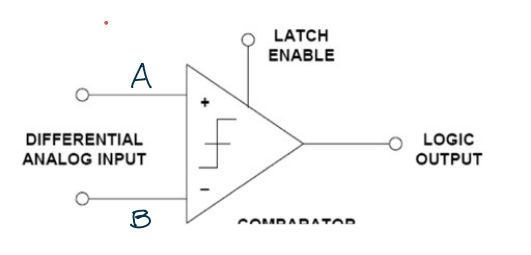
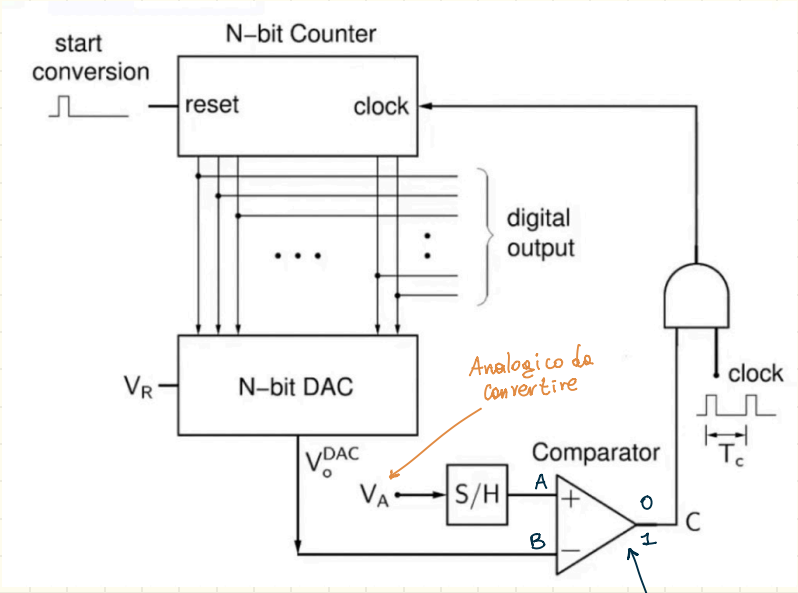
COMPARATORE



Il più semplice dei convertitori analogico digitale è un semplice **comparatore**: se B è maggiore di A l'uscita è zero, altrimenti 1.

In questo caso abbiamo 1 bit di conversione.

CONVERTITORE A CONTATORE



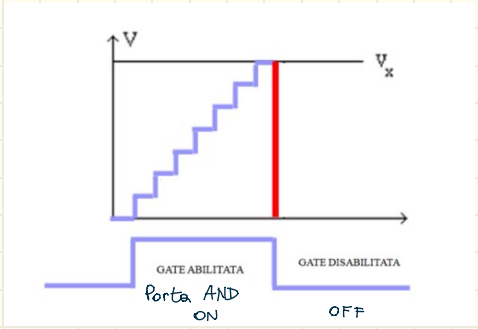
1 Se A≠B
0 Se A==B

Per questo convertitore **analogico -> digitale** utilizziamo due componenti chiave: un **contatore ad N bit** ed un **convertitore digitale -> analogico**.

La conversione avviene in questo modo:
Abbiamo un clock in ingresso al contatore che conta. L'uscita di questo contatore (stringa di N bit) viene data in ingresso al convertitore DA (che appunto riceve una word di bit com input). Questo convertitore **genera un valore analogico** che viene poi **confrontato con il valore da convertire da analogico a digitale** (tenuto costante da un sample & hold). Siccome il DAC ha come uscita una "scaletta", prima o poi questo segnale supererà quello di controllo (ovvero quello da convertire A->D) e quindi **il comparatore avrà come uscita 0**.

Quando il comparatore da come uscita 0, **l'AND** il cui secondo input è proprio il clock, **non lascia più passare alcun segnale**, e di conseguenza il contatore smette di contare.

Possiamo prelevare il valore in uscita al contatore che sarà proprio la stringa di bit corrispondente alla conversione analogico digitale.



Quando il valore del contatore supera quello del valore e analogico da convertire, il contatore smette di contare perché il comparatore ha in out 0 (e quindi l'and non fa più passare il clock)

#Domande esame

Operazione di conversione in successione
1) la prima operazione è quella di reset: quando abbiamo un valore alto sul pin di reset iniziamo la conversione, ed il contatore inizia a contare (il clock riprende perché il comparatore non segna più 0 ma 1, visto che uno degli input parte da zero, ovvero l'output del contatore).
2) il periodo dei gradini Tc è costante e corrisponde proprio al periodo del clock.
3) quando il counter supera la soglia (valore da convertire) il clock non passa più (per via dell'AND).
4) **L'output del DAC è incrementata da 0 ad una tensione VFS (fondo scala) ovvero il massimo possibile**. I gradini hanno ampiezza (altezza) pari a Q (ovvero valore di LSB).

Contatore a b bit
=> Max numero di incrementi: 2^b
ed avviene per $V_x = V_{FS}$
↑ Tensione da convertire

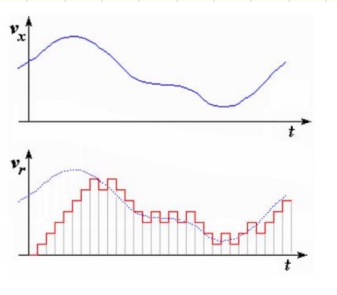
FREQ DI CAMPIONAMENTO MASSIMA

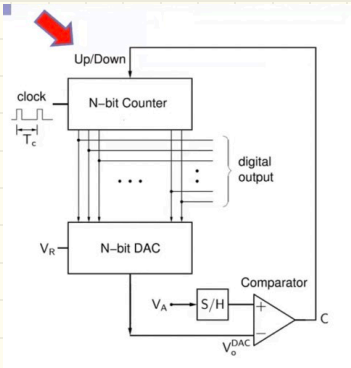
$$F_S = \frac{f_{clock}}{2^b}$$

Questo perché per ogni conversione A->D dobbiamo scorrere potenzialmente **tutti i gradini** del convertitore (ovvero 2^b). Maggiore è la frequenza di clock, e meno tempo ci si mette a "controllare" tutti i gradini.

Il tempo di conversione **è funzione del valore posto in ingresso**.

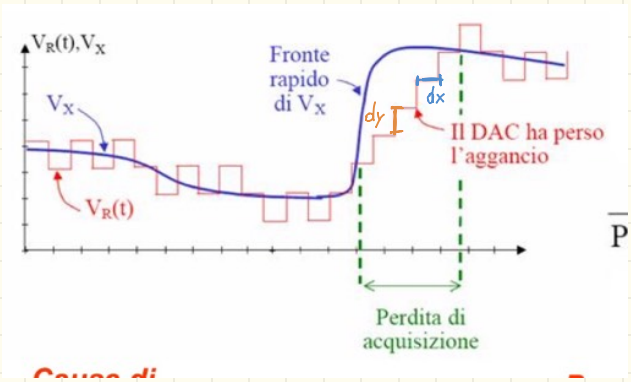
CONVERTITORE AD INSEGUIMENTO – UP-DOWN CONVERTER





In questa configurazione scompare l'AND del comparatore con il clock, ma l'uscita del comparatore entra in un **contatore up/down** proprio come pin up/down, che **decide se il contatore deve contare verso l'alto o verso il basso**.

Questo convertitore **è estremamente veloce** nelle conversioni di **tendono ad essere costanti nel tempo**, come ad esempio la temperatura, che non subisce forti variazioni.



$$\text{Pendenza} = \frac{dy}{dx} = \frac{1\,LSB}{\text{periodo di CLK}} = 1\,LSB \cdot f_{CLK}$$

$$\frac{1}{T} = f_{CLK}$$

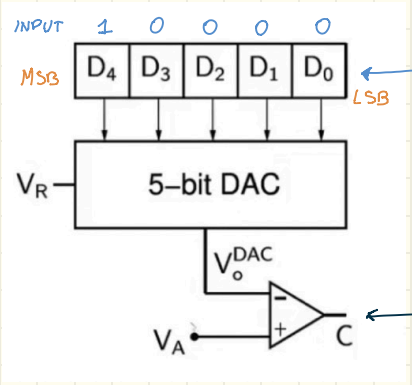
$$\text{ma} \quad 1\,LSB = \frac{V_{FS}}{2^b} \Rightarrow \boxed{\text{Pendenza} = \frac{V_{FS}}{2^b} \cdot f_{CLK}}$$

PENDENZA DELLA "SCALA"

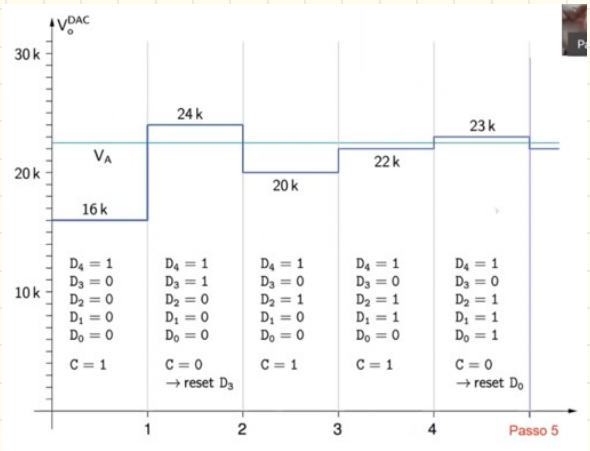
Se la pendenza del segnale è maggiore del valore calcolato sopra, allora abbiamo la **perdita del tracking** del segnale. Per risolvere questo problema **si pone un filtro passa basso** (le alte frequenze sono responsabili del repentino cambio di valore del segnale). Ovviamente **abbiamo perdita di definizione.**

CONVERTITORE AD APPROSSIMAZIONI SUCCESSIVE SAR

* RECAP



Se IN ha 1 al MSB e tutti zeri, abbiamo in out ½ V_{FS}. (Lez prec)



Con questo convertitore facciamo una sorta di *ricerca binaria*. Tenendo a mente che il MSB vale la metà della tensione di fondo scala Vfs, possiamo procedere in questo modo:

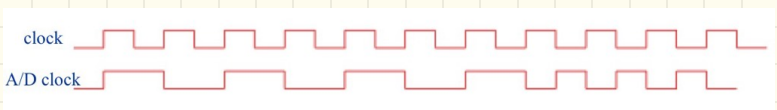
1) pongo alto il primo MSB, ottengo quindi un valore (che su un FS=32k) in uscita pari a 32k/2 = 16k. Lo confronto con il valore di riferimento e mi accorgo che è **minore**, quindi…
2) pongo alto anche il secondo MSB, che vale 1/4; ottengo 24k > 22.5k quindi…
3) riporto a zero il secondo MSB e porto ad uno il terzo MSB. Ottengo 20k < 22.5k quindi…
4) pongo ad 1 anche il quarto MSB, ottengo 22k<22.5k quindi…
5) pongo anche l'ultimo MSB (cioè il LSB) alto, ottengo 23k>22.5k quindi devo tornare alla precedente configurazione, ovvero: **10110**.

In sintesi
A seguito del confronto (comparatore) possiamo aggiungere un bit via via meno significativo, in modo da arrivare sempre più vicini al valore analogico. Quando aggiungiamo un bit ed otteniamo un valore più alto del valore analogico, **sappiamo per certo che quel bit non ci serve**, perché supera il valore di confronto. Siccome più ci avviciniamo al LSB e meno questo "vale", avvicinandoci al LSB riusciamo via via ad ottenere un valore quanto più vicino possibile a quello analogico.

Incertezza
I due valori di "errore" **saranno sempre all'interno della fascia di incertezza dell'LSB** (più di LSB non si riesce a sbagliare).

Per via di come è strutturato questo convertitore, riusciamo ad effettuare la conversione SEMPRE IN N CICLI DI CLOCK. In questo caso avevamo 5 bit ed infatti abbiamo impiegato 5 passaggi.

Questo tipo di convertitore **è molto versatile** e sopratutto veloce, quindi utilizzato in un'ampia gamma di applicazioni.



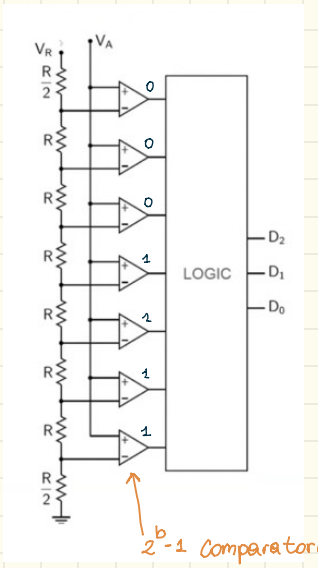
Il clock riportato a sinistra è un accorgimento usato per **evitare impulsi in uscita**: i primi bit generano un cambiamento repentino nell'uscita, e quindi potrebbero avvenire degli impulsi. Per evitare ciò basta utilizzare un clock "piu lento" all'inizio, cioè in concomitanza dei campi più repentini.

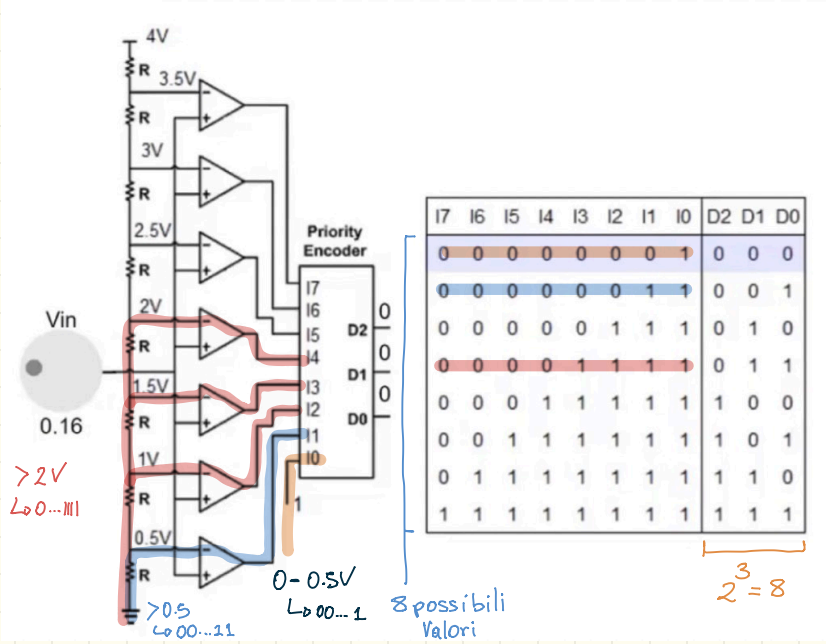
CONVERTITORI FLASH

I due convertitori visti prima sono dei convertitori che per operare **confrontano** il valore analogico con quello di un valore generato da un convertitore DAC. I convertitori flash, invece, **riescono a convertire il valore in input in un solo ciclo di clock**.

Questo è possibile perché invece di confrontare il singolo valore analogico N volte con N valori analogici, viene confrontato **contemporaneamente** con N valori analogici.

Funzionamento del convertitore
Abbiamo una lunga serie di resistori che formano dei partitori di tensione, che vanno a finire in (2^n)-1 comparatori **in parallelo**. Questi comparatori confrontano la tensione che gli arriva con la tensione da convertire. Il circuito logico riceve in input l'output dei convertitori e **codifica in 3 bit** (in questo caso) l'informazione.





I7	I6	I5	I4	I3	I2	I1	I0	D2	D1	D0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	1	0	0	1
0	0	0	0	0	1	1	1	0	1	0
0	0	0	0	1	1	1	1	0	1	1
0	0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1	1	1	1

Guardando la torre a sinistra, se dobbiamo convertire un valore analogico di 2.5V arriveremo fino ad I4 che corrisponde a 00001111, ed essendo il quarto elemento può essere codificato con il binario di 4, ovvero 011.

Capiamo quindi che possiamo avere 2^b valori, in questo caso 4 valori diversi.

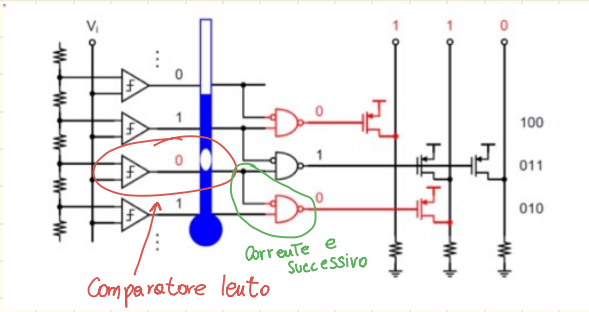
#ToDo Approfondim.

52.00 Architetture ADC

PROBLEMI DEI CONVERTITORI FLASH

Nella realtà i comparatori **non hanno tutti le stesse caratteristiche,** e potrebbe accadere che uno risulta essere **più lento**. Questo genera delle **"bolle"**, ovvero un bit che doveva andare ad 1, non ce la fa in tempo, e quindi abbiamo una stringa del tipo 00001**0**11 invece di 00001**1**11. Di conseguenza la parte logica che dovrebbe convertire la word di bit da 8 a 3 bit (appena vista) **non riesce a capire dove finisce effettivamente la serie di 1**; nel senso che la stringa 00001011 (invece di 00001111) potrebbe essere interpretata come 00000011.

Risolvere il problema
Si creano dei sistemi a più ingressi: abbiamo una porta che confronta il bit corrente con quello successivo, in modo tale che se il corrente è zero ed il successivo è 1 (dopo uno zero non dovrebbero esserci altri uno) riusciamo a capire che c'è un problema.

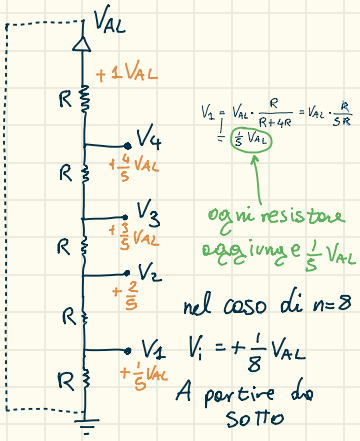


ENCODER

L'encoder è il responsabile di passare dalla word in uscita ai comparatori al codice binario finale, in questo caso di 3 bit.

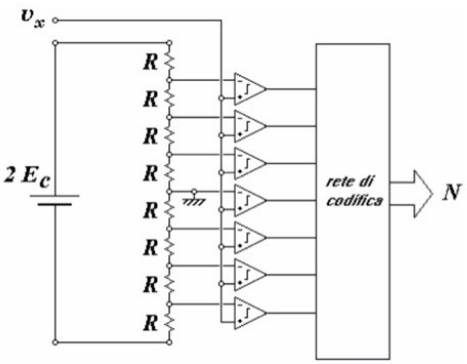
I7	I6	I5	I4	I3	I2	I1	I0	D2	D1	D0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	1	0	0	1
0	0	0	0	0	1	1	1	0	1	0
0	0	0	0	1	1	1	1	0	1	1
0	0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1	1	1	1

ENCODER

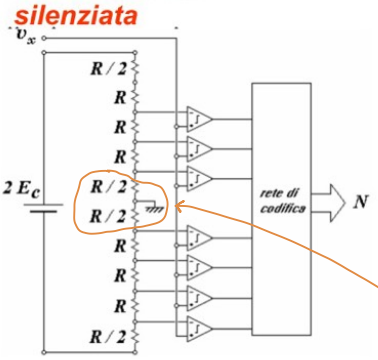


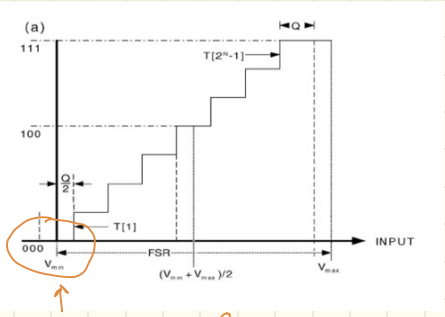
CARATTERISTICA SILENZIATA

Quantizzazione non Silenziata



Quantizzazione silenziata



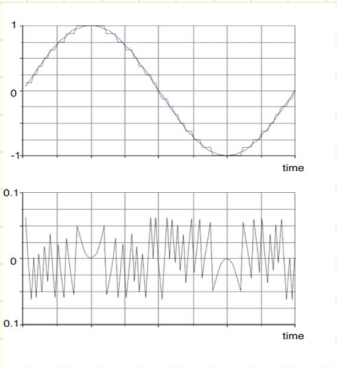


Zero a cavallo di un gradino

MORALE DELLA FAVOLA

Questi convertitori sono estremamente veloci, però abbiamo anche altri problemi.
- il primo problema è sicuramente **l'elevata quantità di componenti utilizzati**
- il secondo problema deriva dal primo: tutti i componenti dovrebbero funzionanre "idealmente" e **tutti in modo uguale**, ma questo è impossibile nel mondo reale.
- questo tipo di convertitore **non può avere una risoluzione elevata**: questo perché vuol dire avere **(2^N)-1 comparatori**; se volessimo fare un convertitore a 10 bit dovremmo avere 1023 comparatori ed altrettanti resistori! Peggio ancora, **dobbiamo avere una rete di codifica ad elevatissimo numero di ingressi**!

CONVERTITORI A SOVRACAMPIONAMENTO



ERRORE DI QUANTIZZAZIONE

Potenza errore di quant.

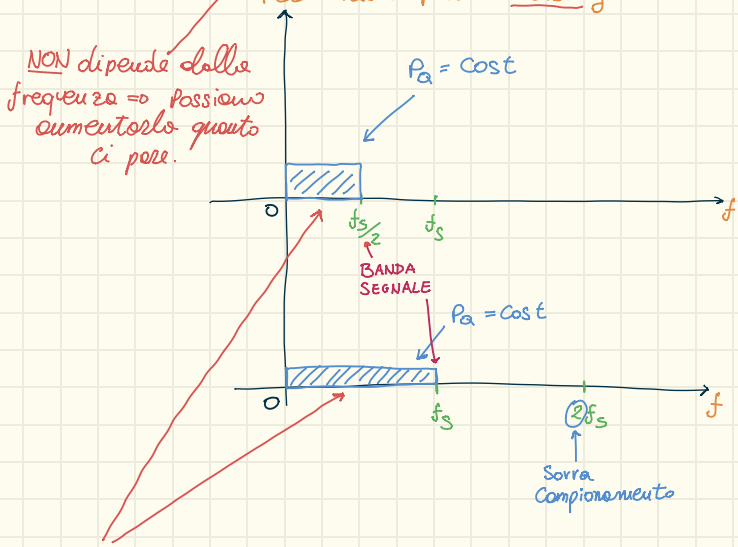
$$P_Q = \int_{-\infty}^{\infty} \varepsilon_Q p(\varepsilon_Q) d\varepsilon_Q = \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} \frac{\varepsilon_Q^2}{\Delta} d\varepsilon_Q = \frac{\Delta^2}{12}$$

Passo di quantizzazione

*esempio Bacinella

PSD - Power Spectral Density

NON dipende dalla frequenza => Possiamo aumentarlo quanto ci pare.



Siccome la potenza dell'errore di quantizzazione rimane **costante** a parità di numero di bit (livelli di quantizzazione) possiamo **distribuire** questo errore lungo un **intervallo più ampio** (ad esempio 2fs, frequenza di sampling-campionamento) in modo da **migliorare il rapporto segnale rumore.**

Abbiamo distribuito la potenza dell'errore di campionamento solo **fino ad Fs/2 perché la frequenza di campionamento deve essere pari ad almeno il doppio della frequenza del segnale** (da Nyquist)

Il problema di questo metodo è che **abbiamo una grande quantità di dati** (che non ci serve). Di conseguenza a valle dobbiamo avere una memoria abbastanza capiente. Solitamente una parte dei dati **viene scartata**.

Per effettuare il sovracampionamento è preferibile lavorare con **segnali di per se a bassa frequenza**, in modo da non dover campionare a frequenze estremamente alte.