Consiglio Nazionale delle Ricerche

*M*etodologie di analisi di segnali audio orientate al progetto di un sistema per la sintesi additiva basato su chip VLSI dedicati

Graziano Bertini, Massimo Magrini, Leonello Tarabella

B4-34
dic-1999

Metodologie di analisi di segnali audio orientate al progetto di un sistema per la sintesi additiva basato su chip VLSI dedicati.

Graziano Bertini *, Massimo Magrini **, Leonello Tarabella ***

Progetto Finalizzato CNR MADESS II
Sottoprogetto SP3 Architetture e Sistemi VLSI
Linea di ricerca 3.2.2 Sistemi Multimediali

* IEI/CNR
** Collaboratore Esterno
*** CNUCE/CNR

Indice

1 Sommario

Nella nota vengono riportate le problematiche relative ai metodi di analisi/sintesi di segnali musicali per la tecnica additiva, come supporto al progetto di un sistema basato su VLSI dedicati.
Sono state studiate ed adottate le tecniche più adatte al nostro scopo ed implementate con MATLAB due procedure: una per l'estrazione dei parametri spettrali e una per la riduzione dei dati di codifica per la resintesi, sperimentandole su campioni di segnali reali.
I dati di codifica sono poi utilizzati come parametri di input per la messa a punto di un simulatore del chip VLSI che implementa una originale tecnica additiva, come pure per la valutazione qualitativa dei suoni ottenibili.

2 Introduzione

La disponibilità di microprocessori molto veloci, dovuta ai continui progressi nella tecnologia dei circuiti integrati, ha incoraggiato lo sviluppo di algoritmi di elaborazione sempre più complessi e sofisticati in un numero crescente di applicazioni; infatti, oltre che nei settori tradizionalmente trattati (militare, telecomunicazioni, biomedicina, controlli di processo industriali, ecc.) le tecniche digitali si sono oramai diffuse anche nei settori civili nell'audio/video professionale e consumer, in elettroacustica, nelle apparecchiature di supporto all'intrattenimento e molti altri ancora [BM95]. L'applicazione di tali metodologie ha permesso un notevole sviluppo anche nel settore della della Computer Music (Informatica Musicale), che nel corso della sua evoluzione ha esplorato vari aspetti collegati in maniera diversa all'attività musicale. Citiamo ad esempio la composizione assistita di testi, l'analisi musicologica, gli studi sulle tecniche di sintesi e di trattamento dei suoni, il progetto e la realizzazione di workstation musicali in grado di offrire alti livelli di efficienza [TB92].
In merito a ciò è bene tenere presente che l'argomento non riguarda solo l'aspetto culturale, scientifico e artistico ma anche quello commerciale. Infatti l'industria nazionale pur avendo sempre occupato una posizione di prestigio nella produzione di strumenti musicali tradizionali, è rimasta in ritardo per quelli di tipo elettronico rispetto a nazioni che hanno mostrato maggior interesse per il settore.

Nell'ultima decade, si è osservata una ripresa nell'impegno di proposte e un aumento di risorse da parte di alcune ditte italiane (GEM, Bontempi-Farfisa, ecc.) assieme ad alcuni laboratori di ricerca pubblici (Università e CNR) con l'obiettivo di riguadagnare quote di mercato anche nella strumentazione musicale di tipo elettronico. Nell'ambito di un progetto analizzato del CNR per la microelettronica è stata accolta e finanziata una proposta avanzata dagli istituti IEI/CNUCE, attivi da anni sui sistemi per la sintesi sui segnali audio, in collaborazione con il Dipartimento di Ing. Elettronica Informatica delle Telecomunicazioni dell'Università di Pisa con la coordinazione dell'istituto IRIS (laboratorio di ricerca del gruppo Bontempi), per la realizzazione di un sistema basato su VLSI dedicati per l'implementazione di una tecnica di sintesi additiva su larga scala per ottenere un elevato grado di polifonia [BRST&B97].
Il presente lavoro si occupa di alcuni aspetti di affinamento e ottimizzazione architetturale del progetto del chip VLSI e di vari supporti software di verifica e di simulazione utili allo sviluppo del sistema di cui sopra.

2.1 Cenni alle tecniche e ai sistemi per la sintesi audio

Come è noto, le tecniche di sintesi sono basate su determinati modelli di tipo matematico, fisici o percettivi; in generale sono espresse per mezzo di opportuni algoritmi di calcolo che permettono di ottenere delle forme d'onda equivalenti ai segnali in banda audio con caratteristiche assegnate [Pol83]. Citiamo ad esempio la Sintesi Additiva , la Sintesi per Modulazione di Frequenza (FM), la Sintesi Sottrattiva, la Sintesi per Modelli Fisici (MF). Con la tecnologia analogica era stato possibile realizzare un certo tipo di apparecchiature usando alcune di tali tecniche, basta ricordare gli organi di tipo Hammond e una serie di sintetizzatori dai più semplici fino a quelli professionali, composti da pannelli con cablaggi complicatissimi (Moog). Con l'impiego di sistemi digitali, l'implementazione delle tecniche di sintesi audio avviene tramite l'esecuzione ciclica di procedure software su calcolatori di tipo general purpose o dedicati. Queste procedure ricevono in ingresso opportuni parametri di controllo per generare delle sequenze numeriche con frequenza di campionamento prefissata, che convertite in analogico, forniscono determinati segnali sonori. Tenendo conto della larghezza di banda (circa 20-20000Hz), dell'elevato numero di forme d'onda elementari (seni, coseni) per formare una nota di un qualsiasi strumento musicale, del grado di polifonia richiesto, del tipo di tecnica di sintesi e di altri requisiti, può essere richiesta l'esecuzione di centinaia di milioni di operazioni al secondo per generare da parte di un sistema digitale complessi segnali sonori in tempo reale.
Infatti, in passato, l'uso di elaboratori anche molto potenti obbligava per la maggior parte delle applicazioni musicali a operare in tempo differito, utilizzando le prime versioni di linguaggi musicali (gli storici MusicV, Csound ecc). In un primo momento si impegnava l'elaboratore nel calcolo e nella memorizzazione delle sequenze numeriche in memorie di massa e successivamente si procedeva alla lettura, alla conversione digitale/analogica e alla memorizzazione su supporto analogico del segnale audio (le cosiddette "composizioni per nastro").
Per consentire l'elaborazione e la sintesi in tempo reale, dagli anni '60 in poi, sono state studiate delle soluzioni consentite dagli aggiornamenti della tecnologia, che hanno permesso di realizzare sistemi dedicati basati su architetture specializzate, sia da connettere come coprocessori ad elaboratori general purpose e in alternativa, di funzionare autonomamente. Tra le apparecchiature sperimentali va ricordato il TAU2 [BCD77][BBG80], realizzato presso l'IEI/CNR di Pisa negli anni '70, che veniva controllato da calcolatori di tipo main-frame dell'IBM funzionanti in time-sharing; mediante originali soluzioni hardware analogico-digitale si potevano generare e gestire gli in viluppi di 84 onde sinusoidali implementando una tecnica additiva con una polifonia di 12 voci e tre timbriche diverse allo stesso tempo. Altre realizzazioni notevoli sono state le macchine completamente digitali progettate negli anni '80 all'IRCAM (istituto di ricerca musicale di Parigi) denominate 4N (4A, 4B, 4C, 4I, 4X), in grado di generare 64 onde sinusoidali alla frequenza di campionamento di 32 KHz.
Con l'introduzione dei microprocessori cosiddetti per DSP e i nuovi modelli di convertitori analogico/digitali è stato possibile realizzare altre macchine con architettura modulare ed ottenere delle buone prestazioni a costi relativamente contenuti; fra queste citiamo il Sinc Circuitu [Jan91], la stazione con schede PC-conpatibili Leonard-C25 [BF93] e tante altre gestite da calcolatori host di tipo PC. In effetti tutti questi sistemi sono ancora relativamente costosi e perciò prodotti in quantitativi limitati.

2.2 Problematiche realizzative di sistemi VLSI per la sintesi audio

Per cercare di abbassare il costo, recentemente si stanno studiando e sperimentando delle realizzazioni che prevedono il progetto degli oscillatori digitali direttamente in hardware con circuiti realizzati in tecnologia VLSI.
Un primo lavoro ripropone l'implementazione di 128 oscillatori digitali tramite il noto meccanismo della LUT (Look-up table): per poter contenere le dimensioni e il costo del chip entro limiti ragionevoli, gli autori hanno posto delle limitazioni alle dimensioni della tabella e introdotto interpolazioni ai valori letti per avere un rapporto segnale/rumore accettabile nelle onde sinusoidali generate [HFT95].
La proposta che è stata avanzata dall'IEI/CNUCE-CNR e dal DEIT dell'Università di Pisa e di cui ci occupiamo in questo lavoro, prevede la costruzione di un chip per la sintesi additiva che implementa un migliaio di oscillatori realizzati in hardware basandosi su filtri IIR del secondo ordine, risonanti di volta in volta alla frequenza desiderata [BRST&B97].
Tramite una soluzione architetturale cosiddetta "sistolica" e pipeline fra chip, è possibile connetterne una decina in cascata ed ottenere una configurazione che consente di avere a disposizione una quantità di oscillatori dell'ordine delle decine di migliaia (ammesso di riuscire a fornire i parametri di input e gestirli). Dal momento che lo sviluppo hardware del chip VLSI è un'operazione complessa e costosa, in genere la realizzazione effettiva dei prototipi viene attuata solo se poi è prevista la replica di migliaia degli stessi dispositivi e cioè il loro impiego in sistemi per i quali esista un "mercato", (come può essere in effetti quello degli strumenti musicali elettronici). Prima della costruzione del prototipo (fusione del silicio) occorre far precedere una accurata fase di validazione del progetto stesso sia a livello di simulazione delle specifiche funzionali (in genere sequenze di funzioni logico-matematiche) e sia di specifiche hardware (in generale operazioni logico-elettroniche).
Nel nostro caso ciò non è sufficiente, perché l'output consiste in una "sequenza di campioni" equivalenti a segnali audio, e la natura di questi impone che essi debbano possedere una determinata "qualità sonora" e quindi essere esenti da disturbi, distorsioni e artefatti di vario tipo. Perciò rispetto ai chip, diciamo così, "numerici", impiegati cioè esclusivamente per funzioni di calcolo e/o controllo, dove esistono comunque i soliti problemi di velocità, consumi, affidabilità ecc., la realizzazione del chip di sintesi audio richiede un livello di simulazione in più, corrispondente a quello della verifica della qualità dei suoni ottenuti dopo la conversione digitale-analogica dei campioni in uscita (rapporto segnale/rumore, rispondenza del tipo di suono all'insieme dei parametri posti in input).
Un altro problema da tenere in considerazione, deriva dal controllo dell'elevato numero di segnali elementari prodotti dal chip (e a maggior ragione da più chip); come vedremo ogni singolo segnale richiede almeno tre parametri da aggiornare continuamente a microlivello (ogni 22 microsec.) a cui si aggiunge la gestione dei controlli a livello degli "inviluppi d'ampiezza" e a livello dei "cambi di nota". Questi dati devono essere forniti ai chip di sintesi tramite una apposita unità di interfaccia (verso un host, tipicamente un PC, o verso un dispositivo di input musicale, tipo tastiera MIDI) che deve gestire il flusso dei dati ed evitare che si verifichi il cosiddetto "collo di bottiglia". Nella proposta di cui sopra, in prima istanza, questa unità era stata pensata essere basata su un microprocessore per DSP general purpose; in una successiva rielaborazione del progetto, invece, è stato ipotizzato l'impiego di un altro chip VLSI dedicato, realizzato per questo compito di controllo. Elenchiamo di seguito alcune fasi del lavoro condotto negli anni '98 e '99 dall'UO dell'IEI in collaborazione con ricercatori del DEIT, in parte descritte nel seguito della relazione:

- studio e realizzazione di procedure software per l'estrazione, la codifica e il trattamento di parametri caratteristici relativi a determinati segnali acustici " campione" (non essendo tali dati reperibili, allo stato attuale, in accordo alle nostre esigenze);
- verifica, con una tecnica additiva di tipo classico, simulata su host, che la modalità di estrazione dei parametri con il tipo di analisi adottato sui segnali campione sia valida confrontando i suoni ottenuti con quelli reali;
- verifica, utilizzando il tool sviluppato per simulare la tecnica additiva effettivamente implementata dal chip, che i campi di variabilità dei parametri ipotizzati nella prima release di progetto del chip siano validi per una buona qualità dei suoni (verifica effettuata dagli autori in collaborazione di musicisti esperti di informatica musicale);
- proposta di una strutturazione della codifica dei parametri adatta all'architettura del chip di controllo che dovrà gestire la sintesi audio, da individuare assieme ai progettisti.

3 La Sintesi additiva

La tecnica della "*sintesi additiva*" consiste di una somma istante per istante di un numero finito di onde sinusoidali delle quali sono definiti i parametri di frequenza, ampiezza e fase. Tale tecnica non ha i limiti della sintesi per FM, permette teoricamente la creazione di qualsiasi suono immaginabile e inoltre ha il vantaggio che si conosce un metodo di analisi appropriato, basato sulla trasformata di Fourier , che trasla i segnali audio dal dominio del tempo in un insieme di onde seno nel dominio della frequenza. In certi casi si ottiene una drastica riduzione della quantità di dati per rappresentare un segnale. Infatti, facendo un esempio, un segnale stazionario (periodico) di un secondo campionato a 44,1 KHz richiede 44100 campioni per essere rappresentato nel dominio del tempo; nel dominio della frequenza, invece, bastano alcune decine di parametri per identificare lo stesso segnale (ignorando le componenti, che hanno un'energia al di sotto di certi livelli e che sono ininfluenti a livello pratico). Le manipolazioni nel dominio della frequenza richiedono spesso meno potenza computazionale che le corrispondenti manipolazioni nel dominio del tempo. Un ovvio esempio è il filtraggio: complessi filtri, che cambiano dinamicamente, possono essere implementati facilmente nel dominio della frequenza, mentre nel dominio del tempo è difficile e spesso impossibile realizzarli.
In termini matematici, l'espressione analitica che la rappresenta è la seguente:

$$y(n) = \sum_{i=1}^{k} A_i(n) sin(2\pi n \cdot \frac{F_i(n)}{F_c} + \phi_i(n)) \qquad (3.1)$$

dove:
A_i (n) è l'ampiezza dell'onda i-esima nel campione n-esimo;
F_i (n) è la frequenza dell'onda i-esima nel campione n-esimo;
ϕ_i (n) è la fase dell'onda i-esima nel campione n-esimo;
F_{camp} è la frequenza di campionamento.
Come si può notare l'espressione (3.1) ha una certa somiglianza con lo sviluppo in serie di Fourier ed è per questo che viene impropriamente chiamata "sintesi additiva di Fourier". In realtà le due relazioni sono concettualmente e analiticamente diverse, infatti:
- l'espressione (3.1) è una tecnica di sintesi mentre la serie di Fourier è una formulazione di analisi,
- la sommatoria è fatta su un numero finito di termini, mentre la serie di Fourier è una somma di infiniti termini (questo è valido teoricamente, in pratica basta considerare un numero sufficientemente elevato di termini),

- le frequenze $F_i(n)$ delle varie sinusoidi non sono multiple della fondamentale (come nella serie di Fourier), ma in genere sono funzioni del tempo e quindi variabili indipendentemente le une dalle altre. In questo modo si possono generare degli spettri qualunque, suoni inarmonici (nel caso le frequenze presenti non siano multiple della fondamentale) o addirittura rumori e suoni con componenti di rumore. Inoltre, si può dire, che la sintesi additiva è una tecnica con parametri deboli, in quanto variando l'inviluppo di ampiezza di una sola armonica il cambiamento può risultare completamente non percepibile [Jaf95]. E' però possibile, controllando un insieme di parametri (metaparametri) tramite un algoritmo opportuno, simulare quelli di un'altra tecnica. Le caratteristiche appena descritte, non mettono in luce il problema fondamentale che questa sintesi presenta: volendo ottenere suoni di buona qualità occorrono, per ognuno, alcune decine di componenti; considerando fasce sonore con polifonia elevata possono essere necessarie anche migliaia di sinusoidi. Perciò il costo computazionale eccessivo che ne consegue ha impedito l'affermazione su scala industriale di hardware che implementa tale tecnica.

3.1 Implementazioni della sintesi additiva

Di seguito vengono descritte in maniera sommaria alcune tecniche per la sintesi additiva: la prima sfrutta il controllo dell'intero spettro del segnale nel dominio della frequenza, mentre le altre controllano la generazione delle singole componenti.

IFFT

Un approccio per implementare in modo indiretto la sintesi additiva è quello basato sulla trasformata inversa di Fourier (IFFT). Singole porzioni di segnale (frames) della durata di alcuni millisecondi sono ottenute con una sequenza di antitrasformate di Fourier di descrizioni nel dominio della frequenza del segnale da generare, cioè ampiezza e fase delle parziali negli intervalli di tempo considerati. Il segnale è ottenuto sovrapponendo tramite cross-fading porzioni consecutive di segnale generato, generalmente sovrapposte per almeno il 50%.
Questo procedimento implica un elevato costo computazionale dovuto alla alta risoluzione in frequenza necessaria alla sintesi di segnali musicali.
Per applicazioni pratiche si sono studiati metodi di riduzione delle righe di partenza della FFT^{-1} con criteri dettati dalla psicoacustica [RD92]. In un'altra proposta [BB93] si riduce fortemente il numero di righe su cui vengono calcolate le FFT^{-1} in modo da permettere una realizzazione real-time senza far uso di processori DSP general-purpose o di soluzioni hardware dedicate.

Metodo della look-up table

L'algoritmo è basato sull'uso di una tabella (Look Up Table, [TB92]) di n valori della funzione seno (per motivi di efficienza è impensabile calcolare il seno in modo diretto): questi sono espressi con una precisione di m bit e coprono, di solito equamente distribuiti, un periodo della forma d'onda o la metà. Entrambi i parametri n e m (soprattutto il primo) influenzano il rapporto segnale-rumore relativo all'uscita.
La lettura della tabella viene effettuata ciclicamente con un passo che aumenta all'aumentare della frequenza del segnale da generare, in modo da simulare tabelle di lunghezza variabile (in realtà la tabella è sempre la stessa). Per ridurre l'approssimazione e migliorare la qualità del suono, si può ricorrere ad un'interpolazione lineare tra elementi consecutivi della tabella.
Considerando che per la produzione di un singolo campione (per ogni intervallo di campionamento) i valori memorizzati devono essere letti un numero di volte pari alle sinusoidi

generate (quindi anche diverse migliaia), poi essere moltiplicati per le rispettive ampiezze e sommati fra loro, si ripresenta il problema dell'enorme costo computazionale di questa tecnica; a questo va aggiunto anche quello relativo alla procedura di memorizzazione della fase istantanea che avviene per garantire la continuità interframe. Tale costo aumenta linearmente con l'aumentare del numero di armoniche, e questo ha spinto a cercare delle ottimizzazioni particolari.

L' oscillatore digitale con filtri IIR del secondo ordine

Questo metodo, alla base del progetto di cui tratta la presente relazioni, non è stato utilizzato in maniera così diffusa come gli altri metodi citati. E' basato sulla generazione di una sinusoide utilizzando un filtro IIR del secondo ordine marginalmente instabile. Come illustrato in Figura (4.1 (a)) scegliendo opportunamente i poli sul cerchio unitario nel progetto del filtro IIR è possibile generare un'onda seno a qualunque frequenza desiderata (Figura (4.1) (b)).

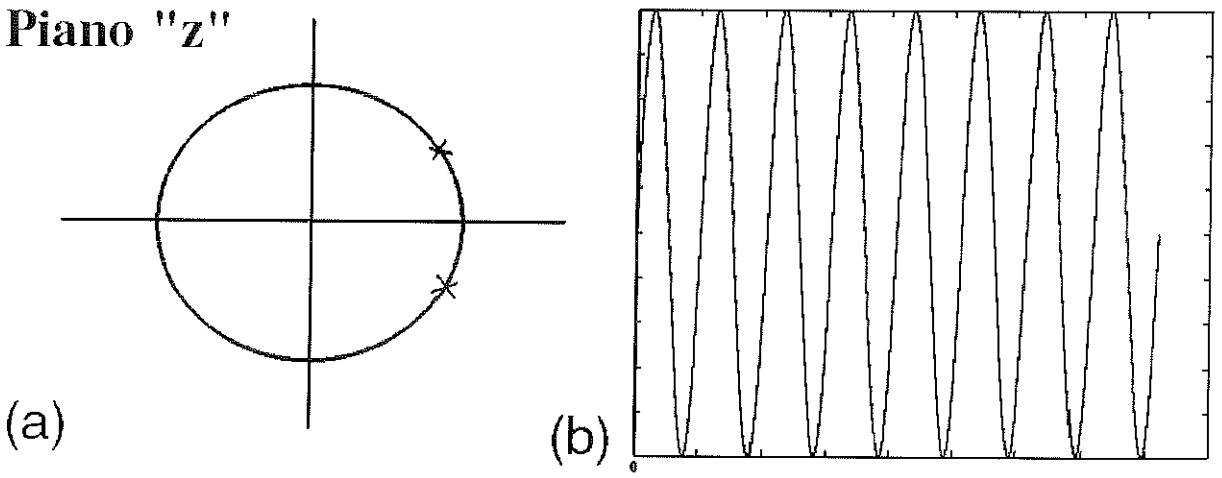

Figura 4.1. Diagramma dei poli e risposta impulsiva del filtro IIR in questione

Diamo un accenno al principio di funzionamento.
Nel dominio del tempo l'equazione alle differenze che rappresenta tale filtro può essere ricavata dalle formule di prostaferesi.

$$\cos[\Omega(k+1)] = \cos\Omega\cos(k\Omega) - \operatorname{sen}\Omega\operatorname{sen}(k\Omega)$$

$$\cos[\Omega(k-1)] = \cos\Omega\cos(k\Omega) + \operatorname{sen}\Omega\operatorname{sen}(k\Omega)$$

da cui sommando membro a membro, si ottiene la seguente equazione:

$$\cos[\Omega(k+1)] = 2\cos\Omega\cos(k\Omega) + \cos[\Omega(k-1)]$$

che è della forma tipica

$$y(k+1) = \alpha y(k) + \beta(k-1)$$

con

$$\alpha = 2\cos\Omega$$

$$\beta = -1$$

In pratica l'espressione ricavata esprime l'uscita (k+1)-esima della sinusoide in funzione delle due uscite immediatamente precedenti senza avere in questo caso nessun ingresso. Partendo dalla trasformata z nel dominio della frequenza:

$$\cos[\Omega k + \varphi)] \Leftrightarrow \frac{\cos\varphi \cdot z^2 - \cos(\varphi - \Omega) \cdot z}{z^2 - 2\cos\Omega \cdot z + 1} \qquad (4.2)$$

è immediato ricavare i coefficienti che deve avere il polinomio caratteristico del sistema in considerazione; la frequenza di oscillazione è rappresentata da $\Omega = 2\pi T_c \cdot f$ dove T_c è il periodo di campionamento della cosinusoide.
Effettuando opportuni calcoli otteniamo che l'uscita y nel dominio della frequenza assume la seguente forma:

$$y(z) = \frac{x_2(0)z^2 - x_1(0)z}{x^2 - 2\cos\Omega \cdot z + 1} \qquad (4.3)$$

da cui si possono ricavare per confronto con la (4.2), le condizioni iniziali da imporre per avere un cosinusoide con frequenza e fase volute. La Figura (4.2) illustra lo schema a blocchi dell'algoritmo.

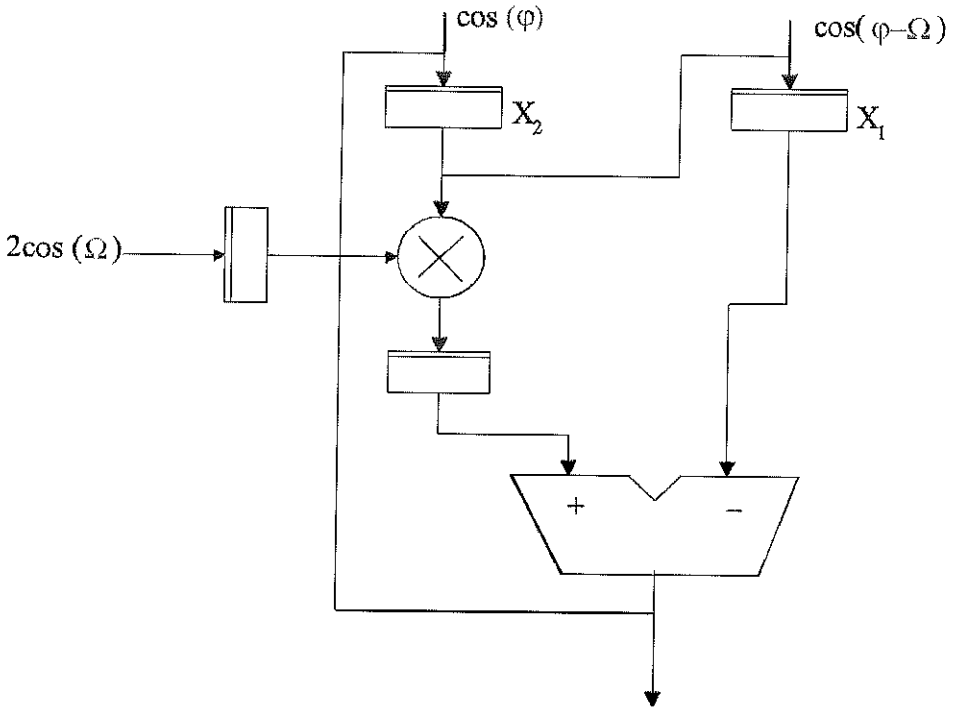

Figura 4.2. Schema di principio utilizzato nel progetto

3.2 Alcuni sistemi basati sulla sintesi additiva

Nonostante le sue notevoli possibilità la sintesi additiva, come già sostenuto, non ha avuto a tutt'oggi uno sviluppo e una diffusione appropriata a causa dell'elevata potenza di calcolo necessaria per la generazione dei suoni e dei costi troppo elevati delle sue realizzazioni. I pochi sistemi sviluppati sono stati essenzialmente di tipo sperimentale e utilizzati per supporti "real-time" in ricerche su analisi e sintesi di suoni [BCD77].
Anche le macchine basate su processori programmabili per DSP, pur essendo stati proposti a costi non proibitivi [Jan91], hanno avuto una scarsa diffusione. Al contrario, a livello commerciale, hanno avuto successo le macchine per lo più basate sul campionamento e sulla tecnica FM, caratterizzate da buona qualità sonora, non programmabili e con un rapporto "prestazioni/costo", tutto sommato, soddisfacente. La riduzione dei costi per le architetture VLSI ha di nuovo stimolato le ricerche per l'implementazione della tecnica additiva: una prima realizzazione è quella di un chip basato ancora sul metodo LUT (ASWS) [HFT95], mentre una recentissima proposta è quella discussa in questa nota [BRST&B97], che prevede la realizzazione di un chip nel quale gli oscillatori sono basati su filtri digitali IIR del secondo ordine risonanti. Accenniamo brevemente al principio di funzionamento delle macchine prima citate, per evidenziare l'evoluzione nelle prestazioni che si è avuta in pochi lustri.

Il TAU2

Il terminale audio TAU2, realizzato all'istituto di Elaborazione dell'Informazione (CNR) di Pisa, era composto da due sottounità, una interamente digitale di controllo e di interfaccia con l'elaboratore, e l'altra digitale-analogica per produrre segnali. La sintesi additiva veniva effettuata con controllo parametrico della frequenza fondamentale e delle singole armoniche, e del volume totale del suono; le onde sinusoidali erano ottenute in modo analogico e selezionate digitalmente. Le caratteristiche strutturali e operative del TAU2 furono scelte in base alla disponibilità dei componenti elettronici di allora, privilegiando l'attività di esecuzione di brani polifonici in tempo reale, invece di quella della ricerca timbrica. Mediante originali soluzioni circuitali furono ottenute prestazioni quali:

- risoluzione in frequenza pari a 1=3 di semitono temperato;
- polifonia a 12 voci e politimbricità su tre canali di timbrica diversa;
- sintesi armonica a 7 componenti sinusoidali e spettri variabili a intervalli di tempo al di sotto dei 10ms per ognuna delle tre timbriche, per un totale di 84 oscillatori.

Si vuole precisare che il TAU2 è un sistema sperimentale e di ricerca; la codifica e le matrici delle frequenze, ampiezze e durate venivano manipolate da un potente calcolatore, time sharing, della serie IBM 360 e successivi, certamente non proponibili in sistemi musicali commerciali negli anni '70, '80. Il TAU2 si trova attualmente presso il museo degli strumenti musicali del Conservatorio di Musica L. Cherubini di Firenze.

3.1.2 The Sine Circuitu

Tra le diverse macchine basate su DSP citiamo questo sistema perché in effetti offriva un supporto hardware/software, orientato alla gestione della tecnica additiva. L'architettura del sistema prevede una tipica struttura modulare, basata su schede per DSP. Il controller, che è basato, invece, su un processore RISC, guida il sistema e comunica con l'host e con altre periferiche come i disk drives. L'interfaccia SCBUS sul controller funge da protocollo di comunicazione tra il controller stesso e il bus. Connesse allo SCBUS si trovano poi le speciali schede per l'elaborazione dei segnali, gli SCDSP. Per gli scopi di flessibilità gli autori hanno sviluppato una scheda capace di generare una grossa quantità di onde sinusoidali e con una notevole potenza di calcolo che permette di effettuare sia modifiche nel dominio della frequenza

sia l'analisi del segnale. Ciò era garantito da schede che utilizzavano uno o più chip DSP (Digital Signal Processing) fixed point, a 24 bit i DSP56001/2 della Motorola, che al momento della realizzazione avevano il miglior rapporto qualità prezzo. Ogni SCDSP è in grado di generare circa 240 onde seno a una frequenza di campionamento di 44100 Hz. Il sistema nella sua forma estesa con 34 schede SCDSP può generare circa 8000 onde sinusoidali. Il notevole costo complessivo per tale sistema si aggirava nel 1993 intorno ai 30000 Dollari e ciò non ne ha permesso una larga diffusione.

L'ASWS (Additive sine-wave synthesis)

Progettato presso l'Università di Sheffield (UK), l'ASWS [HFT95] è stato costruito su un ASIC (Circuito integrato per applicazioni specifiche) con lo scopo di fornire, grazie alla sintesi additiva, una alternativa hardware agli approcci più tradizionali per la sintesi del suono. La capacità dell'ASWS è di creare e sommare diverse onde seno con frequenza, ampiezza e fase programmabili sfruttando la nota tecnica della Look-Up Table.

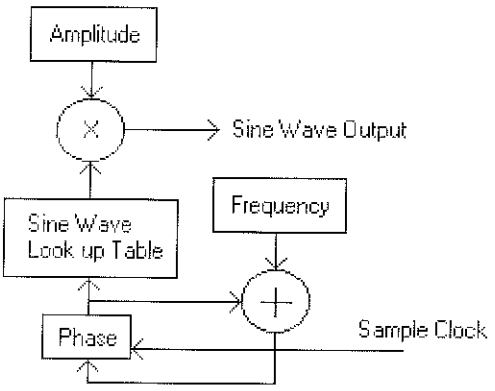

Figura 3.1. Schema dell'algoritmo di generazione delle onde seno

La Figura (3.1) mostra, in forma schematica, il semplice algoritmo che viene utilizzato per produrre onde seno digitalizzate. L'algoritmo usa tre registri che contengono ampiezza, frequenza e fase, e una tabella (Par. 3.1). Effettuando opportune modifiche sulla LUT, gli autori sono riusciti a ridurre drasticamente la quantità di memoria necessaria ad implementare, grazie a delle interpolazioni fra i valori letti, effettuate in hardware. Il progetto è stato disegnato con la tecnologia di e l'area totale del chip è circa 27mm. Un singolo ASIC può generare 127 onde sinusoidali. L'ASWS non è ancora un prodotto commerciale, essendo stato presentato di recente.

4 Architettura del sistema e codifica dei dati

Questa proposta d'implementazione per la sintesi additiva presenta delle aspettative superiori a tutte le precedenti, in quanto un solo chip Synth, in fase di sviluppo può generare, in real-time, 1200 sinusoidi con caratteristiche di buona qualità [BRST&B97]. Le onde seno sono ricavate matematicamente senza usare la LUT (Table Look-Up) e ciò semplifica notevolmente il progetto globale. La lunghezza interna di parola adottata (20 bit) consente di ottenere un elevato rapporto segnale-rumore (– 80 dB), inoltre il sistema prevede la possibilità di impostare i parametri per la sintesi, ampiezza, frequenza e fase a intervalli molto brevi, per sopperire ai rapidi cambiamenti nelle caratteristiche del suono e infine ha la possibilità di ottenere in uscita otto canali indipendenti.

Il problema più rilevante nella realizzazione è la gestione del passaggio dei parametri, che non può essere eseguita direttamente dall'host, in quanto quest'ultimo non è in grado di garantire un flusso continuo di dati, fattore essenziale per l'esecuzione dei brani sonori. La soluzione adottata prevede una sezione di gestione e di controllo CTRL che ha il compito di tradurre i blocchi di comandi che arrivano dall'esterno in parametri accettati dal Synth mantenendo continuo il flusso di dati significativi verso quest'ultimo. L'architettura del sistema risulta quella di Figura 4.1 e descritta in dettaglio in [BRST&B98].

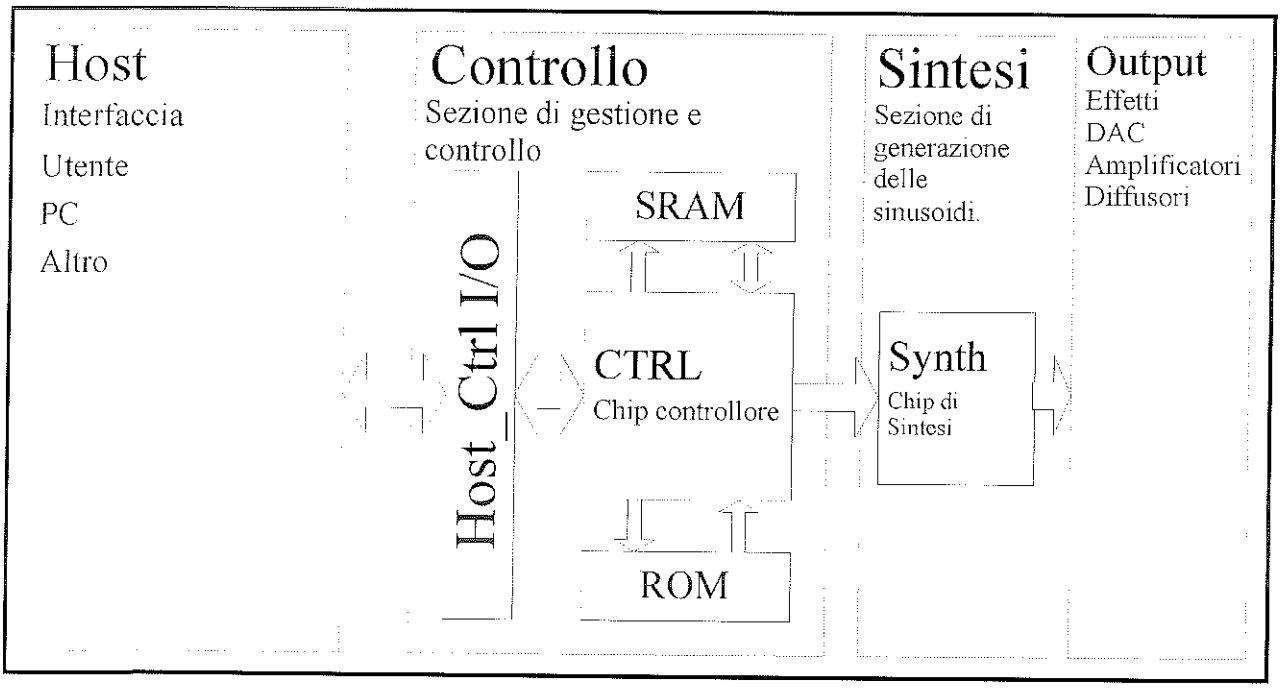

Figura 4.1. Architettura del sistema di sintesi.

Siccome il sistema deve essere in grado di gestire ciò che l'esecutore (o il programma su host) gli impone tramite i comandi, tenendo conto dei vincoli temporali, è necessaria una struttura e una *codifica* opportuna dei parametri significativi che consenta di ridurre al minimo il flusso di dati dall'host verso il controllore .

Quest'ultimo deve operare allo stesso tempo una specie di *decompressione* e riorganizzare i parametri per consentirne il passaggio agli oscillatori in modo continuo. I dati sono stati, quindi,

organizzati in *cluster*, ossia in insiemi di sinusoidi caratterizzate da parametri comuni, utilizzando opportuni comandi per indicare il trattamento riservato ad ognuno dei cluster.

Quando si attiva il cluster, esso rimane nella memoria del controllore, per un certo periodo di tempo, può essere aggiornato e appena non è più utilizzato viene cancellato. L'organizzazione viene effettuata sull'host e i comandi seguono delle regole: tutti i cluster vengono generati con un numero d'ordine progressivo e ad esso vengono associati un certo numero di oscillatori, che poi dovrà alimentare per tutto il tempo della sua allocazione; gli oscillatori associati possono diminuire, ma non aumentare. Quando un cluster viene disallocato, per tutti gli altri, con numero d'ordine superiore, viene decrementato il proprio numero, ricompattando la tabella in modo da ridurre la frammentazione della memoria del controllore (Figura 4.2)). Infatti una eccessiva frammentazione potrebbe impedire l'allocazione di un cluster che non trova nella tabella una zona contigua capace di contenerlo.

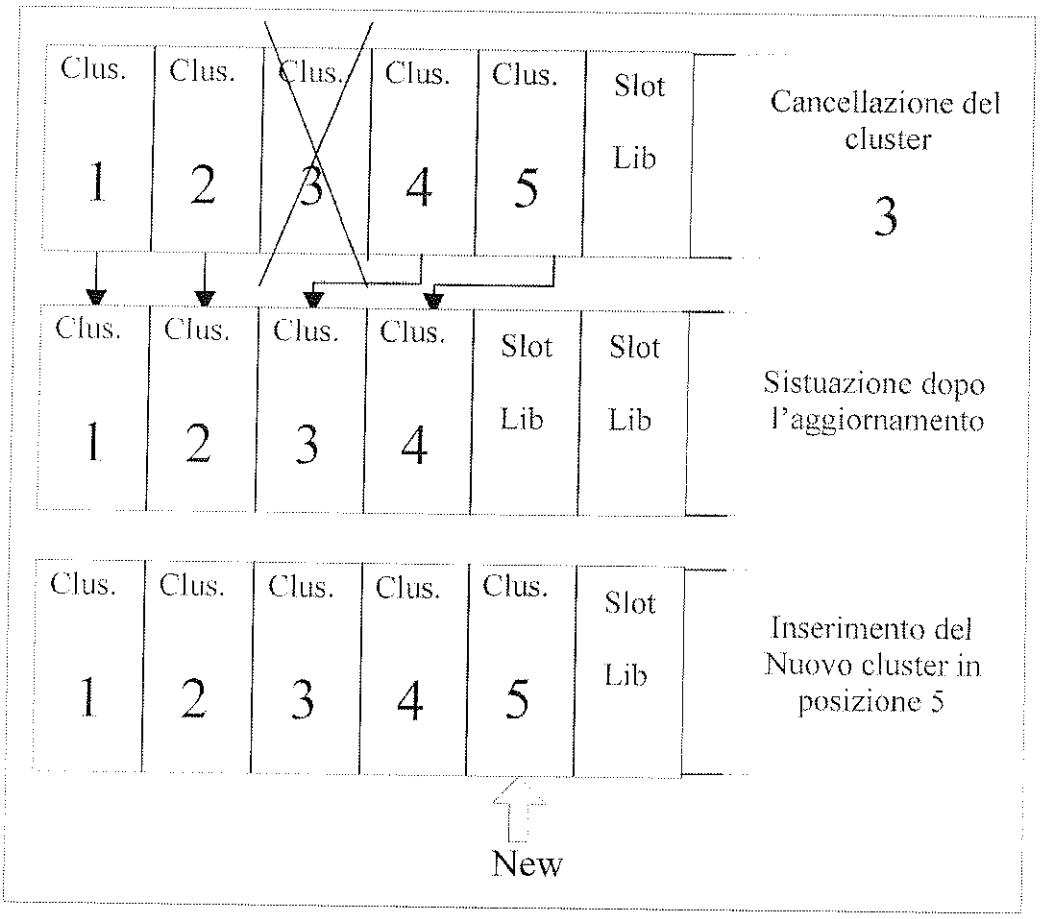

Figura 4.2. Tipico esempio di riorganizzazione dei cluster

Naturalmente il controller verrà progettato per garantire la consistenza tra i dati forniti dall'esterno e quelli memorizzati nella sua SRAM, per questo motivo, in caso di cancellazione, viene aggiornata la numerazione dei cluster in coerenza all'host. Ogni cluster (Figura 4.3(a)) è caratterizzato da un header (intestazione) i cui campi indicano al controllore come comportarsi; in particolare il valore del campo Comando indica:

- 0: annullamento di tutti i campi di un oscillatore e conseguente riduzione delle dimensioni di un cluster. Utile quando occorre ridurre il numero di sinusoidi.

- 1: inserimento dell'header di un cluster, ovvero del numero del cluster seguito dai 4 campi della struttura. In questo caso l'header deve essere seguito da un numero di comandi di aggiornamento del singolo oscillatore (comando 4 oppure 6) pari al numero di sinusoidi del cluster;

- 2: cancellazione di un cluster. Deve essere immediatamente seguito dal numero del cluster stesso

- 4: aggiornamento dei parametri di un oscillatore (esclusa la fase). Questo comando (come il 6) deve essere seguito dall'elenco di tutti i valori che caratterizzano il singolo filtro IIR.

- 6: aggiornamento dei parametri di un oscillatore (compresa la fase)

- 7: aggiornamento dell'header di un cluster. Tale comando necessita del numero del cluster più i 4 nuovi valori dell'header.

- 5000000: fine frame se non è seguito da altri 5000000, altrimenti i successivi indicano che occorre riutilizzare per il frame successivo gli stessi valori di quello precedente.

Nei comandi 4 e 6, l'header è seguito dal gruppo di strutture dati che contengono i parametri necessari agli oscillatori, oltre a ampiezza, frequenza e fase, si trovano, dei primi due, le variazioni tra il valore del frame attuale e di quello successivo, il coefficiente cexp per le approssimazioni esponenziali, NLA e NLF, (questi dati)

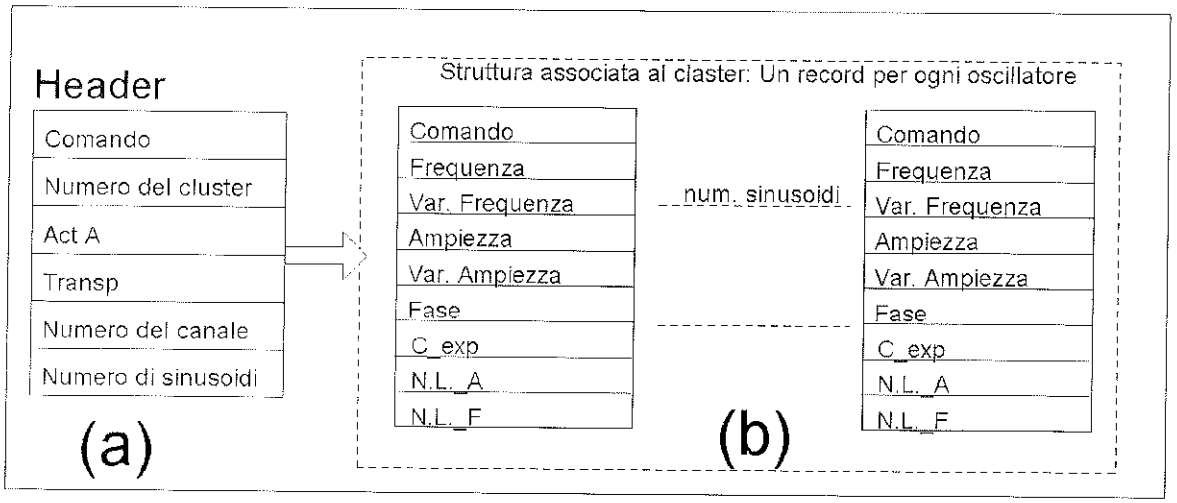

Fig. 4.3

4.3 Architettura del chip di sintesi

In questo paragrafo si fornirà una descrizione sommaria dell'architettura riassunta nello schema a blocchi di Figura (4.6). Per maggiori approfondimenti si rimanda alla relativa bibliografia [BRST&B97]. Il blocco "AddSynth" mostrato in dettaglio nella Figura (4.4) è il chip di sintesi implementato con gli oscillatori digitali basati sui filtri IIR (Par. 3.2).

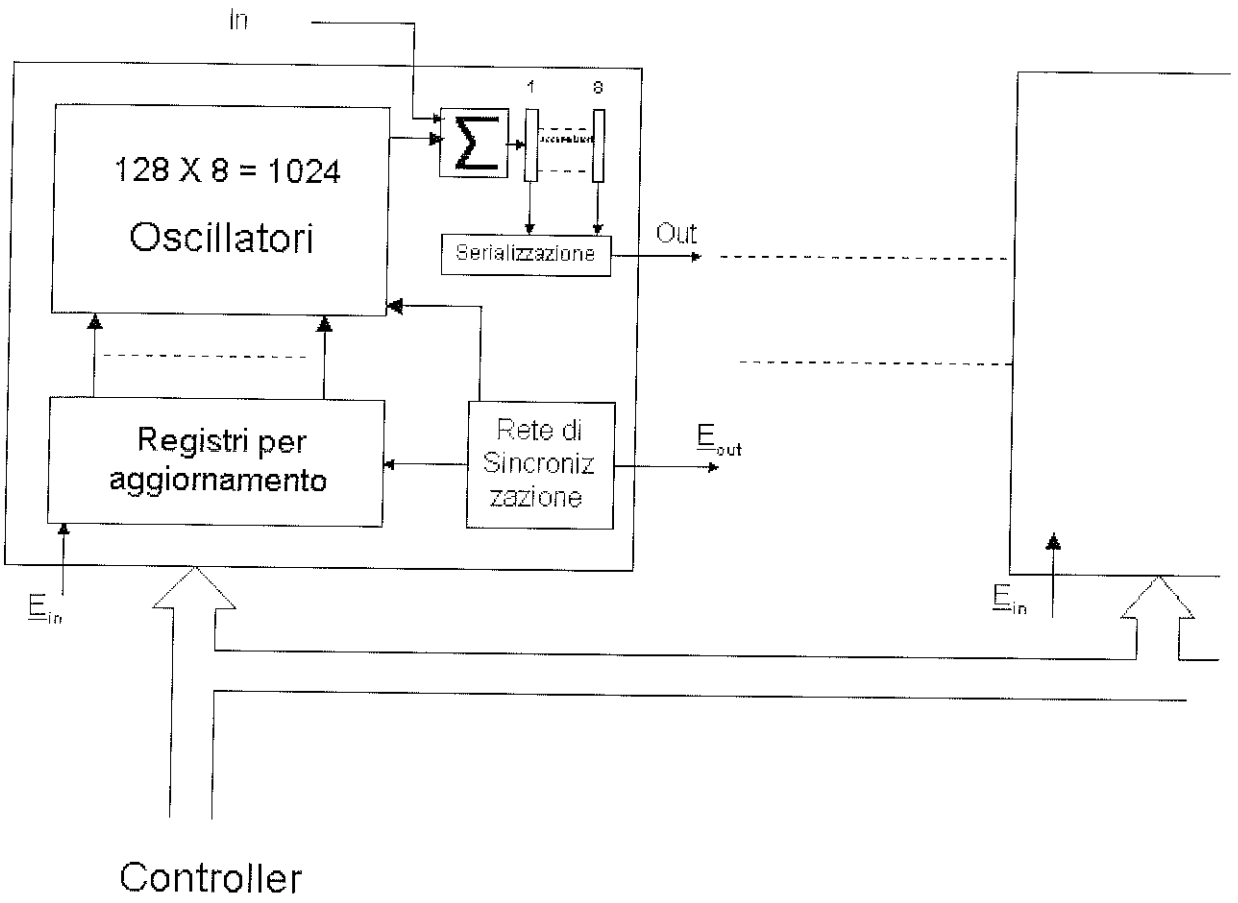

Figura 4.4. Schema a blocchi del chip di sintesi

Uno schema del funzionamento è riportato qui di seguito:

Prima Fase

- il controller ad ogni intervallo di tempo T a manda un E in al primo chip;
- il controller comincia ad inviare i dati a "raffica" sul bus;
- il primo chip riceve e memorizza nel Registro di Aggiornamento i dati necessari;
- quando l'aggiornamento è stato completato il primo chip abilita il successivo alla ricezione mediante il segnale di E_{out} ;
- questo procedimento è eseguito da tutti i chip fino a quando l'ultimo manda il segnale di E_{out} al controller;

Seconda fase

- gli oscillatori elaborano i campioni su dati coerenti;

- i risultati vengono distribuiti sugli 8 accumulatori e successivamente serializzati e mandati al chip successivo per essere sommati;
- l'ultimo chip invierà i risultati della somma al blocco di Output. L'intervallo regolare T a è stato scelto in modo da generare suoni che risultino all'orecchio assolutamente privi di "granulosità".

Da studi specifici in questo campo e da prove effettuate su varie tipologie di suoni è stato evidenziato che l'orecchio non è in grado di distinguere un suono (a frequenza variabile nel tempo) generato in maniera esatta (con modulazione continua) da uno generato con dei frames di durata 2 $T_a = 128T_c$ ovvero $T_a = 2{:}667$ms a 48KHz o $T_a = 2;9$ms a 44,1KHz.

In figura (3.6) si può notare i parametri che alimentano l'oscillatore Amp, ΔA e k (legato alla frequenza dalla relazione $k = 2\cos(2\pi T_c \cdot f)$), assieme ai registri X_0 e X_1, contenenti variabili necessarie all'algoritmo di sintesi, e che vengono aggiornati ad ogni frame per mantenere la fase consistente con i frames precedenti.

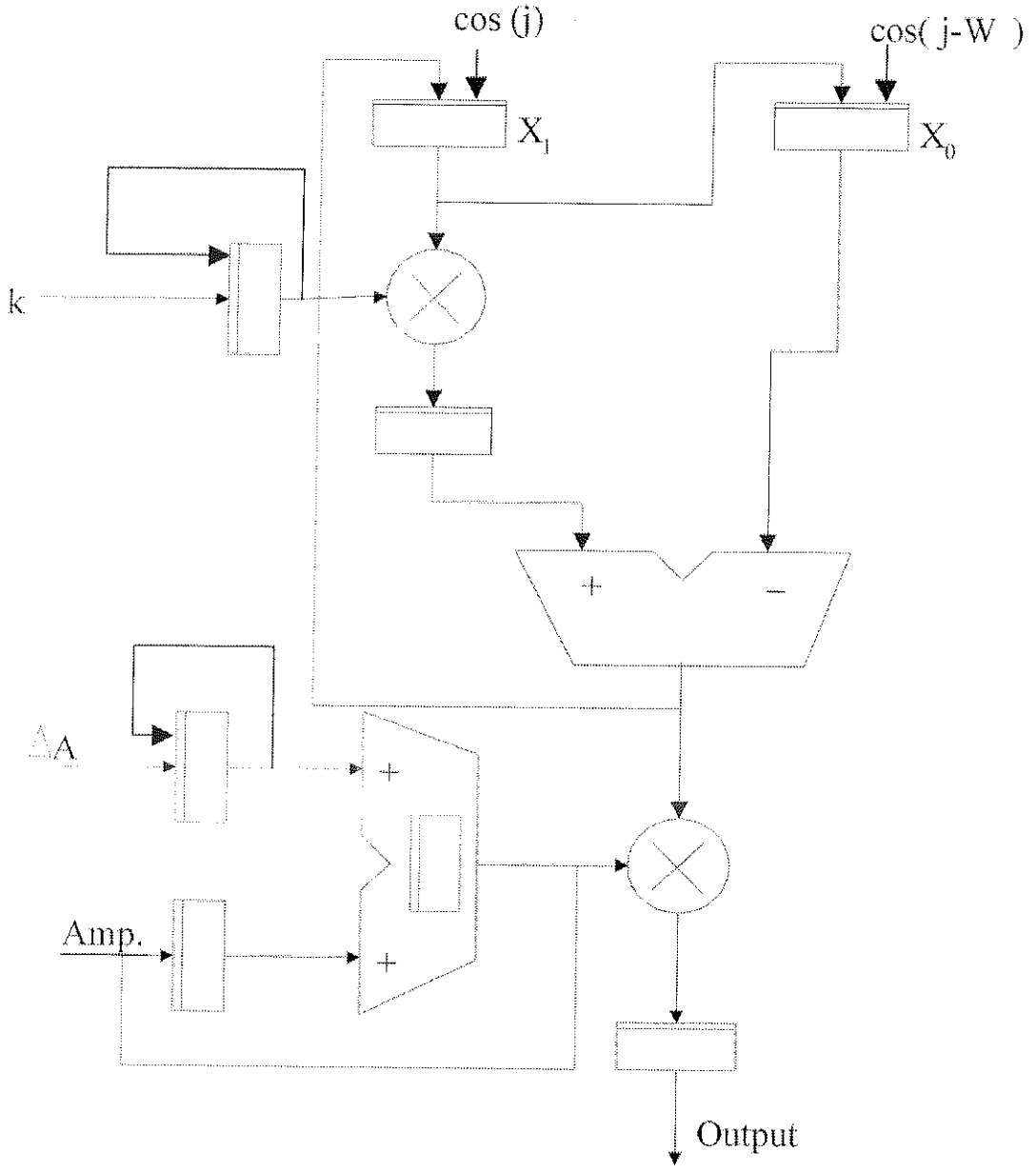

Figura 4.5. Schema di principio dell'oscillatore IIR e dell'interpolatore d'ampiezza

Nel chip sono presenti circa 1200 oscillatori di questo tipo. L'AddSynth contiene inoltre altre unità di controllo con i seguenti compiti:

- Acquisire i parametri dal controllore
- Generare tutti i segnali di controllo interni al chip
- Gestire l'interpolazione di ampiezza intra-frame
- Sommare i parziali dei campioni generati dagli oscillatori.
- Suddividere l'uscita tra gli otto canali

La sezione centrale di Figura (3.2) corrisponde alla parte di controllo del sistema che è la più complessa del simulatore. Il controllore ha il compito di leggere i comandi inviati dall'host sull'interfaccia di ingresso (HOST-Ctrl I/O), tradurre il contenuto dei cluster, elaborare e inviare i parametri accettati dal chip. L'architettura prevede anche una memoria SRAM dove vengono immagazzinati i parametri degli oscillatori attivi. Tali parametri rimangono in memoria finché non vengono aggiornati o cancellati. Nella struttura (3.2) in basso si può anche notare una memoria ROM che contiene la look-up table del coseno per fornire le condizioni iniziali agli oscillatori in base ai dati letti dalla SRAM. Infatti, la differenza sostanziale tra l'oscillatore tradizionale e quello con filtri IIR è la collocazione della LUT, mentre nel primo la tabella è implementata all'interno di ciascun oscillatore, nel secondo caso è inserita una sola volta all'interno del controllore, con evidente risparmio di spazio ed aumento di efficienza. Inoltre, in questo sistema si ha un accesso alla tabella ogni 128 campioni generati (per ogni oscillatore attivo) e non per ogni campione come nella sintesi tramite LUT. Altri compiti del controllore sono:

- Allocare i nuovi cluster;
- Spostare interi gruppi di cluster per eliminare i vuoti dovuti alla cancellazione di uno di di essi non più attivo (deframmentazione);
- Gestire i segnali di controllo e di abilitazione;
- Trasferire in uscita i parametri delle sinusoidi, fornendo il formato sistolico richiesto dall' AddSynth.

3.4 Il Simulatore

Prima di sviluppare la parte hardware del progetto controllore-oscillatori, si è resa necessaria la realizzazione di un simulatore software che ricalcasse il più possibile lo schema di principio del progetto hardware. In questo modo è stato creato uno strumento utile per analizzare le variazioni delle sinusoidi in uscita dai filtri e dai canali al variare dei parametri di ingresso di segnali campioni forniti ai singoli filtri IIR. Ciò è indispensabile per verificare la bontà del progetto e in caso negativo di apportare le opportune modifiche all'hardware. Per la stesura dei listati del simulatore è stato utilizzato il linguaggio C++ nell'ambiente di sviluppo GNU. Rispettando un approccio gerarchico, il simulatore è stato suddiviso in moduli, ognuno dei quali comprende un file header, con le dichiarazioni delle strutture dati, e il sorgente vero e proprio. Il simulatore è perciò costituito dai seguenti file:

- Define:h:

definizione di tutti i parametri numerici utilizzati;

- Bit true:h e bit true:cc:

definzione delle strutture dati e delle operazioni su di esse definite;

- AddSyn:h ed addSyn:cc:
simulatore delle azioni del chip di sintesi e dell'accumulatore finale;

- Ctrl:h e ctrl:cc:
simulazione delle azioni del controllore;

- Oscill:cc: lancio della simulazione effettiva.

5 Il problema della riduzione dei dati

Nel campo musicale, un modello di analisi/sintesi di un suono reale può essere considerato "buono" se usando i dati risultanti dall'analisi, troviamo che un suono risintetizzato è percettivamete indistinguibile dal suono reale. Tuttavia, definire completamente un suono, in modo da riprodurlo, conduce ad una rappresentazione quadridimensionale, con ampiezza, frequenza, tempo e fase come parametri fisici. In generale la fase non è un parametro significativo perché in molti casi esso può essere perturbata (per esempio mediante una stanza riverberante) senza apprezzabile perdita di informazione [Cha81]. La sintesi additiva, benché capace di produrre eccellenti risultati, è stata spesso considerata impraticabile dai musicisti e costruttori di strumenti elettronici, perché la quantità di dati da gestire è veramente grande anche per un semplice spartito a bassa polifonia. Utilizzando solo tre dimensioni fisiche, la quantità di dati necessaria per un tono individuale resta ancora considerevole poiché consiste di due funzioni tempo-varianti (ampiezza e frequenza) per ogni parziale. Per rispettare il teorema di Nyquist (1.1), il numero di componenti frequenziali associate ad un suono periodico complesso di 1000 Hz, per esempio, è potenzialmente 20 (la banda audio per l'HI-FI è 20000 Hz), durante la porzione relativamente periodica del suono (le porzioni del suono aperiodiche, possono avere uno spettro distribuito in molto più di 20 frequenze). Per un tono di 100 Hz sono necessarie in teoria più di 200 componenti, perciò per specificare una sola nota con ampiezza e frequenza sarebbero necessari più di 400 controlli differenti. Se si considera un sistema polifonico in cui si vogliono fare suonare k voci contemporaneamente, occorre fornire al sintetizzatore ad ogni istante di campionamento, tutti i parametri necessari a generare ogni singolo campione come si può vedere in Figura (5.1).

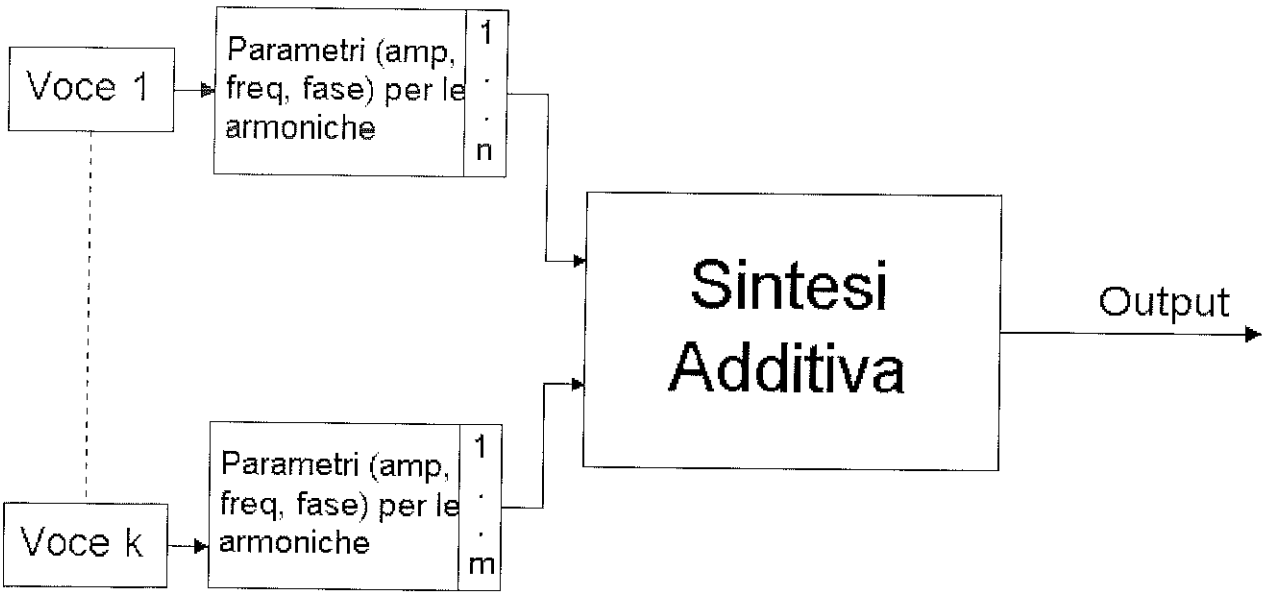

Figura 5.1. Schema di sintesi additiva polifonica

I parametri di ogni voce, visti prima, devono essere memorizzati in opportune strutture dati, da cui poterli estrarre durante il processo di sintesi. Se si pensa che questo lavoro deve essere effettuato per ogni nota di ogni "strumento" che si vuole simulare risulta evidente la necessità di riduzione dei dati cercando di non degradare la qualità sonora del risultato. Nei sistemi reali che implementano tale sintesi di cui si parlerà nel seguito, i parametri vengono aggiornati ad intervalli regolari T_a e tale intervallo deve essere scelto in maniera da generare suoni che risultino all'orecchio assolutamente privi di "granulosità". Questo però non è ancora sufficiente, in quanto il flusso di dati dall'host al sintetizzatore vero e proprio sarebbe ancora troppo elevato.
Con lo sviluppo della psicoacustica [She62] [BA73] si è stabilito che il timbro è un attributo multidimensionale del suono [Gre77]. Ciò potrebbe giustificare a priori la necessità per una grande quantità di dati per definire questo attributo. Dall'analisi del timbro appena citato e da numerose prove effettuate su suoni reali, si evince che esso è connesso a pochi fattori psicologici, perciò i dettagli delle funzioni tempo-varianti (ampiezza e frequenza) probabilmente non sono significativi per la percezione del timbro. La riduzione dei dati è non solo di interesse per la sintesi, la trasformazione, o la trasmissione del suono, essa permette anche una comprensione più profonda delle caratteristiche veramente rilevanti dell'udito (in particolare gli elementi essenziali della percezione del suono).

5.1 Riduzione dei dati per l'ampiezza

Le ricerche effettuate sulla composizione di un tono (citate prima) indicano chiaramente che il fattore più significativo per identificare il timbro è legato strettamente alla composizione spettrale del suono. Questa composizione può essere valutata in molti modi, includendo la larghezza di banda, l'equilibrio dell'energia tra frequenze basse e alte, e anche la natura armonica o quasi-armonica del segnale. E' stato dimostrato [GG78] che una delle caratteristiche più rilevanti è la forma dell'inviluppo dello spettro d'ampiezza, cioè la variazione temporale delle ampiezze delle componenti parziali (Figura (1.15)). Una tecnica sviluppata da Charbonneau [Cha81] cerca di ridurre al minimo la quantità di memoria necessaria per la memorizzazione dei parametri. Si considera l'inviluppo d'ampiezza di ogni armonica e si stabilisce di memorizzare il valore massimo di ognuna. Calcolando la media aritmetica delle ampiezze di tutte le armoniche in ogni punto nel tempo, si ottiene per ogni strumento una curva di ampiezza media, che in seguito si normalizza in maniera che abbia un'ampiezza massima uguale a 1. La descrizione delle ampiezze tempo-varianti per un tono è ridotta alle seguenti informazioni :

- Una curva di ampiezza media normalizzata a 1;
- Un insieme di tre valori TS_i , TE_i , MAX_i , (i=1,2, . . . , N) che rappresentano rispettivamente il tempo iniziale, il tempo finale, e l'ampiezza del picco di ognuna delle N armoniche.

La ricostruzione del tono semplificato consta di sintetizzare le N armoniche con la stessa variazione di ampiezza come quella calcolata per la curva media di ampiezza normalizzata. Per ogni armonica, questa curva è moltiplicata per l'ampiezza MAX_i in tal modo, il valore del picco coincide con il massimo MAX_i e nel tempo la i^{th} armonica inizia al tempo TS_i e finisce al tempo TE_i (Figura (5.2)). Nonostante i valori di ampiezza del picco siano gli stessi, il tempo al quale l'ampiezza di una armonica arriva al valore del suo picco qualche volta è sostanzialmente spostato.

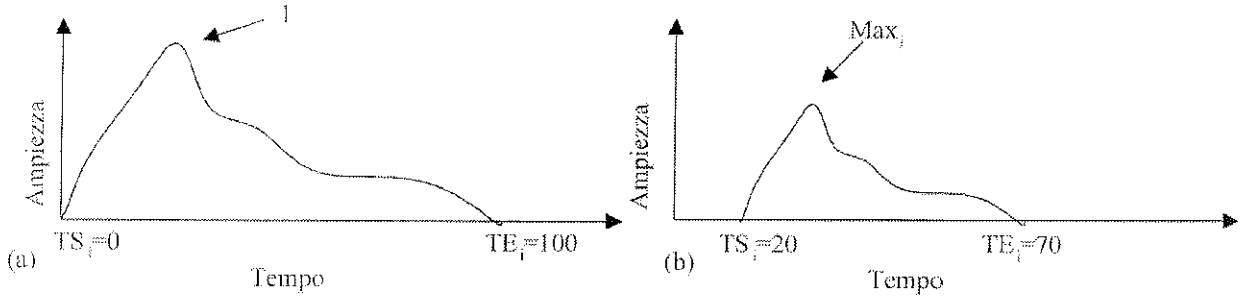

Figura 5.2. (a) Curva media normalizzata. (b) Curva della i th armonica

5.2 Riduzione dei dati per la frequenza

L'altra dimensione fondamentale del suono è la frequenza. Lo scopo in questo caso è di ottenere una singola curva di variazione dalla quale sia possibile dedurre mediante semplice trasformazione una curva di variazione di frequenza per ciascuna parziale. Dopo l'esame dell'analisi spettrografica dei toni suonati su strumenti differenti, nel lavoro di Charbonneau [Cha81] non sembrò possibile calcolare una curva media come quella calcolata per l'ampiezza. Infatti, sebbene piuttosto diverso in dettaglio, le curve di ampiezza delle varie componenti di un tono mostrano caratteristiche comuni, specialmente forme comuni: esse partono da 0, crescono fino ad un massimo e poi decrescono a 0. In questo caso, la curva media riflette la forma delle diverse curve.
D'altra parte, le curve di variazione di frequenza non presentano, anche per un singolo strumento, una forma così generale. La somiglianza nelle variazioni locali delle diverse armoniche sembra essere incontrata più frequentemente (nonostante ciò non può essere accettato come una generalizzazione).
Per questa ragione, si sceglie di prendere come curva di riferimento della frequenza la funzione corrispondente alla variazione della frequenza fondamentale. In questo tipo di semplificazione, la scelta della armonica fondamentale non è imperativo, e probabilmente è possibile ottenere risultati simili con altre armoniche. Comunque, l'armonica fondamentale presenta un importante vantaggio: in tutti i toni di riferimento ha una durata più lunga rispetto alle altre armoniche. La frequenza tempo-variante di ciascuna armonica è ottenuta moltiplicando ciascun punto della curva di riferimento della frequenza per un numero appropriato di armoniche tra il tempo iniziale e finale della armonica considerata (la frequenza è impostata a 0 altrove). Usando la fondamentale, evitiamo di estrapolare le variazioni di frequenza nell'istante in cui l'armonica non esiste. In questo tipo di semplificazione, lo spettro risultante è esattamente armonico in ogni istante, ma il periodo della forma d'onda fluttua nel tempo come quello della frequenza fondamentale nel tono di riferimento. Così l'informazione rilevante per la frequenza è ridotta per i seguenti dati:

- Una curva di riferimento della frequenza (la funzione per la fondamentale).
- Un insieme di due valori, TS_i e TE_i , che rappresentano il tempo iniziale e finale della i^{th} armonica (notare che questa informazione esiste già nella riduzione dei dati ampiezza).

Un risultato interessante che ha riscontrato l'autore di queste tecniche è che, a fronte di una forte riduzione dei dati, si ottiene un suono abbastanza riconoscibile e appartenente alla famiglia di quello di partenza. Inoltre si è osservato che, per quasi tutti gli strumenti esaminati, la riduzione sull'andamento temporale dell'ampiezza è un fattore molto più sensibile della riduzione dei dati per la frequenza.

5.3 Approssimazione lineare (PLA)

Un approccio più adeguato per la sintesi additiva è basato sull'idea che le proprietà acustiche del segnale devono essere riprodotte con un grado di approssimazione preservando proprietà oggettive del suono sorgente, come lo spettro a tempo variante. Tale tecnica ha avuto origine dalle ricerche sulla percezione del timbro [HB96]. Da tale lavoro si è dedotto che i comportamenti delle frequenze e delle ampiezze di ogni parziale di un suono musicale è meno importante del suo comportamento complessivo o medio. Portato all'estremo questo metodo è equivalente a rimpiazzare le ampiezze e le frequenze di una singola componente con il valor medio delle parziali costituenti la nota musicale. Questa è una delle tecniche più comuni di riduzione dei dati e consiste nel rimpiazzare la variazione di una parziale con un ridotto numero di valori che rappresentano i breakpoint di un'approssimazione lineare, cercando di inseguire l'andamento della curva originale con una funzione lineare a tratti (Figura 5.3).

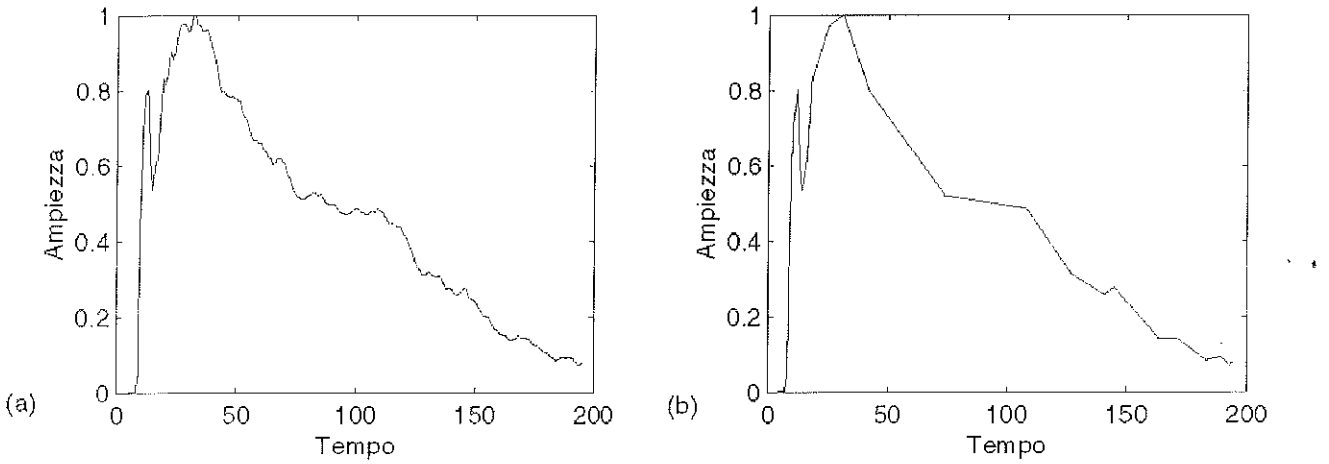

Figura 5.3. Inviluppo d'ampiezza dell'armonica fondamentale. (a) Con 200 punti, (b) Con 30 breakpoints

Formalmente il metodo cerca di trovare il migliore insieme di coordinate di ampiezza [$t_{A(k;j)}$; A(k;j)] (i breakpoint), dove A(k;j) è l'ampiezza della k-esima armonica al tempo $t_{A(k,j)}$. Si noti che il termine armoniche è usato impropriamente per riferirsi a una parziale anche se non si tratta di un multiplo intero della fondamentale. Tali coordinate vengono utilizzate per costruire inviluppi di ampiezza approssimati $a'_k(t)$ che siano una buona approssimazione dell'ampiezza originale. Non esiste una soluzione a forma chiusa per approssimare tali curve con pezzi di segmenti in quanto il problema è di natura non lineare. La maggior parte delle ricerche ha riportato inviluppi di ampiezza e frequenza appropriate per molti strumenti, ma sono state realizzate empiricamente [Moo77] e solo qualche autore ha proposto dei metodi semiautomatici per realizzare tale processo [HB96].
In questa lavoro è proposta e sperimentata una procedura automatica per l'individuazione dei breakpoint, i cui dettagli verranno esposti nel paragrafo 6.

6 Realizzazione

In questo lavoro si è cercato di affrontare uno studio su alcune categorie di segnali audio (tipicamente note di diversi strumenti musicali reali quali, il flauto, la tromba, la chitarra ecc.) con lo scopo di individuare ed estrarre il minimo insieme di dati per la codifica dei parametri significativi che servono per risintetizzare tali suoni mediante la tecnica della sintesi additiva, tutto questo è guidato da obiettivi ben delineati:

1. Ricostruzione dei segnali sonori originali (tramite Analisi STFT) e successiva verifica del risultato tramite prove di ascolto, per confermare la scelta dei parametri effettuata .
2. Riduzione dei dati, del segnale scomposto, per evitare eccessiva occupazione di memoria e ridurre il flusso dei parametri dall'host al controllore, nell'ambito AddSynth
3. Formattazione dei dati relativi a gruppi di parametri estrapolati per essere utilizzati dal simulatore software dell'AddSynth in maniera da poterne effettuare il test e la verifica delle uscite audio.
4. Fornire delle basi teoriche e alcuni spunti di implementazione utilizzabili per approfondire ed estendere tale ricerca su una gamma più completa di suoni. Il processo generale di Analisi/Sintesi è schematizzato in Figura (6.1); di seguito ne verrà mostrata una implementazione.

Figura 61. Passi di analisi/sintesi

Per lo sviluppo del lavoro si è fatto uso di MatLab (Matrix Laboratory), un linguaggio ad alto livello che integra calcoli, visualizzazione e programmazione in un unico ambiente, che generalmente è usato per:

- manipolazioni matematiche;
- sviluppo di algoritmi;
- modellizzazione e simulazione;
- analisi di dati;
- grafici scientifici;

Inoltre, MatLab dispone di applicazioni specifiche chiamate "toolboxes": insieme di funzioni (scritte in MatLab) che estendono l'ambiente per risolvere particolari classi di problemi

Uno di questi toolbox è il Signal Processing ToolBox, dedicato all'elaborazione di segnali numerici, di cui si è fatto uso in maniera massiccia.

6.1 Analisi STFT

Come si è descritto precedentemente per poter effettuare l'analisi STFT (Shorttime Fourier transform) di un tono, è necessario dimensionare esattamente la finestra di analisi, per evitare "artefatti" nello spettro risultante. Al fine di ottenere la dimensione più appropriata della finestra è necessario conoscere la frequenza fondamentale ($Freq_{fond}$) del segnale che si cerca di analizzare. Per le proprietà della trasformata discreta di Fourier, eseguendo una FFT su una porzione del segnale, di $Lung_{Fin}$ punti, e con frequenza di campionamento di $Freq_{camp}$, dal risultato è possibile calcolare, relativamente all'intervallo considerato, lo spettro d'ampiezza, che conterrà $Lung_{Fin}/2$ frequenze, equamente spaziate, variabili tra 0 e $Freq_{camp}/2$. L'analisi consiste nel dividere lo spettro in intervalli di frequenze Int_{freq}, che vengono detti canali:

$$Int_{freq} = \frac{Freq_{camp}}{Lung_{Fin}} \qquad (6.1)$$

quindi tanto più è grande $Lung_{Fin}$, tanto più definita è l'analisi in termini di risoluzione in frequenza; la lunghezza ottimale viene stabilita in base alle caratteristiche del segnale.

Esempio 4.1 *Si considera un segnale audio con frequenza fondamentale uguale a 700 Hz e con* $Freq_{camp}$ *= 44100 Hz. Prendendo una finestra di analisi di 200 campioni si avrà che* Int_{freq}*=44100/200 = 220,5 Hz. L'analisi verrà fatta, quindi, dividendo l'intero spettro in canali di 220,5 Hz.*

primo canale	*0 – 220.5 Hz*
secondo canale	*220.5 – 441 Hz*
terzo canale	*441 – 661.5 Hz*
quarto canale	*661.5 – 882 HZ*

In questo caso la frequenza fondamentale sarà analizzata nel quarto canale (Figura (6.2) (b)), la seconda armonica nell'ottavo e così via.

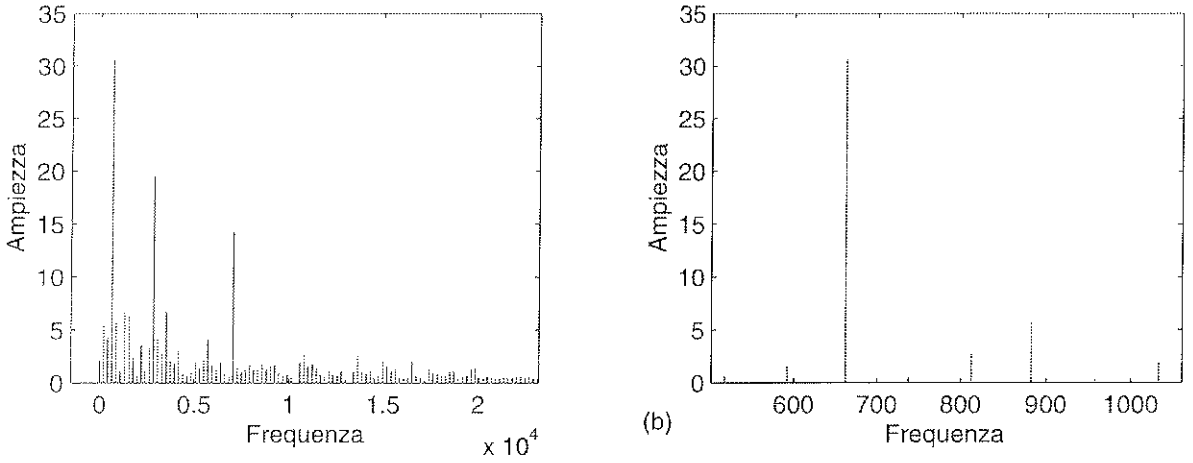

Figura 6.2. (a) Spettro d'ampiezza su 200 campioni di un segnale, (b) Zoom del quarto canale di frequenza di (a)

Fino a quando in ogni canale viene analizzata una sola armonica l'analisi procede correttamente. Se si hanno più frequenze che cadono nello stesso intervallo, dato che nell'analisi viene fatta una media di ciò che esiste in ogni canale, si otterrà una scarsa definizione dal punto di vista frequenziale. Per le considerazioni fatte precedentemente, si è scelto di porre $Int_{freq} = Freq_{fond}$ ottenendo dalla formula (6.1) la lunghezza della minima finestra di analisi:

$$Lung_{Fin} = Freq_{camp} / Freq_{fond} \qquad (6.2)$$

L'espressione (6.2) rappresenta la minima finestra su cui si può applicare la STFT, infatti considerando finestre più piccole si ha una perdita del contenuto armonico, quindi di informazione, mentre utilizzando intervalli molto grandi si ha una perdita dell'esatto andamento temporale degli spettri. Tutto questo comporta la scelta di un compromesso che è quello di utilizzare, comunque, sequenze di campioni multiple di $Lung_{Fin}$, e operare un overlapping, in modo che ogni finestra non abbia in comune con la successiva i primi 128 campioni . Questo per garantire una sintesi del segnale sonoro privo di artefatti e soddisfare le specifiche dell'AddSynth

Nel caso più generale che non si conosca a priori la frequenza fondamentale del suono da analizzare, bisogna applicare una FFT su un intervallo temporale abbastanza grande, tipicamente 44100 campioni, in questo modo si ottiene una risoluzione in frequenza di 1 Hz. Le simulazioni sono state effettuate su suoni campionati a 44100Hz e finestrati opportunamente con la finestra di Hamming della stessa lunghezza. Dopo aver effettuato la FFT sulla sequenza risultante, è possibile estrarre l'armonica fondamentale con buona approssimazione.

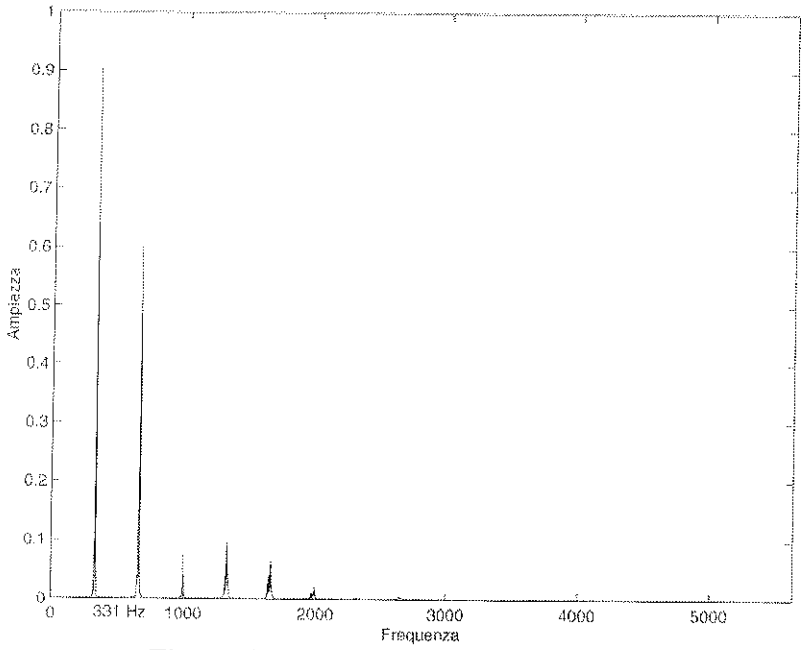

Figura 6.3. Spettro d'ampiezza della nota precedente

La Figura (6.3) mostra il segnale nel dominio della frequenza a cui è stato applicato il processo di estrazione dell'armonica fondamentale.

6.2 Riduzione dei dati e selezione dei breakpoint

Dopo la fase di analisi, sorge il problema di estrarre, dalla grossa quantità di dati trovati un sottoinsieme dei parametri caratteristici dell'evento sonoro (ampiezza, frequenza), dai quali si possa ricostruire un suono "percettivamente" simile a quello originale una volta passati al chip di sintesi. Il metodo di riduzione utilizzato è il PLA (Par. 2.5.3) in quanto permette di ridurre il numero di breakpoint di circa il 90% rispetto al numero dei frames nella STFT di partenza e soprattutto perché diminuisce notevolmente il flusso dei dati dall'host al controllore. Infatti i dati verranno inviati ad ogni cambio di pendenza degli inviluppi d'ampiezza e il controllore si occuperà di gestire tali parametri fino al successivo aggiornamento. La procedura trova dei breakpoint comuni per tutte le armoniche, cioè ritorna come risultato un vettore di lunghezza L break , contenente la posizione di ciascun breakpoint e una matrice $N_{Arm} \times L_{break}$ in cui ogni riga rappresenta un'armonica e contiene il valore dell'ampiezza di ciascun breakpoint. La scelta di breakpoint comuni può sembrare troppo restrittiva, invece è motivata da:

- risparmio di memoria, infatti basterà memorizzare un numero n di breakpoint invece di n × Numero delle Armoniche;
- velocità di esecuzione, il metodo esposto è considerevolmente più veloce quando è ristretto a cercare breakpoint comuni;
- sintesi accettabile, tempi comuni danno risultati ugualmente buoni, se non migliori, rispetto a quelli indipendenti, considerando equivalenti requisiti di memoria.

Per scegliere l'insieme dei breakpoint comuni, che approssimano meglio gli inviluppi originali, l'algoritmo si basa su una funzione errore così definita ([HB96]):

$$\text{errore relativo} = \frac{1}{N_{frames}} \times \sum_{n=1}^{Nframes} \sqrt{\frac{\sum_{k=1}^{N_{har}} (a_k(n) - a'_k(n))^2}{\sum_{k=1}^{N_{har}} (a_k(n))^2}} \qquad (6.3)$$

dove:
$a'_k(n)$ è l'ampiez<za approssimata della k-esima armonica al tempo n,
$a_k(n)$ è l'ampiezza originale della k-esima armonica al tempo n,
N_{har} è il numero di armoniche usate,
N_{frames} è il numero di finestre usate per l'analisi.
Questa misura dell'errore relativo dà un peso ad ogni frame durante il tono. Una misura dell'errore assoluto come:

$$\text{ErroreAssoluto} = \frac{1}{N_{frames}} \times \sum_{n=1}^{Nframes} \sqrt{\sum_{k=1}^{N_{har}} (a_k(n) - a'_k(n))^2} \qquad (6.4)$$

potrebbe dare eccessivo peso alle porzioni del tono con altezza più alta, che è indesiderabile, in quanto i frames di attacco e decadimento, a bassa ampiezza, sono importanti quanto i relativi stati di sostegno. L'errore relativo è più adatto a rappresentare il grado di sensibilità della percezione umana. La definizione (6.3) non dà abbastanza peso all'attacco che, in generale, è solo una piccola frazione dell'intera durata del suono. E' possibile allora modificare l'equazione precedente in modo che l'attacco riceva il peso desiderato:

(6.5) **ErroreAtt**

$$\frac{W_1}{FrameAttac-1} \times \sum_{n=1}^{FrameAttac-1} \sqrt{\frac{\sum_{k=1}^{N_{har}} (a_k(n) - a'_k(n))^2}{\sum_{k=1}^{N_{har}} (a_k(n))^2}} + \frac{W_2}{N_{frame} - FrameAttac+1} \times \sum_{n=FrameAttac}^{N_{frame}} \sqrt{\frac{\sum_{k=1}^{N_{har}} (a_k(n) - a'_k(n))^2}{\sum_{k=1}^{N_{har}} (a_k(n))^2}}$$

dove:
W_1 e W_2 sono rispettivamente il peso che si vuole dare all'attacco e al resto del tono, (ad esempio si consiglia di utilizzare $W_1 = W_2 = 0{:}5$ per dare lo stesso peso alle due porzioni del suono);
FrameAttac è il numero di frames per l'attacco;
E' necessario chiarire che la misura dell'errore è la chiave del successo di un algoritmo di approssimazione; come determinare la migliore formulazione per la misura dell'errore percettivo tra due spettri tempo-variante, simili è una questione ancora aperta. La funzione appena enunciata ha lo svantaggio che bisogna conoscere a priori l'intervallo temporale della fase di attacco, FrameAttac. In questo lavoro si è utilizzata la funzione (6.3) in quanto fornisce un buon compromesso tra semplicità d'implementazione, velocità di esecuzione e bontà dei risultati. La parte del codice che si occupa della ricerca dei breakpoints (Appendice C) può essere così schematizzata:

```
N break := :::;
σ := :::;
num := soglia := 0;
Approx;
Error;
while (num ! N break ) and (soglia ? oe)
Maxvettoreerrore;
Approx;
Error;
num := num + 1;
```

soglia := error;
end

dove "Approx" è una procedura che partendo da una fase iniziale, in cui si fissano i breakpoints iniziale e finale, genera l'inviluppo approssimato (con i breakpoint trovati fino a quel momento) con cui la funzione "Error" calcola un vettore di errori relativi (uno per ogni finestra o frame, n) e ritorna ad ogni passo l'errore totale (attuale) e il vettore degli errori relativi per ogni frame calcolato con la formula:

$$\text{errorel(n)} = \sum_{n=1}^{Nframes} \sqrt{\frac{\sum_{k=1}^{N_{bar}} (a_k(n) - a'_k(n))^2}{\sum_{k=1}^{N_{bar}} (a_k(n))^2}} \qquad (6.6)$$

"Max vettore errore" scleziona, mediante il vettore degli errori relativi, il massimo valore ponendolo come breakpoint per l'iterazione successiva.

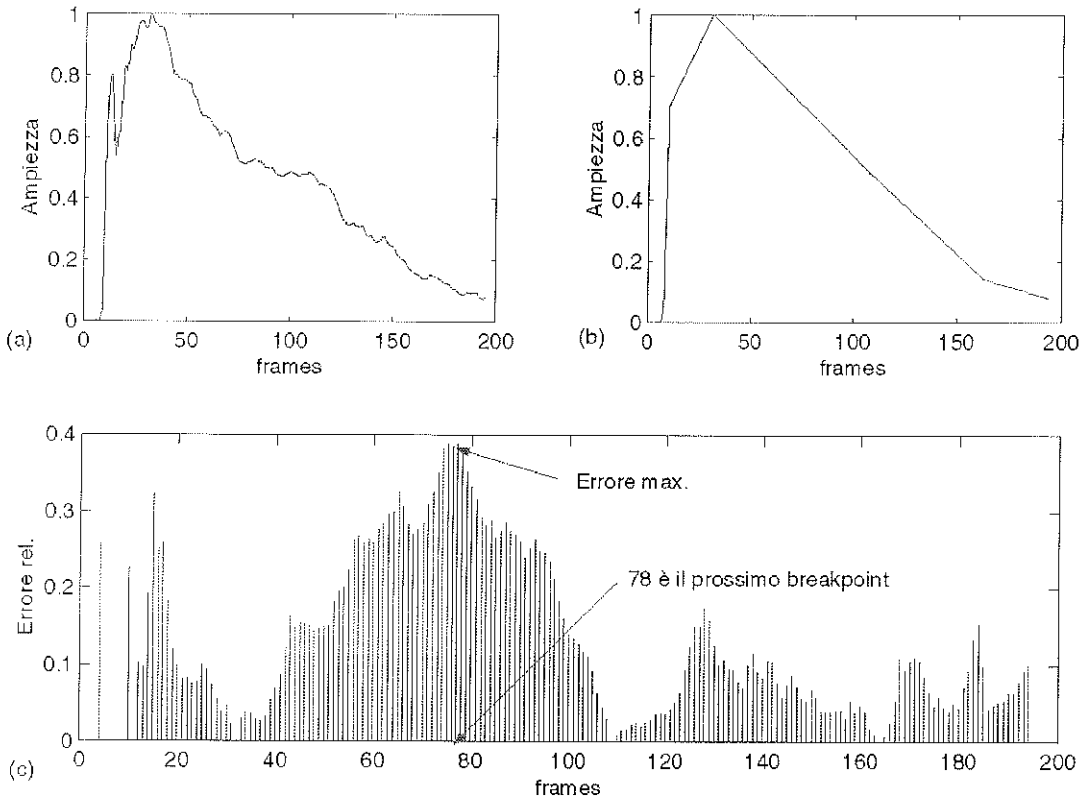

Figura 6.4.

Nella Figura (6.4) si può notare in (a) l'inviluppo originale della frequenza fondamentale di una nota; in (b) l'inviluppo approssimato dopo 13 iterazioni; in (c) il grafico associato al vettore degli errori relativi calcolato con la formula (6.6) in cui viene anche evidenziato il prossimo breakpoint.

Come si può vedere dal frammento di codice i due parametri fondamentali sono: numero di breakpoint, N break e soglia d'errore, σ. Da numerose prove effettuate si è stabilito che per il numero di breakpoint può essere scelto il 10 percento del numero delle finestre. Scegliendo una soglia di errore troppo piccola si può incorrere nell'esecuzione di tutti gli N break passi dell'algoritmo descritto senza raggiungere il σ desiderato, viceversa una soglia d'errore elevata comporta una approssimazione troppo grossolana, che in fase di sintesi produce effetti indesiderati. Dalle prove effettuate si è stabilito che per $0{,}01 < \sigma < 0{:}08$ si ottengono dei risultati soddisfacenti in fase di resintesi. Si noti che un valore della soglia uguale a 0 significa coincidenza degli inviluppi approssimati con quelli reali, mentre un valore di 0; 1 indica uno scostamento medio del 10%. In appendice è riportato un esempio completo della tecnica dell'approssimazione lineare a tratti usata in questo lavoro.

6.3 Strutturazione del file di input

La successiva fase è quella di costruire il file di input per il chip di sintesi. Come si è visto precedentemente i parametri d'ingresso dell'AddSynth devono essere opportunamente strutturati in cluster. Per questo motivo è stata creata una procedura automatica gen_clast che presi in input i dati risultanti dalla ricerca dei breakpoint li trasforma in quelli accettati dal controller del chip. La trasformazione consiste in:

- inserire nel file di input al chip una intestazione che prevede alcuni campi quali il numero d'ordine del cluster, il volume, il numero di oscillatori che devono operare, ecc.;
- trasformare alcune grandezze a seconda del numero di bit con cui vengono rappresentate all'interno del chip, cioè per ogni parame- tro esiste una funzione che "mappa" il suo valore reale in quello accettato dall' AddSynth:

✓ la frequenza (20 bit) F ([0, 22050]) $\rightarrow$ [0; $2^{20}-1$];
✓ l'ampiezza (16 bit) F ([0, 1]) $\rightarrow$ [0; $2^{16}-1$];
✓ variazione di ampiezza (16 bit) F ([0, 1]) $\rightarrow$ [0; $2^{16}-1$];

- creare un nuovo cluster quando per l'esecuzione del brano musicale è necessario allocare altri oscillatori;
- disallocare alcuni oscillatori dai cluster, quando non vengono più utilizzati (ad esempio alla fine della fase di attacco);
- eliminare un cluster, quando gli oscillatori associati non sono più necessari;

I parametri di input accettati dalla procedura gen_clast sono:

- vettore breakpoint, è unico perché abbiamo considerato tempi comuni per tutti gli inviluppi,
- matrice delle ampiezze, per ogni breakpoint i e per ogni armonica j abbiamo un valore $amp_{i;j}$ normalizzato tra [0; 1],
- $Freq_{fond}$, tutte le altre vengono generate come multiple di questa. Tale limitazione è motivata per suoni che hanno un contenuto spettrale quasi armonico.

Un frammento del file di input per l'AddSynth è mostrato in Appendice D.

6.4 Simulazione e verifica dei risultati

Nelle prove effettuate sono stati considerati dei suoni campionati a 44100 Hz con varie caratteristiche. I risultati più apprezzabili sono stati ottenuti con segnali audio quasi periodici (flauto, oboe, tromba ecc) e con una fase di attacco che si evolve lentamente nel tempo; in particolare si sono incontrate delle notevoli difficoltà per alcuni tipi di strumenti con attacco

veloce come il piano e la chitarra. Infatti, le note di tali strumenti presentano nello spettro d'ampiezza delle componenti non armoniche (rumore) non trascurabili: nel piano, poiché ad ogni nota corrispondono tre corde, durante la pressione del tasto, soprattutto nella fase di attacco, si vengono a creare degli effetti difficilmente riproducibili nella resintesi. nella chitarra esistono due fattori principali di disturbo, un attacco velocissimo e l'effetto della cassa di risonanza, ignorando i quali il suono sintetizzato sembra più pulito, ma è evidente la differenza con l'originale. Difficoltà ancora maggiori riguardano i suoni inarmonici (tamburo, piatti, ecc.) poiché presentano un contenuto spettrale molto complesso e distribuito su moltissime frequenze: per poter applicare efficacemente la tecnica della sintesi additiva occorre agire per via sperimentale, som- mando molte più parziali (anche componenti non armoniche) o mettere a punto delle metodologie di analisi appropriate che esulano da questo lavoro. Le prove di ascolto sono state effettuate in collaborazione di musicisti esperti di informatica musicale. Per valutare i suoni di test è stato realizzato un esperimento così composto:

4 strumenti (flauto, oboe, tromba, chitarra);
2 note per ogni strumento
Ogni nota viene fatta ascoltare 4 volte;
7 ascoltatori

L'esperimento compara il tono reale di riferimento a un tono prodotto dal metodo precedentemente descritto. Tra le coppie di toni è stato previsto un intervallo di 5 secondi in cui ogni ascoltatore poteva giudicare i due suoni in base ad una scala di riferimento:

0 -- I toni sono identici
1 -- Forse c'è una leggera differenza
2 -- C'è una leggera differenza di qualche tipo
3 -- C'è di sicuro una differenza
4 -- C'è una considerevole differenza, ma il timbro dello strumento è ancora riconoscibile
5 -- La differenza timbrica tra le coppie dei toni è abbastanza ampia da alterare significativamente l'evidente suono strumentale.

Per ogni strumento, i dati sono stati mediati attraverso tutte le ripetizioni per ognuno degli ascoltatori. La tabella seguente mostra i risultati ottenuti.

	Nota 1							Nota 2					
	0	1	2	3	4	5		0	1	2	3	4	5
Oboe	.14	.5	.25	.08	.03	0		.11	.43	.21	.11	.14	0
Tromba	.18	.34	.29	.11	.08	0		.18	.32	.25	.11	.14	0
Chitarra	.08	.18	.34	.18	.14	.08		.11	.18	.31	.14	.18	.08
Flauto	.25	.34	.25	.08	.08	0		.29	.34	.18	.11	.08	0

Come si può notare gli strumenti meglio imitati sono stati quelli a fiato che hanno la caratteristica di avere un suono quasi armonico.

In figura 6.5 riportiamo l'andamento delle prime 40 armoniche di una note di clarino (siB –f), mentre in 6.6 quello delle stesse armoniche ridotto con la tecnica presentata. Come si nota dal numero di vertici degli inviluppi ottenuti, la riduzione dei dati è notevole. Tuttavia il segnale risintetizzato sulla base di questi inviluppi ridotti è estremamente simile all'originale anche ad un ascolto di persone esperte.

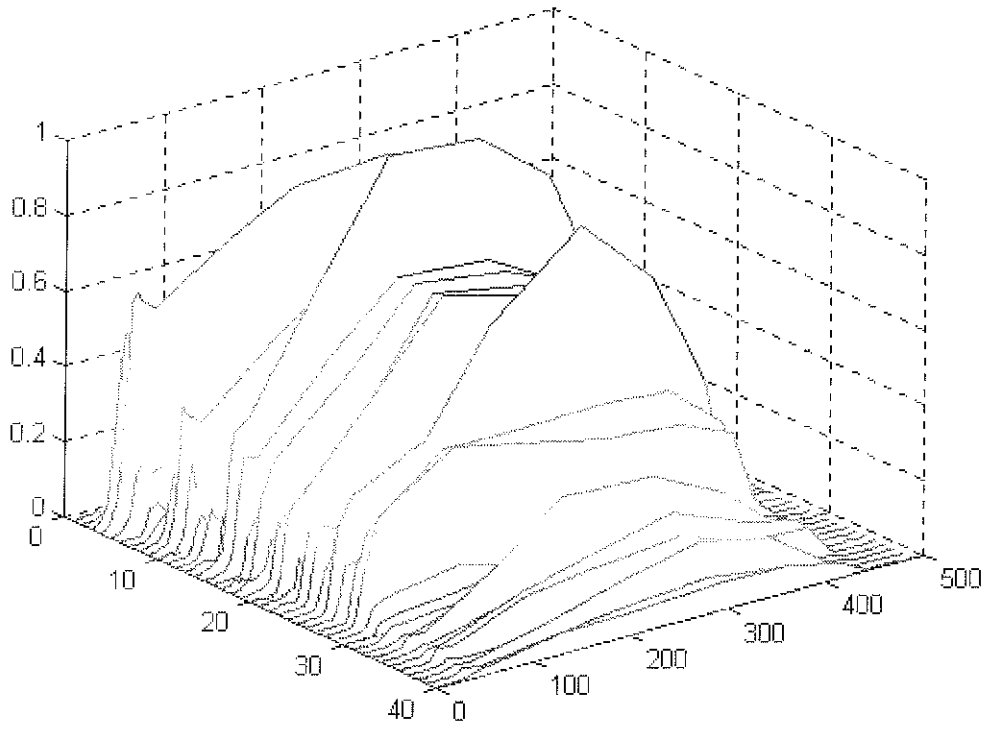
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40
0
100
200
300
400
500

Fig.6.5

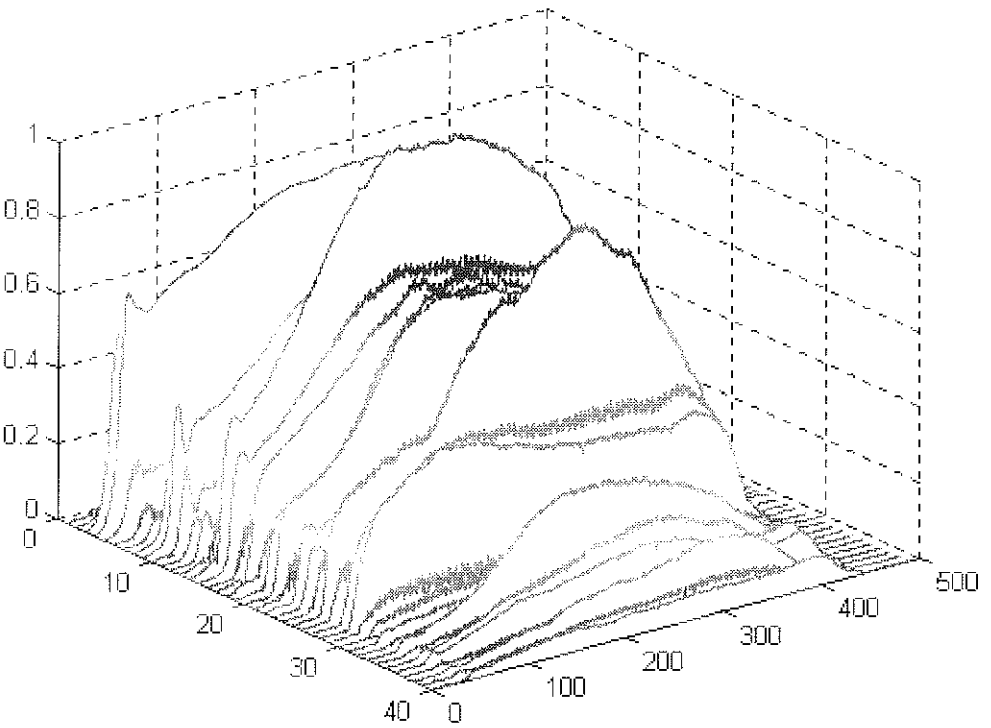
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40
0
100
200
300
400
500

Fig. 6.6

E' doveroso precisare che tali risultati non possono essere considerati di validità generale a causa del ridotto campione considerato (strumenti e ascoltatori), però induce a ben sperare per un successivo approfondimento di questo studio. Comunque il risultato è indicativo sulla relativa bontà della tecnica utilizzata.

7 Conclusioni

In questo lavoro sono state affrontate le problematiche relative all'analisi di segnali in banda audio di tipo musicale e sono state applicate delle metodologie per estrapolare e codificare i parametri caratteristici dei suoni al fine della loro utilizzazione in fase di resintesi. Le procedure sono state sviluppate in ambiente MatLab basandosi su tecniche derivate dall'analisi di Fourier, in particolare la STFT (Short Time Fourier Transform) per ricavare l'andamento degli spettri d'ampiezza delle componenti armoniche, per l'appunto, a "tempi brevi". Sono state messe a punto delle procedure per operare una riduzione dei punti di definizione delle ampiezze mediante un'approssimazione basata sulla "linearizzazione a tratti" e con la verifica dei limiti massimi di scostamento dall'andamento originario, fatta da musicisti esperti. E' stata completata la messa a punto di un simulatore per la verifica di funzionamento del Chip VLSI che implementa una tecnica di sintesi additiva su larga scala. Gli oltre mille oscillatori sinusoidali sono realizzati con filtri digitali risonanti. Inoltre è stata proposta una strutturazione dei gruppi di parametri in modo da poter essere gestiti da un'interfaccia "controller" che dovrebbe riceverli da un'unità esterna (quale può essere un PC host o una tastiera MIDI), e smistarli verso un certo numero di chip di sintesi (al massimo un decina), evitando la creazione di "colli di bottiglia". A seguito della sperimentazione effettuata utilizzando i parametri precedentemente estratti da alcuni tipi di segnale campione, sono state individuate delle correzioni da apportare ai campi di definizione dei parametri, rispetto alla versione iniziale del progetto. A tale proposito è comunque opportuno effettuare ulteriori sperimentazioni sulla sintesi di un più completa gamma di suoni, inclusi quelli con uno spettro non armonico, come i suoni di strumenti a percussione. Infine si ritiene interessante effettuare un lavoro di analisi di segnali ottenuti da uno strumento tradizionale con lo scopo di osservare l'andamento degli spettri d'ampiezza relativi a note musicali di ottave diverse e, per la stessa nota, con diverse modalità di esecuzione (piano, mezzo-piano, forte, ecc.); lo studio sullo spazio dei timbri potrebbe risultare utile per snellire la gestione di una macchina basata sulla sintesi additiva.

8 Bibliografia

[Arf79] D. Arfib. Digital Synthesis of Complex Spectra by Means of Multiplication of Nonlinear Distorted Sine Waves. Journal of the Audio Engineering Society, 27(10), 1979.

[BA73] J. P. Berzecri and Al. L'Analyse des donnees: l'analyse des correspondances. 1973.

[BB93] M. Barutti and G. Bertini. An Implementation of the Additive Synthesis Based on FFT^{-1}. Atti del Colloquio di informatica musicale, pages 127-133, 1993

[BBG80]G. Bertini, T. Bolognesi, and P. Grossi. TAU2-TAUMUS: il sistema di Computer Music in tempo reale realizzato a Pisa. Descrizione ed esperienze. Automazione e strumentazione, 28(2):134--143, 1980.

[BCD77] G. Bertini, M. Chimenti, and F. Denoth. TAU2: Un terminale audio per esperimenti di ``Computer Music''. Alta Frequenza, vol. 12, Dicembre 1977.

[BF93] G. Bertini and D. Fabbri. MultiC25 un sistema multi DSP con schede LeonardC25. Rapporto Interno del PF/CNR "Sistemi Informatici e calcolo parallelo", 1993.

[BM95] G. Bertini and M. Marani. Methodology and Digital Systems for Electroacustical Application. Proc. 2nd Int. Conf on Acoustic and Musical Research, pages 189--194, 1995.

[BRST&B97] F. De Bernardinis, R.Roncella, R. Saletti, P. Terreni. & G.Bertini "A Single Chip 1200 Sinusoids Real Time Generator for Ad- ditive Additive Synthis of Musical Signal". Int. Conf. On Acust. Speech Signal Processing IEEE, pp.:427--430, April 1997.

[BRST&B98] F. De Bernardinis, R.Roncella, R. Saletti, P. Terreni. & G.Bertini "A new VLSI Implementation of Additive Synthesis" Computer Music Journal 22:3 pp. 49-61, Mass. Inst Technology, Fall 1998.

[Bru79] M. Le Brun. "Digital Waveshaping Additive Synthesis." Journal of the Audio Engineering Society, 27(4), 1979.

[Cha81] Gerard R. Charbonneau. Timbre and the Perceptual Ef- fects of Three Types of Data Reduction. Computer Music Journal, 1981.

[Duc88] L. M. Del Duca."Musica digitale". 1988.

[GG78] J. M. Grey and J. W. Gordon. Perceptual Effects of Spectral Modifications on Musical Timbres. Journal of the Acoustical Society of America, 63:1493--1500, 1978.

[Gre77] J. M. Grey. Multidimensional Perceptual Scaling of Musical Timbre. Journal of the Acoustical Society of America, 61:1270--1277, 1977.

[HB96] A. Horner and J. Beauchamp."Piecewise-Linear Approximation of Additive Additive Synthis Envelopes: A Comparison of Various Methods. Compuer Music Journal, 1996.

[HFT95] A. D. Houghton, A. J. Fisher, and F. Thierry."An ASIC for Digital Additive Sine-wav AddSynthesys". Computer Music Journal, 19:3:23:31, 1995.

[Jaf95] David A. Jaffe. Ten Criteria for Evaluating Additive Synthesis Techniques. Computer Music Journal, 19(1):76--87, 1995.

[Jan91] C. Jansen. Sine Circuitu, 10000 High Quality Sine Waves Without Detours. Proceedings International Computer Music Conference, 1991.

[Moo77] J. A. Moorer. Signal Processing Aspects of Computer Music -- A Survey. Computer Music Journal, 1977.

[Pol83] De Poli. A Tutorial on Digital Sound Additive Synthis Tecniques. Computer Music Journal, 7(4):453--456, 1983.

[RD92] X. Rodet and P. Depalle. A New Additive Additive Synthis Method Using Inverse Fourier Trasform and Special Envelope. Proc. ICMC, pages 410,411, 1992.

[She62] R.N. Shepard. The Analisis of Proximities: Multidimensional Scaling with an Unknown Distance Function. Psychometrika, 27:125--140, 1962.

[TB92] L. Tarabella and G. Bertini."Informatica e Musica". Milano, 1992.

[BB93] M.Barutti, G.Bertini, Una Nuova Tecnica di sintesi Additiva basata sulla Trasformata Inversa di Fourier, pp.127-133, Atti del Colloquio di Informatica Musicale, 1993