

## 4

## L'algoritmo di misura

In questo capitolo viene presentato l'algoritmo di misura implementato sul DSP. Lo schema di principio è quello riportato nel cap. 1, par. 4, sul quale sono state apportate sensibili modifiche per rendere minimo il tempo di calcolo necessario nei singoli stadi, e quindi il peso computazionale dello schema nel suo complesso. In figura viene riportato nuovamente tale schema.

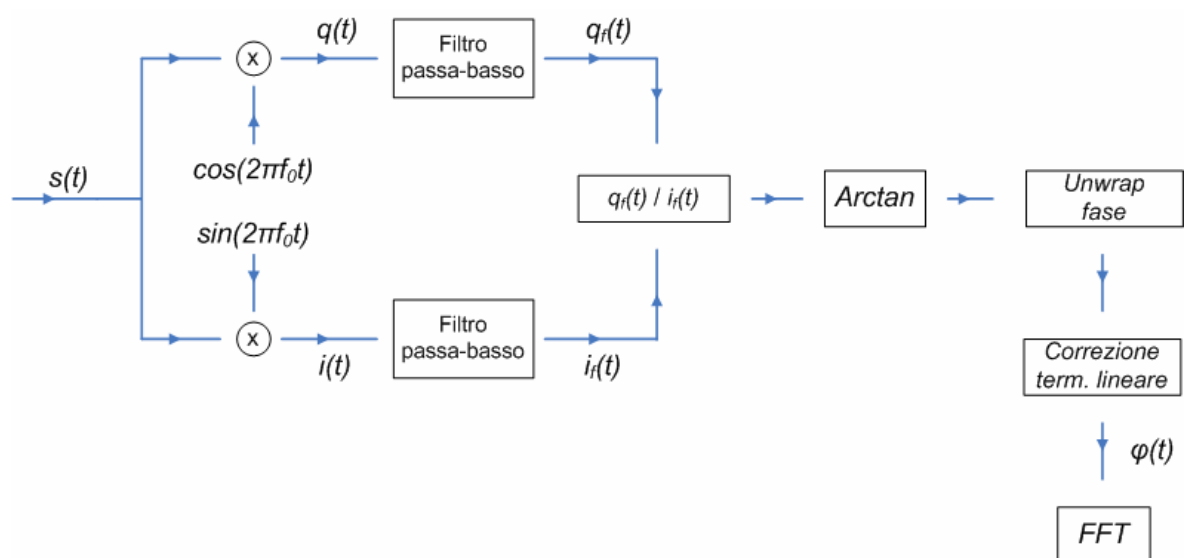


Fig. 4.1 – Schema teorico per l'estrazione del rumore di fase

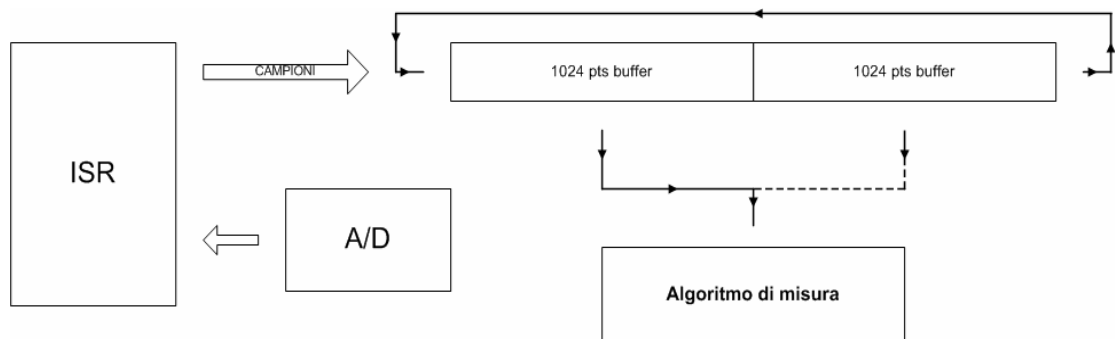
## 4.1 Implementazione per calcolo real-time

L'idea di fondo per l'implementazione real-time del codice è analoga a quella usata nelle tecniche base di computer graphic, ovvero quella di sfruttare il vantaggio competitivo in termini di velocità offerto dal sistema di calcolo rispetto a quello di acquisizione. Difatti, nelle tecniche di animazione grafica, si sfrutta il principio della frequenza di campionamento finita dell'occhio umano, la quale ha un valore indicativo di  $25 \div 30$  Hz, ovvero un periodo pari a 33 ms. Se, in un tempo pari a un quarto di tale valore, si riesce a produrre il prossimo frame da visualizzare, mentre l'utente sta ancora visualizzando il frame corrente, la discretizzazione dovuta al cambio frame non viene affatto avvertita, e l'animazione diventa fluida. Tale possibilità è offerta dai processori grafici, i quali hanno frequenze operative che superano le unità di GHz.

Ispirandosi allo stesso principio, è stata sfruttata la potenza di calcolo offerta dal DSP contro la frequenza di campionamento del sistema di acquisizione dati (DAS). E' ovvio che, diversamente dall'esempio sopra riportato, i DAS moderni possono avere frequenze di campionamento che superano la potenza computazionale dello SHARC, ed è proprio tale contrasto che limita superiormente in frequenza la gamma di segnali portanti che tale processore riesce ad analizzare con l'algoritmo proposto. Analogamente, la possibilità di rendere fluida un'animazione grafica dipende strettamente dagli algoritmi implementati, i quali, diventando sempre più complessi, "tirano giù" la frequenza di cambio frame, vanificando il vantaggio competitivo offerto dal processore grafico.

La tecnica implementata è quella "double-buffer", ovvero viene allocato un buffer per accogliere i dati provenienti dal DAS, e tale buffer viene diviso in due parti uguali. Una parte viene riempita con i campioni acquisiti dal DAS, ad una frequenza di campionamento fissata, mentre l'altra viene analizzata dall'algoritmo di misura nei tempi "morti" tra due istanti di campionamento. Una volta completata una prima metà, il DAS continua a riempire il buffer nella seconda metà, la quale non viene più utilizzata dall'algoritmo di misura, e quest'ultimo può elaborare i

campioni depositati nella prima metà (figura 4.2). E' evidente che la condizione necessaria e sufficiente affinché il sistema real-time funzioni è data dal tempo impiegato nell'elaborazione dei campioni, il quale deve essere inferiore al tempo impiegato dal DAS per riempire metà del buffer d'ingresso. Ovviamente tale tecnica si presta anche ad un'elaborazione "off-line": in tal caso, la frequenza di campionamento del DAS può essere scelta a piacimento, in quanto esso viene disattivato tra un'elaborazione e l'altra.



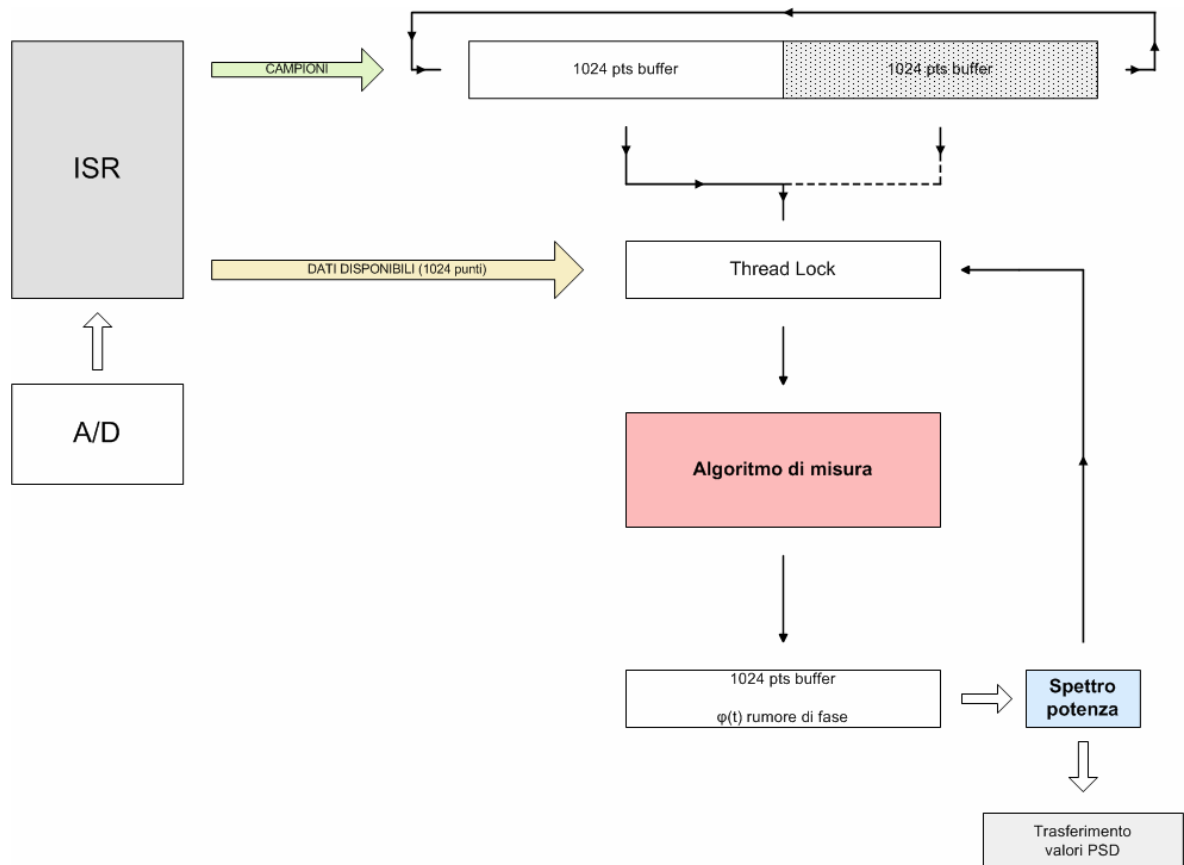
**Fig. 4.2 – Tecnica "double-buffer"**

## 4.2 Gli stadi dell'algoritmo di misura

L'algoritmo è diviso in due sezioni separate. La prima, installata come ISR (Interrupt Service Routine), si occupa di configurare l'I/O per i trasferimenti dei campioni e di segnalare alla seconda sezione quando è disponibile il buffer contenente i valori campionati del segnale per l'elaborazione.

La seconda costituisce l'algoritmo di misura, ed è divisa in sette stadi (figura 4.3):

- mixing del segnale campionato con i due toni in quadratura;
- filtraggio passa-basso delle due componenti I/Q;
- calcolo dell'arcotangente;
- correzione del salto di fase (unwrap);
- stima e correzione del termine lineare;
- FFT;
- trasferimento dei risultati all'unità di uscita.



**Fig. 4.3 – Diagramma dell'algoritmo di calcolo**

Gli unici due stadi implementati con codice di libreria fornito dalla Analog Devices sono il calcolo dell'arcotangente e la FFT; gli altri, pur se disponibili (come il filtraggio passa-basso), sono stati implementati ad hoc per incrementarne l'efficienza rispetto alla particolare scelta dei parametri di calcolo. Nei paragrafi

successivi verrà dedicato un paragrafo per ogni stadio. I parametri su cui si basa l'elaborazione vengono riassunti nella seguente tabella:

Lunghezza del buffer	1024 punti
Rapporto tra freq. di camp. e freq. tono portante	4
Tipo filtro	Low-pass FIR
Lunghezza filtro	64 TAPS
Punti utilizzati per FFT	1024

Le scelte di tali parametri verranno motivate durante l'analisi di ogni stadio. La scelta di fattori quali le potenze di  $2^n$  non è casuale, ma è dettata dall'esigenza di avere una "inizializzazione automatica" dei singoli stadi dopo ogni calcolo. Tanto per fare un esempio, se per il filtro FIR fossero stati scelti 63 TAPS, si sarebbe dovuto tener memoria della posizione dell'ultimo elemento inserito nel registro di stato, con un'ulteriore perdita di cicli e memoria. Da un'analisi grossolana del tempo occupato da ogni singolo stadio, è stata misurata una frequenza massima di lavoro pari a 500kS/s. Durante l'esecuzione sul processore DSP, si è notato in realtà che salendo anche a 600kS/s, non si sono riscontrati problemi di sincronizzazione.

### 4.3 Inizializzazione

In una prima fase, scorrelata dall'esecuzione continua dell'algoritmo di calcolo e non riportata in figura 4.3, vengono inizializzate le strutture dati utili al sistema per l'esecuzione degli stadi sopra riportati, e in particolare:

- viene inizializzato un oggetto di tipo "evento", chiamato *Data\_Avail\_Evt*, che verrà utilizzato per sincronizzare la ISR con il thread destinato alla elaborazione del segnale, di seguito denominato *IQThr*;
- viene creata una matrice di esponenziali complessi, chiamati "twiddle factors", la quale verrà usata nello stadio FFT;
- un buffer di 1024 punti viene riempito con i campioni di una finestra di Hanning, la quale sarà utilizzata per implementare il metodo di analisi

spettrale WOSA, e un secondo buffer di 512 punti viene azzerato, che conterrà le somme parziali derivanti sempre dal metodo WOSA.

Essendo tali operazioni effettuate “off-line”, il tempo impiegato per la loro esecuzione è ininfluenza ai fini della determinazione delle prestazioni dello strumento. Difatti, al termine di tali operazioni, viene smascherato l'interrupt su cui agisce la ISR, permettendo ai campioni di essere depositati nel buffer. Prima di eseguire il ciclo infinito che porta all'elaborazione real-time, *IQThr* attende un primo buffer di 1024 punti, di cui solo gli ultimi 63 vengono utilizzati per riempire i registri di stato dei due filtri FIR. Questo passo è necessario per evitare la presenza di “spike” non desiderati nel rumore di fase ottenuto nei primi 1024 punti, i quali, oltre ad alterare in ogni caso la misura prodotta, falsano la stima della correzione locale del termine lineare nel rumore di fase. Da questo punto in poi, viene realizzato lo schema riportato in figura 4.3.

#### 4.4 ISR

La sezione ISR si occupa di programmare la porta parallela per il trasferimento dei campioni dalla memoria esterna presente sulla board EZKIT in quella interna al DSP. Tale trasferimento avviene sul trigger di un timer periodico, la cui frequenza è impostata in modo tale da rispettare la condizione di funzionamento real-time. Ad ogni esecuzione della ISR, vengono trasferiti 64 campioni, in modo da simulare il comportamento del DMA nei confronti del sistema di acquisizione dati. La porta parallela ha una capacità di trasferimento, alla sua massima velocità, di  $\frac{f_{core}}{3}$ , dove  $f_{core}$  è la frequenza di clock del DSP, nel nostro caso 200MHz, ottenendo una banda di trasferimento pari a 66 Mbyte/s, quindi 64 campioni vengono trasferiti in 1  $\mu$ s. Dal punto di vista della ISR, il buffer viene riempito in modalità circolare su 2048 punti, con la condizione di segnalare, ogni 1024 punti raccolti, il bit nella maschera

eventi di *Data\_Avail\_Evt*, denominato *Data\_Avail*, in modo tale da “svegliare” il thread in attesa dei dati.

La relazione che lega il tempo di timer alla frequenza di campionamento è:

$$T_{timer} = \frac{64}{f_c \cdot f_{core}}$$

Tale valore può essere agevolmente impostato, in millisecondi, sul pannello “Kernel” dell’ambiente VisualDSP, nonché variato, laddove fosse necessario, al tempo d’esecuzione modificando il valore del registro TPERIOD.

Essendo la porta parallela utilizzata anche dal bus dedicato all’interfaccia con un sistema di acquisizione dati, l’integrazione di tale codice con un DAS collegato al DSP è immediata; basta modificare opportunamente il codice della ISR reindirizzando la porta parallela verso i registri destinati all’interfacciamento con il DAS.

## 4.5 Mixing

Nel primo stadio viene realizzata la moltiplicazione tra il segnale portante campionato e le sequenze corrispondenti ai toni in quadratura. La scelta di un rapporto tra frequenza di campionamento e frequenza portante pari a 4 permette di utilizzare unicamente i data register file del DSP, senza dover precalcolare e depositare le sequenze in memoria: difatti, sono sufficienti le sequenze [0, 1, 0, -1] e [1, 0, -1, 0], lette in modalità circolare, per costruire rispettivamente i segnali seno e coseno. La figura 4.4 mostra il precaricamento di tali sequenze nei data register file delle unità PE, e il calcolo effettuato per ottenere le sequenze contenente i segnali I/Q non filtrati. Il vettore *Q-signal* contiene i campioni del segnale in ingresso, i quali vengono sovrascritti con il segnale Q non filtrato. In un secondo vettore, denominato *I-signal*, vengono memorizzati i campioni del segnale I. Il tutto avviene in modalità SIMD, quindi ogni operazione effettuata dall’unità PEx sull’elemento di posto *k* viene realizzata in parallelo dalla unità PEy sull’elemento di posto *k+1*.

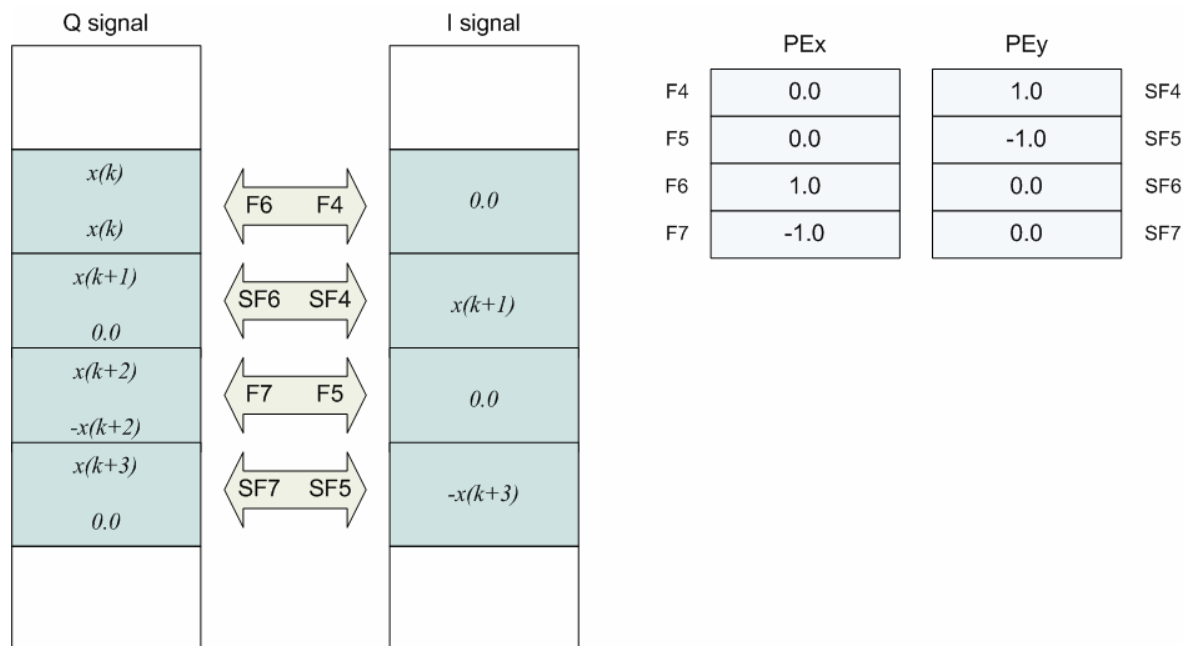


Fig. 4.4 – Mixing

Tale computazione avviene lungo il seguente loop:

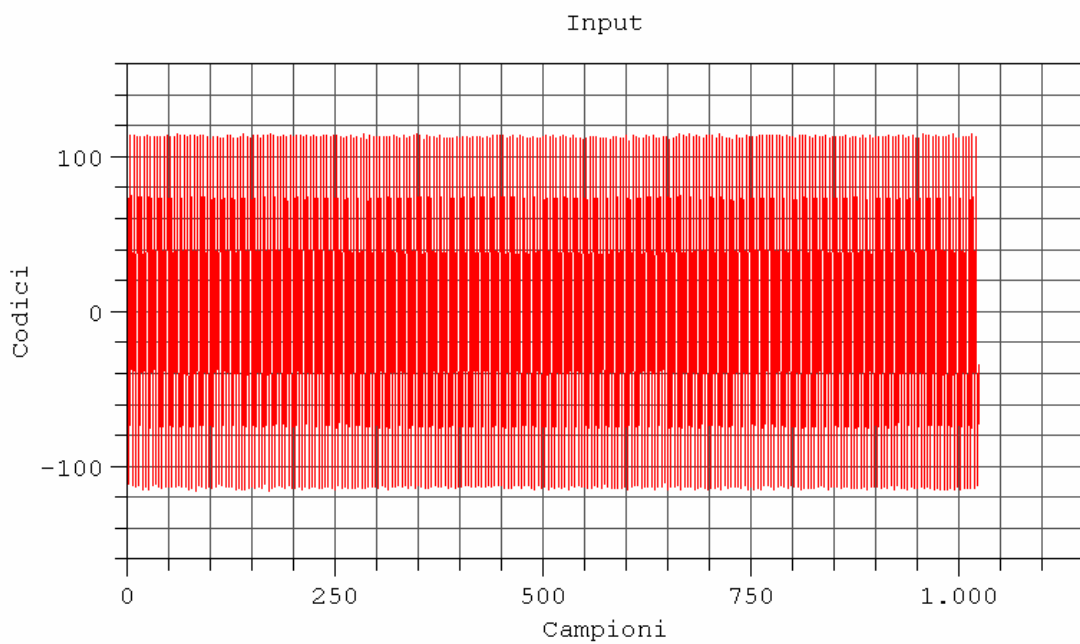
```

lcntr = r12, do (pc, loopend-1) until lce;
  f0 = dm(0, i4);
  f8 = f0*f4, f1 = dm(2, i4);
  f12 = f1*f5, dm(i1, m1) = f8;
  f0 = f0*f6, dm(i1, m1) = f12;
  f1 = f1*f7, dm(i4, m4) = f0;
  dm(i4, m4) = f1;
loopend:

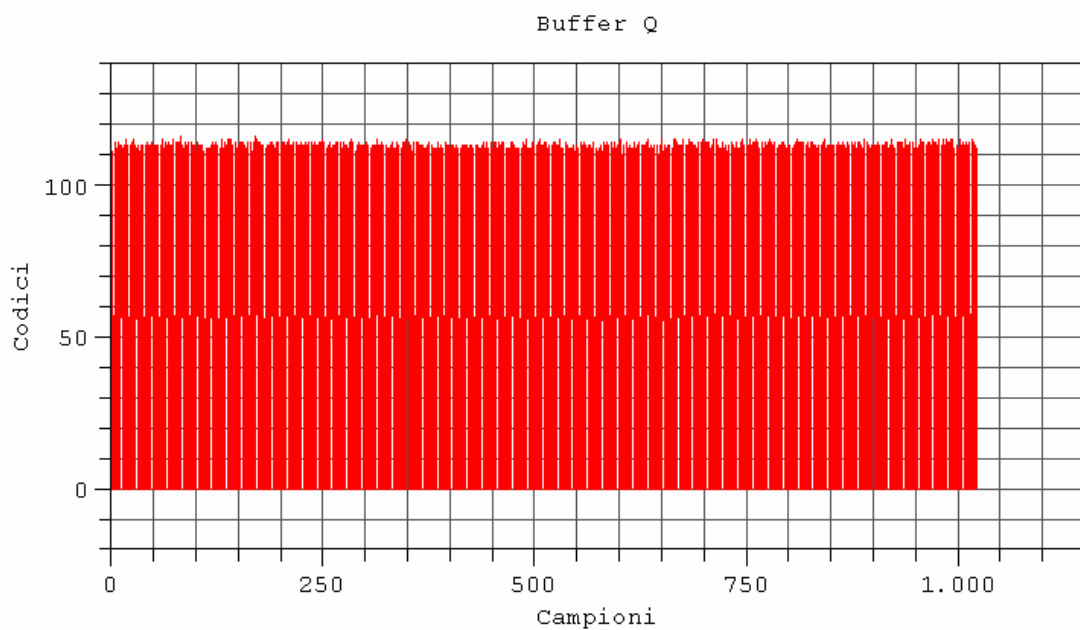
```

Come è possibile notare, gli accessi in memoria vengono effettuati parallelamente ai calcoli, tranne per la prima e ultima istruzione. Poiché il ciclo di loop non svuota la pipeline, l'intero processo di calcolo del primo stadio dura  $\frac{6 \cdot 1024}{4}$  cicli. La scelta del fattore 4 si è rivelata limitante nel caso di acquisizione di segnali tramite l'oscilloscopio LeCroy LC584AL, poiché la frequenza di campionamento di quest'ultimo non è regolabile in modo fine, ma può assumere solo valori limitati. Tuttavia, ciò non costituisce un limite futuro per il prototipo, poiché la sua

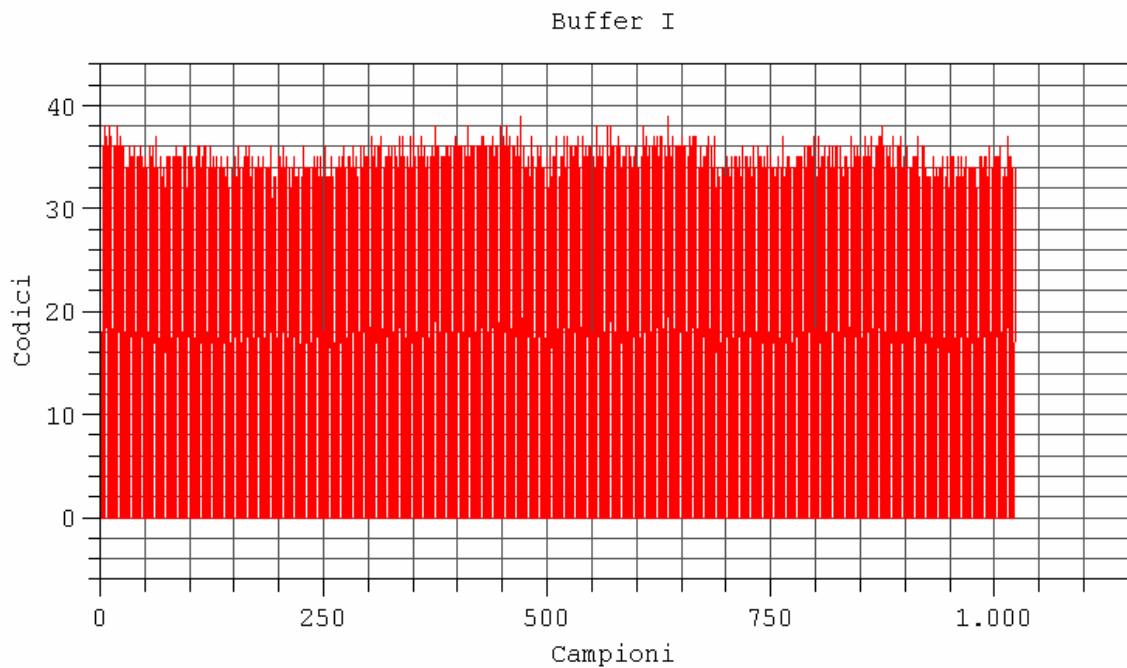
integrazione con il sistema di conversione A/D della Analog Devices permetterà di regolare in modo estremamente preciso la frequenza di campionamento, permettendo di rispettare sempre la relazione tra le frequenze. Le figure 4.5, 4.6 e 4.7 mostrano rispettivamente il contenuto dei buffer Q e I prima e dopo l'operazione di mixing.



**Fig. 4.5 – Buffer Q contenente i campioni del segnale d'ingresso**



**Fig. 4.6 – Buffer Q dopo il mixing**

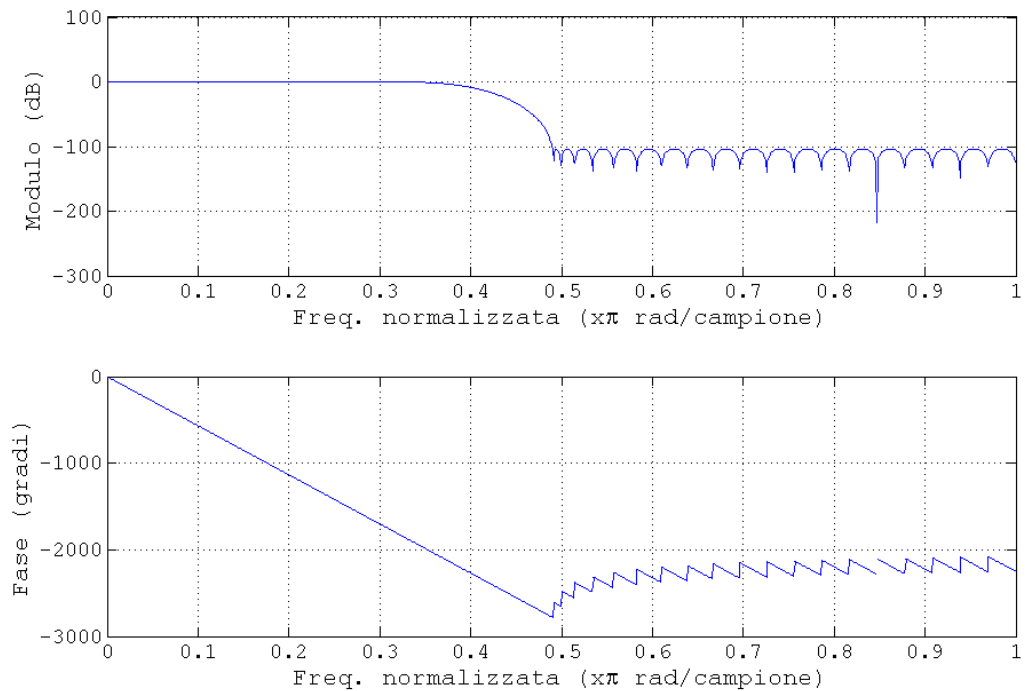


**Fig. 4.7 – Buffer I dopo il mixing**

## 4.6 Filtraggio

Nel secondo stadio entrambi i segnali Q e I vengono filtrati tramite un filtro passa-basso, per rimuovere le componenti ad alta frequenza. Per realizzare tale filtraggio, è stato implementato un filtro FIR (Finite Impulse Response) a 64 coefficienti, le cui risposte in frequenza sono riportate in figura 4.8. Tale filtro risponde in modo ottimale sia all'esigenza di avere un pesante abbattimento delle componenti ad alta frequenza e di avere una risposta lineare in banda passante, sia alla necessità di non aggravare eccessivamente il carico computazionale. Tale necessità viene soddisfatta tramite l'uso combinato di istruzioni MACS e modalità SIMD.

La progettazione dell'algoritmo FIR presenta tuttavia un forte limite, dovuto al vincolo di allineamento su indirizzi pari dei dati da analizzare: questo limite si presenta quando deve essere analizzato un elemento di posto dispari, sul quale non è possibile utilizzare la tecnica SIMD.



**Fig. 4.8 - Risposte in ampiezza e fase del filtro FIR**

Il filtro è dotato di due registri, denominati *coeffs* e *state*, i quali contengono rispettivamente i coefficienti che determinano la struttura del filtro, e i 63 valori passati dell'ingresso rispetto al valore del campione sottoposto al filtraggio.

E' utile ricordare la relazione ingresso-uscita di un filtro FIR:

$$y(k) = \sum_{j=0}^{L-1} c_j x(k-j) \quad k = 0 \dots N-1$$

dove:

- $L$  è la lunghezza del filtro;
- $c_j$  è il  $j$ -esimo coefficiente del filtro;
- $x(k)$  è il campione del segnale d'ingresso;
- $y(k)$  è il campione del segnale filtrato.

Inizialmente il vettore *state* viene inizializzato come illustrato in figura 4.9 (per semplicità di trattazione, si fa riferimento al solo segnale Q, essendo la procedura perfettamente analoga anche per il segnale I).

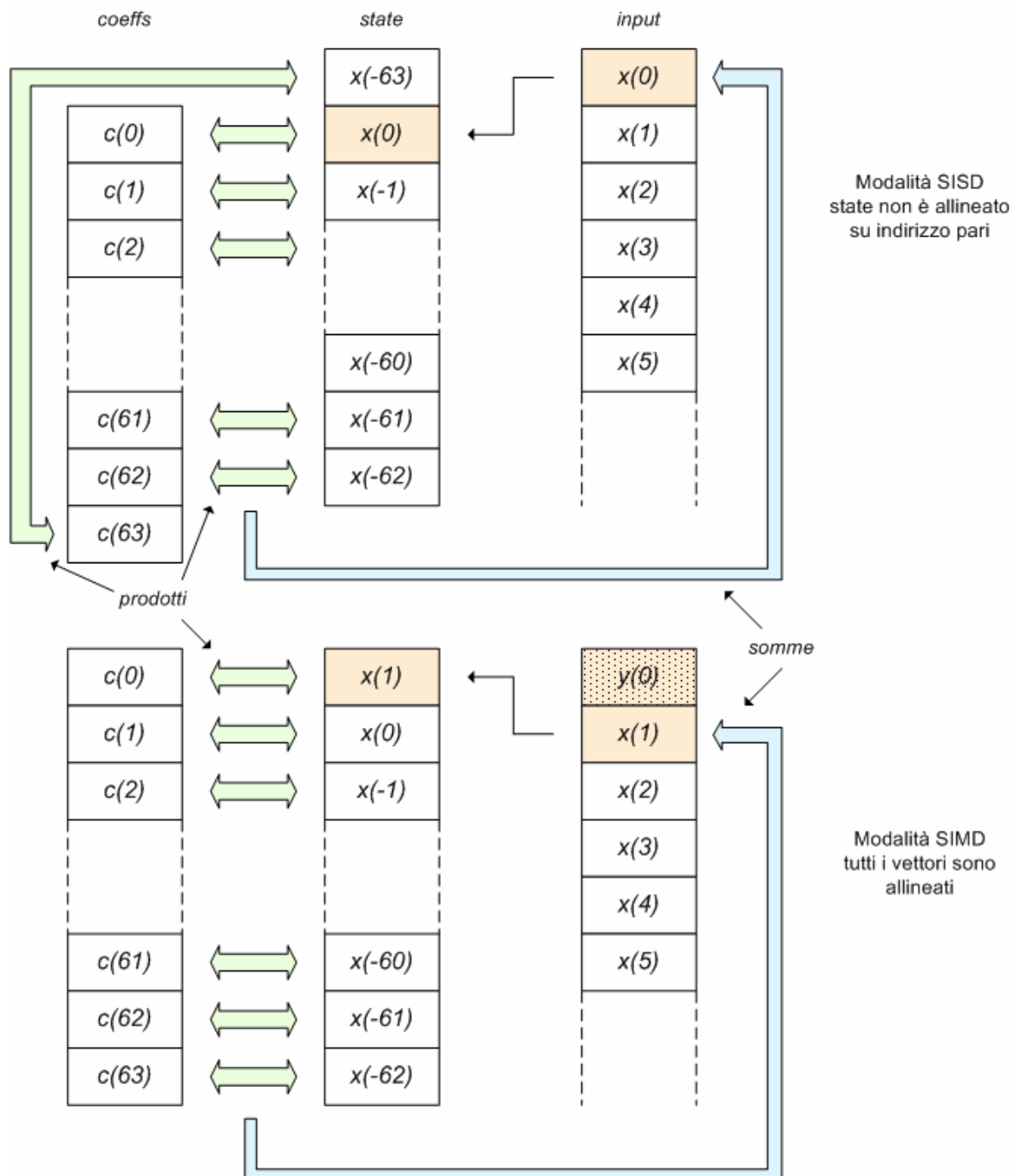
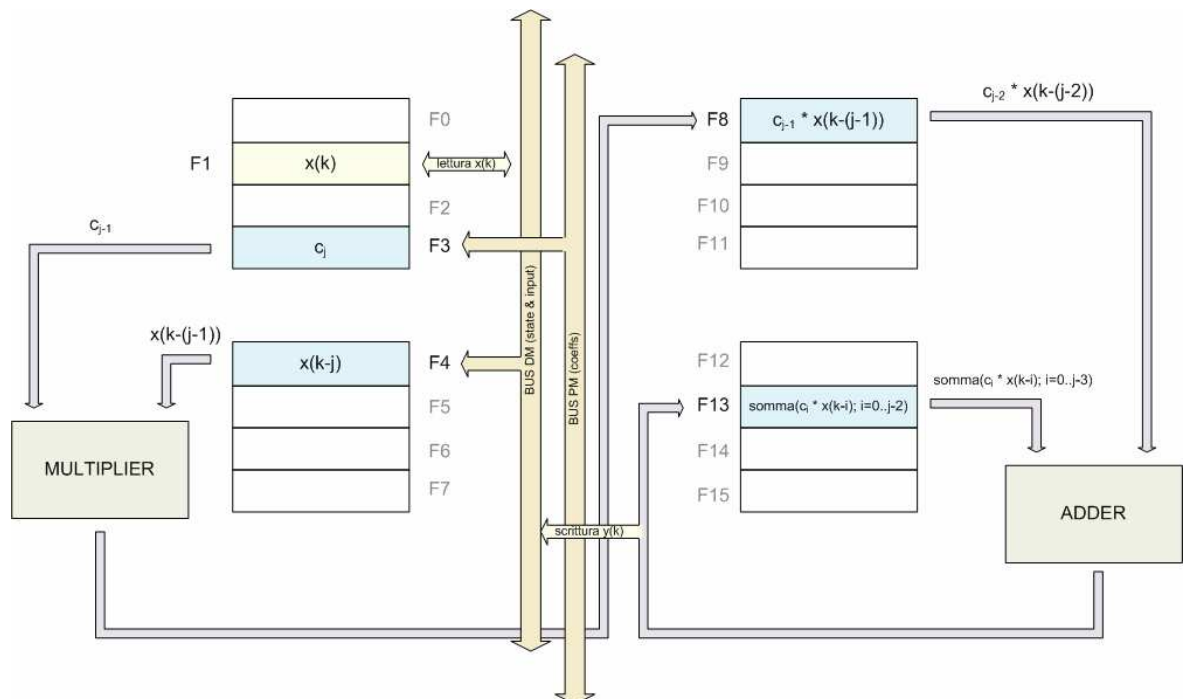


Fig. 4.9 – Schema logico di calcolo del filtro FIR

I vettori *state* e *coeffs* vengono gestiti in modalità di accesso circolare. Il primo elemento di *input* viene trasferito nel registro *state*, e viene effettuato un prodotto scalare tra i vettori *coeffs* e *state*. Tale prodotto rappresenta il campione  $y(0)$ . Tuttavia, il primo elemento di *state* utile per effettuare il prodotto scalare si trova allineato su un indirizzo dispari, quindi la modalità SIMD non può essere realizzata. Al secondo passo, il secondo elemento di *input* viene trasferito nella posizione precedente a quella utilizzata nel primo passo, la quale si trova in un indirizzo di memoria pari. In questo caso, la modalità SIMD può essere attivata. Quindi solo 512 punti del vettore di ingresso potranno essere calcolati in modalità SIMD, la restante metà viene trattata in modalità SISD.

In entrambi i casi, la combinazione di prodotti e somme viene realizzata utilizzando una MACS, rappresentata in figura 4.10:



**Fig. 4.10 – MMACS per singola operazione FIR**

Tale figura rappresenta il flusso di dati in un solo ciclo, ed è una dimostrazione efficace della potenza di calcolo del processore SHARC. Il loop che esegue il calcolo di  $y(k)$  è il seguente:

```

r8 = r8-r8, f1 = dm(0,i1);
r13 = r13-r13, dm(0,i2) = f1;

f4 = dm(i2,m2), f3 = pm(i8,m8);

lcntr = r2, do (pc,loopend_fir-1) until lce;
    f8 = f3*f4, f13 = f8+f13, f4 = dm(i2,m2), f3 = pm(i8,m8);
loopend_fir:
    f13=f8+f13, f4 = dm(i2,-2);

modify(i8,-1);
dm(i1,m1) = f13;

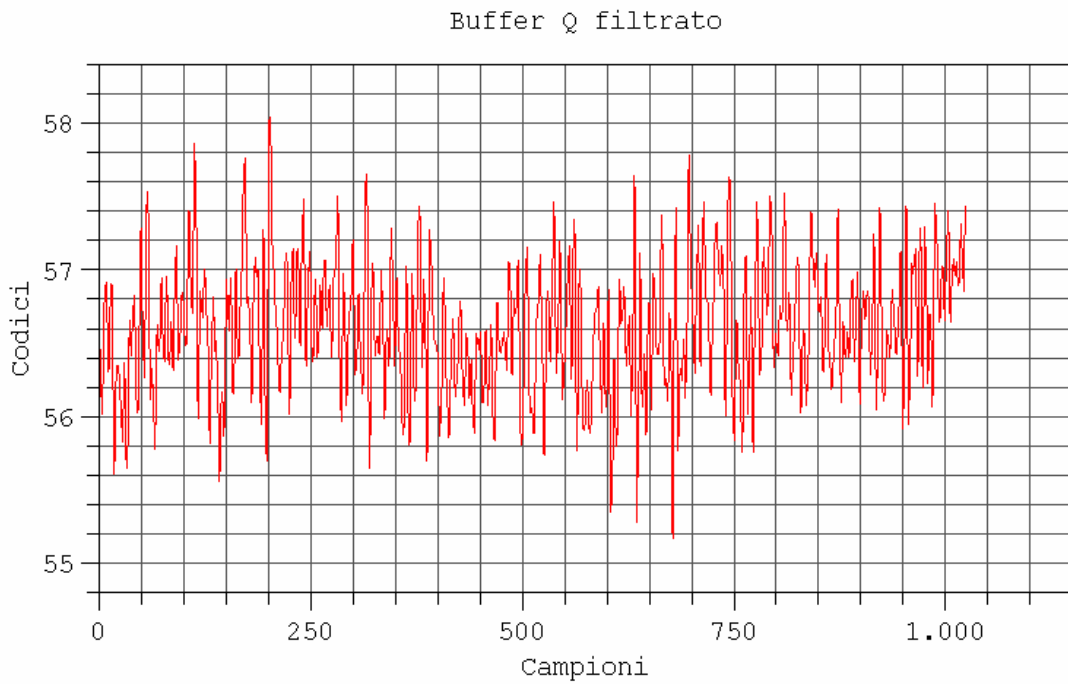
```

L'istruzione interna al loop esegue due accessi in memoria, una somma e un prodotto in un solo ciclo. Affinchè il lettore possa comprendere cosa avviene in tale istruzione, basta tener presente che gli operandi presenti sulla parte destra di un'operazione assumono il valore che essi hanno immediatamente prima dell'esecuzione della MACS; ovvero, il registro F4 viene aggiornato con il valore letto dalla memoria DM puntato da I2, ma il valore utilizzato per il prodotto  $F8 = F3 * F4$  è quello presente in F4 prima dell'aggiornamento. Ciò è possibile, in quanto il valore di F4, prima di essere sostituito, viene memorizzato in un registro privato del moltiplicatore.

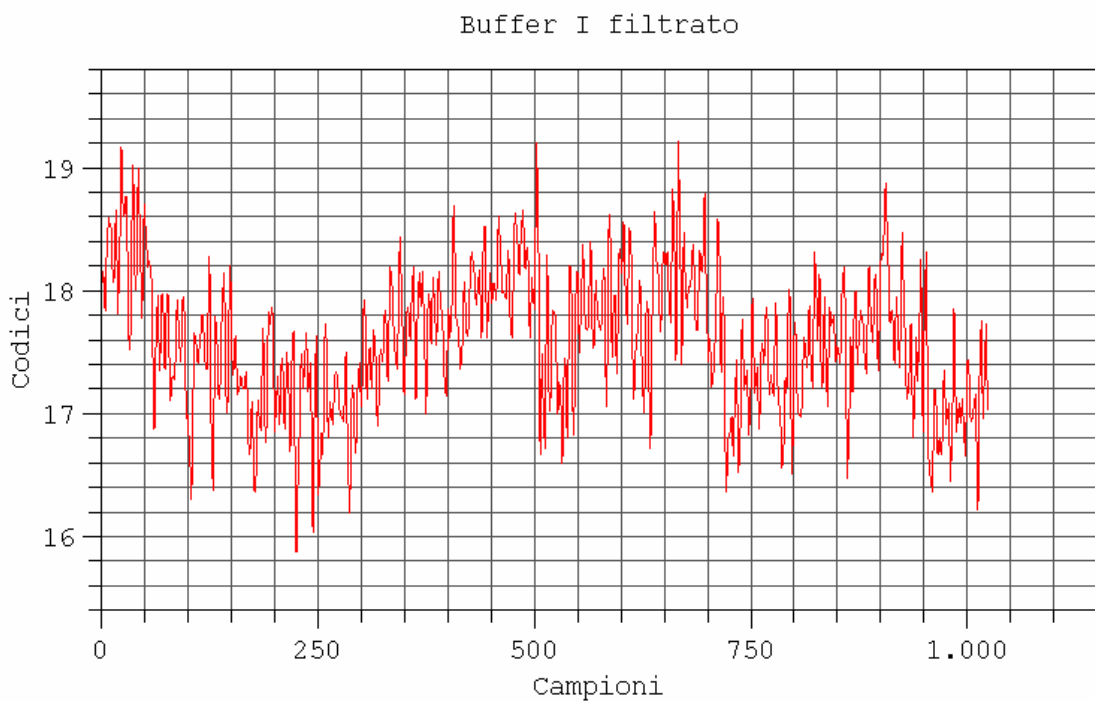
Nonostante il calcolo del filtro FIR spinga al massimo le prestazioni del processore SHARC, tale calcolo rappresenta uno dei due anelli più gravosi dell'intera elaborazione. Ciò è dovuto alla definizione teorica di un sistema FIR: difatti, per un filtro di 64 TAPS vanno realizzate, per ogni singolo punto filtrato, 64 prodotti e 63 somme. Su un processore che non prevede l'utilizzo di MMACS, i cicli persi, senza contare gli accessi in memoria, sarebbero stati 127; invece, nel

caso del DSP, il numero di cicli è bloccato a 64 (32 nel caso SIMD), e vengono utilizzati complessivamente  $512 \cdot 64 + 512 \cdot 32$  cicli.

In figura 4.11 e 4.12 vengono riportati i segnali Q e I (vedi figura 4.6 e 4.7) filtrati.



**Fig. 4.11 – Segnale Q filtrato**



**Fig. 4 12 – Segnale I filtrato**

Una implementazione molto più veloce del filtraggio passa-basso è quella effettuata mediante il prodotto dello spettro del segnale e del filtro nel dominio della frequenza, tuttavia tale soluzione comporta un maggior consumo di memoria e una probabile perdita di campioni tra i vari segmenti del rumore di fase analizzati, a causa della convoluzione circolare.

#### 4.7 Arcotangente e unwrap

Il calcolo della funzione arcotangente, utile per estrarre il rumore di fase, viene effettuata tramite la funzione *atan2f*, disponibile nelle librerie matematiche del VisualDSP++. Tale funzione, così come riportato dalle specifiche Analog Devices, impiega una media di 83 cicli per singolo punto, consumando 84992 cicli per segmento di rumore di fase, che risulta essere circa il doppio del tempo impiegato dal filtro FIR. Purtroppo il processore SHARC non possiede unità dedicate ai calcoli logaritmici e trigonometrici (come molte floating-point unit in commercio), e questa assenza viene pagata pesantemente con l'uso di tale funzione.

In uscita a tale stadio, troviamo l'argomento della tangente, i cui valori sono contenuti nell'intervallo  $[-\pi; \pi]$ . In prossimità degli estremi si verificano quindi delle discontinuità "circolari", ovvero laddove la fase supera il valore di  $\pi$ , essa riprende da  $-\pi$ . La figura 5.13 mostra tale fenomeno. Per eliminare tali discontinuità, è stata implementata una Macchina a Stati Finiti a 5 stati, la quale "segue" l'andamento della fase e registra i salti di  $2\pi$  che avvengono durante il suo andamento. L'idea alla base del diagramma di stato che implementa tale MSF è quella di spostare la discontinuità esistente sui valori  $\pi$  e  $-\pi$  su soglie maggiormente "controllabili", ovvero  $\pi/3$  e  $-\pi/3$ . In tal modo, si hanno informazioni sulla variazione di fase tra un punto ed un altro, e insensibilità al rumore presente sulla fase, il quale, con altri metodi di unwrap, potrebbe portare alla presenza di "spike" di  $2\pi$  non desiderati.

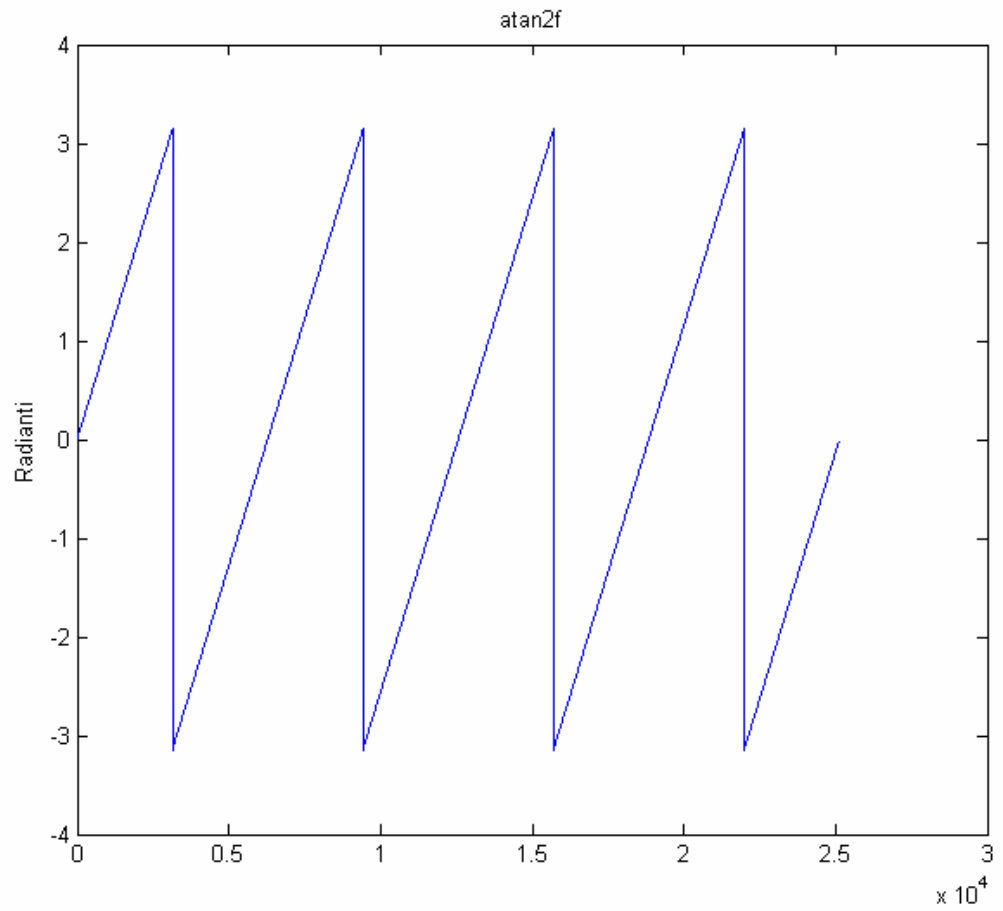


Fig. 4.13 – Fenomeno del “wrap” di fase

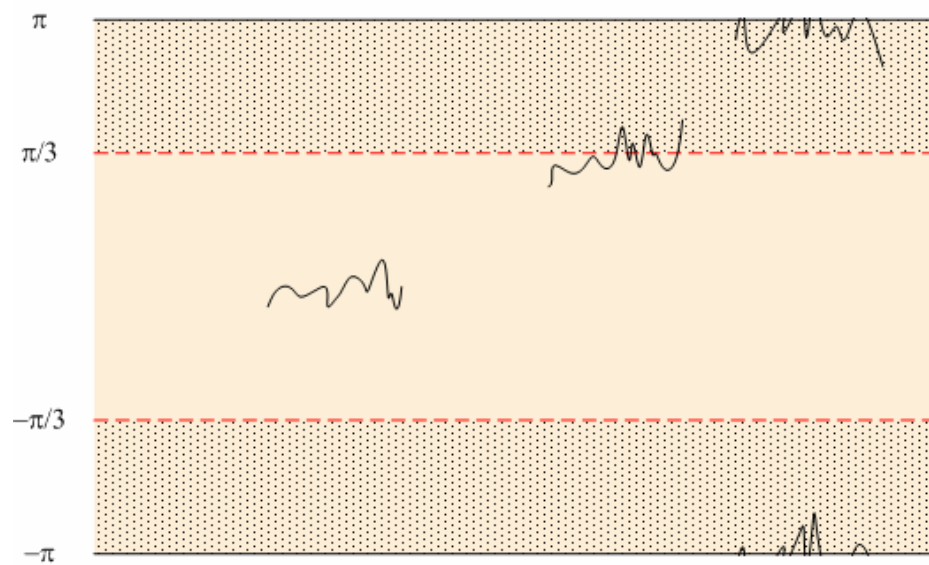
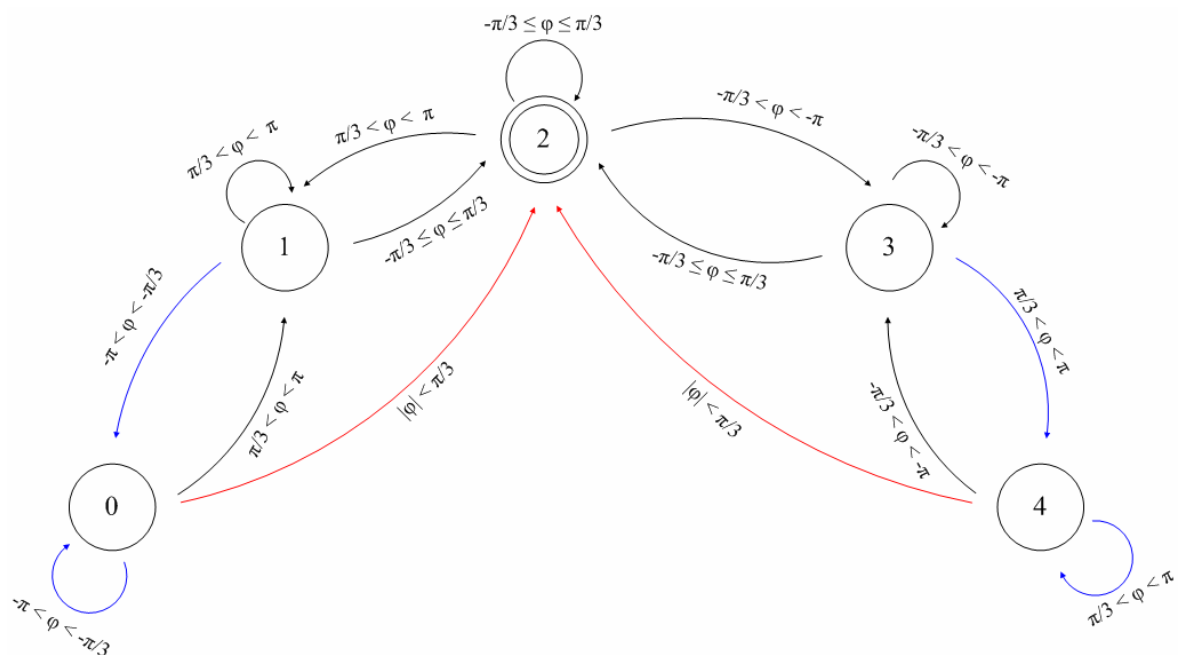


Fig. 4.14 – Effetto del rumore sulla discontinuità

Difatti, come mostra la figura 4.14, per effetto del rumore, se non si hanno informazioni riguardo alla variazione del segnale nel tempo, possono essere memorizzati salti di  $2\pi$  “temporanei”, i quali poi porterebbero ad altri salti di  $2\pi$ , non introdotti dall'arcotangente, ma dal metodo di unwrap stesso. Registrando invece un primo salto nell'intorno dei valori  $-\pi/3$  e  $\pi/3$ , è possibile controllare tale fenomeno, e registrare il salto di  $2\pi$  solo nell'istante in cui la fase abbia effettivamente compiuto un salto di  $2\pi$ . Per comprendere l'algoritmo di unwrap implementato, basta osservare il diagramma di stato della MSF in figura 4.15.

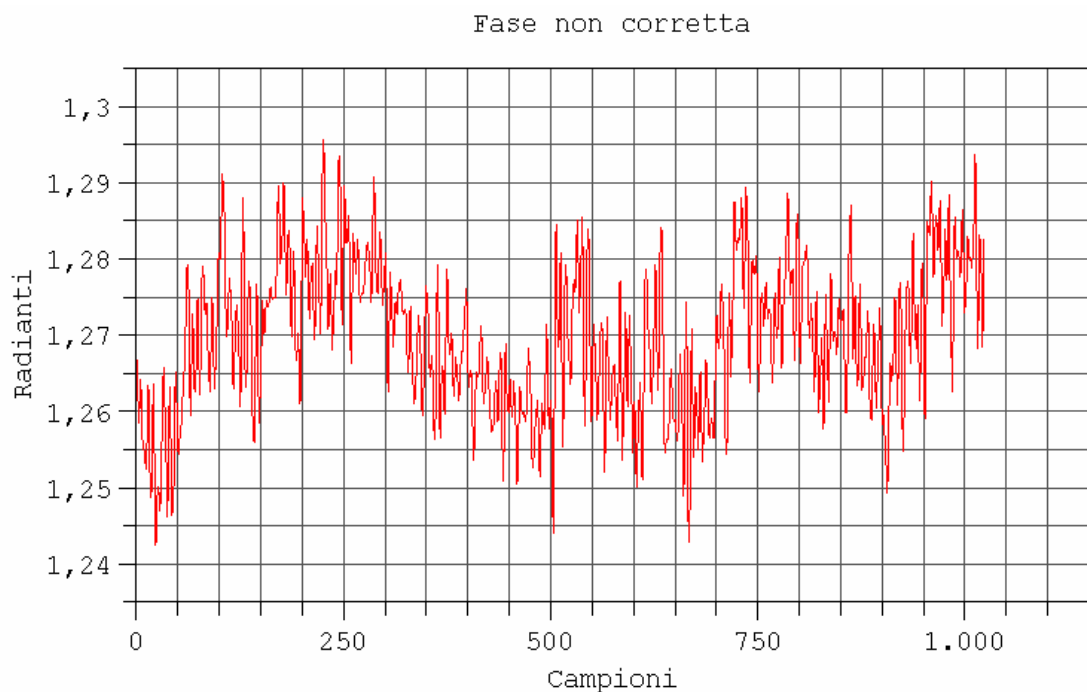


**Fig. 4.15 – Diagramma di stato della MSF**

La MSF prevede due registri di stato, uno dedicato alla memorizzazione dello stato della MSF, e l'altro dedicato alla memorizzazione dei multipli di  $2\pi$  da sommare ai campioni durante l'elaborazione. Lo stato di “reset” corrisponde allo stato 2, ovvero la fascia di valori compresa tra  $-\pi/3$  e  $\pi/3$ . Negli stati 1, 2 e 3, viene sommato al campione il valore corrente del registro contenente i valori di  $2k\pi$ , senza alterarlo. Quando si verifica una transizione “blu”, ovvero negli stati 0 o 4, la fase sta attraversando la discontinuità nell'intorno di  $\pi$  o  $-\pi$ ; tuttavia potrebbe non trattarsi di

un vero salto di  $2\pi$ , in quanto l'effetto del rumore potrebbe far scendere la fase all'interno della discontinuità, e in tal caso il salto non va memorizzato. In tal caso, vengono sommati, oltre al valore di  $2k\pi$ , altri  $2\pi$  nel caso in cui, nello stato 0, il valore da positivo cambia segno (specularmente nello stato 4). Laddove invece, permanendo nello stato 0, la fase continuasse a salire fino ad arrivare a  $-\pi/3$ , il salto di  $2\pi$  può essere registrato e sommato (transizione rossa). In tal caso, si ritorna nello stato 2.

Il carico computazionale dello stadio di unwrap è molto leggero, in quanto, per ogni singolo campione, viene eseguito il blocco di istruzioni appartenente ad uno ed un solo stato, il quale è lungo in media 7 cicli. In figura 4.16, è riportato il rumore di fase senza correzione del termine lineare. Il limite di tale metodo è che, tra due punti successivi del rumore di fase, la differenza in modulo deve essere minore di  $2\pi/3$ , condizione ampiamente soddisfatta per i segnali d'interesse.



**Fig. 4.16 - Rumore di fase senza correzione del termine lineare**

## 4.8 Correzione del termine lineare

Così come descritto nel capitolo 1, a valle dello stadio di unwrap abbiamo il seguente segnale:

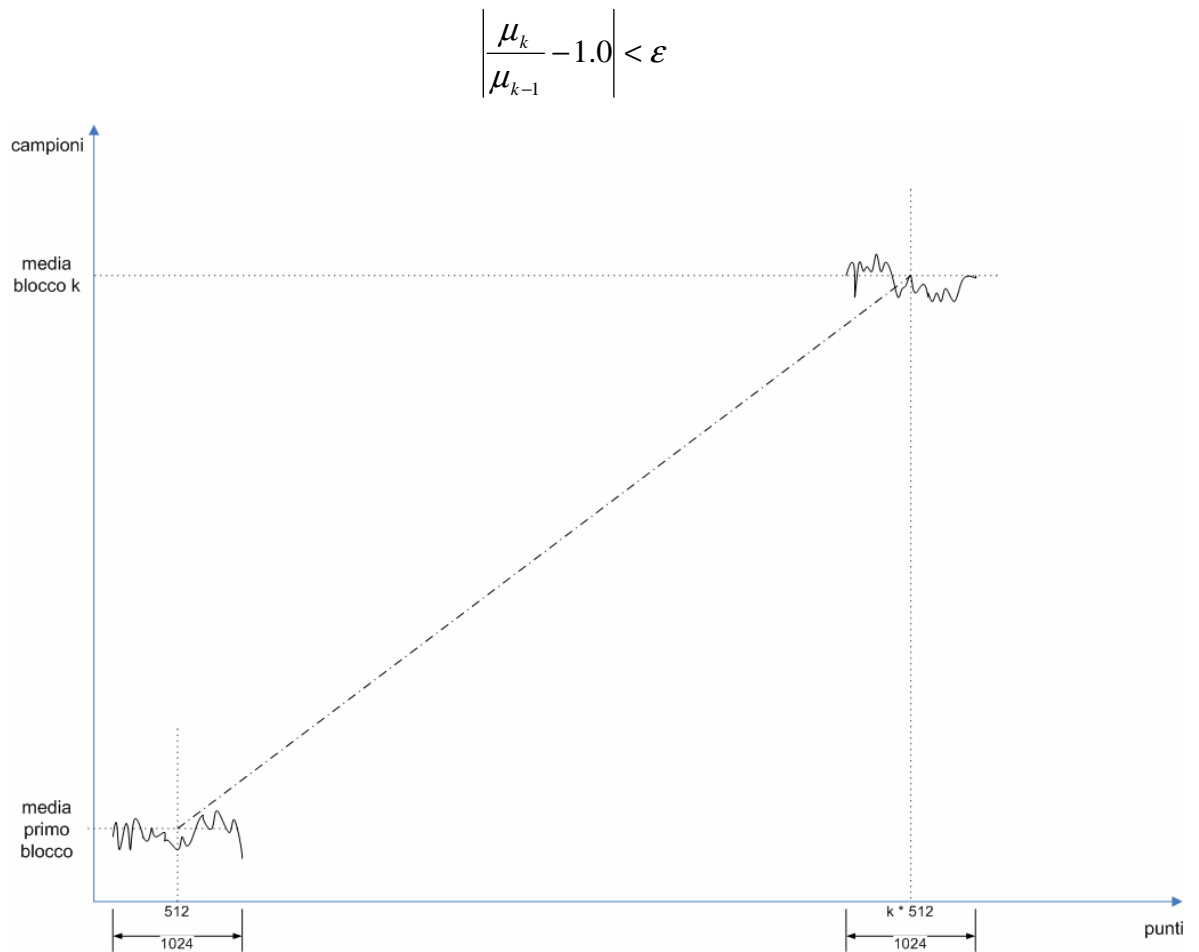
$$\Phi(t) = 2\pi(f_s - f_0)t + \varphi_0 + \varphi(t)$$

Il termine  $2\pi(f_s - f_0)t$  è un drift in frequenza dovuto al fatto che il tono portante del segnale ricevuto è solo nominalmente pari a  $f_0$ ; tuttavia esso presenta degli scostamenti molto piccoli (dell'ordine di  $10^{-6} \div 10^{-9}$  Hz), i quali, sul lungo periodo, portano ad una divergenza di tale segnale. Il termine costante  $\varphi_0$  invece è dovuto allo sfasamento esistente tra il segnale portante e i toni in quadratura, e dipende strettamente dall'istante temporale in cui inizia il campionamento.

In teoria, per ottenere una stima con la massima affidabilità, va implementato un metodo ai minimi quadrati sull'intero segmento di 1024 punti. Tuttavia, l'enorme quantitativo di somme e prodotti coinvolti in tale operazione renderebbe tale stadio ancora più gravoso rispetto all'arcotangente, tale soluzione è stata quindi scartata.

Un secondo metodo, meno preciso rispetto al precedente, è quello di usare la classica formula del coefficiente angolare di una retta passante per due punti, costruiti come in figura 4.17. In pratica, per ogni blocco di 1024 punti viene effettuata una media aritmetica e si costruisce un punto avente come ordinata tale media e come ascissa l'indice centrale del blocco moltiplicato per  $k$ , dove  $k$  è il numero di blocco elaborato. Il punto del primo blocco elaborato è fisso, mentre il secondo punto appartiene al blocco corrente, e viene di volta in volta aggiornato. Dai due punti ottenuti viene calcolato il coefficiente angolare della "retta ideale" che congiunge i "centri" dei blocchi di dati analizzati.

Le pendenze ottenute vengono raccolte in un buffer lungo 32 punti, e su tale buffer si opera una media mobile. A questo punto, vengono fissate due soglie, ottenute valutando la precisione massima offerta dal floating-point del DSP (che prevede 6 cifre decimali per la mantissa), e viene controllata se la seguente relazione è soddisfatta:



**Fig. 4.17 – Stima del drift tra i segmenti del rumore di fase**

La soglia  $\varepsilon$  viene impostata in un primo momento a  $5 \cdot 10^{-4}$ , dove  $\mu_k$  è il valore di pendenza ottenuto dalla media sul buffer di 32 punti. Quando la relazione è verificata, il valore della soglia  $\varepsilon$  viene impostato a  $5 \cdot 10^{-6}$ , e di volta in volta viene sottratto il valore di pendenza calcolato e operata una ulteriore media sul blocco di 1024 punti depurato del termine lineare, ma non del termine costante. Tali medie vengono raccolte in un secondo buffer di 32 punti, e anche su di esso viene operata una media mobile, il cui valore rappresenta la stima della intercetta, ovvero del valore costante  $\varphi_0$ . Se la relazione viene soddisfatta per la seconda volta, il valore di pendenza e intercetta ottenuti rappresentano le stime migliori del drift.

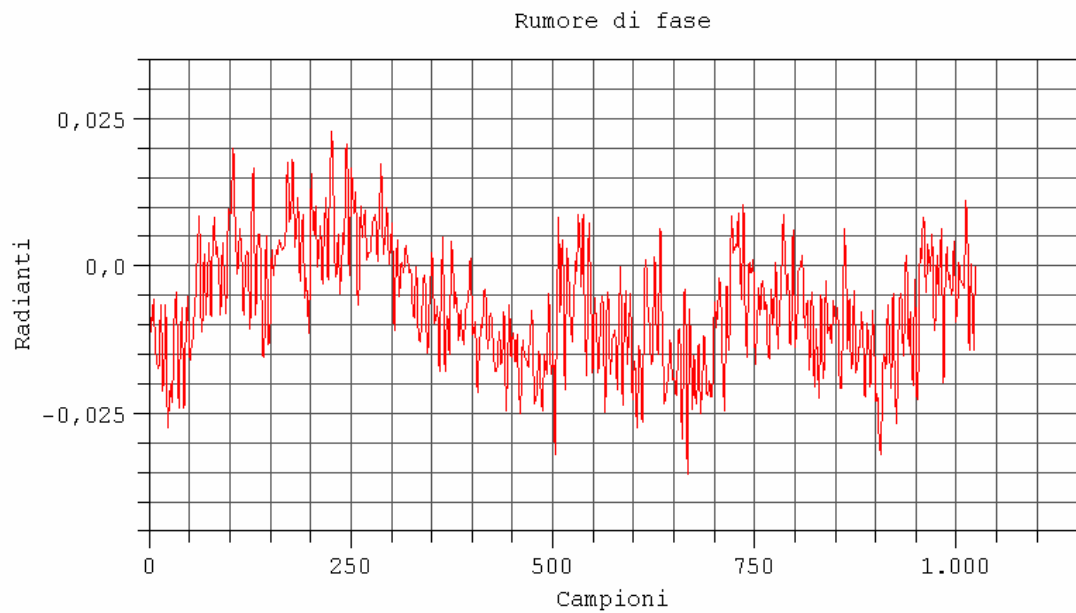
Tale metodo, anche se computazionalmente molto efficiente rispetto alla qualità dei risultati ottenuti, presenta il pesante svantaggio di non poter produrre la misura dello spettro di potenza prima di aver stimato il drift da correggere, così come il

metodo dei minimi quadrati. Tuttavia, rispetto a quest'ultimo, potrebbero occorrere milioni di punti per realizzare tale stima, il che, tradotto in termini temporali, potrebbe significare un'attesa di alcuni secondi, supponendo di lavorare con 500kS/s come frequenza di campionamento. Inoltre, una volta calcolato tale drift, questo metodo prevede l'ipotesi, non sempre rispettata in realtà, che la pendenza e l'intercetta non varino più nel tempo.

Il metodo implementato sul DSP si basa su una correzione locale sul singolo segmento, calcolando il coefficiente angolare in base al primo e ultimo punto del blocco. Successivamente i punti finali e iniziali di due blocchi contigui vengono uguagliati compensando l'offset esistente tra un segmento e l'altro. Ovviamente tale metodo non fornisce la stima del drift in modo corretto, in quanto la valutazione della pendenza è fortemente legata dalla posizione del primo e ultimo punto, posizione enormemente influenzata dal rumore. Tuttavia, considerando gli ordini di grandezza in gioco, si è potuto verificare dalle simulazioni che il rumore di fase estratto è pressochè equivalente, come sequenza di campioni, a quello calcolato con i precedenti metodi. Inoltre, la correzione locale presenta tre enormi vantaggi:

- è virtualmente istantanea (la stima del drift consiste in due differenze e una divisione), considerando il peso computazionale dei due metodi precedenti;
- non è influenzata da cambiamenti della pendenza durante l'acquisizione;
- permette di produrre sin dal primo blocco la misura dello spettro di potenza.

In figura 4.18 è riportato un blocco del rumore di fase estratto.



**Fig. 4.18 - Rumore di fase**

## 4.9 Spettro di potenza

Nell'ultimo stadio di calcolo viene realizzata una analisi spettrale del rumore di fase ottenuto con due diversi metodi, i quali verranno illustrati nel capitolo 6. Entrambi i metodi fanno uso di una routine Radix-2 Fast Fourier Transform, la quale implementa il classico calcolo "a farfalla" proprio della FFT, ed è una funzione di libreria della Analog Devices. Essa impiega circa 11000 cicli per operare una Trasformata di Fourier su 1024 punti.

Una volta ottenuti i valori reali e complessi dello spettro, per valutare lo spettro di potenza viene effettuato il seguente calcolo:

$$S(f) = \begin{cases} \frac{1}{N} |X(f)|^2 & f = 0 \\ \frac{1}{N} W_{ss} & \\ \frac{2}{N} |X(f)|^2 & f > 0 \\ \frac{1}{N} W_{ss} & \end{cases}$$

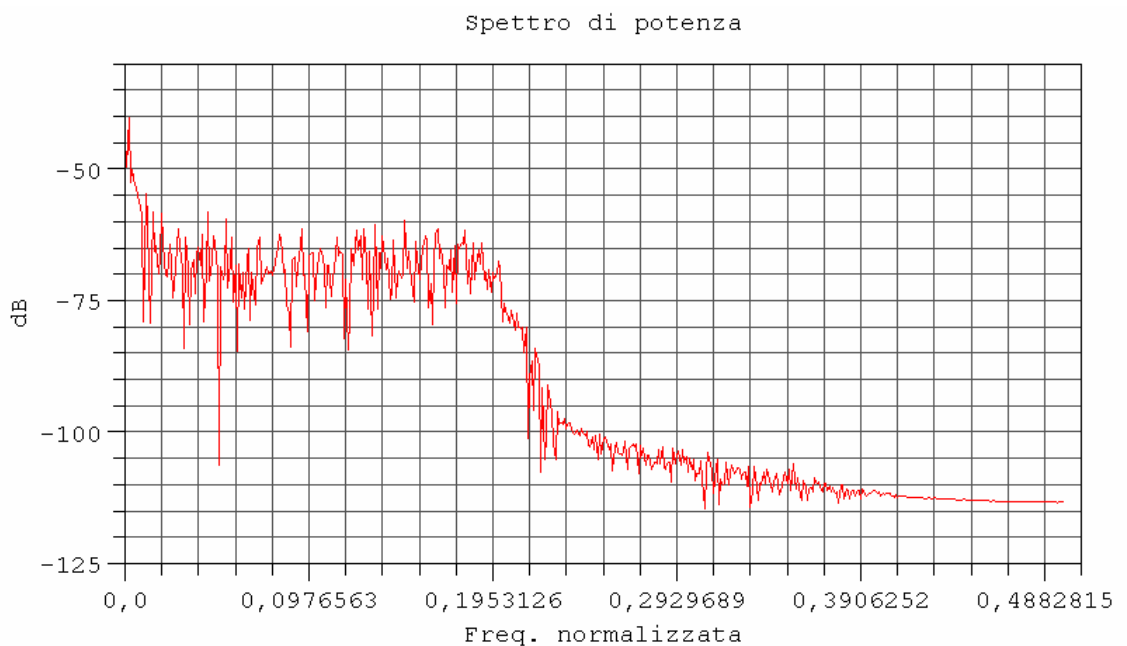
dove  $W_{ss}$  è la potenza della finestra utilizzata, definita come:

$$W_{ss} = \sum_{j=0}^{N-1} w_j^2$$

Per ottenere una misura corretta del periodogramma, la quantità  $S(f)$  andrebbe scalata per la frequenza di campionamento. Poiché tale quantità non è nota all'algoritmo di calcolo, la correzione può essere effettuata dal dispositivo destinato alla visualizzazione (in genere il calcolatore), al quale può essere fornita l'informazione sulla frequenza di lavoro.

A valle di tale stadio 512 punti dello spettro di potenza vengono trasferiti in memoria esterna, tramite porta parallela.

In figura 4.19, viene riportato lo spettro di potenza del primo blocco di punti, utilizzando una finestra rettangolare.



**Fig. 4.19 – Spettro di potenza su 1024 punti del rumore di fase**