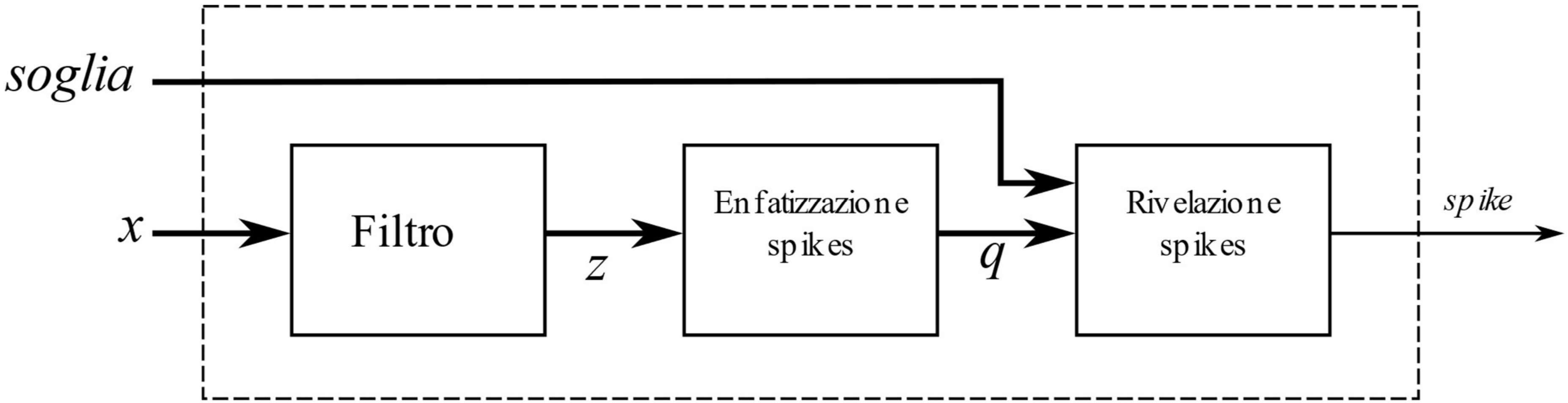
Corso di Architettura dei Sistemi Integrati
Progetto

Il sistema da realizzare

Il circuito che realizzeremo quest'anno ha il compito di individuare degli impulsi (*spikes*) all'interno di un segnale proveniente da un sensore; è questo uno scenario comune in numerose applicazioni (ad esempio, in campo biomedico).
La figura seguente mostra uno schema di massima del circuito:



Segnale di ingresso

Il segnale di ingresso, x, rappresenta i dati provenienti dal sensore esterno.

Assumeremo che questi dati siano stati campionati (da un convertitore Analogico-Digitale a monte, non mostrato in Figura) con un frequenza di 100MHz. Questa sarà quindi anche la frequenza del clock che pilota il nostro sistema: f_{clk}=100MHz

I dati di ingresso sono in ***virgola fissa***; rappresentano valori compresi nell'intervallo $[-1, 1)$ ed il peso del bit meno significativo (LSB) è: $LSB = 2^{-10}$

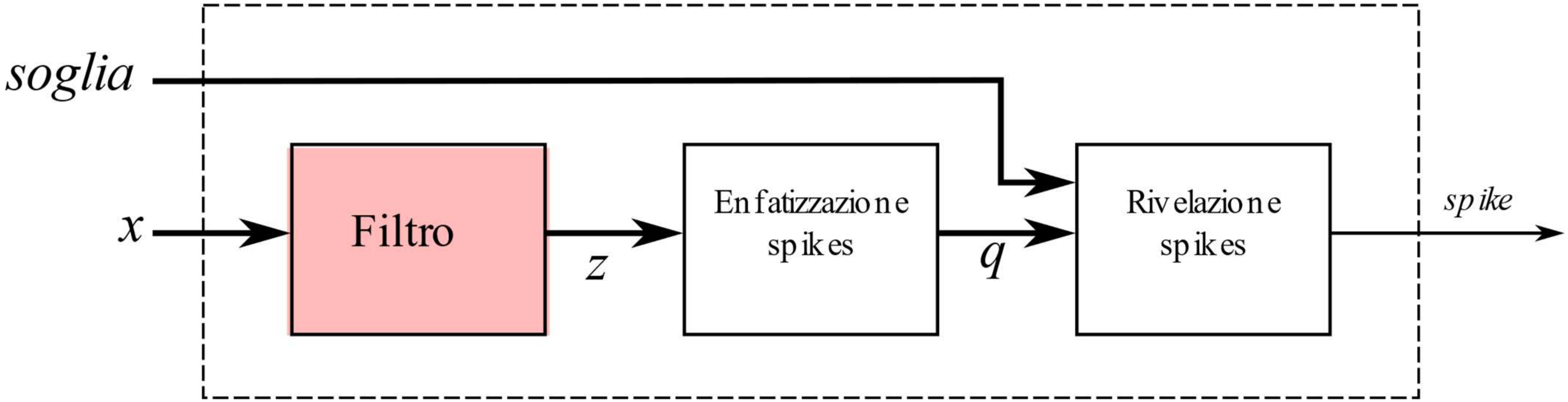
Filtro di ingresso

Filtro di ingresso

Il primo elemento del nostro sistema è un filtro passa-basso, necessario ad attenuare il rumore presente nel segnale di ingresso.

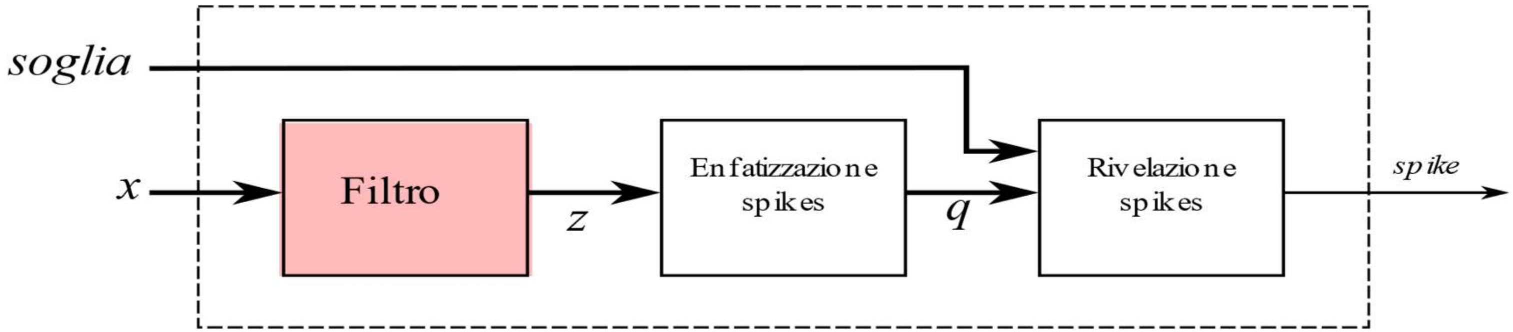
Assumiamo che:

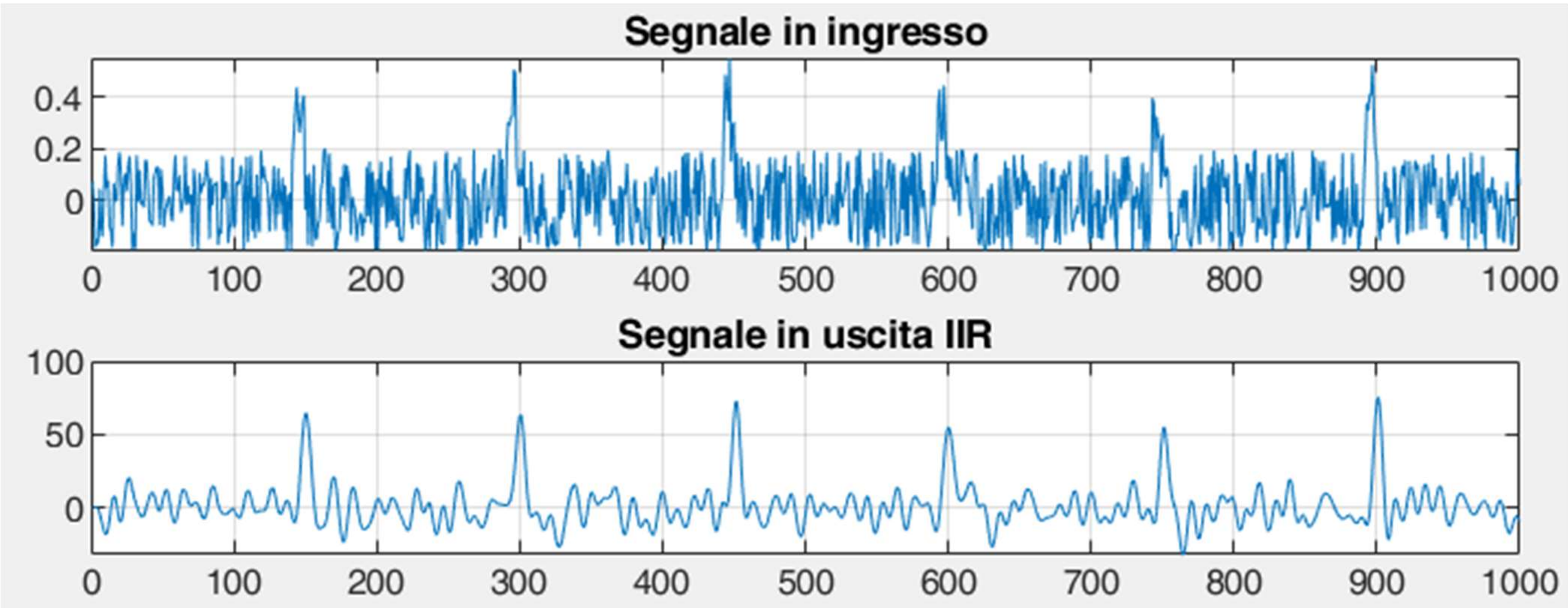
- La frequenza di taglio sia di 10MHz
- La frequenza di campionamento (di clock) è: f_{clk}=100MHz



Assumeremo che filtro di ingresso sia di tipo IIR (a risposta impulsiva infinita)

Filtro di ingresso





Filtro di ingresso

Un filtro è un sistema lineare, tempo-invariante, il cui funzionamento può essere descritto come convoluzione fra il segnale di ingresso *x* e la risposta impulsiva *h*:

$$y(n) = \sum_{k=0}^{+\infty} h(k) \cdot x(n-k)$$

Nel caso dei sistemi FIR (Finite-Impulse-Response), $h(k)$ ha una durata finita ($h(k)=0$ se $k > k_{max}$).

Nei sistemi IIR (Infinite-Impulse-Response), $h(n)$ ha una durata infinita. Un filtro IIR non è realizzabile implementando la sommatoria precedente (sarebbe necessario un numero infinito di moltiplicatori...)

Vantaggi filtri FIR: Sempre stabili, facilmente dimensionabili
Vantaggio filtri IIR: A parità di specifiche, richiedono meno hardware

Filtro IIR

Consideriamo i filtri IIR il cui comportamento è descritto dall'equazione seguente (in cui i coefficienti a_i e b_i sono costanti):

$$y(n)+\sum_{k=1}^{N} a_k \cdot y\left(n-k\right)=\sum_{k=0}^{M} b_k \cdot x\left(n-k\right)$$

Questa classe di filtri IIR è realizzabile attraverso una architettura ricorsiva (nella quale l'uscita $y(n)$ è funzione delle uscite precedenti, $y(n\text{-}1)$, $y(n\text{-}2)$ ecc.).

La risposta del filtro dipende dalla scelta dei coefficienti a_k e b_k.

Filtro IIR

In pratica, i filtri IIR vengono molto spesso realizzati come cascate di sezioni del secondo ordine (*second-order sections*, SOS).
Una sezione del secondo ordine (filtro biquadratico) realizza la funzione:

$$y(n) + a_1 y(n-1) + a_2 y(n-2) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2)$$

Possiamo ottenere la funzione di trasferimento del filtro applicando la *Z* trasformata:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

Nel nostro caso, utilizzeremo un sistema composto dalla cascata di due sezioni del secondo ordine.

Coefficienti del filtro

La risposta del filtro (passa-alto, passa-basso, attenuazione fuori banda ecc. ecc.) dipende dalla scelta dei coefficienti a_k, b_k. Per il calcolo dei coefficienti del filtro ci affidiamo a matlab.

Senza entrare in ulteriori dettagli (riportati appendice a queste slides per i più curiosi) osserviamo che i coefficienti forniti da matlab devono essere manipolati prima di passare alla realizzazione hardware:

- I coefficienti devono essere opportunamente **scalati** (ovvero, moltiplicati tutti per una medesima costante) in modo da poter assumere il medesimo formato per ingresso e uscita delle due sezioni del filtro (nel nostro caso, ingresso e uscita del filtro sono rappresentate nel formato: Q1.10)
- I coefficienti devono essere **quantizzati**,

Utilizziamo per i coefficienti la rappresentazione **Q2.10**

Valori dei coefficienti

Vi verrà fornito un file di testo con i valori da utilizzare nel codice HDL

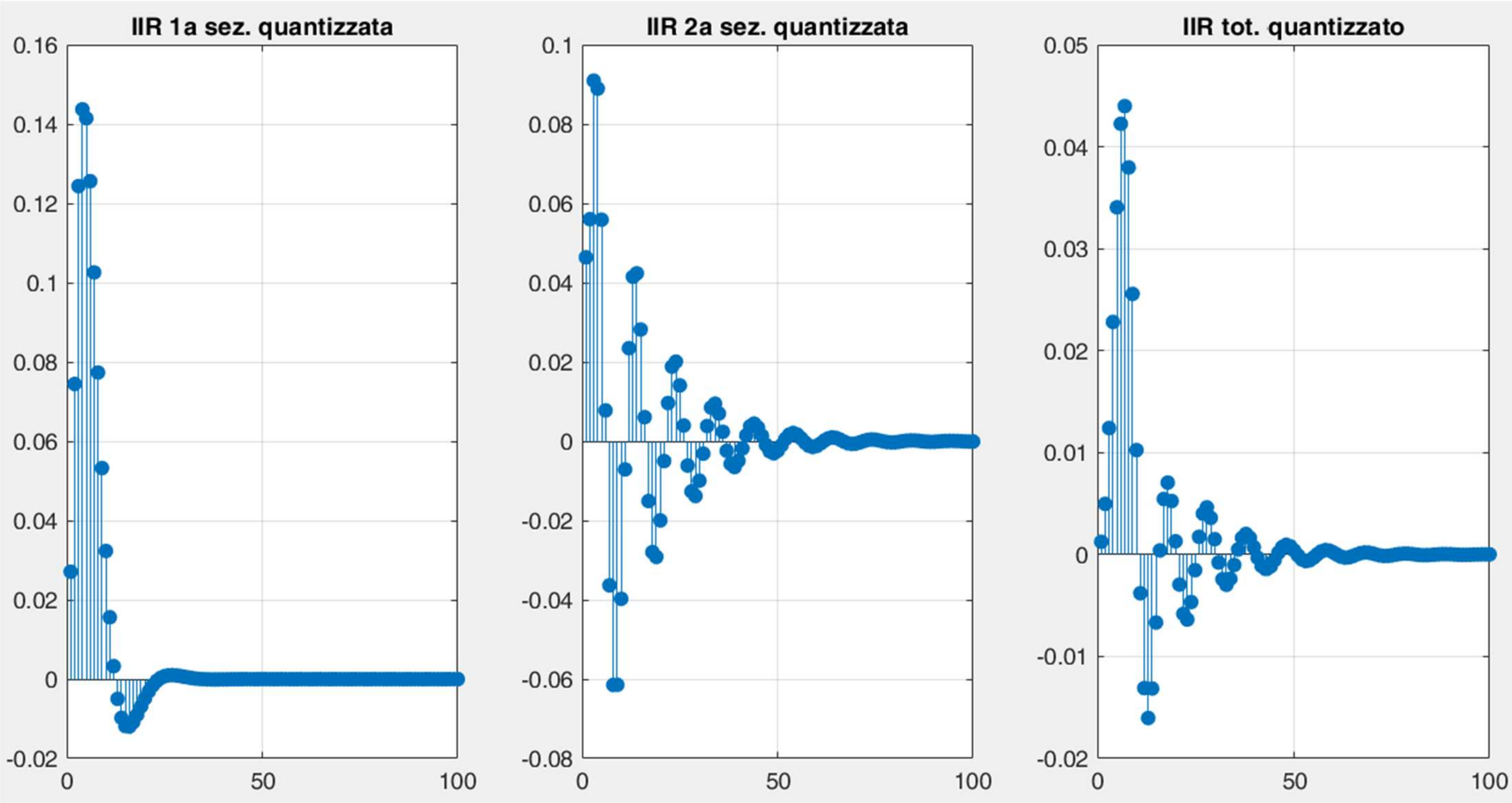
```
Coefficienti quantizzati della PRIMA sezione del filtro.
Numeratore della funzione di trasferimento:
b0:
000000011100      0.0273437500000
b1:
000000100001      0.0322265625000
b2:
000000011100      0.0273437500000
Denominatore della funzione di trasferimento:
NOTA: sono riportati i valori di a cambiati di segno
-a0:
110000000000      -1.0000000000000
-a1:
011000101001      1.5400390625000
-a2:
110101101110      -0.6425781250000
```

```
Coefficienti quantizzati della SECONDA sezione del filtro.
Numeratore della funzione di trasferimento:
b0:
000000101111      0.0458984375000
b1:
111111110010      -0.0136718750000
b2:
000000101111      0.0458984375000
Denominatore della funzione di trasferimento:
NOTA: sono riportati i valori di a cambiati di segno
-a0:
110000000000      -1.0000000000000
-a1:
011000001000      1.5078125000000
-a2:
110010001111      -0.8603515625000
```

Notare che vengono riportati i valori del denominatore (a_1, a_2) **cambiati di segno**, in quanto così meglio si adattano alla realizzazione hardware. E' riportato anche il valore di a_0, che però sarà inutilizzato (è unitario).

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

Risposta impulsiva del filtro



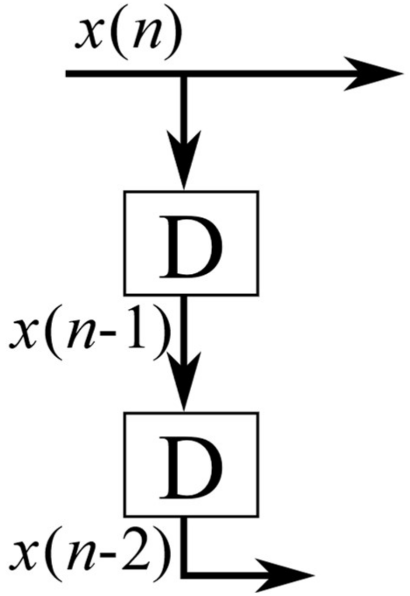
Realizzazione del Filtro IIR

Realizzazione filtro IIR

Il modo più semplice per realizzare una sezione del secondo ordine parte dalla relazione che definisce il sistema:

$$y(n)+a_1 y(n-1)+a_2 y(n-2)=b_0 x(n)+b_1 x(n-1)+b_2 x(n-2)$$

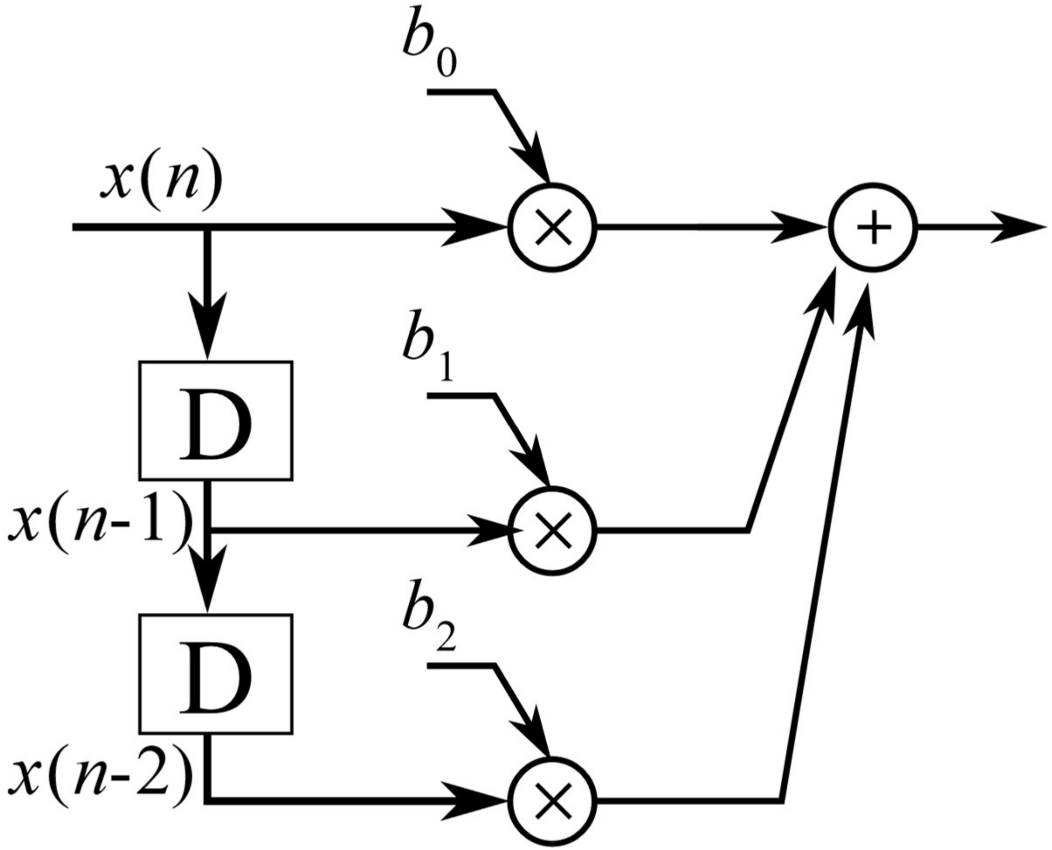
Il secondo membro di questa eguaglianza mostra che abbiamo bisogno di $x(n\text{-}1)$ ed $x(n\text{-}2)$. Per ottenere questi valori utilizziamo due registri:



Realizzazione filtro IIR

$$y(n) + a_1 y(n-1) + a_2 y(n-2) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2)$$

Dobbiamo poi moltiplicare $x(n) \times b_0$, $x(n\text{-}1) \times b_1$ ecc. e sommare i risultati:



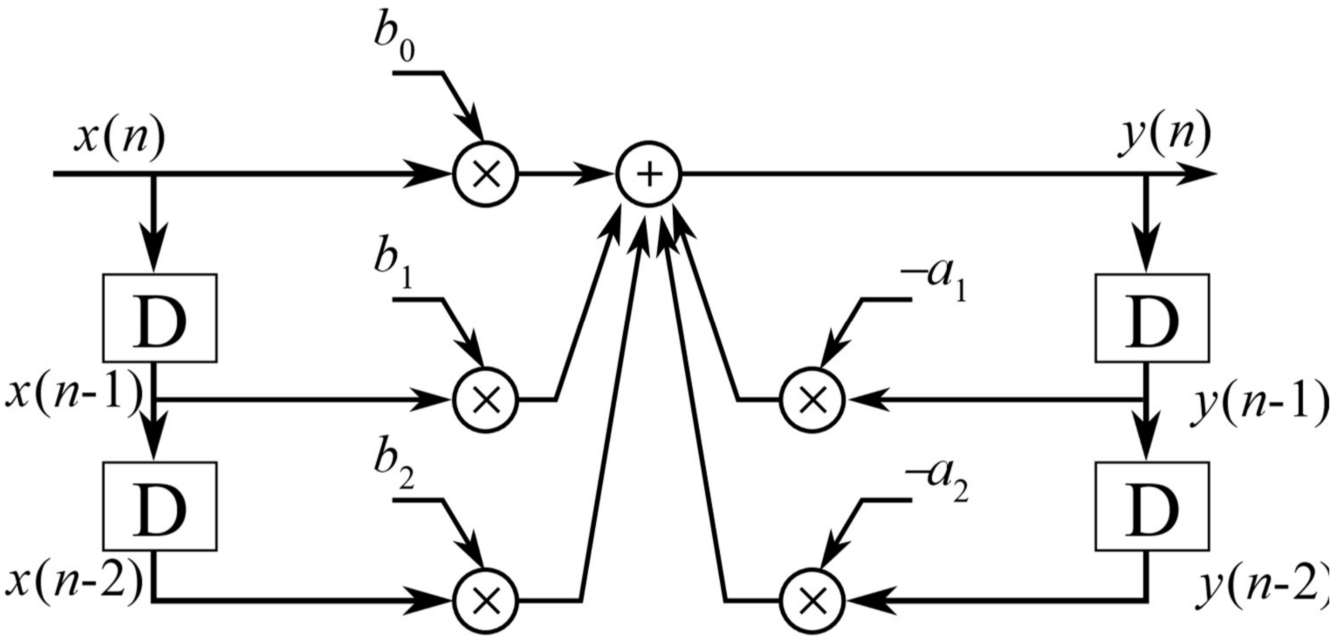
Realizzazione filtro IIR

$$y(n)+a_1y(n-1)+a_2y(n-2)=b_0x(n)+b_1x(n-1)+b_2x(n-2)$$

Ri-arrangiamo quest'ultima relazione:

$$y(n)=b_0x(n)+b_1x(n-1)+b_2x(n-2)-a_1y(n-1)-a_2y(n-2)$$

Al valore ottenuto in precedenza dobbiamo quindi sottrarre: $a_1 \times y(n\text{-}1)$ e: $a_2 \times y(n\text{-}2)$. A tal fine ci servono due registri (per ottenere $y(n\text{-}1)$ e $y(n\text{-}2)$) e due moltiplicatori:



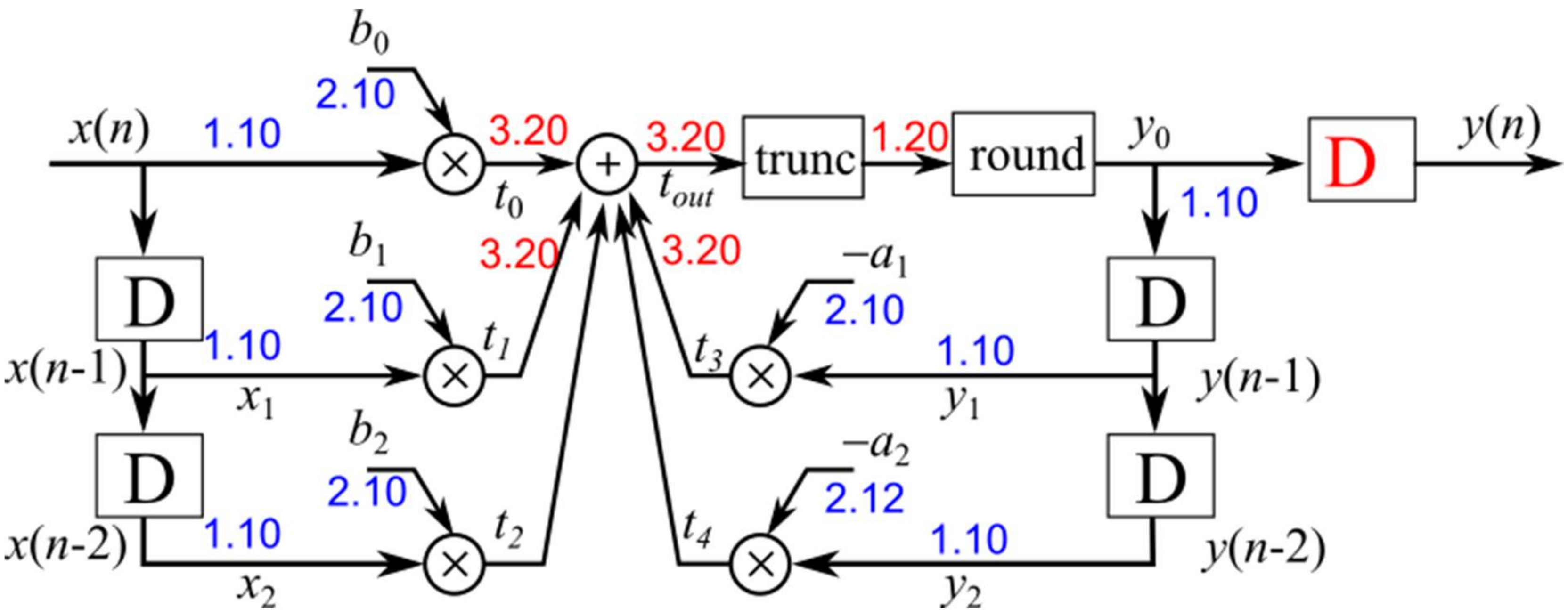
Rappresentazione dei segnali

Rappresentazioni utilizzate (ricordiamo che abbiamo scalato opportunamente i coefficienti in modo da essere sicuri che l'uscita y sia rappresentabile come l'ingresso x):

x => Q1.10

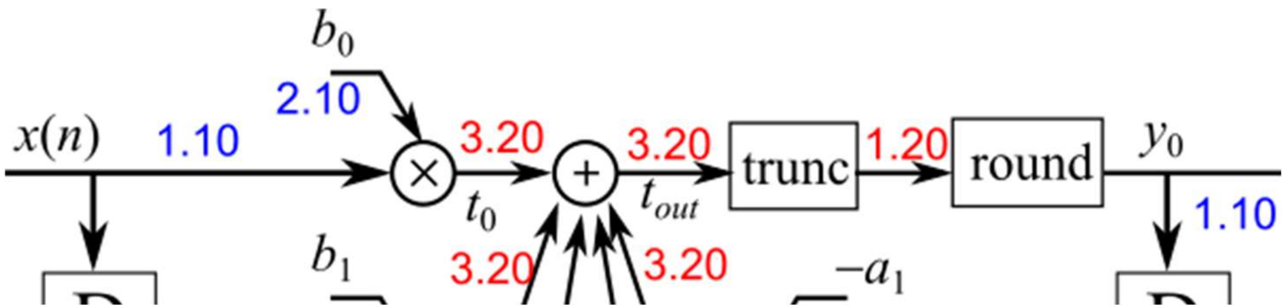
y => Q1.10

a_i, b_i => Q2.10



Rappresentazione dei segnali

Osserviamo che l'uscita dei moltiplicatori e dell'addizionatore ha rappresentazione 3.20 (differente da quella di y):

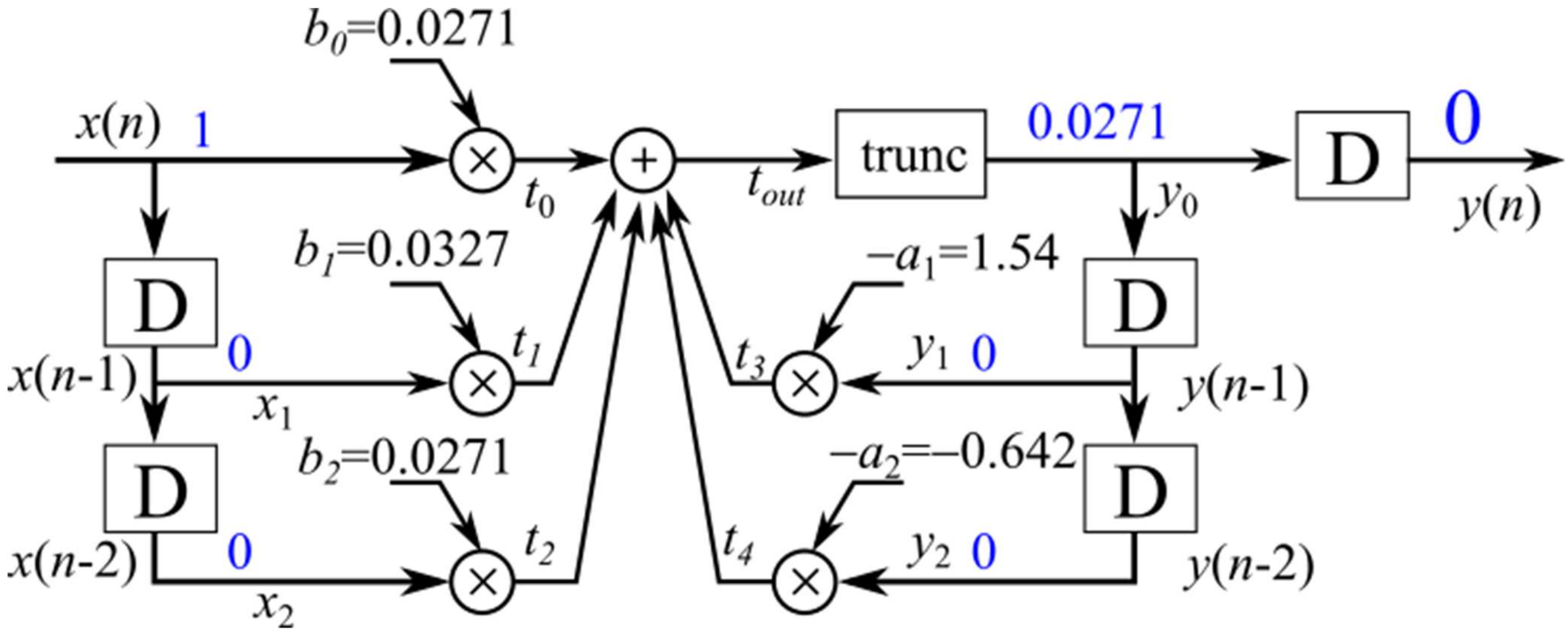


Noi sappiamo *a priori* (grazie allo scaling dei coefficienti) che l'uscita è correttamente rappresentabile nel formato Q1.10. Possiamo quindi procedere ad un troncamento ed ad un arrotondamento.

- Il troncamento interesserà i 2 bit a sinistra di t_{out} (vedi figura)
- L'arrotondamento verrà effettuato in modo che: LSB=2^{-10}

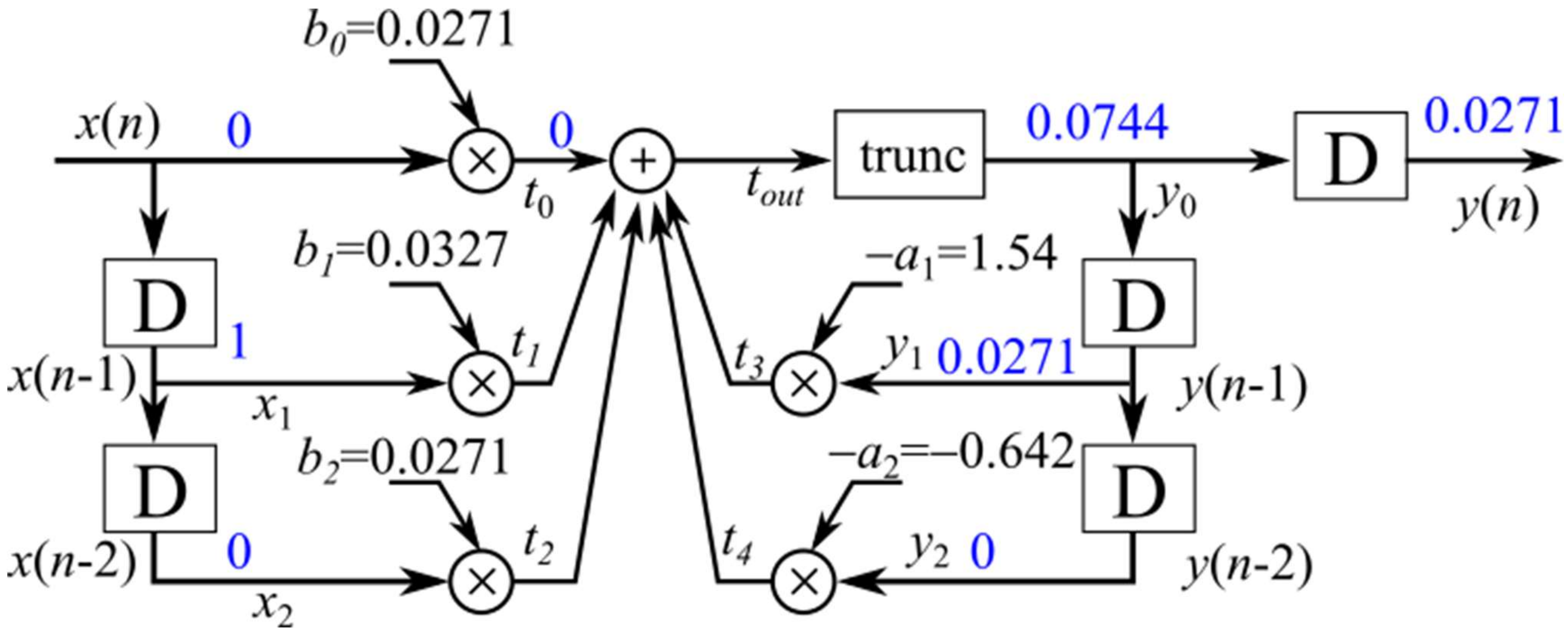
Risposta impulsiva

Proviamo a simulare la risposta impulsiva del filtro.
La figura mostra i valori dei vari segnali, dopo il reset, assumendo che l'ingresso x sia 1 per un ciclo di clock, per poi portarsi a zero:



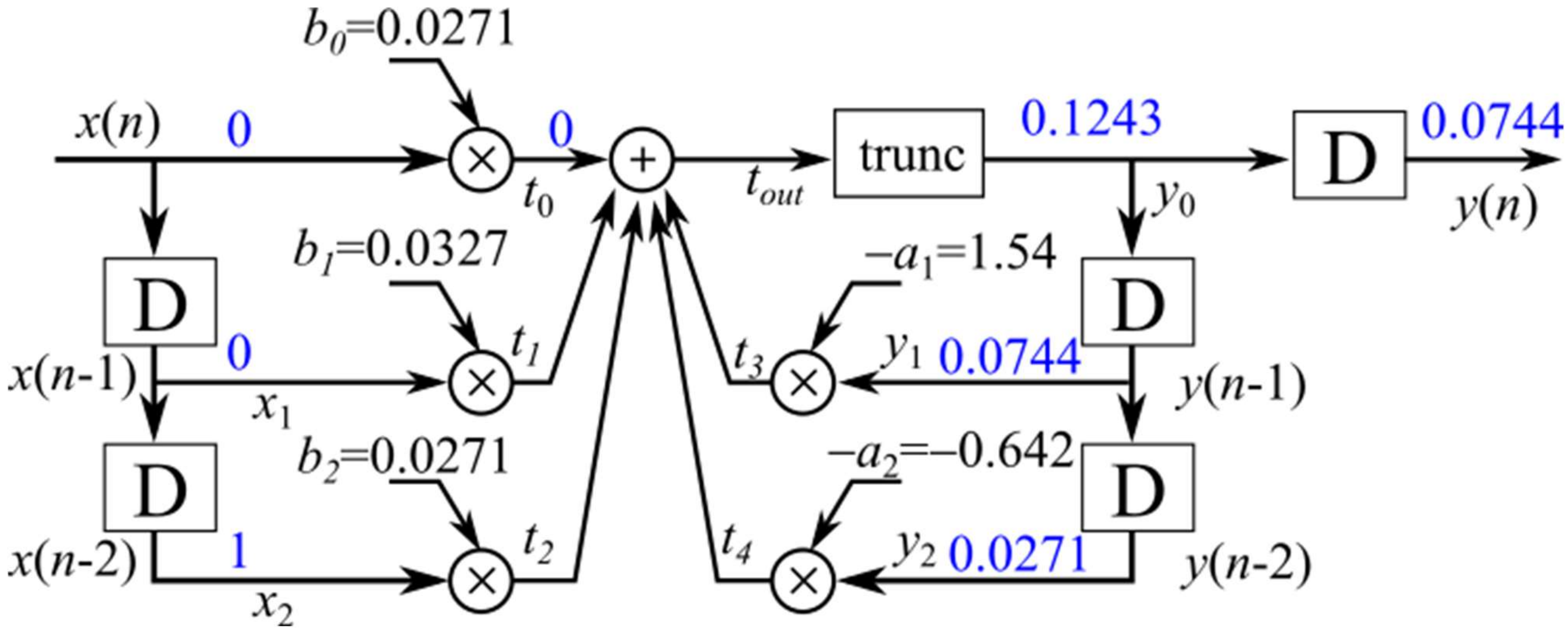
Risposta impulsiva

Situazione dopo il successivo ciclo di clock. La prima uscita è: 0.0271



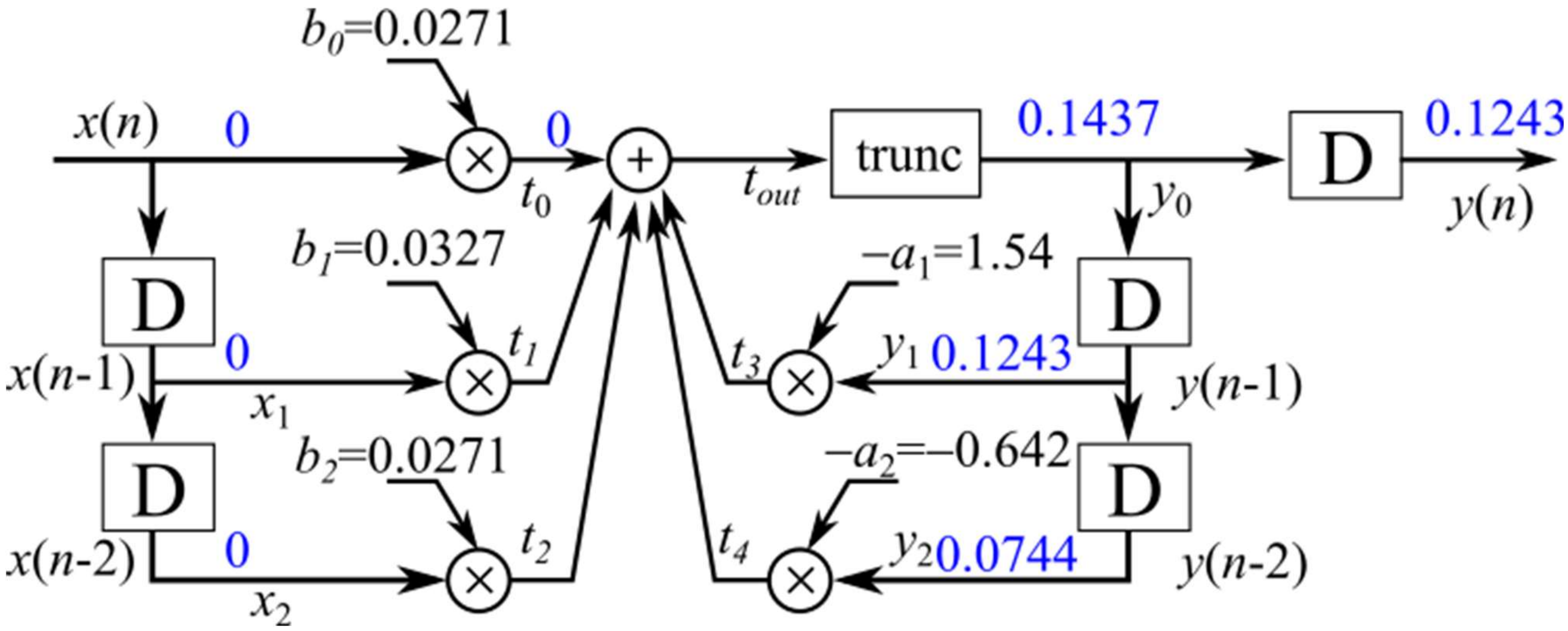
Risposta impulsiva

Situazione dopo il successivo ciclo di clock



Risposta impulsiva

Situazione dopo il successivo ciclo di clock



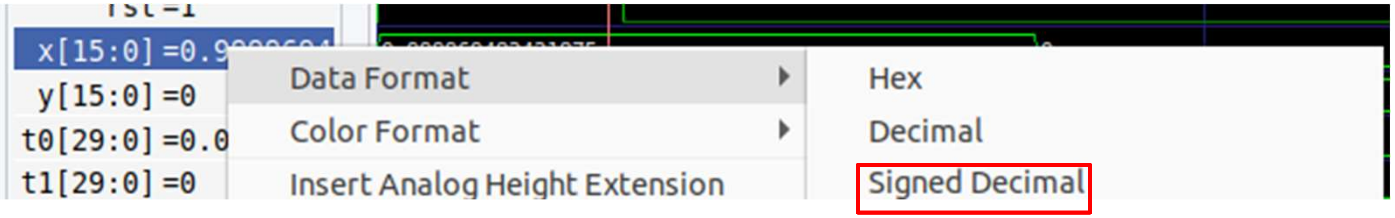
Risposta impulsiva

Dunque, i primi campioni della risposta impulsiva della prima sezione del filtro sono (all'incirca – i valori sono approssimati):
0.0271, 0.0744, 0.1243, 0.1437

Note per il visualizzatore di segnali

Con il visualizzatore di forma d'onda è possibile visualizzare i valori numerici dei segnali corrispondenti al formato a virgola fissa. Consideriamo ad esempio il segnale y, rappresentato nel formato **Q1.10**. Selezioniamo il segnale nel visualizzatore di forme d'onda con *il tasto destro* del mouse, quindi scegliamo:

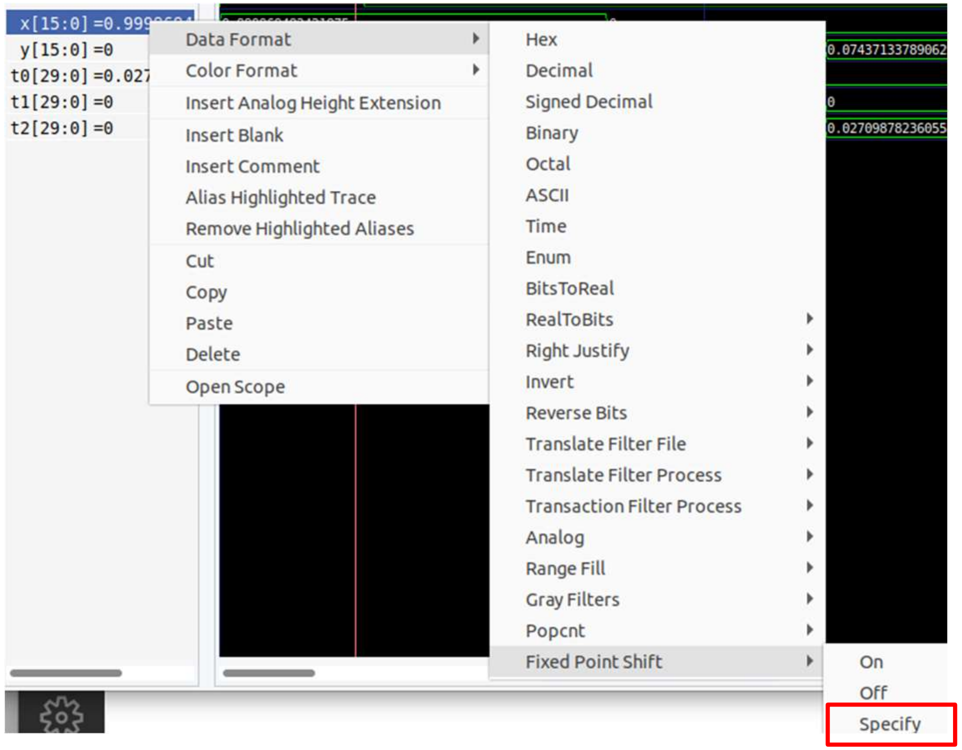
Data Format > Signed Decimal



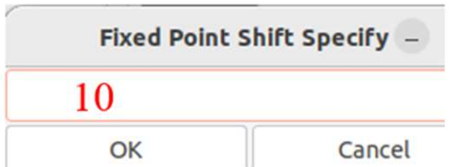
continua nella slide seguente...

Note per il visualizzatore di segnali

Selezioniamo nuovamente il segnale con *il tasto destro* del mouse, e scegliamo:
Data Format > Fixed Point Shift > Specify



Compare una finestra che ci chiede di inserire il numero di cifre decimali. Nel nostro caso (formato **1.10**) inseriamo 10 e premiamo OK.



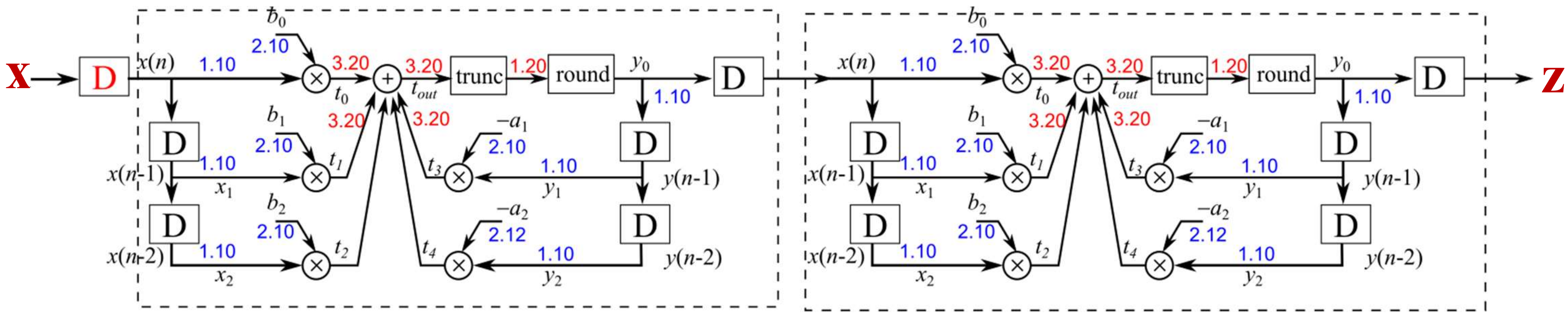
Filtro complessivo

Il filtro complessivo sarà costituito dalla cascata di due blocchi aventi la medesima struttura, in cui cambierà solo il valore dei coefficienti a_i e b_i

Notare la presenza di un registro di ingresso.

Nota importante:

- il segnale di ingresso del modulo si chiamerà **x**
- il segnale di uscita del modulo si chiamerà **z**



Filtro IIR: note operative

Usiamo:

- reset sincrono.
- nome del segnale di reset: **rst**
- clock attivo sul fronte di salita
- nome del segnale di clock: **clk**
- nome del segnale di ingresso: **x**
- nome del segnale di uscita: **z**

Vi verranno forniti:

- I valori dei coefficienti

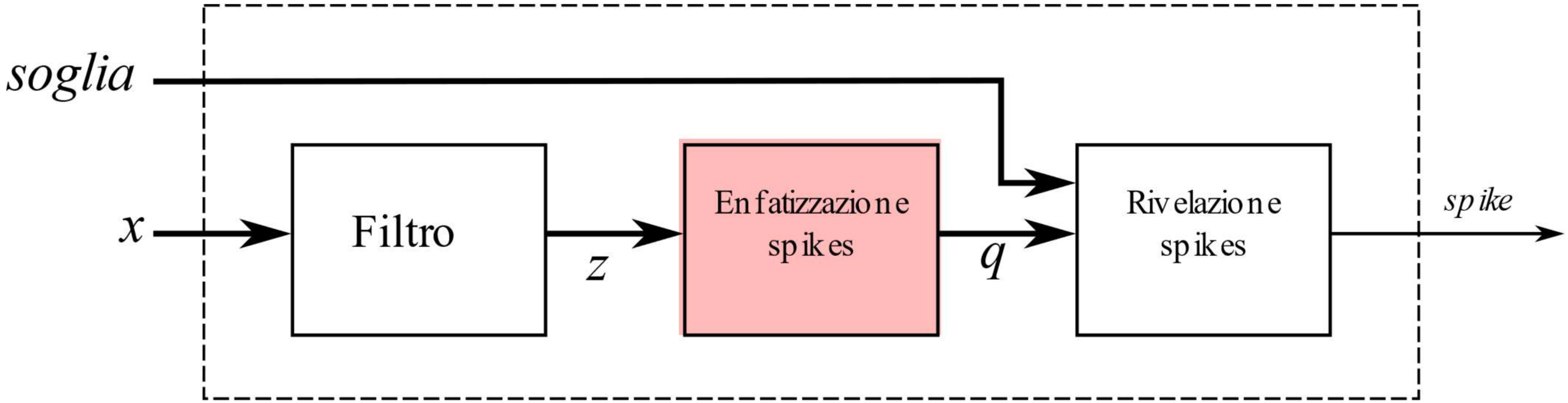
File di testo con le uscite attese per:

- Risposta impulsiva della prima sezione del filtro
- Risposta impulsiva della seconda sezione del filtro
- Risposta impulsiva complessiva, prodotta dalla cascata delle due sezioni.

Enfatizzazione degli spikes

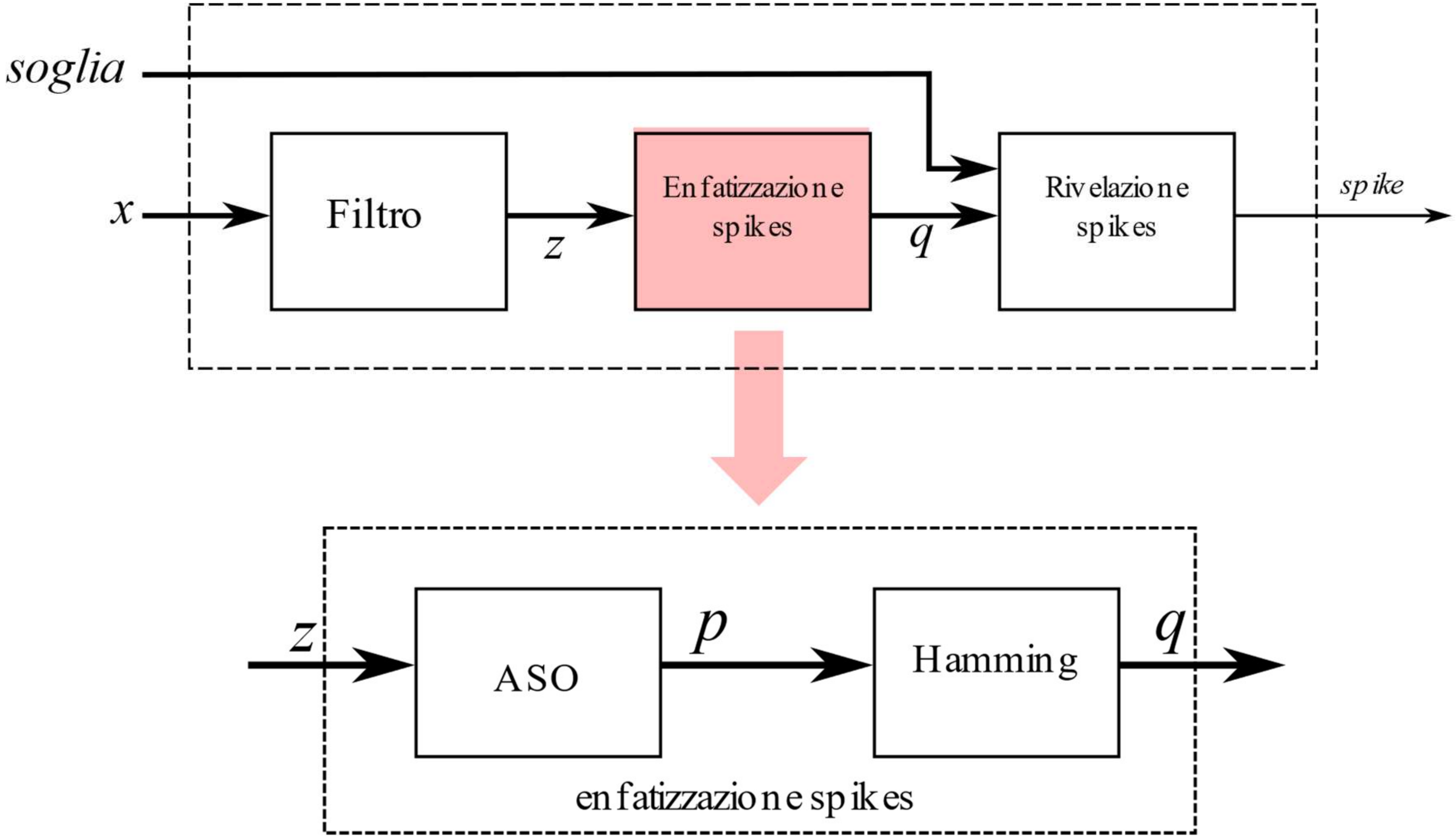
Enfatizzazione degli spikes

Il secondo elemento del nostro sistema è il circuito che enfatizza gli impulsi.



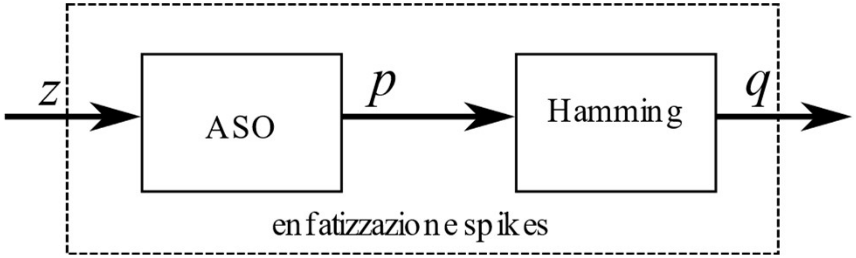
Il circuito implementa il cosiddetto *smoothed Amplitude Slope Operator (ASO) – smoothed ASO* ed è costituito a sua volta dalla cascata di due sottoblocchi.

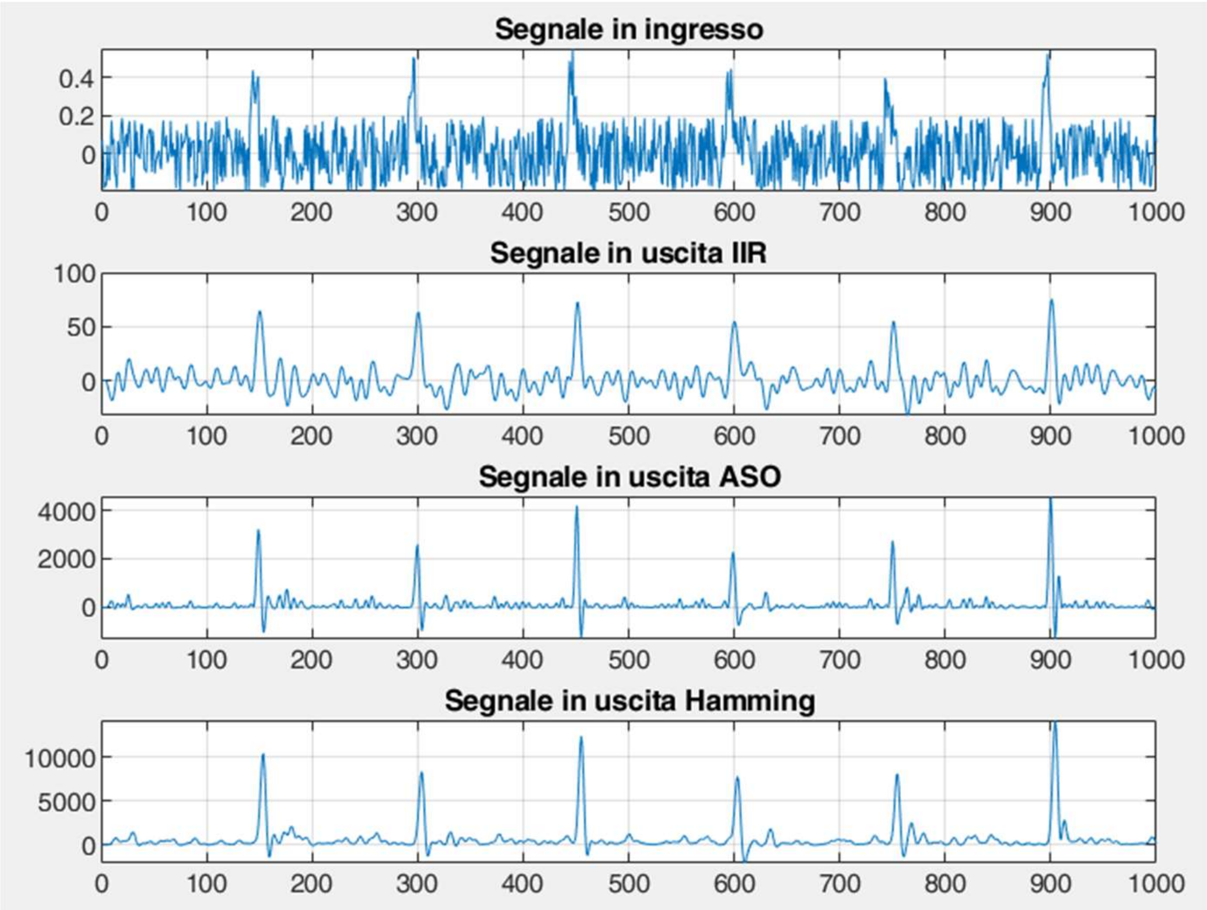
Smoothed ASO



Il primo blocco implementa l'operatore ASO, mentre il secondo effettua un filtraggio con una finestra di Hamming.

Smoothed ASO



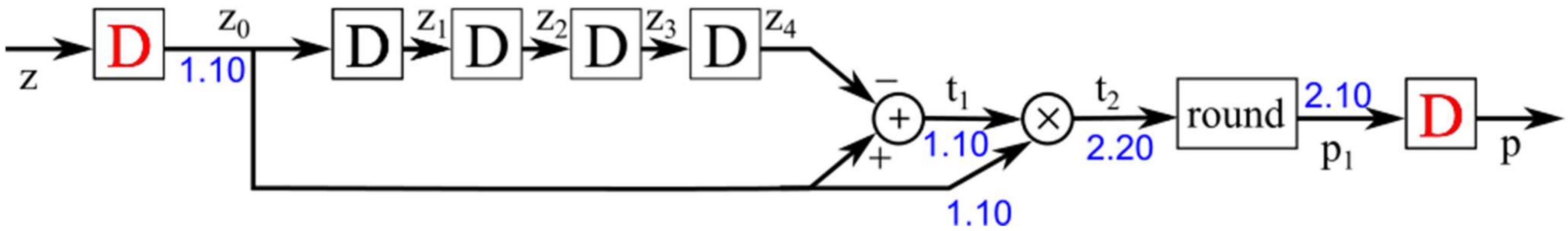


Operatore ASO

L'operatore ASO, a partire dalla sequenza $z(n)$, genera l'uscita $p(n)$ nel modo seguente:

$$p(n) = z(n)\times[z(n) - z(n-k)]$$

In questa equazione k è un parametro (da scegliere in base alla durata attesa degli spikes ed alla frequenza di campionamento); noi useremo k=4. L'implementazione hardware è molto semplice, come mostra la Figura seguente. Si noti l'arrotondamento finale che porta l'uscita nel formato Q2.10



Da notare anche i registri inziali e finali, in rosso.

La timing analysis di questo blocco non avrà cammini in-to-reg o reg-to-out

Smoothed ASO

L'uscita $p(n)$ dell'operatore ASO viene filtrata con un filtro FIR, che realizza una funzione del tipo:

$$q(n) = b_0\, p(n) + b_1\, p(n-1) + b_2\, p(n-2) + \ldots .\ b_L\, p(n-L+1)$$

I coefficienti b_i corrispondono ad una opportuna "finestra" di lunghezza L; noi sceglieremo una finestra d: Hamming con L=8.
Da matlab i coefficienti si ottengono con il comando: `hamming(8)` I valori sono simmetrici (il primo è uguale all'ultimo, il secondo al penultimo ecc.)

```
>> hamming(8)
ans =
 0.0800
 0.2532
 0.6424
 0.9544
 0.9544
 0.6424
 0.2532
 0.0800
```

Scaling e quantizzazione dei coefficienti

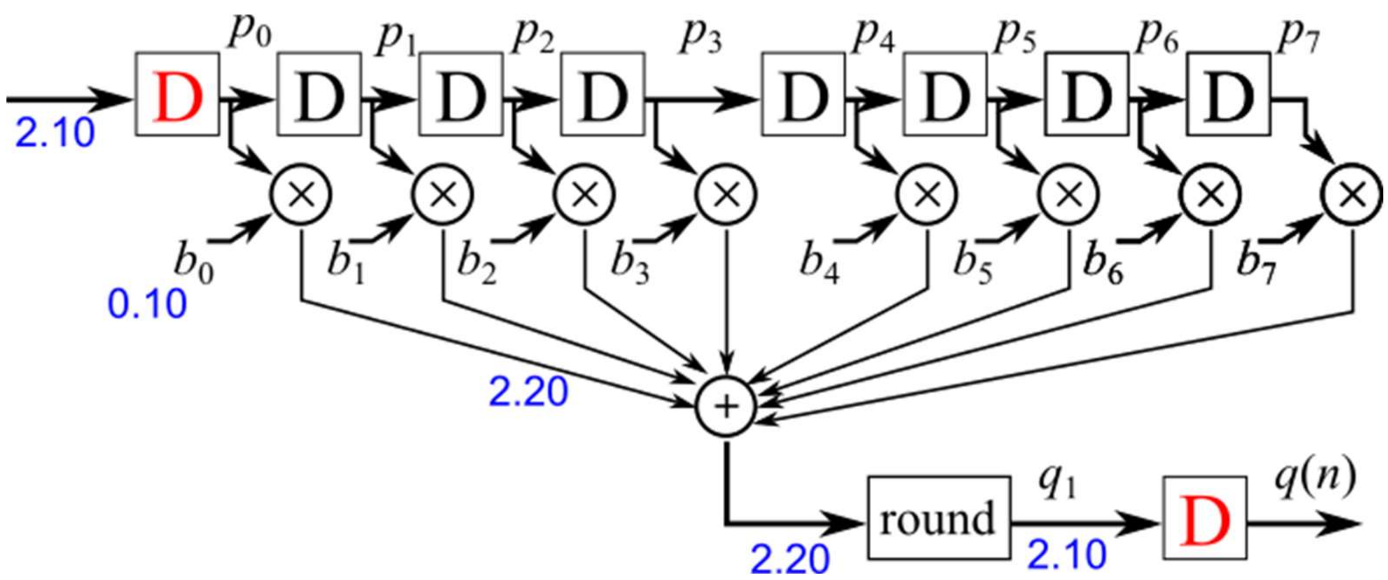
Anche per il filtro FIR scaliamo i coefficienti, in modo che l'uscita del filtro sia rappresentabile allo stesso modo dell'ingresso e li quantizziamo. Complessivamente il formato è **Q0.10** (tutti i coefficienti sono minori di uno)

```
Coefficienti quantizzati del filtro FIR.
b1:
0000010101     0.0205078125000
b2:
0001000011     0.0654296875000
b3:
0010101010     0.1660156250000
b4:
0011111101     0.2470703125000
b5:
0011111101     0.2470703125000
b6:
0010101010     0.1660156250000
b7:
0001000011     0.0654296875000
b8:
0000010101     0.0205078125000
```

Realizzazione filtro FIR

La realizzazione del filtro FIR può essere ottenuta calcolando i termini: $p(n-1)$, $p(n-2)$ ecc. con dei registri, realizzando le moltiplicazioni per i coefficienti b_i e sommando infine i prodotti. La Figura seguente mostra la struttura del sistema (analoga alla "prima metà" del filtro IIR).

I coefficienti sono in formato Q0.10; il risultato finale è arrotondato al formato Q2.10. Notare i registri di ingresso e uscita.



Blocco di Enfatizzazione: note operative

Usiamo:

- reset sincrono.
- nome del segnale di reset: **rst**
- clock attivo sul fronte di salita
- nome del segnale di clock: **clk**
- nome del segnale di ingresso: **z**
- nome del segnale intermedio (uscita del blocco ASO): **p**
- nome del segnale di uscita: **q**

Vi verranno forniti:

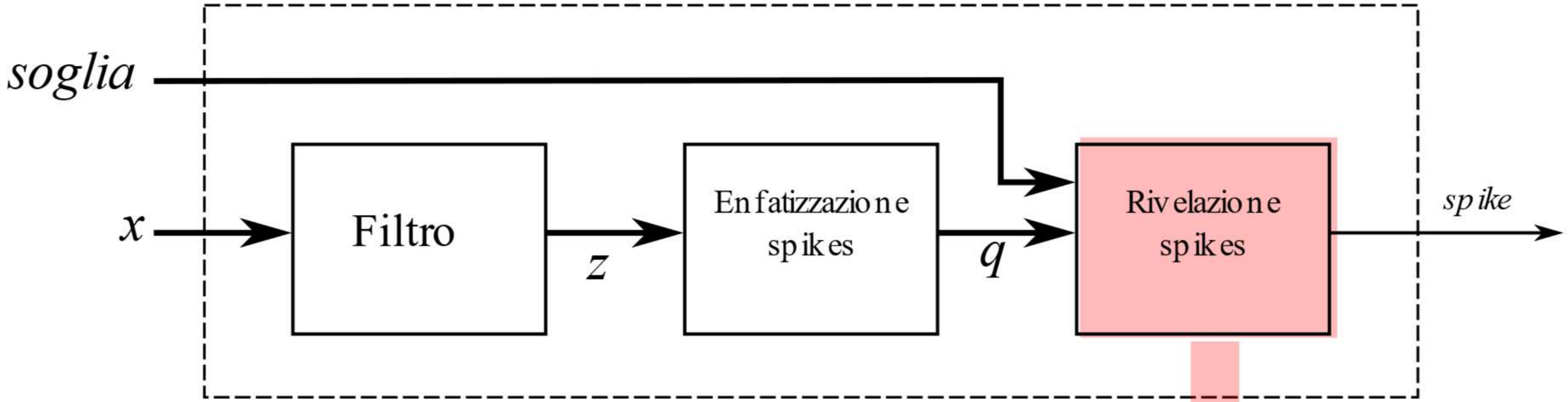
- I valori dei coefficienti

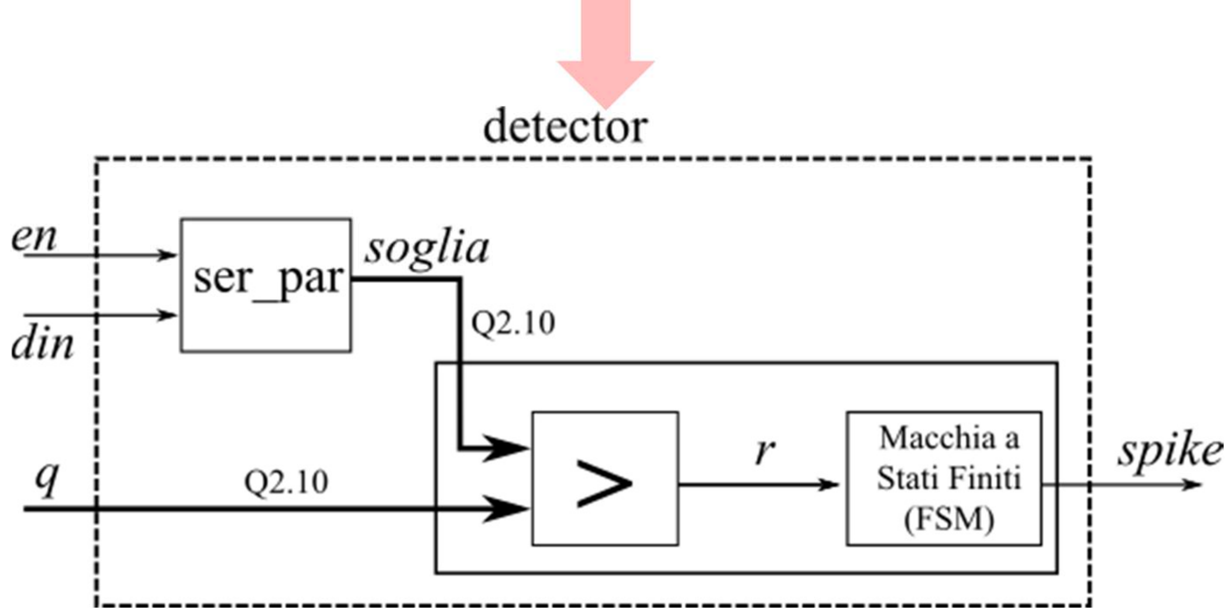
  File di testo con le uscite attese per i due sottoblocchi.

Rivelazione degli spikes

Rivelazione degli spikes

L'ultimo elemento del nostro sistema è il circuito che rivela la presenza degli spikes.

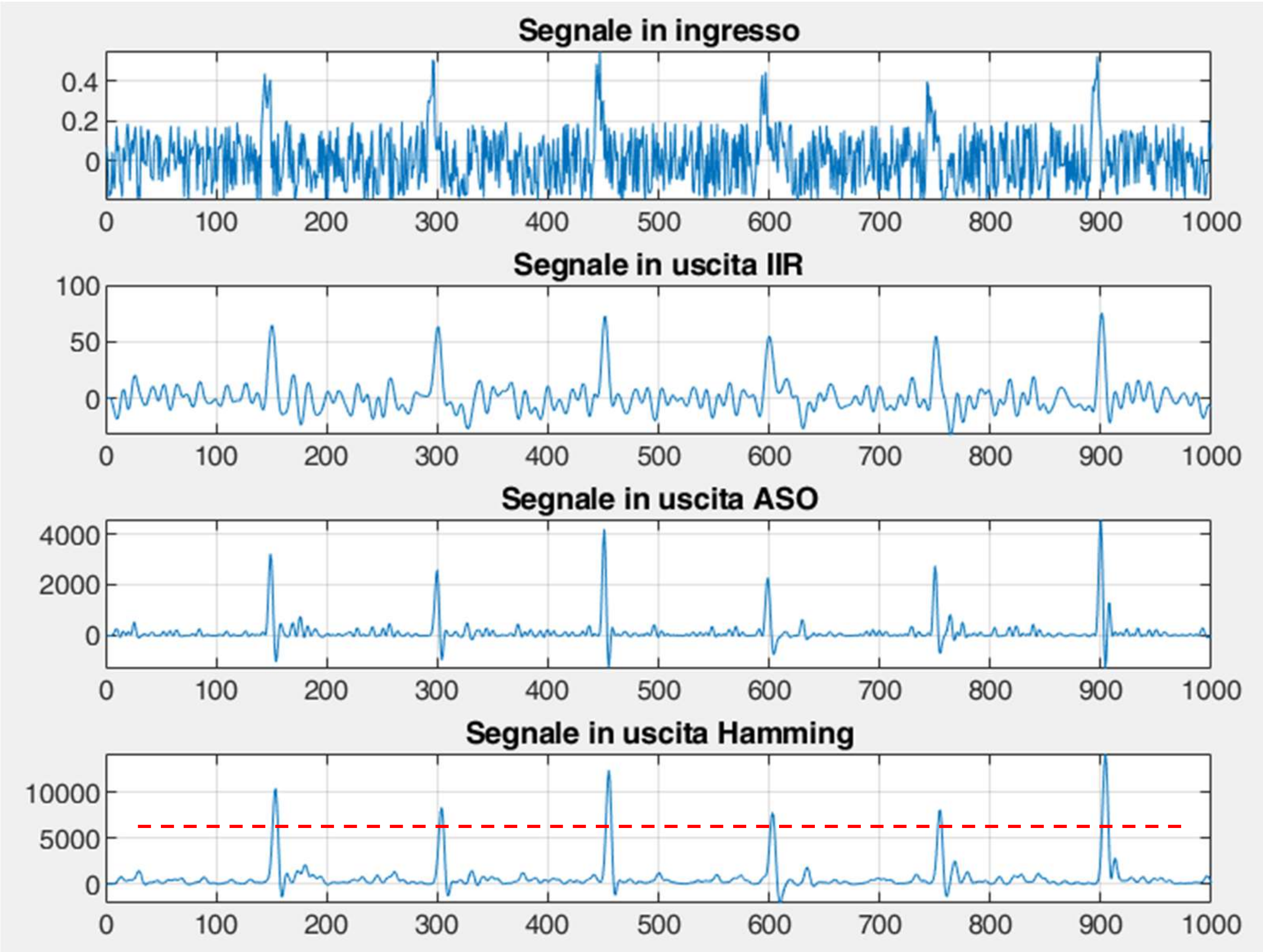




Il detector include:

- un convertitore seriale/parallelo per il caricamento della soglia
- un comparatore (che determina se il segnale q è maggiore della soglia) e una macchina a stati finiti (FSM)

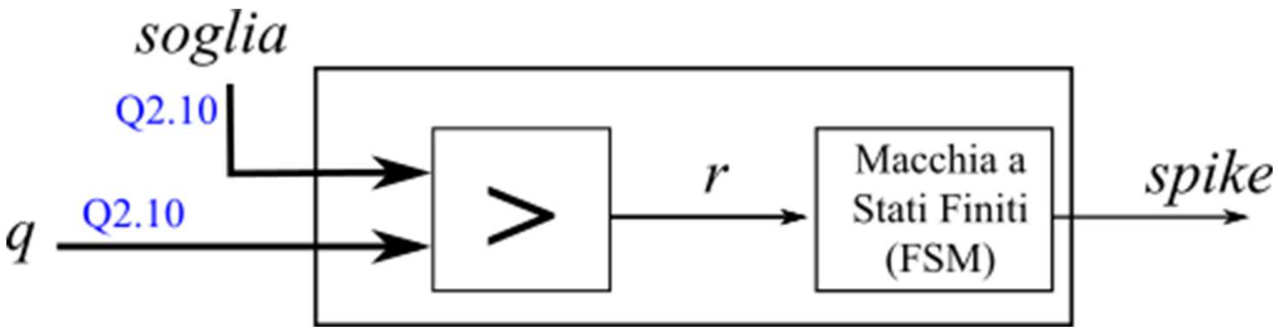
Scelta della soglia



Sulla base di una simulazione complessiva del sistema, sceglieremo per la soglia il valore:
7×10^{-3}
Questo valore è fornito come ingresso al circuito e deve ovviamente essere rappresentato nello stesso formato del segnale q, ovvero Q2.10

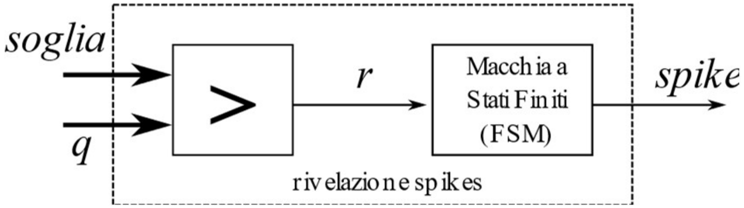
FSM

La macchia a stati finiti (Finite State Machine, FSM) è introdotta per limitare erronee identificazioni degli spikes dovute alla presenza di rumore additivo sul segnale *q*.

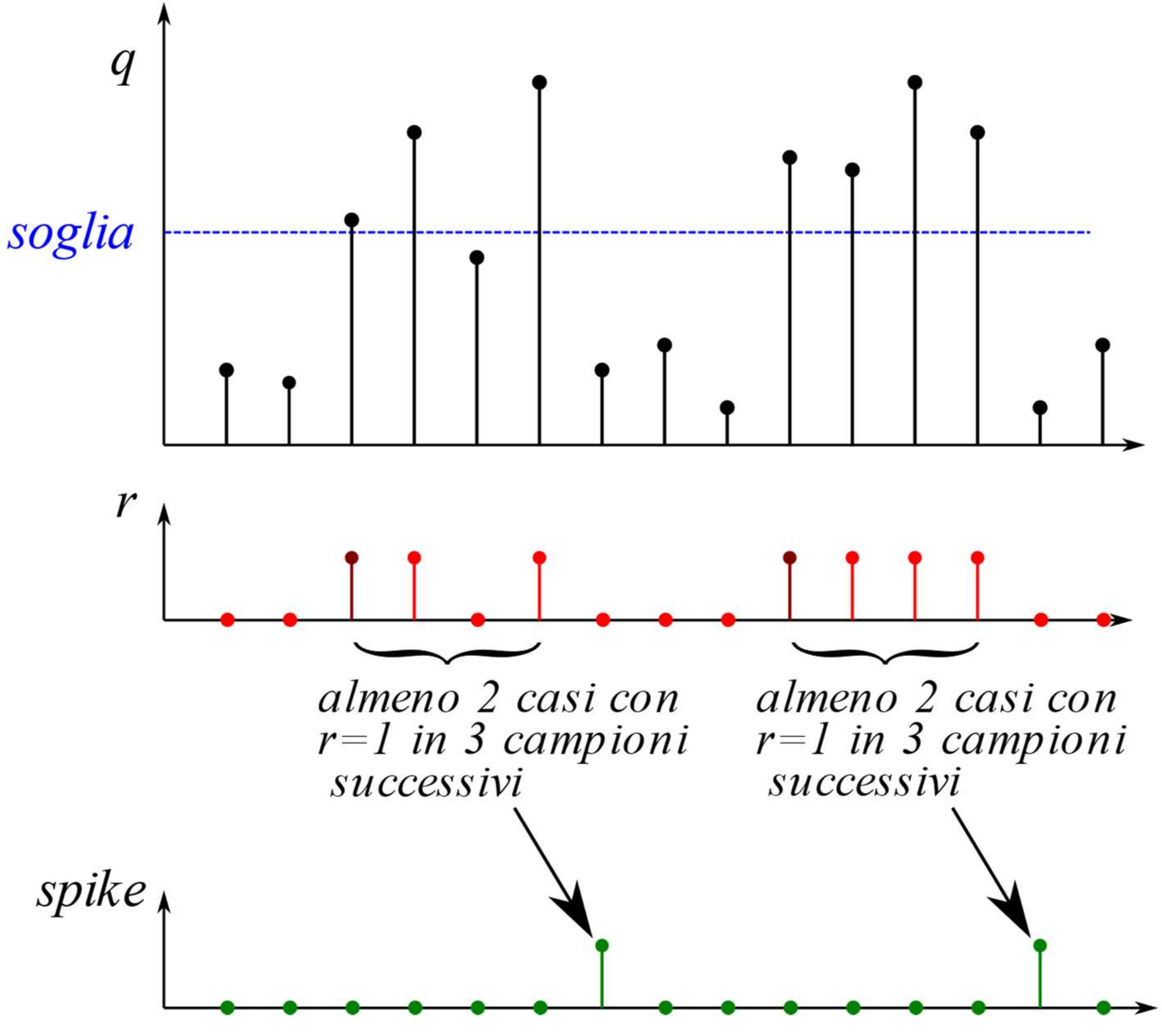


La figura nella slide seguente mostra il funzionamento della FSM: il segnale di uscita *spike* si attiva se, dopo aver individuato un '1' nel segnale *r*, lo stesso segnale rimane alto almeno due volte nei tre campioni successivi.

FSM



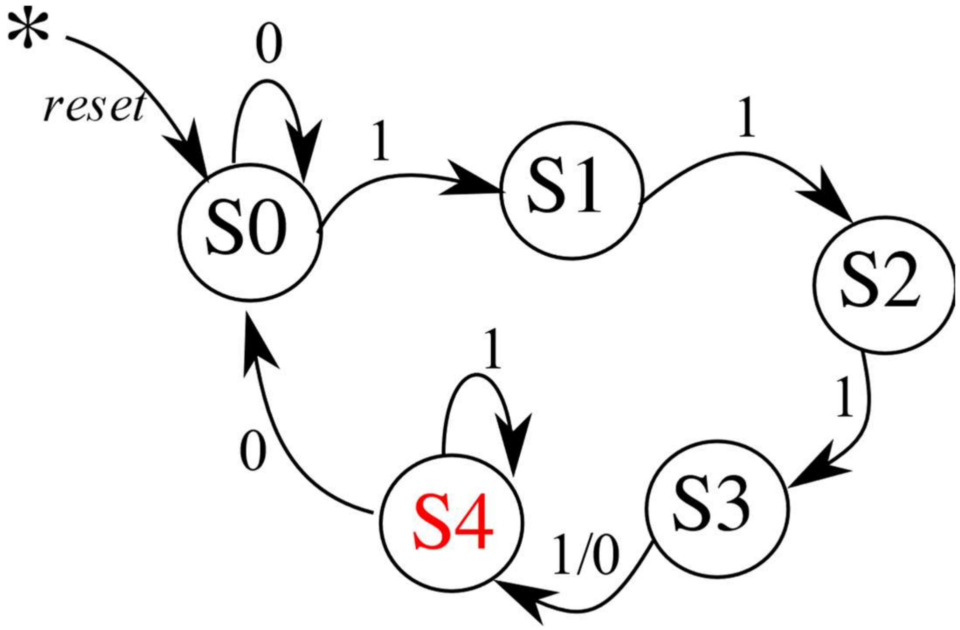
soglia
q
>
r
Macchia a Stati Finiti (FSM)
spike
rivelazione spikes



q
soglia
r
almeno 2 casi con r=1 in 3 campioni successivi
almeno 2 casi con r=1 in 3 campioni successivi
spike

FSM

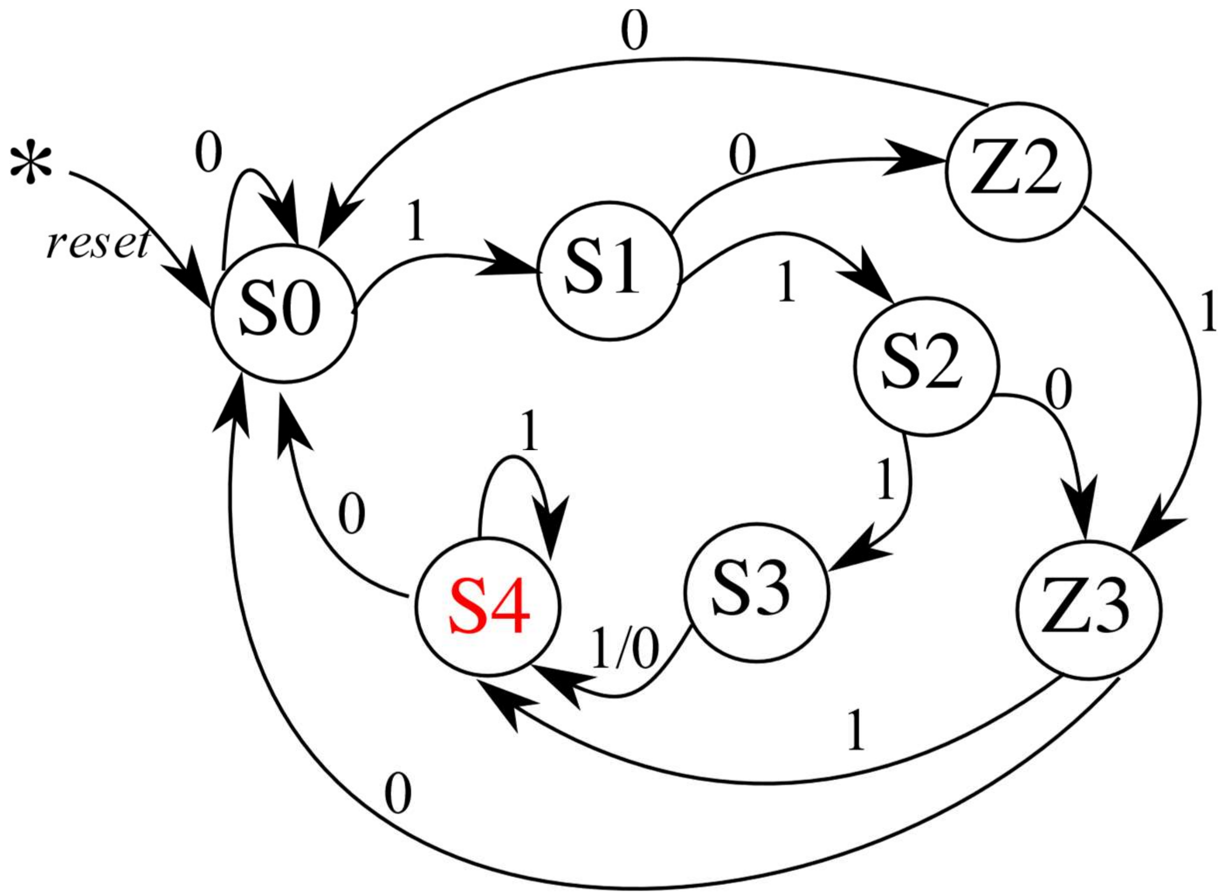
Cominciamo a tracciare una prima bozza del diagramma di stato della FSM:



Dopo il reset iniziale, la FSM si porta nello stato S0 e vi resta fino a quando non si ha *r*=1.
Quando *r*=1, la FSM raggiunge lo stato S1 e, se l'ingresso *r* continua ad essere 1, si porta negli stati S2 ed S3. Lo stato S3 viene raggiunto per questa via solo se sono stati rivelati due '1' consecutivi (oltre il primo). Pertanto, indipendentemente dall'ingresso, il sistema si porta in S4, in corrispondenza del quale l'uscita *spike* verrà attivata.
Quando *r* si abbassa la FSM ritorna in S0.

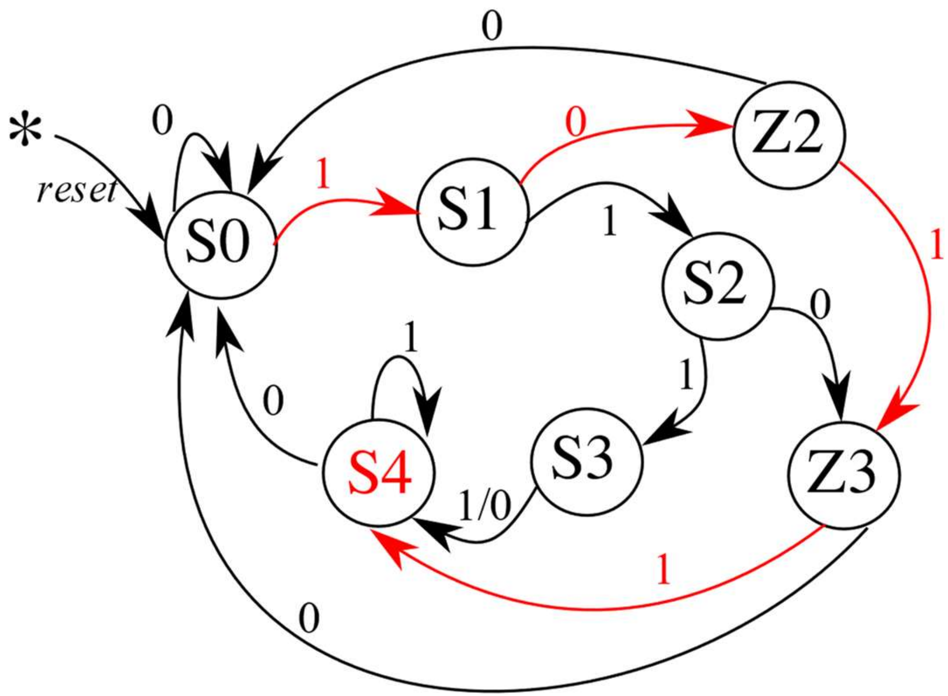
FSM

Completiamo il diagramma di stato, con l'introduzione degli stati Z2 e Z3 che consentono di attivare l'uscita *spike* in tutti i casi in cui *r* è alto almeno altre due volte nei tre campioni successivi:

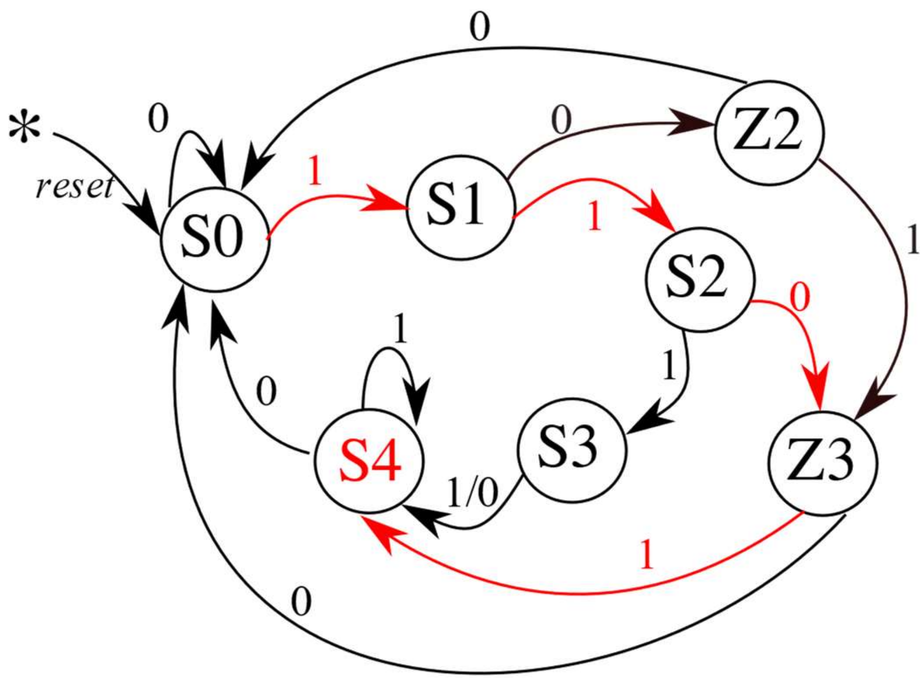


FSM

Esempi di attivazione:

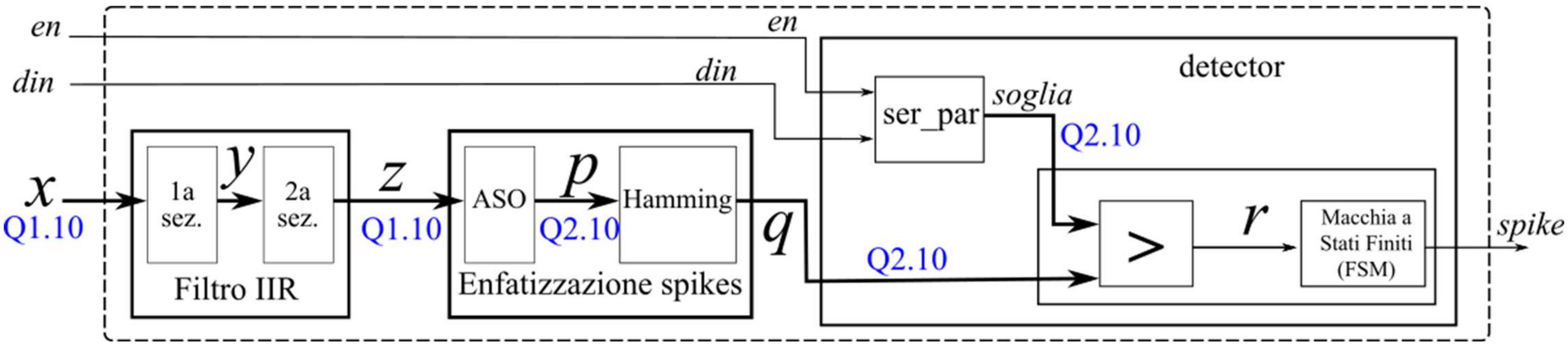


Sequenza: 1 0 1 1



Sequenza: 1 1 0 1

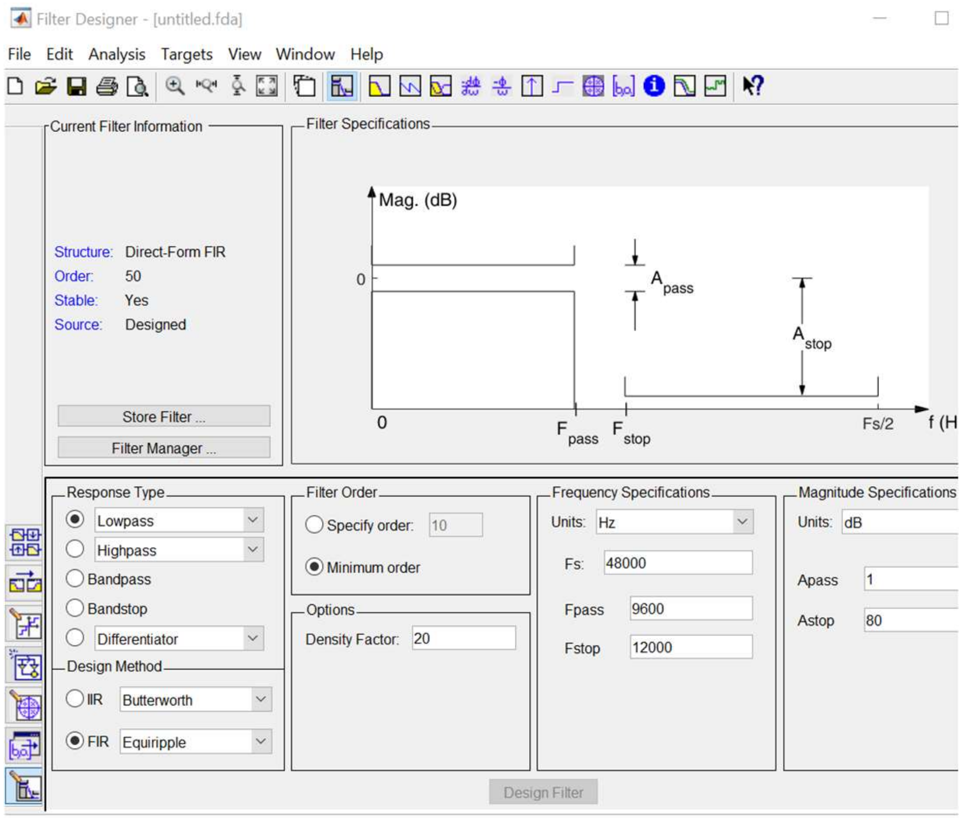
Sistema Complessivo



Appendice
Uso di Matlab per il calcolo e la scalatura dei coefficienti del filtro

Coefficienti del Filtro IIR

Per il calcolo dei coefficienti del filtro ci affideremo a matlab.
La funzione da utilizzare è: `filterDesigner` che è contenuta nel *signal processing and communications toolbox*.



Type: lowpass
Design Method: IIR elliptic
Filter order: 4
Fs: 100MHz
Fpass: 10MHz
Apass: 1dB
Astop: 60dB

Coefficienti del Filtro IIR

La figura seguente mostra un esempio dei risultati prodotti da matlab

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

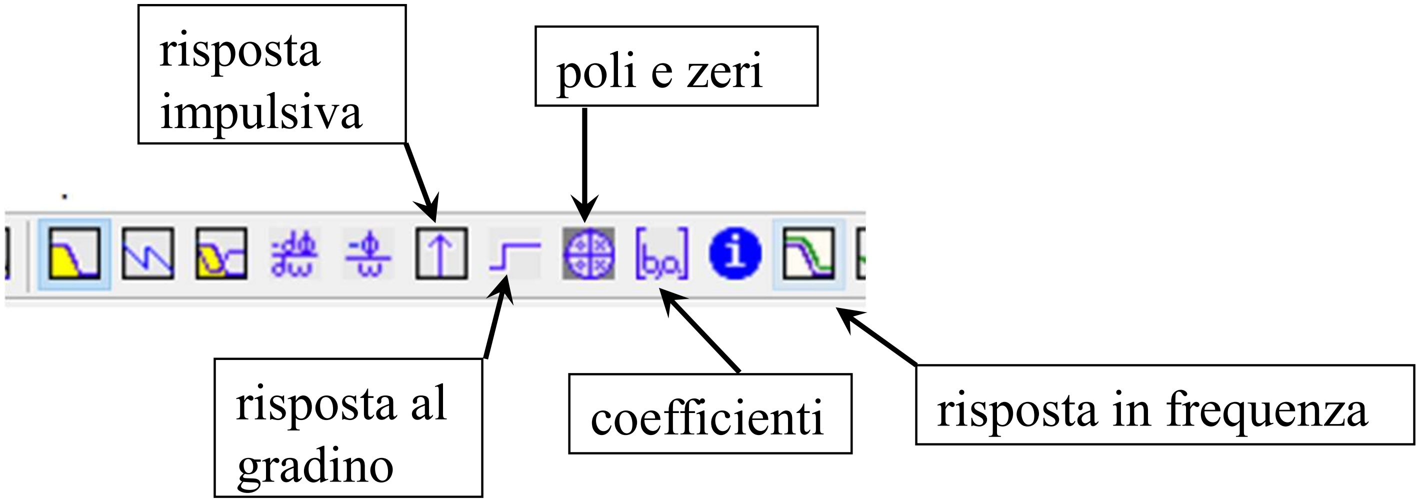
```
% Generated by MATLAB(R) 9.8 and Signal Processing Toolbox 8.4.
% Generated on: 02-Oct-2020 10:59:55

% Coefficient Format: Decimal

% Discrete-Time IIR Filter (real)
% -------------------------------
% Filter Structure    : Direct-Form II, Second-Order Sections
% Number of Sections  : 2
% Stable              : Yes
% Linear Phase        : No


SOS Matrix:
1  1.6347118068676917967962936018011532723 9   1  1  -1.28684372812905545302
1  0.5807391836216522218450108994147740304 47  1  1  -1.07047460462535504177
```

Coefficienti della prima sezione del filtro:
b_0=1; b_1=1.6347; b_2=1; a_0=1; a_1= −1.2868…

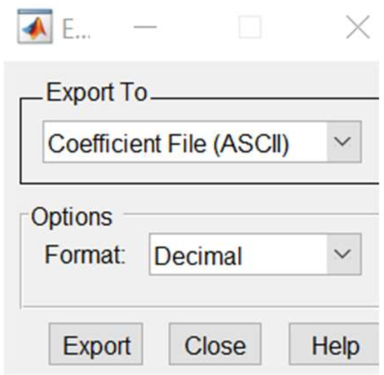
Coefficienti della seconda sezione del filtro:
b_0=1; b_1= 0.5807; b_2=1; a_0=1; a_1= −1.0704…

Coefficienti del Filtro IIR

Completato il calcolo dei coefficienti (richiede pochi secondi), potete usare le icone sulla parte superiore della finestra per analizzare i risultati:



Per salvare i risultati dal menù:
`File => Export`
esportate i valori in un file di testo



Scaling dei coefficienti

Consideriamo la funzione di trasferimento di una sezione del secondo ordine:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

Possiamo moltiplicare la funzione di trasferimento per una costante K (detta *fattore di scala*) cambiando così il guadagno del filtro (senza modificare le frequenze di taglio):

$$H(z) = \frac{K\left(b_0 + b_1 z^{-1} + b_2 z^{-2}\right)}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

Sceglieremo il fattore K in modo che l'uscita di ognuna delle sezioni del filtro abbia lo stesso range di valori dell'ingresso ed abbia quindi la stessa rappresentazione in virgola fissa dell'ingresso (questa scelta semplifica molto il progetto complessivo).

Scelta del fattore di scala

Osserviamo che:

$$y(n) = \sum_{k=0}^{+\infty} h(k) \cdot x(n-k) \le \sum_{k=0}^{+\infty} |h(k)| \cdot x_{max} = \sigma \cdot x_{max}$$

Assumeremo che l'ingresso x sia compreso nel range [-1, 1), per cui $x_{max}=1$.
Dalla relazione precedente siamo certi che l'uscita y appartiene allo stesso range [-1,1) se:

$$\sigma = \sum_{k=0}^{+\infty} |h(k)| \le 1$$

Scegliamo quindi il fattore di scala in modo da rispettare la condizione $\sigma<1$.
In questo modo potremo assumere sia per x che per y lo stesso formato.
In particolare, assumeremo il formato: **Q1.10**

Scelta del fattore di scala

Operativamente:

- Partiamo dai coefficienti calcolati da matlab per la prima sezione del filtro: b_0=1; b_1=1.6347; b_2=1; a_0=1; a_1= –1.2868…
- Calcoliamo la risposta all'impulso $h(n)$ (limitandoci, ad esempio, ai primi 100 campioni)
- Calcoliamo: $S = \sum_{k=0}^{+\infty} |h(k)|$
- Calcoliamo i coefficienti b finali (gli a restano invariati): $b_{0f}=b_0$/S; $b_{1f}=b_1$/S; $b_{2f}=b_2$/S;
- Ripetiamo lo stesso procedimento per la seconda sezione del filtro.

Quantizzazione dei coefficienti

In hardware dobbiamo utilizzare anche per i coefficienti una rappresentazione a virgola fissa.

Utilizziamo per i coefficienti la rappresentazione **Q2.10**
Dobbiamo quindi quantizzare i coefficienti, rappresentandoli con $LSB = 2^{-10}$

Scaling e quantizzazione dei coefficienti

Anche per il filtro FIR scaliamo i coefficienti, in modo che l'uscita del filtro sia rappresentabile allo stesso modo dell'ingresso.
Il procedimento è del tutto analogo a quanto visto per il filtro IIR:

- Partiamo dai coefficienti forniti da matlab
- Calcoliamo la risposta all'impulso $h(n)$ (per un filtro FIR questo calcolo è banale….)
- Calcoliamo: $S = \sum_{k=0}^{+\infty} |h(k)|$
- Calcoliamo i coefficienti b finali :
  $b_{0f}=b_0/S$; $b_{1f}=b_1/S$; …..

Dopo lo scaling è necessario quantizzare i coefficienti; assumeremo il formato Q0.10 (tutti i coefficienti sono minori di uno)