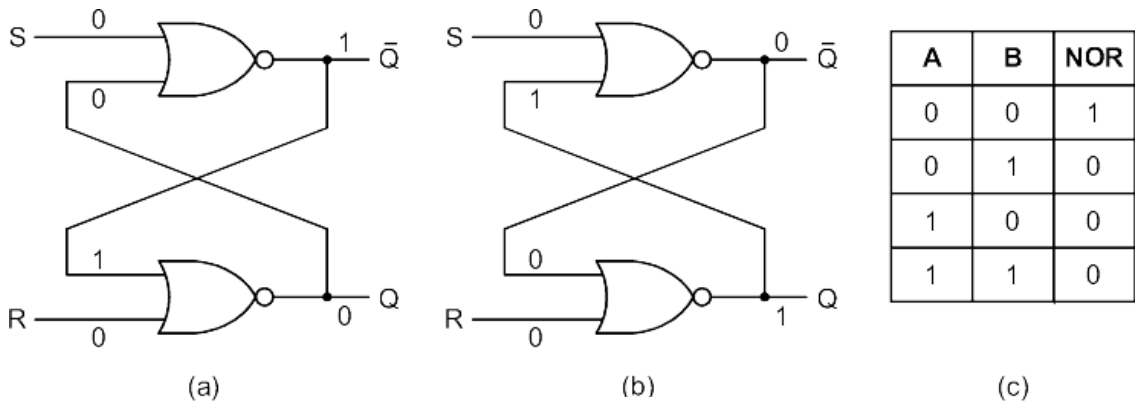
**1 KHz** (KiloHertz) = $1 \cdot 10^{3}$ Hertz
**1 MHz** (MegaHertz) = $1 \cdot 10^{6}$ Hertz
**1 GHz** (GigaHertz) = $1 \cdot 10^{9}$ Hertz

# Latch e Flip-Flop

## Il Latch: un circuito bistabile

Un circuito **bistabile** è un circuito in grado di mantenersi in posizione di stabilità in due differenti stati. Il cambio di stato è provocato da un particolare segnale o impulso; a seguito dell'impulso il circuito si mantiene nello stato raggiunto a meno che non vengano inviati ulteriori segnali/impulsi.

Tramite due semplici porte NOR è possibile realizzare un circuito bistabile denominato **Latch SR**:




| A | B | NOR |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

Stato stabile 0 — Stato stabile 1

- L'input **S** (setting) è utilizzato per impostare lo stato 1
- L'input **R** (reset) è utilizzato per impostare lo stato 0
- **Q** è l'output (stato) mentre $\overline{Q}$ è il suo complemento

Quando **S** ed **R** sono 0 (situazione di normalità), il circuito conserva lo stato, infatti se:

- Q vale 0 (stato stabile 0), la porta NOR superiore continua ad avere output 1 e quella inferiore output 0.
- Q vale 1 (stato stabile 1), la porta NOR superiore continua ad avere output 0 e quella inferiore output 1.

Uno stato dove Q e $\overline{Q}$ sono entrambi 0 o entrambi 1 non è coerente ! *Perche ?*

# Latch e Flip-Flop (2)

**SET:** Se nel latch in stato 0, **S** diventa 1, gli input della porta superiore sono 1 e 0 e quindi l'output $\overline{Q}$ = 0; non appena tale output si propaga sull'input della porta inferiore questa viene ad avere entrambi gli input a 0 e quindi l'output Q diventa 1: il latch è passato allo stato 1.

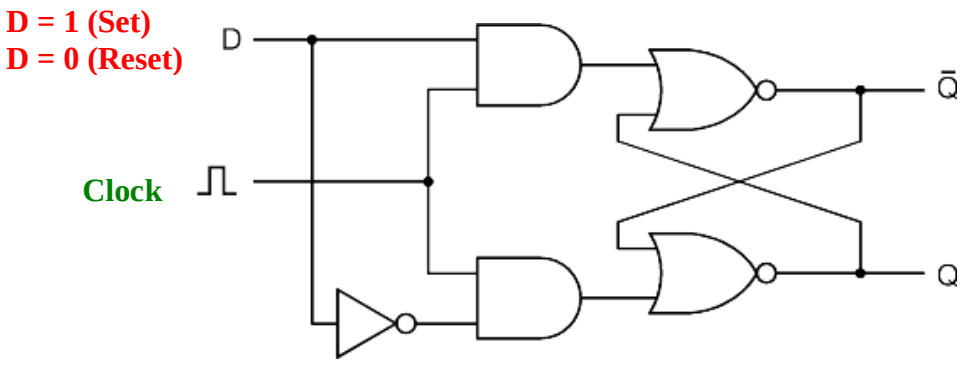
**RESET:** Se nel latch in stato 1, **R** diventa 1, gli input della porta inferiore sono 1 e 0 e quindi l'output Q = 0; non appena tale output si propaga sull'input della porta inferiore questa viene ad avere entrambi gli input a 0 e quindi l'output $\overline{Q}$ diventa 1: il latch è passato allo stato 0.

*L'attivazione di S ed R in stato 1 e in stato 0 rispettivamente non hanno effetto. Perché ?*

In definitiva, un latch SR **ricorda** (fino a che è alimentato elettricamente) qual'è stato l'ultimo valore impostato (0 o 1) e pertanto come vedremo può essere utilizzato come elemento di base per la realizzazione di **memorie RAM**.

*Cosa succede quando S = R = 1 ? Lo stato cambia ?*

Questa situazione il cui output è indeterminato dovrebbe essere evitata. La figura mostra un **Latch di tipo D sincronizzato**, che risolve il problema di avere S ed R contemporaneamente ad 1, e ha un input per il clock che permette di sincronizzare la scrittura con un evento preciso:




**Impedisce che S e R siano contemporaneamente a 1**

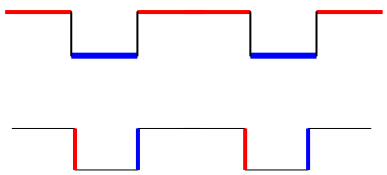
# Latch e Flip-Flop (3)

## Flip-Flop di tipo D

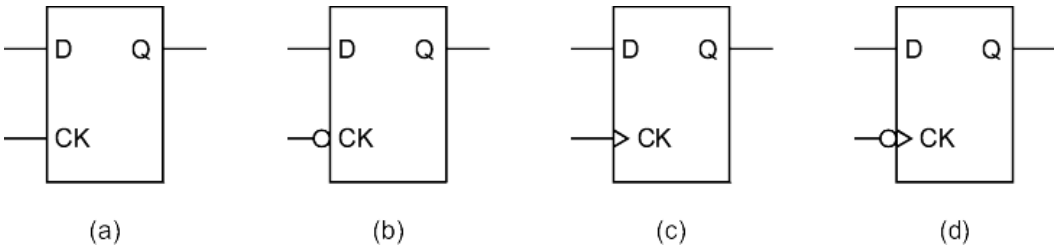
Un **Flip-Flop** è un circuito molto simile a un **Latch**, tanto che spesso i due termini vengono scambiati.

L'unica differenza è relativa all'istante in cui il segnale di clock determina il cambiamento di stato:

- Nel Latch il cambiamento di stato è determinato dal **livello** (alto/basso) del clock.
- nel Flip Flop è determinato dal **fronte** (salita/discesa) del clock.

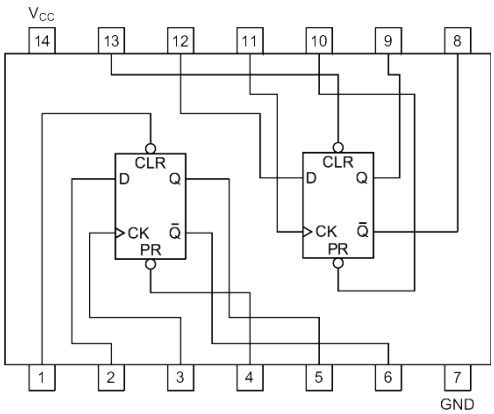


Graficamente, Latch e Flip-Flop sono rappresentati nel modo seguente:




**Latch** attivato con Clock **alto** | **Latch** attivato con Clock **basso** | **Flip-Flop** attivato da fronte **salita** | **Flip-Flop** attivato da fronte **discesa**

Quasi tutti i Latch e Flip-Flop hanno anche l'uscita $\overline{Q}$ e due ingressi supplementari **SET** (o PRESET) e **RESET** (o CLEAR): SET forza l'uscita a 1; RESET forza l'uscita a 0.




Un circuito integrato contenente due Flip-Flop (indipendenti e completi)

# Memorie

Come mostrato in tabella esistono diversi tipi di memoria, di cui le più comuni sono senza dubbio **RAM** e **ROM**.

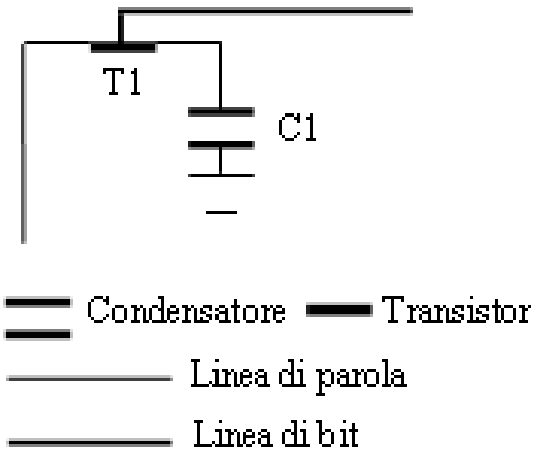
| Tipo | | Categ. | Modalità cancellazione | Cancellazione Singolo byte | Volatile | Uso tipico |
|---|---|---|---|---|---|---|
| RAM | SRAM | R/W | Elettrica | Sì | Sì | Cache livello 2 |
| | DRAM | R/W | Elettrica | Sì | Sì | Memoria principale |
| ROM | | R | No | No | No | Grandi volumi |
| ROM Programmabili | PROM | R | No | No | No | Piccoli volumi |
| | EPROM | R/W | Luce UVA | No | No | Prototipazione |
| | EEPROM | R/W | Elettrica | Sì | No | Prototipazione |
| | FLASH | R/W | Elettrica | No | No | Memory cards |

## RAM

**RAM**: Il termine RAM (**R**andom **A**ccess **M**emory) significa memoria il cui accesso può avvenire in modo casuale (a una cella prescelta) e non obbligatoriamente in modo sequenziale. Tale termine è però piuttosto infelice perché anche gli altri tipi di memoria sopraelencati possono essere acceduti in modo casuale. La RAM può essere statica (SRAM) o dinamica (DRAM): entrambe conservano il valore solo se alimentate (**volatili**).

Le **SRAM** sono realizzate come vedremo con circuiti Flip-Flop tipo D. Tali memorie sono molto veloci (alcuni nanosecondi per la lettura o scrittura) ma anche piuttosto costose, pertanto vengono normalmente utilizzate nei calcolatori odierni solo come Cache di 2 livello.

# Memorie (2)




Le **DRAM** non sono realizzate con Flip-Flop ma con array di celle ognuna costituita da un transistor più un condensatore.
Poiché la carica elettrica immagazzinata dal condensatore tende a disperdersi, è necessario prevedere un circuito di rinfresco delle cariche che (ogni pochi millisecondi) ricarica le celle attive consentendo loro di mantenere lo stato logico.

Le DRAM sono più economiche e dense (circa un 1 transistor contro 6 per ogni bit) ma anche significativamente più lente (decine di nanosecondi per un accesso).

Le DRAM tradizionali sono **asincrone** (ovvero la comunicazione con la CPU non è sincronizzata da un segnale di clock, e sono necessarie linee di handshaking). Sono recentemente stati introdotti varianti più efficienti della DRAM tradizionale, tra cui:

- **DRAM sincrona** (SDRAM): è sincronizzata con la CPU da un segnale di clock e viaggia alla piena velocità del bus (es: 100MHz): la risposta è sempre inviata dopo un numero prefissato di cicli di clock, e quindi la CPU dopo aver inviato la richiesta può dedicarsi ad altro, nell'attesa che la memoria sia pronta. Per l'accesso a sequenze di byte ad indirizzi contigui è possibile utilizzare la modalità burst che consente un transfer rate maggiore.

# Memorie (3)

- **DDR SDRAM**: sono le memorie oggi più utilizzate. Si tratta di una variante di SDRAM (quindi anch'essa sincrona) nota come Double Data Rate SDRAM in quanto può inviare i dati alla CPU due volte ogni ciclo di clock.

  Inoltre nella specifica DDR sono previsti due canali di accesso paralleli 64 bit (128 bit per volta).

  Esistono diverse generazioni con caratteristiche sempre più performanti. Attenzione a distinguere: **larghezza di banda** (bandwidth) da **latenza** (latency).

  Per una DDR che opera con frequenza di bus pari a N MHz, la larghezza di banda si calcola come: N × 2 (trasf/ciclo) × 8 (byte ampiezza bus). Questo valore va ulteriormente raddoppiato in caso di 2 canali di accesso paralleli.

  Le DDR hanno un'elevata banda grazie a un meccanismo interno di pipelining che una volta reperita la prima parola produce 2 parole a ogni ciclo.

  Il tempo necessario per reperire la prima parola è invece noto come latenza e raramente è minore di 10 ns (tipicamente 15-20 ns).

  Caratteristiche delle diverse generazioni di DDR:

  - DDR: fino a 200MHz → banda dual channel = 200 × 2 (trasf/ciclo) × 2 (canali) × 8 (byte ampiezza bus) = 6,4 GB/s. Moduli DIMM tipici da 512 MB.
  - DDR2: fino a 600MHz → banda dual channel 19,2 GB/s. Moduli DIMM tipici da 1 GB.
  - DDR3: fino a 1066MHz → banda dual channel 34,13 GB/s. Moduli DIMM tipici da 2-4 GB.
  - DDR4: fino a 1600MHz → banda dual channel 51,2 GB/s. Moduli DIMM tipici da 8-16 GB.

# Memorie (4)

## ROM

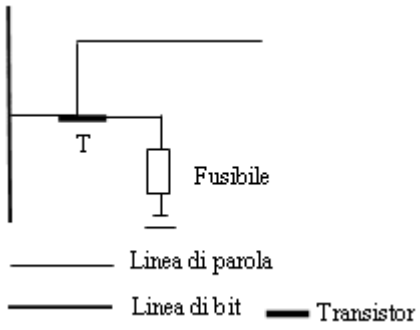
Il termine ROM (**R**ead **O**nly **M**emory) significa memoria di sola lettura. Questo tipo di memoria non può essere cancellata o riscritta; d'altro canto conserva il proprio valore (**non volatile**) anche se non alimentata. Pertanto la ROM viene solitamente utilizzata per la memorizzazione permanente di programmi stabili.

La realizzazione di ROM non può essere fatta nella pratica dalle aziende che progettano circuiti in quanto richiede una particolare tecnica costruttiva (chiamata mascheratura) per la quale è necessario operare direttamente sul chip di silicio per modificarne la struttura.

Pertanto, la produzione di ROM deve essere per forza appaltata a grosse aziende produttrici di chip, e per questo motivo è conveniente solo per grosse quantità di chip tutti uguali.

Inizialmente il BIOS dei PC era memorizzato su una memoria di questo tipo. Successivamente è stato usata memoria EPROM e attualmente FLASH.

## ROM Programmabili




**PROM**: in fase di prototipazione di nuovi circuiti è molto più conveniente utilizzare chip PROM (**P**rogrammable **ROM**) che, possono essere scritti una sola volta, ma la cui scrittura avviene attraverso segnali elettrici generati da apparecchiature di basso costo che operano bruciando dei fusibili.

# Memorie (5)

**EPROM**: una configurazione circuitale opportuna può trasformare i transistor, solitamente impiegati nei circuiti digitali come interruttori comandati, in vere e proprie "trappole di carica" impiegabili per memorizzare un bit proprio come i condensatori delle DRAM. Grazie al miglior isolamento, in assenza di alimentazione questi transistor mantengono il loro stato di carica non per decine di millisecondi come le DRAM, ma per decine di anni.


Una EPROM è un particolare tipo di PROM (**E**rasable **PROM**) che viene programmata analogamente alla PROM, ma il cui contenuto può essere cancellato con luce ultravioletta (esposizione di circa 15 minuti) e nuovamente riscritto utilizzando l'apposito programmatore.

**EEPROM**: Una EEPROM (**E**lectrically **EPROM**) è un particolare tipo di EPROM che può essere cancellata elettricamente in modo molto più veloce di una EPROM. La ri-programmazione può essere fatta "**in-circuit**" ovvero senza smontare il chip dal circuito! D'altro canto le EEPROM sono circa 64 volte piccole (meno capacità di memoria) e circa il doppio più lente delle EPROM. Non possono poi competere in termini di velocità con SRAM o DRAM delle quali sono 10 volte più lente, 100 volte più piccole (come capacità) e molto più costose.

# Memorie (6)

**FLASH**: Una tecnologia di EEPROM piuttosto recente è la FLASH. Questo tipo di memoria viene utilizzata oggigiorno in moltissimi dispostivi "mobili" che necessitano di supporto di memorizzazione non volatile ("chiavette USB", riproduttori audio MP3, fotocamere digitali, palmari, cellulari, ecc.). I tempi di accesso sono buoni (circa 100 ns) e la memoria consente letture di singole parole. Come la EEPROM, può essere riprogrammata in-circuit indirizzando però non una singola parola ma un "blocco". Allo stato attuale l'unico limite tecnologico è il limitato (si fa per dire) tempo di vita: da circa 3000 a 1 Milione di riscritture a seconda dei modelli!

Le flash memory sono vendute sotto forma di chip ma anche sotto forma di memory card.

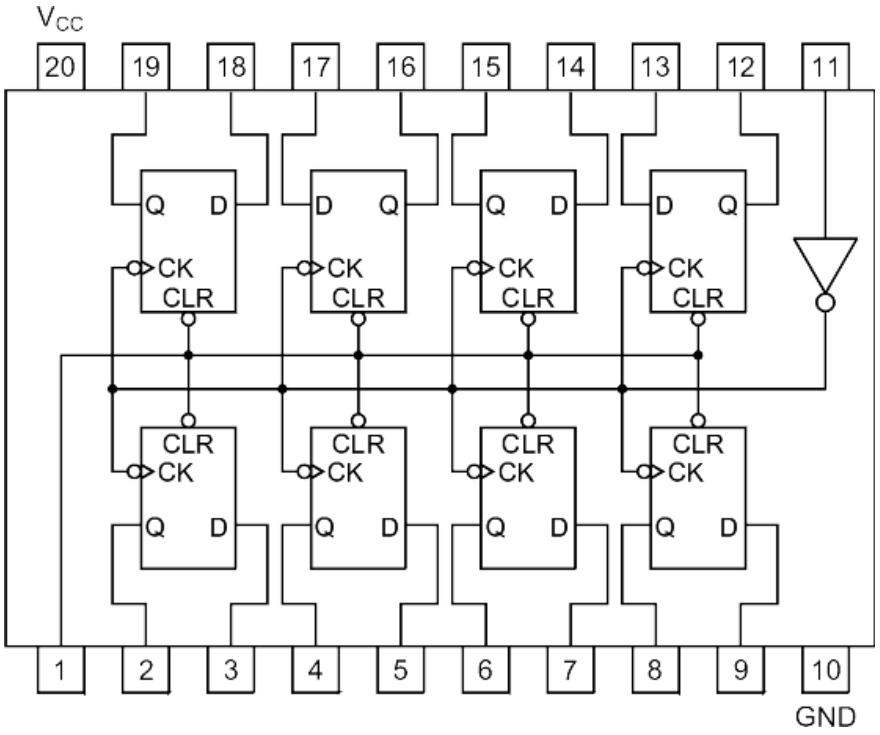
Attualmente esistono varie tipologie di flash memory card:

- CF Compact Flash
- SM SmartMedia
- MMC MultiMediaCard
- MS Memory Stick
- SD Secure Digital
- TF TransFlash
- xD xD-Picture Card
- USB memory Flash
- SSD Solid State Disk

# Memorie (7)

## Realizzazione di un Registro 8-bit

La figura mostra un Circuito Integrato che realizza un registro a 8-bit o in altre parole una semplice RAM 1×8 realizzata con 8 Flip-Flop D. Tale memoria contiene una singola parola e conseguentemente la sua realizzazione è molto più semplice di un generico chip di SRAM.

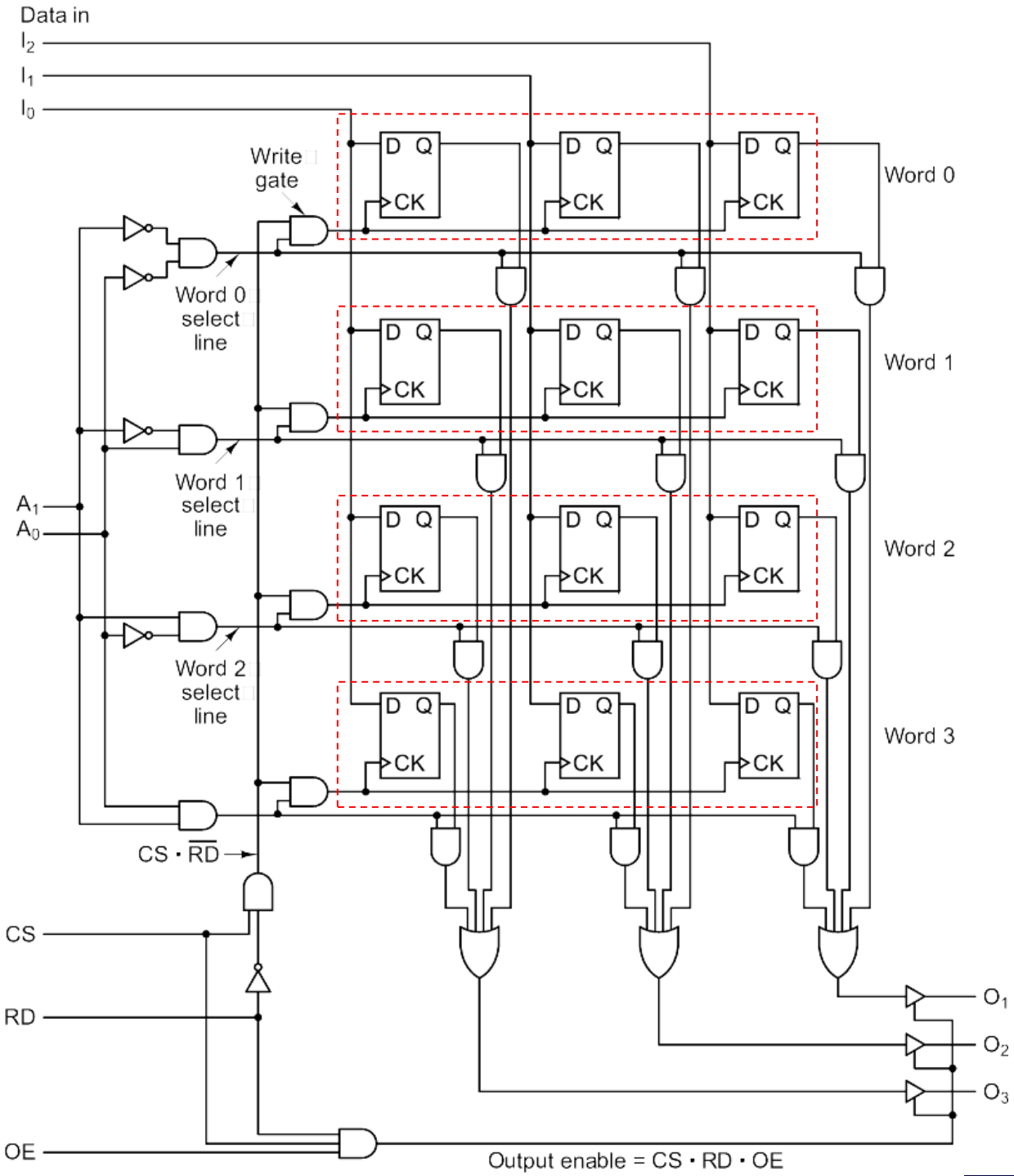



- Essendo contenuta una sola parola, non è necessaria nessuna selezione di indirizzo.
- Gli 8 Flip-Flop hanno gli input Clock collegati tra loro e sono pilotati da una porta NOT collegata al PIN 11. *La memoria da quale fronte (salita/discesa) viene attivata ?*
- Input (D) e Output (Q) dei dati **sono separati**.
- Non sono presenti WR, OE, e CS.
- Per la scrittura è sufficiente rendere i dati stabili sui PIN D, e inviare un impulso su CK.
- Il byte memorizzato è sempre disponibile sui PIN Q.

# Memorie (8)

## Realizzazione di un chip SRAM

La figura mostra lo schema di una SRAM 4×3 realizzata con 12 Flip-Flop D. Sebbene tale memoria abbia una capacità molto limitata, lo schema generale di questo circuito si presta a realizzare memorie anche molto più grandi.

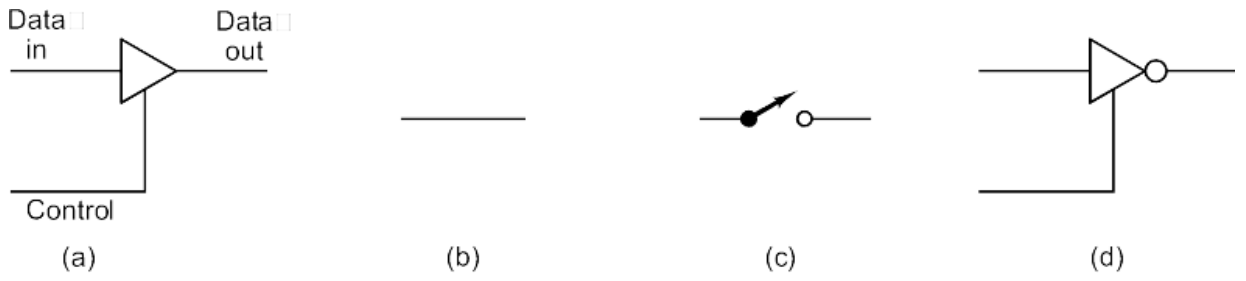


# Memorie (9)

- $I_0$, $I_1$ e $I_2$ sono gli input dati (nei quali devono essere impostate le parole che devono essere scritte nella memoria).
- $O_1$, $O_2$ e $O_3$ sono gli output dati (nei quali verranno posti dalla memoria le parole che devono essere messe in output).
- $A_0$ e $A_1$ sono i due input per la selezione dell'indirizzo (0..3) della parola di interesse. Tali input pilotano un semplice decoder (le 4 porte AND sulla sinistra) che attiva una sola delle parole della memoria per volta.
- L'input CS abilita il Chip per scritture o letture

**READ**: in fase di lettura CS (selezione del Chip), RD (Read) e OE (Output Enable) sono alti. Pertanto l'output della porta $CS \cdot \overline{RD}$ è basso e il CK dei Flip-Flop non viene attivato. Gli output (Q) della parola selezionata "passano" attraverso le porte AND selezionate dal decodificatore di indirizzi per giungere agli OR a 4 input. Come ultimo stadio le uscite degli OR sono collegate ai 3 buffer tri-state (simbolo triangolare) che agiscono come interruttori elettronici capaci di collegare/scollegare elettricamente due parti di un circuito. L'AND a tre input $CS \cdot RD \cdot OE$ abilita i buffer tri-state di uscita rendendo disponibile la parola selezionata su $O_1$, $O_2$ e $O_3$.

**WRITE**: in fase di scrittura CS (selezione del Chip) è alto mentre RD (Read) e OE (Output Enable) sono bassi. I dati devono essere disponibili su $I_0$, $I_1$ e $I_2$. L'output della porta $CS \cdot \overline{RD}$ è ora alto e il CK dei Flip-Flop della parola selezionata viene pilotato permettendo la scrittura nella RAM.




Buffer tri-state | Chiuso | Aperto | Invertente

# Memorie (10)

## Organizzazione Chip di memoria

La memoria è organizzata in Chip. Ogni Chip è caratterizzato da:

- Una **lunghezza di parola**, che specifica la dimensione in bit di ogni unità di informazione indirizzabile (leggibile o scrivibile) singolarmente. Le tipiche lunghezze di parola sono: 1, 4, 8 e 16 bit.
- Un **numero di parole**, che indica quante parole sono contenute all'interno del Chip.

La dimensione (capacità) in bit di un Chip di memoria si ottiene dal prodotto: **numero parole × lunghezza parola**

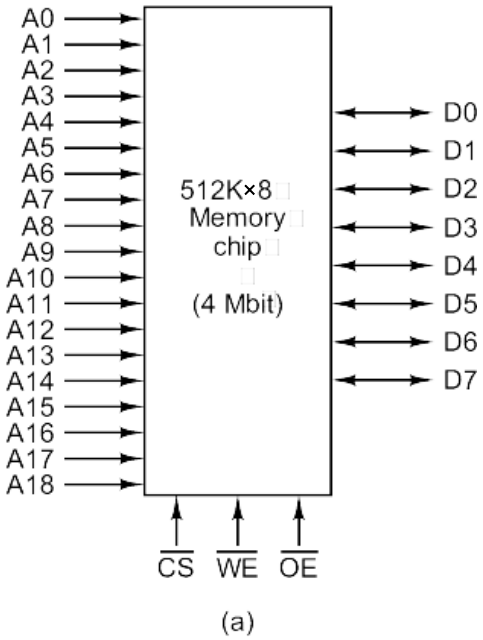
Man mano che la tecnologia migliora la capacità di un Chip di memoria aumenta, solitamente di due volte ogni 18 mesi (Legge di Moore). I Chip di SDRAM utilizzati oggigiorno nei calcolatori hanno capacità tipiche di alcuni Gbit (Gigabit).

Un Chip di memoria, indipendentemente dall'organizzazione interna (*che nel caso di SRAM vedremo nel seguito*), è dotato di una serie di piedini di I/O necessarie per il collegamento alla CPU e al BUS:

- Se la lunghezza della parola è **N**, saranno presenti **N** PIN attraverso i quali i dati vengono letti e scritti.
- Per selezionare quale parola leggere o scrivere saranno necessari una serie di PIN di input che specificano l'indirizzo. *Quanti ?*

# Memorie (11)

La figura mostra una possibile organizzazione di un Chip di RAM da 4 Mbit:



(a)

Nell'esempio di figura, per indirizzare una delle **512K** parole di **8-bit** sono necessari:

$\log_2(512x1024) = 19$ bit

La memoria RAM di un calcolatore è generalmente costituita da diversi Chip. A seconda dell'indirizzo di memoria richiesto dalla CPU o da una periferica sarà necessario abilitare/o meno un certo Chip di memoria: l'input **CS** (Chip Select) serve proprio a questo; quando CS (attivo basso) viene attivato il Chip è selezionato.

L'input **WE** (Write Enable, anch'esso attivo basso) serve ad indicare alla memoria RAM se intendiamo leggere o scrivere il byte indirizzato.

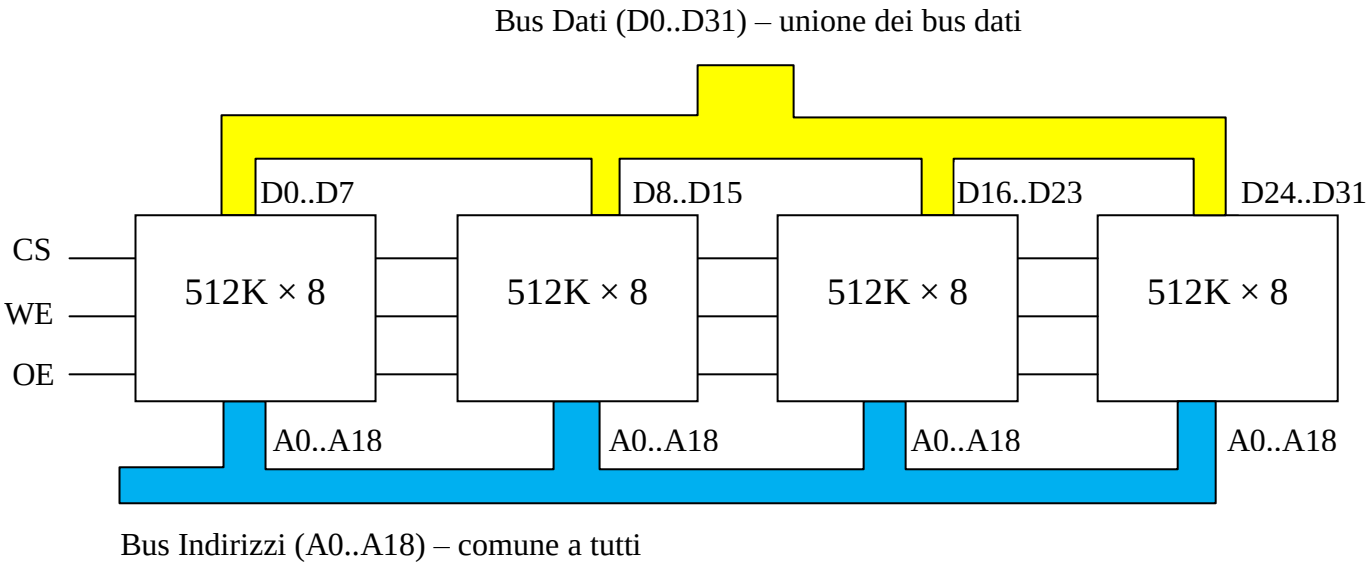
L'input **OE** (Output Enable, anch'esso attivo basso) serve invece per controllare i segnali di Output; quando questo non è attivato l'output del Chip è "staccato" dal circuito.

# Memorie (12)

*Come realizzare una memoria 512K × 32 utilizzando Chip 512K × 8 ?*

È sufficiente utilizzare 4 Chip identici. Ciascuna parola di memoria è suddivisa in 4 parti e ogni parte memorizzata da un diverso chip (allo stesso indirizzo). Pertanto:

- le 19 linee indirizzi sono comuni a tutti i Chip;
- stesso dicasi per i 3 input CS, WE e OE; infatti tutti i chip lavorano sempre insieme;
- relativamente al bus dati, le 4 parole (8-bit ciascuna) dei singoli chip vengono affiancate a costituire la parola a 32 bit desiderata.




Questo rappresenta il normale schema di collegamento di singoli Chip di Memoria per la **realizzazione di un modulo DIMM**.

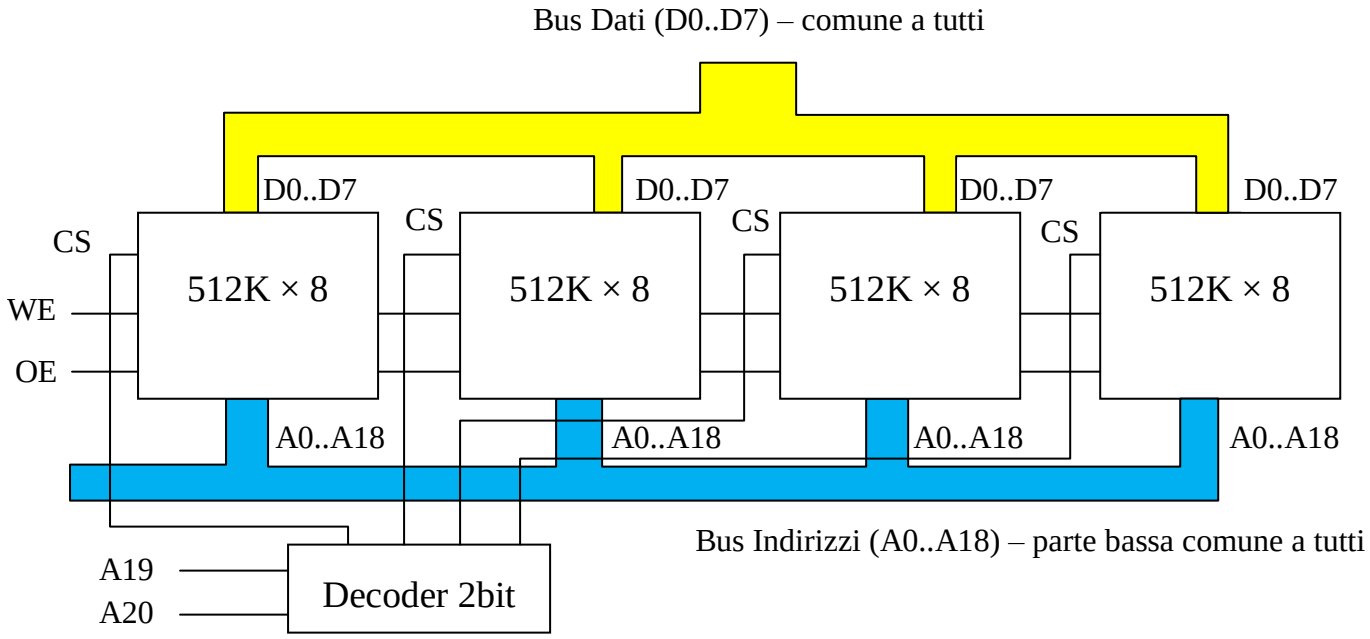
*Ad esempio per realizzare un modulo DIMM di capacità 1 GB (organizzato 128M×64) possiamo usare 8 Chip con capacità 1024Mbit (organizzazione 128M×8).*

# Memorie (13)

*Come realizzare una memoria 2M × 8 utilizzando Chip 512K × 8 ?*

Anche in questo caso possiamo utilizzare 4 Chip identici, ma questa volta i 4 chip contengono parole complete (8 bit) di indirizzi diversi. Dobbiamo fare in modo che, a seconda dell'indirizzo richiesto, si attivi il chip che contiene la parola corrispondente.

- i bus dati possono essere collegati insieme (uno solo chip per volta è attivo e quindi non ci sono conflitti);
- stesso dicasi per i 2 input WE e OE;
- per poter indirizzare 2M parole occorre estendere il bus indirizzi da 19 a 21 linee. Le linee A0..A18 sono ancora collegate in parallelo a tutti i chip; i due bit più significativi degli indirizzi (A19 e A20) sono invece utilizzati per selezionare (attraverso un decoder) quale chip attivare (collegamento a CS).



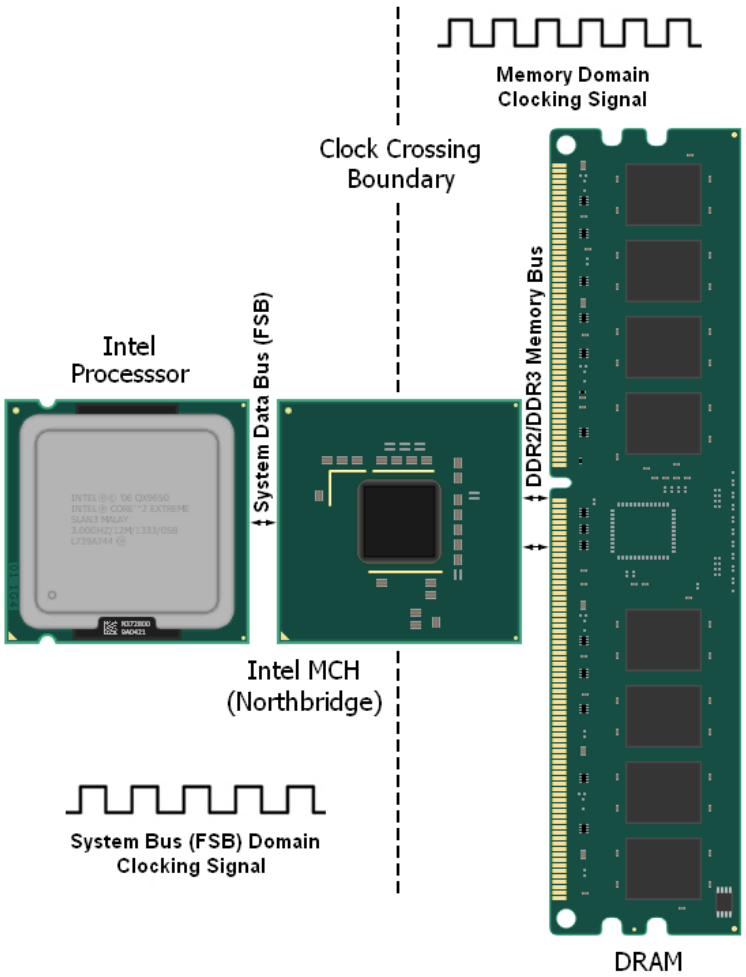

Questo tipo di schema (o meglio un caso ibrido tra questo e il procedente) è utilizzato per organizzazione **DIMM** a banchi (rank), con la quale aumentare la capacità del modulo usando chip di pezzature "disponibili".

# La memoria cache (1)

Come più volte detto la memoria principale costituita da RAM dinamica (DRAM) è molto più lenta della CPU, e pertanto la necessità di leggere opcode e operandi dalla memoria causa un rallentamento rispetto alle prestazioni teoriche della CPU.

Anche se la banda di trasferimento con memorie DDR è molto elevata (grazie anche al pipelining DDR), il problema rimane la latenza iniziale (tempo di attesa trasferimento prima parola) raramente inferiore a 10 ns.

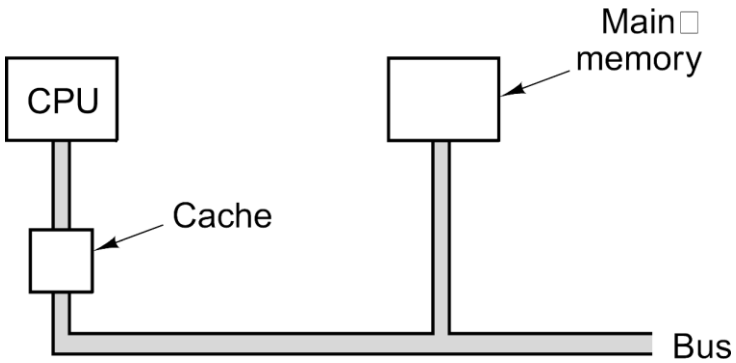
Nel calcolo della latenza globale non dobbiamo conteggiare il solo tempo di recupero del dato da parte della memoria ma tenere conto anche del controller della memoria (quando questo è integrato nel North Bridge):




Programmable Clock Crossing Scheme with
Dynamic Read Delay Compensation

# La memoria cache (2)

Uno dei modi per far fronte a questo problema potrebbe essere quello di utilizzare SRAM invece di DRAM: le SRAM sono molto più veloci (pochi ns), ma ahimè, anche molto più costose. Proprio per motivi di costo, non potendo realizzare con SRAM tutta la memoria principale, viene introdotto il meccanismo della **cache**, ovvero di una memoria veloce, di dimensione limitata rispetto all'intera RAM, da "interporre" tra CPU e RAM.




Il **principio di funzionamento** è semplice. La CPU reperisce sempre i dati dalla memoria cache, come se questa potesse contenere tutta l'informazione memorizzabile in RAM:

- qualora la parola desiderata sia effettivamente presente in cache (**cache hit**) otteniamo un indiscutibile vantaggio nel tempo di accesso.
- d'altro canto se la parola non è presente (**cache miss**), è necessario trasferirla da DRAM a cache e poi leggerla; in questo caso il tempo totale è sostanzialmente maggiore rispetto alla lettura da DRAM.

Pertanto l'utilizzo di cache è **vantaggioso** solo quando la percentuale di hit è sufficientemente alta.

# La memoria cache (3)

Il **caricamento di un dato** in un sistema dotato di cache è il seguente:

1) Richiedi il dato alla cache
2) Se il dato è presente in cache caricalo subito
3) Altrimenti:
    a. Leggi il dato dalla memoria
    b. Carica il nuovo dato in cache; se la cache è piena, applica la politica di rimpiazzamento per caricare il nuovo dato al posto di uno di quelli esistenti.

Lo spostamento di dati tra memoria principale a cache non avviene per parole singole ma per blocchi denominati **linee di cache**. La dimensione di una linea di cache varia da 8..512 parole: 64 byte (byte non bit!) per CPU x86 Intel.

Le **tecniche di gestione** della cache, ovvero le politiche di allocazione e rimpiazzamento di porzioni di cache sono basate sul principio di località spaziale e temporale:

- **località spaziale** significa che vi è alta probabilità di accedere, entro un breve intervallo di tempo, a celle di memoria con indirizzo vicino. Le politiche di allocazione di cache tengono conto di ciò leggendo normalmente più dati di quelli necessari (un intero blocco denominato linea di cache), con la speranza che questi vengano in seguito richiesti.

- **località temporale** significa invece che vi è alta probabilità di accedere nuovamente, entro un breve intervallo di tempo, ad una cella di memoria alla quale si è appena avuto accesso. Le politiche di rimpiazzamento sfruttano questa proprietà per decidere quale blocco di cache rimpiazzare (cioè quale blocco debba essere sovrascritto dal nuovo blocco entrante). Normalmente la politica utilizzata è **LRU** (viene rimpiazzato il blocco utilizzato meno recentemente **L**east **R**ecently **U**sed).

# La memoria cache (4)

Nei calcolatori moderni vengono spesso utilizzate **più cache**:

- Diversi **livelli di cache** (L1, L2, ...) sono impiegati in cascata per ottimizzare il trade-off costi/prestazioni: di solito 2 livelli di cache.

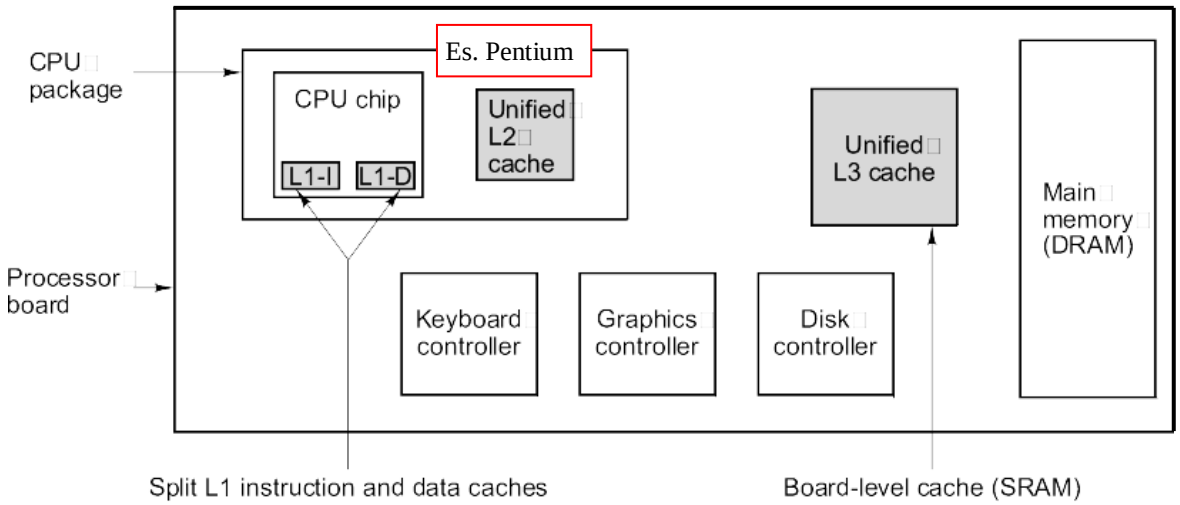
  La cache di livello 1 (**L1**) è più veloce di quella di livello 2 (**L2**) e così via. D'altro canto il maggior costo della memoria per la realizzazione di cache L1 fa sì che la sua dimensione sia inferiore di L2 e così via.

  Normalmente la cache di livello i+1-esimo mantiene l'intero contenuto di quella di livello i-esimo.

- **Cache diverse** sono utilizzate per **istruzioni** e **dati**. Infatti, per i due gruppi di informazioni le località sono spesso diverse; *perché ?*

Nella figura è mostrata una microarchitettura a **3 livelli di cache**:

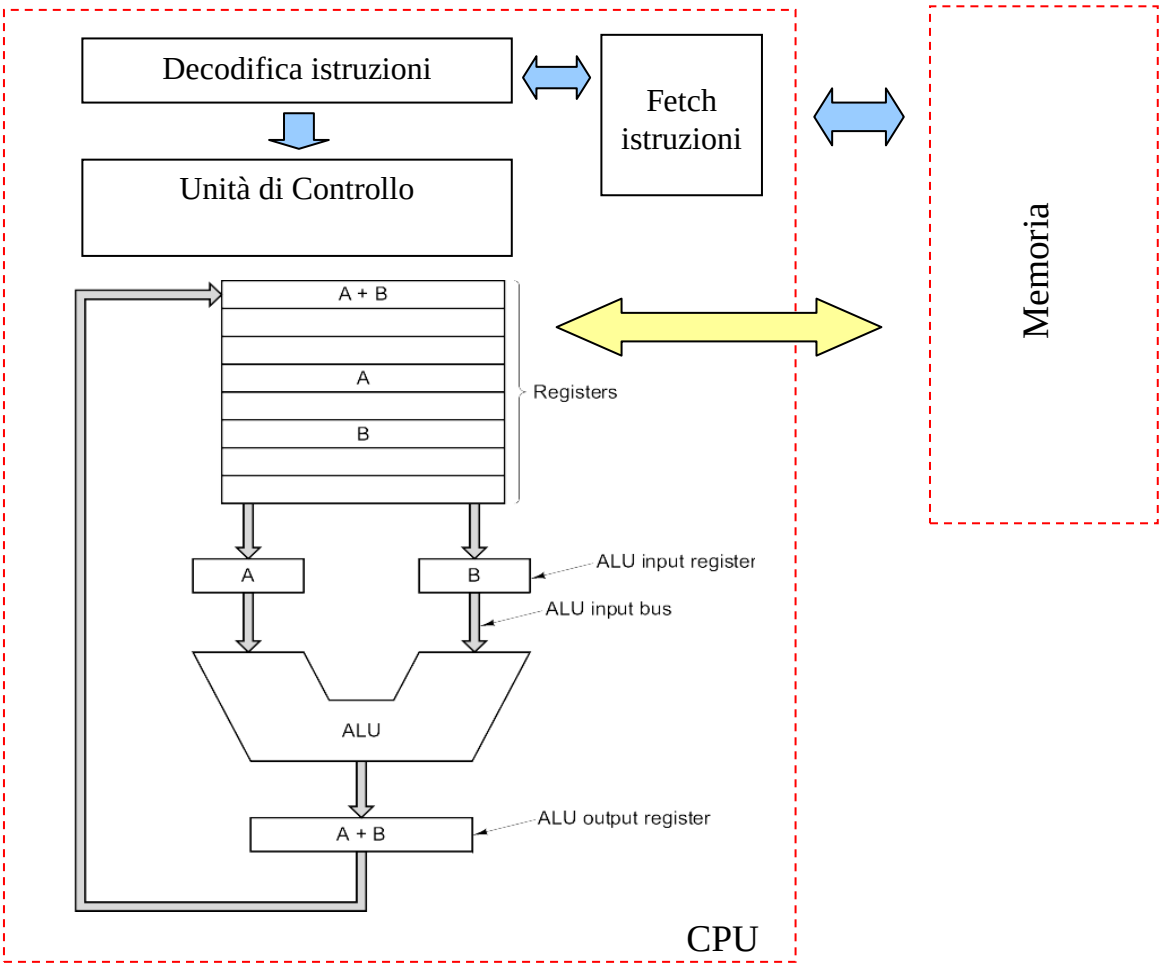
- la CPU contiene al suo interno una piccola cache (L1-I) per le istruzioni (16KB) e una piccola cache (L1-D) per i dati (64 KB);
- la cache di livello 2 (da 512 KB a 1 MB) non è interna alla CPU ma assemblata nel package della CPU.
- una cache di livello 3 (alcuni megabyte) è presente infine sulla motherboard.




Il montaggio di cache direttamente all'**interno** della CPU o del package della CPU consente di realizzare Bus interni (molto veloci) riservati esclusivamente alla comunicazione CPU-cache e quindi non contesi da altri master.

# Il Chip della CPU (1)

Un semplice schema a blocchi che mostra il **funzionamento di massima**:

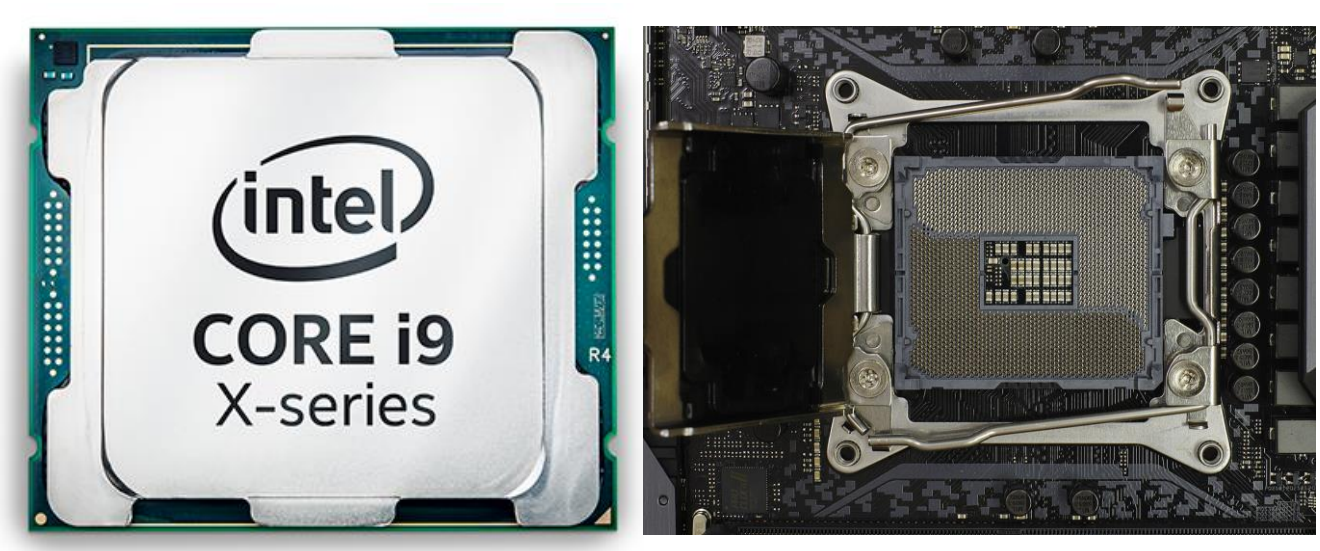



- La CPU legge le istruzioni assembler (ISA) del programma da eseguire dalla memoria (**Fetch**).
- Le istruzioni sono internamente **decodificate** e passate all'**unità di controllo** che può essere:
  - **microprogrammata** (CPU CISC): ad ogni istruzione ISA corrisponde un microprogramma che indica la sequenza di microistruzioni da eseguire.
  - **cablata** (CPU RISC): l'esecuzione avviene attraverso un circuito digitale sequenziale che implementa una macchina a stati.
- La CPU contiene una **ALU** e alcuni **registri** (memoria di lavoro) che le permettono di eseguire diversi compiti elementari.

# Il Chip della CPU (2)

Quasi tutte le CPU moderne sono realizzate all'interno di un unico Chip VLSI. Questo Chip è dotato di un insieme di piedini (PIN) che servono al collegamento e all'interazione della CPU con il mondo esterno. Socket di CPU recenti hanno un numero molto enorme di contatti (2066 contatti per il socket LGA 2066 sotto)!

La CPU Core i9-7900X e il relativo socket LGA 2066.



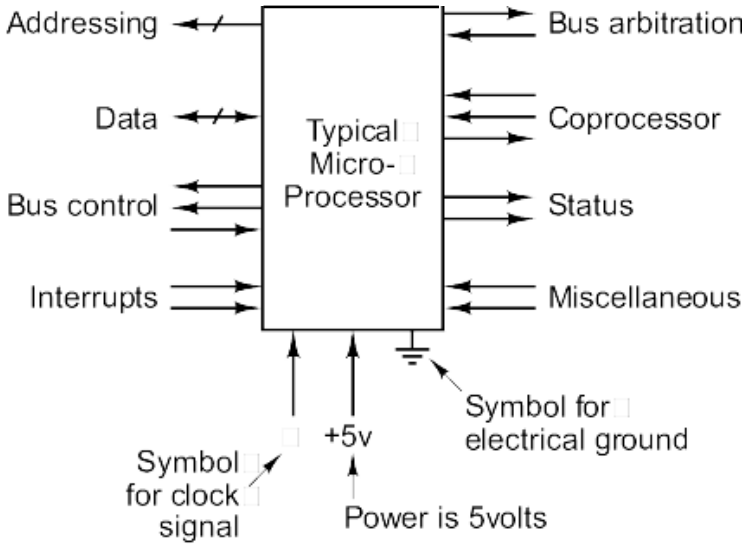

I **PIN** di un Chip CPU sono essenzialmente di tre tipi: **Indirizzo**, **Dati** e **Controllo**.

- I PIN **indirizzo** servono a specificare l'indirizzo di memoria (o di I/O) che la CPU vuole leggere o scrivere. Le dimensioni più comuni per il bus indirizzi sono: 16, 20, 32 e 64. *Quali sono i corrispondenti spazi di indirizzamento ?*

- I PIN **dati** servono a leggere o scrivere parole dalla/sulla memoria o sui dispositivi di I/O. Le dimensioni più comuni per il bus dati sono: 8, 16, 32 e 64. Una CPU con un bus dati ampio è in grado di leggere/scrivere in una sola volta una maggior quantità di informazione dalla memoria e quindi è più efficiente.

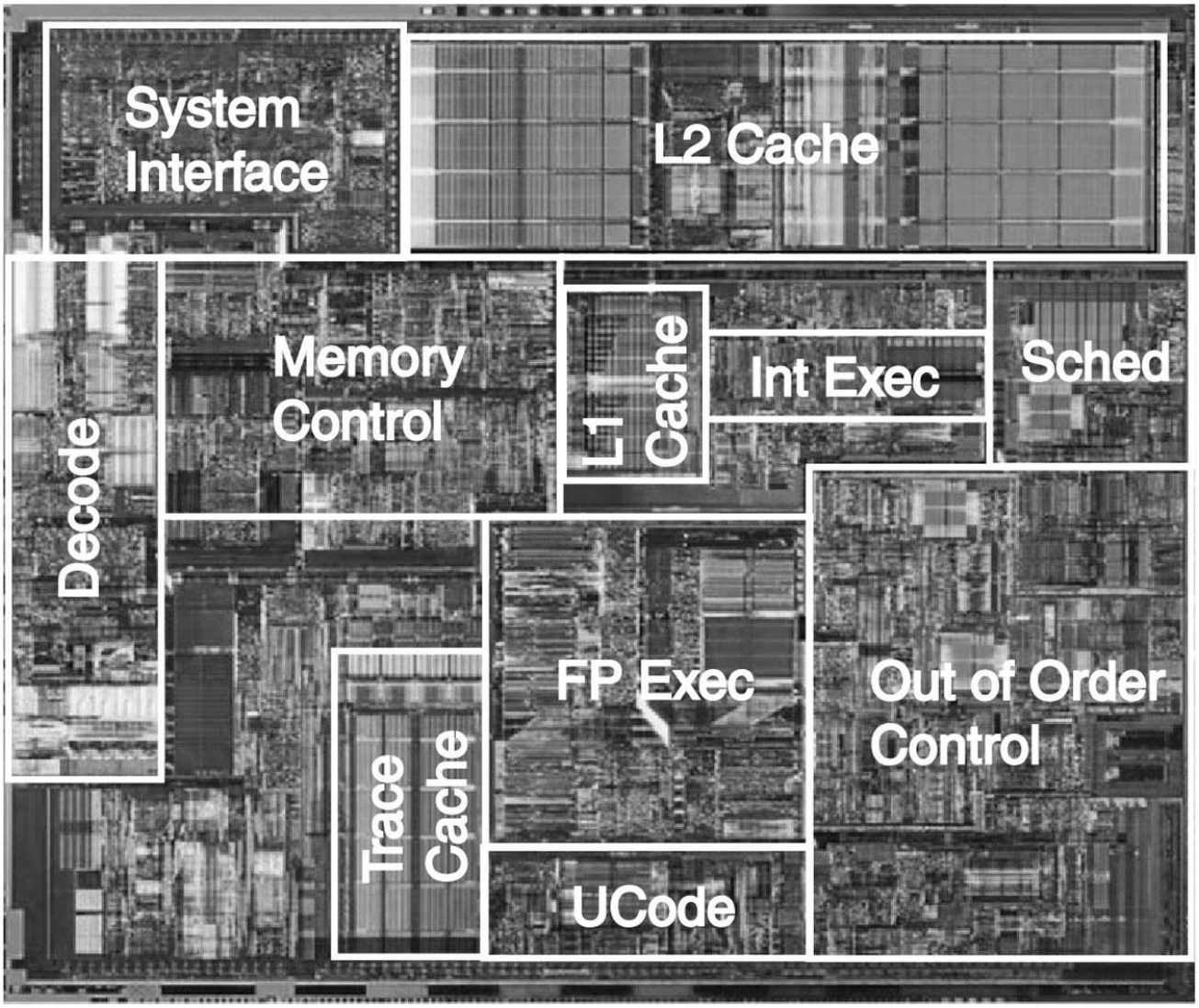
# Il Chip della CPU (3)

- I PIN di **controllo** hanno vari compiti:
  - controllo del BUS
  - Interrupt
  - Arbitraggio del BUS
  - Segnali del coprocessore
  - Stato
  - Varie



# Il Chip della CPU (4)

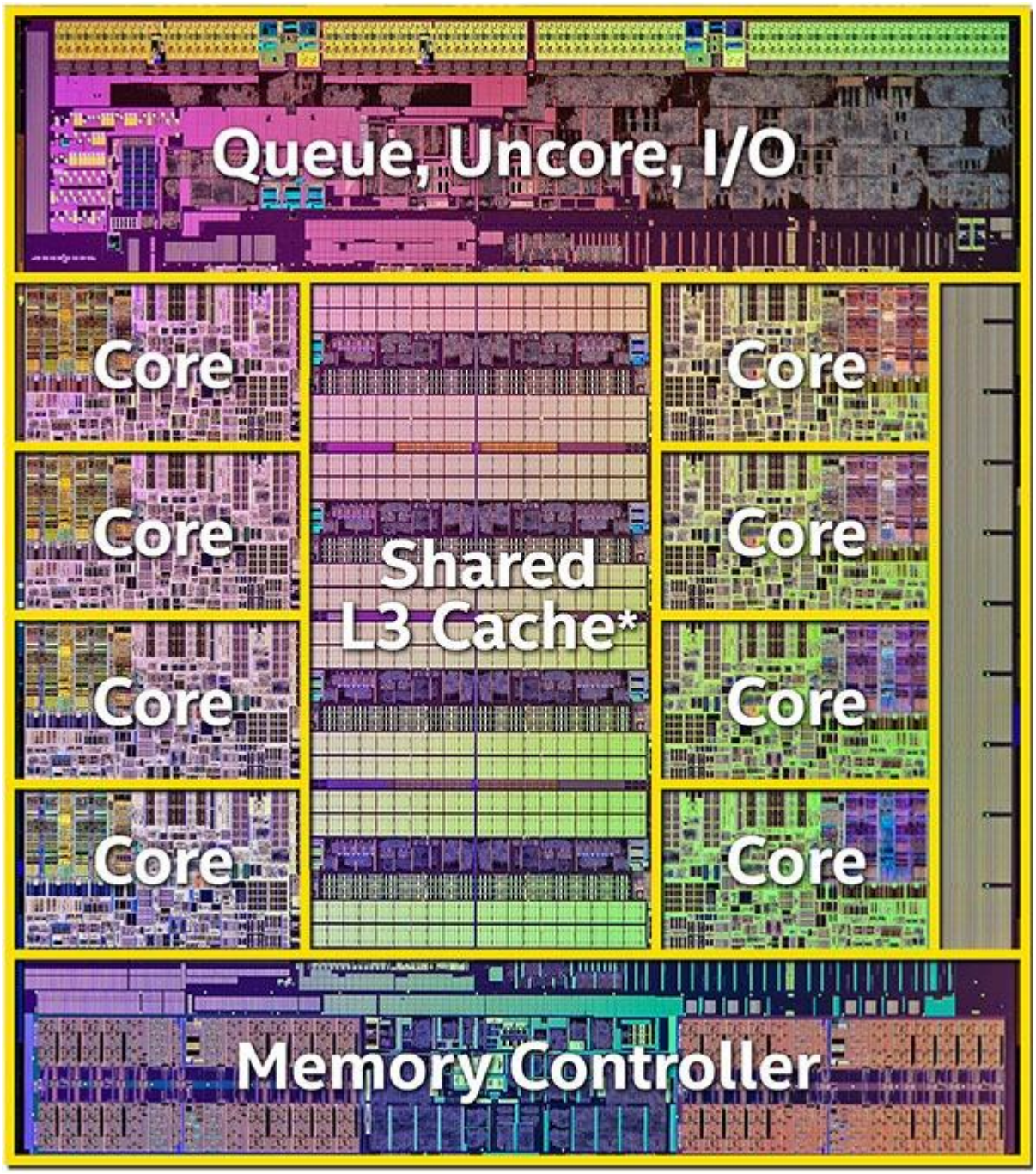
Il chip del **Pentium IV**:



# Il Chip della CPU (5)

Chip **Core i7-5960X** (8 core) – 2014:

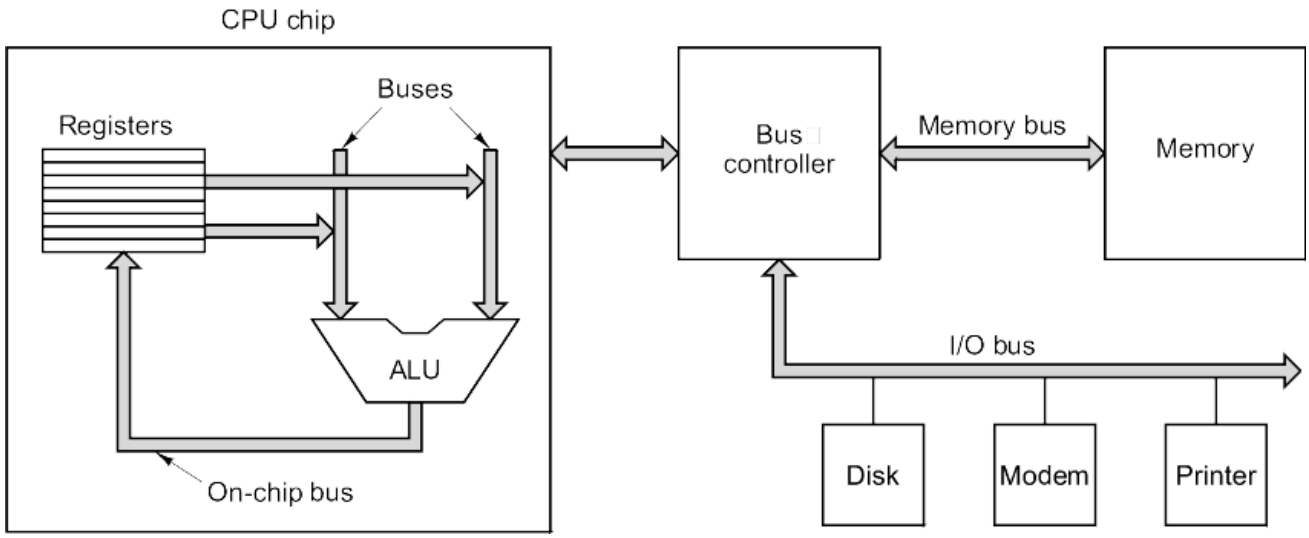
- 2.6 Miliardi di Transistor
- 20 MB cache di terzo livello shared tra gli 8 core



# BUS dei Calcolatori

Un **BUS** è un collegamento elettrico (comunemente multi-linea) comune fra più dispositivi. I BUS possono appartenere a diverse categorie: possono essere interni alla CPU per trasportare i dati dai registri alla ALU, o all'esterno della CPU per collegare quest'ultima alla memoria e ai dispositivi di I/O.

Nella figura è mostrata un'architettura di un calcolatore dove sono evidenziati i BUS interni alla CPU, un BUS per il collegamento della CPU alla memoria e un BUS per il dialogo con le periferiche. Un **BUS controller** si rende necessario per interconnettere tra loro i 2 diversi bus "esterni".




Se da un lato i progettisti di CPU sono liberi di utilizzare qualsiasi tipo di BUS all'interno del Chip, i BUS esterni, e in particolar modo quelli che prevedono il collegamento di periferiche e schede di terze parti, devono attenersi a uno schema di funzionamento ben preciso denominato **protocollo del bus**. Devono essere inoltre perfettamente specificate caratteristiche meccaniche ed elettriche per l'interfacciamento.

Tra i BUS oggigiorno più noti e diffusi: **PCI-Express** (PC), Bus **PCI** (bus espansione PC), **Bus SCSI** (per collegamento a periferiche), Universal Serial Bus **USB** (periferiche), **FireWire** (elettronica consumer), …

# BUS dei Calcolatori (2)

I dispositivi che possono iniziare un trasferimento si dicono **attivi** o **master** (padroni), mentre quelli che possono solo rimanere in attesa di richieste sono detti **passivi** o **slave** (schiavi).

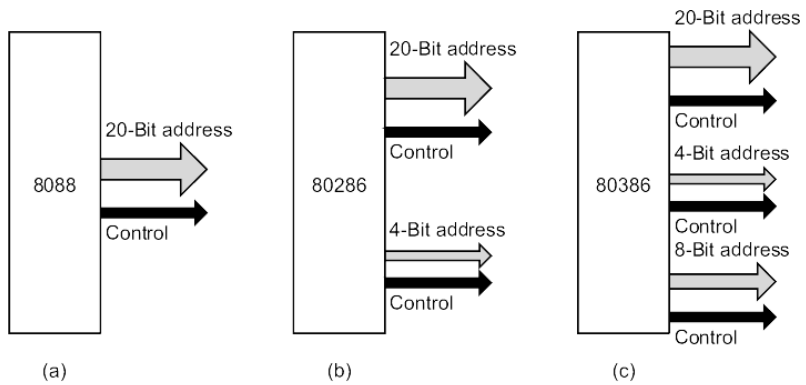
*Quando la CPU ordina al controllore del dischetto di leggere un blocco, essa si comporta come master e il controllore del dischetto come slave; successivamente il controllore del dischetto può comportarsi come master e ordinare alla memoria di immagazzinare le parole da lui lette.*

La tabella riporta alcuni esempi di operazioni svolte da master e slave:

| Master | Slave | Esempio |
|---|---|---|
| CPU | Memoria | Fetch (caricamento) di istruzioni e dati |
| CPU | Dispositivi di I/O | Inizio di un'operazione di I/O |
| CPU | Coprocessore | La CPU da istruzioni al coprocessore |
| I/O | Memoria | DMA (Direct Memory Access) |
| Coprocessore | CPU | Il Coprocessore legge operandi dalla CPU |

## Larghezza del Bus

La larghezza è uno dei parametri più importanti per la progettazione di un BUS. Sembrerebbe ovvio progettare BUS molto ampi per aumentare lo spazio d'indirizzamento e le prestazioni. D'altro canto questi richiedono molti collegamenti, richiedono più spazio sulla scheda madre e connettori più grandi. *Molto spesso i progettisti di CPU hanno dimostrato di non essere abbastanza lungimiranti (vedi caso Intel)*:

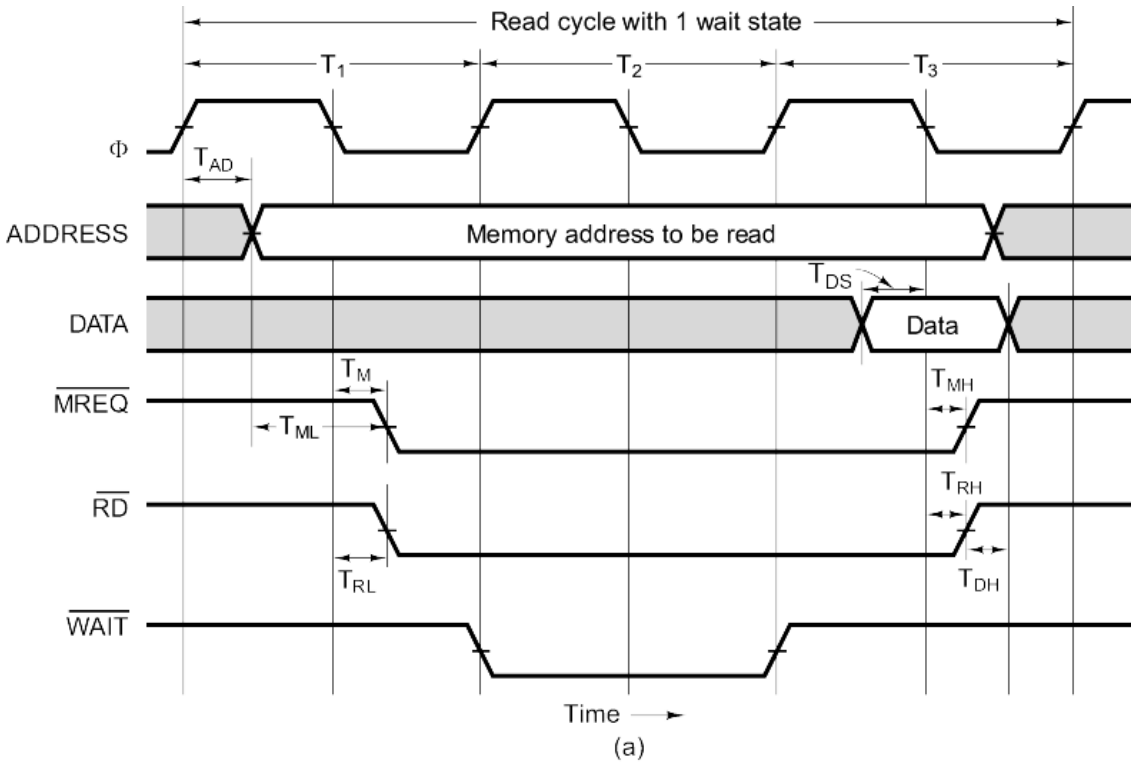


# BUS dei Calcolatori (3)

I BUS possono essere sincroni e asincroni:

- Un **BUS sincrono** ha una delle linee pilotata da un segnale di Clock che stabilisce la cadenza di tutte le operazioni.
- Un **BUS asincrono** non è dotato di un clock principale ma sono le parti che comunicano su di esso a doversi esplicitamente sincronizzare.

## Bus sincrono




(a)

| Symbol | Parameter | Min | Max | Unit |
|---|---|---|---|---|
| $T_{AD}$ | Address output delay | | 11 | nsec |
| $T_{ML}$ | Address stable prior to $\overline{MREQ}$ | 6 | | nsec |
| $T_M$ | $\overline{MREQ}$ delay from falling edge of Φ in $T_1$ | | 8 | nsec |
| $T_{RL}$ | $\overline{RD}$ delay from falling edge of Φ in $T_1$ | | 8 | nsec |
| $T_{DS}$ | Data setup time prior to falling edge of Φ | 5 | | nsec |
| $T_{MH}$ | $\overline{MREQ}$ delay from falling edge of Φ in $T_3$ | | 8 | nsec |
| $T_{RH}$ | $\overline{RD}$ delay from falling edge of Φ in $T_3$ | | 8 | nsec |
| $T_{DH}$ | Data hold time from negation of $\overline{RD}$ | 0 | | nsec |

(b)

# BUS dei Calcolatori (4)

Nell'esempio della figura precedente, assumiamo che il Clock operi a 40 MHz (tempo di ciclo di 25 ns). Assumiamo inoltre che la lettura dalla memoria richieda 40ns dal momento in cui l'indirizzo è stabile sul BUS.

- Per eseguire una lettura la CPU mette l'indirizzo sul BUS sul fronte di salita di $T_1$.
- MREQ ed RD vengono attivati (attivi bassi) dalla CPU: il primo indica che si sta accedendo alla memoria (e non a un dispositivo di I/O), il secondo che si vuole leggere e non scrivere.
- Poiché la memoria richiede 40 ns dal momento in cui l'indirizzo è stabile, essa non è in grado di fornire la risposta entro $T_2$; per avvertire la CPU di non aspettare dati attiva il segnale WAIT (attivo basso) che fa sì che vengano aggiunti 1 o più stati di attesa.
- Quando la memoria è pronta a fornire la parola disabilita WAIT (nell'esempio WAIT è stato mantenuto attivo solo durante $T_2$); durante la prima metà di $T_3$ la memoria mette i dati sul BUS.
- Sul fronte di salita di $T_3$ la CPU legge i dati sul BUS e disattiva MREQ e RD.

Tutti gli eventi sono **sincronizzati con i fronti** di salita o discesa del Clock.

Tutti i tempi riportati in tabella costituiscono dei **vincoli** (specificati come tempo minimo o massimo) che devono essere rigidamente rispettati al fine di consentire un dialogo corretto tra le due parti.

I BUS operano solitamente a **frequenze più ridotte** rispetto alle frequenze di funzionamento delle CPU; infatti l'aumento della frequenza di funzionamento sul BUS porta con sé diversi problemi tecnologici di progetto (es: bus skew).

Mentre le CPU recenti operano a circa 3 GHz i corrispondenti BUS veloci (CPU-Memoria) normalmente non superano 1 GHz.

# BUS dei Calcolatori (5)

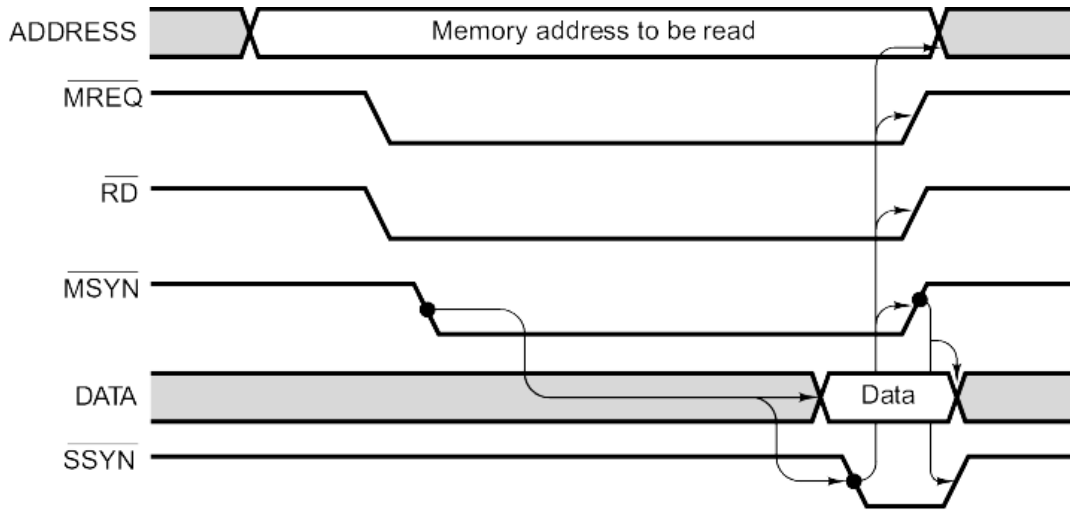
Il limite principale dei BUS sincroni è dovuto al fatto che tutto deve avvenire in tempi **multipli** del tempo di ciclo.

*Se CPU e memoria fossero in grado di concludere un trasferimento in 3.1 cicli, la durata effettiva dovrebbe comunque essere di 4 cicli (0.9 cicli persi per ogni accesso alla memoria).*

Se nell'esempio precedente utilizzassimo memorie con un tempo di accesso di 20ns potremmo eliminare lo stato di WAIT (riduzione di un ciclo); d'altro canto l'utilizzo di memorie da 10ns non porterebbe nessun ulteriore velocizzazione.

Se un BUS sincrono deve poter operare **con dispositivi veloci e dispositivi lenti**, esso dovrà per forza adattarsi a quelli più lenti e di conseguenza i dispositivi veloci non potranno essere sfruttati appieno.

## Bus asincrono




- Non appena il master del BUS ha reso disponibile l'indirizzo e attivato MREQ e RD, attiva MSYN (**Master Syncronization**).
- Quando lo slave vede MSYN esegue il lavoro nel minor tempo possibile, e non appena è pronto attiva SSYN (**Slave Syncronization**).
- Quando il Master vede SSYN attivo, legge i dati (già disponibili sul BUS) e disattiva MREQ, RD e MSYN; lo Slave a sua volta, conscio che il ciclo è terminato, disattiva SSYN.

# BUS dei Calcolatori (6)

La sequenza di eventi interlacciati che caratterizza le transazioni sul BUS asincrono prende il nome di **handshake** (stretta di mano).

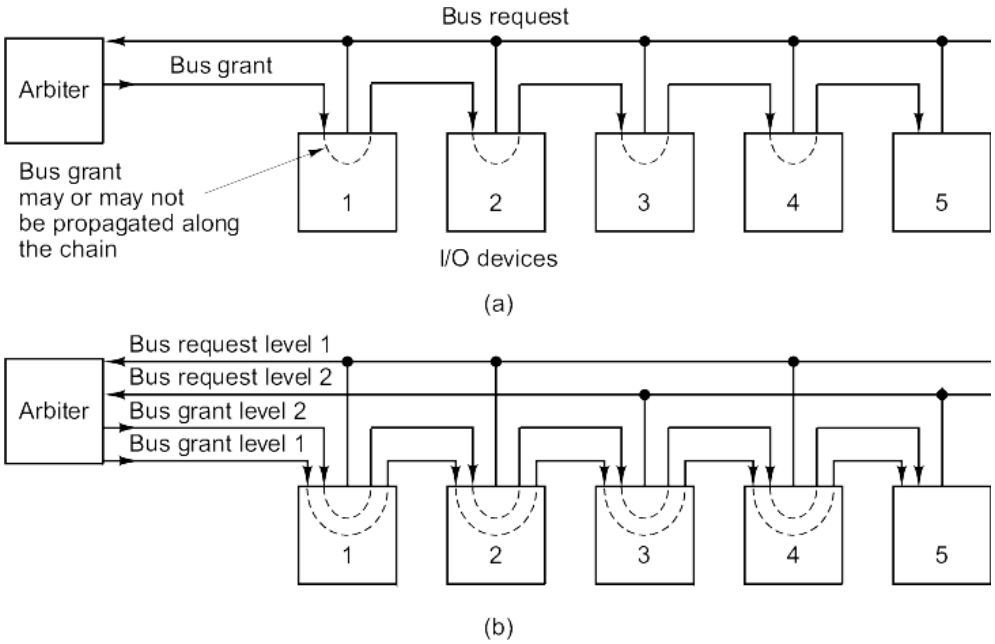
Sembrerebbe che l'utilizzo di BUS asincroni fosse sempre da preferire; nella pratica la maggior parte dei BUS commerciali sono sincroni, è ciò è dovuto essenzialmente alla minor complessità di progettazione (lato CPU).

D'altro canto, la costante crescita delle frequenze di funzionamento dei BUS, e l'utilizzo di BUS separati per periferiche lente, rende di secondaria importanza i problemi evidenziati per i dei BUS sincroni. Oggi la maggior parte dei BUS asincroni sono BUS "esterni" (tipo USB).

## Arbitraggio del Bus

Cosa succede se sullo stesso BUS due dispositivi vogliono prendere contemporaneamente il controllo, ovvero diventare Master del BUS ?
Ovviamente questa è una situazione elettricamente non possibile e quindi la soluzione consiste nel **sequenzializzare** le richieste: uno dei due deve attendere. Per arbitraggio del BUS si intende una politica di gestione del BUS che anche a seguito di richieste multiple assegna il BUS a un dispositivo per volta. L'arbitraggio può essere **centralizzato** o **distribuito**.

Il metodo di arbitraggio centralizzato più noto è detto **daisy chaining**:



# BUS dei Calcolatori (7)

Nel **daisy chaining**:

- Una linea di Grant (concessione), in output dal chip arbitro, è collegata in serie a tutti i dispositivi che possono richiedere di diventare Master.
- Una linea di richiesta, comune a tutte i dispositivi, è collegata in input al chip arbitro.
- Quando un dispositivo necessita di diventare Master, manda un segnale sulla linea di richiesta. Questo può avvenire anche per più dispositivi per volta. L'arbitro a seguito della richiesta (in condizioni normali) attiva la linea di Grant pur senza sapere da chi è arrivata la richiesta.
- Il dispositivo più vicino all'arbitro (che ha la precedenza su tutti) controlla per sapere se è stato lui il richiedente. In caso affermativo, non propaga al successivo il segnale di Grant. In caso negativo invece il segnale viene propagato e il secondo dispositivo, come il primo controlla …

La **distanza dall'arbitro** stabilisce dunque le priorità dei dispositivi sul BUS. Normalmente le periferiche hanno priorità maggiore della CPU, in modo che non vadano persi dati pronti per essere letti o scritti.

Come mostrato in figura (b) del lucido precedente il **daisy chaining** può essere implementato **a più livelli** aggiungendo coppie di linee richiesta/grant. In questo i livelli più bassi hanno precedenza su quelli più alti.

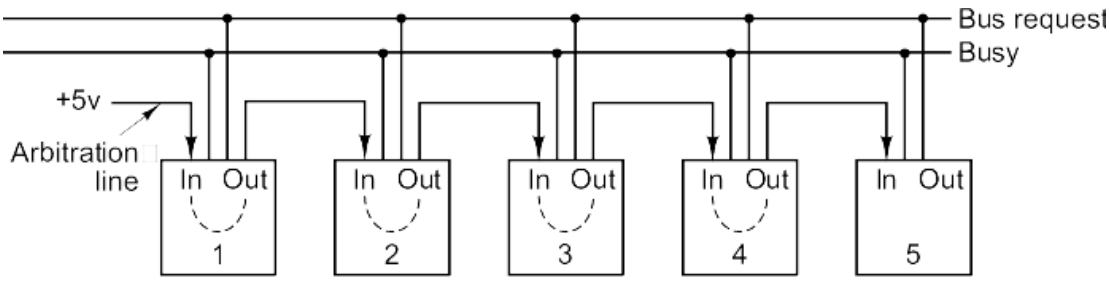
Un esempio di **arbitraggio decentralizzato**, è quello in cui vengono utilizzate 16 linee di richiesta del BUS.

Ogni dispositivo è collegato a tutte le 16 linee ed è sempre in grado di valutare autonomamente se è lui il richiedente con la priorità più elevata.

Non occorre in tal caso nessun arbitro. D'altro canto il numero massimo di dispositivi è limitato al numero di linee di richiesta e ogni dispositivo deve essere collegato a tutte le linee.

# BUS dei Calcolatori (8)

E' possibile in realtà adottare una variante **decentralizzata** del **daisy chaining**:




- La linea di Grant (concessione), invece che all'arbitro è collegata allo stato logico 1.
- Quando un dispositivo necessita del BUS, controlla se il BUS è in quel momento occupato (linea BUSY) e se il proprio Input di Grant (IN) è attivo.
- Quando il dispositivo prende il controllo del BUS, nega il proprio Output di Grant (OUT) e attiva la linea BUSY.

Come nel caso di daisy chaining i dispositivi **più a sinistra** hanno priorità maggiore.

# BUS dei Calcolatori (9)

## I primi BUS per PC

- **PC Bus IBM**: Si tratta del primo BUS per **PC/XT** introdotto dall'IBM per sistemi basati su Intel 8088 (1981). Questo Bus (sincrono) aveva 8 linee dati, 20 linee indirizzo, 2 per lettura/scrittura in memoria, 2 per lettura/scrittura di I/O, alcune linee per Interrupt e DMA: in totale 63.
- Bus **PC/AT**: variante del 1982 per veicolare le 16 linee dati e di indirizzare i 16 Mbyte indirizzabili.
- Bus **ISA**: Quando l'IBM introdusse sul mercato il PS/2 (1987), decise di ristrutturare radicalmente il BUS ormai obsoleto; tale decisione in realtà fu presa anche e soprattutto per arginare il fenomeno dei PC Cloni (IBM compatibili) che stavano togliendo a IBM grosse fette di mercato. Il nuovo BUS (Microchannel) venne coperto da un muro di brevetti ed avvocati !

  Il resto dell'industria reagì adottando uno standard proprio, il bus **ISA** (**I**ndustry **S**tandard **A**rchitecture) che è essenzialmente il bus IBM PC/AT, sincrono e funzionante a 8.33 MHz. ISA divenne presto uno standard affermato, al contrario di Microchannel, **presente** nella **maggior parte dei PC** per molti anni.

## PCI

Nel 1990 il bus ISA aveva una **larghezza di banda** (transito di informazione per unità di tempo) non sufficiente per molte periferiche e applicazioni: ad esempio le applicazioni multimediali che richiedono la riproduzione di video e audio in tempo reale, o per la grafica 3D dei videogiochi.

Infatti il bus ISA funziona a una velocità massima di 8.33 MHz ed è in grado di trasferire 2 byte per ciclo (16 linee dati); pertanto la larghezza di banda massima è di 16,7 MB/sec.

*Che larghezza di banda richiede la visualizzazione in tempo reale (30 frame per secondo) di un filmato 1024×768 a 3 byte per pixel (RGB) ?*

Circa 135 MB/sec. considerando la lettura da DVD e l'output su scheda video.

# BUS dei Calcolatori (10)

Il bus **PCI** (**P**eripheral **C**omponent **I**nterconnect bus) venne introdotto da Intel nel 1990, per far fronte alla sempre crescente necessità di ampiezza di banda.

Il bus PCI originale è un BUS sincrono con 32 linee dati e opera a 33 MHz (tempo di ciclo 30 ns) e ciò consente una larghezza di banda di 133 MB/sec.

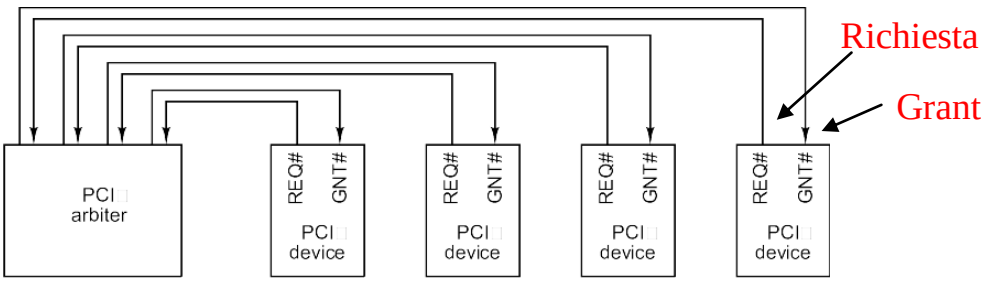
Le versioni successive di bus PCI (2.0, 2.1, 2.2, …) introdotte nel seguito hanno ulteriormente ampliato la banda massima (528 MB/sec.) portando a 64 le linee dati e a 66 MHz la frequenza di funzionamento.

Gli slot PCI sulla motherboard hanno forme leggermente diverse a seconda della tensione di funzionamento e del bus dati (32 o 64 bit).


Sul bus PCI, per limitare il numero dei contatti, le linee dati e indirizzi sono comuni (**64** bit); pertanto indirizzi e dati devono occupare fisicamente le linee in istanti diversi (sincronizzati dai fronti del clock).

Il protocollo del Bus prevede diversi tipi di **transazioni** (che durano al minimo tre cicli); alcune transazioni sono ottimizzate per limitare il numero di cicli per trasferimenti di grossi blocchi dati.

L'**arbitraggio** del Bus è centralizzato:




L'arbitro, spesso implementato in uno dei chip "Bridge", decide quale dei dispositivi richiedenti ha priorità maggiore (l'algoritmo di arbitraggio non è specificato).

# BUS dei Calcolatori (11)

## USB

L'**USB** (**U**niversal **S**erial **B**us) nacque nella metà degli anni 90' da una collaborazione tra sette aziende (Compaq, DEC, IBM, Intel, Microsoft, NEC e Nortrhern Telecom) per risolvere (una volta per tutte!) il problema del collegamento al PC di periferiche "lente". I principali obiettivi che i progettisti dell'USB si prefiggevano erano:

- Non obbligare l'utente a smontare il PC per impostare "jumper" o interruttori.
- Un solo tipo di cavo poco costoso e sufficientemente lungo.
- Alimentare i dispositivi, o almeno la maggior parte di essi, attraverso questo cavo (senza alimentatore esterno).
- Collegare molti dispositivi (fino a 127) allo stesso PC
- Supportare dispositivi che richiedono di operare in tempo reale (es. dispositivi audio e masterizzatori).
- Poter collegare/scollegare le periferiche a PC accesso e senza necessità di riavviare il PC.

L'obiettivo è stato centrato, e l'USB è oggi uno standard diffuso e in continua espansione:

- l'USB è un BUS con ampiezza di banda di **1.5 MB/s** nella versione **1.1**, di **60 MB/s** nella versione **2.0**, di **600 MB/s** nella versione **3.0** e di **2500 MB/s** nella versione **3.2**.
- il cavo contiene 4 linee (2 per i dati e 2 per l'alimentazione) nel caso di USB 1.0 e 2.0; USB 3.x aggiunge altre 5 linee per i dati.
- il protocollo supporta 4 modalità di comunicazione: control (per transazioni di configurazione e controllo), isochronous (per dispositivi che necessitano di una ampiezza di banda continuativa), bulk (per il trasferimento di grosse quantità di dati; es. stampante o scanner) e interrupt (per dispositivi che devono trasferire pochi dati ma che devono essere frequentemente interrogati; es. tastiera o mouse).
- esistono inoltre 4 tipi di pacchetti detti: token, data, handshake e special.

# BUS dei Calcolatori (12)

## ... ancora su USB

**USB OTG** (On-The-Go) consente ad una periferica di collegarsi al PC, ma anche di poter agire da Host nei confronti di altre periferiche.

Altre tecnologie della famiglia USB sono: **WUSB** (2007), che consente a USB di diventare wireless e **USB PD** (2012), che definisce uno standard di alimentazione flessibile (un canale di comunicazione dati con l'alimentatore permette alle periferiche di richiedere il livello di tensione e corrente desiderati).

## FireWire e Thunderbolt

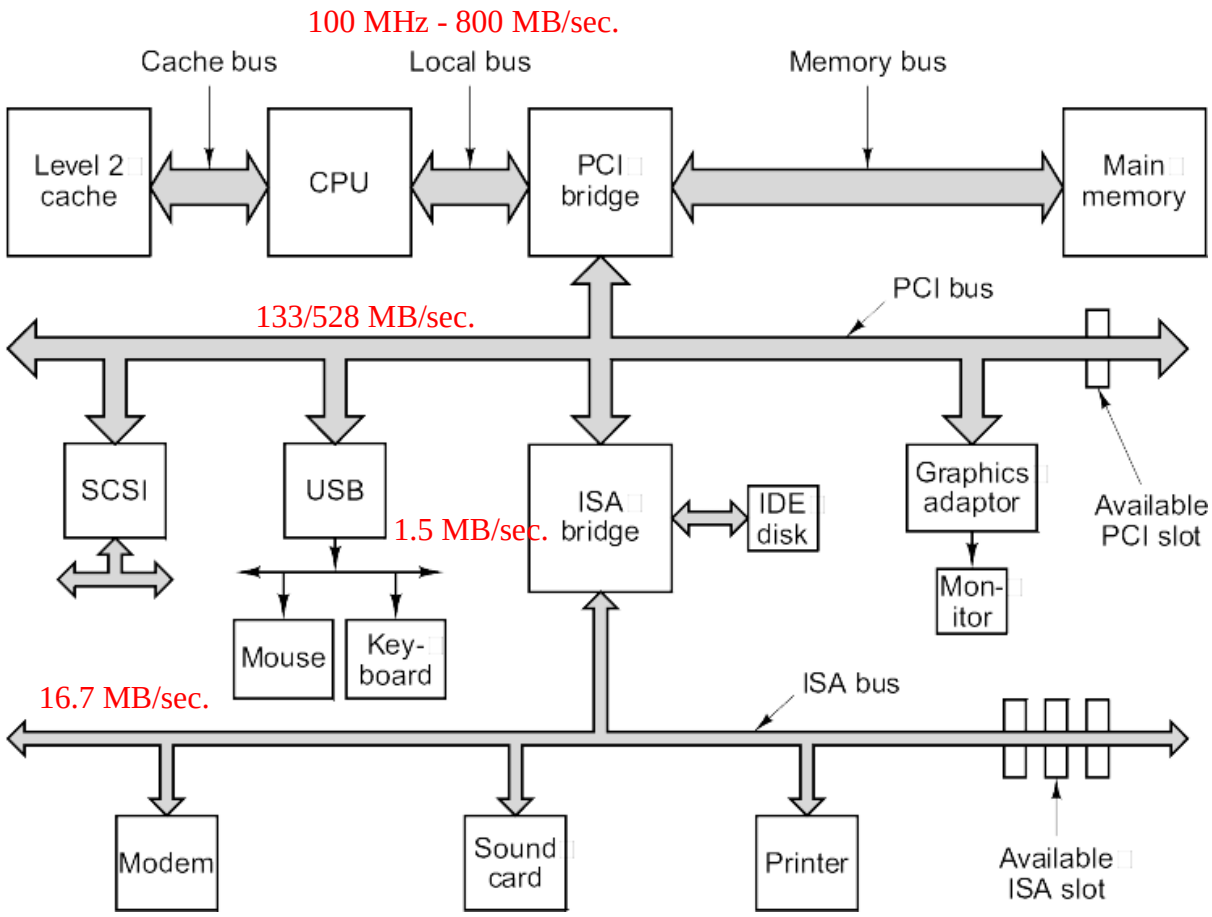
FireWire è un BUS di comunicazione seriale per dispositivi ad altre prestazioni, analogo a USB.

- La velocità di trasferimento, di 100-400Mbit/sec (**12.5-50 MB/sec**) nella prima versione, poi aumentata fino a 400MB/s nelle versioni successive.
- Possibilità di connettere fino a 63 dispositivi (fino a 1022 utilizzando bridge intermedi).
- Rispetto a USB 2.0 FireWire aveva la potenzialità di trasferire una maggiore quantità di informazioni in un tempo prefissato ed era quindi adottato da periferiche di fascia professionale (telecamere digitali, dispositivi di audio/video streaming). L'interfaccia FireWire era però più costosa di USB 2.0.
- Fra il 2008 e il 2010 FireWire è stato sostanzialmente abbandonato, a favore di USB 3 e di Thunderbolt, una tecnologia sviluppata da Intel in collaborazione con Apple, che combina i protocolli di trasferimento dati DisplayPort e PCI Express in un unico flusso dati, permettendo al connettore di gestire sia monitor che periferiche generiche. Ampiezza di banda: **1250 MB/s** (versione 1, 2011), **2500 MB/s** (versione 2, 2013), **5000 MB/s** (versione 3, 2015).

# BUS dei Calcolatori (13)

## I diversi BUS nell'architettura di un PC

L'architettura di un PC basato su **Pentium II** (1998) dove sono ben evidenziati i diversi BUS, è rappresentata in figura:



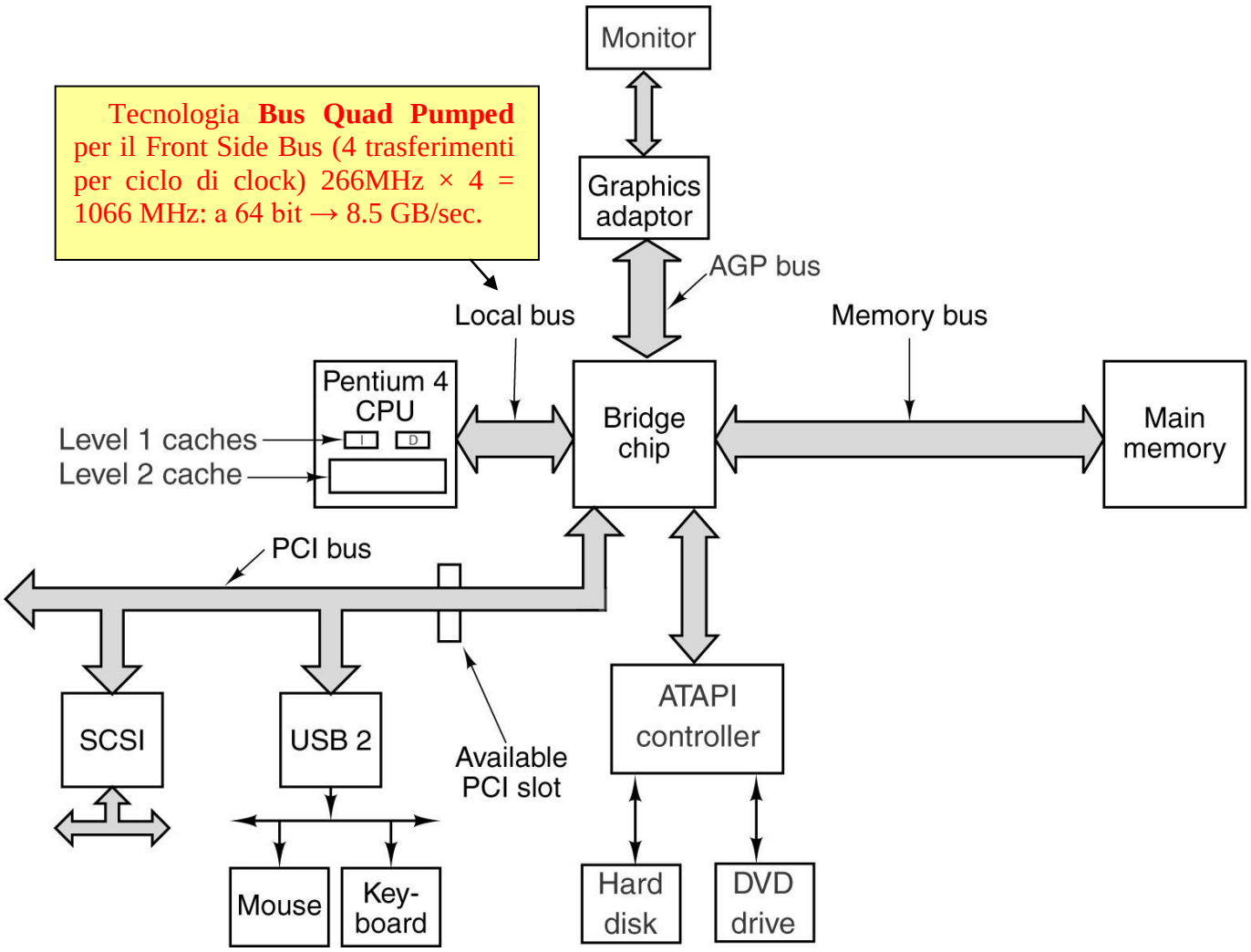

*Perché una tale complicazione ?*

- Innanzitutto per motivi di **compatibilità**, i vecchi BUS (vedi ISA), oramai lenti ed inadeguati, vengono mantenuti per consentire il funzionamento di numerosissime schede e periferiche esistenti.
- D'altro canto, bus più performanti (vedi PCI) sono necessari per l'interfacciamento di **periferiche veloci** capaci di trasferire grosse quantità di dati (es. video in tempo reale).

# BUS dei Calcolatori (14)

- sebbene PCI sia un BUS "veloce", le elevate frequenze di funzionamento delle CPU rendono necessari **BUS a più larga ampiezza di banda** per il collegamento della CPU con Cache e memoria. (es: Intel Willamette con FSB a 400 MHz, 64 bit → 3.2 GB/sec.)
- Infine, esigenze di collegamento "**plug and play**" di periferiche "**esterne**" ha portato all'introduzione del pratico e versatile BUS USB.
- Per poter connettere dispositivi diversi su un sistema multi-BUS sono necessari particolari Chip detti "**Bridge**" in grado di adattare i diversi segnali elettrici e gestire i diversi protocolli.

L'architettura di un PC (2003-2004) basato su **Pentium IV** (è sparito il bus ISA):



# BUS dei Calcolatori (15)

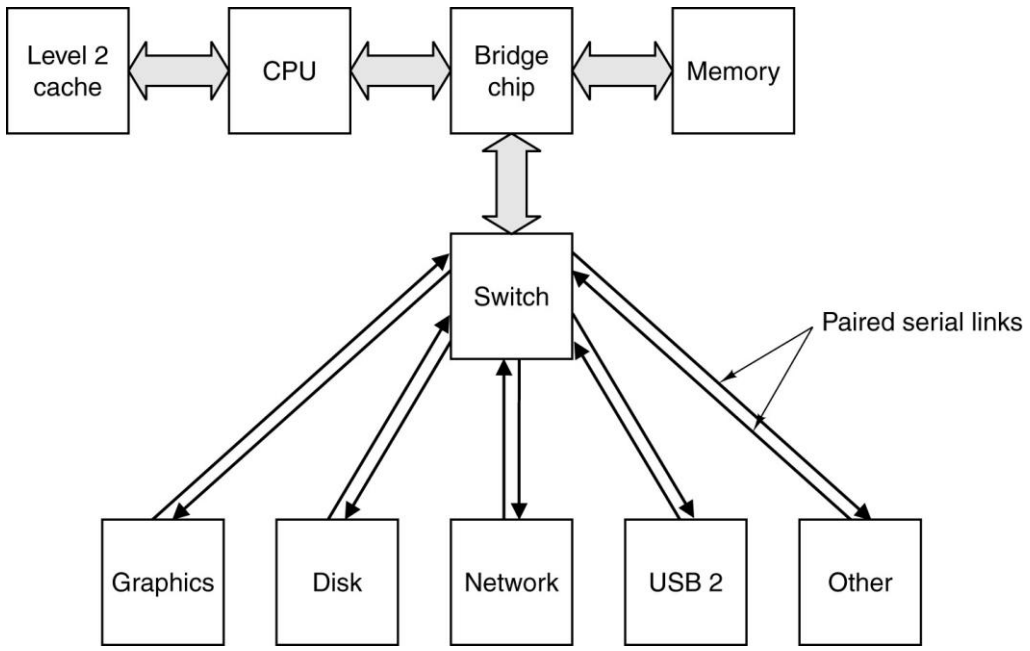
## PCI Express

Dallo schema di cui al lucido precedente appare chiaro che il bus PCI non è più l'elemento centrale che tiene unite le varie parti del PC, parte di questo ruolo è stato assunto dal/i chip Bridge (chipset).

Il cuore del problema è che vengono introdotti dispositivi sempre più veloci per i quali il bus PCI non è più adeguato, e la soluzione fino ad ora portata avanti è stata quella di prevedere porte addizionale sul/i bridge.

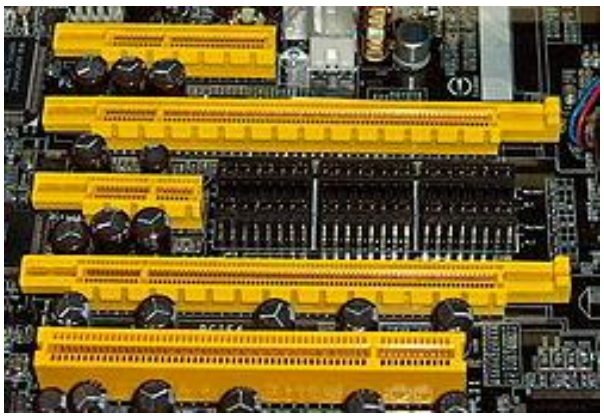
La strada intrapresa (dal 2003) prende il nome di **PCI Express** o **PCIe** (il nome deriva da una pura operazione commerciale legata alla fama del nome PCI, in quanto non si tratta di un'estensione ma di una cosa completamente diversa!) determina un cambio di rotta radicale rispetto al passato:

- il bus diventa **seriale** con connessioni **punto a punto** ad alta velocità;
- sono possibili collegamenti punto a punto simultanei tra coppie master-slave (**multi-master**).



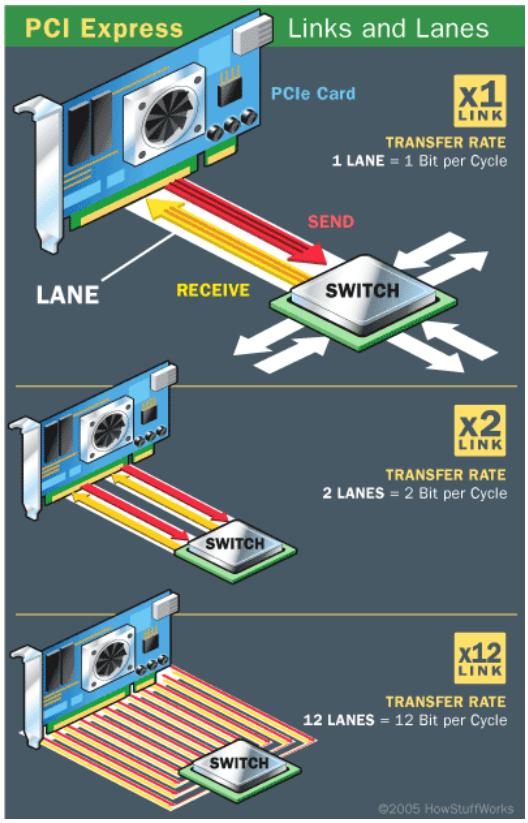
# BUS dei Calcolatori (16)

- prevede più **canali** (lanes) **indipendenti** (fino a **32**) che possono essere combinati per aumentare la banda: slot ×1, ×4, ×8,×16, ×32;
- i device che richiedono elevata banda eseguono comunicazione in parallelo su più canali usando slot appositi: ad esempio è tipico per la scheda grafica usare uno slot ×16;



Dall'alto verso il basso:

- PCI Express ×4
- PCI Express ×16
- PCI Express ×1
- PCI Express ×16
- Conventional PCI (32-bit)




- lo scambio di dati avviene attraverso un protocollo basato su pacchetti (simile alla comunicazione in rete).
- La **larghezza di banda** di ciascun canale dipende dalla versione PCI Express:

| Versione | Banda per Canale | Banda ×16 |
|---|---|---|
| 1.0a (2003) | 250 MB/s | 4 GB/s |
| 2.0 (2007) | 500 MB/s | 8 GB/s |
| 3.0 (2010) | 1 GB/s | 16 GB/s |
| 4.0 (2017) | 2 GB/s | 32 GB/s |

# Core i7-5960X and Chipset X99

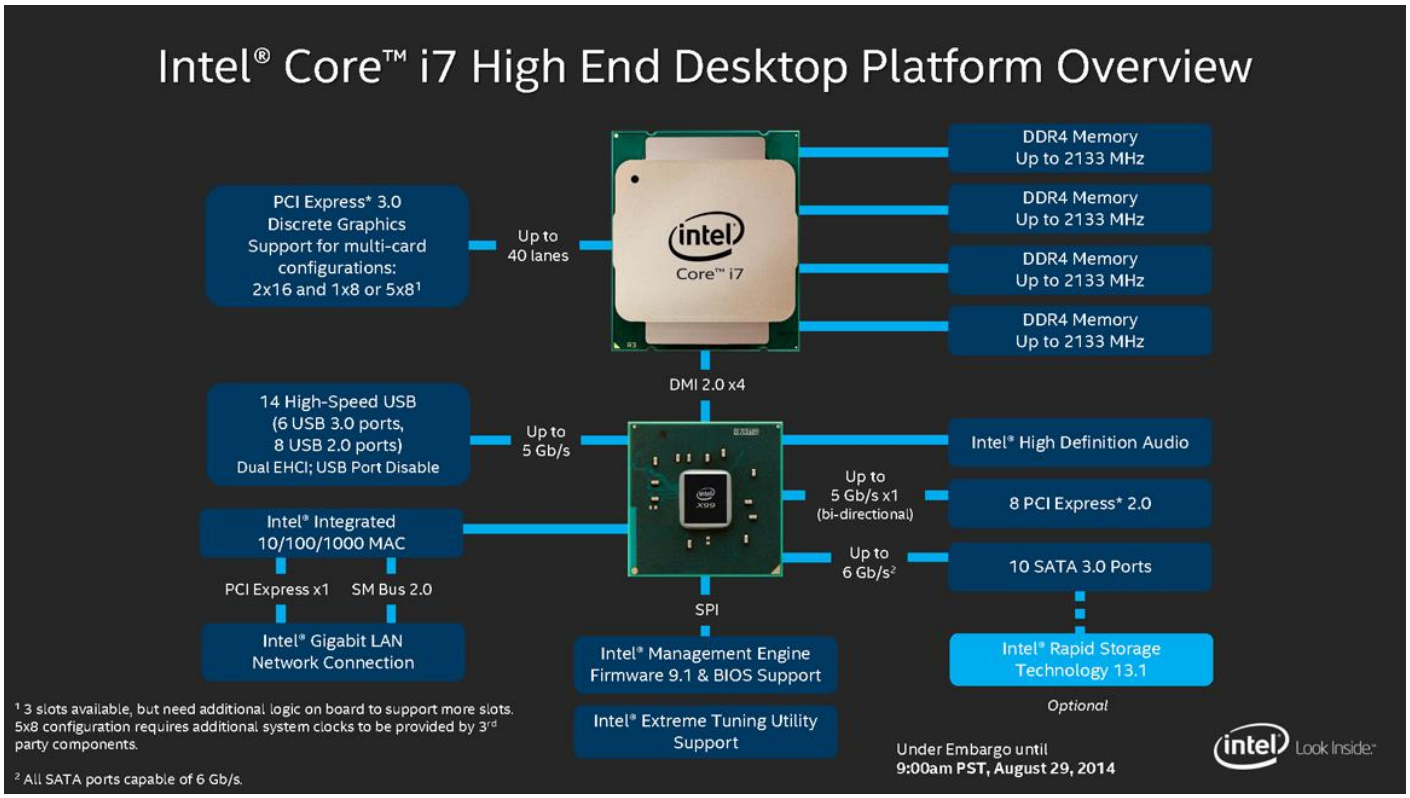
**(2014)**

La **CPU** contiene al suo interno:

- Fino a 40 canali PCI Express 3.0
- Interfacce alla memoria RAM DDR4

Il collegamento tra CPU-Chipset su Interfaccia Intel DMI 2.0 è basato su 4 canali i cui segnali sono simili a PCI-E 2.0 per una banda totale effettiva di circa 2 GB/s.

Il **Chip X99** consente il collegamento con il resto delle periferiche (USB, rete, audio, dischi, ecc.):



# Core i9 serie X and Chipset X299

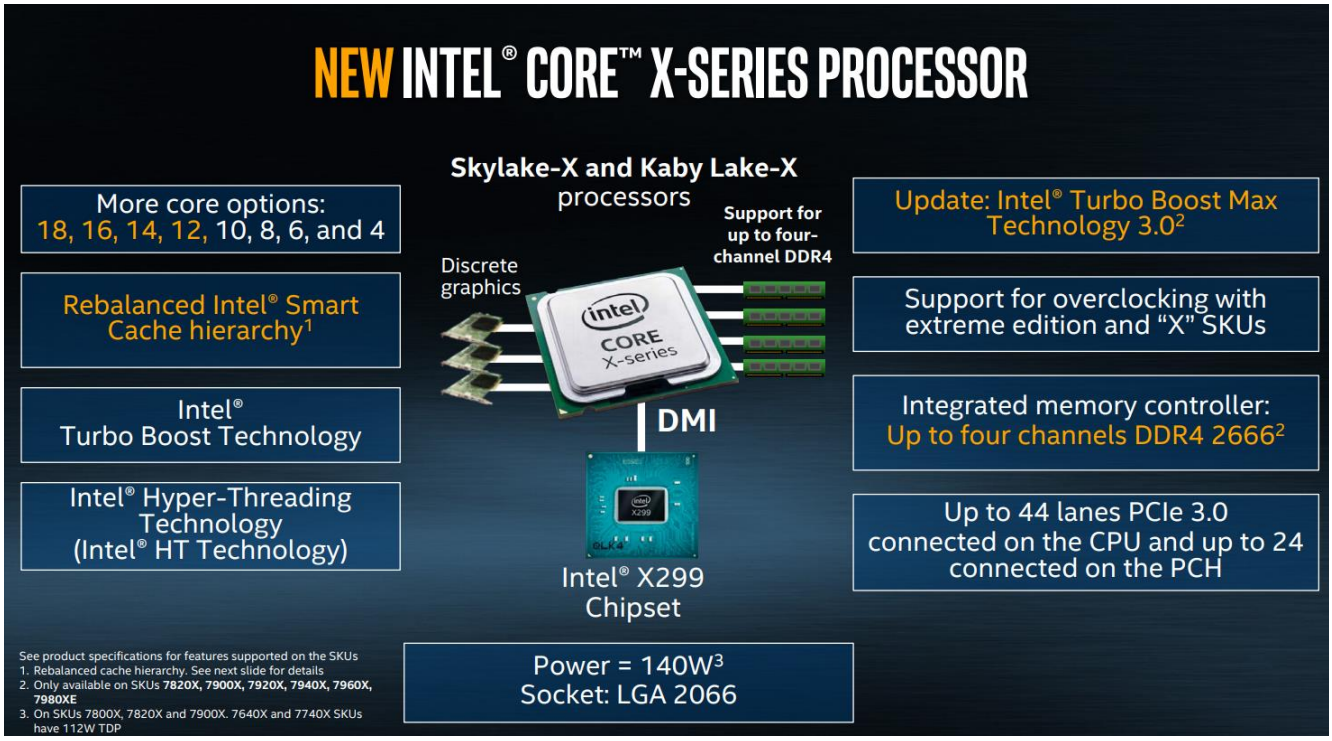
**(2017)**

La **CPU** contiene al suo interno:

- Fino a 44 canali PCI Express 3.0
- Interfacce alla memoria RAM DDR4

Il collegamento tra CPU-Chipset su Interfaccia Intel DMI 3.0 è basato su 4 canali i cui segnali sono simili a PCI-E 3.0, per una banda totale effettiva di circa 4GB/s.

Il **Chip X299** consente il collegamento con il resto delle periferiche (USB, rete, audio, dischi, ecc.):



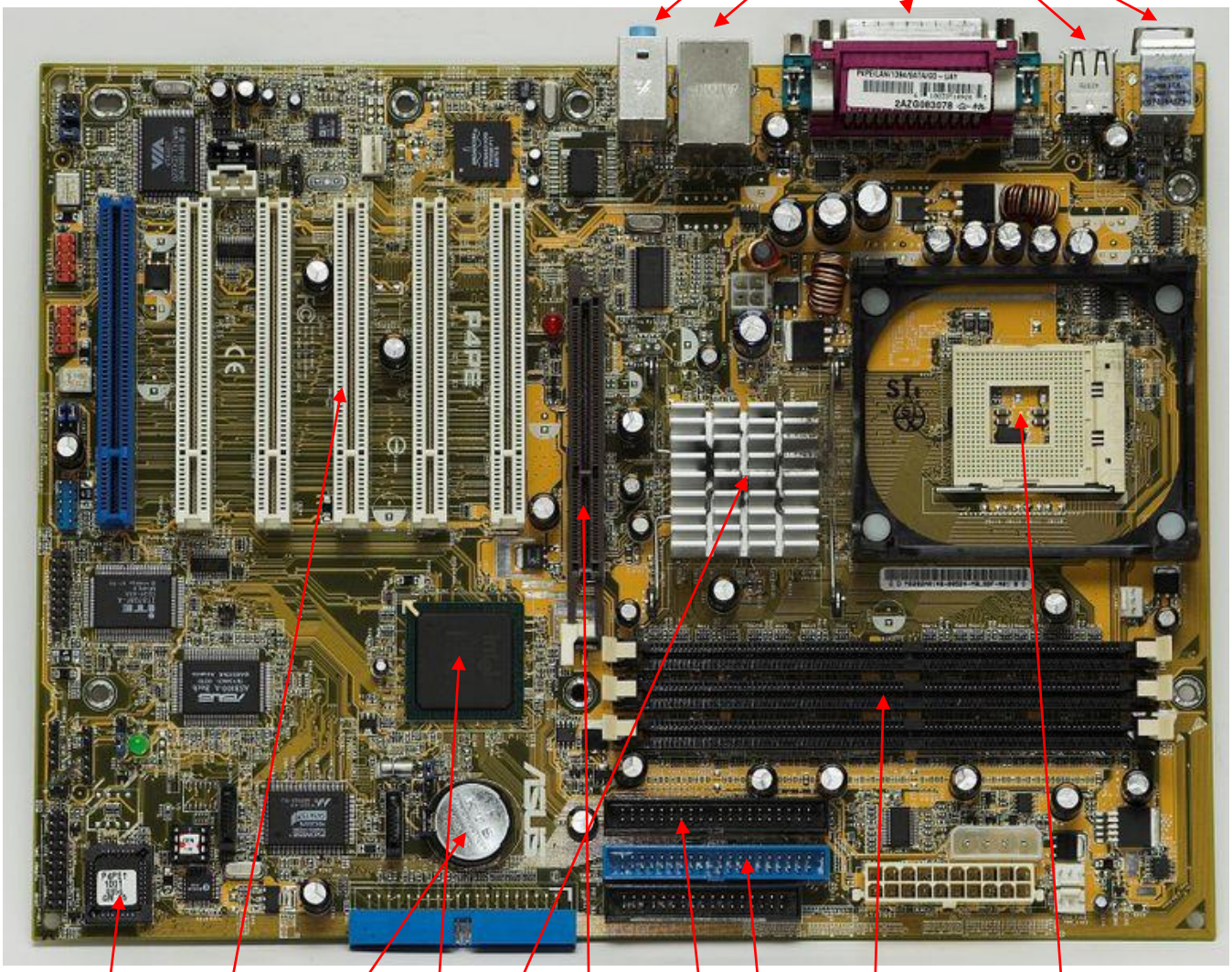
# Motherboard

La **scheda madre** (motherboard) è una scheda elettronica che raccoglie in sé tutta la circuiteria e i collegamenti di interfaccia tra i vari componenti interni principali del PC:

- L'alloggiamento per la CPU: CPU socket
- Il bus interno: è la via principale per lo scambio di informazioni tra i vari dispositivi. Vi possono essere più bus differenziati in base alla velocità di comunicazione (PCI, PCI-E, ...).
- Slot di espansione: si tratta di porte (connettori) verso il/i bus interno/i progettate per permettere di collegare alla scheda madre altre schede di espansione.
- Gli slot per l'inserimento dei moduli di RAM: tipicamente memorie sincrone di tipo DDR. Coppie di slot di colori diversi corrispondono a *double channel* DDR.
- I controllori dei dispositivi di I/O fondamentali: SATA, PATA per hard disk, CD e DVD.
- I connettori per ulteriori dispositivi di I/O: al fine di semplificare la comunicazione tra calcolatori e dispositivi di I/O sono stati definiti alcuni protocolli standard per la loro connessione: USB, FireWire, Ethernet, ecc.
- Il chipset: è l'insieme di chip (uno o più) che permette il collegamento della CPU alla maggior parte delle periferiche:
  - USB
  - Dischi
  - Rete
  - Audio
  - …

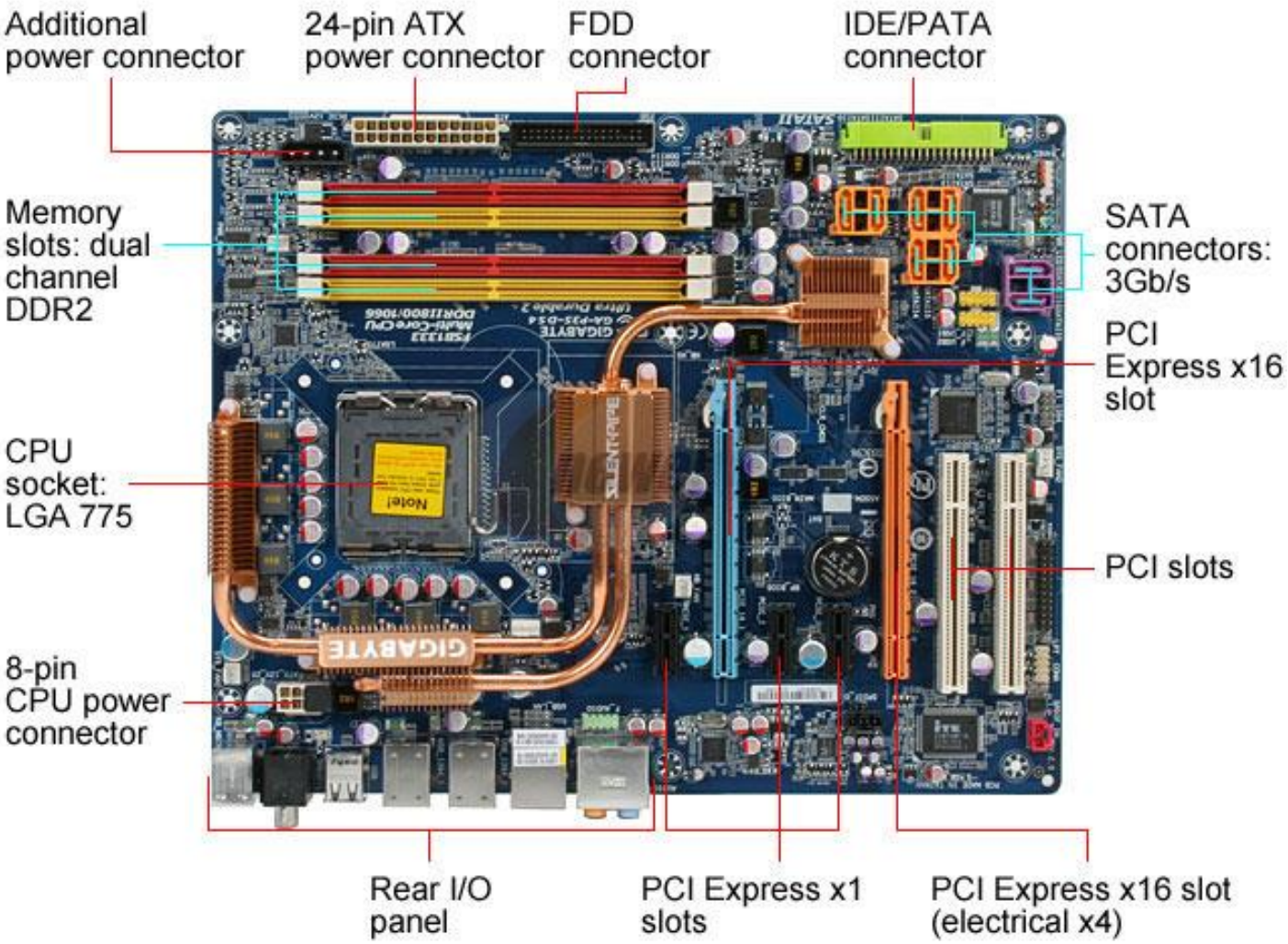
# Motherboard (2)

In figura è mostrata la scheda madre di un PC di classe Pentium (2002)



# Motherboard (3)

In figura è mostrata la scheda madre di un PC (2007)



# Interfacciamento di Periferiche

L'interfacciamento "intelligente" di periferiche richiede l'implementazione di alcune tecniche hardware e/o software che permettono di massimizzare l'efficienza del sistema. Gli obiettivi sono molteplici:

- Da un lato si vuole evitare che una periferica, ad esempio un disco che deve trasferire in memoria un 1 MB di dati tenga impegnata la CPU durante tutto il trasferimento, ovvero che richieda che ogni parola venga prima trasferita dal disco alla CPU e in seguito da questa alla memoria. La tecnica del **DMA** (**D**irect **M**emory **A**ccess) consente di mettere in diretto contatto dispositivi e memoria in modo che questi possano operare sul bus indipendentemente dalla CPU.
- In secondo luogo, si vuole evitare di mantenere la CPU costantemente impegnata nel monitorare (polling) una o più periferiche che devono compiere un certo lavoro. Tramite la tecnica dell'**Interrupt** la CPU può dedicarsi ad altro ed essere interrotta solo al momento opportuno.
- Infine si pone il problema di come pilotare tramite la CPU dispositivi esterni "lenti", quali interruttori, schede di I/O, dispositivi seriali; il loro collegamento diretto sul BUS risulta alquanto problematico: è preferibile in questi casi utilizzare **Chip di interfacciamento** seriali (UART) o paralleli (PIO) che semplificano il compito di controllo alla CPU.

## DMA

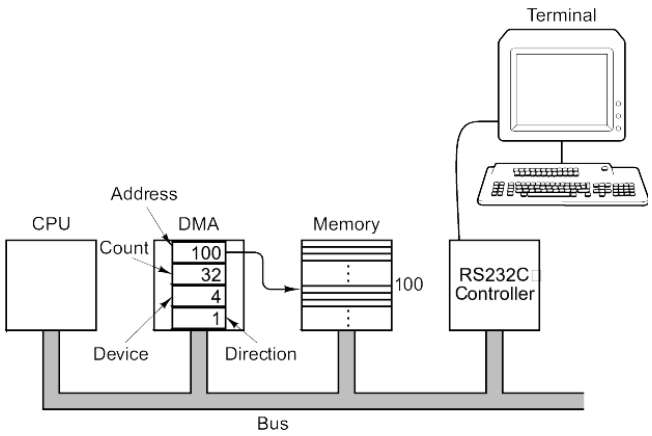
La tecnica del DMA (accesso diretto alla memoria) consente il trasferimento di blocchi dati, anche di notevoli dimensioni, tra la memoria e le periferiche, senza richiedere l'intervento della CPU. In particolar modo è utilizzato da: controller dei dischi, schede grafiche, schede audio, ecc...
A tal fine è necessario disporre di un circuito denominato **controllore DMA** (che può essere interno alla CPU o in un Chip esterno collegato al BUS) che governa il funzionamento:

- Il controllore DMA ha al suo interno 4 registri, che vengono inizialmente caricati dal software attraverso la CPU:

# Interfacciamento di Periferiche (2)

1. Il primo contiene l'indirizzo iniziale di memoria dal quale leggere o scrivere.
2. Il secondo indica quanti byte o parole devono essere trasferiti
3. Il terzo specifica il numero di identificazione del dispositivo o l'indirizzo dello spazio di I/O da usare.
4. Il quarto indica se i dati vanno letti o scritti dal/sul dispositivo.

Per scrivere un blocco di 32 byte dall'indirizzo 100 di memoria al dispositivo 4, la CPU scrive 100, 32, 4, 1 (supponiamo che 1 indichi WRITE) nei 4 registri.




- A questo punto la CPU da il via all'operazione di trasferimento e il controllore usa il Bus per leggere 1 per volta i 32 byte dal dispositivo e scriverli in memoria. Il controllore agisce dunque come master del Bus ed è in grado di pilotare tutte le linee necessarie così come farebbe la CPU.
- Al termine del trasferimento la CPU viene avvertita che l'operazione è terminata. Questo avviene normalmente tramite interrupt.

Durante il trasferimento in DMA la CPU è rimasta dunque libera, anche se il Bus è stato occupato dal DMA e quindi l'esecuzione parallela di altro codice che richiede l'accesso al Bus non è stata possibile. In realtà, grazie alla presenza di cache e di bus multipli, diverse operazioni di DMA e di utilizzo contemporaneo della CPU sono oggigiorno eseguite dai calcolatori.

Un tipico esempio è il trasferimento di grandi quantità di dati verso la scheda grafica in concomitanza con la normale esecuzione della CPU. Operare in ambienti grafici complessi (es. Windows) richiede il trasferimento continuo di grossi volumi dati e ciò costituirebbe un grosso problema se fosse la CPU a doversi occupare di tutto.

# Interfacciamento di Periferiche (3)

## Interrupt

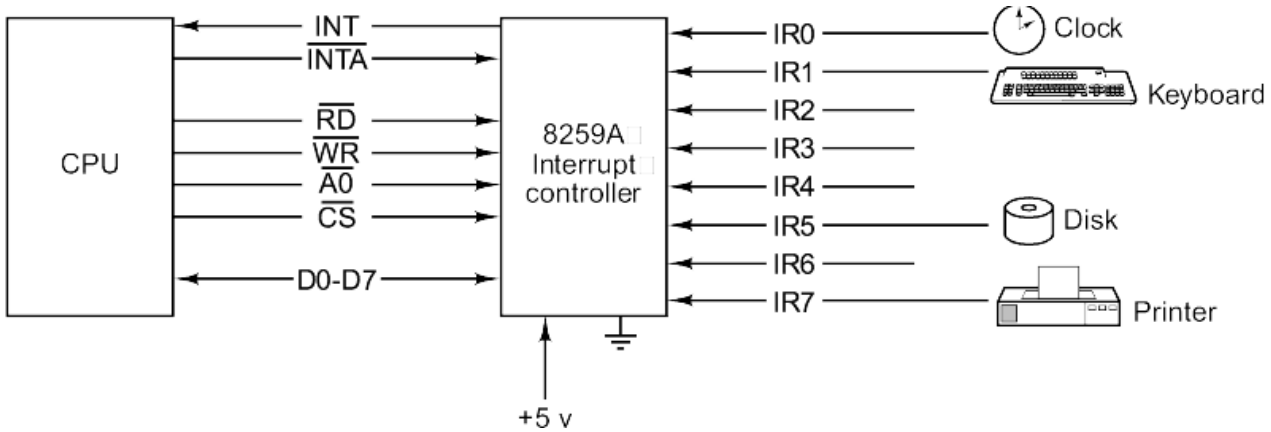
La tecnica più elementare per far sì che la CPU sia costantemente informata sullo stato di una periferica è il cosiddetto **polling**, ovvero un ciclo l'interrogazione periodica e regolare della periferica. Questo implica però la necessità di:

- Sprecare preziosi cicli (e quindi tempo) per l'interrogazione periodica.
- Scrivere codice molto più complesso; si pensi al caso di decine di periferiche che devono essere gestite dalla stessa CPU.

Attraverso la tecnica dell'**interrupt** una periferica può inviare un segnale elettrico su una determinata linea della CPU (INT) per segnalare un determinato evento. La CPU può decidere di accettare o meno l'interruzione ed eseguire o meno la richiesta.

Data la presenza nel sistema di diverse periferiche, diversi interrupt potrebbero essere **scatenati in concomitanza** e occorre un sistema di arbitraggio basato su priorità (qualcosa di analogo a quello che abbiamo visto per l'arbitraggio del Bus).

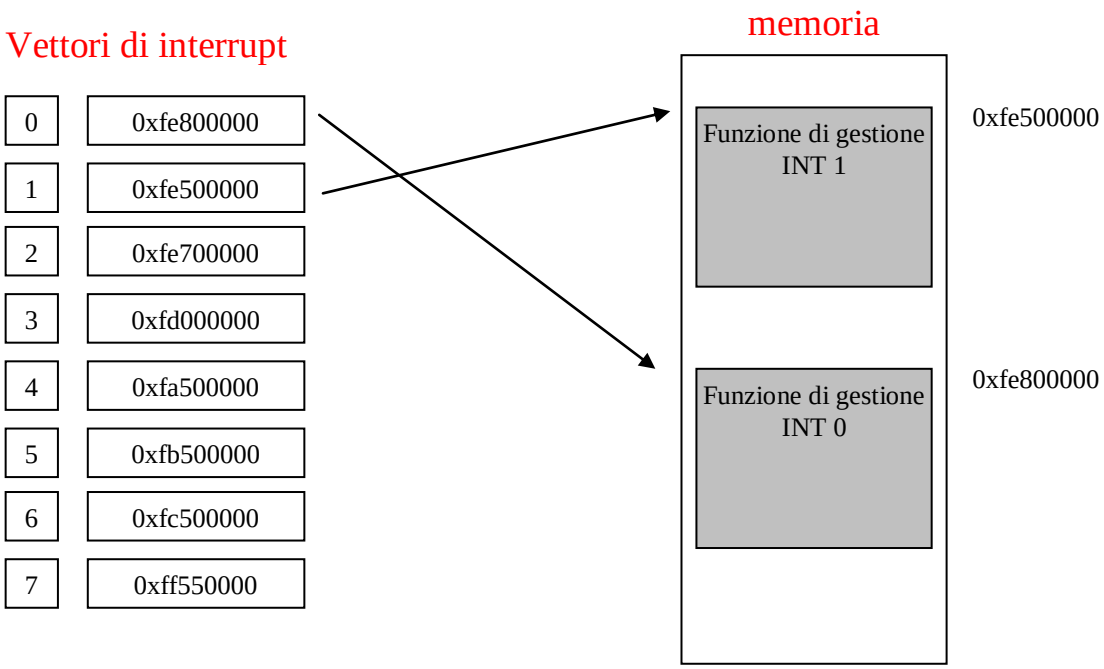
Per la gestione degli interrupt, i sistemi PC utilizzano normalmente un Chip (Intel 8259A) che si interpone tra la CPU e le linee di richiesta interrupt provenienti dalle periferiche.



# Interfacciamento di Periferiche (4)

Il Chip 8259A può controllare fino a **8 linee** di richiesta di interrupt:

- Quando un dispositivo richiede interrupt è sufficiente che attivi la propria linea IRx.
- Quando vengono attivate una o più linee il controllore attiva la linea INT (INTerrupt) della CPU. Quando la CPU è in grado di accettare l'interrupt lo segnala attraverso la linea INTA (INTerrupt Acknowledge).
- A questo punto il controllore invia alla CPU informazione circa la periferica (quella con più elevata priorità se sono presenti più richieste) che ha inviato la richiesta; tale informazione viene inviata sul bus dati utilizzando una particolare transazione di bus.
- L'hardware della CPU utilizza questa informazione come offset in una tabella di puntatori, chiamati **vettori di interrupt**, per trovare l'indirizzo della funzione da eseguire in risposta all'interrupt.




- Al termine dell'esecuzione la CPU riprende la normale esecuzione del programma.