

## Codificatore, Decodificatore, ROM, PLA, Multiplexer e Demultiplexer

Prof. Daniele Gorla

### Codificatore (4-a-2)

**Input:** 4 linee di ingresso, di cui solo una alla volta può essere a "1"

**Output:** 2 linee di uscita che portano la codifica binaria del numero della linea di ingresso che vale "1"

$x_3$	$x_2$	$x_1$	$x_0$	$y_1$	$y_0$
0	0	0	0	-	-
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	-	-
0	1	0	0	1	0
0	1	0	1	-	-
0	1	1	0	-	-
0	1	1	1	-	-
1	0	0	0	1	1
1	0	0	1	-	-
1	0	1	0	-	-
1	0	1	1	-	-
1	1	0	0	-	-
1	1	0	1	-	-
1	1	1	0	-	-
1	1	1	1	-	-

$x_3$	$x_2$	00	01	11	10
$x_1$	$x_0$	-	-	-	-
00	-	1	-	-	1
01	0	-	-	-	-
11	-	-	-	-	-
10	0	-	-	-	-

$$y_1 = x_3 + x_2$$
  

$x_3$	$x_2$	00	01	11	10
$x_1$	$x_0$	-	0	-	1
00	-	-	0	-	-
01	0	-	-	-	-
11	-	-	-	-	-
10	1	-	-	-	-

$$y_0 = x_3 + x_1$$
  

Matrice di OR:

$x_3$	$x_2$	$x_1$	$x_0$	$y_1$	$y_0$
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	0	0
0	1	0	0	1	0
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	1	0
1	0	0	0	1	1
1	0	0	1	1	1
1	0	1	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	1	1

### Codificatore 2<sup>n</sup>-a-n Codificatore generalizzato

Codificatore 8-a-3:

$x_7$	$x_6$	$x_5$	$x_4$	$x_3$	$x_2$	$x_1$	$x_0$	$y_2$	$y_1$	$y_0$
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	1
0	0	1	0	0	1	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0	1
0	0	1	0	1	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	1
0	0	1	0	1	0	1	0	0	0	0
0	0	1	0	1	0	1	0	0	0	1
0	0	1	0	1	1	0	0	0	0	0
0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	1	0	0	0	0
0	0	1	0	1	1	1	0	0	0	1
0	0	1	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	1
0	0	1	1	0	0	1	0	0	0	0
0	0	1	1	0	0	1	0	0	0	1
0	0	1	1	0	1	0	0	0	0	0
0	0	1	1	0	1	0	0	0	0	1
0	0	1	1	0	1	1	0	0	0	0
0	0	1	1	0	1	1	0	0	0	1
0	0	1	1	1	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	1
0	0	1	1	1	0	1	0	0	0	0
0	0	1	1	1	0	1	0	0	0	1
0	0	1	1	1	1	0	0	0	0	0
0	0	1	1	1	1	0	0	0	0	1
0	0	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	1	0	0	0	1
0	1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	1	0	0	0	0
0	1	0	0	0	0	1	0	0	0	1
0	1	0	0	0	1	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	1
0	1	0	0	1	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0	1
0	1	0	0	1	0	1	0	0	0	0
0	1	0	0	1	0	1	0	0	0	1
0	1	0	0	1	1	0	0	0	0	0
0	1	0	0	1	1	0	0	0	0	1
0	1	0	0	1	1	1	0	0	0	0
0	1	0	0	1	1	1	0	0	0	1
0	1	0	1	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	1
0	1	0	1	0	0	1	0	0	0	0
0	1	0	1	0	0	1	0	0	0	1
0	1	0	1	0	1	0	0	0	0	0
0	1	0	1	0	1	0	0	0	0	1
0	1	0	1	0	1	1	0	0	0	0
0	1	0	1	0	1	1	0	0	0	1
0	1	0	1	1	0	0	0	0	0	0
0	1	0	1	1	0	0	0	0	0	1
0	1	0	1	1	0	1	0	0	0	0
0	1	0	1	1	0	1	0	0	0	1
0	1	0	1	1	1	0	0	0	0	0
0	1	0	1	1	1	0	0	0	0	1
0	1	0	1	1	1	1	0	0	0	0
0	1	0	1	1	1	1	0	0	0	1
0	1	1	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1
0	1	1	0	0	0	1	0	0	0	0
0	1	1	0	0	0	1	0	0	0	1
0	1	1	0	0	1	0	0	0	0	0
0	1	1	0	0	1	0	0	0	0	1
0	1	1	0	0	1	1	0	0	0	0
0	1	1	0	0	1	1	0	0	0	1
0	1	1	0	1	0	0	0	0	0	0
0	1	1	0	1	0	0	0	0	0	1
0	1	1	0	1	0	1	0	0	0	0
0	1	1	0	1	0	1	0	0	0	1
0	1	1	0	1	1	0	0	0	0	0
0	1	1	0	1	1	0	0	0	0	1
0	1	1	0	1	1	1	0	0	0	0
0	1	1	0	1	1	1	0	0	0	1
0	1	1	1	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	1
0	1	1	1	0	0	1	0	0	0	0
0	1	1	1	0	0	1	0	0	0	1
0	1	1	1	0	1	0	0	0	0	0
0	1	1	1	0	1	0	0	0	0	1
0	1	1	1	0	1	1	0	0	0	0
0	1	1	1	0	1	1	0	0	0	1
0	1	1	1	1	0	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0	1
0	1	1	1	1	0	1	0	0	0	0
0	1	1	1	1	0	1	0	0	0	1
0	1	1	1	1	1	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0	1
0	1	1	1	1	1	1	0	0	0	0
0	1	1	1	1	1	1	0	0	0	1
0	1	1	1	1	1	1	1	0	0	0
0	1	1	1	1	1	1	1	0	0	1
0	1	1	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	1

#### Codificatore 2<sup>n</sup>-a-n:

la linea verticale associata a  $x_i$  contiene la codifica binaria di  $i$ , vedendo il pallino come 1 e l'assenza di pallino come 0.

#### Codificatore generalizzato

Input:  $n$  linee, di cui solo una alla volta vale "1"

Output: la codifica binaria (in  $m$  bit) di  $f(i)$ , per  $f$  fissata e  $i$  la linea che vale 1

Es.:

$f$	$x_3$	$x_2$	$x_1$	$x_0$	$y_4$	$y_3$	$y_2$	$y_1$	$y_0$
0	0	0	0	1	1	0	0	0	1
1	0	0	1	0	0	0	1	0	0
2	0	1	0	0	1	0	0	0	1
3	1	0	0	0	1	1	1	1	1

$x_3$	$x_2$	$x_1$	$x_0$	$y_4$	$y_3$	$y_2$	$y_1$	$y_0$
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0
0	1	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0
1	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0
1	1	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0

### Decodificatore

## ROM

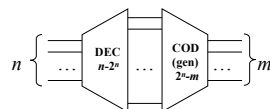


Una ROM (Read Only Memory) è un dispositivo con  $n$  ingressi (dette *linee di indirizzamento*) ed  $m$  uscite (dette *linee dati*).

All'interno del dispositivo, le linee di indirizzamento selezionano una fra  $2^n$  righe di una matrice  $2^n \times m$ .

La selezione della riga  $i$ -esima della matrice consente di leggere, su ciascuna delle  $m$  linee dati, il valore binario stabilmente memorizzato nella cella di coordinate  $(i, j)$ , per  $j \in \{1, \dots, m\}$ .

E' la composizione di un decodificatore e un codificatore generalizzato:



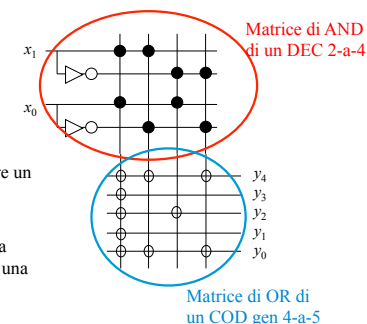
cioè di una matrice di AND (le cui entrate sono le linee di indirizzamento e le cui uscite sono le entrate di una matrice di OR) e di una matrice di OR (le cui uscite sono le linee dati).

5

## Realizzazione di FB tramite ROM



$x_1$	$x_0$	$y_4$	$y_3$	$y_2$	$y_1$	$y_0$
0	0	1	0	0	0	1
0	1	0	0	1	0	0
1	0	1	0	0	0	1
1	1	1	1	1	1	1



Una ROM può essere usata per realizzare un insieme di  $m$  funzioni booleane

$$\{0,1\}^n \rightarrow \{0,1\}$$

Uso un decodificatore  $n$ -a- $2^n$  e "copio" la parte destra della tabella di verità (che è una matrice  $2^n \times m$ ) nella matrice di OR di un codificatore generalizzato  $2^n$ -a- $m$ .

Sull'uscita associata al mintermine  $m_i$  del decodificatore (matrice AND) metto un  $\square$  in corrispondenza di ogni 1 presente sulla riga  $i$  della TV.

Fisicamente, è un diodo che trasferisce un segnale dall'uscita del DEC alla corrispondente linea dati solo se il segnale è 1.

6

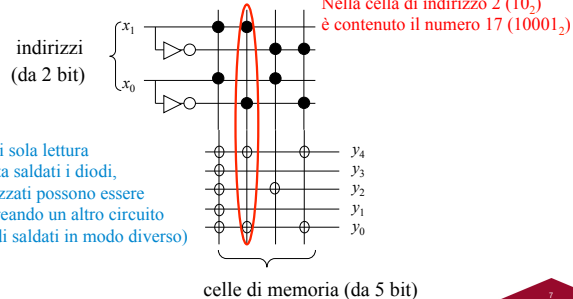
## ROM come memoria di sola lettura



Il nome ROM (read-only memory) deriva dal fatto che questo modulo combinatorio può essere visto come

- una memoria (cioè un insieme di celle di dimensione fissata, ognuna con un suo proprio indirizzo)
- non riscrivibile (quindi, di sola lettura)

Es.:



La memoria è di sola lettura perché, una volta saldati i diodi, i valori memorizzati possono essere cambiati solo creando un altro circuito (cioè, con i diodi saldati in modo diverso)

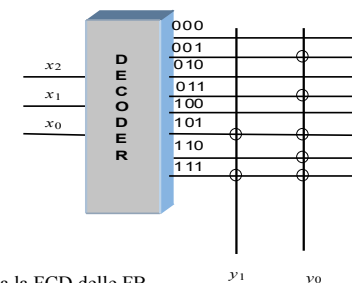
7

## ROM con DECODER



**N.B.:** spesso non si esplicita il decodificatore e lo si lascia indicato come modulo noto. Così bisogna solo riempire le righe della matrice di OR, che viene quindi chiamata *matrice della ROM*.

$x_2$	$x_1$	$x_0$	$y_1$	$y_0$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	1
1	1	1	1	1



**OSS.1:** una ROM realizza la FCD delle FB

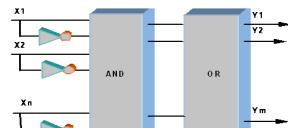
**OSS.2:** alcuni mintermini vengono calcolati nel DEC ma mai usati nella matrice della ROM (Es.:  $m_0$ ,  $m_2$  ed  $m_4$ )

8

## PLA



Un PLA (Programmable Logic Array) è una rete combinatoria integrata con  $n$  ingressi,  $m$  uscite e tre stadi interni: uno stadio di inversione dei segnali di ingresso, una matrice di AND ed una matrice di OR.



→ come una ROM

Un PLA consente di implementare espressioni booleane in forma FND, in particolare EB in FND minime

→ più efficiente ed economico di una ROM

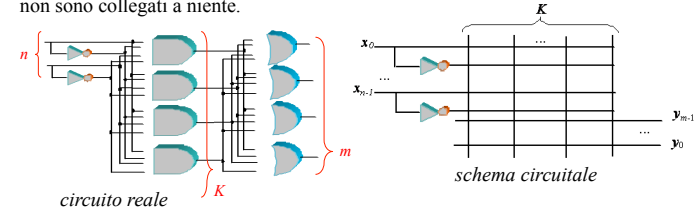
Come? Anche la matrice di AND (e non solo quella di OR) è personalizzabile

9

## Programmare un PLA



Quindi un PLA viene venduto con tutti gli  $n$  ingressi sia affermati che negati; inoltre, avrà una serie di  $K$  porte AND e  $m$  porte OR i cui ingressi non sono collegati a niente.



I collegamenti vengono effettuati sulla base delle specifiche fornite dal committente, formulate in termini di funzioni in FND.

L'utente deve quindi fornire la TV, da cui si ricaveranno le FND minime per ognuna delle  $m$  FB (più laborioso rispetto a una ROM!!)

10

## Esempio

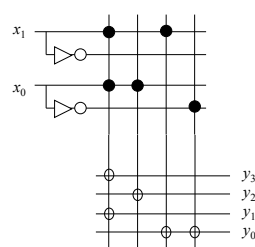
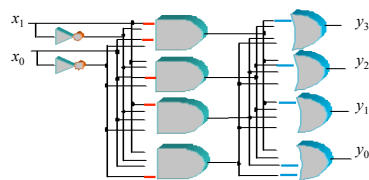


$x_1 x_0$	$y_3 y_2 y_1 y_0$
0 0	0 0 0 1
0 1	0 1 0 0
1 0	0 0 0 1
1 1	1 1 1 1

$$y_3 = y_1 = x_1 x_0$$

$$y_2 = x_0$$

$$y_0 = x_1 + \bar{x}_0$$



11

## Multiplexer in senso stretto



**Input:**  $n$  linee dati e  $n$  linee di controllo, di cui al più una vale "1" ad ogni istante

**Output:** 1 linea che prende il valore della linea dati  $i$ -esima, se l' $i$ -esima linea di controllo vale 1.

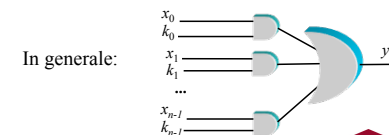
Es. ( $n = 2$ )

$x_1 x_0 k_1 k_0$	$y$
0 0 0 1	0
0 1 0 1	1
1 0 0 1	0
1 1 0 1	1
0 0 1 0	0
0 1 1 0	0
1 0 1 0	1
1 1 1 0	1

$x_1 x_0$	00	01	11	10
$k_1 k_0$	00	01	11	10
00	-	1	1	0
01	0	-	-	-
11	-	0	-	1
10	0	0	1	1

$$y = x_1 k_1 + x_0 k_0$$

N.B.: le righe con  $k_1 = k_0$  sono dei don't care



12

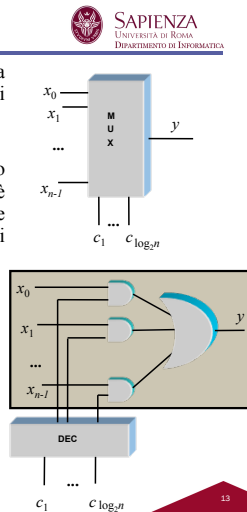
## Multiplexer

Un **multiplexer** (MUX) è una rete combinatoria con  $n$  ingressi, una uscita e  $\log_2 n$  segnali di controllo:

In ogni istante, l'uscita  $y$  è uguale al valore di uno ed uno solo degli ingressi,  $x_i$ . Il valore di  $i$  è determinato dai segnali di controllo: è il valore (espresso come numero naturale) che i segnali codificano in binario.

Un MUX è costituito da una porta OR, che riceve le uscite di  $n$  porte AND, le quali funzionano da interruttori; infine, un decodificatore attiva l'interruttore selezionato dai segnali di controllo:

N.B.: è un MUX in senso stretto i cui segnali di controllo sono le uscite del DEC.



## Demultiplexer in senso stretto

**Input:** 1 linea dati e  $n$  linee di controllo, di cui al più una vale "1" ad ogni istante

**Output:**  $n$  linee, di cui la  $i$ -esima prende il valore della linea dati se l' $i$ -esima linea di controllo vale 1.

Es. ( $n = 2$ )

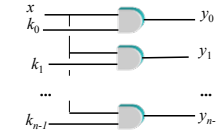
$x$	$k_1$	$k_0$	$y_1$	$y_0$
0	0	1	0	0
1	0	1	0	1
0	1	0	0	0
1	1	0	1	0

$$y_1 = xk_1$$

$$y_0 = xk_0$$

In generale:

N.B.: le righe con  $k_1 = k_0$  sono dei don't care



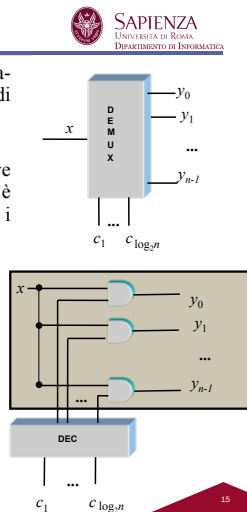
## Demultiplexer

Un **demultiplexer** (DEMUX) è una rete combinatoria con un ingresso,  $n$  uscite e  $\log_2 n$  segnali di controllo:

In ogni istante, l'uscita  $y_i$  è uguale all'ingresso, dove il valore di  $i$  è determinato dai segnali di controllo: è il valore (espresso come numero naturale) che i segnali codificano in binario.

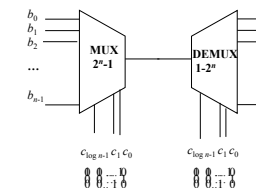
Un DEMUX è costituito da  $n$  porte AND, le quali funzionano da interruttori, e da un decodificatore che attiva l'interruttore selezionato dai segnali di controllo:

N.B.: è un DEMUX in senso stretto i cui segnali di controllo sono le uscite del DEC.



## Utilizzo di MUX e DEMUX

- Conversione parallelo/seriale (MUX) e seriale/parallelo (DEMUX) → a intervalli prestabiliti incremento le linee di controllo:



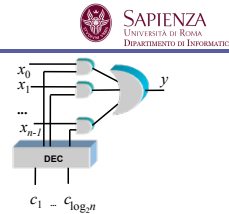
N.B.: per fare questo abbiamo bisogno di un particolare circuito (detto *contatore*) che vedremo alla fine del corso

- Usare MUX per calcolare FB

### MUX per realizzare FB (1)

Dalla def. di MUX, abbiamo che

$$y = \sum_{i=0}^{2^n-1} x_i \cdot m_i = \sum_{i: x_i=1} m_i$$



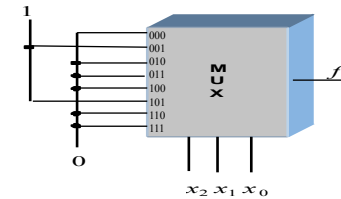
Ricordiamo che una FB di  $n$  variabili in FCD è  $f = \sum_{i: f(i_2)=1} m_i$

Data una FB di  $n$  variabili,

- si utilizza un MUX con  $2^n$  con entrate;
- gli  $n$  segnali di controllo del MUX sono le  $n$  variabili;
- i  $2^n$  ingressi dati vengono individualmente cablati al valore "0" o "1" secondo quanto specificato dalla TV.

### Esempio

$x_2$	$x_1$	$x_0$	$f$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



### MUX per realizzare FB (2)

Per realizzare una FB a  $n$  variabili, posso anche usare un MUX con meno di  $n$  linee di controllo; in questo caso, alcune variabili saranno linee di controllo mentre altre saranno usate in EB sulle linee dati (che, quindi non saranno più semplicemente 0 o 1).

Es. ( $n = 3$ ):  $f = \sum_{i=0}^7 f(i_2) \cdot m_i =$

$$\begin{aligned} &= f(000) \cdot \bar{x} \cdot \bar{y} \cdot \bar{z} + f(001) \cdot \bar{x} \cdot \bar{y} \cdot z + f(010) \cdot \bar{x} \cdot y \cdot \bar{z} + f(011) \cdot \bar{x} \cdot y \cdot z + \\ &+ f(100) \cdot x \cdot \bar{y} \cdot \bar{z} + f(101) \cdot x \cdot \bar{y} \cdot z + f(110) \cdot x \cdot y \cdot \bar{z} + f(111) \cdot x \cdot y \cdot z = \\ &= (f(000) \cdot \bar{z} + f(001) \cdot z) \cdot \bar{x} \cdot \bar{y} + (f(010) \cdot \bar{z} + f(011) \cdot z) \cdot \bar{x} \cdot y + \\ &+ (f(100) \cdot \bar{z} + f(101) \cdot z) \cdot x \cdot \bar{y} + (f(110) \cdot \bar{z} + f(111) \cdot z) \cdot x \cdot y \end{aligned}$$

