Memorie

Tipi di memorie, organizzazione della memoria

Sandro Savino(sandro.savino@dei.unipd.it)
Department of Information Engineering, University of Padova

Argomenti:

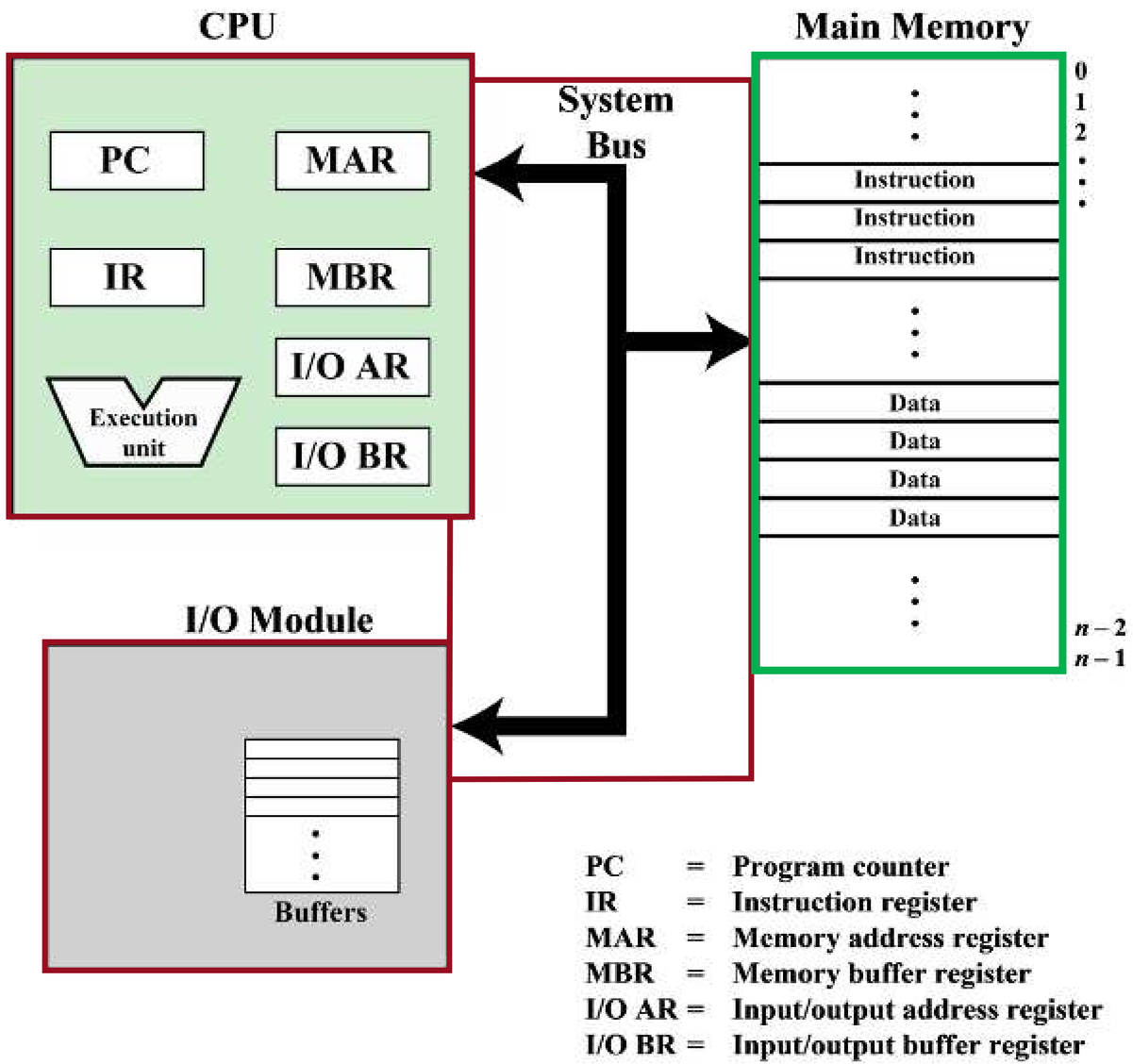
- Tipi di memorie
- SRAM / DRAM

Materiale:

- Capitolo 7

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Architettura di un elaboratore



Un semplice elaboratore è composto da:

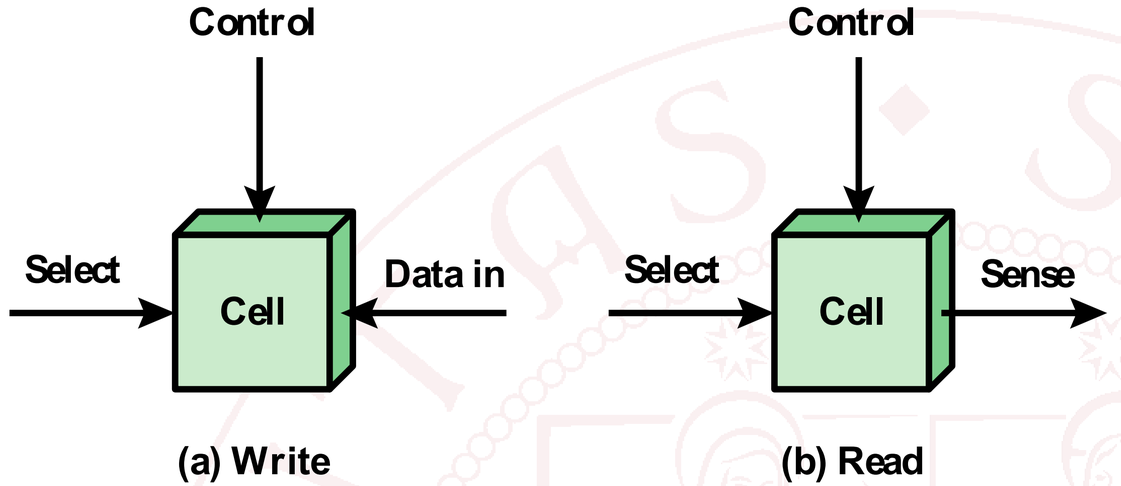
- CPU
- Modulo Input/Output
- Bus di sistema
- Memoria centrale

Le memorie vengono utilizzate in diversi componenti dell'elaboratore, ad esempio:

- CPU (MAR, MBR)
- I/O Module (buffers)
- Memoria centrale

Definizione di memoria

- Una memoria è una collezione di celle in grado di immagazzinare informazioni binarie
- Ogni cella può assumere valore 1 o 0
- Le informazioni vengono recuperate dalla memoria, processate dall'ALU e il risultato viene re-immagazzinato nella memoria (nella stessa o in una diversa locazione)



- Anche nel caso di I/O devices, le informazione da/per il device vengono temporaneamente immagazzinate in memorie (buffer)

Tipi di memorie

- Esistono vari modi di classificare le memorie:
  - Memorie sequenziali / Memorie ad accesso casuale

Memorie Sequenziali

- Tempo di accesso legato alla posizione

Memorie ad accesso casuale

- Tempo di accesso costante

Tipi di memorie

- Esistono vari modi di classificare le memorie:
  - Memorie persistenti / Memorie volatili

Memorie ROM

- Memorie persistenti
- Possono essere lette più volte

Memorie RAM

- Memorie volatili
- Possono essere lette e scritte più volte

Tipi di memorie

- Esistono due grandi famiglie di memorie:
  - Random-Access Memory (RAM)
  - Read-Only Memory (ROM)

Memorie ROM

- Memorie persistenti
- Possono essere lette più volte

Memorie RAM

- Memorie volatili
- Possono essere lette e scritte più volte

Tipi di memorie

<table>
<tr><th>Memory Type</th><th>Category</th><th>Erasure</th><th>Write Mechanism</th><th>Volatility</th></tr>
<tr><td>Random-access memory (RAM)</td><td>Read-write memory</td><td>Electrically, byte-level</td><td>Electrically</td><td>Volatile</td></tr>
<tr><td>Read-only memory (ROM)</td><td rowspan="2">Read-only memory</td><td rowspan="2">Not possible</td><td rowspan="5">Masks

Electrically</td><td rowspan="5">Nonvolatile</td></tr>
<tr><td>Programmable ROM (PROM)</td></tr>
<tr><td>Erasable PROM (EPROM)</td><td rowspan="3">Read-mostly memory</td><td>UV light, chip-level</td></tr>
<tr><td>Electrically Erasable PROM (EEPROM)</td><td>Electrically, byte-level</td></tr>
<tr><td>Flash memory</td><td>Electrically, block-level</td></tr>
</table>

Read-Only Memory (ROM)

- Il contenuto della memoria è **permanente** e non può essere cambiato
- Non è richiesta alimentazione per mantenere il valore in memoria
- I dati sono cablati nel chip durante il processo di fabbricazione
- Svantaggi:
  - Non ci devono essere errori perché non si possono correggere
  - Inserimento dati costoso

Programmable ROM (PROM)

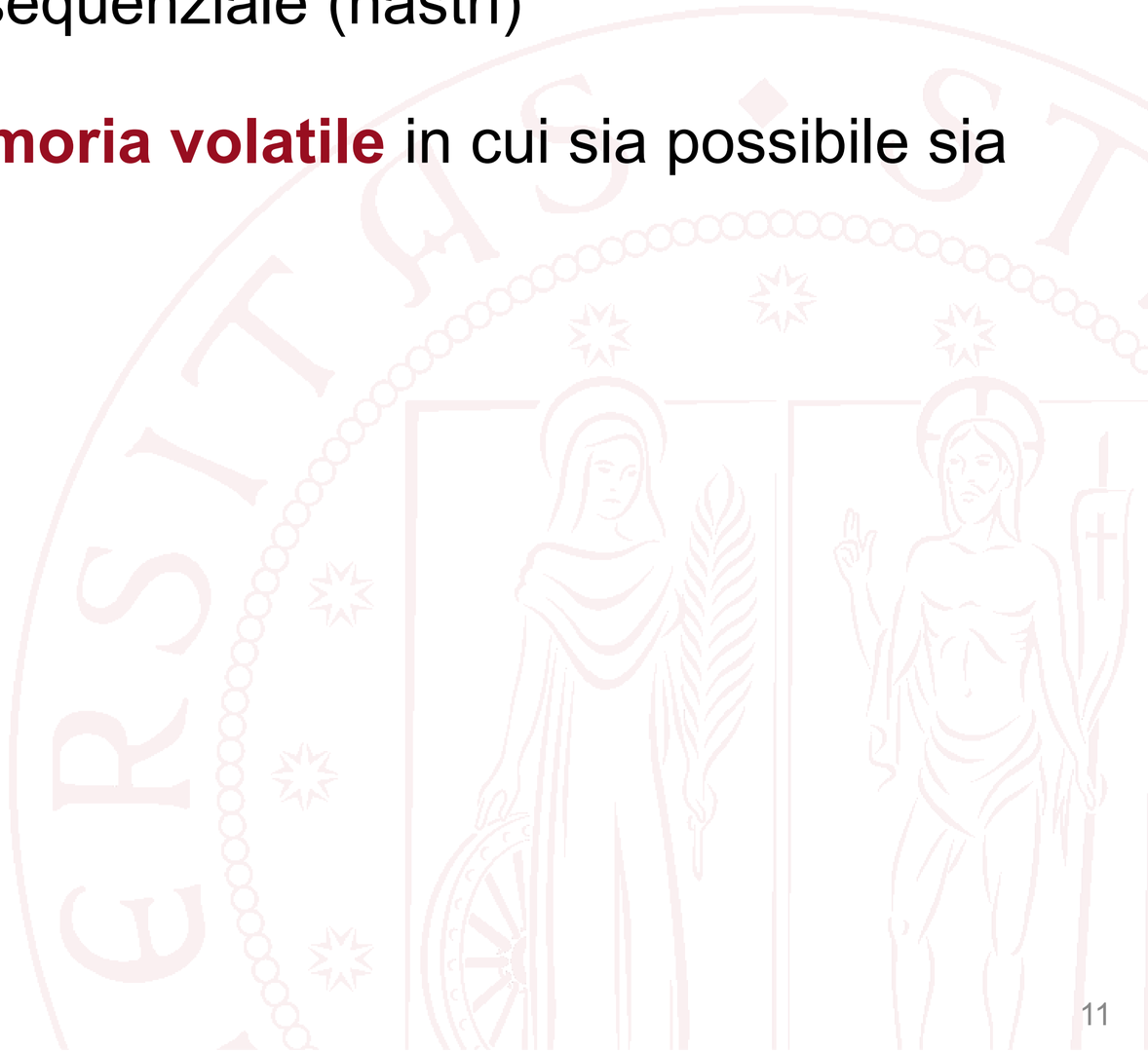
- Il contenuto della memoria è **permanente** e non può essere cambiato
- La scrittura può essere effettuata **dopo** il processo di fabbricazione del chip tramite un'opportuna apparecchiatura elettrica
- Non è richiesta alimentazione per mantenere il valore in memoria
- Maggior flessibilità e minor costo rispetto la ROM

Memorie persistenti con scrittura

- Le memorie persistenti permettono la modifica del loro contenuto
- **Erasable programmable read-only memory (EPROM)**
  - La cancellazione dell'intera memoria avviene tramite radiazione ultravioletta
- **Electrically erasable programmable read-only memory (EEPROM)**
  - E' possibile cancellare solo singoli byte tramite segnali elettrici
  - Più costosa dell'EPROM e con densità minore (meno bit per chip): ogni bit richiede due transistor
- **Flash Memory**
  - Posizione intermedia tra EPROM e EEPROM per costi e funzionalità
  - La cancellazione avviene per settori
  - La densità è paragonabile a quelle di una EPROM: ogni bit richiede un transistor

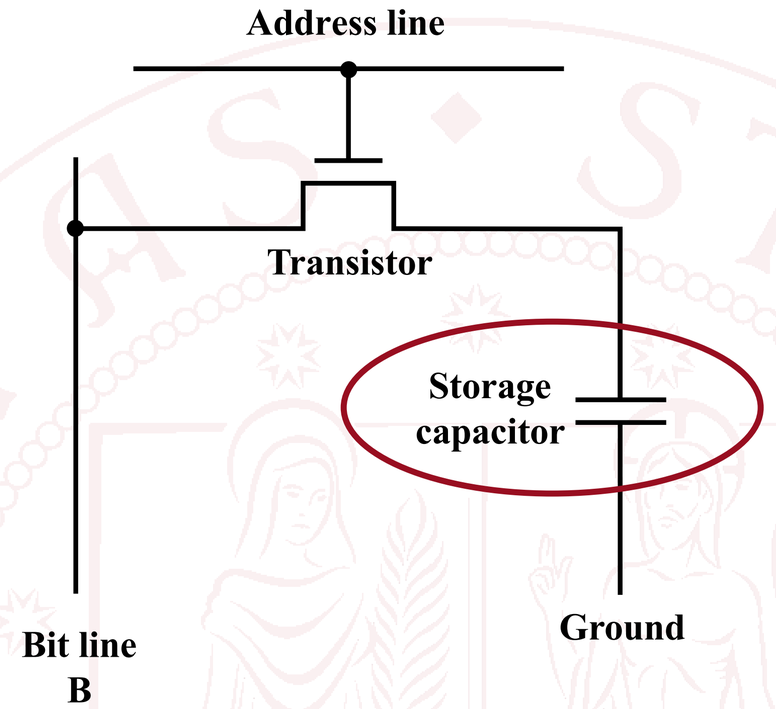
Random-Access Memory

- Tutte le memorie permettono l'accesso casuale alle locazioni di memoria
  - Si contrappone alle memorie ad accesso sequenziale (nastri)
- Con RAM si denota tradizionalmente una **memoria volatile** in cui sia possibile sia **leggere** che **scrivere** efficientemente
- Memorie RAM principali:
  - **Dynamic RAM (DRAM)**
  - **Static RAM (SRAM)**



Dynamic RAM (DRAM)

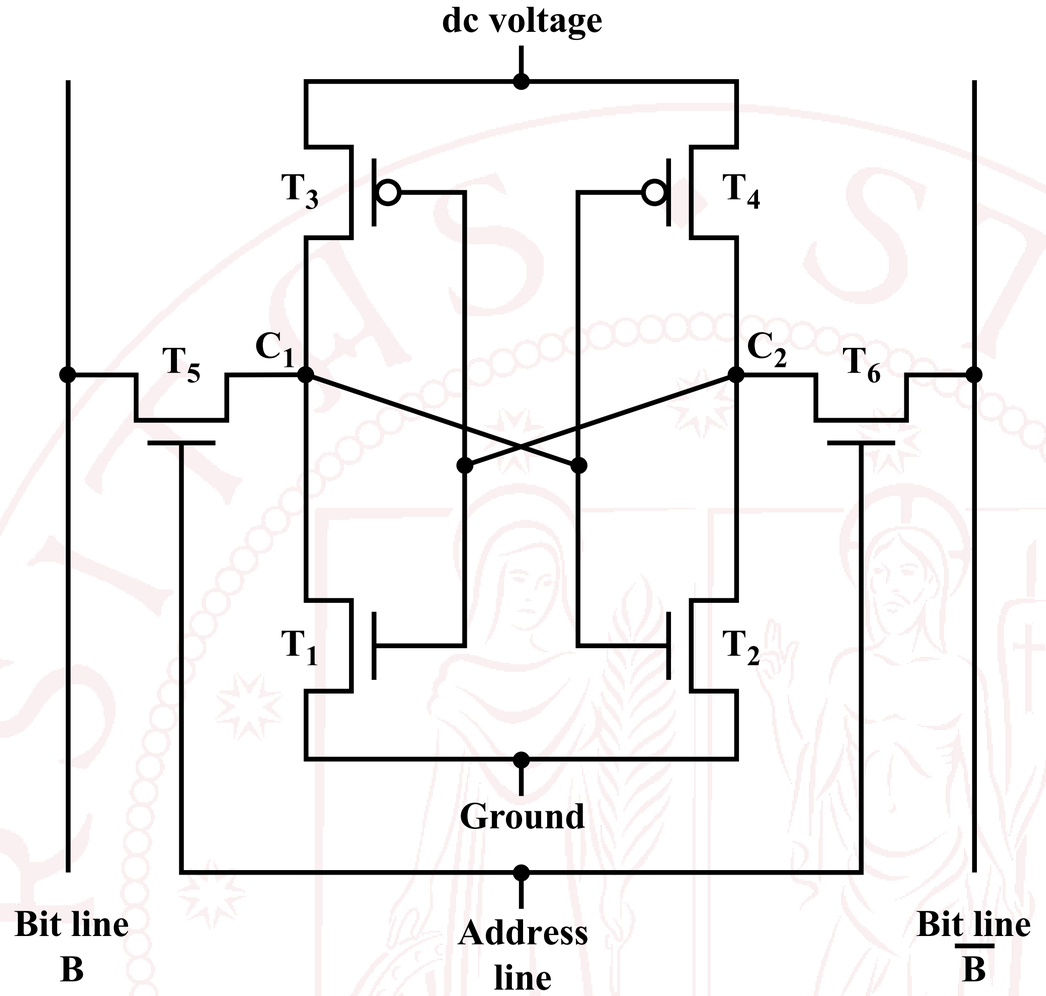
- Il valore di un bit viene rappresentato tramite la presenza o meno di una carica in un condensatore
- Richiede un **refresh periodico** per mantenere il valore
  - Un condensatore carico tende a disperdere la carica
  - Refresh: il dato viene letto e poi riscritto
- Alta densità (1 transistor, 1 condensatore), limitata velocità di accesso (carica condensatore)



(a) Dynamic RAM (DRAM) cell

Static RAM (SRAM)

- Il valore di un bit viene salvato tramite flip-flop
- Mantengono il valore finché è presente l'alimentazione
- Bassa densità (6 transistors), alta velocità di accesso



(b) Static RAM (SRAM) cell

DRAM vs. SRAM

DRAM		SRAM
• Memoria volatile	⟷	• Memoria volatile
• Facili da costruire, basso costo	⟷	• Alto costo
• Alta densità: 2 componenti per bit	⟷	• Bassa densità: 6 componenti per bit
• Limitata velocità di accesso	⟷	• Alta velocità di accesso
• Richiede refresh	⟷	• Non richiede refresh
• Utilizzate come memoria principale	⟷	• Utilizzata come cache

DRAM vs. SRAM: tempi di accesso

- Memorie statiche (SRAM):
  - 1.4 ÷ 15 ns (high speed)
  - 35 ÷ 100 ns (low power)

- Memorie dinamiche (DRAM):
  - 50 ÷ 70 ns (DRAM asincrone)
  - 7 ÷ 12 ns (SDRAM sincrone | Synchronous DRAM)

  (+ latenza: per il primo dato ci vuole un tempo 4-5 volte quello per i dati successivi)

Caratteristiche della memoria

- Le informazioni sono immagazzinate in memoria in gruppi di bits
- Ogni gruppo è chiamato **word**
- Un gruppo di **8 bit** è chiamato **byte**
- La maggior parte delle memorie usa words che sono multipli di 8 bit (1 byte)
- La parola **word** è usata per denotare l'insieme di bit/byte che compongono un dato in un'architettura
- La dimensione di una word è variabile
  - word è stato usato per indicare 16 bit (2 byte), 32 bit (4 byte), 64 bit (8 byte), …

Caratteristiche della memoria

- Un'unità di memoria è definite da:
  - Il **numero di words** che contiene
  - Il **numero di bits** in ogni word
- Ad ogni word è assegnato un **indirizzo (address)**
- Il numero di indirizzi va da 0 a $2^k - 1$ dove k è numero di address lines
- Per selezionare una word si applica il **k-bit address alla address line**
- Un decoder seleziona la parola corrispondente

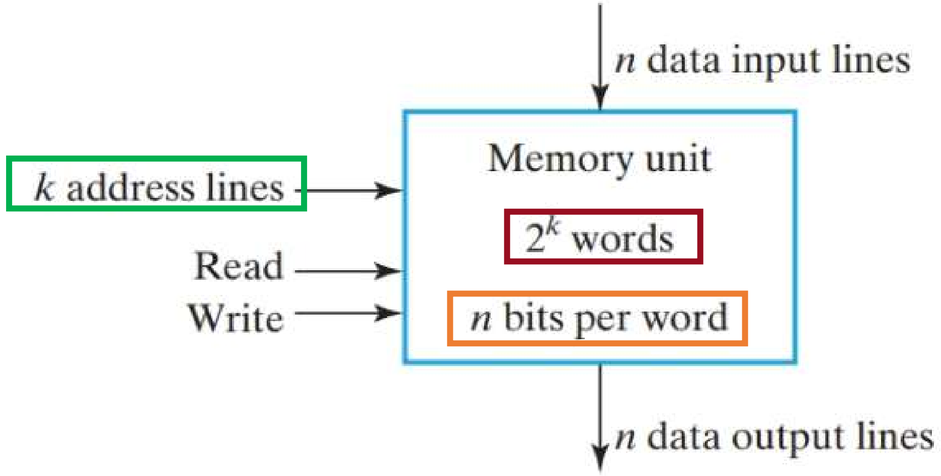


FIGURE 7-1
Block Diagram of Memory

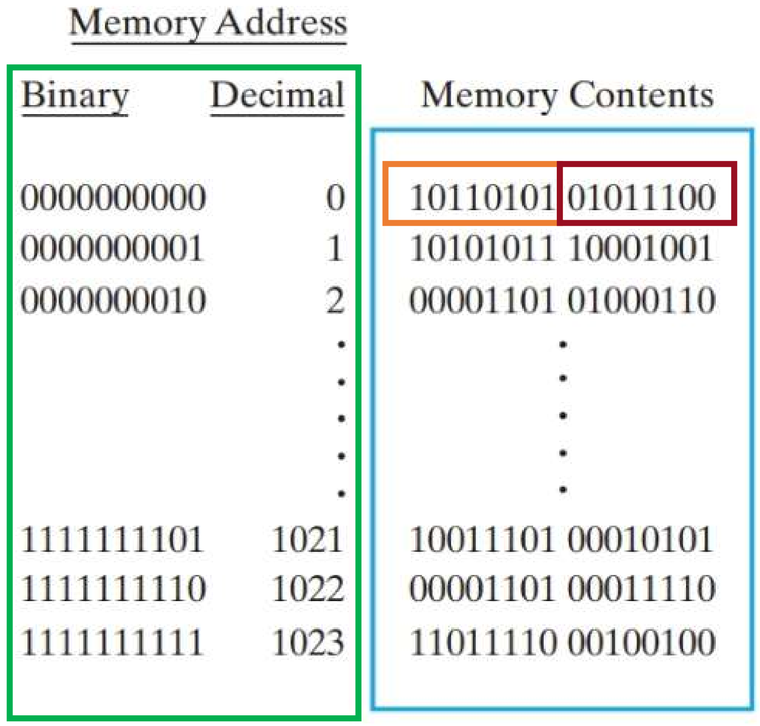
Nota: $2^{10} = 1024 = 1K$, $2^{20} = 1024 * 1024 = 1M$, $2^{30} = 1024 * 1024 * 1024 = 1G$

Caratteristiche della memoria

- La capacità totale della memoria è misurata in bytes

- Ad esempio, considerate una memora con capacità 1K words da 16 bits
  - $1K = 1024 = 2^{10}$
  - 16 bits = 2 bytes
  - La memoria contiene $2048 = 2K = 2 \cdot 2^{10}$ bytes

- I 1024 indirizzi sono identificati dai numeri:
  0 a 1023 → 0000000000 a 1111111111
- Ad ogni indirizzo c'è una word da 16 bit,
  cioè da 2 bytes
- Quando una word viene letta/scritta, la memoria opera su tutti i 16 bits
- Il numero di bit dell'address deve essere tale da poter indirizzare il numero di word presenti



Memory Address		Memory Contents
Binary	Decimal	
0000000000	0	10110101 01011100
0000000001	1	10101011 10001001
0000000010	2	00001101 01000110
⋮	⋮	⋮
1111111101	1021	10011101 00010101
1111111110	1022	00001101 00011110
1111111111	1023	11011110 00100100

FIGURE 7-2
Contents of a 1024 × 16 Memory

Ordinamento dei bytes in memoria

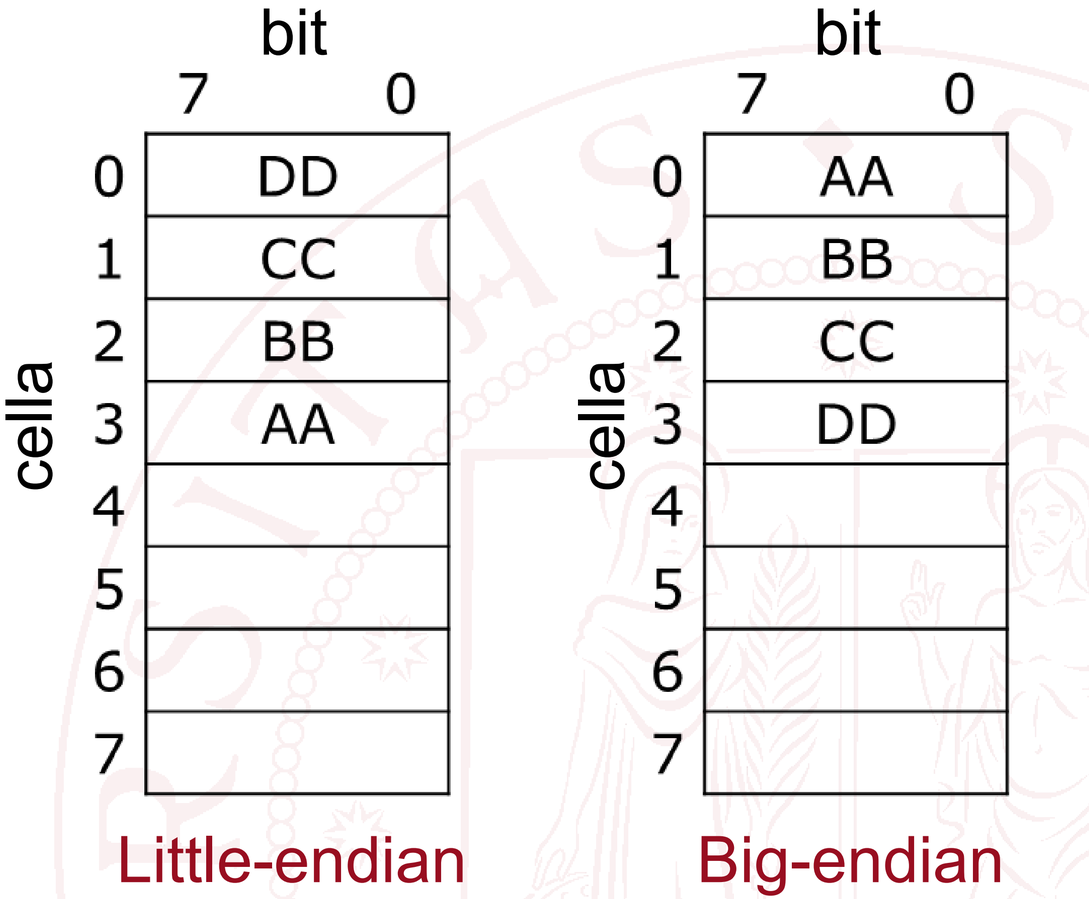
Problema:

Se la memoria è organizzata in byte (1 cella = 1 byte), come salvo una word (4 byte) in memoria?

word: 0xAABBCCDD

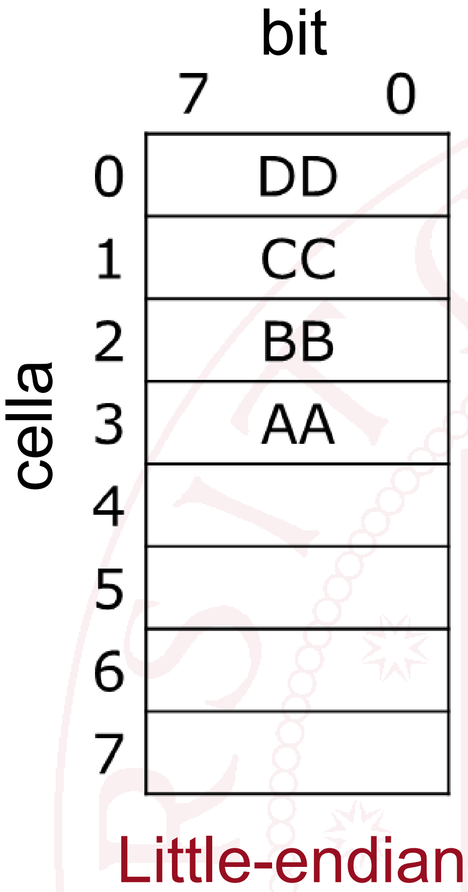
Due alternative:

- Big-endian
- Little-endian

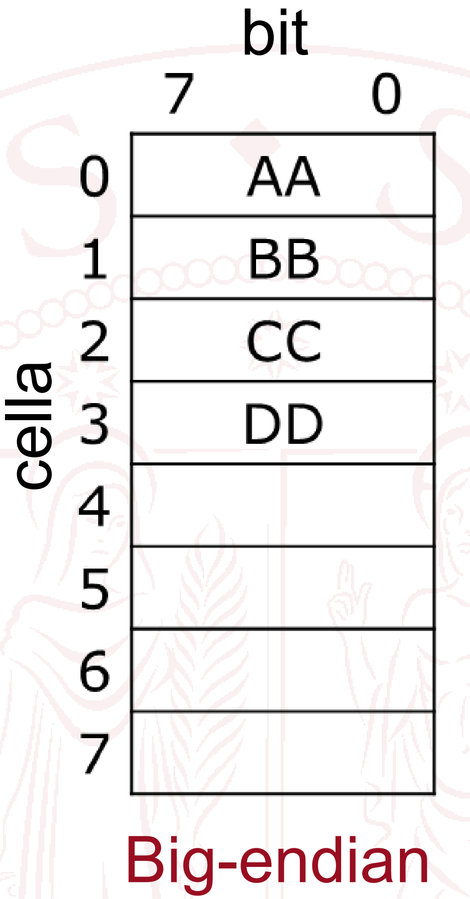


Big- e little-endian

- **Big-endian** è l'ordine per cui la parte più significativa (BIG END) viene memorizzata per prima (all'indirizzo più basso di memoria).
- **Little-endian** è l'ordine per cui la parte meno significativa (LITTLE END) viene memorizzata per prima.
- Le due alternative sono entrambe utilizzabili e utilizzate.



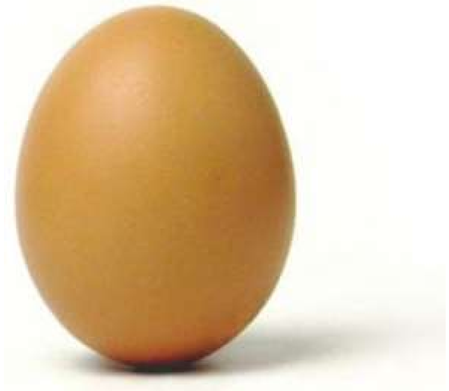
Little-endian

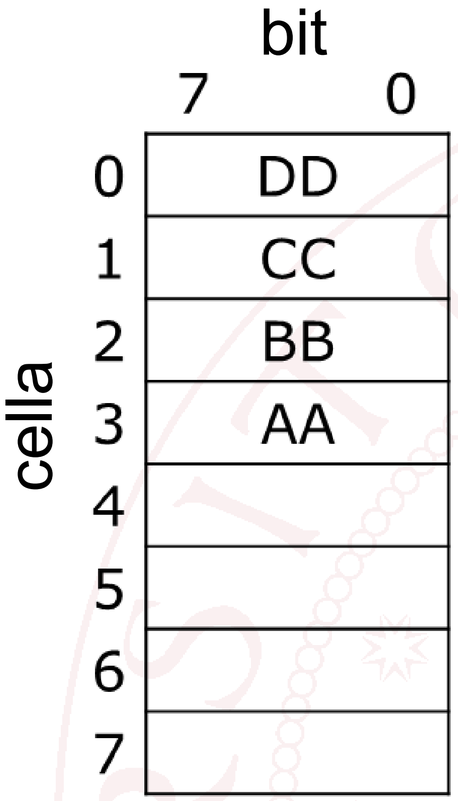


Big-endian

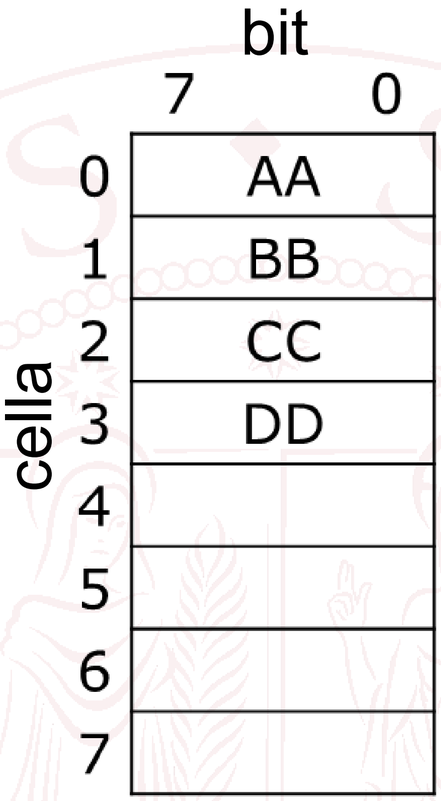
Big- e little-endian

- Nomi tratti dal libro “I viaggi di Gulliver” (Jonathan Swift)
- I Big Endians erano una fazione conservatrice che rompeva le uova sode dalla parte più larga del guscio, in contrapposizione al re dei Lillipuziani che richiedeva ai suoi sudditi (i Little Endians) di aprire le uova dalla punta.





Little-endian



Big-endian

Ordinamento dei bytes: big-endian o little-endian?

- I computer storici (IBM370) e i processori Motorola usano il metodo big-endian
- I processori INTEL adottano l'ordinamento little-endian, ritenuto più conveniente nella trasmissione dei dati, ove è trasmessa per prima la parte meno significativa

- Il PowerPC e l'ARM possono funzionare in tutte e due le modalità

- Anche nei formati grafici vi sono scelte diverse:
  - GIF → Little Endian
  - JPEG → Big Endian

SRAM: Static Random Access Memory

- SR Latch come modello di una cella SRAM
- S=1 memorizzo 1 R=0 memorizzo 0 S=R=0 hold

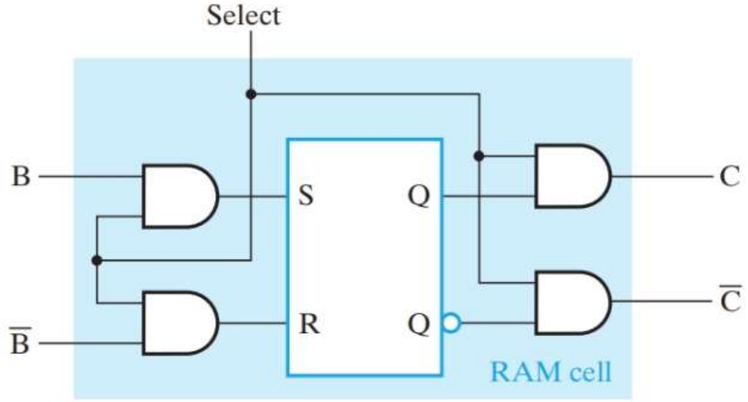
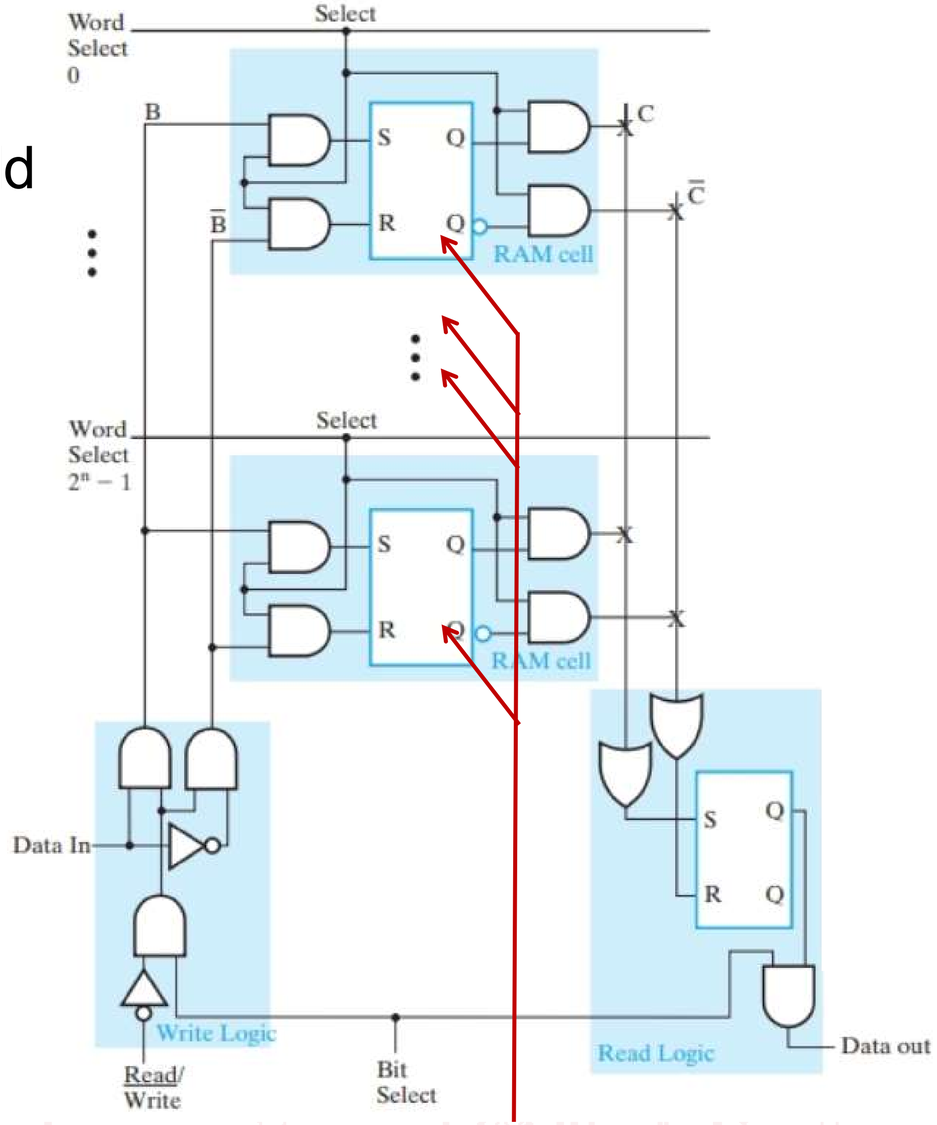


FIGURE 7-4
Static RAM Cell

- Con $Select = 1$, il contenuto della cella è determinato dal valore di $B, \overline{B}$
- Con $Select = 0$, entrambi $C, \overline{C} = 0$
- $Select$ abilita o meno l'accesso alla cella
- RAM bit slice → contiene i circuiti associati a una singola bit in un insieme di words
- la logica è di controllo, scrittura, lettura, è condivisa con lo stesso bit di altre word



bit k-esimo di
tutte le 2k words

SRAM: Static Random Access Memory

- SR Latch come modello di una cella SRAM
- S=1 memorizzo 1 R=0 memorizzo 0 S=R=0 hold

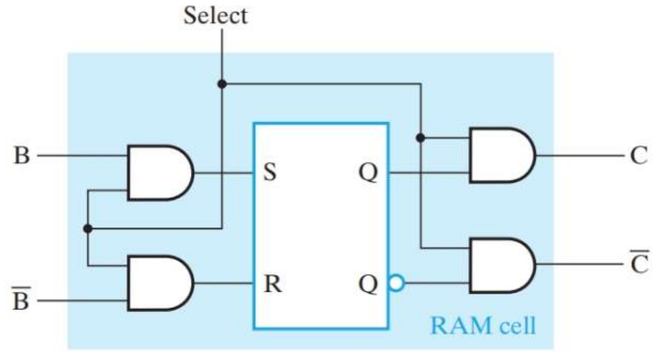
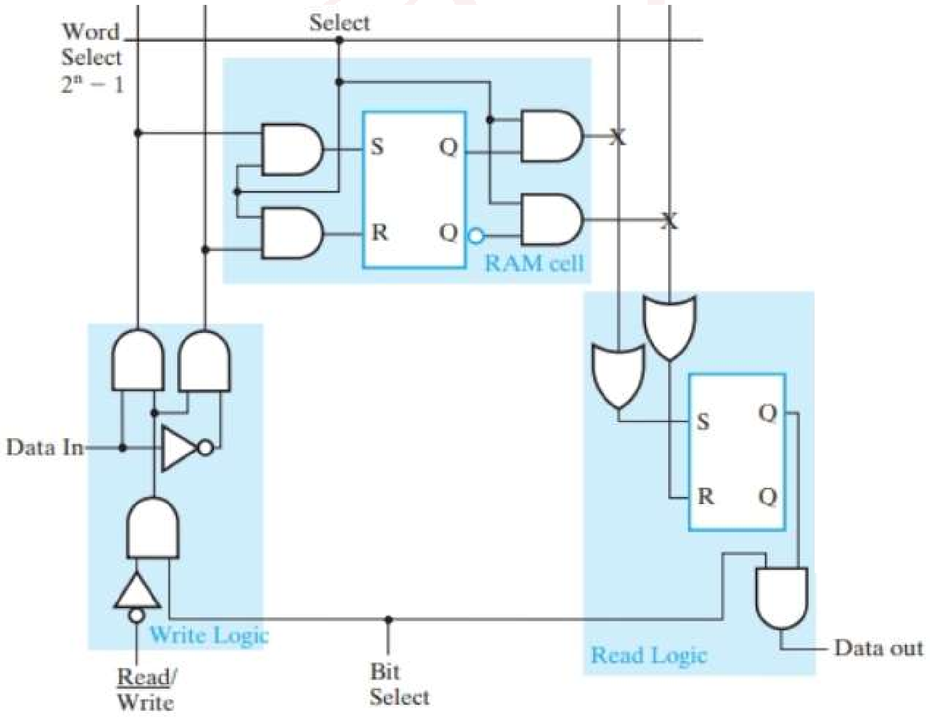


FIGURE 7-4
Static RAM Cell

- $Select$ abilita o meno l'accesso alla cella
- Con $Select = 1$, il contenuto della cella è determinato dal valore di $B, \overline{B}$

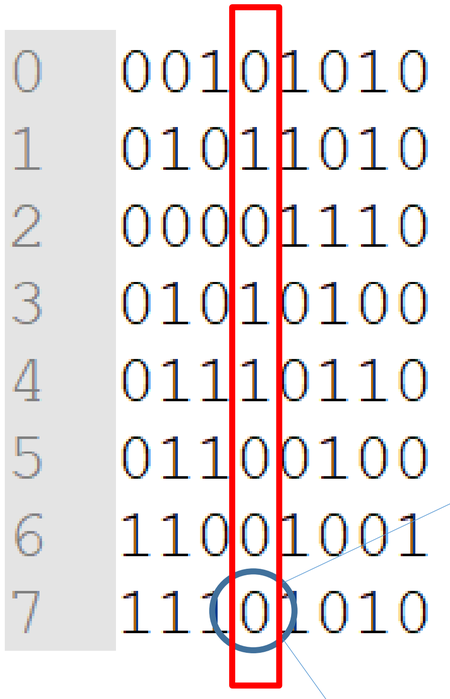
- Per leggere il bit ci serve inoltre:
  - Della logica per fare il write
  - Della logica per fare il read
  - Una linea di select

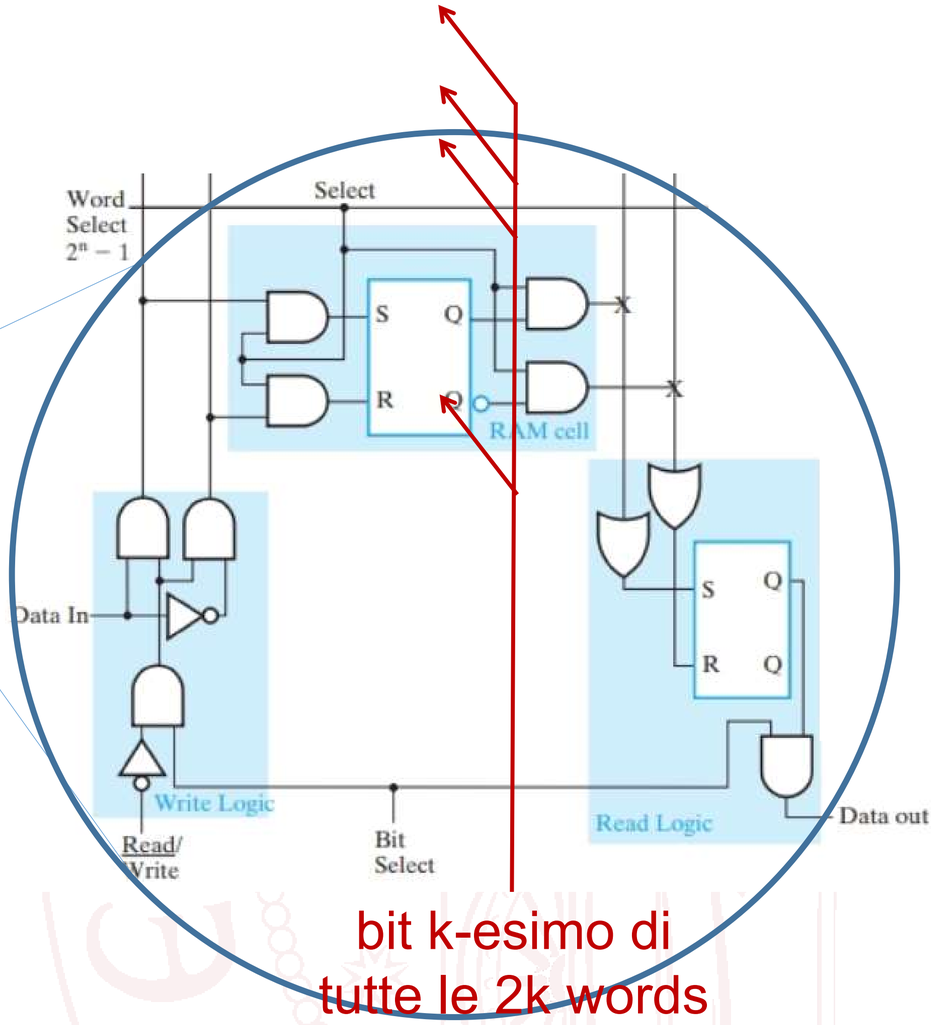


SRAM Bit Slice

- Ogni bit della nostra memoria ha bisogno di tutta questa logica
- Possiamo disegnare il circuito in modo che sia più economico?
- In memoria possiamo accedere a solo UNA locazione (=riga) per volta
- Possiamo decidere di condividere la logica di select, read, write del singolo bit con tutti i bit della stessa colonna!

- Chiamiamo questo componente BIT SLICE



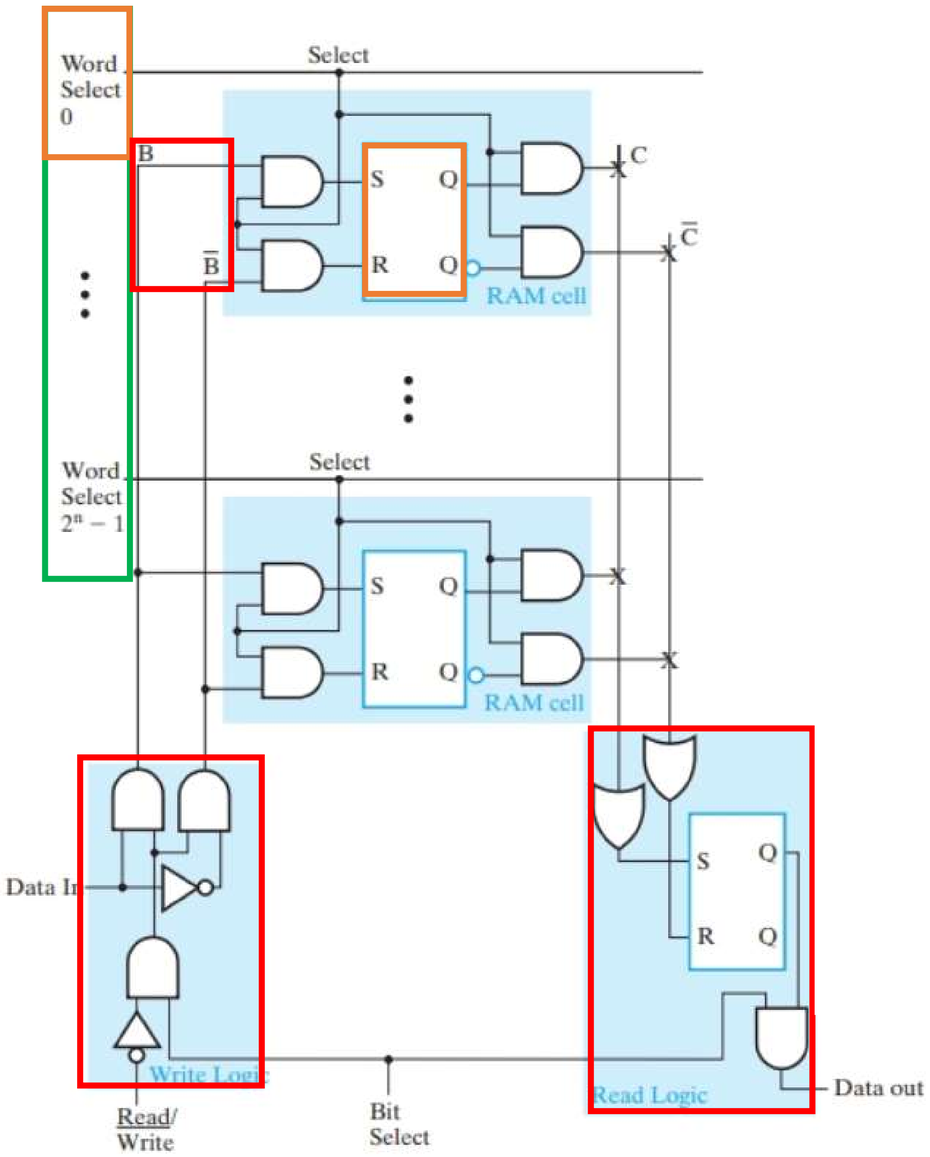


bit k-esimo di tutte le 2k words

SRAM

- La singola cella è controllata da un segnale Word Select
  - *Posso richiedere accesso al bit k nella word all'indirizzo* $0\ o\ 1\ o\ 2\ o\ \ldots\ 2^n - 1$
- Quando porto a 1 la linea «*Word Select 0*» posso accedere ai bit del word 0
- Se «*Word Select 0*» = *1 e ho il segnale di read,* il bit k della word 0 viene salvato nel flip-flop di uscita, pronto per essere letto
- Se «*Word Select 0*» = *1 e ho il segnale di write,* il bit k della word 0 viene alterato con i valori in ingresso $B, \overline{B}$ a loro volta controllati da Write Logic

- Questo vale per un bit della word
- Vengono usate m RAM bit slice affiancate per realizzare chip di memoria con word da m bit



Chip SRAM 16 word by 1-bit

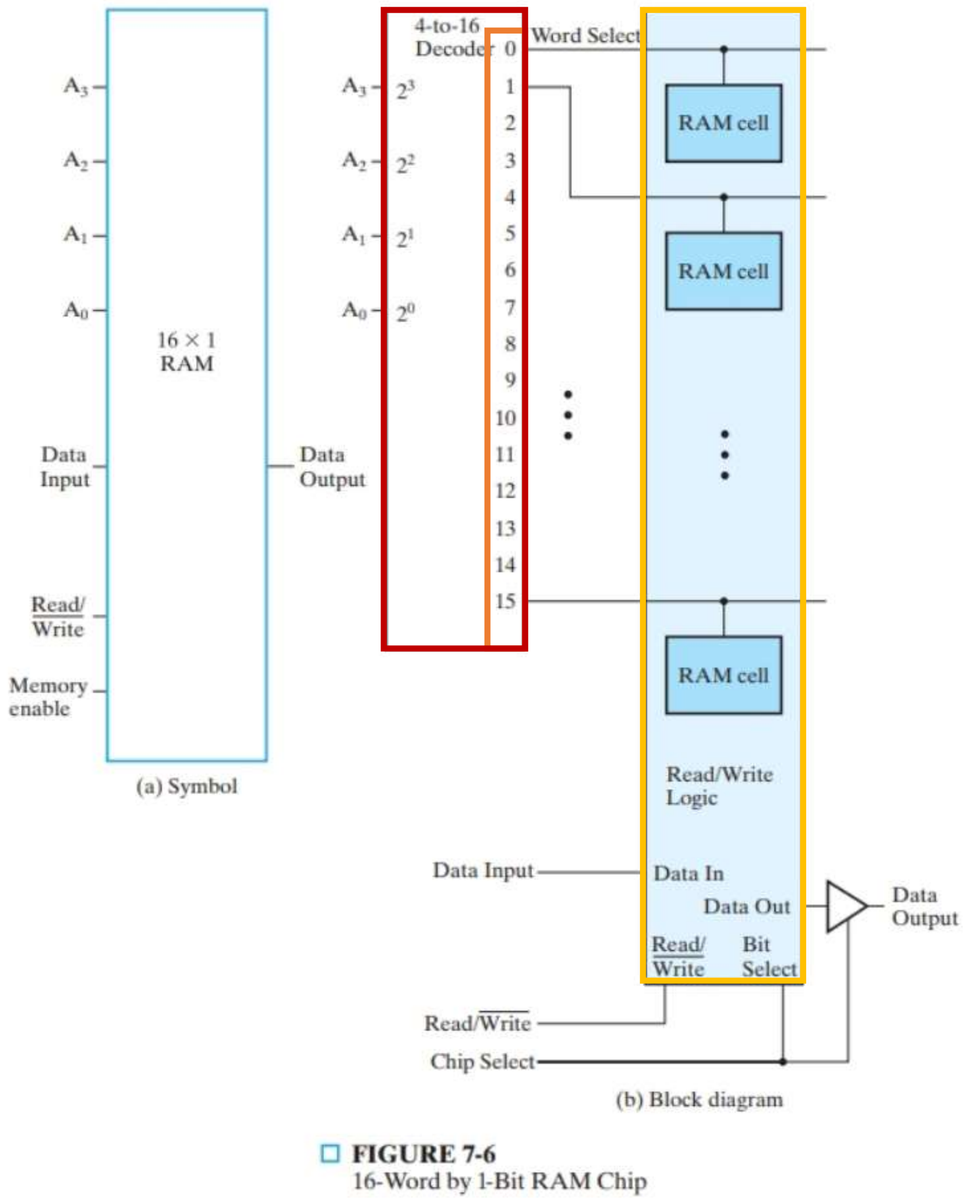


FIGURE 7-6
16-Word by 1-Bit RAM Chip

- Per semplicità, immaginiamo un chip SRAM con 16 words e un solo bit per word
- Questo corrisponde a un singolo RAM bit slice
- Per selezionare una word delle 16 abbiamo bisogno di un indirizzo a 4 bit
- Nello slice abbiamo bisogno di 16 word select
- Useremo un 4-to-16 decoder

In questo caso un indirizzo ha 4 bits ($16 = 2^4$)

A_3	A_2	A_1	A_0

Chip 16 x 1 SRAM con 4 x 4 RAM Array

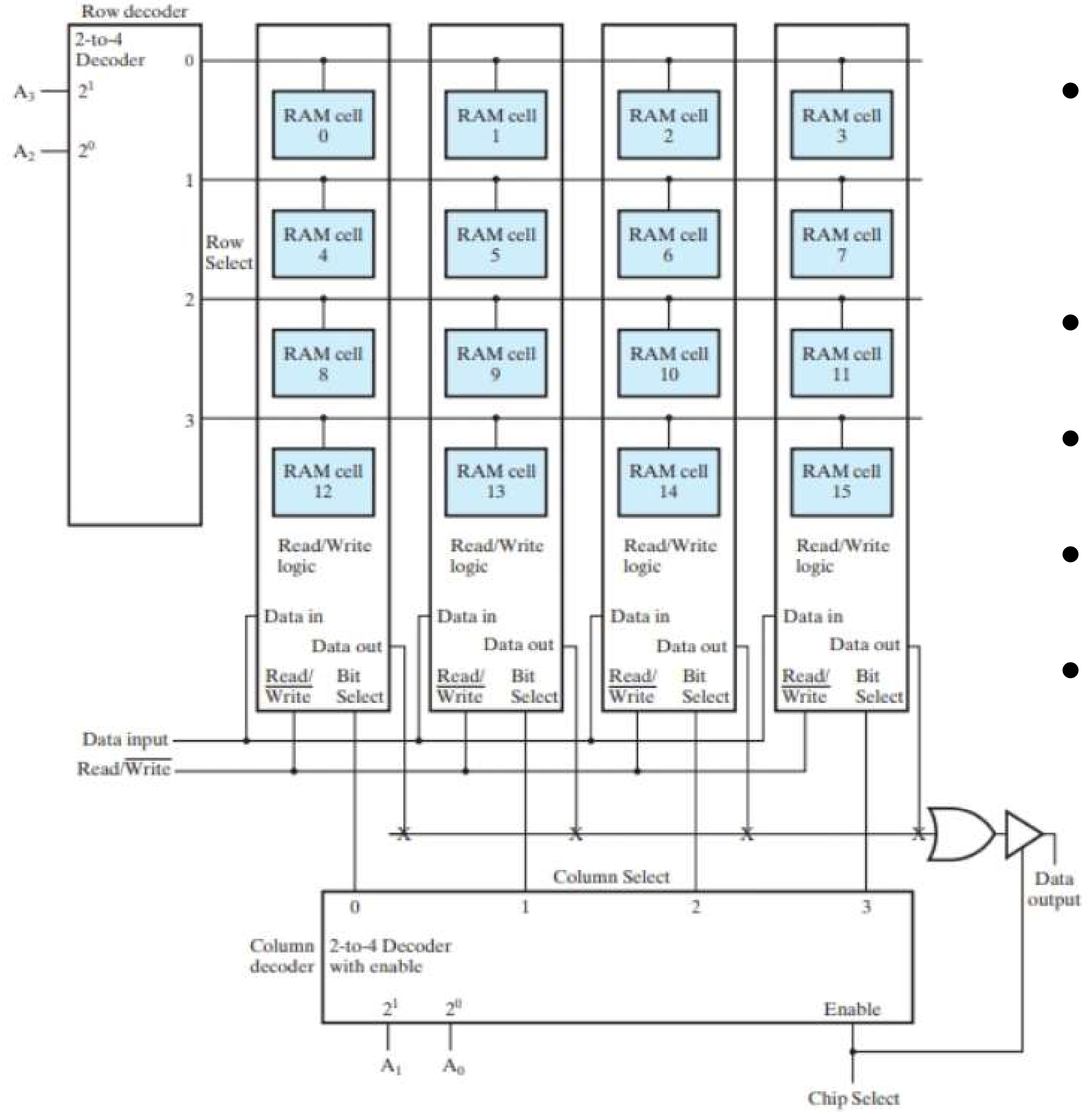
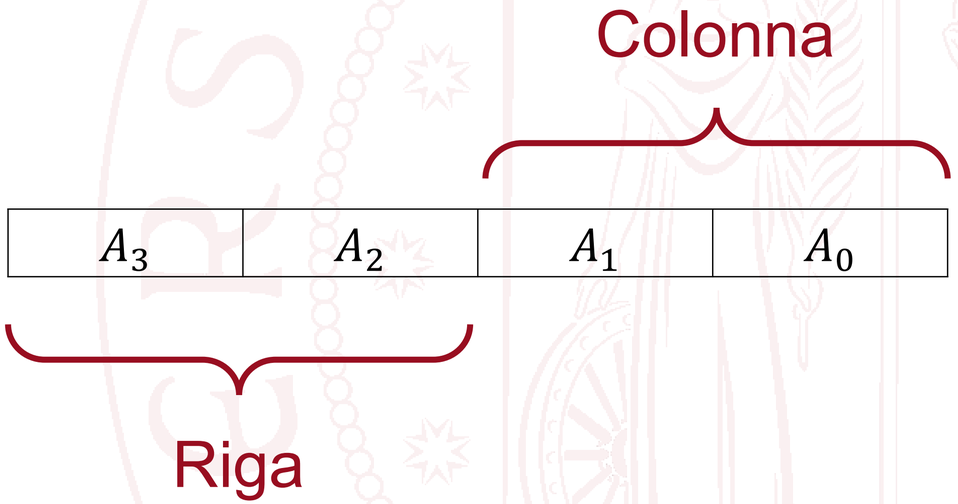


FIGURE 7-7
Diagram of a 16 × 1 RAM Using a 4 × 4 RAM Cell Array

- Possiamo organizzare diversamente i componenti interno al chip per risparmiare
- Facciamo una 2D matrix di RAM cell
- La selezione avviene per riga e colonna
- Indirizzo sempre a 4 bit
- Ma usiamo decoder più piccoli!



Chip SRAM con word a più bit

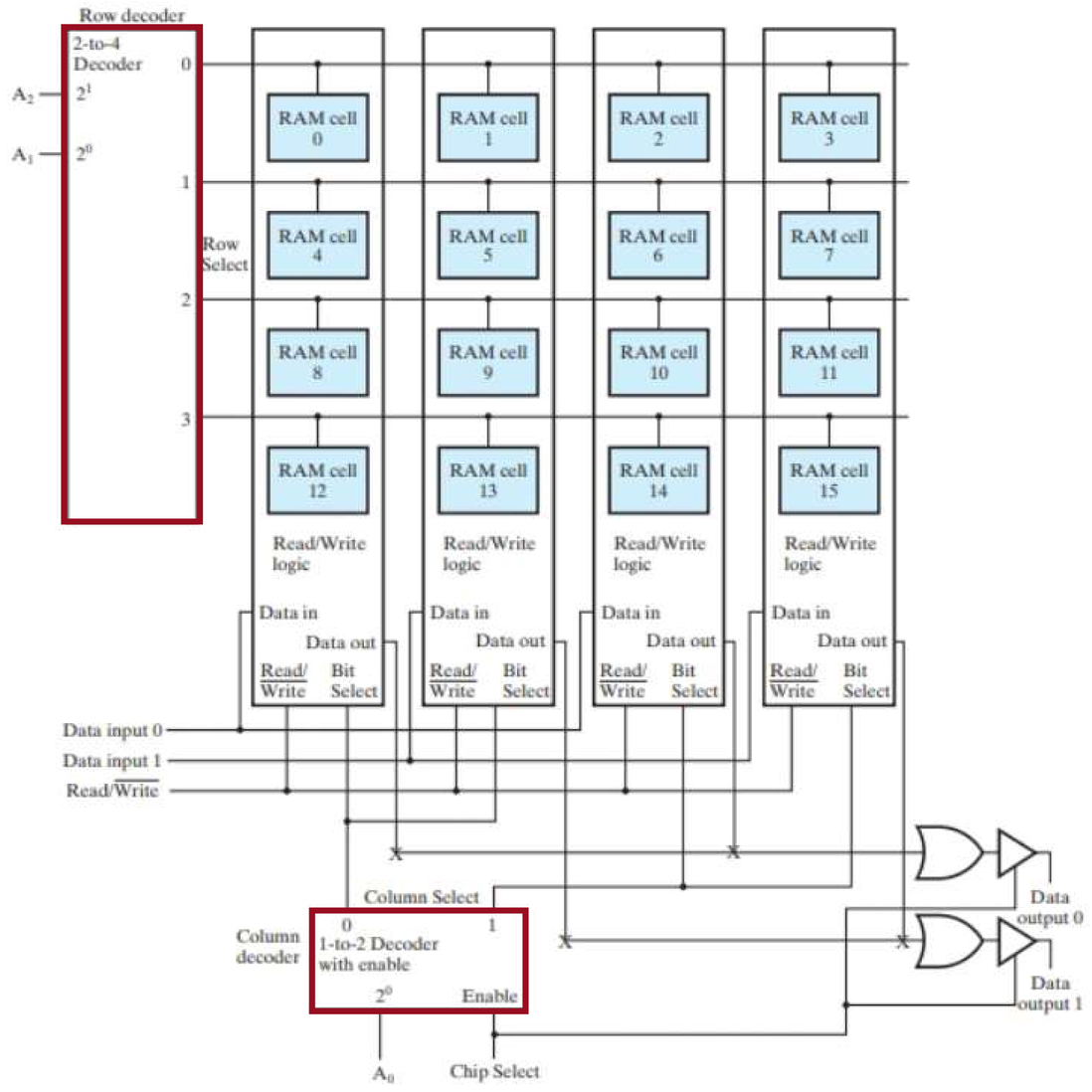
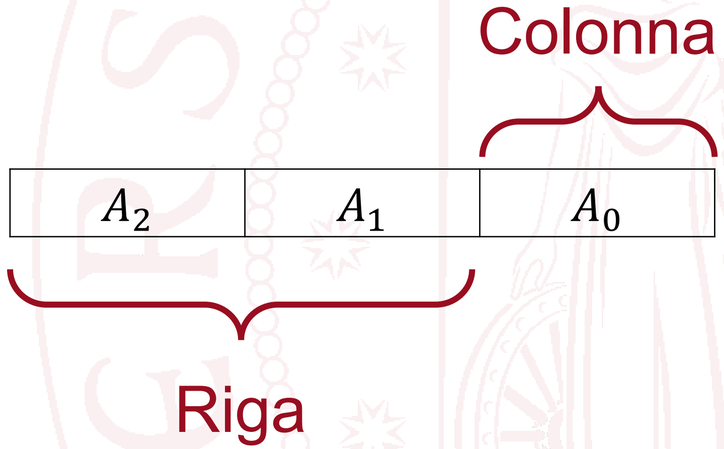


FIGURE 7-8
Block Diagram of an 8 × 2 RAM Using a 4 × 4 RAM Cell Array

- Chip SRAM 8 x 2 (8 words da due bits)
- Stesso approccio del caso precedente
- Numero di bit nell'indirizzo necessario per identificare una word: 3 ($8 = 2^3$)
- 2 bit dell'indirizzo per il row decoder
- 1 bit dell'indirizzo per il column decoder



Chip SRAM con word a più bit

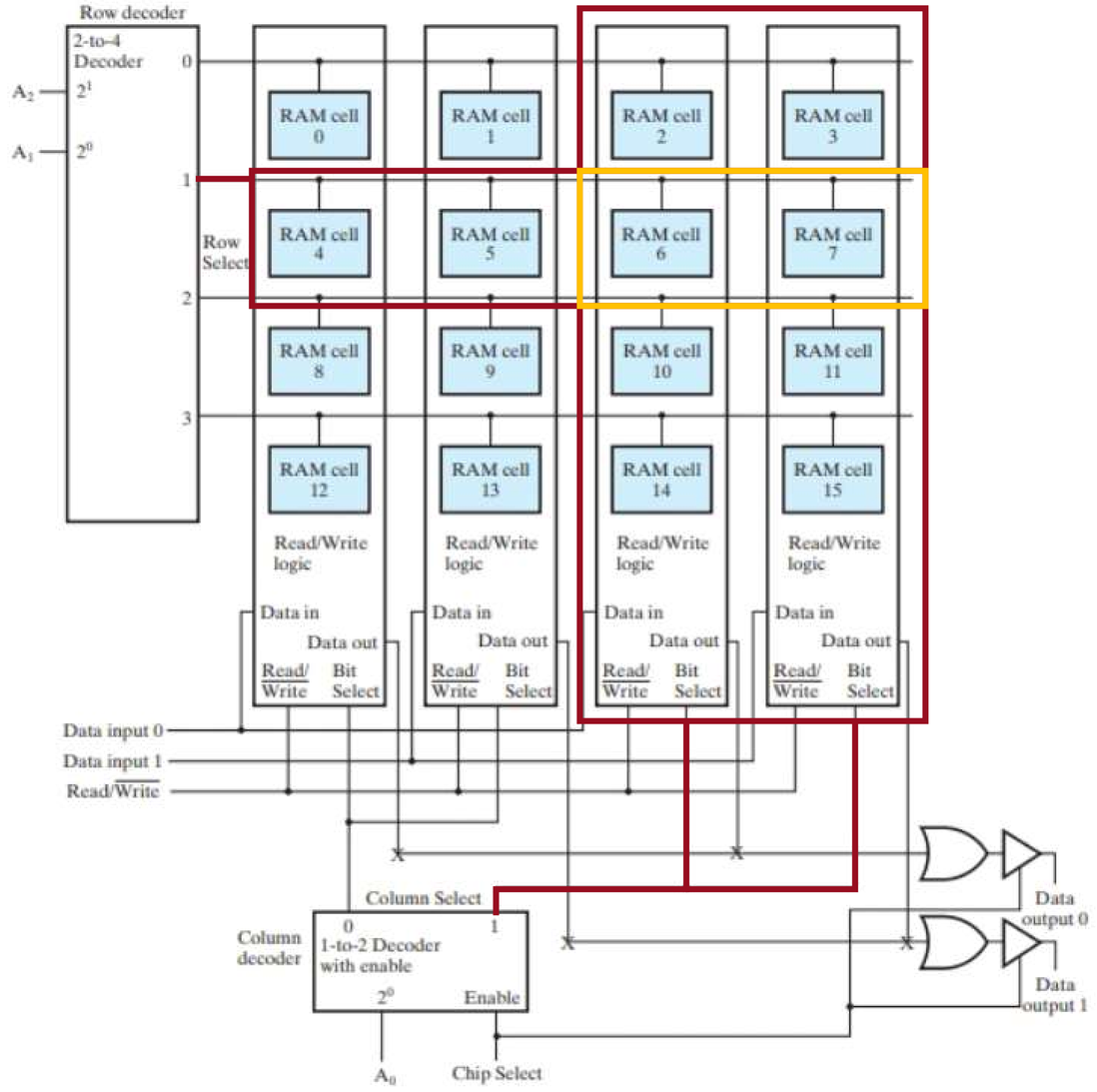


FIGURE 7-8
Block Diagram of an 8 × 2 RAM Using a 4 × 4 RAM Cell Array

Esempio: A = 011_2

A_2	A_1	A_0
0	1	1

- Row decoder:
  - Celle selezionate: 4, 5, 6, 7
- Column decoder:
  - Celle selezionate: 2, 6, 10, 14, 3, 7, 11, 15
- Word selezionata: celle 6 e 7 (2 bit)

Esempio SRAM chip

Cypress SRAM CT62256.pdf

- Vogliamo disegnare un chip SRAM 32K x 8
  - Cioè: con 32K word di lunghezza di 8 bit
- Il chip SRAM conterrà 256K bits (32K x 8)
- Il numero minimo di indirizzi date 32K words è k tale per cui: $2^k \geq words$ è verificata
- $2^k \geq 32 \cdot 1024 \Rightarrow k \geq \log_2 32768 \Rightarrow k \geq 15$
- Come organizziamo la matrice interna dei bit in bit slice?
  - quante righe (=altezza del bit slice) e colonne (=numero di bit slice affiancate)?
  - $nrighe, ncolonne = \sqrt{256K} = \sqrt{(256 \cdot 1024)} = 512 = 2^9$

9 bit

poichè indirizziamo i word (gruppi da 8 bit) e non i bit singoli, ci bastano 9-3=6 bit

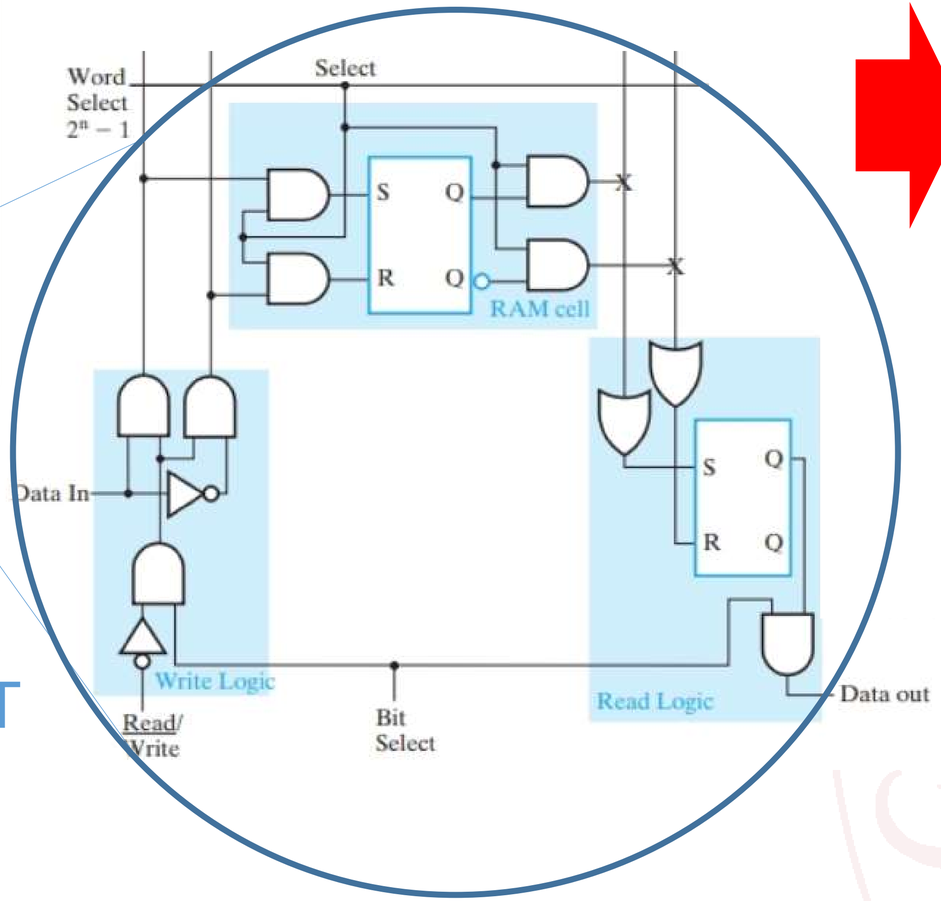
A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
Riga									Colonna					

Riassumendo

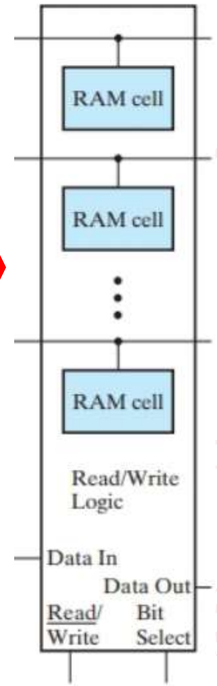
LOGICA DI UNA COLONNA DI BIT (RAM slice)

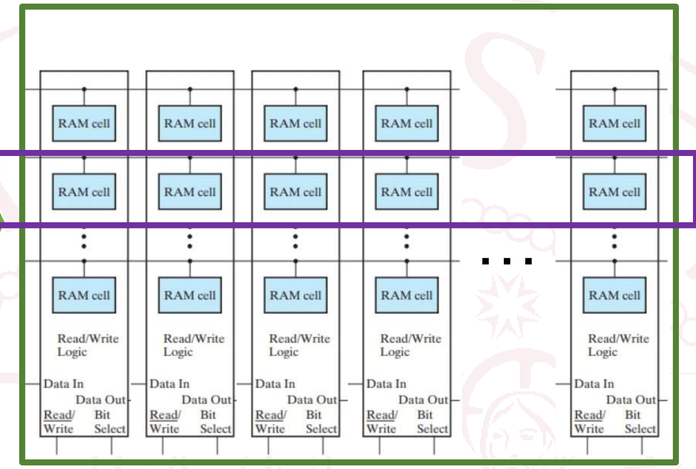
TANTI RAM SLICE = TUTTA LA RAM (organizzata come matrice)

0	00101010
1	01011010
2	00001110
3	01010100
4	01110110
5	01100100
6	11001001
7	11101010



LOGICA DI UN BIT





I bit di un WORD (locazione di memoria) sono memorizzati su più RAM slice

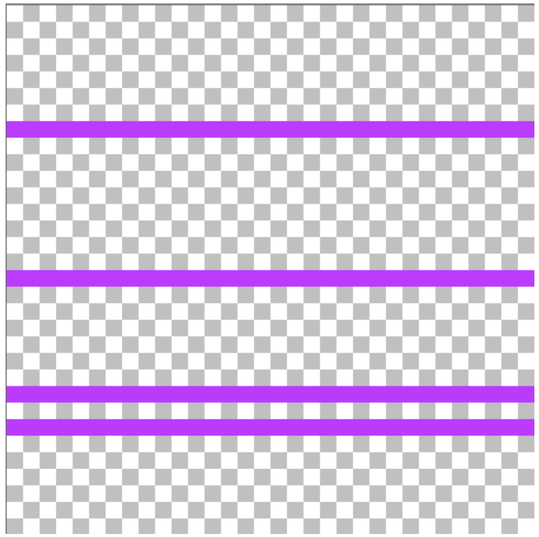
Riassumendo

Nei chip reali (di grandi dimensioni) un word non corrisponde ad una "riga" della matrice (sarebbe troppo grande!) ma solo ad una parte di riga:

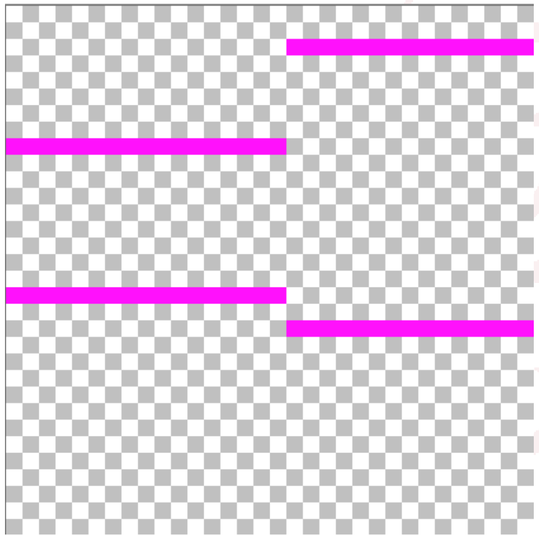
- Una riga contiene più word
- I bit di una word si trovano su n colonne adiacenti dove n è la lunghezza della word

Esempio:
chip di memoria con matrice di 32 x 32 bit

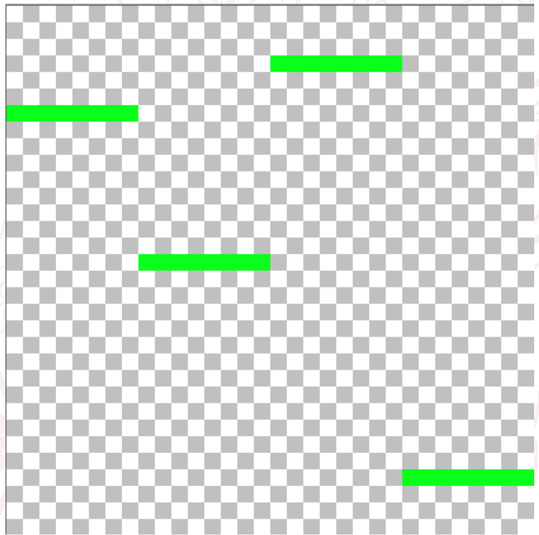
Posizione di 4 word da 32 bit



Posizione di 4 word da 16 bit



Posizione di 4 word da 8 bit



DRAM: Dynamic Random Access Memory

- Basso costo e alta capacità
- Fortemente utilizzate nelle memorie dei moderni computer
- Implementazione simile alla SRAM, ma con **importanti differenze**:
  - Elettronica più impegnativa
  - Volatile per design
  - Richiede periodicamente un refresh per mantenere il dato

Cella DRAM

- Dal punto di vista elettronico, una cella DRAM consiste in un transitor (T) e un condensatore (C)
- Se c'è abbastanza carica in C, il valore immagazzinato è 1, altrimenti è 0
- Il transistor agisce con interruttore:
  - Quando è aperto la carica in C rimane circa fissa (il dato è immagazzinato)
  - Quando è chiuso, la carica in C può scorrere tramite la Bit Line (B)
- Queste operazioni si riferiscono alla lettura/scrittura della cella

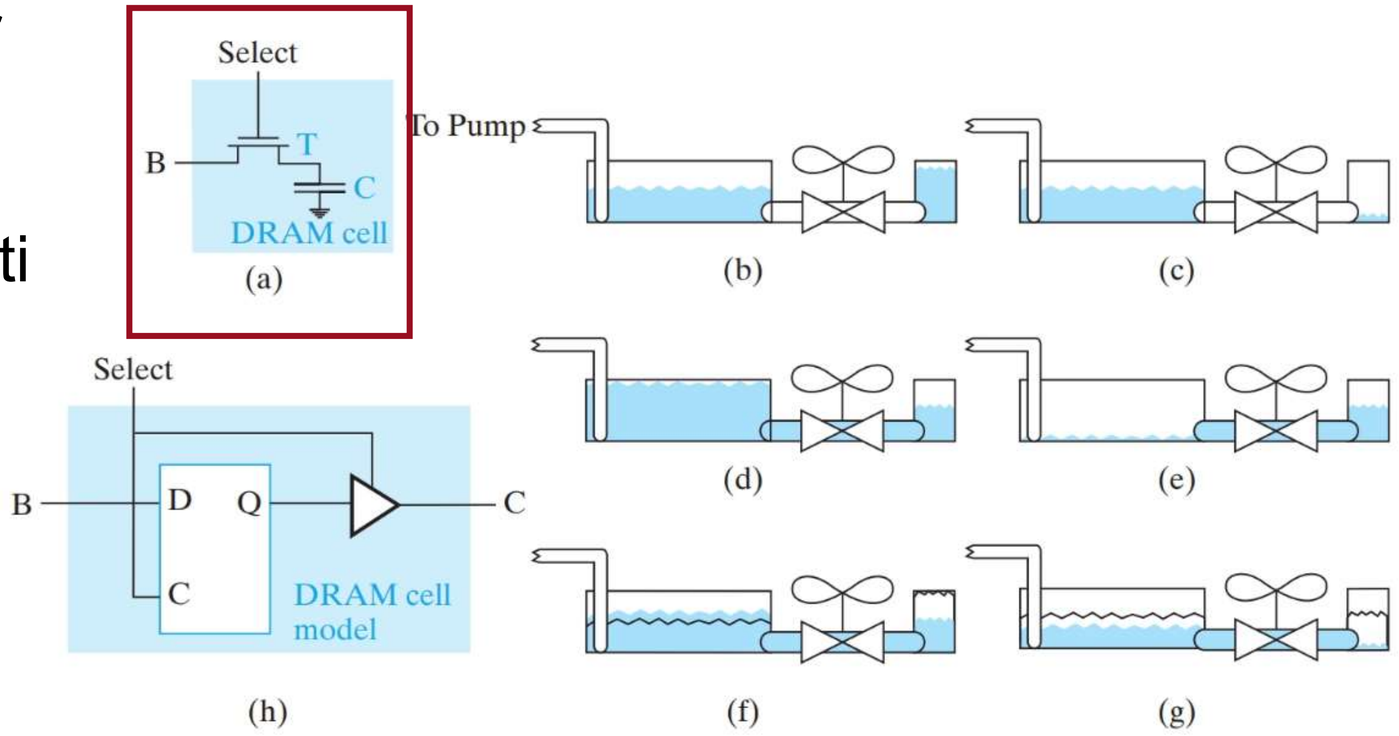


FIGURE 7-12
Dynamic RAM cell, hydraulic analogy of cell operation, and cell model

Cella DRAM: analogia idraulica

- Con valvola chiusa (T aperto):
  - C = 1
  - C = 0

storage

- Con valvola aperta (T chiuso):
  - B = 0 →1; C = 0 → 1
  - B = 1 → 0; C = 1 → 0

write

- Con valvola aperta (T chiuso):
  - C = 1; B = + dC = 1
  - C = 0; B = - dC = 0

read

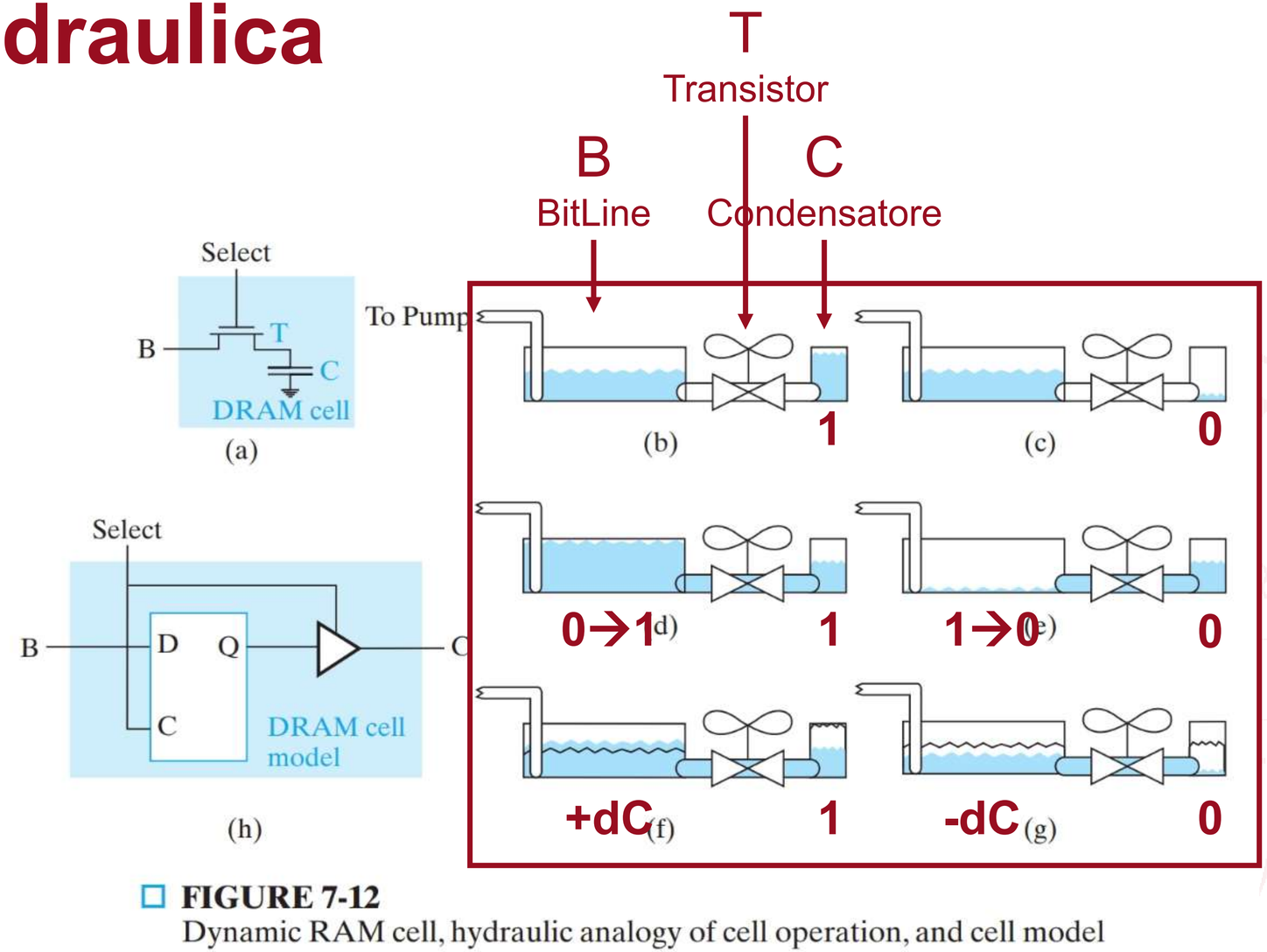
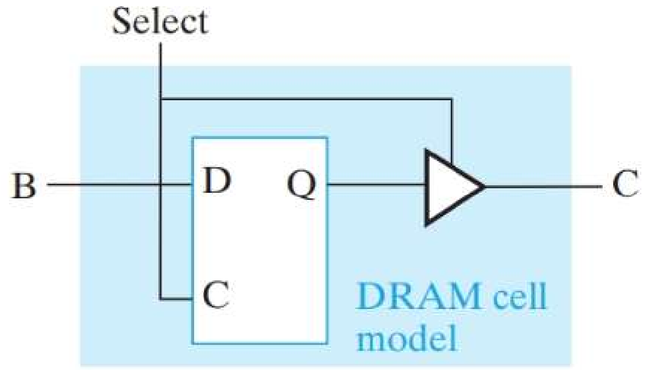


FIGURE 7-12
Dynamic RAM cell, hydraulic analogy of cell operation, and cell model

- Nella operazione di read, il contenuto in C viene perso (destructive read)
- Ogni operazione di read deve avere quindi un'operazione di restore
- Anche con la valvola chiusa (T aperto), ci sono perdite di carica → refresh

DRAM bit slice

- *Latch D come modello* per una cella DRAM



- In realtà le component circuitali sono diverse:
  - DRAM →Transistor + condensatore (2)
  - SRAM → 6 transistors (6)
- L'organizzazione logica di una DRAM bit slice è simile a quella di una SRAM, ma il numero di componenti è decisamente minore
- Data un'area di silicio, il numero di celle in una DRAM è più alto di quelle in una SRAM
- Capacità memorie DRAM maggiore

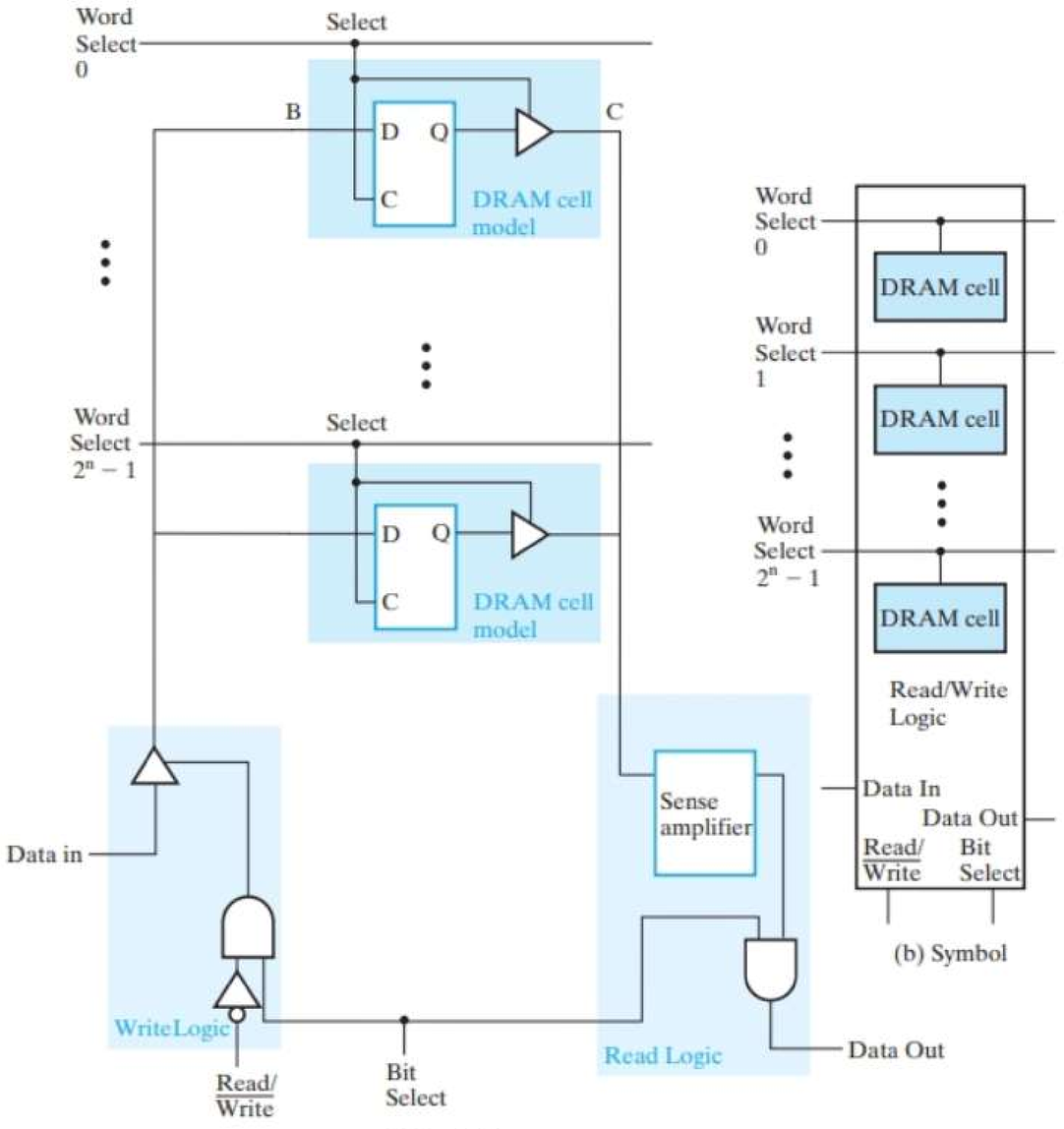


FIGURE 7-13
DRAM Bit-Slice Model

Indirizzamento nelle memorie DRAM

- Chip DRAM di grandi dimensioni richiedono 20 o più bit di indirizzo
- Per ridurre la dimensione fisica di un chip DRAM si usa un'organizzazione leggermente diversa
- L'indirizzo viene applicato in modo seriale alla cella:
  - Prima la parte relativa al **row address**
  - Poi la parte relativa al **column address**
- Gli indirizzi di riga e di colonna vengono salvati in registri
- $\overline{RAS}$ (Row Address Strobe) e $\overline{CAS}$ (Column Address Strobe) vengono utilizzati per abilitare l'accesso temporizzato ai registri

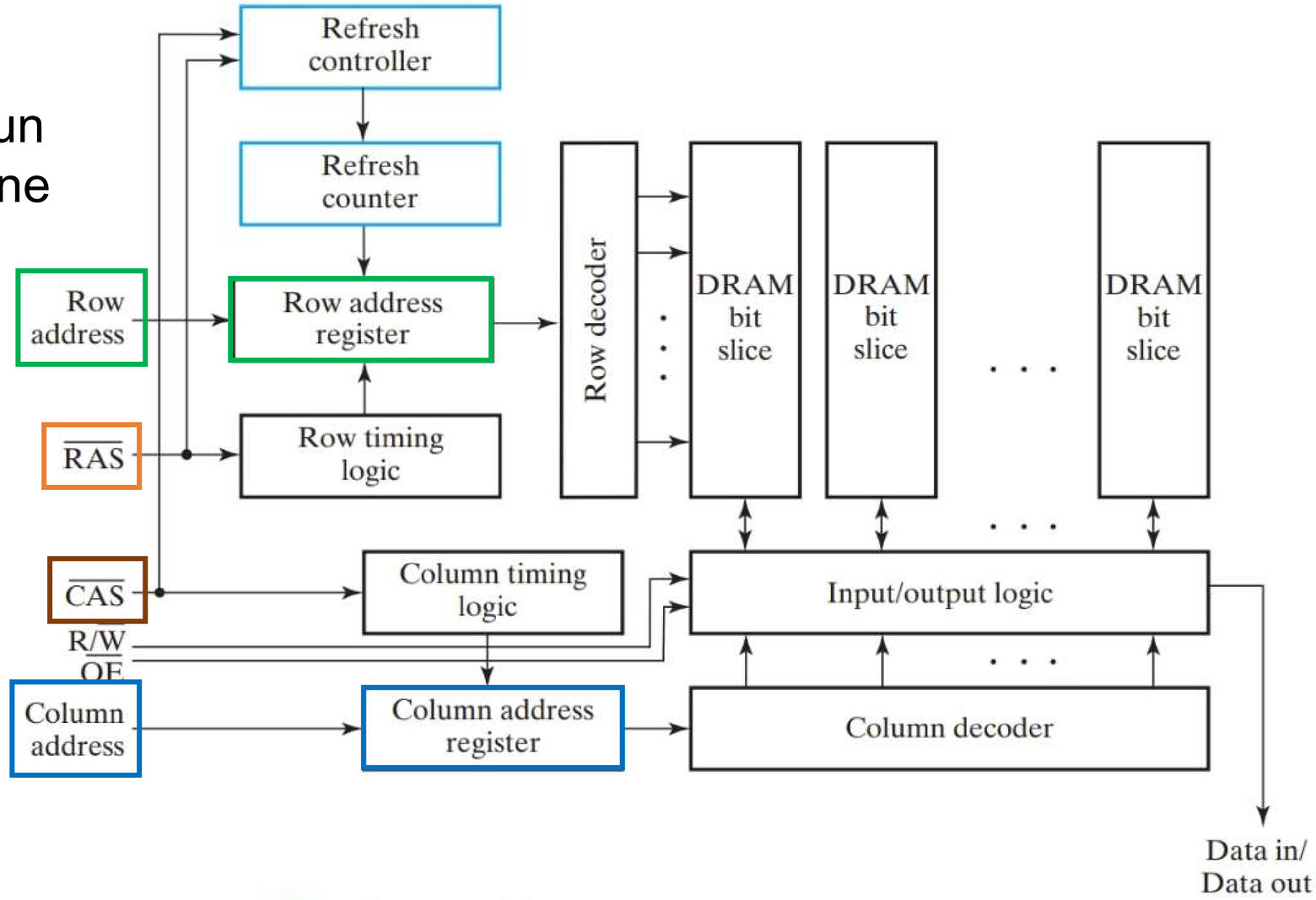
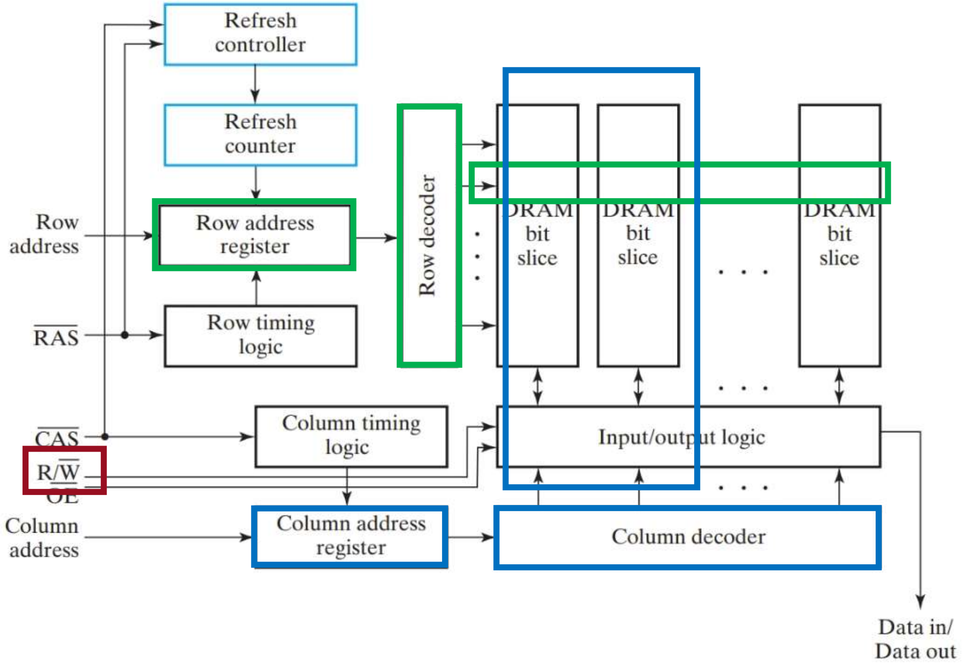
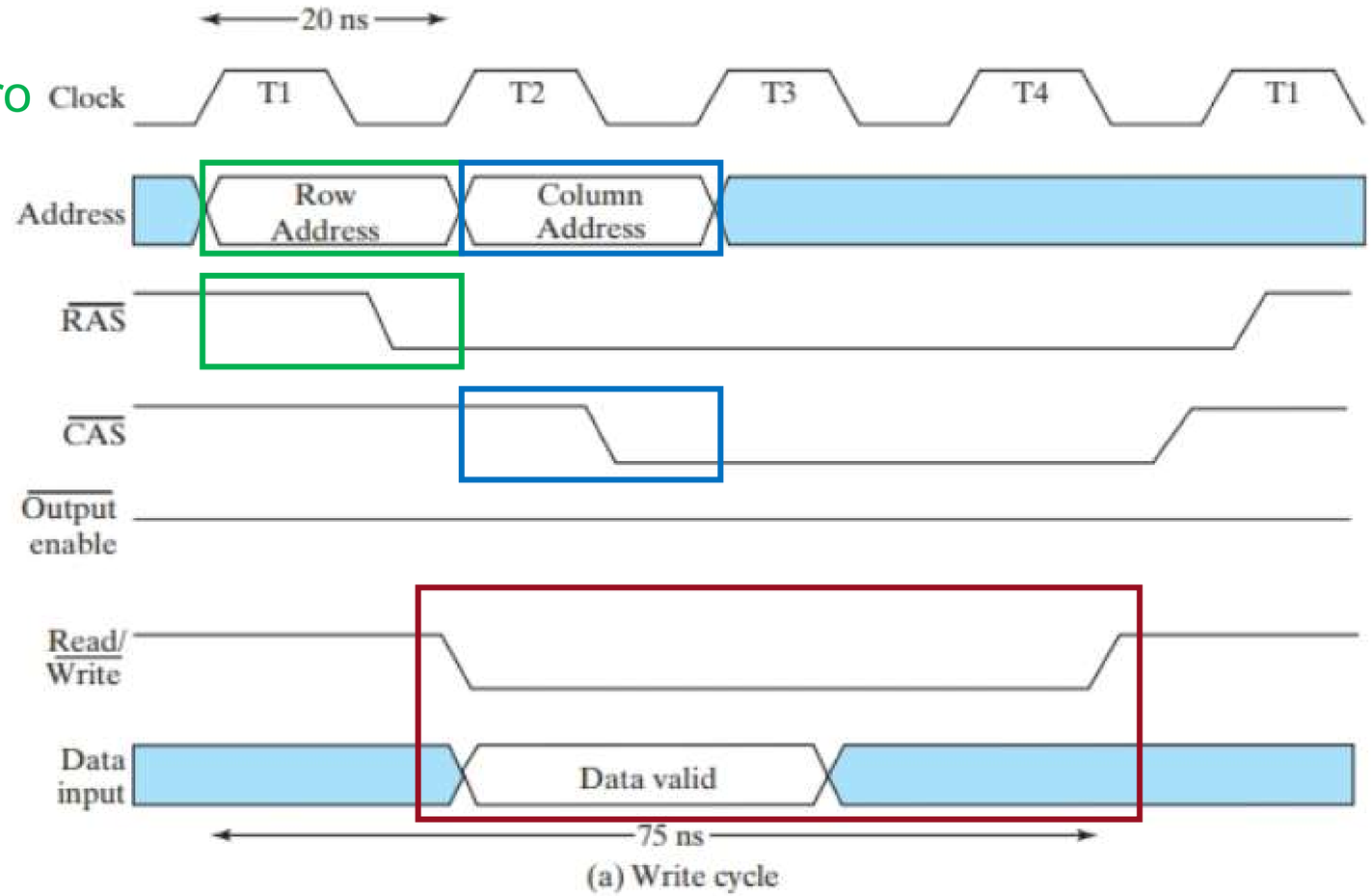


FIGURE 7-14
Block Diagram of a DRAM Including Refresh Logic

Temporizzazione nelle memorie DRAM: scrittura

1. Row adress applicato
2. RAS →0, row address caricato nel registro
3. Row address applicato al row decoder
4. Row selezionata
5. Column address
6. CAS →0, column address caricato
7. Column address al column decoder
8. Column selezionata

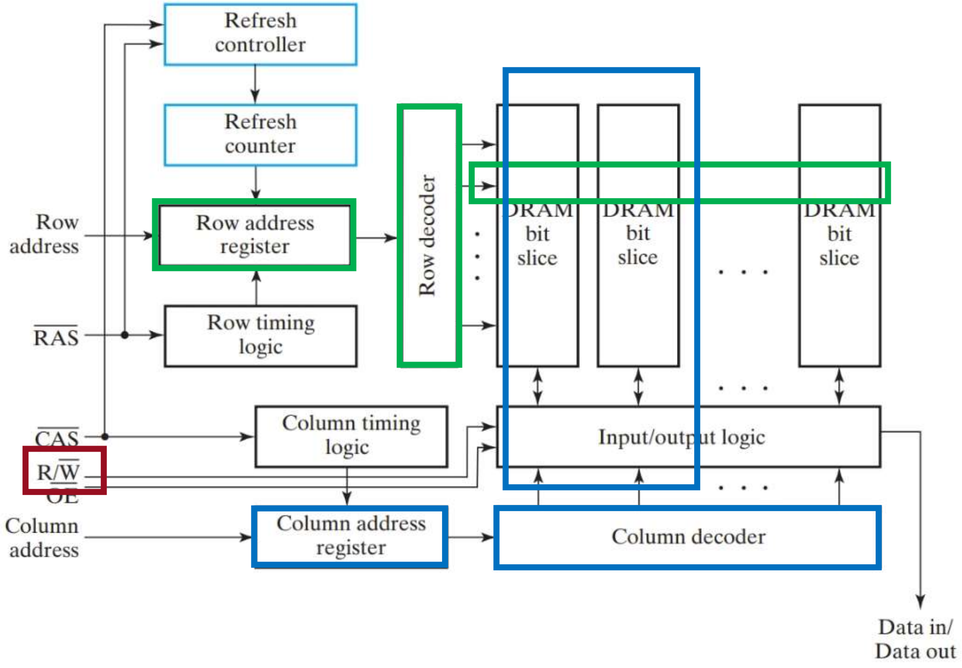


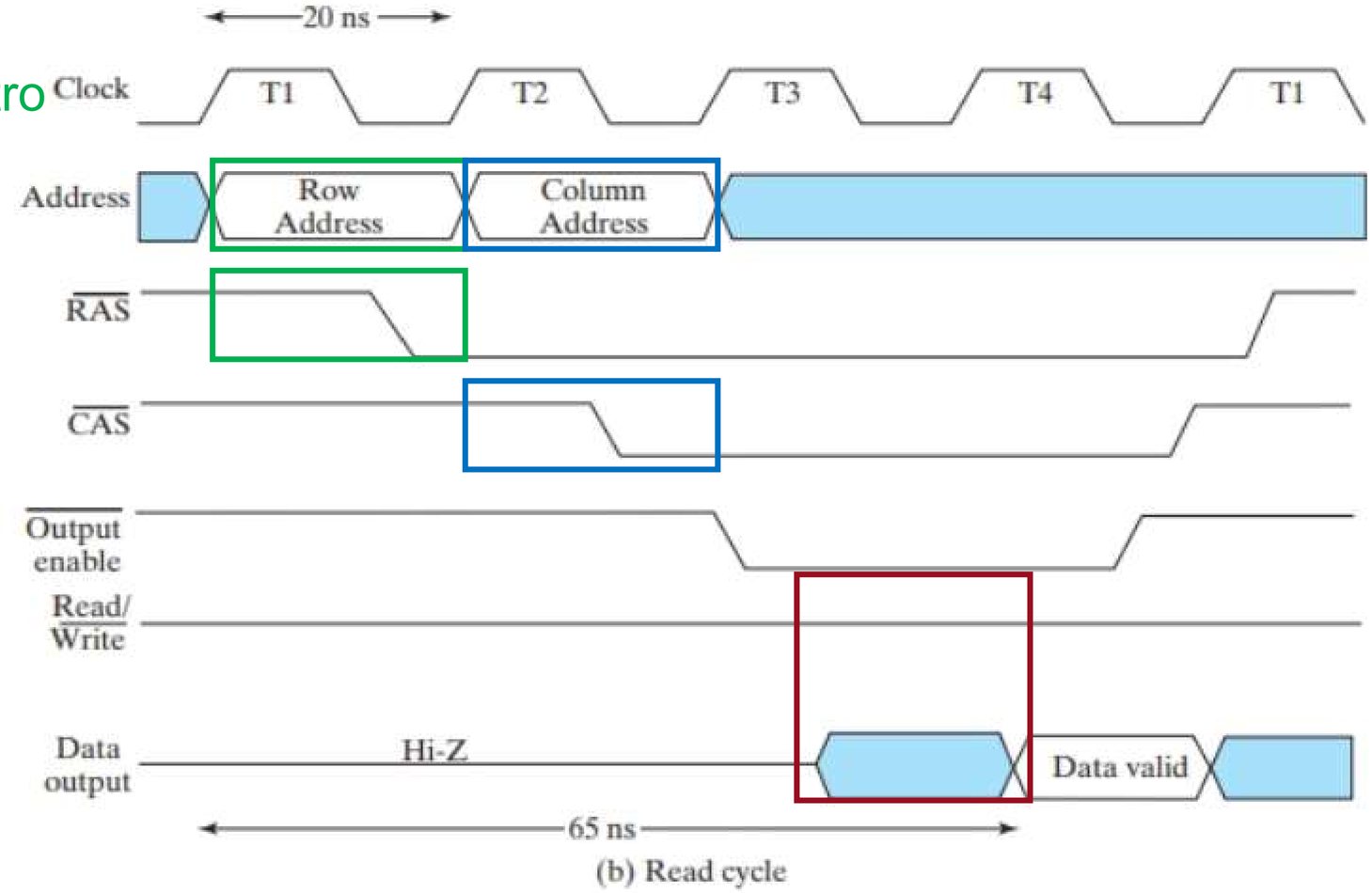


- RW → 0, scrittura di input nelle celle selezionate
- Dati nelle altre celle viene restored

Temporizzazione nelle memorie DRAM: lettura

1. Row adress applicato
2. RAS →0, row address caricato nel registro
3. Row address applicato al row decoder
4. Row selezionata
5. Column address
6. CAS →0, column address caricato
7. Column address al column decoder
8. Column selezionata





(b) Read cycle

- RW → 1, lettura dalle celle selezionate
- Dati nelle altre celle viene restored

Refresh nelle memorie DRAM

- **Refresh counter**: il numero della riga su cui fare il refresh
- **Refresh controller**: abilita il refresh
- Tre modi per triggerare un refresh:
  1. **RAS-only refresh**
     - Refresh di una specifica riga
     - Controllato esternamente
  2. **CAS-before-RAS refresh**
     - CAS →0 e RAS →0
     - Controllato dal refresh counter
  3. **Hidden refresh**
     - Applicato dopo ogni lettura/scrittura
     - Tempo maggiore per leggere/scrivere i dati

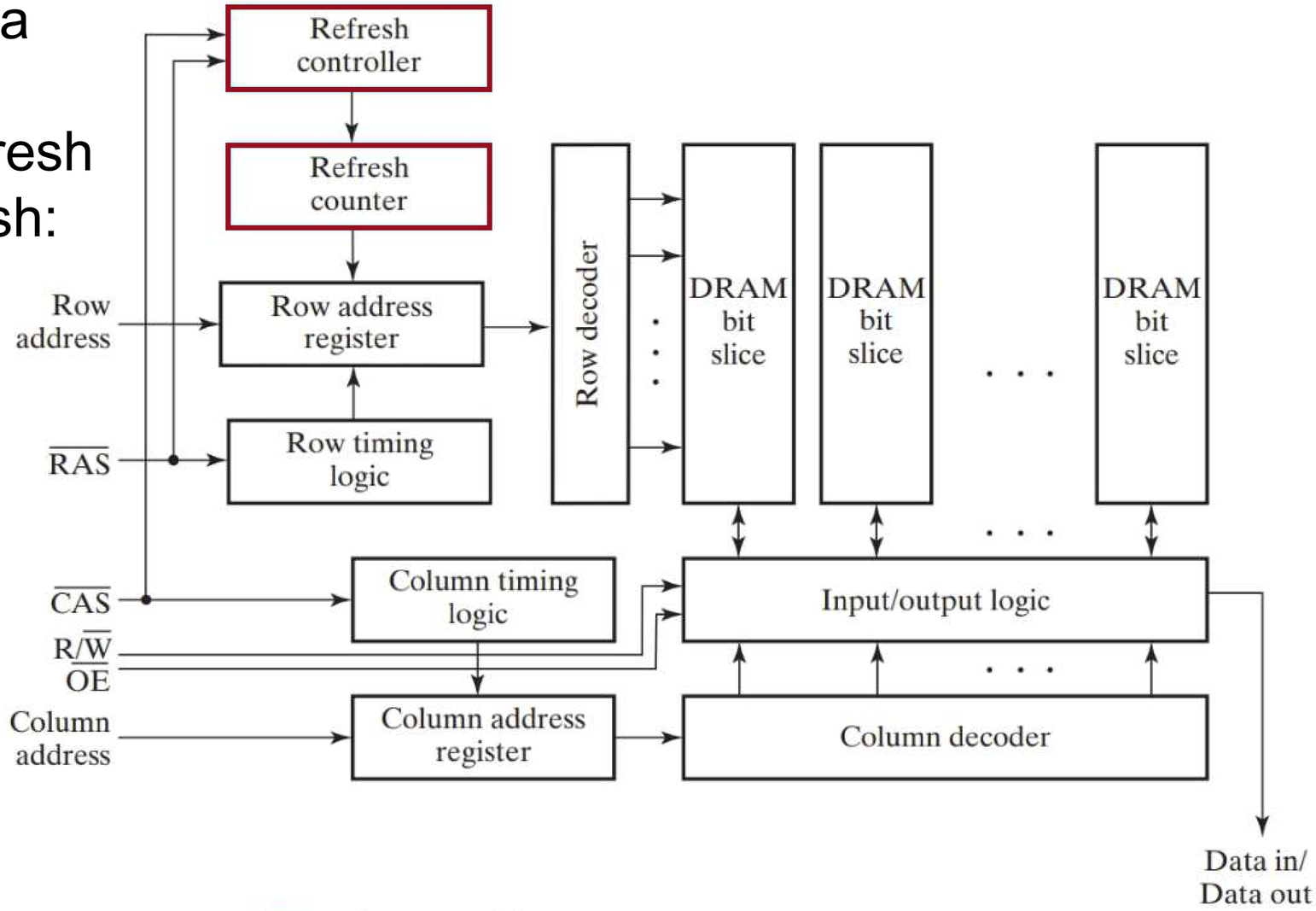
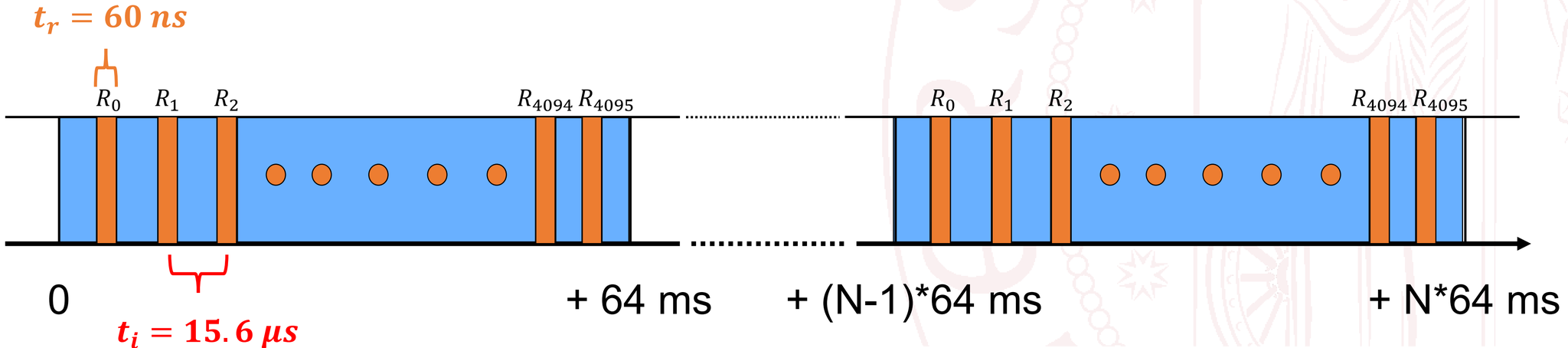


FIGURE 7-14
Block Diagram of a DRAM Including Refresh Logic

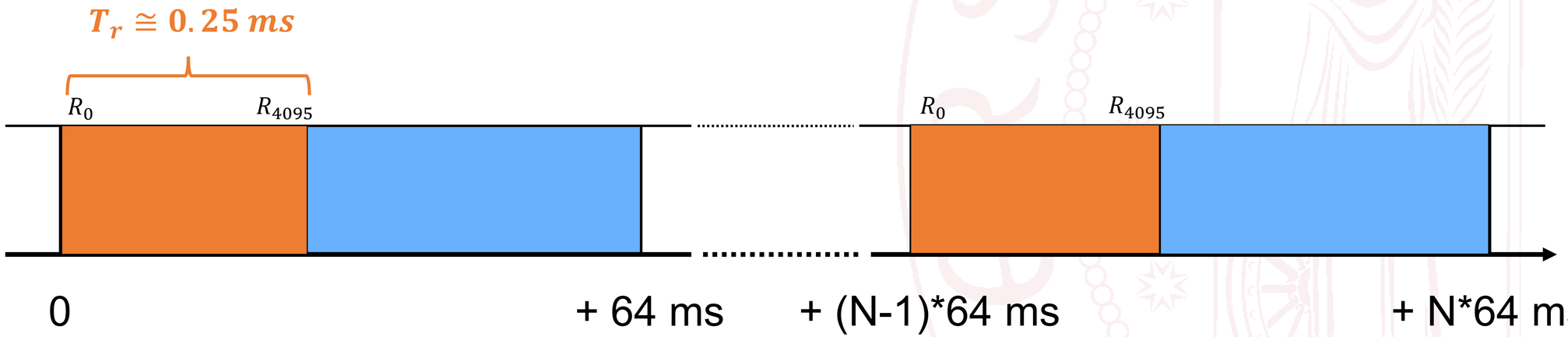
Refresh nelle memorie DRAM

- In un chip DRAM ogni riga deve essere aggiornata (refresh) in un tempo massimo altrimenti «perde» il dato (tipicamente fra 16 e 64 ms)
- **Refresh distribuito**: il refresh di ogni riga viene applicato in modo uniforme all'interno del suo tempo massimo di refresh (**refresh time**)
  - **Esempio:** 4M x 4 DRAM con refresh time di 64 ms
  - $N_r = 4 \cdot 1024 = 4096$ righe da aggiornare (refresh)
  - Un **singolo refresh** viene eseguito in $t_r = 60\ ns$
  - **L'intervallo fra i refresh** di ogni riga: $t_i = \frac{64\ ms}{4096} = 15.6\ \mu s$
  - Il tempo totale dei refreshes è $T_R = t_r \cdot N_R = 60\ ns\ \cdot 4096 = 0.00024576 \cong 0.25\ ms$
  - Questo lascia «libera» la memoria per il resto del tempo



Refresh nelle memorie DRAM

- In un chip DRAM ogni riga deve essere aggiornata (refresh) in un tempo massimo altrimenti «perde» il dato (tipicamente fra 16 e 64 ms)
- **Burst refresh**: i refreshes di tutte le righe vengono fatti uno dopo l'altro (**refresh time**)
  - **Esempio:** 4M x 4 DRAM con refresh time di 64 ms
  - $N_r = 4 \cdot 1024 = 4096$ righe da aggiornare (refresh)
  - Un **singolo refresh** viene eseguito in $t_r = 60\ ns$
  - I **4096 refresh** vengono eseguiti uno dopo l'altro
  - Il tempo totale dei refreshes è $T_R = t_r \cdot N_R = 60\ ns\ \cdot 4096 = 0.00024576 \cong 0.25\ ms$
  - Questo «occupa» la memoria per 0.25 ms di consecutivi



Synchronous DRAM (SDRAM)

Concetti preliminari:

- Nei moderni computer la CPU **NON** interagisce direttamente con la DRAM
- Esiste una gerarchia di memoria: CPU ←→ Cache (L1, L2) ←→Memoria
- Data una *read*, di solito vengono letti **bytes in indirizzi di memoria contigui (line)**
- Inoltre è spesso utile favorire la **lettura di tutti i bit di una riga**

16 MB SDRAM

- Simile organizzazione della DRAM
- Aggiunta del **clock** per operazioni sincrone
- Logica di controllo più complessa

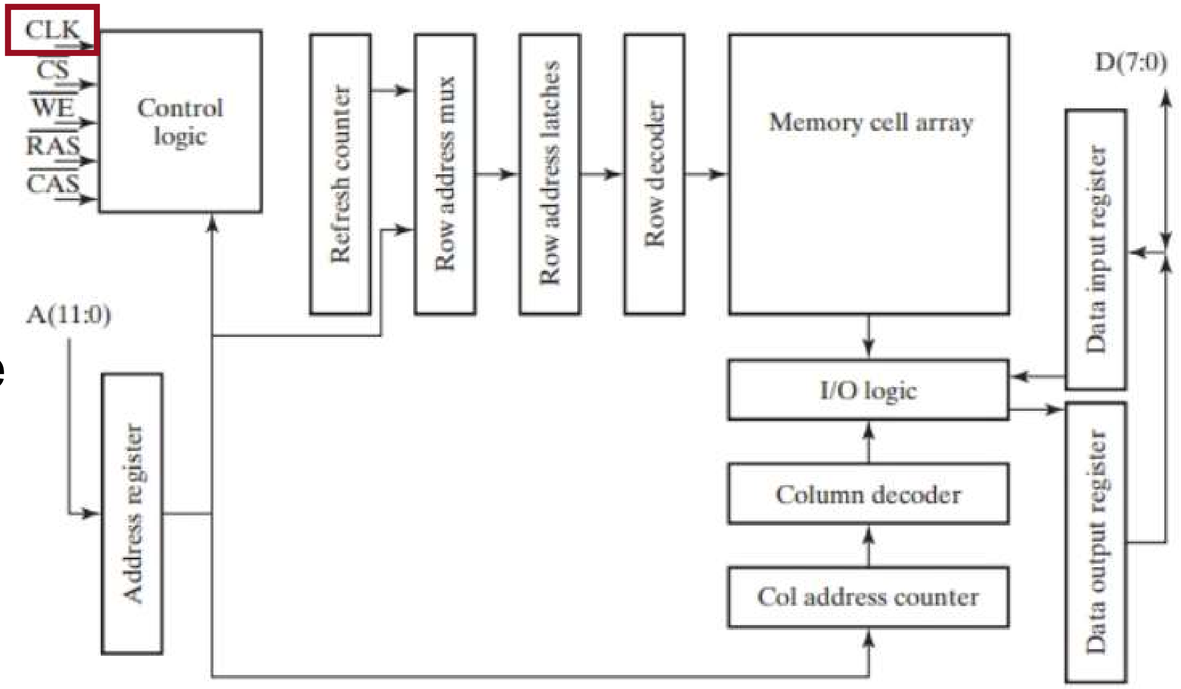


FIGURE 7-16
Block Diagram of a 16 MB SDRAM

Synchronous DRAM (SDRAM)

Esempio 16 MB SDRAM:

- $N_{bits} = 16 \cdot 2^{20} \cdot 8 = 134217728\ bits$
- $N_{rows} = 8192,\ N_{cols} = 16384 \Rightarrow N_{rows} \cdot N_{cols} = N_{bits}$
- $N_{ARbit} = \log_2 8192 = 13\ bits$
- $N_{ACbit} = \log_2 \frac{16384}{8\ bit} = 11\ bits$
- $N_{ARbit} + N_{ACbit} = 24 = \log_2(16 \cdot 2^{20})$

- **Row address** applicato
- Intera riga caricata nella **I/O logic**
- **Column address** applicato
- Data nel **registro di output**, 1 byte per clock

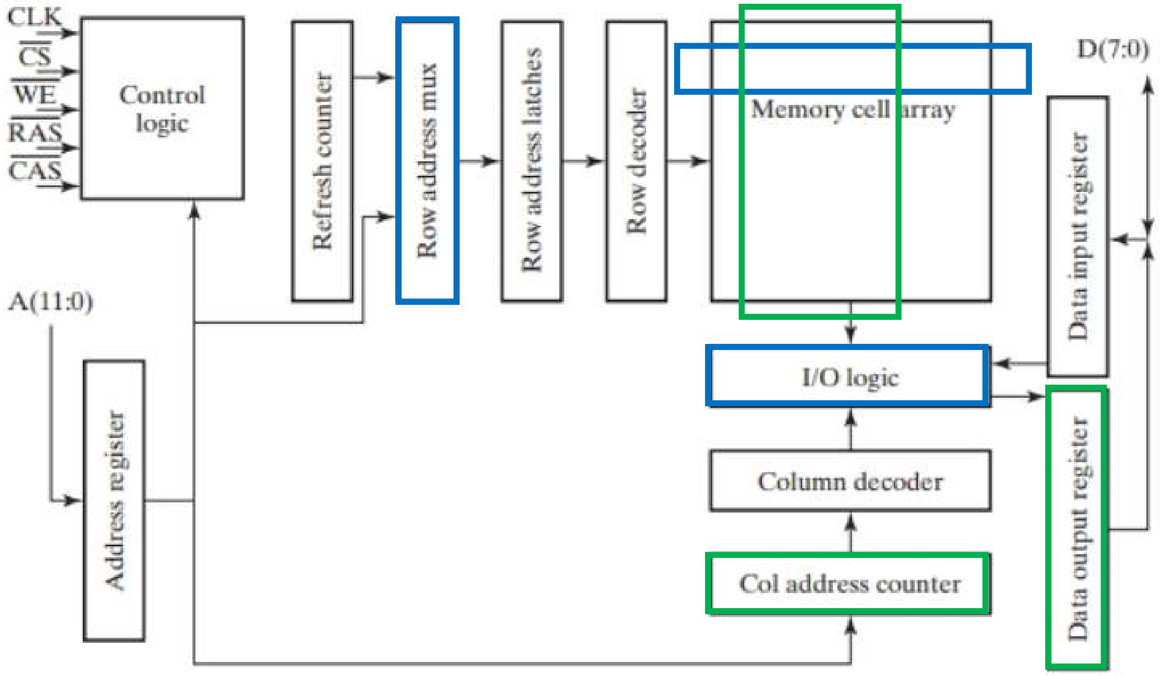


FIGURE 7-16
Block Diagram of a 16 MB SDRAM

Synchronous DRAM (SDRAM)

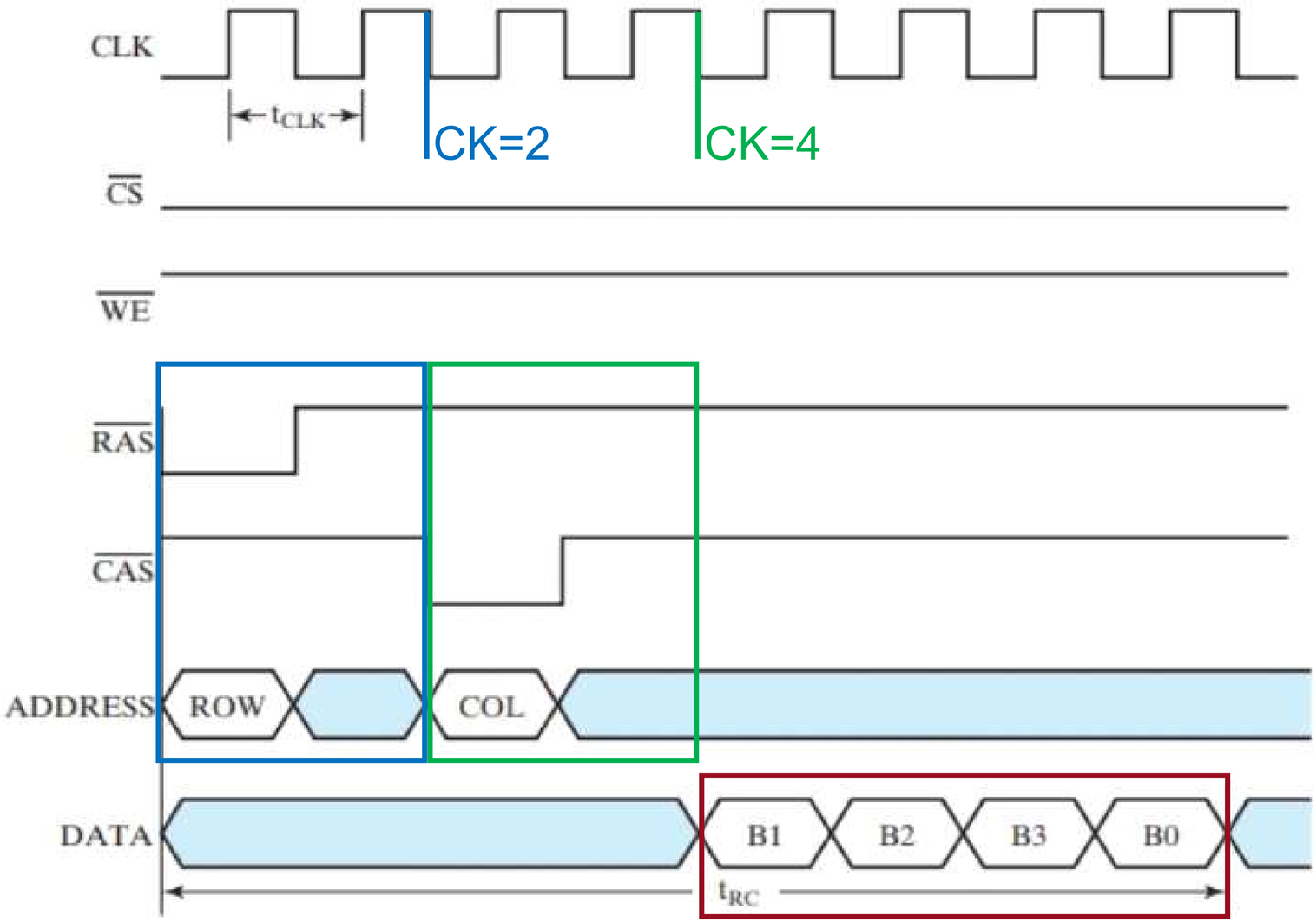


FIGURE 7-17
Timing Diagram for an SDRAM

- **Row address** nel registro
- **Lettura della riga**
- **Column address** nel registro
- **Lettura dei bytes**
- **Uscita di 4 bytes**, uno ad ogni ciclo di clock

SDRAM vs. DRAM

DRAM

- Ciclo di lettura: $t_{RC} = 60\ ns$
- 1 byte per ciclo di lettura
- $BitRate = \frac{1}{t_{rc}} = 16.67\ MB/s$

SDRAM

Se si leggono 4 bytes:

- Clock: $t_{CLK} = 7.5\ ns$
- $t_{RC} = 7.5\ ns \cdot (4\ cicli\ fissi + 4\ cicli) = 60\ ns$
- $BitRate = \frac{1}{t_{rc}} = 66.67\ MB/s$

Se si leggono 8 bytes invece di 4:

- $t_{RC} = 7.5\ ns \cdot (4\ cicli\ fissi + 8\ cicli) = 90\ ns$
- $BitRate = \frac{1}{t_{RC}} = 88.89\ MB/s$

Se si legge tutta una riga (2048 bytes):

- $t_{RC} = 7.5\ ns \cdot (4\ cicli\ fissi + 2048\ cicli) = 15390\ ns$
- $BitRate = \frac{1}{t_{RC}} = 133.07\ MB/s$

Approssima un byte per ciclo di clock ←