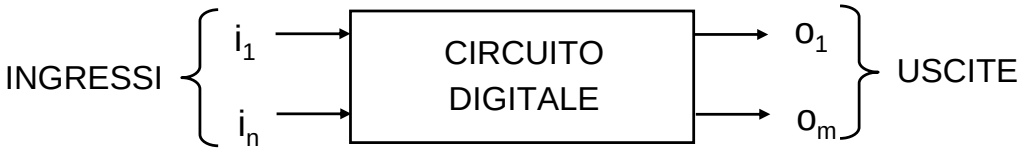
Parte III

Logica Digitale e Memorie

Circuiti digitali



- Circuiti elettronici i cui ingressi e le cui uscite assumono solo due livelli
- Al circuito sono associate le *funzioni* che calcolano le uscite a partire dagli ingressi

$$\begin{cases} o_1 = f_1(i_1,\ldots,i_n) \\ \vdots \\ o_m = f_m(i_1,\ldots,i_n) \end{cases}$$

Funzioni logiche (booleane)

$$y = f(x_1,\ldots,x_n) \qquad y, x_1,\ldots,x_n \in \{0,1\}$$

$$\{0,1\}^n \xrightarrow{f} \{0,1\}$$

- Sia le variabili indipendenti che la variabile dipendente sono *booleane*
- Le variabili booleane sono definite in un dominio con due soli valori possibili:

Tavola della verità

x_1	x_2	.....	x_{n-1}	x_n	f
0	0	.....	0	0	0
0	0	.....	0	1	1
:	:		:	:	:
1	1	.....	1	1	0

- Una funzione booleana è rappresentabile tramite la sua *tavola della verità*
- La tavola della verità specifica i punti in cui la funzione assume valore 1 (cioè *vero*)
- 2^n combinazioni di ingresso
- 2^{2^n} *funzioni distinte* di n variabili

Funzioni booleane (esempi)

• Con n=1 si hanno 4 funzioni:

x_1	f_0	f_1	f_2	f_3
0	0	0	1	1
1	0	1	0	1

(f_2 è detta **NOT**)

• Con n=2 si hanno 16 funzioni distinte, tra cui:

x_1	x_2	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1

• Le funzione f_1 è detta **AND** a funzione f_7 è detta **OR**

Algebra circuitale (algebra booleana)

• È una *struttura algebrica* (insieme più operatori)

• *Reticolo distributivo complementato*

Insieme: ***I* = { 0,1 }**
Operatori: **AND** , **OR**
Complementazione: **NOT**

Notazione

• Se x e y sono due variabili *booleane*:
  – L' **AND** di **x** e **y** si indica con **x · y**
  – L' **OR** di **x** e **y** si indica con **x + y**
  – Il **NOT** di **x** si indica con $\overline{\mathbf{x}}$

Proprietà dell'algebra booleana

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

Espressioni algebriche

Teor.: *ogni funzione booleana è* algebrica*, cioè rappresentabile con un'espressione dell'algebra*

Prima Forma Canonica $f = \sum_{j=1..m} \prod_{i=1..n} x_{ij}^{*}$

- $\mathbf{x_{ij}}^{*}$ vale $\mathbf{x_i}$ oppure $\overline{\mathbf{x_i}}$
- **f** è espressa come **OR** di tutte le combinazioni per cui la funzione assume valore vero (somma di *mintermini*)

Qualsiasi funzione booleana può essere messa in prima forma canonica

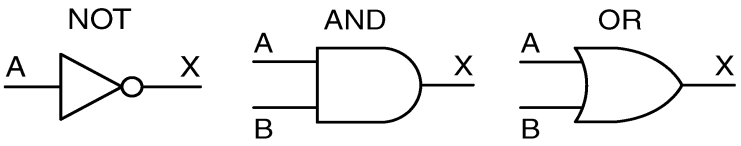
Funzioni booleane (esempio)

- Tre variabili booleane **A**, **B**, **C**
- *Funzione di maggioranza* M: è vera solo se almeno due delle tre variabili sono vere

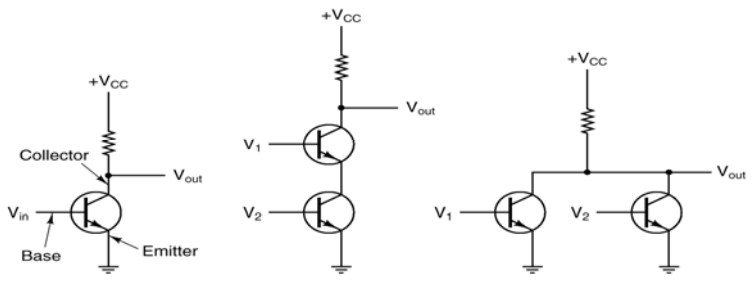
A	B	C	M	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	← $\overline{A}BC$
1	0	0	0	
1	0	1	1	← $A\overline{B}C$
1	1	0	1	← $AB\overline{C}$
1	1	1	1	← ABC

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

Porte logiche



- Circuiti elementari che realizzano gli operatori dell'algebra
- Le porte logiche vengono realizzate con circuiti elettronici:



NOT **NAND** **NOR**

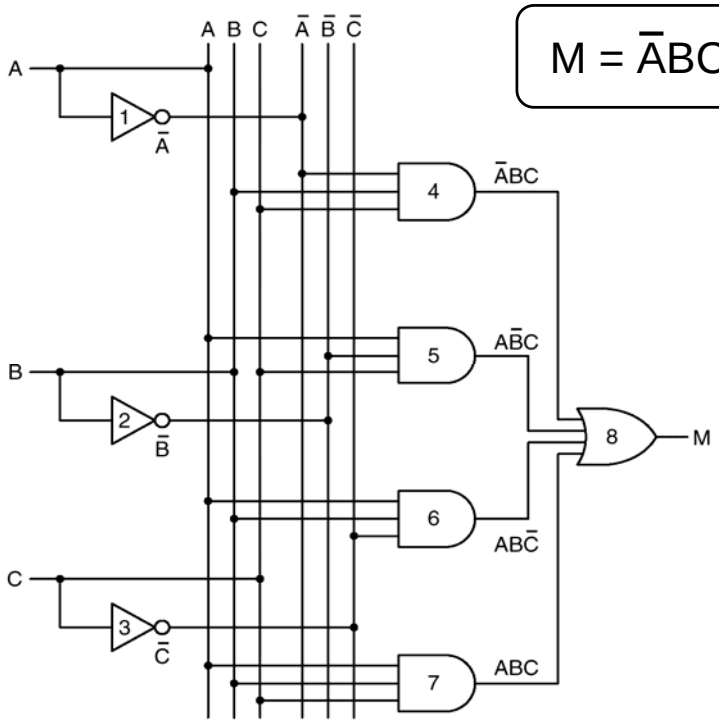
Circuiti logici

- Tutte le funzioni booleane sono algebriche
- Un'espressione algebrica comprende solo operatori dell'algebra di Boole: **AND**, **OR**, **NOT**

Qualsiasi funzione booleana può essere calcolata con un circuito realizzato con sole porte AND, OR *e* NOT

- Ogni funzione booleana può essere messa in *prima forma canonica*
- È immediato passare dalla prima forma canonica al circuito logico che la calcola

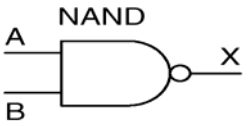
Esempio: funzione di maggioranza

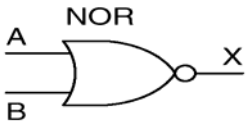


$$M = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

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

Porte NAND e NOR

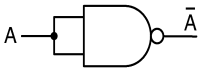


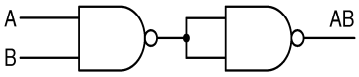


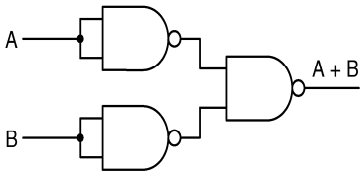
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

È possibile simulare **AND**, **OR** *e* **NOT**, *e quindi realizzare qualsiasi circuito*, *usando soli* **NAND** *oppure soli* **NOR**

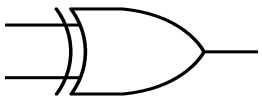




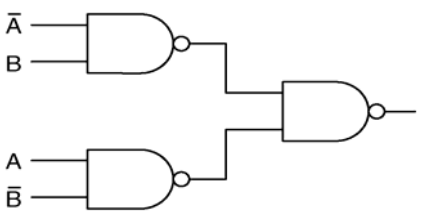


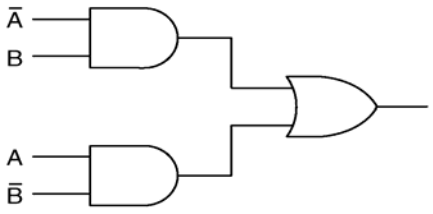
Porta XOR

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

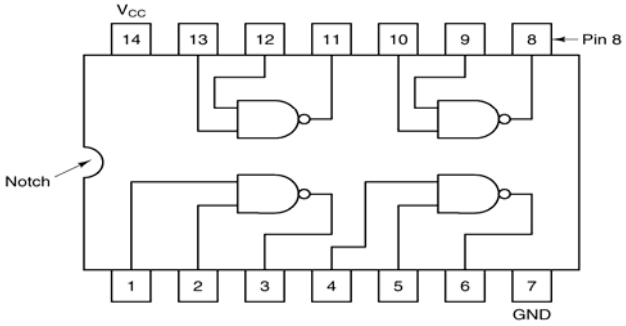


- Calcola la funzione *OR Esclusivo*: dà uscita **1** (vero) quando uno solo degli ingressi (*ma non entrambi*) vale **1**
- Facilmente realizzabile con porte **AND**, **OR** e **NAND**





Circuiti integrati

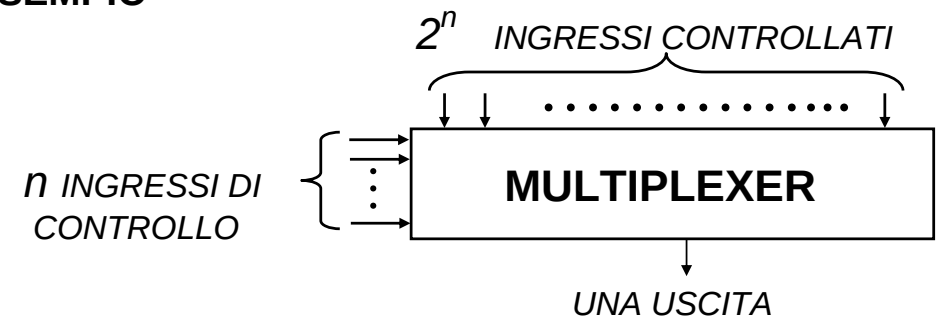


- Molte porte realizzate sulla stessa piastrina di silicio (*chip*)
- Contenitori da 14 a 68 piedini
- Vari livelli di integrazione:
  - **SSI** (*Small Scale*) 1-10 porte
  - **MSI** (*Medium Scale*) 10-100 ”
  - **LSI** (*Large Scale*) 10^2-10^5 ”
  - **VLSI** (*Very Large Sc.*) > 10^5 ”
- Tempi di commutazione 1-20 nsec

Circuiti combinatori

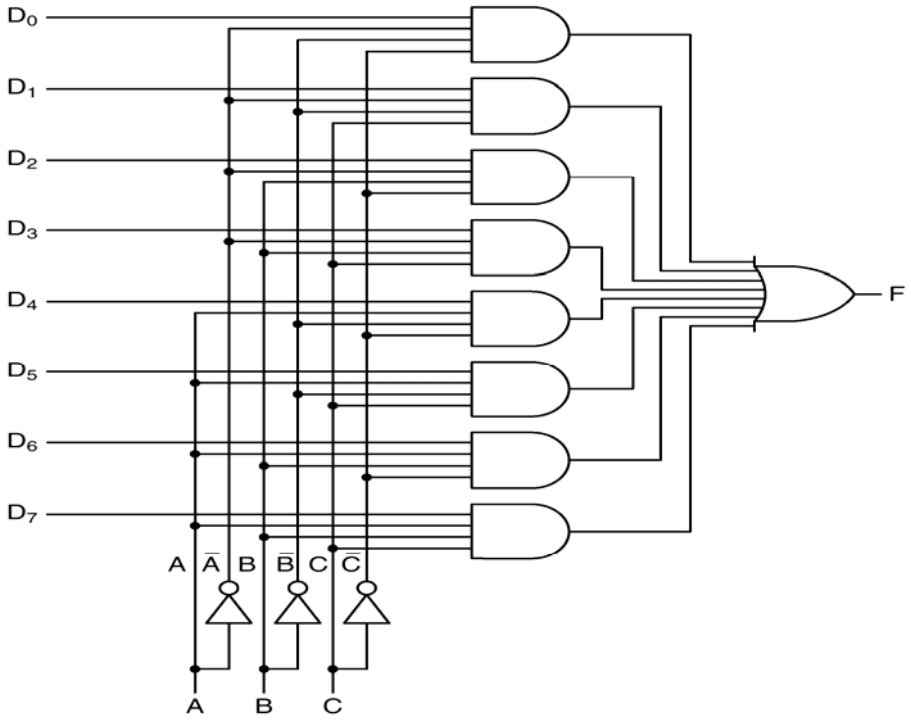
Circuiti in cui l’uscita dipende solo dagli ingressi, e non dallo stato cioè dalla storia passata

ESEMPIO



Gli ingressi di controllo selezionano quale degli ingressi controllati viene mandato in uscita

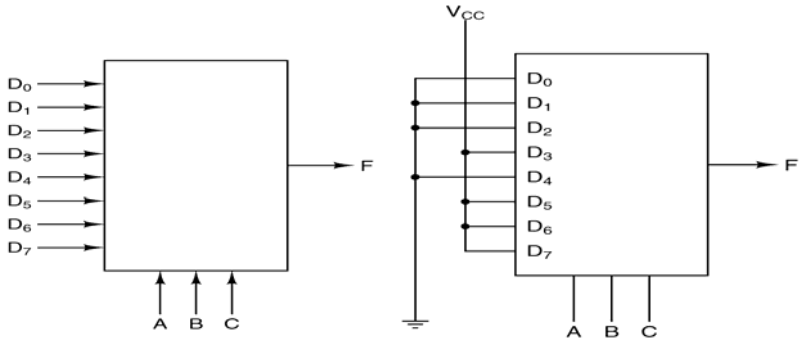
Multiplexer (circuito)



Calcolo di funzioni tramite multiplexer

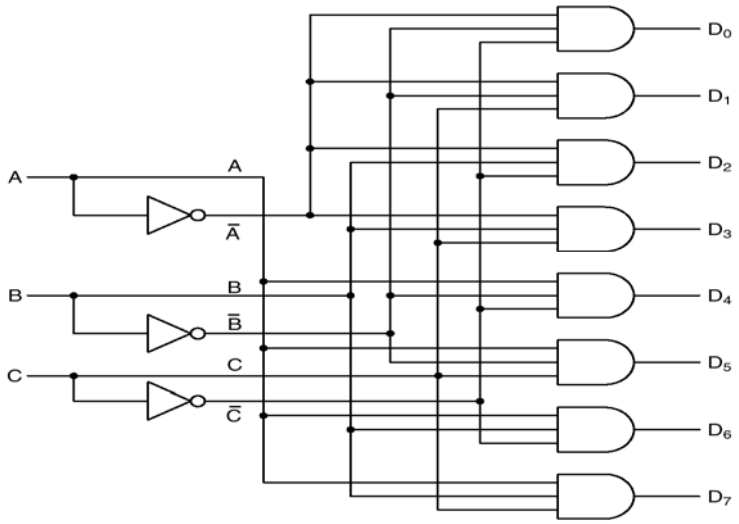
- Con un multiplexer ad **n** si può calcolare qualsiasi funzione di **n** variabili booleane
- Gli ingressi controllati corrispondono ai *mintermini*
- Si cablano a **0** o **1**, a seconda che il *mintermine* compaia o meno nella forma canonica

ESEMPIO: *Funzione di maggioranza*



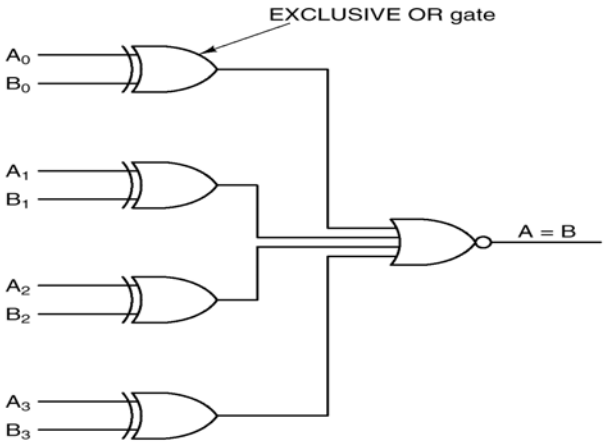
A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Decodificatore



- Circuito a n ingressi e 2^n uscite
- Una ed una sola delle 2^n uscite assume valore vero in corrispondenza a ciascuna delle 2^n configurazioni di ingresso

Comparatore



- Compara i bit omologhi di due stringhe
- L'uscita vale 1 se e solo se $A_i=B_i$ $\forall$ i
- Se **$A_i=B_i$** allora **A_i XOR $B_i = 0$**
- Il **NOR** dà uscita **1** solo quando *tutti* i suoi ingressi valgono **0**

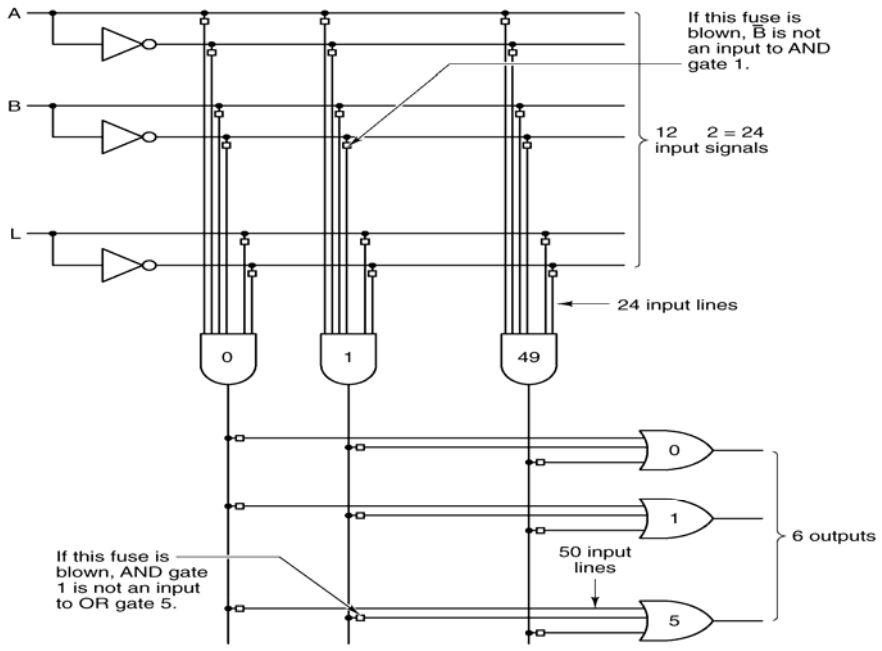
Programmable Logic Arrays (PLA)

- Permette di realizzare una ‘qualsiasi’ funzione (in prima forma canonica)
- Circuito configurabile tramite bruciatura (interruzione) di connessioni
- Permette di calcolare più funzioni
- Due sezioni nel circuito:

  A) *Generazione di un insieme di mintermini*

  B) *Selezione dei mintermini da inviare a ciascuna delle uscite*
- **Limitazioni**
  - Numero limitato di mintermini generati << 2^n
  - Numero limitato di mintermini in ingresso a ciascuno degli OR di uscita

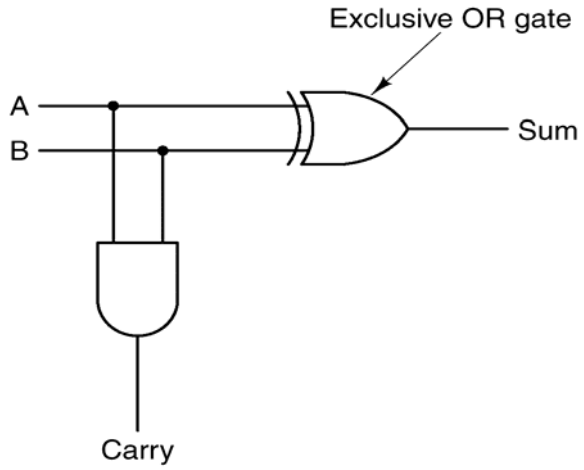
PLA: realizzazione circuitale



Genera 6 funzioni di 12 variabili; massimo 50 mintermini (su 4096)

Semiaddizionatore (*half adder*)

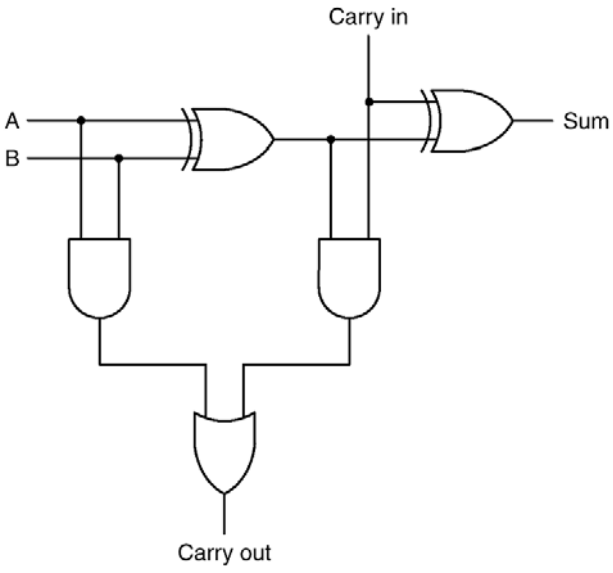
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



- Circuito a 2 ingressi e 2 uscite: somma e riporto (*carry*)
- Non può essere usato per la somma di numerali a più bit, dove occorre sommare anche il riporto della cifra precedente

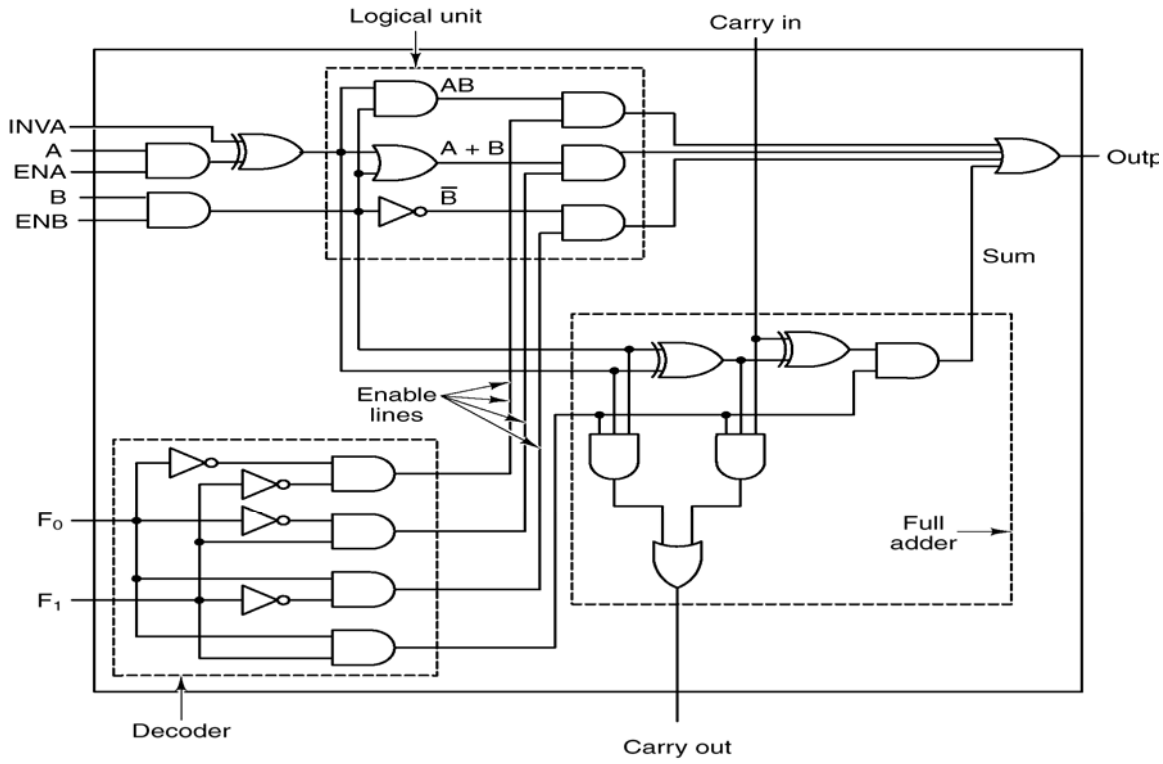
Addizionatore completo (*full adder*)

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



- Circuito a 3 ingressi e 2 uscite
- Riceve il riporto dalla cifra precedente

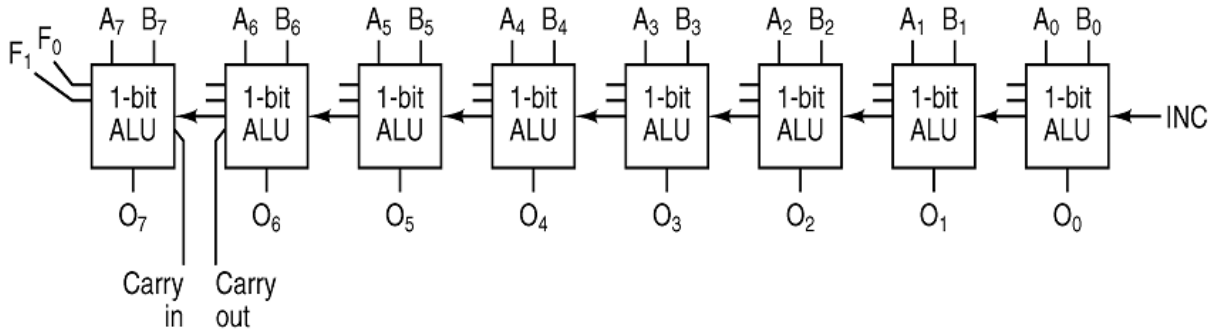
ALU a 1 bit (bit slice)



ALU a 1 bit (continua)

- Costituisce il modulo base (*slice*) con cui sono realizzate *Unità Aritmetico Logiche* (ALU) per operandi a n bit
- Gli ingressi **A** e **B** sono bit omologhi degli operandi
- Gli ingressi $\mathbf{F_0}$ e $\mathbf{F_1}$ selezionano la funzione calcolata:
  - **00**: **AND**
  - **01**: **OR**
  - **10**: **NOT**
  - **11**: **SUM**
- Gli ingressi **ENA** ed **ENB** sono segnali di *enable*
- **INVA** permette di negare l'ingresso **A**
- Valore di default **ENA=ENB=1** e **INVA=0**
- Le uscite sono due: **Out** per il risultato e **Carry** per il riporto

ALU a n bit



- Realizzata connettendo **n** ALU ad 1 bit (*bit slices*)
- *Problema*: propagazione dei riporti
- Ciascuno stadio deve attendere il riporto dal precedente
- Tempo di addizione lineare con **n**
- Nessun ritardo per le altre operazioni
- L'ingresso **INC** incrementa la somma di **1** (**A+1**, **A+B+1**)

Circuiti sequenziali

Sono circuiti in cui i valori delle uscite del circuito non dipendono solo dai valori attuali *degli ingressi ma anche dai loro* valori precedenti*, cioè dalla* storia passata

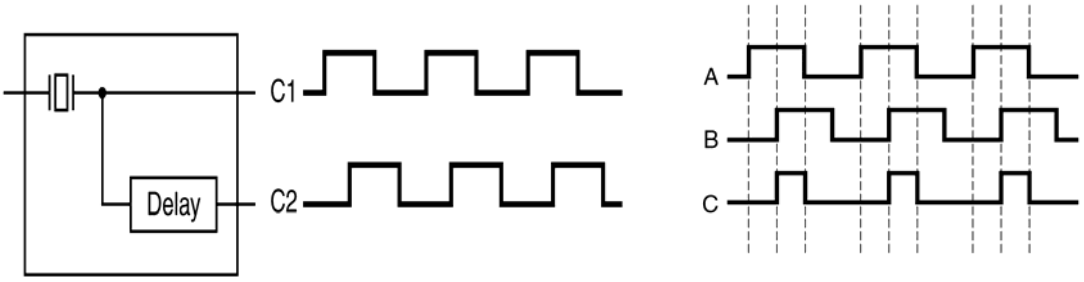
$$\begin{cases} o_i = f_i(i_1,\ldots,i_n,s_1,\ldots,s_r) & i=1,\ldots,m \\ s'_i = g_i(i_1,\ldots,i_n,s_1,\ldots,s_r) & j=1,\ldots,r \end{cases}$$

Circuiti sequenziali (continua)

$$\begin{cases} o_i = f_i(i_1,\ldots,i_n, s_1,\ldots,s_r) & i=1,\ldots,m \\ s'_i = g_i(i_1,\ldots,i_n, s_1,\ldots,s_r) & j=1,\ldots,r \end{cases}$$

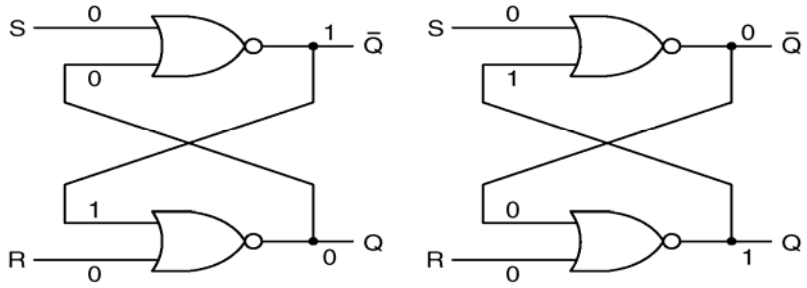
- Le uscite o_i del circuito dipendono dagli ingressi attuali e dalla *storia passata*
- La storia passata è riassunta nello *stato* del circuito
- Lo stato è codificato tramite variabili di stato booleane $s_1,\ldots,s_r$ (r variabili di stato codificano 2^r stati)
- Realizzazione del circuito:
  - Le variabili di stato s_i sono memorizzate in elementi di memoria binari
  - Circuiti combinatori calcolano le uscite e il nuovo valore dello stato

Clock



- I cambiamenti di stato del circuito vengono sincronizzati da un segnale (*clock*)
- Da un *clock primario* ne possono essere ricavati altri per sfasatura, sottrazione ecc.
- Le transizioni di stato del circuito possono avvenire:
  - **A)** In corrispondenza dei *livelli*
  - **B)** In corrispondenza dei *fronti*

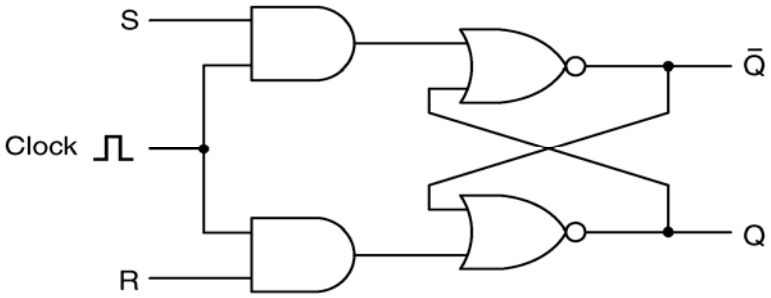
Latch RS



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

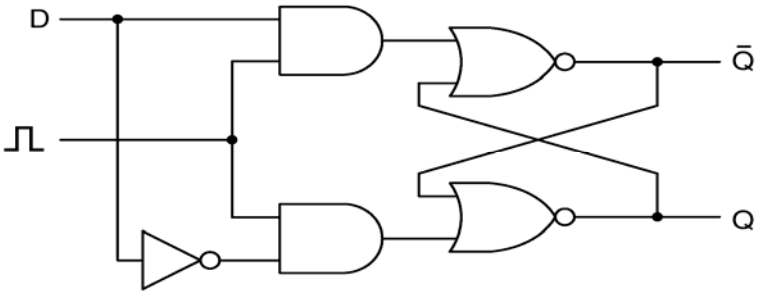
- Dispositivo di memoria elementare
- Due stati stabili **Q=0** e **Q=1**
  - **S** (**SET**): forza **Q** a **1**
  - **R** (**RESET**): forza **Q** a **0**
- Con **S=R=0** il circuito *mantiene lo stato*
- Il circuito commuta sui livelli cioè quando **S** o **R** passano a **1**
- **S** ed **R** non devono mai andare insieme ad **1**

Latch con clock



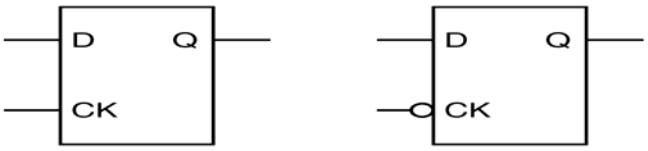
- I segnali **R** ed **S** vengono trasferiti sugli ingressi del latch solo quando il clock è ad **1**
- Quando il clock è a **0** gi segnali **R** ed **S** vengono ignorati
- Il filtro del clock consente di non influenzare il latch con i transitori dei circuiti che calcolano i segnali **R** ed **S**

Latch D (*Delay*)



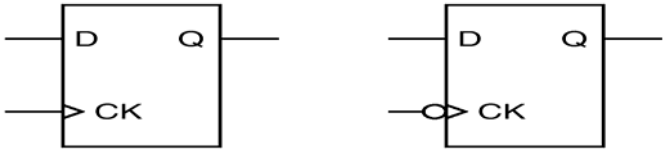
- C'è un solo ingresso **D** (i segnali **R** e **S** sono sempre uno il negato dell'altro)
- Quando il clock va ad **1** il latch registra nello stato **Q** il valore dell'ingresso **D** (da cui il nome *Delay*, cioè ritardo)
- Evita il verificarsi della situazione scorretta **R=S=1**
- Semplifica la realizzazione dei circuiti, perché è necessario un solo segnale di eccitazione (**D**)

Latch e Flip-Flop



a) **b)**

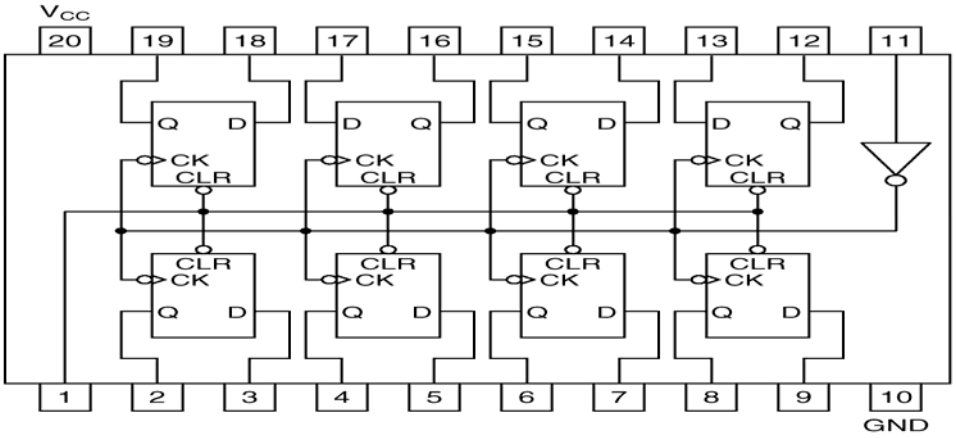
- I **Latch** commutano sui *livelli* del clock (**a)** alto, **b)** basso)



c) **d)**

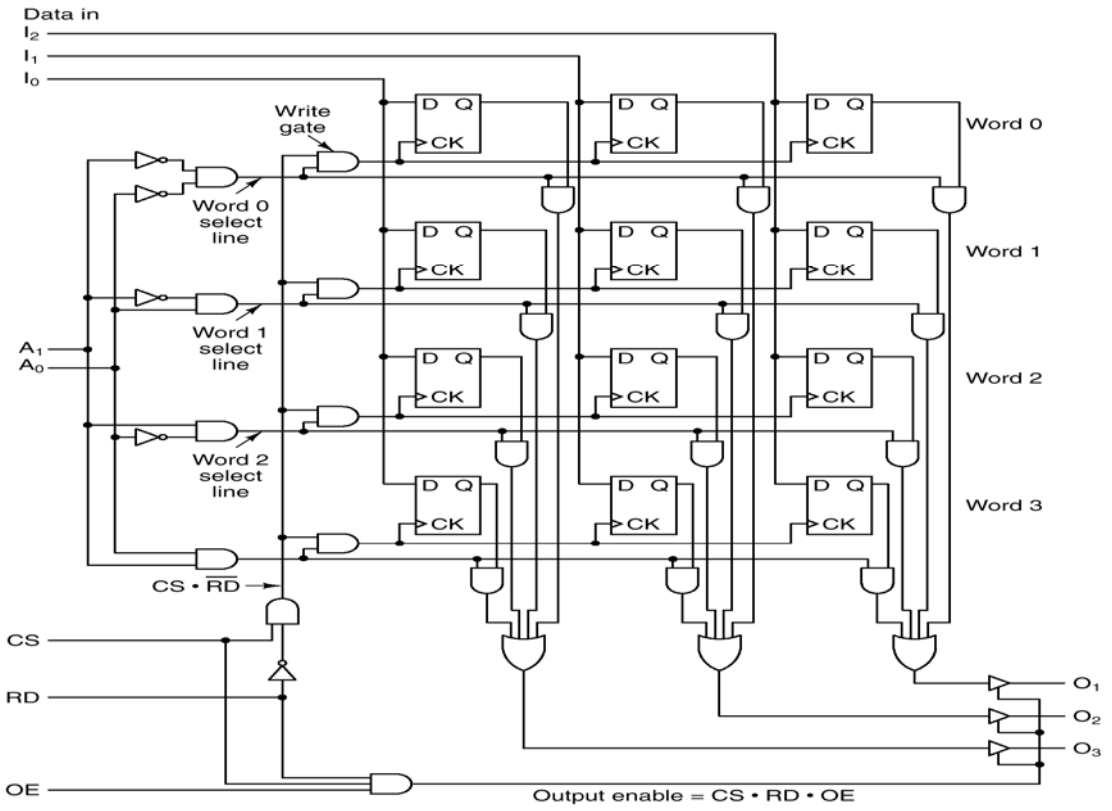
- I **Flip-Flop** commutano sui *fronti* del clock:
  - **c)** Commuta sul fronte di salita
  - **d)** Commuta sul fronte di discesa

Registri



- I Flip-Flop sono gli elementi base di memorizzazione
- I registri sono insiemi di Flip-Flop: memorizzano parole di n bit
- Molti Flip-Flop possono essere messi su un unico chip
- Occorrono in genere da 6 a 10 transistor per ogni Flip-Flop

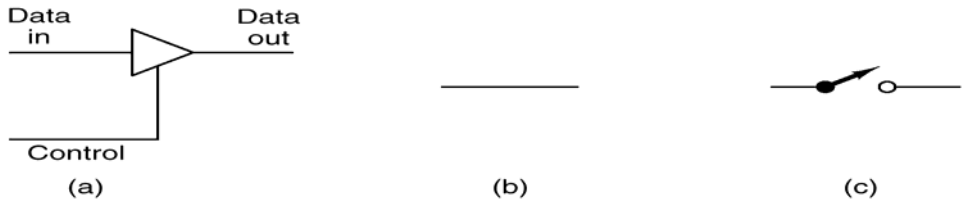
Organizzazione della memoria



Segnali di ingresso e di uscita

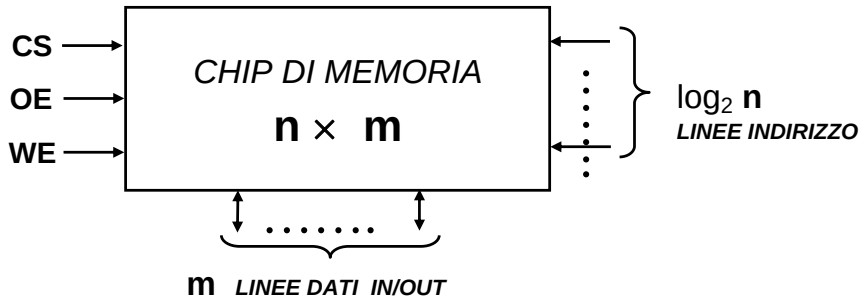
- Memoria da **4** *celle* di **3** bit ciascuna
- I flip-flop di ogni riga rappresentano una *cella*
- Segnali di ingresso e di uscita:
  - **A_0**, **A_1** indirizzo della parola letta o scritta; selezionano una parola, *abilitando* riga di flip-flop
  - **I_0**, **I_1**, **I_2** dati da immagazzinare in scrittura
  - **RD=1**: lettura; **RD=0** scrittura
  - **CS** (chip select) trigger per la lettura e la scrittura; in **AND** con **RD=0** e con il segnale di abilitazione sulla riga generano il fronte del clock che causa la scrittura **I_0**, **I_1**, **I_2** nei flip-flop
  - **OE** abilita il trasferimento dei dati in uscita, attivando i dispositivi a tre stati
  - **O_1**, **O_2**, **O_3** uscite di lettura, abilitate dai dispositivi a tre stati

Dispositivi a 3 stati



- In base ad un segnale di controllo **C** si comporta:
  - (b) **C=1**: come *circuito chiuso*
  - (c) **C=0**: come *circuito aperto*
- Tempo di commutazione: pochi nsec
- Consente di usare gli stessi piedini sia per la lettura che per scrittura
- Usato anche per la connessione ai bus e a qualsiasi linea bidirezionale

Chip di memoria



- Chip da **n** × **m** bit complessivi
- **m** linee dati bidirezionali
- **$\log_2$ n** linee di indirizzo
- *Segnali di controllo*:
  - **CS:** *Chip Select*, seleziona il chip interessato
  - **OE** *Output Enable*, abilita la lettura delle uscite
  - **WE** *Write Enable*, specifica tra lettura e scrittura

Chip di memoria: numero di piedini

- Occorre un grande numero di piedini nel chip
  - **m** linee dati (per parole di **m** bit)
  - **$\log_2$ n** linee di indirizzo per **n** parole di memoria
  - segnali di controllo e alimentazione

ESEMPIO *Chip da 256 MByte*
- **m=8** linee dati
- **$\log_2$ 256 M= 28** linee indirizzo
- 3+2 linee di controllo e alimentazione
- totale 41 piedini: troppi

- Soluzione
  - **m=1:** memorie bit slice
  - Decodifica a matrice: riduce linee di indirizzo

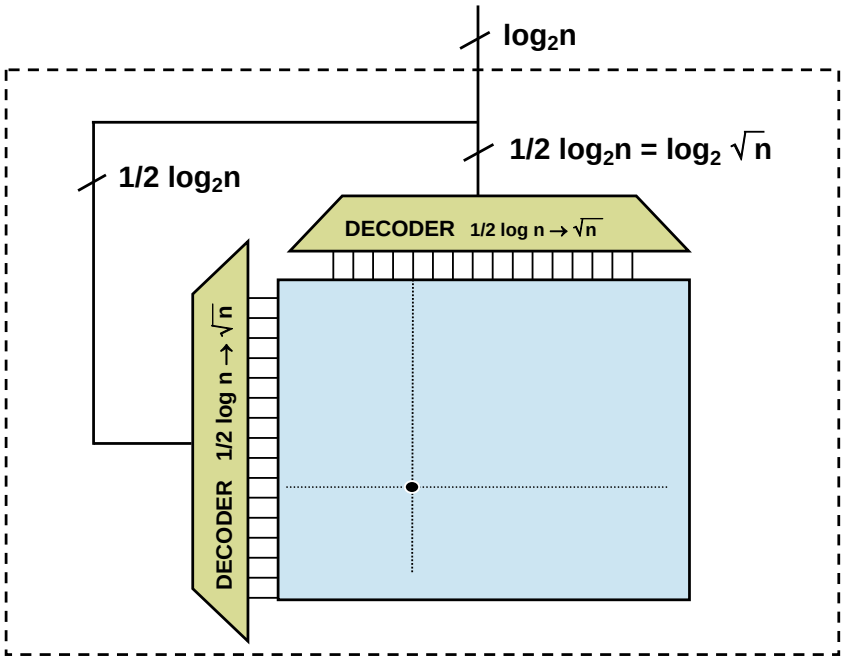
Matrice bidimensionale di selezione

- Le celle di memoria elementari sono organizzate secondo un *array bidimensionale*
- Ciascuna cella è individuata da un *indirizzo di riga* e da un *indirizzo di colonna*
- Per 2^m celle occorrono $2^{m/2}$ righe e $2^{m/2}$ colonne
- Indirizzi di riga e colonna inviati in sequenza
- Si dimezzano i piedini e si risparmia nella complessità della logica di decodifica
- Un decoder $\mathbf{m \rightarrow 2^m}$ richiede $\mathbf{2^m}$ porte **AND**

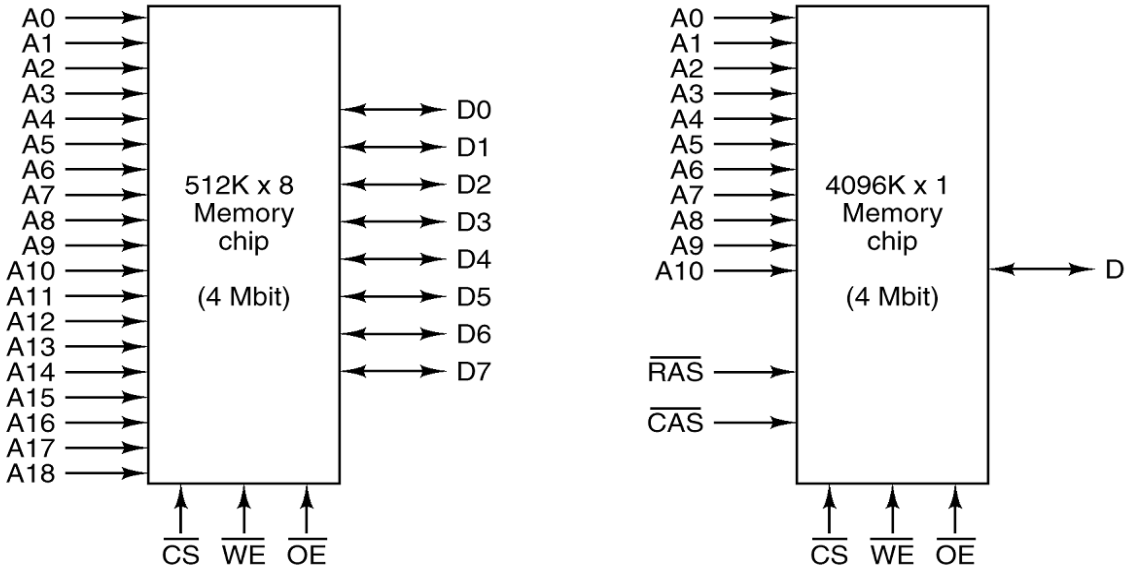
ESEMPIO

– 4M parole (m=22) $\rightarrow$ 22 linee di indirizzo
– 1 decoder a 22 $\rightarrow$ 4M porte AND
– 2 decoder a 11 $\rightarrow 2 \cdot 2^{11} =$ 4k porte AND

Matrice di selezione

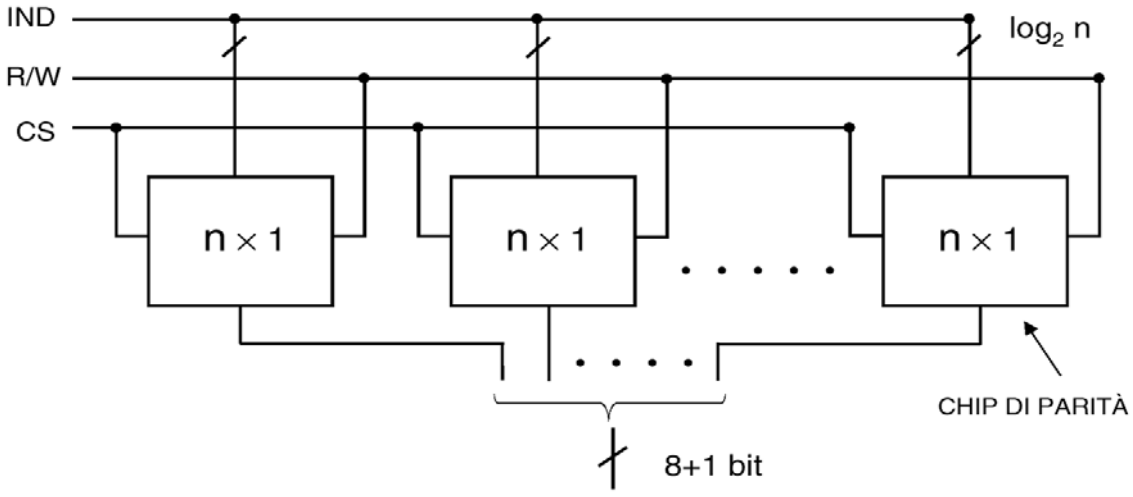


Chip di memoria: esempi



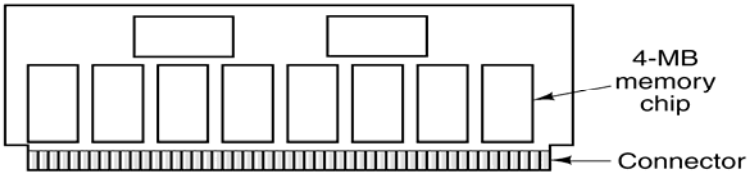
- **RAS** (*Row Address Strobe*), **CAS** (*Column Address Strobe*)
- Indirizzi di riga e di colonna *multiplexati* sugli stessi piedini

Schede di memoria



- La scheda memorizza **n** parole di 8+1 bit
- Si usano **n+1** chip se si vuole il controllo di parità
- Bit di parità gestito dal controller della memoria

Schede SIMM e DIMM



- **SIMM** (*Single Inline Memory Module*) ormai in disuso
  - 72 piedini, 32 bit, 8-16 chip, massimo 32 MByte
- **DIMM** (*Double Inline Memory Module*)
  - 168 piedini, 64 bit, 16 chip, 512 MByte
- Il *controller* gestisce più SIMM (o DIMM)
- Ogni SIMM informa il controller della sua dimensione (segnali su certi piedini)
- Il controller determina al momento del boot il tipo di RAM
- Dall'indirizzo e dalla configurazione il controller calcola a quale SIMM mandare il segnale di *Chip Select*

Tassonomia delle memorie

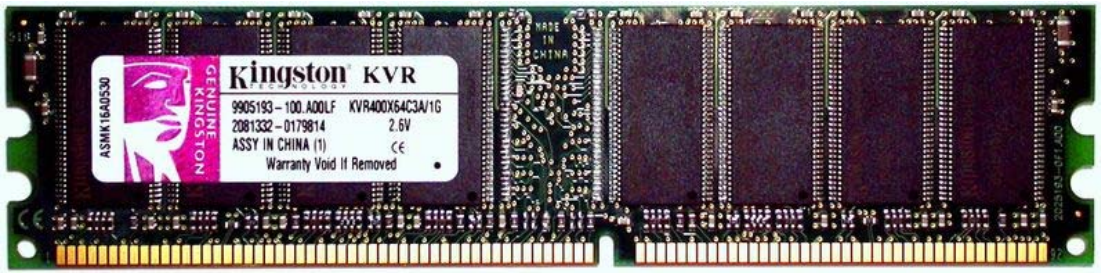
- **RAM** (*Random Access Memory*)
- **SRAM** (*Static RAM*): a Flip-Flop, molto veloce (~5nsec)
- **DRAM** (*Dynamic RAM*): basata su capacità parassite; richiede *refresh*, alta densità, basso costo (~70 nsec)
  - **FPM**: selezione a matrice
  - **EDO**: (*Extended Data Output*) lettura in *pipeline*, più banda
- **SDRAM** (*Synchronous DRAM*): sincrona, prestazioni migliori
- **DDRAM** (*Double Data DRAM*): lettura su entrambi i fronti del clock
- **ROM** (*Read Only Memory*)
- **PROM** (*Programmable ROM*)
- **EPROM** (*Erasable PROM*) raggi UV
- **EEPROM**: cancellabile elettricamente
- **Flash Memory**: particolare tipo di EEPROM, ciclo 100nsec, max 10.000 riscritture

Tipi di memorie e loro impieghi

Type	Category	Erasure	Byte alterable	Volatile	Typical use
SRAM	Read/write	Electrical	Yes	Yes	Level 2 cache
DRAM	Read/write	Electrical	Yes	Yes	Main memory
ROM	Read-only	Not possible	No	No	Large volume appliances
PROM	Read-only	Not possible	No	No	Small volume equipment
EPROM	Read-mostly	UV light	No	No	Device prototyping
EEPROM	Read-mostly	Electrical	Yes	No	Device prototyping
Flash	Read/write	Electrical	No	No	Film for digital camera

Solo le memorie ROM, PROM, EPROM, EEPROM e Flash mantengono il loro contenuto in assenza di alimentazione: le altre (SRAM, DRAM etc.) sono volatili

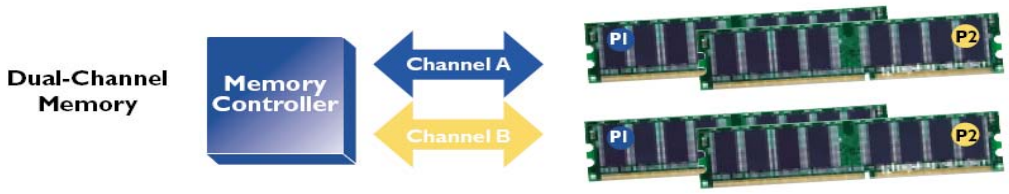
Memorie SDRAM e DDR



- Le memorie SDRAM sono sincrone cioè comandate dal clock
- Il clock elimina la necessità di alcuni segnali di sincronismo
- Le più moderne SDRAM sono anche DDR (Double Data Rate)
- Le DDR raddoppiano la banda trasferendo dati sia sul fronte di salita che su quello di discesa del clock
- Per una frequenza di 100 MHz a gruppi di 8 byte:

Banda di picco = 100 MHz x 2 x 8 byte = 1.6 GB/s

Tecnologia Dual Channel



PEAK BANDWIDTH	DATA BITS ACCESSED	PC-133	PC2100 DDR266	PC2700 DDR333	PC3200 DDR400
Single-Channel	64	1.1GB/s	2.1GB/s	2.7GB/s	3.2GB/s
Dual-Channel	128		4.2GB/s	5.4GB/s	6.4GB/s

- Ciascun banco è composto da una coppia di DIMM che lavorano in parallelo
- Banda di picco DDR400 (Clock 200 MHz):

Banda = 200 MHz x 2 x 8 byte x 2 canali = 6400 MB/s = 6.4 GB/s