

## 6

# Esercitazione 6 - Logica programmabile e memorie a semiconduttore

▼ Creatori originali: @Francesco Ambrosino, @Gianbattista Busonera

---

## Esercizio 1 - Logica programmabile 1

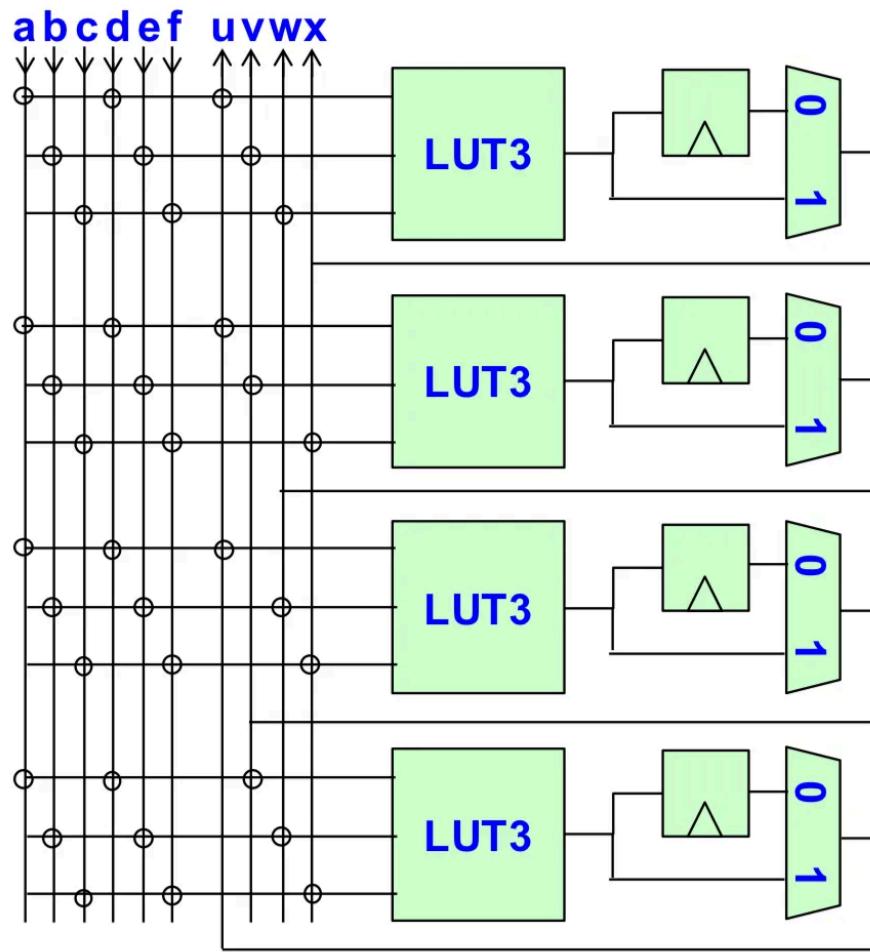
▼ Creatore originale: @Francesco Ambrosino

Determinare

- la programmazione delle connessioni
- la programmazione delle LUT
- la programmazione dei multiplexer

per realizzare le funzioni logiche:

- $u = abc + def$
- $v = (ab)^* = \overline{ab}$
- $w = def$
- $x = \text{NON USATO}$



## ▼ Considerazioni iniziali

1.  $a, b, c, d, e, f$  hanno freccia entrante quindi rappresentano gli **ingressi**.
2.  $u, v, w, x$  hanno freccia uscente quindi rappresentano le **uscite**.
3.  $LUT3$  sta per lookup table a 3 ingressi (possiede quindi  $2^3 = 8$  celle di memoria da 3 bit in cui possiamo memorizzare una qualsiasi funzione logica di questi 3 ingressi).
4. Su un cavo collegato alla LUT può o non essere attivo nessun ingresso/uscita o esserne attivo solo uno, non più di un pallino può essere riempito per ogni riga.
5. Ogni mux (multiplexer) all'uscita delle LUT è collegato ad una linea di una uscita (**l'uscita del mux comanda le uscite del circuito**). Seguendo infatti l'uscita dei mux si può vedere che:
  - a. La prima LUT comanda l'uscita x;

- b. La seconda LUT comanda l'uscita w;
  - c. La terza LUT comanda l'uscita v;
  - d. La quarta LUT comanda l'uscita u.
6. Leggendo le funzioni logiche da realizzare possiamo affermare che vogliamo realizzare un **circuito puramente combinatorio** (visto che le uscite dipendono solamente dagli ingressi), quindi non avremo bisogno di utilizzare i flip-flop (ne vedremo un esempio di utilizzo nel prossimo esercizio), per questo motivo i valori di tutti i mux saranno a 1 in questo esercizio (l'uscita delle LUT comanda DIRETTAMENTE le uscite).
- Non è infatti necessario sincronizzare l'uscita con il valore di un ipotetico clock (che comanderebbe anche il FF).

## ▼ Soluzione

**1** Osserviamo in primis che  $u = abc + def$  è **funzione di 6 ingressi**, non è quindi realizzabile con una sola LUT (poiché **ogni LUT ha massimo 3 ingressi**).

**2** In secondo luogo notiamo che  $def$ , operando di  $u$ , definisce l'uscita  $w$ , possiamo dunque riutilizzarlo riducendo gli ingressi di "u":  $u = abc + w$  che però è funzione di 4 ingressi e dunque non ancora realizzabile.

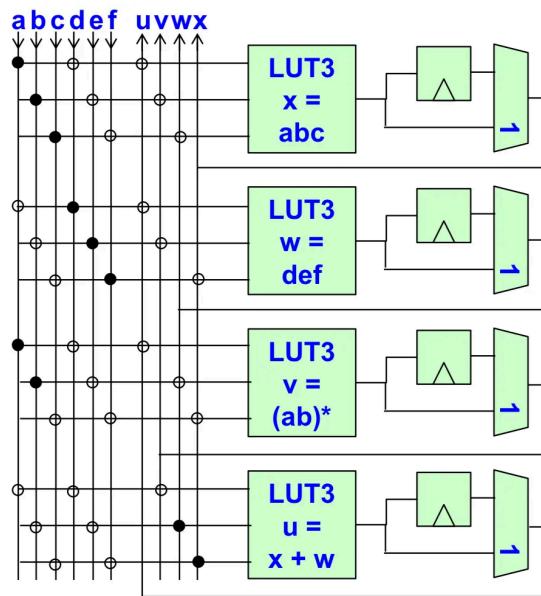
**3** Possiamo utilizzare  $x$ , lasciato libero dalle specifiche del testo, e porlo uguale a  $abc$ .

Sistemiamo così la situazione per  $u = x + w$  con  $x = abc$  e  $w = def$ .

**4** Infine guardando a  $v = (ab)^*$  notiamo subito che non abbiamo problemi di implementazione in questo caso perché  $v$  è già funzione con meno di tre ingressi.

Nella tabella che segue ci sono:

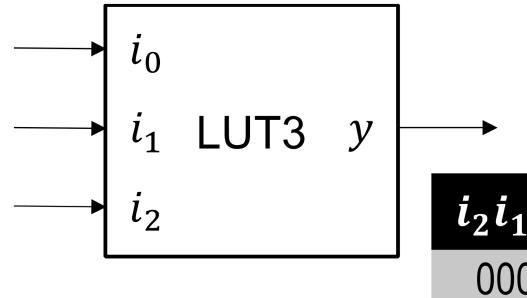
- connessioni definite (un pallino nero definisce una connessione riga/colonna),
- funzioni logiche esplicitate dentro le LUT (NOTA: le LUT contengono della logica combinatoria modificabile, non farti problemi su come le LUT capiscano che gli ingressi sia in OR, in AND, negati o affermati... c'è della logica dentro.),
- uscite dei mux tutte a 1 per quanto spiegato prima.



Volendo la LUT3( $u$ ) si poteva realizzare collegando gli ingressi  $c$ ,  $v$  e  $w$ .

L'esercizio non è però finito qui poiché dobbiamo **implementare la programmazione** delle LUT.

Lo schema generale è



da questo scriveremo le **tabelle di verità** per le quattro LUT.

! La terza e la quarta LUT (che comandano rispettivamente le uscite  $v$  e  $u$ ) hanno attivi solo 2 ingressi, l'ingresso non attivo deve essere considerato un *don't care*

$i_2 i_1 i_0$	LUT3x	LUT3w	LUT3v	LUT3u
000	0	0	1	0
001	0	0	1	0
010	0	0	1	1
011	0	0	0	1
100	0	0	1	1
101	0	0	1	1
110	0	0	1	1
111	1	1	0	1

La tabella è di facile comprensione, l'unica attenzione da porre è quella nel **far corrispondere  $i_0$ ,  $i_1$  e  $i_2$  agli ingressi rispettivi delle LUT**. Ad esempio nella  $LUT_3(v)$   $i_0$  corrisponde ad  $a$ ,  $i_1$  corrisponde a  $b$  e  $i_2$  è libero, mentre invece nella  $LUT_3(w)$  è  $i_0$  l'ingresso libero (da interpretare come *don't care*).

# Esercizio 2 - Logica programmabile 2

▼ Creatore originale: @Francesco Ambrosino

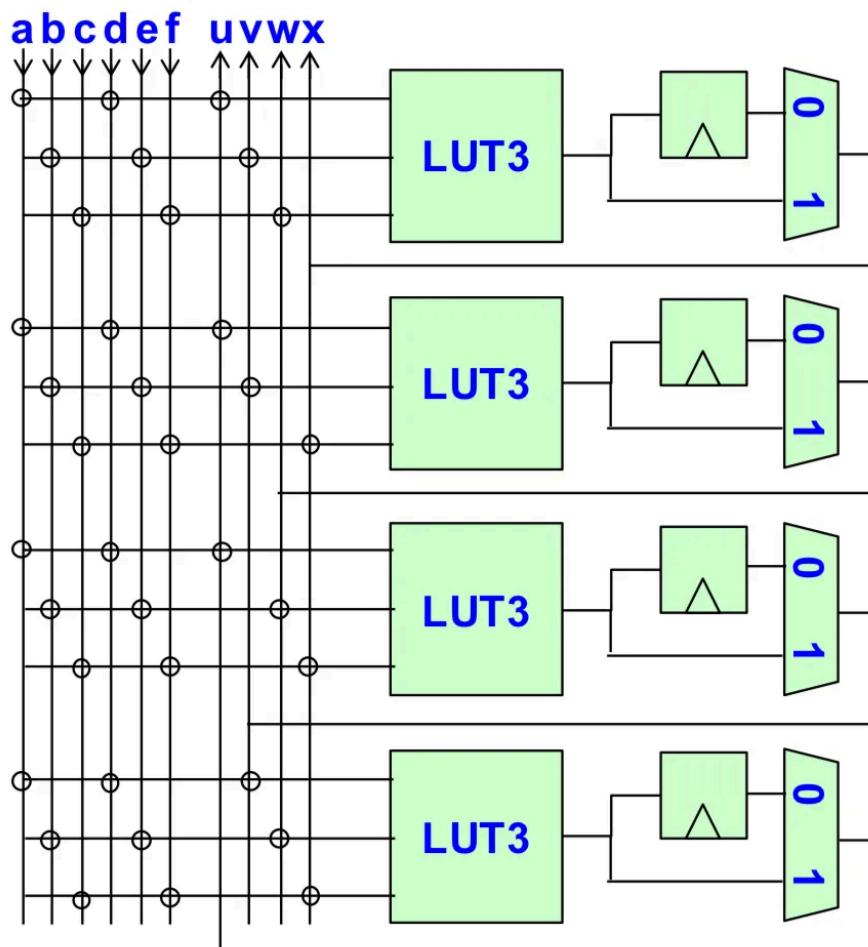
## Obiettivo 1 - Registro SIPO

Determinare

- la programmazione delle connessioni
- la programmazione delle LUT
- la programmazione dei multiplexer

per realizzare le funzioni logiche:

Registro SIPO (Serial Input Parallel Output) a 4 bit con ingresso seriale  $S_{in} = a$  e uscite  $Q_{0,1,2,3} = (u, v, w, x)$ .



Si noti che  $S_{in} = a$  e che le uscite  $Q_{0\dots3} = u, v, w, x$

## ▼ Considerazioni iniziali

UGUALI ALLO SCORSO ESERCIZIO:

1.  $a, b, c, d, e, f$  hanno freccia entrante quindi rappresentano gli ingressi.
2.  $u, v, w, x$  hanno freccia uscente quindi rappresentano le uscite.
3.  $LUT3$  sta per lookup table a 3 ingressi infatti ad ogni LUT sono collegati 3 cavi.
4. Su un cavo collegato alla LUT può o non essere attivo nessun ingresso/uscita o esserne attivo solo uno, non più di un pallino può essere riempito per ogni riga.
5. Ogni mux (multiplexer) all'uscita delle LUT è collegato ad una linea di una uscita (l'uscita del mux comanda le uscite del circuito).

DIVERSA:

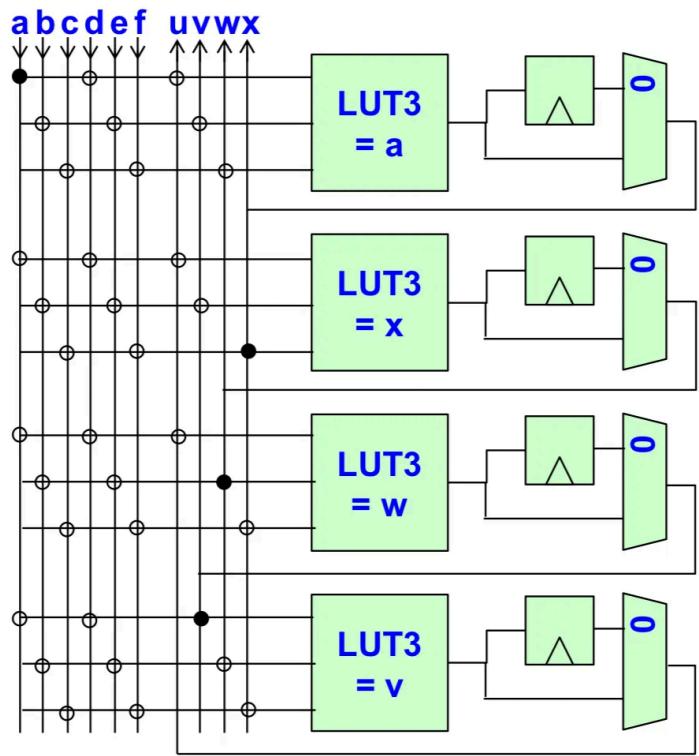
Dovendo implementare un registro sappiamo di star lavorando con un circuito **sequenziale**, dobbiamo dunque **utilizzare i flip-flop**.

Quanti flip-flop dobbiamo utilizzare? Registro a 4 bit → ci servono 4 flip-flop, per questo **i valori di tutti i mux saranno a 0 in questo esercizio** (l'uscita delle LUT NON comanda direttamente le uscite ma lo fa attraverso il flip-flop).

## ▼ Soluzione

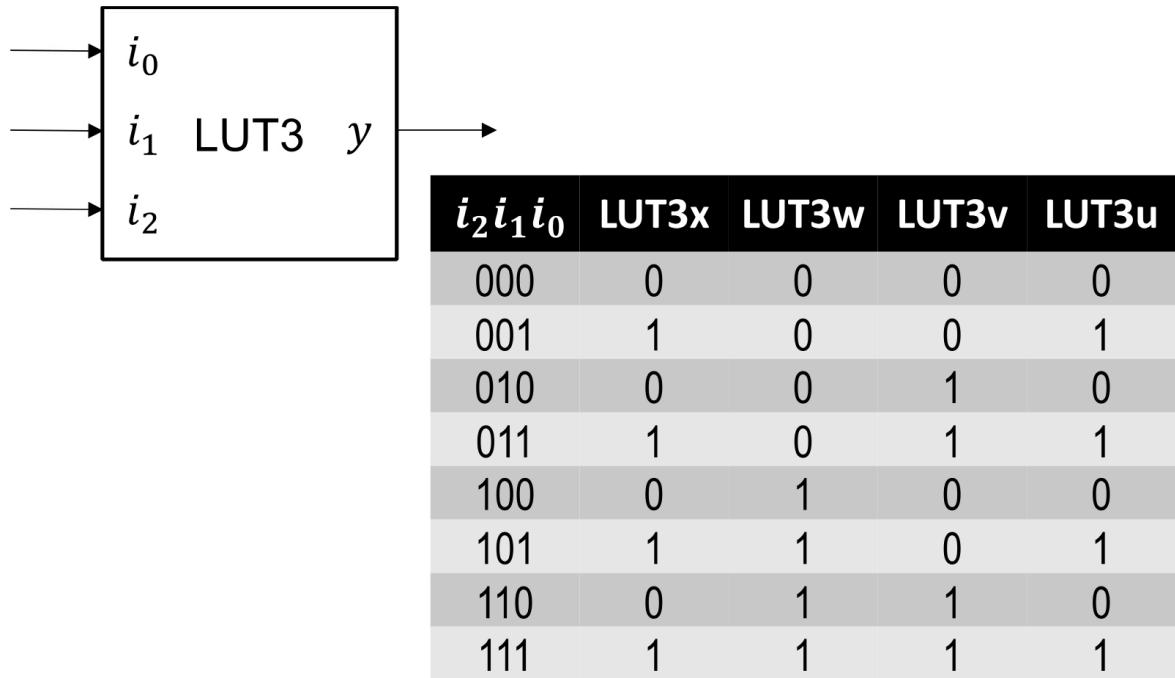
Dobbiamo realizzare un registro SIPO (banalmente uno shift register), cioè un circuito in cui:

- il 1° flip-flop riceve (attraverso la prima LUT) l'ingresso seriale  $a$
- il 2° flip-flop riceve (attraverso la seconda LUT) l'uscita del 1° flip-flop
- il 3° flip-flop riceve (attraverso la terza LUT) l'uscita del 2° flip-flop
- il 4° flip-flop riceve (attraverso la quarta LUT) l'uscita del 3° flip-flop



In questo esercizio le LUT non implementano una vera e propria logica combinatoria, banalmente riportano all'ingresso del flip-flop l'ingresso che ricevono a loro volta.

Riporto di seguito le tabelle di verità.



Come per l'esercizio 1 l'unica difficoltà può essere nel capire la corrispondenza tra gli ingressi:

- $LUT_3(x) = a = i_0$
- $LUT_3(w) = x = i_2$
- $LUT_3(v) = w = i_1$
- $LUT_3(u) = v = i_0$

## Obiettivo 2 - Registro PIPO

Determinare

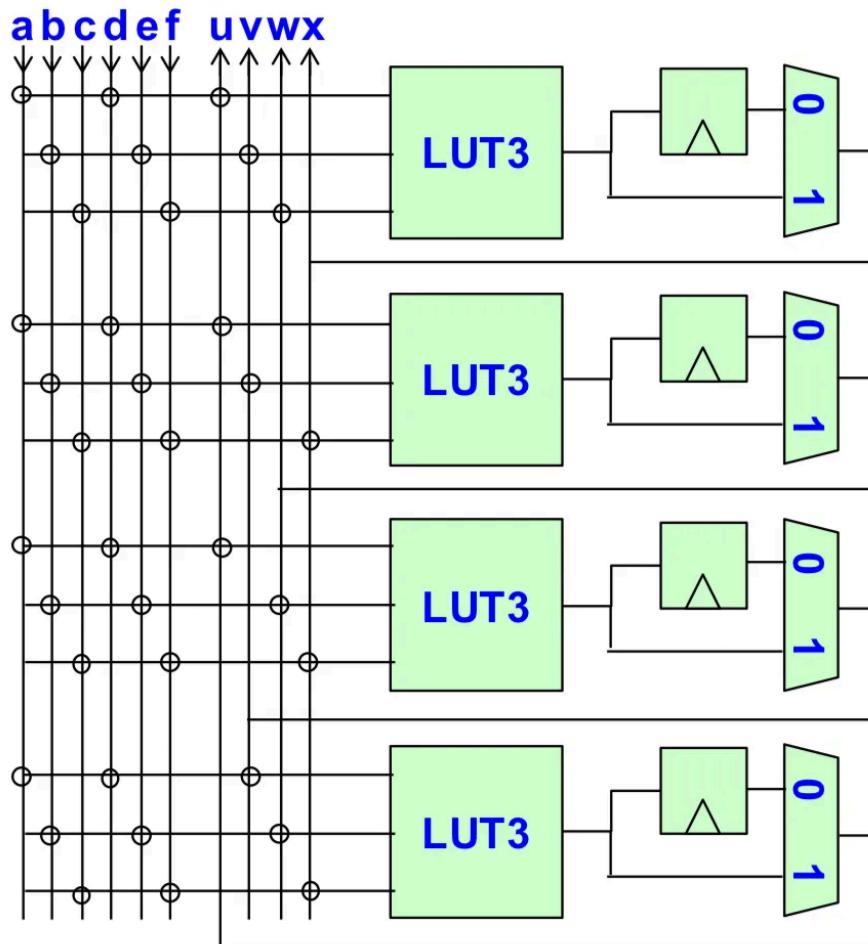
- la programmazione delle connessioni
- la programmazione delle LUT
- la programmazione dei multiplexer

per realizzare le funzioni logiche:

Registro PIPO (Parallel Input Parallel Output) a 3 bit con:

- ingresso parallelo  $(a, b, c)$

- reset ( $e$ ) attivo alto
- uscite  $Q_{0,1,2} = (u, v, w)$ .
  - L'uscita  $w$  corrisponde all'ingresso  $a$ ;
  - L'uscita  $v$  corrisponde all'ingresso  $b$ ;
  - L'uscita  $u$  corrisponde all'ingresso  $c$ .



## ▼ Considerazioni iniziali

UGUALI ALLO SCORSO OBIETTIVO:

1.  $a, b, c, d, e, f$  hanno freccia entrante quindi rappresentano gli ingressi.
2.  $u, v, w, x$  hanno freccia uscente quindi rappresentano le uscite.
3.  $LUT3$  sta per lookup table a 3 ingressi infatti ad ogni LUT sono collegati 3 cavi.

4. Su un cavo collegato alla LUT può o non essere attivo nessun ingresso/uscita o esserne attivo solo uno, non più di un pallino può essere riempito per ogni riga.
5. Ogni mux (multiplexer) all'uscita delle LUT è collegato ad una linea di una uscita (l'uscita del mux comanda le uscite del circuito).

MODIFICATA:

Come prima, dovendo implementare un registro sappiamo di star lavorando con un circuito sequenziale, dobbiamo dunque utilizzare i flip-flop.

Quanti flip-flop dobbiamo utilizzare? Registro a 3 bit → ci servono 3 flip-flop.

La quarta LUT verrà utilizzata in modo asincrono (senza flip-flop) per gestire il reset.

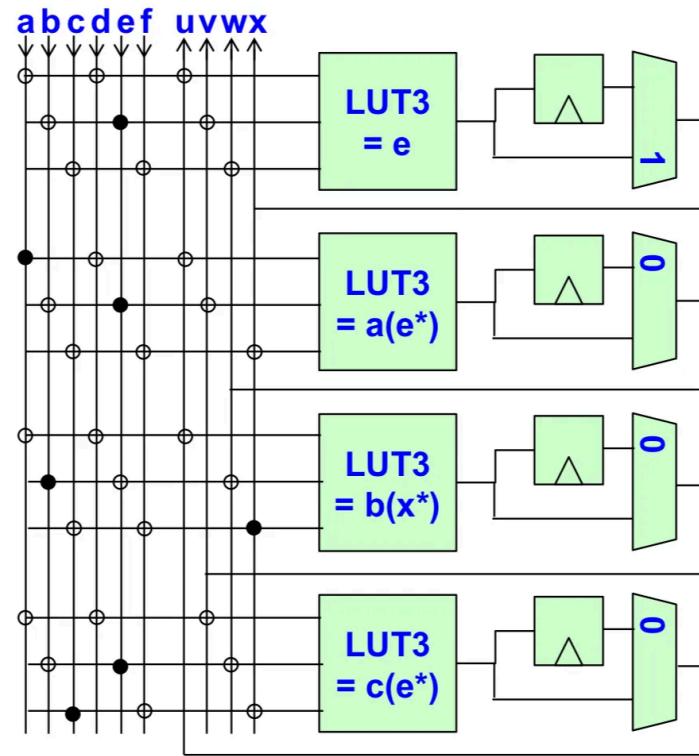
## ▼ Soluzione

Le specifiche di progetto specificano che le 3 uscite del registro debbano essere  $u, v, w$ , da questo deduciamo che le LUT che comandano quelle uscite (le ultime tre) debbano avere il valore del mux alle rispettive uscite collegato a 0 mentre la prima LUT, che comanda  $x$ , avrà in ingresso  $e$  e il mux che comanda la sua uscita a 1.

I 3 ingressi del registro parallelo sono  $a, b, c$  per cui avremo che:

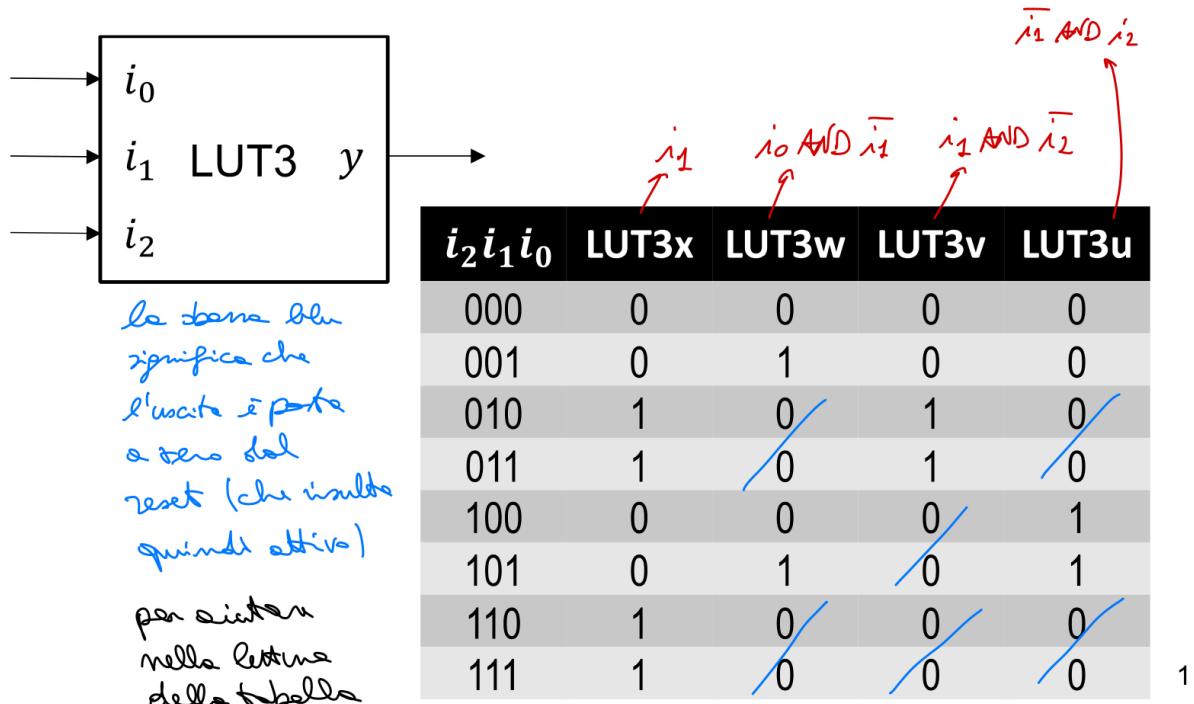
- il 2° flip-flop riceve (attraverso la seconda LUT) in ingresso  $a$
- il 3° flip-flop riceve (attraverso la terza LUT) in ingresso  $b$
- il 4° flip-flop riceve (attraverso la quarta LUT) in ingresso  $c$

Questo però non basta perché dobbiamo collegare in ingresso a queste 3 LUT anche il **reset**. Per la seconda e la quarta LUT possiamo collegare  $e$  in maniera diretta, nell'architettura della terza non possiamo collegare  $e$  poiché si trova sulla stessa riga (sullo stesso cavo) di  $b$  che deve essere ingresso di questa LUT, possiamo risolvere questo problema semplicemente collegando  $x$  (uscita asincrona della prima LUT) a cui è collegato in ingresso  $e$  (si ha infatti  $x = e$ ).



NOTA: l'ingresso di reset compare negato e in AND con l'ingresso parallelo in modo che quando è posto a 1, la sua negazione è 0 e quindi manda a 0 l'uscita delle LUT indipendentemente dai valori di  $a, b, c$  perché questi sarebbero in AND con valore logico 0.

Riporto di seguito le tabelle di verità.



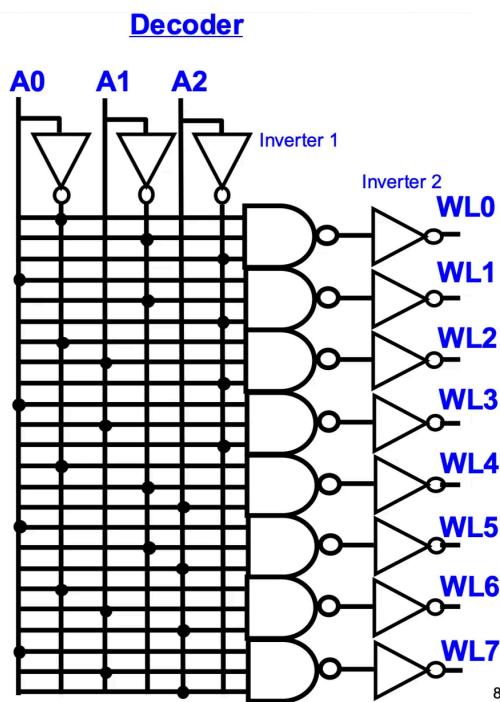
## Esercizio 3 - Decoder indirizzi e ritardo wordline

▼ Creatore originale @Francesco Ambrosino

### Obiettivo 1 - 64 celle

Si ha un banco di memoria DRAM con 8 wordlines connesse a  $N = 64$  celle ciascuna. Il pass transistor della cella ha una capacità di gate di  $0,1 \text{ fF}$  e la linea metallica di una wordline ha una resistenza complessiva di  $10 \Omega \cdot N = 640 \Omega$ .

- Progettare con porte CMOS il **decoder** che pilota le 8 wordlines a partire dai segnali di indirizzo.
- Sapendo che ogni transistore MOS usato nel decoder ha resistenza  $R_{ON} = 100 \Omega$  e capacità di gate  $C_g = 1 \text{ fF}$ , determinare il massimo ritardo di attivazione delle wordlines.



Commenti al progetto:

- 8 wordlines  $\rightarrow$  3 bit di indirizzo ( $A_0, A_1, A_2$ )
- Ogni bit di indirizzo viene preso sia affermato che negato (con gli Inverter 1)
- Ogni porta NAND ha 3 ingressi corrispondenti ai 3 bit di ingresso (affermati o negati)
- Alle uscite della NAND c'è un inverter (chiamato Inverter 2) così che i valori delle wordline corrispondono all'**AND logico dei 3 bit in ingresso** (NAND in serie con NOT da AND).
- Come visto negli altri esercizi il pallino pieno corrisponde alla connessione riga/colonna.

## ▼ Commento preliminare

La formula che si utilizzerà per il calcolo dei ritardi è:

$$t = 0.69\tau = 0.69 \cdot R_{\text{eq}} \cdot C_{\text{carico}}$$

Perché?

[LINK](#) teoria

[LINK](#) esercitazione 3

Il massimo ritardo di propagazione in una wordline è dato dall'attraversamento di un Inverter 1 + NAND + inverter 2.

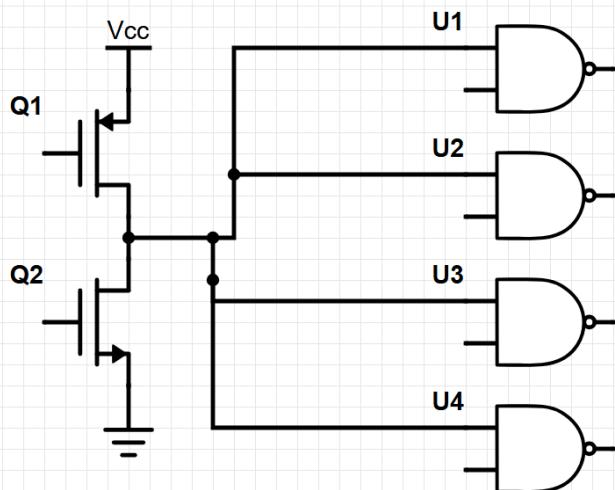


Vogliamo valutare il massimo ritardo di attivazione delle wordlines.

Attivare una wordline significa avere una "inverter 2" che commuta da uno stato basso a uno stato alto e, pertanto, ricerchiamo il tempo di propagazione di:

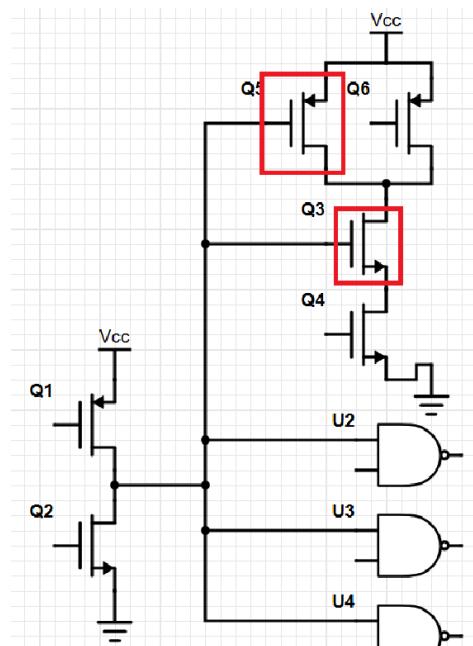
- Inverter 1: L→H (si noti che ogni inverter 1 ha un carico pari a 4 porte NAND)
- NAND: H→L
- Inverter 2: L→H (così la linea è attiva)

## ▼ Ritardo Inverter 1



Ogni Inverter 1 (a sinistra) è collegato a 4 porte NAND (sarebbero da tre ingressi ma è poco influente per la comprensione) che danno ognuna  $2C_g$  di carico (la singola uscita dell'inverter è collegata a 2 MOS per ogni porta NAND come in foto di fianco). Quindi la capacità di carico dell'Inverter 1 è pari a  $C_{carico} = 8C_g$ .

Nel dettaglio, per ogni NAND, si ha una situazione del tipo:



Per ogni NAND (da 2 ingressi anziché tre per facilitare il disegno) l'inverter "pilota" due MOS e, quindi, un carico di  $2C_g$  ognuna.

Come si può leggere dalla caption della prima immagine risulta dunque  $C_{carico} = 8C_g$ .

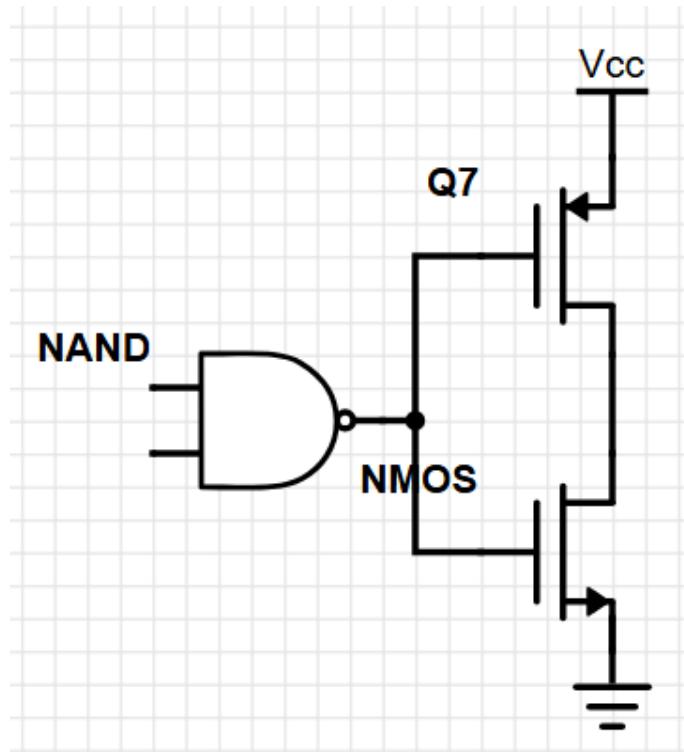
La sua  $R_{eq}$  sappiamo essere uguale proprio a  $R_{ON}$ , visto che sia nel caso di tempo di propagazione da L→H (si guarda la PUN), sia nel caso di tempo di propagazione H→L (si guarda la PDN) si ha a che fare con un solo MOS e quindi con una sola  $R_{ON}$ .

$$t = 0,69(R_{ON} \cdot 4 \cdot 2C_g) = 0,69 \cdot 100 \Omega \cdot 4 \cdot (2 \cdot 1 \text{ fF}) = 0,55 \text{ ps}$$

- Fattore “4”: ciascun output di Inverter 1 è collegato a 4 inputs di porta NAND
- Fattore “2”: ciascun input NAND è collegato a un gate PMOS ( $1 C_g$ ) e a un gate NMOS ( $1 C_g$ ), in parallelo

## ▼ Ritardo NANDs

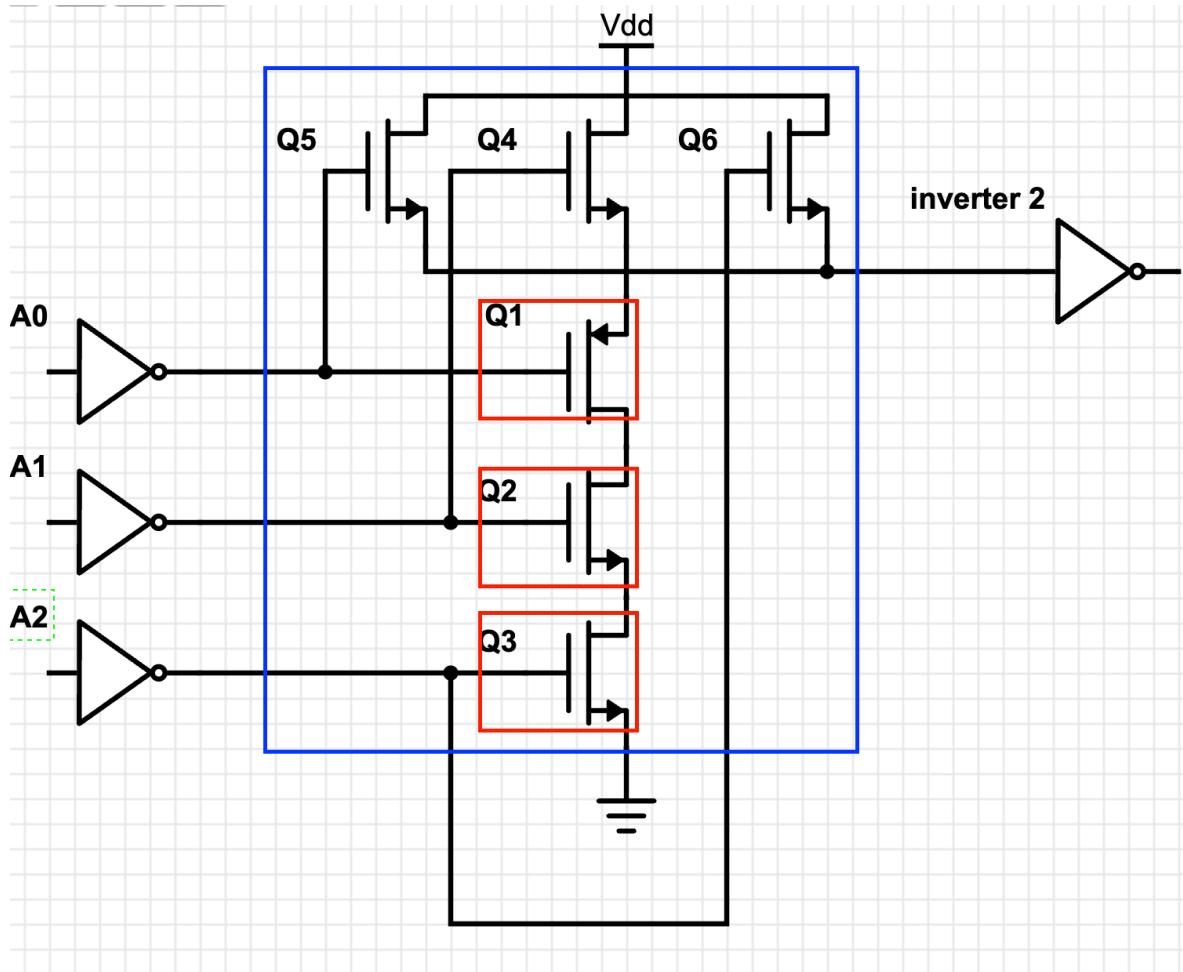
Ogni NAND è collegata a 1 inverter che dà  $C_{carico} = 2C_g$  (come prima data dai transistori in parallelo collegati all’ingresso).



In rosso i 2 MOS dell’inveter 2 ciascuno di capacità  $C_g$ . Si noti che la porta NAND sarebbe da tre ingressi anzichè 2.

La sua  $R_{eq}$  può assumere diversi valori a seconda del numero di MOS attivi al suo ingresso (a seconda che si consideri una propagazione da alto a basso o da basso a alto). Volendo noi calcolare il ritardo massimo prenderemo la situazione di caso

peggiore ovvero quando ci sono 3 MOS (uno per ogni linea) attivi in serie (quindi propagazione H $\rightarrow$ L) della porta NAND. Segue disegno esplicativo della nostra situazione.



Il riquadro blu evidenzia la porta NAND a 3 ingressi rappresentata in porte MOS.

In rosso sono evidenziati i 3 MOS in serie ciascuno con  $R=R_{on}$  che danno complessivamente contributo di  $3R_{on}=R_{eq}$  di caso peggiore.

Nota: i tre ingressi sono collegati a 3 inverter, per attivare gli nMOS bisogna avere 1 in uscita e quindi 0 in ingresso,

il fulcro del disegno rimane capire perché  $R_{eq}=3R_{on}$ .

$$0,69(3R_{on} \cdot 2C_g) = 0,41 \text{ ps}$$

- Fattore “3”: nel caso peggiore, l’uscita è collegata a GND con 3 MOS in serie

## ▼ Ritardo Inverter 2

Ogni Inverter 2 è collegato ad una wordline che è formata da 64 celle ognuna con  $C_{g, \text{cella}} = 0,1 \text{ fF}$  (dato del testo). Questo significa che la  $C_{\text{carico}} = 64C_{g, \text{cella}}$ .

La  $R_{\text{eq}}$  in questo caso deve tenere conto della resistenza dell'Inverter  $R_{\text{ON}}$  e della resistenza della wordline (non ideale)  $R_{\text{WL}} = 10 \Omega \cdot 64 \text{ celle} = 640 \Omega$  (dato del testo). Queste sono collegate in serie e ci danno quindi una  $R_{\text{eq}} = R_{\text{ON}} + R_{\text{WL}}$ .

$$\begin{aligned} & 0,69(R_{\text{ON}} + R_{\text{WL}})C_{\text{WL}} \\ & = 0,69(100 \Omega + 64 \cdot 10 \Omega) \cdot (64 \cdot 0,1 \text{ fF}) = 3,27 \text{ ps} \end{aligned}$$

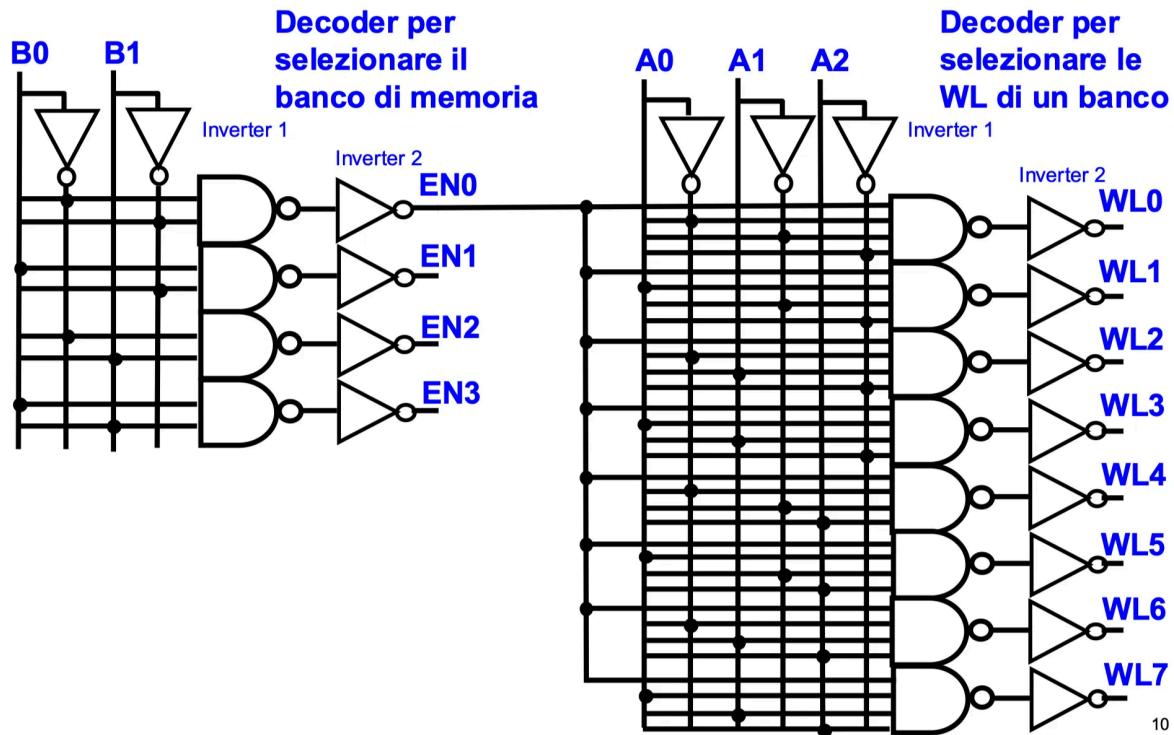
### ▼ Ritardo Totale

Sommo i tre valori parziali calcolati e ottengo:

$$0,55 \text{ ps} + 0,41 \text{ ps} + 3,27 \text{ ps} = 4,23 \text{ ps}$$

## Obiettivo 2 - 16 celle

Ripetere il progetto e il calcolo del ritardo se la memoria è partizionata in  $B = 4$  banchi ciascuno con 8 wordlines connesse a  $N = \frac{64}{B} = 16$  celle.



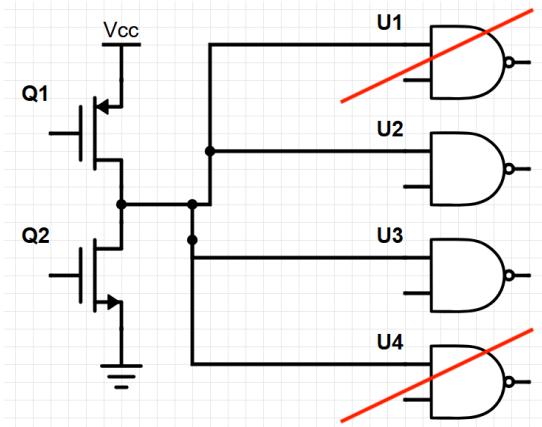
Commenti al progetto:

- 4 banchi  $\rightarrow$  2 bit di indirizzo ( $B_0, B_1$ )
- Ogni bit di indirizzo viene preso sia affermato che negato (con gli Inverter 1 del banco)
- Ogni porta NAND del banco ha 2 ingressi corrispondenti ai 2 bit di ingresso (affermati o negati)
- Alle uscite della NAND del banco c'è un inverter (chiamato Inverter 2 del banco) così che i valori delle wordline corrispondano all'AND logico dei 2 bit in ingresso, questo funge da vero e proprio abilitatore/attivatore del rispettivo decoder di linea.
- Ogni porta NAND della wordline ha ora infatti 4 ingressi corrispondenti ai suoi 3 bit di ingresso (affermati o negati) + l'ingresso relativo all'inverter 2 di banco.
- Come visto negli altri esercizi il pallino pieno corrisponde alla connessione riga/colonna.
- Ovviamente il disegno è incompleto poiché anche a  $EN_1, EN_2, EN_3$  è collegato un decoder di linea identico a quello visibile nel disegno collegato a  $EN_0$ .

Dividiamo la trattazione in:

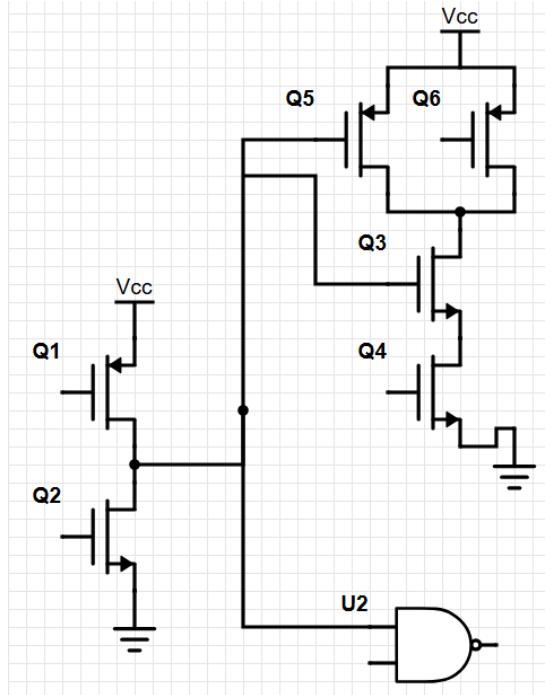
## ▼ Decoder banco

- Ritardo Inverter 1



Ogni Inverter 1 (a sinistra) è collegato a 2 porte NAND che danno ognuna  $2C_g$  di carico (la singola uscita dell'inverter è collegata a 2 MOS per ogni porta NAND). Quindi la capacità di carico dell'Inverter 1 è pari a  $C_{carico} = 4C_g$ .

Nota: l'immagine è ripresa dall'obiettivo precedente con lo scopo di far capire che la situazione è la medesima con un numero di porte logiche (NAND in questo caso) inferiore.



Rappresentazione più pulita della situazione in esame.

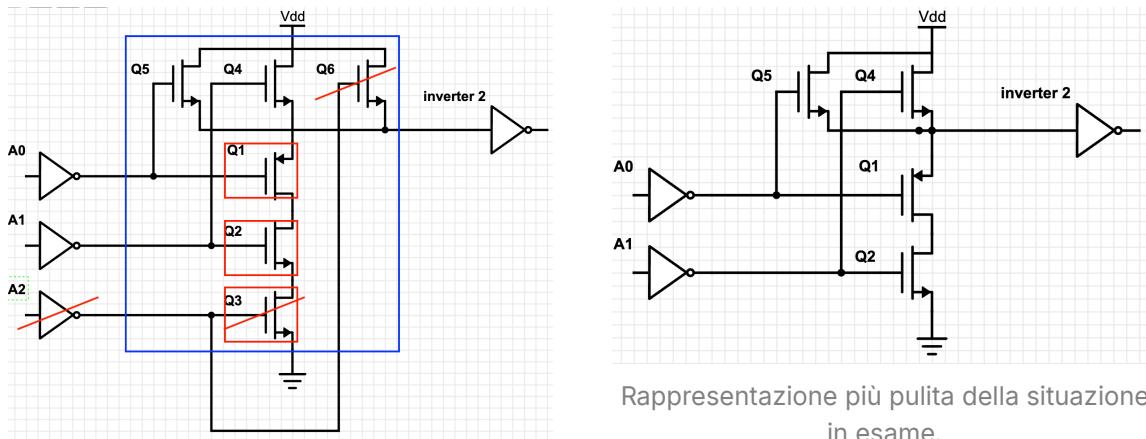
Ogni Inverter 1 è collegato a 2 porte NAND che danno ognuna  $2C_g$  di carico (data dai transistori in parallelo collegati all'ingresso). Quindi la capacità di carico dell'Inverter 1 è pari a  $C_{carico} = 4C_g$ .

La sua  $R_{eq}$  sappiamo essere uguale proprio a  $R_{ON}$  essendo un inverter.

- Ritardo NANDs

Ogni NAND è collegata a 1 inverter che dà  $C_{carico} = 2C_g$  (come prima data dai transistori in parallelo collegati all'ingresso).

La sua  $R_{eq}$  può assumere diversi valori a seconda del numero di MOS attivi al suo ingresso. Volendo noi calcolare il **ritardo massimo** prenderemo la situazione di caso peggiore ovvero quando ci sono 2 MOS (uno per ogni linea) attivi in serie e quindi  $R_{eq} = 2R_{ON}$ .



Rappresentazione più pulita della situazione in esame.

Come prima il riquadro blu evidenzia la porta NAND a 3 2 ingressi rappresentata in porte MOS e in rosso sono evidenziati i 3 2 MOS in serie ciascuno con  $R=R_{ON}$  che danno complessivamente contributo di  $2R_{ON}=Req$  di caso peggiore.

- **Ritardo Inverter 2**

Ogni Inverter 2 è collegato alle 8 NAND del wordline decoder (le abilita/disabilita tutte). Questo significa che la  $C_{carico} = 8 \cdot 2C_g$ , ricordando che le porte NAND che danno ognuna  $2C_g$  di carico.

La  $R_{eq}$  in questo caso è proprio solo quella dell'inverter, quindi pari a  $R_{ON}$ .

Di seguito sono riportate formule e valori di quanto detto fino ad ora a parole

- **Inverter 1:**  $0,69 \cdot [R_{ON} \cdot 2 \cdot (2C_g)] = 0,69 \cdot [100 \Omega \cdot 2 \cdot (2 \cdot 1 fF)] = 0,28 \text{ ps}$
- **NAND:**  $0,69 \cdot (2R_{ON} \cdot 2C_g) = 0,69 \cdot (2 \cdot 100 \Omega \cdot 2 \cdot 1 fF) = 0,28 \text{ ps}$
- **Inverter 2:**  $0,69 \cdot [R_{ON} \cdot 8 \cdot (2C_g)] = 0,69 \cdot [100 \Omega \cdot 8 \cdot (2 \cdot 1 fF)] = 1,1 \text{ ps}$

## ▼ Wordline decoder

- **Ritardo Inverter 1 (non è cambiato nulla)**

Ogni Inverter 1 è collegato a 4 porte NAND che danno ognuna  $2C_g$  di carico (data dai transistori in parallelo collegati all'ingresso). Quindi la capacità di carico dell'Inverter 1 è pari a  $C_{carico} = 8C_g$ .

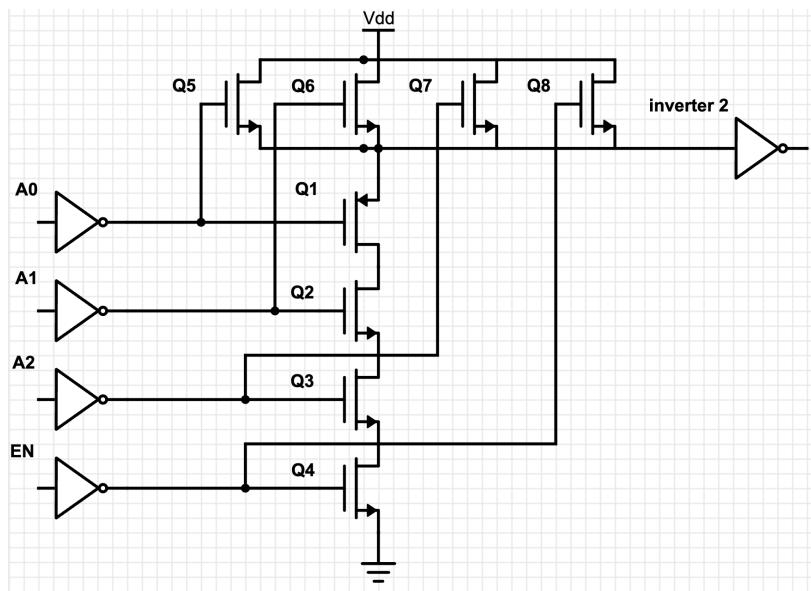
La sua  $R_{eq}$  sappiamo essere uguale proprio a  $R_{ON}$ .

- Ritardo NANDs

Ogni NAND è collegata a 1 inverter che dà  $C_{\text{carico}} = 2C_g$  (come prima data dai transistori in parallelo collegati all'ingresso).

Visto che stiamo parlando di una NAND a 4 ingressi (3 dati da A0, A1, A2 e uno dato dal segnale di EN), il caso peggiore si ha quando il segnale attraverso la NAND si propaga da uno stato alto a uno basso ( $H \rightarrow L$ ), in tal senso vanno considerati i 4 NMOS in serie e, pertanto, assumono  $R_{eq} = 4R_{ON}$ . E' inoltre necessario considerare tale stato affinchè dopo l'inverter 2 commuti da  $L \rightarrow H$ .

La rappresentazione è analoga a quelle già viste con l'unica differenza che essendo una porta NAND a 4 ingressi avrà 4 nMOS collegati in serie a GND che sono pilotati come detto dalle 3 linee di dato e dal segnale di enable (EN) che arriva dall'inverter 2 del decoder di banco.



Come dovrebbe essere chiaro le 4 resistenze che in serie danno  $Req=4Ron$  sono Q1, Q2, Q3 e Q4.

- Ritardo Inverter 2

Ogni Inverter 2 è collegato ad una wordline che è a sua volta collegata a 16 celle ognuna con  $C_{g, \text{cella}} = 0.1 \text{ fF}$  (dato del testo). Questo significa che la  $C_{\text{carico}} = 16C_{g, \text{cella}}$ .

La  $R_{eq}$  in questo caso deve tenere conto della resistenza dell'Inverter  $R_{ON}$  e della resistenza della wordline  $R_{WL} = 10 \Omega \cdot 16 \text{ celle} = 160 \Omega$  (dato del testo). Queste sono collegate in serie e ci danno quindi una  $R_{eq} = R_{ON} + R_{WL}$ .

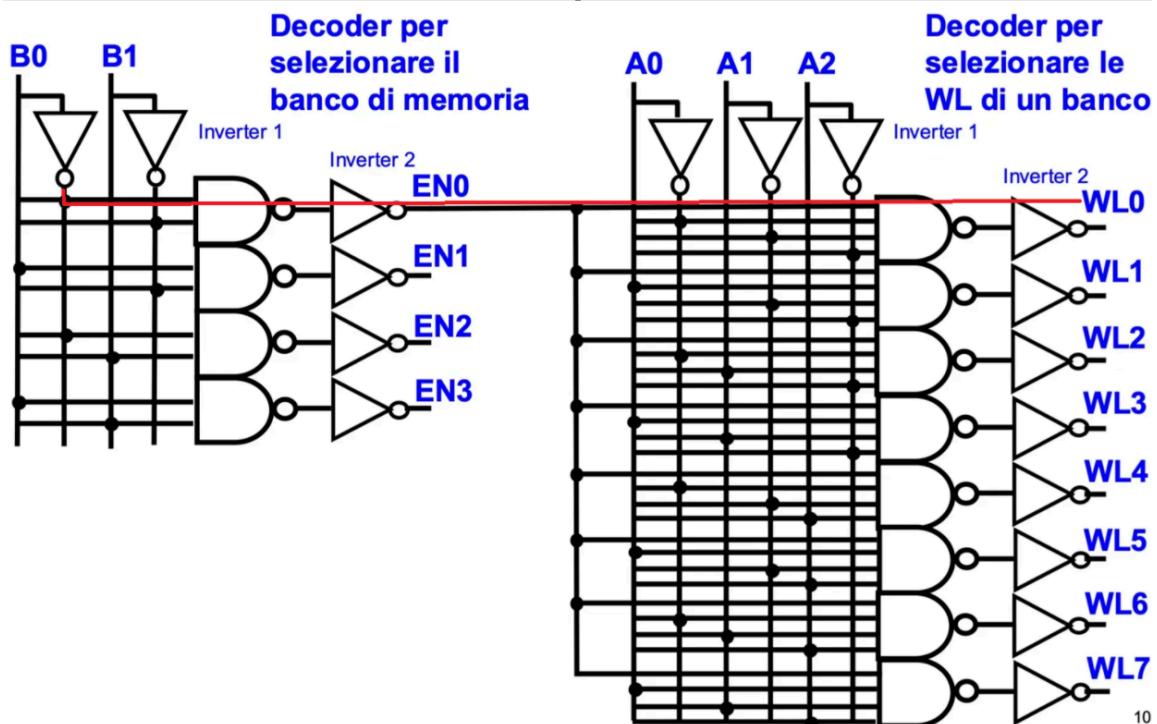
Di seguito sono riportate formule e valori di quanto detto fin ora a parole

- **Inverter 1:**  $0,69 \cdot (R_{ON} \cdot 4 \cdot 2C_g) = 0,69 \cdot 100 \Omega \cdot 4 \cdot 2 \cdot 1 \text{ fF} = 0,55 \text{ ps}$ 
  - Fattore "4": l'output di ciascun Inverter 1 è collegato a 4 inputs porta NAND
- **NAND:**  $0,69 \cdot (4R_{ON} \cdot 2C_g) = 0,69 \cdot (4 \cdot 100 \Omega \cdot 2 \cdot 1 \text{ fF}) = 0,55 \text{ ps}$
- **Inverter 2:**  $0,69 \cdot (R_{ON} + R_{WL}) \cdot C_{WL}$   
 $= 0,69 \cdot (100 \Omega + 16 \cdot 10 \Omega) \cdot 16 \cdot 0,1 \text{ fF} = 0,29 \text{ ps}$

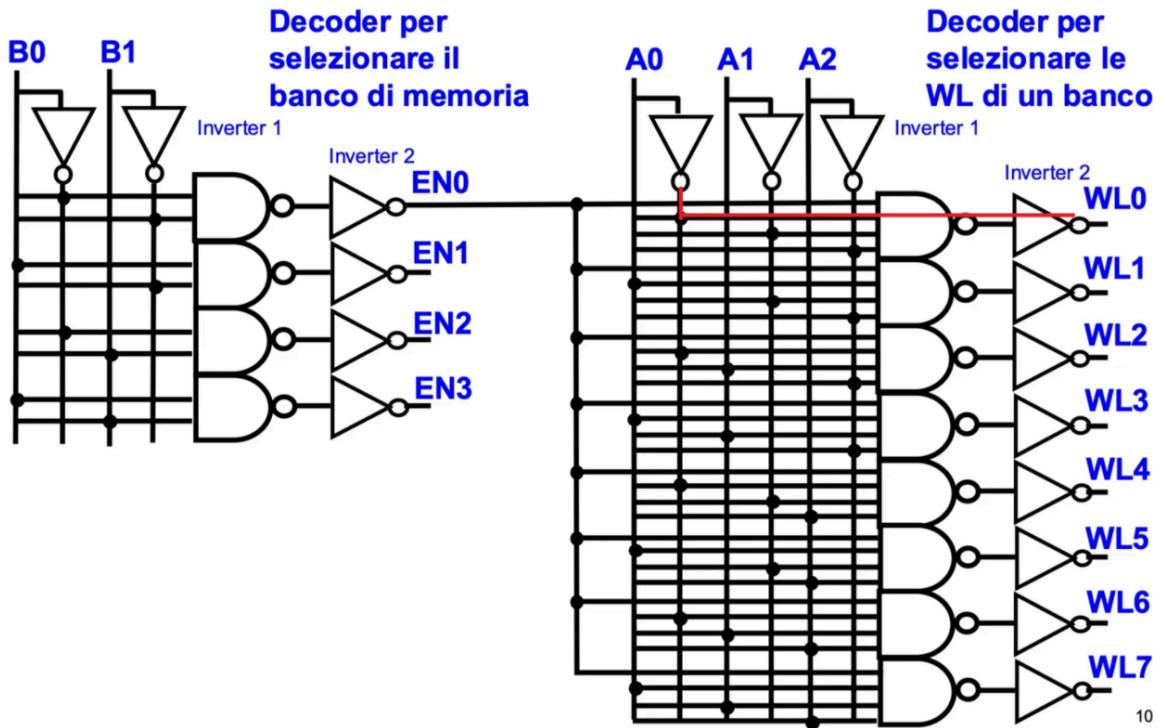
## ▼ Ritardo totale

Per il calcolo del ritardo totale possiamo considerare 2 percorsi possibili:

- $B \rightarrow EN \rightarrow WL$



- $A \rightarrow WL$



Questi due percorsi hanno in comune il ritardo della NAND ( $0.55 \text{ ps}$ ) e dell'Inverter 2 della wordline ( $0.29 \text{ ps}$ ).

Prima di arrivare alla NAND, il primo percorso accumula un ritardo massimo dato dalla somma di Inverter 1, NAND e Inverter 2 del banco ( $0.28 + 0.28 + 1.1$ ) mentre il percorso 2 ha il solo contributo dell'Inverter 1 della wordline ( $0.55$ ).

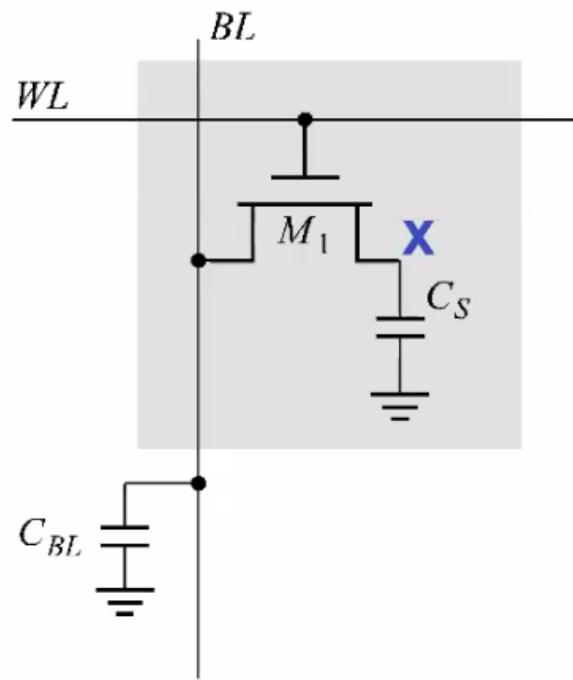
Dobbiamo prendere ovviamente il massimo tra questi due valori, ovvero come potevamo aspettarci quello del percorso che parte dal banco, e sommarlo ai ritardi di NAND e Inverter 2 della wordline.

$$\begin{aligned} & \max(0,28 \text{ ps} + 0,28 \text{ ps} + 1,1 \text{ ps}, 0,55 \text{ ps}) + 0,55 \text{ ps} + 0,29 \text{ ps} \\ & = (0,28 \text{ ps} + 0,28 \text{ ps} + 1,1 \text{ ps}) + 0,55 \text{ ps} + 0,29 \text{ ps} = 2,5 \text{ ps} \end{aligned}$$

## Esercizio 4 - Tensioni nella cella DRAM (teorico)

▼ Creatore originale @Gianbattista Busonera

Calcolare la variazione di tensione della bitline di una memoria *DRAM* alimentata con tensione  $V_{DD}$  e che ha capacità di storage  $C_S$  e capacità della bitline  $C_{BL}$ .



EA06 - Cella DRAM standard

## ▼ Soluzione

Si procede valutando inizialmente:

- Tensione  $V_{BL}$  iniziale:  $V_{pre}$ 
  - Normalmente la linea di bitline è precaricata ad una certa tensione, solitamente pari alla tensione di soglia  $V_T = \frac{V_{DD}}{2}$
- Tensione  $V_S$  iniziale:  $V_{bit}$ 
  - Tale valore consente di valutare se il condensatore  $C_S$  "contiene" un uno/zero logico:
    - $V_{bit} = V_{DD} \Rightarrow C_S$  contiene un uno logico
    - $V_{bit} = 0 \Rightarrow C_S$  contiene uno zero logico

Quando accendiamo il transistore  $M_1$  le due tensioni  $V_{BL}$  e  $V_S$  si egualgano come visibile in seguito:



Ripasso formule condensatori:

$$Q = cV \rightarrow \Delta Q = c\Delta V$$

Quando due condensatori  $C_1$  e  $C_2$  idealmente isolati tra loro vengono collegati tra loro:

- $\Delta Q_{tot} = 0$  (principio di conservazione della carica)
- $V_f = V_1 = V_2$

Quando si collegano  $C_S$  e  $C_{BL}$ , la carica totale si conserva, si passa dunque da:

1.  $Q_{tot,i} = Q_S + Q_{BL} = C_S V_{bit} + C_{BL} V_{pre}$
2.  $Q_{tot,f} = C_S V_S + C_{BL} V_{BL}$ 
  - a. Le due capacità  $C_S$  e  $C_{BL}$  si trovano però alla stessa tensione  $V_f$  dopo la redistribuzione delle cariche, dunque:  $Q_{tot,f} = (C_S + C_{BL})V_f$
3. Imponendo  $\Delta Q_{tot} = 0$  si ottiene  $Q_{tot,f} = Q_{tot,i}$  e dunque:
  - a.  $C_S V_{bit} + C_{BL} V_{pre} = (C_S + C_{BL})V_f$
  - b. Si ricava dunque  $V_f = \frac{C_S V_{bit} + C_{BL} V_{pre}}{C_S + C_{BL}}$
4. Si ricavano dunque dunque:
  - a.  $\Delta V_{BL} = V_f - V_{pre} = \frac{C_S V_{bit} + C_{BL} V_{pre}}{C_S + C_{BL}} - V_{pre} = \frac{C_S (V_{bit} - V_{pre})}{C_S + C_{BL}}$
  - b.  $\Delta V_S = V_f - V_{bit} = \frac{C_S V_{bit} + C_{BL} V_{pre}}{C_S + C_{BL}} - V_{bit} = \frac{C_{BL} (V_{pre} - V_{bit})}{C_S + C_{BL}}$



Si ricavano dunque due formule importanti per gli esercizi successivi:

$$\Delta V_{BL} = (V_{bit} - V_{pre}) \frac{C_S}{C_S + C_{BL}}$$

$$\Delta V_S = (V_{pre} - V_{bit}) \frac{C_{BL}}{C_S + C_{BL}}$$

P.S. La capacità  $C_{BL}$ , cioè la capacità vista dalla bitline è approssimabile con la capacità equivalente di tutti gli M transistor con capacità  $C_{Drain}$  e, pertanto:

$$C_{BL} \simeq M \cdot C_{Drain}$$

Distinguiamo i due casi:

Bit memorizzato in $C_S$	$V_{bit}$	$\Delta V_{BL}$	Dopo il "sense amplifier"
0	0	$-V_{pre} \frac{C_S}{C_S + C_{BL}} < 0$	$\Delta V_{BL} = 0 = 0$ logico
1	$V_{DD} - V_{th}$	$(V_{DD} - V_{th} - V_{pre}) \frac{C_S}{C_S + C_{BL}} > 0$	$\Delta V_{BL} = V_{DD} = 1$ logico

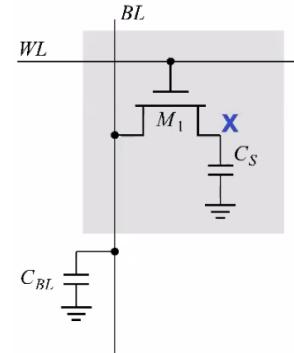
La capacità di storage  $C_S$  è solitamente molto piccola mentre  $C_{BL}$  molto grande  $\Rightarrow$  La variazione  $\Delta V_{BL}$  è una variazione piccola di tensione e, quindi, non è sufficiente per interpretare un uno/zero logico. E' necessario introdurre un cosiddetto "sense amplifier" il quale amplifica la tensione portandola a zero se negativa o a  $V_{DD}$  se positiva.

## Esercizio 5 - Lettura in DRAM

▼ Creatore originale: @Gianbattista Busonera

Una cella DRAM ha:

- $C_S = 20 \text{ fF}$
- $M = 512$  celle collegate alla bitline. (Si noti come una cella è formata da un transistor  $M_i$  e una cella di storage  $C_S$  come visibile nell'evidenziazione grigia in figura).
- Pass transistor  $M_i$  con:
  - $C_{Dra\in} = C_D = 0.1 \text{ fF}$
  - Tensione di soglia  $V_{th} = 0.1V$
- Tensione di alimentazione  $V_{DD} = 1V$



Visto che ci sono 512 celle collegate alla bitline, considereremo la capacità equivalente vista dalla bitline:

$$C_{BL} = M \cdot C_{Dra\in} = 512 \cdot 0.1 \text{ fF} = 51.2 \text{ fF}$$

Visto che ci sono  $M$  celle, ci sono anche  $M$  condensatori di Drain di ogni transistor (i quali sono collegati in parallelo alla bitline).

## Obiettivo 1 - Variazione di tensione a seguito di una lettura di un 1

Determinare la variazione di tensione sulla bitline  $\Delta V_{BL}$  e sul condensatore di storage  $\Delta V_S$  a seguito di una lettura di un 1 logico memorizzato nella cella di storage  $C_S$ .

### ▼ Calcolo variazione di tensione di bitline

Facciamo riferimento alla formula trovata nell'esercizio 4:



$$\Delta V_{BL} = (V_{bit} - V_{pre}) \frac{C_S}{C_S + C_{BL}}$$

In questo caso, ipotizzando di leggere uno 0 logico si ha che:

- $V_{bit} = V_{DD} - V_{th} = 1 - 0.1 = 0.9V$  (avendo ipotizzato un 1 logico)

- $V_{pre} = \frac{V_{DD}}{2} = \frac{1}{2} = 0.5V$

Sostituendo nella formula:

$$\Delta V_{BL} = (0.9 - 0.5) \frac{C_S}{C_S + 512C_{D_{rain}}} = 0.4 \frac{20}{71.2} = 112 \text{ mV}$$

### ▼ Calcolo variazione di tensione di storage

Facciamo riferimento alla formula trovata nell'esercizio 4:



$$\Delta V_S = (V_{pre} - V_{bit}) \frac{C_{BL}}{C_S + C_{BL}}$$

In questo caso, ipotizzando di leggere uno 0 logico si ha che:

- $V_{bit} = V_{DD} - V_{th} = 1 - 0.1 = 0.9V$  (avendo ipotizzato un 1 logico)
- $V_{pre} = \frac{V_{DD}}{2} = \frac{1}{2} = 0.5V$

Sostituendo nella formula:

$$\Delta V_S = (0.5 - 0.9) \frac{51.2}{20 + 51.2} = -0.4 \frac{51.2}{71.2} = -288 \text{ mV}$$



In seguito a una lettura mi piacerebbe non modificare il contenuto della cella! Notiamo però che  $\Delta V_S$  si degrada ( $-288 \text{ mV}$ )

Dopo la lettura di un uno logico, il sense amplifier amplifica  $\Delta V_{BL}$ , la porta su alla tensione di alimentazione  $V_{DD}$  e riscrive nella cella  $C_S$  il valore appena letto.

## Obiettivo 2 - Variazione di tensione a seguito di una lettura di uno 0

Determinare la variazione di tensione sulla bitline  $\Delta V_{BL}$  e sul condensatore di storage  $\Delta V_S$  a seguito di una lettura di uno 0 logico memorizzato nella cella di storage  $C_S$ .

### ▼ Calcolo variazione di tensione di bitline

Facciamo riferimento alla formula trovata nell'esercizio 4:



$$\Delta V_{BL} = (V_{bit} - V_{pre}) \frac{C_S}{C_S + C_{BL}}$$

In questo caso, ipotizzando di leggere uno 0 logico si ha che:

- $V_{bit} = 0V$  (avendo ipotizzato uno 0 logico)
- $V_{pre} = \frac{V_{DD}}{2} = \frac{1}{2} = 0.5V$

Sostituendo nella formula:

$$\Delta V_{BL} = (0 - 0.5) \frac{C_S}{C_S + 512C_{Dra\text{in}}} = -0.5 \frac{20}{20 + 51.2} = -0.5 \frac{20}{71.2} = -140 mV$$

### ▼ Calcolo variazione di tensione di storage

Facciamo riferimento alla formula trovata nell'esercizio 4:



$$\Delta V_S = (V_{pre} - V_{bit}) \frac{C_{BL}}{C_S + C_{BL}}$$

In questo caso, ipotizzando di leggere uno 0 logico si ha che:

- $V_{bit} = 0V$  (avendo ipotizzato uno 0 logico)
- $V_{pre} = \frac{V_{DD}}{2} = \frac{1}{2} = 0.5V$

Sostituendo nella formula:

$$\Delta V_S = (0.5 - 0) \frac{51.2}{20 + 51.2} = 0.5 \frac{51.2}{71.2} = 360 mV$$



In seguito a una lettura mi piacerebbe non modificare il contenuto della cella! Notiamo però che  $\Delta V_S$  aumenta ( $360 \text{ mV}$ )

Dopo la lettura di uno zero logico, il sense amplifier porta a zero  $\Delta V_{BL}$  e riscrive nella cella  $C_S$  il valore appena letto (0 logico).

## Obiettivo 3 - Massimo numero M di celle connesse

Nel caso in cui la **sensibilità** del Sense Amplifier fosse di  $50\text{mV}$  (nel caso in cui la variazione fosse inferiore il sense amplifier non riuscirebbe a identificare se  $\Delta V_{BL}$  corrisponda a un uno o uno zero logico), **determinare il massimo numero M di celle connesse alla bitline.**

### ▼ Soluzione

L'incognita è in questo caso il numero massimo di celle connesse alla bitline.

Quello che vogliamo è che  $|\Delta V_{BL}| \geq 50 \text{ mV}$  così che il sense amplifier si accorga di variazioni.

Nota la formula:

$$\Delta V_{BL} = (V_{bit} - V_{pre}) \frac{C_S}{C_S + C_{BL}}$$

Esplicitando  $C_{BL} = M \cdot C_{D_{rain}}$ :

$$\Delta V_{BL} = (V_{bit} - V_{pre}) \frac{C_S}{C_S + M \cdot C_{D_{rain}}}$$

Risolviamo, per semplicità, la disequazione senza valore assoluto:

$$(V_{bit} - V_{pre}) \frac{C_S}{C_S + M \cdot C_{D_{rain}}} \geq 50\text{mV}$$

$$\frac{C_S(V_{bit} - V_{pre}) - 0.05(C_S + M \cdot C_{D_{rain}})}{C_S + M \cdot C_{D_{rain}}} \geq 0$$

$$C_S(V_{bit} - V_{pre}) - 0.05C_S \geq 0.05M \cdot C_{D_{rain}}$$

$$\frac{C_S(V_{bit} - V_{pre}) - 0.05C_S}{0.05C_{Dra\text{in}}} \geq M$$

$$M \leq \frac{C_S}{C_{Dra\text{in}}} \left( \frac{V_{bit} - V_{pre}}{0.05} - 1 \right)$$

Bisognerebbe distinguere chiaramente il caso “peggiore” di lettura tra 0 e 1 logico. Bisogna dunque osservare le variazioni di  $\Delta V_{BL}$  in entrambi i casi. Dai punti precedenti:

- Nel caso di 1 logico:  $\Delta V_{BL} = 112 \text{ mV}$
- Nel caso di 0 logico:  $\Delta V_{BL} = -140 \text{ mV}$

Visibilmente, la lettura di un uno logico lascia un margine minore per il sense amplifier, dunque considereremo come caso peggiore la lettura di un uno logico.

Otteniamo dunque:

$$M \leq \frac{20}{0.1} \left( \frac{0.9 - 0.5}{0.05} - 1 \right) = 1400 \text{ celle}$$

Da cui  $M \leq 1400$  celle.

## Esercizio 6 - Rinfresco celle DRAM (todo)

▼ Creatore originale: @Gianbattista Busonera

La corrente di leakage nei pass transistor delle celle di una DRAM è  $I_{leak} = 25 \text{ fA}$  e la capacità di storage è  $C_S = 20 \text{ fF}$ .

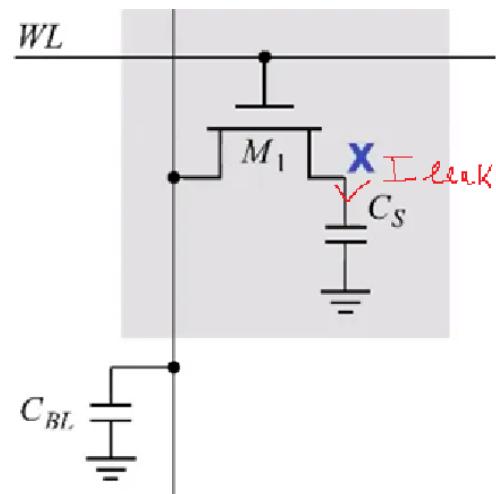
Determinare il periodo di refresh per garantire una variazione massima della tensione del condensatore di storage di  $0.1V$ , cioè  $|\Delta V_S| < 0.1V$

▼ Introduzione

Il transistore di una cella, quando non conduce, non è esattamente un circuito aperto!

In realtà fa passare la corrente di leakage che, a lungo andare, modifica la tensione sul condensatore di storage

$C_S$ . Pertanto è necessario, di tanto in tanto, fare una operazione di rinfresco in cui il valore di tutte le celle viene letto e successivamente riscritto.



Vogliamo trovare la condizione tale per cui il refresh viene fatto quando la tensione di storage sul condensatore varia di  $0.1V = 100mV$ .

### ▼ Svolgimento

Dall'equazione costitutiva di un condensatore  $C_S$ :

$$q = C_S V \rightarrow i(t) = C_S \frac{dV}{dt}$$

Se però la corrente di leakage è costante otteniamo  $I_{leak} = C_S \frac{\Delta V}{\Delta t}$ .

Esplicitiamo la variazione di tensione e imponiamola inferiore a 0.1V.

$$\Delta V = \frac{I_{leak} \Delta t}{C_S} \Rightarrow \frac{I_{leak} \Delta t}{C_S} \leq 0.1V$$

Da cui si ricava:

$$\Delta t = T_{refresh} \leq \frac{0.1}{I_{leak}} C_S = \frac{0.1V}{25fA} 20fF = 80 ms$$



Se aspettassi più di 80 ms, il condensatore di storage varierebbe la propria tensione di più di 0.1V.

## Esercizio 7 - Ritenzione in cella Flash

### ▼ Creatore originale: @Gianbattista Busonera

Una cella FLASH ha una capacità tra il gate di controllo e il gate flottante  $C_{PP} = 50 \text{ aF} = 50 \cdot 10^{-18} \text{ F}$ .

Se la massima variazione ammissibile della tensione di soglia per garantire la ritenzione del dato memorizzato per 10 anni nella cella è  $\Delta V_{th} = 1V$ , determinare la massima corrente di perdita  $I_{leak}$  tra il gate flottante e substrato (in ampere e in elettroni persi/settimana).

### ▼ Introduzione

Nelle celle FLASH, così come in generale in tutte le memorie non volatili, ci sono delle cariche immagazzinate nel gate flottante (floating gate).

Tali cariche, per quanto pensiamo siano intrappolate, hanno una piccola possibilità di scappare. Esiste infatti una piccola corrente di perdita tra il gate flottante e il substrato dovuta a imperfezioni varie...

Vogliamo stabilire qual è la massima corrente di perdita accettabile affinchè la variazione della quantità di carica contenuta dentro il gate flottante sia contenuta entro una certa soglia.

La presenza di queste cariche intrappolate nel gate flottante modifica la tensione di soglia (corrispettivo dell'informazione memorizzata nella cella).



Se ci sono cariche  $\Rightarrow$  tensione di soglia aumenta  $\Rightarrow$  1 logico

Se non ci sono cariche  $\Rightarrow$  tensione di soglia diminuisce  $\Rightarrow$  0 logico

Si vuole che, in un tempo molto lungo (10 anni), tale variazione di tensione di soglia sia di al massimo 1 V.

### ▼ Soluzione

Dall'equazione costitutiva di un condensatore  $C_{PP}$ :

$$Q = C_{PP}V \rightarrow \Delta Q = C_{PP}\Delta V$$

Di conseguenza, posto  $\Delta V_{th} = 1V$ , otteniamo che la variazione di carica sul condensatore  $C_{PP}$  è pari a  $\Delta Q = 50 \text{ aF} \cdot 1V = 50 \text{ aC}$

Ricaviamo il numero di elettroni con la formula:  $q = n \cdot e$

$$n_{elettroni} = \frac{\Delta Q}{e} = \frac{50 \text{ aC}}{1.6 \cdot 10^{-19} C} = \frac{50 \cdot 10}{1.6} = 313 \text{ elettroni}$$



Significa che al massimo ci si può permettere di perdere 313 elettroni per non far variare la tensione di soglia più di un volt.

Volendo ricavare la corrente  $I_{leak} = \frac{\Delta Q}{\Delta t}$ , cioè la quantità di carica che fluisce nell'intervallo temporale di 10 anni, basta risolvere l'equazione considerando come intervallo temporale:

$$\Delta t = 10 \text{ anni} = 10 \cdot 365 \cdot 24 \cdot 60 \cdot 60 \text{ secondi}$$

$$\text{Si ottiene } I_{leak} = \frac{\Delta Q}{\Delta t} = \frac{50 \cdot 10^{-18}}{10 \cdot 365 \cdot 24 \cdot 60 \cdot 60} = 1.6 \cdot 10^{-25} A$$

Volendo poi calcolare invece la quantità di elettroni persi per ogni settimana, consideriamo il numero di settimane totale in 10 anni:  $N_s = 10 \cdot 52 = 520$  settimane.

$$I_s = \frac{\text{elettroni totali}}{N_s} = \frac{313}{520} = 0.6 \frac{\text{elettroni}}{\text{settimana}}$$

## Esercizio 8 - Max numero di cicli P/E in celle Flash

▼ Creatore originale: @Gianbattista Busonera

Le celle di una memoria flash da  $4096 = 4 \cdot 1024 = 2^{12}$  blocchi possono essere programmate e cancellate al massimo per  $10^4$  volte.

In un ipotetico scenario di utilizzo, vengono continuamente eseguiti cicli di P/E (programmazione / erasing, cancellazione) su file di 50 blocchi ad un tasso medio di 1 file ogni 10 minuti. I file occupano al massimo 200 blocchi complessivi simultaneamente.

Determinare la **durata massima della flash** (in anni) nei due seguenti casi:

- No "wear levelling", ossia i file sono scritti sempre negli stessi 200 blocchi.
- "wear levelling", ossia i file vengono distribuiti uniformemente su tutti i 4096 blocchi.

## ▼ Introduzione generale

Quando si fanno delle operazioni di programmazione e cancellazione su una memoria non volatile, si alterano le caratteristiche di tale memoria. Dopo un certo numeri di cicli non è più affidabile fare cicli di programmazione e cancellazione sulla stessa cella.

Il "wear levelling" consiste nel distribuire gli accessi in maniera tale che si massimizzi la durata della memoria. Se facessimo accesso negli stessi blocchi finiremmo per renderli inutilizzabili dopo un certo numero di P/E.

## ▼ Ripassino sulle FLASH e contesto

- **Pagina** = unità minima di **lettura e scrittura** (es. 2–16 KB) e **unità minima di lettura e scrittura**
- **Blocco** = gruppo di pagine (es. 64–256 pagine) e **unità minima di cancellazione**
  - Prima di poter riscrivere una pagina è necessario cancellare tutto il blocco che la contiene
- **File** = entità logica che può estendersi su più blocchi

Nel nostro esercizio:

- La memoria non volatile FLASH ha 4096 blocchi (4096 set di pagine)
- Ogni file occupa 50 blocchi
- Al massimo possono esserci 200 blocchi "attivi" simultaneamente nel caso di assenza di wear levelling ⇒ al massimo 4 file attivi per volta.
  - Nel caso di wear levelling si possono invece utilizzare tutti i 4096 blocchi.

## Obiettivo 1 - Caso senza wear levelling

### ▼ Introduzione

Non appena le celle su cui sto facendo accesso arrivano a  $10^4 = 10000$  cicli, la memoria non è più affidabile.

Il testo ci dice che ogni 10 minuti viene scritto un nuovo file e che quindi ci troviamo in una situazione del tipo:

0-9 minuti	10-19 minuti	20-29 minuti	30-39 minuti
------------	--------------	--------------	--------------

Scrivo un file nei primi 50 blocchi	Scrivo un file nei secondi 50 blocchi	Scrivo un file nel terzo set da 50 blocchi	Scrivo un file nel quarto set da 50 blocchi
-------------------------------------	---------------------------------------	--	---

Al 40° minuto tenteremo nuovamente di scrivere ma ciò non sarà possibile perché dobbiamo prima cancellare! Notiamo dunque che riscriviamo sullo stesso blocco ogni 40 minuti!

### ▼ Soluzione

Mediamente accedo allo stesso file una volta ogni 4 accessi visto che i su 200 blocchi entrano 4 file (ognuno da 50 blocchi)  $\Rightarrow T_a = \text{periodo di accesso} = \frac{200}{50} \cdot T_{scrittura} = 4 \cdot 10 \text{ min.}$

Visto che mediamente, ogni 40 minuti, tento di scrivere sullo stesso set di blocchi (e quindi dovrei cancellare), impiegherò un tempo per fare  $N_{max} = 10000$  cicli di P/E su una cella pari a:

$$T = N_{max} \cdot T_a = 10000 \cdot 40 \text{ min} = 400000 \text{ minuti} = 278 \text{ giorni} < 1 \text{ anno}$$

## Obiettivo 2 - Caso con wear levelling

### ▼ Introduzione

Non appena le celle su cui sto facendo accesso arrivano a  $10^4 = 10000$  cicli, la memoria non è più affidabile.

Il testo ci dice che mediamente accediamo ad un file composto da 50 blocchi (ma con wear levelling di blocchi utilizzabili ne avrei 4096  $\Rightarrow 4096/50$  file) mediamente ogni 10 minuti.

### ▼ Soluzione

Si calcola il tempo necessario per un ciclo di P/E per lo stesso set di blocchi:

$$T_{accesso} = \frac{N_{blocchi\ tot}}{N_{blocchi\ per\ file}} \cdot T_{scrittura} = \frac{4096}{50} \cdot 10 \text{ min} = \frac{4096}{5} \text{ min}$$

Maiamente, dunque, scriviamo e cancelliamo (P/E) uno stesso set di blocchi (file) ogni 812,2 minuti.

Calcoliamo dunque il tempo necessario affinchè si effettuino 10000 cicli di P/E su uno stesso set di blocchi:

$$T = N_{max} \cdot T_{accesso} = 10000 \cdot \frac{4096}{5} \text{ min} = 8192000 \text{ min}$$

$$8192000 \text{ min} = \frac{8192999}{60 \cdot 24 \cdot 365} \simeq 15.59 \text{ anni}$$