

# Lezione 11 Circuiti combinatori 3

mercoledì 18 ottobre 2023 16:08

Cominciamo con un esempio : rappresentiamo la funzione f come SOP e POS. Inoltre poi minimizziamo i circuiti :

$$F=ab+c$$

Andiamo ad usare gli implicant per trovare la forma minima : moltiplichiamo per variabile + variabile negata , per le regole dell'algebra booleana (-variabile=negata) :

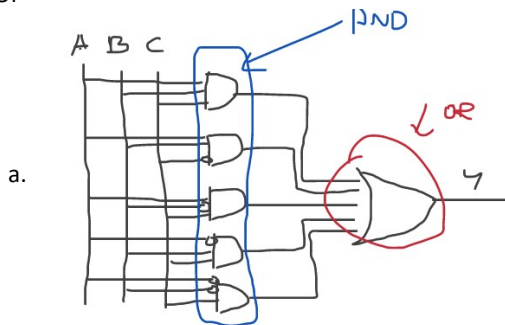
$$F=ab+c= ab(c+(-c))+c(a+(-a))(b+(-b)) = abc+ab(-c)+a(-b)c+(-a)bc+(-a)(-b)+c$$

Da notare che questo passaggio è un più : in quanto potevamo direttamente ricavare l'uscita dalla funzione . Vediamo il circuito :

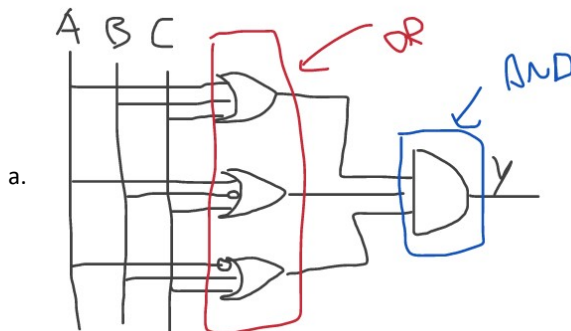
a	b	c	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Che a seconda se POS o SOP diventa il seguente :

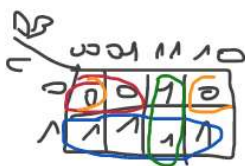
1. SOP



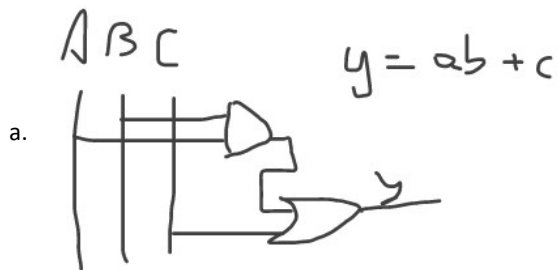
2. POS



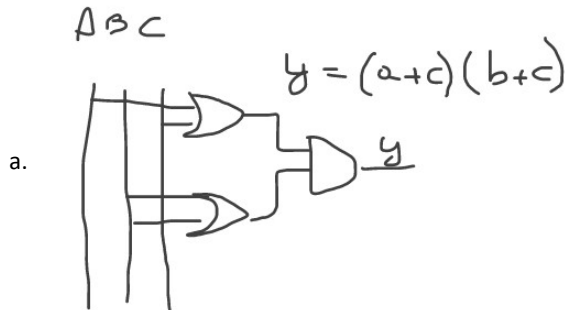
Usiamo ora invece le mappe k per minimizzare , usando la tabella di verità :



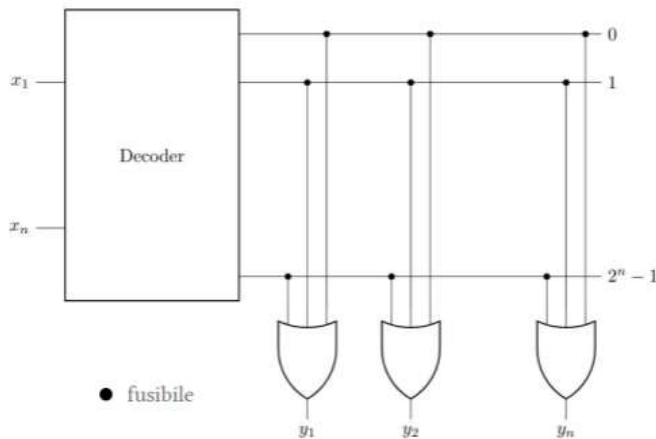
1. SOP



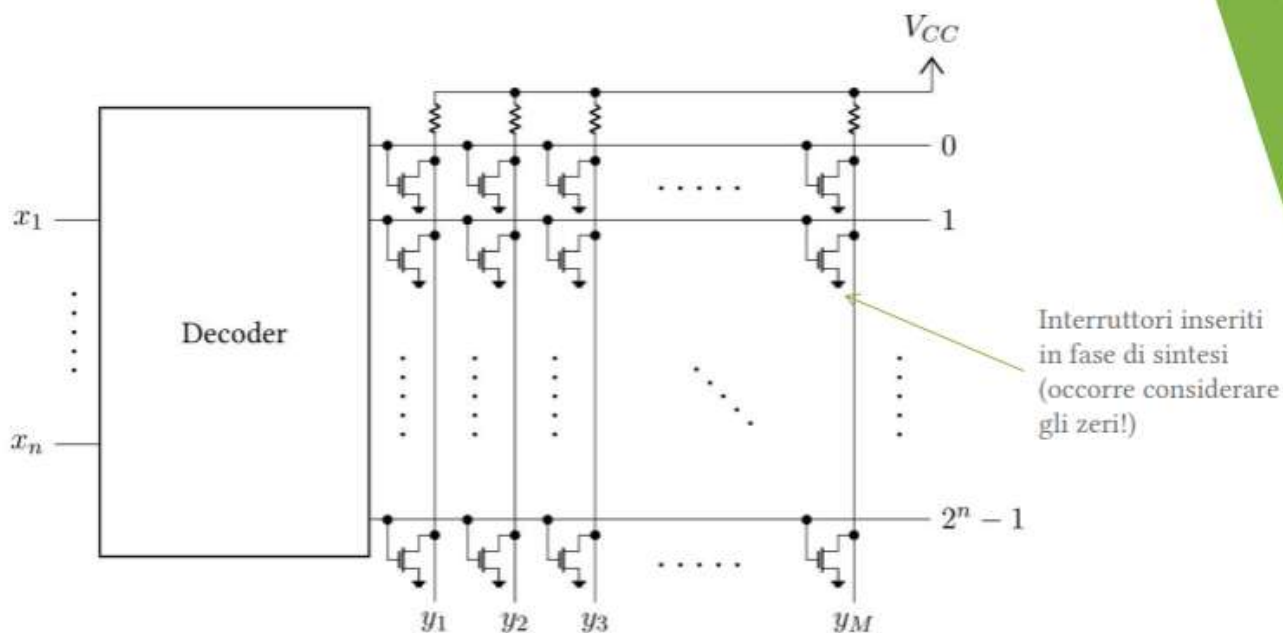
2. POS



Vediamo ora la **ROM (read only memory)**: circuito combinatorio in cui le componenti associano ad input un output ben definito . E' un dispositivo non riprogrammabile . Ha un dominio ben definito, uno spazio di indirizzi ed un co-dominio ben definito . In base alle porte or, si ha il numero di bit che memorizzo. Formata da or e and :



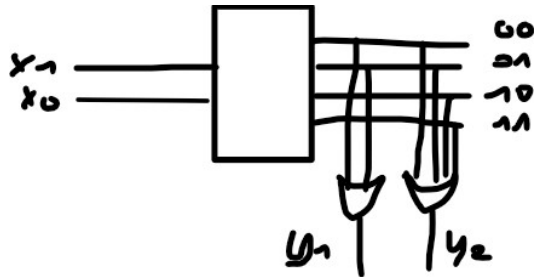
**Viene selezionata una sola uscita per via del decoder, quindi uscita attiva una sola : insieme di bit che devono costituire il valore che la mia memoria deve tornare .** Vediamo ora come sono fatti i collegamenti :



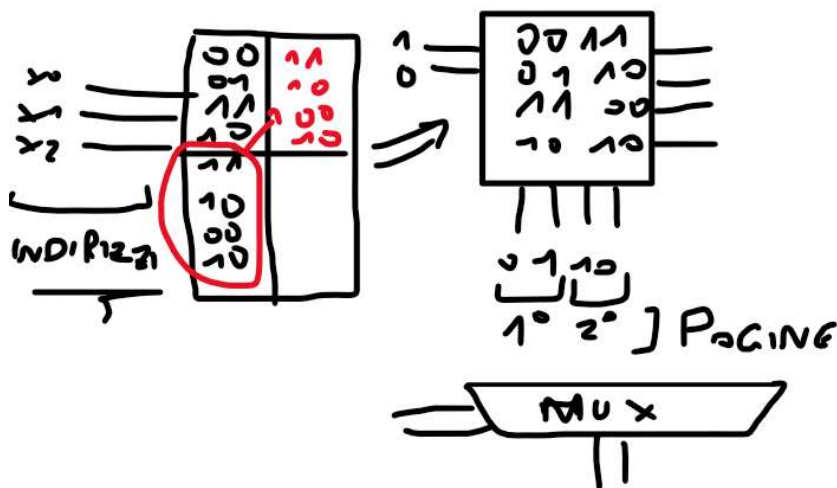
Vediamo un esempio a 2 bit :

X1	X0	Y1	Y0
0	0	1	1
0	1	1	1
1	0	0	1
1	1	0	1

In blu viene rappresentato lo spazio degli indirizzi, mentre in viola sono i valori memorizzati nella memoria. Circuitualmente :

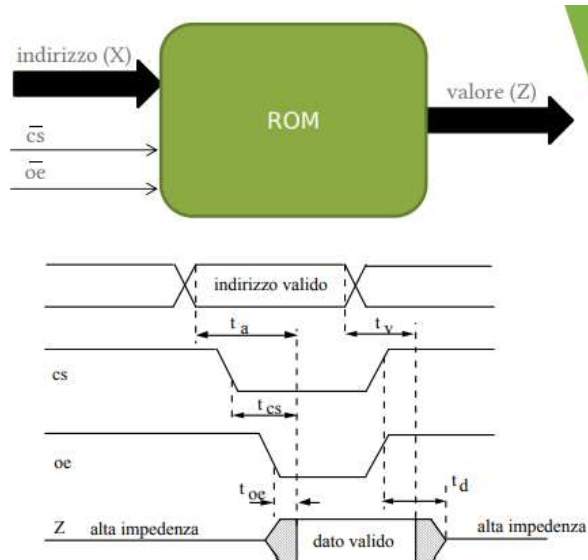


Di solito le Rom non hanno forma regolare : spesso è rettangolare (molto sconveniente dal punto di vista di occupazione dello spazio sul processore): quindi si cerca di farla diventare quadrata . Quindi in base a questa trasformazione si parla di **ROM PAGINATA** : rom divisa a pagine ( pezzi del rettangolo che ho spostato):



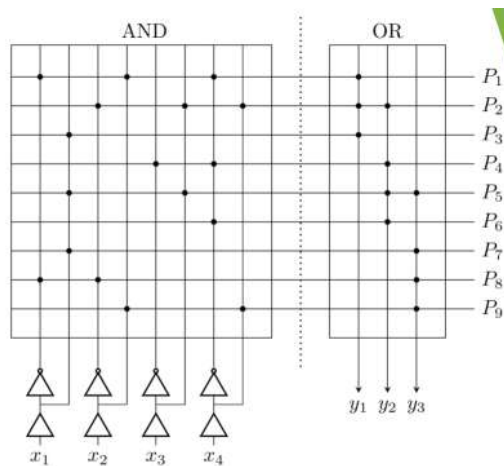
Il mux viene usato come selettore dell'uscita: usa  $x_2$  come segnale di ingresso . Ogni circuito genera un ritardo nel propagare il segnale :

- $t_a$ : tempo di propagazione dall'ingresso X all'uscita Z
- $t_{cs}$ : tempo di propagazione dall'ingresso cs all'uscita Z
- $t_{oe}$ : tempo di propagazione dall'ingresso oe all'uscita Z
- $t_v$ : tempo di mantenimento dell'uscita da quando commuta X o cs o oe
- $t_d$ : tempo di disabilitazione dell'uscita da quando commuta cs o oe



Attenzione al tempo di propagazione : se circuiti interconnessi , e campiono output prima che questo tempo sia trascorso , ho solo mondezza : da non usare come segnale da mandare al prossimo dispositivo. Per evitare ciò uso dei buffer tri-state che fanno da interruttore. Il segnale cs (chip select) viene usato per identificare la rom che uso : lavora in logica negata, mentre oe ( output enable) se abilitata permette di fare inserire il valore corretto proveniente dalla rom .

Vediamo ora la **PLA (programmable logical array)**: dispositivo che permette di realizzare tutte le operazioni , usando solo le porte and (decoder) ed or . Permette di diminuire il numero di componenti che devo usare :

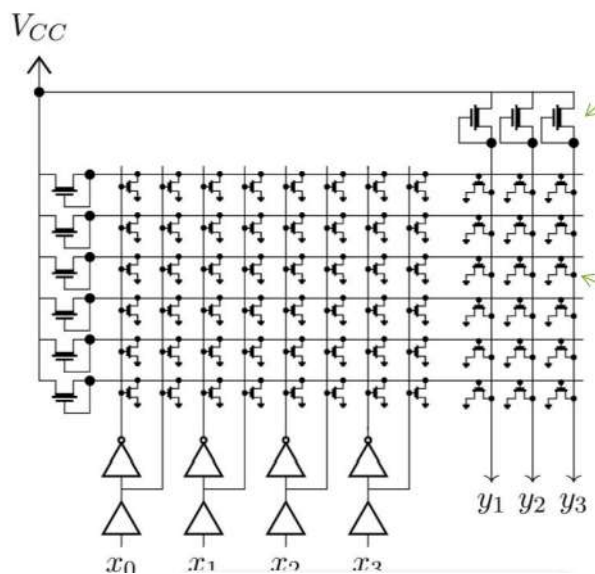


$$y_1 = \overline{x_1}x_2\overline{x_4} + \overline{x_2}x_3x_4 + x_1$$

$$y_2 = \overline{x_2}x_3x_4 + \overline{x_3}\overline{x_4} + x_1x_3 + x_2\overline{x_4}$$

$$y_3 = x_1x_2 + \overline{x_1}\overline{x_2} + x_1x_3 + x_2x_4$$

Dal punto di vista circuitale :



Quindi aumenta il numero di funzioni booleane , riducendo al minimo le componenti .