Lez. 1° dic. 2011
(2h)

MEMORIE

Tipo	Organizzazione	Uso	Densità	Velocità	Costo per Bit	Elemento Base	Refresh	Bus MPX
STATICA	per parola	Cache	→	↑ (t_{ACC}↓)	↑	flip-flop	NO	NO
DINAMICA	per bit	MM	←	↓ (t_{ACC}↑)	↓	condensatore	SÌ	SÌ

Sono partito dalla gerarchia di memoria.

Il proc. pensa di lavorare con la MM e basta. In realtà c'è tutta un'elettronica (+ il software di SO) che gli fa sembrare di avere a che fare con una MM di GRANDI DIMENSIONI e MOLTO VELOCE. Ora: questo è ottenuto con algoritmi molto scaltri, non è infatti "gratis" che si hanno alta velocità e grandi dimensioni: sono due cose in conflitto.

Addirittura il processore esce con m bit di indirizzo, ma non è detto che $\exists\ 2^m$ parole di MEMORIA fisica: si parlerà di MEMORIA VIRTUALE, che apparentemente è una MM, ma in realtà è per la maggior parte spazio DISCO (grande capacità, bassa velocità). Similmente, la CACHE (piccola capacità, grande velocità) si usa per aumentare la velocità apparente:

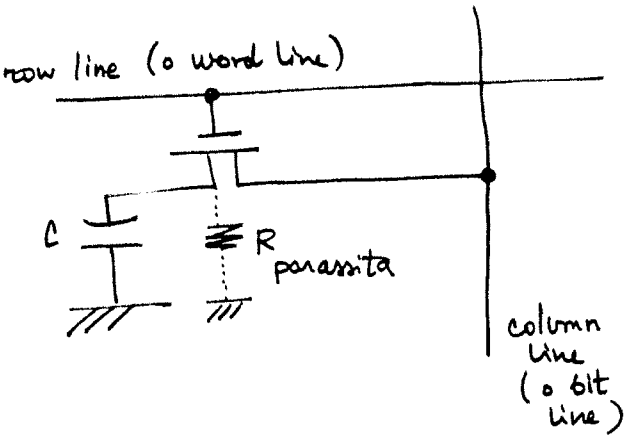
$$t_{ACC} = t_{MM} \quad \text{(senza cache)}$$

$$t_{ACC} = h\,\underbrace{t_{CACHE}}_{t_{hit}} + \underbrace{(1-h)}_{m}\,\underbrace{t'_{MM}}_{t_{miss} > t_{MM}}$$

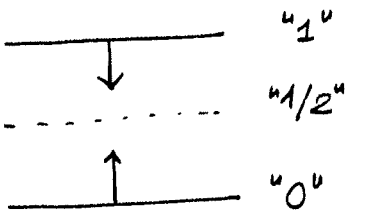
Dunque la cache si usa per enfatizzare la velocità, e la MM per assicurarsi una buona capacità, anche a scapito della velocità. Il BUS MULTIPLEXING, ad es., dimezza la velocità. Per contro, l'organizzazione PER BIT favorisce l'aumento della capacità a parità di # di piedini del chip, perché minimizza la somma $m+n$ a parità di capacità.

(m: ind., n: dati)

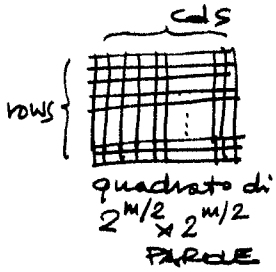
Dopo aver parlato di BUS MULTIPLEXING (riferendomi allo schema temporale del libro BUCCI, contrastato con quello della RAM STATICA), sono arrivato — — a parlare dell'elemento base di memorizzazione della RAM DINAMICA: il μC:



Il C è come un SECCHIO che perde: se la carica è "1", esso tende a scendere a "1/2", e se è "0", tende a salire a "1/2"



Il refresh serve a interrompere questo processo che porterebbe inevitabilmente (se arrivassimo a "1/2") alla perdita dell'informazione memorizzata.

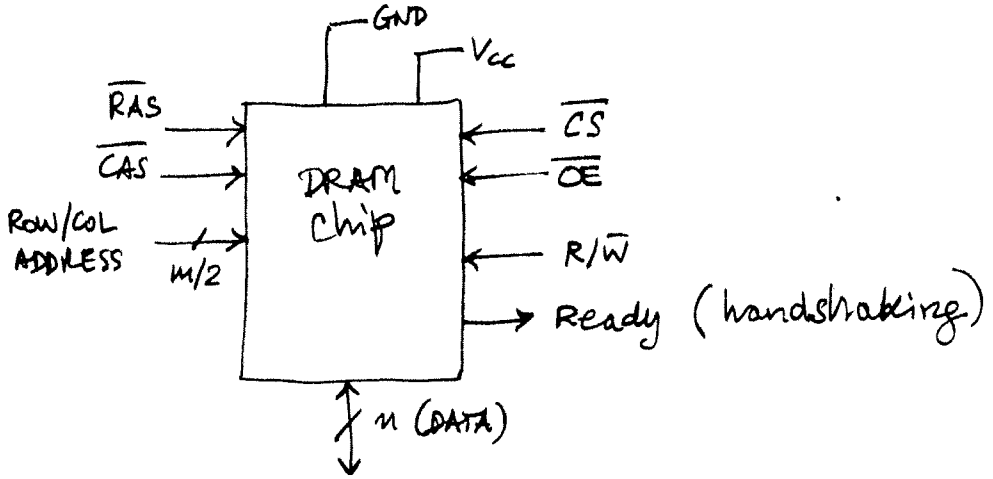


Cosa vuol dire col refresh:

- si effettua eseguendo semplicemente una LETTURA RIGA PER RIGA DELLA MEMORIA.
- del refresh si occupa una logica (esternamente posta sulla scheda di memoria) che fa da tramite tra il chip e il processore. Durante il refresh, questa logica impedisce alla CPU di accedere alla memoria, di fatto rendendendo più lungo (e variabile nel tempo) il tacc.
- Durante il refresh il $\overline{CS}$ è asserito, ma — nonostante si stia facendo una lettura — l'$\overline{OE}$ è DISASSERITO.
- Il refresh non incide sulle prestazioni in maniera significativa: prende circa il 3% del tempo di funzionamento della MM.

Ho poi fatto una digressione sulle recenti evoluzioni (non trattate nel ns corso): SDRAM, ovvero DRAM SINCRONA.

Nelle SDRAM il parametro importante, più che t_{ACC} (che nelle DRAM va confrontato con $1/f_{CLK}$ del processore) è la FREQUENZA DI ESERCIZIO f_{MEM}. Questa viene scelta in modo da adeguarsi alla f_{BUS}. Quest'ultima deriva da un'altra innovazione (anche questa non trattata nella nostra trattazione, dove abbiamo parlato solo di t_{ACC} e di f_{CLK}): l'aver differenziato la frequenza INTERNA della CPU (f_{CLK}) e quella usata per collegarsi all'esterno coi dispositivi ancillari (f_{BUS}). Se f_{BUS} fosse pari a f_{CLK}, ci sarebbero problemi di accoppiamento elettromagnetico — i collegamenti esterni sono lunghi e il fascio che diventa antenna è più facile che non all'interno della CPU.



ESERCIZI: COSTRUZIONE DI BANCHI DI MEMORIA

Elenco dei compiti che contengono esercizi di questo tipo:

- 30 aprile 2003
- 12 aprile 2006
- 16 aprile 2007 ($n_{BANCO} < n_{CHIP}$)
- 9 gennaio 2008
- 30 marzo 2009
- 16 febbraio 2010

} chip diversi tra loro

1 Sono partito dal caso più facile:

- chip uguali tra loro
- $n_{BANCO} \equiv n_{CHIP}$

} è il classico caso di ESPANSIONE DI INDIRIZZO

(cfr. figura Libro BUCCI)

Si ha:

$$\#chip = \frac{S_{BANCO}}{S_{CHIP}}$$

I chip sono attivi UNO ALLA VOLTA, pilotati dal decoder che asserisce una sola linea: quella del CS del chip che "lavora". Il decoder ha in ingresso gli $(n_{BANCO} - n_{CHIP})$ più significativi, mentre i bit di indirizzo $A_{n_{CHIP}-1} \dots A_0$ vanno in parallelo a tutti i chip.

2 Generalizzazione:

- chip uguali tra loro, ma
- $n_{BANCO} \neq n_{CHIP}$ [ho parlato solo di $n_{BANCO} > n_{CHIP}$]

↙ ma ho dato in passato anche es. $n_{BANCO} < n_{CHIP}$

%

Qui abbiamo, come prima:

$$\#\,chip = \frac{S_{BANCO}}{S_{CHIP}},$$

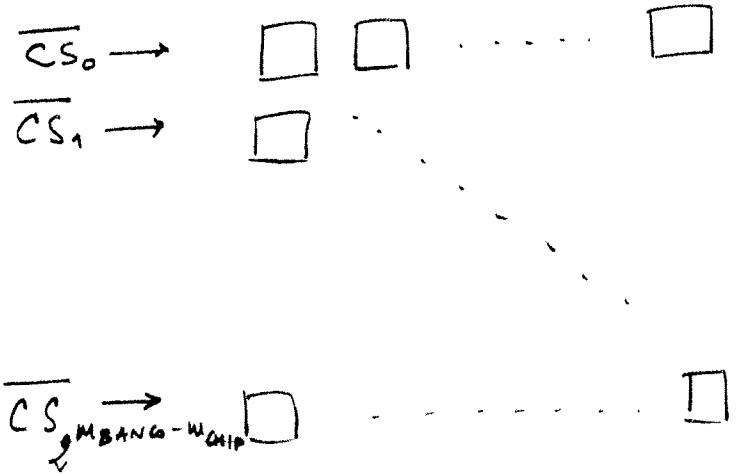
ma ci sono $\frac{n_{BANCO}}{n_{CHIP}}$ chip attivi contemporaneamente,

facenti cioè parte di una "schiera" di chip, ognuno gestendo un sottinsieme di bit di uscita.

Il numero di schiere è pari a

$$\#\,schiere = \frac{\#\,chip}{\#\,chip/schiera} \left(= \frac{S_{BANCO}}{S_{CHIP}} \Big/ \frac{n_{BANCO}}{n_{CHIP}} = \frac{S_{BANCO}/n_{BANCO}}{S_{CHIP}/n_{CHIP}} = \frac{2^{m_{BANCO}}}{2^{m_{CHIP}}} = 2^{m_{BANCO}-m_{CHIP}} \right)$$

Ci sarà sempre un decoder, che invece di pilotare singoli chip, pilota schiere di chip:



3 Ulteriore generalizzazione:

- chip diversi
- $n_{BANCO} > n_{CHIP}$

In questo caso, conviene rappresentare la memoria come un RETTANGOLO, che verrà suddiviso in BLOCCHI di dimensioni pari alle dimensioni più piccole (lungo entrambi gli assi: #parole (2^m) e #bit d'uscita (n)) dei vari chip in gioco.

Esempio(*): $S_{BANCO} = 1GB = 256M \times 4B$

Chip utilizzabili:

$S_A = 256M \times 1B = 256MB$

$S_B = 128M \times 2B = 256MB$

$S_C = 128M \times \underline{1B} = 128MB$ (min sudd. orizz.)

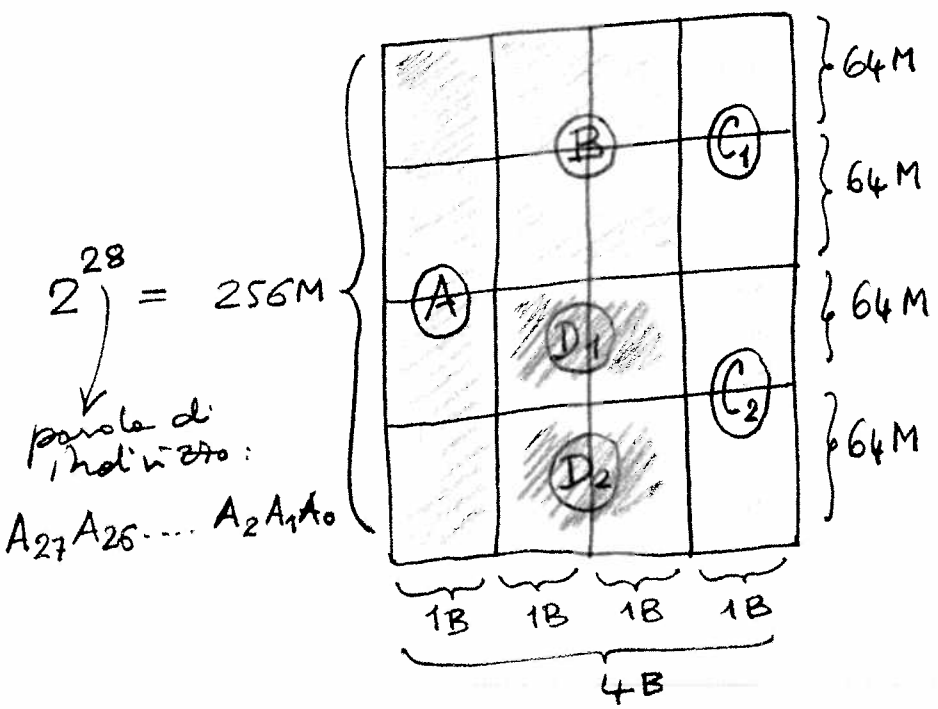
$S_D = \underline{64M} \times 2B = 128MB$ (min sudd. vert.)

(*) relativ. del compito marzo 09

Supongo di usare:

tipo	quantità	dim.
A	1	256MB
B	1	256MB
C	2	256MB
D	2	256MB

Tot. 1GB



$2^{28} = 256M$

parola di indirizzo:

$A_{27}A_{26}\ldots A_2A_1A_0$

che riempiono il rettangolo così (*)

(*) ho qui parlato dei 12 PENTAMINI e dei problemi della costruzione di rettangoli 3x20, 4x15, 5x12, 6x10. E' un GIOCO COMBINATORIO.

Ora: i Chip saranno asseriti oppure no a seconda dell'indirizzo applicato al banco.
La logica per la generazione del $\overline{CS}$ di ogni chip si ricava facilmente analizzando il rettangolo nelle sue 4 FASCE ORIZZONTALI, e registrando per ogni fascia <u>QUALI CHIP partecipano alla parola dati</u>. Si ha qui:

A è asserito in tutte e 4 le fasce da 64M $\Rightarrow$ $\overline{CS_A} = 0$ (ossia A SEMPRE ASSERITO)

B " " nelle prime due fasce, caratterizzate da $A_{27} = 0$ $\Rightarrow$ $\overline{CS_B} = A_{27}$

C_1 " " come B $\Rightarrow$ $\overline{CS_{C_1}} = A_{27}$

C_2 " " nelle ultime due fasce $\Rightarrow$ $\overline{CS_{C_2}} = \overline{A_{27}}$

D_1 " " nella 3ª fascia ($A_{27}=1$ & $A_{26}=0$) $\Rightarrow$ $\overline{CS_{D_1}} = \overline{A_{27} \cdot \overline{A_{26}}} = \overline{A_{27}} + A_{26}$

D_2 " " nella 4ª fascia ($A_{27}=1$ & $A_{26}=1$) $\Rightarrow$ $\overline{CS_{D_2}} = \overline{A_{27} \cdot A_{26}} = \overline{A_{27}} + \overline{A_{26}}$

NOTA: Questo metodo di soluzione generalizza i precedenti.
Nel caso [2] si ha infatti

$\overline{CS_{1a}} = \overline{CS_{1b}} = \overline{CS_{1c}} = \overline{CS_{1d}} \doteq \overline{CS_1} = A_{M_{BANCO}-1} + A_{M_{BANCO}-2}$

e sim. $\overline{CS_2} = A_{M_{BANCO}-1} + \overline{A}_{M_{BANCO}-2}$

$\overline{CS_3} = \overline{A}_{M_{BANCO}-1} + A_{M_{BANCO}-2}$

$\overline{CS_4} = \overline{A}_{M_{BANCO}-1} + \overline{A}_{M_{BANCO}-2}$