INTRODUZIONE AL CHIP SET LH 9124 E LH9320. E PROPOSTA DI APPLICAZIONE AD UNA SCHEDA MULTI-DSP PER ANALISI ON-LINE DI SEGNALI RADIOASTRONOMICI.

S. Montebugnoli, C. Bortolotti, S. Buttaccio, A. Cattani,
N. D'Amico, G. Grueff, A. Maccaferri, G. Maccaferri,
A. Orfei, M. Roma, M. Tugnoli, G. Tuccari (IRA -Bo-)

Sergio Tassinari (Facolta` di Ingegneria -Bo-)
Paolo Vendruscolo (Facolta`di Ingegneria -Bo-)

Ira 197/95

(file: sharp.doc)

Tutti i Files sono nella directory: \Windows

PREMESSA

In questa nota interna viene presentata una serie di considerazioni sul chip set SHARP LH9124/LH9320 e sull' architettura di una scheda multi DSP, orientata alla trattazione on-line di segnali radioastronomici. Tutto questo e` stato oggetto anche di due tesi presso la facoltà di Ingegneria dell'Universita` di Bologna (di cui uno degli scriventi è stato correlatore) a cui si farà molto spesso riferimento. Prima di affrontare l' argomento oggetto della nota tecnica ci si soffermerà sugli algoritmi FFT (computo veloce della DFT) per i quali il chip set sopra menzionato e` particolarmente efficiente.

INTRODUZIONE

Agli inizi degli anni 90, la SHARP ha immesso sul mercato un chip-set, incredibilmente avanzato nei confronti dei DSP esistenti, costituito dai chips LH9124 e LH9320. L' architettura interna è completamente diversa, come si vedrà più avanti, dagli altri DSP commerciali e le prestazioni raggiunte sono notevoli. Il chip set può essere vantaggiosamente impiegato in tutte quelle applicazioni in cui si rendono necessarie elaborazioni on-line che coinvolgono algoritmi richiedenti grosse quantità di calcolo. L' esempio più evidente è il calcolo della FFT on line, in applicazioni tipo spettroscopia ad elevata risoluzione temporale e frequenziale, misura della scintillazione nello spazio interstellare, studio della variabilità su tempi scala molto brevi di Maser, timing di precisione di Pulsars ed, infine, SETI. Essendo, comunque, la scheda completamente programmabile, non dovrebbe essere difficile renderla idonea alla soluzione di qualsiasi problema osservativo in cui venga richiesta l' elaborazione al volo dei dati.

CAP. 1

L' ANALISI DI SPETTRO

Prima di affrontare l' argomento di questa nota tecnica, soffermiamoci un attimo sul problema dell' analisi di spettro (in appendice A si entera`un po' piu`in dettaglio). L' output del radiotelescopio è rappresentato da un segnale funzione del tempo V(t) e quello che si vuole ottenere e` la densità spettrale di potenza P(w) di V(t). E' possibile seguire due diversi procedimenti per arrivare a questo, cioe`:

$$P(f) = \int_{-\infty}^{+\infty} R(\tau) e^{-j2\pi f\tau} d\tau$$

dove $R(\tau)$ rappresenta la funzione di autocorrelazione ottenuta come in fig.1a:

$$R(\tau) = \lim_{T_0 \to \infty} \frac{1}{T_0} \int_{-\frac{T_0}{2}}^{+\frac{T_0}{2}} V(t)\, V(t+\tau)\, dt$$

oppure:

$$P(f) \propto V^2(f)$$

dove V(f) data da:

$$V(f) = \int_{-\infty}^{+\infty} V(t) e^{-j2\pi ft} dt$$

è una funzione complessa che rappresenta la trasformata di Fourier del segnale. Nella pratica, i due diversi procedimenti sopra riportati sono affrontati dopo che il segnale radioastronomico è stato convertito in digitale, per cui le stesse relazioni dovrebbero essere riscritte considerando i segnali campionati. In particolare la funzione di autocorrelazione (1b) assume la forma:

$$R(n) = \sum_{k=-\infty}^{\infty} V(k) V(n+k)$$

dove k ed n rappresentano rispettivamente t e τ del caso precedente (nella pratica k copre un range limitato di valori); mentre la trasformata di Fourier, che in questo caso assume il nome di DFT (Discrete-Fourier-Transform), diventa:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N} \qquad \text{k=0,1,2,3,...N-1} \qquad (1)$$

per cui lo spettro di potenza nel primo caso e` dato da:

THE ANALOG IMPLEMENTATION OF CORRELATION

FOR ANALOG IMPLEMENTATION, THE HARDWARE SOLUTION IS SOMEWHAT CLUMSY

- DELAY → DELAY LINES
- STORED SIGNAL → ANALOG STORAGE
- MULTIPLICATION → OK
- INTEGRATION → OK

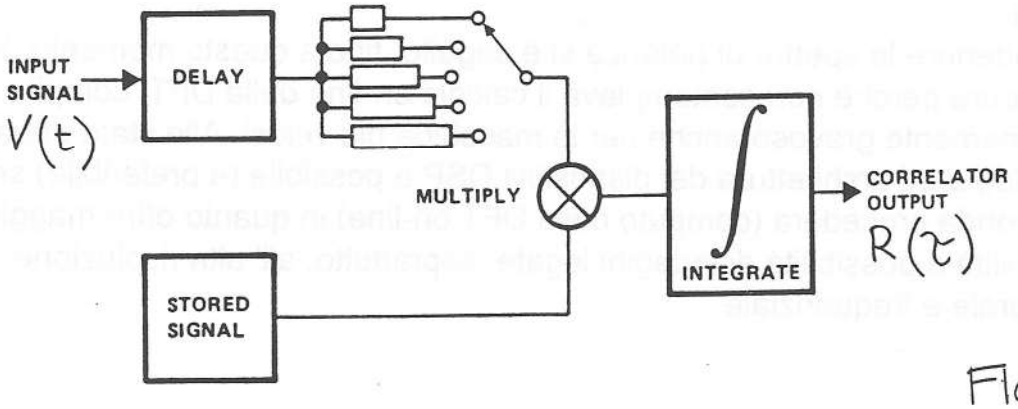


FIG. 1a

HARDWARE IMPLEMENTATION OF A CORRELATOR

DIGITAL IMPLEMENTATION OF CORRELATION

FOR DIGITAL IMPLEMENTATION, HARDWARE IS STRAIGHTFORWARD, A/D CONVERTER NOT ALWAYS NEEDED.

SIMPLE LOGIC BLOCKS SUCH AS

- SHIFT REGISTERS
- EXCLUSIVE NORs
- SUMMER → FULL ADDERS AND HALF ADDERS

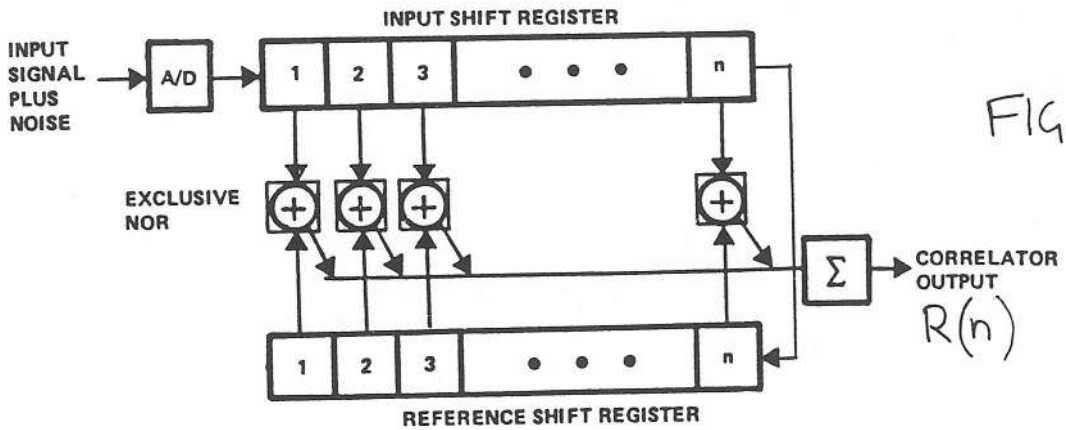


FIG. 1b

$$X(k)^2 = \sum_{n=0}^{N-1} R(n) e^{-j2\pi nk/N}$$

e nel secondo caso:

$$|X(k)^2| = X_r^2(k) + X_i^2(k)$$

La valutazione dello spettro di potenza data sopra nel caso in cui il segnale campionato sia reale e non deterministico, non e` proprio rigorosa ma puo`comunque dare una buona indicazione sullo spettro di potenza del segnale, specialmente se si prende, come risultato, la media di un certo numero di spettri.

Per ottenere lo spettro di potenza si è seguito, fino a questo momento, la prima procedura perchè non contemplava il calcolo on-line della DFT, compito estremamente gravoso anche per le macchine più veloci. Allo stato attuale della tecnologia ed architettura dei dispositivi DSP è possibile (e preferibile) seguire la seconda procedura (computo della DFT on-line) in quanto offre maggiori flessibilità e possibilità di indagini legate, soprattutto, all' alta risoluzione temporale e frequenziale.

LA FAST FOURIER TRANSFORM

Vista l' importanza che ricopre, in generale, la DFT nel signal processing, e quindi la necessita` di computarla on line, vediamo come velocizzarne il calcolo; l' algoritmo che permette questo si chiama FFT (Fast Fourier Transform). Consideramo l' espressione (1) e poniamo:

$$W_N = e^{-j2\pi/N} \qquad (1.a)$$

che inserito nella (1) fornisce :

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \qquad k=0,1,2,3,....N-1 \qquad (2)$$

Consideriamo la espressione (2) della DFT e, per generalità, supponiamo che sia la funzione x(n) che X(k) siano complesse. Per ogni singolo valore di X(k), nella (2) vengono richieste N moltiplicazioni ed (N-1) addizioni per cui, considerando che k assume N valori, il computo globale della DFT richiede N*N moltiplicazioni complesse ed N(N-1) addizioni complesse, supponendo che i coefficienti W_N^{nk} (detti twiddle factor) siano già stati precedentemente calcolati e memorizati in un opportuno buffer di memoria. Da queste considerazioni scaturisce che il numero di operazioni da eseguire ci fornisce un indice di complessità computazionale dell' algoritmo e, come conseguenza, del tempo necessario per il calcolo. Il calcolo della DFT come sopra riportato, richiede un tempo che è proporzionale a N^2, per cui è impensabile calcolare direttamente la DFT in casi in cui il numero dei campionamenti N supera i 1000. Qualche miglioramento (max di un fattore 2) si era ottenuto sfruttando la simmetria della DFT e periodicità dei coefficienti, senza però ridurre sensibilmente il tempo richiesto. Nel 1965, finalmente, venne presentato da Cooley e Tukey un algoritmo per il calcolo veloce ed efficiente della serie di Fourier, che diede una forte spinta allo sviluppo di una certa varietà di nuovi ed efficienti algoritmi per il calcolo della DFT. Il principio fondamentale su cui tutti i suddetti algoritmi sono basati, è quello di frazionare il calcolo della DFT degli N campionamenti, in una serie di DFT via via più piccole e con modalità di suddivisione proprie di ogni tipo di algoritmo. In base alle sopradette modalità di decomposizione, si possono raggruppare gli algoritmi di FFT in due grandi categorie:

-1) DECIMAZIONE NEL TEMPO: la scomposizione in trasformate via via più piccole come dimensioni, avviene operando scomposizioni sulla serie di campionamenti in ingresso e, quindi, ancora nel dominio del tempo.

-2) DECIMAZIONE IN FREQUENZA: in questo caso è la sequenza di uscita X(k) della (2) ad essere scomposta.

Siccome il chip set della SHARP (su cui verte tutta la presente nota tecnica) ha una architettura ottimizzata per l' implementazione dell' algoritmo a decimazione nel tempo, si prenderà in considerazione proprio questo tipo di scomposizione. Se si assume che il numero di punti della serie di ingresso sia una potenza di 2, si può scomporre la serie di campionamenti in ingresso in due serie: una ad indici pari ed una ad indici dispari. Dopo questa scomposizione, la espressione della DFT assume la forma:

$$X(k) = \sum_{\text{n pari}} x(n) W_N^{nk} + \sum_{\text{n dispari}} x(n) W_N^{nk}$$

cioè, introducendo l' indice r=0,1,2,3.... (N/2)-1:

$$\begin{aligned} X(k) &= \sum_{r=0}^{(N/2)-1} x(2r) W_N^{2rk} + \sum_{r=0}^{(N/2)-1} x(2r+1) W_N^{(2r+1)k} \\ &= \sum_{r=0}^{(N/2)-1} x(2r)(W_N^2)^{rk} + W_N^k \sum_{r=0}^{(N/2)-1} x(2r+1)(W_N^2)^{rk} \end{aligned} \qquad (3)$$

Dalle (1.a) si ha:

$$W_N^2 = e^{-2j(2\pi/N)} = e^{-j2\pi/(N/2)} = W_{N/2}$$

che sostituito nella (3) porta alla espressione:

$$X(k) = \sum_{r=0}^{(N/2)-1} x(2r) W_{N/2}^{rk} + W_N^k \sum_{r=0}^{(N/2)-1} x(2r+1) W_{N/2}^{rk} \qquad (4)$$

Si definiscano ora due sequenze di (N/2) punti a partire dagli elementi pari e dispari di x(r):

$$g(r) = x(2r)$$
$$h(r) = x(2r+1)$$

queste sostituite nella (4) danno:

$$\begin{aligned} X(k) &= \sum_{r=0}^{(N/2)-1} g(r) W_{N/2}^{rk} + W_N^k \sum_{r=0}^{(N/2)} h(r) W_{N/2}^{rk} \\ &= G(k) + W_N^k H(k) \end{aligned} \qquad (5)$$

I termini G(K) e H(k) si possono quindi considerare come DFT dei campioni rispettivamente pari e dispari di N/2 punti della serie originaria. Osserviamo però che la trasformata X(k) è definita per k=0,1,2,3,...N-1 mentre le G(k) e H(k) sono definite per k=0,1,2,3,...(N/2)-1, per cui occorre dare una interpretazione della (5) per valori di k superiori ad N/2. Sfruttando la periodicità della DFT e

ricordando che G(k) e H(k) hanno periodo pari a N/2 per il fatto che sono DFT di N/2, si può scrivere:

$$X(k) = \begin{cases} G(k) + W_N^k H(k) & k = 0,1,2,3,.....(N/2)-1 \\ G(k-N/2) + W_N^k H(k)(k-N/2) & k = N/2,.....N-1 \end{cases} \quad (6)$$

Proviamo a fare un esempio pratico in cui si consideri N=8. X(0) si ottiene sommando a G(0) il termine H(0) moltiplicato per W_8^0, analogamente X(1) si ottiene sommando a G(1) il prodotto $H(1)W_8^1$ e così via fino a 4. Arrivati all' indice 4, X(4) dovrebbe essere uguale a $G(4)+W_8^4 H(4)$, però G(k) e H(k) hanno periodo uguale a 4 di conseguenza G(4)=G(0) e H(4)=H(0) per cui X(4) risulta essere uguale alla somma di G(0) con il prodotto $H(0)W_8^4$. Possiamo quindi scrivere:

$X(0) = G(0) + W_8^0 H(0)$

.....

$X(3) = G(3) + W_8^3 H(3)$

$X(4) = G(0) + W_8^4 H(0)$

.....

$X(7) = G(3) + W_8^7 H(3)$

Considerando poi che $W_N^{k+N/2} = -W_N^k$, la (6) si può scrivere:

$$X(k) = \begin{cases} G(k) + W_N^k H(k) & k = 0,1,2.....(N/2)-1 \\ G(k-N/2) - W_N^{k-N/2} H(k-N/2) & k = N/2,.....N-1 \end{cases}$$

Quanto sopra esposto, risulta essere più chiaro se rappresentato graficamente (fig1). A questo punto si può fare una prima considerazione, a livello di efficienza di calcolo, tra la (1) e la (5) notando che nella (5) le operazioni richieste sono già di molto inferiori a quelle richieste dalla (1). Considerando che il numero di campionamenti (o punti) scelti è comunque pari perchè scelti essere una potenza di due, si può iterare il procedimento di suddivisione appena visto dividendo ulteriormente le sommatorie della (5) in due DFT da (N/4) punti e combinandole per poi riformare le originarie DFT da (N/2), come di seguito indicato:

$G(k) = A(k) + W_{N/2}^k B(k) = A(k) + W_N^{2k} B(k)$

$H(k) = C(k) + W_{N/2}^k D(k) = C(k) + W_N^{2k} D(k)$

con:
k=0,1,2,3,...(N/2)-1

A(k)=DFT dei (N/4) punti ad indice pari di g(n)
B(k)=DFT dei (N/4) punti ad indice dispari di g(n)
C(k)=DFT dei (N/4) punti ad indice pari di h(n)
D(k)=DFT dei (N/4) punti ad indice dispari di h(n)

come rappresentato nello schema di fig.2. Il procedimento di scomposizioni successive sopra visto, può poi essere continuato fino a quando siano rimaste da calcolare DFT di soli due punti. Nel caso del nostro esempio con N=8, questo è visibile in fig 3. Si nota, quindi, che l' unità minima è la DFT di 2 punti che può essere calcolata, in pratica, senza moltiplicazioni. Di fatto se N=2 e k=0,1, la (5) si può scrivere:

$$X(0) = x(0) + W_2^0 x(1)$$
$$X(1) = x(0) + W_2^1 x(1)$$

ma ricordando che $W_N = e^{-j2\pi/N}$, si ha $W_2^0 = 0 \quad W_2^1 = -1$ per cui non sono richieste moltiplicazioni per il calcolo della DFT su due soli punti ma solo per combinare insieme la varie DFT in modo tale da ottenere quelle di ordine maggiore. L' algoritmo che prevede di scomporre, procedendo a ritroso, il set di N dati fino ad ottenere una serie di DFT di due soli punti, si chiama **radix-2 decimation in time** che ha, ovviamente, come blocco "operativo" di base la **radix-2 butterfly**. Quest' ultimo fornisce in uscita la combinazione dei due ingressi A e B come segue:

$$\begin{aligned} X &= A + W_N^k B \equiv X(0) \qquad k = 0 \\ Y &= A - W_N^k B \equiv X(1) \end{aligned} \tag{7}$$

In fig. 4 viene raffigurata graficamente la struttura base appena esposta. Nella (7) si nota che quando e` effettivamente necessario effettuare le moltiplicazioni (cioe` quando i coefficienti W_N^k non sono +/- 1 come accade dopo il primo passo radix-2) e` necessaria una sola moltiplicazione $W_N^k B$ in quanto il valore calcolato puo` essere memorizzato e riutilizzato; non solo ma i valori parziali di uscita dal blocco radix-2 possono essere rilocati nella stessa locazione di memoria dove erano allocati i dati, con conseguente risparmio di memoria (algoritmi in-place). Riguardo al risparmio del numero di moltiplicazioni richieste, si nota in fig. 4 che il numero degli stadi necessari per completare la FFT sono $\log_2 N$ all'interno dei quali occorrono N/2 moltiplicazioni complesse per ricombinare i dati in uscita dallo stadio precedente, per un totale di $(N/2)\log_2 N$. Di addizioni ne occorrono, invece, 2 per ognuno dei $\log_2 N$ stadi, per un totale di $N \log_2 N$. Questo rappresenta un grande miglioramento nei confronti del calcolo diretto della DFT che, per array molto grandi (da 128K in su), puo` raggiungere i 4 ordini di grandezza.

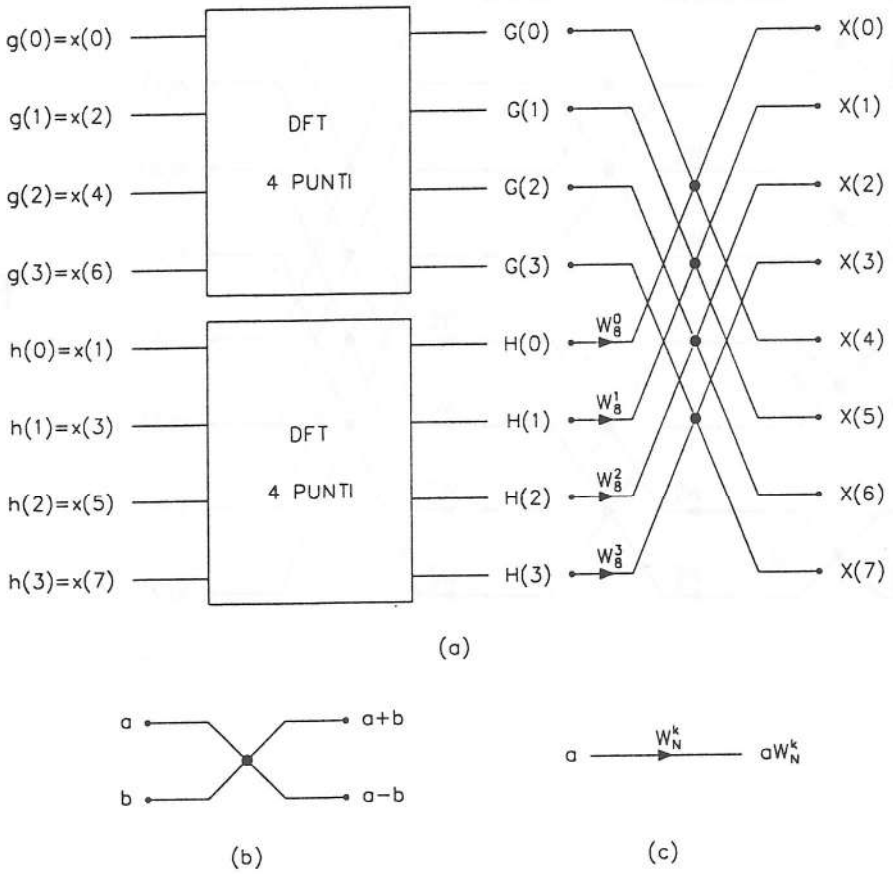

g(0)=x(0)
g(1)=x(2)
g(2)=x(4)
g(3)=x(6)
DFT
4 PUNTI
h(0)=x(1)
h(1)=x(3)
h(2)=x(5)
h(3)=x(7)
DFT
4 PUNTI
G(0)
G(1)
G(2)
G(3)
H(0)
H(1)
H(2)
H(3)
W_8^0
W_8^1
W_8^2
W_8^3
X(0)
X(1)
X(2)
X(3)
X(4)
X(5)
X(6)
X(7)
(a)
a
b
a+b
a−b
(b)
a
W_N^k
aW_N^k
(c)

FIG. 1

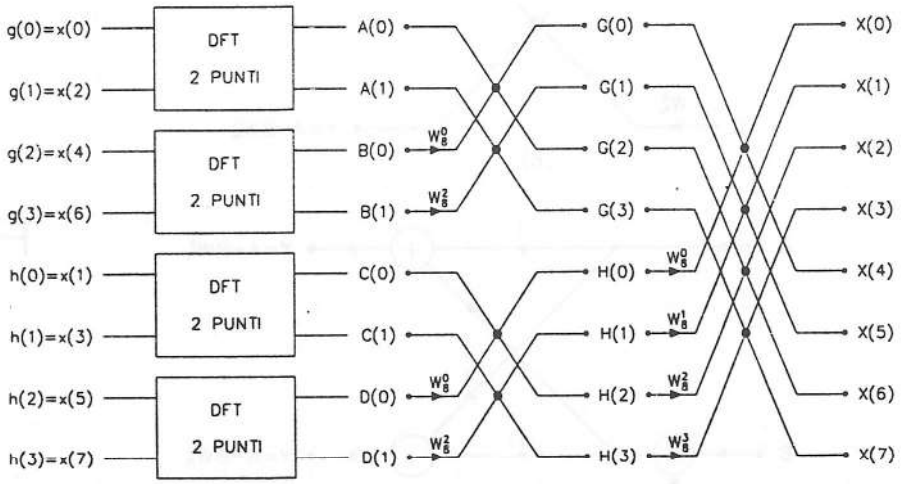

g(0)=x(0)
g(1)=x(2)
g(2)=x(4)
g(3)=x(6)
h(0)=x(1)
h(1)=x(3)
h(2)=x(5)
h(3)=x(7)
DFT
2 PUNTI
DFT
2 PUNTI
DFT
2 PUNTI
DFT
2 PUNTI
A(0)
A(1)
B(0)
B(1)
C(0)
C(1)
D(0)
D(1)
W_8^0
W_8^2
W_8^0
W_8^2
G(0)
G(1)
G(2)
G(3)
H(0)
H(1)
H(2)
H(3)
W_8^0
W_8^1
W_8^2
W_8^3
X(0)
X(1)
X(2)
X(3)
X(4)
X(5)
X(6)
X(7)

FIG. 2

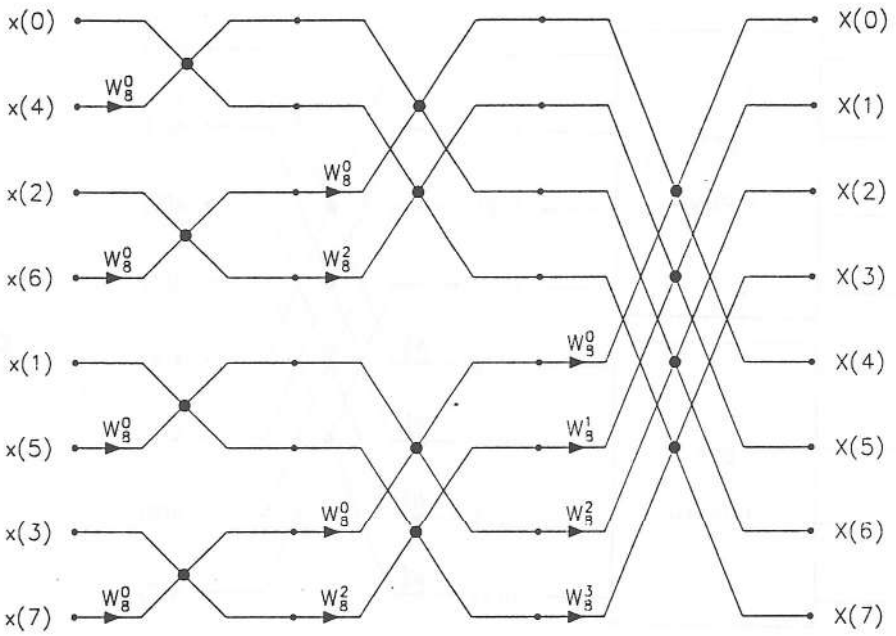

x(0)
x(4)
x(2)
x(6)
x(1)
x(5)
x(3)
x(7)
X(0)
X(1)
X(2)
X(3)
X(4)
X(5)
X(6)
X(7)

FIG. 3

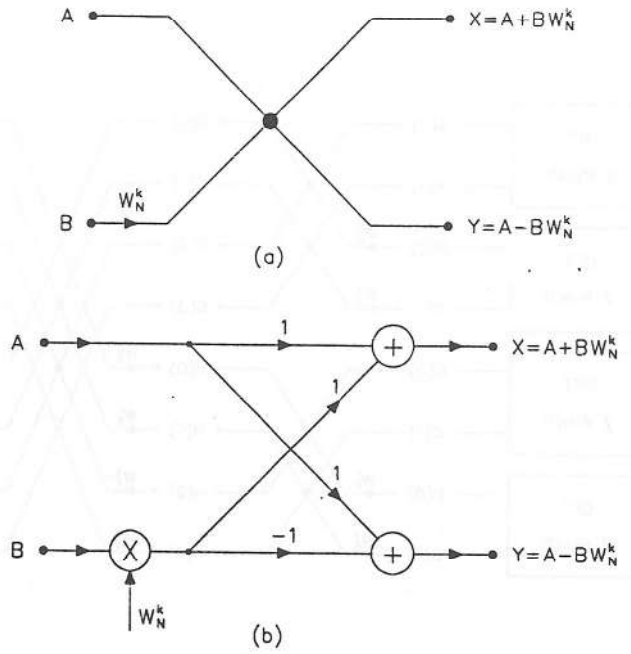

A
B
X=A+BW_N^k
Y=A−BW_N^k
(a)
1
−1
(b)

FIG. 4

UN ALGORITMO PER FFT SU DATI REALI.

L'algoritmo della FFT appena visto, opera con dati complessi in ingresso, nella pratica, pero`, si ha a che fare con dati reali che provengono da un convertitore A/D. Questo algoritmo sara`affrontato anche piu` avanti alla luce della conoscenza delle funzioni implementate a bordo del DSP LH9124.
Come primo approccio al problema, si potrebbe pensare di riempire con tutti zeri la parte immaginaria, ma questo porterebbe ad una efficienza dell' algoritmo molto bassa per il fatto che si obbligherebbe il DSP ad effettuare una certa quantita` di calcoli inutilmente. Il problema si risolve facendo effettuare una DFT di 2N punti reali usando una FFT di N punti complessi. Considerando un array di 2N punti reali con N pari ad una potenza di 2, lo si suddivida in due array singoli h(n) e g(n), formati rispettivamente da punti ad indice pari e dispari, cioe`:

$$\begin{aligned} h(n) &= x(2n) \\ g(k) &= x(2n,+1) \end{aligned} \qquad n = 0,1,2,3,\ldots N-1 \tag{8}$$

Considerando la formula base della DFT (2) e tenendo conto della espressione (8), si puo` scrivere:

$$\begin{aligned} X(k) &= \sum_{n=0}^{2N-1} x(n) W_{2N}^{nk} \\ &= \sum_{n=0}^{N-1} x(2n) W_{2N}^{2nk} + \sum_{n=0}^{N-1} x(2n+1) W_{2N}^{(2n+1)k} \\ &= \sum_{n=0}^{N-1} h(n) W_N^{nk} + W_{2N}^{k} \sum_{n=0}^{N-1} g(n) W_N^{nk} \end{aligned} \tag{9}$$

Le due sommatorie che compongono l' ultima espressione della (9) si possono considerare come due DFT H(k) e G(k) di N punti, per cui e` possibile scrivere:

$$X(k) = H(k) + W_{2N}^{k} G(k) \tag{9.a}$$

Si definisce ora una funzione complessa che ha come parte reale i campioni ad indice pari della x(n) e come parte immaginaria quelli ad indice dispari:

$$y(n) = h(n) + jg(n) \tag{9.b}$$

La DFT di questa funzione complessa, come gia`visto nella (9), risulta essere:

$$Y(k)=\sum_{n=0}^{N-1} y(n)W_N^{nk}$$
$$=\sum_{n=0}^{N-1} h(n)W_N^{nk} + j\sum_{n=0}^{N-1} g(n)W_N^{nk}$$
$$= H(k) + jG(k)$$

scomponendo prima in parte reale ed immaginaria poi raccogliendo le parti reali ed immaginarie di H(k) e G(k) si ha:

$$Y(k)=[H_r(k)+jH_i(k)]+j[G_r(k)+jG_i(k)]$$
$$=[H_r(k)-G_i(k)]+j[H_i(k)+G_r(k)] \qquad (10)$$

Seguendo lo stesso ragionamento che ci ha portato alla precedente espressione (10), si puo` arrivare alla stessa espressione per la trasformata Y(N-k)

$$Y(N-k)=\sum_{n=0}^{N-1} y(n)W_N^{n(N-k)}$$
$$=\sum_{n=0}^{N-1} h(n)W_N^{-nk} + j\sum_{n=0}^{N-1} g(n)W_N^{-nk}$$
$$= H^*(k) + jG^*(k)$$

che porta a definire, come nel caso della (10), l'espressione:

$$Y(N-k)=[H_r(k)+jH_i(k)]+j[G_r(k)+jG_i(k)$$
$$=[H_r(k)-G_i(k)]+j[H_i(k)+G_r(k)] \qquad (10.a)$$

Dato che dalle formule di Eulero ($e^{j\alpha}=\cos\alpha+j\,\mathrm{sen}\,\alpha$) risulta essere:

$$W_{2N}^{k} = e^{-j\pi k/N} = \cos(\pi k/N) - j\,\mathrm{sen}(\pi k/N)$$

è possibile scrivere la parte reale $X_r(k)$ ed immaginaria $X_i(k)$ di $X(k)$, che altro non è che la FFT della sequenza di dati x(n) di lunghezza 2N, combinando la (9.a),(10) e (10.a):

$$X_r(k)=\frac{1}{2}\mathrm{Re}[y(k)+y(N-k)]$$
$$+\frac{1}{2}\cos(\frac{\pi k}{N})\,\mathrm{Im}[Y(k)+Y(N-k)] \qquad (11)$$
$$-\frac{1}{2}sin(\frac{\pi k}{N})\,\mathrm{Re}[Y(k)-Y(N-k)]$$

$$X_i(k) = \frac{1}{2}\mathrm{Im}[y(k) - y(N-k)]$$
$$-\frac{1}{2}\cos(\frac{\pi k}{N})\,\mathrm{Re}[Y(k) - Y(N-k)] \qquad (12)$$
$$-\frac{1}{2}sin(\frac{\pi k}{N})\,\mathrm{Im}[Y(k) + Y(N-k)]$$

In pratica per il calcolo della FFT di un array di dati reali x(n) di lunghezza 2N si deve scomporre la funzione x(n) in due componenti h(n) e g(n), come in (8), per formare l' equazione complessa (9.b) tramite la cui FFT Y(t) è possibile risalire alle due componenti reali ed immaginarie di X come in (11) e (12).

Il fatto di trattare array di dati reali porta, oltre che alla definizione del precedente algoritmo, anche ad alcune considerazioni. La piu` immediata e` che la parte reale ed immaginaria sono simmetriche attorno al punto di indice N/2 e solo la prima meta` dei dati (dati con indice compreso tra 0 e N/2-1) e` significativa per il fatto che quelli con indice compreso tra N/2 e N-1 rappresentano il risultato della DFT a frequenze "negative" e quindi senza un vero significato fisico; infatti una DFT di 256 K ci fornisce solo 128 K canali. Altra considerazione e data dal fatto che la DFT e` una approssimazione della trasformata "continua" di Fourier. Questa la approssima tanto piu' quanto minore e` l'intervallo di campionamento T per il fatto che i risultati (canali) in uscita della DFT (FFT) sono spaziati fra di loro di $\Delta B = 1/NT$ (risoluzione). La "finestra" NT e` chiamata time-record e dovrebbe essere scelta in modo tale da coprire almeno un ciclo del segnale periodico in esame. Se questo e` aperiodico (oppure periodico ma troncato non esattamente ad un numero intero di periodi), la ripetizione periodica dei campioni risultante dal troncamento (finestra rettangolare) genera delle componenti frequenziali che, in realta`, non erano presenti nel segnale originario. Queste risposte spurie possono deteriorare lo spettro "vero" del segnale originario per cui e` necessario, visto che non si possono eliminare completamente, minimizzarne l'effetto (Spectral Leakage). Questo e` possibile "pesando" i dati campionati con opportune funzioni di "windowing" (ne esistono di molti tipi ognuna delle quali ha caratteristiche proprie) che minimizzano le discontinuita` nella funzione periodica. Questo procedimento, pero', deteriora la risoluzione del sistema in quanto il lobo principale relativo alla particolare window e`sempre piu` largo di quello della finestra rettangolare che (a parita`di time-record) presenta quello piu` stretto in assoluto, anche se affetto da lobi secondari alti. Nel Prossimo paragrafo si affrontera`in maggior dettaglio il problema del windowing.

WINDOWING DEI DATI.

Per mettere a fuoco meglio il problema della "finestratura" dei dati, consideriamo ora alcuni risultati ottenuti simulando una determinata finestra con il package software MathCad. La finestra adottata e' quella tipo Blackman-Harris che e' una di quelle che forse fornisce i migliori risultati per l' ampiezza dei suoi lobi secondari ad oltre 90 dB sotto quello principale. L' espressione di tale finestra risulta essere:

$$w(n) = a_0 - a_1 \cos(\frac{2\pi n}{N}) + a_2 \cos(\frac{4\pi n}{N}) - a_3 \cos(\frac{6\pi n}{N})$$

con

$a_0 = 0.35875$ $\quad a_1 = 0.48829$

$a_2 = 0.14127$ $\quad a_3 = 0.01168$

ed il numero di punti su cui si e' effettuato il calcolo della FFT ed il relativo modulo quadro (sempre con MathCad), e' di 1024. Il segnale, poi, da cui si sono ottenuti i 1024 campionamenti risulta essere un segnale reale generato internamente allo stesso pacchetto software. In fig.5 viene riportato lo spettro di potenza di un segnale formato da due sinusoidi la cui frequenza cade esattamente dentro a due canali, canale 128 e 256, di cui una piu' alta di ben 60 db dell'altra ed ottenuto con una finestra rettangolare. Da questa figura si puo' notare come le due righe siano ben visibili e questo perche' il fatto che cadono dentro a due canali, corrisponde a troncare la funzione di ingresso esattamente ad un numero intero di cicli. Nella fig.5a viene riportato lo stesso spettro di potenza dopo avere avvicinato leggermente il tono ad ampiezza maggiore all' altro, mantenendo la stessa finestra rettangolare del caso precedente. In questo caso tale riga non cade piu' esattamente dentro al canale 128 come prima, ma tra un canale e l' altro per cui questo non rientra piu' nella finestra rettangolare con un numero intero di cicli. Avendo mantenuto la stessa scala del caso precedente, in fig.5a si puo' notare come questa situazione si traduca in una discontinuita' che, oltre a causare la nascita di componenti non "vere", introduce una una notevole perdita. Lo spettro originato dalle discontinuita' dovute al troncamento maschera in maniera determinante la riga prodotta dal tono ad ampiezza minore, al punto da non renderla assolutamente visibile.

Per ovviare a questo tipo di problema, come gia'accennato in precedenza, si effettua nel dominio del tempo, cioe' prima di trasformare, una finestratura dei dati ottenuta moltiplicandoli per una determinata funzione detta di windowing. Questo porta a cambiare totalmente il profilo della finestra che da quello rettangolare passa ad un profilo che decresce ai lati fino ad arrivare a zero in maniera molto "dolce, ai bordi". Se si effettua il windowing dei dati nella situazione precedente (tono che non cade esattamente entro un canale) con la

window Blakman-Harris, si ottiene la fig.5b. Osservando questa figura, si puo' subito dire che la perdita dovuta al troncamento e' ancora presente pero` in maniera molto meno marcata (circa -100 dB) al punto da "scoprire" la riga dovuta al tono di ampiezza minore, e questo grazie al processo di windowing. Come gia` detto nel paragrafo precedente. la finetra rettangolare, cioe` l' acquisizione naturale dei dati senza windowing, ha il potere risolutivo migliore per cui l' uso di una qualsiasi finestra diversa da quella rettangolare introduce una perdita di risoluzione. Per mostrare questo, prendiamo in considerazione la fig.5c dove e` rappresentato il grafico riguardante lo spettro con window di tipo Blackman-Harris dei due toni di cui lo spettro senza windowing (cioe` window rettangolare) e` riportato nella fig.5. La caratteristica piu` evidente e` che le righe relative ai due toni nello spettro "finestrato" di fig.5c appaiono molto piu` larghe delle stesse righe relative allo stesso spettro non finestrato di fig.5. Si puo` quindi affermare che l'uso di una qualsiasi finestra diversa da quella rettangolare, anche se di notevole aiuto per minimizzare gli effetti del troncamento, causa una perdita di risoluzione. Questo concetto e` ancora piu` evidente se si considerano i due spettri riportati nelle fig. 5d e 5e relativi a due toni molto vicini in frequenza ma all' interno di canali diversi della FFT, calcolati facendo uso il primo della window rettangolare ed il secondo di quella Blackman-Harris . Nel primo caso i toni sono nettamente distinguibili, nel secondo, essendosi allargate le righe, diventa difficile distinguerle per il fatto che tendono a confondersi tra loro. Per concludere, occorre quindi analizzare caso per caso per scegliere il tipo di finestra piu` adatta; in questo ultimo caso riportato come esempio, e` chiaro che la finestra piu` adatta da apllicare e` quella rettangolare. Per completezza, si riportano qui di seguito le principali forme e caratteristiche delle windows più usate. Si analizzano prima i significati di alcuni termini riportati nelle tabelle:

1- **3 db bandwidth**: valore della frequenza a cui la window $|W((j\omega)|$si riduce di 3 db in riferimento al proprio valore DC W(0).

2- **Asymptotic roll-off**: pendenza dell' inviluppo $|W(j\omega)|$ alle alte frequenze.

3- **Peak Side-lobe level**: livello massimo dei lobi laterali, relativo a W(0).

4- **6 dB bandwidth**: frequenza a cui $|W((j\omega)|$si riduce di 6 dB in riferimento al livello DC di W(0).

5- **Equivalent noise bandwidth**: larghezza di banda di un filtro passa basso ideale (cioè con $|H(j\omega)|=1$ per $\omega<B$ e 0 ovunque) che passa la stessa quenatità di energia che attraverso la window.

6- **Choerent signal gain**: DC gain W(0) della finestra.

7- **Processing gain**: rapporto del SNR in uscita con il SNR di ingresso.

8- **Scalopping loss**: rapporto tra la metà del bin | $W(j\omega_x/2N)$| e W(0).

9- **Worst case processing gain**: processing gain (in dB) meno il valore dello scalopping loss al quadrato (in dB).

Vengono, qui di seguito, riportate anche le funzioni e le shapes delle varie windows.

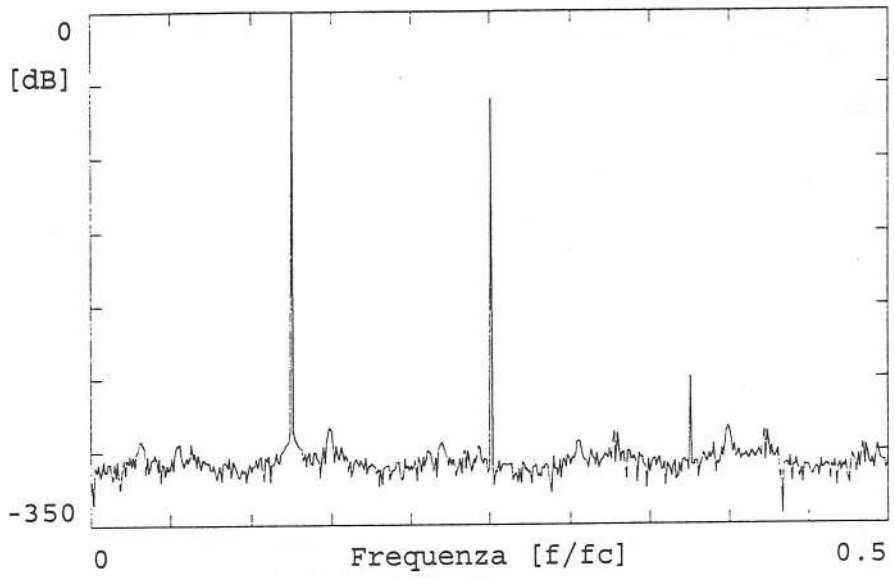


Figura 5 - Modulo della FFT su 1024 punti di un segnale composto da due toni sinusoidali, di frequenza 0.128fc (0 dB) e 0.256fc (-60 dB). Finestra rettangolare.

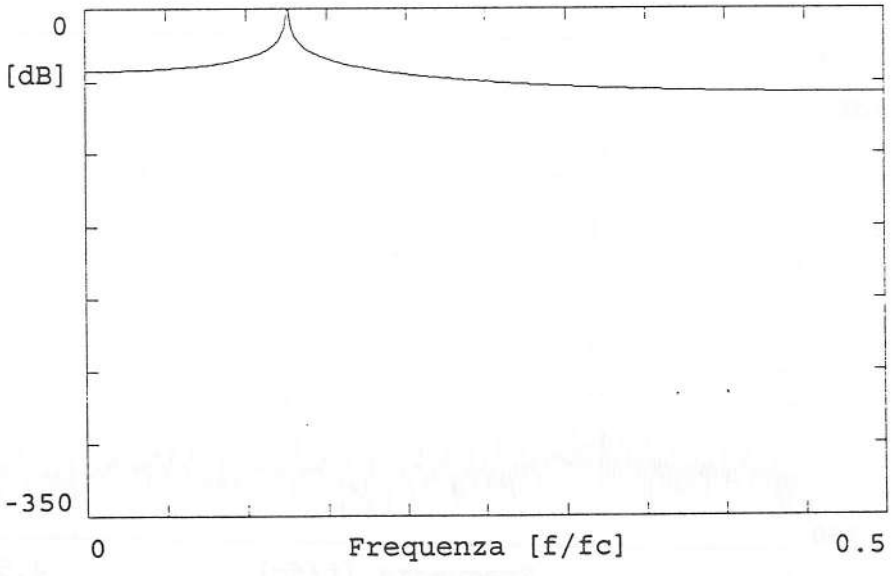


Figura 5a - Come Fig. 5 , ma con segnale a due toni di frequenza 0.1285fc (0 dB) e 0.256fc (-60 dB). Finestra rettangolare.

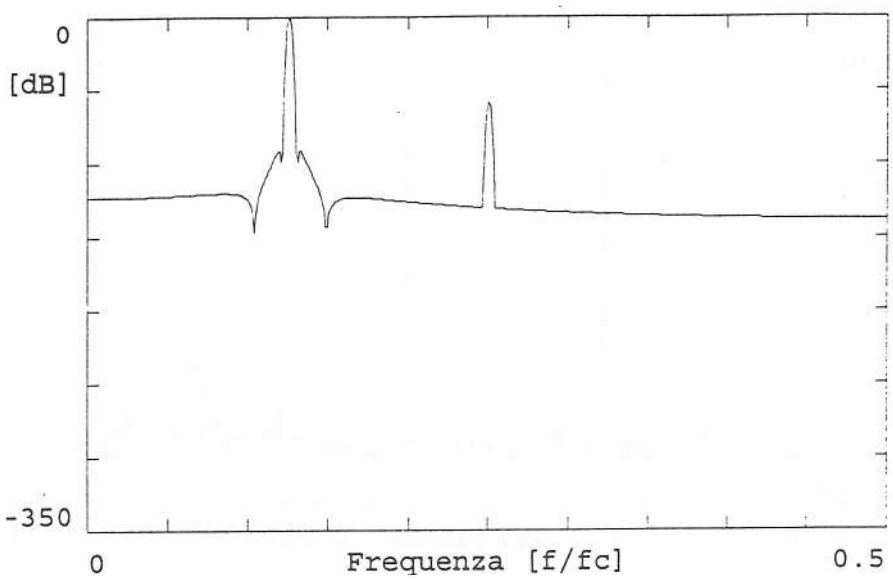


Figura 5b - Come Fig. 5a (stesso segnale), ma con finestra di Blackman-Harris.

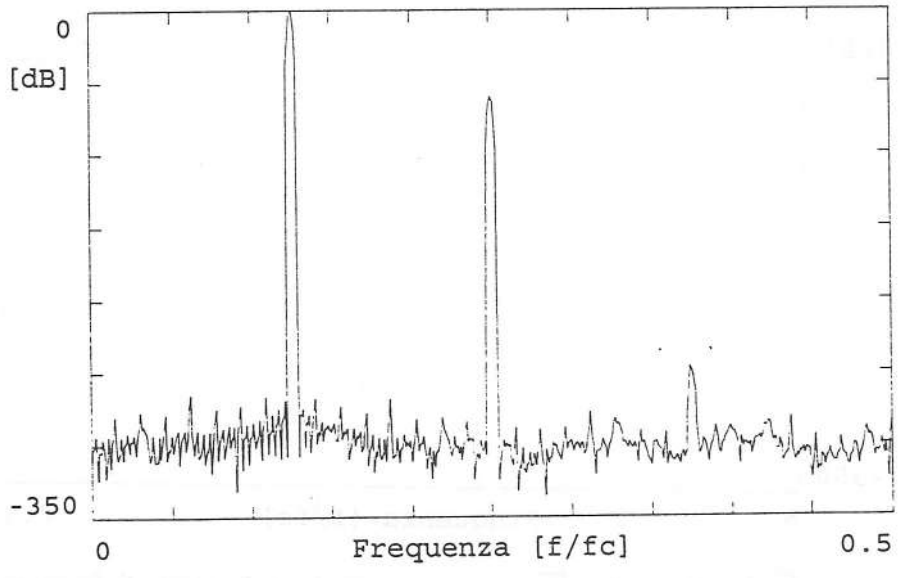


Figura 5c - Come Fig. 5 (stesso segnale), ma con finestra di Blackman-Harris.

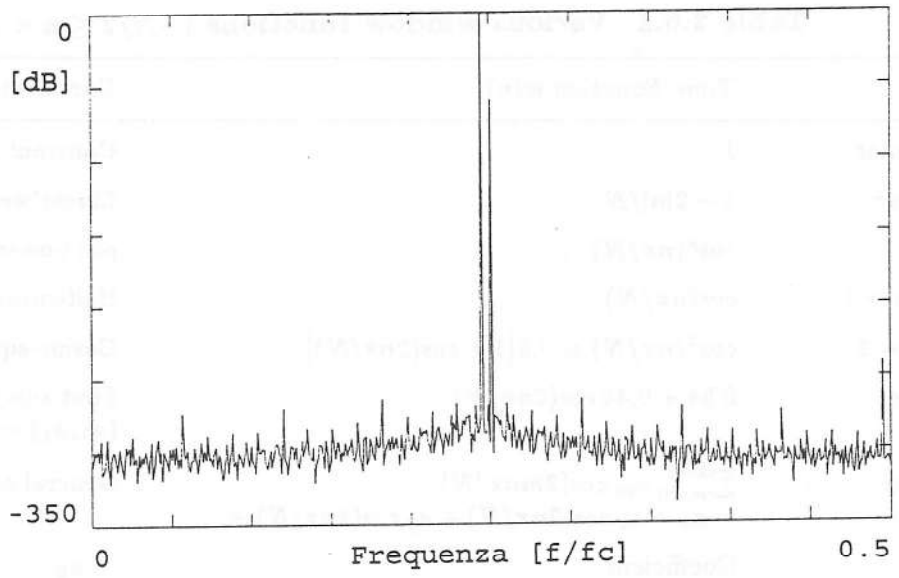


Figura 5d - Modulo della FFT su 1024 punti di un segnale composto da due toni sinusoidali, di frequenza 0.250fc (0 dB) e 0.256fc (-60dB). Finestra rettangolare.

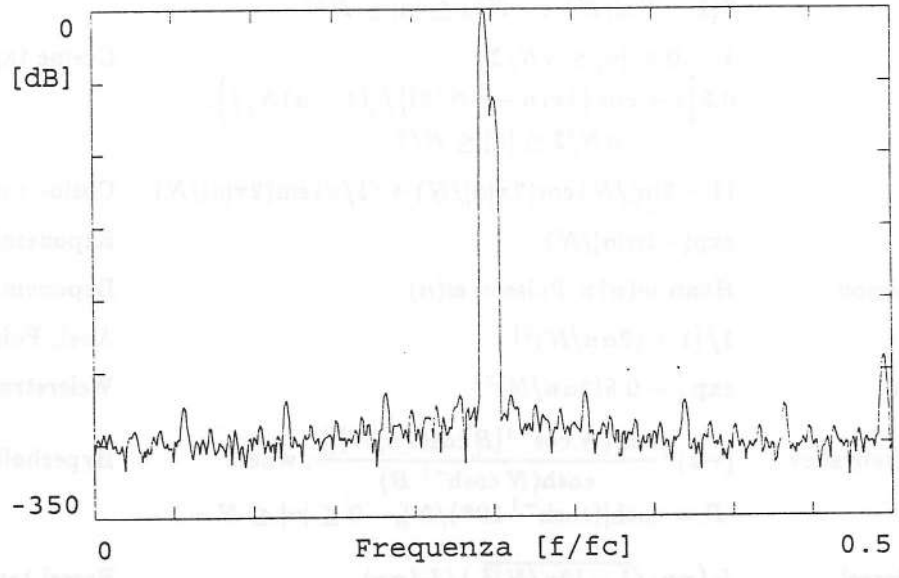


Figura 5c - Come Fig. 5d (stesso segnale), ma con finestra di Blackman-Harris.

Table 2.6.1 Various window functions ($-N/2 \le n < N/2$).[5]

Name	Time Function $w(n)$	Comments
Rectangular	1	Constant weighting [6]
Triangular	$1 - 2\lvert n\rvert/N$	Linear weight (Bartlett, Fejer) [7]
Cosine^p	$\cos^p(n\pi/N)$	pth power of cosine pulse
Cosine, $p = 1$	$\cos(n\pi/N)$	Half-cosine pulse
Hann, $p = 2$	$\cos^2(n\pi/N) = 0.5\left[1 + \cos(2n\pi/N)\right]$	Cosine-squared, raised-cosine [7]
Hamming	$0.54 + 0.46\cos(2n\pi/N)$	First side-lobe cancelled for $(a_0, a_1) = (0.54349, 0.45651)$ [7]
Blackman	$\sum_{m=0}^{k-1} a_m \cos(2mn\pi/N)$ $= a_0 + a_1\cos(2n\pi/N) + a_2\cos(4n\pi/N) + \ldots$	General cosine series expansion [8]
	Coefficient	a_0 a_1 a_2 a_3
	3-term (−67 dB)	0.42323 0.49755 0.07922 —
	3-term (−61 dB)	0.44959 0.49364 0.05677 —
	3-term (−58 dB, −18 dB/dec roll-off)	0.42 0.50 0.08 —
	3-term (−51 dB, cancels 2nd & 3rd side-lobes)	0.42659 0.49656 0.07685 —
	4-term (−92 dB)	0.35875 0.48829 0.14128 0.01168
	4-term (−74 dB)	0.40217 0.49703 0.09392 0.00183
Riesz	$1 - (2n/N)^2$	Parabolic [9]
Riemann	$\left[\sin(2n\pi/N)\right]/(2n\pi/N)$	Main lobe of sinc function [10]
de la Valle-Poussin	$1 - 6\left[1 - 2\lvert n\rvert/N\right]\left[2n/N\right]^2$, $0 \le \lvert n\rvert \le N/4$ $2\left[1 - 2\lvert n\rvert/N\right]^3$, $N/4 \le \lvert n\rvert \le N/2$	Jackson, Parzen, piece-wise cubic [9]
Tukey	1, $0 \le \lvert n\rvert \le \alpha N/2$ $0.5\left[1 + \cos\left\{\left[\pi(n - \alpha N/2)\right]/\left[(1-\alpha)N\right]\right\}\right]$, $\alpha N/2 \le \lvert n\rvert \le N/2$	Cosine tapered over α% interval [11]
Bohman	$(1 - 2\lvert n\rvert/N)\cos(2\pi\lvert n\rvert/N) + (1/\pi)\sin(2\pi\lvert n\rvert/N)$	Cosine tapered [12]
Poisson	$\exp(-2\alpha\lvert n\rvert/N)$	Exponential [10]
Hann-Poisson	Hann $w(n)\times$ Poisson $w(n)$	Exponentially-weighted Hann
Cauchy	$1/\left[1 + (2\alpha n/N)^2\right]$	Abel, Poisson [13]
Gaussian	$\exp\left[-0.5(2\alpha n/N)^2\right]$	Weierstrass, normal [13]
Dolph-Chebyshev	$(-1)^n \dfrac{\cos\left(N\cos^{-1}[B\cos(\pi n/N)]\right)}{\cosh(N\cosh^{-1}B)}$, where $B = \cosh[(\cosh^{-1}10^{\alpha})/N]$, $0 \le \lvert n\rvert \le N-1$	Hyperbolic taper [14]
Kaiser-Bessel	$I_0\left(\pi\alpha\sqrt{1-(2n/N)^2}\right)/I_0(\pi\alpha)$, where $I_0(x) = 1 + \sum_{k=1}^{\infty}[(x/2)^k/k!]^2$	Bessel taper [15]
Barcilon-Temes	Prolate-spheroidal wave functions	Minimizes energy outside band of frequencies [16]

Table 2.6.2 Properties of window functions. (Courtesy of F.J. Harris.[5])

WINDOW	HIGHEST SIDE-LOBE LEVEL (dB)	SIDE-LOBE FALL-OFF (dB/OCT)	COHERENT GAIN	EQUIV. NOISE BW (BINS)	3.0-dB BW (BINS)	SCALLOP LOSS (dB)	WORST CASE PROCESS LOSS (dB)	6.0-dB BW (BINS)	OVERLAP CORRELATION (PCNT)	
									75% OL	50% OL
RECTANGLE	-13	-6	1.00	1.00	0.89	3.92	3.92	1.21	75.0	50.0
TRIANGLE	-27	-12	0.50	1.33	1.28	1.82	3.07	1.78	71.9	25.0
$\cos^{\alpha}(x)$ HANN $\alpha = 1.0$	-23	-12	0.64	1.23	1.20	2.10	3.01	1.65	75.5	31.8
$\alpha = 2.0$	-32	-18	0.50	1.50	1.44	1.42	3.18	2.00	65.9	16.7
$\alpha = 3.0$	-39	-24	0.42	1.73	1.66	1.08	3.47	2.32	56.7	8.5
$\alpha = 4.0$	-47	-30	0.38	1.94	1.86	0.86	3.75	2.59	48.6	4.3
HAMMING	-43	-6	0.54	1.36	1.30	1.78	3.10	1.81	70.7	23.5
RIESZ	-21	-12	0.67	1.20	1.16	2.22	3.01	1.59	76.5	34.4
RIEMANN	-26	-12	0.59	1.30	1.26	1.89	3.03	1.74	73.4	27.4
DE LA VALLE-POUSSIN	-53	-24	0.38	1.92	1.82	0.90	3.72	2.55	49.3	5.0
TUKEY $\alpha = 0.25$	-14	-18	0.88	1.10	1.01	2.96	3.39	1.38	74.1	44.4
$\alpha = 0.50$	-15	-18	0.75	1.22	1.15	2.24	3.11	1.57	72.7	36.4
$\alpha = 0.75$	-19	-18	0.63	1.36	1.31	1.73	3.07	1.80	70.5	25.1
BOHMAN	-46	-24	0.41	1.79	1.71	1.02	3.54	2.38	54.5	7.4
POISSON $\alpha = 2.0$	-19	-6	0.44	1.30	1.21	2.09	3.23	1.69	69.9	27.8
$\alpha = 3.0$	-24	-6	0.32	1.65	1.45	1.46	3.64	2.08	54.8	15.1
$\alpha = 4.0$	-31	-6	0.25	2.08	1.75	1.03	4.21	2.58	40.4	7.4
HANN POISSON $\alpha = 0.5$	-35	-18	0.43	1.61	1.54	1.26	3.33	2.14	61.3	12.6
$\alpha = 1.0$	-39	-18	0.38	1.73	1.64	1.11	3.50	2.30	56.0	9.2
$\alpha = 2.0$	NONE	-18	0.29	2.02	1.87	0.87	3.94	2.65	44.6	4.7
CAUCHY $\alpha = 3.0$	-31	-6	0.42	1.48	1.34	1.71	3.40	1.90	61.6	20.2
$\alpha = 4.0$	-35	-6	0.33	1.76	1.50	1.36	3.83	2.20	48.8	13.2
$\alpha = 5.0$	-30	-6	0.28	2.06	1.68	1.13	4.28	2.53	38.3	9.0
GAUSSIAN $\alpha = 2.5$	-42	-6	0.51	1.39	1.33	1.69	3.14	1.86	67.7	20.0
$\alpha = 3.0$	-55	-6	0.43	1.64	1.55	1.25	3.40	2.18	57.5	10.6
$\alpha = 3.5$	-69	-6	0.37	1.90	1.79	0.94	3.73	2.52	47.2	4.9
DOLPH-CHEBYSHEV $\alpha = 2.5$	-50	0	0.53	1.39	1.33	1.70	3.12	1.85	69.6	22.3
$\alpha = 3.0$	-60	0	0.48	1.51	1.44	1.44	3.23	2.01	64.7	16.3
$\alpha = 3.5$	-70	0	0.45	1.62	1.55	1.25	3.35	2.17	60.2	11.9
$\alpha = 4.0$	-80	0	0.42	1.73	1.65	1.10	3.48	2.31	55.9	8.7
KAISER-BESSEL $\alpha = 2.0$	-46	-6	0.49	1.50	1.43	1.46	3.20	1.99	65.7	16.9
$\alpha = 2.5$	-57	-6	0.44	1.65	1.57	1.20	3.38	2.20	59.5	11.2
$\alpha = 3.0$	-69	-6	0.40	1.80	1.71	1.02	3.56	2.39	53.9	7.4
$\alpha = 3.5$	-82	-6	0.37	1.93	1.83	0.89	3.74	2.57	48.8	4.8
BARCILON-TEMES $\alpha = 3.0$	-53	-6	0.47	1.56	1.49	1.34	3.27	2.07	63.0	14.2
$\alpha = 3.5$	-58	-6	0.43	1.67	1.59	1.18	3.40	2.23	58.6	10.4
$\alpha = 4.0$	-68	-6	0.41	1.77	1.69	1.05	3.52	2.36	54.4	7.6
EXACT BLACKMAN	-51	-6	0.46	1.57	1.52	1.33	3.29	2.13	62.7	14.0
BLACKMAN (0.42, 0.50, 0.08)	-58	-18	0.42	1.73	1.68	1.10	3.47	2.35	56.7	9.0
MINIMUM 3-SAMPLE BLACKMAN-HARRIS	-67	-6	0.42	1.71	1.66	1.13	3.45	1 81	57.2	9.6
* MINIMUM 4-SAMPLE BLACKMAN-HARRIS	-92	-6	0.36	2.00	1.90	0.83	3.85	2.72	46.0	3.8
*61 dB 3-SAMPLE BLACKMAN-HARRIS	-61	-6	0.45	1.61	1.56	1.27	3.34	2.19	61.0	12.6
74 dB 4-SAMPLE BLACKMAN-HARRIS	-74	-6	0.40	1.79	1.74	1.03	3.56	2.44	53.9	7.4
4-SAMPLE $\alpha = 3.0$ KAISER-BESSEL	-69	-6	0.40	1.80	1.74	1.02	3.56	2.44	53.9	7.4

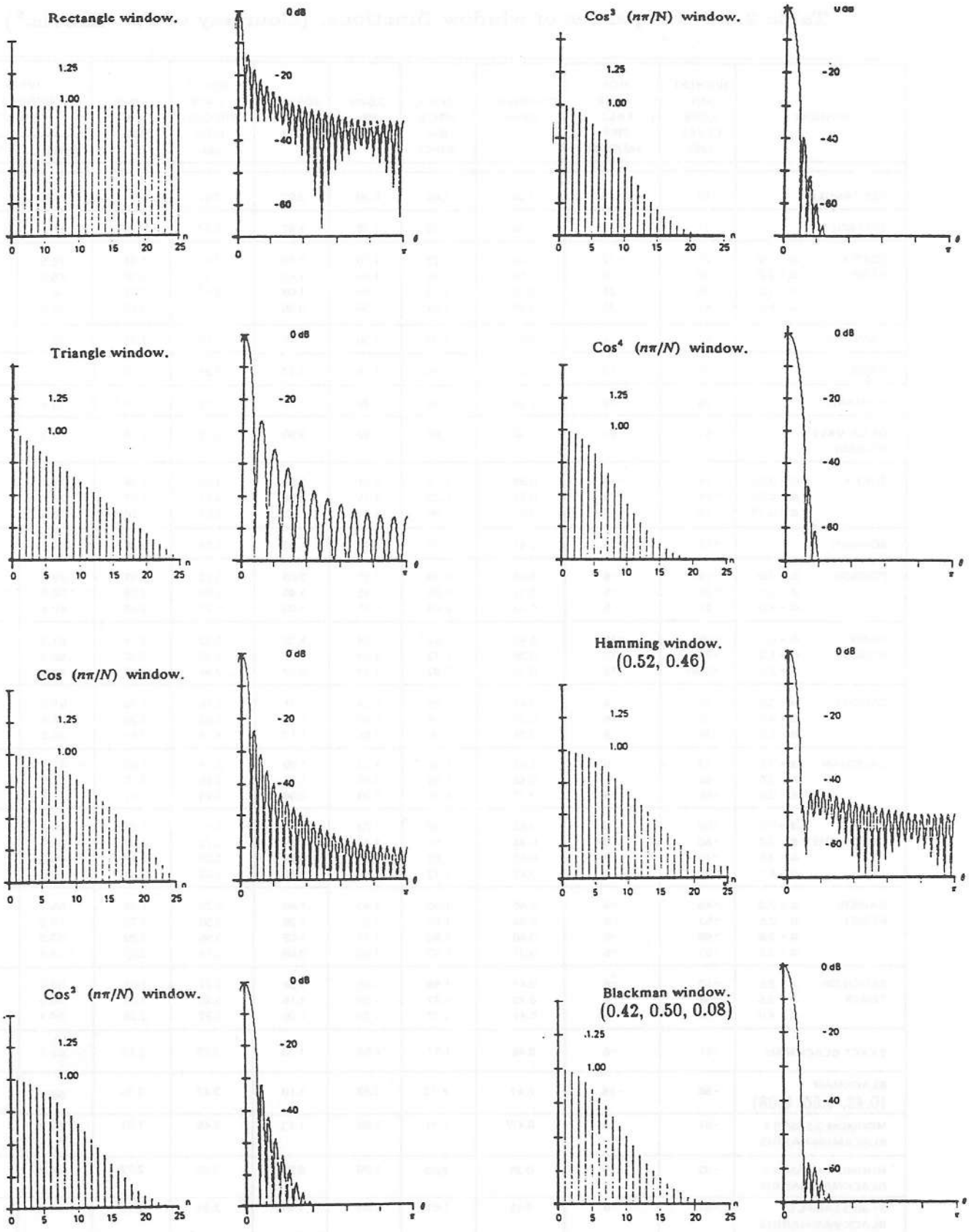


Fig. 2.6.1 Various time domain windows and their magnitude spectra. (Courtesy of F.J. Harris.[5])

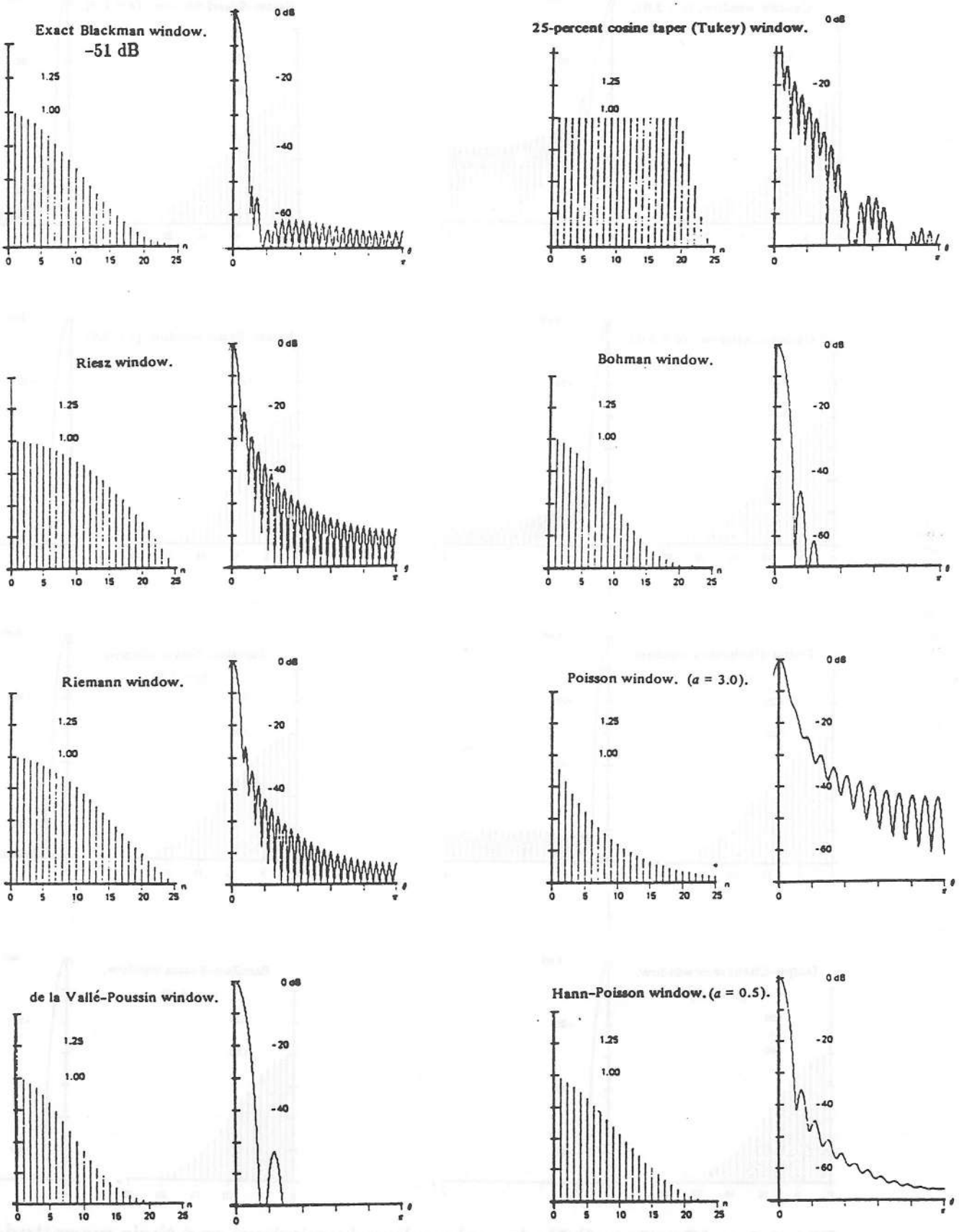


Fig. 2.6.1 (Continued) Various time domain windows and their magnitude spectra. (Courtesy of F.J. Harris.[5])

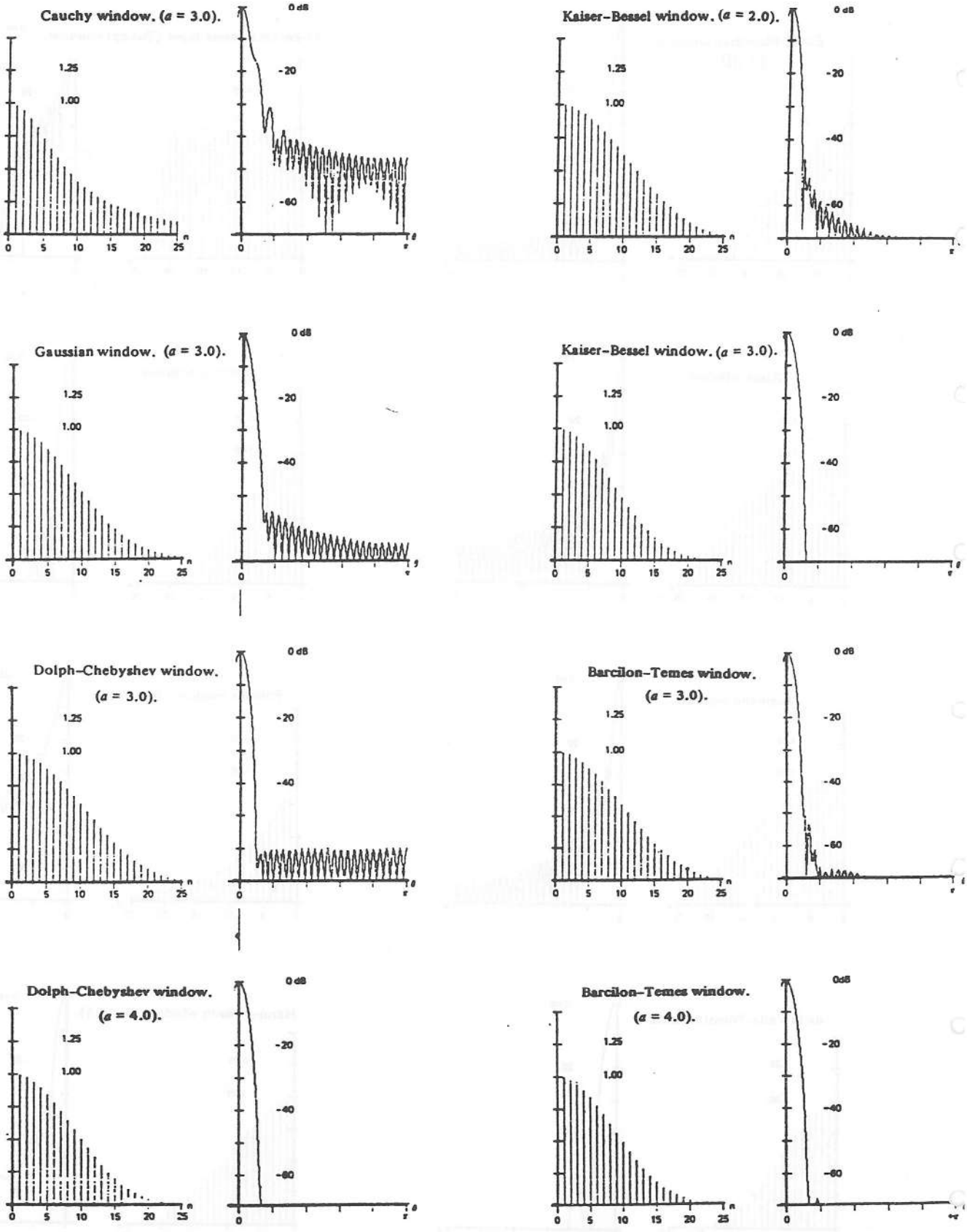


Fig. 2.6.1 (Continued) Various time domain windows and their magnitude spectra. (Courtesy of F.J. Harris.[5])

Per terminare il paragrafo, si presentano sinteticamente i problemi introdotti dal computo della FFT di un segnale e relativi rimedi.

A) Distorsione dovuta all' Aliasing:

a.1- Aumentare la sampling frequency $\omega_S = 1/T$.
a.2- Usare filtri passa basso con opportuna frequenza di taglio e roll-off, prima della digitizzazione.

B) Distorsione dovuta allo spectral Leakage:

b.1- Usare funzioni di windowing con lobi secondari bassi. Ridurre la stop band del segnale (è in pratica quello che si ottiene aumentando il roll-off del caso A-1 precedente).
b.2- Aumentare la lunghezza del time record NT (larghezza della finestra di acquisizione). Questo implica l' aumento della samplig rate 1/T e/o il numero dei punti N considerati.

C- Bassa risoluzione spettrale (Spectral Scalloping o Picket fence effect):

c.1- Aumentare il numero dei samples N mantenendo costante la sample rate 1/T. Come visto al punto b.2, questo aumenta il time record NT, cioè la larghezza della finestra.
c.2- Usare la tecnica del così detto zero padding. Questa consiste nello aggiungere "zeri" ad un determinato array di samples N. E' è equivalente all' aumento del time record visto in c.1, ma meno efficiente.

CONSIDERAZIONI RELATIVE ALL' INDIRIZZAMENTO.

Abbiamo analizzato in precedenza l' algoritmo di calcolo della FFT a decimazione di tempo, per il fatto che e`questo ad essere implementato sui chips DSP della SHARP. Questo algoritmo necessita in ingresso l' array di dati da trasformare gia` ordinati secondo un particolare ordine, detto ordine inverso (bit reversed order), e fornisce in uscita i dati in "ordine normale". L' algoritmo di calcolo della FFT a decimazione di frequenza, accetta in ingresso l' array di dati in ordine normale, ma fornisce poi i dati in uscita in ordine inverso per cui un processo di riordinamento sarebbe comunque richiesto. Questo implica che, siccome prima del computo della FFT l' array di dati risiede in un buffer di memoria in ordine normale, e` necessario affrontare il problema dell'indirizzamento corretto dei dati. Si puo` affrontare il problema operando in due modi. Il primo consiste nel calcolare direttamente gli indici "bit reversed" per indirizzare, poi, correttamente i punti da leggere nel buffer. La seconda possibilita` e`quella di memorizzare i punti in memoria gia` scambiati in ordine per poi leggerli linearmente. Il problema dell'indirizzamento suddetto, non e` un problema del solo primo stadio, ma di tutti gli stadi successivi della elaborazione.

CONSIDERAZIONI RELATIVE AI COEFFICIENTI.

Un altra serie di considerazioni deve essere fatta a riguardo dei coefficienti W_N^r. Le strade per produrli possono essere, anche in questo caso, due. Nella prima, si puo` optare per il calcolo degli stessi di volta in volta quando servono; soluzione questa che se da un lato risparmia memoria, dall' altro richiede piu` tempo per ovvi motivi. Nella seconda si puo` scegliere di calcolarli a priori e memorizzarli in un buffer di memoria; soluzione questa certamente piu` efficiente da un punto di vista della velocita` ma che richiede molta piu` memoria. Questo potrebbe rappresentare, in certe condizioni, un vero problema. Fortunatamente, questi coefficienti per come vengono definiti, presentano notevoli proprieta` di simmetria e periodicita` che possono essere efficacemente sfruttate per ridurre al minimo la richiesta di memoria. Infatti:

$$W_N = e^{-j2\pi/N} = \cos(2\pi / N) + j\,\mathrm{sen}(2\pi / N)$$

e

$$W_N^r = e^{-j2\pi r/N} = \cos(2\pi r / N) + j\,\mathrm{sen}(2\pi r / N)$$

Considerando i valori r=0,1,2,3,4....N, e` semplice constatare che i suddetti valori giacciono, equidistanti, sul cerchio unitario a distanza angolare di 360/N

gradi. Per esempio, scegliendo N=16, ci troviamo nella situazione di fig.6 osservando la quale, si possono fare alcune considerazioni. Dalla conoscenza dei soli coefficienti appartenenti, ad esempio, al quarto quadrante si possono ottenere quelli del primo quadrante cambiando il segno della sola parte immaginaria, quelli del secondo cambiando il segno della parte reale ed immaginaria e quelli del terzo invertendo solo il segno della parte reale. Come si evince dalla stessa fig.6, partendo per esempio dal coefficiente ad indice 3 si puo` ricavare quello di indice 13 cambiando il segno della parte immaginaria, quello di indice 11 cambiando entrambi i segni della parte reale ed immaginaria ed infine, quello di indice 5 cambiando il segno della sola parte reale. Questo significa che non e` necessario memorizzare tutti i 360 gradi di coefficienti ma solo, come nell' esempio, il quarto quadrante piu` il punto a 270 gradi ed operare sulle complementazioni, riducendo, grazie a tale periodicita`, ad un quarto la dimensione della memoria richiesta senza perdere in velocita. Inoltre, un sistema ottimizzato per il computo della FFT deve anche tenere conto del fatto che se i primi stadi della elaborazione sono blocchi radix-2 o radix-4, i relativi coefficienti sono tutti 1 per cui, invece di effettuare tali moltiplicazioni inutili, si puo` ottimizzare il sistema effettuando, se richiesto, il windowing dei dati.

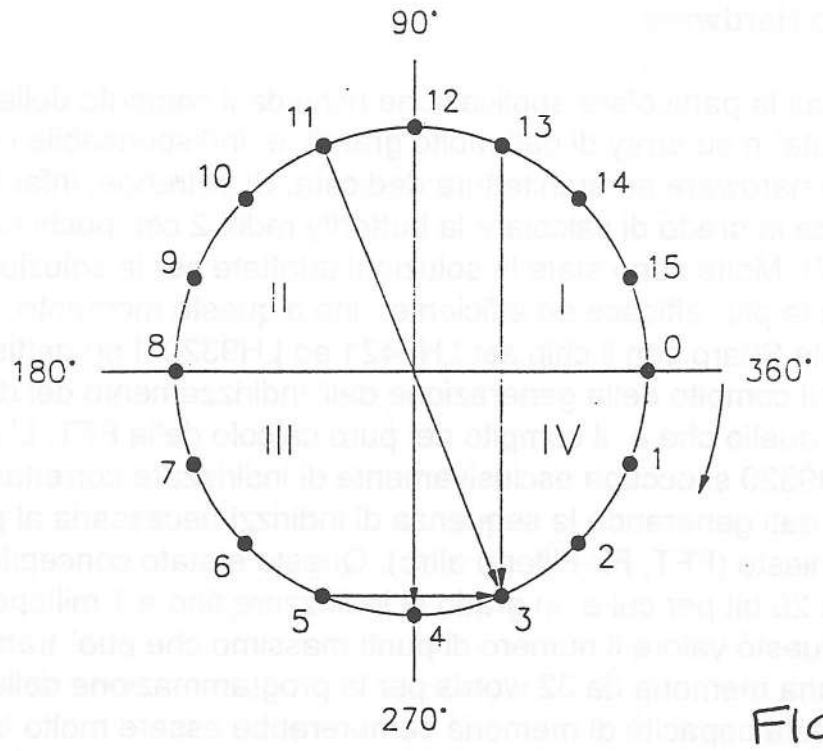


FIG.6

DIVERSI TIPI DI APPROCCIO AL PROBLEMA DEL COMPUTO DELLA FFT

1- Approccio Software:

L'approccio piu` immediato e comodo per la soluzione del problema del calcolo della FFT, e` quello software. Avendo a disposizione un calcolatore e`, infatti, relativamente semplice scrivere programmi ottimizzati ad alto livello per questo scopo. Il problema pero' nasce dalla bassa velocita` di esecuzione degli algoritmi per il fatto che la macchina non possiede una architettura specifica e, soprattutto, e` continuamente rallentata da altri processi che parallelamente impegnano la CPU, primo fra tutti il sistema operativo. Questa soluzione immediata e` quindi adatta solo nei casi in cui la velocita` di esecuzione non riveste una grande importanza come, per esempio, in processi off-line e con array di dati di piccole dimensioni. Lo step successivo potrebbe essere quello in cui la CPU usata per l' esecuzione dell' algoritmo, sia scaricata da tutti gli atri compiti aumentando, in tal modo, la velocita` di esecuzione ma non al punto tale da giustificarne un incremento del costo e complessita` del sistema. Il vero impulso, in questo senso, lo si e` avuto con l' introduzione nel mercato dei Digital Signal Processor (DSP) che, pur essendo simili a normali microprocessori, sono caratterizzati da una architettura orientata a certi tipi di calcoli od algoritmi e per il fatto che hanno a bordo moltiplicatori hardware (in grado di effettuare moltiplicazioni in un solo ciclo di clock), che ne aumentano enormemente le prestazioni.

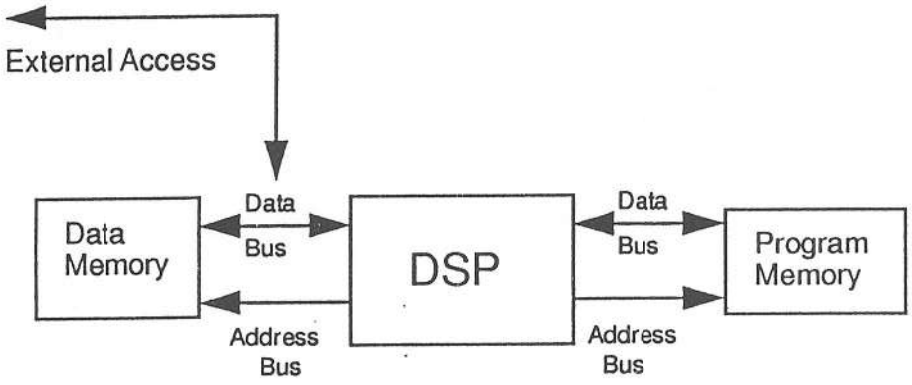
2- Approccio Hardware

Nel caso in cui la particolare applicazione richieda il computo della FFT ad elevate velocita` e su array di dati molto grandi, e` indispensabile orientarsi verso sistemi hardware ad architettura dedicata. Si potrebbe, infatti, realizzare una rete logica in grado di calcolare la butterfly radix-2 con pochi cicli di clock secondo la (7). Molte sono state le soluzioni adottate per la soluzione di questo problema ma la piu` efficace ed efficiente, fino a questo momento, e` quella introdotta dalla Sharp con il chip set LH9421 ed LH9320. I progettisti Sharp hanno diviso il compito della generazione dell' indirizzamento dei dati in memoria, da quello che e` il compito del puro calcolo della FFT. L' address generator LH9320 si occupa esclusivamente di indirizzare correttamente la memoria dei dati generando la sequenza di indirizzi necessaria al particolare algoritmo richiesto (FFT, Fir-Filter o altro). Questo e`stato concepito per lavorare con un bus a 20 bit per cui e` in grado di indirizzare fino a 1 milione di punti, limitando a questo valore il numero di punti massimo che puo` trattare. Ha al suo interno una memoria da 32 words per la programmazione delle sequenze richieste; questa capacità di memoria sembrerebbe essere molto bassa, in realtà è più che sufficiente perche`, per esempio, una FFT di 1 milione di punti richiede un programmino di soli 5 passi.

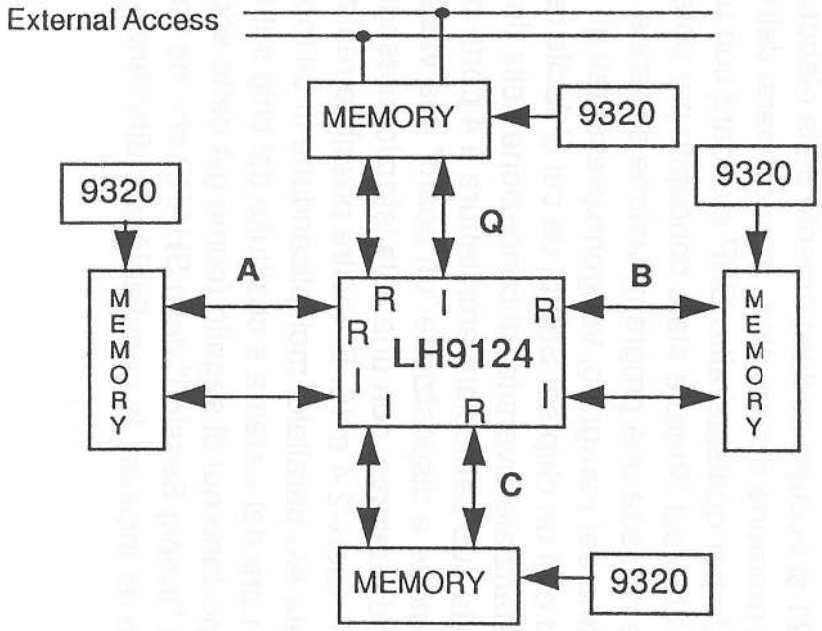
Il DSP LH9421 si occupa esclusivamente della elaborazione dei dati che arrivano dalla memoria senza, cioe`, preoccuparsi dell'indirizzamento come invece succede per qualsiasi altro DSP standard con un ovvio rallentamento del processing. Il chip set Sharp e`stato concepito per potere lavorare in perfetto sincronismo e richiede una programmazione separata. Nella prima tavola riportata alla fine del paragrafo, vengono presentati gli schemi a blocchi di un DSP standard e di un chip set Sharp, da cui e` facile rendersi conto della effettiva e sostanziale diversita` di concezione delle due architteture DSP. Lo Sharp, oltre alla sua particolare architettura a 4 porte programmabili di I/O da 48 bit ciascuna, mette a disposizione "on board" una vasta gamma di macro per cui e` possibile, ad esempio, con una sola istruzione eseguire un intero passo dell' algoritmo FFT, radix-2,4 e 16. E` inoltre possibile realizzare sistemi con piu` DSP in cascata e/o parallelo, moltiplicandone in tal modo le gia` alte prestazioni.
Visto che il cuore del sistema è costituito dal chip set SHARP LH9124 ed LH9320, si riportano qui di seguito (come già detto sopra) una serie di tavole prese da una "trainig Session" della SHARP che ne sottolineano, in maniera molto concisa ed incisiva, le peculiarità e le differenze dagli altri DSP "standard".

DSP Architectures

CLASSICAL DSP SYSTEM



SHARP DSP SYSTEM



SHARP

Intro: Historical -vs- Sharp's Approach to DSP

Classical Approach:

- Single-chip
- Software intensive; DSP algorithms are implemented in a high level language (C, Pascal, Fortran, etc.) compilers necessary
- Micro-processor oriented (op-code driven)
- Built around convolution in the time-domain
- Precision: typically an average of 16 bits data, 32 bit internal precision

Sharp's Approach:

- Chip-set
- Easy to use, DSP algorithms are coded into LH9124; minimal external software needed to implement algorithms
- DSP array oriented (algorithm specific)
- Built around the Fast Fourier Transform in the frequency domain (resolution is free)
- Precision: fixed-point, with block floating point capability, 24 bits data, 60 bits internal; unrivaled!

SHARP

DSP Competitor Analysis

Competitor	Performance of 1K Complex FF	Precision Int/Ext	Functionality
Plessey PDS216510	*96 µsec	16	FFT only
**TRW TMC2310	514 µsec	19/16	FFT, FIR
Array ADSP66110/20	128 µsec	20/16	FFT, FIR, I^2Q^2
SHARP LH9124	80 µsec	60/24	FFT,FIR,I^2 Q^2 ,FCT, DCT, arithmetic and logic, plus 155 address patterns

SHARP

DSP Competitors: PLESSEY

PDSP16510

- Single chip device optimized for FFT function only
- ☆Sharp's LH9124 implements all major standard high-level DSP algorithms
- Very limited array size = 1K
- ☆Sharp's array size = 1M
  - Impact: curtails range of applications for device
- Less performance = 16 bits: 1K Complex FFT in 147 µs
- ☆Sharp's performance = 24 bits: 1K Complex FFT in 87µs
- Cost estimate = $800 - $1100

SHARP

DSP Competitor: Array Microsystems

HDSP6610/66210

- Limited functionality
  - Instruction set not as complete as LH9124
- Limited array size = 64K

☆Sharp's array size = 1M

In an image processing application, to perform a 1Kx1K 2D image, you would need only one LH9124 but two HDSP6610's

- Awkward bus structure = 3 busses

☆Sharp's bus structure = 4 busses

- Less performance = 16 bits: 1K Complex FFT in 122 µs

☆Sharp's performance = 24 bits: 1K Complex FFT in 87 µs

- Cost estimate = $600 - $700

SHARP

More DSP Competition

Analog Devices Part # ADSP21020

- Functionality - general purpose device
- Performance - 1K Complex FFT in 0.8 ms
- Cost = ~$220/unit at 0.1% performance

Motorola Part # DSP96002

- Functionality - 32 bit floating point, general purpose
- Performance - 1K Complex FFT in 1.0 ms
- Cost = $441

Texas Instruments Part #TMS320C40

- Functionality - 32 bit floating point, general purpose
- Performance - 1K Complex FFT in 2.0 ms
- Cost = ~$400 - $500 at 0.1% performance

SHARP

DSP Benchmarks - 1K Complex FFT

Part Number	Performance	Data-Width	Scale:
80386 (20MHz)	200 ms	16-bit Fixed point	=1
VAX 11/780	150 ms	16-bit Fixed Point	1.3x
TMS320C25	15.8 ms	16-bit Fixed Point	12.6x
Moto 56001	5 ms	24-bit Fixed Point	40x
TMS320C40	2 ms	32-bit Floating Point	100x
i860(40MHz)	1.5 ms	64-bit Floating Point	133x
Cray X-MP	1 ms	64-bit Floating Point	200x
Moto 96002	1 ms	32-bit Floating Point	200x
ADSP21020	0.8 ms	40-bit Floating Point	250x
Sharp LH9124 (40MHz)	**81 µs**	24-bit Fixed Point	**2500x**

SHARP

DSP Benchmarks - 1K Complex FFT (cont.)

Part Number	Performance	Data-Width	Scale:
80386	200 ms	16-bit Fixed Point	= 1
TRWTM2310	514 µs	16-bit Fixed Point	390x
PDSP16510	147 µs	16-bit Fixed point	1360x
ADSP6610	122 µs	16-bit Fixed point	1639x
Sharp LH9124 (40MHz)	**81 µs**	24-bit Fixed Point	**2500x**

SHARP

Customer Applications for Sharp's DSP Chip-set

Signal Processing

- Speech processing
- Radar and sonar
- Seismic processing

Telecommunications

- High frequency
- Frequency multiplexing
- Echo cancelling
- Phase filters

Instrumentation

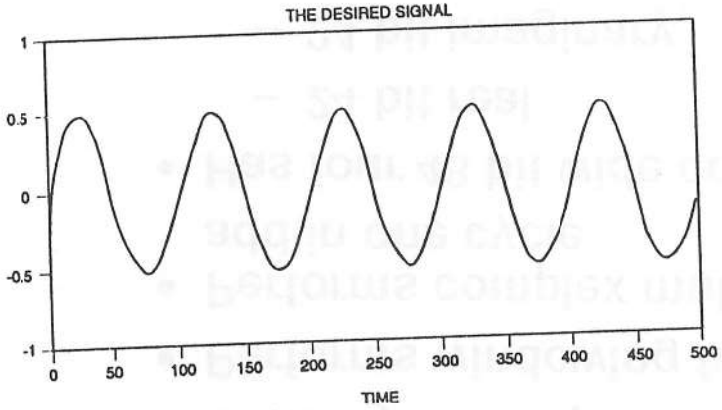
- Spectrum analysis
- Test equipment

Image Processing

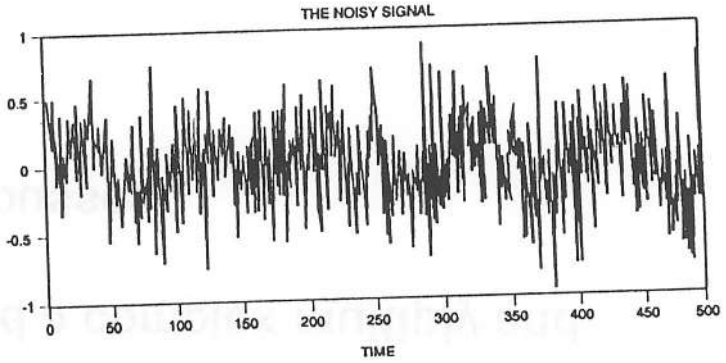
- Medical Imaging
- Pattern recognition
- Image enhancement
- Image Correlation
- Image compression
- JPEG (DCT)

SHARP

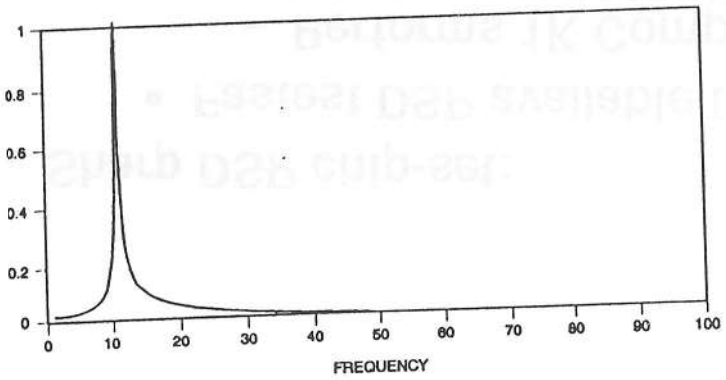
Applications: Sonar System Detection



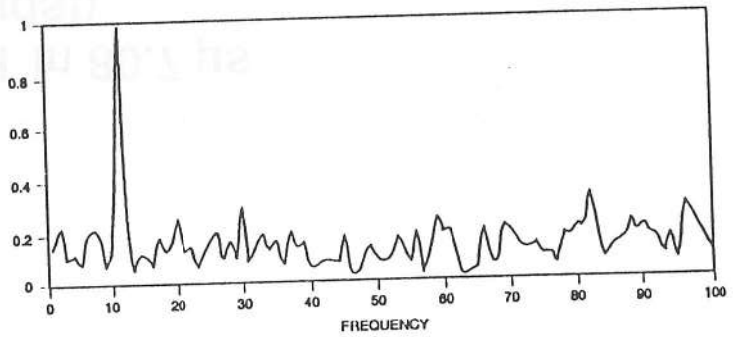
- The desired signal is periodic signal machinery



- Received signal is noisy due to noise of sea



- Detection algorithm to perform Fourier Transform



- Results in clear distinction between noise and desired signal

SHARP

Summary of Unique Highlights

Sharp DSP chip-set:

- Fastest DSP available today
  - Performs 1K Complext FFT in 80.7 µs (80.7 µs = 0.0000807 seconds!)
- Only DSP available today that performs Radix-16 butterfly in one pass
- Performs windowing function on first pass of FFT
- Performs complex multiply and a complex multiply and add in one cycle
- Has four 48 bit wide complex busses
  - 24 bit real
  - 24 bit imaginary

SHARP

Summary of Unique Highlights (cont)

Sharp DSP chip-set

- Uses block floating point approach
  - Takes advantage of floating point and fixed point data representation
- Minimizes software implmentation of high-level DSP algorithms

  EXAMPLE:

  To implement 1K Complex FFT:
  - Motorola general purpose chip = 140 lines of asembly code and 1ms
  - Sharp DSP = 3 lines of code and 80.7 µs
- Easy to use, flexible and streamlines system design (through 4-port architecture)

SHARP

(file:shar1.doc)

CAP. 2

IL DSP SHARP LH9124.

Il DSP della SHARP LH9124, è un elaboratore digitale di segnali ad altissime prestazioni che non si presenta con una architettura simile a quella dei più comuni DSP, ma piuttosto come una unità di calcolo orientata ed ottimizzata per l' esecuzione di un certo numero di calcoli e/o algoritmi. Infatti, come già visto in precedenza, questa unità non è assolutamente in grado di generare gli indirizzi come tutti gli altri DSP standard, ma necessita di un address generator: l' LH9320. Anche se è difficile classificare tale dispositivo, lo possiamo pensare come un "processore vettoriale" in quanto ogni istruzione ha come operando l' intera memoria o l' intero vettore contenente i dati da elaborare. Ogni singolo passo in cui l' elaborazione viene scomposta, viene eseguito da una singola istruzione che opera così o sull' intero array di dati che si presenta alla porta di ingresso, o sui risultati intermedi. Questo modo di operare trova particolare vantaggio nel LH9124 per la caratteristica presenza di ben 4 porte di I/O bidirezionali a 48 bit (24 reali e 24 complessi) programmabili (Fig.7). La porta che permette l' input dei dati dal mondo esterno (buffer di memoria di ingresso collegato al convertitore A/D), è la QR/QI. Attraverso la stessa porta, opportunamente programmata, escono i risultati della elaborazione verso lo stesso mondo esterno. Le due porte AR/AI e BR/BI sono dedicate all' handling dei risultati intermedi, mentre quella CA/CI all' ingresso dei "coefficienti" nel caso in cui l' operazione in corso abbia due operandi (ad es. i coefficienti W_N^k in butterfly di una FFT). Il flusso dei dati, quindi, è caratterizzato dall' ingresso al chip dalla porta Q, scambio dei risultati parziali tra le porte A e B per tutta la durata dell' elaborazione, eventuale acquisizione dei coefficienti dalla C e uscita dei dati ancora attraverso la Q. Attraverso la porta A o B è poi possibile prelevare dati, con una opportuna circuitistica, attraverso il bus del sistema ospitante che il più delle volte è un sistema VME. Una tale configurazione a quattro porte di I/O è quindi molto flessibile e, soprattutto, riduce al minimo la circuiteria esterna per il fatto che la gestione del flusso e direzione dei dati tra una porta e l' altra, è svolta internamente al chip. Come già accennato, ogni istruzione che il DSP esegue, è in pratica un passo dell' algoritmo in "processo" in quel momento. Ad ogni istruzione da eseguire (cioè operazione da effettuare sui dati) corrisponde un codice che viene introdotto dall' esterno attraverso il bus Function Code (FC[4:0]), mentre il percorso e la direzione del flusso dei dati tra una porta e l' altra, dal bus Data Flow (DF[2:0]). Il processore non dispone di memoria interna per il programma come i normali DSP e non dispone neppure dei classici registri che caratterizzano tutte le architetture dei microprocessori e DSP in generale (come ad esempio il program counter) per

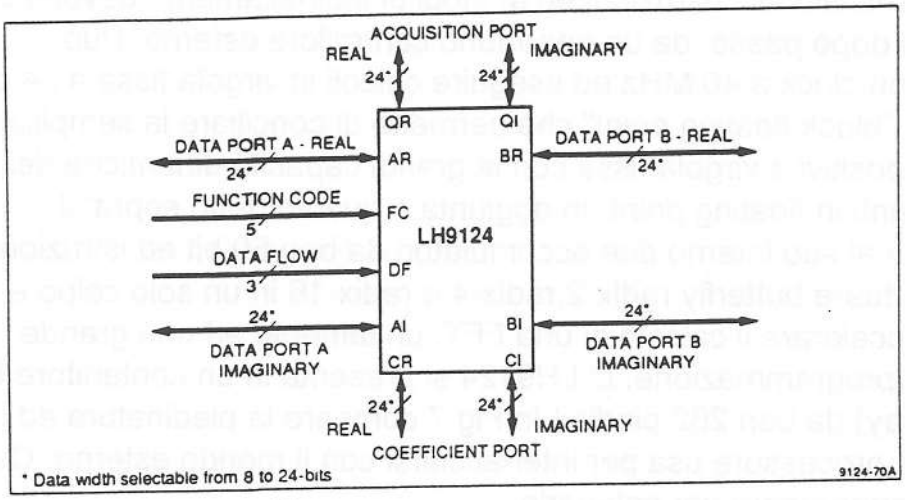


Figure 7 LH9124 Four Port Data Flow Structure

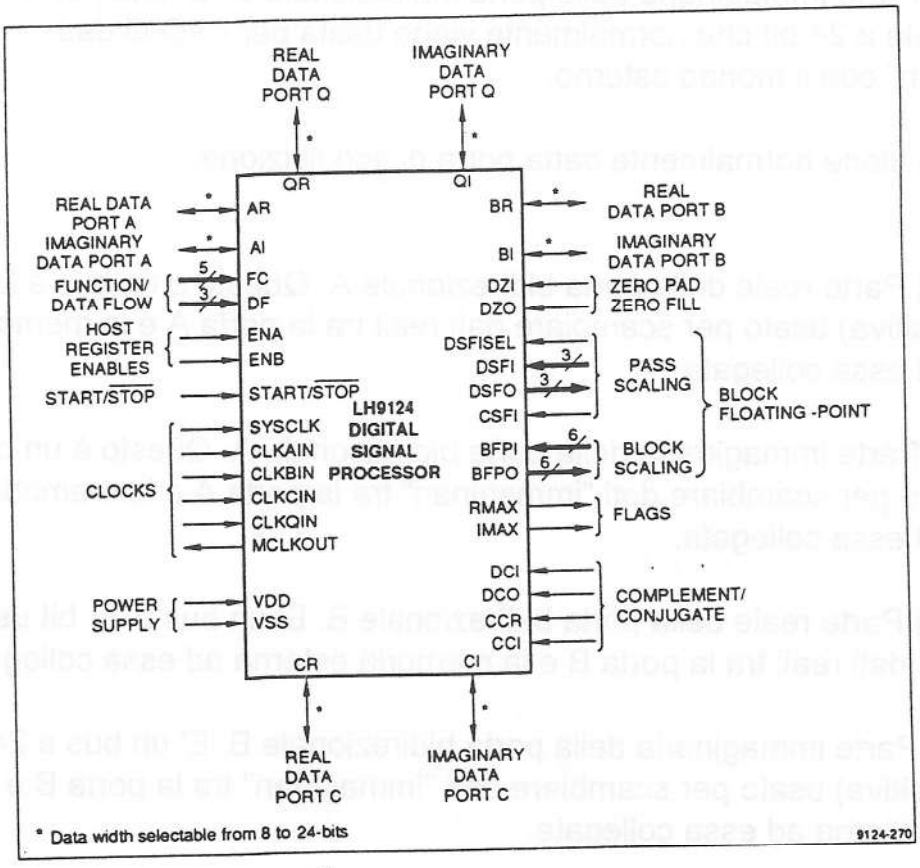


Figure 7a LH9124 Signal-Pin Descriptions

cui il codice relativo alle istruzioni ed ai modi di indirizzamento devono essere forniti, passo dopo passo, da un opportuno controllore esterno. Può funzionare con clock a 40 MHz ed eseguire calcoli in virgola fissa a 24 bit con l' estensione "block floating point" che permette di conciliare la semplicità tipica dei dispositivi a virgola fissa con le grandi capacità dinamiche dei sistemi operanti in floating point. In aggiunta a quanto detto sopra, il dispositivo ha al suo interno due accumulatori da ben 60 bit ed istruzioni in grado di effettuare butterfly radix-2,radix-4 e radix-16 in un solo colpo e quindi in grado di accelerare il calcolo di una FFT, unitamente ad una grande semplicità di programmazione. L' LH9124 si presenta in un contenitore PGA (Pin Grid Array) da ben 262 piedini! In Fig 7 compare la piedinatura ed i segnali che il processore usa per interfacciarsi con il mondo esterno. Questi si possono raggruppare per categorie.

A) BUS DATI

- **QR[23:0]** Parte reale della porta bidirezionale Q. E' una porta bidirezionale a 24 bit che normalmente viene usata per l' I/O di dati reali con il mondo esterno.

- **QI[23:0]** Parte immaginaria della porta bidirezionale Q. E' una porta bidirezionale a 24 bit che normalmente viene usata per l' I/O di dati "immaginari" con il mondo esterno.

La porta Q viene normalmente detta porta di acquisizione.

- **AR[23:0]** Parte reale della porta bidirezionale A. Questo è un bus a 24 bit (logica positiva) usato per scambiare dati reali tra la porta A e la memoria esterna ad essa collegata.

- **AI[23:0]** Parte immaginaria della porta bidirezionale A. Questo è un bus a 24 bit usato per scambiare dati "immaginari" tra la porta A e la memoria esterna ad essa collegata.

- **BR[23:0]** Parte reale della porta bidirezionale B. E' un bus a 24 bit usato per scambiare dati reali tra la porta B e la memoria esterna ad essa collegata.

- **BI[23:0]** Parte immaginaria della porta bidirezionale B. E' un bus a 24 bit (logica positiva) usato per scambiare dati "immaginari" tra la porta B e la memoria esterna ad essa collegata.

Le porte A e B sono anche dette porte dati

- **CR[23:0]** Parte reale della porta bidirezionale C. Questo è un bus a 24 bit usato per lo scambio dei coefficienti (nel senso generale di "secondo operando") tra la porta C e la memoria esterna.

- **CI[23:0]** Parte immaginaria della porta bidirezionale C. Questo è un bus a 24 bit usato per lo scambio dei coefficienti "immaginari" tra la porta C e la memoria esterna.

La porta C viene detta porta dei coefficienti.

B) SEGNALI DI CONTROLLO

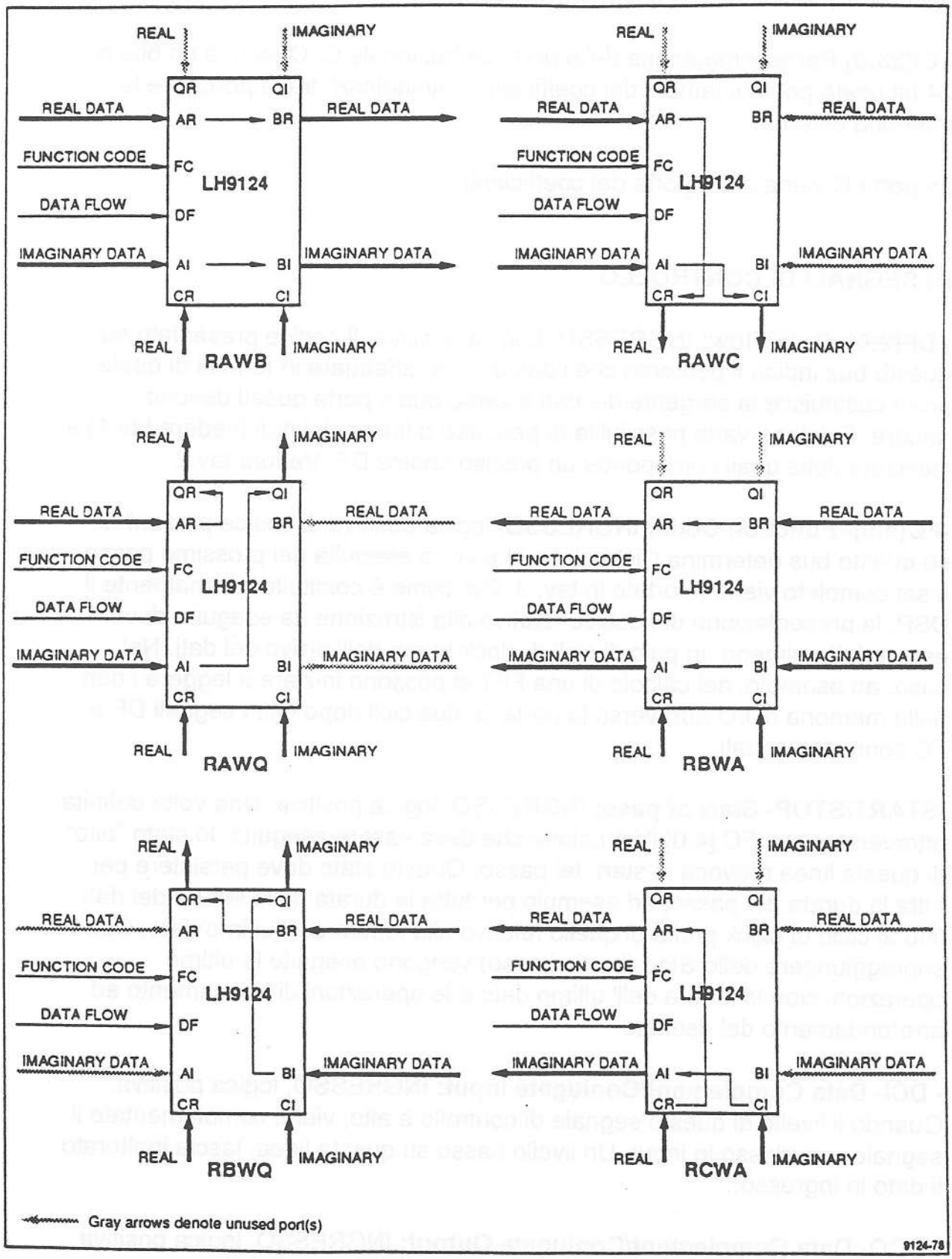
- **DF[2:0]- Data Flow:** INGRESSO, Logica positiva. Il codice presentato su questo bus indica il percorso che i dati devono effettuare in termini di quale porta costituisce la sorgente dei dati e verso quale porta questi devono andare. Esistono varie possibilità di percorso o flusso dei dati (vedere tav.1) a ciascuna delle quali corrisponde un preciso codice DF. Vedere tav.2

-**FC[4:0]- Function Code:** INGRESSO, logica positiva. Il codice presentato su questo bus determina l' istruzione che verrà eseguita nel prossimo passo. Il set completo viene riportato in tav. 3. Per come è costituito internamente il DSP, la presentazione del codice relativo alla istruzione da eseguire deve essere fatta almeno un paio di cicli di clock prima dell' arrivo dei dati. Nel caso, ad esempio, del calcolo di una FFT si possono iniziare a leggere i dati nella memoria di I/O attraverso la porta Q, due cicli dopo che i segnali DF e FC sono stati settati.

-**START/STOP- Start of pass:** INGRESSO, logica positiva. Una volta definita attraverso il bus FC [4:0] l' istruzione che deve essere eseguita, lo stato "alto" di questa linea provoca lo start del passo. Questo stato deve persistere per tutta la durata del passo, ad esempio per tutta la durata della lettura dei dati fino al ciclo di clock prima di quello relativo alla lettura dell' ultimo dato. Con il sopraggiungere dello Stop (livello basso) vengono eseguite le ultime operazioni cioè la lettura dell' ultimo dato e le operazioni di troncamento ed arrotondamento del risultato.

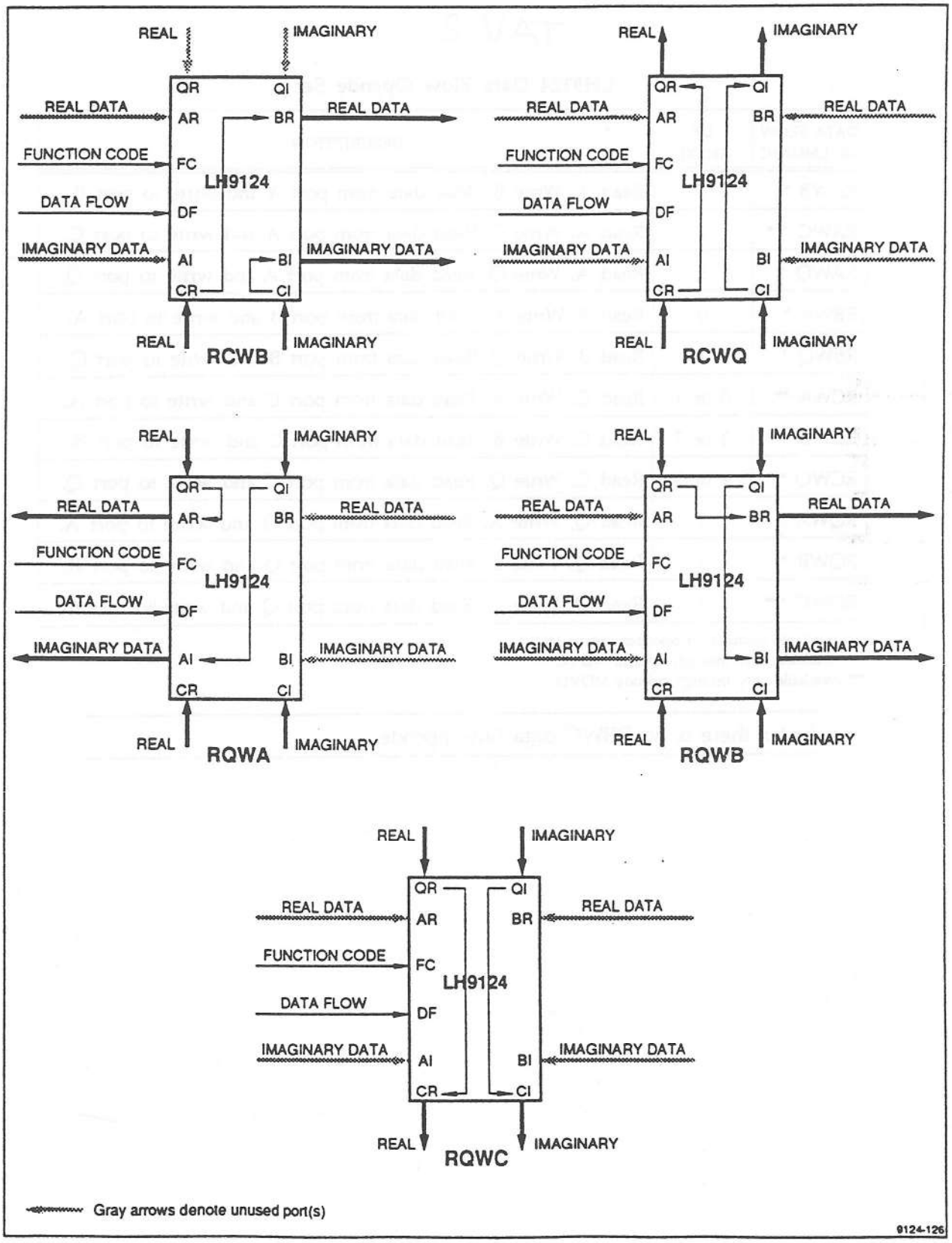
- **DCI- Data Complement/Coniugate Input:** INGRESSO, logica positiva. Quando il livello di questo segnale di controllo è alto, viene complementato il segnale complesso in input. Un livello basso su questa linea, lascia inalterato il dato in ingresso.

- **DCO- Data Complement/Coniugate Output:** INGRESSO, logica positiva. Un livello alto in questa linea, porta a complementare i dati di uscita mentre un livello basso, lascia inalterati gli stessi.



LH9124 Data Flow Opcodes Diagram

TAV. 1


REAL
IMAGINARY
QR
QI
REAL DATA
AR
BR
REAL DATA
FUNCTION CODE
FC
LH9124
DATA FLOW
DF
IMAGINARY DATA
AI
BI
IMAGINARY DATA
CR
CI
REAL
IMAGINARY
RCWB
RCWQ
RQWA
RQWB
RQWC
Gray arrows denote unused port(s)
9124-126

TAV. 2

LH9124 Data Flow Opcode Set

DATA FLOW MNEMONIC	DF (HEX)	DESCRIPTION
RAWB *	7	Read A, Write B. Read data from port A and write to port B.
RAWC ***	6	Read A, Write C. Read data from port A and write to port C.
RAWQ *	5	Read A, Write Q. Read data from port A and write to port Q.
RBWA *	0	Read B, Write A. Read data from port B and write to port A.
RBWQ *	4	Read B, Write Q. Read data from port B and write to port Q.
RCWA **	0 or 1	Read C, Write A. Read data from port C and write to port A.
RCWB **	3 or 7	Read C, Write B. Read data from port C and write to port B.
RCWQ **	4 or 5	Read C, Write Q. Read data from port C and write to port Q.
RQWA *	1	Read Q, Write A. Read data from port Q and write to port A.
RQWB *	3	Read Q, Write B. Read data from port Q and write to port B.
RQWC ***	2	Read Q, Write C. Read data from port Q and write to port C.

* Available through all opcodes except MOVC.
** Available only through opcode MOVC.
*** Available only through opcode MOVD.

Note: there is no RBWC data flow opcode.

- CCR- Coefficient Complement Real: INGRESSO, logica positiva. Il livello di questa linea, controlla il segno della parte reale dei coefficienti lettiattraverso la porta C. CCR=1 significa che il segno viene cambiato. Per contro CCR=0 lascia il segno del coefficiente letto, inalterato.

- CCR- Coefficient Complementary Immaginary: INGRESSO, logica positiva. Questa linea di controllo si comporta come la CCR precedente, ma esclusivamente per la parte immaginaria dei coefficienti.

Le linee CCR e CCI sono molto utili per ridurre le dimensioni della memoria che contiene i coefficienti, come già visto in precedenza.

- DZI- Zero Input Data: INGRESSO, logica positiva. Un livello alto in questa linea di controllo, porta a sostituire il set di dati in igresso con tutti 0. Un livello 0 permette la lettura del set dei dati in maniera normale.

- DZO- Zero Output Data: INGRESSO, logica positiva. Un livello alto su questa linea, forza tutti i dati in uscita a zero. Il livello zero permette ai dati di raggiungere la porta di destinazione con segno inalterato.

- ENA- Enable A: INGRESSO, logica positiva. Quando il livello di questa linea è alto, viene abilitata la lettura (al ciclo di clock successivo) dei bus FC[4:0] e DF[2:0].

- ENB- Enable B: INGRESSO, logica positiva. Un livello alto in questa linea, abilita (al ciclo successivo di clock) la lettura dei livelli dei segnali DCI, DCO, DSFI, DSFI, DSFISEL, BFPI[5:0].

Per questi due ultimi segnali devono essere soddisfatte alcune condizioni: il segnale ENA deve essere settato prima di ENB ed almeno due cicli di clock prima della attivazione del segnale Start/Stop.

C) GESTIONE DEL BLOCK FLOATING-POINT E FATTORE DI SCALA.

- DSFISEL- Data Scaling Factor Input Select: INGRESSO. Le modalità con cui vengono trattati i dati all' interno del DSP, sono determinate dallo stato di questo segnale. Con DSFISEL=1 si abilita il trattamento "non automatico", nel senso che i dati vengono scalati verso sinistra di tanti posti quanti ne sono indicati nel bus DSFI (data scaling factor). Con il livello DSFISEL=0 si abilita il trattamento automatico dei dati nel formato Block Floating Point. Come esempio, se si vuole abilitare lo scaling dei dati in ingresso in modo non

automatico di 2 posti a sinistra, si deve impostare DSFISEL=1 e DSFI[2:0]=010.

- BFPI[5:0]- Block Floating point Input: INGRESSO, logica positiva. Quando è attivata l' aritmetica in Block Floating Point (DSFISEL=0), attraverso questo bus è possibile leggere il fattore di scala accumulato nel passo precedente, per poterne tenere conto nel passo che segue per evitare l' overflow.

- BFPO[5:0]- Block Floating Point Output: USCITA, logica positiva. Quando è attivata l' aritmetica in Block Floating Point (DSFISEL=0), il dato letto nel bus BFPO[5:0] alla fine del passo, rappresenta il numero di bit di cui è stato scalato il dato in ingresso fino a quel momento o, più precisamente, la somma di tutti i fattori di scala relativi ad ogni passo precedente a quello corrente compreso quello impostato sul bus DSFI[2:0]. Nel caso di un sistema a singolo processore, il bus BFPI[5:0] deve essere collegato con il bus BFPO[5:0], nel caso di un sistema a multiprocessore (più LH9124 collegati in cascata) si devono collegare le uscite BFPO del primo DSP agli ingressi BFPI del DSP successivo.

- DSFI[2:0]- Data Scaling Data Input: INGRESSO, logica positiva. Su questo bus viene impostato il fattore di scala relativo al passo in corso. Se si pone DSFISEL=1 (scaling manuale) il fattore di scala deve essere impostato manualmente. Se si pone DSFISEL=0 lo scaling avviene automaticamente, collegando il bus DSFI[2:0] a quello DSFO[2:0].

- DSFO[2:0]- Data Scaling Factor Output: USCITA, logica positiva. Su questo bus viene fornito il fattore di scala da applicare al passo successivo. Questo viene valutato sulla base di una stima in eccesso dei risultati dei passi precedenti.

- CSFI- Coefficient Scaling Factor Input: INGRESSO, logica positiva. Quando CSFI è alto, il coefficiente letto viene scalato di un bit verso a destra. Con CSFI=0, i coefficienti letti rimangono inalterati.

D) SEGNALI DI TEMPORIZZAZIONE.

- SYSCLK- System clock: INGRESSO. Ingresso del segnale di clock per l' intero DSP LH9124.

- CLKQIN- Clock Input for PORT Q: INGRESSO. Questo ingresso gestisce un registro interno per la memorizzazione dei dati presenti sulla porta Q. Lo stesso registro viene letto al ciclo successivo di SYSCLK per dare poi il via alla elaborazione. Questo segnale di clock è separato dal segnale SYSCLK.

- CLKAIN- Clock Input for PORT A: INGRESSO. Simile al CLKQIN ma agisce sulla porta A.

- CLKBIN- Clock Input for PORT B: INGRESSO. Simile al CLKQIN ma agisce sulla porta B.

- CLKCIN- Clock Input for PORT C: INGRESSO. Simile al CLKQIN ma agisce sulla porta C.

- MCLKOUT- Machine-Cycle Clock Output: USCITA. Da questa uscota è possibile prelevare il segnale SYSCLK diviso per 4.

E) SEGNALI DI STATO.

- RMAX- Real Maximum: USCITA, logica positiva. Questo segnale viene generato dalla comparazione dei sette bit più significativi del dato corrente con gli omologhi sette bit del dato immediatamente precedente. RMAX diventa alto quando la parte reale del dato corrente risulta essere maggiore di tutti i precedenti.

- IMAX- Immaginary Maximum: USCITA, logica positiva. Stessa funzione di RMAX, ma riferita alla parte immaginaria.

TEMPORIZZAZIONI

In questo paragrafo vengono analizzate le temporizzazioni dei vari segnali di controllo e di interfacciamento dei dati da elaborare e relativi risultati. Il primo diagramma di base per il timing del sistema, viene riportato in fig 8.
Vista la complessità delle varie operazioni, per semplicità si considera di fare eseguire una sola istruzione (un solo passo) su 4 valori e 4 coefficienti letti rispettivamente dalla porta Q e C, mettendo poi il risultato nella porta A. Il diagramma di timing per questo tipo di passo e' riportato in fig.9. Al ciclo T_0, il livello 1 del segnale ENA abilita (per il fronte di salita del clock successivo T_1) i registri di I/O per l' impostazione del settaggio in FC[4:0] e DF[2:0]. Nello stesso modo, un livello alto del segnale ENB (al fronte di salita del clock T_3) abilita i registri dei segnali di controllo DCI/DCO e dei segnali di gestione della Block Floating point e fattori di scaling DSFI[2:0], DSFISEL, CSFI e BFPI[5:0]. Come gia' detto in precedenza, ci devono essere 2 cicli di ritardo tra i setting di ENA e ENB. Se nel ciclo di clock T_2 si rende attivo il segnale di START/STOP, l' elaborazione parte al fronte di salita dell' impulso di clock successivo T_3. Nello stesso tempo devono essere resi disponibili i dati ed elaborati a seconda del livello delle linee DZI (Zero Input Data) e dei segnali di complemetazione della parte reale ed immaginaria dei coefficienti (letti dalla porta C) CCR e CCI. A questo punto i dati vengono memorizzati nei latches di ingresso al sopraggiungere del successivo clock delle porte interessate CLKQIN e CLKCIN,che seguono con leggero ritardo il clock di sistema SYSCLK, mentre DZI, CCR e CCI vengono abilitati dallo stesso START/STOP (che controlla il doppio accumulatore da 60 bit) e, successivamente, letti al ciclo di clock dopo. I dati, quindi, vengono letti con il fronte di salita del clock della "porta x" CLKxIN e passano al DSP per la elaborazione solo al ciclo seguente del clock di sistema T_4; durante lo stesso T_4 viene letto il secondo set di dati che passa al DSP, per l' elaborazione, al ciclo dopo di SYSCLK (T_5). Durante il ciclo T_5 viene letto con le stesse modalita' viste prima, la penultima serie di dati elaborata il ciclo di clock dopo. A questo punto, si deve disabilitare il segnale di START/STOP (livello zero) per potere essere rilevato dal clock successivo e dare cosi' il via alla lettura del set di dati. A lettura avvenuta, prima di avere in uscita il risultato bisogna attendere un certo numero di cicli di clock, cioe' la **latenza** L relativa al passo in corso, dovuta alla pipeline interna ed al managing dei dati, che e` calcolata a partire dalla lettura del primo dato. Ovviamente la latenza dipende dal tipo e complessita' del passo in corso, come si puo` riscontrare dalla tab 4. Nel nostro caso, quindi, il primo dato e' stato letto al clock T_3., di conseguenza il segnale MCLKOUT verra' settato alto al sopraggiungere del risultato in uscita, dopo la latenza L (T_{3+L}). Il segnale DZO (Zero Output Data) viene attivato in corrispondenza del fronte di salita di T_{3+L} per mostrare con quale ritardo esso agisce sull' uscita. Durante lo stesso ciclo T_{3+L} contemporaneamente ai risultati, vengono attivati i flag RMAX e IMAX (il

primo risultato parziale e' sempre maggiore del precedente), DSFO[2:0] e BFPO[5:0], mentre al ciclo successivo T_{3+L+1} il risultato e' zero per effetto della attivazione del segnale DZO ed i segnale RMAX,IMAX,BFPO e DSFO possono cambiare di stato.

In fig. 10 viene riportato il timing per il caso di un FIR filter. Al ciclo di clock T_0 il segnale ENA abilita i registri di I/O per l' impostazione dei parametri relativi a FC e DF, al sopraggiungere del clock T_1. Sull' impulso T_2 il segnale ENB abilita i registri di I/O per il latching di DSFI, DSFISEL,CSFI,DCO, DCI e BFPI con il fronte di salita di T_3; i segnali CCR,CCI,DCI e START sono attivati per registrare il loro valore sul SYSCLK prima che il corrispondente dato venga registrato con il segnale CLKCIN e CLKQIN. Lo START/STOP, che controlla il doppio accumulatore da 60 bit, e' settato alto per il primo dato in ingresso e rimane tale per tutta la durata dell'input dei dati andando poi basso prima della lettura dell' ultimo dato. All 'impulso di clock T_3 CLKCIN e CLKQIN registrano i dati nell' I/O e sul fronte di salita di T_4 parte l' esecuzione all' interno dell' LH9124. Il prossimo dato letto, viene processato sul fronte di salita di T_5, come pure il dato successivo viene processato sul fronte di salita di T_6. Sullo stesso T_6 viene settato a zero il livello della linea START/STOP per processare il punto $r_3 i_3$ come ultimo punto del passo. Si ricorda, come visto in precedenza, che questa linea va' portata bassa un ciclo di clock prima della lettura dell' ultimo dato. All' istante $T_{3+(L-1)}$ la linea DZO (Data Zero Output) e' settata bassa per azzerare il dato in uscita al ciclo di clock $T_{3+(L+1)}$. Al successivo ciclo $T_{3+(L+1)}$ viene portato a livello alto il segnale RMAX e IMAX (si ricorda che per il primo risultato si ha sempre valore max.) ed il livello MCLKOUT indica che il primo risultato $x_0 y_0$ e' gia' disponibile. Al ciclo $T_{3+(L+2)}$ il secondo risultato e cosi`via fino all'ultimo risultato.

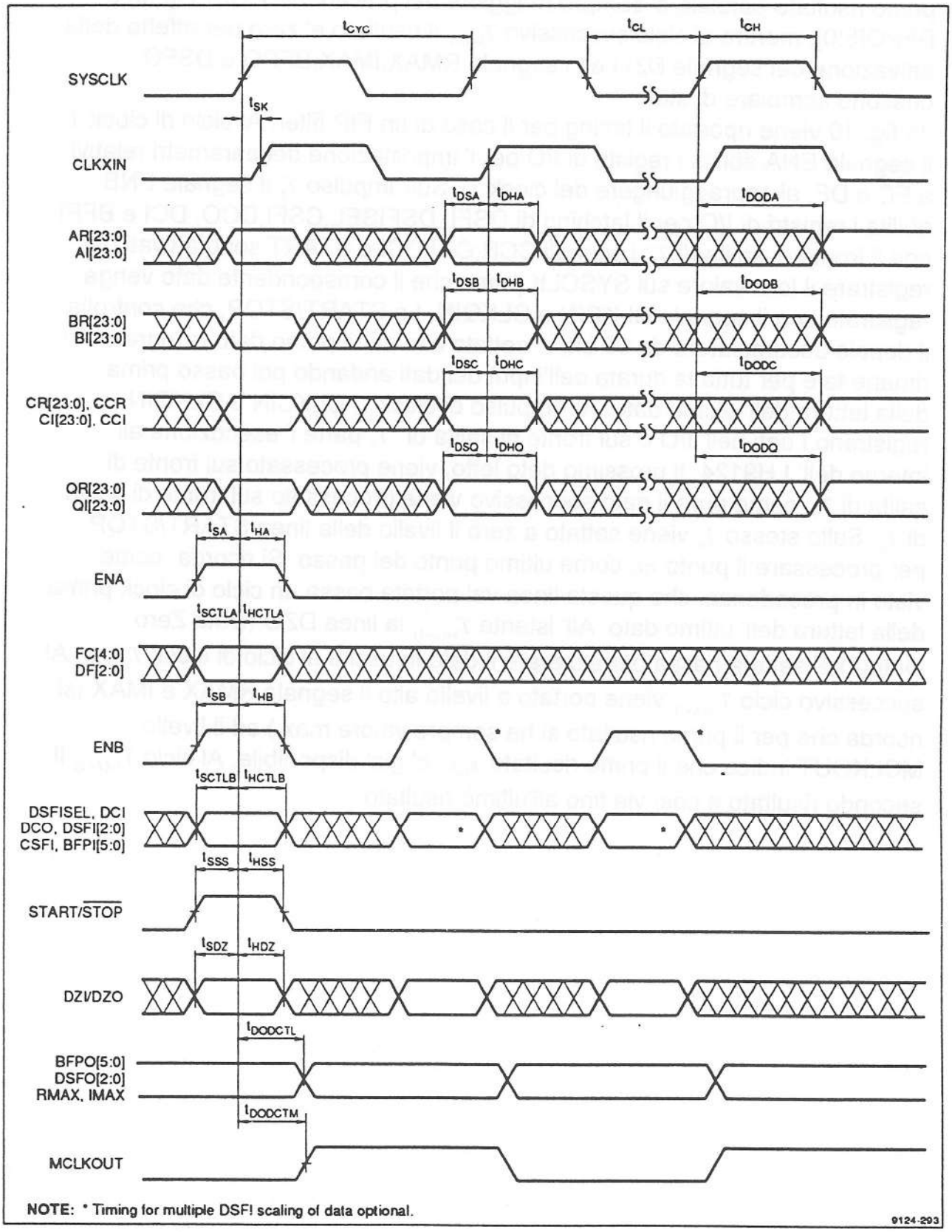
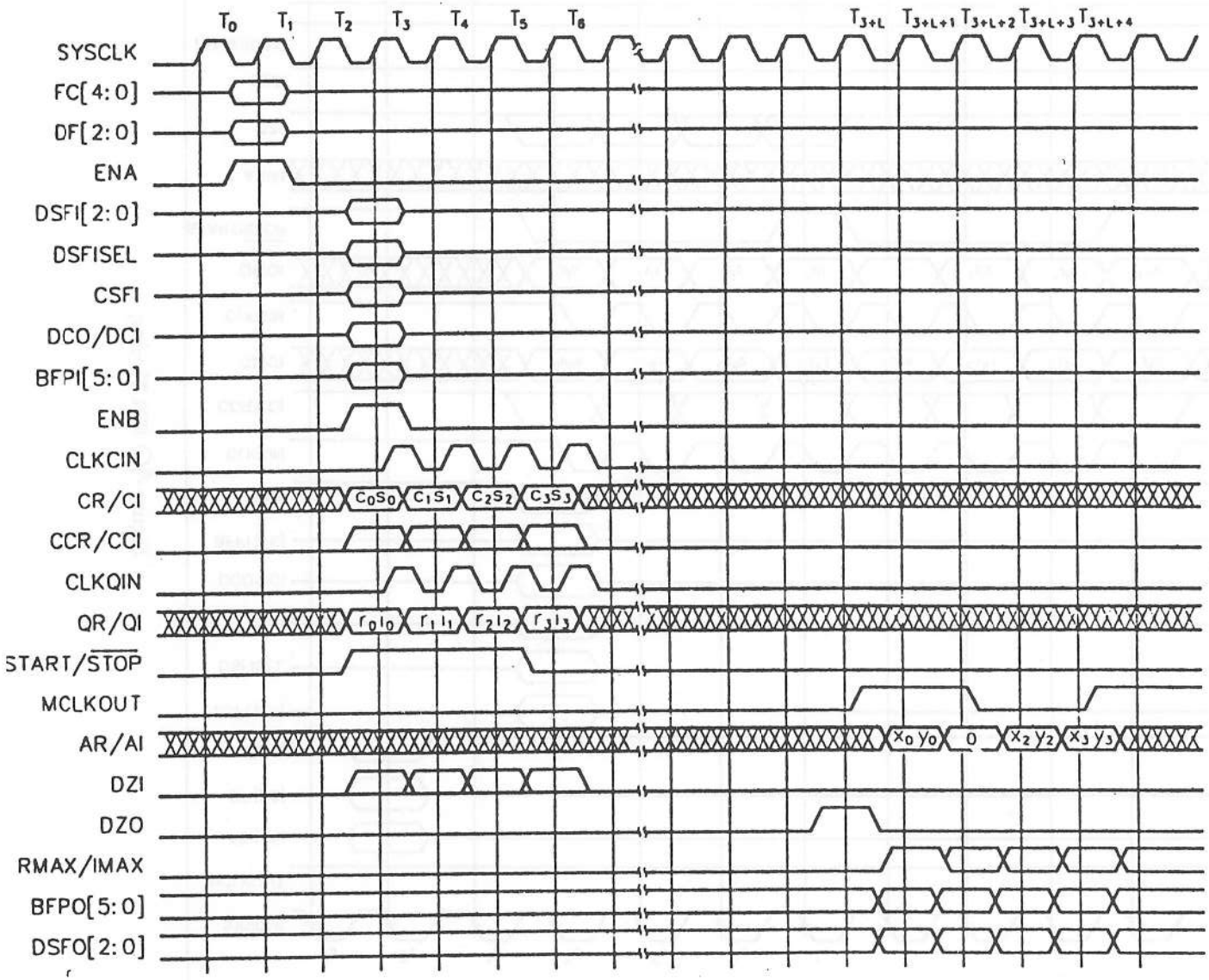


Figure 8 Basic Timing Example *

* This timing diagram is current as of this printing. Contact your SHARP sales representative for the latest specifications.



FUNZIONE	LATENZA
BWND2, BWND4, VMXM	20 cicli
BFLY16	68 cicli
MOVD	18 cicli
MOVD (+RQWC, RAWC)	8 cicli
Tutte le altre funzioni	18 cicli

- Latenza dovuta alla pipeline interna del DSP LH912-

FIG. 9

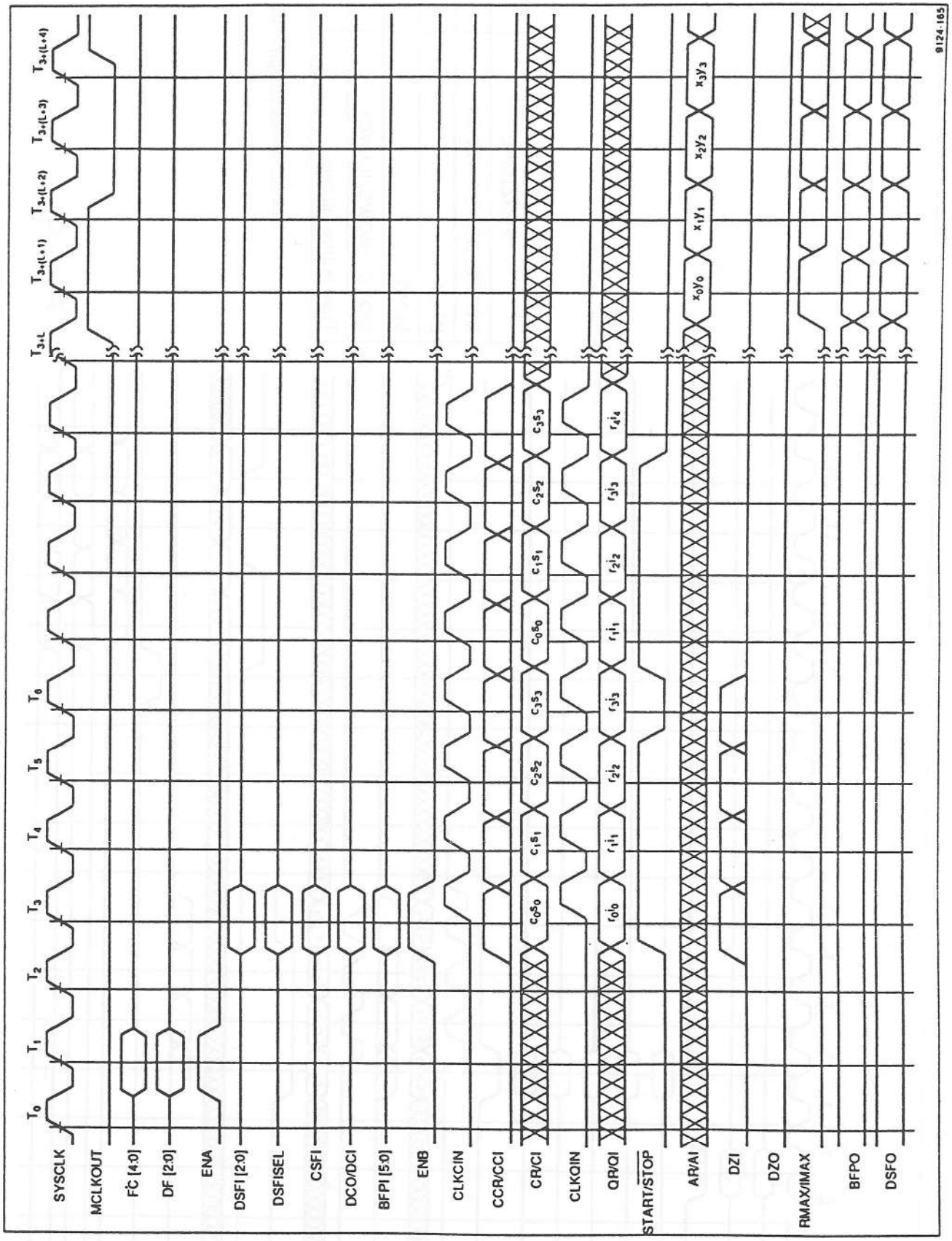


Figure 10 FIR Timing

LE ISTRUZIONI BASE.

Il DSP LH9124 implementa 26 funzioni, raggruppabili nei seguenti insiemi:

- Digital Signal Processing. (DSP)
- Aritmetica Vettoriale. (Vector Aritmetic)
- Utilizzo Generale. (General Purpose)
- Aritmetica Complessa. (Complex Aritmetic)
- Logica Vettoriale (Vector Logical)

Nella tabella che segue vengono riportate le funzioni raggruppate secondo quanto detto sopra. Nella prima colonna si trova l' opcode (in esadecimale) che viene impostato sul bus a 5 bit della Function Code (FC). Questo determina quale tipo di funzione il chip implementa sui dati nel passo corrente. Nella seconda colonna viene riportato il codice mnemonico per ogni funzione e nella terza, una breve descrizione della funzione.

Table 3
LH9124 Function Set

FC OPCODE (HEX)	FUNCTION MNEMONIC	DESCRIPTION
		DSP FUNCTIONS
02	BFLY2	Radix-2 butterflies.
01	BFLY4	Radix-4 butterfly.
00	BFLY16	Radix-16 butterfly.
05	BWND2	Radix-2 complex window butterflies. Performs radix-2 butterflies with a complex window function that is multiplied with the incoming data.
04	BWND4	Radix-4 complex window butterfly. Performs a radix-4 butterfly with a complex window function that is multiplied with the incoming data.
07	BRFT	Dual real FFT separation. Performs the separation of two real data streams for a two-at-a-time FFT operation.
06	BFCT	Fast cosine transform/double length (N output). Performs the fast cosine transform (FCT) operation.
0E	BFCT2	Fast cosine transform/double length (2N output). Performs the fast cosine transform double length operation.
08	BCFIR	Complex finite impulse response (FIR) filter.
09	BDFIR	Double real finite impulse response (FIR) filter.
0A	BRFIR	Real finite impulse response (FIR) filter.
		COMPLEX ARITHMETIC FUNCTIONS
10	CADD	Complex add. Adds complex input data to complex coefficient data.
0C	CMAG	Complex magnitude squared. Performs the magnitude squared of an input.
0D	CMUL	Complex multiply. Multiplies complex input data by complex coefficient data.
11	CSUB	Complex subtract. Subtracts complex coefficient data from complex input data.
		VECTOR ARITHMETIC FUNCTIONS
13	VABS	Vector absolute value. Determines the absolute value of the input data.
10	VADD	Vector add. Adds the input data pairs to the coefficient data pairs.
12	VMUL	Vector multiply. Multiplies the input data by the coefficient data.
1E	VMXM	Vector maximum and minimum. Determines the maximum/minimum vector of the input data.
11	VSUB	Vector subtract. Subtracts the coefficient data from the input data.
		VECTOR LOGICAL FUNCTIONS
18	VNAND	Vector logical NAND. Performs a logical NAND between the input data and the coefficient data.
1A	VNOR	Vector logical NOR. Performs a logical NOR between the input data pairs and the coefficient pairs.
19	VPAS	Vector logical pass. Performs a VPAS on the input data.
1B	VXNOR	Vector logical exclusive NOR. Performs a logical exclusive NOR between the input data pairs and the coefficient pairs.
		GENERAL PURPOSE FUNCTIONS
1D	MOVC	Move coefficient data from the coefficient port to another (A,B,Q) port.
1C	MOVD	Move data from one port (A,B,Q) to another port (A,B,C,Q).

ANALISI DELLE FUNZIONI BASE.

Come abbiamo appena visto, il set di funzioni del DSP è tale da potere risolvere una vasta gamma di problematiche di elaborazione. Visto che in questa sede non è possibile descrivere dettagliatamente funzione per funzione, se ne descriverà in dettaglio solo una piccola parte a puro scopo dimostrativo, mentre alla rimanente parte si accennerà in modo sommario. Siccome uno dei set di funzioni che verrà utilizzato maggiormente nel particolare campo di impiego qui considerato è quello relativo calcolo delle FFT, si entrerà nel dettaglio delle singole funzioni BFLY2, BFLY4, BFLY16 più alcune funzioni correlate con queste.

BFLY2 (Opcode: 02) Radix-2 Butterfly
Latenza=18

In fig.11 viene riportata la rappresentazione grafica dell' operazione eseguita dalla funione BFLY2. Questa funzione calcola due Butterfly radix-2 complesse alla volta, per cui necessita, prima di essere implementata, che siano già disponibili almeno quattro campionamenti del segnale da trasformare. In sintesi le operazioni che esegue sono quelle già viste in precedenza e che qui ripetiamo:

$$\begin{aligned} A' &= A + BW_N^{k1} \\ B' &= A - BW_N^{k1} \\ C' &= C + DW_N^{k3} \\ D' &= C - DW_N^{k3} \end{aligned} \qquad (2.1)$$

dove A,B,C,D,A',B',C',D' sono dati complessi ed il termine $W = e^{-j2\pi/N}$
I coefficienti W_N^k vengono letti attraverso la porta C.
Per fare un esempio, consideriamo quattro punti e poniamo:

$$\begin{aligned} A &= r_0 + ji_0 \\ B &= r_1 + ji_1 \\ C &= r_2 + ji_2 \\ D &= r_3 + ji_3 \end{aligned} \qquad (2.2)$$

e

$$\begin{aligned} W_N^{K1} &= c_1 + js_1 \\ W_N^{K3} &= c_3 + js_3 \end{aligned} \qquad (2.3)$$

sostituendo le (2.2) e (2.3) nelle (2.1) si ottiene:

$$
\begin{aligned}
A' &= [r_0 + (r_1c_1 - i_1s_1)] + j[i_0 + (i_1c_1 + r_1s_1)] = x_0 + jy_0 \\
B' &= [r_0 - (r_1c_1 - i_1s_1)] + j[i_0 - (i_1c_1 + r_1s_1)] = x_1 + jy_1 \\
C' &= [r_2 + (r_3c_3 - i_3s_3)] + j[i_2 + (i_3c_3 + r_3s_3)] = x_2 + jy_2 \\
D' &= [r_2 - (r_3c_3 - i_3s_3)] + j[i_2 - (i_3c_3 + r_3s_3)] = x_3 + jy_3
\end{aligned}
\qquad (2.4)
$$

Le parti reali di A',B',C' e D' vengono presentate nella parte reale della porta scelta nella definizione del flusso dei dati, mentre quelle immaginarie nella parte immaginaria della stessa porta nell' ordine x_0, x_1, x_2, x_3 e y_0, y_1, y_2, y_3 rispettivamente, con aprossimazione ai 24+24 bit più significativi.

BLY4 (opcode 01) Radix-4 Butterfly
Latenza=18

Questa istruzione calcola la Butterfly Radix-4 e viene rappresentata graficamente in fig 12. La BFLY4 calcola con un solo stadio la FFT su quattro punti dove, invece, occorrerebbero 2 stadi BFLY2. Dalla stessa figura si possono ricavare facilmente le relazioni fra ingresso ed uscita dello stadio BFLY4:

$$
\begin{aligned}
A' &= A + jBW_N^{k1} + CW_N^{k2} + DW_N^{k3} \qquad N = 4 \qquad W_N = e^{-j\frac{2\pi}{4}} \\
B' &= A - jBW_N^{k1} - CW_N^{k2} + jDW_N^{k3} \\
C' &= A - BW_N^{k1} + CW_N^{k2} - DW_N^{k3} \\
D' &= A + jBW_N^{k1} - CW_N^{k2} - DW_N^{k3}
\end{aligned}
\qquad (2.5)
$$

Come nel caso della BFLY2, i dati in ingresso sono considerati essere complessi, i twiddle coefficient inseriti tramite la solita porta C e sono necessari almeno 4 dati campionati prima di fare partire il passo. Anche in questo caso, per maggiore chiarezza, mostriamo un esempio di BFLY4 operata su 4 punti complessi, ponendo:

$$
\begin{aligned}
A &= r_0 + ji_0 \\
B &= r_1 + ji_1 \\
C &= r_2 + ji_2 \\
D &= r_3 + ji_3
\end{aligned}
\qquad (2.6)
$$

con coefficienti:

$$W_N^{k1} = c_1 + js_1$$
$$W_N^{k2} = c_2 + js_2 \qquad (2.7)$$
$$W_N^{k3} = c_3 + js_3$$

Analogamente al caso precedente, sostituendo le (2.6) e (2.7) in (2.5) si ottiene:

$$\begin{aligned}
A' &= [r_0 + (r_1c_1 - i_1s_1) + (r_2c_2 - i_2s_2) + (r_3c_3 - i_3s_3)] \\
&\quad + j[i_0 + (i_1c_1 + r_1s_1) + (i_2c_2 + r_2s_2) + (i_3c_3 + i_3s_3)] \\
&= x_0 + jy_0 \\
B' &= [r_0 + (i_1c_1 - r_1s_1) - (r_2c_2 - i_2s_2) - (i_3c_3 + r_3s_3)] \\
&\quad + j[i_0 - (r_1c_1 - i_1s_1) - (i_2c_2 + r_2s_2) + (r_3c_3 - i_3s_3)] \\
&= x_1 + jy_{10} \\
C' &= [r_0 - (r_1c_1 - i_1s_1) + (r_2c_2 - i_2s_2) - (r_3c_3 - i_3s_3)] \\
&\quad + j[i_0 - (i_1c_1 + r_1s_1) + (i_2c_2 + r_2s_2) - (i_3c_3 + i_3s_3)] \\
&= x_2 + jy_2 \\
D' &= [r_0 - (i_1c_1 + r_1s_1) - (r_2c_2 - i_2s_2) + (i_3c_3 + r_3s_3)] \\
&\quad + j[i_0 + (r_1c_1 - i_1s_1) - (i_2c_2 + r_2s_2) - (r_3c_3 - i_3s_3)] \\
&= x_3 + jy_3
\end{aligned} \qquad (2.8)$$

Come si può notare, l' ordine di uscita dei dati rimane lo stesso di quello del passo BLFY2. La funzione BLY4 richiede una particolare sequenza di indirizzamento dei dati e coefficienti, che viene fornita dal generatore di indirizzi LH9320. Confrontando questa funzione con la precedente BLFY2, si nota che i coefficienti del primo stadio sono tutti unitari e, trascurando i cicli di latenza, è veloce il doppio. Anche in questo caso, siccome nel primo stadio non è necessario leggere i coefficienti twiddle dalla porta C, si può approffittare, quando richiesto, per effettuare la windowing dei campionamenti che entrano attraverso la porta Q. Questo si ottiene semplicemente, e senza perdita aggiuntiva di tempo, moltiplicando gli stessi man mano che entrano per i corrispondenti coefficienti della finestra. In ambedue i casi (butterfly radix-2 e 4) è possibile quindi usare le funzioni BNWD2 e BNWD4, implementate nell' LH9124 che permettono di effettuare i suddetti butterfly contemporaneamente alla windowing.

BFLY16 (opcode 00) Radix-16 Butterfly
Latenza=68

Questa funzione calcola una butterfly radix-16 mediante il calcolo di due butterfly radix-4 adiacenti come rappresentato graficamente in fig.13. L' architettura dell' LH9124 è ottimizzata per mettere a disposizione questa

potente funzione che è in grado di eseguire la butterfly in 16 cicli per ogni 16 punti eguagliando in tempo di esecuzione, la BFLY2 e BFLY4. L' esecuzione della funzione BFLY16 porta il chip ad esprimere le sue massime prestazioni con 6 moltiplicazioni ed 11 addizioni per ogni ciclo di clock che, a 40 MHz, corrispondono a ben 680 milioni di operazioni in fixed point al secondo! Siccome l' esecuzione di questa funzione equivale alla esecuzione di due BFLY4 e 4 BFLY2, si può dire che fornisce prestazioni doppie e quadruple rispetto alle stesse. Nel calcolo di una qualsiasi FFT non è sempre possibile scegliere solo passi radix-16, perchè la scelta dei passi per la struttura ottimale è comunque legata al numero di punti da trattare. Da una attenta analisi della funzione, ci si rende anche conto che non è possibile avere una funzione di windowing nel primo stadio per il fatto che non tutti i coefficienti sono unitari per cui non esiste, purtroppo, una funzione BWND16. Se il primo stadio della FFT è un radix-16 ed è richiesta una windowing, occorre un passo addizionale per tale operazione. Per questa funzione viene richiesto che l' address generator LH9320 fornisca un pattern di indirizzi ad hoc, per la lettura dei coefficienti dalla porta C. Viene ora affrontata, come nei due casi precedenti, una analisi del passo che, risultando essere molto complessa, mette ancor più in evidenza la particolare architettura che differenzia questo DSP da tutti gli altri cosiddetti " standard".
I 16 dati (complessi) letti dalla porta Q sono esprimibili come segue:

$$\begin{array}{llll} A = r_0 + ji_0 & \mathrm{E} = \mathrm{r}_4 + ji_4 & \mathrm{J} = \mathrm{r}_8 + ji_8 & \mathrm{N} = \mathrm{r}_{12} + ji_{12} \\ B = r_1 + ji_1 & \mathrm{F} = \mathrm{r}_5 + ji_5 & \mathrm{K} = \mathrm{r}_9 + ji_9 & \mathrm{P} = \mathrm{r}_{13} + ji_{13} \\ C = r_2 + ji_3 & \mathrm{G} = \mathrm{r}_6 + ji_6 & \mathrm{L} = \mathrm{r}_{10} + ji_{10} & \mathrm{R} = \mathrm{r}_{14} + ji_{14} \\ D = r_3 + ji_3 & \mathrm{H} = \mathrm{r}_7 + ji_7 & \mathrm{M} = \mathrm{r}_{11} + ji_{11} & \mathrm{S} = \mathrm{r}_{15} + ji_{15} \end{array} \qquad (2.9)$$

mentre i coefficienti letti dalla porta C:

$$\begin{array}{llll} {}^{0} = c_0 js_0 & W^{k4} = c_4 + js_4 & W^{k8} = c_8 + js_8 & W^{k12} = c_{12} + js_{12} \\ {}^{k} = c_1 + js_1 & W^{k5} = c_5 + js_5 & W^{k9} = c_9 + js_9 & W^{k13} = c_{13} + js_{13} \\ {}^{k2} = c_{2+} js_2 & W^{k6} = c_6 + js_6 & W^{k10} = c_{10} + js_{10} & W^{k14} = c_{14} + js_{14} \\ {}^{k3} = c_3 + js_3 & W^{k7} = c_7 + js_7 & W^{k11} = c_{11} + js_{11} & W^{k15} = c_{15} + js_{15} \end{array} \qquad (2.10)$$

A questo punto combinando le (2.9) e (2.10) secondo il diagramma riportato per la BFLY16, si ottengono le espressioni delle uscite:

$$\begin{aligned} A' &= A + EW^{k1} + JW^{k2} + NW^{k3} \\ B' &= A - jEW^{k1} - JW^{k2} + jNW^{k3} \\ C' &= A - EW^{k1} + JW^{k2} - NW^{k3} \\ A' &= A + jEW^{k1} - JW^{k2} - jNW^{k3} \end{aligned} \qquad (2.11)$$

dove le uscite A',B',C',D' sono a loro volta espresse da:

$$
\begin{aligned}
A' &= A + BW^{k} + CW^{2k} + DW^{3k} = \\
&= \left[r_0 + (r_1c_1 - i_1s_1) + (r_2c_2 - i_2s_2) + (r_3c_3 - i_3s_3)\right] + \\
&\quad j\left[i_0 + (i_1c_1 + r_1s_1) + (i_2c_2 + i_2s_2) + (i_3c_3 - r_3s_3)\right] \\
E' &= EW^{4k} + FW^{5k} + GW^{6k} + HW^{7k} = \\
&= \left[r_4 + (r_5c_5 - i_5s_5) + (r_6c_6 - i_6s_6) + (r_7c_7 - i_7s_7)\right] + \\
&\quad j\left[i_4 + (i_5c_5 + r_5s_5) + (i_6c_6 + i_6s_6) + (i_7c_7 + r_7s_7)\right]
\end{aligned}
\qquad (2.12)
$$

$$
\begin{aligned}
J' &= JW^{8k} + KW^{9k} + LW^{11k} + MW^{11k} = \\
&= \left[r_8 + (r_9c_9 - i_9s_9) + (r_{10}c_{10} - i_{10}s_{10}) + (r_{11}c_{11} - i_{11}s_{11})\right] + \\
&\quad j\left[i_8 + (i_9c_9 + r_9s_9) + (i_{10}c_{10} + i_{10}s_{10}) + (i_{11}c_{11} - r_{11}s_{11})\right] \\
N' &= NW^{12k} + PW^{13k} + RW^{14k} + SW^{15k} = \\
&= \left[r_{12} + (r_{13}c_{13} - i_{13}s_{13}) + (r_{14}c_{14} - i_{14}s_{14}) + (r_{15}c_{15} - i_{15}s_{15})\right] + \\
&\quad j\left[i_{12} + (i_{13}c_{13} + r_{13}s_{13}) + (i_{14}c_{14} + i_{14}s_{14}) + (i_{14}c_{14} - r_{14}s_{14})\right]
\end{aligned}
$$

Vengono ora descritte, sommariamente, le rimanenti funzioni implementate sull' LH9124. Per chiarimenti od approfondimenti sulle funioni descritte, consultare il data sheet dell' LH9124 [cap.3].

BWND2 (opcode 05) Radix-2 Complex Window Butterfly
Latenza=20

La funzione implementa il windowing dei dati (primo stadio), moltiplicandoli per i coefficienti della finestra complessa man mano che arrivano dalla porta Q. Quando la particolare applicazione richiede tale windowing, questa funzione riduce i tempi che sarebbero richiesti da una architettura standard. I coefficienti della window entrano dalla porta C ed almeno 4 punti sono richiesti prima del lancio di tale funzione. La descrizione grafica appare in fig.14. I dati in uscita sono espressi dalle:

$$
\begin{aligned}
A' &= AW^{K0} + BW^{k1} \\
B' &= AW^{K0} - BW^{k1} \\
C' &= CW^{K2} + DW^{k3} \\
D' &= CW^{K2} + DW^{k3}
\end{aligned}
\qquad (2.13)
$$

con A,B,C,D,A';B';C' e D' numeri complessi ed i coefficienti W sono i coefficienti della window. Se i dati in ingresso sono espressi dalle:

$$
\begin{aligned}
A &= r_0 + ji_0 \qquad & \mathrm{W}^{k0} &= c_0 + js_0 \\
B &= r_1 + ji_1 \qquad & \mathrm{W}^{k1} &= c_1 + js_1 \\
C &= r_2 + ji_2 \qquad & \mathrm{W}^{k2} &= c_2 + js_2 \\
D &= r_3 + ji_3 \qquad & \mathrm{W}^{k3} &= c_3 + js_3
\end{aligned}
\tag{2.14}
$$

sostituendo le (2.14) nella (2.13) si hanno le espressioni delle uscite:

$$
\begin{aligned}
A' &= \left[(r_0 * c_0 - i_0 * s_0) + (r_1 * c_1 - i_1 * s_1)\right] + j\left[(i_0 * c_0 + r_0 * s_0) + (i_1 * c_1 + r_1 * s_1)\right] = x_0 + y_0 \\
B' &= \left[(r_0 * c_0 - i_0 * s_0) - (r_1 * c_1 - i_1 * s_1)\right] + j\left[(i_0 * c_0 + r_0 * s_0) - (i_1 * c_1 + r_1 * s_1)\right] = x_1 + y_1 \\
C' &= \left[(r_2 * c_2 - i_2 * s_2) + (r_3 * c_3 - i_3 * s_3)\right] + j\left[(i_2 * c_2 + r_2 * s_2) + (i_3 * c_3 + r_3 * s_3)\right] = x_2 + y_2 \\
D' &= \left[(r_2 * c_2 - i_2 * s_2) - (r_3 * c_3 - i_3 * s_3)\right] + j\left[(i_2 * c_2 + r_2 * s_2) - (i_3 * c_3 + r_3 * s_3)\right] = x_3 + y_3
\end{aligned}
$$

Le parti reali di A',B',C' e D' sono disponibili sulla porta reale scelta, mentre la parte immaginaria degli stessi, sulla porta immaginaria.

BWND4 (opcode 04) Radix-4 Complex Window Butterfly
Latenza=20

Questa funzione implementa il windowing dei dati (primo stadio), moltiplicandoli per i coefficienti di una funzione complessa di windowing, man mano che arrivano dalla porta di ingresso (lettura) dei dati Q, aumentando in tal modo la velocità. Anche in questo caso i coefficienti della window, arrivano dalla porta C. La rappresentazione grafica compare in fig.15. La parte reale è disponibile in ordine naturale di indici nella porta reale di destinazione scelta, la parte immaginaria è, allo stesso modo, posta sulla parte immaginaria della stessa porta. Il risultato è esprimibile tramite le formule seguenti:

$$
\begin{aligned}
A' &= AW^{k0} + BW^{k1} + CW^{k2} + KW^{k3} \\
A' &= BW^{k0} - BW^{k1} - CW^{k2} - KW^{k3} \\
C' &= AW^{k0} - BW^{k1} + CW^{k2} - KW^{k3} \\
A' &= AW^{k0} + BW^{k1} - CW^{k2} - KW^{k3}
\end{aligned}
$$

con A',B',C',D',A,B,C, e D sono numeri complessi mentre i coefficienti W sono i coefficienti della window. Se i dati e coefficienti in ingresso sono espressi da:

$$
\begin{aligned}
A &= r_0 + ji_0 \qquad & \mathrm{W}^{k0} &= c_0 + js_0 \\
B &= r_1 + ji_1 \qquad & \mathrm{W}^{k1} &= c_1 + js_1 \\
C &= r_2 + ji_2 \qquad & \mathrm{W}^{k2} &= c_2 + js_2 \\
D &= r_3 + ji_3 \qquad & \mathrm{W}^{k3} &= c_3 + js_3
\end{aligned}
$$

i dati in uscita assumono la forma:

$$A' = [(r_0 * c_0 - i_0 * s_0) + (r_1 * c_1 - i_1 * s_1) + (r_2 * c_2 - i_2 * s_2) + (r_3 * c_3 - i_3 * s_3)] +$$
$$j[(i_0 * c_0 + r_0 * s_0) + (i_1 * c_1 + r_1 * s_1) + (i_2 * c_2 + r_2 * s_2) + (i_3 * c_3 + r_3 * s_3)]$$
$$= x_0 + jy_0$$
$$B' = [(r_0 * c_0 - i_0 * s_0) + (r_1 * c_1 + i_1 * s_1) - (r_2 * c_2 - i_2 * s_2) - (r_3 * c_3 + i_3 * s_3)] +$$
$$j[(i_0 * c_0 + r_0 * s_0) - (i_1 * c_1 - r_1 * s_1) - (i_2 * c_2 + r_2 * s_2) + (i_3 * c_3 - r_3 * s_3)]$$
$$= x_1 + jy_1$$
$$C' = [(r_0 * c_0 - i_0 * s_0) - (r_1 * c_1 - i_1 * s_1) + (r_2 * c_2 - i_2 * s_2) - (r_3 * c_3 - i_3 * s_3)] +$$
$$j[(i_0 * c_0 + r_0 * s_0) - (i_1 * c_1 + r_1 * s_1) + (i_2 * c_2 + r_2 * s_2) - (i_3 * c_3 + r_3 * s_3)]$$
$$= x_2 + jy_2$$
$$D' = [(r_0 * c_0 - i_0 * s_0) - (r_1 * c_1 + i_1 * s_1) - (r_2 * c_2 - i_2 * s_2) + (r_3 * c_3 + i_3 * s_3)] +$$
$$j[(i_0 * c_0 + r_0 * s_0) + (i_1 * c_1 - r_1 * s_1) - (i_2 * c_2 + r_2 * s_2) - (i_3 * c_3 - r_3 * s_3)]$$
$$= x_3 + jy_3$$

BRFT (opcode 07) Dual Real FFT Separation.
Latenza=18

Tale funzione esegue la separazione in uscita della FFT globale in due FFT reali e necessita di almeno 4 punti in ingresso. Per calcolare la FFT di due set di N dati reali come singolo set di N dati complessi (2N), si pongono i primi N dati reali sul bus reale ed i rimanenti N dati reali sul bus immaginario della catena di funzioni BFLYx o BFLYxx ed il risultato complesso, parcheggiato tramite una delle porte A o B, fornito alla funzione BFRT che ne rende le due FFT reali come in fig.16. In questa funzione la porta dei coefficienti C, viene ignorata. L' uscita della BRFT è formata da due FFT di N punti distinte, composte da coppie alternate di numeri complessi nel senso che i punti ad indice pari costituiscono la FFT dei primi N punti reali entrati e quelli dispari la FFT dei rimanenti N punti reali entrati. Con la simbologia usata nella stessa fig.16 possiamo scrivere l' espressione delle due FFT presenti in uscita come:

$$FFT1 = [R(k) + R(N-k)] + j[I(k) - I(N-k)]$$
$$FFT2 = [I(k) + I(N-k)] - j[R(k) - R(N-k)]$$

con K=1,2,3,....N-1

La parti reali di FFT1 e FFT2 vengono trasferite nella parte reale della porta di destinazione in ordine naturale, quella immaginaria, allo stesso modo, nella parte immagina della stessa.

BFCT (opcode 06) Fast Cosine Transform/Double Length (N Output)
Latenza=18

La funzione BFCT effettua il calcolo della FCT (Fast Cosine Transform). Ricombina in uscita (fig.17) i dati presentati all' ingresso del passo come risultato di una FFT complessa processata come FFT reale di 2N punti forniti come N punti complessi. I dati in uscita sono una sequenza ripetuta di coppie complesse che forniscono solo lo spettro positivo della trasformazione (i punti ad indice pari sono presentati nel bus reale e quelli dispari su quello immaginario). La parte reale dello spettro di 2N punti reali, viene presentata in uscita sulla parte reale della porta scelta, mentre quella immaginaria sulla parte immaginaria della stessa. L' ordine di presentazione è quello naturale e, ovviamente, vengono forniti in uscita N dati.

BFCT2 (opcode 0E) Fast Cosine Transform/Double Lenght (2N Output)
Latenza=18

Ricombina in uscita i dati complessi, presenti in ingresso, che provengono da una butterfly (BFLYx o xx) complesso di 2N dati reali forniti come array di N dati complessi (fig.18). E' simile alla BFCT, fornisce in più lo spettro "riflesso" utile in eventuali IFFT. I dati in uscita vengono presentati nello stesso modo visto per la BFCT, ma il numero dei dati forniti è in questo caso 2N.

BCFIR (opcode 08) Complex Finite Impulse Response Filter
Latenza=18

La funzione implementa un filtraggio di tipo FIR, non recursivo, sui dati in imgresso (fig.19). Questo è il filtro comunemente conosciuto come Tapped Delay Line o filtro a struttura trasversale. In questo caso la risposta all' impulso desiderata (che determina le caratteristiche del filtro), deve essere caricata nella memoria dei coefficienti raggiungibile tramite la porta C. La risposta è data da:

$$y(n) = c(0)x(n) + c(1)x(n-1)....+c(N-1)x(n-N+1)$$

dove:

y(n)= uscita filtrata.
c(n)= risposta del filtro all' impulso di Dirac.
x(n)= dati complessi in ingresso.

La parte reale dei risultati intermedi del FIR filter, è disponibile in uscita nella parte reale della porta prescelta ed in ordine naturale, la parte immaginaria nella parte immaginaria della stessa porta.

BDFIR (opcode 09) Double Real FIR Filter.
Latenza=18

Questa funzione è del tutto simile alla precedente, solo che esegue il FIR filter su un doppio set di dati (fig 20).

BRFIR (opcode 0A) Real FIR FIlter.
Latenza=18

Esegue la stessa operazione di filtraggio di tipo FIR, ma su dati di ingresso reali (fig.21). L' uscita è data da:

$$y(n) = c(0)x(n) + c(1)x(n-1)....+c(N-1)x(n-N+1)$$

dove y(n), c(n) e x(n) sono numeri reali. Da ricordare che i coefficienti devono essere forniti come coppie nel modo che segue:

(a(0),a(1)), (a(2),a(3)), a(4),a(5)).....a(N-1)a(N))

CADD (opcode 10) Complex Add
Latenza=18

La funzione CADD (fig.22) esegue la somma dei dati complessi e coefficienti. I coefficienti sono inseriti tramite la porta C ed i dati da qualunque porta. L' uscita è data da:

A'=A+C

dove A', A e C sono numeri complessi. Occorrono almeno quattro punti per iniziare l' operazione e, ovviamente, quattro coefficienti:

$A = r_0 + ji_0 \qquad C_0 = c_0 + js_0$
$B = r_1 + ji_1 \qquad C_1 = c_1 + js_1$
$C = r_2 + ji_2 \qquad C_2 = c_2 + js_2$
$D = r_3 + ji_3 \qquad C_3 = c_3 + js_3$

L' uscita del blocco di fig.22 è data dalle espressioni che seguono:

$$A' = (r_0 + c_0) + j(i_0 + s_0) = x_0 + jy_0$$
$$B' = (r_1 + c_1) + j(i_1 + s_1) = x_1 + jy_1$$
$$C' = (r_2 + c_2) + j(i_2 + s_2) = x_2 + jy_2$$
$$D' = (r_3 + c_3) + j(i_3 + s_3) = x_3 + jy_3$$

Il risultato è disponibile in uscita con lo stesso ordine dei dati in ingresso.

CMAG (opcode 0C) Magnitude Squared.
Latenza=18

CMAG fornisce il modulo quadro di due numeri complessi dati in ingresso (fig.23) da qualsiasi delle tre porte disponibili (la porta dei coefficienti C non è ovviamente usata). L' uscita è presentata nella forma:

$$A' = A_r^2 + A_i^2$$

dove A', A_r, A_i sono il primo è un numero reale e gli ultimi due la parte reale ed immaginaria del dato in ingresso. Considerando, ad esempio, di avere quattro dati complessi in ingresso di cui calcolarne il modulo quadro:

$$A = r_0 + ji_0 \qquad C_0 = ignorati$$
$$B = r_1 + ji_1 \qquad C_1 = ignorati$$
$$C = r_2 + ji_2 \qquad C_2 = ignorati$$
$$D = r_3 + ji_3 \qquad C_3 = ignorati$$

le relative uscite (48 bit) risulteranno essere:

$$A' = MSB(r_0 * r_0 + i_0 * i_0), MSB(r_1 * r_1 + i_1 * i_1) = x_0, y_0$$
$$B' = LSB(r_0 * r_0 + i_0 * i_0), LSB(r_1 * r_1 + i_1 * i_1) = x_1, y_1$$
$$C' = MSB(r_2 * r_2 + i_2 * i_2), MSB(r_3 * r_3 + i_3 * i_3) = x_2, y_2$$
$$D' = LSB(r_2 * r_2 + i_2 * i_2), LSB(r_3 * r_3 + i_3 * i_3) = x_3, y_3$$

Il timing relativo a questa funzione, è presentato in fig.24

CMUL (opcode 0D) Complex Multiply.
Latenza=18

Con questa funzione si ha la possibilità di moltiplicare un ingresso complesso con i coefficienti complessi provenienti dalla porta C (fig 25):

A'=A*C

con A', A e C numeri complessi. Questa funzione richiede almeno 4 numeri in ingresso. Se, ad esempio, si hanno in ingresso i dati:

$$A = r_0 + ji_0 \qquad C_0 = c_0 + js_0$$
$$B = r_1 + ji_1 \qquad C_1 = c_1 + js_1$$
$$C = r_2 + ji_2 \qquad C_2 = c_2 + js_2$$
$$D = r_3 + ji_3 \qquad C_3 = c_3 + js_3$$

le uscite complesse sono date dalle relazioni:

$$A' = (r_0 * c_0 - i_0 * s_0) + j(i_0 * c_0 + r_0 * s_0) = x_0 + jy_0$$
$$B' = (r_1 * c_1 - i_1 * s_1) + j(i_1 * c_1 + r_1 * s_1) = x_1 + jy_1$$
$$C' = (r_2 * c_2 - i_2 * s_2) + j(i_2 * c_2 + r_2 * s_2) = x_2 + jy_2$$
$$D' = (r_3 * c_3 - i_3 * s_3) + j(i_3 * c_3 + r_3 * s_3) = x_3 + jy_3$$

ed hanno lo stesso ordine dei dati in ingresso.

CSUB (opcode 11) Complex Subtract.
Latenza=18

La CSUB esegue la sottrazione tra i coefficienti complessi e i dati complessi in ingresso (fig. 26). L' uscita è nella forma:

A'=A-C

con A', A,C numeri complessi. Richiede almeno quattro numeri in ingresso:

$$A = r_0 + ji_0 \qquad C_0 = c_0 + js_0$$
$$B = r_1 + ji_1 \qquad C_1 = c_1 + js_1$$
$$C = r_2 + ji_2 \qquad C_2 = c_2 + js_2$$
$$D = r_3 + ji_3 \qquad C_3 = c_3 + js_3$$

I dati in uscita hanno la forma:

$$A' = (r_0 + c_0) + j(i_0 + s_0) = x_0 + jy_0$$
$$B' = (r_1 + c_1) + j(i_1 + s_1) = x_1 + jy_1$$
$$C' = (r_2 + c_2) + j(i_2 + s_2) = x_2 + jy_2$$
$$D' = (r_3 + c_3) + j(i_3 + s_3) = x_3 + jy_3$$

ed hanno lo stesso ordine di come sono stati introdotti nel blocco.

VABS (opcode 13) Vector Absolute Value.
Latenza=18

Determina il valore assoluto del dato in ingresso (fig.27). L' uscita assume la forma

A'=|A|

con A' e A coppie di numeri reali. La funzione richiede almeno 4 numeri in ingresso del tipo

$A = r_0, i_0 \qquad C_0 = ignorat$
$B = r_1, i_1 \qquad C_1 = ignorat$
$C = r_2, i_2 \qquad C_2 = ignorat$
$D = r_3, i_3 \qquad C_3 = ignorat$

I dati in uscita assumono la forma:

$A' = (|r_0|), (|i_0|) = x_0, y_0$
$B' = (|r_1|), (|i_1|) = x_1, y_1$
$C' = (|r_2|), (|i_2|) = x_2, y_2$
$D' = (|r_3|), (|i_3|) = x_3, y_3$

e sono resi disponibili nello stesso ordine con cui sono stati inseriti.

VADD (opcode 10) Vector Add.
Latenza=18

Somma coppie di dati in ingresso con coppie di coefficienti (fig.28). L' uscita si presenta nella forma:

A'=A+C

con A', A e C coppie di dati reali. Occorrono almeno quattro punti per iniziare il processing dei dati in ingresso, che possono essere del tipo:

$A = r_0, i_0 \qquad C_0 = c_0, s_0$
$B = r_1, i_1 \qquad C_1 = c_1, s_1$
$C = r_2, i_2 \qquad C_2 = c_2, s_2$
$D = r_3, i_3 \qquad C_3 = c_3, s_3$

I risultati in uscita sono disponibili nello stesso ordine con cui si sono entrati i dati:

$A' = (r_0 + c_0), (i_0 + s_0) = x_0, y_0$
$B' = (r_1 + c_1), (i_1 + s_1) = x_1, y_1$
$C' = (r_2 + c_2), (i_2 + s_2) = x_2, y_2$
$D' = (r_3 + c_3), (i_3 + s_3) = x_3, y_3$

VMUL (opcode12) Vector Multiply.
Latenza=18

La funzione permette il prodotto tra coppie di numeri reali in ingresso e coppie di coefficienti reali (fig.29). L' uscita viene data nella forma:

A'=A*C

dove A',A e C sono coppie reali. del tipo:

$A = r_0, i_0 \qquad C_0 = c_0, s_0$
$B = r_1, i_1 \qquad C_1 = c_1, s_1$
$C = r_2, i_2 \qquad C_2 = c_2, s_2$
$D = r_3, i_3 \qquad C_3 = c_3, s_3$

I dati in uscita sono, in particolare, dati dalle:

$A' = (r_0 * c_0), (i_0 * s_0) = x_0, y_0$
$B' = (r_1 * c_1), (i_1 * s_1) = x_1, y_1$
$C' = (r_2 * c_2), (i_2 * s_2) = x_2, y_2$
$D' = (r_3 * c_3), (i_3 * s_3) = x_3, y_3$

e vengono forniti nello stesso ordine con cui vengono entrati.

VMXM (opcode 1E) Vector Maximum and Minimum.
Latenza=20

Questa determina il vettore Max o Min dei dati forniti in ingresso. (fig.30). L' uscita si presenta nella forma:

$$A'_r = MAX(A, A'_{r-1}); A'_i = MIN(A, A'_{i-1})$$

con:
A =coppia in ingresso attuale
A'_r =nuovo numero massimo sulla porta reale.
A'_i =nuovo numero minimo porta immaginaria.
A'_{r-1} =vecchio numero massimo sulla porta reale.

A'_{i-1} =vecchio numero minimo sulla porta immaginaria.

e fornisce in continuazione la coppia di dati max (posta sul bus reale) e min. (posta sul bus immaginario) secondo il diagramma di timing riportato in fig.31. Dati in ingresso almeno 4 numeri del tipo:

$A = r_0, i_0 \qquad C_0 = ignorat$
$B = r_1, i_1 \qquad C_1 = ignorat$
$C = r_2, i_2 \qquad C_2 = ignorat$
$D = r_3, i_3 \qquad C_3 = ignorat$

le uscite sono:

$A' = MAX\,(r_0, i_0), MIN(r_0, i_0) = x_0, y_0$
$B' = MAX\,(A', r_1, i_1), MIN(A', r_1, i_1) = x_1, y_1$
$C' = MAX\,(B', r_2, i_2), MIN(B', r_2, i_2) = x_2, y_2$
$D' = MAX\,(C', r_3, i_3), MIN(C', r_3, i3) = x_3, y_3$

Le uscite x_i , y_i, come già detto in precedenza, sono fornite sul bus reale ed immaginario rispettivamente.

VSUB (opcode 11) Vector Subtract.
Latenza=18

Sottrae coppie di coefficienti da coppie di dati presenti in ingresso.(fig.32). L' uscita viene fornita nella forma:

A'=A-C

con A', A e C coppie di dati reali. La funzione richiede un numero minimo di dati in ingresso pari a 4, per potere iniziare il processing:

$A = r_0, i_0 \qquad C_0 = c_0, s_0$
$B = r_1, i_1 \qquad C_1 = c_1, s_1$
$C = r_2, i_2 \qquad C_2 = c_2, s_2$
$D = r_3, i_3 \qquad C_3 = c_3, s_3$

I risultati in uscita sono del tipo:

$A' = (r_0 - c_0), (i_0 - s_0) = x_0, y_0$
$B' = (r_1 - c_1), (i_1 - s_1) = x_1, y_1$
$C' = (r_2 - c_2), (i_2 - s_2) = x_2, y_2$
$D' = (r_3 - c_3), (i_3 - s_3) = x_3, y_3$

e sono resi disponibili nell' ordine con cui sono forniti in ingresso.

VNAND (opcode 18) Vector Logical Nand
Latenza=18

Funzione logica che esegue il NAND logico tra i dati in ingresso ed i coefficienti (fig.33). Usando il segnale DCI=1, la funzione restituisce il NAND logico degli stessi. In ambedue i casi, l' ordine dei dati in uscita è lo stesso di quelli di ingresso.

VNOR (opcode 1A) Vector Logical NOR
Latenza=18

Funzione che esegue il NOR logico tra i dati in ingresso ed i coefficienti (fig.34). Anche in questo caso, come nel precedente, posizionando il segnale DCI=1, si ottiene l' OR logico. In ambedue i casi, l' ordine dei dati in uscita è lo stesso di quello di ingresso.

VPAS (opcode 19) Vector Logical Pass.
Latenza=18

Questa semplice funzione, lascia passare le coppie di dati così come si presentano all' ingresso (fig.35).

VXNOR (opcode 1B) Vectro Logical Exclusive NOR.
Latenza=18

Esegue il NOR eslusivo logico tra i dati in ingresso ed i coefficienti (fig 36). Ponendo il livello del segnale DCI=1 la stessa funione fornisce l' OR esclusivo. In ambedue i casi, i dati si presentano in uscita con lo stesso ordine con cui entrano.

MOVC (opcode 1D) Move Coefficient Data.
Latenza=18

Questa funzione è molto utile per spostare i coefficienti dalla porta C, a qualsiasi porta scelta tra le A,B,Q (fig.37). Può essere usata per la verifica dei coefficienti prima dell' esecuzione di un qualsiasi algoritmo, se necessario.

Durante l' esecuzione della funzione, eventuali dati alle porte A,B,Q vengono ignorati.

A'=C

con C complesso. I coefficienti spostati dalla porta C a quella di destinazione, sono disponibili su quest' ultima nello stesso ordine con cui sono stati mossi. La funzione richiede almeno 4 punti per potere iniziare il processing:

$A = ignorati \qquad C_0 = c_0, s_0$
$B = ignorati \qquad C_1 = c_1, s_1$
$C = ignorati \qquad C_2 = c_2, s_2$
$D = ignorati \qquad C_3 = c_3, s_3$

in uscita troveremo:

$A' = c_0, s_0 = x_0, y_0$
$B' = c_1, s_1 = x_1, y_1$
$C' = c_2, s_2 = x_2, y_2$
$D' = c_3, s_3 = x_3, y_3$

MOVD (opcode 1C) Move Data.
Latenza=8 con RQWC e RAWC
Latenza=18 in tutti gli altri casi.

Questa funzione sposta dati da qualsiasi porta (A,B,C;Q) a qualsiasi porta (A,B,C;Q) come da fig.~~38~~ 37a. In questo particolare caso si ha:

A=A'

con A e A' sono complessi. Questa è una funzione fondamentale per caricare dati e coefficenti. Questi ultimi, di norma, entrano nel relativo buffer collegato alla porta C secondo il percorso *Bus VME --> porta B --> porta C*. Richiede che siano disponibili almeno quattro dati da spostare:

$A = r_0, i_0 \qquad C_0 = ignorat$
$B = r_1, i_1 \qquad C_1 = ignorat$
$C = r_2, i_2 \qquad C_2 = ignorat$
$D = r_3, i_3 \qquad C_3 = ignorat$

I dati nella porta di destinazione, vengono presentati nello stesso ordine con cui sono stati prelevati dalla porta di provenienza.

$A' = (r_0), (i_0) = x_0, y_0$
$B' = (r_1), (i_1) = x_1, y_1$
$C' = (r_2), (i_2) = x_2, y_2$
$D' = (r_3), (i_3) = x_3, y_3$

La stessa funzione è anche usata per introdurre i dati dalla porta Q.

BFLY2

Opcode: 02

Radix-2 Butterflies

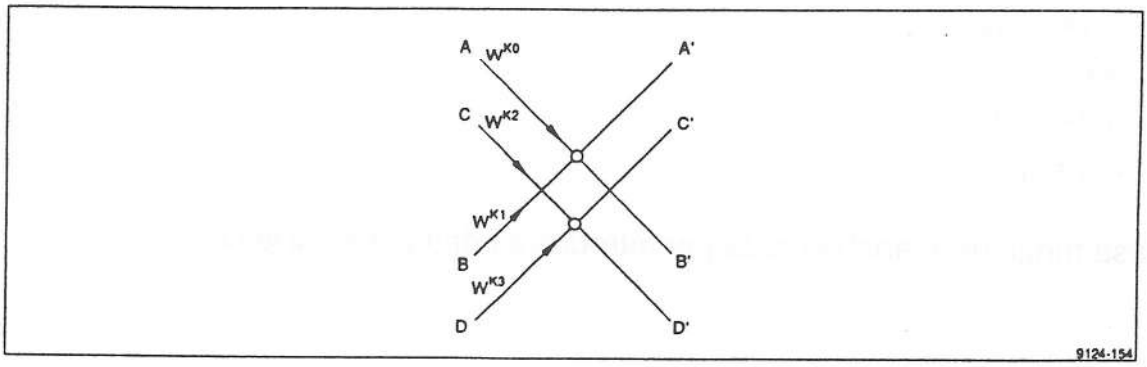


Figure 11 Radix-2 Butterflies

BFLY4

Opcode: 01

Radix-4 Butterfly

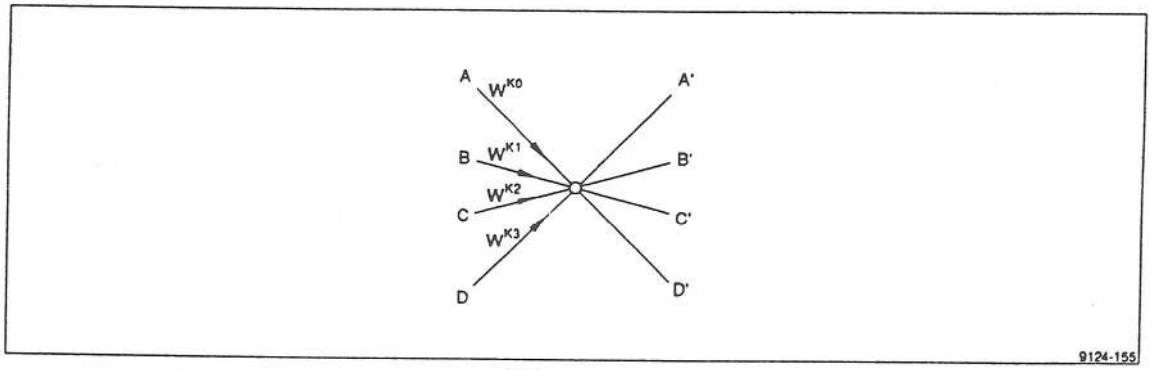


Figure 12 Radix-4 Butterfly

BFLY16

Opcode: 00

Radix-16 Butterfly

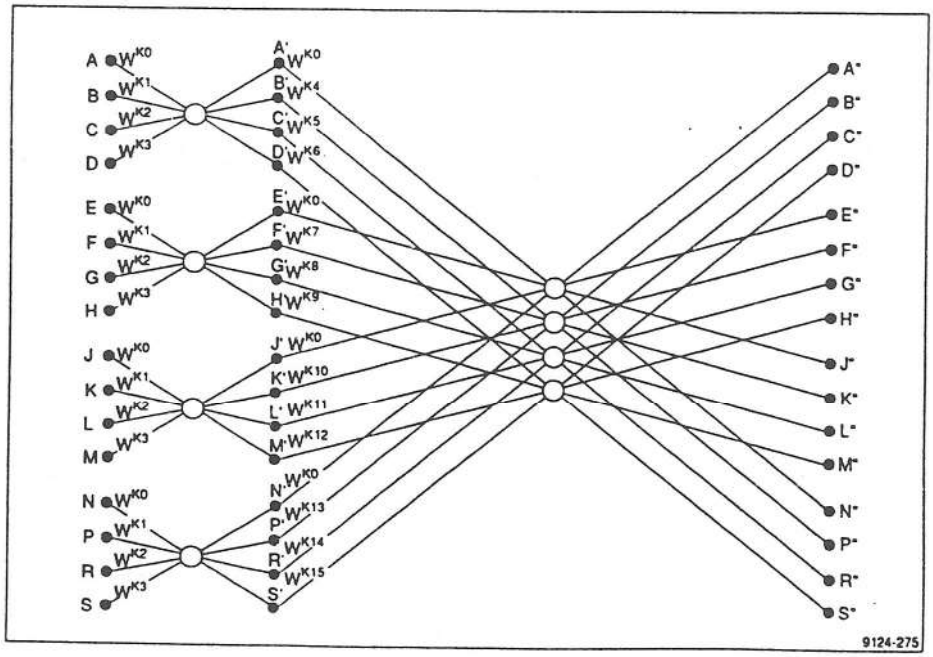


Figure 13 Radix-16 Butterfly

BWND2

Radix-2 Complex Window Butterflies

Opcode: 05

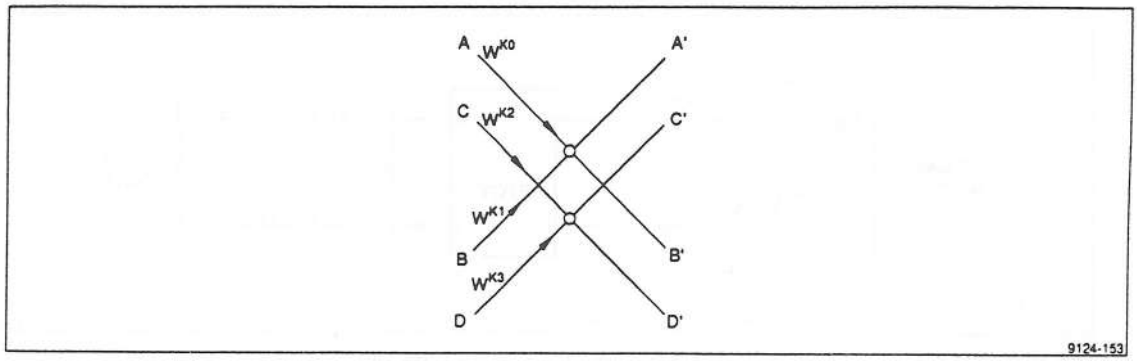


Figure 14 Radix-2 Complex Window Butterflies

BWND4

Radix-4 Complex Window Butterfly

Opcode: 04

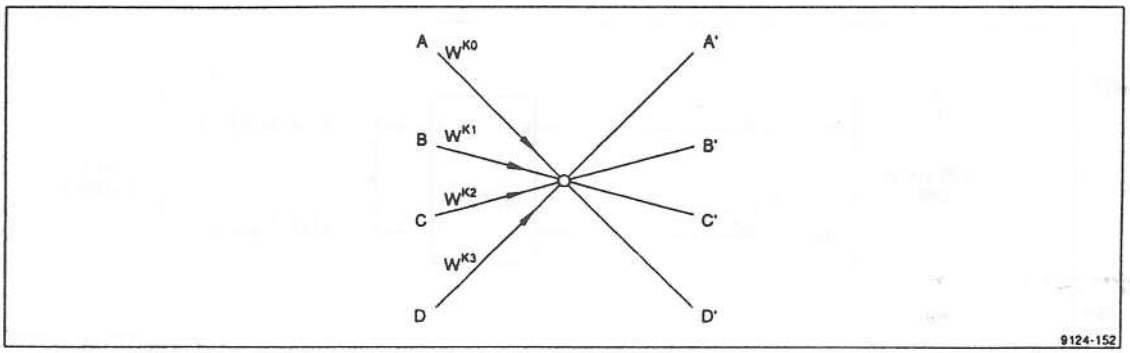


Figure 15 Radix-4 Complex Window Butterfly

BRFT

Dual Real FFT Separation

Opcode: 07

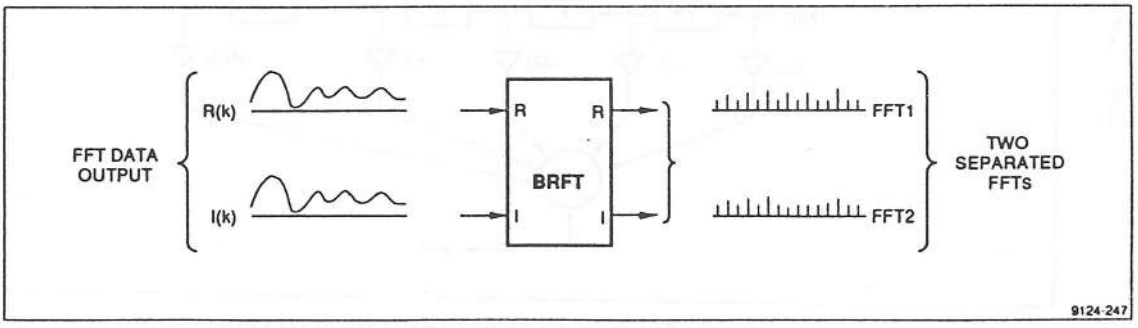


Figure 16 Dual Real FFT Separation

BFCT

Fast Cosine Transform/Double Length (N Output)

Opcode: 06

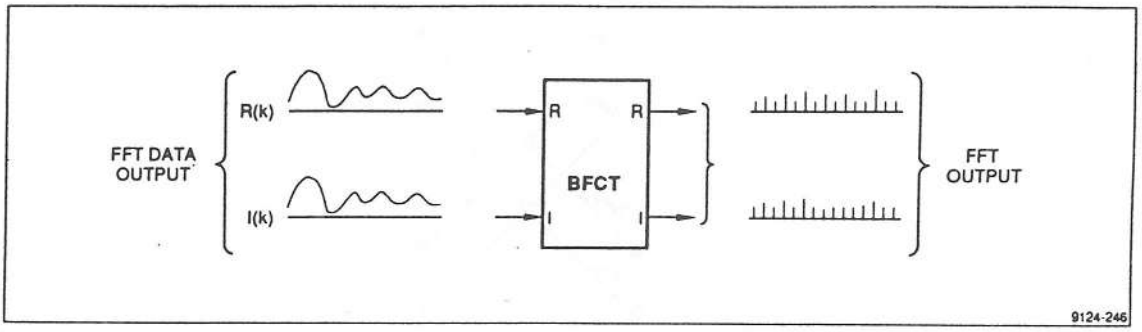


Figure 17 Fast Cosine Transform/Double Length (N Output)

BFCT2

Fast Cosine Transform/Double Length (2N Output)

Opcode: 0E

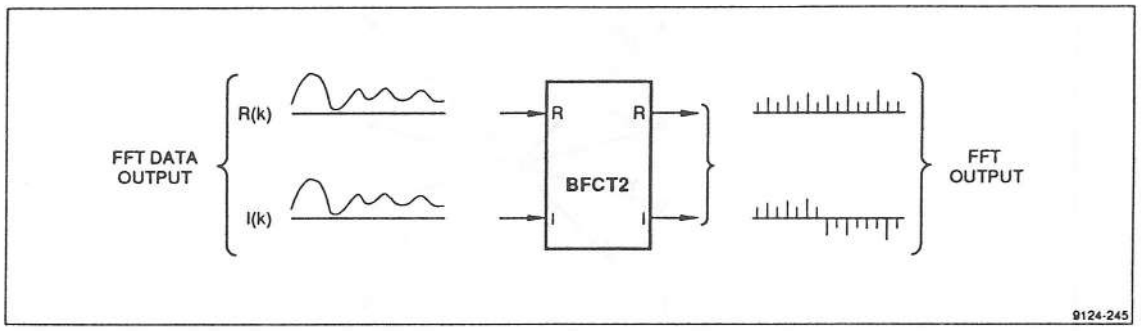


Figure 18 Fast Cosine Transform/Double Length (2N Output)

BCFIR

Complex Finite Impulse Response Filter

Opcode: 08

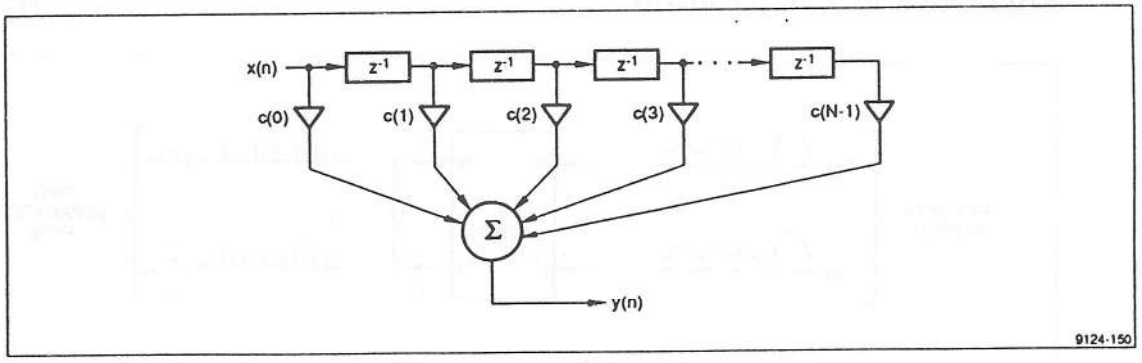
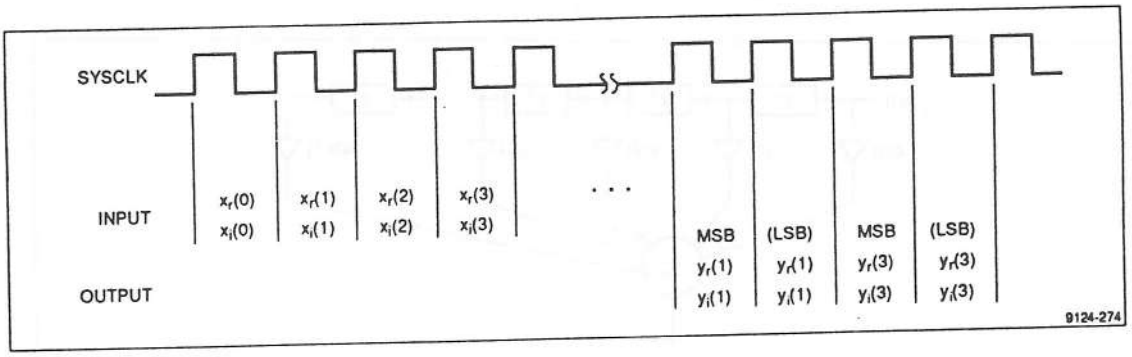


Figure 19 Complex Finite Impulse Response Filter



BCFIR Timing Outputs

BDFIR

Opcode: 09

Double Real Finite Impulse Response Filter

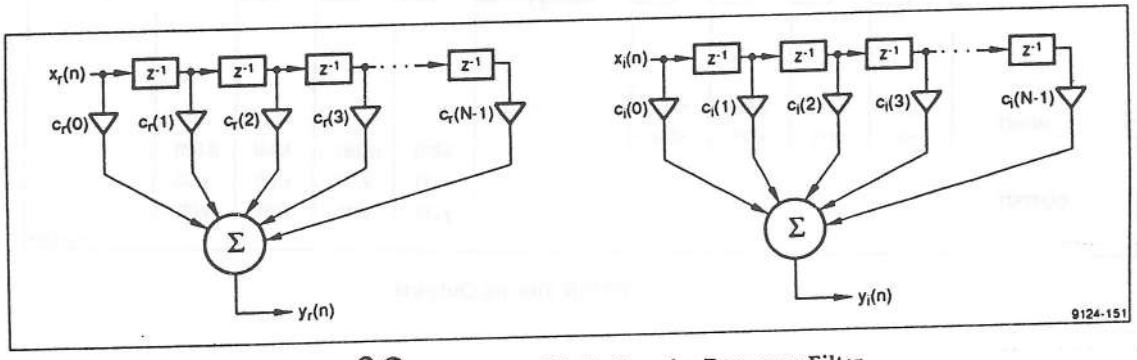
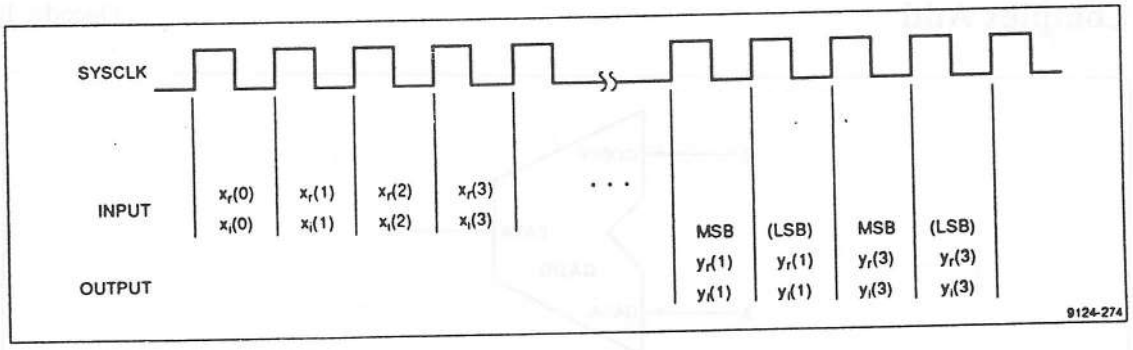


Figure 20 Double Real Finite Impulse Response Filter



BDFIR Timing Outputs

BRFIR

Real Finite Impulse Response Filter

Opcode: OA

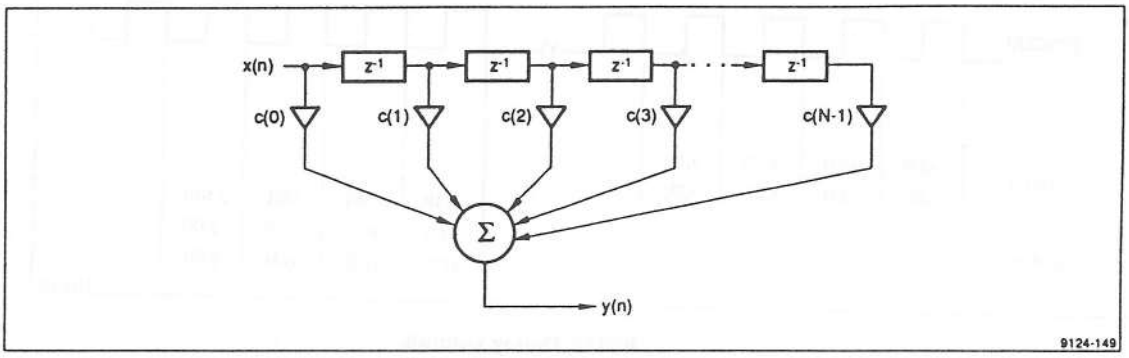
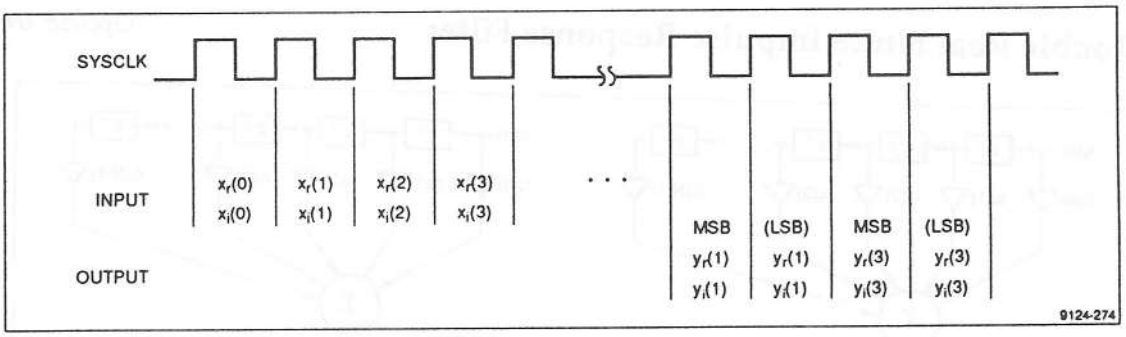


Figure 21 Real Finite Impulse Response Filter



BRFIR Timing Outputs

CADD

Complex Add

Opcode: 10

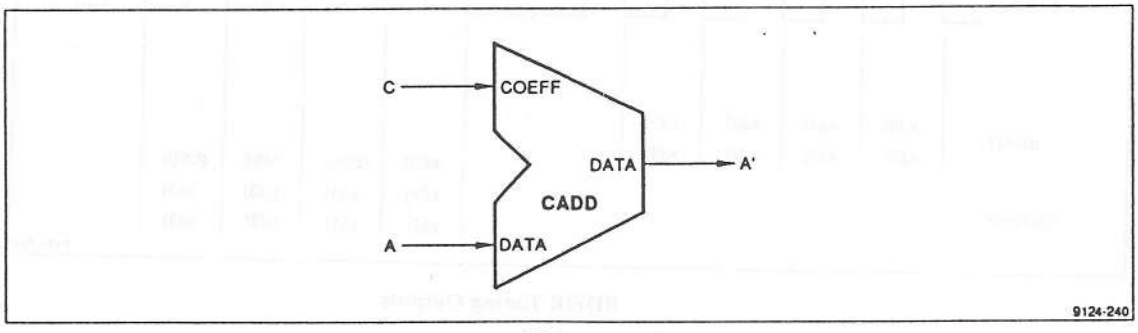


Figure 22 Complex Add

CMAG

Opcode: 0C

Magnitude Squared

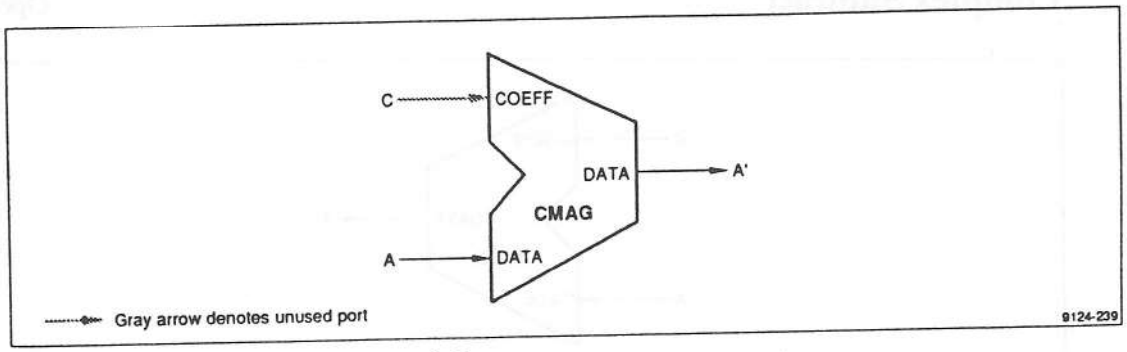


Figure 23 Complex Magnitude Squared

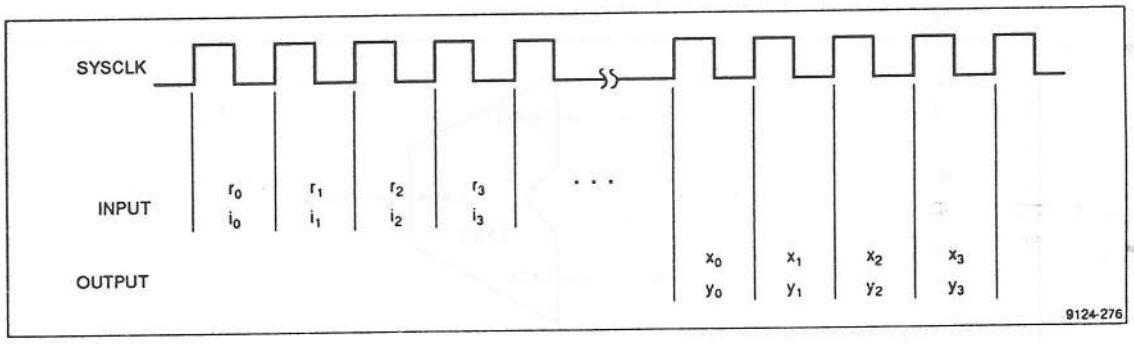


Figure 24 CMAG Timing Outputs

CMUL

Opcode: 0D

Complex Multiply

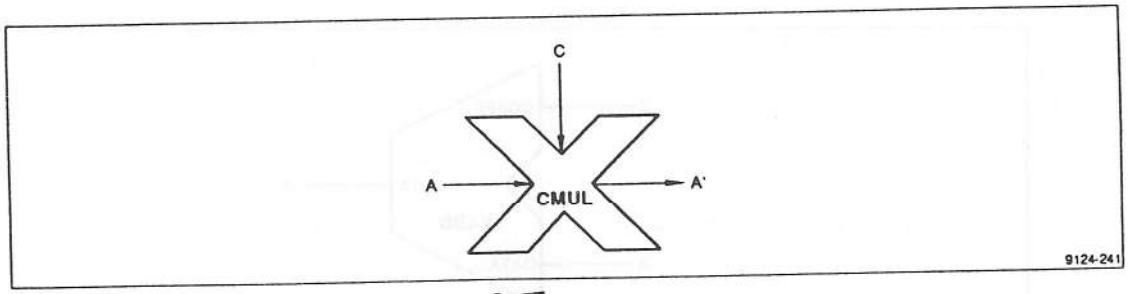


Figure 25 Complex Multiply

CSUB

Complex Subtract

Opcode: 11

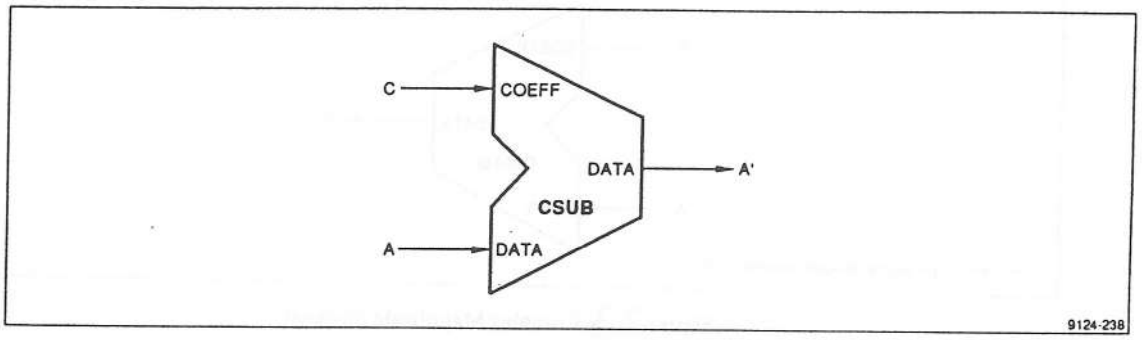


Figure 26 Complex Subtract

VABS

Opcode: 13

Vector Absolute Value

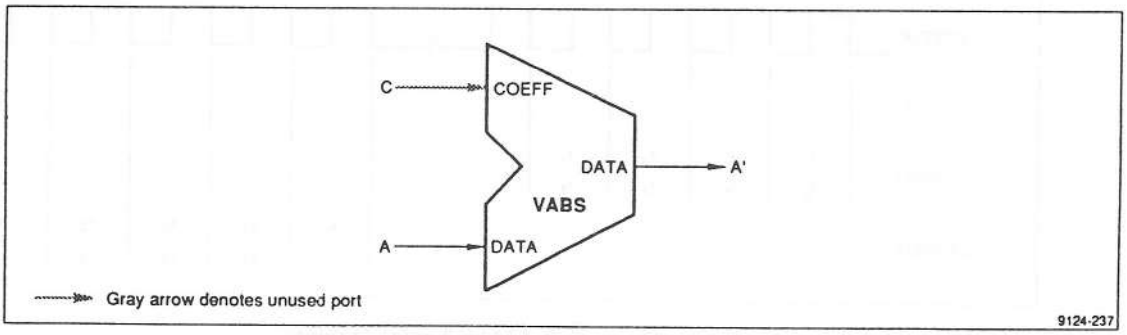


Figure 27 Vector Absolute Value

VADD

Vector Add

Opcode: 10

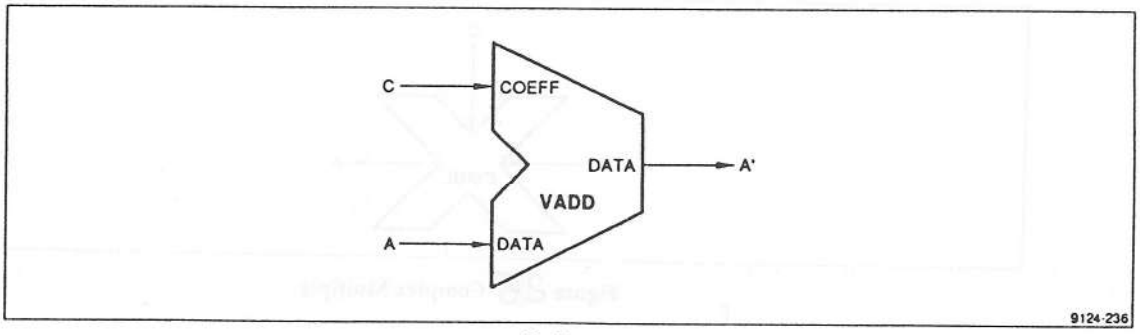


Figure 28 Vector Add

VMUL

Opcode: 12

Vector Multiply

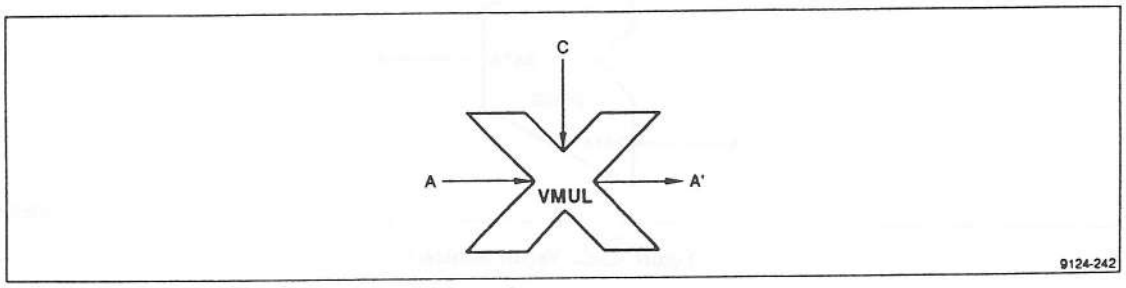


Figure 29 Vector Multiply

VMXM

Vector Maximum and Minimum

Opcode: 1E

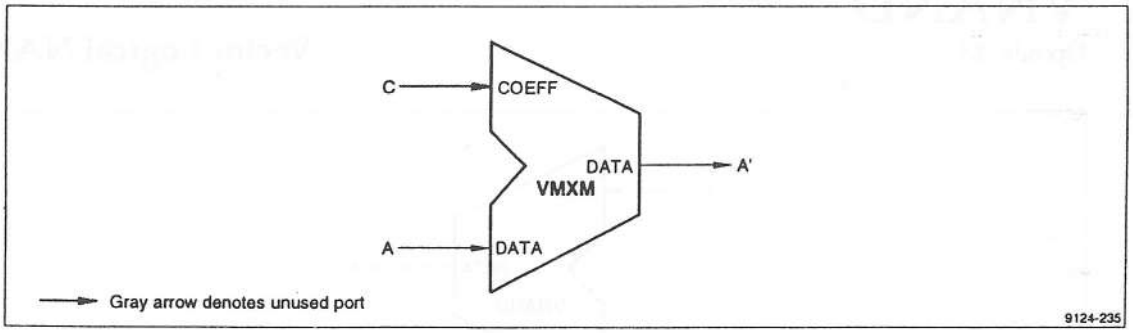


Figure 30 Vector Maximum and Minimum

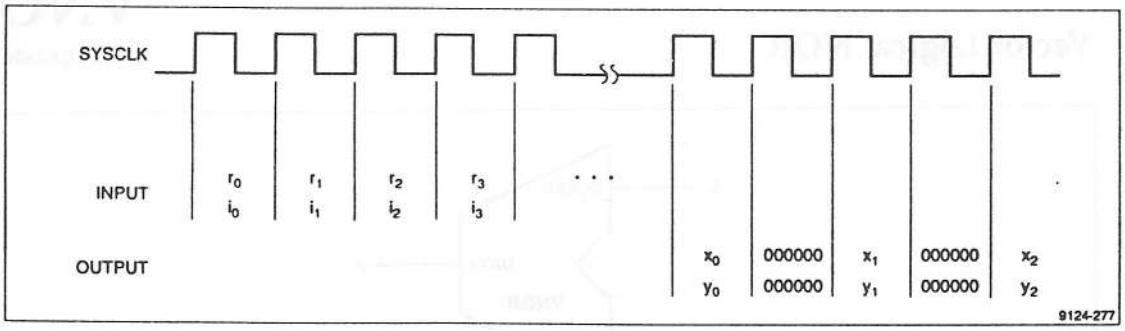


Figure 31 VMXM Timing Outputs

VSUB

Vector Subtract

Opcode: 11

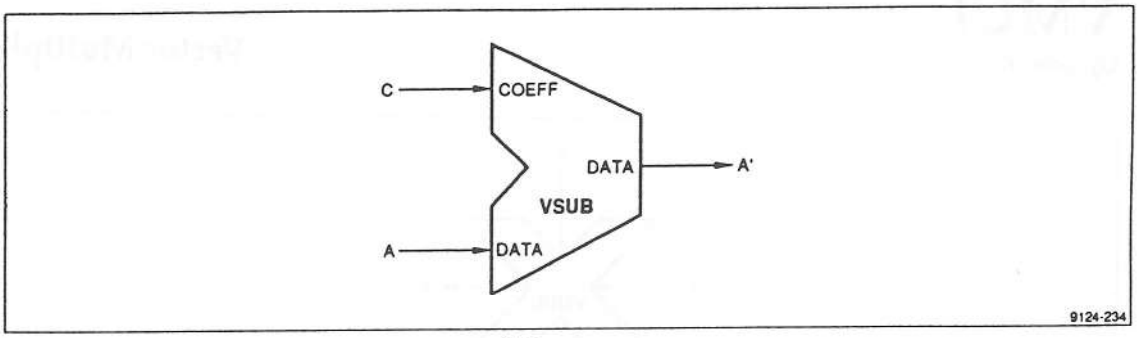


Figure 32 Vector Subtract

VNAND

Opcode: 18

Vector Logical NAND

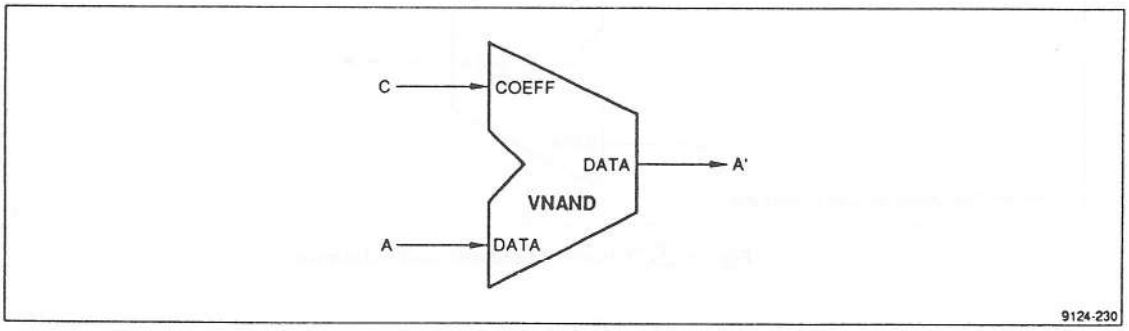


Figure 33 Vector Logical NAND

VNOR

Vector Logical NOR

Opcode: 1A

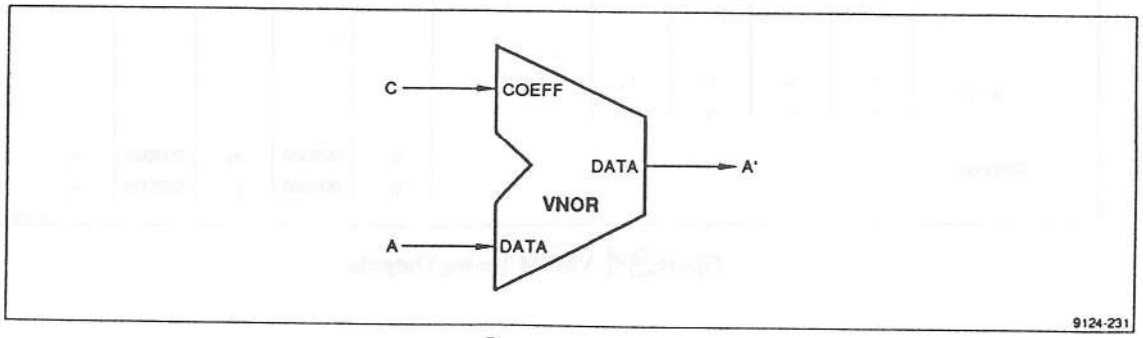


Figure 34 Vector Logical NOR

VPAS

Opcode: 19

Vector Logical Pass

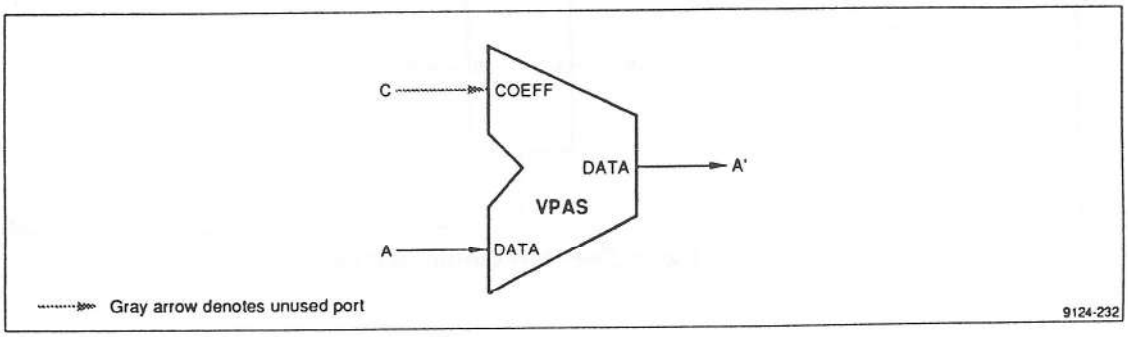


Figure 35 Vector Logical Pass

VXNOR

Opcode: 1B

Vector Logical Exclusive NOR

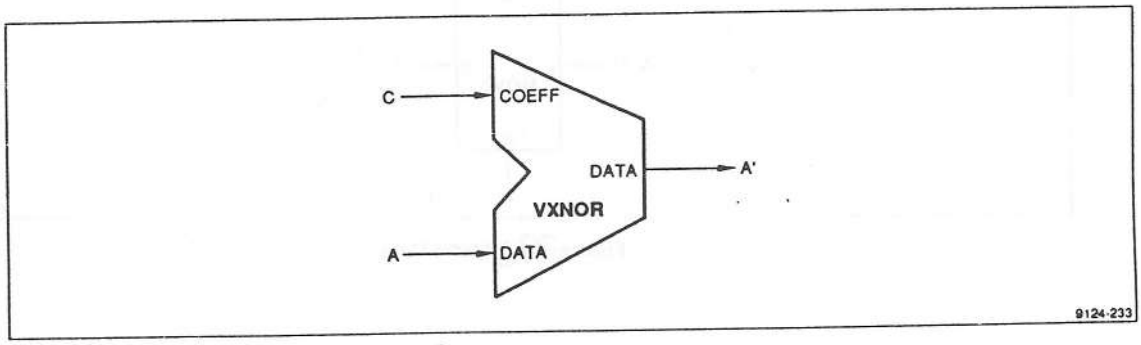


Figure 36 Vector Logical Exclusive NOR

MOVC

Opcode: 1D

Move Coefficient Data

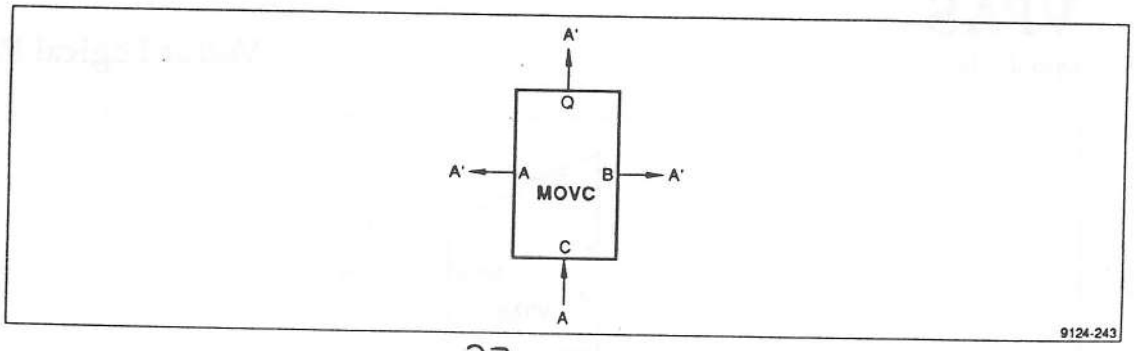


Figure 37 Move Coefficient Data

MOVD

Opcode: 1C

Move Data

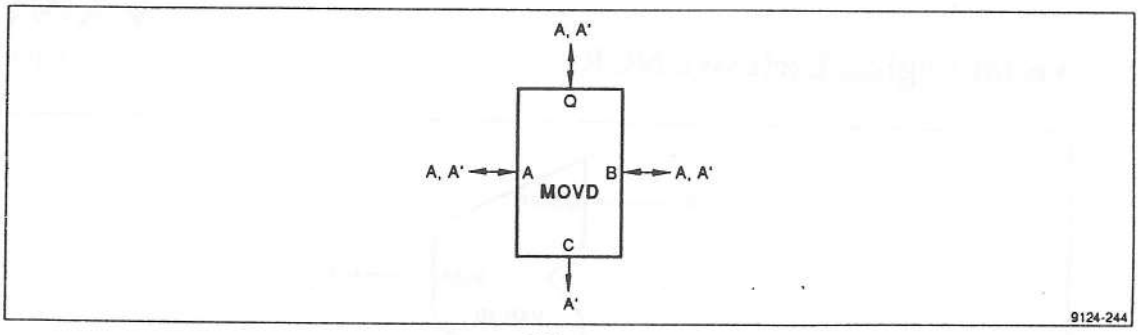


Figure 37a Move Data

(file: shar2.doc)

CAP. 3

FFT DI DATI REALI.

Già nella prima parte di questa nota tecnica si era introdotto un algoritmo per il calcolo della FFT di sequenze di dati reali. Dopo avere introdotto la serie di funzioni esistenti a bordo del DSP, affrontiamo la discussione di una serie di algoritmi, detti a *Separazione di Passo*, che permettono al chip set il calcolo di FFT di dati reali ad elevata velocità. Questo in considerazione anche del fatto che il calcolo della FFT di dati reali ricopre un ruolo di notevole importanza nel processing di dati radioastronomici, settore questo, in cui il sistema si troverà ad operare. Questo algoritmo incrementa la velocità di elaborazione processando i dati reali formattati, prima di mandare in esecuzione la serie di passi BFLYx o BFLYxx, come se fossero immaginari e separando poi i risultati in due separate FFT tramite la funzione BRFT. Il sistema LH9124/LH9320 può implementare due diversi tipi di algoritmi a *Separazione di Passo:*

a) Separazione di passo di dati reali "due alla volta"
b) Separazionedi passo di dati reali di lunghezza 2N.

Analizziamoli separatamente per potere poi in seguito scegliere di volta in volta l' algoritmo più adatto alla applicazione in corso.

a) Separazione di Passo: "solo dati reali -due alla volta"-"

L' algoritmo a separazione di passo detto "due alla volta", trae il suo nome dal fatto che i due frame di dati reali vengono elaborati contemporaneamente uno nella parte reale della memoria e l' altro nella parte immaginaria, come se l' intero frame comprendesse un unico numero complesso. L' algoritmo viene rappresentato in fig.38 da cui si deduce che per il calcolo della FFT di una serie di dati reali, occorre eseguire gli step che seguono:

1) Memorizzare, inizialmente, la serie dei 2N punti (letti tramite la porta Q) nel buffer scelto, che chiameremo Buffer1. Dati due array di dati reali in ingresso h(n) e g(n), possono essere processati in parallelo come se fossero, in realta', un solo array complesso x(n) di dimensioni N e quindi x(n)=h(n)+jg(n)

2) A questo punto si esegue il computo della FFT sui dati dell' array x(n). Il computo della DFT di x(n) potra' essere espresso come DFT di h(n) e g(n) nel modo seguente:

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{nk} = \sum_{n=0}^{N-1} h(n) W_N^{nk} + j \sum_{n=0}^{N-1} g(n) W_N^{nk} \\ &= H(k) + jG(k) = [H_r(k) + jH_i(k)] + j[G_r + jG_i(k)] \\ &= [H_r(k) - G_i(K)] + j[H_i(k) + G_r(k)] \\ &= R(k) + jI(k) = [R_e(k) + R_o(k)] + j[I_e(k) + I_i(k)] \end{aligned} \qquad (3.1)$$

dove R(k) e I(k) rappresentano rispettivamente la parte reale ed immaginaria della X(k), mentre i pedici "e" ed "o"denotano la parte pari (Even) e dispari (Odd). Per effetto delle proprietà di simmetria della FFT, solo i punti reali di ingresso producono in uscita punti reali "pari" e punti immaginari "dispari" come segue:

$$H(k) = R_e(k) + jI_0(k)$$

ed, analogamente, solo punti di ingresso immaginari producono in uscita reali "dispari" ed immaginari "pari" come segue:

$$jG(k) = R_o(k) + jI_e(k)$$
$$G(k) = I_e(k) - jR_o(k)$$

Si può risalire alla X(k), come combinazione lineare di H(k) e G(k)

3) Si memorizzano, a questo punto, gli N punti della risposta complessa X(k) nel buffer 2 iniziando dall' indirizzo zero.

4) Si può ora effettuare la ricombinazione dei dati così ottenuti. Consideriamo, a tale proposito, che campionamenti in frequenza H(k) e G(k) possono essere rappresentati in termini di campionamenti frequenziali di X(k) nei punti k ed N-k come segue:

$$\begin{aligned} H(k) &= R_e(k) + jI_0(k) \\ &= \frac{1}{2}(\mathrm{Re}[X(k) + X(N-k)] + j\,\mathrm{Im}[X(k) - X(N-k)]) \end{aligned}$$

e

$$\begin{aligned} G(k) &= I_e(k) - jR_0(k) \\ &= \frac{1}{2}(\mathrm{Im}[X(k) + X(N-k)] - j\,\mathrm{Re}[X(k) + X(N-k)]) \end{aligned}$$

dove le notazioni Re[X(k)] e Im[X(k)] rappresentano, rispettivamente, la parte reale e complessa della FFT complessa X(k).
Il processo di ricombinazione dei dati, si ottiene in questo caso usando la funzione BRFT che, con in ingresso uno spettro complesso fornisce in uscita due FFT separate.
5) Dopo l' avere applicato la funzione BRFT, si memorizzano le due sequenze complesse (da N punti) H(k) e G(k) nel buffer 3.

Indirizzamento.

L' address generator LH 9320, è in grado di fornire il pattern di indirizzi BRFTL, BRFTLS e BRFTU, BRFTLS, per la generazione della sequenza di indirizzamento richiesta dalle due possibili implementazioni di questo algoritmo. Le BRFTLe BRFTLS generano i pattern di indirizzamento dei dati in ingresso (fig.39 e 40) per la separazione delle due FFT reali di N punti, durante il processo di ricombinazione che compete alla funzione BFRT. Ambedue assumono che prima del computo della FFT del passo precedente, la prima sequenza sia stata memorizzata nella memoria reale e la seconda in quella immaginaria. Come visibile in fig. 39, l' address pattern BRFTL genera la sequenza degli indirizzi per la cosìddetta *full length computation* (2N punti), per contro il BRFTLS genera la sequenza di indirizzi per la *half length computation* (primi N punti). In uscita, le BRFTU e BRFTUS address patterns, generano l' indirizzamento dell' output, durante il processo di ricombinazione. Come si può vedere dalle figure 38 e 39 gli address patterns BRFTLx (input) ed i relativi BRFTUx (output) sono accoppiati nel seguente modo:

BRFTL---------->BRFTU
BRFTLS---------->BRFTUS

Nel caso *full length computation* (2N punti) ogni sequenza complessa in uscita, contiene N punti. H(k) inizia all' indirizzo zero del buffer e G(k) inizia dall' indirizzo N per cui i dati in ingresso all' indirizzo "i" e "N+i" generano due sequenze di dati in uscita memorizzati in "i" e "N-i". Nel caso della *half length computation* (primi N punti) dove si usa la BRFTUS, ci si trova nella condizione in cui ogni sequenza complessa in uscita contiene solo i primi N/2. In questo caso la H(k) inizia all' indirizzo zero e la G(k) all' indirizzo N/2. Analogamente a quanto visto sopra, i dati di ingresso agli indirizzi "i" e "N-i" generano due sequenze di dati in uscita che sono memorizzate alle locazioni di indirizzo "i" e "N+i". In altri termini, sono richiesti solo i primi N/2 punti per definire la DFT dei punti reali in ingresso.

Separazione di Passo "Solo dati reali, doppia lunghezza -2N dati-" (Fast Cosine Transform)

Questo secondo algoritmo a separazione di passo, esegue la FFT di una sequenza di dati in ingresso di dimensioni 2N, trattandola come una sequenza complessa di dimensione N (fast cosine transform) per cui la velocita` di esecuzione della FFT di 2N dati reali e' la stessa di quella di N dati complessi. La filosofia che sta alla base di questo algoritmo e' del tutto simile a quella precedente del "due alla volta. In questo secondo caso la sequenza dei 2N dati reali in ingresso e`espressa da:

$$X_k(k) = \sum_{n=0}^{2N-1} x(n) W_{2N}^{k}$$
$$= \sum_{2=0}^{N-1} x(2n) W_N^{nk} + W_{2N}^{k} \sum_{n=0}^{N-1} x(2n+1) W_N^{nk} \qquad (3.2)$$

Ponendo h(n)=x(2n) e g(n)=x(2n+1) la (3.2) si puo' riscrivere nella forma:

$$X(k) = \sum_{n=0}^{N-1} h(n) W_N^{nk} + W_{2N}^{k} \sum_{n=0}^{N-1} g(n) W_N^{nk}$$
$$= H(k) + W_{2N}^{k} G(k)$$

Usando la stessa tecnica usata nel caso "due alla volta" per il calcolo di H(k) e G(k) si ottiene:

$$y(n) = h(n) + jg(n$$
$$Y(k) = H(k) + jG(k)$$
$$Y(k) = [H_r(k) - G_i(k)] + j[H_i(k) + jG_r(k)]$$
$$Y(k) = R(k) + jI(k) = [R_e(k) + R_o(k)] + j[I_e(k) + I_o(k)]$$
$$\mathrm{H(k)} = \mathrm{R_e}\,(k) + jI_o(k)$$
$$G(k) = I_e(k) - jR_o(k)$$

Si ricorda che in questo caso il numero di coefficienti (twiddle) sono 2N perche' nella fast cosine transform i coefficienti sono tutti reali. Come abbiamo visto in precedenza, il processo di ricombinazione viene effettuato mediante la funzioni BFCT (N punti) o BFCT2 (2N punti) che forniscono, come parte reale di X(k) :

$$X_r = \mathrm{Re}[R_e(k) + jI_o(k) + W_{2N}^k (I_e(k) - jR_o(k)]$$

$$X_r(k) = \frac{1}{2}\mathrm{Re}[Y(k) + Y(N-k)]$$
$$+\frac{1}{2}\cos(\pi k / N)\mathrm{Im}[Y(k) + Y(k-N)]$$
$$-\frac{1}{2}sin(\pi k / N)\mathrm{Re}[Y(k) - Y(k-N)]$$

e come parte immaginaria di X(k):

$$X_i = \mathrm{Im}[R_e(k) + jI_o(k) + W_{2N}^k (I_e(k) - jR_o(k)]$$

$$X_i(k) = \frac{1}{2}\mathrm{Im}[Y(k) - Y(N-k)]$$
$$+\frac{1}{2}sin(\pi k / N)\mathrm{Im}[Y(k) + Y(k-N)]$$
$$-\frac{1}{2}\cos(\pi k / N)\mathrm{Re}[Y(k) - Y(k-N)]$$

L' uscita di questo algoritmo e` costituita da una sequenza di dati di lunghezza 2N invece che di due sequenze di dati di lunghezza N. In funzione delle proprieta` di simmetria della DFT, nella serie di 2N dati quelli reali sono simmetrici mentre gli immaginari sono asimmetrici rispetto al punto centrale N che non puo` essere ottenuto dai processi di ricombinazione BFCT e BFCT2. Il punto N rappresenta il campionamento in frequenza piu` alto di tutta la serie 2N, contenuto all'indirizzo N. Il dato in questo punto e`un dato reale e puo` essere ottenuto come X(0)=H(0)-G(0).

Indirizzamento.

Come nel caso precedente, l' address generator LH9320 deve essere in grado di indirizzare i dati in memoria direttamente in modo tale da implementare l' algoritmo. Infatti l' LH9320 e` in grado di provvedere per i pattern di indirizzamento BFCTL e BFCTT, e BFCTU/P/EP/LP, BFCTUS. Come si puo`vedere nelle fig.41 (Full lenght computation) e 42 (Half length computation), BFCTL genera (carica) una sequenza di indirizzamento tale da estrarre i 2N punti come un unico set di N numeri complessi. BFCTT address pattern genera la sequenza di indirizzamento dei twiddle factors, per il parametro W_{2N}^k, usando un pattern lungo 2N invece di N. Per quello che riguarda i dati in uscita, BFCTU e BFCTUS generano le relative sequenze di indirizzamento. In particolare, BFCTU genera la corretta sequenza di indirizzamento (per l'algoritmo Full Leght Computation: 2N) per i dati in uscita durante il processo di ricombinazione sulla FFT complessa proveniente dal passo precedente. BFCTUS, invece, genera la corretta sequenza di indirizzamento per i dati in uscita durante il processo

di ricombinazione, per quello che riguarda l' algoritmo Half Length Computation. BFCTUS genera una sequenza di indirizzamento che, in pratica, e` un conteggio diretto da 0 a N-1 ripetendo ogni indirizzamento due volte. In questo caso i primi N punti forniscono dati sufficienti per descrivere l'intera sequenza di 2N punti, a parte il punto N.

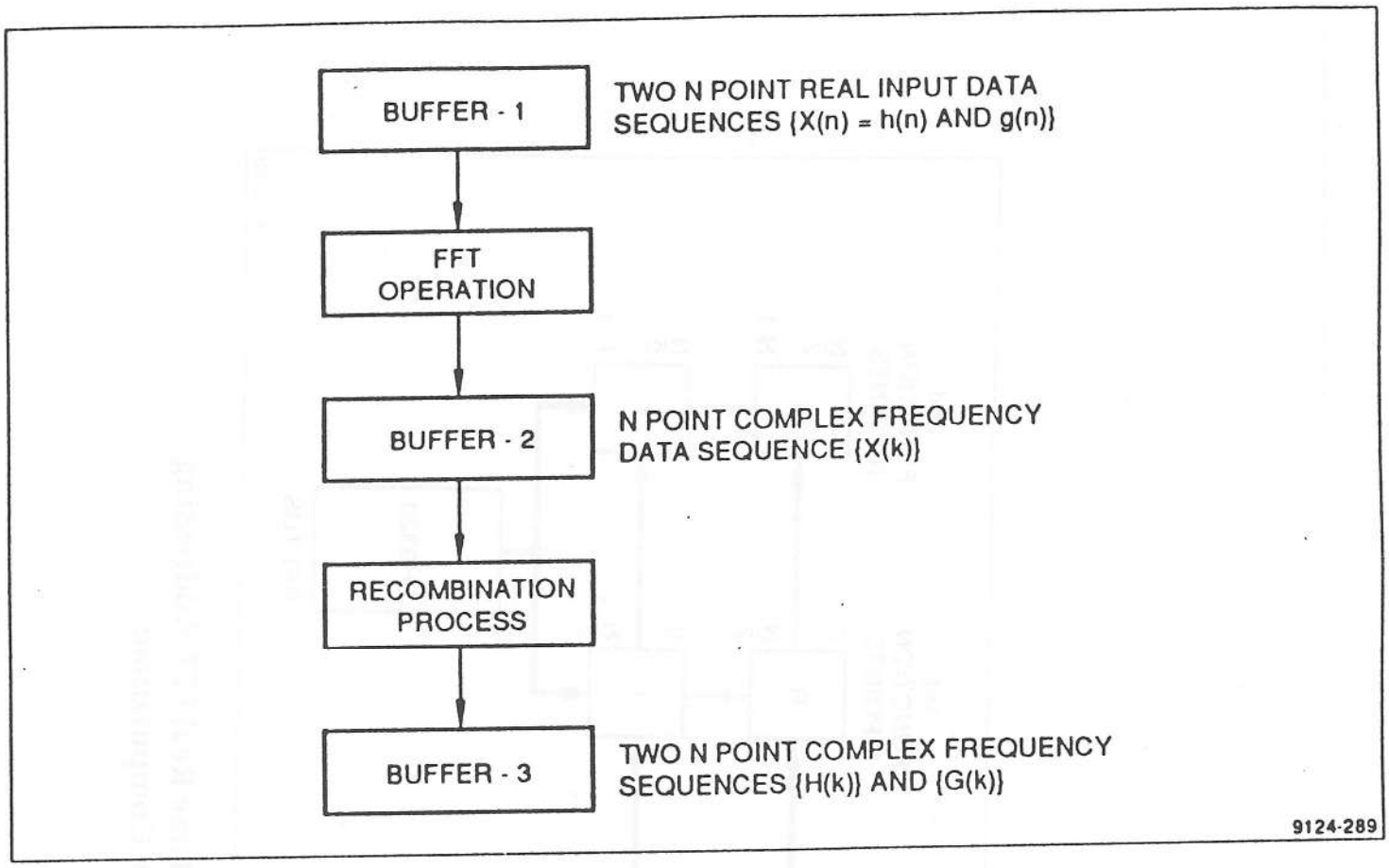


Figure 38. Two-at-a-Time Real FFT Addressing

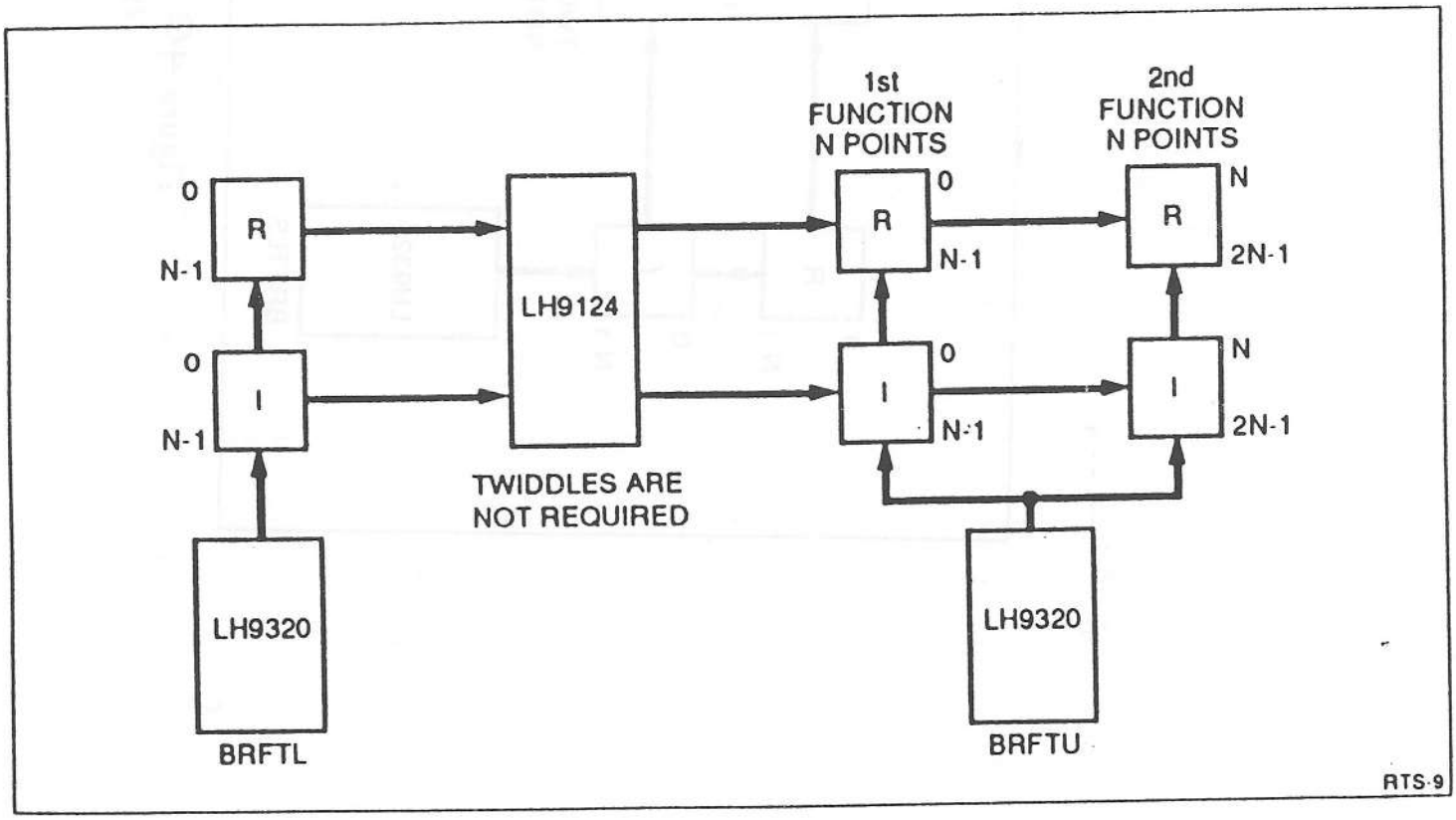


Figure 39 Two-at-a-Time Real FFT Addressing
(Full Length Computation)

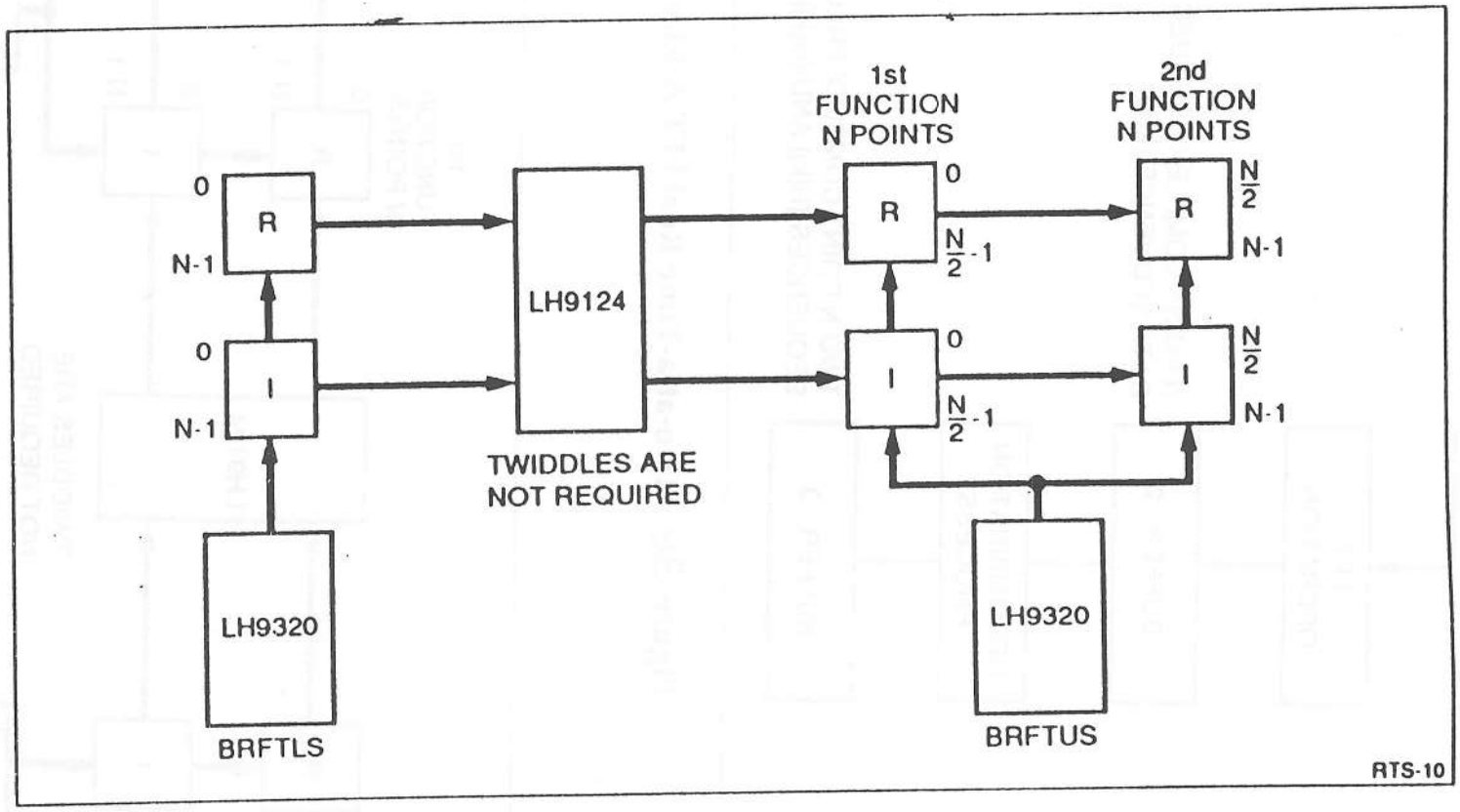


Figure 40 Two-at-a-Time Real FFT Addressing (Half Length Computation)

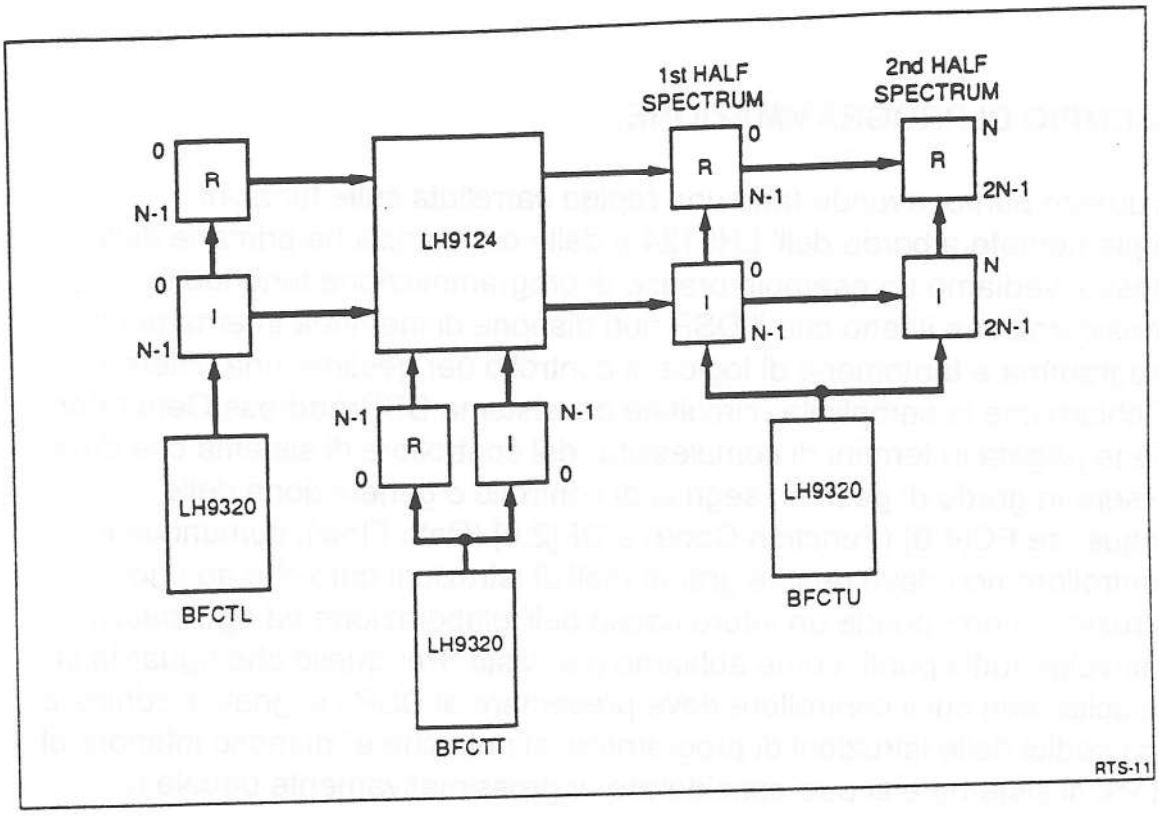


Figure 41 Double Length (2N) Real FFT Addressing
(Full Length Computation)

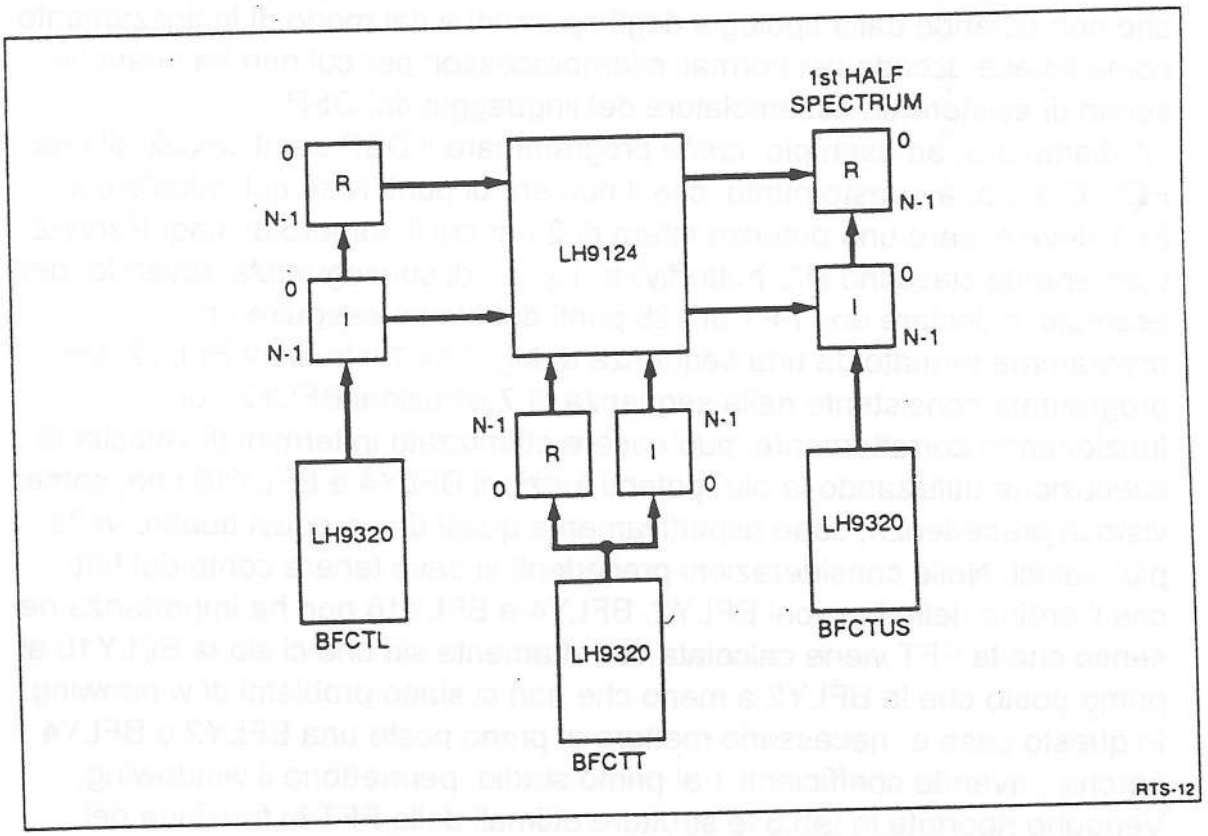


Figure 42 Double Length (2N) Real FFT Addressing
(Half Length Computation)

ESEMPIO DI PROGRAMMAZIONE.

A questo punto, avendo fatto una rapida carrellata sulle funzioni implementate a bordo dell' LH9124 e delle caratteristiche primarie dello stesso, vediamo un esempio pratico di programmazione tenendo in considerazione il fatto che il DSP non dispone di memoria interna per il programma e tantomeno di logica di controllo per gestirne una esterna. E`chiaro che la semplicita` circuitale del sistema DSP/Address Generator, viene pagata in termini di complessita` del controllore di sistema che deve essere in grado di gestire i segnali di controllo e generazione delle sequenze FC[4:0] (Function Code) e DF[2:0] (Data Flow); comunque il controllore non deve gestire grandi moli di istruzioni dato che ad ogni istruzione corrisponde un intero passo dell' elaborazione ed ogni passo coinvolge tutti i punti, come abbiamo gia` visto. Per quello che riguarda la velocita` con cui il controllore deve presentare al DSP i segnali di controllo ed i codici delle istruzioni di programma, si nota che e` di molto inferiore al clock di sistema e si puo`considerare approssimativamente uguale a $\frac{f_{SYSCLK}}{N}$, dove N e`il numero di punti da elaborare. Inoltre, queste macroistruzioni sono in numero limitato ed hanno un codice operativo fisso che non dipende dalla tipologia degli operandi e dal modo di indirizzamento come invece accade nei normali microprocessori per cui non ha neanche senso di esistere un assemblatore del linguaggio del DSP.

Vediamo ora, ad esempio, come programmare il DSP per il calcolo di una FFT. E` noto, a questo punto, che il numero di punti N su cui calcolare la FFT deve essere una potenza intera di 2 per cui il numero di stadi Radix-2 (contenente ciascuno N/2 butterfly) e` $\log_2 N$; di conseguenza dovendo, per esempio, calcolare una FFT di 128 punti dobbiamo eseguire un programma formato da una sequenza di $\log_2 128 = 7$ istruzioni BFLY2. Un programma consistente nella sequenza di 7 istruzioni BFLY2 pur funzionando correttamente, puo`essere ottimizzato in termini di velocita`di esecuzione utilizzando le piu` potenti funzioni BFLY4 e BFLY16 che, come visto in precedenza, sono rispettivamente quasi due e quasi quattro volte piu` veloci. Nelle considerazioni precedenti si deve tenere conto del fatto che l' ordine delle funzioni BFLY2, BFLY4 e BFLY16 non ha importanza nel senso che la FFT viene calcolata correttamente sia che ci sia la BFLY16 al primo posto che la BFLY2 a meno che non ci siano problemi di windowing. In questo caso e` necessario mettere al primo posto una BFLY2 o BFLY4 perche`, avendo coefficienti 1 al primo stadio, permettono il windowing. Vengono riportate in tab.3 le strutture ottimali della FFT in funzione dei punti da trasformare. Da notare come per il calcolo di una FFT di 16 punti venga consigliato un doppio radix-4 invece che un unico passo radix-16. Questa costituisce una unica eccezione per il fatto che una radix-16 ha una latenza di 68 cicli che la rende piu` lenta di due radix-4 (36 cicli in tutto).

CONSIDERAZIONI SULLE PRESTAZIONI TEORICHE.

A questo punto e` interessante tentare di fare una stima dei tempi richiesti per il computo di una FFT su set di dati composti da diversi numeri di punti. Il calcolo del tempo di esecuzione risulta essere relativamente semplice dato che ogni passo ha, latenza a parte, durata pari al tempo richiesto per la lettura dei dati dal buffer su cui sono immagazzinati, attraverso la relativa porta (di solito la Q). Un esempio pratico: supponiamo di dovere effettuare una FFT su un set di 1024 punti (1K) complessi. Dalla tab. ~~3~~ 3a si evince che la sequenza dei passi richiesta per un computo ottimizzato risulta essere:

BFLY4
BFLY16
BFLY16

La prima istruzione viene eseguita in un tempo che, in termini di cicli di clock, e` formato da 1024 cicli per la lettura dei 1024 punti dalla porta Q piu` i 18 cicli di latenza richiesti dal passo stesso e la scrittura nella memoria temporanea attraverso la relativa porta, ad esempio la A. La seconda istruzione porta via, sempre in termini di cicli di clock, i soliti 1024 cicli di clock per la lettura del set memorizzato alla fine del passo precedente nel buffer della porta A. Ai suddetti 1024 cicli di clock bisogna aggiungere i 68 di latenza che riguardano la BFLY16 e lo scarico, attraverso la porta B, dei risultati del passo nella relativa memoria. Il terzo passo e` perfettamente simile al secondo, con la sola differenza che ora i risultati parziali vengono presi dal buffer relativo alla portaB e, dopo il calcolo del passo, resi disponibili in uscita attraverso la porta Q. Un calcolo approssimativo del tempo preso dal calcolo della FFT di 1024 punti e` quindi esprimibile, considerando un clock di sistema di 40 MHz (25 nS di ciclo), come segue:

(18+68+68+3x1024)x25 nS=80.6 μS

Vengono riportati nella tabella i tempi relativi al calcolo di FFT su array di varia lunghezza. Per il calcolo del tempo richiesto dallo stesso set di dati reali, occorre aggiungere al calcolo della FFT di N/2 la latenza del passo di ricombinazione, che e` 18 cicli, piu` N cicli di clock.

N.ro di Punti	Struttura	N.ro di Punti	Struttura
8	2x4	4096	16x16x16
16	4x4	8192	2x16x16x16
32	2x16	16384	4x16x16x16
64	4x16	32768	2x4x16x16x16
128	2x4x16	65536	16x16x16x16
256	16x16	131072	2x16x16x16x16
512	2x16x16	262144	4x16x16x16x16
1024	4x16x16	524288	2x4x16x16x16x16
2048	2x4x16x16	1048576	16x16x16x16x16

Tabella 3a Struttura ottimale della FFT in base al numero di punti.

N.ro di Punti	Tempo	N.ro di Punti	Tempo
512	42.25µs	32K	4.102ms
1K	80.65µs	64K	6.560ms
2K	209.1µs	128K	16.39ms
4K	312.3µs	256K	32.78ms
8K	824.8µs	512K	78.65ms
16K	1.644ms	1M	131.08ms

Tabella 4 · Tempi di calcolo di FFT complesse a seconda del numero di punti.

(file: shar3.doc)

CAP. 4

L' ADDRESS GENERATOR LH 9320.

Come gia' accennato in precedenza, il DSP LH9124 e` stato espressamente disegnato per lavorare in coppia con il suo generatore di indirizzi LH9320. Questo e` un address generator programmabile in grado di generare oltre 150 sequenze diverse su un range di indirizzamento che, operando a 20 bit, spazia tra 4 punti ed 1 milione di punti. Sfruttando la sua architettura interna dedicata e` in grado di generare tutte quelle particolari sequenze di indirizzamento che sono richieste dalle funzioni implementate a bordo del DSP LH9124; in altri termini l' LH9320 non richiede, come gli altri address generator standard, la programmazione degli algoritmi di indirizzamento ma ha gia' implementato potenti istruzioni che generano direttamente la sequenza per l' algoritmo richiesto anche se, il 75% sono dedicate al calcolo della FFT in tutte le sue forme. Il chip e`, per la sua architettura, ottimizzato per fornire i pattern per gli algoritmi:

a) FFT	(Fast Fourier Transform)
b) DFT	(Discrete Fourier Transfom)
c) FIR filters	(Finite Impulse Response)
d) FCT	(Fast Cosine Transform)
e) DCT	(Discrete Cosine Transform)

L'LH9320 puo' generare 150 diverse sequenze di indirizzamento orientate agli algoritmi:

1) *Digit Reverse for FFT*
2) *Two at a time real data only*
   FFT Separation Pass.
3) *Decimation/ auto-decimation.*
4) *Modulo Increment / decrement.*
5) *Overlap / discard*
6) *Radix-n (n=2,4,16) and mixed radix FFTs.*
7) *Double length (2N) real data only FFT*
   FFT separation Pass.
8) *FIR Filtering.*
9) *Interpolation addressing (index filling / index padding)*
10) *Circular buffering*

Il chip, contrariamente al DSP, ha un suo piccolo buffer di memoria in grado di contenere 32 istruzioni che anche se sembra essere di piccole dimensioni, e`

piu' che sufficiente per il fatto che ogni istruzione indirizza un intero passo di programma. E' comunque possibile, per algoritmi molto complessi, usare una memoria esterna al chip. Il dispositivo viene visto dal sistema di controllo come una qualsiasi periferica, grazie alla presenza di una interfaccia standard ad 8 bit ed e` in grado di generare tutti i segnali di controllo per la gestione dei processi di lettura e scrittura della memoria con i segnali di clock necessari al DSP per ricevere o fornire i dati attraverso la porta interessata. E` anche possibile generare indirizzamenti multicanale, che permettono al DSP di elaborare segnali diversi in sequenza, con l' ausilio di una logica di arbitraggio interna dei vari canali. Questo significa che da un solo canale e` possibile gestire 1 milione di indirizzi diversi, sino a 32 canali con ciascuno un massimo di 32 K indirizzi (32Kx32K=1 M).
Nella tavola che segue,viene riportato il set completo di pattern che il chip puo` generare.

LH9320 Address Pattern Set Summary

MNEMONIC	DESCRIPTION
Fast Fourier Transforms (FFTs)	
BF2n (0 to 19)	Radix-2 data addresses
BF4n (0 to 9)	Radix-4 data addresses
BF16n (0 to 4)	Radix-16 data addresses
TF2n (0 to 19)	Radix-2 twiddle addresses
TF4n (0 to 9)	Radix-4 twiddle addresses
TF16n (0 to 4)	Radix-16 twiddle addresses
MXB24n (0 to 8)	Mixed radix (2,4) data addresses
MXB216n (0 to 3)	Mixed radix (2,16) data addresses
MXB416n (0 to 3)	Mixed radix (4,16) data addresses
MXB2416n (0 to 3)	Mixed radix (2,4,16) data addresses
MXT24n (0 to 8)	Mixed radix (2,4) twiddle addresses
MXT216n (0 to 3)	Mixed radix (2,16) twiddle addresses
MXT416n (0 to 3)	Mixed radix (4,16) twiddle addresses
MXT2416n (0 to 3)	Mixed radix (2,4,16) twiddle addresses
RBF0	Digit-reversed data address column 0
Separation Passes	
BRFTL	2-at-a-time real FFT separation pass, 2N, load
BRFTLS	2-at-a-time real FFT separation pass, N, load
BRFTU	2-at-a-time real FFT separation pass, 2N, unload
BRFTUS	2-at-a-time real FFT separation pass, N, unload
BFCTL	Fast cosine transform separation pass data addresses, 2N (long), load
BFCTT	Fast cosine transform separation pass twiddle addresses, 2N
BFCTUS	Fast cosine transform separation pass data addresses, N (short), load
BFCTU	Fast cosine transform separation pass data addresses, 2N, unload
BFCTUP	Fast cosine transform separation pass data addresses, 2N, unload using PO flag
BFCTUEP	Fast cosine transform separation pass data addresses, 2N, unload using early PO flag
BFCTULP	Fast cosine transform separation pass data addresses, 2N (long), unload using late PO flag
Decimation	
DECIM	Decimate
ADECIM	Auto-decimate

MNEMONIC	DESCRIPTION
Finite Impulse Response (FIR) Filters	
LPFIR1	Linear phase FIR, odd length, even symmetry
LPFIR2	Linear phase FIR, even length, even symmetry
LPFIR3	Linear phase FIR, odd length, odd symmetry
LPFIR4	Linear phase FIR, even length, odd symmetry
General Purpose Addressing/Utilities	
NOP	No operation
INC	Index increment
MODINC	Modulo increment
MODDEC	Modulo decrement
INTER	Interpolate/index fill
INTERP	Interpolate/index fill using PO flag
INTEREP	Interpolate/index fill using early PO flag
INTERLP	Interpolate/index fill using late PO flag
PADHIGH	Pad at end of sequence
PADHIGHP	Pad at end of sequence using PO flag
PADHIGHEP	Pad at end of sequence using early PO flag
PADHIGHLP	Pad at end of sequence using late PO flag
PADLOW	Pad at start of sequence
PADLOWP	Pad at start of sequence using PO flag
PADLOWEP	Pad at start of sequence using early PO flag
PADLOWLP	Pad at start of sequence using late PO flag
OVERLAP	Overlap
DISCARD	Discard
DISCARDP	Discard using PO flag
DISCARDEP	Discard using early PO flag
DISCARDLP	Discard using late PO flag
CMAG	Square of magnitude of a complex number
Circular Buffer Addressing	
CBUFFIR	Circular buffering for FIRs
CBUFFFT	Circular buffering for FFTs
Other	
CLRSIG	Clear signature
VIEWSIG	View signature

TAV. 42

IL CHIP LH9320.

La rappresentazione schematica del chip con tutti i segnali di sistema e di interfaccia, appare in fig 43 ed una sommaria descrizione dei pin è illustrata in tavola 5 Vediamo ora di analizzarli brevemente uno per uno, premettendo che esistono due modi fondamentali:

1- Funzionamento con memoria di programma interna (*Internal Memory Mode)*
2- Funzionamento con memoria di programma esterna *(External Memory Mode)*

Nel caso in cui intenderemo usare il chip set, si prevede l' uso della memoria di programma interna per cui affronteremo specificatamente questo caso.

A) Segnali di interfaccia con il sistema di controllo.

- DB[7:0] (Data Bus). Bus bidirezionale, a logica positiva, che permette la scrittura o lettura dei numerosi registri interni del chip; tale processo di scrittura o lettura avviene in due fasi: si scrive prima l'indirizzo del registro interessato ed immediatamente si scrive o si legge il dato relativo. Questo avviene a seconda del livello impostato in A0 , A1.

- A0,A1 (Host Control). Questi ingressi, attivi alti, oltre che gestire il modo di funzionamento, arbitrano anche lo scambio dei dati. Per sfruttare la memoria interna *(Internal Memory Mode)* si deve mantenere il livello del pin A1 basso, per cui il modo di funzionamento del chip e` determinato dal livello a cui e` portato il pin A0:

-A0=1 --> Il byte presente sul bus DB[7:0] e` interpretato come un indirizzo di un registro interno da scrivere.

-A0=0 --> Il byte presente sul bus DB[7:0] e` interpretato come un indirizzo di un registro interno da leggere.

- R/W (Read / Write). Ingresso, logica positiva. Questo segnale costituisce il clock vero e proprio per i registri interni dell' LH9320, alcuni dei quali (esempio il registro degli indirizzi) sono sensibili al fronte di salita ed altri (esempio il registro di programma) a quello di discesa. Questo porta a rendere disponibile un dato dopo un intero ciclo di clock (transizione basso --> alto e alto --> basso). La gestione, da parte del controllore di sistema, di questo segnale deve essere particolarmente curata per non introdurre conflitti sul bus per il fatto che questo segnale gestisce anche la direzione dei dati sullo stesso (R/W=1).

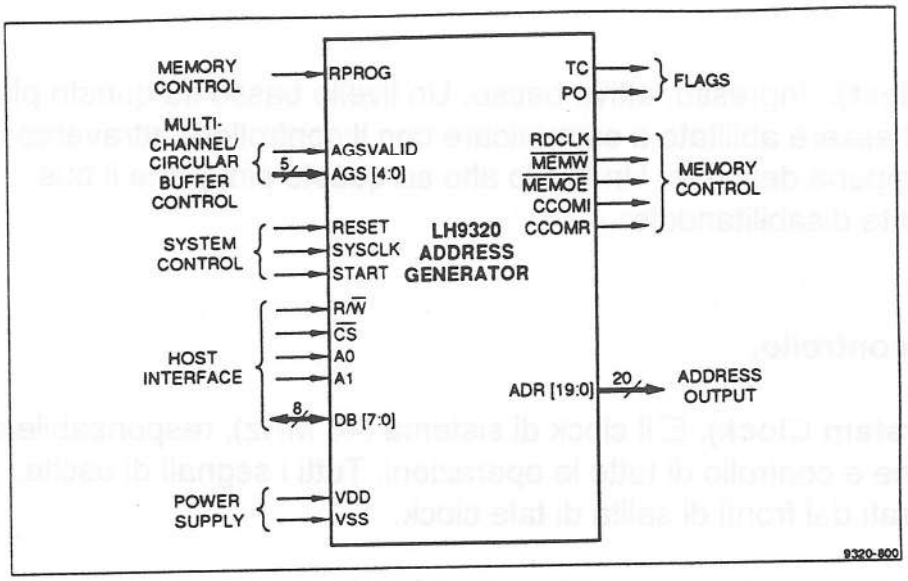


Figure 43 LH9320 Simplified Block Diagram

Table 6-3
LH9320 Signal-Pin Descriptions

SIGNAL *	DIRECTION **	SIGNAL NAME
DB[7:0]	I/O	Host Interface data bus
A1, A0	I	Host control
R/$\overline{W}$	I	Read/write host
$\overline{CS}$	I	Chip select
SYSCLK	I	System clock
START	I	Start of a pass
TC	O	Terminal count
ADR[19:0]	O	Address outputs
AGS[4:0]	I	Circular buffer channel sample request
AGSVALID	I	Circular buffer valid channel flag
CCOMR	O	Coefficient complement real
CCOMI	O	Coefficient complement imaginary
RESET	I	Chip reset
PO	O	Programmable output
$\overline{MEMW}$	O	Memory write
$\overline{MEMOE}$	O	Memory output enable
RDCLK	O	Read clock
RPROG	I	Programmable Pulsewidth for $\overline{MEMW}$
VDD	P	+ 5 volt power supply
VSS	P	Ground for the chip

* Numbers in brackets indicate the number of bits. For example, ADR[19:0] indicates 20 bits.
** I = Input, O = Output, I/O = Input and Output, P = Power.

TAV. 5

- **CS (Chip Select).** Ingresso, attivo basso. Un livello basso su questo pin, porta il chip ad essere abilitato a comunicare con il controllore attraverso il meccanismo appena descritto. Un livello alto su questo pin, porta il bus DB[7:0] in 3-state disabilitandolo.

B) Segnali di controllo.

- **SYSCLK (System Clock).** E`il clock di sistema (40 MHz), responsabile della sincronizzazione e controllo di tutte le operazioni. Tutti i segnali di uscita, vengono generati dai fronti di salita di tale clock.

- **RESET (Chip Reset).** Ingresso, attivo alto. Tramite questo pin, e` possibile resettare tutto il sistema ponendolo a livello alto; e` chiaro che durante il normale funzionamento, questo pin deve essere forzato a livello basso. Affinche` tale operazione di system reset sia effettiva, occorre attendere almeno due cicli di SYSCLK e due cicli di R/W.

- **START (Start of Pass).** Ingresso, attivo sul fronte di salita. Al fronte di salita di questo segnale viene dato il via alla generazione degli indirizzi della sequenza programmata. La prima sequenza in uscita viene fornita dopo 5 cicli di clock (latenza=5) dallo START, mentre le successive si ottengono senza, in pratica, nessuna latenza se il relativo segnale di START viene dato mentre e` ancora attiva la sequenza precedente.

C) Segnali di Controllo della Memoria.

- **MEMW (Memory Write).** Uscita, attiva bassa. Questo segnale controlla la scrittura dei dati sulla memoria indirizzata dallo stesso LH9320.

MEMW=1 --> dopo un reset, durante la lettura dati dal buffer.
MEMW=0 --> durante la scrittura del buffer di memoria.

Il livello di questo pin e` alto subito dopo un system reset e quando si leggono i dati dalla memoria. E` a livello basso quando si scrivono i dati sui buffer di memoria.

- **MEMOE (Memory Output Enable).** Uscita, attiva bassa. Il livello di questo pin, controlla la direzione del flusso dei dati verso o dalla memoria:

MEMOE=1 --> dopo un reset, scrittura dei buffer.
MEMOE=0 --> lettura del buffer.

- **RDCLK (Read Clock).** Uscita. Questo segnale e` normalmente basso. Il fronte di salita indica che i dati presenti sulle RAM sono pronti per essere letti. Questo segnale viene usato per il pilotaggio degli ingressi CLKxIN del DSP LH9124.

- **CCOMR (Coefficient Complement Real).** Uscita, logica positiva.

- **CCOMI (Coefficient Complement Immaginary).** Uscita, logica positiva.

Ambedue i segnali, vengono usati in combinazione con alcune istruzioni per il pilotaggio diretto degli ingressi CCR e CCI dell' LH9124. Mediante l'uso di questi segnali, e` possibile ottenere coefficienti su 360 gradi, memorizzandone solo 90 gradi. Dopo un segnale di system reset e durante l'esecuzione di istruzioni che non hanno il controllo di CCOMR e CCOMI, questi vengono posti in 3-state.

- **RPROG (Programmable MEMW Pulsewidth).** Ingresso. Questo pin viene usato per definire la giusta temporizzazione per il segnale MEMW, in modo da garantire che rimanga a livello alto per tutta la durata della generazione della sequenza di indirizzi. Tipicamente, questo segnale viene portato a livello alto tramite una resistenza da $0K\Omega$ che consente di ottenere le temporizzazioni consigliate dalla Sharp.

D) Segnali di Flag.

- **TC (Terminal Count).** Uscita, logica positiva. Questo segnale indica la fine della sequenza di indirizzamento corrente. Questo segnale passa da livello basso a livello alto, un ciclo di clock prima della fine della sequenza per poi rimanervi fino al successivo segnale di START per la generazione della nuova sequenza di indirizzamento relativa al passo che segue. E` possibile ritardare di un ciclo la sopradetta transizione, agendo su di un particolare registro dell' Address Generator stesso.

- **PO (Programmable Output).** Uscita, logica positiva. Il segnale PO lavora solo in congiunzione ad alcune istruzioni e rende possibile la indicizzazione dei dati. E`, ad esempio, possibile attivare PO ogni M indirizzi generati per un tempo pari ad un certo numero di cicli di clock. Oppure si puo` programmare LH9320 in modo tale che PO sia attivo per un certo numero di cicli prima dell' inizio di una determinata sequenza, o lo stesso numero di cicli dopo la fine della stessa sequenza, ecc..... Pilotando in modo opportuno con PO i pin di LH9124 DCI,DCO,DZI o DZO, si possono effettuare decimazioni, interpolazioni o riempimenti (padding) con zeri dei dati da elaborare.

IL SET DI ISTRUZIONI PER LA FFT.

Il chip LH9320 e` particolarmente ottimizzato per operare con LH9124 nel computo, ad elevatissima velocità, della FFT; per questo motivo ci soffermeremo piu` in dettaglio sul set di istruzioni relative agli algoritmi per la FFT. L'address generator e` in grado di generare ad ogni istruzione, la sequenza richiesta da ogni stadio di elaborazione con la sola programmazione di alcuni parametri nei registri interni del chip quali, ad esempio, il numero N di punti da generare e le dimensioni della memoria (registro MEMSIZE). Consideriamo il calcolo di una FFT complessa di N punti, con N pari ad una potenza di 2 e sapendo che l'LH9320 opera a 20 bit, la lunghezza dell' array che il DSP deve trasformare, deve per forza essere compresa tra 4 e 1 milione (2^{20}). Ai fini della programmazione di quest' ultimo, introduciamo il concetto di *colonna* dicendo che una FFT di 2^M punti è costituita da M colonne. Se ciascuna di queste può essere formata da uno stadio radix-2, portando ad avere in tal modo la FFT formata da M colonne, due o quattro colonne contigue possono essere implementate con uno stadio radix-4 o radix-16 rispettivamente. Numerando le colonne così ottenute dalla colonna 0 alla colonna M-1, si può dire che la colonna zero può essere formata da uno stadio radix-2,4 o16. Ogni istruzione del DSP è in grado di calcolare uno stadio (non una colonna) e tre istruzioni di indirizzamento da inviare all' address generator: una per i dati di ingresso, una per i coefficienti twiddle da "pescare" attraverso la porta C e l' ultima, per la scrittura dei dati di uscita sia che questi siano definitivi che risultati intermedi. Il tipo di istruzione necessaria per indirizzare sia i dati che i coefficienti è funzione di due soli parametri che, in particolare, sono: l' ordine della colonna iniziale dello stadio ed il tipo di operazione con cui tale stadio viene calcolato (radix-2,4,o16). In tab.7 si riporta il set di istruzioni a disposizione, per l' indirizzamento dei dati in funzione dei parametri suddetti. Per commentare e capire meglio tale set di di istruzioni, facciamo un esempio pratico in cui si debba calcolare una FFT di 128 punti complessi. Dalla precedente tab. 6 si ricava che la migliore struttura per tale FFT risulta essere la 2X4X16, cioè una struttura radix-2,radix-4 e radix-16, per cui, essendo la prima colonna implementata con una BFLY2, dalla stessa tab.6 si deduce che si devono indirizzare le memorie di ingresso (porta Q) ed uscita, programmando i relativi address generator con la istruzione BF20. Lo stadio successivo della precedente struttura, è un BFLY4 ed inizia alla colonna 1 (occupando poi 2 colonne come visto in precedenza) per cui ora occorre programmare l' address generator dei dati con l' istruzione MXB240. Il terzo ed ultimo stadio della struttura inizia alla colonna 3 per cui dalla tab.6 si vede che si deve indirizare i dati di I/O programmando i relativi generatori di indirizzi con la istruzione MXB24160. Per l' indirizzamento dei coefficienti, sono previste istruzioni che sono scelte in base ad un criterio analogo a quello appena esposto, e rappresentate nella loro completezza nella tab.7. Proseguendo con l' esempio pratico appena trattato per quello che riguarda la determinazione

delle istruzioni per l' address generator, ricaviamo la analoga serie di istruzioni per l' indirizzamento dei twiddle. Dalla tab.7 si ricava per la colonna 0 TF20, per la colonna 1 (secondo passo radix-4) MXT240 e per la colonna 3 (il precedente prendeva 2 colonne) MXT24160. Gli algoritmi utilizzati per generare queste sequenze sono molto complessi come pure complesse sono le motivazioni che stanno alla base della loro filosofia di funzionamento, per cui viene fornito dalla Sharp un apposito software che genera le sequenze di indirizzamento a valle di ognuna delle istruzioni appena viste, sia per i dati che per i coefficienti.

Indirizzamento dei dati in ordine inverso.

Osservando la tabella riportata nell' esempio precedente, si nota che, a seconda della radix-2,4 o16 scelta, si avrebbero le sequenze di indirizzamento, relative alla prima colonna, BF20,BF40 e BF160 rispettivamente. Queste istruzioni richiedono che i dati che vanno ad indirizzare, siano stati precedentemente memorizzati in ordine inverso *(bit reversed)*. Questo succede per il fatto che le istruzioni BF20,BF40,BF160 generano delle sequenze di indirizzamento lineari da 0 a N-1 (se N è il numero di punti). In pratica, nella maggior parte dei casi, i dati provengono da un convertitiore Analogico/Digitale per cui sono memorizzati in ordine temporale "naturale". In questo caso occorre che sia il dispositivo di indirizzamente a farsi carico della generazione della sequenza di indirizzamento adatta alla lettura in ordine inverso degli stessi: il chip LH9320 è in grado di fare questo attraverso l' istruzione **RBFO**.Questa istruzione genera sequenze in ordine inverso che sono adatte all' indirizzamento dei dati in ingresso al primo stadio della FFT in funzione dei parametri: numero di punti N e quello denominato DIGITREV. Quest' ultimo fornisce l' informazione di come la FFT è strutturata perchè, di fatto, è da quale tipo e di quanti stadi è formata che dipende l' ordine dei dati in ingresso. Di conseguenza è di fondamentale importanza conoscere a priori la struttura della FFT perchè l' LH9320 deve essere istruito a questo proposito, non potendo "sapere" quale tipo di struttura sara' usata. Un modo immediato per definire il contenuto del registro DIGITREV è il seguente:

1) Si scrive la struttura della FFT nell' ordine in cui questa verrà computata, partendo con il primo stadio alla sinistra e l' ultimo sulla destra tipo:

radix-2 x radix-4 x radix-2 x radix-2 x radix-16 x radix-2

2) Scrivere una sequenza binaria ponendo rispettivamente 0 per i passi radix-2, 11 per i passi radix-4 e 1111 per quelli radix-16. L' esempio riportato al punto 1 diventa:

0110011110

COLONNA	RADIX-2	RADIX-4	RADIX-16
0	BF20	BF40	BF160
1	BF21	MXB240	MXB2160
2	BF22	BF41	MXB4160
3	BF23	MXB241	MXB24160
4	BF24	BF42	BF161
5	BF25	MXB242	MXB2161
6	BF26	BF43	MXB4161
7	BF27	MXB243	MXB24161
8	BF28	BF44	BF162
9	BF29	MXB244	MXB2162
10	BF210	BF45	MXB4162
11	BF211	MXB245	MXB24162
12	BF212	BF46	BF163
13	BF213	MXB246	MXB2163
14	BF214	BF47	MXB4163
15	BF215	MXB247	MXB24163
16	BF216	BF48	BF164
17	BF217	MXB248	
18	BF218	BF49	
19	BF219		

Tabella 6 Set di istruzioni per l'indirizzamento dei *dati* nella FFT, in funzione della colonna iniziale e del tipo dello stadio.

COLONNA	RADIX-2	RADIX-4	RADIX-16
0	TF20	TF40	TF160
1	TF21	MXT240	MXT2160
2	TF22	TF41	MXT4160
3	TF23	MXT241	MXT24160
4	TF24	TF42	TF161
5	TF25	MXT242	MXT2161
6	TF26	TF43	MXT4161
7	TF27	MXT243	MXT24161
8	TF28	TF44	TF162
9	TF29	MXT244	MXT2162
10	TF210	TF45	MXT4162
11	TF211	MXT245	MXT24162
12	TF212	TF46	TF163
13	TF213	MXT246	MXT2163
14	TF214	TF47	MXT4163
15	TF215	MXT247	MXT24163
16	TF216	TF48	TF164
17	TF217	MXT248	
18	TF218	TF49	
19	TF219		

TAV. 7

3) Alla sequenza di numeri binari così ottenuta, si aggiungono alla sinistra tanti zeri quanti ne mancano per arrivare ai 20 bit. Nel caso dell' esempio si ottiene:

00000000000110011110

Concludendo, nel caso in cui i dati in ingresso siano memorizzati nell' ordine naturale con cui arrivano, ad esempio, dal convertitore Analogico -Digitale, si dovrà programmare il generatore di indirizzi per questi ultimi con l' istruzione RBF0 anzichè con BFx0.

Indirizzamento dei coefficienti.

Le istruzioni per la programmazione delle sequenze di indirizzamento dei coefficienti richiedono, per un corretto funzionamento, la definizione di tre parametri. Il primo è costituito dalla lunghezza dell' array N da maneggiare che equivale anche al numero di indirizzi da generare, il secondo parametro è la dimensione del buffer di memoria (MEMSIZE) in cui i coefficienti sono memorizzati ed il terzo, da un flag (bit Mode) che indica se sono disponibili in memoria tutti i 360^0 di coefficienti oppure solo i 90^0 piu' il punto a 270^0. Per fare alcune considerazioni pratiche, consideriamo di essere nella situazione in cui si hanno tutti i coefficienti. Il numero degli indirizzi N da generare può essere diverso dal numero dei coefficienti memorizzati di un fattore *d* che deve essere una potenza di 2. In altri termini, il numero dei coefficienti che sono realmente memorizzati può essere uguale a N oppure uguale a N*x*d. In questo caso l'effettivo indirizzo di ogni singolo coefficiente, viene ottenuto moltiplicando quello calcolato per il fattore *d,* realizzando in tal modo una specie di decimazione dei coefficienti ottenuta leggendone uno ogni *d*. Apparentemente sembra un controsenso avere una memoria di una certa dimensione per poi sfruttarla solo in parte , però la flessibilità di tutto il sistema aumenta notevolmente per il fatto che diventa possibile calcolare FFT di lunghezze diverse, minori od uguali a MEMSIZE sfruttando gli stessi coefficienti. Lo stesso meccanismo si può considerare valido anche per il caso in cui si abbiano solo 90^0 di coefficienti tenendo però conto del fatto che i coefficienti ottenuti dal solo quarto quadrante, devono essere aggiustati in segno sulla parte reale od immaginaria per ottenere tutti i rimanenti, come già visto in precedenza. L' operazione di aggiustaggio dei segni per la determinazione dei coefficienti mancanti, è di competenza dei segnali CCOMR e CCOMI gestiti dall' LH9320 qualora il flag Mode indichi che è stato memorizzato solo un quarto dei coefficienti. Da un punto di vista hardware le suddette operazioni si ottengono collegando i pin CCOMR e CCOMI dell ' LH9320, rispettivamente ai pin CCR e CCI del DSP. Nella tabella che segue vengono mostrati gli stati di questi segnali in funzione dell' indirizzo generato Adr, ricordando che un livello alto di uno dei due segnali di controllo CCOMR e/o CCOMI significa "cambia segno" rispettivamente sulla parte reale e/o immaginaria letta e sottolineando

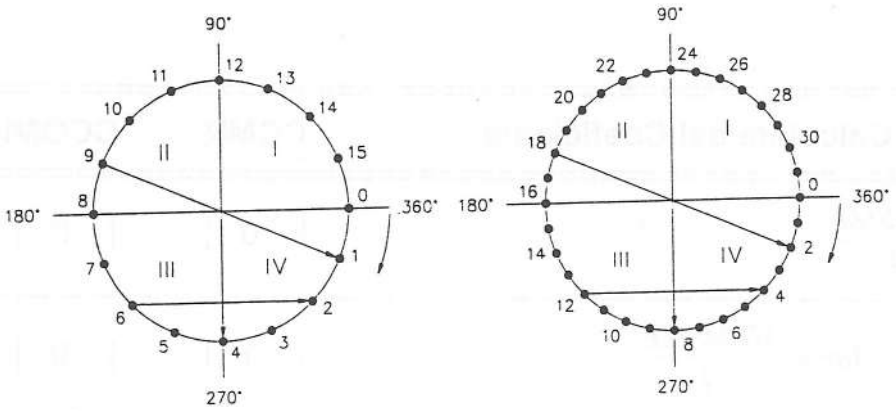


Figura 43a Rappresentazione grafica dei coefficienti twiddle per N=16 e N=32 a dimostrazione di quanto affermato nella Tab. 4.4.

Mode[4]	N	Memsize		1..5	6	7	8	9	10	11	12	13	14	15	16
0	16	16	Adr	0	1	2	3	0	2	4	6	0	3	6	9
			CCOMR	0	0	0	0	0	0	0	0	0	0	0	0
			CCOMI	0	0	0	0	0	0	0	0	0	0	0	0
0	16	32	Adr	0	2	4	6	0	4	8	12	0	6	12	18
			CCOMR	0	0	0	0	0	0	0	0	0	0	0	0
			CCOMI	0	0	0	0	0	0	0	0	0	0	0	0
1	16	16	Adr	0	1	2	3	0	2	4	2	0	3	2	1
			CCOMR	0	0	0	0	0	0	0	1	0	0	1	1
			CCOMI	0	0	0	0	0	0	0	0	0	0	0	1
1	16	32	Adr	0	2	4	6	0	4	8	4	0	6	4	2
			CCOMR	0	0	0	0	0	0	0	1	0	0	1	1
			CCOMI	0	0	0	0	0	0	0	0	0	0	0	1

Tabella 8 - Indirizzi generati e stato dei segnali di controllo del segno relativi ad una istruzione TF40, per diversi valori dei parametri.

ancora che questi due segnali sono mantenuti in 3-state durante tutte le altre operazioni.

Posizione Calcolata del Coeficiente	CCMR	CCOMI
$Adr \le \frac{MEMSIZE}{4}$	0	0
$\frac{MEMSIZE}{4} < Adr < \frac{MEMSIZE}{2}$	1	0
$\frac{MEMSIZE}{2} < Adr < \frac{3 \times MEMSIZE}{4}$	1	1
$\frac{3 \times MEMSIZE}{4} < Adr < MEMSIZE$	0	1

Come ultima considerazione sull' indirizzamento dei coefficienti, ne riportiamo nella tab.8 le combinazioni in funzione dello stato dei vari segnali relativi ad una istruzione TF40 (stadio radix-4 nella prima colonna) riportandone anche la fig.43a in cui N=16, MEMSIZE=16 e N=32, MEMSIZE=32

Mappa di Memoria dell' LH9320.

Ogni istruzione dell' LH9320 per la generazione delle sequenze di indirizzamento, comprede le specificazioni di parametri che devono essere programmati negli appositi registri, prima che l' elaborazione abbia inizio, dal controllore di sistema. Nella tab.9 viene riportata la mappa dei registri interni dell' Address Generator come fornita dalla stessa Sharp. Si sono riportati in grassetto quelli che maggiormente ci interessano, fra cui alcuni già a noi noti come il registro N o il registro DIGITREV, e che ora commentiamo molto brevemente. Nei primi 32 registri "ADDRESS PATTERN 0....31", con indirizzo 0-->1F, viene memorizzato il programma che contiene la lista delle sequenze da generare relative ai vari passi dell' algoritmo di calcolo corrente. Subito dopo, dall' indirizzo 20 all' indirizzo 3F, troviamo altri 32 registri, da 8 bit come i precedenti, della "PIPELINE LATENCY 0.....31" che hanno il compito di memorizzare i valori di latenza per ogni istruzione di programma; in altri termini, ad ogni istruzione corrisponde un ben determinato valore di latenza che altro non è che il ritardo con cui si deve fare partire la generazione degli indirizzi della sequenza relativa. Questo fornisce la possibilità di sincronizzare tutto il chip set LH9124 e tutti gli LH9320 ad esso collegati, con un unico segnale di START. Il byte memorizzato in ciascun registro della "PIPELINE

LATENCY", è composto dai primi 7 bit che costituiscono il valore vero e proprio, mentre l' ottavo bit è quello che stabilisce se scrivere o leggere in memoria. Proseguendo, si incontrano i registri PCSTART e PCEND che memorizzano gli indirizzi di partenza e di terminazione del programma. L' LH9320 appena è fatto partire da START, inizia a generare la sequenza di indirizzamento relativa all' istruzione " puntata" da PCSTART, continuando nella generazione, fino a quella dell' istruzione puntata da PCEND. Un nuvo ciclo di generazione indirizzi, richiede un nuovo segnale di START. Come ultimo abbiamo il registro MODE di cui conosciamo il bit 4 già usato nell' indirizzamento dei coefficienti e dove i rimanenti bit servono per definire:

a) Il tipo di memoria (Interna o Esterna).
b) Se usare o meno la funzione di Breakpoint.
c) Indirizzamento multicanale.
d) Se attivare con un ritardo di un ciclo, TC.
e) Se moltiplicare per 2 il valore della latenza memorizzata nei relativi registri.

Informazioni dettagliate di tutti i registri, vegono riportate dai data sheets della SHARP. Facciamo ora alcune considerazioni sull' Address Generator LH9320, prima di concludere il capitolo. Un problema che si è dimostrato essere irrisolvibile è quello di potere fare iniziare la sequenza a partire da un determinato indirizzo, perlomeno con certe istruzioni. Un secondo problema che, alla attuale nostra conoscenza del chip, sembra essere di non facile soluzione, è quello di potere fermare a piacimento ed in qualsiasi istante la sequena corrente e farla poi ripartire successivamente, dallo stesso punto. In realtà, questa opportunità sembra essere stata prevista con l' inserzione del registro HALT precludendo qualsiasi gestione hardware del processo di generazione degli indirizzi.

1	**ADDRESS PATTERN 1**	A1	NCHANNEL[4:0]
2	**ADDRESS PATTERN 2**	A2	**MODE**
	⋮	A3	OVERLAP[7:0]
1F	**ADDRESS PATTERN 31**	A4	OVERLAP[15:8]
20	**PIPELINE LATENCY 0**	A5	OVERLAP[19:16]
21	**PIPELINE LATENCY 1**	A6	PAUSE[7:0]
	⋮	A7	PAUSE[15:8]
3F	**PIPELINE LATENCY 31**	A8	PAUSE[19:16]
40	CHAN.0 LENGTH[7:0]	A9	BREAKPOINT[7:0]
41	CHAN.1 LENGTH[7:0]	AA	BREAKPOINT[15:8]
	⋮	AB	BREAKPOINT[19:16]
5F	CHAN.31 LENGTH[7:0]	AC	**N[7:0]**
60	CHAN.0 LENGTH[15:8]	AD	**N[15:8]**
61	CHAN.1 LENGTH[15:8]	AE	**N[20:16]**
	⋮	AF	IFACTOR[7:0]
7F	CHAN.31 LENGTH[15:8]	B0	IFACTOR[15:8]
80	CHAN.0 LENGTH[19:16]	B1	IFACTOR[19:16]
81	CHAN.1 LENGTH[19:16]	B2	DFACTOR[7:0]
	⋮	B3	DFACTOR[15:8]
9F	CHAN.31 LENGTH[19:16]	B4	DFACTOR[19:16]
B5	LEAP[7:0]	CA	**ADRDEC[7:0]**
B6	LEAP[15:8]	CB	**ADRDEC[15:8]**
B7	LEAP[19:16]	CC	**ADRDEC[19:16]**
B8	NLEAP[7:0]	CD	INDEXADR[7:0]
B9	NLEAP[15:8]	CE	INDEXADR[15:8]
BA	NLEAP[19:16]	CF	INDEXADR[19:16]
BB	INTERADR[7:0]	D0	INDEXINC[7:0]
BC	INTERADR[15:8]	D1	INDEXINC[15:8]
BD	INTERADR[19:16]	D2	INDEXINC[19:16]
BE	ZEROPAD[7:0]	D3	**PCEND[4:0]**
BF	ZEROPAD[15:8]	D4	HALT
C0	ZEROPAD[19:16]	D5	**DIGITREV[7:0]**
C1	**ADRSTART[7:0]**	D6	**DIGITREV[15:8]**
C2	**ADRSTART[15:8]**	D7	**DIGITREV[19:16]**
C3	**ADRSTART[19:16]**	D8	**MEMSIZE[7:0]**
C4	**ADRLENGTH[7:0]**	D9	**MEMSIZE[15:8]**
C5	**ADRLENGTH[15:8]**	DA	**MEMSIZE[20:16]**
C6	**ADRLENGTH[19:16]**	DB	CHIPID
C7	**ADRINC[7:0]**	DC	DISCARD[7:0]
C8	**ADRINC[15:8]**	DD	DISCARD[15:8]
C9	**ADRINC[19:16]**	DE	DISCARD[19:16]

TAV 9- Mappa della memoria interna del generatore di indirizzi LH9320. Seconda parte.

LH9320 REAL TIME SIMULATOR.

Dopo avere brevemente introdotto il chip LH9320 ed avere effettuato una rapida panoramica sulle istruzioni che questo mette a disposizione per il calcolo della FFT, consideriamo ora il simulatore che la SHARP mette a disposizione per la creazione di sequenze di indirizzamento in funzione delle istruzioni introdotte. Questo ci darà l' opportunità di capire meglio le differenze tra una istruzione a l' altra e, soprattutto, le differenze tra le sequenze generate dalla stessa istruzione in condizioni diverse.

Risultati delle simulazioni.

Nella prima tabella vengono riportate tre uscite diverse generate dalla stessa istruzione RBF0 che, come noto, è in grado di generare le sequenze di ordine inverso necessarie per l' indirizzamento dei dati di ingresso di FFT a decimazione di tempo. Tutte e tre le sequenze di uscita, sono state calcolate ponendo MEMSIZE=8 e N=8, ma nell' ambito di tre FFTdi diversa struttura, cioè con tre diversi valori programmati nel registro DIGITREV. Considerando che una FFT di 8 punti è formata da 3 colonne, la si può pensare realizzata da 3 stadi radix-2 o da un radix-2 ed un radix-4 dove i due termini possono essere scambiati, come ordine, fra di loro. Le tre diverse maniere di realizzare la stessa FFT, portano ad un riordinamento dei dati di ingresso, come visibile in tab.20, ottenuto seguendo tre criteri leggermente diversi.
Si consideri ora FFT di 16 punti (4 colonne), relizzate con 4 stadi radix-2. Nella tab.21 sono state riportate entrambe le sequenze di indirizzamento dei dati e dei coefficienti con MEMSIZE=16. Si nota subito come la sequenza RBF0 generi gli indirizzi in ordine inverso, la BF20 al primo stadio della FFT la generi in ordine lineare perche` si aspetta che i dati siano stati memorizzati nel buffer di ingresso in ordine inverso come gia' sottolineato in precedenza. La istruzione per l' indirizzamento dei coefficienti del primo stadio TF20, genera un sequenza di indirizzi tutti a zero; questo significa che si indirizza per tutto il primo stadio lo stesso indirizzo "zero" che contiene il valore 1 e questo, se necessario, offre la possibilita` di effettuare il windowing dei dati. Nella tab.22 sono state riportate le sequenze prodotte dalle stesse istruzioni per una FFT della stessa lunghezza e struttura della precedente ma con MEMSIZE=32 (coefficienti in memoria doppi rispetto al caso precedente). Come si puo` notare dalla stessa tabella, il cambiamento di MEMSIZE non cambia le sequenze di indirizzamento generate dalle istruzioni RBF0,RBF20-1-2-3 ma solo i coefficienti che risultano essere esattamente il doppio. Nella tab.13 che segue, vengono riportate le sequenze relative ad una FFT con N=16 e EMSIZE=16 ma con stadi radix-4. Notiamo che qui l' istruzione RBF0 genera una sequenza che e` diversa dalla corrispondente del caso precedente per il fatto della radix-4 al primo stadio. Anche in questo caso gli indirizzi generati da TF20 per i coefficienti riultano essere tutti zeri, confermando quanto detto in

precedenza a proposito della possibilita` di potere effettuare il windowing dei dati anche con stadi radix-4, per il fatto di avere tutti i coefficienti uguali a 1 nel primo stadio. Come ultimo esempio riportiamo in tab.14 le sequenze di indirizzamento relative allo stesso caso di prima ma in cui si usa un radix-16. L' istruzione RBF0 produce la stessa sequenza di prima per via dello stesso valore di DIGITREV ed inoltre si nota come in effetti uno stadio radix-16 equivalga a due stadi radix-4 in cascata da un punto di vista dei dati da elaborare. Una ultima cosa che si puo` notare, e` che in questo caso la sequenza degli indirizzi generata da TF0 non e` piu` costituita da soli zeri come nel caso precedente, per cui non sarebbe possibile effettuare nessun tipo di windowing al primo stadio.

RBF0 2 x 2 x 2 DIGITREV=000	**RBF0** 2 x 4 DIGITREV=011	**RBF0** 4 x 2 DIGITREV=110
00000	00000	00000
00004	00004	00002
00002	00001	00004
00006	00005	00006
00001	00002	00001
00005	00006	00003
00003	00003	00005
00007	00007	00007

Tabella 10 Risultato dell'esecuzione della istruzione RBF0, con N=8, MEMSIZE=8, ma nell'ambito di tre strutture diverse.

FFT 2 x 2 x 2 x 2 (DIGITREV=0000), N=16, MEMSIZE=16								
RBF0	BF20	BF21	BF22	BF23	TF20	TF21	TF22	TF23
00000	00000	00000	00000	00000	00000	00000	00000	00000
00008	00001	00002	00004	00008	00000	00000	00000	00000
00004	00002	00004	00008	00001	00000	00000	00000	00000
0000C	00003	00006	0000C	00009	00000	00000	00000	00001
00002	00004	00008	00001	00002	00000	00000	00000	00000
0000A	00005	0000A	00005	0000A	00000	00000	00002	00002
00006	00006	0000C	00009	00003	00000	00000	00000	00000
0000E	00007	0000E	0000D	0000B	00000	00000	00002	00003
00001	00008	00001	00002	00004	00000	00000	00000	00000
00009	00009	00003	00006	0000C	00000	00004	00004	00004
00005	0000A	00005	0000A	00005	00000	00000	00000	00000
0000D	0000B	00007	0000E	0000D	00000	00004	00004	00005
00003	0000C	00009	00003	00006	00000	00000	00000	00000
0000B	0000D	0000B	00007	0000E	00000	00004	00006	00006
00007	0000E	0000D	0000B	00007	00000	00000	00000	00000
0000F	0000F	0000F	0000F	0000F	00000	00004	00006	00007

Tabella 11 - Risultato dell'esecuzione di tutte le possibili istruzioni per l'indirizzamento di una FFT costituita da 4 stadi radix-2. MEMSIZE=16, N=16.

FFT 2 x 2 x 2 x 2 (DIGITREV=0000), N=16, MEMSIZE=32								
RBF0	BF20	BF21	BF22	BF23	TF20	TF21	TF22	TF23
00000	00000	00000	00000	00000	00000	00000	00000	00000
00008	00001	00002	00004	00008	00000	00000	00000	00000
00004	00002	00004	00008	00001	00000	00000	00000	00000
0000C	00003	00006	0000C	00009	00000	00000	00000	00002
00002	00004	00008	00001	00002	00000	00000	00000	00000
0000A	00005	0000A	00005	0000A	00000	00000	00004	00004
00006	00006	0000C	00009	00003	00000	00000	00000	00000
0000E	00007	0000E	0000D	0000B	00000	00000	00004	00006
00001	00008	00001	00002	00004	00000	00000	00000	00000
00009	00009	00003	00006	0000C	00000	00008	00008	00008
00005	0000A	00005	0000A	00005	00000	00000	00000	00000
0000D	0000B	00007	0000E	0000D	00000	00008	00008	0000A
00003	0000C	00009	00003	00006	00000	00000	00000	00000
0000B	0000D	0000B	00007	0000E	00000	00008	0000C	0000C
00007	0000E	0000D	0000B	00007	00000	00000	00000	00000
0000F	0000F	0000F	0000F	0000F	00000	00008	0000C	0000E

Tabella 12 - Risultato dell'esecuzione di tutte le possibili istruzioni per l'indirizzamento di una FFT costituita da 4 stadi radix-2. MEMSIZE=32, N=16.

FFT 4 x 4 (DIGITREV=1111), N=16, MEMSIZE=16				
RBF0	BF40	BF41	TF40	TF41
00000	00000	00000	00000	00000
00004	00001	00004	00000	00000
00008	00002	00008	00000	00000
0000C	00003	0000C	00000	00000
00001	00004	00001	00000	00000
00005	00005	00005	00000	00001
00009	00006	00009	00000	00002
0000D	00007	0000D	00000	00003
00002	00008	00002	00000	00000
00006	00009	00006	00000	00002
0000A	0000A	0000A	00000	00004
0000E	0000B	0000E	00000	00006
00003	0000C	00003	00000	00000
00007	0000D	00007	00000	00003
0000B	0000E	0000B	00000	00006
0000F	0000F	0000F	00000	00009

Tabella 13 - Risultato dell'esecuzione di tutte le possibili istruzioni per l'indirizzamento di una FFT costituita da 2 stadi radix-4. MEMSIZE=16, N=16.

FFT 16 (DIGITREV=1111), N=16, MEMSIZE=16		
RBF0	BF160	TF160
00000	00000	00000
00004	00001	00000
00008	00002	00000
0000C	00003	00000
00001	00004	00000
00005	00005	00000
00009	00006	00000
0000D	00007	00001
00002	00008	00002
00006	00009	00003
0000A	0000A	00002
0000E	0000B	00004
00003	0000C	00006
00007	0000D	00003
0000B	0000E	00006
0000F	0000F	00009

Tabella 14 - Risultato dell'esecuzione di tutte le possibili istruzioni per l'indirizzamento di una FFT costituita da un solo stadio radix-16. MEMSIZE=16, N=16.

(file: shar4.doc)

CAP.5

INTRODUZIONE ALLA ARCHITTETURA DEL CHIP SET.

Abbiamo già avuto occasione di sottolineare la differenza tra le architetture dei DSP standard e quella del DSP della SHARP LH9124, soprattutto per quello che riguarda la disposizione della memoria che sta attorno al chip stessso. Siccome lo scopo di questa nota tecnica è quello di esplorare la possibilità di usare soluzioni multiDSP in cascata, uno dei problemi principali da risolvere è costituito dalla grande quantità di memoria SRAM veloce (15/20 ns.) richiesta. Dalla fig. 44 si vede che attorno all' LH9124 occorrono ben 4 pagine di memoria da N words (N=numero di punti da trasformare) ciascuna e considerando che si usano words da 24 bit, si intuisce subito quale quantità di memoria SRAM veloce si renda necessaria. Osservando la fig.44 notiamo anche che occorre un address generator per ogni buffer di memoria (costituito da buffer lunghi N/2 per il reale ed N/2 per quello immaginario) relativo a ciascuna delle 4 porte a disposizione. Vista la velocità del chip set, si nota dall' architettura base mostrata nella fig.50 (CAP.6) è estremamente adatta ad implementazioni in ambiente VME. Una considerazione particolare deve essere poi fatta sulla parte relativa alla porta Q di ingresso/uscita dei dati dove occorrono più pagine di memoria che possano costituire un sistema in cui, mentre si riempie una pagina, il DSP processa il contenuto di quella precedentemente " riempita"; questo per aumentare il duty-cycle. Uno schema a blocchi più dettagliato viene riportato in fig. 45 dove si nota la parte di I/O, a cui abbiamo accennato prima, che permette la immissione di dati nel sistema DSP ed anche l' accesso al bus VME effettuato attraverso la porta A. L' accesso al bus VME è di vitale importanza per il fatto che rende possibile il trasferimento dei coefficienti, nella fase di inizializzazione del programma, dal controllore VME Sparc al banco di memoria dedicato agli stessi attraverso la porta C. In particolare questo avviene trasferendo i coefficienti dal controllore al buffer relativo alla porta A, poi con una operazione RAWC (Read A Write C) si trasferiscono dal buffer di A a quello di C. E' anche utile perchè rende possibile, se richiesto, il trasferimento dei dati elaborati dalla memoria di A verso il bus VME e quindi, verso l' esterno per scopi di post processing od altro. La parte, comunque, più critica rimane quella relativa al buffer di ingresso dei dati alla porta Q, perchè è anche dalla velocità con cui questa riesce a fornire i dati al sistema DSP che dipendono le prestazioni di tutto il sistema: sarebbe inutile disporre di un potentissimo e veloce DSP se poi questo deve passare gran parte del suo tempo ad aspettare o che gli venganno passati i dati per la elaborazione o che i risultati stessi delle elaborazioni vengano trasferiti verso l' esterno con velocità non compatibili

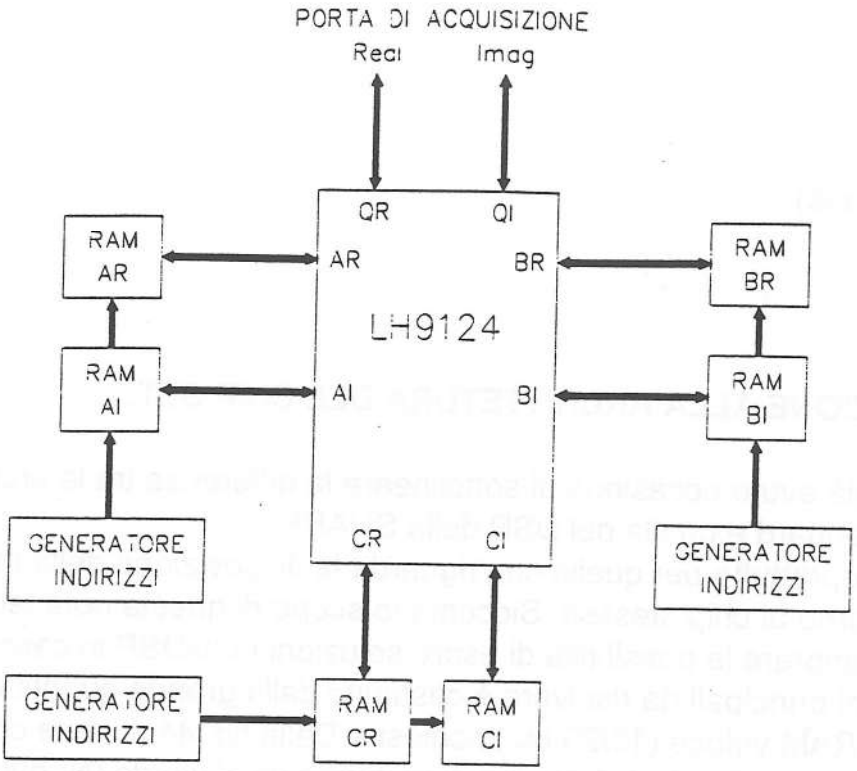


Figura 44 Configurazione tipica di un sistema utilizzante l'LH9124.

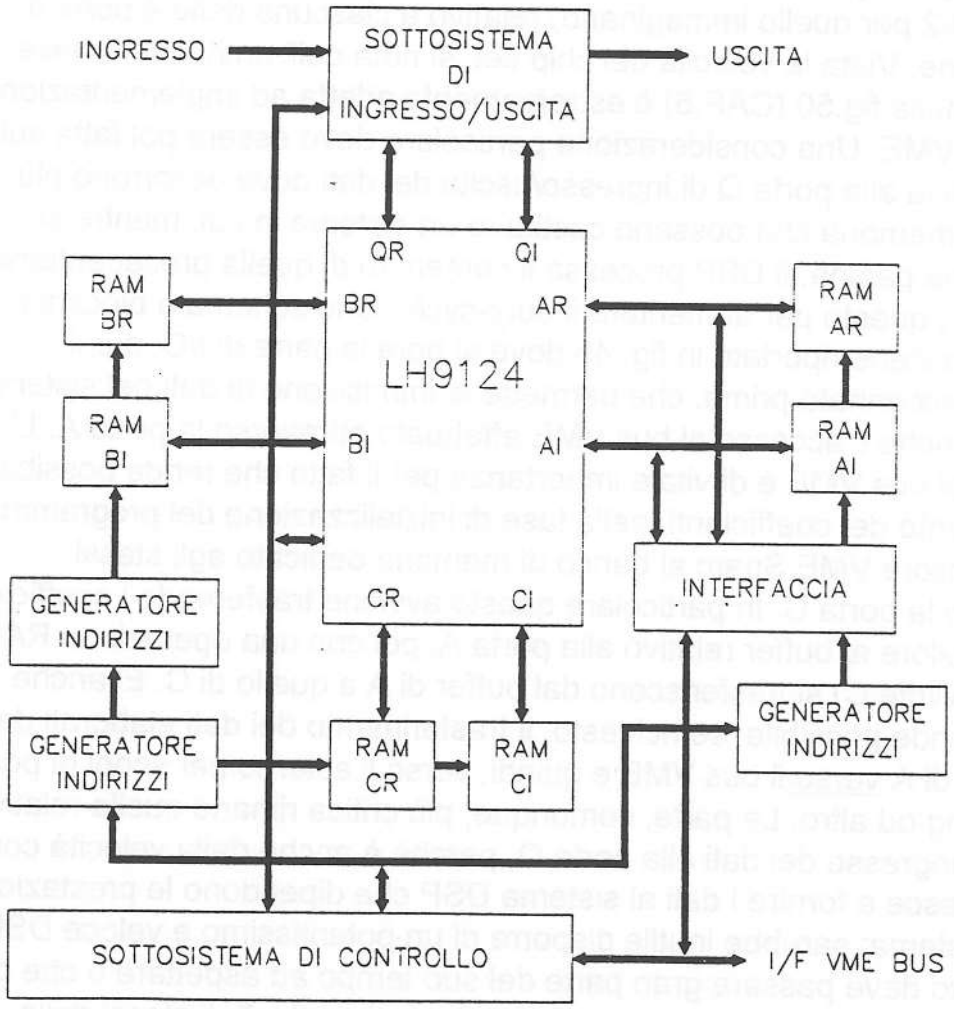


Figura 45 Schema a blocchi dell'architettura scelta per il nostro sistema.

con quelle di elaborazione del DSP stesso. Siccome l' obiettivo di questa nota tecnica, oltre che introdurre il chip set SHARP, è quello di esaminare la possibilità di progettare una scheda multiprocessore (4 o 6 LH9124 con buffers da 1 M), inizieremo ad esaminere, almeno nelle linee essenziali, questo blocco di ingresso. Questo deve essere il più "intelligente" possibile nel senso che deve essere in grado di sollevare il DSP dalle operazioni di I/O in modo tale da potere sfruttare tutta la grande potenza, esclusivamente per il calcolo evitando sprechi e perdite di tempo in attese inutili. Esamineremo ora alcune possibili architetture tra cui poi scegliere la più adatta. La prima soluzione che viene in mente, è quella di usare per le due pagine di memoria di I/O FIFO con velocità adeguata, visto che questo blocco si può ritenere come un blocco di sincronizzazione dei dati tra due sistemi, quello di acquisizione e quello di calcolo, che hanno velocità diverse come indicato in fig.46. Qui si notano due banchi di FIFO, il primo per sincronizzare i dati di ingresso ed il secondo per quelli di uscita e costituirebbe una soluzione molto lineare ed elegante per il fatto che non si rende necessaria dall' esterno, nessun tipo di generazione di indirizzi. In questo caso, per come sono strutturate internamente le FIFO, non occorre nessun indirizzamento ed il clock di sincronizzazione potrebbe essere quello ottenuto dall' address generator della porta "destinazione" A o B. Se da un lato la mancanza della necessità di indirizzamento semplifica la progettazione ed una eventuale realizzazione di un tale stadio, dall' altro limita le potenzialità per il fatto che i dati devono essere per forza letti nell' ordine naturale con cui sono stati scritti dal blocco A/D. Si dovrebbe, quindi, trasferire i dati dalla FIFO al buffer A o B con tutti i problemi di tempo richiesto indotti dal fatto che si devono trasferire i dati a blocchi di dimensioni pari alla metà della capacità della FIFO stessa ed allocandoli poi nello stesso ordine nella memoria della porta destinazione A o B. Questa che apparentemente potrebbe sembrare una operazione facile per l' LH9320, si rivela una procedura molto macchinosa per il fatto che, strano a dirsi, l' address generator non prevede questo tipo di indirizzamento a blocchi per cui dovrebbe essere riprogrammato ad ogni trasferimento di blocco. Un altro parametro di valutazione che rende sconsigliabile l' uso di FIFO in questo blocco, è che in commercio è possibile trovare dispositivi con capacità massime di 8Kx9bit. Scartata l' idea di usare FIFO in questo blocco, la prima soluzione che viene alla mente, è quella che vede le due FIFO del caso precedente sostituite da altrettanti banchi di memoria SRAM ciascuno gestito dal proprio generatore di indirizzi (del tutto simili ai banchi relativi alle porte A,B, e C) come appare in fig. 47. La circuiteria necessaria in questo caso, è molto più complessa ma i vantaggi ottenuti dal fatto che ora è possibile "prelevare" i dati con la sequenza richiesta dagli algoritmi della FFT e le capacità di memoria enormemente superiori, la rendono preferibile alla precedente. C' è comunque un aspetto negativo in questa soluzione: mentre si trasferiscono i dati dal buffer al DSP, non è possibile acquisire i dati che continuano ad arrivare dal convertitore A/D e di conseguenza questi

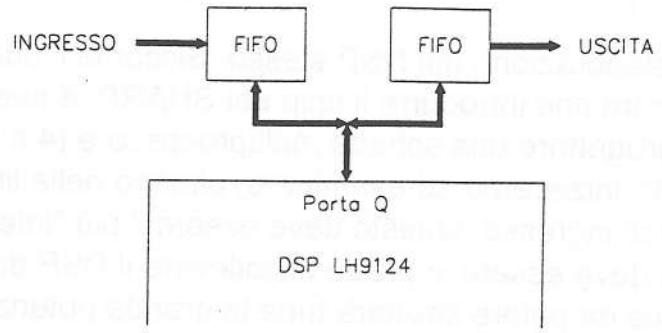


Figura 46 Esempi delle possibili architetture della sezione di I/O: soluzione con memorie FIFO.

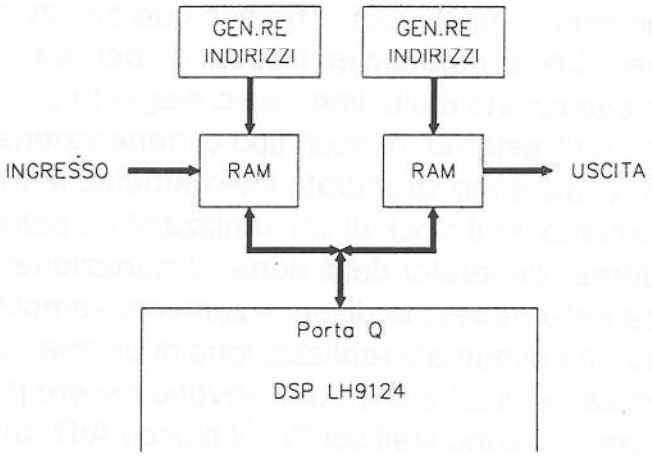


Figura 47 Esempi delle possibili architetture della sezione di I/O: soluzione con due banchi di memoria RAM.

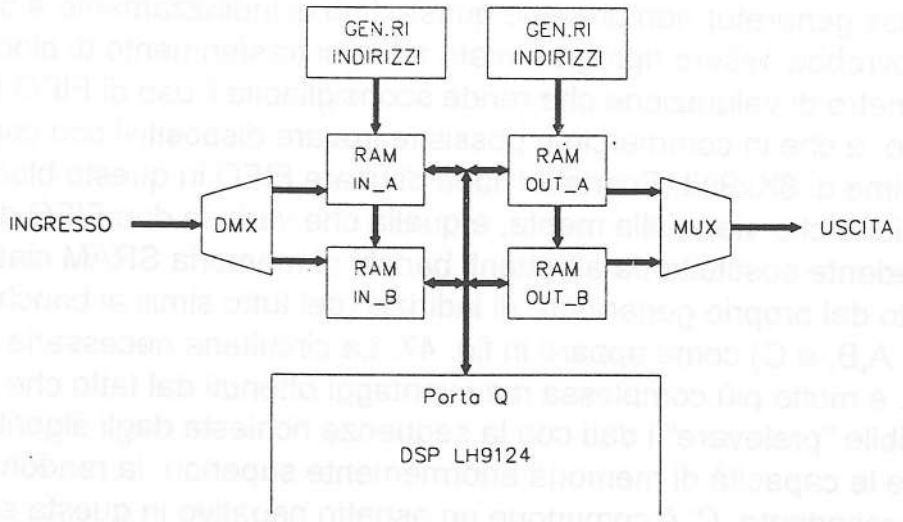


Figura 48 Esempi delle possibili architetture della sezione di I/O: soluzione con due banchi di memoria RAM per l'ingresso e due banchi per l'uscita.

ultimi vengono persi. La soluzione più ovvia a questo punto è quella di sostituire ai due banchi di memoria due buffer costituiti da due pagine ciascuno, su cui alternativamente scrivere e leggere come in fig.48. Qui, come si può ben notare, occorrono 4 generatori di indirizzi, uno per ogni buffer di memoria. Nel blocco di ingresso/uscita ogni generatore indirizza alternativamente i due blocchi di memoria ed inoltre, ciascuno di questi generatori si sincronizza con il clock esterno proveniente il primo dal sistema di acquisizione ed il secondo da quello a valle. In una configurazione del tipo di fig.48, non si ha perdita di informazione per il fatto che una volta riempita la RAM A mentre questa viene trasferita nel DSP, i dati vanno immagazzinati nella RAM B; nello stesso tempo allo step successivo mentre si trasferiscono i dati della RAM B al DSP, si riparte con una nuova scrittura dei dati in arrivo sulla RAM A, e così via come raffigurato in fig. 49. In questa figura, per non avere perdita di dati, si è fatto l' ipotesi che la frequenza a cui vengono acquisiti sia di molto inferiore alla stessa frequenza di clock (40 MHz) e che il tempo di acquisizione dell' intero buffer di dati sia maggiore del tempo totale necessario al DSP per leggere, elaborare e scrivere i dati nel buffer di uscita. Qualche considerazione deve ora essere fatta sui banchi di memoria necessari per il funzionamento del sistema. Questi buffer devono avere una larghezza di parola pari a 48 bit per sfruttare al massimo la precisione di calcolo fornibile dal DSP. L' unica eccezzione a questa regola è costituita dai doppi buffer di ingresso del blocco di I/O che sono formati da memorie atte a memorizzare i campioni che arrivano dal convertitore A/D che, nella stragrande maggioranza dei casi, lavora da 6 a 12 bit. Il sistema che ci si propone di progettare, deve essere in grado di elaborare al massimo 1 Milione di punti reali (trasformata di 2N punti reali calcolata come trasformata di N punti complessi). Considerando nello stadio di ingresso della fig. 48 i due b e IN_B, si nota che per potere lavorare correttamente devono essere almeno da 512 KWord complesse ciascuna ed in cui potranno trovare spazio 1 MWord reali e con dimensione della word di 12 bit. I due buffer di uscita OUT_A e OUT_B, a parte la larghezza della word che in questo caso è di 24 bit, devono essere in grado di contenere almeno 512 KWord complesse. I due banchi di memoria collegati alle porte A e B, devono avere una capacità tale da potere memorizzare i risultati parziali di 512 Kword. Per quello che riguarda il banco di memoria per i coefficienti collegato alla porta C, deve essere in grado di memorizzare sia i coefficienti trigonometrici per il computo della FFT (twiddle factors) che quelli per il windowing dei dati di ingresso. Considerando il solo windowing si nota subito che se trattiamo 1 M di punti reali occorre memorizzare 1 M di punti in memoria per la memorizzazione di una intera finestra, se ne trattiamo 512 K complessi occorrono 512 Kwords complesse. In questo ultimo caso, occorre organizzare la memoria "complessa" in modo tale che i coefficienti pari della window siano disposti nella parte reale e quelli ad indice dispari in quella immaginaria. Per quello che riguarda la memorizzazione dei twiddle factors, abbiamo visto che l' LH9320 prevede la possibilità di memorizzare solo i

coefficienti appartenenti al quarto quadrante più un dato, quello relativo ai 270 gradi; in questo modo basta un buffer da (128K +1) word complesse. Questa parola addizionale per il punto a 270 gradi, complica l' organizzazione della memoria. Per la realizzazione pratica di questo banco occorre quindi, un banco da 128 K ed una piccola RAM che contiene una sola parola complessa da 48 bit o un registro da 48 bit. Si deve tenere presente che il generatore di indirizzi LH9320 del banco C, deve indirizzare sia i coefficienti della window sia quelli twiddle, per cui occorre decodificare esattamente l' indirizzo relativo al registro "solitario" del punto a 270 gradi (questo può essere implementato con una PAL programmabile). Una soluzione abbastanza elegante per ovviare alle complicazioni introdotte dalla configurazione di memoria 128 Kword+1, è quella che vede l'uso di banchi da 256 Kword che, impiegando chips ad alta densità, permette addirittura alla fine di usare meno chip anche se, di memoria, ne viene sfruttata solo una parte. Ricapitolando, la situazione banchi di memoria per una applicazione da 1 Milione di punti reali o 512 K complessi, vedrebbe l' uso (funzionamento con singolo processore) della seguente configurazione:

1- Due banchi da 512KWordx48 per la sezione di Input.
2- Due banchi da 512KxWordx48 per la sezione di output.
3- Un banco da 512KWordx48 per i risultati internedi A.
4- Un banco da 512 KWordx48 per i risultati intermedi B.
5- Un banco da 512 KWordx48 per i coefficienti della window ed un banco da 256 KWordx48 per i twiddle factors.

Particolare cura deve essere prestata nella scelta dei dispositivi di memoria tenendo conto della elevata velocita` richiesta dal sistema, lavorando con SYSCLK di 40 MHz.

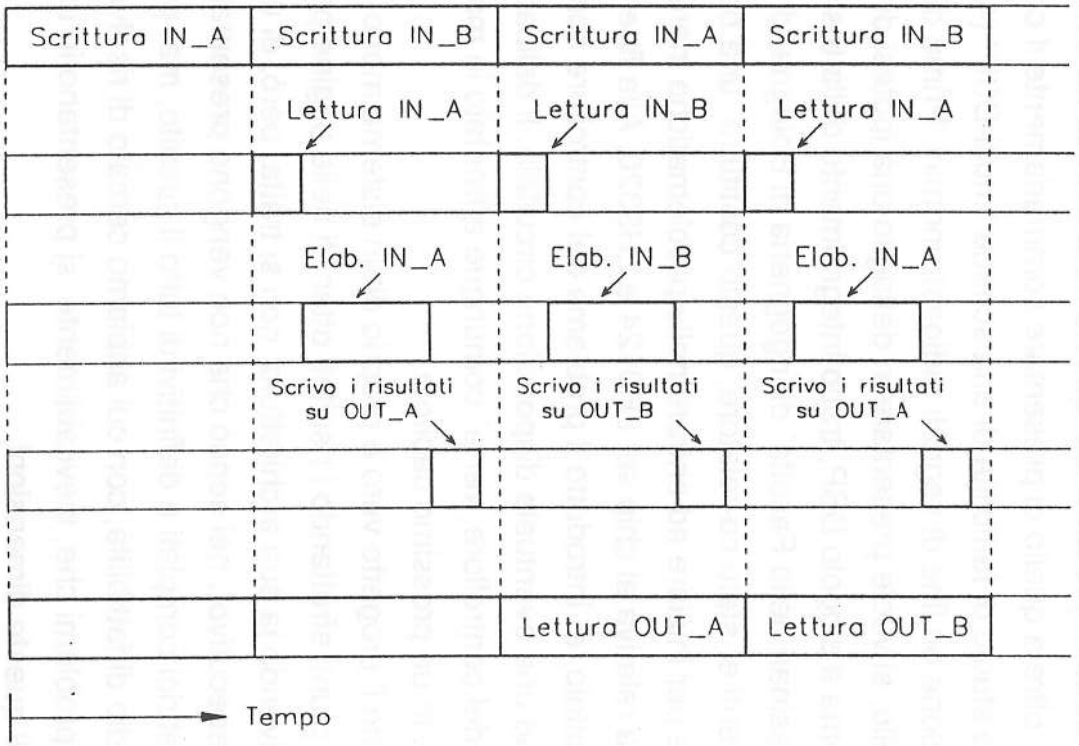


Figura 49- Rappresentazione della successione delle varie fasi del funzio namento della sezione di I/O

(file: sharp6.doc)

IPOTESI DI PROGETTO DI UN SISTEMA A SINGOLO LH9124.

Come gia`detto in precedenza, questa nota tecnica ha come scopo principale, oltre a quello di presentare sommariamente il chip set SHARP, lo studio di fattibilita`di una scheda "multi-DSP" (4 o 6) per l' elaborazione on line di segnali radioastronomici. Prima di arrivare a quel livello, si vuole presentare in dettaglio una ipotesi di progetto di un sistema a singolo DSP, tratto integralmente dalla tesi di Sergio Tassinari della Facolta` di Ingegneria di Bologna, di cui uno degli scriventi e` stato co-relatore; questo costituira` una base di lancio utile per iniziare ad entrare nelle problematiche inerenti la circuitistica relativa al chip set LH9124 e LH9320. Alla fine dello stesso capitolo, e` introdotto il problema del controllore con un accenno ad una eventuale disposizione circuitale. Il delicato problema del controllore, verra` comunque affrontato in maniera esauriente in un prossimo capitolo.
Esaminiamo il progetto vero e proprio di un sistema mono-DSP, al quale si è giunti sfruttando i risultati ottenuti nelle pagine precedenti. Pur descrivendo la sua architettura, non si tratta, però, di un progetto “esecutivo”, nel senso che non vengono presentati gli schemi elettrici completi e definitivi di tutto il circuito, ma, piuttosto, di uno studio di fattibilità, con cui abbiamo cercato di risolvere i numerosi problemi che, inevitabilmente, si presentano in un progetto di queste dimensioni.
Dopo aver scomposto il sistema in tante sottosezioni, studieremo ognuna di esse in dettaglio, analizzando, ad esempio, le varie temporizza-zioni nei punti critici e la scelta dei componenti.
Infine, presenteremo anche una delle possibili soluzioni costruttive e accenneremo agli eventuali sviluppi futuri di questo progetto.

Architettura generale

Nel capitolo precedente abbiamo già visto una schema-tizzazione dell'architettura che è stata scelta per il sistema.
Riprendiamo ora tale rappresentazione, mettendo, però, in evidenza i singoli blocchi funzionali, come si può osservare in Fig. 50
Notiamo subito, al centro, il blocco contenente il DSP LH9124, che comunica con l'esterno attraverso le quattro porte bidirezionali A, B, C e Q. I dati da elaborare ed i risultati finali vengono gestiti dal sottosistema di I/O, rispettivamente dalla sezione di ingresso e dalla sezione di uscita, che contiene la struttura a quattro banchi di memoria, ed i relativi generatori di indirizzi, analizzata nel capitolo precedente.
I risultati intermedi vengono memorizzati nei banchi A e B, collegati alle omonime porte del DSP. Il sottosistema di memoria A è, inoltre, collegato direttamente con il mondo esterno, attraverso il Bus VME, in modo da permettere il *download* dei coefficienti e per, eventualmente, elaborare dati presenti nella memoria del calcolatore *host*.

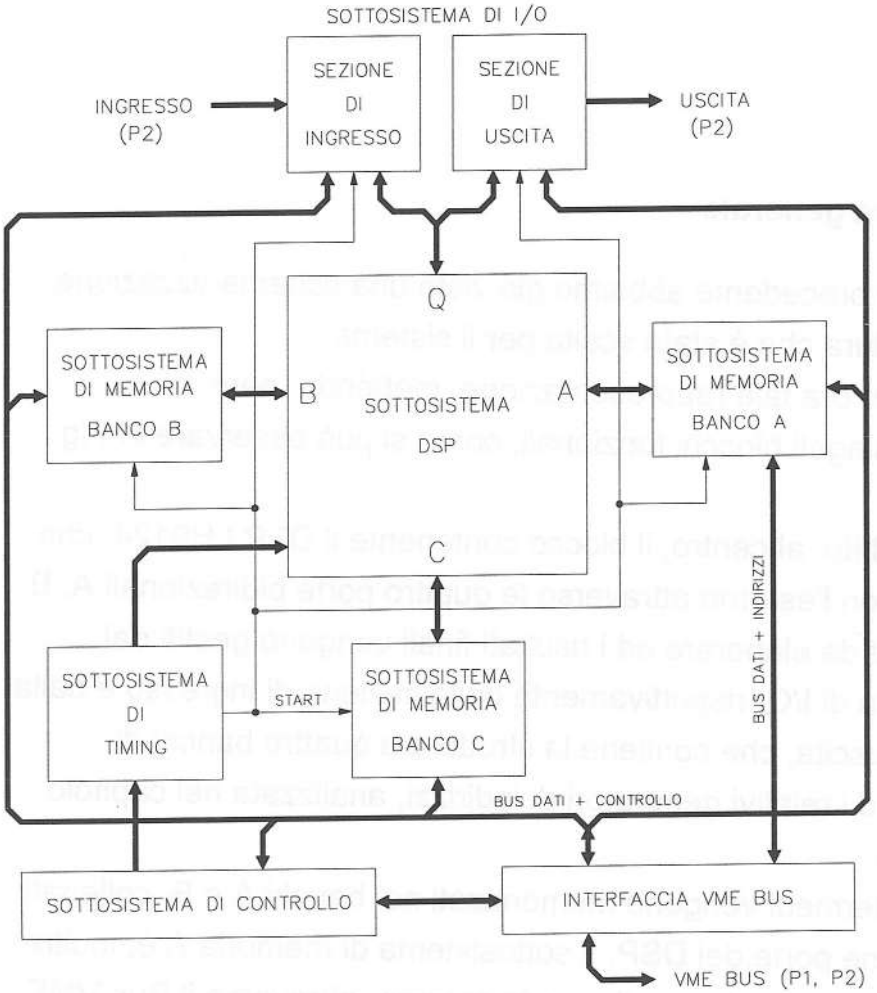


Figura 50 - Rappresentazione dell'architettura generale del nostro sistema, con i vari blocchi funzionali in evidenza.

I coefficienti, sia quelli trigonometrici per la FFT che quelli relativi al windowing dei dati, sono memorizzati nel banco C, che è, dunque, dotato di due blocchi di memoria e della logica necessaria al loro indirizzamento.

Alla gestione di tutto il complesso, provvede il sottosistema di controllo, un vero e proprio sistema a microprocessore, anche se molto limitato, che viene programmato dal computer host, attraverso l'interfaccia VME; a questa sezione spetta il compito della generazione delle istruzioni per il DSP e della "supervisione", istante per istante, dello stato di funzionamento di tutti i vari blocchi.

Tutti i dispositivi programmabili, cioè tutti i generatori di indirizzi ed il controllore di sistema, sono interfacciati direttamente con il bus VME e vengono gestiti come delle comuni periferiche.
Il sistema di controllo comprende anche il generatore di clock a 40MHz necessario per sincronizzare tutti i singoli dispositivi, e gestisce, inoltre, il sottosistema di timing, che provvede a generare tutti i segnali che sono critici dal punto di vista della temporizzazione; in particolare, esso genera il segnale START/STOP per il DSP ed un segnale di start comune per tutti i generatori di indirizzi: l'istante preciso in cui iniziare il calcolo degli indirizzi viene poi stabilito via software, programmando, in ciascun generatore, la latenza relativa ad ogni singola istruzione.

Sottosistema DSP LH9124

Questa sezione contiene, appunto, il DSP Sharp LH9124, la cui piedinatura è stata rappresentata in Fig. 51, e la poca logica necessaria a gestire i vari segnali di controllo secondari, che, per evitare confusione, abbiamo preferito analizzare separatamente.

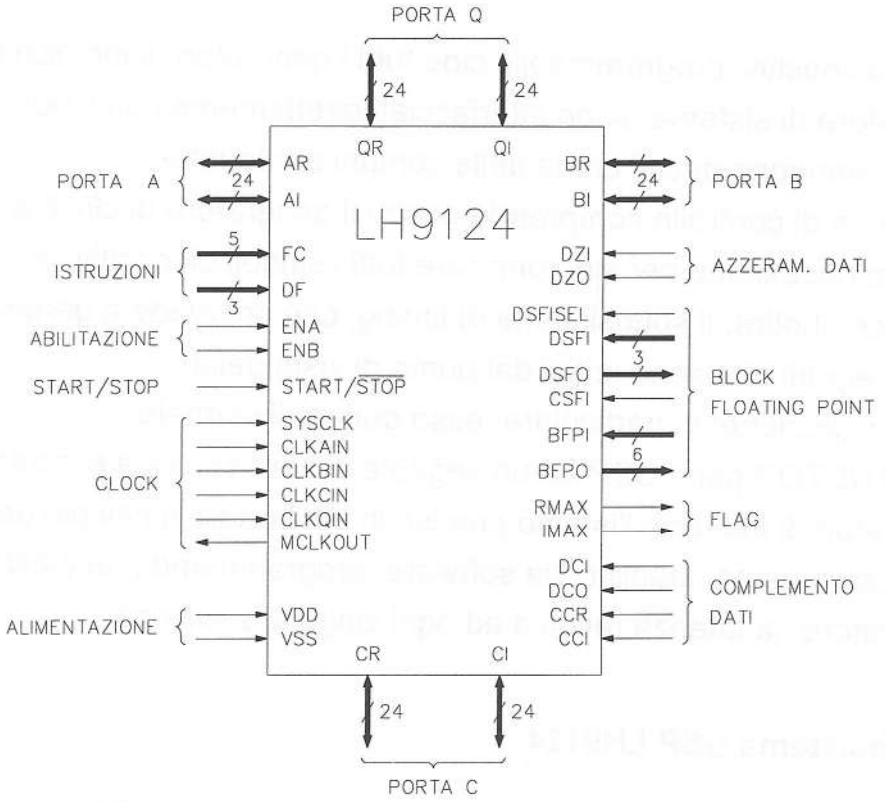


Figura 51 - Piedinatura del DSP Sharp LH9124.

Come abbiamo già visto, i quattro bus relativi alle quattro porte di I/O fanno capo ai vari sottosistemi di memoria, che analizzeremo in seguito, e che provvederanno anche a fornire i segnali di clock delle porte, cioè CLKAIN, CLKBIN, CLKCIN e CLKQIN.
I due bus relativi al codice operativo (FC[4:0]) e al flusso dati (DF[2:0]) di ogni istruzione vengono gestiti dal sottosistema di controllo, mentre le due linee di abilitazione e quella di START/STOP vengono governate dal sistema di timing.
Rimangono, dunque, da controllare tutti i segnali relativi alla gestione dell'aritmetica in Block Floating-Point e quelli per complementare e azzerare i dati.
Per quanto riguarda i primi, esaminiamo, anzitutto, le linee DSFISEL, DSFI e DSFO.
Come abbiamo visto in precedenza, è possibile gestire lo scaling dei dati in ingresso al passo corrente sia in modo manuale, impostando il numero di posizioni da scalare (al massimo 7) sulle

linee DSFI[2:0] e ponendo DSFISEL=1, oppure in modo automatico, impostando DSFISEL=0.
In quest'ultimo modo di funzionamento, ad ogni passo viene calcolato il fattore di scala più adatto per il passo successivo: lo stato delle linee DSFO[2:0] rappresenta, appunto, questo fattore, che deve essere riportato agli ingressi DSFI affinchè il meccanismo di scaling automatico funzioni correttamente.
E' necessario prevedere entrambi i modi di funzionamento, in quanto, almeno per il primo passo di una elaborazione, cioè quando non esiste una storia precedente su cui basarsi, occorre scalare i dati in modo manuale. Per i passi successivi al primo, invece, è consigliabile affidarsi al sistema automatico.
Per poter operare in entrambi i modi si può pensare di gestire queste linee con una rete del tipo di quella di Fig. 58, in cui un multiplexer (MUX) provvede a presentare agli ingressi DSFI: o un fattore di scala definito dall'utente, impostando le linee USDSFI[2:0], oppure il fattore di scala letto alle uscite DSFO, il tutto in base allo stato del segnale DSFI. Le linee USDSFI[2:0] devono essere gestite dal sistema di controllo, mentre il multiplexer può essere realizzato, molto semplicemente, con un dispositivo TTL 74xxx157 [20] [21], ad esempio della serie LS o HCT, visto che non servono, in questo caso, velocità particolarmente elevate.

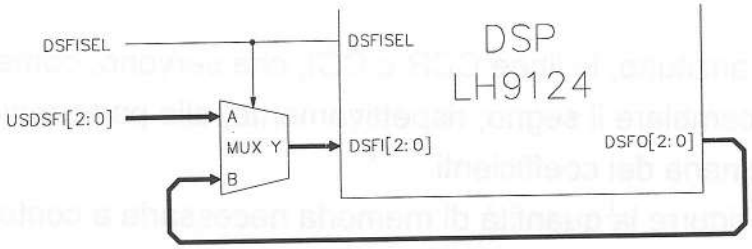


Figura 52 - Rete di gestione del fattore di scala dei dati in ingresso.

Sempre in riferimento alla gestione del Block Floating-Point ed a quanto gia` visto , occorre prevedere il collegamento delle linee BFPO agli ingressi BFPI, in modo che il DSP possa tenere traccia

del fattore di scala complessivo. Per lo stesso motivo, è indispensabile poter leggere lo stato di queste linee dall'esterno, per conoscere l'*esponente* dei risultati in uscita.
In Fig. 53 è mostrato il semplice collegamento di queste linee; analogamente alla situazione precedente, sarà il sottosistema di controllo a gestire la lettura delle linee BFPO[5:0].

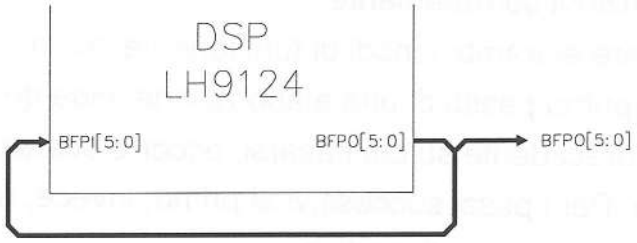


Figura 53 - Schema di collegamento delle linee BFPI e BFPO.

Per concludere il discorso sui segnali di controllo del Block Floating-Point, rimane da considerare la sola linea CSFI, la quale, quando è attivata, permette di scalare i coefficienti di una posizione a destra. Anche tale linea dovrà essere gestita direttamente dal sistema di controllo.
La gestione dei segnali di complementazione e azzeramento dei dati risulta, invece, un poco più complessa, perchè molte sono le possibilità che il chipset Sharp mette a disposizione a questo scopo.
Analizziamo, anzitutto, le linee CCR e CCI, che servono, come sappiamo, a cambiare il segno, rispettivamente, alla parte reale e a quella immaginaria dei coefficienti.
Allo scopo di ridurre la quantità di memoria necessaria a contenere i coefficienti, sfruttando il meccanismo visto nei capitoli precedenti , queste linee devono poter essere gestite direttamente dal relativo generatore di indirizzi (del banco C), ma, anche, dall'esterno, in modo da garantire la possibilità di fissare a priori lo stato di questi segnali, cosa che potrebbe essere utile in particolari applicazioni.

Anche in questo caso, dunque, sarà necessario prevedere un multiplexer, come si vede in Fig.54.

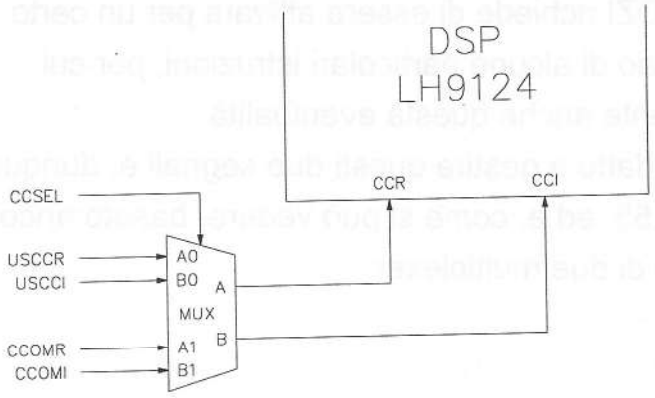


Figura 54 - Schema di collegamento dei segnali CCR e CCI.

Le linee CCSEL, USCCR e USCCI verranno gestite, come al solito, dal sistema di controllo, mentre CCOMR e CCOMI sono pilotate dalle omonime uscite del generatore di indirizzi del banco C. In questo caso è consigliabile utilizzare un dispositivo veloce, in modo da rispettare i tempi di setup e hold relativi agli ingressi CCR e CCI, come, ad esempio, un 74F157 oppure un 74AS157 [20] [21].
Per quanto riguarda le linee DZI e DZO, che controllano, rispettivamente l'azzeramento dei dati in ingresso o in uscita, occorre prevedere un numero ancora maggiore di possibilità.
Infatti, per garantire una grande flessibilità al sistema, è necessario, poter gestire questi segnali attraverso le uscite PO dei generatori di indirizzi del banco A o del banco B, oltre, ovviamente, alla possibilità di fissare a priori il loro stato. L'uscita PO di ogni generatore di indirizzi può essere programmata per attivarsi, ad esempio, per un certo numero di cicli prima dell'inizio di una sequenza oppure dopo la sua fine, oppure ancora, per un certo numero di cicli ogni n indirizzi generati. Collegando queste linee agli ingressi DZI o DZO diviene dunque possibile eseguire decimazioni, interpolazioni e riempimenti con zeri di sequenze di dati.

E' sufficiente considerare le uscite PO dei soli generatori relativi ai banchi A e B, in quanto tali banchi costituiscono, invariabilmente, sorgente oppure destinazione dei dati.
Inoltre, la sola linea DZI richiede di essere attivata per un certo numero di cicli all'inizio di alcune particolari istruzioni, per cui occorre tenere presente anche questa eventualità.
Il circuito completo adatto a gestire questi due segnali è, dunque, rappresentato in Fig.55, ed è, come si può vedere, basato ancora una volta sull'utilizzo di due multiplexer.

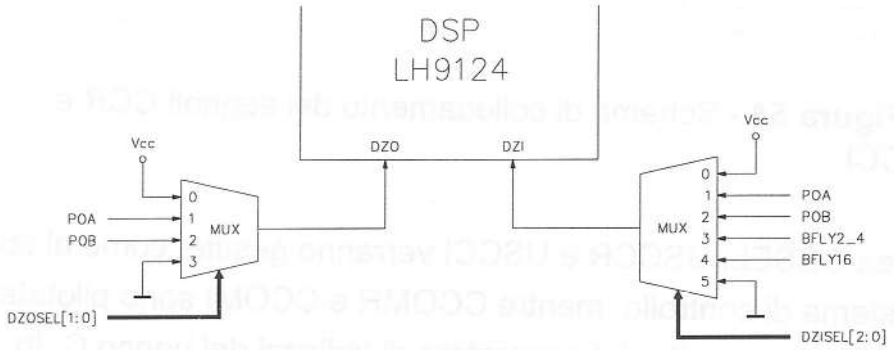


Figura 55 - Schema di collegamento dei segnali DZI e DZO.

Il multiplexer che controlla l'ingresso DZO può essere realizzato con un dispositivo 74F153 o 74AS153, mentre per quello sulla linea DZI si può utilizzare un 74F151 o 74AS151 [20] [21].
Attraverso le linee DZOSEL[1:0] diviene, così, possibile agire sullo stato dell'ingresso DZO secondo lo schema rappresentato nella tabella che segue:

DZOSEL[1]	DZOSEL[0]	STATO
0	0	DZO=1
0	1	DZO=PO_A
1	0	DZO=PO_B
1	1	DZO=0

Stato dell'ingresso DZO in base alla configurazione delle linee di controllo DZOSEL[1:0].

Per quanto riguarda, invece, la linea DZI, è possibile programmare il suo stato attraverso i segnali DZISEL[2:0], come risulta dalla seguente tabella:

DZISEL[2]	DZISEL[1]	DZISEL[0]	STATO
0	0	0	DZI=1
0	0	1	DZI=PO_A
0	1	0	DZI=PO_B
0	1	1	DZI=BFLY2_4
1	0	0	DZI=BFLY16
1	0	1	DZI=0

Stato dell'ingresso DZI in base alla configurazione delle linee di controllo DZISEL[2:0].

Le linee DZOSEL e DZISEL dovranno, ovviamente, essere gestite dal sistema di controllo, mentre i segnali denominati BFLY2_4 e BFLY16 sono generati dal sottosistema di timing e devono essere utilizzati qualora si eseguano istruzioni di tipo BFLY2 o 4 oppure BFLY16, che richiedono, rispettivamente, che la linea DZI sia attiva durante i primi 4 od i primi 16 cicli di clock.
Abbiamo dunque mostrato come possono essere gestite le linee di controllo secondarie del DSP. All'appello mancano solo i segnali DCO e DCI che possono essere pilotati direttamente dal controller di sistema, e le uscite Rmax e Imax, che, invece, possono essere portate direttamente all'esterno.

Sottosistema di temporizzazione

Nei capitoli precedenti abbiamo gia` esaminato la tipica temporizzazione dei segnali di controllo e dei dati relativi al DSP

LH9124, in una configurazione classica, del tutto simile a quella che stiamo studiando.
Tale temporizzazione è stata presentata inp precedenza ed in essa non si considerava il timing relativo ai vari generatori di indirizzi, perchè non erano ancora stati trattati. Ora che anche questa "lacuna" è stata colmata, possiamo presentare, in Fig.56 un diagramma completo delle principali temporizzazioni da rispettare nel sistema che stiamo considerando.
Come è facile notare, rispetto alla precedente diagramma di timing, sono state fatte delle piccole ma sostanziali aggiunte: la traccia relativa al segnale di start dei generatori di indirizzi (AGSTART) e quelle relative agli indirizzi stessi.

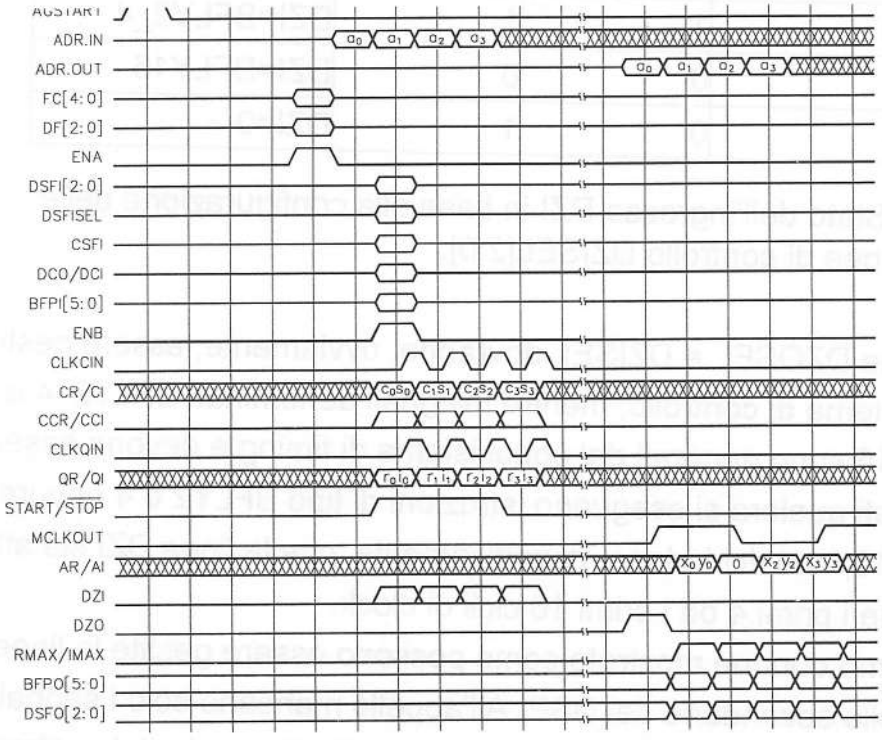


Figura 56 - Temporizzazione di dati e segnali di interfaccia e controllo nel sistema in esame.

Dal grafico notiamo, anzitutto, i 5 cicli di latenza del generatore di indirizzi (il ritardo tra l'istante in cui viene campionato il segnale di start e quello in cui si ottiene il primo indirizzo valido), oltre al ritardo tra l'istante in cui diventa disponibile il primo indirizzo e quello in cui

appare il primo dato valido (il tempo di accesso della memoria).
Tutto il resto è identico alla precedente diagramma di timing, per cui ritroviamo le medesime temporizzazioni per i segnali START/STOP, ENA ed ENB.
Si presenta dunque il problema di generare i vari segnali di controllo in modo che siano rispettate queste precise relazioni temporali e che venga mantenuto il sincronismo con il clock di sistema, che, vale la pena di ripeterlo, "viaggia" alla frequenza di ben 40MHz.
Il sistema di controllo, essendo, come vedremo, un sistema basato su di un microcontrollore, non è assolutamente in grado di gestire i segnali in questione ad una tale velocità, per cui diventa indispensabile utilizzare un circuito dedicato a questo scopo.

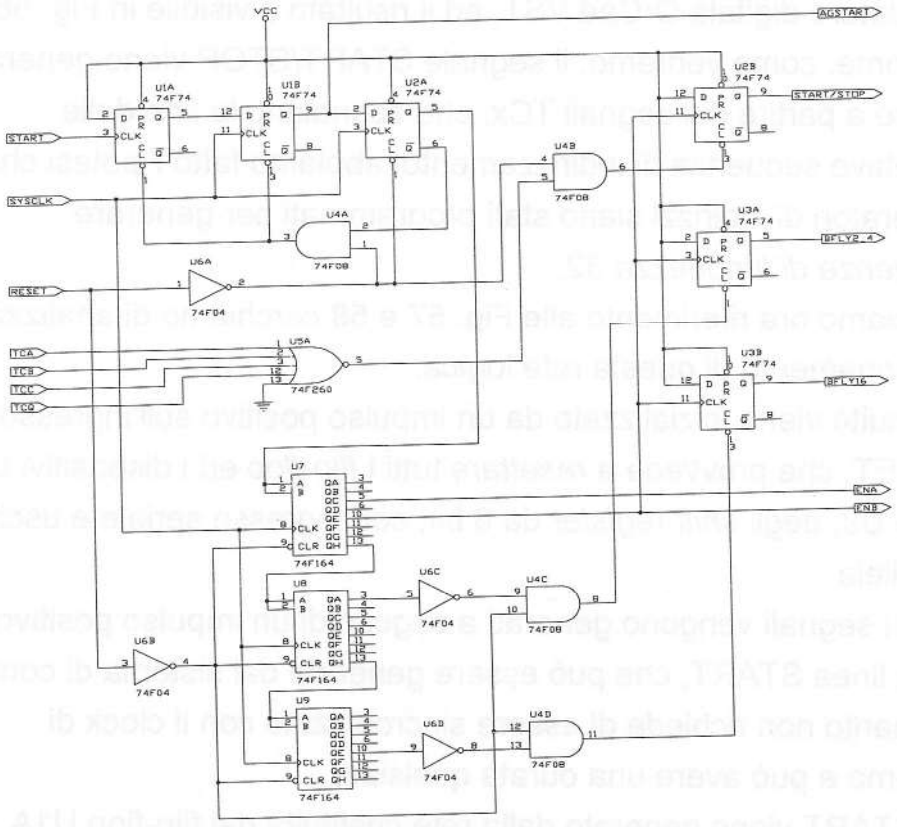


Figura 57 - Schema elettrico completo di una delle possibili reti logiche in grado di generare i segnali di abilitazione e controllo del DSP e dei relativi generatori di indirizzi.

In Fig.57 appare una semplice (è stata progettata "a mano"...) rete logica capace di generare i segnali di controllo necessari, secondo le speci-fiche della Sharp.
Riceve in ingresso il segnale di clock di sistema SYSCLK, il segnale di RESET, un segnale di START generale, prodotto dal sistema di controllo e le quattro linee TCA, TCB, TCC e TCQ, pilotate dalle uscite TC dei corrispondenti generatori di indirizzi, e produce in uscita il segnale AGSTART, che serve ad avviare i generatori di indirizzi (tutti), il segnale START/STOP, che avvia il processo di elaborazione dei dati, i due segnali, ENA e ENB, di abilitazione dei registri interni al DSP ed, infine, i due *flag* BFLY2_4 e BFLY16, utili a pilotare l'ingresso DZI dell'LH9124 durante i primi cicli di esecuzione delle istruzioni BFLYxx.
Questo circuito è stato simulato al calcolatore, mediante il simulatore digitale OrCad VST, ed il risultato è visibile in Fig. 58
Siccome, come vedremo, il segnale START/STOP viene generato *anche* a partire dai segnali TCx, che segnalano la fine delle rispettive sequenze di indirizzamento, abbiamo fatto l'ipotesi che i generatori di indirizzi siano stati programmati per generare *sequenze di lunghezza 32*.
Facciamo ora riferimento alle Fig. 57 e 58 cerchiamo di analizzare il funzionamento di questa rete logica.
Il circuito viene inizializzato da un impulso positivo sull'ingresso RESET, che provvede a *resettare* tutti i *flip-flop* ed i dispositivi U7, U8 e U9, degli *shift register* da 8 bit, con ingresso seriale e uscita parallela.
Tutti i segnali vengono generati a seguito di un impulso positivo sulla linea START, che può essere generato dal sistema di controllo in quanto non richiede di essere sincronizzato con il clock di sistema e può avere una durata qualsiasi.
AGSTART viene generato dalla rete costituita dai flip-flop U1A, U1B e U2A e dalla porta AND U4A, rete che provvede anche a pilotare, con un singolo impulso, la catena di shift register.

Il flip-flop U1A è comandato sull'ingresso di clock direttamente dal segnale START ed ha l'ingresso dati tenuto a livello alto, per cui la sua uscita Q diviene immediatamente attiva in corrispondenza del fronte di salita di START.

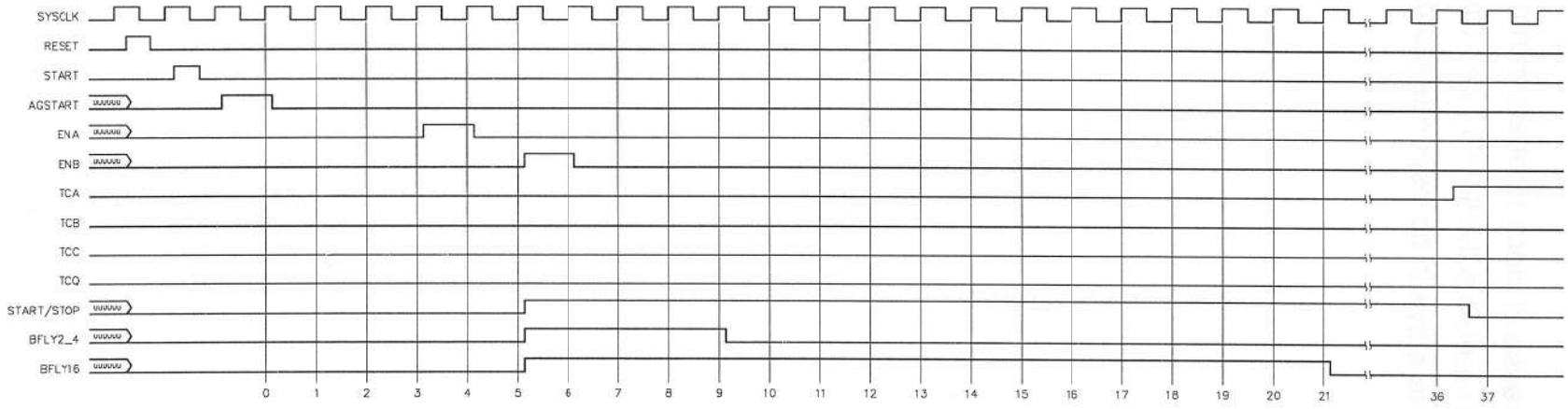

SYSCLK
RESET
START
AGSTART
ENA
ENB
TCA
TCB
TCC
TCQ
START/STOP
BFLY2_4
BFLY16
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
36
37

Figura 58 - Risultato della simulazione con il programma OrCad VST della rete logica di Fig. 63.

Al successivo fronte di salita di SYSCLK, tale uscita viene campionata, per cui la linea AGSTART diviene attiva e lo rimane fino al clock successivo, quando il flip-flop U1B viene resettato da U2A e U4A.

Abbiamo dunque generato l'impulso di start per i generatori di indirizzi, che partono tutti insieme al ciclo 0 (vedi Fig. 58).

Il generatore che indirizza la memoria sorgente dovrà, ovviamente, partire appena possibile, mentre quello che pilota la memoria di destinazione deve iniziare a produrre la sua sequenza solo dopo che sia trascorsa la latenza del primo generatore e quella del DSP (vedi Fig. 56): questa attesa può essere programmata in ogni generatore di indirizzi e per ognuna delle sue istruzioni, per cui è possibile utilizzare un segnale di start unico per tutti i generatori.

A questo punto, il generatore di indirizzi relativo alla memoria sorgente è stato avviato, ed il DSP può iniziare ad elaborare i dati non appena il primo di questi diviene disponibile. Ciò avviene 5+1 cicli dopo che AGSTART è stato campionato: 5 cicli di latenza del generatore di indirizzi più 1 dovuto al tempo di accesso della RAM, che non deve essere maggiore di 20ns.

Al ciclo 5, quindi, può essere attivata la linea START/STOP, da parte del flip-flop U2B, il quale è pilotato dall'uscita QE (la quinta) del primo shft register.

La linea START/STOP deve, poi, rimanere attiva fino al campionamento del penultimo dato, cioè fino a quando l'uscita TC del generatore di indirizzi che pilota la memoria sorgente passa a livello alto, il che avviene, appunto, un ciclo prima della fine della sequenza corrente.

Le uscite QA e QE, del secondo e del terzo shift register, provvedono a resettare i flip flop U3A e U3B, le cui uscite si attivano contempora-neamente a START/STOP, rispettivamente dopo 4 e dopo 16 cicli di clock, in modo da generare i segnali utili al

pilotaggio dell'ingresso DZI del DSP durante i primi cicli dell'esecuzione delle istruzioni BFLY2 e 4 e BFLY16.
Infine, due delle uscite di U7 vengono sfruttate per ricavare i segnali di abilitazione ENB, contemporaneo a START/STOP, in corrispondenza del primo dato, ed ENA, due cicli prima, come da specifiche dell'LH9124.
Per concludere, notiamo come le uscite TC dei vari generatori di indirizzi, siano combinate tra loro per mezzo di una porta NOR: la disattivazione del segnale START/STOP sarà, quindi, provocata dal primo, tra tutti i generatori di indirizzi, che terminerà la propria sequenza. Dato che tutti i generatori saranno programmati per produrre sequenze della stessa lunghezza, il primo a terminare sarà, ovviamente, quello che è partito per primo, cioè proprio il generatore che pilota la memoria sorgente, come richiesto.
Anche, e soprattutto, in questo circuito è necessario utilizzare dispositivi TTL delle serie "veloci", come, ad esempio, la serie F oppure la serie AS, le quali possono funzionare correttamente fino a frequenze prossime ai 100MHz.

Sottosistema di memoria - Banco B

Continuiamo nell'analisi dello schema a blocchi di Fig. 50, esaminando, tra tutti i banchi di memoria, per primo quello che fa capo alla porta B, perchè è il più semplice in assoluto.
Infatti, come appare evidente dall'esame del suo schema, in Fig. 59, esso è costituito dalla sola memoria e dal relativo generatore di indirizzi, collegati tra loro in modo "canonico".

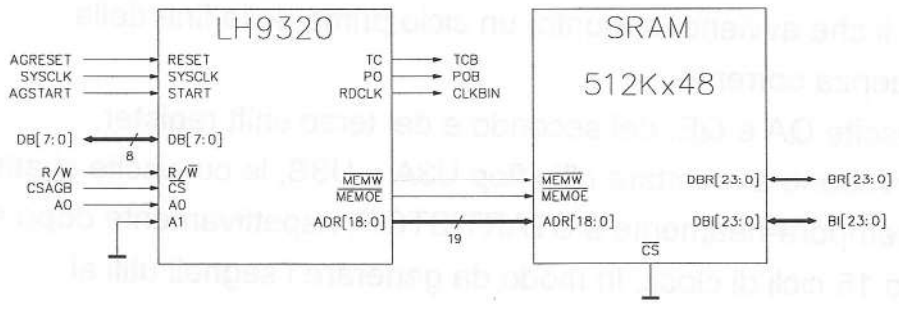


Figura 59 - Schema del sottosistema di memoria, banco B.

Il generatore di indirizzi LH9320 pilota direttamente la memoria con il proprio bus indirizzi e con le uscite MEMW# e MEMOE#; la sua uscita RDCLK comanda il campionamento dei dati alla porta B del DSP (cioè pilota direttamente l'ingresso CLKBIN), mentre le linee TC e PO raggiungono, rispettivamente, la rete di temporizzazione appena vista ed i multiplexer che gestiscono gli ingressi DZI e DZO del DSP (vedi Fig. 55).
Infine, la sua interfaccia di programmazione (DB[7:0], CS#, R/W e A0) sono collegati direttamente al controller VME, che lo gestisce come una periferica qualsiasi: il bus dati e le linee R/W e A0 sono quindi comuni a tutti gli altri generatori di indirizzi presenti, ed è il solo segnale CS#, sempre generato dal controller VME, a stabilire quale, tra essi, deve essere abilitato a colloquiare con il sistema host.
Per quanto riguarda la memoria, notiamo, anzitutto, che può essere sempre tenuta abilitata, ponendo il suo pin CS# direttamente a massa.

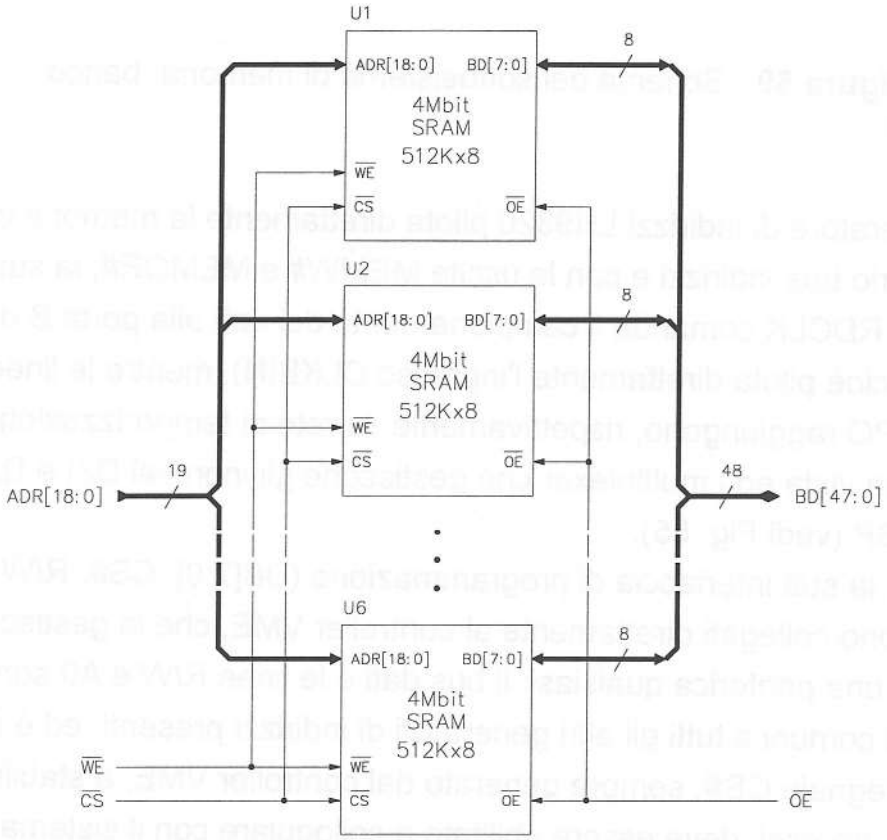


Figura 60 - Schema di una possibile realizzazione del banco di memoria da 512Kx48, che sfrutta i nuovi chip RAM da 512Kx8 (4Mbit).

Si tratta, come abbiamo visto alla fine del capitolo precedente, di in banco di RAM statica, dalla capacità di 512K parole di 48 ognuna, cioè, in altri termini, da ben 3Mbyte, e con un tempo di accesso che non deve essere superiore ai 20ns, il che giustifica la scelta della memoria di tipo *statico*.

Sebbene siano da tempo disponibili chip SRAM (Static RAM) della capacità di 1Mbit e con le caratteristiche richieste, si è pensato di sfruttare, in questo progetto, i nuovi componenti dalla densità quadrupla (4Mbit) che molti costruttori stanno annunciando in questo periodo (fine 1994), e che *non* sono ancora disponibili in grossi volumi.

Infatti, come si può osservare in Fig.60, grazie a questi componenti sarà presto possibile realizzare il banco da 512Kx48 con solo sei dispositivi di tipo 512Kx8bit, come, ad esempio il chip KM684002,

annunciato dalla Samsung e disponibile, probabilmemte, nei primi mesi del 1995, nelle versioni con tempo di accesso di 15, 20 e 25ns.

Sottosistema di memoria - Banco C

Lo schema del sistema di memoria dedicato ai coefficienti è rappresentato in Fig. 61. Come si può facilmente notare, esso è quasi identico a quello appena esaminato.

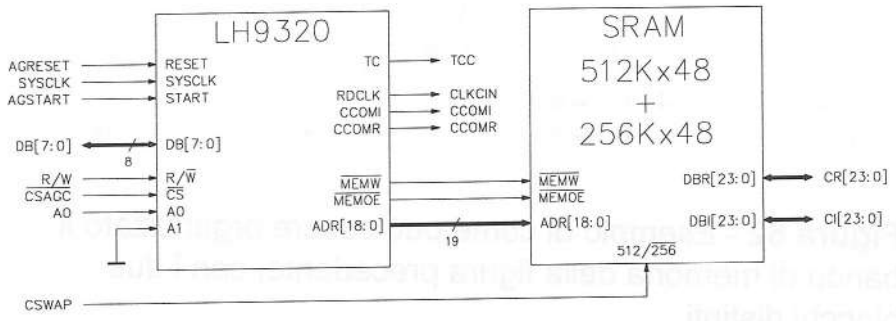


Figura 61 - Schema del sottosistema di memoria, banco C.

Le uniche differenze sono costituite dall'uso delle uscite CCOMR e CCOMI, che nel caso precedente rimanevano libere e che sono dirette al multiplexer che pilota gli ingressi CCR e CCI del DSP (vedi Fig. 54) e dalla messa in opera di un banco di memoria suddiviso in due blocchi, le cui dimensioni derivano dalle considerazioni fatte nel capitolo precedente.
Il blocco più grande, da 512Kx48, è destinato a contenere i coefficienti di pesatura dei dati (fino a 512K complessi o 1M reali), mentre quello più piccolo, da 256Kx48, serve per i coefficienti trigonometrici: si può, quindi, memorizzare un intero set di coefficienti (360°) per FFT complesse fino a 256K punti (o, che è la stessa cosa, 512K punti reali), oppure si può memorizzare solo un quarto di questi coefficienti (90°) qualora si desideri calcolare FFT complesse fino a 512K punti (o 1M punti reali). Questo banco di

memoria può essere organizzato secondo lo schema che appare in Fig. 62.

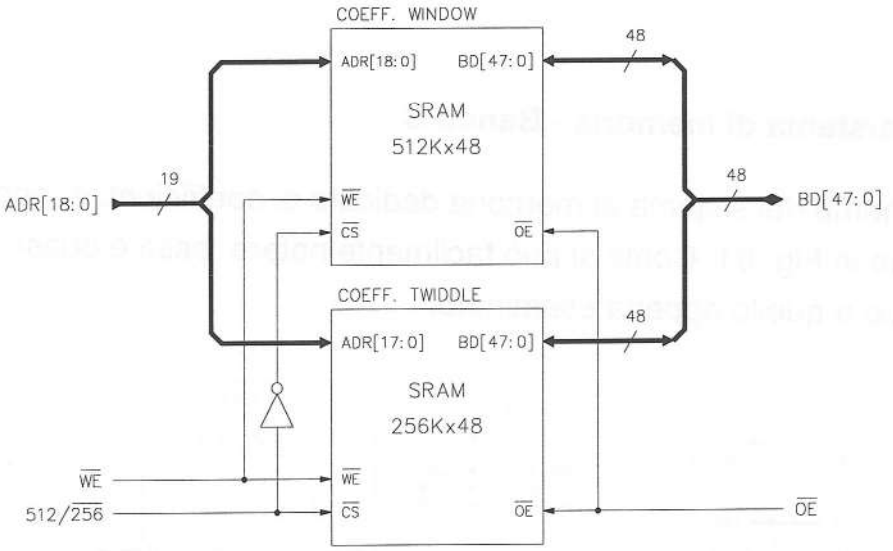


Figura 62 - Esempio di come può essere organizzato il banco di memoria della figura precedente, con i due blocchi distinti.

Come si vede, le linee di controllo ed il bus degli indirizzi sono comuni ai due blocchi, anche se, ovviamente, il blocco più piccolo sfrutta solo 18 dei 19 bit di indirizzamento disponibili.
Il passaggio da un blocco all'altro viene gestito dal sistema di controllo, attraverso l'opportuno pilotaggio delle linee Chip Select dei due blocchi: quando CSWAP (cioè 512/256#) è a livello logico alto, viene selezionato il blocco da 512K, mentre quando è a livello logico basso è il blocco più piccolo ad essere abilitato.
Mentre il blocco maggiore è, ovviamente, identico in tutto e per tutto a quello di Fig. 60, la sezione da 256K può essere realizzata secondo lo schema di Fig. 63, dove abbiamo, ancora, considerato l'uso di dispositivi da 4Mbit, come nel caso precedente, ma dalla diversa organizzazione.
Si tratta, infatti di memorie da 256Kx16, come, ad esempio, il tipo KM6164002, pure annunciato dalla Samsung per i primi mesi del 1995.

Grazie a questi dispositivi è possibile realizzare il blocco di memoria in questione con soli 3 componenti, come appare evidente dalla Fig. 63.

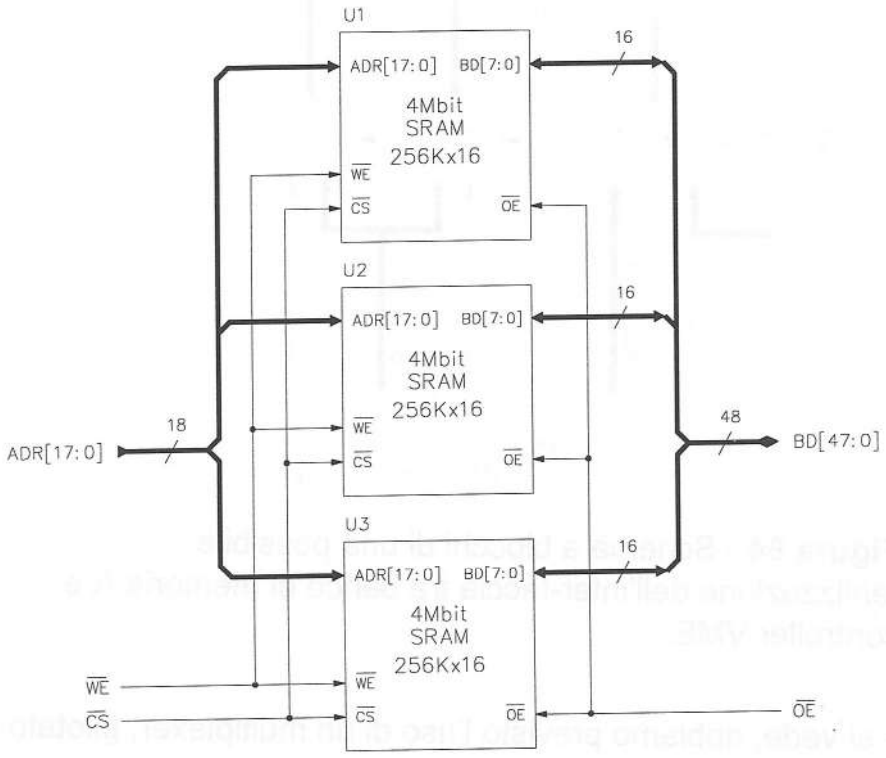


Figura 63 - Schema di una possibile realizzazione del blocco di memoria da 256Kx48, che sfrutta i nuovi chip RAM da 256Kx16 (4Mbit).

Sottosistema di memoria - Banco A

Tra i banchi di memoria, per così dire, di servizio, questo è il più complesso, in quanto, come abbiamo già visto, occorre prevedere la possibilità di interfacciamento diretto, oltre che con il DSP, anche con il bus VME, in modo da permettere la lettura e la scrittura di dati e coeffi-cienti da parte di un calcolatore esterno al sistema. Occorre quindi considerare un meccanismo che permetta di collegare la memoria, *alternativamente*, al sistema costituito da generatore di indiriz-zi e DSP, oppure al bus VME, ad esempio secondo lo schema di Fig. 64.

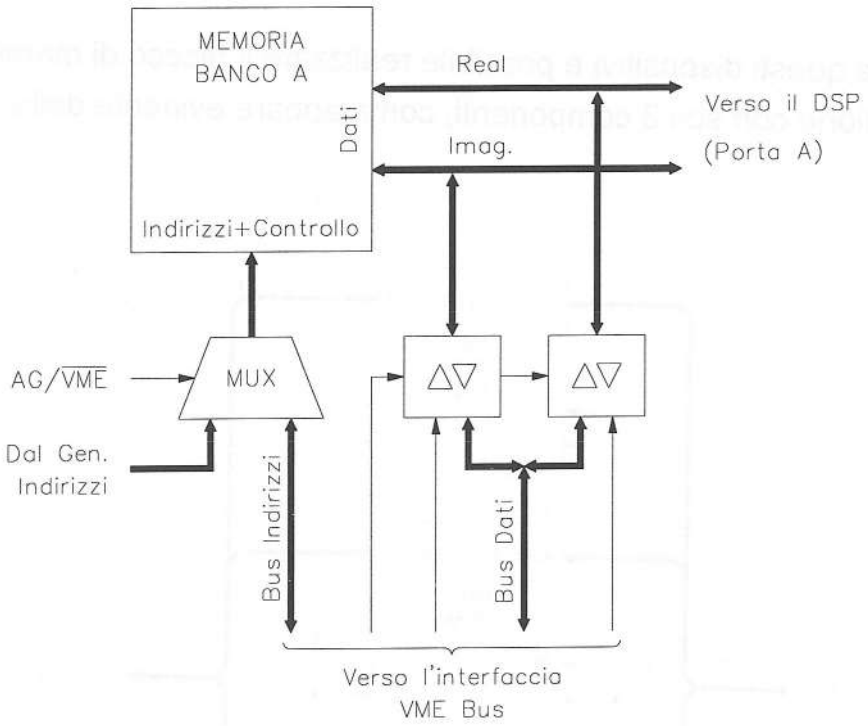


Figura 64 - Schema a blocchi di una possibile realizzazione dell'inter-faccia tra banco di memoria A e controller VME.

Come si vede, abbiamo previsto l'uso di un multiplexer, pilotato dal sistema di controllo (o, addirittura, dalla stessa interfaccia VME), sul bus indirizzi della memoria. Esso permette di inviare alla memoria gli indirizzi ed i segnali di controllo provenienti dal generatore di indirizzi, oppure quelli che si ottengono dall'interfaccia VME.
Per quanto riguarda il bus dati della memoria, esso si collega *sia* alla porta A del DSP, direttamente, come nei casi precedenti, *sia* ad un banco di *transceiver* che si interfacciano con il bus dati VME. Questo schema è stato preferito in quanto permette di ridurre al minimo i ritardi sul cammino "veloce" dei dati (cioè quello nei riguardi del DSP), anche se rende un poco macchinosa la gestione del trasferimento dati da e per il bus VME.
Infatti, qualora si desideri scrivere o leggere dati sulla memoria da parte del computer host, *occorre predisporre in lettura la porta A del DSP prima di abilitare i transceiver*, in modo da evitare conflitti sul bus dati. Questa operazione può essere fatta facendo eseguire al

DSP una qualsiasi istruzione che preveda l'uso della porta A come ingresso, e quindi richiede un certo lavoro di programmazione.
In Fig.65 presentiamo uno schema un poco più dettagliato, in cui sono riportati tutti i segnali utili. Rispetto ai sottosistemi di memoria che abbiamo già esaminato, questo si distingue soprattutto per la presenza del multiplexer, che gestisce la commutazione delle linee di indirizzamento e delle linee di controllo della memoria, cioè MEMW# e MEMOE#.
Per quanto riguarda il bus VME, l'idea è quella di sfruttare le capacità di trasferimento dati a 24bit, trattando separatamente la parte reale e quella immaginaria dei dati contenuti nella RAM.
L'unico inconveniente è rappresentato dal grande spazio di indirizzamento richiesto, pari a 1Mword da 24bit ciascuna (3Mbyte), ma ciò non dovrebbe costituire un problema eccessivo data l'enorme capacità di indirizzamento (fino a 32bit, cioè fino a 4GB) che il bus VME è in grado di sostenere.
L'interfaccia VME dovrà fornire anche i segnali che servono a gestire i due transceiver, oltre, ovviamente, agli indirizzi ed ai segnali MEMAWR# e MEMAEO#; quest'ultimo viene anche utilizzato per gestire la "direzione" dei transceiver, come si vede chiaramente nello schema.

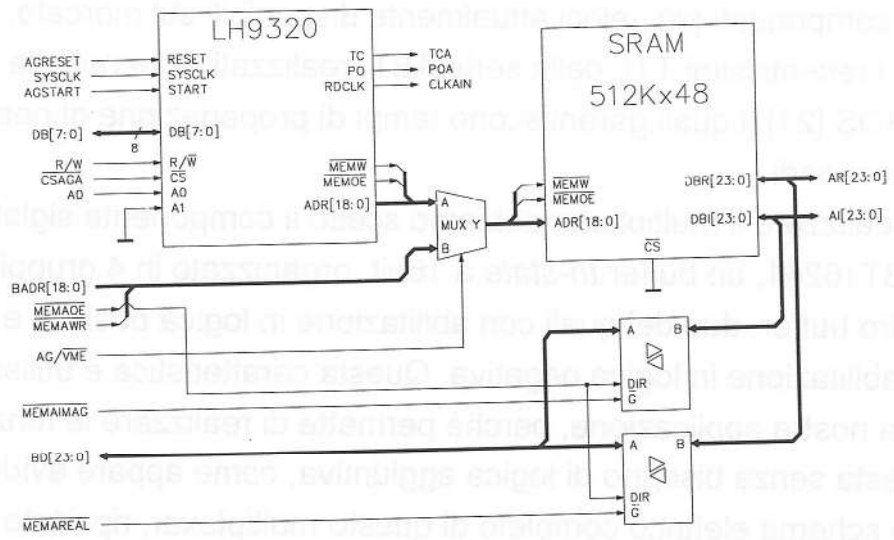


Figura 65 - Schema dettagliato della stessa realizzazione di Fig. 64.

Il banco di memoria da 512Kx48 è, ancora una volta, del tutto simile a quelli già visti e può, quindi, essere realizzato con lo schema di Fig. 60.
Il multiplexer, invece, è un componente piuttosto critico, in quanto rallenta il cammino degli indirizzi generati dall'LH9320. Per mantenere minimo il ritardo, si è quindi pensato di realizzare tale elemento mediante *due banchi di buffer a 3 stati*, con le uscite in parallelo e collegati in modo che l'abilitazione di un banco disabiliti automaticamente l'altro, così come appare nello schema a blocchi di Fig. 66.

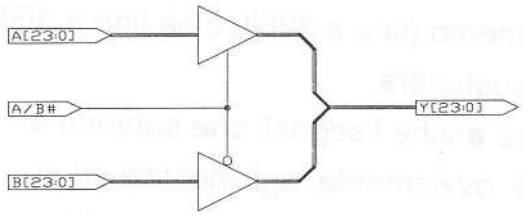


Figura 66 - Schema a blocchi del multiplexer utilizzato nel sottosistema di memoria di Fig. 66.

Tra i componenti più veloci attualmente disponibili sul mercato, vi sono i recentissimi TTL della serie ABT, realizzati in tecnologia BiCMOS [21], i quali garantiscono tempi di propagazione di pochi nanosecondi.
Per realizzare il multiplexer abbiamo scelto il componente siglato 74ABT16241, un buffer *tri-state* a 16bit, organizzato in 4 gruppi da quattro buffer, due dei quali con abilitazione in logica positiva e due con abilitazione in logica negativa. Questa caratteristica è utilissima per la nostra applicazione, perchè permette di realizzare la funzione richiesta senza bisogno di logica aggiuntiva, come appare evidente dallo schema elettrico completo di questo multiplexer, riportato in Fig. 67.

Si noti che, delle 24 linee che tale multiplexer mette a disposizione, solo 21 vengono utilizzate, per cui è consigliabile tenere a livello logico basso gli ingressi che non vengono sfruttati.
Per quanto riguarda, invece, il banco di transceiver sul bus dati, le richieste in termini di rapidità sono assai meno stringenti, in quanto, di solito, il sistema host non ha necessità di trasferire i dati alla massima velocità permessa dalla memoria, anche perchè il bus VME difficilmente potrebbe sostenerla.

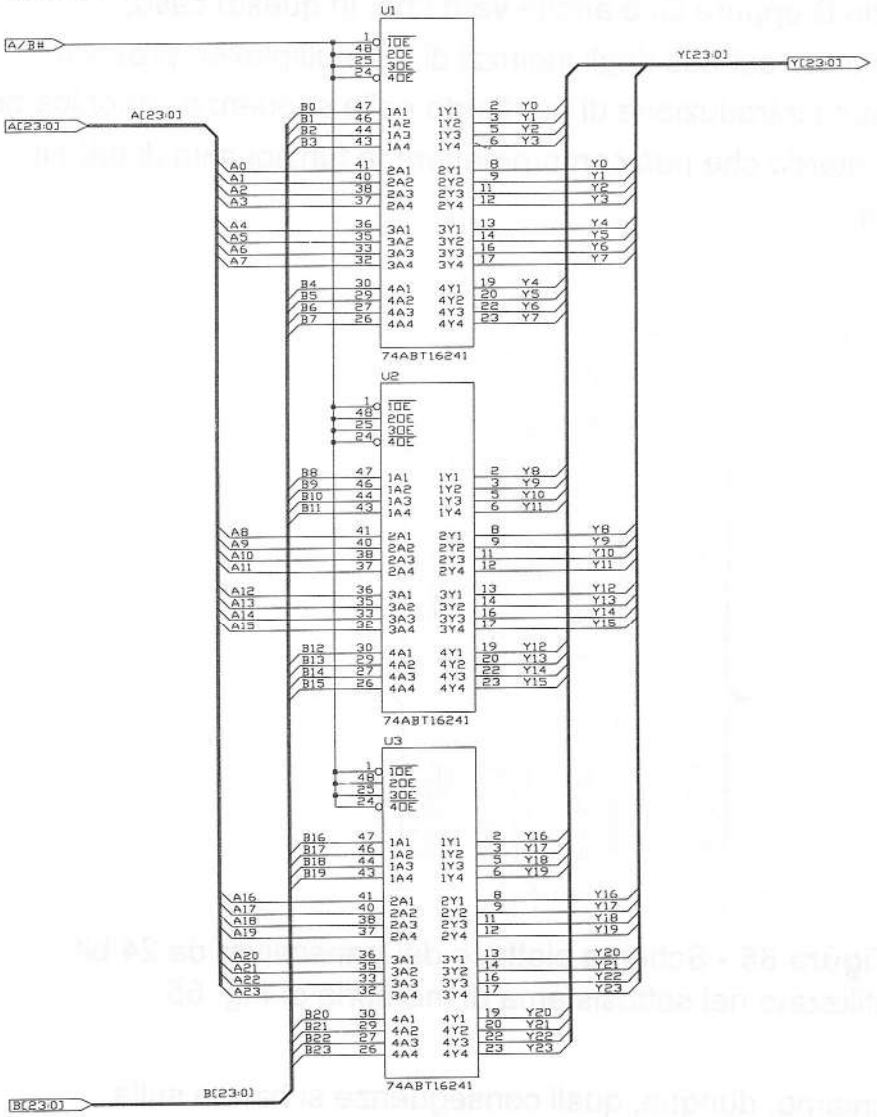


Figura 67 - Schema elettrico del multiplexer a 24 bit utilizzato nel sottosistema di memoria di Fig. 65.

Comunque sia, anche in questo caso abbiamo deciso di utilizzare componenti di tipo BiCMOS, realizzando il transceiver da 24 bit con i dispositivi 74ABT16245 e 74ABT245, che possono trattare, rispetti-vamente, 16 e 8 bit, così come è mostrato nello schema di Fig. 68.
A questo punto, però, s'impone una verifica: se è vero che il DSP LH9124 ed il generatore di indirizzi sono fatti l'uno per l'altro, per cui è garantita la funzionalità di circuiti come quelli relativi ai sistemi di memoria B oppure C, è anche vero che, in questo caso, l'inserimento sul bus degli indirizzi di un multiplexer, provoca senz'altro l'introduzione di un ritardo nella sequenza canonica degli eventi, ritardo che può compromettere la funzionalità di tutto il sistema.

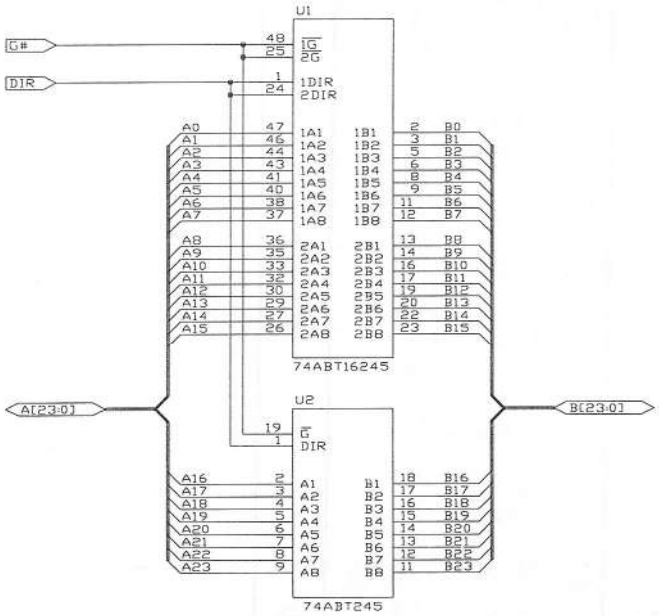


Figura 68 - Schema elettrico del transceiver da 24 bit utilizzato nel sottosistema di memoria di Fig. 65.

Verifichiamo, dunque, quali conseguenze si hanno sulla temporizza-zione a seguito dell'inserimento del multiplexer.
La Texas Instruments, per il dispositivo 74ABT16241, dichiara, tra gli altri, i seguenti tempi di propagazione:

Parametro	Ingresso	Uscita	MIN	TYP	MAX
t_{PLH}	A	Y	1	2.7	3.4
t_{PHL}	A	Y	1	2.7	3.9

Tabella- Principali tempi di propagazione dichiarati dal costruttore per il dispositivo 74ABT16241. Vcc=5V, Ta=25°C, tempi espressi in ns.

Visto che la presenza del multiplexer non fa altro che allungare il tempo di risposta della memoria, ciò che ci interessa maggiormente verificare è il rispetto dei tempi *minimi* di *setup* e di *hold* dei registri di ingresso relativi alla porta A del DSP, tenendo conto che il campionamento dei dati è gestito dal segnale RDCLK prodotto dal generatore di indirizzi, ed, in particolare, avviene in corrispondenza dei fronti di salita di quest'ultimo.
Per maggior chiarezza, in Fig. 69 abbiamo riportato la temporizzazione dei segnali che ci interessano, ricavata adattando al nostro caso i diagrammi presenti nei data sheet Sharp.

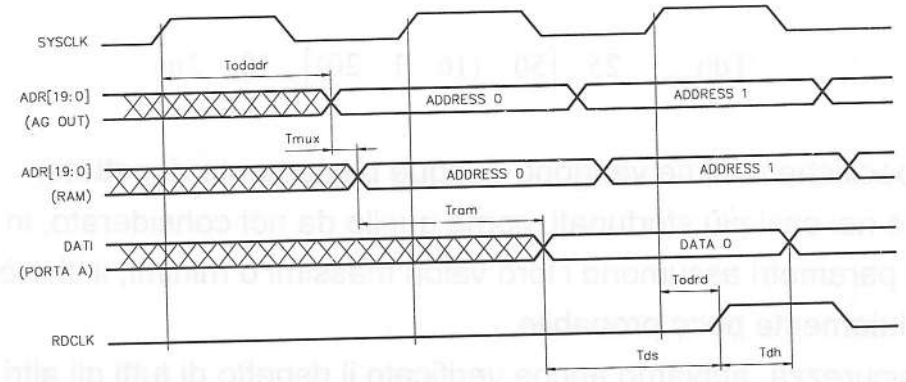


Figura 69 - Temporizzazione dei segnali critici nel circuito di Fig. 65.

In questa figura, possiamo distinguere, dall'alto in basso, il segnale di clock di sistema, gli indirizzi così come vengono generati dall'LH9320, gli indirizzi che invece arrivano alla RAM, ritardati dalla

presenza del multiplexer, i dati che escono dalla RAM e raggiungono la porta A del DSP ed, infine, il segnale RDCLK, che pilota l'ingresso CLKAIN del DSP e che serve a campionare i dati.
Ciò che dobbiamo verificare è che siano rispettati i tempi **Tds_{min}** e **Tdh_{min}**, che la Sharp dichiara essere pari a, rispettivamente, **7ns** e **-1.5ns**.
Tutti tempi indicati nel diagramma sono quelli riportati nei data sheet dei dispositivi LH9124 e LH9320, fatta eccezione per Tmux, che è il tempo di propagazione del multiplexer e che dovrà essere scelto tra quelli elencati nella tabella precedente, e Tram, che è il tempo di accesso della memoria e che assumeremo pari a 20 nanosecondi.
Ricordando che il tempo di ciclo è di 25 nanosecondi, possiamo calcolare Tds_{min} e Tdh_{min} come segue:

$$Tds_{min} \quad 50ns \quad (Todadr_{max} \quad Tmux_{max} \quad Tram_{max}) \quad Todrd_{min}$$

$$Tdh_{min} \quad 25ns \quad [50ns \quad (Todadr_{min} \quad Tmux_{min} \quad Tram_{min})] \quad Todrd_{max}$$

Sostituendo i valori dichiarati, otteniamo:

$$Tds_{min} \quad 50 \quad (20 \quad 3.9 \quad 20) \quad 2 \quad 8.1ns$$

$$Tdh_{min} \quad 25 \quad [50 \quad (16 \quad 1 \quad 20)] \quad 10 \quad 2ns$$

Le specifiche minime vengono dunque pienamente rispettate, anche nei casi più sfortunati, come quello da noi considerato, in cui tutti i parametri assumono i loro valori massimi o minimi, il che è assolutamente poco probabile.
Per sicurezza, abbiamo anche verificato il rispetto di tutti gli altri parametri secondari che, sia pur marginalmente, sono influenzati dal ritardo dovuto al multiplexer. Anche se tale verifica non viene qui riportata, per evitare di appesantire troppo la trattazione, tutti i valori risultano rientrare "abbondantemente" nelle specifiche.

Sottosistema di ingresso/uscita

Passiamo, finalmente, ad analizzare questa particolare sezione del sistema, che, grazie alla sua architettura, permette al DSP di lavorare al massimo delle sue possibilità.
Come abbiamo descritto nel capitolo precedente, ma giova ricordarlo qui, questo sottosistema deve permettere la scrittura di nuovi dati su uno dei banchi della sezione di ingresso mentre, contemporaneamente, il DSP legge i dati precedenti dall'altro banco. La stessa cosa deve essere possibile in uscita: devo poter leggere una serie di risultati da uno dei banchi mentre il DSP scrive nuovi risultati sull'altro.
Scrittura dei dati e lettura dei risultati devono essere asincroni rispetto al sistema di calcolo e devono poter avvenire ad una qualsiasi frequenza di clock minore od uguale a quella di sistema, che, come sappiamo, è pari a 40MHz.
Infine, la sezione di ingresso deve poter trattare sia dati immaginari che dati reali, eseguendo, in quest'ultimo caso, l'operazione di interleaving dei campioni, più volte ricordata nei capitoli precedenti. Dato che quest'ultima operazione non è richiesta alla sezione di uscita, che quindi è più semplice, concentreremo la nostra attenzione sulla sola sezione di ingresso.
Consideriamo, per ora, *il solo bus dati* dei due banchi di memoria, che supporremo avere, per maggior generalità, una capacità di 512Kx48, anche se, come nel nostro caso, si può utilizzare un parallelismo minore quando i dati ingresso sono costituiti da parole più corte. Chiamiamo IN_A e IN_B queste due memorie.
Affinchè sia possibile scrivere su uno dei banchi mentre si legge dall'altro, occorre, perlomeno, prevedere un sistema di buffer del tipo di quello rappresentato in Fig. 70.

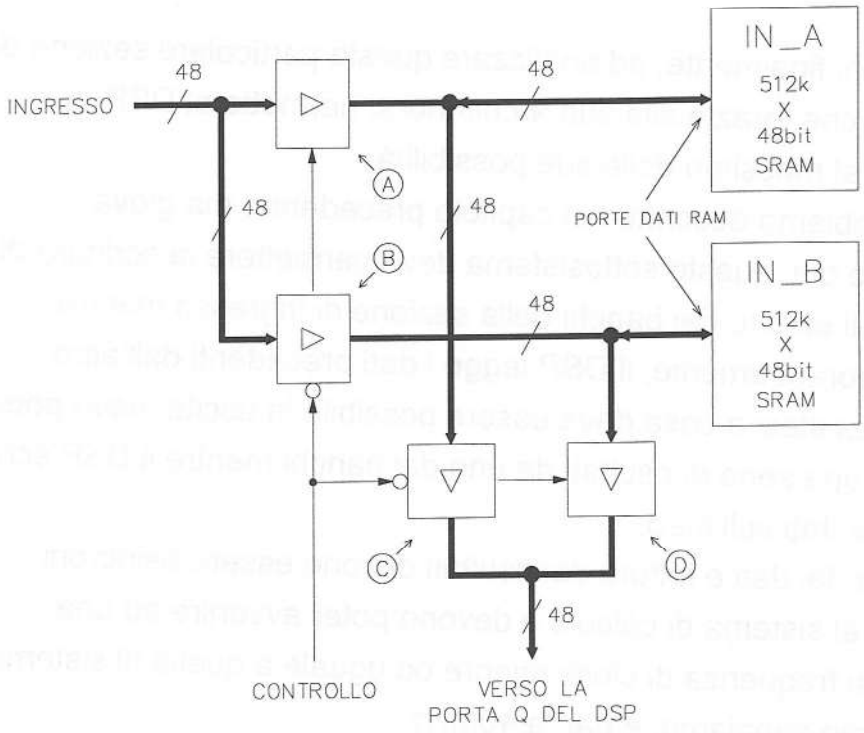


Figura 70 - Schema di principio della sezione di ingresso del sottosistema di I/O, dal lato dati.

Abbiamo, dunque, quattro buffer da 48 bit ciascuno, di cui due (A e B) servono a smistare i dati in arrivo dall'esterno, mentre gli altri due (C e D) trattano i dati che sono diretti al DSP.
Come si vede nello schema, possiamo pensare, in linea di principio, di gestire tutti e quattro i buffer con una sola linea di controllo, se supponiamo che le uscite dei dispositivi B e C si attivino quando tale linea è a livello logico basso e che il contrario valga per i buffer A e D.
Dunque, quando la linea di controllo si trova a livello logico basso, i dati in ingresso possono attraversare il solo buffer B, raggiungendo, quindi, il banco IN_B, mentre il DSP può leggere dal banco IN_A attraverso il buffer C, senza alcun pericolo di conflitti.
Quando, invece, la linea di controllo si porta a livello alto, si ottiene la situazione opposta, in cui posso scrivere sul banco IN_A e leggere dal banco IN_B.
Notiamo subito, comunque, che a questo schema di principio occorre fare delle modifiche. Anzitutto, è necessario poter

disabilitare *entrambi* i buffer C e D, in modo da evitare conflitti quando la porta Q del DSP funziona da uscita, durante il trasferimento dei risultati all'altra sezione del sottosistema di I/O, per cui serve almeno un'altra linea di controllo.
Inoltre, occorre prevedere un meccanismo per effettuare l'inter-leaving dei dati quando questi sono di tipo reale.
Dallo schema di principio di Fig. 70 passiamo, allora, allo schema di Fig. 71, che rappresenta la struttura definitiva del "lato dati" della sezione di ingresso, anche se si tratta sempre di uno schema a blocchi.
Anche se l'architettura rimane la stessa della figura precedente, appare ora evidente la distinzione tra parte reale e parte immaginaria, sia per quanto riguarda la memoria, sia per quanto riguarda i relativi bus dati.
Ogni banco di memoria è stato quindi rappresentato come formato da due blocchi distinti da 512Kx24: uno destinato ad accogliere la parte reale dei dati, oppure i soli campioni di ordine pari, nel caso di dati reali, e l'altro per i dati immaginari oppure per i campioni di ordine dispari.
Indirizzando le memorie IN_A e IN_B come se fossero dei blocchi unici da 512Kx48 e mantenendo disabilitato il buffer A, si possono trattare dati complessi a 24+24 bit, secondo le modalità che abbiamo già illustrato.
Qualora si desideri, invece, trattare dati di tipo reale, con precisioni fino a 24 bit, occorerà abilitare il buffer A ed indirizzare, alternativamente, i blocchi REAL/EVEN e IMAG/ODD di ogni banco, in modo da realiz-zare l'interleaving dei campioni.

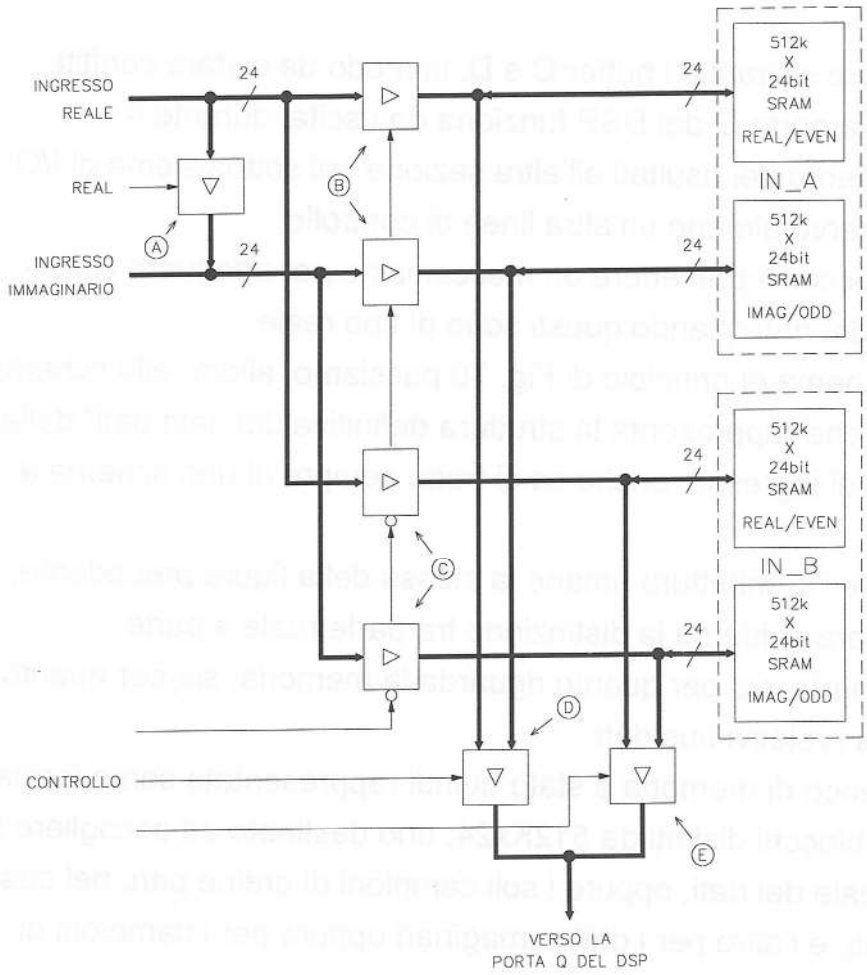


Figura 71 - Schema a blocchi della sezione di ingresso del sottosistema di I/O, dal lato dati.

Facciamo un esempio, per spiegarci meglio, supponendo di voler scrivere dati *reali* sul banco IN_A (e leggere, di conseguenza, dal banco IN_B). Per fare questo, dovremo anzitutto, tramite i segnali di controllo, abilitare i buffer B, E ed, in particolare, anche quello indicato con A.
L'abilitazione del buffer A mette in comune i due bus di ingresso, in modo che i dati presenti all'ingresso reale vengono trasferiti contemporaneamente ad entrambi i blocchi.
L'interleaving si ottiene *durante la scrittura dei dati*, operando dal lato indirizzi, come vedremo tra poco, abilitando il solo blocco REAL/EVEN quando l'indirizzo generato è pari, oppure abilitando il blocco IMAG/ODD quando l'indirizzo è dispari. *La successiva*

lettura, invece, deve essere fatta indirizzando insieme entrambi i blocchi, in modo da leggere parole complesse da 24+24 bit, in cui la parte reale è formata dagli elementi pari della sequenza di ingresso, e la parte immaginaria da quelli dispari.
A seconda dello schema di indirizzamento della memoria e di abilitazione dei buffer, possiamo dunque ottenere due modi diversi di funzionamento, che chiameremo "modo complesso" e "modo reale".
Vediamo quindi come può essere organizzato il sistema di indirizzamento della memoria, affinchè si possa operare in entrambi i modi complesso e reale.
Come mostrato in Fig. 72, servono, anzitutto, due generatori di indirizzi indipendenti, in modo che si possa indirizzare uno dei due banchi di memoria per la scrittura, mentre si legge dall'altro secondo una diversa sequenza ed, eventualmente, ad una diversa velocità.

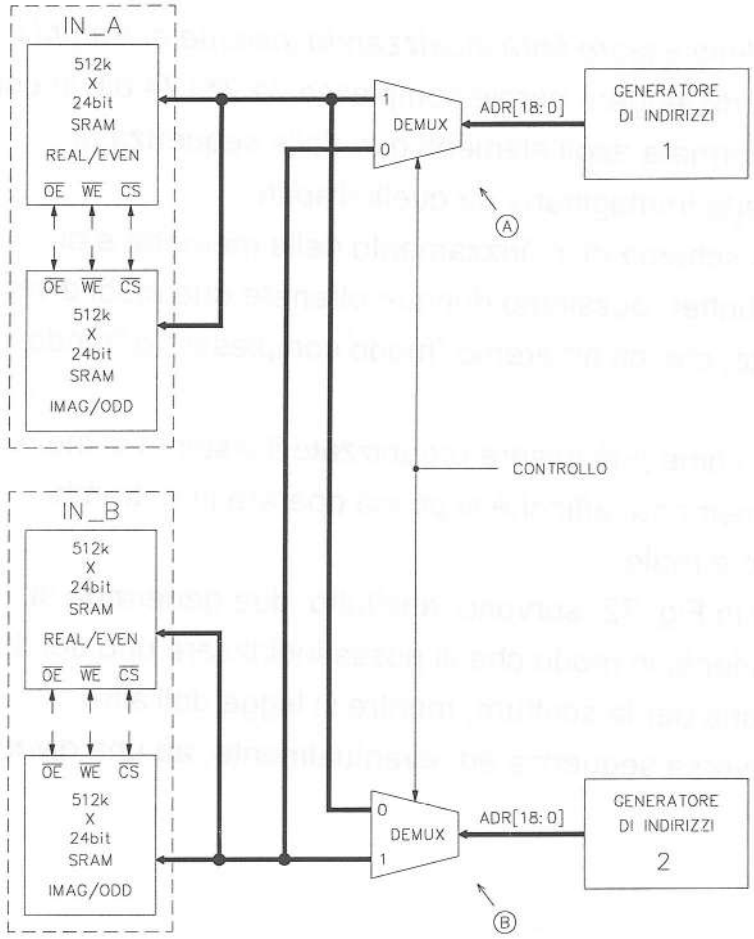


Figura 72 - Schema di principio del circuito di indirizzamento della sezione di ingresso, nel sottosistema di I/O.

Il generatore di indirizzi 1, ad esempio, può generare gli indirizzi utili per la lettura, mentre il 2 quelli per la scrittura.

I due demultiplexer provvedono ad inviare gli indirizzi prodotti da ciascun generatore alternativamente ai due banchi: quando la linea di controllo è a livello logico basso, ad esempio, il generatore 1 può indirizzare il banco IN_B, mentre il generatore di indirizzi 2 pilota il banco IN_A. Quando il segnale di controllo è, invece, a livello logico alto i due generatori si scambiano i compiti.

A questo punto, abbiamo il problema di gestire questo e tutti gli altri segnali di controllo, in modo da poter implementare i due modi di funzionamento visti prima.

Il generatore di indirizzi 1, a cui abbiamo assegnato il compito di indirizzare la memoria durante la lettura dei dati da parte del DSP,

può tranquillamente essere uno Sharp LH9320, in quanto ad esso si chiede di generare, alla massima velocità possibile, cioè con un clock di 40MHz, le sequenze relative all'indirizzamento del primo stadio della FFT o, quelle utili per il windowing dei dati.
Il generatore di indirizzi 2, invece, che ha dunque il compito di indirizzare la memoria durante la scrittura dall'esterno, deve essere un componente diverso. Vediamo perchè.
Anzitutto, esso deve essere sincronizzato con il sistema a monte, che fornisce i dati da elaborare e, quindi, anche il segnale di clock per questo generatore. Inoltre, deve poter generare esclusivamente sequenze di indirizzamento lineari, visto che, per la lettura dei dati, abbiamo a disposizione lo Sharp LH9320, che è in grado di indirizzare i dati secondo una sequenza qualsiasi.
Ma, soprattutto, e questo è il motivo principale per cui non si può utilizzare un altro LH9320 per indirizzare la memoria in scrittura, questo generatore di indirizzi deve poter essere controllato dall'esterno, per via hardware, in modo da poter interrompere la sequenza corrente in qualsiasi momento, per poi riprenderla successivamente esattamente da dove era stata interrotta.
Questa possibiltà, che, come abbiamo visto, è preclusa al dispositivo Sharp, è indispensabile per implementare un meccanismo, anche minimo, di *handshaking* nei confronti del sistema che fornisce i dati, i quali possono non essere disponibili ad un ritmo regolare nel tempo.
Un esempio tipico può essere quello in cui il sistema a monte è dotato di una memoria FIFO in uscita, proprio come il convertitore A/D attualmente in uso presso il Radiotelescopio di Medicina. In questo caso, infatti, per evitare underflow nella FIFO, occorre gestire i segnali di stato che la FIFO stessa mette a disposizione, bloccando, se necessario, la lettura dei dati.
Il componente ideale è dunque rappresentato da un semplice *contatore programmabile*, ad almeno 20 bit e gestibile per mezzo dei vari segnali di start ed enable tipici di questi componenti.

Si potrebbe pensare, poi, ed è proprio ciò che è stato fatto, di utilizzare proprio questo contatore per gestire i meccanismi di swap tra i due banchi di memoria e di interleaving dei dati.
Lo swap tra i due banchi, infatti, può essere pilotato da uno dei bit più significativi di questo generatore di indirizzi: una volta raggiunto un certo conteggio, ovvero una volta che sia stato riempito il banco di memoria su cui si sta scrivendo, l'attivazione del bit più significativo, che non viene usato per indirizzare la memoria, può essere utilizzata per comandare il passaggio al banco successivo.
La Fig.73 mostra una sequenza di indirizzamento lineare da 20 bit. Si possono utilizzare i 19 bit meno significativi per indirizzare la memoria (fino a 512K), sfruttando, invece, quello più significativo per passare alla scrittura sul banco successivo, una volta che quella sul primo è stata completata.
Analogamente, si può sfruttare il bit meno significativo (che, lo ricordiamo, è 0 negli indirizzi pari e 1 in quelli dispari) per passare da un blocco all'altro, all'interno di uno stesso banco di memoria, e realizzare così la desiderata formattazione dei dati, qualora questi siano di tipo reale.
In questo caso occorrerà riservare il bit meno significativo degli indirizzi alla gestione del passaggio tra un blocco e l'altro, ed indirizzando entrambi i blocchi con la altre linee. Possiamo osservare, a questo proposito, ancora la Fig.73.
Quando i dati sono di tipo *reale*, quindi, occorre indirizzare i due blocchi all'interno di ogni banco con i bit 1..18, abilitando il blocco pari oppure quello dispari a seconda dello stato del bit 0 (LSB), e riservando il bit 19 (MSB) alla gestione dello swap tra il banco IN_A ed il banco IN_B.
Con i dati in ingresso di tipo complesso, invece, bisogna indirizzare entrambi i blocchi (che saranno sempre abilitati insieme) di ciascun banco con i bit 0..18, continuando ad effettuare lo scambio tra i banchi IN_A e IN_B con la linea corrispondente al bit 19 (MSB).

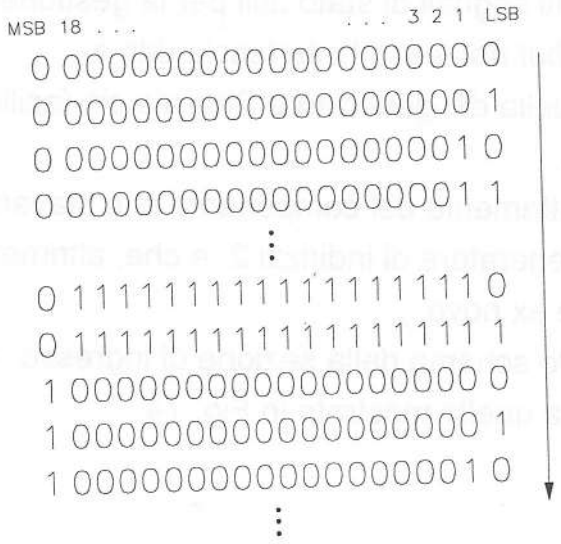


Figura 73 - Esempio di sequenza di indirizzamento lineare a 20 bit, per dimostrare come il bit più significativo e quello meno significativo possono essere sfruttati, rispettivamente, per per passare da un banco all'altro e da un blocco all'altro.

Se si desidera far convivere i due modi di funzionamento, reale e complesso, con la possibilità di passare da uno all'altro a seconda dei casi, la gestione degli indirizzi diviene, dunque, piuttosto complicata.
Fortunatamente, la Harris produce un generatore di indirizzi, dalla sigla HSP45240, che risolve, molto semplicemente, tutti questi problemi.
Questo dispositivo viene presentato brevemente in appendice. Per ora, ci basti sapere che si tratta di un generatore di indirizzi molto più semplice di quello Sharp, che può produrre indirizzi fino a 24 bit e che le uscite del generatore interno (che produce solo indirizzamenti lineari) possono essere portate ai terminali esterni in un ordine qualsiasi, grazie ad un "commu-tatore programmabile". Questa possibilità è utilissima proprio per poter passare dal modo reale al modo complesso, in quanto permette di indirizzare la memoria, di volta in volta, con le linee più adatte.
Inoltre, questo generatore di indirizzi dispone di un ingresso di controllo attraverso il quale si può interrompere il processo di

generazione, e produce molti segnali di stato utili per la gestione del trasferimento dei dati. *Last but not least*, l'interfaccia dati è assolutamente identica a quella del generatore Sharp e ciò facilita enormemente il suo utilizzo.
Si tratta, in altre parole, esattamente del componente che stavamo cercando per realizzare il generatore di indirizzi 2, e che, altrimenti, avremmo dovuto progettare ex novo.
Utilizzando tale dispositivo lo schema della sezione di ingresso, dal lato indirizzi, diventa simile a quello mostrato in Fig. 74.

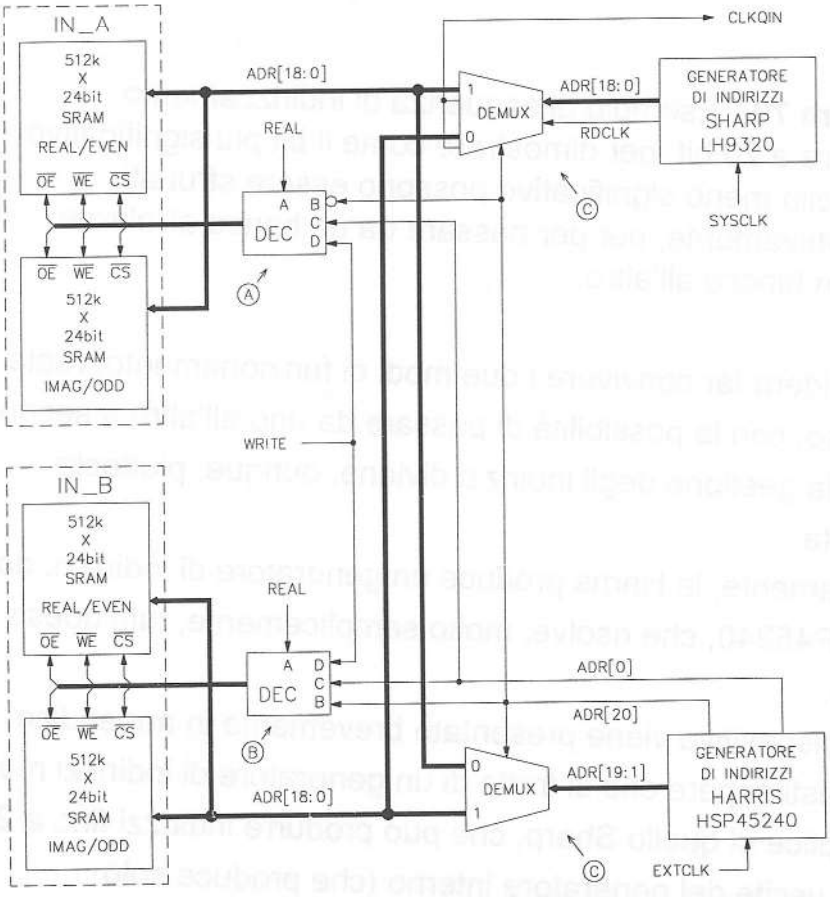


Figura 74 - Schema a blocchi della sezione di ingresso del sottosistema di I/O, dal lato indirizzi.

Notiamo, anzitutto, i due decoder A e B, che provvedono a generare i segnali di controllo della memoria, in base allo stato del flag REAL, che indica, quando è attivo, che si desidera leggere dati di tipo reale, del segnale WRITE, fornito dal sistema a monte, e a

seconda dello stato delle linee ADR[0] (bit meno significativo degli indirizzi) e ADR[20] (bit più significativo) in uscita dal generatore Harris.
Il generatore Sharp viene utilizzato per generare gli indirizzi in lettura (da parte del DSP), e di esso si utilizzano solo le linee di indirizzamento 0..18 ed il segnale RDCLK, che serve per comandare il campionamento dei dati letti. A questo segnale viene fatto attraversare il demultiplexer per questioni di temporizzazione, che vedremo tra poco.
Nel modo di funzionamento complesso, il generatore Harris deve essere programmato in modo che le linee meno significative del generatore interno raggiungano, ordinatamente, le uscite ADR[19:1] che pilotano il demultiplexer (senza utilizzare, dunque, l'uscita ADR[0]). Essendo, in questo caso, il flag REAL disattivato, i due decoder faranno in modo che entrambi i blocchi di ciascun banco risultino abilitati (attivando i relativi CS *contemporaneamente*), e provvederanno a generare i segnali WE e OE, in base allo stato della linea ADR[20] (sulla quale dovrà essere dirottato il bit più significativo della sequenza corrente), in modo da permettere letture e scritture alternate sui due banchi.
In questo modo di funzionamento, gli indirizzi generati in lettura e scrittura avranno l'ordine mostrato in Fig. 75.

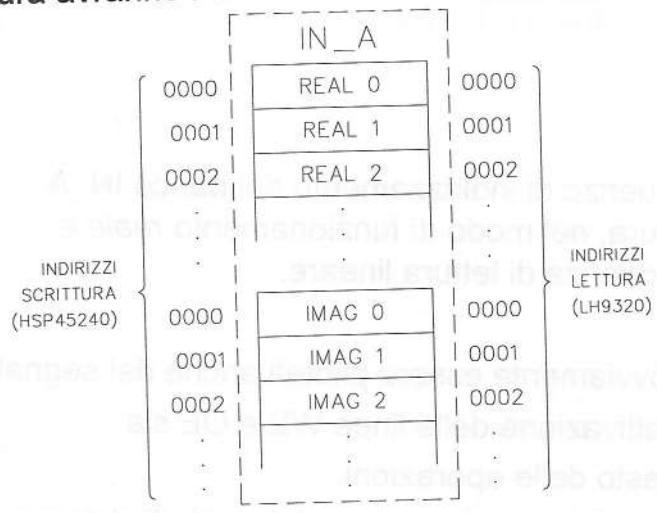


Figura 75 - Sequenze di indirizzamento del banco IN_A in scrittura e lettura, nel modo di funzionamento *complesso* e nell'ipotesi di sequenza di lettura lineare.

Nel modo di funzionamento reale, invece, il generatore Harris deve essere programmato in modo che le uscite ADR[0] ADR[19:1] e ADR[20] siano effettivamente collegate alle corrispondenti linee del generatore interno. In questo modo, i decoder, informati dell'intenzione di trattare dati reali grazie all'attivazione del flag REAL, possono abilitare a turno i due blocchi di ogni banco (attivando i relativi CS *alternativamente*) in base allo stato della linea ADR[0], realizzando così l'interleaving dei dati.
Il resto delle operazioni procederà, invece, come nel caso precedente, ottenendo così delle sequenze di indirizzamento simili a quelli mostrate in Fig. 76.

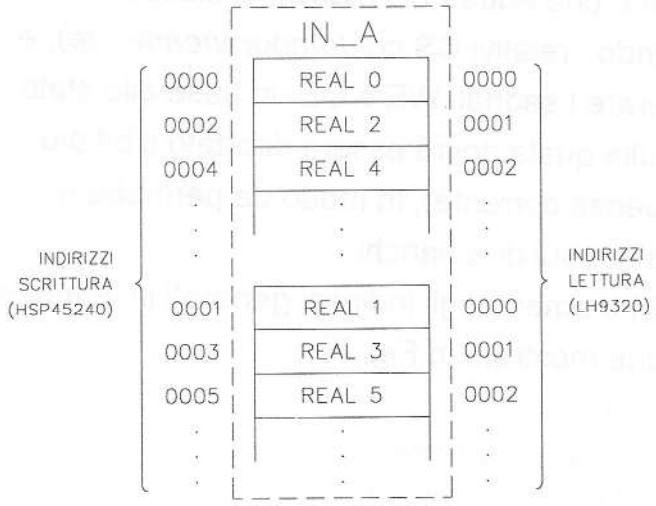


Figura 76 - Sequenze di indirizzamento del banco IN_A in scrittura e lettura, nel modo di funzionamento *reale* e nell'ipotesi di sequenza di lettura lineare.

I decoder dovranno ovviamente essere pilotati anche dai segnali di clock, in modo che l'attivazione delle linee WE e OE sia sincronizzata con il resto delle operazioni.
Il generatore di indirizzi Sharp, comandato dal clock di sistema SYSCLK, può ricevere il segnale di start dal generatore Harris, in

modo che possa iniziare le operazioni di lettura dei dati da uno dei banchi non appena sono terminate le operazioni di scrittura su quel banco. Questo stesso segnale dovrà essere usato per far partire tutto il resto del sistema, passando attraverso il sottosistema di timing.
Il generatore di indirizzi Harris, invece, può ricevere il segnale di start sia dall'esterno, via hardware, sia via software. Inoltre, è comandato sull'ingresso di clock dal sistema che fornisce i dati e risulta, quindi, sincronizzato con esso.
Per quanto riguarda la scelta dei componenti, valgono le stesse considerazioni fatte durante lo studio del sottosistema di memoria A.
Ogni blocco di memoria da 512Kx24 può essere realizzato con soli 3 chip da 4Mbit, così come mostrato in Fig. 77. Anche in questo caso, possiamo trovare il dispositivo che ci serve nel catalogo Samsung, la quale ha annunciato anche RAM da 512Kx8, siglate KM684002.

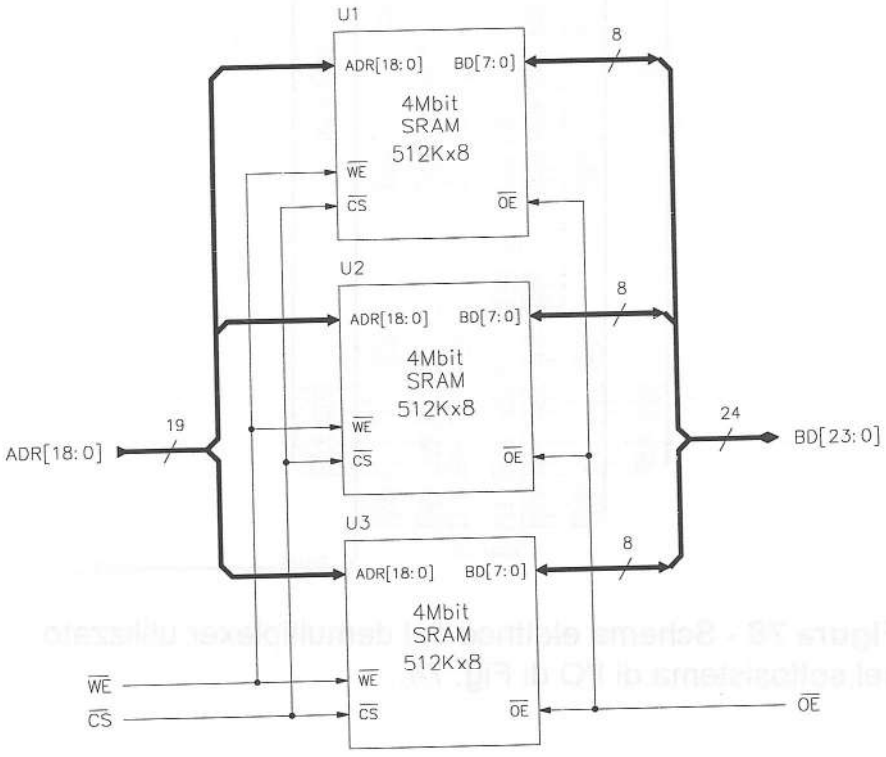


Figura 77 - Schema elettrico di una possibile realizzazione del blocco di memoria da 512Kx24, che sfrutta i nuovi chip RAM da 512Kx8 (4Mbit).

Infine, possiamo realizzare il demultiplexer con gli stessi componenti che abbiamo impiegato nel multiplexer di Fig.64, cioè i BiCMOS di tipo 74ABT16241. Anche il suo schema, visibile in Fig.78, è simile a quello del multiplexer, tranne per il fatto che, in questo caso, sono gli ingressi ad essere posti in comune.

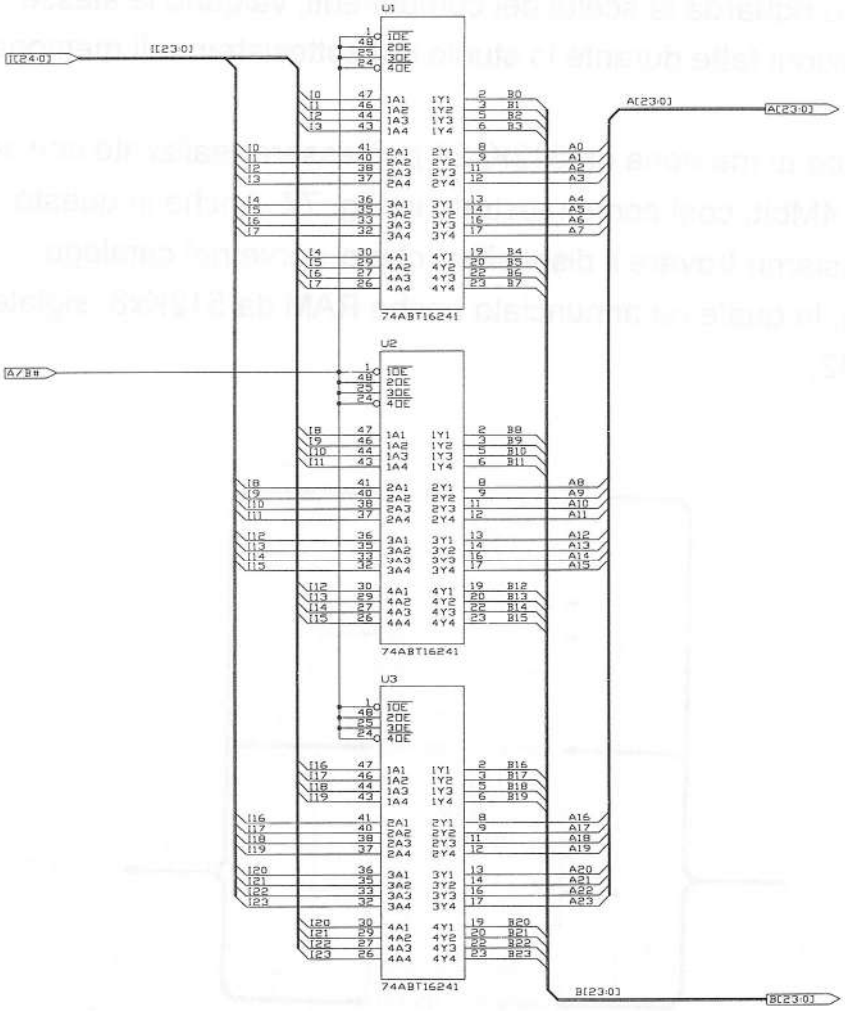


Figura 78 - Schema elettrico del demultiplexer utilizzato nel sottosistema di I/O di Fig. 74.

Per completare l'analisi di questo sottosistema è necessario verificare le temporizzazioni dei segnali più critici, così come abbiamo già fatto nel paragrafo precedente.
Come nel sistema di memoria relativo al banco A, anche in questo caso abbiamo inserito un "componente estraneo" sul bus indirizzi della memoria. In più, questa volta è indispensabile "interrompere" anche il bus dati, della stessa memoria, con un banco di buffer, il quale introduce un ulteriore ritardo.
Le osservazioni che stiamo per fare valgono, ovviamente, sia per la lettura che per la scrittura sulla RAM, ma, siccome la velocità limite è quella raggiunta dal generatore di indirizzi Sharp, che serve per la lettura, faremo riferimento solo a questa particolare parte del circuito.
Come abbiamo gia' visto, il problema sarà ancora quello di rientrare entro le specifiche per quanto riguarda i tempi di setup e hold relativi al registro di ingresso della porta Q.
Siccome, in questo caso, abbiamo un ritardo aggiuntivo dovuto al buffer sulle linee dati, abbiamo pensato di ritardare il segnale RDCLK, facendolo passare attraverso il demultiplexer assieme alle linee di indirizzamento. Otteniamo, quindi, le temporizzazioni rappresentate nel seguente diagramma.

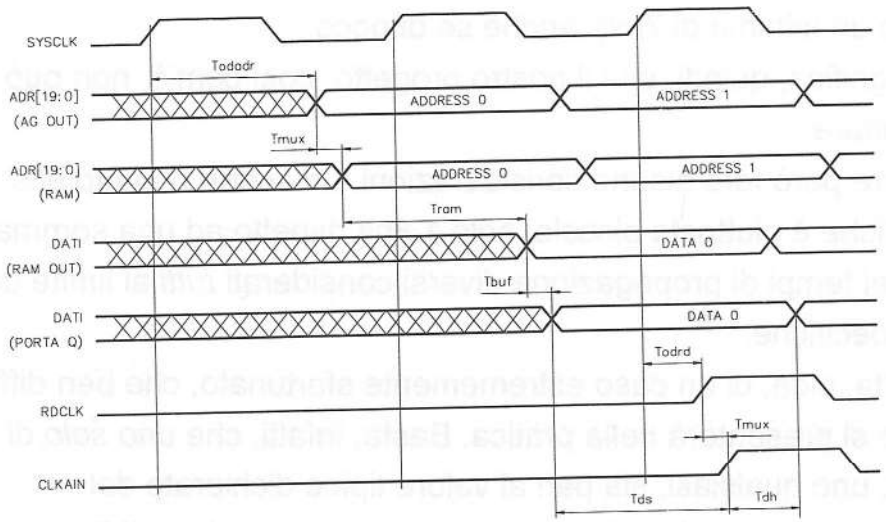


Figura 79 - Temporizzazione dei segnali critici nella sezione di ingresso del sottosistema di ingresso/uscita.

Da questo diagramma possiamo facilmente ricavare le espressioni dei parametri chi ci interessano, cioè:

$$Tds_{min} \quad 50ns \quad (Todadr_{max} \quad Tmux_{max} \quad Tram_{max} \quad Tbuf_{max}) \quad Todrd_{min} \quad Tmux_{min}$$

$$Tdh_{min} \quad 25ns \quad [50ns \quad (Todadr_{min} \quad Tmux_{min} \quad Tram_{min} \quad Tbuf_{min})] \quad (Todrd_{max} \quad Tmux_{max})$$

Poichè i componenti utilizzati per realizzare sia il demultiplexer sia i buffer sono sempre i 74ABT16241, possiamo ancora utilizzare i tempi limite elencati nella Tab. 6.3. Sostituendo, otteniamo allora:

$$Tds_{min} \quad 50 \quad (20 \quad 3.9 \quad 20 \quad 3.9) \quad 2 \quad 1 \quad 5.2ns$$

$$Tdh_{min} \quad 25 \quad [50 \quad (16 \quad 1 \quad 20 \quad 1)] \quad (10 \quad 3.9) \quad 0.9ns$$

Mentre il tempo di hold rientra nelle specifiche (circa 0.9ns contro un minimo di circa 1.5ns), *il tempo di setup è troppo breve* (5.2ns contro un minimo di 7ns), anche se di poco.
Ciò significa, quindi, che il nostro progetto, così com'è, non può funzionare.
Occorre però fare alcune considerazioni. Lo scarto rispetto alle specifiche è piuttosto piccolo, solo 1.8ns rispetto ad una somma di ben sei tempi di propagazione diversi considerati *tutti* al limite delle loro specifiche.
Si tratta, cioè, di un caso estrememente sfortunato, che ben difficilmente si presenterà nella pratica. Basta, infatti, che *uno solo* di quei tempi, uno qualsiasi, sia pari al valore tipico dichiarato dal costruttore, che subito il tempo Tds diventa maggiore di 7ns, rientrando così nelle specifiche.

Il progetto presentato, dunque, è perfettamente realizzabile. Per garantire il perfetto funzionamento, sarà sufficiente eseguire, in fase di costruzione, una minima selezione dei componenti, oppure, scelta che eliminerebbe qualsiasi problema, utilizzare memorie con tempo di accesso pari a 15ns, che, però, non saranno disponibili prima della fine del 1995.
Per quanto riguarda, infine, la sezione di uscita di questo sottosistema, abbiamo già detto che risulta del tutto simile a quella di ingresso appena esaminata, per cui, avendo la sola accortezza di scambiare tra loro i termini "scrittura" e "lettura", valgono le stesse considerazioni già fatte. Anzi, non dovendo distinguere tra dati complessi e dati reali (saranno sempre di tipo complesso) saranno molto semplificati sia i decoder, sia la programmazione del generatore di indirizzi Harris.

Sottosistema di controllo

A questa parte del sistema spetta il compito di gestire il funzionamento di tutti gli altri dispositivi presenti.
Oltre a generare, ad esempio, il segnale di start "ad alto livello" a seguito di un comando software ed a controllare il funzionamento dell'aritmetica in Block Floating-Point, la sua funzione principale è quella di fornire i codici operativi per le istruzioni che deve eseguire il DSP.
Grazie al particolare modo di operare dei componenti Sharp, a questo "controllore" non è richiesto di operare alla stessa, elevata, velocità del resto del sistema. Infatti, sia la formazione del codice delle singole istruzioni sia le altre operazioni, sono uniche per ogni passo, per cui, mentre ogni singola operazione del DSP all'interno di ogni passo viene eseguita in ragione di una ogni 25 nanosecondi, le operazioni, per così dire, esterne all'elaborazione vera e propria hanno una cadenza N volte più lenta, dove N è il numero di punti che viene elaborato ad ogni passo.

Ad esempio, nel caso di una FFT su 16K punti, occorre predisporre una nuova istruzione ogni 16K x 25ns = 409 s circa, che è una cadenza che qualsiasi sistema a microprocessore è in grado di seguire.
Per realizzare il cuore di questo sottosistema la scelta è caduta, appunto, su di un semplice microcontrollore a 8 bit, il tipo 68HC705C8 prodotto dalla Motorola.
Senza addentrarci troppo in una descrizione particolareggiata di questo componente, per la quale si rimanda alla documentazione citata, possiamo ricordare che in questo dispositivo, che si presenta in un contenitore da soli 40 pin, sono integrati:

una CPU completa, della famiglia 6805 Motorola
7744 byte di memoria EPROM (o OTPROM)
304 byte di memoria RAM
un sistema di I/O con ben 24 linee bidirezionali, oltre a 7 ingressi
un timer programmabile
un timer di "watchdog"

oltre ad una completa interfaccia seriale e ad altri componenti minori, non interessanti ai fini di questo progetto. La sua massima frequenza di clock è pari a 2MHz, cui corrisponde un tempo di ciclo di 500ns. L'idea è quella di utilizzare questo componente in uno schema del tipo di quello proposto in Fig. 80.

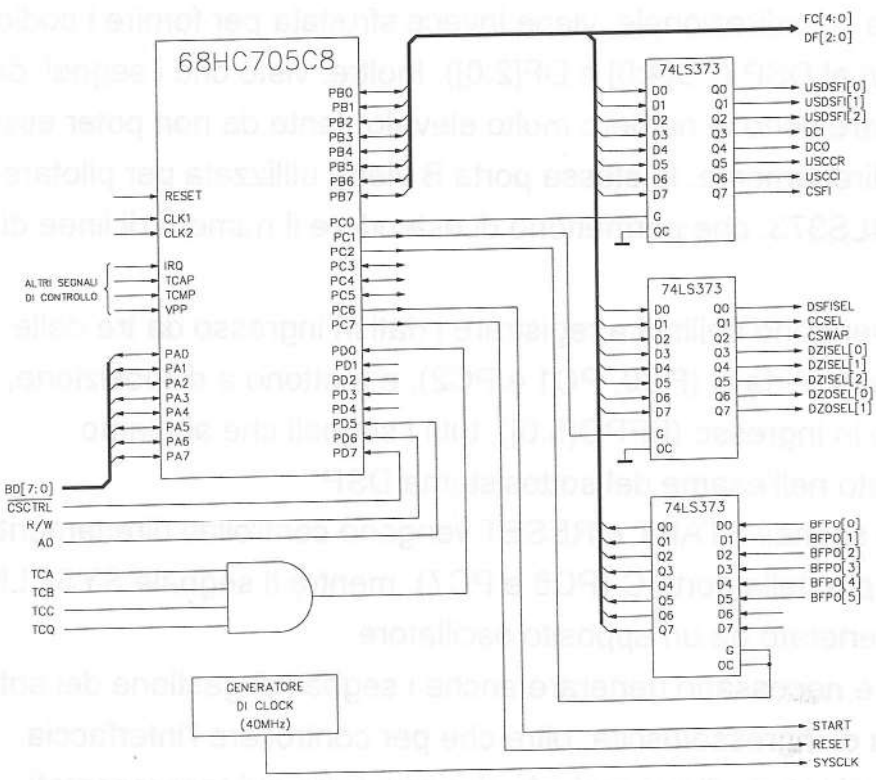


Figura 80 - Schema elettrico di una delle possibili realizzazioni del sottositema di controllo.

Come si vede, l'utilizzazione di questo dispositivo riduce al minimo l'hardware necessario, e le eventuali difficoltà saranno tutte legate alla programmazione del firmware di gestione, che risiederà sulla EPROM interna.

Analizzando la Fig. 80, possiamo osservare come l'intera porta A (bidirezionale) e tre dei sette ingressi siano stati utilizzati per implementare una interfaccia analoga a quella degli altri dispositivi utilizzati nel resto del sistema.

Attraverso questa interfaccia potrà essere programmata, ad esempio, la sequenza delle istruzioni che deve eseguire il DSP, il fattore di scala per i dati di ingresso oppure si potrà inviare il comando di avvio o di arresto delle operazioni, come vedremo tra poco.

Un altro ingresso (PD0) viene sfruttato per rilevare la fine di ciascun passo di elaborazione, che corrisponde al termine di tutte le sequenze di indirizzamento (tutti i segnali TC a livello alto).

La porta B, bidirezionale, viene invece sfruttata per fornire i codici operativi al DSP (FC[4:0] e DF[2:0]). Inoltre, visto che i segnali da controllare sono in numero molto elevato, tanto da non poter essere gestiti direttamente, la stessa porta B viene utilizzata per pilotare i *latch* 74LS373, che permettono di estendere il numero di linee di I/O.
I latch vengono abilitati a registrare i dati in ingresso da tre delle linee della porta C (PC0, PC1 e PC2), e mettono a disposizione, in uscita o in ingresso (BFPO[5:0]), tutti i segnali che abbiamo incontrato nell'esame del sottosistema DSP.
Infine, i segnali START e RESET vengono controllati direttamente da due pin della porta C (PC6 e PC7), mentre il segnale SYSCLK viene generato da un apposito oscillatore.
Poichè è necessario generare anche i segnali di gestione del sotto-sistema di ingresso/uscita, oltre che per controllare l'interfaccia relativa al banco di memoria A, che non sono qui rappresentati, dovremo considerare l'uso di altri latch, in aggiunta a quelli già presenti, in modo da estendere ulteriormente il numero di linee di I/O del controllore.
Come dicevamo in precedenza, la realizzazione di questa parte del sistema si riconduce alla scrittura del software di gestione, che dovrà, anzitutto, controllare l'interfaccia, seguendo le stesse specifiche caratteri-stiche dei generatori di indirizzi, in modo che si possa sfruttare la RAM interna del controllore come "memoria di programma".
Si potrebbe, cioè, pensare di utilizzare tale spazio di memoria per implementare una mappa del tipo mostrato in Fig. 81
Dato che ogni generatore di indirizzi dispone di una memoria interna per un massimo di 32 istruzioni, dovremo allora, anche in questo caso, prevedere la possibilità di memorizzare fino a 32 istruzioni per il DSP, che, come si vede in figura, possono risiedere nelle prime 32 locazioni della parte utilizzabile della RAM del controllore.

Corrispondentemente ad ogni istruzione, dovremo poi prevedere una serie di registri in cui programmare lo stato dei vari segnali di controllo.

00	ISTRUZIONE 0 (FC+DF)
01	ISTRUZIONE 1 (FC+DF)
1F	ISTRUZIONE 31 (FC+DF)
20	MODO, Istruzione 0
21	MODO, Istruzione 1
3F	MODO, Istruzione 31
40	STATO, Istruzione 0
41	STATO, Istruzione 1
5F	STATO, Istruzione 31
60	CONTROLLO, Istruzione 0
61	CONTROLLO, Istruzione 1
7F	CONTROLLO, Istruzione 31
80	BFPO
81	GESTIONE SISTEMA

Figura 87 - Esempio di come può essere organizzata la memoria interna del 68HC705C8.

Avremo, quindi, si faccia sempre riferimento alla Fig. 87, altre serie da 32 registri in cui memorizzeremo queste informazioni. Poichè potremo configurare tutti i segnali istruzione per istruzione, il controllo sul sistema sarà completo.
Una configurazione del genere, inoltre, può essere tranquillamente ospitata nella RAM del controllore 68HC705C8, dato che occupa solo 130 byte (4x32+2) su 304 disponibili. Anzi, pur tenendo conto che nei 304 byte di memoria devono trovare posto anche lo stack della CPU e le variabili di programma, vi è, con ogni probabilità,

spazio utile per ampliare ulteriormente la serie di registri di controllo del sistema.

Nella configurazione considerata, comunque, ogni registro MODO potrebbe avere i propri bit utilizzati, ad esempio, come segue:

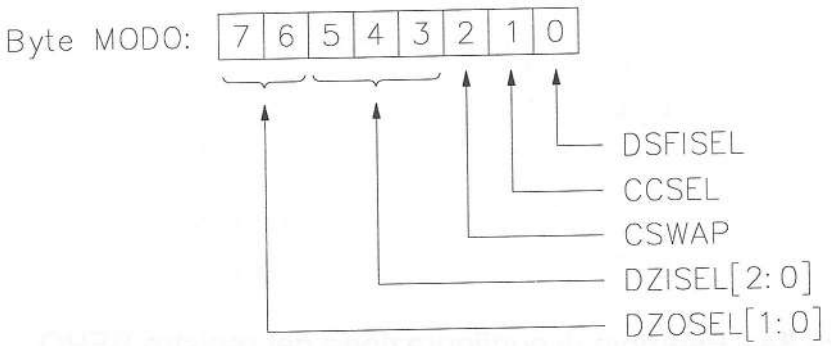


Figura 82 - Esempio di configurazione dei registri MODO.

Ovviamente, ogni bit corrisponde ad una delle linee di uscita nello schema di Fig. 80, le quali pilotano i vari ingressi di controllo del DSP e la logica di contorno che abbiamo visto.
Analogamente, i registri STATO possono avere la seguente configurazione:

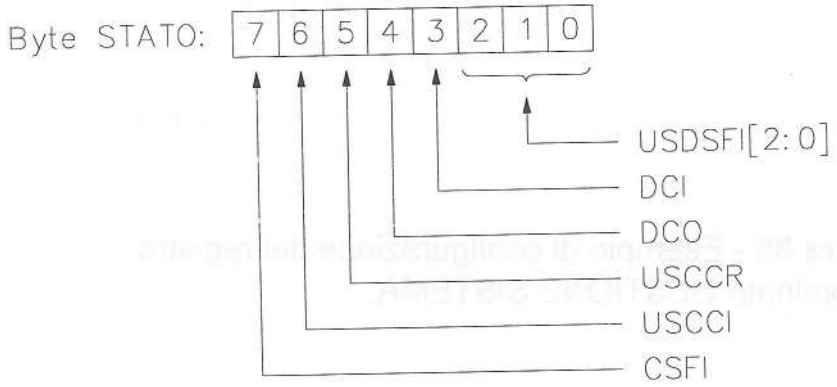


Figura 83 - Esempio di configurazione dei registri STATO.

Per quanto riguarda, invece, i registri denominati CONTROLLO, essi sono destinati alla gestione del sottosistema di ingresso/uscita,

per cui vanno definiti esattamente una volta che il progetto di tale sezione sia stato completato.
Il registro chiamato BFPO serve a leggere lo stato delle corrispondenti linee di uscita del DSP, ed è unico, in quanto l'uso di tale valore ha significato solo al termine dell'elaborazione. Nulla vieta, comunque, di leggere questo registro al termine di ogni passo.

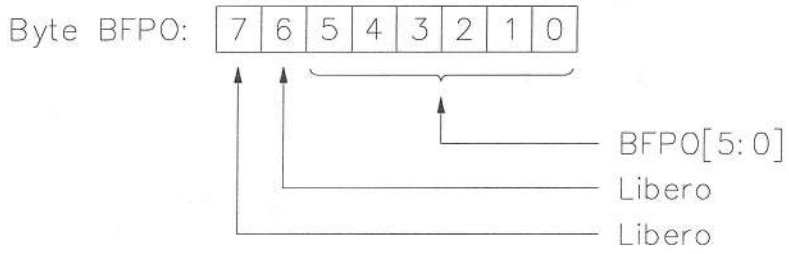


Figura 84 - Esempio di configurazione del registro BFPO.

Infine, scrivendo nel byte denominato GESTIONE SISTEMA, si possono eseguire operazioni quali l'avvio del processo di elaborazione oppure il reset generale di tutto il sistema. La configurazione di questo registro potrebbe essere del tipo mostrato in Fig. 85.

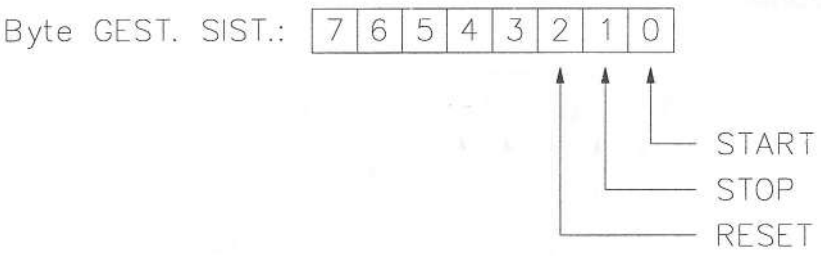


Figura 85 - Esempio di configurazione del registro denominato GESTIONE SISTEMA.

Interfaccia VMEbus

Completiamo la descrizione del nostro sistema esaminando, brevemente, l'interfaccia con il bus VME.
Diciamo subito che questa sezione non è stata studiata a fondo, come invece è stato fatto con il resto del sistema, e questo per diversi motivi.
La ragione principale è che lo sviluppo di una interfaccia VME completa, anche solamente di tipo "Slave", come nel nostro caso, è molto impegnativo e richiede conoscenze specifiche ed esperienza.
Inoltre, è stato piuttosto difficile reperire controller VME integrati, che avrebbero, almeno in parte, risolto il problema.
Infatti, a parte il dispositivo Signetics SCB68172, che sarebbe stato adatto, ma che è ormai uscito di produzione, l'unico controller di cui siamo riusciti a recuperare la documentazione è il VIC068A prodotto dalla Cypress.
Questo componente, però, risulta assolutamente sovradimensionato rispetto alle nostre esigenze, in quanto è stato progettato per essere impiegato in schede di tipo "Master", sfruttando al massimo tutte le potenzialità del sistema VME.
Comunque, anche se non svilupperemo questa interfaccia, vediamo almeno quali sono le funzioni minime che deve realizzare.
In Fig. 86possiamo vedere una sua schematizzazione.

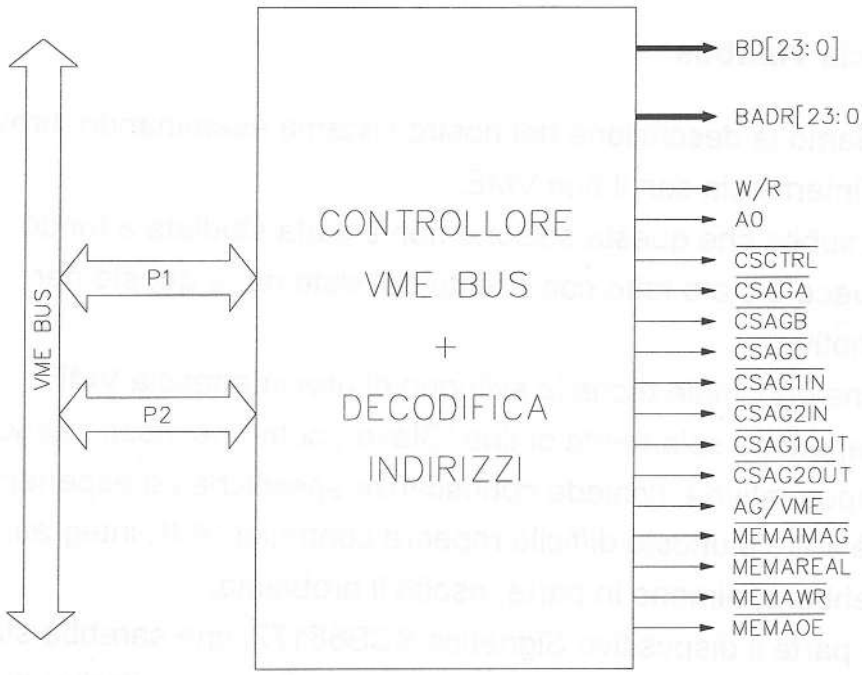


Figura 86 - Schematizzazione dell'interfaccia VMEbus. In evidenza il set minimo di segnali che devono essere resi disponibili.

In figura abbiamo rappresentato con un unico blocco sia l'interfaccia vera e propria, sia la decodifica degli indirizzi.
L'interfaccia vera e propria dovrà fornire, almeno il bus dati, ad almeno 24 bit, per poter leggere e scrivere sul banco di memoria A, il bus indirizzi ed i segnali R/W e A0 (che potrebbe essere, ma non neces-sariamente, il bit meno significativo degli indirizzi).
Dalla linea R/W e dal bus indirizzi, la decodifica, che può essere realizzata con dei dispositivi programmabili (PAL, PLA, GAL, ecc.), ricava poi tutti i segnali di Chip Select, per indirizzare i vari dispositivi.
Nella figura possiamo riconoscere quelli per tutti i generatori di indirizzi (ben sette), i quali sono programmati direttamente via bus VME, e per il controllore di sistema, oltre a quelli che servono per gestire il trasferimento dati nei confronti del banco di memoria A.

Alcune considerazioni pratiche

A questo punto, terminata, finalmente, l'analisi del sistema, possiamo fare alcune "osservazioni" qualitative riguardanti la sua, eventuale, realizzazione.
Come abbiamo potuto notare, si tratta di un circuito notevolmente complesso, con un gran numero di componenti che operano a frequenze molto elevate, e che, inoltre, deve trovare posto preferibilmente su di un'unica scheda VME da 6 unità[1].
Affinchè ciò sia possibile, occorre, anzitutto, contenere il più possibile il numero dei componenti, integrando, ad esempio, l'intero sotto-sistema di temporizzazione e la logica di contorno del DSP entro un unico dispositivo di tipo FPGA (Field Programmable Gate Array).
Inoltre, poichè, anche operando in questo modo, il numero dei componenti rimane elevato, si potrebbe pensare di costruire fisicamente il sistema suddividendolo in due parti, in modo da ottenere una struttura simile a quella mostrata in Fig.87.
La scheda principale ("mother-board"), di dimensioni standard e dotata dei connettori P1 e P2 di collegamento al bus, potrebbe ospitare il circuito principale, formato dal DSP, dai sottosistemi di memoria A, B e C e dai sistemi di controllo e temporizzazione, oltre, ovviamente, all'interfaccia VME.
La scheda secondaria ("daughter-board"), potrebbe invece ospitare l'intero sottosistema di ingresso/uscita (memoria, generatori di indirizzi e relativa logica di controllo), e verrebbe montata a ridosso della scheda principale, alla quale rimarrebbe fissata grazie agli stessi connettori di collegamento ("piggy-backing").
Grazie a questo sistema, peraltro per nulla originale e largamente utilizzato, si potrebbe limitare ad uno solo il numero di *slot* occupate nel sistema VME.

---

1 Sei unità rack (6U): indica l'altezza della scheda, pari a circa 230 mm, mentre la profondità è fissata a 160 mm.

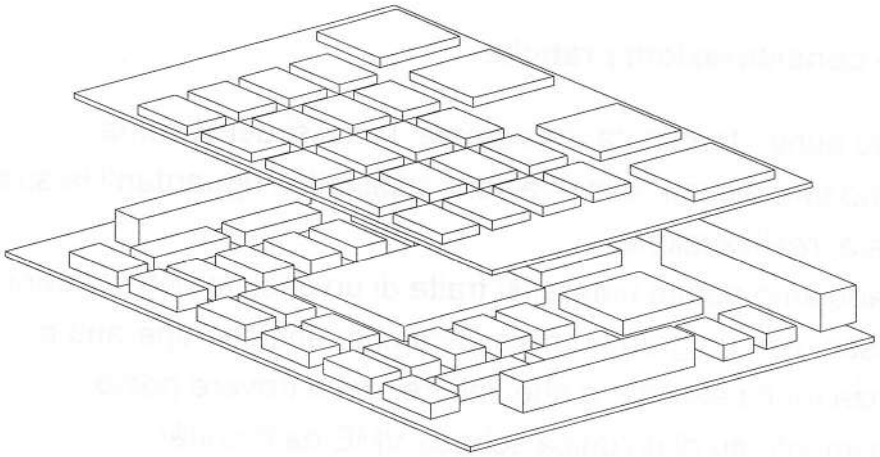


Figura 87 - Esempio di una possibile soluzione costruttiva per il sistema.

Infine, è necessario fare alcune considerazioni riguardo al consumo di corrente, che risulta piuttosto elevato soprattutto a causa della grande quantità di RAM (statica) impiegato.
Poichè l'idea è, come sappiamo, quella di utilizzare dispositivi non ancora disponibili sul mercato, possiamo solo fare delle stime riguardo al loro consumo di corrente.
Mentre la Samsung non fornisce ancora alcun dato certo a questo proposito, la Toshiba, che pure ha annunciato dispositivi analoghi, dichiara (nel relativo "Tentative Data Sheet"...) per il componente TC551402J (una RAM statica da 4Mbit, configurabile in 4Mx1 o 1Mx4) un consumo approssimativo di 150mA, a 5V.
Considerando questo valore, *la sola memoria del nostro sistema*, che, come abbiamo visto in precedenza, impiega ben 45 dispositivi da 4Mbit, assorbe almeno **6.75A** a 5V, che è da sola, una corrente prossima a quella massima trasferibile attraverso i connettori P1 e P2 del bus VME, che è di circa 7.2A.
Se a questo valore aggiungiamo la corrente assorbita dagli altri dispositivi (oltre 800mA, ad esempio, per il DSP LH9124), andiamo sicuramente oltre i limiti massimi.
Per ovviare a questo inconveniente, si potrebbe pensare di alimentare il sistema attraverso un apposito connettore, posto, ad esempio, sul pannello frontale, oppure suddividere la corrente, nei

connettori VME, su di un numero maggiore di pin (rispetto ai 6 standard) sfruttando quelli lasciati liberi per applicazioni particolari, nel connettore P2.

(file: shar7.doc)

CAP. 6

SOLUZIONI CIRCUITALI SHARP.

Prima di terminare il capitolo sulla architettura circuitale di utilizzo del chip set, si esegue una rapida carellata a quelli che sono in questo senso i suggerimenti della SHARP.
Nel paragrafo che segue, vengono, infatti,presentati i suggerimenti dati dalla stessa SHARP nel volume LH9124 Digital Signal Processor User's Guide. La configurazione base viene riportata in fig. 88. Si puo` subito notare come l' address generator LH9320 sia di vitale importanza nella generazione degli indirizzi per le relative memorie. La memoria dedicata alla acquisizione, quella cioe` che sta tra il convertitore ed il sistema DSP, in particolare puo` essere costituita da memorie SRAM, da FIFO o entrambi. Nelle applicazioni standard, si usano gli address generator LH9320 per la generazione dei pattern degli indirizzi delle memorie delle porte A, B e C lasciando al caso singolo la scelta dell'utilizzo o meno dello stesso nel banco di memoria della porta Q a seconda che si usino SRAM o FIFO. Nello schema di principio di fig.88, ogni address generator e` configurato per:

- contenere le linee per le richieste dei dati dai canali di ingresso.
- fare uso dello stesso segnale di clock SYSCLK, usato dal DSP LH9124.
- ricevere i segnali di controllo dallo schedulatore (controller).
- ricevere il controllo degli Address Register / Data Register dal controllore.
- essere collegato, via apposito bus, al controllore.
- avere connessioni al controllore separate, di TC e CS/.
- provvedere all' LH9124 il segnale di PO.
- provvedere il clock di lettura RDCLK al DSP.
- Comunicare con le SRAM attraverso i segnali MEMW/ e MEMOE/.
- dividere il bus comune ed i segnali di R/W.

Nello schema di fig. 88, ogni LH9320 e` inizializzato e controllato attraverso una serie di comandi programmabili provenienti dal controllore (SCHEDULER). Il segnale di START generato dal controllore, fa partire le operazioni di generazione della sequenza attuale di indirizzamento e subito dopo che questa e`stata generata l' LH9320 genera un segnale di TC (Terminal Count) mettendosi in attesa del prossimo segnale di START. In relazione alla funzione corrente il segnale relativo all' uscita programmabile (PO), puo` essere mandato o al controllore o ai pin di ingresso DCI,DCO,DZI or DZO del DSP LH9124. Se e` richiesto solo uno di questi segnali di controllo sulla manipolazione dei coefficienti (coniugazione e cambio segno) e "zero padding" dell' I/O , gli altri possono essere collegati insieme o

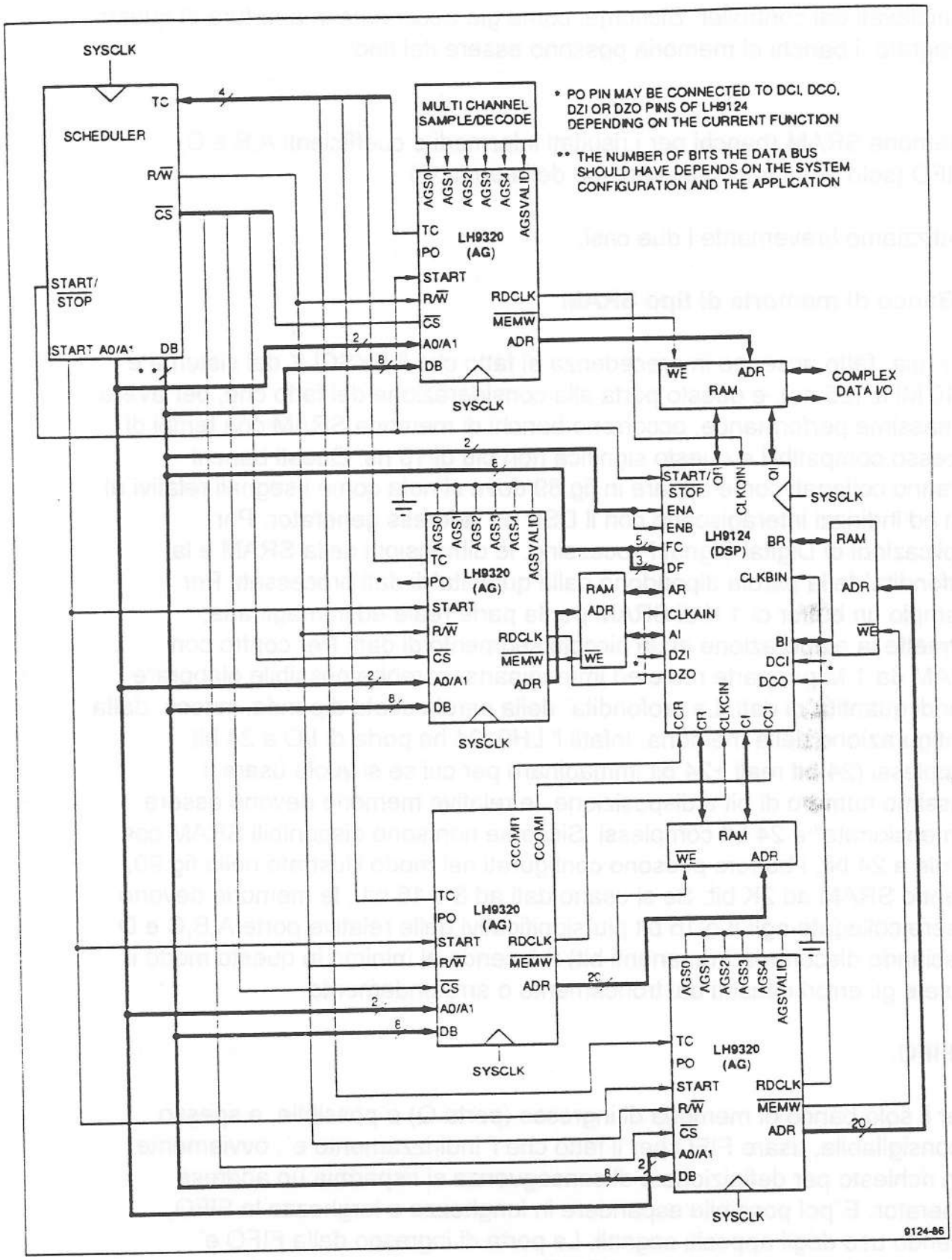

SYSCLK
SCHEDULER
TC
R/W
CS
START/
STOP
START A0/A1 DB
MULTI CHANNEL
SAMPLE/DECODE
* PO PIN MAY BE CONNECTED TO DCI, DCO, DZI OR DZO PINS OF LH9124 DEPENDING ON THE CURRENT FUNCTION
** THE NUMBER OF BITS THE DATA BUS SHOULD HAVE DEPENDS ON THE SYSTEM CONFIGURATION AND THE APPLICATION
AGS0 AGS1 AGS2 AGS3 AGS4 AGSVALID
LH9320 (AG)
PO
START
RDCLK
MEMW
ADR
A0/A1
DB
WE
RAM
COMPLEX DATA I/O
LH9124 (DSP)
ENA
ENB
FC
DF
AR
CLKAIN
AI
DZI
DZO
CLKBIN
BR
BI
DCI
DCO
CCR
CR
CLKCIN
CI
CCI
CCOMR
CCOMI
9124-86

FIG. 88

multiplexati dal controller. Siccome, come gia`accennato in apertura di questo paragrafo, i banchi di memoria possono essere del tipo:

- Memorie SRAM (banchi per i risultati intermedi e coefficienti A,B e C)
- FIFO (solo per il banco di ingresso della porta Q)

Analizziamo brevemente i due casi:

1) Banco di memoria di tipo SRAM

Si e` gia` fatto accenno in precedenza al fatto che il SYSCLK del sistema e` di 40 MHz (25 ns), e questo porta alla considerazione del fatto che, per avere le massime performance, occorrono banchi di memoria SRAM con tempi di accesso compatibili e questo significa non piu di 15 ns. Questi banchi saranno collegati come appare in fig.89 dove si nota come i segnali relativi ai dati ed indirizzi interagiscano con il DSP e l' address generator. Per applicazioni di Digital Signal Processing. le dimensioni della SRAM e la profondita`della parola dipendono dalla quantita`di dati processati. Per esempio un buffer di 1 K di SRAM per la parte reale ed immaginaria, permette la elaborazione di un piccolo segmento di dati. Per contro con SRAM da 1 M per parte reale ed immaginaria, sarebbe possibile elaborare grandi quantita`di dati. La profondita` della parola usata dipende, invece, dalla configurazione della memoria. Infatti l' LH9124 ha porte di I/O a 24 bit complessi (24 bit reali +24 bit immaginari) per cui se si vuole usare il massimo numero di bit a disposizione, le relative memorie devono essere "dimensionate" a 24 bit complessi. Siccome non sono disponibili SRAM con parole a 24 bit, i buffers possono configurati nel modo illustrato nella fig.90, usando SRAM ad 2K bit. Se si usano dati ad 8 o 16 bit , le memorie devono essere collegate agli 8 o 16 bit piu`significativi delle relative porte A,B,C e D (lasciando disconnesi i rimanenti bit) riducendo al minimo in questo modo il noise e gli errori causati dal troncamento o arrotondamento.

2) FIFO.

Per il solo banco di memoria di ingresso (porta Q) e`possibile, e spesso e`consigliabile, usare FIFO per il fatto che l' indirizzamento e`, ovviamente, non richiesto per definizione e di conseguenza si risparmia un address generator. E`poi possibile espandere in lunghezza e larghezza la FIFO, facendo uso degli appositi segnali. La porta di ingresso della FIFO e` collegata al blocco convertitore A/D attraverso un opportuno data-bus, mentre la porta di uscita della stessa viene collegata alla porta Q dell'LH9124. In genere le FIFO disponibili sono da 8 bit (width) e da 512 ad un massimo di 4/8 K (depth), per cui se si rende necessaria una configurazione a più di 8 bit (word width expansion), la disposizione circuitale diventa quella riportata in

fig.91 ottenuta ponendo più dispositivi in parallelo. Se la particolare configurazione richiede invece FIFO con profondità maggiori, il metodo di espansione è quello riportato in fig.92 dove ogni FIFO singola è collegata in una "circular fashion" che vede il pin Expansion OUT (XO) di ciascuna, collegata al pin Expansion IN (XI) di quella dopo. Se la particolare applicazione richiede una configurazione con FIFO espansa in larghezza e profondità, l' architettura di questa risulta essere quella riportata in fig.93.

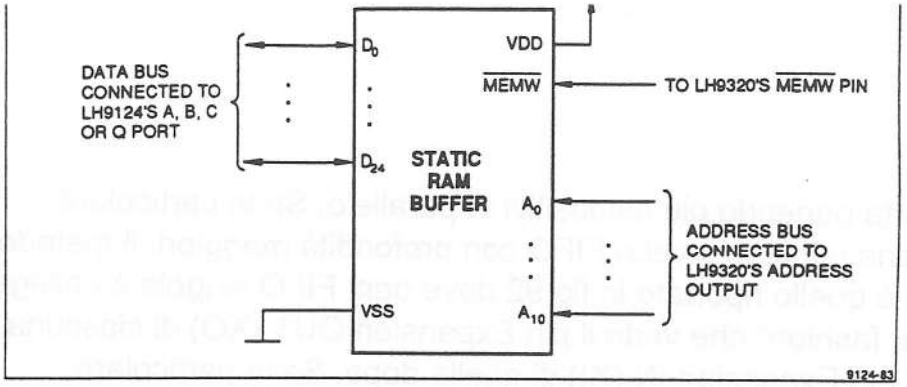


Figure 89 SRAM Buffer

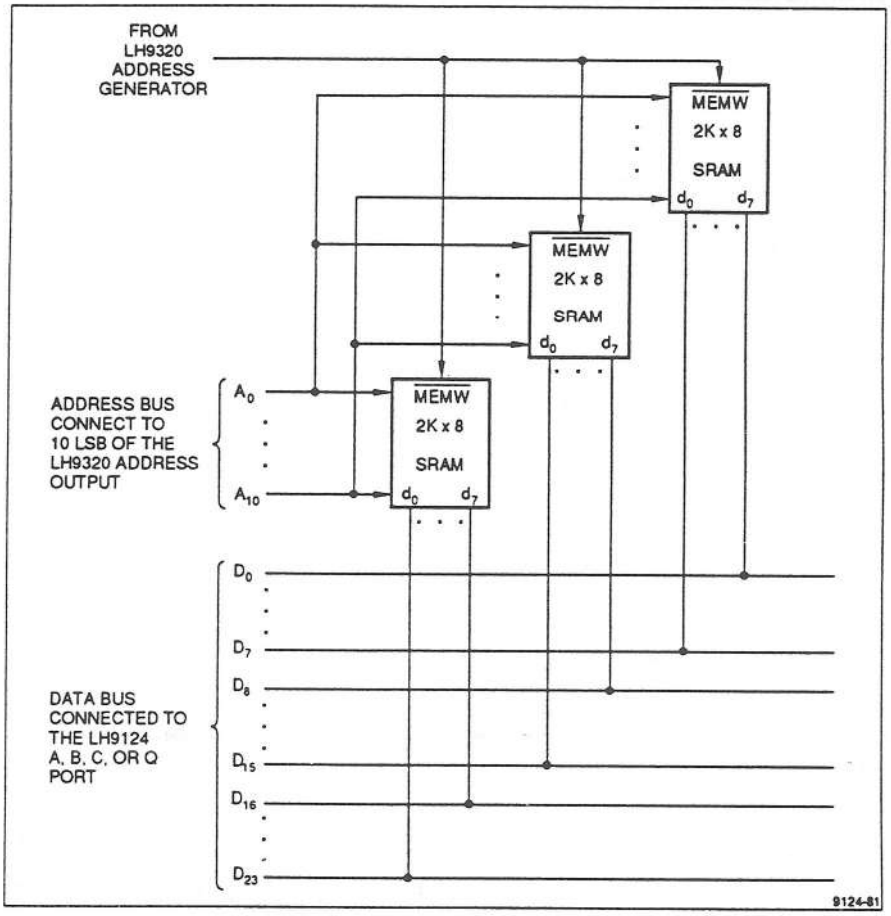


Figure 90 SRAM Word Width Expansion

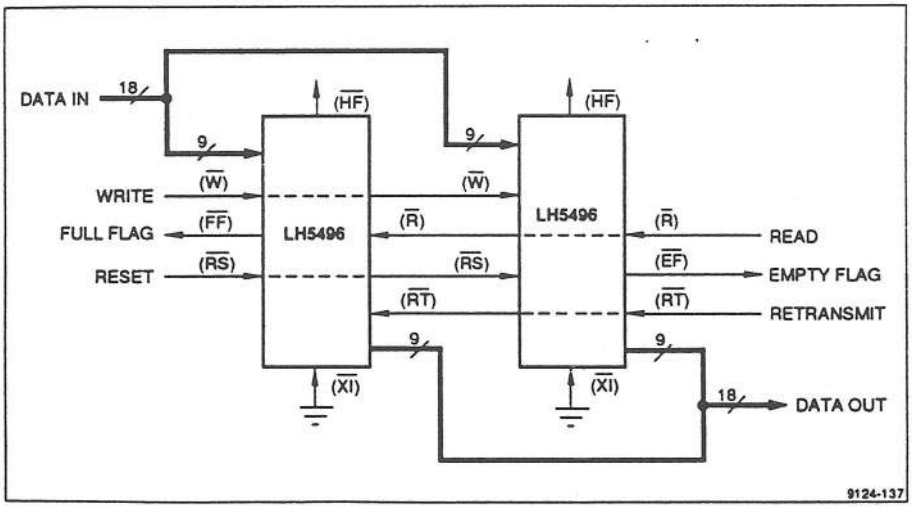


Figure 91 FIFO Word Width Expansion

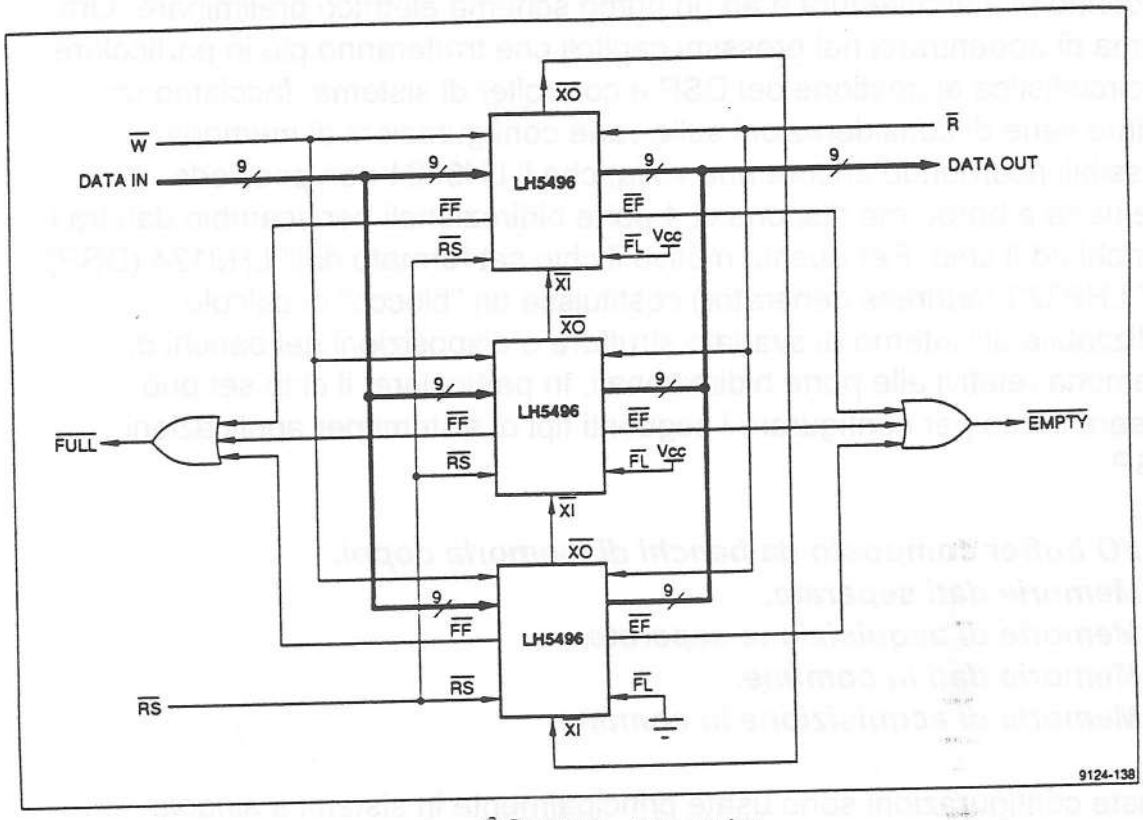


Figure 92 FIFO Depth Expansion

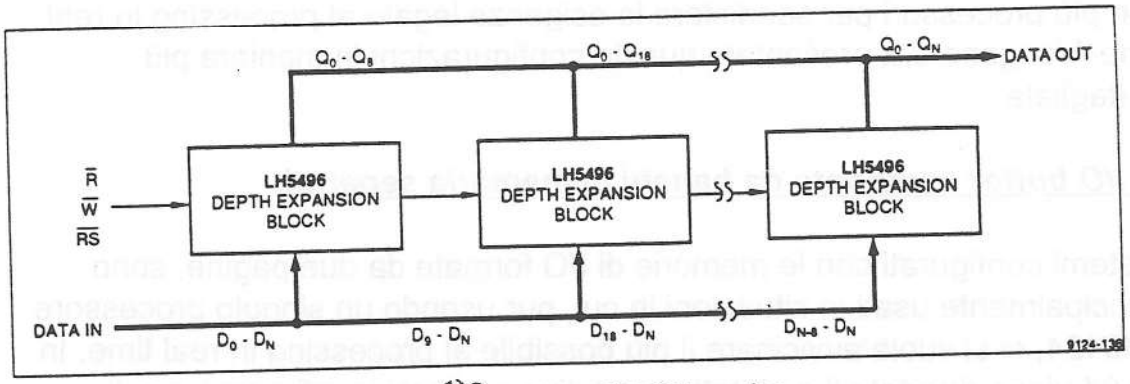


Figure 93 Compound FIFO Expansion

(file: shar8.doc)

CONFIGURAZIONI DEI BUFFER DI MEMORIA

Fino a questo punto, per quello che riguarda l' hardware, si è fatto un accenno alla architettura e ad un primo schema elettrico preliminare. Ora, prima di addentrarci nel prossimi capitoli che tratteranno più in particolare la circuitistica di gestione del DSP e controller di sistema, facciamo un' ultima serie di considerazioni sulle varie configurazioni di memoria possibili ricordando ancora una volta, che l' LH9124 non possiede memoria a bordo ma dispone di 4 porte bidirezionali per scambio dati tra i banchi ed il chip. Per questo motivo il chip set formato dall' LH9124 (DSP) e l' LH9320 (address generator) costituisce un "blocco" di calcolo utilizzabile all' interno di svariate strutture o disposizioni dei banchi di memoria relativi alle porte bidirezionali. In particolare, il chip set può essere usato per configurare i seguenti tipi di sistemi per applicazioni DSP:

1) I/O buffer composto da banchi di memoria doppi.
2) Memorie dati separate.
3) Memorie di acquisizione separate.
4) Memoria dati in comune.
5) Memoria di acquisizione in comune.

Quste configurazioni sono usate principalmente in sistemi a *singolo processore;* vedremo più avanti come sarà possibile implementare sistemi con più processori per soddisfare le esigenze legate al processing in real time. Vengono ora presentate queste configurazioni in maniera più dettagliata.

1) I/O buffer composto da banchi di memoria separati.

Sistemi configurati con le memorie di I/O formate da due pagine, sono principalmente usati in situazioni in cui, pur usando un singolo processore LH9124, ci si vuole avvicinare il più possibile al processing in real time. In fig.94 viene riportata l' architettura relativa a questa configurazione di memoria, dove si vede come i due buffer di I/O (collegati alle porte bidirezionali A e B) siano composti da due pagine ciascuno (real A00, Imag. B00 e Real A01 e Imag.A01 per la porta A e Real B00, Imag. B00 e Reai B01 e Imag. B01 per quella B) utilizzabili a coppie per caricare e scaricare i dati mentre i due rimanenti come memoria per i risultati intermedi all' interno del passo in corso, scambiandosi poi i ruoli nel passo successivo. In altri termini questa configurazione permette di operare in maniera molto efficiente, commutando in continuazione i banchi e le

relative funzioni. Infatti, osservando la fig.95 si nota che per un passo i dati vengono caricati e scaricati attraverso i due banchi esterni (outside set) Real A00, Imag. A00 e Real B00, Imag. B00 mentre vengono processati i dati nei due banchi più interni (Inside set). Nel passo successivo, l' inside set viene caricato (Real A01 e Imag. A01) e scaricato (Real B01 e Imag. B01), mentre i dati introdotti nella precedente situazione, vengono processati attraverso l' outside set e così via. Questa configurazione permette il calcolo di una FFT di 1K punti in 80.7 us usando tre passi del tipo Radix-4 x Radix-16 x Radix-16. Questa configurazione richiede 5 address generator LH9320 tra cui i quattro relativi alle doppie pagine di ingresso/uscita sono gestiti dal controllore che sovraintende alla commutazione dei banchi durante le operazioni di processing dei dati. Più in particolare, ciascuno di questi buffer necessita di un LH9320 che fornisce al proprio buffer di memoria, i segnali WE/, il clock CLKxIN (con x=A,B o C) e le relative sequenze di indirizzamento.

2) Memorie dati separate.

Questo tipo di organizzazione dei banchi di memoria attorno al DSP, è una configurazione abbastanza comune nelle applicazioni a singolo processore in cui si utilizzano in pieno le 4 porte di I/O dell' LH9124, come riportato in fig. 96, formando un sistema abbastanza efficiente. Ciascuna porta è collegata, come già detto più volte, al proprio banco di memoria "complessa" come nella tabella che segue:

PORTA LH9124	Buffer Real	Buffer Immaginary
Q00	Real Q00	Immag. Q00
A00	Real A00	Immag. A00
B00	Real B00	Immag B00
C00	Real C00	Immag. C00

Alla partenza dell 'esecuzione dell' algoritmo, il dato viene introdotto nel chip leggendolo dalla memoria A00 attraverso la porta Q. Al segnale di START i dati vengono trasferiti dalla porta Q alla porta A con una istruzione RQWA (Read Q Write A). Nel passo successivo i dati vengono elaborati secondo la funzione richiesta ed immagazzinati, come risultato parziale, nella memoria complessa B00 relativa a B. Lo stesso meccanismo si applica nel passo successivi che vede, però, il flusso dei dati invertirsi nel senso che i dati sono letti da B00, elaborati e scritti su A00 caricando i coefficienti dalla porta C, se richiesto. Questo processo continua così fino alla fine dell' algoritmo, trasferendo i dati

recursivamente tra A0 e B0 e viceversa. Ogni banco di memoria è controllato da un suo proprio address generator abilitato alla generazione della sequenza di indirizzamento, dal controllore. Nella tabella che segue vengono riportati i programmi per il DSP ed address generators per la convoluzione di 1024 punti nel dominio delle frequenze.

3) Memorie di acquisizione separate.

In un sistema a singolo processore, si può usare anche questo tipo di configurazione che ricorda da vicino la precedente, come riportato in fig. 97. Il doppio buffer, in questo caso, è collegato alla porta "naturale" di ingresso al chip, cioè la porta Q, e necessita di due set separati di memorie uno doppio per l' ingresso ed un altro doppio per l' uscita come segue (buffer complessi):

Input Data: **Q00 (Real Q00, Immag.Q00)**
Output Data: **Q01 (Real Q01, Immag.Q01)**

mentre la configurazione delle memorie relativa alle altre porte (A,B e C) rimane invariata. In questa disposizione i dati vengono acquisiti nel buffer Q00 e trasferiti (con un comando del tipo RQWA) alla memoria della porta A al sopraggiungere del segnale di START/STOP/. Nel passo successivo, il DSP processa i dati secondo la funzione richiesta riversandoli poi in B ed eventualmente usando i coefficineti, presi dalla porta C, se richiesti. Nei passi successivi il DSP esegue le ulteriori funzioni richieste trasferendo recursivamente i dati tra A e B. Nel passo finale i dati vengono scaricati sul buffer complesso di uscita Q01 e da qui verso l' esterno sotto la gestione del controllore. Ogni pagina di memoria (Q00 e Q01) è gestita da un proprio LH9320. Il primo ha il compito di eseguire un efficiente overlapping dei dati in ingresso mentre il secondo un indirizzamento adatto al "data discard".

4)Memorie dati comuni

Questa è una configurazione che fornisce prestazioni ridotte nei confronti delle precedenti, anche se ha il vantaggio di ridurre il numero degli addres generator necessari (uno in meno) come si può notare in fig. 98. Questa configurazione usa una "dual port memory", che afferisce alle due porte A e B, ed è molto adatta ad algoritmi per la FFT *in place* (il risultato del passo corrente viene memorizzato nella stessa memoria dove erano i precedenti risultati). Come si può notare dallo schema, se la porta A è configurata per leggere dal buffer A01 (Real A01, Immag. A01), quella B

deve essere configurata per scrivere in A00 (Real A00, Immag. A00). I8 segnali CLKAIN e SYSCLK, provvedono alla sincronizzazione. Durante il processing, l' array dei dati viene caricato, attraverso il banco Q00 che puo` essere configurato in uno dei modi appena descritti, al giungere del segnale di START. I dati vengono poi mandati alla memoria A00 e processati secondo la funzione richiesta poi, uscendo dalla porta B, il risultato viene memorizzato in A01. Se il passo richiede l' uso di coefficienti, questi vengono presi, tramite C, dal Buffer complesso C00. Dopo l' ultimo passo, i risultati vengono mandati verso l' esterno attraverso la solita porta Q, sotto il controllo dello "scheduler".

5) Memoria di acquisizione in comune.

Questa configurazione e simile alla precedente, ma per quello che riguarda la memoria di acquisizione (porta Q), come visibile in fig. 99. In questo caso e` la memoria di acquisizione ad essere usata parzialmente per l' Input dei dati e parzialmente per l'Output dei risultati ed e`gestita da un solo address generator. Anche in questo caso, come nel precedente, si possono usare banchi tipo "dual memory". Durante il processing i dati sono inizialmente trasferiti dalla memoria Q00 al DSP, attraverso la porta Q e, al sopraggiungere del segnale di START, traferiti (con o senza calcoli) al banco A00. Nel passo successivo, come gia` visto in precedenza, il DSP esegue la funzione richiesta e scarica i risultati sul banco di memoria B00 usando, se necessario, i coefficienti presi dalla porta C. Il processing va avanti in questo modo fino all'esecuzione dell' ultimo passo dopo il quale, i dati vengono mandati verso l' esterno attraverso Q ed il buffer Q00. Da questo punto in poi i dati passano sotto il controllo dello scheduler.

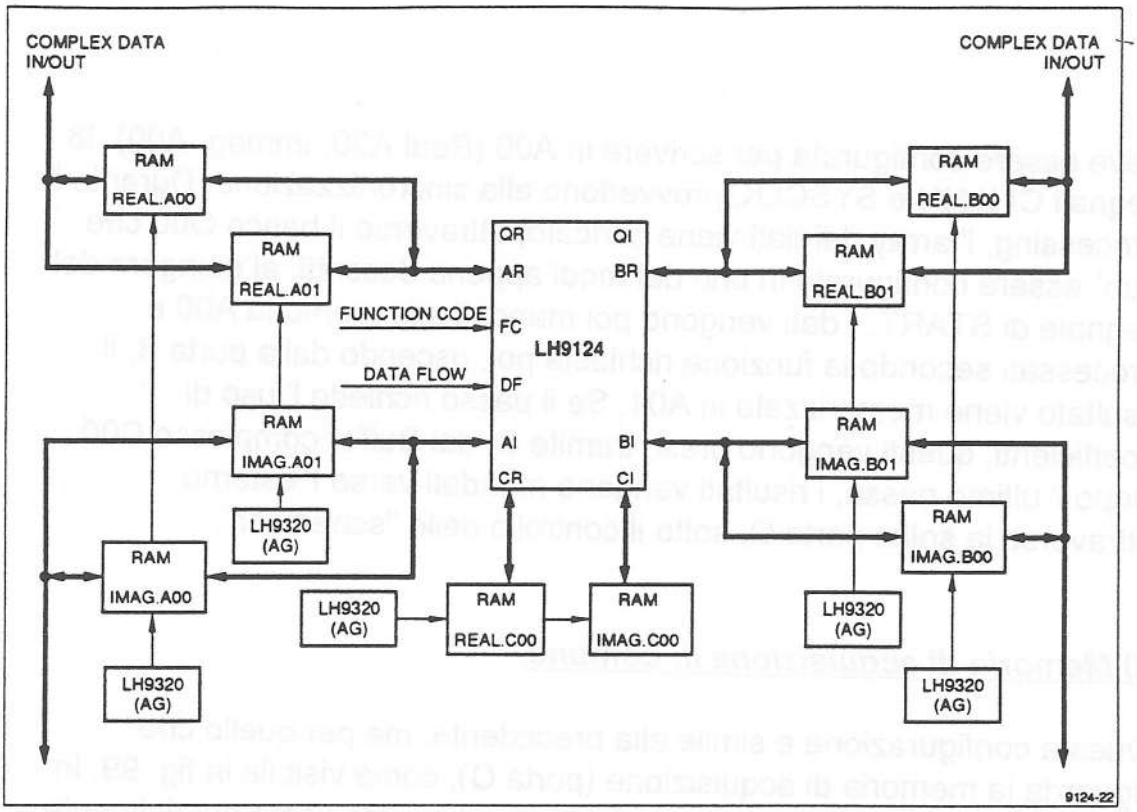


Figure 94 Dual Memory I/O Buffer Configuration

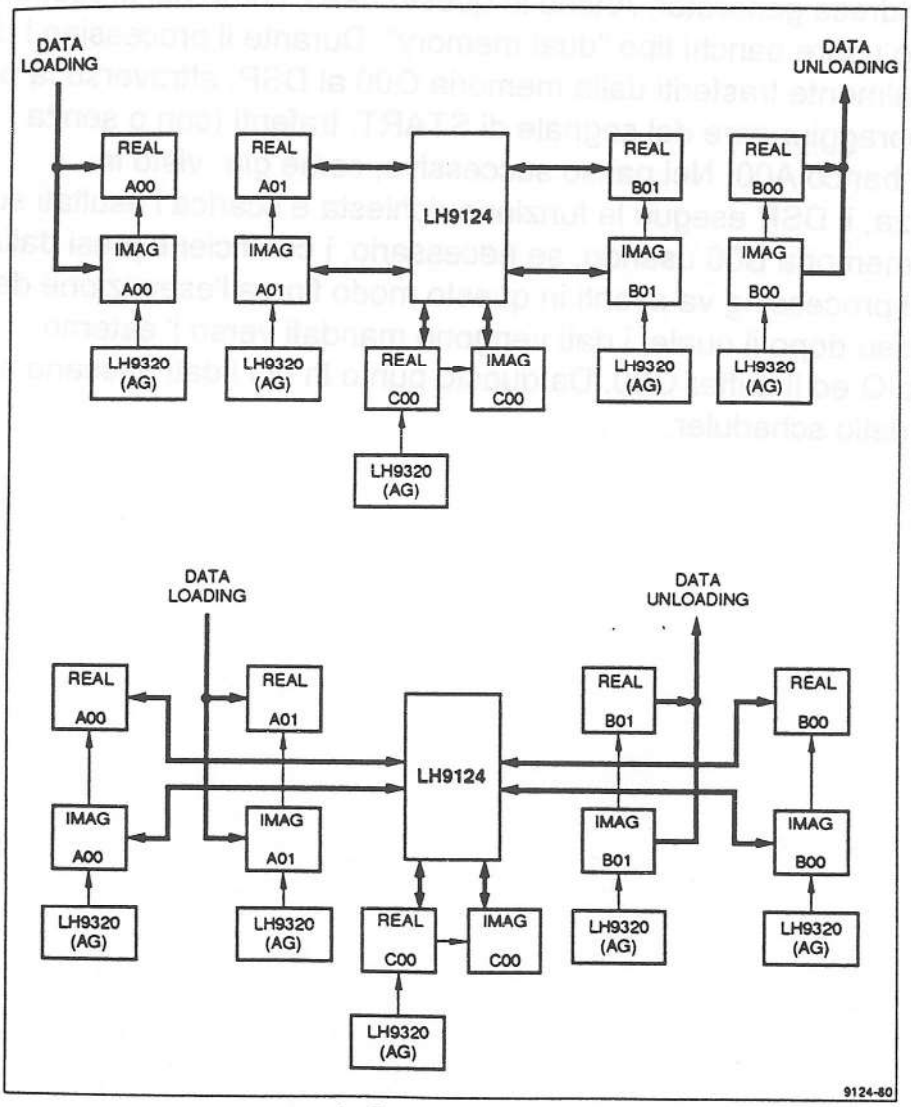


Figure 95 Memory Reallocation for Dual Memory I/O Buffer

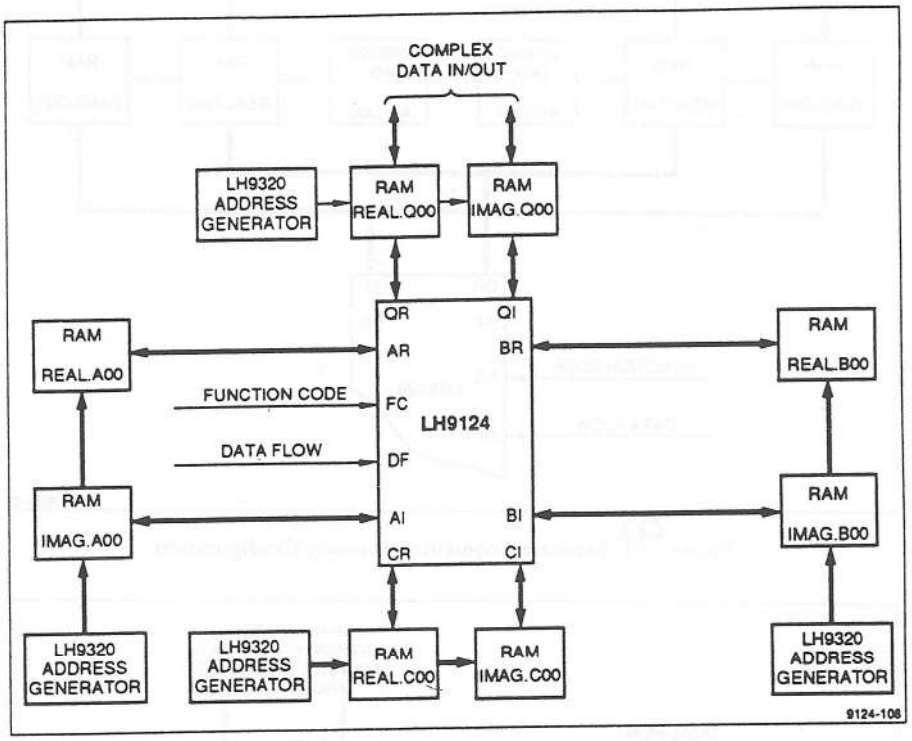


Figure 96 Separate Data Memory Configuration

Table 6-5
1,024-Point Frequency Domain Convolution Program

PROGRAM FOR LH9124				PROGRAM FOR LH9320 *				
DCI	DCO	FC	DF	AG_Q00 (INPUT)	AG_A00	AG_B00	AG_C00	AG_Q01 (OUTPUT)
0	0	MOVD	RQWA	OVERLAP	INC **	X	X	X
0	0	BWND4 ♦	RAWB	X	RBF0	BF40 **	INC	X
0	0	BFLY4	RBWA	X	BF41 **	BF41	TF41	X
0	0	BFLY4	RAWB	X	BF42	BF42 **	TF42	X
0	0	BFLY4	RBWA	X	BF43 **	BF43	TF43	X
0	0	BFLY4	RAWB	X	BF44	BF44 **	TF44	X
1	0	BWND4 †	RBWA	X	BF40 **	RBF0	INC	X
0	0	BFLY4	RAWB	X	BF41	BF41 **	TF41	X
0	0	BFLY4	RBWA	X	BF42 **	BF42	TF42	X
0	0	BFLY4	RAWB	X	BF43	BF43 **	TF43	X
0	1	BFLY4	RBWA	X	BF44 **	BF44	TF44	X
0	0	MOVD	RAWQ	X	INC	X	X	DISCARD **

* AG = Address patterns for the Q00 (input), A00, B00, C00 and Q01 (output) memory
** Latency compensated
♦ Wd – Window coefficient
† Resp – Frequency response

Note: an inverse fast Fourier transform (IFFT) conjugates the input data (DCI) in the first pass, and conjugates the output data (DCO) on the last pass.

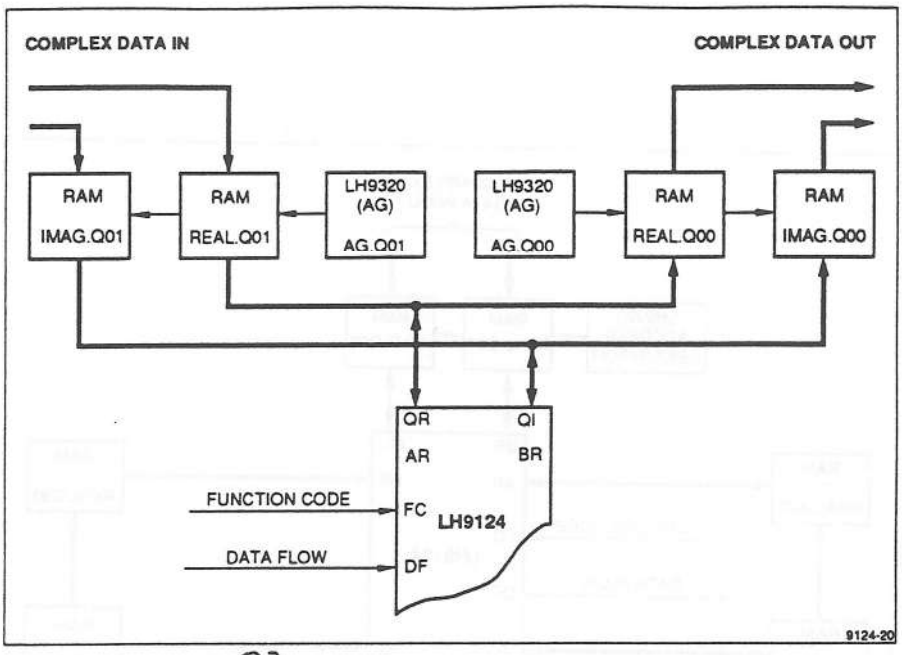


Figure 97 Separate Acquisition Memory Configuration

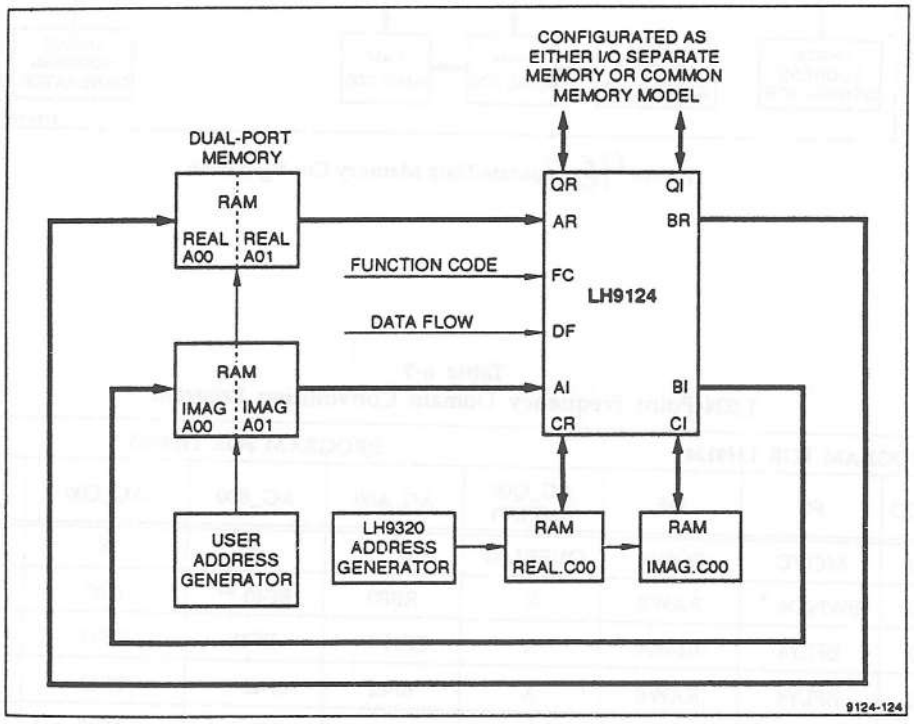


Figure 98 Common Data Memory Configuration

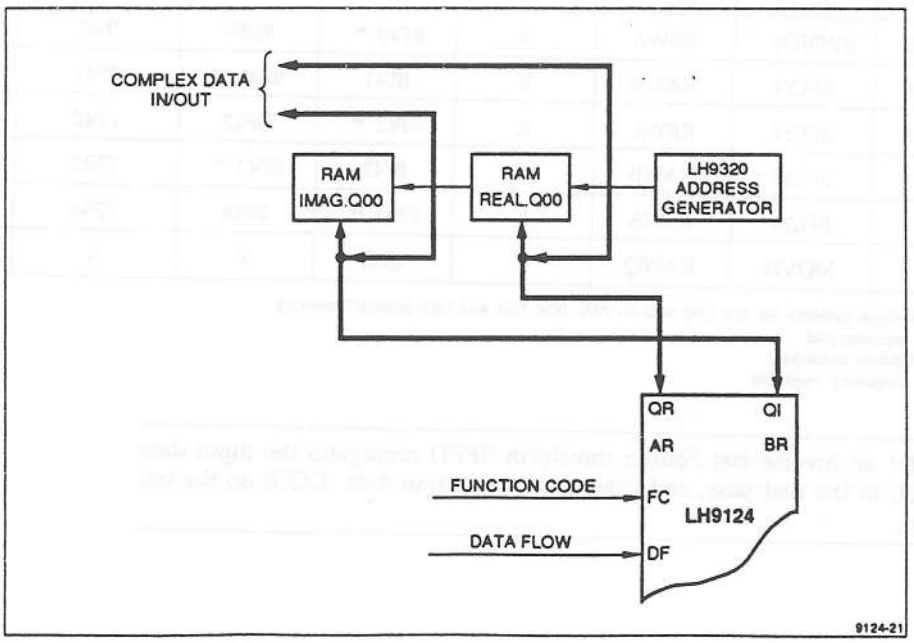
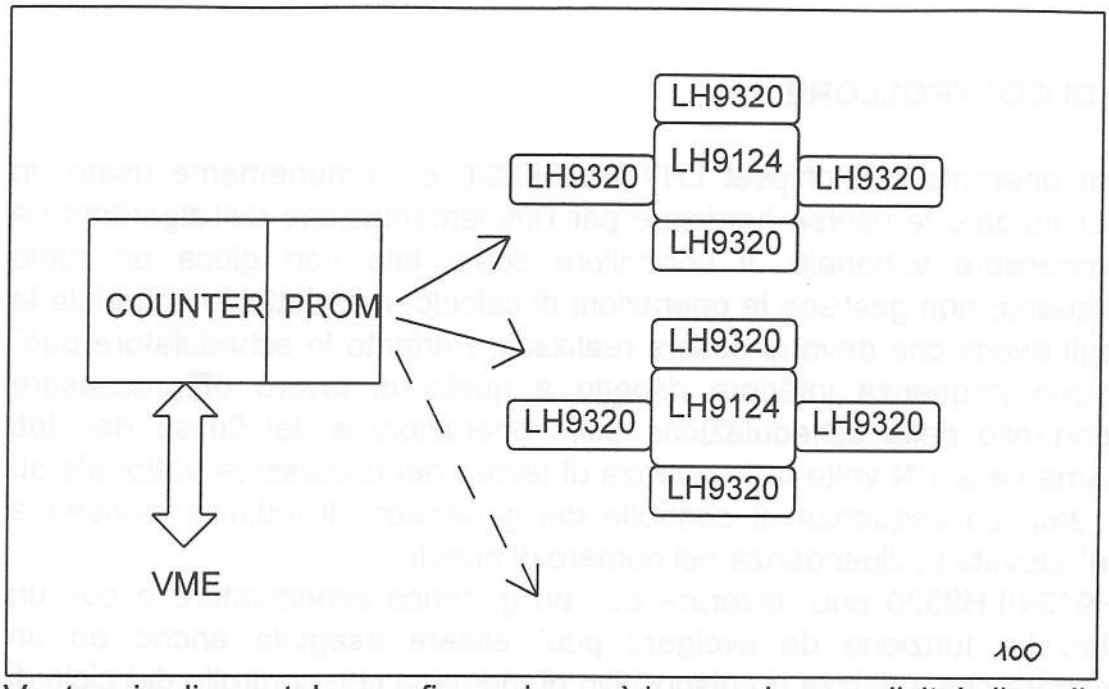


Figure 99 Common Acquisition Memory Configuration

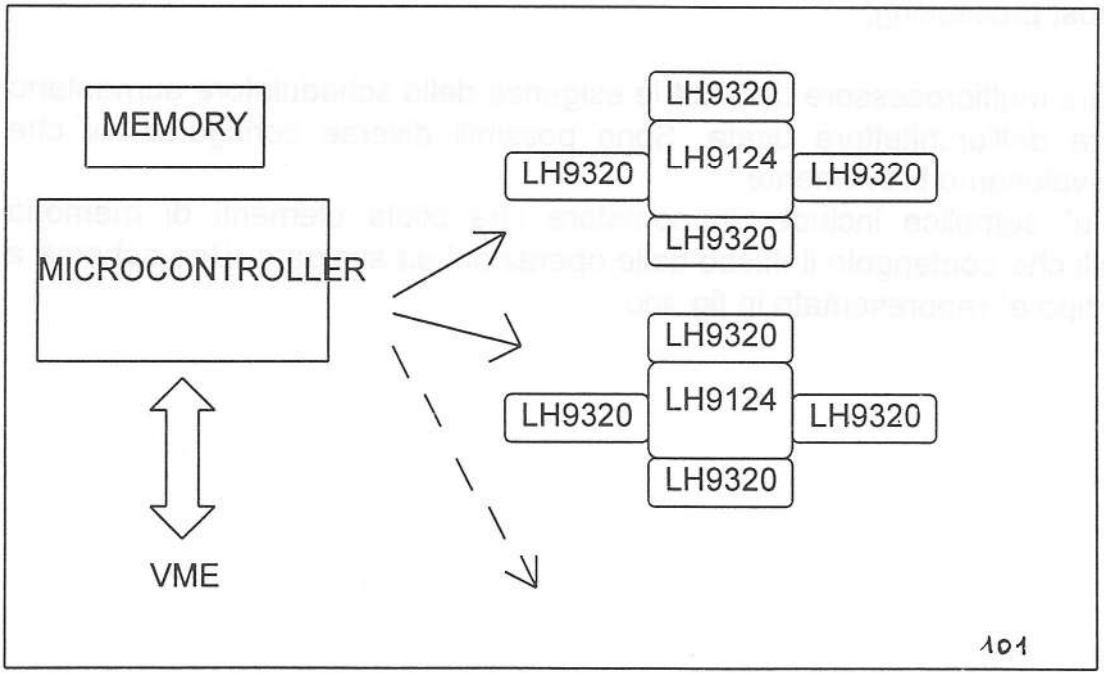
PROPOSTA DI CONTROLLORE

In un sistema operante con chipset LH9124/LH9320 e` comunemente usato un controllore per guidare le risorse hardware per l'implementazione dell'algoritmo da parte del processore vettoriale. Il controllore come tale non gioca un ruolo sofisticato in quanto non gestisce le operazioni di calcolo, ma piuttosto schedula la sequenza degli eventi che devono essere realizzati. Pertanto lo schedulatore puo` lavorare ad una frequenza inferiore rispetto a quella di lavoro del processore LH9124. Il controllo della schedulazione delle operazioni e del flusso dei dati avviene tipicamente a 1/N volte la frequenza di lavoro del processore vettoriale su un set di N dati. La sequenza di controllo dei generatori di indirizzi avviene a frequenza piu` elevata in dipendenza del numero di questi.
Il chipset LH9124/LH9320 puo` lavorare con un generico schedulatore o con un microcontroller. La funzione da svolgere puo` essere eseguita anche da un semplice contatore che indirizza un dispositivo di memoria con controllo del ciclo di esecuzione. In particolare un generico schedulatore deve essere in grado di generare codici di controllo arbitrari, gestire l'avvio delle operazioni e riavviare il programma in ogni fase di esecuzione.
Le funzioni che uno schedulatore o controllore devono svolgere sono:
- inizializzazione del LH9124;
- inizializzazione della generazione della sequenza di indirizzi;
- controllo di input/output dei dati;
- eventuale post processing;
- diagnostica.

In un sistema a multiprocessore LH9124 le esigenze dello schedulatore aumentano in dipendenza dell'architettura usata. Sono possibili diverse configurazioni che esponiamo e valutiamo brevemente.
Il metodo piu` semplice include un contatore che pilota elementi di memoria programmabili che contengolo il flusso delle operazioni da svolgere. Uno schema a blocchi di tal tipo e` rappresentato in fig. 4



Vantaggio di una tale configurazione e` la grande semplicita` di realizzazione con utilizzo molto limitato di spazio su scheda. Svantaggio la scarsa flessibilita` e la mancanza di soddisfacimento di parte delle richieste sopra elencate. Un approccio piu`sofisticato puo`prevedere un controllore a bordo della scheda con un suo spazio di indirizzamento. Lo schema a blocchi nella fig.101



I vantaggi di una tale configurazione includono il soddisfacimento di tutte le richieste. Svantaggi sono la complicazione su scheda, la richiesta di sincronizzazione delle operazioni.
Una terza possibilita` include línserimento di uno schedulatore alla configurazione appena descritta. Al microcontroller sono in questo caso demandati i compiti di programmazione e accessori descritti.
Lo schedulatore e` usato per generare la sequenza delle operazione nelle diverse sezioni di elaborazione. I vantaggi del sistema sono la grande flessibilita` e la elevata velocita` di esecuzione raggiungibile. Svantaggi sono una forte complicazione e notevole spazio occupato sulla scheda. Lo schema a blocchi e`raffigurato in fig.102

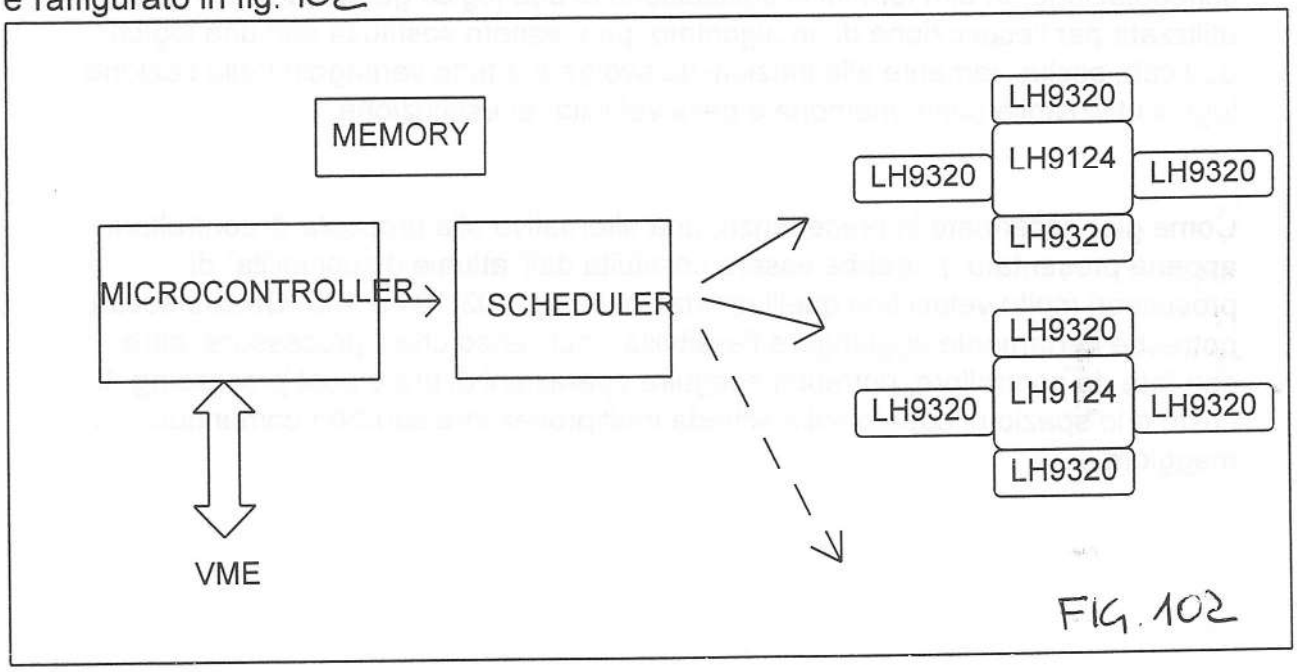


FIG. 102

Una quarta possibilita` include un controllore esterno su operante su bus con accesso moderatamente veloce tra le due schede. Le operazioni veloci sono anche in tal caso demandate ad uno schedulatore interno. Le operazione di programmazione e di servizio sono affidate al processore su scheda. I vantaggi di tale soluzione comprendono un utilizzo limitato dello spazio su scheda con processori vettoriali, una complessita` modesta. Svantaggio e` il costo del sistema computer aggiuntivo.
Il terzo e quarto metodo illustrati sono quelli che soddisfano tutte le esigenze citate e risultano inoltre estremamente flessibili. Elemento comune e` lo schedulatore veloce che deve guidare le operazioni dei singoli blocchi LH9124/LH9320 in modo sincrono rispetto al clock di questi. Tali schedulatori devono comprendere uno spazio di memoria programmabile, devono essere dotati di un elevato numero di terminali per soddisfare le esigenze di lavoro dei diversi blocchi del sistema. Un FPGA riconfigurabile on field puo` soddisfare tutte le richieste. Cio` che e` richiesto a tale sistema possibilmente in single chip e`:
- connessione a tutti i blocchi del sistema multiprocessore (elevato numero di pin);

- Interazioni con l' esterno (interfaccia VME inclusa);
- Frequenza massima di lavoro pari a quella di sistema (40 MHz);
- Operazioni in parallelo (elevata disponibilita`di logica interna);
- memoria di sistema interna(Ram o Rom).

Tutte queste funzioni possono essere racchiuse all'interno di un singolo chip riprogrammabile con il vantaggio di occupare uno spazio limitato sulla scheda multiprocessore. La riconfigurabilita` on fly puo` inoltre permettere l'ottimizzazione della configurazione in funzione dell' algoritmo di lavoro e puo` rappresentare un metodo per massimizzare la memoria su chip disponibile per la schedulazione. In altri termini la sostituzione di una logica generica solo in parte utilizzata per l'esecuzione di un algoritmo, puo` essere sostituita con una logica dedicata esclusivamente alle funzioni da svolgere a tutto vantaggio della sezione logica utilizzabile come memoria e della velocita` di esecuzione.

Come gia` accennato in precedenza, una alternativa alla proposta di controllore appena presentata, potrebbe essere costituita dall' attuale disponibilita` di processori molto veloci tipo quelli riportati nelle fig. 103,104 e 105. Questa scelta potrebbe certamente aggiungere flessibilita`, nel senso che il processore, oltre che fare da controllore, potrebbe eseguire operazioni di pre o post processing. Il costo e lo spazio occupato sulla scheda multiprocessore sarebbe comunque maggiore.

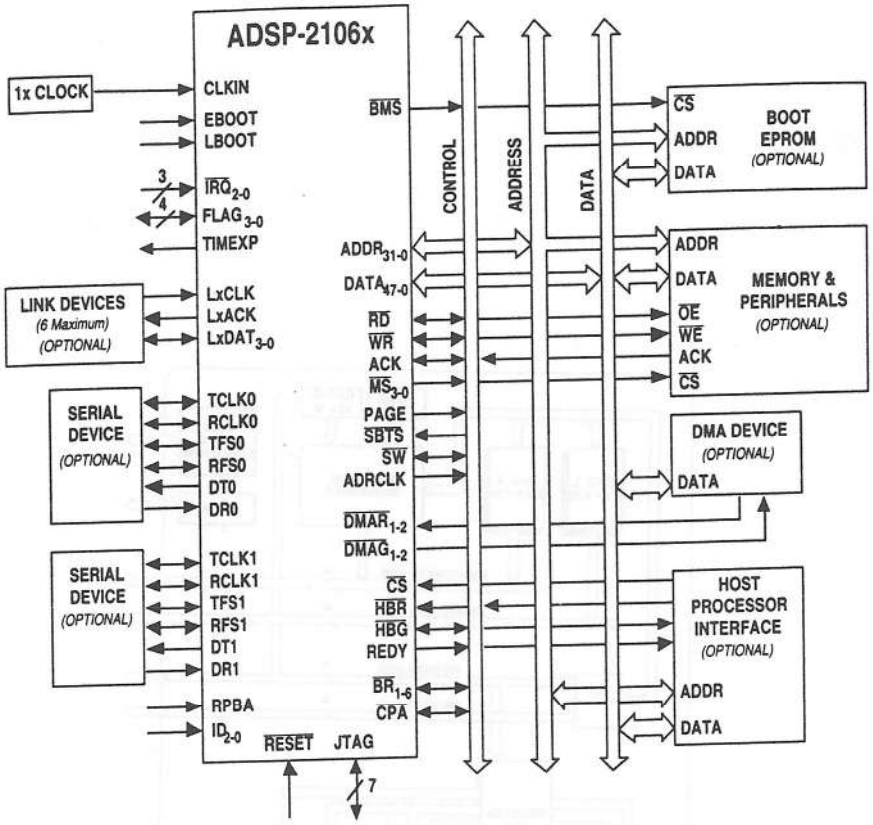


Figure 3 ADSP-2106x System

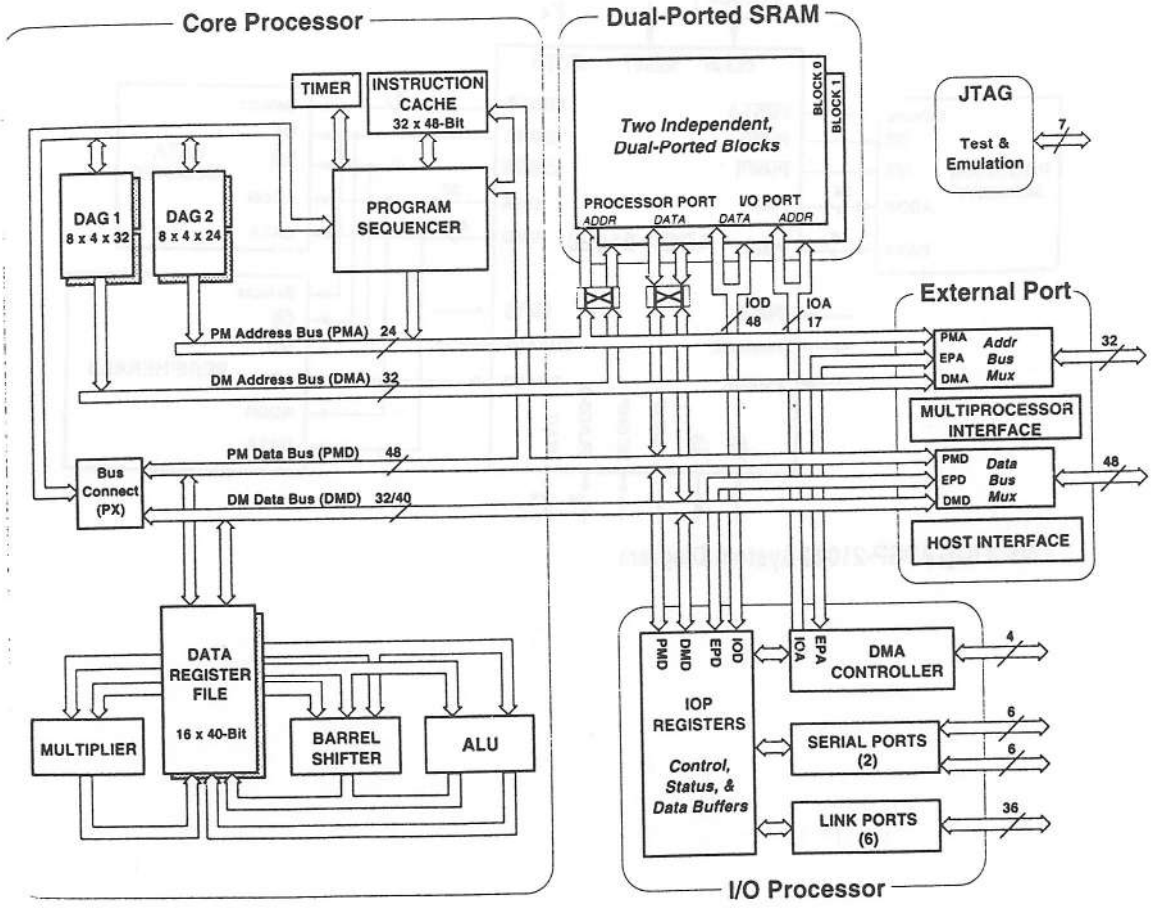
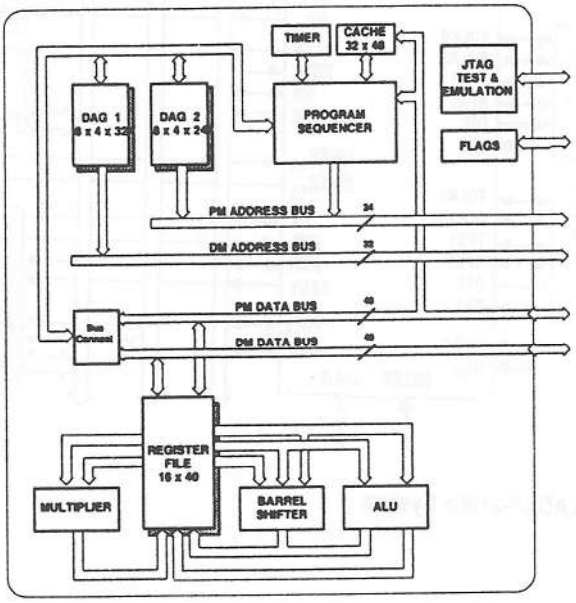


Figure 4 ADSP-2106x SHARC Block Diagram



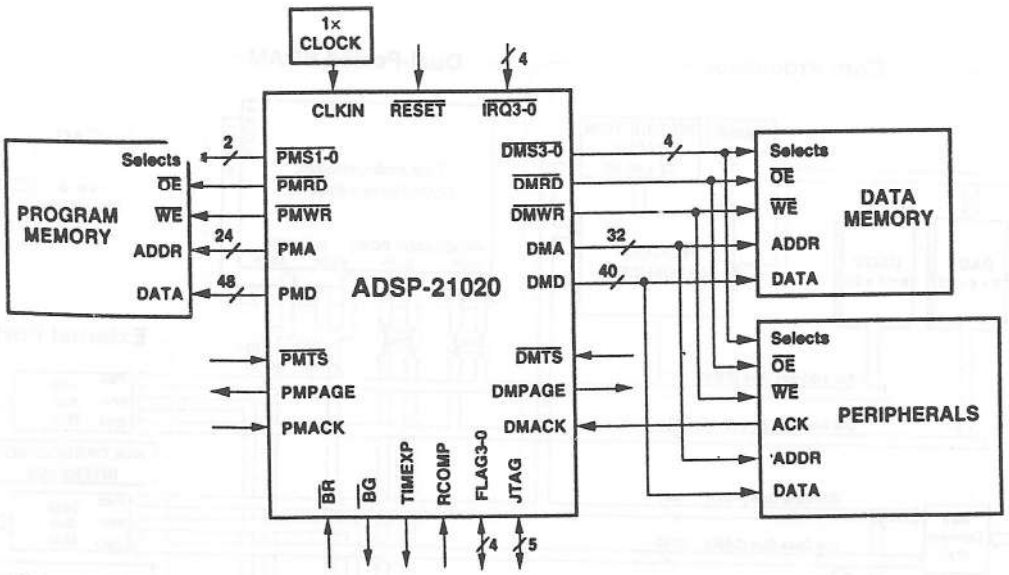


Figure 1.5 ADSP-21020 System Diagram

(file: shar10.doc)

SISTEMA MULTIPROCESSORE.

Fino a questo punto abbiamo cercato di esaminare, anche se in maniera succinta, i segnali di controllo, programmazione, funzioni, modalità di applicazioni, disposizioni circuitali, configurazioni di memoria e controllore del chip set LH9124/LH9320. In realtà, il fine di questa nota tecnica è quello di applicare tale chip set all' elaborazione on-line di segnali radioastronomici provenienti da un convertitore A/D ultra veloce (fino a 500 Ms/Sec.) in frame che possono arrivare ad 1 Milione di punti complessi. Considerando che alla massima rate di conversione i sample distano 2 nS l' uno dall' altro ed in considerazione del fatto che la mole di dati da processare è enorme, per non ridurre il duty-cycle del sistema a valori estremamente bassi si dovranno risolvere grossi problemi di velocità di acquisizione/trasferimento dei dati e, soprattutto, di elaborazione on-line degli stessi. Quello che si può supporre a questo stadio della trattazione è che certamente occorrerà fare uso di architetture multiprocessore, cioè di configurazioni in cui piu' di un DSP prende parte all' esecuzione dell' algoritmo. Facciamo ora una rapida panoramica sulle varie configurazioni a più processori prima di scegliere quella che si riterrà essere la più opportuna per la nostra particolare applicazione. Queste configurazioni possono essere:

- Configurazione con DSP in cascata.
- Configurazione con DSP in parallelo.
- Configurazione a DSP in parallelo con "stesse istruzioni".
- Configurazione multicanale.
- Configurazioni miste cascata/parallelo.

Vediamo ora di analizzare brevemente le configurazioni sopraddette tenendo presente il principio generale consistenete nel fatto che *DSP in cascata aumentano la throughput rate mentre DSP in parallelo migliorano i problemi indotti dalle latenze.*

- Configurazione con DSP in cascata.

Questo tipo di configurazione fornisce un buon aumento delle performance, soprattutto per quello che concerne la riduzione dei ritardi dovuti alle pipeline e latenze di memoria. Come già visto in precedenza, il processo di decomposizione della DFT sottolinea che l' algoritmo della FFT ha una struttura formata da un certo numeri di passi in cascata. Ogni stadio della FFT può essere, quindi, implementato con una funzione Radix di un LH9124 per cui una FFT ad N stadi può essere implementata da un

sistema risultante dalla cascata di N DSP come visibile in fig.106 nel caso in cui N=3. Più in generale, un algoritmo formato da N passi può essere distribuito su N stadi in cui ciascuno esegue una determinata funzione del tipo BFLY16, BCFIR o BWND4 ecc..In questa configurazione non si hanno particolari aumenti di efficienza se si esegue la FFT (o un qualsiasi algoritmo) di un solo array di dati, mentre si ha una diminuzione dei tempi di calcolo di un fattore circa pari al numero N degli stadi, se si eseguono FFT di array consecutivi di uguali dimensioni. Nella stessa figura 106 si nota come sia stata scelta come configurazione della memoria tra uno stadio e l' altro, quella a doppio buffer (memory reallocation) per il fatto che mentre il processore **i** sta processando i dati della serie **n** presi da uno dei due buffer di memoria in out di **i-1,** lo stesso processore **i-1** scarica sul rimanente buffer di out il risultato del processing effettuato sulla sequenza **n+1**. La data rate attraverso tutti gli stadi in cascata, rimane la stessa perchè è lo stesso il tempo richiesto per la esecuzione della stessa funzione radix sullo stesso numero di punti. Il tempo richiesto dal sistema riportato in fig.106, per l' esecuzione di una FFT di 4K punti è di 102.4 uS contro i 312.3 uS per lo stesso computo effettuato da un singolo processore. In fig.107 appare la stessa configurazione della fig.106, nel caso in cui N=6.

- Configurazione con DSP in parallelo.

La configurazione a DSP in parallelo, come riportato in fig.108, fornisce un grosso aumento delle performance e, sotto un certo punto di vista, costituisce una via alternativa a quella precedente appena vista. Il vantaggio offerto dalla architettura parallela, è che ogni LH9124 può eseguire job o algoritmi diversi. Un singolo processore per una FFT (con M stadi) di N punti, impiega N cicli (se il data transfer è pari al clock di sistema) per caricare gli N punti nel processore e circa N*M cicli di clock per portarne a termine il computo. A questo punto in una struttura di DSP paralleli, si possono caricare gli N*M dati in M processori a blocchi di N punti per processore. In questo modo il tempo richiesto dal caricamento dei dati è più bilanciato con quello richiesto dal processing. La throughput rate è, in questo caso, aumentata di un fattore M. Una configurazione di questo genere può anche essere impiegata in casi in cui diversi algoritmi devono essere applicati separatamente ad un blocco e non all' altro. Ritorniamo alla fig.108. I dati sono introdotti nei DSPs attraverso il data bus Dii e Dir con una sola operazione di loading poi ciascuno esegue l' algoritmo assegnato con il supporto del relativo address generator e del controllore. Alla fine del computo, lo schedulatore provvede a scaricare i risultati di ogni DSP sul bus di uscita Dor e Doi evitando di sovrapporre i dati con l' introduzione di opportune latenze. Per sistemi in cui viene richiesta una elevata efficienza e velocità di esecuzione, una configurazione parallela

può essere organizzata per l' esecuzione recursiva di uno o più stadi. Per esempio, riferendosi sempre alla fig.108, il primo LH9124 può essere configurato per il computo recursivo della FFT su un certo N di punti, mentre il secondo DSP esegue un algoritmo di analisi di spettro ed il terzo una IFFT.

- Configurazione a DSP in parallelo con "stesse istruzioni".

Questa architettura è la più adatta per l' handling di grosse moli di dati e prevede il funzionamento degli "n" DSP.in parallelo in modo tale da ricevere contemporaneamente le stesse istruzioni e le stesse sequenze di indirizzamento delle relative memorie come appare in fig.109. Come già detto in precedenza, questa architettura è estremamente adatta nel caso in cui si debbano processare grosse moli di dati con un singolo algoritmo; è semplice da programmare perchè lo stesso FC (Function Code) e DF (Data Flow) viene mandato dallo schedulatore agli "n" DSP come se fossero uno solo, però presenta una maggiore complessità perchè richiede la suddivisione degli array in ingresso in "n" sottoinsiemi uguali da "distribuire", sotto la supervisione dello scheduler, ai vari processori. Dalla stessa fig.109 si nota che, siccome le sequenze di indirizzamento sono le stesse, necessita un solo address generator per ogni porta A,B,C, e Q. La throughput rate in questo caso può essere adattata alla richiesta, ponendo il necessario numero di DSP in parallelo. Questo è in contrasto con quello che accade nella configurazione in cascata dove il numero degli stadi deve essere "rigidamente" collegato al numero dei passi dell' algoritmo da implementare.

- Configurazione multicanale.

La configurazione multicanale è applicabile nei casi in cui il problema da risolvere è rappresentato dalla presenza di molti canali e non dalla elevata velocità di calcolo. Nello schema di fig.110 viene riportato uno schema di principio di un tale sistema multicanale, che si può usare in tutta una serie di applicazioni che vanno dalla compressione vocale per reti, all' imaging in campo medicale, modulazione e demodulazione nel campo delle fibre ottiche ecc....Essendo comunque i suddetti campi di impiego al di fuori del nostro interesse, non si approfondisce ulteriormente l' argomento rimandando gli eventuali interessati ai data sheet dell' LH9124.

- Configurazioni miste cascata/parallelo.

L' architettura mista cascata/parallelo raffigurata in fig.111, è in grado di presentare contemporaneamente i vantaggi della configurazione in cascata ed i vantaggi di quella in parallelo. Si possono, infatti, migliorare le latenze di calcolo parallelizzando più DSP e, nello stesso tempo, aumentare la throughput rate ponendo in cascata più gruppi "parallelo"; questo complica enormemente la progettazione e la costruzione però un sistema, tipo quello di fig.111, è in grado di aumentare le prestazioni globali di un fattore 6. Infatti considerando il computo di una FFT di 4K punti, il blocco A composto da un solo LH9124 impiega:

(4096 x 3 Radix 16 x 25 nS)+(3 Radix x 68 clk latencies x 25nS)= 307.3+5.1=**312.3 uS**

Aggiungendo in parallelo al precedente un secondo DSP (gruppo B), il tempo rochiesto per il calcolo della stessa FFT e`:

(4096 x 3 Radix16 x 25 nS)+(3 Radix16 x 68 CLK latencies x 25nS) / 2= (307.3 uS+5.1 uS) / 2=**159 uS**

Ponendo invece 3 stadi in cascata come il gruppo C, lo stesso tempo di computo diventa:

(4096 x Radix16 x 25 nS per nodo)+(1 Radix16 per nodo x 3 nodi x 68 CLK latencies x 25 nS)=(102.4 uS+4.9 uS)=**107.3** uS

Se, infine, si considera un sistema risultante dalla connessione in cascata di tre gruppi in parallelo come raffigurato nel blocco D, si ottiene:

[(4096 x Radix16 per nodo x 25 nS) / 2 (nodi)]+(Radix16 per stadio x 3stadi x 68 CLK latencies x 25 nS)=(51.2 uS+ 5.1 uS)=**56.3** uS

Come si puo`notare si e`passati dai 312.3 uS del processore singolo ai 56.3 uS con 6 processori.

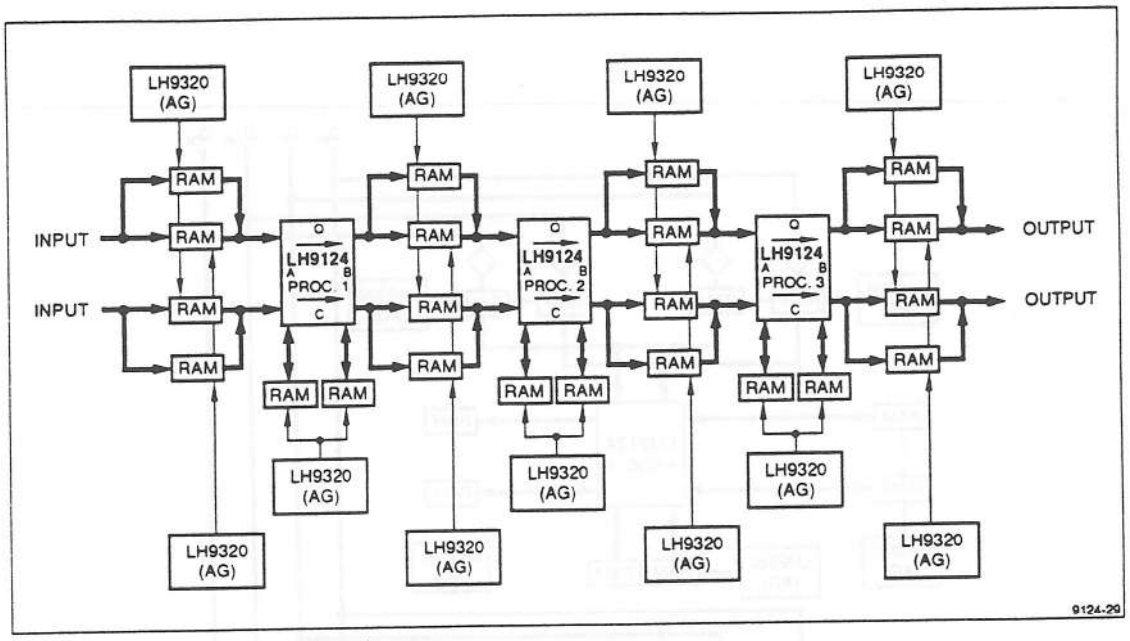


Figure 106 Three Stage Cascaded System Configuration

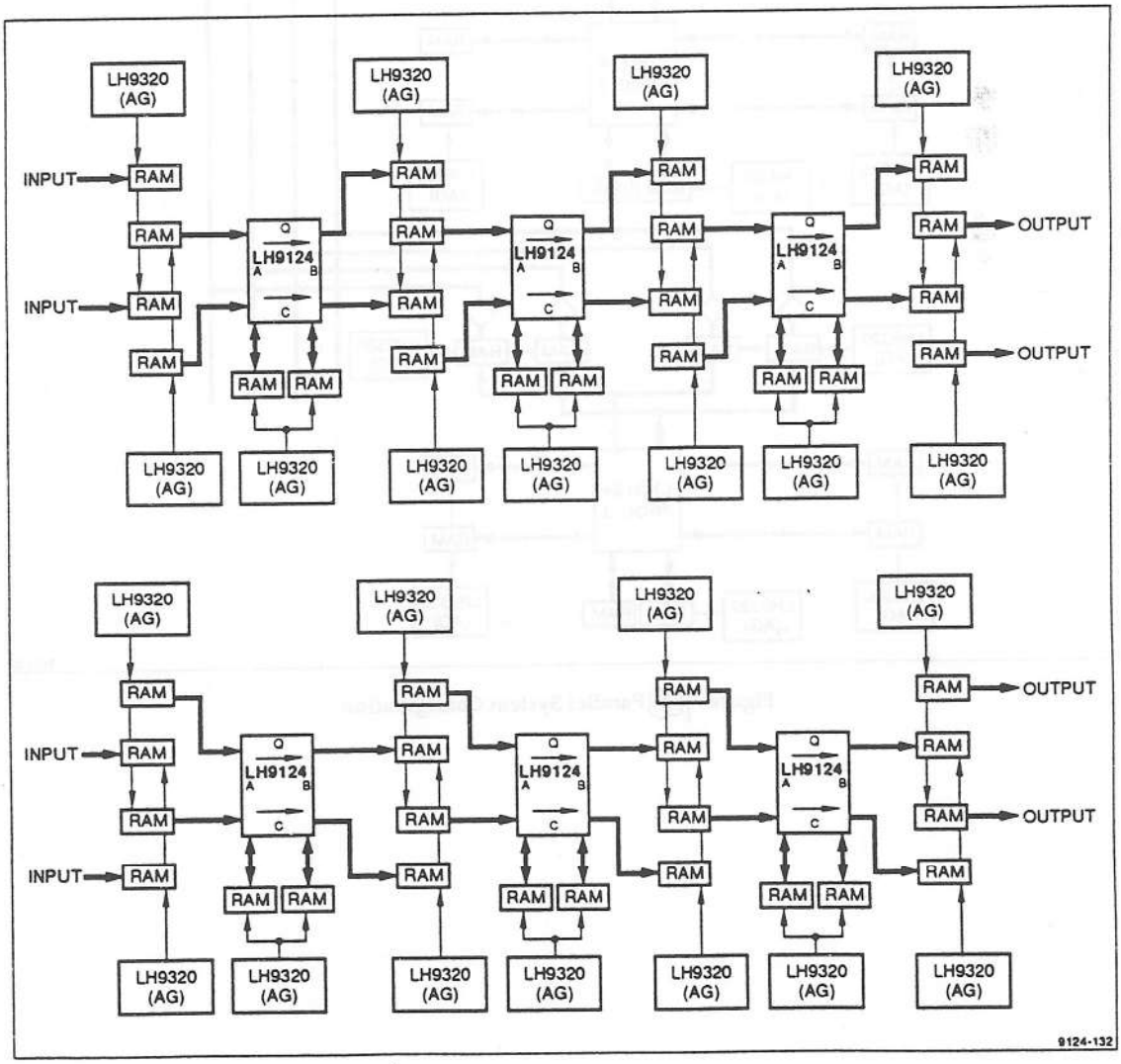


Figure 107 Cascaded System, Memory Reallocation

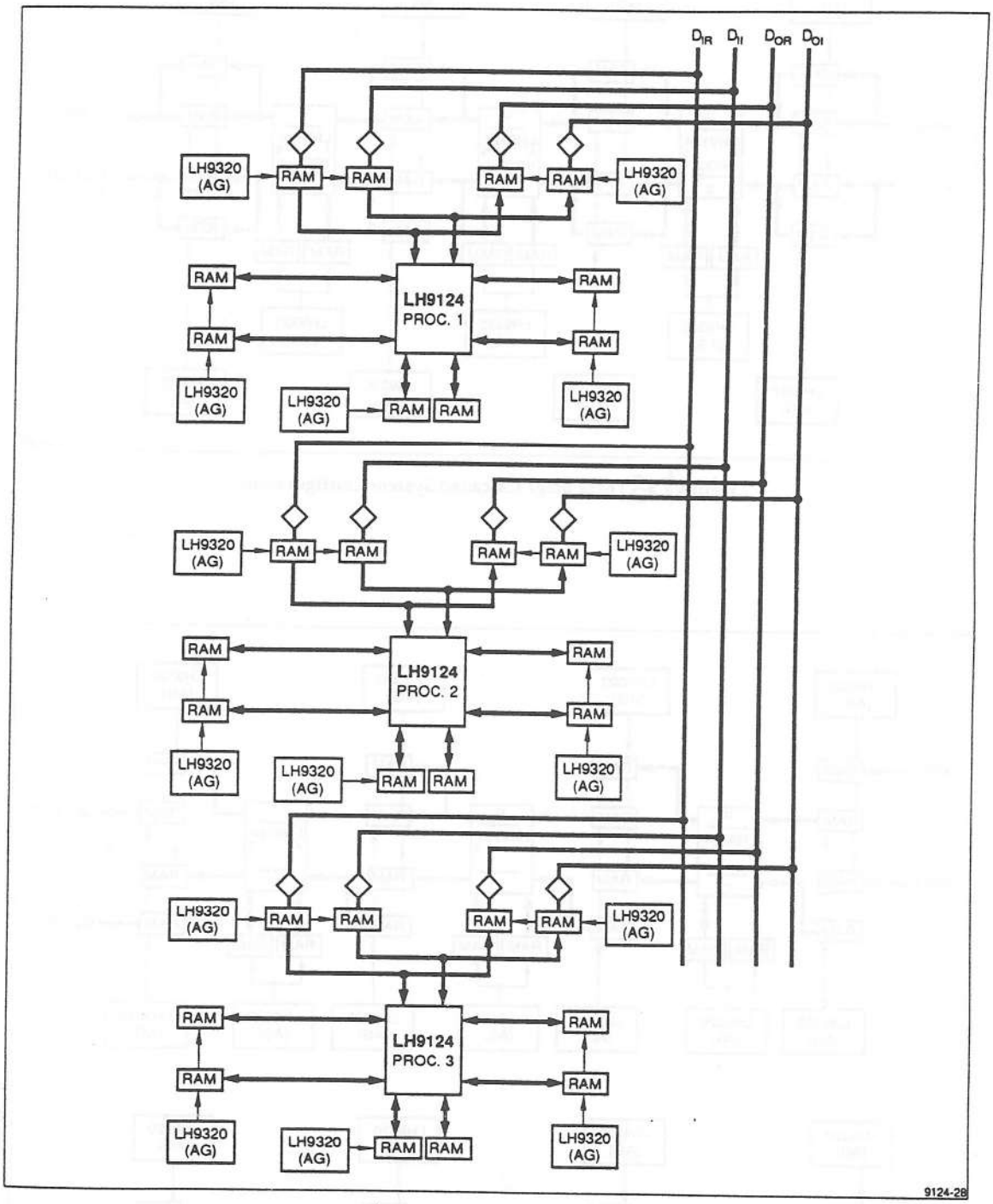


Figure 108 Parallel System Configuration

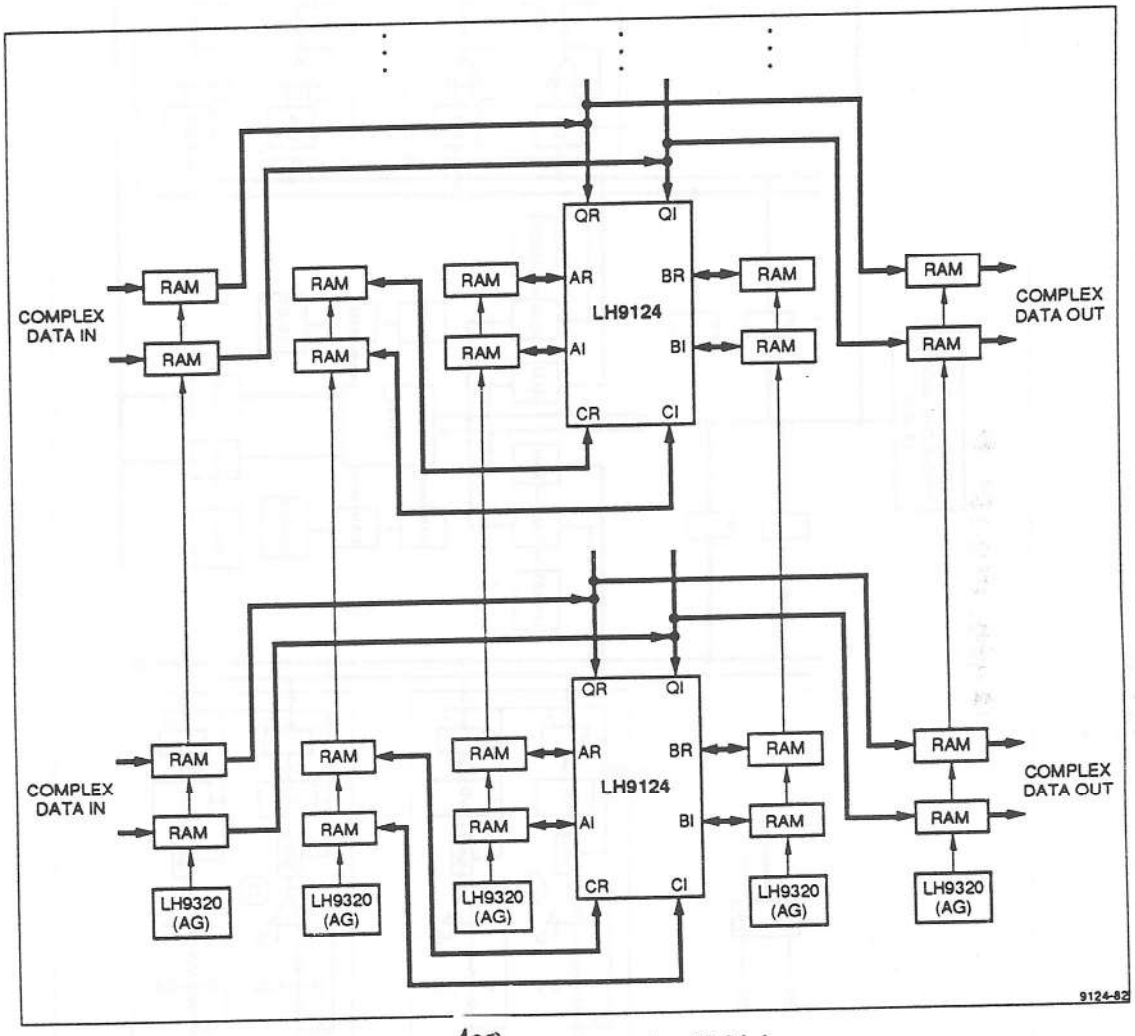


Figure 109 Same Instruction Multiple Data Configuration

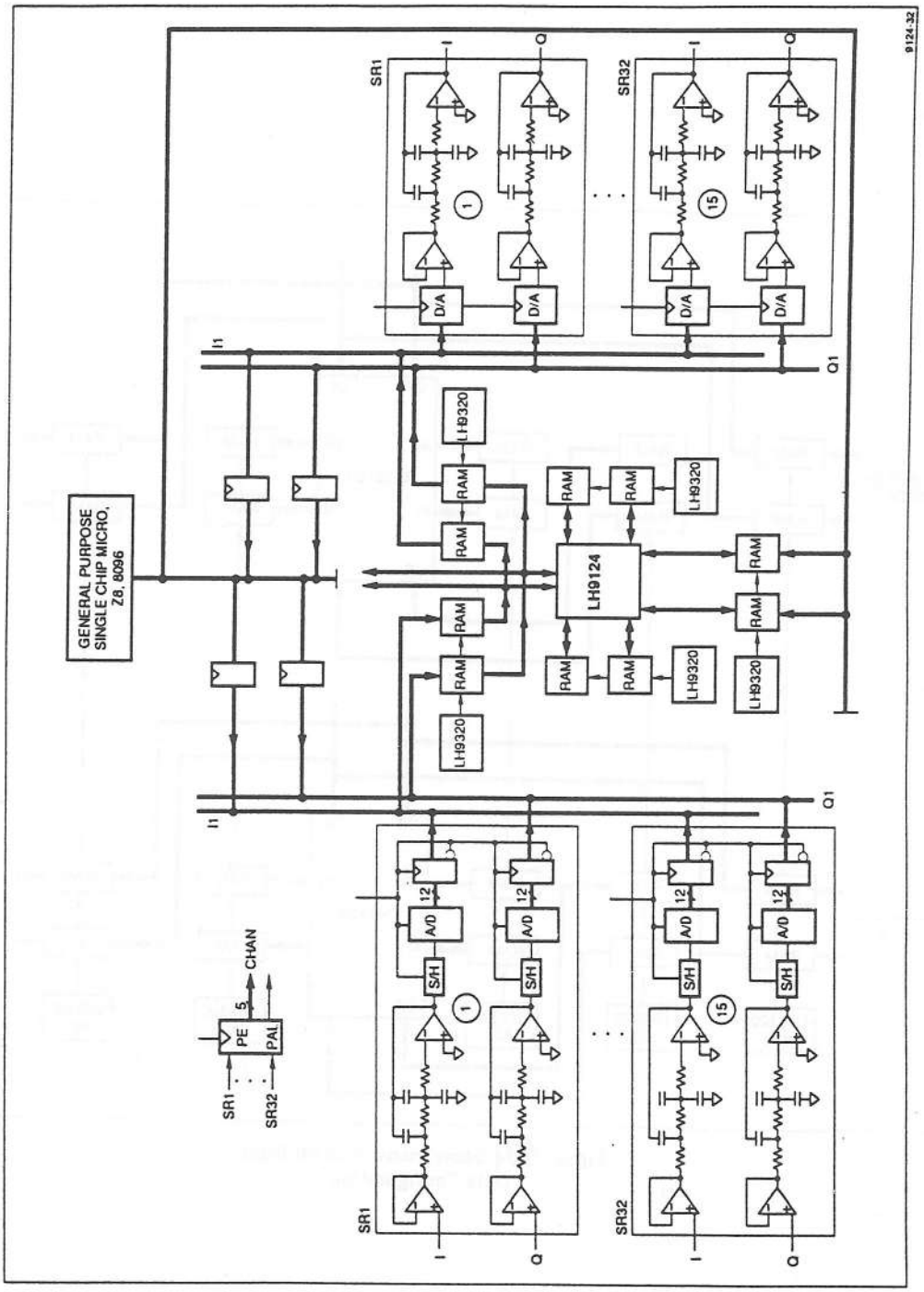


Figure 110 Multiple Channel Processing Configuration

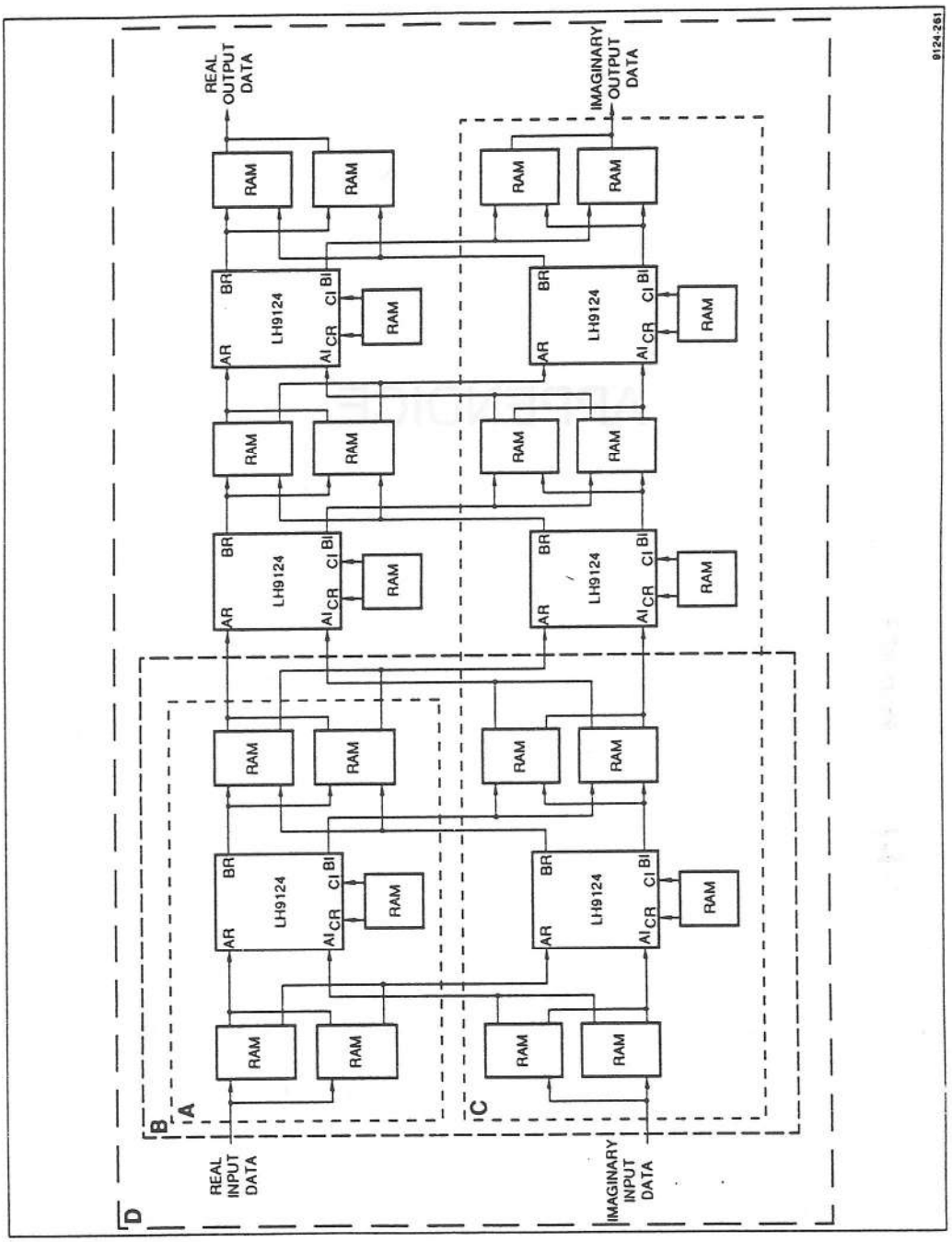


Figure 11 Combined Three Stage Cascaded Configuration With Two Parallel LH9124 Units in Each Stage

APPENDICE

In appendice vengono riportati alcuni argomenti di interesse generale per l'uso del chip set SHARP e del schede VALLEY Tech.

APPENDICE A:

(file: appa.doc)

CONSIDERAZIONI GENERALI SULL' ANALISI SPETTRALE E SCHEDE VALLEY TECH.

Lo spettro di potenza di un segnale deterministico (ossia privo di rumore) si trova calcolando il modulo quadrato della sua trasformata di Fourier, esso rappresenta come è noto la potenza trasportata da ciascuna componente dello spettro. Vedremo in seguito che se il segnale è rumoroso la stima della trasformata di un solo campione x(n) non è sufficiente per determinare il risultato, in generale sarà necessario fare una stima spettrale su un certo numero di sezioni x(n) del segnale e poi mediare il risultato.

Si faranno dapprima alcune considerazioni preliminari a proposito della stima di uno spettro di potenza per segnali rumorosi, dopodiché verrà illustrato uno dei possibili algoritmi per la risoluzione del problema ed infine si esporrà una sua possibile realizzazione hardware mediante l'uso di una scheda ausiliaria da collegare a quella del DSP, in grado di calcolare la stima dello spettro di potenza di un segnale stazionario rumoroso usando lo stesso formato a virgola fissa e la stessa precisione dei valori in uscita dall'operazione di calcolo della trasformata. La descrizione dello stimatore dello spettro di potenza è tratta da Rabiner-Gold:"Theory and application of digital signal processing", mentre la realizzazione hardware è originale.

A.1.Stimatori dell' autocovarianza

Si prenda un processo stazionario casuale x_n, con $-\infty<n<\infty$ e con media nulla.

La funzione di autocovarianza di un tale processo è:

$$\gamma_{xx}=(m)=E[x_n x^*_{n+m}$$

Che, essendo il segnale a media nulla, è uguale a quella di autocorrelazione.

In seguito assumeremo la aspettazione matematica della funzione di autocorrelazione uguale alla media dei valori campionati, in altre parole supporremo che il processo possa considerarsi **ergodico**.

$$\gamma_{xx}=\langle x(n)x^*(n+m)\rangle=\langle g_m(n)\rangle$$

dati n valori consecutivi di x(n), abbiamo (N-m) campioni consecutivi di $g_m(n)$ da cui calcolare la stima.

La stima della sequenza di autocorrelazione è allora:

$$c'_{xx}=\frac{1}{N-|m|}\sum_{n=0}^{N-|m|-1}x(n)x^*(n+m)$$

ove |m|<N.

se la sequenza $g_m(n)$ è gaussiana allora questa espressione costituisce il miglior stimatore per l'autocorrelazione, in generale però l'espressione del miglior stimatore consiste di un sistema di equazioni che non può essere risolto nemmeno se si conosce l' espressione di $g_m(n)$. Tuttavia, anche se non costituisce una stima ottima della funzione di autocorrelazione la si può generalmente utilizzare con una certa tranquillità in quanto costituisce lo stimatore ottimo per :

$$\phi_{xx}(m)=E[x(n)x^*(n+m)$$

Uno stimatore alternativo per la sequenza di autocorrelazione è costituito da:

$$c_{xx}(m)=\frac{1}{N}\sum_{n=0}^{N-|m|-1}x(n)x(n+m)$$

che differisce solo per il fattore moltiplicativo prima della somma, in effetti risulta che :

$$c_{xx}(m)=\frac{N-|m|}{N}c'_{xx}(m)$$

mentre la relazione fra le aspettazioni matematiche risulta essere:

$$E[c_{xx}(m)] = \frac{N-|m|}{N}\phi_{xx}(m)$$

A.1.1.Il periodogramma come stima dello spettro di potenza

Si è dunque ottenuto uno stimatore per la sequenza di autocovarianza, si è ora tentati di concludere che la trasformata di Fourier di tale stimatore è a sua volta un stimatore dello spettro di potenza della sequenza x(n), sfortunatamente non è così in quanto la varianza della trasformata non tende a zero al crescere di N e quindi lo stimatore non può essere definito come coerente.

Si mostrerà però come, con l'uso di particolari finestre nella trasformata di Fourier, si riesca ad utilizzare la stima della covarianza per calcolare lo spettro di potenza.

La trasformata dello stimatore dell'autocorrelazione è, evidentemente:

$$I_N(\omega) = \sum_{m=-(N-1)}^{N-1} c_{xx}(m)e^{-j\omega m}.$$

Ricordando che la ben nota espressione della trasformata discreta della sequenza x(n) è:

$$X(e^{j\omega}) = \sum_{n=0}^{N-1} x(n)e^{-j\omega n}$$

si può mostrare che l'espressione $I_N(\omega)$ risulta essere:

$$I_N(\omega) = \frac{1}{N}\left|X(e^{j\omega})\right|^2$$

Tale espressione si chiama ***periodogramma.***

Interessa calcolare la polarizzazione e la varianza del periodogramma per determinare se si tratta di un buon stimatore dello spettro di potenza.

Il valore atteso è:

$$E[I_N(\omega)] = \sum_{m=-(N-1)}^{N-1} E[c_{xx}(m)]e^{-j\omega m}$$

Abbiamo visto come per un processo a media nulla :

$$E[c_{xx}(m)] = \frac{N-|m|}{N}\phi_{xx}(m), |m| < N$$

allora:

$$E[I_N(\omega)] = \sum_{m=-|N-1|}^{N-1} \left(\frac{N-|m|}{N}\right)\phi_{xx}(m)e^{-j\omega m}$$

A causa del numero finito di addendi della sommatoria e per il fattore

(N-|m|)/N che vi compare, tale aspettazione non è uguale alla trasformata di Fourier di $\phi_{XX}(m)$ e quindi il periodiogramma non è uno stimatore coerente dello spettro di potenza.

Possiamo però interpretare l'ultima equazione come la trasformata di Fourier della sequenza di autocorrelazione finestrata da:

$$w_B(m)=\begin{cases}\dfrac{N-|m|}{N}, |m|<N\\ 0, altrove\end{cases}$$

tale finestra è detta di Bartlett o triangolare.

Può altresì essere interpretata come la convoluzione nel dominio delle frequenze:

$$E[I_N(\omega)]=\frac{1}{2\pi}\int_{-\pi}^{\pi}P_{xx}(\theta)W_B(e^{j(\omega-\theta)})d\theta$$

ove :

$$_B(e^{j\omega})=\frac{1}{N}\left(\frac{sin\left[\frac{\omega N}{2}\right]}{sin\left[\frac{\omega}{2}\right]}\right)^2$$

che costituisce naturalmente la trasformata di Fourier della finestra di Bartlett.

varianza del periodogramma

Si supponga che la sequenza sia relativa ad un segnale reale, a valor medio nullo e di spettro bianco con funzione di densità di probabilità gaussiana.

Il periodogramma si può esprimere come:

$$I_N(\omega)=\frac{1}{N}|X(e^{j\omega})|^2=\frac{1}{N}\sum_{l=0}^{N-1}\sum_{m=0}^{N-1}x(l)x(m)e^{j\omega(m-l)}$$

per valutare la covarianza di $I_N(\omega)$si considera:

$$E[I_N(\omega_1)I_N(\omega_2)]=\frac{1}{N^2}\sum_{K=0}^{N-1}\sum_{l=0}^{N-1}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}E[x(k)x(l)x(m)x(n)]e^{-j[\omega_1(k-l)+\omega_2(m-n)}$$

In caso di processo gaussiano si può mostrare che :

$$E[x(k)x(l)x(m)x(n)]=E[x(k)x(l)]E[x(m)x(n)]+E[x(k)x(m)]E[x(l)x(n)]+E[x(k)x(n)]E[x(l)x(m)$$

quindi:

$$E[x(k)x(l)x(m)x(n)]=\begin{cases}s_x^2,(k=1 \text{ and } m=n) \text{ or } (k=m \text{ and } l=n) \text{ or } (k=n \text{ and } l=m)\\ 0,\text{altrove}\end{cases}$$

Sostituendo otteniamo:

$$E[I_N(\omega_1)I_N(\omega_2)]=\frac{o_x^2}{N^2}\left\{N^2+\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}e^{j(m-n)(\omega_1+\omega_2)}+\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}e^{j(n-m)(\omega_1-\omega_2)}\right\}$$

$$\frac{2}{x}\left\{1+\left(\frac{sin[(\omega_1+\omega_2)N/2]}{Nsin[(\omega_1+\omega_2)/2]}\right)^2+\left(\frac{sin[(\omega_1-\omega_2)N/2]}{Nsin[(\omega_1-\omega_2)/2]}\right)^2\right\}$$

la covarianza del periodogramma è :

$cov[I_N(\omega_1)I_N(\omega_2)]=E[I_N(\omega_1)I_N(\omega_2)]-E[I_N(\omega_1)]E[I_N(\omega_2)$

poiché $E[I_N(\omega_1)]=E[I_N(\omega_2)]=o_x^2$ otteniamo:

$$\text{cov}[I_N(\omega_1),I_N(\omega_2)]=o_x^2\left\{\left(\frac{sin[(\omega_1+\omega_2)N/2]}{Nsin[(\omega_1+\omega_2)/2]}\right)^2+\left(\frac{sin[(\omega_1-\omega_2)N/2]}{Nsin[(\omega_1-\omega_2)/2]}\right)^2\right\}$$

possiamo a questo punto fare alcune interessanti considerazioni a proposito dei periodogrammi, innanzitutto la varianza dello stimatore dello spettro ad una determinata frequenza è:

$$\text{var}[I_N(\omega)]=\text{cov}[I_N(\omega),I_N(\omega)]=o_x^2\left\{1+\left(\frac{sin[\omega N]}{Nsin\omega}\right)^2\right\}$$

che non tende a zero quando N tende ad infinito, costituendo quindi una conferma del fatto che il periodogramma non è una stima consistente dello spettro di potenza.

Si vede anche che, per frequenze $\omega_1=2\pi k/N$ e $\omega_2=2\pi l/N$ ove k e l sono interi:

$$\text{cov}[I_N(\omega_1),I_N(\omega_2)]=o_x^4\left\{\left(\frac{sin[\pi(k+l)]}{Nsin[\pi(k+l)/N]}\right)^2+\left(\frac{sin[\pi(k-l)]}{Nsin[\pi(k-l)/N]}\right)^2\right\}$$

è uguale a 0 per $k\neq l$, quindi i valori del periodogramma spaziati di frequenze multiple di $2\pi/N$ sono fra loro incorrelati.

All' aumentare di N tali frequenze diventano via via più ravvicinate, una conseguenza di questo fatto è che le fluttuazioni del periodogramma aumentano.

A.2.stimatori dello spettro finestrati

A.2.1.media di Bartlett dei periodogrammi

Una procedura standard per ridurre la varianza degli stimatori dei periodogrammi è di fare la media di un certo numero di stime indipendenti
Il data record di N campioni è suddiviso in K segmenti di lunghezza M, ove K=N/M, ci sono K periodogrammi:

$$I_M^{(i)}(\omega) = \frac{1}{MU}\left|\sum_{n=0}^{M-1} x^{(i)}(n) w(n) e^{j\omega n}\right|^2$$

ove i=0,1,...,K
l'indice i indica il numero di segmento.

$$= \frac{1}{M}\sum_{n=0}^{M-1} w^2(n)$$

costituisce il fattore di normalizzazione per la finestra di Bartlett.
Infine la stima della densità spettrale di potenza è costituita dalla media:

$$\beta_{xx}(\omega) = \frac{1}{K}\sum_{i=1}^{K} I_M^{(i)}(\omega)$$

L'aspettazione matematica di tale stimatore è:

$$E\left[\beta_{xx}(\omega)\right] = \frac{1}{K}\sum_{i=1}^{K} E\left[I_M^{(i)}(\omega)\right] = E\left[I_M^{(i)}(\omega)\right] = \frac{1}{2\pi M}\int_{-\pi}^{\pi} P_{xx}(\theta)\left(\frac{sin\left[(\omega-\theta)M/2\right]}{sin\left[(\omega-\theta)/2\right]}\right)^2 d\theta$$

Quindi l'aspettazione dello stimatore è la convoluzione dello spettro vero P_{XX} con la trasformata di Fourier della finestra triangolare.
Lo stimatore di Bartlett è evidentemente polarizzato ma se assumiamo che i K periodogrammi siano statisticamente indipendenti allora la varianza di β_{XX} è:

$$\mathrm{var}\left[\beta_{xx}(\omega)\right] = \frac{1}{K}\mathrm{var}\left[I_M(\omega)\right] \approx \frac{1}{K}P_{xx}^2(\omega)\left\{1+\left(\frac{sin[\omega M]}{Msin\omega}\right)^2\right\}$$

Quindi la varianza è inversamente proporzionale al numero di periodogrammi di cui si calcola la media e all' aumentare di K tende a zero, conseguentemente lo stimatore di Bartlett è uno stimatore consistente.
La polarizzazione di β_{XX} è superiore a quella di I_N come si ricava dall' esame delle espressioni dei due stimatori, che possono essere visti come la convoluzione di una funzione per la trasformata di una certa finestra di cui quella di β_{XX} ha un lobo principale più esteso.
La polarizzazione può essere interpretata in termini di effetti sulla risoluzione dello spettro: quando aumenta il numero di periodogrammi, a parità di lunghezza del record di ciascun periodogramma la varianza

diminuisce ma diminuisce anche M e quindi diminuisce la risoluzione dello spettro. Di conseguenza la scelta di M e N deve essere frutto di un compromesso che in generale dipende dal tipo di segnale.

A.2.2.windowing

Abbiamo visto come la varianza dello stimatore di Bartlett è ridotta a spese dell' aumento della polarizzazione e della riduzione della risoluzione.
Un altro possibile approccio consiste nello schermare il periodogramma per convoluzione con una appropriata finestra S_{XX}:

$$S_{xx}(\omega)=\frac{1}{2\pi}\int_{-\pi}^{\pi} I_N(\theta)W(e^{j(\omega-\theta)}d\theta$$

ove $W(e^{j\omega})$ è lo spettro della finestra.
Poiché il periodogramma è la trasformata di Fourier di c_{XX}, allora S_{XX} è la trasformata del prodotto fra c_{XX} e la trasformata inversa di $W(e^{j\omega})$. Cioè:

$$S_{xx}(\omega)=\sum_{m=-(M-1)}^{M-1} c_{xx}(m)w(m)e^{-j\omega m}$$

Poiché lo spettro di frequenza è non-negativo si impone che S_{XX} sia a sua volta non-negativa, perché ciò accada la trasformata della finestra deve essere non-negativa, ciò è vero per le finestre triangolari, ma non, ad esempio, per quelle di Hamming o Hanning che pure portano ad una migliore stima dello spettro di potenza (se esso risulta positivo).
L'aspettazione dello stimatore con finestra risulta:

$$E\left[S_{xx}(\omega)\right]=\frac{1}{2\pi}\int_{-\pi}^{\pi} E\left[I_N(\theta)\right]W\left(e^{j(\omega-\theta)}\right)d\theta$$

In cui:

$$E\left[I_N(\theta)\right]=\frac{1}{2\pi}\int_{-\pi}^{\pi} P_{xx}(\phi)W_B\left(e^{j(\theta-\phi)}\right)d\phi$$

Quindi $E[S_{XX}]$ è la trasformata di Fourier di $\phi_{XX}(m)$ volte il prodotto delle finestre triangolari $w_B(m)$ e $w(m)$ cioè:

$$E\left[S_{xx}(\omega)\right]=\sum_{m=-(M-1)}^{M-1}\phi_{xx}(m)w_B(m)w(m)e^{-j\omega m}$$

ove

$$w_B(m)=1-\frac{|m|}{N},|m|<N$$

Se M è piccolo rispetto a N allora W($e^{j\omega}$) sarà più grande di $_B(e^{j\omega})$, quindi l'ultima equazione si può approssimare come:

$$E[S_{xx}(\omega)] \approx \frac{1}{2\pi}\int_{-\pi}^{\pi} P_{xx}(\theta)W(e^{j(\omega-\theta)})d\theta$$

Da questa si evidenzia che un aumento dello spettro della finestra è causa di una schermatura addizionale dello spettro del segnale e riduce la risoluzione in frequenza.

A.2.3.metodo Bartlett-Welch della media dei periodogrammi modificati

Si tratta della modificazione del metodo di Bartlett appena visto, che lo rende particolarmente adatto ad essere applicato alla stima dello spettro di potenza mediante l'uso di FFT (le idee guida sono tratte dagli application note di LH9124).

Come nel metodo originale si divide il record originale in K=N/M segmenti di M campioni ciascuno, in questo caso però la finestra w(n) è applicata direttamente al segmento dati, prima del calcolo del periodogramma, si definiscono in questo modo K *periodogrammi modificati*:

$$J_M^{(i)}(\omega) = \frac{1}{MU}\left|\sum_{n=0}^{M-1} x^{(i)}(n)w(n)e^{-j\omega n}\right|^2, i = 1,2,\ldots,K$$

ove

$$= \frac{1}{M}\sum_{n=0}^{M-1} w^2(n)$$

e lo stimatore dello spettro è:

$$B_{xx}^{w}(\omega) = \frac{1}{K}\sum_{i=1}^{K} J_M^{(i)}(\omega)$$

la cui aspettazione matematica è:

$$E[B_{xx}^{w}(\omega)] = \frac{1}{2\pi}\int_{-\pi}^{\pi} P_{xx}(\theta)W(e^{j(\omega-\theta)})d\theta$$

ove

$$(e^{j\omega}) = \frac{1}{MU}\left|\sum_{n=0}^{M-1} w(n)e^{-j\omega n}\right|^2$$

Il fattore di normalizzazione U è richiesto perché lo stimatore sia asintoticamente senza polarizzazione.

Se i segmenti sono non sovrapposti la varianza dello stimatore appena descritto è:

$$\mathrm{var}\left[B_{xx}^2(\omega)\right] \approx \frac{1}{K} P_{xx}^2(\omega)$$

come succede del resto per la procedura di Bartlett senza modificazioni.

Nel caso invece in cui si abbia sovrapposizione fra i segmenti adiacenti essi risultano non indipendenti. Allora finestrando i segmenti di dati prima di calcolare il periodogramma noi compiamo una riduzione della varianza rispetto allo stimatore di Bartlett e allo stesso modo realizziamo la mascheratura dello spettro con la relativa diminuzione della risoluzione.

In questo caso lo spettro della finestra è proporzionale alla magnitudine quadrata della trasformata di Fourier della finestra, anziché alla trasformata stessa.

Ciò significa che indipendentemente dalla finestra usata, il suo spettro sarà sempre non-negativo e lo stimatore dello spettro sarà dunque pure esso non-negativo.

A.3. applicazioni della FFT alla stima spettrale

La FFT realizza un modo efficiente per il calcolo dello spettro di potenza per frequenze egualmente spaziate $\omega_k=2\pi k/M$.

per realizzare il metodo di Bartlett-Welch si deve calcolare il valore dello stimatore:

$$\beta_{xx}^{w}\left(\frac{2\pi}{M}k\right) = \frac{1}{K}\sum_{i=1}^{K} J_M^{(i)}\left(\frac{2\pi}{M}k\right) \quad \mathrm{k} = 0,1,\ldots,\mathrm{M}-1$$

ove

$$J_M^{(i)}\left(\frac{2\pi}{M}k\right) = \frac{1}{MU}\left|\sum_{n=0}^{M-1} x^{(i)}(n)w(n)e^{-j\left(\frac{2\pi}{M}\right)kn}\right|^2 \quad \mathrm{i} = 1,2,\ldots\mathrm{K}\ ;\ \mathrm{k} = 0,1,\ldots,\mathrm{M}-$$

il che può essere fatto calcolando:

$$X_M^{(i)}(k) = \sum_{n=0}^{M-1} x^{(i)}(n)w(n)e^{-j\left(\frac{2\pi}{M}\right)kn}, k =,1,\ldots M-1.$$

per ogni sezione, applicando uno qualsiasi degli algoritmi di FFT, dopodiché si calcola il modulo quadrato $|X_M^{(i)}(k)|^2$ per ciascuna sezione ed infine sommando i K risultati e dividendo il tutto per KMU.
Per dati reali si può applicare l'algoritmo per il calcolo di due trasformate reali mediante una sola complessa, per ridurre i tempi di calcolo.

A.3.1.calcolo degli stimatori di correlazione

Si possono usare gli algoritmi FFT per il calcolo dello stimatore di autocorrelazione:

$$c_{xx}(m) = \frac{1}{N} \sum_{n=0}^{N-|m|-1} x(n)x(n+m) \qquad 0 \le m \le M -$$

ove M≤N.
Si osserva che la correlazione corrisponde al calcolo della convoluzione discreta fra x(n) e x(-n).
In pratica si calcola la trasformata X(k) e la si moltiplica per la complessa coniugata e quindi si calcola la trasformata inversa, questo corrisponde al calcolo della correlazione fra x(n) e x(n), cioè, come si è detto, al calcolo della funzione di autocorrelazione.
Aggiungendo alla sequenza x(n): (L-N) valori nulli e calcolando una DFT da L punti si può forzare la correlazione circolare ad essere corretta nell' intervallo 0≤m≤M-1.
Per scegliere la lunghezza L si considerino i seguenti grafici relativi alle sequenze x(n) e x(n+m) con m positivo.

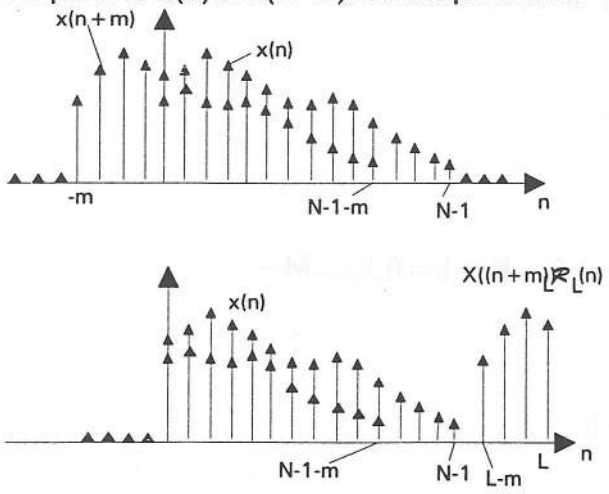


Figura A.1 calcolo dello stimatore dell'autocorrelazione

Il primo mostra le due sequenze x(n) e x(m+n) per un dato valore di m, la seconda le sequenze x(n) e $x(n+m)_L$ coinvolte nel calcolo della correlazione circolare.

Tale correlazione sarà uguale a $Nc_{xx}(m)$ per $0 \le m \le M-1$ se $x(n+m)_L$ non si sovrappone a x(n) per $0 \le m \le M-1$ (che è il caso in cui $L \ge N+M-1$).

Per calcolare lo stimatore $c_{xx}(m)$ della correlazione:

1. Si forma una sequenza di L punti aggiungendo M-1 zeri in coda a x(n)
2. Si calcola la DFT da L punti: $X(k) = \sum_{n=0}^{L-1} x(n) e^{-j\left(\frac{2\pi}{L}\right)kn}$

A.4. Uso di LH9124 per lo studio della densità spettrale di potenza

Semplificando la scrittura delle espressioni ottenute per adattarle al loro uso con LH9124, la stima della media diventa:

$$\beta_{xx}\left(\frac{2\pi k}{M}\right) = \frac{1}{K}\sum_{i=1}^{K} I_M^{(i)}\left(\frac{2\pi k}{M}\right) \qquad k = 0,1,\ldots,M-1$$

quindi:

$$I_M^{(i)}\left(\frac{2\pi}{M}k\right) = \frac{1}{MU}\left|\sum_{n=0}^{M-1} x^{(i)}(n) w(n) e^{-j\left(\frac{2\pi}{M}\right)kn}\right|^2$$

poiché poi:

$$X_M^{(i)} = \sum_{n=0}^{M-1} x_n^{(i)} w(n) e^{-j\left(\frac{2\pi kn}{M}\right)}$$

si vede chiaramente che è possibile calcolare la β_{xx} calcolando la K trasformate di Fourier su M punti, quindi il loro modulo mediante la CMAG ed infine sommandole insieme e dividendo per KMU il risultato.

A.4.1. calcolo della stima di correlazione

Abbiamo già visto come la stima della auto-correlazione di un segnale si indichi con:

$$c_{xx}(m) = \frac{1}{N} \sum_{n=0}^{N-|m|-1} x(n)x(n+m), 0 \leq |m| \leq M-1, M < N$$

La convoluzione di x(n) ed x(-n) è:

$$x(m) * x(-m) = \sum_{n=-\infty}^{\infty} x(n)x(n+m) = \sum_{n=0}^{N-|M|-1} x(n)x(n+m)$$

supponiamo di chiamare X(k)=FFT[x(m)], allora X*(k)=FFT[X(-m)] e X(k)·X*(k)=$|X(k)|^2$=FFT[X(m)*X(-m)]

La trasformata inversa di $|X(k)|^2$ è la convoluzione circolare di x(n) e x(-n). Quindi aumentando la sequenza x(n) con (L-N) zeri e calcolando una FFT da L punti si trova il valore corretto della convoluzione circolare per 0≤m≤ M-1.

La sequenza di operazioni da eseguire è allora:

1. aggiungere M-1 zeri in coda a x(n)
2. calcolare: $X(k) = \sum_{n=0}^{L-1} x(n) e^{j\left(\frac{2\pi}{L}\right)kn}$
3. calcolare il modulo quadro di X(k): $|X(k)|^2$.
4. calcolare la trasformata inversa di $|X(k)|^2$: $c(m) = \frac{1}{L} \sum_{k=0}^{L-1} |X(k)|^2 e^{j\left(\frac{2\pi}{L}\right)km}, m = 0,1,\ldots L-1$
5. dividere il risultato per N per ottenere la stima dell' autocorrelazione

A.5 Proposte di soluzione del problema dell' accumulazione dei risultati per il calcolo dei periodogrammi modificati e stima dell' errore introdotto dalle diverse modalità .

LH9124 è in grado di calcolare i moduli quadri di vettori complessi, necessari per la determinazione dello spettro di potenza del segnale in ingresso, ciò viene fatto mediante l'istruzione CMAG che appunto esegue l' operazione R^2+I^2, ove con R e I si sono naturalmente indicate la parte reale ed immaginaria del vettore complesso in uscita dal calcolo della FFT.

L'istruzione ha un pattern di uscita dei dati diverso da quello comune a tutte le altre istruzioni di LH9124, infatti fornisce dati reali a 48 bit di precisione, portati però all' esterno in due cicli successivi di clock, in particolare: al primo colpo di clock sono resi disponibili sui piedini corrispondenti ai 24 bit reali i 24 bit più significativi del modulo di X(0) e sui piedini dei 24 bit immaginari, quelli più significativi del modulo di X(1), al secondo colpo di clock sono invece presenti, sui piedini dei dati reali i 24 bit meno significativi di X(0) e su quelli immaginari i meno significativi di X(1). Dopodiché naturalmente il ciclo ricomincia e vengono rese disponibili le word più significative di X(2) e di X(3) e così via.
Lo schema può essere riassunto dalla tabella:

piedini reali	piedini immaginari
$i_{2i}^{(47-24)}$	$i_{2i+1}^{(47-24)}$
$i_{2i}^{(23-0)}$	$i_{2i+1}^{(23-0)}$
$i_{2i+2}^{(47-24)}$	$i_{2i+3}^{(47-24)}$
$i_{2i+2}^{(23-0)}$	$i_{2i+3}^{(23-0)}$

Tabella 1. schema di uscita dei risultati da una operazione CMAG

Si può, in base a queste premesse, pensare a due soluzioni distinte per il calcolo della somma delle trasformate dei vari spettri, necessarie per applicare il metodo di Bartlett-Welch, ossia dei periodogrammi modificati.
La prima prevede l'uso dei registri interni dello stesso LH9124 per eseguire la somma degli spettri, la seconda nell' usare invece un sommatore esterno per numeri in virgola fissa o un altro qualsivoglia processore (ad esempio un '860, quest'ultima soluzione ha il vantaggio di una maggiore flessibilità ma complica notevolmente il progetto rispetto all' uso di un semplice sommatore a virgola fissa). In entrambi i casi i dati uscenti dal DSP devono essere sommati al contenuto di una apposita memoria di accumulazione dei risultati, di dimensioni sufficienti a contenere i risultati delle somme.

A.5.1.uso dei sommatori interni di LH9124

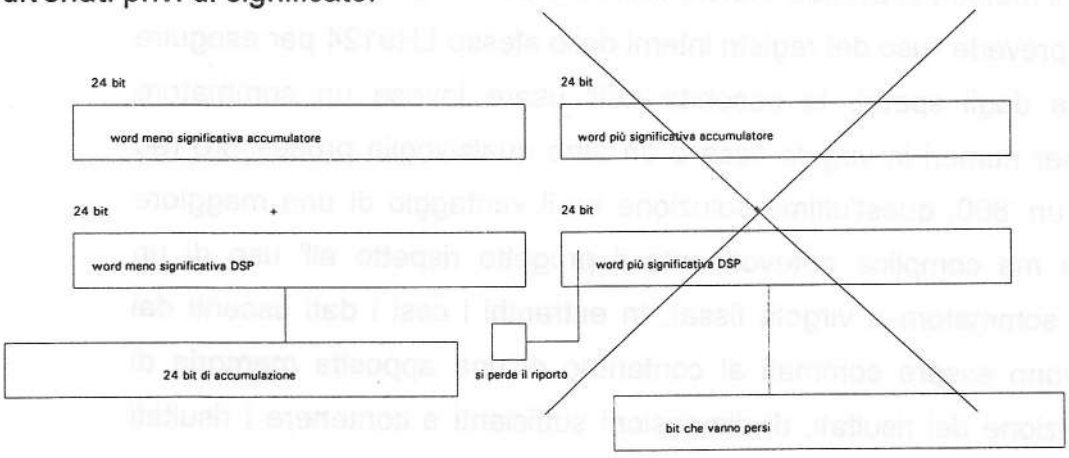
L'usare le istruzioni di somma dello Sharp LH9124 (ossia la VADD o la CADD, che sono equivalenti) ha evidentemente il pregio della semplicità e dell'economicità di implementazione (non è necessario usare dei componenti esterni) ma, come vedremo, ha il grosso difetto di introdurre errori di troncamento dei risultati spesso intollerabili se non si ha a che fare con dei segnali con uno spettro praticamente bianco.

L'origine di questo problema risiede nel fatto che l' istruzione CADD esegue somme di numeri a soli 24 bit, contro i 48 con cui sono noti i risultati del calcolo del modulo con CMAG ([1]), ne risulta la necessità di troncare i 24 bit meno significativi.

La ragione per cui l'operazione di calcolo del modulo (CMAG) produce risultati a 48 bit è che essi sono i quadrati di numeri a 24 bit, tali bit sono forniti in due cicli successivi di clock.

Si potrebbe erroneamente pensare di sommare prima la parte più significativa al contenuto dei 24 bit più significativi della memoria di accumulazione e poi la parte meno significativa, ma non vi è modo di propagare l'eventuale riporto della somma della parte meno significativa a quella più significativa, ciò evidentemente introduce un errore sul 24-mo bit della somma rendendo di fatto inutile la conoscenza dei 24 bit seguenti.

In pratica si è costretti a considerare come validi nell'accumulazione solamente i bit più significativi, mentre quelli meno sono tralasciati perché divenuti privi di significato.



[1]questo rappresenta una limitazione nell' uso di LH9124 che dispone al suo interno di sommatori a 60 bit, in grado di eseguire le somme relative ai passaggi interni del calcolo della butterfly senza perdite di precisione

Figura A.2. troncamento dei bit meno significativi

Una ulteriore perdita di precisione deve essere introdotta allo scopo di evitare l'overflow, ad esempio volendo sommare il contenuto di 256 spettri sarà necessario eseguire 8 traslazioni verso destra dei bit dello spettro, in sostanza si considerano come significativi solamente i primi 16 bit del modulo, contro i 48 con cui esso è noto !

Nonostante ciò si può dimostrare che per i segnali di spettri bianchi, o quasi-bianchi (ossia praticamente piatti) tale perdita di precisione non costituisce un problema, ed il rapporto rumore-di-calcolo/segnale per l'operazione di accumulazione e media dei risultati è anzi migliore rispetto a quello introdotto da altri passi del processo, come ad esempio lo stesso calcolo della FFT (e quindi anche del campionamento, che essendo a 10 bit non ha un rapporto molto elevato).

Per dimostrare l'affermaione calcoliamo il rapporto rumore segnale che si ottiene nel calcolo dello spettro di potenza secondo il metodo dei periodogrammi modificati, usando registri di accumulazione di 24 bit, con dati iniziali noti a 48, di cui ovviamente sono troncati i 24 meno significativi

.

Consideriamo per semplicità il risultato del calcolo del modulo quadro di uno spettro con una sola linea, supponiamo inoltre che questa sia puramente reale. Il numero di bit del risultato dell' operazione sarà evidentemente doppio rispetto a quello con cui è rappresentata la linea in uscita dal DSP, sommando L quadrati, si devono aggiungere altri Log_2L bit alla lunghezza del risultato per non avere overflow. L'operazione che si esegue è rappresentata da:

$$s(n) = \sum_{i=1}^{L} v_i^2(n)$$

Useremo il seguente schema per simulare l'effetto del rumore introdotto dalla quantizzazione degli operandi, tratto dal libro Phillips-Magle "Digital Control System", nel quale il ragionamento era fatto per il calcolo del rapporto noise/signal di filtri digitali e che è quindi stato modificato opportunamente per adattarlo al problema attuale, pur conservandone le grandi linee:

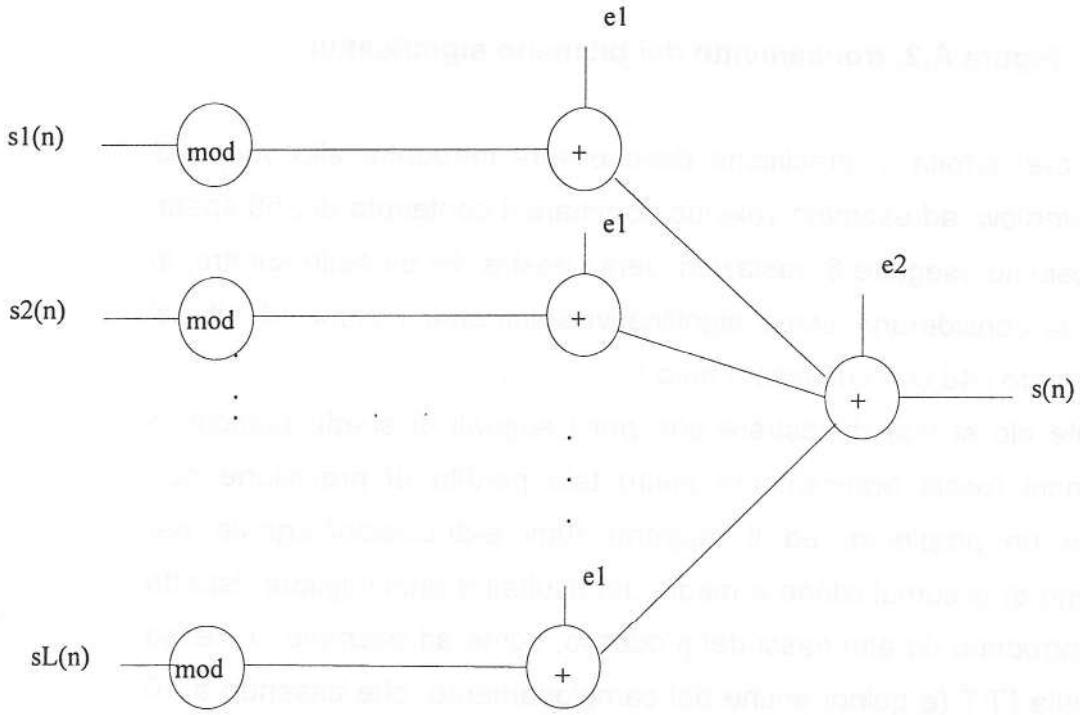


Figura A.3 schema per il calcolo del rumore di quantizzazione

in cui gli $s_i(n)$ rappresentano i valori della riga n-ma dell'i-mo spettro calcolato, e_1 l'errore di troncamento introdotto dall' operazione di elevamento a potenza e_2 quello della sommatoria finale delle L righe.

In sostanza ad ognuna delle operazioni di elevamento a potenza e di sommatoria dei risultati è associato un errore di quantizzazione.

La somma degli effetti di tutti i rumori dà il rumore finale di cui andremo a calcolare la media e la varianza.

Assumiamo innanzitutto che l' errore introdotto dall' elevamento a potenza e dalla sommatoria abbiano la stessa varianza, esaminando lo schema di simulazione, si vede il rumore generato dal calcolo del modulo (mod) ha in uscita, ossia dopo la sommatoria, una varianza:

$$\sigma_\varepsilon^2 = \sum_{i=1}^{L} \sigma_{e_1}^2 = L\sigma_{e_1}^2 = L\sigma_{e_2}^2$$

In altre parole la quantizzazione introdotta dal modulo genera L volte più rumore di quella relativa alla sommatoria.

Esaminiamo dunque la natura dell' errore di quantizzazione dalle due operazioni, cominciamo dal prodotto:

$$v_i = (m_{10}.m_{11}m_{12}....m_{1a}) \times$$

$$v_i = (m_{10}.m_{11}m_{12}....m_{1a}) =$$

$$v_i^2 = (m_{2,0}.m_{21}m_{22}....m_{2a}m_{2,a+1}....m_{2,2a})$$

Un numero a 2a bit che deve essere troncato, per poter essere contenuto nei registri, a soli a bit, ne risulta un errore di quantizzazione:

$$e_r = Q_t^a(v_i^2) - v_i^2$$

in cui con Q^a_t è indicata l'operazione di troncamento ad a bit, ossia:

$$Q_t^a(v_i^2) = (m_{30}.m_{31}m_{32}....m_{3a})$$

consideriamo poi che nell'operazione di troncamento il valore:

$$v_i^2 = (m_0^{"}.m_1^{"}m_2^{"}...m_a^{"}m_{a+1}^{"}m_{a+2}^{"}...m_{2a}^{"})$$

è troncato a:

$$Q(v_i^2) = (m_0^{"}.m_1^{"}...m_a^{"})$$

allora l'errore si può esprimere come:

$$e_r = 2^{-a}(0.m_{a+1}^{"}m_{a+2}^{"}...m_{2a}^{"})$$

che si può poi semplificare ponendo $m_i=m"_{a+1}$ per i da 1 ad a

$$e_r = 2^{-a}(0.m_1m_2...m_a)$$

poiché l'errore di troncamento può assumere valori compresi fra $0 \geq e \geq -2^{-a}$ allora l'errore di quantizzazione ha come valori estremi:

$$0 \geq e_r \geq -2^{-a} + 2^{-2a}$$

la sua funzione di densità di probabilità è discreta e consiste in una serie di funzioni impulsive, ognuna di ampiezza 2^{-a} , ed è espressa come:

$$p(e_r) = 2^{-a}\sum_{i=0}^{2^a-1}\delta\left[e_r - 2^{-2a}i\right]$$

in cui δ è la funzione impulsiva di Dirac. Da qui segue che l'aspettazione matematica dell' errore si può facilmente calcolare come:

$$E[e_r] = \int_{-\infty}^{\infty} e_r p(e_r)de_r = \int_{-\infty}^{\infty} e_r\left\{2^{-a}\sum_{i=0}^{2^a-1}\delta\left[e_r - i2^{-2a}\right]\right\}de_r =$$

$$= 2^{-a}\sum_{i=0}^{2^a-1}\int_{-\infty}^{\infty} e_r\delta\left[e_r - i2^{-2a}\right]de_r$$

ma come è noto:

$$\int_{-\infty}^{\infty} x^r\delta(x-y)dx = y^r$$

cosicché:

$$E[e_r] = 2^{-a}\sum_{i=0}^{2^a-1}\left[2^{-2a}i\right]$$

quindi, poiché evidentemente:

$$\sum_{i=0}^{n} i = \frac{n(n-1)}{2}$$

allora:

$$E[e_r] = \frac{2^a 2^{-2a}(2^a - 1)2^a}{2} = \frac{2^{-a} + 2^{-2a}}{2}$$

A questo punto si può calcolare il rapporto rumore/segnale per poterlo confrontare con l'analogo parametro derivato a suo tempo per il calcolo della FFT e per l'operazione di campionamento del degnale analogico. Per poterlo fare abbiamo però bisogno di conoscere l' aspettazione matematica di e^2_r dopodiché come si sa la varianza si può calcolare semplicemente con la relazione:

$$\sigma^2_{e_r} = E[e_r^2] - E^2[e_r]$$

orbene:

$$E[e_r^2] = \int_{-\infty}^{\infty} e_r^2 p(e_r) de_r = \int_{-\infty}^{\infty} e_r^2 \left\{ 2^{-2a} \sum_{i=0}^{2^a-1} \delta[e_r - i2^{-2a}] \right\} de_r =$$

$$= 2^{-2a} \sum_{i=0}^{2^a-1} e_r^2 \delta[e_r - i2^{-2a}] de_r =$$

$$= 2^{-2a} \sum_{i=0}^{2^a-1} i^2 2^{-4a} = 2^{-5a} \sum_{i=0}^{2^a-1} i^2$$

ma come è noto:

$$\sum_{i=0}^{2^a-1} i^2 = \frac{n(n+1)(2n+1)}{6}$$

e di conseguenza:

$$E[e_r^2] = \frac{2^{-5a}(2^a - 1)(2^a)(2^{a+1})}{6} =$$

$$= \frac{2^{-4a}(2 \cdot 2^{2a} - 2 \cdot 2^a)}{6} = \frac{2^{-2a} - 2^{-3a}}{3}$$

quindi come si è detto la varianza di può calcolare mediante:

$$\sigma^2_{e_r} = E[e_r^2] - E^2[e_r] = \left(\frac{2^{-2a} - 2^{-3a}}{3} \right) - \frac{2^{-2a} + 2^{-4a}}{4} \cong \frac{4 \cdot 2^{-2a} - 3 \cdot 2^{-2a}}{12} = \frac{2^{-2a}}{12}$$

ottenuto trascurando 2^{-4a} e 2^{-3a} rispetto a 2^{-2a} come si può certamente fare per a superiori a poche unità.

Questi risultati valgono per un singolo elevamento a potenza, volendo calcolare la varianza della somma di L diversi spettri dovremo aspettarci un valore:

$$\sigma^2_{\varepsilon} = \sum_{i=1}^{L} \sigma^2_e = L\sigma^2_e$$

nel nostro caso a=24 e L può essere uguale a 256 (ad esempio) allora la varianza globale dell' errore del calcolo della media sarà:

$$\sigma^2_{\varepsilon} = \frac{2^{-24}}{12} 256 = 7.57 \cdot 10^{-14}$$

e dunque $\sigma_{e_r} = \sqrt{\sigma_{e_r}^2} = 2 \cdot 10^{-7}$

mentre la media dell' errore è stimata pari a :

$$E[e_r] = \frac{2^{-a} + 2^{-2a}}{2} = \frac{2^{-24} + 2^{-48}}{2} = 2.98 \cdot 10^{-8}$$

Quindi in generale utilizzando il troncamento ai 24 bit più significativi per il calcolo della media degli spettri di potenza non è possibile considerare nullo l'errore di arrotondamento che si viene a creare.

Per quanto riguarda invece il rapporto noise/signal si può derivare la varianza del segnale per un segnale di tipo bianco dal fatto che esso, per evitare l'overflow, deve essere compreso fra -1/L e 1/L il che significa che per distribuzioni di probabilità uniformi, quale si suppone per i segnali in ingresso:

$$\sigma_X^2 = E\left[\left|X(n)\right|^2\right] = \frac{1}{3L^2}$$

Essendo lo spettro di potenza determinato dalla somma dei moduli quadri di L misurazioni, ossia:

$$\sum_{i=1}^{L} X(n)^2$$

per trovarne il valor quadratico medio occorre conoscere quello di:

$$E\left[\left|X_n\right|^4\right] = \frac{1}{3}\left(\frac{1}{2L}\right)^4$$

quindi la varianza della media è data dalla somma delle varianze dei segnali componenti:

$$E\left[\left|S(n)\right|^2\right] = \sum_{i=0}^{L} E\left[\left|X(n)\right|^4\right] = L \cdot E\left[\left|X(n)\right|^4\right] = \frac{1}{48L^3}$$

da cui si ricava facilmente il rapporto rumore/segnale

$$\frac{E\left[\left|F(n)\right|\right]}{E\left[\left|X(n)\right|^4\right]} = \frac{\frac{2^{-2a}}{12} \cdot L}{\frac{1}{48L^3}} = 2^{-2a} \cdot 4L^2 = 2^{-2(a+1)} L^2$$

A.5.2.sommatori esterni

Il secondo metodo prevede l'uso di sommatori a virgola fissa esterni a LH9124, posti insieme alla loro logica di controllo su una scheda a parte, collegabile a quella di DSP mediante un bus dedicato ad alta velocità.
Il loro uso è per questo più oneroso dal punto di vista della complessità circuitale, ma permette di eseguire le somme senza generare rumore di troncamento, ad esempio usando sommatori a 60 bit si possono avere fino a 2^{12}=4096 accumulazioni senza alcuna perdita di precisione.
Non pare molto conveniente usare un vero e proprio microprocessore per eseguire le somme in quanto esso necessita di una sua circuiteria di servizio che appesantisce inutilmente la progettazione aumentandone ulteriormente le difficoltà, e risulta alla fine sottoutilizzato, essendo dedicato esclusivamente alla esecuzione di somme (a meno che non si riesca ad utilizzarlo anche per altri scopi).
E' questa però la soluzione attualmente in uso sul sistema installato presso il laboratorio di Radioastronomia del CNR, vedremo nel prossimo capitolo i pregi e i difetti di tale soluzione.
La soluzione che andremo ad esporre prevede l'uso di chip che eseguono esclusivamente operazioni di somma.
Oltre alla maggiore precisione l'uso di sommatori esterni migliora anche le prestazioni relative ai tempi di calcolo globali del sistema, infatti usando i sommatori contenuti in LH9124 occorrono due uteriori passi di programma oltre al CMAG per calcolare la somma, una per la somma vera e propria ed una per

La principale difficoltà che si incontra nella realizzazione di tale struttura risiede nel reperire sommatori in grado di eseguire operazioni con dati di dimensioni varianti dai 56 ai 60 bit con tempi di latenza brevissimi, di non più di 5-10 ns, in quanto il restante tempo del periodo di clock è necessario per poter campionare il dato sulla memoria RAM che funge da accumulatore del risultato.

Tale difficoltà si può aggirare duplicando la struttura dell' averager in modo da renderla parallela, sfuttando il particolare formato con cui LH9124 genera i dati in uscita dal calcolo dello spettro di potenza (CMAG).

Infatti come spiegato ampiamente più sopra, esso produce sui morsetti i dati a 48 bit in due cicli consecutivi di clock, nel primo viene trasferita la parte più significativa di una parola pari sui morsetti corrispondenti alla parte reale e di una dispari sui quelli relativi alla parte immaginaria, nel secondo ciclo sono invece messi a disposizione i 24 bit meno significativi sempre delle parola pari e dispari, nelle medesime configurazioni.

Dunque per ottenere la parola a 48 bit da sommare a quella a 56 contenuta nell'accumulatore occorre aspettare due cicli di clock anche se evidentemente la somma vera e propria può essere iniziata solo al secondo ciclo quando tutta la parola è nota e campionata in un latch in ingresso al sommatore. In pratica si esegue una somma ogni due cicli di clock, e tale fatto si può usare per raddoppiare i tempi massimi di latenza dei sommatori, ciò si realizza utilizzando otto latch a 24 bit e non uno ma due sommatori con un numero di bit adeguato alla precisione e al numero di somme che si vuole ottenere (56 bit nel nostro esempio).
I risultati delle accumulazioni sono messi in due blocchi distinti di RAM, uno contiene i dati di indice pari e uno quelli di indice dispari, e le elaborazioni avvengono di fatto in parallelo sui due sommatori.
Lo schema dell' averager in parallelo è riportato ancora in allegato.
In pratica il circuito è costituito da una batteria di registri, che servono a memorizzare secondo un pattern predefinito i dati provenienti da LH9124, secondo i comandi provenienti da una apposita logica, di cui si vedranno più avanti le funzioni.
A questo primo stadio ne segue un secondo, che è quello dei sommatori i cui ingressi provegono da uno dei registri, selezionato da un apposito segnale di output enable (i registri non selezionati devono essere in three state).
I risultati sono accumulati in una apposita RAM che, insieme con i suoi circuiti di controllo e di generazione degli indirizzi, costituisce il terzo stadio del sistema.
Occorrono, come vedremo, due generatori di indirizzi, uno per quelli di lettura ed uno per quelli di scrittura, il primo può essere un semplice contatore in grado di contare fino a 2^{20} (ossia a 20 bit), con comando di

reset per ricominciare il conteggio (non è necessario che sia programmabile in quanto deve sempre generare la stessa sequenza di indirizzi, quando la sequenza è finita provvede il segnale di reset a ricominciare il conteggio), mentre per il secondo si è preferito usare un LH9320, ossia un altro address generator come quelli usati per equipaggiare LH9124, questo perché si possa aumentare la flessibilità del sistema, utilizzando questa RAM come buffer di uscita dei dati provenienti direttamente da LH9124 (ossia senza subire l'operazione di accumulazione), in questo caso deve essere possibile disabilitare il conteggio dell' altro generatore, per evitare conflitti sulla RAM.
Analizziamo l'ordine con cui avvengono le operazioni: al primo colpo di clock vengono latchati nei corrispondenti registri il bit più significativo della $|X(2i)|^2$ (indici pari) e della $|X(2i+1)|^2$ (indici dispari), al secondo invece sono campionate le due parole meno significative delle stesse grandezze.
La gestione dei segnali di campionamento è effettuata da una apposita unità di controllo, in modo che il dato in uscita da LH9124, presente sui morsetti di ingresso di tutti i registri, venga memorizzato solo su quelli che interessano in quel determinato istante.
I dati memorizzati sui registri sono poi portati in ingresso ai sommatori, che essendo due agiscono in parallelo sui dati di indice pari e dispari, e possono dunque avere un tempo di latenza doppio rispetto al caso in cui si utilizzi un solo sommatore.
Le somme sono quindi inserite in una memoria di accumulazione dalla quale sono anche prelevati i secondi addendi in ingresso ai sommatori

architettura

Commentiamo ora più in dettaglio gli schemi allegati relativi all'architettura dell' unità di averaging e delle temporizzazioni.
I dati, provenienti dalla porta Q di LH9124, vengono come si è detto, memorizzati temporaneamente in una serie di registri a 24 bit (per la precisione 8 registri).

Il segnale di scrittura per ciascun registro proviene da una apposita rete logica, realizzabile facilmente mediante una PAL a 3 ingressi e 4 uscite. Il generatore di indirizzi LH9320, oltre a fornire gli indirizzi alla RAM produce anche un segnale di MEMWR* che serve a comandare la scrittura del dato in uscita dalla porta Q, quando questo è pronto. Tale segnale si può usare come ingresso alla PAL, insieme alle uscite di un contatore per 4 che ha lo stesso clock del sistema (si è visto come i dati alle uscite di LH9124 si ripresentino con pattern che ha una periodicità di 4 cicli di clock , quando si esegue una operazione CMAG)

Le varie configurazioni dei due bit di uscita del contatore servono a selezionare i registri sui quali si deve campionare i dati.

La tabella della verità di tale rete è dunque:

MEMWR*	CONT0	CONT1	EN1	EN2	EN3	EN4
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	1	1	0	0	0	1
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	0	0	0	0
1	1	1	0	0	0	0

Tabella A.1 tabella della verità della rete di controllo dei registri dell' averager

In sostanza, come si vede anche dallo schema, se MEMWR* è attivo sono attivati in sequenza:

1. i registri i^- e $i+2^-$
2. i registri i^+ e $i+2^+$
3. i registri $i+1^-$ e $i+3^-$
4. i registri $i+1^+$ e $i+3^+$

proprio nell' ordine in cui i dati corrispondenti escono da LH9124.

In pratica ciò assicura che i dati omologhi vengano sempre memorizzati sugli stessi registri, ad esempio nel registro 1 saranno contenuti le parti

meno significative dei dati di indice i nel 2 le più significative, in 3 le mono dei dati i+1 e così via.
Ancora più semplice è la generazione dei segnali di output enable per i registri infatti si vede che il loro contenuto deve essere reso disponibile quando sono presenti sia la parte più significativa che quella meno del dato. In pratica poiché un dato è acquisito in due cicli di clock si può usare come segnale di abilitazione il primo bit di uscita del contatore, che in pratica funge da divisore per due del clock do sistema.
Si noti che in questo modo si dà il segnale di abilitazione all'output dei registri anche se essi non contengono dati significativi, ma questo non ha importanza perché comunque, come si vedrà, non si hanno effetti sul contenuto della RAM di average, che è fatta in modo di campionare i dati solo quando questi sono significativi.
In pratica in ingresso ai due sommatori si presentano così, alternativamente:

sommatore pari	sommatore dispari
i^- e i^+	$i+1^-$ e $i+1^+$
$i+2^-$ e $i+2^+$	$i+3^-$ e $i+3^+$

con una commutazione del valore dei dati ogni due cicli del clock di sistema.
Contemporaneamente all' altro ingresso del sommatore è già stato portato il contenuto della cella corrispondente della RAM, immagazzinata in un apposito registro durante il ciclo precedente rispetto a quello in cui si è messo a disposizione il dato valido, ossia il ciclo in cui erano stati generati da LH9124 i bit meno significativi della parola di uscita.
Dalla tabella delle temporizzazioni si può, a questo punto, facilmente calcolare il tempo massimo che a disposizione del sommatore per calcolare la somma.
Cominciando a contare i cicli da quello in cui si presenta la parte meno significativa dell' i-mo dato si vede che il dato risulta effettivamente disponibile durante il terzo ciclo, in particolare lo si può considerare valido

da quando va basso il segnale di MEMWR* proveniente dal generatore di indirizzi, cosa che avviene tipicamente 5 ns dall' inizio del sysclock.
Il dato rimane valido fino a che il MEMWR* non va di nuovo attivo, considerando che, secondo le tavole di temporizzazione dei manuali di LH9320 tale segnale rimane attivo da circa 16ns dall' inizio del ciclo relativo, fino a circa 5 del ciclo successivo si deduce che, essendo la durata del ciclo di clock a 40 MHz di 25ns il tempo di calcolo è approssimativamente di 25-5+16 =36ns, questo utilizzando memorie con un tempo di accesso di non più di 14ns.
I dati sommati al contenuto della RAM di accumulazione devono poi essere in essa riscritti. Il generatore di indirizzi per la RAM di averaging costituisce uno dei problemi del sistema, infatti non è possibile utilizzare un semplice contatore perché come si è visto gli indirizzi a cui si accede non sono consecutivi ma hanno un pattern particolare che andremo ora ad esaminare.
L'averager elabora i dati in modo da trattare contemporaneamente due parole una di indice pari e una dispari. Conviene, come si vedrà, dividere logicamente la memoria in due blocchi da 512 Kbyte ciascuno, uno per i dati pari e uno per quelli dispari. Vediamo una tabella che illustra la successione delle operazioni sulla RAM :

blocco pari	blocco dispari
read (0)	read(1)
-	-
read (2)	read(3)
write(0)	write(1)
read(4)	read(5)
write(2)	write(3)
read(6)	read(7)
....	....
read(2i)	read(2i+1)
write(2i-2)	write(2i-1)
read(2i+2)	read(21+3)
write(2i)	write(2i+1)

Tabella A.2 sequenza delle operazioni di lettura e scrittura in memoria di accumulazione

Tale tabella fa riferimento ad indirizzi "assoluti" ossia assumendo che vi sia un solo blocco di memoria.
La stessa sequenza di istruzioni con indirizzi "relativi", cioè riferiti ad una struttura a due blocchi diventa :

blocco pari	blocco dispari
read (0)	read(0)
-	-
read (1)	read(1)
write(0)	write(0)
read(2)	read(2)
write(1)	write(1)
read(3)	read(3)
....	....
read(i)	read(i)
write(i-1)	write(i-1)
read(i+1)	read(i+1)
write(i)	write(i)

Tabella A.3 ordine delle operazioni di lettura e scrittura su ciascun blocco di memoria

quindi in effetti gli indirizzi "logici" sono gli stessi per i due blocchi, ciò diventa peraltro evidente se si nota che l'architettura del sistema è replicata in parallelo per i due blocchi.
D'altronde all' atto della lettura dei risultati finali tale divisione deve cadere e si deve poter accedere ai dati in maniera sequenziale, cioè l'intera struttura di generazione degli indirizzi deve essere trasparente per chi legge. Vedremo poi come questo si possa realizzare. Per ora basti distinguere i due regimi in cui deve operare il generatore di indirizzi: uno

per la elaborazione dei dati, che prevede un pattern particolare e uno per la lettura dei risultati in cui l'ordine deve essere sempre lineare.
Gli indirizzi di scrittura sono forniti direttamente da LH9320 mentre come si è detto quelli di lettura provengono dal contatore ausiliario, questo anche se in realtà LH9124 si incrementa ad ogni ciclo del sysclock, per fornire i corretti segnali di lettura e scrittura, un apposito multiplexer provvede ad inviare alla RAM i dati corretti per quell'istante. La generazione degli indirizzi di scrittura comincia con un ciclo di latenza rispetto a quelli di lettura per consentire l'esecuzione della somma.
Poiché un solo indirizzo deve servire sia per il blocco pari che per quello dispari si deve ricorrere ad un trucco per avere gli indirizzi esatti per ciascun blocco: in pratica si genera solo la metà degli indirizzi necessari (ad esempio per indirizzare 1M di memoria si conta solo fino a 512K) e quindi si completa l'indirizzo mettendo uno 0 in coda agli indirizzi pari e un uno in coda a quelli dispari.
Per poter considerare la memoria come un tutto unico, cosa che è utile ad esempio quando la si deve leggere, è bene che sia possibile bypassare tale meccanismo, in modo che l' AG-LH9320 veda tutta la memoria, per poterlo fare l'ultimo bit degli indirizzi viene selezionato mediante un multiplexer 2:1 utilizzando un segnale apposito proveniente dalla mother board e che qui viene chiamato VME (per indicare che si sta' eseguendo una lettura da parte del VMEbus) e che seleziona appunto il bit 0 per i pari o quello 1 per i dispari oppure l'ultimo bit di indirizzo proveniente dal generatore.
Rimane da esaminare il modo con cui vengono forniti gli indirizzi e i segnali di abilitazione ai due blocchi di RAM.
Per quanto riguarda i segnali di RD*, di WR* e di CS le cose sono simili a quanto visto per i registri di memorizzazione, ossia si può usare ancora i segnali provenienti da LH9320 divisi però per due perché la scrittura e la lettura avvengono un ciclo di clock si ed uno no (in realtà deve essere possibile bypassare la divisione per due sempre per garantire un accesso diretto alla memoria da parte di LH9320). L'usare i segnali del generatore di indirizzi assicura tra l'altro l'esattezza della temporizzazione del segnale, sempreché si possa trascurare il tempo necessario ad aggiornare le uscite

del divisore di frequenza (un contatore per 2), in ogni caso si può modificare la temporizzazione del segnale variando il valore di una resistenza collegata ad uno dei piedini del generatore di indirizzi.
Quanto detto è riassunto nella seguente tabella delle temporizzazioni dei segnali coinvolti, le durate temporali sono ricavate dai manuali di uso di LH9320 e di LH9124:

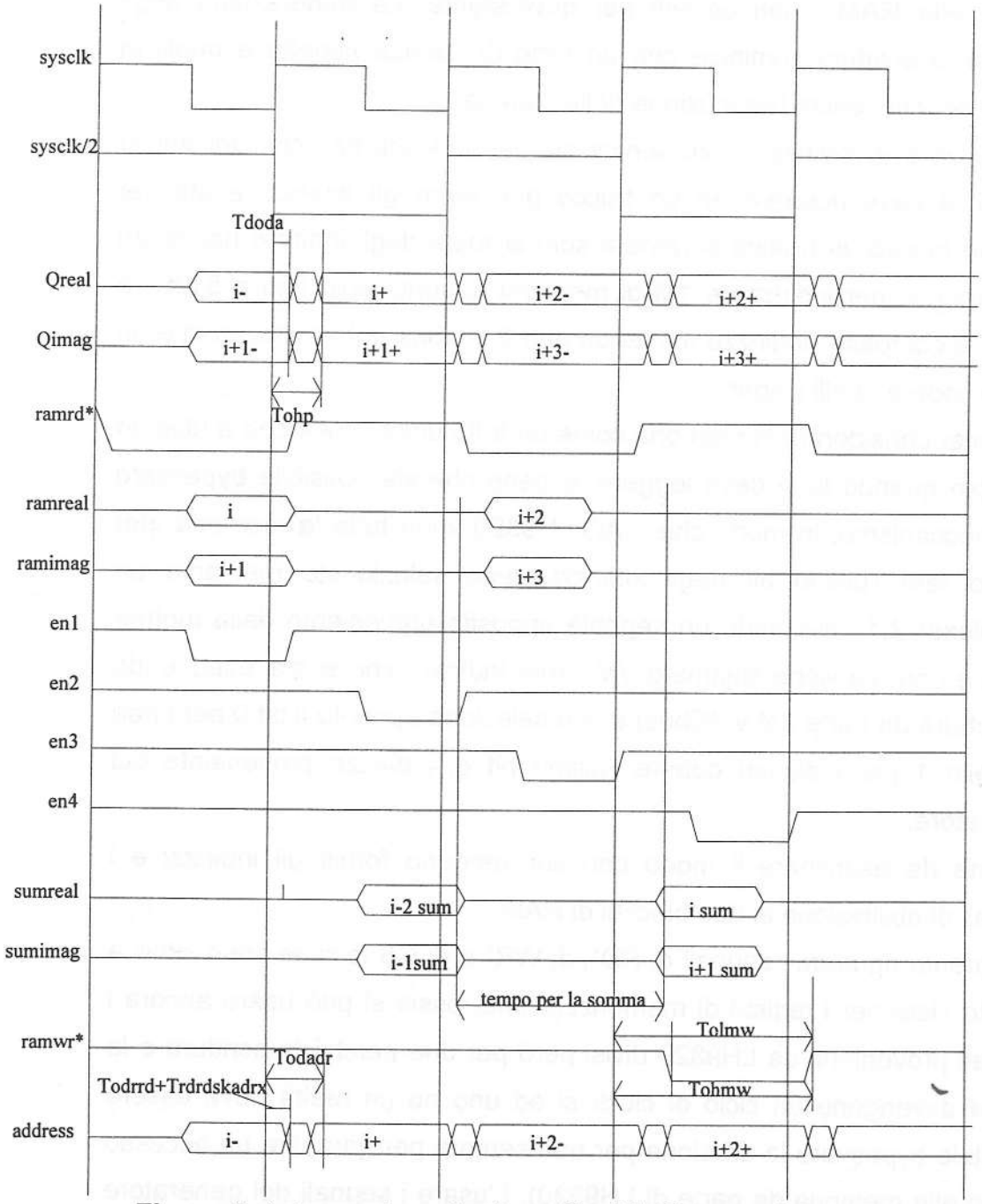


Figura A.4. temporizzazioni per l'unità di averaging

A.6.Simulazione del calcolo di uno spettro di potenza di un segnale rumoroso eseguito col metodo Bartlett-Welch.

Per mostrare un esempio dell' applicazione di quanto detto in questo capitolo si è simulato il calcolo della media di una serie di spettri di potenza di un segnale rumoroso, calcolati col simulatore software del chip LH9124 e quindi sommati in modo da ridurre gli effetti del rumore sullo spettro.
Per generare il segnale rumoroso si è fatto uso di un programma in Pascal, introducendo il rumore mediante la somma di numeri casuali generati dal programma stesso, il che con ottima approssimazione simula la presenza di un rumore bianco sommato al segnale.
Si è imposto un livello massimo del rumore di non più di 1/100 di quello del segnale in modo da avere un segnale rumoroso ma non troppo, il che imporrebbe il calcolo di un numero molto grande di spettri da mediare per filtrare il rumore.
Il segnale base usato come prova è costituito dalla somma di 10 armoniche.
Il seguente grafico mostra lo spettro di potenza del segnale senza rumore, il che evidenzia la posizione e l'ampiezza delle linee dello spettro e sopratutto servirà da confronto con i risultati dell' averaging.
I valori riportati in ordinata non sono normalizzati, secondo la consuetudine della rappresentazione in virgola fissa, ma hanno uno spettro di variazione da 0 a 2^{23} in quanto, trattandosi di spettri di potenza, hanno sempre valori positivi, questa convenzione si è usata per comodità di trattamento dei dati in uscita dal simulatore e per il fatto che non cambia qualitativamente l'aspetto dello spettro (è solo una questione di scala diversa rispetto al formato standard).
Si sono quindi calcolati gli spettri di 24 segnali rumorosi, 24 non è un numero molto elevato per ottenere una buona stima della media (un buon valore sarebbe almeno di 100), ma costituisce un compromesso fra la lunghezza delle operazioni di calcolo e la significatività dei risultati che si riescono ad ottenere, in pratica è il numero minimo di spettri che consente

di apprezzare miglioramenti nel calcolo della media secondo Bartlett-Welch.

Un paio di esempi di spettri di segnali rumorosi sono riportati di seguito per il confronto con lo spettro mediato e con lo spettro ideale del segnale privo di rumore:

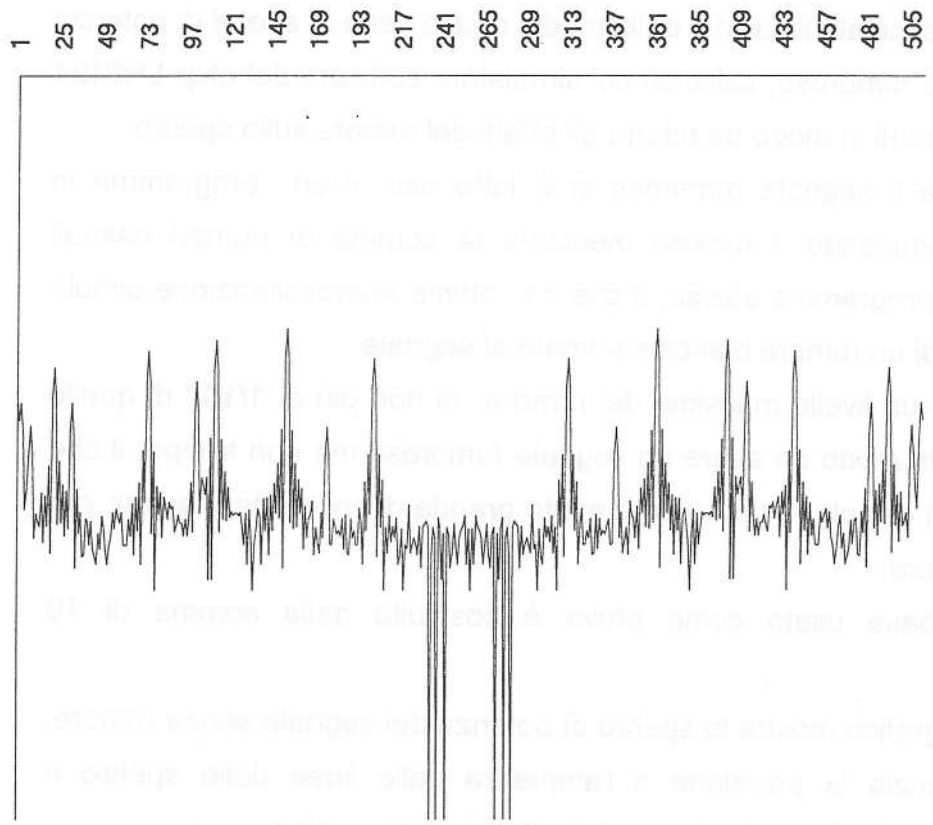


Figura A.5. spettro di segnale rumoroso (1)

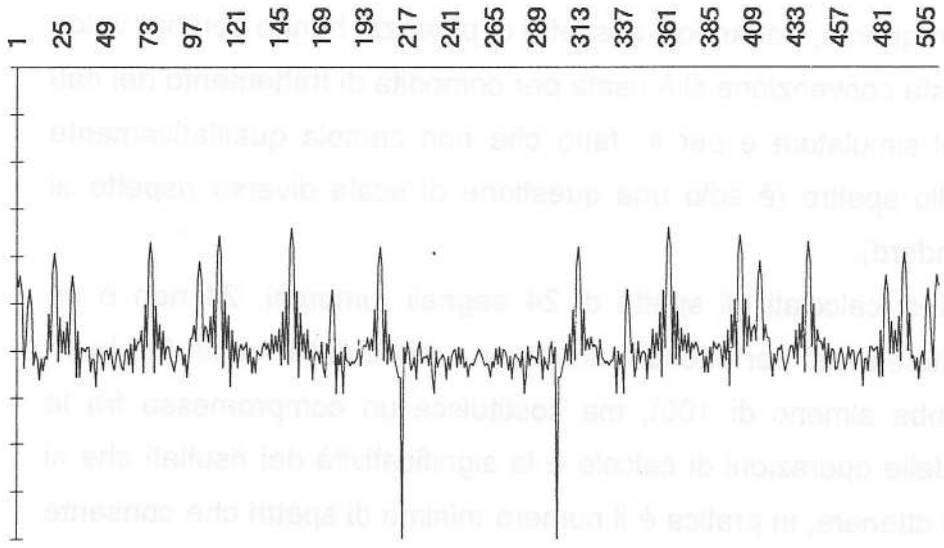


Figura A.6 spettro di segnale rumoroso (2)

Infine si mostra il grafico relativo alla media dei 24 spettri, eseguita col metodo Bartlett-Welch:

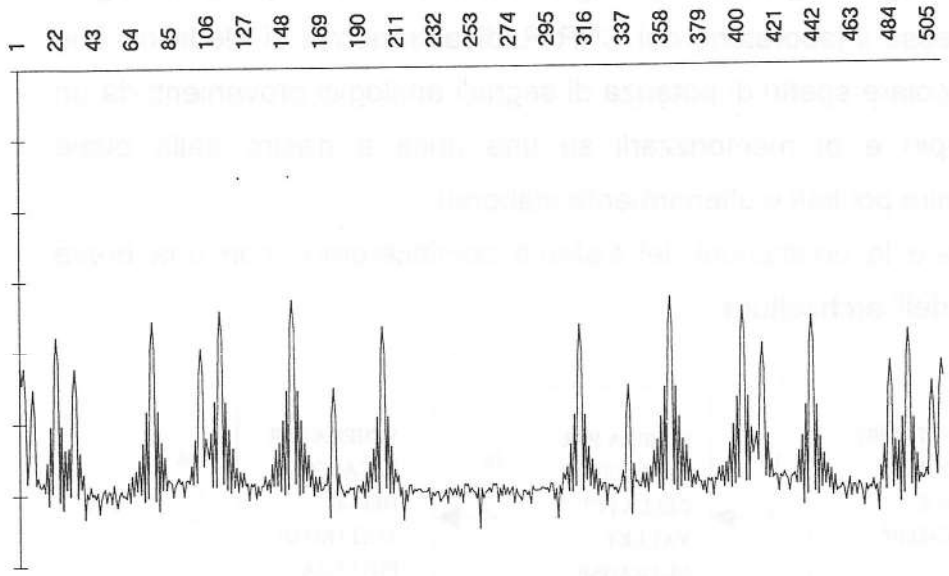


Figura A.7 media degli spettri di 24 segnali rumorosi, con il metodo di Bartlett-Welch

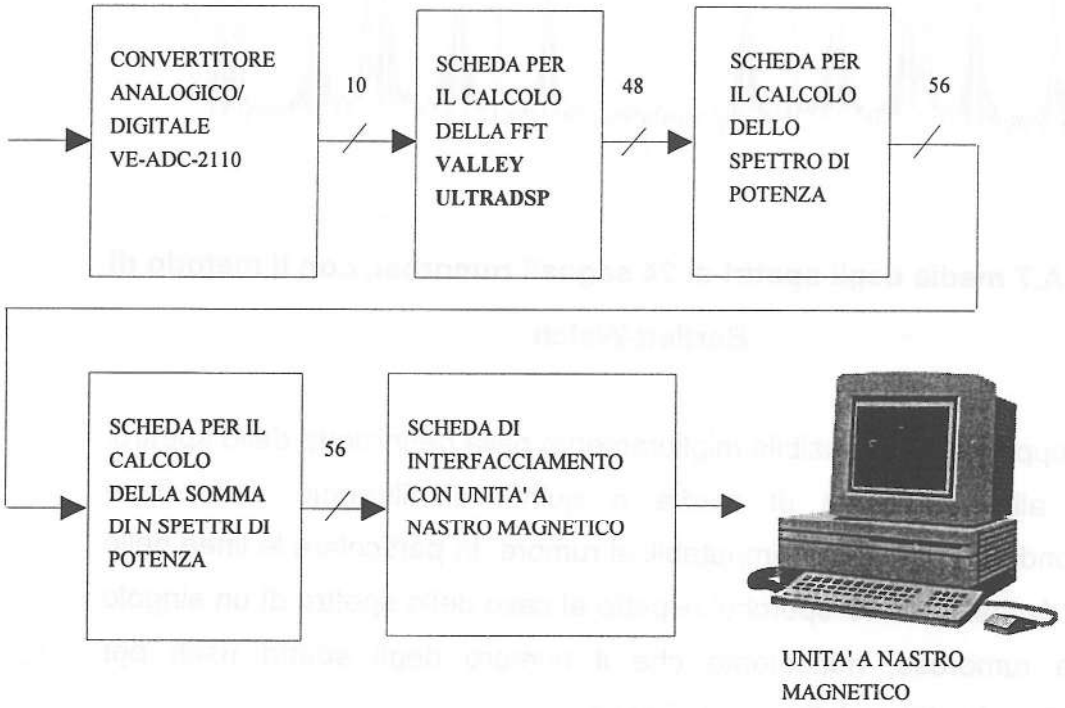
Si può apprezzare il sensibile miglioramento nella definizione dello spettro, dovuta all' operazione di media e quindi di filtraggio delle linee corrispondenti a frequenze imputabili al rumore. In particolare le linee dello spettro risultano meno 'sporche' rispetto al caso dello spettro di un singolo segnale rumoroso, nonostante che il numero degli spettri usati per l'averaging sia stato relativamente basso.

A.7 LA SCHEDA ULTRADSP DELLA VALLEY TECHNOLOGIES

A.7.1.breve descrizione dell' architettura del sistema di digital signal processing (DSP)

Il sistema di digital signal processing prodotto dalla Valley Technologies installato presso il laboratorio del CNR Radioastronomia di Medicina è in grado di calcolare spettri di potenza di segnali analogici provenienti da un radiotelescopio e di memorizzarli su una unità a nastro dalla quale possono venire poi letti e ulteriormente elaborati.

Per analizzare le prestazioni del sistema cominceremo con una breve descrizione dell' architettura



"Figura A.8 schema dell'architettura del sistema di digital signal processing

In sostanza esso è formato da cinque schede residenti in altrettanti slot di un bus VME (vedi appendice A) che viene usato solo per l'inizializzazione del sistema e per la sua programmazione, mentre i dati e i segnali di sincronizzazione fra le schede sono trasportati da bus dedicati.

Per comodità nei riferimenti successivi si riportano le denominazioni di ciascuna scheda, secondo la nomenclatura della stessa Valley:

1.**convertitore analogico digitale VE-ADC-2110**

2.**scheda UltraDSP** con a bordo un processore **LH9124**, uno **AT&T DSP32C** ed equipaggiata con mezzanine board **type 1** (2 Kbyte FIFO) in grado di eseguire trasformate di Fourier discrete di record di lunghezza non superiore ai 256Ksamples, ad una velocità di clock di 40Mhz, quindi estremamente elevata.

3.**unità di conversione,** esegue la conversione dei dati in uscita dall' UltraDSP dal formato block-floating-point verso il formato floating point e ne calcola il modulo quadro, il tutto in tempo reale, ossia in modo sincrono rispetto al flusso di uscita dei dati dalla scheda UltraDSP.

4.**Unità di accumulazione** dei risultati per il calcolo dello spettro di potenza secondo il metodo di Bartlett-Welch, è costituita in pratica da una seconda scheda UltraDSP, identica alla prima ma fortemente sottoutilizzata rispetto alle sue capacità, in quanto viene effettivamente usato solamente il processore AT&T DSP32C per il calcolo della media in formato floating-point delle uscite delle elaborazioni di un certo numero definibile dall'utente di spettri di potenza, allo scopo di ridurre l'influenza del rumore sulla misura.

Esaminiamo dunque un po' più in dettaglio i componenti del sistema:

A.7.1.1. Il campionatore VE-ADC-2110

Riportiamo preliminarmente alcune specifiche d'uso del convertitore analogico-digitale, tratte dai manuali d'uso forniti dalla Valley:

risoluzione	10 bit
voltaggi di ingresso a fondo scala	±0.5 V
larghezza di banda	da 0 a più di 100Mhz
impedenza di ingresso	50 ohm
sampling rate	da 0 a 40 MSPS
TTL clock threshold	+1.8 V
impedenza di ingresso (terminale w/o)	10K ohm

rapporto segnale/rumore per un singolo tono a 5Mhz con un sampling clock di 21 MSPS	52 dB
livello segnale di uscita	±1.25 V_{peack}

Tabella A.4 specifiche del convertitore A/D

Una serie di jumper può essere settata per variare il formato di uscita dei dati digitali dal convertitore, per poter essere interfacciato direttamente con la scheda UltraDSP tali dati devono essere sempre in complemento a due. Dallo stesso manuale è tratto lo schema a blocchi del convertitore allegato. Per quanto riguarda le operazioni di acquisizione e di trasferimento dei dati verso la scheda il convertitore usa i piedini definiti dall'utente del bus VME che sono situati sul modulo P2 dell' interfaccia, tali piedini sono in pratica dei corti circuiti quindi il trasferimento non è effettuato con il protocollo tipico del VMEbus ma può essere gestito dall'utente.

Per bufferare i dati VE-ADC-2110 è dotato di una piccola FIFO (1024 byte) quindi prima di iniziare il trasferimento è necessario prevedere un reset di tale FIFO dato direttamente al convertitore. il campionamento inizia quando va basso un apposito segnale di WR_EN*, non sovrapposto al reset.

I segnali di FIFO_RESET* e di WR_EN* dovrebbero rimanere bassi per non meno di tre cicli di clock.

Una volta iniziata l'acquisizione dei dati si può iniziare a trasferirli dalla FIFO verso la mezzanine board del DSP. Per regolare tale flusso di dati sono presenti in uscita dal convertitore tre flags :

FIFO_MT*	FIFO vuota
FIFO_HALF*	FIFO mezza piena
FIFO_FULL*	FIFO piena

Tabella A.5 flag di stato della FIFO

tali segnali sono letti dalla scheda DSP(in particolare sono indirizzati al DSP32C che provvede in base ad essi a generare i segnali di lettura).

Per leggere la FIFO occorre mettere bassi due segnali RD_EN* e DATA_OE*, dopodiché i dati sono letti al rate imposto dal RD_CLK, che può essere variato a seconda delle esigenze di lettura dello stadio che segue (ad esempio nel nostro sistema, con una mezzanine board type 1 il rate non può essere superiore a 36MSPS).

La lettura può iniziare in qualsiasi momento dopo che il FIFO_MT* è andato alto, il che indica la presenza di dati sulla FIFO.

L'ammontare dei dati che possono essere acquisiti e trasferiti dipende dal rapporto fra i segnali di CLOCK_IN e di RD_CLK, il primo determina infatti il rate del campionamento e il secondo quello del trasferimento. Se CLOCK_IN è più veloce di RD_CLK la FIFO si riempie e il processo di acquisizione termina, se è più lento il processo di acquisizione e di trasferimento può continuare indeterminatamente, salvo se la FIFO risulta vuota, al che il trasferimento deve interrompersi in attesa dell' arrivo di nuovi dati.

L'elenco completo dei segnali segnali necessari al funzionamento del digitalizzatore sono riassunti nella seguente tabella:

SIGNAL_IN	input	segnale analogico di ingresso
CLOCK_IN	input	clock di campionamento
FIFO_RESET*	input	segnale di azzeramento della FIFO interna
WR_EN*	input	abilitazione alla scrittura della FIFO
RD_CLK	input	clock di lettura dalla FIFO
RD_EN*	input	abilitazione di lettura della FIFO
CHANID*(0..2)	input	identificatore del canale
ID_INSERT	input	1=inserisci il channel ID nel bus dati 2=inserisci 0 nel bus dati
DATA_OK*	input	abilita l'uscita dei dati sul bus dati
SYNC_IN*	input	legato a SYNC_OUT*
DAC_IN(4-15)	input	dati esterni per il bus dati
DAC_LATCH	input	campionatore di dati esterni
DAC_IN_EN	input	registro interno tri-state per dati esterni
SPARE_IN	input	non usato
CLOCK_OUT	output	uguale a CLOCK_IN
DAC_OUT*	output	segnale DAC non filtrato in uscita
SYNC_OUT*	output	segnale di sincronizzazione per digitalizzatori multipli
FIFO_MT*	output	flag di FIFO vuota
FIFO_HALF*	output	flag di FIFO mezza piena
FIFO_FULL*	output	flag di FIFO piena
DATA(0-15)	output	dati in uscita (tri-state)

Tabella A.6 segnali di comando convertitore anlogico/digitale

I segnali necessari alla sincronizzazione, come si era già accennato, sono generati sulla scheda UltraDSP dal processore DSP32C in base all' esame dei flag di stato della FIFO che provvede ad inviare i segnali di lettura RD_EN* e DATA_OE* con le opportune temporizzazioni.

Come sempre per ultriori informazioni si rimanda al manuale d'uso del convertitore analogico digitale.

A.7.1.2. la scheda UltraDSP della Valley Technologies

Costituisce il cuore del sistema di calcolo. E' equipaggiata con un **LH9124** e con tre **LH9320** per la generazione degli indirizzi di tre blocchi di RAM (due da 128×48Kword ed uno, associato alla porta Q di LH9124, da 128× 64 perché possa contenere dati in formato floating point).
Inoltre per la gestione dei segnali della scheda ed in particolare del protocollo DMA col VMEbus è prevista la presenza di un microprocessore **AT&T DSP32C** che oltre a ciò permette la programmazione dei dispositivi presenti sulla scheda in linguaggio C ed esegue operazioni in floating point.

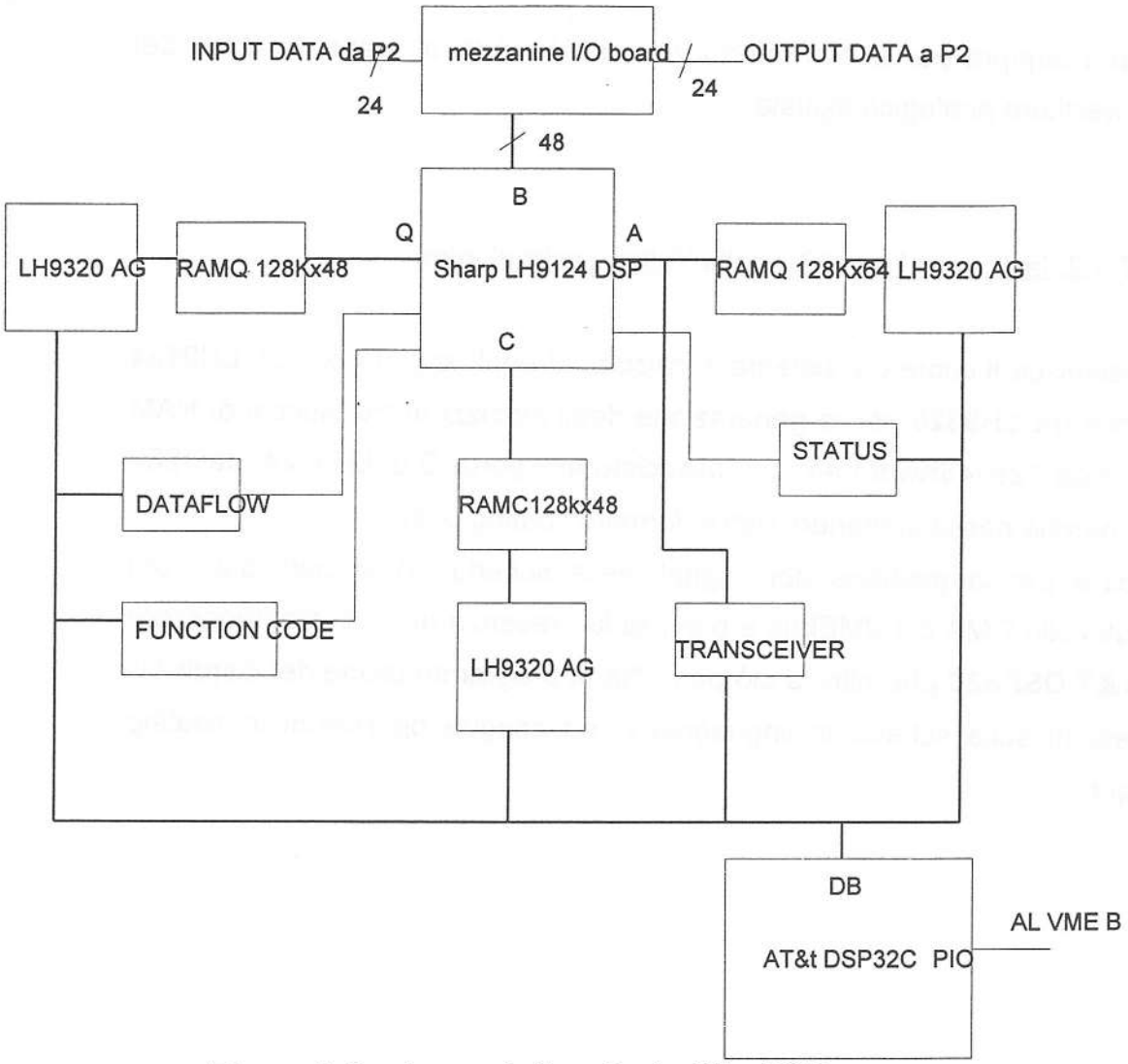


Figura A.9 schema della scheda UltraDSP

Per l'acquisizione dei dati provenienti dal convertitore analogico-digitale e per il deposito di quelli elaborati ed in attesa di essere letti, è possibile collegare, attraverso due appositi connettori J1 e J2 presenti sulla scheda madre, una **mezzanine board type 1** costituita in pratica da due memorie FIFO da 2K×48, una per i dati in ingresso e una per quelli in uscita e dalla necessaria logica di controllo. Per ovviare agli inconvenienti insiti in tale scheda di interfaccia saranno presto disponibili anche delle mezzanine board dette type 3 dotate di due blocchi da 128×48K SRAM.

Assieme alla scheda viene venduto un software di gestione per i sistemi operativi UNIX oppure DOS (al CNR radioastronomia è installata la versione per UNIX).

Vediamo un tipico flusso dei dati che devono essere elaborati dalla scheda: i dati, i segnali di controllo ed il clock entrano attraverso la colonna A del connettore P2 del VMEbus (vedi appendice A) e sono girati direttamente al connettore J1 della UltraDSP da dove entrano nella mezzanine board nella quale sono temporaneamente immagazzinati.
Durante il primo passo del processo i dati sono letti attraverso la porta B di LH9124, passando per il connettore J2 (diverso quindi da quello usato per i dati in ingresso) ed immagazzinati nel blocco di RAM associato alla porta A o a quella Q, ai passi successivi i dati sono alternativamente letti dalla porta A e scritti nella Q e viceversa. Contemporaneamente dalla porta C entrano i coefficienti di twiddle o quelli per le finestre.
All'utimo passo i dati escono dalla porta B verso la mezzanine board ove sono depositati temporaneamente e da dove vengono letti dall'esterno attraverso il connettore P2 (alternativamente possono essere messi nella RAMA e da qui letti direttamente dal VME, infatti RAMA può essere indirizzata ed acceduta direttamente dal VMEbus).

A.7.1.3. generalità sulla scheda mezzanine

La scheda per l'acquisizione dei dati non è collegata direttamente ad una delle backplanes del bus VME ma a due connettori J1 e J2 della scheda DSP dimodochè i dati che la interessano entrano in quest'ultima per i pin del connettore P2 del VMEbus e quindi vengono girati attraverso il connettore J1 della scheda stessa alla mezzanine board dove vengono bufferati temporanemente su una FIFO da 2×48K divisa in due banchi da 24 byte l'uno (il che permette di bufferare direttamente i dati reali in due banchi, uno pari e uno dispari, per l'esecuzione di una trasformata reale da 2N punti mediante una complessa da N punti, secondo l'algoritmo nel capitolo 2).
Dalla scheda DSP provengono anche tutti i segnali di controllo della FIFO e delle altre operazioni che la mezzanine board è in grado di eseguire.
I dati bufferati sono letti dalla scheda DSP quando essa lo ritiene opportuno passando per la porta B di acquisizione dei dati di LH9124 (non

passando quindi per il VME).Un diagramma esemplificativo di quanto detto è mostrato qui di seguito:

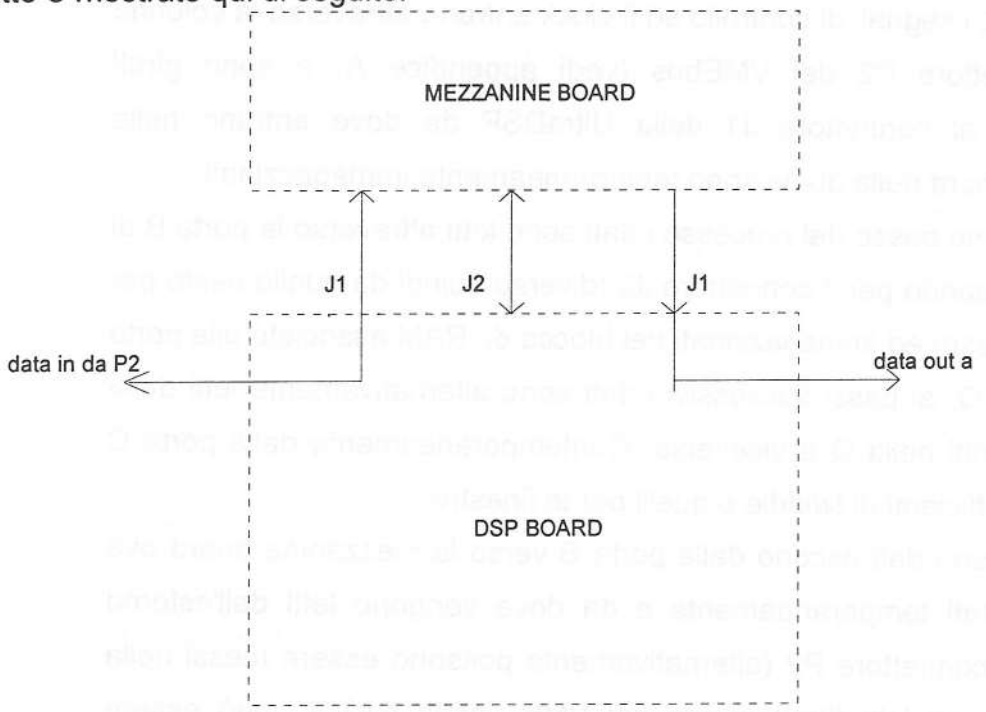


Figura A.10 connessioni della mezzanine board

Il bus controller, necessario per garantire una efficiente interfaccia fra la scheda ed il VMEbus è costituito da un microprocessore AT&T DSP32C che si trova sulla UltraDSP board ma che gestisce anche i segnali per il controllo del funzionamento della mezzanine board.
I dati ed i comandi che devono essere scambiati attraverso i connettori J1/P2 si possono riassumere in:

DI[0..23]	I	dati in ingresso dal connettore con la DSP board, provenienti a loro volta dal convertitore AD, sono dati reali in complemento a 2.
INFIFO_RST*	I	segnale di reset iniziale della FIFO dei dati in ingresso
INCLK	I	segnale di sincronizzazione per i dati in ingresso

INDAV*	I	qualificatore di dato valido deve essere fornito alla PAL di controllo della FIFO, proveniente dal DSP
INFIFO_OEN*	I	segnale di abilitazione alla lettura della FIFO da parte del DSP
INFIFO_RD	I	segnale di sincronizzazione della lettura sulla FIFO, è un clock a 40MHz
INPUT STATUS(0,1)	O	flag di abilitazione alla lettura della FIFO da parte del DSP, devono essere esaminati dal DSP stesso che se è il caso provvede a generare i segnali di lettura
BR[0..23] & BI[0..23]	I/0	dati da e verso la DSP board
OUTDAV*	I	presenza di un dato BR e BI valido proveniente dal DSP, indica anche al generatore di indirizzi che deve incrementare il suo valore durante l'averaging
READEN	I	segnale proveniente dalla DSP board che indica la volontà di questa di leggere il contenuto della RAM di accumulazione, serve al generatore di indirizzi per cominciare a generare indirizzi sequenziali
RAMSELECT	I	se è alto abilita la RAM di accumulazione per le operazioni di lettura e scrittura
OUTCLK	I	segnale di sincronizzazione dei dati in uscita

A.7.1.4.descrizione del funzionamento della mezzanine board

La scheda deve essere prima di tutto inizializzata con una operazione di reset provvista dalla DSP board col segnale (INFIFO_RST*), dopodiché i dati entrano nella scheda attraverso un registro di ingresso dove sono risincronizzati con il clock INCLK che per una buona utilizzazione della scheda deve essere uguale al segnale di clock della scheda che è di 40 MHz.

I dati sono campionati su un registro di ingresso da un segnale di INCLK e da qui possono venire scritti sulla FIFO vera e propria mediante un segnale proveniente da una unità di controllo (una PAL) che invia il segnale di scrittura all' una o all' altra FIFO alternativamente a seconda che l' indice del dato sia pari o dispari e solo se è andato basso un segnale di INDAV* ossia di data valid, proveniente dalla scheda DSP.

La scheda DSP per la lettura dei dati dalla FIFO deve andare a leggere il contenuto di 2 appositi INPUT STATUS flags che indicano appunto lo stato della FIFO e che possono assumere le seguenti configurazioni:

IS 0	IS 1	IS 2	significato
0	0	0	FIFO piena, contiene 2048 parole
0	0	1	FIFO vuota
0	1	0	FIFO quasi piena
0	1	1	FIFO quasi vuota

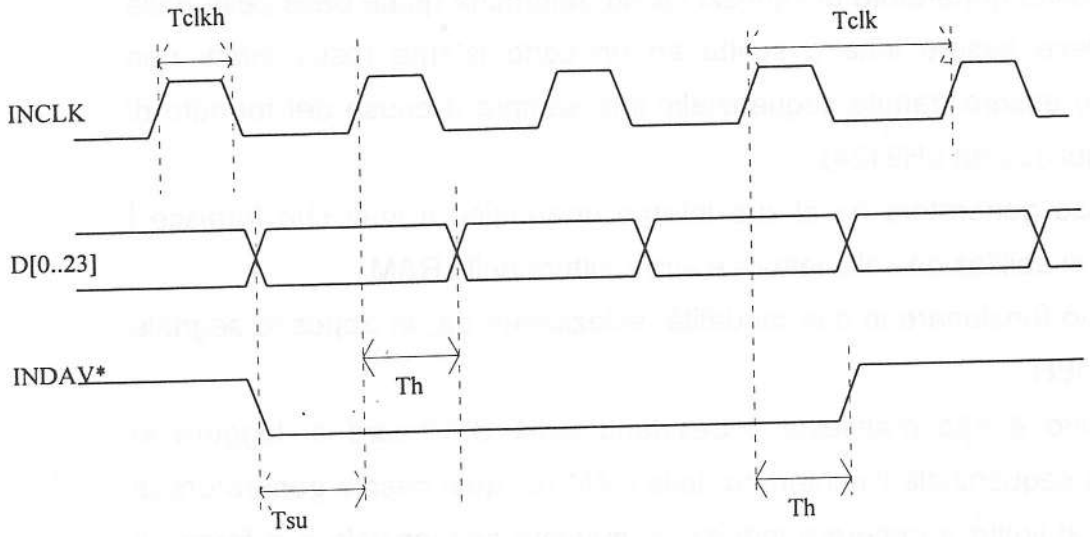
Tabella A.11 significato dei registri di status

Il numero di word che determinano la condizione di FIFO quasi piena o quasi vuota è configurabile all' installazione della FIFO stessa.

Quando ci sono dati disponibili sulla FIFO, ossia quando questa è piena il DSP può inviare i segnali di lettura del contenuto della FIFO stessa che sono INFIFO_OEN* cioè l'abilitazione alla lettura e INFIFO_RD che è il segnale di clock di lettura dalla FIFO stessa è che è di 40 MHz.

Per la lettura della FIFO bisognerà programmare

Le temporizzazioni dei segnali descritti si possono schematizzare con la seguente figura:



Tclkh(min)=11.1ns

Tclk(min)=27.8ns

Tsu=10ns

Th=5ns

Figura A.12 temporizzazioni segnali per mezzanine board

Per quanto riguarda i segnali necessari a comandare l'unità di averaging e la RAM di memorizzazione delle uscite a 56 bit le cose sono un po' più complicate. Il dato proveniente dall' elaborazione dello spettro di potenza deve essere memorizzato in uno dei registri dell' averager, che sono 8, tutti da 24 bit e controllati da una apposita PAL che riceve un segnale di OUTDAV* ossia di dato valido in uscita ed abilita due dei registri sui quali devono essere memorizzate le parti reali ed immaginarie dei dati in base ad un pattern ben preciso che dipende dal particolare modo in cui LH9124 presenta in uscita i risultati di una operazione di calcolo del modulo di un vettore complesso (CMAG).

La PAL in particolare avrà in ingresso le uscite di un contatore per 4 in quanto il pattern si ripete periodicamente ogni 4 dati acquisiti.

Le uscite di detti registri corrispondono agli ingressi del sommatore a 56 bit che calcola la somma fra dette uscite ed il contenuto della RAM di accumulazione.

Un apposito generatore di indirizzi locale determina quale delle celle della RAM deve essere letta o scritta ad un certo istante (esse infatti non possono essere trattate sequenzialmente sempre a causa del formato di uscita dei dati da LH9124).
Lo stesso generatore ha al suo interno una logica a stati che fornisce i segnali di abilitazione alla lettura e alla scrittura della RAM.
Esso può funzionare in due modalità, selezionate da un apposito segnale di READEN.
Se questo è alto manifesta il desiderio della DSPboard di leggere in maniera sequenziale il contenuto della RAM nel qual caso il generatore di indirizzi si limita a generare indirizzi in maniera sequenziale e a fornire il segnale di lettura del contenuto della RAM, che viene passato direttamente al connettore di interfacciamento col VMEbus.
Se invece READEN è basso siamo nella fase di accumulazione dei risultati e gli indirizzi sono generati secondo il pattern necessario per le operazioni di questo tipo.
Per la descrizione particolare dello schema dell' averager si fa ancora riferimento al capitolo sul calcolo degli spettri di potenza.
In pratica la RAM è strutturata in modo da funzionare con segnali analoghi a quelli di ciascuno dei blocchi di memoria associati a ciascuna delle porte di LH9124 sulla DSP board, con in più la possibilità di essere letta dal VMEbus.

A.7.1.5.l'unità di conversione per l'averager e la seconda UltraDSP

Serve ad interfacciare le uscite della UltraDSP che sono quelle di un LH9124, quindi in formato block floating point con gli ingressi del DSP32C, in formato floating point della seconda scheda UltraDSP che viene questa volta utilizzata per l'accumulazione ed il calcolo della media dei risultati del

calcolo dello spettro di potenza calcolati col metodo Bartlett-Welch in floating point (è evidente come questa seconda UltraDSP sia gravemente sottoutilizzata, in quanto praticamente tutti i chip della serie LHxx della Sharp che pure sono presenti non sono interessati al processo, eccetto che per la banale gestione della porta di acquisizione dei dati e dei blocchi di memoria in cui sono depositati i dati, elaborati dal solo DSP32C).
L'unità di interfaccia è costituita da un microprocessore che calcola il modulo dei dati in uscita dalla scheda UltraDSP e funge da convertitore dal formato in fixed point in uscita da LH9124 verso il formato in floating point che è necessario per eseguire le operazioni di media sul DSP32C dell' AT&T.

A.7.2. Risultati delle misurazioni eseguite sul sistema Valley

Per provare l'efficienza di un sistema reale di elaborazione di spettri si sono effettuate alcune prove di funzionamento sulla scheda installata presso i laboratori della stazione radioastronomica do Medicina, natura e scopo delle varie prove verranno esposte a mano a mano che ne verranno descritti i risultati.
Bisogna premettere che il valore dei risultati è purtroppo solamente indicativo delle prestazioni ottime ottenibili dal sistema Valley in quanto diversi fattori esterni non eliminabili ne influenzano negativamente la performance.
In particolare ci si riferisce al funzionamento dell'unità di immagazzinamento dei dati finali delle elaborazioni, costituita da una unità a nastro magnetico, che attualmente costituisce il vero collo di bottiglia del sistema, infatti essa è particolarmente lenta, tanto che mentre la scheda garantisce in teoria un' elaborazione dei dati in real-time, almeno per le rilevazioni astronomiche fatte attualmente dal laboratorio ed eseguite usando una frequenza di campionamento di 12-18 MHz con un numero di campioni di 8-16Ksamples, la reale performance del sistema non supera il rate del 25% (ossia solo un quarto dei dati in ingresso viene effettivamente

elaborato), inoltre per poterla utilizzare è necessario eseguire una conversione di formato dal floating point usato da DSP32C per l'averaging verso un altro sempre in floating point ma incompatibile con il primo, il che impone una complicazione all'hardware.

Per valutare le prestazioni della sola scheda UltraDSP, trascurando le influenze esterne si sono effettuate, come vedremo in dettaglio nel seguito, delle prove "a vuoto", ossia senza memorizzare i risultati ottenuti e utilizzando come dati in ingresso dei valori già in precedenza memorizzati su uno dei banchi di memoria della scheda, questo per escludere i tempi di acquisizione e di stoccaggio dei dati.

Inoltre i tempi sperimentali sono stati ottenuti facendo uso di un cronometro astronomico, la cui precisione è nota solamente al secondo, i decimi di secondo sono valutati solo approssimativamente, osservando la successione dei tempi di calcolo e notando che ad esempio per un tempo di 1 secondo e 1/10 ogni dieci spettri l'incremento del tempo astronomico è di 2 sec anziché uno solo.

A.7.2.1.Stima dei tempi di calcolo globali del sistema

Lo scopo delle misurazioni che abbiamo effettuato sul sistema è quello di valutare i tempi globali di elaborazione dei dati provenienti dal campionatore e quindi di individuare all' interno di questi le varie componenti dovute alle singole fasi di elaborazione che sono state descritte qui sopra.

Il software necessario per gestire il programma di calcolo in gran parte era già installato sul sistema ma talvolta è stato necessario riscriverlo per poter individuare e considerare separatamente ciascuna fase del processo di elaborazione.

Cominciamo a vedere i risultati globali relativi all' intero processo, calcolati per varie lunghezze dei record e per varie frequenze di campionamento. In pratica le misurazioni sono state effettuate integrando 100 osservazioni successive allo scopo di ridurre l'effetto del rumore secondo il metodo di Bartelett-Welch visto nel capitolo relativo agli spettri di potenza, la tabella riporta i tempi rilevati e la struttura della FFT usata per il calcolo :

	struttura della FFT	6MHz	12MHz	18MHz	32MHz
4K	VMUL,R2,R16,R16,R4,BFCT	1.1s	1.1s	1.1s	1.1s
32K	VMUL,R4,R16,R16,R16,BFCT	2.7s	1.8s	1.7s	1.7s
128K	VMUL,R16,R16,R16,R16,BFCT	7.3s	5.5s	4.0s	3.2s
256K	VMUL,R2,R16,R16,R16,R16,BF CT	14.0s	9.5s	8.2s	6.1s

Tabella A.7 sommario dei tempi di calcolo rilevati sperimentalmente sul sistema Valley

Le misurazioni successive hanno lo scopo di valutare i tempi relativi a ciascun passo del processo cioè di scomporre i tempi della tabella nei loro componenti elementari.

Preliminarmente è utile valutare in modo deduttivo i tempi necessari per il solo calcolo della trasformata, escludendo quindi le fasi di acquisizione e di

scrittura dei risultati ed anche i tempi di programmazione dei registri di LH9124 e di LH9320. Tali tempi si calcolano semplicemente secondo la formula:

$$(\text{Numero dei canali} + \text{tempo di latenza}) \times \text{periodo di clock} \times \text{numero di passi}$$

ove il numero di canali indica la dimensione del record di dati , il tempo di latenza varia a seconda del passo di programma, il periodo di clock è di 0.025μs.

I tempi così ottenuti sono riassunti nella seguente tabella, nella quale sono anche indicati gli algoritmi impiegati, che sono quelli ottimi, ossia quelli che eseguono il calcolo nel minor numero di passi:

	struttura della FFT	
4K	VMUL,R2,R16,R16,R4,BFCT	0.064s
32K	VMUL,R4,R16,R16,R16,BFCT	0.49s
128K	VMUL,R16,R16,R16,R16,BFCT	1.97s
256K	VMUL,R2,R16,R16,R16,R16,BFCT	4.59us

Tabella A.8 tempi di calcolo della trasformata con l'algoritmo indicato, sono trascurati i tempi necessari alla programmazione dei registri

Il calcolo dei tempi di programmazione e la valutazione degli effetti che essi hanno sui tempi globali del sistema è eseguibile in modo indiretto ripetendo le operazioni necessarie per il calcolo della FFT (compresa quindi la programmazione di tutti i registri interni necessari) per un certo numero di volte (ad es. 10000), in modo da rendere i tempi rilevabili, e quindi sottraendo i tempi dedicati al puro calcolo.

Per eseguire l'operazione si è dovuto impiegare il seguente programma, scritto in linguaggio C, che fa uso di alcune macro di libreria fornite assieme alla scheda dalla stessa Valley e per il cui significato si rimanda sia all'appendice B per quanto riguarda i principi di programmazione, sia ai manuali della UltraDSP per una descrizione più generale:

```
/*****************************************************************
```

```
*nome del file: esempi                                                    *
*descrizione: il programma C serve a verificare il tempo necessario a
programmare i                                                             *
*dispositivi della scheda UltraDSP, ciò è fatto  ripetendo in un loop
sufficentemente                                                           *
*lungo l'esecuzione di una FFT, senza considerare l'operazione di
acquisizione dei dati                                                     *
*misurando il tempo impiegato e quindi sottraendo il tempo di calcolo veo e
proprio                                                                   *
*Come esempio si è adottato il calcolo di una trasformata di 16K reali
mediante una                                                              *
*complessa di 8
*************************************************************************/

/*include files*/
include "lh9124.h"   /*definizione delle caratteristiche del DSP*/
include"lh9320.h"          /*definizione del generatore di indirizzi*/
include CLANGUAGE 1    /*abilita le macro C definite in ultradsp.h*/
include"ultradsp.h"  /*definizioni e macro*/
includehostio.h"           /*abilita l'accesso alle funzioni di 'stdio' usando
'hostio'*/

define K 8*1024                  /*definisce la lunghezza della trasformata
complessa*/
define TWSIZE   16*1024   /*definisce la lunghezza della memoria dei
coefficenti*/
define LOOP 10000          /*definsce il numero di cicli di programmazione*/

/*definizioni locali*/
static struct tagMemAddr AddrGen;

/*dati external*/
extern int dfdata          /*copia software del registro di dataflow*/
extern int fcdata          /*copia software del registro di function code*/
/*****MAIN*****/
void main()
register int k;            /*indici dei loop*/
unsigned int scale,max;     /*fattori di scala  dei dati*/
for (k=1;k<LOOP;k++)     /*inizio dei cicli di programmazione*/
{load_ag_int(AGMEMSIZE,TWSIZE);/*pone  AGMEMSIZE  a  16K,  la
dimensione dei coefficenti*/
load_ag_int(NUMSAMPL,K);                  /*pone NUMSAMPL ad 8K,la
dimensione della lunghezza della FFT*/
fft8k(0,scale);                  /*esegue la FFT da 8K e pone i risultati in
RAMA*/
```

```
set_ag_byte(LATENCY00,0x848518);   /*MOVE dei risultati della FFT in
RAMQ*/
set_ag3(AG_PROG_MEM00,BCTL,BFCTT,BFCTUS);/*programma      i
registri di AG per eseguire il passo di ricombinazione*/
set_df(BFCT);              /*setta il function code register*/
set_df(RAMQ|AGA|DSPA)/*programma i registri di data-flow per trasferire i
dati da A a Q*/
wait_till_done}
```

I tempi ottenuti in questo modo non sono necessariamente i migliori possibili in quanto utilizzano delle macro fornite assieme alla scheda dalla Valley, ma non dovrebbero discostarsi troppo da questi.

Nel nostro caso la durata del calcolo per 10000 cicli è stata di 8.56sec per la FFT più 4.2sec per il passo di ricombinazione , sottraendo il tempo di calcolo della FFT che è di 4*8*1024*10000*0.0025μs=8.192s, si può approssimare il tempo di programmazione di ognuno dei 4 passi dell' algoritmo in (8.56-8.192)/(4*10000)=**9.2μs**, mentre per il passo di ricombinazione il tempo di programmazione è di 5.5μs.

Sono altresì stati rilevati i seguenti tempi di esecuzione della sola FFT (senza passo di ricombinazione):

FFT	struttura	tempo di esec.
fft1k()	16,16,4	105μs
fft2k()	16,16,4,2	240μs
fft4k()	16,16,16	335μs
fft8k	16,16,16,2	855μs
fft16K	16,16,16,4	1.7ms
fft32k	16,16,16,4,2	1.1ms
fft64k	16,16,16,16	6.6ms
fft128k	16,16,16,16,2	16.4ms

Tabella A.9 tempi di esecuzione delle FFT usando le macro di libreria

Interessante è poi calcolare quali sono le influenze sul tempo di elaborazione imputabili alla semplice acquisizione dei dati alle varie frequenze di campionamento considerate, lo scopo di tale misura è di valutare l'efficienza del buffer di interfaccia che è fortemente sospettato di essere inadeguato, disponendo di una sola FIFO per il deposito dei dati ed obbligando quindi il sistema a delle pause per il caricamento dei dati. Detto N il numero di campioni ed f la frequenza di campionamento il tempo di campionamento del record da analizzare sarà dato semplicemente da N/f, quindi:

	6MHz	12MHz	18MHz	32MHz
4K	682us	341us	227us	113us
32K	5.46ms	2.7ms	1.8ms	910us
128K	21.8ms	10.9ms	7.2ms	3.6ms
256K	43.6ms	21.8ms	14.5ms	7.2ms

Tabella A.10 tempi di riempimento della FIFO da parte del convertitore analogico/digitale, alle varie frequenze di esercizio

Come è logico il time record aumenta a mano a mano che la frequenza di campionamento diminuisce, per giustificare i tempi di calcolo rilevati sperimentalmente bisogna considerare che il buffer di acquisizione è costituito da una semplice FIFO che viene letta a blocchi di 1K alla volta, utilizzando i segnali provenienti da un generatore di indirizzi LH9320 che deve essere riprogrammato ad ogni blocco letto, inoltre bisogna aspettare che la FIFO sia piena e che quindi venga inviato un apposito segnale al DSP32C, che a sua volta provvederà a generare gli opportuni segnali di lettura per il calcolo dei tempi di programmazione dei passi di acquisizione dei dati dalla FIFO si è fatto ricorso al seguente programma, che ripete per 10000 volte la sequenza di operazioni necessaria:

```
/*************************************************************************
*nome: esempio
*
*descrizione:il programma serve a valutare i tempi necesari a
programmare le FIFO di ingresso*
```

```
*dei dati,                                                              *
*ciò e fatto eseguendo  per un numero sufficente di volte un loop di
acquisizione dei dati e     *
*quindi sottraendo tempo impiegato per il trasferimento vero e proprio
                                                                        *
************************************************************************/
/*include files*/
include "lh9124.h"          /*definizione delle caratteristiche del DSP*/
include"lh9320.h"           /*definizione del generatore di indirizzi*/
include CLANGUAGE 1     /*abilita le macro C definite in ultradsp.h*/
include"ultradsp.h"         /*definizioni e macro*/
include"hostio.h"           /*abilita l'accesso alle funzioni di 'stdio' usando
'hostio'*/

define K 8*1024                 /*definisce la lunghezza della trasformata
complessa*/
define TWSIZE  16*1024      /*definisce la lunghezza della memoria dei
coefficenti*/
define LOOP 10000                   /*definsce il numero di cicli di
programmazione*/

/*definizioni locali*/
static struct tagMemAddr AddrGen;
unsigned int *streg=(unsigned int *)ST_REGISTER ; /*puntatore allo status
register*/
unsigned int *dfreg=(unsigned int *)DF_REGISTER ; /*puntatore allo DF
register*/
unsigned int *fcreg=(unsigned int *)FC_REGISTER ; /*puntatore allo FC
register*/

/*dati external*/
extern int dfdata           /*copia software del registro di dataflow*/
extern int fcdata           /*copia software del registro di function code*/

/*****MAIN*****/
void main()
FILE *fp;                 /* puntatore al file di uscita dei risultati*/
register int *Rptr, *Iptr;   /*puntatore ai dati reali ed immaginari*/
float real1,imag1;              /*dati reali ed immaginari in formato floating
point, per DSP32C*/
register int i,j,k;          /*indici dei loop*/
unsigned int scale,max;     /*fattori di scala  dei dati*/
for(k=1;k<LOOP;k++);        /*inizio loop di ripetizione della procedura di
acquisizione dati*/

/*definisci gli indirizzi di partrenza*/
```

```
AddrGen.A=0;
/*indirizzo di base del generatore A */
AddrGen.B=0;
/*indirizzo di base del generatore B */
AddrGen.C=0;
/*indirizzo di base del generatore C */

load_ag_int(ADRLENGHT,1024);          /*lunghezza del buffer FIFO
di input*/

                                      /*reset delle FIFO di input*/
i=dfdata & 0x8FFF;                    /*copia software del registro
DF*/                                  ;
*dfreg=i | 0x6000
/*reset della FIFO di input*/
*dfreg = i;                           /*abilita le FIFO*/
fcdata = fcdata | 2;                  /*setta il bit di MISC_OUT*/
*fcreg=fcdata;                        /*immagazzinalo nel registro
di function code*/

max=0;
/*inizializza 'max'*/
scale =0
/*inizializza 'scale'*/

{for(i=0;i<8;i++)                     /*acquisisci 8 blocchi da 1K
ciascuno dalla FIFO*/
load_ag_address(&AddrGen);            /*setta l'indirizzo di partenza
nei genartori di indirizzi*/
while(*streg & 2)                     /*attendi fino a che non diventa
attivo il flag di FIFO piena*/
move_bufferBtoA(MOVD | 2);            /*move di un blocco da 1K
nella RAMA, e settaggio del bit di MISC_OUT per caricare i successivi 8K
bit*/

/calcolo della massimo fattore di scala e suo invio alla routine di FFT per il
calcolo del fattore di scala iniziale*/
max=/*streg & 0xE000);                /*acquisizione dei bit di
DSFO e selezione del bit DSISEL(vedi maunali di LH9124)*/
if(scale<max) scale = max;            /*increment2o dell' indirizzo
dei blocchi A e Q per il passo successivo*/
AddrGen.A+=1024;
AddrGen.Q+=1024;
/*fine dell'acquisizione di 8 blocchi da 1K*/}
/*fine di un loop*/}
```

In questo caso il tempo rilevato è stato di circa 2.4s per acquisire 10000 volte 8 blocchi di FIFO da 1K, poichè il tempo che il sistema impiega a leggere 10000 blocchi da 1K senza considerare la programmazione è di 8*1024*10000*0.025μs=2.05s si può calcolare per la programmazione di ogni blocco da 1K un tempo approssimativo di (2.4μs-2.04μs)/(8*10000)=4.5μs. La tabella indica i tempi necessari per riempire la FIFO da 1K alle varie frequenze, si rileva che con questo tipo di interfaccia il massimo transfer rate possibile è quello di 34Msps , infatti la lettura di un blocco da 1024 word prende:

(1024·0.025+4.5)μs=30.1μs

quindi una frequenza di campionamento superiore ai 34MHz provocherebbe il riempimento della FIFO prima che sia completata la lettura, con conseguente perdita di dati.

6MHz	170.6us
12MHz	85.3us
18MHz	56.8us
32MHz	32.4us

Tabella A.11 durata della finestra necessaria a campionare 1Kbyte alle varie frequenze

la tabella dei tempi di scrittura per 100 sequenze risulta dunque dal prodotto dei tempi di cui alla tabella qui sopra per il numero di K del record:

	6MHz	12MHz	18MHz	32MHz
4K	0.068s	0.034s	0.022s	0.011s
32K	0.54s	0.27s	0.182s	0.091s
128K	2.18s	1.09s	0.72s	0.36s
256K	4.37s	2.18s	1.45s	0.72s

Tabella A.12 durata della finestra necessaria a campionare il record di ingresso per varie frequenze e per varie lunghezze del record stesso

In pratica mentre la FIFO continua a caricare dati senza mai fermarsi la scheda UltraDSP non va a leggere prima di aver terminato il calcolo della trasformata, per poterlo fare naturalmente deve attendere che sia terminata la sequenza che si stava scrivendo all' istante in cui è terminato il calcolo, in effetti ciò introduce un tempo morto fra la fine dell'elaborazione da parte della UltraDSP e l'inizio del caricamento della successiva sequenza calcolabile.
A questo si aggiunge il fatto che ogni certo numero di medie (nel nostro esempio 100) bisogna scrivere sull' unità a nastro il risultato dell' integrazione che è una operazione come abbiamo visto molto lenta (si deve anche eseguire una trasformazione di formato).
Per stimare il tempo necessario all' operazione si sono eseguite delle trasformate di record molto corti (2K) e senza integrare su più spettri, ossia senza eseguire l' averaging.
I risultati ottenuti sono riassunti nella seguente tabella :

	12MHz	18MHz
2K	≈1 sec	≈1 sec
32K	≈1.2sec	≈1.15 sec

Tabella A.13 tempi di calcolo e memorizzazione dei risultati, senza averaging

come si vede il tempo di accesso all' unità a nastro è di circa 1 sec., trascurando i tempi di calcolo vero e proprio della trasformata.
In sostanza l' accesso all' unità di memorizzazione dei risultati blocca per circa 1 sec. l'elaborazione. Inoltre dalla scomposizione dei tempi di esecuzione per un record di 32K si deduce che il transfer rate è dell' ordine delle poche centinaia di Kbyte al secondo, il che aggrava ulteriormente i tempi per record di dimensioni elevate.
Nelle nostre osservazioni per semplicità considereremo sempre dominate il tempo di un secondo necessario per il primo accesso.
A questo punto giustificare i tempi sperimentali della prima tabella diventa semplice, basta verificare che essi sono uguali al minimo multiplo del tempo necessario all' acquisizione dell' intero record , in grado di

contenere il tempo di calcolo della FFT, il tutto sommato ancora al tempo di acquisizione di ogni record ed al tempo necessario per scrivere i risultati sull' unità a nastro.

In pratica l'espressione che consente di stimare i tempi di elaborazione e di confrontarlo con i risultati sperimentali è:

$$\left(\text{int}\frac{tempo\_calcolo\_FFT}{tempo\_acquisiziuone\_dati}+1\right)+tempo\_acquisizione\_dati\cdot M+tempo\_nastro$$

in cui int indica l'approssimazione all' intero superiore della divisione e M il numero di elaborazioni da fare per ottenere la media (nel nostro caso, come si è detto,100).

Il ragionamento diventa ancora più semplice se lo si fa graficamente: prendiamo ad esempio il calcolo della trasformata di 256Kpunti campionati a 18MHz; ricordiamo che il tempo di calcolo è in questo caso di 4.59 sec (per 100 record) ed il tempo di acquisizione di 1.45 sec:

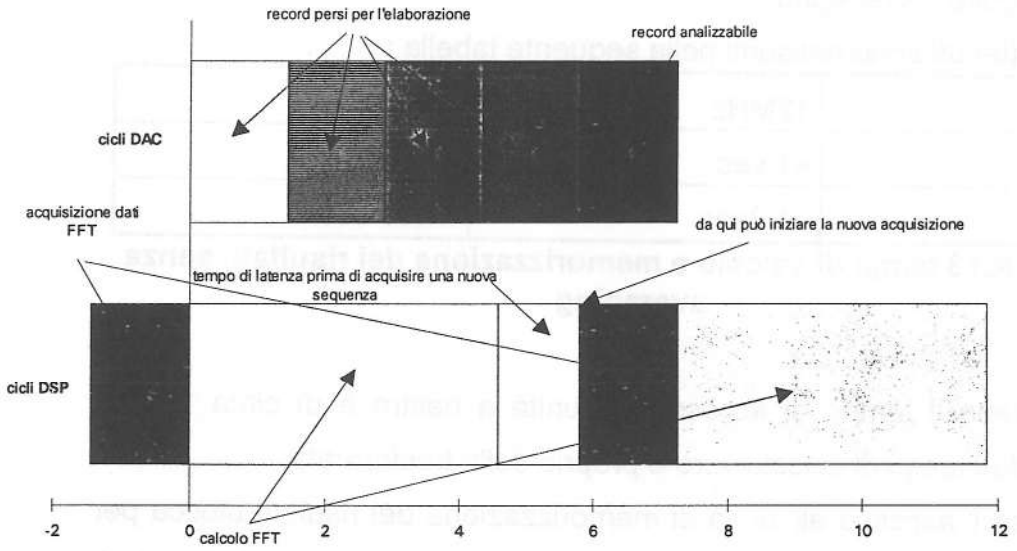


Figura A.13 visualizzazione dei tempi di calcolo per l'acquisizione e per l'elaborazione dei dati

in pratica come si vede nel tempo necessario per eseguire la trasformata il campionatore emette ben 4 record completi, solo al termine delle quali si può iniziare l'acquisizione di una nuova. ciò significa che solamente una sequenza su 5 riesce ad essere elaborata.

Passando ai termini numerici il tempo rilevato dalle prove sperimentali si giustifica osservando che una intera elaborazione prende 5 volte il tempo

necessario a campionare tutto il record, cioè 4 per attendere che UltraDSP sia di nuovo libera ed 1 per il caricamento della sequenza vero e proprio, cioè appunto 1.45 sec × 5 = 7.25, a cui bisogna sommare il tempo di scrittura sul nastro, di circa 1 **sec**, ottenendo 8,25, vicino quindi al valore di 8,2 sec rilevato.

Un discorso analogo vale campionando a 32 MHz, in questo caso il tempo necessario all' acquisizione di 100 sequenze è di 0,72 sec, quindi si fa in tempo a campionare int(4.58/0,72)=7 record completi, il tempo sperimentale è quindi dato da 7 volte il tempo di campionamento più una per il caricamento della sequenza utile, cioè 0,72 sec × 8 + 1 = 6.76 confrontabile con i 6.4 sec osservati sperimentalmente (efficienza di 1/5).

Campionando a 12MHz il tempo per avere un 100 record completi è 2.18, quindi int(4.58/2.18) = 3, più uno per l'acquisizione da 4 (efficienza di 1/4), con un tempo globale di 4×2.18 sec +1 = 9.72, confrontabile con i 9.5 sec approssimativamente calcolati al laboratorio.

Nel caso di campionamento a 6MHz (int(4,58/4.37)+1)·4.37+1= 14.11 a fronte di un 14 sec misurato.

se le dimensioni del record sono di 128K punti:

32MHz : (int(1.96/0.36)+1)*.36 sec + 1 sec = 3.52 rilevato:3.2

18MHz: (int(1.96/.72)+1)*0.36 sec + 1 sec = 3.88 rilevato:4.0

12MHz : (int(1.96/1.09)+1)*1.09 sec + 1 sec = 4.27 rilevati:5.5

6MHz : (int(1.96/2.18)+1)*2.18 sec +1 sec = 7.54 rilevati:7.3

lo stesso dicasi per record da 32K punti:

32MHz : (int(0.6/0.91)+1)*0.091+1=1.72 sec rilevati:1.7

18MHz : (int(0.6/0.18)+1)*0.18 +1 =1.9 sec rilevati:1.8

12MHz :(int(0.6/0.27)+1)*0.27 +1 = 2.08 sec rilevati:1.8

6MHz: (int(0.6/0.54)+1)*.54 +1 = 2.62 sec rilevati:2.7

Il calcolo cade in difetto nel caso di dimensioni del record molto piccole come 4Kpunti per i quali il tempo necessario all' elaborazione diventa trascurabile rispetto a quello impiegato per la memorizzazione su nastro magnetico, fisso per tutte le frequenze di campionamento e di circa 1 sec (stima ottenuta dai dati sperimentali)

A.7.3.considerazioni sul funzionamento in real time della scheda UltraDSP

Nel ragionamento che segue escluderemo la considerazione del tempo necessario alla memorizzazione dei risultati sull' unità a nastro in quanto non è influente ai fini del ragionamento sul funzionamento in tempo reale della scheda.

E' evidente che per avere un funzionamento in real time ossia senza perdita di campioni il calcolo della trasformata deve essere completato prima che si riempia la FIFO ossia prima che si campionino 1024 valori.

E' questa una grave limitazione che nell' attuale configurazione del sistema riserva il funzionamento in real time a frequenze di campionamento estremamente basse.

Riportiamo infatti la tabella dei tempi necessari al solo calcolo della trasformata per varie lunghezze del record, relativi alla migliore struttura possibile per il calcolo della FFT su record di quella lunghezza:

FFT	struttura	tempo di esec.
fft1k()	16,16,4	105μs
fft2k()	16,16,4,2	240μs
fft4k()	16,16,16	335μs
fft8k	16,16,16,2	855μs
fft16K	16,16,16,4	1.7ms
fft32k	16,16,16,4,2	1.1ms
fft64k	16,16,16,16	6.6ms
fft128k	16,16,16,16,2	16.4ms

La frequenza massima possibile per garantire il real time è data da:

$$f_{\max} = \frac{1024}{T_{FFT}}$$

in cui T_{FFT} è naturalmente il tempo necessario al calcolo della FFT

quindi le frequenze di funzionamento del convertitore A/D sono date da:

FFT	f_{max}
fft1k()	9.7MHz
fft2k()	4.2MHz
fft4k()	3.05MHz
fft8k	1.19MHz
fft16K	602KHz
fft32k	330.3KHz
fft64k	155Khz
fft128k	62.43Khz

Tabella A.14 massimo rate di campionamento per vere funzionamento in real time

Considerando che la banda massima del segnale per il teorema di Shannon non può essere superiore alla metà di tali frequenze si vede come non sia in generale possibile impiegare il sistema (almeno nella attuale configurazione) per il calcolo in real time quando il segnale abbia frequenze appena un po' elevate.

A.7.4. Considerazioni sull'architettura del sistema Valley

Dalle precedenti misurazioni si possono trarre alcune considerazioni sull' efficienza della struttura.

Innanzitutto si individuano due principali colli di bottiglia che sono il sistema di acquisizione dei dati provenienti dal convertitore A/D e quello di immagazzinamento dei risultati nello streamer, posti rispettivamente all' inizio ed alla fine dell' elaborazione.

Il primo collo è generato dall' impiego di una FIFO come buffer dei dati in ingresso, questa deve essere letta a blocchi di 1K alla volta con l'ulteriore complicazione che il generatore di indirizzi LH9320 non dispone di un flag di arresto del conteggio in corrispondenza della condizione di FIFO vuota e quindi ad ogni blocco occorre riprogrammare il generatore in modo da

fargli generare i corretti segnali per la lettura del blocco. Ciò significa che è necessario eseguire un numero di riprogrammazioni pari al numero di blocchi da 1024 punti da elaborare, ed ogni riprogrammazione prende evidentemente un certo tempo che in generale dipende dal numero di parametri che occorre riprogrammare, ma che, utilizzando le macro messe a disposizione dalla Valley ed usate nelle nostre rilevazioni sperimentali, si può calcolare di circa 10.3μs per ogni blocco. Fortunatamente , come si è messo in evidenza, tali riprogrammazioni non influiscono sul tempo di acquisizione dei dati, che è determinato dalla frequenza di campionamento, ma impongono solamente un limite superiore a detta frequenza, che in effetti si è visto non possa essere superiore ai 34MHz (il convertitore VE-ADC-2110 arriva fino ad una frequenza di campionamento di 40MHz).

Questo avviene perché la FIFO è di 2Kbyte, quindi mentre il Kbyte inferiore viene riempito (ad una velocità dipendente dal periodo del clock di campionamento) quello superiore viene letto alla frequenza fissa di 40 MHz, che è la frequenza di LH9124. Quindi la riprogrammazione può essere effettuata nell'intervallo si tempo fra la fine della lettura del blocco della parte superiore e la fine della scrittura di quello inferiore, che deve quindi essere fatta a velocità inferiore.

Infine l'uso di un interfaccia come la mezzanine board type-1 obbliga a trasferire, prima di iniziare l'elaborazione vera e propria, i dati in uno dei blocchi di memoria associati alle porte A e Q (si ricorda che nella UltraDSP i dati entrano dalla porta B), il che equivale all'aggiunta di un passo al programma per la sola operazione di caricamento dei dati.

Si osserva poi che il sistema di streaming dei dati in uscita anche se è utilizzato solo alla fine di ogni integrazione, ossia solo dopo che si sono eseguite un certo numero di medie di spettri successivi dello stesso segnale, impone un forte rallentamento alle operazioni a causa della sua lentezza. Infatti mentre esso funziona blocca il funzionamento del resto del sistema, quindi costituisce un tempo che si somma a quello globale (non lo si può far funzionare a pipeline perché va a leggere da una memoria che verrebbe alterata da una successiva elaborazione).

Le ragioni di tale lentezza sono ancora sostanzialmente architetturali, in particolare risiedono nell'uso di una FIFO anche per l'uscita dei dati prelevati da uno dei blocchi della memoria "di servizio" (ossia o la A-RAM o la Q-RAM) di LH9124, blocchi il cui contenuto verrebbe corrotto se si eseguisse il calcolo di una nuova trasformata.

Secondariamente sono dovute, come si è già osservato, all' impiego di una unità di backup dei dati a nastro magnetico, quindi alquanto lenta, e che oltretutto richiede la conversione del formato floating point rispetto a quello usato da DSP32C per il calcolo della media degli spettri (con conseguente complicazione della struttura del sistema di calcolo).

Si sono dunque calcolati i seguenti throughput effettivi (il throughput si calcola dal rapporto fra le dimensioni del blocco ed il tempo impiegato per la sua elaborazione):

$$f_s = \frac{\text{dimensioni record}}{\text{tempo di elaborazione}}.$$

dimensione campione	tempo di elaborazione	throughput
4K·100	1.1	0.37 Msps
32K·100	1.7	1.92 Msps
128K·100	2.9	4.5 Msps
256K·100	6.1	4.3 Msps

Tabella A.15 throughput misurati sul sistema Valley

Nel calcolo si è supposto di integrare i risultati di 100 FFT su campioni rumorosi, per applicare il metodo di Bartelett-Welch.

Il throughput è un parametro importante perché è uguale alla massima frequenza a cui possono essere campionati i dati in ingresso per avere elaborazione in real-time ovverosia senza perdita di campioni.

Interessante, per avere un'idea delle influenze dell' ambiente sulle prestazioni della UtraDSP, è confrontare questi risultati con quelli stimabili, ossia sulle prestazioni ottime della UltraDSP e con quelli rilevati sperimentalmente per il calcolo di una sola FFT, senza quindi eseguire la

media di Bartlett-Welch degli spettri di potenza e senza immagazzinare i risultati.

Per calcolare i throughput teorici si eseguono i seguenti passi:

1. Si scrive la struttura dell' algoritmo divisa in passi
2. Si calcola il tempo di processo per ciascun passo
3. Si sommano i tempi
4. Si divide la dimensione del blocco per il tempo del processo.

Poiché i dati campionati sono reali si è utilizzato l'algoritmo di calcolo della trasformata reale di 2N punti con una complessa di N punti, seguita da un passo di ricombinazione, inoltre per semplificare la programmazione delle operazioni si è fatto uso delle macro della libreria messa a disposizione dalla stessa Valley.

Cominciamo dal calcolo per trasformate da 4K punti reali eseguite con una da 2K punti immaginaria:

passo	descrizione	tempo di processo
1	2 passi di move	2×(1024×0.025+10.3)=72μs
2	FFT2K	241μs
3	BFCT (ricombinazione)	2048x0.025+5.5=56.7μs
4	2 passi di move	2×2048x0.025+10.3=72μs
		tot=441.7μs

Tabella A.16 tempi di esecuzione di una trasformata da 2K punti

quindi il throughput teorico massimo risulta:

$$f_s = \frac{\text{dimensioni blocco}}{\text{tempo di calcolo}} = \frac{4096}{0.0004417} = 9.27 \times 10^6 \,\text{samples/secondo}$$

per record di 32K invece :

passo	descrizione	tempo di processo
1	16 passi di move	16×(1024×0.025+10.3)=574.4μs
2	FFT16K	1.67ms

3	BFCT (ricombinazione)	16×1024×0.025+5.5=412μs
4	16 passi di move	16×(1024×0.025+10.3)=574.4μs
		tot=3.23ms

Tabella 2. tempi di esecuzione di una trasformata da 32Kpunti

troughput:

$$f_s = \frac{32768}{0.00323} = 10.1\,Msps$$

record di 128K:

passo	descrizione	tempo di processo
1	64 passi di move	64×(1024×0.025+10.3)=2297μs
2	FFT64K()	6.59ms
3	BFCT (ricombinazione)	64×1024×0.025+5.5=1644μs
4	64 passi di move	64×(1024×0.025+10.3)=2297μs
		tot=12.8ms

Tabella A.17 tempi di esecuzione di una trasformata da 128Kpunti

troughput:

$$f_s = \frac{131072}{0.0128} = 10.02\,Msps$$

ed infine per record di 256K:

passo	descrizione	tempo di processo
1	128 passi di move	128×(1024×0.025+10.3)=4695μs
2	FFT128K()	16.43ms
3	BFCT(ricombinazione)	128×1024×0.025+5.5=3282μs
4	128 passi di move	128×(1024×0.025+10.3)=4695μs
		tot=29.1ms

Tabella A.18 tempi di esecuzine di una trasformata da 256Kpunti

troughput:

$$f_s = \frac{262144}{0.029} = 9\,Msps$$

come si vede i throughput sono molto superiori rispetto a quelli effettivamente misurati per le stesse dimensioni dei record, ciò è imputabile essenzialmente ai già visti problemi di interfacciamento del sistema col resto del mondo, più che ad una effettiva inefficienza della scheda.

A.7.5 Panorama delle possibili modifiche dell' architettura per risolvere alcuni dei problemi incontrati

Vengono ora proposte delle soluzioni per ottimizzare le prestazioni del sistema Valley. Questo argomento, tra l'altro, era gia` stato affrontato in un' altra sezione di questa nota tecnica.

A.7.5.1.modifiche alla mezzanine board

Qui si vuole solamente rilevare i vantaggi che i cambiamenti gia` accennati in precedenza implicano dal punto di vista delle prestazioni.
Visti i problemi che comporta l'uso della FIFO un bel vantaggio si avrebbe nel sostituirla con una RAM statica, di dimensioni sufficienti a contenere tutto il record da elaborare (cioè nel caso della UltraDSP di 4×128Kword complessi, nel caso della nostra scheda di 1 milione di words).
Una mezzanine board siffatta per la verità e già stata annunciata dalla Valley, ma non ancora messa in commercio, anche se la stessa Valley fornisce i seguenti test di benchmarks calcolati sempre con record da 2N punti reali elaborati mediante l'uso di trasformate da N punti complessi :

dimensioni	struttura	throughput(Msps)	tempo
1K	VMUL,R2,R16,R16,BFCT	11.6 Msps	89µs
2K	VMUL,R4,R16,R16,BFCT	13.4 Msps	153µs
4K	VMUL,R2,R16,R16,R4,BFCT	12.2 Msps	336µs
8K	VMUL,R16,R16,R16,BFCT	15.2 Msps	538µs
16K	VMUL,R2,R16,R16,R16,BFCT	13.0 Msps	1259µs
32K	VMUL,R4,R16,R16,R16,BFCT	13.2 Msps	2488µs
64K	VMUL,R2,R16,R16,R16,R4,BFCT	11.4 Msps	5769µs
128K	VMUL,R16,R16,R16,R16,BFCT	13.2 Msps	9862µs

256K	VMUL,R2,R16,R16,R16,R16,BFCT	11.4 Msps	22974µs

Il vantaggio principale è che in pratica si eliminano i costosi passi di acquisizione dei dati con grande beneficio per il throughput, inoltre poiché i dati possono essere letti dalla RAM già in ordine invertito si ha il risparmio di un passo di programma rispetto all' implementazione con la FIFO.
Ciò è possibile perché in questo caso quando si è terminata l'elaborazione di un record quello successivo è già presente nella RAM di ingresso.
Il diagramma delle temporizzazioni che si era visto per un record da 256K punti campionato alla frequenza di18KHz diventa quindi in questo caso:

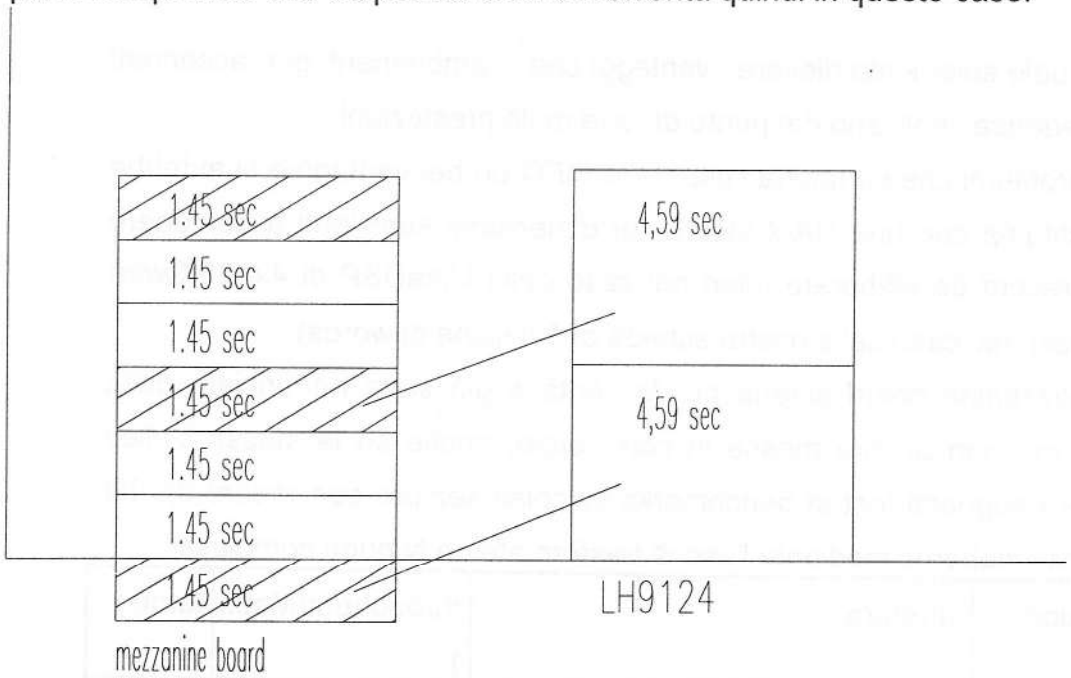


Figura A.14 visualizzazione dei tempi per il riempimento della RAM e per il calcolo della FFT

I valori rappresentati si riferiscono rispettivamente al tempo necessario a campionare il record a 18MHz (Mezzanine board) e ad elaborarlo per trovarne lo spettro (LH9124).
Come si vede non c'è perdita di tempo nell' acquisizione dei dati, quindi LH9124 non deve attendere che venga generato un nuovo record fra una elaborazione e l'altra, le frecce mostrano quali sono i record che vengono analizzati, gli altri vanno persi nell' attesa.

In pratica il tempo di calcolo è solamente quello strettamente necessario per il calcolo della trasformata.
Altri notevoli vantaggi si hanno nell' impiego di RAM statiche anche nell' uscita dei dati. Vengono in particolare semplificate le operazioni di lettura dei dati di uscita da parte della sezione di elaborazione che segue.
Si è visto come una delle cause che formano il collo di bottiglia all' uscita sia l'impossibilità di proseguire nell'elaborazione degli spettri mentre si stanno scrivendo i risultati sullo streamer a nastro. La ragione di ciò è che i dati scritti sono prelevati da uno dei blocchi di RAM usati da LH9124 per scrivere i risultati intermedi e che quindi non possono essere corrotti, come accadrebbe se durante la lettura si eseguisse il calcolo di una altra trasformata.
Usando invece la RAM in uscita i risultati finali dell'elaborazione possono essere inviati direttamente a tale blocco di memoria, estraneo al flusso dei dati delle elaborazioni intermedie della FFT e che quindi può essere letto con tutta tranquillità mentre il sistema procede con e successive elaborazioni.

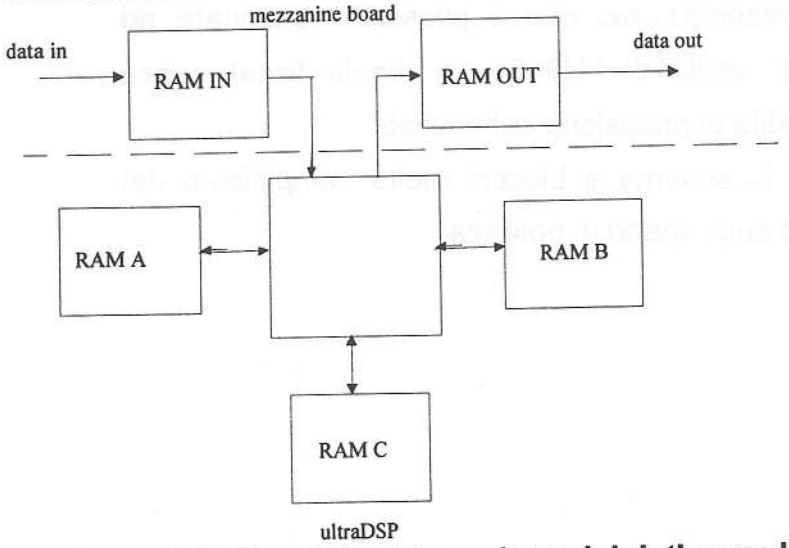


Figura A.15 flussi di elaborazione dei dati usando una mezzanine board con RAM statica

In pratica i dati elaborati sono scritti in RAM OUT e da qui letti mentre l'elaborazione che avviene palleggiando i dati intermedi fra RAMA e RAMB può continuare.

Incidentalmente si nota come l'uso della RAM in uscita può essere integrato con il sistema per l'averaging dei dati descritto nel capitolo relativo al calcolo degli spettri di potenza direttamente nella mezzanine board, identificando la memoria di accumulazione con quella di uscita (i dati sono poi scritti sullo streamer a nastro magnetico prelevandoli direttamente da quest' ultima).

Questo costituirebbe una soluzione ben più elegante di quella (peraltro dichiaratamente provvisoria) attualmente adottata e che, come ricordiamo, fa uso di una seconda scheda UltraDSP che realizza la sola operazione di accumulazione dei risultati.

Di questa scheda è impiegato solamente il microprocessore DSP32C della AT&T che gestisce le operazioni di memorizzazione dei risultati su uno dei banchi di memoria ma che necessita della conversione dei dati dal formato in virgola fissa a quello in virgola mobile.

E questa come si vede una soluzione "sovrabbondante" del problema in quanto usa una costosa UltraDSP laddove si potrebbero usare semplici sommatori a virgola fissa e necessita inoltre di un inutile interfaccia per il cambio di formato di rappresentazione, ove è possibile continuare ad usare lo stesso formato di uscita di LH9124 (a virgola fissa) senza sostanziali aumenti della perdita di precisione dei risultati.

La figura illustra in breve lo schema a blocchi molto semplificato del sistema descritto nel capitolo sugli spettri di potenza:

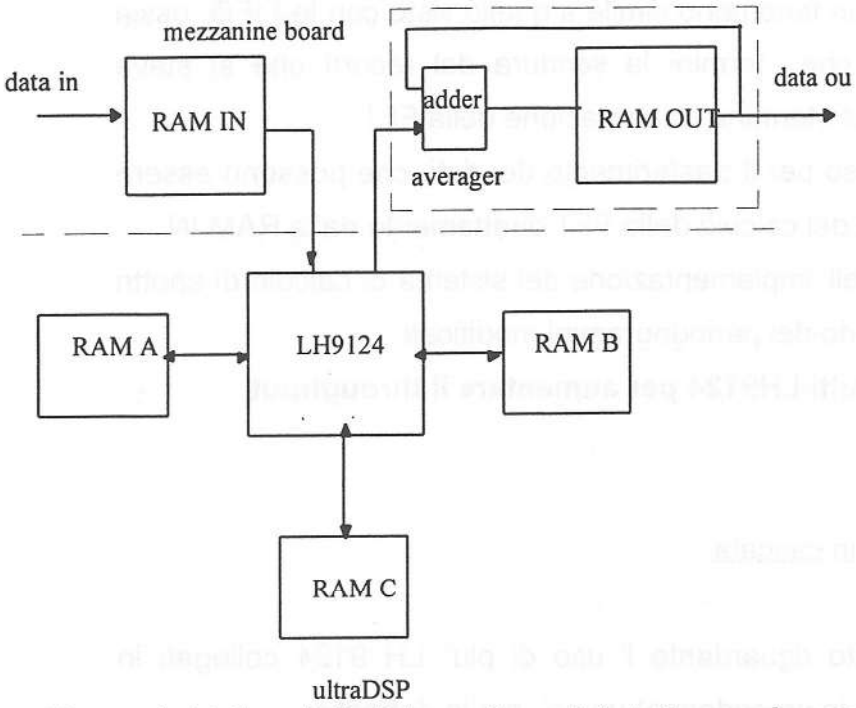


Figura A.16 flussi di elaborazione dei dati usando una mezzanine board con averager a virgola fissa

Naturalmente per poter memorizzare i dati nello streamer è ancora necessario eseguire una conversione di formato dei dati ed essendo in questo caso il buffer di uscita implicato nell' accumulazione dei risultati intermedi occorrerà interrompere l' elaborazione alla fine di ogni integrazione per poter scrivere i risultati.

Per evitare il blocco delle operazioni sarebbe auspicabile l'adozione di un sistema di memorizzazione dei risultati più efficiente, in particolare dovrebbe essere in grado di eseguire il trasferimento di un record di risultati in un tempo inferiore a quello usato da LH9124 per elaborarlo (sempre per rimanere al nostro esempio di record da 256Kword in 0.046 sec. ossia con un transfer rate di 5.7Mbyte/sec).

Ricapitolando, l'uso di una mezzanine board con RAM statica ed averager degli spettri di potenza presenta i seguenti vantaggi rispetto al sistema attuale:

1. consistenti risparmi nel tempo di acquisizione dei dati, si eliminano le inefficiente viste con l'uso della FIFO, è però preferibile prevedere un sistema che blocchi il deposito dei dati quando la RAM è piena

altrimenti succede un fenomeno simile a quello visto con le FIFO, ossia bisogna aspettare che termini la scrittura del record che si stava campionando quando termina l'elaborazione della FFT.

2. risparmio di un passo per il trasferimento dei dati, che possono essere letti al primo passo del calcolo della FFT direttamente dalla RAM IN.
3. razionalizzazione dell' implementazione del sistema di calcolo di spettri di potenza col metodo dei periodogrammi modificati.

5.6. architetture multi-LH9124 per aumentare il throughput

A.7.6.1 uso di LH9124 in cascata

Analizziamo l'argomento riguardante l' uso di piu` LH 9124 collegati in cascata (gia` affrontato in precedenza) un po' piu`in dettaglio.
Il motivo per cui in generale non è possibile eseguire calcoli di trasformate in real-time, almeno per frequenze di campionamento abbastanza elevate, è che un solo LH9124 è presente sulla scheda, e quindi bisogna aspettare che esso abbia finito il computo di una sequenza prima di poter caricare la successiva. Per aumentare il throughput una soluzione che si evidenzia in modo immediato dalla semplice osservazione della tecnica usata per calcolare le Fast Fourier Transform è quella di aumentare il numero di LH9124 della scheda, mettendoli in cascata, in modo che l'uscita di uno costituisca l'ingresso del successivo. Si mostrerà che usando un numero di LH9124 pari al numero di passi che si devono eseguire per il calcolo della trasformata, viene garantito il real-time anche per frequenze di campionamento qualsivoglia, quindi anche uguali ai 40 MHz che costituiscono la capacità massima del convertitore A/D da noi usato.
In pratica ogni chip esegue il calcolo di un passo dell'algoritmo della FFT, questo fa si che a regime venga emesso uno spettro ad ogni passo del programma. Ad esempio se per eseguire il calcolo di uno stadio di una FFT di 4K punti (da calcolare con tre stadi radix-16) ci vogliono 4096x25ns=102.4us (trascurando i tempi di programmazione e le latenze per semplicità), al termine di questo periodo il primo LH9124 della cascata

riversa i suoi dati nella RAM del secondo e può iniziare ad elaborare il record successivo, contemporaneamente il secondo esegue il secondo passo dell' algoritmo sul primo record e così via. La seguente tabella aiuta a comprendere quali sono le azioni che i vari stadi in cascata compiono ad un certo istante, considerando ad esempio tre stadi in cascata :

sequenza caricata	LH9124-1	LH9124-2	LH9124-3
1	idle	idle	idle
2	1	idle	idle
3	2	1	idle
4	3	2	1
...	...	...	...
n	n-1	n-2	n-3
n+1	n	n-1	n-2
n+2	n+1	n	n-1
n+3	n+2	n+1	n
n+4	n+3	n+2	n+1

Tabella A.19 indice del record analizzato ad ogni passo da ciascun LH9124

Per ribadire il concetto si considera il seguente esempio che evidenzia la struttura a pipeline per il calcolo di una FFT da 1Kpunti complessi, eseguita con l'uso di finestre (BWND2-BFLY16-BFLY16):

LH9124-1	BWND4	BWND4	BWND4	BWND4	BWND4
LH9124-2	BFLY16	BFLY16	BFLY16	BFLY16	BFLY16
LH9124-3	BFLY16	BFLY16	BFLY16	BFLY16	BFLY16

La diversa retinatura indica quale sequenza sta per essere elaborata dal particolare microprocessore della cascata a quel passo:

	record n-3
	record n-2

	record n-1
	record n
	record n+1
	record n+2
	record n+3

in pratica è garantita l'emissione di una trasformata ogni 102.4us, anziché ogni 312.3 come accadeva nel caso di un solo microprocessore.

Si noti che il tempo dopo il quale viene emesso lo spettro di un certo segnale in ingresso non varia rispetto al caso ad un solo processore, quello che aumenta è il numero di spettri emessi nell' unità di tempo.

Va detto che questa è una soluzione che pur non essendo troppo complessa da punto di vista dell' implementazione (per garantire la sincronizzazione è sufficiente programmare esattamente i vari generatori di indirizzi senza l'aggiunta di logica supplementare), risulta in pratica molto dispendiosa. Questo perche' necessita, oltre che dei vari LH9124 anche di un gran numero di blocchi di memoria in cui depositare i dati in trasferimento da un chip all' altro (ne servono due da 48 bit l'uno). Perché sia possibile il funzionamento a pipeline del sistema, in uno vengono depositati i dati in uscita dal processore precedente e contemporaneamente dall'altro vengono letti i dati che servono a quello successivo, al passo seguente le funzioni dei due blocchi si invertono. Inoltre naturalmente ci vogliono altri due blocchi in ingresso ed in uscita per l'I/O, in uno sono depositati i dati in ingresso e dall' altro sono prelevati i dati in elaborazione immagazzinati nel passo precedente.

Il tutto non potrà in generale essere contenuto in una sola scheda VME ma dovrà essere dislocato su più schede, collegate in modo modulare con bus dedicati ad alta velocità, ed in numero variabile a seconda della dimensione del campione e quindi del numero di passi dell' algoritmo di calcolo della FFT (per avere una struttura efficiente, come si è visto, il numero delle schede in cascata deve essere uguale al numero di passi dell'algoritmo di FFT che si è impiegato).

A.7.6.2. uso di LH9124 in parallelo.

Si possono individuare varie tipologie di connessione in parallelo delle schede, per ognuna andremo a valutare le caratteristiche e i vantaggi.
Il più semplice dei modi per incrementare il troughput del sistema ed anche il più economico consiste nell'usare più schede identiche e dal funzionamento indipendente inviando alternativamente ad ognuna un pacchetto di dati da elaborare mediante un sistema i commutazione mediante un multiplexer delle uscite dal campionatore.
L'efficenza è in tal modo aumentata di un fattore uguale al numero di schede messe in parallelo.

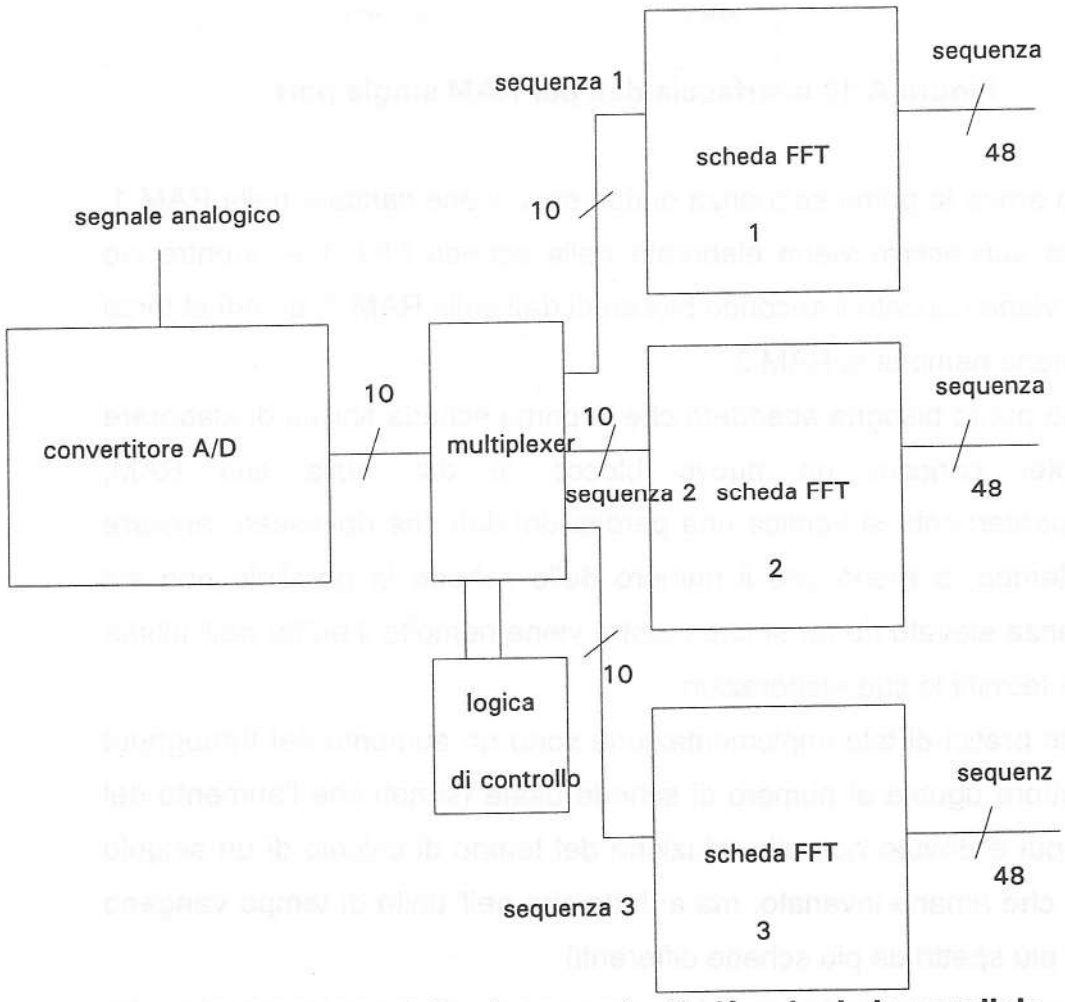


Figura A.17 schema di funzionamento di più schede in parallelo

Andiamo dunque a studiare il flusso dei dati del sistema, per semplicità supporremo che la scheda di interfaccia sia una RAM statica di dimensioni sufficienti a contenere tutto il record dei dati. Poiché la RAM è del tipo ad una sola porta è necessario averne una per ogni scheda FFT , inoltre deve esserci una logica di multiplexing per l'indirizzamento corretto dei dati

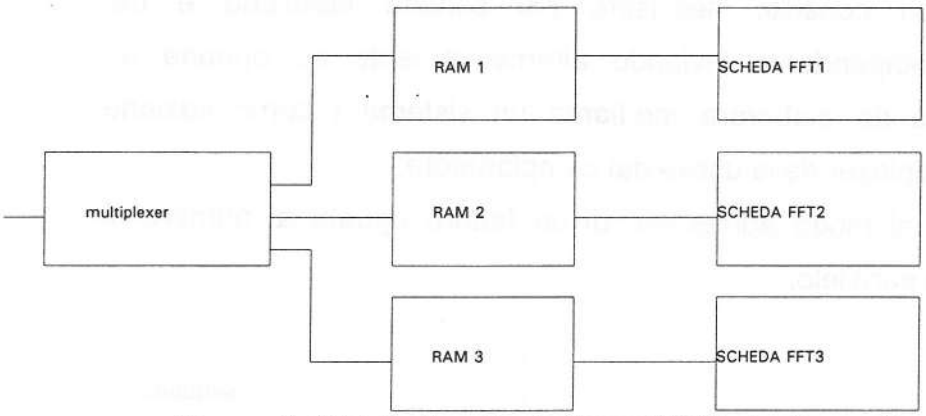


Figura A.18 Interfaccia dati per RAM single port

Quando arriva la prima sequenza di dati essa viene caricata nella RAM 1, al passo successivo viene elaborata dalla scheda FFT 1 e, mentre ciò accade viene caricato il secondo blocco di dati sulla RAM 2, quindi al terzo passo viene riempita la RAM 3.

A questo punto bisogna attendere che la prima scheda finisca di elaborare per poter caricare un nuovo blocco di dati sulla sua RAM, conseguentemente si verifica una perdita dei dati che dovessero arrivare nel frattempo, a meno che il numero delle schede in parallelo non sia abbastanza elevato da far sì che mentre viene riempito il buffer dell' ultima la prima termini le sue elaborazioni.

Gli effetti pratici di tale implementazione sono un aumento del throughput di un fattore uguale al numero di schede usate (si noti che l'aumento del throughput è dovuto non alla riduzione del tempo di calcolo di un singolo spettro, che rimane invariato, ma al fatto che nell' unità di tempo vangano prodotti più spettri da più schede differenti) .

Per quanto riguarda la massima frequenza di campionamento che consente l'elaborazione in real time basta assicurarsi che il tempo

necessario a riempire tutte le RAM delle singole schede sia inferiore a quello usato per l'elaborazione di un solo spettro.
Questo implica ancora un incremento del valore della massima frequenza di campionamento permessa di un fattore pari al numero di schede usate.
Ciascuna scheda occupa uno slot del bus VME ed è di fatto scollegata dalle altre tranne per quanto riguarda la logica di multpexaggio dei dati provenienti dal campionatore.
Concludendo si può vedere come in pratica questa soluzione non faccia altro che clonare quella a scheda singola, migliorandone di conseguenza le performance globali.
Nel caso di calcolo di trasformate di record di grandi dimensioni per cui il tempo di elaborazione è molto elevato per i quali il numero di record persi nell' attesa che la prima scheda abbia terminato l'elaborazione sia alto si può dare una seconda soluzione con l'obiettivo di minimizzare l'hardware necessario alla realizzazione del buffer di interfaccia, pur con qualche limitazione nella flessibilità del sistema.

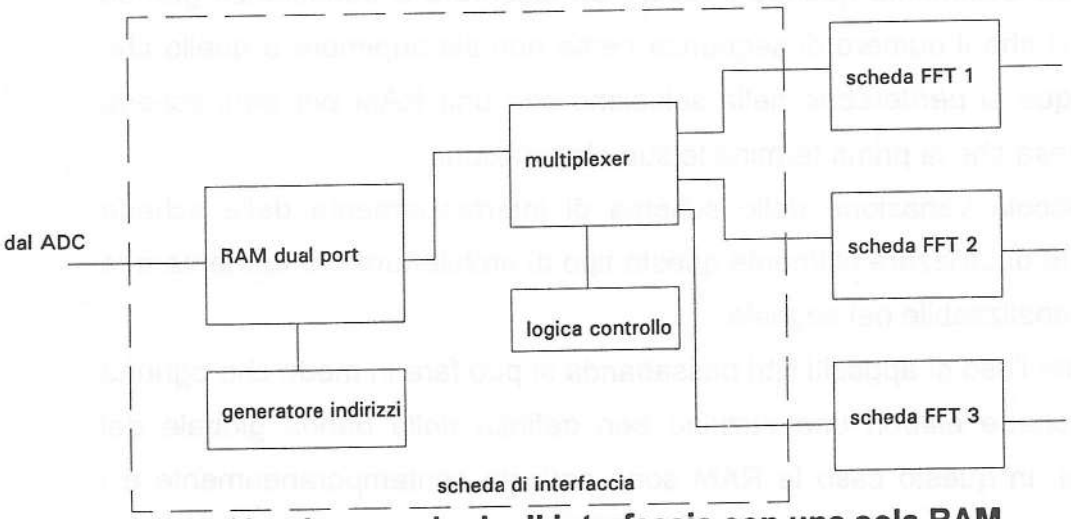


Figura A.19 schema scheda di interfaccia con una sola RAM

In pratica si fa uso di una sola RAM e si ingloba nella scheda del buffer anche la logica di multiplexing.
Inizialmente si caricano i dati sulla RAM, quindi questi vengono letti, opportunamente indirizzati dal multiplexer ad una delle schede. I dati in arrivo durante questa seconda fase sono naturalmente persi. Per poterne

caricare di nuovi occorre aspettare che il buffer si liberi e questa naturalmente è una limitazione della soluzione, non molto pesante però se si considera che anche con la prima soluzione un certo numero di sequenze è perso, se il tempo di elaborazione è abbastanza lungo.

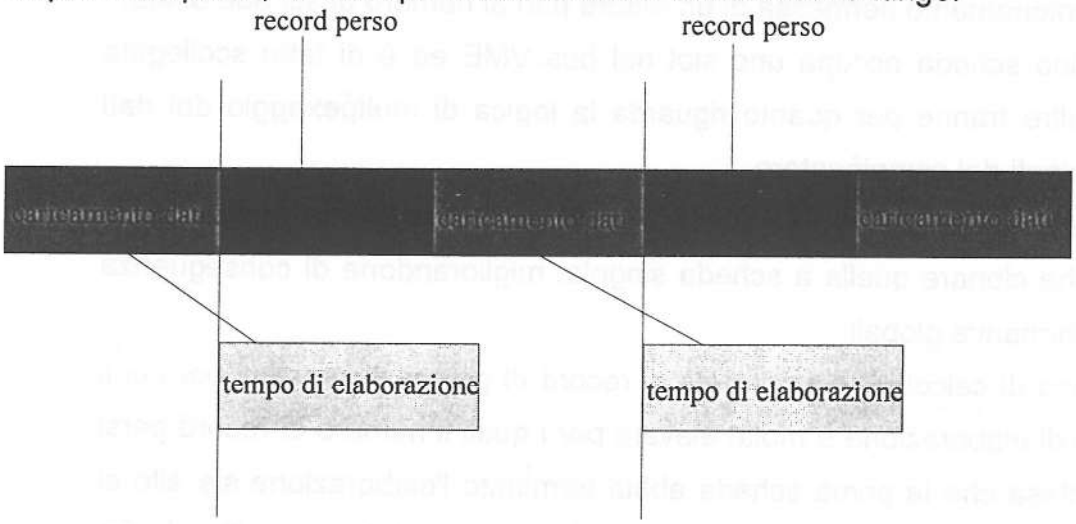


Figura A.20 temporizzazioni per la soluzione con una sola RAM

In conclusione la soluzione ad una sola RAM è utilizzabile senza perdita di efficienza solamente quando il record da elaborare è abbastanza grande da far si che il numero di sequenze perse non sia superiore a quello che comunque si perderebbe nella soluzione con una RAM per ogni scheda nell' attesa che la prima termine le sue elaborazioni.

Una piccola variazione dello schema di interfacciamento della scheda permette di utilizzare utilmente questo tipo di architettura per aumentare la banda analizzabile del segnale.

Mediante l'uso di appositi filtri passabanda si può fare in modo che ognuna delle schede elabori una sezione ben definita della banda globale del segnale, in questo caso le RAM sono caricate contemporaneamente e i risultati escono contemporaneamente, non si ha dunque un aumento del throughput del sistema, ma solo della sua banda utile.

lo schema di realizzazione in questo caso diventa:

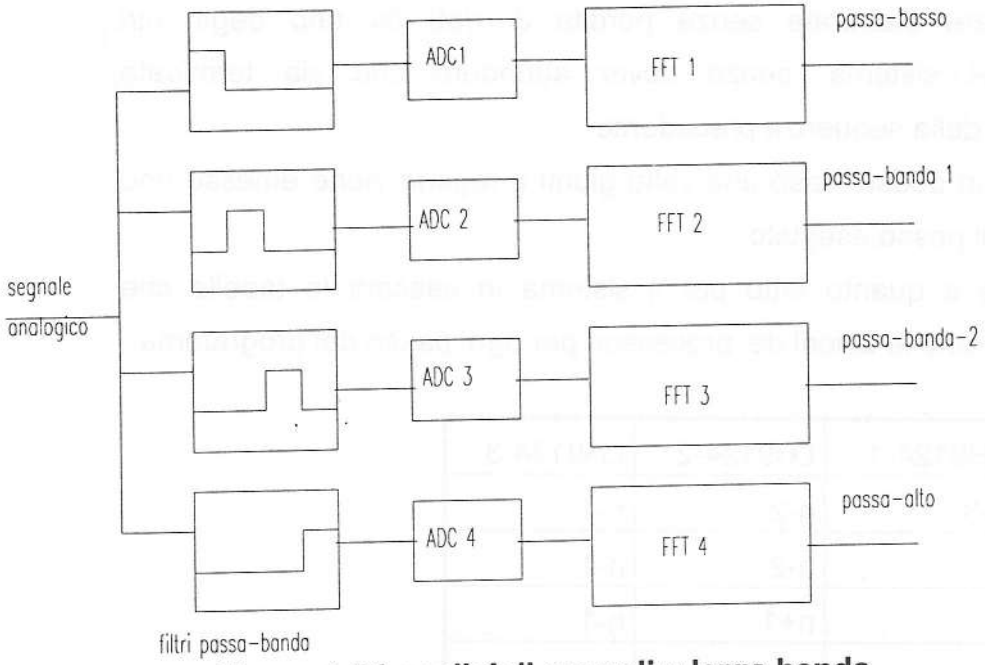


Figura A.21 analisi di segnali a larga banda

I filtri servono appunto a selezionare la particolare sezione della banda che deve essere analizzata dalla scheda FFT a cui sono connessi.

L'ampiezza della banda del segnale in questo caso dipende, come è evidente, dal numero di cloni (N) della struttura base presenti e dalla frequenza di campionamento di ciascun convertitore analogico-digitale (F).

L'ampiezza massima della banda analizzabile da ciascun clone è naturalmente pari alla metà della frequenza di campionamento dell'ADC e quindi la banda del segnale è data da N·F/2.

Un'ultima soluzione, rappresentata nella tavola allegata, generalizza i vantaggi di quella a più LH9124 esaminata per prima nel caso che il numero di schede messe in parallelo sia uguale a quello dei passi della trasformata.

Valgono quindi le stesse considerazioni fatte allora con il vantaggio che ogni processore in questo caso può eseguire un differente algoritmo.

Una apposita logica di controllo provvede a caricare ciascuna replica con la sequenza che la riguarda, in questo caso ogni LH9124 utilizza algoritmi di calcolo indipendenti dagli altri continuando a palleggiare i risultati di ogni passo fra i suoi banchi di memoria, come nel caso della configurazione a singolo chip, la differenza è che le sequenze che seguono la prima

possono essere elaborate senza perdita di dati da uno degli altri processori del sistema, senza dover attendere che sia terminata l'elaborazione della sequenza precedente.
Quindi anche in questo caso una volta giunti a regime viene emesso uno spettro ad ogni passo eseguito.
Analogamente a quanto fatto per il sistema in cascata le tabelle che seguono mostrano le azioni dei processori per ogni passo del programma:

sequenza	LH9124-1	LH9124-2	LH9124-3
n	n-3	n-2	n-1
n+1	n	n-2	n-1
n+2	n	n+1	n-1
n+3	n	n+1	n+2
n+4	n+3	n+1	n+2

Tabella A.20 indici dei record analizzati ad ogni passo da ciascun LH9124

Si riprende l'esempio della trasformata complessa di 1Kpunti vista per l'architettura a cascata:

LH9124-1	BFLY16	BWND2	BFLY16	BFLY16	BWND2
LH9124-2	BFLY16	BFLY16	BWND2	BFLY16	BFLY16
LH9124-3	BWND2	BFLY16	BFLY16	BWND2	BFLY16

Si riporta per comodità la legenda della retinatura che comunque adotta le stesse convenzioni del caso precedente:

	sequenza n-3
	sequenza n-2
	sequenza n-1
	sequenza n
	sequenza n+1

	sequenza n+2
	sequenza n+3

Per quanto riguarda il costo del sistema dal punto di vista dell' implementazione esso è circa pari al costo di un singolo processore, con i suoi blocchi di memoria e gli usuali dispositivi al contorno, moltiplicato per il numero di processori messi in parallelo, in più bisogna progettare una apposita rete per smistare i segnali in ingresso fra i vari LH9124 componenti.

A.7.6.3. Architettura combinata cascata/parallelo

In entrambe le soluzioni viste sopra il tempo di latenza tra l'ingresso dei campioni e la produzione della FFT rimane invariato rispetto al caso monoprocessore.
Non si hanno dunque vantaggi dal punto di vista della velocità di elaborazione di un singolo spettro, ma aumenta solo la quantità di spettri che vengono emessi nell' unità di tempo.
Se invece il problema della latenza di un singolo spettro è quello principale si può adottare una architettura mista serie/parallelo, che ha il vantaggio di ridurre il tempo di calcolo di ogni singolo passo.
Il principio è quello di partizionare la struttura di ogni passo per un fattore k, potenza di due, riducendo così il tempo di esecuzione di ciascun passo dello stesso fattore.
In pratica si utilizza lo stesso principio che sta' alla base del calcolo della Fast Fourier Transform che consiste nel dividere il record in sottogruppi di dimensioni più ridotte e quindi più rapidi da calcolare.
In base a tale principio, usando due stadi in parallelo si possono eseguire contemporaneamente due passi radix-2, usandone 4 due radix-4 e così via.
Riferiamoci come esempio alla realizzazione riportata in figura e supponiamo di dover calcolare la trasformata di un record di 1024 punti.

Inizialmente, come si era visto nella descrizione del calcolo della FFT si scompone l'insieme di campioni in due gruppi:

$$X(k) = \sum_{n=0}^{1023} x(n) W_{1024}^{kn} = \sum_{n=0}^{511} x(2n) W_{1024}^{2nk} + W_{1024}^{k} \sum_{n=0}^{511} x(2n+1) W_{1024}^{2nk} =$$

$$= H(k) + W_{1024}^{k} G(k)$$

in cui:

$$H(k) = \sum_{n=0}^{511} x(2n) W_{512}^{kn}$$

$$G(k) = \sum_{n=0}^{511} x(2n+1) W_{512}^{kn}$$

I dati in ingresso (reali) sono inviati alternativamente alla parte superiore e inferiore delle schema.

In pratica la parte superiore esegue in cascata secondo l'algoritmo 16×16× 2 su 512 punti, quindi con tre stadi, il calcolo della H(k) mentre quella inferiore della G(k), il particolare schema di cablaggio dei processori provvede automaticamente alla ricombinazione dei risultati per ottenere .

Il fatto che i calcoli di ciascuna sezione siano eseguiti in cascata fa sì, come abbiamo visto, che sia possibile iniziare l'elaborazione della sequenza successiva immediatamente dopo la fine della precedente.

In pratica l'operazione fatta è equivalente al calcolo in parallelo di due trasformate di lunghezza dimezzata rispetto a quella originaria, con conseguente riduzione della latenza e del tempo di calcolo.

In pratica la latenza, sempre nello schema considerato cala di un fattore due ossia si dimezza il tempo necessario ad ottenere la trasformata di un determinato insieme di campioni e si moltiplica di un fattore 6 il throughput.

Il problema di questo tipo di architettura è che essa presenta grossi problemi dal punto di vista dell' implementazione per la sua notevole complessità.

Bisognerà prevedere infatti una logica di indirizzamento dei segnali e dei dati. Adottando poi una soluzione che usa due processori in parallelo per

ogni stadio in cascata conviene usare una memoria RAM del tipo Dual-port per semplificare gli interfacciamenti.

confronto di prestazioni fra le varie architetture

Per quanto riguarda il calcolo del throughput assumeremo che:

- in ingresso vi sia una RAM statica in grado di immagazzinare l'intero record da elaborare, in modo che non ci siano perdite di dati in lettura.
- il numero degli stadi nelle architettura in serie ed in parallelo e quello degli stadi in serie nell'architettura mista sia pari al numero dei passi dell' algoritmo di calcolo della FFT.
- si trascurino i cicli di ritardo causati dall'implementazione (cicli per la programmazione dei dispositivi), ossia si assuma il caso ideale in cui le prestazioni del sistema siano le massime possibili

Supponiamo ad esempio di dover calcolare una FFT da 4K punti, con una struttura 16x16x16:

Per un **singolo LH9124** che funziona in maniera recursiva ciò prende:

4096 punti x 3 passi radix-16 x 40MHz + 3 passi radix-16 x 68 cicli di latenza x 40MHz = 312.3μs

Eseguendo lo stesso algoritmo su **due stadi in parallelo** si divide questo tempo per due:

(4096 punti x 3 passi radix-16 x 40MHz + 3 passi radix-16 x 68 cicli di latenza x 40MHz)/2=156μs

usando **tre stadi in cascata**:

4096 punti x 1 passo radix-16 per stadio x 40MHz + 1 passo radix-16 x 3 nodi x 68 cicli di latenza x 40MHz= 108μs

usando infine un **sistema misto cascata/parallelo**, con due stadi paralleli e tre in cascata:

(4096 punti x 1 passo radix-16 per stadio x 40MHz)/2 stadi+(1 passo radix-16 x 3 nodi x 68 cicli di latenza)x 40MHz=56.3μs

La seguente tabella indica i tempi di attesa per l'emissione di uno spettro in funzione delle varie architetture che è possibile usare, i tempi sono espressi in secondi:

dim.	struttura (FFT reale)	1 chip	parallelo	cascata	misto
2048	R16xR16xR4	1.59E-0	7.94E-05	5.63E-05	2.82E-05
4096	R16xR16xR16	3.12E-0	1.56E-04	1.08E-04	5.38E-05
8192	R16xR16xR16xR2	8.26E-0	4.13E-04	2.12E-04	1.06E-04
16384	R16xR16xR16xR4	1.65E-0	8.23E-04	4.16E-04	2.08E-04
32768	R16xR16xR16xR4x2	4.10E-0	2.05E-03	8.28E-04	4.14E-04
65536	R16xR16xR16xR16	6.56E-0	3.28E-03	1.65E-03	8.23E-04
131072	R16xR16xR16xR16xR2	1.64E-0	8.20E-03	3.29E-03	1.64E-03
262144	R16xR16xR16xR16xR4	3.28E-0	1.64E-02	6.56E-03	3.28E-03
524188	R16xR16xR16xR16xR4xR2	7.86E-0	3.93E-02	1.31E-02	6.56E-03

che messo in grafico mostra il forte miglioramento nei tempi di calcolo:

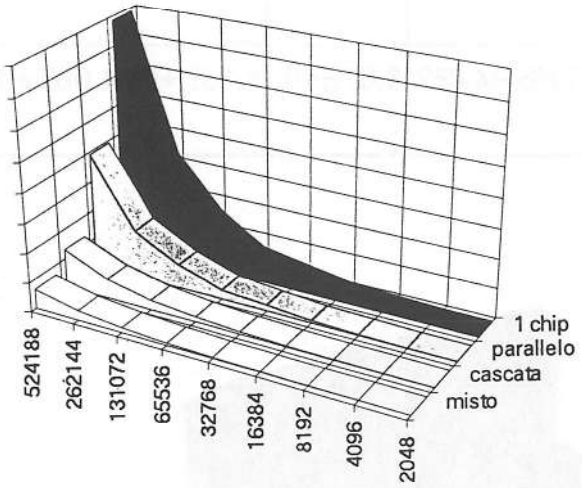


Figura A.22 comparazione tempi di calcolo per varie ipotesi di architettura

come si vede il rapporto fra i tempi di elaborazione fra il caso di architettura mista seriale/parallela e a singolo chip può essere anche di 6:1
Si riporta anche la tabella del throughput nei vari casi:

dim.	struttura (FFT reale)	1 CHIP	parallelo	CASCATA	MISTO
2048	R16xR16xR4	1.29E+0	2.58E+0	3.64E+07	7.28E+07
4096	R16xR16xR16	1.31E+0	2.62E+0	3.81E+07	7.62E+07
8192	R16xR16xR16xR2	9.92E+0	1.98E+0	3.87E+07	7.74E+07
16384	R16xR16xR16xR4	9.96E+0	1.99E+0	3.93E+07	7.87E+07
32768	R16xR16xR16xR4x2	7.98E+0	1.60E+0	3.96E+07	7.92E+07
65536	R16xR16xR16xR16	9.99E+0	2.00E+0	3.98E+07	7.97E+07
131072	R16xR16xR16xR16xR2	8.00E+0	1.60E+0	3.99E+07	7.98E+07

262144	R16xR16xR16xR16xR4	8.00E+0	1.60E+0	3.99E+07	7.99E+07
524188	R16xR16xR16xR16xR4xR2	6.67E+0	1.33E+0	4.00E+07	7.99E+07

con relativo grafico:

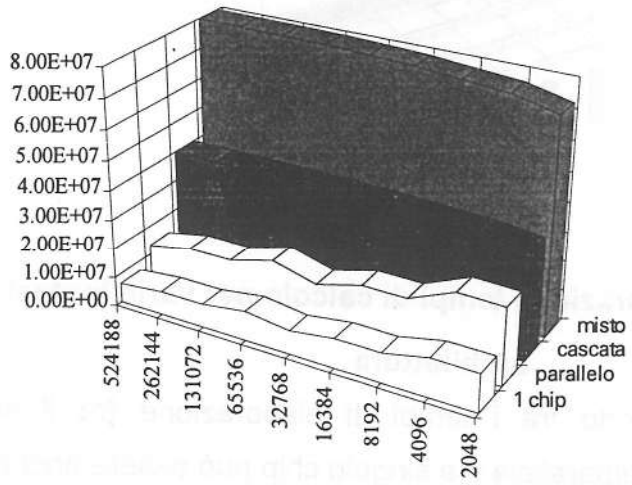


Figura A.23 comparazione throughput per varie ipotesi di architettura

CONCLUSIONI

Per concludere il lavoro si riassumono qui i principali risultati ottenuti.
Si riporteranno brevemente le caratteristiche dei vari parametri che descrivono il sistema per il calcolo degli spettri di potenza ed in base ad essi si illustreranno brevemente le modifiche all'hardware esistente che si sono suggerite durante lo svolgimento. Infine si confronteranno le principali prestazioni che ci si può attendere dalla nuova architettura con quelle misurate sul sistema esistente.

definizione dello spettro

Si è visto come la definizione (ossia la minima distanza in termini di frequenza fra una riga e l' altra dello spettro) dipenda dal numero di canali utilizzati e dalla frequenza di campionamento del segnale analogico da parte del convertitore A/D.
A causa del teorema di Shannon la massima frequenza che può avere il segnale in ingresso non può essere superiore alla metà di quella del sampling rate.
Il convertitore di cui si dispone nel sistema Valley può campionare con frequenze variabili da 0 a 40 MHz; naturalmente quanto più elevato è tale rate, tanto migliore sarà la definizione dello spettro finale, a parità di dimensione del record di campioni. Ad esempio, campionando a 40 MHz (cioè con un periodo di campionamento $\tau = 0.25\mu s$), ed utilizzando l'intera memoria indirizzabile del sistema, che è di $N=2^{20}$ punti, otterremo, nel caso ideale, una **risoluzione in frequenza** :

$$\Delta f = \frac{1}{T_R} = 38.15Hz$$

in cui con $T_R = N\tau$ è stato indicato il **time record**, cioè l'intervallo temporale necessario ad acquisire tutti i campioni.
Nel caso della scheda UltraDSP lo stesso parametro vale, considerando la massima dimensione del record (128Kpunti) :

$$\Delta f = \frac{1}{T_R} = 305Hz$$

Per quanto riguarda la dimensione massima della banda del segnale analogico in ingresso essa è ovviamente la stessa per entrambi i casi, dipendendo dalla frequenza di campionamento:

$$0, f_{\max}] = \frac{1}{\alpha\tau} = 20\,MHz$$

in cui α è il coefficiente usato dal teorema di Shannon ossia α=2.

Si nota che al diminuire della frequenza di campionamento aumenta la definizione dello spettro ossia diminuisce la distanza fra le righe, il che è una cosa augurabile ma che ha come contraltare la diminuzione della dimensione della banda del segnale. Tale fenomeno, come si è visto è usato nella **decimazione** quando interessa avere una alta risoluzione in frequenza nell' analisi di segnali a banda stretta.

Vediamo una tabella che indica la risoluzione massima ottenibile e la banda corrispondente per alcune frequenze di campionamento e dimensioni del record:

	1K	16K	64K	128K	256K	1M	banda utile
10Mhz	9765.6	610.35	152.58	76.29	38.14	9.53	5Mhz
20Mhz	19531.25	1220.7	305.178	152.58	76.29	19.07	10Mhz
30Mhz	29296.88	1831.05	457.76	228.8	114.44	28.61	15Mhz
40Mhz	39062.5	2441.4	610.35	305.16	152.58	38.14	20Mhz

Tabella A.21 definizione in frequenza e frequenza massima del segnale per vari sampling rate e per varie dimensioni del numero di campioni nel time record

Questi risultati rappresentano le massime definizioni ottenibili con segnali in ingresso di periodo multiplo intero del time record.

Avendo a che fare con segnali provenienti da osservazioni astronomiche, o comunque da sorgenti esistenti nella tecnica, essi non si potranno mai considerare rigorosamente periodici o in quanto affetti da rumore, o perché non ne è nota a priori la periodicità, o perché sono segnali transienti.

In questi casi si è visto come per il loro studio occorra ricorrere all' uso di funzioni di pesatura dei dati ossia di **windows** che in generale hanno l'effetto diminuire la definizione e la banda utile del segnale in maniera dipendente dal tipo di finestra usato .

Si può esprimere la definizione di un segnale pesato con una windows come:

$$\Delta f = \beta\left(\frac{1}{\tau N}\right)$$

ove β rappresenta il **coefficiente di perdita di definizione**. I valori di β per le finestre più usate sono riportati nella tabella alla fine del capitolo 1. Ad esempio per la Hamming window (una delle più usate) β=1.36, mentre per la finestra triangolare è di 1.33.

Vediamo dunque come si modifica la tabella delle definizioni impiegando una finestra di pesatura dei dati di tipo Hamming[1], una delle più usate:

	1K	4K	16K	64K	128K	256K	1M
10Mhz	13.2Khz	3.3KHz	830	207.51	103.74	51.87	12.96
20Mhz	26.5Khz	6.6Khz	1660.1	415.03	207.5	103.75	25.93
30Mhz	40Khz	9.9KHz	2490.2	622.55	311.26	155.63	38.90
40Mhz	53Khz	13.2KHz	3320.3	830.07	415.02	207.51	51.87

Tabella A.22 definizioni in frequenza per vari sampling rate e per varie dimensioni del numero di campioni del time record

Allo scopo di aumentare la definizione abbiamo esaminato alcune soluzioni algoritmiche implementabili direttamente o indirettamente su LH9124. Abbiamo visto quindi come si può sfruttare anche la metà degli ingressi del chip relativa ad ingressi immaginari per raddoppiare il numero di punti di cui si può calcolare la trasformata, utilizzando algoritmi di calcolo per campioni esclusivamente reali ed ottenendo in questo modo trasformate fino a 2 milioni di punti (quindi raddoppiando la definizione).

---

[1] l'espressione che caratterizza la finestra di Hamming è:

$$w(n) = \begin{cases} 0.45 + 0.46\cos\left[\frac{2\pi n}{N}\right], n = -\frac{N}{2},\ldots,-1.0,1,\ldots,\frac{N}{2} \\ 0.54 - 0.46\cos\left[\frac{2\pi n}{N}\right], n = 0,1,2,\ldots,N-1 \end{cases}$$

Abbiamo anche visto come effettuare zoom della frequenza, a parità di punti della trasformata (in questo caso a scapito della larghezza di banda rappresentabile).
Per illustrare i risultati di questi algoritmi si sono eseguite delle simulazioni software del comportamento del LH9124 mediante il programma Real Time Simulator, prodotto dalla stessa Sharp e le si è confrontate con i valori sperimentali ottenuti sul sistema commerciale UltraDSP della Valley installato nel laboratorio di radioastronomia del CNR.

rumore di calcolo

Si sono poi calcolate le prestazioni del sistema relativamente al rumore di quantizzazione dei campioni causato dalla lunghezza limitata dei registri del chip. E' noto come per rappresentare una grandezza in termini digitali si abbia inevitabilmente una perdita di precisione rispetto alla sua conoscenza analogica.
Ognuna delle tre fasi in cui si divide la produzione dello spettro di potenza del segnale (ossia la conversione A/D, il calcolo della FFT e l'averaging di una serie di misurazioni successive per il filtraggio del rumore che si sovrappone al segnale in ingresso) potenzialmente costituisce una sorgente di rumore di quantizzazione.
Si è visto come il notevole numero di bit dei registri del LH9124, unitamente all'adozione del formato a block floating point per la rappresentazione dei numeri digitali e soprattutto alla mancanza di generazione di rumore di calcolo internamente al chip durante il computo di una butterfly, renda sostanzialmente trascurabile il rumore di quantizzazione durante il calcolo della FFT. Abbiamo infatti mostrato prima di tutto come il formato adottato per la rappresentazione decimale, ossia il block floating point (una variazione del fixed point in cui si riduce l'effetto del rumore shiftando i bit meno significativi ad ogni passo dell'algritmo), abbia prestazioni dal punto di vista della precisione paragonabili a quello in virgola mobile e sia comunque molto superiore a quello in virgola fissa.
A questo proposito si riporta il grafico di confronto, in unità logaritmiche, fra i rapporti segnale/rumore per il calcolo di FFT nei vari formati, ricordando

che esso è stato calcolato supponendo dei vincoli fortemente restrittivi sull' impiego del block floating point, che in realtà ha un rapporto signal/noise sempre migliore di quello calcolato:

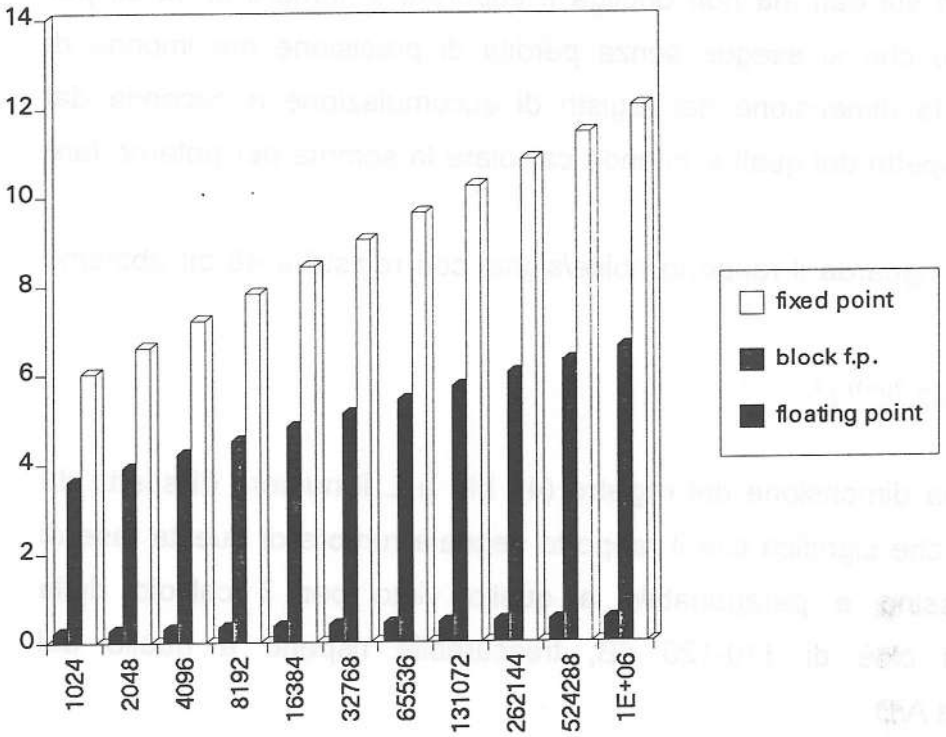


Figura A.24 raffronto del rapporto rumore/segnale in scala logaritmica

Si è visto poi come il rapporto segnale/rumore per il convertitore A/D non superi i 60dB, campionando con una precisione di soli 10 bit, mentre sia prossimo ai 110-120 per il calcolo della trasformata anche su un numero di campioni molto alto (si ricorda che il rapporto noise/signal dipende dalla dimensione del record di ingresso e dal tipo di algoritmo di calcolo usato, anche se l'uso del formato block floating point riduce notevolmente la dipendenza da questi parametri).

Per quanto riguarda il rumore generato dall'averaging degli spettri di potenza secondo il metodo dei periodogrammi modificati, si sono previste due possibili soluzioni implementative, una che fa uso dei soli sommatori interni ai LH9124 e una più complessa ma assai più precisa che necessita di sommatori esterni a virgola fissa. Si è mostrato come la prima soluzione

introduca un errore di calcolo quasi sempre intollerabile (tranne per alcuni segnali con spettro bianco o quasi-bianco), riducendo di fatto il numero di bit significativi del dato da 48 a soli 16. La seconda soluzione può invece essere realizzata in due modi diversi, uno che introduce errore di troncamento sui dati ma non obbliga a superare il numero di 48 bit per dato, l' altro che si esegue senza perdita di precisione ma impone di aumentare la dimensione dei registri di accumulazione a seconda dal numero di spettri dei quali si intende calcolare la somma per poterne fare la media.

Per quanto riguarda il rapporto noise/signal con registri a 48 bit abbiamo calcolato:

$$\frac{E\left[\left|F(n)\right|\right]}{E\left[\left|X(n)\right|^2\right]} = 2^{-2(a+1)} L^2$$

in cui a è la dimensione del registro (48 bit) e L il numero di spettri da mediare, il che significa che il rapporto segnale rumore di questa fase di post-processing è paragonabile a quello visto per il calcolo della trasformata cioè di 110-120 dB, trascurabile rispetto a quello del convertitore A/D.

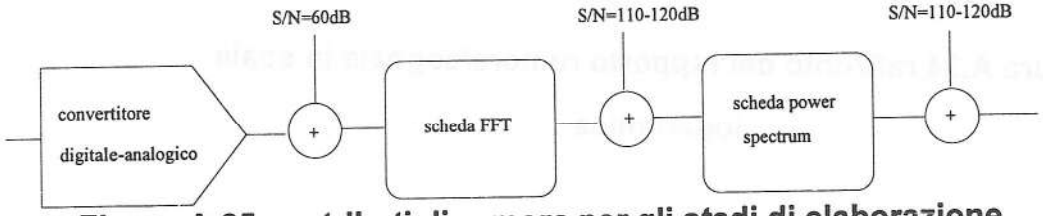


Figura A.25 contributi di rumore per gli stadi di elaborazione

TEMPI DI CALCOLO

Si sono anche calcolati i tempi di calcolo necessari per generare uno spettro di potenza, confrontando quelli ottimi teorici con quelli effettivamente rilevati mediante sperimentazioni sul sistema installato al CNR.

La loro conoscenza è importante per valutare il numero di campioni che vengono persi campionando continuamente nell' attesa che venga concluso il calcolo della FFT. L'ideale sarebbe naturalmente ridurre a zero

tale numero di campioni persi, ossia di rendere possibile la cosiddetta elaborazione in **real time**. Nel corso del lavoro si è visto anche entro quali limiti tale elaborazione è realmente possibile.

Si riportano qui i principali risultati visti, per rendere confrontabili i tempi di calcolo teorici e reali essi sono calcolati relativamente all' elaborazione di 100 spettri :Sempre allo scopo di uniformare i risultati si sono operate le operazioni di finestratura dei dati e la struttura di calcolo è quella di una trasformata di N/2 punti reali più un passo di ricombinazione

dimensione campione	tempo di elaborazione	throughput
4K·100	1.1	0.37 Msps
32K·100	1.7	1.92 Msps
128K·100	2.9	4.5 Msps
256K·100	6.1	4.3 Msps

Tabella A.23 tempi di elaborazione e throughput misurati sul sistema UltraDSP

dimensione campione	tempo di elaborazione	throughput
4k·100	0.0335 sec	12,2 Msps
32k·100	0.24 sec	13,2 Msps
128k·100	1.64 sec	13,2 Msps
256k ·100	3,27 sec	11,4 Msps

Tabella A.24 tempi di elaborazione e relativi throughput teorici per vare lunghezze del record

FFT	f_{max}
fft1k()	9.7MHz
fft2k()	4.2MHz
fft4k()	3.05MHz
fft8k	1.19MHz
fft16K	602KHz
fft32k	330.3KHz
fft64k	155Khz

fft128k	62.43Khz

Tabella A.25 massimo rate di campionamento per avere funzionamento in real time

In base alle osservazioni sul sistema esistente si sono formulate alcune ipotesi di modifica dell'architettura dello stesso per ottimizzare i tempi di calcolo .

In particolare la modifica del buffer di interfaccia per l'acquisizione dei dati provenienti dal convertitore A/D (denominato **mezzanine board**) è stata sviluppata nella tesi di Sergio Tassinari, mentre qui sono illustrate alcune migliorie al sistema di post-processing per i dati che rende più razionale ed efficiente il calcolo dell' averaging degli spettri di potenza secondo il metodo dei periodogrammi modificati.

Sempre allo scopo di migliorare le performance temporali del sistema il lavoro illustra una serie di soluzioni che usano batterie in serie od in parallelo di più LH9124 (con i relativi LH9320), in grado di realizzare sistemi di calcolo ad altissime prestazioni.

Si riporta come sintesi dei risultati ottenibili in tal modo la tabella che confronta i tempi di calcolo determinati dalle varie soluzioni descritte:

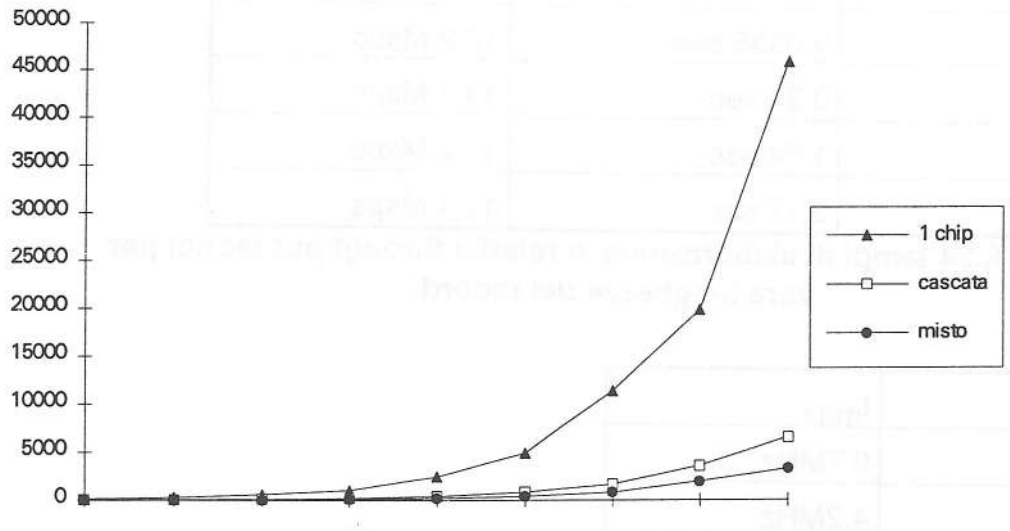


Figura A.26 confronto dei tempi di calcolo per varie architetture, al variare del numero di campioni del time record

In pratica si riesce a produrre spettri di potenza in real time anche usando record di 1 milione di punti ad una frequenza di campionamento di 40MHz, quindi sfruttando al massimo le capacità del sistema.

Con l'architettura mista serie/parallelo si possono raggiungere anche throughput di 80 megaword al secondo, andando quindi ben oltre il rate di campionamento del DAC.

GENERATORE DI INDIRIZZI HARRIS HSP45240

Come abbiamo visto nel sottosistema di ingresso/uscita, oltre al generatore di indirizzi Sharp, viene utilizzato anche il generatore di indirizzi Harris HSP45240 che, pur essendo più semplice, dispone di funzioni che non sono presenti nel dispositivo Sharp.

Esso permette di generare indirizzi a 24 bit, ad una frequenza di clock che da 0Hz può arrivare fino a 50MHz, il che lo rende adattissimo ad essere utilizzato per la generazione degli indirizzi di ingresso e di uscita, che avviene, come sappiamo, ad una frequenza diversa da quella di sistema e che non è nota a priori. E' dotato di una interfaccia standard, dello stesso tipo di quella dell'LH9320, ed il processo di generazione può essere controllato dall'esterno.

In questa appendice cercheremo di analizzare brevemente le caratteristiche principali di questo componente, invitando chi volesse saperne di più a consultare la documentazione fornita dalla Harris [18].

Descrizione generale

Facciamo riferimento alla Fig. D.1, in cui abbiamo riportato lo schema a blocchi dell'HSP45240.

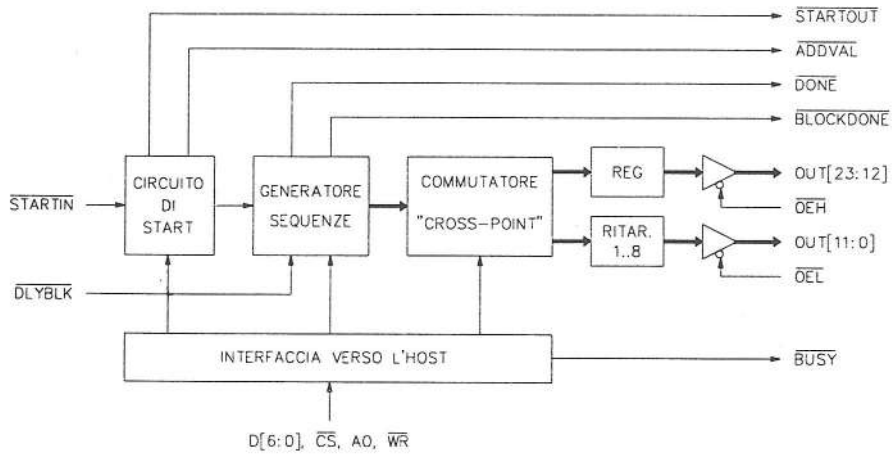


Figura D.1 - Schema a blocchi del dispositivo Harris HSP45240.

Come possiamo notare, l'architettura interna è molto semplice ed, in effetti, non vi sono molti commenti da fare.

Le uniche particolarità sono rappresentate dal fatto che il generatore da 24 bit può essere programmato per calcolare due sequenze indipendenti da 12 bit ognuna, una delle quali può essere ritardata rispetto all'altra fino a otto cicli di clock, e dalla presenza del blocco denominato "commutatore cross-point".

Si tratta di "multiplexer" programmabile, grazie al quale è possibile collegare ognuna delle uscite del generatore di sequenze interno a qualsiasi terminale fisico di uscita, in qualsiasi ordine.

Questa caratteristica permette di generare qualsiasi tipo di sequenza (anche di tipo *bit-reversed*), anche se il generatore interno è in grado di generare esclusivamente sequenze di tipo *lineare*.

Sempre osservando la Fig. D.1, notiamo come vengano anche generati molti segnali di stato, che, come vedremo, saranno molto utili nel nostro sistema.

Piedinatura e interfacciamento

Il dispositivo HSP45240 si interfaccia con il mondo esterno attraverso i segnali mostrati in Fig. D.2, di cui diamo, qui di seguito, una breve descrizione, suddividendoli per categorie.

A) Segnali di interfaccia con il sistema di controllo

- **D[6:0]** (Data Bus): Ingresso, logica positiva.

Si tratta del bus dati, *a soli 7 bit*, che viene utilizzato per la programmazione dei registri interni. La scrittura di tali registri avviene, come nello Sharp LH9320, secondo uno schema a due passi: prima si scrive l'*indirizzo* del registro e, subito dopo, si scrive il *dato* relativo. Che si tratti di un indirizzo oppure di un dato, viene stabilito dallo stato del pin A0.

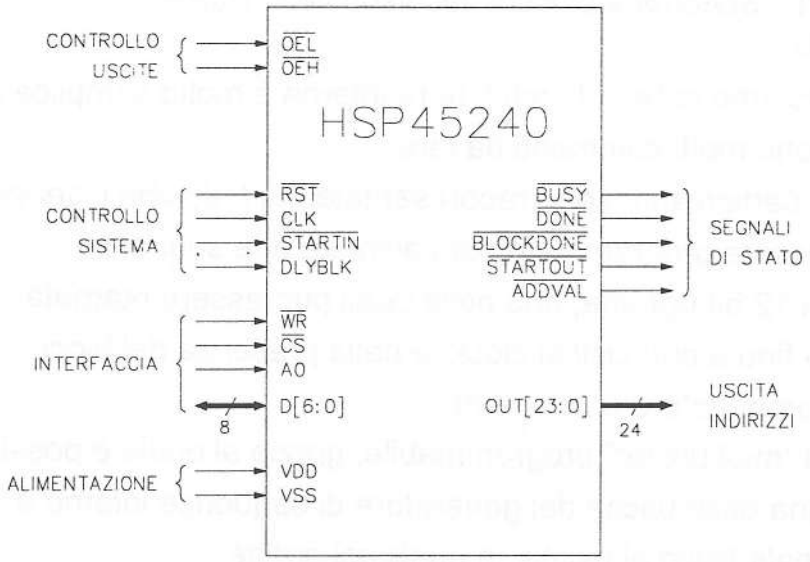


Figura D.2 - Piedinatura del Generatore di Indirizzi Harris HSP45240.

• **A0** (Address 0): Ingresso, attivo alto.

Quando questo segnale è attivo, lo stato delle linee D[6:0] definisce un *indirizzo*, mentre quando è a livello logico basso la configurazione delle linee D[6:0] viene interpretata come un *dato*.

• **WR** (Write): Ingresso, attivo sul fronte di salita.

Il "dato" (dato o indirizzo) presente agli ingressi D[6:0] viene campionato al fronte di salita di questo segnale.

• **CS** (Chip Select): Ingresso, attivo basso.

Questo segnale abilita, quando attivo, la lettura di tutti i segnali di interfaccia che abbiamo finora esaminato, permettendo, così, il colloquio con il sistema host.

B) Segnali di controllo

• **RST** (Reset): Ingresso, attivo basso.

Si tratta del segnale di reset generale del sistema. E' richiesta, a tale scopo, la presenza del segnale di clock: il processo di inizializzazione inizia non appena la linea RESET torna a livello alto e dura per 26 cicli.

- **STARTIN** (Start In): Ingresso, attivo basso.

L'attivazione di questo segnale provoca l'avvio della generazione della sequenza programmata. Dovrebbe rimanere attivo solamente per la durata di un singolo ciclo di clock, perchè il processo parte solo quando STARTIN torna a livello alto.

- **DLYBLK** (Delay Block): Ingresso, attivo alto.

Questa linea, se attivata, permette di interrompere la sequenza al termine del blocco corrente (si veda più avanti).

- **OEL, OEH** (Output Enable Low, High): ingressi, attivi bassi.

Questi segnali (asincroni) controllano l'abilitazione dei buffer di uscita, relativi, rispettivamente, alle 12 linee meno significative e alle 12 più significative.

C) Segnali di stato

- **BLOCKDONE** (Block Done): Uscita, attiva bassa.

Questa uscita segnala quando l'ultimo indirizzo del *blocco* corrente è presente alle uscite...

- **DONE** (Done): Uscita, attiva bassa.

...mentre questa segnala la presenza alle uscite dell'ultimo indirizzo dell'intera *sequenza*.

- **ADDVAL** (Address Valid): Uscita, attiva bassa.

Questa uscita segnala quando il primo indirizzo della sequenza è presente alle uscite.

- **STARTOUT** (Start Out): Uscita, attiva bassa.

Si tratta di un segnale particolare, che viene attivato quando la sequenza corrente ha inizio, a causa, però, di un meccanismo diverso da un impulso sulla linea STARTIN.

• **BUSY** (Busy): Uscita, attiva bassa.

Questa linea viene attivata durante il processo di inizializzazione del componente, a seguito di un impulso sul terminale RST. Durante questo periodo l'intera sezione di interfaccia rimane disabilitata.

Modalità di funzionamento

L'avvio della sequenza programmata può avvenire a seguito di (1) un impulso sull'ingresso STARTIN, (2) scrivendo, attraverso l'interfaccia in un apposito registro oppure (3) al termine della sequenza immediatamente precedente, qualora sia stato programmato il modo di funzionamento detto "One Shot with Restart". Solo in questi ultimi due casi viene generato il segnale STARTOUT.

La generazione vera e propria degli indirizzi può essere ritardata da 1 fino a 31 cicli di clock rispetto ad uno dei segnali di start che abbiamo appena elencato.

I possibili modi di funzionamento sono tre:

A) "One Shot Mode without Restart", in cui il generatore si arresta al termine della sequenza corrente, rimanendo in attesa di un nuovo segnale di start, che può essere inviato nei tre modi diversi sopra elencati. Al termine dell'ultimo blocco della sequenza vengono generati i segnali DONE e BLOCKDONE.

B) "One Shot Mode with Restart", che è identico al modo precedente, con la sola differenza che al termine di una sequenza viene, automaticamente, generato un nuovo segnale di start, in modo che il primo indirizzo della nuova sequenza segua *immediatamente* l'ultimo di quella appena terminata. La fine di ogni sequenza viene segnalata dall'attivazione dei segnali DONE, BLOCKDONE e STARTOUT.

C) "Continuous Mode", in cui la sequenza procede indefinitamente, senza che alcun segnale venga generato ed ignorando il parametro che definisce il numero di blocchi.

Come sarà ormai chiaro, ogni *sequenza* risulta composto da un certo numero di *blocchi*. L'HSP45240 dispone, al suo interno, di registri (*tutti a 24 bit*) in cui è possibile programmare: (1) l'indirizzo di partenza della sequenza, (2) la dimensione di ciascun blocco, (3) il numero di blocchi che compongono la sequenza, (4) l'incremento degli indirizzi tra un blocco e l'altro e (5) il passo tra un indirizzo e l'altro.

La grande flessibilità che questi registri permettono di ottenere, unitamente alla programmabilità delle uscite, fa dell'Harris HSP45240 il componente ideale per la nostra applicazione.

Diventa molto semplice, ad esempio, fare in modo che il sottosistema di I/O acquisisca M blocchi da N campioni ciascuno, oppure, utilizzando il segnale DLYBLK, leggere continuamente da una FIFO a piccoli blocchi, fermandosi alla fine di ogni blocco.

BIBLIOGRAFIA

•AA.VV. "SCB68172 VMEbus controller BUSCON", Signetics, dic.1988.

•AA.VV. "using the CY7C964 with VIC" Cypress semiconductor application note.

•AA.VV. "using VIC068A on a board without a microprocessor", Cypress semiconductor application note.

•AA.VV, "VE-ADC-2110 user's manual", Valley technolgies, Tamaqua, PA, 1991

•AA.VV. "DSP application note", Sharp Inc.

•AA.VV. "LH9124 Digital signal Processor", Sharp Inc. Rev B 3-16-93.

•AA.VV. "LH9320 DSP address generator",Sharp Inc. Rev. B 6-16-93.

•AA.VV."Ultra DSP User's manual", Valley technologies, Tamaqua, PA.maggio 1994

•Baxes "Digital image processing", Prentice-Hall, 1981.

•Bellanger "Digital Signal processing", Jhon wiley & Sons, 1984.

•Bellanger, "adaptive digital filters and sygnal analysis", Marcel Dekker, N.Y. 1987

•Bellanger,"Digital processing of signals".

•Bowen,Brown "VLSI system for digital signal processing ", Prentice-Halll, 1982

•Brigham,"The fast Fourier Transform",Prentice-Hall,1974.

•Brigham E.O.,"Fast Fourier transform and its application" Prentice-Hall,1988.

•Cooley,J.W.Tukey "an algorithm for machine calculation of complex Fourier series".Math computation (apr 1965, Vol 19, pp.297-301.

•Crochiere, Rabiner "Multirate digital signal processing", Prentice-Hall, 1983.

•DeFatta, Lucas,Hodgkiss,"Digital signal processing, a system design approach", Jhon Wiley & Sons, 1988.

•Devore "Probability & statistic for engeneering and science" Brooks/Cole 1982

•Elliot "Handbook of digital image processing", Academic Press, 1987
•G-AE subcommitte of measurement concepts,"What is the Fourier Transform?",IEEE audio and electroacustic,Giu. 1967, Vol.55.
•Gentleman,Sande "Fast Fourier Transform for fun and profit.",AFIPS Proc.1966,Fall Jount Computer Conf.,Vol.29,pp.563-678.
•Gonzales, Wintz,"Digital Image Processing", Addison-Wesley, 1987.
•Gradner "Statistica Spectral analyis - A nonprobabilities theory", Prentice-Hall,1988.
•Hamming, "digital filters, 3rd edition", Prentice-Hall 1988.
•Harris "On the use of windows for harmonic analisys with the discrete Fourier transform", proc.IEEE,Vol 66, N0.1,pp.51-83 gen.1979.
•Haykin "adaptive filter analysis", Prentice-Hall,1986
•Haykin "introduction to adaptive filters" MacMillan N.Y. 1984.
•IEEE ASSP Committee "Programs for digital signal processing ", IEEE press, 1979.
•Jackson "digital filters and signal processing", Kluwer Academic publisher, 1986.
•Jackson, "an analisys of roundoff noise in digital filters ". Sc.D. dissertation, department of electrical engeneering,Stevens institute of technology,1969.
•Jackson,"on interaction of roundoff noise and dynamic range in digital filters",Bell system tech.Vol 49,1970,pp. 159-184.
•Jackson,"roundoff-noise analysis for fixed points digital filters in cascade or parallel forms",IEEE trans. audio electroacust.,Vol AU-18, Giu.1970 pp.107,122
•Jain "Fundamentals of digital Image processing",Prentice-Hall.1989
•Kung, WhiteHouse, Kailath "VLSI and modern digital processing" Prentice-Hall,1985.
•Lim, Oppenheim "advanced topics in signal processing", Prentice-Hall.
•Liu,T.Kaneko,"Error analisys of digital filters realized with floating points aritmetic".proc IEEE Vol 57, Ott 1969,pp.1735-1747.
•Oppenheim, "application of digital signal processing", Prentice-Hall.
•Oppenheim, Shafer "digital signal processing", Prentice-Hall, 1975.

•Oppenheim,R.Schafer"Discrete time signal processing",Prentice-Hall,1989.
•Peled, Liu, "Digital signal processing:Theory design and implementation", John Wiley & Sons, 1988.
•Raberts, Mulley "Digital signal processing", Prentice-Hall, 1987.
•Rabiner L., B.Gold "theory and application of digital signal processing", Prentice-Hall 1975.
•Sohie,"implementation of fast fourier transform on Motorola's DSP 56000/56001 and DSP6002 ".
•Stanley, Dougherty, Dougherty, "Digital signal processing ", Reston Publish, 1984.
•Taylor "Digital Filters design Handbook", Marcell Dekker, N.Y. 1983.
•Wade D.Peterson "The VMEbus handbook", VMEbus international trade association, Scottsdale-Zaltbommel nov.1988
•Weinstein, A.V.Oppenheim,"A comparison of roundoff-noise in floating points and fixed points digital filter realizations".Proc IEEE Vol 57, giu.69, pp.1775-1791
•Weinstein,"quantization effect of digital filter", MIT Lincoln lab. Tech.rept.486, ASTIA DOC.DDC AD-706862,Nov.21,1969.
•Widow,Sterns "adaptive signal processing" Prentice-Hall 1985