Università degli Studi di Trieste

Dipartimento di Ingegneria e Architettura

Corso di Laurea in Ingegneria Elettronica

Realizzazione di un filtro audio digitale dinamico

"AUTOWAH"

su TMS320VC5510 DSK

Esame di Elettronica III

Progetto di:
Manuel Iurissevich

Prof.
S. Carrato

25 Giugno 2014

AUTOWAH SU TMS320VC5510 DSK

MANUEL IURISSEVICH

Sintesi

Si è realizzato un filtro digitale passabanda dinamico per applicazioni musicali. Tale filtro è chiamato generalmente chiamato AUTOWAH o *Envelope*, basandosi sulla rilevazione dell'inviluppo del segnale in ingresso. È stato realizzato su un processore TMS320C5510 DSP, montato sulla relativa scheda di sviluppo. Si è ottenuto un effetto che risponde bene pur ponendo vincoli di utilizzo molto severi.

Indice

Premessa

La tesina è allegata al progetto finale dell'esame *Elettronica III: DSP e MCU*. Lo scopo di questa è illustrare -al professore in primo luogo- le caratteristiche e il processo di sviluppo dell'applicazione (filtro), per dimostrare che lo studente ha colto, compreso e sa mettere in pratica gli insegnamenti appresi durante il corso (e i corsi propedeutici o affini).

Non si tratteranno quindi le generalità sull'elaborazione e sugli elaboratori digitali di segnale (DSP), né sulla scheda che lo monta e il suo equipaggiamento (DSK), né sull'ambiente di sviluppo, a meno che queste non siano strettamente necessarie alla comprensione della particolare applicazione. Queste informazioni saranno infatti oggetto seconda parte dell'esame e saranno incluse nella documentazione soltanto se e quando ci sarà la volontà di emancipare il progetto dall'esame e farne un realtà a sé.

Per comprendere questo testo è quindi necessario possedere -e si assumerà che il lettore le possieda- le basi di Analisi matematica, Teoria dei segnali, Elaborazione elettronica del segnale, Controllo digitale, Architettura dei calcolatori, Fondamenti di informatica (linguaggio C). L'autore è tuttavia convinto che lo stile di scrittura sia abbastanza semplice e i riferimenti bibliografici sufficientemente completi da permettere a chiunque sia realmente interessato alla materia di comprendere il testo.

1. Introduzione

1.1. **Cos'è un AUTOWAH.** L'effetto WAH (pron. uàa) consiste nello spostare tramite un pedale il timbro del suono che si sta producendo, tipicamente con una chitarra o un basso elettrici: un AUTOWAH dev'essere in grado di filtrare adeguatamente il segnale senza l'intervento umano, intuendo quando il musicista avrebbe mosso il pedale. La versatilità d questo effetto automatico non è paragonabile a quella del corrispettivo manuale (podale?) ma nella maggior parte delle situazioni risulta più che soddisfacente, oltre a permettere all'interprete all'interprete di dedicarsi ad altri aspetti importanti dell'esibizione come saltare sugli amplificatori e via dicendo.

1.2. **Perché un AUTOWAH.** Oltre a fornire quei suoni che vanno dal lamento di bambino agli ambienti sottomarini l'aspetto più interessante dell'AUTOWAH è dare risalto alle frequenze giuste al momento giusto; soprattutto suonando il basso elettrico fa molto piacere avere definizione e attacco prima, profondità e durata poi.

Dal punto di vista accademico i filtri audio sono perfetti per misurarsi con gli elaboratori di segnali digitali (dsp) e danno un risultato udibile e concreto; tra tutti l'AUTOWAH sembra il più intelligente proprio per la richiesta capacità di capire in tempo reale la volontà dell'umano e di emularne il comportamento.

1.3. **Com'è fatto un AUTOWAH.** Sono necessarii due blocchi, due filtri: uno è il passabanda che altera le componenti in frequenza del suono, l'altro è il filtro a inviluppo che determina come si sposta la frequenza centrale del passabanda. La versione analogica dell'effetto usa un diodo emettitore di luce e un fotodiodo per passare il parametro da uno stadio all'altro. La funzione non lineare dovrebbe quindi essere logaritmica, ma si è preferito comunque usare la tangente iperbolica.

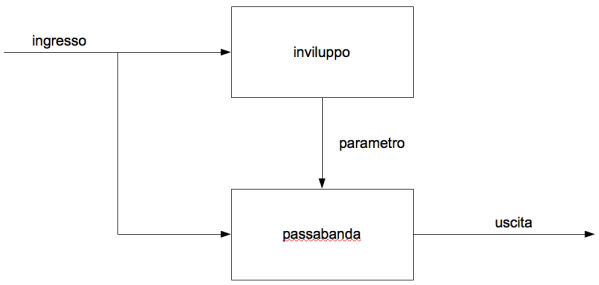

Figura 1. Schema a blocchi globale

2. Implementazione

2.1. **Entrando nel dettaglio.** L'inviluppo è a propria volta costituito da un raddrizzatore e un filtro passabasso. Si è preferito ragionare in tempo continuo per poi digitalizzare il filtro:

$$\frac{V_U}{V_R}(s) = \frac{1}{sRC+1} \qquad RC := 0.3s^{-1} \tag{1}$$

$$\frac{V_U}{V_R}(z) = \frac{bz+b}{z+a} \qquad b = \frac{T}{2\cdot RC+T} \quad a = \frac{2\cdot RC-T}{2\cdot RC+T} \tag{2}$$

dove si è posto RC := $0{,}3s^{-1}$ (definisce il ritardo dell'effetto) e si sono calcolati con la trasformata di Tustin [1] i parametri del filtro in tempo discreto. RC determina il ritardo dell'effetto. Figura(2).

Per il passabanda si è usata la forma

$$F(z) = A^2\frac{z^2-1}{z^2-2A\cos(\Omega_0)z+A^2} \qquad A := 0.9 \tag{3}$$

Il passabanda ha due zeri in ± 1 e due poli complessi coniugati che si aprono per aumentare la frequenza centrale [2]. Figura(3).

Il parametro che collega i due filtri è $c = \cos(\Omega_0)$: per ottenerlo si definiscono i due parametri `FORMA` e `GUADAGNO`, che potrebbero essere due manopoline in un pedale o, in un effetto (software) virtuale, due barre di scorrimento o in qualche modo due variabili accessibili all'utente. Figura(4)

Quindi

$$\begin{aligned} f_0 = F\cdot f_m\cdot[\tanh(G\cdot[x-0,1])+1]+F\cdot f_q \\ f_q := 20Hz \quad f_m := 1100Hz \end{aligned} \tag{4}$$

e finalmente

$$c = \cos\left(2\pi\frac{f_0}{f_c}\right), \qquad f_c = 24kHz \tag{5}$$

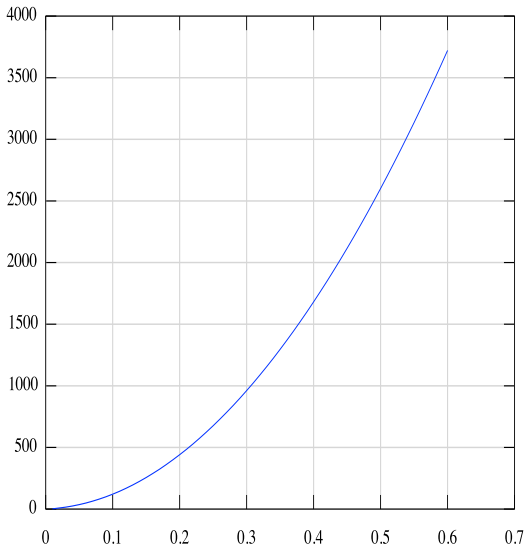

FIGURA 2. Il ritardo (in campioni) aumenta con RC

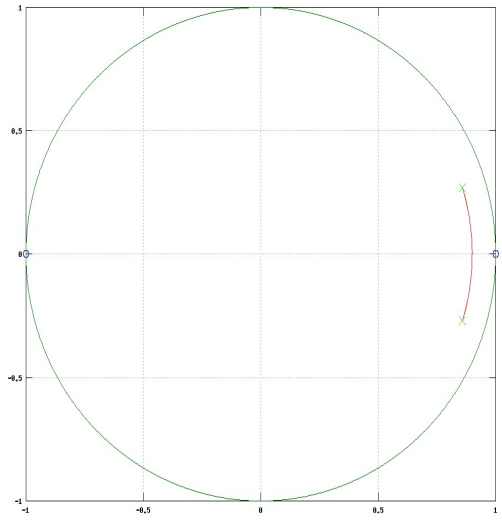

FIGURA 3. Zeri e poli del passabanda nel piano Z

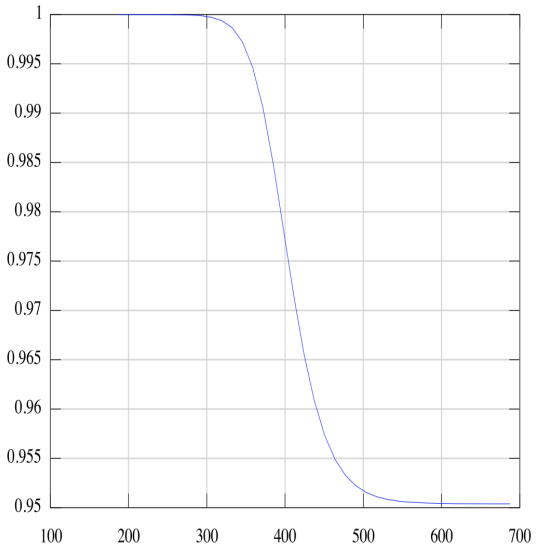

FIGURA 4. Il grafico di $\cos(\Omega_0)$ in funzione dell'*inviluppo*

2.2. **La tavola numerica.** Un dsp è fatto per calcolare ad altissima velocità somme e moltiplicazioni; non è adatto a calcoli impegnativi quali sono le funzioni trigonometriche: non gli si può quindi chiedere di calcolare -men che meno in pochi cicli di clock- la funzione (4). La soluzione più adatta a risolvere il problema è una semplice tavola numerica calcolata con Octave e fornita pronta al dispositivo: quello che si chiede al processore è di cercare il valore x nella prima colonna, che si troverà nella riga i-esima, e restituire il valore y alla i-esima riga della seconda colonna. La ricerca di x è semplificata per il fatto che la distanza d tra i valori della prima colonna è costante e non occorre quindi scorrere tutto il vettore o utilizzare complessi (e lenti) algoritmi di ricerca: sapendo che basta calcolare $x_i = x_0 - i \cdot d$ e fornire in uscita $i = (x - x_0)/d$. Se $x < x0$, basta restituire $y = 1$ mentre quando $x > x_{max}$ si restituisce $y = 0.950390$.

x	i	y
< 187	n.d.	1
200	1	0.99999
...	...	...
398.03	16	0.97819
411.18	17	0.97146
490.13	18	0.96552
...	...	...
424,34	37	0.95040
> 687.50	n.d.	0.95039

TABELLA 1. Estratto della tavola numerica

2.3. **Il filtro a inviluppo.** Antitrasformando la FORMULA(2) si ottiene

$$y_n = b \cdot |x_n| + b \cdot |x_{n-1}| - a \cdot y_{n-1} \tag{6}$$

dove x è il segnale d'ingresso e y il suo inviluppo. Si nota che occorre salvare i valori appena calcolati di x e di y per i cicli successivi. Non resta che calcolare $c = f(y)$ con la lut.

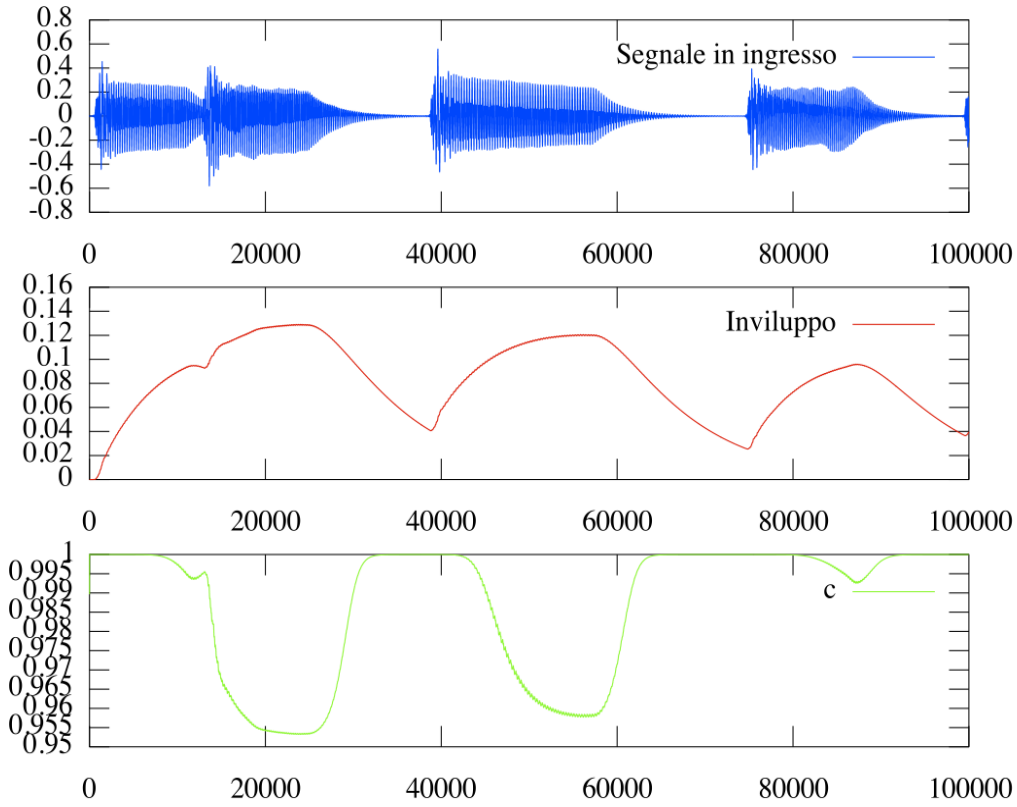

FIGURA 5. Esempio di *ingresso*, *inviluppo* e c

I grafici di FIGURA(5) sono stati ottenuti in fase di simulazione utilizzando in ingresso un audiowave: i valori di questo e del suo inviluppo sono alterati rispetto quelli nel progetto finale in tempo reale.

2.4. **Il filtro passabanda.** Antitrasformando la FORMULA(3) si ottiene l'equazione alle differenze

$$y_n = A^2 \cdot x_n - A^2 \cdot x_{n-2} + 2A \cdot c \cdot y_{n-1} - A^2 \cdot y_{n-2} \tag{7}$$

dove y è il segnale d'uscita dal passabanda.

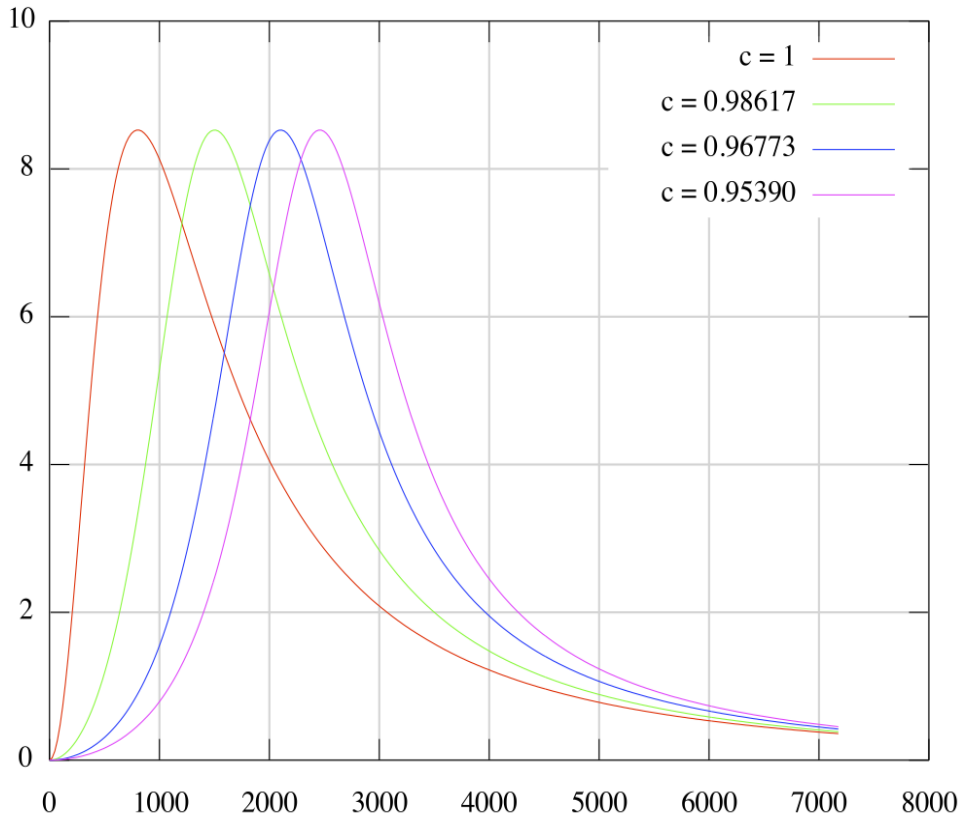

FIGURA 6. La risposta in frequenza del filtro passabanda

3. Realizzazione

Per realizzare questo progetto sono stati usati i modelli forniti da V. Toffoli, in particolare `test_aic23` per la configurazione del *codec* [3].

3.1. **Preambolo.** Il foglio `main.c` inizia con una breve descrizione di ciò che fa il programma. Come suggerito dal creatore del modello, ogni modulo della scheda ha il suo `include`. Oltre a quelli strettamente necessari si usano qui i moduli per comunicare con il *codec*, comandare i *led*, leggere lo stato degli interruttori. Sono poi presenti le librerie proprie del progetto AUTOWAH: `CONTROLLI.H` contiene i parametri a cui l'utente finale dovrebbe aver accesso; `lut.h` contiene la tavola numerica e variabili necessarie al suo utilizzo; `filtro.h` contiene le rimanenti variabili globali.

Segue la configurazione del *codec*: questa consiste nella definizione di dieci registri che regolano l'interfacciamento del DSP con gli ingressi e le uscite analogiche. Riassumendo, si impone al *codec* escludere l'ingresso audio in linea e usare quello del microfono, con *boost*; le uscite sono abilitate, sia cuffie che in linea [4].

3.2. **Main.** Si entra nel `main()`.

Dopo le inizializzazioni della scheda, per segnalare che l'avviamento è andato a buon fine un *led* lampeggerà un paio di volte.

Viene quindi avviato il *codec* con la configurazione descritta precedentemente. Si imposta la frequenza di campionamento a 24kHz e la si passa al *codec*.

Si preparano la tavola numerica e il filtro basandosi sui parametri definiti dall'utilizzatore.

3.3. **Inizio del ciclo.** Il ciclo infinito inizia con le procedure di acquisizione dei campioni sui due canali. L'elaborazione inizia con registrare il campione appena acquisito nel buffer d'ingresso, dopo averlo scalato per il guadagno in ingresso (di defalut = 0.15).

La variabile *corr* servirà a correggere il volume in uscita dopo i vari filtraggi.

Se la distorsione è abilitata, si distorce moltiplicando ancora il campione attuale per il guadagno del distorsore (di default = 10'000), saturando e dividendo di nuovo per ritornare entro valori ascoltabili. Si aggiorna *corr*.

Nelle prossime righe si raddrizza (*rad*) il segnale (valore assoluto) e si calcola l'inviluppo (*env*). Dalla Formula(6) si nota che bisogna salvare il valore precedente sia di *rad* che di *env*, che sono quindi due vettori di due elementi: *vec*chio e *nuo*vo. Al prossimo ciclo non si scambiano i valori ma i puntatori.

Il valore dell'inviluppo è usato per accendere una quantità proporzionale di *led* in maniera da avere anche una sorta di monitor sul segnale che si sta elaborando, per eventualmente aggiustare il guadagno.

3.4. **Wahwah.** Se sono abilitati l'autowah o il phaser, si inizia chiama la Lut: nel primo caso si cerca il valore nella tabella, nel secondo lo si estrae scorrendo periodicamente la tabella in su e in giù.

Con il valore opportuno di c si può quindi filtra con il passabanda, per poi aggiornare i valori nei buffer d'ingresso, d'uscita e di *corr*.

Se invece il passabanda è disabilitato, si copia l'ingresso ing_0 nell'uscita y_0.

3.5. **Fine del ciclo e del main.** Si cambiano -come scritto sopra- i puntatori *vec*chio e *nuo*vo.

Si legge lo stato degli interruttori.

Si moltiplica l'uscita per il volume (eventualmente corretto) e si procede con le scritture verso il *codec*.

3.6. **NuovaLut.** Sono necessarie tre funzioni per gestire la Lut.

La procedura `inizNuovaLut()` legge il guadagno da `CONTROLLI.H` e con questo trasla la curva di FIGURA(4) in avanti sull'asse delle ascisse, per far sì che l'effetto funzioni anche con strumenti di potenza diversa. In *contropasso* si salva l'inverso della differenza d tra due elementi della prima colonna della tabella.

Della funzione di ricerca nella tavola numerica è stata si è già scritto nella sezione dedicata.

La funzione `perLut()` è uno scorrimento periodico della tabella: questo ha permesso, in fase di debug, di estrarre i valori in uscita dalla tavola quando i valori in ingresso non erano ancora stati aggiustati con la traslazione. È stata lasciata nel testo definitivo perché già di per se stessa è un effetto audio completo.

3.7. **Filtro.** Qui si preparano i coefficienti dei due filtri in base ai valori impostati di $freq$, che non è una frequenza ma il parametro che la identifica (Es: $freq = 3$ pone $f_c = 24kHz$), e di RC definito in `CONTROLLI.H`.

Altre inizializzazioni di indici, contatori e vettori, con la chiamata alla procedura `inizVett()`.

La funzione `tastino()` serve se si usano interruttori monostabili, che non rimangono nelle due posizioni *acceso/spento* ma possono essere soltanto premuti come il tasto delle penne biro o appunto quelli degli effetti audio, per poi ritornare da soli in posizione di riposo. Gli interruttori della scheda usata possono essere bloccati in posizione $'0'$, quindi volendo si può evitare di usare questa funzione.

Ad esempio, invece di scrivere

```
wah = tastino(1);
```

si scriverà direttamente

```
wah = !DSK5510_DIP_get(1);.
```

Nelle prove effettuate in laboratorio i tasti sono stati usati come se fossero monostabili -basta non premere fino in fondo; è stata quindi usata la funzione `tastino()` e non si sono osservati problemi di rimbalzo.

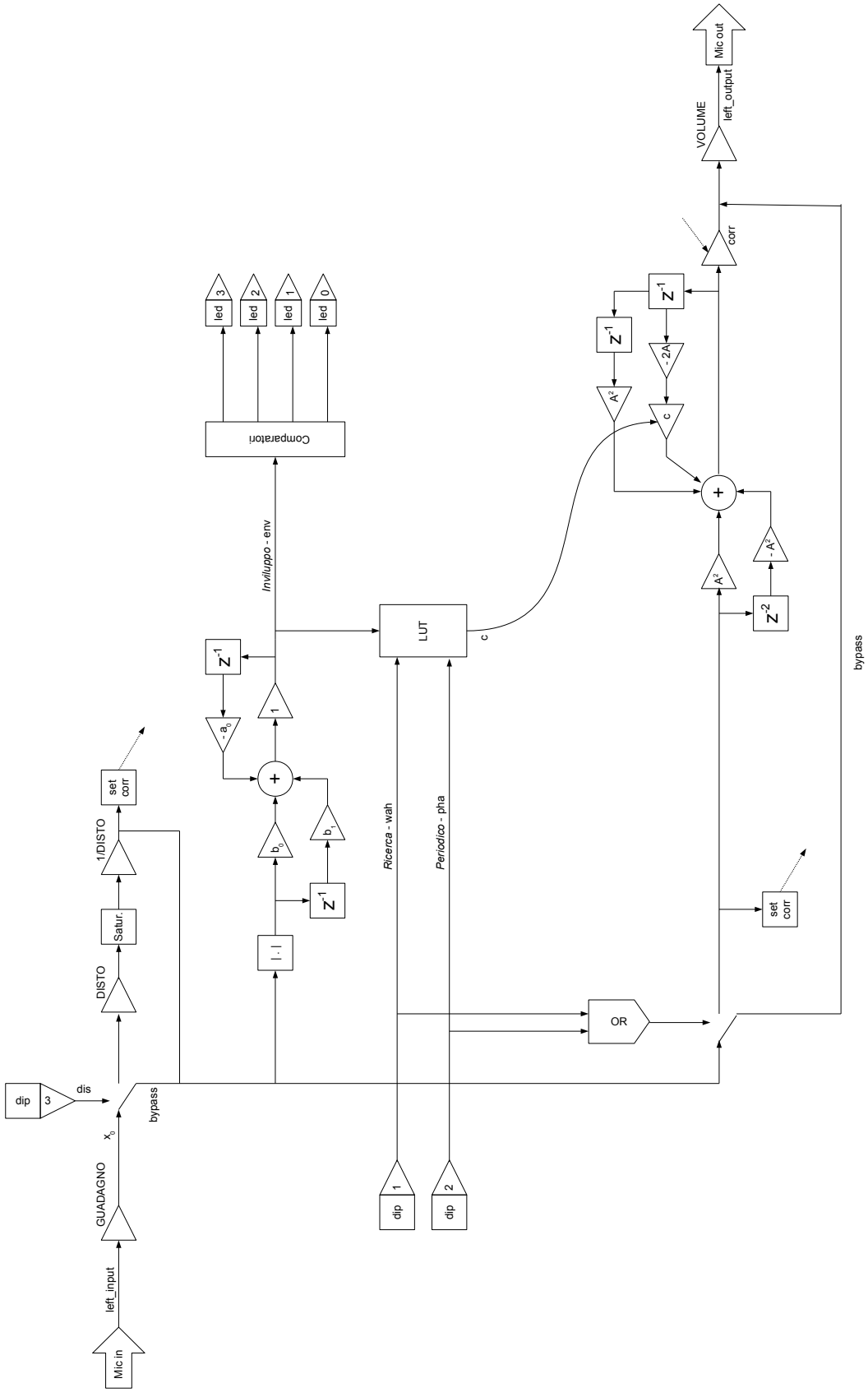

FIGURA 7. Schema a blocchi completo

4. Conclusioni

4.1. **Riassunto.** È stato realizzato un filtro digitale passabanda dinamico per applicazioni musicali.

La progettazione si è basata sia sull'analisi degli schematici analogici dei dispositivi in commercio, sia sull'applicazione direttamente in tempo discreto dei filtri studiati durante i corsi di *Elaborazione elettronica del segnale* e *Controllo digitale*. Dalla progettazione nel dominio della frequenza si sono ottenuti poi i coefficienti necessari a operare nel dominio del tempo.

A questa è poi seguita una fase di simulazione in Octave (clone libero di MatLab) e in C con XCode, entrambe *non* in tempo reale.

Gran parte del tempo è stato impiegato per trovare ad orecchio il valore corretto di certi parametri quali ad esempio RC o la funzione della Lut. Alla fine si è ottenuto un effetto che risponde bene pur ponendo vincoli di utilizzo molto severi: bisogna regolare il volume in uscita dallo strumento e il GUADAGNO, pizzicare la corda con abbastanza intensità da attivare l'effetto, ma non abbastanza da distorcere il suono (fatte le due regolazioni precedenti, non si dovrebbero incontrare grossi problemi). Se si riesce a sottostare a questi pochi ma severi vincoli, il dispositivo fa un bell'effetto che potrebbe competere con i più economici multieffetto commerciali.

4.2. **Sviluppi futuri.** Il progetto va sicuramente ottimizzato sotto ogni aspetto, a partire da quelli più evidenti legati al suono, per finire con quelli più tecnici legati al codice, all'esecuzione e alla memoria.

Si potrebbe aggiungere un *compressore logaritmico* in ingresso per uniformare l'ampiezza del suono così che il passabanda si sposti quasi allo stesso modo su note *urlate* piuttosto che *sussurrate*. In alcuni pedali commerciali si usa comunque una manopola che aggiusta la *sensibilità* dell'effetto [8].

Si potrebbe assicurarsi che il dispositivo esegua in parallelo più operazioni possibile, scrivendo in codice assemblativo almeno le somme e le moltiplicazioni [5].

5. Appendice

5.1. **Elaborazione digitale del segnale.** Se un segnale è digitale, è scontato che sia tempo-discreto. Nella teoria dei segnali si trattano però segnali tempo-discreto *non* digitali.

Digitale si riferisce al codominio di una funzione, *tempodiscreto* al dominio.

Ponendo un segnale impulsivo all'ingresso di un filtro lineare tempo-invariante H, si avrà all'uscita di questo il segnale h_n, detto *risposta all'impulso* del filtro. Sia x_n un segnale tempo discreto qualunque posto all'ingresso di un filtro lineare tempo-invariante con risposta all'impulso h_n, si osserverà in uscita il segnale

$$y_n = \sum_i x_{n-i} \cdot h_i = x_n * h_n \tag{8}$$

questa operazione è detta *convoluzione*.

Se la risposta all'impulso è finita, cioè nulla da un certo istante in poi, il filtro è appunto detto *filtro a risposta impulsiva finita*, acronimo FIR (dall'inglese). Se è infinita, il filtro si dice IIR e l'operazione di convoluzione diventa

$$y_n = \sum_i x_{n-i} \cdot b_i + \sum_i y_{n-i} \cdot a_i. \tag{9}$$

I filtri FIR sono sempre stabili; i filtri IIR possono diventare instabili a causa della retroazione ma a parità di prestazioni necessitano di molti meno coefficienti, quindi memoria e operazioni.

Per rendere più semplice l'analisi dei filtri e le operazioni connesse, si introduce la trasformata Z

$$X(z) = \sum_{i=0}^{\infty} x_n \cdot z^{-n}, \qquad z = A \cdot e^{j\Omega} \tag{10}$$

z è quindi un numero del piano complesso, dotato di modulo A e fase Ω (argomento), nonché di parte reale $A \cdot \cos(\Omega)$ e immaginaria $A \cdot j \sin(\Omega)$.

In questo modo la convoluzione $x_n * h_n$ si può scrivere più semplicemente come prodotto $X(z) \cdot H(z)$. Una cascata di più filtri si scrive $H_1(z) \cdot H_2(z) \cdot H_3(z) \cdot ...$

5.2. **Filtri.**

• Semplice FIR passabasso:

$$H(z) = \frac{1 + z^{-1}}{2} \tag{11}$$

quindi

$$Y(z) = \frac{1 + z^{-1}}{2} X(z) = \frac{1}{2} X(z) + \frac{1}{2} z^{-1} X(z) \tag{12}$$

da cui si ottiene l'equazione alle differenze

$$y_n = \frac{1}{2} x_n + \frac{1}{2} x_{n-1} \tag{13}$$

prende il valore attuale, il valore precedente e fa la media.

• Semplice IIR passabasso:

$$H_{basso}(z) = \frac{1 - \alpha}{2} \cdot \frac{1 + z^{-1}}{1 - \alpha z^{-1}} \tag{14}$$

da cui si ottiene l'equazione alle differenze

$$y_n = \frac{1-\alpha}{2}x_n + \frac{1-\alpha}{2}x_{n-1} + \alpha y_{n-1} \tag{15}$$

per la stabilità dev'essere $|\alpha| < 1$.

- Semplice IIR passalto:

$$H_{alto}(z) = \frac{1+\alpha}{2} \cdot \frac{1-z^{-1}}{1-\alpha z^{-1}} \tag{16}$$

da cui si ottiene l'equazione alle differenze

$$y_n = \frac{1+\alpha}{2}x_n - \frac{1+\alpha}{2}x_{n-1} + \alpha y_{n-1} \tag{17}$$

per la stabilità dev'essere $|\alpha| < 1$.

- Semplice IIR passabanda:

$$H_{banda}(z) = \frac{1-\alpha}{2} \cdot \frac{1-z^{-2}}{1-\beta(1+\alpha)z^{-1}+\alpha z^{-2}} \tag{18}$$

da cui si ottiene l'equazione alle differenze

$$y_n = \frac{1-\alpha}{2}x_n - \frac{1-\alpha}{2}x_{n-2} + \beta(1+\alpha)y_{n-1} - \alpha y_{n-2} \tag{19}$$

che è la versione generale del filtro usato nel progetto.

Qui

$$\beta = \cos(\omega_{centrale}), \qquad \frac{2\alpha}{1+\alpha^2} = \cos(Banda). \tag{20}$$

Se si pone $\alpha = 0.6$ e $\beta = 0.5$ e si applica l'algoritmo si vede che per approssimarsi allo zero, la risposta all'impulso impiega circa 40 campioni: un filtro FIR con lo stesso effetto avrebbe bisogno di 40 coefficienti contro i 5 del filtro IIR. Senza contare che per spostare la risposta in frequenza, basta cambiare un coefficiente dell'IIR mentre bisognerebbe cambiare tutti i quaranta coefficienti del FIR.

5.3. **Mac.** Qualunque sia il filtro da implementare è ormai chiaro che il nocciolo del *signal processing* è *moltiplicare* e *sommare*. La caratteristica che più distingue i *signal processors* è appunto il *moltiplicatore hardware*:

```
mac *ar2+, *ar3+, a
```

è l'istruzione per moltiplicare due fattori e sommare il risultato in un accumulatore, in un singolo colpo di *clock*.

5.4. **Elaboratori digitali di segnali.** I DSP hanno quindi:

- hardware dedicato a sveltire le operazioni
- hardware dedicato alla generazione degli indirizzi
- indirizzamento circolare
- spazi di memoria separati per dati e istruzioni: Harvard
- molti bus separati per prelievo istruzioni, lettura e scrittura dati
- uso intensivo della pipeline.

5.5. **C5510 DSP.** Alcune caratteristiche particolari del C5510:

- memoria unificata per dati e istruzioni
- bus multipli: 1 prelievo istruzioni, 3 lettura dati, 2 scrittura dati
- due mac $17 \times 17 + 40$ bit
- due generatori di indirizzi con otto registri ausiliari
- pipeline a sette stadi
- unità funzionali multiple parallele: superscalare
- clock 200MHz.

5.6. **VC5510 DSK.** La scheda che lo monta offre inoltre:

- codec stereo AIC23
- memoria RAM dinamica sincrona
- memoria flash: 512KB
- quattro interruttori e quattro led
- CPLD Altera
- connettori per le espansioni
- emulatore JTAG
- collegamento USB con il computer

5.7. **Codec AIC23.** Il codec è un *cod*ificatore e *dec*odificatore di segnali: campiona segnali analogici e li converte in dati digitali che il processore può elaborare; poi li riconverte a elaborazione terminata.

Il codec AIC23 ha due ingressi e due uscite analogici: microfono e line-in, di cui solo uno alla volta può essere abilitato; cuffie e line-out (possono essere entrambe attive contemporaneamente). Formato: minijack 3,5mm.

La frequenza di campionamento è un sottomultiplo di 12MHz.

Per scegliere quali ingressi e uscite abilitare, per scegliere la frequenza di campionamento, il volume e altre impostazioni, si usano dieci registri.

Elenco delle figure

Elenco delle tabelle

Riferimenti bibliografici

[1] G. Fenu, 2009
Controllo digitale: Progetto del controllore, dispense del corso.
[2] S. K. Mitra, G. Ramponi, 2001
Elaborazione elettronica del segnale: Ch4(5)
dispense del corso.
[3] V.Toffoli, 2007
Realizzazione di una serie di progetti introduttivi all'utilizzo di un processore TMS320C5510
Tesina per l'esame: Elettronica III - DSP e MCU.
[4] Texas Instruments Inc., 2004
TLV320AIC23B Data Manual
(tlv320aic23b).
[5] Texas Instruments Inc., 2001
TMS320C55x DSP Programmer's Guide
(spru376a).
[6] Spectrum Digital Inc., 2002
TMS320VC5510 DSK Tecnical Reference
(dsk5510_techref).
[7] Bateman, Paterson-Stephens, 2002
The DSP Handbook
Prentice-Hall.
[8] Ed Friedland, 2010
MXR Bass Envelope
Bass Whisperer TV, Licenza YouTube standard.