



# Circuiti combinatori notevoli

Prof. Alberto Borghese  
Dipartimento di Informatica  
[alberto.borghese@unimi.it](mailto:alberto.borghese@unimi.it)

Università degli Studi di Milano

Riferimenti sul Patterson Hennessy: Sezione B3.



## Sommario

**Implementazione circuitale mediante PLA o ROM**

Circuiti combinatori notevoli



## Circuiti combinatori



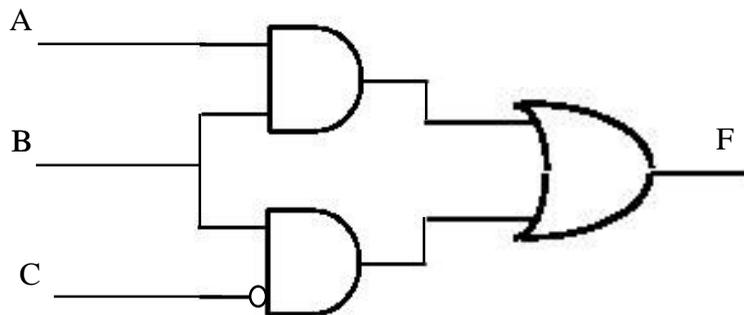
- Circuiti logici digitali in cui le operazioni (logiche) dipendono solo da una combinazione degli input.
- Circuiti senza memoria. Ogni volta che si inseriscono in ingresso gli stessi valori, si ottengono le stesse uscite. Il risultato non dipende dallo stato del circuito.
- I circuiti combinatori descrivono delle funzioni Booleane. Queste funzioni si ottengono combinando tra loro (in parallelo o in cascata) gli operatori logici: **NOT, AND, OR**.
- Il loro funzionamento può essere descritto come **tabella della verità**.
- Come nelle funzioni algebriche, il risultato è aggiornato immediatamente dopo il cambiamento dell'input (si suppone il tempo di commutazione trascurabile, tempo di attesa prima di guardare l'output sufficientemente ampio per permettere a tutti i circuiti la commutazione).



## Funzione come circuito



$$F = A B + B \bar{C}$$



Le operazioni algebriche hanno un ordine di precedenza.



## Razionale della prima forma canonica



$$F = A B + B \bar{C}$$

A B C	F
0 0 0	0
0 0 1	0
0 1 0	1
0 1 1	0
1 0 0	0
1 0 1	0
1 1 0	1
1 1 1	1

$$F = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + ABC$$

$$F = 1$$

iff

$$A = 0 \ B = 1 \ C = 0$$

OR

$$A = 1 \ B = 1 \ C = 0$$

OR

$$A = 1 \ B = 1 \ C = 1$$



## La prima forma canonica



- Esiste un metodo per ricavare automaticamente un circuito che implementi una tabella di verità?
- Una **forma canonica** garantisce di poter realizzare una qualunque tabella di verità con solo due livelli di porte OR, AND e NOT:
- Somme di Prodotti (SOP) è la prima forma canonica (OR di AND):

A B C	F
0 0 0	0
0 0 1	0
0 1 0	1 $m_2$
0 1 1	0
1 0 0	0
1 0 1	0
1 1 0	1 $m_6$
1 1 1	1 $m_7$

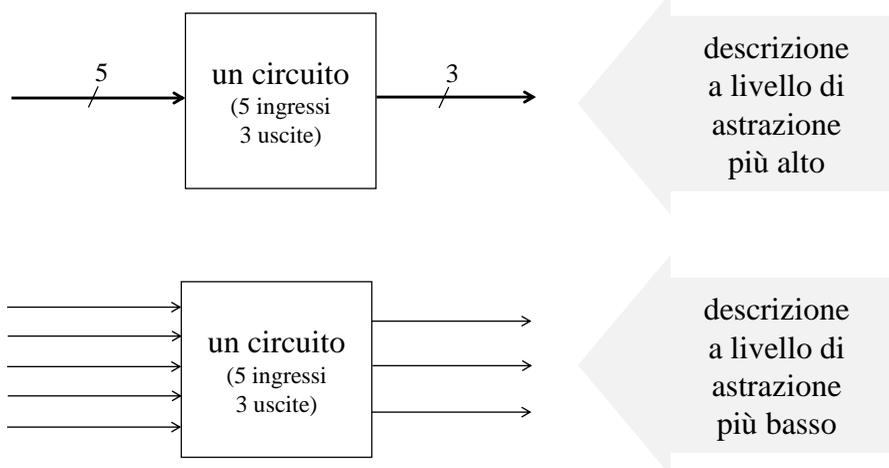
$$F = m_2 + m_6 + m_7$$

mintermini

$$F = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + ABC$$



## Livelli di astrazione intermedi: anche per le linee



## Tipi di circuiti che implementano le SOP



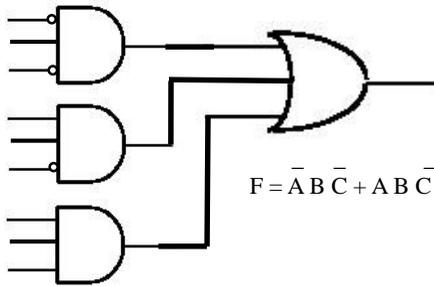
In generale abbiamo funzioni logiche booleane multi-input / multi-output.

**I circuiti complessi si ottengono collegando «piccoli» blocchi funzionali, dove ciascun blocco implementa una funzione logica.**

- **Logica distribuita.**
- **PLA:** Programmable Logic Array: matrici regolari AND e OR in successione, personalizzabili dall'utente.
- **ROM:** Read Only Memory circuiti ad hoc che implementano una particolare funzione in modo irreversibile.



## Logica distribuita



CO = 8

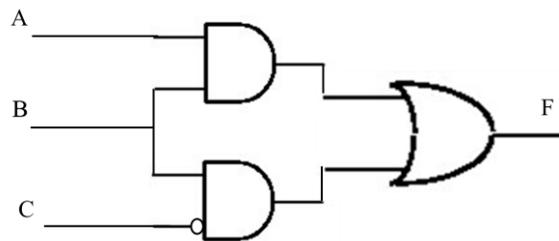
CC = 4

$$F = \bar{A} \bar{B} \bar{C} + A \bar{B} \bar{C} + A B C = m_2 + m_6 + m_7$$

$$F = A B + B \bar{C}$$

CO = 3

CC = 2



A.A. 2023-2024

9/47

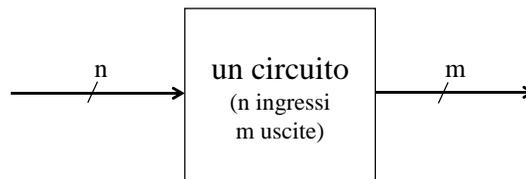
<http://borghese.di.unimi.it/>



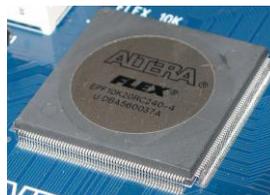
## PLA (Programmable Logical Array)



- Architettura a 2 livelli:
  - Un primo livello di AND
  - Un secondo livello di OR



- La matrice degli AND ha n linee di ingresso: ciascuna porta ha in ingresso **n linee** e il loro complemento.
- L'utente specifica per ogni porta AND a disposizione se la linea in ingresso entra direttamente o dopo una negazione.
  - Crea la matrice dei mintermini, bruciando in ingresso alle porte AND le linee che non servono.
- Le uscite della matrice AND entrano nella matrice OR programmata come la precedente per ottenere l'OR dei mintermini della funzione.
  - Si utilizza una porta OR per ogni funzione calcolata (m OR per **m linee** di uscita)

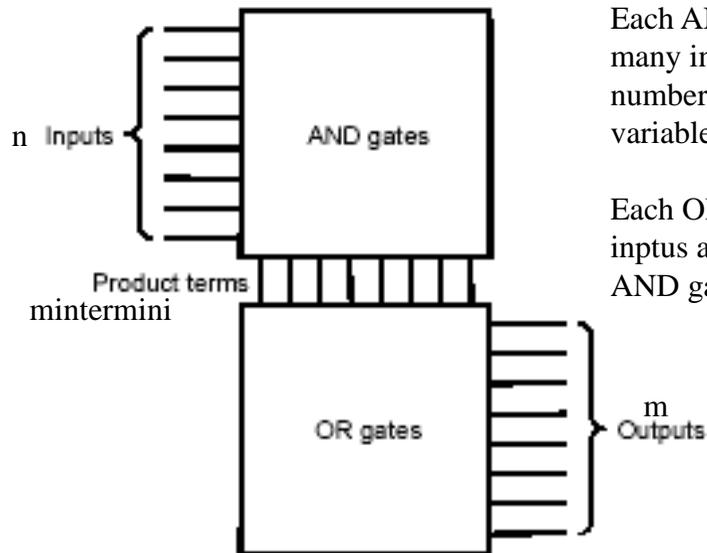


A.A. 2023-2024

<http://borghese.di.unimi.it/>



## Struttura di una PLA



Each AND has as many inputs as the number of input variables

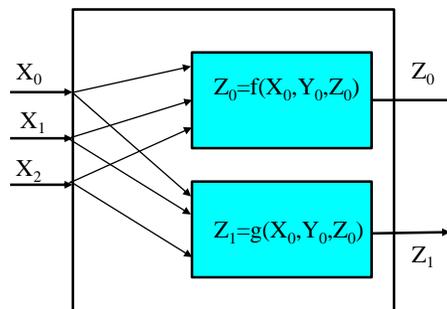
Each OR has as many inputs as the number of AND gates



## Sintesi separata delle uscite



$X_0$	$X_1$	$X_2$	$Z_0$	$Z_1$
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



$$Z_0 = \overline{X_0} \overline{X_1} X_2 + \overline{X_0} X_1 \overline{X_2} + X_0 \overline{X_1} \overline{X_2} + X_0 X_1 X_2 = m_1 + m_2 + m_4 + m_7$$

$$Z_1 = \overline{X_0} X_1 X_2 + X_0 \overline{X_1} X_2 + X_0 X_1 \overline{X_2} + X_0 X_1 X_2 = m_3 + m_5 + m_6 + m_7$$

Non è efficiente perché non si sfruttano le operazioni in comune alle due funzioni.

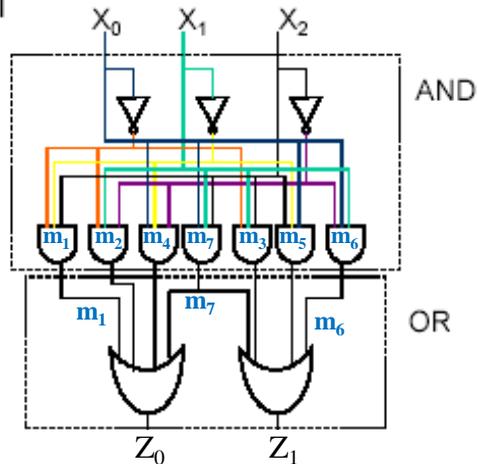


## Sintesi mediante PLA



- Realizzare con un PLA la funzione descritta dalla seguente TT

	$X_0$	$X_1$	$X_2$	$Z_0$	$Z_1$
	0	0	0	0	0
$m_1 \rightarrow$	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
$m_6 \rightarrow$	1	1	0	0	1
$m_7 \rightarrow$	1	1	1	1	1



$$Z_0 = \overline{X_0} \overline{X_1} X_2 + \overline{X_0} X_1 \overline{X_2} + X_0 \overline{X_1} X_2 + X_0 X_1 \overline{X_2} = m_1 + m_2 + m_4 + m_7$$

$$Z_1 = \overline{X_0} X_1 X_2 + X_0 \overline{X_1} X_2 + X_0 X_1 \overline{X_2} + X_0 X_1 X_2 = m_3 + m_5 + m_6 + m_7$$

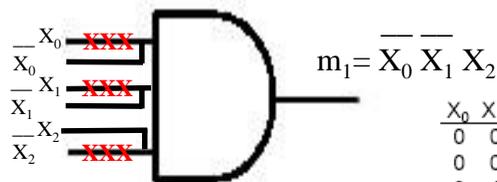
ii.it\



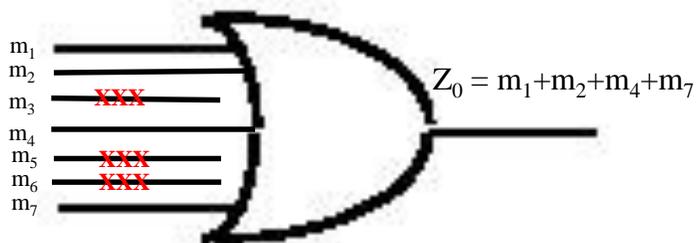
## Configurazione di una PLA



Costruzione di una porta AND ( $m_1$ )



$X_0$	$X_1$	$X_2$	$Z_0$	$Z_1$
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

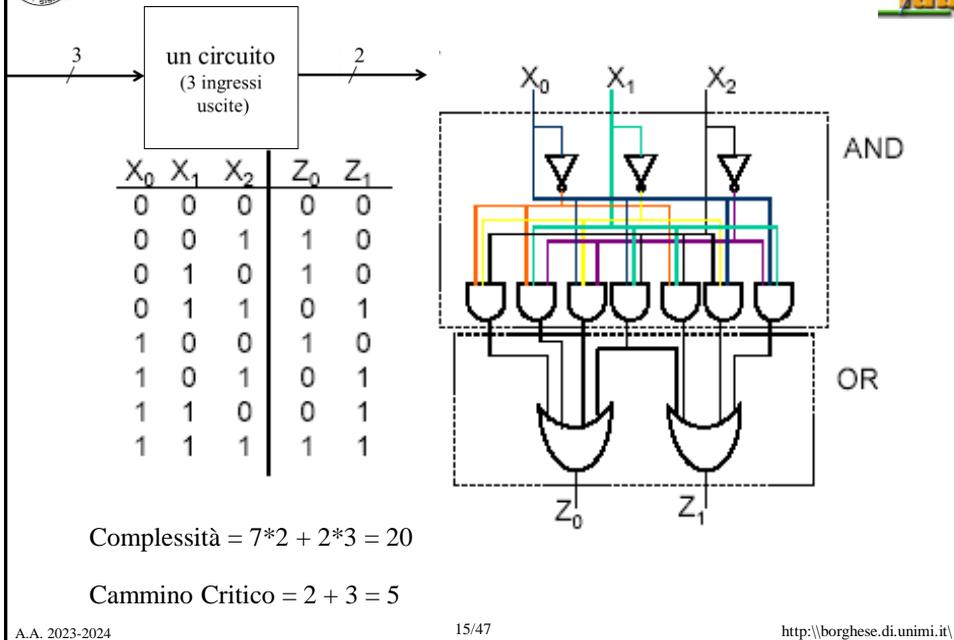


A.A. 202

<http://borghese.di.unimi.it/>



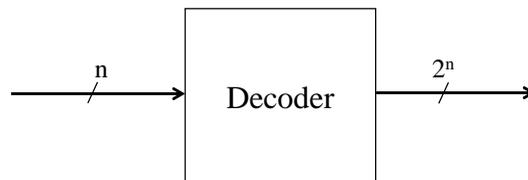
## Complessità e cammino critico



## Decodificatore (decoder)



- E' caratterizzato da  $n$  linee di input e  $2^n$  linee di output



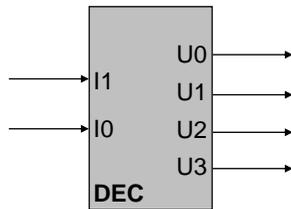
- il numero binario espresso dalla configurazione delle linee di input è usato per asserire la **sola** linea di output di ugual indice (tutte le altre portano 0)
- es.: con 4 linee di input e 16 di output (da 0 a 15), se in ingresso arriva il valore 0110 (6 decimale), in uscita si alza la linea di indice 5 (la sesta!), tutte le altre hanno valore 0
- utilizzato per indirizzare la memoria ad esempio (cf. ROM)



# Decoder a 2 ingressi



Tabella della verità



I1	I0	U0	U1	U2	U3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



# Decoder a 2 ingressi: realizzazione



I1	I0	U0	U1	U2	U3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

$$U_i = f_i(I_0, I_1)$$

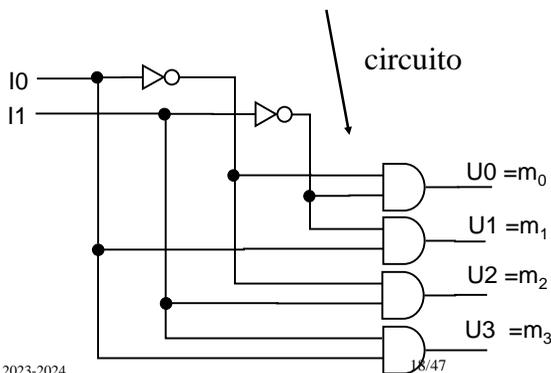
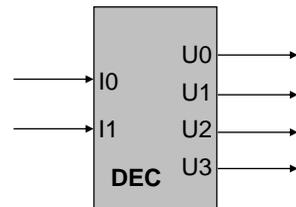
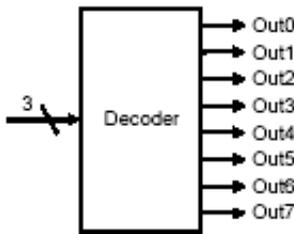


Tabella della verità

I1	I0	U0	U1	U2	U3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



## Decoder a 3 ingressi



a. A 3-bit decoder

A	B	C	U <sub>0</sub>	U <sub>1</sub>	U <sub>2</sub>	U <sub>3</sub>	U <sub>4</sub>	U <sub>5</sub>	U <sub>6</sub>	U <sub>7</sub>
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Le funzioni di uscita sono  $2^n$  per  $n$  input:

$$U_0 = \sim A \sim B \sim C$$

...

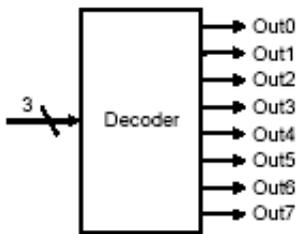
$$U_7 = A B C$$

$$U_j = m_j$$

I mintermini sono AND a 3 ingressi



## Valutazione del decoder a 3 ingressi



a. A 3-bit decoder

A	B	C	U <sub>0</sub>	U <sub>1</sub>	U <sub>2</sub>	U <sub>3</sub>	U <sub>4</sub>	U <sub>5</sub>	U <sub>6</sub>	U <sub>7</sub>
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Le funzioni di uscita sono  $2^n$  per  $n$  input:

$$U_0 = \sim A \sim B \sim C$$

...

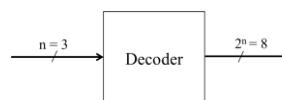
$$U_7 = A B C$$

$$U_j = m_j$$

I mintermini sono AND a 3 ingressi

Complessità:  $8 \cdot 2 = 16$

Cammino critico: 2





# Rappresentazione circuitale mediante ROM



- Read Only Memory, memoria di sola lettura.  
Funge anche da modulo combinatorio a uscita multipla.
- n linee di ingresso, m linee di uscita (ampiezza)  
a ciascuna delle  $2^n$  (altezza) configurazioni di ingresso (parole di memoria) è associata permanentemente una combinazione delle m linee di uscita.
- l'input seleziona la parola da leggere di m bit, che appare in uscita
- L'input funziona da **indice** all'interno della ROM.  
Viene realizzato con un decoder n-a- $2^n$  seguito da una matrice di m porte OR.

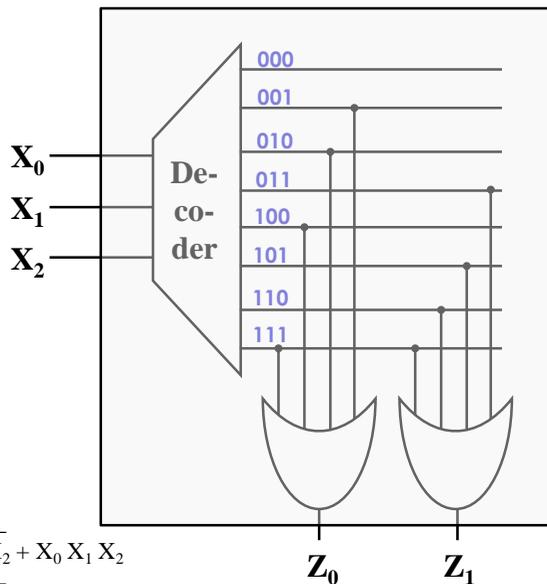
$X_0$	$X_1$	$X_2$	$Z_0$	$Z_1$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



## ROM - esempio



	$X_0$	$X_1$	$X_2$	$Z_0$	$Z_1$
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1



$$Z_0 = \overline{X_0} \overline{X_1} X_2 + \overline{X_0} X_1 \overline{X_2} + X_0 \overline{X_1} \overline{X_2} + X_0 X_1 X_2$$

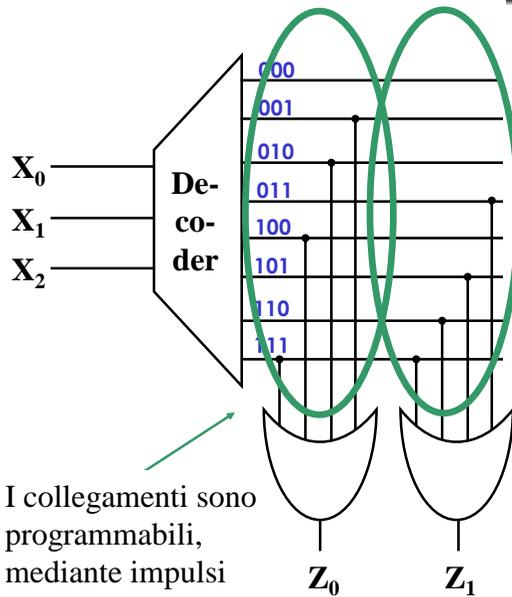
$$Z_1 = \overline{X_0} X_1 X_2 + X_0 \overline{X_1} X_2 + X_0 X_1 \overline{X_2} + X_0 X_1 X_2$$



# EEPROM



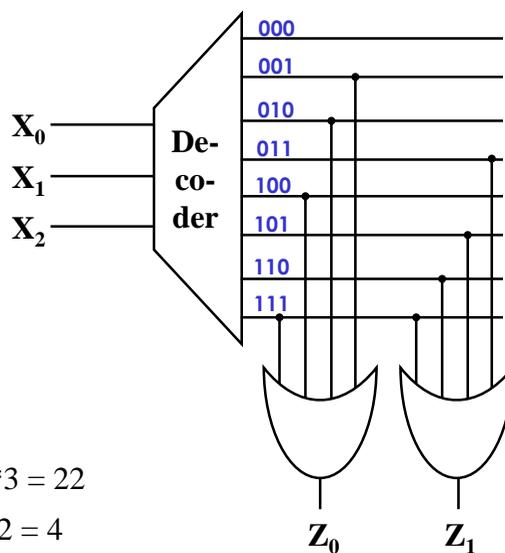
X <sub>0</sub>	X <sub>1</sub>	X <sub>2</sub>	Z <sub>0</sub>	Z <sub>1</sub>
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



# Valutazione ROM



X <sub>0</sub>	X <sub>1</sub>	X <sub>2</sub>	Z <sub>0</sub>	Z <sub>1</sub>
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

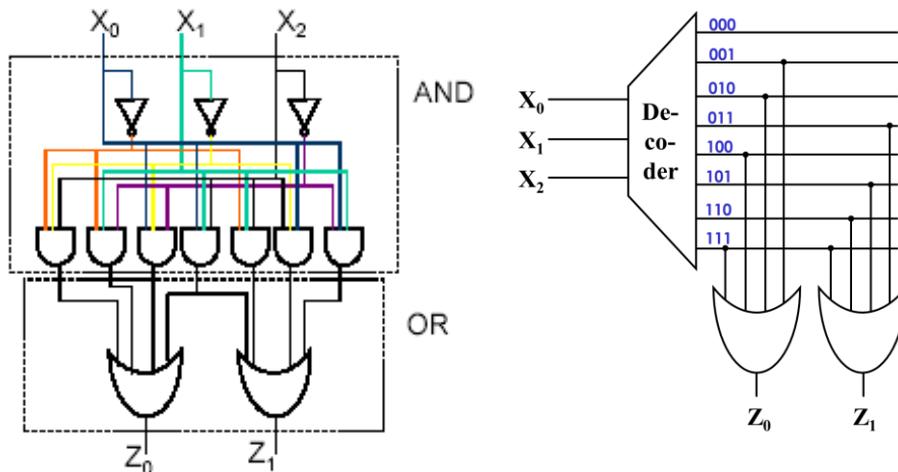


Complessità:  $8 \cdot 2 + 2 \cdot 3 = 22$

Cammino critico:  $2 + 2 = 4$



## Confronto PLA / ROM



## Confronto PLA - ROM



ROM – fornisce un'uscita per ognuna delle combinazioni degli ingressi (mintermini e maxtermini). Decoder con  $2^n$  uscite, dove  $n$  è il numero di variabili in ingresso alla ROM. Crescita esponenziale delle uscite. Approccio più generale. Può implementare una qualsiasi funzione, dato un certo numero di input e output.

PLA – contiene solamente i mintermini in uscita al primo livello.  
Il loro numero cresce meno che esponenzialmente.

E' più veloce una PLA o una ROM? Valutare in termini di cammino critico.

FPGA – Maggiore libertà. E' costituita da celle: moduli di PLA. E' una rete di strutture a 2 livelli.



## Esercizi sulla PLA



Realizzare mediante PLA con 3 ingressi con il numero adeguato di linee interne:

- la funzione maggioranza.
- la funzione che vale 1 se e solo se 1 solo bit di ingresso vale 1
- un decoder
- la funzione che vale 0 se l'input è pari, 1 se dispari
- la funzione che calcola i multipli di 3 (con 4 ingressi)

Realizzare le stesse funzioni con una ROM o con logica distribuita e operare un confronto. Fare una valutazione comparata in termini di complessità e cammino critico.



## Sommario



Implementazione circuitale mediante PLA o ROM.

**Circuiti combinatori notevoli.**



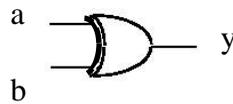
# Porta XOR



a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

$$\text{SOP: } y = \bar{a}b + a\bar{b}$$

OR esclusivo  
Disuguaglianza di 2 ingressi



Complessità e cammino critico di 1 porta logica

$$y = a \oplus b$$



# Uscite indifferenti di un tabella delle verità



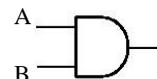
X = "don't care" = {0,1} a seconda di quello che conviene.

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

Ho 2 possibilità:

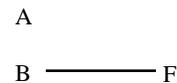
1)  $X = 0$

$$F = A B$$



2)  $X = 1$

$$F = \bar{A} B + A B = B$$



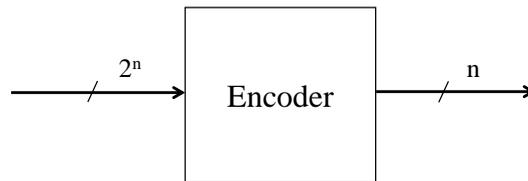
Diminuisce il numero di porte e si accorcia il cammino critico.



## Codificatore (encoder)



- E' caratterizzato da n linee di input e  $\text{ceil}(\log_2 n)$  linee di output



- Una sola linea di ingresso può essere attiva.
- il numero binario espresso dalla configurazione delle linee di output rappresenta la linea di ingresso attiva.
- es.: con 16 linee di input e 4 di output, se in ingresso arriva il valore 0000 0100 0000 0000, in uscita leggiamo il numero 10.



## La funzione encoder



ABCD	$Y_1 Y_2$
0000	X X
0001	0 0
0010	0 1
0011	X X
0100	1 0
0101	X X
0110	X X
0111	X X
1000	1 1
1001	X X
1010	X X
1011	X X
1100	X X
1101	X X
1110	X X
1111	X X

Codifica quattro linee in ingresso: 0,1,2,3

Suppongo  $X = 0$

$$Y_1 = \overline{A}BCD + A\overline{B}CD = CD (A \oplus B)$$

$$m_4 + m_8$$

$$Y_2 = \overline{A}BC\overline{D} + A\overline{B}C\overline{D} = \overline{B}C\overline{D} (A \oplus C)$$

$$m_2 + m_8$$

Complessità:  $2 \cdot 3 + 1 = 7$

Complessità circuito semplificato: 3

Cammino critico:  $2 + 1 = 3$

Cammino critico circuito semplificato: 2

Si può fare di meglio?

Come è conveniente impostare le X?



## La funzione encoder: ottimizzazione - I



ABCD	$Y_1 Y_2$
0000	0 X
0001	0 0
0010	0 1
0011	0 X
0100	1 0
0101	1 X
0110	1 X
0111	1 X
1000	1 1
1001	1 X
1010	1 X
1011	1 X
1100	0 X
1101	0 X
1110	0 X
1111	0 X

Consideriamo la prima funzione,  $Y_1$

Specifichiamo in modo opportuno i valori di uscita indifferenti, X

$$Y_1 = \bar{A} \bar{B} \bar{C} \bar{D} + \bar{A} \bar{B} C \bar{D} + \bar{A} B \bar{C} \bar{D} + \bar{A} B C \bar{D} + \bar{A} B C D + A \bar{B} \bar{C} \bar{D} + A \bar{B} C \bar{D} + A \bar{B} C D + A B \bar{C} \bar{D} + A B C \bar{D} + A B C D = (A \oplus B)$$

Complessità: 1

Cammino critico: 1



## La funzione encoder: ottimizzazione - II



ABCD	$Y_1 Y_2$
0000	0 0
0001	0 0
0010	0 1
0011	0 1
0100	1 0
0101	1 0
0110	1 1
0111	1 1
1000	1 1
1001	1 1
1010	1 0
1011	1 0
1100	0 1
1101	0 1
1110	0 0
1111	0 0

Consideriamo la prima funzione,  $Y_1$

Specifichiamo in modo opportuno i valori di uscita indifferenti, X

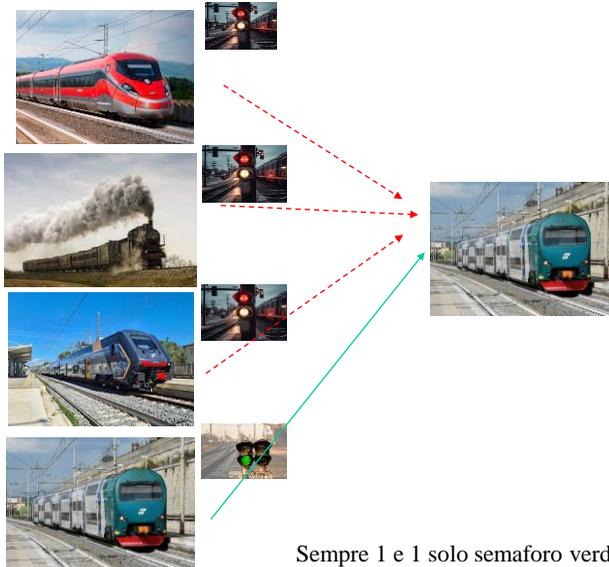
$$Y_2 = \bar{A} \bar{B} \bar{C} \bar{D} + \bar{A} \bar{B} C \bar{D} + \bar{A} B \bar{C} \bar{D} + \bar{A} B C \bar{D} + \bar{A} B C D + A \bar{B} \bar{C} \bar{D} + A \bar{B} C \bar{D} + A \bar{B} C D + A B \bar{C} \bar{D} + A B C \bar{D} + A B C D = (A \oplus C)$$

Complessità: 1

Cammino critico: 1



## Multiplexer



## Multiplexer (selettore)

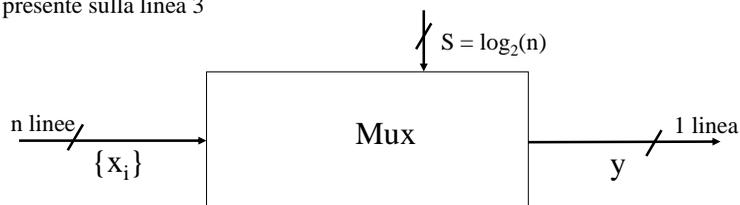


- Ha la funzione di un sistema di semafori.
- E' caratterizzato da:
  - n linee di input (data) -  $\{x_i\}$
  - k linee di controllo (selezione) -  $\{S\}$ .
  - 1 linea di output.
- In base alla linea di controllo viene connessa all'uscita la linea di ingresso selezionata.

Quante linee di controllo, k, servono?

$$k = \text{ceil}(\log_2 n)$$

Esempio: con 4 linee di input (da 0 a 3), se sulle linee di controllo c'è 11, in uscita si avrà il valore presente sulla linea 3



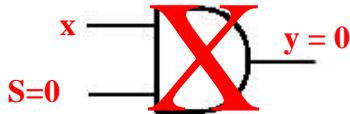
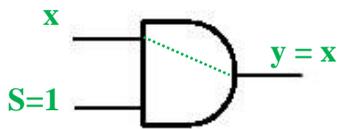


## AND come semaforo (interruttore)



Il segnale di selezione S, “apre” la porta opportuna, cioè chiude il cammino opportuno.

L'AND funziona da porta di uscita (da semaforo)



S	x	y
0	0	0
0	1	0
1	0	0
1	1	1

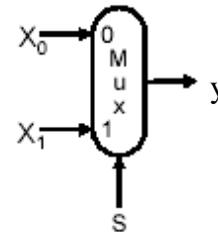
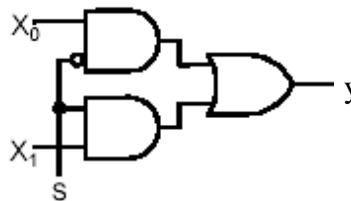


## Multiplexer



S	$x_0$	$x_1$	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Il circuito deve portare in uscita il contenuto della linea  $x_0$  o  $x_1$  a seconda del valore di S





## Sintesi della funzione Mux



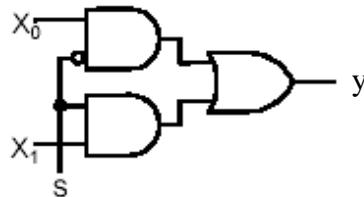
S	x <sub>0</sub>	x <sub>1</sub>	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$$\begin{aligned} \text{SOP: } y &= \bar{S} x_0 \bar{x}_1 + \bar{S} x_0 x_1 + S \bar{x}_0 x_1 + S x_0 x_1 \\ &= \bar{S} x_0 + S x_1 \quad \text{cvd} \end{aligned}$$

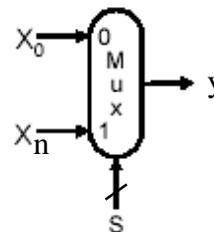
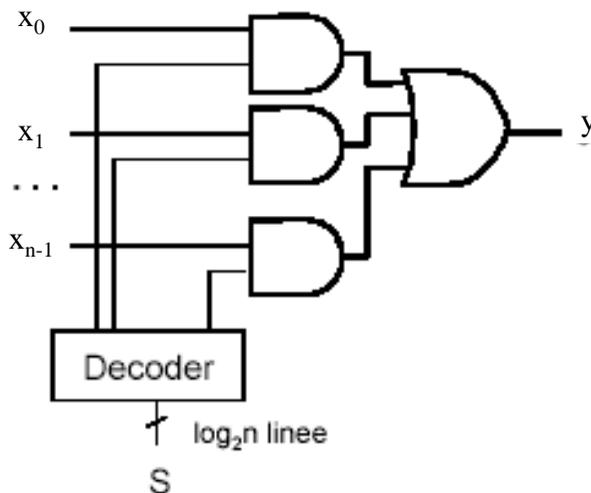
Il mux si comporta come interruttore:

If (S == 1) then  
 $y = x_1$   
 If (S == 0) then  
 $y = x_0$

Complessità: 3  
 Cammino critico: 2



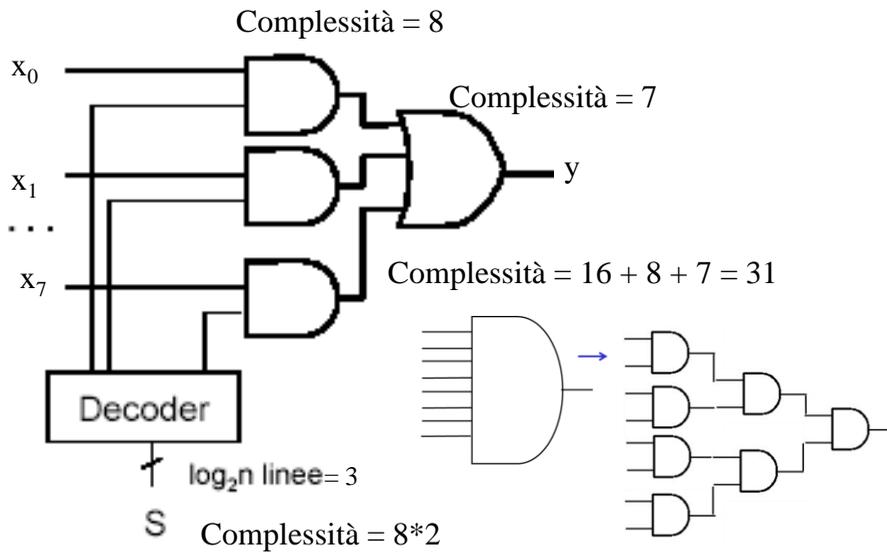
## Mux a n vie.



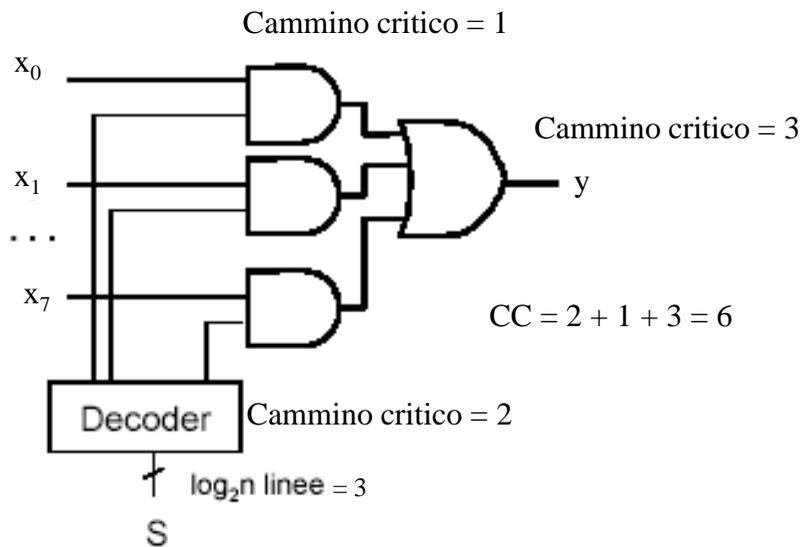
Una sola linea alla volta viene aperta dal segnale S. **Le linee sono mutuamente esclusive.**



## Complessità di un Mux a 8 vie



## Cammino critico di un Mux a 8 vie





## Il Multiplexer a un ingresso di selezione è l' "if" dell'hardware

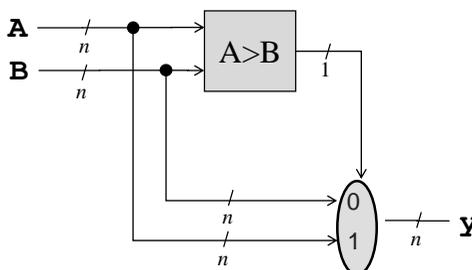


- Software

```
if A>B
  then y=A
  else y=B
```

**Esecuzione  
condizionata**

- Hardware



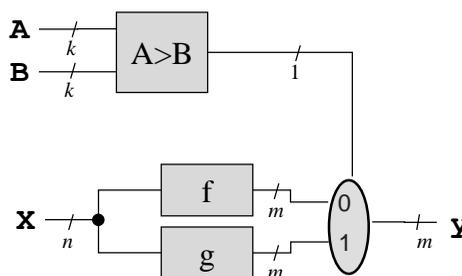
## Il Multiplexer a un ingresso di selezione è l' "if" dell'hardware



- Software

```
if A>B
  then y=g(X)
  else y=f(X)
```

- Hardware



Entrambe le funzioni sono calcolate, ma solo un risultato alla volta viene usato!



## Comparatore a 1 bit



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

XOR

$A_0$	$B_0$	$C_0$
0	0	1
0	1	0
1	0	0
1	1	1

XNOR

$$C_0 = \overline{A_0 \oplus B_0}$$



## Comparatore a n bit



- E' caratterizzato da:
  - 1 numero su n bit in ingresso, A
  - 1 numero su n bit in ingresso, B
  - 1 uscita

$$C_k = \overline{A_k \oplus B_k}$$

$A_0$	$B_0$	$C_0$	$A_1$	$B_1$	$C_1$	
0	0	1	0	0	1	
0	1	0	0	1	0	...
1	0	0	1	0	0	
1	1	1	1	1	1	

$$C = C_0 C_1 C_2 \dots C_{n-1} = (A_0 \oplus B_0) (A_1 \oplus B_1) \dots (A_{n-1} \oplus B_{n-1})$$

Attraverso De Morgan:

$$C = \overline{C_0} + \overline{C_1} + \overline{C_2} \dots + \overline{C_{n-1}} = (A_0 \oplus B_0) (A_1 \oplus B_1) \dots (A_{n-1} \oplus B_{n-1})$$



## Sommario



Implementazione circuitale mediante PLA o ROM.

Circuiti combinatori notevoli.