

# Reti Combinatorie

Giovanni Stea

Reti Logiche

Corso di Laurea in Ingegneria Informatica, Universita' di Pisa

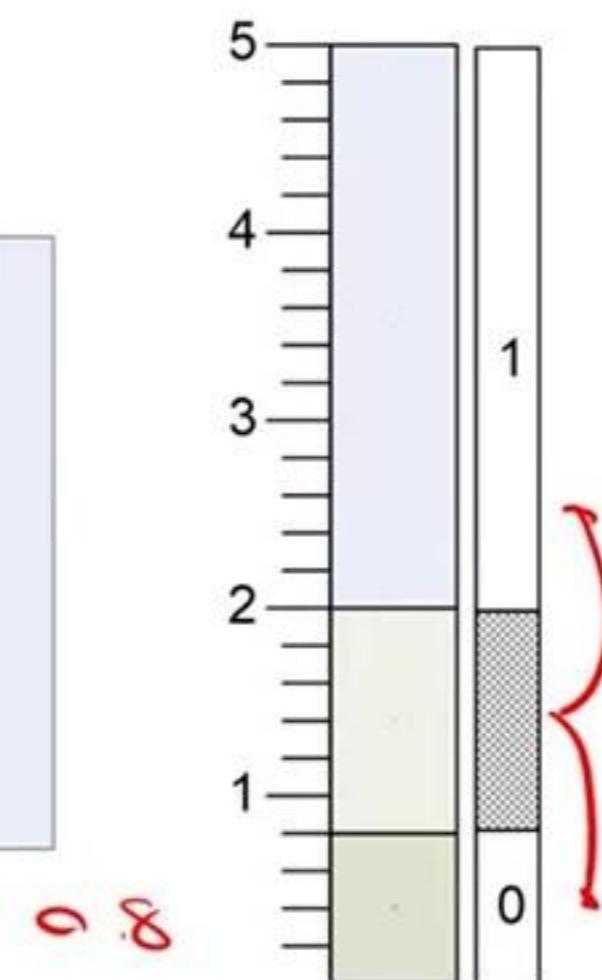
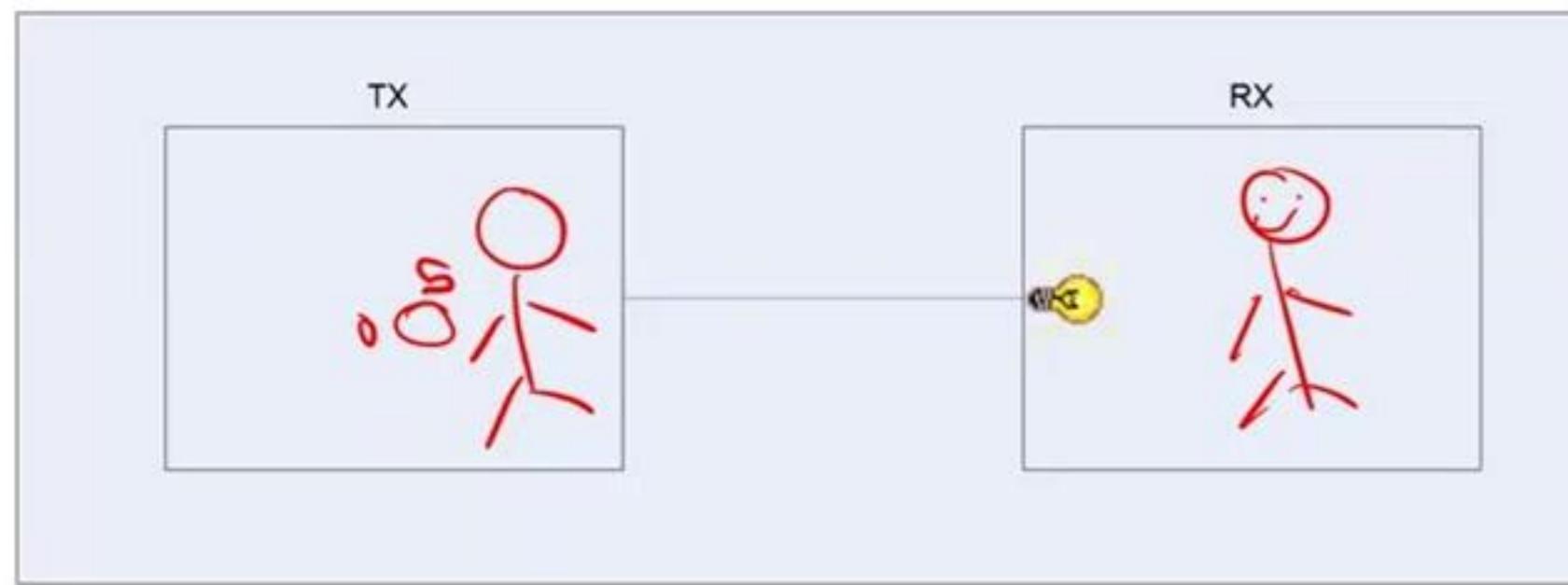
a.a. 2020/21

# Materiale di riferimento

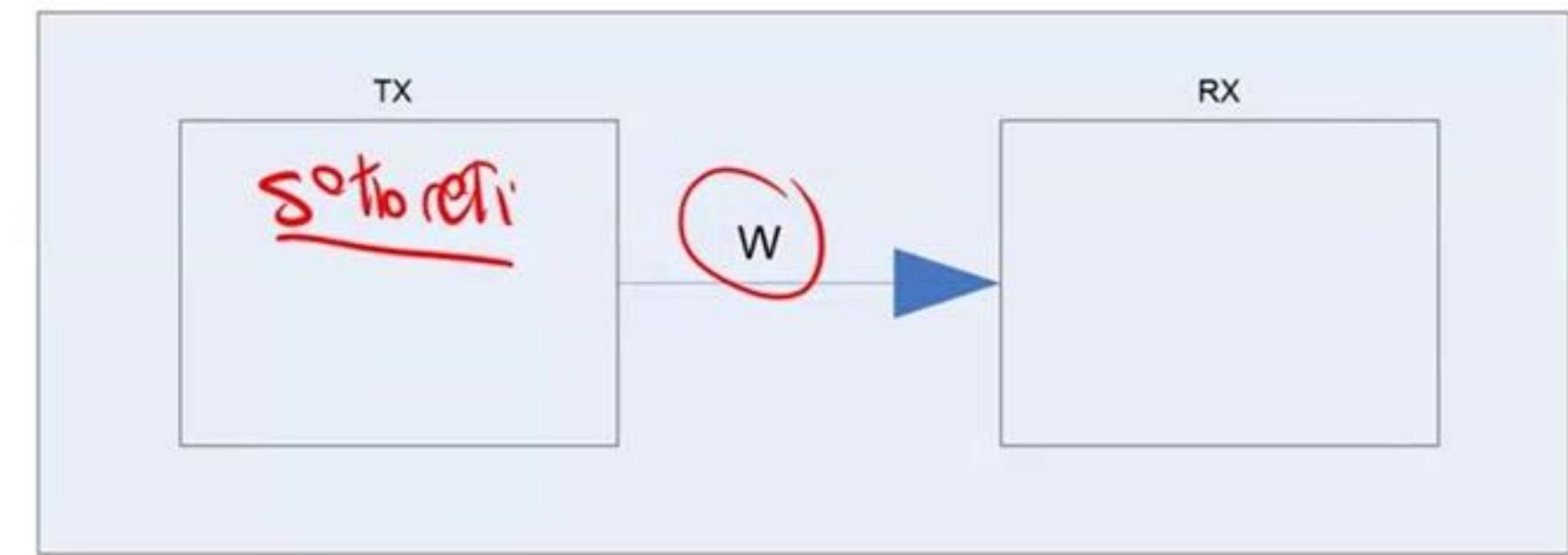
- Paolo Corsini, "Dalle porte AND, OR, NOT al Sistema calcolatore", edizioni ETS
- Appunti sulle reti combinatorie - versione 25/09/2020 (con esercizi svolti)

# Rete logica come modello astratto

- Sistema fisico



- Modello astratto



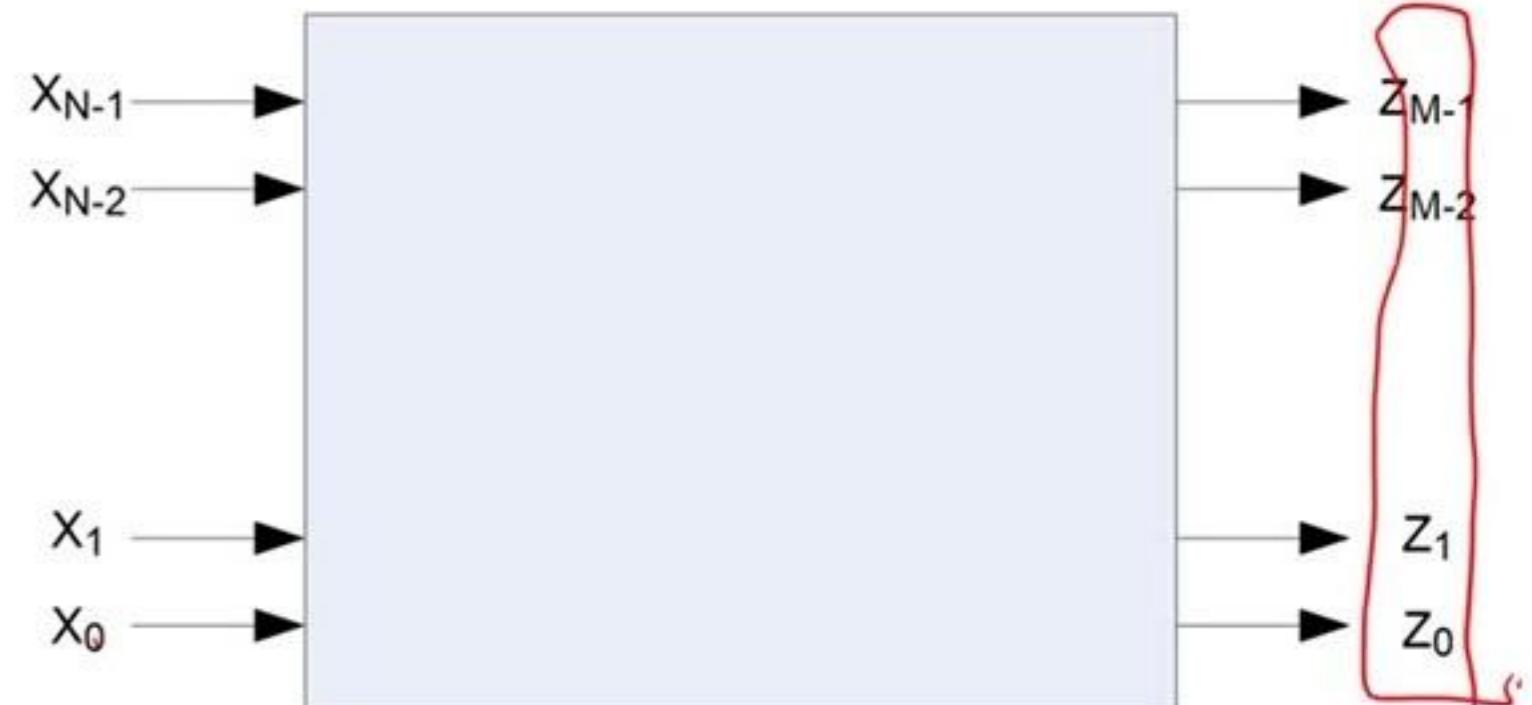
—  $Q$  bit 1 binary digit

# Rete logica come modello astratto

- Una rete logica è un modello astratto di un sistema fisico, quest'ultimo costituito da dispositivi tra loro interconnessi.
- Tali dispositivi si scambiano informazioni codificate.
- Le informazioni sono codificate tramite fenomeni fisici che si presentano ad un osservatore in due aspetti distinti.
- Ad esempio:
  - corrente forte, corrente debole
  - tensione alta, tensione bassa
  - perforazione di una zona di un foglio di carta, assenza di perforazione
  - magnetizzazione positiva/negativa di un'areola di materiale ferromagnetico
  - ...

# Caratterizzare una rete logica

- Una rete logica è caratterizzata da:
  1. un insieme di N variabili logiche di ingresso. Il loro valore all'istante  $t$  si chiama stato di ingresso all'istante  $t$ . L'insieme di tutti i  $2^N$  stati di ingresso verrà indicato con  $X$ .  $X = \{(x_{N-1}x_{N-2}\dots x_1x_0)\}$ .
  2. un insieme di M variabili logiche di uscita. Il loro valore all'istante  $t$  si chiama stato di uscita all'istante  $t$ . L'insieme di tutti i  $2^M$  stati di uscita verrà indicato con  $Z$ .  $Z = \{(z_{M-1}z_{M-2}\dots z_1z_0)\}$ .
  3. una legge di evoluzione nel tempo, che dice come le uscite si evolvono in funzione degli ingressi



# Classificazione delle reti logiche

$$Z = F(X)$$



- in base a **due criteri** riguardanti il tipo di legge di evoluzione nel tempo

## a) presenza/assenza di memoria:

Presente

- **reti combinatorie**: lo stato di uscita dipende soltanto dallo stato di ingresso. Ad un certo stato di ingresso corrisponde uno ed un solo stato di uscita. Analogia con il concetto di **funzione matematica**.
- **reti sequenziali**: lo stato di uscita dipende dalla storia degli stati di ingresso precedenti. Queste ultime sono **reti con memoria**, in quanto per decidere quale sia l'uscita hanno bisogno di ricordare il passato.

## b) temporizzazione della legge di evoluzione

t

- **reti asincrone**, in cui l'aggiornamento delle uscite avviene continuamente nel tempo
- **reti sincronizzate**, in cui l'aggiornamento delle uscite avviene ad istanti (di sincronizzazione) separati nel tempo

# Classificazione delle reti logiche

	<b>asincrono</b>	<b>sincronizzato</b>
<b>combinatorio</b>	1) Reti combinatorie	(non hanno un nome, sotto*aso del caso 3)
<b>sequenziale</b>	2) Reti sequenziali asincrone	3) Reti sequenziali sincronizzate

The table classifies logic networks into two main categories: **asincrono** (top row) and **sequenziale** (bottom row). The **asincrono** category contains '1) Reti combinatorie'. The **sequenziale** category contains '2) Reti sequenziali asincrone' and '3) Reti sequenziali sincronizzate'. Handwritten red annotations include a horizontal wavy line under '1)' and a curved arrow pointing from '2)' to '3)'.

# Interfaccia, interpretazione e fisica

- Una rete logica comunica con l'esterno tramite **variabili logiche** (0 e 1)
- La descrizione che ne faremo non dipende da:
  - L'interpretazione di queste variabili logiche (e.g., come numero intero, colore di un pixel, etc.) – che sta nella testa di chi le produce/consuma
  - Il fenomeno fisico con cui si codificano
- Useremo **in genere** questo modello per descrivere **circuiti elettronici** all'interno del calcolatore
  - Le variabili logiche sono codificate come tensioni (bassa: 0, alta: 1)
  - Scala di tensione dipende dall'implementazione (5V, 3V...)

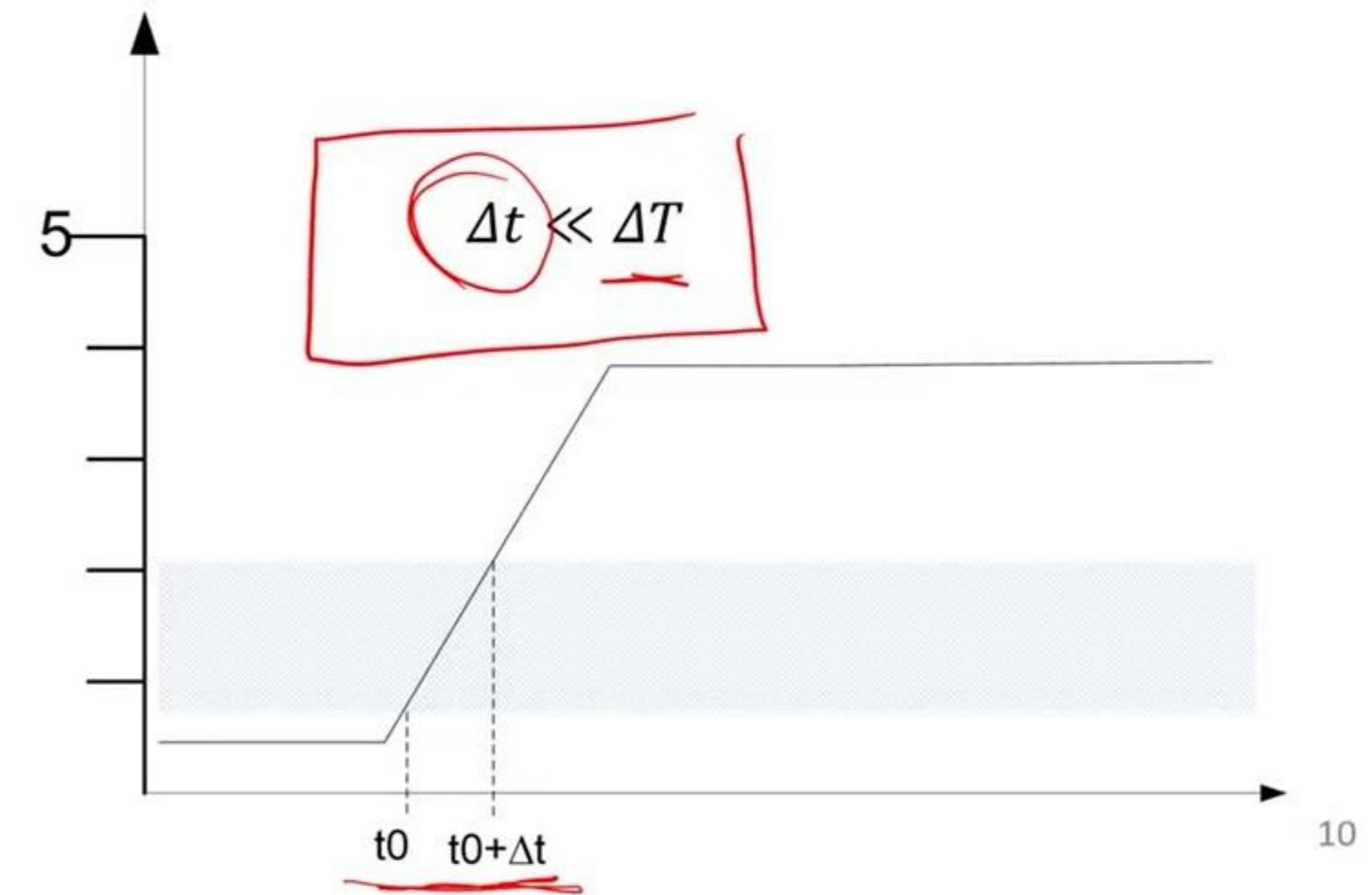
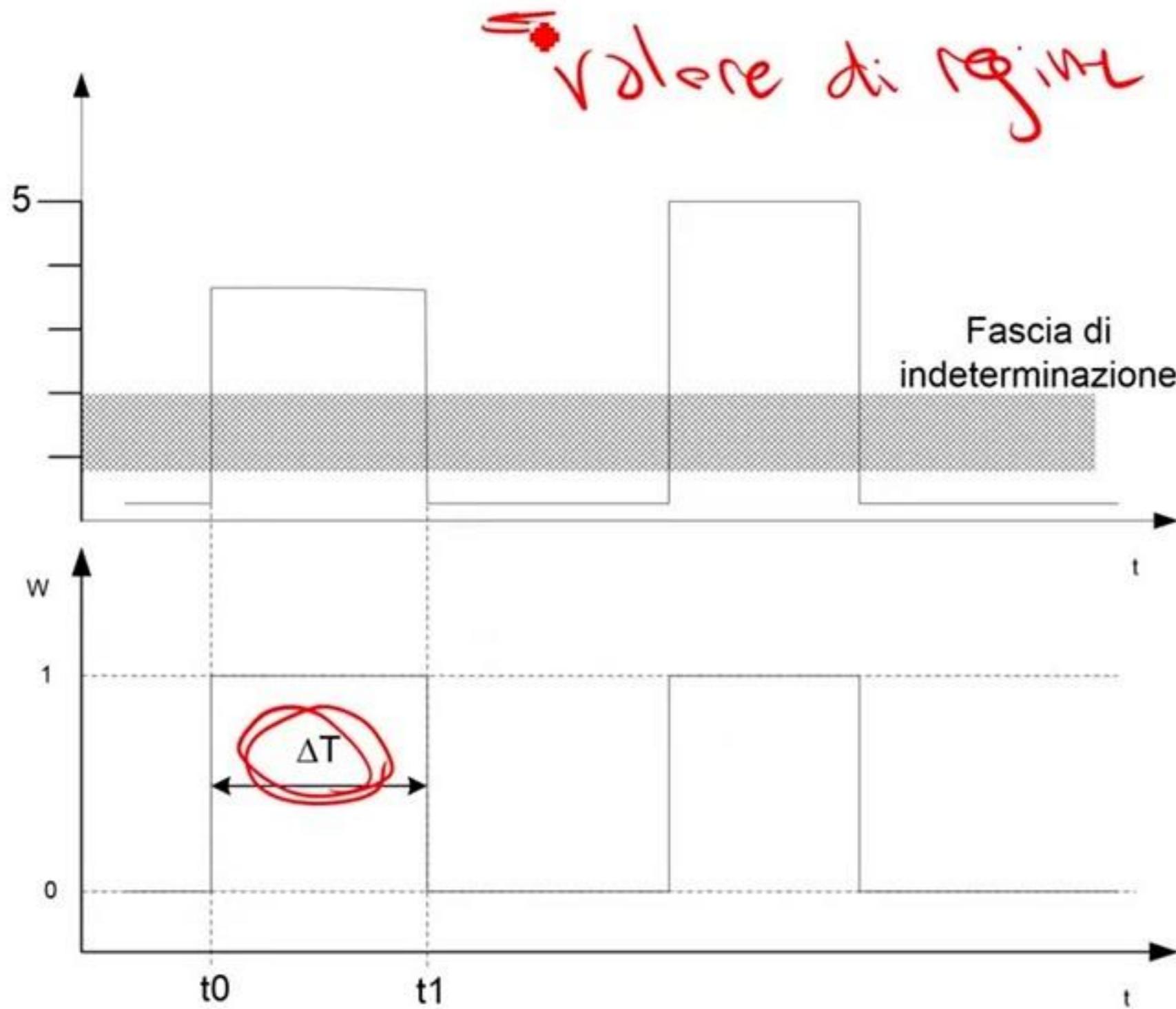
# Limiti del modello

• masso  
→ dimensioni / struttura interna

- Un modello è un modo **semplificato** di descrivere alcuni aspetti di una realtà complessa.
  - Fisica: punto materiale
  - Elettrotecnica: un circuito a parametri concentrati
- Tutte le volte che si modella un sistema, si trascurano alcuni suoi aspetti
- Necessario conoscere i limiti del modello

# Transizione dei segnali

- all'istante  $t_0$  la variabile logica  $w$  si setta, oppure transisce ad 1, etc.
- all'istante  $t_1$  la variabile logica  $w$  si resetta, oppure transisce a 0, etc.



# Non contemporaneita'

- Il ragionamento di prima non implica che si possano assumere variazioni **contemporanee** di piu' variabili



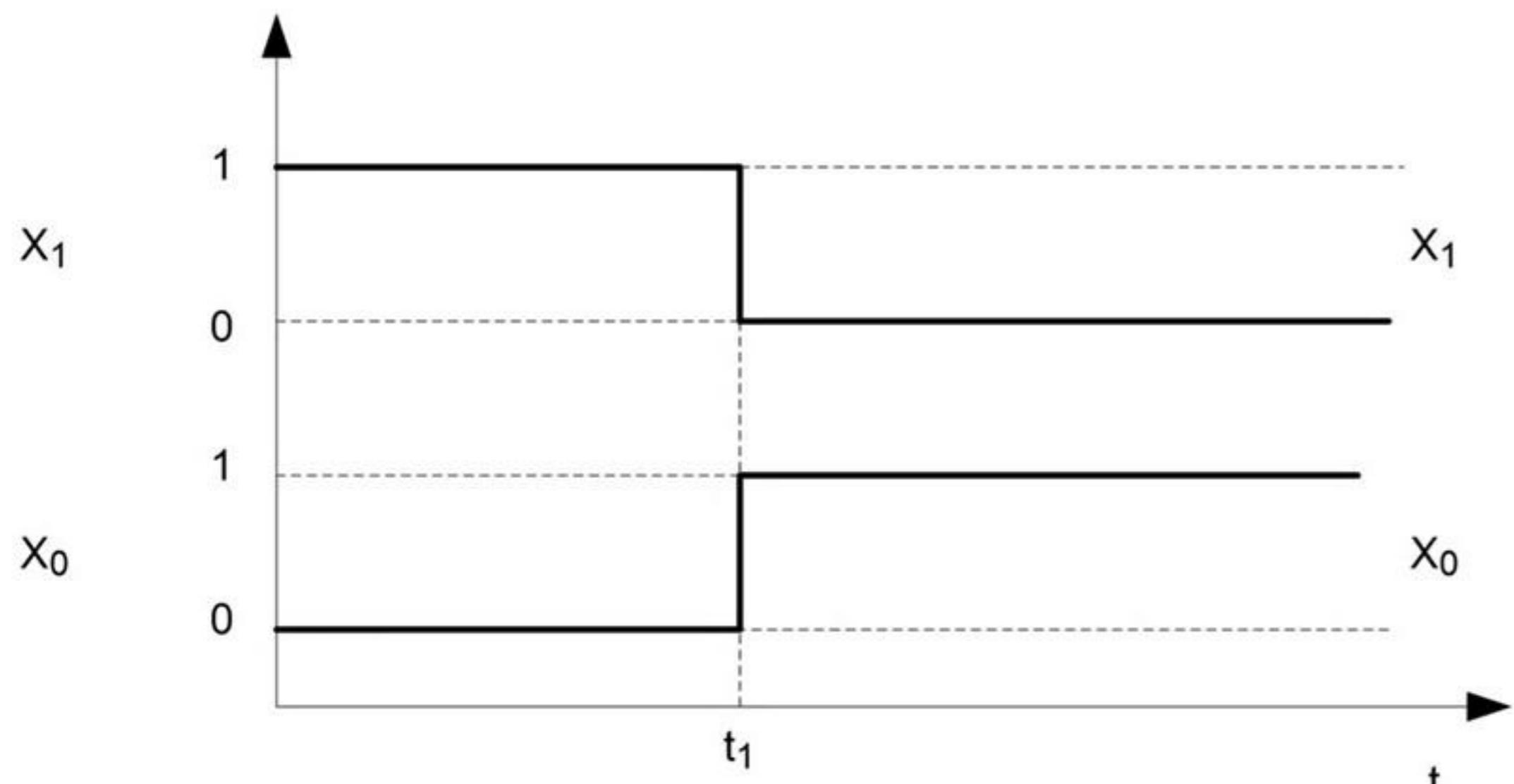
$\cancel{x_1}, \cancel{x_0}$

$$X = \{(00), (10), (01), (11)\}, Z = \{0,1\}$$

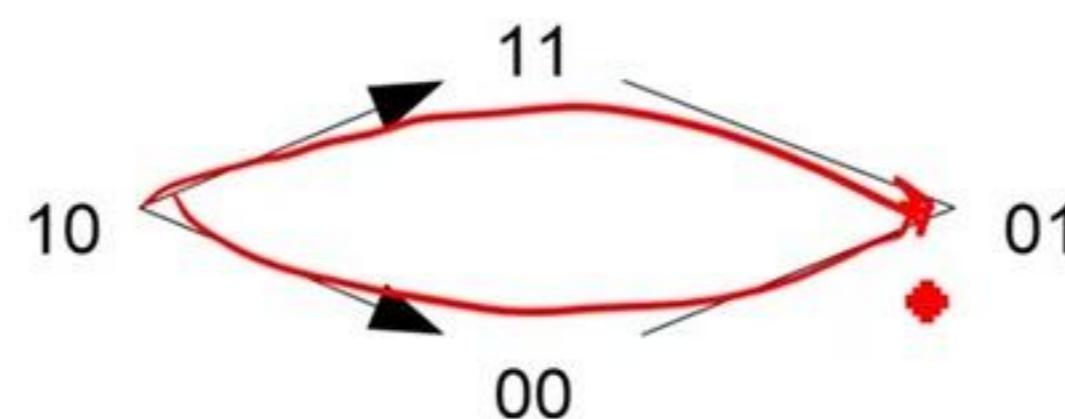
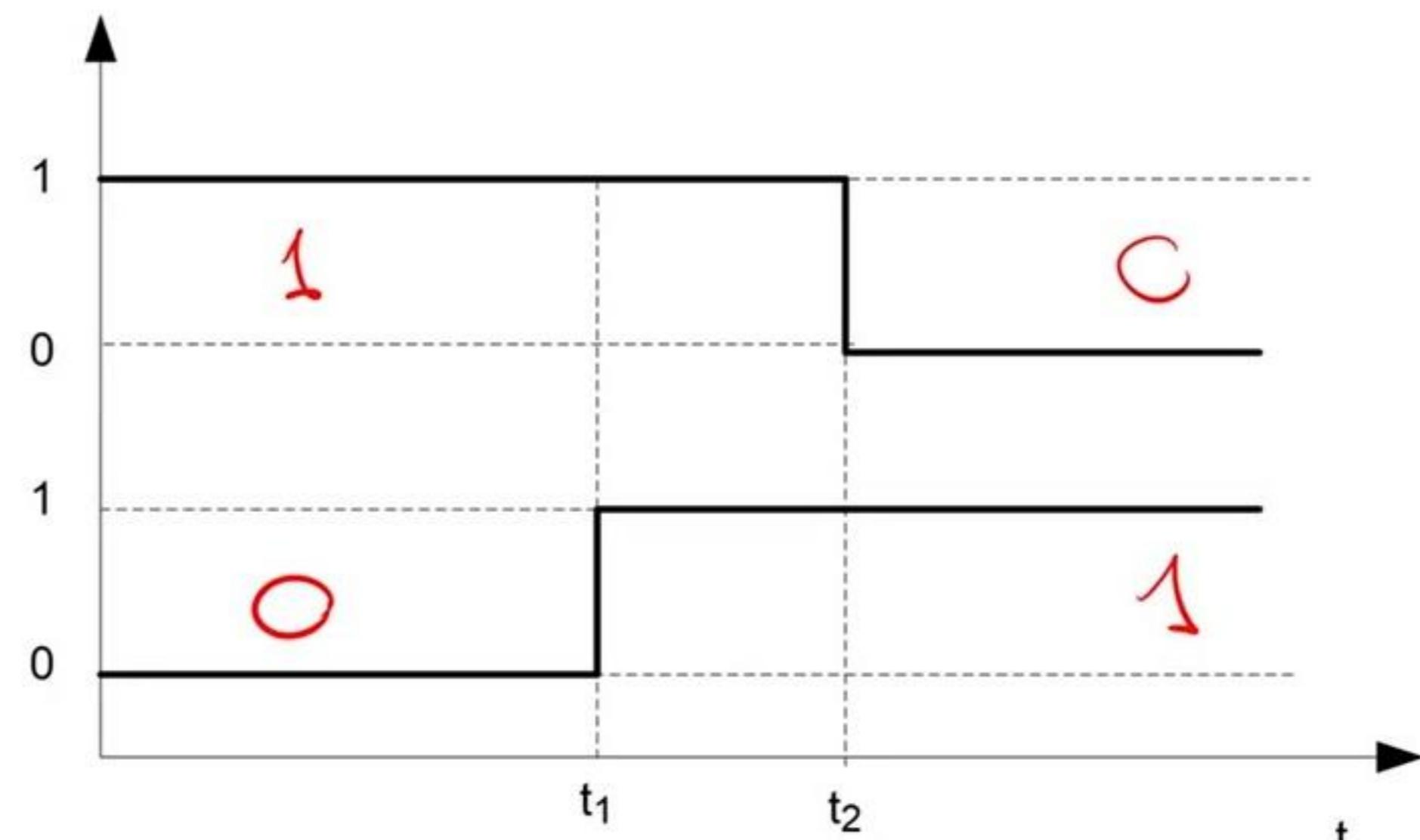
- Supponiamo che **prima dell'istante  $t_1$**  lo stato di ingresso presente sia, ad esempio, (1,0), e che **all'istante  $t_1$**  lo stato di ingresso diventi (0,1).

# Non contemporaneita'

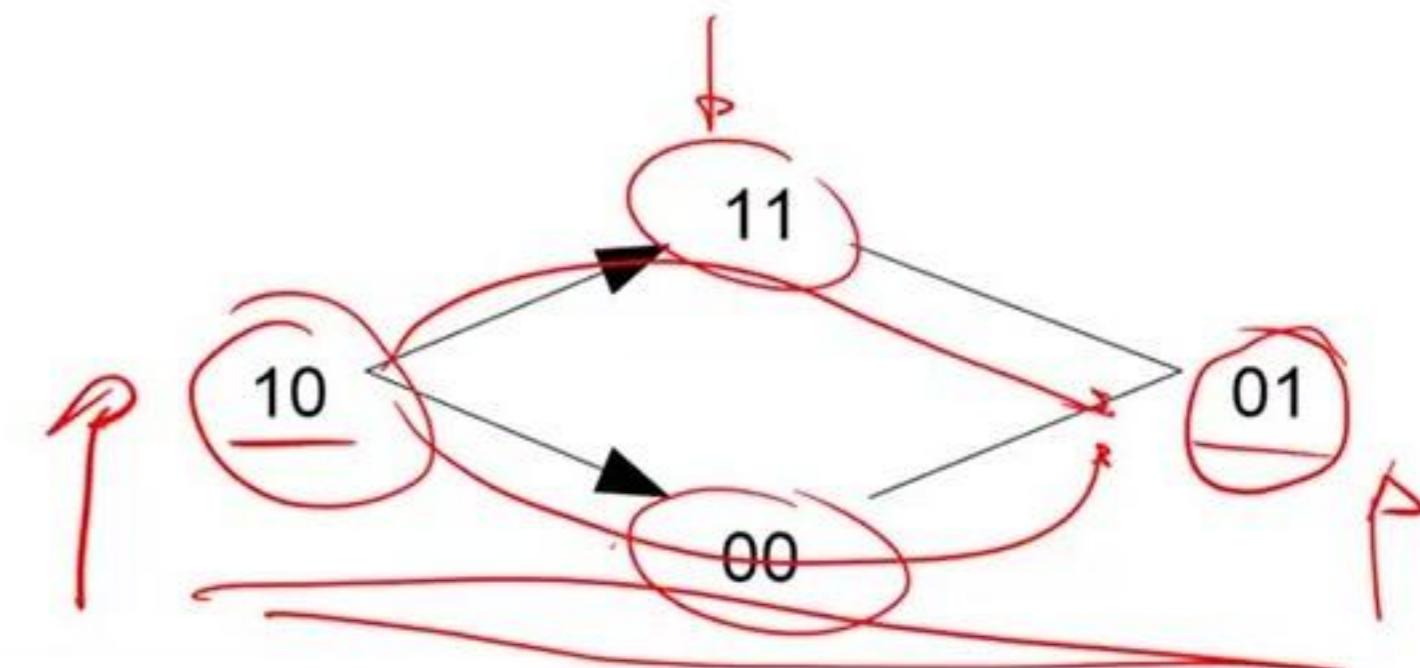
Ciò che *secondo me* accade in ingresso



Ciò che la rete vede (esempio)



# Non contemporaneita'



- **Evitare di supporre che due variabili di ingresso varino contemporaneamente**

- Nessun sistema obbedira' mai a questa ipotesi
- Per un certo intervallo di tempo vedra' in ingresso uno stato intermedio
- Rete sincronizzata, -> non e' un problema, in genere
- Rete asincrona
  - Potrebbe presentare in uscita stati spuri
  - Se sequenziale, si evolve in modo **imprevedibile**

ingresso

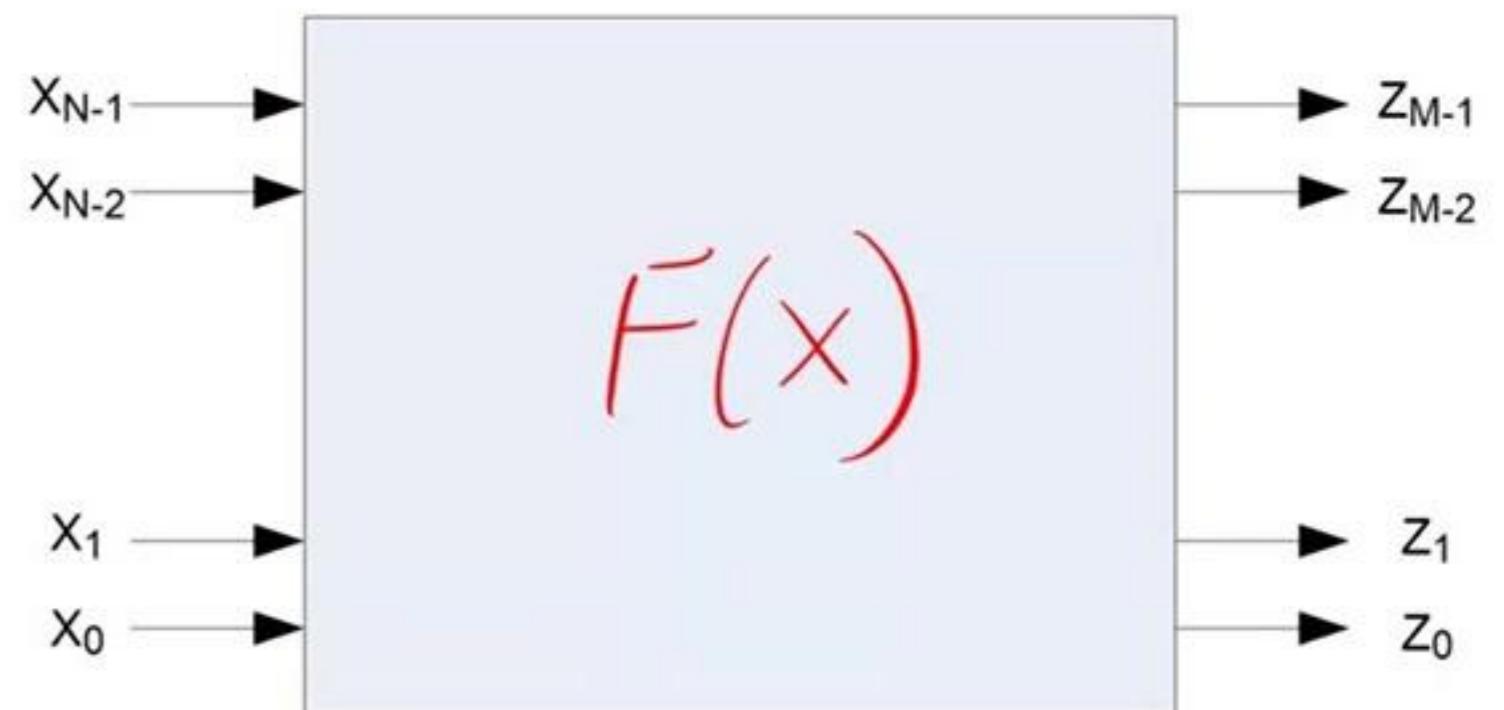
- **Stati di ingresso consecutivi devono essere adiacenti**

differenza pe 1 bit

# Reti Combinatorie

# Caratterizzare una rete combinatoria

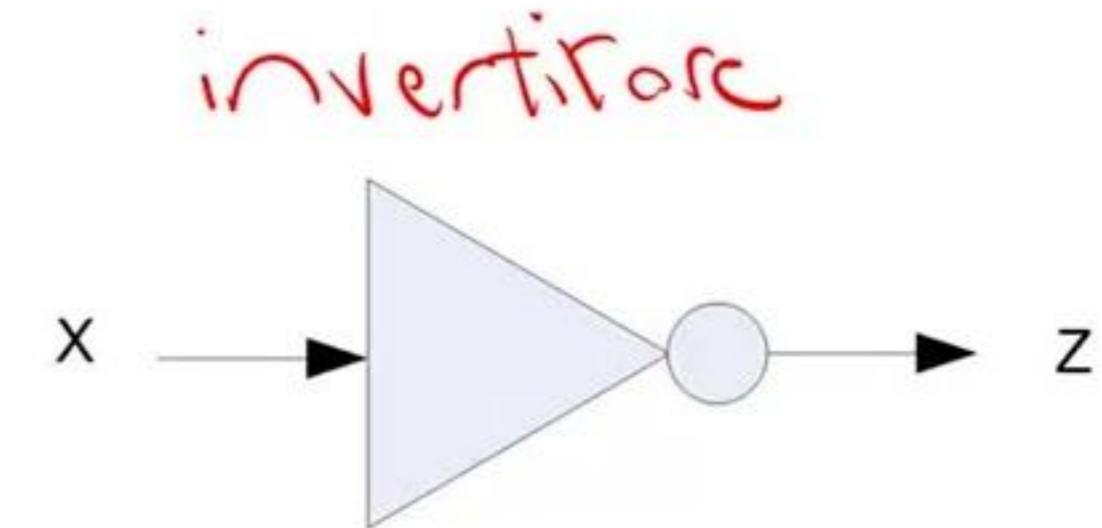
- Una rete combinatoria è caratterizzata da:
  1. Un insieme di **N variabili logiche di ingresso**.
  2. Un insieme di **M variabili logiche di uscita**.
  3. Una descrizione funzionale  $F: X \rightarrow Z$ , che mappa stati di ingresso in stati di uscita
  4. Una legge di evoluzione nel tempo, che dice: continuamente, se  $X$  e' lo stato di ingresso presente, adegua lo stato di uscita ad  $F(X)$



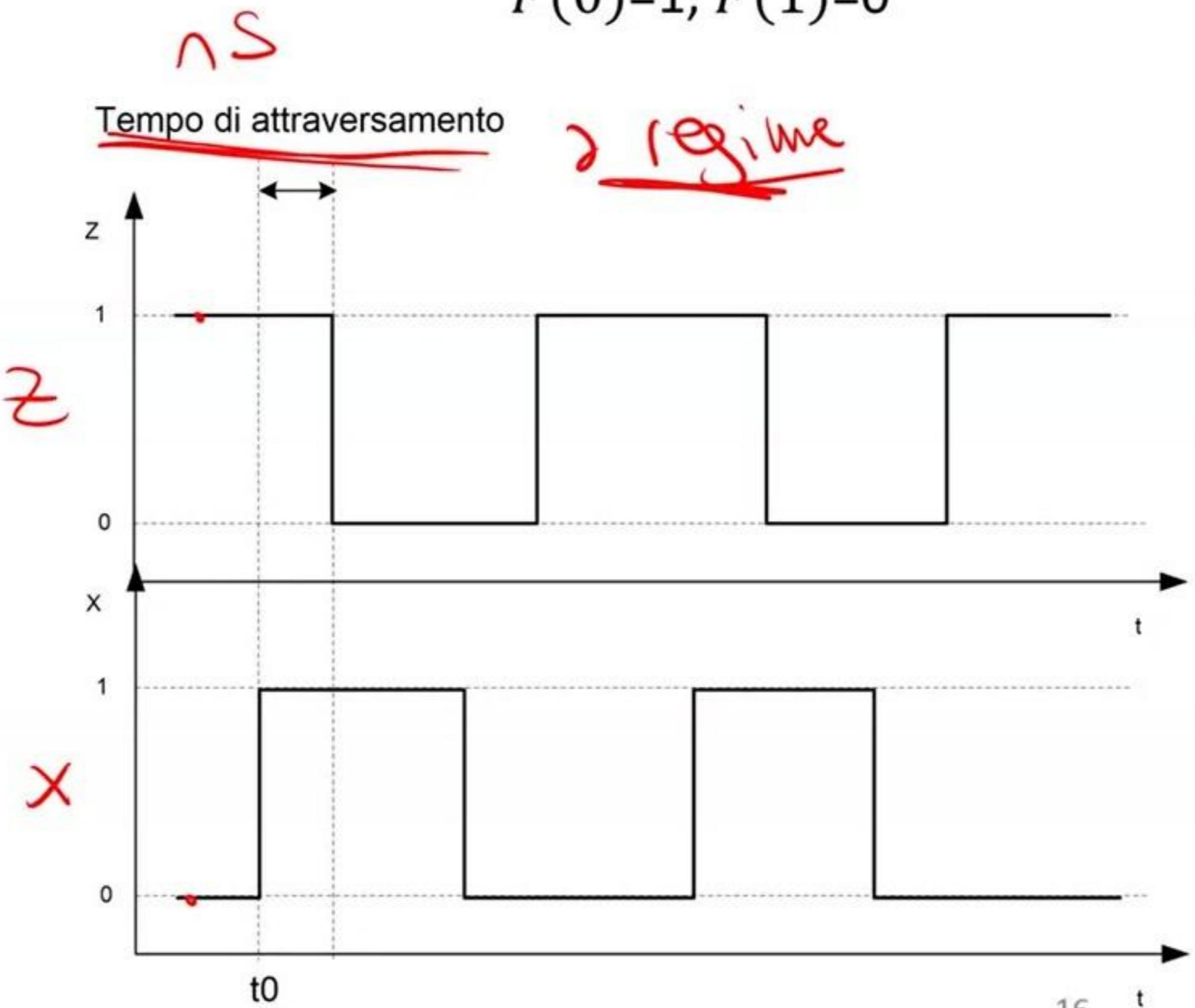
# Tempo di attraversamento

- Tempo di accesso, o di risposta
- Il tempo che ci mette la rete ad adeguare lo stato di uscita al nuovo stato di ingresso
  - Non nullo
  - Necessario attendere che l'uscita sia andata a **regime** prima di variare ancora lo stato di ingresso

Pilotaggio in modo fondamentale



$$N = 1, M = 1, X = \{0,1\}, Z = \{0,1\}$$
$$F(0)=1, F(1)=0$$



# Descrizione di reti combinatorie

- Una rete combinatoria si descrive dicendo:
  - quanti sono gli **ingressi**;
  - quante sono le **uscite**;
  - qual è la **funzione  $F$** , cioè la descrizione funzionale
- a parole
- notazioni testuali (vedremo il linguaggio **Verilog**)
- **tabelle di verità**



# Tabella di verita'

non specificata  
don't care

$x$        $z$

	$x_2$	$x_1$	$x_0$	$z_1$	$z_0$
	0	0	0	0	0
$z_1$	0	0	1	0	1
$z_0$	0	1	0	1	0
$z_1$	0	1	1	1	0
$z_0$	1	0	0	1	1
	1	0	1	1	1
	1	1	0	0	0
	1	1	1	0	0

	$x_2$	$x_1$	$x_0$	$z_1$	$z_0$
	0	0	0	0	0
	0	0	1	0	1
	0	1	0	1	0
	0	1	1	-	0
	1	0	0	1	-
	1	0	1	1	1
	1	1	0	0	-
	1	1	1	-	-

$\rightarrow$

	$x_2$	$x_1$	$x_0$	$z_1$	$z_0$
	0	0	0	0	0
$z_1$	0	0	1	0	1
$z_0$	0	1	0	1	0
$z_1$	0	1	1	1	0
$z_0$	1	0	0	1	1
	1	0	1	1	1
	1	1	0	0	0
	1	1	1	0	0

$\rightarrow$

	$x_2$	$x_1$	$x_0$	$z_1$	$z_0$
	0	0	0	0	0
	0	0	1	0	1
	0	1	0	1	0
	0	1	1	-	0
	1	0	0	1	-
	1	0	1	1	1
	1	1	0	0	-
	1	1	1	-	-

$z_1$  "riconsale"

↑

•

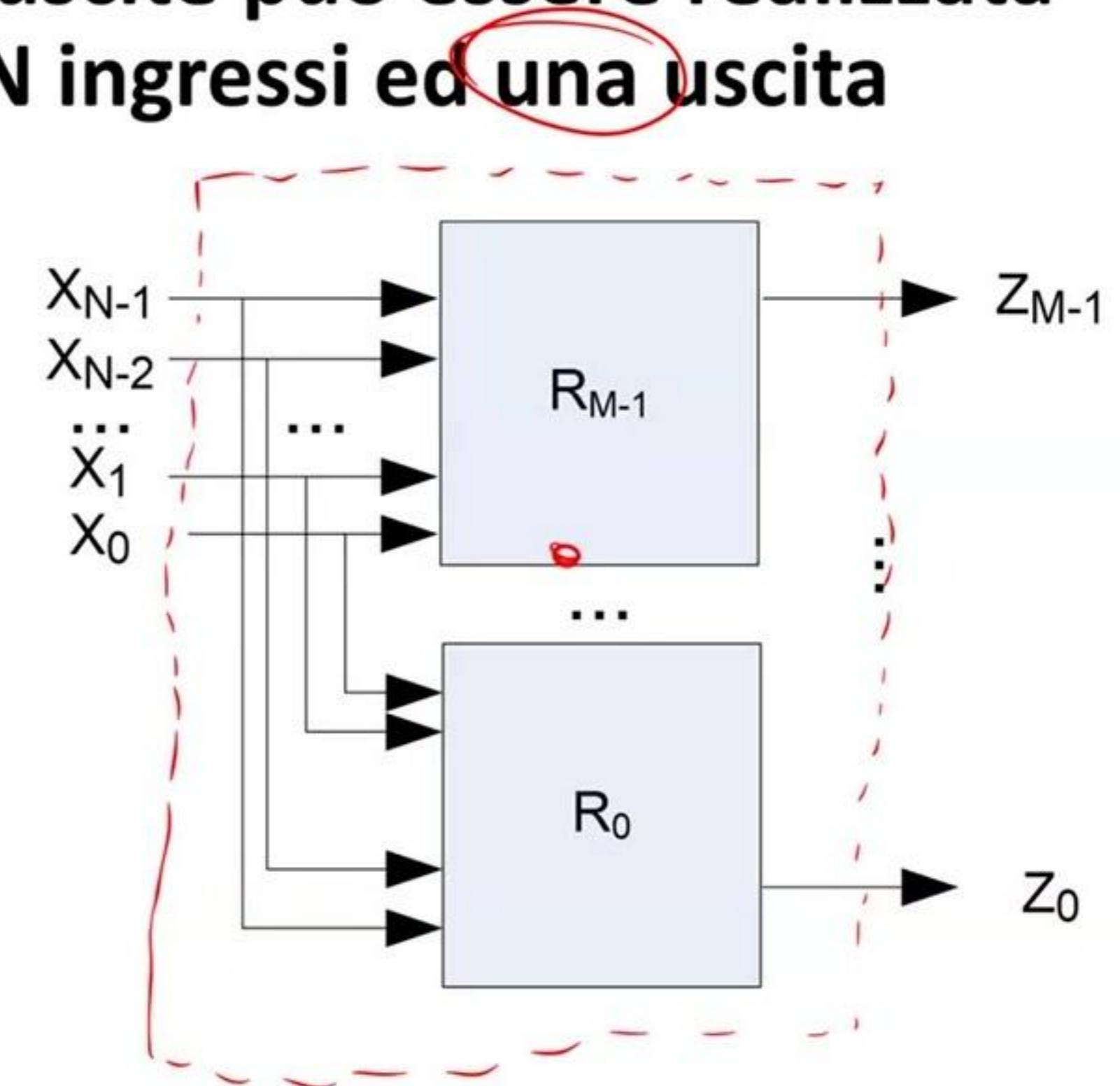
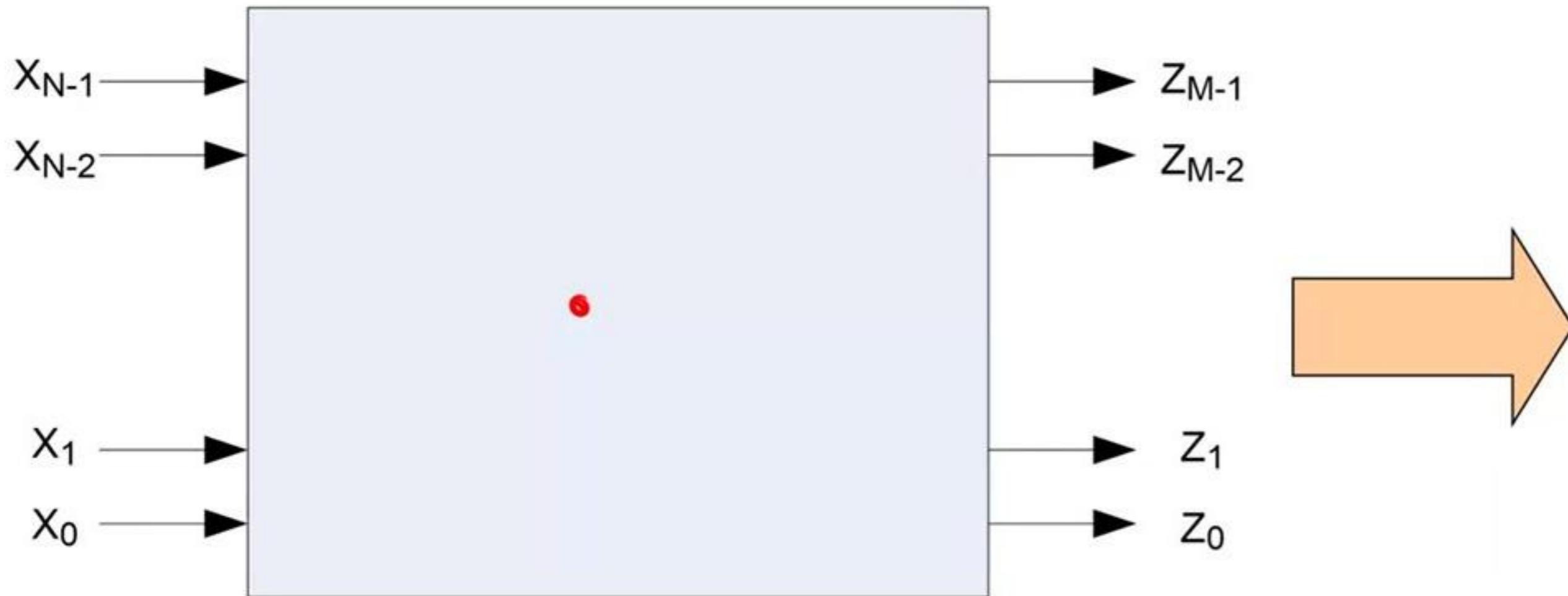
# Descrizione e sintesi

- La descrizione di una rete è un modo **formale** per dire **che cosa fa** quella rete, qual è cioè il suo comportamento osservabile.
  - Una tabella di verità.
- La sintesi di una rete è il progetto della realizzazione di una rete
  - la decisione di quali “scatolette” vanno messe, e connesse come, in modo da far sì che la rete si comporti come indicato dalla descrizione  
*realizzazione implementazione*
- Di una rete si da' prima la descrizione, e poi la sintesi

*tab*  $\forall N$   $\forall M$   $\forall F( )$

# Una proprieta' delle reti combinatorie

- Una rete combinatoria ad  $N$  ingressi ed  $M$  uscite può essere realizzata interconnettendo  $M$  reti combinatorie ad  $N$  ingressi ed una uscita



- Possiamo limitarci a trattare reti con una sola uscita

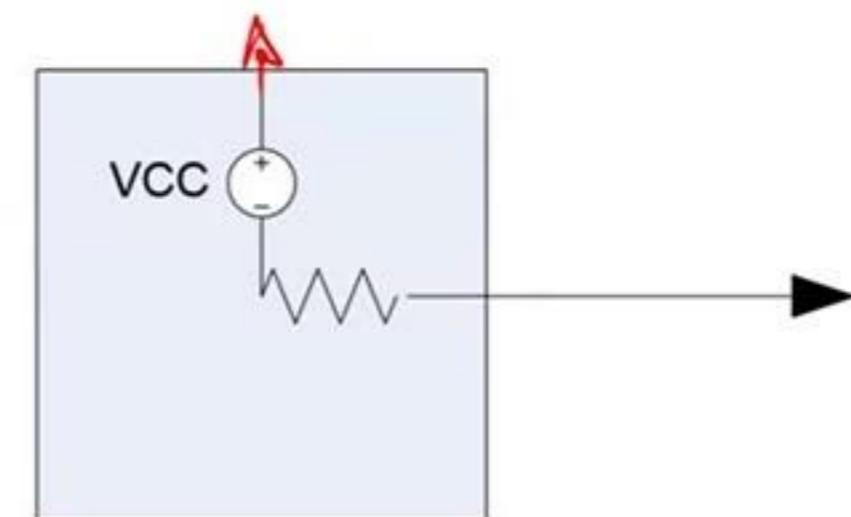
# Reti combinatorie elementari: zero ingressi

- Generatori di costante (caso degenero)

logico

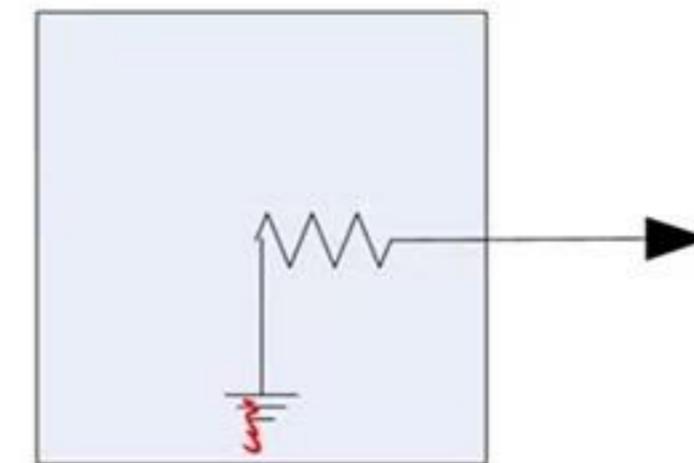
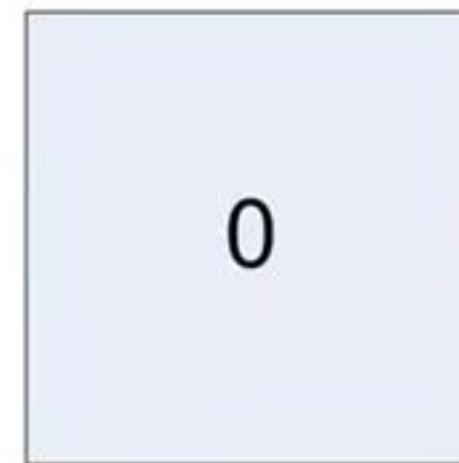


fisico



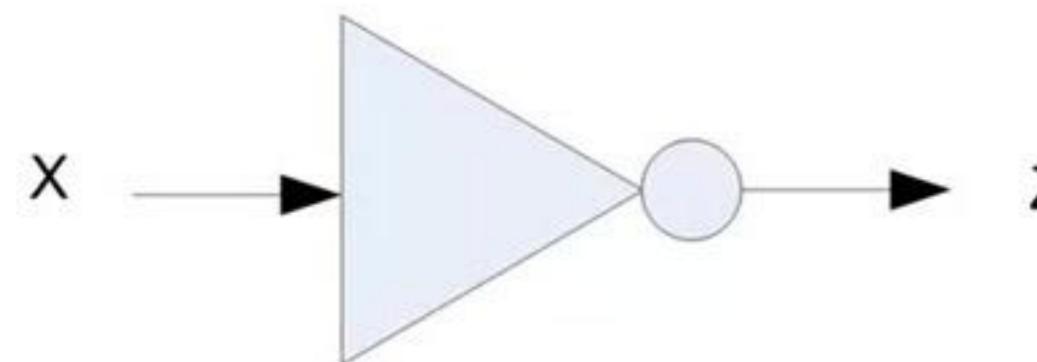
com plessit  nulla.

0



# Reti combinatorie ad un ingresso

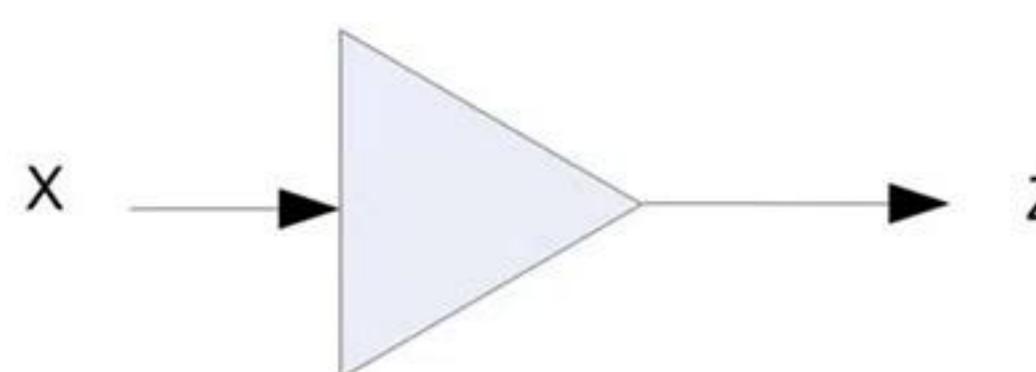
- Invertitore



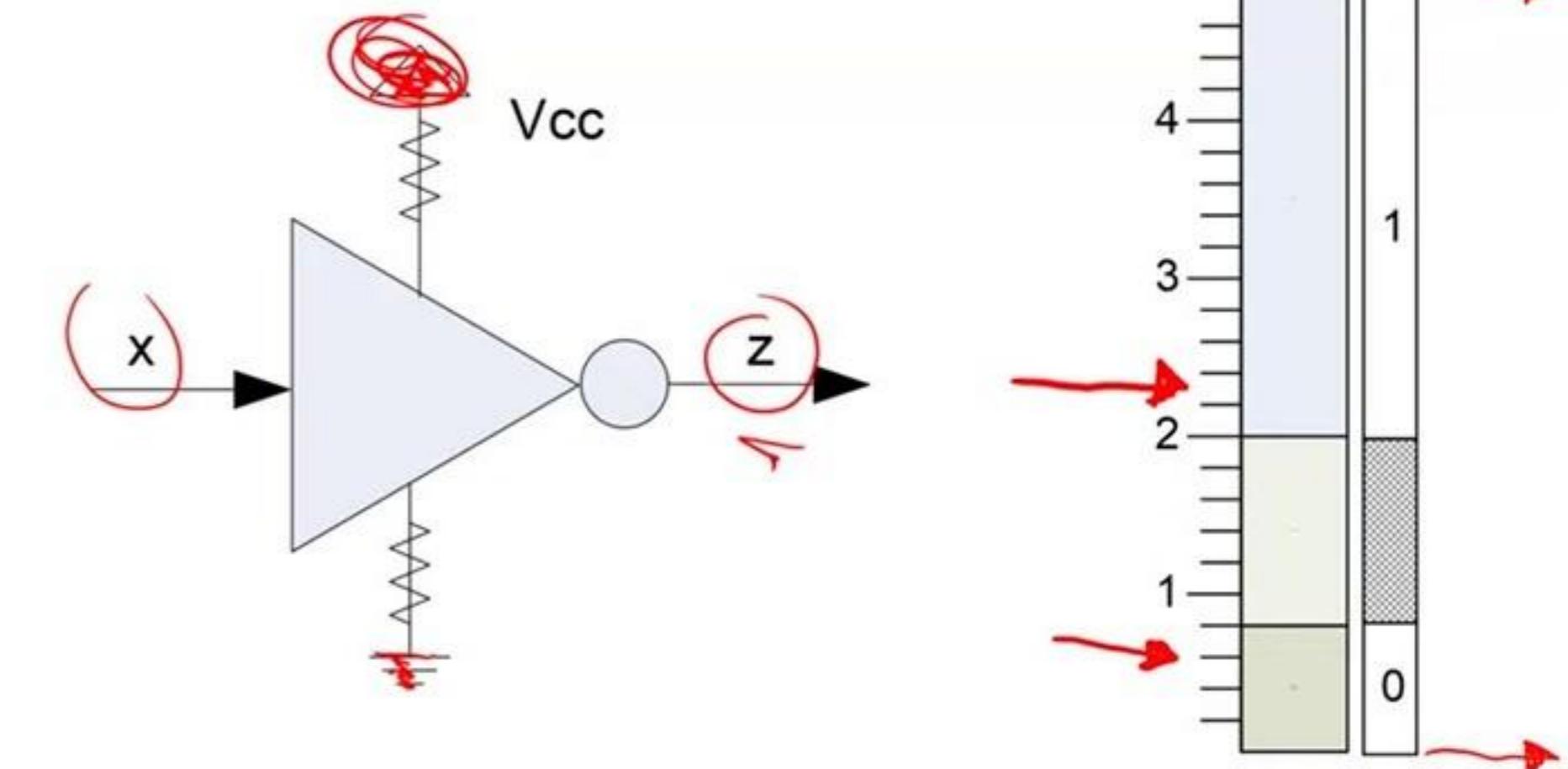
x	z
0	1
1	0

- Elemento neutro

*buffer*



x	z
0	0
1	1



- genera ritardo (utile per le temporizzazioni)
- **rigenera** i segnali elettrici degradati.

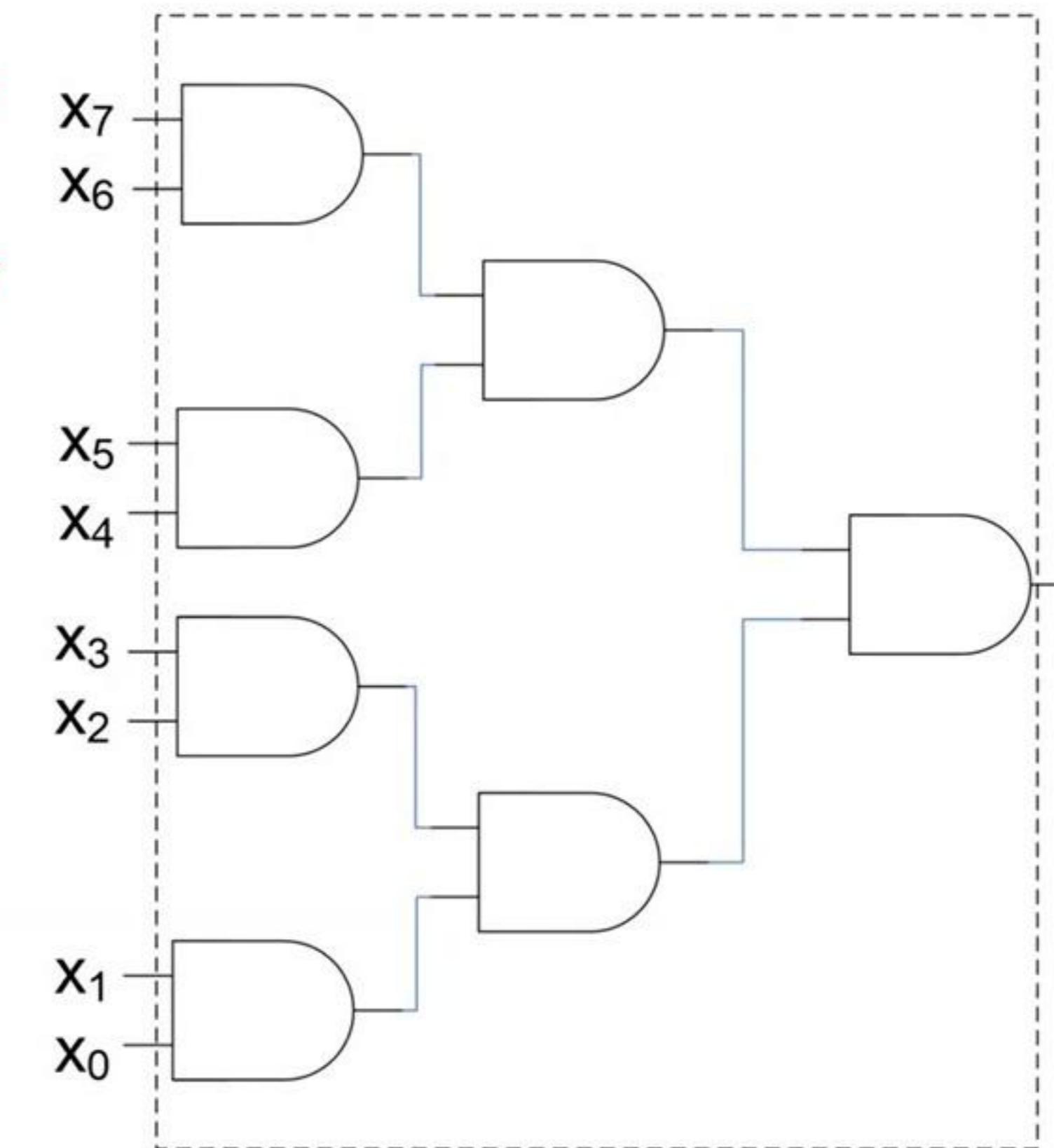
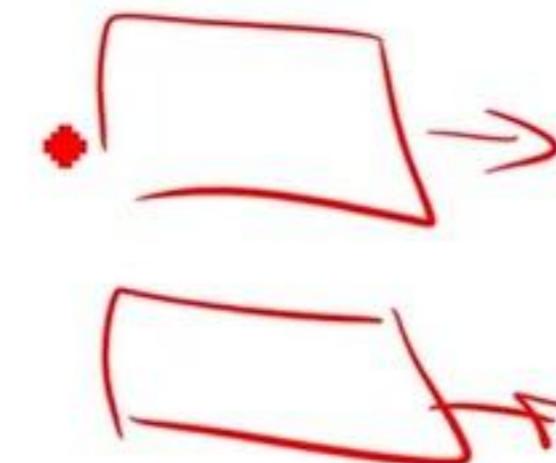
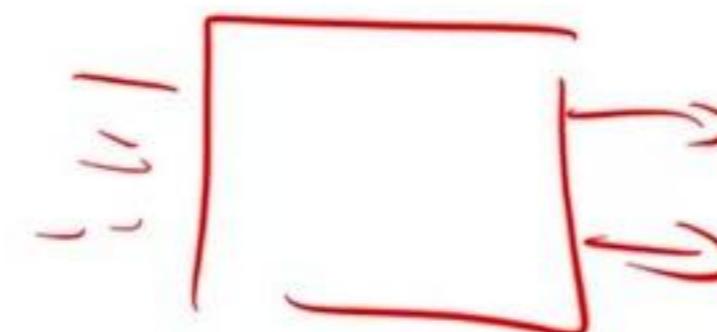
# AND a $2^N$ ingressi

- Disporre le porte ad **albero binario bilanciato**
- Numero di LL attraversati e' minimo
- Stesso ragionamento vale per le OR (fare per casa)

NOR    NAND

XOR

Reti comb.  
"complese"  
↑  
reti comb. "semplici"



# Reti combinatorie a due ingressi

- Quante sono in tutto?
- Tante quante le possibili tabelle di verita' diverse che posso costruire con due ingressi
- Una TdV per rete con  $N$  ingressi ha  $2^N$  righe
- Le possibili diverse tabelle di verita' sono  $\underline{2^{2^N}}$
- Quindi per  $N=2$ , abbiamo 16 diverse reti

N									
$X_{N-1}$	$X_{N-2}$	...	$X_1$	$X_0$	$Z^{(1)}$	$Z^{(2)}$	...	$Z^{(?)}$	$Z^{(?)}$
0	0	...	0	0	0	0	...	1	1
0	0	...	0	1	0	0	...	1	1
1	1	...	1	0	0	0	...	1	1
1	1	...	1	1	0	1	...	0	1

$2^{(2^N)}$

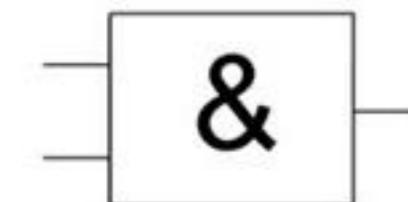
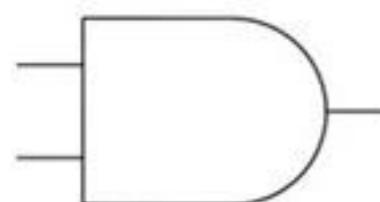
# Reti combinatorie a due ingressi

porte

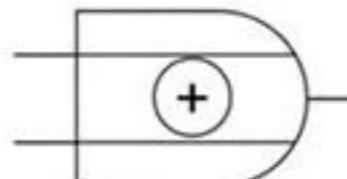
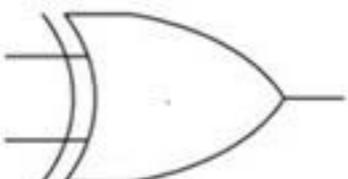
- Ad alcune corrispondono nomi speciali

$x_1$	$x_0$	$z^{(0)}$	$z^{(1)}$	$z^{(2)}$	$z^{(3)}$	$z^{(4)}$	$z^{(5)}$	$z^{(6)}$	$z^{(7)}$	$z^{(8)}$	$z^{(9)}$	$z^{(10)}$	$z^{(11)}$	$z^{(12)}$	$z^{(13)}$	$z^{(14)}$	$z^{(15)}$
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	1	1	1	1	0	0	0	1	1	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
<u>AND</u>		(*)	(A)	•	(+)	•	(+)	(B)	(C)	(D)	(E)	(-)	•	(-)	•	(F)	(*)

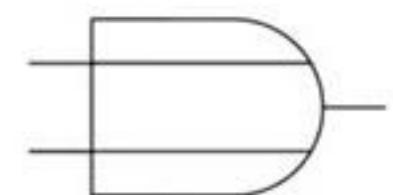
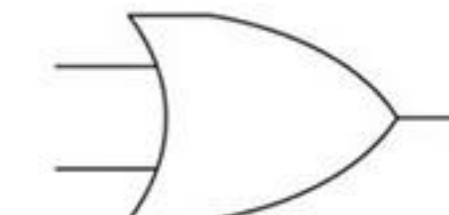
A) porta AND:  $z = 1 \Leftrightarrow x_0 = x_1 = 1$



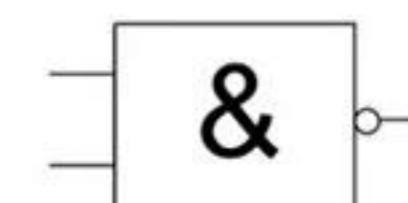
B) porta XOR:  $z = 1 \Leftrightarrow x_0 \neq x_1$



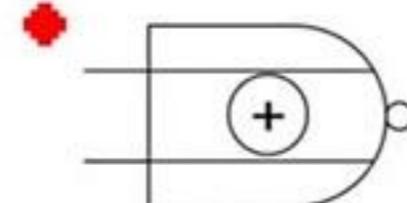
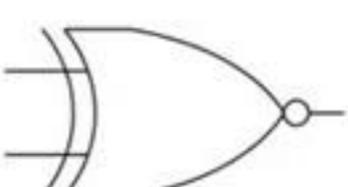
C) porta OR:  $z = 0 \Leftrightarrow x_0 = x_1 = 0$



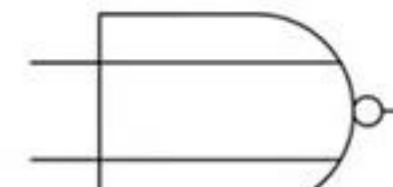
F) porta NAND:  $z = 0 \Leftrightarrow x_0 = x_1 = 1$



E) porta XNOR:  $z = 1 \Leftrightarrow x_0 = x_1$

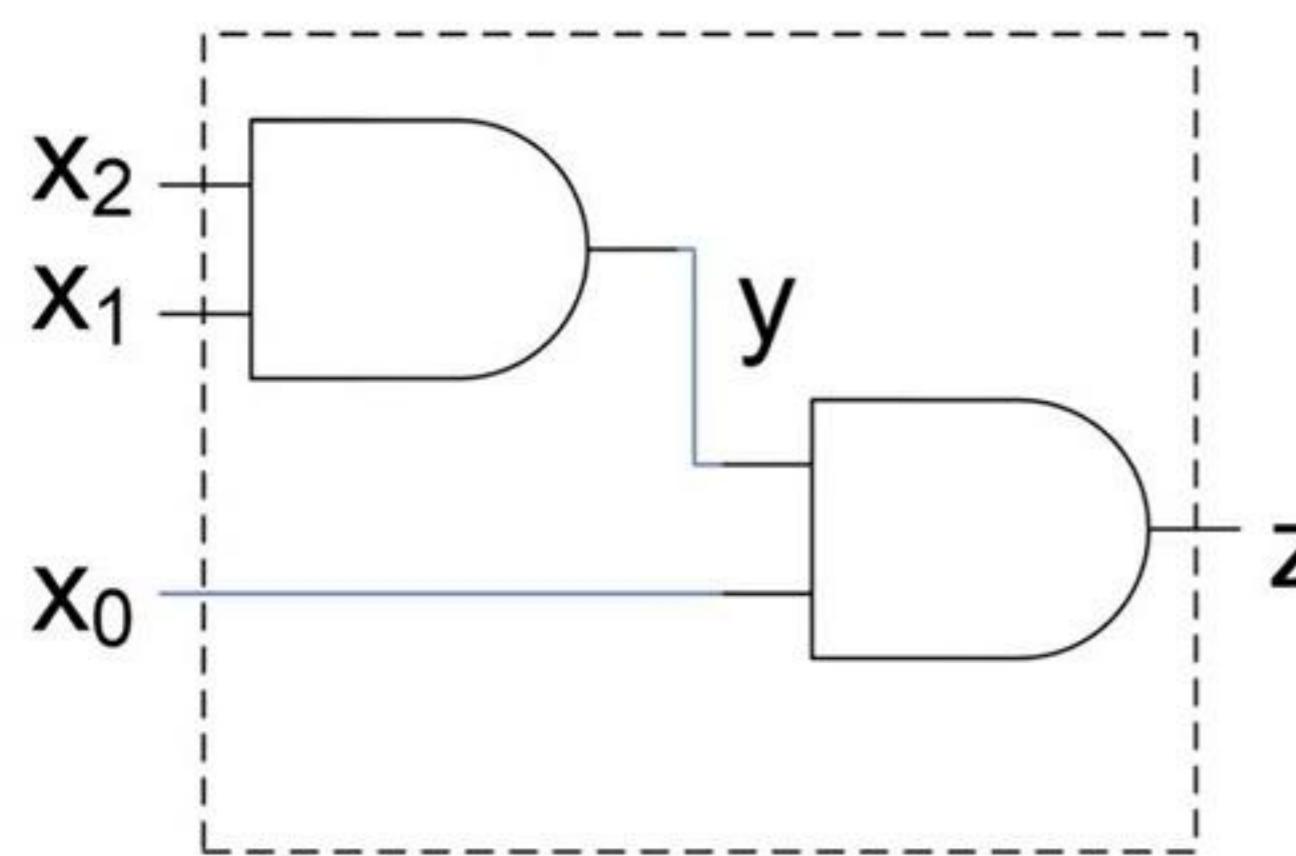


D) porta NOR:  $z = 1 \Leftrightarrow x_0 = x_1 = 0$



# AND e OR a piu' ingressi

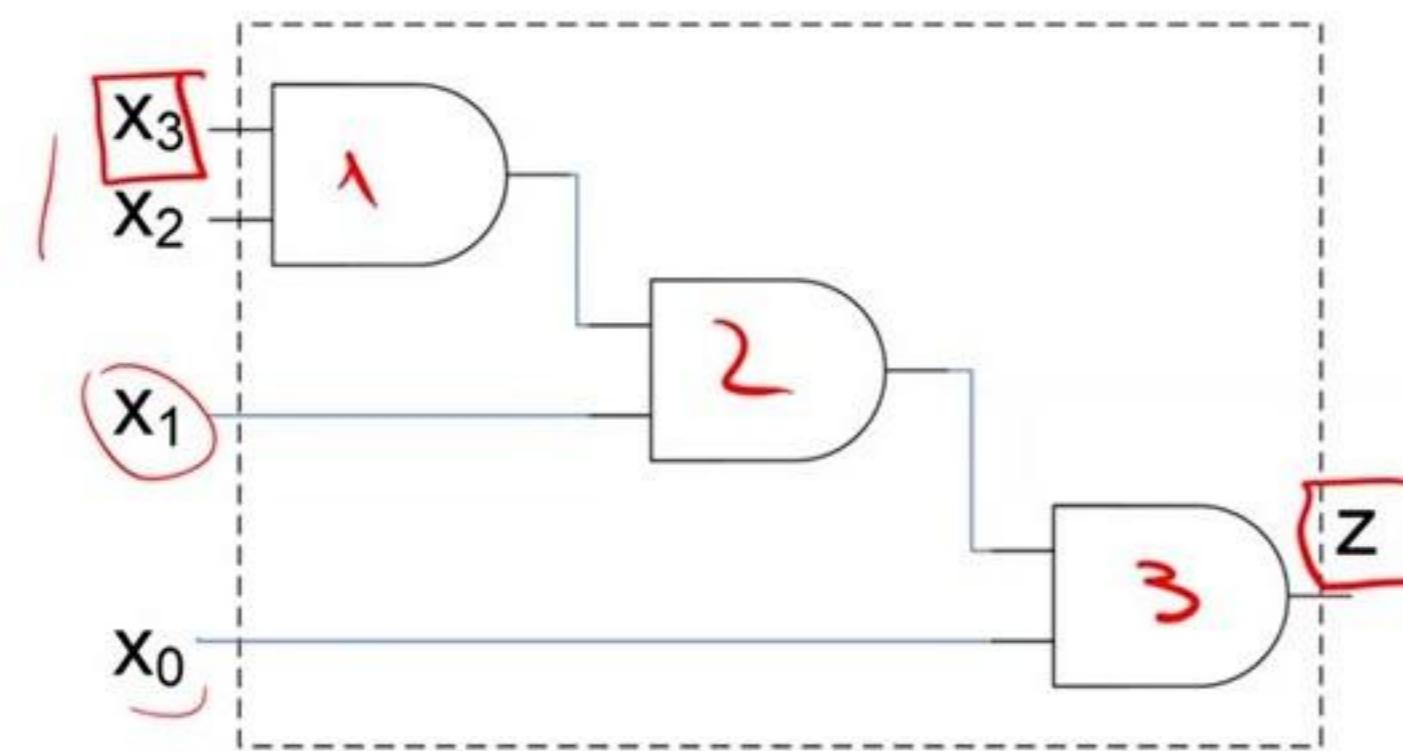
- Posso pensare di **estendere** la funzione logica realizzata dalle porte AND e OR al caso di  $N$  ingressi.
  - AND ad  $N$  ingressi: l'uscita vale 1 se e solo se **tutti gli ingressi valgono 1**
  - OR ad  $N$  ingressi: l'uscita vale 1 se e solo se **almeno un ingresso vale 1**



$x_2$	$x_1$	$x_0$	$y$	$z$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	1	0
1	1	1	1	1

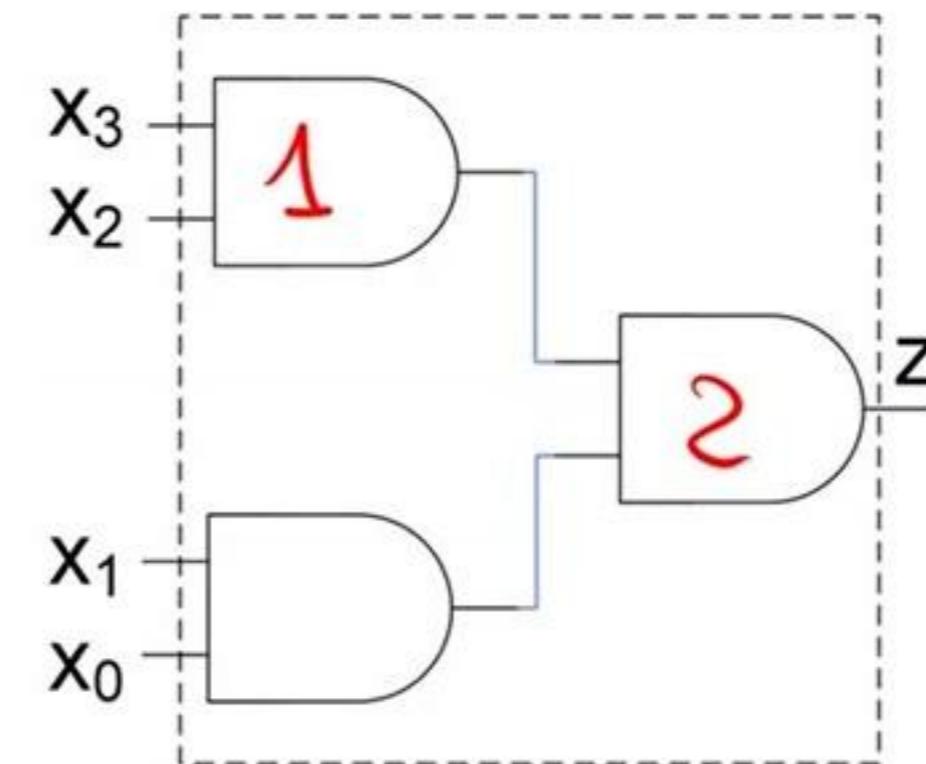
# AND e OR a piu' ingressi

- Modo poco furbo



8  
AND 2 ingressi

- Modo piu' furbo

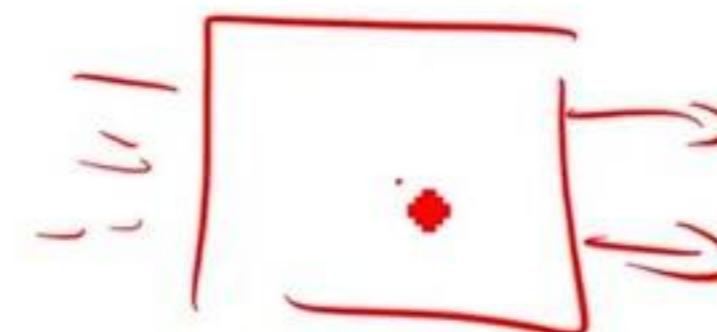


- Tre livelli di logica

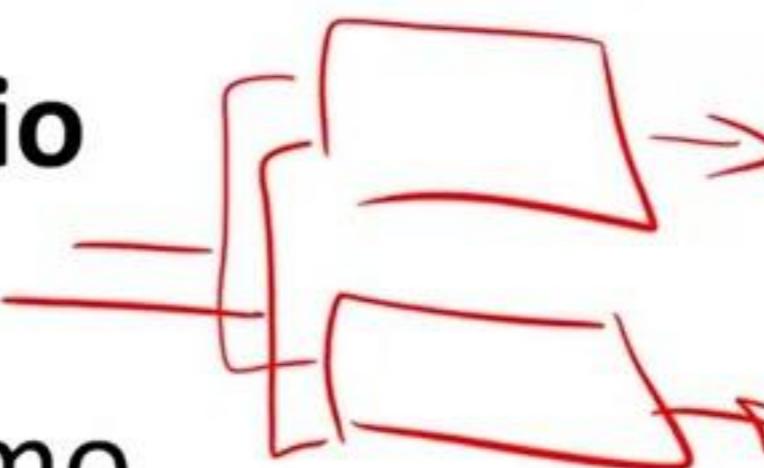
- Due livelli di logica

- Tempo di attraversamento minore

# AND a $2^N$ ingressi

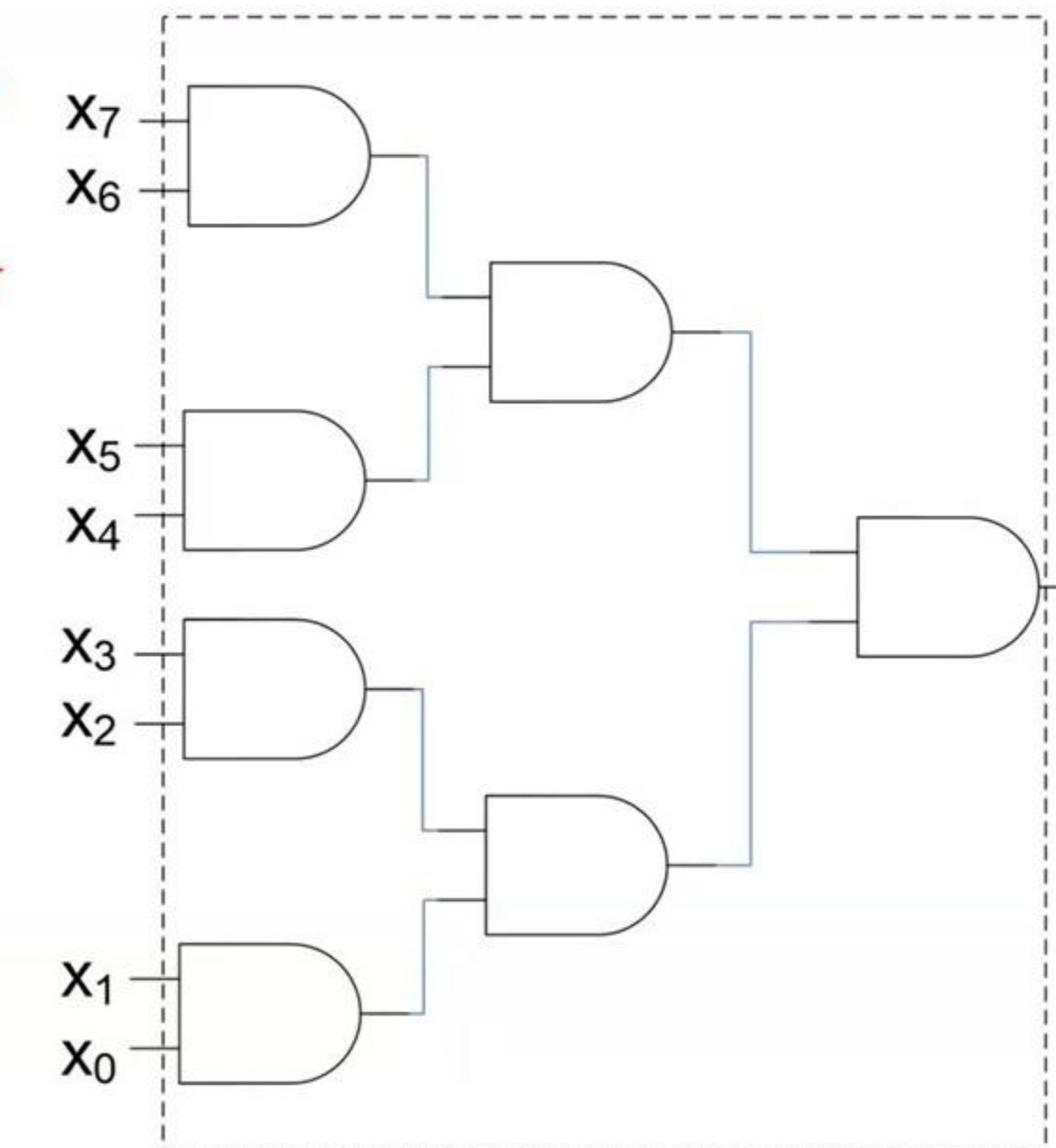


- Disporre le porte ad **albero binario bilanciato**
- Numero di LL attraversati e' minimo
- Stesso ragionamento vale per le OR  
(fare per casa)



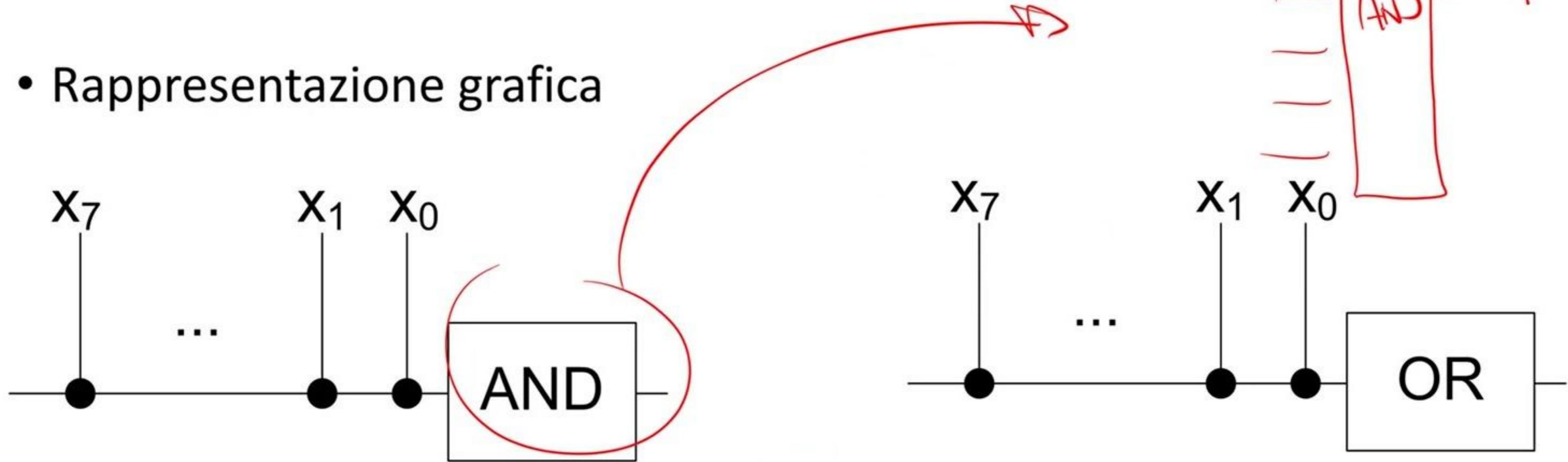
NOR    NAND  
  
XOR

Reti comb.  
"complese"  
↑  
reti comb. "semplici"



# AND e OR con molti ingressi

- Rappresentazione grafica



# Algebra di Boole

- Formalismo per rappresentare le leggi di corrispondenza delle reti combinatorie
- Sistema algebrico basato su:
  - **variabili logiche**, capaci di assumere due valori (0 e 1)
  - **operatori logici**: si applicano alle variabili logiche per costruire espressioni algebriche

# Operatori logici



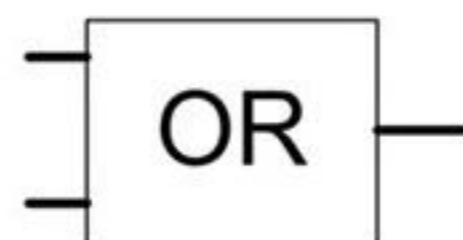
- **Complemento:** Operatore unario. Si indica con:  $\bar{x}$ , (anche:  $!x$ ,  $/x$ ). È definito come:  $\bar{0} = 1$ ,  $\bar{1} = 0$ .

- **Prodotto logico:**  $x \cdot y$ , definito come



$$\begin{aligned}0 \cdot 0 &= 0 \\0 \cdot 1 &= 0 \\1 \cdot 0 &= 0 \\1 \cdot 1 &= 1\end{aligned}$$

- **Somma logica:**  $x + y$ , definita come



$$\begin{aligned}0 + 0 &= 0 \\0 + 1 &= 1 \\1 + 0 &= 1 \\1 + 1 &= 1\end{aligned}$$

# Proprieta' degli operatori booleani

- Alcune sono ovvie, altre molto meno

Proprietà degli operatori booleani						
x	y	z	$y \cdot z$	$x + (y \cdot z)$	$(x+y) \cdot (x+z)$	$(x+y) \cdot (x+z)$
0	0	0	0	0	0	0
0	0	1	0	0	0	1
0	1	0	0	0	1	0
0	1	1	1	1	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	0	1	1	1
1	1	1	1	1	1	1

Involutiva (del complemento):  $\bar{\bar{x}} = x$

commutativa (della somma e del prodotto):

$$x + y = y + x, x \cdot y = y \cdot x$$

associativa (della somma e del prodotto):

$$x + y + z = (x + y) + z = x + (y + z)$$

$$x \cdot y \cdot z = (x \cdot y) \cdot z = x \cdot (y \cdot z)$$

distributiva della somma rispetto al prodotto e  
viceversa:

$$\rightarrow x \cdot (y + z) = (x \cdot y) + (x \cdot z)$$

$$\rightarrow x + (y \cdot z) = (x + y) \cdot (x + z) \leftarrow$$

(disegnare tabella di verità)

x	y	z	$y \cdot z$	$x + (y \cdot z)$	$(x+y) \cdot (x+z)$	$(x+y) \cdot (x+z)$
0	0	0	0	0	0	0
0	0	1	0	0	0	1
0	1	0	0	0	1	0
0	1	1	1	1	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	0	1	1	1
1	1	1	1	1	1	1

## Esercizio

- Utilizzando le proprietà dell'algebra, semplificare al massimo la seguente espressione

$$\begin{aligned} & \overline{a} \cdot \overline{b} \cdot \overline{c} \cdot \overline{d} + \overline{a} \cdot b \cdot \overline{c} \cdot \overline{d} + a \cdot \overline{b} \cdot \overline{c} \cdot \overline{d} + \overline{a} \cdot b \cdot d + b \cdot c \cdot d + a \cdot b \cdot d \\ & + \overline{a} \cdot \overline{b} \cdot \overline{c} \cdot \overline{d} \\ & = \overline{a} \cdot \overline{c} \cdot \overline{d} + \overline{b} \cdot \overline{c} \cdot \overline{d} + \overline{b} \cdot d + b \cdot c \cdot d \\ & = \overline{a} \cdot \overline{c} \cdot \overline{d} + \overline{b} \cdot \overline{c} \cdot \overline{d} + b \cdot d \end{aligned}$$

$b \cdot d \cdot (1+c)$   
 $\hline$  $b+c+d$

• Sintesi di costo minimo

# Teoremi di De Morgan per N variabili logiche

- I Teoremi di De Morgan valgono per un numero qualunque di variabili logiche.

$$1. \overline{x_0 \cdot x_1 \cdot \dots \cdot x_{N-1}} = \overline{x_0} + \overline{x_1} + \dots + \overline{x_{N-1}}$$

$$2. \overline{x_0 + x_1 + \dots + x_{N-1}} = \overline{x_0} \cdot \overline{x_1} \cdot \dots \cdot \overline{x_{N-1}}$$

- Si dimostrano **per induzione** sul n. di variabili logiche

- )
  - dimostrare che la proprietà vale per un certo numero  $n_0$  (passo base);
  - dimostrare che, se vale per un generico  $n \geq n_0$ , allora vale anche per  $n + 1$  (passo induttivo)

# Dimostrazione della prima tesi

- Tesi:  $\overline{x_0 \cdot x_1 \cdot \dots \cdot x_{N-1}} = \overline{x_0} + \overline{x_1} + \dots + \overline{x_{N-1}}$
- Passo base:  $n_0 = 2$ : già fatto (tabella di verità)

- Passo induttivo:

- **Hp:**  $\overline{x_0 \cdot x_1 \cdot \dots \cdot x_{N-1}} = \overline{x_0} + \overline{x_1} + \dots + \overline{x_{N-1}}$

- **Th:**  $\overline{x_0 \cdot x_1 \cdot \dots \cdot x_{N-1} \cdot x_N} = \overline{x_0} + \overline{x_1} + \dots + \overline{x_{N-1}} + \overline{x_N}$

- Pongo  $x_0 \cdot x_1 \cdot \dots \cdot x_{N-1} = \alpha$

→ •  $\overline{\alpha} = \overline{(x_0 \cdot x_1 \cdot \dots \cdot x_{N-1})} = \overline{x_0} + \overline{x_1} + \dots + \overline{x_{N-1}}$  (ipotesi induttiva)

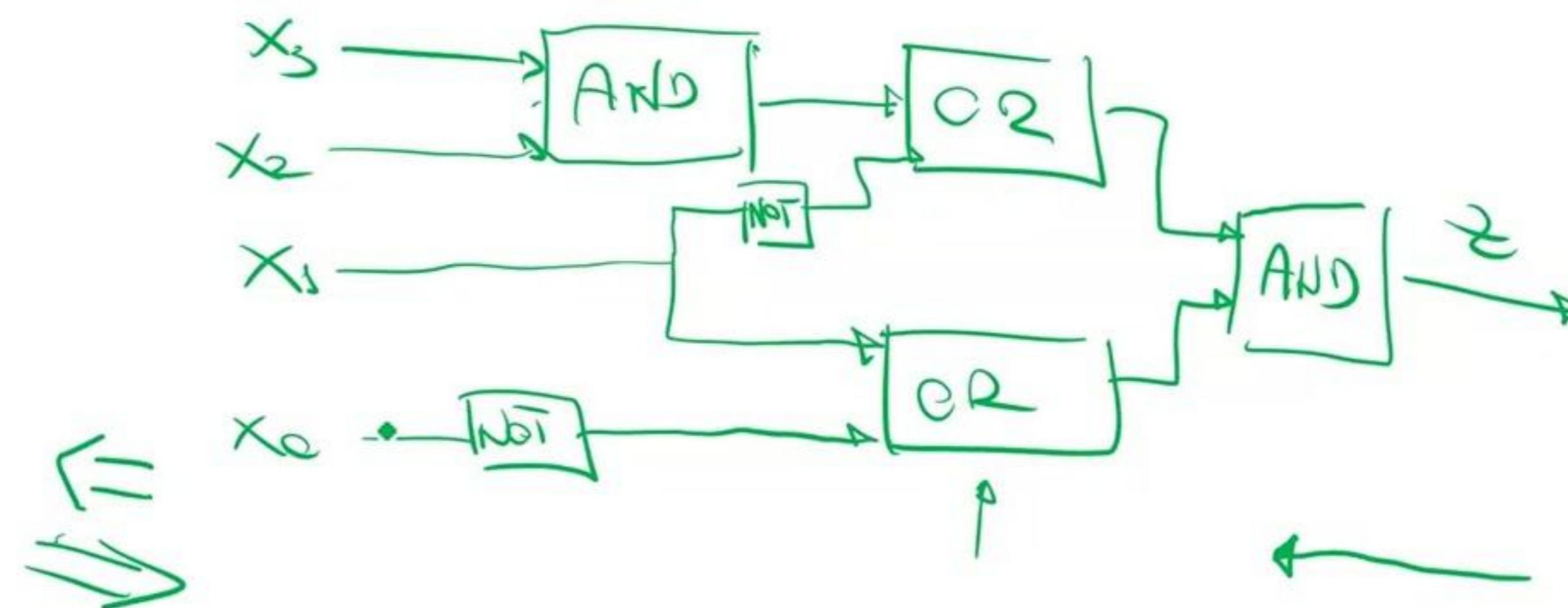
•  $\overline{(x_0 \cdot x_1 \cdot \dots \cdot x_{N-1}) \cdot x_N} = \overline{\alpha \cdot x_N} = \overline{\alpha} + \overline{x_N}$  (passo base)

•  $= \overline{x_0} + \overline{x_1} + \dots + \overline{x_{N-1}} + \overline{x_N}$

# Algebra di Boole e reti combinatorie

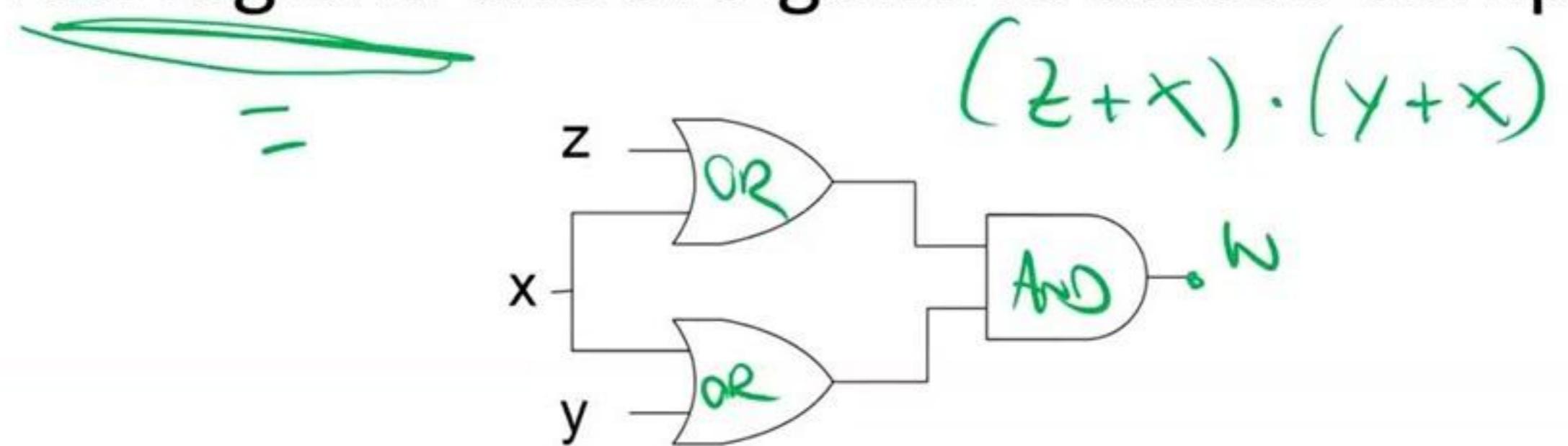
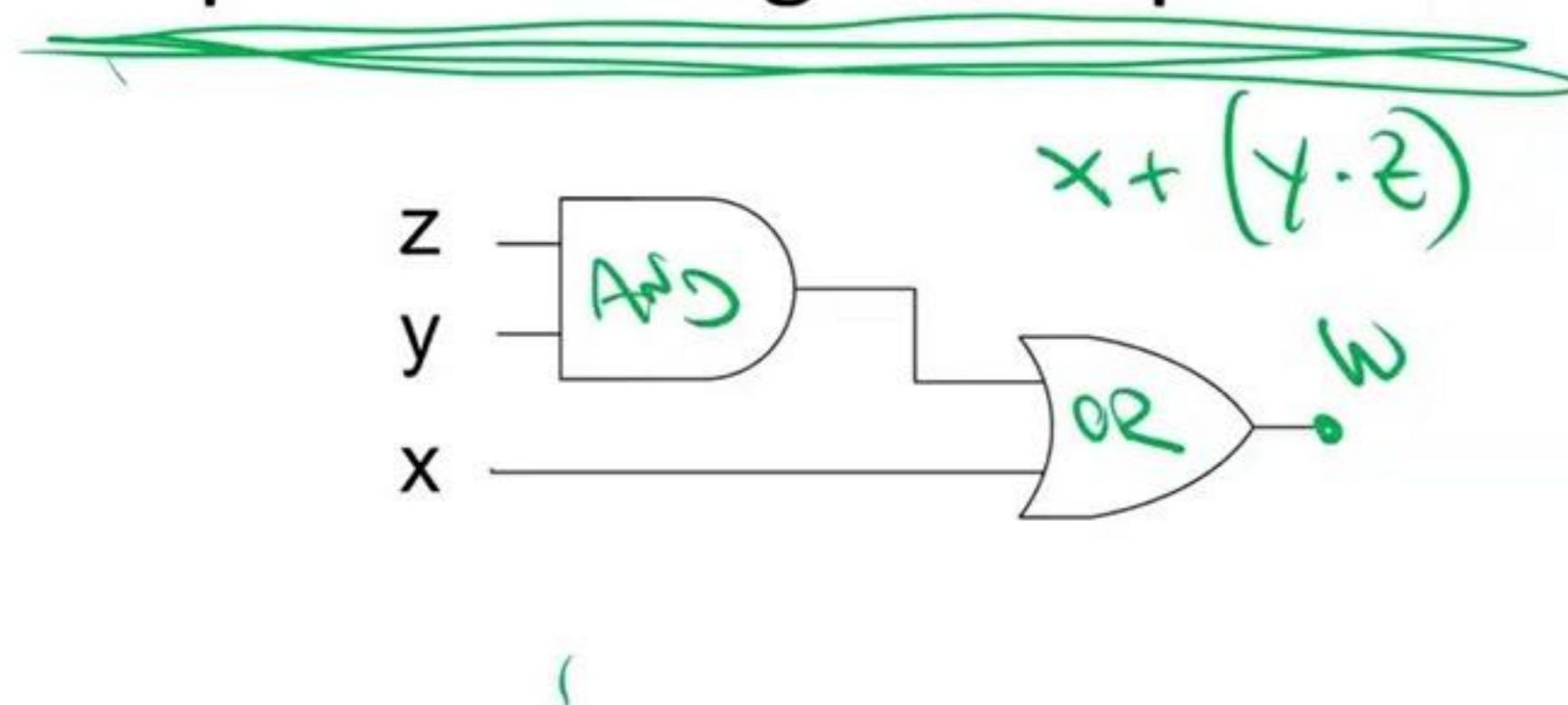
- **data una rete combinatoria** (comunque complessa), è sempre possibile trovare un'espressione booleana che mette in relazione ogni sua uscita con gli ingressi (un'espressione per ogni uscita, in realtà);
- **data un'espressione booleana**, è sempre possibile sintetizzare una rete combinatoria (ad un'uscita) in cui la relazione tra ingresso ed uscita è data dall'espressione.

$$Z = \left( x_3 + \bar{x}_6 \right) \cdot \left( \bar{x}_5 + (x_3 \cdot x_6) \right)$$



# Algebra di Boole e reti combinatorie (cont.)

- Espressioni logiche equivalenti  $\Leftrightarrow$  reti logiche che svolgono lo stesso compito



- Non necessariamente con lo stesso costo

- Le porte logiche dissipano energia, costano, si rompono, ritardano
  - Meno ne mettiamo, meglio è
- Dobbiamo trovare il modo per minimizzare il costo della sintesi di una data legge di corrispondenza

## Esercizio

- Utilizzando le proprietà dell'algebra, semplificare al massimo la seguente espressione

$$\overline{a} \cdot \overline{b} \cdot \overline{c} \cdot \overline{d} + \overline{a} \cdot b \cdot \overline{c} \cdot \overline{d} + a \cdot \overline{b} \cdot \overline{c} \cdot \overline{d} + \overline{a} \cdot b \cdot d + b \cdot c \cdot d + a \cdot b \cdot d \\ + \overline{a} \cdot \overline{b} \cdot \overline{c} \cdot \overline{d}$$

$$= \overline{a} \cdot \overline{c} \cdot \overline{d} + \overline{b} \cdot \overline{c} \cdot \overline{d} + \overline{b} \cdot d + b \cdot c \cdot d$$

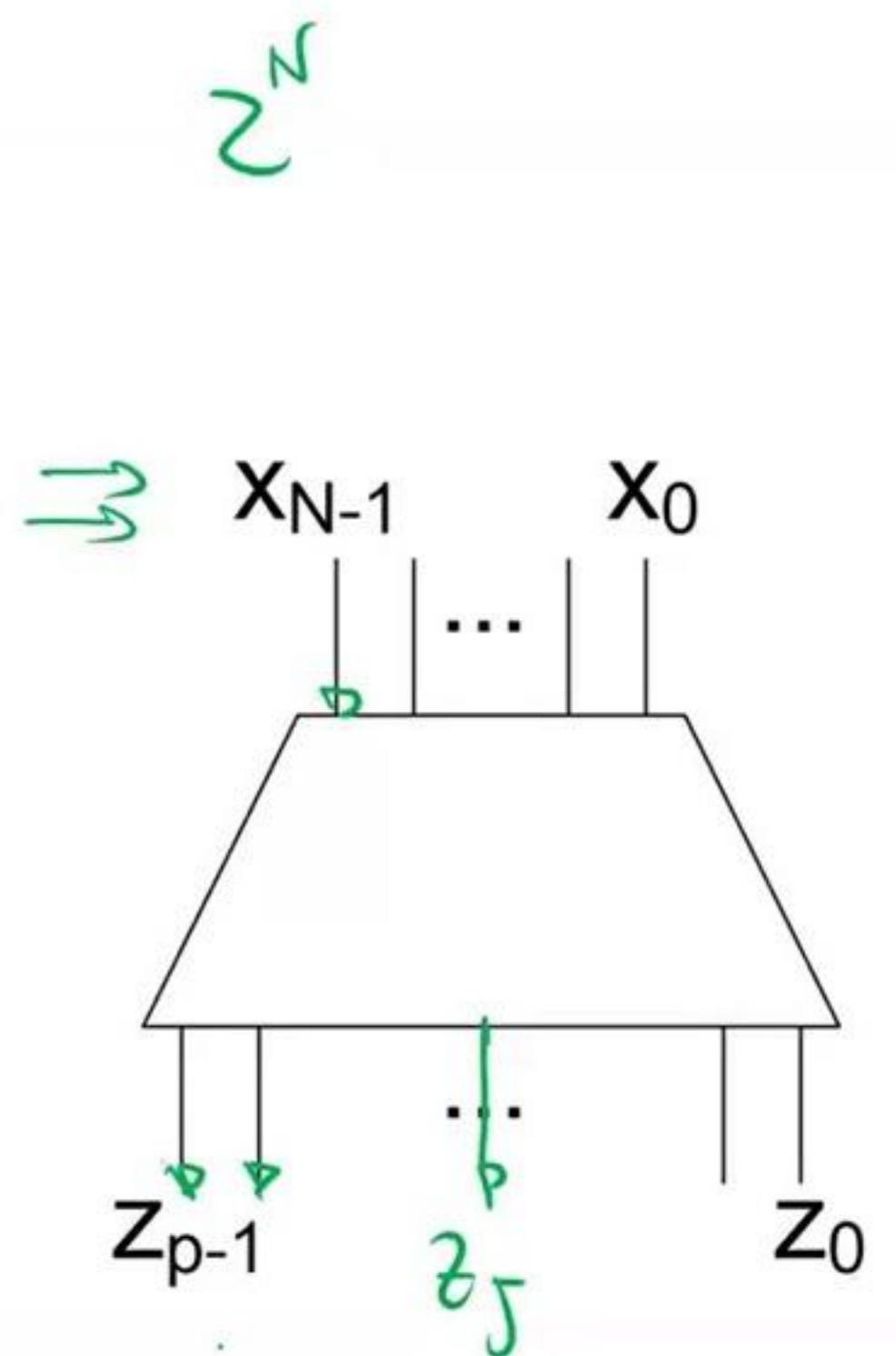
$$= \overline{a} \cdot \overline{c} \cdot \overline{d} + \overline{b} \cdot \overline{c} \cdot \overline{d} + b \cdot d$$

$$\overline{b} \cdot d \cdot \overline{(1+c)} \\ \overline{b+c+d}$$

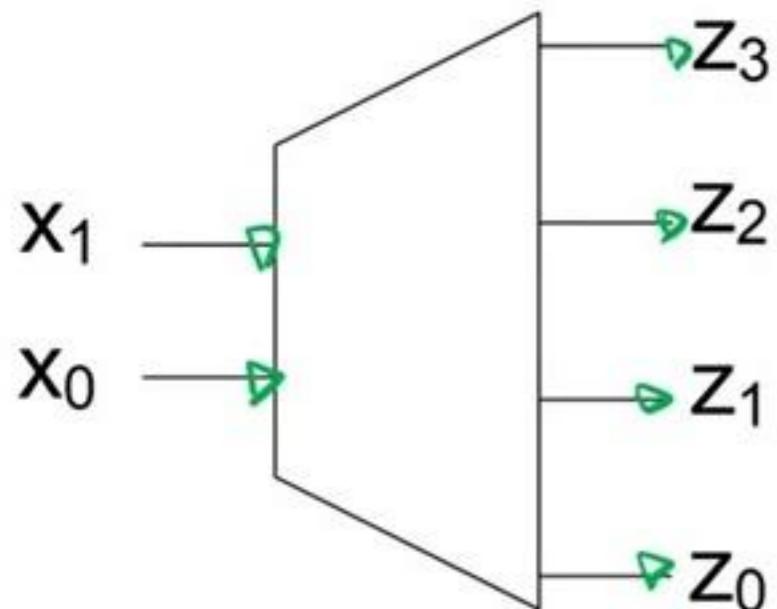
→ Sintesi di costo minimo

# Decoder

- È una rete con  $N$  ingressi e  $p$  uscite, con  $p = \underline{2^N}$ .
- legge di corrispondenza (a parole): “ogni uscita riconosce uno ed un solo stato di ingresso, in particolare l’uscita  $j$ -sima riconosce lo stato di ingresso i cui bit sono la codifica di  $j$  in base 2, cioè se  $(x_{N-1}x_{N-2}\dots x_1x_0)_{b2} \equiv j$ ”



# Esempio: Decoder 2 to 4



X <sub>1</sub>	X <sub>0</sub>	Z <sub>0</sub>	Z <sub>1</sub>	Z <sub>2</sub>	Z <sub>3</sub>
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

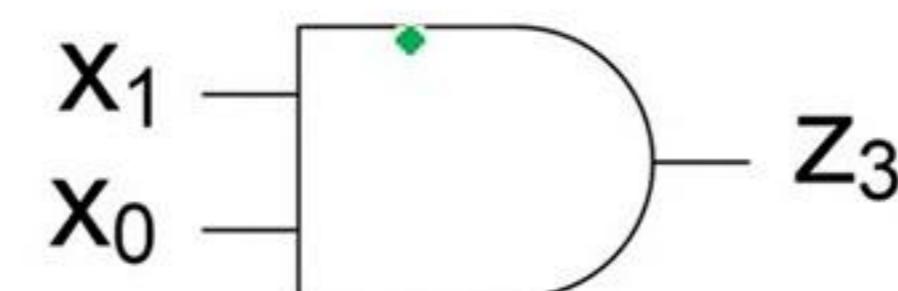
desc

- Come si sintetizza una rete descritta da questa tabella di verita'?
- Prendo in esame l'uscita  $z_3$

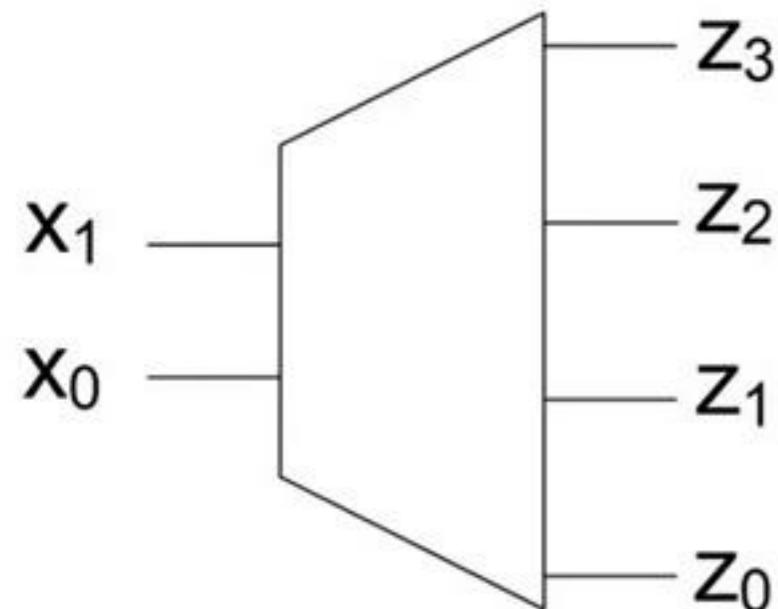
$$z_3 = \begin{cases} 1 & x_1 x_0 = 11 \\ 0 & \text{altrimenti} \end{cases}$$



$$z_3 = x_1 \cdot x_0$$



# Esempio: Decoder 2 to 4



x <sub>1</sub>	x <sub>0</sub>	z <sub>0</sub>	z <sub>1</sub>	z <sub>2</sub>	z <sub>3</sub>
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

$$x_1 \cdot x_0 = 0 \Leftrightarrow \overline{x}_1 \cdot \overline{x}_0 = 1$$

$$z_1 = \overline{x}_1 \cdot x_0$$

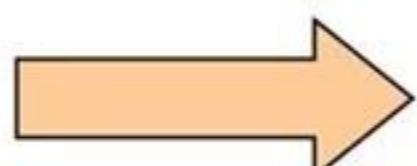
$$z_0 = \overline{x}_1 \cdot \overline{x}_0$$

$$\begin{aligned} x_1 \cdot x_0 &= 0 \\ \overline{x}_1 \cdot \overline{x}_0 &= 1 \end{aligned}$$

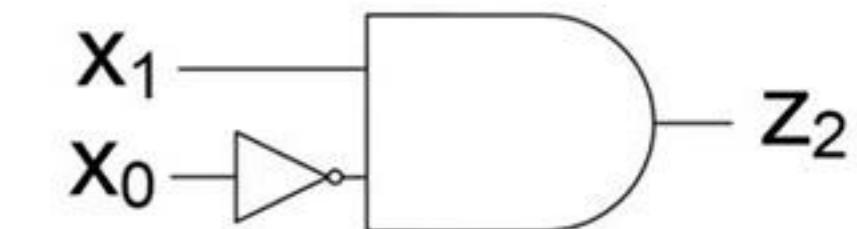
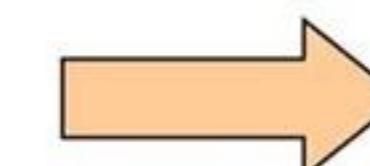
$$z_2 = x_1 \cdot \overline{x}_0$$

- Prendo in esame l'uscita  $z_2$

$$z_2 = \begin{cases} 1 & x_1 \cdot \overline{x}_0 = 1 \\ 0 & \text{altrimenti} \end{cases}$$



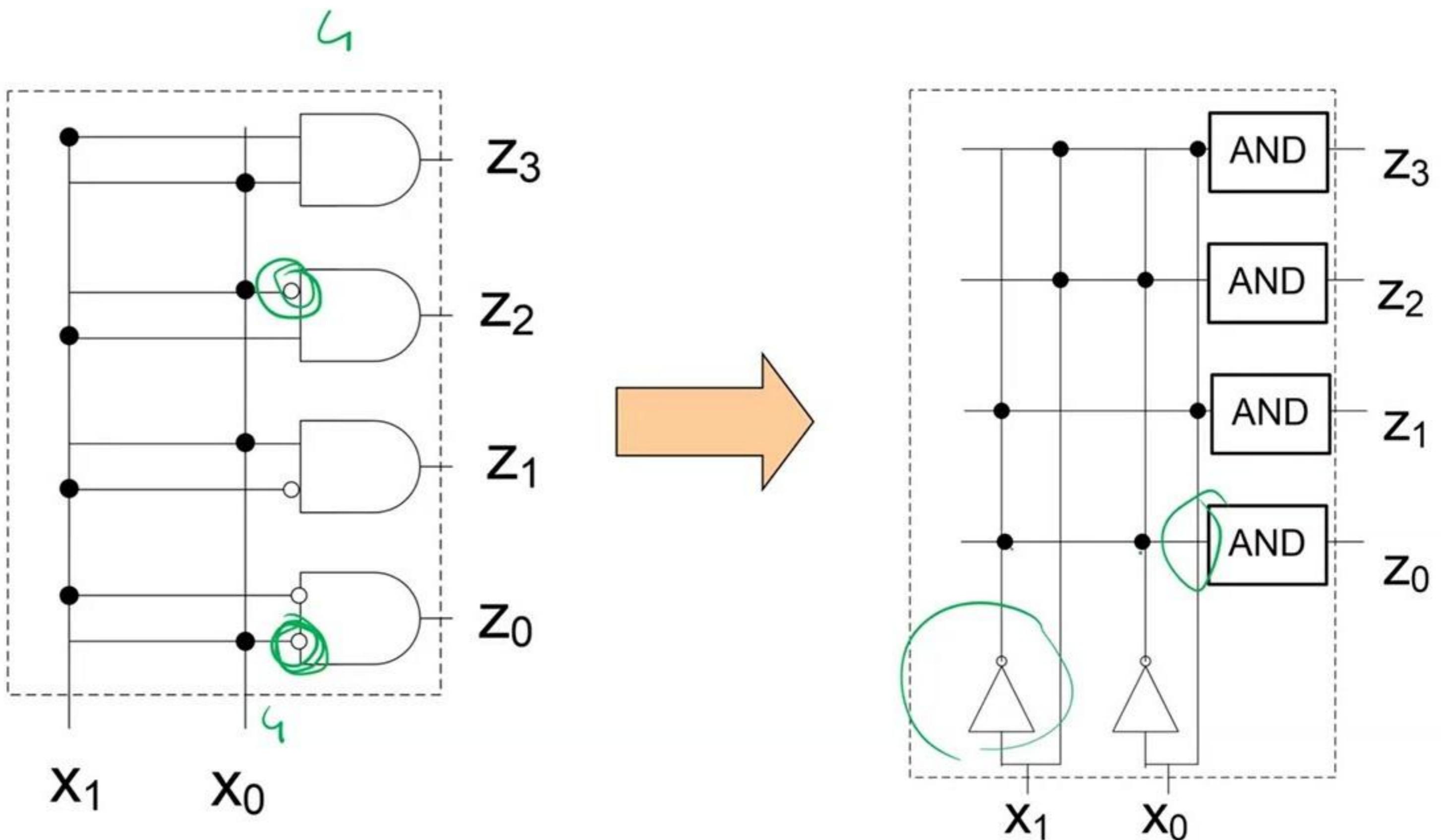
$$z_2 = \begin{cases} 1 & x_1 \cdot \overline{x}_0 = 1 \\ 0 & \text{altrimenti} \end{cases}$$



Lo stesso ragionamento vale per le altre uscite:  $z_1 = \overline{x}_1 \cdot x_0$ ,  $z_0 = \overline{x}_1 \cdot \overline{x}_0$ .

# Sintesi del decoder 2 to 4

- $z_3 = x_1 \cdot x_0$
- $z_2 = x_1 \cdot \overline{x_0}$
- $z_1 = \overline{x_1} \cdot x_0$
- $z_0 = \overline{x_1} \cdot \overline{x_0}$



# Decoder $N$ to $2^N$

$$2^N = P$$

$$z_0 = \overline{x_{N-1}} \cdot \overline{x_{N-2}} \cdot \dots \cdot \overline{x_1} \cdot \overline{x_0}$$

$$z_1 = \overline{x_{N-1}} \cdot \overline{x_{N-2}} \cdot \dots \cdot \overline{x_1} \cdot x_0$$

...

$$\text{III...IV}$$

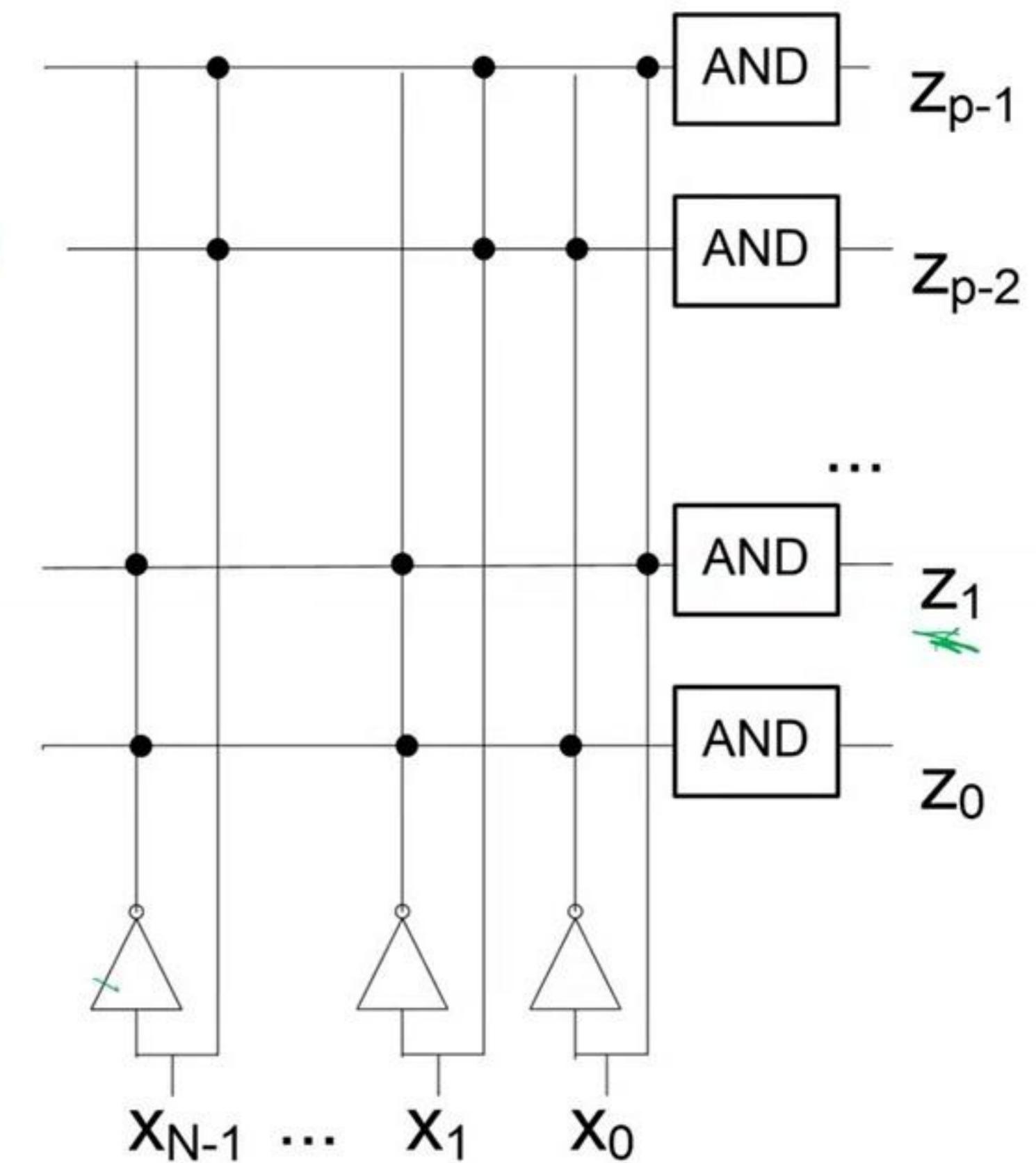
$$\text{III...IO} \rightarrow$$

$$\text{OO...O1}$$

$$\text{OO...O9}$$

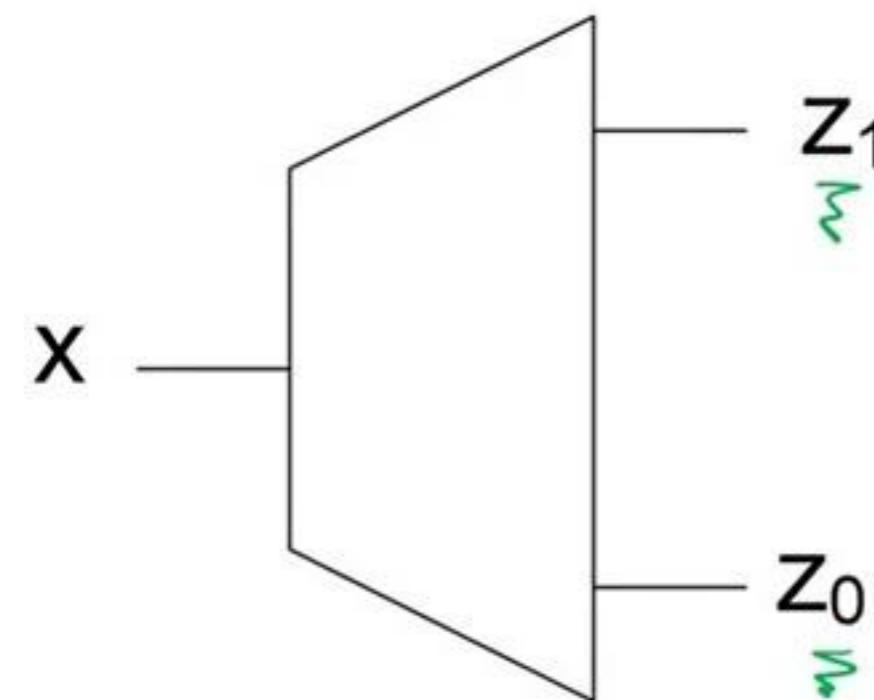
$$z_{p-2} = x_{N-1} \cdot x_{N-2} \cdot \dots \cdot x_1 \cdot \overline{x_0}$$

$$\rightarrow z_{p-1} = x_{N-1} \cdot x_{N-2} \cdot \dots \cdot x_1 \cdot x_0$$

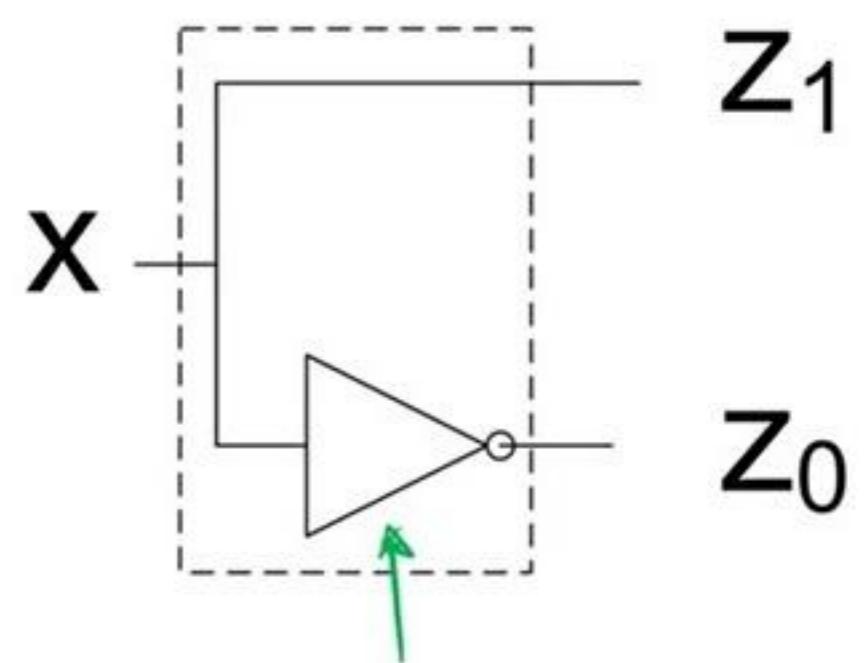


# Esempio: Decoder 1 to 2

- Caso particolare, non richiede logica



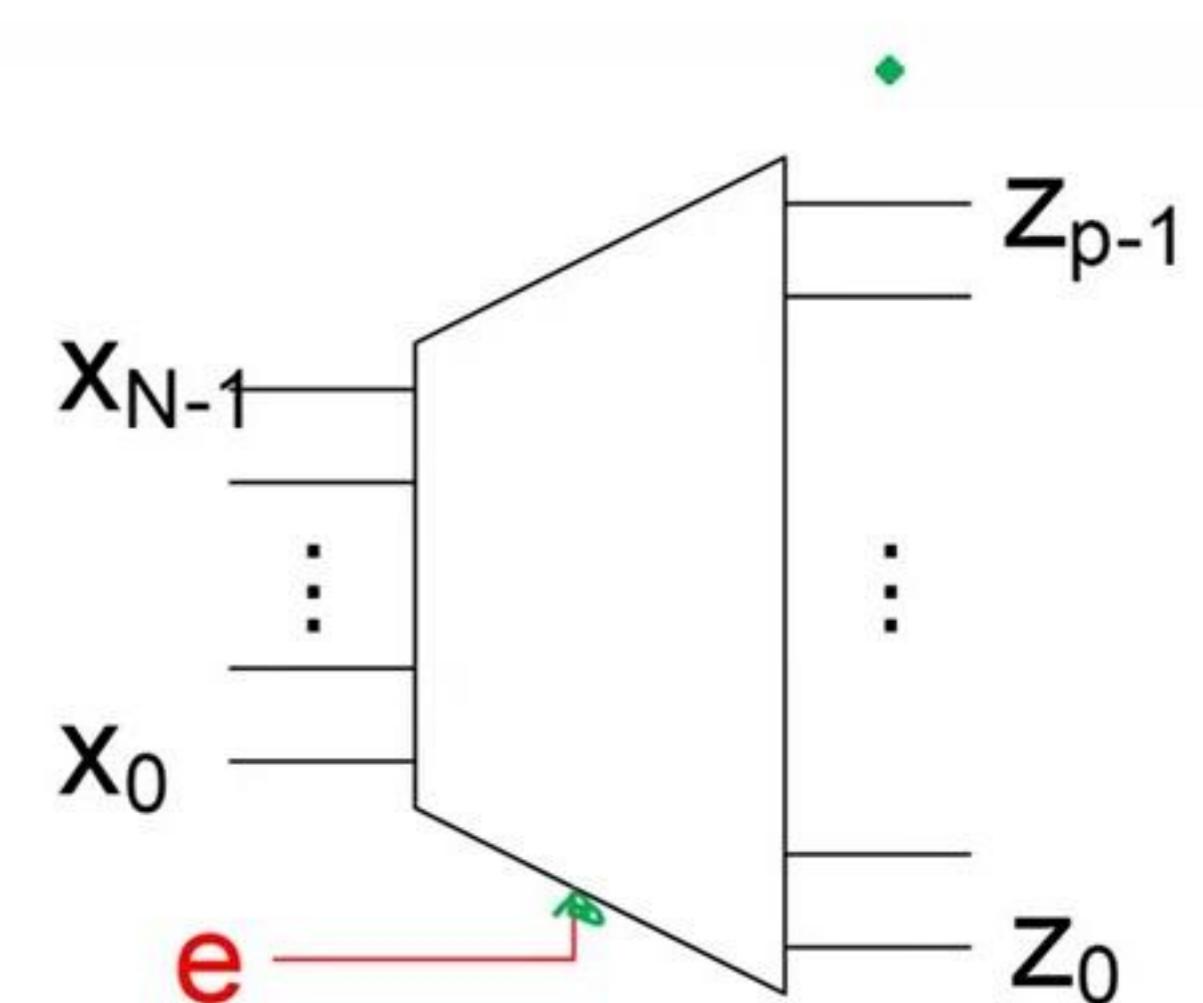
X	$z_0$	$z_1$
0	1	0
1	0	1



# Decoder con enabler (espandibile)

- Il decoder che abbiamo appena descritto **non è espandibile**.
- Non posso costruire decoder grandi combinando decoder **più piccoli**.
- Per questo motivo, esistono decoder dotati di un ingresso aggiuntivo, detto **di abilitazione (enabler)**

- Ha  $N + 1$  ingressi e  $2^N$  uscite
- se l'ingresso di **enabler** è 1, la rete si comporta come un decoder  $N$  to  $2^N$ . Altrimenti, tutte le uscite sono a 0
- l'ingresso di abilitazione “accende” il decoder



# Sintesi del decoder con enabler

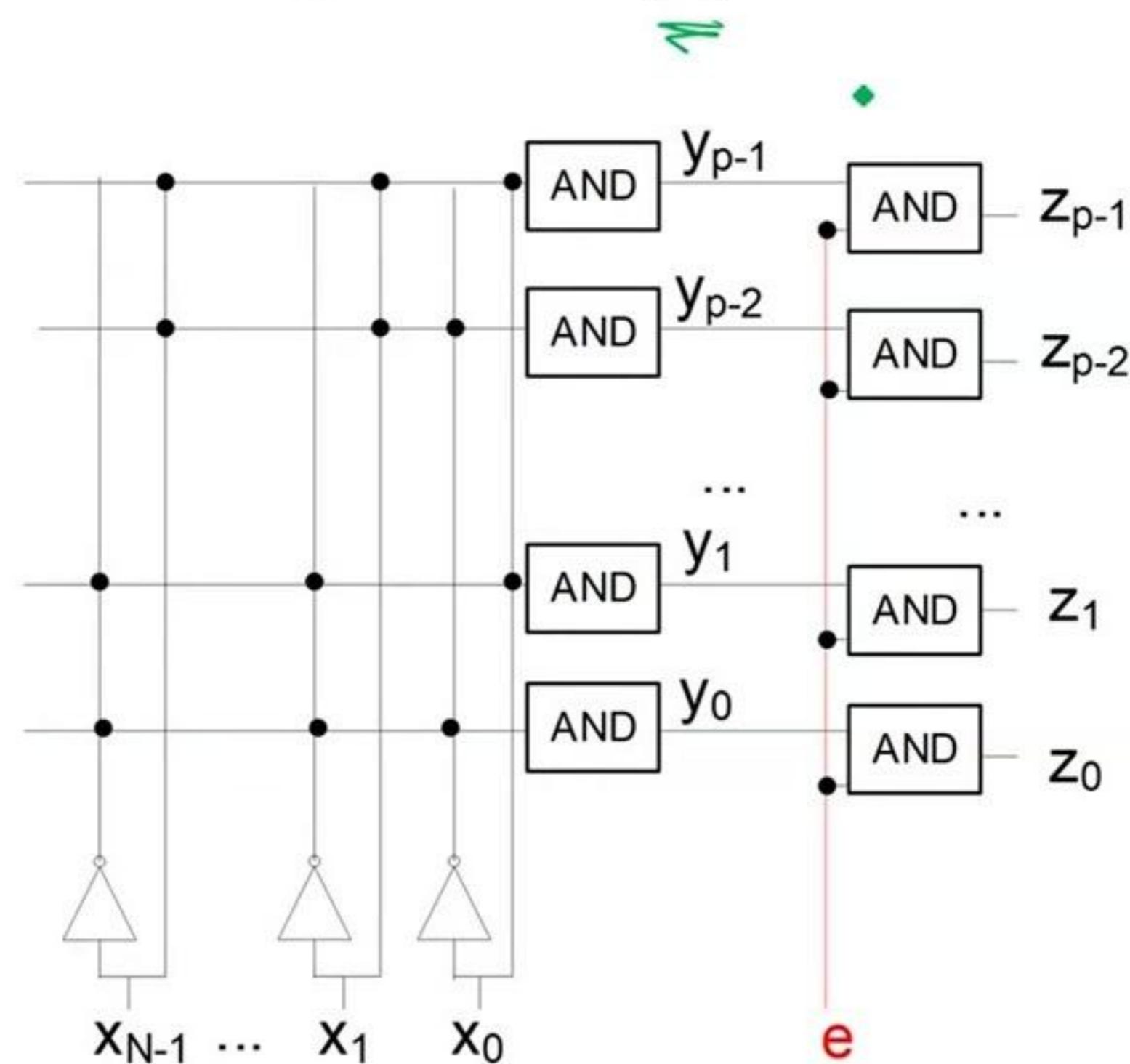
- Prendiamo un decoder senza enabler, e chiamiamo  $y_0 \dots y_{p-1}$  le sue uscite
- Ciascuna uscita  $z_i$  vale 0 se  $e = 0$ , altrimenti è uguale ad  $y_i$ .

$$z_i = \begin{cases} y_i & e = 1 \\ 0 & e = 0 \end{cases}$$

$$z_i = y_i \cdot e$$

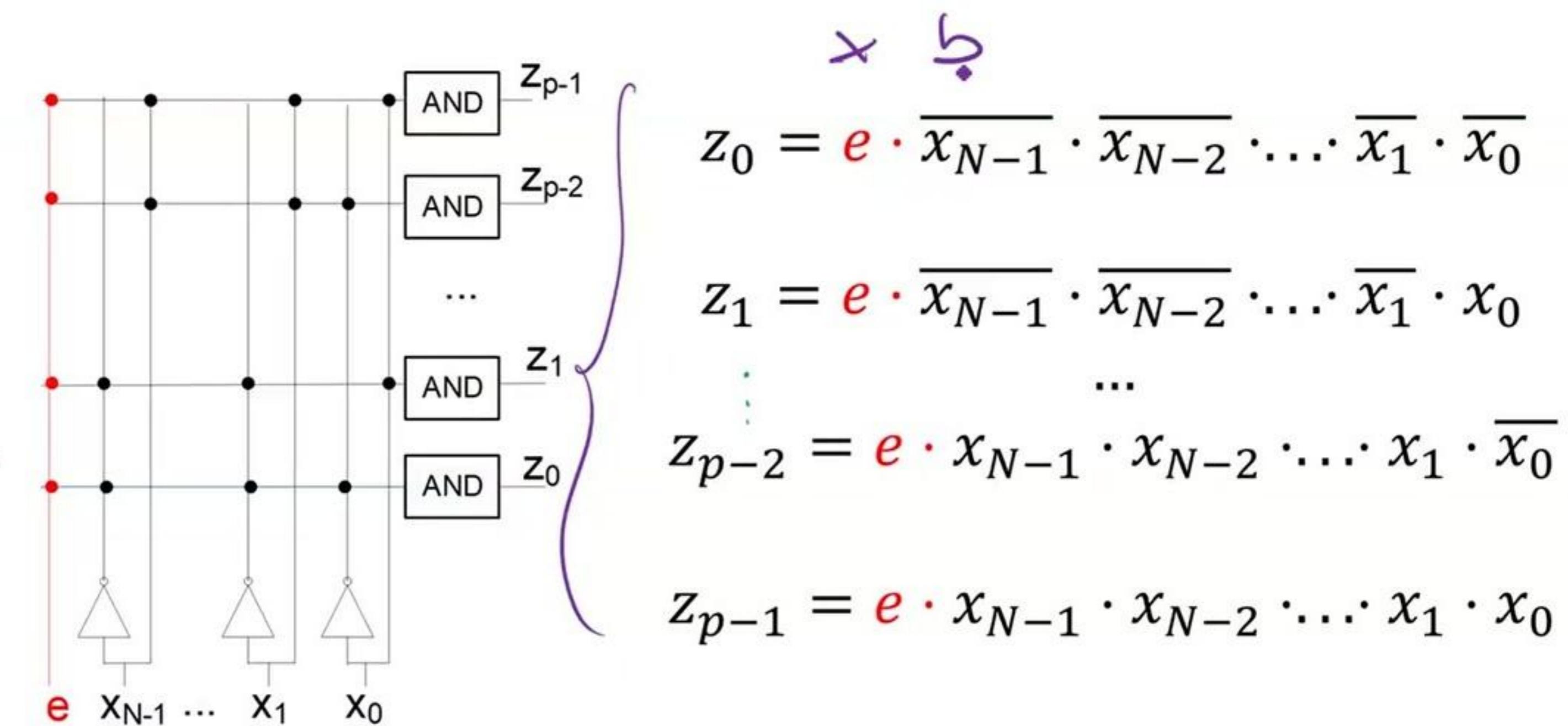
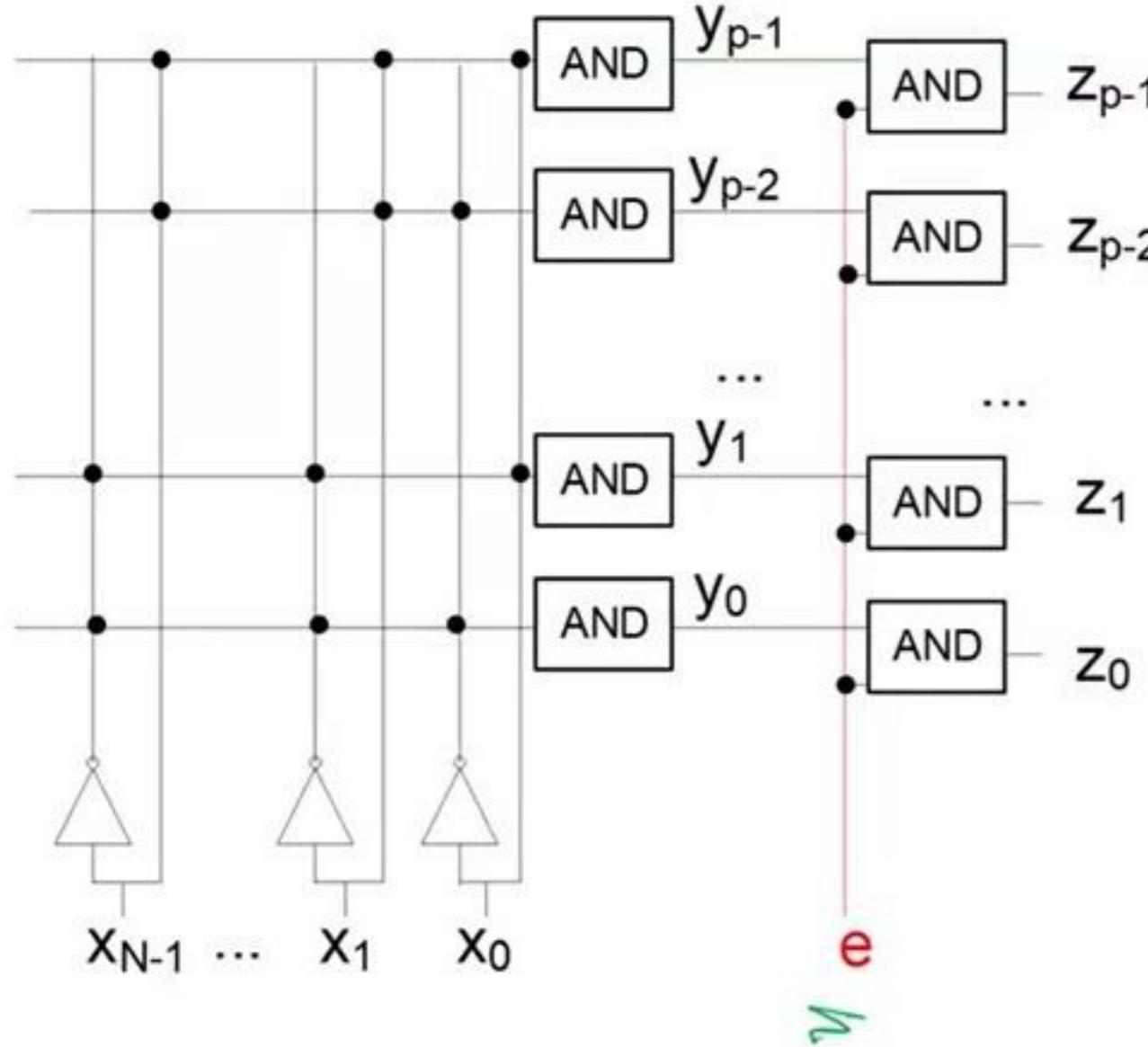
- Dato che:  $x \cdot 0 = 0, x \cdot 1 = x$

$$z_i = y_i \cdot e$$



# Sintesi del decoder con enabler

- Due porte AND in sequenza sono uno spreco
- Meglio aggiungere un ingresso (l'AND è associativo)



$$z_0 = e \cdot \overline{x_{N-1}} \cdot \overline{x_{N-2}} \cdots \overline{x_1} \cdot \overline{x_0}$$

$$z_1 = e \cdot \overline{x_{N-1}} \cdot \overline{x_{N-2}} \cdots \overline{x_1} \cdot x_0$$

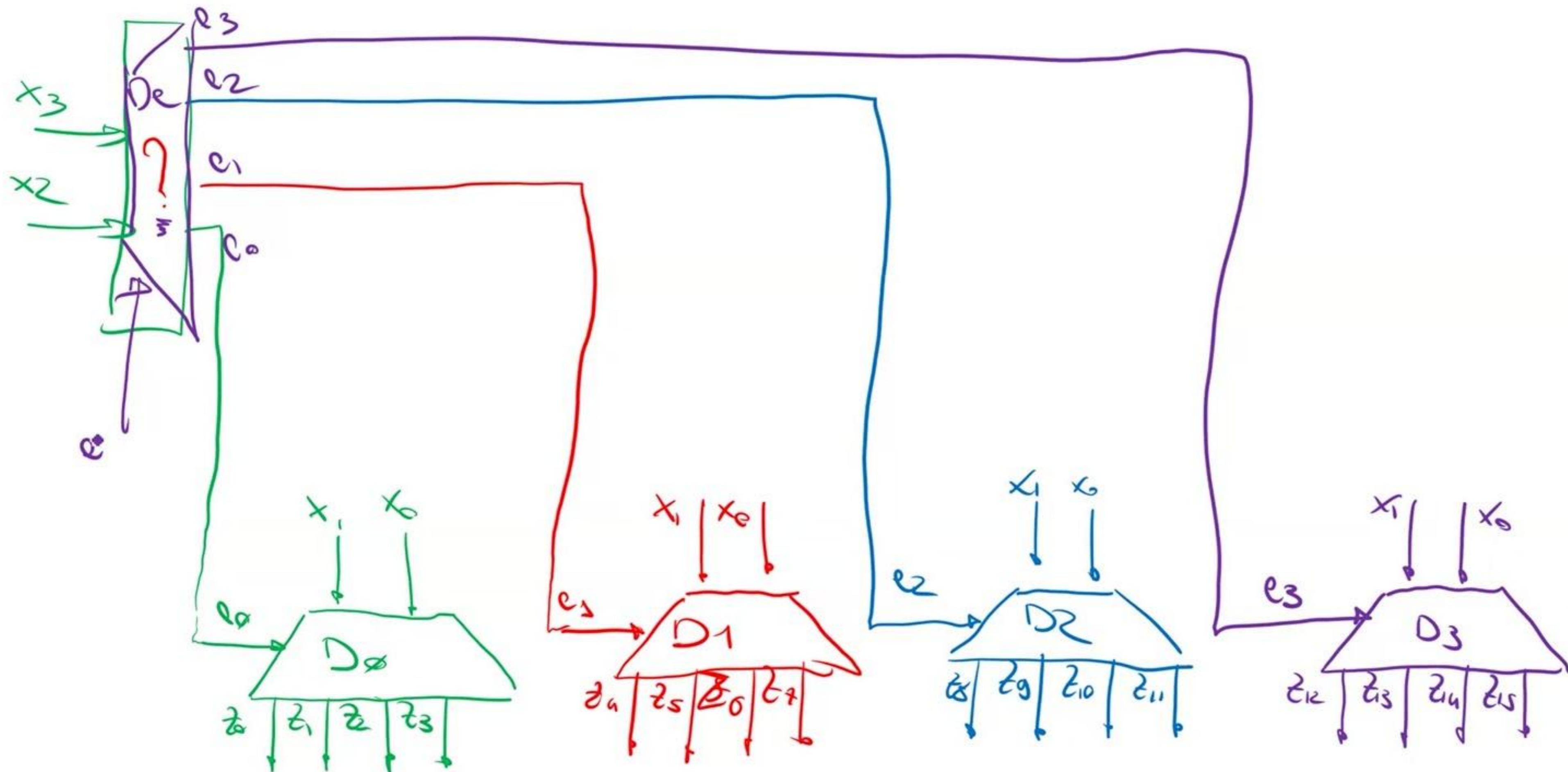
$$\vdots$$
$$z_{p-2} = e \cdot x_{N-1} \cdot x_{N-2} \cdots x_1 \cdot \overline{x_0}$$

$$z_{p-1} = e \cdot x_{N-1} \cdot x_{N-2} \cdots x_1 \cdot x_0$$

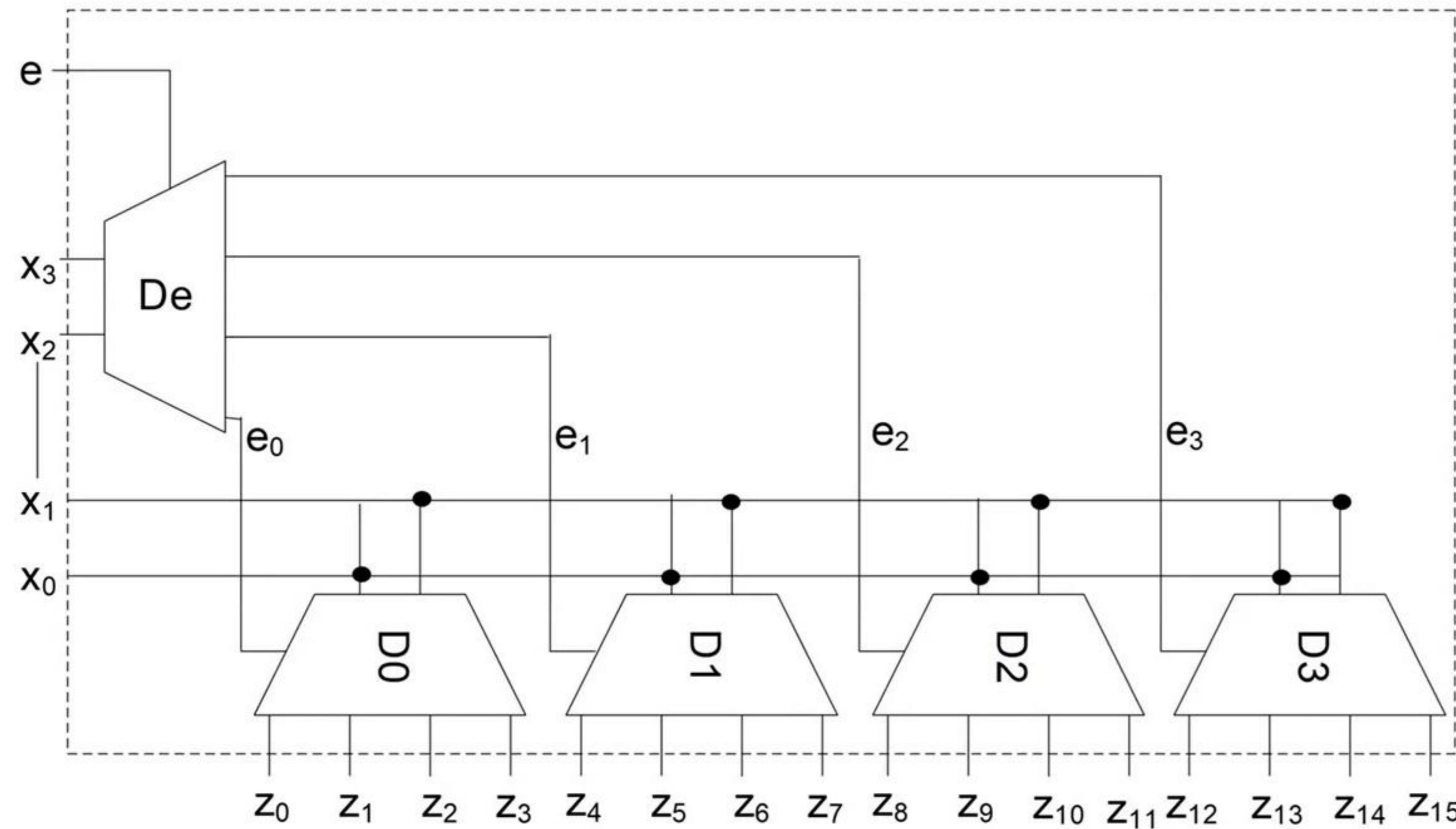
# Costruzione di decoder 4to16 da decoder 2to4

X <sub>3</sub>	X <sub>2</sub>	X <sub>1</sub>	X <sub>0</sub>	Z <sub>0</sub>	Z <sub>1</sub>	Z <sub>2</sub>	Z <sub>3</sub>	Z <sub>4</sub>	Z <sub>5</sub>	Z <sub>6</sub>	Z <sub>7</sub>	Z <sub>8</sub>	Z <sub>9</sub>	Z <sub>10</sub>	Z <sub>11</sub>	Z <sub>12</sub>	Z <sub>13</sub>	Z <sub>14</sub>	Z <sub>15</sub>
0	0	0	0	1	0	0	0												
0	0	0	1	0	1	0	0												
0	0	1	0	0	0	1	0												
0	0	1	1	0	0	0	1												
0	1	0	0	0	0	0	0	1	0	0	0								
0	1	0	1					0	1	0	0								
0	1	1	0					0	0	1	0								
0	1	1	1					0	0	0	1								
1	0	0	0									1	0	0	0				
1	0	0	1									0	1	0	0				
1	0	1	0									0	0	1	0				
1	0	1	1									0	0	0	1				
1	1	0	0												1	0	0	0	
1	1	0	1												0	1	0	0	
1	1	1	0												0	0	1	0	
1	1	1	1												0	0	0	1	

# Costruzione di decoder 4to16 da decoder 2to4



# Costruzione di decoder 4to16 da decoder 2to4



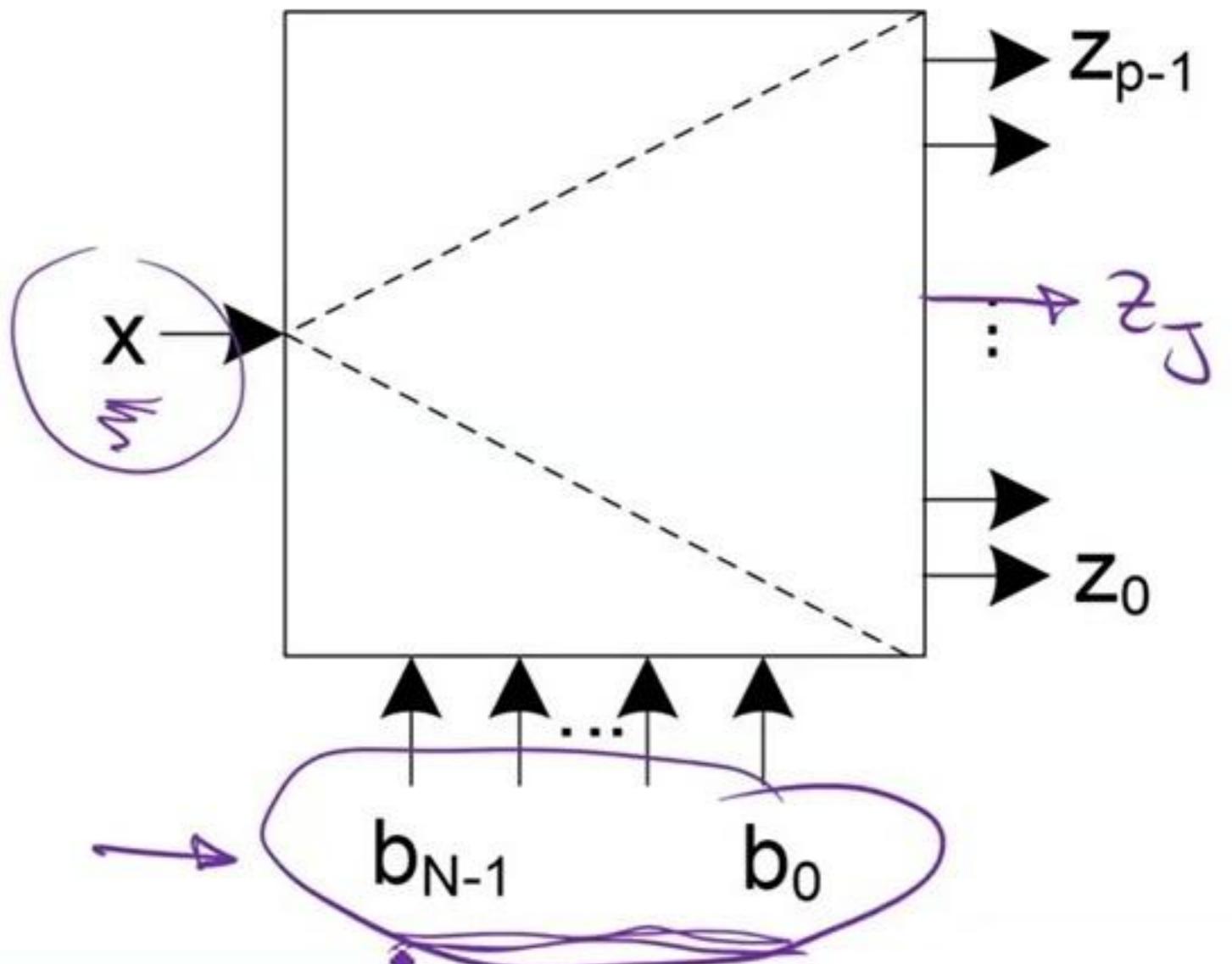
# Demultiplexer

- Rete con  $N + 1$  ingressi e  $p = 2^N$  uscite
  - $x$  si chiama **variabile da commutare**,
  - gli altri si chiamano **variabili di comando**

- la  $j$ -sima uscita **insegue** la variabile da commutare se e solo se

$$(b_{N-1} b_{N-2} \dots b_1 b_0)_2 \equiv j$$

- altrimenti vale 0



$$\overline{b_{N-1}} \cdot \overline{b_{N-2}} \cdot \dots \cdot \overline{b_1} \cdot \overline{b_0} = 1$$

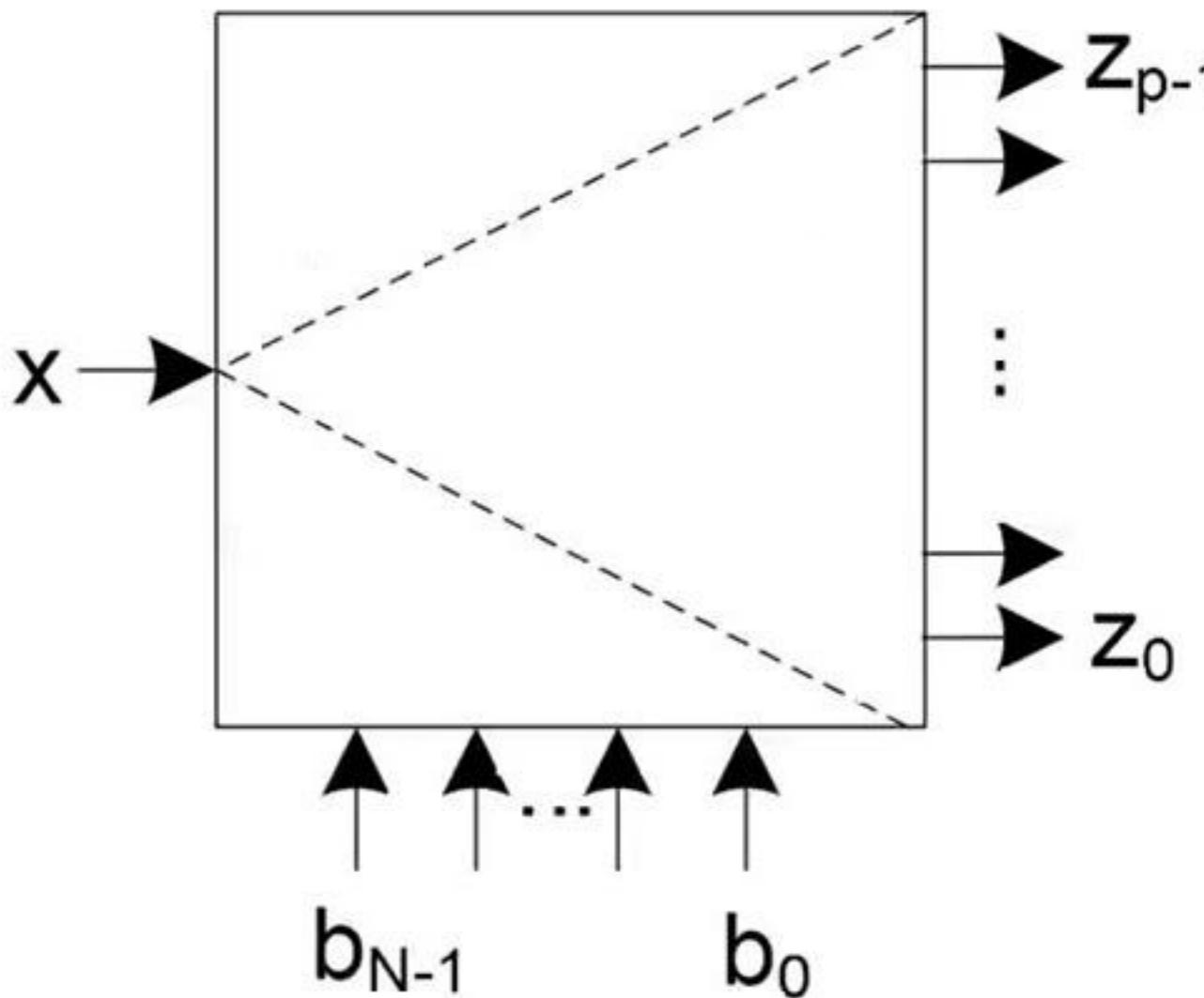
## Demultiplexer (cont.)

- Descrizione in termini algebrici

$$\overline{b_{N-1}} \cdot \overline{b_{N-2}} \cdot \dots \cdot \overline{b_1} \cdot \overline{b_0} = 1 \dots 1$$

- $z_0 = \begin{cases} x & (b_{N-1} \dots b_1 b_0) = (0 \dots 00) \\ 0 & \text{altrimenti} \end{cases}$

$$( \quad )^{x \cdot K}$$



# Demultiplexer (cont.)

- Descrizione in termini algebrici

$$\rightarrow z_0 = \underbrace{x \cdot \overline{b_{N-1}} \cdot \overline{b_{N-2}} \cdots \overline{b_1} \cdot \overline{b_0}}$$

$$z_1 = \underbrace{x \cdot \overline{b_{N-1}} \cdot \overline{b_{N-2}} \cdots \overline{b_1} \cdot b_0}$$

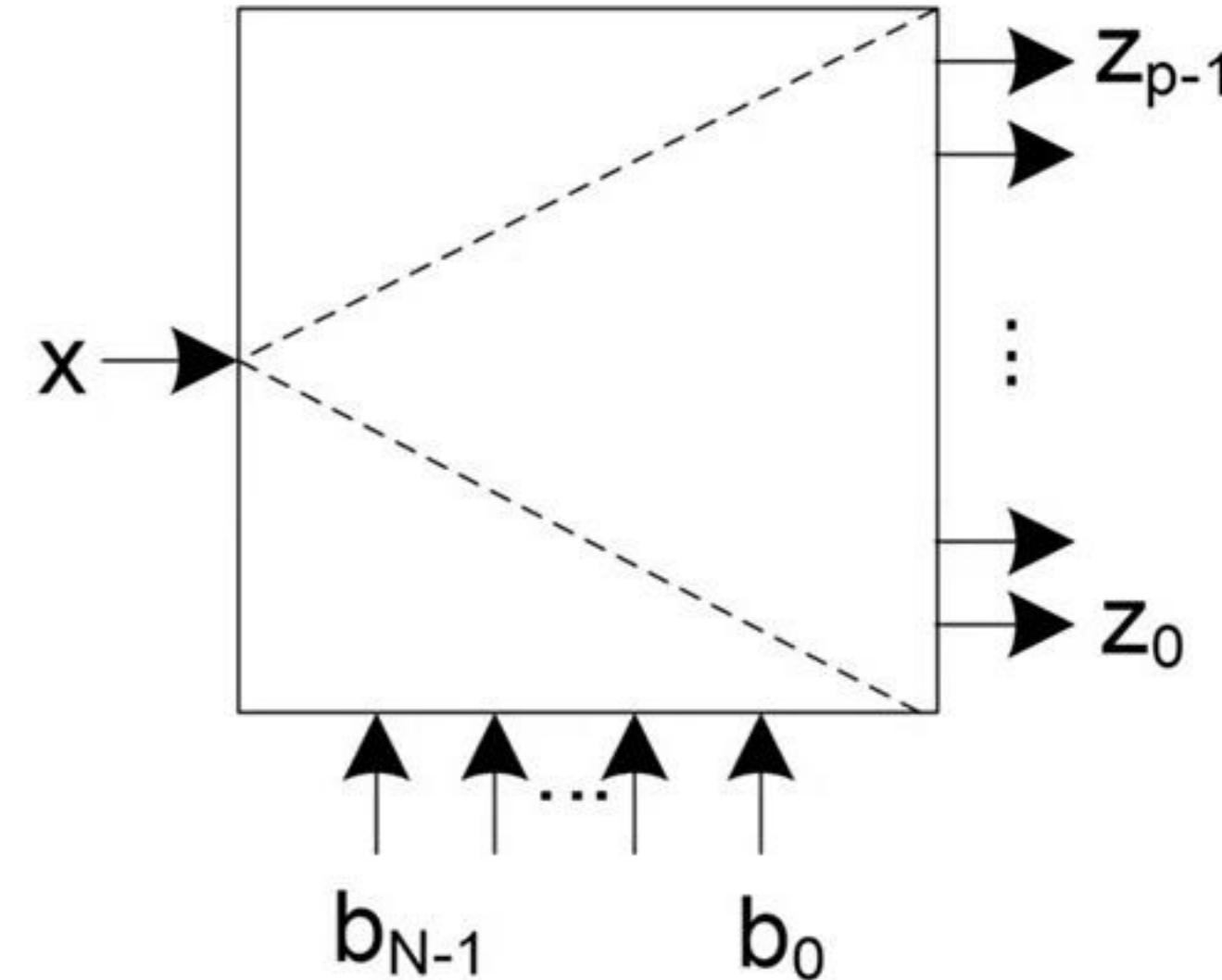
...

$$z_{p-2} = x \cdot b_{N-1} \cdot b_{N-2} \cdot \dots \cdot b_1 \cdot \overline{b_0}$$

$$\rightarrow z_{p-1} = x \cdot b_{N-1} \cdot b_{N-2} \cdot \dots \cdot b_1 \cdot b_0$$

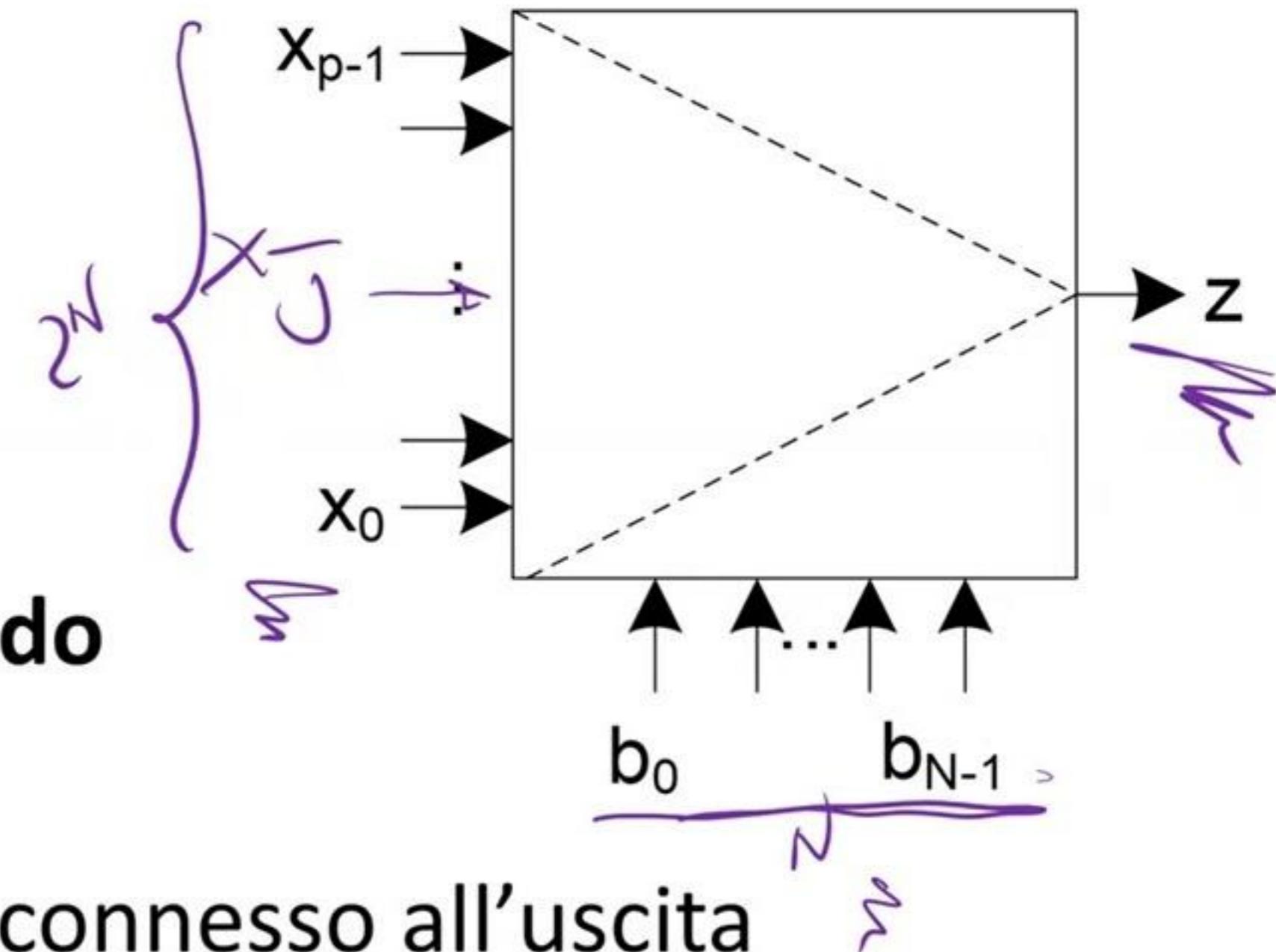
e

- identica a quella di un decoder con enabler



# Multiplexer

- rete con  $N + 2^N$  ingressi ed 1 uscita
- ingressi  $b_i$  si chiamano **variabili di comando**
- duale del demultiplexer
- Le var di comando selezionano l'ingresso connesso all'uscita



$$z = x_i \Leftrightarrow (b_{N-1} \dots b_1 b_0)_2 \equiv i$$

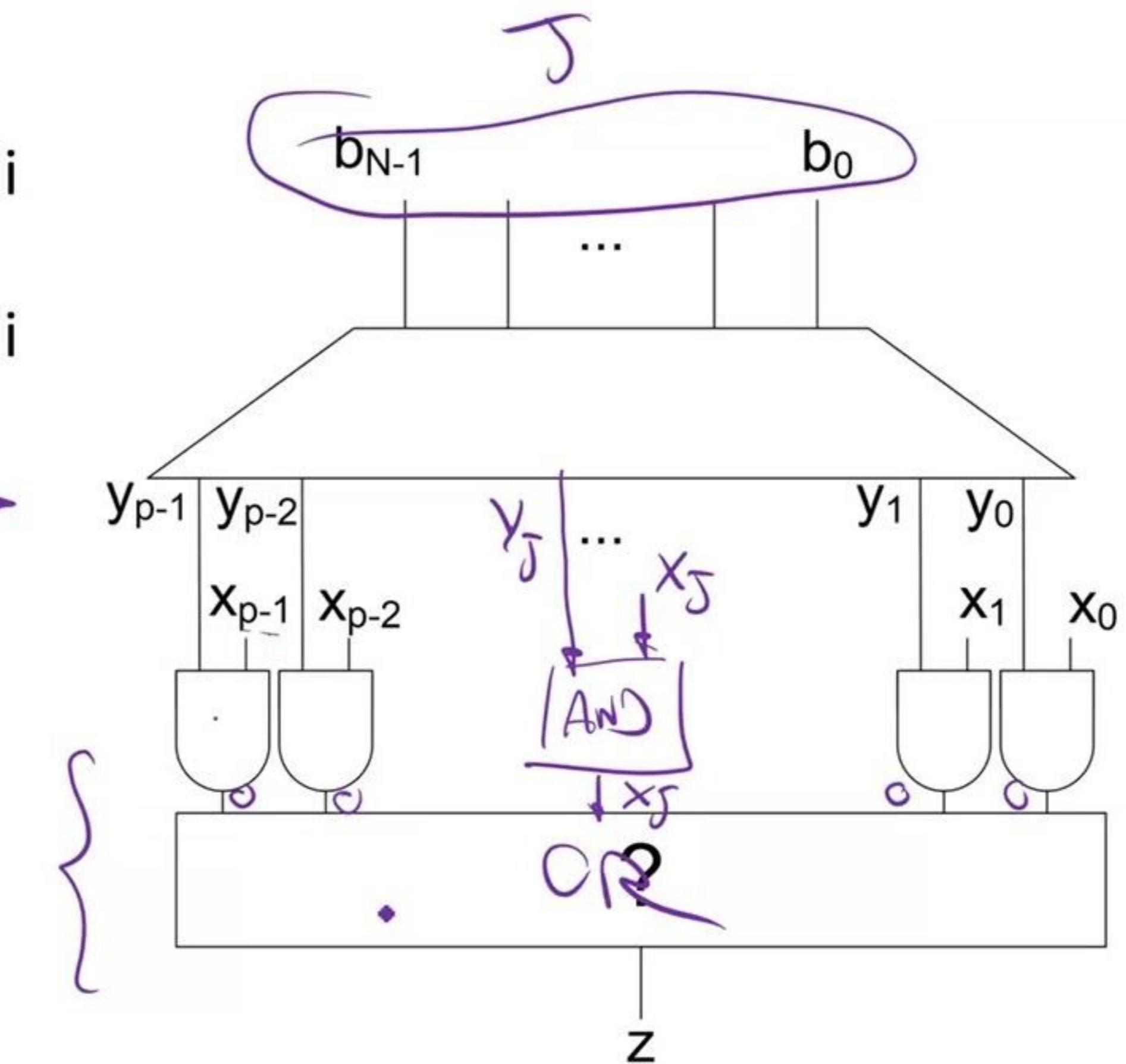
- Mux/Demux usati (molti anni fa) nelle centrali telefoniche

descrizione

sintesi

# Multiplexer (cont.)

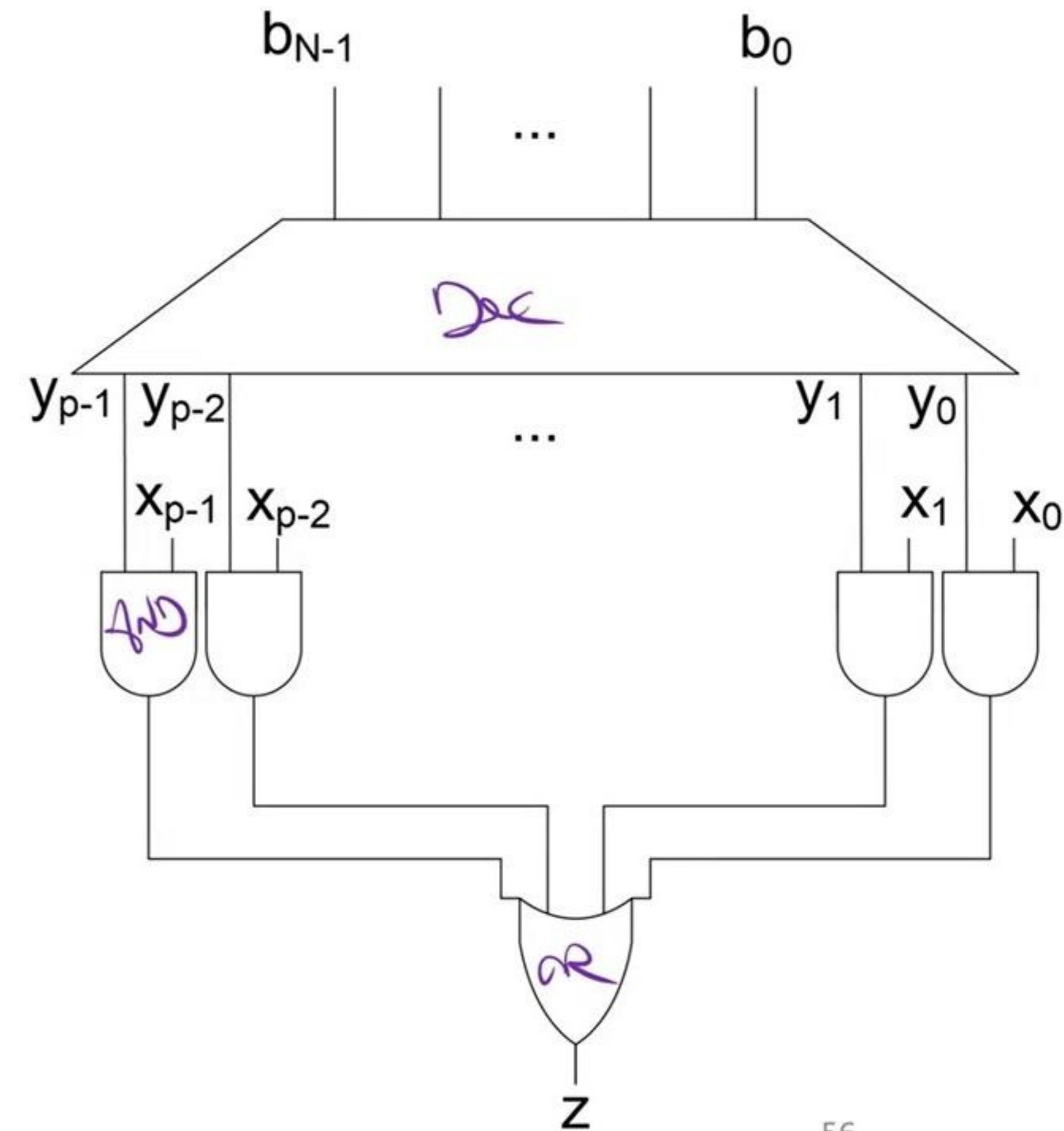
- Partiamo dalla **sintesi** (piu' semplice)
  - **decoder**  $N$  to  $2^N$ , come ingressi ha le var di comando
  - metto in AND a ciascuna uscita una delle var di ingresso
- $2^N$  variabili logiche in uscita:
  - tutte meno una sono sicuramente nulle.
  - quell'unica che puo' non essere nulla (sia la  $j$ -sima) vale  $x_j$



# Multiplexer (cont.)

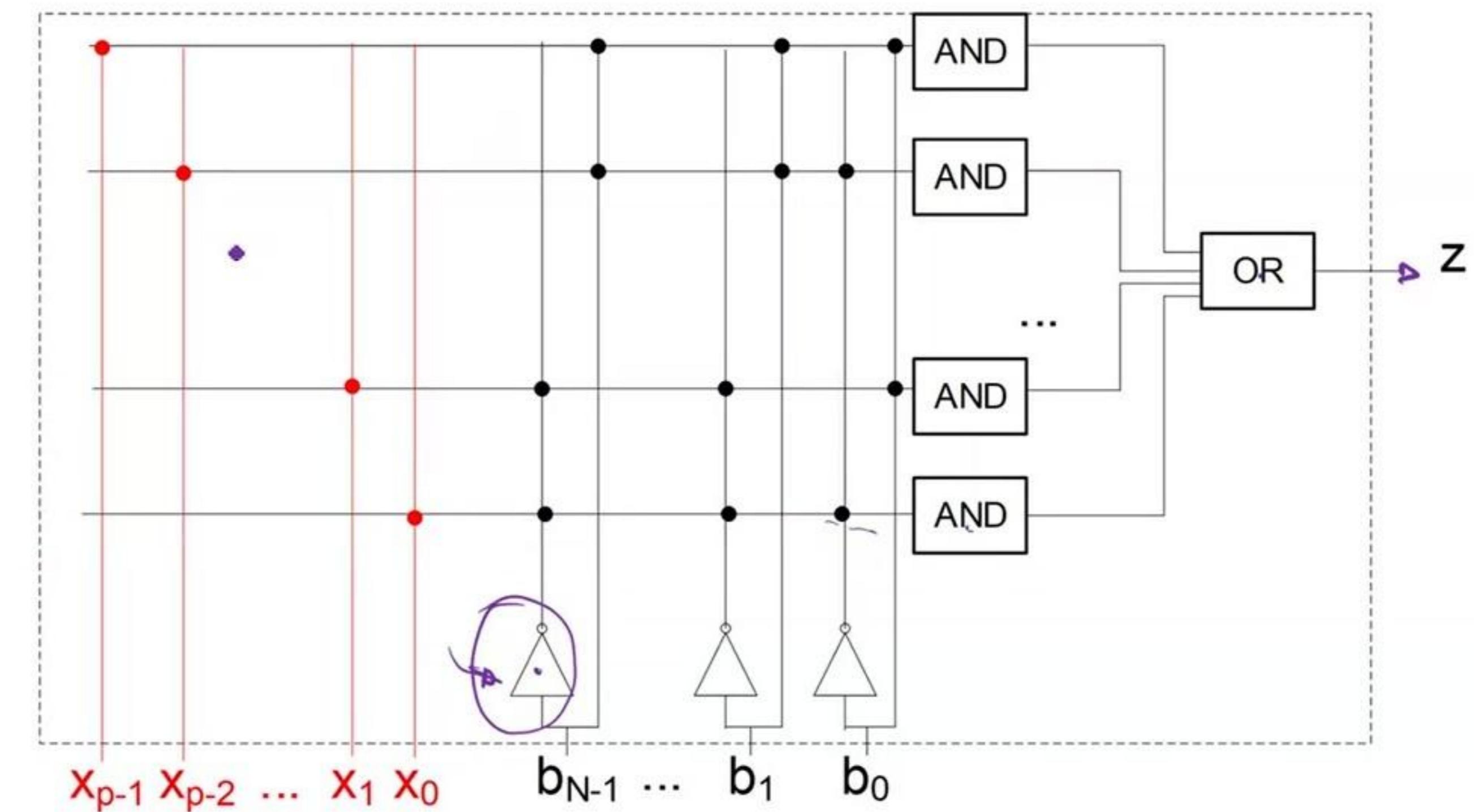
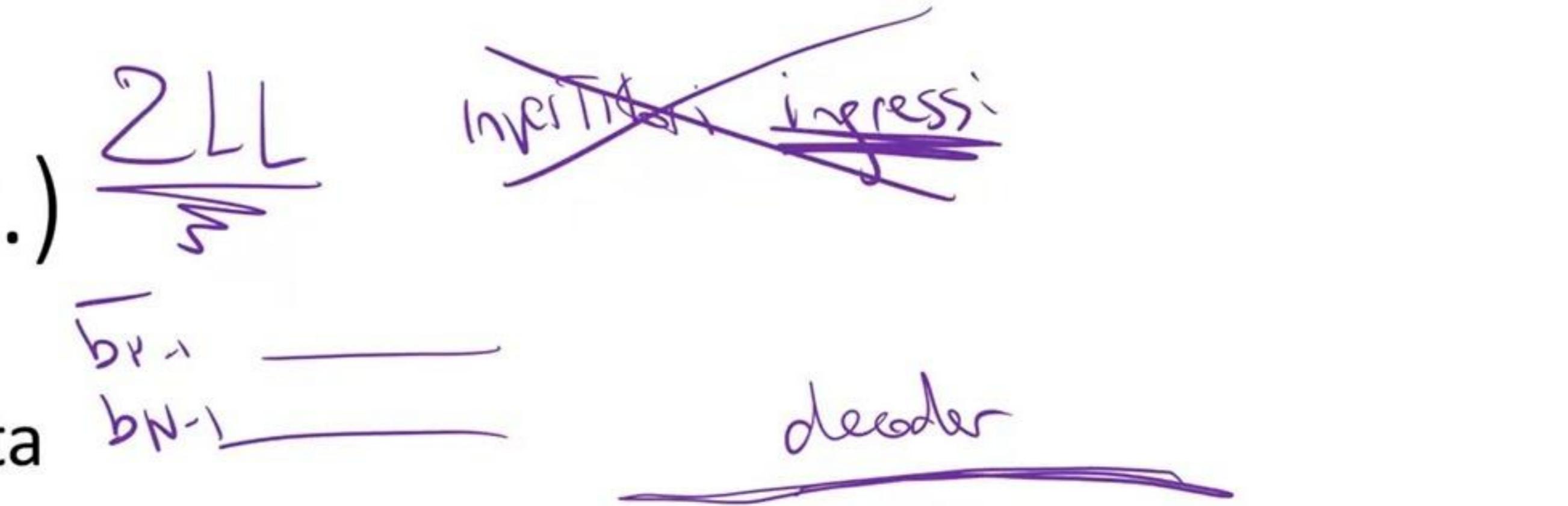
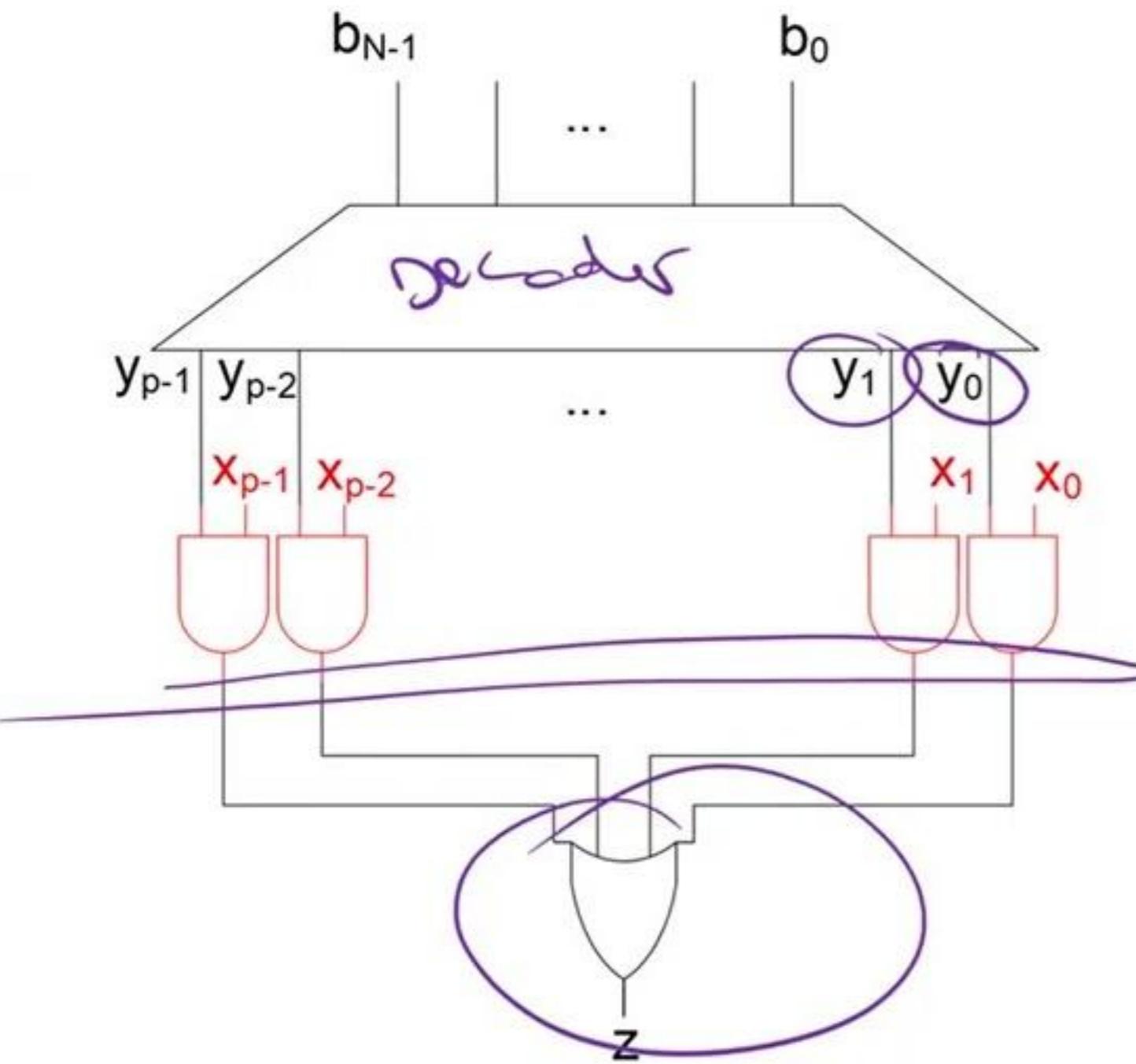
- Come faccio ad ottenere l'uscita  $z$  con queste variabili?
- Basta ricordare che  $0 + \alpha = \alpha$ . Infatti, di queste  $p$  variabili,  $p - 1$  sono sicuramente nulle
- Metto una OR in fondo

$$P=2^N$$



# Multiplexer (cont.)

- Eliminiamo AND in cascata



# Multiplexer (cont.)

- Dal disegno si ricava facilmente una descrizione algebrica

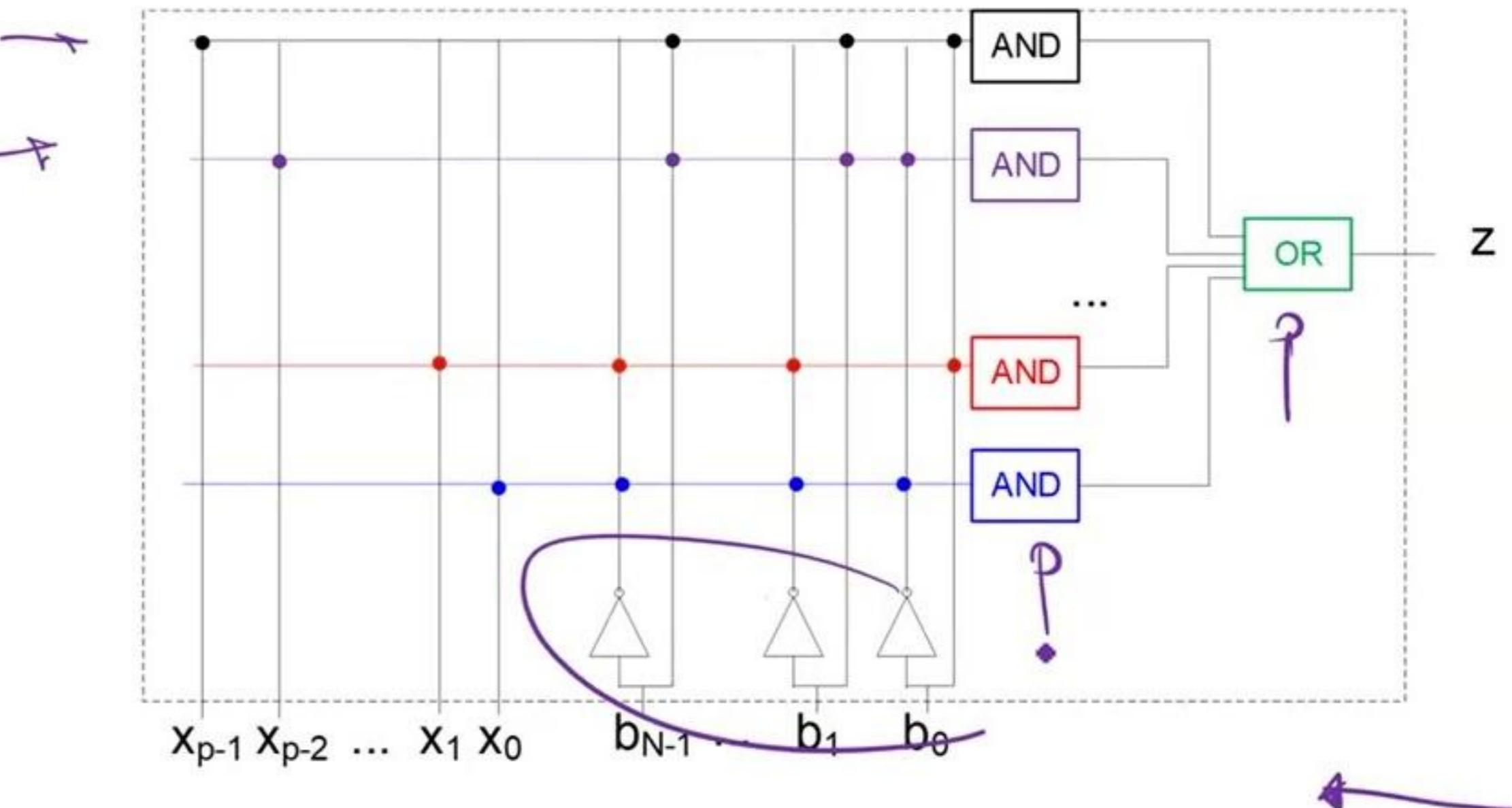
$$z = x_0 \cdot \overline{b_{N-1}} \cdot \overline{b_{N-2}} \cdots \overline{b_1} \cdot \overline{b_0} +$$

$$x_1 \cdot \overline{b_{N-1}} \cdot \overline{b_{N-2}} \cdots \overline{b_1} \cdot b_0 +$$

...

$$x_{p-2} \cdot b_{N-1} \cdot b_{N-2} \cdots b_1 \cdot \overline{b_0} +$$

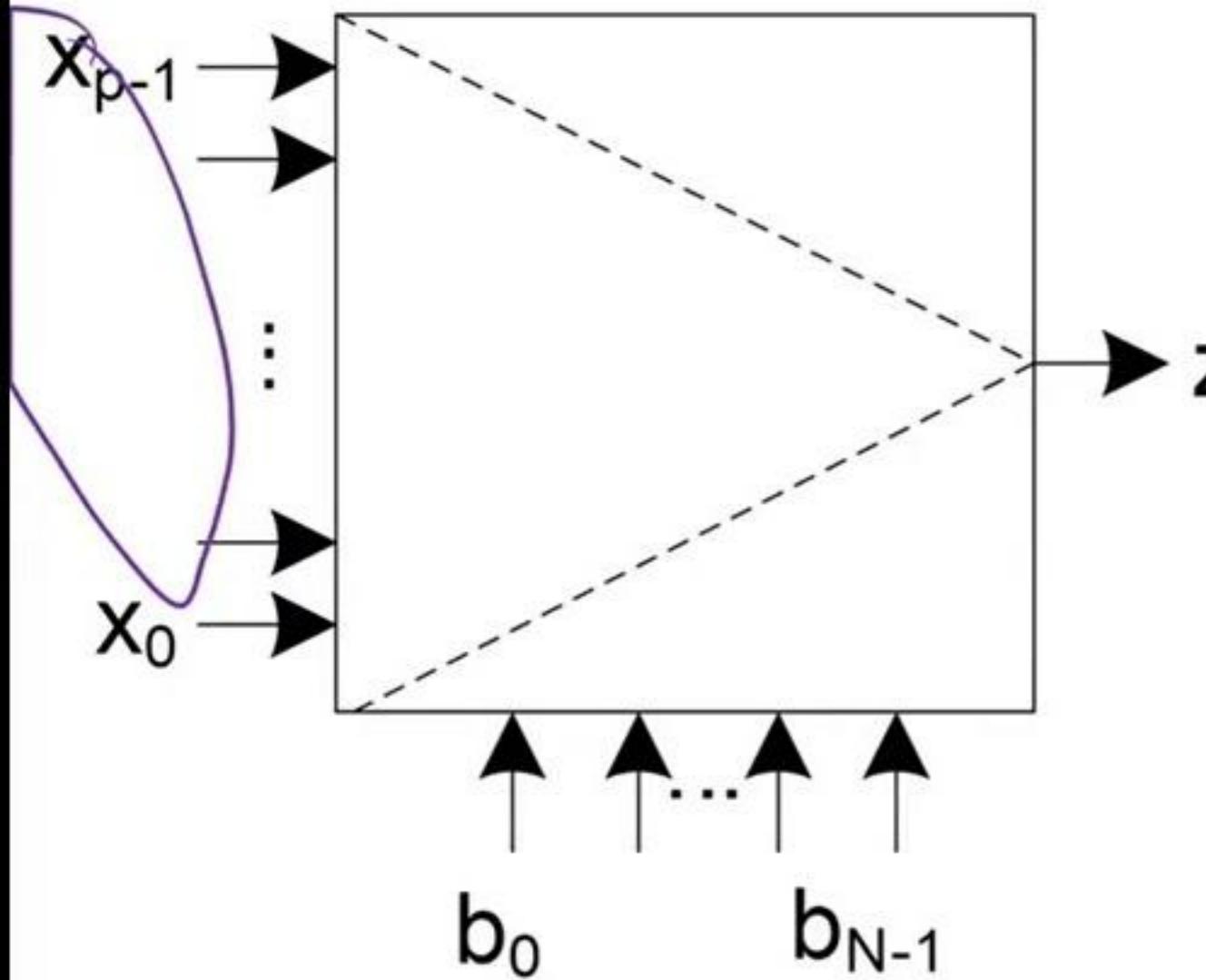
$$x_{p-1} \cdot b_{N-1} \cdot b_{N-2} \cdots b_1 \cdot b_0$$



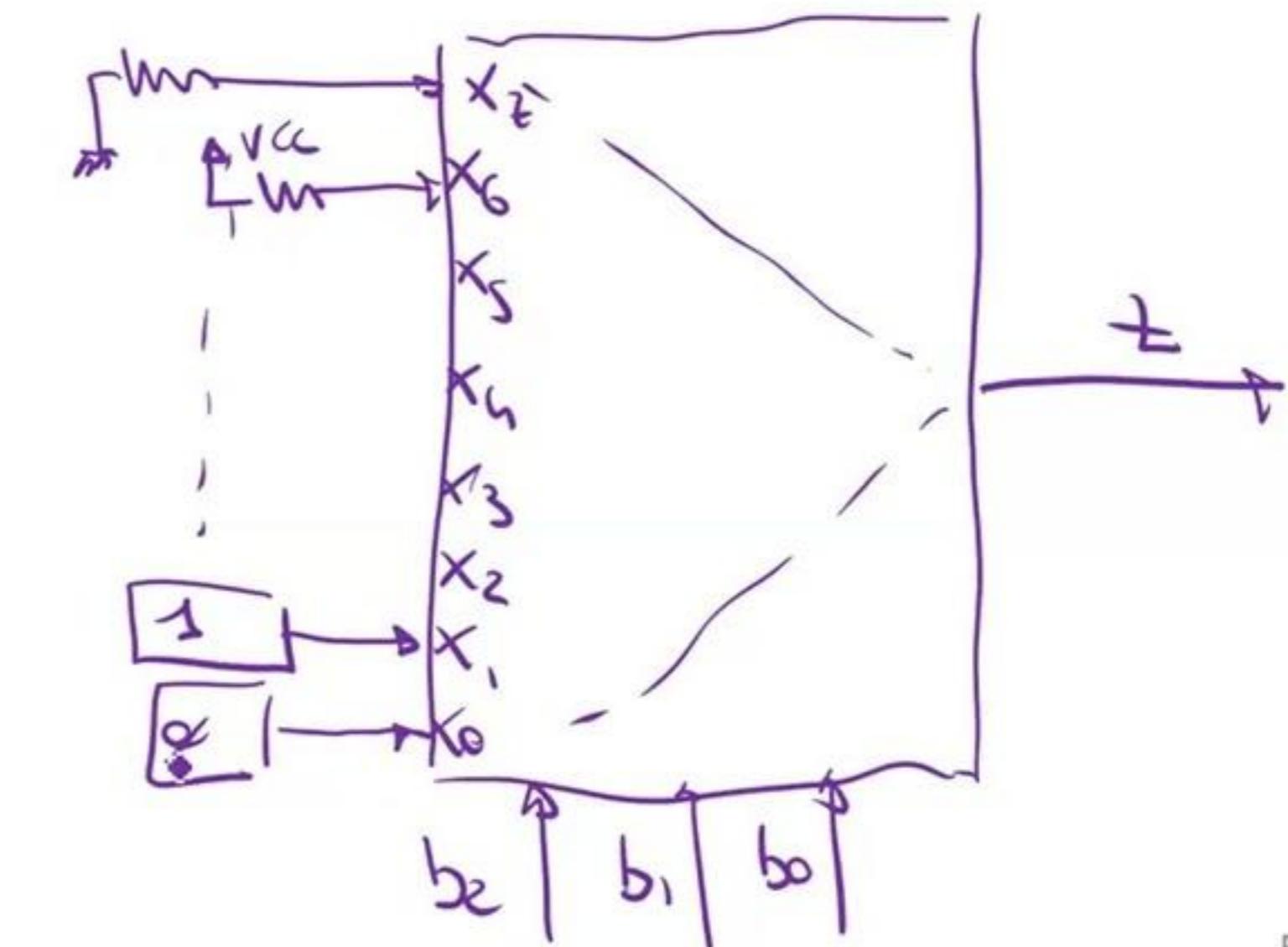
- Rete a **due livelli di logica**
- La **strada più lunga** tra ingresso e uscita passa attraverso **due porte** (AND + OR)
- le NOT sugli ingressi non si contano
  - Il motivo sara' chiaro piu' avanti (registri)

# Multiplexer come rete combinatoria universale

- un multiplexer con  $N$  variabili di comando è in grado di realizzare qualunque legge combinatoria ad  $N$  ingressi ed un'uscita
  - basta connettere ai  $2^N$  ingressi dei generatori di costante



$j$	$b_2$	$b_1$	$b_0$	$z$
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0



# Multiplexer come rete combinatoria universale

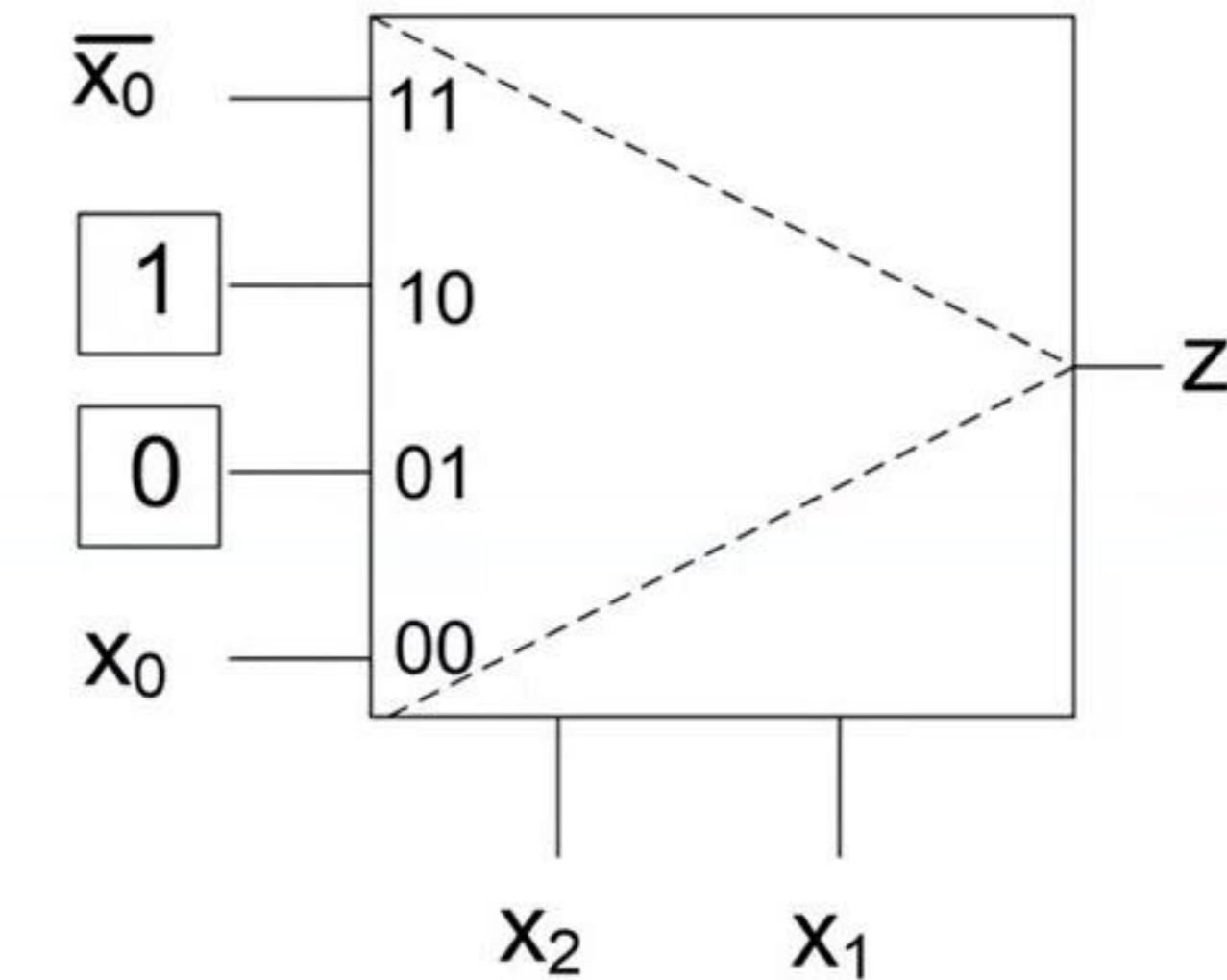
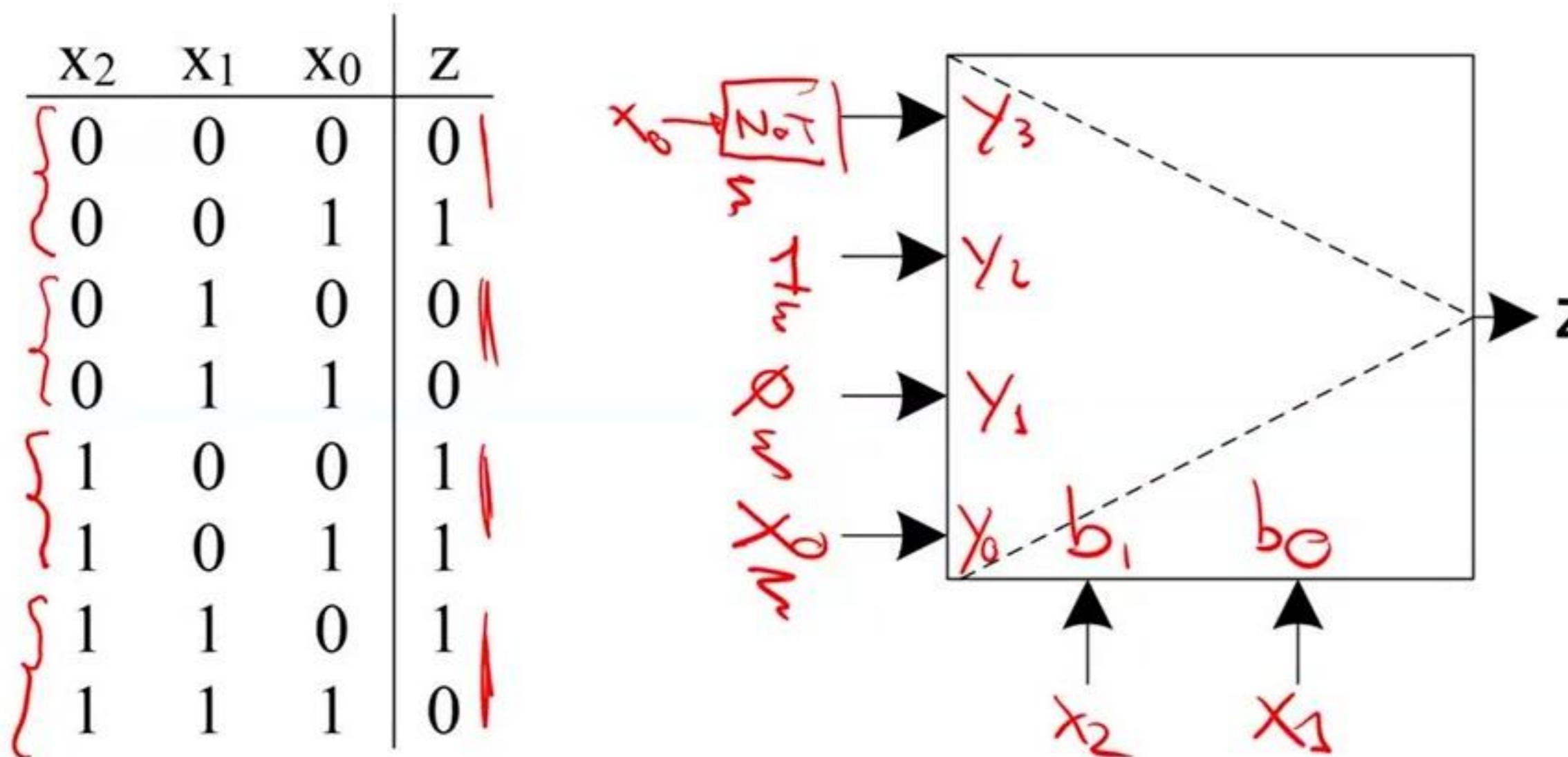
1. Un multiplexer si realizza con porte AND, OR, NOT, ed è una rete a **due livelli di logica**.
  2. Un multiplexer realizza una **qualunque** rete combinatoria ad un'uscita.
  3. una rete combinatoria a più uscite può essere scomposta in reti ad un'uscita messe “in parallelo”.
- Quindi:

**Ogni rete combinatoria può essere costruita combinando AND, OR, NOT  
in al più due livelli di logica**

Mux N

# Esercizio

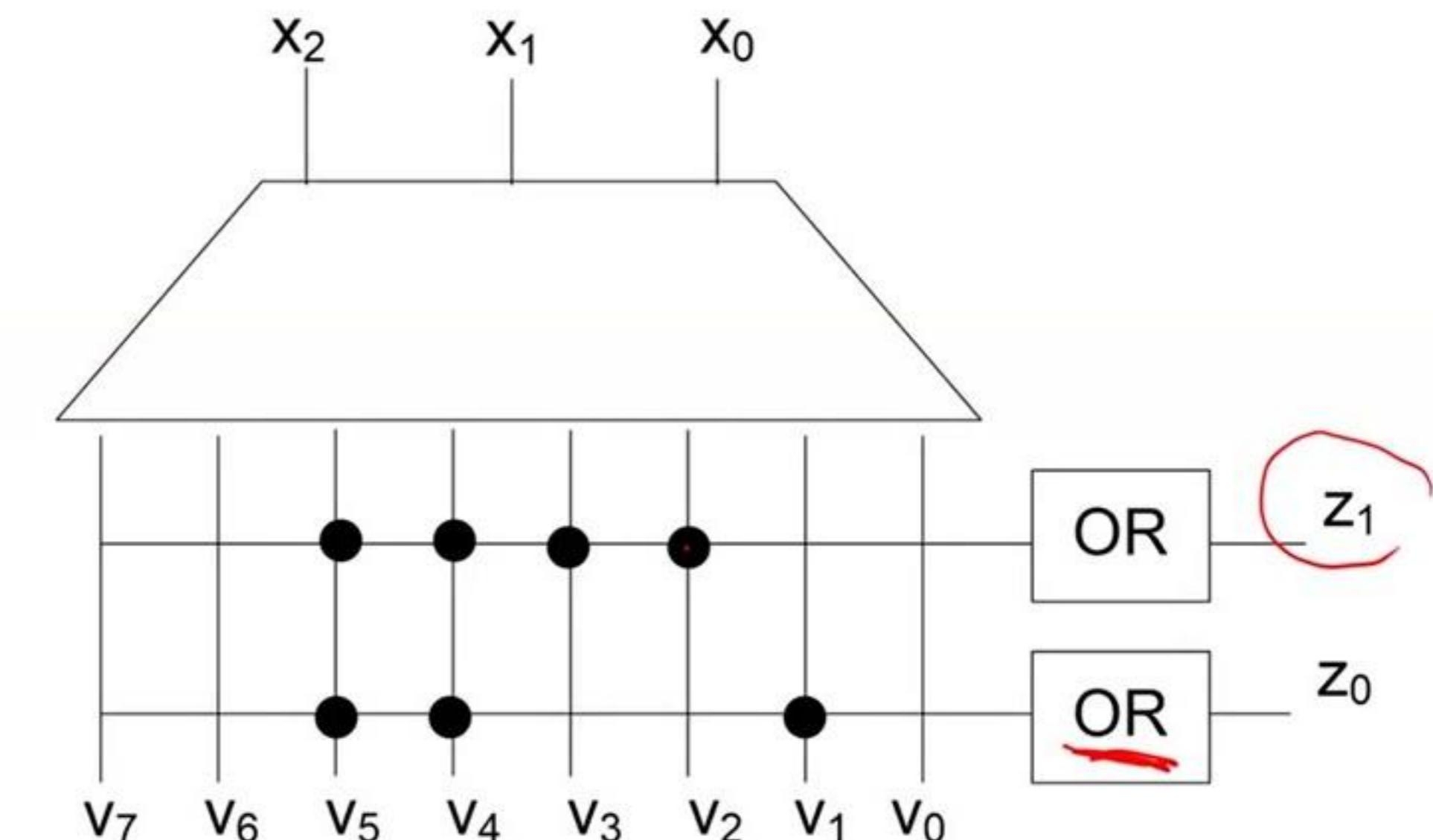
- Data una qualunque tabella di verità per rete ad  $N$  ingressi, è sempre possibile realizzare la rete che la implementa tramite:
  - un multiplexer ad  $N - 1$  variabili di comando
  - al più porte NOT



# Modello strutturale universale per reti comb.

- Vediamo adesso **un modo per sintetizzare** una rete logica ad  $N$  ingressi ed  $M$  uscite a partire da una tabella di verità.

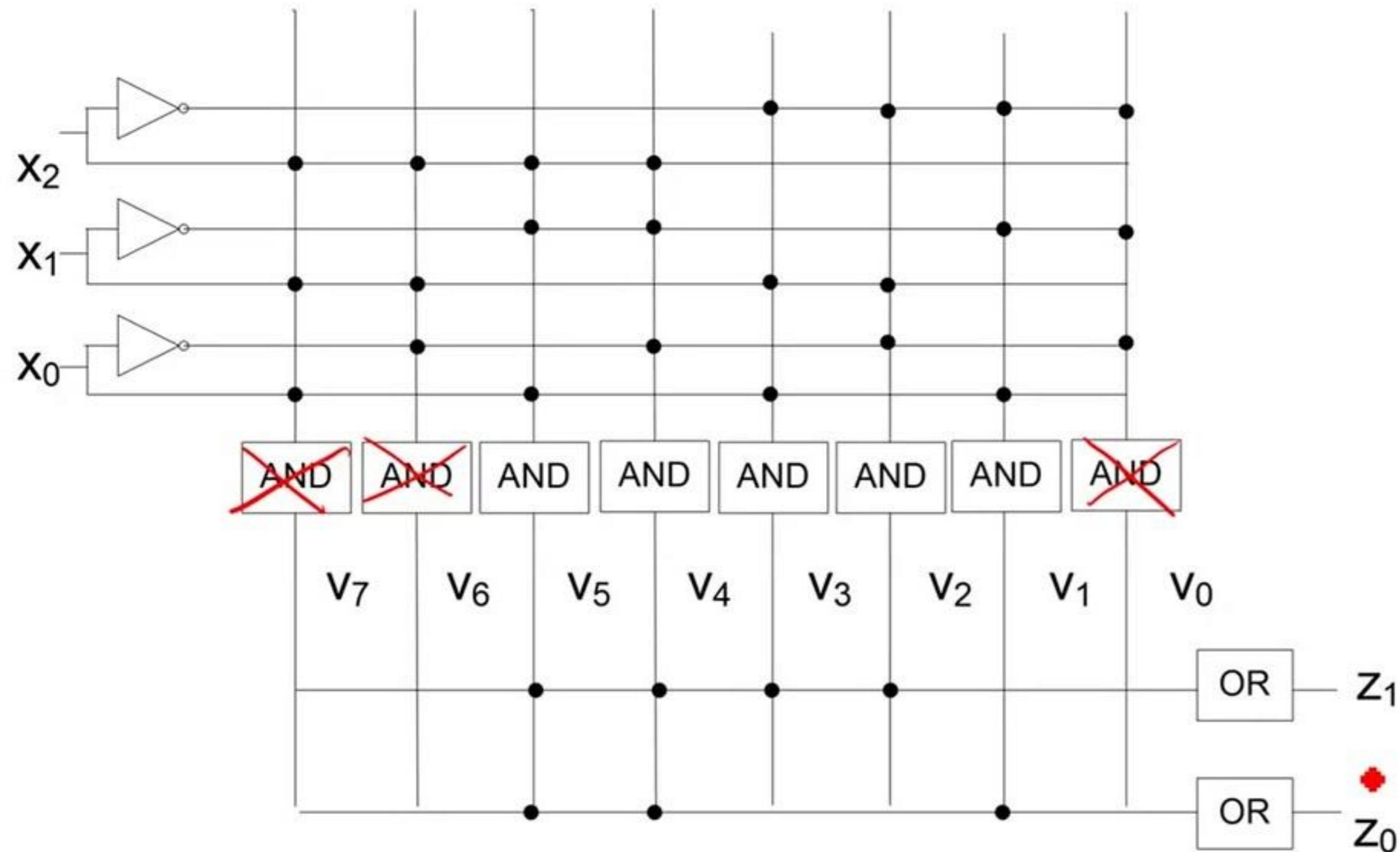
	$x_2$	$x_1$	$x_0$	$z_1$	$z_0$
$v_0$	0	0	0	0	0
$v_1$	0	0	1	0	1
$v_2$	0	1	0	1	0
$v_3$	0	1	1	1	0
$v_4$	1	0	0	1	1
$v_5$	1	0	1	1	1
$v_6$	1	1	0	0	0
$v_7$	1	1	1	0	0



$$\begin{cases} z_1 = v_2 + v_3 + v_4 + v_5 \\ z_0 = v_1 + v_4 + v_5 \end{cases}$$

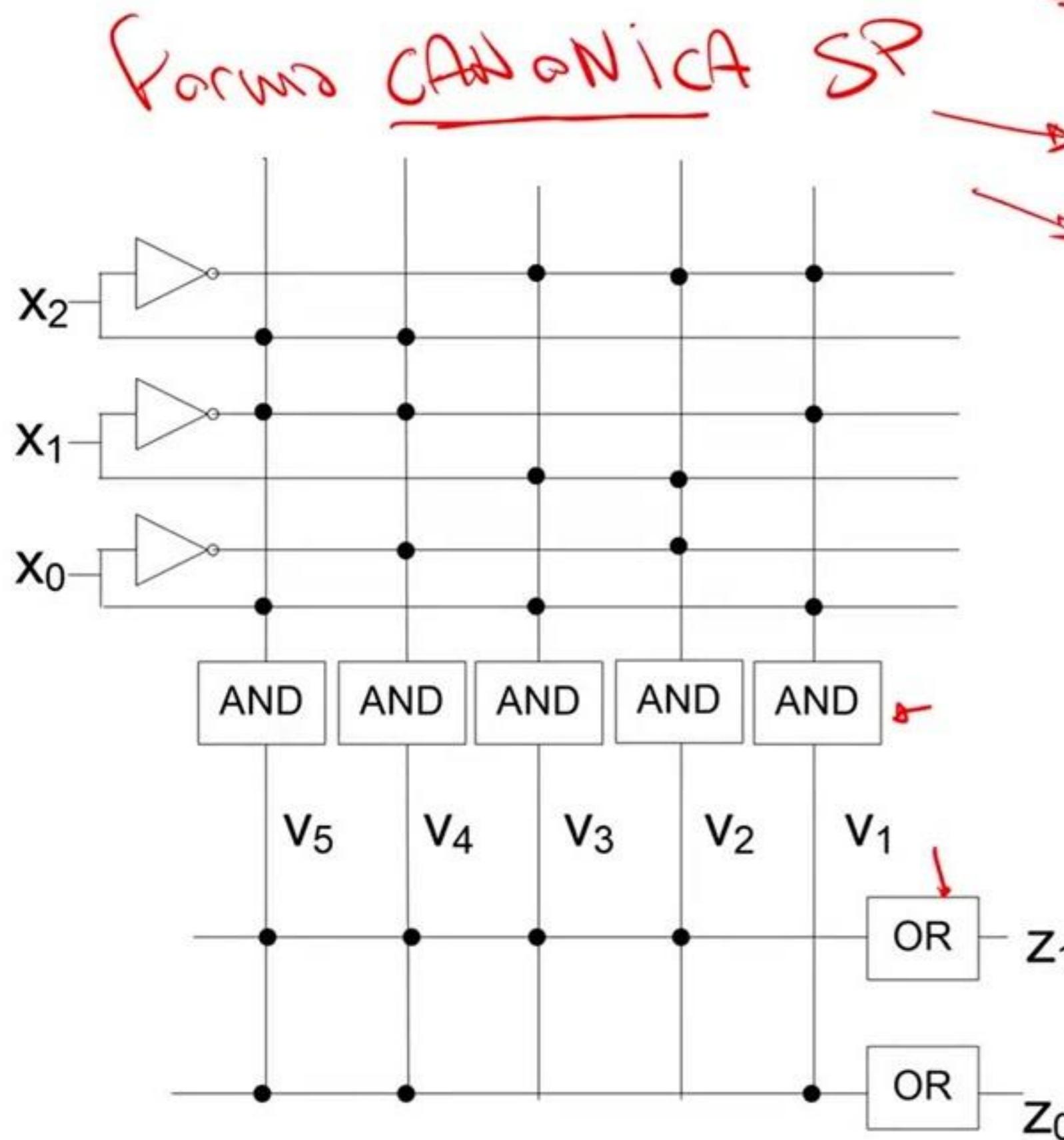
# Ridurre il costo della sintesi

- Partiamo dalla sintesi del MSU e vediamo cosa possiamo togliere



# Ridurre il costo della sintesi (cont.)

- Cerchiamo aiuto nelle proprietà dell'Algebra di Boole



$$\begin{cases} z_1 = \overline{x_2} \cdot x_1 \cdot \overline{x_0} + \overline{x_2} \cdot x_1 \cdot x_0 + x_2 \cdot \overline{x_1} \cdot \overline{x_0} + x_2 \cdot \overline{x_1} \cdot x_0 \\ z_0 = \overline{x_2} \cdot \overline{x_1} \cdot x_0 + x_2 \cdot \overline{x_1} \cdot \overline{x_0} + x_2 \cdot \overline{x_1} \cdot x_0 \end{cases}$$

$$\overline{x_2} \cdot x_1 \cdot (\cancel{x_0 + x_0})$$

$$x_2 \cdot \overline{x_1}$$

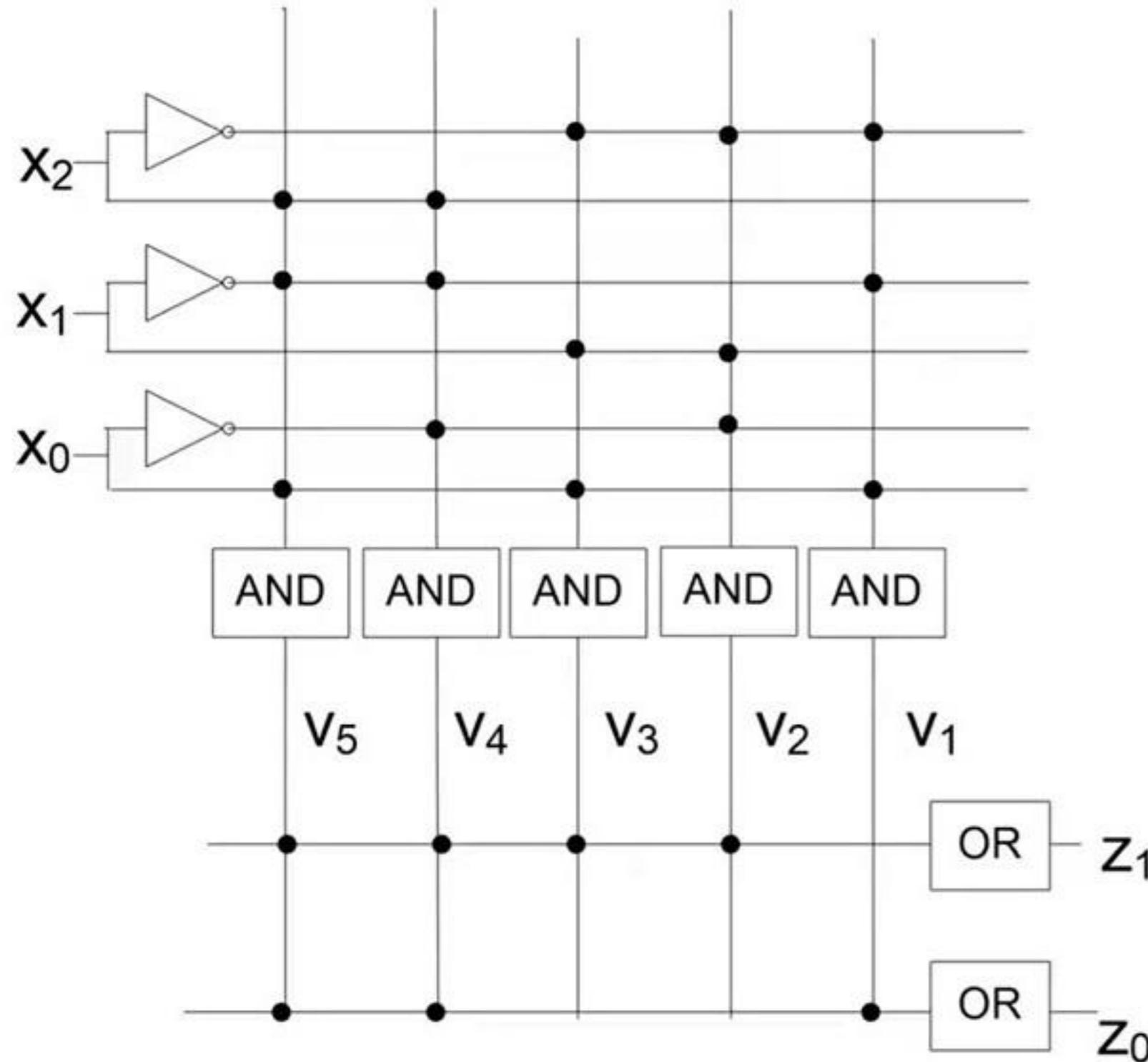
$$z_1 = \underline{\overline{x_2} \cdot x_1 \cdot \overline{x_0}} + \underline{\overline{x_2} \cdot x_1 \cdot x_0} + x_2 \cdot \overline{x_1} \cdot \overline{x_0} + x_2 \cdot \overline{x_1} \cdot x_0$$

$$= \overline{x_2} \cdot x_1 \cdot (\cancel{x_0} + x_0) + x_2 \cdot \overline{x_1} \cdot (\cancel{x_0} + x_0)$$

$$z_1 = \overline{x_2} \cdot x_1 + x_2 \cdot \overline{x_1}$$

# Ridurre il costo della sintesi (cont.)

- Cerchiamo aiuto nelle proprie' dell'Algebra di Boole



$$\begin{cases} Z_1 = \overline{x_2} \cdot x_1 \cdot \overline{x_0} + \overline{x_2} \cdot x_1 \cdot x_0 + x_2 \cdot \overline{x_1} \cdot \overline{x_0} + x_2 \cdot \overline{x_1} \cdot x_0 \\ Z_0 = \overline{x_2} \cdot \overline{x_1} \cdot x_0 + x_2 \cdot \overline{x_1} \cdot \overline{x_0} + x_2 \cdot \overline{x_1} \cdot x_0 \end{cases}$$

$$\begin{aligned} Z_0 &= \cancel{\overline{x_2} \cdot \overline{x_1} \cdot x_0} + \cancel{x_2 \cdot \overline{x_1} \cdot \overline{x_0}} + \cancel{x_2 \cdot \overline{x_1} \cdot x_0} + \cancel{x_2 \cdot \overline{x_1} \cdot x_0} \\ &= \overline{x_1} \cdot x_0 \cdot (\overline{x_2} + x_2) + x_2 \cdot \overline{x_1} \cdot (\overline{x_0} + x_0) \end{aligned}$$

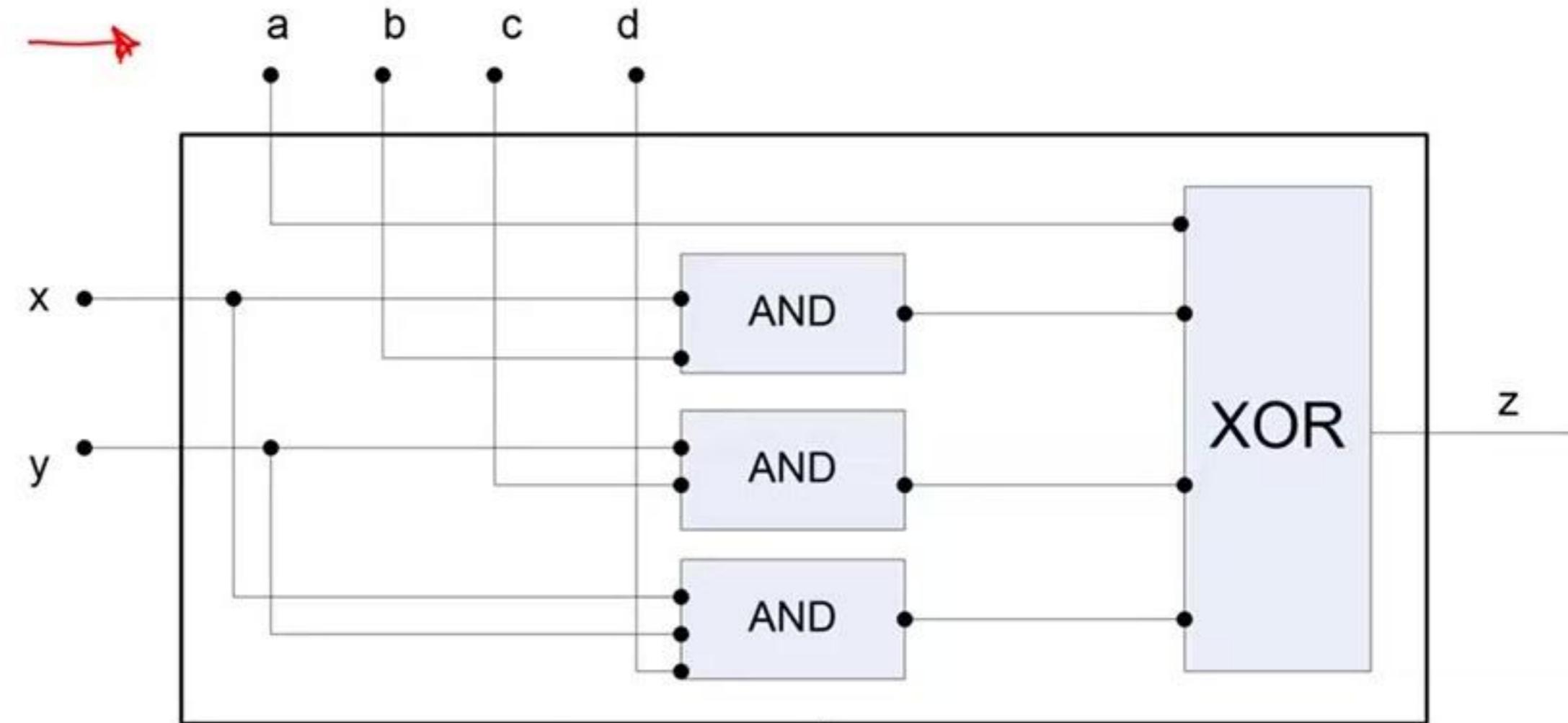
z<sub>0</sub> =  $\overline{x_1} \cdot x_0 + x_2 \cdot \overline{x_1}$

# Ridurre il costo della sintesi (cont.)

- Esistono **diverse sintesi** per una stessa legge combinatoria
- Hanno **costo diverso** in termini di numero di porte
- Riduzione del costo: basata su proprieta' **algebriche**
- Abbiamo bisogno di un metodo **formale** (algoritmico) per minimizzare il costo di una sintesi

# Esercizio

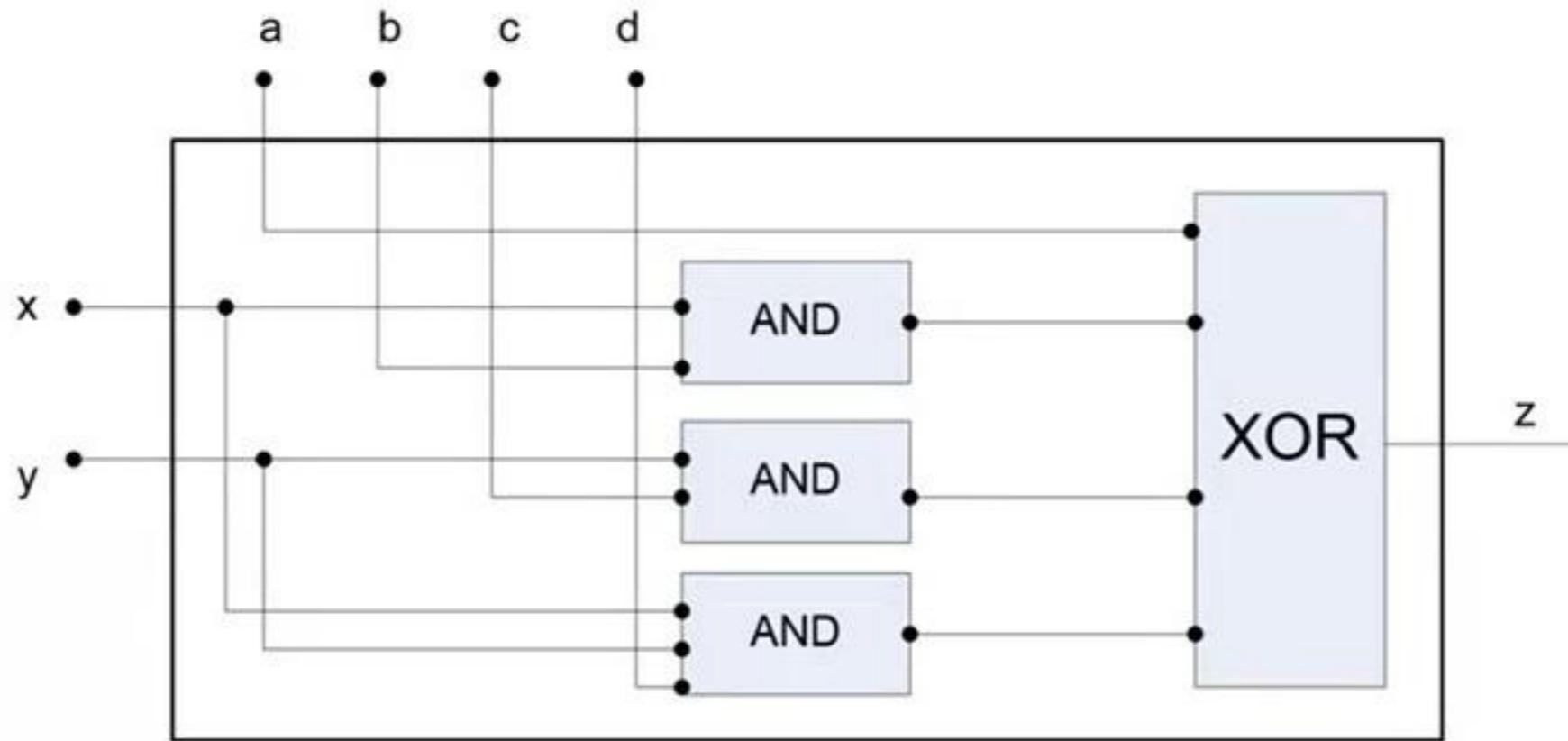
- La rete ha 2 ingressi ( $x, y$ ), un'uscita ( $z$ ), e 4 variabili di comando  $a, b, c, d$ .
- implementa una legge  $f(x, y)$  diversa a seconda del valore delle variabili di comando.



$w = \dots$	$x$	$y$	$z$
	0	0	$f(0,0)$
	0	1	$f(0,1)$
	1	0	$f(1,0)$
	1	1	$f(1,1)$

- Scrivere l'espressione algebrica che lega  $z$  agli ingressi e alle variabili di comando
- Calcolare  $a, b, c, d$  in modo da implementare una generica funzione  $f(x, y)$  nota (assumendo, cioè, di conoscere  $f(0,0), f(1,0), f(0,1), f(1,1)$ ).
- calcolare  $a, b, c, d$  per i *casi particolari* a)  $f(x, y) = \overline{xy}$ , b)  $f(x, y) = x\overline{y}$ ,

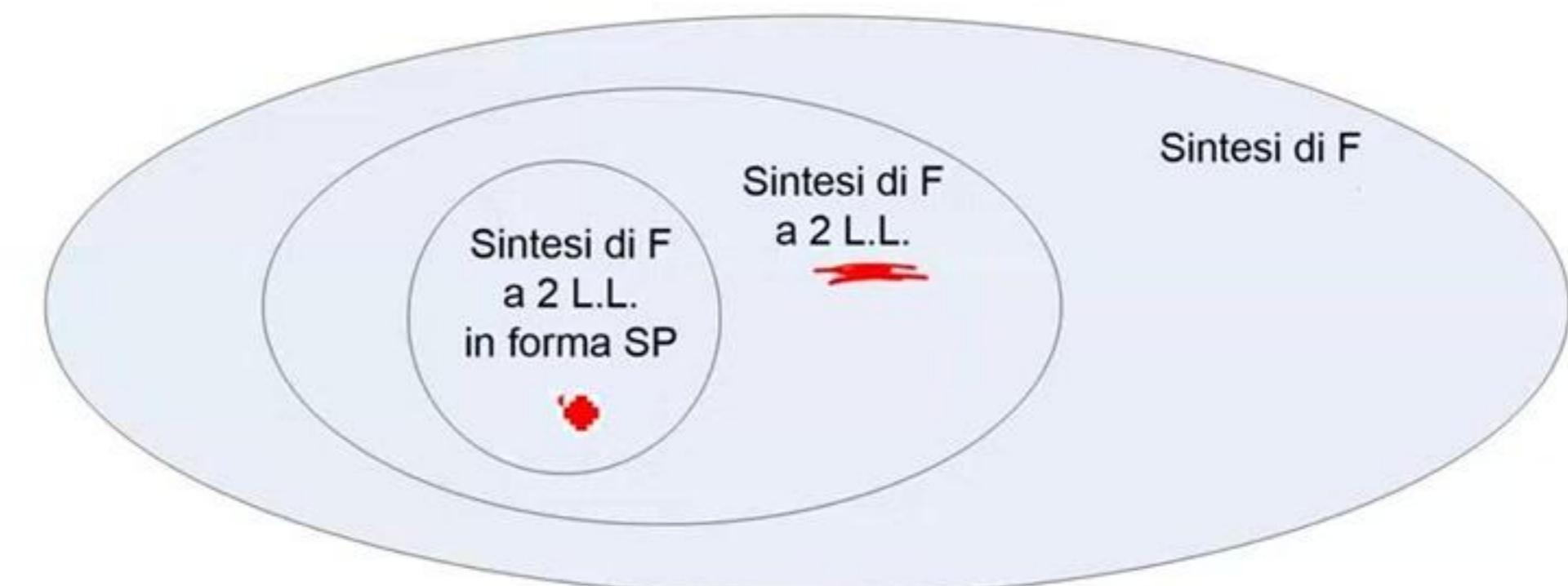
# Esercizio (cont.)



x	y	z
0	0	$f(0,0)$
0	1	$f(0,1)$
1	0	$f(1,0)$
1	1	$f(1,1)$

# Sintesi di reti in forma SP a costo minimo

- Esistono **due criteri di costo**
  - a porte: ogni porta conta per un'unità di costo.
  - a diodi: ogni **ingresso** conta per un'unità di costo.
- Costo(AND a 10 ingressi)=Costo(AND a 2 ingressi) con il primo criterio
- Costo(AND a 10 ingressi)=5 Costo(AND a 2 ingressi) con il secondo
- Metodo che si applica a reti con un'uscita
- Produce **reti in forma SP a 2 livelli di logica**
  - Tra tutte queste, trova quella di costo minimo



# Espansione di Shannon

- è sempre possibile scrivere **qualunque** legge combinatoria  $F$  come **somma di prodotti degli ingressi (diretti o negati)**
- $z = f(x_{N-1}, \dots, x_0)$

$$z = f(0, \dots, 0, 0) \cdot \overline{x_{N-1}} \cdot \overline{x_{N-2}} \cdot \dots \cdot \overline{x_1} \cdot \overline{x_0}$$

$$+ f(0, \dots, 0, 1) \cdot \overline{x_{N-1}} \cdot \overline{x_{N-2}} \cdot \dots \cdot \overline{x_1} \cdot x_0$$

...

$$+ f(1, \dots, 1, 0) \cdot x_{N-1} \cdot x_{N-2} \cdot \dots \cdot x_1 \cdot \overline{x_0}$$

$$+ f(1, \dots, 1, 1) \cdot x_{N-1} \cdot x_{N-2} \cdot \dots \cdot x_1 \cdot x_0$$

- $f(x_{N-1}, \dots, x_0) = 0,$ 
  - $0 \cdot \alpha = 0$ , la riga vale 0.
  - $0 + \alpha = \alpha$ , posso togliere la riga

- $f(x_{N-1}, \dots, x_0) = 1,$ 
  - $1 \cdot \alpha = \alpha$ , posso togliere  $f(\dots)$

# Forma canonica SP

- **SP:** z è somma di prodotti
- **Canonica:** ogni prodotto ha come fattori **tutti gli ingressi** diretti o negati
- ciascuno dei termini della somma si chiama **mintermine**,
- corrisponde ad uno stato di ingresso riconosciuto dalla rete.

$x_3$	$x_2$	$x_1$	$x_0$	$z_0$
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

$$\begin{array}{ll} 1 \cdot \alpha = \alpha & 0 + \beta = \beta \\ 0 \cdot \alpha = 0 & \end{array}$$



Espansione di Shannon

$$\begin{aligned} z = & 0 \cdot \overline{x_3} \cdot \overline{x_2} \cdot \overline{x_1} \cdot \overline{x_0} \\ & \cancel{+ 1 \cdot x_3 \cdot x_2 \cdot x_1 \cdot x_0} \\ & \dots \\ & + 0 \cdot x_3 \cdot x_2 \cdot x_1 \cdot \overline{x_0} \\ & \cancel{+ 0 \cdot x_3 \cdot x_2 \cdot \overline{x_1} \cdot x_0} \end{aligned}$$



Forma canonica SP  
(lista dei mintermini)

$$\begin{aligned} z = & \overline{x_3} \cdot \overline{x_2} \cdot x_1 \cdot x_0 \\ & + \overline{x_3} \cdot x_2 \cdot x_1 \cdot x_0 \\ & + \overline{x_3} \cdot x_2 \cdot x_1 \cdot \overline{x_0} \\ & + x_3 \cdot x_2 \cdot x_1 \cdot x_0 \\ & + x_3 \cdot x_2 \cdot x_1 \cdot \overline{x_0} \\ & + x_3 \cdot \overline{x_2} \cdot x_1 \cdot x_0 \\ & + x_3 \cdot \overline{x_2} \cdot x_1 \cdot \overline{x_0} \\ & + \overline{x_3} \cdot x_2 \cdot \overline{x_1} \cdot x_0 \\ & + \overline{x_3} \cdot x_2 \cdot \overline{x_1} \cdot \overline{x_0} \\ & + x_3 \cdot x_2 \cdot \overline{x_1} \cdot x_0 \\ & + x_3 \cdot x_2 \cdot \overline{x_1} \cdot \overline{x_0} \end{aligned}$$

# Metodo di Quine-McCluskey

- Modo piu' efficiente di cercare fusioni tra mintermini/implicanti
- Basta ordinarli per **numero crescente di 1** (var. dirette)
  - Fusione possibile solo tra partizioni adiacenti
  - Numero di confronti ridotto
- Efficienza dell'algoritmo dipende dalla facilita' di trovare **stati di ingresso adiacenti** riconosciuti dalla legge

$\times_5$

$\overline{x_5}$

Grafic

Mappe di Karnaugh

# Semplificare la forma canonica SP (cont.)

$$\overline{x_3} \cdot \overline{x_2} \cdot \overline{x_1} \cdot x_0$$

$$\overline{x_3} \cdot x_2 \cdot \overline{x_1} \cdot x_0$$

$$\overline{x_3} \cdot x_2 \cdot x_1 \cdot x_0$$

$$\overline{x_3} \cdot x_2 \cdot x_1 \cdot \overline{x_0}$$

$$\overline{x_3} \cdot x_2 \cdot x_1 \cdot x_0$$

$$x_3 \cdot \overline{x_2} \cdot \overline{x_1} \cdot x_0$$

$$x_3 \cdot x_2 \cdot \overline{x_1} \cdot x_0$$

$$x_3 \cdot x_2 \cdot x_1 \cdot \overline{x_0}$$

$$\overline{x_3} \cdot \overline{x_2} \cdot x_0$$

$$\overline{x_2} \cdot \overline{x_1} \cdot x_0$$

$$\overline{x_3} \cdot \overline{x_2} \cdot x_1$$

$$\overline{x_3} \cdot x_1 \cdot \overline{x_0}$$

$$\overline{x_2} \cdot x_1 \cdot \overline{x_0}$$

$$\overline{x_3} \cdot x_1 \cdot x_0$$

$$\overline{x_3} \cdot x_2 \cdot x_1$$

$$x_3 \cdot \overline{x_2} \cdot \overline{x_1}$$

$$x_3 \cdot \overline{x_2} \cdot \overline{x_0}$$

$$\overline{x_3} \cdot x_1$$

$$\cancel{\overline{x_3} \cdot x_1}$$

## Semplificare la forma canonica SP (cont.)

- Applicare esaustivamente la legge

$$\alpha(\beta\gamma) \cancel{=} \alpha\beta\gamma$$

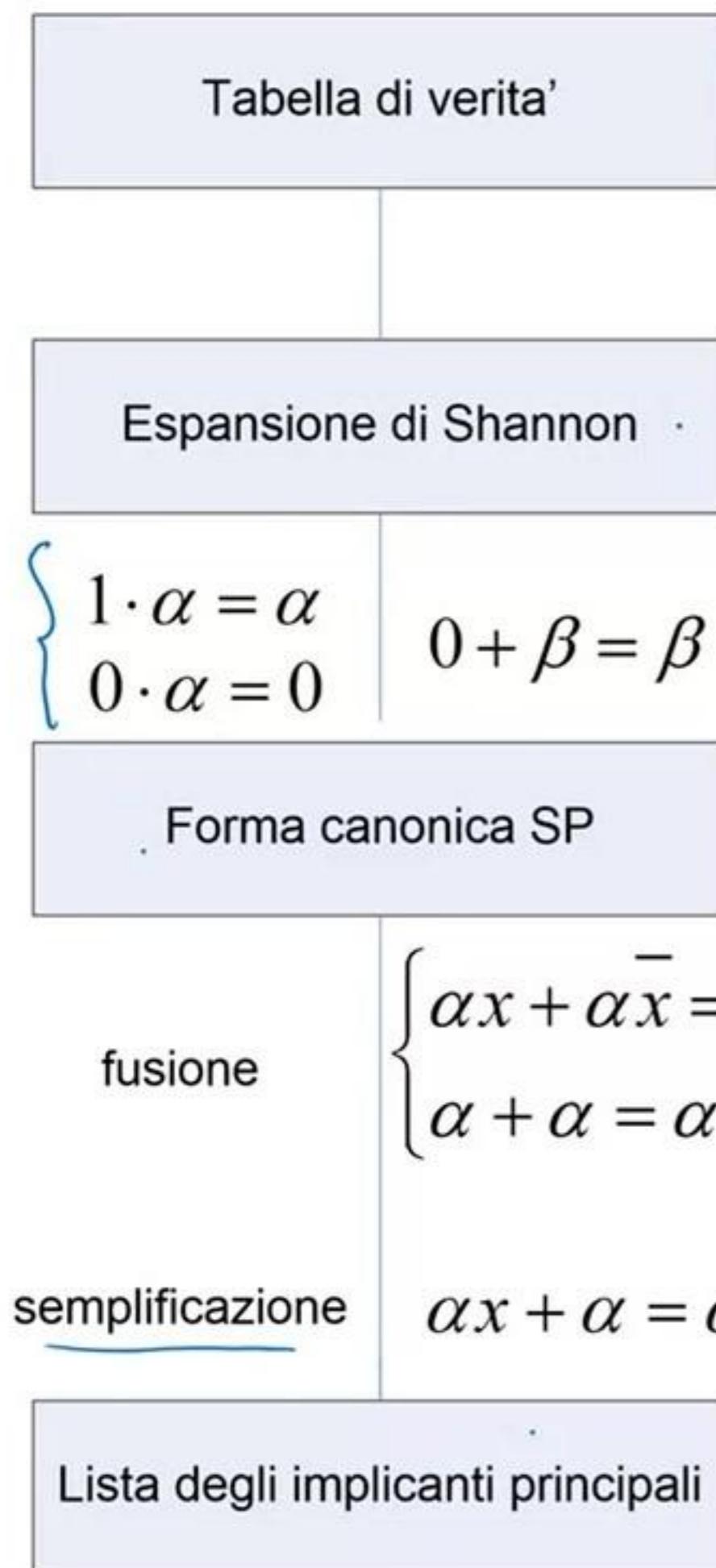
$$\alpha x + \alpha = \alpha$$

- Togliere tutti gli implicanti che hanno prodotto qualcosa per fusione.
- Si ottiene la lista degli implicanti principali

$$z = \overline{x_3} \cdot x_1 + \overline{x_3} \cdot \overline{x_2} \cdot x_0 + \overline{x_2} \cdot \overline{x_1} \cdot x_0 + \overline{x_2} \cdot x_1 \cdot \overline{x_0} + x_3 \cdot \overline{x_2} \cdot \overline{x_1} + x_3 \cdot \overline{x_2} \cdot \overline{x_0}$$

4                    2

# Riepilogo



- La lista degli implicanti principali **costa meno** della forma canonica SP.

- Contiene meno termini
- Ciascuno con meno ingressi.

- E' sintesi di costo minimo?

- **Non è detto.** Potrebbe ancora essere ridondante.

# Liste di copertura (non) ridondanti

- **Lista di copertura**: lista di implicanti, la cui somma è una forma SP per la funzione  $f$ .
  - lista dei mintermini
  - lista degli implicanti (cioè quella ottenuta prima della semplificazione)
  - lista degli implicanti principali
- Lista di copertura **non ridondante**: se tolgo un elemento smette di essere una lista di copertura.
- La lista dei mintermini è una lista **non ridondante**.
  - Infatti, se ne tolgo uno, uno stato di ingresso riconosciuto non sarà coperto (quindi non avrò più una lista di copertura)
- La lista degli implicanti principali può essere **ridondante**. Questa lo è.

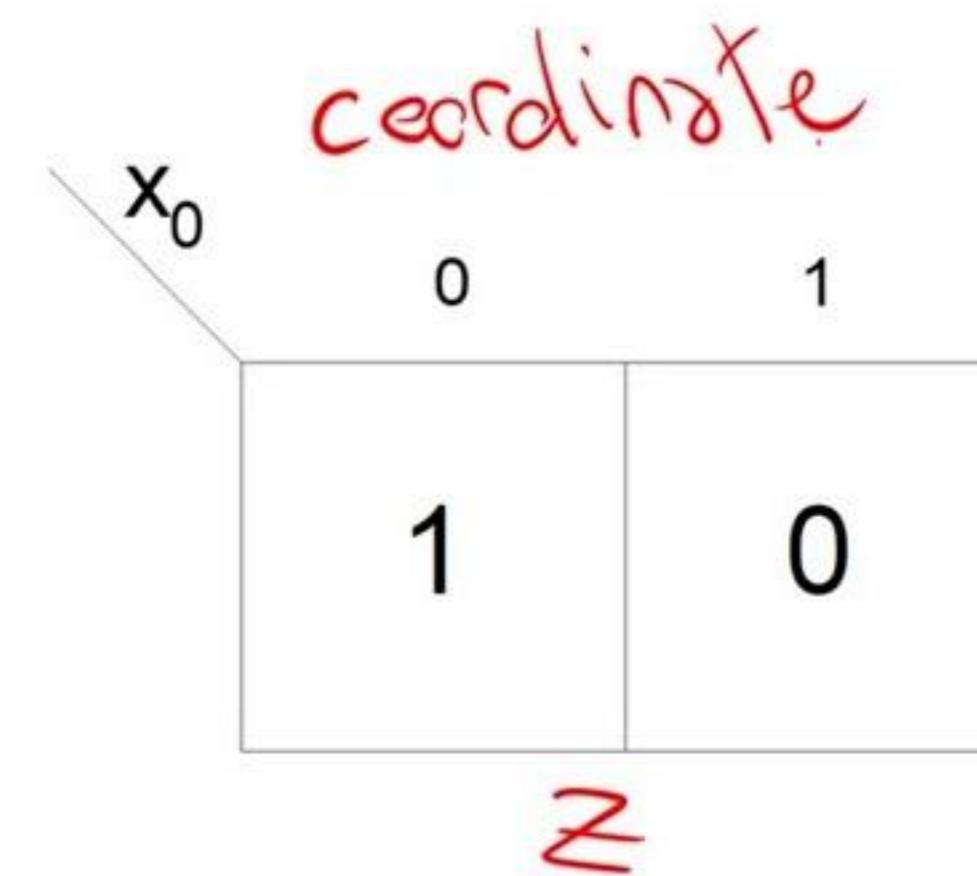
$$z = \overline{x_3} \cdot x_1 + \overline{x_3} \cdot \overline{x_2} \cdot x_0 + \overline{x_2} \cdot \overline{x_1} \cdot x_0 + \overline{x_2} \cdot x_1 \cdot \overline{x_0} + x_3 \cdot \overline{x_2} \cdot \overline{x_1} + x_3 \cdot \overline{x_2} \cdot \overline{x_0}$$

# Metodo di Quine-McCluskey

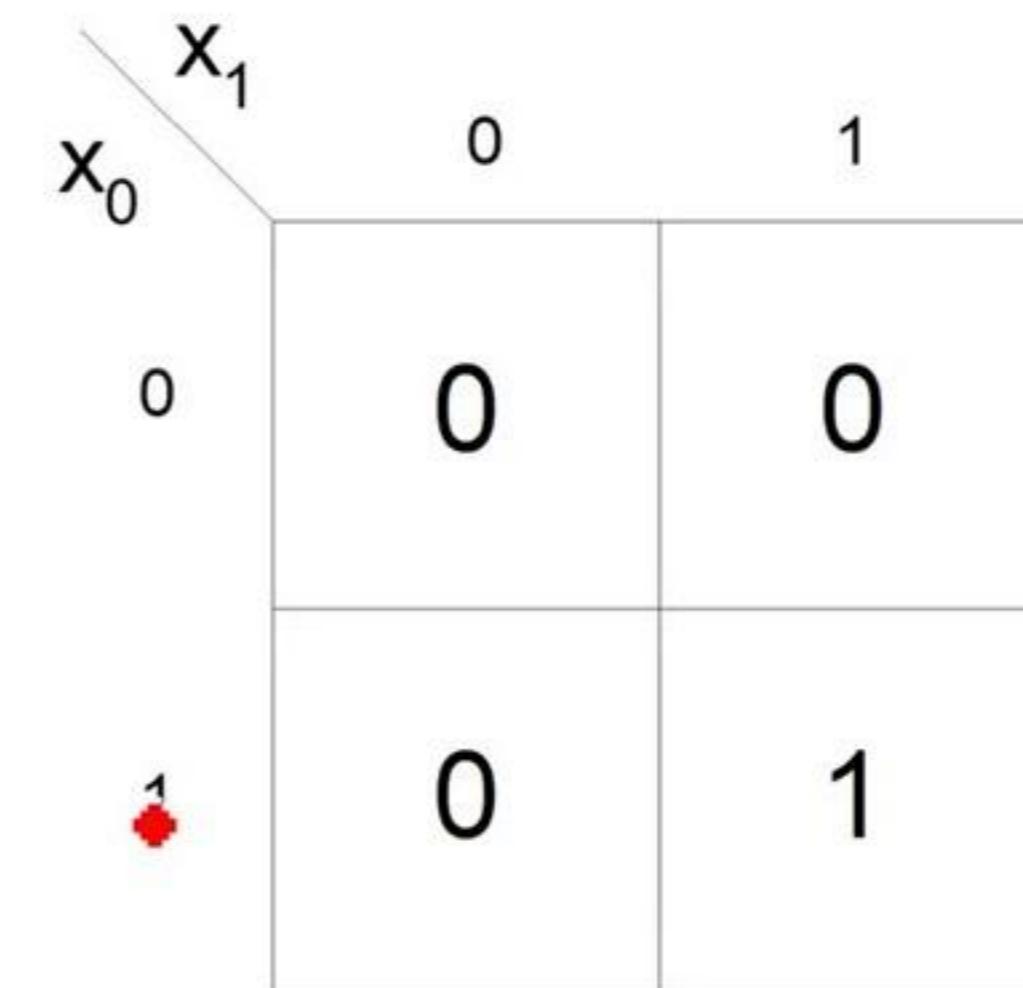
- Modo piu' efficiente di cercare fusioni tra mintermini/implicanti
- Basta ordinarli per **numero crescente di 1** (var. dirette)
  - Fusione possibile solo tra partizioni adiacenti
  - Numero di confronti ridotto
- Efficienza dell'algoritmo dipende dalla facilita' di trovare **stati di ingresso adiacenti** riconosciuti dalla legge

# Mappe di Karnaugh

- Per una rete ad  $N$  ingressi è una matrice di  $2^N$  celle



$$N = 1$$

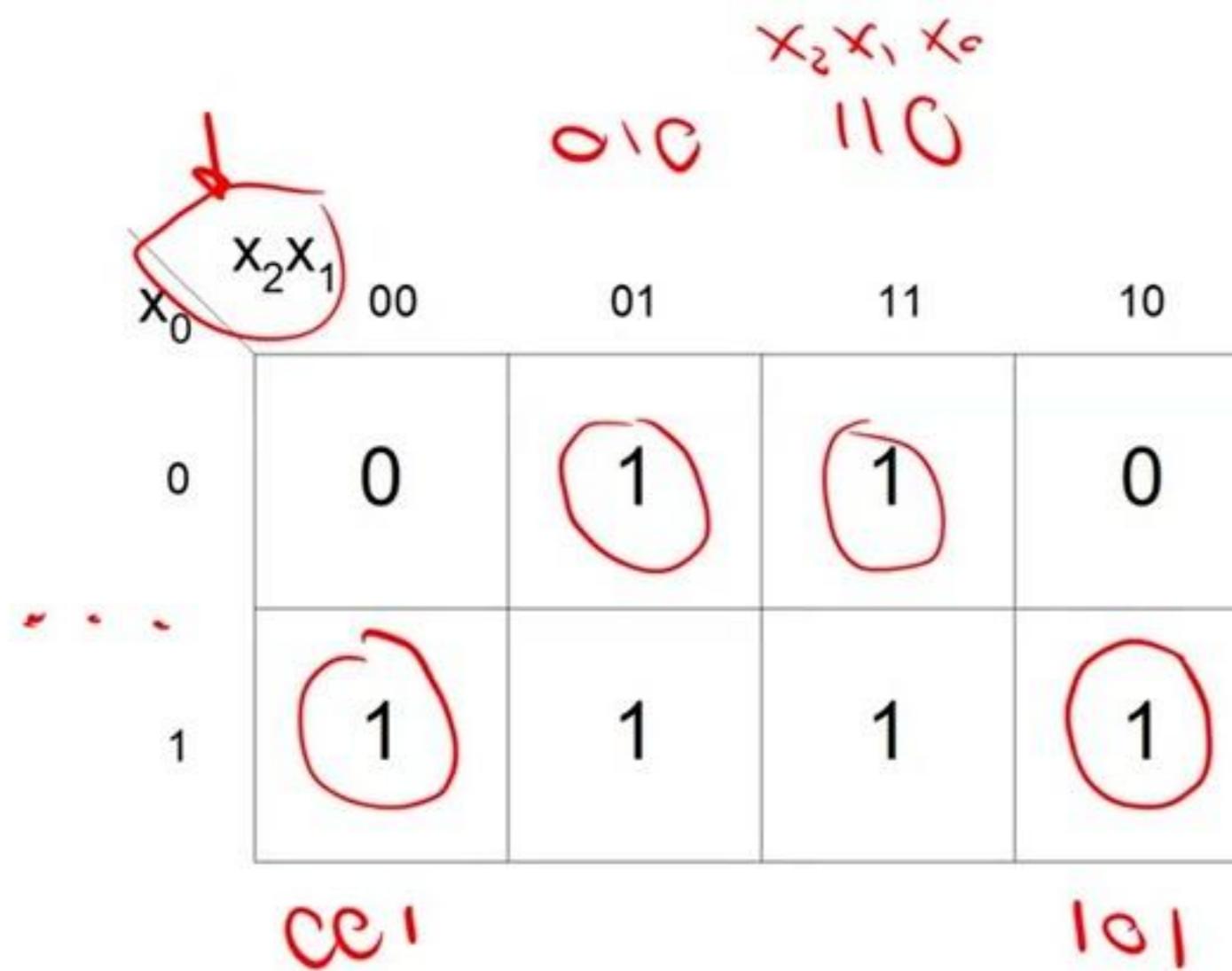


*AND*

$$N = 2$$

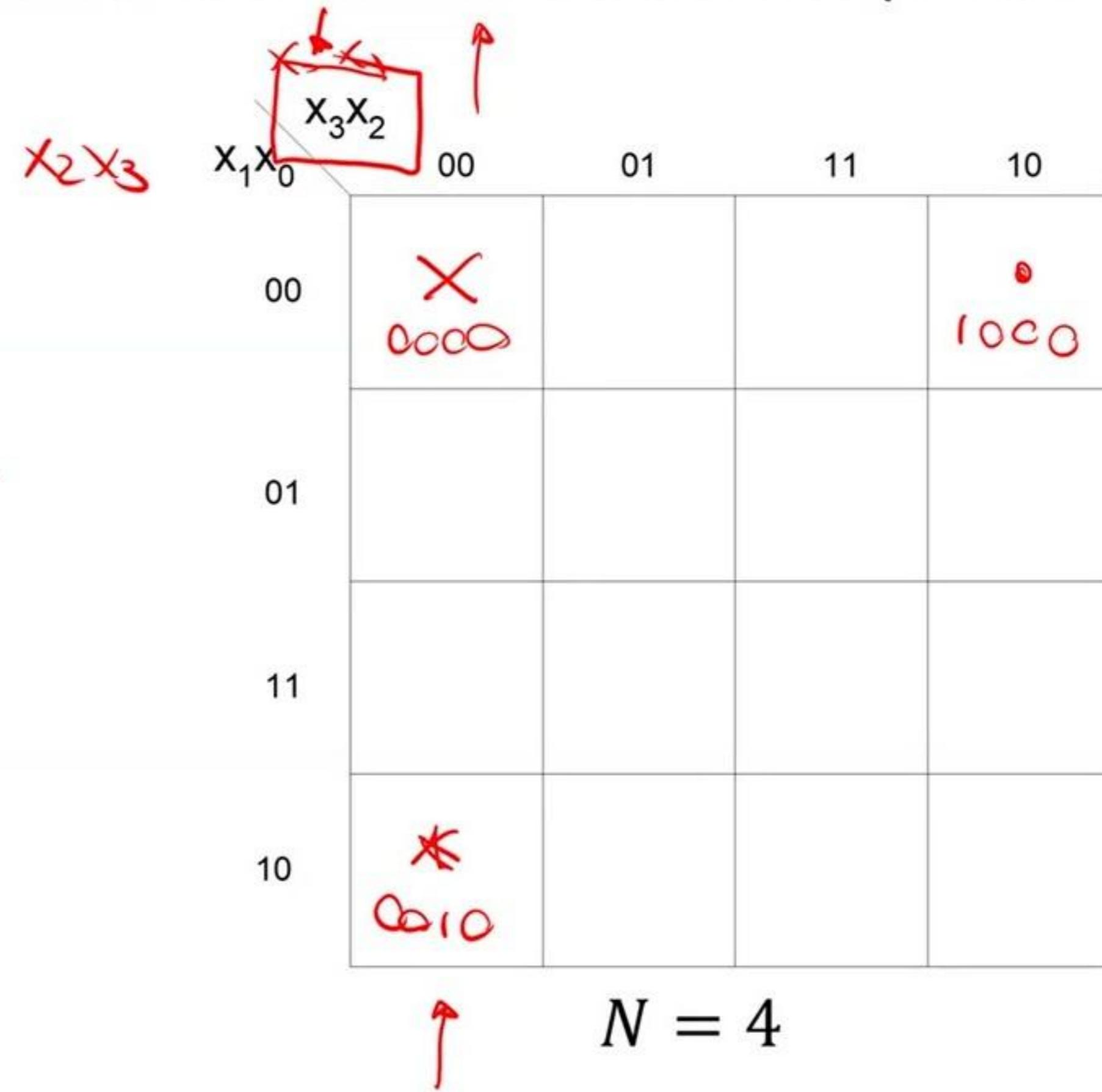
# Mappe di Karnaugh (cont.)

- Celle **contigue** sulla mappa hanno coordinate **adiacenti** (e viceversa)



$$N = 3$$

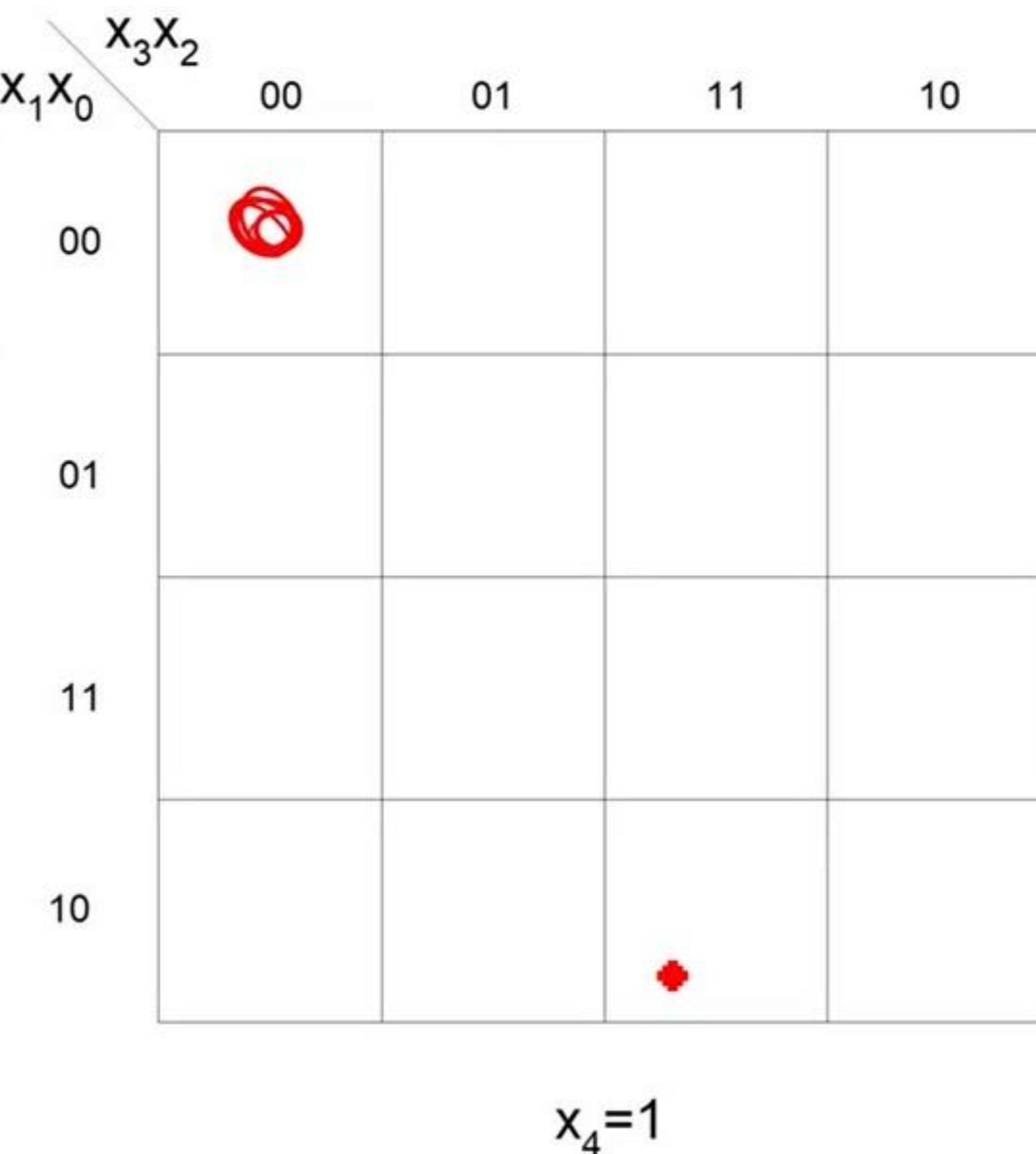
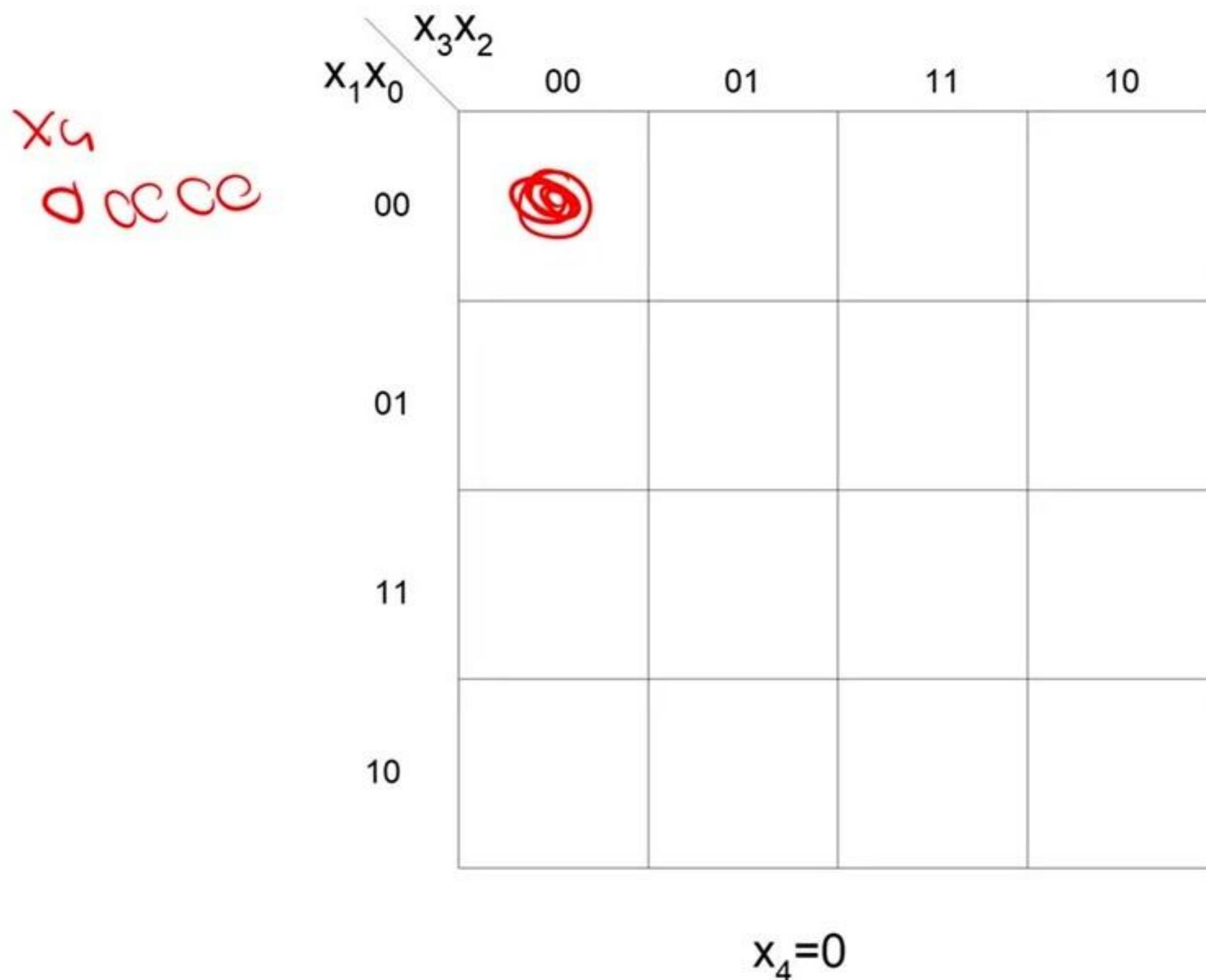
B. ...  
3



$$N = 4$$

# Mappe di Karnaugh (cont.)

- Oltre le 4 coordinate serve la 3a dimensione



$$N = 5$$

# Sottocubi di ordine 1

$x_2 x_1 x_0$

A 011  
B 111

- **Sottocubo di ordine 1**

- casella che contiene un 1, corrispondente quindi ad uno stato di ingresso riconosciuto dalla rete

- **Coordinate di un sottocubo di ordine 1**

- stato di ingresso corrispondente al sottocubo

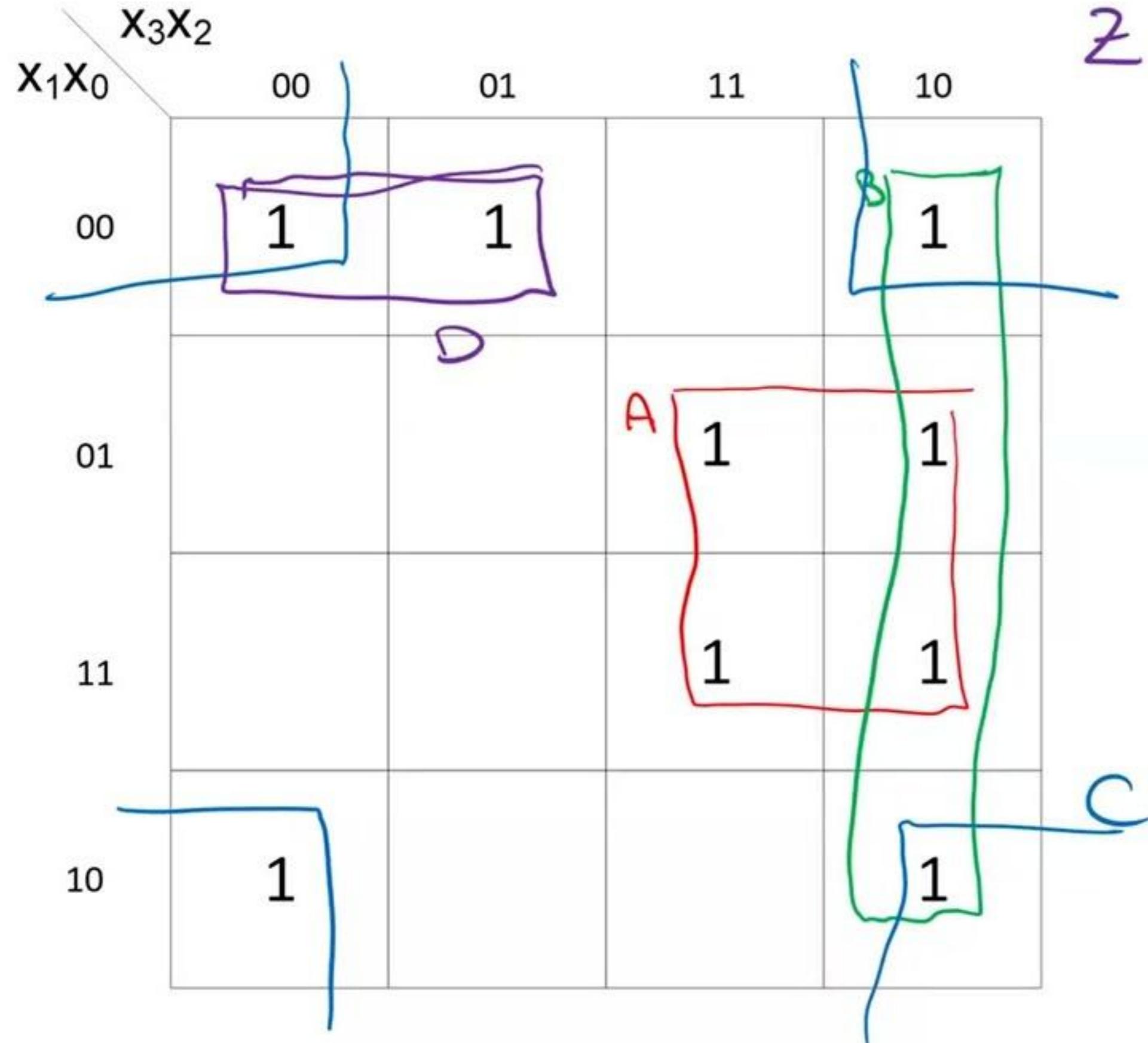
- **Adiacenza tra sottocubi di ordine 1**

- due SO1 sono adiacenti se differiscono tra loro per una sola coordinata

$x_0 \backslash x_2 x_1$	00	01	11	10
0	0	1	1	0
1	1	1	1	1

# Esempio

$P = \cancel{X}_2$



## Sottocubi principali

$$Z = X_3 \cdot X_0 + X_3 \cdot \overline{X}_2 + \overline{X}_2 \cdot \overline{X}_0 + \overline{X}_3 \cdot \overline{X}_1 \cdot \overline{X}_0$$

	X <sub>3</sub>	X <sub>2</sub>	X <sub>1</sub>	X <sub>0</sub>
A	1	-	-	1
B	1	0	-	-
C	-	0	-	0
D	0	-	0	0

$$X_3 \cdot X_0$$

# Sottocubi di ordine 2

$\overline{x_2 \cdot x_1}$

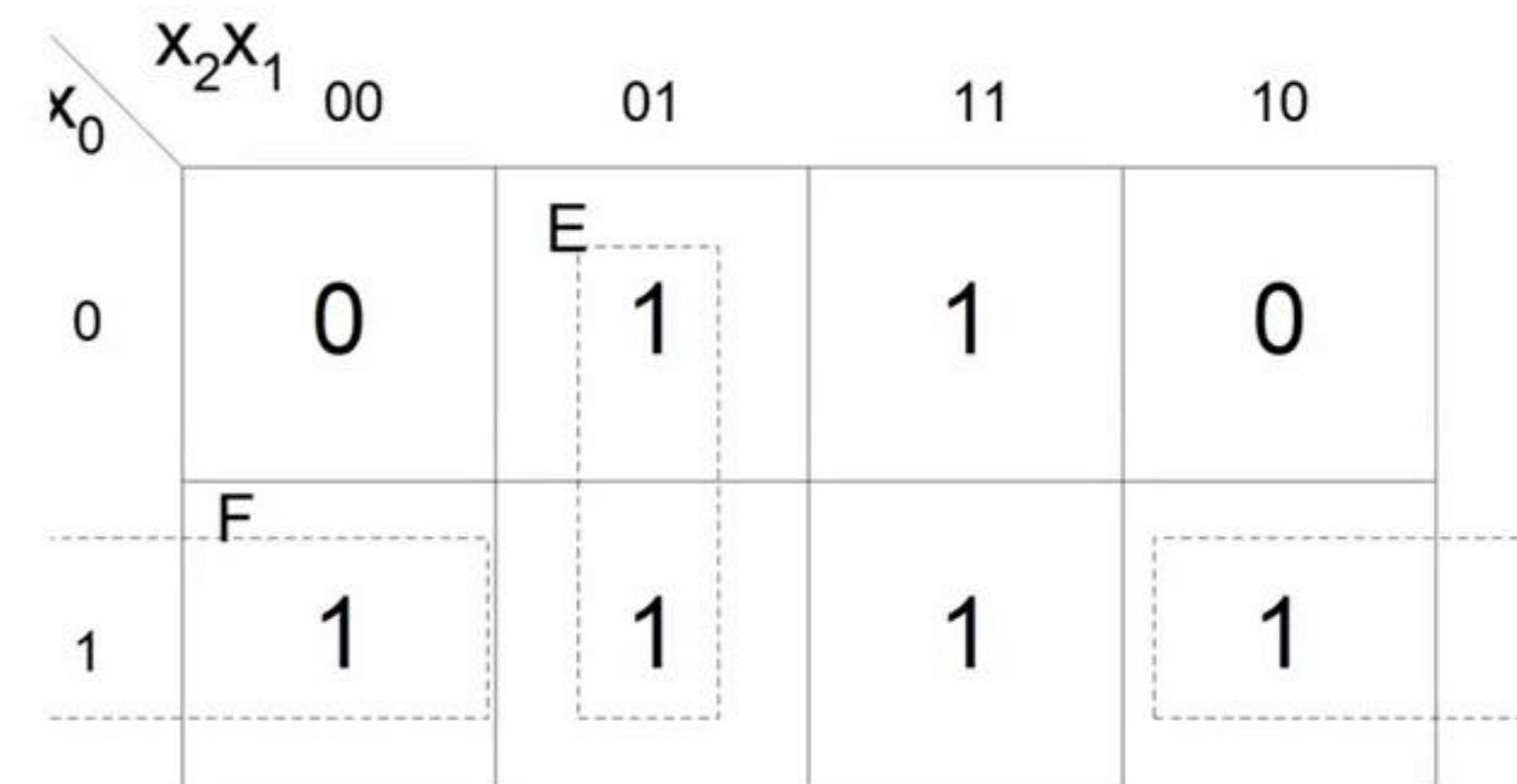
- **Sottocubo di ordine 2**

- costituito da **sottocubi adiacenti di ordine 1**.
- C è un SO2.
- Le sue coordinate sono:  $x_2 = -x_1 = 1$   $x_0 = 1$

- Il SO2 C **copre** i SO1 A e B

- **Adiacenza tra sottocubi di ordine 2**

- due SO2 sono adiacenti se differiscono tra loro per una sola coordinata



	X2	X1	X0
E	0	1	-
F	-	0	1

# Sottocubi di ordine $n$ e liste di copertura

- Sottocubi di ordine 4, 8, ...

~~Diagram of a 2D grid with red highlights.~~

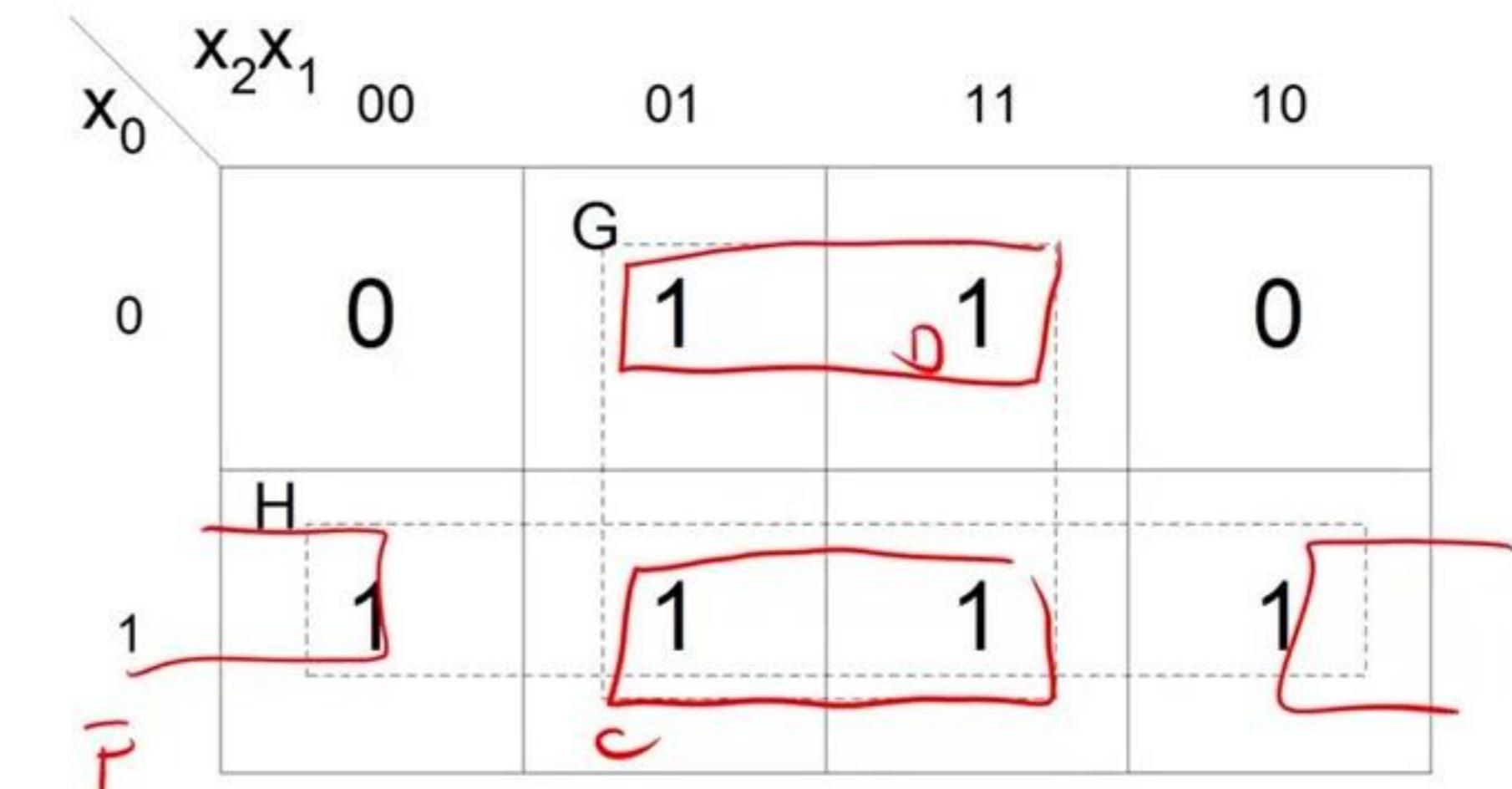
~~Diagram of a 2D grid with red highlights.~~

-

~~Diagram of a 2D grid with red highlights.~~

$$\begin{array}{c} x_2 \quad x_1 \quad x_0 \\ \hline C & - & 1 & 1 \\ D & - & 1 & 0 \\ F & - & 0 & 1 \end{array}$$

SR  
 $\{G, H\}$     $\{\bar{D}, \bar{F}\}$



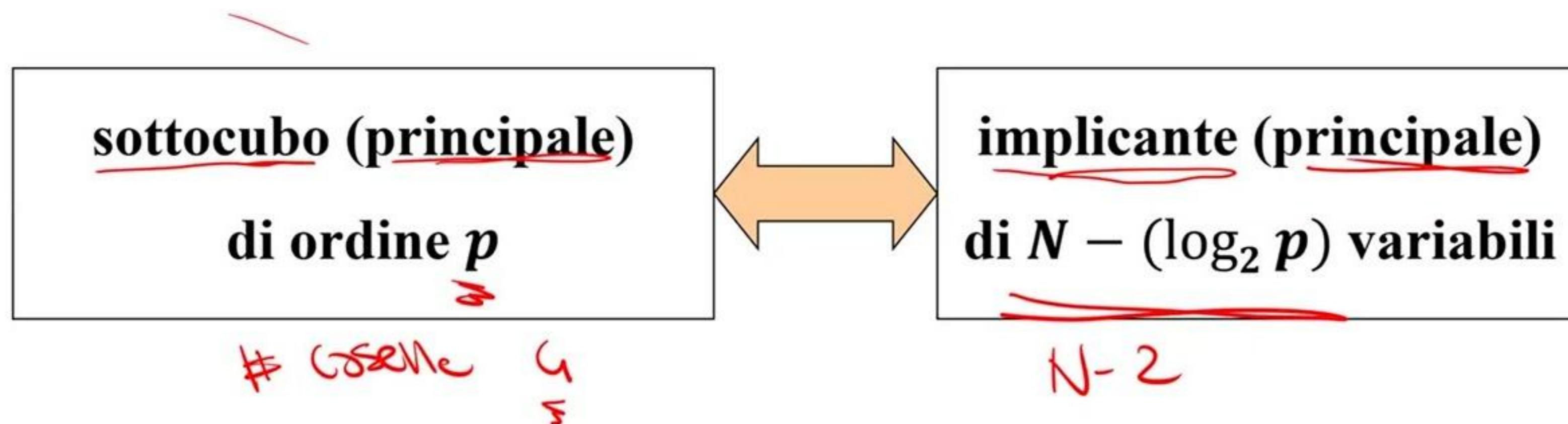
$$Z = x_1 + x_0$$

↑      ↑      →      →

	x <sub>2</sub>	x <sub>1</sub>	x <sub>0</sub>
G	-	1	-
H	-	-	1

# Sottocubi e implicanti

- Sono piu' o meno la stessa cosa



- Cercare i sottocubi principali è facile
  - si fa ad occhio

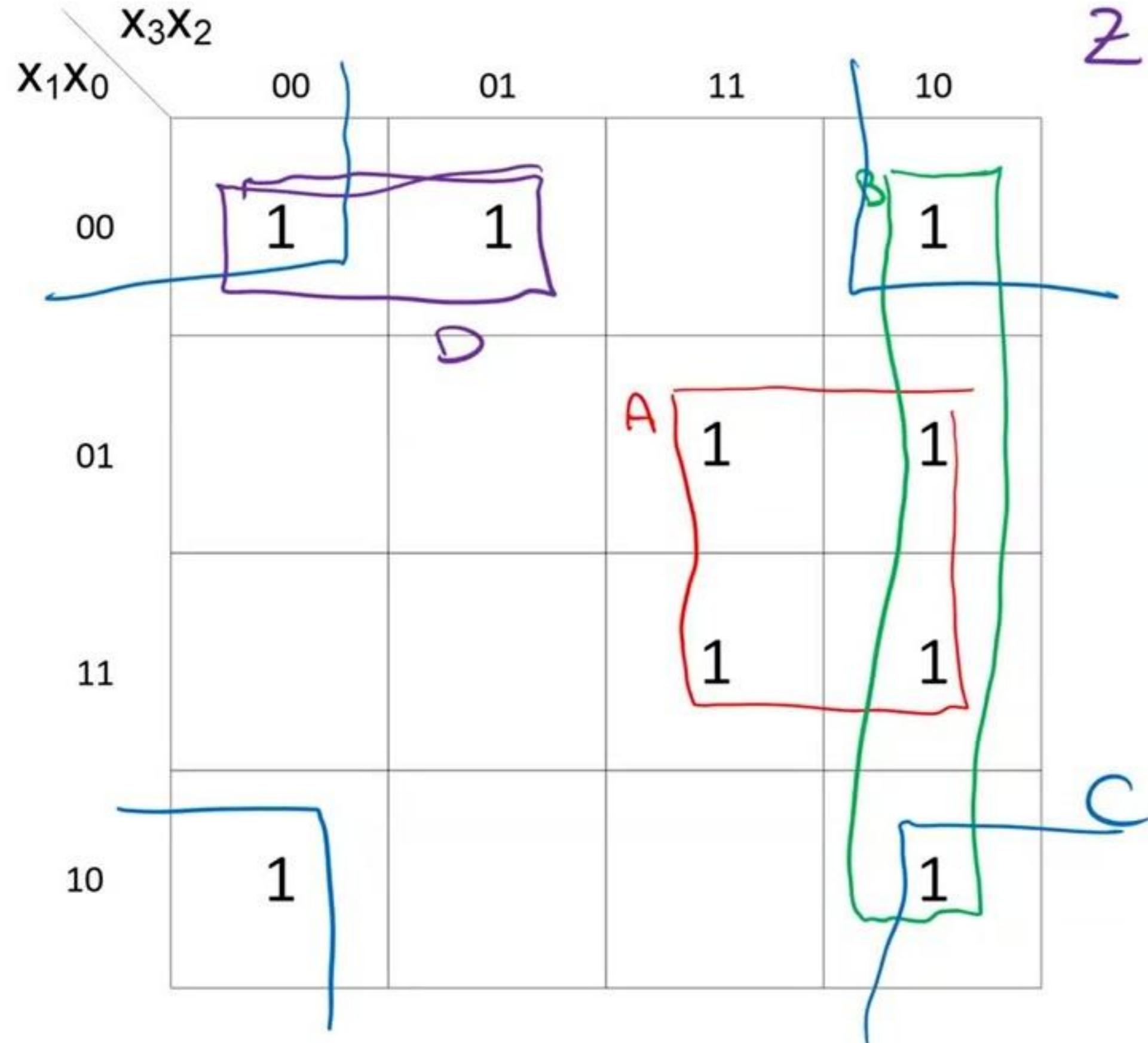
$$\bar{x}_2 \cdot x_1 \cdot x_0$$

# Algoritmo per la ricerca dei S. principali

- Parto dai sottocubi di ordine più grande che trovo sulla mappa
- R
1. Considero i sottocubi di ordine  $p$  non interamente contenuti in sottocubi di ordine piu' grande
    - • Li segno tutti (attenzione!)
    - Non essendo contenuti in sottocubi di ordine maggiore sono senz'altro principali
  2. **Domanda:** l'insieme dei sottocubi che ho trovato finora basta a coprire tutta la mappa?
    - • Se sì, ho finito.
    - Se no,  $p \leftarrow p/2$  e ripeto il ciclo
- ↓
- L'algoritmo termina di sicuro (al più quando  $p = 1$ )

# Esempio

$P = \cancel{X}_2$



## Sottocubi principali

$$Z = X_3 \cdot X_0 + X_3 \cdot \overline{X}_2 + \overline{X}_2 \cdot \overline{X}_0 + \overline{X}_3 \cdot \overline{X}_1 \cdot \overline{X}_0$$

	X <sub>3</sub>	X <sub>2</sub>	X <sub>1</sub>	X <sub>0</sub>
A	1	-	-	1
B	1	0	-	-
C	-	0	-	0
D	0	-	0	0

$$X_3 \cdot X_0$$

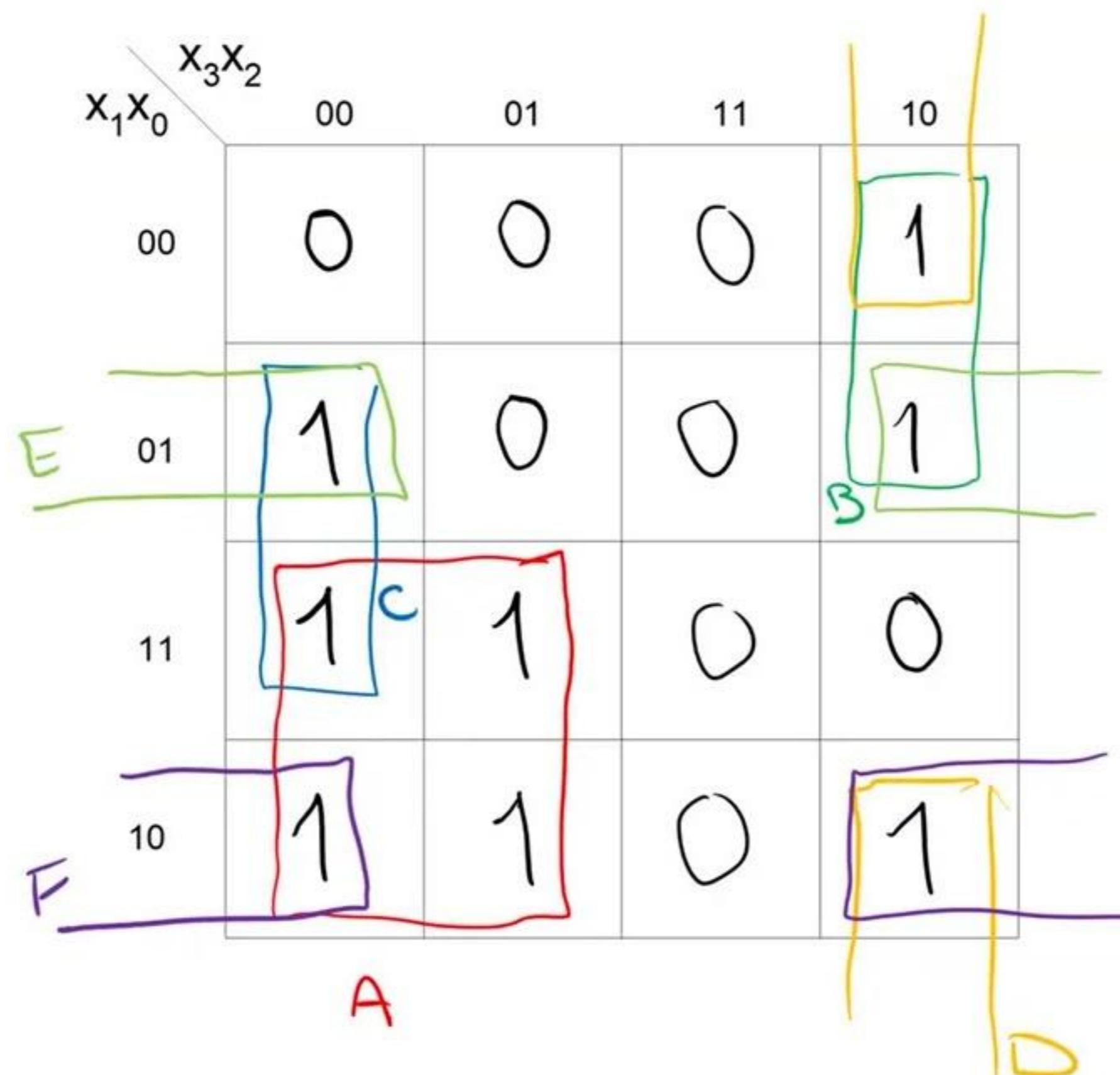
# Esercizio

$P = \times 2$

$$Z = \bar{x}_3 \cdot x_1 + x_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + \bar{x}_3 \cdot \bar{x}_2 \cdot x_0 + x_3 \cdot \bar{x}_2 \cdot \bar{x}_0 + \bar{x}_2 \cdot \bar{x}_1 \cdot x_0 + \bar{x}_2 \cdot x_1 \cdot \bar{x}_0$$

Sottocubi principali

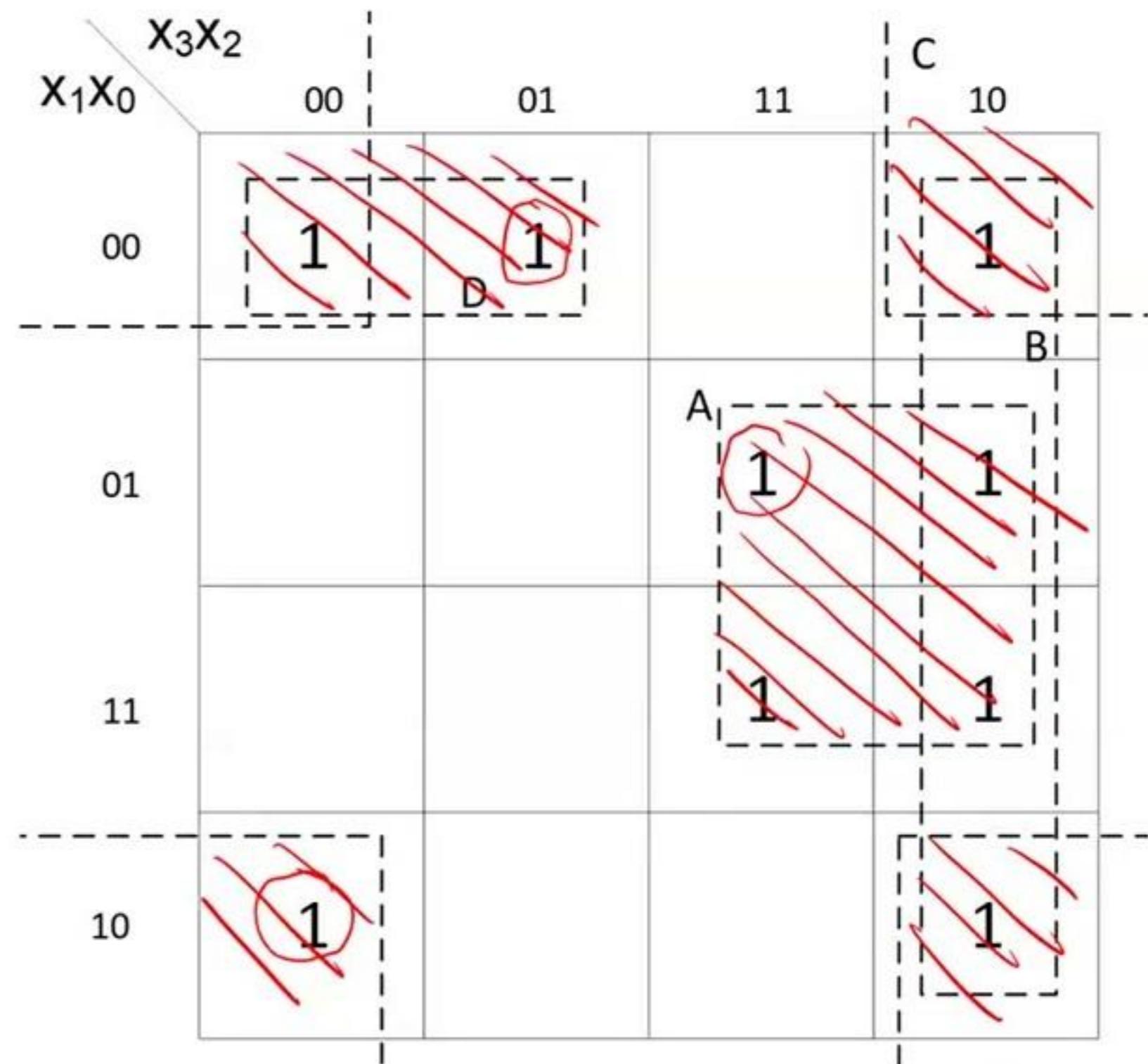
$x_3$	$x_2$	$x_1$	$x_0$	$Z_0$
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



	$x_3$	$x_2$	$x_1$	$x_0$
A	0	-	1	-
B	1	0	0	-
C	0	0	-	1
D	1	0	-	0
E	-	0	0	1
F	-	0	1	0

# Ricerca delle liste di copertura non ridondanti

- I sottocubi principali vanno classificati B?
- Alcuni sono **gli unici a coprire un dato sottocubo di ordine 1.**
  - non possono essere tolti, se vogliamo coprire tutti gli stati riconosciuti dalla rete.
  - Sono detti essenziali, e costituiscono il **cuore della mappa**.
- Come trovarli?
  - Basta guardare se ci sono degli 1 cerchiati una volta sola.



$$z = \bar{x}_3 \cdot x_0 + \bar{x}_2 \cdot \bar{x}_0 + \bar{x}_3 \cdot x_1 \cdot x_0$$

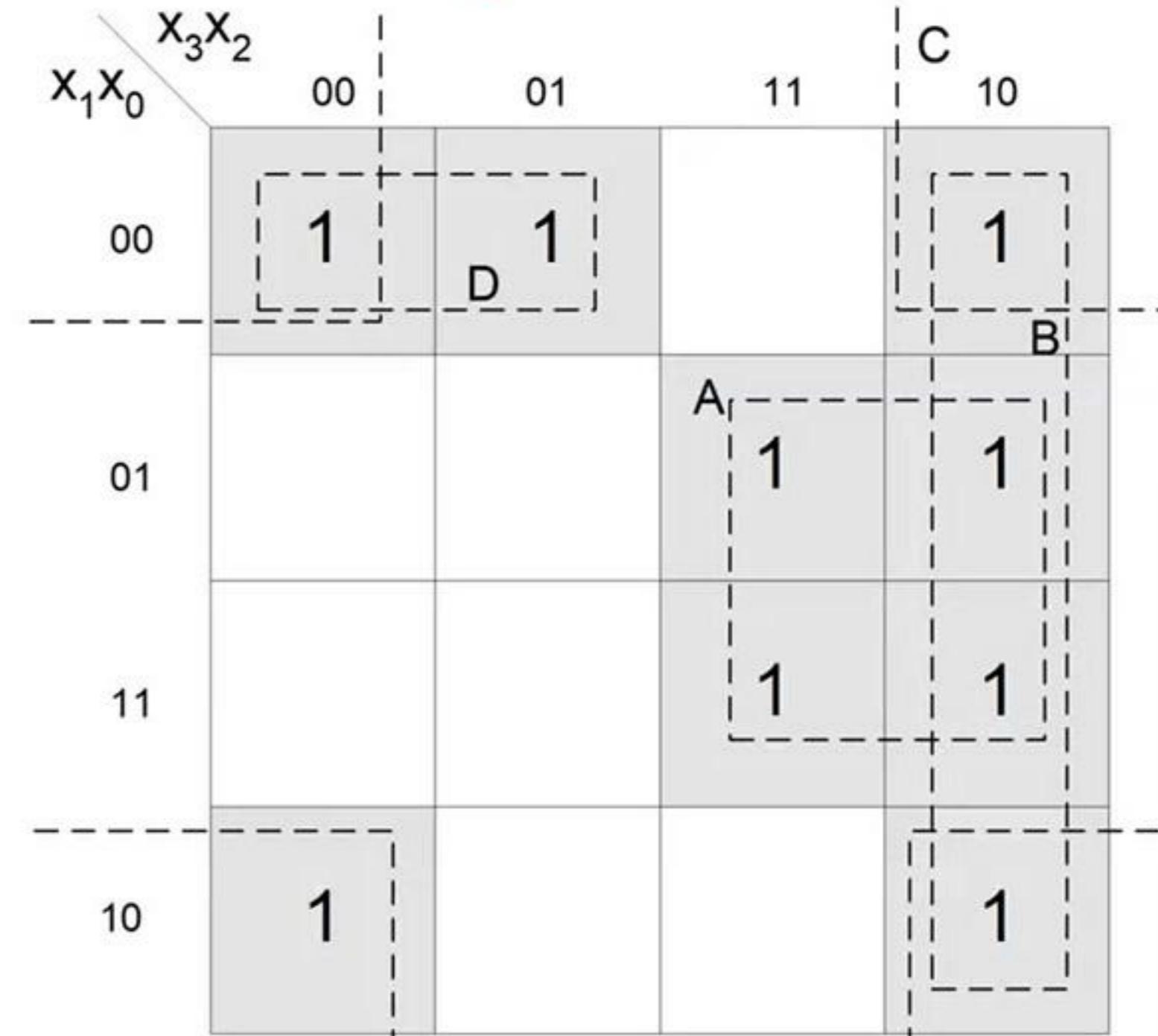
## Sottocubi essenziali e assolutamente eliminabili

- Si spuntano tutti i sottocubi di ordine 1 contenuti in sottocubi principali **essenziali**
- Se si copre completamente **anche** qualche altro sottocubo
  - quel sottocubo è **ridondante**.
  - **assolutamente eliminabile, non deve comparire in una lista di copertura irridondante.**
- Un sottocubo principale è **assolutamente eliminabile** se riconosce **soltanto** stati di ingresso già riconosciuti da sottocubi principali **essenziali**.

$\{A, B, C, D\}$

A, C, D  
B

css.  
ass. el.m.



{A, C, D} l.c. GJE minimo

P=2

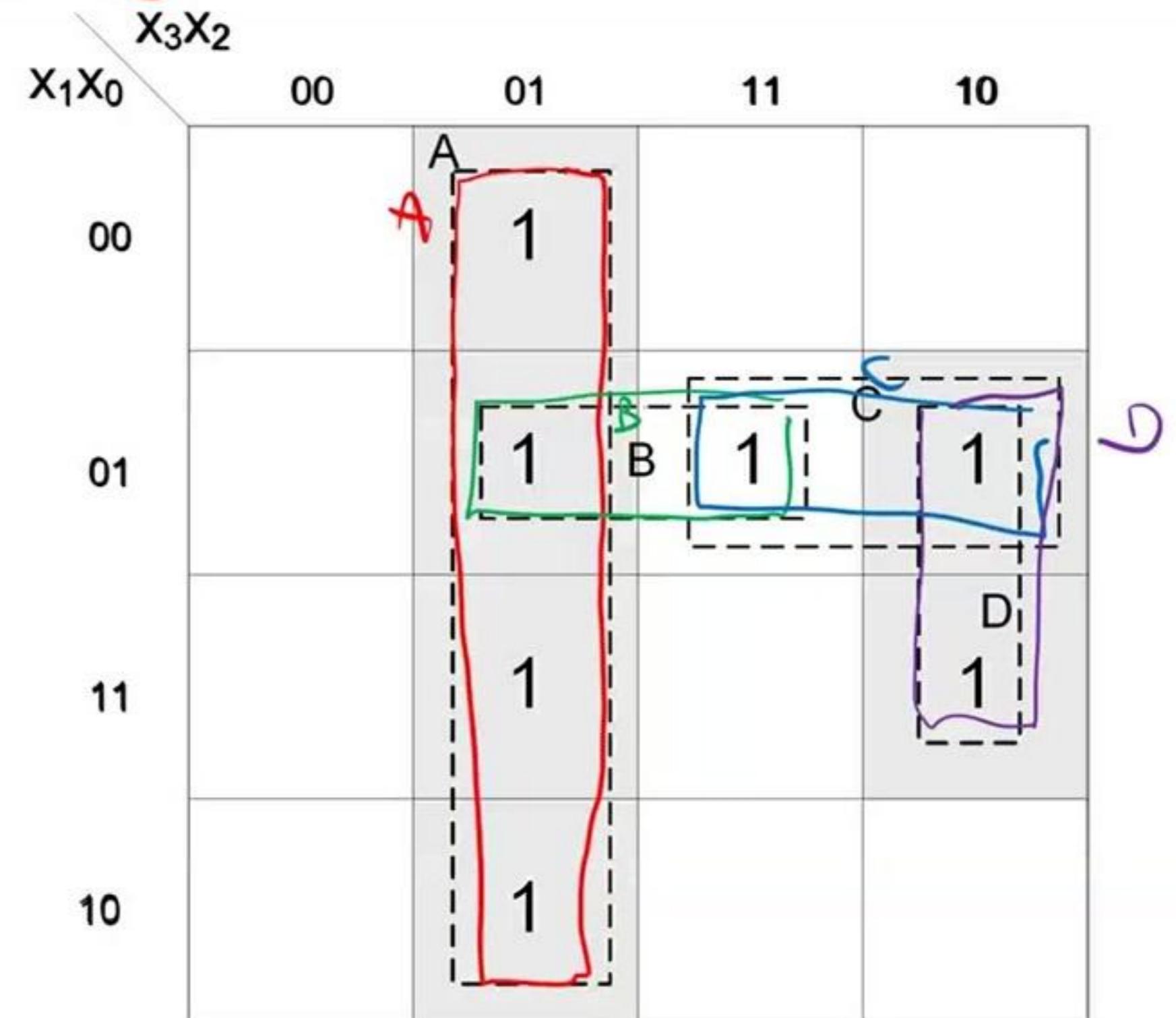
## Classificazione dei sottocubi

A,D

• Ipotesi alternative

- Ricerca dei **sottocubi principali**: ce n'è uno di ordine 4 (A), e poi devo passare a quelli di ordine 2 (B,C,D)

	$x_3$	$x_2$	$x_1$	$x_0$	
A	0	1	-	-	ESS
B	-	1	0	1	
C	1	-	0	1	
D	1	0	-	1	ESS



Semplificazione eliminabile

# Lista di copertura di costo minimo

- Generare **tutte le possibili liste di copertura non ridondanti** che
  - includono **tutti** i sottocubi principali **essenziali**
  - non includono **nessun** sottocubo principale **assolutamente eliminabile**
  - includono **un sottoinsieme** di sottocubi principali semplicemente eliminabili
- Tra tutte queste, **valuto quella di costo minimo** applicando il criterio di costo richiesto.

# Lista di copertura di costo minimo (cont.)

- Considero un **qualsiasi** sottocubo S.E.
  - formulo **due ipotesi alternative**

a) **Come se fosse essenziale**

- lo aggiungo alla lista di copertura
- qualche **altro sottocubo** diventa **assolutamente eliminabile**

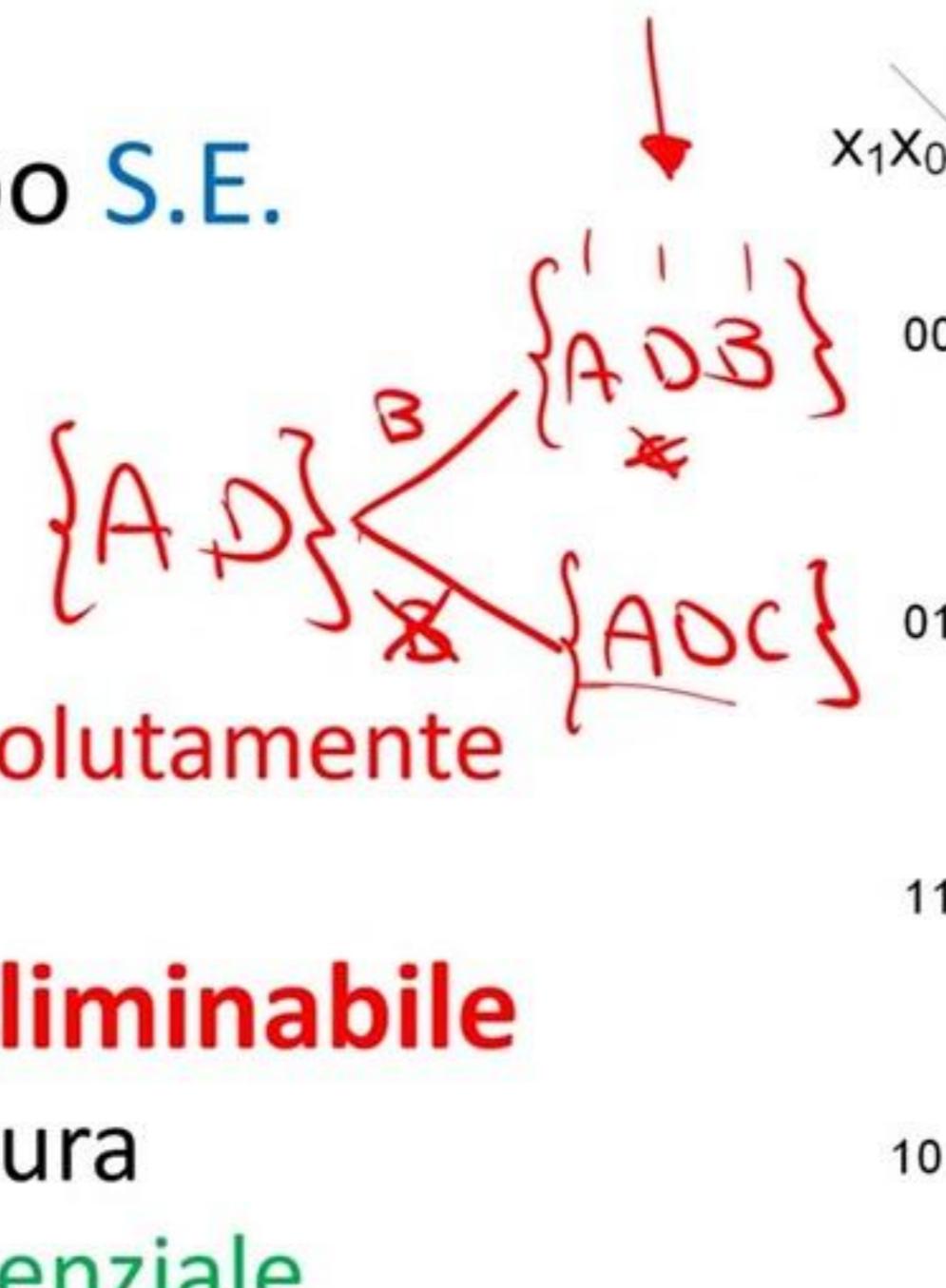
b) **Come se fosse assolutamente eliminabile**

- lo tolgo da qualunque lista di copertura
- qualche **altro sottocubo** diventa **essenziale**

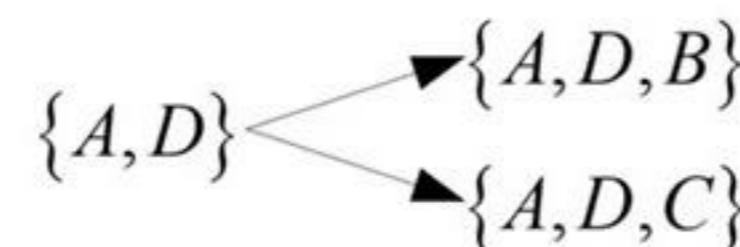
**criterio**

• **Albero binario di decisioni**

- Foglie sono L.C. non ridondanti



x <sub>3</sub> x <sub>2</sub>	x <sub>1</sub> x <sub>0</sub>	00	01	11	10
x <sub>3</sub> x <sub>2</sub>	00	A	1		
x <sub>3</sub> x <sub>2</sub>	01	B	1	C	1
x <sub>3</sub> x <sub>2</sub>	11		1		
x <sub>3</sub> x <sub>2</sub>	10		1	D	1

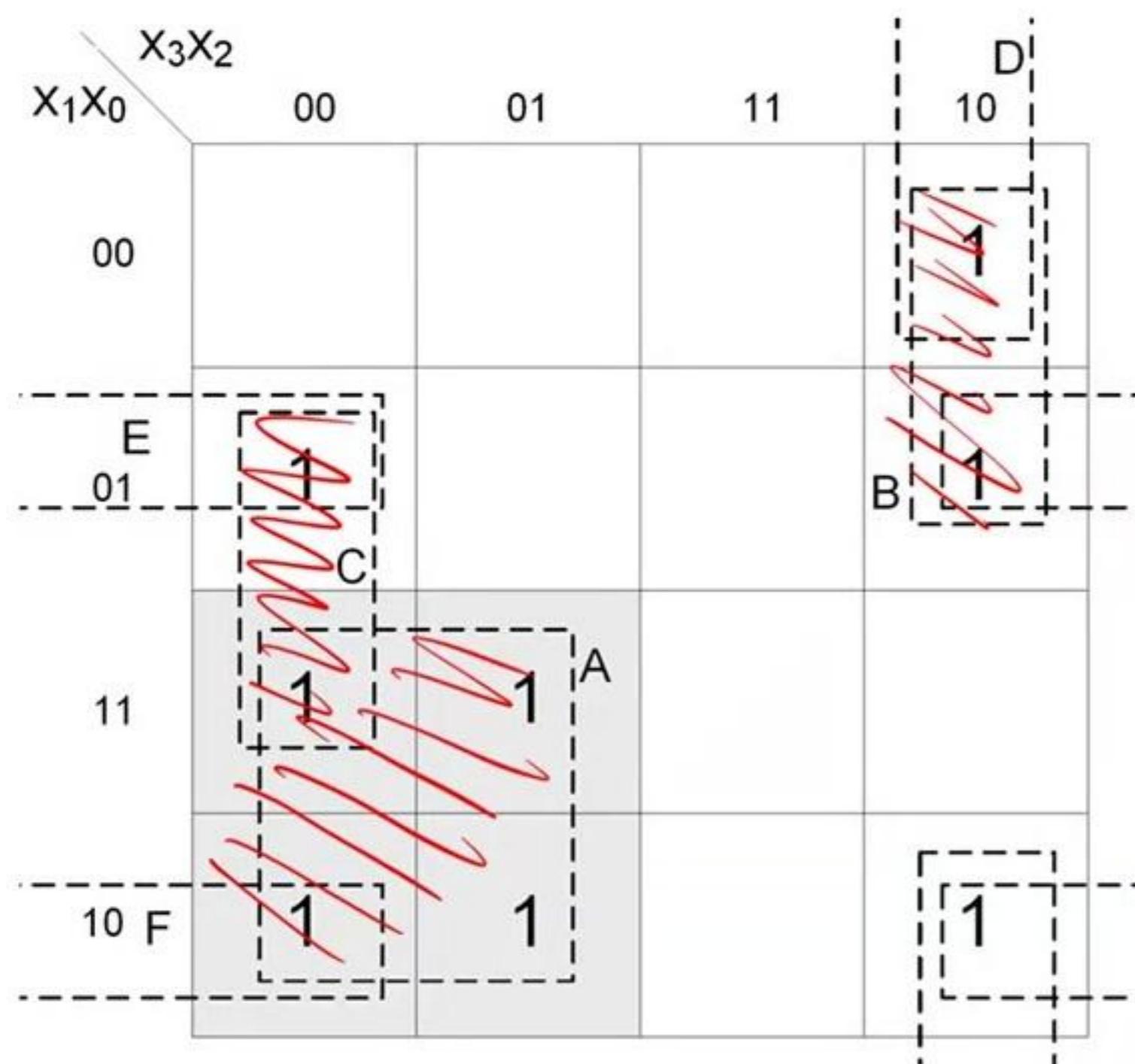


$$z = \overline{x_3} \cdot x_2 + x_3 \cdot \overline{x_2} \cdot x_0$$

$$\begin{aligned} & \overline{x_3} \cdot x_2 \\ &+ x_3 \cdot \overline{x_2} \cdot x_0 \\ &+ x_3 \cdot x_1 \cdot x_0 \end{aligned}$$

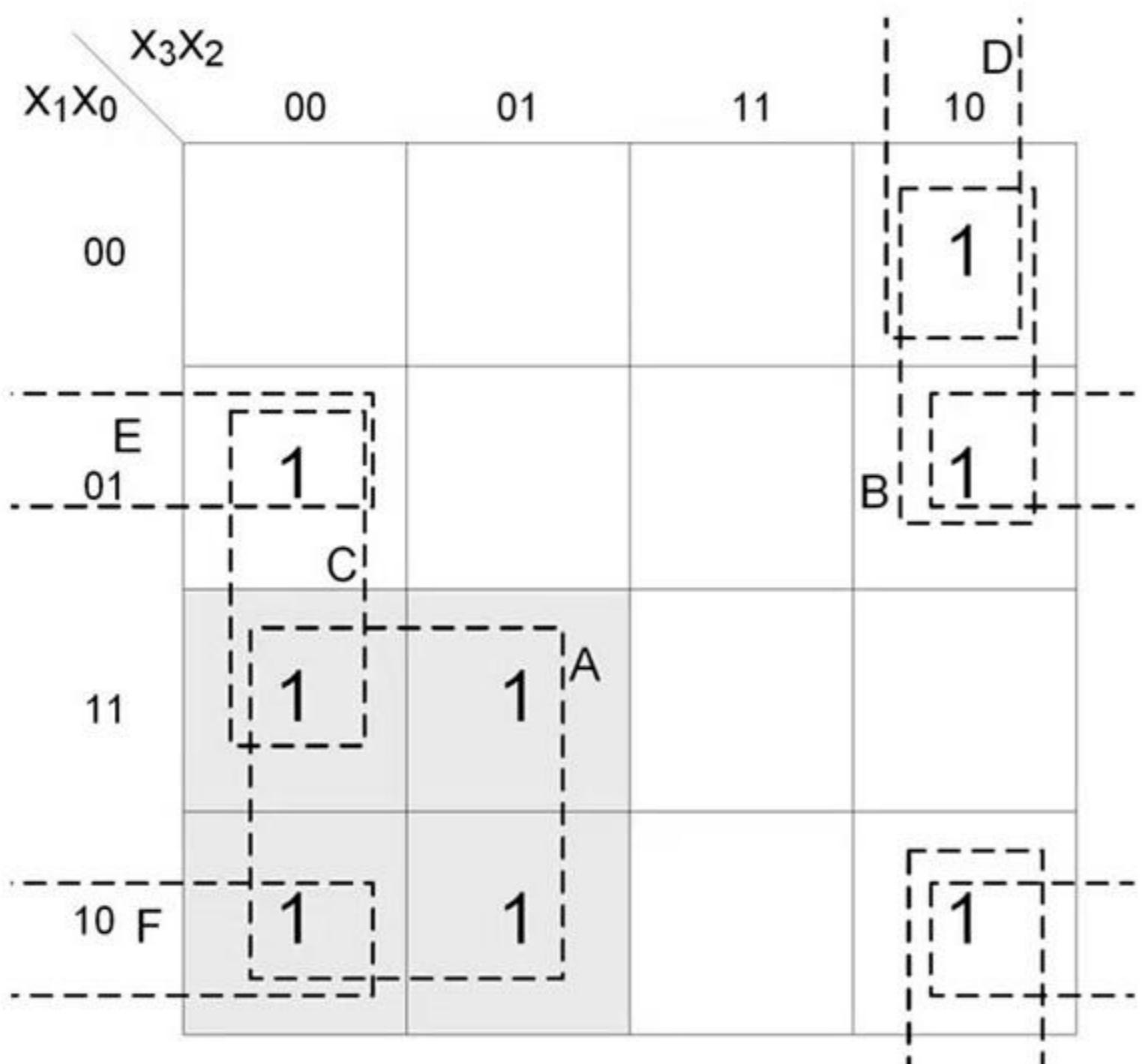
# Esempio

- Classificare gli implicanti e trovare tutte le LC non ridondanti

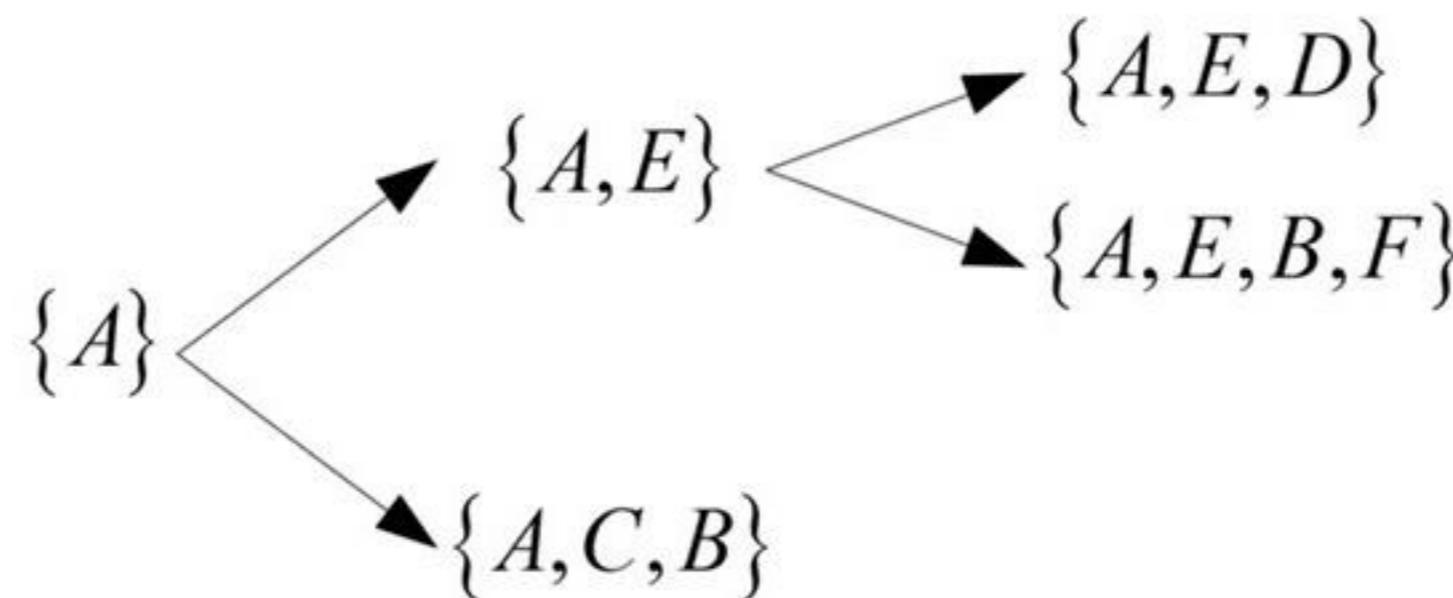


• S. essenziali: A

# Esempio (cont.)



- S. essenziali: **A**
- S. assolutamente eliminabili: **nessuno**
- S. semplicemente eliminabili: **B,C,D,E,F**
- Decisione su **D**
  - Come se **ESS** -> B,F diventano **AE**
  - Come se **AE** -> B,F diventano **ESS**

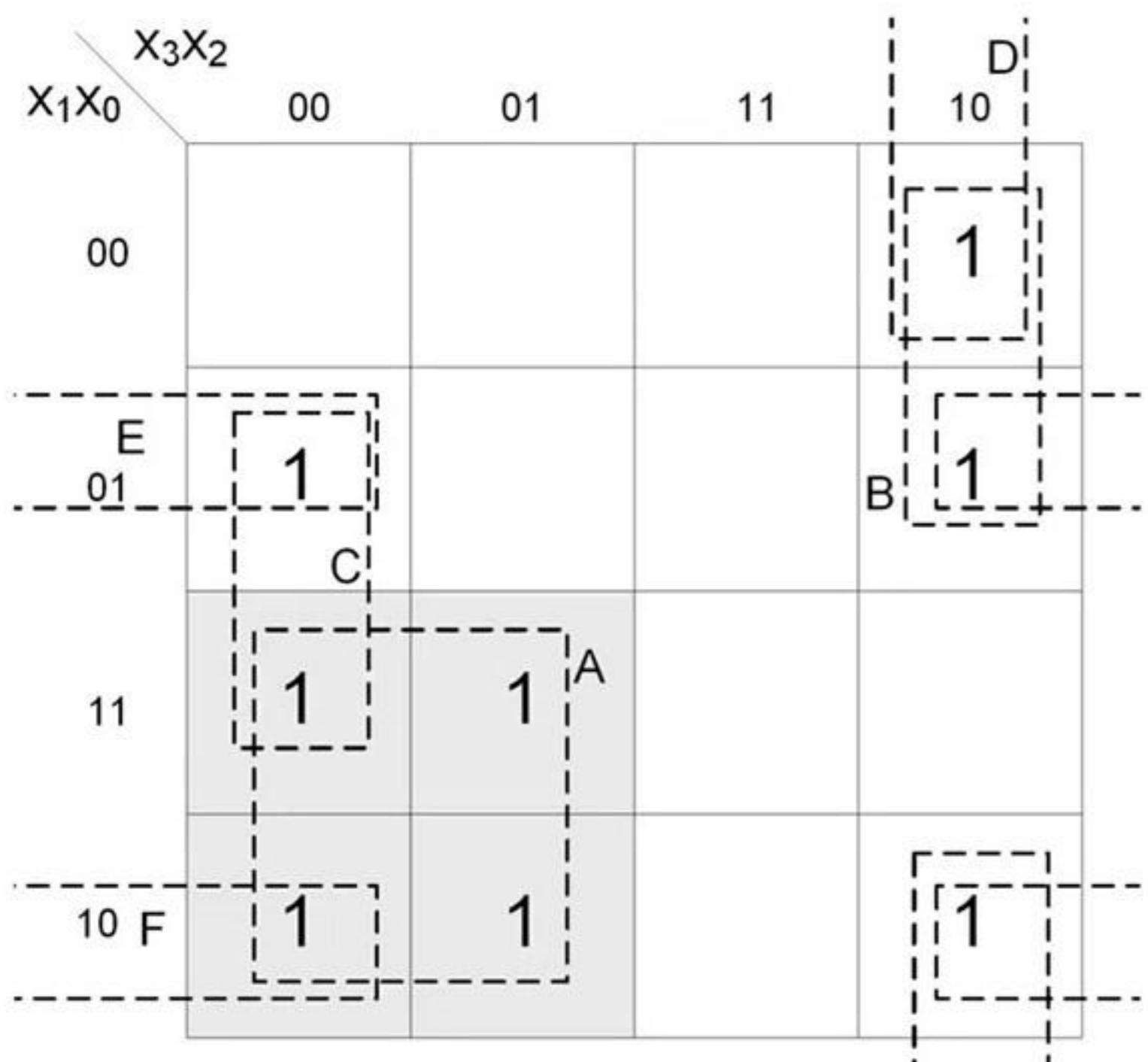


## Esempio (cont.)

$$Z = \overline{x_3} \cdot x_5 + \overline{x_2} \cdot \overline{x_1} \cdot x_6 + x_3 \cdot \overline{x_2} \cdot \overline{x_6}$$

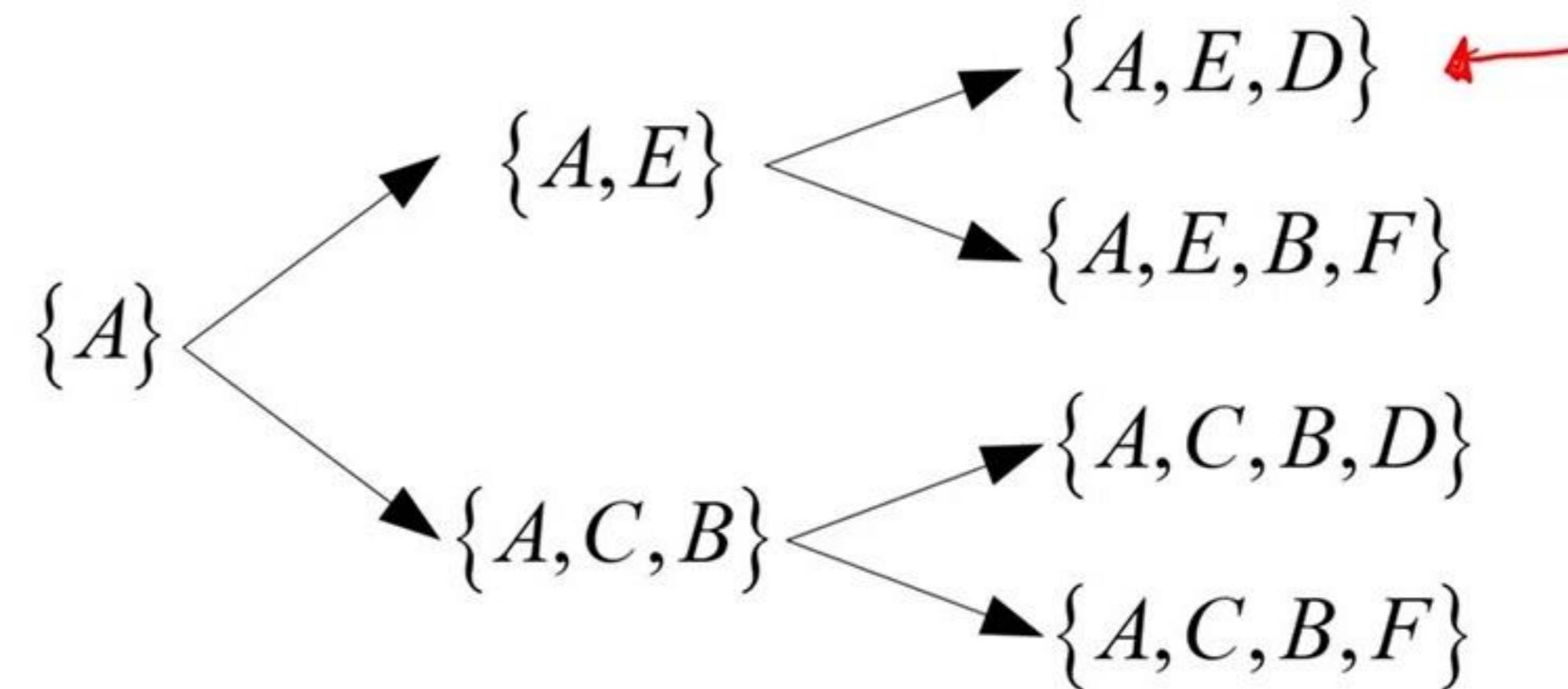
A              E              D

- Ramo inferiore {A,C,B}



- Decisione su D

- Come se ESS -> F diventa AE
- Come se AE -> F diventa ESS

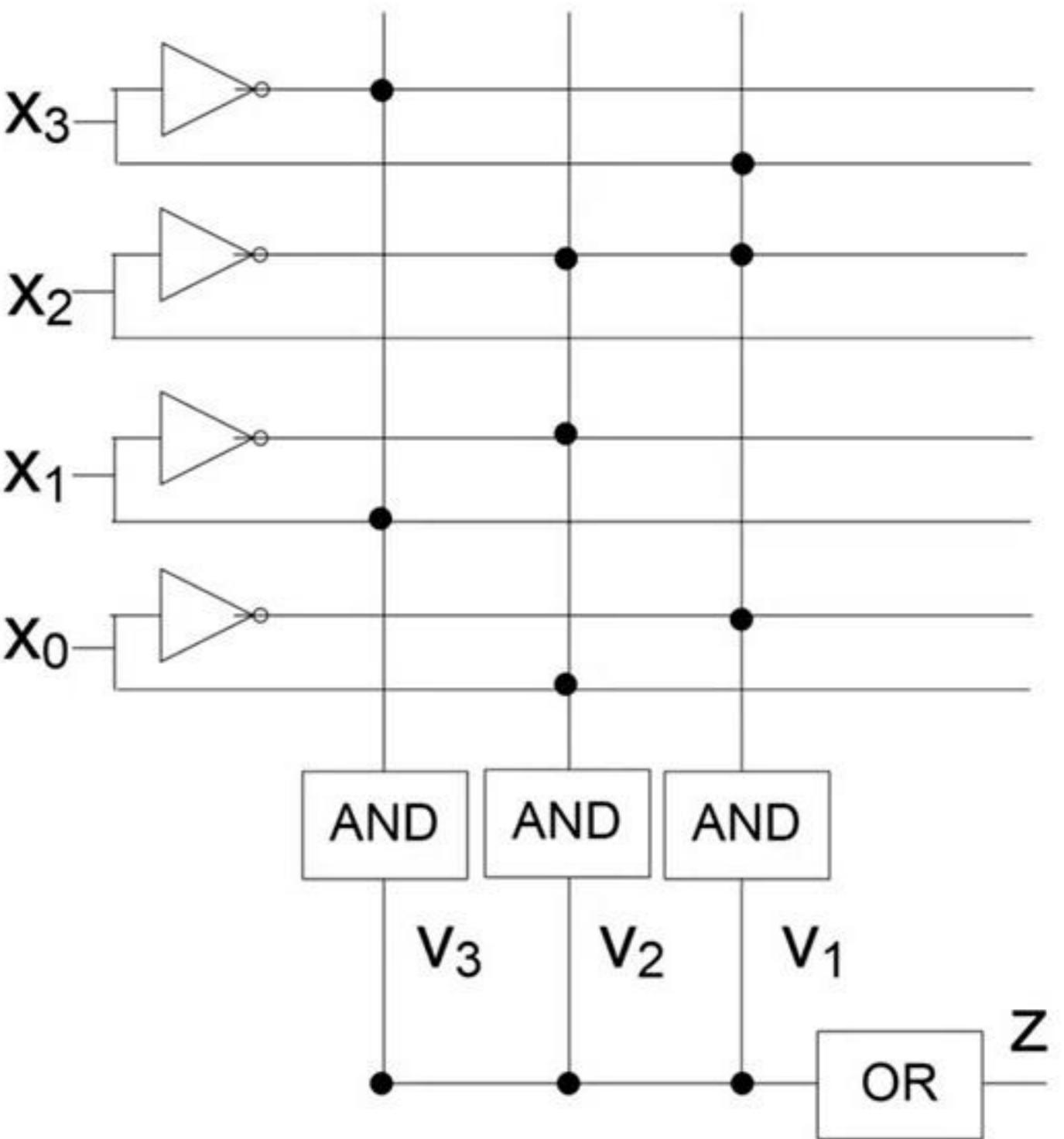
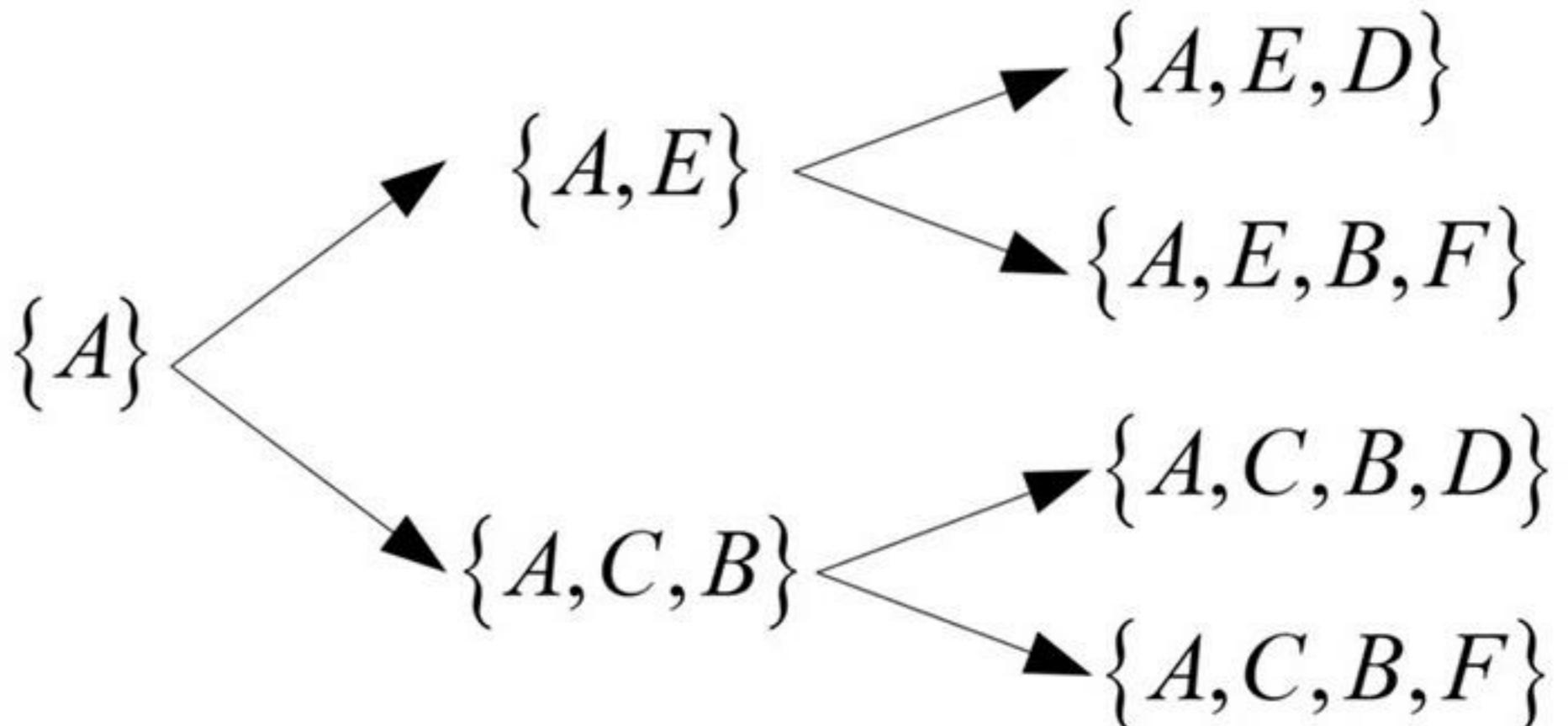


## Esempio (fine)

- ho tutte le liste di copertura non ridondanti

- Uso il criterio di costo
- Trovo la LC di costo minimo

$$z = \overline{x_3} \cdot x_1 + \overline{x_2} \cdot \overline{x_1} \cdot x_0 + x_3 \cdot \overline{x_2} \cdot \overline{x_0}$$



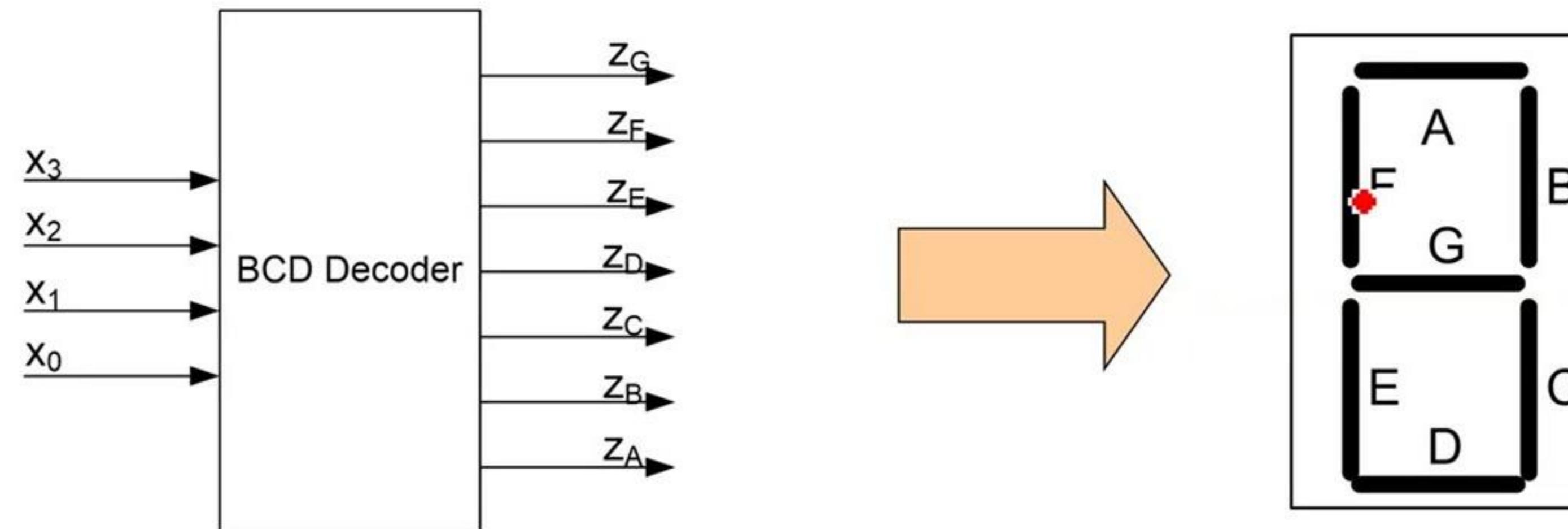
# Esercizio - sintesi di leggi non completamente specificate

Binary-Coded Decimal

- **Decodificatore BCD a 7 segmenti**

- 4 variabili di ingresso,
- interpretate come la **codifica in base 2** di una cifra decimale,
- **7 uscite**, che vanno ad accendere i segmenti di un display LCD

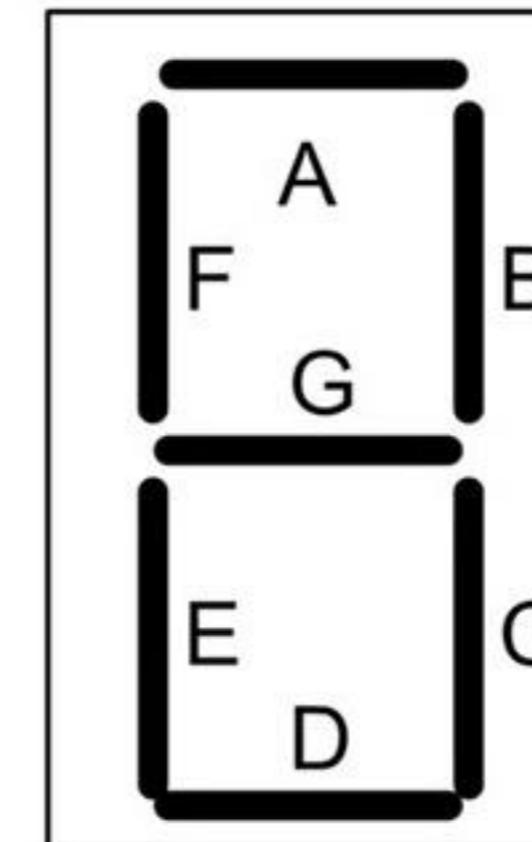
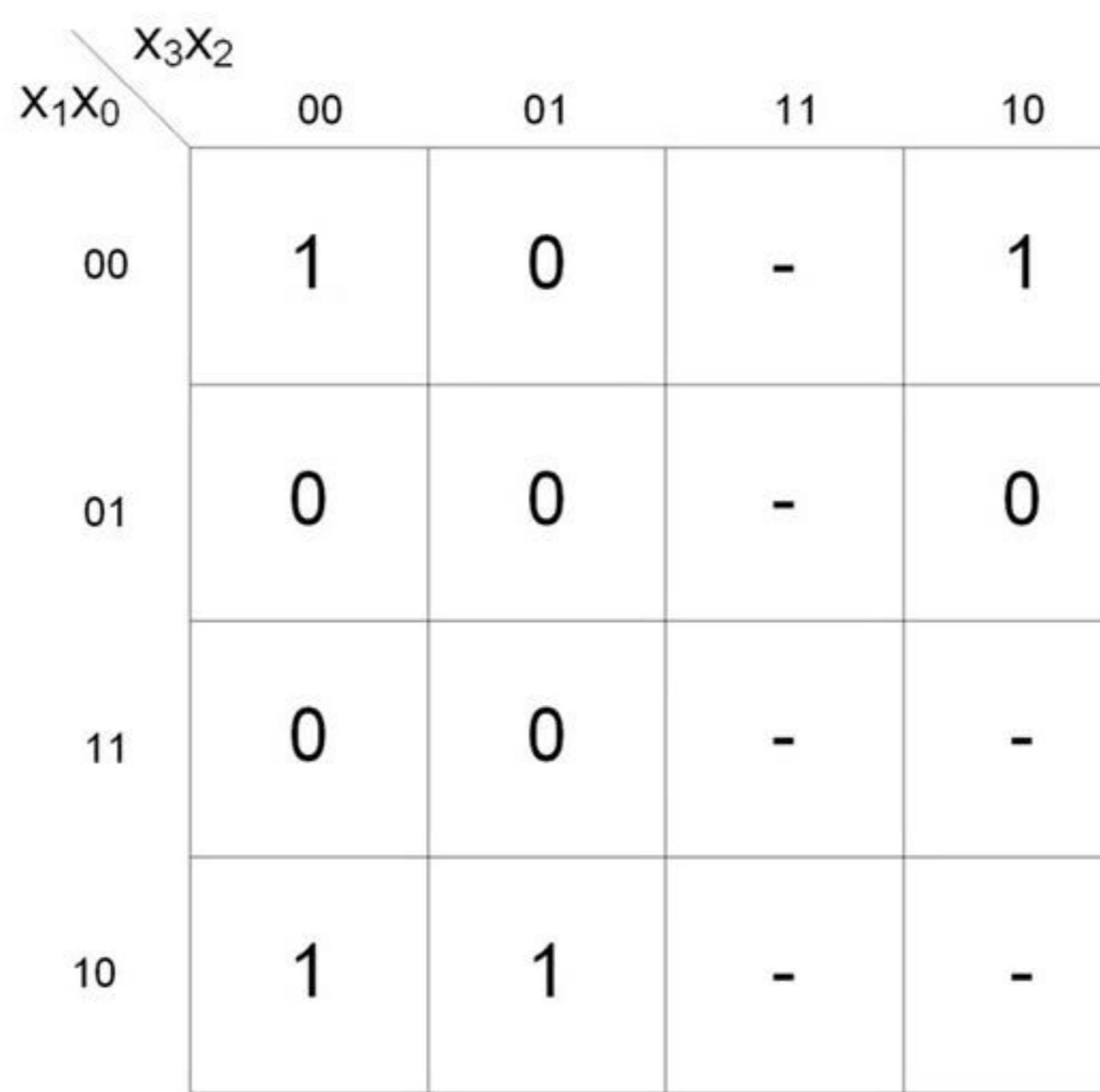
0 - - 9



## Esercizio (cont.)

- Sintesi a costo minimo dell'uscita  $z_F$

$J$	$x_3$	$x_2$	$x_1$	$x_0$	ZE
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	0
?	1	0	1	0	-
?	1	0	1	1	-
?	1	1	0	0	-
?	1	1	0	1	-
?	1	1	1	0	-
?	1	1	1	1	-



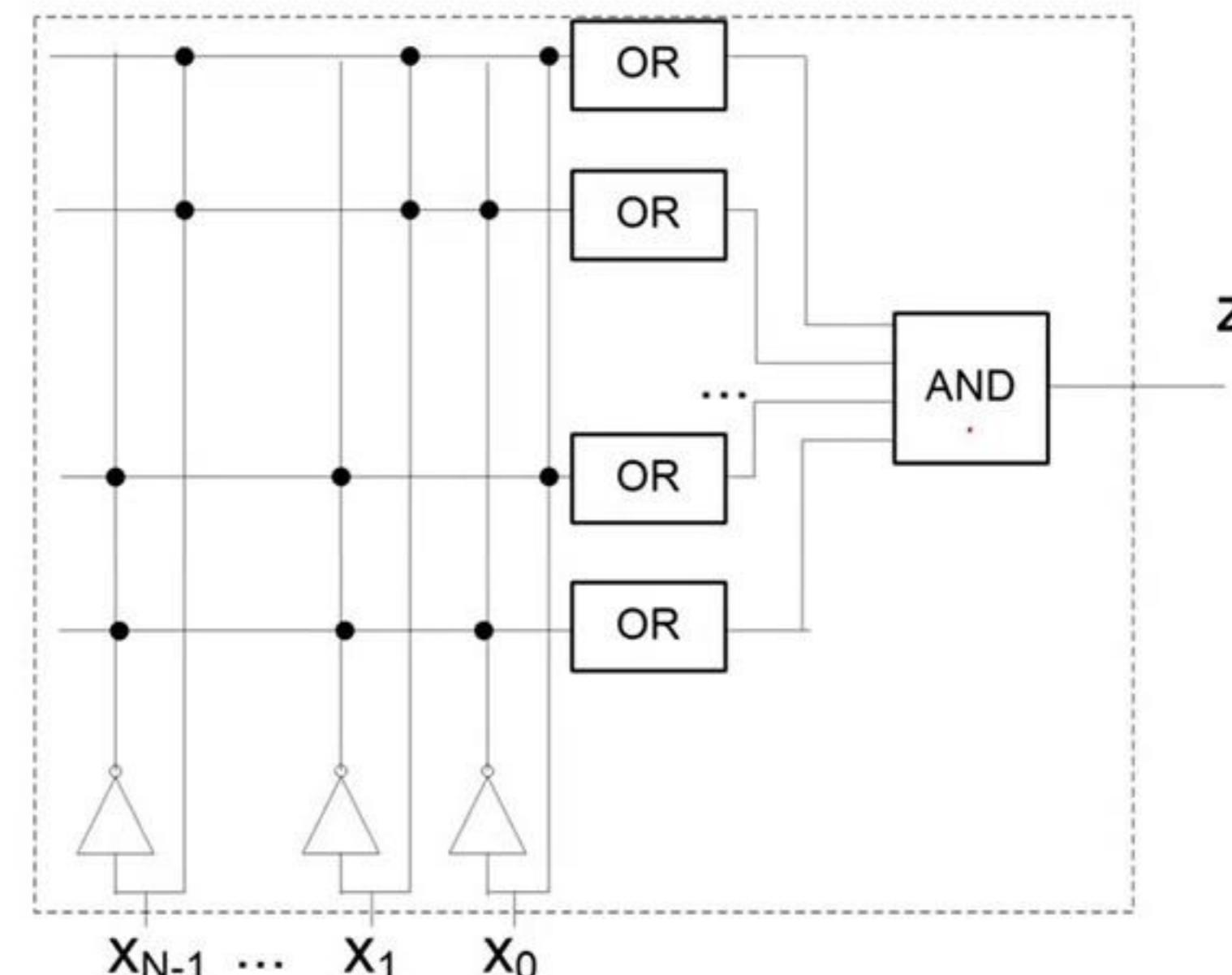
- Considero “-”  $\leftrightarrow$  1
  - Trovo IP piu’ grandi  
  - Considero “-”  $\leftrightarrow$  0
  - Non devo coprire “-”

# Sintesi in forma PS

- Esiste anche – **sempre** – la possibilità di sintetizzare una rete in formato **PS**, cioè prodotto di somme.
- Teoria **duale**: espansione di Shannon, forma canonica PS, maxtermini, implicati, etc.
  - • Che non faremo
    - Esiste un metodo più veloce
    - Che sfrutta conoscenze già acquisite

$$z = S_1 \cdot S_2 \cdot \dots \cdot S_k$$

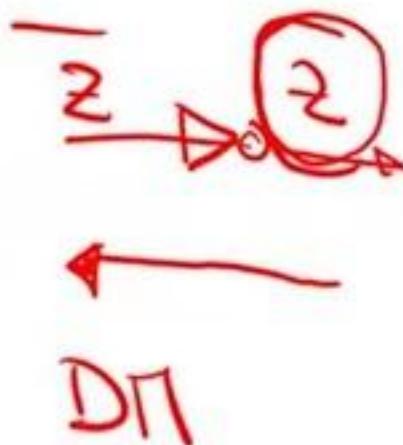
$$S_i = \sum x_j$$



## Sintesi in forma PS (cont.)

(F)

$\bar{z}$	$\bar{z}$
0	1
1	0



- **Punto 1:** data  $F$ , ricavo  $\bar{F}$ 
  - In pratica, riscrivo la tabella di verità con 1 al posto dello 0 e viceversa.
- **Punto 2:** realizzo una sintesi SP della legge  $\bar{F}$
- **Punto 3:** ottengo una sintesi di  $F$  inserendo un **invertitore** in uscita alla rete SP che sintetizza  $\bar{F}$ .
- **Punto 4:** applico i **Teoremi di De Morgan**
  - da destra verso sinistra

# Sintesi in forma PS (cont.)

F

$$\overline{Y_1 + \dots + Y_K} = \overline{Y_1} \cdot \dots \cdot \overline{Y_K}$$

- Punto 1,2**

- Sintesi di  $\overline{F}$  in forma SP

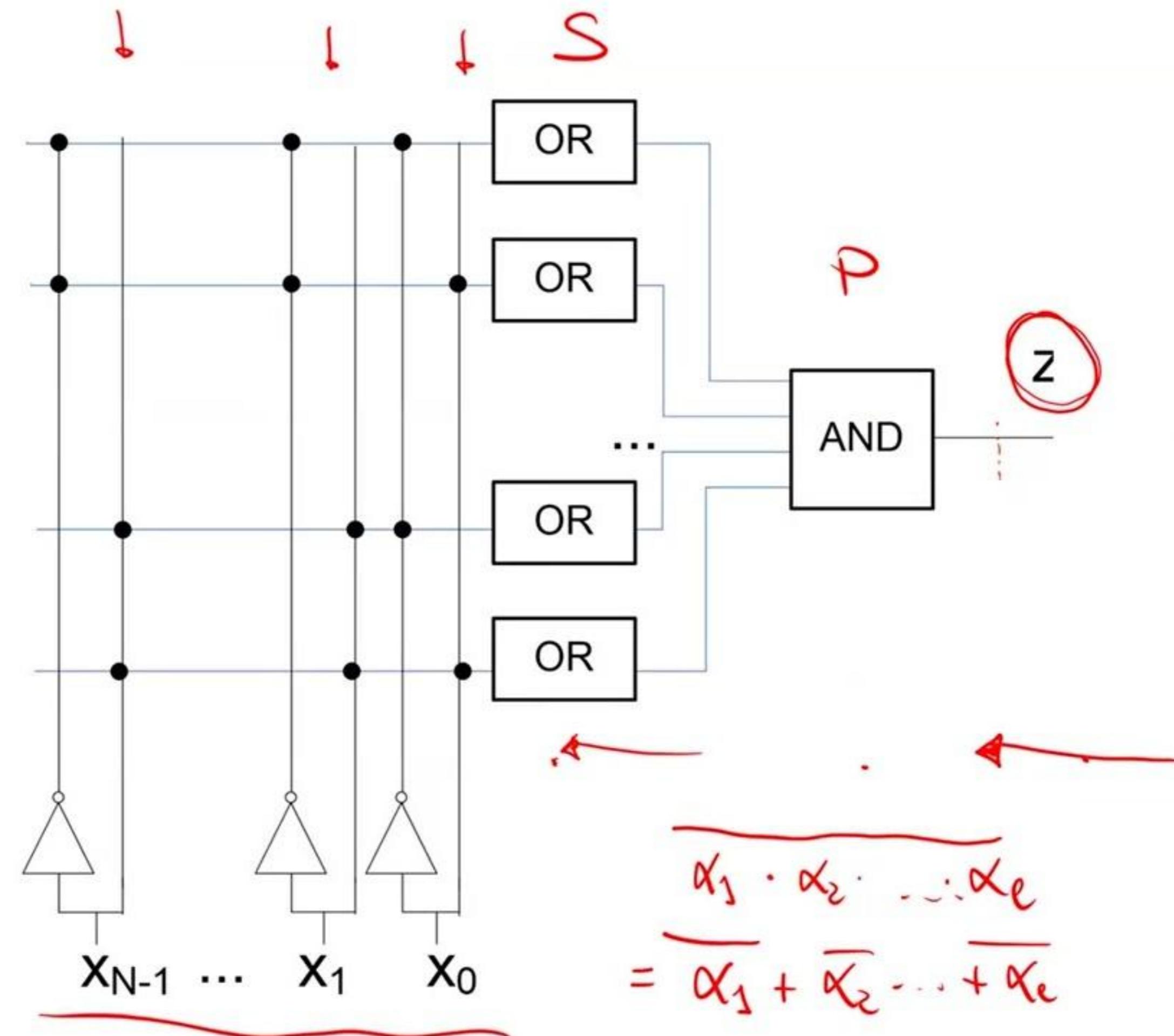
- Punto 3**

- Applico un invertitore
- Ottengo z

- Punto 4**

- Applico DM:  $\overline{a + b} = \overline{a} \cdot \overline{b}$

- Applico DM:  $\overline{a \cdot b} = \overline{a} + \overline{b}$



## Sintesi in forma PS (cont.)

- Dal punto di vista algebrico:

- scrivo  $\bar{F}$  in forma SP:  $\bar{z} = P_1 + P_2 + \dots + P_k$
- $P_i$  sono prodotti di variabili *di ingresso*

- Applico De Morgan, ottenendo:

$$z = \bar{\bar{z}} = \overline{P_1 + P_2 + \dots + P_k} = \overline{P_1} \cdot \overline{P_2} \cdot \dots \cdot \overline{P_k}$$

•

- Applico ancora De Morgan per ottenere

$$\overline{P_i} = \overline{\prod x_j} = \sum \overline{x_j}$$

## Sintesi in forma PS (cont.)

- Con questo procedimento
- Se  $\bar{F}$  è in forma canonica SP, allora  $F$  è in forma canonica PS
- • Se la sintesi SP di  $\bar{F}$  costa **C**, allora la sintesi PS di  $F$  costa **C**
- | • Se la sintesi SP di  $\bar{F}$  è di costo minimo (tra le possibili sintesi SP), lo è anche la sintesi PS di  $F$  (tra le possibili sintesi PS)
  - Se esistesse una sintesi PS di  $F$  a costo minore, applicando DM<sup>2</sup> troverei una sintesi SP di  $\bar{F}$  a costo **minore del minimo**, contro l'ipotesi

$$\rightarrow \underset{\text{SP}(\bar{F})}{\text{SP}(\bar{F})} \equiv \underset{\text{PS}(\bar{F})}{\text{PS}(\bar{F})} \quad \Delta_0 \quad Dn^2$$
$$SP(\bar{F}) \equiv PS(\bar{F})$$

## Riepilogando

$$\boxed{\text{SP } \bar{F} = \text{PS } \bar{F}}$$

$\text{SP } \bar{F}$        $\text{PS } \bar{F}$

- Data una legge  $\bar{F}$ , sappiamo effettuarne la sintesi a **costo minimo in forma SP**
- Sappiamo anche effettuarne la sintesi a **costo minimo in forma PS**
  - Sintetizziamo  $\bar{F}$  a costo minimo in forma SP
  - Aggiungiamo un invertitore ed applichiamo DM due volte
- Quale delle due sintesi a costo minimo (di F) costa meno?
  - Non e' possibile dirlo
  - Dipende dalla tabella di verita'

SP  
PS

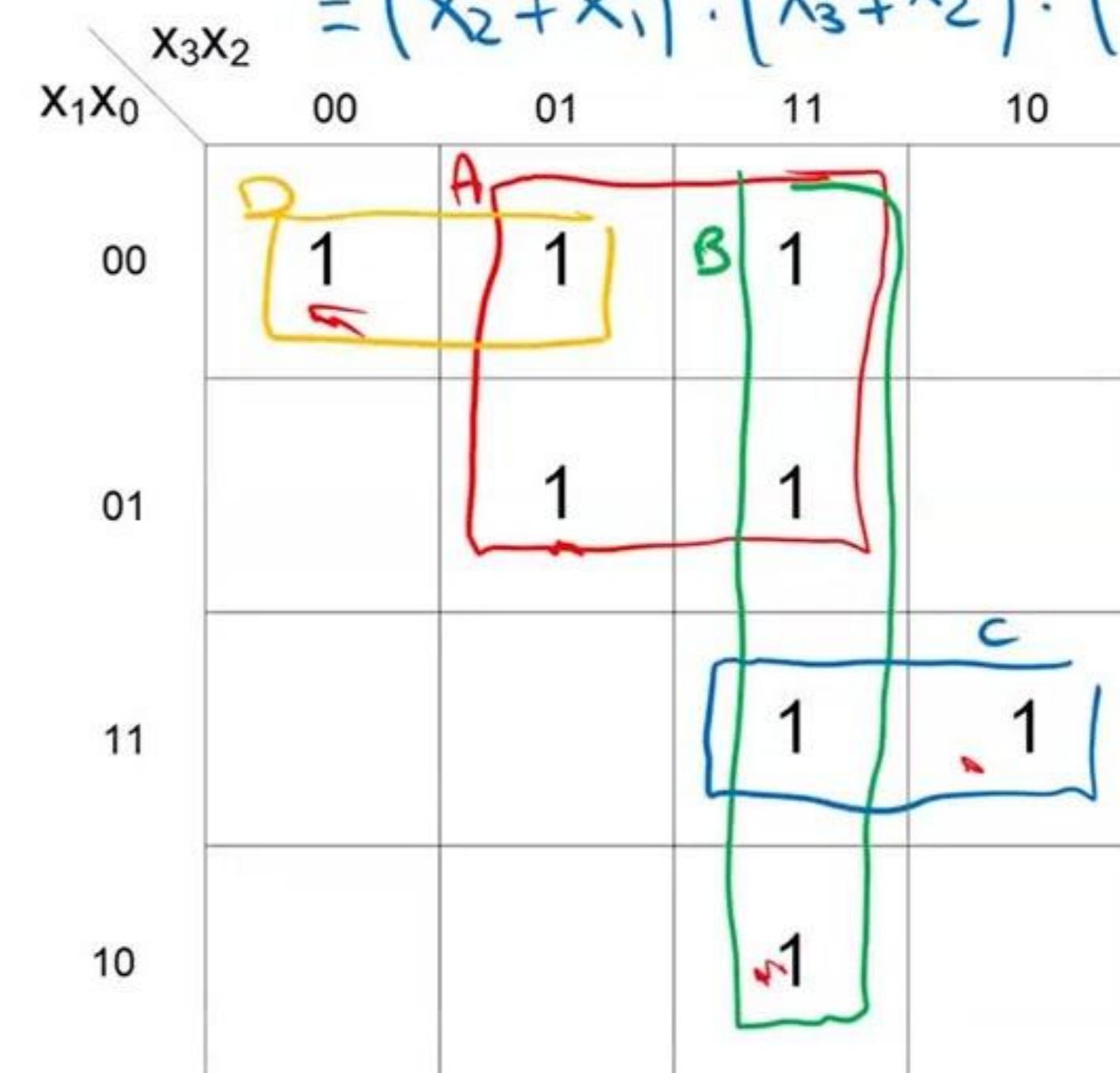
Esercizio  $\frac{z}{\overline{z}} = \overline{\overline{z}} = \overline{x_2 \cdot \overline{x}_1 + x_3 \cdot x_2 + x_3 \cdot x_1 \cdot x_0 + \overline{x}_3 \cdot \overline{x}_1 \cdot \overline{x}_0}$

- Sintesi PS a costo minimo di legge  $F$  già vista

$$\cdot SP \overline{z} \\ \cdot PS z \\ = (\overline{x}_2 + x_1) \cdot (\overline{x}_3 + \overline{x}_2) \cdot (\overline{x}_3 + \overline{x}_1 + \overline{x}_0) \cdot (x_3 + x_1 + x_0)$$



Mappa di Karnaugh per  $F$



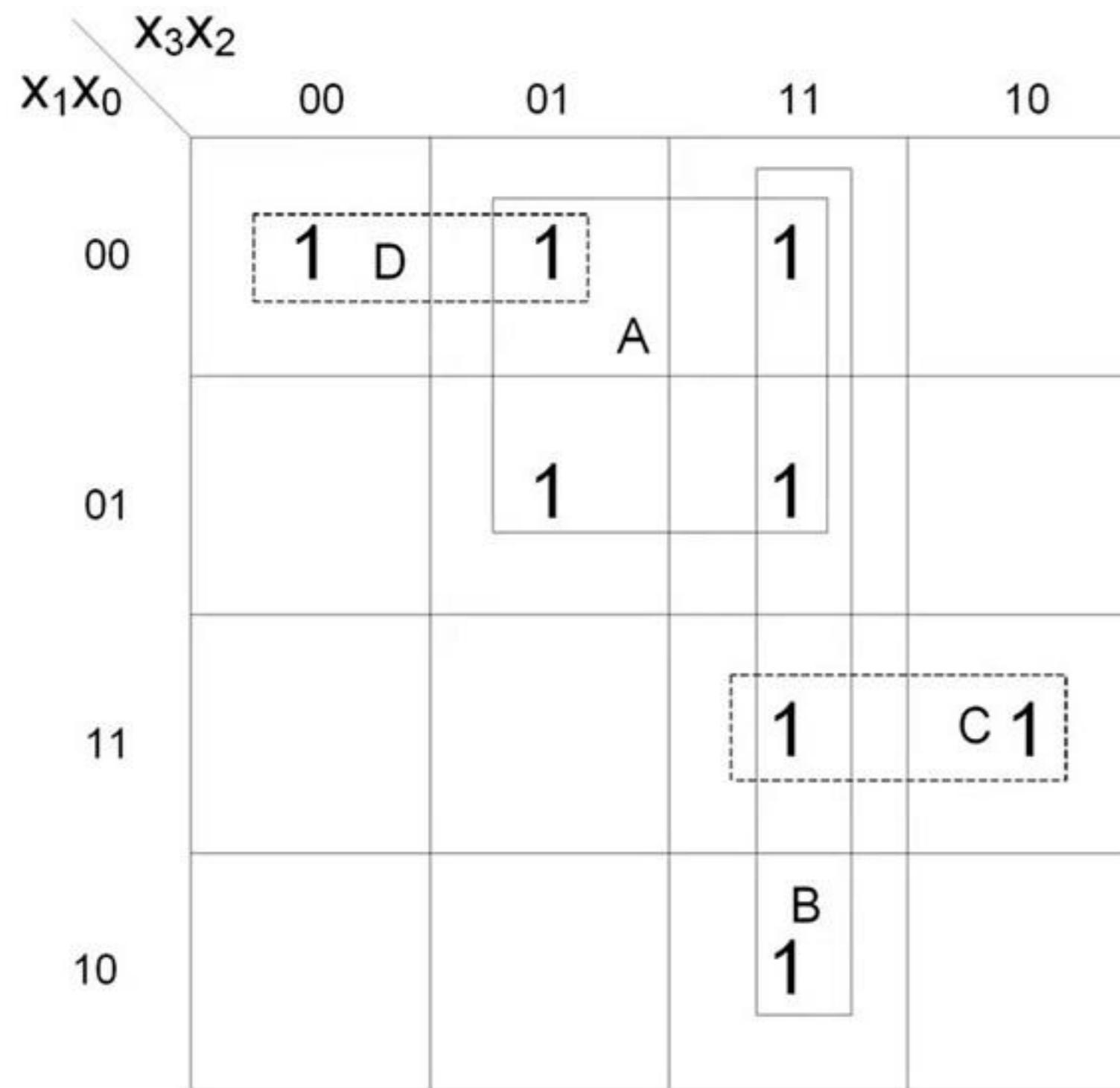
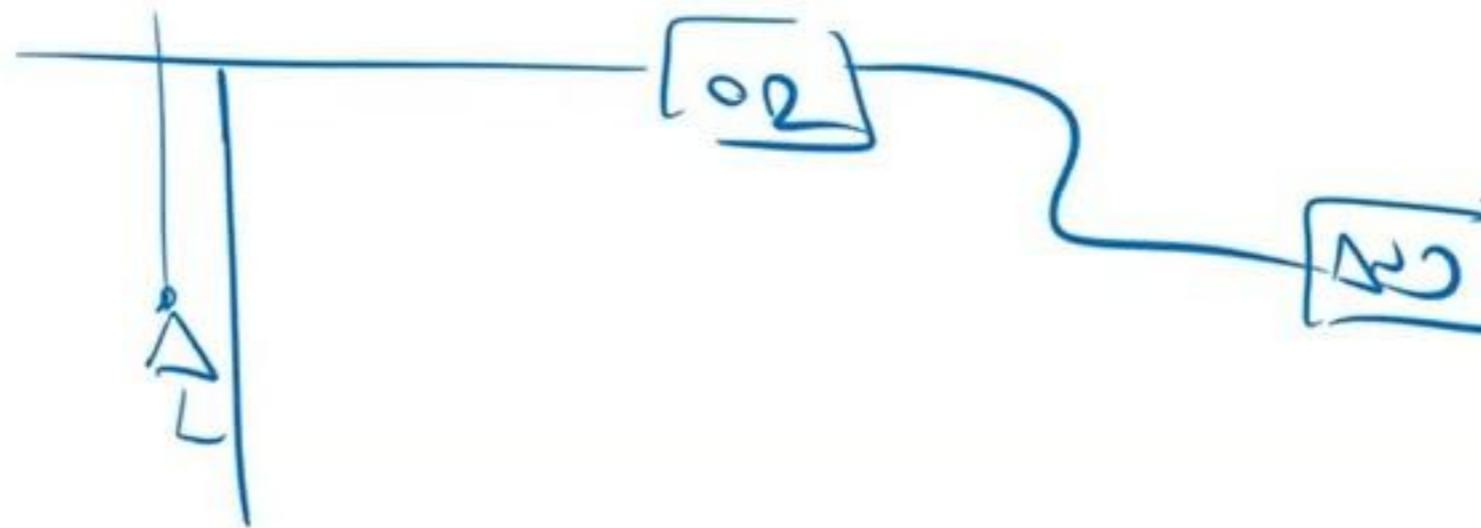
Mappa di Karnaugh per  $\overline{z}$

	X <sub>3</sub>	X <sub>2</sub>	X <sub>1</sub>	X <sub>0</sub>
A	-	1	0	-
B	1	1	-	-
C	1	-	1	1
D	0	-	0	0

5 costi porte

$$\overline{z} = \overline{x}_1 \cdot x_2 + x_3 \cdot x_2 + x_3 \cdot x_1 \cdot x_0 + \overline{x}_3 \cdot \overline{x}_1 \cdot \overline{x}_0$$

# Esercizio (cont.)



$$\bar{z} = \overline{x_1} \cdot x_2 + x_3 \cdot x_2 + x_3 \cdot x_1 \cdot x_0 + \overline{x_3} \cdot \overline{x_1} \cdot \overline{x_0}$$

$$\begin{aligned}
 z &= \overline{\overline{x_1} \cdot x_2} \cdot \overline{x_3 \cdot x_2} \cdot \overline{x_3 \cdot x_1 \cdot x_0} \cdot \overline{x_3 \cdot x_1 \cdot x_0} \\
 &= (\overline{x_1} \cdot x_2) \cdot (\overline{x_3 \cdot x_2}) \cdot (\overline{x_3 \cdot x_1 \cdot x_0}) \cdot (\overline{x_3 \cdot x_1 \cdot x_0}) \\
 &= (x_1 + \overline{x_2}) \cdot (\overline{x_3} + \overline{x_2}) \cdot (\overline{x_3} + \overline{x_1} + \overline{x_0}) \cdot (x_3 + x_1 + x_0)
 \end{aligned}$$

# Porte logiche universali

- NAND e NOR sono porte logiche universali
  - Posso realizzare AND, OR, NOT con sole porta NAND o sole porta NOR

	Rel. algebrica	Porta	Realizzazione a NAND	Realizzazione a NOR
NOT	$x = x \cdot x \Rightarrow \bar{x} = \overline{x \cdot x}$ $\rightarrow x = x + x \Rightarrow \bar{x} = \overline{x + x}$	$x \rightarrow \text{NOT} \rightarrow z$	$x \rightarrow \text{NAND} \rightarrow z$	$x \rightarrow \text{NOR} \rightarrow z$
AND	$x \cdot y = \overline{(\overline{x} \cdot \overline{y})}$ $\rightarrow x \cdot y = \overline{\overline{x} + \overline{y}}$	$x \rightarrow \text{AND} \rightarrow z$	$x \rightarrow \text{NAND} \rightarrow \text{NAND} \rightarrow z$	$x \rightarrow \text{NOR} \rightarrow \text{NOR} \rightarrow z$
OR	$x + y = \overline{\overline{x} \cdot \overline{y}}$ $\rightarrow x + y = \overline{(\overline{x} + \overline{y})}$	$x \rightarrow \text{OR} \rightarrow z$	$x \rightarrow \text{NAND} \rightarrow \text{NAND} \rightarrow z$	$x \rightarrow \text{NOR} \rightarrow \text{NOR} \rightarrow z$

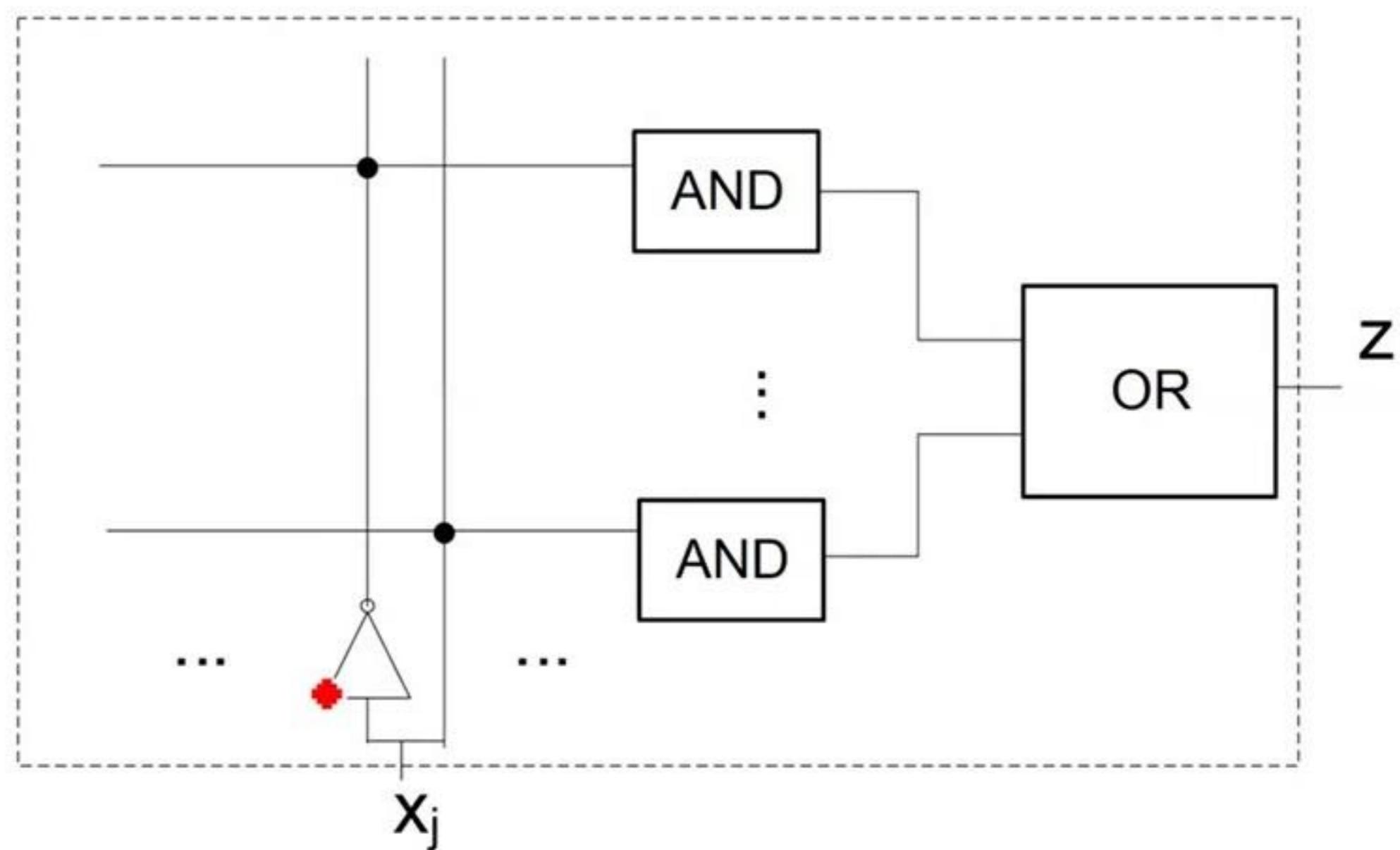
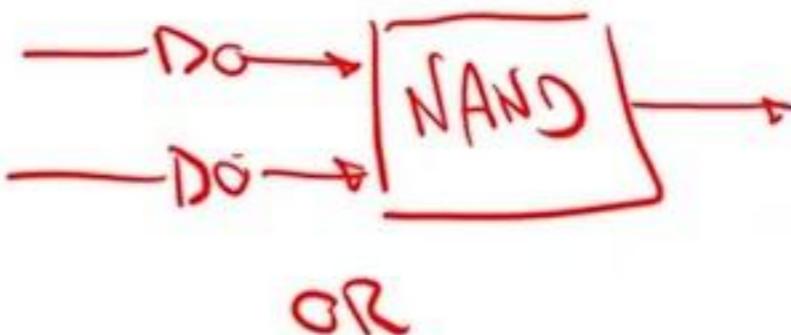
# Porte logiche universali

- Visto che **qualunque legge combinatoria** puo' essere sintetizzata con porte AND, OR, NOT
- Qualunque legge combinatoria puo' essere sintetizzata
  - **Soltanto** con porte NAND
  - **Soltanto** con porte NOR

# Sintesi a porte NAND

F Z

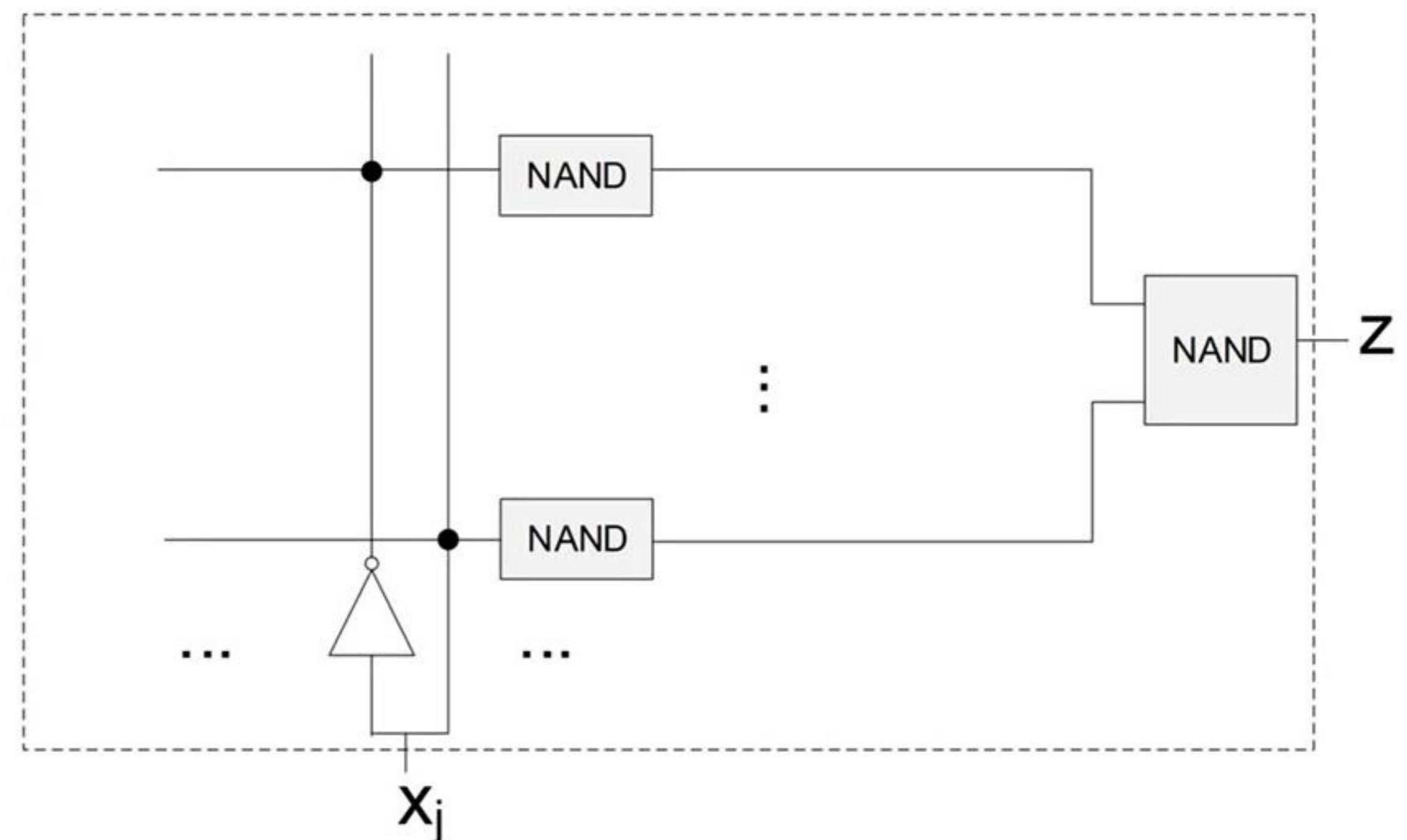
- Si parte da un circuito **in forma SP**
- Sostituisco la porta OR con il suo equivalente a NAND
- Sostituisco ciascuna AND con il suo equivalente a NAND
- I NOT sugli ingressi li posso lasciar stare, tanto **non fanno parte della sintesi** (sono gratis)



# Sintesi a porte NAND (cont.)

- Elimino le coppie di NOT in cascata
- Ottengo una rete a sole porte NAND
- A due livelli di logica

2LL



# SP

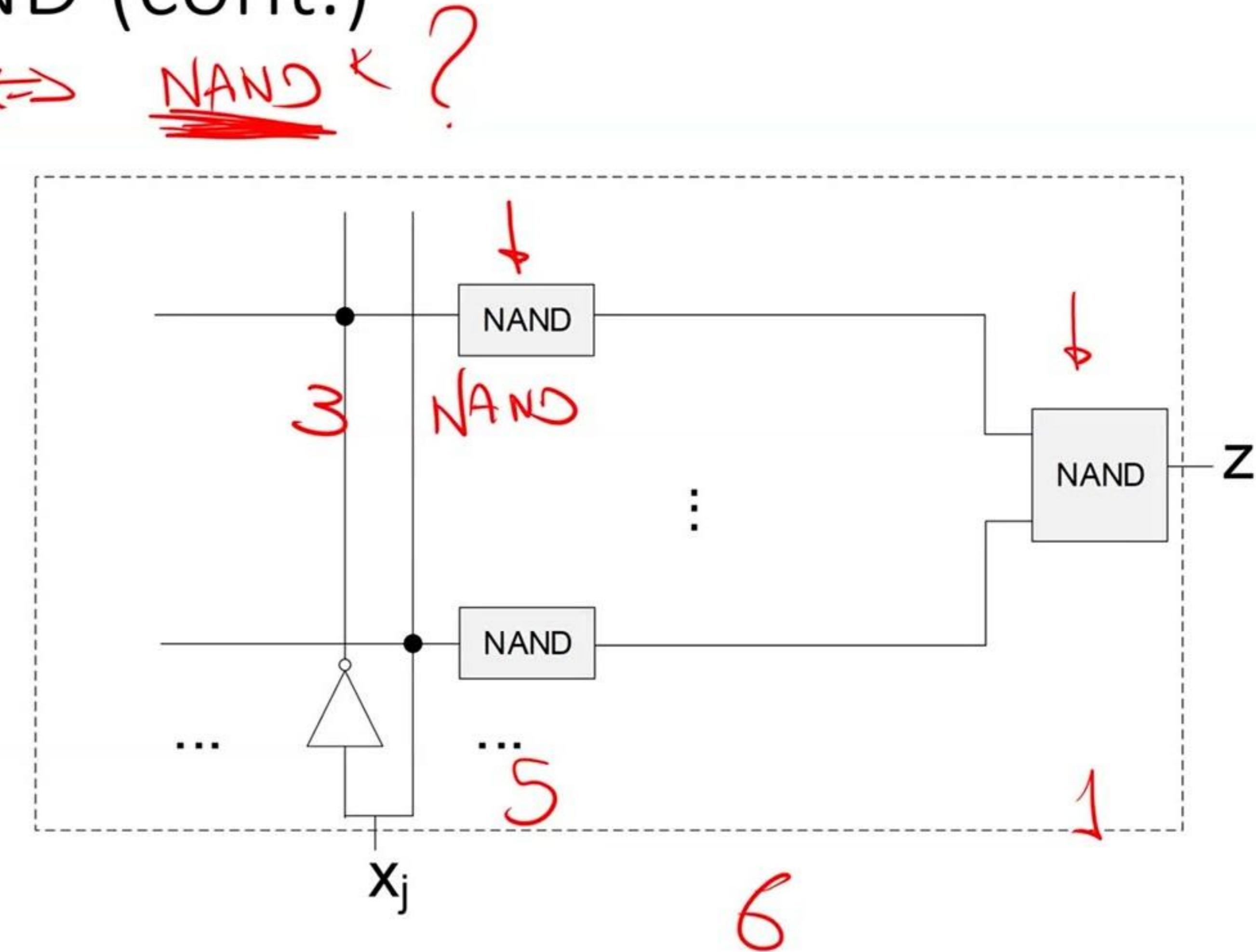
## Sintesi a porte NAND (cont.)

- Dal punto di vista algebrico

$$\cancel{SP} \rightarrow z = P_1 + P_2 + \dots + P_K$$

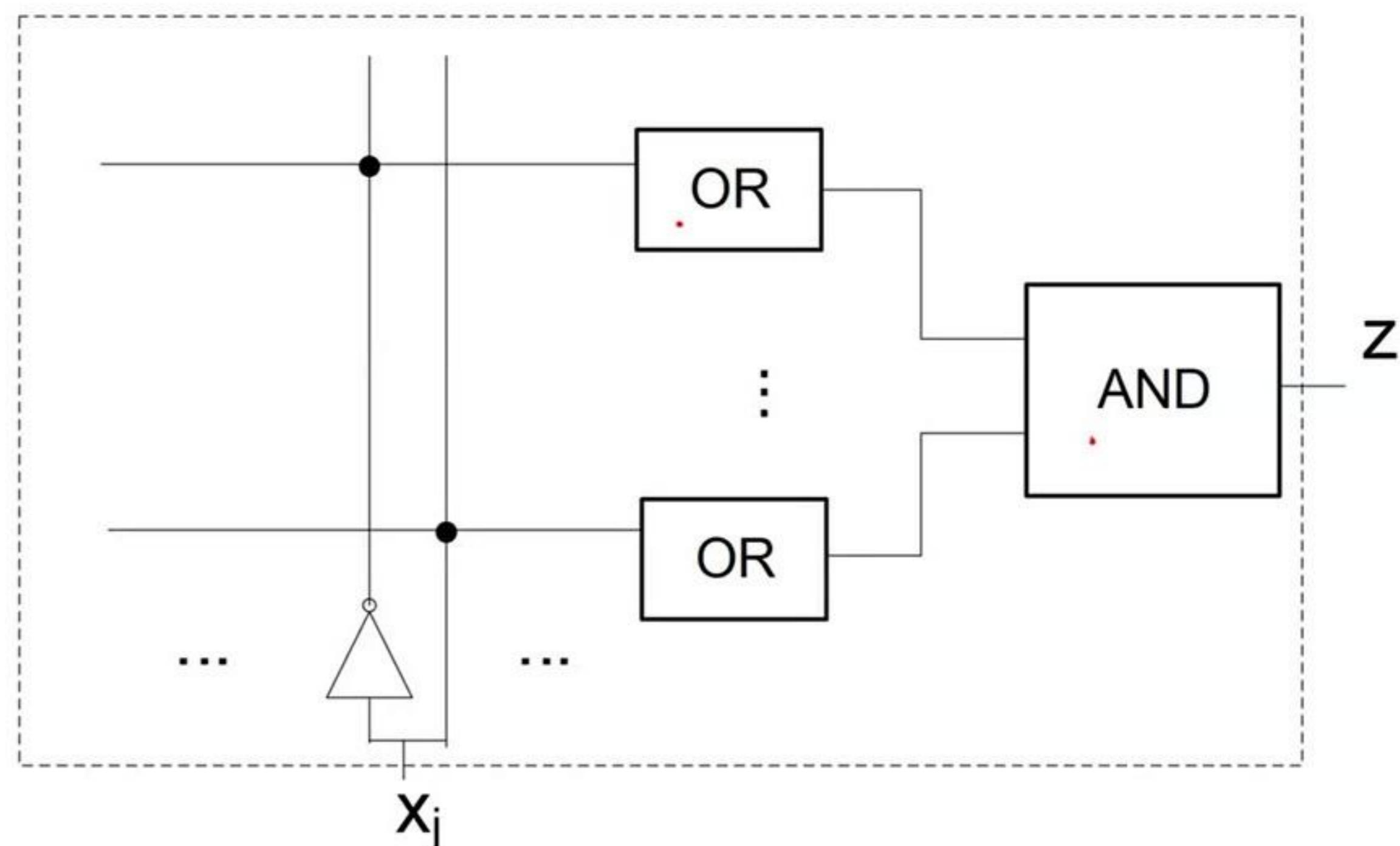
$$2c \quad = \overline{\overline{P_1 + P_2 + \dots + P_K}}$$

$$\cancel{DNF} \quad = (\overline{\overline{P_1}} \cdot \overline{\overline{P_2}} \cdot \dots \cdot \overline{\overline{P_K}})$$



# Sintesi a porte NOR

- Si parte da un circuito **in forma PS**
- Sostituisco la porta AND con il suo equivalente a NOR
- Sostituisco la porta OR con il suo equivalente a NOR
- Non sto a toccare i NOT sugli ingressi

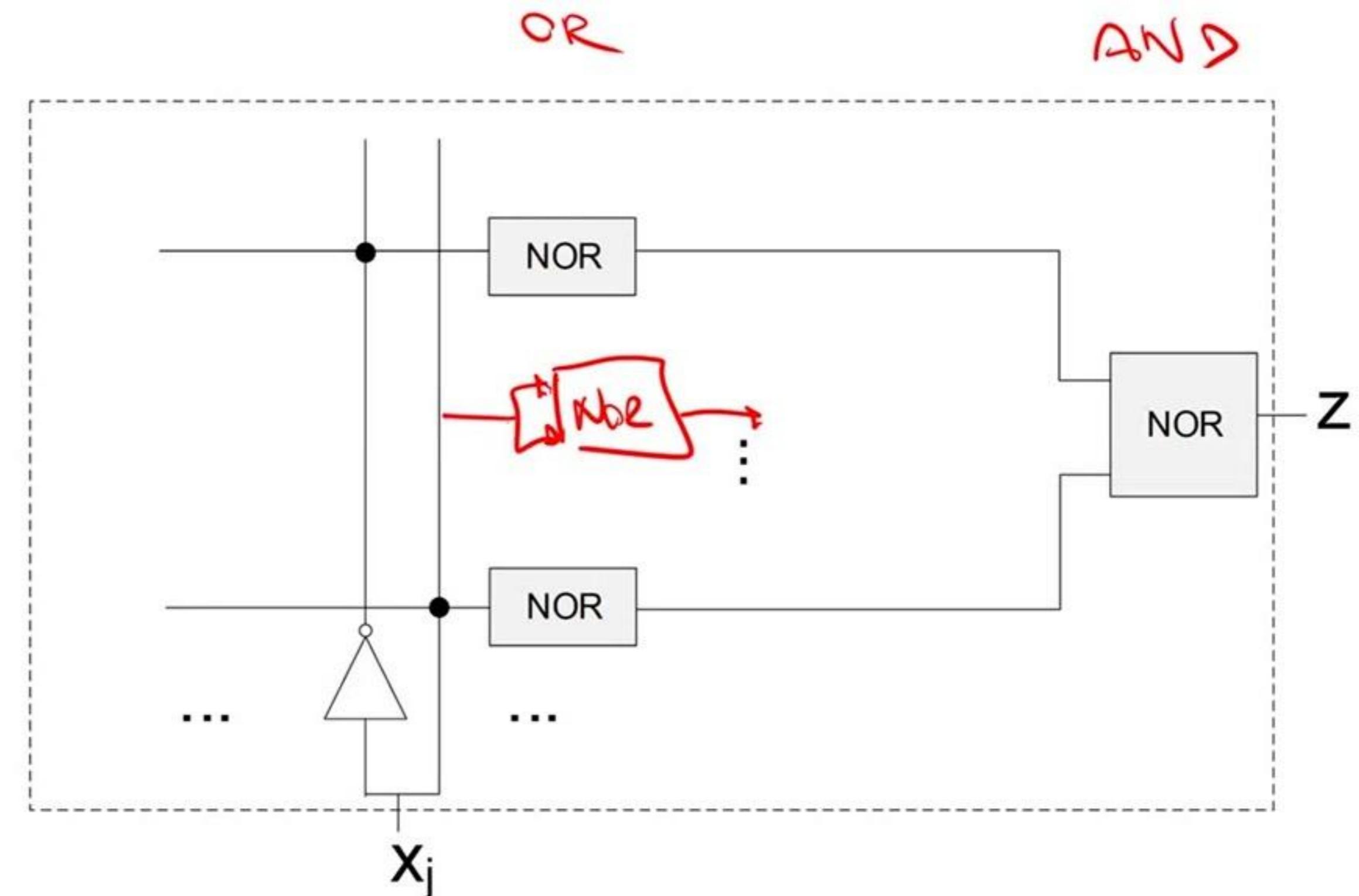


# Sintesi a porte NOR (cont.)

PS

NCR

- Elimino le coppie di NOT in cascata
  - Ottengo una rete a sole porte NOR
  - A due livelli di logica

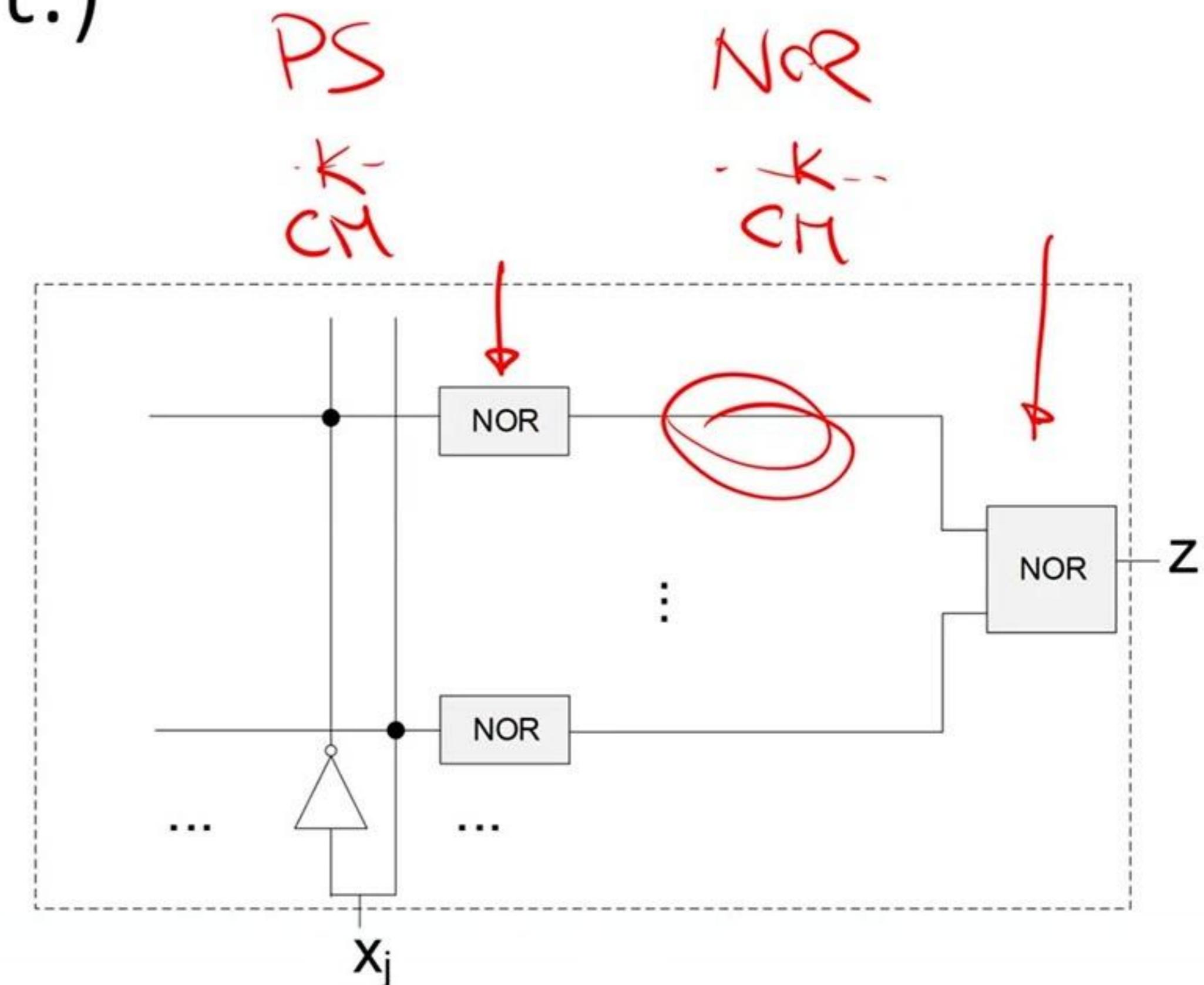


SP

## Sintesi a porte NOR (cont.)

- Dal punto di vista algebrico

$$\begin{aligned} PS &\rightarrow z = S_1 \cdot S_2 \cdot \dots \cdot S_K \\ &= \overline{S_1 \cdot S_2 \cdot \dots \cdot S_K} \\ &= (\overline{\overline{S_1}} + \overline{\overline{S_2}} + \dots + \overline{\overline{S_K}}) \\ &\quad \text{DNI} \downarrow \quad \parallel \end{aligned}$$

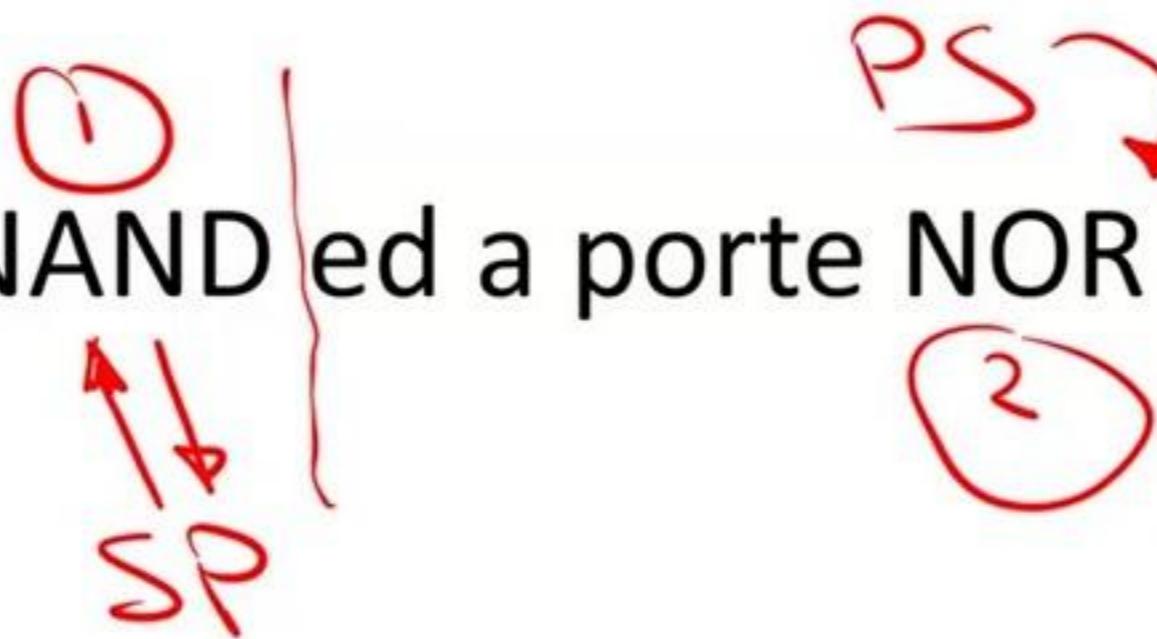


# Esercizio

→ GS1c minime

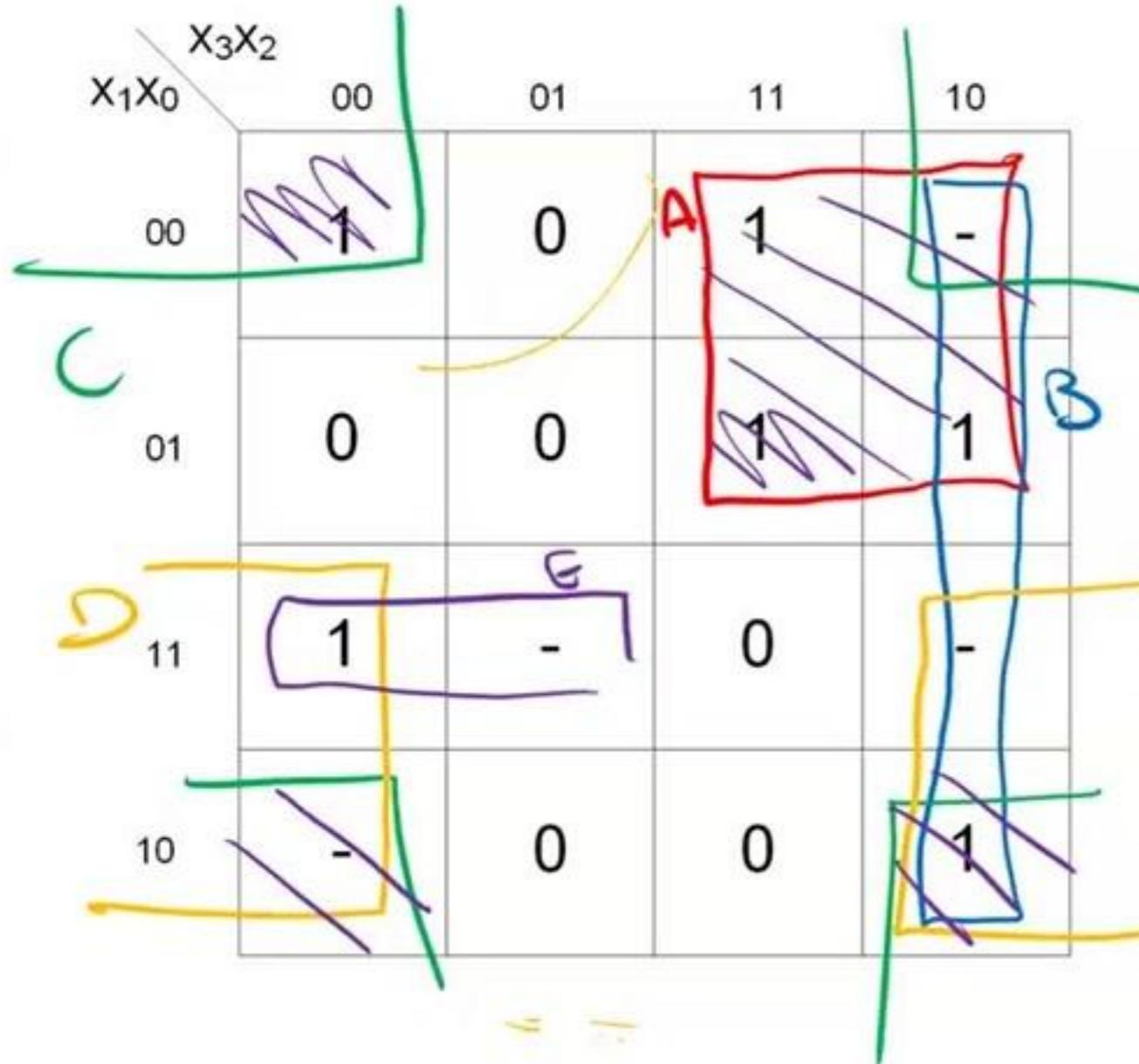
- Sintetizzare la rete a porte NAND ed a porte NOR

$x_1x_0$	$x_3x_2$	00	01	11	10
00	1	0	1	-	
01	0	0	1	1	
11	1	-	0	-	
10	-	0	0	1	



SP CM

# Sintesi a porte NAND



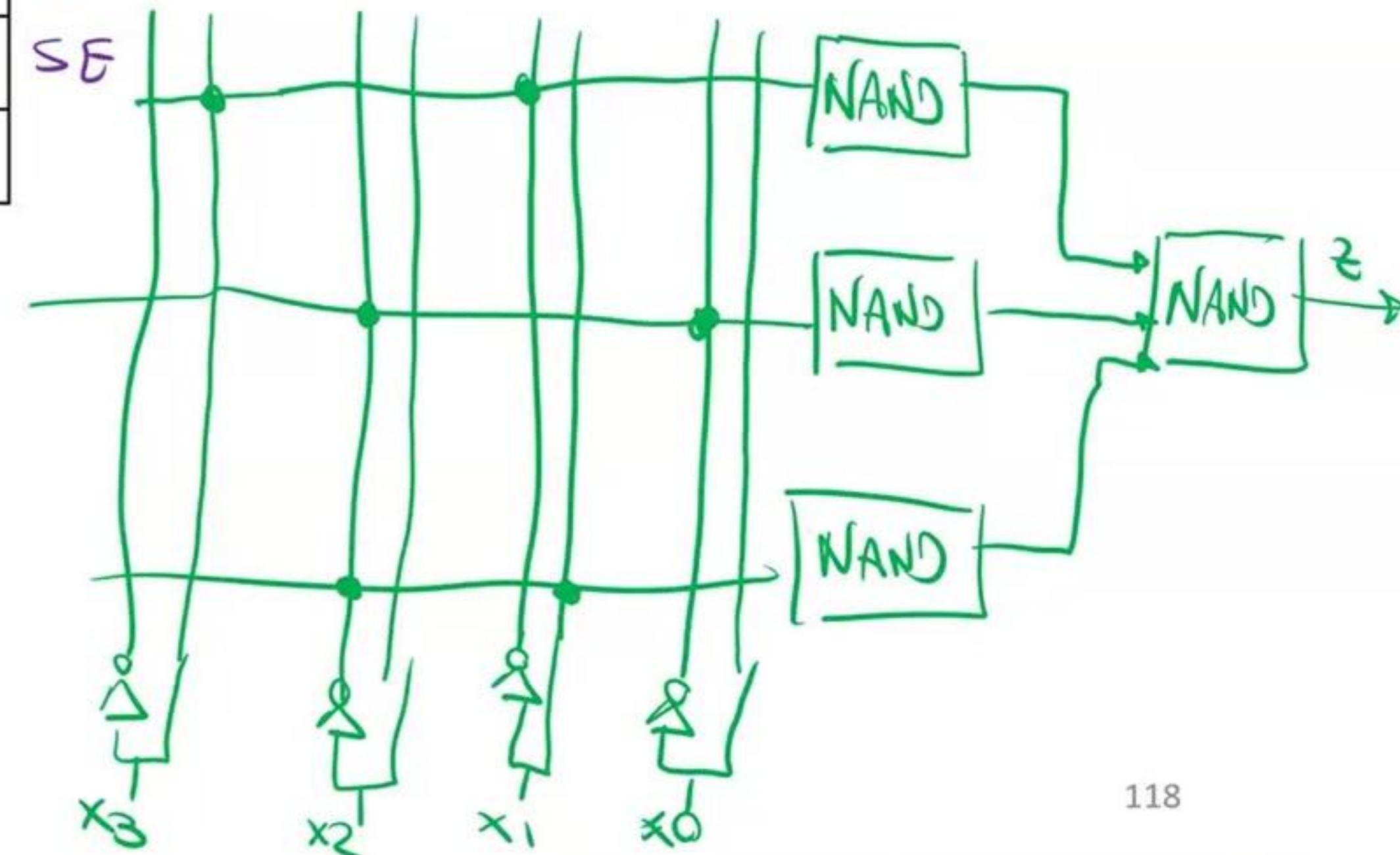
	X <sub>3</sub>	X <sub>2</sub>	X <sub>1</sub>	X <sub>0</sub>
A	1	-	0	-
B	1	0	-	-
C	-	0	-	0
D	-	0	1	-
E	0	-	1	1

AC  
D  
ACE

$$Z = \overline{x_3 \cdot x_1} + \overline{x_2 \cdot x_0} + \overline{x_2 \cdot x_1}$$

$$= (\overline{x_3 \cdot x_1}) \cdot (\overline{x_2 \cdot x_0}) \cdot (\overline{x_2 \cdot x_1})$$

4  
CSS  
ass.elim.  
ESS  
SG  
SE



# Sintesi a porte NOR

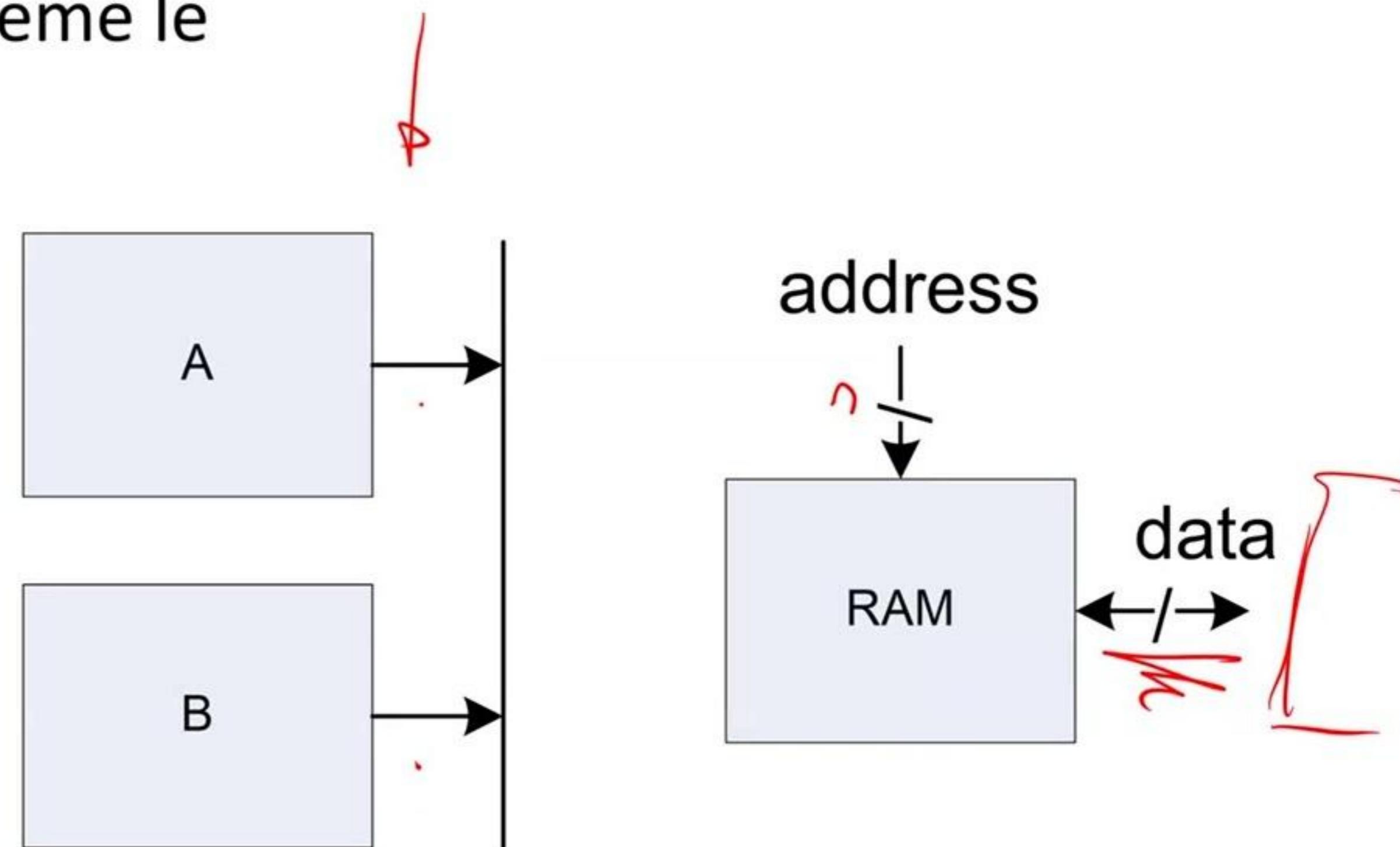
$x_1x_0$	$x_3x_2$	00	01	11	10
00		1	0	1	-
01		0	0	1	1
11		1	-	0	-
10		-	0	0	1

$x_1x_0$	$x_3x_2$	00	01	11	10
00					
01					
11					
10					

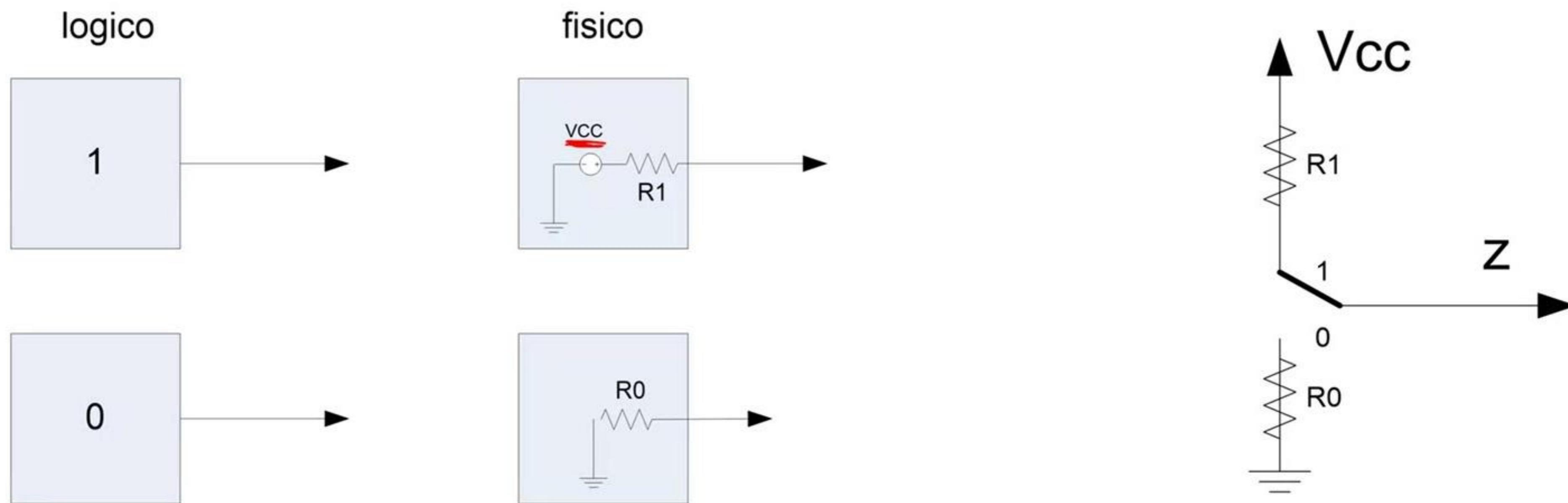
	X3	X2	X1	X0
0				
1				
2				
3				
4				
5				
6				
7				

# Porte tri-state

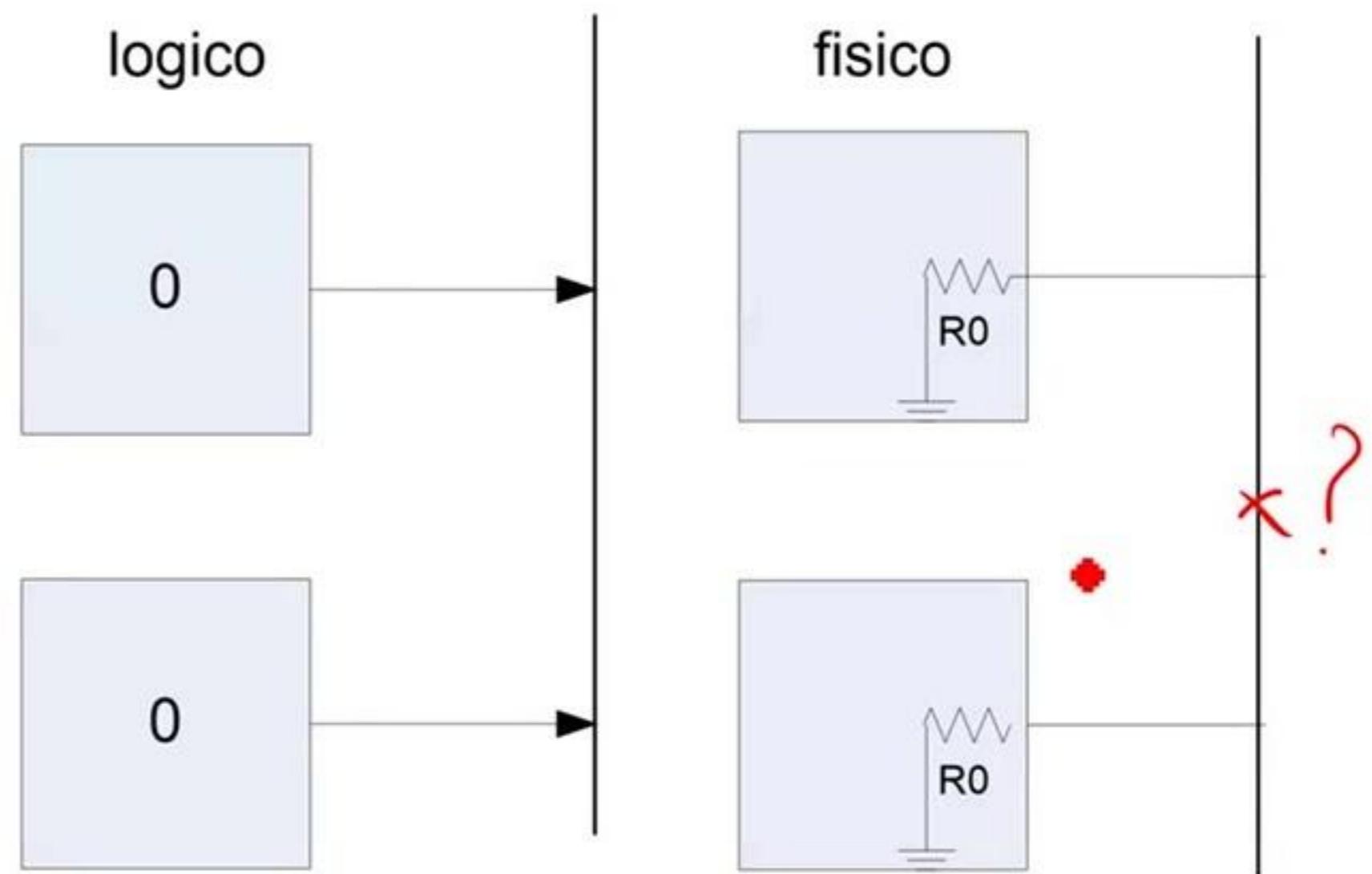
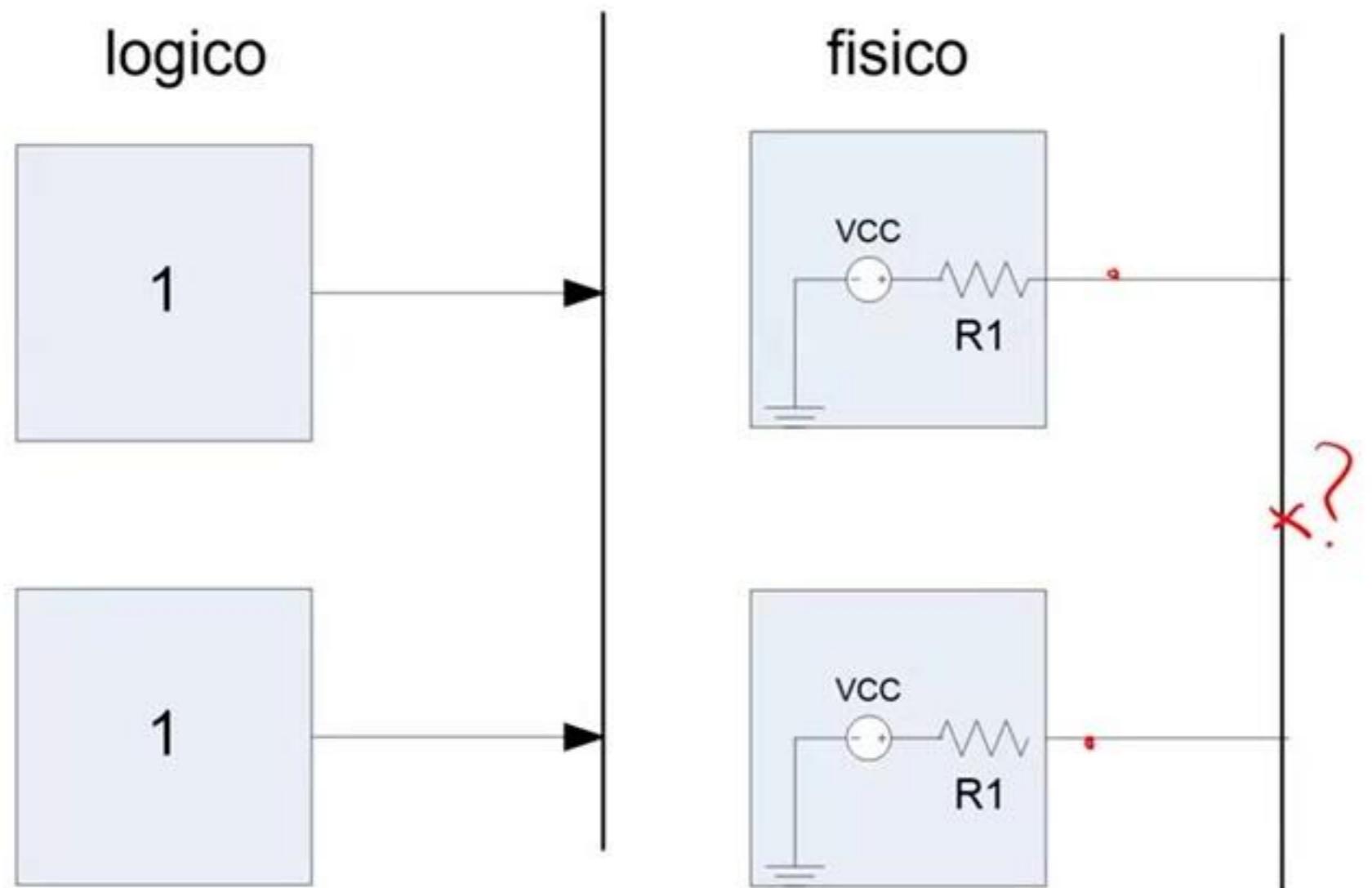
- fa comodo poter connettere insieme le **uscite** delle reti
  - Bus condivisi
  - Linee di ingresso-uscita



# Modello elettrico dell'uscita di una rete



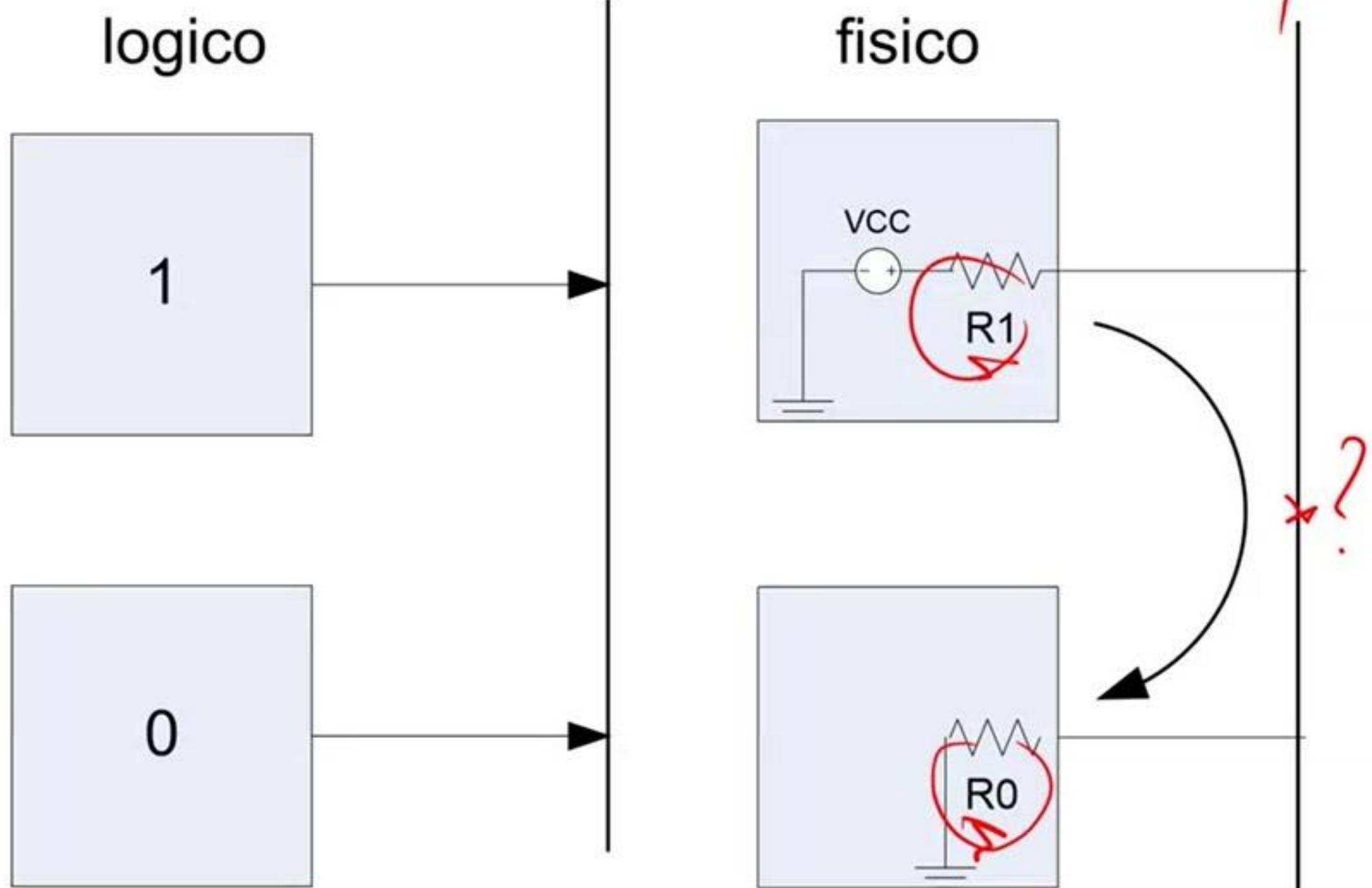
# Problemi elettrici e logici



# Problemi elettrici e logici

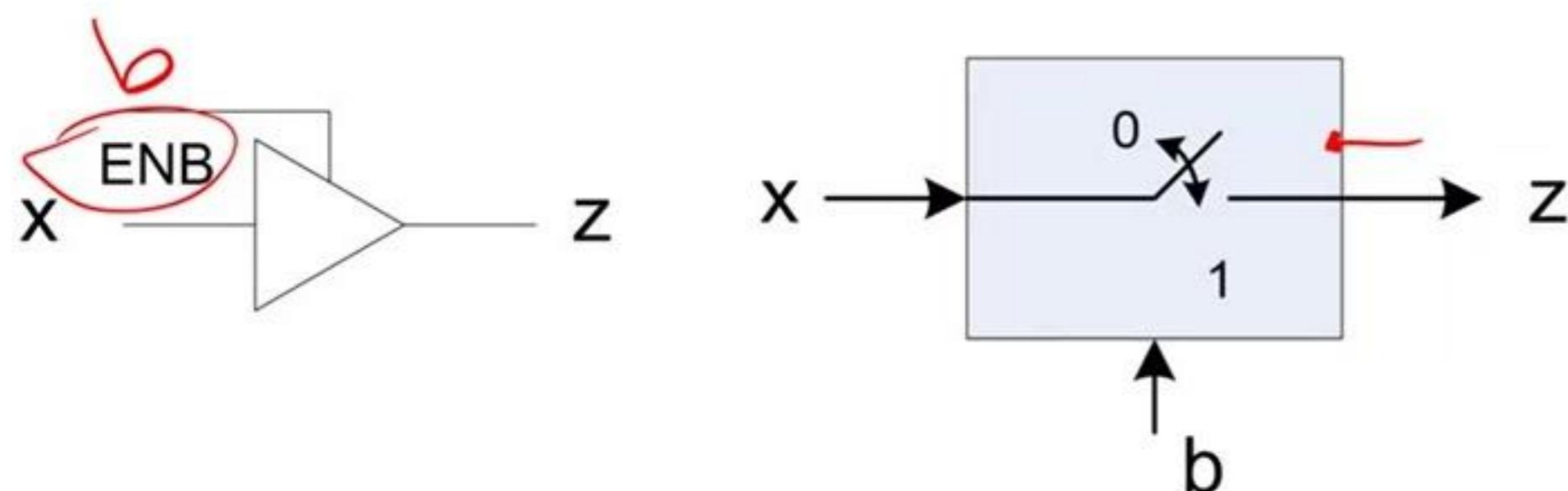
- Scorre corrente
  - Anche molta
  - Circuiti si bruciano
- Quale e' il valore logico sulla linea?

||



# Porte tri-state

- servono porte **capaci di disconnettere fisicamente** un'uscita da una linea condivisa.
- Tali porte prendono il nome di **porte tri-state**



b	x	z
1	0	0
1	1	1
0	-	HiZ

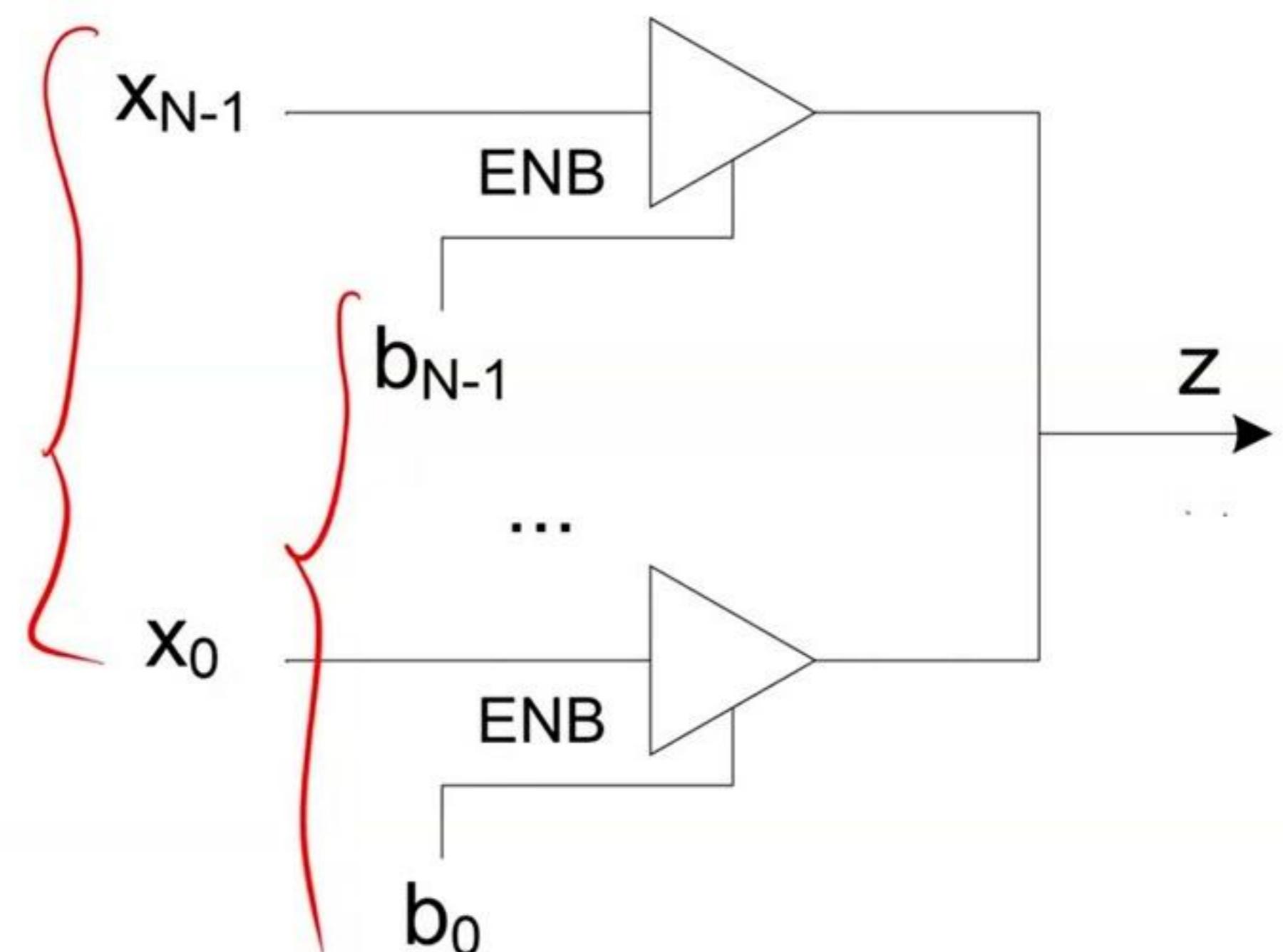
- **b=1**: la porta si comporta da elemento neutro
- **b=0**: la porta e' in **alta impedenza**
  - L'uscita e' disconnessa

N-1 HiZ

# Esempio: multiplexer decodificato

- Se tutte le variabili di comando valgono 0, l'uscita è in **alta impedenza**
- Altrimenti, **una** sola variabile di comando deve essere ad 1
  - l'uscita insegue l'ingresso corrispondente

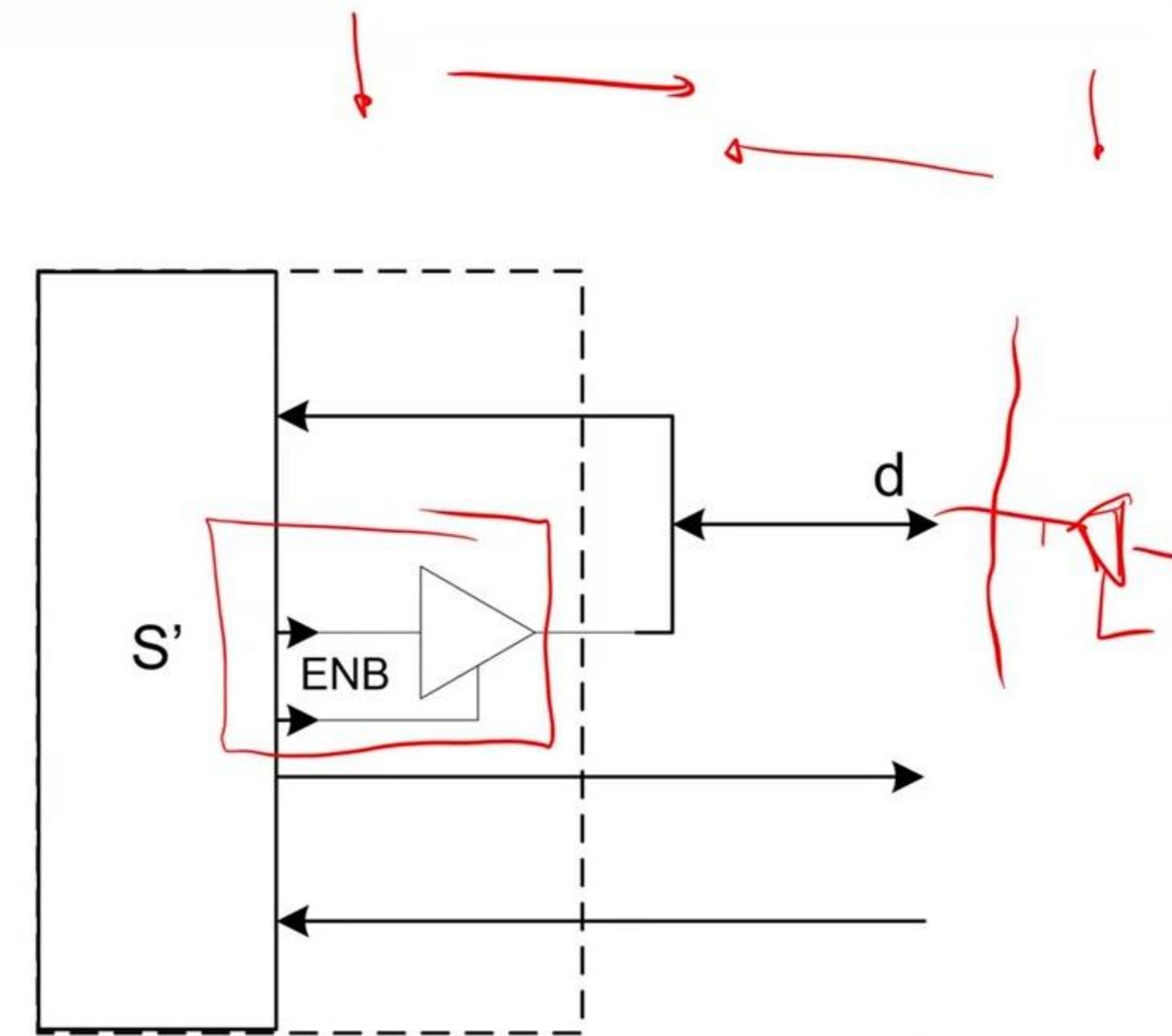
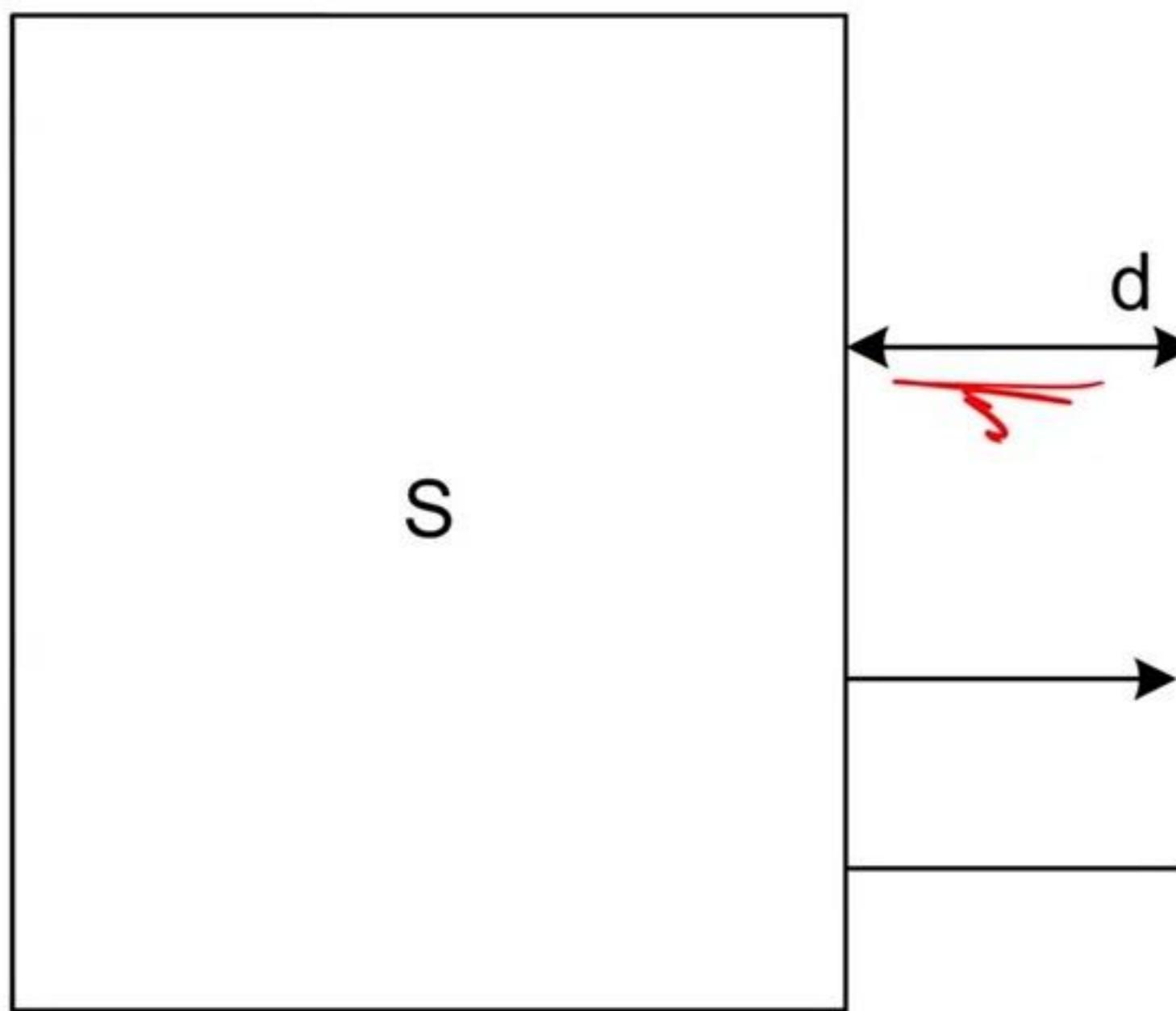
00 - 00 → z + z  
010 - 00 → z  
↑



# Esempio: linea di ingresso/uscita

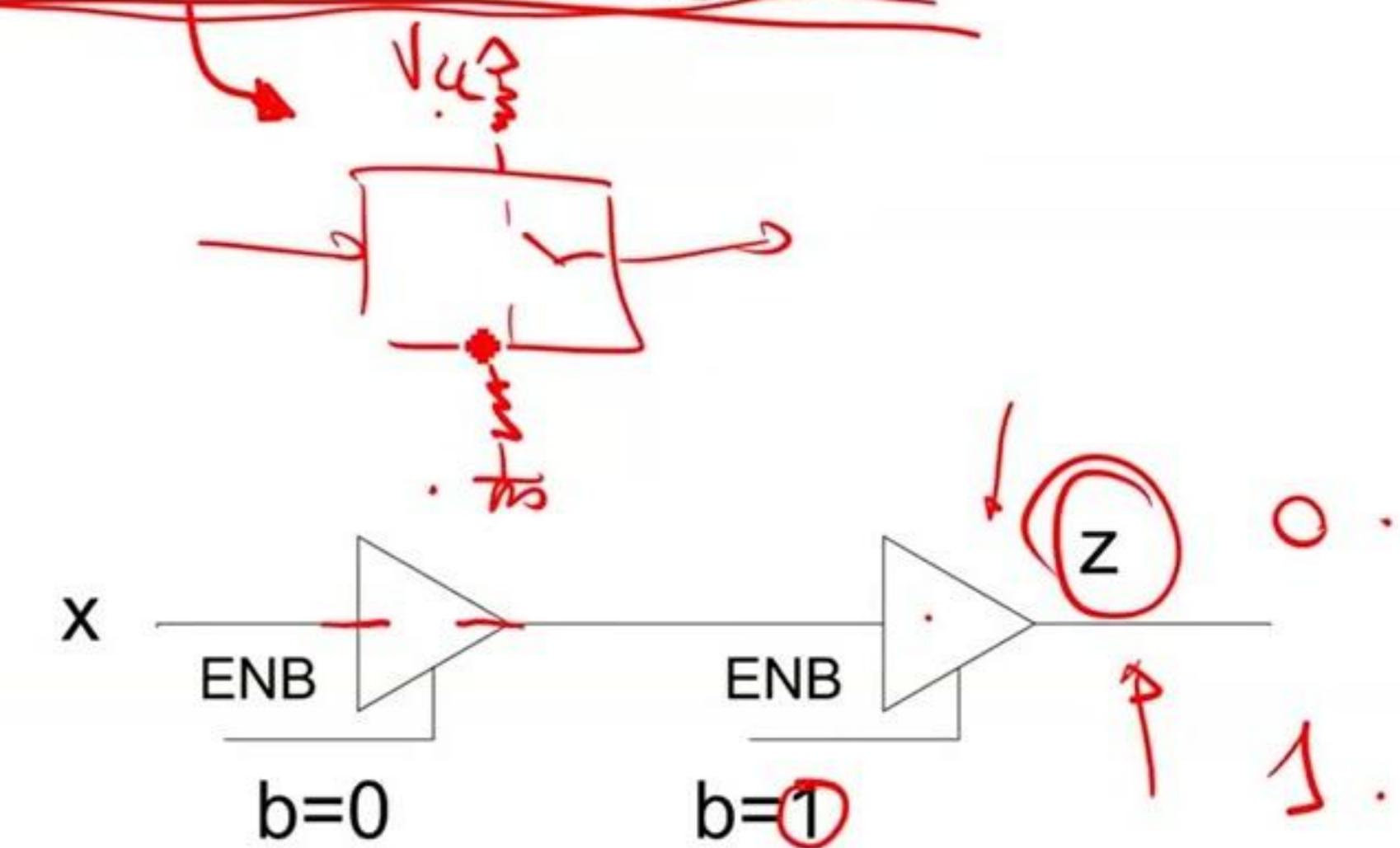
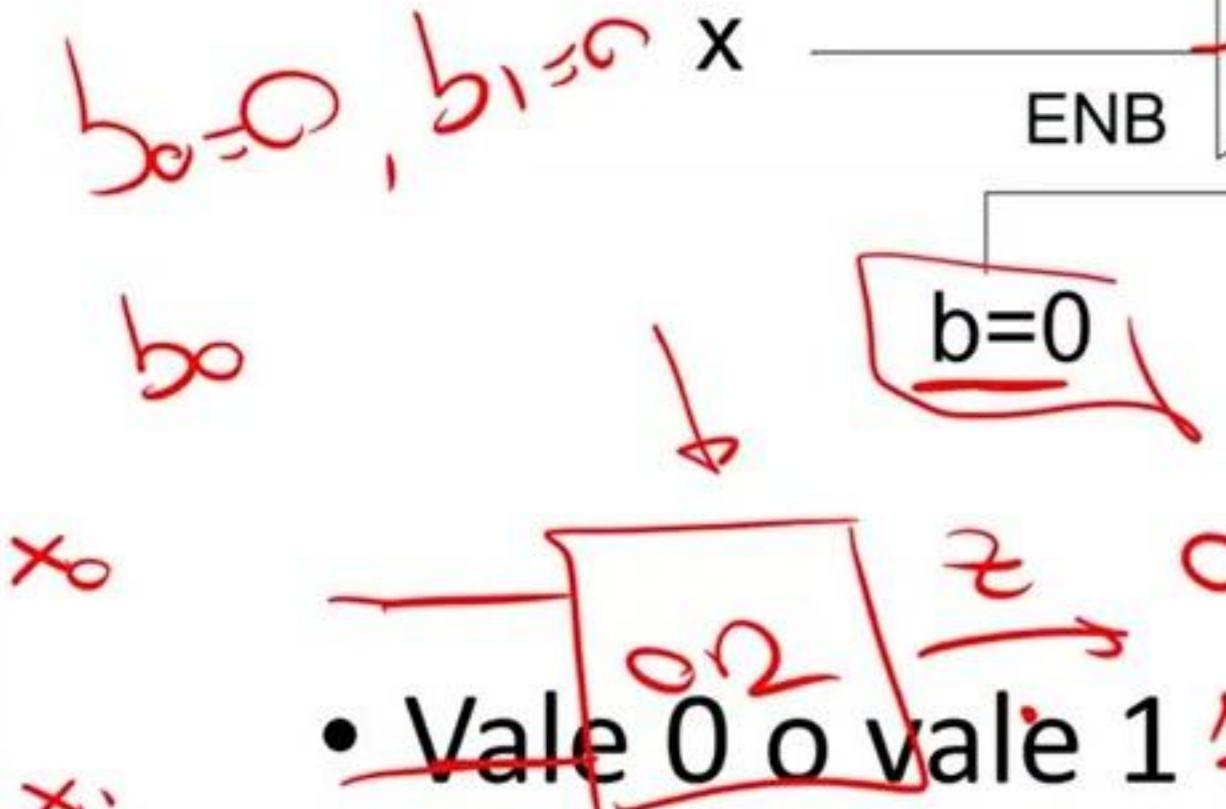
- Forchetta

- $b=0$ : linee di ingresso
- $b=1$ : linee di uscita



# L'alta impedenza non e' un valore logico

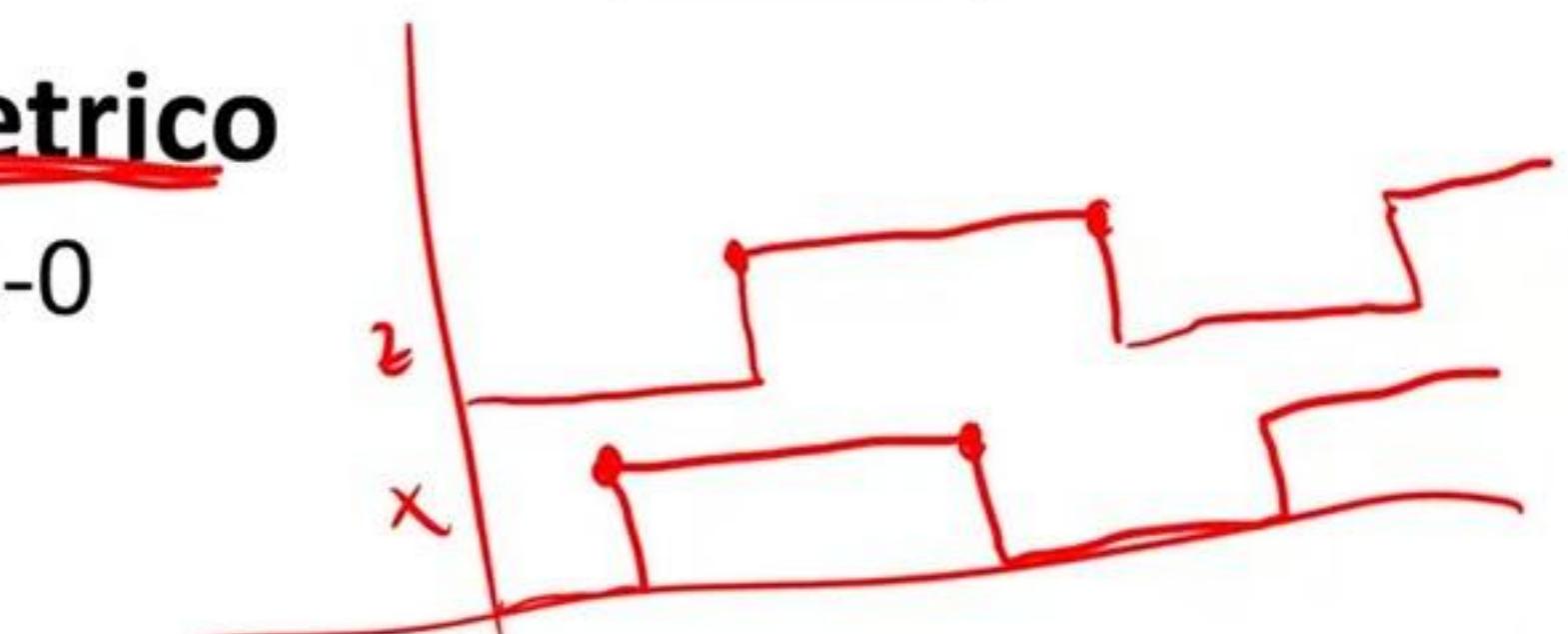
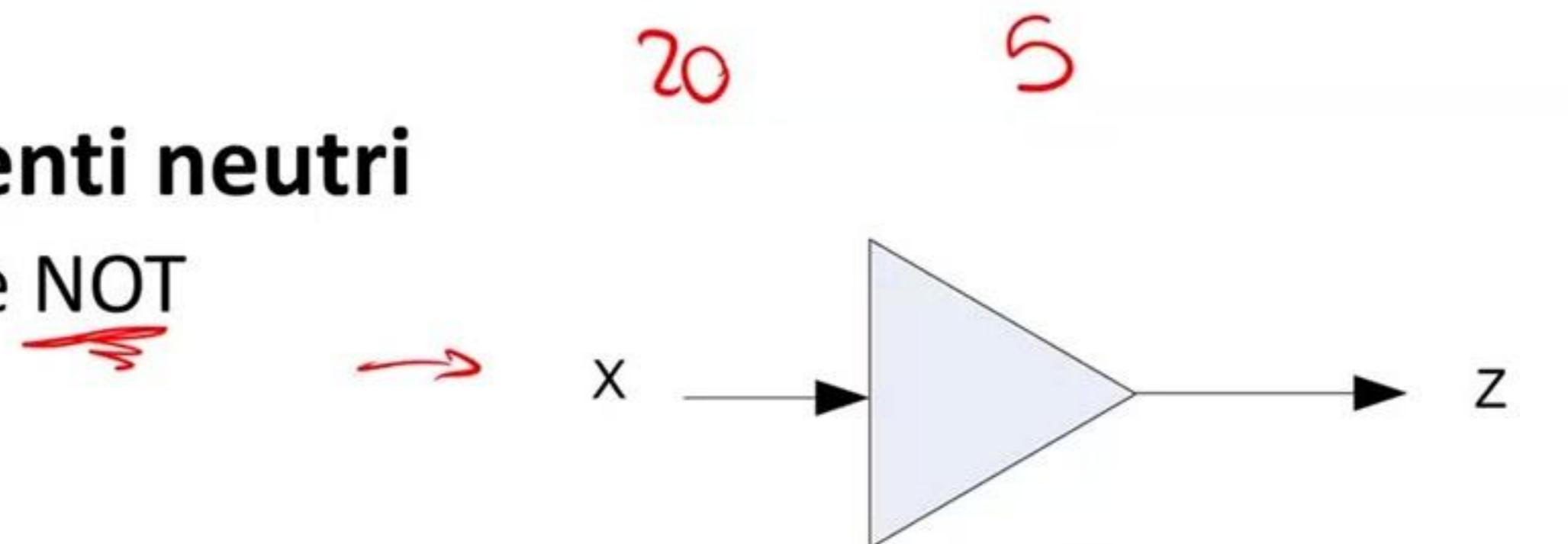
- Quanto vale z nelle due reti qui sotto?



- Dipende dalla tecnologia sottostante
- Le uscite degli stadi finali sono connesse alla tensione Vcc o a massa.

# Circuiti di ritardo e formatori di impulsi

- I circuiti di ritardo si fanno con **gli elementi neutri**
  - O, piu' spesso, con un numero pari di porte NOT
- Hanno ritardo **simmetrico**
  - Identico sulle transizioni 0-1 e 1-0
- Vorrei poter realizzare circuiti con ritardo **asimmetrico**
  - Grande sulle transizioni 0-1, **piccolo** sulle transizioni 1-0
  - O viceversa

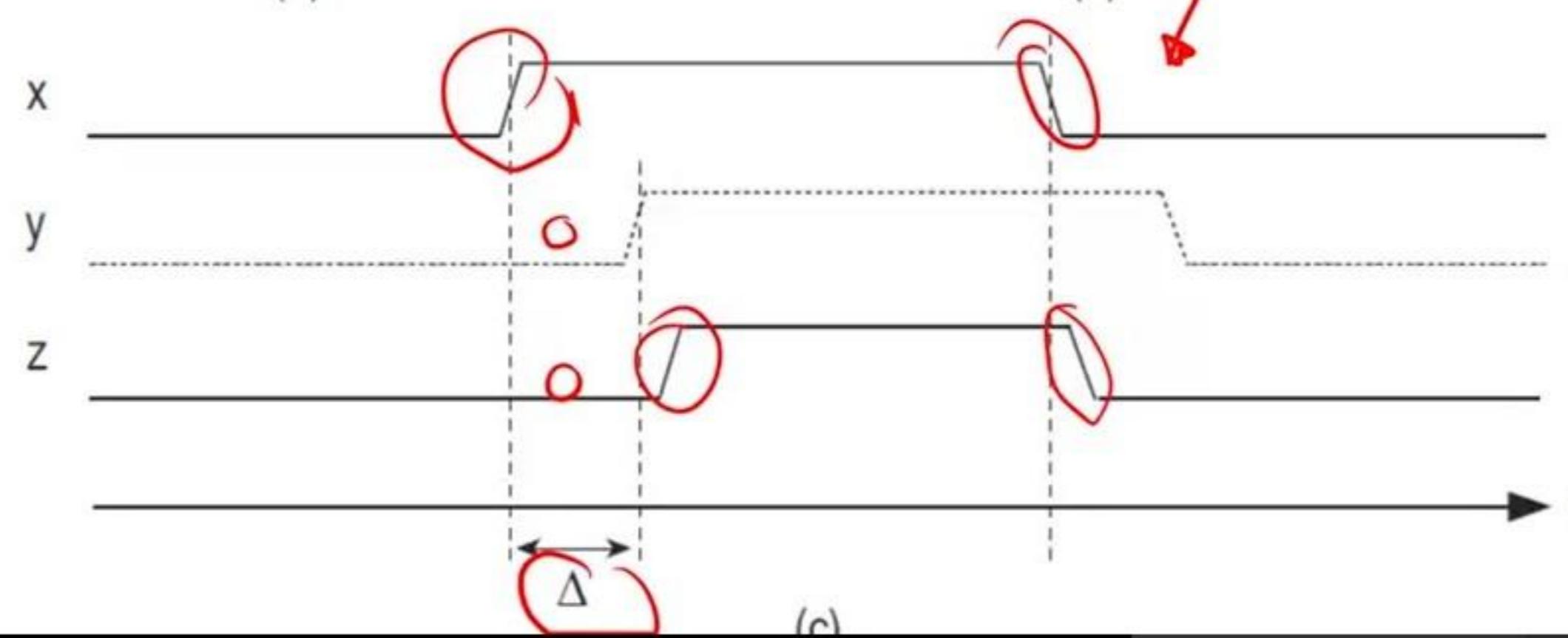
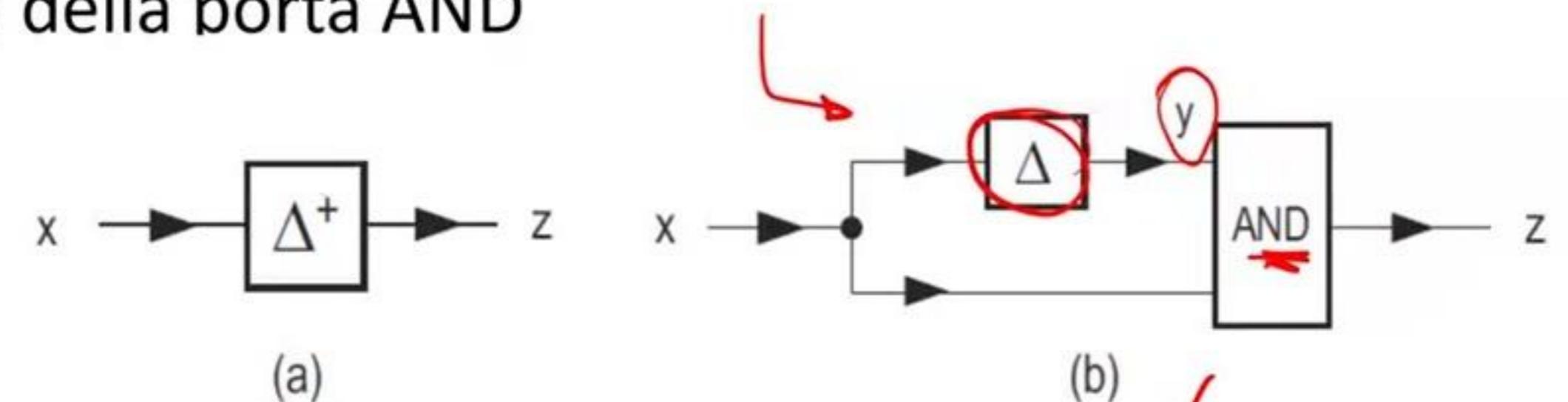


# Circuito di ritardo sul fronte di salita

- Si indica con  $\Delta^+$

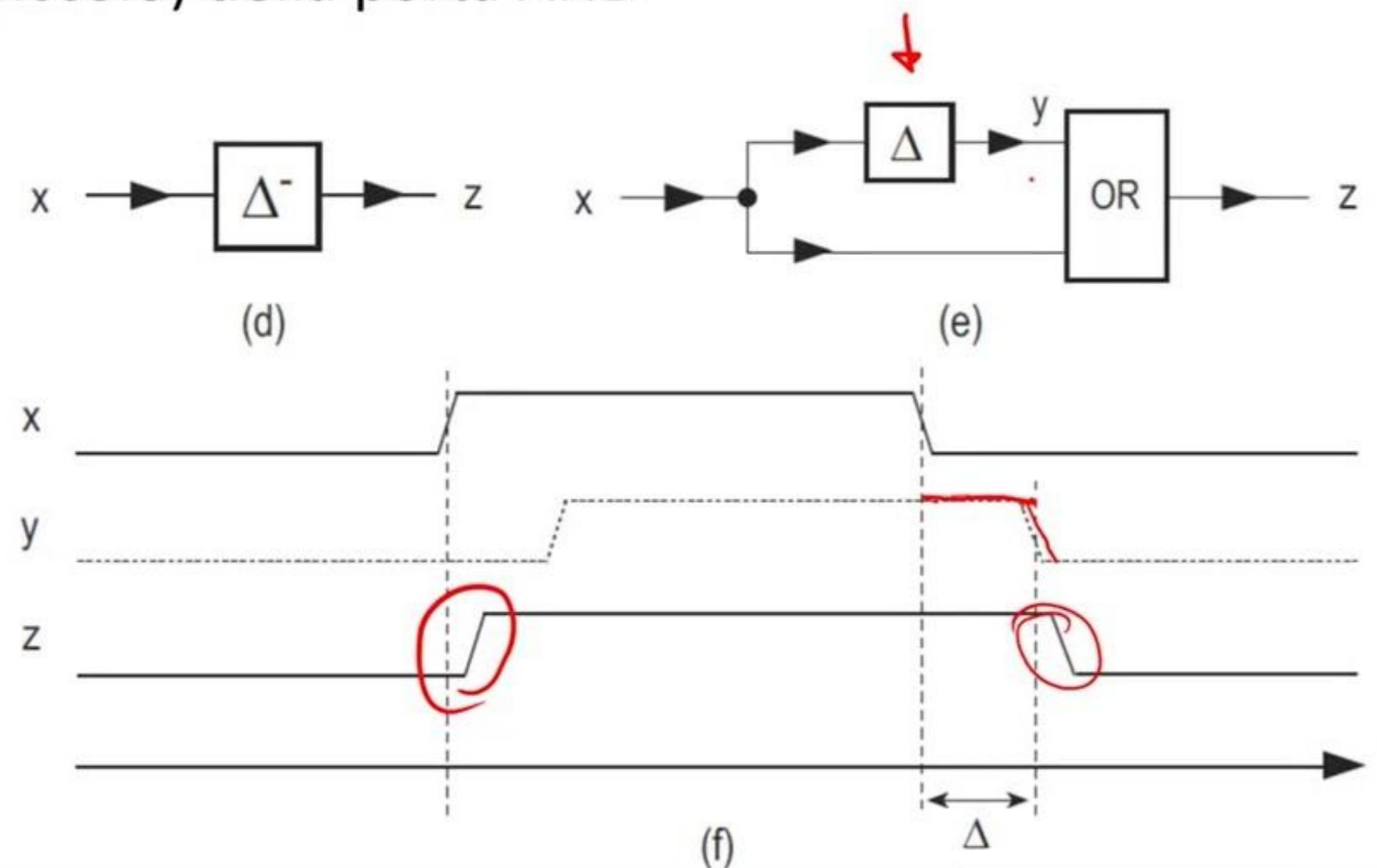
$\Delta^+$

- Si realizza con una AND
- Transizione 1-0, il **primo** ingresso che va a 0 (x) porta a zero l'uscita. Ritardo **piccolo** della porta AND
- Transizione 0-1, il **secondo** ingresso che va a 1 (y) porta a 1 l'uscita. Ritardo (**grande**) del buffer, più quello (piccolo) della porta AND



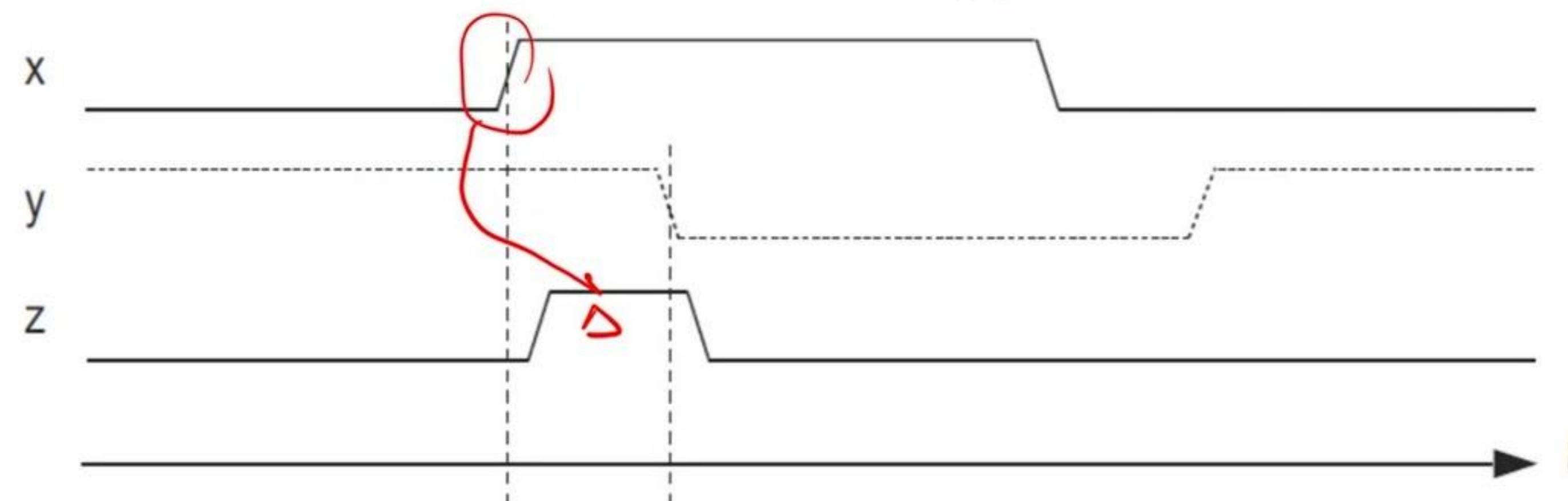
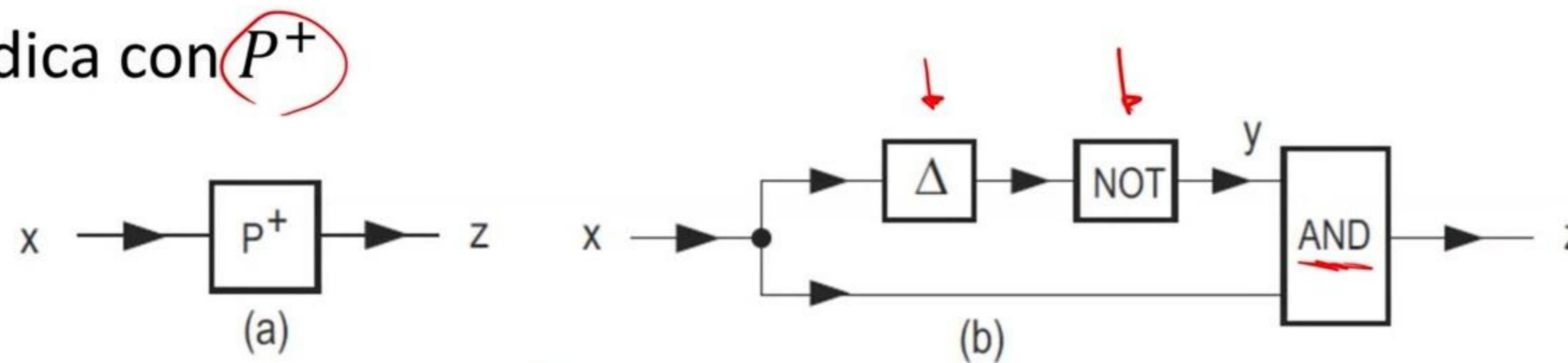
# Circuito di ritardo sul fronte di discesa

- Si indica con  $\Delta^-$ 
  - Si realizza con una OR
  - Transizione 0-1, il **primo** ingresso che va a 1 (y) porta a 1 l'uscita. Ritardo (**piccolo**) della porta OR
  - Transizione 1-0, il **secondo** ingresso che va a zero (x) porta a zero l'uscita. Ritardo **grande** del buffer, piu' quello (piccolo) della porta AND



# Formatore di impulsi sul fronte di salita

- Rete combinatoria che genera in uscita **un impulso di durata nota** quando l'ingresso ha un fronte di salita
- Si indica con  $P^+$



# Formatore di impulsi sul fronte di discesa

- Rete combinatoria che genera in uscita **un impulso di durata nota** quando l'ingresso ha un fronte di discesa
- Si indica con  $P^-$

