La Riverberazione Digitale

Luca Spanedda

June 19, 2021

Abstract

La riverberazione digitale è un argomento sempre attuale e da sempre discusso nell'ambito della computer music e della musica elettroacustica in generale. Le sue applicazioni e studi coinvolgono sia l'ambito di ricerca commerciale che quello accademico; delineandosi dunque una storia fatta di svariate ramificazioni e risvolti che hanno portato nel tempo ad una proliferazione di diversi metodi e topologie di implementazione. In questo studio andremo ad approfondire l'argomento passando per una disamina di quelle che sono state le principali implementazioni; in un'ultima parte procederemo ad un'illustrazione di questi vari metodi implementati nel linguaggio di programmazione FAUST (Grame), e ad un'applicazione pratica di un algoritmo di riverberazione.

1 Il Riverbero

Il riverbero è la persistenza di un suono dopo che esso è stato prodotto. Si parla di un fenomeno acustico legato alla riflessione dell'onda sonora da parte di un ostacolo posto davanti alla sorgente del suono. L'orecchio umano infatti non riesce a distinguere due fenomeni sonori timbricamente simili se essi sono percepiti a meno di 100 millisecondi di distanza uno dall'altro. E in presenza di svariati elementi che danno origine ad un fenomeno di riflessione dell'onda acustica, il tempo che essa impiega a decadere mentre si riflette viene percepito come fenomeno di riverberazione. La velocità del suono nell'aria a 20 °C è di circa 340 m/s. Se presupponiamo che la fonte sonora e l'ascoltatore si trovano nello stesso punto di fronte all'ostacolo, in uno spazio aperto si può parlare di riverbero quando l'ostacolo si trova a meno di 17 metri dalla fonte

del suono. Infatti, fino a questa distanza, il percorso dell'onda sonora dalla fonte all'ostacolo e ritorno è inferiore a 34 metri e quindi il suono impiega meno di 100 millisecondi per tornare al punto di partenza confondendosi nell'orecchio dell'ascoltatore con il suono originario. Se l'ostacolo si trova a più di 17 metri di distanza dalla fonte, allora il ritardo del suono riflesso rispetto al suono diretto è superiore ai 100 millisecondi e i due suoni risultano quindi distinti; in questo caso si parla di eco.

1.1 La durata di un riverbero

I fattori che influenzano la durata di un riverbero sono molteplici. Quelli più influenti sono:

La dimensione di una stanza. - stanze più grandi producono riverberi più lunghi e piccole stanze producono riverberi corti.

I materiali. - materiali duri come creta e plastici riflettono di più il suono i materiali morbidi come il legno assorbono molto di più il suono per questi motivi legati ai materiali una piccola stanza come il bagno ha tempi di riverberazione più lunghi di una grande stanza in legno.

Il miglior modo per ascoltare il riverbero di uno spazio riverberante è quello di produrre un suono impulsivo; come un battito di mani o uno schiocco di dita.

1.2 Il Riverbero nella musica

La musica ha sempre fatto largo uso di riverberi per migliaia di anni. Archeologi e Musicologi pensano che sin dall'antichità veniva utilizzato artisticamente il riverbero prodotto dalle caverne nelle cerimonie.

Molte cattedrali in Europa hanno riverberi con una durata più lunga di 10 secondi, e la musica corale di certe ere funziona particolarmente bene sfruttando il riverbero all'interno di queste cattedrali. Infatti se pensiamo ad una monodia il riverbero delle singole note si sovrappone sulle note a seguito trasformando una melodia monofonica in un suono polifonico.

1.3 Le Riflessioni

Considerando una stanza convenzionale che ha 6 superfici:

muro a destra, muro a sinistra, muro frontale, muro dietro, soffitto, pavimento.

Un suono che nel momento in cui viene prodotto rimbalza sulle superfici e viene dopo una sola riflessione ascoltato, verrà percepito con il nome di quelle che vengono chiamate riflessioni del primo ordine. Ognuna di queste produrrà altri 6 echo: 6 echo, ognuno rimbalzante sulle 6 superfici produrrà 36 echo; queste vengono chiamate

riflessioni di secondo ordine producendo nel totale 42 echo in un brevissimo periodo di tempo e così via... Di tutti questi echo non se ne percepisce nessuno singolarmente, ma piuttosto si percepisce il loro insieme e la dispersione di questi nel tempo. Il riverbero è dunque composto da migliaia di echo del suono originale che persistono e decadono nel tempo

1.4 I modelli di riverberazione artificiali

Il riverbero è artificiale quando non è presente nella stanza dove sta avvenendo la registrazione, ma viene invece aggiunto in un secondo momento. Passiamo per una disamina delle principali implementazioni di riverberazione artificiale.

IL riverbero a nastro: si utilizza un particolare registratore/riproduttore a nastro magnetico che fa scorrere a velocità costante un anello di nastro dentro una meccanica dotata di una testina di registrazione fissa e di una di riproduzione mobile. Il segnale registrato dalla prima testina viene letto dalla seconda e miscelato all'originale generando l'effetto di riflessione ritardata del suono. Il tempo di ritardo dipende dalla distanza tra le due testine e permette di generare sia l'effetto riverbero sia l'effetto eco. Questi apparecchi sono ingombranti e pesanti. Come in ogni registrazione a nastro, si ha un rumore di fondo simile a un fruscio.

Il riverbero a molla: il segnale viene fatto passare, tramite un trasduttore, attraverso una spirale metallica (la molla). All'altro capo della molla un trasduttore equivalente al primo reimmette il segnale nel circuito di amplificazione miscelandolo a quello originale. Il segnale prelevato risulta leggermente ritardato rispetto a quello applicato al primo originando nell'orecchio dell'ascoltatore l'effetto del riverbero.

Il riverbero a camera: è sulla falsariga del riverbero a molla, in una scatola isolata acusticamente dall'esterno viene inserito un tubo curvato in maniera da creare il percorso più lungo possibile. Ad un'estremità del tubo viene posto un piccolo altoparlante mentre all'altra estremità c'è un microfono. Il suono emesso dall'altoparlante impiegherà un certo tempo per percorrere tutto il tubo ed arrivare al microfono generando così il ritardo necessario.

Infine il riverbero digitale: il segnale analogico viene digitalizzato ed immagazzinato in banchi di memoria RAM all'interno di un computer o di macchine DSP dedicate (Microprocessori, Microcontrollori) e poi prelevato in un secondo momento dopo averlo processato digitalmente.

1.5 I Riverberi Digitali

Vengono prodotti quindi da un computer o da circuiti integrati DSP dedicati. Esistono sul mercato circuiti integrati che comprendono già i convertitori A/D e D/A, le memorie ed i circuiti di temporizzazione. Un segnale acustico viene trasdotto e convertito in numeri che entrano dentro memorie, Infatti i byte vengono fatti "scorrere" da un banco al successivo fino al raggiungimento dell'ultimo. Il segnale digitale prelevato dall'ultima memoria viene poi riconvertito in analogico e miscelato al segnale originale ottenendo l'effetto riverbero; più lontano è il punto di lettura rispetto al punto di scrittura e più lungo sarà il tempo dell'echo. la grandezza di queste memorie di lettura e scrittura è chiamata: linea di ritardo, ed è espressa in campioni. La grande capacità delle memorie RAM permette di raggiungere anche ritardi di parecchi secondi e quindi passare agevolmente da riverbero a eco. La strategia utilizzata in seguito è quella di retroazionare l'output della linea di ritardo sommandola all'input, creando così un circuito di feedback. Tutto questo processo viene preferito rispetto agli altri metodi di implementazione poiché i computer moderni sono molto potenti e capaci di creare implementazioni più realistiche, tuttavia non sono ancora abbastanza potenti da essere in grado di generare tutte le riflessioni ascoltate in una grande stanza, una per una. E quindi di conseguenza l'obiettivo di creare riverberi digitali è quello di implementare modelli e strategie funzionali a replicare l'impressione della riverberazione di una stanza piuttosto che la simulazione della realisticità del processo. Il processo di replica ha generato nel corso della storia dei riverberi digitali, dei veri e propri suoni tipici differenti fra loro che possono essere implementati e ad oggi anche preferiti e scelti dai musicisti per ragioni estetiche.

2 Topologie e design dei Riverberi Digitali

Procediamo ora ad una disamina dei principali modelli di riverberazione digitale basati su linee di ritardo che si sono susseguiti nel corso della storia. A partire dal modello di Schroeder considerato come il paradigma di partenza che ha posto le basi per i modelli di riverberazione venuti seguito, sia per fini commerciali che di ricerca.

2.1 Riverbero di Schroeder

Nel 1962 Manfred Schroeder propone un'applicazione efficiente di riverberazione digitale nel suo articolo "Natural Sounding Artificial Reverb". Schroeder propone l'utilizzo di filtri allpass e comb combinati fra di loro per ottenere un riverbero che non colori (grazie ai filtri allpass) e che crei una densità degli echi sufficiente a simulare la densità di un effetto di riverberazione naturale (almeno 1000 echi per secondo)

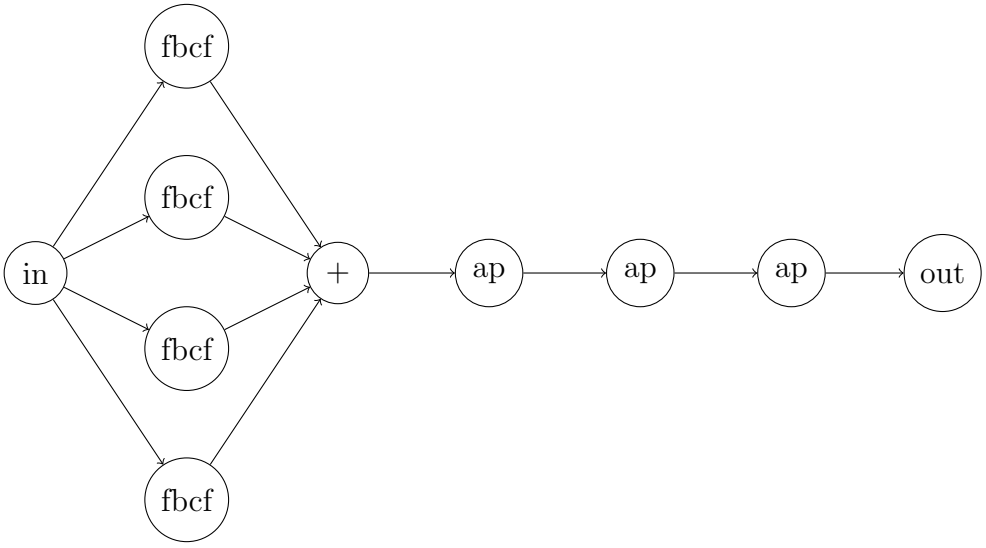

Figure 1: Topologia del riverbero di Schroeder. Con filtri AP e FBCF:

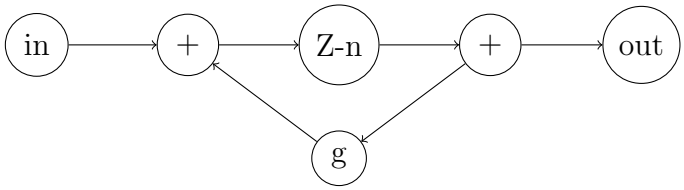

Figure 2: Filtro FBCF (Feedback Comb Filter di Schroeder)

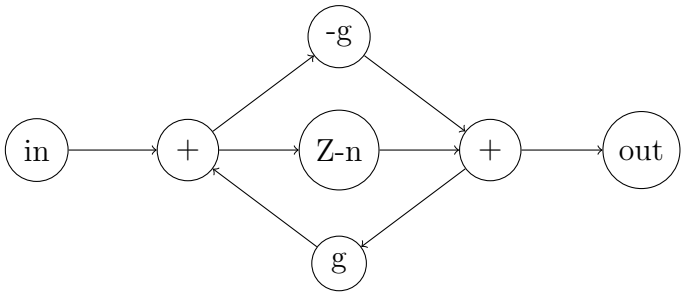

Figure 3: Filtro AP (Allpass di Schroeder)

2.2 Riverbero di Moorer

Nel 1979 nella sua pubblicazione "About This Reverberation Business" James Moorer, seguendo le proposte esposte da Schroeder nel suo articolo, implementa a seguito una topologia che fa uso delle TDL (tapped delay lines) per una simulazione delle prime riflessioni, ed inserisce all'interno della retroazione del FBCF (feedback comb filter) un filtro Lowpass, creando così i filtri LBCF (lowpass feedback comb filter) per ottenere una simulazione di assorbimento dell'aria all'interno del suo modello di riverberazione.

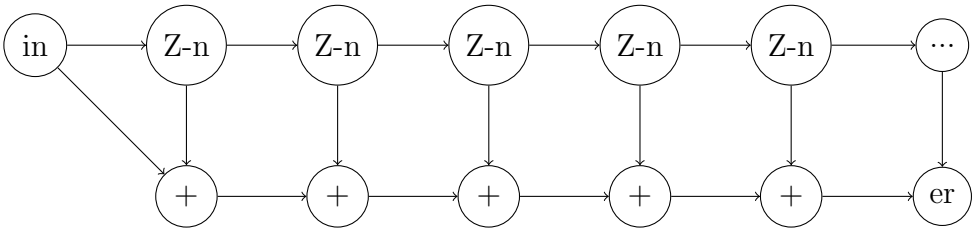

Figure 4: Topologia delle ER (mediante TDL)

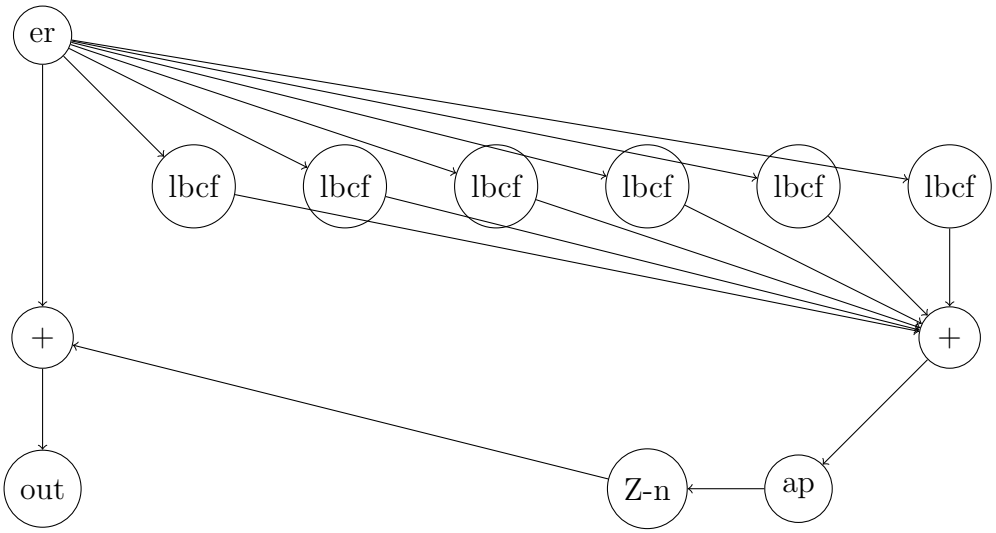

Figure 5: Topologia del riverbero di Moorer. In un implementazione chiamata Freeverb di Jezar at Dreampoint

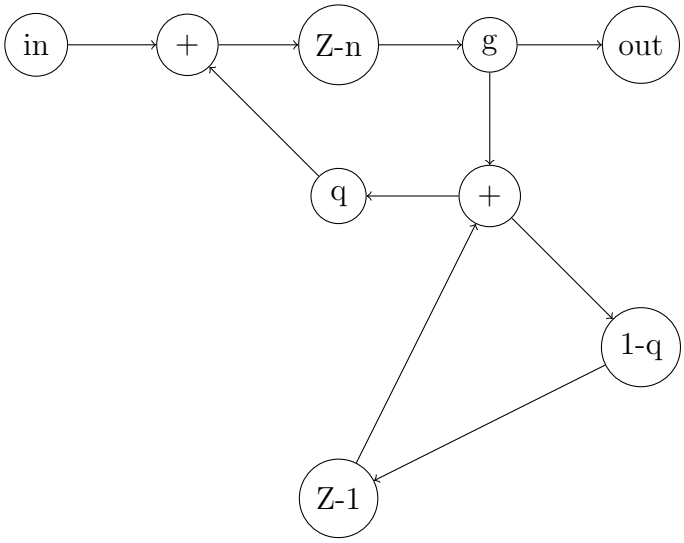

Figure 6: LBCF (lowpass feedback comb filter)

2.3 Riverbero FDN

Nel 1982 Stautner e Puckette presentano nella loro pubblicazione "Designing multichannel reverberators" un algoritmo di riverberazione multicanale chiamato: "Feedback Delay Network" che tende a voler simulare il comportamento delle riflessioni all'interno di una stanza, utilizzando solo una serie di filtri comb filter paralleli, ma con le retroazioni interconnesse fra loro.

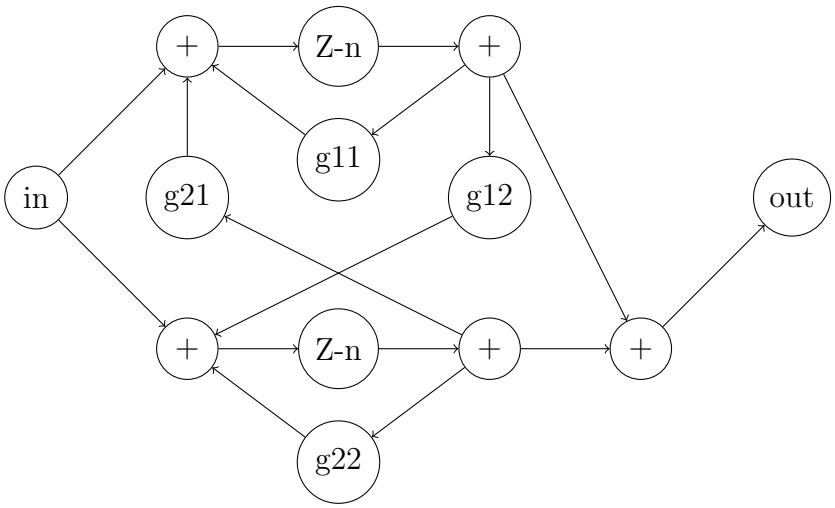

Figure 7: Topologia di una FDN a soli due Comb Filter

2.4 Altri modelli di riverberazione

Fino ad ora abbiamo esposto le principali teorie e implementazioni percorse nell'ambito della ricerca; tuttavia le derivazioni e creazioni all'infuori di queste sono le più varie. La riverberazione digitale è un prodotto di grande interesse nell'ambito commerciale e di conseguenza esistono molte altre topologie, la maggior parte di queste sono segretate, ma quando non lo sono solitamente si può notare che diversi ingegneri hanno percorso strade diverse nella creazione di una proprio metodo di implementazione. Non esiste un peggiore o migliore riverbero digitale o una peggiore o migliore topologia, ma semplicemente diversi sistemi che funzionano e rendono il proprio potenziale in diverso modo. La riverberazione digitale è un arte sperimentale e creativa.

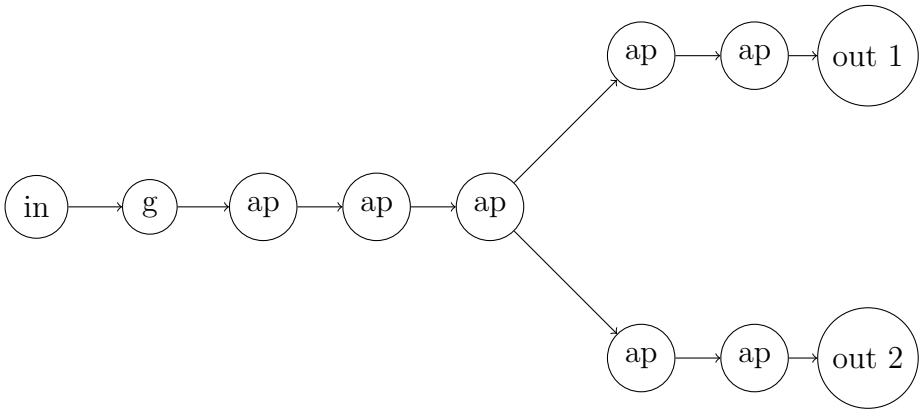

Figure 8: Riverbero di soli filtri allpass di Hal Chamberlin

3 Codici di implementazioni di Riverberazione in FAUST (Grame)

3.1 le linee di ritardo in FAUST

```
// LINEE DI RITARDO IN FAUST
/*
Le linee di ritardo in Faust
si suddividono nelle seguenti categorie:
mem - indica un solo campione di ritardo
@ - indica un numero (ex. 44100) di campioni di ritardo variabile
x'- x indica un qualsiasi ingresso e: ' un campione di ritardo,
''(2), ecc.
rdtable - indica una tabella di sola lettura
rwtable - indica una tabella di scrittura e lettura
*/


// IMPULSO DI DIRAC tramite linea di ritardo
// Importo la libreria
import("stdfaust.lib");

// tramite le linee di ritardo
// possiamo creare un impulso di Dirac, che rappresenta
// la nostra unità minima, ovvero il singolo campione
// mettendo un numero 1 e sottraendo da esso lo stesso valore
// ma facendolo ad un campione di ritardo.

// Impulso di Dirac
dirac = 1-1';
// Process
process = dirac, dirac;
```

3.2 le Early Reflections

```
// Importo libreria standard di FAUST
import("stdfaust.lib");

// ----------------------------------------
// EARLY REFLECTIONS (PRIME RIFLESSIONI)
// ----------------------------------------


/*
Simulazione delle prime riflessioni in una stanza con
il punto sorgente che coincide col punto di ascolto:
*/


// col router definisco gli input che mi devo passare dentro
// il codice
// in questo caso 7 input: (a,b,c,d,e,f,g)
in_router(a,b,c,d,e,f,g) = a, b, c, d, e, f, g;
// process = router;

// mando il segnale in ingresso ai 7 input del router
input = _ <: in_router;
//process = input;

// definisco le prime riflessioni - multitap delay lines
// (NO FEEDBACK)
multitap_delay(frnt,bck,sx,dx,up,dwn,direct) =
frnt@(4481),bck@(2713),sx@(3719),dx@(3739),up@(1877),dwn@(659),
direct;
//process = early_reflections;

// definisco un router per l'output che prenda 7 ingressi e
// li sommi
// in un unica uscita
out_router(a,b,c,d,e,f,g) = a+b+c+d+e+f+g;
//process = earlyrelfect_router;
```

```
// definisco una funzione dove esplico il percorso del segnale
early_reflections = input : multitap_delay :> out_router;

// output e test con un impulso di dirac (1 sample)
process = os.impulse : early_reflections  <: _,_;
```

3.3 Riverbero di H.Chamberlin

```
//-----------------------------------------
// CHAMBERLIN REVERB
// -----------------------------------------
// High-quality stereo reverberator:
// Musical Applications of Microprocessor
// -----------------------------------------

// Standard Library FAUST
import("stdfaust.lib");

// MS TO SAMPLES
// (t) = give time in milliseconds we want to know in samples
msasamps(t) = (ma.SR/1000.)*t;

// ALLPASS CHAMBERLIN
// (t,g) = give: delay in samples, feedback gain 0-1
apfch(t,g) = (+: _<: @(min(max(t-1,0),ma.SR)), *(-g))~ *(g)
: mem, _ : + : _;

// CHAMBERLIN REVERB
ap3ch = apfch(msasamps(49.6),0.75) :
apfch(msasamps(34.75),0.72) : apfch(msasamps(24.18),0.691);
apout1ch = apfch(msasamps(17.85),0.649)
: apfch(msasamps(10.98),0.662);
apout2ch = apfch(msasamps(18.01),0.646)
: apfch(msasamps(10.82),0.666);
process = os.impulse : ap3ch <: apout1ch, apout2ch;
```

3.4 Riverbero Schroeder in un implementazione di John Chowning - SATREV

```
// ----------------------------------------
// SCHROEDER-CHOWNING SATREV REVERBERATOR
// ----------------------------------------

// Importo libreria standard di FAUST
import("stdfaust.lib");



/*
Simulazione di Riverbero secondo il modello di John Chowning.
Modello SATREV, basato sul modello riverberante di Schroeder.
4 Comb IIR Paralleli e 3 Allpass in serie.
*/



// ------------ FILTER SECTION ------------
// FEEDBACK COMB FILTER
// (t,g) = give: delay time in samples, feedback gain 0-1
fbcf(t,g) = _ : (+  @(t-1)~ *(g)) : mem;

// ALLPASS FILTER
// (t,g) = give: delay in samples, feedback gain 0-1
apffp(t,g) = (+: _<: @(min(max(t-1,0),ma.SR)), *(-g))~ *(g)
: mem, _ : + : _;
// ----------------------------------------


// ------------ COMB SECTION --------------
    schroeder4comb = combsection
    with {
    // ROUTING PARALLELO
    in_router(a,b,c,d)= a, b, c, d;
    input = _ <: in_router;
```

```
    // 4 FILTRI COMB PARALLELI
    combs=fbcf(901,0.805),fbcf(778,0.827),fbcf(1011,0.783),
    fbcf(1123,0.764);

    // SUM SEGNALI PARALLELI
    out_router(a,b,c,d) = a+b+c+d;
    // COMBS OUT
    combsection = input : combs :> out_router;
    };
// ---------------------------------------


// ------------ ALLPASS SECTION -----------
schroederallp = apffp(125,0.7):apffp(42,0.7):apffp(12,0.7);
// ---------------------------------------


// ------------ OUT PATH ------------------
satreverb = _ : schroeder4comb : schroederallp;
process = os.impulse : satreverb <: _,_;
```

3.5 Riverbero Schroeder in un implementazione di John Chowning - JCREV

```
// ---------------------------------------
// SCHROEDER SAMSON BOX REVERBERATOR
// ---------------------------------------


// Importo libreria standard di FAUST
import("stdfaust.lib");



/*
Simulazione di Riverbero secondo il modello della
Samson Box - 1977 CCRMA.
Modello JCREV, basato sul modello riverberante di Schroeder.
3 Allpass in serie e 4 Comb IIR Paralleli.
*/



// ------------ FILTER SECTION ------------
// FEEDBACK COMB FILTER
// (t,g) = give: delay time in samples, feedback gain 0-1
fbcf(t,g) = _ : (+  @(t-1)~ *(g)) : mem;

// ALLPASS FILTER
// (t,g) = give: delay in samples, feedback gain 0-1
apffp(t,g) = (+: _<: @(min(max(t-1,0),ma.SR)), *(-g))~ *(g)
: mem, _ : + : _;
// ---------------------------------------


// ------------ ALLPASS SECTION -----------
schroederallp = apffp(1051,0.7):apffp(337,0.7):apffp(113,0.7);
// ---------------------------------------
```

```
// ------------ COMB SECTION --------------
    schroeder4comb = combsection
    with {
    // ROUTING PARALLELO
    in_router(a,b,c,d)= a, b, c, d;
    input = _ <: in_router;

    // 4 FILTRI COMB PARALLELI
    combs=fbcf(4799,0.742),fbcf(4999,0.733),fbcf(5399,0.715),
    fbcf(5801,0.697);

    // SUM SEGNALI PARALLELI
    out_router(a,b,c,d) = a+b+c+d;
    // COMBS OUT
    combsection = input : combs :> out_router;
    };
// ----------------------------------------


// ------------ OUT PATH ------------------
samsonboxverb = _ : schroederallp : schroeder4comb;
process = os.impulse : samsonboxverb <: _,_;


// Sarebbe necessario decorrelare in uscita le somme dei comb
// per avere un buon effetto spaziale.
// nel modello originale è incluso un mixer
// con 4 differenti uscite
// che hanno 4 differenti linee di ritardo, che sono :
// A = z-0.046fs
// B = z-0.057fs
// C = z-0.041fs
// D = z-0.054fs
```

3.6 Riverbero di J.Moorer

```
// ----------------------------------------
// RIVERBERO DI MOORER
// ----------------------------------------

// Importo libreria standard di FAUST
import("stdfaust.lib");



/*
Simulazione di Riverbero secondo il modello di James A. Moorer
il punto sorgente che coincide col punto di ascolto
*/



// ------------ FILTER SECTION ------------
    // LOWPASS FEEDBACK COMB FILTER
    lfbcf(delsamps, g, lowcut) =
        // lfbcf(delay in samples, comb filter gain, lowcut)
        (+ : @(delsamps-1) : _*lowcut : +~(_ : *(1- lowcut)))~
        *(g) : mem;
        // process = _ : lfbcf(3000, 0.999, 0.2) <:_,_;

    // ALLPASS FILTER
    apf(delaysamples, g) =
          (+ : _ <: @(delaysamples-1), *(g)) ~
         *(-g) : mem, _ : + : _;
// ----------------------------------------


// ------------ EARLY REFLECTIONS NETWORK -
    early_reflections = reflections
    with {
        in_router(a,b,c,d,e,f,g) = a, b, c, d, e, f, g;
        // process = in_router;
```

```
        input = _ <: in_router;
        //process = input;

        // multitap delay lines (NO FEEDBACK)
        multitap_delay(frnt,bck,sx,dx,up,dwn,direct) =
        frnt@(4481) *0.2,
        bck@(2713) *0.2,
        sx@(3719) *0.2,
        dx@(3739) *0.2,
        up@(1877) *0.2,
        dwn@(1783) *0.2,
        direct *0.2;
        //process = early_reflections;

        out_router(a,b,c,d,e,f,g) = a+b+c+d+e+f+g;
        //process = out_router;

        // early reflections routing
        reflections = input : multitap_delay :> out_router;
    };
// ----------------------------------------

// ------------ LATE REFLECTIONS NETWORK --
    moorerverbtail = apf_section
    with{
        in_router(a,b,c,d,e,f) = a, b, c, d, e, f;
        //process = in_router;

        // COMBS FILTER SECTION
        comb_section =
        lfbcf(4481, 0.98, 0.8),
        lfbcf(2713, 0.98, 0.8),
        lfbcf(3719, 0.98, 0.8),
        lfbcf(3739, 0.98, 0.8),
        lfbcf(1847, 0.98, 0.8),
        lfbcf(1783, 0.98, 0.8);
        //process = comb_section;
```

```
        out_router(a,b,c,d,e,f) = a+b+c+d+e+f;
        //process = out_router;

        combsrouting =  early_reflections
        <: in_router : comb_section :> out_router;
        //process = input;

        apf_section = combsrouting : apf(556, 0.5) : @(4800);
        //process = apf_section;
    };
// ----------------------------------------

// ------------ OUT PATH ------------------
moorerverb = _<: moorerverbtail + early_reflections;
process = os.impulse : moorerverb <: _,_;
```

3.7 Riverbero Freeverb di Jezar at Dreampoint

```
// ----------------------------------------
// FREEVERB di Jezar at Dreampoint
// ----------------------------------------

// Importo libreria standard di FAUST
import("stdfaust.lib");



/*
Simulazione di Riverbero secondo il modello di Jezar at
Dreampoint.
Utilizza 4 Allpass di Schroeder in serie,
ed 8 Schroeder-Moorer Filtered-feedback comb-filters in
parallelo.
*/



// ------------ FILTER SECTION ------------
    // LOWPASS FEEDBACK COMB FILTER
    lfbcf(delsamps, g, lowcut) =
        // lfbcf(delay in samples, comb filter gain, lowcut)
        (+ : @(delsamps-1) : _*lowcut : +~(_ : *(1- lowcut)))~
        *(g) : mem;
        // process = _ : lfbcf(3000, 0.999, 0.2) <:_,_;

    // ALLPASS FILTER
    apf(delaysamples, g) =
          (+ : _ <: @(delaysamples-1), *(g)) ~
         *(-g) : mem, _ : + : _;
// ----------------------------------------

// ------------ EARLY REFLECTIONS NETWORK -
    early_reflections = reflections
    with {
```

```
        in_router(a,b,c,d,e,f,g) = a, b, c, d, e, f, g;
        // process = in_router;

        input = _ <: in_router;
        //process = input;

        // multitap delay lines (NO FEEDBACK)
        multitap_delay(frnt,bck,sx,dx,up,dwn,direct) =
        frnt@(1617) *0.4,
        bck@(1557) *0.4,
        sx@(1491) *0.4,
        dx@(1422) *0.4,
        up@(1277) *0.4,
        dwn@(1356) *0.4,
        direct *0.2;
        //process = early_reflections;

        out_router(a,b,c,d,e,f,g) = a+b+c+d+e+f+g;
        //process = out_router;

        // early reflections routing
        reflections = input : multitap_delay :> out_router;
    };
// ----------------------------------------

// ------------ LATE REFLECTIONS NETWORK --
    freeverbtail = apf_section
    with{
        in_router(a,b,c,d,e,f,g,h) = a, b, c, d, e, f, g, h;
        //process = in_router;

        // COMBS FILTER SECTION
        comb_section =
        lfbcf(1557, 0.84, 0.2),
        lfbcf(1617, 0.84, 0.2),
        lfbcf(1491, 0.84, 0.2),
        lfbcf(1422, 0.84, 0.2),
        lfbcf(1277, 0.84, 0.2),
```

```
        lfbcf(1356, 0.84, 0.2),
        lfbcf(1188, 0.84, 0.2),
        lfbcf(1116, 0.84, 0.2);
        //process = comb_section;

        out_router(a,b,c,d,e,f,g,h) = a+b+c+d+e+f+g+h;
        //process = out_router;

        combsrouting =  early_reflections
        <: in_router : comb_section :> out_router;
        //process = input;

        apf_section = combsrouting :
        apf(225, 0.5) : apf(556, 0.5)
        : apf(441, 0.5) : apf(341, 0.5);
        //process = apf_section;
    };
// ----------------------------------------

// ------------ OUT PATH ------------------
freeverb = _<: freeverbtail + early_reflections;
process = os.impulse : freeverb <: _,_;

// Freeverb mono channel.
// Processing for the second channel is obtained by adding
// an integer to each of the twelve delay-line lengths.
// This integer is called stereospread, and
// it's default value is 23.
```

4 Creazione di un riverbero digitale: freevrev

Illustrate le principali topologie e esposte le implementazioni dei riverberi in FAUST (Grame), ci occuperemo ora in questa ultima parte della creazione di un riverbero digitale: il "freevrev". Il freevrev non è nient'altro che una implementazione variante del freeverb: partendo dalla topologia del freeverb ho impostato un controllo T60 globale che viene applicato all'interno del codice localmente sul calcolo dei campioni di ogni comb filter "lfbcf", ho impostato un parametro per il controllo globale del filtro lowpass per il filtraggio dovuto all'aria, e infine il codice partendo dallo stesso input crea due processi paralleli di riverberazione che si differenziano nei campioni dei filtri: nei ritardi delle early reflections, dei comb e degli allpass nelle late reflections, arrivando così in due output differenti e simulando una stereofonia dovuta alle differenze di fase percepite.

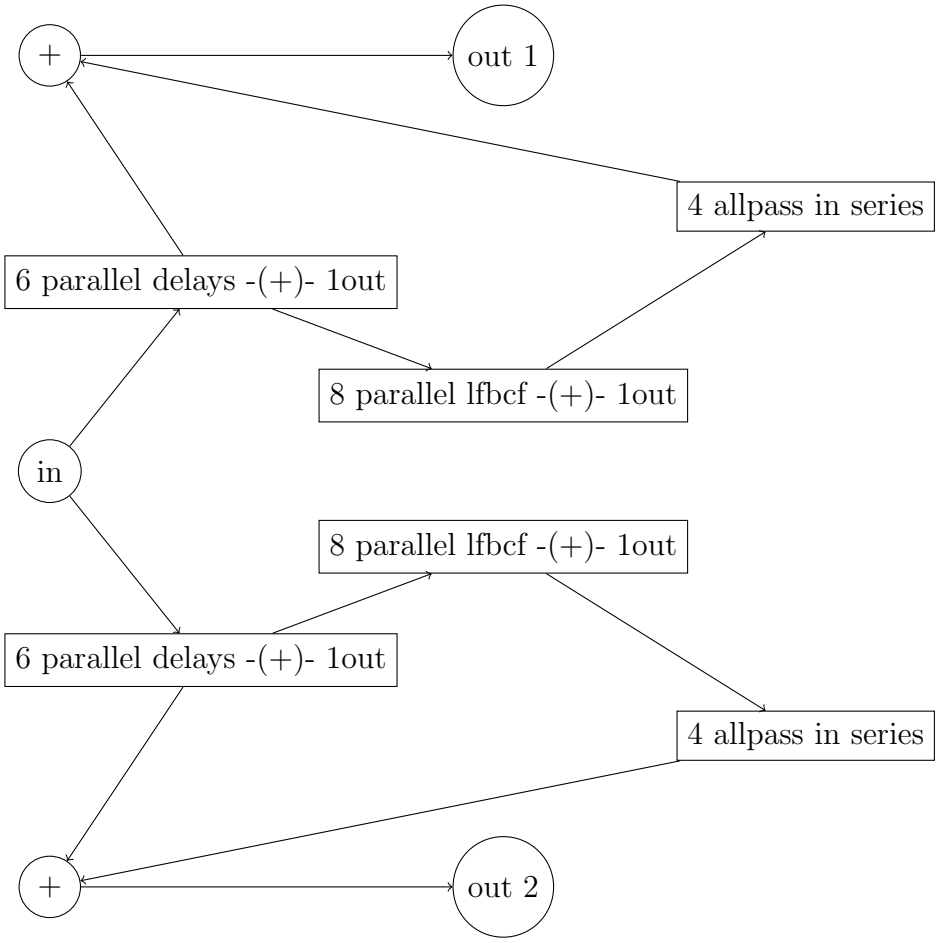

Figure 9: Topologia del freevrev

4.1 Riverbero freevrev di Luca Spanedda

```
// ----------------------------------------
// FREEVREV di Luca Spanedda.
// Implementazione basata su:
// FREEVERB di Jezar at Dreampoint
// ----------------------------------------

// Importo libreria standard di FAUST
import("stdfaust.lib");



/*
Simulazione di Riverbero secondo il modello di Jezar
at Dreampoint.
Utilizza 4 Allpass di Schroeder in serie,
ed 8 Schroeder-Moorer Filtered-feedback comb-filters
in parallelo.
2 Sezioni parallele distinte del processo.
*/



// ------------ CONTROLLI -----------------
gaincontrol = vslider("gain[style:knob]",1,0,1,0.001)
: si.smoo;
lowcutcontrol = vslider("lowcut[style:knob]",1,0,1,0.001)
: si.smoo;
t60control = vslider("decay-seconds[style:knob]",1,0,100,1)
: si.smoo;
// ----------------------------------------

// ------------ T60 SECTION ---------------
/*
Inserisci all'interno degli argomenti della funzione:
    - il valore in campioni del filtro
    che stai usando per il ritardo.
```

```
    - il valore di decadimento in T60
    (tempo di decadimento di 60 dB in secondi)
    = OTTIENI in uscita dalla funzione,
    il valore che devi passare come amplificazione
    alla retroazione del filtro per ottenere
    il tempo di decadimento T60 che si desidera
*/
// (samps,seconds) = give: samples of the filter,
// seconds we want for t60 decay
dect60(samps,seconds) =
1/(10^((3*(((1000 / ma.SR)*samps)/1000))/seconds));
// ----------------------------------------

// ------------ FILTER SECTION ------------
    // LOWPASS FEEDBACK COMB FILTER
    lfbcf(delsamps, g, lowcut) =
        // lfbcf(delay in samples, comb filter gain, lowcut)
        (+ : @(delsamps-1) : _*lowcut : +~(_ : *(1- lowcut)))~
        *(g) : mem;
        // process = _ : lfbcf(3000, 0.999, 0.2) <:_,_;

    // ALLPASS FILTER
    apf(delaysamples, g) =
          (+ : _ <: @(delaysamples-1), *(g)) ~
         *(-g) : mem, _ : + : _;
// ----------------------------------------

// ------------ EARLY REFLECTIONS NETWORK 1
    early_reflections1 = reflections1
    with {
        in_router(a,b,c,d,e,f,g) = a, b, c, d, e, f, g;
        // process = in_router;

        input = _ <: in_router;
        //process = input;

        // multitap delay lines (NO FEEDBACK)
        multitap_delay(frnt,bck,sx,dx,up,dwn,direct) =
```

```
        frnt@(1619),
        bck@(1559),
        sx@(1493),
        dx@(1423),
        up@(1277),
        dwn@(1361),
        direct;
        //process = early_reflections;

        out_router(a,b,c,d,e,f,g) = a+b+c+d+e+f+g;
        //process = out_router;

        // early reflections routing
        reflections1 = input : multitap_delay :> out_router;
    };
// ----------------------------------------

// ------------ EARLY REFLECTIONS NETWORK 2
    early_reflections2 = reflections2
    with {
        in_router(a,b,c,d,e,f,g) = a, b, c, d, e, f, g;
        // process = in_router;

        input = _ <: in_router;
        //process = input;

        // multitap delay lines (NO FEEDBACK)
        multitap_delay(frnt,bck,sx,dx,up,dwn,direct) =
        frnt@(1621),
        bck@(1567),
        sx@(1499),
        dx@(1433),
        up@(1283),
        dwn@(1373),
        direct;
        //process = early_reflections;

        out_router(a,b,c,d,e,f,g) = a+b+c+d+e+f+g;
```

```
        //process = out_router;

        // early reflections routing
        reflections2 = input : multitap_delay :> out_router;
    };
// ----------------------------------------

// ------------ LATE REFLECTIONS NETWORK 1-
    freeverbtail1(seconds,absorb) = apf_section1
    with{
        in_router(a,b,c,d,e,f,g,h) = a, b, c, d, e, f, g, h;
        //process = in_router;

        // COMBS FILTER SECTION
        comb_section =
        lfbcf(1559, dect60(1559,seconds), absorb),
        lfbcf(1619, dect60(1619,seconds), absorb),
        lfbcf(1493, dect60(1493,seconds), absorb),
        lfbcf(1423, dect60(1423,seconds), absorb),
        lfbcf(1277, dect60(1277,seconds), absorb),
        lfbcf(1361, dect60(1361,seconds), absorb),
        lfbcf(1187, dect60(1187,seconds), absorb),
        lfbcf(1117, dect60(1117,seconds), absorb);
        //process = comb_section;

        out_router(a,b,c,d,e,f,g,h) = a+b+c+d+e+f+g+h;
        //process = out_router;

        combsrouting =  early_reflections1
        <: in_router : comb_section :> out_router;
        //process = input;

        apf_section1 = combsrouting :
        apf(223, 0.5) : apf(563, 0.5) :
        apf(441, 0.5) : apf(341, 0.5);
        //process = apf_section;
    };
// ----------------------------------------
```

```
// ------------ LATE REFLECTIONS NETWORK 2-
    freeverbtail2(seconds,absorb) = apf_section2
    with{
        in_router(a,b,c,d,e,f,g,h) = a, b, c, d, e, f, g, h;
        //process = in_router;

        // COMBS FILTER SECTION
        comb_section =
        lfbcf(1621, dect60(1621,seconds), absorb),
        lfbcf(1567, dect60(1567,seconds), absorb),
        lfbcf(1499, dect60(1499,seconds), absorb),
        lfbcf(1433, dect60(1433,seconds), absorb),
        lfbcf(1283, dect60(1283,seconds), absorb),
        lfbcf(1373, dect60(1373,seconds), absorb),
        lfbcf(1187, dect60(1187,seconds), absorb),
        lfbcf(1123, dect60(1123,seconds), absorb);
        //process = comb_section;

        out_router(a,b,c,d,e,f,g,h) = a+b+c+d+e+f+g+h;
        //process = out_router;

        combsrouting =  early_reflections2
        <: in_router : comb_section :> out_router;
        //process = input;

        apf_section2 = combsrouting :
        apf(227, 0.5) : apf(557, 0.5) :
        apf(433, 0.5) : apf(353, 0.5);
        //process = apf_section;
    };
// ----------------------------------------

// ------------ OUT PATH ------------------
freeverb1(t60,lowcut) = _ <: freeverbtail1(t60,lowcut)
+ early_reflections1;
freeverb2(t60,lowcut) = _ <: freeverbtail2(t60,lowcut)
+ early_reflections2;
```

```
process = _ *(gaincontrol*0.04) <:
freeverb1(t60control,lowcutcontrol),
freeverb2(t60control,lowcutcontrol);
```

Ulteriore documentazione (codici, referenze, compilazioni, ricerche) sono esposte all'interno del seguente repository Github che ho dedicato alla Riverberazione Digitale:

`https://github.com/LucaSpanedda/Riverberazione_Digitale_in_FAUST`

References

[1] Schroeder, Manfred R. *Natural Sounding Artificial Reverberation.* Bell Telephone Laboratories, Inc., Murray Hill, NJ, October 1, 1961

[2] James Moorer *About This Reverberation Business.* January 1985Computer Music Journal 3(2):605-639

[3] John Stautner and Miller Puckette *Designing Multi-Channel Reverberators.* Computer Music Journal Vol. 6, No. 1 (Spring, 1982), pp. 52-65 (14 pages)

[4] Jon Dattorro *Effect Design* 1: Reverberator and Other Filters.* CCRMA, Stanford University, Stanford, CA, USA

[5] Julius O. Smith III *PHYSICAL AUDIO SIGNAL PROCESSING FOR VIRTUAL MUSICAL INSTRUMENTS AND AUDIO EFFECTS.*

[6] Hal Chamberlin *Musical Applications of Microprocessors.* (The Hayden microcomputer series)