

---

# Architettura degli Elaboratori

## Circuiti combinatori

slide a cura di Salvatore Orlando, Andrea Torsello, Marta Simeoni



# Circuiti integrati

---

I circuiti logici sono realizzati come IC (circuiti integrati)

- realizzati su chip di silicio (piastrina)
- Porte (gate) e fili depositati su chip di silicio, inseriti in un package e collegati all'esterno con un certo insieme di pin (piedini)
- gli IC si distinguono per grado di integrazione
  - da singole porte indipendenti, a circuiti più complessi



# Circuiti integrati

---

## Integrazione

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

Con tecnologia SSI, gli IC contenevano poche porte, direttamente collegate ai pin esterni

Con tecnologia MSI, gli IC contenevano alcuni componenti base

- circuiti comunemente usati nel progetto di un computer

Con tecnologia **VLSI**, un IC può oggi contenere una CPU completa (o più)

- microprocessore



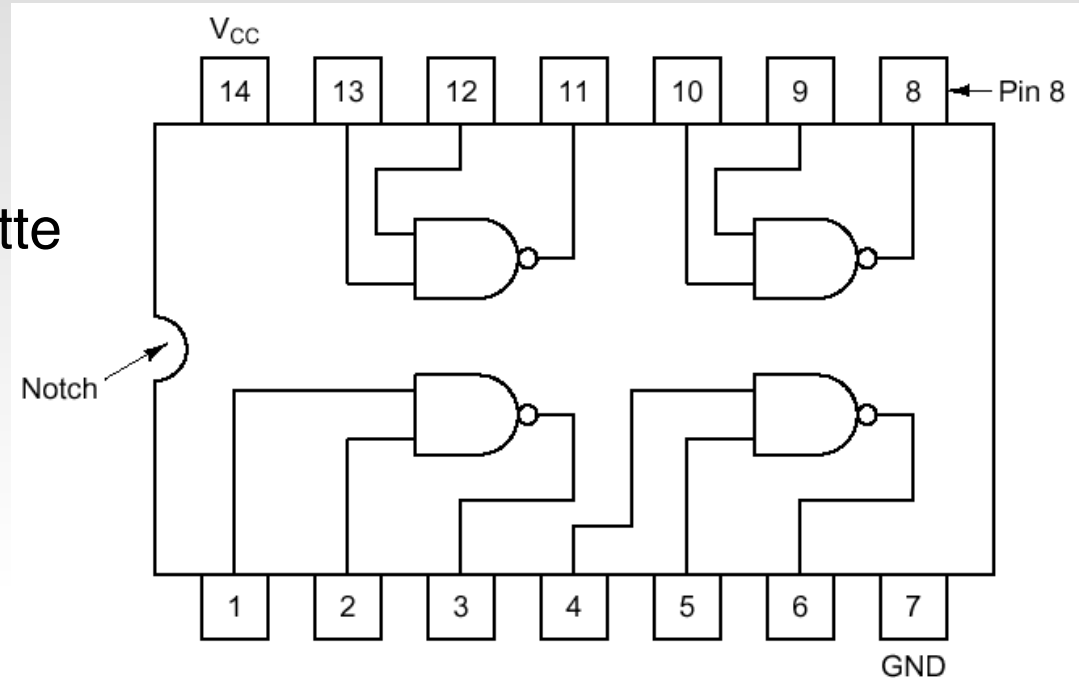
# Esempio di chip SSI

Texas Instruments 7400

- DIP (Dual Inline Package)
- Tensione e terra condivisi da tutte le porte
- Tacca per individuare l'orientamento del chip

SSI: Rapporto **pin/gate** (piedini/porte) grande

- Con l'aumento del grado di integrazione il rapporto **pin/gate** diminuisce (molti gate rispetto ai pin)
  - **chip più specializzati**
  - **es. chip che implementano particolari circuiti combinatori**



# Circuiti combinatori

---

Circuiti combinatori usati quindi come blocchi base per costruire circuiti più complessi

- spesso realizzati direttamente come componenti MSI

Tratteremo:

- Multiplexer e Demultiplexer
- Decoder
- ALU

Per implementare circuiti combinatori useremo

- PLA
- ROM



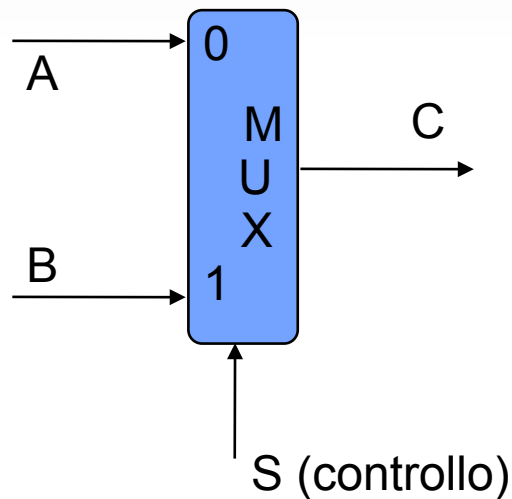
# Multiplexer (1)

$n$  input ed  $1$  output

$\log_2 n$  segnali di *controllo* (da considerare come ulteriori *input* del circuito)

Il multiplexer, sulla base dei *segnali di controllo*, seleziona quale tra gli  $n$  input verrà presentato come output del circuito

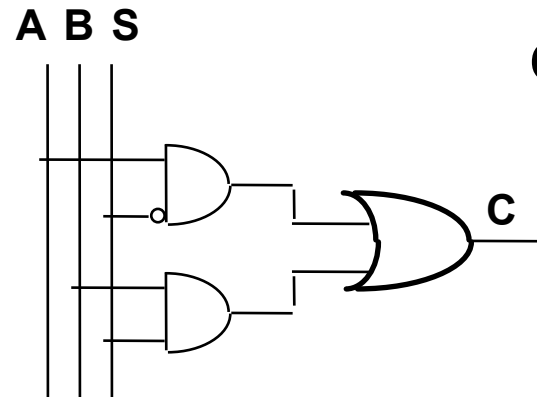
Caso semplice: **Multiplexer 2:1**, con un solo bit di controllo



se  $S=0$ : passa A  
se  $S=1$ : passa B

S \ AB	00	01	11	10
0			1	1
1		1	1	

$$C = A \sim S + BS$$



# Multiplexer (2)

Multiplexer 8:1

$8=2^3$  input e 3 segnali di controllo

$8=2^3$  porte AND e 1 porta OR

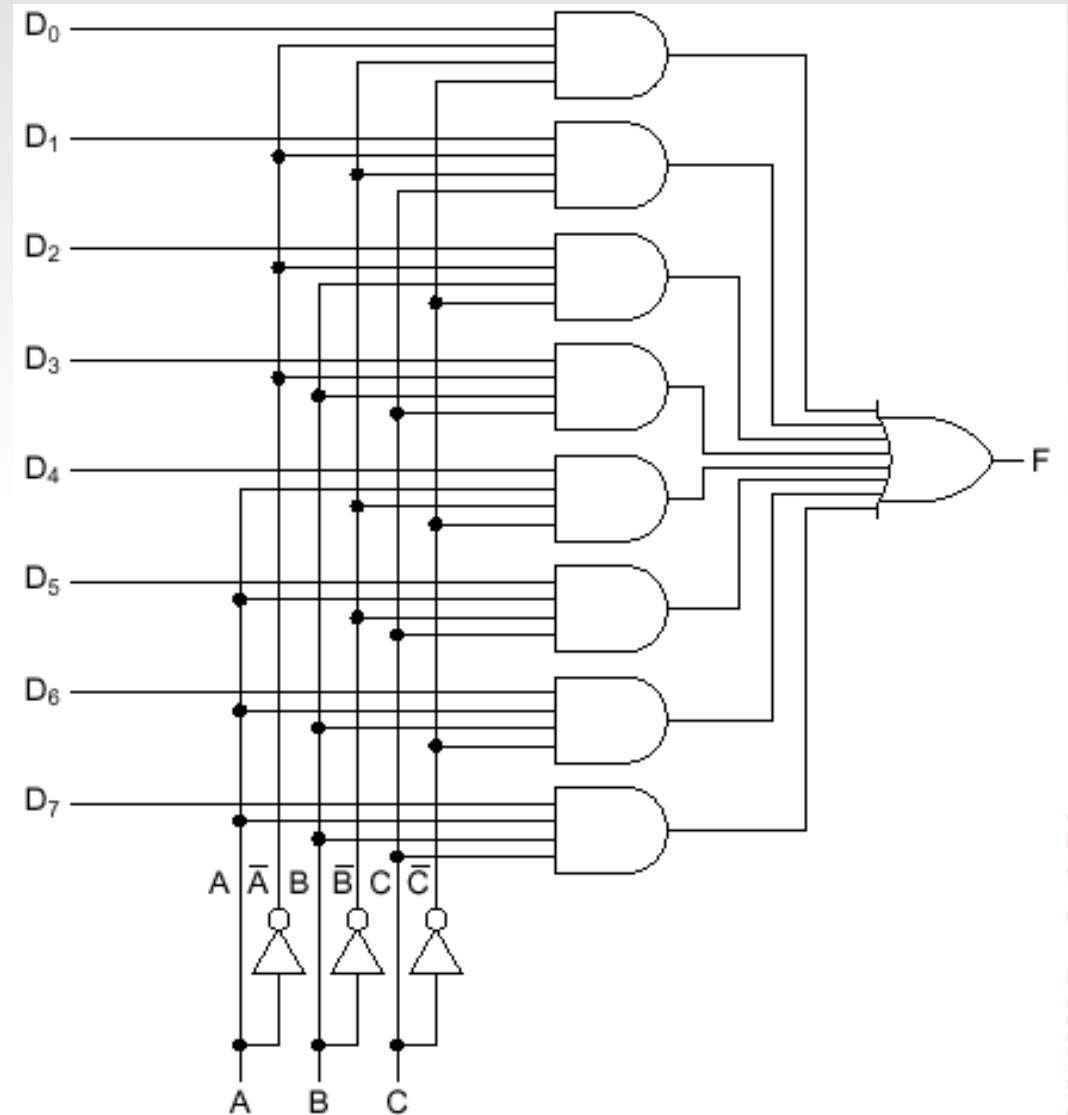
In ogni porta AND entra una combinazione diversa dei segnali di controllo  $A, B, C$

- $8=2^3$  combinazioni possibili

in dipendenza dei valori assunti da  $A, B$  e  $C$ , tutte le porte AND (**eccetto una**) avranno sicuramente output uguale a 0

solo una delle porte produrrà *eventualmente* un valore 1

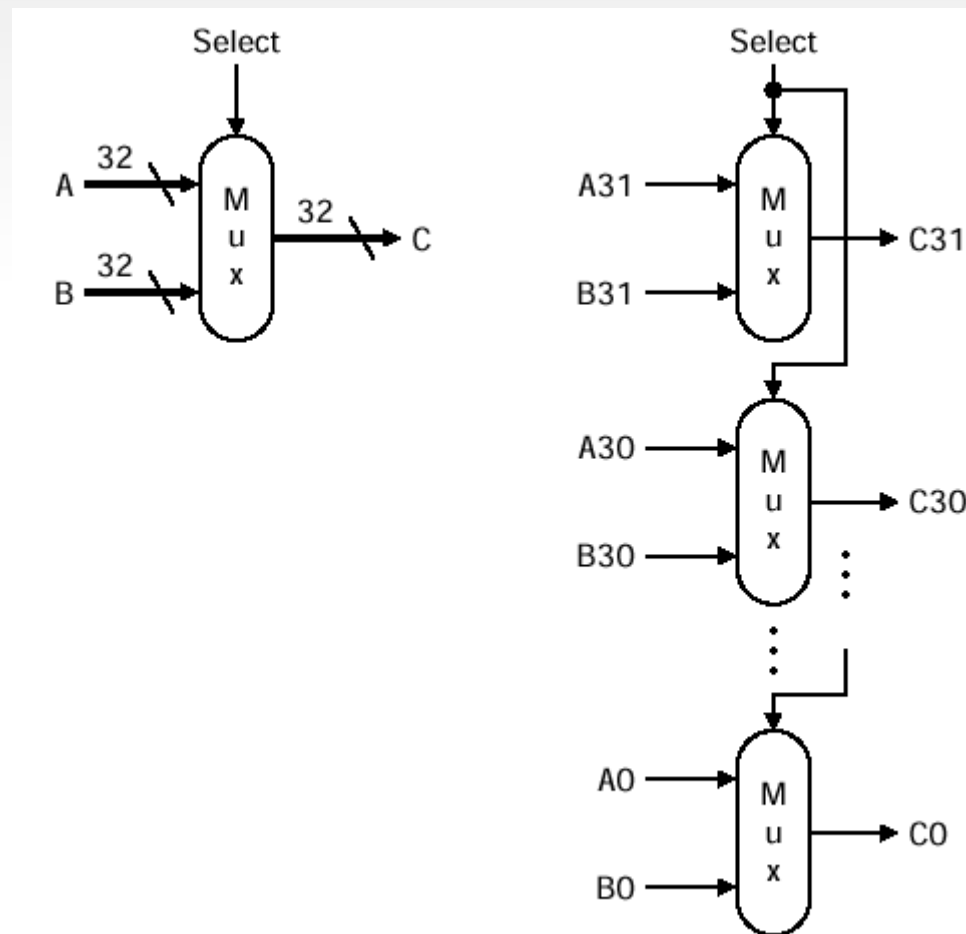
- es.  $A \sim B C = 1 \Rightarrow$  passa  $D_5$



# Multiplexer

Multiplexer 2:1 a 32-bit (con *fili larghi* di 32 bit)

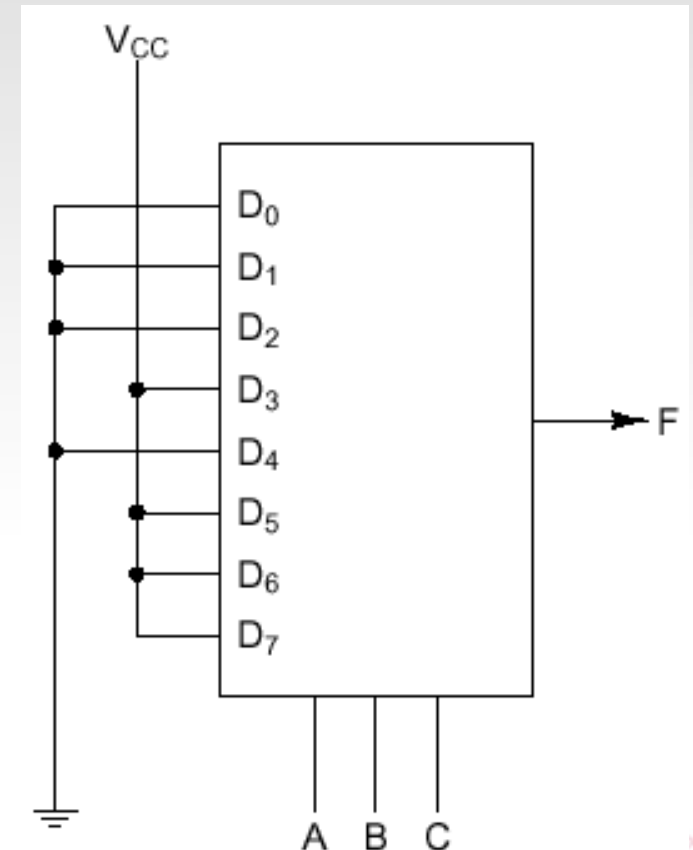
- costruito usando 32 **1-bit Multiplexer 2:1** con un segnale di controllo distribuito ai vari Multiplexer



# Multiplexer e funzioni logiche

## Multiplexer $n:1$

- possono essere usati per definire una qualsiasi funzione logica in  $\log_2 n$  variabili
  - funzione definita da una **tabella di verità** con  $n$  righe
  - le  $\log_2 n$  variabili in input della funzione logica *diventano* i segnali di controllo del multiplexer
- ogni riga della tabella di verità
  - corrisponde ad uno degli  $n$  **input del multiplexer**, collegati ad un generatore di tensione (o alla terra)
    - se l'output, associato alla riga della tabella di verità, è 1 (o 0)
- grande spreco di porte
  - circuito *fully encoded*
  - porte AND con arietà maggiore del necessario (+1)



Componente MSI che realizza un multiplexer 8:1

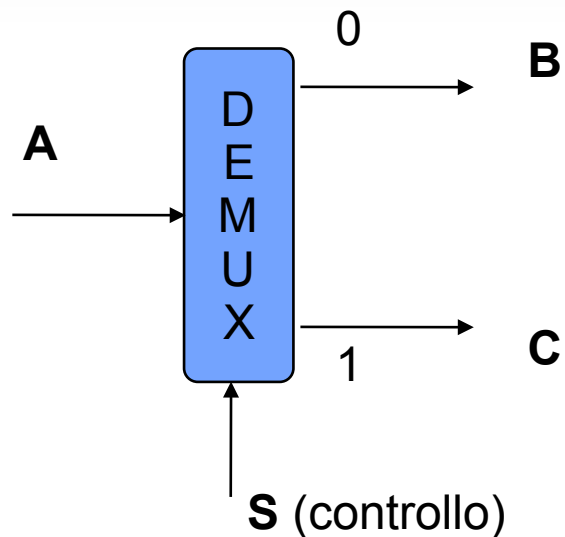
Collegamento per ottenere la funzione:

$$F = \sim ABC + A \sim BC + AB \sim C + ABC$$

# Demultiplexer

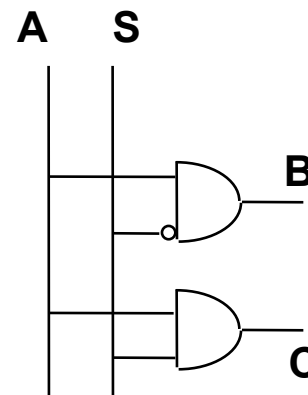
Da 1 singola linea in **input**, a **n** linee in **output**

- $\log_2 n$  segnali di controllo (S)
- se la linea in input è uguale a 0
  - tutti gli output dovranno essere uguali a 0, indipendentemente da S
- se la linea in input è uguale a 1
  - un solo output dovrà essere uguale a 1, tutti gli altri saranno 0
  - l'output da affermare dipende da S



$$B = A \sim S$$

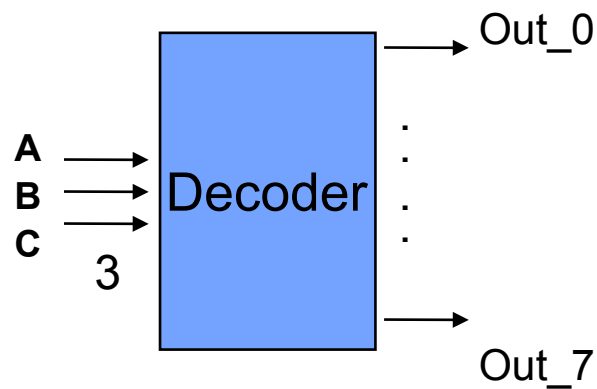
$$C = AS$$



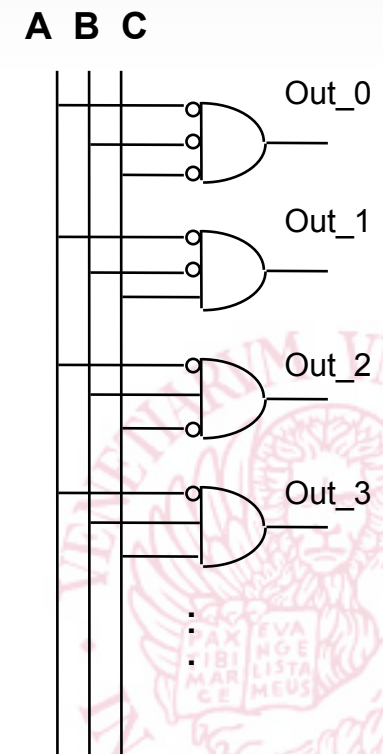
# Decoder

Componente con  $n$  input e  $2^n$  output

- gli  $n$  input sono interpretati come un numero unsigned
- se questo numero rappresenta il numero  $i$ , allora
  - solo il bit in output di indice  $i$  ( $i=0,1,\dots,2^n-1$ ) verrà posto ad 1
  - tutti gli altri verranno posti a 0



A	B	C	0	1	2	3	4	5	6	7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1



# PLA

## Programming Logic Array (PLA)

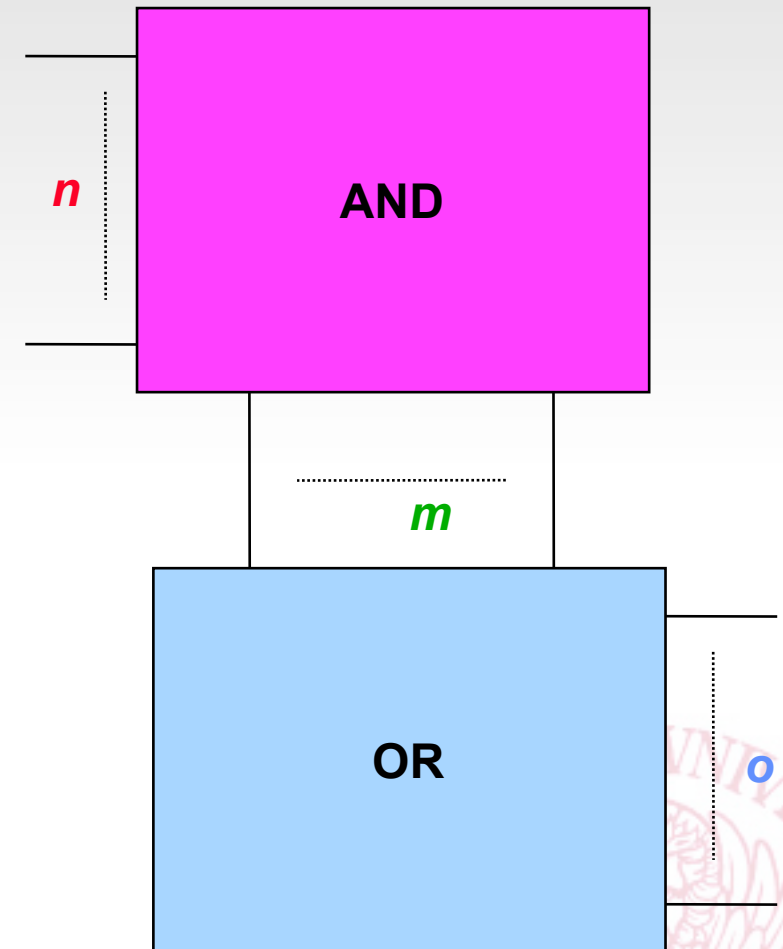
Componente per costruire funzioni logiche arbitrarie

- permette di costruire funzioni in forma SP
  - porte AND al primo livello, e porte OR al secondo livello

–  $n$  input e  $o$  output

- $m$  porte AND
- $o$  porte OR

- $m$  fissa un limite al numero di **mintermini** esprimibili
- $o$  fissa un limite al numero di funzioni differenti in forma canonica SP



# PLA

Esempio:

- $n=12$
- $o=6$
- $m=50$

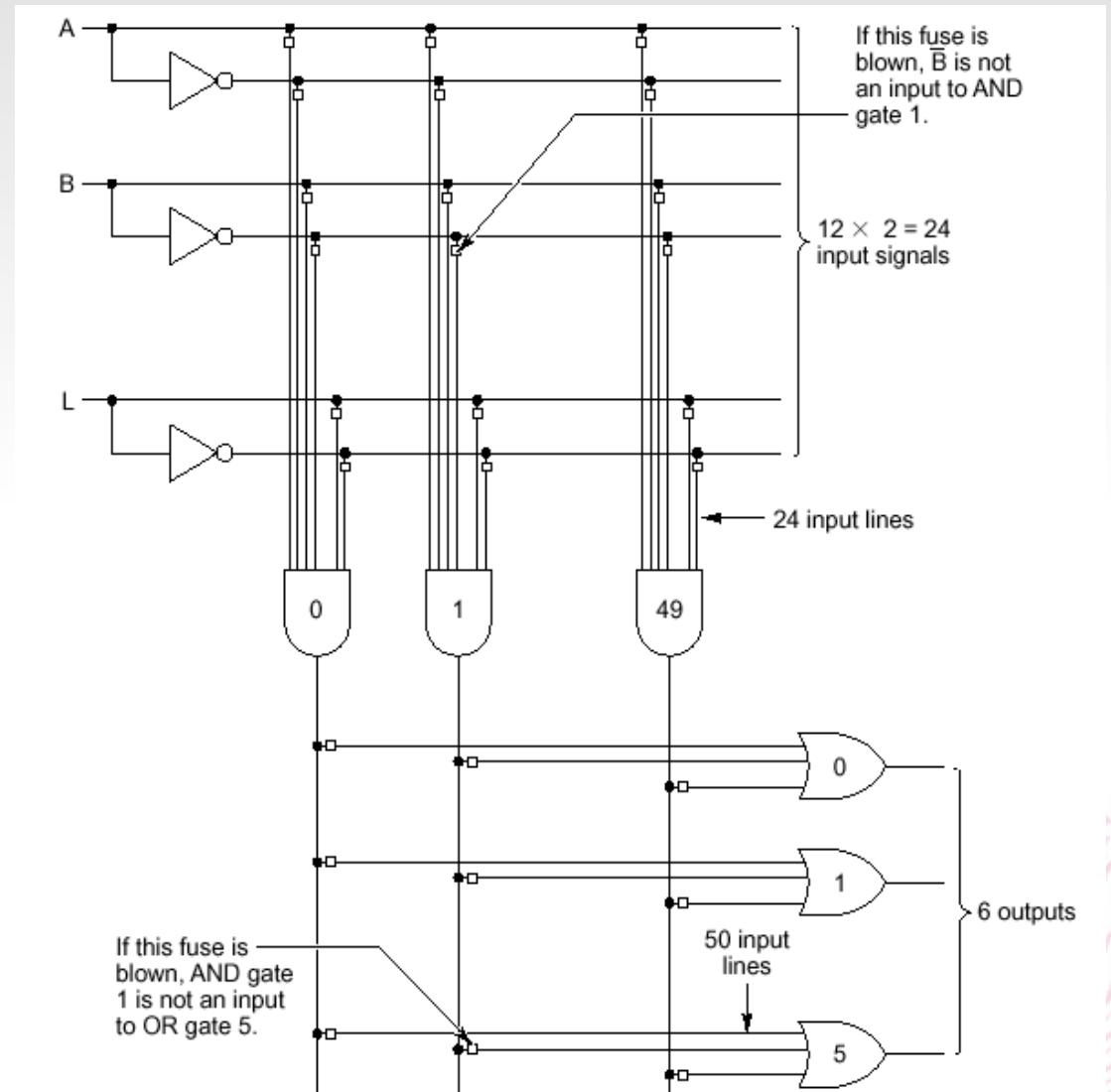
In ogni porta AND entrano  $2n=24$  input

- normali e invertiti

In ogni porta OR entrano  $m=50$  input

Fusibili da bruciare per decidere

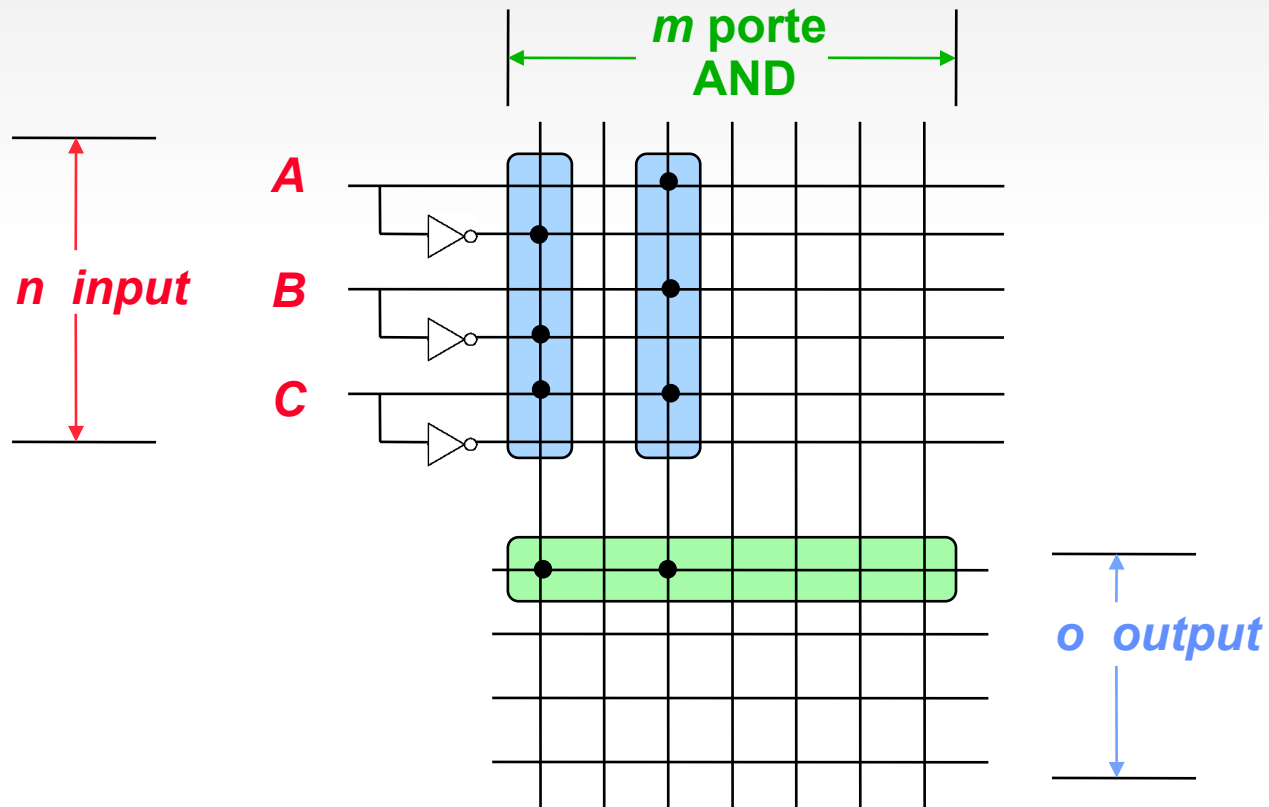
- quali sono gli input di ogni porta AND
- quali sono gli input delle varie porte OR



# PLA

Esempio di funzione:

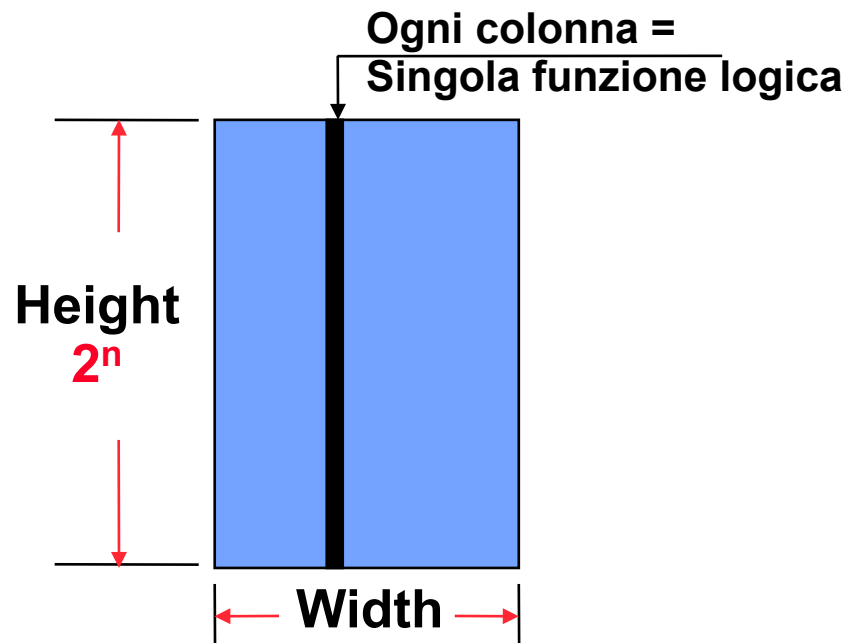
$$O = \sim A \sim B C + A B C$$



# ROM

Memorie usabili anche per implementare, in maniera non minima, funzioni logiche arbitrarie

- ROM (Read Only Memory)
  - PROM (Programmable ROM)
    - scrivibili solo una volta
  - EPROM (Erasable PROM)
    - cancellabile con luce ultravioletta



- In pratica
  - data una **Tabella di Verità**, le ROM sono usate per memorizzare direttamente le diverse funzioni logiche (corrispondenti a colonne distinte nella Tabella)
  - Indirizzo a  $n$  bit
    - individua una specifica combinazione delle  $n$  variabili logiche in input
    - individua una cella di **Width** bit della ROM
- Ogni funzione:
  - Singola colonna della ROM
  - Funzioni **fully encoded**
  - **PLA** più efficiente

