Architettura degli Elaboratori

Moduli combinatori

Barbara Masucci
UNIVERSITA DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA
DIPARTIMENTO DI ECCELLENZA

Punto della situazione

- Abbiamo studiato le reti logiche e la loro minimizzazione

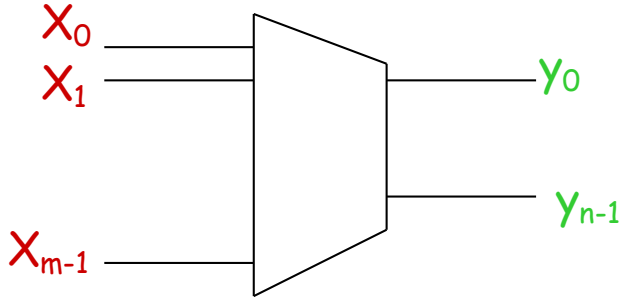
- Obiettivo di oggi: studio dei moduli combinatori di base utilizzati nei calcolatori
  - Codificatore
  - Decodificatore
  - Multiplexer
  - Demultiplexer
  - PLA (Programmable Logic Array)
  - ROM (Read Only Memory)

Circuiti integrati

- I moduli combinatori di base sono realizzati come circuiti integrati
  - Realizzati su chip di silicio (piastrine)
  - Porte (gate) e fili depositati su piastrine, inseriti in un package e collegati all'esterno con un certo insieme di pin (piedini)
- Esistono circuiti integrati per singole porte logiche e circuiti più complessi

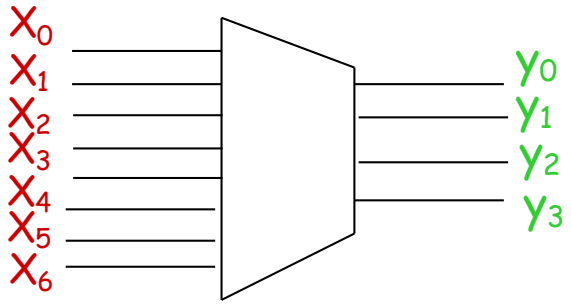
Codificatore

- Componente con
  - m input $x_0,..,x_{m-1}$
  - n output $y_0,..,y_{n-1}$, dove $2^n \geq m$
  - Ogni input è interpretato come elemento di un insieme di m simboli
  - Tra le linee di input, solo una è attiva istante per istante: quando l'input è il simbolo i-esimo, $x_i=1$ e $x_j=0$ per ogni $j \neq i$
- Associa ad ogni elemento di input la sequenza di n bit corrispondente alla sua rappresentazione binaria
  - Quando l'input è il simbolo i-esimo, in uscita è presente il codice binario corrispondente



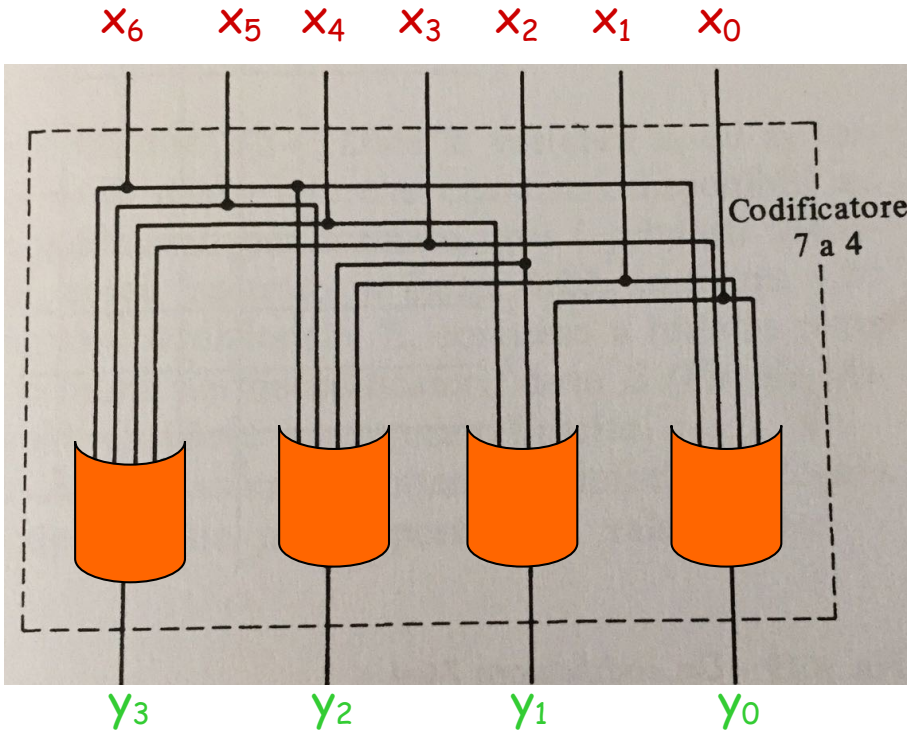
Codificatore

- Supponiamo di avere m=7 linee in ingresso e n=4 linee in uscita



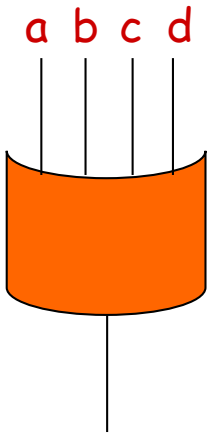
		y_3	y_2	y_1	y_0
x_0	3	0	0	1	1
x_1	5	0	1	0	1
x_2	6	0	1	1	0
x_3	9	1	0	0	1
x_4	10	1	0	1	0
x_5	12	1	1	0	0
x_6	13	1	1	0	1

Un codificatore è un insieme di porte OR

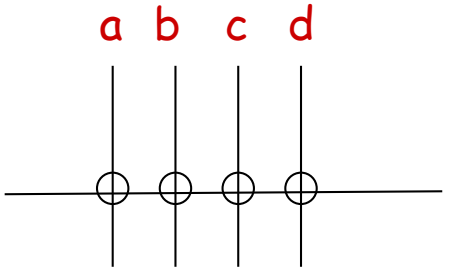


Griglie di porte OR

Spesso viene utilizzato un simbolo grafico alternativo per le porte OR



è equivalente a

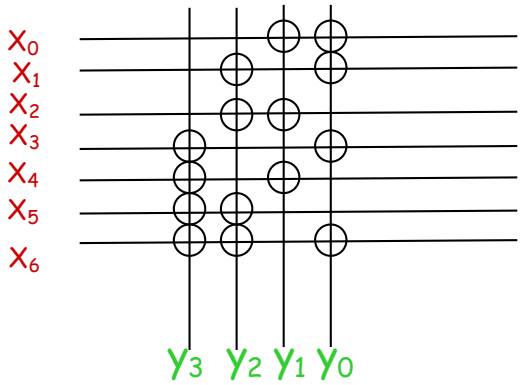


Tale rappresentazione alternativa è particolarmente utile per illustrare i codificatori

Codificatore

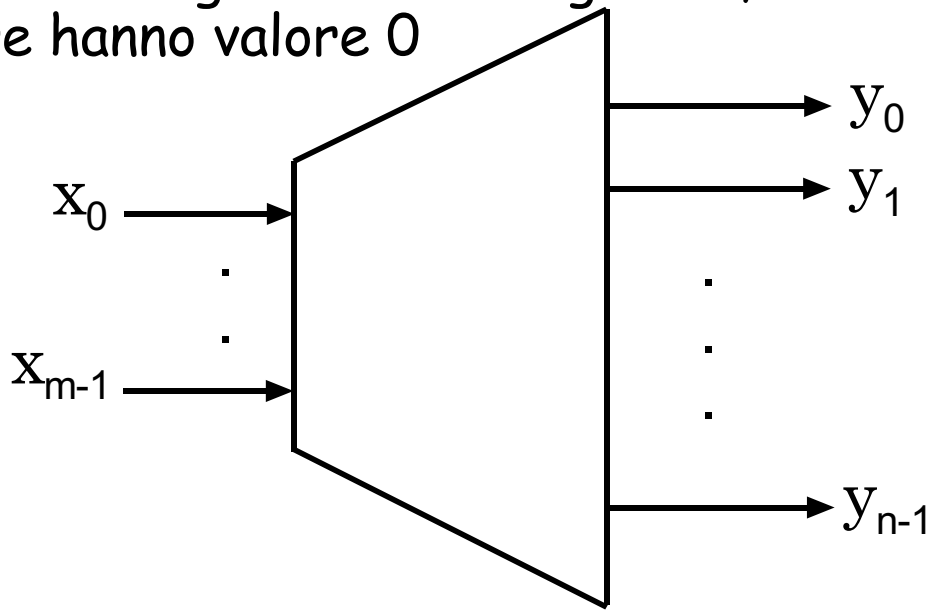
Con la rappresentazione alternativa per le porte OR, il codificatore può essere rappresentato come una griglia

		y_3	y_2	y_1	y_0
x_0	3	0	0	1	1
x_1	5	0	1	0	1
x_2	6	0	1	1	0
x_3	9	1	0	0	1
x_4	10	1	0	1	0
x_5	12	1	1	0	0
x_6	13	1	1	0	1

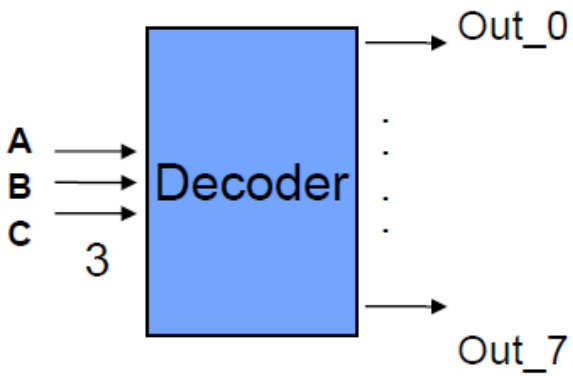


Decodificatore

- Componente con
  - m input $x_0,..,x_{m-1}$
  - n output $y_0,..,y_{n-1}$, dove $2^m=n$
- Realizza la funzione inversa del codificatore
  - A partire da una sequenza di bit genera in output il simbolo corrispondente
  - Per ogni configurazione di ingresso, una sola uscita vale 1, le altre hanno valore 0

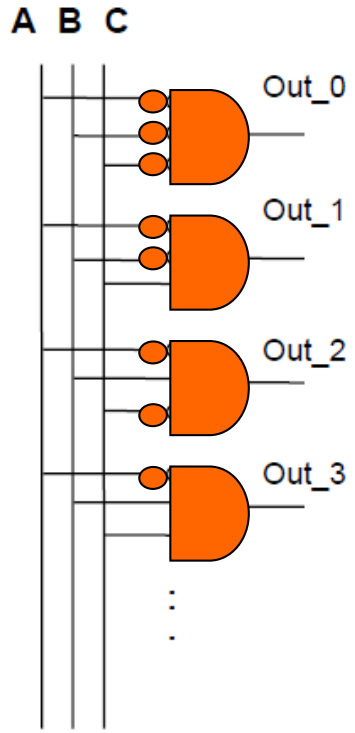


Decodificatore



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

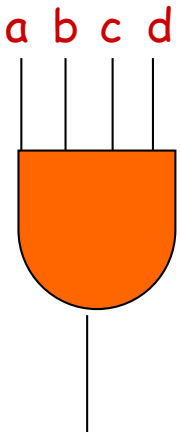
Tavola di verità



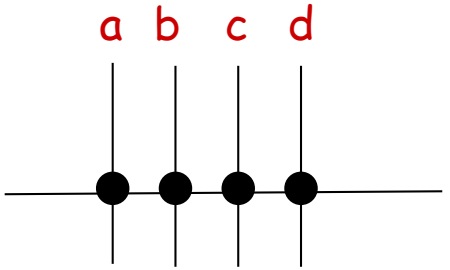
Un decodificatore è un insieme di porte AND

Griglie di porte AND

Anche per le porte AND esiste un simbolo grafico alternativo



è equivalente a

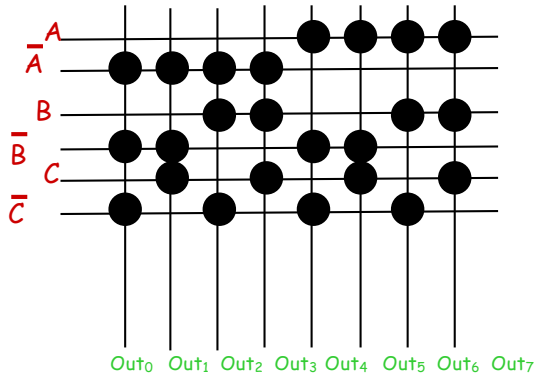


Tale rappresentazione alternativa è particolarmente utile per illustrare i decodificatori

Decodificatore

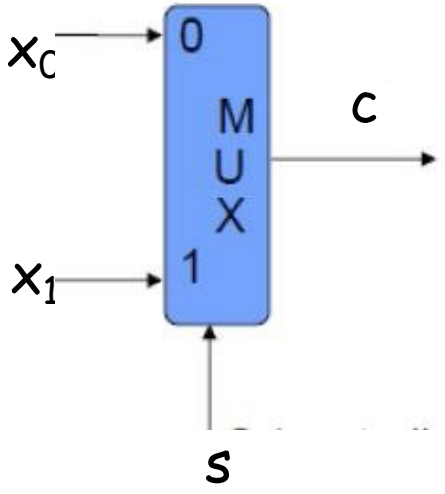
Con la rappresentazione alternativa per le porte AND, il decodificatore può essere rappresentato come una griglia

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



Multiplexer (MUX 2^n:1)

- Componente con
  - m linee di input dati $x_0,..,x_{m-1}$
  - n linee di selezione $s_0,..,s_{n-1}$, dove $n=\log_2 m$
  - una linea di output c
- Seleziona, in base ai segnali di selezione, quale degli input verrà fornito in output
- Esempio: Multiplexer 2:1
  - Se s=0 passa x_0
  - Se s=1 passa x_1



x_0	x_1	s	c
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Tavola di verità

Multiplexer 2:1

Troviamo i mintermini corrispondenti alle occorrenze di 1 nella tavola di verità e sommiamoli

x_0	x_1	s	c	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\overline{x_0}\cdot x_1\cdot s$
1	0	0	1	$x_0\cdot\overline{x_1}\cdot\overline{s}$
1	0	1	0	
1	1	0	1	$x_0\cdot x_1\cdot\overline{s}$
1	1	1	1	$x_0\cdot x_1\cdot s$

Tavola di verità

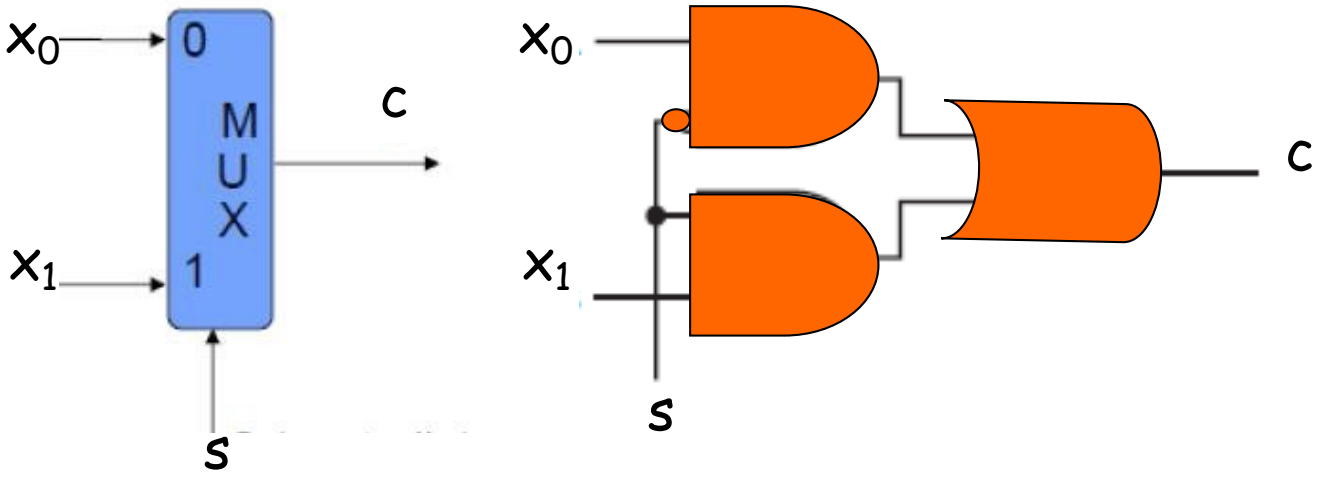
Espressione canonica SOP

$$c = \overline{x_0}\cdot x_1\cdot s + x_0\cdot\overline{x_1}\cdot\overline{s} + x_0\cdot x_1\cdot\overline{s} + x_0\cdot x_1\cdot s$$
$$= (\overline{x_0} + x_0)\cdot x_1\cdot s + (\overline{x_1} + x_1)\cdot x_0\cdot\overline{s}$$
$$= x_1\cdot s + x_0\cdot\overline{s}$$

Espressione minimale

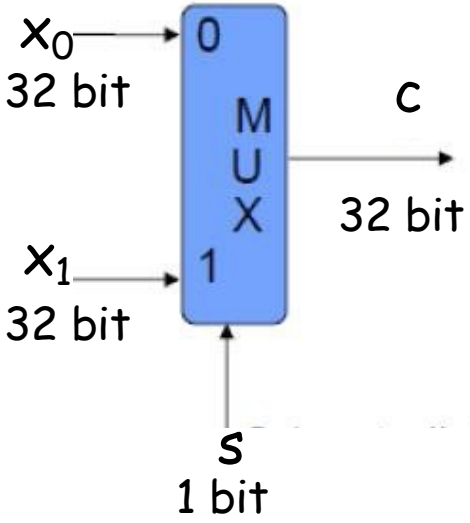
Multiplexer 2:1

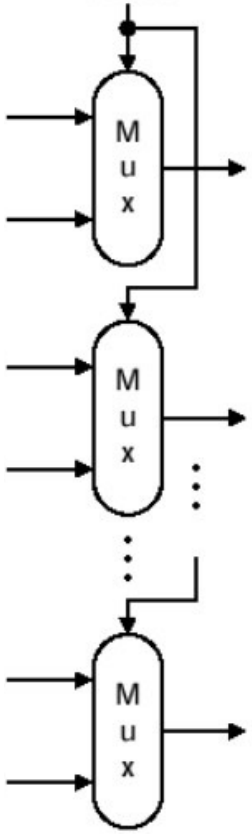
$$c = x_0\overline{s} + x_1 s$$



Multiplexer 2:1 a 32 bit

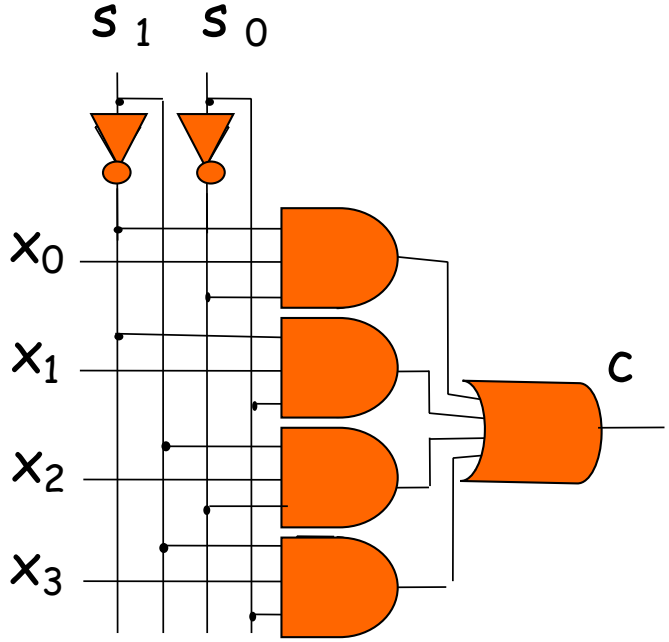
- Costruito con 32 multiplexer 2:1 con un segnale di controllo distribuito ai vari multiplexer





Multiplexer 4:1

- Componente con
  - 4 input dati: x_0, x_1, x_2, x_3
  - 2 segnali di controllo: s_1, s_0
  - 1 output: c
- Comportamento:
  - Se s_1s_0=00 allora $c = x_0$
  - Se s_1s_0=01 allora $c = x_1$
  - Se s_1s_0=10 allora $c = x_2$
  - Se s_1s_0=11 allora $c = x_3$



$$c = x_0 \, \overline{s_1} \, \overline{s_0} + x_1 \, \overline{s_1} \, s_0 + x_2 \, s_1 \, \overline{s_0} + x_3 \, s_1 \, s_0$$

Circuito con 4 porte AND (a 3 ingressi) e 1 porta OR

Multiplexer e funzioni logiche

- Un multiplexer con n variabili di selezione può calcolare qualsiasi funzione booleana di n variabili, ponendo
  - Le n variabili sulle linee di selezione
  - La tavola di verità della funzione sulle 2^n linee dati

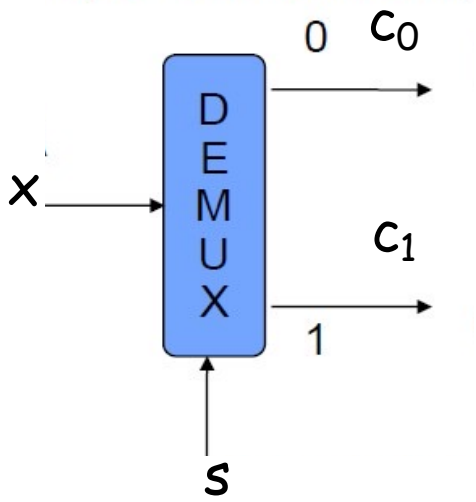
Demultiplexer

- Componente con
  - Una linea di input x
  - m=$\log_2$n linee di selezione $s_0,..,s_{m-1}$
  - n linee di output $c_0,..,c_{n-1}$

- Comportamento
  - Usato per selezionare una linea verso una tra più destinazioni possibili (funzione inversa del multiplexer)
  - Se la linea di input è uguale a 0, tutti gli output sono uguali a 0 (indipendentemente dai segnali di selezione)
  - Se la linea di input è uguale a 1, un solo output è uguale a 1 (dipende dai segnali di selezione)

Demultiplexer con n=2

- Indirizza una linea di input verso due destinazioni possibili

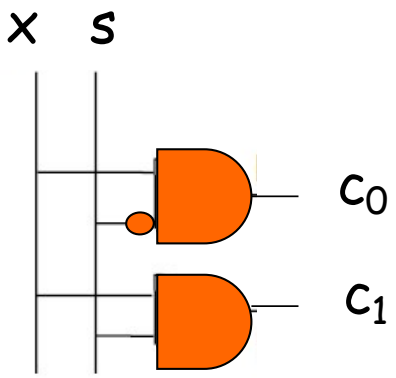


x	s	c_0	c_1
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

Tavola di verità

$c_0 = x \overline{s}$

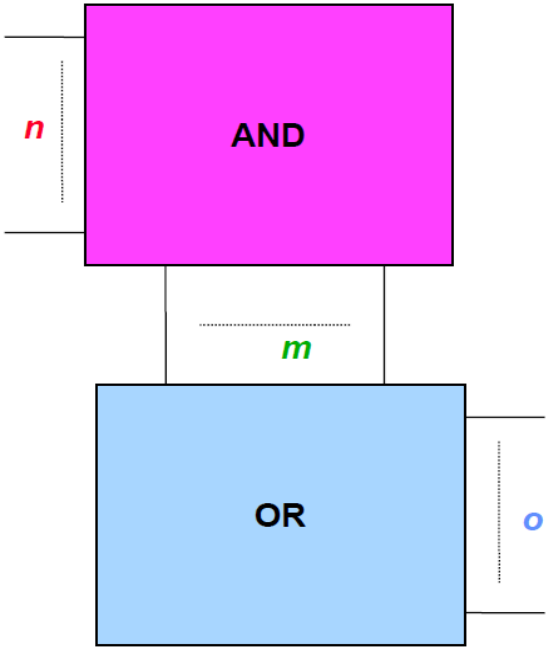
$c_1 = x\ s$



PLA
Programmable Logic Array

- Abbiamo visto i circuiti AND-to-OR per realizzare una funzione logica
- Con un PLA è possibile rappresentare un insieme di funzioni mediante circuiti AND-to-OR
- Un PLA ha
  - n input
  - o output

  ed utilizza
  - m porte AND
  - o porte OR



Esempio PLA

- Si consideri una funzione logica con
  - tre input A, B, C
  - tre output D, E, F

  definita come segue
  - D è uguale a 1 se almeno uno dei tre input è uguale a 1
  - E è uguale a 1 se esattamente due dei tre input sono uguali a 1
  - F è uguale a 1 se e solo se tutti e tre gli input sono uguali a 1

Esempio PLA

La tavola di verità della funzione è la seguente

A	B	C	D	E	F
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	0	1

$$D = \overline{A}\cdot\overline{B}\cdot C + \overline{A}\cdot B\cdot\overline{C} + \overline{A}\cdot B\cdot C + A\cdot\overline{B}\cdot\overline{C} + A\cdot\overline{B}\cdot C + A\cdot B\cdot\overline{C} + A\cdot B\cdot C$$

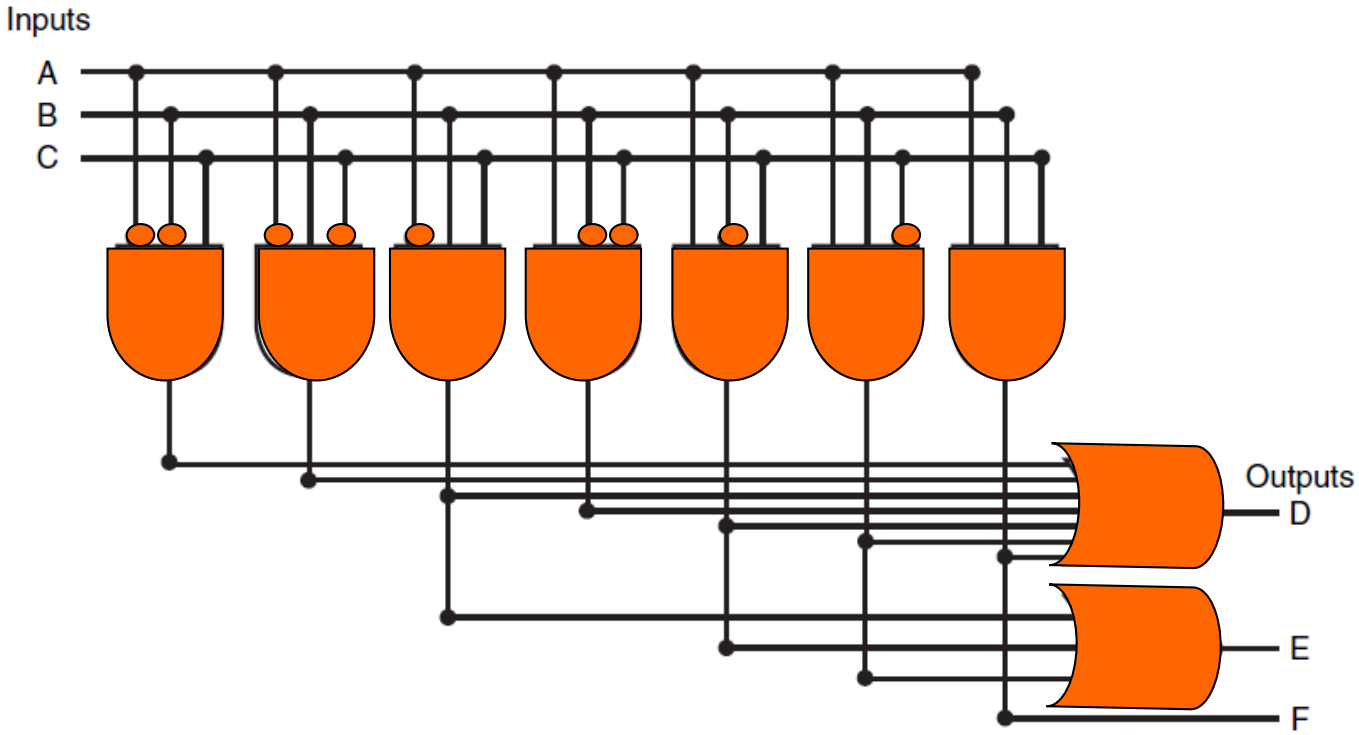
$$E = \overline{A}\cdot B\cdot C + A\cdot\overline{B}\cdot C + A\cdot B\cdot\overline{C}$$

$$F = A\cdot B\cdot C$$

Espressioni canoniche SOP

Esempio PLA

- Otteniamo il circuito AND-to-OR per la funzione

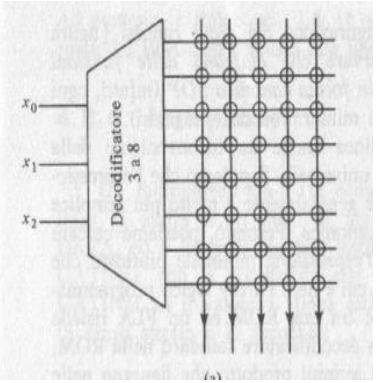


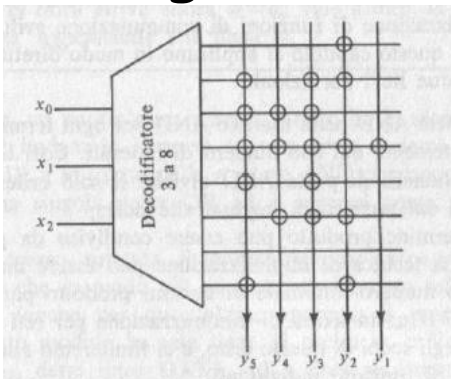
ROM
Read Only Memory

- Componente per memorizzare informazioni che non è necessario modificare
  - Vista come una tabella con heigth righe e width colonne
  - Ciascuna cella della tabella rappresenta una locazione di memoria, il cui contenuto può essere letto
- Diversi tipi di ROM
  - PROM (Programmable ROM): scrivibile una sola volta
  - EPROM (Erasable ROM): cancellabile con luce ultravioletta

ROM
Read Only Memory

- Costituita da un decodificatore legato a una catena di porte OR
  - L'input al decodificatore determina il numero di locazioni di memoria
  - Ciascuna uscita del decodificatore è connessa a ciascuna porta OR
  - Scrivere in una locazione di memoria corrisponde a "rompere" dei collegamenti





Indirizzo			Contenuto				
x_2	x_1	x_0	y_5	y_4	y_3	y_2	y_1
0	0	0	0	0	0	1	0
0	0	1	1	0	1	1	0
0	1	0	1	1	1	0	0
0	1	1	1	1	1	1	1
1	0	0	1	0	1	0	0
1	0	1	0	1	1	1	0
1	1	0	0	0	0	0	0
1	1	1	1	0	0	1	0

ROM
Read Only Memory

- Possiamo usare una ROM per implementare diverse funzioni logiche
  - Ogni colonna della ROM memorizza la tavola di verità di una distinta funzione
  - Un indirizzo a n bit individua una specifica combinazione delle variabili di input
- Soluzione meno efficiente rispetto all'uso di PLA
  - Con PLA, le funzioni logiche possono essere rappresentate in forma minimale

Riepilogo e riferimenti

- Moduli combinatori:
  - [P] par. 4.8
  - [PH] appendice B.3
  - [PH_IIIed] appendice C.3