Consiglio Nazionale delle ricerche

ISTITUTO DI ELABORAZIONE DELLA INFORMAZIONE

PISA

Editore/Compilatore Grafico: uno strumento per la definizione di algoritmi di sintesi sonora

L. Tarabella, G.Bertini

Nota interna B4 - 30

Ottobre 1991

Editore/Compilatore Grafico: uno strumento per la definizione di algoritmi di sintesi sonora

L. Tarabella, G.Bertini

Nota interna B4 - 30

Ottobre 1991

PROGETTO FINALIZZATO
SISTEMI INFORMATICI E CALCOLO PARALLELO

SOTTOPROGETTO 2
Coordinatore Prof. Franco Denoth
Processori Dedicati

L.Tarabella [1], G.Bertini [2]

Editore/Compilatore Grafico: uno strumento per la definizione di algoritmi di sintesi sonora

N. R/2/54 **Agosto 1991**

Rapporto Interno

1 - Istituto CNUCE, Consiglio Nazionale delle Ricerche, Via S.Maria 36 - Pisa

2 - Istituto IEI, Consiglio Nazionale delle Ricerche, Via S.Maria 46 - Pisa

Indice

Editore/Compilatore Grafico: uno strumento per la definizione di algoritmi di sintesi sonora

L.Tarabella [1], G.Bertini [2]

1 - Istituto CNUCE, Consiglio Nazionale delle Ricerche, Via S.Maria 36 - Pisa
2 - Istituto IEI, Consiglio Nazionale delle Ricerche, Via S.Maria 46 - Pisa

Sommario

In questo lavoro viene presentato il progetto di una speciale applicazione definita *Editore/Compilatore Grafico*, atto alla definizione in forma grafica di algoritmi di sintesi e di elaborazione di segnali audio. Tale lavoro si inserisce nell'ambito del sottoprogetto *Processori Dedicati* del *Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo*, ed in particolare per quello che riguarda l'argomento *Progetto di sistemi programmabili da connettere a personal computer per il trattamento di segnali in banda acustica*, portato avanti congiuntamente nel Reparto di Elaborazione dei Segnali ed Immagini dell'IEI/CNR e nel Reparto di Informatica Musicale del CNUCE/CNR.

L'obiettivo è quello di dare un contributo allo sviluppo di stazioni di lavoro basate su calcolatori *personal*, da impiegare nel settore del trattamento digitale di segnali audio e musicali, per soddisfare quelle esigenze della ricerca, della didattica e della composizione musicale assistita da calcolatore, non ancora soddisfatte dalle apparecchiature comunemente reperibili in commercio. L'Editore Grafico è stato realizzato in versione prototipale con programmi scritti in linguaggio C su macchine Macintosh. Con l'uso del mouse è possibile posizionare sulla finestra di lavoro i simboli tipici (generatori, inviluppi, linee di ritardo, etc.) presi da una palette, e

collegarli con strumenti del tipo lapis, gomma etc. L'Editore comprende un analizzatore sintattico e di consistenza dell'algoritmo disegnato e ne rende possibile la simulazione producendo la forma d'onda ed il suono corrispondente. Attualmente è in fase di realizzazione la funzione di traduzione dell'algoritmo disegnato nel corrispondente codice macchina di un prefissato microprocessore DSP.

In questa relazione, dopo aver dato un cenno alle problematiche relative alle procedure per l'implentazione di alcune classi di algoritmi di sintesi e di elaborazione di segnali sonori, verranno esposte le caratteritische principali e le funzionalità disponibili della versione attuale dell'Editore/Compilatore Grafico.

Introduzione

L'evoluzione della tecnologia digitale ha consentito la progressiva sostituzione dei sistemi analogici portando la risoluzione di problemi di trattamento dei segnali dal dominio del tempo continuo a quello del discreto. Nel passato sono stati realizzati sistemi speciali per il trattamento numerico dei segnali con vari livelli di prestazioni e di costo dipendentemente dai loro settori di impiego: medicale, militare, ricerca, etc...

In seguito alla disponibilità di personal computer con potenzialità di calcolo sempre crescenti e di microprocessori specializzati all'elaborazione numerica di segnali (DSP, Digital Signal Processing), l'orientamento attuale della tecnologia è quello di sviluppare moduli di calcolo che svolgono funzioni di co-processori programmabili, per rendere più efficaci l'uso degli stessi personal computers per l'elaborazione numerica in tempo reale di segnali di varia natura: voce, immagini e altre grandezze fisiche.

Questi moduli sono costituiti da schede provviste di uno o più processori DSP, memorie e sezioni di conversione AD e DA, e vengono utilizzate in varie applicazioni di misura e di controllo industriale. Di solito le case costruttrici di tali schede forniscono i relativi pacchetti di sviluppo e librerie sempre più evoluti per facilitare lo sviluppo di applicazioni tipicamente costituite da operazioni di acquisizione ed elaborazione (DFT, filtraggi, etc.).

Alcuni tipi di schede, opportunamente equipaggiate con circuiteria di conversione dalle caratteristiche tipiche della banda audio di alta fedeltà, cominciano ad essere proposte anche per l'impiego nel settore audio professionale, e per applicazioni di produzione musicale assistita da calcolatore. La caratteristica principale di questo approccio è la programmabilità: è possibile infatti caricare su queste schede e fare eseguire in tempi successivi o anche contemporaneamente, algoritmi relativi a diverse tecniche di sintesi, dove per *algoritmo di sintesi* (concetto ripreso più ampiamente nel seguito) si intende un modello matematico che tradotto in programma eseguito iterativamente, produce i valori numerici corrispondenti ai valori istantanei del segnale audio desiderato, secondo un prestabilito tasso di campionamento.

Questa versatilità, indispensabile in applicazioni di ricerca e di composizione musicale creativa che consente fra l'altro anche un decisivo allungamento dei tempi di obsolescenza delle apparecchiature unitamente ad una maggiore consapevolezza del fenomeno acustico, non è ugualmente offerta dagli

apparati commerciali dello standard MIDI (Musical Instrument Digital Interface) caratterizzati come si sa da una sola (o al massimo due) tecniche di sintesi prefissate dalla casa costruttrice, e non più modificabili se non nella scelta dei valori parametrici. D'altro canto la letteratura in proposito propone alcune decine di tecniche di sintesi con svariate modalità di combinazioni, e per di più le attività di ricerca in questa direzione sono in continua evoluzione grazie anche alla aumentata potenzialità di questi nuovi supporti di calcolo [1].

Nella situazione attuale esiste dunque la necessità di strumenti che consentano la programmabilità con livelli di sofisticazione più alti possibile, evitando perciò le difficoltà non sempre desiderabili della programmazione in assembler sia da parte di utenti (tipicamente musicisti) giustamente poco disponibili e non interessati agli aspetti più tecnici dell'argomento, sia per i ricercatori che lavorano per sviluppare algoritmi di sintesi nella preparazione di opportune librerie. A riprova di ciò, alcuni centri di ricerca nel settore e ambienti operanti nella didattica musicale , sono interessati a tali argomenti [2].

Oggetto di questa nota è la descrizione del progetto e della realizzazione prototipale di uno strumento atto alla definizione di algoritmi di sintesi ed elaborazione di segnali audio in forma grafica con l'uso delle tipiche simbologie e funzionalità presenti in tutti i personal computer: mouse, palette di simboli, etc...

Nella sua forma finale lo strumento dovrebbe offrire dunque la possibilità di progettare in maniera grafica "oggetti" utilizzabili per la definizione di segnali con determinate caratteristiche timbriche da affiancare alle altre informazioni strutturali per agevolare il lavoro della produzione musicale assistita da calcolatore.

Essendo la materia trattata di interesse sia per il *musicista informatico* sia per lo specialista del suono, verranno inizialmente richiamati i concetti di base utilizzati per la descrizione di tecniche di sintesi ed di elaborazione di segnali audio in ambito musicale.

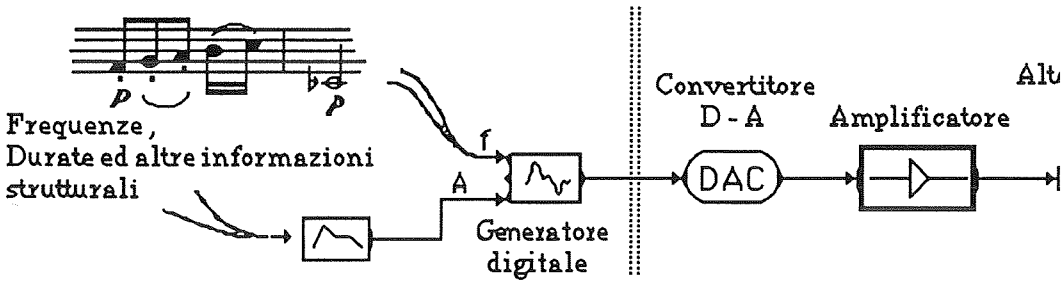
1 Il trattamento digitale di segnali in banda audio

Se fino agli inizi del secolo le fonti sonore musicali sono stati gli strumenti stessi o la voce, negli ultimi settanta anni si è aggiunta una nuova sorgente sonora: l'*altoparlante*, la cui rilevanza sta nella generalità del suo funzionamento. Questo strumento è in grado di produrre delle onde sonore a seguito di una variazione di pressione dell 'aria circostante, provocata delle vibrazioni di un "cono di carta" solidale col movimento di una bobina immersa in un campo magnetico. Il movimento del cono in funzione del tempo, e quindi la risultante variazione di pressione dell'aria, sono determinate dall'andamento della tensione elettrica V(t) applicata alla bobina.

Concepito inizialmente per *riprodurre* un segnale prelevato da un microfono e/o registrato su supporto magnetico, in un secondo tempo si pensò di utilizzare tale trasduttore come elemento terminale di una catena che vedeva l'inizio in "generatori di segnali elettrici" e quindi utilizzato in apparati per *produrre* segnali acustici/musicali "artificiali" (i cosiddetti sintetizzatori elettronici).

A quel punto perciò il problema di generare una determinata variazione di pressione dell'aria nel tempo, si era tramutato in quello di generare una adeguata variazione di tensione elettrica: problema tipico dell'elettronica risolto di volta in volta in modi diversi dipendentemente dai mezzi e dalla tecnologia a disposizione. Se infatti per molti anni gli strumenti musicali elettronici si sono basati su tecnologia analogica (oscillatori RC, VCO, VCA, filtri LC, etc...), con l'avvento della tecnologia digitale, il problema della generazione dei segnali, e cioè della sintesi, si è spostato verso lo studio di modelli matematici e della loro attualizzazione in opportune procedure di calcolo che producono come risultato una successioni di numeri rappresentanti la forma d'onda desiderata. Poiché alla parte analogica è lasciata, sostanzialmente, la sola funzione di amplificazione e, in parte, quella di conversione dei segnali, mentre tutta la parte che si occupa della sintesi è ottenuta con tecniche numeriche, sono intuibili i vantaggi che ne derivano ad es.: maggiore flessibiltà e accuratezza nella determinazione delle forme d'onda dei segnali , stabilità, ripetibilità etc...

Nella figura seguente è riportato uno schema in cui sono rappresentati alcuni elementi funzionali essenziali, e il flusso delle informazioni principali di un semplice ma completo strumento musicale digitale.



Sezione digitale Sezione analogica

E' sufficiente per il momento qui dire che nella "sezione digitale" viene concentrata tutta la parte di calcolo numerico attuato tramite computer ed altri apparati digitali che, ricevendo in ingresso le informazioni prettamente musicali, opportunamente codificate, genera la sequenza dei campioni che dopo essere stati applicati alla sezione di conversione analogica e di amplificazione vanno a costituire il segnale acustico. Per quello che riguarda il tipo e la codifica delle informazioni musicali da applicare in ingresso agli apparati digitali, basta qui dire che sono state approntate diverse metodologie di volta in volta adatte alle varie situazioni.

La parte computazionale prima definita con l'espressione "algoritmi di sintesi", come già introdotto, viene realizzata seguendo delle metodologie, opportunamente messe a punto per produrre suoni con attributi e caratteristiche timbriche desiderate.

Come si sa ogni suono è caratterizzato da una determinata legge (generalmente periodica) di oscillazione le cui principali caratteristiche "percettive" definibili in ambito musicale sono: l'altezza, il volume, il timbro, la spazialità ed altre ancora. Tranne che al timbro, a queste caratteristiche sono associabili delle precise grandezze fisiche che intervengono come parametri che specificano l'onda di pressione e la funzione che la rappresenta con particolari leggi di dipendenza [3]

.

La caratteristica timbrica di un suono, infatti, non è facilmente quantizzabile e associabile a determinate grandezze fisiche. In generale sappiamo che il timbro dipende fortemente dal contenuto spettrale del segnale, (ma non unicamente) e quindi dalla forma d'onda dell'oscillazione, dal modo con cui questa si evolve nel tempo e da molti altri fattori dipendenti dalla sorgente e dall'ambiente. Utilizzando anche concetti derivati dall'analisi in serie di Fourier, è stato accertato che il timbro di un suono, nell' intervallo in cui si possa considerare stazionario, può essere messo in relazione diretta con le sue componenti parziali, in questo caso dette

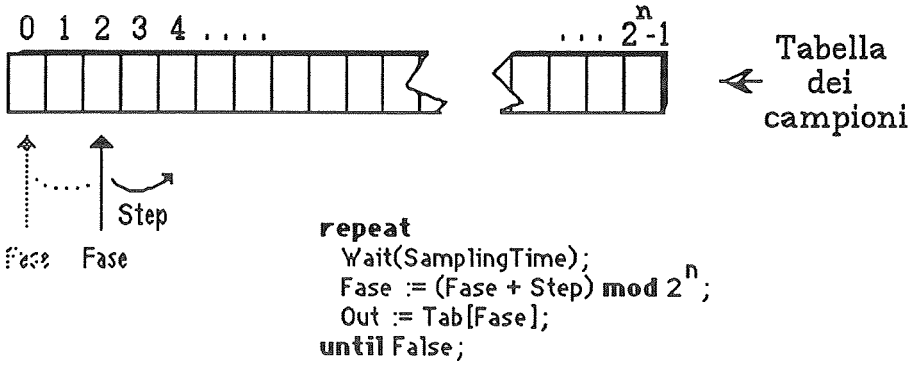
armoniche: più componenti armoniche sono presenti, e più il suono risulta acusticamente ricco e "brillante".

Per tutti i problemi di quantizzazione e discretizzazione (numero di bit, tasso di campionamento, etc.) dei valori frequenziali e di ampiezza delle funzioni, si rimanda ai numerosi testi della letteratura [4].

Come verrà meglio detto nel seguito le diverse tecniche di sintesi e di elaborazione di suoni (Additiva, Modulazione di Frequenza, Wave-shaping, Filtraggio digitale, etc..) si contraddistingono per i particolari attributi timbrici che conferiscono ai suoni prodotti ed usano dei componenti fondamentali sui quali è utile fare delle considerazioni. Si tratta dell'oscillatore digitale e del generatore di inviluppo.

1.1 L'oscillatore digitale a lettura tabellare

Poiché in generale non è possibile (e comunque non conveniente) calcolare in forma diretta e in tempo reale i punti di funzioni periodiche in banda audio e ad alta fedeltà (es. la funzione seno) con le potenzialità di elaborazione offerte dai normali sistemi di calcolo, si usa di solito il metodo della lettura tabellare. Supponendo che una tabella di lunghezza prefissata sia stata caricata con i valori di una funzione seno opportunamente campionata, il funzionamento dell'oscillatore può essere così rappresentato:




```
repeat
  Wait(SamplingTime);
  Fase := (Fase + Step) mod 2^n;
  Out := Tab[Fase];
until False;
```

L'oscillatore è realizzato tramite una procedura ciclica che ad ogni iterazione, aggiorna un puntatore di lettura alla tabella (*Fase*) di un incremento costante (*Step*). nella procedura sopra riportata in linguaggio Pascal-like, l'istruzione indicata con "wait(SamplingTime);" e' naturalmente metaforica, ed indica una serie di No-OPeration o di altre operazioni che e' possibile eseguire nell'intervallo di tempo tra la generazione di due campioni successivi. Tale intervallo corrisponde al **Tempo**

di Campionamento prestabilito.

Il valore della variabile *Step* determina la frequenza alla quale viene generata la forma d'onda per un dato Sr (Sampling rate) ed una data lunghezza di tabella secondo la relazione:

$$\text{frequenza} = \textit{Step} * \frac{\text{Sr}}{(\text{lung. tabella})}$$

da cui

$$\textit{Step} = \frac{(\text{freq.})^{*}(\text{lung. tabella})}{\text{Sr}}$$

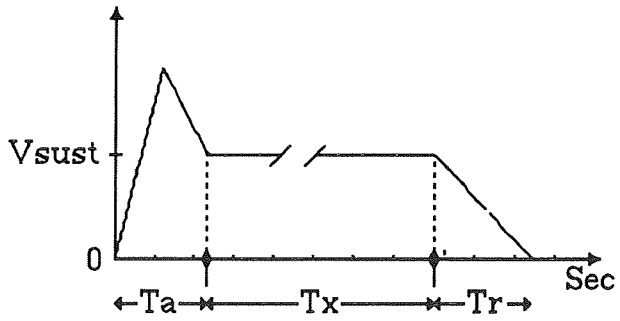
I criteri per la determinazione della lunghezza della tabella, della risoluzione della fase e dello step e sui metodi di approssimazione coinvolti, sono ampiamente riportati in letteratura [7].

1.2 L'inviluppo

I suoni naturali e quelli generati dagli strumenti musicali tradizionali, sono caratterizzati dalla particolare forma d'onda dei segnali ad essi associata e soprattutto dal modo con cui questa si evolve nel tempo. Nell'evoluzione di un suono, si possono individuare almeno tre fasi principali:

- fase di *attacco* [Ta] che rappresenta la modalità con cui lo strumento risponde all'impulso di partenza ed è contraddistinta dal tempo che occorre per portarsi da un volume zero a quello di regime.
- fase di *sostegno* o *tenuta* [Tx] e cioè il periodo di tempo in cui il volume rimane costante ad un certo valore; la durata di questa fase dipende, oltre che dalle caratteristiche fisiche dello strumento, dal desiderio e/o dalla necessità dello strumentista di allungare o meno il suono.
- fase di *rilascio* [Tr] e cioè il tempo che occorre per riportarsi al volume zero dal momento in cui viene a mancare l'agente che sostiene il suono.

All'andamento volumetrico, che si può dunque rappresentare come una funzione del tempo A(t), è stato dato convenzionalmente in nome di *inviluppo*.



E' evidente comunque, che la suddivisione in tre fasi è piuttosto rigida e approssimativa della realtà, ma per la sua praticità è oramai entrato nell'uso comune.

2 Elementi atomici e loro rappresentazione

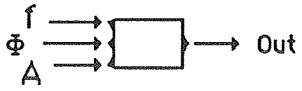
Si tratta ora di specificare le categorie di elementi di base per la sintesi sonor digitale a cui possiamo attingere nella fase progettuale dell'algoritmo di sintesi Estendendo il formalismo grafico già proposto ed adottato da Max Matthews nell realizzazione di MusicV [6] faremo uso di elementi che chiameremo **Atomi** per l loro evidenti caratteristiche di irriducibilità funzionale. Si individuano le seguent categorie:

Generatori
Inviluppi
Operatori aritmetici
Ritardi
Entrate ed Uscite

ognuna comprende più blocchi funzionali a ciascuno dei quali viene associato un simbolo da utilizzare nell'ambiente creato per l'*Editing* grafico.

2.1 Generatori

Col termine di "generatore" si indica quell'elemento presente in tutti gl algoritmi di sintesi, in grado di produrre i campioni di una forma d'onda, sia essa una semplice sinusoide, una funzione comunque complessa o un rumore bianco L'atomo di tipo "Generatore" viene simboleggiato da una generica "scatola nera" con tre ingressi ed una uscita.

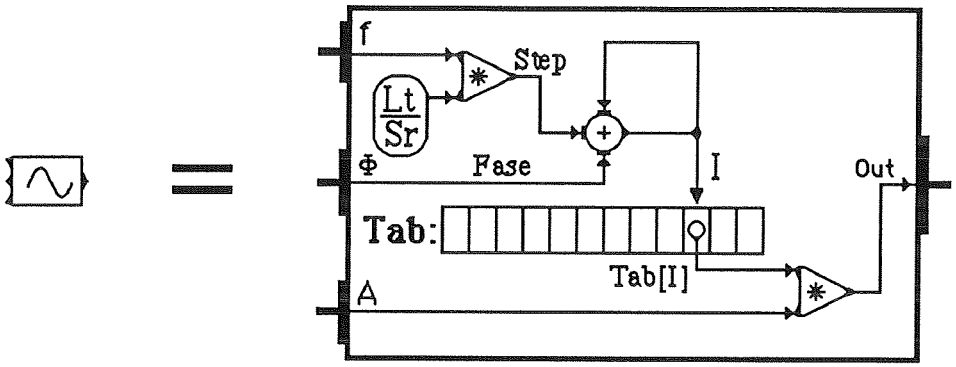


I tre ingressi riportano l'informazione di frequenza (f), quella di fase (Φ) e quella di ampiezza (A) ed influiscono sul risultato secondo il particolare tipo d generatore che la scatola rappresenta e dal modo con cui è stata realizzata. Avere a disposizione il tipo "Generatore" risulterebbe però troppo vago per i nostri scopi, s possono in effetti individuare, dall'analisi degli algoritmi di sintesi noti, almeno tre particolari generatori la cui presenza risulta necessaria: l'**Oscillatore**, il **Rumore Bianco** e la **Tabella Semplice**. Solo questi generatori verranno dunque a far parte

dei tipi ammissibili nel nostro formalismo.

2.1.1 Oscillatore

Il simbolo che si userà per l'atomo di tipo "Oscillatore" è indicato in figura :



Atomo Oscillatore

Le funzionalità interne mostrano chiaramente che siamo nel caso dell'oscillatore digitale a lettura tabellare (Table Look-Up Oscillator) dove una tabella **Tab** viene riempita con un ciclo di una determinata forma d'onda e letta con un indice I che dipende da:

$$I = (\ I \text{ precedente} + Step + F\ ) \ \textbf{mod}\ lt$$

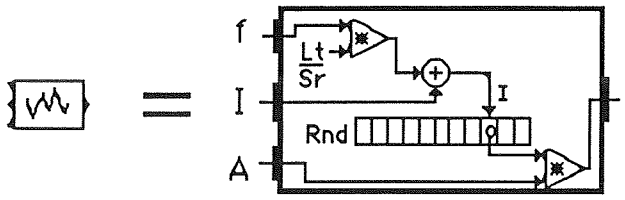
dove lt = lunghezza della tabella **Tab**, Step dipende dalla frequenza f del segnale da produrre come visto nel paragrafo 1.1

$$Out = A * \textbf{Tab}\ [I]$$

In molti algoritmi la fase non è rilevante, ed infatti l'oscillatore in letteratura è sempre indicato come una scatola con due soli ingressi corrispondenti ad f ed ad A.

2.1.2 Rumore Bianco

Il simbolo che si userà per l'atomo di tipo "Generatore di Rumore Bianco" è indicato nella seguente figura:

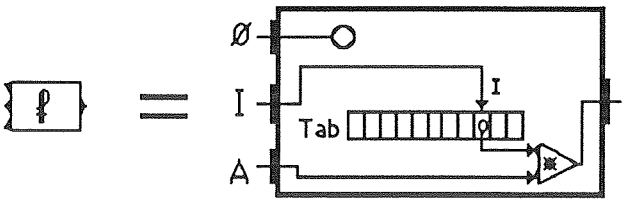


Atomo Rumore Bianco

Quando si vuole una forma d'onda spettralmente molto ricca avendo a disposizione un calcolatore, la si simula con un generatore di numeri casuali Rnd tale generatore aleatorio è in grado di fornire numeri scelti a caso su un dato intervallo con distribuzione uniforme; in questo modo si ricava un segnale dallo spettro continuo ovvero una forma d'onda ottenibile tramite la somma di tutte le frequenze da cui, per analogia con l'evento luminoso, il nome di rumore "bianco" Per poter agire sia sulla frequenza che sulla fase del segnale da produrre, vengono assegnati una serie di numeri casuali ad una tabella chiamata **Rnd**. La funzione tabulata può così essere letta con lo stesso procedimento descritto a proposito dell'oscillatore.

2.1.3 Tabella semplice

Il simbolo che si userà per l'atomo di tipo "Tabella Semplice" è indicato nella seguente figura:

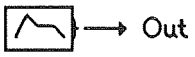


Atomo Tabella Semplice

Si tratta della rappresentazione in tabella di una opportuna funzione di trasferimento, usata di solito negli algoritmi di sintesi di tipo wave-shaping. La caratteristica di questo atomo è quella di poter essere letta direttamente mediante l'uso di un ingresso esterno. Pur sfruttando lo scheletro di un generatore, gli ingressi assumono un diverso significato.

2.2 Inviluppo

Come abbiamo già visto, la possibilità di modulare l'ampiezza di un segnale tramite inviluppo è di fondamentale importanza per tutti gli algoritmi di sintesi. Il simbolo usato per descrivere questo oggetto sara' dunque una scatola senza parametri d'ingresso, contenente un simbolo di ADSR (acronimo per Attack-Decay-Sustain-Release) che richiama l'evolversi tipico di un inviluppo.



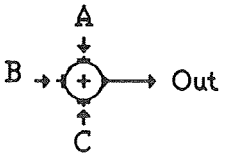
Le funzionalità di questo atomo si realizzano con una tabella lineare che viene scandita una sola volta per ogni suono. Seguendo la notazione fin qui adottata, l'uscita dell'inviluppo viene di solito collegata con l'ingresso A dei generatori.

2.3 Operatori Aritmetici

L'insieme degli operatori aritmetici, dai quali viene per il momento escluso quello di divisione perché di solito non presente nei processori DSP, vengono esplicitati graficamente e nelle loro funzioni, come segue:

2.3.1 Sommatore

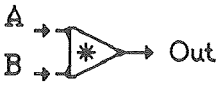
L'atomo sommatore realizza la somma di tre ingressi per problemi pratici



Out = A + B + C

2.3.2 Moltiplicatore

L'atomo moltiplicatore realizza la moltiplicazione di due ingressi.



Out = A * B

2.3.3 Negatore

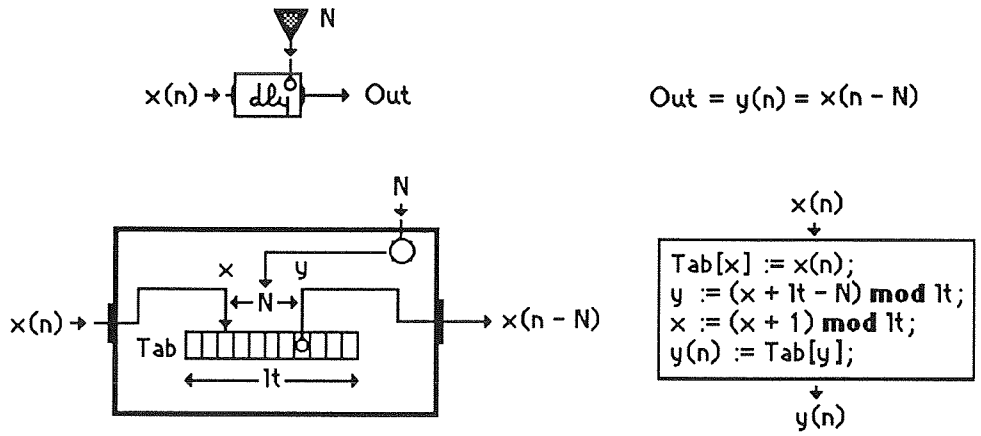
L'atomo negatore realizza l'inversione del segno del suo ingresso.

Out = - A

2.4 Ritardo

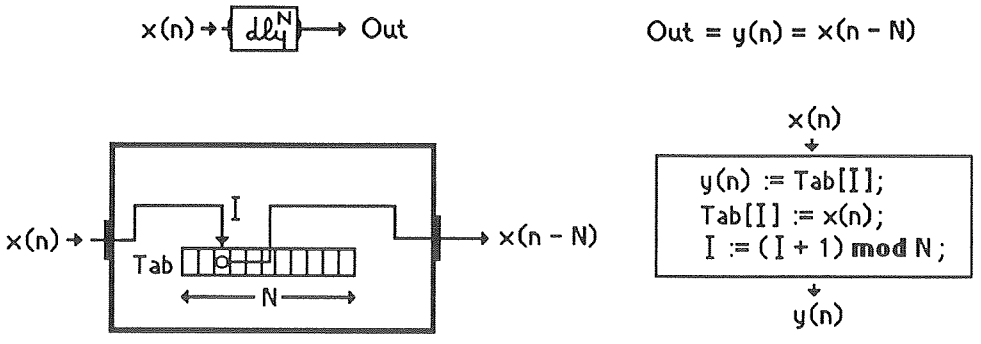
In generale, l'atomo di tipo "Ritardo" può essere visto come un sistema ch memorizza l'ingresso e lo restituisce dopo un certo lasso di tempo: se l'ingresso x(n indica un campione sonoro allora l'uscita y(n) sara' uguale a x(n-N), ovvero l'entrat ritardata di N campioni. L'unico parametro da definire è quindi N, per cui esiston due tecniche implementative: 1) N variabile, 2) N costante.

La prima tecnica si realizza con una tabella circolare puntata da due indic uno di lettura y ed uno di scrittura x, distanti l'un l'altro N elementi. Con quest tecnica è evidente che N non può essere maggiore della lunghezza lt della tabella.



Atomo Ritardo con N variabile

Nel secondo caso invece, se N viene dichiarato come costante all'inizio de programma di sintesi, allora si puo' predisporre la tabella del ritardo lung esattamente N. In questo modo per la realizzazione occorrera' un solo indice utilizzabile sia per la memorizzazione dell'ingresso come per la lettura dell'uscita.



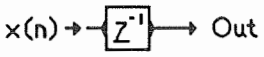
Atomo Ritardo con N costante

L'effetto prodotto sul suono è legato al valore di N, ma per un calcolo preciso del tempo di ritardo, occorre conoscere esattamente anche quanto vale la frequenza di campionamento Sr, essendo il tempo di ritardo ricavabile in questo modo:

$$\text{Tempo di Ritardo} = \frac{\text{Distanza in N campioni}}{\text{Freq. di campionamento}}$$

2.4.1 Ritardo unitario

Questo atomo permette il progetto dei diagrammi a blocchi tipicamente usati per descrivere la procedura computazionale implementativa di filtri digitali.



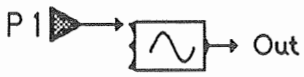
Out = x(n - 1)

2.5 Entrata e uscita

L'atomo d'entrata viene simboleggiato da un triangolo con una sola uscita; per comodita' di disegno ne introdurremo due, diversi solo nell'orientamento, ma utilizzabili con lo stesso identico significato.

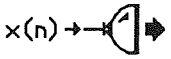
Atomi Entrata

In pratica, ognuno di questi moduli lo si può immaginare come una variabile, o una cella di memoria, il cui contenuto venga letto dall'algoritmo di sintesi e possa venir modificato dall'esterno. Esemplificando, l'atomo d'entrata (che chiameremo P1) potrebbe essere usato come entrata frequenziale di un oscillatore:



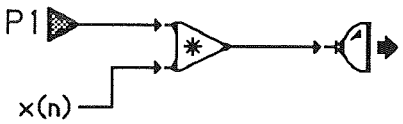
Entrata frequenziale per un Oscillatore

L'atomo uscita è l'elemento che raccoglie il risultato finale del calcolo di un algoritmo di sintesi. La simbologia che si trova in letteratura a tal proposito, solitamente tende a ricordare un altoparlante, volendo con questo rappresentare tutta la catena di conversione in analogico e di amplificazione sonora .



Atomo Uscita

L'unico ingresso di cui è dotato questo atomo è quindi il risultato di tut l'elaborazione precedente. Visto a basso livello, il compito dell' algoritmo di sinte per la realizzazione di questo atomo e' quello di assegnare il campione calcolato a una variabile o cella di memoria associata ad un canale fisico di uscita . Siccome i genere si deve poter agire sul volume di ogni singolo canale, se P1 e' il parametr che contiene l'informazione volumetrica, un semplice collegamento come quell mostrato qui sotto risolve completamente il problema.



Controllo del volume per l'Uscita

3 Gli algoritmi di sintesi

Come abbiamo già precedentemente visto, per algoritmo di sintesi si intende un modello matematico che tradotto in programma eseguito iterativamente, produce i valori numerici corrispondenti ai valori istantanei del segnale audio desiderato, secondo un prestabilito tasso di campionamento. Nella moltitudine di algoritmi di sintesi proposti negli anni nel settore della computer music, e che si differenziano per efficienza computazionale, tipi di sonorità ottenute, facilità di controllo etc., se ne possono individuare quattro ampiamente trattati in letteratura che per le loro caratteristiche possono ritenersi rappresentativi:

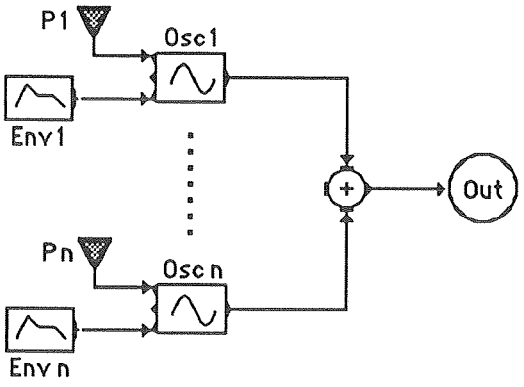
1) Sintesi Additiva.
2) Modulazione di Frequenza.
3) Distorsione non Lineare (WaveShaping).
4) Sintesi Sottrattiva.

3.1 Sintesi Additiva

Si tratta di sommare l'uscita di N oscillatori, di solito sinusoidali, secondo la classica formula che si rifà all'analisi spettrale di Fourier (per semplicità viene qui omessa l'informazione di fase)

$$v(t) = \sum_{k=1}^{n} A_k(t) \sin(k w t)$$

Nello schema viene indicato, per ciascuno degli oscillatori usati, anche l'inviluppo Env_j corrispondente al valore d'ampiezza A_i nella formula citata, ed il parametro P_i, che specifica l'entrata frequenziale dell'i-esimo oscillatore.



Sintesi Additiva

E' questo un metodo basato soprattutto sull'analisi del suono di sorgenti musicali reali: frammenti di suoni isolati vengono digitalizzati ed analizzati nelle loro componenti temporali e spettrali da cui si ricavano sufficienti informazioni per poter il piu' fedelmente possibile ricostruire il suono registrato.

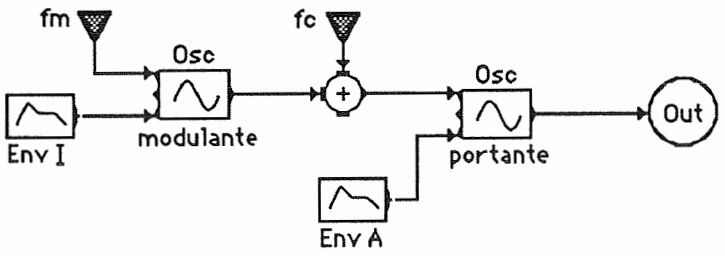
3.2 Modulazione di Frequenza

Si tratta dell'applicazione del procedimento della modulazione di frequenza usata nella trasmissioni radio-televisive, opportunamente adattata per la banda audio [7] e si basa su formule del tipo:

$$y(t) = A(t) \sin [2\pi f_c t + I \sin(2\pi f_m t)]$$

con $I = C/2\pi f_m$ *indice di profondità di modulazione*

La configurazione base per questa tecnica è composta da due oscillatori connessi in modo che l'uscita del primo (fm= frequenza modulante) vada ad influire sull'entrata frequenziale del secondo (fc = frequenza portante).



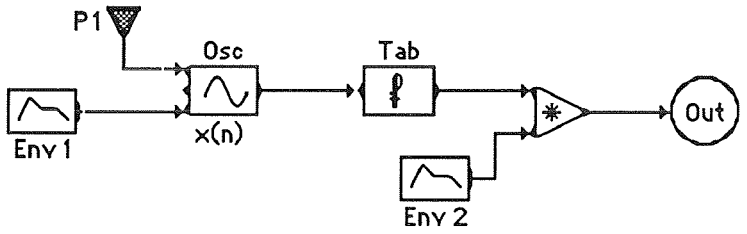
Modulazione di frequenza

Uno dei maggiori vantaggi della sintesi tramite modulazione di frequenza è che pur avendo un basso numero di parametri da controllare si ottengono segnali con spettri molto ricchi di componenti spettrali e ai quali corrisponde una gamma di sonorità molto ampia e diversificata. E' vero però che spesso si deve procedere empiricamente nella ricerca, avendo poche possibilità di prevedere il risultato sonoro globale al variare dei parametri della modulazione; come pure non esiste il modo per controllare gli andamenti di ciascuna componente, e questo anche in relazione al basso numero di "gradi di libertà" della metodologia [8].

Il concetto di modulazione viene poi esteso, e spesso si trovano modulazioni in cascata, somme di modulazioni che modulano una portante, etc. Essendo stata usata ampiamente in apparecchiature commerciali, a proposito della sintesi FM esiste una nutrita bibliografia. [8]

3.3 WaveShaping

Questa tecnica è nota anche col nome di *Distorsione non Lineare* [9]. Il concetto di distorsione dell'entrata x(n) si ottiene tramite una legge di trasformazione F(x) indipendente dal tempo che associa ad ogni valore della funzione in ingresso un valore d'uscita F(x(n)). In campo digitale si realizza con una tabella contenenete tale funzione che viene indicizzata tramite i valori che si ricavano dall'oscillatore "contenente" x(n). Nel caso più comune x(n) è una sinusoide.



WaveShaping

Le funzioni di trasferimento abitualmente usate sono i polinomi di Chebycev Pk(x) la cui proprietà è quella di moltiplicare la frequenza della sinusoide datagli come argomento: vale infatti l'identità

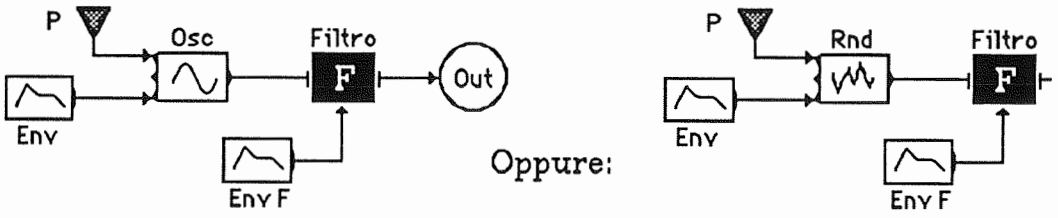
$$P_k(\text{sen } w_o) = \text{sen}(\, kw_o)$$

Si possono così utilizzare combinazioni lineari di questi polinomi per ottenere eguali combinazioni tra le armoniche della sinusoide d'ingresso. Caratteristica importante è che la distorsione cresce in proporzione col livello d'ampiezza di x(n), ed anche l'uscita, allora, sarà caratterizzata da un più alto contenuto frequenziale; questo fatto implica che si potrà avere facilmente uno spettro dinamico, fatto questo molto utile, per la sintesi di certi strumenti dal suono sostenuto. In sostanza, a discapito di una certa perdita di controllo nel dettaglio spettrale dell'uscita, la facilità d'uso in termini di parametri da controllare è eccellente e così pure l'efficienza computazionale.

3.4 Sintesi Sottrattiva

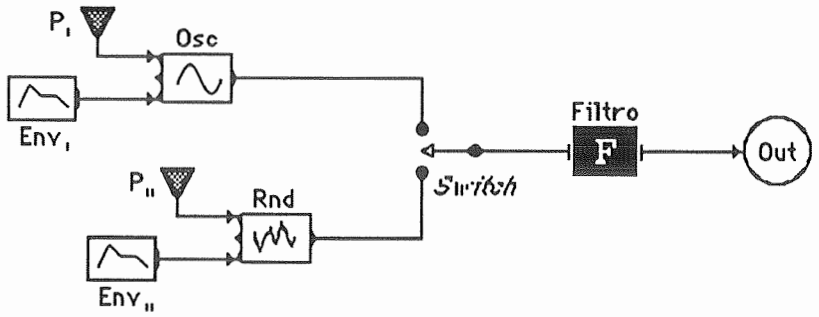
Il metodo è stato ampiamente usato nei sintetizzatori analogici, ed in effetti tutto il bagaglio conoscitivo legato a questa tecnica può essere riversato efficaciemente nel campo digitale. In generale si utilizza un oscillatore ricco di armoniche, seguito da un filtro o da una catena di filtri, i cui parametri possono cambiare durante la sintesi sonora influenzando così le ampiezze delle frequenze, (o meglio delle bande di frequenza) passanti ed in definitiva agendo sullo spettro del segnale.

Come si può vedere dalla figura seguente, per realizzare un cambiamento dinamico nel corso del tempo, può essere utile connettere uno o più inviluppi EnvF ai parametri che determinano la risposta in frequenza del filtro.



Sintesi Sottrattiva

Suoni percussivi sono ben modellati filtrando un rumore bianco e si possono ottenere suoni di tipo *vowel* connettendo piu' filtri in cascata. Se l'oscillatore produce una forma d'onda di una certa complessità, si ha un buon controllo sulla dinamica. Una variante di questa tecnica si ottiene connettendo allo stesso filtro, alternativamente, sia l'oscillatore che il rumore bianco; questo fatto si rappresenta di solito introducendo un interruttore (*switch*) come mostra lo schema di figura seguente, noto col nome di *Sintesi Predittiva Lineare*..(LPC)



Sintesi Predittiva Lineare

Questo metodo di sintesi è usata soprattutto per la simulazione del parlato e prevede quindi una fase precedente di accurata analisi dei tipi dei suoni da emettere: l'oscillatore simula le corde vocali, la sorgente di rumore viene usata per "generare" le consonanti, e il filtro simula la risonanza della cavità orale. Avendo poi a disposizione un linguaggio di programmazione è possibile realizzare opportuni strumenti che controllano il variare degli stati dello switch e dei valori paramentri del filtro, che con la giusta sequenzializzazione crea i fonemi di base per la simulazione del parlato.

In generale per descrivere l'effetto di un filtro su tutto lo spettro delle frequenze da 0 a Sr/2 normalmente vengono utilizzati due grafici: la risposta in ampiezza e la risposta di fase. Dalla ipotesi di linearita' del filtro si possono ricavare i grafici semplicemente dando in ingresso al filtro sinusoidi di diverse frequenze e misurando per ognuno l'evoluzione dell'ampiezza e della fase in uscita.

Sia ad esempio

$$y(n) = x(n) + x(n-1) \qquad n = 1,2,....$$

l'equazione alle differenze che descrive il calcolo da effettuare su x(n) per realizzare un semplice filtro. Una sua implementazione come procedura *Pascal* prevede che y(n) ed x(n) siano due *array* di M+1 elementi e che x(n) sia costituito dagli M campioni del segnale da filtrare. (Notare che dovra' essere x[0]=x[M]).

```
procedure Filtro (x,y: array [0..M] of real);
  var
    n : integer;
  begin
        for  n:=1 to M do
             y[n] := x[n] + x[n-1];
        y[0] := y[M]
  end;
end;
```

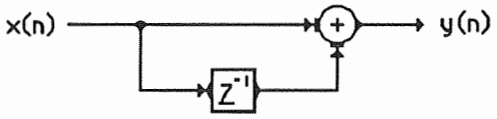
Si sa che esistono principalmente tre tipi di filtro: passa basso, passa alto, passa banda, per ognuno dei quali si può sempre ricavare il segnale in uscita conoscendo il segnale entrante e la risposta del filtro. Per calcolare l'uscita dei sistemi LTI (Lineari nel Tempo Invarianti) si puo' utilizzare la seguente equazione alle differenze finite a coefficienti costanti:

$$y(n) = \sum_{K=1}^{N} a_K y(n-K) + \sum_{K=0}^{M} b_K x(n-K)$$

Poiché è possibile considerare questa equazione come una descrizione

generale di tutti i sistemi LTI realizzabili (computabili), allora si nota come i calcoli implicati in un filtro digitale possano essere ben rappresentati da equazioni e/o da frammenti di programma.

Un altro modo molto usato di descrivere i filtri digitali è quello che fa uso di opportuni diagrammi a blocchi del tipo mostrato in figura,



ormai universalmente usati per la loro facilità di lettura.

4 Gli algoritmi di sintesi come grafi

Come abbiamo visto, la produzione di suoni tramite elaboratore si configura come l'esecuzione ciclica di un processo (per l'appunto l'algoritmo di sintesi) volto alla produzione di campioni numerici che rappresentano il segnale corrispondente. Sotto precise ipotesi la rappresentazione numerica è percettivamente associata ad un evento sonoro "continuo", realizzando così una comoda modalità di elaborazione del segnale acustico mediante la sua "immagine" numerica.

In questo capitolo ci riproponiamo di raggiungere un maggiore grado di dettaglio nell'analisi degli algoritmi di sintesi, con l'approfondimento di una serie di nozioni e concetti che ci permetteranno da un lato di verificare alcune proprietà di correttezza degli schemi disegnati dall'utente, dall'altro di sviluppare un metodo efficace per la traduzione di questi schemi in una sorta di codice intermedio.

Nella sua generalità un algoritmo di sintesi è una computazione che produce dei risultati numerici (campioni) in base allo stato di alcune variabili interne ed a determinati flussi di input. La rappresentazione proposta in questa sede si basa su una notazione essenzialmente grafica, nella quale lo schema della computazione è espresso tramite collegamenti fra elementi atomici in una struttura di grafo orientato.

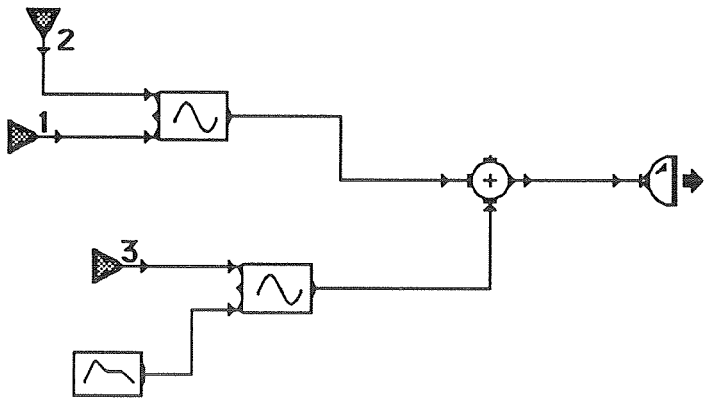


fig. 4.1

Dopo aver chiarito precedentemente i motivi della scelta degli elementi atomici, ci preme qui individuare i significati e le implicazioni di questa notazione. Un "grafo di sintesi" come sopra introdotto è un insieme di archi (linee di collegamento) e nodi (elementi atomici) con le seguenti caratteristiche:

i) esiste un unico nodo "iniziale" (il senso preciso di questa affermazione verrà chiarito più avanti), l'altoparlante, che non ha archi uscenti
ii) ogni nodo può avere uno stato interno, cioè non rappresenta necessariamente una funzione ma più genericamente un sottoprogramma
iii) ogni arco esprime passaggio di *dati*, e non di controllo, fra due nodi
iv) la computazione espressa da un elemento atomico è un processo che si suppone avvenire in ambiente locale, senza interferenze quindi con altri elementi atomici tranne lo "scambio di messaggi" esplicito degli archi
v) deve valere la proprietà di connessione del grafo. Eventuali componenti non connesse all'altoparlante sono infatti ininfluenti ai fini della formazione del suono e sono quindi superflue.

Si veda ad esempio il grafo presentato nella figura seguente: il compito di un algoritmo di "compilazione" di un generico grafo di questo tipo, oltre che nell'espansione di eventuali elementi *sottografo* ed al linking dei vari ingressi ed uscite, consiste essenzialmente nella sequenzializzazione dei moduli a, b, c e dell'altoparlante in modo da rispettare il corretto flusso dei dati.

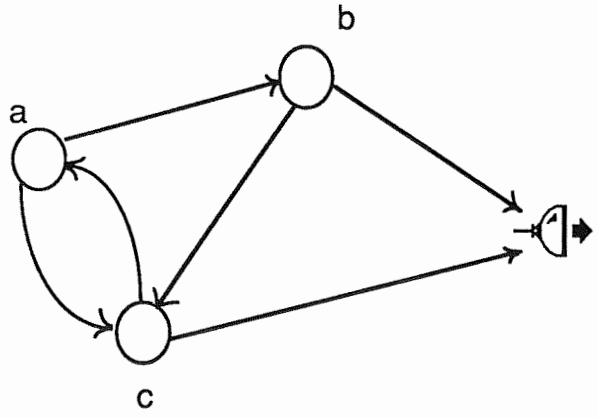


fig 4.2

Questa operazione di sequenzializzazione non è sempre possibile, come sarà chiaro in seguito, e comunque si presenteranno una serie di problemi computazionali che illustreremo nei paragrafi che seguono.

4.1 Il problema del parallelismo

Secondo la definizione data di "grafo di sintesi", nulla vieta che esso possa essere preso a modello per esprimere delle computazioni in generale parallele, come in figura

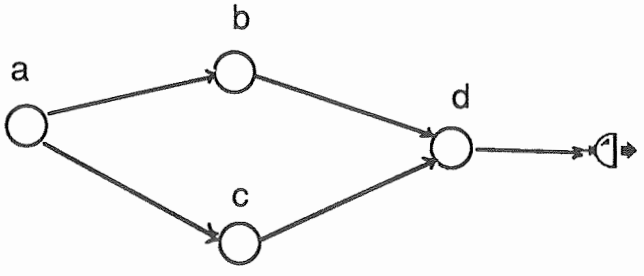


fig 4.3

dove i dati in ingresso a *d* sono forniti da una computazione non sequenzializzata di *b* e *c*, nel senso che il diagramma non impone nessun ordine di priorità fra i due nodi. Dovendo simulare questa computazione su una macchina sequenziale, si presenta il problema di stabilire con precisione se e come l'esecuzione dei moduli può essere sequenzializzata e quali sono i criteri che vanno adottati. A questo fine, definiamo una relazione tra i nodi del grafo che sia in grado di esprimere il concetto di *dipendenza funzionale* e quindi di *precedenza* nell'esecuzione tra nodi.

Definizione: dati due nodi **a** e **b** diremo che **a** *precede* **b** se esiste un cammino da **a** a **b** all'interno del grafo. Questa proprietà verrà indicata anche dicendo anche che **b** *dipende funzionalmente* da **a**.

Il senso della relazione introdotta risulta chiaro quando si vede che, nel caso in cui **a** *preceda* **b**, i dati prodotti dal nodo **a** sono prima o poi necessari all'esecuzione del nodo **b**, e che quindi l'esecuzione del primo nodo ha una priorità maggiore rispetto a quella del secondo. Questa relazione non è in generale una relazione d'ordine, per i motivi che emergeranno in seguito, ma è semplice vedere fin da ora che se **a** non precede **b** e neanche il viceversa è vero, allora i due nodi non hanno dipendenze funzionali l'uno dall'altro e possono essere quindi eseguiti in un qualsiasi ordine di precedenza (è questo il caso dei nodi *b* e *c* del grafo in figura 4.3), dal momento che i nodi rappresentano computazioni locali e senza mutue interferenze. Un esempio di ciò si ha nella sintesi additiva (fig 4.4), dove più elementi oscillatori contribuiscono parallelamente alla formazione di un suono complesso.

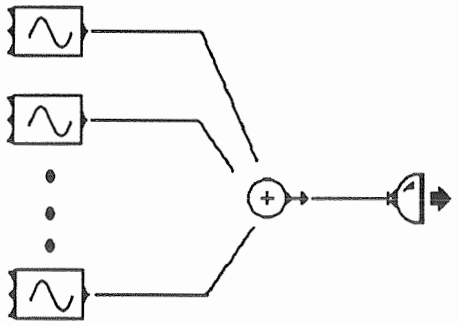


fig. 4.4

Nella simulazione di un algoritmo di questo tipo, l'ordine con il quale viene calcolato il successivo valore di ogni oscillatore non ha importanza, fermo restando il vincolo che l'esecuzione di tutti gli oscillatori debba precedere l'esecuzione della somma.

I nodi di grafi di sintesi possono essere associati a nodi di computazioni data-

flow, nel senso che tutti i dati devono essere presenti agli ingressi di un elemento affinche questo possa calcolare il successivo valore e propagarlo lungo il suo arco d'uscita [10]. La differenza fondamentale consiste nel fatto che le architetture di elaborazione per cui è necessario produrre codice non sono in generale parallele, e che quindi il problema in quest'ambito è quello di "schiacciare" un grafo di sintesi in modo da sequenzializzarlo. Il modello data-flow, invece, si presenta come una notazione per algoritmi eseguibili su architetture a parallelismo massiccio, e quindi il problema, esattamente opposto, è quello di trovare nella computazione da effettuare la massima esplicitazione possibile del parallelismo insito nei processi.

4.2 Il problema dei cicli

Se la relazione *precede* non è una relazione d'ordine, ciò è dovuto solo dalla possibile presenza di cammini ciclici nei grafi di sintesi. Non vale infatti in generale la proprietà antisimmetrica: può esistere un cammino da un nodo **a** ad un altro nodo **b** e contemporaneamente il cammino opposto senza che necessariamente **a** e **b** siano lo stesso nodo.

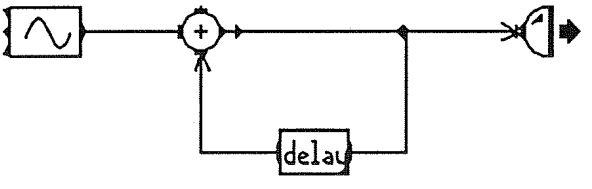


fig. 4.5

Nella figura 4.5 esiste un ciclo fra l'atomo sommatore ed il delay: l'algoritmo può essere associato ad un rudimentale meccanismo di eco. Questi tipi di cicli rappresentano un frequente mezzo per consentire la retroazione di segnali sonori e la realizzazione di effetti acustici che vanno dalla spazializzazione ai filtri digitali.

E' possibile trasformare la relazione *precede* in relazione d'ordine, considerando un nuovo grafo i cui elementi siano le classi di equivalenza ottenute applicando la relazione sui nodi definita come "a *e* b *sono in relazione sse esiste un ciclo tra loro*", secondo un procedimento del tutto generale per le relazioni di preordinamento descritto in [11]. Ogni classe di equivalenza conterrà così interi cicli del grafo, oppure elementi singoli quando essi non facciano parte di cicli. Per ciò che riguarda l'esempio di figura 4.5, la classe di equivalenza dell'atomo somma conterrà anche l'atomo delay, mentre tutti gli altri atomi formeranno classi di equivalenza di un solo elemento. Il grafo risultante è

chiaramente aciclico, e per questa ragione ordinabile (parzialmente) secondo la relazione di *precedenza* funzionale. Ogni nodo appartenente alla stessa classe di equivalenza, facendo parte dello stesso ciclo, verrebbe così a ricevere lo stesso ordine di precedenza di tutti gli altri nodi del ciclo e questo, ai fini dell'esecuzione sequenzializzata dei nodi, non rispecchia l'ordine di dipendenza di un nodo dall'altro che pure continua "parzialmente" ad esistere all'interno di un ciclo.

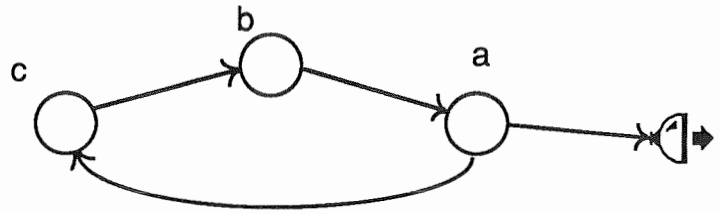


fig. 4.6

In questa figura, ad esempio, il ciclo formato dai tre nodi *a*, *b* e *c*, per quanto non ordinabile (per ogni coppia di nodi esiste una dipendenza del primo dal secondo e viceversa) mostra però il fatto che il nodo *a*, in diretto collegamento con l'altoparlante, dovrebbe essere l'ultimo ad essere eseguito, preceduto dalla catena di affluenze dalle quali esso ricava i dati necessari alla sua elaborazione. Resta ovviamente in ogni ciclo almeno un arco i cui valori sono inizialmente indefiniti (in questo caso, l'arco *a* -> *c*): una volta però stabilito questo valore, sia esso un valore per default oppure fornito dall'esterno, la computazione evolve rispettando i vincoli di precedenza tra nodi.

Emerge da questa discussione anche un criterio per confrontare i nodi all'interno di cicli e stabilirne la precedenza: nodi appartenenti ad un ciclo devono essere ordinati in modo tale che la "testa del ciclo" (cioè l'elemento che fornisce i dati all'esterno) sia eseguita per ultima, preceduta dalla sequenza di nodi che forma la sua catena funzionale di affluenze. Sotto precise ipotesi, la "testa di un ciclo" è unica, ed è univocamente determinabile la sequenza di nodi che forma la "catena di affluenze". Il nostro obbiettivo sarà dunque quello di definire con precisione il significato di "testa di un ciclo" e di esprimere attraverso una funzione di confronto tra nodi il concetto di precedenza funzionale esteso anche a nodi componenti un ciclo.

Per continuare l'analisi dei grafi di sintesi ad un livello di maggiore formalizzazione, consideriamo un grafo ottenuto da quello in esame invertendone l'orientamento degli archi; questa operazione, pur conservando le proprietà strutturali del grafo stesso (i cicli vengono conservati, ecc.) ci consentirà di implementare algoritmi di

visita e di ordinamento partendo da una radice (l'atomo altoparlante) e procedendo in avanti lungo gli archi.

4.3 La consistenza dei grafi di sintesi

In questo paragrafo utilizzeremo una notazione abbastanza informale per descrivere gli algoritmi sui grafi e le strutture dati per rappresentare gli stessi. Il linguaggio utilizzato avrà uno stile Pascal-like corredato, ove occorra, di descrizioni discorsive e comandi di significato intuitivo. Per ciò che riguarda le strutture dati, immagineremo un grafo rappresentato come un record contenente la specifica dei dati di un nodo e due liste di puntatori a nodi: quelle corrispondenti agli archi uscenti (forward star) ed entranti (backward star). Un grafo verrà così ad essere rappresentato da un elemento *nodo*.

Definiamo per prima cosa l'albero di copertura "depth-first", o dfst di un grafo, come l'albero di copertura ottenuto visitando i nodi del grafo secondo la modalità *depth-first* . Questa modalità di visita prevede che a partire da un nodo vengano visitati per primi i "figli" e, dopo la conclusione di questa visita, i fratelli del nodo stesso. La visita può essere implementata facilmente tramite chiamate ricorsive e tramite marcatura dei nodi via via che essi vengono visitati. Algoritmi per la costruzione di tali alberi sono presenti in letteratura, e ne forniamo una versione di riferimento.

```
function dfst (n: nodo): albero;
  begin
  crea un albero, sia esso tree, di radice n;
  marca(n);
  foreach m successore di n do
    if not marcato(m) then
      aggiungi dfst(m) ai figli di tree;
  return(tree)
end
```

Dato un grafo qualunque, è possibile costruire diversi dfst per esso, a seconda della scelta dei nodi successori effettuata nel comando *foreach* in figura. Tutti gli alberi di questo tipo comunque godono di alcune proprietà importanti ai fini della nostra analisi: ci riferiremo al dfst di un grafo intendendo uno qualsiasi di quelli possibili.

Un'altra importante relazione fra i nodi del grafo è la seguente:

Definizione:si dice che un nodo a domina b se ogni cammino radice, a , b passa necessariamente per a.

Questa relazione d'ordine tra nodi è studiata in letteratura [12] ed esistono efficienti algoritmi per il suo calcolo su grafi opportunamente rappresentati. Per i nostri scopi, è sufficiente definire una funzione booleana DOMINA(a,b) che valga true se a domina b, false altrimenti. L'algoritmo per la funzione DOMINA, illustrato dalla coppia di funzioni mutuamente ricorsive in fig. 4.9, si basa sul fatto che dati due nodi distinti *a* e *b*, se *a* domina *b* allora esso domina tutti i predecessori di *b*. Se i nodi sono uguali, la funzione vale true (ogni nodo domina se stesso) come nel caso in cui *a* sia la radice del grafo (la radice domina tutti gli altri nodi). Nel caso invece in cui b sia la radice, la funzione vale false: questo caso viene raggiunto quando procedendo da *b* a ritroso si incontra la radice senza aver incontrato il nodo *a*. Come in tutti gli algoritmi che implicano una visita al grafo, i nodi già elaborati vengono opportunamente marcati per tenere traccia della visita su di essi. Così la condizione di terminazione su false non consiste solo nel raggiungimento della radice, ma anche nell'aver incontrato un nodo precedentemente marcato.

```
function Domina (a,b:nodo): boolean;
  var verofalso: boolean;
  begin
  if a è radice or a=b then return(true)
  else if b è radice or marcato(b) then  return(false)
       else
        begin
          marca(b);
          verofalso=AuxDom(a,backstar(b));
          Unmark(b);
         return(verofalso)
        end
  end


function AuxDom(a:nodo,ln:listanodi):boolean;
  begin
  if ln = NIL then return(true)
  else return(Domina(a,primo nodo in ln) and
       AuxDom(a,successivi nodi di ln))
  end
```

fig 4.9

In figura, si è utilizzata la "funzione" backstar(*n*) per individuare la lista dei nodi che affluiscono ad un dato nodo *n* (la *backward star* del nodo).

Gli alberi di copertura depth-first rivestono un'importanza particolare per l'individuazione di cicli all'interno di un grafo, poiché può essere dimostrato che nell'insieme degli archi costituenti un cammino ciclico ne esiste almeno uno che collega

un discendente ad un antenato dell'albero dfst. Tale archi, detti *retreating edges*, possono quindi essere utilizzati per l'individuazione di cammini ciclici. In figura 4.10 è illustrato un grafo ciclico, il suo albero di copertura e, tratteggiato, l'arco *retreating* corrispondente al ciclo.

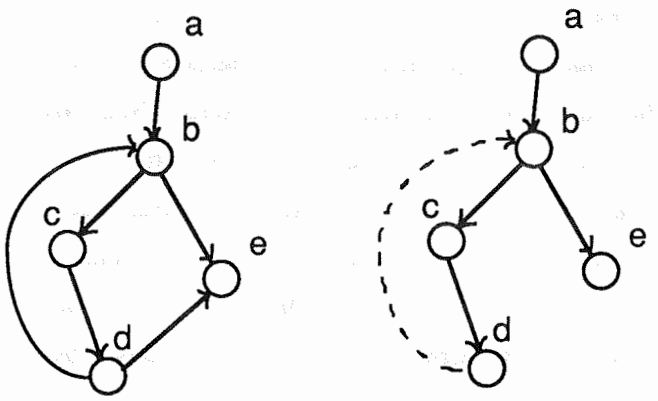


fig. 4.10

Questa proprietà dei retreating edges deriva dal fatto che una visita depth-first del grafo attraversa completamente eventuali cicli presenti, e che quindi l'ultimo arco di tale ciclo deve per forza di cose connettere un nodo con un'altro già visitato (quindi suo antenato nel dfst). Più formalmente: si faccia l'ipotesi che il grafo presenti un ciclo: allora una visita al grafo incontrerà prima o poi uno dei nodi che compongono il ciclo, sia esso *n*. Nel sottoalbero di radice *n* sarà presente quindi l'intero ciclo di cui *n* fa parte e, siccome il cammino tra questi nodi si richiude su se stesso, esisterà un nodo *m* tale che *m* -> *n*. Quest'arco tra *m* ed *n* collega quindi un discendente ad un antenato dell'albero dfst. L'implicazione inversa è anch'essa vera, poiché se un dfst presenta un retreating edge, allora sicuramente il grafo è ciclico.

L'importante ipotesi che esista un nodo del ciclo che domini tutti gli altri è alla base della nozione di *grafo riducibile* ([12]) ed è la chiave di interpretazione del concetto di "testa di ciclo". Diamo qui una delle possibili formulazioni della nozione di riducibilità di un grafo.

Definizione: un grafo si dice riducibile se ogni ciclo all'interno di esso presenta un nodo che domina tutti gli altri. Questo nodo è la testa del ciclo in esame.

Se un grafo non è riducibile (fig. 4.11) esistono cicli al suo interno che sono

raggiungibili da più punti, quindi non esiste un nodo iniziale per esso. I retreating edges associati a grafi irriducibili inoltre variano al variare dell'albero di copertura che viene costruito per il grafo: ciò è diretta conseguenza del fatto che i cicli in questo tipo di grafi non hanno una "testa" univocamente determinata.

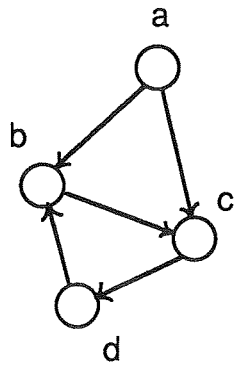


fig. 4.11

Nel grafo irriducibile in figura, ad esempio, il ciclo composto dai nodi {b, c, d} ha due punti di ingresso e per esso non è possibile definire quindi una "testa di ciclo". E' importante a questo punto notare che i grafi irriducibili non sono nemmeno parzialmente ordinabili, pur facendo uso di archi con valori di default: nel ciclo sopra illustrato si ha ad esempio un un insieme di nodi per il quale non è possibile stabilire quale di essi debba essere eseguito per ultimo (quale di essi sia il punto di collegamento con l'*esterno*). E' anche vero che una differente scelta della precedenza da attribuire a nodi facenti parte di un ciclo di un grafo irriducibile porta a diversi risultati computazionali, anche a parità di valori default assegnati agli archi. Ai fini della nostra analisi sugli algoritmi di sintesi, siamo quindi costretti a limitare i possibili schemi solo a quelli che siano rappresentabili tramite grafi riducibili. Chiameremo *consistenti* questi algoritmi, e svilupperemo un insieme di metodologie software per individuare la consistenza in un grafo di sintesi e per stabilire efficacemente l'ordine con il quale i nodi devono essere eseguiti.

La relazione che lega la *dominazione* come precedentemente definita agli alberi dfst è la seguente: *Se un nodo* **a** *domina un altro nodo* **b**, *allora* **a** *è un antenato di* **b** *nel dfst associato*. Infatti, supponendo per ipotesi assurda che a non sia antenato di b nell'albero, si avrebbe b appartenente ad un'altro sottoalbero (che non contenga a) e quindi sarebbe falso che ogni cammino dalla radice a b passa necessariamente per a.

Nell'ipotesi che il grafo sia riducibile (ipotesi che sarà sottintesa in ciò che segue) la testa di un ciclo sarà anche il primo nodo del ciclo ad essere incontrato in una visita depth-first al grafo. Proseguendo nella visita, si incontreranno via via tutti i nodi del ciclo

in un ordine che, sotto precise ipotesi, corrisponde all'ordine stabilito precedentemente nella nostra discussione. Queste ipotesi riguardano il fatto che ci sia un rapporto di dominazione di ogni nodo del ciclo nei riguardi del successivo.

Bisogna chiarire comunque che questo ordine non è sempre ottenibile tramite la semplice visita depth-first del ciclo: un esempio di ciclo per cui non valgono queste ipotesi è dato in fig. 4.12, dove viene presentato un grafo con l' albero di copertura ad esso associato.

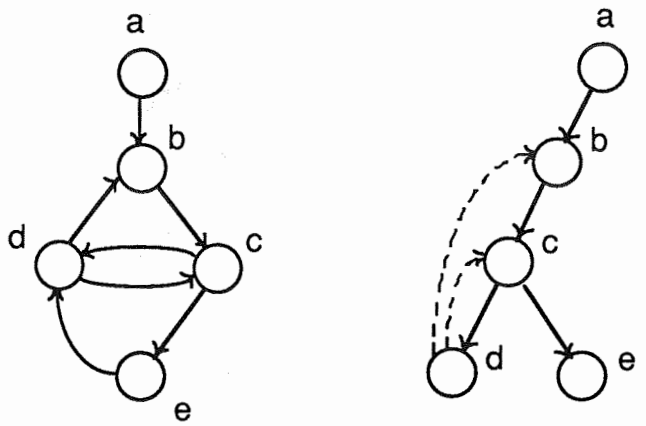


fig. 4.12

In questo esempio, il nodo *e* non domina il nodo *d*, che pure è suo successore in ogni ciclo di cui *e* fa parte.

A conclusione della nostra discussione, si può riassumere quanto detto come segue: l'ordine con cui si vogliono eseguire i vari nodi di un grafo di sintesi è quello che rispetta la priorità di chi fornisce i dati nei confronti di chi li usa. L'unico caso in cui questo non è realizzabile è quando si hanno dei cicli all'interno del grafo: in questo caso si desidera un ordine d'esecuzione nel quale la testa di ogni ciclo viene eseguita dopo il *corpo* del ciclo, interrompendo così l'ambiguità di quale nodo vada eseguito per ultimo. Nel caso in cui il grafo sia riducibile è possibile identificare con precisione la testa di ogni ciclo: altrimenti, non possiamo stabilire un ordinamento consistente dei nodi; limiteremo quindi la nostra analisi a questa classe di grafi riducibili. Pur scegliendo in un ciclo quale dei nodi sarà l'ultimo, si può avere (fig. 4.12) una situazione di ambiguità tra i restanti nodi di un ciclo: per risolverla utilizzeremo ora dei concetti introdotti nella trattazione. Abbiamo visto in precedenza come un grafo ciclico presenti almeno un retreating edge nel suo dfst, e come ad ognuno di questi archi corrisponda uno o più cicli. Con queste

premesse, risulta chiaro come il grafo ottenuto dal primo eliminando tutti e soli gli archi *retreating* non sia più ciclico e contemporaneamente possa essere ricoperto dallo stesso dfst di partenza. Sul grafo non ciclico ottenuto vale la relazione *precede* come relazione d'ordine. L'ordine ottenuto con questo meccanismo corrisponde "intuitivamente" all'ordine con cui ci aspettiamo che vengano eseguiti i nodi del ciclo. Nel caso della figura 4.12, l'eliminazione degli archi retreating produce il grafo seguente:

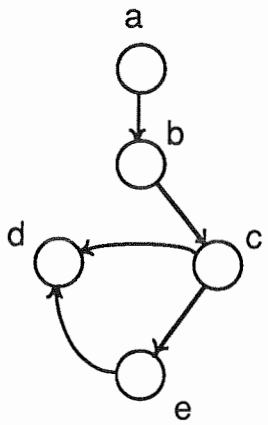


fig. 4.13

Su questo grafo, privo di cicli, è possibile imporre un ordinamento dei nodi senza conflitti di priorità. L'ordine risultante sarà il seguente:

a - b - c - e - d

Ordine che, opportunamente invertito per tener conto dell'opposto orientamento degli archi rispetto al grafo iniziale, vede eseguito per *ultimo* l'elemento altoparlante (il nodo *a* nel nostro esempio) e così via via tutti gli altri elementi fino ad arrivare ai primi elementi ad essere eseguiti, che sono quelli "iniziali", cioè senza archi entranti (il nodo *d*): questi elementi corrispondono nel nostro sistema agli ingressi esterni e agli inviluppi.

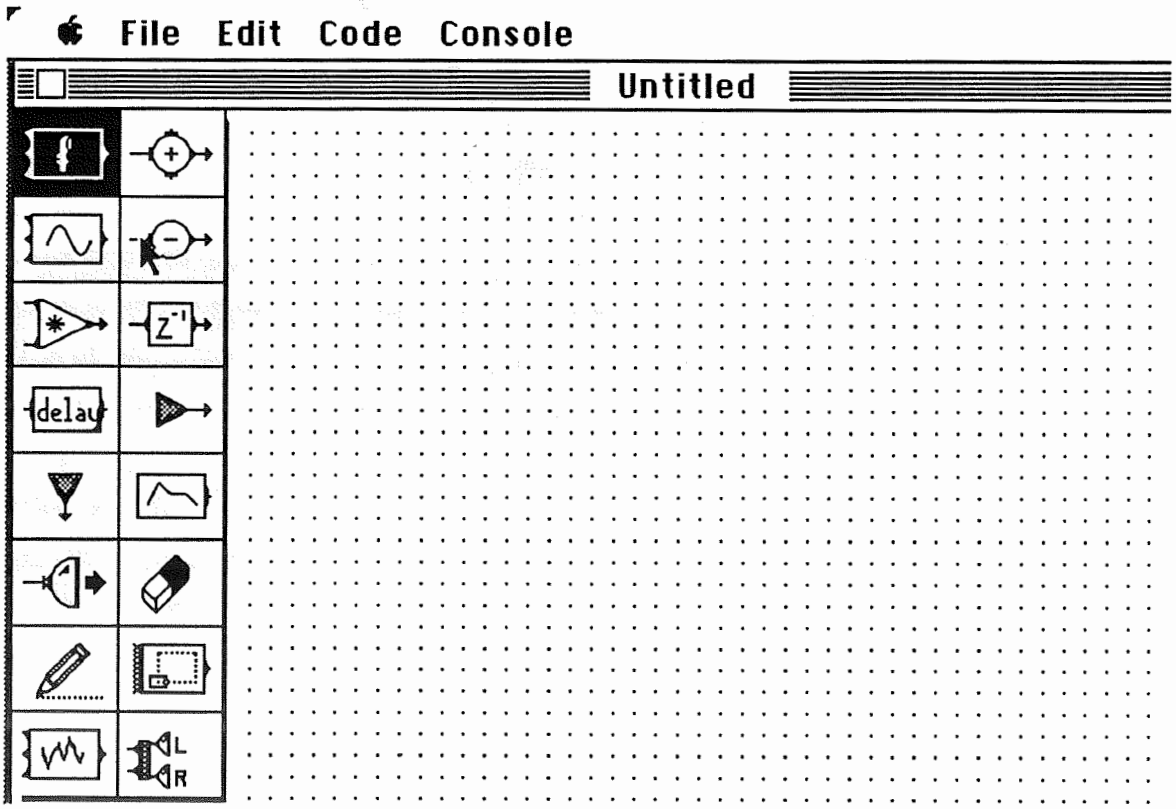
Gli archi retreating, in definitiva, sono quegli archi che devono ricevere un valore default perché inizialmente indefiniti. Dato un grafo (riducibile), questi archi sono unici (non dipendono cioè dall'albero di copertura che si associa al grafo). Ciò sta ad indicare che questi archi sono "strutturalmente" quelli che richiudono su se stesso un ciclo, e che qualunque sia l'approccio che si adotta essi sono inizialmente indefinibili.

5 L'Editore

In questo paragrafo vengono presentate le principali funzioni che l'Editore mette a disposizione nell'attuale versione. Per quello che riguarda le funzionalità interne, è sufficiente qui dire che l'ambiente di sviluppo utilizzato per realizzare questo strumento è il linguaggio C della Think Technologies, e che esiste in proposito un'ampia documentazione comprensiva dell'intero listato del programma [13] .

5.1 Piano di lavoro

All'attivazione, l'Editore propone un piano di lavoro entro il quale eseguire tutte le operazioni. Seguendo la filosofia di lavoro delle applicazioni Macintosh, la zona in alto dello schermo è costituita dalla cosiddetta **barra dei menu**, e tutto lo spazio restante viene occupato da un'unica finestra a dimensioni fisse nella quale si possono individuare tre distinte regioni:

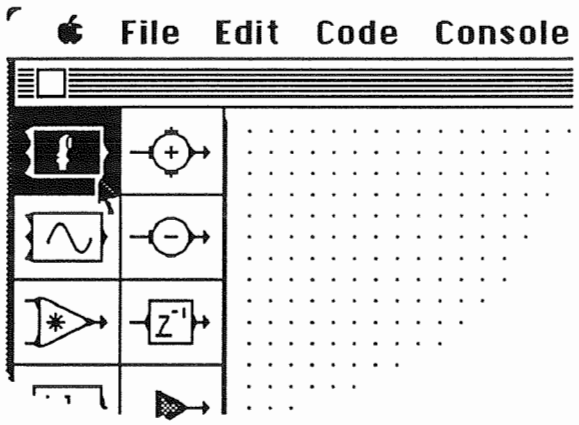


Finestra Principale

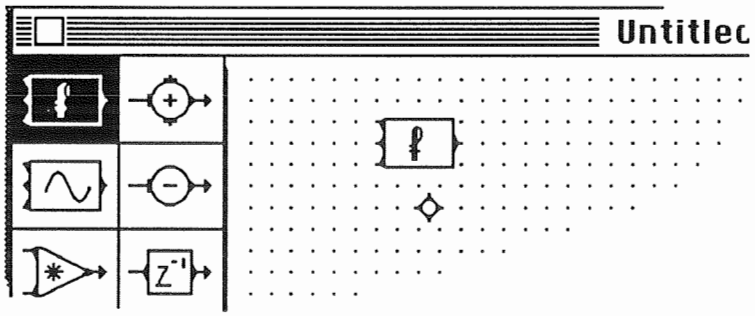
la regione della barra del titolo detta **TitleBar** di nome "**Untitled**", la regione **Palette** contenente le icone degli elementi atomici e degli operatori grafici, e la regione **Drawing**

costituita da tutto lo spazio restante che realizza il piano effettivo di disegno.

Per selezionare un elemento, è sufficiente posizionare il cursore a freccia comandato dal mouse su una delle icone che raffigurano gli elementi, e premere il pulsante del mouse stesso. L'avvenuta selezione viene indicata dal cambiamento di sfondo dell'elemento, che viene rappresentato a colori invertiti. In ogni istante, l'icona invertita rappresenta l'elemento correntemente selezionato.



A questo punto, un click del mouse sul piano di lavoro crea una istanza dell'elemento selezionato disegnando l'icona ad esso corrispondente sul punto del click.



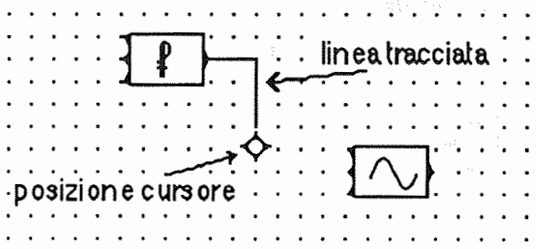
Per quello che riguarda gli elementi generatori e gli atomi di input (indicati con un triangolino orizzontale verso destra o verticale verso il basso) va fatto notare che per essi è prevista la numerazione automatica che viene attivata al momento della loro collocazione sul piano di lavoro. Il primo generatore (o input) disegnato apparirà con il numero 1, il secondo con il numero 2 ecc.. Speciali convenzioni sono presenti nella cancellazione di questi elementi.

5.2 Tracciamento delle linee

Selezionando l'elemento "penna", il penultimo in basso a sinistra nella palette, il

cursore viene abilitato a tracciare linee sullo schermo fra un elemento e l'altro. L'operazione si effettua posizionandosi sull'uscita di un elemento già presente sullo schermo, oppure su un punto di angolo di una linea già tracciata, e premendo il pulsante del mouse. Con il pulsante sempre premuto, ci si sposta nel piano di lavoro tracciando una linea che termina nell'ingresso di un elemento preesistente. Per terminare una linea, è sufficiente rilasciare il pulsante del mouse. Il tracciamento delle linee è soggetto ad alcune regole, le più importanti delle quali sono:

- ogni linea può iniziare solo nel punto di uscita di un elemento, oppure in un angolo di una linea già esistente
- il termine della linea deve essere l'ingresso di un altro elemento
- non si possono tracciare linee che finiscano laddove termina una linea preesistente

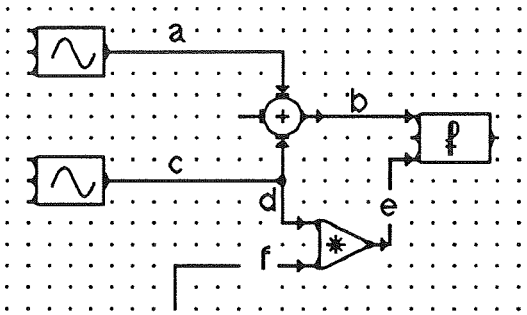


In caso di errore la linea viene "cancellata" o, se l'errore è nel punto iniziale, essa non "parte" neppure. E' possibile cancellare dei tratti di linea erroneamente tracciati facendo il percorso inverso con il mouse (ripassando cioè, sempre a pulsante premuto, sul tratto da cancellare). Le linee possono intersecarsi e perfino passare sotto elementi atomici: questi usi della penna non sono però consigliabili a causa della scarsa leggibilità del diagramma risultante. Tracciando linee troppo lunghe (con troppi angoli) si può incorrere in situazioni d'errore.

5.3 Cancellazione di linee ed elementi

L'uso dell'atomo gomma serve a rimuovere dal piano di lavoro elementi o linee erroneamente disegnati. Per cancellare un elemento oppure una linea, basta selezionare la gomma dalla palette nel modo usuale, e premere il pulsante del mouse dopo aver posizionato il cursore sull'elemento o sulla linea da rimuovere. Per gli elementi input, i quali sono sempre numerati automaticamente all'atto del disegno, l'operazione di cancellazione comprende la memorizzazione su uno stack del numero dell'ingresso rimosso. Un successivo disegno di un input verrà associato al numero precedentemente cancellato. In generale, i numeri degli input cancellati compariranno in ordine inverso quando si disegneranno nuovi input.

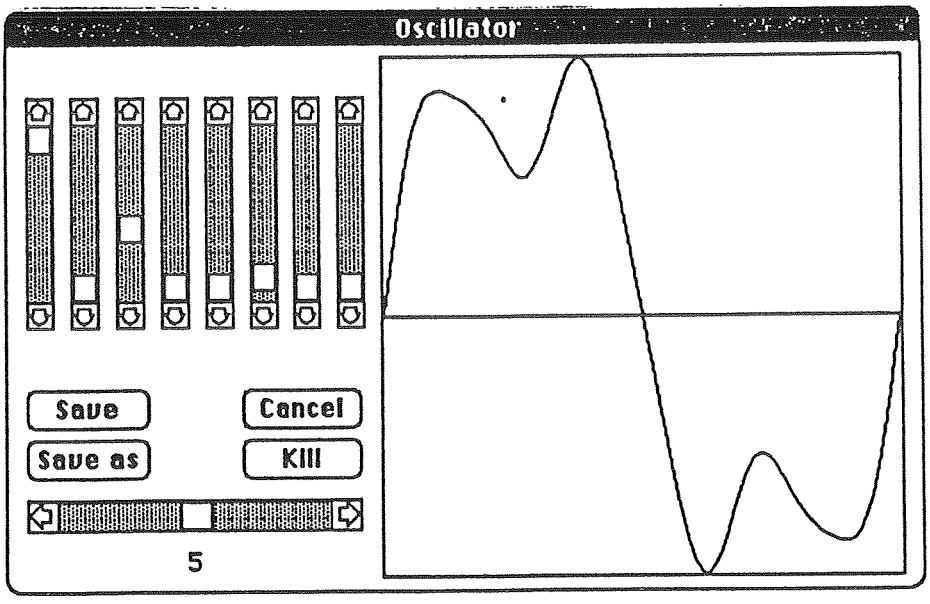
Nelle operazioni di cancellazione bisogna prestare attenzione al fatto chè il programma opera un controllo di consistenza sul disegno, dopo che un elemento è stato rimosso: dopo la cancellazione, vengono rimosse anche tutte le linee che ad esso affluivano o che ne partivano. La rimozione di una linea inoltre provoca la cancellazione di tutte le linee che iniziavano su un suo punto angolare. Con queste semplici regole, un'unica operazione di cancellazione può provocare una catena di aggiornamenti che bisogna saper prevedere per non provocare rimozioni indesiderate.



Nella figura, la cancellazione dell'atomo somma provocherebbe la scomparsa delle linee a, b e c. La linea d, che prende origine in uno spigolo della c, verrebbe rimossa anch'essa. Nello schema rimarrebbero quindi le sole linee e ed f.

5.4 Oscillatori

Una volta posizionato un atomo oscillatore sul piano di lavoro, è possibile definire la sua forma d'onda in termini del suo contenuto spettrale delle prime 8 armoniche. Un doppio click del mouse su un atomo presente sul piano di lavoro provoca l'apertura di una finestra di definizione dell'oscillatore stesso:



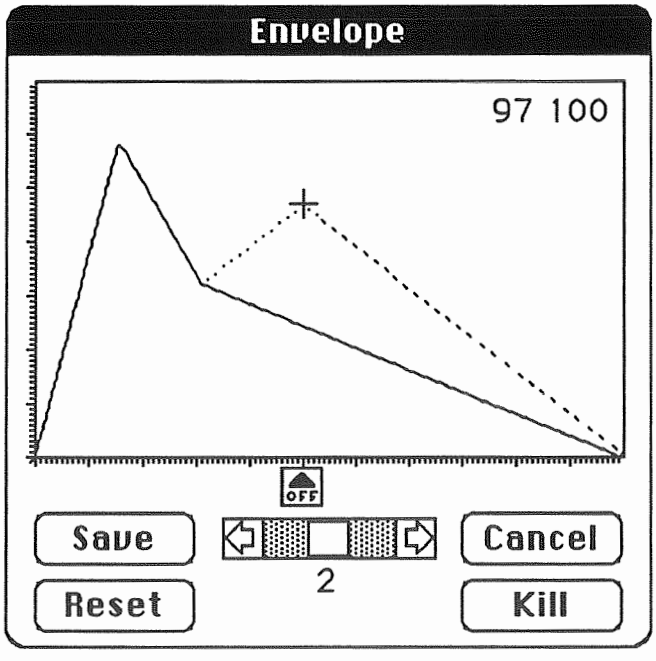
I controlli a cursore verticali servono a definire singolarmente le arm dell'onda, e dopo ogni spostamento del cursore il grafico risultante viene ritracciat parte della finestra dedicata alle onde. Inoltre sono presenti quattro pulsanti (S SAVE AS, CANCEL e KILL) dal significato

SAVE	Assegna all'elemento su cui è stato fatto double-click l'onda presente in quel m sullo schermo. E' utilizzato per scegliere elementi di libreria predefiniti, o per aggi alla libreria un elemento definito. L'aggiunta di un nuovo oscillatore avviene nell posizione della libreria.
SAVE AS	A differenza del Save, assegna all'onda corrente il primo posto libero in libreria l libero da una precedente cancellazione. Per il resto, il comportamento è analogo.
CANCEL	Termina immediatamente la definizione dell'elemento.
KILL	Serve a cancellare dalla libreria l'onda corrente. Viene chiesta una verifica prima di p termine l'operazione.

ed un cursore orizzontale che serve a selezionare eventuali oscillatori predefiniti.

5.5 Inviluppi

La fase di definizione degli inviluppi, analogamente a quella degli oscill consiste nella definizione grafica dell'elemento e nella gestione di una libreria asso Un "doppio click" del mouse su un atomo di tipo "Env" posizionato sul piano di la provoca l'apertura du una finestra mediante la quale è possibile definire una segmentata che tracci l'andamento volumetrico desiderato.



Finestra per la definizione dell'Inviluppo.

Le coordinate riportate nell'angolo in alto a destra di questo riquadro sono quelle del punto in cui si trova il cursore mentre si muove all'interno dell riquadro stesso, questo punto diventa un "punto di rottura" facendo un "click" col muose; l'effetto visivo del comando corrisponde alla effettiva modifica riportata sull'andamento dell'inviluppo: la linea tratteggiata che fino a quel momento univa il cursore ai due piu' vicini punti di rottura A e B le cui ascisse limitavano inferiormente e superiormente quella del cursore viene disegnata mentre viene cancellato il segmento che univa A a B. Oltre alla possibilità di aggiungere punti di rottura esiste naturalmente anche la possibilità di eliminare quelli non desiderati. Per determinare il punto di "Note Off", si utilizza il simbolo visibile subito sotto l'asse delle ascisse, facendolo scorrere per mezzo del mouse sino al punto desiderato.

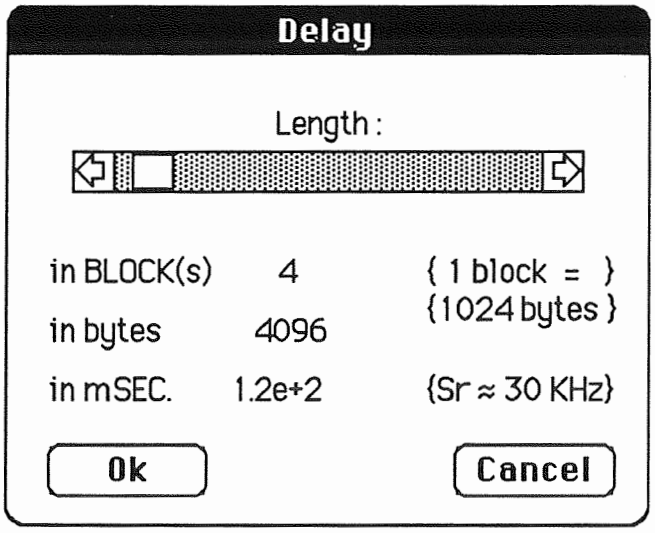
Infine anche in questo caso, come succedeva per le forme d'onda, e' possibile memorizzare tutti questi dati e creare un archivio di inviluppi; la parte della finestra in basso predispone i comandi a questo scopo: si notano alcuni "bottoni di controllo" da utilizzare per il salvataggio e la cancellazione, ed una sbarra di scorrimento (*scrollbar*) viene utilizzata per la ricerca all'interno dell'archivio.

La finestra presentata ha una zona dedicata al disegno dell'inviluppo, una barra orizzontale con significati analoghi a quelli della stessa barra dell'oscillatore, e quattro pulsanti (SAVE, CANCEL, RESET, KILL). L'unico pulsante non introdotto nell'oscillatore, il RESET, ha la funzione di azzerare il disegno presente in quel momento sullo schermo.

Per definire graficamente gli inviluppi, è sufficiente agire con il mouse sulla zona del disegno premendo il pulsante ogni volta che si desidera l'inserimento di un nuovo spigolo. Il programma mostra con una linea continua i segmenti esistenti fino a quel punto, e segue con una linea tratteggiata i movimenti del cursore. Un segmento può essere spezzato in due e da esso possono essere ricavati due segmenti, ma una volta stabilito uno spigolo non è possibile eliminarlo (se non tramite reset). Per questo motivo, il disegno di ogni spigolo deve essere accurato. La barra orizzontale ha la funzione di spostamento all'interno della libreria. Le coordinate in alto a destra nella finestra di definizione grafica indicano la posizione del cursore istante per istante.

5.6 Delay

Un doppio click su un elemento delay provoca l'apertura di una finestra che consente, tramite l'uso di un unico cursore orizzontale, di specificarne la lunghezza.



Finestra per la definizione del Delay

Contestualmente allo spostamento di questo cursore, vengono mostrati i va blocchi, in bytes e in millisecondi del delay. Due pulsanti, OK e CANCEL hai funzione di confermare o di abbandonare la definizione dell'elemento.

5.7 I sottografi

Un doppio click su un atomo sottografo ha un comportamento diverso a se che l'elemento sia stato precedentemente assegnato o meno. Se non è ancora stat nessuna assegnazione all'elemento atomico, viene presentata una finestra di (*minifinder*) che illustra tutti i files di tipo *codice* presenti. L'utente può spostarsi directories nel modo usuale, e può selezionare uno di questi files per asso all'elemento su cui è stato effettuato il doppio click. A questo punto, viene presen nome e il numero di ingressi dell'algoritmo selezionato, con tre pulsanti di scelta ZOOM, CANCEL) il cui significato è il seguente:

OK	serve a confermare la scelta effettuata
ZOOM	permette di avere una visione, in formato ridotto, dello schema grafico che ha gen codice prescelto. In questa finestra di zoom viene mostrato anche il nome dello s grafico.
CANCEL	serve a rinunciare alla selezione: l'elemento non viene assegnato.

Se l'elemento è stato precedentemente selezionato, la finestra che appare il solo il nome e il numero di ingressi unitamente ai due pulsanti (OK, ZOOM) già mc nel caso precedente. Una volta effettuata una scelta, infatti, non è possibile modif (non esiste il pulsante CANCEL): questo avviene perchè la modifica di un a sottografo può comportare anche la modifica del numero di ingressi, rend

potenzialmente non valide alcune delle linee tracciate sullo schema che affluiscono all'elemento. Ovviamente, un modo per cambiare il file di codice associato ad un sottografo è quello di cancellare l'elemento dal piano di lavoro esplicitamente tramite la gomma. Ridisegnandolo nuovamente si ha infatti la possibilità di riassegnarlo in modo diverso.

5.8 Archiviazione di schemi grafici: il menu File

Questo menu consente le normali operazioni di gestione dei files su disco, per quanto riguarda gli schemi prodotti dall'applicazione. Le voci del menu sono quelle tipiche di quasi tutte le applicazioni sviluppate su Macintosh:

NEW	permette di resettare l'editing del documento corrente, cancellando gli elementi e le linee dallo schermo e assegnando al nuovo schema il titolo "Untitled". Prima di eseguire le cancellazioni, il programma chiede la conferma che non si voglia salvare l'editing corrente.
OPEN	tramite questa opzione, è possibile accedere agli schemi precedentemente salvati su disco. Su questi schemi si possono effettuare modifiche, compilazioni, ecc. La modalità con la quale l'utente specifica il file da aprire è la tipica finestra del *minifinder* presente anche in tutte le altre parti dell'applicazione in cui bisogna accedere a files su disco.
CLOSE	Attualmente, questa opzione svolge le stesse funzioni del NEW. Infatti, un solo documento alla volta può essere aperto dall'applicazione, per cui l'operazione di New implica una Close del documento corrente.
SAVE	Procede al salvataggio dello schema corrente.
SAVE AS	Salva lo schema corrente consentendo all'utente di specificare il nome (e la cartella) in cui salvarlo.
QUIT	E' il comando di uscita dall'applicazione. Registra su disco le librerie le cui modifiche sono state effettuate solo in memoria (Envelops File, Code File) e chiede conferma prima di chiudere eventuali schemi presenti in quel momento sul piano di lavoro.

5.9 Il menu Edit

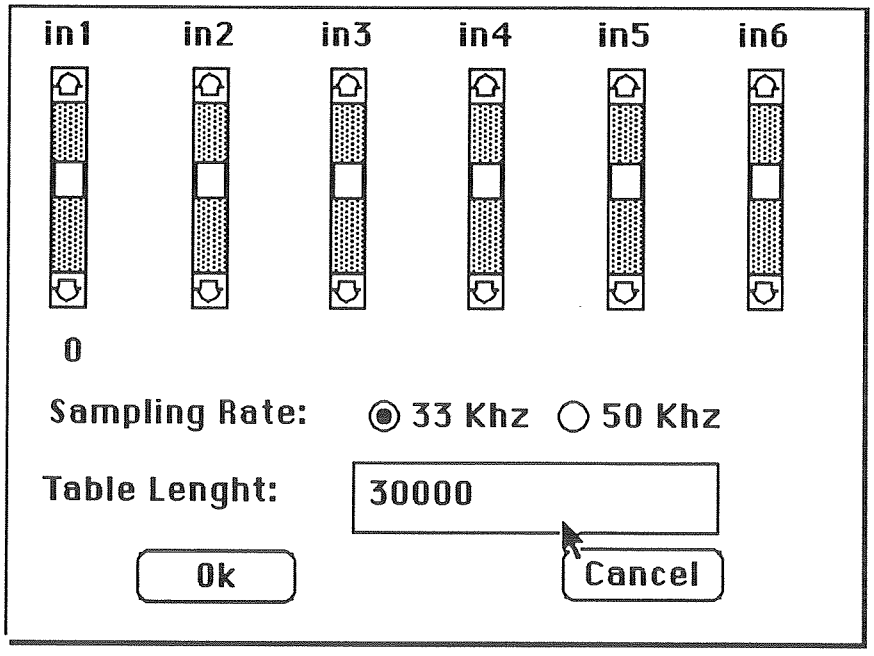
In questo menu sono presenti le usuali opzioni standard delle applicazioni su Macintosh (UNDO, CUT, COPY, PASTE) e due opzioni per modificare l'aspetto grafico della finestra di lavoro: PATTERN e ARROWS. Queste ultime consentono di eliminare la griglia di puntini nel piano di lavoro (PATTERN) e di disegnare le frecce con cui termina una linea solo laddove è indispensabile (ARROWS). Le prime quattro opzioni standard di editing non hanno una funzione specifica all'interno di questa applicazione, e sono presenti solo per l'eventuale loro utilizzo nell'ambito di accessori di scrivania, richiamabili in modo usuale tramite il menu

6 Generazione di campioni sonori e compilazione degli

Lo scopo principale di questo strumento, ricordiamolo, è quello di poter ge il codice macchina di un opportuno microprocessore DSP corrispondente ad un alg di sintesi disegnato con i meccanismi fin qui descritti . Abbiamo tuttavia considera mettere a disposizione una funzione intermedia che consente di verificare il t sonorità ottenibile senza ricorrere all'uso di apparati DSP esterni. Il calcolo dei ca del segnale relativo ad un intervallo di tempo prefissato, nel gergo chiamato "can sonoro", è infatti ottenibile eseguendo la simulazione dell'algoritmo di sintesi all'i dell'editore stesso attraverso la chiamata "coordinata" di procedure corrisponder funzioni specifiche di ogni singolo atomo. I campioni numerici così calc memorizzati in un buffer possono essere inviati a dei comuni convertitori D/A, op campionatori commerciali opportunamente collegati al calcolatore stesso.

6.1 Campioni sonori

Tramite la voce presente nel menu bar **SHOW CONSOLE**, è possibile richi sullo schermo una finestra contenente una serie di cursori verticali per l'assegnazio valori agli input, due pulsanti di tipo *radio button* per la scelta della freque campionamento, e un campo editabile per la scrittura della lunghezza della tab generare (numero di campioni da produrre).



Console

I cursori attivi sono solo quelli relativi agli input effettivamente presenti sullo schema dell'algoritmo in quel momento attivo: gli altri controlli sono visualizzati ma non sono accessibili nè modificabili dall'utente. Tramite i cursori verticali, l'utente può specificare il valore (intero) degli ingressi presenti sullo schema. Il range è quello tipico -32000÷32000 degli interi su 16 bit, e si può modificare il valore con passo unitario. Mentre si sposta il cursore, il valore attuale viene aggiornato con continuità e appare sotto il cursore stesso. Le frequenze di campionamento possibili sono quelle previste da un particolare campionatore reperibile in commercio ed utilizzato per eseguire prove di laboratorio, il TX16W YAMAHA. E' possibile selezionare la frequenza di campionamento facendo click sul pulsante opportuno (33Khz oppure 50Khz). Il campo Table Lenght è destinato a contenere il numero di campioni, fornito dall'utente, che il programma dovrà generare; questo valore influenza in maniera determinante il tempo impiegato dal sistema in fase di esecuzione. I valori di default che compaiono all'apertura della console sono 33Khz di frequenza di campionamento e 30000 campioni da generare (circa un secondo di suono).

Questa fase viene invocata attraverso l'opzione TRANSMIT del menu Code. La voce TRANSMIT viene resa attiva solo quando esiste effettivamente del codice da eseguire, e rimane attiva fintanto che esso continua ad esistere (quindi anche se si opera un NEW sullo schema corrente). In questa fase, devono essere pronti tutti i parametri della console e stabiliti tutti i collegamenti con il campionatore. Per il collegamento con il campionatore è necessaria una interfaccia MIDI standard [14] collegata ad una uscita seriale del Macintosh: Le uscite MidiIN e MidiOUT di questa interfaccia andranno collegate invece nelle porte MidiOUT e MidiIN, rispettivamente, del campionatore.

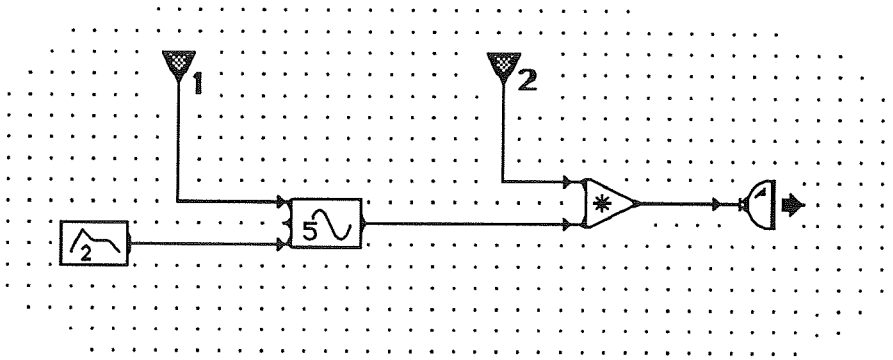
6.2 Compilazione degli schemi

La traduzione automatica degli schemi di algoritmi di sintesi in linguaggio macchina di un microprocessori DSP, è lo scopo finale di questo strumento. Si è previsto di strutturare questa operazione in due fasi distinte: generazione di un metacodice, e generazione del codice macchina vero e proprio. Questa suddivisione in due fasi consente una facile estendibilità della generazione del codice a microprocessori DSP di tipo diverso. Infatti ad ogni linea di metacodice è facilmente associabile una macro predefinita che ne esegue le funzionalità: la traduzione viene così ad essere in realtà il *linkaggio* di macro appartenenti ad una libreria specifica di un particolare micro.

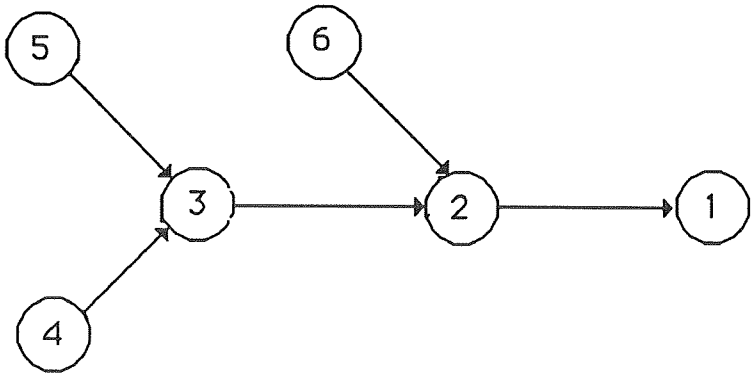
Per invocare l'operazione di compilazione non è necessario aver stabilito i valori all'interno di una console, ovviamente perché il programma compilato girerà sul micro

DSP, ed in tempo reale gli verranno passati tutti i valori parametrici nec dipendentemente dall'*oggetto musicale* da sintetizzare. E' necessario invece che a t elementi su cui è possibile fare doppio click siano stati assegnati dei precisi insieme di valori delle componenti armoniche per gli oscillatori, forma degli inv etc. Nella seguenti figure vengono riportate in maniera schematica e riassuntiva dell'operazione di compilazione.

Lo schema in esempio rappresenta un semplice strumento costituito oscillatore controllabile in frequenza (ingresso P1) e da un inviluppo, con volu uscita controllabile dall'ingresso P2.



Il primo passo della traduzione consiste dunque nella creazione del grafo corrispond



e quindi nella produzione della successione di istruzioni in un particolare linguag noi adottato come metacodice:

```
Mem2  := P2
Mem0  := P1
Mem1  := Env2
Step  := Mem0 *Lt / Sr
Index := Index + Step + zero
Mem3  := TabOsc5 [Index] * Mem1
Mem4  := Mem2 * Mem0
Out Mem4
```

Scelto un particolare microprocessore DSP, come potrebbe esse il TMS320C25 della Texas o il 56000 della Motorola, viene operata la traduzione nel relativo linguaggio macchina semplicemente associando ad ogni istruzione di metacodice una macro opportunamente predefinita.

7 Conclusioni

L'Editore/Compilatore per algoritmi di sintesi descritto in questo rapporto, è stato presentato nelle sue funzionalità essenziali ed in forma prototipale in varie occasioni congressuali e seminariali nel settore (Colloqui di Informatica Musicale, International Computer Music Conference), dove è stato ritenuto un valido contributo allo sviluppo di interfacce ad alto livello per la gestione di applicazioni nel campo dell'elaborazione e della sintesi di segnali audio musicali [15] [16].

Poiché una delle attività del Reparto di Informatica Musicale del CNUCE e del Gruppo di Elaborazione dei Segnali ed Immagini dell'IEI è quella relativa allo sviluppo di stazioni di lavoro musicali basate su personal computer ed apparati multi-DSP è particolarmente sentita la necessità di portare a compimento la realizzazione dell'Editore/Compilatore Grafico nel complesso delle sue funzioni così come sono state descritte, ed eventualmente migliorarle ed ampliarle dipendentemente dai risultati operativi che risulteranno dalla sperimentazione.

Ringraziamenti

Si ringraziano il Dr. Andrea Lombardini ed il Dr. Enzo Maggi che con le loro tesi di Laurea svolte presso il Reparto di Informatica Musicale del CNUCE e discusse presso il Dipartimento di Scienze dell'Informazione dell'Università di Pisa, hanno dato un sostanzioso contributo alla realizzazione dell'Editore/Compilatore Grafico descritto in questa rapporto.

In particolare, il capitolo 4 di questo rapporto, è stato tratto dalla tesi di E.Maggi "Rappresentazione simbolica e compilazione di algoritmi di sintesi: un sistema di sviluppo", 1989.

Bibliografia

[1] De Poli G., 1983 - *A Tutorial on Digital Synthesis Techniques* - Computer Music Journal (Mit Press) Vol. 7, n. 4

[2] Minnick M., 1990 - *A Graphical Editor for Building Unit Generator Patches* - ICMC90 Proceedings.

[3] Pierce J., 1988 - *La scienza del suono* - Zanichelli Editore S.p.a, Bologna

[4] Moorer J.A., 1977 - *Signal Processing Aspects of Computer Music: a Survey* - Computer Music Journal (MIT Press), Vol 1, n.1 -

[5] Moore F.R., 1977 - *Table Look-up Noise for Sinusoidal Digital Oscillator* - Computer Music Journal (MIT Press), Vol 1, n.2

[6] Mathews M., 1969 - *The Technology of Computer Music* - MIT Press

[7] Chowning J., 1973 - *The Synthesis of Complex Audio Spectra by means of Frequency Modulation* - Journal of Audio Eng. Soc. 21 (7)

[8] Chowning J., Bristow D., 1986 - *FM Theory and Applications* - Yamaha Music Fund., Tokyo

[9] LeBrun M., 1979 - *Digital Waveshaping Synthesis* - Journal of the AES, 27(4)

[10] Cesari C.A. ,1981 - *Application of Data Flow Architetture to Computer Music Synthesis* - Lab. of Computer Science, MIT, (MIT/LCS/TR-257)

[11] Prodi G., 1980 - *Analisi Matematica* - Boringhieri, Torino

[12] Aho A.V., Ullman J.D., 1979 - *Principles of Compiler Design* - Addison/Wesley

[13] Kernighan B.W.,Ritchie D.M.,1985 - *Linguaggio C* - Gruppo Editoriale Jackson, Milano

[14] Perotti G., 1990 - *Tecniche di Interfacciamento: MIDI , Computer e Musica* - Gruppo Editoriale Jackson, Milano

[15] Tarabella L., Bertini G., 1989 - *Un Sistema MultiDSP per l'elaborazione di segnali audio ed un Editore Grafico pre algoritmi di sintesi e di filtraggio* - Atti dell'VIII CIM, Cagliari

[16] Tarabella L., Bertini G., 1989 - *A Digital Signal Processing System and a Graphic Editor for Synthesis Algorithms* - ICMC89 Proceedinds, Ohio, USA