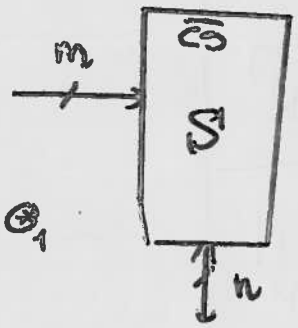
Modulo (chip) di memoria




ingresso indirizzi

$S'_{bit} = 2^m \times n$

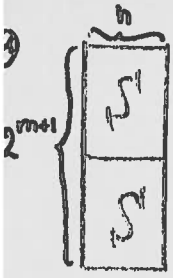
CS chip select abilita il chip (funziona quando è a 0)

n ingresso/uscita dati

Supponiamo di voler costruire un banco di memoria a partire da due chip del tipo ①.

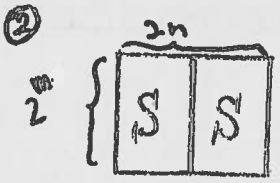
Poniamo individuare due casi

ESTENSIONE INDIRIZZO




$2^{m+1}$ parole indirizzabili di dimensione n

ESTENSIONE DATI



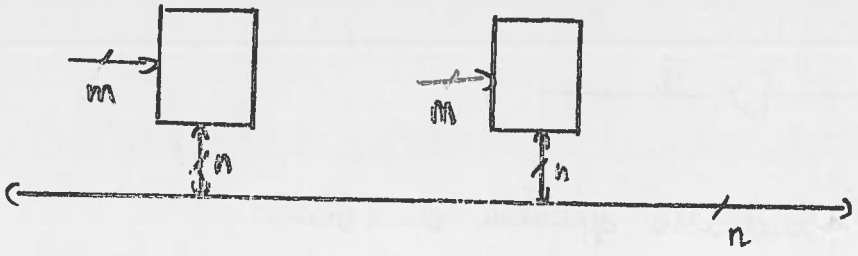

$2^m$ parole indirizzabili di dimensione 2n

$S'_{①} = 2 \cdot 2^m \times n = 2^{m+1} \times n$

$S'_{②} = 2^m \times (2n) = 2 \cdot 2^m \times n = 2^{m+1} \times n$

In entrambi i casi si ha un raddoppiamento della dimensione della memoria

Studiamo il caso ①. Circuitalmente affianco i moduli, mantenendo il comportamento logico estensione indirizzi




Nella lettura/scrittura dati dobbiamo estrometterne uno dei due moduli, perché la dimensione della parola dati è n e non 2n

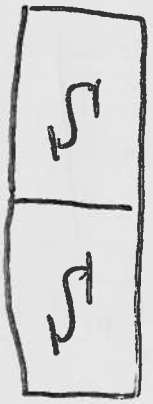
$A_m A_{m-1} \dots A_0$ parola di indirizzo di m+1 bit

tutte le parole che si trovano nella prima parte del banco hanno indirizzo del tipo

$0 x \dots x$

quelle della seconda parte hanno invece forma

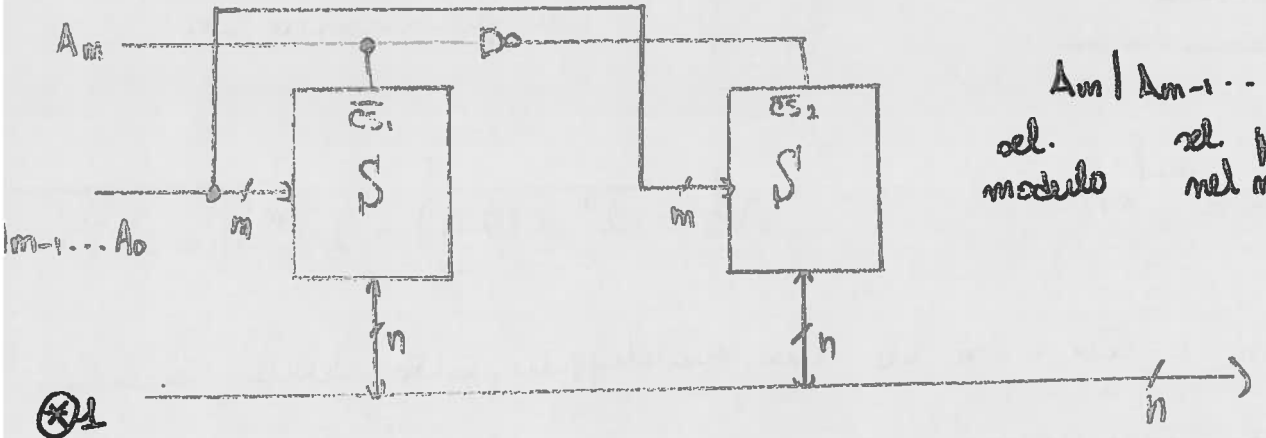
$1 x \dots x$



$A_{m-1} \dots A_0$ si riferisce a due locazioni di memoria, una nel primo e una nel secondo blocco

$A_m$ è il bit decisivo

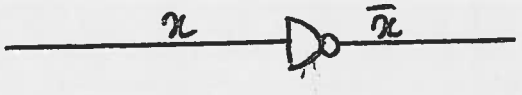
Con questa suddivisione lo schema circuitale è del tipo




$A_m$ | $A_{m-1} \dots A_0$
sel. modulo | sl parola nel modulo

$\overline{CS}_1 = A_m \qquad \overline{CS}_2 = f(A_m)$ funzione LOGICA (scelore)

$f(A_m) = \overline{A}_m \Rightarrow \begin{matrix} A_m = 0 \hookrightarrow f(A_m) = 1 \\ A_m = 1 \hookrightarrow f(A_m) = 0 \end{matrix}$ è la funzione NOT

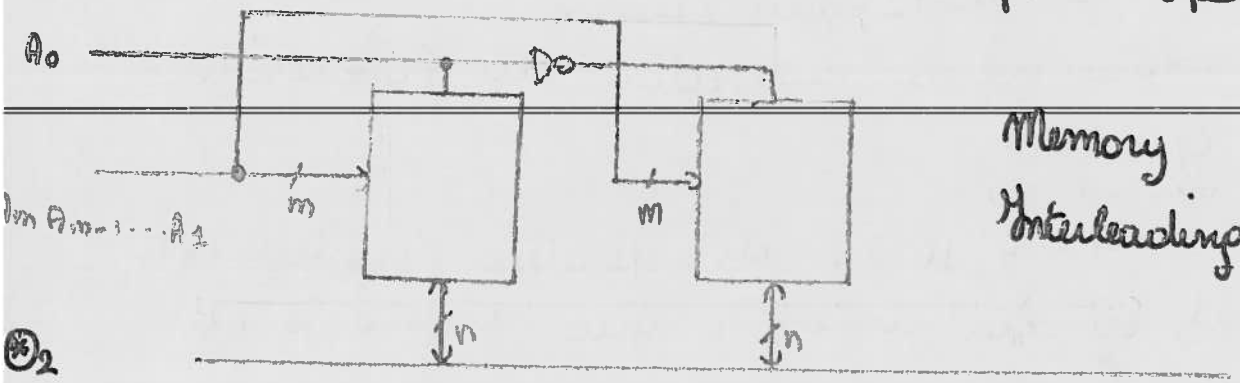
$f(A_m)$ 


Es. $f(A_m) = 0 \iff$ l'indirizzo appartiene al 1° blocco
$f(A_m) = 1 \iff$ l'indirizzo appartiene al 2° blocco

Supponiamo di fare una scelta diversa rispetto al come mandare i bit

$A_m\ A_{m-1} \dots A_1 \mid A_0$
↑ sel. modulo
sel parola nel modulo


in questo modo $\begin{cases} x\,x \dots x\,0 \rightarrow 1^{\text{a}} \text{ parte (parole pari)} \\ x\,x \dots x\,1 \rightarrow 2^{\text{a}} \text{ parte (parole dispari)} \end{cases}$




Memory Interleaving

Il partire dal medesimo problema (costruire un banco di memoria di dimensione 2S a partire da due chip di memoria di dimensione S') siamo arrivati a due soluzioni, $⊛_1$ e $⊛_2$, che pur impiegando la medesima tecnologia hanno organizzazione diversa.
Lo schema $⊛_2$ è più vantaggioso di $⊛_1$ dal punto di vista della CPU.
Supponiamo in fatti che il processore voglia leggere una serie consecutiva di dati.
Con $⊛_1$, dopo aver posto un indirizzo, deve attendere che il modulo di memoria (che ha tempi di risposta maggiori rispetto alla CPU) abbia posto il dato richiesto sul bus dati prima di poter procedere con la lettura del successivo.
Con $⊛_2$, invece, posso sfruttare l'alternanza pari/dispari degli indirizzi per ottimizzare i tempi.
Infatti, mandato l'indirizzo da leggere sul un modulo,

sapendo che il accesso apparterrà all'altro chip resto di $\overline{CS}_2$, posso inoltrare la richiesta anche all'altro modulo.

In questo modo, la CPU riesce ad inoltrare due richieste di lettura nel tempo in cui in linea ne avrebbe potuta effettuare una.

Conclusioni: - A parità di tecnologia impiegata, organizzazioni diverse portano a prestazioni diverse.

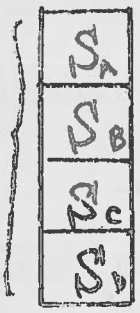
- Esistono schemi realizzativi che favoriscono certi tipi di operazioni

Supponiamo adesso, a partire da quattro moduli di dimensione S, di voler costruire una memoria di dimensione 4 S.

Abbiamo tre schemi organizzativi possibili

ESTENSIONE INDIRIZZI

①

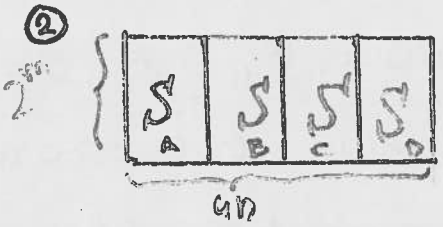



$2^{m+2} \times n$

$A_{m+1} A_m \dots A_0$
parola di indirizzo

ESTENSIONE DATI

②

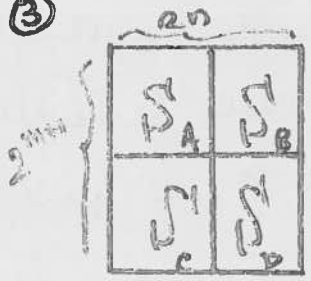



$2^m \times 4n$

$A_{m-1} \dots A_0$
parola di indirizzo

ESTENSIONE DATI E INDIRIZZI

③




$2^{m+1} \times 2n$

$A_m A_{m-1} \dots A_0$
parola di indirizzo

In ①, ②, ③ la dimensione totale della memoria è $2^{m+2} \times n$

Analizziamo il caso ③.

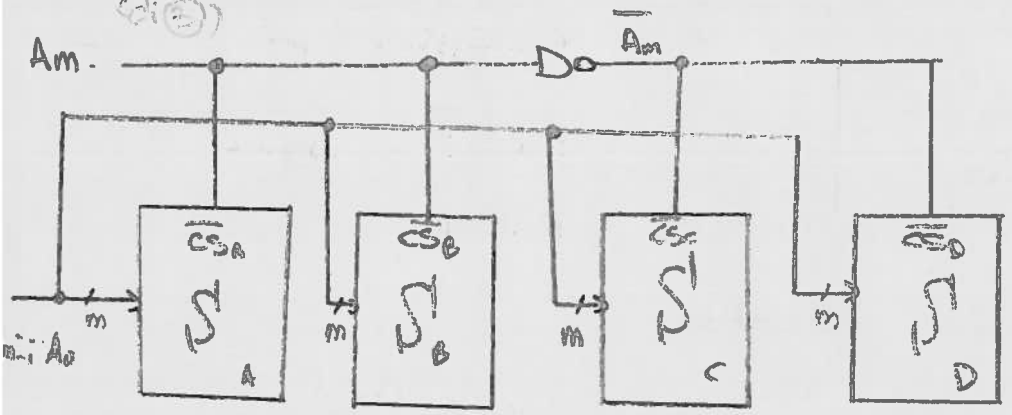
Qui le schede funzionano a coppie $A-B$ e $C-D$ (infatti $\overline{CS}_A = \overline{CS}_B$ e $\overline{CS}_C = \overline{CS}_D$, con $\overline{CS}_A = \overline{CS}_C$).

$A_m \mid A_{m-1} \dots A_0$

selezione di scheda

$A_m = \begin{cases} 0 & \text{blocchi } A-B \\ 1 & \text{blocchi } C-D \end{cases}$

il suo schema circuitale sarà pertanto




Analizziamo ora il caso ①. Qui i chip dovranno funzionare uno alla volta, pertanto

| ② | $A_{m+1}$ | $A_m$ | $\overline{CS_A}$ | $\overline{CS_B}$ | $\overline{CS_C}$ | $\overline{CS_D}$ |
|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 1 | 1 | 1 |
| | 0 | 1 | 1 | 0 | 1 | 1 |
| | 1 | 0 | 1 | 1 | 0 | 1 |
| | 1 | 1 | 1 | 1 | 1 | 0 |

← ∀ combinazione di $A_{m+1}, A_m$ c'è sempre un solo $\overline{CS}$ uscito

Oss. Date k variabili, il numero di funzioni che è possibile implementare è $2^{2^k}$.

Infatti ② è solo una delle 16 funzioni possibili.

La rete combinatoria che funziona secondo questo schema è detta decoder: a partire da k ingressi genera $2^k$ uscite, delle quali una sola ha valore diverso dalle altre

DECODER (DECODIFICATORE BINARIO.

Lo schema circuitale corrispondente a ① è