Appunti IoT Systems

Capitolo 3

Digital and Analog I/O

Simone Coco

Digital and Analog I/O

Il microcontrollore deve interagire con il mondo esterno e, per farlo, utilizza i PIN di I/O. All'interno del microcontrollore vi sono Pin di I/O ci sono dei Pin che possono essere utilizzati nell'ambito digitale, dei pin che possono essere impiegati nell'ambito analogico e altri ancora che possono essere impiegati in entrambi.

Digital Input

Quando un Pin ha in ingresso un segnale digitale significa che l'informazione d'interesse può assumere solo due valori: 0 e 1. Generalmente, si hanno dagli 8 ai 32 pin di I/O

Le comunicazioni sui pin avvengono mediante operazioni in cui questi pin vengono considerati a **gruppi di 8** alla volta poiché ognuno di essi trasporta un bit (1 o 0) che, se raggruppati, formano 1 byte (8 bit = 1 byte), quindi una singola istruzione ad alto livello comporta, a basso livello, il settaggio simultaneo di 8 pin fisicamente distinti.

I pin possono essere:

- Pin di Input
- Pin di Output
- Pin Bidirezionali

Come abbiamo anticipato, alcuni pin hanno delle **funzioni accessorie** che permettono il loro utilizzo non solo come pin "trasportatore" di bit per la comunicazione ma come **pin addetti a determinate funzioni**: alcuni di essi, ad esempio, all'occorrenza possono essere utilizzati come pin analogici, altri possono fungere da timer, etc…
Tale funzionalità possono essere assegnate dinamicamente dal programmatore.

Consideriamo quindi un gruppo di pin (per convenzione 8); ogni **gruppo di pin** è caratterizzato da **3 informazioni (3 registri)**:

1. **Data Direction Register (DDR):** Stabilisce se il pin deve essere utilizzato se **in input** o **in output,** il valore può essere chiaramente 1 o 0, in base alla convenzione attribuita al microcontrollore questo valore può assumere quello di input o di output (esempio: 1 per input e 0 per output).
   Dopo un reset, i bit DDR sono genericamente inizializzati in input.

2. **Port Register (PORT):** qualora il pin venga presentato in output, il port register contiene il bit che il microcontrollore deve presentare all'esterno; il pin deve essere per forza contenuto da qualche parte, viene contenuto da questo registro fino a quando non viene ceduto in output. Quando il pin è configurato invece in input nel PORT c'è ugualmente una

copia del dato, anche se quest'ultimo sarà mandato in input piuttosto che in output (come verifica).

3. **Port Input Register (PIN):** se invece il pin viene presentato in input, il dato che deve essere passato all'interno del microcontrollore deve essere localizzato all'interno di questo registro fin quando non viene ceduto. Quando il pin è configurato invece in output in questo registro c'è ugualmente una copia del dato, anche se quest'ultimo sarà mandato in output piuttosto che in input (come controverifica).

Quindi il pin misura i dati e li "salva" nei registri che ha a disposizione.

Problema:

Nei pin digitali le uniche informazioni che si trasportano sono di natura binaria: Alto/Basso -> 1/0. Il valore che si associa tra 0 e 1 dipende dal microcontrollore, e non vi è un intermezzo

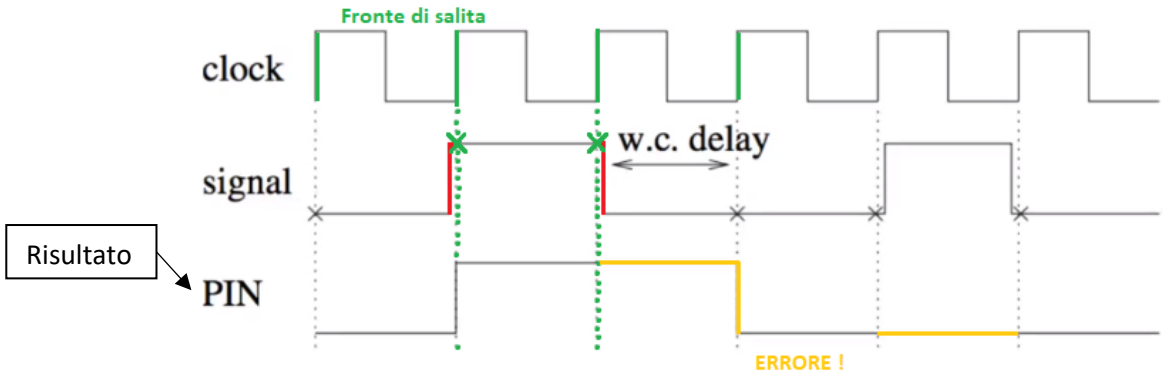
Facciamo un esempio:

AtMega16 è un microcontrollore che ha una tensione che varia da 4.5 a 5.5V considera basso un input che arriva in una percentuale del 20% in giù della Vcc (tensione ai suoi capi che vale in media 5V) e considera alto un input che va dal 60% in più della Vcc.
Resta scoperta un range che va dal 20% al 60%, in cui il valore non è né alto né basso e questo causerà dei potenziali problemi.

Campionamento digitale

Il pin Digitale effettua delle misurazioni in certi istanti precisi, comandati dalla frequenza di clock, quindi la misurazione non è continua. Un sistema stabilisce in quale segnale di clock effettuare la misura e si ripete periodicamente, scandendo il campionamento. Tipicamente le azioni del segnale sono scandite dal clock e agiscono nel **tempo di salita** di quest'ultimo, ovvero quando il clock passa dal valore zero al valore uno.

Guardiamo in figura un esempio del funzionamento del sistema:



- Abbiamo il clock che comanda negli istanti di salita;
- Il segnale che è la realtà che viene misurata dal PIN, avente un proprio andamento;

❖ Il PIN che misura il segnale, nella figura è riportata la misurazione che esso effettua.

Inizialmente, al primo fronte di salita del clock, il segnale si trova basso ed effettivamente il pin misura un segnale basso, quindi la misura è fedele;

Dopo di che il segnale raggiunge il valore alto e lo raggiunge ancor prima che il clock raggiunga il proprio fronte di salita, tuttavia fortunatamente il risultato non è falsato poiché riusciamo ad avere la conseguenza nella misurazione nel PIN poiché il ritardo minimo tra la salita del segnale e il fronte di salita del clock non ha introdotto misurazioni falsate.

Nel terzo caso, poco prima che il segnale effettivo scendesse, il clock misura un valore alto e il PIN misura un valore diverso dal segnale effettivo, a causa di questo ritardo che prende il nome di **w.c. delay** abbiamo perso l'informazione: il segnale; infatti, come si può vedere, il segnale misurato dal PIN e il segnale effettivo differiscono.

Dovrò quindi aspettare il prossimo ciclo di clock per capire che il segnale in realtà era basso. Si veda che successivamente si ha un altro errore.
Questo processo è il campionamento del segnale, quindi abbiamo verificato che il campionamento del segnale può indurre ad errori sulla misurazione effettiva di quest'ultimo.

Problemi di meta-stabilità

A questa situazione, in cui già presentiamo dei problemi dovuti al campionamento scandito per ogni tempo di salita del clock, si aggiunga inoltre il fatto che il segnale in natura non è digitale, quindi tutti i segnali naturali sono analogici e solo successivamente ad una conversione essi appaiono in formato digitale.

Il segnale si presenta con una forma simile:

Il segnale è di tipo analogico (come il semplice segnale della tensione); vengono stabilite due soglie: Vhi (High) come valore alto e Vlo (Low) come valore basso. Da Vhi in su abbiamo il valore alto, da Vlo in giù abbiamo il valore basso e tra Vhi e Vlo abbiamo un valore intermedio, che non vale né 0 né 1.

Andiamo incontro ad un altro problema: **il problema della meta-stabilità** che nasce quando non si sa come interpretare i valori intermedi, che stanno tra la soglia dei valori alti e quella dei valori bassi.

A questo problema se ne lega un altro: non solo il valore resta nella zona intermedia ma, tra l'altro, non sa nemmeno **quanto tempo resti nella zona intermedia**, non possiamo quindi stimare

un tempo in cui esso sta in tale zona. Abbiamo un problema di momento intermedio del segnale che non è limitato a priori.

La meta stabilità sancisce un grande problema per i circuiti digitali, **infatti, quando in input ad un circuito digitale viene fornita tensione e tale tensione sta nella zona intermedia, il circuito ha una probabilità P di entrare e rimanere nello stato di meta-stabilità.** Il circuito potrebbe quindi oscillare in valori intermedi (che non valgono né 0 né 1) e il microcontrollore potrebbe non funzionare per un tempo indefinito.

NB: la meta-stabilità non dipende dalla frequenza di clock, qualsiasi sia la frequenza di clock c'è sempre una probabilità di finire in meta-stabilità.

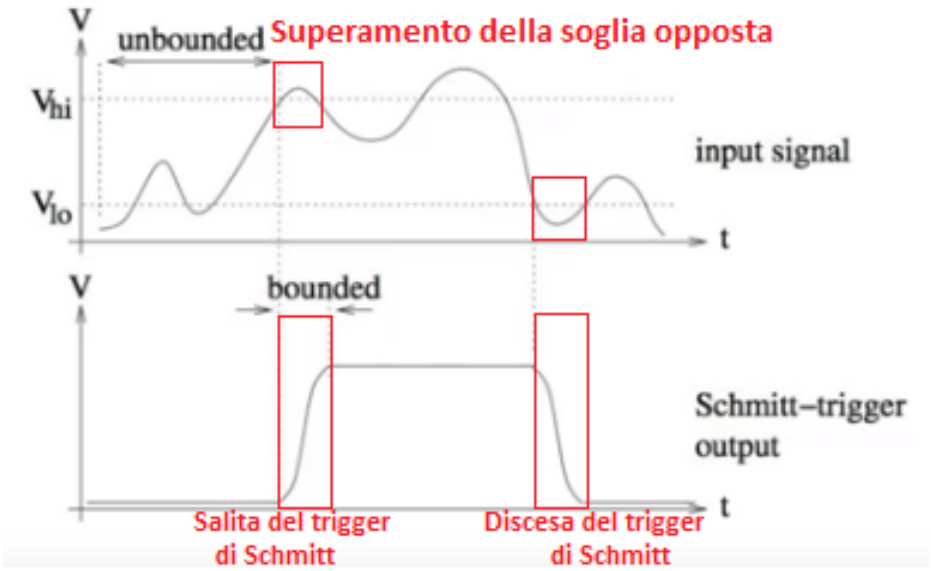
Normalmente si analizza la probabilità di finire in meta-stabilità, affermiamo dunque che:

La probabilità di meta-stabilità è la probabilità che, partendo dal campionamento di un segnale, esso si trovi in una zona intermedia cui nulla di può dire riguardo al valore assunto dal segnale.

Il trigger di Schmitt

Per minimizzare i problemi della meta-stabilità potremmo utilizzare il trigger di schmitt, un circuito che risolve parzialmente i problemi di meta-stabilità.

Esso ha due soglie: una per i valori alti e una per i valori bassi. In maniera deterministica switcha tra basso ad alto quando vengono superate queste soglie.



In questo esempio il trigger di Schmitt parte basso, come il segnale; dopo di che il segnale ha una piccola salita che lo fa uscire al di fuori della soglia Vlo, tuttavia, il trigger di Schmitt ignora questo minimo cambiamento.

Dopo di che superata la soglia alta Vhi, il trigger capta il superamento ti tale soglia e, seppur per poco tempo, in maniera deterministica inizia la salita, a prescindere dal comportamento del segnale effettivo.

A questo punto si mantiene alto fintanto che ci si trova nella soglia alta e, anche se il segnale supera tale soglia, il trigger non capta alcun cambiamento e rimane alto.

Esso rimane alto fin quando non capta il superamento della soglia opposta, ovvero della soglia Vlo, solo a quel punto allora fa iniziare la discesa, qualsiasi sia l'evoluzione del segnale

Il trigger di Schmitt si manterrà basso o alto fintanto che la soglia opposta al valore in cui si trova non viene superata, in tal caso si abbasserà o si alzerà.

[Fintanto che si trova nel valore basso, il trigger di Schmitt resta in tale valore anche se la soglia viene superata e si alza se e solo se il segnale supera la soglia alta (la soglia opposta). La stessa cosa vale al viceversa, passando all'alto al basso]

In questo modo si è risolto parzialmente il problema derivante dalla meta-stabilità: **si è risolto il problema dell'incertezza dei tempi di meta-stabilità**. Il trigger di Schmitt viene messo all'ingresso dei PIN.

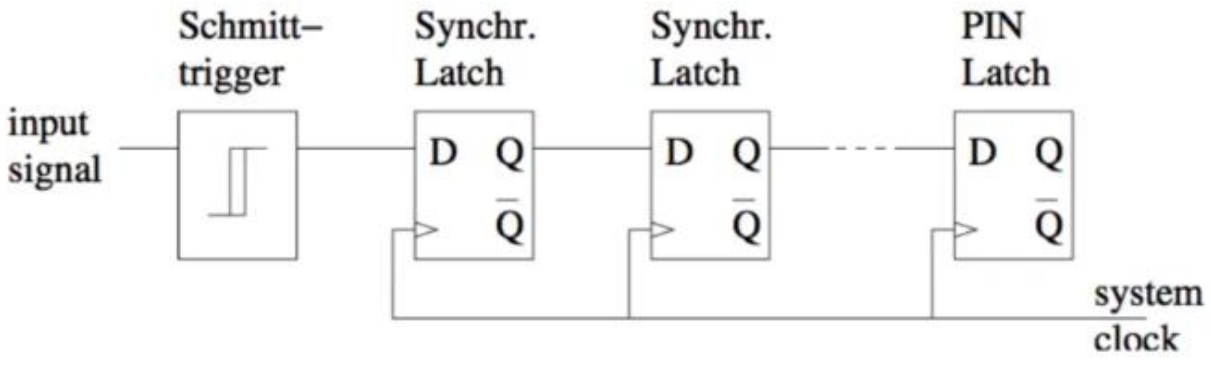
Problema:

Tuttavia, se durante la salita si ha un altro cambiamento di stato, potrei finire in una situazione metastabile. Vi è quindi una certa probabilità di finire nella meta-stabilità ma tale probabilità è diminuita con il trigger di Schmitt.

Il Sincronizzatore

Ovviamente ancora non basta, si deve rendere tale probabilità pressocché nulla, per evitare la condizione di meta-stabilità e quindi evitare conseguenze. Per questo motivo si utilizza il sincronizzatore

Abbiamo il segnale di input in cui viene posto il trigger di Schmitt e quest'ultimo ci garantisce che il tempo di regione intermedia non si protrae all'infinito. Dopo di che si hanno dei Latch che memorizzano i bit.



Il nostro scopo è di annullare la probabilità che si abbia la meta-stabilità; supponiamo di avere una percentuale del 10% di finire in condizione metastabile (cioè su un fronte di salita o discesa del trigger di Schmitt).

Il segnale entra nel trigger di Schmitt, questo lo fa diventare alto e basso con delle transazioni avente range limitato. Dopo di che si collegano una serie di latch, semplici memorizzatori di bit che

lo copiano a livello successivo, ognuno di essi essendo un circuito ha una probabilità che, ricevendo un valore intermedio, resti nella zona metastabile, nel nostro caso il 10% di possibilità.

Il 90% delle volte abbiamo dei valori non oscillanti senza meta-stabilità e quindi il segnale esce pulito e corretto dal trigger di Schmitt e da questo momento il segnale arriverà corretto a tutti i latch.

Se invece ci ritroviamo nel 10% di probabilità che dopo il trigger di Schmitt finiamo nella condizione di meta-stabilità, al primo latch potrebbe arrivare un valore che potrebbe essere zero o uno o un valore che sia anch'esso intermedio; analizziamo quest'ultimo caso, quello peggiore.

Questo ha il 90% di probabilità di non finire in meta-stabilità, quindi al caso peggiore abbiamo il 10% di probabilità di finire in meta-stabilità dopo il trigger di Schmitt e il 10% di probabilità che il primo latch cada in meta-stabilità.

Si parla quindi di **probabilità composta**. Questo significa che dal momento in cui uno dei latch assume un valore non intermedio, il trasferimento dei bit si "aggiusta" e non cade più in meta-stabilità; la probabilità è di $\boldsymbol{P^k}$ **dove k è il numero di latch**. In questo modo abbiamo isolato il segnale oscillante che creava problemi al nostro sistema.

Avendo, ad esempio, una probabilità di meta-stabilità del 10% già con 3 Latch (quindi per k= 3) Avremo una probabilità composta P = $0{,}1^3 = \frac{1}{1000}$ che è una probabilità molto bassa.

Ovviamente **più Latch si mettono e più ritardo di introduce** poiché ogni circuito introduce una certa latenza, quindi non possiamo mettere un numero troppo elevato di Latch per la risoluzione della meta-stabilità

Osservazione: il valore output del latch

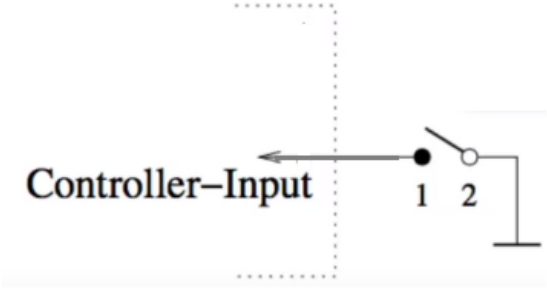
Quindi i latch ci permettono di non entrare in una condizione di meta-stabilità che porta il sistema a malfunzionamenti. Tuttavia, si noti un'importante osservazione:

Il latch quando ha in input un valore metastabile, stabilisce (nel caso migliore) un valore tra 0 e 1 ma questi valori potrebbero non essere esatti: se il segnale è alto ma il trigger di Schmitt dà in output un valore metastabile, può capitare che il latch interpretando questo segnale indefinito determini un valore basso, cadendo in errore. Quindi è importante segnalare che il latch garantisce di non finire in meta-stabilità ma non assicura nulla dal punto di vista del risultato finale, questo potrebbe anche essere errato.

Concludiamo con il dire che il sincronizzatore si usa per far sì che il segnale presentato in input al microprocessore non sia metastabile, garantendo che valori di quest'ultimo siano chiari: 0 o 1. Esso è formato da due parti: il trigger di Schmitt e una serie di Latch (tanti quanti bastano per garantire una probabilità di meta-stabilità pressocché nulla)

Pull-up resistors

Si immagini di avere un pin di input, come quello rappresentato in figura. Si immagini che a questo PIN di Input sia collegato un interruttore, in modo che se chiuso vale 0 altrimenti vale 1. Il tasto si chiude collegando due capi: 1 e 2; il capo 2 è il collegamento a massa e dal suo lato vede sempre il valore 0 mentre il capo 1 è collegato al pin in input. Chiudendo il tasto (e quindi il circuito) si ha un cortocircuito con la messa a terra e quindi leggeremo il valore 0.



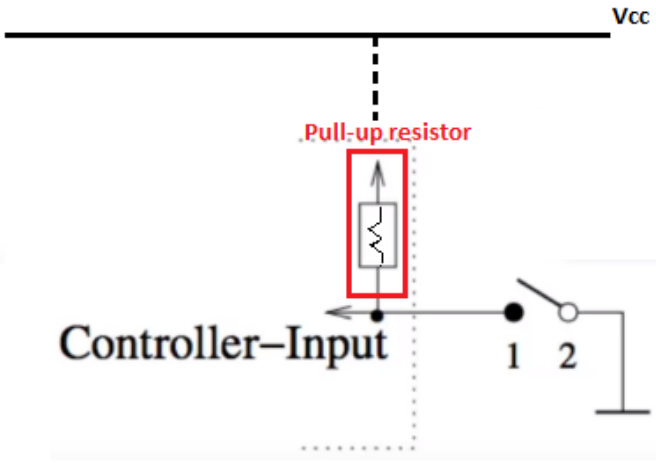
Immaginiamo però cosa accade se il tasto fosse aperto: se abbiamo indentificato la chiusura del tasto con il valore 0 allora l'apertura di esso è riconducibile al valore 1.

Ma ciò che accade è che se il tasto è chiuso il valore è 0 perché, come detto, è in cortocircuito ma quando è aperto il PIN non è collegato a nulla, poiché è staccato. Quindi in questo caso al microcontrollore non arriva l'input che intendevamo dare ma una serie di rumori e disturbi dovuti all'ambiente circostante. E tali valori possono creare alcuni problemi.

Quindi si deve trovare una soluzione per comunicare al microcontrollore che quando il tasto è aperto il valore che deve essere passato è alto. A tal proposito, nei PIN Input viene messo il **pull-up resistor.**

Il pull-up resistor è un resistore attaccata all'alimentazione Vcc. Vcc in digitale ha il significato di Alto.

Con il pull-up resistor quando il tasto è chiuso non cambia nulla, ma quando il tasto è aperto il resistore non è attraversato da corrente, e non essendo attraversato da corrente non si ha caduta di tensione e quindi esso ha un valore di Vcc, quindi il punto 1 si trova al valore di Vcc che in digitale vale "Alto": 1.



Il ruolo della resistenza è quasi celato in quanto ne sfruttiamo il vantaggio quando in esso non circola corrente, tuttavia, se non vi fosse la resistenza, un volta chiuso il tasto, si avrebbe una circuitazione che collega direttamente la Vcc a massa e questo porterebbe a danni irreversibili.

Il pull-up resistor evita il cortocircuito quando viene chiuso il circuito.

La resistenza deve avere un valore che garantisca un valore di corrente tale da poter essere assorbita dal microcontrollore ma questo riguarda delle specifiche tecniche non utili ai fini del corso.

Collegamenti:

1. Digital output

Il digital output si occupa del problema speculare: produrre un valore 0 o 1 in uscita. Tali valori dipendono dalle soglie del microcontrollore ma generalmente a 0 si associa il valore dell'informazione che va da 0 a Vlo (da zero alla soglia di valore minimo), a 1 si associa il valore che va da Vhi in su.

Le informazioni in output dipendono direttamente dal microcontrollore e, per questo motivo, assumiamo che esse abbiano minor criticità rispetto ai digital input, che dipendono dall'ambiente esterno.

2. Analog I/O

I segnali che percepiamo dal mondo sono tutti di natura digitale: assumono valori compresi tra una soglia di massimo e una soglia di minimo in modo continuo.

Spesso il microcontrollore allora si deve interfacciare con segnali in input analogici, opportunamente convertiti per essere interpretati e deve dare in output dei segnali altrettanto analogici per essere correttamente interpretati dal mondo esterno: spesso non basta una informazione binaria, serve un'informazione che assuma valori diversi. Basti pensare all'illuminazione di un led: spesso non basta "acceso" e "spento" ma servono differenti intensità di luminosità.

Analog I/O

Come abbiamo detto, nella realtà non esistono segnali di natura digitale: il mondo con cui si interfaccia il microcontrollore comprende dei segnali analogici, provenienti da fenomeni fisici; quindi spesso gli input che il microcontrollore deve gestire sono **input analogici**.

Lo stesso discorso vale per l'output: molte volte il microcontrollore si ritrova a dover comandare dei sistemi con segnali analogici, quindi attraverso **output analogici** piuttosto che digitali.

Il microcontrollore però lavora con informazioni binarie e quindi con segnali analogici convertiti in segnali digitali attraverso la **digitalizzazione**: la conversione analogico-digitale in input e viceversa in output.

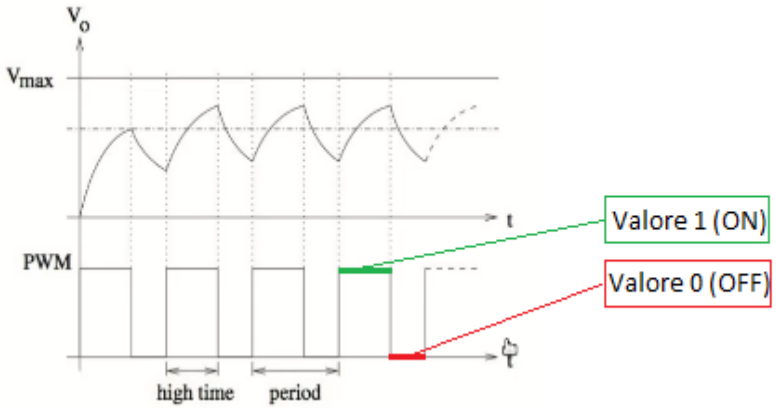
Conversione digitale/analogica

Ci poniamo prima il problema di convertire un dato digitale in analogico, è quello che avviene quando un Microcontrollore deve fornire in output delle informazioni di natura analogica.

Partendo da R bit possiamo codificare 2^R valori (da 0 a 2^R-1 Valori) ma **attenzione:** il massimo valore resta sempre e comunque Vcc (la tensione che si fornisce al microcontrollore); quindi **il range massimo di valori codificabili dipendono dai bit a disposizione**, essi danno informazioni sulle possibilità di suddivisione di un intervallo ma **il massimo valore codificabile dipende da Vcc**. Ad esempio: con 8bit possiamo codificare 256 valori ($2^8 = 256$) che vanno da 0 a 255; se Vcc = 120, il valore massimo che si può avere in uscita sarà sempre e comunque 120, non si potrà superare Vcc.
Dati questi R-bit vogliamo generare un valore in uscita **proporzionale** al numero binario che convertiamo.
I microcontrollori possono supportare nativamente delle parti hardware per il meccanismo di conversione ma esistono microcontrollori che possono anche non supportarlo nativamente.

Conversione digitale-analogica con l'utilizzo di PWM

Il primo meccanismo di conversione da digitale ad analogico che analizzeremo è il PWM (Pulse-Width Modulation); alcuni microcontrollori hanno dei pin che possono supportare il PWM e possono essere utilizzati per implementare un sistema di conversone Digitale-Analogico attraverso un **circuito R-C**

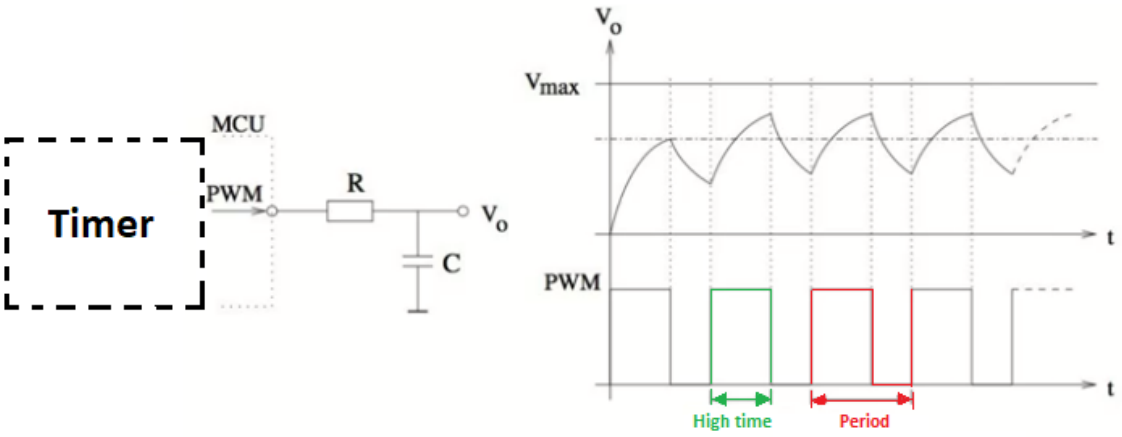


Il funzionamento è il seguente: viene stabilito un **periodo di oscillazione** in cui il pin switcha da 1 a 0 e/o viceversa. Quando il pin viene acceso (1) il condensatore inizia a caricarsi, se sta acceso fino alla carica massima l'uscita sarebbe praticamente massima: Vcc. La pulsazione di accendimento-spegnimento stabilisce la percentuale di tempo di accensione (valore 1) e la percentuale di tempo di spegnimento (valore 0) in proporzione al valore che deve fornire in uscita.

Se il segnale deve valere Vmax l'idea è quella di mantenere il segnale costante (avremo quindi che il segnale vale 1 per l'intero periodo, senza switchare a 0). Nel momento in cui abbiamo un valore che è all'incirca la metà di Vmax, avremo che il segnale varrà 1 solo per metà periodo, oscillando tra 1 a 0; se abbiamo un segnale che oscilla per il 50% a 1 ed il 50% a 0, avremo un valore di tensione che, mediamente, si mantiene al 50% dell'intero segnale.
Se invece il segnale si mantiene a 0, avremo un segnale sempre nullo.

Esempio: con 3 bit, se il valore è 111 (valore massimo) allora il PVM resta acceso tutto il tempo per garantire che l'uscita sia Vmax. Se invece abbiamo un valore che esattamente la metà di Vmax, avremo che il PVM alternerà lo stato di acceso e quello di spento con un periodo che è metà del periodo di oscillazione massimo.
Quando accade ciò si impone al circuito R-C di caricarsi e scaricarsi in tempi più ristretti, appunto metà del tempo totale; in questo modo si può notare che il circuito assume valori leggermente differenti tra loro ma tutti si aggirano attorno al 50% del valore massimo.
Se il segnale PVM sta sempre a 0, il circuito non viene mai caricato e quindi l'uscita sarà sempre 0.

Quello che abbiamo fatto è stato utilizzare un timer che ci permette di utilizzare numero proporzionale che mette in relazione la durata del periodo con il valore della tensione (facendo corrispondere, ad esempio, al 50% del tempo passato nello stato di accesso il 50% del valore massimo [Vmax])



In questo modo si garantisce che il segnale è **proporzionale al periodo di oscillazione.** Chiaramente il segnale uscente non sarà "pulito", il segnale sarà oscillante e l'oscillazione dipende dal valore di R (resistore) e da quello di C (condensatore) ma, seppur non è molto preciso, l'utilizzo del PWM è **molto semplice.**

Svantaggi:

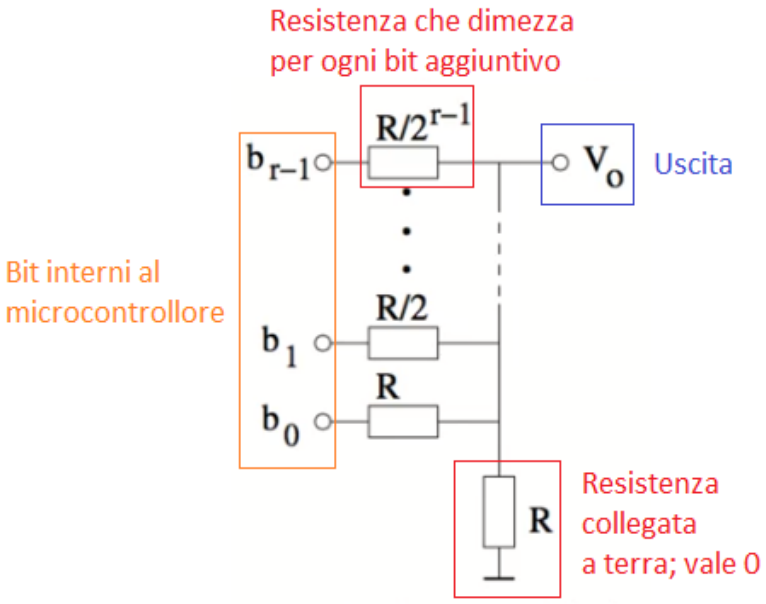
- × serve un timer addizionale che comanda l'accensione e lo spegnimento del pin in proporzione al periodo
- × Il segnale uscente può essere "sporcato" a seconda dei valori di R e di C

Vantaggi:

- √ utilizza un solo pin
- √ È un metodo molto semplice

Conversione digitale-analogica con l'utilizzo del Binary-Weighted Resistor Circuit

Un altro metodo per la conversione analogico/digitale è quello di utilizzare il seguente circuito di resistori;



Ogni bit che si vuole trasferire è rappresentato da un diverso ramo (da b_{r-1} a b_0) ed ognuno di esso è collegato ad una resistenza.
La prima resistenza viene collegata a terra con il valore 0; il resto delle resistenze sono caratterizzate dal fatto che dimezzano il proprio valore ogni volta che si avanza lungo l'intero insieme delle resistenze (b_0 avrà resistenza R, b_1 avrà una resistenza $\frac{R}{2}$, etc..); ognuna ha, praticamente, valore $\frac{R}{2^{r-1}}$ [avremo così R per il bit 0, R/2 per il bit 1, R/4 per il bit 2, R/8 per il bit 3, etc…];

Il capo destro di ogni resistenza è collegato in uno stesso punto che darà poi un uscita V_0.

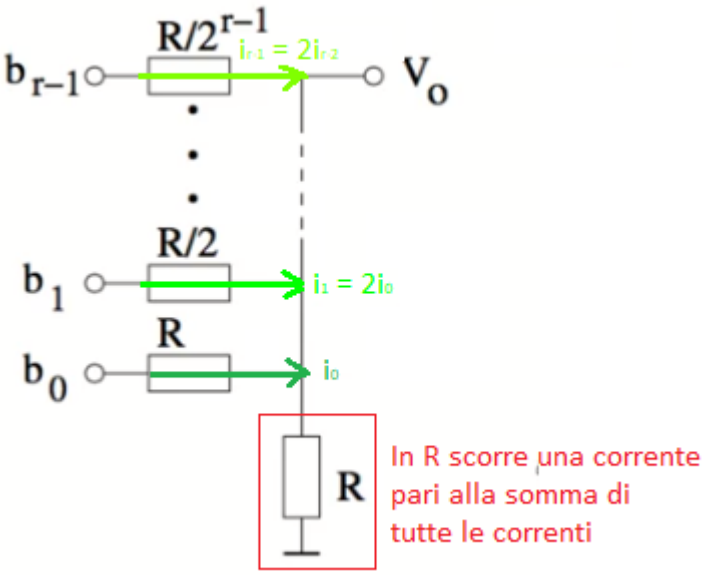
Se settassimo ogni bit a 1 staremmo praticamente passando il valore Vcc ad ogni resistenza; possiamo notare, ad esempio, che nel primo bit avremmo il valore di Vcc a sinistra e il valore V a destra, uscente dalla resistenza e avente un valore diverso da Vcc (Poiché V = R * i ≠ Vcc), la corrente che circola allora sarà data dal rapporto della differenza tra le due tensioni e la resistenza: $i = \frac{(Vcc-V)}{R}$.
Dato che le resistenze sono in parallelo, esse avranno nel capo destro lo stesso valore di differenza di potenziale V; Dato che si mantiene la stessa differenza di potenziale in uscita da ogni resistore, si dovrà garantire un determinato livello di corrente i. Nello specifico diremo che **si avrà una corrente che sarà crescente man mano che la resistenza diventa sempre più piccola** (le resistenze si dimezzano via via che si sale e, in contrapposizione, le correnti aumentano).

Quindi se in R circolerà una corrente i, in $R/2$ circola una corrente 2i, in R/4 la corrente sarà 4i, etc… **Le resistenze si dimezzano e, dualmente, le correnti raddoppiano, man mano che si sale lungo il circuito.**
Il comportamento della corrente ci ricorda la codifica dei numeri binari: avanzando da destra verso sinistra le cifre binarie danno un contributo doppio:

1	0	1	1	0
2^4	2^3	2^2	2^1	2^0

Il bit che occupa la posizione 2^4 ha un peso maggiore rispetto a quello che occupa la posizione 2^3 e così via…

L'andamento delle correnti rispecchia quello delle cifre binarie che compongono un qualsiasi numero. Esse confluiscono poi in un unico punto, che sarà quello che ci darà il valore Vcc.



Le correnti che si forniscono sono l'una il doppio dell'altra; adesso ci poniamo il quesito: che corrente scorre su R (la resistenza collegata a terra)? In R scorre la somma dei flussi di corrente, ogni corrente è "pesata" a seconda della posizione che occupa il bit del ramo di appartenenze: la corrente più alta appartiene al bit più in alto, quella più bassa a quello più in basso.

La somma di queste correnti da effettivamente un valore proporzionale al valore che stiamo codificando: ogni corrente è **"pesata"** come sarebbe pesato il valore binario corrispondente.

Il nostro intento è quello di misurare la corrente finale in R così che alla fine otteniamo una caduta di tensione totale proporzionale alla combinazione di bit da trasmettere; ogni componente di

quest'ultima ha, come detto, una corrente pesata a seconda del bit che le corrisponde e questo garantisce la proporzionalità tra bit digitali e sagnale analogico.

Quando il bit è 0 possiamo immaginare come se la resistenza fosse "scollegata", non da alcun contributo in corrente.

Svantaggi:

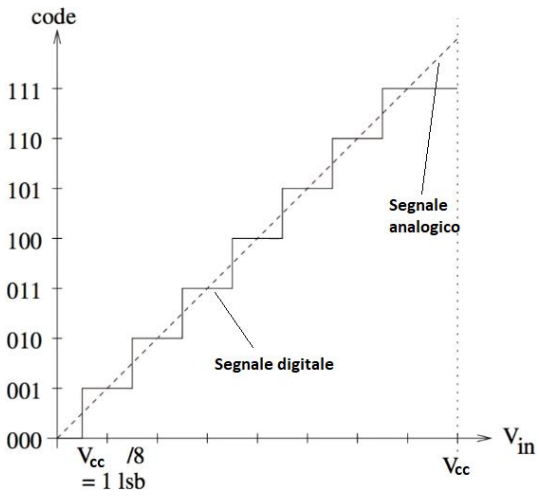
- × Il principale svantaggio sta nel fatto che ogni resistenza va scelta appositamente con una certa precisione: per dare un certo valore di corrente i devono essere abbastanza **accurate**, quindi si deve scegliere un valore di R che non sia approssimativo bensì che sia certo.

Questo approccio è notevolmente differente dall'approccio PWM

Conversione analogica/digitale

Supponiamo adesso di occuparci del problema opposto: la conversione di un segnale analogico in un segnale digitale; questa torna utile, ad esempio, quando dobbiamo gestire un segnale proveniente dall'esterno di natura analogica nel microcontrollore, che è in grado di gestire solo segnali di natura digitale.

Avremo a disposizione un range di tensione che va da 0 a Vcc (valore di tensione massimo), il nostro obiettivo è quello di porre in ingresso un segnale che varia in questo intervallo (in maniera continua) all'interno di un microcontrollore, "trasformando" i valori analogici del segnale in valori 0 - 1 (digitali) per essere gestiti dal microcontrollore. Raggruppando r bit alla volta potremo utilizzare $N = 2^r$ Simboli, che utilizzeremo per "mappare" i valori analogici per convertirli;



Attenzione: ripetiamo un'altra volta che il numero massimo di bit che si possono utilizzare non sancisce il numero massimo che si può trasmettere: un numero elevato di bit non consente di trattare un valore maggiore di Vcc, bensì di poter

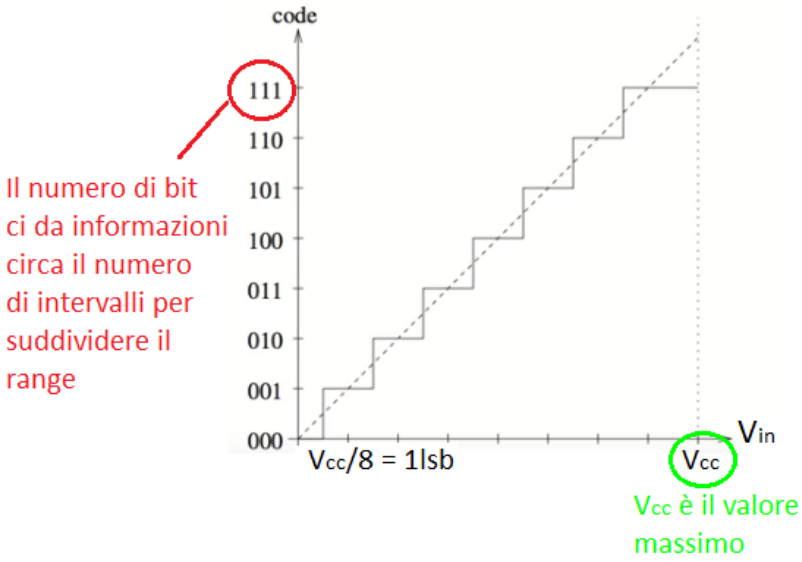
suddividere il range [0 - Vcc] in più intervalli, aumentando il numero dei simboli aumenta la "granularità", la precisione del sistema. Ogni volta che si aggiunge un bit si raddoppia il numero di intervalli possibili all'interno del range.

Si osservi il seguente esempio:

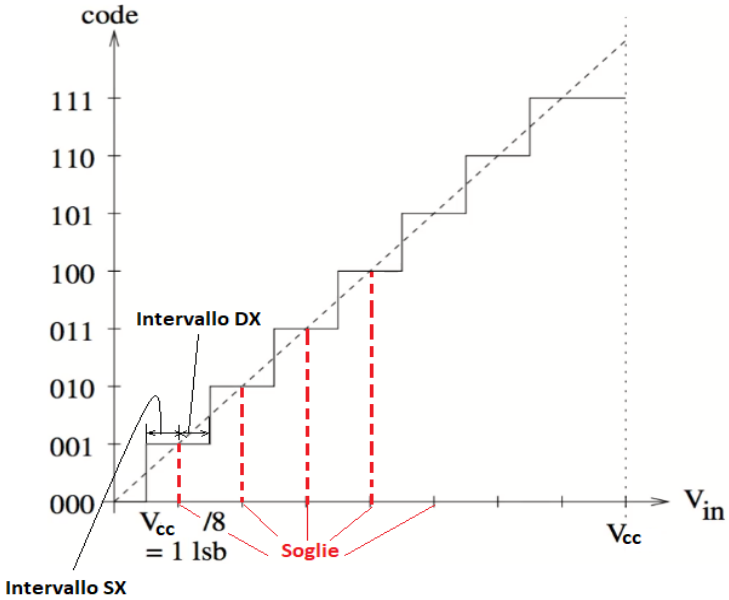
Avendo 3 bit otteniamo 8 simboli, le 8 combinazioni da tre bit che vengono rappresentate nell'asse delle ordinate (asse "code").

Il segnala analogico va da 0 a Vcc e può assumere in modo continuo uno dei valori all'interno del range, noi abbiamo al massimo 8 bit per "mappare" tale segnale.

Il numero di bit ci dà una **risoluzione** più o meno migliore: aumentando il numero di bit l'approssimazione del segnale analogico è migliore.



Si suddivide l'intervallo in base al numero di bit disponibili: nel caso di 3bit si ottengono 8 intervalli, e si definisce una soglia minima e una soglia massima per ogni intervallo. Da un certo valore LSB (Least Significant Bit) in giù il segnale vale 0, dopo di che codifico ogni range di valori Vin con un valore codificato



Il segnale analogico può assumere in maniera continua ogni valore all'interno dell'intervallo [0 - Vcc] (linea tratteggiata), mentre il segnale digitale suddivide l'intervallo in range in cui il segnale può assumere uno dei valori mappati (codificati), ad esempio:

in figura il primo range va da 0 a Vcc1, in questo caso il valore codificato è 000 -LSB-; il secondo range va da Vcc1 a Vcc2, qui il valore assunto è 001, etc…
Chiaramente, deve essere deciso entro quale range il segnale varrà, ad esempio, il simbolo 001 ed entro quale altro range esso assumerà il valore 010, etc..

Per stabilire questi range si suddivide l'intervallo delle tensioni in N sotto-intervalli (con N pari al numero di simboli utilizzati: $N = 2^r$), dividendo Vcc per il numero N otterremo la durata di ogni singolo intervallo.

Ricapitolando: da meno infinito ad una certa soglia (Chiamata lsb: least significant bit) si stabilisce che il segnale valga 0. Dopo di che vengono fissate delle soglie equidistanti in modo che esse

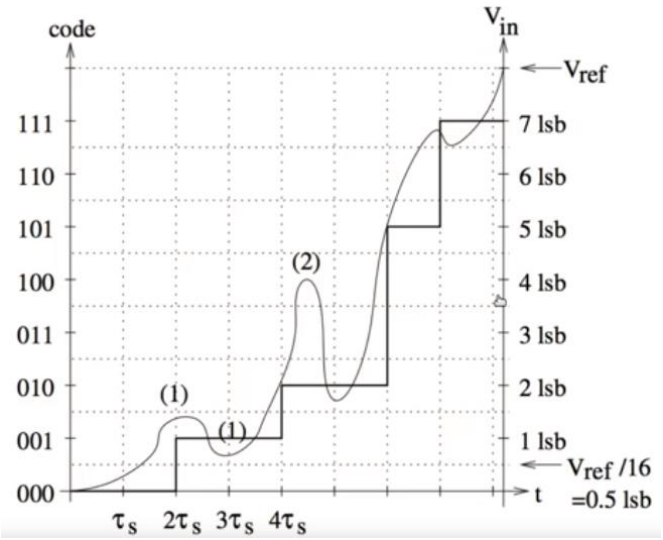
risultino al centro di ogni simbolo rappresentato: in questo modo avremo un determinato range a destra e a sinistra della soglia in cui il segnale assumerà il valore di un determinato simbolo. Quindi quando il segnale analogico si trova entro un certo range, ad esso si associa un certo valore digitale.
Chiaramente, il segnale ricevuto spesso non sarà lineare, ma potrebbe assumere vari valori e questo ci porterebbe ad un certo errore di "approssimazione" inevitabile, dovuto alla digitalizzazione.

NB: non sempre avere una risoluzione maggiore possibile è un aspetto positivo, in quanto l'aumento di risoluzione implica un aumento dei costi di produzione: in base al contesto potrebbe bastare una bassa risoluzione (un basso numero di bit di raggruppamento) per stabilire un corretto funzionamento con costi contenuti.

Problematiche della conversione analogico-digitale

Il segnale digitale è un segnale discreto: esso assume dei valori scandendo il tempo in maniera discreta, suddividendo in intervalli temporali chiamati periodi di clock (o tempo di campionamento): in ogni periodo il simbolo assume un determinato valore. Indichiamo con τ il periodo di campionamento.
Il clock scandisce ogni τ il segnale analogico, ricreandone una sua "copia" in formato digitale. Il segnale digitale viene quindi scandito dal clock e quindi esso non è un segnale non continuo;
Attenzione: il valore di τ non deve necessariamente essere pari al ciclo di clock; un segnale, infatti, può essere campionato ogni 2,3,.. cicli di clock, sicuramente comunque parliamo di un periodo **maggiore o uguale al ciclo di clock**, quindi corrisponde al ciclo di clock o ad un multiplo di questo.

Osserviamo un esempio per capire alcuni problemi della conversione analogico-digitale:



Da 0 a $-\infty$ il segnale vale 0 e viene codificato dai bit 000. Al tempo Ts il segnale analogico inizia a crescere e durante l'evoluzione del periodo inizia a crescere ma questo incremento viene letto solo al tempo 2Ts.

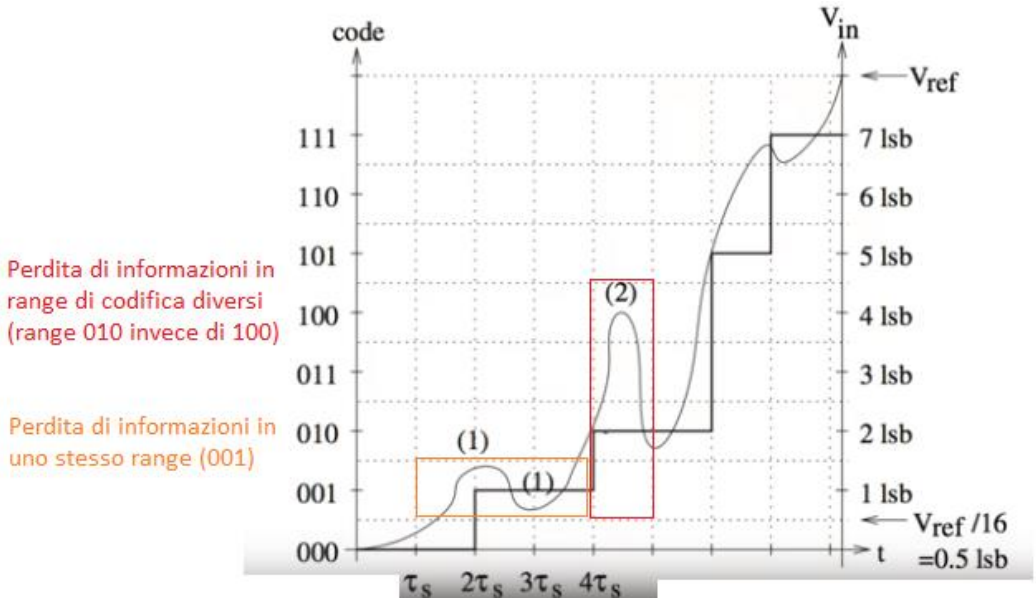
Si osservi allora che tra 2Ts e 4Ts il segnale oscilla con valori che comunque ricadono all'interno del range codificato dai bit 001 quindi non viene registrata alcuna variazione del segnale onda quadra.

Questo errore dovuto alla mancata "registrazione" della variazione oscillante del segnale avviene anche tra 4Ts e 5Ts, tuttavia stavolta il segnale non oscilla in uno stesso range, bensì vi è un picco che dovrebbe essere codificato dai bit (100) ma dato che questo picco rientra nella durata del periodo non viene captato alcuna variazione e quindi il segnale viene codificato (erroneamente) dai bit 010.

Approfondiamo accuratamente i due problemi:

Il primo problema è dovuto al fatto che il segnale oscilla all'interno dell'intervallo tra un tempo di campionamento e l'altro, questa perdita di informazione si può minimizzare accorciando l'intervallo da un valore all'altro ma non si può annullare, parliamo di **perdita informazione dovuta all'intrinseca risoluzione che non può catturare tutti i valori analogici del segnale.**
Il secondo problema è distinto dalla prima perdita di informazione e si va a sommare al primo: precedentemente l'oscillazione portava ad una parziale perdita di informazione che rimaneva nel range di appartenenza, nel secondo caso **il range di appartenenza viene "sforato" da un picco che lo porta in delle regioni diverse da quelle codificate** (prima 011 e poi 100).

Quindi nel **primo caso** abbiamo perso un'**informazione riguardante la forma del segnale,** nel **secondo caso**, oltre ad aver perso la forma del segnale, abbiamo perso anche un'**informazione sulla codifica** di quest'ultimo.



Il primo problema persiste poiché il segnale analogico è sempre caratterizzato da forme sinusoidali che variano, esso non è grave poiché la forma d'onda digitalizzata non deve essere identica a quella originariamente analogica, in quanto la conversione da analogico a digitale porta ad una sua approssimazione e questo ne giustifica la perdita della sua forma originale.
Il secondo problema si sarebbe risolto campionando il segnale in intervalli più stretti, esso è più grave poiché porta ad una perdita di informazione che porta ad un'informazione contraffatta. Quindi per la corretta conversione del segnale da analogico-digitale occorre ovviare a tale problema, considerando un intervallo di campionamento più "stretto".

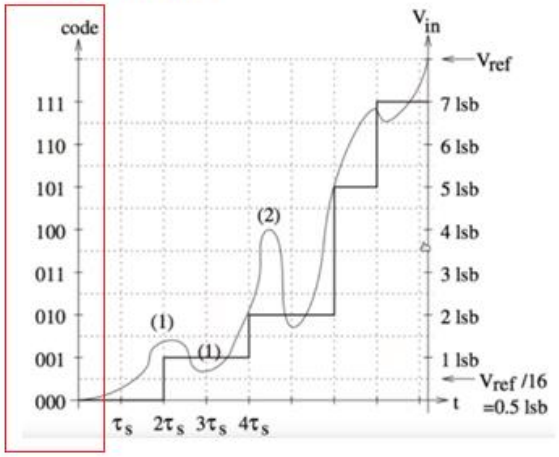
A tal proposito interviene il **teorema di Nyquist**, secondo il quale per supportare una certa frequenza di variazione del segnale si deve garantire una frequenza di campionamento appropriata; se il segnale ha un periodo di variazione Tmax allora si deve garantire che il tempo di campionamento sia minore della metà del tempo di oscillazione:

$$f_{max} < \frac{fs}{2} = \frac{1}{2ts}$$

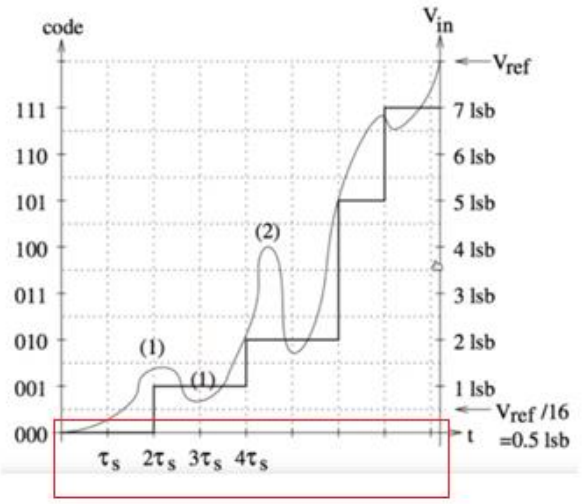
Il tempo di campionamento deve essere più stretto della metà del periodo di variazione del segnale.
Quindi quando voglio codificare in binario so già che avrò due problemi:

1. Problema riguardante alla perdita di informazione nella variazione del valore analogico; soluzione a questo problema: **aumento della risoluzione**, ovvero aumento il numero di Bit usati per codificare, in questo modo si avranno più combinazioni e quindi si copriranno range molto stretti e precisi.

Aumentando il numero di bit
aumentano le combinazioni



2. Fissata la risoluzione, non si devono perdere informazioni sulla codifica del segnale: se un segnale passa da un range di codifica ad un altro questo cambiamento deve essere presente nel segnale digitale. Soluzione a questo problema è il **restringimento dell'intervallo** attraverso il teorema di Nyquist.



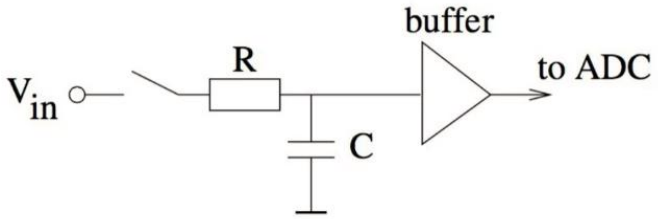
Restringimento degli intervalli (Nyquist)

Convertitori analogico/digitali

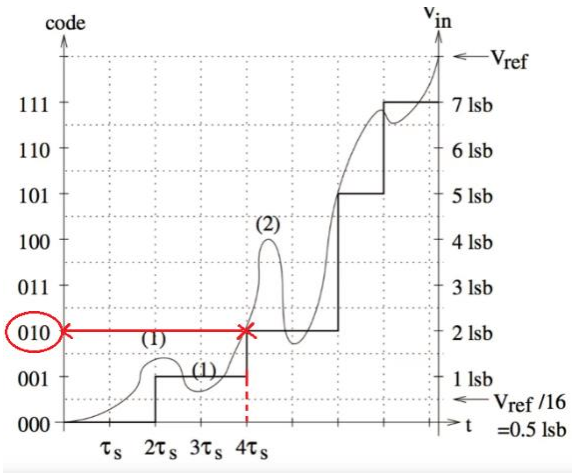
Abbiamo in qualche modo parlato di "campionamento" del segnale e spesso abbiamo immaginato che in un certo istante di clock viene "visualizzato" il valore del segnale per essere convertito in digitale; per far ciò, chiaramente, si deve far si che l'osservazione si alterni nei momenti di campionamento. Si deve far si che il pin che converte il segnale analogico si alterni tra acceso e

spento, poiché se così non fosse il segnale risulterebbe essere continuo e non avverrebbe la conversione in digitale.

Nei microcontrollori questo funzionamento è intrinsecamente adattato di default dal **Sample/Hold:** negli istanti di tempo in cui si deve campionare (al fine della digitalizzazione) viene chiuso l'interruttore (negli istanti τ); istantaneamente il condensatore si caricherà, dopo di che, scattato il periodo τ, l'interruttore verrà nuovamente aperto. Il valore del segnale resterà "immagazzinato" nel condensatore carico. Le azioni di [chiusura dell'interruttore → carica del condensatore → apertura dell'interruttore] avviene praticamente in maniera istantanea, così da non avere dei campioni che entrino in contrasto tra loro in diversi periodi.



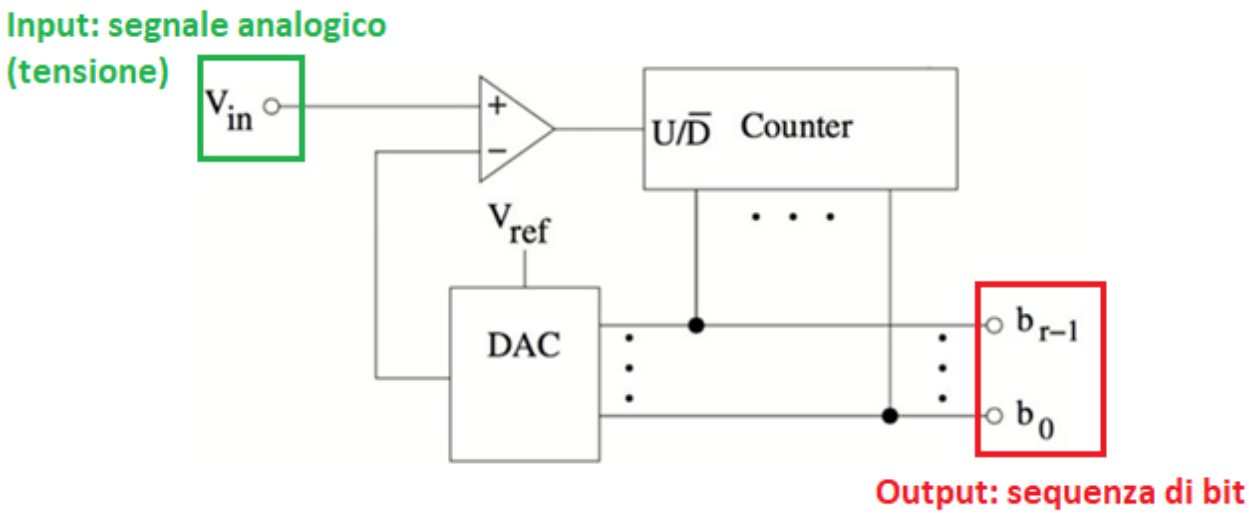
Al campione del segnale analogico verrà poi assegnato un determinato valore digitale, ci chiediamo quindi come avviene questo processo.



Analizziamo questo procedimento utilizzando un particolare convertitore analogico digitale: il **tracking converter.**

Tracking Converter

Analizziamo il procedimento descritto in un particolare convertitore analogico/digitale (ADC: Analog/Digital Converter): il **tracking converter;** questo si occupa di convertire un segnale analogico in ingresso in una sequenza di bit (in un segnale digitale).

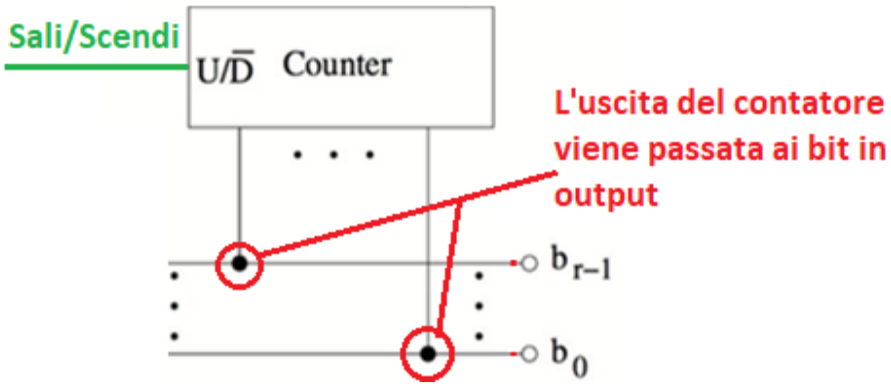


Come si può notare, la particolarità di questo ADC è che al suo interno contiene la sua "controparte": il convertitore digitale/analogico (DAC: Digital/Analog Converter), implementato con tecnica PWM o con il circuito di resistori binary weighted.

Descriviamo in dettaglio il funzionamento del tracking converter:

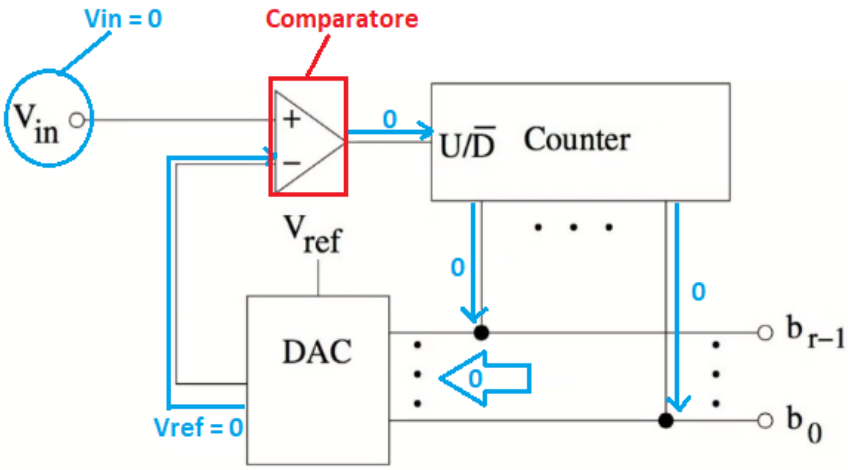
Avremo in ingresso un segnale (una tensione) Vin che oscilla tra 0 e Vcc; l'obiettivo è quello di "trasformare" in una sequenza digitale (= segnale digitale) Vin: ad un determinato valore di Vin deve essere attribuita una ben determinata sequenza di bit.

Nel circuito possiamo osservare un contatore: dati R bit il contatore conta verso l'alto e verso il basso, per chiarire meglio il funzionamento spieghiamo un pratico esempio: supponiamo di avere 3bit, se si deve salire il contatore inizierà ad incrementare il proprio valore passando in sequenza dai valori: 000 -> 001 -> 010 -> 011 -> … -> 111;
se invece deve scendere, inizierà a decrementare il numero attuale, scendendo fino al valore minimo 000. Quindi l'unica cosa che fa è **contare verso l'alto o verso il basso dati in ingresso R bit.** Il counter dà in output il proprio valore attuale, passandolo nei bit rappresentati l'uscita.



I bit in uscita dal counter (il valore attuale del counter), oltre che a trovarsi nei rami dei bit in output, si trovano anche in ingresso ad un convertitore Digitale/Analogico (DAC). Per capire meglio perché avviene ciò facciamo un esempio di conversione di un singolo segnale analogico:

Arriva in ingresso al tracking converter un segnale analogico Vin (oscillante in modo continuo tra 0 e Vcc); fintanto che il segnale sia nullo, il contatore non dovrà né incrementare né decrementare: i suoi bit in uscita saranno 0. In ingresso al DAC si avranno allora bit 0, quest'ultimo, convertendo i bit in segnale analogico, darà in output un segnale nullo. Il segnale in uscita dal DAC viene posto in ingresso ad un **comparatore:** questo compara Vref e Vin, controllando se Vin è maggiore/minore di Vref; traendo le conclusioni che: **se Vin è minore di Vref da in uscita 1, altrimenti da in uscita 0**; questo perché se Vin è maggiore di Vref, si deve far incrementare il contatore, in modo che esso passi da 000 (valore di Vin iniziale) a 001.

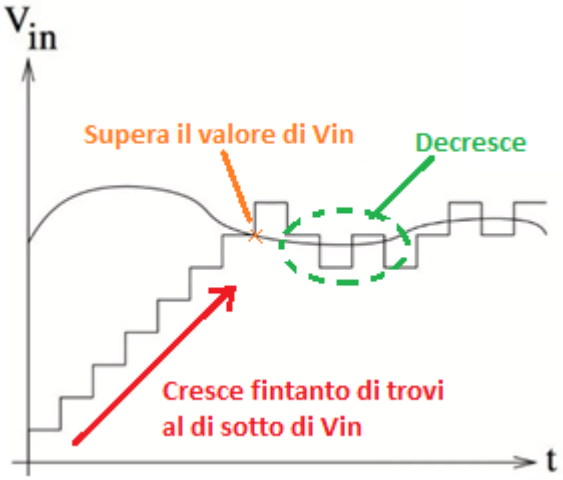


Vref quindi serve per "tener memoria" del segnale precedentemente inviato, così che questo possa essere comparato con il segnale attualmente in ingresso Vin e, in base alla comparazione, il counter incrementerà o decrementerà il suo valore.

Una volta che il counter incrementa i bit in uscita, il convertitore digitale analogico DAC non avrà più in input 000 ma 001, quindi, convertendo questo segnale digitale, otterrà una piccola tensione $Vref_2$ (= un segnale analogico), questo verrà passato al comparatore e in base al confronto con Vin si incrementa/decrementa il contatore, continuando il processo.

Dopo un certo numeri di cicli, avremo un certo $Vref_n$ che avrà raggiunto il valore di Vin (sarà maggiore o uguale ad esso), dunque il comparatore genererà il valore 0 e il contatore dovrà decrementare.

Possiamo osservare il grafico dei valori assunti dal convertitore:



Come si può notare, il segnale Vin avrà un andamento oscillatorio; l'uscita del counter (che ricordiamo, viene passata ai rami dei bit in uscita e viene posta in ingresso al convertitore DAC), ha un andamento "a scala". Fintanto che non raggiunge/supera Vin possiamo notare una crescita, non appena lo supera possiamo notare una decrescita.

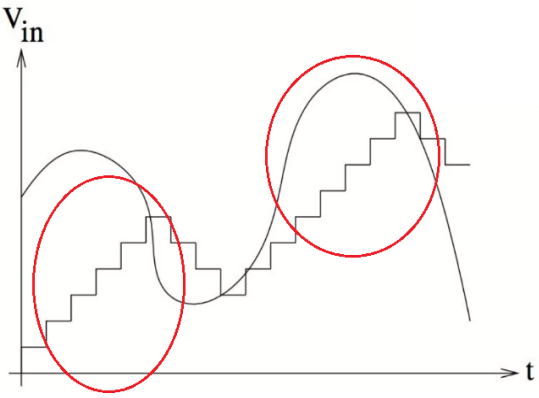
Quindi il funzionamento del tracker converter lo si può riassumere nei seguenti punti:

1. Il segnale in ingresso al convertitore ADC Tracker converter viene passato al **contatore** (nel primo passaggio ignoriamo volutamente il comparatore poiché non avrà nulla da comparare)
2. Supponendo che Vin sia pari a 0 (valore iniziale), il **contatore** non farà altro che generare dei bit nulli che passerà ai **rami in uscita** (rappresentati i bit) e al **convertitore DAC**

3. Il **convertitore DAC genererà una tensione** (se gli viene passato un segnale digitale nullo anche la tensione sarà a sua volta nulla)
4. Il segnale in uscita dal DAC sarà posto in ingresso ad un **comparatore** che lo comparerà con Vin: **se sarà minore allora genererà 1 altrimenti 0**.
5. Da qui inizia nuovamente il ciclo: in base all'uscita del comparatore, il contatore incrementerà o decrementerà, generando dei nuovi bit, che passerà sia ai rami in output dei singoli bit sia al DAC; quest'ultimo genererà una tensione (Stavolta presumibilmente diversa dalla precedente); la tensione sarà passata al comparatore che la comparerà con Vin e si riprende il ciclo.

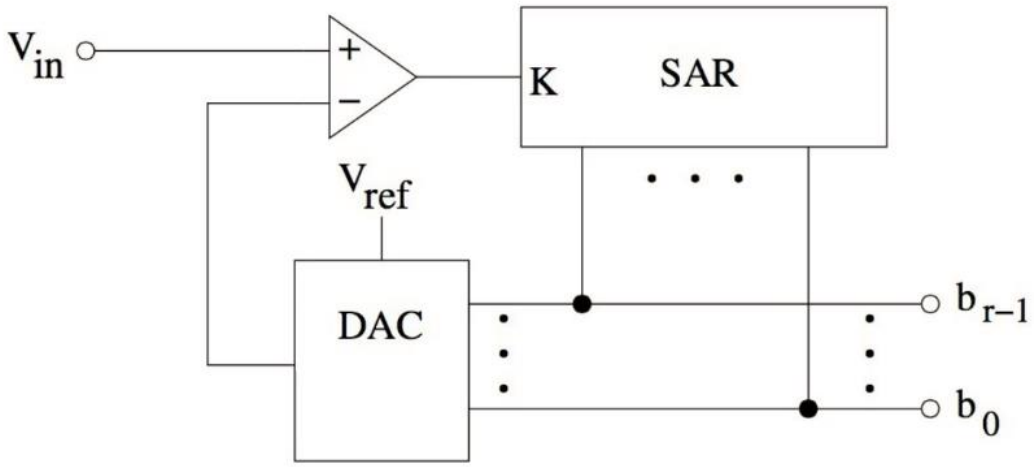
Problematica:

Il difetto principale di questa tecnica è che, iniziando da zero, il segnale digitale partirà piuttosto "lentamente" e dovrà perdere del tempo per "raggiungere" il segnale Vin. **Se il valore di input Vin varia molto velocemente, i tempi di reazione del contatore** (che varia il numero in base all'incremento e al decremento) **potrebbero essere piuttosto lenti.** Rimanendo più indietro, chiaramente, il segnale risulterebbe falsato e quindi vi è un errore piuttosto grossolano nella conversione. [Come si può notare dall'immagine]



Successive Approximation Converter

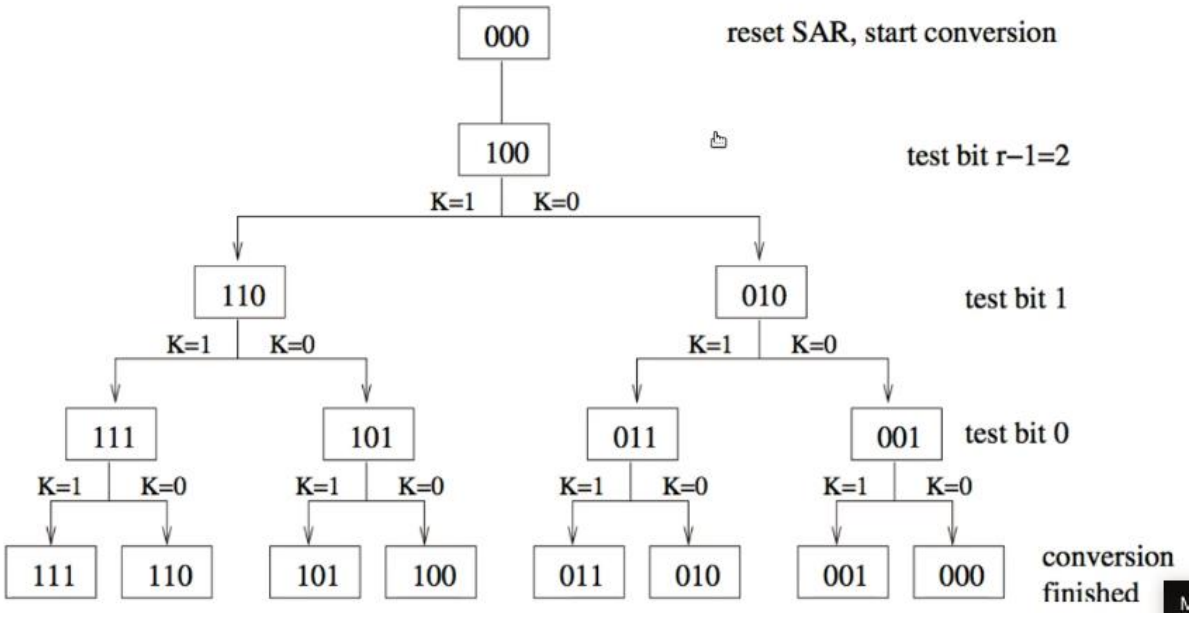
Per ovviare al problema della lentezza dei tempi di reazione del tracking converter parleremo del **Successive Approximation Converter** (convertitore ad approssimazione successiva)



Come si può notare, lo schema è molto simile a quello del Tracking Converter, tuttavia essi differiscono poiché nel Successive Approximation Converter vi è il blocco **SAR: Successive Approximation Register** al posto del contatore. Il comportamento del Successive Approximation Converter, infatti, è analogo a quello del Tracking Converter, con l'unica differenza che esso utilizza il SAR.
Per spiegare come funziona quest'ultimo facciamo un esempio:

Supponiamo di avere parole di codice di 3bit; il SAR, a differenza del contatore, non partirà dal valore 000 ma da un **valore intermedio**, ad esempio **100** [Il valore minimo assumibile è 000, equivalente a 0 in decimale ed il valore massimo raggiungibile è 111 pari al valore decimale 7; il valore intermedio sarà 100 equivalente a 4 in decimale].
A questo punto il SAR passa il valore intermedio al DAC che lo converte e lo riporta in input al comparatore, quest'ultimo compara il valore intermedio e il valore in ingresso. Se il valore in input è **maggiore** allora viene passato al SAR il **valore 1** e quest'ultimo **incrementa il valore intermedio,** se viene passato il **valore 0** e il SAR **decrementa il valore intermedio**.

Spieghiamo graficamente il comportamento del SAR:



Quando il comparatore dà in input al controllore il valore 1, quest'ultimo provvede ad incrementare nel valore 110, se Vin > Vref allora il comparatore torna 0 e il comportamento del contatore si sposta, nel grafico, verso destra: provvede ad un decremento.

Successivamente, viene effettuato un'altra comparazione e ci si comporta nella medesima maniera.

In questo modo possiamo lavorare nella metà dell'intervallo superiore o inferiore rispetto al valore intermedio considerato: se il valore intermedio è 4 (100), quando il contatore riceve 1 si sposta nel valore 6 (110) che è la metà dell'intervallo superiore [5 **6** 7], quando il contatore riceve 0 si sposta nel valore 2 (010) che è la metà dell'intervallo inferiore [0 1 **2** 3]. In altre parole, attraverso una **ricerca binaria** diminuisce il tempo di raggiungimento del valore corrispondente, spostandosi tra i range possibili.

Utilizzando il Successive Approximation Converter (che utilizza il SAR) abbiamo ottimizzato i tempi, avendo una complessità circuitale che è praticamente analoga a quella del Tracking Register (che utilizza il contatore).