Esercizio Supponiamo di voler costruire un banco di memoria di dimensione $S_{banco} = 1\,GB$, $n_{banco} = 32$, avendo a disposizione i chip sottoelencati

Tipo	dim (MB)	m chip = bit indirizzo	n chip #bit dati	quantità
A	256	28	8	1
B	256	27	16	1
C	128	26	16	1
D	128	27	8	1
E	64	26	8	4

① Scoprire il valore degli elementi incognita

② Progettare il banco

① → m_A) $256\,MB = 2^m \times 8$ $\quad 1B = 8\,bit$

$2^{8+20+3} = 2^{m+3}$

$31 = m+3 \Rightarrow m = 28$

n_b) $256\,MB = 2^{27} \times n_b$

$2^{28+3-27} = n_b \Rightarrow n_b = 2^4 = 16$

dim_C) $dim\ C = 2^{26} \cdot 16 = 2^{10}\,Mbit = 2^7\,MB = 128\,MB$

M_d) $128\,MB = 2^m \cdot 8$

$2^{7+20+3} = 2^{m+3} \Rightarrow m = 27$

n_E)

$64\,MB = 2^{26} \times n_E \Rightarrow 2^{26+3} = 2^{26} \times n_E \Rightarrow n_E = 2^3 = 8$

q_E) $1\,GB = 1024\,MB$

$1024 = 256 + 256 + 128 + 128 + q_E \times 64 \Rightarrow q_E = 4$

2) → Ovvero che

$$1\,GB = 2^{30}\,MB = 2^{30}\,B = 2^{m_{banco}} \times GB \Rightarrow m_{banco} = 28$$

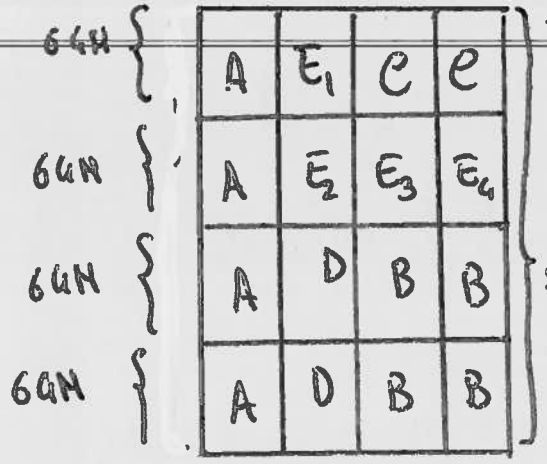
2^{28} → 256 M (megaread di memoria)

numero degli indirizzi presenti nel banco

$A_{27} A_{26} \dots A_0$ indirizzo del banco

28 bit

1 GB



← Suddivido il banco di memoria in blocchi di dimensione E utilizzando la formula della lezione precedente

Per la disposizione dei blocchi che ho a disposizione (la cui dimensione è multiplo intero di 64 MB) la scelta è arbitraria

Considerando ora la logica di funzionamento dei $\overline{CS}$ dei vari blocchi

• Lo spazio di indirizzamento totale dei blocchi A, per come li ho disposti, è pari allo spazio di indirizzamento del banco, pertanto i blocchi A devono funzionare tutti contemporaneamente

$$\overline{CS}_A = 0$$

• I blocchi B devono funzionare quando l'indirizzo appartiene alla seconda metà del banco, cioè è del tipo $1x \dots x$.

Pertanto

$$\overline{CS}_B = \begin{cases} 1 & \text{se } 1x\dots x \\ 0 & \text{se } 0x\dots x \end{cases} \Rightarrow \overline{CS}_B = \overline{A}_{27}$$

I blocchi D funzionano come i B, dunque $\overline{CS}_D = \overline{CS}_B$

Il blocco E_1 ed i blocchi C devono funzionare quando l'indirizzo si trova nella prima schiera del banco

A_{27}	A_{26}	$\overline{CS_C}$
0	0	0
0	1	1
1	0	1
1	1	1

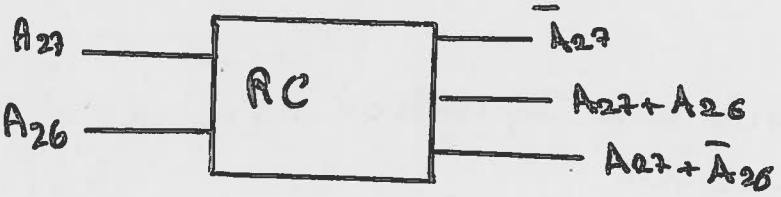
$$\overline{CS_C} = \overline{CS_{E_1}} = \overline{\overline{A_{27}} \cdot \overline{A_{26}}} = A_{27} + A_{26}$$

- I blocchi E_2, E_3, E_4 funzionano quando l'indirizzo si trova nella seconda schiera, pertanto

A_{27}	A_{26}	$\overline{CS_{E_i}}$
0	0	1
0	1	0
1	0	1
1	1	1

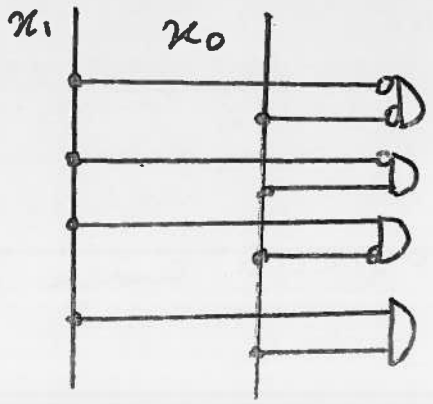
$$\overline{CS_{E_i}} = \overline{\overline{A_{27}} A_{26}} = A_{27} + \overline{A_{26}} \qquad i = 2,3,4$$

Per gestire il banco di memoria sarà necessaria una rete combinatoria del tipo



DECODER → rete combinatoria che, dati n bit in ingresso, presenta in uscita 2^n linee, una sola delle quali presenta un valore diverso dalle altre

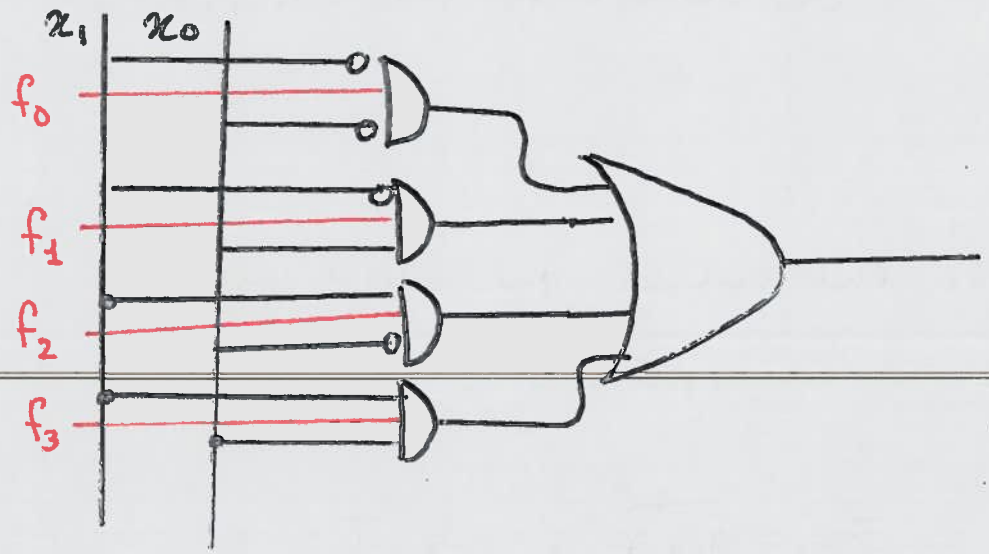
es. decoder in due variabili



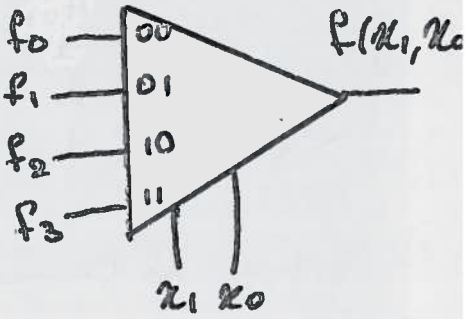
Il Multiplexer (MUX) è una rete combinatoria che funziona da selettore secondo una funzione algebrica.

Es. $f(x_0, x_1) = (\bar{x}_1 \bar{x}_0) f_0 + (\bar{x}_1 x_0) f_1 + (x_1 \bar{x}_0) f_2 + (x_1 x_0) f_3$

Circuitalmente questo corrisponde a

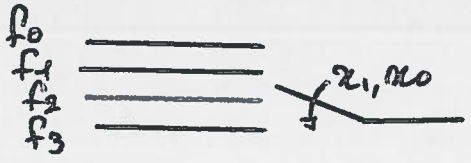


Spesso viene indicato anche con il simbolo



x_1, x_0 sono detti INGRESSI di SELEZIONE

MUX
- realizza una qualsiasi funzione combinatoria in n variabili
- funziona come un interruttore a più scatti, permettendo la propagazione di un unico segnale fra 2^n scelte (con n uno qui sopra)



Esempio costruiamo la seguente tabella di verità

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

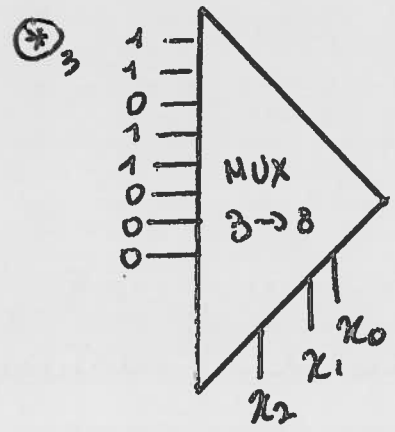
Forma canonica:

SP: $(\bar{x}_2 \bar{x}_1 \bar{x}_0) + \bar{x}_2 \bar{x}_1 x_0 + \bar{x}_2 x_1 x_0 + x_2 \bar{x}_1 \bar{x}_0$

PS: $(x_2 + \bar{x}_1 + x_0) \cdot (\bar{x}_2 + x_1 + \bar{x}_0)(\bar{x}_2 + \bar{x}_1 + x_0)(\bar{x}_2 + \bar{x}_1 + \bar{x}_0)$

con Karnaugh

x_2 \ x_1x_0	00	01	11	10
0	1	1	1	0
1	1	0	0	0

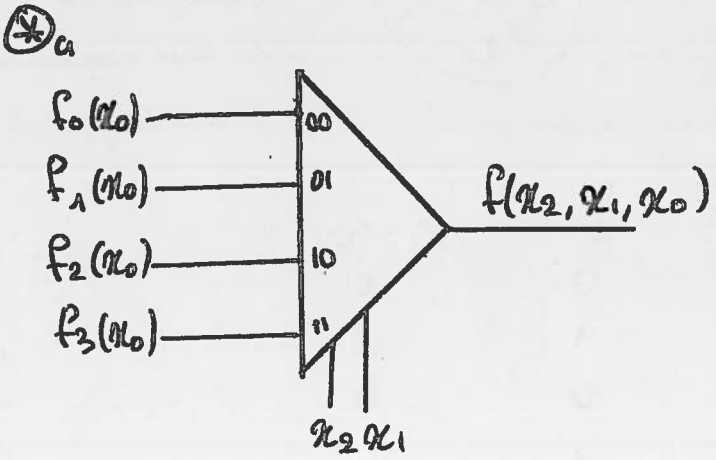


← per realizzare ⊛3 sono necessarie

- 1 porta OR a 8 ingressi
- 8 porta AND a 4 ingressi

E' possibile realizzare un MUX più economico di ⊛3 che realizzi tuttavia la medesima funzione logica → usando la f funzione di una delle variabili di ingresso di ⊛3 (es. $f = f(x_0)$).

x_2	x_1	x_0	f	
0	0	0	1	$f_0(x_0) = 1$
0	0	1	1	
0	1	0	0	$f_1(x_0) = x_0$
0	1	1	1	
1	0	0	1	$f(x_0) = \bar{x}_0$
1	0	1	0	
1	1	0	0	$f_3(x_0) = 0$
1	1	1	0	



$(x_0), f_1(x_0), f_2(x_0), f_3(x_0)$ sono tutte e sole le f. semp. con una var. booleana

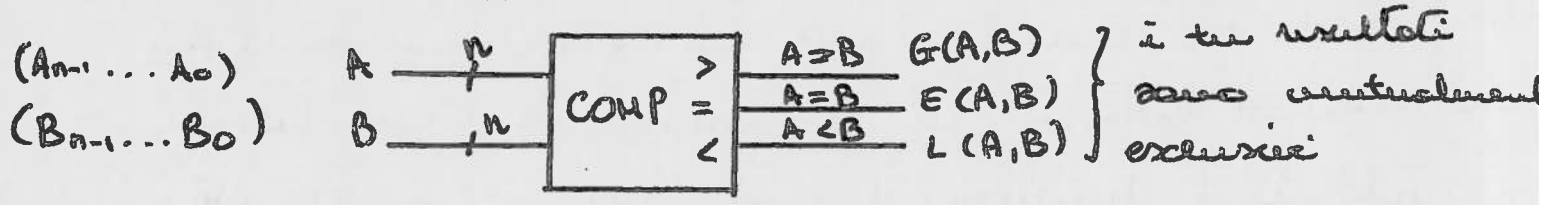
Per realizzare Φ_4 sono necessarie 1 porta OR a 4 ingressi e 4 porte AND a 3 ingressi.

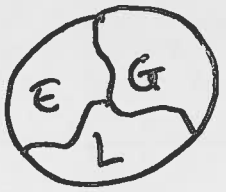
Es. La soluzione della disparità fra Φ_3 e Φ_4 in termini dimensionali risiede nel fatto che il multiplexer realizza 1:1 una tabella di verità, e pertanto ci è una dipendenza esponenziale dal numero di ingressi: aggiungere una variabile raddoppia la dimensione del MUX (da 4 a 8 porte AND con ingressi opportunamente negati).

L'aritmetica dinanzi la logica

Il calcolatore al suo interno lavora con funzioni logiche, non aritmetiche; per implementare al suo interno funzioni di tipo aritmetico è pertanto necessario utilizzare operatori logici

Il COMPARATORE è una rete combinatoria che, dati due numeri A e B che in ingresso in rappresentazione naturale, stabilisce se $A \gtrless B$.

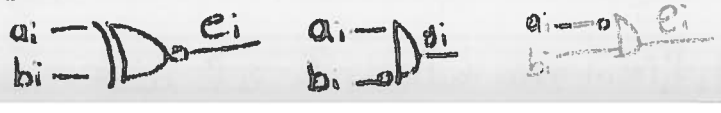




da somma di questi tre eventi (ognuno dei quali puo' assumere esclusivamente valore 0 o 1) deve essere 1

a_i	b_i	e_i = "$a_i = b_i$"	g = "$a_i > b_i$"	l = "$a_i < b_i$"
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

XOR negato

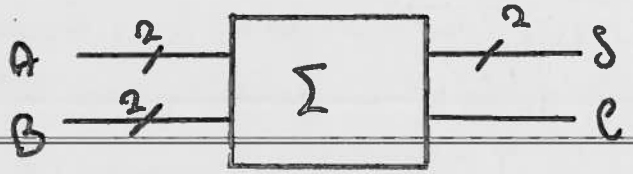


$E = e_{n-1} \cdot e_{n-2} \cdot \ldots \cdot e_1 \cdot e_0$ se $E=0$ $a_i > b_i \overset{*}{\vee} a_i < b_i$

$G = g_{n-1} + g_{n-2} e_{n-1} + e_{n-1} e_{n-2} g_{n-3} + \ldots$

$L = \ldots$

Supponiamo di voler realizzare un dispositivo che implementi la somma aritmetica fra due numeri.



e come bit (bit di riporto della somma)

La tabella di verità è

a_1	a_0	b_1	b_0	C	S_1	S_0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

la sua mappa di Karnaugh è'

a_1, a_0 \ $b_1 b_0$	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	0	1	1	1
10	0	0	1	1

$C = a_0 b_1 b_0 + a_1 b_1 + a_1 a_0 b_0$

Consideriamo

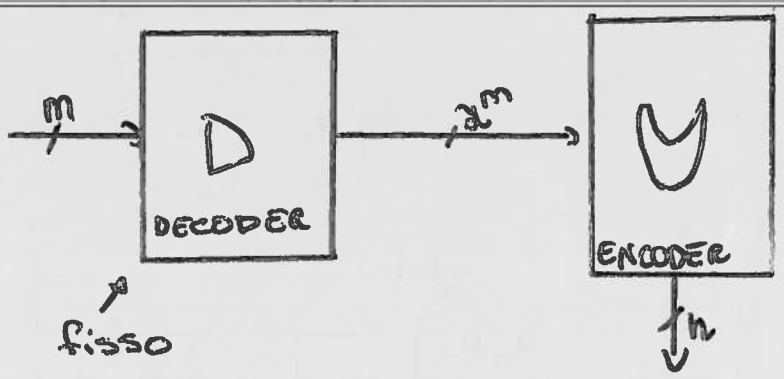
$a_0 b_1 b_0 \Rightarrow$ $a_1 a_0$ è dispari (perché $a_0 = 1$) $b_1 b_0 \overset{\text{"vale"}}{=} 3$ (perché entrambi sono a 1

$a_1 b_1 \Rightarrow$ $a_1 a_0$ e $b_1 b_0$ valgono entrambi almeno 2 perché' $a_1 = 1$ $b_1 = 1$

$a_1 a_0 b_0 \Rightarrow$ $a_1 a_0$ vale tre $b_1 b_0$ è dispari (perché' $b_0 = 1$)

ROM

Una ROM (Read Only Memory) è un dispositivo che realizza in modo 1:1 una tabella di verità in logica positiva. E' una memoria di sola lettura, nella quale tutti i possibili stati sono dati già precalcolati. Circuitalmente, ogni ROM è costituita da un blocco AND che si comporta da decoder (da m ingressi presenta in uscita 2^m stati) e da un blocco OR che codifica una funzione da n bit a partire da 2^m ingressi



è un encoder è un disp. programmabile: realizza 1:1 una funzione logica

Dal punto di vista funzionale, una ROM si comporta come una memoria non volatile di dimensione

$S' = 2^m \times n$

↓

memoria che permette la lettura ma non la scrittura dei suoi dati e che mantiene quanto memorizzato

→ il decoder genera tutti i possibili stati e li trasmette all'encoder