



# La struttura delle memorie

Prof. Alberto Borghese  
Dipartimento di Scienze dell'Informazione  
[alberto.borghese@unimi.it](mailto:alberto.borghese@unimi.it)

Università degli Studi di Milano

Riferimento Patterson v.5: 2.11 – 5.2, 5.5, 5.12, B8, B9.



# Sommario

SRAM.

DRAM.

Codici di errore



## Memoria Principale



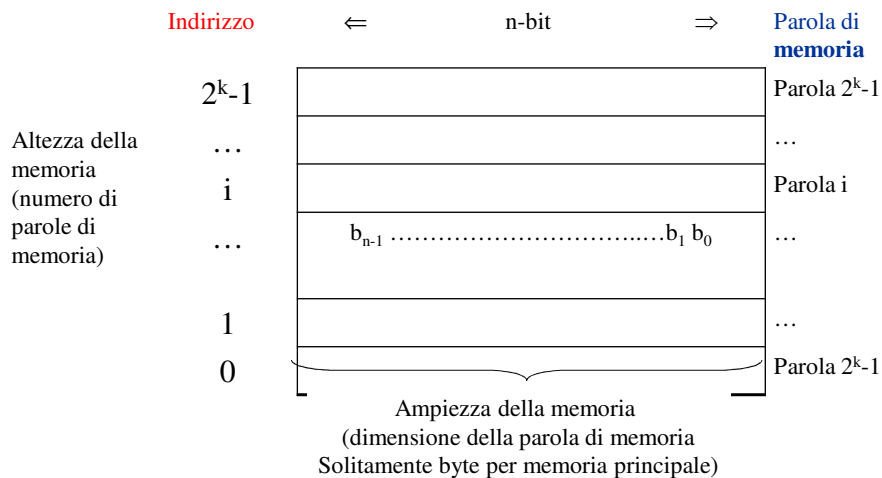
- Le memorie in cui ogni locazione può essere raggiunta in un breve e prefissato intervallo di tempo misurato a partire dall'istante in cui si specifica l'indirizzo desiderato, vengono chiamate memorie ad accesso casuale (*Random Access Memory – RAM*)
- Nelle RAM il *tempo di accesso alla memoria* (tempo necessario per accedere ad una parola di memoria) è *fisso e indipendente* dalla posizione della parola alla quale si vuole accedere.
- Il contenuto delle locazioni di memoria può rappresentare sia istruzioni che dati, sui quali l'architettura sta lavorando.
- La memoria può essere vista come un array monodimensionale.



## La memoria



- La memoria è vista come un unico grande array uni-dimensionale.
- Un **indirizzo di memoria** costituisce un **indice** all'interno dell'array.





## Indirizzi nella memoria principale



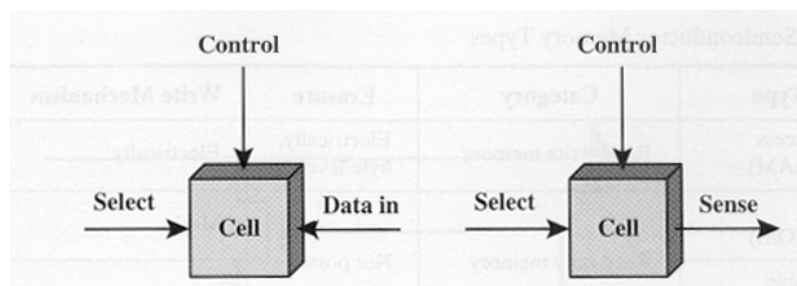
- La memoria è organizzata in *parole* composte da  $n$ -bit che possono essere indirizzati separatamente.
- Ogni **parola** di memoria è associata ad un **indirizzo** composto da  $k$ -bit.
- I  $2^k$  indirizzi costituiscono lo *spazio di indirizzamento* del calcolatore. Ad esempio un indirizzo composto da 32-bit genera uno spazio di indirizzamento di  $2^{32}$  o 4Gbyte.
- Ottimizzazione della struttura
  - Tecnologia three-state
  - Struttura a matrice



## Cella di memoria



La memoria è suddivisa in celle, ciascuna delle quali assume un valore binario stabile.  
Si può scrivere il valore 0/1 in una cella.  
Si può leggere il valore di ciascuna cella.

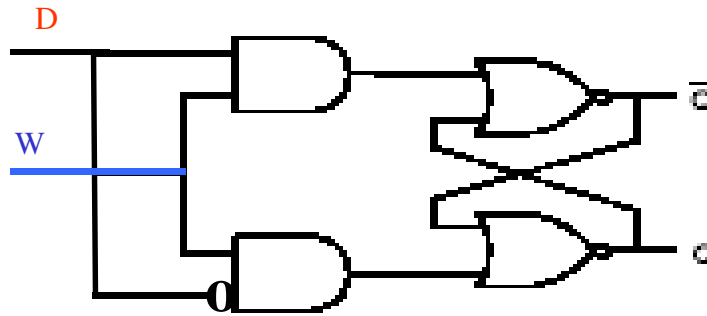


Quale struttura di memoria abbiamo già incontrato?

Control (lettura – abilitazione; scrittura)  
Select (cf. dataport)  
Data in & Sense (Data in & Data out).



# Cella SRAM



Selezione (porta di lettura e porta di scrittura)  
 Lettura - sempre disponibile in uscita  
 Scrittura - segnale esplicito (in AND con il clock in caso di cella sincrona).



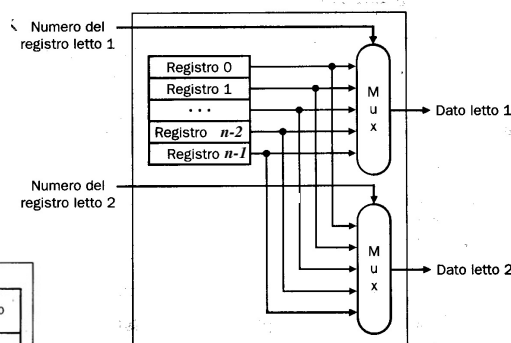
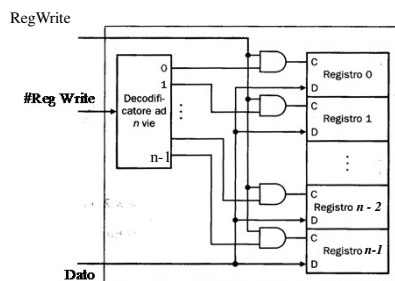
# Register file



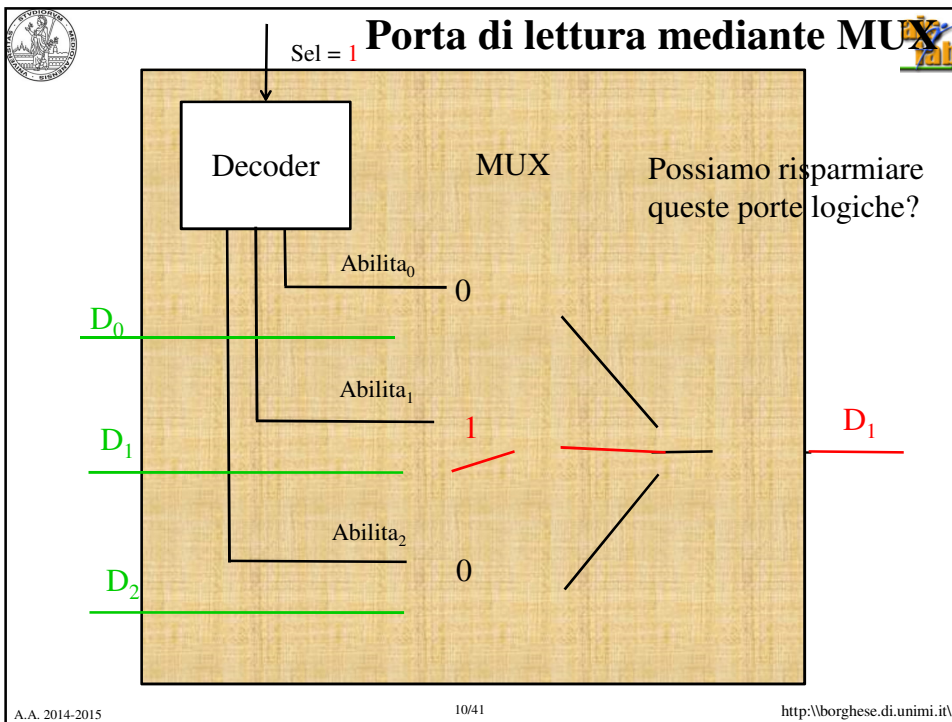
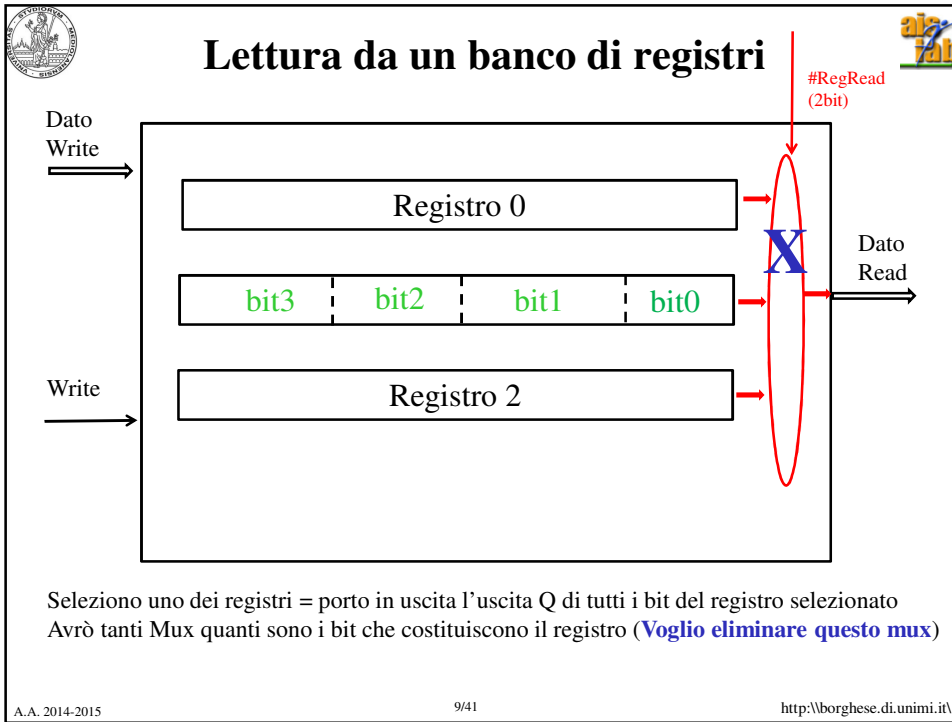
Il tempo di lettura dipende dal cammino critico dei Mux.

Il tempo di scrittura dipende dal cammino critico del Decoder.

Numero\_registro = selettore.

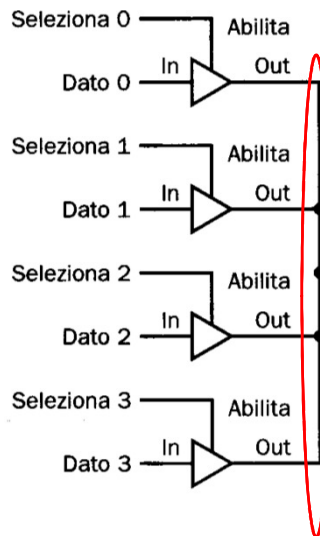


Selezione - #registro  
 Lettura - sempre disponibile in uscita (dopo tempo di commutazione del MUX)  
 Scrittura - segnale esplicito (in AND con il clock in caso di cella sincrona).



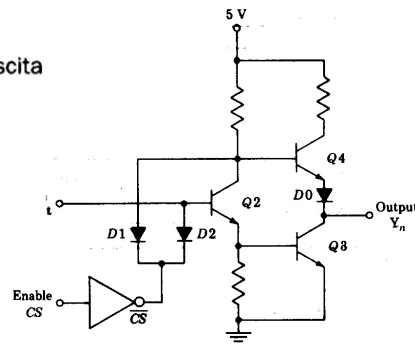


## Memoria three-state (soluzione HW)

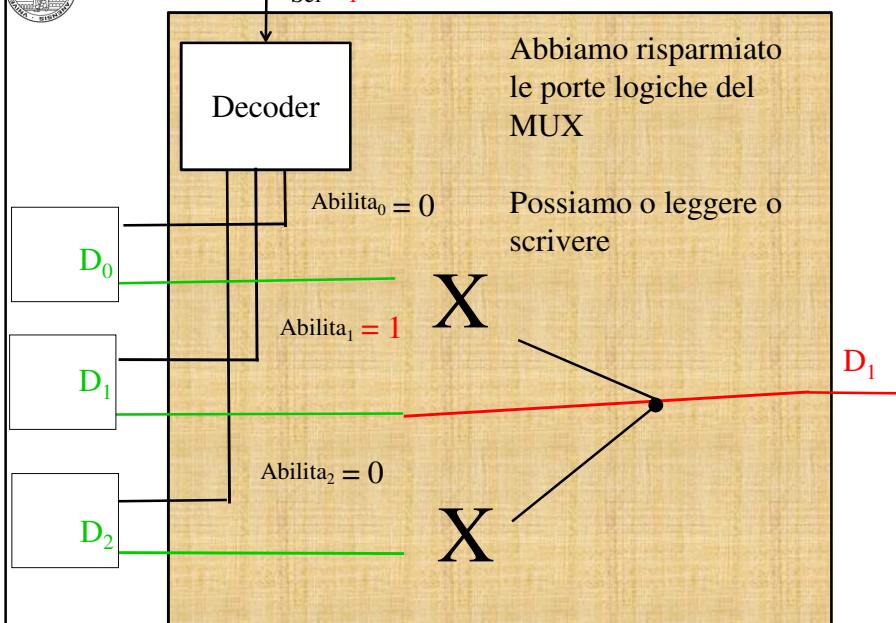


Tutte le uscite delle celle sono collegate ad un'uscita comune => E' necessario evitare conflitti fra le uscite.

Uscite "isolate" con porte *three-state*  
Seleziono una sola cella alla volta



## Porta di lettura three state



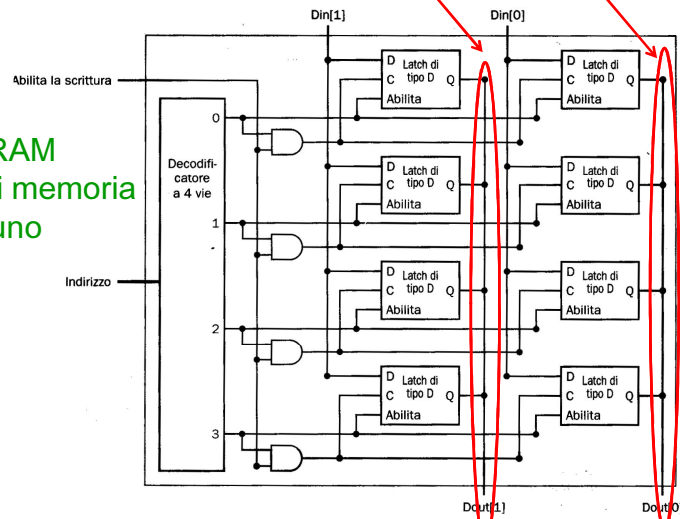


## Esempio di SRAM

Risparmio il Mux di uscita



Esempio: SRAM  
4 elementi di memoria  
x 2 bit ciascuno



Problemi con il crescere del numero di linee. Esempio: SRAM  $2M \times 16$  ( $2M$  elementi  $\times$  16 bit)  
Decodificatore a  $21$  ( $\log_2 2M$ ) bit. 16 uscite.  $2M$  linee di abilitazione e di selezione (ingresso C) dei bistabili.



## Migliore strutturazione



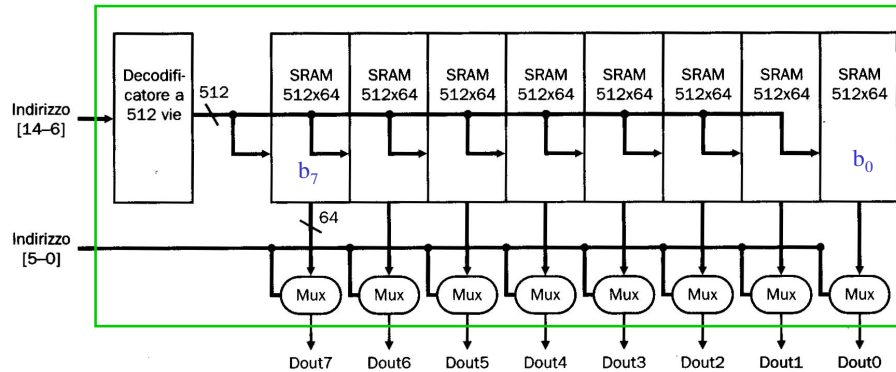
- C'è un limite elettrico al numero di linee che si possono collegare assieme.
- È più conveniente separare la lettura delle linee dalla lettura delle colonne (estrazione di una linea «lunga» dalla memoria).



## Indirizzamento SRAM a matrice



Esempio: SRAM 32K x 8 bit. Trasformo 32K linee in una matrice: 512 linee x 64 elementi.  
Ciascun elemento della matrice contiene 8 bit.



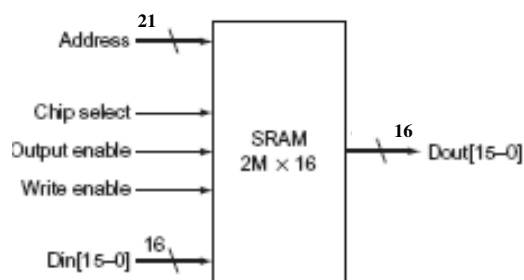
Il decodificatore sarà a 9 bit ( $\log_2 512$ ) per selezionare una delle 512 linee (cf. cache). Ciascuna linea fornisce 64 bit. Ne seleziono uno con il Mux (controllato dai 6 bit meno significativi).

Nell'approccio non a matrice avrei avuto bisogno di un decodificatore a 15 bit ( $\log_2 32K$ ). Qual è il vantaggio?

Quale gruppo di 64 bit associati al  $b_j$  bit della parola su 8 bit leggo dalle SRAM?



## Chip di SRAM (2M x 16)



Tempo di accesso:  
da Address a Dout.

Selezione – indirizzo + Chip select.

Scrittura – Write enable.

Letture – Output enable.

**E' una memoria veloce, ma costosa.**

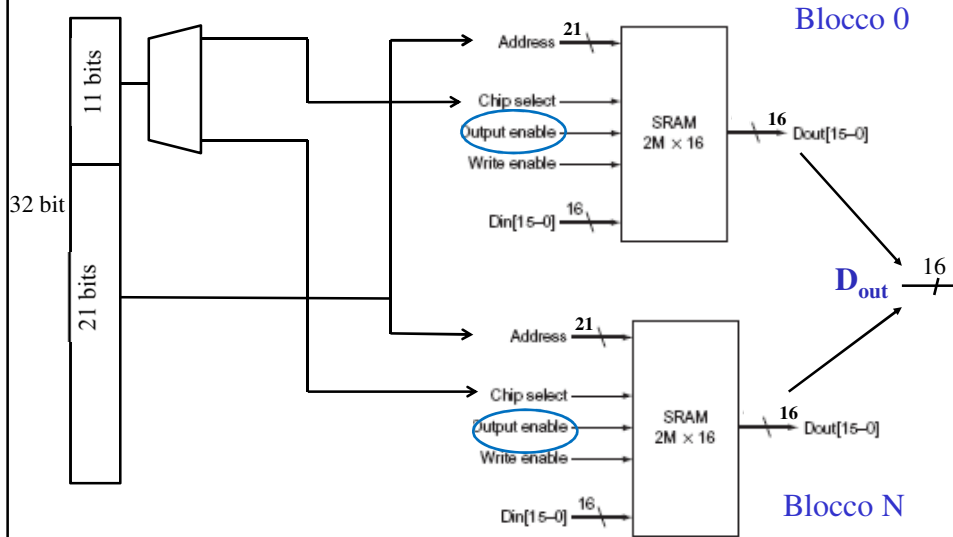




## Memoria RAM come insieme di chip



Capacità totale =  $2M * N$  Byte



## Sommario



SRAM

DRAM

Codici di errore



## Memorie DRAM - scrittura

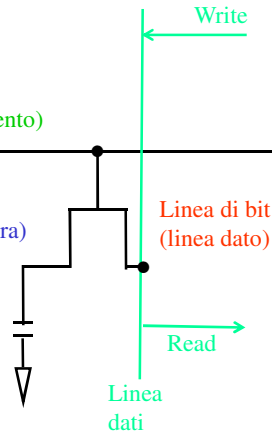


Dynamic RAM. Condensatore che viene caricato.  $\forall$ bit (1 condensatore contro 4-6 SRAM).

Linea di parola (indirizzamento)

Pass transistor  
(transistor di lettura / scrittura)

Condensatore  
(cella di memoria)



Scrittura: carica / scarica

La scrittura carica (memoria = 1) o scarica (memoria = 0) il condensatore.

1 Pass transistor + 1 condensatore.



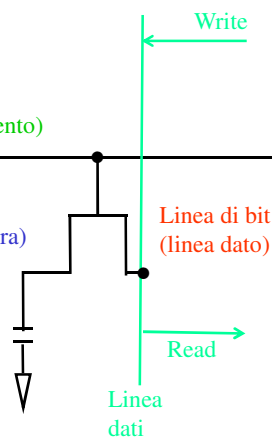
## Memorie DRAM - lettura



Linea di parola (indirizzamento)

Pass transistor  
(transistor di lettura / scrittura)

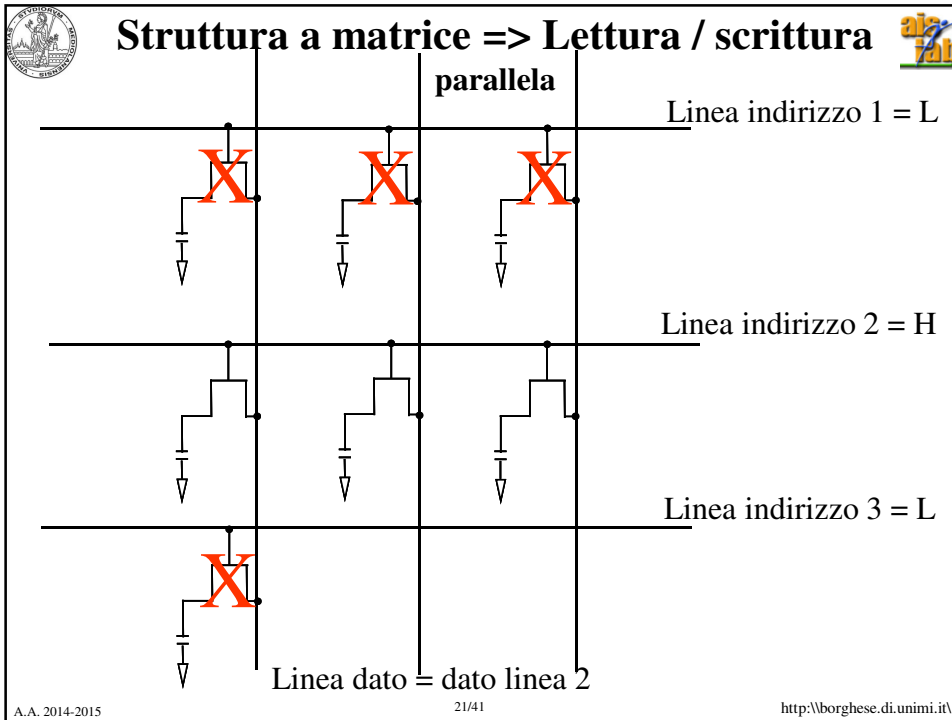
Condensatore  
(cella di memoria)



Lettura: la linea di bit è portata ad una tensione intermedia, e poi tirata verso low o high a seconda che il condensatore sia carico / scarico: amplificazione della carica.

1 Pass transistor + 1 condensatore.

La lettura scarica la memoria che deve essere ricaricata.



## I problemi delle DRAM

I condensatori .... si scaricano (qualche millisecondo)

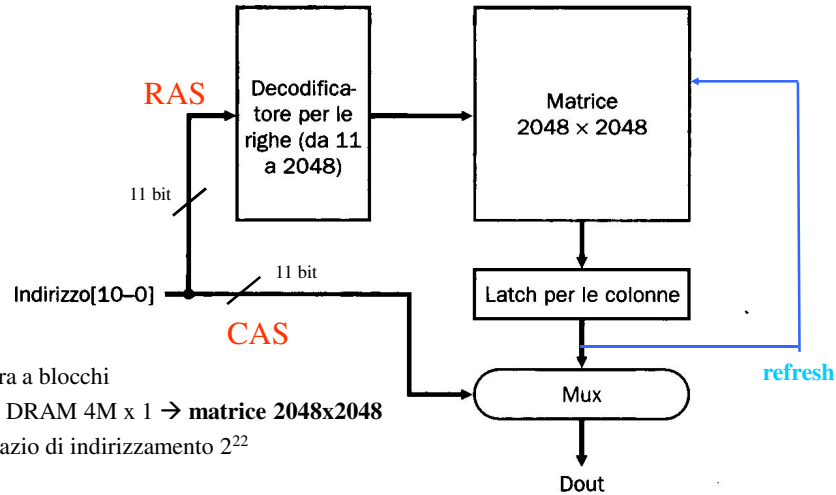
**refresh** gestito autonomamente dal controllore della memoria mediante ciclo lettura/scrittura

Cosa leggo/scrivo? Quale/i bit?

A.A. 2014-2015 22/41 <http://borghese.di.unimi.it/>



## Struttura a 2 livelli (matrice) di una DRAM



- Struttura a blocchi
  - es. DRAM 4M x 1 → matrice 2048x2048
  - Spazio di indirizzamento  $2^{22}$

•Accesso:  
 selezione riga (RAS) + selezione colonna (CAS)  
 Efficiente per il refresh (refresh di riga) – (35-70ms, tempo di carica dei condensatori).  
 Utilizzo per la Memoria Principale.

ese.di.unimi.it/



## Specifiche delle DRAM (Patterson and Hennessy, 2008)



Year introduced	Chip size	\$ per GB	Total access time to a new row/column	Column access time to existing row
1980	64 Kbit	\$1,500,000	250 ns	150 ns
1983	256 Kbit	\$500,000	185 ns	100 ns
1985	1 Mbit	\$200,000	135 ns	40 ns
1989	4 Mbit	\$50,000	110 ns	40 ns
1992	16 Mbit	\$15,000	90 ns	30 ns
1996	64 Mbit	\$10,000	60 ns	12 ns
1998	128 Mbit	\$4,000	60 ns	10 ns
2000	256 Mbit	\$1,000	55 ns	7 ns
2004	512 Mbit	\$250	50 ns	5 ns
2007	1 Gbit	\$50	40 ns	1.25 ns

Tempo di accesso della cache primaria = 0,16ns  
 Tempo di accesso della cache secondaria = 0.3ns



## Evoluzione delle SRAM e DRAM



Synchronous version.

Trasferimento *a burst* o a pagina: trasferimento consecutivo di parole ad indirizzi consecutivi (e.g. RAM EDO).

### SDRAM (Synchronous DRAM)

La fase di indirizzamento e di recupero dei dati vengono separate in modo da ridurre al minimo l'impatto della latenza.

Tra l'indirizzamento ed il recupero dei dati, il processore può eseguire altri compiti (NB il processore può essere la CPU o il controllore della memoria, o altro: il dispositivo che controlla la memoria).

**DDR-SDRAM.** Riescono a trasferire 2 bit per ciclo di clock. Frequenza doppia rispetto alla frequenza del clock del bus.



## Prestazione di una memoria



### Tempo di accesso.

Per una memoria *a random-access*, è il tempo richiesto per eseguire un'operazione di lettura / scrittura. Più precisamente è il tempo che intercorre dall'istante in cui l'indirizzo si presenta alla porta di lettura della memoria all'istante in cui il dato diventa disponibile.

Per una memoria *non random-access*, è il tempo richiesto per posizionare il dispositivo di lettura / scrittura in corrispondenza dell'informazione da leggere / scrivere.

**Memory cycle time:** si applica ad una memoria random-access. È il tempo di accesso più il tempo necessario perché possa avvenire un secondo accesso a memoria.

**Transfer rate:** quantità di informazione trasferita nell'unità di tempo si misura in:

Numero\_dati (in byte) / s.



## Organizzazione della memoria



Organizzazione a matrice (cf. cache)

Organizzazione gerarchica.



## Sommario



SRAM.

DRAM.

Codici di errore



## Errori nella memoria



Mean Time To Failure (MTTF) di un disco arriva a 1.000.000 di ore = 114 anni.

Un grosso centro di calcolo può contenere 50.000 server, ciascuno contiene facilmente 2 dischi per un totale di 100.000 dischi.

Numero di ore in un anno è:  $24 \times 365 = 8.760$  ore / anno

MTTF = 1.000.000 di ore corrisponde a un Annual Failure Rate (AFR) =  
 $8.760 / 1.000.000 = 0,876\%$

Un AFR di 0,876% corrisponde per 100.000 dischi a una media di 876 guasti per anno, cioè più di 2 guasti al giorno!

Da qui la ragione di proteggere i dati.

Per i dati in Memoria a semiconduttore si usano i codici di parità.



## ECC



- Errori dovuti a malfunzionamenti HW o SW.
  - Date le dimensioni delle memorie ( **$10^{10}$  celle**) la probabilità d'errore non è più trascurabile.
  - Per applicazioni sensibili, è di fondamentale importanza gestirli.
- **Codici rivelatori d'errore**
  - Es: codice di parità.
  - Consente di individuare errori singoli in una parola.
  - Non consente di individuare su quale bit si è verificato l'errore.
- **Codici correttori d'errore (error-correcting codes – ECC)**
  - Consentono anche la correzione degli errori.
  - Richiedono più bit per ogni dato (più ridondanza)
    - Per la correzione di 1 errore per parole e l'individuazione di 2 errori, occorrono 8bit /128 bit.



## Codice rilevatore di errore: il codice di parità



- **Es: Bit di parità (even):**
  - aggiungo un bit ad una sequenza in modo da avere un n. pari (even) di “1”
    - 0000 1010 0 ← bit di parità
    - 0001 1010 1
  - Un errore su uno dei bit porta ad un n. dispari di “1”
- Prestazioni del codice
  - mi accorgo dell’errore, ma non so dov’è
  - rivelo ma non correggo errori singoli
  - COSTO: 1 bit aggiuntivo ogni 8  $\rightarrow 9/8 = +12,5\%$



## Codici correttori d’errore



- **Es: Codice a ripetizione**
  - Ripeto ogni singolo bit della sequenza originale per altre 2 volte  $\rightarrow$  triplico ogni bit  
0 00 1 11 1 11 0 00 1 11 0 00 0 00 1 11 ...
  - Un errore su un bit di ciascuna terna può essere corretto:  
000  $\rightarrow$  010  $\rightarrow$  000  
111  $\rightarrow$  110  $\rightarrow$  111
- Prestazioni del codice
  - rivelo e correggo errori singoli
  - COSTO: 2 bit aggiuntivi ogni 1  $\rightarrow 3/1 = +200\%$





## Definizioni



- **Distanza di Hamming,  $d$**  (tra 2 sequenze di  $N$  bit)
  - il numero di cifre differenti, giustapponendole  
 $01001000$   
 $01000010 \rightarrow d = 2$
- **Distanza minima** di un codice,  $d_{MIN}$ 
  - il valor minimo di  $d$  tra tutte le coppie di sequenze di un codice
- **Capacità di rivelazione** di un codice:  $t = d_{MIN} - 1$
- **Capacità di correzione** di un codice:  $r = (d_{MIN} - 1) / 2$
- **Esempi:**
  - Codice a bit di **parità**:  $d_{MIN} = 2 \rightarrow t = 1, r = 0$
  - Codice a **ripetizione (3,1)**:  $d_{MIN} = 3 \rightarrow t = 2, r = 1$

**Codice ECC di Hamming (Turing award del 1968)**



## Come calcolare il codice ECC Hamming



- 1) Enumeriamo i bit partendo da sinistra: ( $d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8$ ).
- 2) Segnare le posizioni potenze di 2 come posizione dei bit di parità (1, 2, 4, 8, 16...)
- 3) I bit dei dati vanno inseriti in tutti gli altri bit (3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, ...)
- 4) La posizione del bit di parità determina di quali bit sarà responsabile ( $p_1$  di tutti i bit che hanno indirizzo che termina con 1: i numeri dispari;  $p_2$  di tutti i bit che hanno il secondo bit da destra = 1: 2,3 – 10,11; 6,7 – 110,111; 10,11 – 1010,1011,....).

Bit position	1	2	3	4	5	6	7	8	9	10	11	12
Encoded data bits	p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8
Parity bit coverage	p1	X		X		X		X		X		X
	p2		X	X			X	X			X	X
	p4				X	X	X	X				X
	p8								X	X	X	X

Le quattro cifre  $p_1 p_2 p_3 p_4$  ci dicono dove c'è un errore eventualmente e quindi ci dà la possibilità di correggerlo (e.g. se valessero 1011, l'errore sarebbe in posizione 11 ( $d_7$ )).

Con un parola a 12 bit possiamo correggere errori su dati a 8 bit.



## Esempio



Dato: 10011010 = 154 in base 10

1	2	3	4	5	6	7	8	9	10	11	12
0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100
$p_1$	$p_2$	$d_1$	$p_3$	$d_2$	$d_3$	$d_4$	$p_4$	$d_5$	$d_6$	$d_7$	$d_8$

Determino i 4 bit di parità:

$p_1$ . Considero i bit la cui posizione contiene un 1 nel bit meno significativo: 1, 3, 5, 7, 9, 11:

$p_1$   $p_2$  1  $p_3$  0 0 1  $p_4$  1 0 1 0 => bit di parità  $p_1 = 0$

$p_2$ . Considero i bit la cui posizione contiene un 1 nel secondo bit meno significativo: 2, 3, 6, 7, 10, 11:

$p_1$   $p_2$  1  $p_3$  0 0 1  $p_4$  1 0 1 0 => bit di parità  $p_2 = 1$

$p_3$ . Considero i bit la cui posizione contiene un 1 nel terzo bit meno significativo: 4, 5, 6, 7, (12, 13, 14, ...):

$p_1$   $p_2$  1  $p_3$  0 0 1  $p_4$  1 0 1 0 => bit di parità  $p_3 = 1$

$p_4$ . Considero i bit la cui posizione contiene un 1 nel quarto bit meno significativo: 8,9,10,11,..

$p_1$   $p_2$  1  $p_3$  0 0 1  $p_4$  1 0 1 0 => bit di parità  $p_4 = 0$

Parola memorizzata: 0 1 1 1 0 0 1 0 1 0 1 0



## Errore di 1 bit



Codice memorizzato:

0	1	1	1	0	0	1	0	1	0	1	0
1	2	3	4	5	6	7	8	9	10	11	12
$p_1$	$p_2$	$d_1$	$p_3$	$d_2$	$d_3$	$d_4$	$p_4$	$d_5$	$d_6$	$d_7$	$d_8$

Supponiamo un errore nel bit  $d_6$  (bit 10) -> dato letto: 1001 **1**10 invece di 1001 1010. La parola di memoria conterrà:

0	1	1	1	0	0	1	0	1	1	1	0
1	2	3	4	5	6	7	8	9	10	11	12
$p_1$	$p_2$	$d_1$	$p_3$	$d_2$	$d_3$	$d_4$	$p_4$	$d_5$	$d_6$	$d_7$	$d_8$

Analizzo i bit di parità (parità pari, 0 = numero pari di 1)

	prima	dopo												
P1: 0 ->	0	OK	0	1	1	1	0	0	1	0	1	1	1	0
P2: 1 ->	0	Not OK	0	1	1	1	0	0	1	0	1	1	1	0
P4: 1 ->	1	OK	0	1	1	1	0	0	1	0	1	1	1	0
P8: 0 ->	0	Not OK	0	1	1	1	0	0	1	0	1	1	1	0

I bit di parità che sono cambiati sono: 2 e 8 -> il bit dati che ha errore è il bit  $2+8 = 10$ .



## Esempio



Dato: 10011010 = 154 in base 10, dato letto: 10011110

Confronto i 2 codici:

Codice memorizzato: 0 1 1 1 0 0 1 0 1 0 1 0

Codice memorizzato con errore: 0 0 1 1 0 0 1 1 1 1 1 0

Dato: 10011010 = 154 in base 10. Dato letto con errore: 11011010 (bit 5 del codice)

Confronto i 2 codici:

Codice memorizzato: 0 1 1 1 0 0 1 0 1 0 1 0

Codice memorizzato con errore: 1 1 1 0 0 0 1 0 1 1 1 0

Analizzo i bit di parità

P1: 0->1 Not OK

P2: 1->1 Ok

P4: 1->0 Not OK

P8: 0->0 OK

I bit di parità che sono cambiati sono: 1 e 4 -> il bit dati che ha errore è il bit 1+4 = 5 (cioè d2).



## Dimensione di codici ECC



- Conviene applicare ECC a parole più lunghe possibile → aggiungo meno ridondanza → maggiore efficienza del codice
  - A costo di complessità maggiori di codifica/decodifica

Data Bits	Single-Error Correction		Single-Error Correction/ Double-Error Detection	
	Check Bits	% Increase	Check Bits	% Increase
8	4	50	5	62.5
16	5	31.25	6	37.5
32	6	18.75	7	21.875
64	7	10.94	8	12.5
128	8	6.25	9	7.03
256	9	3.52	10	3.91

**E' conveniente avere parole di memoria lunghe.**



## Applicazioni nelle memorie



- RAM con controllo di parità
  - Aggiungo un bit di parità ad ogni byte
  - Es: RAM 1 M x 9 bit (8+1)
- RAM con codice correttore di errori (ECC)
  - si usa nelle memorie cache
  - codici ECC evoluti (alta efficienza)
    - Hamming, CCITT-32, Reed Solomon, ...



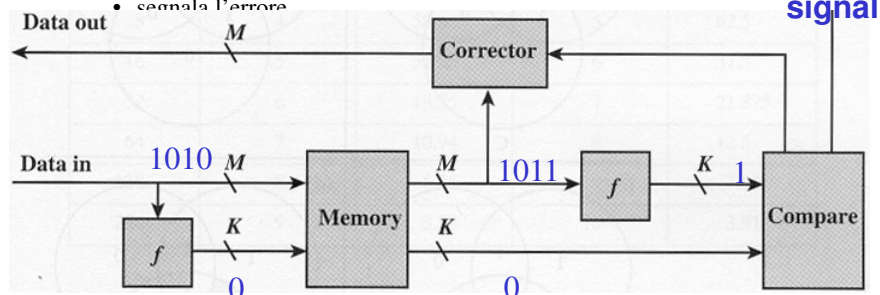
## Correzione degli Errori



Il controllore della memoria gestisce l'accesso e il riconoscimento degli errori.

OUT possibili:

- No errors detected
  - I dati letti possono essere inviati in uscita così come sono.
- 1 errore è stato individuato e corretto
  - I bit del dato, più il codice associato vengono inviati al correttore, il quale provvede a correggere il dato.
- 1 errore individuato, ma impossibile da correggere
  - segnala l'errore



*f può essere ad esempio il codice di parità e  $K$  il codice di parità*



## Sommario



Memorie associative

Memorie n-associative

Codici di errore