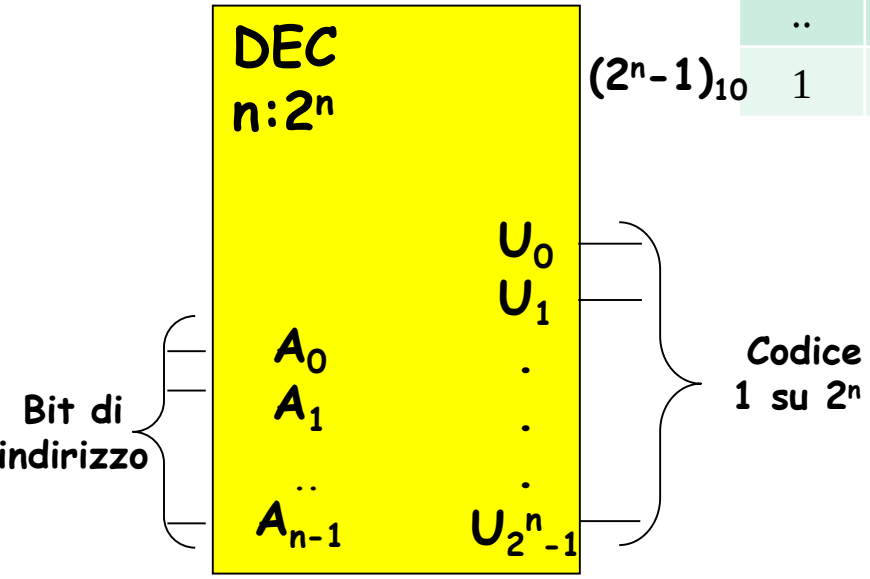
3.5 Sintesi con decoder e OR

Il DEC n:2^n

- Rete di trascodifica da codice binario a codice «1 su N»
- Gli n ingressi vengono spesso indicati come indirizzi (A, address), con A_0 indirizzo di minor peso
- L'indice i dell'uscita U_i attivata è pari al numero rappresentato dalla configurazione binaria degli ingressi: $i = A_{n-1}.2^{n-1} + .. + A_1.2^1 + A_0.2^0$

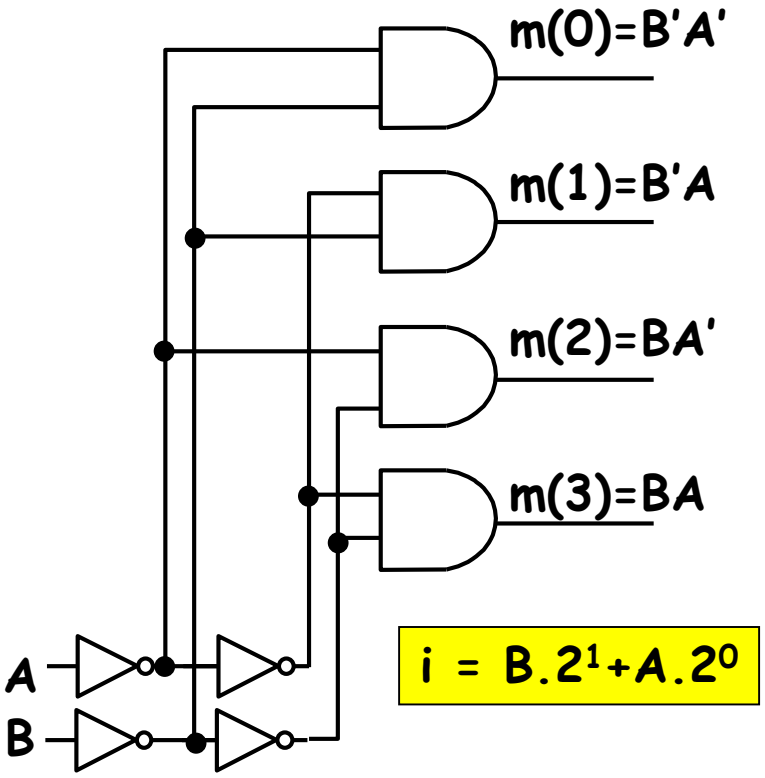
	A_{n-1}	..	A_2	A_1	A_0	U_{2^n-1}	..	U_3	U_2	U_1	U_0
$(0)_{10}$	0	..	0	0	0	0	..	0	0	0	1
$(1)_{10}$	0	..	0	0	1	0	..	0	0	1	0
$(2)_{10}$	0	..	0	1	0	0	..	0	1	0	0
	0	..	0	1	1	0	..	1	0	0	0
	..	..	..	..	..	..	..	..	..	..	..
$(2^n-1)_{10}$	1	..	1	1	1	1	..	0	0	0	0

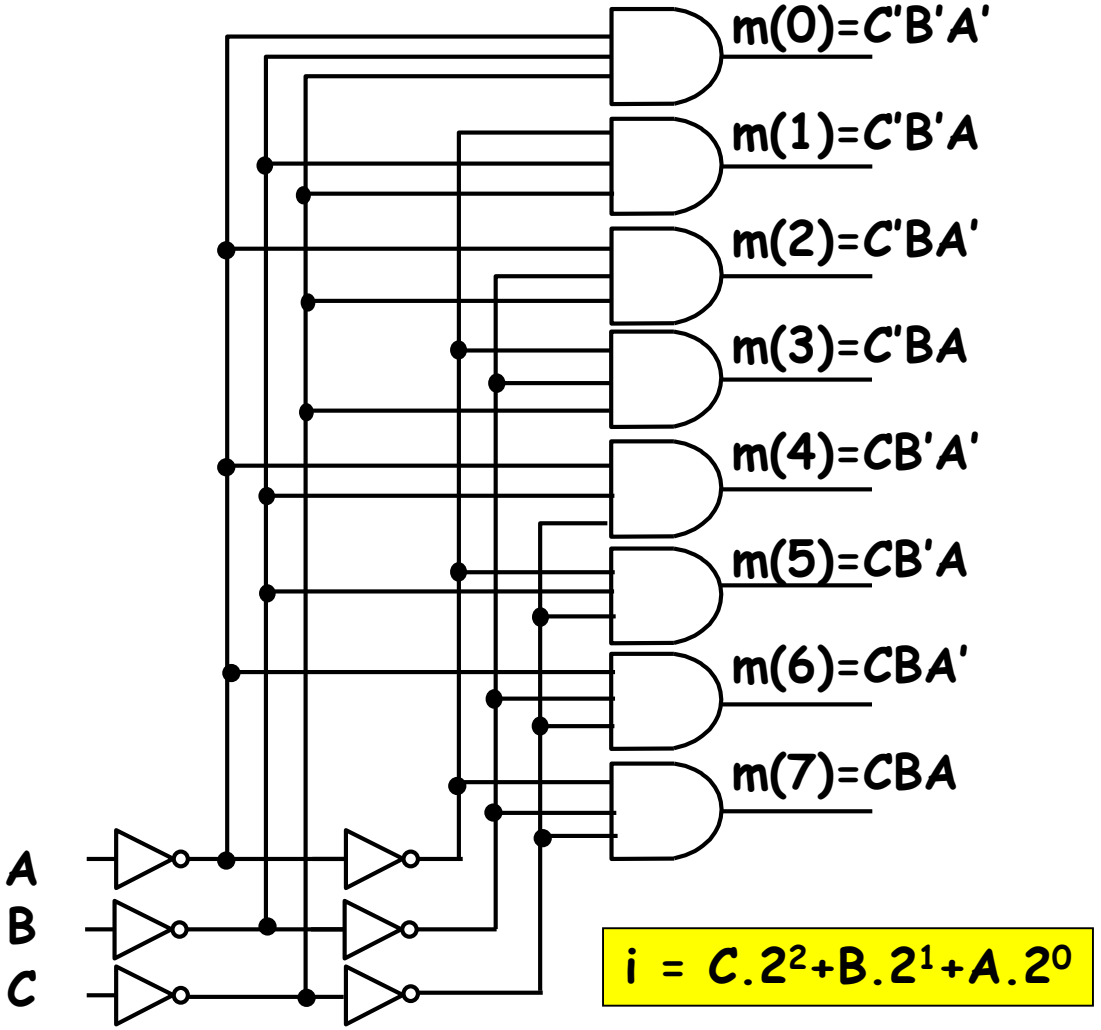


Sintesi SP:

- $U_0 = A_0' \cdot A_1' \cdot A_2' \cdot .. \cdot A_{n-1}'$
- $U_1 = A_0 \cdot A_1' \cdot A_2' \cdot .. \cdot A_{n-1}'$
- $U_2 = A_0' \cdot A_1 \cdot A_2' \cdot .. \cdot A_{n-1}'$
- ..
- $U_{2^n-1} = A_0 \cdot A_1 \cdot A_2 \cdot .. \cdot A_{n-1}$

I DECODER 2:4 e 3:8



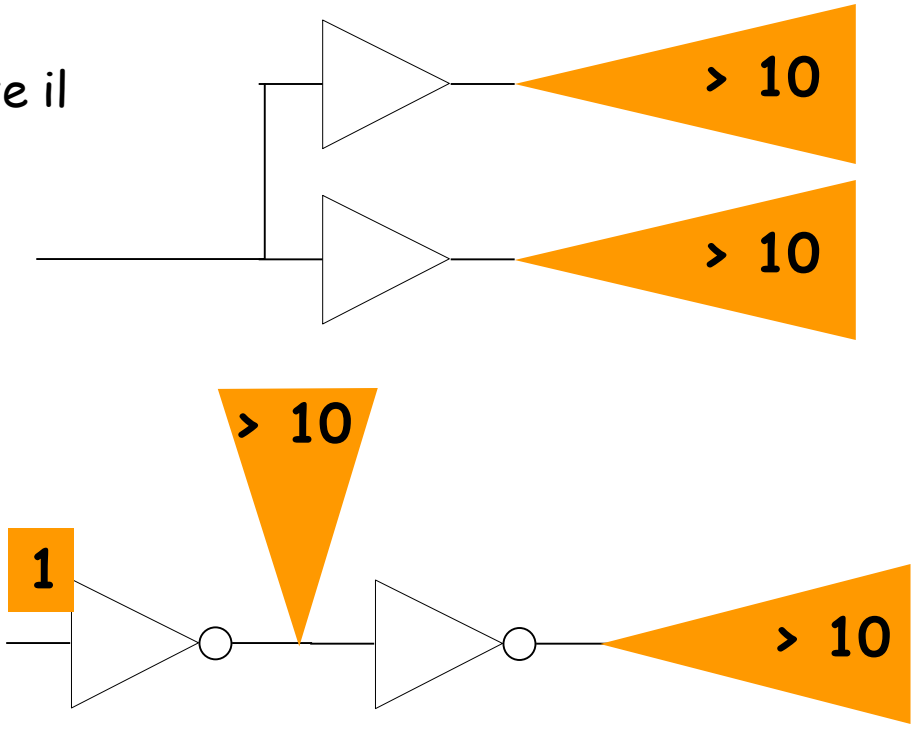


- 2^n AND a n ingressi, 2n NOT (e non n per aumentarne il fan-out, v. prossima slide)
- Il decoder realizza implicitamente **tutti i possibili mintermini** di una espressione a n variabili
- In questa rappresentazione, *A* è il bit in ingresso di peso minore

Effetto di carico: Fan-out

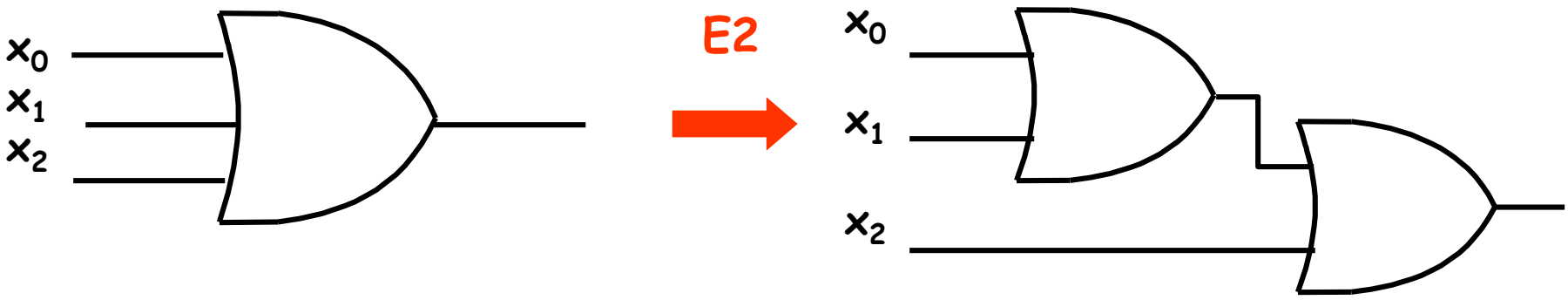
"l'uscita di un gate ha un numero massimo di ingressi di altri gate a cui può essere collegata"

- Il **fan-out** di un gate dipende dalla tecnologia con cui è realizzato ed è generalmente >10, ma è comunque limitato
- per aumentarlo posso interporre gate «buffer» per amplificare elettricamente il segnale
- In alternativa, utilizzo due gate «not» in serie, così da disporre anche del segnale in forma negata (se utile)
- **Decoder**: utilizzo due NOT in serie per far sì che ogni segnale d'indirizzo abbia, all'esterno, un *carico unitario*
- In caso contrario, il collegamento di ogni segnale d'indirizzo ai tanti AND interni sottrarrebbe carico ai segnali di indirizzo nel caso essi fossero utilizzati come ingressi per altri componenti oltre al Decoder

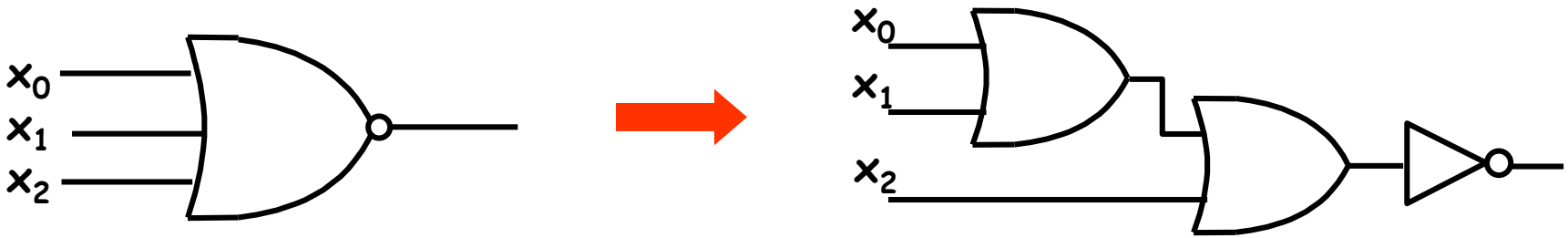


Fan-in

- I gate disponibili hanno un numero di ingressi limitato (spesso 2, tipicamente non più di 8) detto **fan-in**
- Per eseguire operazioni di somma e prodotto logico con un più elevato numero di ingressi si può sfruttare la **proprietà associativa** di AND, OR e EX-OR:

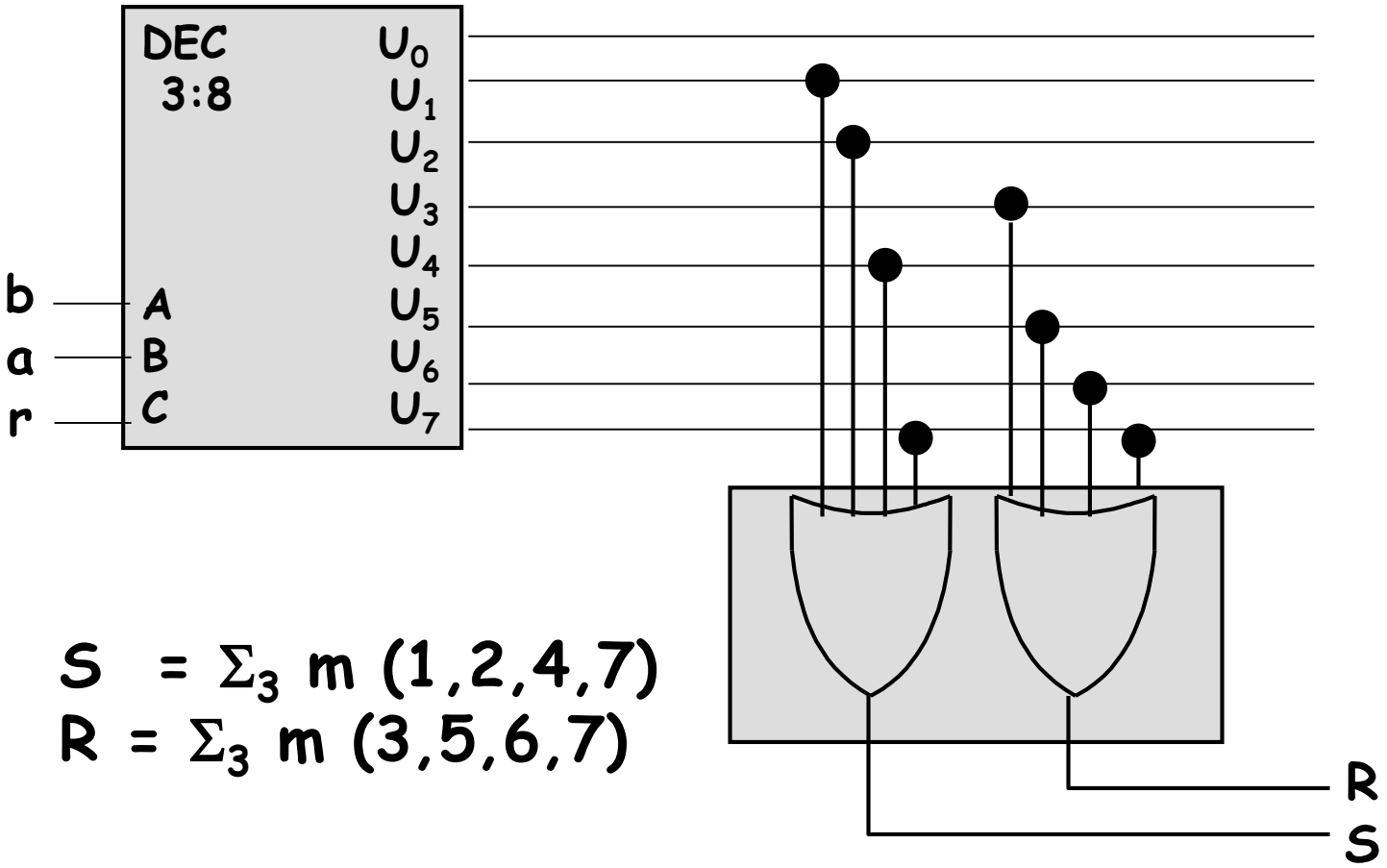


- Nel caso di gate NAND e NOR (per i quali non vale la proprietà associativa) basta applicare un NOT a valle di un AND/OR con il fan-in desiderato



Sintesi del Full Adder con Decoder e Or

- Utilizzando opportunamente Decoder e OR si può sintetizzare qualsiasi espressione in forma canonica SP
- No mappe, no sintesi minima -> tempo di progettazione (quasi) zero!
- Circuito a due livelli: stessa velocità di elaborazione delle reti minime SP/PS

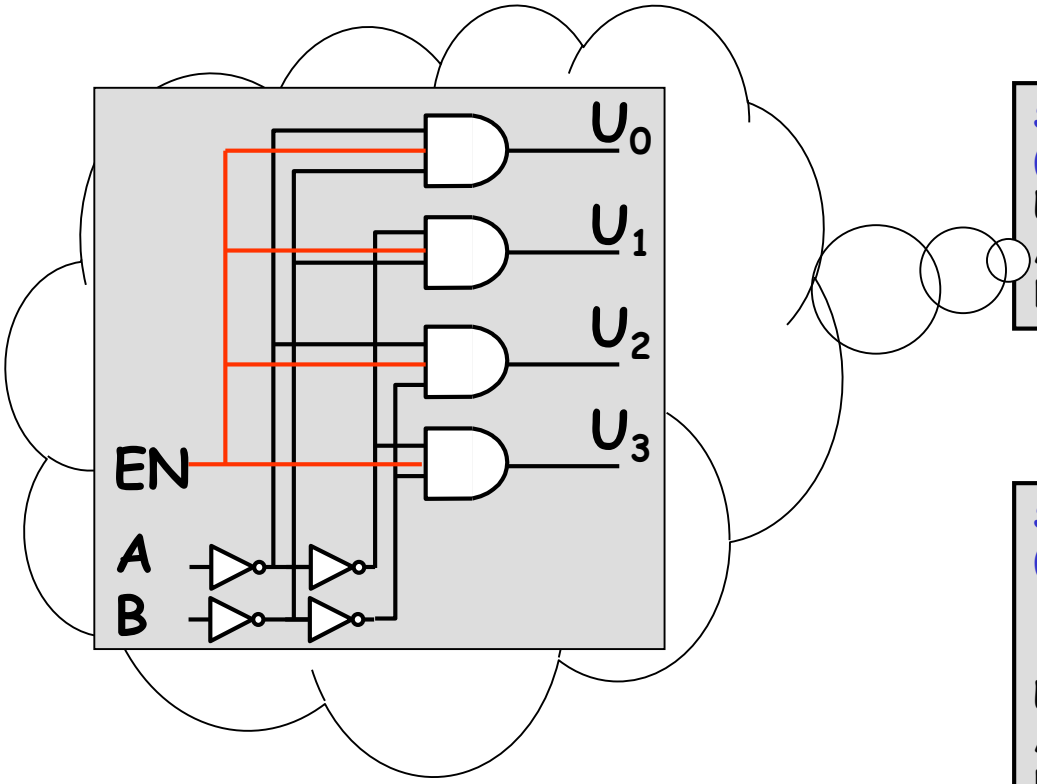


$S = \Sigma_3\, m\,(1,2,4,7)$

$R = \Sigma_3\, m\,(3,5,6,7)$

Il circuito integrato DECODER

- **Decoder** o **Rete di decodifica**: rete logica combinatoria che realizza i 2^n distinti mintermini di n variabili (***n = 2,3,4***)
- Medium Scale Integration (**MSI**)



SN74139 (MSI)	
	U_0
	U_1
EN	U_2
A	U_3
B	

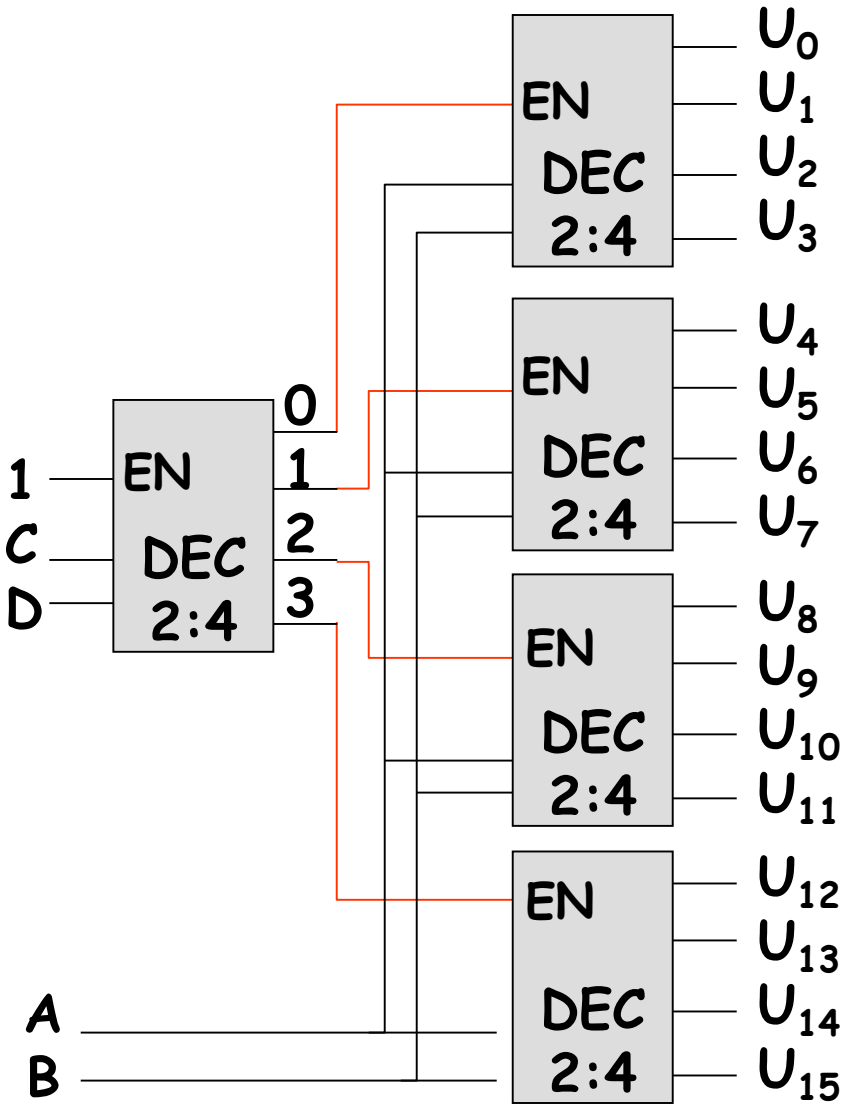
SN74138 (MSI)	
	U_0
	U_1
	U_2
	U_3
EN	U_4
A	U_5
B	U_6
C	U_7

SN74154 (MSI)	
	U_0
	U_1
	U_2
	U_3
	U_4
	U_5
	U_6
	U_7
	U_8
	U_9
	U_{10}
EN	U_{11}
A	U_{12}
B	U_{13}
C	U_{14}
D	U_{15}

Segnale di enable: l'ingresso EN=0 disabilita tutte le uscite

Composizione modulare di un Decoder 4:16

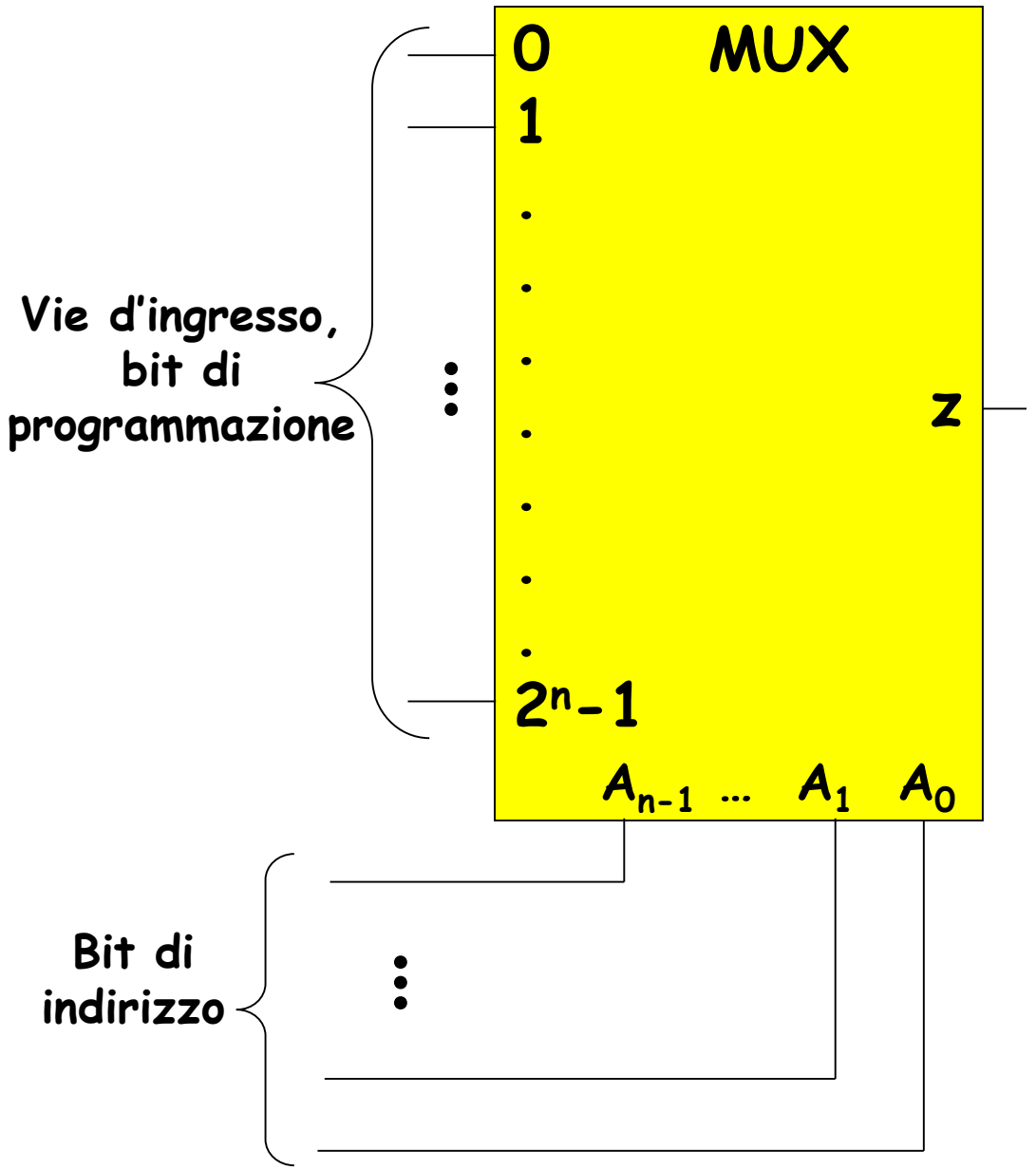
- Limiti della sintesi con Decoder: con n elevato (es. n>4) i Decoder non sono disponibili in quanto il numero di componenti e di uscite cresce esponenzialmente con n
- Soluzione: composizione modulare (il prodotto logico gode della proprietà associativa)
- Indirizzi di peso minore (bit meno significativi) portati negli stadi a valle (es. AB)



3.6 Multiplexer

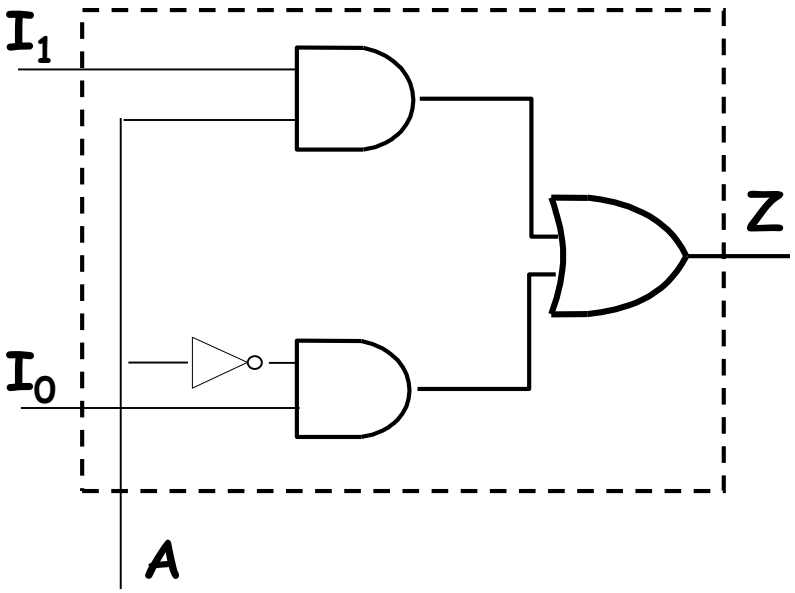
Multiplexer

- Selettore a 2^n ingressi
- Gli n bit di indirizzo selezionano uno tra i 2^n ingressi detti «vie d'ingresso» o «bit di programmazione» a seconda dell'uso che si fa del MUX (come selettore o generatore di funzione, v. prossime slides)
- Al crescere di n cresce esponenzialmente il n° delle vie!
- L'ingresso correntemente attivato si determina dagli indirizzi secondo la relazione: $i = A_{n-1}.2^{n-1} + .. + A_1.2^1 + A_0.2^0$

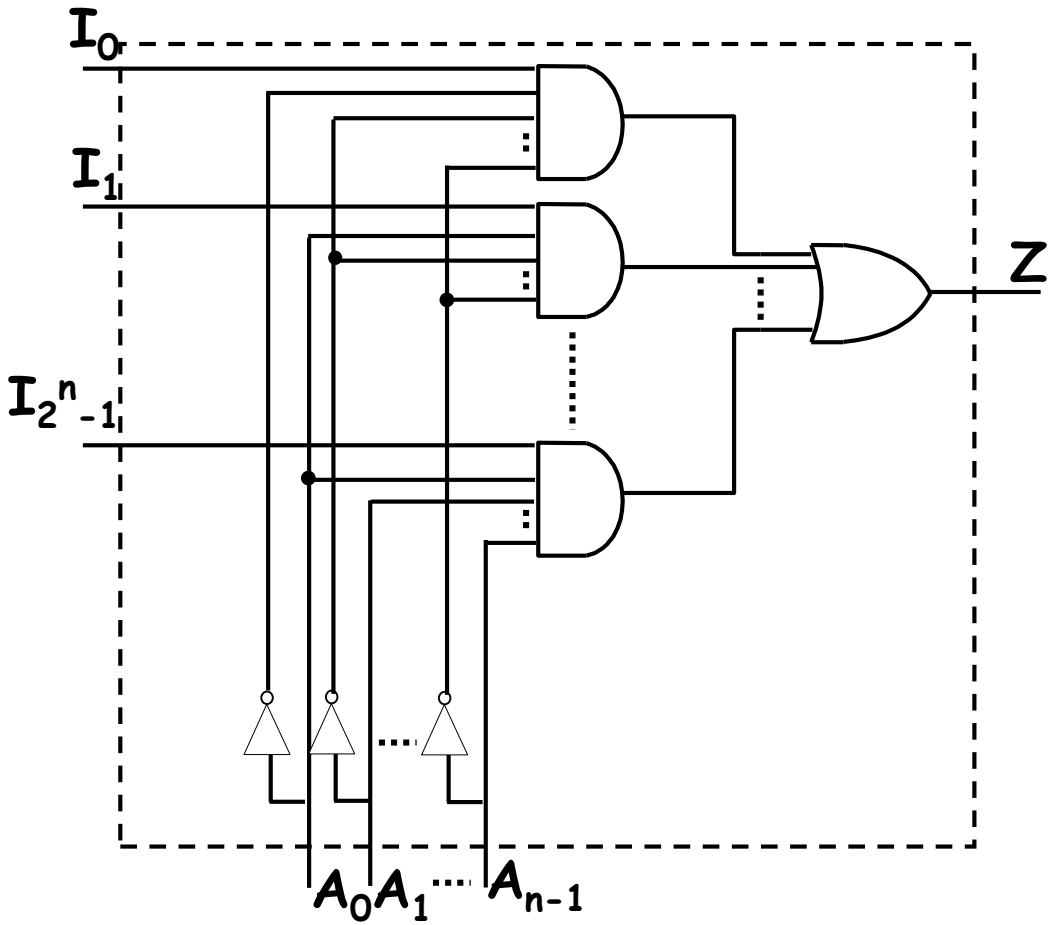


Multiplexer a 2 e 2^n vie

- 2^n AND a n+1 ingressi (n bit di indirizzo + 1 via di ingresso)
- 1 OR a 2^n ingressi
- n (oppure 2n) NOT per avere in forma vera e complementata gli n bit di indirizzo



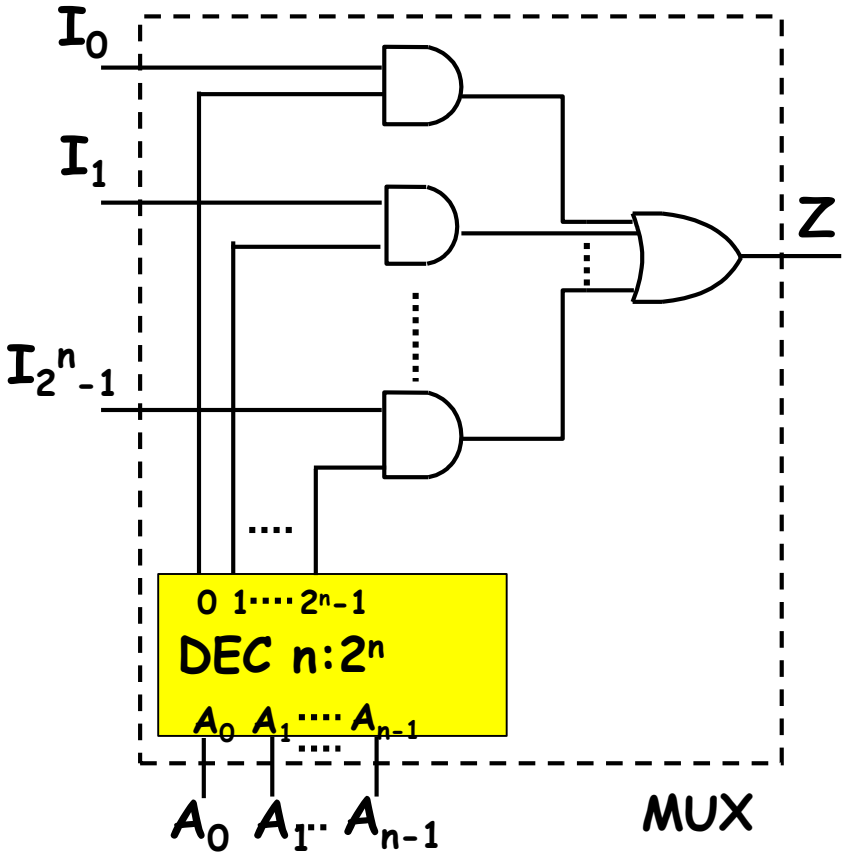
2 vie, 1 bit di indirizzo

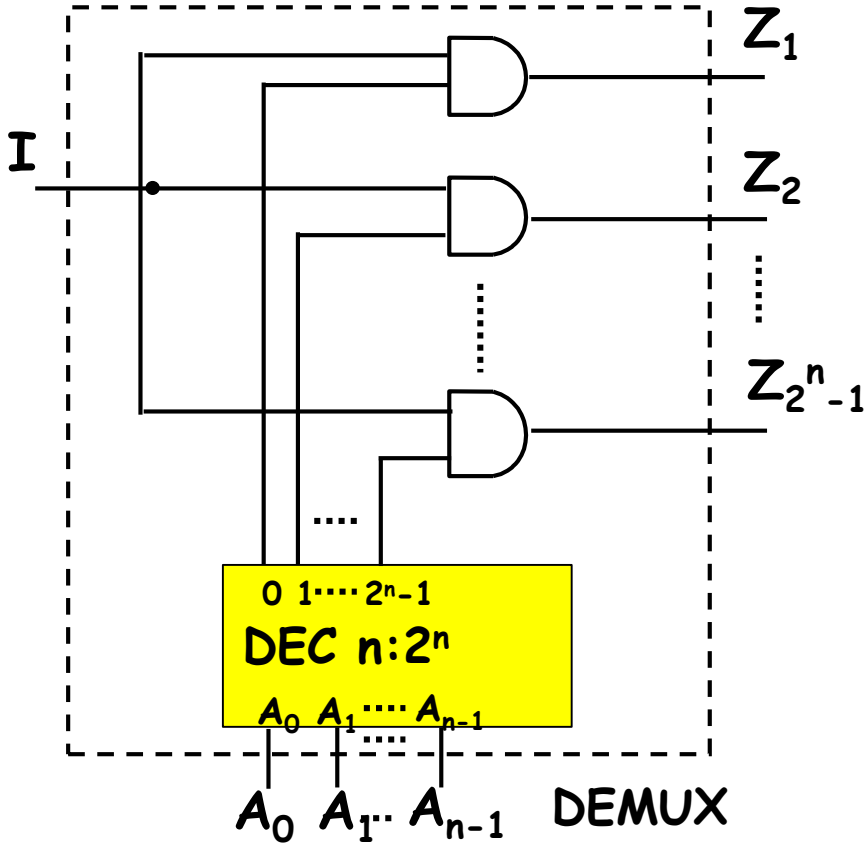


2^n vie, n bit di indirizzo

Mux e Demux con Decoder

- Poichè mux e demux sfruttano al loro interno tutti i mintermini generabili a partire dai bit d'indirizzo, si può utilizzare un decoder per semplificarne la realizzazione
- **Demux**: stessa struttura del Decoder con ingresso di Enable (questo componente è infatti noto anche come ***Decoder-Demultiplexer***)





Il componente MUX può essere sfruttato per la sintesi di espressioni canoniche?

Full Adder : espressioni generali

- Partendo dall'espressione canonica SP della funzione R di un Full-Adder, «complichiamola» utilizzando le equivalenze notevoli:

$$R = m(3) + m(5) + m(6) + m(7)$$

$$= a' b r + a b' r + a b r' + a b r$$

Identità – E5

$$R = m(3) \cdot 1 + m(5) \cdot 1 + m(6) \cdot 1 + m(7) \cdot 1$$

$$= a' b r \; 1 + a b' r \; 1 + a b r' \; 1 + a b r \; 1$$

Limite – E6

$$R = m(3) \cdot 1 + m(5) \cdot 1 + m(6) \cdot 1 + m(7) \cdot 1 +$$
$$m(0) \cdot 0 + m(1) \cdot 0 + m(2) \cdot 0 + m(4) \cdot 0$$
$$= a' b r \; 1 + a b' r \; 1 + a b r' \; 1 + a b r \; 1 +$$
$$a' b' r' \; 0 + a' b' r \; 0 + a' b r' \; 0 + a b' r' \; 0$$

Espressioni generali

- Dal **Teorema di espansione di Shannon** è possibile dedurre le seguenti regole:

- **Caso SP:** ogni funzione di n variabili è descritta da una espressione in cui compaiono, in somma logica, tutti i mintermini di n variabili ciascuno in prodotto logico con il valore della funzione per la configurazione per cui esso vale 1

$$F(x_1,x_2,\ldots x_i,..x_n) = \sum_{i=0}^{2^n-1} m(i) \cdot F(i)$$

m(i) : mintermine di n bit
F(i) : **valore** della funzione relativo alla configurazione per cui m(i)=1

- **Caso PS:** ogni funzione di n variabili è descritta da una espressione in cui compaiono, in prodotto logico, tutti i maxtermini di n variabili, ciascuno in somma logica con il valore della funzione per la configurazione per cui esso vale 0

$$F(x_1,x_2,\ldots x_i,..x_n) = \prod_{i=0}^{2^n-1} (M(i) + F(i))$$

M(i) : maxtermine di n bit
F(i): **valore** della funzione relativo alla configurazione per cui M(i)=0

- Ciò si ottiene applicando iterativamente l'espansione della funzione rispetto a ciascuna variabile x_i:

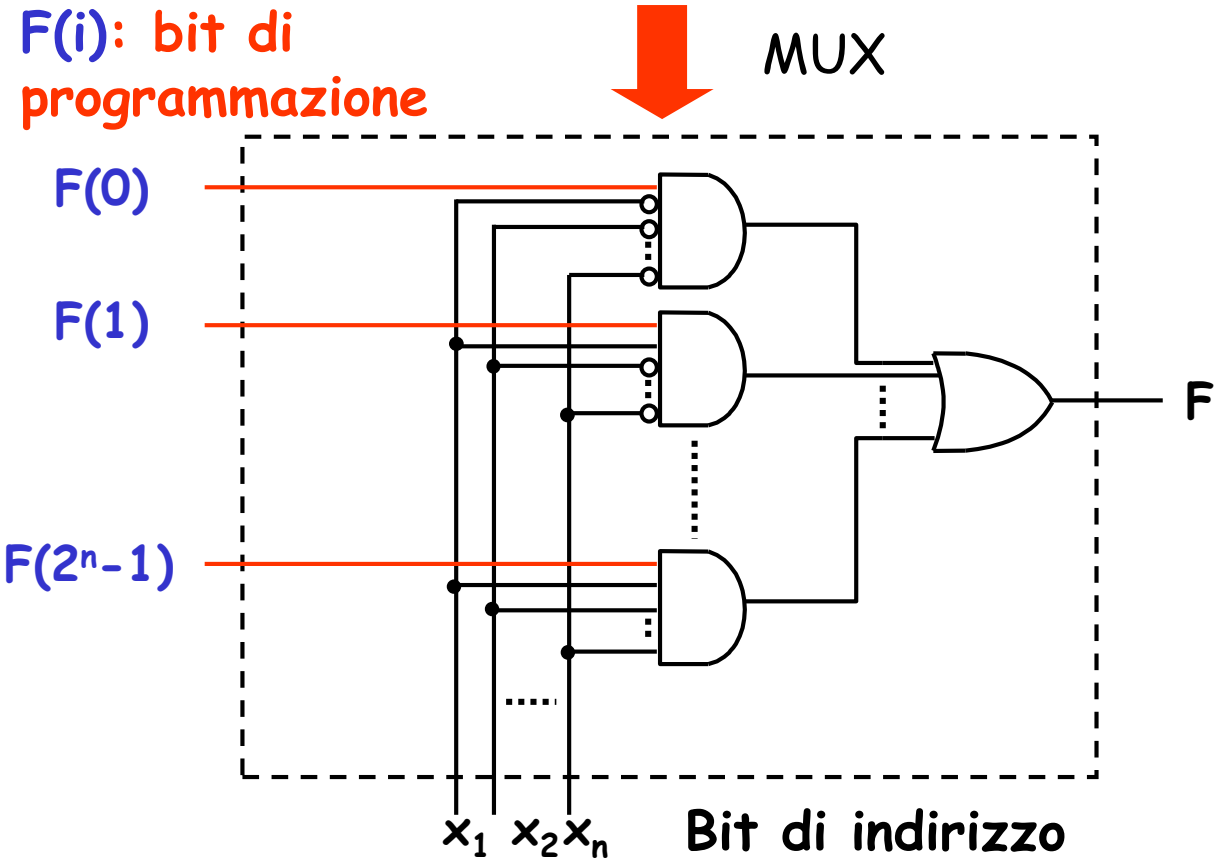
$$F(x_1,x_2,\ldots x_i,..x_n) = x_i\, F(x_1,..x_i=1,..x_n) + x_i'\, F(x_1,..x_i=0,..x_n)$$

Il MUX come rete programmabile

- Il MUX si adatta bene a realizzare l'espressione precedente nel caso SP
- **n variabili** -> occorre un MUX a **n bit di indirizzo**
- il MUX viene qui utilizzato come *generatore di funzioni* (medesimo componente, *programmabile* a seconda della funzione che si vuole realizzare)

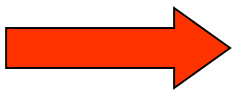
$$F(x_1, x_2, \ldots x_i, \ldots x_n) = \sum_{i=0}^{2^n-1} m(i) \cdot F(i)$$

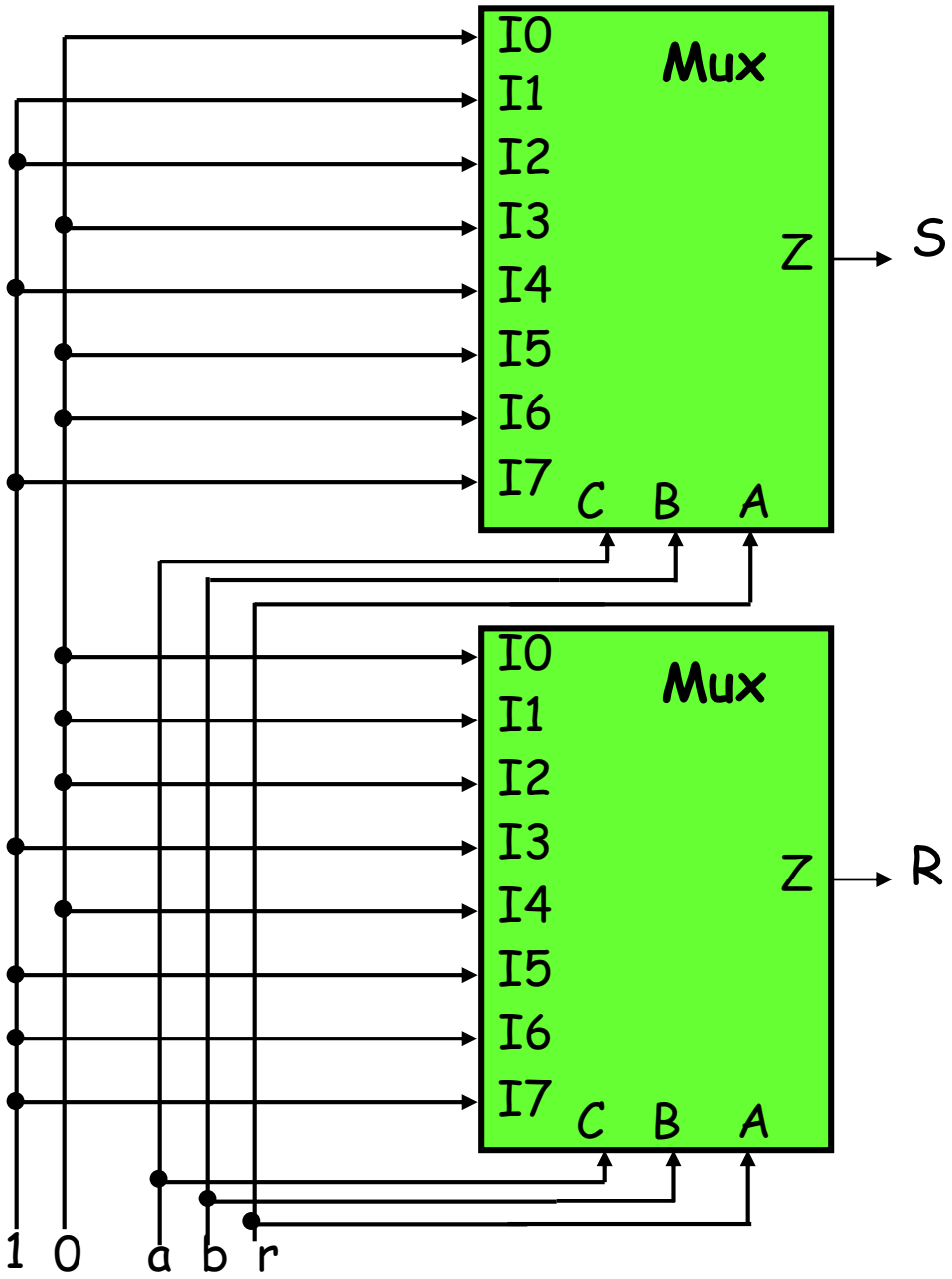
Espressione generale SP



Sintesi di un full-adder con MUX

a	b	r	S	R
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

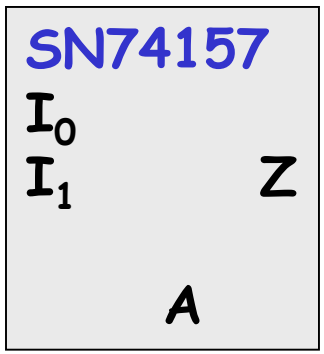


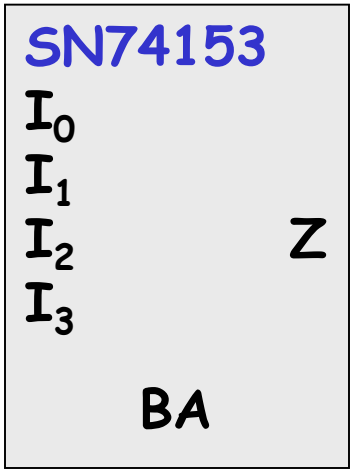


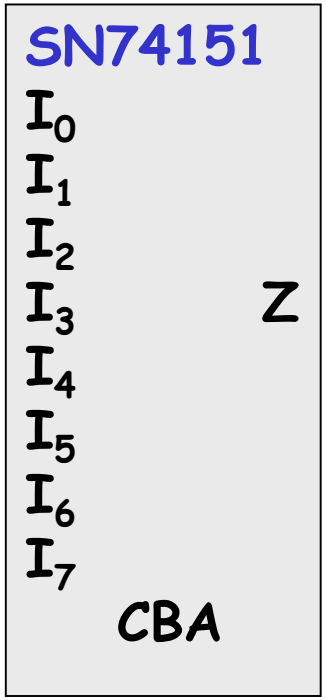
- Riproduco in ingresso al MUX i contenuti della tabella della verità
- Massimizzo dunque il criterio della **rapidità di progetto**
- La velocità di elaborazione rimane invece quella delle reti minime (come si vede dalla realizzazione del MUX)

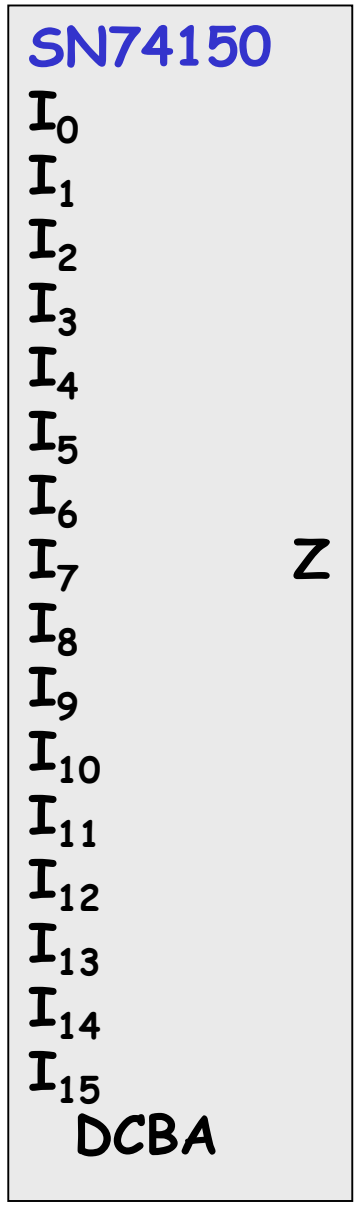
I Multiplexer in forma integrata

- Ulteriore esempio di MSI: MUX a 2,4,8 e 16 bit di programmazione
- Il numero di vie cresce esponenzialmente con il numero di bit d'indirizzo: ma il numero di pin di un chip è limitato
- i MUX disponibili sotto forma di circuito integrato hanno al più 16 bit di programmazione





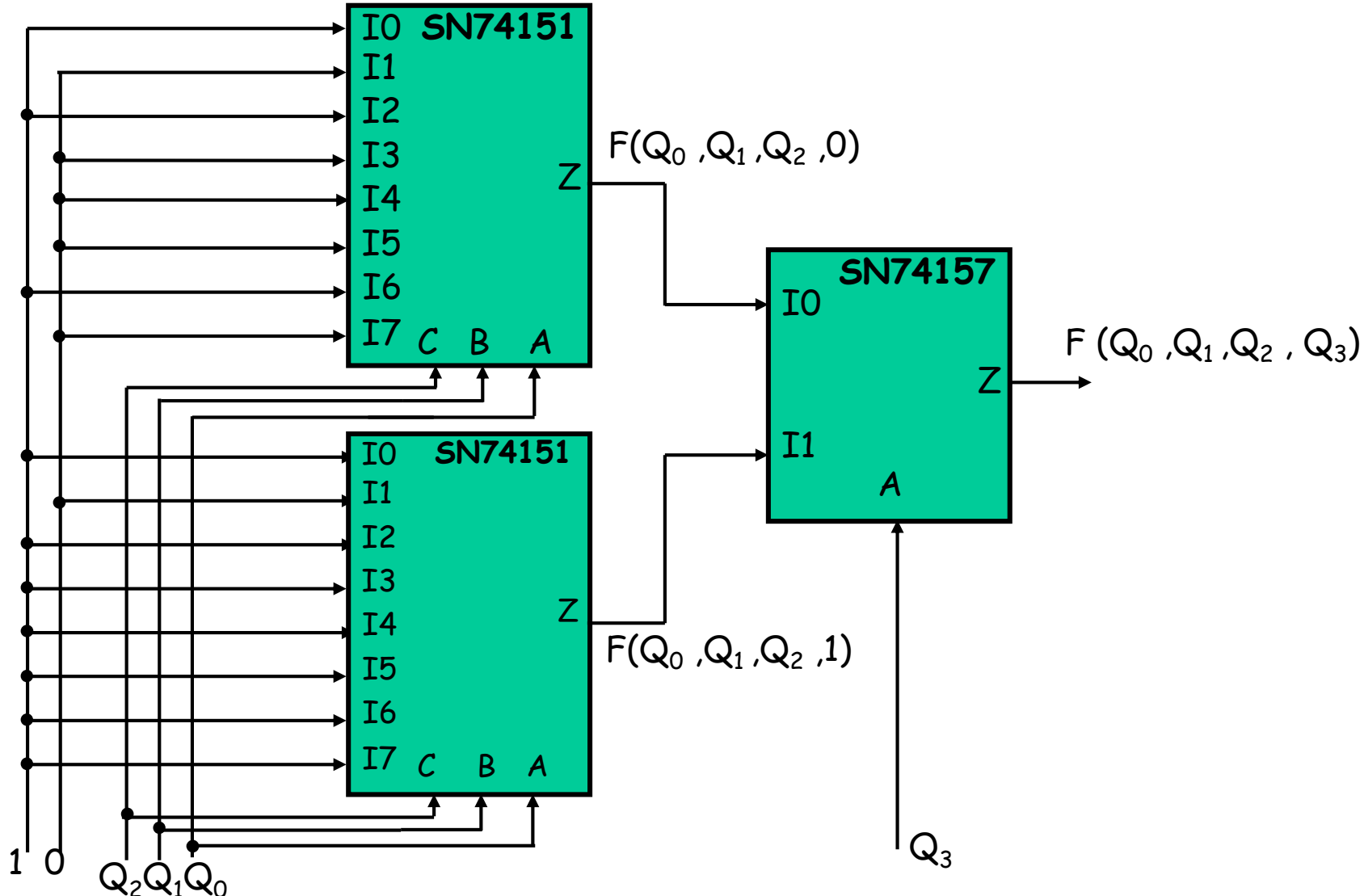




A, B, C, D *bit d'indirizzo*
I_i *via o bit di programmazione*

Sintesi a MUX di funzioni di 4 variabili

- Composizione modulare è possibile anche con MUX per sopperire ai limiti del numero di ingressi
- Indirizzi dei dispositivi a valle corrispondono ai bit di **maggior** peso (es. Q_3)

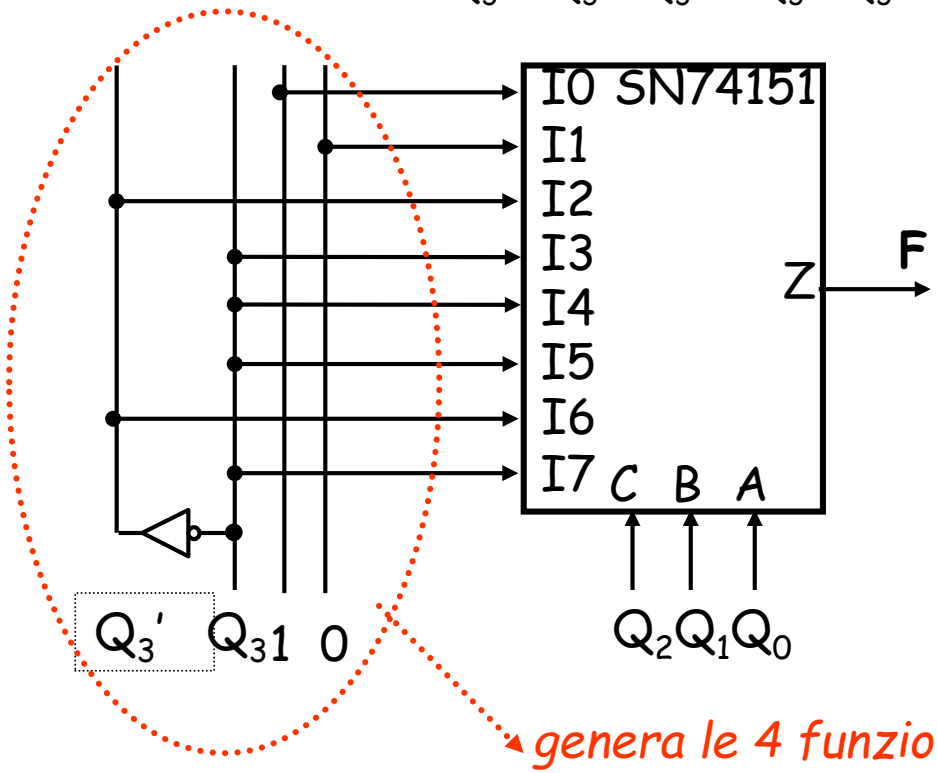


Sintesi con MUX a n-1 bit d'indirizzo

- Caso più specifico del precedente in cui occorre sintetizzare una espressione a n variabili e si dispone di un MUX a n-1 indirizzi

Q_3 \ $Q_2Q_1Q_0$	000	001	010	011	100	101	110	111
0	1	0	1	0	0	0	1	0
1	1	0	0	1	1	1	0	1
F	1	0	Q_3'	Q_3	Q_3	Q_3	Q_3'	Q_3

$F(0,0,0,Q_3) = 1$
$F(0,0,1,Q_3) = 0$
$F(0,1,0,Q_3) = Q_3'$
$F(0,1,1,Q_3) = Q_3$
$F(1,0,0,Q_3) = Q_3$
$F(1,0,1,Q_3) = Q_3$
$F(1,1,0,Q_3) = Q_3'$
$F(1,1,1,Q_3) = Q_3$



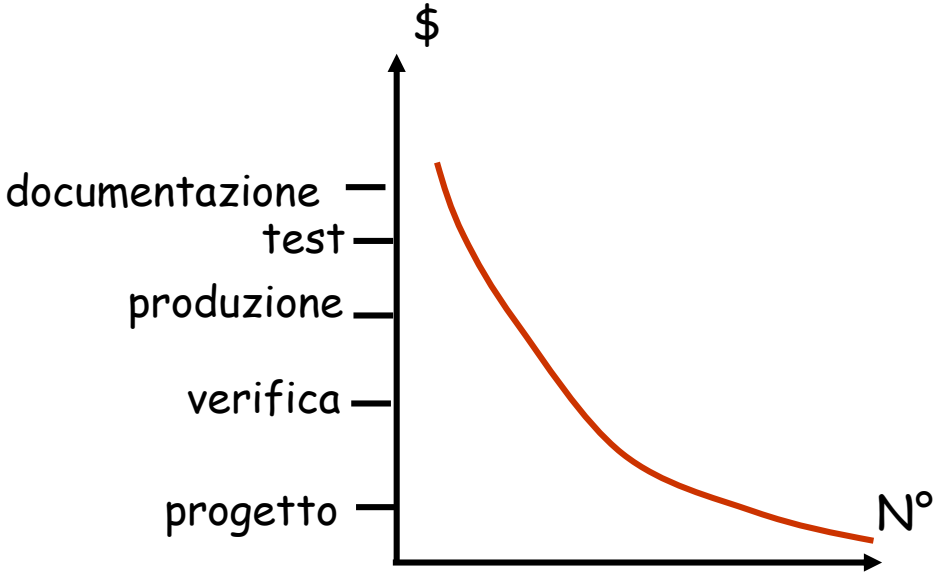
Qualsiasi funzione di n variabili è realizzabile con un NOT e con un MUX dotato di (n-1) bit di indirizzo

genera le 4 funzioni di una variabile !

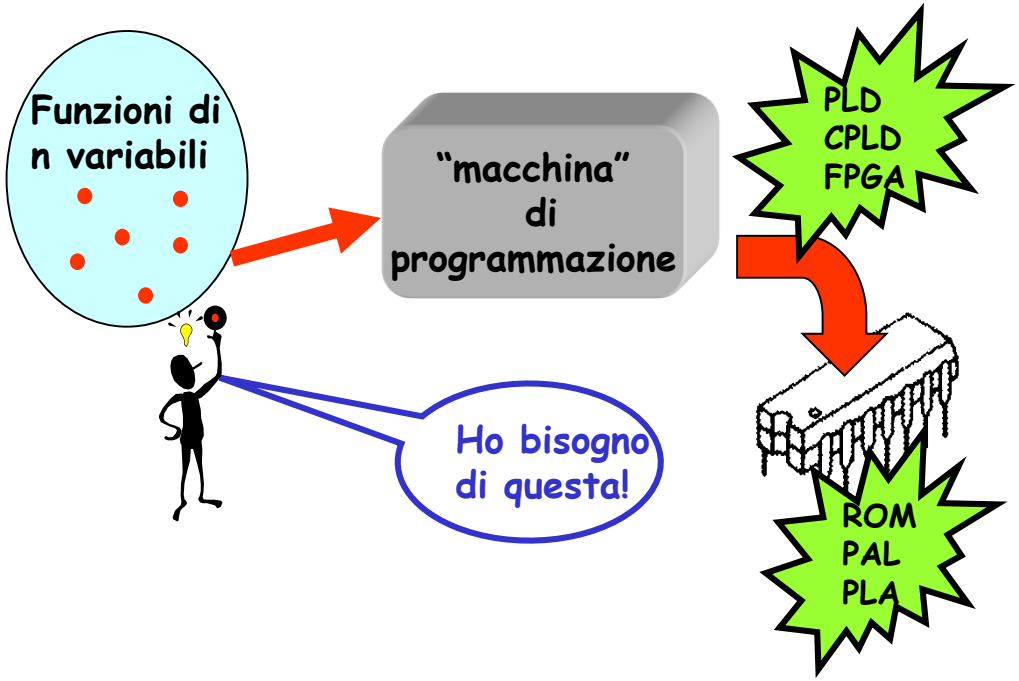
3.7 Reti programmabili

VLSI

- Large Scale Integration (**LSI**) e Very Large Scale Integration (**VLSI**):
  - milioni di gate integrati all'interno dello stesso chip
  - comporta costi molto onerosi in tutte le sue fasi di realizzazione
  - per ammortizzarne le spese ne occorre un vasto impiego

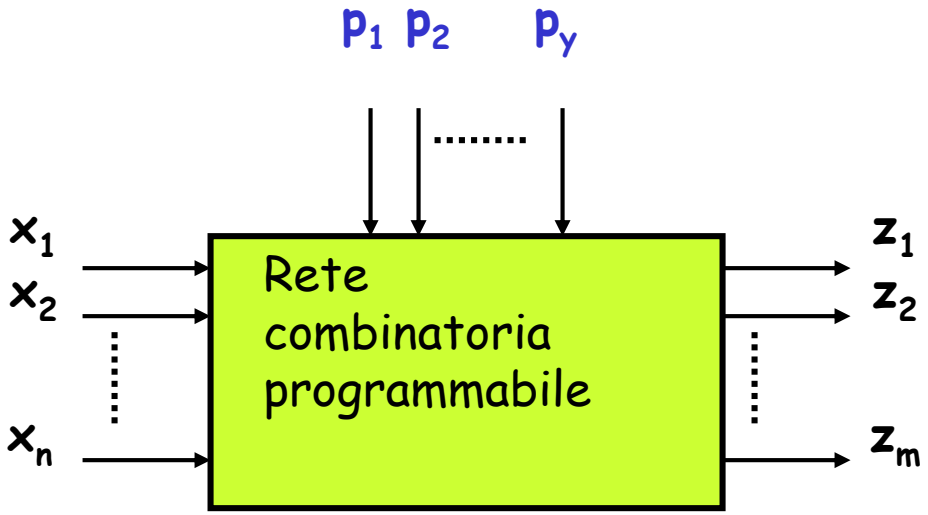


- Impossibile la realizzazione ad hoc, si rende necessaria la possibilità di programmazione
- Questo permette di ammortizzare i costi elevati attraverso grandi volumi di vendita dello stesso prodotto



Le reti combinatorie programmabili

Rete combinatoria programmabile: rete combinatoria in grado di presentare diverse relazioni ingresso/uscita singolarmente selezionabili mediante l'attribuzione di una determinata configurazione di valori ad un gruppo di segnali detti **bit di programmazione**.



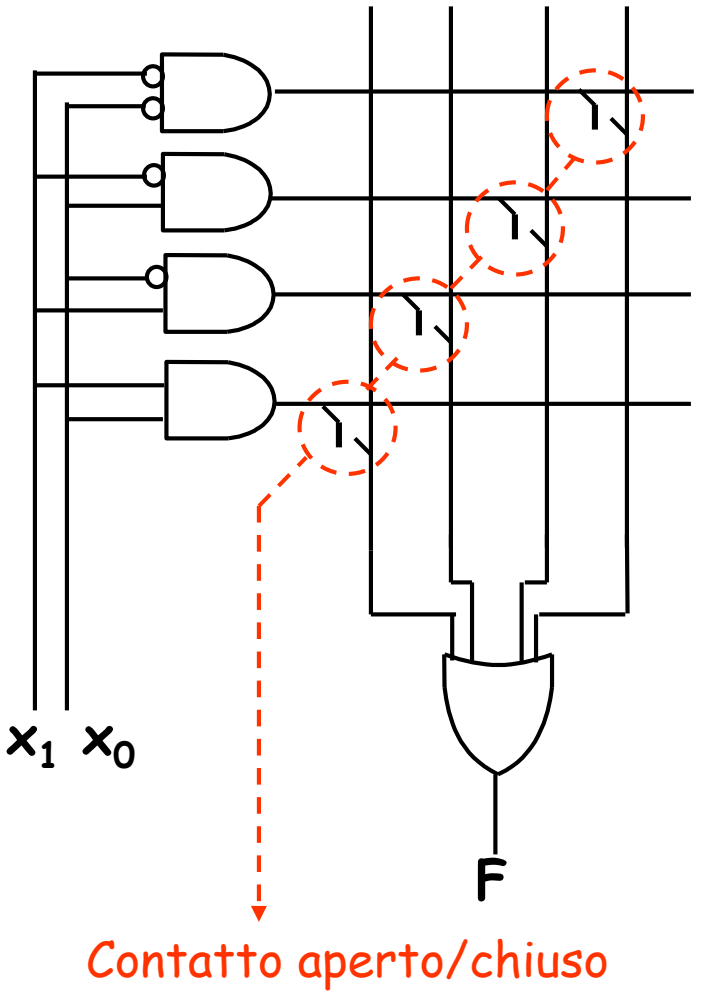
$$z_i = F_i(p_1, p_2, \ldots, p_y, x_1, x_2, \ldots, x_n) = F_P(x_1, x_2, \ldots, x_n)$$

Memorie a sola lettura (ROM)

- Basate sulla **sintesi Decoder+OR**: realizzano espressioni di tipo SP (2^n AND + 1 OR a elevato fan-in)
- Sono dunque reti programmabili in grado di realizzare qualsiasi funzione di n variabili

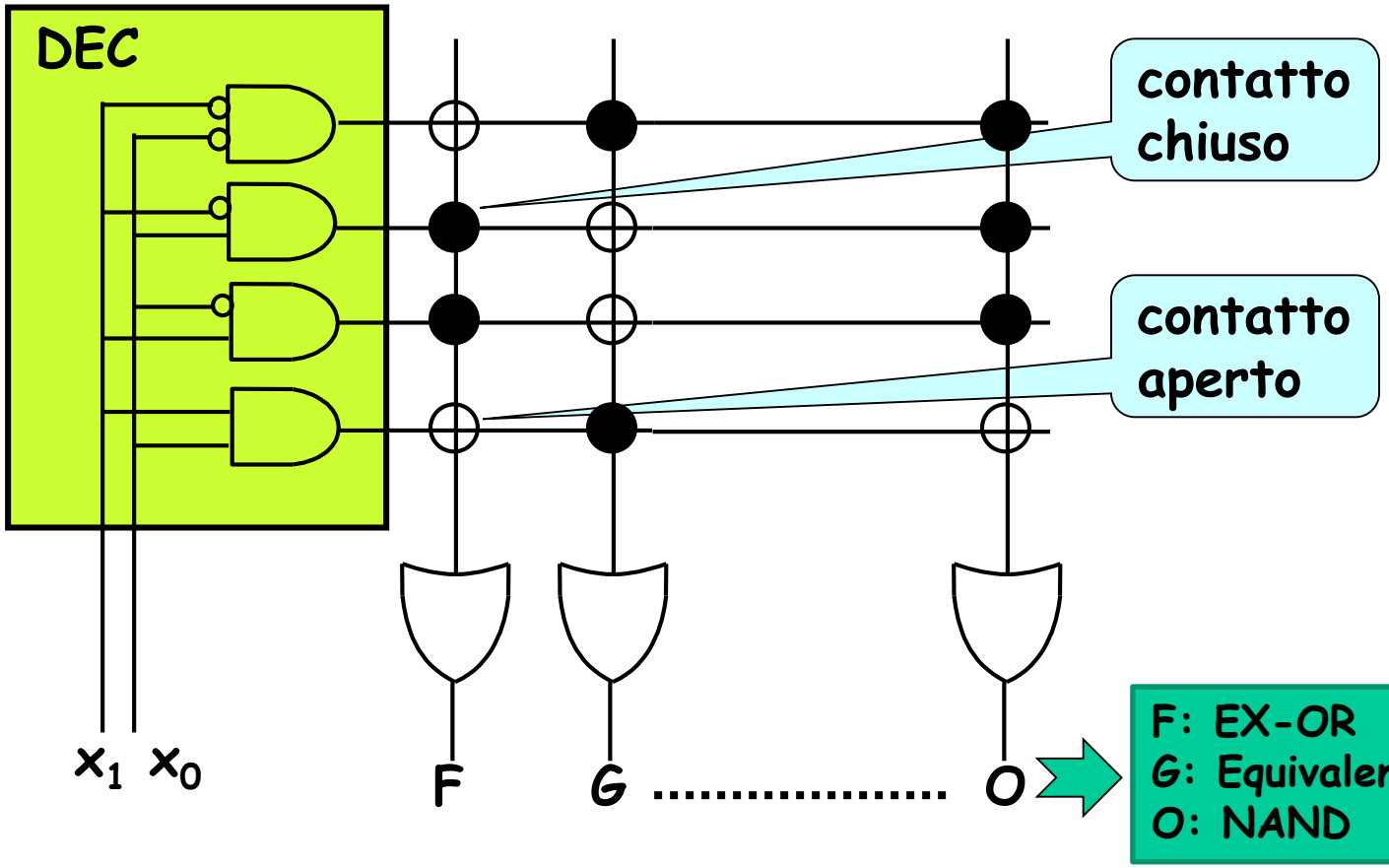
- Programmabili mediante predisposizione opportuna dei contatti tra la batteria di AND e l'OR e NON attraverso segnali esterni (con n=16 il contenitore del chip dovrebbe avere 65536 pin!) -> superato un limite della sintesi a DEC, MUX

- il contatto è chiuso in corrispondenza delle configurazioni che realizzano un mintermine nell'espressione canonica della funzione



Memorie a sola lettura (ROM)

- Rappresentazione "compatta" della struttura di una ROM :



- è possibile sfruttare il *fan-out* del DEC per realizzare in maniera integrata più funzioni

Le ROM come circuiti di memoria

- Una ROM, oltre a essere interpretata come circuito generatore di funzioni, può essere utilizzata anche come circuito di **memoria**
- Es.: ROM per realizzare 8 funzioni con 2 variabili indipendenti

x_1	x_0	$F_{7..0}$	F_7	.....	F_0
0	0	0xAE=1100 1110	1		0
0	1	0x94=1001 0100	1		0
1	0	0x1D=0001 1101	0	.....	1
1	1	0x00=0000 0000	0		0

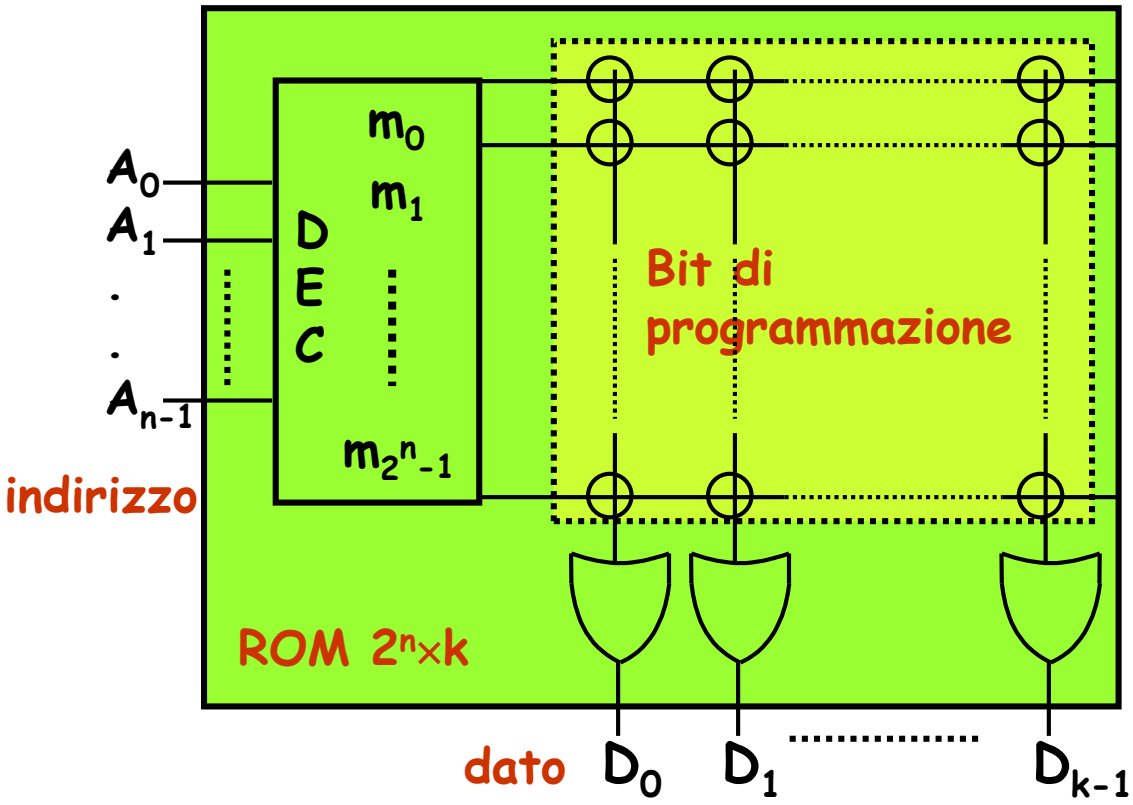
Voglio **memorizzare** 4 byte, 1 per ciascuna configurazione degli ingressi

«programmo» opportunamente la *funzione* F_0 (e le altre 7)

- Ogni configurazione delle variabili di ingresso può essere interpretata come l'indirizzo di un dato formato dai bit programmati nella riga corrispondente della matrice

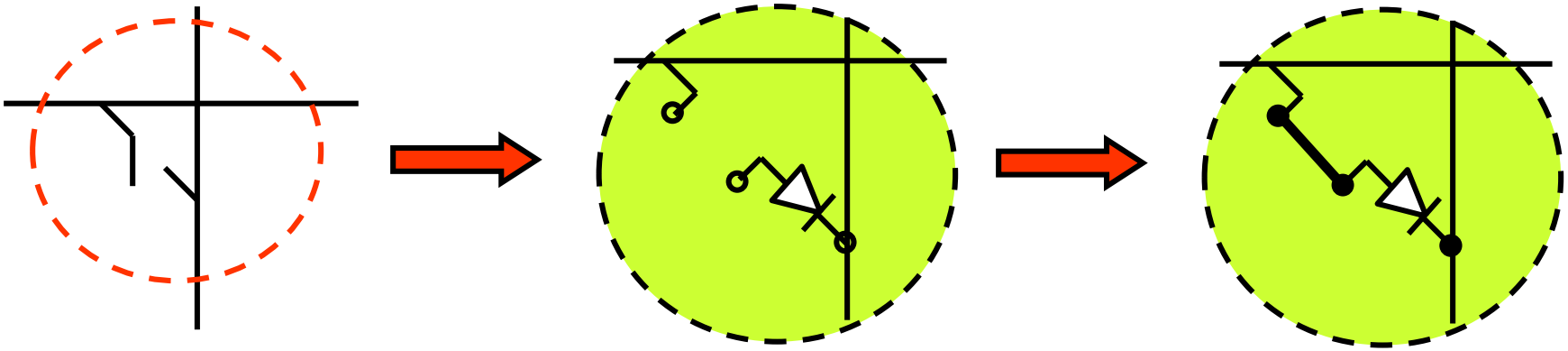
Le ROM come circuiti di memoria (2)

- Chiudere un contatto equivale dunque a **memorizzare** un «1» piuttosto che uno «0»
  - A_0, A_{n-1}: bit d'indirizzo
  - D_0, D_{k-1}: bit di dato (segnali d'uscita, in quanto *non scrivibili*)
- ROM come circuito di **memoria** in grado di memorizzare 2^n dati, ciascuno di k bit (tipicamente, k=8=1 byte, e 2^n = 1K, 2K, 4K, 8K, ..)
- Una ROM in grado di memorizzare N dati (ciascuno a k bit) sarà dunque indirizzabile mediante $n=\log_2 N$ bit d'indirizzo



La programmazione delle ROM

- I contatti vengono realizzati dal costruttore nell'ultimo stadio del processo di fabbricazione del circuito su ordine dell'acquirente:

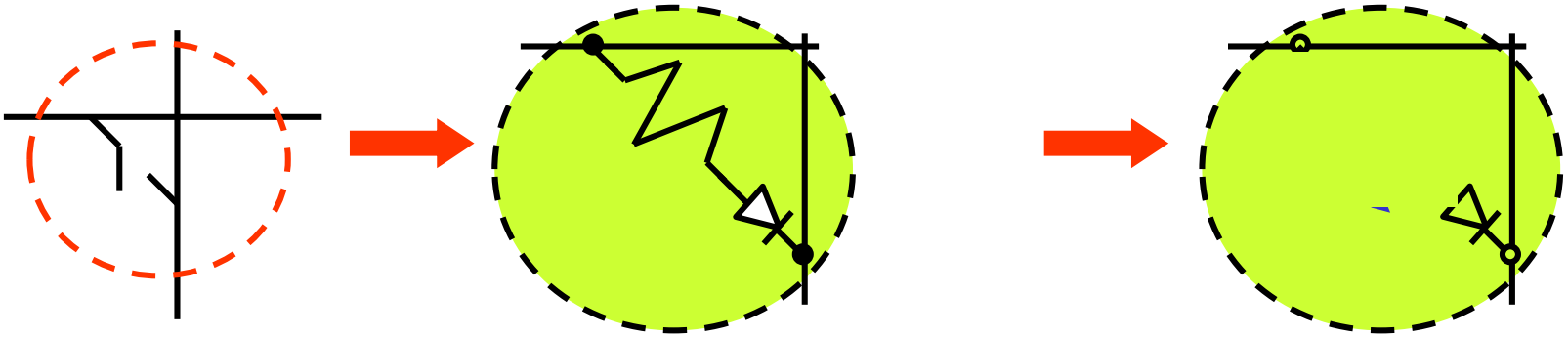


- Le ROM sono programmabili una sola volta (OTP: one time programmable).

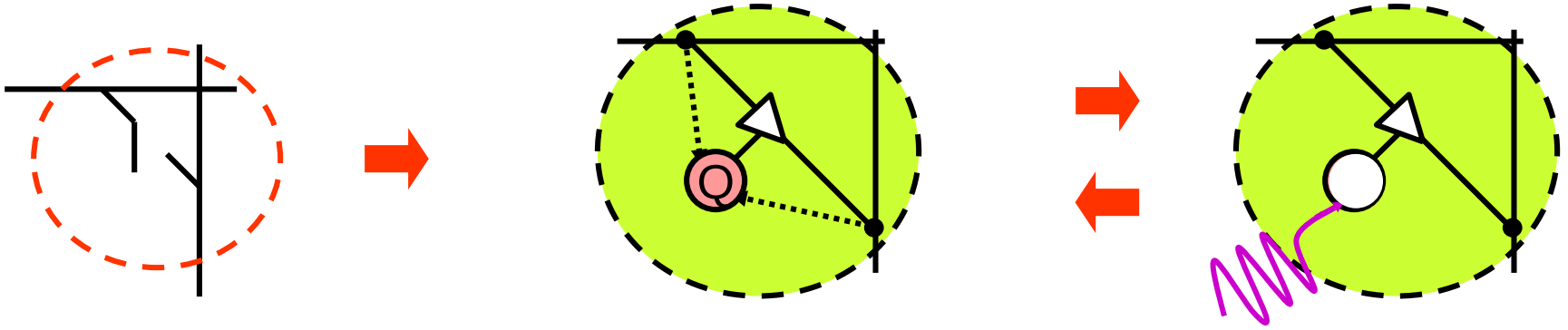
→ memorie a sola lettura (Read Only) non volatili

- Memorizzazione di informazioni che devono permanere quando il sistema non è alimentato e che non non cambiano durante il funzionamento (es. programma di un sistema «embedded», BIOS del PC, ..)
- Limiti: non riprogrammabili -> non adatte allo sviluppo, necessari ancora grossi volumi di produzione

PROM e EPROM



• **PROM** (Programmable ROM): La programmazione viene effettuata dall'utente mediante un'apposita apparecchiatura che consente di "bruciare" selettivamente i fusibili inseriti dal costruttore.
•Dispositivi programmabili una sola volta



- **EPROM** (Erasable PROM): Programmazione effettuata dall'utente mediante **un'apposita apparecchiatura**, cancellazione ottenuta convogliando luce ultravioletta sui terminali di comando
- Dispositivi riprogrammabili (a fronte di un costo maggiore di quello delle PROM)

Memorie a sola lettura cancellabili elettricamente

- **EEPROM** (*Electrically Erasable* PROM) : si programmano e cancellano byte-per-byte tramite segnali elettrici e **senza rimuovere il dispositivo dalla piastra stampata**.

- **FLASH-EPROM**: si programmano/cancellano elettricamente direttamente sulla piastra. **La cancellazione è più veloce rispetto alle EEPROM**: con un'unica operazione è possibile cancellare l'intero dispositivo oppure uno o più "settori".

TMS29F010
131072 BY 8-BIT
FLASH MEMORY
SMJS840A – NOVEMBER 1997 – REVISED JUNE 1998

memory sector architecture

	A16	A15	A14	Address Range
Sector 0	0	0	0	00000h – 03FFFh
Sector 1	0	0	1	04000h – 07FFFh
Sector 2	0	1	0	08000h – 0BFFFh
Sector 3	0	1	1	0C000h – 0FFFFh
Sector 4	1	0	0	10000h – 13FFFh
Sector 5	1	0	1	14000h – 17FFFh
Sector 6	1	1	0	18000h – 1BFFFh
Sector 7	1	1	1	1C000h – 1FFFFh

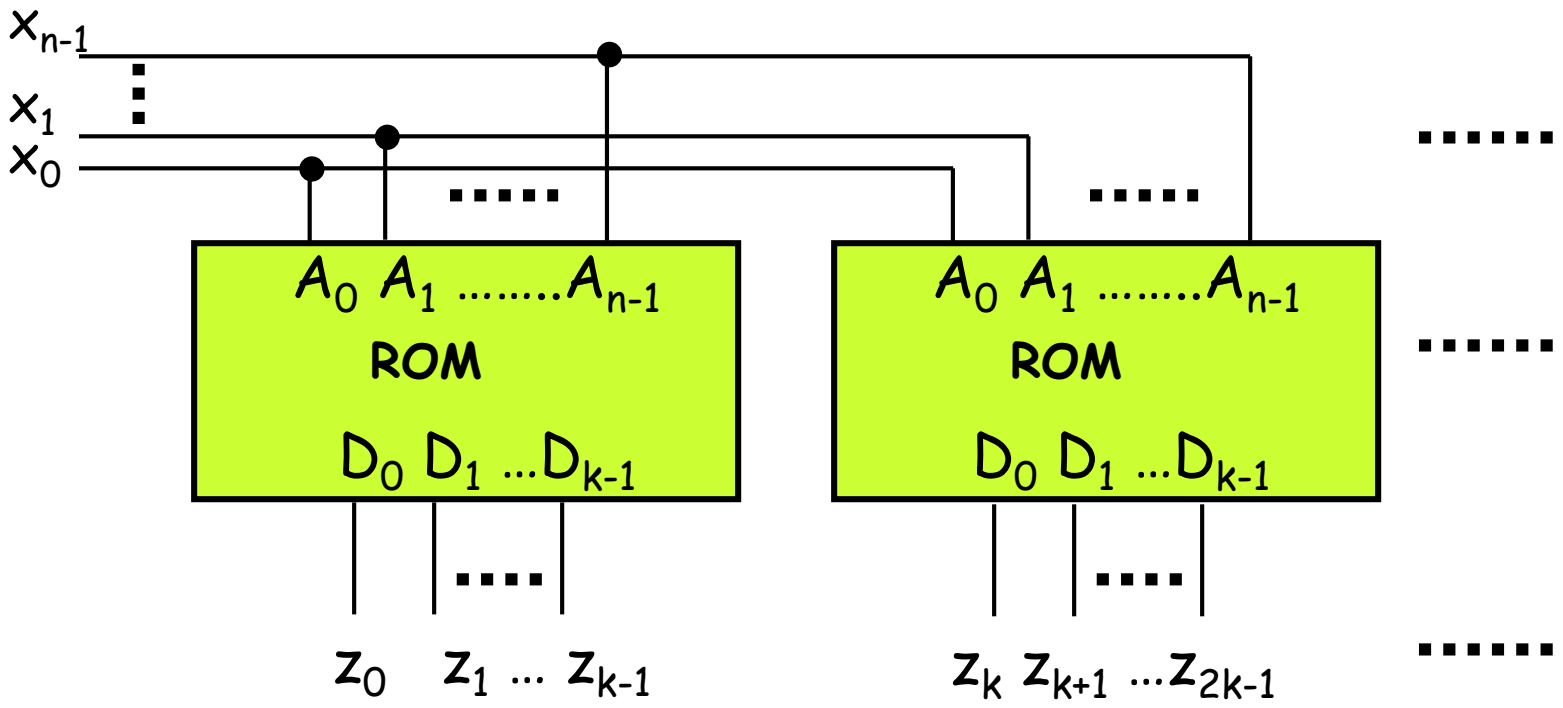
Memorie non volatili, a sola lettura e programmabili

Tipo	Proprietà				
ROM	n.v.	off line	una volta	costruttore	molte copie
PROM	n.v.	off line	una volta	utente	poche copie
EPROM	n.v.	off line	più volte	utente	prototipi
EEPROM, FLASH	n.v.	on line	più volte	utente	personale

Estensione del numero di uscite

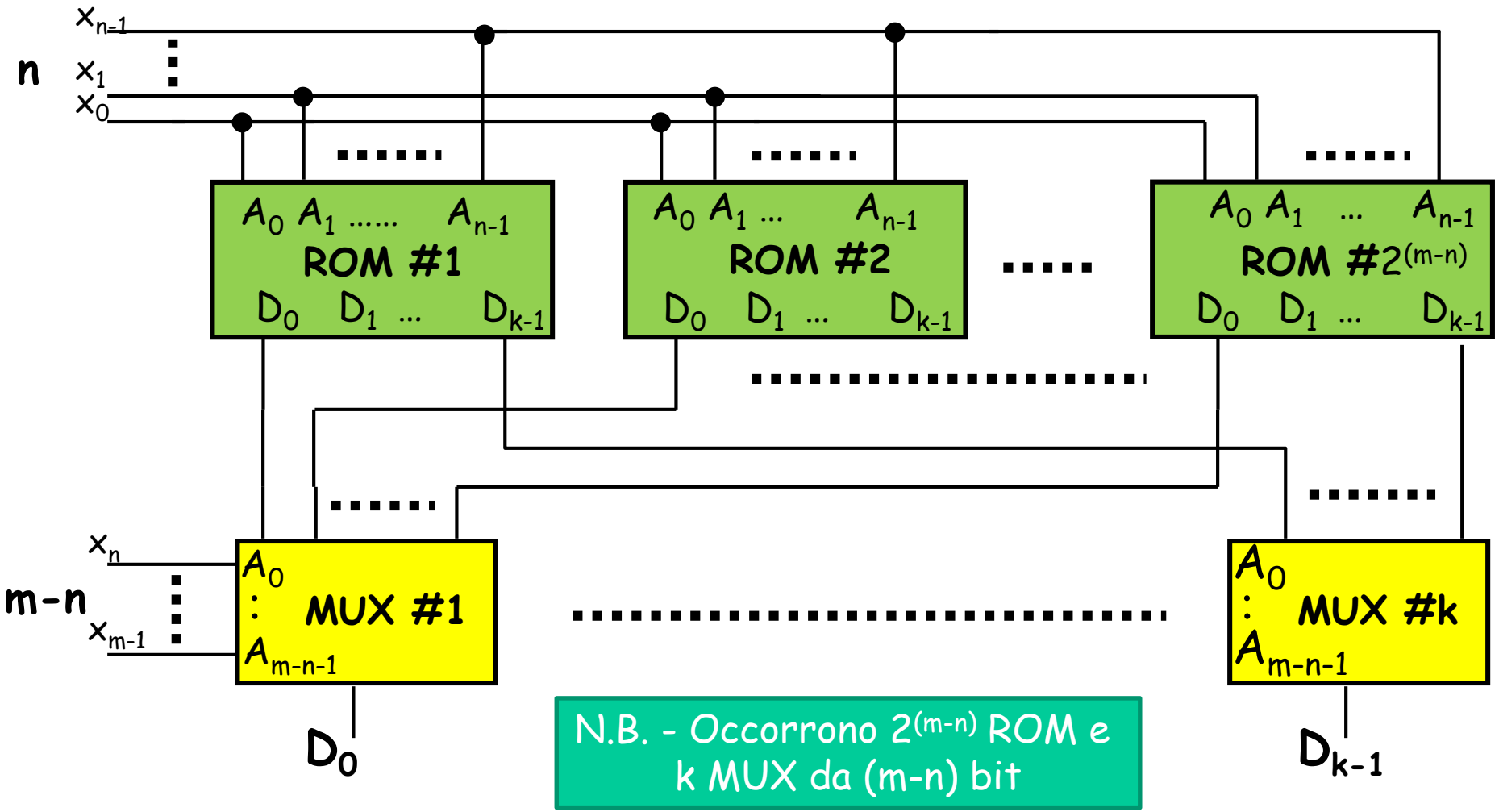
- **Problema 1**: limite al numero di morsetti di uscita della ROM
- Ciò limita il numero di funzioni realizzabili da ciascuna ROM o, equivalentemente, il parallelismo dei dati in essa memorizzati
- Soluzione: collegamento in **parallelo** di più ROM

- Es.: ROM con n ingressi e k uscite, da interfacciare su un *bus dati* con parallelismo > k



Estensione del numero di ingressi

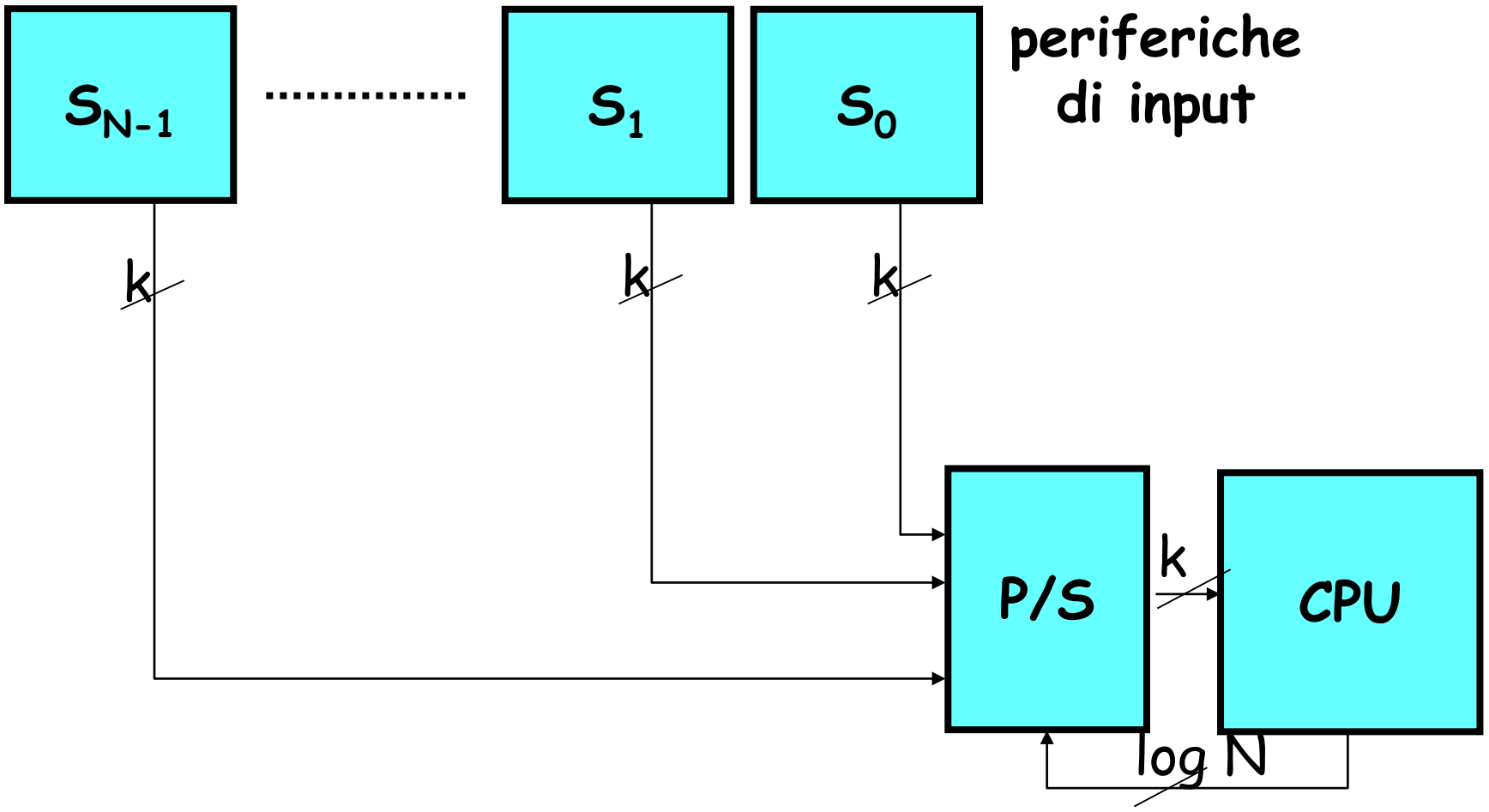
- **Problema 2**: limite al numero di morsetti di ingresso della ROM
- Ciò limita la capacità in termini di celle contenute in ciascuna ROM
- Soluzione: collegamento in **serie** di più ROM
- Es.: utilizzare più ROM con 2^n celle da k bit ciascuna per realizzare un sistema di memoria complessivo da 2^m celle -> m > n!



N.B. - Occorrono $2^{(m-n)}$ ROM e k MUX da (m-n) bit

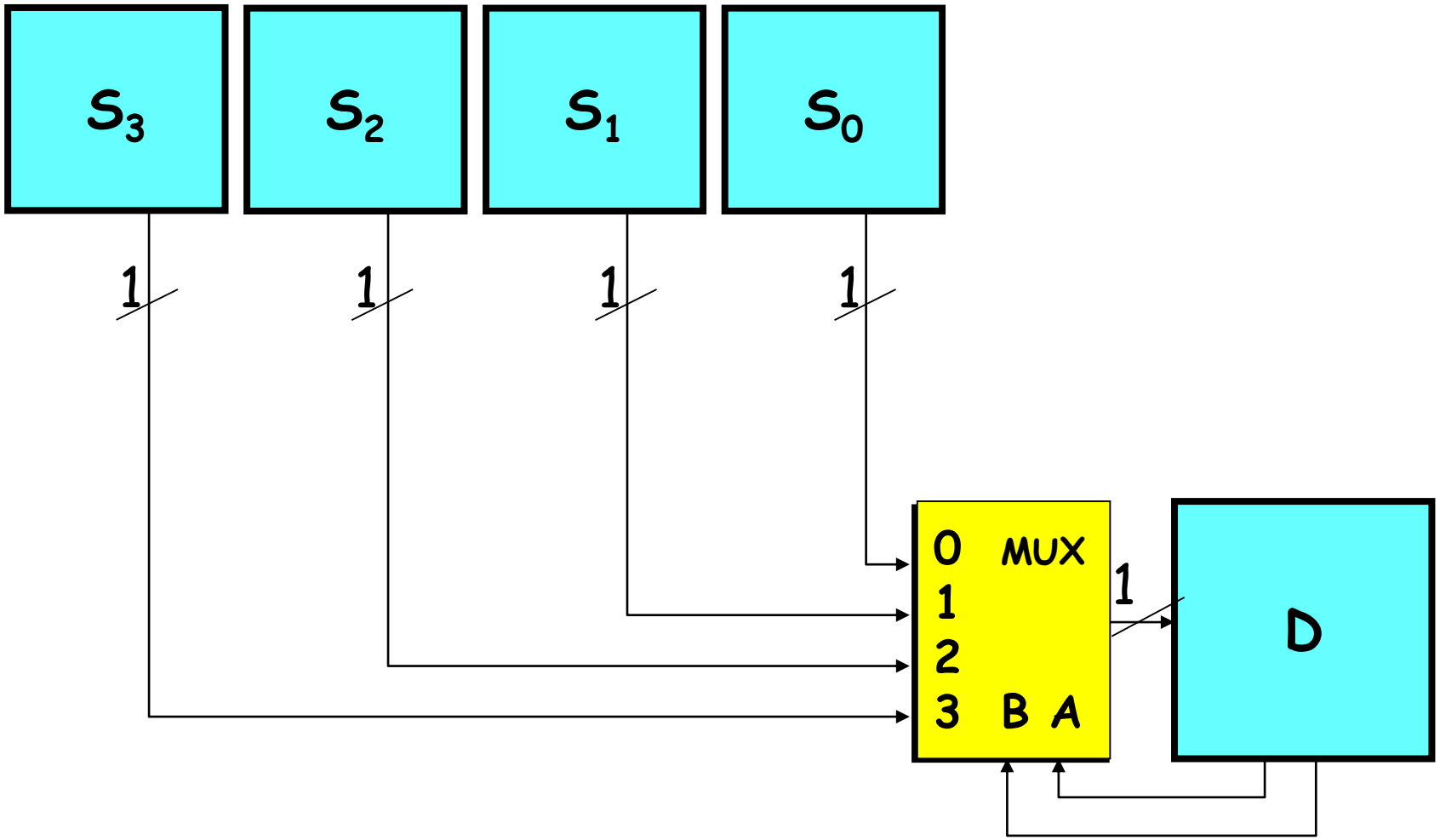
Data Selector

- Un problema architettonico più generale: mettere in comunicazione N sorgenti con 1 destinazione (es.: N ROM con CPU)
  La selezione impiega **Nxk** segnali (k: parallelismo dati da trasferire)



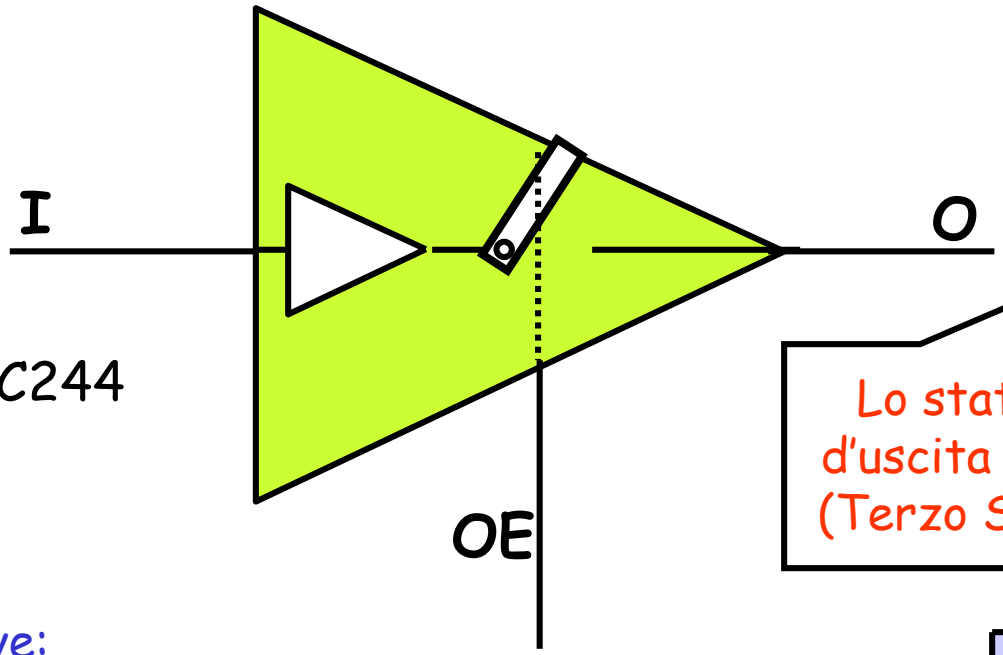
Esempio: selezione con MUX

- Una soluzione è rappresentata dall'utilizzo di **k MUX a N vie**
- Esempio con **N=4** e **k=1**:



Amplificatore a 3 stati d'uscita

- Modellabile come un gate Buffer con a valle un interruttore elettronico azionato da un comando detto **Output Enable** (OE)

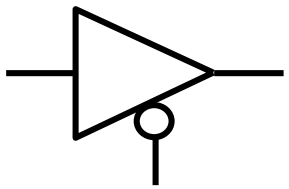


SN74AC244

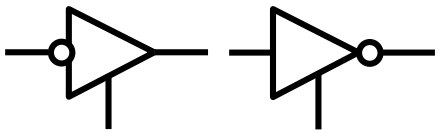
OE	I	O
H	L	L
H	H	H
L	X	Z

Lo stato elettrico del segnale d'uscita è indefinito o fluttuante (Terzo Stato, Stato di Alta Imp.)

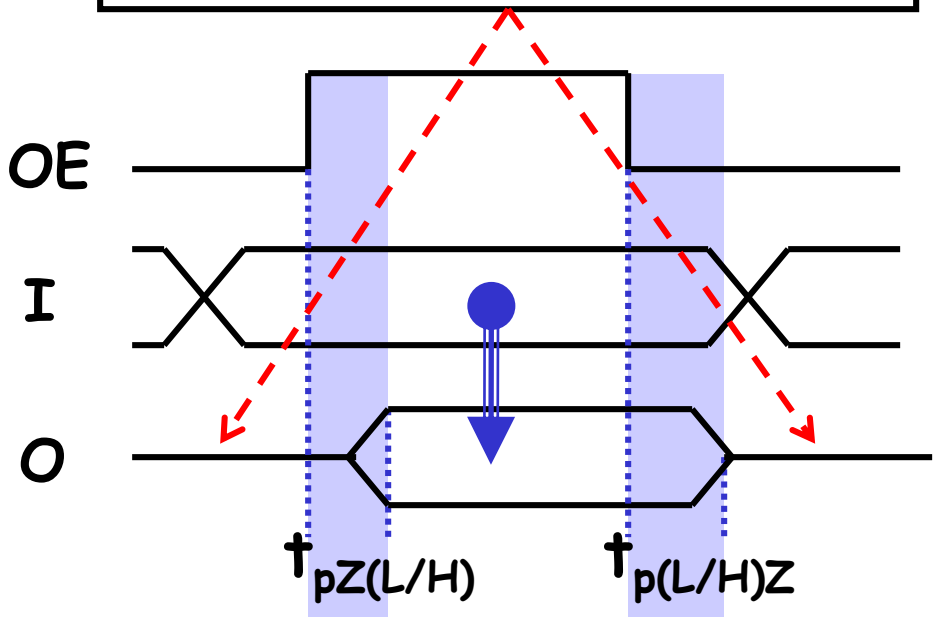
Alternative:



Alta impedenza con OE=H

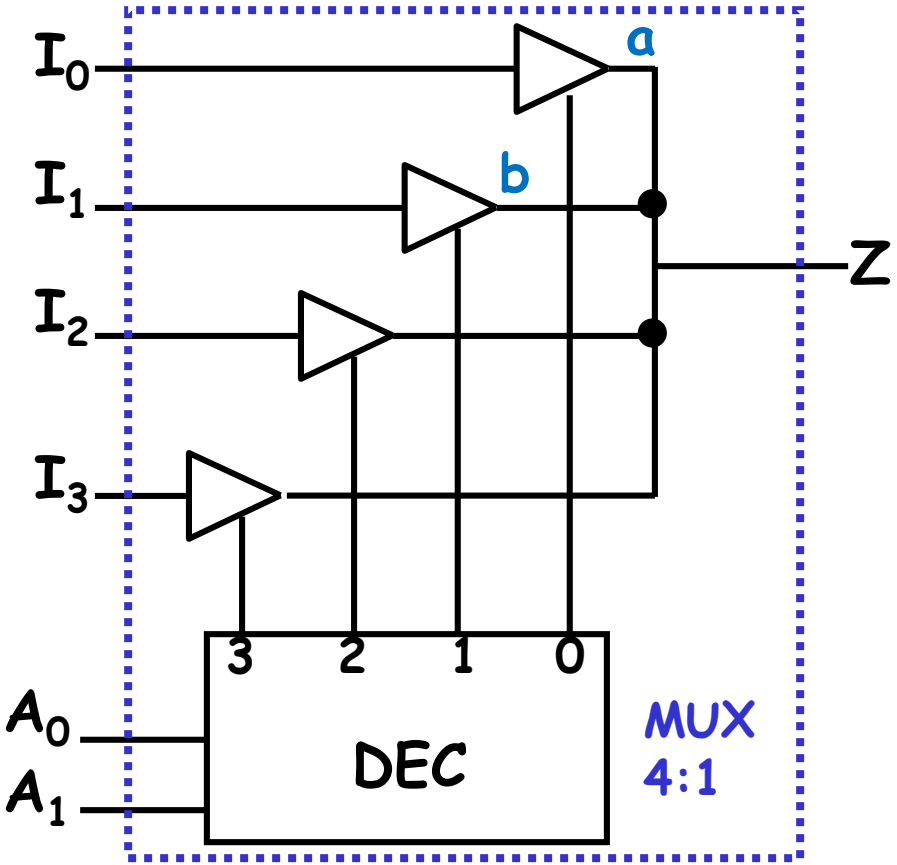


Segnale di ingresso negato (NOT + interrutt.)



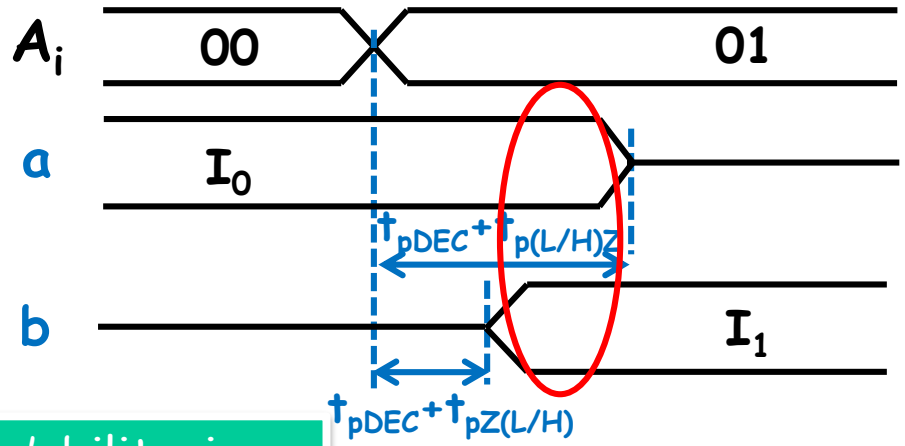
MUX con amplificatori 3-state

- Mediante 3-state è possibile realizzare una versione alternativa del MUX (selettore)



In ogni istante di tempo non deve esserci più di un 3-state abilitato, per evitare situazioni di **corto circuito** tra l'uscita e più di un ingresso (*conflitto elettrico*), che causerebbe possibili malfunzionamenti del sistema

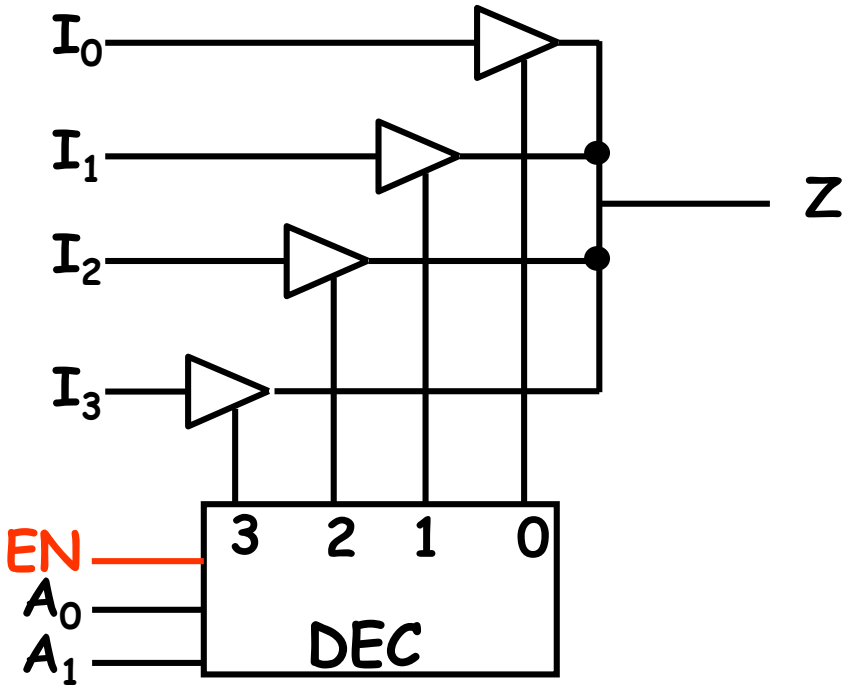
Ma: nei 3-state, il tempo di risposta all'abilitazione (t_{pZH}, t_{pZL}) è **inferiore** a quello necessario per il ritorno nel terzo stato (t_{pHZ}, t_{pLZ})

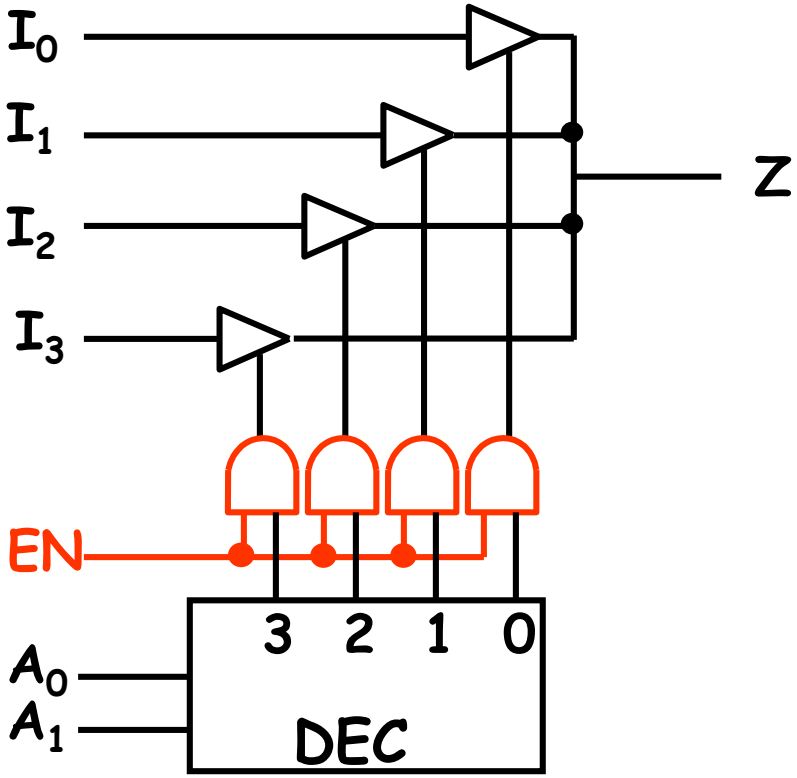


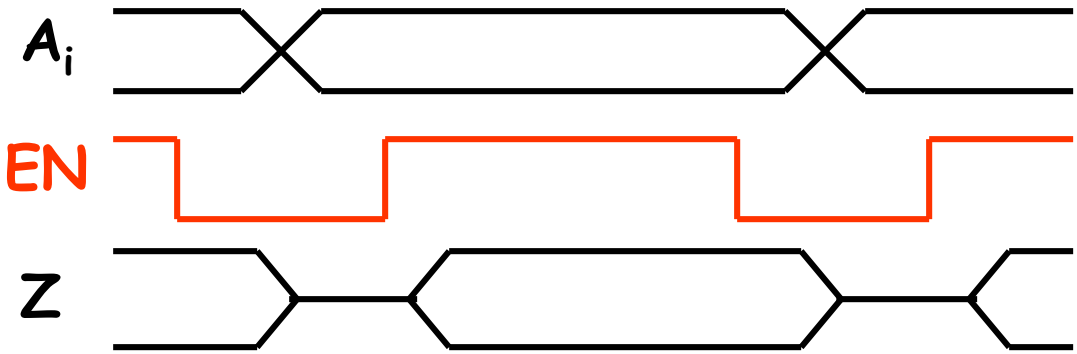
Occorre una **pausa** tra un'abilitazione e l'altra (v. prossima slide)

MUX con amplificatori 3-state

- Utilizzo di un segnale di ENABLE che viene tenuto basso tra la disattivazione di una via e l'attivazione della successiva

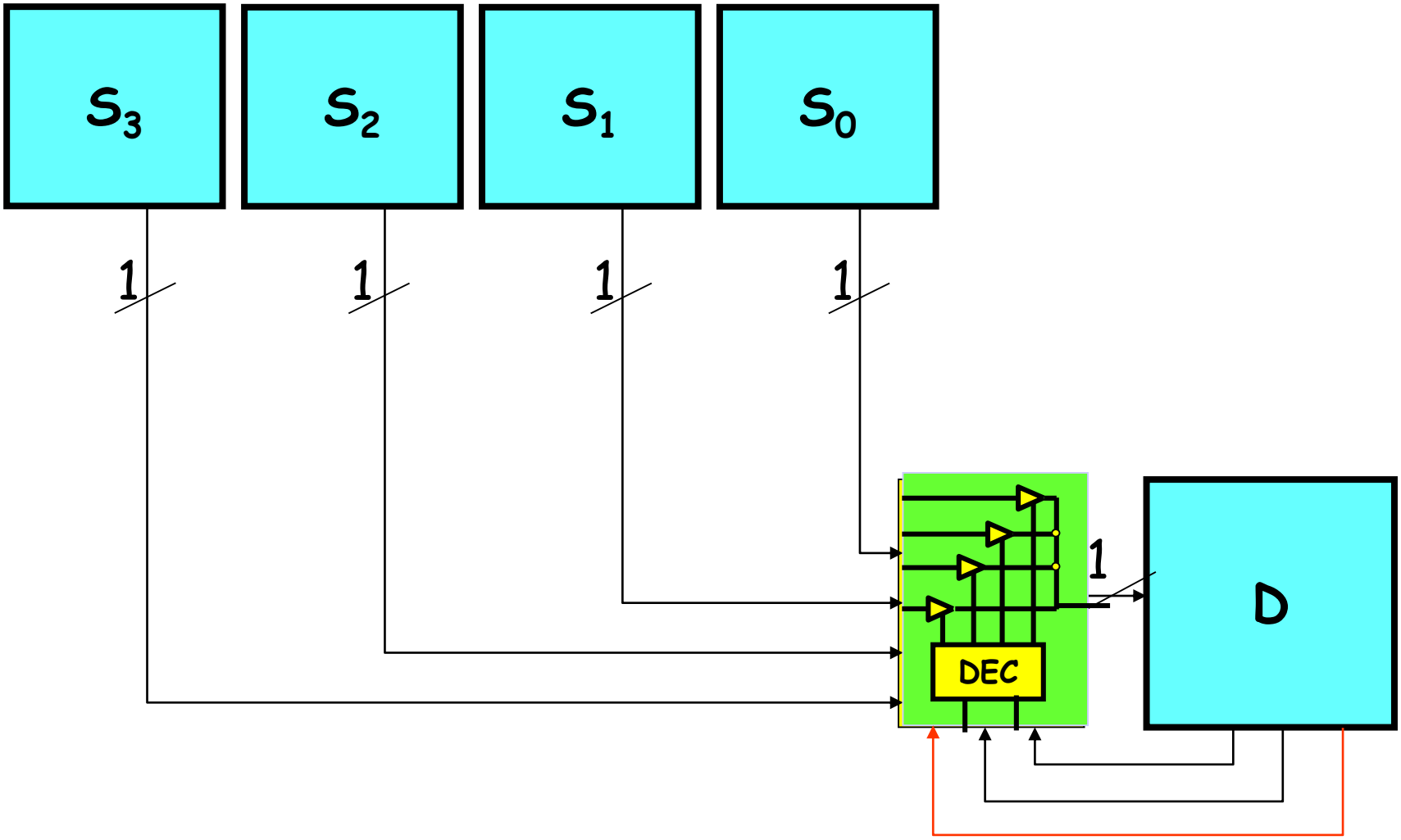






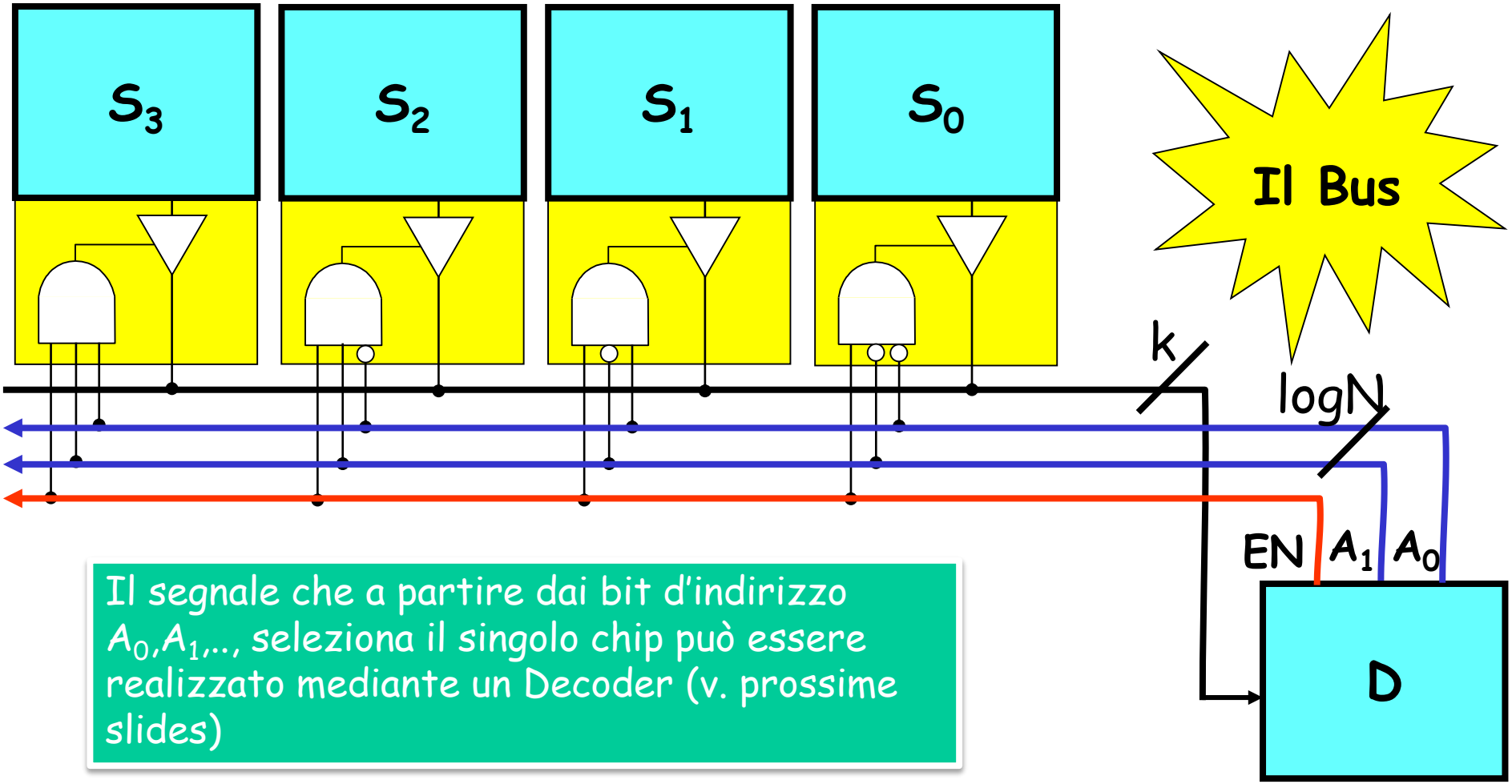
Selezione con amplificatori a 3 stati

- Il vantaggio di questa variante del MUX è che può essere «distribuito» tra le sorgenti $S_0..S_N$



Selezione con amplificatori a 3 stati

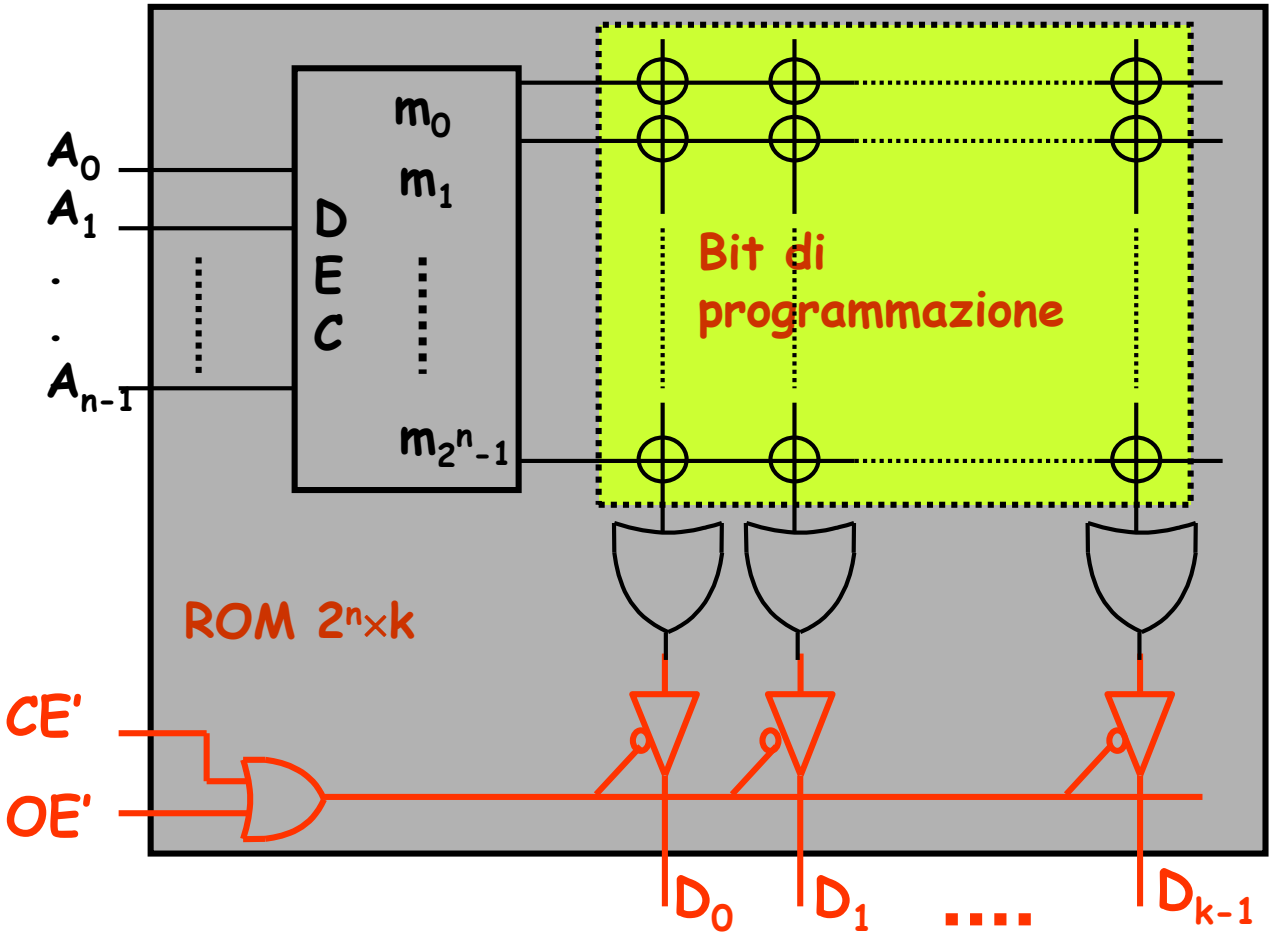
- Si ottiene così una riduzione del numero di segnali per il collegamento da Nxk a **(k+logN+1)**
  - **k: bus dati**
  - **logN : bus indirizzi**
  - **1: segnale di controllo (EN)**



Il segnale che a partire dai bit d'indirizzo A_0,A_1,.., seleziona il singolo chip può essere realizzato mediante un Decoder (v. prossime slides)

Stadio di uscita di una ROM

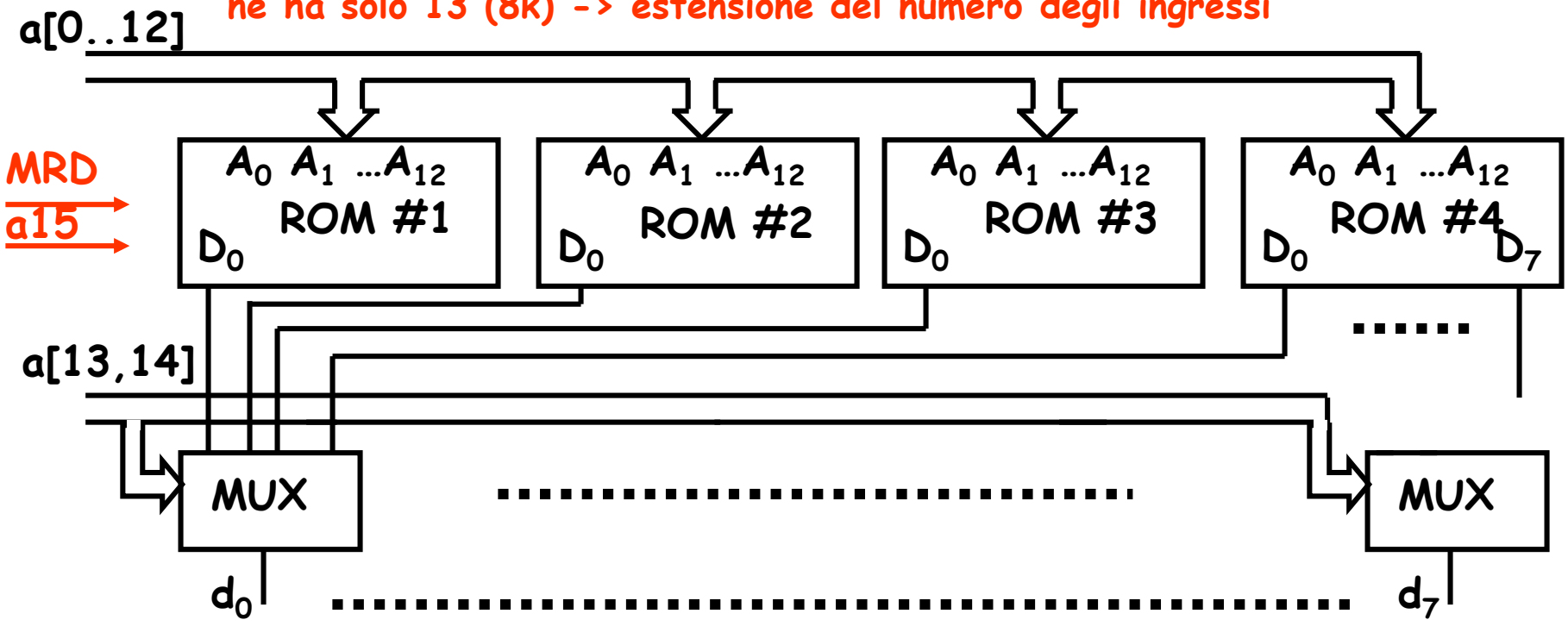
- Il segnale EN e il segnale che seleziona il singolo dispositivo sono generalmente segnali di input per il chip di ROM in quanto entrambi generati dalla destinazione (es. CPU)
- Il primo è indicato come **Output Enable** (**OE**, il medesimo per tutti i dispositivi ROM), il secondo come **Chip Enable** (**CE**, specifico per ogni chip e generabile a partire dai segnali d'indirizzo mediante **DECODER**)
- CE e OE sono tipicamente *attivi bassi* -> utilizzato un OR al posto di un AND



Esercizio: progetto di un banco di ROM

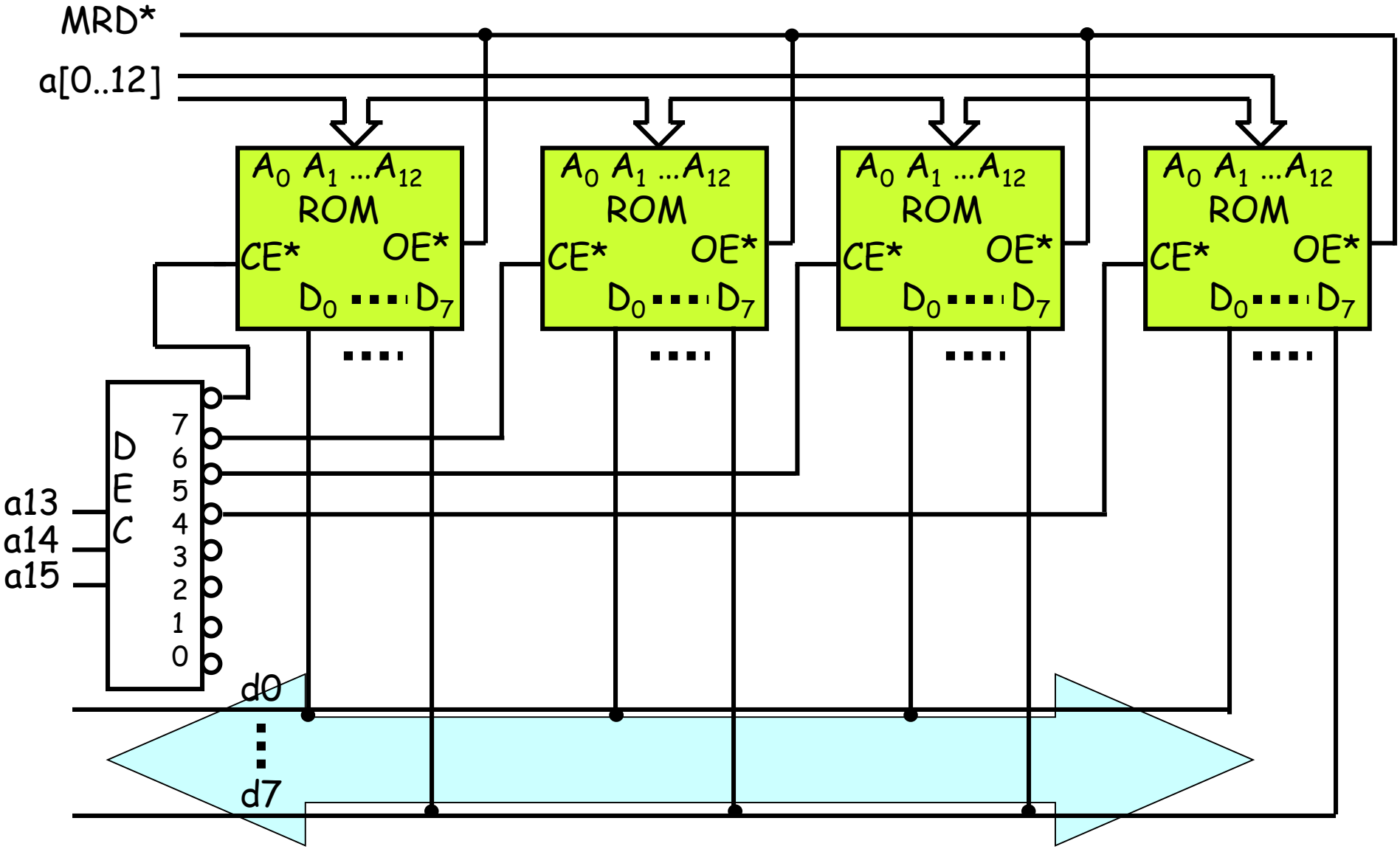
Supponiamo di voler connettere 32Kbyte di ROM ad una CPU con 16 bit di indirizzo ($A_0..A_{15}$) ed 8 bit di dato e di avere a disposizione dispositivi 4 ROM da 8K. Supponiamo inoltre che la CPU, con spazio di indirizzamento totale a 64k, veda il banco di ROM nella parte alta del suo spazio di indirizzamento (8000H - FFFFH, cioè A_{15}=1).
Si vuole ottenere una soluzione che non utilizzi un MUX per ogni segnale di uscita (utilizzo di ROM con tri-state).

15 bit di indirizzo (32k) totali (+ A15 sempre =1), ma ciascuna ROM ne ha solo 13 (8k) -> estensione del numero degli ingressi



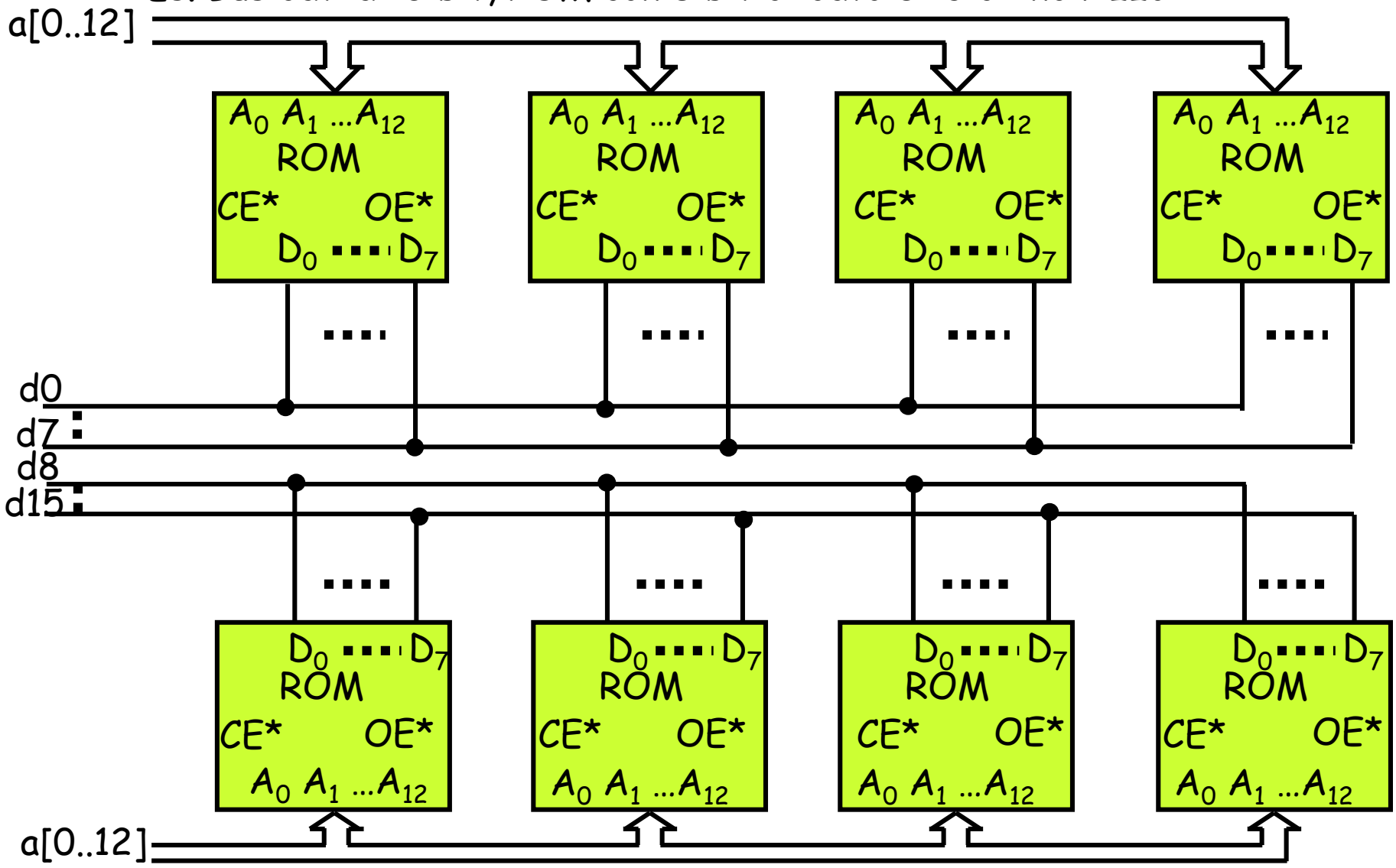
Esercizio: progetto di un banco di ROM

- Utilizzo di ROM con segnale di CE e OE e di un Decoder per generare i segnali di CE (famiglia TTL basata su NAND con uscite negate)



Caso più generale: numero di ingressi e di uscite insufficienti

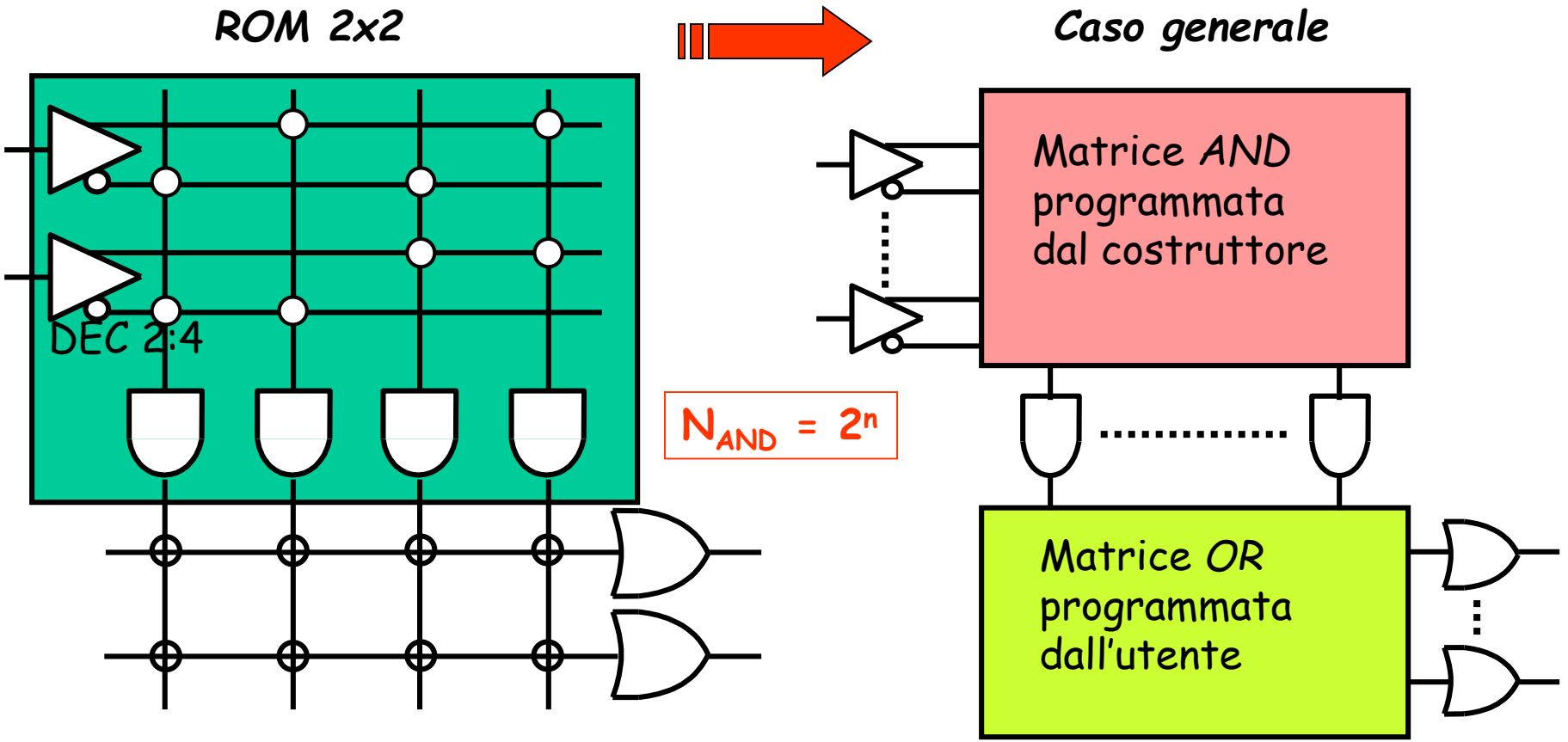
- Es. Bus dati a 16 bit, ROM con 8 bit di dato e 13 di indirizzo




PLA, PAL, FPGA

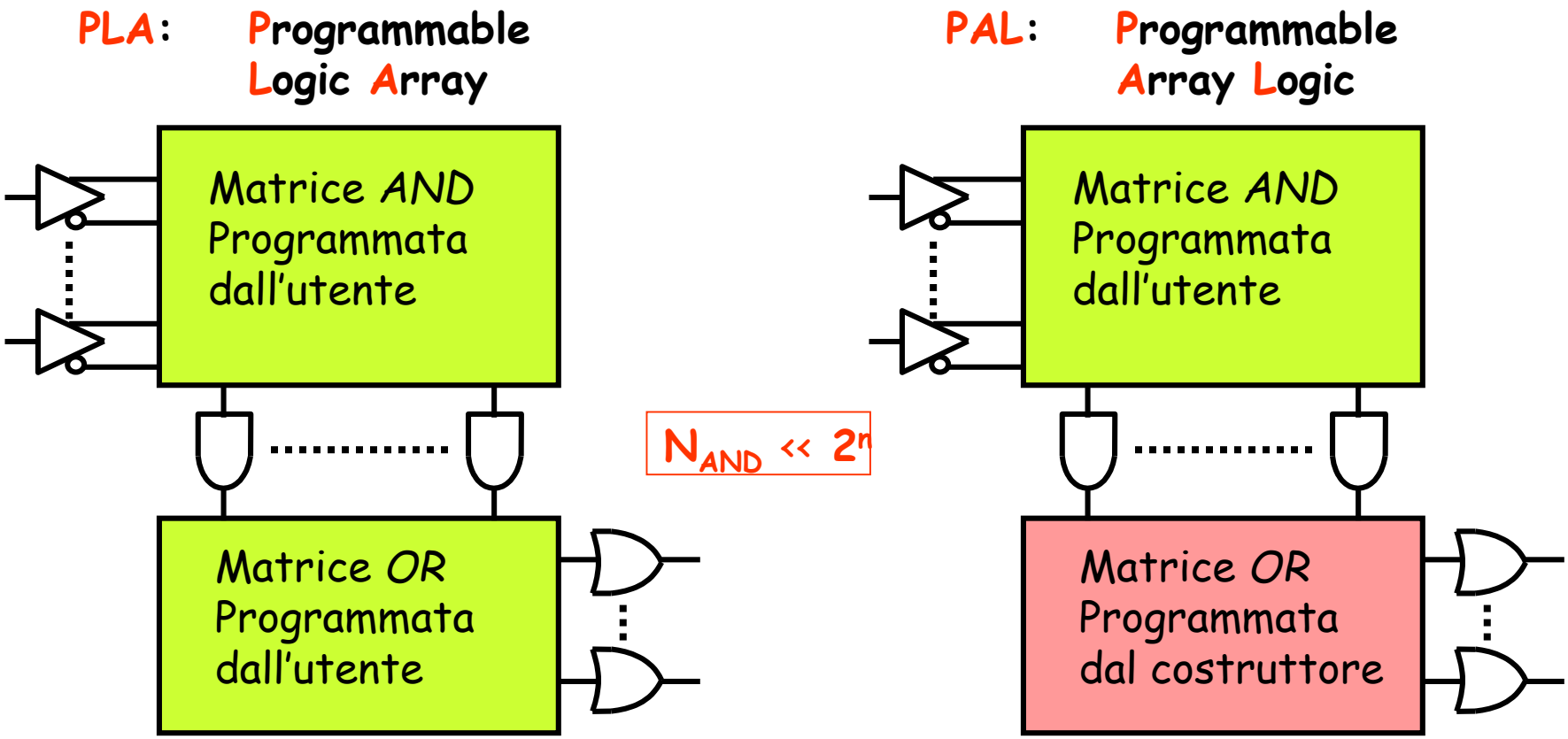
Rappresentazione di una ROM in termini di matrici AND e OR

- In una ROM la matrice degli AND realizza tutti i possibili **mintermini** a partire dagli ingressi (un mintermine per ciascun AND)
- Può essere così utilizzata per sintetizzare una espressione generale SP, ma richiede un numero di AND pari a 2^n



PLA e PAL

- Nelle PLA e nelle PAL la matrice degli AND è programmabile da utente
- Tale matrice può essere così utilizzata per realizzare gli **implicanti** di una espressione normale SP, riducendo notevolmente il numero di AND presenti
- **PLA**: più flessibilità di programmazione (es. gli implicanti che sono «comuni» a più funzioni possono essere riutilizzati, v. prossimo esempio)
- **PAL**: progettazione più vincolata ma più **semplice**



Esempio: sintesi con PLA

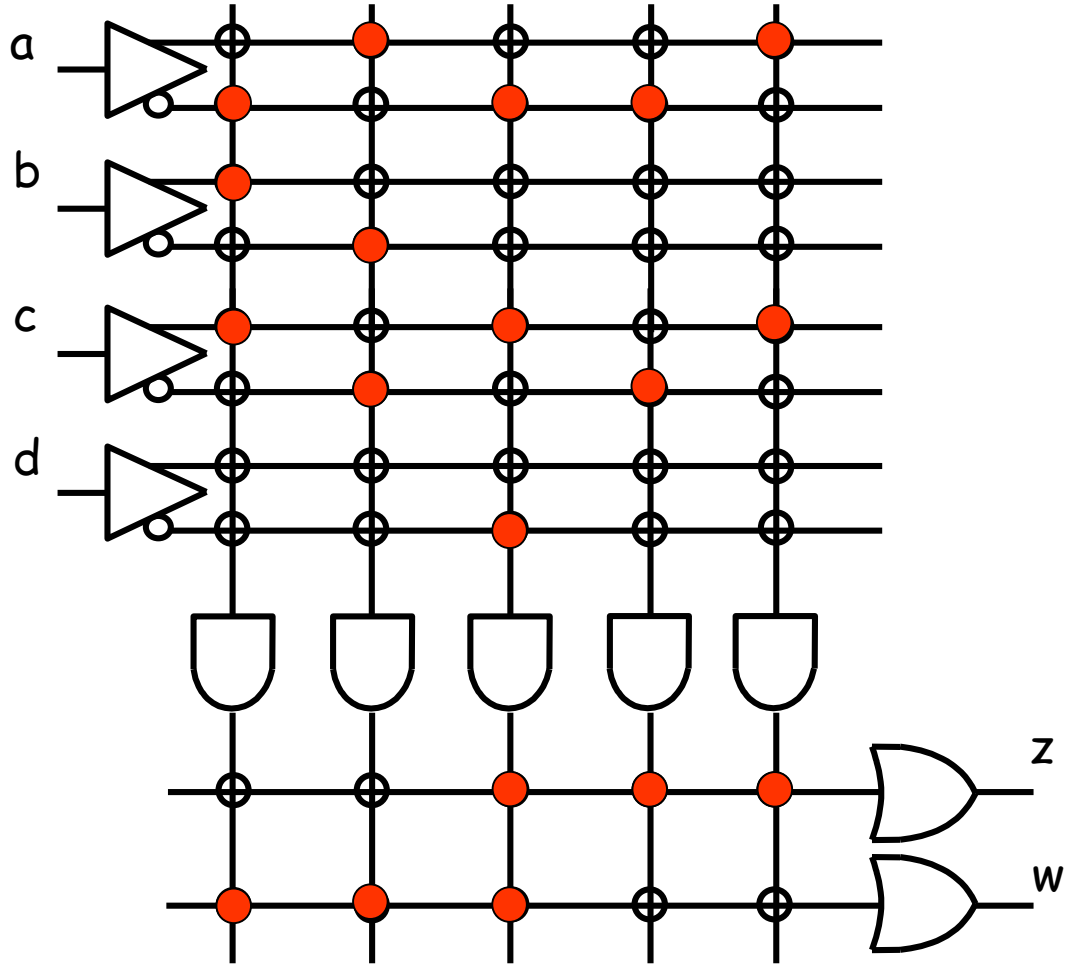
- Sintesi di due funzioni **w,z** a 4 variabili mediante PLA a 5 AND e 2 OR

a b \ cd	00	01	11	10
00	0	0	0	1
01	0	0	1	1
11	0	0	0	0
10	1	1	0	0

w

a b \ cd	00	01	11	10
00	1	1	0	1
01	1	1	0	1
11	0	0	1	1
10	0	0	1	1

z



- Con 5 AND a disposizione, scelgo una **copertura non minima** per la funzione **z** in modo da poter programmare la PLA riutilizzando il mintermine in comune tra le due funzioni (**a'cd'**)

Esempio:sintesi con PAL

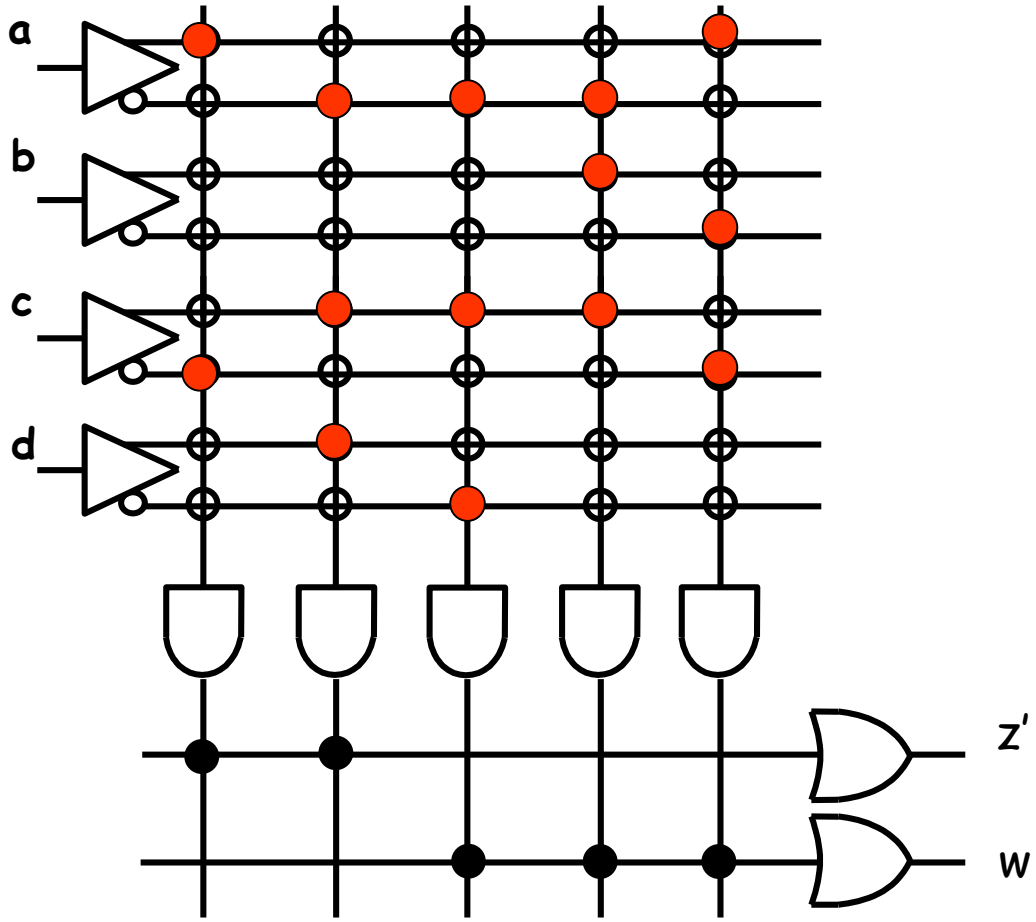
- Sintesi delle due funzioni **w,z** dell'esempio precedente mediante PAL a 5 AND e 2 OR

a b \ cd	00	01	11	10
00	0	0	0	1
01	0	0	1	1
11	0	0	0	0
10	1	1	0	0

w

a b \ cd	00	01	11	10
00	1	1	0	1
01	1	1	0	1
11	0	0	1	1
10	0	0	1	1

z

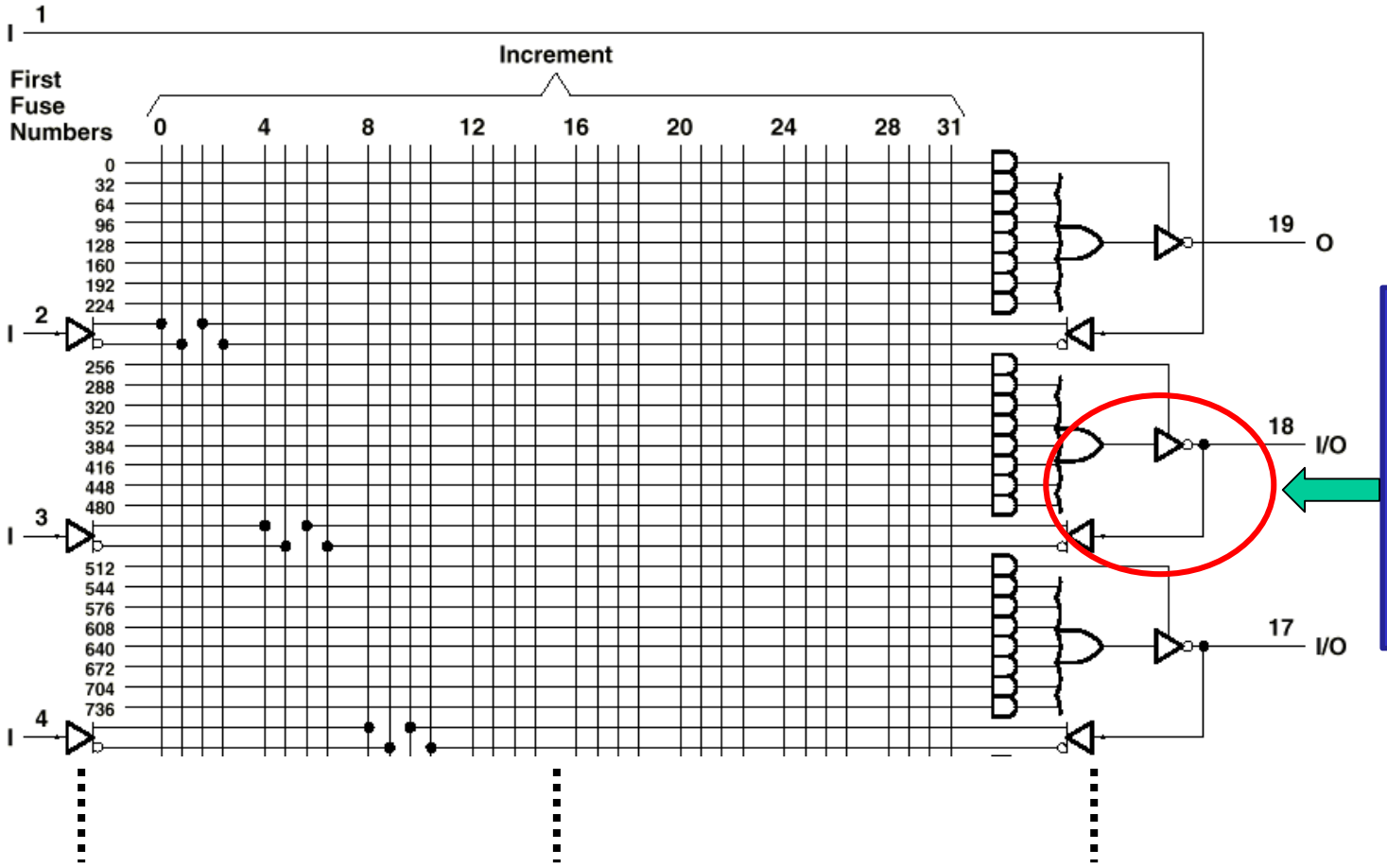


- In questo caso la matrice degli OR non è programmabile
- per gestire un numero insufficiente di AND sintetizzo la funzione complemento (ovvero realizzo z' anzichè z)

PAL con I/O programmabile (16L8)

- Problema : il numero di ingressi di una PAL è limitato
- Soluzione : esistono alcuni chip in cui è possibile programmare la funzionalità di I/O di alcuni pin che il dispositivo impiega per comunicare con l'esterno (ovvero trasformarli da pin d'uscita a pin d'ingresso e viceversa)

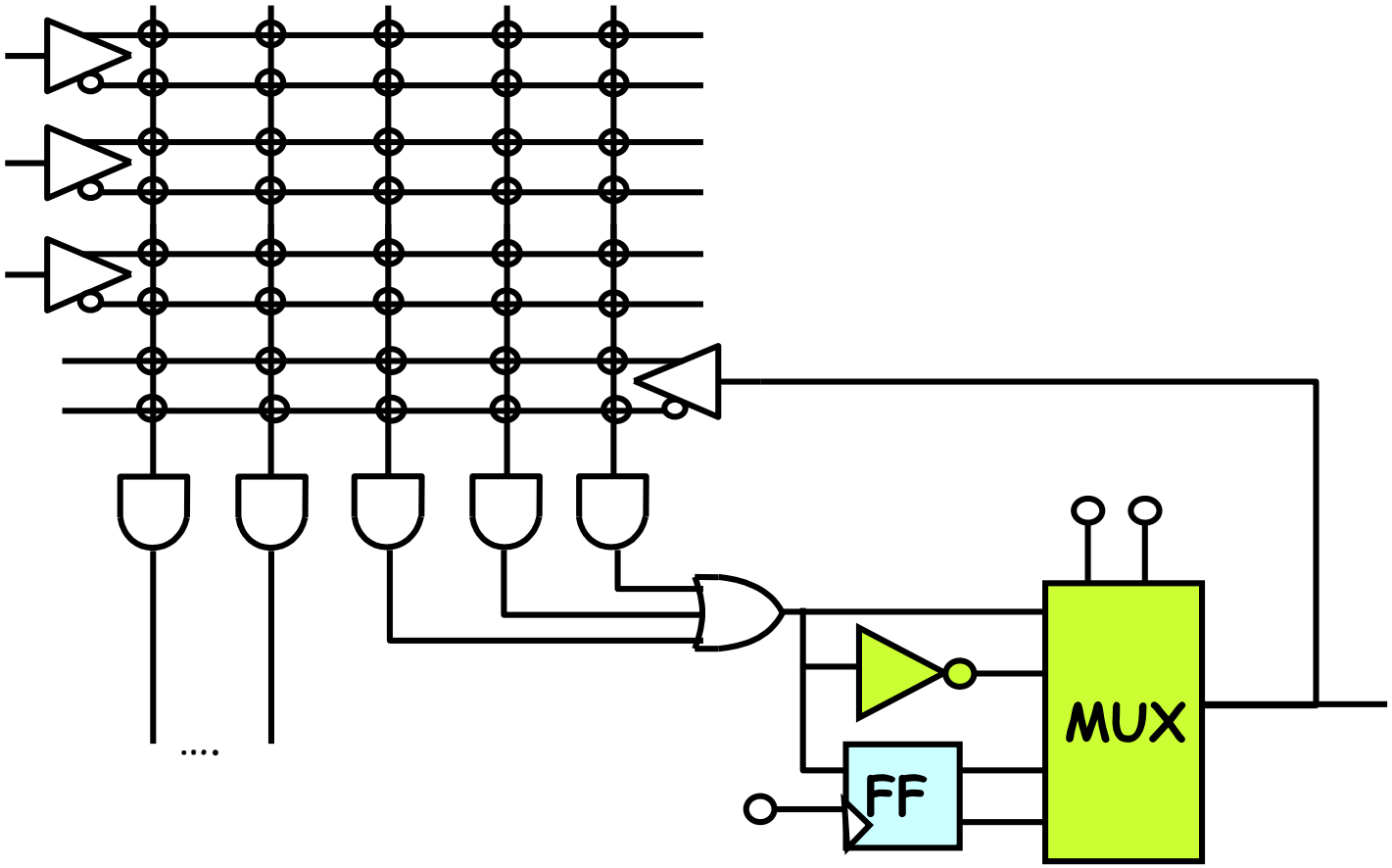
- Es: PAL con 16 ingressi e 8 uscite ***nominali***



Tramite amplificatori 3state, un pin d'uscita può essere riprogrammato per funzionare come pin d'ingresso

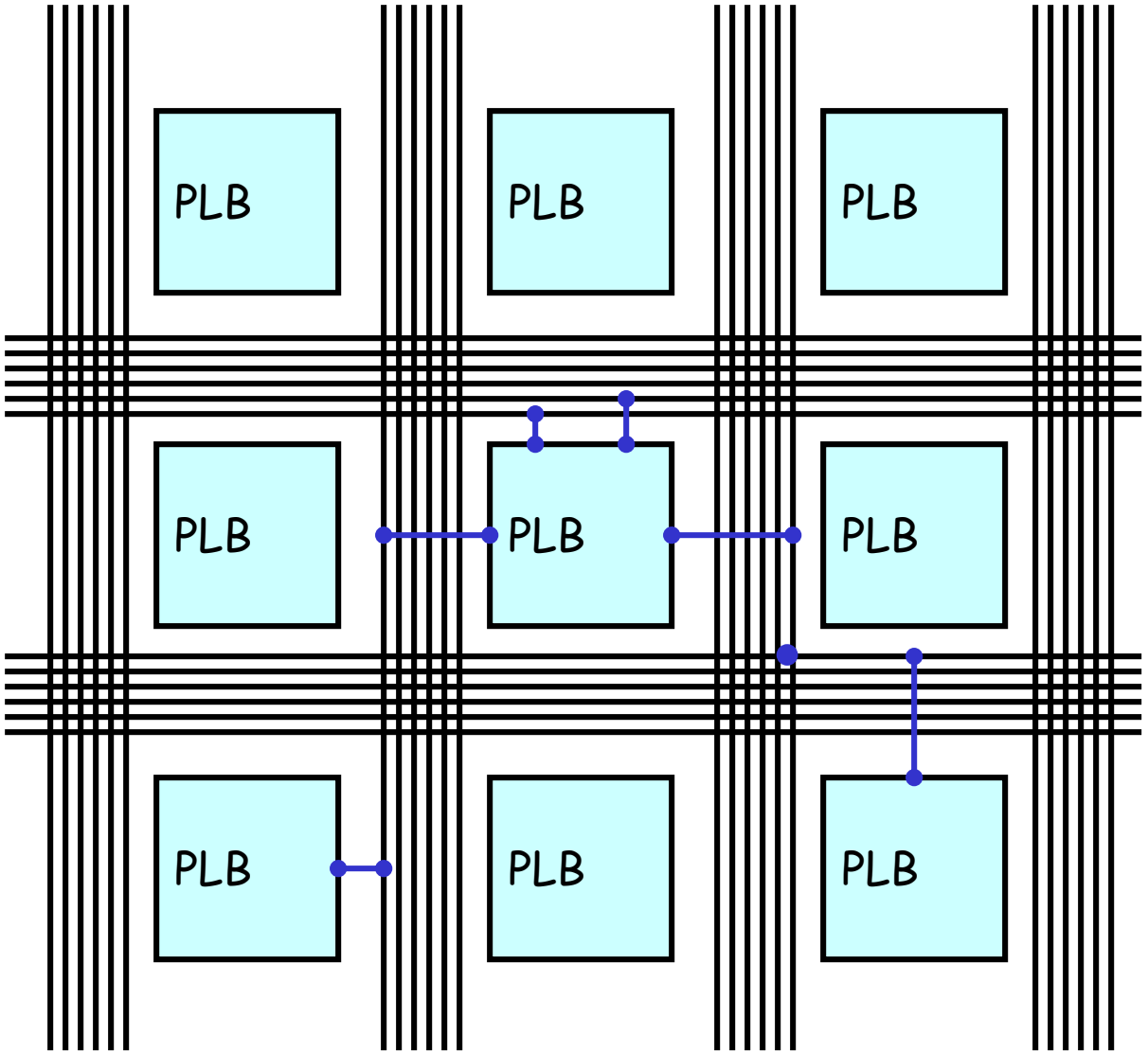
Complex Programmable Logic Device (CPLD)

- PAL e PLA sono esempi di **Programmable Logic Devices** (PLD)
- Dispositivi più avanzati: **Complex PLD, Field Programmable Gate Array (FPGA)**
- Presenza aggiuntiva di blocchi funzionali specializzati, presenza di elementi di **memoria** on-chip e possibilità di retroazione permettono di programmare non solo reti combinatorie ma anche reti sequenziali (che inizieremo ora a studiare)



CPLD, FPGA

- Utilizzo di unità base (Programmable Logic Blocks) ripetute e collegabili tra loro in modo programmabile



CAD (Xilinx, Altera Quartus)