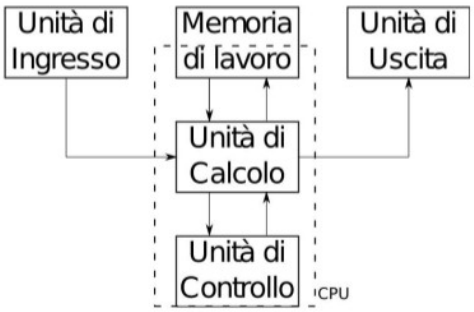
Lezione 35 Gerarchia di memoria

martedì 5 marzo 2024 17:09

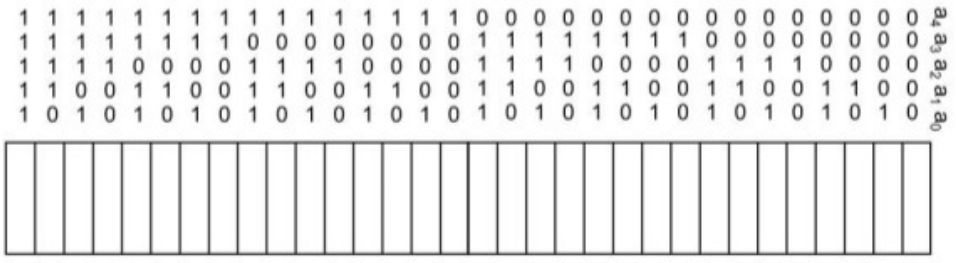
Andiamo a vedere ora come si ha l'interazione con la memoria . Per memoria si intende un dispositivo (esterno al processore) che permette di memorizzare informazioni . **Per memoria si intende generalmente la memoria primaria/di lavoro secondo architettura Von Neumann**.



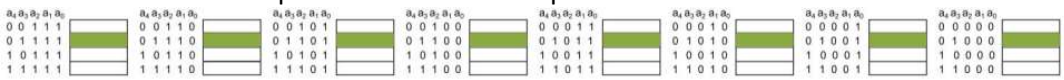
Quindi la memoria , generalmente viene suddivisa in 3 macro categorie :

1. Interna
    a. Sono quelle memorie che vanno a lavorare sui registri
    b. Lettura/scrittura dati da/verso processore
2. Principale
    a. DRAM
    b. Memorizza istruzioni e dati dei nostri programmi
    c. Più lenta della memoria interna (3 a 1 colpi di clock)
3. Secondaria
    a. Memoria di massa
    b. Dimensioni della memoria più grande, ma più lenta

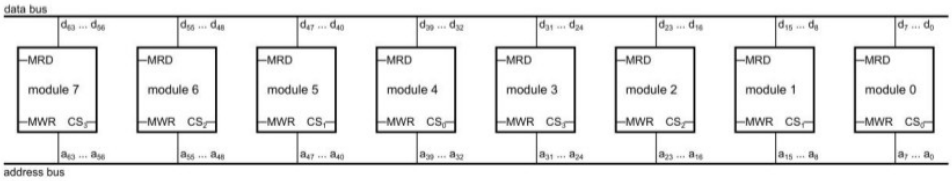
Quindi in virtù delle tecnologie , si cerca di arrivare ad un trade-off : memorie costruite con transistor (aumenta il costo per bit immagazzinato), ma diminuisce la velocità di accesso/latenza.
Quindi costi vs tempo accesso.
Andiamo a vedere ora su quella principale (modello flat):



Ma da notare che l'organizzazione logica della memoria non corrisponde ad organizzazione dei banchi di ram : non è facile accedervi. Quindi vista questa difficoltà, andiamo ad organizzare la ram in **banchi di memoria** : possibilità di letture in parallelo



In generale , si potrebbe essere interessati a porzioni di dati (8,16,32,64) byte di dati , quindi bisogna prevedere un meccanismo che faccia ciò :



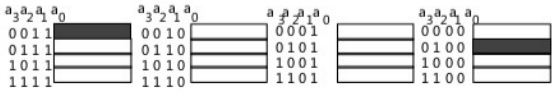
Abilitando solo il numero opportuno disegnali CS(chip select): questo segnale **(aggiuntivo)** viene generato dal processore e mandato sul bus dati/indirizzo/control , in modo che il processore legga n

moduli (n moduli con log2n per il numero di bit). Questo segnale rappresenta la taglia dell'operando in memoria con il quale abbiamo a che fare (8,16,32,64 -> byte , word , long word, quadword). Sembrerebbe funzionare , ma in realtà non si è tenuto conto del disallineamento dei byte :

padding :

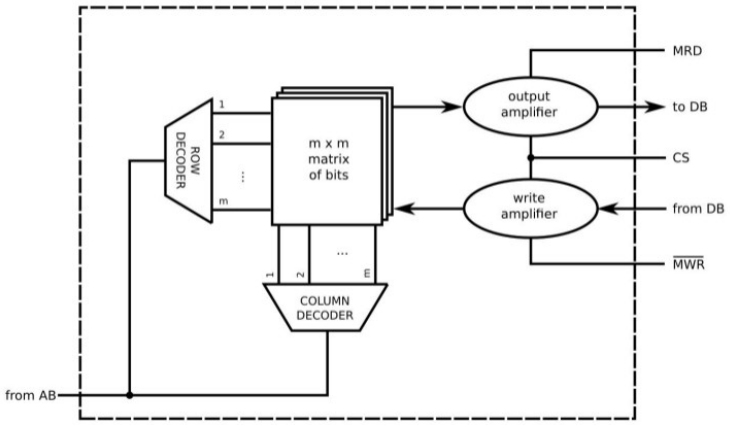
Quattro byte allineati sullo stesso indirizzo di riga.

Quattro byte disallineati.

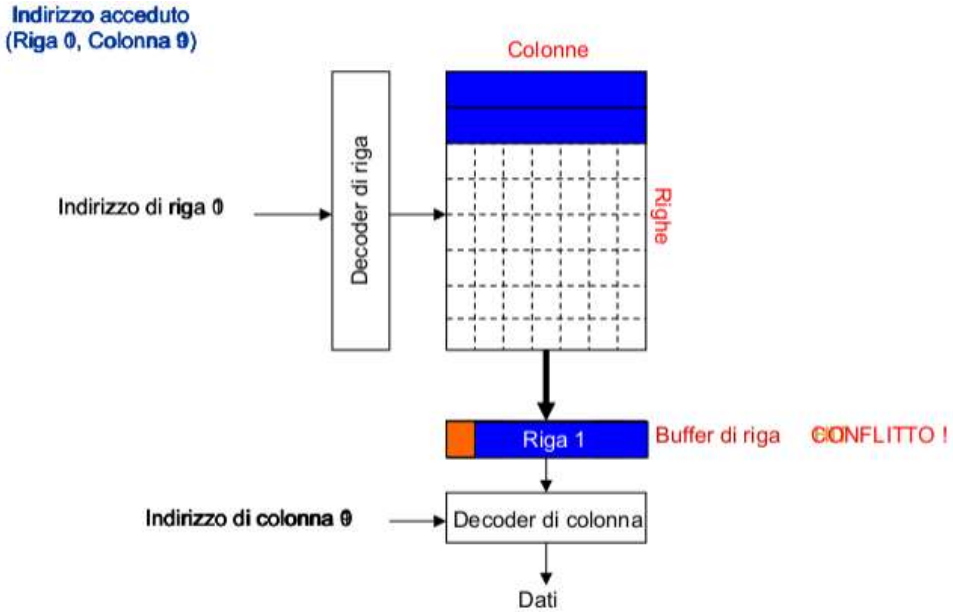


Due byte disallineati

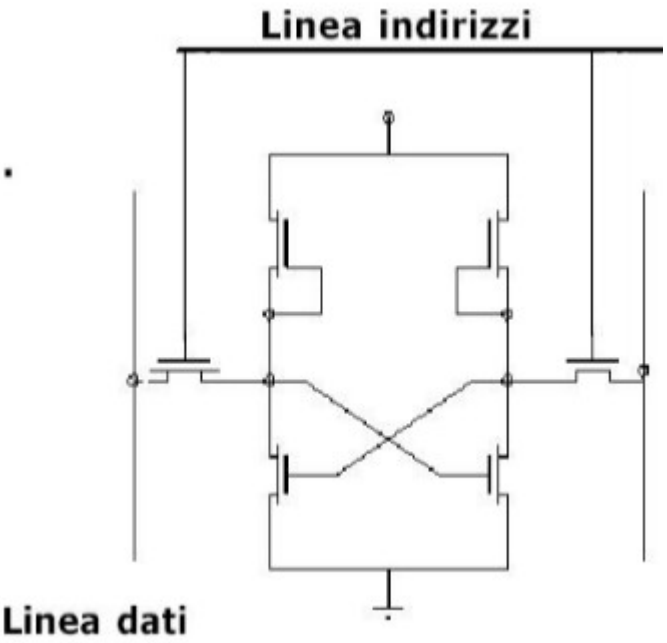
Notiamo che al massimo il disallineamento massimo è di 2 byte. Quindi in generale la memoria viene organizzata secondo matrici , nelle quali l'indirizzo di memoria che forniamo viene utilizzato per individuare una riga ed una colonna :



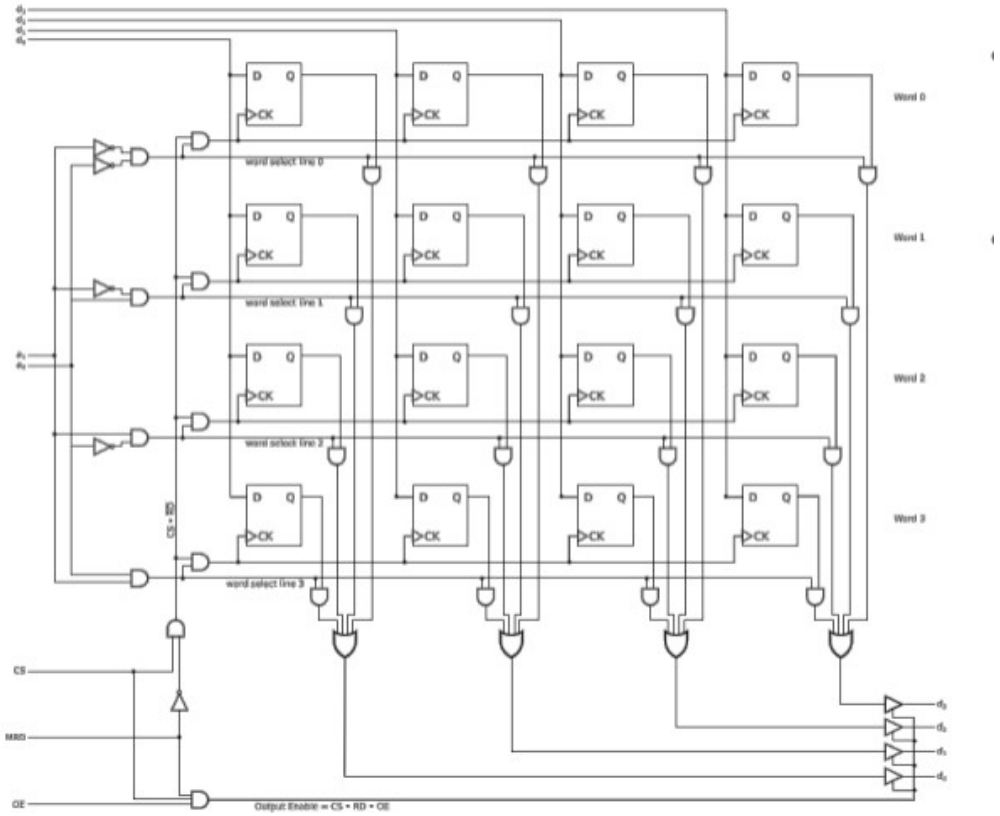
Attenzione però al ritardo di propagazione . La quale circuiteria è la seguente :



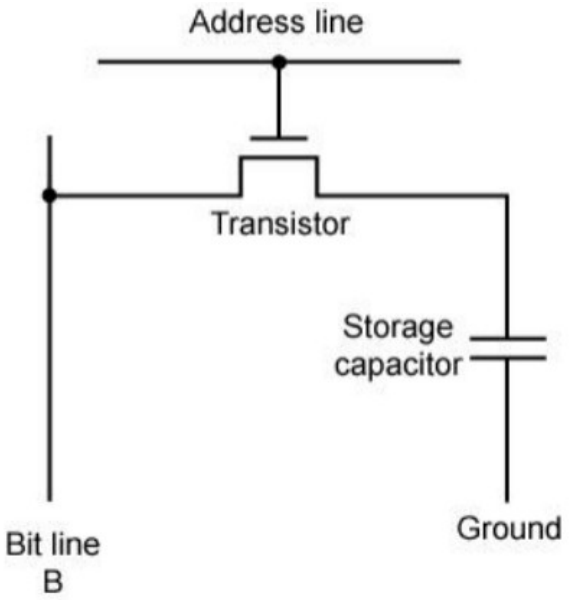
Tornando ai tempi di propagazione : le memorie per ovviare a ciò si usano di buffer tamponi (buffer riga), il quale contiene i dati che poi verranno amplificati . In dettaglio le ram sono fatte cosi' : un flip flop fatto da 6 transistor (aumenta le prestazioni , ma aumentano i costi):



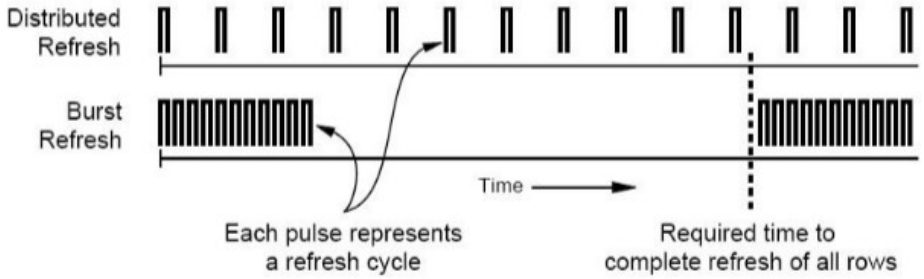
In dettaglio :



Tutto bello fino a qui , ma visto il costo e le attuali capacità della memoria questa soluzione non è accettabile : costebbere assai realizzarla, quindi basandosi su quanto detto fino ad ora : si usano le **ram dinamiche**: utilizza un condensatore ed un transistor (carico vale 1 , scarico vale 0) , il quale funziona da interruttore.



Quindi se il condensatore è carico si ha lettura/scrittura : se si ha tensione alta (1) su linea del bit si carica il condensatore e si scrive il dato , mentre per la lettura si seleziona la linea di indirizzo che si vuole , la carica del condensatore viene trasferita (attraverso amplificatore) al bit di linea ed infine si ripristina la carica del condensatore. Ricordiamo che il tempo di scarica/carica del condensatore è molto veloce . In più bisogna stare attenti anche al refresh delle memorie : tempo che impiega per rileggere i dati. In antitesi a questo tipo di refresh si ha quello **distribuito**: ovvero il refresh viene fatto a pezzi ogni 64 ms :



Ma come contro ha che non si possono perdere pezzi di memoria , ma come pro ha che abbatte i tempi (si ferma solo quando c'è richiesta di dati dal banco di memoria)