CODEN:LUTEDX/(TEIE-3088)/1-62(2019)

Industrial Electrical Engineering and Automation

Bandbreddsbegränsad modellering av analog sågtandsoscillator för digital implementering

Jonas Persson

Division of Industrial Electrical Engineering and Automation
Faculty of Engineering, Lund University

LUNDS UNIVERSITET

Lunds Tekniska Högskola

Bandbreddsbegränsad modellering av analog sågtandsoscillator för digital implementering

Jonas Persson
IDA3, Datateknik

Kurskod: EIEL05
Handledare: Mats Lilja
Examinator: Morten Hemmingsson

14 oktober 2019

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

Printed in Sweden
Lunds universitet
Lund 2019

Sammanfattning

Idag finns mängder av kommersiella VST-syntar som modellerar klassiska analoga syntar men det finns lite publik forskning på området. Anledningen till att det är av intresse att efterlikna klassiska analoga syntar är att deras oscillatorers grundvågformer sällan är perfekta. Denna imperfektion ger anledning att undersöka och digitalt implementera klassiska analoga syntar för att ta reda på vad det är som karakteriserar olika analoga syntar samt försöka återskapa denna karaktär. Ett oundvikligt problem vid digital implementation av analoga syntar är att ta fram en oscillatoralgoritm som genererar signaler utan vikning i ett audiospektrum som är musikaliskt intressant.

Detta examensarbete har realiserats i tre faser. Den första fasen innebar implementation av en VST-synt samt vikningsreducering av denna synts sågtandsoscillator. Under den andra fasen analyserades fyra analoga syntars sågtandsvågformer och under den tredje fasen togs det fram en ad-hoc modellering av en dessa, nämligen Roland SH-101's sågtandsvågform.

Tidsbegränsningen av detta examensarbete medförde att resultatet av vikningsreduceringen inte lyckades uppfylla examensarbetets mål. Vad gäller egenskaper hos de analyserade analoga sågtandsvågformerna visar examensarbetets resultat på ett gemensamt karaktärsdrag. I frekvensdomän har alla fyra analyserade analoga sågtandsvågformer en snabbare avtagande amplitud hos dess övertoner än en ideal sågtandsvågform. Detta betyder att man kan beskriva de analyserade syntarnas sågtandsvågformer som lågpassfiltrerade ideala sågtandsvågformer.

Nyckelord— synt, oscillator, vikning, BLEP, modellering

Abstract

There are alot of commercial VST-synthesizers that models classic analog synthesizers today but there is little public research available on the topic. The main reason that it is of interest to model classic analog syntesizers is that their oscillator waveforms rarely have a perfect standard waveform. This imperfection gives justification to research and digital implementation of these classic analog synthesizers to find out what characterizes different the analog synthesizers oscillators and also try to recreate this character. An inevitable problem when digitally implementing an analog sawtooth waveform is to produce an oscillator algorithm that generates signals without aliasing in a musically interesting frequency spectrum.

This thesis have been carried out in three phases. During the first phase a VST-synthesizer with reduced aliasing was implemented. During the second phase four analog sawtooth waveforms where analyzed and throughout the third and last phase an ad-hoc method for modeling Roland SH-101's sawtooth oscillator was developed.

Due to the time limitation of this thesis the goals for the alias reduction weren't met. When it comes to the analysis of analog waveforms one conclusion was made. In the frequency domain all four analysed analog sawtooth waveforms had a faster decay in amplitude of the signals overtones compared to an ideal sawtooth waveform. This means that the analysed sawtooth waveforms can be described as lowpass filtered ideal sawtooth waveforms.

Keywords— synthesizer, oscillator, aliasing, BLEP, modeling

Innehåll

1 Inledning

I detta kapitel ges en introduktion till detta examensarbete genom en beskrivning hur examensarbetet uppkommit, vilket syfte examensarbetet har samt motivering av examensarbetet.

1.1 Bakgrund

Denna uppgift har uppkommit med anledning av universitetslektor Mats Lilja (LTH) och mitt gemensamma intresse för musikinstrumentet synt. Efter många timmars möten, inläsning på området samt flertalet diskussioner med Mats Lilja som också är handledare för detta examensarbete kom vi fram till att examensarbetet ska behandla vikningsreducering och digital implementation av analog synt. Examensarbetet görs för Avdelningen för Industriell elektroteknik och automation, Lunds Tekniska Högskola.

I detta examensarbete kommer fokus ligga på att dimplementera en VST-synt (mjukvarusynt som insticksprogram) där en implementation av en analog synts sågtandsvågform sedan integreras.

Ett oundvikligt problem vid digital implementation av en analog sågtandsvågform är att ta fram en oscillator-algoritm som uppfyller följande tre kriterier (Pekonen 2014):

1. genererar signaler utan vikning i ett audiospektrum som är musikaliskt intressant, t ex 20 Hz till 8000 Hz,
2. är beräkningsmässigt effektiv och har låg minnesåtgång samt
3. inte kräver division av en tidsvarierande parameter, som t ex grundfrekvensen.

Undersökning har alltså gjorts på området tidigare men ingen metod har hittats som samtidigt uppfyller ovanstående egenskaper.

För att begränsa urvalet av analoga syntar kommer några syntar vars oscillator bygger på den integrerade kretsen CEM3340 analyseras och slutligen kommer en av dessa modelleras och implementeras. CEM3340 är en Voltage Controlled Oscillator (VCO) med möjlighet till utmatning av fyra vågformer: sågtand, fyrkant, puls och triangel. CEM3340 används i många analoga syntklassiker som till exempel Sequential Circuits Prophet 5 Rev 3 (1978), Roland SH-101 (1983) och Moog Memorymoog (1982) för att nämna några.

1.2 Syfte

Syftet med att modellera vågformens karaktär hos någon analog synt är att analoga syntars vågformer skiljer sig från ideala grundvågformer (Pekonen 2014). De analoga vågformerna låter sällan lika skarpa som de ideala vågformerna. Vidare har endast MiniMoog Voyager's sågtandsvågform modellerats i tidigare forskning (Pekonen 2014). Det är alltså rimligt att ytterligare undersökning på området görs.

En avsikt med examensarbetet var att försöka lösa problemet med att ta fram en oscillatoralgoritm som uppfyller de tre kriterier som beskrivs i bakgrund. Med anledning av mina begränsade förkunskaper inom digital signalbehandling samt examensarbetets tidsbegränsning var detta ett problem som tyvärr inte löstes under detta examensarbete.

En ytterligare förhoppning är att detta examensarbete ska kunna användas som referens vid framtagning av en mjukvarusynt där stor vikt läggs vid att modellera och implementera någon analog synts sågtandsvågform.

1.3 Motivering av examensarbetet

Den största personliga motivationen till examensarbetet har varit att jag fått kombinera några av mina favoritområden musik, synt, programmering och matematik. Dessutom har jag fått möjlighet att lära mig mer om digital signalbehandling som är ett ämne jag länge varit intresserad av.

En annan motivering är att det idag finns mängder av kommersiella VST-syntar som efterliknar klassiska analoga syntar men det finns lite publik forskning på området. En anledning till att det är av intresse att efterlikna klassiska analoga syntar är att deras oscillatorers grundvågformer sällan är perfekta. Denna imperfektion ger anledning att undersöka och digitalt implementera klassiska analoga syntar för att ta reda på vad det är som karakteriserar olika analoga syntars oscillatorer samt försöka återskapa denna karaktär.

Implementation av just sågtandsvågformen är intressant på det sätt att sågtandsvågformen innehåller flest övertoner av grundvågformerna. Mängden övertoner bidrar till att det är den grundvågform som vid lägst frekvens ger upphov till vikning enligt Nyquistteoremet. Det är alltså den vågform som ger upphov till mest problem med vikning.

1.4 Målformulering

Examensarbetets mål är att:

- undersöka karaktären hos ett antal sågtandsvågformer från några analoga syntar som använder sig av den integrerade kretsen CEM3340,
- skapa en matematisk modell av en dessa vågformer,
- digitalt implementera samma vågform,
- implementera en VST-synt för att sedan
- integrera implementationen som oscillator i denna VST-synt.

Målsättningen är att digital implementation av analog vågform ska:

- vara fri från vikning i ett audiospektrum som är musikaliskt intressant,
- vara beräkningsmässigt effektiv samt
- ha låg minnesåtgång

Målsättningen är att VST-synten ska ha:

- ett enkelt grafiskt användargränssnitt samt
- en sågtandsoscillator

1.5 Problemformulering

Vid implementation av VST-synt samt modellering och implementation av oscillator i detta examensarbete finns några frågor som har för avsikt att besvaras, nämligen:

1. Vad karakteriserar olika analoga syntars sågtandsvågformer?
2. Vad finns det för metoder för att bandbreddsbegränsa en analog signal med syfte att minimera vikningseffekten som uppstår när signalen samplas och hur kan dessa metoder förbättras?

3. Vilket ramverk är lämpligast när VST-synten implementeras för att mest fokus ska ligga på att ta fram algoritmer för oscillator och mindre fokus på grafiskt användargränssnitt?

1.6 Avgränsningar

Fokus kommer att ligga på att modellera en analog sågtandsvågform, modellering av resterande grundvågformer utelämnas på grund av risken att det skulle bli för tidskrävande.

Med anledning av att det saknas tillgång till analoga syntar kommer mätning inte kunna göras direkt från någon fysisk synt. Mätning kommer istället göras från ljudfiler som finns tillgängliga på internet. En nackdel med att göra mätningar från ljudfiler är att kontinuerliga signaler från en analog synt har omvandlats till diskret digital information i en ljudfil. Under denna omvandlingsprocess kan tyvärr sampling och kvantisering introducera felaktig information i den digitala signalen.

En ytterliggare begränsning är att inga vetenskapliga hörselstudier av examensarbetets resultat kommer att göras, även denna gång är motiveringen att det skulle bli för tidskrävande.

En aspekt som det inte tas hänsyn till i detta examensarbete är den digitala till analoga omvandling som sker i en dators ljudkort. Denna digitala till analoga omvandling fungerar som ett filter och beroende på frekvens på ljud och samplingsfrekvens kan denna omvandling komma att färga ljudet som examensarbetets VST-synt genererar.

2 Teknisk bakgrund

I detta kapitel beskrivs den grundläggande tekniska bakgrund som anses nödvändig för resten av rapportens innehåll.

2.1 Synt

En synt eller synthesizer är ett elektroniskt musikinstrument som genom någon form av källa, t ex oscillator eller brusgenerator genererar ljud. Det finns både analoga och digitala syntar. Analoga syntar generar ljud med hjälp av analog elektronik vilket medför att de producerar en kontinuerlig analog signal. Digitala syntar generar ljud genom digital signalbehandling vilket innebär att de producerar en diskontinuerlig signal med diskreta punkter i tiden.

För att förenkla kan man säga att syntar typiskt består av tre komponenter (Pirkle 2014). En eller flera källor som genererar ljudsignaler, en enhet som modifierar ljudet, t ex filter och effekter samt en enhet som styr källornas och/eller modifieringsenheternas parametrar över tid, t ex en low frequency oscillator (LFO). Det finns en mängd olika metoder för syntetisera ljud, additiv och subtrativ syntes är exempel på två av dessa metoder. Majoriteten av de första syntarna från 60- och 70-talet där ljudkällan bestod av analog elektronik använde sig av subtrativ syntes (Pekonen och Välimäki 2011).

2.1.1 Subtraktiv vs. additiv ljudsyntes

Med additiv ljudsyntes syntetiseras önskat ljud genom att addera sinusvågor av olika amplitud och frekvens. Subtraktiv syntes kan ses som motsatsen, med denna metod genereras först periodiska grundvågformer med rikt frekvensinnehåll som t ex en sågtand eller fyrkantsvågform, dessa vågformer kan sedan mixas ihop för mer komplexitet. Slutligen filteras dessa signaler ofta med ett lågpass filter för att uppnå önskat ljud (Pekonen och Välimäki 2011). Man filterar alltså bort frekvensinnehåll, därav namnet subtraktiv syntes.

2.1.2 Analog sågtandsoscillator

En analog sågtandsoscillator kan beskrivas matematisk av $f(t) = t \mod 1$.

2.1.3 Digital sågtandsoscillator

En enkel design av en digital sågtandsoscillator kan realiseras med Kodexempel 1.

```
1 double sawtooth(){
2     //f = grundfrekvens i Hz, fs = samplingsfrekvens i Hz
3     double modulo += f/fs;
4
5     if(modulo >= 1.0)
6         modulo -= 1.0;
7
8     return modulo;
9 }
```

Kodexempel 1 – Simpel sågtandsoscillator

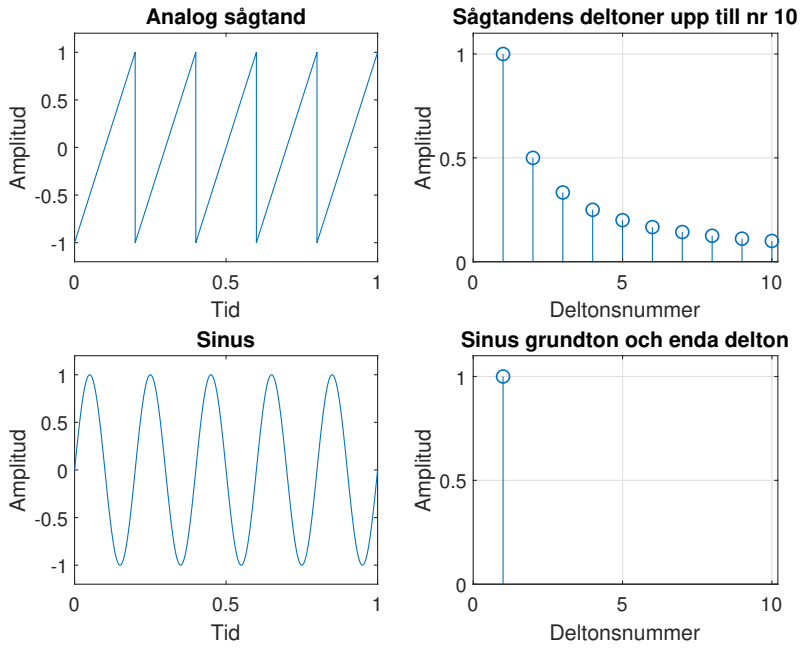
Denna enkla implementation som kallas för ideal sågtand orsakar omfattande vikning, se Figur 2.

2.2 Vikning

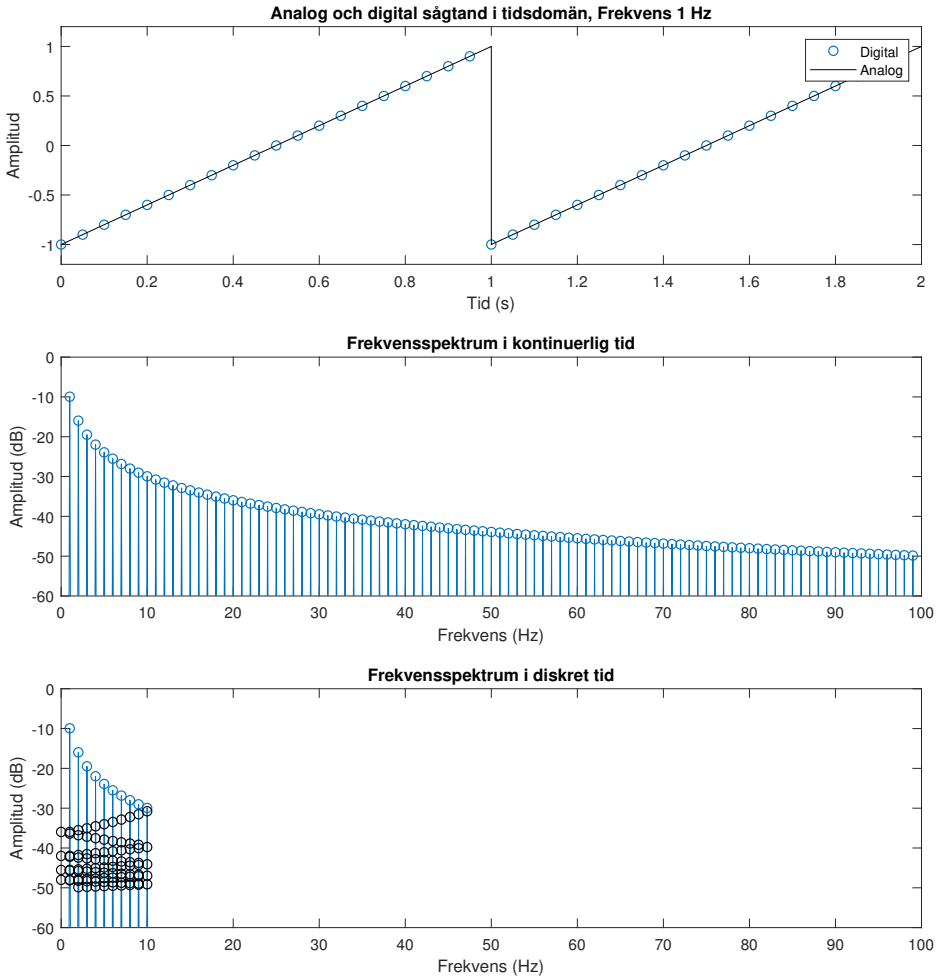
En analog signal med diskontinuitet i vågformen, t ex en sågtandsvågform, måste bandbegränsas (filteras) till under hälften av samplingsfrekvensen, med andra ord till under Nyquistfrekvensen för att erhålla en motsvarande diskret signal (Stilson och Smith 1996).

Figur 1 ger en illustration av en analog sågtand och dess deltoner upp till 10, samt jämförelse med en sinusvågform. Till skillnad från en sinusvåg som endast har en delton (dess grundton/grundfrekens), har sågtanden oändligt många deltoner, en för varje heltalsmultipel av grundtonen. En deltons amplitud är proportionell mot inversen av deltonsnumret (under förutsättning att grundtonens amplitud är 1). Figur 2 illustrerar en digital (diskret signal) sågtand skapad på liknande sätt som Kodexempel 1 samt ger en jämförelse med en analog sågtand. Figur 2 gör även ett försök att visa och hur frekvenser högre än Nyquistfrekvensen (10 Hz i detta fall) speglas ner i det hörbara frekvensbandet. Det är denna oönskade spegling som kallas vikning.

Om en sågtandsvågform genereras med hjälp av Kodexempel 1 går det att likställa med att sampla en kontinuerlig analog signal. Eftersom den analoga signalen inte är bandbegränsad, kommer den samplade versionen orsaka vikning (Stilson och Smith 1996).



Figur 1 – Illustration av en analog sågtands deltoner



Figur 2 – Illustration av en av en kontinuerlig (analog) samt digital sågtand i tidsdomän (överst), den digitala sågtanden är skapad på liknande sätt som Kodexempel 1. Även de båda signalernas frekvensspektrum illusteras. Eftersom den analoga signalen som samplas i Kodexempel 1 inte är bandbegränsad, kommer den samplade versionen innehålla den analoga signalens oändliga övertoner "nedvikta" under Nyquistfrekvensen (10 Hz i detta fall) som går att se av frekvensspektrum nederst i bild.

2.3 Metoder för att reducera vikning

Man har två möjligheter när det kommer till vikning vid digital oscillatordesign, antingen totalt eliminera eller reducera tillräckligt mycket vikning. För att eliminera eller minimera vikning finns det en del olika algoritmer. Dessa algoritmer går att dela in i tre olika kategorier, idealt bandbegränsade, vikningsdämpande och delvis bandbegränsade (Valencia Otalvaro 2015). Det

är önskvärt att dessa metoder är beräkningsmässigt effektiva. Metoderna är särskilt attraktiva om de inte kräver några anrop av trigonometriska funktioner, inte är beroende av att veta som händer här näst samt inte är i behov av information sparad i tabeller (Pirkle 2014).

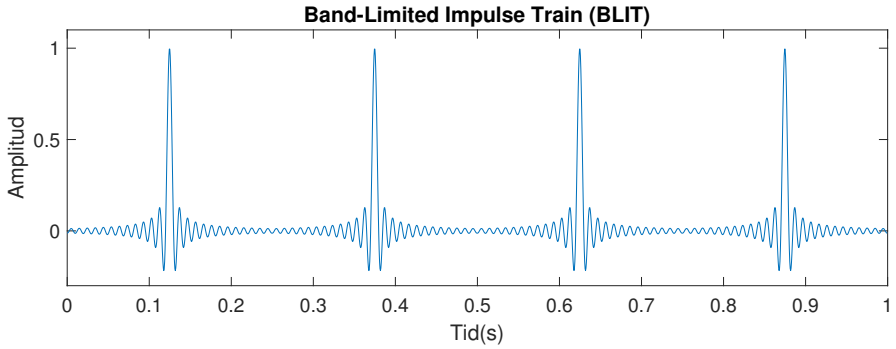
I kategorin idealt bandbegränsade algoritmer har många metoder utvecklats för att helt och hållet eliminera vikning, men alla dessa är beräkningsmässigt dyra (Valencia Otalvaro 2015). En av dessa metoder är additiv syntes. Additiv syntes är per definition fri från vikning så länge endast frekvenser upp till Nyquistfrekvensen adderas (Stilson och Smith 1996). Additiv syntes är beräkningsmässigt dyr, speciellt för låga frekvenser. En sågtandsvågform med grundfrekvensen 27 Hz har 800 övertoner innan Nyquistfrekvensen, vilket innebär 801 anrop av en $\sin(x)$ funktion under en period (Pirkle 2014).

I kategorin vikningsdämpande algoritmer genereras en brant lutning i frekvensplanet. Lutningen realiseras genom att multiplicera med en exponentiellt avtagande funktion på övertonernas amplitud innan signalen samplas, därefter återställs lutningen med ett digitalt filter. Dessa algoritmer tillåter all vikning men amplituden för alla vikningskomponenter reduceras (Välimäki och Huovilainen 2007).

I kategorin av delvis bandbegränsade algoritmer, vilket är den kategori som kommer behandlas i detta examensarbete, tillåts viss vikning vid höga frekvenser eftersom vi människor är mindre känsliga för de förvrängningar vikning orskar vid höga frekvenser än vid låga och mellanfrekvenser (Valencia Otalvaro 2015). Dessa metoder utnyttjar alltså att människors hörsel är mer känslig vid vissa frekvenser. Band-limited Impulse Trains (BLIT) (Stilson och Smith 1996) samt Band-limited Step Functions (BLEP) (Brandt 2001; Leary och Bright 2009) är exempel på två algoritmer i denna kategori.

2.3.1 BLIT

Band-limited Impulse Trains är i princip vad det låter som, ett tåg med periodiska bandbegränsade impulser, se Fig 3 för illustration.



Figur 3 – Illustration av ett tåg med periodiska bandbegränsade impulser (BLIT)

BLIT utnyttjar det faktum att ett idealt antivikningsfilter består av ett så kallat rektangulärt fönster (1) i frekvensdomänen med ett gränsvärde på $F_s/2$ (halva samplingsfrekvensen).

$$H_{F_s}(f) = \begin{cases} 1 & |f| < F_s/2, \\ 0 & \text{annars} \end{cases} \tag{1}$$

Invers fouriertransform ger Ekvation (2) i tidsdomän.

$$h_s(t) = \mathrm{sinc}(F_s t) \triangleq \frac{\sin(\pi F_s t)}{\pi F_s t} \tag{2}$$

Ett idealt impulståg med period T_1 ges av (3)

$$x(t) = \sum_{l=-\infty}^{\infty} \delta(t + lT_1) \tag{3}$$

för att bandbegränsa impulståget kan vi applicera det ideala antivikningsfiltret h_s genom faltning (4)

$$x_f(t) \triangleq (x * h_s)(t) = \sum_{l=-\infty}^{\infty} h_s(t + lT_1) = \sum_{l=-\infty}^{\infty} \mathrm{sinc}(t/T_s + lP) \tag{4}$$

där $P = T_1/T_s$ är signalens period uttryckt i antalet samplingsperioder, eftersom $x_f(t)$ är bandbegränsat går det att sampla utan vikning som (5)

$$y[n] \triangleq x_f(nT_s) = \sum_{l=-\infty}^{\infty} \mathrm{sinc}(n + lP) \tag{5}$$

denna oändliga summa kan skrivas om till (6) (Stilson och Smith 1996)

$$y[n] = (M/P)\frac{\sin(\pi(M/P)n)}{M\sin((\pi(M/P)n)/M)} \tag{6}$$

där M är antalet frekvenser (deltoner), M är alltid udda och får inte överskrida perioden P i antalet samplingar (Scavone 2002).

För att generera en sågtandsvågform med reducerad vikning kan man integrera det bandbegränsade impulståget minus en konstant som fås genom att integrera en period över impulståget (Scavone 2002).

Det finns ett antal problem med BLIT. Ett av problemen är att det kan det hända att impulserna överlappar, vilket betyder att svansen från en impuls måste mixas ihop huvudet på nästa. Detta kräver en fördröjning av signalen för att veta när nästa diskontinuitet ska ske, detta är ett problem eftersom det inte är önskvärt med fördröjning när man genererar en vågform (Pirkle 2014). Ett annat problem är att integrering av signalen orsakar en oönskad DC-förskjutning vid oscillatorns start (Pirkle 2014).

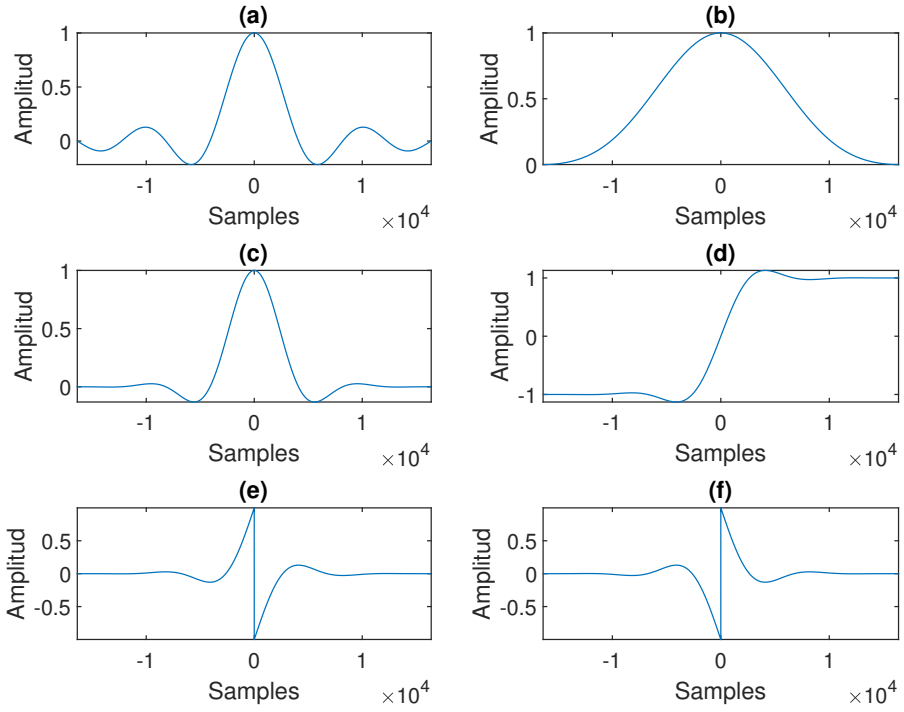
2.3.2 BLEP

Band-limited Step Functions (BLEP) är en förbättring (Pirkle 2014) och utökning av BLIT. Syftet med BLEP är att slippa integrera i realtid. De finns olika implementationer av BLEP, en av dessa är BLEP-Look-Up-Table (BLEP-LUT). BLEP-LUT bygger på att man börjar med en enda bandbegränsad impuls, integrerar denna impuls och lagrar den i en tabell. Värdena från denna tabell adderas sedan till ett bestämt antal punkter/samples kring vågformens diskontinuitet(er), samma antal punkter ändras på vardera sida om diskontinuiteten. Ju fler punkter som ändras desto mer lågpassfiltrerad blir signalen. En annan egenskap som är värd att nämna är att det inte går att ändra så många punkter vid höga frekvenser. Anledningen till detta är att det inte finns lika många samples under en period vid höga frekvenser som vid låga frekvenser. För enkelhetens skull kommer BLEP-LUT benämnas BLEP i forsättningen. BLEP är lite mer

beräkningsmässigt dyr än andra delvis bandbegränsade och vikningsdämpande metoder (Välimäki och Huovilainen 2007) samt kräver minnesresurser. En anledning till att BLEP är lite mer beräkningsmässigt dyr är att algoritmen kräver division av grundfrekvensen för varje sample som ändras kring vågformens diskontinuitet. I (Valencia Otalvaro 2015) presenteras en metod för att ta fram tabellvärden i Matlab, denna metod går till enligt följande:

1. Utgå från Ekvation (2) i BLIT-algoritmen som är en sinc funktion.
2. Multiplicera Ekvation (2) med en fönsterfunktion.
3. Integrera resultatet av steg 2.
4. Subtrahera resultatet av steg 3 med ett enhetssteg som är -1 för $t < 0$ och 1 för alla andra t. Detta steg ger en så kallad BLEP residual som kan lagras i tabell och användas för signaler med stigande vågform innan diskontinuitet.
5. Invertera resultatet av steg 4 för signaler med fallande vågform innan diskontinuitet.

Anledningen till att sinc funktionen multipliceras med en fönsterfunktion i steg 2 är att sinc funktionen är oändlig och fönsterfunktionen används för att begränsa funktionen. BLEP algoritmen kan modifieras med val av fönsterfunktion, sinc funktionens skärningar med 0 samt tabellstorlek. För en illustration av stegen ovan se Figur 4. För en detaljerad beskrivning och implementation se Bilaga D.

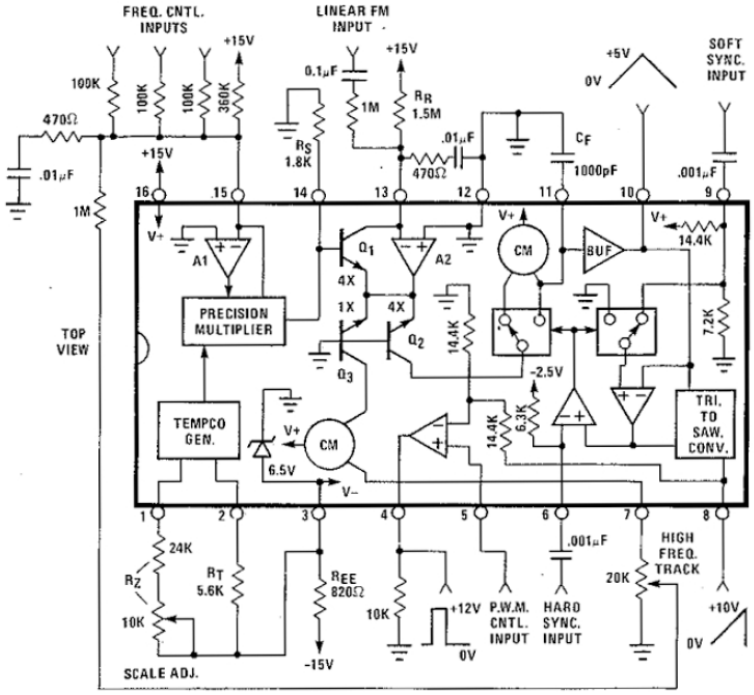


Figur 4 – Steg för att ta fram en BLEP residual. (a) sinc funktion, (b) fönsterfunktion av typ Blackman, (c) sinc funktion multiplicerad med Blackmanfönster, (d) integrering av (c), (e) BLEP residual som fås genom att subtrahera (d) med ett enhetssteg som är -1 för $t < 0$ och 1 för alla andra t, (f) inverterad BLEP residual

2.4 CEM3340

Figur 5 – Den integrerade kretsen CEM3340 [Foto av Dave Dewar]

Den integrerade kretsen CEM3340, se Figur 5 och 6, är en Voltage Controlled Oscillator (VCO) med möjlighet till utmatning av fyra vågformer: sågtand, fyrkant, puls och triangel. CEM3340 används i många analoga syntklassiker som till exempel Sequential Circuits Prophet 5 Rev 3 (1978), Roland SH-101 (1983) och Moog Memorymoog (1982) för att nämna några. Se bilaga F för exempel på kopplingsscheman av oscillatorer där CEM3340 utnyttjas.



Figur 6 – CEM3340 Krets- och blockdiagram från CEM3340's datablad

2.5 Virtual Studio Technology och Digital Audio Workstation

Virtual Studio Technology (VST) är ett gränsnitt skapat av Steinberg Media Technologies GmbH för att utveckla VST-instickspro gram till Digital Audio Workstation's (DAW's) som har stöd för VST. DAW är mer eller mindre omfattande programvara eller hårdvara som till exempel möjliggör komponering, mixning och producering av ljud och musik. Ett exempel på en DAW är Bitwig studio som också är den DAW som använts som testmiljö under detta examensarbete. Ett exempel på ett VST-instickspro gram är Signalizer. Signalizer erbjuder bland annat möjlighet att visualisera en ljudsignals frekvensspektrum samt dess periodiska vågform.

2.6 JUCE

JUCE är ett C++ ramverk för att underlätta implementation av VST-instickspro gram. Genom att välja Create New Project -> Audio Plug-In i medföljande programvara Projucer tar JUCE hand om implementering av VST gränsnitt samt erbjuder funktionalitet för ljudbehandling och skapande av grafiska användargränssnitt. JUCE abstraherar alltså bort VST-implementation från användaren, vilket innebär att användaren kan fokusera på att utveckla funktionalitet istället för hur instickspro grammet ska fungera tillsammans med DAWs. På JUCE's webbplats juce.com erbjuds handledning för att skapa enkla VST-instickspro gram. JUCE tillhandahåller olika licenser för användning av dess ramverk, bland annat en gratislicens som innebär att produkten som utvecklas med JUCE kommer ha en ”splashscreen” innan uppstart samt begränsningar av bland annat vilken inkomst produkten får generera om man väljer att utveckla en kommersiell produkt. Vill man slippa denna splashscreen samt andra begräsningar får man välja en av deras betallicenser.

2.7 RackAFX

RackAFX är likt JUCE på många sätt, precis som JUCE är RackAFX ett C++ ramverk för att underlätta implementation av bland annat VST-insticksprogram. Syftet med RackAFX är användaren ska kunna fokusera på signalbehandling snarare än grafiska användargränssnitt och VST-implementation precis som med JUCE. RackAFX är framtaget av Will Pirkle som även har skrivit boken 'Designing Software Synthesizer Plug-Ins in C++' (Pirkle 2014). Precis som boktiteln antyder lär den ut hur man kan gå tillväga för utveckla synt-insticksprogram för bland annat VST. I boken går det att följa handledning som beskriver utveckling av några olika exempel synt-insticksprogram i RackAFX. På hemsidan för RackAFX finns dessa färdiga exempel synt-insticksprogram att ladda ned för att bygga i Visual Studio och testköra i någon DAW. Med RackAFX krävs inga licenser för att utveckla kommersiella produkter och det finns inga splashscreens.

3 Metod

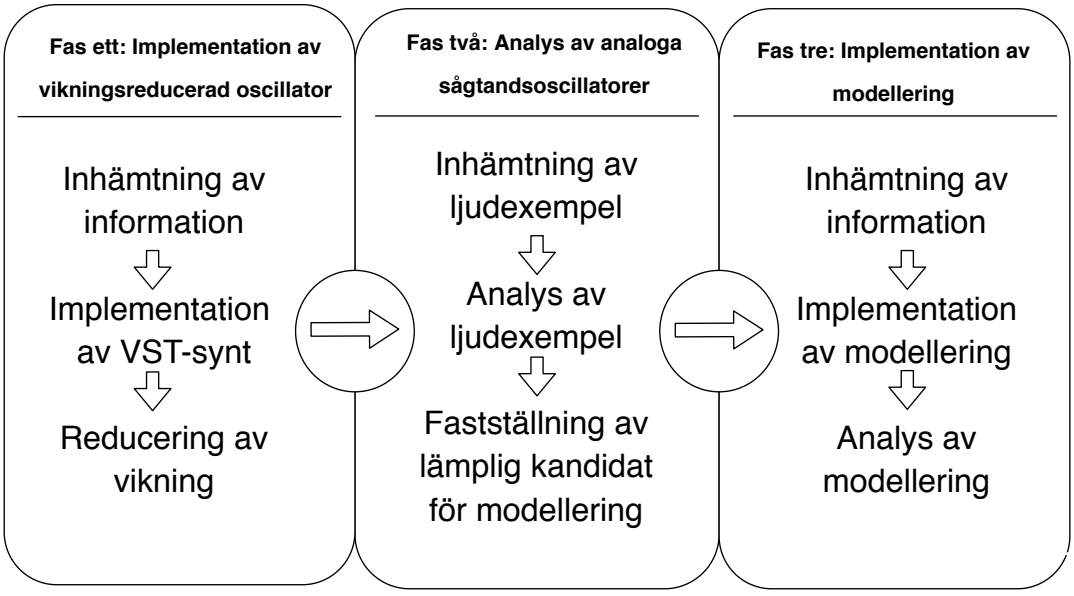
I detta kapitel presenteras och motiveras examensarbetets arbetsmetoder samt verktyg som använts under examensarbetet.

3.1 Uppstart

Största delen av examensarbetets uppstart bestod av att hitta en lämplig vetenskaplig infallsvinkel som berör ämnena synt och mjukvaruutveckling. I början övervägdes att göra något machine-learning-relaterat och därför gjorde inläsning på detta område. Ganska tidigt konstaterades att något machine-learning-relaterat skulle ta ganska stort fokus från det förutbestämda huvudämnet synt. Istället valdes det att fokusera på ett problem som är oundvikligt vid mjukvarurutveckling av en synt, det vill säga vikning. Eftersom ämnet vikning redan behandlats en del i forskning gjordes bedömning att examensarbetet behövde en unik infallsvinkel. Därför gjordes valet att modellera samt vikningsreducera en digital implementation av någon analog synts oscillator. På detta specifika område hittades endast en vetenskaplig artikel, nämligen (Pekonen, Lazzarini m. fl. 2011). Av den anledningen ansågs det vara rimligt att ytterligare undersökning på området görs.

3.2 Arbetsmetod

Ingen etablerad arbetsmetod eller process har utnyttjats under examensarbetets gång. Genomförandet av examensarbetets implementering av VST-synt och digital implementation av analog oscillator realiserades istället sekventiellt i tre på förhand väntade faser, se Figur 7. Inför och under alla faser har information inom områden som berörs i faserna inhämtats genom litteraturstudier.



Figur 7 – Examensarbetets olika faser

3.3 Fas ett: Implementation av vikningsreducerad sågtandsoscillator

Detta examensarbetes första fas bestod i huvudsak av att reducera de digitala artefakter som sampling av en analog sågtandsvågform orsakar, med andra ord vikning. Fas ett realiserades genom tre steg, inhämtning av information, implementation av VST-synt samt reducering av vikning.

Innan arbetet med implementation av VST-synt påbörjades inhämtades information med hjälp av söktjänster på internet samt internetforum. Information om vilka ramverk som fanns tillgängliga för implementation av VST-syntar inhämtades huvudsakligen ifrån söktjänsten google, audio plug-in forumet kvr-audio.com samt från boken (Pirkle 2014). Då det finns få ramverk för VST-instickSprogram begränsades snabbt alternativen till de två mest omtalade på kvr-audio, nämligen RackAFX och JUCE. För att få en uppfattning av ramverket togs beslutet att testa båda för att sedan välja det ramverk som passade för detta examensarbete. Jämförelse mellan dessa ramverk gjordes utifrån stabilitet, licensavtal, dokumentation samt vilken funktionalitet de erbjöd.

För att ha möjlighet att eliminera eller reducera vikning när VST-synten hade implementerats inhämtades även information om metoder för detta ändamål genom litteraturstudier. Information hämtades främst från vetenskapliga artiklar som hittades genom söktjänsten LUBsearch. Cirka 20-talet artiklar hittades, efter inläsning kunde dessa artiklar begränsas till cirka 10 relevanta artiklar. Relevanta artiklar ansågs vara de som beskrev metoder som erbjöd bra ljudkvalite, var beräkningsmässigt effektiva samt tog upp få minnesresurser.

När ramverk valts ut var det dags att implementera en VST-synt med en ideal sågtandsoscillator, det vill säga en oscillator utan vikningsreducering. Vid implementation låg fokus på att skriva så effektiv kod som möjligt. Eftersom många sågtandsoscillatorer redan implementerats och fokus i detta examens arbete inte låg på att implementera en ideal sågtandsoscillator ansågs det inte vara nödvändigt att skriva några egna algoritmer för detta ändamål. Istället användes redan etablerade algoritmer som inspiration, med motivering att dessa säkert redan optimerats för effektivitet. När VST-synten implementerats fick den namnet SAW-X, SAW med anledning av sågtandsvågformen och X ifrån eXamensarbete. Motivet att namnge VST-synten var för att enkelt kunna referera till den i denna rapport.

När valet av metod för vikningsreducering hade gjorts implementerades och integrerades denna metod i SAW-X. SAW-X förseddes med ett grafiskt användargränssnitt för att enkelt kunna testa den vikningsreducerade algoritmens olika parametrar. Testning av anti-viknings metodens parametrar utfördes för att avgöra vilka värden som gav upphov till minst vikning i ett så brett frekvensspektrum som möjligt. Testmiljön för detta ändamål fastställdes vara Bitwig tillsammans med instickSprogrammet och analysverktyget Signalizer. Motivering till att använda Bitwig var att det redan fanns installerat på datorn som användes under examensarbetet. Motivering till val av analysverktyg var att signalizer är gratis samt erbjöd möjlighet att analysera signaler både i tidsdomän och frekvensdomän. När parametrarnas värden fastställts noterades dessa och implementerades i SAW-X. En signal ansågs vara vikningsfri om alla vikningskomponenter hade en amplitud under -90 dB (med avseende på signalens grundton) i ett frekvensspektrum från 0 - 20000 Hz. Dessa siffror är hämtade från (Frei 2010).

3.4 Fas två: Analys av analoga sågtandsoscillatorer

Tyngdpunkten i den andra fasen bestod av att analysera klassiska syntars analoga sågtandsvågformer. Denna fas kan ses som en förberedelsefas inför fas tre. Anledningen till detta är att

fas tre är beroende av att det inhämtats information om karaktären hos någon analog synts sågtandsvågform innan det går att modellera karaktären.

Eftersom det saknades tillgång till analoga syntar togs beslut att söka upp ljudexempel på internet. På grund av tidigare kännedom om att kretsen CEM3340 använts i många analoga syntklassiker gjordes valet att begränsa sökningen till syntar med CEM3340. Fyra olika samlingar med ljudexempel från tre analoga syntar samt ljudexempel inspelade direkt från CEM3340 kretsen hittades.

När ljudexempel hittats var det dags att välja vilket mjukvaruverktyg som skulle användas för att analysera de olika vågformerna. Verktygen som det valdes mellan var Matlab och Octave. Jämförelser som gjordes mellan verktygen var funktionalitet, tillgänglighet och dokumentation. Att enkelt kunna anlysera ljudfiler i både tidsdomän och frekvensdomän var funktionalitet som ansågs vara viktig.

När verktyg valts och ljudfiler analyserats i både tid och frekvensdomän var det dags att välja en lämplig kandidat för modellering. Valet av vågform att modellera baserades på ett enda kriterium, den vågform som ansågs minst tidskrävande att modellera skulle väljas. Motivet till detta kriterium var examensarbetets tidsbegränsning.

3.5 Fas tre: Modellering och implementering för att uppnå önskad karaktär

Fas tre började precis som med fas ett med inhämtning av information, denna gång hämtades information om vilka metoder som fanns tillgängliga för att modellera vågformer. Inhämtning av information gjordes genom att söka efter artiklar där någon analog synts oscillator modellerats, söktjänsterna som användes var LUBsearch, Google och Google Scholar. Det enda som hittades var artikeln (Pekonen, Lazzarini m. fl. 2011). I denna artikel presenterades två olika metoder för modellering. Den första metoden gick ut på på att ändra en vågforms utseende genom förvränga vågformens fas, den andra metoden gick ut på att matcha en analog vågforms frekvensspektrum genom att filtrera en redan vikningsreducerad digital sågtandsoscillator tills önskat frekvensspektrum uppnåts. Ett tredje alternativ som presenterats av min handledare Mats Lilja var att ta fram en ad hoc metod, det vill säga en metod tillsatt för ett särskilt ändamål. När modelleringsmetod skulle väljas togs det först och främst hänsyn till vilken metod som skulle vara minst tidkrävande, motivet till detta var även denna gång examensarbetets tidbegränsning.

När modelleringsmetod valts var det dags att implementera och integrera modelleringen i SAW-X. När implementering av modellering var genomförd blev nästa steg att analysera resultatet av implementationen. Eftersom analysen görs på liknande sätt som steg två i fas två användes samma verktyg för denna analys, nämligen Bitwig och Signalizer.

3.6 Källkritik

I detta avsnitt ges motiveringar till varför det går att lita på de referenser som finns med i referenslistan.

(Brandt 2001),(Pekonen och Välimäki 2011),(Pekonen, Lazzarini m. fl. 2011),(Välimäki och Huovilainen 2007),(Stilson och Smith 1996) kan alla anses vara pålitliga källor eftersom de är vetenskapliga artiklar där alla material granskas av experter före publicering.

(Leary och Bright 2009) är ett amerikanskt patent och kan anses pålitligt eftersom patent granskas innan de godkänns.

(Pekonen 2014) är ett doktorsavhandling och kan anses som pålitlig eftersom den granskats av experter.

(Pirkle 2014) är en bok och kan anses som pålitlig eftersom teknikerna som beskrivs refererar i sin tur till vetenskapliga artiklar som granskats av experter.

(Scavone 2002) är en ogranskad artikel men har endast uttnyttjas för att förenkla en ekvation med ursprung från (Stilson och Smith 1996) som är granskad av experter, därför kan källan anses pålitlig.

(Valencia Otalvaro 2015) är ett examensarbete granskad av experter, därför kan den anses vara pålitlig.

(Frei 2010) är en onlinebok och kan anses pålitlig eftersom den i sin tur refererar till vetenskapliga artiklar.

4 Analys/Genomförande

I detta avsnitt beskrivs och motiveras alla val som gjorts. Även resultat från examensarbetets tre faser samt problem och deras lösningar beskrivs och motiveras. För alla tester har en samplingsfrekvens på 44100 Hz använts med motiveringen att det är en vanlig samplingsfrekvens i ljudsammanhang. För en beskrivning av hur FFT utförts under detta examensarbete vid plottning i frekvensdomän se Bilaga E. Eftersom FFT antar att signalen den ska transformera är repetiv inom tidsintervallet användes en fönsterfunktion. Valet av fönsterfunktion valdes till samma som den i (Pekonen och Välimäki 2011), nämligen Chebyshev, med motivering att författarna till artikeln är experter på området samt att detta examensarbete inte har för avsikt att genomföra en undersökning av vilka fönsterfunktioner som lämpar sig bäst för FFT.

4.1 Fas ett: Implementation av vikningsreducerad sågtandsoscillator

I denna fas implementerades en VST-synt med en vikningsreducerad sågtandsoscillator.

4.1.1 Implementation av SAW-X

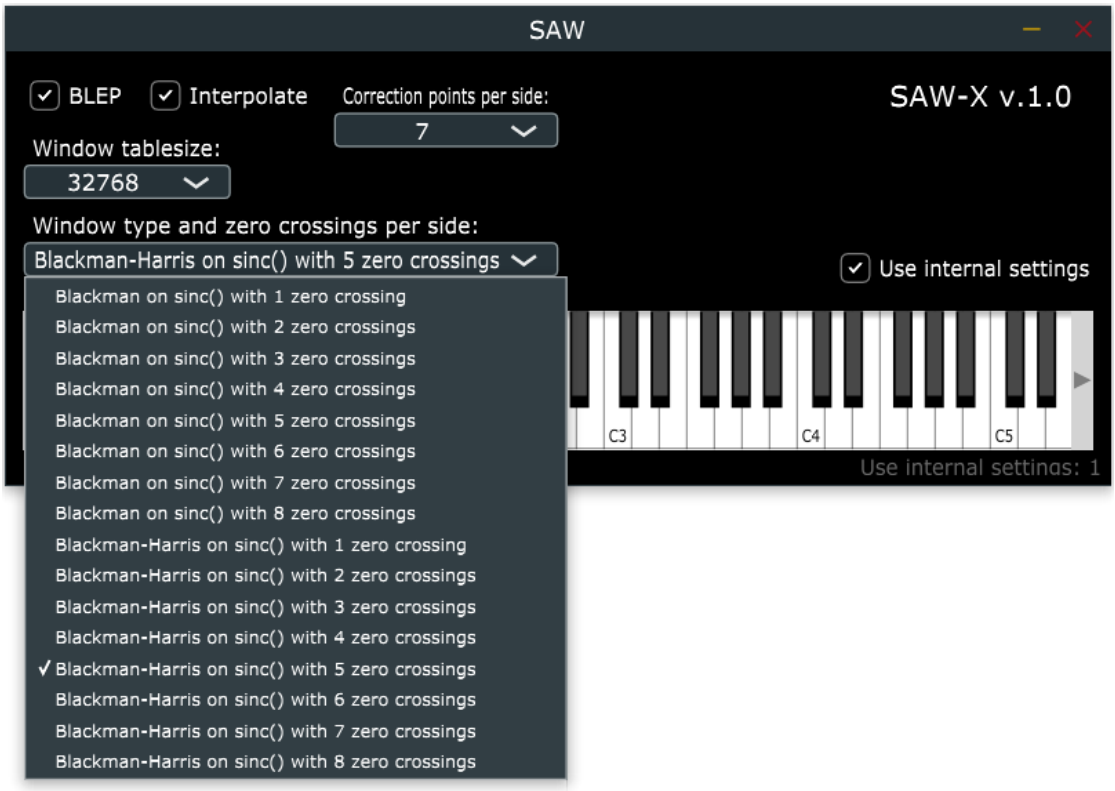
För implementation av SAW-X valdes slutligen ramverket JUCE framför RackAFX. Detta valet kom bland annat av att ramverket RackAFX först testades men problem uppstod redan när medföljande exempelprojekt inte gick att bygga. När JUCE testades var upplevelsen mer stabil och det gick snabbt att bygga ett av deras exempelprojekt. Trots att jämförelse var tänkt att göras utifrån stabilitet, liscensavtal, dokumentation samt vilken funktionalitet de erbjöd, fick stabilitet vara den faktor som avgjorde på grund av tidsbegränsning. JUCE var alltså det ramverk som valdes för implementation av SAW-X. Implementation och kompilering av SAW-X gjordes i Visual Studio 2017 med anledning av att datorn som användes under detta examensarbete var en Windows-dator samt att JUCE's Projucer har stöd för att öppna projekt i Visual Studio 2017. Grunden till SAW-X bygger på JUCE's tutorial 'Build a MIDI synthesiser'. Anledningen till att den valdes var för att den erbjuder en enkel grund att bygga vidare på. Oscillatorn i JUCE's tutorial genererar en sinus vågform, denna ersattes med en ideal sågtandsoscillator enligt Kodexempel 1. Se Figur 8 för en illustration av SAW-X grafiska användargränssnitt.

4.1.2 Val av metod för vikningsreducering

Det största problemet under fas ett visade sig vara att läsa in sig på alla matematisk avancerade tekniker för att reducera vikning. Detta visade sig vara mer tidskrävande än tänkt. På grund av att det spenderades mycket tid på inläsning fick målet om att försöka förbättra någon metod ändras till att istället hitta den metod som dokumenterat erbjöd bra ljudkvalite, var beräkningsmässigt effektiv samt tog upp få minnesresurser. Efter omfattande inläsning valdes slutligen metoden Band-limited Step Functions (BLEP) för att reducera vikning. BLEP är lite mer beräkningsmässigt dyr än andra delvis bandbegränsade och vikningsdämpande metoder (Välimäki och Huovilainen 2007) samt kräver minnesresurser. Anledning till att BLEP valdes trots dessa kostnader är att det är den metod som kommer närmst den idealt bandbegränsade metoden additiv syntes ljudmässigt (Välimäki och Huovilainen 2007).

4.1.3 Integrering av BLEP i SAW-X

BLEP-algoritmen realiserades med verktygen Visual Studio 2017 och Matlab. Implementation av BLEP-algoritmen bygger på algoritmen från (Pirkle 2014), se Bilaga A för detta examensarbetets



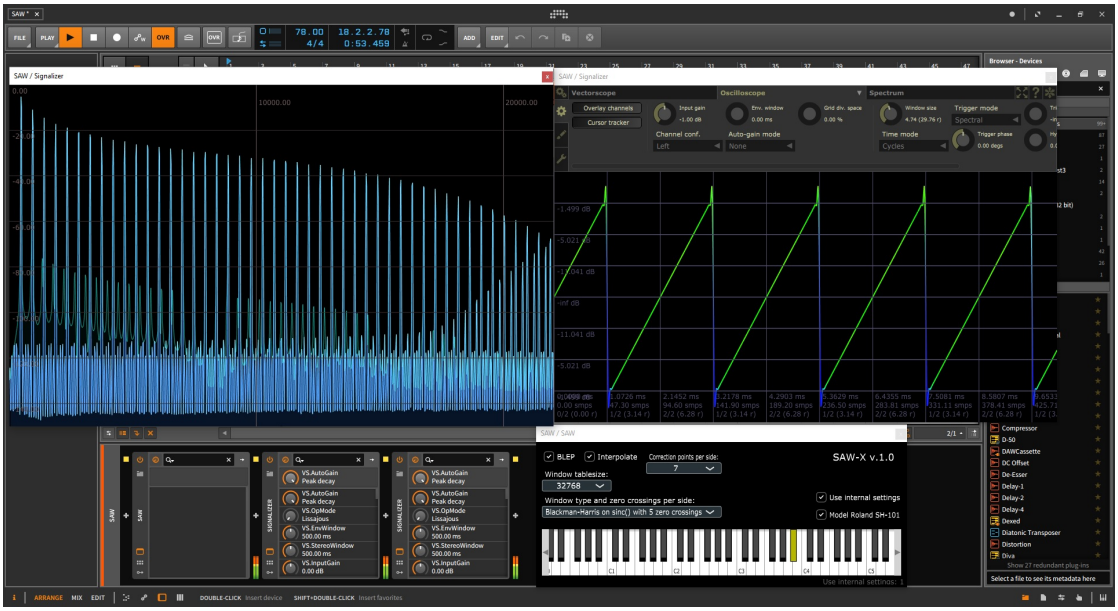
Figur 8 – SAW-X, VST-synt som implementerades under fas ett

version. För framtagning av tabellvärden användes Matlab och en metod av (Valencia Otalvaro 2015) som modifierats för att passa detta examensarbete, se Bilaga D för detaljerad beskrivning. Anledning till att just verktyget Matlab användes var för att det finns tillgängligt för studenter vid Lunds Universitet.

Vid framtagning av tabellvärden till BLEP användes (Valencia Otalvaro 2015) som källa, i (Valencia Otalvaro 2015) görs omfattande undersökning av vilka fönsterfunktioner, sinc funktionens antal skärningar med 0 samt vilka tabellstorlekar som ger bäst resultat för BLEP-algoritmen. I (Valencia Otalvaro 2015) ligger fokus på att optimera BLEP-algoritmen och 17 olika fönsterfunktioner testas. Eftersom fokus i detta examensarbete var att digitalt implementera en bandbreddsbegränsad modellering av en analog sågtandsoscillator med recuderad vikning samt integrera denna implementation i en VST-synt fanns det inte utrymme att göra lika omfattande undersökning av BLEP-parametrar. Istället testades endast fönstertyperna Blackman och Blackman-Harris eftersom (Valencia Otalvaro 2015) nämner dessa två och Dolph-Chevyshev som de bäst lämpade standardfönsterfunktionerna för BLEP samt att dessa finns som färdigimplementerade funktioner i Matlab. I (Valencia Otalvaro 2015) tas specialdesignade fönstertyper fram som ger bättre resultat än Blackman och Blackman-Harris, dessa utelämnas här på grund av att de anses för tidskrävade att återskapa. Tabellstorlekar valdes till 4096, 8192, 16484 och 32768 för att 4096 var den minsta storleken och 32768 största storleken som användes i (Valencia Otalvaro 2015). sinc funktionens antal skärningar med 0 valdes till samma som (Valencia Otalvaro 2015), det vill säga 1-8. Till skillnad från (Valencia Otalvaro 2015) implementerades linjär inter-

polering för hämtning av tabellvärden. Anledningen till att linjär interpolering implementerades var att ansågs vara ett enkelt tillägg som var värt att testa. Interpolering användes eftersom BLEP-algoritmens uträknade tabellindex sällan är ett heltal. Se Bilaga A för implementation av interpolering.

Vid testning av SAW-X BLEP-algoritm användes Bitwig Studio som DAW tillsammans med Signalizer. För en illustration av denna testmiljö se Figur 9.



Figur 9 – Testmiljö för att ta fram optimala BLEP parametrar

SAW-X parametrar 'Window type and zero crossings per side' och 'Correction points per side' har 16 olika valmöjligheter i deras rullgardinmeny vardera. 'Correction points per side' valmöjligheter är heltalen 1-16. 'Window type and zero crossings per side' valmöjligheter kan ses av Figur 8. Testning genomfördes genom att låta Bitwig spela igenom alla 128 midinoter från C-2 (8.18 Hz) till G8 (12543.85 Hz). Under tiden visualiserades alla 256 (16x16) möjliga kombinationer av SAW-X parametrar 'Window type and zero crossings per side' och 'Correction points per side' i Signalizers grafiska frekvensband och samtidigt dokumenterades vilka parametrar som gav upphov till minst vikning. Anledning till att endast en kombination av dessa parametrar testades var att det tidigt bestämdes att tabellstorleken 32768 fick vara den som användes. Motivering till denna tabellstorlek var att det inte är en stor tabellstorlek med dagens minnesstorlekar samt för att minska antalet kombinationer från 1024 (16x16x4) till 256 (16x16). Även interpoleringsparametern lämnades utanför testet då det tidigt noterades att denna bidrog till relativt lite vikningsreducering.

Efter testning implementerades de parametrar som gav upphov till minst vikning i SAW-X. För att kunna utnyttja dessa parametrar lades en kryssruta med texten 'Use internal settings' till i SAW-X grafiska användargränssnitt. När denna kryssruta är aktiverad ignoreras de andra parametrarna.

4.1.4 Presentation av resultat

Alla tre BLEP-tabeller och deras parametrar presenteras i Tabell 1 under Resultat. Viktigt att tänka på när Tabell 1 tolkas är att en BLEP-tabells fasta värden är fönstertyp och antal skärningar med 0. Val av antal korrigerade punkter påverkar alltså inte val av BLEP-tabell. I rullgardinsmenyn under texten 'Window type and zero crossings per side:' på Figur 8 väljer man alltså vilken tabell korrigeringspunkterna ska läsas ifrån.

För att kunna presentera resultatet av BLEP-algoritmen i denna rapport exporterades ljudfiler från Bitwig. För att kunna skapa Figur 12 - 15 importerades dessa ljudfiler till Matlab där FFT utfördes och slutligen exporterades figurerna.

4.2 Fas två: Analys av analoga vågformer

Under fas två hittades fyra samlingar med ljudexempel från de tre analoga syntarna Roland SH-101, Moog Memorymoog och Octave Plateau Voyetra 8 samt en samling med ljudexempel inspelade direkt ifrån CEM3340 kretsen. De tre samlingarna från syntar är alla från syntar med subtraktiv syntes. Samtliga upphittade ljudexempel hade 44100 Hz samplingsfrekvens och 24-bitars bitdjup.

Ljudexemplen som hittades har ursprung från tre olika källor. Moog Memorymoog och Octave Plateau Voyetra 8 hittades på soundsdevine.com som är en kommersiell återförsäljare av ljudsamlingar. Roland SH-101 hittades liveschool.net som är en skola för elektronisk musik. Ljudexemplen av CEM3340 hittades på forumet dogsonacid.com och är inspelade av en av forumets användare. Innebörden av att ljudexempeln kommer ifrån olika källor är att de med stor sannolikhet spelats in med hjälp av olika utrustning. Detta examensarbete utgår ifrån att det är osannolikt att det är så pass stor skillnad på inspelningsutrustning eller att inspelningsutrustningen är av så pass dålig standard att det skulle påverka examensarbetets analys eller jämförelse av ljudfilerna.

Det är även värt att notera att endast CEM3340 ljudfilerna är inspelade direkt från oscillatorkretsen, de andra ljudfilerna är med största sannolikhet inspelade ifrån syntarnas ljudutgång. Eftersom signalen passerar fler elektroniska komponenter på väg till ljudutgången än om den spelas in direkt från oscillatorn går det inte att utesluta att signalen påverkats mellan oscillator och ljudutgång. För att kunna göra en jämförelse av ljudfilerna utgår detta examensarbete ifrån att signalen inte påverkats nämnvärt.

För att analysera ljudfilernas vågformer importerades de i Matlab där de plottades i tids och frekvensdomän. Ljudexempeln plottades i tre olika frekvenser med lägsta frekvens 65.41 Hz och högsta frekvens 1046.50 Hz. Valet av dessa frekvenser gjordes på grund av att det är den lägsta och högsta frekvensen i samlingen med ljudexempel av CEM3340. För att ytterligare kunna analysera de analoga syntarna i frekvensdomän plottades även en ideal sågtand utan vikning i samma plot som ljudexempeln. Denna ideala sågtand utan vikning framställdes med en fourierserie med övertoner upp till Nyquistfrekvens. Se Figur 16 och 17 samt Bilaga B för resultatet av plotterna i fas två.

4.3 Fas tre: Modellering och implementering för att uppnå önskad karaktär

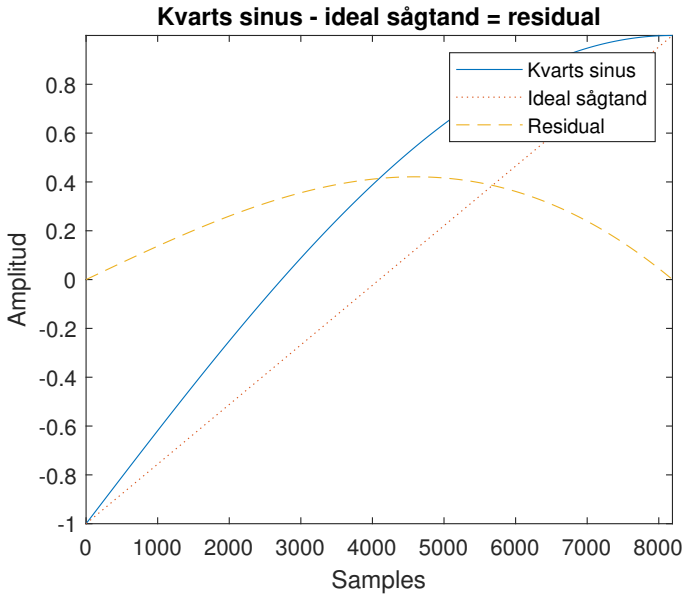
Roland SH-101's sågtandsvågform valdes för modellering främst för att den ansågs vara den enklaste och därmed också minst tidskrävande att modellera. Anledningen till att den ansågs enkel var att upp till 370 Hz påminner en period av vågformen om en kvarts sinuskurva från

sin(0) till sin($\pi/2$). Efter 370 Hz påminner Roland SH-101's sågtandsvågform mer om en ideal sågtand.

För att ta fram en modell av Roland SH-101's sågtandsvågform bestämdes tidigt att det var lämpligt spara värden av en kvarts sinus i en tabell. Motiveringen till detta var att undvika många anrop på en sin() funktion under en period på samma sätt som BLEP-algoritmen undviker detta. Det första som gjordes var att spara värden från sin(0) till sin($\pi/2$) i en vektor i Matlab. Två justeringar behövde göras innan denna vektor kunde sparas undan till en tabell som kunde utnyttjas för modellering. Först justerades värdena så att de matchar en bipolär sågtandsvågform, anledningen till detta är att sågtandsvågformen i SAW-X är bipolär. För att sedan kunna addera värdena av denna vektor till en ideal sågtand räknandes först skillnaden ut mellan denna vektor och en vektor med en ideal sågtand. Se Kodexempel 2 för en detaljerad beskrivning av hur tabellen skapades samt se Figur 10 för att se skillnaden (residualen) illustrerad.

```
Fs = 44100;                    %samplingsfrekvens
f = 8.1757989156;              %lägsta midifrekvens
samplesPerCyclef = Fs/f;       %samples som behövs för lägsta frekvens

%ser till att det finns gott om samples
tableSize = 2^(nextpow2(samplesPerCyclef));
tsin = 0:1:tableSize-1;        %tidsvektor

quartersin = 2*(sin((pi/2)*(1/tableSize)*tsin)) - 1;
saw = sawtooth(2*pi*1/tableSize*tsin);

%lagra skillnad mellan quartersin och saw i tabell
tableResidual = quartersin - saw;

%skriv residual till fil
writematrix(tableResidual,'qsinres.csv');
```

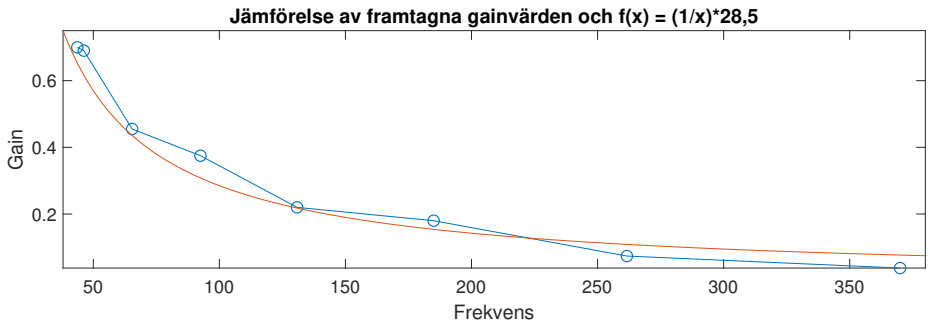
Kodexempel 2 – Algoritm för att spara värden av en kvarts sinus i tabell



Figur 10 – Illustration av framtagning av residual till modellering

Som kan ses av Figur 18 under resultatavsnittet avtar sinuskaraktärens ”styrka” ju högre frekvens Roland SH-101 ljudexempeln har. För att ta reda på med vilken faktor karaktärens styrka avtar testades det att addera residualtabellens värden multiplicerat med en faktor till en ideal sågtand. Denna faktor togs fram genom att först gissa på ett värde mellan 0 och 1 för att sedan förbättra detta värde till modellering var så lik orginal som möjligt. Anledningen att värdena 0 till 1 valdes var för att kvartssinus karaktären inte såg ut att behöva förstärkas någon gång utan snarare försvagas. Detta gjordes för de åtta första ljudexempeln i Figur 18 och 19. Resultaten av dessa slutgiltiga värden plottades i Matlab för att undersöka om de plottade värdena liknade någon matematemisk funktion.

Det finns 66 noter från lägsta midinot C-2 (8.18 Hz) till högsta not som utnyttjar modelleringen F#4 (369.99 Hz), eftersom det bara fanns åtta värden skulle 58 noter få interpoleras fram. Om det istället fanns någon enkel funktion för att beskriva sambandet mellan punkterna ansågs detta vara mer lämpligt än att spara dessa relativt få värden i en tabell och interpolera fram värdena som saknas. När värdena hade plottats lades det märke till att om man drar en rät linje mellan varje diskret värde finns det likheter med funktionen $f(x) = (1/x)$, dock förskjuten. För att ”positionera” $f(x) = (1/x)$ till ungefärligt samma värden som de diskreta värdena multiplicerades funktionen med en gissad faktor som hela tiden förbättrades till slutligen faktorn 28.5 erhållits. För en illustration av de diskreta värdena och $f(x) = (1/x) * 28.5$ se Figur 11. För att se hur $f(x) = (1/x) * 28.5$ utnyttjats se Kodexempel 3.



Figur 11 – Jämförelse av diskreta gainvärden (förstärkningsvärden) och $f(x) = (1/x) * 28.5$*, cirklar representerar de diskreta värdena och den heldragna kurvan representerar funktionen* $f(x) = (1/x) * 28.5$

```
double quartersin(const double& frequency, const double& tableInc
    , double& qsinIndex)
{
  if (frequency > 370)
  {
    return 0.0; //early out
  }

    //linjär interpolering av tabellvärden
  int floorIndex = (int)qsinIndex;
  double qsinValue1 = qsinResTable32768[floorIndex];
  double qsinValue2 = qsinResTable32768[floorIndex + 1];
  double interpolatedValue = qsinValue1 + (qsinValue2 -
    qsinValue1)*(qsinIndex - floorIndex);

  //öka tabellindex
  qsinIndex += tableInc;

  if (qsinIndex >= 32768)
  {
    qsinIndex = 1.0; //reset
  }

    //multiplicera interpolerat tabellvärde med gainfunktion
  if (frequency > 30 && frequency <= 370)
  {
    return (28.5 / frequency)*interpolatedValue;
  }

  //om frekvens <= 30 Hz returnera interpolerat tabellvärde
  else
  {
      return interpolatedValue;
  }
}
```

Kodexempel 3 – Modellerings funktion i C++

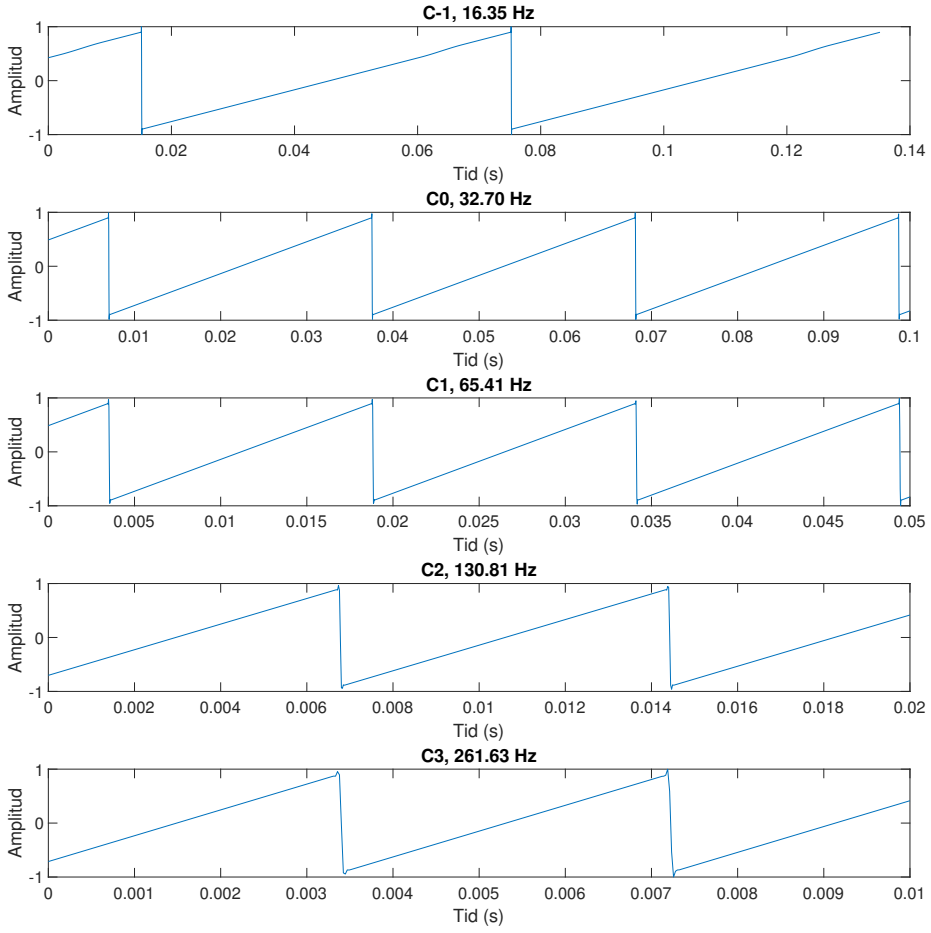
Som går att se av Kodexempel 3 användes linjär interpolering vid hämtning av tabellvärden. Vid framtagning av kvartssinus tabellvärden användes en samplingsfrekvens på 44100 Hz. Motiveringen till att använda interpolering var att det förhoppningsvis gav bättre resultat än ingen interpolering om SAW-X skulle användas i en DAW med högre samplingsfrekvens än 44100 Hz. Under detta examensarbete har endast en samplingsfrekvens på 44100 Hz använts. Detta är alltså inget som är bekräftat.

5 Resultat

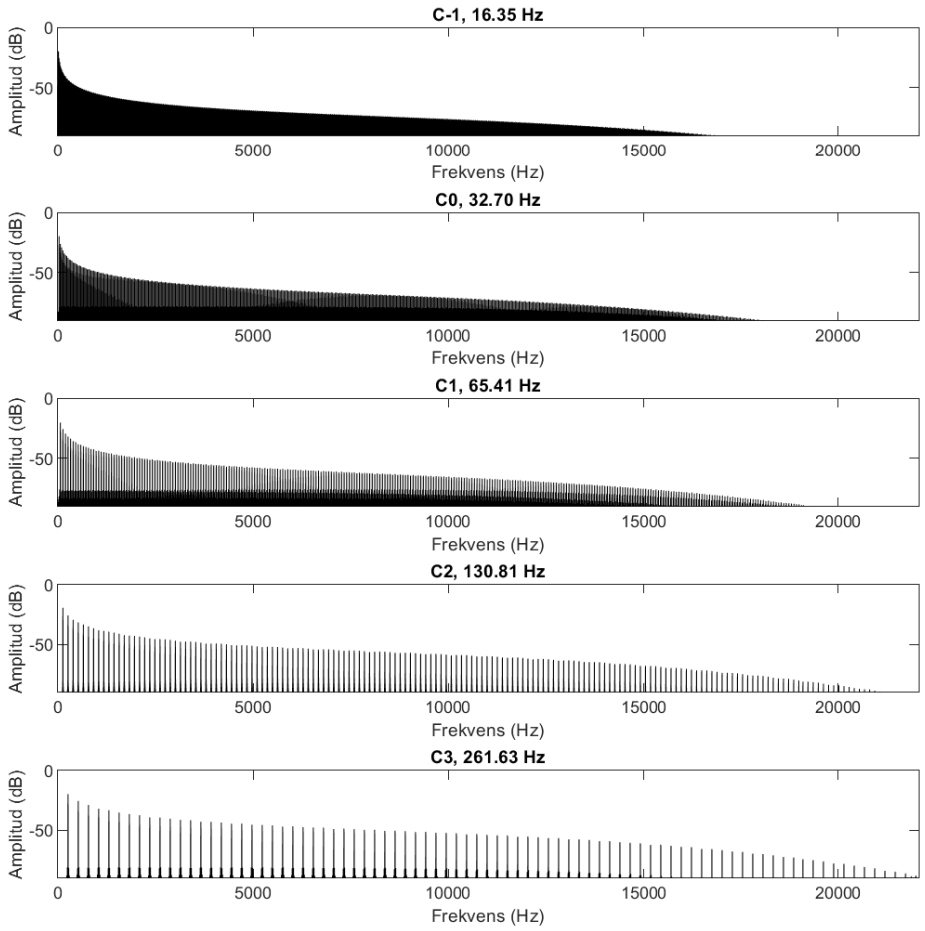
I detta kapitel presenteras examensarbetets resultat uppdelat i de tre olika faserna Implementation av vikningsreducerad sågtandsoscillator, Analys av analoga vågformer samt Modellering för att uppnå önskad karaktär.

5.1 Fas ett: Implementation av vikningsreducerad sågtandsoscilltor

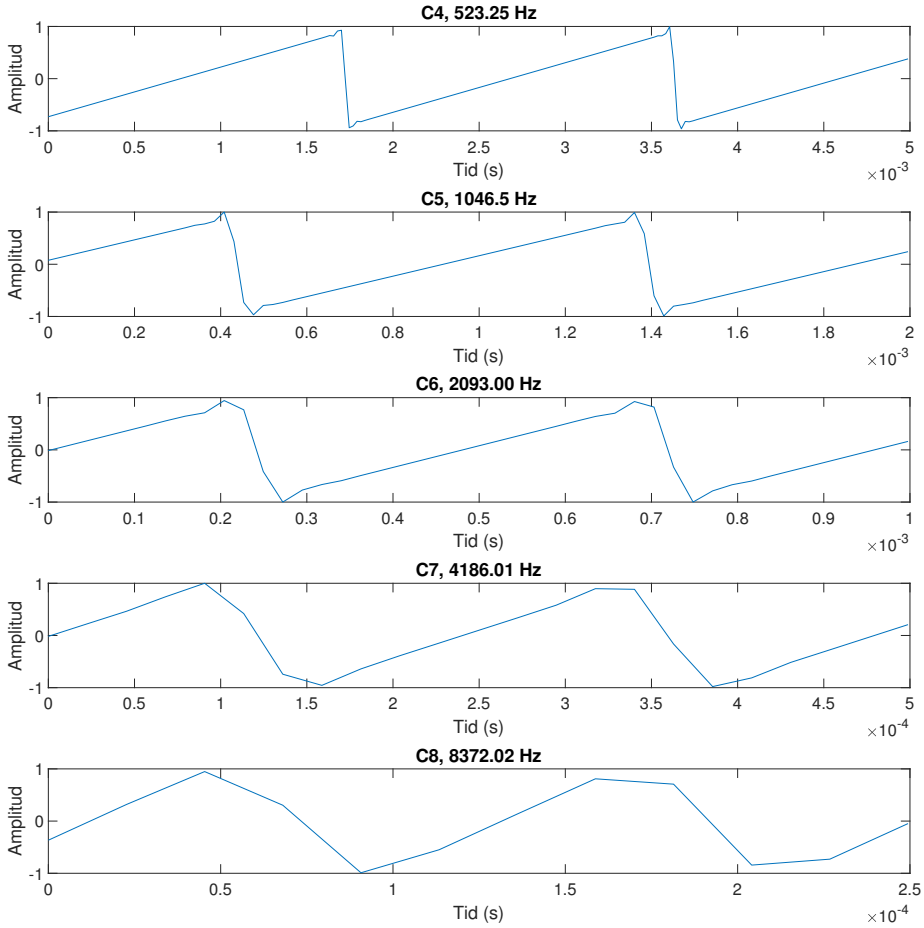
I detta avsnitt presenteras resultatet ifrån fas ett, det vill säga applicering av BLEP-algoritmen på en ideal sågtandsoscillator.



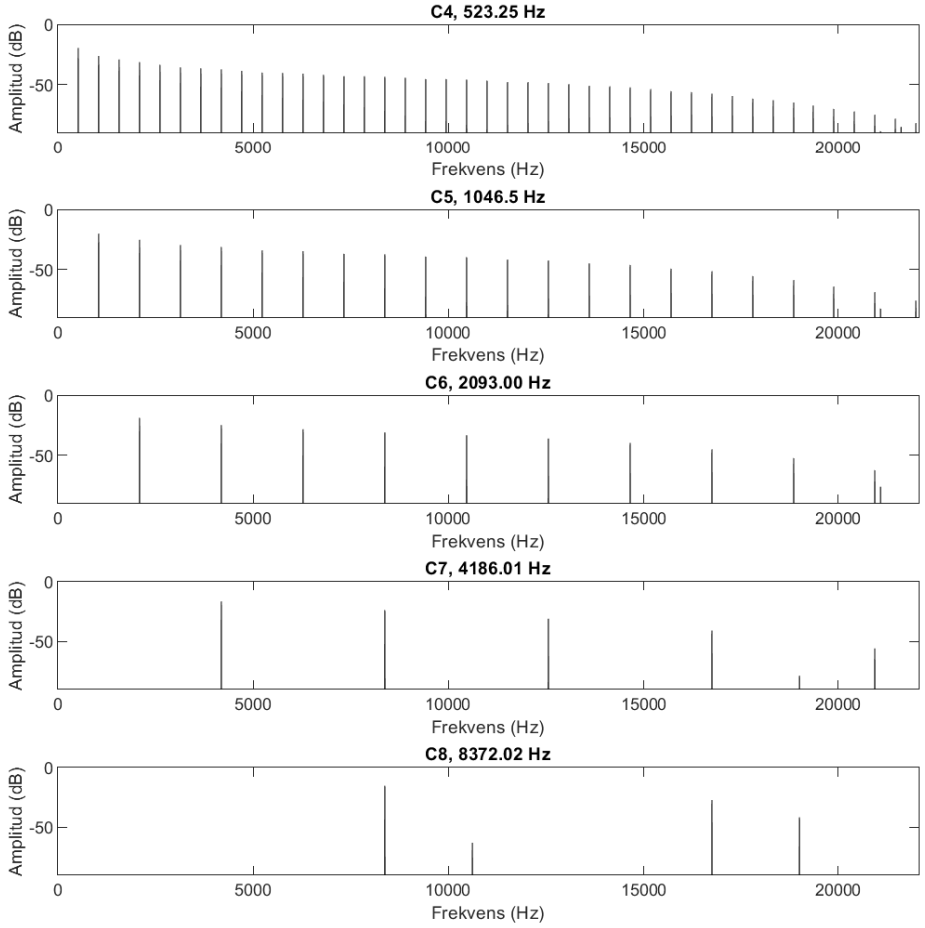
Figur 12 – Resultat av BLEP applicerat på ideal sågtandsoscillator. Från ton C-1 till C3 i tidsdomän.



Figur 13 – Resultat av BLEP applicerat på ideal sågtandsoscillator. Från ton C-1 till C3 i frekvensdomän.



Figur 14 – Resultat av BLEP applicerat på ideal sågtandsoscillator. Från ton C4 till C8 i tidsdomän.



Figur 15 – Resultat av BLEP applicerat på ideal sågtandsoscillator. Från ton C4 till C8 i frekvensdomän.

Tabell 1 – BLEP parametrar vid olika frekvensspan

Frekvensspan	Fönstertyp	Skärningar med noll	Korrigerade punkter (per sida)
C-2 - B6	Blackman-Harris	5	7
C7 - D7	Blackman-Harris	3	5
F7 - B7	Blackman	1	3
C8 - G8	Blackman	1	2

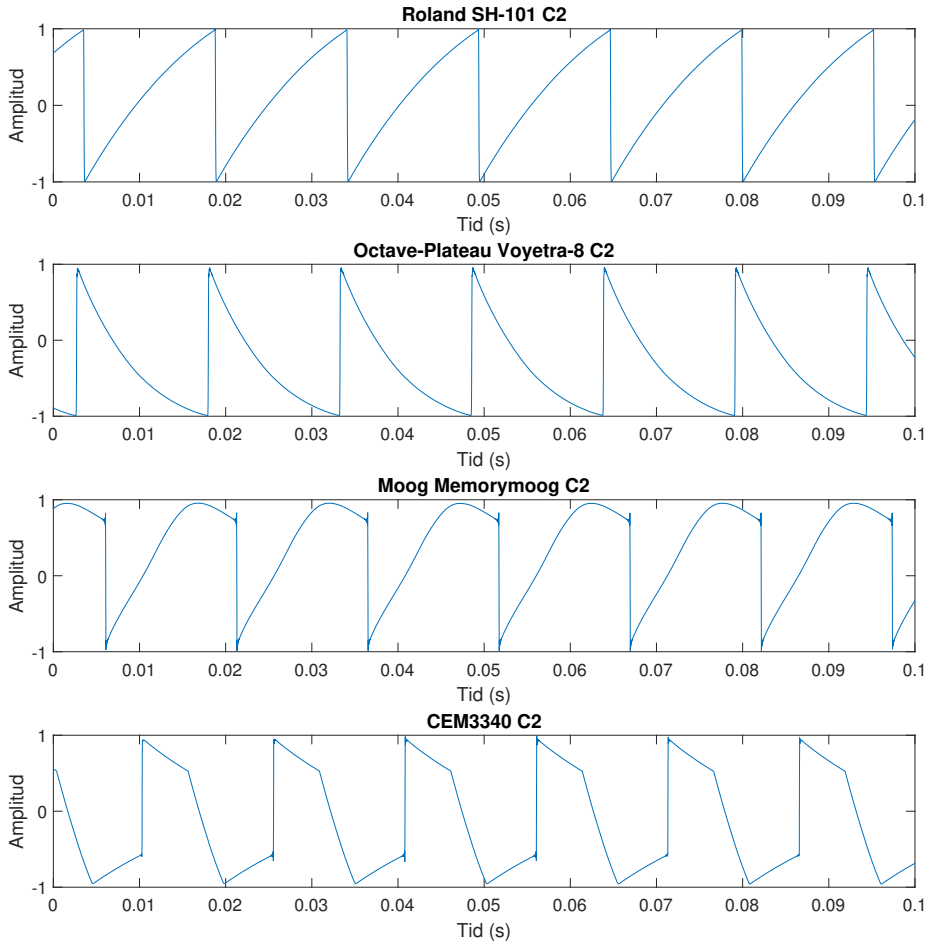
5.1.1 Diskussion kring resultat

Efter testning av parametrar till BLEP-algoritmen erhålldes slutligen parametrar som resulterade i en vikningsfri signal från C-2 (8.18 Hz) till B6 (3951.07 Hz). För detta frekvensomfång räckte det med en BLEP-tabell, men för att förbättra resultaten efter detta frekvensomfång krävdes ytterligare två BLEP-tabeller, vilka tabeller som användes går att utläsa av Tabell 1. Efter det första frekvensomfånget finns det några vikningskomponenter med amplitud runt -50 dB, det är alltså inte helt vikningsreducerat enligt siffrorna som presenterades i metodkapitlet. Figur 15 illustrerar två av dessa komponenter i C8's frekvensspektrum nära 10 och 20 kHz. Vikningsreduceringen lyckades alltså inte uppfylla examensarbetets mål.

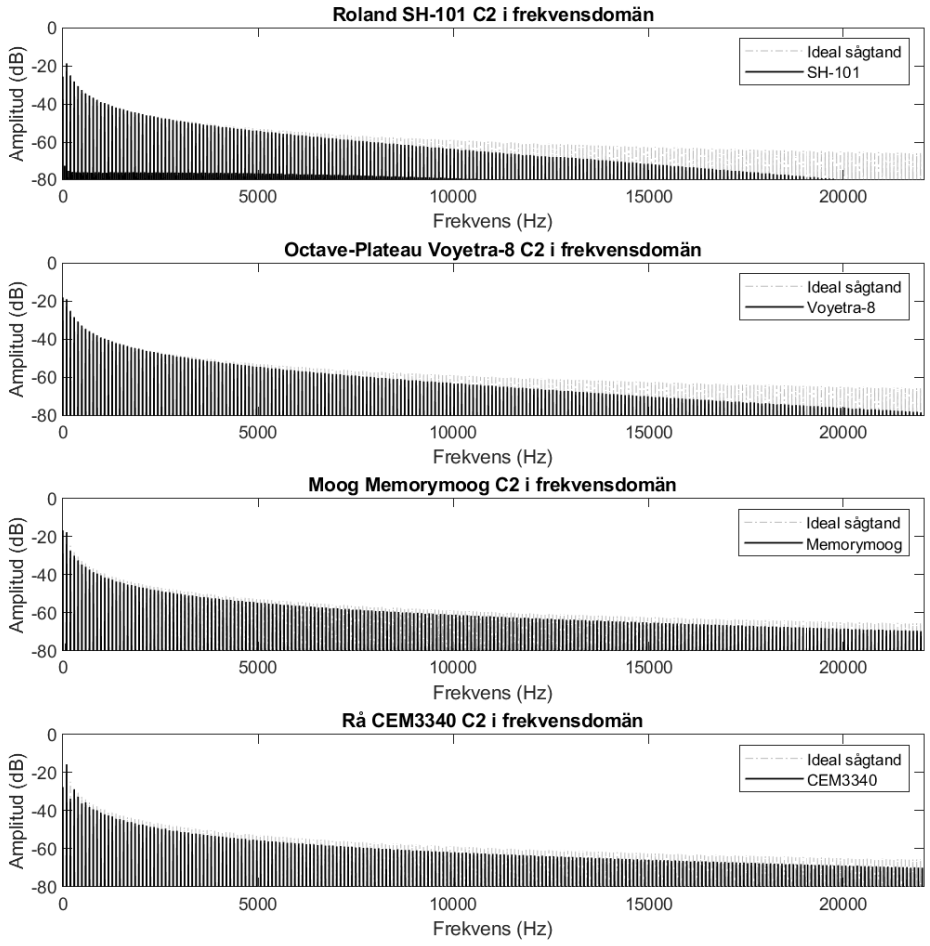
Resultatet av BLEP-algoritmen i tidsdomän illustreras av Figur 12 och 14. Vågformerna i Figur 12 påminner mycket om en ideal sågtand, kollar man noggrannt går det dock att se förändringar av samplingar där en ideal sågtand skulle haft diskontinuitet. Detta syns ännu tydligare av Figur 14 då vågformerna har högre frekvens vilket medför färre samplingar under en period. Förändringar av samples blir alltså mer påtagliga vid högre frekvenser.

5.2 Fas två: Analys av analoga vågformer

I detta avsnitt presenteras resultatet av fas två, det vill säga analysen av några analoga syntars sågtandsvågformer. Nedan visas resultatet för fyra sågtandsvågformer med tonfrekvens 65,4064 Hz, för ytterligare resultat se Bilaga B och C.



Figur 16 – Fyra sågtandsvågformer i tidsdomän från tre analoga syntar samt sågtandsvågform direkt ifrån CEM3340, alla med tonnamn C2 och tonfrekvens 65,4064 Hz



Figur 17 – Fyra sågtandsvågformer i frekvensdomän från tre analoga syntar samt sågtandsvågform direkt ifrån CEM3340, alla med tonnamn C2 och tonfrekvens 65,4064 Hz

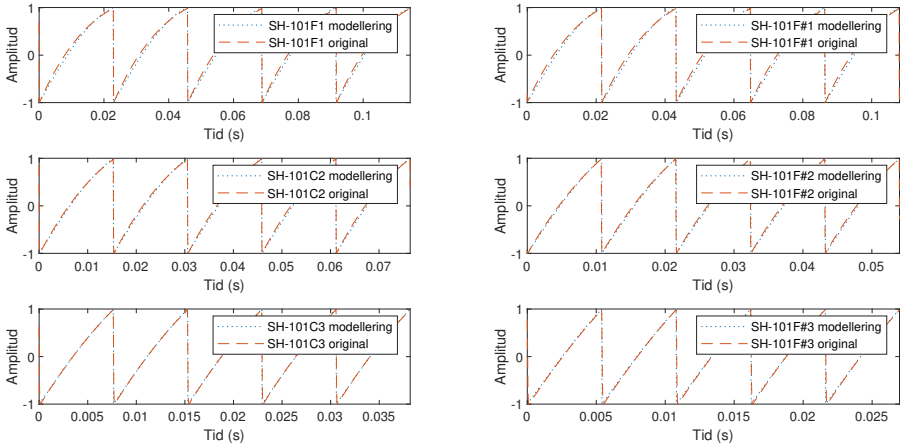
5.2.1 Diskussion kring resultat

Plottning av de tre olika syntarna sågtandsvågformer visar att trots att signalerna har ursprung från samma sorts krets skiljer sig signalernas karakteristik åt. Det är dock värt att nämna att Roland SH-101's sågtand multiplicerad med -1 har stora likheter med Octave Plateau Voyetra 8's sågtand både i tidsdomän och frekvensdomän, se Bilaga C för en illustration av detta. En förklaring till att vågformer i dess orginalform (inte muliplicerade med -1) skiljer sig åt är att tillverkarna av syntarna har adderat elektroniska komponenter till den intregrerade kretsen CEM3340. För två exempel på detta se Bilaga F. Synttillverkarna använder alltså CEM3340 som grund till sina oscillatorer men manipulerar CEM3340 signalerna på olika sätt. Anledningen till detta är inte något detta examensarbete har för avsikt att gå in djupare på.

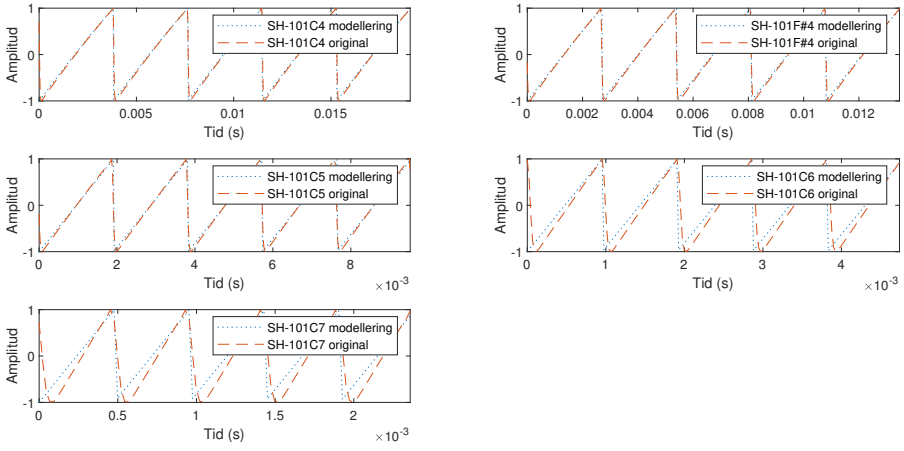
Ifrån plotterna i tidsdomän går det att utläsa att Roland SH-101 och Octave Plateau Voyetra 8 är de som mest liknar en ideal sågtandsvågform och medans Moog Memorymoog och CEM3340 är relativt olika en ideal sågtandsvågform. Trots detta är det Moog Memorymoog och CEM3340 de som har mest lika frevensinnehåll med en ideal sågtand. I frekvensdomän avtar amplituden för Roland SH-101 och Octave Plateau Voyetra 8 övertoner brantare än för Moog Memorymoog och CEM3340.

5.3 Fas tre: Modellering och implementering för att uppnå önskad karaktär

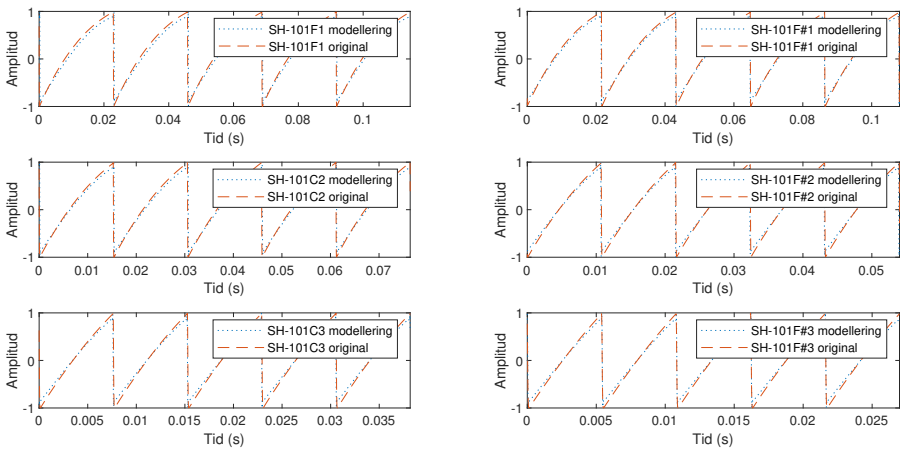
I detta avsnitt presenteras resultatet ifrån fas tre, det vill säga modellering och implementering av Roland SH-101 sågtandsoscillator.



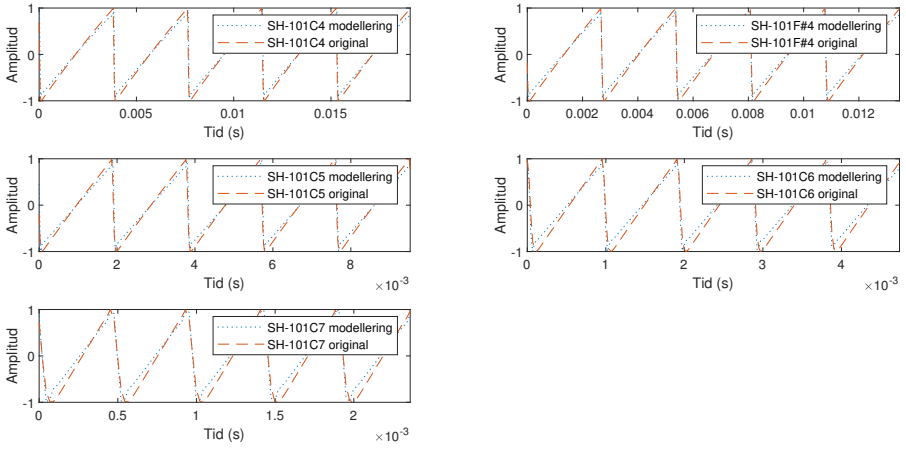
Figur 18 – Jämförelse av Roland SH-101 sågtandsoscillator och detta examenarbetes implementering av Roland SH-101 sågtandsoscillator i tidsdomän utan BLEP applicerat. Figuren illustrerar vågformer för tonerna F1,F#1,C2,F#2,C3 och F#3 från vänster till höger. Streckad linje är original och prickad linje är modellering.



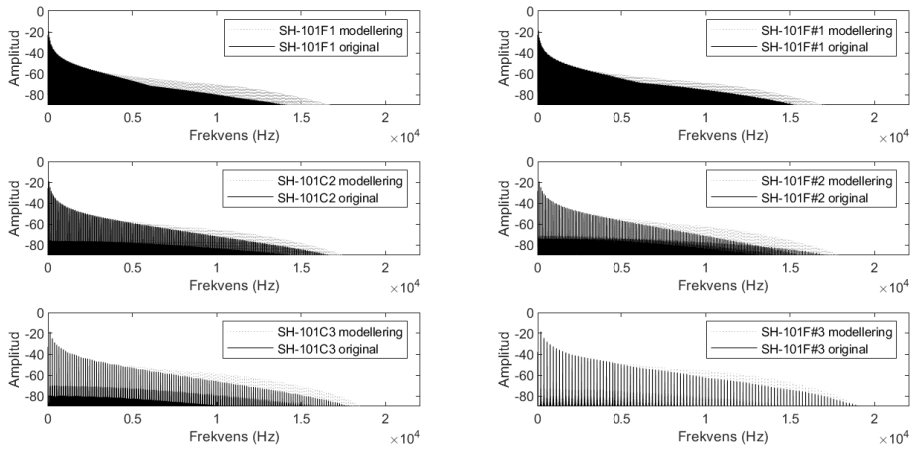
Figur 19 – Jämförelse av Roland SH-101 sågtandsoscillator och detta examenarbetes implementering av Roland SH-101 sågtandsoscillator i tidsdomän utan BLEP applicerat. Figuren illustrerar vågformer för tonerna C4,F#4,C5,C6 och C7 från vänster till höger. Streckad linje är original och prickad linje är modellering.



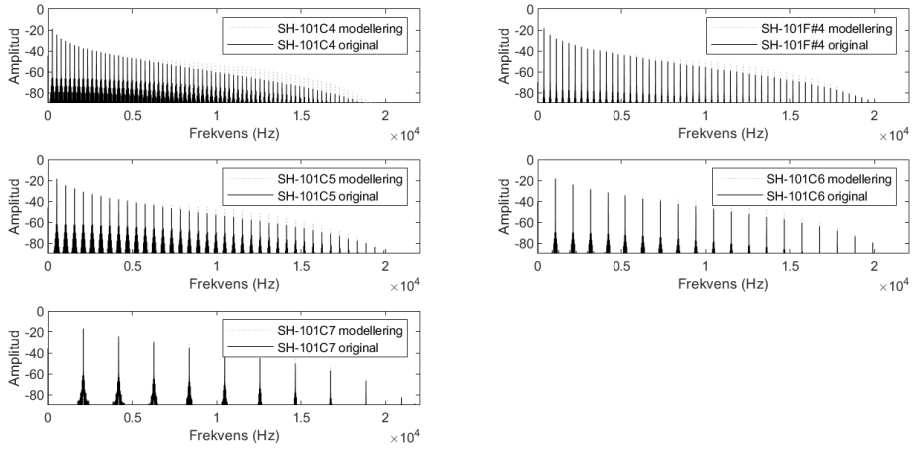
Figur 20 – Jämförelse av Roland SH-101 sågtandsoscillator och detta examenarbetes implementering av Roland SH-101 sågtandsoscillator i tidsdomän med BLEP applicerat. Figuren illustrerar vågformer för tonerna F1,F#1,C2,F#2,C3 och F#3 från vänster till höger.Streckad linje är original och prickad linje är modellering.



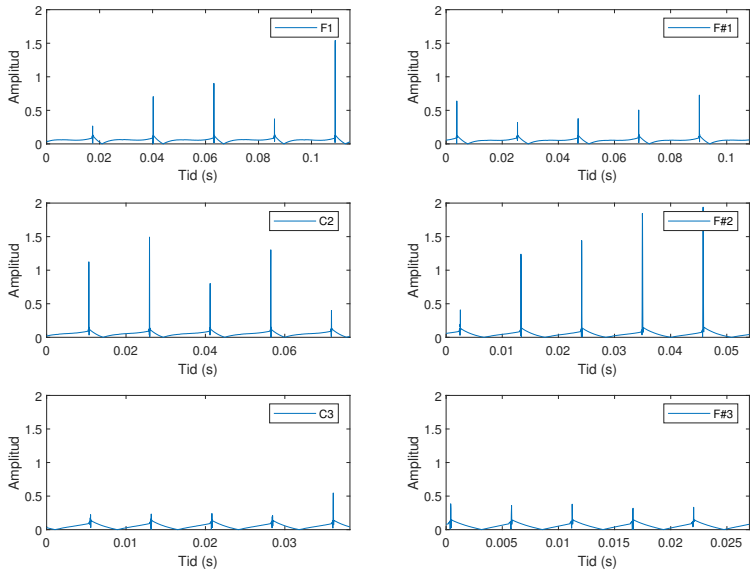
Figur 21 – Jämförelse av Roland SH-101 sågtandsoscillator och detta examenarbetes implementering av Roland SH-101 sågtandsoscillator i tidsdomän med BLEP applicerat. Figuren illustrerar vågformer för tonerna C4,F#4,C5,C6 och C7 från vänster till höger. Streckad linje är original och prickad linje är modellering.



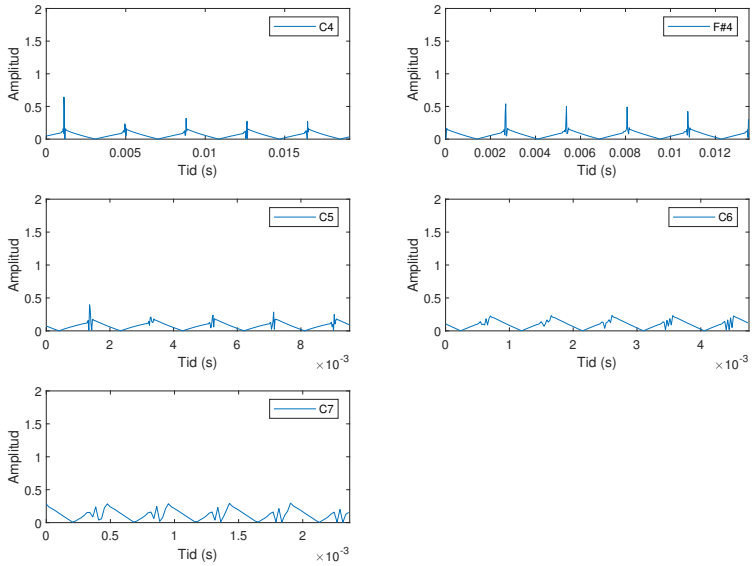
Figur 22 – Jämförelse av Roland SH-101 sågtandsoscillator och detta examenarbetes implementering av Roland SH-101 sågtandsoscillator i frekvensdomän med BLEP applicerat. Figuren illustrerar vågformer för tonerna F1,F#1,C2,F#2,C3 och F#3 från vänster till höger.Streckad linje är original och prickad linje är modellering.



Figur 23 – Jämförelse av Roland SH-101 sågtandsoscillator och detta examenarbetes implementering av Roland SH-101 sågtandsoscillator i frekvensdomän med BLEP applicerat. Figuren illustrerar vågformer för tonerna C4,F#4,C5,C6 och C7 från vänster till höger. Streckad linje är original och prickad linje är modellering.



Figur 24 – Absolutbelopp av skillnad i amplitud mellan Roland SH-101 sågtandsoscillator och detta examenarbetes implementering av Roland SH-101 sågtandsoscillator i tidsdomän med BLEP applicerat. Figuren illustrerar skillnaden för tonerna F1,F#1,C2,F#2,C3 och F#3 från vänster till höger.



Figur 25 – Absolutbelopp av skillnad i amplitud mellan Roland SH-101 sågtandsoscillator och detta examenarbetes implementering av Roland SH-101 sågtandsoscillator i tidsdomän med BLEP applicerat. Figuren illustrerar skillnaden för tonerna C4,F#4,C5,C6 och C7 från vänster till höger.

5.3.1 Diskussion kring resultat

När metoden för att modellera Roland SH-101 tagits fram implementerades denna och implementationen jämfördes med ljudexempeln av Roland SH-101 i tidsdomän. En jämförelse i tidsdomän gjordes först innan BLEP-algoritmen applicerats (Figur 18 och 19) och sedan gjordes ytterligare en efter BLEP-algoritmen applicerats (Figur 20 och 21). Innan BLEP applicerades gjordes bedömningen att implementationen gav godtagbara resultat. Med BLEP applicerat gavs liknande resultat men på grund av att BLEP-algoritmen förändrar samples kring sågtandvågformens diskontinuitet minskade likheterna något när BLEP applicerats. För att tydligöra skillnader i tidsdomän illustrerar Figur 24 och 25 absolutbeloppet av skillnaden i amplitud mellan implementation och ljudexempeln av Roland SH-101 i tidsdomän.

För en jämförelse av implementationen av Roland SH-101 och ljudexempeln av Roland SH-101 i frekvensdomän efter BLEP applicerats se Figur 22 och 23. Figur 22 och 23 visar att implementation har något högre amplitud för övertoner vid högre frekvenser än orignalet. Skillnaden i amplitud är störst för grundtoner med lägre frekvens.

6 Slutsats

Detta kapitel tar upp framtida utveckligsmöjligheter, ger svar på problemformuleringen samt ger en reflektion över etiska aspekter.

6.1 Svar på problemformulering

I detta avsnitt ges svar på de frågor som formulerades under examensarbetets uppstart.

1. Vad karakteriserar olika analoga syntars sågtandsvågformer

Det som karakteriserar de sågtandsvågformer som analyserats i tidsdomänen är att de alla skiljer sig åt i utseende. Det går inte att beskriva något särskilt drag mer än att de alla skiljer sig i utseende från varandra samt från en ideal sågtandsvågform. Det är dock värt att nämna att Roland SH-101's sågtand multiplicerad med -1 har stora likheter med Octave Plateau Voyetra 8's sågtand både i tidsdomän och frekvensdomän. I frekvensdomän utmärker sig alla fyra analyserade analoga sågtandsvågformer med en snabbare avtagande amplitud hos dess övertoner än en ideal sågtandsvågform. Detta betyder att man kan beskriva de analoga syntarnas sågtandsvågformer som lågpassfiltrerade ideala sågtandsvågformer.

2. Vad finns det för metoder för att bandbreddsbegränsa en analog signal med syfte att minimera vikningseffekten som uppstår när signalen samplas och hur kan dessa metoder förbättras?

Att ge ytterliggare en beskrivning utöver den sammanfattning som ges i teknisk bakgrund om vilka metoder om finns för att undvika vikning anses vara överflödigt för detta examensarbete. Det hade dock varit relevant att försöka förbättra en av dessa metoder utifrån de kriterier som beskrivs i bakgrund. Men på grund av exmamensarbetets tidsbegränsning är inte detta något som varit möjligt att utföra.

3. Vilket ramverk är lämpligast när VST-synten implementeras för att mest fokus ska ligga på att ta fram algoritmer för oscillator och mindre fokus på grafiskt användargränsntt?

På grund av att två ramverk valdes ut och det ena tidigt ställde till problem som medförde att detta ramverk fick utelämnas gjordes det aldrig några jämförelser mellan olika ramverk. Således kan detta examensarbete inte ge svar på denna frågan. JUCE visade sig dock vara ett lämpligt alternativ för detta examensarbete på grund av dess stabilitet samt detaljerade tutorials, men det går inte att utesluta att det finns fler lika lämpliga ramverk.

6.2 Reflektion över etiska aspekter

Modelleringen i detta examensarbete går ut på att efterlikna eller kopiera karaktären av någon kommersiell produkt. Produkterna som analyseras och modelleras i detta examensarbete är äldre syntar som inte produceras längre samt vars patent gått ut. Därför görs bedömingen att detta examensarbetes modellering inte kan skada eller orsaka förlust för företagen bakom produkterna.

6.3 Framtida utvecklingsmöjligheter

På grund av att vikningsreduceringen under fas ett inte nådde upp till examenarbetets mål samt att en ad hoc modelleringsmetod användes istället för en allmän modelleringsmetod under fas tre lämnar detta examensarbete utrymme för framtida utvecklingsmöjligheter. I detta avsnitt ges några förslag på möjliga förbättringar under fas ett och tre.

6.3.1 Fas ett: Implementation av vikningsreducerad sågtandsoscillator

I (Valencia Otalvaro 2015) bekrivs fönsterfunktioner som ger bättre resultat än Blackman och Blackman-Harris, dessa utelämnades i detta examensarbete på grund av att de ansågs för tidskrävade att återskapa. De fönsterfunktionerna som beskrevs i (Valencia Otalvaro 2015) var snävare än Blackman och Blackman-Harris. Vid fortsatt utveckling av BLEP-algoritmen som använts i detta examensarbete bör snävare fönsterfunktioner ge bättre resultat än vad som åstadkommits under detta examensarbete.

Under detta examensarbete användes linjär interpolation för att approximera ett värde mellan två kända värden i BLEP-tabellen. Det finns mer avancerade interpoleringsmetoder som kanske ger bättre resultat vad gäller vikningsreducering. Ett förslag vid fortsatt utveckling av BLEP-algoritmen som använts i detta examensarbete är att testa andra interpoleringsmetoder.

6.3.2 Fas två: Analys av analoga vågformer

Under detta examensarbete har ingen mätning av skillnaden mellan en ideal sågtandsvågform och sågtandsvågformer från analoga syntar utförts. Vid fortsatt arbete är ett förslag att någon form av mätning som t ex root-mean-square error (RMSE) utförs för att ytterligare tydliggöra skillnader mellan de olika vågformerna.

6.3.3 Fas tre: Modellering och implementering för att uppnå önskad karaktär

I (Pekonen, Lazzarini m. fl. 2011) presenteras en metod som går ut på att matcha en analog vågforms frekvensspektrum genom att filtrera en redan vikningsreducerad sågtandsoscillator tills önskat frekvensspektrum uppnåts. Denna metod var först tänkt att testas under detta examensarbete men fick utelämnas på grund av tidsbegränsning. Motiveringen till att detta är en mer lämpad metod är att det är en allmän metod istället för en ad hoc metod. Ett förslag är att testa denna metod vid framtida modelleringar av Roland SH-101's sågtandsoscillator.

7 Terminologi

7.1 Termer

- *Oscillator* - En synts ljudkälla, kan generera en mängd olika periodiska vågformer
- *Delton* - Enkel sinuston som tillsammans med andra deltoner bildar en sammansatt ton
- *Grundton* - Den lägsta deltonen i en sammansatt ton
- *Överton* - Alla toner utöver grundtonen i en sammansatt ton
- *Nyquistfrekvensen* - Halva samplingsfrekvensen för en signal
- *Faltning* - Matematisk operation som används när filter appliceras i signalbehandling
- *DC-förskjutning* - En signals förskjutning ifrån amplitudens noll-referens
- *MIDI* - Protokoll samt gränssnitt för kommunikation mellan datorer och musikinstrument
- *Bipolär sågtandsvågform* - Sågtandsvågform som har -1 som lägsta amplitud och 1 som högsta

7.2 Symboler

- $\triangleq$ - definierat lika
- $f[n]$ - som f(t) fast för diskreta värden

Referenser

Brandt, Eli (2001). "Hard sync without aliasing". I: *Proceedings of the International Computer Music Conference*.

Frei, Beat (2010). *Digital sound generation*.

Leary, Andrew B. och Charles T. Bright (2009). "Bandlimited digital synthesis of analog waveforms". Amerikanskt patent US7589272B2.

Pekonen, Jussi (2014). "Filter-Based Oscillator Algorithms for Virtual Analog Synthesis". I: Aalto University publication series DOCTORAL DISSERTATIONS; 26/2014.

Pekonen, Jussi, Victor Lazzarini m. fl. (2011). "Discrete-Time Modelling of the Moog Sawtooth Oscillator Waveform". I:

Pekonen, Jussi och Vesa Välimäki (2011). "The Brief History of Virtual Analog Synthesis". I: *Proceedings of Forum Acusticum*.

Pirkle, Will (2014). *Designing software synthesizer plug-ins in C++: for RackAFX, VST3, and Audio Units*. Routledge.

Scavone, Gary (2002). *Bandlimited Synthesis*. URL: `https://www.music.mcgill.ca/~gary/307/week5/bandlimited.html`.

Stilson, Tim och III Smith Julius O. (1996). "Alias-free digital synthesis of classic analog waveforms." I:

Valencia Otalvaro, Francisco J. (2015). "Aliasing and Harmonic Decay Control of the Bandlimited Step Method Using Psychoacoustic Models and the Genetic Algorithm". Examensarb. University of Miami.

Välimäki, Vesa och A Huovilainen (2007). "Antialiasing Oscillators in Subtractive Synthesis." I: *IEEE Signal Processing Magazine, Signal Processing Magazine, IEEE, IEEE Signal Process. Mag* 2. ISSN: 1053-5888.

Bilaga A BLEP kodexempel

```
inline double BLEP(const bool &interpolate, const bool &
    useInternalSettings, const double &modulo, double &tableSize,
    double &tableIndex, int &correctionPoints, const double &inc)
{
    // -- returvärde
    double blep = 0.0;

    // t = avstånd från diskontinuitet
    double t = 0.0;

    // -- ta fram mitten av tabell (diskontinuiteten)
    double tableCenter = tableSize / 2.0 - 1;

    // -- använd interna inställningar
    if (useInternalSettings)
    {
        if (inc < 0.09)
        {
            tableSize = 32768;
            tableIndex = 12;
            correctionPoints = 7;
        }
        else if (inc >= 0.09 && inc < 0.12)
        {
            tableSize = 32768;
            tableIndex = 10;
            correctionPoints = 5;
        }
        else if (inc >= 0.12 && inc < 0.18)
        {
            tableSize = 32768;
            tableIndex = 0;
            correctionPoints = 3;
        }
        else if (inc >= 0.18)
        {
            tableSize = 32768;
            tableIndex = 0;
            correctionPoints = 2;
        }
    }

    // -- till vänster om diskontinuitet
    // -1 < t < 0
    for (unsigned int i = 1; i <= (unsigned int)correctionPoints;
     i++)
    {
        if (modulo > 1.0 - (double)i * inc)
        {
            // -- räkna ut avstånd från diskontinuitet från vänster
            t = (modulo - 1.0) / (correctionPoints * inc);
```

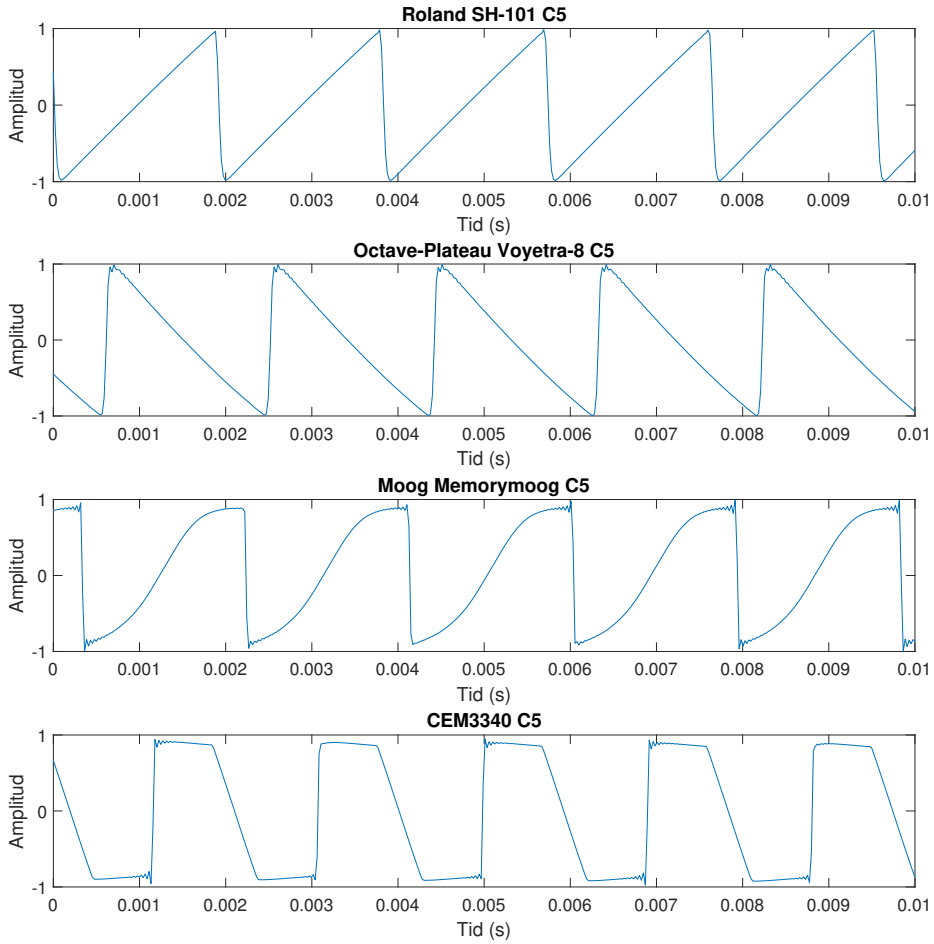
```
49
50              // -- räkna ut residual index för vänster sida
51              double decimalIndex = (1.0 + t) * tableCenter;
52              int floorIndex = (int)decimalIndex;
53
54              blep = getTableValue(interpolate, decimalIndex,
    floorIndex, tableSize, tableIndex);
55
56              // -- subtrahera pga fallande efter diskontinuitet
57              blep *= -1.0;
58
59              return blep;
60          }
61      }
62
63      // -- till höger om diskontinuitet
64      // 0 <= t < 1
65      for (unsigned int i = 1; i <= (unsigned int)correctionPoints;
       i++)
66      {
67          if (modulo < (double)i * inc)
68          {
69              // -- räkna ut avstånd från diskontinuitet från höger
70              t = modulo / (correctionPoints * inc);
71
72              // -- räkna ut residual index för höger sida
73              double decimalIndex = t * tableCenter + (tableCenter
    + 1.0);
74              int floorIndex = (int)decimalIndex;
75
76              blep = getTableValue(interpolate, decimalIndex,
    floorIndex, tableSize, tableIndex);
77
78              // -- subtrahera pga fallande efter diskontinuitet
79              blep *= -1.0;
80
81              return blep;
82          }
83      }
84
85      // -- ingen BLEP residual
86      return 0.0;
87  }
88
89  double getTableValue(const bool &interpolate, const double &
    decimalIndex, const double &floorIndex, const int &tableSize,
    const int &tableIndex)
90  {
91      double blep = 0.0;
92
93      // -- linjär interpolering
94      if (interpolate)
95      {
```

```
96          double firstSample{};
97          double secondSample{};
98          switch (tableSize)
99          {
100         case 4096:
101             firstSample = residualTable4096[tableIndex][
    floorIndex];
102             secondSample = residualTable4096[tableIndex][
    floorIndex+1];
103             break;
104         case 8192:
105             firstSample = residualTable8192[tableIndex][
    floorIndex];
106             secondSample = residualTable8192[tableIndex][
    floorIndex+1];
107             break;
108         case 16384:
109             firstSample = residualTable16384[tableIndex][
    floorIndex];
110             secondSample = residualTable16384[tableIndex][
    floorIndex+1];
111             break;
112         case 32768:
113             firstSample = residualTable32768[tableIndex][
    floorIndex];
114             secondSample = residualTable32768[tableIndex][
    floorIndex+1]];
115             break;
116         default:
117             break;
118         }
119
120         blep = firstSample + (secondSample - firstSample) * (
    decimalIndex - floorIndex);
121     }
122     // -- ingen interpolering
123     else
124     {
125         switch (tableSize)
126         {
127         case 4096:
128             blep = residualTable4096[tableIndex][floorIndex];
129             break;
130         case 8192:
131             blep = residualTable8192[tableIndex][floorIndex];
132             break;
133         case 16384:
134             blep = residualTable16384[tableIndex][floorIndex];
135             break;
136         case 32768:
137             blep = residualTable32768[tableIndex][floorIndex];
138             break;
139         default:
```

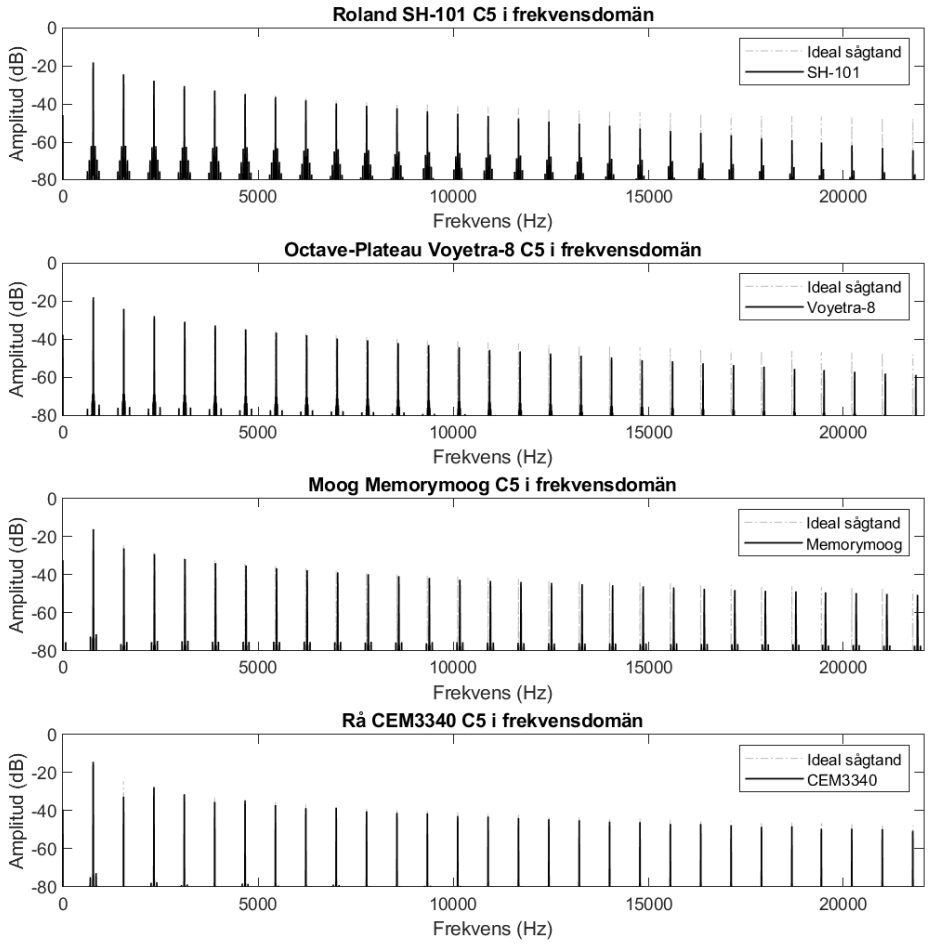
```
140                 break;
141             }
142         }
143         return blep;
144 }
```

Kodexempel 4 – BLEP algoritm i C++

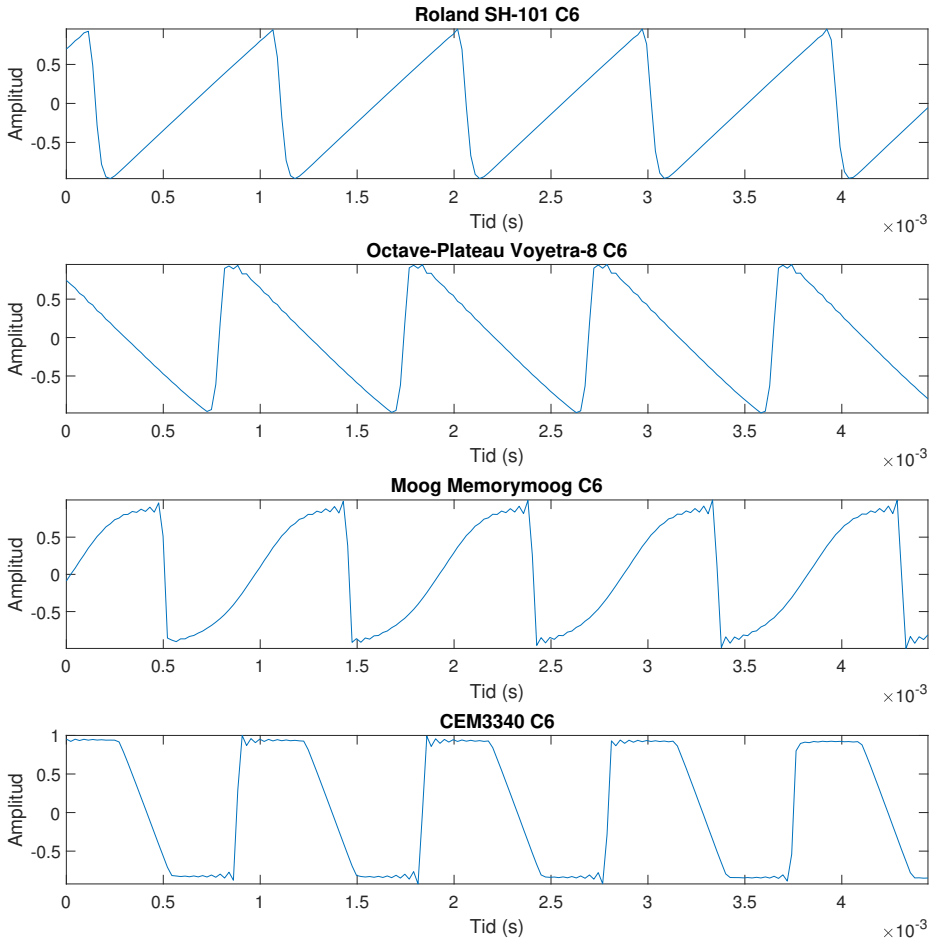
Bilaga B Analys av analoga sågtandsvågformer



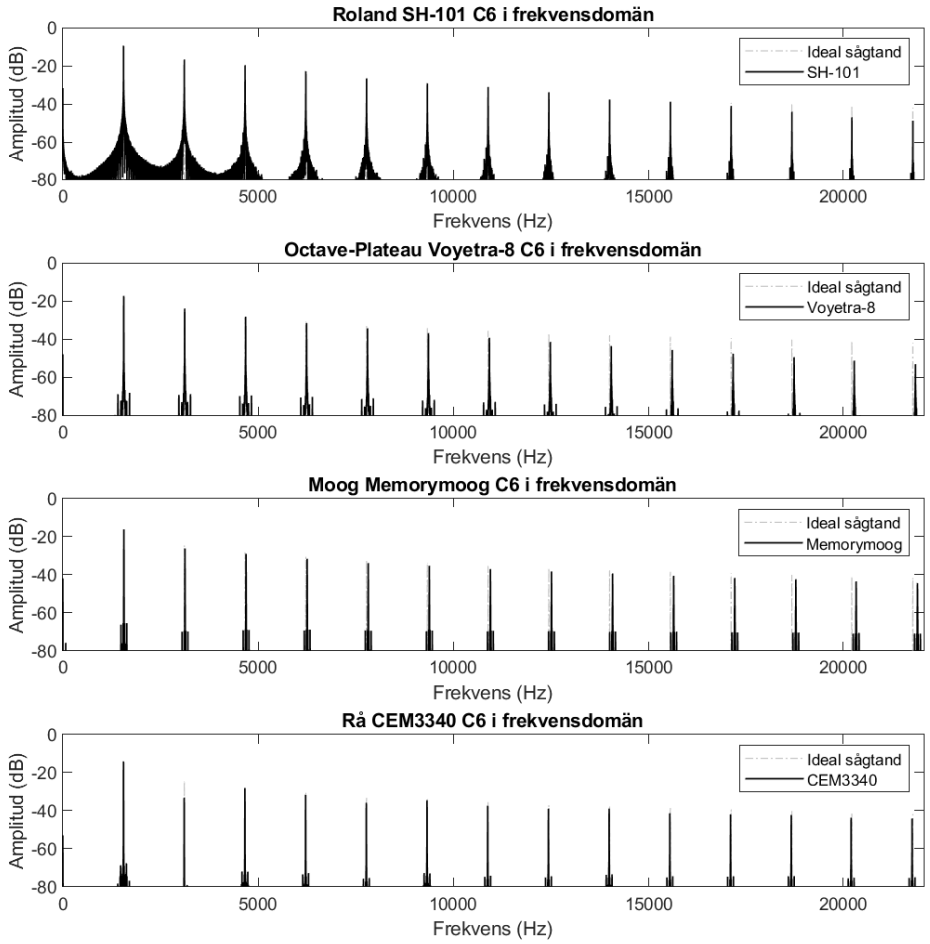
Figur 26 – Fyra sågtandsvågformer i tidsdomän från tre analoga syntar samt sågtandsvågform direkt ifrån CEM3340, alla med tonnamn C5 och tonfrekvens 523,251 Hz



Figur 27 – Fyra sågtandsvågformer i frekvensdomän från tre analoga syntar samt sågtandsvågform direkt ifrån CEM3340, alla med tonnamn C5 och tonfrekvens 523,251 Hz

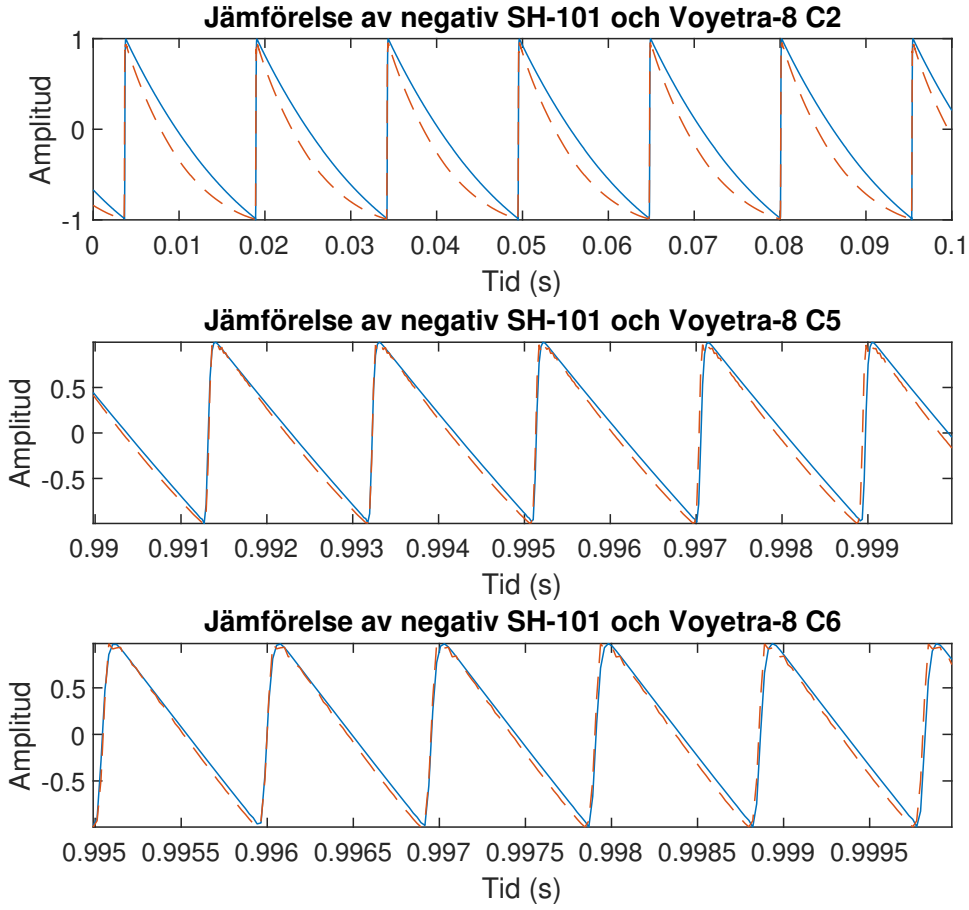


Figur 28 – Fyra sågtandsvågformer i tidsdomän från tre analoga syntar samt sågtandsvågform direkt ifrån CEM3340, alla med tonnamn C6 och tonfrekvens 1046,50 Hz

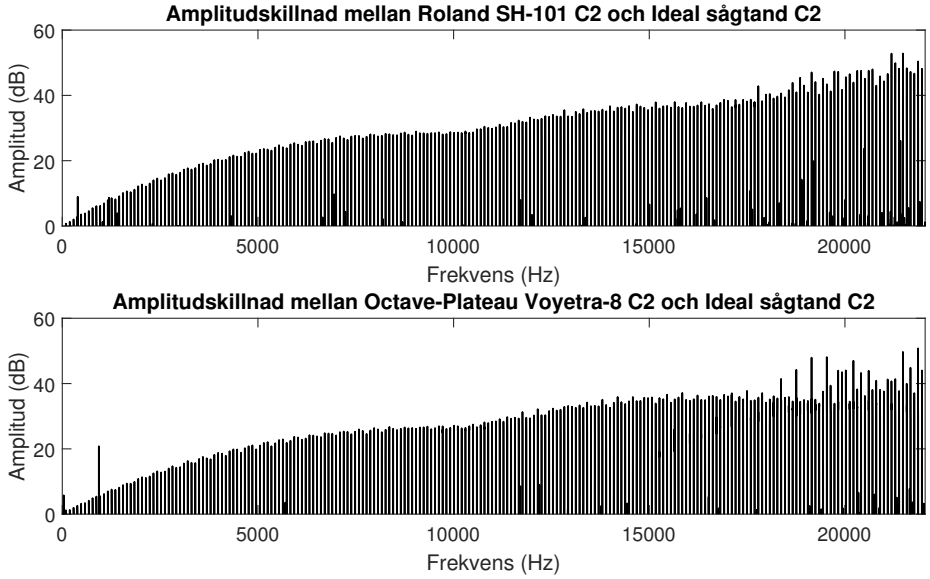


Figur 29 – Fyra sågtandsvågformer i frekvensdomän från tre analoga syntar samt sågtandsvågform direkt ifrån CEM3340, alla med tonnamn C6 och tonfrekvens 1046,50 Hz

Bilaga C Jämförelse Roland SH-101 och Octave Plateau Voyetra 8

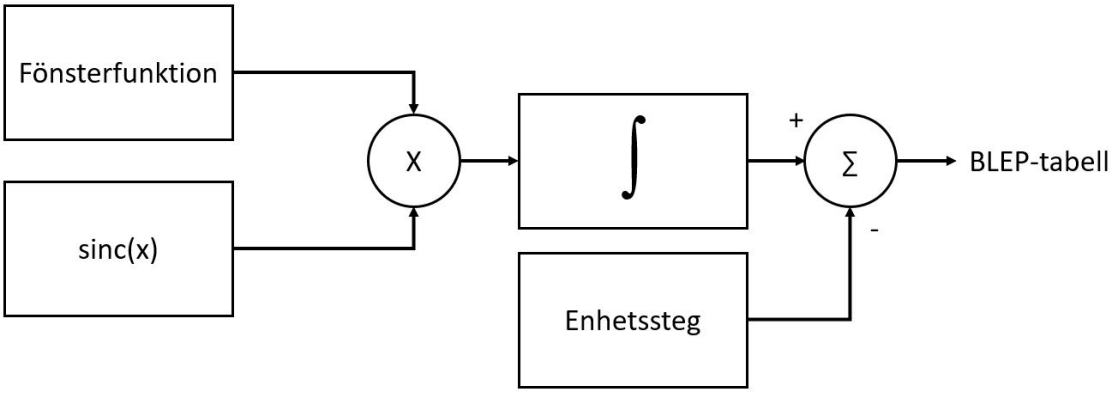


Figur 30 – Jämförelse av negativ Roland SH-101 och Octave Plateau Voyetra 8 i tidsdomän



Figur 31 – Amplitudskillnad mellan Roland SH-101 och ideal sågtand samt Octave Plateau Voyetra 8 och ideal sågtand i frekvensdomän

Bilaga D Algoritm för att ta fram tabellvärden till BLEP



Figur 32 – Illustration av hur BLEP-tabell tagits fram, för detaljerad beskrivning se Kodexempel 5 nedan

```
% Initiering av startvärden
N = 32768 + 1;                %tabellstorlek + 1
fs = N*100;                   %samplingsfrekvens
n = 5;                       %skärningar med tidsaxel per sida

writeBlepTableToFile(N, fs, n, true);

function [] = writeBlepTableToFile(N, fs, n, blackman_harris)
%Tidsvektor
Ts = 1/fs;
T = ((N-1)*Ts)/(2*n);
fc = 1/T;
t = -n*T:Ts:n*T;

% Generera ett impulssvar av ett lågpassfilter, som är en sinc funktion
sinc_impulse = sinc(fc*t);

% Applicera fönsterfunktion
if blackman_harris
    window = blackmanharris(N)';
else
    window = blackman(N)';
end

windowed_sinc = sinc_impulse.*window;

%Integrera windowed sinc
sum = 0;
integral = 1:0:length(windowed_sinc);
for i = 1:length(t)
    sum = sum + windowed_sinc(i);
    integral(i) = sum;
```

```
33 end
34
35 %Konvertera till bipolär
36 bipolar = integral/sum;
37 bipolar = bipolar.*2-1;
38
39
40 %Subtrahera bipolar med ett enhetssteg som är -1 för t < 0 och
41 %1 för alla andra t.
42
43 step = zeros(1,length(t));
44
45 for i = 1:length(t)
46     if t(i) < 0
47         step(i) = -1;
48     else
49         step(i) = 1;
50     end
51 end
52
53 residual = bipolar - step;
54
55 residual(N:N) = [];
56
57 if blackman_harris
58     writematrix(residual,['res_sz' num2str(N-1) '
    _wdwBlackmanHarris_zcps' num2str(n) '.csv']);
59 else
60     writematrix(residual,['res_sz' num2str(N-1) '
    _wdwBlackman_zcps' num2str(n) '.csv']);
61 end
62
63 end
```

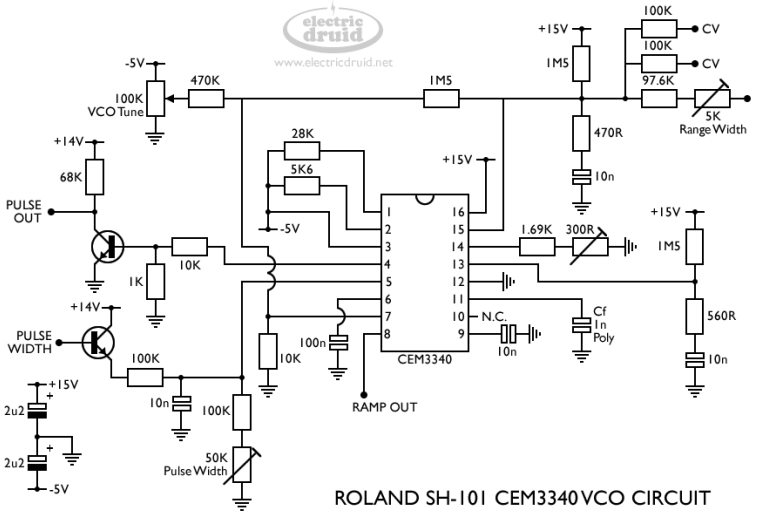
Kodexempel 5 – Algoritm för att ta fram tabellvärden till BLEP

Bilaga E FFT och plot i Matlab

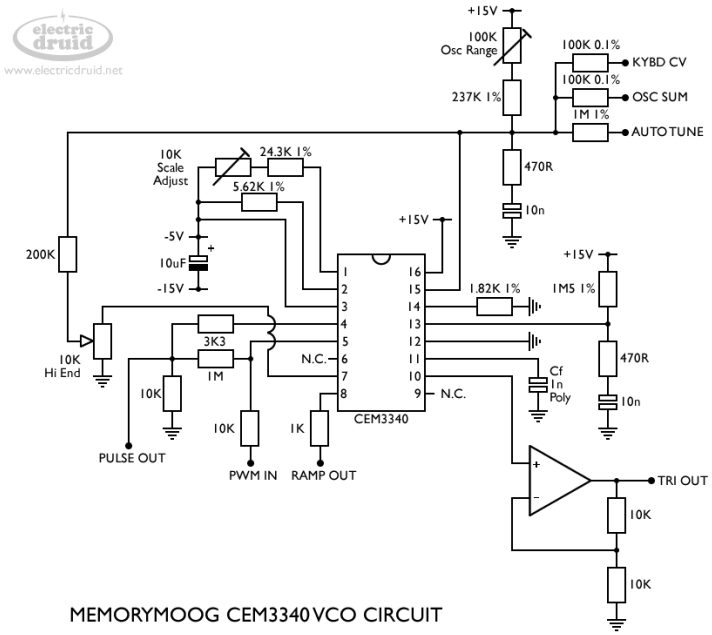
```
% Init
Fs = 44100;
T = 1/Fs;                           %tid för 1 sample
x = 0:T:1-T;                        %tidsvektor
n = length(x);                      %längd på tidsvektor
audiolength = Fs;                   %samples
samples = [1 audiolength];

% Frekvensspecifikationer för FFT:
f = Fs*(0:(n/2))/n;                 %hertz
w = chebwin(n,100);                 %fönsterfunktion

% Ladda audiofiler
[sh101,~] = audioread('sh101C2.wav',samples);
sh101(:,2) = [];                    %ta bort rad 2 pga stereo
sh101 = normalize(sh101, 'range', [-1 1]);

% FFT
sh101 = sh101.*w;                   %applicera fönster innan FFT
SH101 = fft(sh101);    %FFT
SH101stem = abs(SH101/n);
SH101stemdB = mag2db(SH101stem);

% Plot i frekvensdomän
plot(f,SH101stemdB(1:n/2+1))
axis([0 22050 -90 0]);
xlabel('Frekvens (Hz)')
ylabel('Amplitud (dB)')
```

Kodexempel 6 – FFT och plot i Matlab

Bilaga F Kopplingscheman

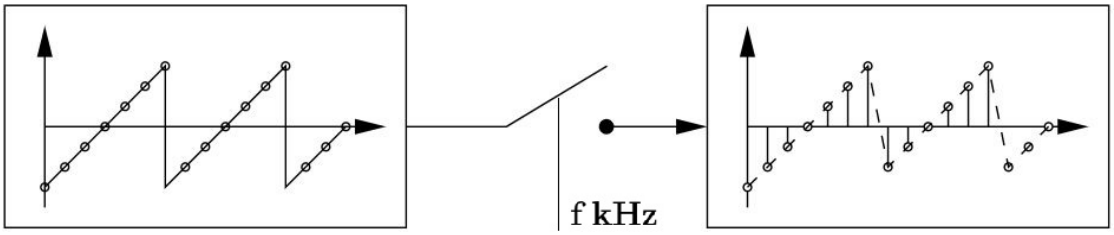


Figur 33 – Roland SH-101 oscillator baserad på CEM3340 [från electricdruid.com, Licensvillkor: https://creativecommons.org/licenses/by-nc-sa/4.0/deed.sv]

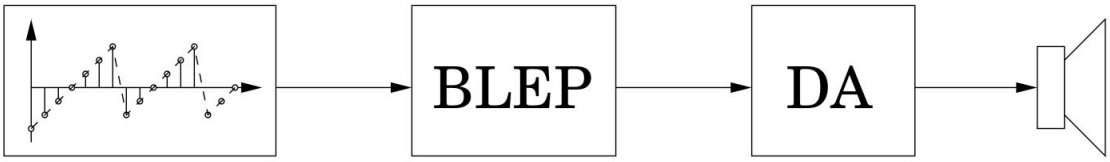


Figur 34 – Moog MemoryMoog oscillator baserad på CEM3340 [från electricdruid.com, Licensvillkor: https://creativecommons.org/licenses/by-nc-sa/4.0/deed.sv]

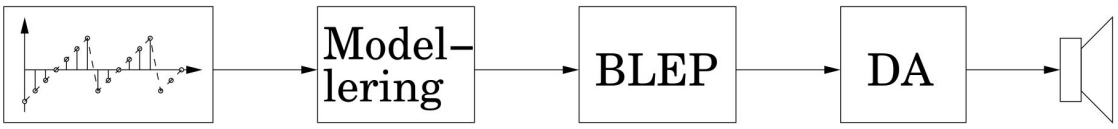
Bilaga G Beskrivning av signalflöde i SAW-X



Figur 35 – Illustration av den sampling som sker i Kodexempel 1, "switchen" mellan blocken symboliserar övergång från analog till diskret signal. eftersom ingen filtrering sker vid sampling så får man ett spektralinnehåll som påminner om det i Figur 2



Figur 36 – Signalflöde hos SAW-X i Fas 1, DA som symboliserar omvandling av digital till analog signal ingår inte i SAW-X signalflöde men är med för att visa att det sker en omvandling innan signalen når högtalare



Figur 37 – Signalflöde hos SAW-X i Fas 3, DA som symboliserar omvandling av digital till analog signal ingår inte i SAW-X signalflöde men är med för att visa att det sker en omvandling innan signalen når högtalare