ARCHITETTURA DEGLI ELABORATORI

A.A. 2020-2021

Università di Napoli Federico II
Corso di Laurea in Informatica

Docenti
Proff. Luigi Sauro gruppo 1 (A-G)
Silvia Rossi gruppo 2 (H-Z)

Shifters

Logical shifter: trasla i bit a sinistra o a destra e riempie gli spazi vuoti con degli 0

Arithmetic shifter: come il logical shifter a sinistra, nella traslazione a destra invece riempie gli spazi vuoti con il bit più significativo (msb)

Rotator: ruota i bit in cerchio, i bit che “escono” da un lato rientrano dall’ ”altro”

Shifters

Logical shifter:

- Ex: 11001 >> 2 = 00110
- Ex: 11001 << 2 = 00100

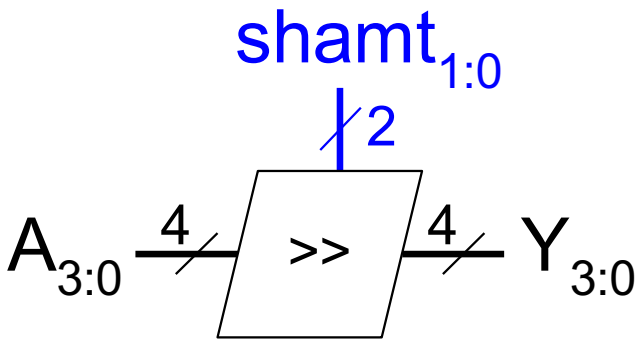
Arithmetic shifter:

- Ex: 11001 >>> 2 = 11110
- Ex: 11001 <<< 2 = 00100

Rotator:

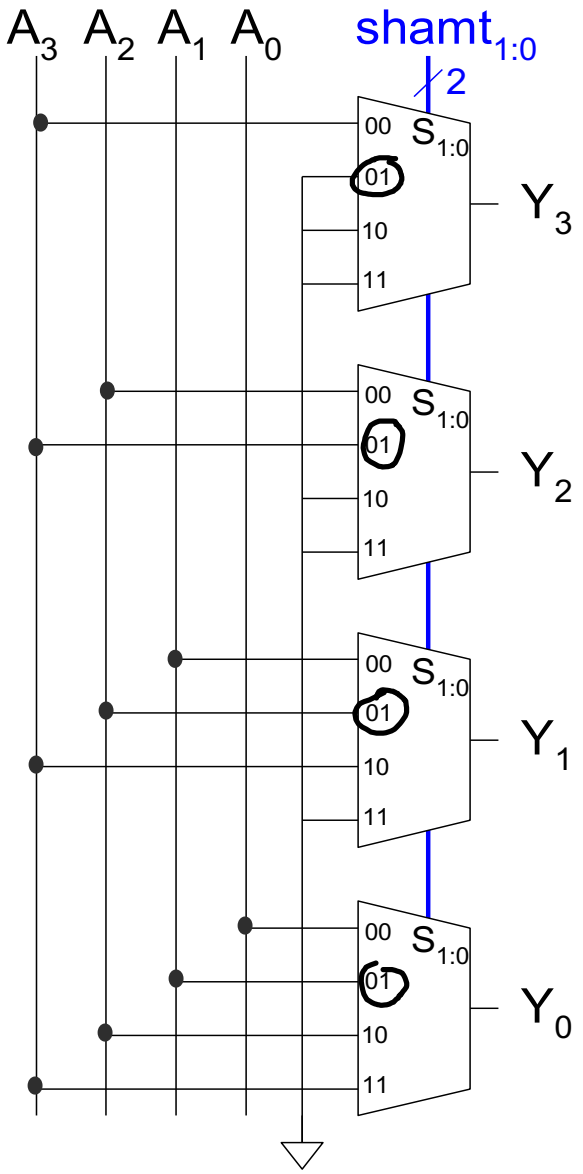
- Ex: 11001 ROR 2 = 01110
- Ex: 11001 ROL 2 = 00111

Shifter Design

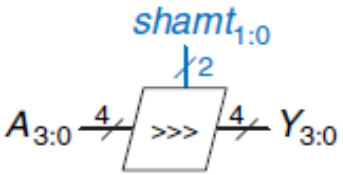


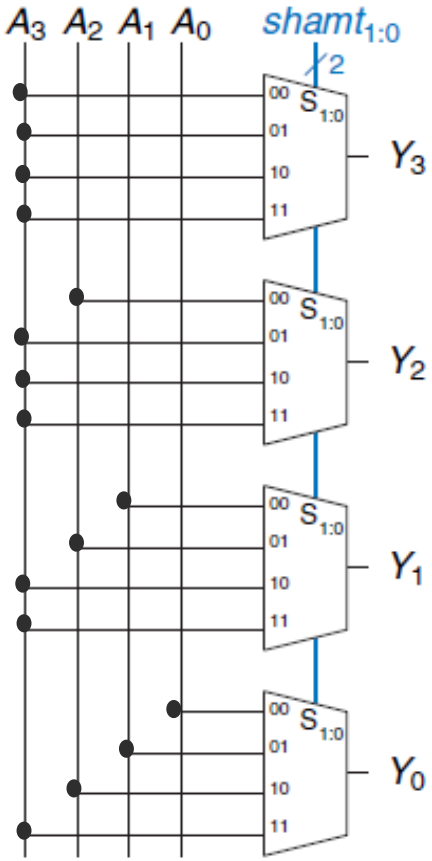
$A_3 A_2 A_1 A_0$

$0\ A_3 A_2 A_1$



Shifter Design





Shifters as Multipliers, Dividers

- $A <<< N = A \times 2^N$
  - **Example:** 00001 << 2 = 00100 ($1 \times 2^2 = 4$)
  - **Example:** 11101 << 2 = 10100 ($-3 \times 2^2 = -12$)
- $A >>> N = A \div 2^N$
  - **Example:** 01000 >>> 2 = 00010 ($8 \div 2^2 = 2$)
  - **Example:** 10000 >>> 2 = 11100 ($-16 \div 2^2 = -4$)

Counters

- Incrementa ad ogni clock
- Usato per eseguire cicli su numeri:

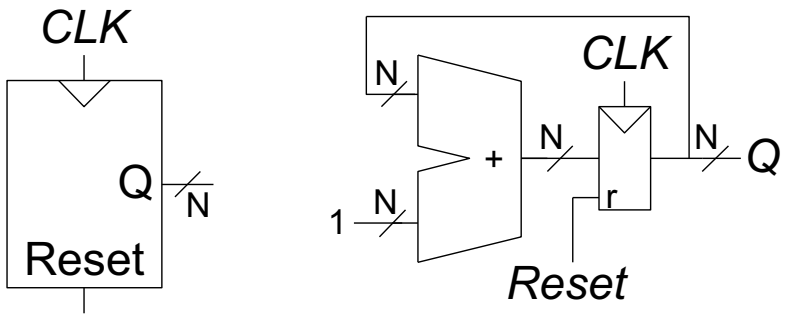
  000, 001, 010, 011, 100, 101, 110, 111, 000, 001…
- Si usa:
  - Come orologio digitale
  - Program counter: tiene traccia della istruzione corrente da eseguire

Counters

- Incrementa ad ogni clock
- Usato per eseguire cicli su numeri:

  000, 001, 010, 011, 100, 101, 110, 111, 000, 001…
- Si usa:
  - Come orologio digitale
  - Program counter: tiene traccia della istruzione corrente da eseguire

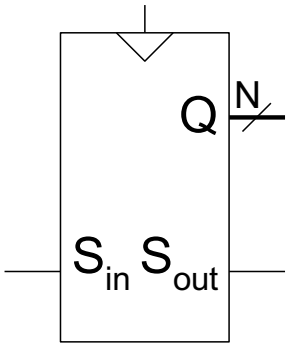
Symbol **Implementation**



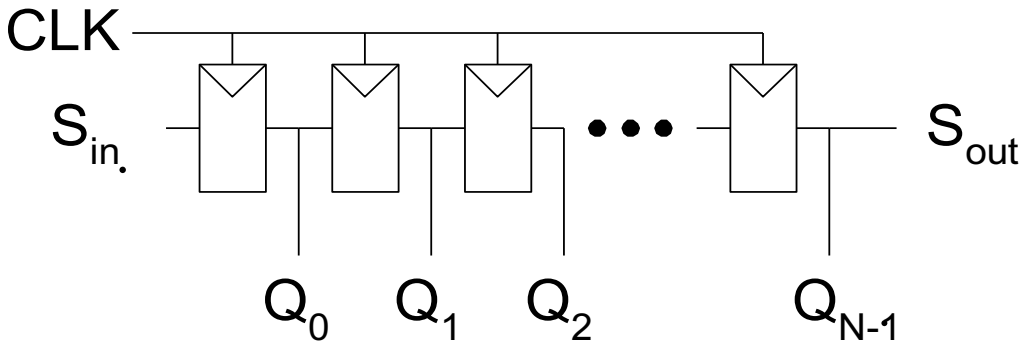
Shift Registers

- Trasla di un bit ad ogni clock
- Ritorna un bit in uscita (S_{out}) ad ogni clock
- *Serial-to-parallel converter*: converte un input seriale (S_{in}) in un output parallelo ($Q_{0:N-1}$)

Symbol:

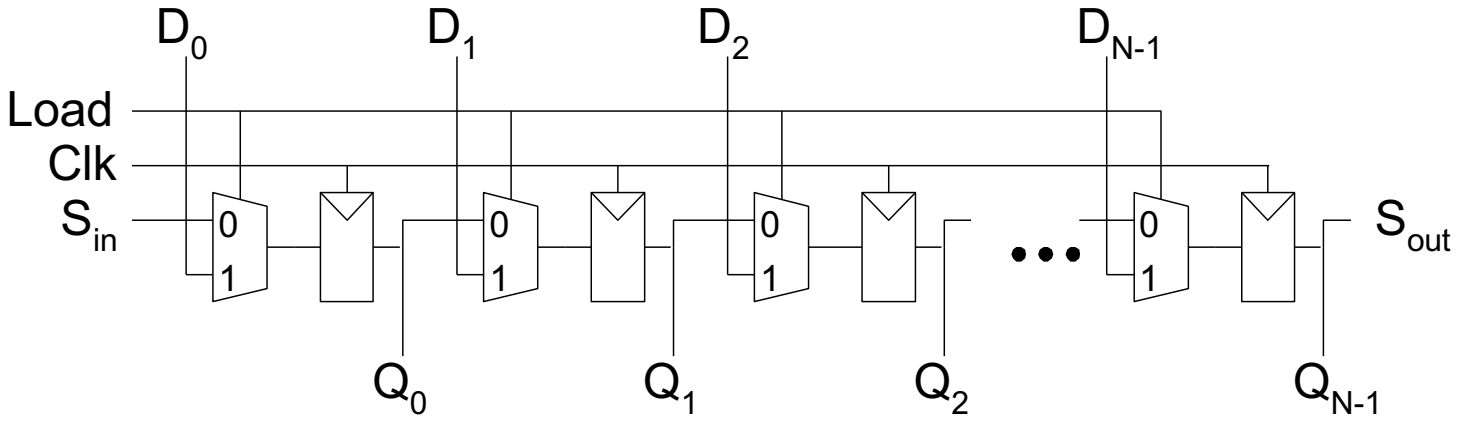


Implementation:



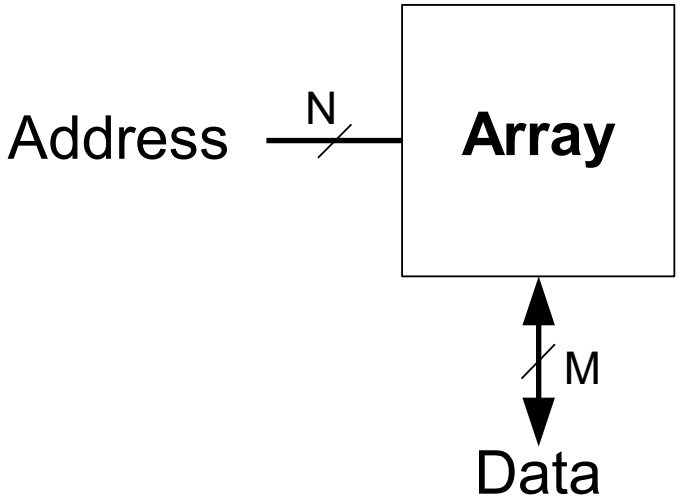
Shift Register con load parallelo

- Quando *Load* = 1, funziona come un usuale registro a *N*-bit
- Quando *Load* = 0, agisce da shift register
- Quindi può agire da convertitore *seriale-parallelo* (da S_{in} a $Q_{0:N-1}$) o da convertitore *parallelo-seriale* ($D_{0:N-1}$ to S_{out})



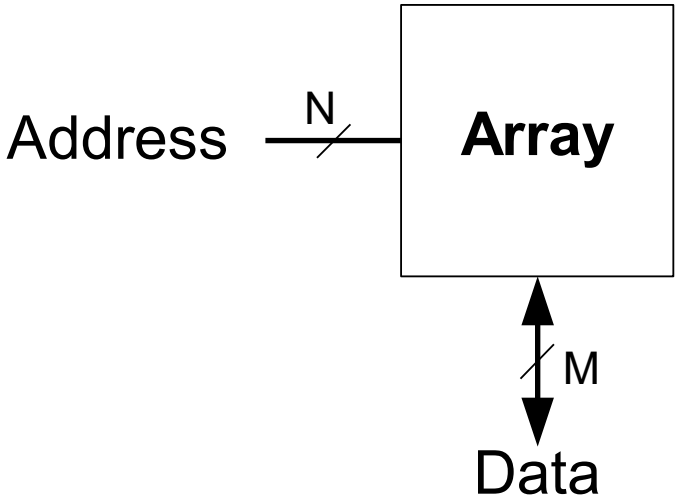
Memory Arrays

- Memorizzare efficacemente una grossa quantità di dati
- 3 tipologie:
  - Dynamic random access memory (DRAM)
  - Static random access memory (SRAM)
  - Read only memory (ROM)
- dato a *M*-bit letto/scritto su di un unico indirizzo a N-bit



Memory Arrays

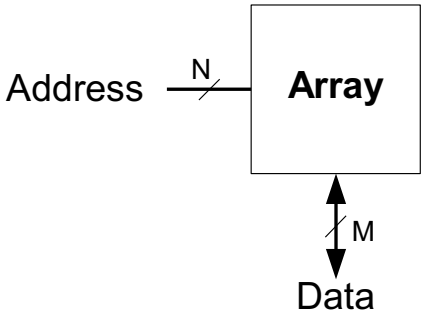
- Memorizzare efficacemente una grossa quantità di dati
- 3 tipologie:
  - Dynamic random access memory (DRAM)
  - Static random access memory (SRAM)
  - Read only memory (ROM)
- dato a *M*-bit letto/scritto su di un unico indirizzo a N-bit

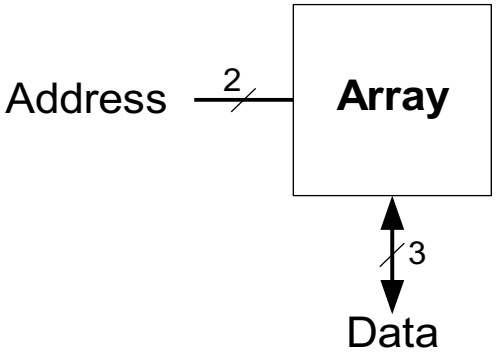


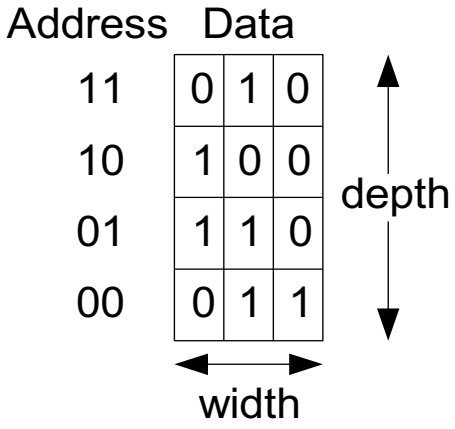
Tipicamente
M=8 (un byte) e
N=32 (o 64)

Memory Arrays

- Array bidimensionali di celle di memoria
- Ogni cella memorizza un bit
- *Con N* bit di indirizzo e *M* bits data:
  - 2^N righe e a *M* colonne
  - **Depth:** numero di righe (parole)
  - **Width:** numero di colonne (lunghezza di una parola)
  - **Dimensione Array :** depth × width = $2^N \times M$

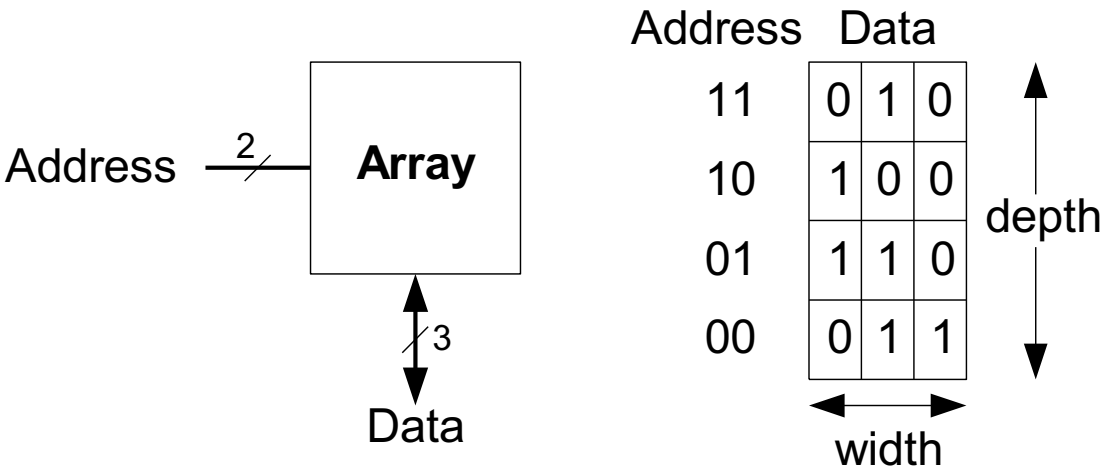




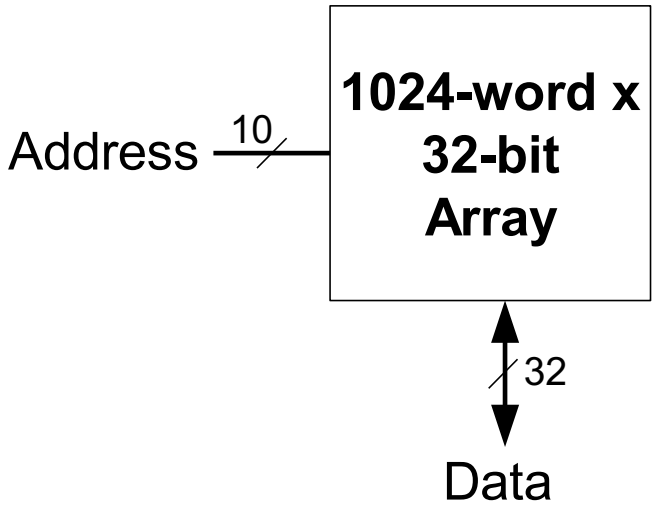


Esempio

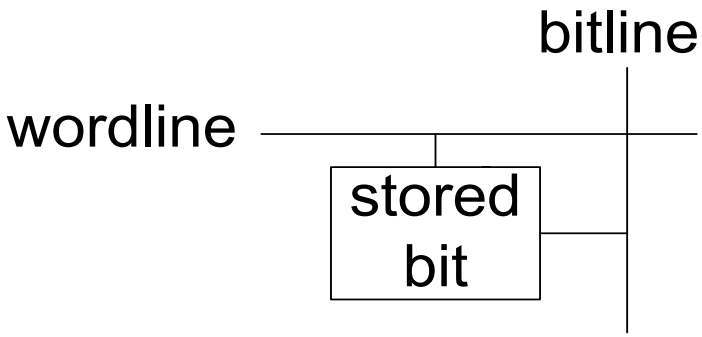
- $2^2 \times$ 3-bit array
- Numero parole: 4
- Lunghezza parole: 3-bits
- All'indirizzo 10 corrisponde la parola 100

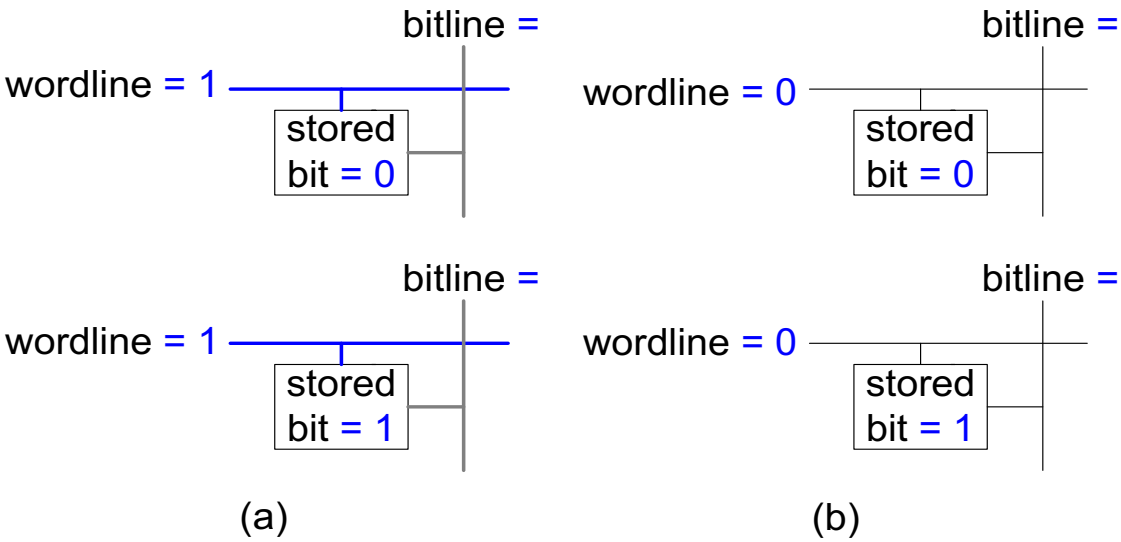


Memory Arrays

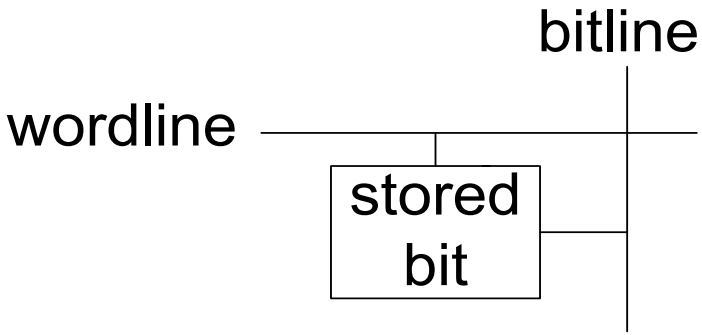


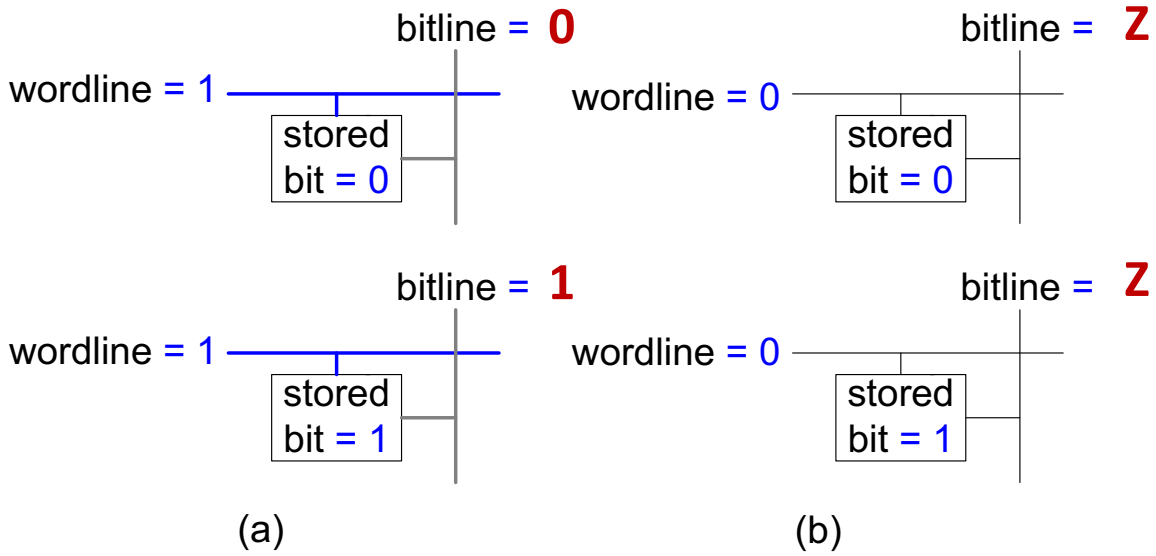
Memory Array Bit Cells





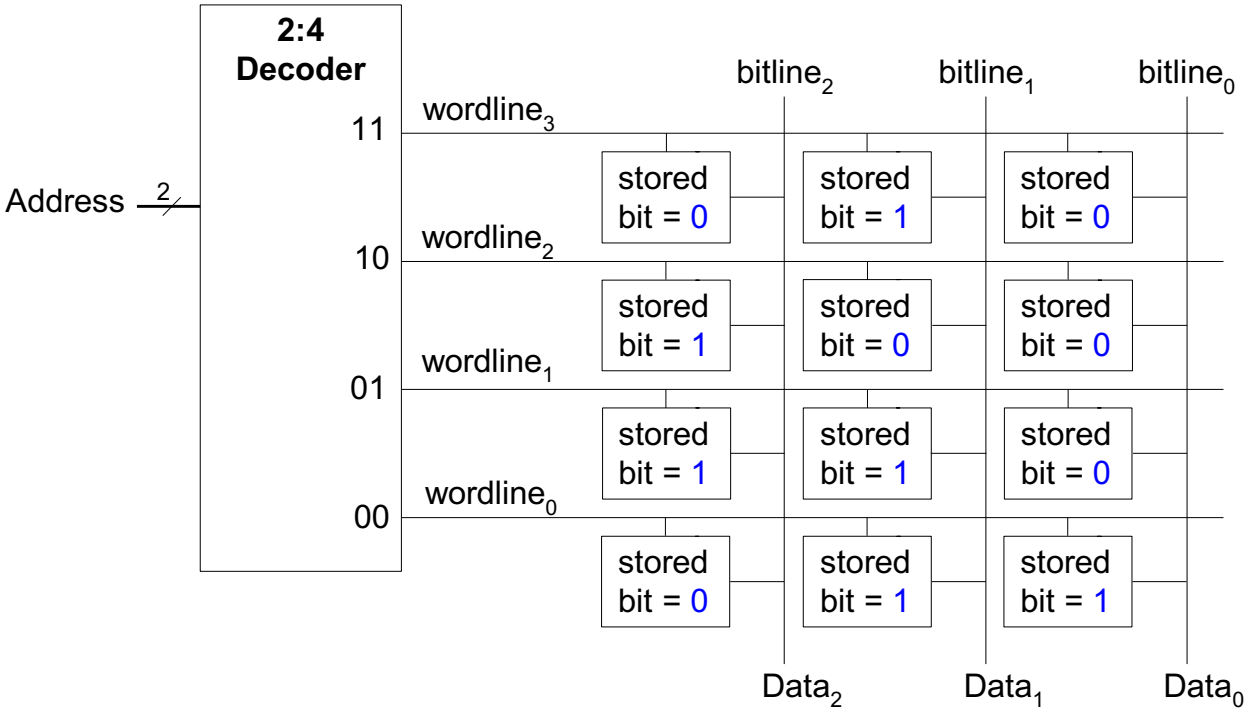
Memory Array Bit Cells

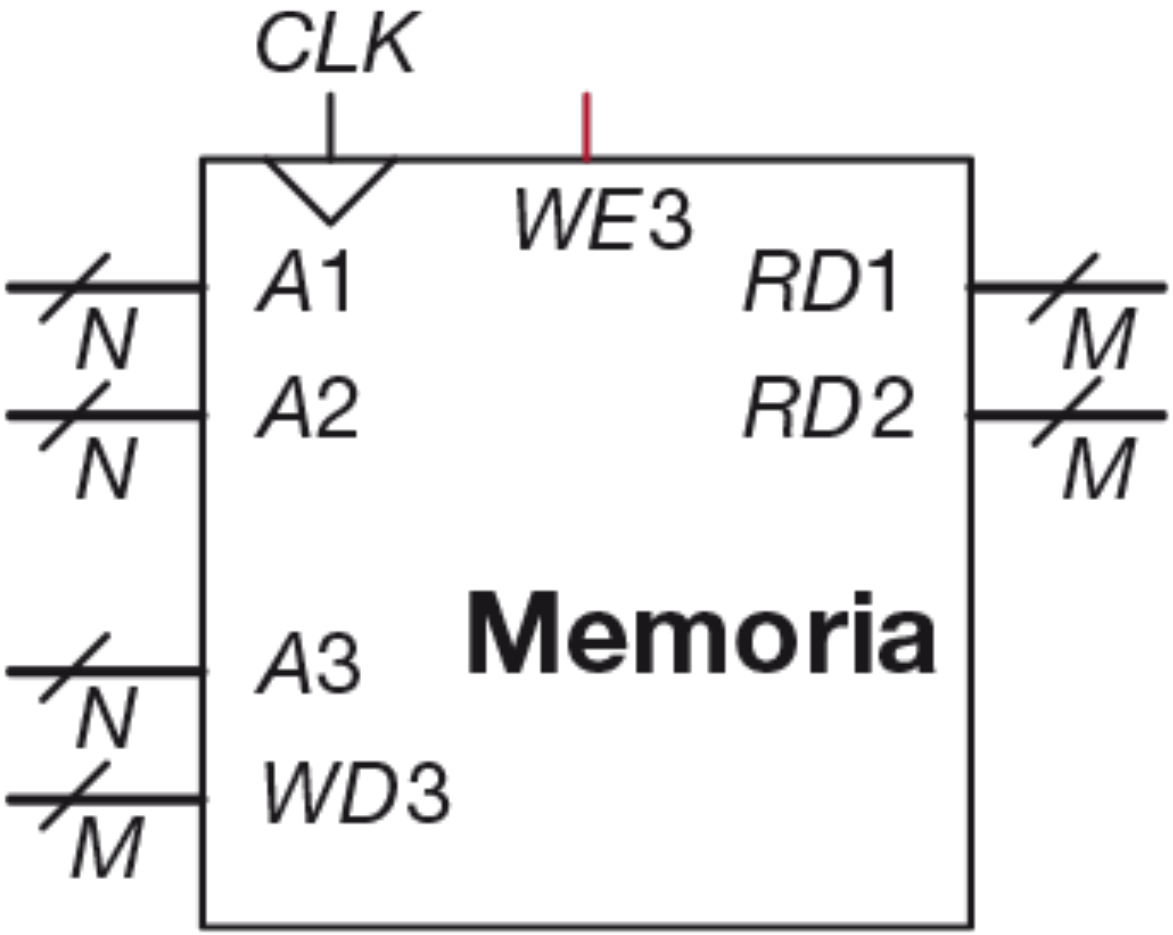




Memory Array

- **Wordline:**
  - Agisce come un enable
  - Seleziona una riga nella memoria
  - Corrisponde ad un unico indirizzo
  - Solo una wordline per volta è attivata




CLK
WE3
A1
A2
A3
WD3
RD1
RD2
Memoria
N
N
N
M
M
M

Tipi di memoria

- Random access memory (RAM): **volatile**
- Read only memory (ROM): **nonvolatile**

RAM: Random

- **Volatile:** i dati sono persi quando il computer è spento
- Le operazioni di lettura e scrittura sono più veloci (rispetto alle ROM)
- E' la memoria primaria di un computer (DRAM)

E' chiamata *random* access memory perché il tempo di accesso ad una parola è lo stesso per ogni indirizzo (a differenza delle memorie ad accesso sequenziale come i nastri che si usavano ai tempi del C64)

ROM: Read Only Memory

- **Non volatile:** mantiene i dati anche quando il computer è spento
- La lettura è relativamente veloce ma la scrittura è lenta o semplicemente non è possibile scrivere
- Drives, flash memory, Bios etc. sono ROMs

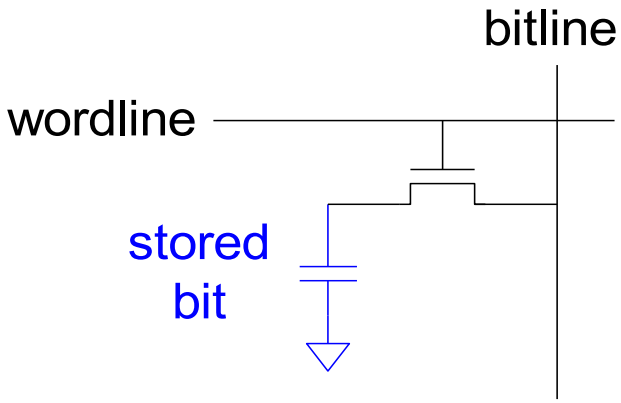
ROM sta per *read only* memory perché inizialmente questo tipo di memorie venivano "scritte" bruciando dei fusibili. Quindi, una volta configurate, non erano più manipolabili. Chiaramente questo non è più il caso per Flash memory e altri tipi di ROMs.

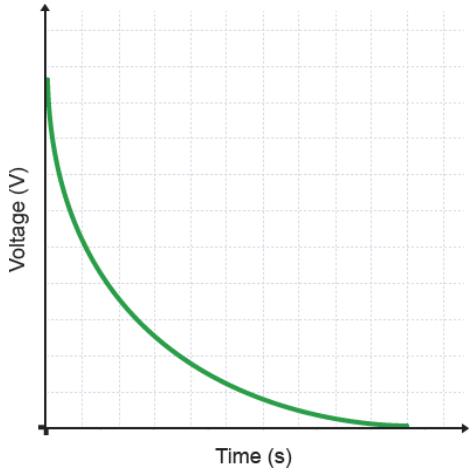
Tipi di RAM

- **DRAM** (Dynamic random access memory)
- **SRAM** (Static random access memory)
- Si differenziano per le componenti usate per memorizzare i dati:
  - DRAM usa delle capacità
  - SRAM usa invertitori

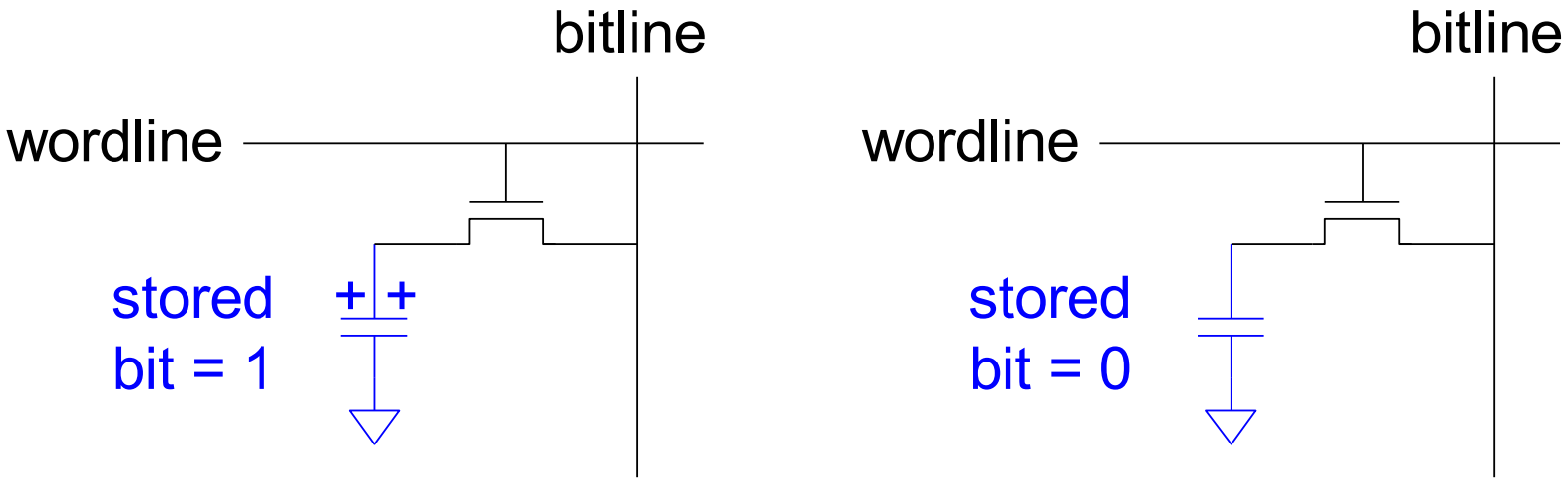
DRAM

- i bit data sono immagazzinati in delle capacità
- *Dynamic* perché il valore di un bit deve essere "refreshed" (riscritto) periodicamente e anche dopo che è stato letto:
  - La perdita di carica di una capacità degrada il valore memorizzato
  - La lettura distrugge il valore letto

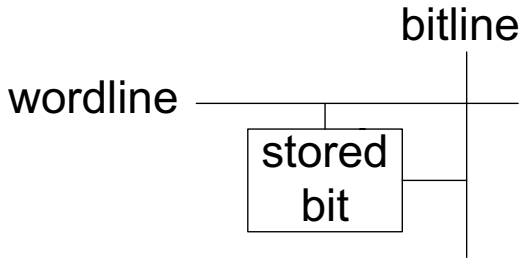


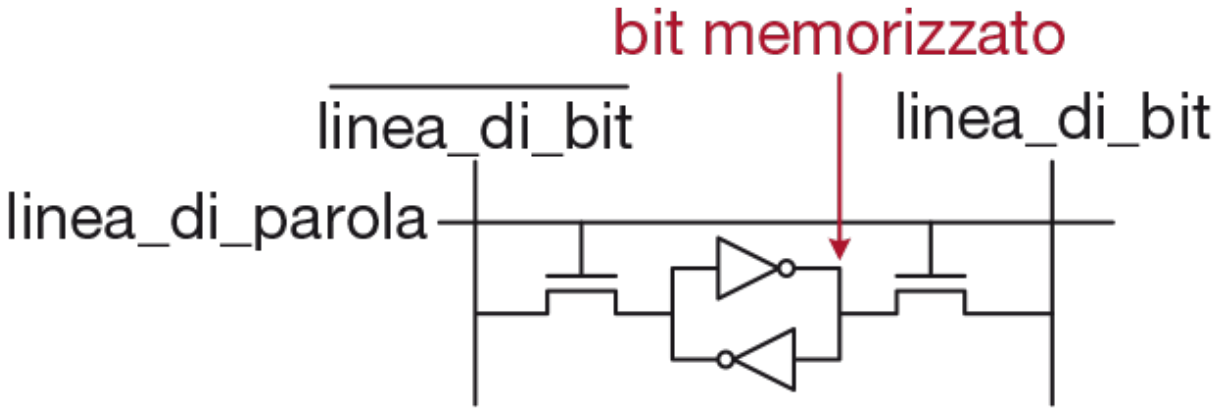


DRAM

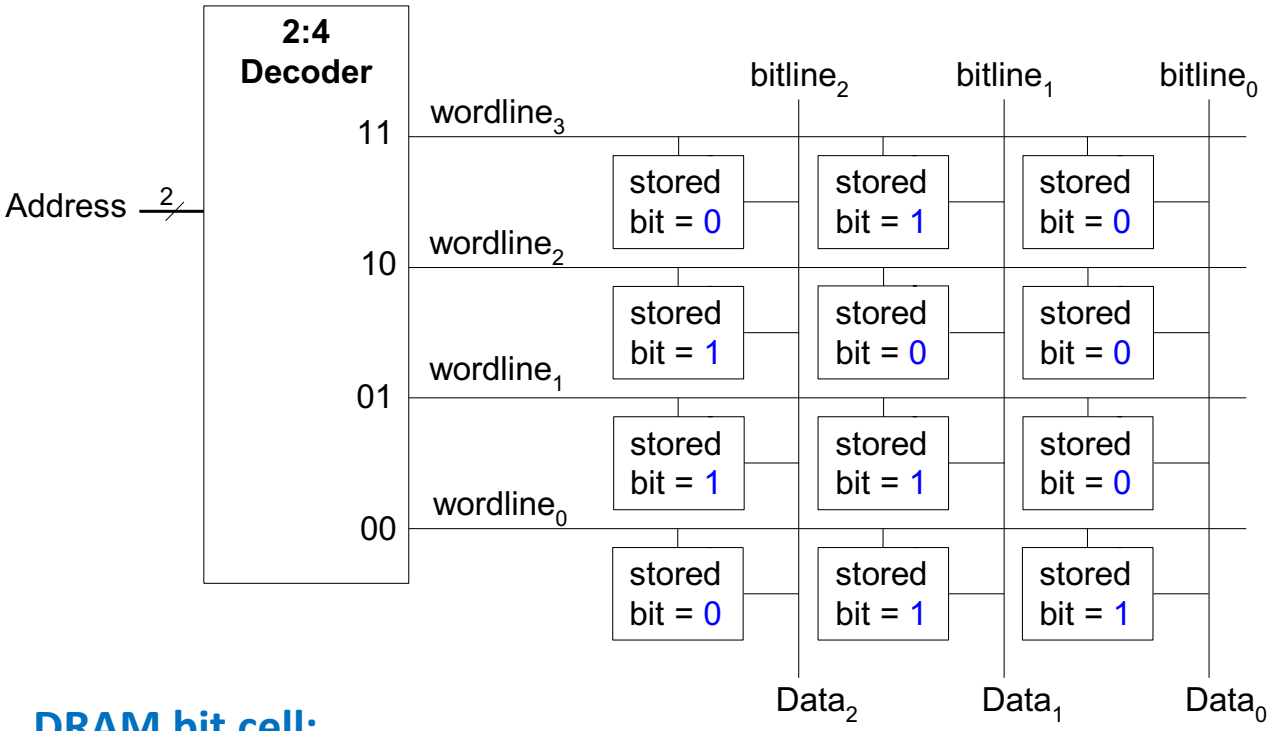

bitline
wordline
stored
bit = 1
+ +
bitline
wordline
stored
bit = 0

SRAM

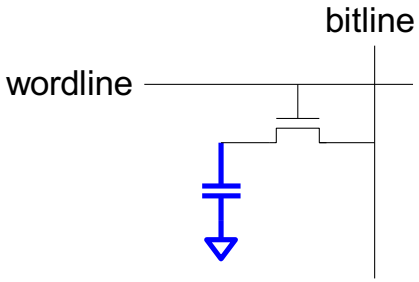

bitline
wordline
stored
bit


bit memorizzato
linea_di_bit
linea_di_bit
linea_di_parola

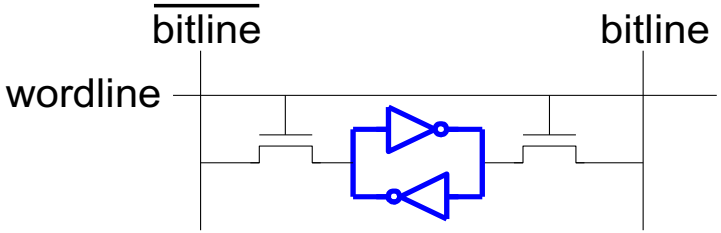
Memory Arrays Review



DRAM bit cell:



SRAM bit cell:



DRAM vs SRAM

- DRAM è più lenta
- Scrittura e refresh (millisecondi) inducono un consumo maggiore di energia
- DRAM, più economiche

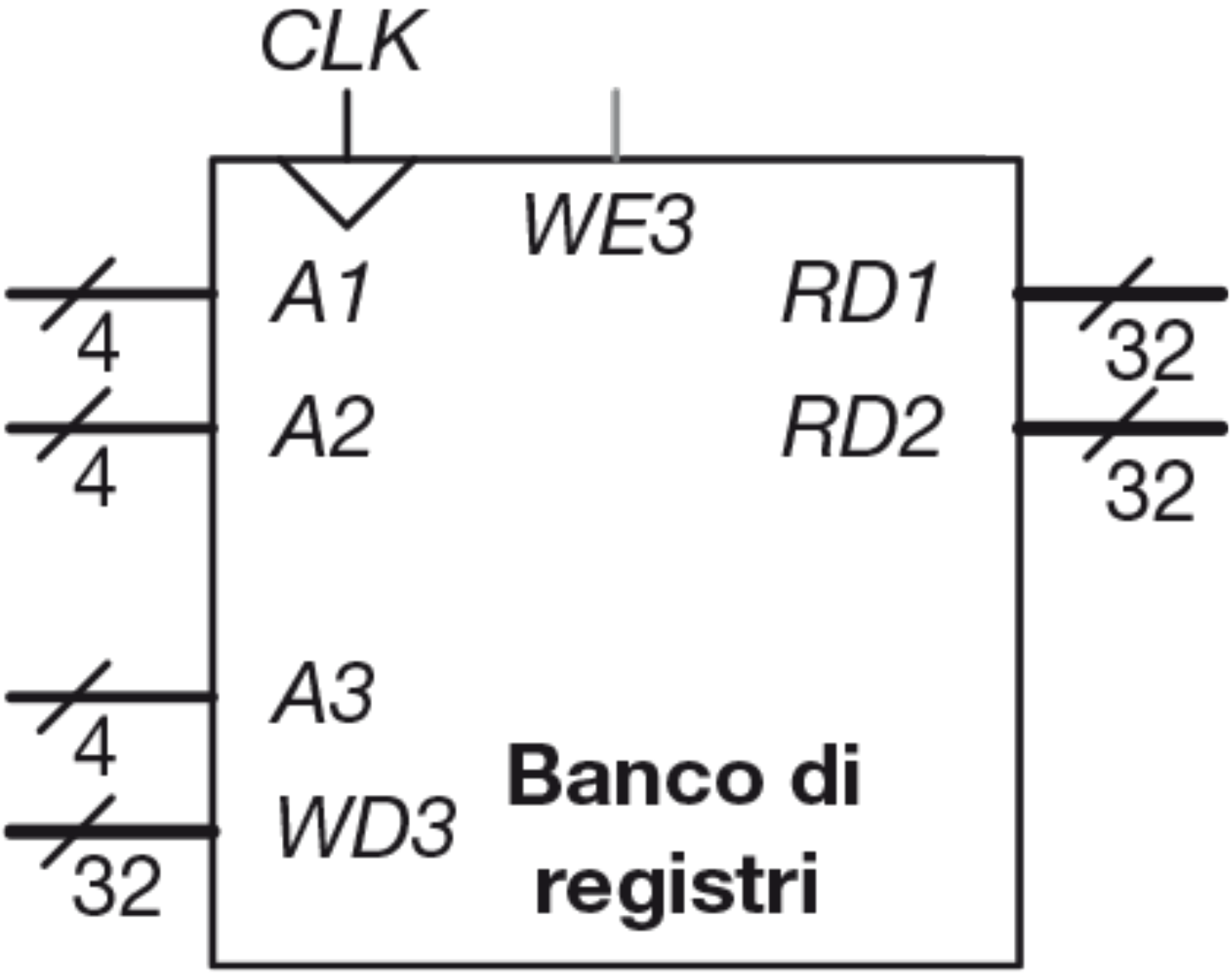
Memory Type	Transistors per Bit Cell	Latency
flip-flop	~20	fast
SRAM	6	medium
DRAM	1	slow

DRAM vs SRAM

- DRAM è più lenta
- Scrittura e refresh (millisecondi) inducono un consumo maggiore di energia
- DRAM, più economiche

Memory Type	Transistors per Bit Cell	Latency
flip-flop	~20	fast
SRAM	6	medium
DRAM	1	slow

registri


CLK
WE3
A1
A2
A3
WD3
RD1
RD2
4
4
4
32
32
32
Banco di registri

DRAM vs SRAM

- DRAM è più lenta
- Scrittura e refresh (millisecondi) inducono un consumo maggiore di energia
- DRAM, più economiche

Memory Type	Transistors per Bit Cell	Latency
flip-flop	~20	fast
SRAM	6	medium
DRAM	1	slow

cache

DRAM vs SRAM

- DRAM è più lenta
- Scrittura e refresh (millisecondi) inducono un consumo maggiore di energia
- DRAM, più economiche

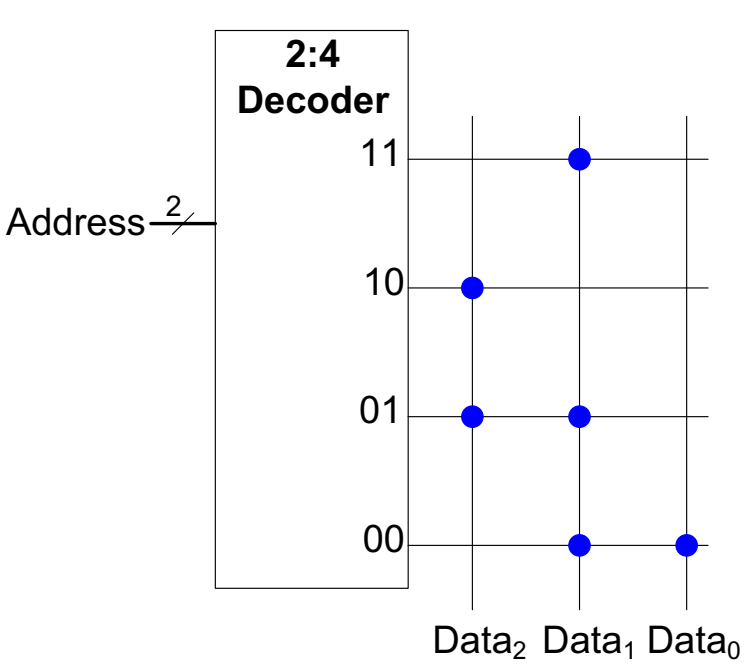
Memory Type	Transistors per Bit Cell	Latency
flip-flop	~20	fast
SRAM	6	medium
DRAM	1	slow

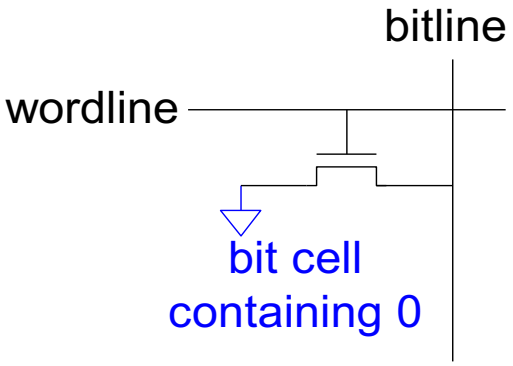
memoria centrale

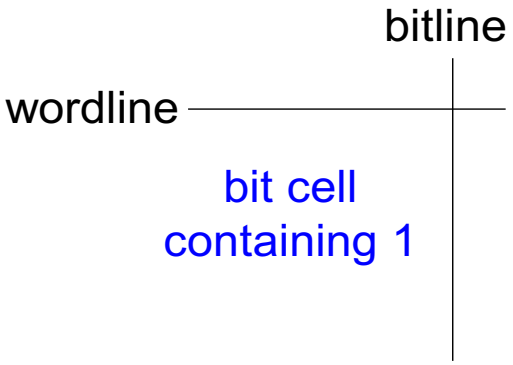
DDR SDRAM

- (DDR) Double data rate (S) synchronous (DRAM) dynamic random access memory
- Le operazioni di lettura e scrittura sono sincronizzate da un clock
- Le trasmissioni avvengono sia nel rising-edge del clock (0→1) che nel che falling-edge (1→0)
- In questo modo il data-bandwith doppio rispetto alla frequenza di clock (double pumping)

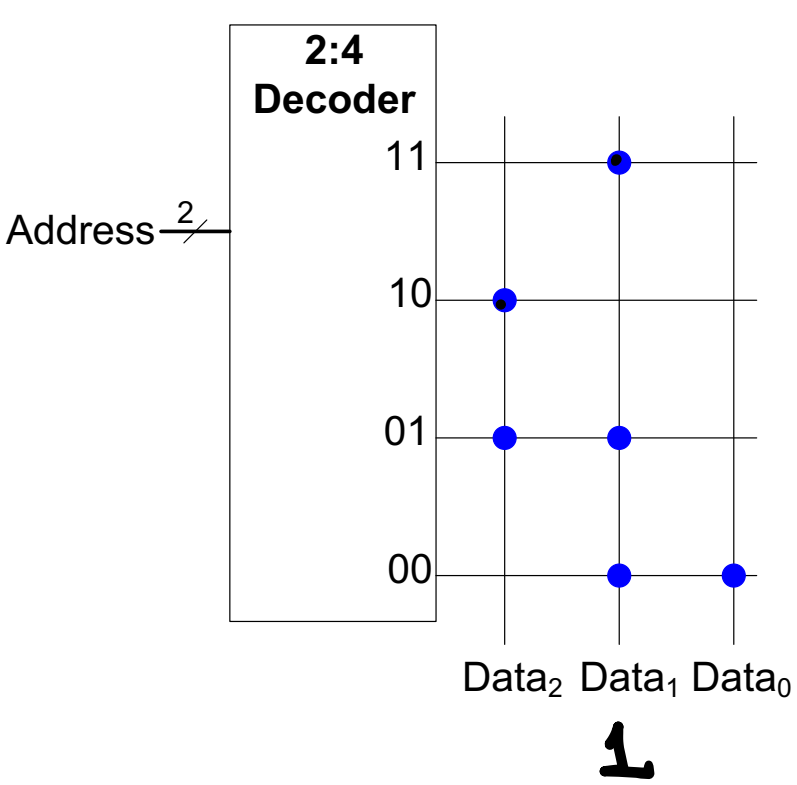
ROM: Dot Notation







ROM Storage



Address	Data		
11	0	1	0
10	1	0	0
01	1	1	0
00	0	1	1

depth

width

Fuse Programmable ROM