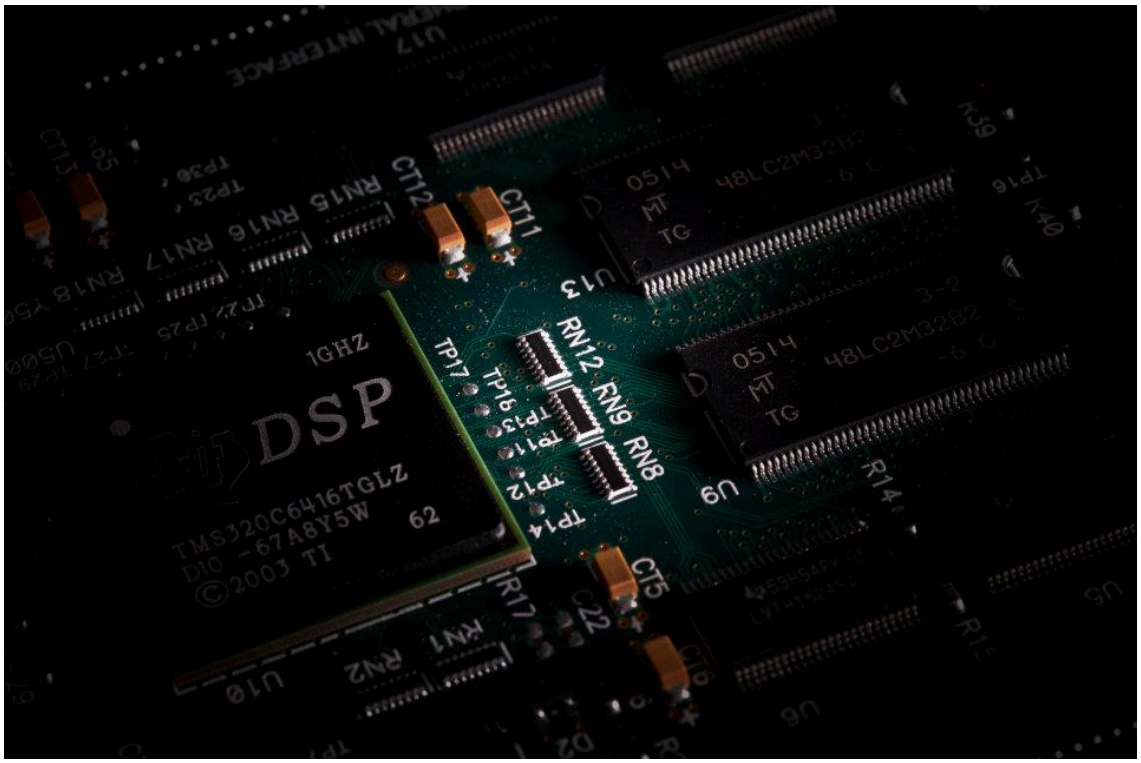
CHALMERS



Musikeffekter med signalprocessor

Realtidsimplementation på plattformen C6416T DSK

LARS BACKEFELDT
SVEN JACOBSSON
HAMPUS MALMBERG
HAMPUS EKSTRAND
EMIL LINDQVIST
JOHAN TOFT

Kandidatarbete vid institutionen för signaler och system
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, 2012 SSYX02-12-37

Förord

Denna rapport avspeglar ett kandidatarbete som utförts vid institutionen för signaler och system och omfattar 15 högskolepoäng. Projektgruppen vill tacka institutionen för ett intressant, utmanande och mycket lärorikt projekt.

Flera personer har under projektets gång bidragit med kunnande och engagemang. Ett extra stort tack vill gruppen uttrycka till professor Tomas McKelvey som ständigt funnits till hands och alltid lett oss i rätt riktning då vi behövt det som mest.

Chalmers 2012

Lars Backefeldt
Hampus Ekstrand
Sven Jacobsson
Emil Lindqvist
Hampus Malmberg
Johan Toft

Abstract

This report describes the different steps involved in implementing audio effects designed to run in real-time, on a digital signal processor (DSP). The presented audio effects are used in a wide variety of applications and are considered to be part of the default suite in any advanced sound processing software.

The effects have been implemented on the development board C6416T DSK, designed around the DSP chip TMS320C6416T. The calculations performed on the development board employs fixed-point arithmetic which greatly affects the necessary steps to be taken, that is, to get the effects from the drawing board, into the hardware.

The purpose of the report is to facilitate and engage the reader, to construct and to implement audio effects on the same or a similar platform. Therefore the reader will be presented with the discussions and solutions, developed throughout the course of the project.

Ten of the more common audio effects used today have been implemented to run in real-time, each with different difficulties in terms of design and implementation. The resulting effects have been evaluated in terms of quality and computational efficiency and may be combined to construct new audio effects.

Sammanfattning

I rapporten behandlas de olika stegen för att modellera och implementera flera välkända musikeffekter. Musikeffekterna har implementeras för att i realtid kunna köras på en digital signalprocessor (DSP) och kan anses utgöra grunden i flera avancerade ljudbehandlingsverktyg.

Musikeffekterna har implementerats på utvecklingskortet C6416T DSK utvecklat kring signalprocessorn TMS320C6416T. Den valda plattformen utför beräkningar med fixtalsaritmetik vilket har stor inverkan på hur effekterna implementeras.

Syftet med rapporten är att beskriva de framtagna musikeffekterna och ge läsaren underlag för att utföra en liknande implementation på samma, eller motsvarande plattform. De lösningar och diskussioner som förs i rapporten är alltså inte begränsade till en specfik hårdvara.

Tio av de vanligaste musikeffekterna har implementerats för att köra i realtid. Varje effekt medför nya svårigheter vad gäller design och implementation. Systemen har utvärderats med avseende på funktion, och belastning av den digitala signalprocessorn, och kan kombineras för att åstadkomma fler musikeffekter.

Innehåll

1 Inledning **1**
- 1.1 Bakgrund . . . 1
- 1.2 Syfte . . . 1
- 1.3 Problemformulering . . . 2
- 1.4 Kravspecifikation och avgränsningar . . . 2

2 Utvecklingskort **3**
- 2.1 Beräkningsenhet . . . 3
- 2.2 Ljudenhet . . . 3
- 2.3 Fixtalsaritmetik . . . 4
  - 2.3.1 Q-format . . . 5
  - 2.3.2 Filterimplementation . . . 6
- 2.4 Bufferthantering . . . 8
  - 2.4.1 Blockbaserad bufferthantering . . . 8
  - 2.4.2 Cirkulär bufferthantering . . . 9

3 Tidsinvarianta musikeffekter **11**
- 3.1 Equalizer . . . 11
  - 3.1.1 Parametrisk equalizer . . . 11
  - 3.1.2 Grafisk equalizer . . . 17
- 3.2 Kompressor . . . 24
- 3.3 Delay/Echo . . . 28
- 3.4 Reverb . . . 31
  - 3.4.1 Tidiga reflektioner . . . 32
  - 3.4.2 Sen efterklang . . . 34
  - 3.4.3 Modellerat impulssvar . . . 36

4 Tidsvarianta musikeffekter **39**
- 4.1 Flanger . . . 39
- 4.2 Phaser . . . 41
- 4.3 Chorus . . . 44
- 4.4 Tremolo . . . 46
- 4.5 Vibrato . . . 47

5 Diskussion **51**

6 Slutsats **53**

A FFT (Fast Fourier Transform **57**

B Musikteori **59**

C Panorering **61**

D Vågekvationen **63**

1 Inledning

Med benämningen *musikeffekt* avses i detta arbete ett system som förändrar ett ljudspår för att uppnå en önskad verkan. De musikeffekter som behandlas i rapporten är equalizer, kompressor, delay, echo, chorus, flanger, phaser, vibrato och tremolo.
Samtliga effekter är vanligt förekommande inom musikindustrin och utgör grunden för många andra effekter. Kortfattade beskrivningar av varje effekt ges löpande i texten, för noggrannare beskrivningar av effekternas ljudkarakteristik och användning uppmanas läsaren söka andra källor.

1.1 Bakgrund

Vid telefonens introduktion var långa förbindelser ett stort problem. De höga frekvenserna dämpades i större utsträckning än de lägre, vilket ledde till att det blev svårt att uttyda vad som sades. Problemet löstes genom att använda en *equalizer*, en av de vanligare musikeffekterna idag. En analog krets konstruerades som skulle göra frekvensbandet lika (equal) vid slutet av transmissionsledningen [1].

Under 1960-talet tog inspelandet av musik på flera kanaler ett språng, vilket tog behandling av musikstycken till en helt ny nivå. Möjligheten med flera spår gav större kontroll av slutprodukten där musikeffekterna kunde appliceras på separata delar av musiken. Det var först nu de musikeffekter som används idag fick en fast plats i studion.

Till en början var musikeffekternas konstruktion helt representerade av elektronrör och andra analoga komponenter. Dock blev dessa komponenter, till följd av värmeutveckling och de höga spänningarna, allt mer förlegade efter transistorns framträdande i den kommersiella marknaden. Den digitala signalbehandlingen tog år 1978 ett stort språng då den första signalprocessorn lanserades av Texas Instruments. Denna typ av processor är särskilt anpassad för att utföra intensiva beräkningar inom signalbehandling och förekommer idag i bland annat mobiltelefoner, förstärkare och CD-spelare [2].
Avancerade musikeffekter kan idag med hjälp av dessa processorer utföras i realtid. Med mindre, mer effektiva, kretsar och överlägsen flexibilitet är de digitala musikeffekterna här för att stanna.

1.2 Syfte

Syftet med rapporten är att redogöra hur musikeffekter formas och implementeras i en signalprocessor som utför fixtalsberäkningar. Signalbehandlingen skall utföras i realtid med krav på prestanda efter de förutsättningar hårdvaran erbjuder. Rapportens syfte är inte att redgöra hur signalbehandling utförs specifikt för den valda hårdvaran, utan snarare att presentera allmänt vedertagna tekniker förknippade med realtidsimplementation på en fixtalsprocessor.

Rapporten redogör hur de olika effekterna är uppbyggda och hur de påverkar ett ljudspår.
Musikeffekternas verkan skall kunna bestämmas av vanligt förekommande parametrar inom kommersiell ljudbehandling och skall kunna varieras för att styra effektens verkan. Dessutom skall flera musikeffekter kunna utföras samtidigt.

Den tänkte läsaren är intresserad av att implementera musikeffekterna på ett liknande system eller alternativt implementera ytterligare effekter på samma system. Syftet är därför också att läsaren ska kunna ta del av de resonemang som förts och vilka beslut som tagits under arbetets gång.

1.3 Problemformulering

Hur de olika effekterna är uppbyggda måste undersökas. Algoritmerna skall vara uppbyggda så att deras verkan skall kunna styras av ett fåtal parametrar.

Musikeffekterna implementeras på en signalprocessor, vilket innebär att hur en sådan fungerar och programmeras måste klargöras. Processorn som används utför fixtalsberäkningar, vilka konsekvenser detta medför måste alltså utredas.

För att analogt ljud skall kunna behandlas i ett diskret system måste det först A/D-omvandlas. När algoritmen är utförd skall de digitala signalerna åter D/A-omvandlas till analog ljud. Signalbehandling skall dessutom ske på ljud i stereo. Hur dessa aspekter påverkar implementationen måste därför utredas.
För att implementera flera av effekterna krävs att en större mängd sampel under en tidsperiod samlas in. Minnet i en DSP är av begränsad storlek vilket innebär att effektiv bufferthantering är ett måste för att utföra uppgiften.
Dessutom ställer en realtidsimplementation krav på systemets effektivitet, då filtrering endast får ske under en begränsad tidsperiod. Ett avvägande måste därför göras mellan systemets fördröjning och algoritmernas beräkningskomplexitet.

Efter implementation skall musikeffekterna utvärderas med avseende på funktion, och beräkningsintensivitet för att uppfylla de kravspecifikationer som statueras nedan.

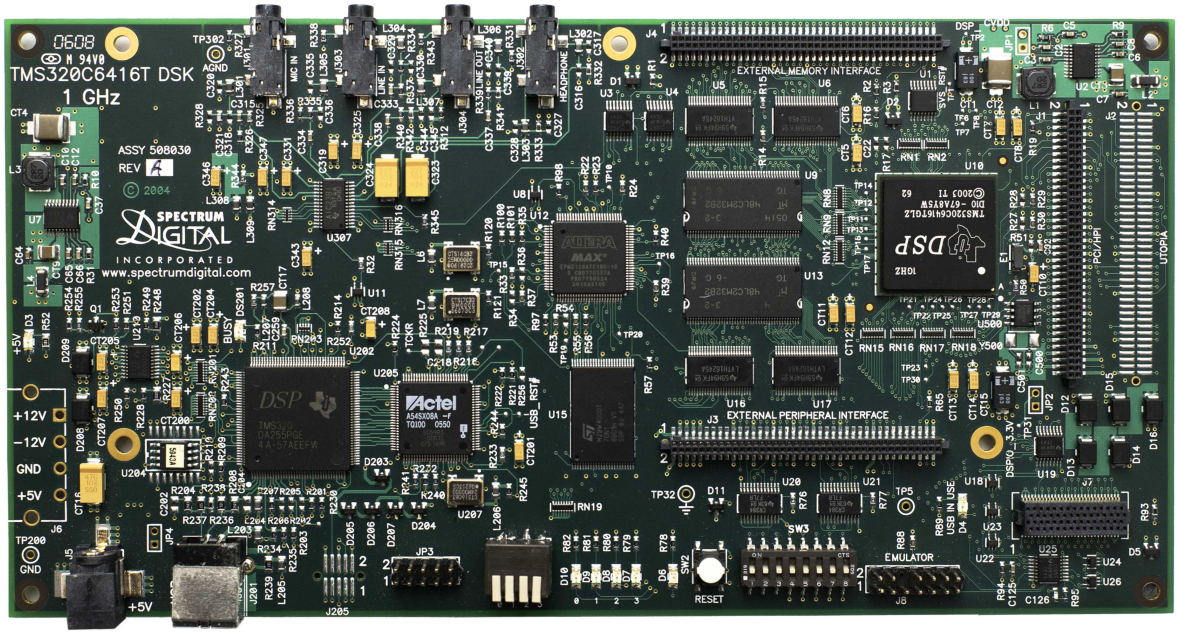
1.4 Kravspecifikation och avgränsningar

Signalbehandling skall ske i realtid, vlket innebär att tiden då ljudet kan behandlas är begränsat. För att behandlingen skall klassas som en realtidsimplementation beslutas att systemet endast får fördröja signalen 20 ms.
Flera musikeffekter skall av systemet kunna utföras samtidigt vilket innebär att en enskild effekt inte får belasta processorn mer än 33.3%. På så sätt ska tre olika musikeffekter kunna användas samtidigt.

Rapporten är avgränsad till att endast behandla musikeffekterna nämnda i inledningen. Även om effekterna är framtagna för att kunna styras av ett antal parametrar har inget fokus lagts på att ta fram ett användargränssnitt.

2 Utvecklingskort

Musikeffekterna implementeras på utvecklingskortet C6416T DSK, se figur 2.1, designat av Spectrum Digital och utvecklat kring signalprocessorn TMS320C6416T DSP.



Figur 2.1: Utvecklingskortet C6416T DSK.

Utvecklingskortet ansluts till datorn via ett USB-gränssnitt där kommunikationen styrs av den inbyggda JTAG-emulatorn. Datautbyte mellan DSP och PC kan ske i realtid vilket utnyttjas för analys och för att i realtid styra systemet.
På utvecklingskortet finns fyra DIP-switchar och fyra LED-lampor vilka används för enklare styrning och indikation.

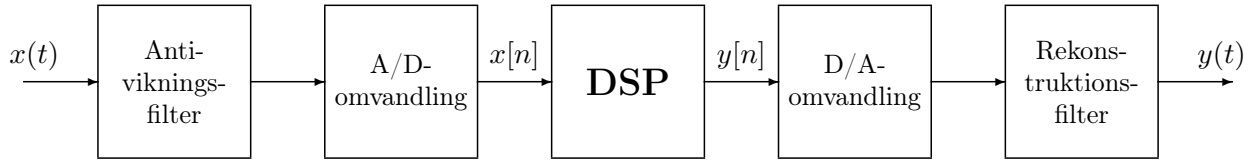
Utvecklingskortet består av ett antal enheter för att processorn skall kunna implementeras i diverse tillämpningar. Dock kan det i huvudsak betraktas som två delar, beräkningsenhet (DSP) och ljudenhet (stereokodek).

2.1 Beräkningsenhet

Beräkningsenheten TMS320C6416T DSP, med klockfrekvensen 1 GHz, är en 32-bitars signalprocessor från Texas Instruments som utför fixtalsberäkningar och har 1 MB internt RAM-minne. Centralprocessorn (CPU) består av två multiplikatorer och sex aritmetiska logiska enheter. Den är också utrustad med en EDMA-modul (Enhanched Direct Memory Access) där en stödprocessor har direkt åtkomst till det interna minnet. Denna enhet kan alltså avlasta CPU genom att hantera överföringar mellan minnesmoduler samtidigt som signalbehandling utförs [3].
Anslutet till processorn finns även 512 kB flashminne samt 16 MB SDRAM

2.2 Ljudenhet

TLV320AIC23 (AIC23) är ett 32-bitars stereokodek som kvantiserar inkommande ljud och skickar det till DSP. Utgående digitala sampel avkodas för att lämna kortet som analogt ljud.



Figur 2.2: Ljudets väg genom utvecklingskortet

Inkommande ljud, $x(t)$, från linje- eller mikrofoningången A/D-omvandlas (kvantiseras) till ett 16-bitars binärt tal, $x[n]$, som kan behandlas i en DSP. Kontinuerliga signaler representeras alltså av insignalens amplitudnivå vid den tidpunkt sampling sker. Upplösningen beror på ordlängden av det kvantiserade värdet och på det förspecificerade spänningsintervall som skall täcka upp signalens högsta och lägsta nivåer. Kvantiseringsfelet kan antas vara jämnt fördelat mellan $-LSB/2$ och $LSB/2$ där LSB betecknar den minst signifikanta biten [4]. Upplösningen i en 16-bitars A/D-omvandling bestäms av LSB enligt ekvation 2.1.

$$1\ LSB = \Delta = \frac{V_{max} - V_{min}}{2^{16} - 1} \tag{2.1}$$

Spänningsnivån i inkommande ljud från linjeingången ligger mellan $0-1\ V_{RMS}$ [5]. Vilket innebär att upplösningen i inkommande ljud är

$$1\ LSB = \Delta = \frac{2\sqrt{2}}{2^{16} - 1} \approx 4.3 \cdot 10^{-5}\ V \tag{2.2}$$

Samplingshastigheten kan ställas in på vanligt förekommande frekvenser inom ljudbehandling mellan 8 och 96 kHz. För att en samplad signal skall kunna återskapas fullständigt måste samplingsteoremet uppfyllas. Teoremet säger att en kontinuerlig signal unikt kan bestämmas av dess sampel om $f_S > 2f_M$, där $f_S = 1/T_S$ betecknar samplingsfrekvensen och f_M signalens högsta frekvensinnehåll [6]. Innan sampling filtreras ljudet genom ett antivikningsfilter för att eventuellt högfrekvent innehåll inte skall orsaka vikning.
Det mänskliga örat sägs kunna registrera ljud mellan cirka 20 Hz till 22 kHz [7]. För att då kunna rekonstruera ljudsignaler måste samplingsfrekvensen väljas högre än 44 kHz. För att få lite marginal valdes samplingsfrekvensen till 48 kHz genomgående i rapporten.

När signalbehandlingen utförts D/A-omvandlas den behandlade signalen, $y[n]$, och skickas ut på linje- eller hörlursutgången. D/A-omvandlingen sker genom att signalens binära värde vid tidpunkterna $T_S, 2T_S, 3T_S, ...$ översätts till en spänningsnivå. Ljudenheten kommer då att hålla denna spänningsnivå konstant tills nästa sampel inkommer och processen upprepas. Resultatet blir en styckvis konstant utsignal. För att erhålla en jämnare utsignal, $y(t)$, interpoleras ljudet av ett rekonstruktionsfilter.

Datautbyte mellan AIC23 och DSP sker via den seriella kanalen McBSP (Multichannel Buffered Serial Port). Eftersom ljudenheten endast kan skicka och ta emot data vid vissa specifika tidpunkter måste ljudenheten och processorn synkronisera dataöverföringen, vilket åstadkoms med avbrottsbaserad datahantering. Då ett nytt sampel från ljudenheten finns tillgängligt kommer processorn automatiskt hoppa till en fördefinierad plats i programmet. Själva synkroniseringen kräver således inget arbete av processorn.

2.3 Fixtalsaritmetik

I en DSP representeras data i binär form. Negativa värden representeras genom att talet skrivs på tvåkomplementsform, där den mest signifikanta biten (MSB) representerar talets tecken.

Ett godtyckligt tal kan då approximeras enligt

$$x = -b_{W-1}2^{W-1} + \sum_{i=0}^{W-2} b_i 2^i, \quad b_i \in \{0, 1\}. \tag{2.3}$$

Där W betecknar det binära talets ordlängd (antal bitar). Med fixtalsaritmetik menas alltså att tal representeras och beräknas i heltal.

En DSP som utför beräkningar i fixtal är både snabbare och billigare än de som räknar i flyttal [4]. Dock minskas det dynamiska omfånget (förhållandet mellan en signals största och minsta värde) och aspekter såsom avrunding och kvantisering måste beaktas.

2.3.1 Q-format

En binärt tals dynamiska omfång begränsas av dess ordlängd. Ett 16-bitars tal på formen ovan rymmer således bara tal inom intervallet $[-32768,\ 32767]$. Tal som ligger utanför detta intervall kommer då att representeras som ett mindre tal inom mängden. Händelsen benämns överskridning och kan orsaka stora störningar i utsignalen [2].
För att kunna representera tal vars absolutbelopp är större än 32768 med 16-bitars ordlängd måste dessa normaliseras. Talet representeras då istället som ett bråktal vars decimala värde ges av

$$x_Q = -b_{N-1}2^0 + b_{N-2}2^{-1} + ... + b_0 2^{-(N-1)}, \quad b_i \in \{0, 1\}. \tag{2.4}$$

Tal som skrivs på denna form sägs vara angivna i Q-format. Ett 16-bitars tal som anges enligt ekvation 2.4 sägs vara angivet på Q15-format och består således av en teckenbit samt 15 bråkdelsbitar.

Genom att skriva binära tal på Q-format kan överskridning vid multiplikation undvikas eftersom produkten av två normaliserade tal alltid kommer vara mindre än ett. Produkten anges då av ett 32-bitars tal på Q30-format med en extra teckenbit. För att återgå till ett 16-bitars tal på Q15-format som kan hanteras av D/A-omvandlaren måste de 15 minst signifikanta bitarna därför trunkeras.
Detta kommer att ge upphov till en skillnad mellan ursprungssignalen och den trunkerade signalen kallad trunkeringsfelet, e_t, som uppskattas enligt

$$0 \leq |e_t| \leq 2^{-15} - 2^{-30} \approx 3.1 \cdot 10^{-5}\ V. \tag{2.5}$$

Vilket är i samma storleksordning som upplösningen och kommer att försummas.

Notationen av att binära tal skrivs på Q-format förhindrar dessvärre inte överskridning vid addition och subtraktion av två tal. Lösningen är då att insignalen halveras upprepade gånger tills dess att inga överskridningar längre kan uppstå. Att halvera ett tal i fixtal motsvarar att skifta talet ett steg åt höger.

Då ljudet filtreras i ett system som endast dämpar insignalen kan denna skalning i volym kompenseras för genom att efter filtrering skifta utsignalen lika många steg åt vänster som insignalen tidigare skiftades åt höger.
Om systemet däremot har en förstärkande verkan fungerar inte denna metod eftersom överskridning då riskerar ske i slutstadiet. Istället måste då insignalen från början skalas ner ett antal gånger.

För varje nedskalning tillåts systemet fördubbla amplituden av signalen på bekostnad av försämrad upplösning i ljudet. Eftersom upplösningen på inkommande ljud enligt avsnitt 2.2 är

såpass bra kommer inkommande ljud skalas ner tre gånger vilket tillåter att signalen förstärks 9 dB. Skalningen innebär att upplösningen på inkommande ljud då kommer att vara

$$\Delta = \frac{2\sqrt{2}}{2^{13}-1} \approx 3.4 \cdot 10^{-4}\ V. \tag{2.6}$$

Trots att denna upplösning är betydligt sämre än innan skalningen kommer den endast ge upphov till mindre fluktuationer i utsignalen.

2.3.2 Filterimplementation

Differentialekvationen som beskriver sambandet mellan in- och utsignal för IIR-filter av ordning N skrivs på formen

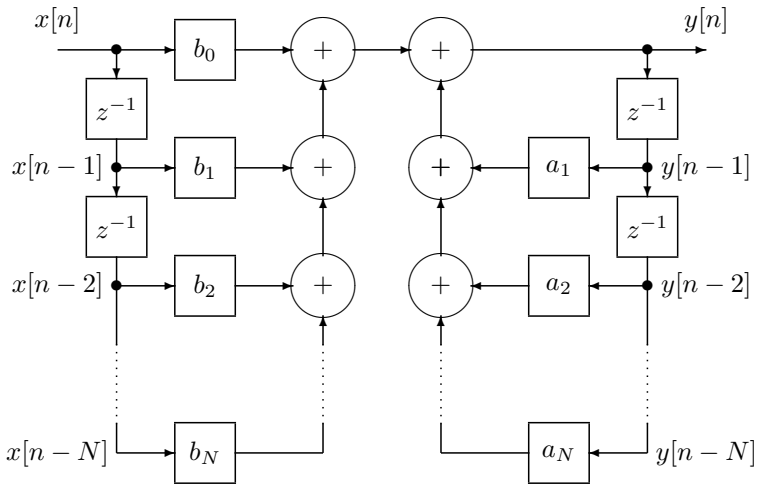
$$y[n] = \sum_{i=0}^{N} b_i x[n-i] - \sum_{j=1}^{N} a_j y[n-j]. \tag{2.7}$$

Utsignalen beror alltså på $N+1$ tillstånd av insignalen och N tillstånd av utsignalens tidigare värden. Överföringsfunktionen ges då genom z-transformering av differentialekvationen som

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^{N} b_i z^{-i}}{1 + \sum_{j=1}^{N} a_j z^{-j}}. \tag{2.8}$$

Vilket då $a_j = 0,\ j = 1, ..., N$ även beskriver ett FIR-filter.

Då överföringsfunktionen i ekvation 2.8 skall realiseras i en processor måste den implementeras på en form som kan hanteras av en dator.
En överföringsfunktion kan skrivas på direkt form I, se figur 2.3. Blockschemat kan beskrivas som en direkt översättning av ekvation 2.8. Om ett filter av ordning N skall realiseras enligt denna modell kommer det att krävas $2N$ minnesplatser, $2N-1$ multiplikationer och $2N-2$ additioner [8].



Figur 2.3: Blockschemerepresentation av ett IIR-filter på direkt form I.

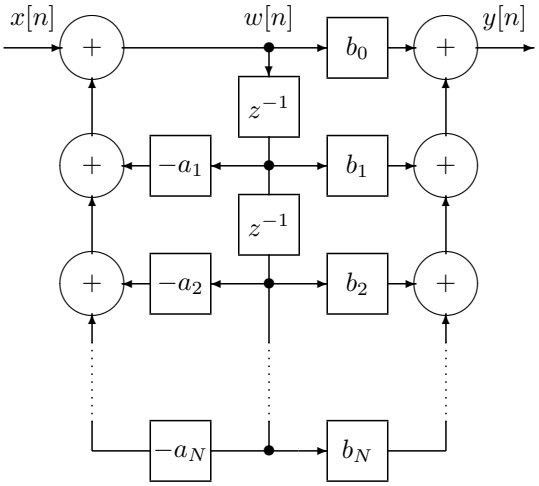
Överföringsfunktionen i ekvation 2.8 kan också skrivas som

$$Y(z) = \sum_{i=0}^{N} b_i z^{-i}\ W(z). \tag{2.9}$$

där variabeln $W(z)$ införts som

$$W(z) = \frac{X(z)}{1 + \sum_{j=1}^{N} a_j z^{-j}} \Rightarrow W(z) = X(z) - \sum_{j=1}^{N} a_j z^{-j}\ W(z). \tag{2.10}$$

Om ett filter implementeras enligt uttrycket ovan sägs det vara skrivet på direkt form II, se figur 2.4. Antalet fördröjningar har alltså minimerats medan antalet multiplikationer och additioner är samma som i direkt form I. Filter implementeras alltså med fördel på direkt form II eftersom det beräkningsmässigt är mindre krävande än direkt form I. Dock är metoden inte tillämpbar om koefficienterna i nämnarpolynomet kräver skalning eftersom mellanlagringsvariabeln, $w[n]$, beror av dessa.



Figur 2.4: Blockschemerepresentation av ett IIR-filter på direkt form II.

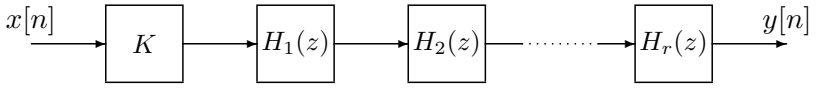
Genom att i överföringsfunktionen för ett IIR-filter multiplicera både nämnare och täljare med z^N kan denna skrivas som

$$H(z) = K\frac{(z - z_1)(z - z_2)...(z - z_N)}{(z - p_1)(z - p_2)...(z - p_N)} = K\prod_{i=1}^{N}\frac{z - z_i}{z - p_i} \tag{2.11}$$

Faktorisering av överföringsfunktionen för ett IIR-filter innebär alltså att ett filter av ordning N kan skrivas som produkten av filter av lägre ordning enligt

$$H(z) = K \cdot H_1(z)H_2(z)...H_r(z). \tag{2.12}$$

Där alltså utsignalen från ett filter är insignalen till nästa enligt figur 2.5.



Figur 2.5: Kaskadkopplade filter

Notera att samma filter kan kaskadkopplas på flera olika sätt genom att variera ordningstalet på de olika kaskadstegen eller para ihop olika nollställen med olika poler.
I teorin är samtliga kaskadkopplingar ekvivalenta med ursprungspolynomet men vid implementation i en fixtalsprocessor kan olika kombinationer ge olika resultat [9]. Detta är en följd av att filterkoefficientera måste skrivas som fixtal och kommer då att kvantiseras. Ett FIR-filter är

relativt tåligt mot kvaniseringseffekter medan IIR-filter på grund av dess rekursiva egenskaper är betydligt känsligare. Om denna aspekt inte beaktas kan ett systems karakteristik helt förändras.

Beakta nu inverkan av kvantisering av filterkoefficienter genom att beakta den kvantiserade motsvarigheten av överföringsfunktionen, $\tilde{H}(z)$, som ges av

$$\tilde{H}(z) = \frac{\sum_{i=0}^{N} \tilde{b}_i z^{-i}}{1 + \sum_{j=1}^{N} \tilde{a}_j z^{-j}}. \tag{2.13}$$

där koefficienterna $\tilde{b}_i = b_i + \Delta b$ och $\tilde{a}_i = a_i + \Delta a$ betecknar de egentliga koefficienternas kvantiserade motsvarighet. Alltså motvsvarar Δb och Δa de kvantiserade koefficienternas avvikelse från det sanna värdet. För att uppskatta avvikelsens inverkan på filtret beaktas det kvantiserade nämnarpolynomet, $\tilde{D}(z)$, som ges av

$$\tilde{D}(z) = 1 + \sum_{j=1}^{N} \tilde{a}_j z^{-j} \tag{2.14}$$

vilket också kan skrivas som

$$\tilde{D}(z) = \prod_{i=1}^{N} (1 - \tilde{p}_i z^{-1}) \tag{2.15}$$

Där $\tilde{p}_i = p_i + \Delta p$ betecknar de kvantiserade polernas placering. Storleken av avvikelsen hos de kvantiserade polernas placering ges av ekvation 2.16 nedan [9].

$$|\Delta p_i| = |\sum_{k=1}^{N} \frac{\partial p_i}{\partial a_k}| = |\sum_{k=1}^{N} \frac{p_i^{N-k}}{\prod_{l=1, l\neq i}^{N} (p_i - p_l)} \Delta a_k| \tag{2.16}$$

Ur ekvation 2.16 framgår att avvikelsen i en pols placering beror på samtliga koefficienters avvikelse. Alltså ökar kvantiseringsfelet med filtrets ordning.
För att minska påverkan föredras att högre ordningens filter implementeras som kaskadkopplade filter av lägre ordning. I många praktiska tillämpningar konstrueras filter på kanonisk form som kaskadkopplade steg av andra ordningens filter [2].

Samtliga musikeffekter som är uppbyggda av IIR-filter implementeras på direkt form som kaskadkopplade steg av andra ordningen. För de fall där koefficienterna inte behöver skalas kommer direkt form II att tillämpas, om denna skalning är nödvändig används istället direkt form I.

2.4 Bufferthantering

Ofta behövs äldre sampel för att beräkna utsignalen, därför krävs en datastruktur där tillräckligt många sampel sparas undan för senare åtkomst. Detta kallas bufferthantering.

2.4.1 Blockbaserad bufferthantering

I vissa tillämpningar ökar prestandan då block av data behandlas åt gången. En effektiv datastruktur är då en förutsättning.
Blockbaserad filtrering förändrar hur exekvering utförs i en realtidsimplementation. Tidigare kunde varje sampel behandlas var för sig under den T_s sekunder långa tidsram tills nästa sampel mottagits. Nu skall istället N sampel behandlas åt gången vilket måste ske under tidsramen $N \cdot T_s$ sekunder, medan nya sampel fortfarande inkommer med T_s sekunders mellanrum.

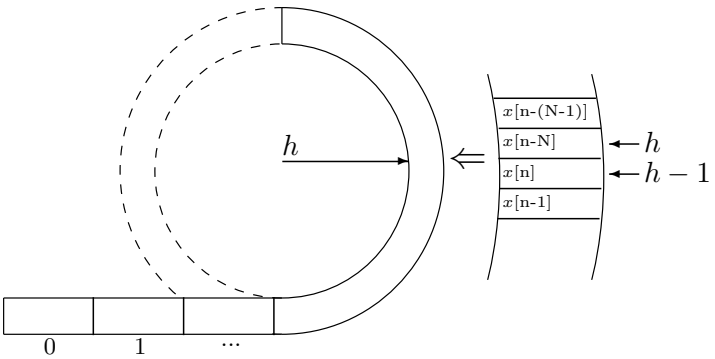
Effektiv realtidsimplementation kräver därför användning av åtminstone tre buffertar. Den första bufferten används för att lagra inkommande sampels, den andra är en arbetsbuffert och den tredje en buffert för utgående sampels [4].

Inkommande sampel från A/D-omvandlaren överförs via McBSP till inbufferten av EDMA-modulen, utan att stjäla datorkraft från CPU. På samma sätt används EDMA-modulen för att på samma kanal överföra filtrerade sampels ur utbufferten till D/A-omvandlaren.
På detta sätt används CPU endast för att behandla arbetsbufferten. För att filtrera ljud i stereo delas arbetsbufferten med fördel upp i två buffertar, en för att lagra data från vänster kanal och den andra för höger.

Efter att tidsfristen har runnit ut och alla sampel i arbetsbuffertarna är färdigbehandlade roteras buffertarna, det vill säga innehållet i arbetsbufferten förs över till utbufferten medan innehållet i inbufferten överförs till arbetsbufferten. Utbufferten används då som ny inbuffert vilket innebär att dess data kommer att skrivas över.
Att ersätta innehållet i en lång buffert kan vara en kostsam operation. Lösningen är att deklarera fyra stycken pekare, tre stycken som pekar på det första elementet i var och en av buffertarna och en tredje mellanlagringspekare. Rotationen av buffertarna sker då istället genom att rotera pekarna, alltså endast fyra operationer.

2.4.2 Cirkulär bufferthantering

Den cirkulära bufferten är vanligt förekommande inom signalbehandling.
En cirkulär buffert har en konstant längd, L, vars sampel uppdateras kontinuerligt. En pekare används för att hålla reda på det aktuella värdet i bufferten. För varje skrivet sampel ökas pekarens värde tills dess att den når buffertens ände. Pekaren kommer då att hoppa till buffertens början och skriva över de sampel som ligger sparade där. Det är alltså möjligt att hämta data L sampel tillbaka i tid.
Det krävs viss logik för att räkna ut när pekaren nått buffertens, men fördelen är att endast en minnesplats behöver ändras varje gång ett nytt sampel ska skrivas.



Figur 2.6: Principskiss över en cirkulär buffert. $x[n]$ betecknar den sparade signalen och N det totala antalet platser i bufferten.

I skissen ovan är h en pekaren till nästa element som ska skrivas över och $h-1$ på senast skrivna sampel. Då ett nytt värde ska skrivas till bufferten, uppdateras elementet som h pekar på och h stegas upp.

3 Tidsinvarianta musikeffekter

En tidsinvariant musikeffekt är ett system som inte förändras med tiden. Vilket innebär att om insignalen $x[n]$ ger upphov till utsignalen $y[n]$ så kommer en tidskiftad insignal, $x[n-L]$, ge upphov till en tidsskiftad utsignal, $y[n-L]$.
Flera av de musikeffekter som förekommer i denna rapport uppfyller detta villkor och presenteras nedan.

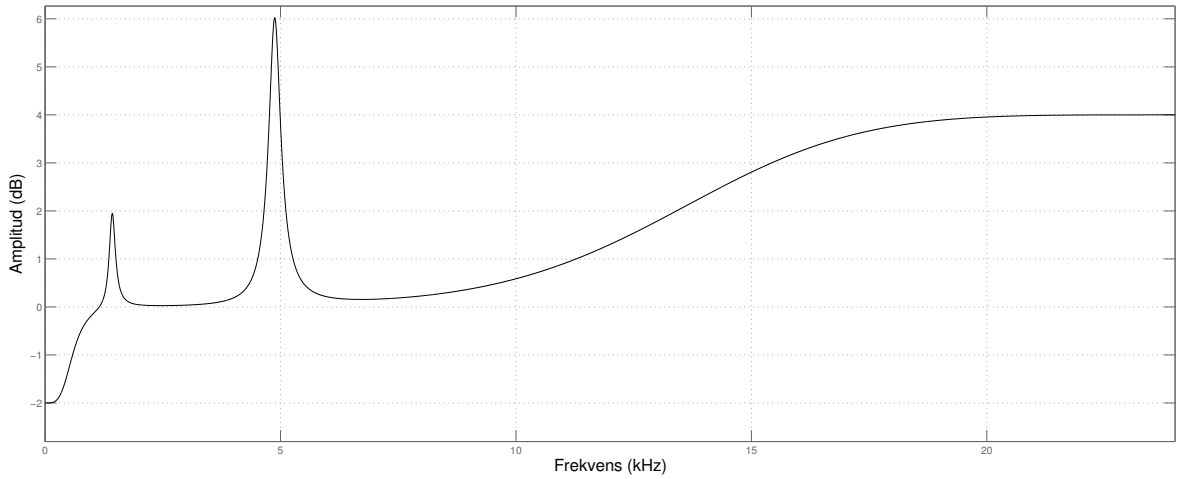
3.1 Equalizer

En equalizer (EQ) är i musiksammanhang ett verktyg som används för att justera frekvenskompositionen i en ljudsignal. Frekvensselektiva filter används för att dämpa eller framhäva specificerade frekvenser i ett ljudspår.
Equalizern tillämpas också ofta tillsammans med andra effekter, där den används för att dela upp ljudets frekvensinnehåll för att utvalda band skall kunna behandlas separat.

I en vanlig förstärkare framhävs olika frekvenser genom att ljudet filteras genom ett lågpass-, bandpass- och högpassfilter. Förstärkningen på dessa filter styrs genom att lyssnaren ställer in parametrarna bas, mellanregister och diskant.
I mer avancerade ljudsystem där användaren kräver större kontroll över frekvenskompositionen förekommer verktyget vanligtvis i ett av två utföranden, *parametrisk* eller *grafisk* equalizer.

3.1.1 Parametrisk equalizer

I en parametrisk equalizer viktas ljudets frekvensinnehåll genom att användaren specificerar parametrarna mittfrekvens (f_0), förstärkning (G) och godhetstal (Q-värde) för de partier som skall framhävas eller dämpas. Vid var och en av de specificerade mittfrekvenserna centreras ett frekvensselektivt filter.
Dessutom kan ljudets högsta och lägsta frekvensinnehåll styras genom att specifera en brytfrekvens (f_c) och förstärkning. På så sätt kan användaren väldigt noggrant styra hur systemet förstärker ljudets olika frekvenser. En variant av effekten är den *semiparametriska* equalizern där mittfrekvenserna är förspecificerade och användaren endast kan styra G och Q.
Ett exempel på hur amplitudfunktionen för en parametrisk equalizer kan se ut ges av figur 3.1.



Figur 3.1: Amplitudfunktion för en parametrisk equalizer.

Q-värdet är en enhetslös parameter som beskriver ett filtrers branthet i termer av bandbredd och mittfrekvens enligt

$$Q = \frac{f_0}{f_{ö} - f_u} = \frac{f_0}{f_b}. \tag{3.1}$$

Där $f_{ö}$ och f_u betecknar övre och undre gränsfrekvens och f_b betecknar bandbredden. Ett högt Q-värde innebär alltså ett brantare filter (smalare bandbredd) och vice versa.

Av de frekvensselektiva filter som skall användas i denna tillämpning ställs stora krav på lågt rippel i både pass- och stoppband. Detta innebär att systemet utgörs av butterworthfilter. Överföringsfunktionen för ett andra ordningens lågpass- respektive högpass-butterworthfilter i kontinuerlig tid ges av ekvation 3.2 och 3.3 respektive.

$$H_{LP}(s) = \frac{\omega_c^2}{s^2 + \sqrt{2}\omega_c s + \omega_c^2} \tag{3.2}$$

$$H_{HP}(s) = \frac{s^2}{s^2 + \sqrt{2}\omega_c s + \omega_c^2} \tag{3.3}$$

där $\omega_c = 2\pi f_c$ betecknar brytfrekvensen för respektive filter.
Dessa båda filter dämpar alla frekvenser utanför brytfrekvenserna vilket är den exakta motsatsen mot vad som efterfrågas i den parametriska equalizern. Ett *shelffilter* däremot, är ett filter som idealt endast påverkar frekvensinnehållet inom brytfrekvensen och lämnar resten av spektrumet opåverkat [10]. Ett sådant filter med lågpassverkan åstadkoms genom att insignalen adderas till ett viktat lågpassfilter och beskrivs av överföringsfunktionen

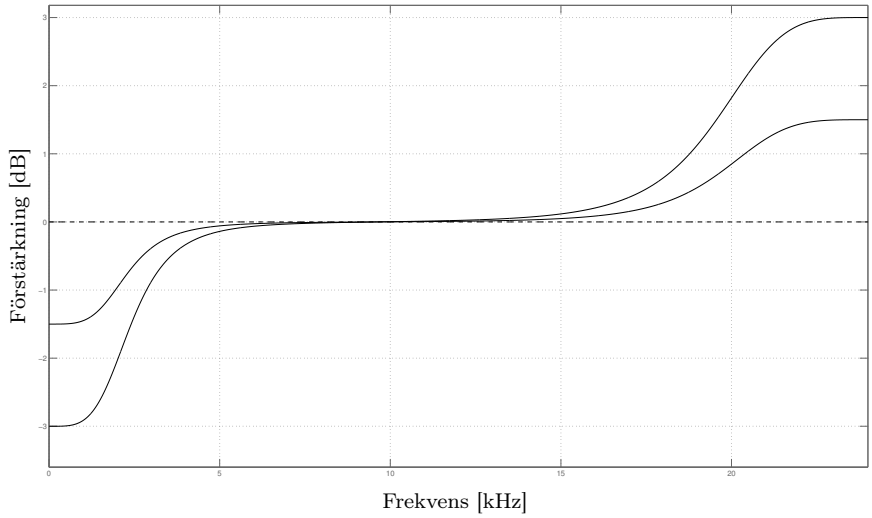
$$H_{S,L}(s) = 1 + G \cdot H_{LP}(s) = \begin{cases} \frac{(\frac{s}{\omega_c})^2 + \sqrt{2(G+1)}\frac{s}{\omega_c} + G + 1}{(\frac{s}{\omega_c})^2 + \sqrt{2}\frac{s}{\omega_c} + 1}, & G > 0 \\ \\ \frac{(\frac{s}{\omega_c})^2 + \sqrt{2}\frac{s}{\omega_c} + 1}{(\frac{s}{\omega_c})^2 + \sqrt{2(G+1)}\frac{s}{\omega_c} + G + 1}, & G < 0 \end{cases} \tag{3.4}$$

Överföringsfunktionen skiljer sig alltså då filtret har förstärkande respektive dämpande verkan där de båda ekvationerna är varandras invers.

På motsvarande sätt kan det högfrekventa innehållet viktas genom filtrering med ett shelffilter med högfrekvensverkan vars överföringsfunktion ges av

$$H_{S,H}(s) = 1 + G \cdot H_{HP}(s) = \begin{cases} \frac{(G+1)(\frac{s}{\omega_c})^2 + \sqrt{2(G+1)}\frac{s}{\omega_c} + 1}{(\frac{s}{\omega_c})^2 + \sqrt{2}\frac{s}{\omega_c} + 1}, & G > 0 \\ \\ \frac{(\frac{s}{\omega_c})^2 + \sqrt{2}\frac{s}{\omega_c} + 1}{(G+1)(\frac{s}{\omega_c})^2 + \sqrt{2(G+1)}\frac{s}{\omega_c} + 1}, & G < 0 \end{cases} \tag{3.5}$$

Även här måste överföringsfunktionen inverteras då användaren går från att dämpa till att förstärka det högfrekventa ljudet. Hög- och lågpassverkande shelffilter med varierande förstärkning visas i figur 3.2.



Figur 3.2: Högpass- och lågpass-shelffilter med varierande förstärkning.

Överföringsfunktionen för ett bandpassfilter ges genom att utgå från ett lågpassfiltret och utföra filtertransformen:

$$s \to \frac{s^2 + \omega_ö \omega_u}{s(\omega_ö - \omega_u)} \omega_c \quad [11] \tag{3.6}$$

Detta ger för butterworthfiltret i ekvation 3.2:

$$H_{BP}(s) = \frac{(\omega_ö - \omega_u)s}{s^2 + (\omega_ö - \omega_u)s + (\omega_ö - \omega_u)^2 Q^2} \tag{3.7}$$

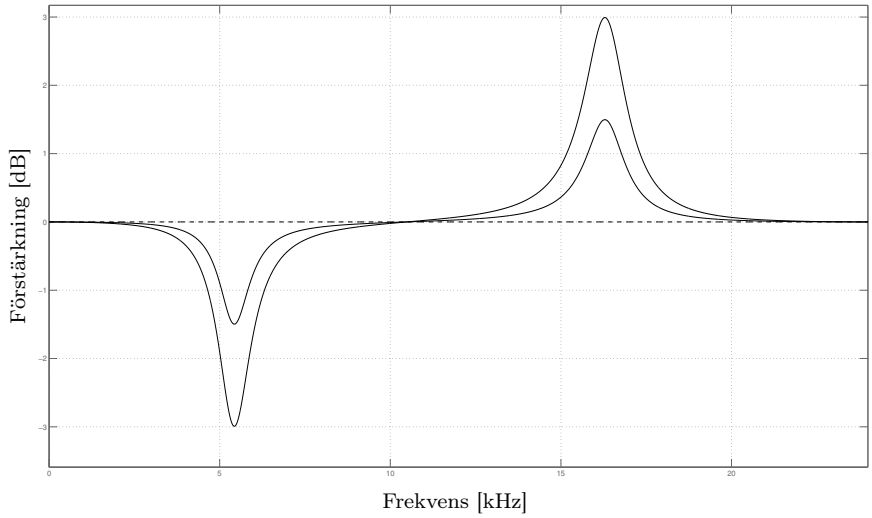
som genom att uttrycka övre och undre gränsfrekvens i termer av mittfrekvens och Q-värde enligt ekvation 3.1 kan skrivas som

$$H_{BP}(s) = \frac{\frac{\omega_0}{Q} s}{s^2 + \frac{\omega_0}{Q} s + \omega_0^2}. \tag{3.8}$$

där $\omega_0 = 2\pi f_0$ Alltså kan bandpassfiltrets överföringsfunktion bestämmas av parametrarna f_0 och Q. Om insignalen framkopplas till utsignalen från ett sådant bandpassfilter, viktat med förstärkningen G, på samma sätt som för shelffiltret ovan ges överföringsfunktionen för ett *peakfilter*:

$$H_P(s) = 1 + G \cdot H_{BP}(s) = \frac{s^2 + \frac{G+1}{Q} s + \omega_0^2}{s^2 + \frac{1}{Q} s + \omega_0^2}, \quad G > 0. \tag{3.9}$$

På samma sätt som för shelffiltret ges överföringsfunktion för ett peakfilter då $G < 0$ genom att byta plats på poler och nollställen i ekvation 3.9. Amplitudfunktionen för dämpande och förstärkande peakfilter illustreras i figur 3.3.



Figur 3.3: Peakfilter med varierande förstärkning och mittfrekvens men konstant Q-värde.

Notera vikten i att filtren ovan utgick från butterworthfiltret. Hade det i ett av filtren förekommit rippel i passbandet hade detta i det förstärkande fallet orsakat störningar i den framhävda delen av ljudet. Hade det också funnits rippel i stoppbandet hade detta påverkat ljudets resterande frekvensinnehåll.

De analoga shelf- och peakfiltren beskrivna i ekvation 3.4, 3.5 och 3.9 bygger upp den parametriska equalizern. För att dessa skall kunna behandlas i en DSP måste de först diskretiseras. Sambandet mellan z-transform och laplacetransform ges av

$$z = e^{sT_s} = \frac{e^{\frac{1}{2}sT_S}}{e^{-\frac{1}{2}sT_S}}, \tag{3.10}$$

vilket genom att utföra en första ordningens taylorutveckling kan approximeras som

$$z \approx \frac{1 + \frac{sT_S}{2}}{1 - \frac{sT_S}{2}} \;\Rightarrow\; s \approx \frac{2}{T_S}\frac{z-1}{z+1} \tag{3.11}$$

Approximationen ovan kallas för bilinjär transform och ger stabila diskreta filter för de stabila analoga filtren ovan. Genom att ersätta $s = j\omega$ och $z = e^{j\Omega}$ där $\Omega = 2\pi f/f_S$ ges sambandet mellan det analoga och diskreta filtrets frekvenser:

$$j\omega \approx \frac{2}{T_S}\frac{e^{j\Omega} - 1}{e^{j\Omega} + 1} \tag{3.12}$$

vilket kan förenklas till

$$\omega \approx \tan(\frac{\Omega}{2}) \tag{3.13}$$

Genom att utföra substitutionerna i ekvation 3.11 och 3.13 i uttrycken för shelf- och peakfiltren kan dessa filters diskreta motsvarighet beräknas. Överföringsfunktionen för de båda filtren kan då skrivas på formen:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}} = \frac{\sum_{i=0}^{2} b_i z^{-i}}{\sum_{j=0}^{2} a_j z^{-j}} \tag{3.14}$$

Uttrycket i 3.14 överensstämmer med den allmänna överföringsfunktionen för ett IIR-filter angiven i avsnitt 2.3.2. Detta innebär att equalizern implementeras som kaskadkopplade steg på

direkt form. Detta för att minska de kvantiseringseffekter som uppkommer vid fixtalsimplementaton. Innan equalizern tillämpas måste nämnarkoefficienternas numeriska värde för varje filter undersökas. Om de är mindre än ett kan steget implementeras på direkt form II annars måste de skalas och implementeras på direkt form I. Samtliga nämnar- och täljarkoefficienter för shelffiltret och en given samplingsfrekvens kan bestämmas ur parametrarna f_c och G. Uttrycken för att beräkna de olika koefficienterna för täljarpolynomen ges av tabell 3.1. I tabellerna har variabeln K införts som

$$K = \tan(\pi \frac{f_c}{f_s}). \tag{3.15}$$

Tabell 3.1: Koefficienterna till de olika shelffiltrens täljarpolynom.

Shelffilter	b_0	b_1	b_2
Lågpass G>0	$\frac{1+\sqrt{2(G+1)}K+(G+1)K^2}{1+\sqrt{2}K+K^2}$	$\frac{2((G+1)K^2-1)}{1+\sqrt{2}K+K^2}$	$\frac{1-\sqrt{2(G+1)}K+(G+1)K^2}{1+\sqrt{2}K+K^2}$
Lågpass G<0	$\frac{1+\sqrt{2}K+K^2}{1+\sqrt{2(G+1)}K+(G+1)K^2}$	$\frac{2((G+1)K^2-1)}{1-\sqrt{2(G+1)}K+(G+1)K^2}$	$\frac{1-\sqrt{2}K+K^2}{1+\sqrt{2(G+1)}K+(G+1)K^2}$
Högpass G>0	$\frac{(G+1)+\sqrt{2(G+1)}K+K^2}{1+\sqrt{2}K+K^2}$	$\frac{2(K^2-(G+1))}{1+\sqrt{2}K+K^2}$	$\frac{(G+1)-\sqrt{2(G+1)}K+K^2}{1+\sqrt{2}K+K^2}$
Högpass G<0	$\frac{1+\sqrt{2}K+K^2}{(G+1)+\sqrt{2(G+1)}K+K^2}$	$\frac{2(K^2-1)}{(G+1)+\sqrt{2(G+1)}K+K^2}$	$\frac{1-\sqrt{2}K+K^2}{(G+1)+\sqrt{2(G+1)}K+K^2}$

Koefficienterna för nämnarpolynomet ges av de liknande uttrycken sammanställda i tabell 3.2 nedan.

Tabell 3.2: Koefficienterna till de olika shelffiltrens nämnarpolynom.

Shelffilter	a_0	a_1	a_2
Låpass G>0	1	$\frac{2(K^2-1)}{1+\sqrt{2}K+K^2}$	$\frac{1-\sqrt{2}K+K^2}{1+\sqrt{2}K+K^2}$
Lågpass G<0	1	$\frac{2((G+1)K^2-1)}{1+\sqrt{2(G+1)}K+(G+1)K^2}$	$\frac{1-\sqrt{2(G+1)}K+(G+1)K^2}{1+\sqrt{2(G+1)}K+(G+1)K^2}$
Högpass G>0	1	$\frac{2(K^2-1)}{1+\sqrt{2}K+K^2}$	$\frac{1-\sqrt{2}K+K^2}{1+\sqrt{2}K+K^2}$
Högpass G<0	1	$\frac{2(\frac{K^2}{(G+1)}-1)}{1+\sqrt{\frac{2}{(G+1)}}K+\frac{K^2}{(G+1)}}$	$\frac{1-\sqrt{\frac{2}{(G+1)}}K+\frac{K^2}{(G+1)}}{1+\sqrt{\frac{2}{(G+1)}}K+\frac{K^2}{(G+1)}}$

Även de diskreta peakfiltren kan beskrivas av uttrycket i ekvation 3.14. Koefficienterna beror i detta fallet på parametrarna f_0, Q och G vilka är de parametrar som eftersöktes i avsnittets början. Uttrycken för de olika koefficienterna har sammanställts i tabell 3.3 där $a_0 = 1$ för samtliga filter och där K nu ges av

$$K = \tan(\pi \frac{f_0}{f_s}). \tag{3.16}$$

Tabell 3.3: Filterkoefficienter för de båda peakfiltren.

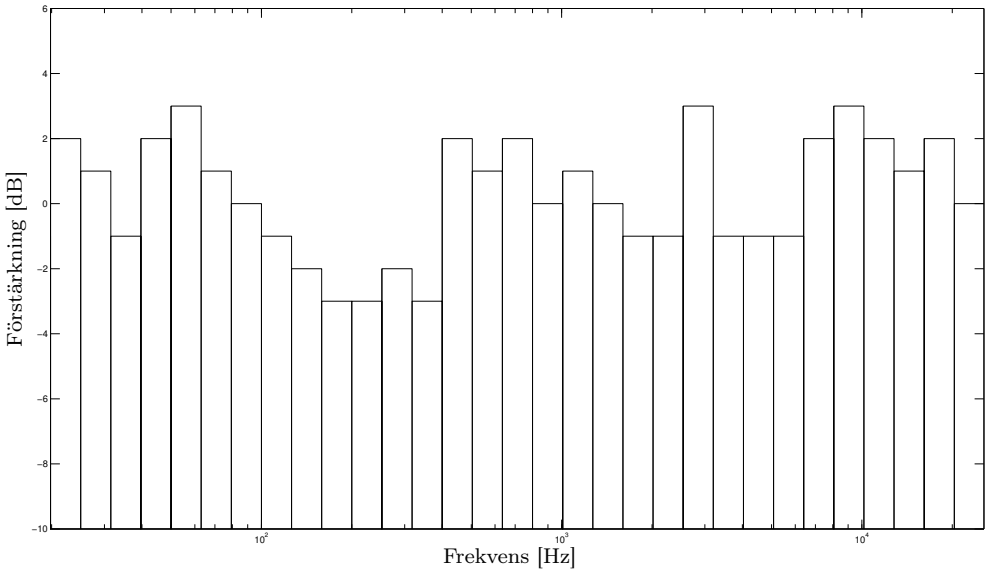
Peakfilter	b_0	b_1	b_2	a_1	a_2
G>0	$\frac{1+\frac{(G+1)}{Q}K+K^2}{1+\frac{1}{Q}K+K^2}$	$\frac{2(K^2-1)}{1+\frac{1}{Q}K+K^2}$	$\frac{1-\frac{(G+1)}{Q}K+K^2}{1+\frac{1}{Q}K+K^2}$	$\frac{2(K^2-1)}{1+\frac{1}{Q}K+K^2}$	$\frac{1-\frac{1}{Q}K+K^2}{1+\frac{1}{Q}K+K^2}$
G<0	$\frac{1+\frac{1}{Q}K+K^2}{1+\frac{(G+1)}{Q}K+K^2}$	$\frac{2(K^2-1)}{1+\frac{(G+1)}{Q}K+K^2}$	$\frac{1-\frac{1}{Q}K+K^2}{1+\frac{(G+1)}{Q}K+K^2}$	$\frac{2(K^2-1)}{1+\frac{(G+1)}{Q}K+K^2}$	$\frac{1-\frac{(G+1)}{Q}K+K^2}{1+\frac{(G+1)}{Q}K+K^2}$

Den parametriska equalizern kan alltså implementeras som en kombination av flera diskreta shelf- och peakfilter. Användaren specificerar parametrarna f_c och G för de båda shelffiltren samt f_0, Q och G för samtliga peakfilter. Filtrens koefficienter kan då beräknas av uttrycken givna i tabellerna 3.1, 3.2 och 3.3 ovan.
Eftersom den parametriska equalizern enligt avsnitt 2.3.1 endast får förstärka signalen 9 dB, måste systemet för att föhindra överskridning, begränsa användarens möjlighet att förstärka signalen ytterligare.

Implementationen av equalizern med IIR-filter av låg ordning innebär att effekten inte kräver särskilt mycket beräkningskraft. I de flesta scenarion kommer antalet filter aldrig att överskrida tio, vilket ger en processorbelastning på 2.8%. Dock kommer belastningen temporärt att öka då användaren ändrar en av de styrbara parametrarna, eftersom ett filter då behöver räknas om. Trots detta är det ingen risk att den parametriska equalizern kommer överskrida den i inledningen statuerade kravspecifikationen.

3.1.2 Grafisk equalizer

En grafisk equalizer utgörs av en mängd bandpassfilter där användaren endast har möjlighet att styra förstärkningen för varje filter. Mittfrekvens och Q-värde för motsvarande filter är specifierade på ett sätt så att varje band täcker olika frekvenser.
Genom att dela ljudets frekvensinnehåll i flera olika band, som tillsammans täcker hela spektrumet, kan signalens frekvenskomposition styras, så gott som, var band för sig [12]. Figur 3.4 illustrerar en ideal grafisk equalizer som utgörs av 31 band, fördelade över frekvensspektrumet.



Figur 3.4: Grafisk equalizer med 31 band.

En problematik i den grafiska equalizern är att banden tenderar att överlappa varandra vilket kan orsaka icke önskvärd förstärkning i gränszonerna. För att minska överlappningen krävs att filtrets ordningstal ökas.
Eftersom IIR-filter av hög ordning enligt avsnitt 2.3.2 är förknippade med kvantiseringsbrus kommer den grafiska equalizern att utgöras av FIR-filter.

Linjär faltning av längre sekvenser i tidsdomän kan beräkningsmässigt vara en väldigt kostsam operation. Antalet operationer kan dock minskas avsevärt om filtrering sker i frekvensdomän och sekvenserna fouriertransformeras med FFT-algoritmen, som är en metod för att beräkna den diskreta fouriertransformen, och som beskrivs i appendix A.

DFT av en diskret signal, $x[n]$, ges av

$$X(k) = \sum_{n=0}^{N-1} x[n] W_N^{nk}, \quad k = 0, 1, ..., N-1 \tag{3.17}$$

Där W betecknar fasvektorn som ges av

$$W_N = e^{-j\frac{2\pi}{N}}. \tag{3.18}$$

Beräkning av uttrycket ovan resulterar i N^2 komplexa multiplikationer och $N^2 - N$ komplexa additioner vilket för stora N innebär en mycket kostsam operation.
För att minska antalet beräkningar utnyttjar FFT-algoritmen att fasvektorn är symmetrisk och

periodisk, transformeringen kräver därför endast $\frac{N}{2}\log N$ komplexa multiplikationer och $N\log N$ komplexa additoner [13].

Linjär faltning av två signaler i tid motsvaras av multiplikation i frekvens om dessa har oändlig utsträckning. Om signalerna däremot är av begränsad längd motsvaras multiplikationen inte av linjär utan av cirkulär faltning som defineras

$$h[n] \circledast x[n] \triangleq \sum_{k=0}^{N-1} h[k]x[(n-k)mod(N)], \quad n = 0, 1, ..., N-1. \tag{3.19}$$

där $x[n]$ och $h[n]$ är två ändliga tidsdiskreta signaler av längd N [14]. Sambandet mellan cirkulär faltining och diskret fouriertranform ges av

$$DFT(h[n] \circledast x[n]) = H[k]X[k] \tag{3.20}$$

och

$$IDFT(H[k]X[k]) = h[n] \circledast x[n]. \tag{3.21}$$

Multiplikation i frekvensdomän motsvaras alltså inte nödvändigtvis av linjär faltning i tid. För att detta skall gälla, ställs krav på längden av den diskreta fouriertransformen. Detta framgår av följande resonemang:

Antag att en begränsad insignal, $x[n]$, av längd L skall filtreras i ett system, $h[n]$, av längd M. Utsignalen ges då som den linjära faltningen av $x[n]$ och $h[n]$:

$$y[n] = h[n] * x[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k] \tag{3.22}$$

där produkten mellan $x[n]$ och $h[n-k]$ är nollskild då $0 \leq n \leq L+M-1$. Motvarande uttryck i frekvensdomän ges av

$$Y(e^{j\omega}) = H(e^{j\omega})X(e^{j\omega}). \tag{3.23}$$

DFT av utsignalen, $y[n]$, kan betraktas som en N-punkters sampling runt enhetscirkeln i frekvensplanet enligt

$$Y[k] = Y(e^{j2\pi k/N}) = H(e^{j2\pi k/N})X(e^{j2\pi k/N}) = H[k]X[k]. \tag{3.24}$$

För att undvika vikning i tidsplanet måste alltså längden av den diskreta fouriertransformen åtminstone vara av längd $N = L+M-1$. Om detta villkor är uppfyllt kan utsignalen återskapas fulltsändigt och ges således av

$$y[n] = IDFT(Y[k]) = IDFT(H[k]X[k]), \quad 0 \leq n \leq N-1. \tag{3.25}$$

Eller med andra ord

$$y[n] = h[n] * x[n] = h[n] \circledast x[n]. \tag{3.26}$$

Alltså är resultatet av den cirkulära faltningen i ekvation 3.19 identiskt med den linjära i ekvation 3.22 då kravet $N \geq L+M-1$ är uppfyllt [12].

Insignalen i en realtidsimplementation kan inte anses vara av begränsad längd till skillnad från datamängden som per tidsenhet kan genomgå filtrering i en signalprocessor.
Behandling av längre sekvenser måste därför delas upp i block och hanteras separat. Om filtrering då sker med linjära filter kan dessa block filtreras en i taget för att sedan adderas ihop och forma utsignalen.

En vanlig metod för att genom FIR-filtrering behandla block av data är OAM (Overlap-Add Method), där systemet i realtid matas med sampel av en insignal $x[n]$ som delas upp i block av längd L så att

$$x[n] = \sum_{r=-\infty}^{\infty} x_r[n - rL] \tag{3.27}$$

där

$$x_r[n] = x[n], \quad n = 0, ..., L-1. \tag{3.28}$$

Om denna insignal filteras med ett linjärt FIR-filter av ordning M med impulssvaret $h[n]$ ges utsignalen från varje block av

$$y_r[n] = h[n] * x_r[n]. \tag{3.29}$$

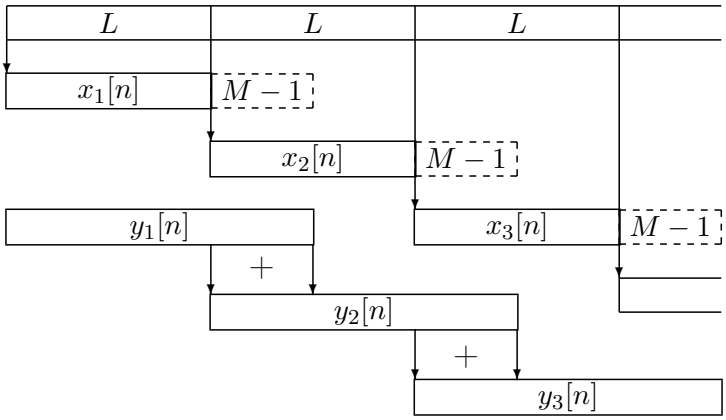
Eftersom systemet är linjärt och tidsinvariant kan sekvenserna adderas samman och forma den totala utsignalen

$$y[n] = h[n] * x[n] = \sum_{r=-\infty}^{\infty} h[n] * x_r[n - rL], \quad n = 0, 1, ..., N-1 \tag{3.30}$$

vilket enligt ekvation 3.25 också kan skrivas som

$$y[n] = \sum_{r=-\infty}^{\infty} IDFT(H[k]X_r[k]), \quad n = 0, 1, ..., N-1. \tag{3.31}$$

Eftersom utsignalen är av längd N måste de kortare sekvenserna $H[k]$ och $X_r[k]$ också utökas till längd N. Detta åstadkomms genom att addera en sekvens av nollor efter signalerna [9].
De filtrerade blocken är av längd N medan utgående block skall vara av samma längd som insignalen, L. De $M-1$ sista elementen i varje block adderas därför istället till nästa block (överlappas) för att forma den totala utsignalen. Det är denna överlappning som gett metoden dess namn, processen illustreras av figur 3.5.



Figur 3.5: Blockbaserad filtrering med OAM. Insignalen nollutökas med nollor av längd $M-1$. Efter filtrering adderas de förlänga blocken samman för att bilda utsignalen.

Resultatet av en DFT/FFT är komplext vilket innebär att operationen

$$Y[k] = H[k]X[k] \tag{3.32}$$

för varje k måste utföras som

$$Re(Y[k]) = Re(H[k])Re(X[k]) - Im(H[k])Im(X[k]) \tag{3.33}$$

och

$$Im(Y[k]) = Re(H[k])Im(X[k]) + Re(H[k])Im(X[k]). \tag{3.34}$$

För att utföra DFT/FFT av en N punkters signal krävs alltså två buffertar av längd $2N$ för att både real- och imaginärdelen för filtret och signalen skall kunna sparas. I FFT-algoritmen är innehållet i fasvektorn statiskt. Om dessa komplexa tal kan tilldelas en buffert av längd $2N$ behöver fasvektorn endast beräknas vid systemets uppstart för att sedan återanvändas.

Samma fasvektorer kan även användas för att utföra IDFT med FFT-algoritmen eftersom

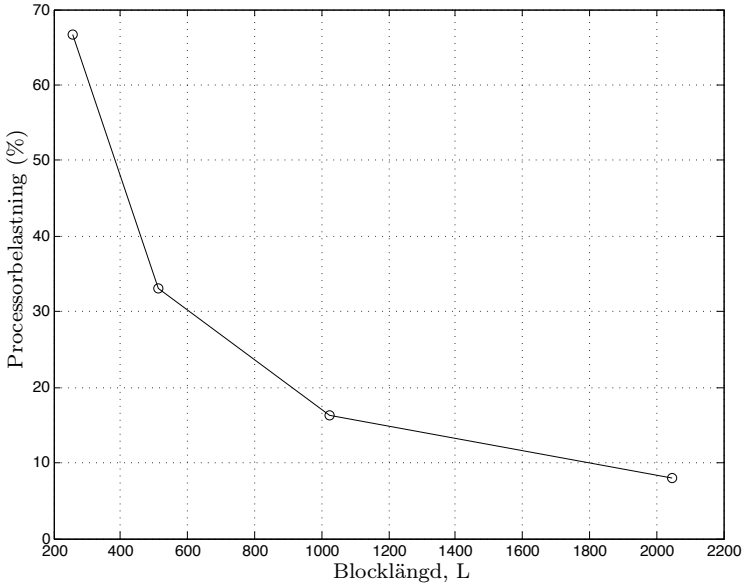
$$y[n] = \frac{1}{N}\sum_{k=0}^{N-1} Y[k]W_N^{-nk} \tag{3.35}$$

även kan skrivas som

$$y[n] = \frac{1}{N}\sum_{k=0}^{N-1} Y[k](W_N^{nk})^* = y[n] = \frac{1}{N}\sum_{k=0}^{N-1} (Y[k]^* W_N^{nk})^*. \tag{3.36}$$

Genom att först komplexkonjugera $Y[k]$, utföra algoritmen och därefter komplexkonjugera resultatet kan alltså samma algoritm och fasvektor användas om utsignalen skalas med N^{-1}.

Enligt kravspecifikationen i avsnitt 1.4 får processorns belastning per musikeffekt inte överstiga 33.3%, dessutom får systemet inte fördröja signalen mer än 20 ms. Dessa båda faktorer begränsar equalizerns uppbyggnad.
Dock finns det ytterligare en begränsning i systemet, nämligen det begränsade lagringsutrymmet i processorns internminne. För att effektivt kunna utföra OAM med FFT-algoritmen då $N = 2^{15} = 16384$ krävs cirka 650 kB lagringsutrymme för att rymma fasvektorerna, filterkoefficienterna och diverse arbetsbuffertar. Ökas N med en faktor två krävs då cirka 1.3 MB minne, vilket överskrider de 1 MB som finns tillgängligt. Processorns belastning vid $N = 16384$ för olika L visas i figur 3.6.



Figur 3.6: Processorbelastning

Genom att välja $L = 512$ ges en belastning av CPU på 33.1% vilket precis uppfyller kravspecifikationen. Att 512 sampel behandlas per block innebär att fördröjningen i systemet kan uppskattas till

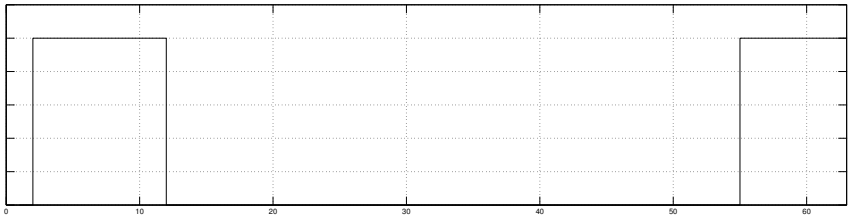
$$T_L \approx L \cdot T_S \approx 10.7\ ms \tag{3.37}$$

vilket väl uppfyller kravspecifikationen. Med N och L bestämda kan M väljas genom villkoret för linjär faltning till

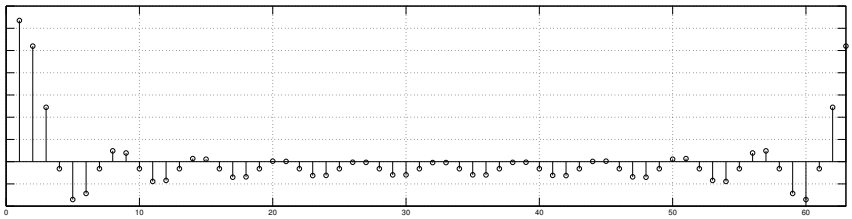
$$M = N - L + 1 = 16384 - 512 + 1 = 15873 \tag{3.38}$$

Observera att M inte en multipel 2 vilket innebär att FFT-algoritmen inte kan nyttjas för att beräkna banden. Eftersom filtren endast beräknas vid systemets uppstart är detta dock inget större problem.

Med M speciferat kan nu de olika filtren beräknas. Varje enskilt band kommer att beräknas genom att utgå från ett idealt filter med ändligt frekvenssvar. Ett sådant filter har ett oändligt impulssvar, $h_{ideal}[n]$. För att kunna implementera filtret måste det alltså fönstras.
En fönsterfunktion är i princip en funktion som är noll utanför ett visst intervall och en multiplikation med en sådan funktion motsvarar en trunkering i tidsplanet vilket kommer att uttrycka sig som rippel och roll-off i frekvensdomänen. I figur 3.7 visas ett idealt lågpassfilter vars impulssvar, $h_{trunk}[n]$, i figur 3.8 har trunkerats med ett rektangulärt fönster.

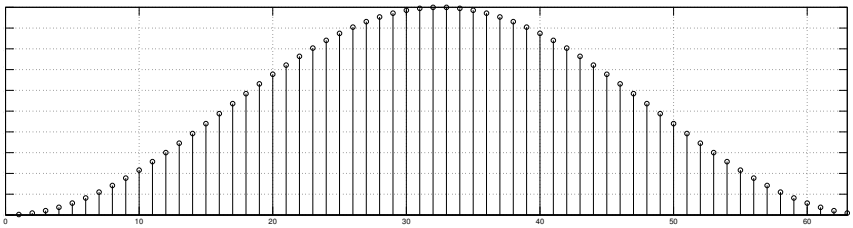


Figur 3.7: Idealt lågpassfilter i frekvensdomän.



Figur 3.8: Trunkerat impulssvar, $M = 64$.

Istället för att trunkera ett filter tvärt med ett rektangulärt fönster, bör man istället multiplicera med ett fönster som avtar mot noll, vilket bättre bevarar frekvenssvaret [15]. Hannfönstret, se figur 3.9, är ett fönster som dämpar sidloberna med konsekvensen att huvudloben breddas något.

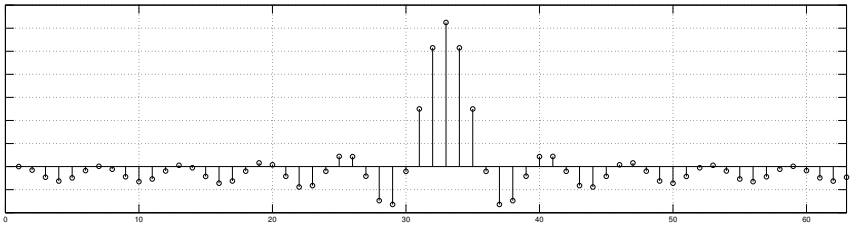


Figur 3.9: Hannfönster

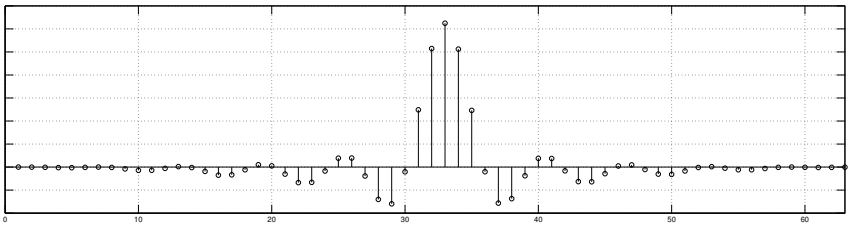
Hannfönstret kan beskrivas som

$$w[n] = \begin{cases} 0.5 - 0.5\cos\left(\frac{2\pi n}{M}\right), & 0 \leq n \leq M \\ 0, & annars. \end{cases} \tag{3.39}$$

Genom att skifta det trunkerade impulssvaret, så att det centreras mitt i fönstrets huvudlob, se figur 3.10, och multiplicera med $w[n]$ ges impulssvaret i figur 3.11. Ur figuren framgår att storleken det fönstrade impulssvarets sidlober har minskats medan huvudloben storlek knappt påverkats.

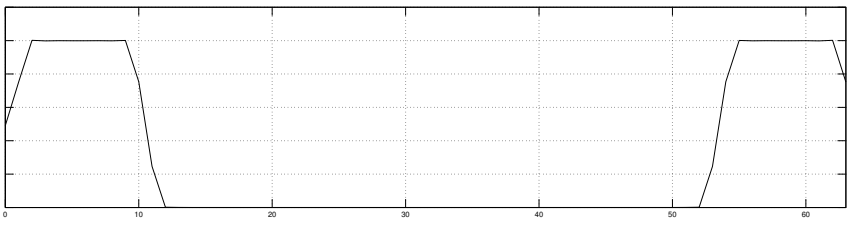


Figur 3.10: Skiftat impulssvar, $M = 64$.



Figur 3.11: Skiftat impulssvar efter fönstring, $M = 64$.

Diskret fouriertransformering av det fönstrade impulssvaret ger då amplitudfunktionen i figur 3.12.



Figur 3.12: Amplitudfunktionen för impulssvar efter fönstring, $M = 64$.

Om ett sådant filter av längd M till nollutökas till längd N och fouriertransformeras ges ett filter som kan tillämpas i OAM av

$$H[n] = DFT(w[n] \cdot IDFT(H_{ideal}[n]) + 0 + ... + 0), \tag{3.40}$$

vilket för $M = 15873$ ger ett mycket brant filter.

Därefter skall brytfrekvenserna för de olika filtren i den grafiska equalizern bestämmas.
Ett naturligt sätt placera banden längs spektrumet, se appendix B, är så att dessa täcker upp

varsin *oktav*.
Oktavernas gränsfrekvenser kan då beräknas som

$$f_{oktav}(n) = f_{ref} \cdot 2^n. \tag{3.41}$$

I professionella sammanhang förekommer ofta 31-bands equalizern där frekvensspektrumet delas upp i tredjedelsoktaver [16]. Gränsfrekvensen beräknas då enligt

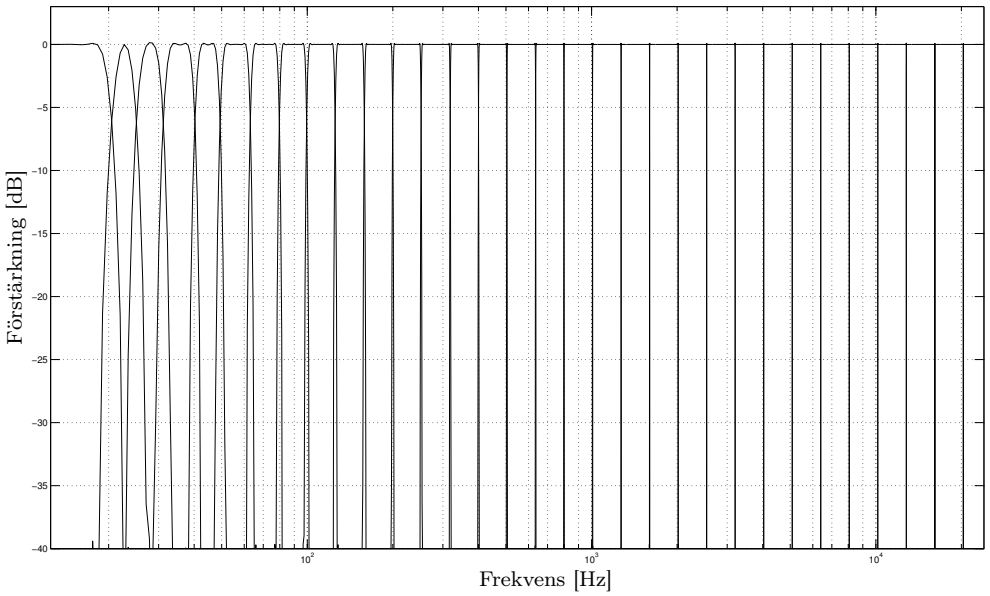
$$f_g(n) = f_{ref} \cdot 2^{\frac{n}{3}}. \tag{3.42}$$

I ett diskret system beräknas motsvarande gränsfrekvens uttryckt i sampel som

$$n_g \approx \frac{f_g}{f_s} \cdot M, \ n_g \in \mathbb{N} \tag{3.43}$$

Den grafiska equalizern beräknas genom att dela upp frekvensspektrat i 31 stycken band som täcker upp varsin tredjedelsoktav.

Gränsfrekvensen för varje filter beräknas således av ekvation 3.43, där varje filter av längd $M = 15873$ nollutökas till längd $N = 16384$ enligt ekvation 3.40.
Referensfrekvensen väljs till 12.5 Hz vilket innebär att spektrumet skulle kunna delas upp i 33 band. Dock ligger gränsfrekvenserna i de lägre såpass nära varandra att systemet inte kan skilja ett band ur ett annat. De tre nedersta banden slås därför ihop till ett.
Amplitudfunktionen för den implementerade grafiska equalizern bestående av 31 band ges av figur 3.13 illustrerad i logaritmisk skala.



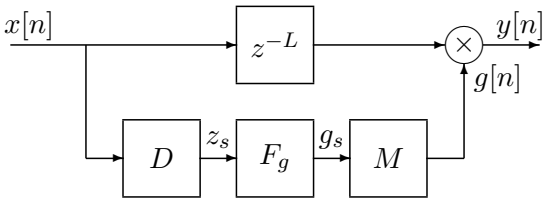
Figur 3.13: Grafisk equalizer bestående av 31 band.

Ett enkelt styrsystem har också tagits fram, där varje bands förstärkning, G, kan höjas och sänkas med ett reglage.

3.2 Kompressor

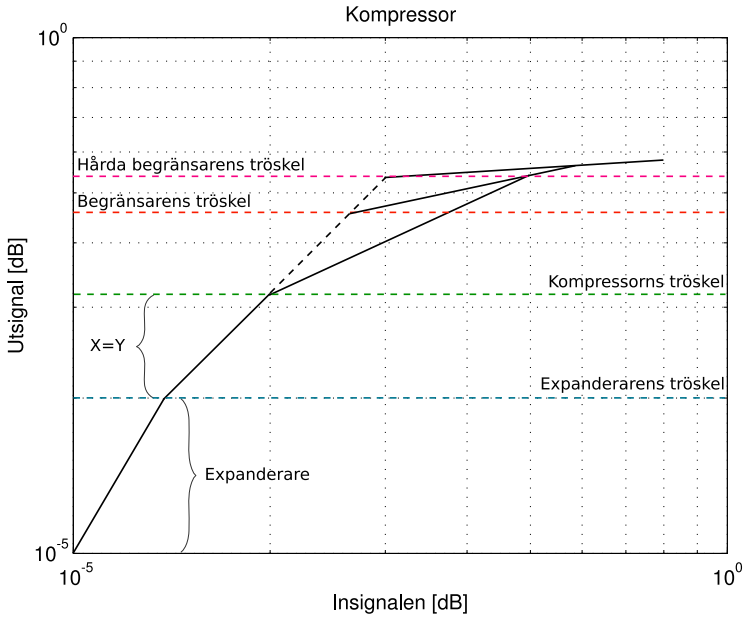
Kompressorn blev först uppmärksammad vid radions introduktion. Utrustningen som användes då var ofta känslig och kunde lätt överbelastas och samtidigt skulle signalstyrkan i sändningarna vara tillräcklig för att en lyssnare skulle kunna uppfatta innehållet. Den första, om något primitiva, lösningen var att en tekniker sänkte eller höjde volymen beroende på insignalens styrka. Eftersom det var svårt att förutsäga signalen, blev resultatet ofta av blandad kvalité. Det var först under 1920-talet, den första elektroniska kompressorn introducerades [17]. Kompressorn har ett brett användningsområde och i musiksammanhang används den ofta för att aktivt skala signalens dynamiska omfång.

Kompressorns uppbyggnad ges av figur 4.12. Där de tre viktigaste delarna är detektorn D, viktningsfunktionen F_g och det mjukgörande filtret M. Fördröjningen, L skall motsvarande den framkopplade grenens fördröjning. Detektorn används för att bestämma vilken amplitud eller effekt en signal har och hänvisar därefter med hjälp av ett styrvärde $z_s[n]$, varje sampel, till motsvarande viktningsfunktion. Viktningsfunktionen genererar i sin tur en skalande faktor $g_s[n]$.



Figur 3.14: Kompressorns uppbyggnad.

Viktningsfunktionen F_g dämpar eller förstärker en signal dynamiskt med exponentiella funktioner, se figur 3.15. Beroende på styrsignalen $z_s[n]$ så använder sig viktningsfunktionen av olika lutningar, reglerade av användaren. Det de har gemensamt, är dess exponentiella karaktär för att människan skall uppfatta dem som linjära. De olika lutningar delas in i olika intervall: expanderare, oförändrade intervallet, kompressor, begränsare och hård begränsare ordnade efter ökande lutning.



Figur 3.15: Skillnaderna mellan de olika delarna i viktningsfunktionen.

Längst ner i figur 3.15 återfinns expanderaren. I motsats till de andra funktionerna, förstärker expanderaren de lägre partierna med förhållandet R_{Exp}. Dock används den ofta med en viss återhållsamhet eftersom brusnivån ökar proportionellt mot signalens styrka.

Ovanför expanderaren anses ingen förändring vara nödvändig. Signalen passerar då igenom systemet utan att skalas.

När signalstyrkan överskrider de högre trösklarna används funktionerna kompressor, begränsare och hård begränsare. Signalen dämpas då med förhållandena R_{Komp}, R_{Beg} respektive $R_{H.Beg}$. Detta kan användas i samband med en förstärkning och på så sätt öka medeleffekten av signalen men inte nödvändigtvis amplituden. Ökningen i medeleffekt sker då på bekostnad av dynamiken [17].

Den hårda begränsaren kan ses som en försäkran att nästkommande system inte överbelastas med en allt för stark insignal. Denna är oberoende av detektorerna och kan plockas bort om så önskas av användaren.

Viktningsfunktionen beräknas som

$$F_g(z_s[n]) = C(z_s[n]) \cdot z_s[n]^{R(z_s[n])^{-1}}$$

där $C(z_s[n])$ beräknas som startvärdet för motsvarande intervall, T är trösklarnas position och $R(z_s[n])$ varierar beroende på styrsignalen enligt:

$$\begin{aligned}
R = R_{H.Beg},\ då \qquad & z_s[n] \geq\ T_{H.Beg}, \\
R = R_{Beg},\ då \qquad & T_{H.Beg} > z_s[n] \geq T_{Beg}, \\
R = R_{Komp},\ då \qquad & T_{Beg} > z_s[n] \geq T_{Komp}, \\
R = 1,\ då \qquad & T_{Komp} > z_s[n] \geq T_{R=1}, \\
1 = R_{Exp},\ då \qquad & T_{Exp} > z_s[n]
\end{aligned}$$

Användaren specificerar lutningarna R för de olika intervallen så att:

$$R_{Exp} < 1 < R_{Komp} \leq R_{Beg} \leq R_{H.Beg} < \infty$$

och trösklarna:

$$T_{Exp} < T_{Komp} \leq T_{Beg} \leq T_{H.Beg} \text{ [18].}$$

Den slutgiltiga skalningsfaktorn $g_s[n]$ beräknas med hjälp av viktningsfunktionen och styrsignalen

$$g_s[n] = \frac{F_g(z_s[n])}{z_s[n]}, \text{ då } z_s[n] \neq 0\,. \tag{3.44}$$

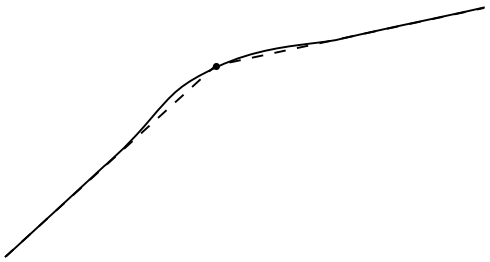
Funktionerna i viktningsfunktionen ovan, upplevs linjära för det mänskliga örat. Till följd blir effekten av kompressorns dämpning väldigt påtaglig vilket inte är uppskattat i alla sammanhang. Ett alternativt sätt att dämpa signalen är med mjuka "knän", det vill säga övergångarna mellan de olika intervallen. De mjuka knäna i kompressorn antar samma värden som de hårda vid tröskelvärdena. Skillnaden är att de hårda viktningsfunktionerna, förblir deriverbara vid trösklarna. Sådana funktioner kallas splinekurvor och för kompressorn användes en tredje ordningen splinekurva [19]. Den nya funktionen beräknades enligt följande:

$$\begin{aligned} F_g(z_s[n]) = (1 - t((z_s[n])))F_{g1}((z_s[n])) + tF_{g2}((z_s[n]))+ \\ +(t((z_s[n])) - t((z_s[n]))^2)(a(1 - t((z_s[n]))) + bt((z_s[n]))) \end{aligned} \tag{3.45}$$

där a, b och $t(z_s)$ kan beräknas som:

$$\begin{aligned} t &= \frac{z_s - z_{s1}}{z_{s2} - z_{s1}} \\ a &= k_1(z_{s2} - z_{s1}) - (F_{g2}(z_s[n]) - F_{g1}(z_s[n])) \\ b &= -k_2(z_{s2} - z_{s1}) + (F_{g2}(z_s[n]) - F_{g1}(z_s[n])) \end{aligned} \tag{3.46}$$

Notera att ekvation 3.45 endast är skild från noll då $z_{s1} \leq z_s \leq z_{s2}$ samt k_1 och k_2, är angränsande intervalls lutningar (z_{s1}, F_{g1}) och (z_{s2}, F_{g2}). Se figur 3.16 för en demonstrativ bild, av en tillämpad splinefunktion.



Figur 3.16: Splineanpassad kurva mellan två olika viktningsfunktioner.

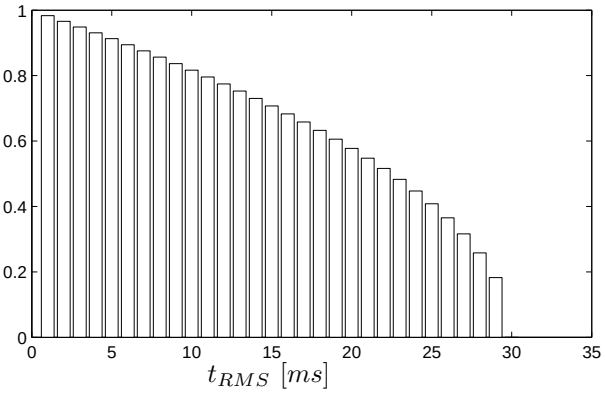
Genom att dra nytta av approximationer kan belastningen minska markant. En sådan approximation är interpolation. Normalt sett så hade viktningsfunktionerna behövts beräknas vid varje exekvering för att i sin tur beräkna den motsvarande skalningsfaktorn. Dessutom kan splinekurvor inte appliceras på exponentiella kurvor utan kräver att funktionerna logaritmeras, för att sedan återgå till sitt tidigare värde. För att undkomma detta beräknas hela spannet med ett fåtal punkter, vilka sparas undan i en buffert. Kurvan utgörs då endast av ett fåtal punkter och

sannolikheten är låg att ett värde i bufferten överensstäms med det eftersökta värdet men genom använda interpolation, kan varje värde beräknas.

Viktningsfunktionen behövs alltså bara beräknas om det har skett någon förändring av parametrarna. Vid initiering beräknas alla nödvändiga värden för de olika intervallen och därifrån kan alla värden erhållas, oavsett om mjuka eller hårda knän används. Implementationen av kompressorn använder 7.4% av processorns kapacitet när mjuka knän med tabellinterpolering används.

Som bekant används detektorn D för att bestämma vilken amplitud eller effekt en signal har och beräknar därefter ett styrvärde $z_s[n]$. Detta åstadkoms genom att beräkna RMS-värdet, jämföra denna med användarens tröskelvärden och därefter bestämma ett styrvärde [20].

Det är främst två typer av detektormodeller som används inom kommersiell apparatur, RMS- och topp-detektor. RMS-detektorn bestämmer en signals medeleffekt under ett visst tidsintervall. Med en skiftande buffert, motsvarande längden t_{RMS} samt kontinuerligt beräknande av buffertens totala RMS-värde, ges ett nytt styrvärde. Buffertlängden kommer då introducera en tröghet i systemet så att styrvärdet $z_s[n]$ fluktuerar betydligt långsammare. Detta medför att kompressionen uppfattas på ett mjukare sätt än vad det annars hade gjort, se figur 3.17.



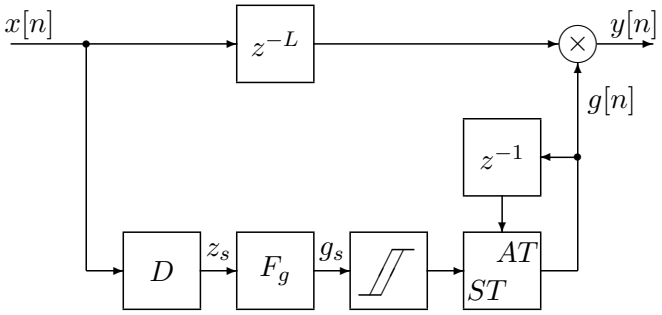
Figur 3.17: RMS-buffertens avklingande när insignalen, abrupt, går från hög till låg.

Toppdetektorn skiljer sig från RMS-detektorn eftersom den även kan efterlikna den analoga kompressorns karakteristik. Detta åstadkoms genom att, förutom att beräkna RMS-värdet för en enstaka sampel, använda attack- och släpptider. Användaren bestämmer attack- och släpptider för att reglera tiden från det att signalen lämnas orörd, till att den dämpas respektive förstärks maximalt. Detta resulterar i att snabba variationer delvis kommer att passera kompressorn okomprimerade vilket är karakteristiskt för äldre analoga kompressorer [17].

Vid implementationen av toppdetektorn används två separata skalningsfaktorer, $g[n]$ och $g_s[n]$. Där $g[n]$ är den skalningsfaktor som signalen slutligen skalas med och $g_s[n]$ är den skalningsfaktor som beräknas för varje sampel. Beroende på förhållandet mellan $g[n-1]$ och $g_s[n]$, minskas eller ökas $g[n]$ enligt ekvation 3.47:

$$\begin{aligned} g[n] &= g[n-1] - \frac{1}{f_s \cdot t_{attack}},\ 0 \leq g[n-1] < g_s[n] \text{ och } t_{attack} \neq 0 \\ g[n] &= g[n-1] + \frac{1}{f_s \cdot t_{släpp}},\ g[n-1] > g_s[n] \text{ och } t_{släpp} \neq 0 \end{aligned} \tag{3.47}$$

En kompressor med implementerad attack- och släpptid kan ses i figur 3.18.



Figur 3.18: Kompressor med attack- och släpptid.

Insignalen $x[n]$ skickas till detektorn, D, där den placeras inom motsvarande spann, exempelvis expanderan. Detta förmedlas via en styrsignal, $z_s[n]$, till viktningsfunktionen, $F_g(z_s[n])$, där skalningsfaktorn $g_s[n]$ erhålls. Vid hystereskurvan och AT-ST blocken, bestäms om $g[n-1]$ skall minska (attacktid) eller öka (släpptid). Slutligen skalas den fördröjda signalen $x[n]$ med $g[n]$.

Det finns inget som hindrar att attack- och släpptiderna används även i samband med RMS-detektorn. Dock måste bufferten vara mindre än de övriga fördröjningarna om attack- och släpptiderna skall ha någon inverkan på signalen.

Den mjukgörande funktionen M med signalen $g_s[n]$ som invärde, beräknar motsvarande skalningsfaktor $g[n]$. Stora fluktueringar kommer att dämpas genom att $g_s[n]$ att skalas om. Detta eftersom för stora amplitudvariationer ger upphov till brus och som följd, en sämre musikupplevelse.

3.3 Delay/Echo

Musikeffekterna *delay* och *echo* kan åstadkommas genom att en signal enligt ett visst mönster adderas med skalade och tidsfördröjda kopior av sig själv.
För att åstadkomma denna effekt genom diskret signalbehandling används kamfiltret, vilket finns i två utföranden, fram- och återkopplat.
Ett framkopplat kamfilter är ett FIR-filter som adderar en skalad, fördröjd version av sig själv till utsignalen. Detta beskrivs som

$$y[n] = x[n] + bx[n-L]. \tag{3.48}$$

Effekten som uppstår då en signal passerar denna typ av filter är alltså ett eko av insignalen om storleken av fördröjningen, L, är tillräcklig för att örat skall kunna särskilja de båda signalerna.

I det återkopplade kamfiltret är det istället utsignalen som fördröjs och skalas. Filtret är av IIR-typ och beskrivs i tidsdomän av

$$y[n] = x[n] + ay[n-L]. \tag{3.49}$$

Kamfiltret är en av byggstenarna i flera av de musikeffekter som senare presenteras i rapporten. I detta avsnitt presenteras hur det framkopplade kamfiltret används för att åstadkomma ett delay och hur det återkopplade används för att åstadkomma ett echo.

Ett delay kan beskrivas som ett eller flera framkopplade kamfilter där de fördröjda signalerna återkommer med en viss periodicitet [21]. Matematiskt kan detta beskrivas som

$$y[n] = b_0x[n] + b_1x[n-L] + b_2x[n-2L] + ... + b_Mx[n-ML] = \sum_{i=0}^{M} b_ix[n-iL] \tag{3.50}$$

där de fördröjda signalerna är dämpade enligt koefficienterna $b_0, b_1, ...b_M$. Dämpningen kan avta efter olika mönster, vanligt är att fördröjningarna avtar exponetiellt.

Även om de fördröjda signalerna regelrätt kan återkomma vid alla möjliga tidpunkter är det ofta önskvärt att dessa återkommer i takt med musiken.
Varje musikstycke kan förknippas med ett tempo och en takt som utgör en referens för hur snabbt det spelas. Steglängden för en takt vid ett visst tempo ges av

$$L = \left(\frac{f_s \cdot \lambda \cdot 60}{f_{tempo}} \right) \tag{3.51}$$

där f_{tempo} är tempot uttryckt i bpm, $f_s = \frac{1}{T_s}$ är samplingsfrekvensen och λ är taktarten.

För att kunna styra effekten krävs att användaren utöver att specifera takt och tempo också specificerar hur mycket de fördröjda signalerna skall höras mot originalsignalen. Därför skalas originalsignalen och de fördröjda signalerna i förhållande till varandra. Denna skalning, α, kallas ofta för effektens mix och kan varieras mellan noll och ett. Efter filtrering med de framkopplade kamfiltren skalas signalen dessutom med en faktor, β som ges av

$$\beta = \frac{1}{\sqrt{(1-\alpha)^2 + (\alpha \sum_{i=1}^{M} b_i)^2}} \tag{3.52}$$

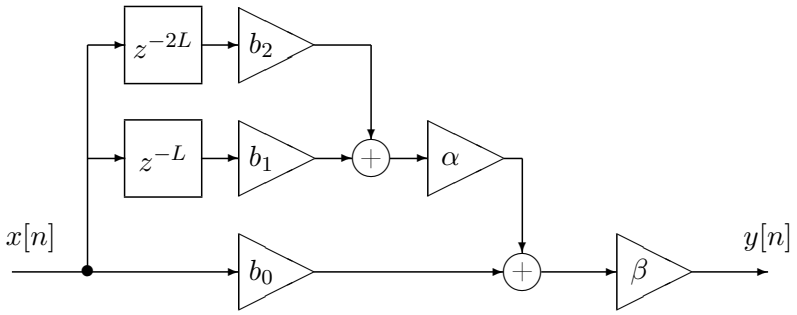
där

$$\sum_{i=1}^{M} |b_i|^2 = 1 \tag{3.53}$$

Denna skalning krävs för att signalens medeleffekt skall hållas konstant oberoende av vilken mix som speciferats. Utsignalen ges således av

$$y[n] = \beta \cdot ((1-\alpha)x[n] + \alpha \sum_{i=1}^{M} b_i x[n - iL]) \tag{3.54}$$

I figur 4.1 nedan illustreras ett delay med två framkopplingar, där $b_0 = 1 - \alpha$.

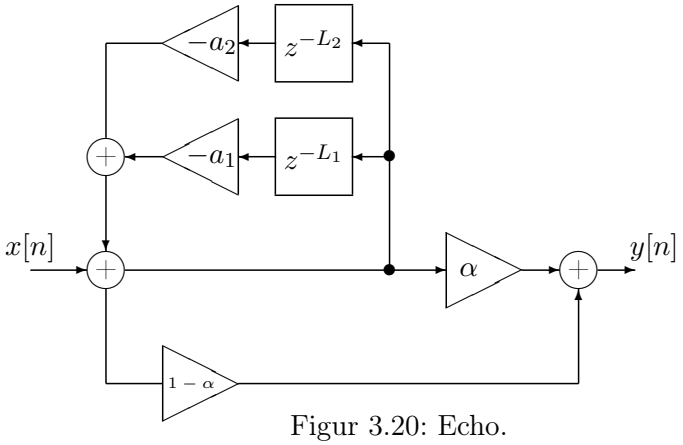


Figur 3.19: Delay.

Musikeffekten echo påminner mycket om det naturliga eko som uppstår då en ljudsignal studsar mot en hårdare yta. Dock skall ekots upprepning och dämpning specificeras av en användare snarare än landskapet.
Till skillnad mot delay är echo uppbyggt av det återkopplade kamfiltret vilket innebär att det är utsignalen och inte insignalen som upprepas enligt

$$y[n] = x[n] + a_1 y[n - L_1] + a_2 y[n - L_2] + ... + a_N y[n - L_N] = x[n] + \sum_{i=1}^{N} a_i y[n - L_i] \tag{3.55}$$

där varje upprepning kallas för ett *genljud*. På samma sätt som för delay styrs steglängderna, L_i, $i = 1, ..., N$ av musikens takt och tempo. Även genljuden mixas med insignalen på samma sätt som tidigare. Blockschema över effekten med två återkopplingar (genljud) ges av figur 3.20.



Figur 3.20: Echo.

där systemet kan beskrivas som

$$\frac{Y(z)}{X(z)} = \frac{\alpha}{1 + \sum_{i=1}^{N} a_i z^{-L_i}} + 1 - \alpha \tag{3.56}$$

vilket ger utsignalen

$$y[n] = x[n] + (1 - \alpha) \sum_{i=1}^{N} a_i y[n - L_i] \tag{3.57}$$

En intressant effekt kan uppnås i både delay och echo om koefficienternas dämpning skiljer sig mellan vänster och höger kanal. Effekten är maximal då fördröjningen växelvis varierar mellan kanalerna, vartannat taktslag i vänster och vartannat i höger.
En annan möjlighet att få denna studsande effekt utan att skilja koefficienterna är att använda sig av en tidsvarierande panorering (se appendix C) i takt med steglängden L.

För att implementera effekterna på utvecklingsplattformen krävs att sampel under en längre tidsperiod kan samlas in och lagras. Den cirkulära bufferten som presenterades i avsnitt 2.4.2 används därför för datahantering.
Buffertens längd per kanal bestäms av antalet fram- eller återkopplingar, N, samt steglängden, L enligt

$$L_{buff} = N \cdot L = N\left(\frac{f_s \cdot \lambda \cdot 60}{f_{tempo}}\right). \tag{3.58}$$

Buffertens längd begränsar alltså hur effekternas parametrar kan väljas. Ett rimligt antagande är att ingen låt spelas saktare än 60 bpm och att antalet fördröjningar aldrig överskrider 16 stycken. Den cirkulära buffertens längd blir då för taktarten $\frac{4}{4}$ och ljud i stereo:

$$L_{buff} = 2 \cdot 16\left(\frac{48000 \cdot \frac{4}{4} \cdot 60}{60}\right) = 1536000 \tag{3.59}$$

Vilket för inkommande ljud med ordlängden 16-bitar (2 bytes) kräver strax över 3 MB ledigt lagringsutrymme och överskrider de 1 MB som finns tillgängligt i internminnet. Vilket innebär att det långsammare SDRAM måste användas. Detta är dock inget problem eftersom effekten inte är särskilt krävande, mätningar visade att processorbelastningen för implementation med 16 steg uppgick till lite drygt 2.1%. Vilket med god marginal uppfyller kravspecifikationen.

3.4 Reverb

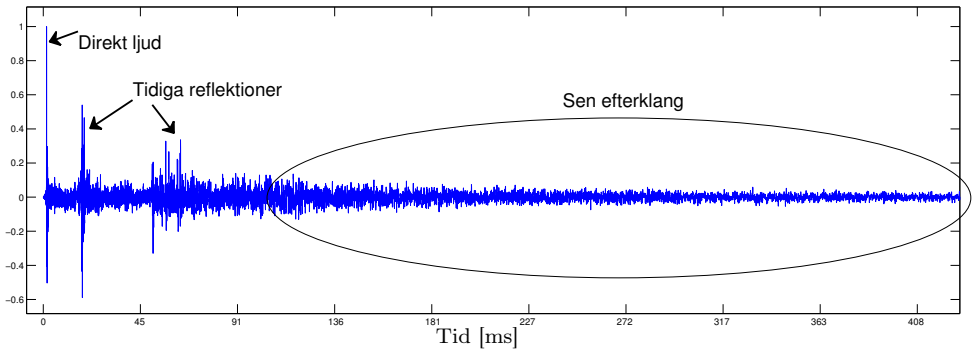
Då en ljudkälla placeras i ett rum kommer en viss del av ljudets energi absorberas i väggarna medan en viss del kommer att reflekteras. Lyssnaren kommer då att höra ljud från både den primära ljudkällan samt de dämpade, reflekterade ljudvågorna i form av ekon som tillsammans utgör vad som kallas rummets efterklang. Efterklangen upplevs olika mycket i olika miljöer. I en kyrka eller konserthall med hårda väggar är effekten påtaglig medan den i dämpade rum knappt är märkbar. I musiksammanhang kallas efterklangen för *reverb*.

En variabel för att mäta ett rums efterklangstid är T_{60} som avser den tid det tar för reflektionerna av en impuls att minska 60 dB [22].
Förutom att mäta upp T_{60} tiden finns även en empirisk formel kallad Sabines ekvation där T_{60} uppskattas utgående från ett rums storlek och materialkonstanter.

$$T_{60} = \frac{V \cdot 0.049}{\sum_{n=1}^{6} S_n \alpha_n} \tag{3.60}$$

där V är rummets volym, $S_n,\ n = 1, 2, ..., 6$ varje sidas area och $\alpha_n,\ n = 1, 2, ..., 6$ motsvarande absorptionskonstant [22].

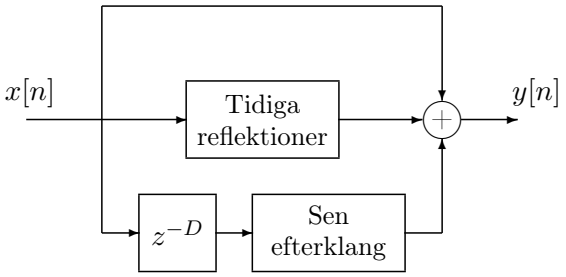
Rummets efterklang kan beskrivas genom dess impulssvar. Vilket beskriver med vilken fördröjning och amplitud varje reflektion når mottagaren. Impulssvaret kan i huvudsak uppskattas på två sätt, genom mätning eller beräkning. En välanvänd metod är att mäta upp ett rum med hjälp av källor och mottagare (högtalare och mikrofoner) [23]. Vilket kan utföras med enkla medel och garanterar precision eftersom ett faktiskt rum mäts. Nackdelar är förutom eventuella felkällor i mätutrustning även okontrollerade störningar från omgivning som påverkar mätresultatet [24].
I figur 3.21 visas ett karakteristiskt impulssvar uppmätt i en koncertlokal i Pori, Finland.



Figur 3.21: Impulssvar från en konserthall i Pori, Finland [25].

Ett rums impulssvar kan delas upp i olika delar. Det *direkta ljudet* är det ljud som når mottagaren innan det reflekteras. De *tidiga reflektionerna* är det ljud som inom ett kort tidsintervall efter det direkta ljudet når lyssnaren, vilka är särskilt viktiga för att uppskatta rummets storlek, material och geometri [20].
Det resterande ljudet kallas för den *sena efterklangen*. Denna del består av en mängd ekon som sammanvävts till den grad att de inte längre kan särskiljas från varandra. Detta ljud är inte lika specifikt som de tidiga reflektionerna och kan uppskattas med en förenklad brusmodell [26, 27].

Insignalen kan behandlas av de tre delarna var för sig, för att sedan adderas ihop och att ge det totala impulssvaret, se figur 3.22 där längden av fördröjningen D motsvarar längden av de tidiga reflektionerna.

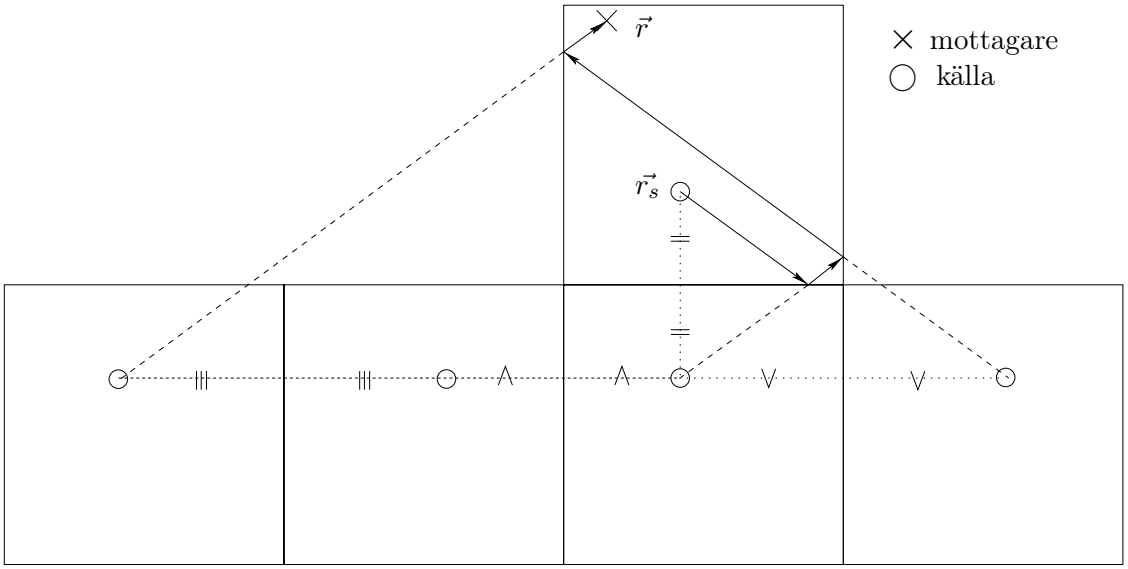


Figur 3.22: Övergripande schema av ett reverb.

Nackdelen med att mäta ett rums impulssvar är då ett nytt rum skall modelleras måste mätningarna göras om. En alternativ metod är uppskatta ett rums impulssvar utgående från ljudets fysiska egenskaper och ett antal förbestämda rumsparametrar. Fördelarna är då framförallt flexibilitet, då rumsparametrarna enkelt kan ändras för att simulera olika rum. Nedan presenteras metoder för att uppskatta de tidiga reflektionerna och den sena efterklangen. I slutet av avsnittet kombineras de båda modellerna för att åstadkomma den slutgiltiga effekten.

3.4.1 Tidiga reflektioner

För att modellera de tidiga reflektionerna används *speglingsmodellen*. Metoden kan anses vara ett specialfall av vågekvationen där ljud beskrivs som tryckvågor som utbreder sig i luft (se appendix D). Speglingsmodellen använder rummets geometri för att spegla en ljudkälla i rummets sidor och på så vis uppskatta ljudets reflektioner mot dessa, se figur 3.23.
Då det är ljudtryckets energiutbredning som beskrivs vid reflektionerna tar inte speglingsmodellen hänsyn till de fasskillnader som skulle kunna orsaka positiv och negativ interferens. Denna begränsning har mindre inverkan då ljudet kan antas innehålla ett brett spektrum av frekvenser då de olika fasvridningarna antas ta ut varandra [28].



Figur 3.23: Speglat rum i två dimensioner [28].

Avståndet som en reflekterad ljudvåg färdas från ljudkällan placerad i $\vec{r_s}$ innan den når mottagaren vid $\vec{r}$ kan uttryckas som

$$\|R_c\| = \|((1-2c)\vec{r_s} - \vec{r})\| = \|((1-2c_1)x_s - x), ((1-2c_2)y_s - y), ((1-2c_3)z_s - z)\| \quad (3.61)$$

där $c = (c_1, c_2, c_2),\ c_1, c_2, c_3 \in \{0, 1\}$ indexerar vilken sidkombination, i det tredimensionella rum, som reflektionerna sker [28]. Vilket innebär att ett rektangulärt rums sex olika sidor kan kombineras på $2^3 = 8$ olika sätt. $\|\cdot\|$ avser L_2-normen.

Genom att öka antalet speglade rum tas fler reflektioner med i beräkningen. Vilket uttrycks som

$$\|R_m\| = \|(2m_xL_x,\ 2m_yL_y,\ 2m_zL_z)\| \quad (3.62)$$

där m_x, m_y och m_z är heltal som kan anta värden mellan $-N \leq m \leq N$, där $(2N)^3$ är antalet speglingar och $L_x,\ L_y$ och L_z är rumsdimensionerna [29].

Den totala vägen som den reflekterade ljudvågen färdats är:

$$d = \|R_c + R_m\|. \quad (3.63)$$

Där tiden, τ, det tar att färdas sträckan enkelt beräknas

$$\tau = \frac{d}{v} \quad (3.64)$$

då $v = 343$ m/s är ljudets utbredningshastighet i luft [28].

Tiden som varje reflektion fördröjs då den speglas i rummets sidor kan alltså uppskattas. För att bestämma motsvarande amplitud, $p(\vec{r}, \vec{r_s}, t)$, används vågekvationen [29]. Lösningen till denna i frekvensdomän då källa och mottagare befinner sig i fri rymd är

$$P(\vec{r}, \vec{r_s}, \omega) = \frac{e^{j\omega\frac{\|\vec{r}-\vec{r_s}\|}{c}}}{4\pi\|\vec{r}-\vec{r_s}\|}e^{-j\omega t} \quad [29]. \quad (3.65)$$

Om en förlustfri vägg introduceras kan ekvation 3.65 skrivas om med hjälp av en speglad lösning [29].

$$P(\vec{r}, \vec{r_s}, \omega) = \left(\frac{e^{j\omega\frac{\|\vec{r}-\vec{r_s}\|}{c}}}{4\pi\|\vec{r}-\vec{r_s}\|} + \frac{e^{j\omega\frac{\|\vec{r}+\vec{r_s}\|}{c}}}{4\pi\|\vec{r}+\vec{r_s}\|}\right)e^{-j\omega t} \quad (3.66)$$

Genom att generalisera uttrycket för de sex sidorna i ett rektangulärt rum samt föra tillbaka lösningen till tidsdomänen ges

$$p(\vec{r}, \vec{r_s}, t) = \sum_{c=0}^{1}\sum_{m=-N}^{N}\frac{\delta(t-\tau)}{4\pi d_{c,m}} \quad [29] \quad (3.67)$$

Lösningen för det icke förlustfria fallet till ekvation 3.66 är mycket komplext, istället förenklas lösningen genom att skala det förlustfria fallet med en vinkeloberoende reflektionskonstant, β [29]. Reflektionskonstanten relateras till Sabines energiabsortionskonstant, α, enligt

$$\beta = \pm\sqrt{1-\alpha} \quad [22]. \quad (3.68)$$

Vid reflektion mot sidorna, som betsår av ett tätare medium, fasvrids även ljudet 180° och den nya modellen ges av

$$p(\vec{r}, \vec{r_s}, t) = \sum_{c=0}^{1}\sum_{m=-N}^{N}\beta\frac{\delta(t-\tau)}{4\pi d_{c,m}} \quad (3.69)$$

där

$$\beta = (-\beta_{x_1}^{|m_x-q|})(-\beta_{x_2}^{|m_x|})(-\beta_{y_1}^{|m_y-j|})(-\beta_{y_2}^{|m_y|})(-\beta_{z_1}^{|m_z-k|})(-\beta_{z_2}^{|m_z|}) \quad (3.70)$$

beskriver reflektionskonstanterna i rummets samtliga sidor och där summan

$$\sum_{c=0}^{1} \sum_{m=-N}^{N}$$

avser

$$\sum_{c_1=0}^{1} \sum_{c_2=0}^{1} \sum_{c_3=0}^{1} \sum_{m_x=-N}^{N} \sum_{m_y=-N}^{N} \sum_{m_z=-N}^{N}.$$

Sammanfattningsvis kan alltså impulsvaret uppskattas genom att specificera $\vec{r}$, $\vec{r}_s$, $\vec{L}$, N och β enligt ekvation 3.69. Parametrarna $\vec{L}$ och β används för att modellera rummets fysiska egenskaper. Lyssnarens och källans placering i rummet kan ändras genom att en användare specificerar $\vec{r}$ och $\vec{r}_s$ Med ökat N tas fler reflektioner med i beräkningen vilket ger ett noggrannare resultat.

Eftersom speglingsmodellen är en mycket resurskrävande algoritm utfördes simuleringarna av två stycken rum, ett mindre och ett större, i MATLAB. De olika rummen beräknades med olika absorbtionskonstanter för att jämföra skillnaden mellan material se tabell 3.4.

Tabell 3.4: Två olika rum simulerade för olika absorbtionskonstanter

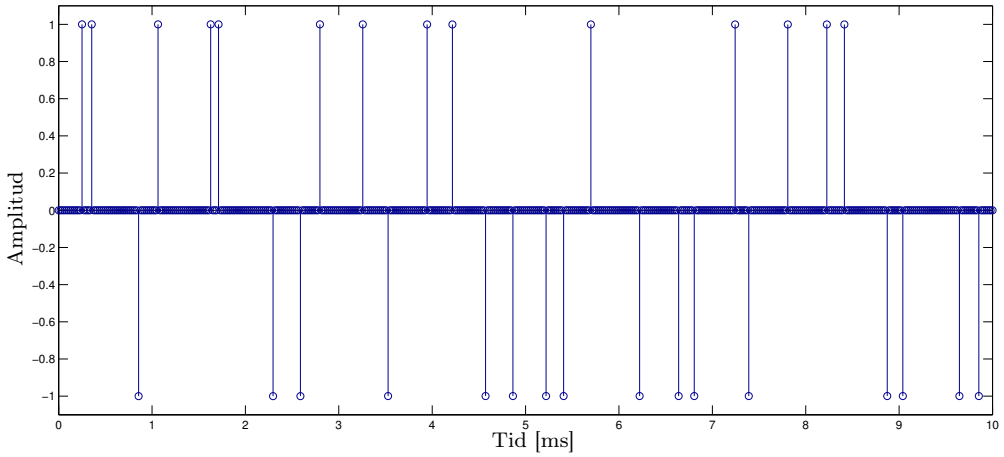
Rum	Rumsdimensioner: bredd,längd,höjd [m]	Ljudkällans dimensioner [m]	Mottagarens dimensioner [m]	α	$T_{60}[s]$	Beräkningstid [timmar]
1	6,4,3	3,2,1.5	2.44,2,1.5	0.08	1.34	2
1	6,4,3	3,2,1.5	2.44,2,1.5	0.15	0.72	0.5
1	6,4,3	3,2,1.5	2.44,2,1.5	0.38	0.34	.25
2	33,23,15	11.5,5,2	28,18.6,5	0.04	2.89	16
2	33,23,15	11.5,5,2	28,18.6,5	0.15	1.53	12
2	33,23,15	11.5,5,2	28,18.6,5	0.38	1.09	5
2	33,23,15	11.5,5,2	28,18.6,5	0.5	0.95	3

Varibeln N valdes på ett sådant sätt att reflektioner med fördröjningar $\tau > T_{60}$ försumades. Då $\alpha < 0.15$ representerar lågabsorberande ytor som glas medan större α i simuleringen representerar olika träsorter [22].

Samtliga simuleringar har gjorts för två snarlika mottagarplaceringar skilda 12 cm för att på så vis uppskatta öronens placering. Notera att ingen hänsyn har tagits till direktivitet associerad med huvudet.

3.4.2 Sen efterklang

Reflektionerna i den sena efterklangen modelleras med slumpmässiga värden, det vill säga ett brus. Modellen är inte alls fysikaliskt korrekt men ändå användbar. Särskilt då ett visst typ av brus, kallat *sammetsbrus* används [27]. Bruset består av en sekvens av enhetsimpulser som placeras slumpmässigt inom givna intervall med varierande tecken, se figur 3.24.



Figur 3.24: 10 ms av sammetsbrus, med 1500 enhetsimpulser per sekund.

Av figur 3.24 framgår att amplituden är densamma för samtliga impulser, endast tecknen skiljer dem åt. Detta gör att sammetsbruset upplevs mjukare än brus där även amplituden varierar [27].
En fördel med att använda sig av denna sortens brus istället för det riktiga impulssvaret, är att det innehåller färre nollskilda koefficienter. Detta kan utnyttjas genom att endast behandla de nollskilda elementen, med ett glest FIR-filter, eller SFIR-filter (Sparse Finite Impulse Response), där koefficienterna representeras av index- och amplitudtabeller.

Sammetsbruset kan beskrivas som

$$n(k) = \sum_{m=0}^{M} a(m)\delta(k - l(m)) \tag{3.71}$$

där $a(m) \in \{-1, 1\}$ varierar slumpmässigt, M är antalet enhetsimpulser och $l(m) \in \mathbb{N}$ är den slumpmässiga positionen, avrundat till närmaste heltal enligt

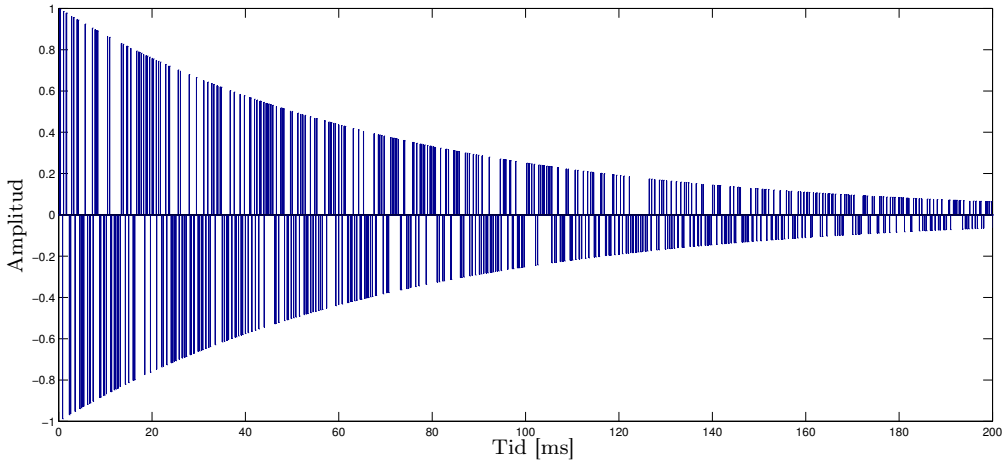
$$l(m) \approx (\frac{T_d}{T_s}(m + s(m)))$$

där T_d är den genomsnittliga tid mellan impulserna, och T_s är samplingstiden [27]. Funktionen $s(m)$ är ett slumpmässigt tal mellan noll och ett.

För att approximera ett verkligt reverb skall efterklangen avta exponentiellt, vilket uppnås om koefficienterna skalas med

$$e(t) = 10^{-3t/T_{60}} \tag{3.72}$$

där T_{60} beräknas enligt ekvation 3.60. Sammetsbruset, efter skalning visas i figur 3.25.



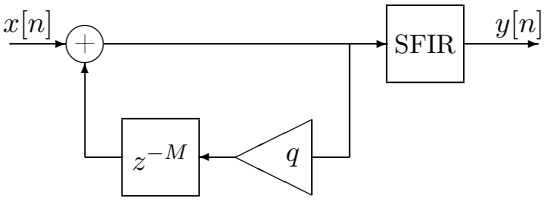
Figur 3.25: Exponentiellt avtagande sammetsbrus.

Då efterklangstiden varierar olika för olika frekvenser kan reverbet förbättras ytterligare genom att bruset behandlas av en equalizer [27]. Hur de olika frekvenserna dämpas beror på rummets dimensioner och materialet i väggarna [22].

Efterklangen kan för vissa rum, till exempel en konsertlokal, vara mycket lång. Trots användning av SFIR-filter kan även implementation med sammetsbrus, för rum med låg dämpning vara alltför krävande. Detta kringgås genom att återkoppla utsignalen. Fördröjningen M motsvaras då av filtrets längd. Återkopplingen skalas dessutom med

$$q = 10^{-\frac{3M}{f_s T_{60}}} \tag{3.73}$$

så att den fördröjda signalen anpassas till efterklangen, se figur 3.26 [27].



Figur 3.26: Modellering av den sena efterklangen.

3.4.3 Modellerat impulssvar

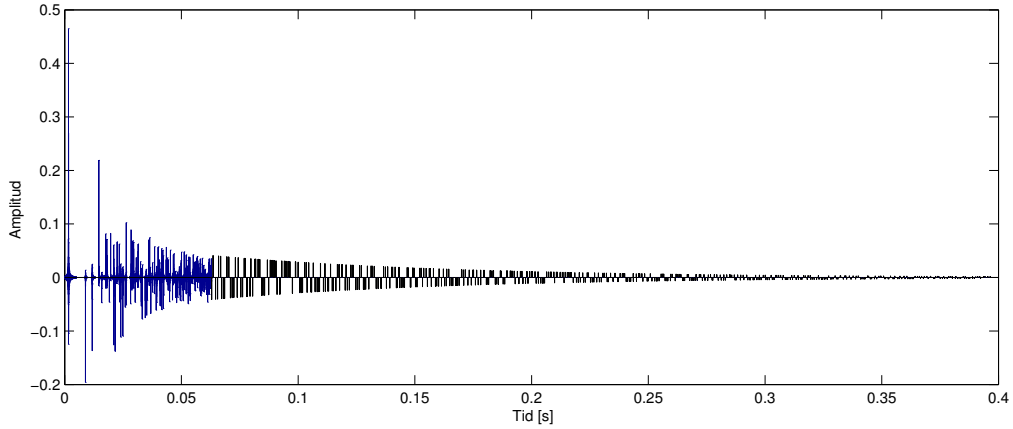
Det totala impulssvaret ges genom att kombinera de tidiga reflektionerna med den sena efterklangen.

I ett impulssvar som beräknas med speglingsmodellen är de flesta koefficienterna nollskilda, vilket innebär att ett FIR-filter av hög ordning krävs för implementering. Dock är amplituden för ett större antal koefficienter nära noll. Dessa koefficienter försummas för att filtrering effektivt skall utföras med ett SFIR-filter även för de tidiga reflektionerna. Tröskeln för vilka sampel som klipps bort kan varieras, så att koefficienterna i små rum är tätt förekommande medan de i stora rum är placeras glesare.

Effekten implementeras med två impulssvar, ett för varje öra. Fördröjda sampel lagras i två cirkulära buffertar. Den ena för det tidiga ljudet och den andra är för sammetsbruset. Dessutom behandlas block av sampel åt gången för att data skall läsas i sekventiell ordning, vilket

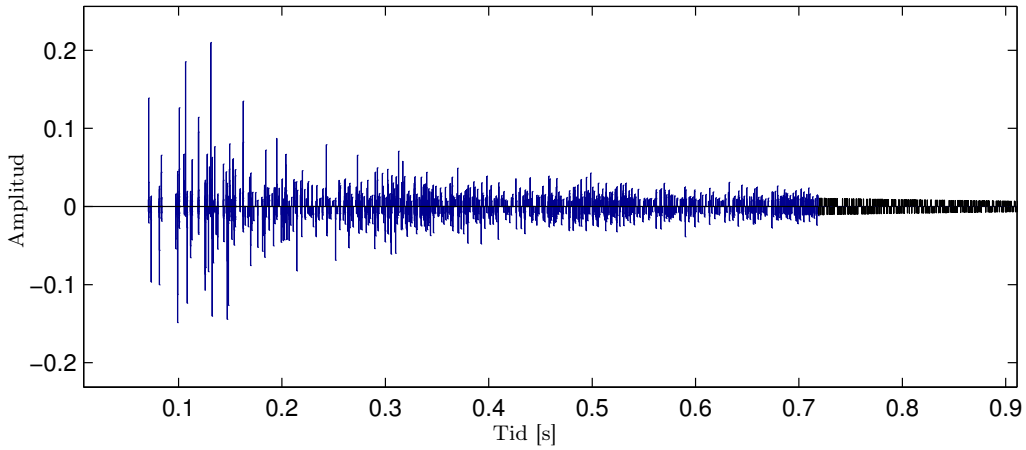
ökar effektiviteten. För att uppfylla den i inledningen specificerade kravspecifikationen kan antalet koefficienter per kanal uppgå till 2200 stycken. Detta ger en processoranvändning på 32.1%. Maximal längd på impulssvaret innan processorn överbelastas är cirka 6600 koefficienter per kanal.

I figur 3.27 har ett litet trärum simulerats genom att sätta $\alpha = 0.15$ och $\vec{L} = (6, 4, 3)$ m. I figuren uppgår antalet nollskilda koefficienter i det tidiga impulssvaret till 2000 stycken, medan resterande 200 är återkopplat sammetsbrus.



Figur 3.27: Impulssvaret för ett litet trärum.

Impulssvaren skiljer sig mycket från rum till rum. I figur 3.28 har ett stort glasrum simulerats genom att sätta $\alpha = 0.04$ och $\vec{L} = (33, 23, 15)$ m, även här är antalet nollskilda koefficienter 2200 stycken.



Figur 3.28: Impulssvaret för ett stort glasrum.

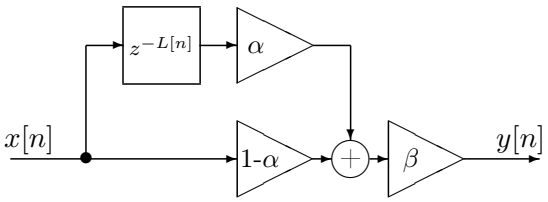
4 Tidsvarianta musikeffekter

En tidsvariant musikeffekt är ett system som är dynamiskt. Ett sådant system kan inte bestämmas av ett impulssvar förutom vid fixa tidpunkter. Om en insignal, $x[n]$, alltså vid en specifik tidpunkt ger utsignalen $y[n]$ behöver nödvändigtvis inte den tidsskiftade insignalen $x[n - L]$ resultera i utsignalen $y[n - L]$.

Det förekommer ett antal sådana musikeffekter i denna rapport vilka presenteras nedan.

4.1 Flanger

Musikeffekten flanger åstadkoms genom att spela upp två kopior av samma spår med olika fördröjningar, vilket ger ett svepande ljud. Effekten skapades ursprungligen med två analoga rullbandsspelare där rullen på den ena hindrades med handen. Musikeffekten kan beskrivas som ett framkopplat kamfilter med varierande tidsfördröjning [30, 31]. Blockschemat av den implementerade effekten visas i figur 4.1.



Figur 4.1: Flanger.

Utsignalen ges alltså av

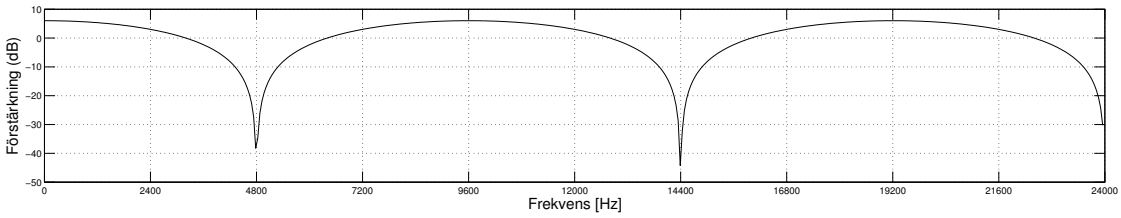
$$y[n] = \beta((1 - \alpha)x[n] + \alpha x[n - L[n]]) \tag{4.1}$$

där α precis som i effekterna delay och echo är effektens mix. Utsignalen skalas med β, som ges av

$$\beta = \frac{1}{\sqrt{(1 - \alpha)^2 + \alpha^2}} \tag{4.2}$$

för att medeleffekten ska hållas konstant.

Tidsfröjdröjningen i flanger varierar mellan $0 - 10$ ms [20]. Vid så små fördröjningar går det inte att höra skillnad på originalsignalen och den fördröjda versionen. Tidsfördröjningen ger då istället upphov till en varierande interferens mellan signalerna. I frekvensspektrat yttrar detta sig som återkommande notchar, vilket ger den karakteristiska form som gett kamfiltret dess namn, se figur 4.2 [30].



Figur 4.2: Amplitudfunktionen för ett framkopplat kamfilter, $\alpha = 0.45$.

Störst effekt erhålls när $\alpha = 0.5$ eftersom signalerna då har lika stor amplitud, vilket ger maximal interferens.

Flangerns fördröjning, $L[n]$, varierar periodiskt, ofta i form av en trigonometrisk funktion. Detta leder till att notcharnas positioner kommer att variera, vilket ger upphov till det svepande ljudet. Den varierande fördröjningen kan skrivas som

$$L[n] = \frac{t_m f_s}{2} \cdot (1 - cos(2\pi f_{mod} \frac{n}{f_s})) \tag{4.3}$$

där t_m beskriver den maximala tidsfördröjningen och f_{mod} är modulationsfrekvensen, vilket är den frekvens fördröjningen varieras med [32]. Notcharnas placering kan då beräknas enligt

$$f_{notch}[k] = k \cdot \frac{fs}{L[n]} \qquad k = 0, 1..., L[n] - 1. \tag{4.4}$$

Användarparametrarna är modulationsfrekvensen (f_{mod}), djupet av effektens verkan (α) och den maximala tidsfördröjningen (t_m).

Eftersom trigonometriska funktioner, som den i ekvation 4.3, är tidskrävande att beräkna utan hårdvarustöd för flyttal lönar det sig att skapa en uppslagstabell med motsvarande fixtalsvärden vid systemets uppstart.

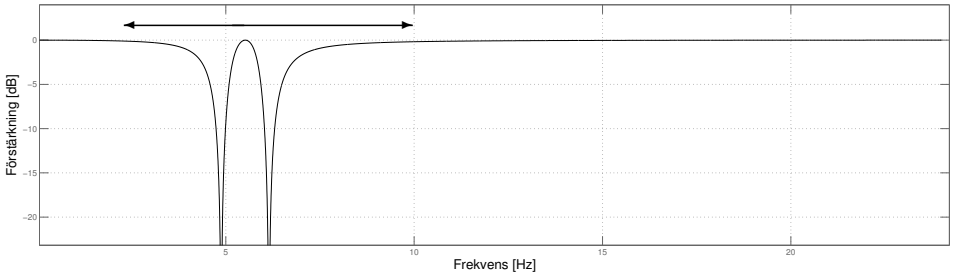
Effekten använder sig av en cirkulär buffert för att spara tidsfördröjda sampel. Då antalet sampel som omfattas av t_m inte är så många, innebär det att flera olika värden hos $L[n]$ kommer att avrundas till samma fördröjning. Följaktligen kommer övergångarna mellan olika fördröjningar uppfattas som diskreta hopp vilket kan orsaka brus. I brist på sampel som ligger på exakt den tidsfördröjning som efterfrågas av $L[n]$, kan signalen approximeras mellan samplingarna genom interpolation.
I implementationen används kvadratiskt interpolation mellan tre närliggande sampel enligt Lagranges interpolationsformel för tre punkter med jämnt mellanrum

$$P(x_0 + th) = \frac{t(t-1)}{2} f(x_0 - h) + (1 - t^2) f(x_0) + \frac{t(t+1)}{2} f(x_0 + h) \quad [33] \tag{4.5}$$

vilket gör att övergångarna mellan olika fördröjningar inte längre uppenbarar sig i utsignalen.
Implementationen av flangern använder endast 3% av processorns belastning, vilket är rimligt med tanke på att antalet beräkningar som krävs.

4.2 Phaser

Phaser är en effekt som tidigare har använts som ett sätt att efterlikna flanger på ett enklare och mer portabelt sätt [34]. Istället för att använda otympliga rullbandspelare kunde en liknande effekt åstadkommas med hjälp av analog elektronik. Effekten kan beskrivas som ett multipelt notchfilter, där notcharnas placering varierar över tid, se figur 4.3. Phaser består i regel av mycket färre notchar än flanger, antalet notchar kan dessutom varieras, och måste inte vara placerade med jämna intervall.



Figur 4.3: En phaser med två notchar som varierar över tid.

Vid design av ett notchfilter, släcks en frekvens f_0 ut genom att placera ett par komplexkonjugerade nollställen på enhetscirkeln enligt:

$$z_0^{\pm} = e^{\pm j2\pi f_0/f_s}. \tag{4.6}$$

Vilket ger överföringsfunktionen av för ett notchfilter av FIR-typ.

$$H(z) = (z - z_o^+)(z - z_0^+) \tag{4.7}$$

För att minska bandbredden placeras ett polpar vid samma frekvens fast något skalade i amplitud för att skapa resonans. Filtret är då av IIR-typ och ges av

$$H(z) = \frac{(z - z_0^+)(z - z_0^-)}{(z - rz_0^+)(z - rz_0^-)},\ z_0^{\pm} = e^{\pm j2\pi f_0/f_s},\ r \lesssim 1 \tag{4.8}$$

där filtrets branthet ökar då r går mot ett. Genom att använda Eulers formel och förlänga bråket med z^{-2} kan uttrycket skrivas som

$$H(z) = \frac{1 - 2\cos(2\pi f_0/f_s)z^{-1} + z^{-2}}{1 - 2r\cos(2\pi f_0/f_s)z^{-1} + r^2 z^{-2}} \tag{4.9}$$

Eftersom polerna kan placeras nära, men inte på enhetscirkeln, kommer systemet att förstärka de övriga frekvenserna i systemet. Vilket kan minskas genom att placera ytterligare poler och nollställen i närheten av resonansfrekvensen. Detta skulle innebära att filter av högre ordningstal krävs för att åstadkomma notcharna.

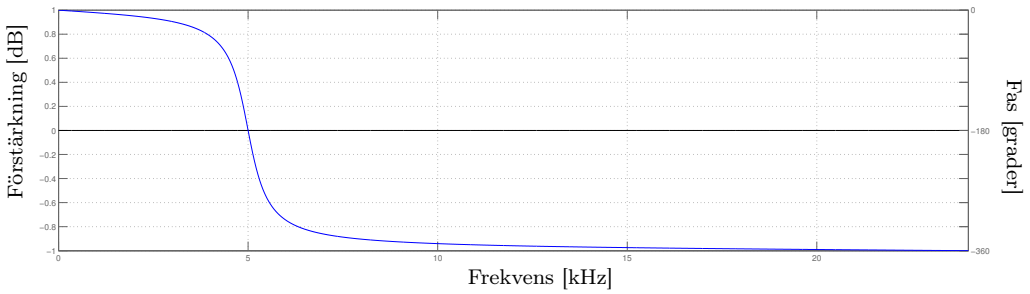
En bättre metod är därför att designa notchfiltret som

$$H(z) = \frac{1}{2}[1 + G_{AP}(z)] \tag{4.10}$$

där allpassfilter, G_{AP} har placerats på samma vinkelfrekvens som notchfiltret ovan, vilket ger överföringsfunktionen

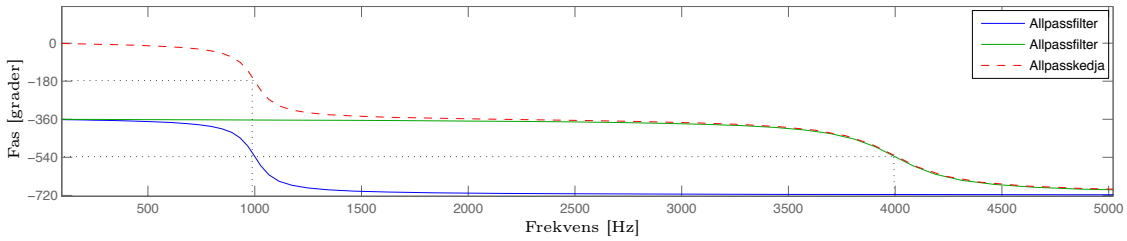
$$G_{AP} = \frac{r^2 - (1+r)^2 cos(2\pi f_0/f_s)z^{-1} + z^{-2}}{1 - (1+r)^2 cos(2\pi f_0/f_s)z^{-1} + r^2 z^{-2}} \tag{4.11}$$

Ett allpassfilter är ett filter som inte förändrar en signals amplitud men skapar en fasvridning, se figur 4.4.



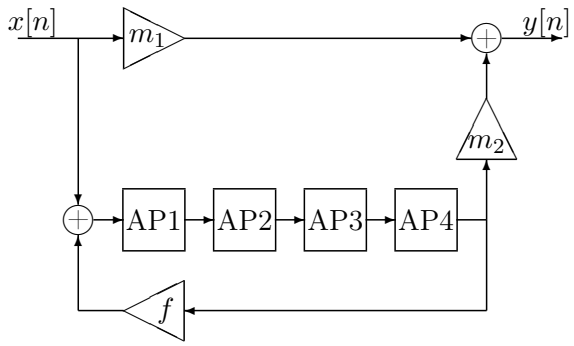
Figur 4.4: Amplitud- och fasfunktion för ett andra ordningens allpassfilter, $f_0 = 5000$ Hz.

När den fasvridna signalen adderas till originalsignalen bildas en notch vid den frekvens där fasskillnaden mellan signalerna är 180°, det vill säga vid f_0. Fördelen med denna design är att då pol- och nollställen placeras på samma avstånd från enhetscirkeln. kommer amplitudfunktionen aldrig att överskrida ett [35].
På ett liknande sätt har en phaser med flera notchar åstadkommits. Genom att kaskadkoppla flera andra ordningens allpassfilter ökar den totala fasskillnaden ytterligare 360° jämfört med orginalsignalen, för varje filter som kopplas på, se figur 4.5. Detta leder till att det skapas en ny notch för varje allpassfilter, dessa uppstår där skillnaden i fas är $180° + n \cdot 360°$. I den implementerade phasern har fyra kaskadkopplade allpassfilter av andra ordningen använts.



Figur 4.5: Fasfunktionen för en kedja med två stycken andra ordningens allpassfilter, där den totala fasvridningen är summan av de ingående allpassfilternas fasvridning.

Om ingen dämpning sker för att hålla nere amplituden, uppstår förstärkning vid addering av de två signalerna, vid de frekvenser där dessa är i fas. Detta på samma sätt som det uppstår notchar där signalerna är i motfas. Ytterligare förstärkning kan åstadkommas genom att införa återkoppling av allpasskedjan på det sätt som visas i figur 4.6 [30].



Figur 4.6: Blockschema över en phaser med återkopplad allpasskedja. Mängden återkoppling styrs av parametern f.

Överföringsfunktionen för allpasskedjan betecknas här $G_k(z)$. Eftersom den består av allpassfilter, så blir amplitudfunktionen konstant $|G_k(j\omega)| = 1$, och frekvenssvaret blir då

$$G_k(j\omega) = 1 \cdot e^{j\angle G_k(j\omega)} = cos(\angle G_k(j\omega)) + jsin(\angle G_k(j\omega)). \tag{4.12}$$

Överföringsfunktionen för allpasskedjan med återkoppling, $G_{kf}(z)$, blir

$$G_{kf}(z) = \frac{G_k(z)}{1 - fG_k(z)} \tag{4.13}$$

vars frekvenssvar blir

$$G_{kf}(j\omega) = \frac{1 \cdot e^{j\angle G_{apk}(j\omega)}}{1 - f \cdot 1 \cdot e^{j\angle G_{apk}(j\omega)}} \tag{4.14}$$

vilket ger en skillnad i amplitudfunktionen, som inte längre är konstant för alla frekvenser. Återkopplingen leder till att de frekvenser som är i fas med orginalsignalen blir förstärkta redan innan de två signalerna adderas samman.

Förutom den mer distinkta förstärkningen, kan återkoppling leda till att notcharna inte går lika djupt, därför togs det i arbetet fram en metod som motverkar detta. De betydligt grundare notcharna beror på att signalen efter den återkopplade allpasskedjan är dämpad vid de frekvenser där fasskillnaden mot orginalsignalen är 180°. Det bildas därför inte lika stor dämpning vid addering med orginalsignalen. För att kraftiga notchar ska bibehållas måste de två signalerna ha samma amplitud, vilket kan justeras med m_1 och m_2 i figur 4.6.

Ur ekvation 4.14 kan amplitudfunktionen för den återkopplade allpasskedjan beräknas som

$$|G_{kf}(j\omega)| = \frac{1}{\sqrt{(1 - f \cdot cos(\angle G_k(j\omega)))^2 + (f \cdot sin(\angle G_k(j\omega)))^2}}. \tag{4.15}$$

Vid notchfrekvenserna blir amplituden därför

$$|G_{kf}(j\omega)|_{\angle G_k(j\omega)=180} = \frac{1}{1+f} \tag{4.16}$$

Vilket ger att maximalt dämpade notchar uppnås om förhållandet mellan m_1 och m_2 är:

$$\frac{m_1}{m_2} = \frac{1}{1+f} \tag{4.17}$$

För att låta användaren styra mängden effekt, på samma sätt som parametern α i flanger, skalas förhållandet i ekvation 4.17 ner med en parameter I, vilket ger följande villkor:

$$m_2 = I(1+f)m_1 \tag{4.18}$$

Dessutom normaliseras mixen med villkoret:

$$m_1 + m_2 = 1 \tag{4.19}$$

Ovanstående två villkor leder till att:

$$m_1 = \frac{1}{1 + I(1 + f)} \tag{4.20}$$

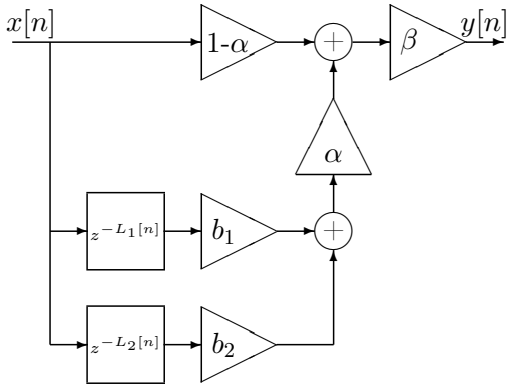
$$m_2 = \frac{I(1 + f)}{1 + I(1 + f)} \tag{4.21}$$

En implementation av phasern har gjorts där notchar sveper mellan 1000 och 10000 Hz, och utifrån metoderna som beskrivits ovan finns det stora möjligheter att experimentera vidare med notcharnas placering och svep. Allpasskedjan har implementerats av kaskadekopplade steg på direkt form på samma sätt som den parametriska equalizern för att minimera kvantiseringseffekter. Användaren kan specifera mängden återkoppling, $f \in [0, 0.5]$ och parametern $I \in [0, 1]$ som styr hur mycket av den fasvridna signalen som ska mixas samman med orginalsignalen. Precis som flanger kan även frekvensen på variationen styras med en parameter. På grund av den låga filterordningen belastade effekten endast signalprocessorn med 1.7%.

4.3 Chorus

Chorus används för att skapa en bredare och fylligare ljudbild. Detta förverkligas genom att flera närliggande versioner av samma signal med slumpmässiga små tidsförskjutningar tolkas som en. Fenomenet uppstår naturligt då exempelvis flera personer i en kör sjunger samma stämma. Deras sammanlagda sångstämma ger då upphov till ett chorus.

Chorus har många likheter med andra frekvensmodulerande effekter som flanger och phaser där samtliga kan representeras med framkopplat kamfilter. Specifikt för chorus är slumpmässigt överlagrade fördröjningar mellan $10 - 25$ ms [32]. För att styra chorus används parametrarna blanding, intensitet och frekvens (α, I och f). Likt tidigare är α effektens blandning med originalsignalen. Intensiteten är ett mått på spannet hos den slumpmässiga fördröjningen. Där den maximala intensiteten $I = 1$ medför slumpmässigt varierande tidsfördröjningar i storleksordningen $10 - 25$ ms och $I = 0$ medför en konstant fördröjning på 17.5 ms. Frekvensvariabeln bestämmer med vilken uppdateringsfrekvens slumpgeneratorn skall generera nya värden.



Figur 4.7: Två stycken tidsvarierande framkopplade fördröjningar som tillsammans utgör ett chorus

Figur 4.7 beskriver en schematisk bild av två stycken framkopplade fördröjningar. Detta kan beskrivas som

$$y[n] = \beta((1-\alpha)x[n] + \alpha(b_1 x[n - L_1[n]] + b_2 x[n - L_2[n]])) \tag{4.22}$$

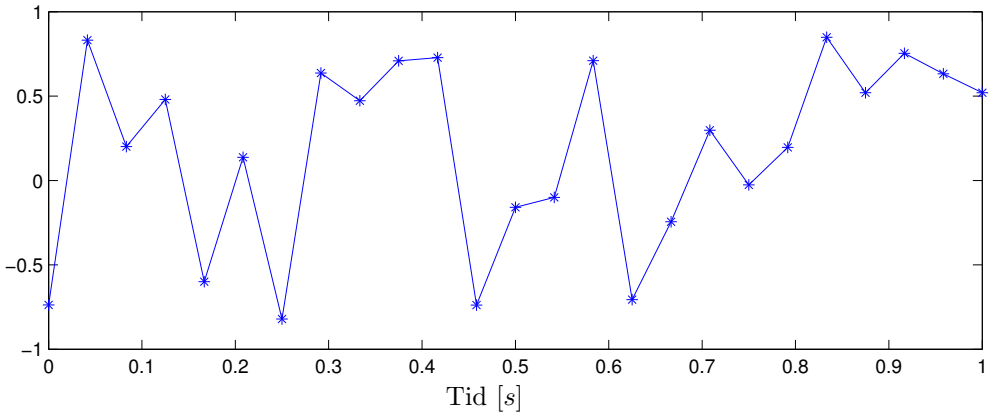
Fördröjningarna $L_1[n]$ och $L_2[n]$ varierar slumpvis mellan 10-25 ms vilket kan beskrivas som

$$L_1[n] = (N_0 + I\frac{15}{2}v_1[f,n])f_s \tag{4.23}$$

där

$$N_0 = 10 + \frac{15}{2}\,ms$$

utgör fördröjningen som de slumpmässiga värdena är centrerade runt. $v_i[f,n]$ representerar en funktion som linjärt vandrar mellan slumpvärden i intervallet [-1,1] med frekvensen f. På samma sätt fås $L_2[n]$.



Figur 4.8: Exempel på hur $v_i[f,n]$ kan variera över tid.

För att choruseffekten skall bli mer påtaglig kan fler antalet slumpstyrda fördröjningar ökas. Varje ny slumpmässig fördröjning representerar ytterligare en musikant som spelar samstämmigt.

Likt delayeffekten viktas alla fördröjningar med olika koefficienter (b_i), skillnaden här är att koefficienterna har ungefär samma värden, de är inte avtagande som i delayeffekten. Likt ekvation 3.54 i delay avsnittet normaliseras choruset med en variabel

$$\beta = \frac{1}{\sqrt{(1-\alpha)^2 + (\alpha\sum_{i=1}^{M} b_i)^2}}. \tag{4.24}$$

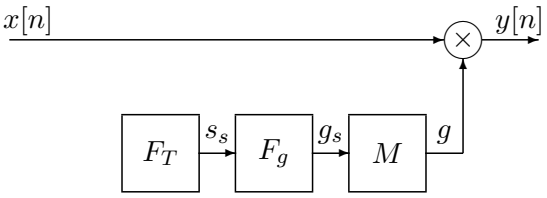
Implementationen av choruset sker på liknande sätt som flanger, en cirkulär buffert används för att spara gamla sampelvärden. Fördröjningsvärden som ligger mellan samplade värdens tid interpoleras genom kvadratisk interpolation. Den stora skillnaden är hur det fördröjda värdet räknas fram, här med slumptalsgenerator istället för en trigonometrisk funktion.

Chorus fördröjda framkopplingar kallas traditionellt för spår. Hur mycket kraft choruseffekten kräver av processorn beror på antalet spår, fler spår leder till fler interpoleringar och mer slumptalsgenerering. Implementering av chorus med tre spår uppmättes belasta processorn 9.3%.

4.4 Tremolo

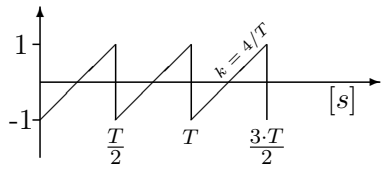
Tremolo är ett sätt att variera amplituden i en upprepande ordning. Genom att införa en snabb variation i signalens amplitud, kan instrument få en känslosam karaktär utan någon skiftning i frekvenserna. Detta kan speciellt vara eftertraktat i instrument som annars har en tendens att låta monotona [36]. Ett av de vanligare instrumenten där tremoloeffekten appliceras på är elgitarrer. Elgitarrer är dessutom ofta utrustade med en tremoloarm men denna skall dock inte förväxlas med en "riktig" tremolo. Eftersom tremolo-armen producerar effekten genom att spänna strängarna så skiftas frekvenserna istället för amplituden. Även om detta också kan ge variationer i amplituden så är det mer förknippat med vibrato.

Tremolots uppbyggnad visas i figur 4.9. Där de tre viktigaste delarna är den trigonometriska funktionen, F_T, viktningsfunktionen, F_g, och det mjukgörande filtret, M. F_T genererar en sinusformad signal, s_s, till F_g där denna skalar om och lyfter signalen. Sist i ledet finns ett mjukgörande filter, M, som hindrar eventuella abrupta övergångar, skapade av g_s. Resulterande skalningsfaktorn g används för att vikta insignalen $x[n]$.



Figur 4.9: Tremolots uppbyggnad.

I den trigonometriska funktionen kan användaren välja bland tre olika funktioner, sinus-, sågtand- och fyrkantsvåg. För att skapa sågtand- och fyrkantsvågen definierades intervall mellan noll och T, där T är den valda periodtiden. Genom att uttrycka lutningskoefficienten, k, i T blir funktionerna skalbara vilket kan ses figur 4.10.



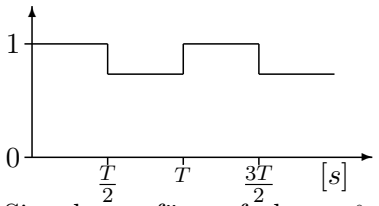
Figur 4.10: Sågtandsfunktion, genererad av F_T.

För att förhindra att signalen överskrider intervallet, T, används modulus. Jämna multipler av T kommer då försvinna och resten placeras inom intervallet. Eftersom det redan, i ett tidigare skede hade implementeras en sinusfunktion, lades ingen vikt på att skapa en motsvarande från grunden. Genom att förse funktionerna med ett stigande värde så generas signalen s_s för viktningsfunktionen.

Viktningsfunktionen tar vara på styrsignalen och använder denna för att beräkna g_s,

$$g_s = (1 - \alpha) - \alpha \cdot s_s. \tag{4.25}$$

Signalen lyfts då till en nivå 1-α där $\alpha \in [0,1]$. Skalande faktorn g_s varieras därefter med $\alpha \cdot s_s$. På så sätt så kommer värdet aldrig överstiga ett, och därmed inte heller förstärka signalen. Istället kommer g_s variera mellan $(1 - 2 \cdot \alpha) \leq s_s \leq 1$. Resultatet ges av figur 4.11.

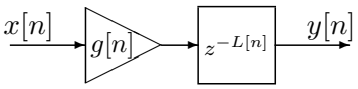


Figur 4.11: Signalen g_s för en fyrkantsvåg med $\alpha = 0.4$.

Mjukgörande funktionen kontrollerar att skalningsfaktorns fluktuation inte genererar oönskat brus. Likt kompressorns RMS-detektor skapas en tom skiftande buffert. Genom att beräkna medelvärdet, g, för hela bufferten, introduceras en tröghet i variationerna. För att erhålla optimal, det vill säga minsta buffertstorlek utan signalbrus, kan användaren ansätta värdet själv. Annars används en buffertstorlek som klarar $\alpha = 1$ med låga brusnivåer. För att minimera processoranvändandet så skapades en tabell med alla värden som g antar. På så sätt behövs viktningsfunktionen bara beräknas vid initiering eller då en parameter ändras. Processoranvändandet hos det implementerade tremolot uppgår till 1,2% av den totala kapaciteten.

4.5 Vibrato

Vibrato är i huvudsak en upprepad variation av tonhöjd, men för att efterlikna vibratot från till exempel en violinist så bör tonhöjdsvariationen kombineras med en variation i amplitud i samma takt [37], likt tremolo, se avsnitt 4.4.
Frekvensskillnaden uppstår genom att periodiskt variera en fördröjning. Detta sker på samma sätt som för flangern se avsnitt 4.1. Den stora skillnaden är att den fördröjda signalen inte mixas samman med originalet [30]. Tonhöjdsvariation uppstår genom att variera fördröjningen, men det är inte fördröjningen i sig, utan hur den ändras som påverkar tonhöjden. Tonhöjdsvariationerna beror alltså på derivatan av fördröjningen.



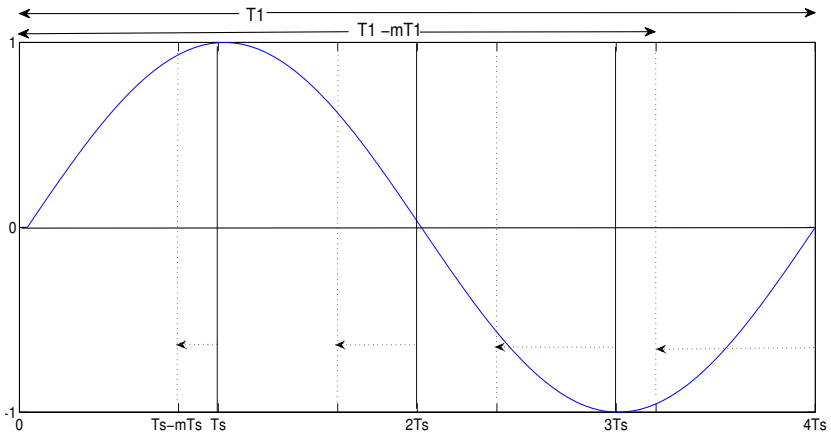
Figur 4.12: Blockmodell av ett vibrato.

När en signal spelas upp ur en buffert med den ökande fördröjningen Δd, så ökar fördröjningen i tid med en term $\Delta d T_s$ för varje sampel som hämtas. Normalt så skulle värden hämtas ur bufferten med T_s sekunders mellanrum, men istället kommer värden hämtas som annars skulle hämtats med $T_s - \Delta d T_s$ sekunders mellanrum, vilket visas i figur 4.13. Om en signal har en godtycklig periodtid $T_1 = aT_s$, så kommer nu endast $T_1 - \Delta d T_1$ av ursprungssignalen att spelas upp under en period. Resultatet av detta blir därför att signalen kommer att spelas upp långsammare än om fördröjningen hade varit konstant, vilket visas i figur 4.14. Periodtiden på signalen som spelas upp blir då istället:

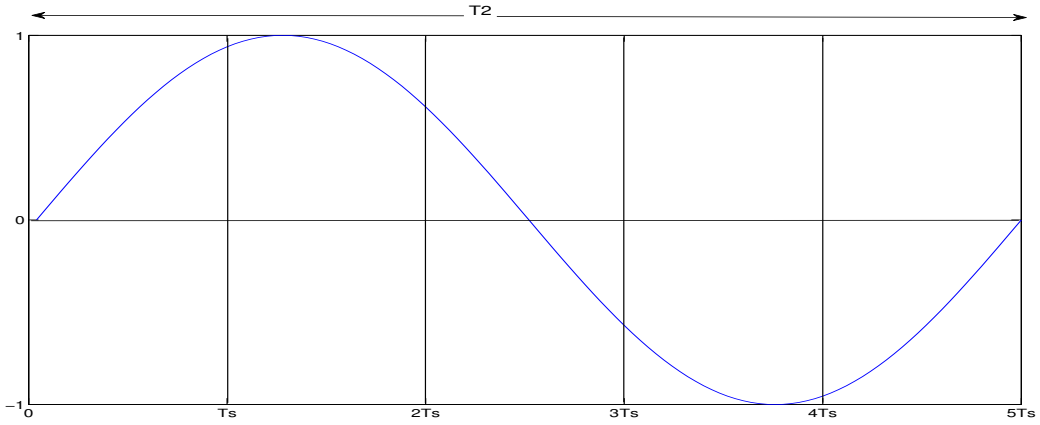
$$T_2 = \frac{T_1}{T_1 - \Delta d T_1} T_1 = \frac{1}{1 - \Delta d} T_1 \tag{4.26}$$

Och därav följer att frekvensen förändras med:

$$f_2 = (1 - \Delta d) f_1 \tag{4.27}$$



Figur 4.13: När en signal hämtas med ökande fördröjning ur en buffert, kommer värdena vid de streckade linjerna att spelas upp istället för värdena vid de heldragna.



Figur 4.14: När en signal hämtas med ökande fördröjning ur en buffer kommer periodtiden upplevas som längre

I appendix B visas att sambandet mellan två frekvenser även kan skrivas utgående från hur många kromatiska intervallsteg, D_p som f_2 skiljer sig från f_1:

$$f_2 = (\sqrt[12]{2})^{-D_p} f_1 \tag{4.28}$$

Ekvation 4.27 kan uttryckas med derivatan av fördröjningen d'[n].

$$f_2 = (1 - d'[n]) f_1 \tag{4.29}$$

Genom att kombinera ekvation 4.29 och 4.28 ovan fås sambandet mellan derivatan av tidsfördröjningen, och tonhöjdsavvikelse till

$$d'[n] = 1 - (\sqrt[12]{2})^{-D_p} \tag{4.30}$$

Variationen av tidsfördröjningen $d[n]$ följer ekvation 4.31, vilket i princip är samma ekvation som hos flanger, ekvation 4.3.

$$d[n] = \frac{d_{max}}{2}(1 - cos(2\pi \cdot f_{mod} \cdot \frac{n}{f_s})); \tag{4.31}$$

Derivatan av 4.31 är då:

$$d'[n] = \frac{d_{max}}{2} sin(2\pi \cdot f_{mod} \cdot \frac{n}{f_s}) \frac{2\pi \cdot f_{mod}}{f_s}; \tag{4.32}$$

Den högsta tonhöjdsavvikelsen inträffar vid det maximala värdet hos $d'[n]$, d'_{max}, vilket inträffar när fördröjningen ökar som mest, alltså när $n = f_s/4 \cdot f_{mod}$, vilket blir:

$$d'_{max} = \frac{d_{max} \cdot 2\pi \cdot f_{mod}}{2f_s} \tag{4.33}$$

En omskrivning av ekvation 4.33 ger:

$$d_{max} = \frac{d'_{max} f_s}{\pi \cdot f_{mod}} \tag{4.34}$$

Vilket tillsammans med ekvation 4.30 ger

$$d_{max} = \frac{(1-k)f_s}{\pi \cdot f_{mod}} = \frac{(1 - (\sqrt[12]{2})^{-D_p})f_s}{\pi \cdot f_{mod}} \tag{4.35}$$

Med ekvation 4.35 fås rätt värde på d_{max} att använda till den varierande tidsfördröjningen i ekvation 4.31. I ett mycket enkelt vibrato hade användaren kunnat styra tonhöjdsvariationen genom att direkt välja värde på d_{max}. Men eftersom tonhöjdsvariationen beror även på hastigheten som tonhöjden varierar med, så kan ett mer intuitivt och lättstyrt vibrato åstadkommas genom att använda ekvation 4.35. Tonhöjdsvariationen och hastigheten är oberoende av varandra i detta vibrato, och styrs alltså med parametrarna f_{mod} och D_p. Det ger även ett vibrato med god kontroll över tonhöjdsvariationen, eftersom det går att välja hur stor del av ett kromatiskt intervallsteg (halvton) som den skall vara. Det går att designa ett vibrato där tonhöjden varierar flera halvtoner, men för att efterlikna ett naturligt vibrato krävs mindre än så. Vibratot från en violinist ligger till exempel på runt en sjättedel av en halvton [37].

Implementering av vibrato följer tillvägagångssättet i flanger, med Interpolering utförs mellan de tidsfördröjda värden som hämtas ur en buffert. Storleken på den buffert som krävs måste därför vara minst lika stor som det största förekommande värdet på d_{max}. Vilket beräknas med ekvation 4.35 utgående från tillåtna värden för inparametrarna f_{mod} och D_p. För $f_{mod} \geq 1$ Hz och $D_p \leq 1$ (halvton) krävs därför en buffertstorlek på minst 448 vid en samplingsfrekvensen $f_s = 48$ kHz.

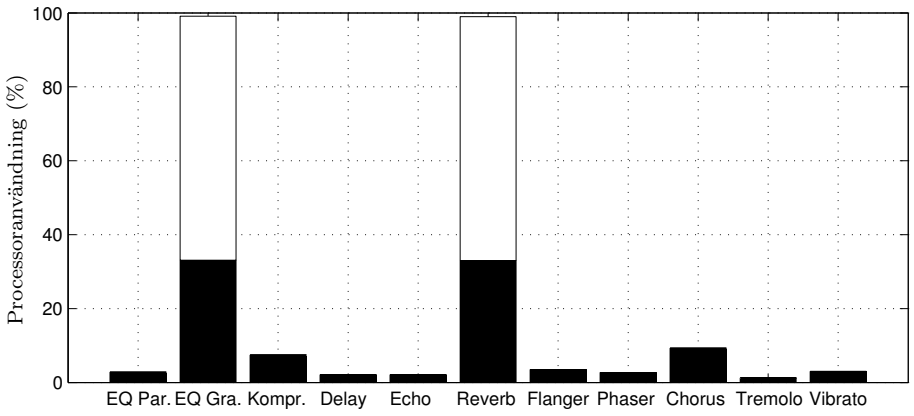
Utöver variationen av tonhöjd läggs det även på en variation i amplitud, med samma form som ett sinusformat tremolo, se avsnitt 4.4. Den maximala dämpningen av amplitud styrs av en parameter, och en annan styr skillnaden i fas mellan amplituddämpningen och tonhöjdsvariationen. Effekten belastar processorn med cirka 3%.

5 Diskussion

I rapporten beskrivs hur tio musikeffekter designats och implementerats. De olika effekterna täcker upp ett brett användingsområde och förekommer i flera avancerade ljudbehandlingsprogram.
Samtliga musikeffekter har utformats med avseende på tre faktorer: funktion, prestanda och styrbarhet.
Med funktion avses hur de framtagna effekterna påverkar ett ljudstycke. Deras funktion har verifierats genom provlyssningar, simuleringar i MATLAB och frekvensanalys av utgående ljud från utvecklingskortet.
Med prestanda avses hur de olika effekterna belastar systemet, de i inledningen statuerade kravspecifikationerna uppfylls av samtliga effekter. Detta har verifierats genom att undersöka processorbelastningen för varje enskild effekt.
Med styrbarhet avses hur en användare kan kontrollera effekternas verkan genom att ställa in ett antal välkända parametrar.

Implementationen i signalprocessorn ledde till många problem där nästan samtliga grundades i processorns oförmåga att beräkna flyttal på ett effektivt sätt. Fixtalsimplementation ger problem med upplösning och överskridning, och flyttal med belastning av signalprocessorn.
En signalprocessor med flyttalsenhet hade underlättat implementationen av flera effekter. Till exempel är olinjära operationer, såsom logaritmiska och trigonometriska uttryck, mycket lättare att implementera. Dessa kan i en fixtalsimplementation med fördel approximeras med taylorutveckling.

Belastningen av signalprocessorn beaktades ständigt i effekternas utformning, eftersom funktionaliteten inte fick bli lidande.
Processorbelastningen för de olika musikeffekterna, som presenterades fortlöpande i rapporten, sammanställs i figur 5.1 nedan. De svarta staplarna representerar de implementerade effekternas processoranvändning. De vita staplarna representerar hur effekterna skulle kunna ha utformats för att maximera prestanda.



Figur 5.1: Processorbelastning för de olika effekterna.

Musikeffekten equalizer har designats i två olika utföranden, detta eftersom de båda varianterna har olika användningsområden.

Samtliga musikeffekter uppfyller det krav på processoranvändning som specificerades i inledningen. Tidsfördröjningen i systemet kan uppskattas till 10.7 ms vilket uppfyller kravet om maximal tidsfördröjning på 20 ms som ställdes i inledningen. Detta värde valdes för att musikeffekterna

skulle kunna användas i en tänkt livespelning utan att fördröjningen blir påtaglig för lyssnaren. Av figuren framgår att inte ens då samtliga musikeffekter körs i serie överbelastas processorn, vilket innebär att samtliga musikeffeker samtidigt skulle kunna tillämpas på ett ljudspår.
De flesta effekters belastning av signalprocessorn är inte särskilt stor, undantagen är reverb och grafiska equalizer. För dessa båda finns det dessutom ett värde i att utforma dem för att till fullo nyttja signalprocessorns beräkningskraft.

Den grafiska equalizern är den enda effekt som kräver att block av data behandlas åt gången och är således den enda effekt som introducerar längre tidsfördöjningar i systemet. Processoranvändningen för denna skulle minska om längre tidsfördöjningar tillåts.
FFT-algoritmens längd, $N = 16384$, begränsades av minnesutrymmet i internminnet. Om signalprocessorn däremot skulle dedikeras enbart till den grafiska equalizern skulle längden kunna ökas genom att spara data i SDRAM och använda EDMA-modulen för att hämta det. De olika banden hade då kunnat göras skarpare.

Antalet nollskilda koefficienter för det med speglingsmodellen beräknade impulssvaret i effekten reverb, kan enkelt ökas genom att höja tröskelnivån och sålla bort färre koefficienter. Detta skulle då ge en exaktare rumsåtergivning men också öka belastningen eftersom längden av FIR-filtret växer.

De i rapporten behandlade musikeffekter valdes eftersom de utgör grunden för många andra musikeffekter. Vid valet undersöktes ett brett urval av kommersiella effekter, vilket gjordes för att få en god översikt och förståelse av deras användningsområde, struktur och funktionalitet. För att de framtagna effekterna skulle vara så intuitiva som möjligt, följdes i största mån de redan definierade standarder vad gäller styrbara parametrar och inställningsmöjligheter.

Utifrån de musikeffekter som redan utvecklats hade, med mindre modifieringar och påbyggnader, flera andra kunnat konstrueras.
Till exempel hade equalizern kunnat byggas vidare till en *Wah Wah* som är ytterligare en välanvänd effekt. Utifrån kompressorn hade flera olika distorsioneffekter såväl som en *noise gate* kunnat skapats. Kompressorn i kombination med den parametriska equalizern hade kunnat utgöra en *De-esser*, som reducerar s-ljud vid tal.

6 Slutsats

Tio olika musikeffekter har modellerats och implementerats. Dessa är: equalizer, kompressor, delay, echo, reverb, flanger phaser, chorus, tremolo och vibrato. Equalizern har presenterats i två utföranden, parametrisk och grafisk.

Att vara begränsad till fixtal inskränker inte möjligheten att tillämpa signalbehandling i syfte att skapa musikeffekter.
Musikeffekterna har implementerats för att köras i realtid på utvecklingskortet C6416T DSK som utformats kring signalprocessorn TMS320C6416T DSP. Ingen av effekterna belastar processorn mer än 33.3%, vilket innebär att flera effekter kan appliceras på ett ljudspår samtidigt.
För att kunna använda den grafiska equalizern krävs att block av sampel behandlas åt gången vilket introducerar en tidsfördöjning i systemet som uppskattades till 10.7 ms.

Systemens funktion har verifierats med simuleringar, provlyssningar och mätningar. Samtliga effekters verkan på ett ljudspår kan styras av vanligt förekommande parametrar inom musikbehandling.

Källförteckning

[1] R. Izhaki, *Mixing Audio (Second Edition).* Boston: Focal Press, 2012, p. 202.

[2] R. Chassaing och D. Reay, *Digital Signal Processing and Applications with the TMS320C6713 and TMS320C6416 DSK.* Hoboken, New Jersey: WILEY-Interscience, 2008.

[3] *TMS320C64x/C64x+ DSP CPU and Instruction Set*, Texas Instruments, Dallas, Texas, Jul. 2010.

[4] N. Kehtarnavaz, *Real-Time Digital Signal Processing Based on the TMS320C6000.* Oxford, England: Elsevier, 2005.

[5] *TTLV320AIC23 Stereo Audio CODEC, 8- to 96-kHz, With integrated headphone amplifier*, Texas Instruments, Dallas, Texas, Maj 2002.

[6] A. V. Oppenheim, A. S. Willsky och S. Hamid, *Signals and Systems.* Upper Saddle River, New Jersey: Prentice-Hall, 1996.

[7] J. D. och K. W. Johansson, *Physics.* New York: WILEY-Interscience, 1998, p. 466.

[8] S. M. Kuo, B. H. Lee och W. Tian, *Real-Time Digital Signal Processing.* West Sussex, England: John Wiley and Sons, 2008.

[9] J. G. Proakis och D. G. Manolakis, *Digital Signal Processing - Principles, Algorithms, and Applications.* Upper Saddle River, New Jersey: Prentice-Hall, 2007.

[10] U. Zölzer, *Digital Audio Signal Processing.* West Sussex, England: John Wiley and Sons, 2008.

[11] B. Karlström, "Signalbehandling," Göteborg, Mars 2011.

[12] A. V. Oppenheim och R. W. Schafer, *Discrete-Time Signal Processing.* Upper Saddle River, New Jersey: Prentice-Hall, 1999.

[13] B. Mulgrew, P. Grant och J. Thompson, *Digital Signal Processing - Concepts and applications.* Hampshire, England: Palgrave MacMillan, 2003.

[14] E. Ström, "The Discrete Fourier Transform," Göteborg, August 2000.

[15] A. V. Oppenheim och R. W. Schafer, *Discrete - Time Signal Processing.* Pearson Education, Inc., 2010.

[16] J. McJunkin, "Digital equalizers," *Sound & Video Contractor*, vol. 22, pp. 40–44, April 2004.

[17] R. Izhaki, *Mixing Audio (Second Edition).* Boston: Focal Press, 2012, pp. 266–330.

[18] U. Zölzer, *Digital audio signal processing.* Hamburg, Tyskland: J. Wiley and Sons, 2008, pp. 225–234.

[19] S. McKinley och Megan Levine, "Cubic spline interpolation," http://online.redwoods.cc.ca.us/instruct/darnold/laproj/Fall98/SkyMeg/Proj.PDF.

[20] U. Zölzer, *DAFX: Digital Audio Effects.* Hamburg, Tyskland: J. Wiley and Sons, 2011.

[21] K. C. Pohlmann, *Principles Of Digital Audio.* Mc-Graw Hill, april 2005.

[22] F. A. Everest, *The Master Handbook Of Acoustics.* McGraw-Hill, Inc., 1994.

[23] V. Granbom, "Digital frequency compensation for loudspeakers used in rooms," Master's thesis, Chalmers University of Technology, 2010.

[24] D. Highfield, *Room Acoustics.* Routledge, 2000, pp. 59–114,234–277.

[25] "Concert hall impulse responses — pori, finland: Reference," Helsinki University of Technology, 2005, http://www.acoustics.hut.fi/projects/poririrs/ [2011-04-26].

[26] McGill University, 2004-2010, http://www.music.mcgill.ca/ gary/307/week3/reverb.html, [2012-04-04].

[27] M. Karjalainen och H. Järveläinen, "Reverberation modeling using velvet noise," in *Audio Engineering Society Conference: 30th International Conference: Intelligent Audio Environments*, mar. 1997.

[28] E. A. P. Habets, "Room impulse response generator," London, September 2010.

[29] J. B. Allen och D. A. Berkley, "Image method for efficiently simulating small-room acoustics," *Acoustical Society of America*, vol. 65, pp. 943–950, Juni 1979.

[30] R.G. Keen, "The technology of phase shifters and flangers," 1999, http://www.geofex.com/Article_Folders/phasers/phase.html [2012-05-01].

[31] J. O. Smith, "Physical audio signal processing, flanging," https://ccrma.stanford.edu/ jos/-pasp/Flanging.html [2012-05-01].

[32] S. J. Orfanidis, *Introduction to Signal Processing.* Pearson Education, Inc., 2010.

[33] L. Råde och B. Westergren, *Mathematics Handbook for Science and Engineering.* Studentlitteratur, 2005.

[34] J. O. Smith, "Physical audio signal processing, phasing," https://ccrma.stanford.edu/ jos/-pasp/Phasing.html [2012-05-01].

[35] C. Tseng, "Stable iir notch filter design with optimal pole placement," *IEEE Transactions on Signal Processing*, vol. 49, pp. 2673–2681, November 2001.

[36] R. Izhaki, *Mixing Audio (Second Edition).* Boston: Focal Press, 2012, pp. 399ñ–400.

[37] J. O. Smith, "Physical audio signal processing, vibrato simulation," https://ccrma.stanford.edu/ jos/pasp/Vibrato_Simulation.html [2012-05-10].

[38] C. Nordling och J. Österman, *Physics Handbook for science and engineering.* Studentlitteratur AB, Lund, Sweden, 2009.

[39] J. W. Jewett och R. A. Serway, *Physics For Scientists and Engineers With Modern Physics.* David Harris, 2008.

[40] K. Eriksson, D. Estep, P. Hansbo och C. Johnson, *Computational Differential Equations.* Studentlitteratur, 1996, pp. 423–451.

A FFT (Fast Fourier Transform

Diskret fouriertransform (DFT) av en diskret signal, $x[n]$, ges av:

$$X(k) = \sum_{n=0}^{N-1} x[n]W_N^{nk}, \quad k = 0, 1, ..., N-1 \tag{A.1}$$

Där W betecknar fasvektorn som ges av:

$$W_N = e^{-j\frac{2\pi}{N}}. \tag{A.2}$$

Beräkning av uttrycket ovan resulterar i N^2 komplexa multiplikationer och $N^2 - N$ komplexa additioner vilket för stora N innebär en oerhört kostsam operation.
FFT-algoritmen utnyttjar att fasvektorn är både periodisk och symmetrisk, det vill säga

$$W_N^{k+N} = W_N^k \tag{A.3}$$

och

$$W_N^{k+N/2} = -W_N^k. \tag{A.4}$$

Sambanden kan utnyttjas för att bryta ner en N sampels DFT i mindre beståndsdelar. Om uttrycket i (A.1) delas upp i två summor enligt

$$X(k) = \sum_{n=0}^{N/2-1} x[n]W_N^{nk} + \sum_{n=N/2}^{N-1} x[n]W_N^{nk} \tag{A.5}$$

ges följande uttryck genom substitutionen $n = n + N/2$ i den andra summan:

$$X(k) = \sum_{n=0}^{N/2-1} x[n]W_N^{nk} + e^{-j\pi k} \sum_{n=0}^{N/2-1} x[n + \frac{N}{2}]W_N^{nk} = \sum_{n=0}^{N/2-1} (x[n] + (-1)^k x[n + \frac{N}{2}])W^{nk} \tag{A.6}$$

Nästa steg är att dela upp signalen i jämn och udda del, vilket ger:

$$X(2k) = \sum_{n=0}^{N/2-1} (x[n] + x[n + \frac{N}{2}])W_{N/2}^{nk} \tag{A.7}$$

och

$$X(2k+1) = \sum_{n=0}^{N/2-1} (x[n] - x[n + \frac{N}{2}])W_N^n W_{N/2}^{nk} \tag{A.8}$$

Genom att tecka $a[n] = x[n] + x[n + \frac{N}{2}]$ samt $b[n] = x[n] - x[n + \frac{N}{2}]$ kan ekvationerna ovan skrivas om som:

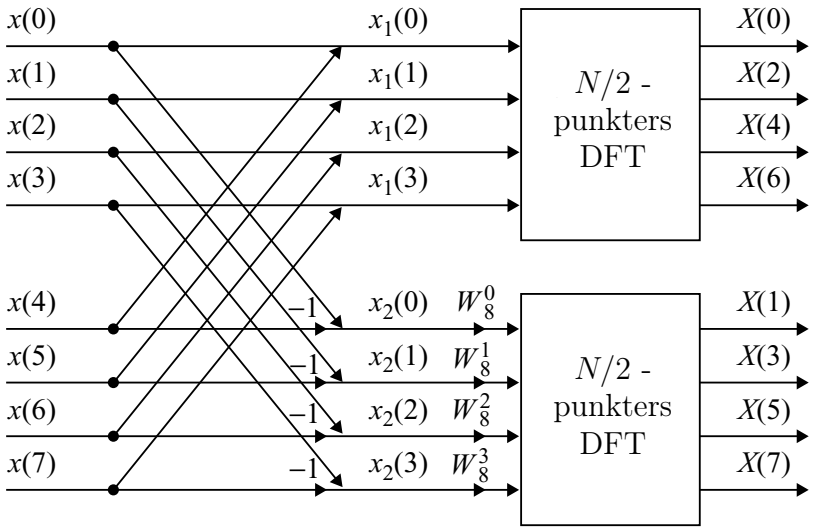
$$X(2k) = \sum_{n=0}^{N/2-1} a[n]W_{N/2}^{nk} \tag{A.9}$$

och

$$X(2k+1) = \sum_{n=0}^{N/2-1} b[n]W_N^n W_{N/2}^{nk} \tag{A.10}$$

Ekvationerna A.9 och A.10 är skrivna på samma form som uttrycket för DFT ovan med skillnaden att antalet sampels nu är $N/2$. Detta innebär att stegen ovan kan upprepas även för dessa serier. När signalen har brutits ner till segment av lämplig längd är processen utförd. Längden på de minsta sekvenserna betecknar algoritmens radix. Nedbrytningsprocessen för algoritmen

med radix 4 illustreras av figur A.1. Denna variant av algoritmen kallas för DIF (Decimation In Frequency) då det är utsignalen som brutits ner i mindre sekvenser [2]. Notera att utsignalen inte kommer i sekvensiell ordning.



Figur A.1: FFT med radix 4 av en 8 sampel lång insignal med reducering i frekvens.

B Musikteori

Varje signal kan identifieras som en multipel av en eller flera grundfrekvenser. I ljudsammanhang kallas den mest intensiva (dominanta) multipeln av grundfrekvensen för en ton [21].

I den västerländska musikteorin finns det totalt tolv grundtoner som upprepas längs frekvensskalan. Att utgå från en ton och höja tolv steg kallas för att gå upp en oktav, där en oktav alltså är det frekvensintervall som motsvarar tolv halvtonsteg. Genom att använda oktaver kan alla toners upprepningar längs frekvensspektrummet beskrivas med hjälp av grundtonen. Utgående från referenstonen $f_0 = 440\ Hz$ kan samtliga toners motsvarande dominanta grundfrekvens beräknas enligt:

$$f_n = \sqrt[12]{2^n} f_0 \tag{B.1}$$

där n betecknar antalet intervallsteg från referenstonen enligt ovan [38]. Således höjer man referenstonen en oktav genom att gå upp tolv steg:

$$f_{12} = \sqrt[12]{2^{12}} f_0 = 2f_0 = 880\ Hz. \tag{B.2}$$

På motsvarande sätt går man ner en oktav genom att sänka tolv steg:

$$f_{-12} = \sqrt[12]{2^{-12}} f_0 = \frac{1}{2} f_0 = 220\ Hz. \tag{B.3}$$

Genom att beteckna $A^1 = f_0$ bestäms det musikaliska alfabetet där varje bokstav representeras av ett intervallsteg. De olika oktaverna betecknas genom olika index enligt tabell B.1.

Tabell B.1: Toner och deras motsvarande frekvenser över tre oktaver.

Ton	Frekvens $[Hz]$	Ton	Frekvens $[Hz]$	Ton	Frekvens $[Hz]$
A_1	220	A^1	440	A^2	880
$A_1^{\#}/B_1^b$	233	$A^{\#1}/B^{b1}$	466	$A^{\#2}/B^{b2}$	932
B_1	247	B^1	494	B^2	988
C_1	262	C^1	523	C^2	1047
$C_1^{\#}/D_1^b$	277	$C^{\#1}/D^{b1}$	554	$C^{\#2}/D^{b2}$	1109
D_1	294	D^1	587	D^2	1175
$D_1^{\#}/E_1^b$	311	$D^{\#1}/E^{b1}$	622	$D^{\#2}/E^{b2}$	1245
E_1	330	E^1	659	E^2	1319
F_1	349	F^1	698	F^2	1397
$F_1^{\#}/G_1^b$	370	$F^{\#1}/G^{b1}$	740	$F^{\#2}/G^{b2}$	1480
G_1	392	G^1	784	G^2	1568
$G_1^{\#}/A_1^b$	415	$G^{\#1}/A^{b1}$	831	$G^{\#2}/A^{b2}$	1661

C Panorering

Stereo innebär att två signaler spelas samtidigt, fast i två kanaler. Ett stereokonfigurerat system kan därför anpassas så att ljudet spelas på vänster- eller höger högtalare. Ljudet kan också blandas ut mellan dem, vilket kallas *panorering*.
Panoreringen styrs genom att signalen placeras på ett gradtal, θ, inom $-30^\circ \leq \theta \leq 30^\circ$ där $\theta = 0^\circ$ motsvarar att signalen spelas lika mycket i båda högtalarna [20].

Relationen mellan respektive kanalförstärkning och gradtal fås av Bennetts tangentlag

$$\frac{\tan(\theta)}{\tan(\theta_{bas})} = \frac{g_1 - g_2}{g_1 + g_2} \tag{C.1}$$

där g_1 och g_2 motsvarar förstärkningen för respektive kanal och $\theta_{bas} = 30^\circ$ är den största vinkeln θ kan anta [20]. Genom att sätta $g_1 = 1$ kan g_2 beräknas enligt ekvation C.1.

$$g_2 = -\frac{\tan(\theta) - \tan(\theta_{bas})}{\tan(\theta) + \tan(\theta_{bas})}. \tag{C.2}$$

Dessutom skall ljudet normaliseras för den totala effekten skall vara oförändrad. Vilket innebär att g_1 och g_2 med

$$D = \sqrt{g_1^2 + g_2^2} \tag{C.3}$$

D Vågekvationen

Ljud kan beskrivas som tryckvågor som utbreder sig som plana vågor i ett medium [39]. I ett luftfyllt rum, vars sidor är av betydligt tätare medium än luft kan ljudets rörelse beskrivas av vågekvationen:

$$\nabla^2 p(\vec{r},t) - \frac{1}{v^2}\frac{\partial^2 p(\vec{r},t)}{\partial t^2} = -s(\vec{r},t) \tag{D.1}$$

där $p(\vec{r},t)$ beskriver ljudtrycket vid $\vec{r} = (x,y,z)$ som en funktion av tiden, v är ljudets hastighet och $s(\vec{r},t)$ är källfunktionen [40]. Genom att studera vågekvationen i frekvensdomänen ges

$$\nabla^2 P(\vec{r},\omega) + k^2 P(\vec{r},\omega) = -S(\vec{r},\omega) \tag{D.2}$$

där $P(\vec{r},\omega)$ och $S(\vec{r},\omega)$ är ljudtryck och källfunktion uttryckta i frekvensdomän och där k är vågtalet:

$$k = \frac{\omega}{v} = \frac{2\pi}{\lambda} \tag{D.3}$$

Genom att sätta källan till en impuls fås impulssvaret:

$$\nabla^2 P(\vec{r},\omega) + k^2 P(\vec{r},\omega) = -\delta(\vec{r} - \vec{r}_s) \tag{D.4}$$

Där $\vec{r}_s = (x_s, y_s, z_s)$ är källans rumskoordinater. Uttrycket medför att ekvationen kan betraktas som ett egenvärdesproblem och kan lösas för olika moder, där varje mod representerar en stående våg i rummet [24].