Componenti combinatori

Reti combinatorie particolari

(5.1.1, 5.3-5.8, 5.10)

Reti logiche per operazioni aritmetiche

Decoder ed encoder

Multiplexer

Dispositivi programmabili: PROM e PLA

Reti combinatorie particolari

- Sono circuiti integrati disponibili in commercio come singolo componente
- Due tipologie
  - Realizzano funzioni tipiche di grande diffusione
    - Funzioni aritmetiche per elaborazioni numeriche
      - Somma-sottrazione, confronto, moltiplicazione
    - Funzioni logiche universali
      - Codifiche e decodifiche, commutatori multipolari
  - La funzionalità può essere programmata dall'utente
    - Grazie a elementi di connettività modificabili dopo la fabbricazione, come i fusibili

Sommatori binari

- Fondamentali per avere **funzioni aritmetiche**
- Esegue l'algoritmo della somma binaria

$$x + y = \sum_{i=0}^{n-1} (x_i + y_i) 2^i = c_n 2^n + \sum_{i=0}^{n-1} s_i 2^i$$

  - Un sommatore tra numeri a *n* cifre binarie è una rete combinatoria a 2*n* ingressi e (*n* + 1) uscite
  - In generale, la somma di **3 cifre binarie** è esprimibile con un **numero binario a 2 cifre**
    - Infatti assume un valore tra 0 e 3
- Un sommatore tra numeri a *n* cifre può essere realizzato con *n* sommatori da 3 bit
  - Sono definiti **full-adder**

Il full adder (1)

Simbolo

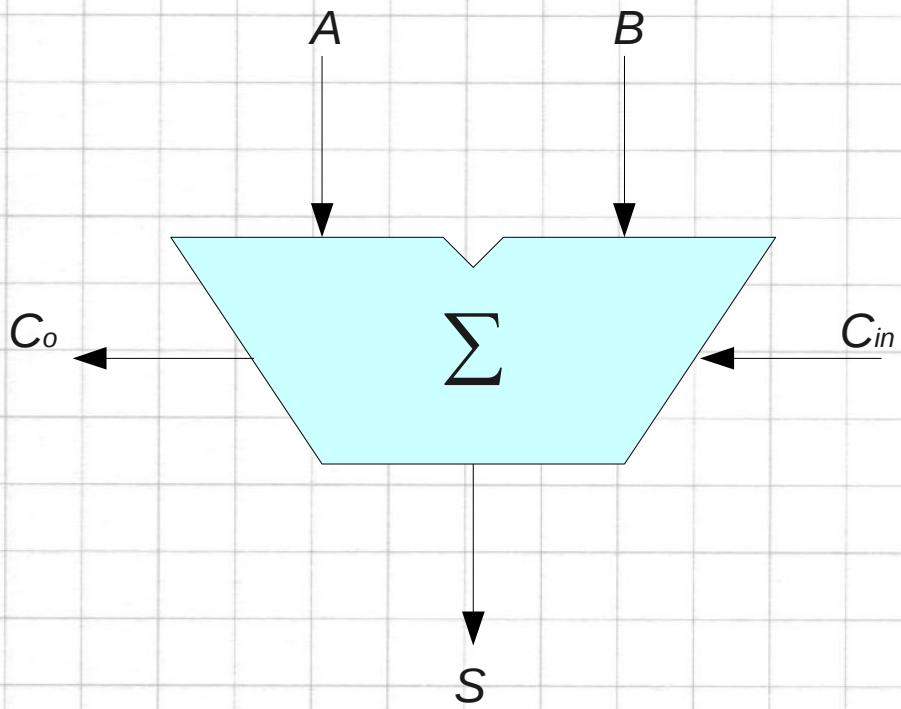
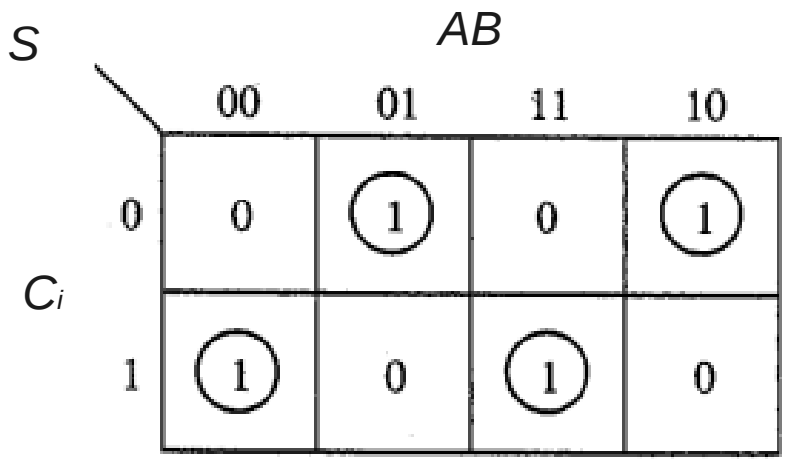


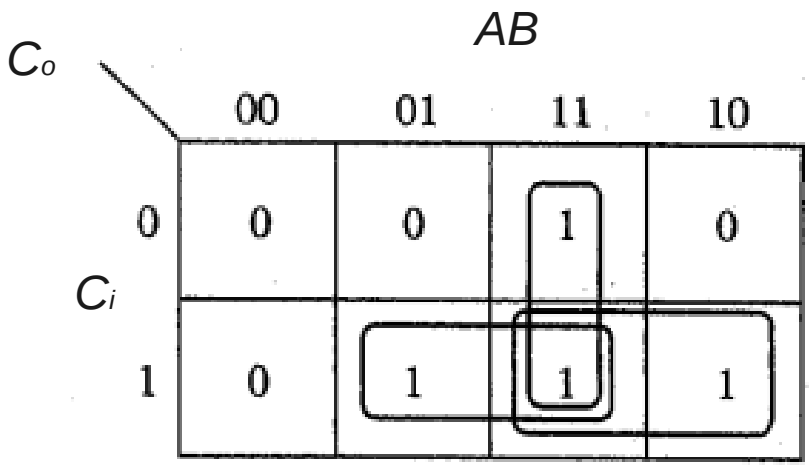
Tabella di verità

A	B	C_{in}	S	C_o
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

Il full adder (2)

Sintesi

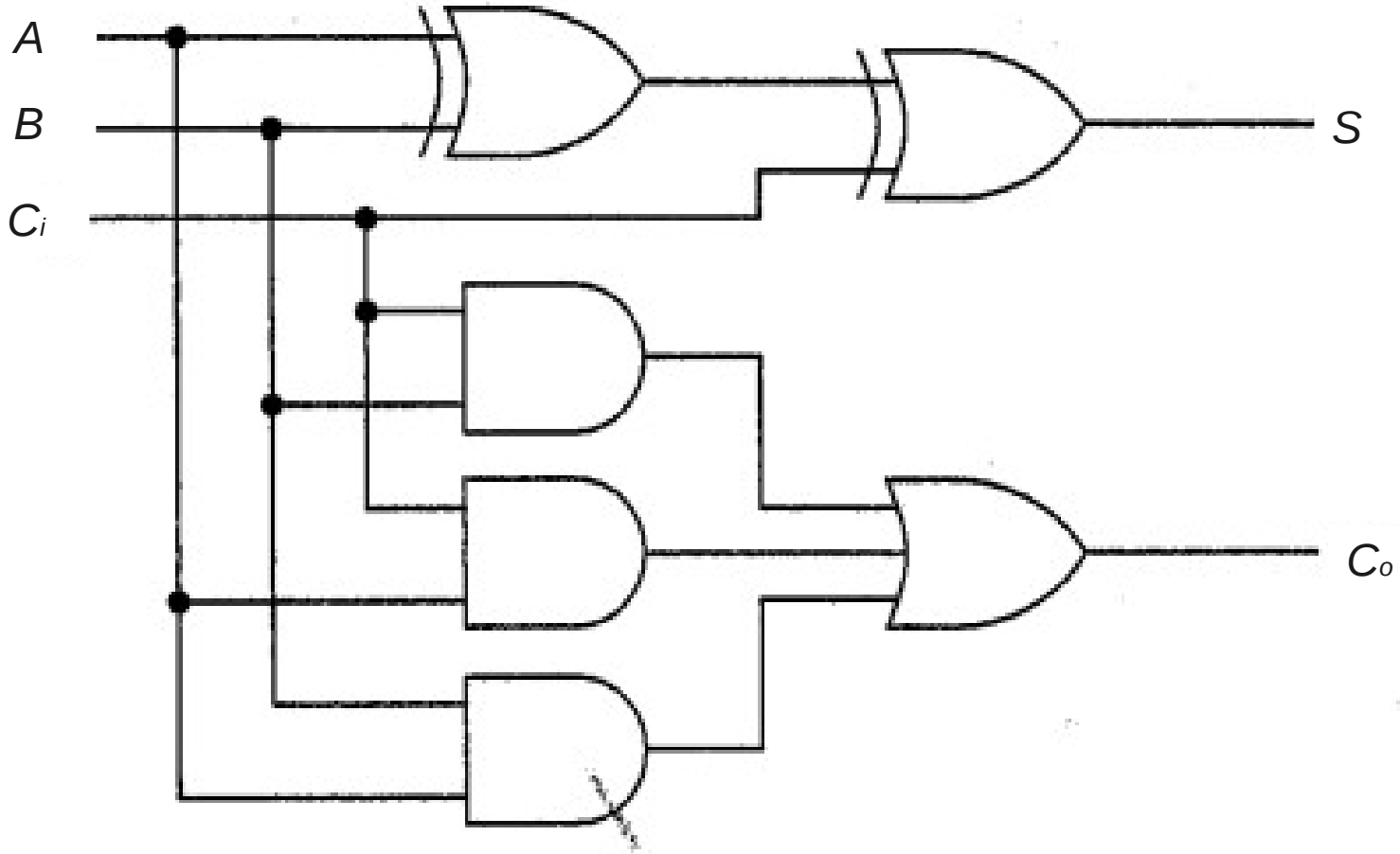




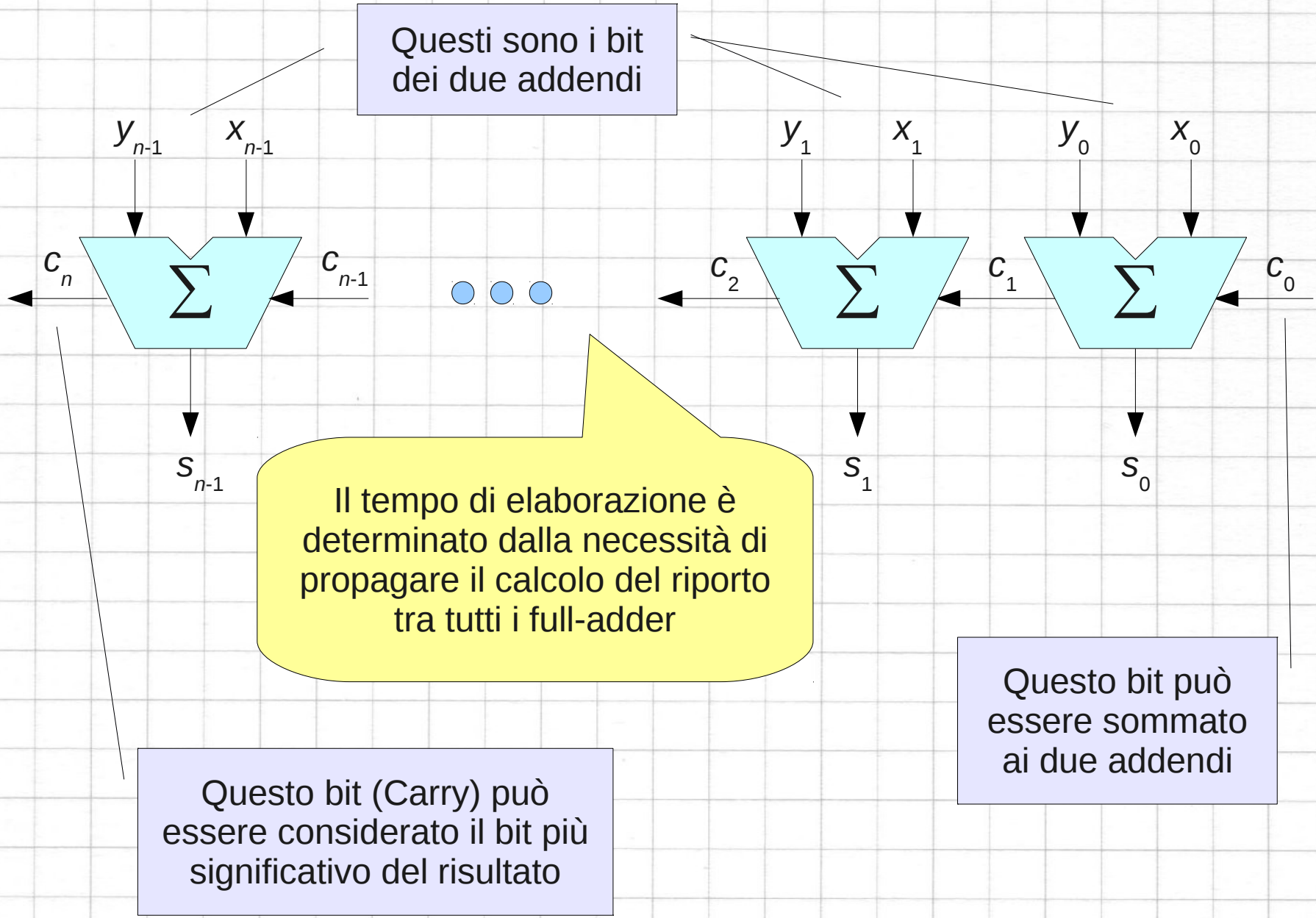
$$S = \overline{A}\,\overline{B}\,C_i + \overline{A}\,B\,\overline{C_i} + A\,\overline{B}\,\overline{C_i} + A\,B\,C_i = A \oplus B \oplus C_i$$
$$C_o = AB + AC_i + BC_i$$

Il full adder (3)

Schema



Sommatore parallelo ripple carry



Sottrattori binari

- Si realizza allo stesso modo del sommatore
  - Con *n* circuiti elementari full-subtracter
    - Eseguono la differenza tra due bit, tenendo conto di un eventuale prestito rappresentato da un terzo bit in ingresso
    - Il risultato varia da un minimo di -2
      - Minuendo nullo, sottraendo e prestito a 1
    - a un massimo di 1
      - Minuendo a 1, sottraendo e prestito nulli
    - Il risultato del full-subtracter è rappresentato dal bit della differenza e da un prestito in uscita di peso -2

Il full subtracter

Simbolo

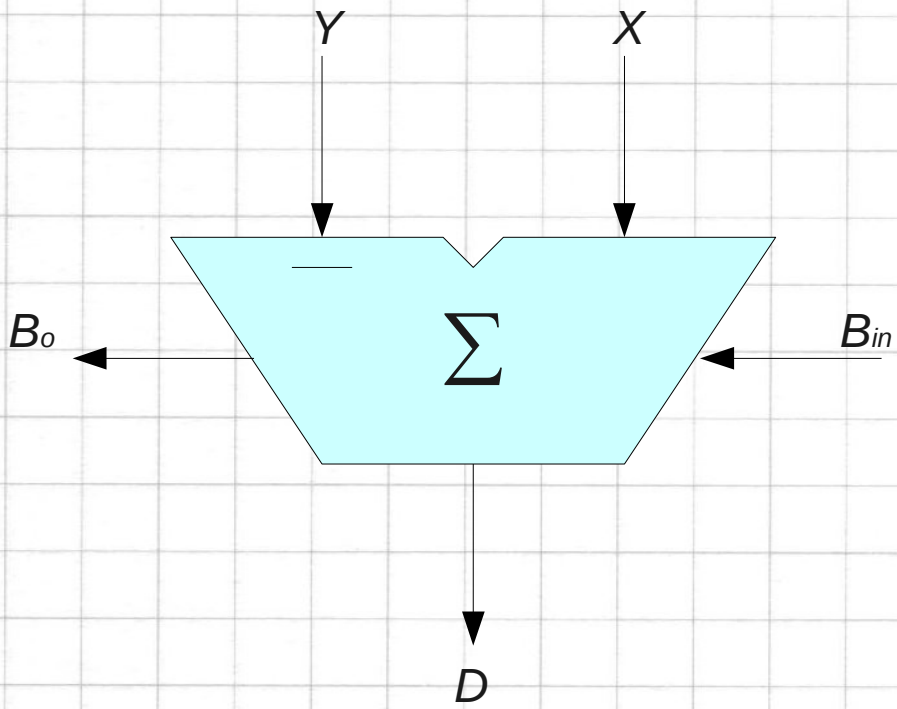
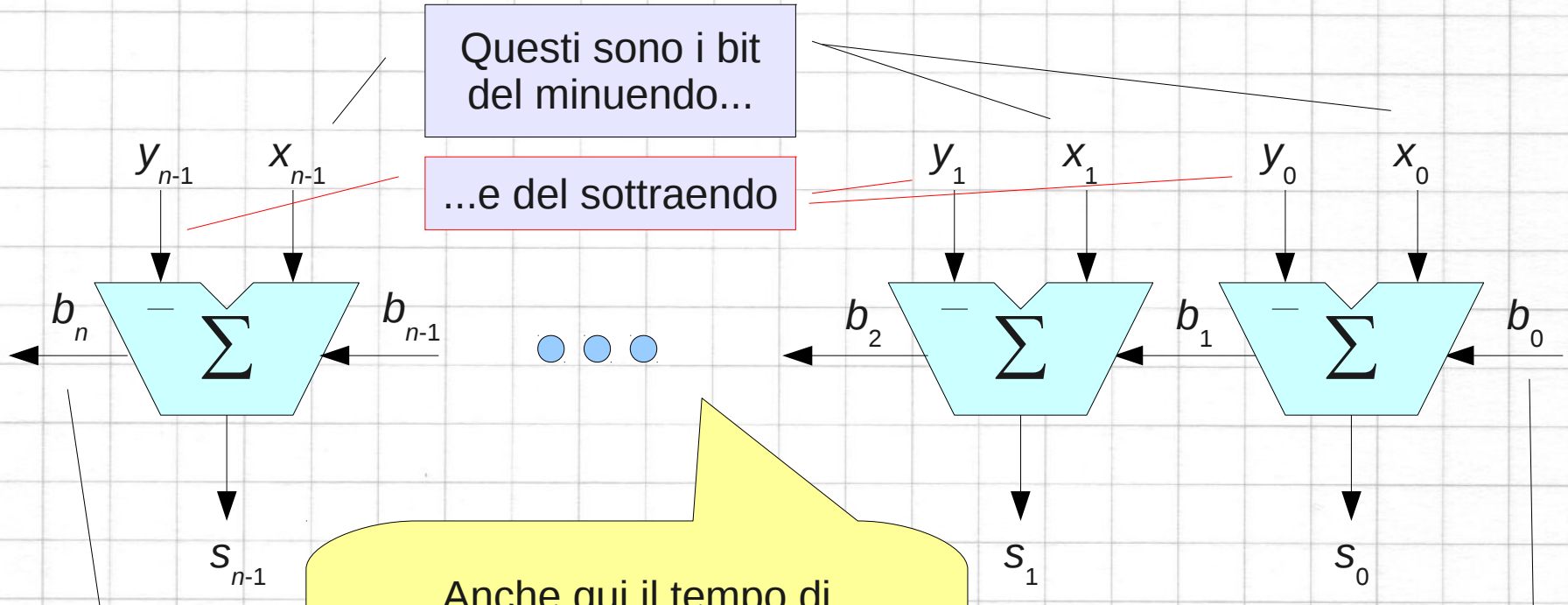


Tabella di verità

X	Y	B_{in}	D	B_o
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Sottrattore parallelo



Anche qui il tempo di elaborazione è determinato dalla necessità di propagare il calcolo del prestito

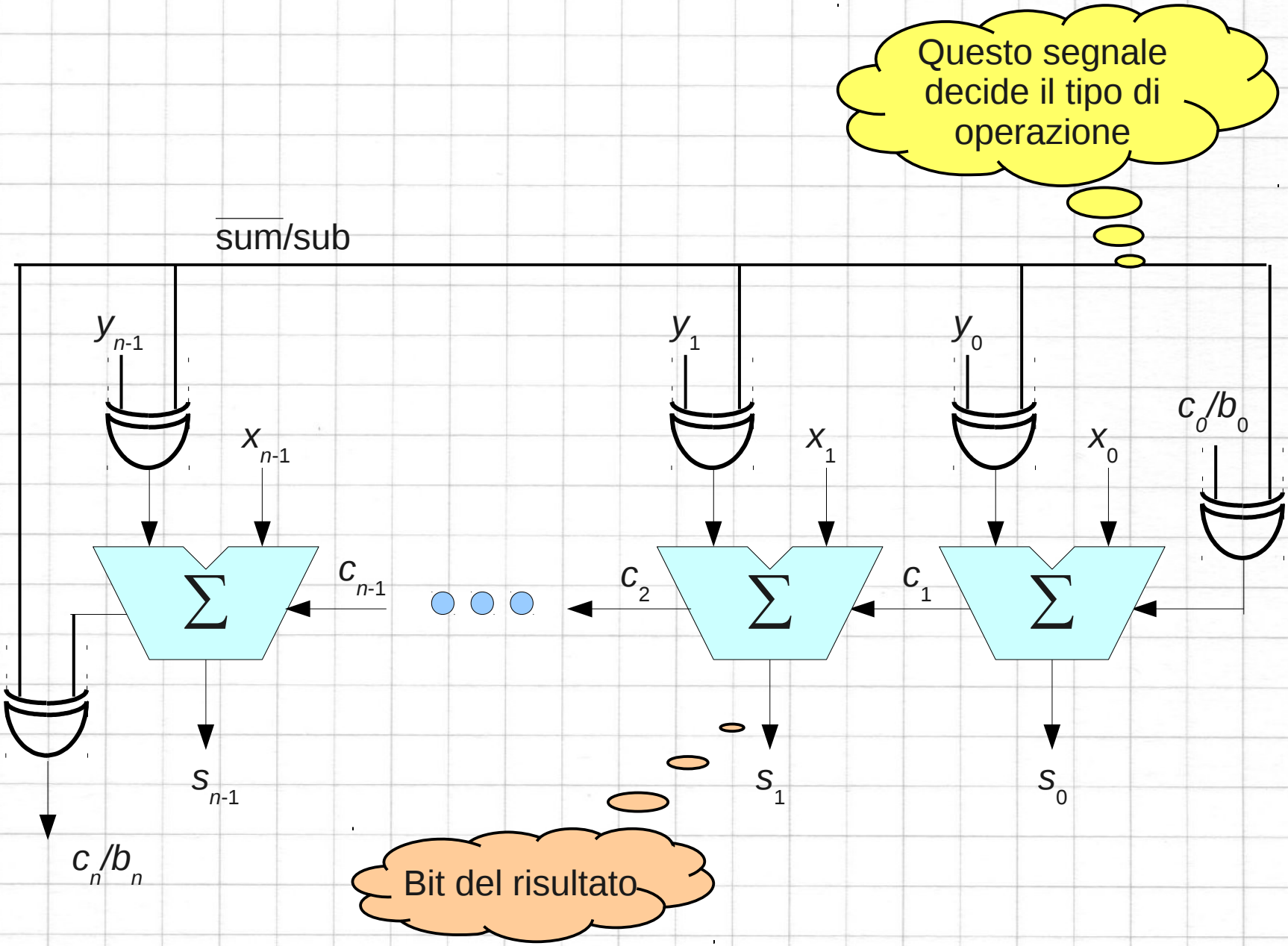
Questo bit può essere sottratto al risultato come un prestito iniziale

Questo bit (Borrow) indica se il risultato è rappresentabile (0) o no (1)

Somma/sottrattori binari

- Sommatore e sottrattore possono essere fusi in un solo circuito
  - Il sommatore esegue $X+Y+c_0=c_n 2^n+S$
  - Ponendo $Y=2^n-1-W \qquad c_0=1-b_0$
  - Si ottiene $X+(2^n-1-W)+(1-b_0)=c_n 2^n+S$
  - Da cui $X-W-b_0=-(1-c_n)2^n+S=-b_n 2^n+D$
  - Che è l'espressione della differenza se $c_n=1-b_n$
- Quindi il sottrattore si ottiene dal sommatore negando bit a bit l'ingresso invertente e i bit dal riporto in ingresso e in uscita
  - Come invertitore comandato si usa la XOR

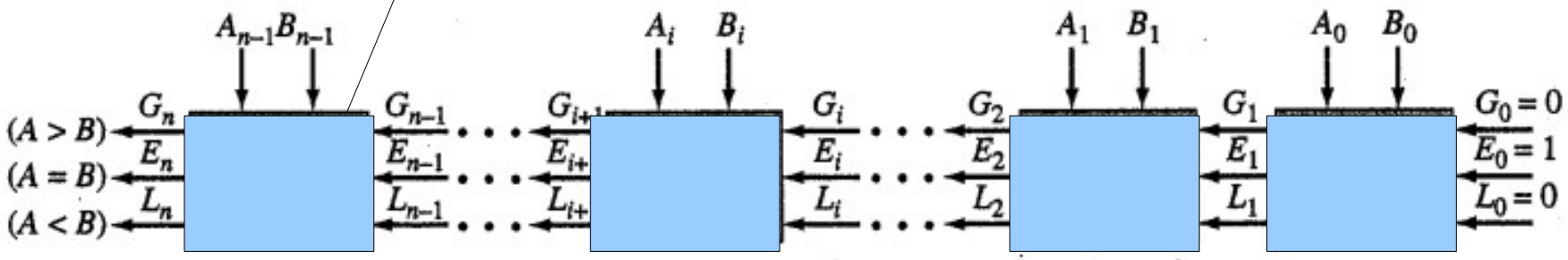
Somma-sottrattore parallelo



Comparatori

- Algoritmo del confronto (LSB 1st)
  - Configurazione iniziale di **uguaglianza**
  - Il confronto esamina due **bit corrispondenti**
    - Se uguali propaga la configurazione di ingresso
    - Se diversi
      - $A_i \overline{B_i}$ propaga la configurazione $A > B$
      - $\overline{A_i} B_i$ propaga la configurazione $A < B$
- Modularità
  - Aumenta però il tempo di propagazione
- Realizzazioni alternative
  - E possibile realizzare un comparatore MSB 1st
    - A parte l'uguaglianza, la configurazione di ingresso **prevale** sull'esito del confronto *i*-esimo

Comparatore parallelo (LSB 1st)



Elemento di confronto

Risultato del confronto

Impostazione iniziale

L'elemento di confronto (1)

A_i	B_i	G_i	E_i	L_i	G_{i+1}	E_{i+1}	L_{i+1}	A_i	B_i	G_i	E_i	L_i	G_{i+1}	E_{i+1}	L_{i+1}
0	0	0	0	0	–	–	–	1	0	0	0	0	–	–	–
0	0	0	0	1	0	0	1	1	0	0	0	1	1	0	0
0	0	0	1	0	0	1	0	1	0	0	1	0	1	0	0
0	0	0	1	1	–	–	–	1	0	0	1	1	–	–	–
0	0	1	0	0	1	0	0	1	0	1	0	0	1	0	0
0	0	1	0	1	–	–	–	1	0	1	0	1	–	–	–
0	0	1	1	0	–	–	–	1	0	1	1	0	–	–	–
0	0	1	1	1	–	–	–	1	0	1	1	1	–	–	–
0	1	0	0	0	–	–	–	1	1	0	0	0	–	–	–
0	1	0	0	1	0	0	1	1	1	0	0	1	0	0	1
0	1	0	1	0	0	0	1	1	1	0	1	0	0	1	0
0	1	0	1	1	–	–	–	1	1	0	1	1	–	–	–
0	1	1	0	0	0	0	1	1	1	1	0	0	1	0	0
0	1	1	0	1	–	–	–	1	1	1	0	1	–	–	–
0	1	1	1	0	–	–	–	1	1	1	1	0	–	–	–
0	1	1	1	1	–	–	–	1	1	1	1	1	–	–	–

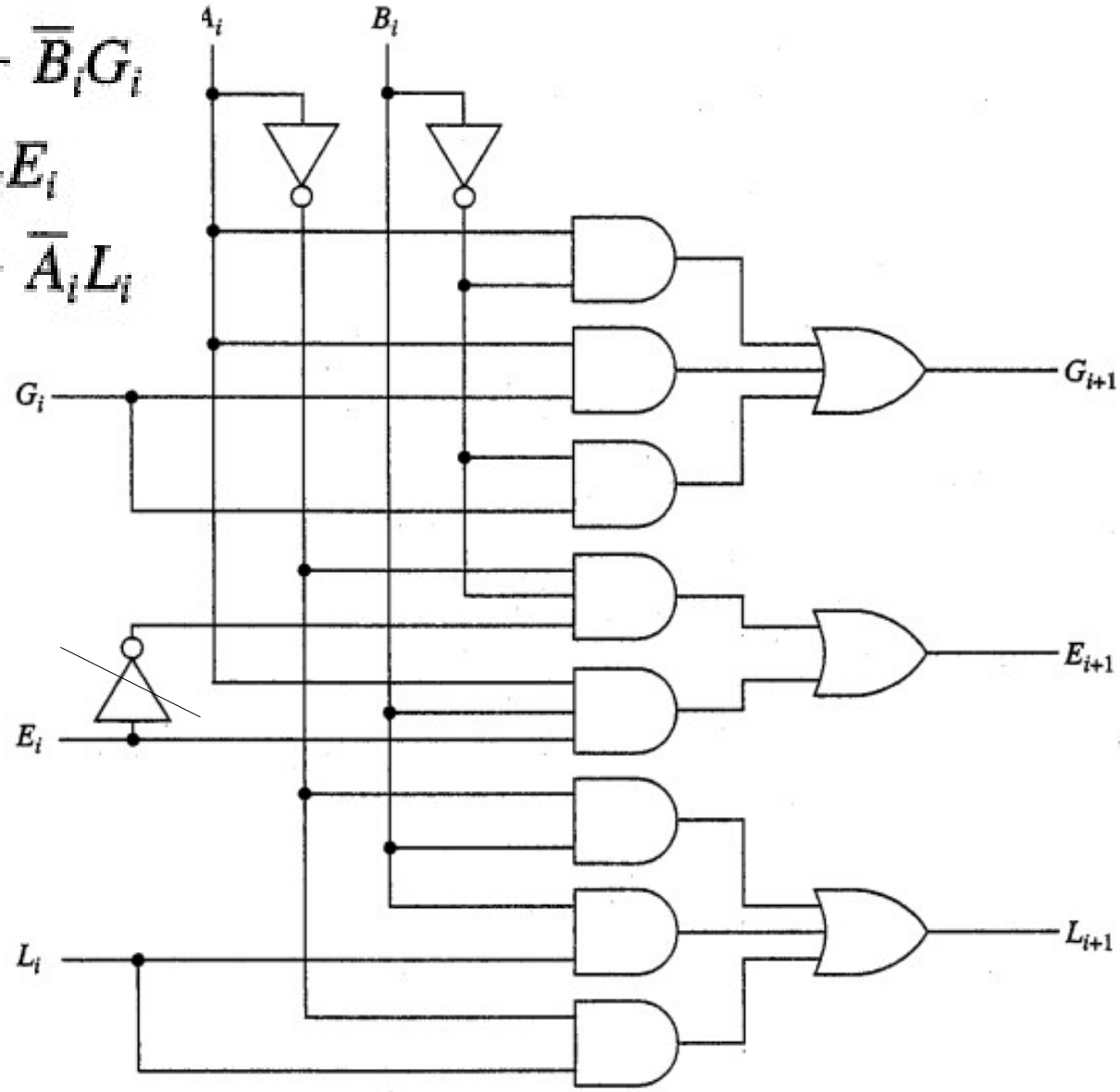
Tabella di verità

L'elemento di confronto (2)

$$G_{i+1} = A_i\overline{B}_i + A_iG_i + \overline{B}_iG_i$$

$$E_{i+1} = \overline{A}_i\overline{B}_iE_i + A_iB_iE_i$$

$$L_{i+1} = \overline{A}_iB_i + B_iL_i + \overline{A}_iL_i$$



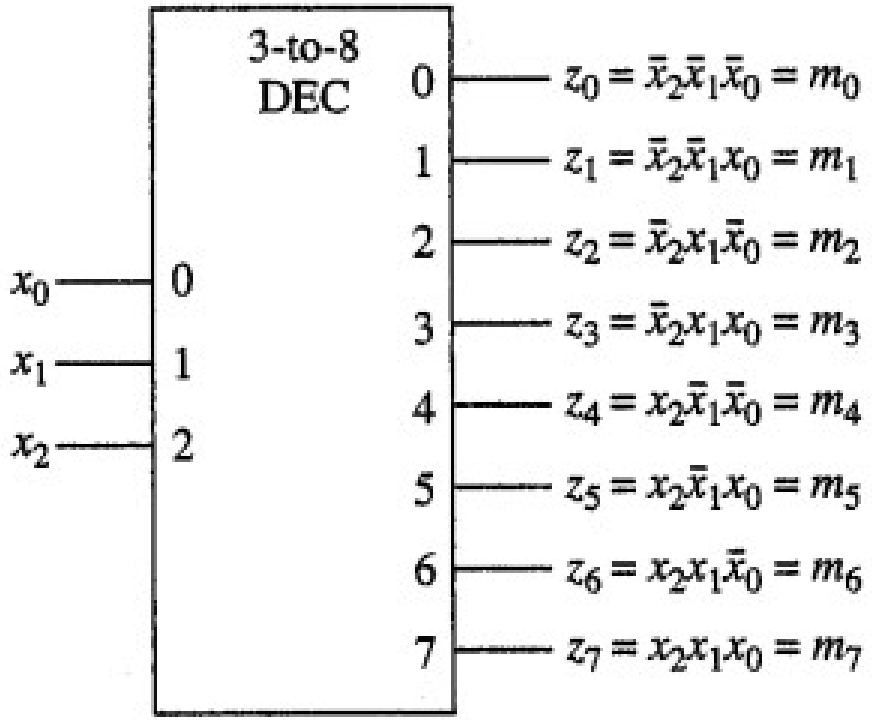
Decoder

- Esplicita l'informazione contenuta in un codice
  - Numero di uscite maggiore del numero di ingressi
    - Con l'eccezione dei codici con ridondanza
      - In cui sono tolti i bit aggiunti per rendere più sicura la trasmissione
- Esempi significativi
  - Decoder $n \rightarrow 2^n$
    - Decodificatore “generale”
    - Rete che seleziona una linea tra quelle che possono indirizzare gli ingressi di selezione
    - Esiste la versione con abilitazione
  - Decoder specifici: decoder BCD - 7 segmenti
    - Pilota i segmenti di un display numerico

Un esempio: decoder 3 a 8 (1)

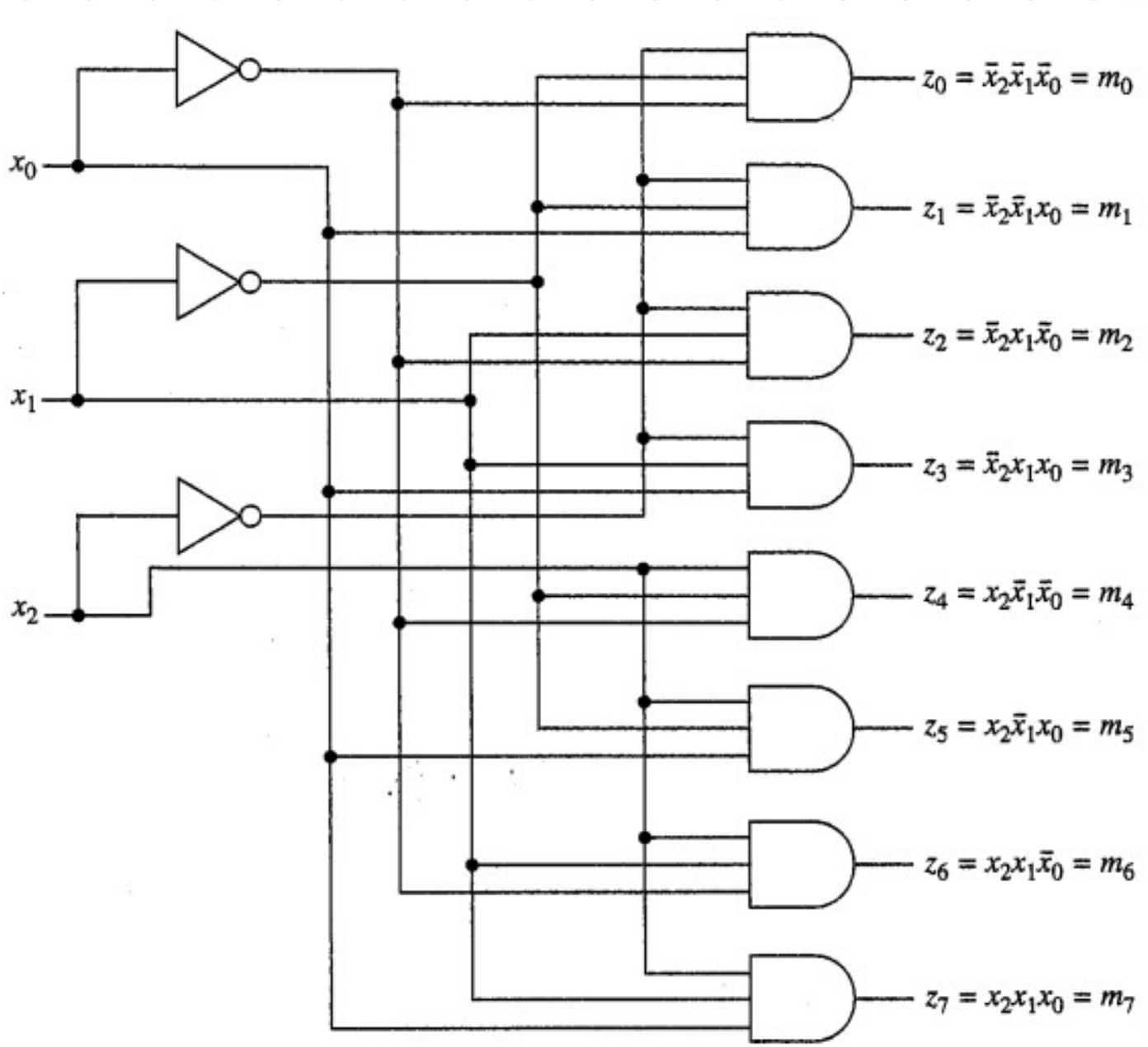
Inputs			Outputs							
x_2	x_1	x_0	z_0	z_1	z_2	z_3	z_4	z_5	z_6	z_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Tabella di verità

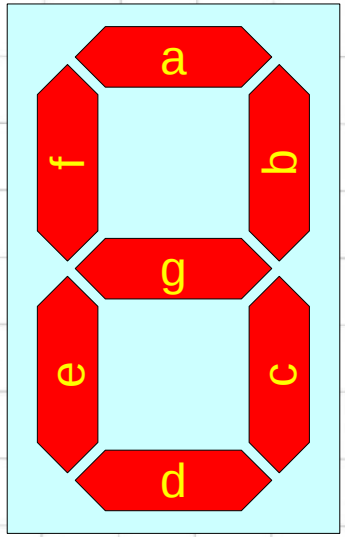


Simbolo grafico a blocco

Un esempio: decoder 3 a 8 (2)



Un esempio: BCD-7segmenti



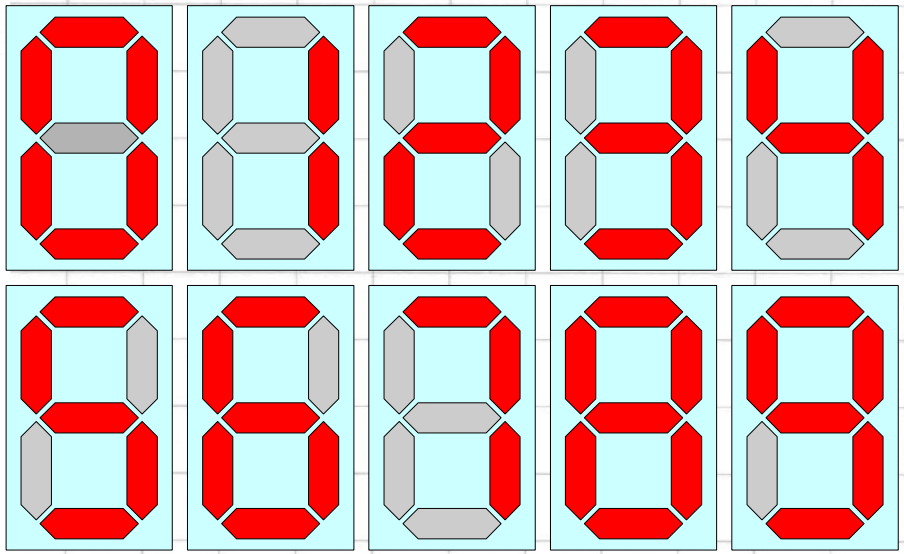


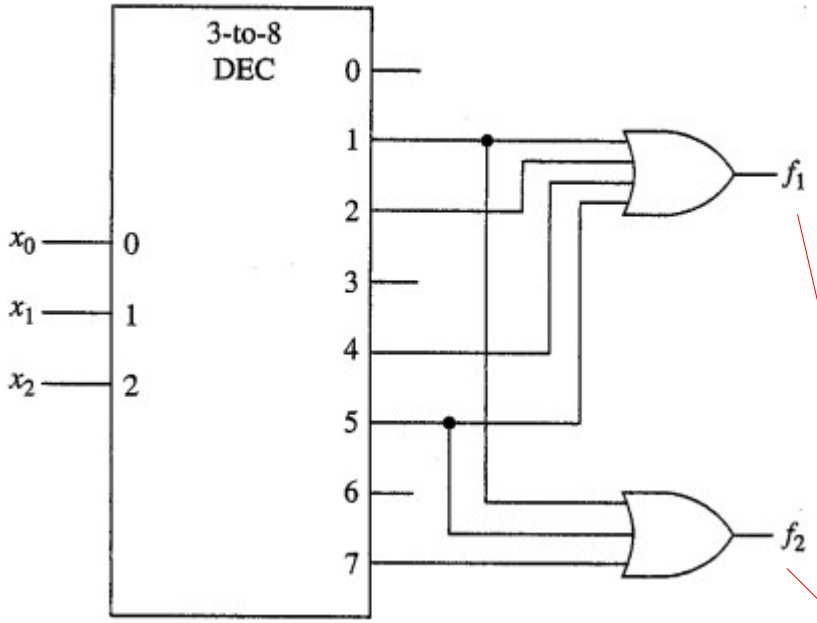
Tabella di verità

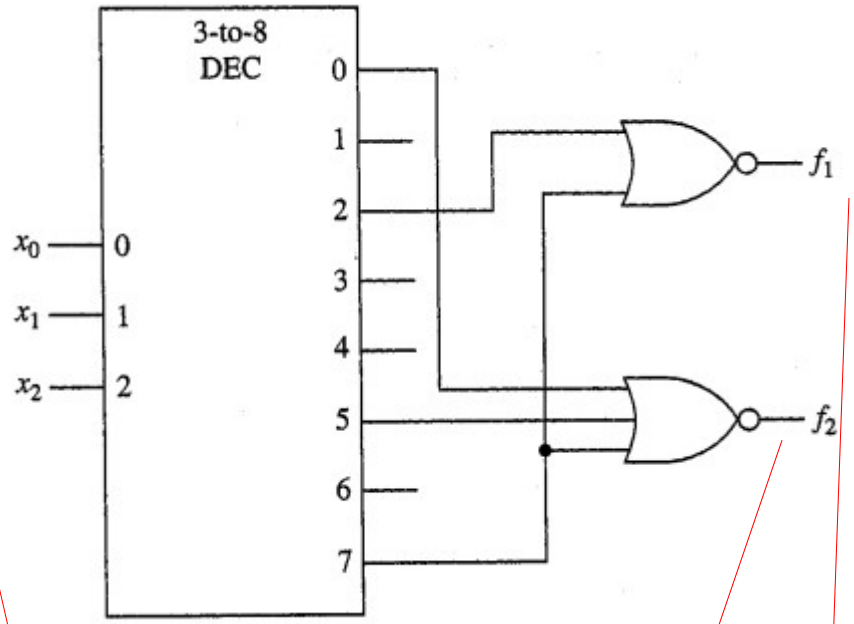
X_3	X_2	X_1	X_0	*a*	*b*	*c*	*d*	*e*	*f*	*g*
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

Logiche con decoder (1)

- I decoder $n \rightarrow 2^n$ possono essere usati per realizzare logiche generiche
  - Le uscite del decoder corrispondono ai mintermini
    - Con una OR si realizza la forma canonica SP
    - Se i mintermini sono più della metà delle uscite del decoder conviene usare una NOR
      - I cui ingressi sono i mintermini non presenti nella forma SP
      - Quindi per gli ingressi che li verificano, la funzione deve essere 0
- Può essere un modo conveniente di ottenere funzioni a molte uscite
  - Con un solo decoder e diverse porte OR/NOR

Logiche con decoder (2)





$$f_2 = x_0 \bar{x}_1 \bar{x}_2 + x_0 \bar{x}_1 x_2 + x_0 x_1 x_2$$

$$f_1 = x_0 \bar{x}_1 \bar{x}_2 + \bar{x}_0 x_1 \bar{x}_2 + \bar{x}_0 \bar{x}_1 x_2 + x_0 \bar{x}_1 x_2$$

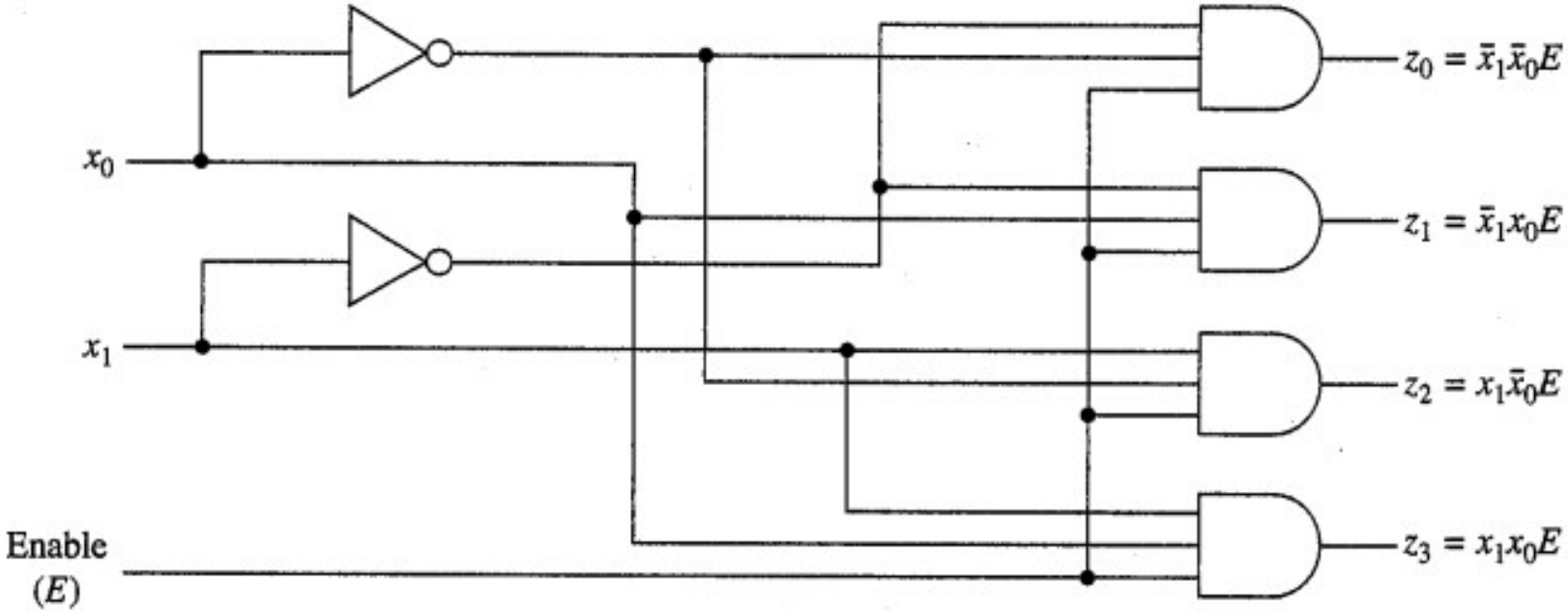
$$f_2 = x_0 \bar{x}_1 \bar{x}_2 + \bar{x}_0 x_1 \bar{x}_2 + x_0 x_1 \bar{x}_2 + \bar{x}_0 \bar{x}_1 x_2 + \bar{x}_0 x_1 x_2$$

$$f_1 = \bar{x}_0 \bar{x}_1 \bar{x}_2 + x_0 \bar{x}_1 \bar{x}_2 + x_0 x_1 \bar{x}_2 + \bar{x}_0 \bar{x}_1 x_2 + x_0 \bar{x}_1 x_2 + \bar{x}_0 x_1 x_2$$

Decoder con abilitazione (1)

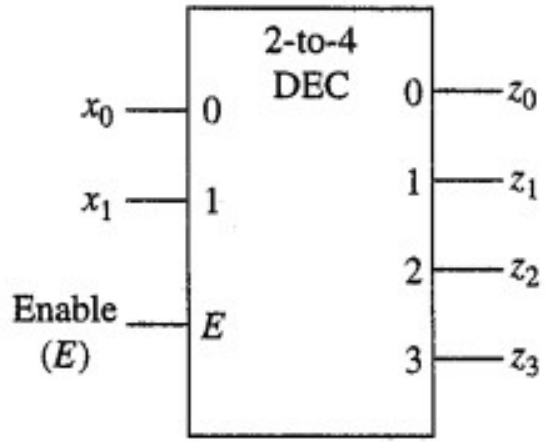
- Oltre agli ingressi di selezione, dispone di un ingresso generale *E* che abilita la selezione stessa
  - Se *E* = 0 tutte le line di uscita sono nulle
  - Aumenta la versatilità del blocco; in qualche caso gli ingressi di abilitazione sono più di uno
  - Permette la modularità, aumentando il numero di linee selezionabili

Decoder con abilitazione (2)



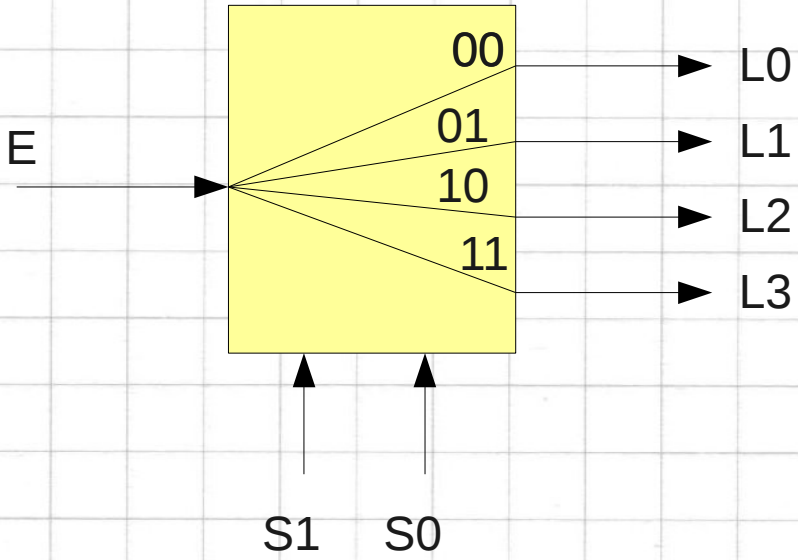
(a)

Inputs			Outputs			
E	x_1	x_0	z_0	z_1	z_2	z_3
0	×	×	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1



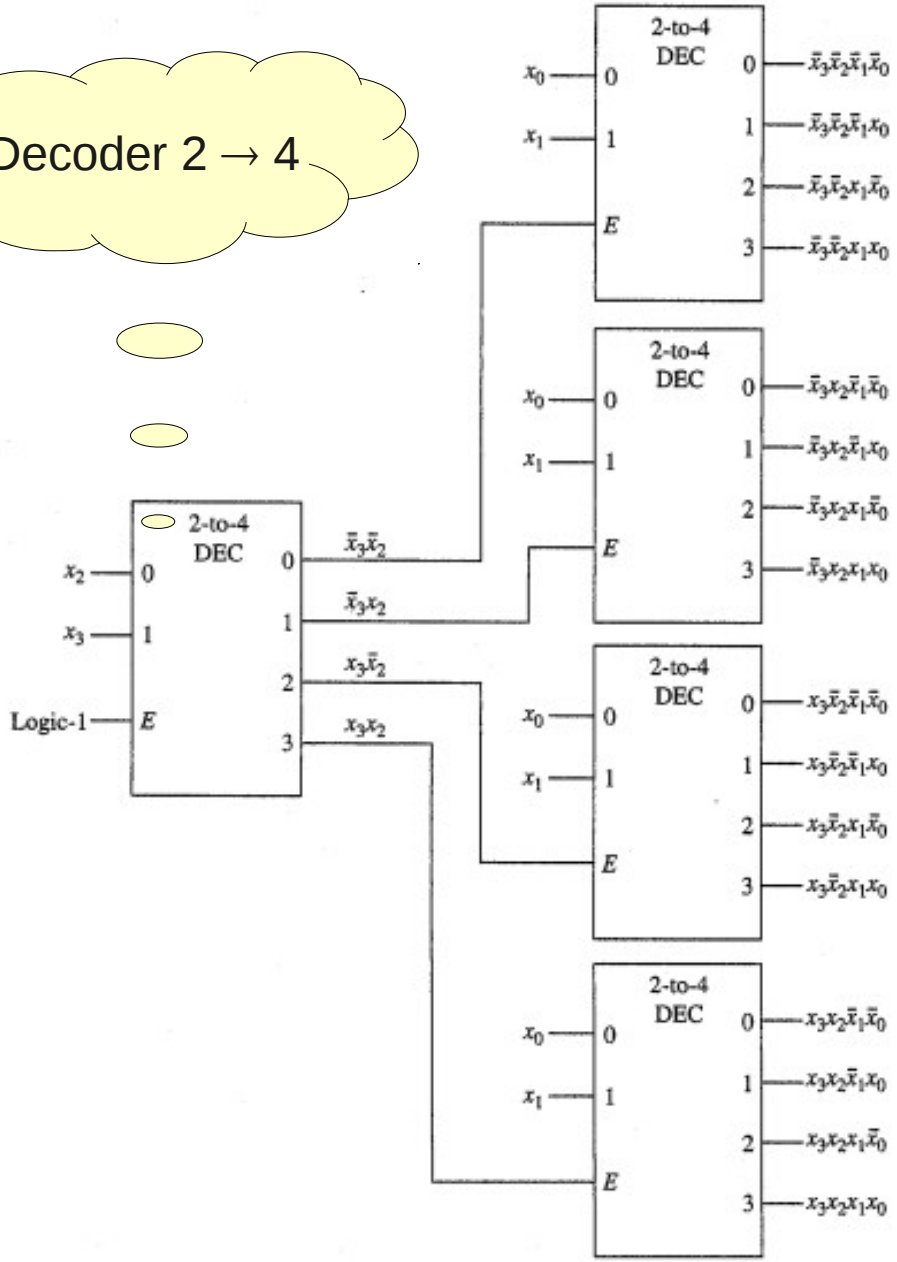
Decoder con abilitazione (3)

- Il decoder con abilitazione è chiamato anche demultiplexer (smistatore)
  - Le linee di selezione indicano su quale linea viene instradato il segnale di abilitazione
  - Ha un simbolo specifico



Modularità dei decoder

Decoder 2 → 4



Decoder 4 → 16

Per ottenere un decoder 6 → 64 si possono aggiungere altri 16 decoder 2 → 4.

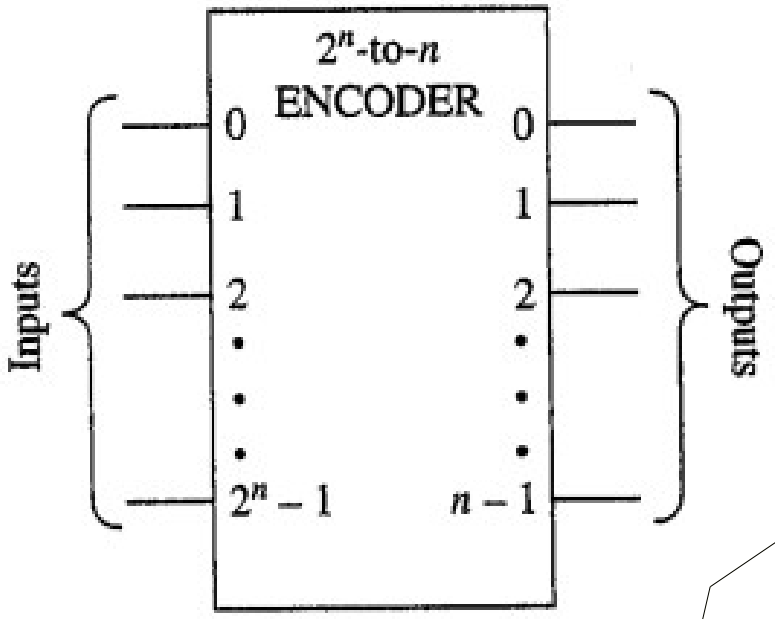
Decoder necessari:
1 per 4 uscite
5 per 16 uscite
21 per 64, 85...

In generale, per ottenere un decoder $2n \rightarrow 2^{(2n)}$ occorrono $[2^{(2n)} -1]/3$ decoder 2 → 4.

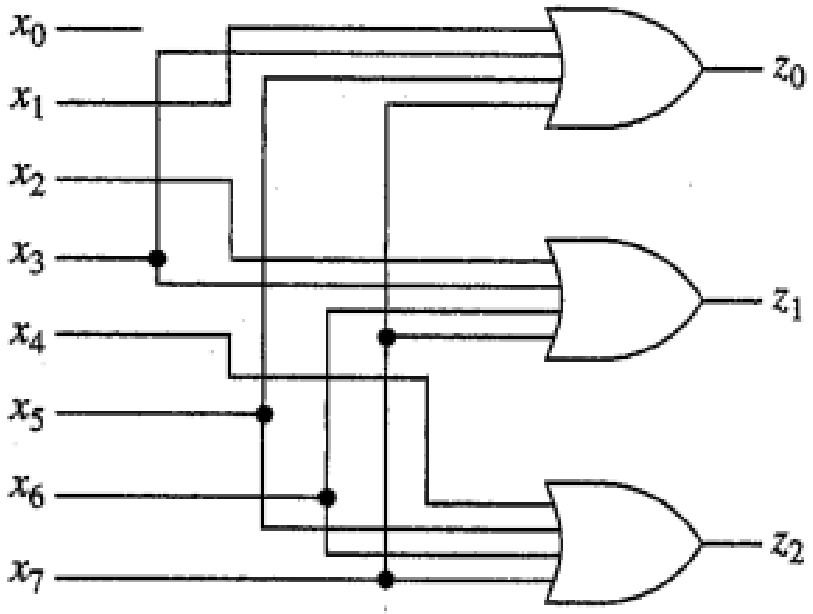
Encoder

- Codifica l'informazione contenuta in forma espansa nel mondo reale
  - Numero di uscite minore numero di ingressi
    - Con l'eccezione dei codici con ridondanza
      - In cui sono aggiunti bit al dato di partenza
      - Per la correzione degli errori di trasmissione
- Esempi significativi
  - Encoder $2^n \rightarrow n$ con priorità
    - Rete che identifica una linea attiva tra n possibili
    - In caso di più linee attive attribuisce una priorità
    - Esiste la versione con abilitazione
  - Encoder specifici
    - Applicazioni di compressione audio/video
    - Compressione file

Encoder $2^n \rightarrow n$



$$z_0 = x_1 + x_3 + x_5 + x_7$$
$$z_1 = x_2 + x_3 + x_6 + x_7$$
$$z_2 = x_4 + x_5 + x_6 + x_7$$



Porta a 1 le cifre corrispondenti al suo numero d'ordine.
È attiva una sola linea di ingresso alla volta.

Se più linee sono attive, il risultato non è significativo

Encoder con priorità (8 a 3)

Linee di ingresso

Uscite

Ingresso valido

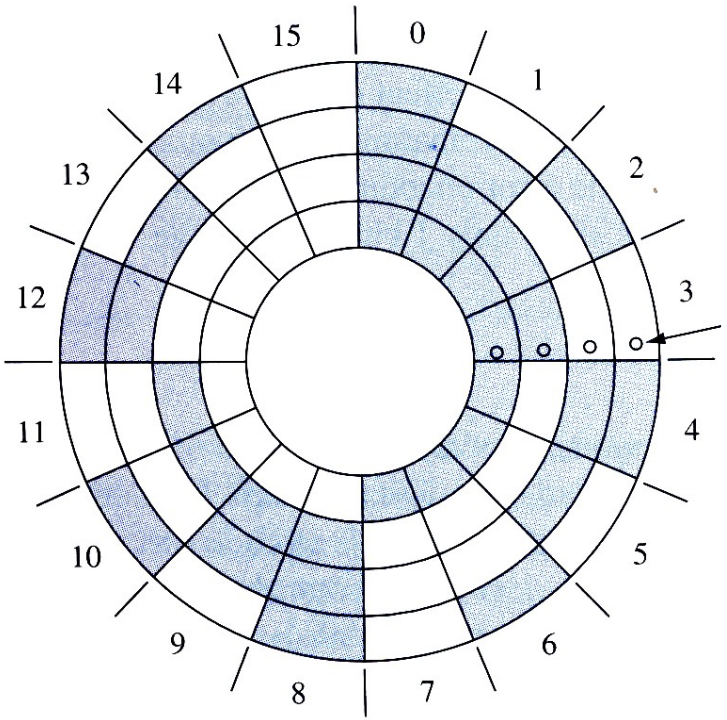
X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_0	Z_2	Z_1	Z_0	V
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	1	×	0	0	1	1
0	0	0	0	0	1	×	×	0	1	0	1
0	0	0	0	1	×	×	×	0	1	1	1
0	0	0	1	×	×	×	×	1	0	0	1
0	0	1	×	×	×	×	×	1	0	1	1
0	1	×	×	×	×	×	×	1	1	0	1
1	×	×	×	×	×	×	×	1	1	1	1

Tabella di verità compressa

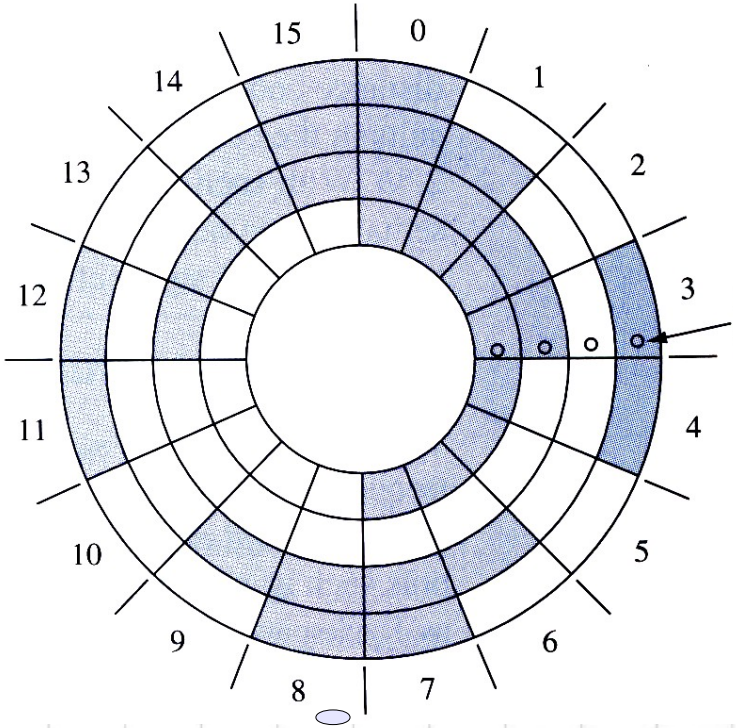
Un particolare codice

- Codice Gray
  - Codice ordinato di valori interi binari
    - Ogni valore differisce dal precedente (e quindi dal successivo) per una sola cifra binaria
      - Il numero di cifre differenti tra due numeri binari si definisce distanza di Hamming
      - In un codice Gray la distanza di Hamming tra numeri vicini è 1
    - Esistono 2^n codici diversi
      - L'insieme dei codici Gray può essere messo in relazione biunivoca con i numeri binari con lo stesso numero di cifre
- Applicazioni
  - Codifica di posizione lineare o angolare
    - Lettura della posizione di un rettangolo o di un cerchio
      - In cui sono disegnati pattern opportuni
      - Tramite fotosensori

En(De)coder Binario-Gray



...
0010
0011
0100
0101
...



...
0011
0010
0110
0111
...

Encoder binario

Encoder Gray

Un disallineamento dei sensori nella regione di transizione provoca nell'encoder binario errori di grandezza arbitraria nell'encoder Gray solo errori di ± 1

Costruire il codice Gray (1)

- Costruzione per induzione
  - Un codice Gray a 1 bit G_1 è banale
  - Dato G_n a n bit, si costruisce G_{n+1} a $(n + 1)$ bit
    - La prima metà del codice G_{n+1} coincide con G_n
      - A cui è aggiunto un bit g_n (per esempio come MSB) di valore 0
    - La seconda metà si ottiene da G_n in ordine inverso
      - E aggiungendo un bit g_n di valore 1
- Il codice Gray non è unico
  - Esistono $2^n n!$ codici diversi che godono della proprietà che definisce una sequenza di numeri come codice Gray

Costruire il codice Gray (2)

Tabella di verità

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
b_3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
b_2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
b_1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
b_0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
g_3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
g_2	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
g_1	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
g_0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0

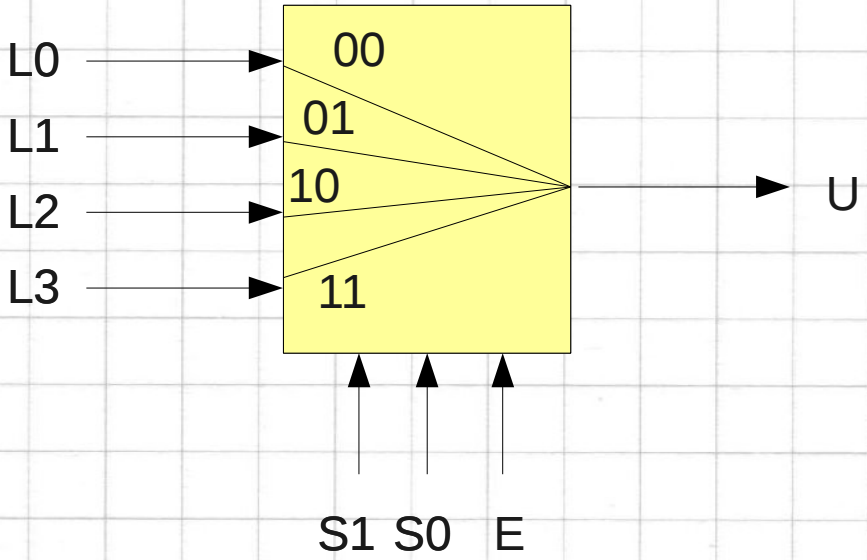
Sintesi ad hoc

$$g_3 = b_3 \qquad g_2 = b_3 \oplus b_2 \qquad g_1 = b_2 \oplus b_1 \qquad g_0 = b_1 \oplus b_0$$

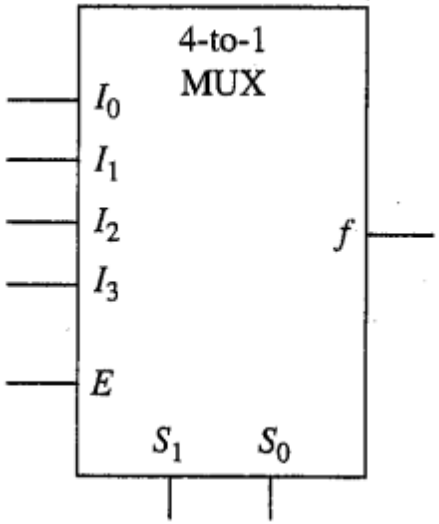
$$b_3 = g_3 \qquad b_2 = g_3 \oplus g_2 \qquad b_1 = g_3 \oplus g_2 \oplus g_1 \qquad b_0 = g_3 \oplus g_2 \oplus g_1 \oplus g_0$$

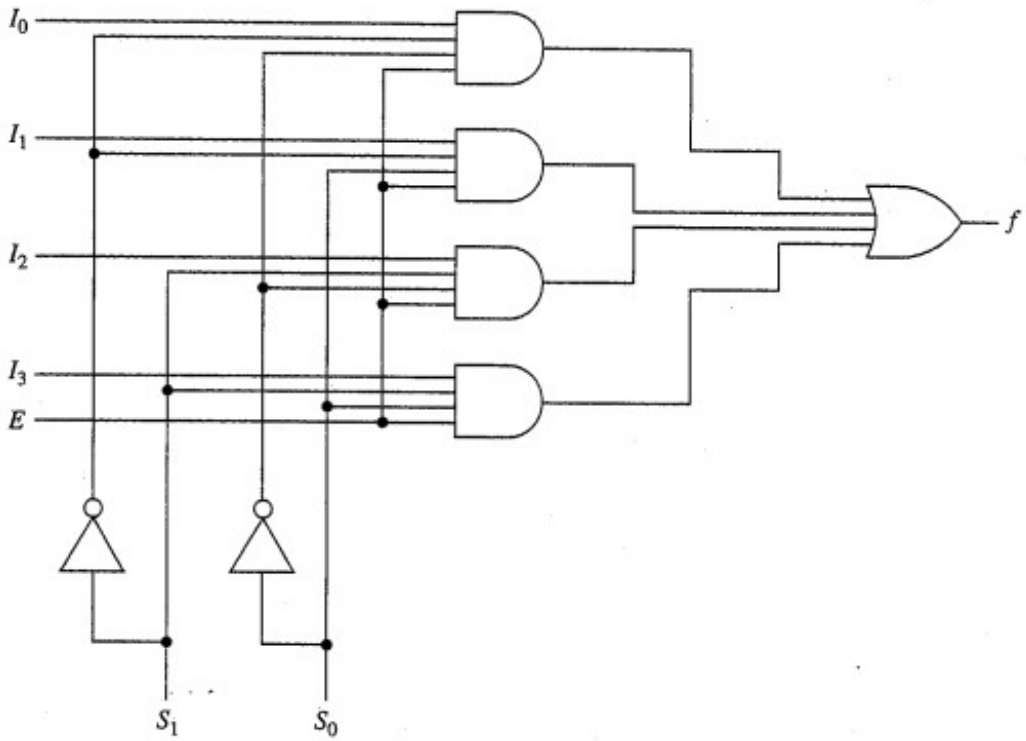
Multiplexer

- Circuito di instradamento
  - Esegue la funzione inversa del demultiplexer
    - Instrada verso l'uscita il valore della linea selezionata
    - Ha quindi *n* ingressi di selezione per 2^n linee di ingresso
  - Può avere una linea di abilitazione
    - Che fissa a 0 il valore dell'uscita, se non attiva



Multiplexer





E	S_1	S_0	I_0	I_1	I_2	I_3	f
0	×	×	×	×	×	×	0
1	0	0	0	×	×	×	0
1	0	0	1	×	×	×	1
1	0	1	×	0	×	×	0
1	0	1	×	1	×	×	1
1	1	0	×	×	0	×	0
1	1	0	×	×	1	×	1
1	1	1	×	×	×	0	0
1	1	1	×	×	×	1	1

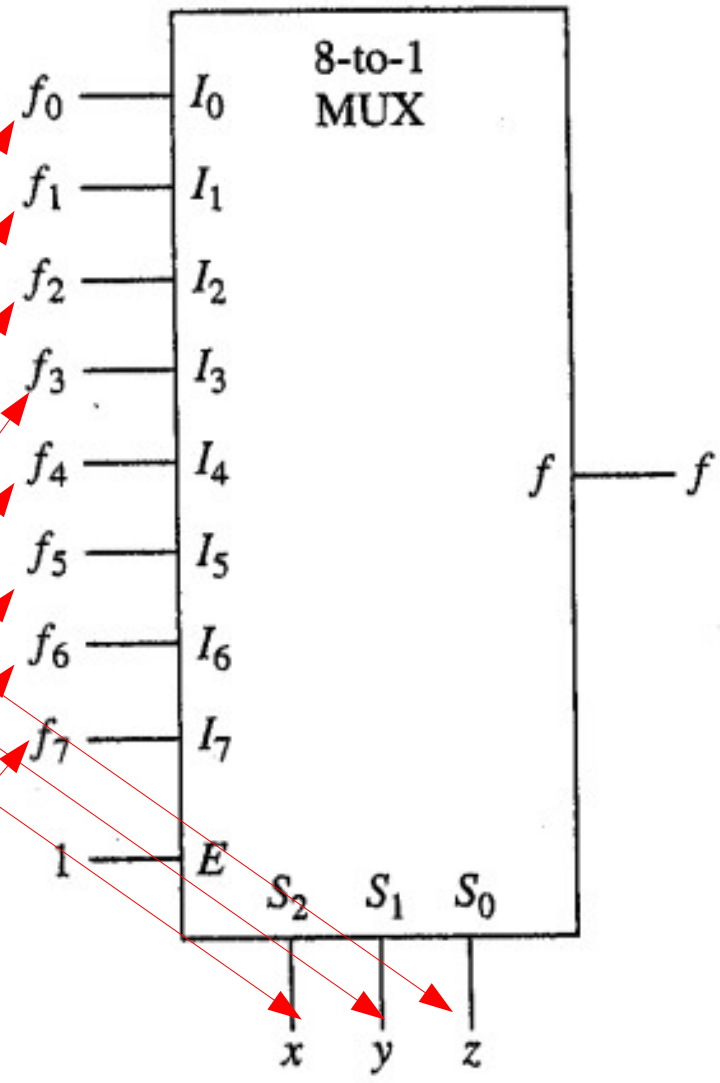
Logiche con multiplexer (1)

- Come il decoder, il mux può essere usato per realizzare logiche generiche
  - In modo diretto
    - Applicando gli ingressi della reta da realizzare alle linee di selezione e mettendo delle costanti in ingresso
  - In modo composto
    - Applicando gli ingressi della rete sia alle linee di selezione sia, parzialmente, alle linee dei dati
    - In questo caso è necessario disporre del dato affermato e negato

Logiche con multiplexer (2)

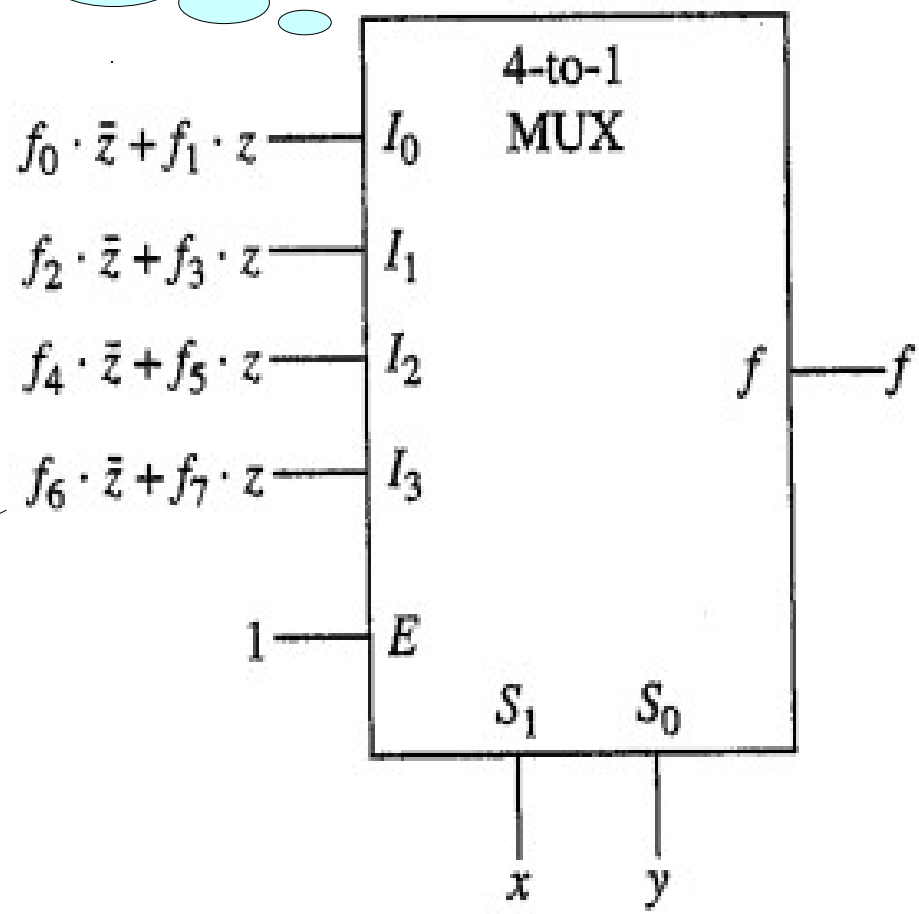
Esempio di generica funzione da realizzare

x	y	z	f
0	0	0	f_0
0	0	1	f_1
0	1	0	f_2
0	1	1	f_3
1	0	0	f_4
1	0	1	f_5
1	1	0	f_6
1	1	1	f_7



Logiche con multiplexer (3)

Avendo una variabile anche in forma negata si può risparmiare un ingresso nel mux

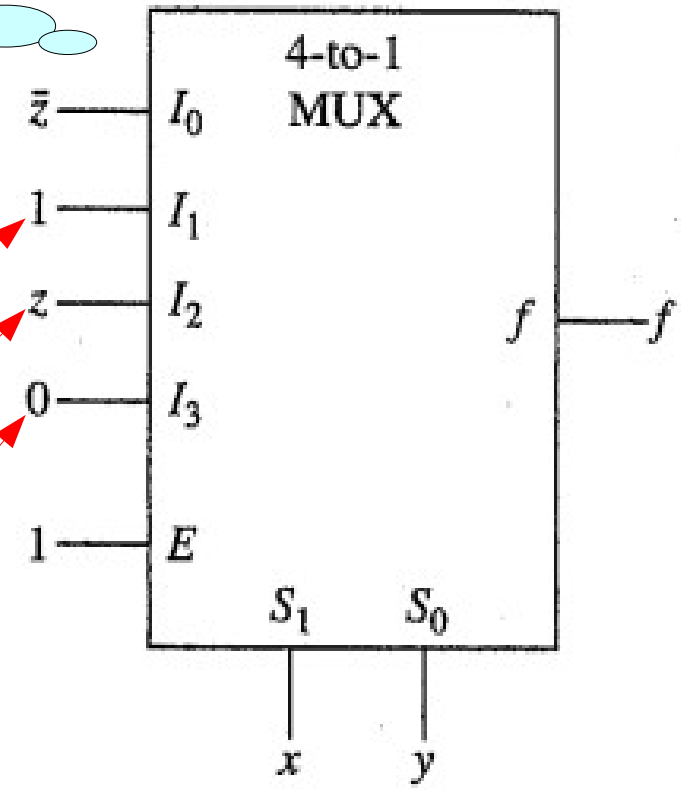


Ogni ingresso può valere 0, 1, z oppure $\bar{z}$

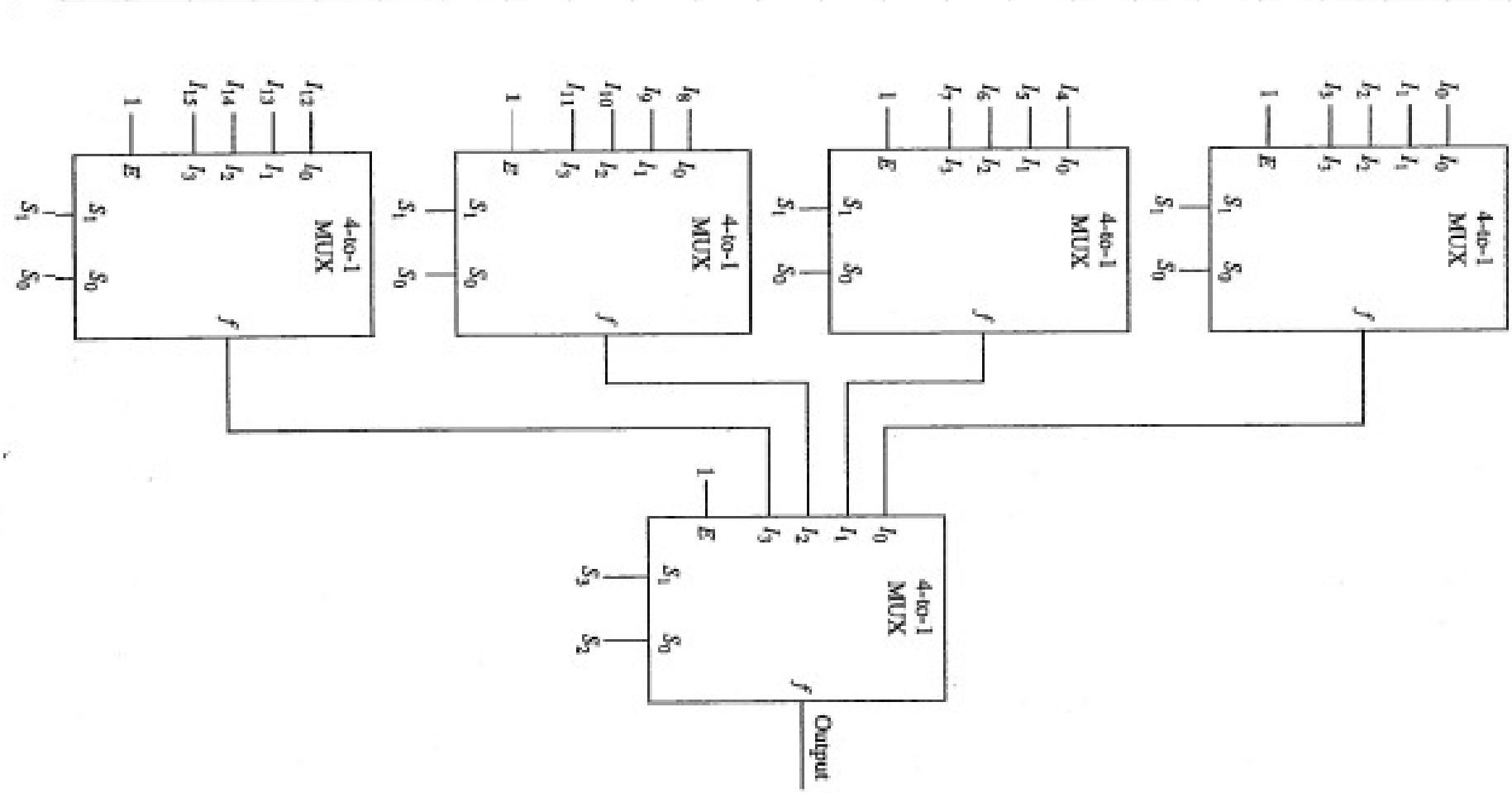
Logiche con multiplexer (4)

Stesso esempio precedentemente realizzato con mux 8 a 1

x	y	z	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



Espansione di multiplexer



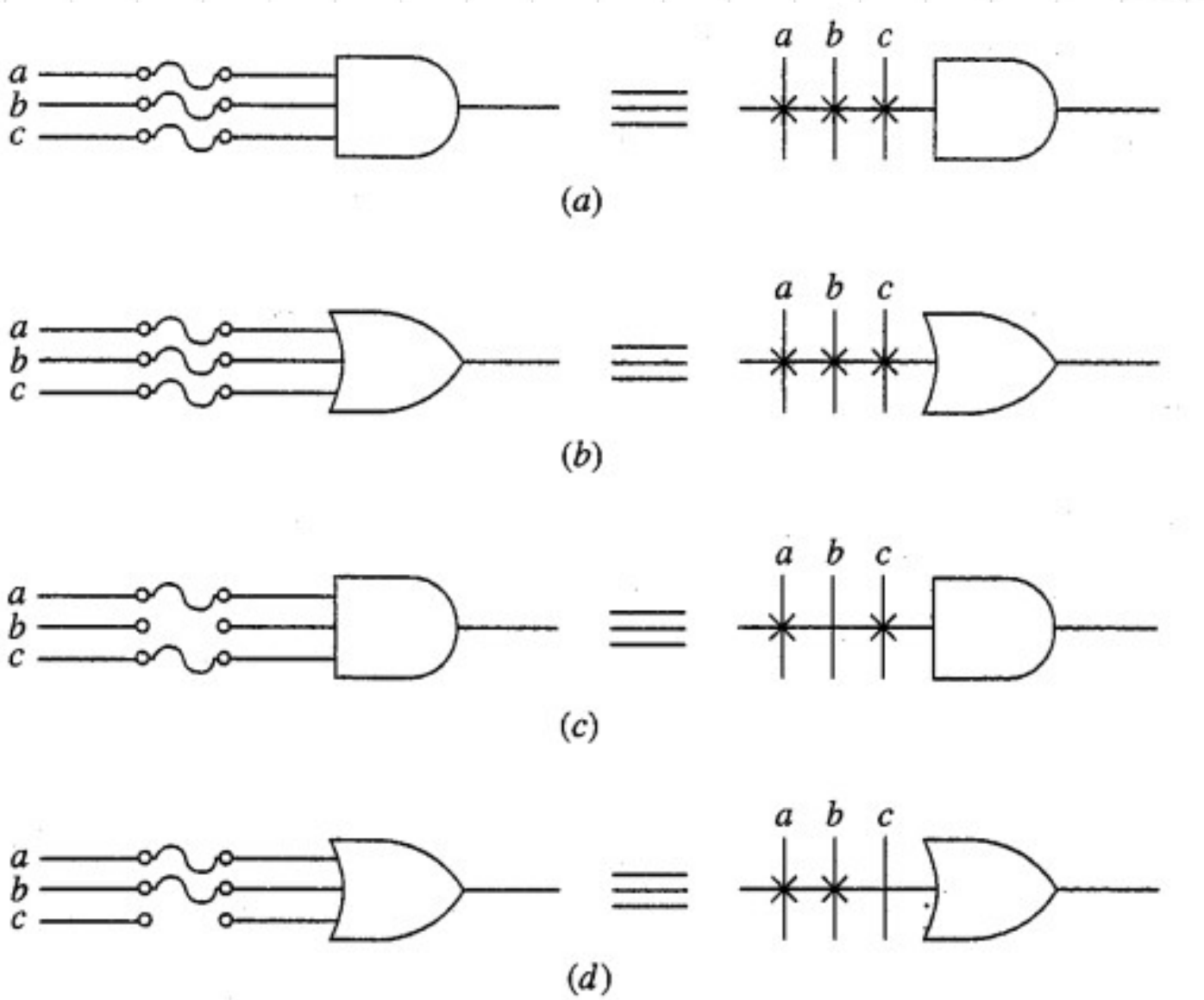
Dispositivi logici programmabili

- Connessioni modificabili dopo la fabbricazione permettono di realizzare logiche **programmabili**
  - Fusibili, antifusibili, switch a stato solido
  - Programmazione da parte dell'utente
    - In fase di montaggio
    - Con componente già montato nel sistema
      - In system programming (ISP)
- Diverse **architetture**
  - Inizialmente basate sulle forme normali
    - PAL, PLA, PLD
  - Poi sempre più libere
    - FPGA, SoPC
    - Grazie a CAD sempre più potenti

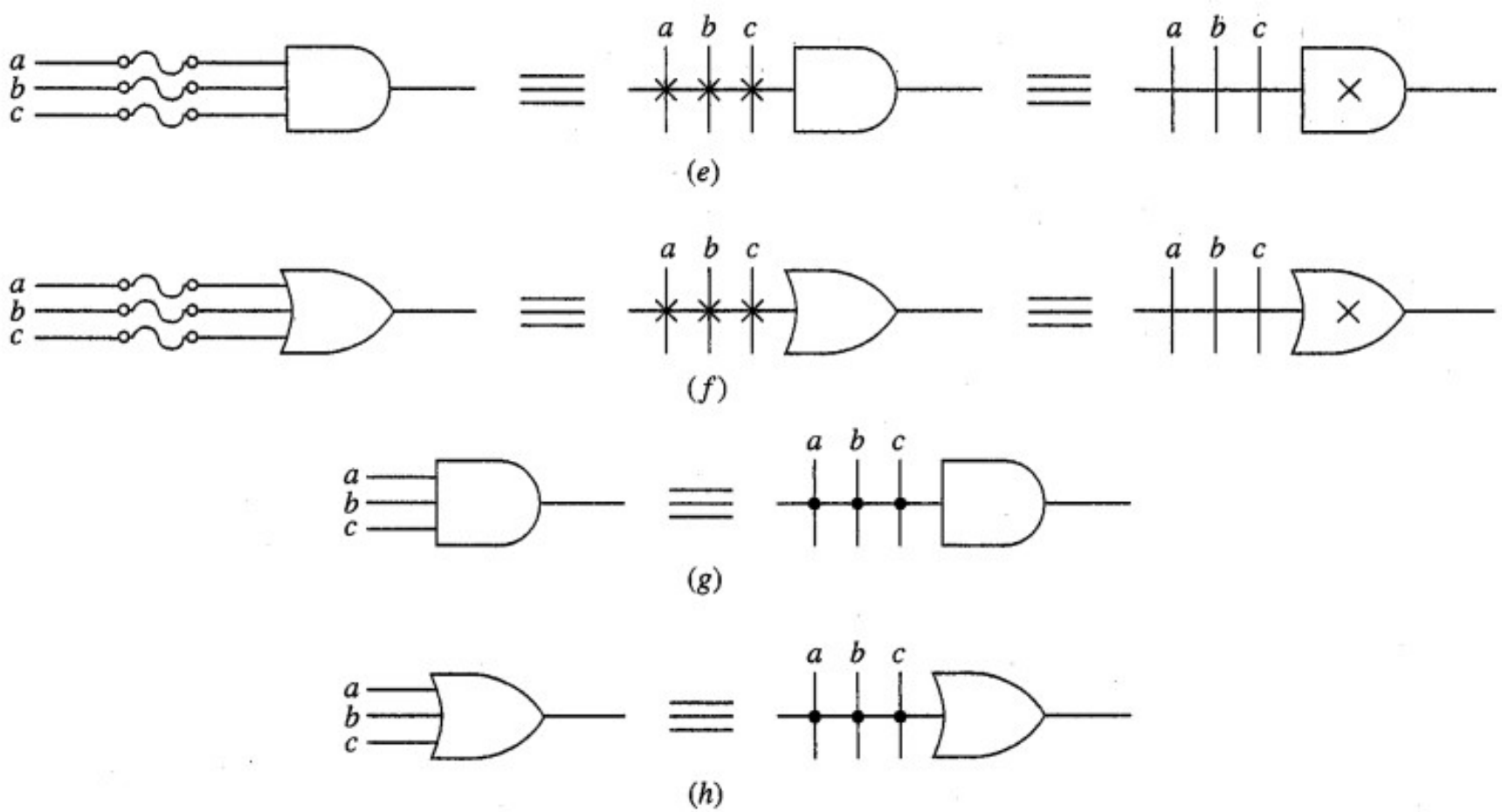
Dispositivi AND-OR

- I primi dispositivi programmabili hanno architettura basata su forme AND-OR
  - Semplicità di progettazione
  - Regolarità della realizzazione
- Normalmente divisi in due regioni
  - Piano AND in cui sono calcolati i termini prodotto
  - Piano OR in cui i termini selezionati sono sommati
- Programmabilità
  - Può coinvolgere la scelta delle variabili che compongono ciascun termine (piano AND)
  - Può riguardare la scelta dei termini da sommare (piano OR)

Convenzioni grafiche (1)



Convenzioni grafiche (2)



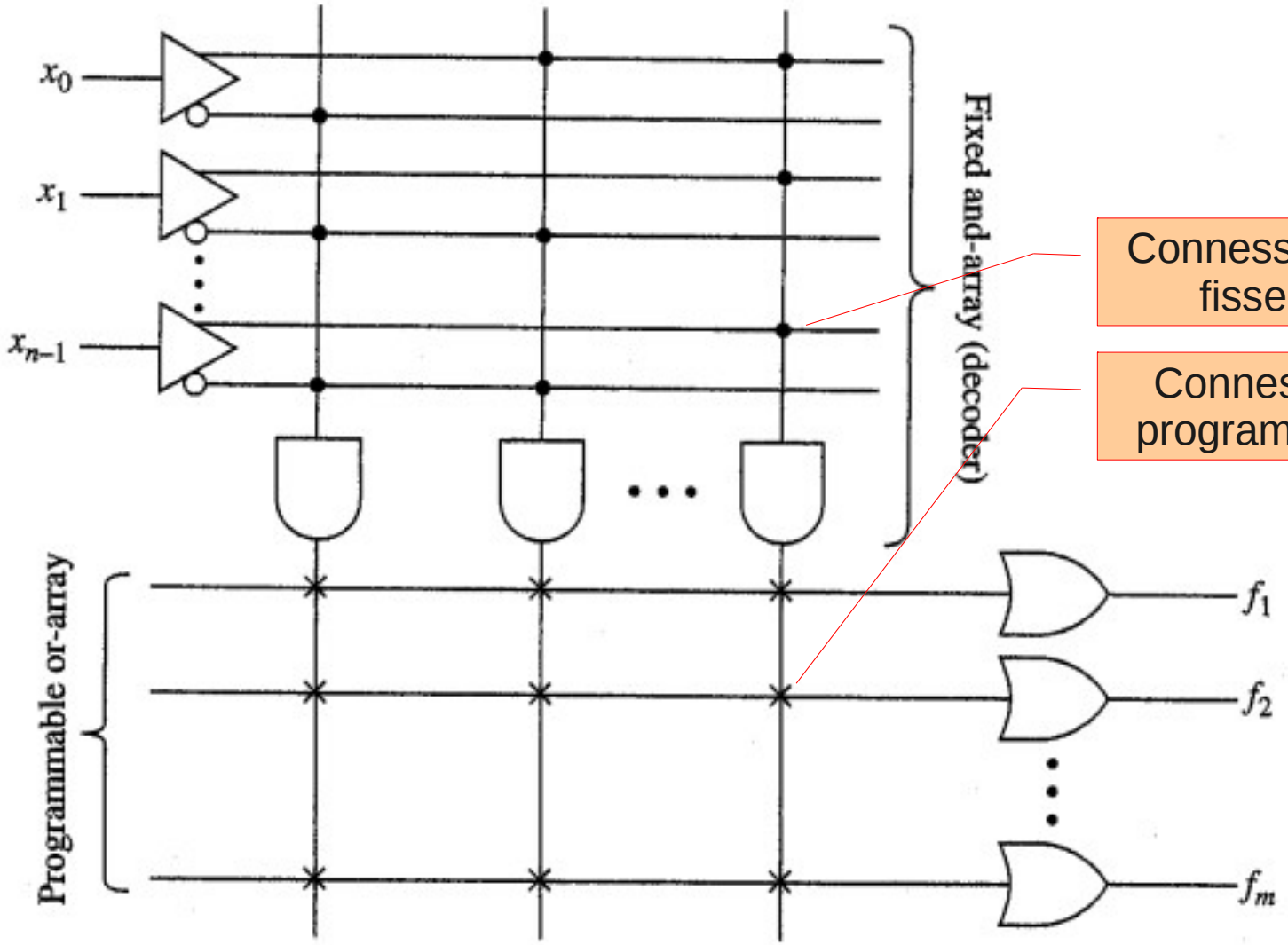
Memorie programmabili

- PROM: programmable read only memory
  - Piano AND fisso e completo
    - Per n ingressi sono calcolati tutti i 2^n mintermini (righe)
  - Piano OR programmabile
    - Ognuna delle uscite genera una somma di prodotti arbitrari
    - Se le uscite sono w i fusibili sono $2^n \times w$
    - Si parla di PROM $2^n \times w$
- La tabella di verità della funzione coincide con il contenuto della memoria

PROM

Mintermine 10...1

Mintermine 11...1



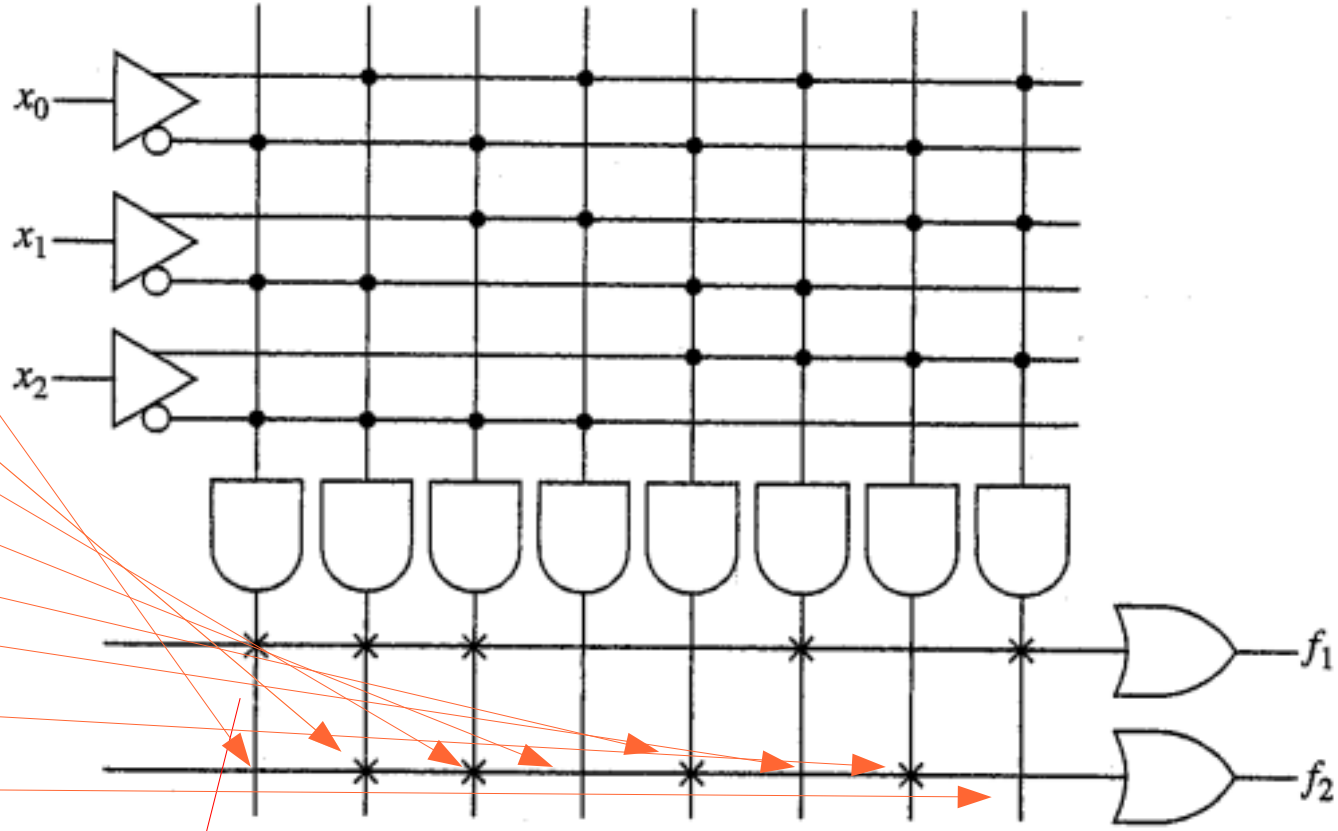
Connessioni fisse

Connessioni programmabili

Condizione della PROM vergine

PROM per logiche generiche

x_2	x_1	x_0	f_1	f_2
0	0	0	1	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	0
1	0	0	0	1
1	0	1	1	0
1	1	0	0	1
1	1	1	1	0

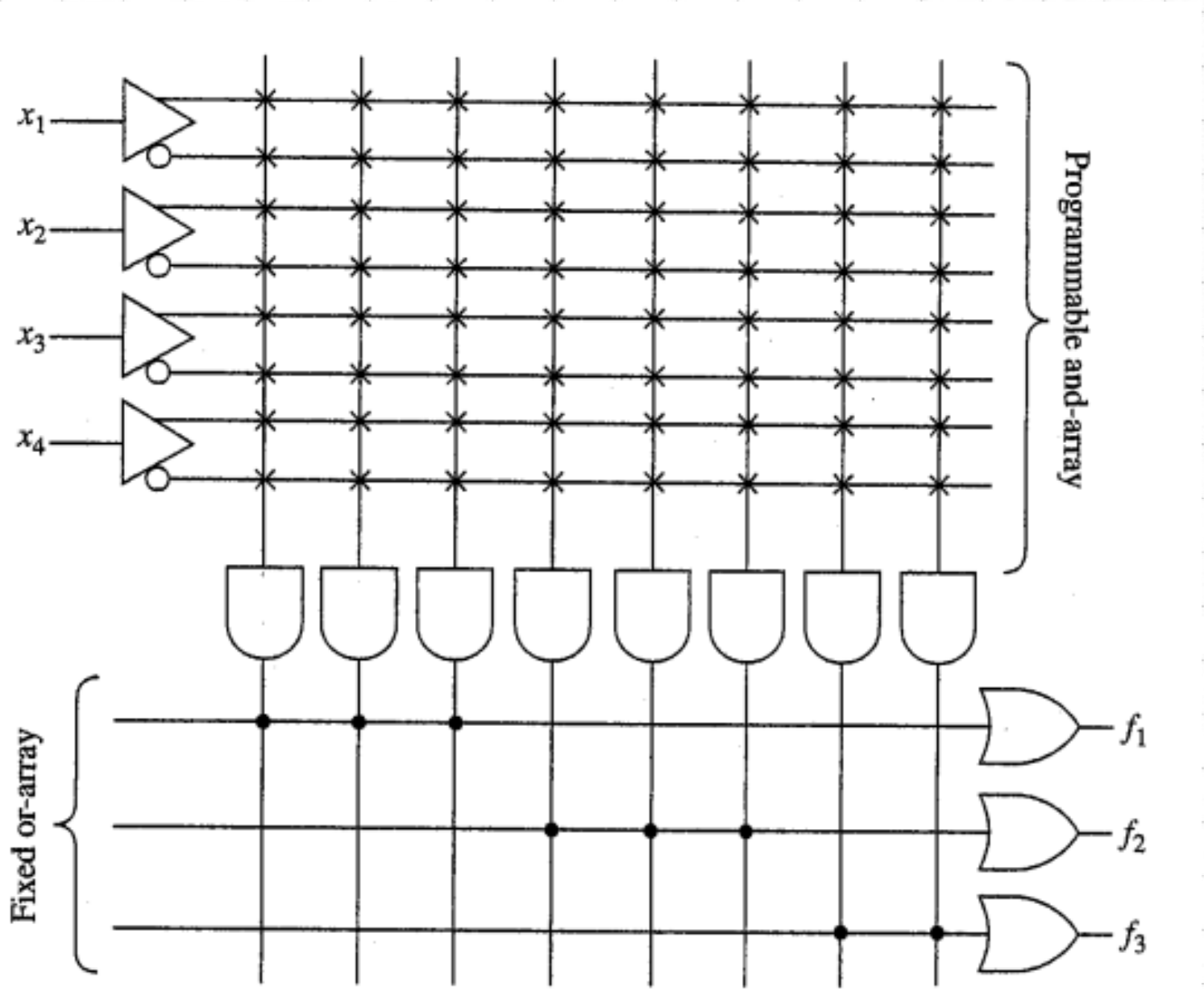


Il contenuto della PROM corrisponde alla tabella di verità

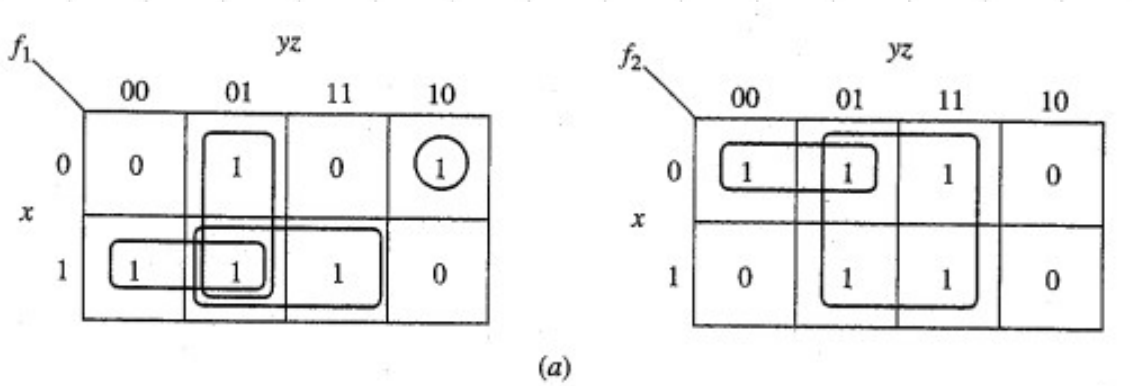
Array logici programmabili

- PAL: Programmable array logic
  - Piano AND programmabile
    - Sono messe a disposizione tutte le variabili e le variabili negate
    - Ogni termine non può essere costituito da un numero di variabili maggiore degli ingressi delle AND
  - Piano OR fisso
    - Sono presenti un certo numero di porte OR, con un numero fisso di ingressi
    - Se una funzione ha meno implicanti, alcuni ingressi della OR possono essere posti facilmente a 0

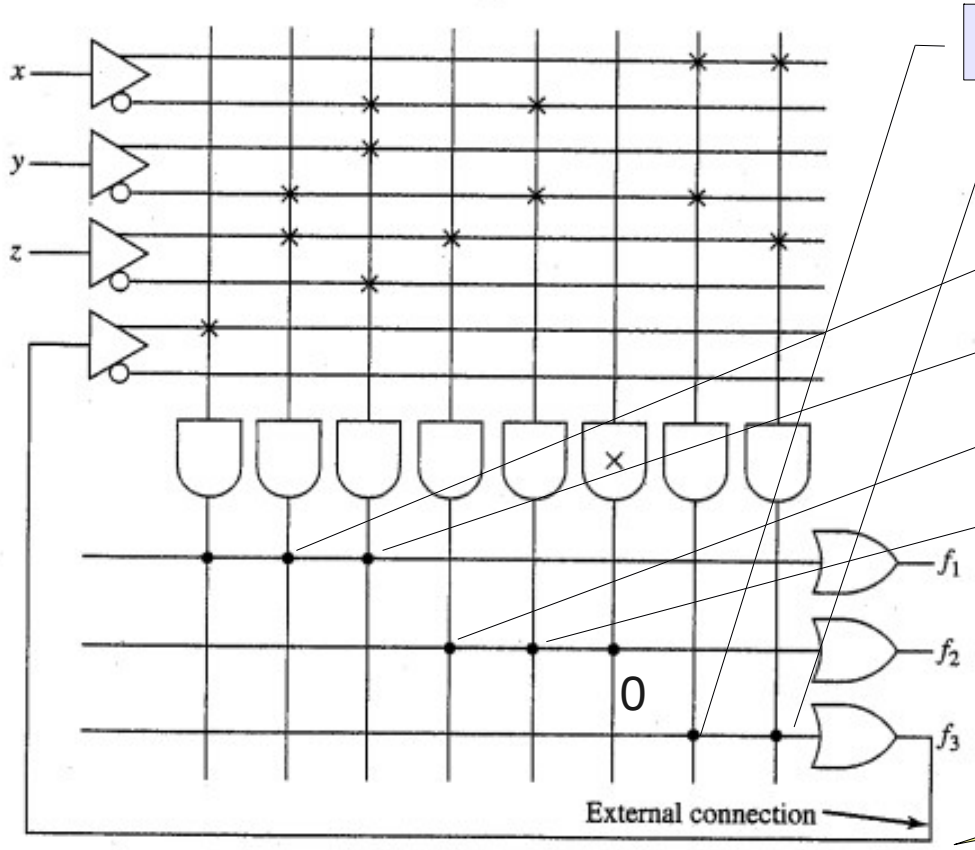
PAL (1)



PAL (2)



(a)



(b)

Esempio di implementazione di due funzioni da 3 ingressi

Trucco per sommare più implicanti di quelli previsti