NTNU

elsys
ELEKTRONISK SYSTEMDESIGN & INNOVASJON

Designnotat

Tittel: Spøkelsestone	
Forfattere: Sindre Mandelid Kvam	
Revisjon: C	Dato: 14. desember 2023

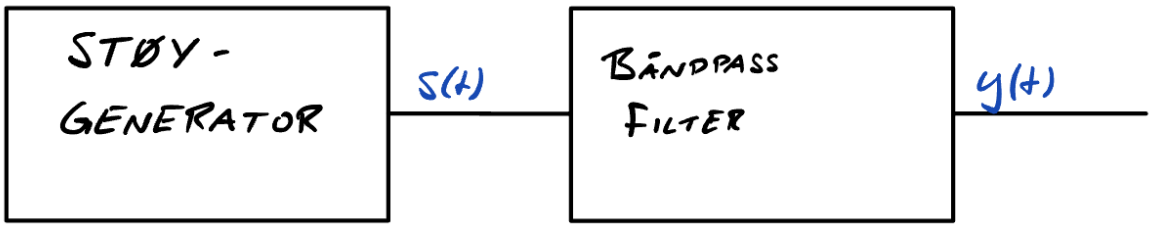
Innhold

1 Brief

Dette designnotatet inneholder nødvendig informasjon for å designe, og å ta i bruk, en støygenerator og et digitalt filter på en FPGA.

2 Problembeskrivelse

Det skal designes et system slik som vist i Figur 1. Systemet består av en støygenerator, som skal generere et signal $s(t)$ som er tilnærmet likt *hvit støy*. Støyen skal båndbegrenses rundt en senterfrekvens $f_0 = 2880\,\text{Hz}$, slik at utgangssignalet $y(t)$ har en hørbar tone. Det skal være mulig å lytte til signalene $s(t)$ og $y(t)$. Kvaliteten på filteret skal kartlegges ved å finne; impulsrespons, båndbredde og Q-faktor til filteret. Systemet skal realiseres på en FPGA.



Figur 1: Blokkskjema av system for generering av båndbegrenset støy.

3 Teori

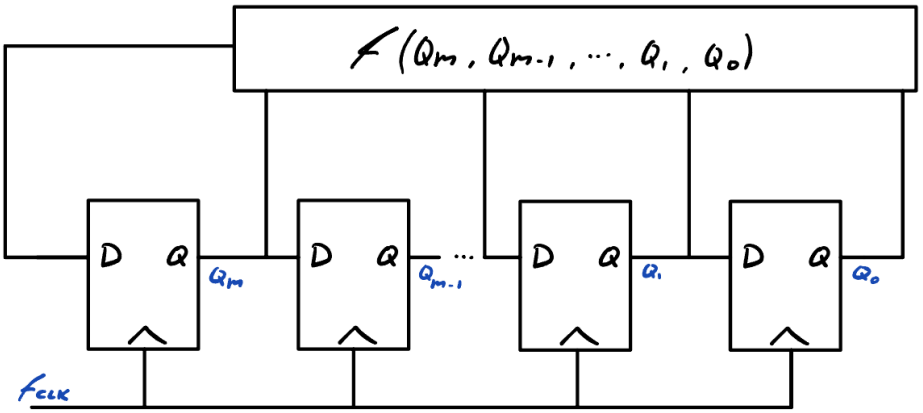
3.1 Hvit støy

Hvit støy er et tilfeldig signal som har lik amplitude ved alle frekvenser. Det vil si at dersom det gjøres en spektrumsanalyse av hvit støy, skal signalet vise en rett linje i frekvensspekteret [1]. Hvit støy kjennetegnes ved at det ikke er noen statistisk sammenheng mellom signalet ved ulike tidspunkter.

3.2 Støygenerator

Det finnes flere måter å designe en digital støygenerator på. Et eksempel er ved å lage et *lineært tilbakekoblet skiftregister* (LFSR[1]) [2]. Et LFSR funker ved det å fylle et register med en tilfeldig verdi (seed), for å så skifte alle bitsene mot en side. Den registerverdien som blir tom vil da bli erstattet av en verdi som er basert på kombinatorisk logikk. For at skiftregisteret skal gå igjennom alle mulige verdier, så må den kombinatoriske logikken velges med omhu. Det finnes dokumenter som har funnet optimal kombinatorisk logikk for å få maksimal periodisitet [3]. Lengre periodisitet gir bedre approksimasjon av hvit støy.

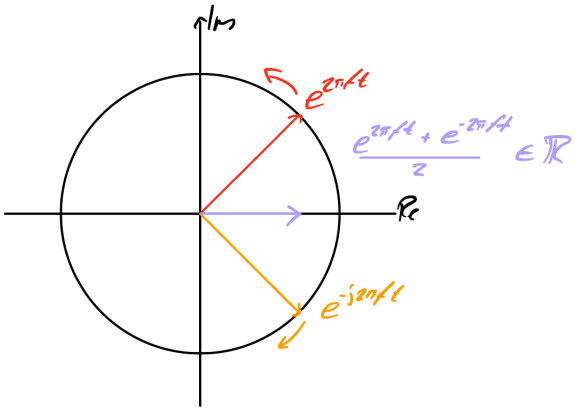
[1] Linear-Feedback Shift Register



Figur 2: Generell topologi for et LFSR

3.3 Manipulasjon av filter i frekvensdomenet

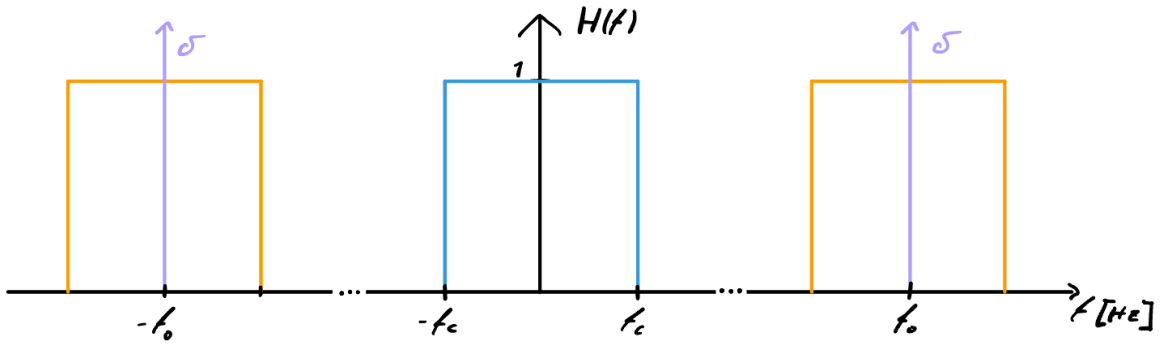
For å finne systemfunksjonen til et båndpassfilter, er det mulig å kombinere systemfunksjoner som allerede er kjent. For å lage et reelt filter, altså et filter som tar inn og sender ut reelle signal, så må systemfunksjonen være symmetrisk om frekvensen 0 Hz. Beviset kan utledes i fra Eulers identitet, dersom man har to vektorer som snurrer i motsatt retning i forhold til hverandre (en med positiv frekvens og en med negativ frekvens). Og summer disse vektorene sammen, vil man alltid ende opp med et reellt tall. Figur 3 viser dette i et tilfelle der summen delt på to, er lik cos, som er den reelle delen av eulers identitet. For at dette skal være sant så må hastigheten (frekvensen) være identisk på begge vektorene.



Figur 3: To vektorer som roterer med lik hastighet men motsatt retning.

I Figur 4 så er det tegnet opp et ideelt lavpass-filter med knekkfrekvens på frekvens f_c, det er også tegnet opp to dirac-pulser på frekvensene f_0 og $-f_0$. Dersom man gjør en konvolusjon mellom disse to funksjonene vil man ende opp med et båndpass filter som er sentrert rundt f_0 og med bredde på $2f_c$.

Denne informasjonen, sammen med formelen for utgangssignalet til et system med impulsrespons $h(t)$;



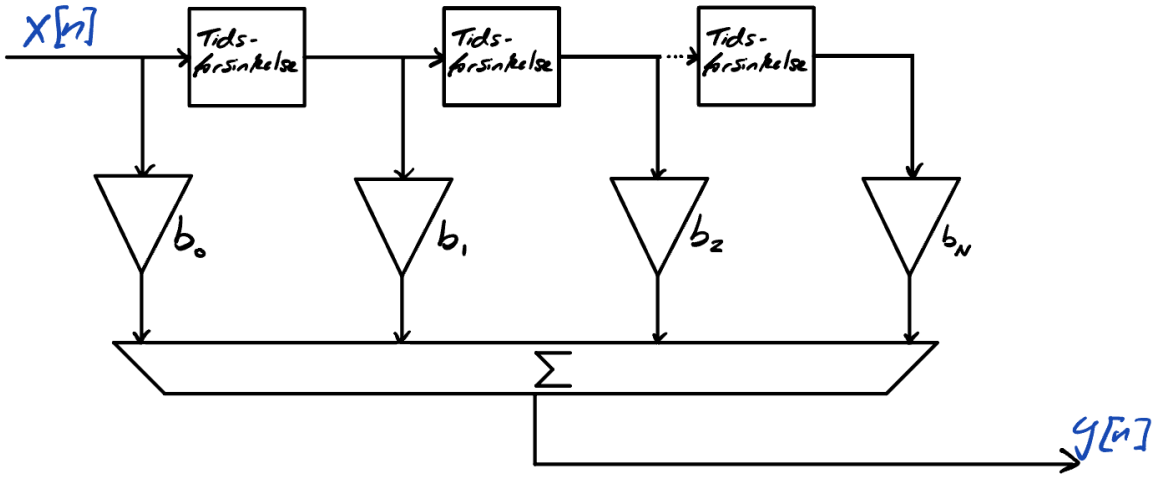
Figur 4: Et lavpass filter (blå), et sinussignal (lilla), og et båndpass filter (oransje) vist i frekvensdomenet

$$Y(f) = H(f) \cdot X(f) \Longleftrightarrow y(t) = h(t) * x(t) \tag{1}$$

Så kan man komme frem til at for at utgangssignalet til et system, vil være gitt ved $y(t)$ dersom det blir gjort en konvolusjon i tid mellom signalet $x(t)$ og impulsresponsen $h(t)$.

3.4 FIR- filter

Et FIR-filter, eller *Finite Impulse Response*- (Avgrenset impulsrespons) filter er et form for digitalt filter som går ut på det å gjøre en diskret konvolusjon. Navnet *Avgrenset impulsrespons* forteller det at for at dette skal kunne realiseres, så må impuls-responsen til filteret begrenses. Figur 5 viser oppbygningen til et FIR-filter, signalet $x(t)$ vil bli gjort diskret, og en og en bit sendes inn igjennom filteret. Der $x[n]$ blir så multiplisert med en koeffisient hvor til slutt alle resultatene summes opp og sendes så ut som et diskret utgangssignal $y[n]$.



Figur 5: Blokkdiagram av et generelt FIR-filter, der b_n er koeffisienter, $x[n]$ og $y[n]$ er diskrete inn- og utgangssignal

3.4.1 Koeffisienter

For at amplituden i DC (f_0 når filteret er flyttet til å bli et båndpassfilter) på filteret skal være lik 1 (0dB), så må koeffisientene normaliseres. Normalisering av koeffisientene kan utføres ved å ha et lavpass-filter, og så summere alle koeffisientene. Summen forteller hvordan amplituden blir påvirket dersom det sendes DC-signal igjennom filteret. På grunn av at filteret skal ha en amplitude på 1 i passbandet, så må hver koeffisient deles på summen av alle koeffisientene.

3.4.2 Kvalitetsfaktor for båndpass filter

For å finne kvaliteten på et båndpass filter, blir det ofte brukt noe som kalles for en Q-faktor (Quality factor). Q-faktoren er gitt ved å dele senterfrekvensen til filteret på båndbredden til filteret. Der båndbredden er definert som differansen mellom de to punktene hvor amplituden synker med 3dB.

$$Q = \frac{f_0}{\Delta f} \tag{2}$$

Der f_0 er senterfrekvensen, og Δf er differansen mellom de to knekkfrekvensene.

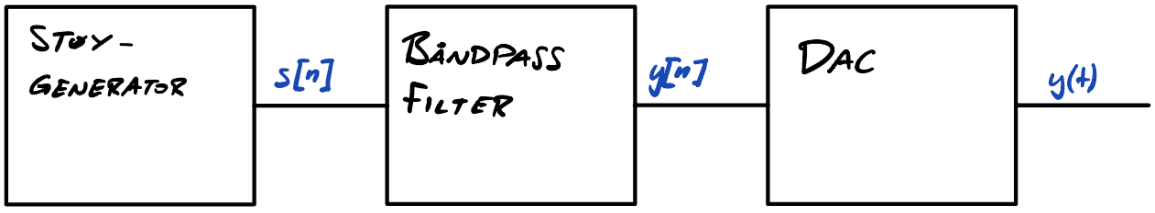
En høy Q-verdi forteller at filteret er veldig smalt, mens en lav Q-faktor forteller at filteret er veldig bredt. Jo smalere filteret er, jo mere konsentrerte frekvenskomponenter vil være mulig å filtrere ut.

4 Metode

I denne seksjonen blir systemet som beskrevet i problembeskrivelsen realisert. Det ble bestemt at designet skal lages digitalt på en Intel Cyclone V FPGA, på et DE1-SoC utviklingskort [4].

4.1 Begrensninger til maskinvare

For å kunne realisere systemet på en FPGA så trengs moduler slik som vist i Figur 6.



Figur 6: Blokkskjema av hvordan systemet kan realiseres på en FPGA.

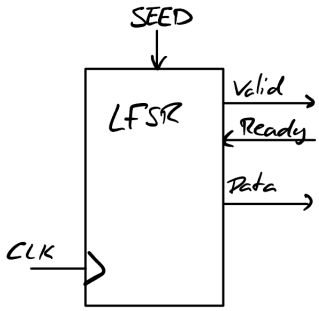
Utviklingskortet DE1-SoC har en fysisk DAC chip, i følge databladet til DACen [5] støttes samplingsrater på 8, 32, 48 kHz. Ut i fra kravene til systemet ble det bestemt å bruke 48 kHz som samplingsrate (f_s). En samplingsrate på 48 kHz vil gi tilnærmet hvit støy i hele det hørbare området.

Cyclone V FPGAen har en klokkegenerator som genererer en klokke på 50 MHz [4]. Den 50 MHz klokken vil kunne brukes til å gjøre beregninger i FIR-filteret. Dersom DACen sampler innkommende signal med en klokkehastighet på 48 kHz, så er signalet ut av båndpassfilteret nødt til å være klart hver $\frac{50\,\text{MHz}}{48\,\text{kHz}} \approx 1042$ klokkesykel.

4.2 Støygenerator

Støy blir generert ved hjelp av å sette opp et LFSR som består av 32 bits. Siste bit i registeret blir hentet ut av LFSR. Dette skjer ved en hastighet på 48 kHz, det vil si at det tar $\frac{2^{32}}{48000\ \text{Hz}\cdot 60\ \text{s}\cdot 60\ \text{m}\cdot 24\ \text{h}} \approx 1$ døgn før støyen vil gjenta seg selv. Denne formen for periodisitet kan neglisjeres. Når et signal ikke er periodisk vil det ikke ha et grunnsignal som kan danne harmoniske signal. Harmoniske signal er ikke ønskelig, da støyen ikke blir hvit lengre dersom de oppstår.

Støygeneratoren ble realisert ved å lage en modul skrevet i VHDL som kan finnes på: [6]. Figur 7 viser en top-level representasjon av hvordan støygeneratoren ble realisert.



Figur 7: Top-level av realisert LFSR

4.3 Båndpassfilter

For realisering av systemet ble det bestemt å lage et digitalt filter av typen FIR-filter. Et FIR-filter er et digitalt filter som er veldig mye brukt, og kan gjøres lett å implementere.

4.3.1 Finne systemfunksjon

For å danne en systemfunksjon til et båndpassfilter så kan metoden introdusert i seksjon 3.3 utnyttes. Man starter først med et lavpassfilter som har en knekkfrekvens f_c. Og så flytter lavpassfilteret opp mot en diracpuls som er plassert på senterfrekvensen til båndpassfilteret.

Ved å sette opp uttrykk for systemfunksjoenen til et lavpass filter og en dirac puls. og ved å slå opp i tabeller som viser fourier-transform par [7], så kan man komme frem til følgende formler;

$$H_{\text{Lavpass}}(f) = \Pi\left(\frac{f}{2f_c}\right) \Longleftrightarrow^{\mathcal{F}} h_{\text{Lavpass}}(t) = \frac{1}{2\pi f_c t}\sin\left(2\pi f_c t\right) \tag{3}$$

$$H_{\cos}(f) = \delta(f - f_0) + \delta(f + f_0) \Longleftrightarrow^{\mathcal{F}} h_{\cos}(t) = 2\cos\left(2\pi f_0 t\right) \tag{4}$$

$$H_{\text{Båndpass}}(f) = H_{\text{Lavpass}}(f) * H_{\cos}(f) \Longleftrightarrow^{\mathcal{F}} h_{\text{Båndpass}}(t) = \frac{1}{2\pi f_c t}\sin\left(2\pi f_c t\right) \cdot 2\cos\left(2\pi f_0 t\right) \tag{5}$$

Der Π symboliserer rektangel-funksjonen [8], f_c er knekkfrekvensen til lavpassfilteret og f_0 er senterfrekvensen.

Formel (1) viser at dersom man gjør en konvolusjon mellom inngangssignalet $s(t)$ med impulsresponsen $h(t)$ så vil man få et utgangssignal $y(t)$. $y(t)$ vil da være et båndbegrenset signal med senterfrekvens f_0 og med knekkfrekvens på $f_0 \pm f_c$.

Et menneske klarer å skille toner der differansen mellom frekvensen ligger på omtrent 0.5% av hverandre [9]. Det vil si, for en frekvens på en senterfrekvens på $f_0 = 2880$Hz, så vil toner som ligger på omtrent $f_0 \pm 14.4$Hz være hørbart. Derfor blir f_c satt til 10Hz i dette systemet.

Den endelige impulsresponsen ender da opp med å være gitt ved følgende formel:

$$h(t) = \frac{1}{20\pi t}\sin\left(20\pi t\right) \cdot 2\cos\left(20\pi \cdot 2880t\right)$$

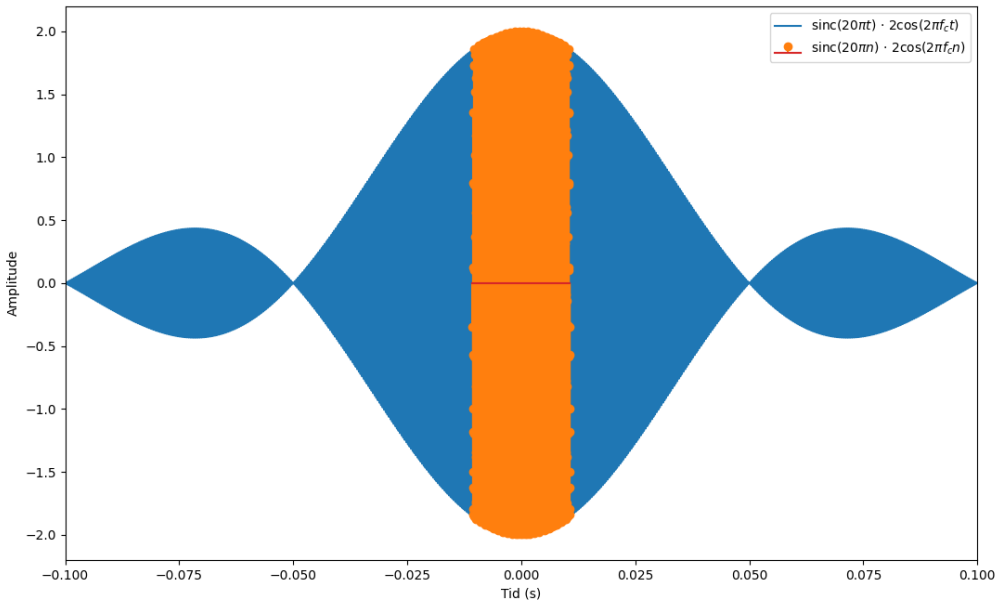
4.3.2 Realisere filter

En måte å gjøre en konvolusjon i tid, er ved å benytte et FIR-filter. Et FIR-filter krever det at impulsresponsen er avgrenset innenfor et gitt område, og at impulsresponsen blir gjort diskret i stedet for kontinuerlig.

Å gjøre signalet diskret kan gjøres ved hjelp av å utnytte samplingsraten f_s som skal brukes når signalet blir sent ut til DAC-en, som så gjør om det diskrete signalet $y[n]$ til et signal som er hørbart $y(t)$.

For å realisere filteret, må det bestemmes orden (N) på filteret. I et FIR-filter så er antall koeffisienter gitt ved $N + 1$ [10]. Antall koeffisienter i et FIR-filter, bestemmer sammen med sampleraten hvor lang tidsforsinkelse filteret vil forårsake. For systemet som skal realiseres, så har ikke tidsforskyvelse noe å si på grunn av at inngangen kun skal være støy. Dersom filteret realiseres med å gjøre en multiplikasjon per klokkesykel så bestemmes maksimalt antall koeffisienter ut i fra begrensningene til maskinvaren. Signalet ut i fra FIR-filteret er nødt til å være klart hver 1042 klokkesykel. For å tilfredsstille tidskravene, så velges det å bruke $2^{10} = 1024$ koeffisienter. 1024 koeffisienter tilsvarer et 1023-ordens filter.

For å finne koeffisientene blir samplingsraten benyttet; først bestemmes det tidssteg på $\frac{1}{f_s} = \frac{1}{48\,\text{kHz}} = 20\,\mu\text{s}$. Så blir dette tidssteget brukt til å definere et "vindu" som bestemmer hvor mye av impulsresponsen $h(t)$ som skal bli brukt: $n \in [20\,\mu\text{s} \cdot \frac{N}{2}\ ,\ 20\,\mu\text{s} \cdot \frac{N}{2}]$. Figur 8 viser kontinuerlig $h(t)$ og den diskrete versjonen $h(n)$.

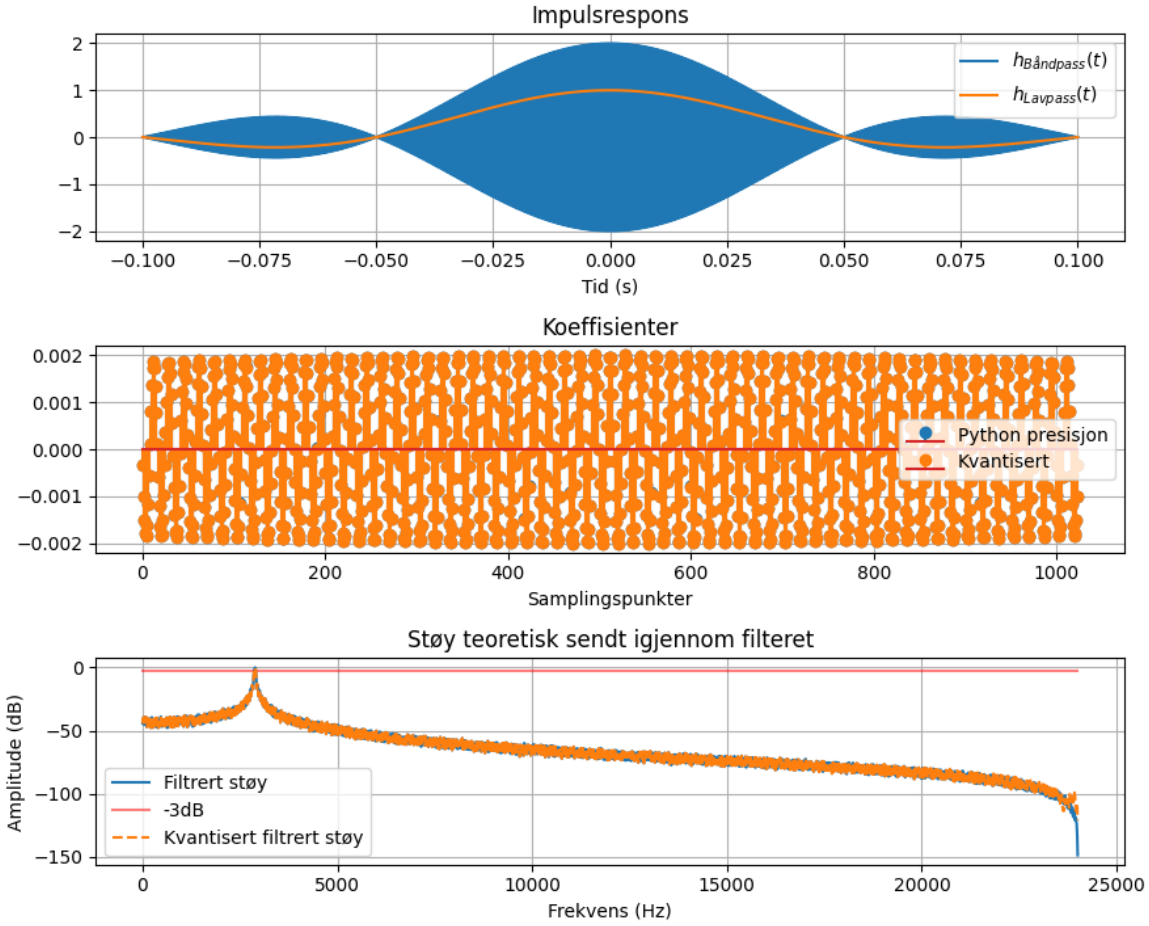


Figur 8: Impulsrespons $h(t)$ av båndpass filteret. Det blå viser kontinuerlig impulsrespons, mens det oransje viser 1024 samplingspunkter av impulsresponsen

4.3.3 Håndtering av koeffisienter

For å gjøre om impulsresponsen til koeffisienter som kan brukes, er det nødvendig å normalisere de, normalisering er for å få en forsterkning på 1 (0 dB) i passbandet. Normalisering av koeffisienter gjøres ved å sende et signal som ligger i passbandet, finne summen av alle koeffisientene som brukes, og dele alle koeffisientene på summen. En enkel måte å sørge for at det sendes inn et signal som ligger i passbandet, er ved å først finne koeffisientene for $h_{Lavpass}(n)$. Så og summere alle koeffisienter sammen, å summere alle koeffisienter tilsvarer det å sende DC-signal igjennom filteret. Koeffisientene for $h_{Lavpass}(n)$ deles så på summen av koeffisientene, til slutt flyttes passbandet opp til senterfrekvensen til båndpassfilteret ved å gange sammen $h_{Lavpass}$ og h_{cos}.

Prosessen ved å normalisere koeffisientene ble gjort i et python skript med navn *filter.py* [6]. Når koeffisientene er definert, så må de *kvantiseres*. I en FPGA kan man bestemme antall bits som skal representere en verdi på. Dette er gjort ved å bestemme hvor mange bits som skal representere heltallsverdien og hvor mange bits som skal representere tallet bak kommaet. I tilfellet, dersom alle positive koeffisientene summeres sammen så når de 0.642. Dette betyr at det ikke trengs noen bits til å representere heltallet, alle bits kan representere tallet bak

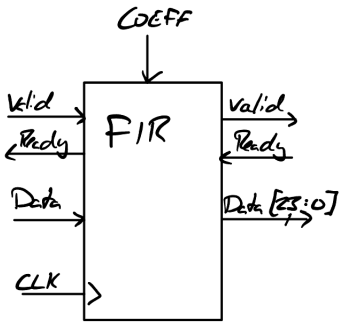


Figur 9: Beregnede verdier for filteret før det implmenteres på FPGA.

kommaet. I dette systemet gir det mening å sette antall bits som skal representere verdiene ut i fra filteret, likt som antall bits som DAC'en forventer. DAC'en på utviklingskortet forventer 24-bits oppløsning på inngangen.

FIR-filteret som er blitt designet har teoretiske verdier som er vist i Figur 9. Knekkfrekvensene til båndpass filteret ender teoretisk opp med å være på $f = 2873$Hz og $f = 2887$Hz. Q-faktor for dette filteret er gitt ved (2) og vil ende opp med å være $Q = \frac{2880}{2887-2873} \approx 206$, en høy Q-verdi forteller det at filteret er smalt. Det er det man ønsker av et båndpass filter.

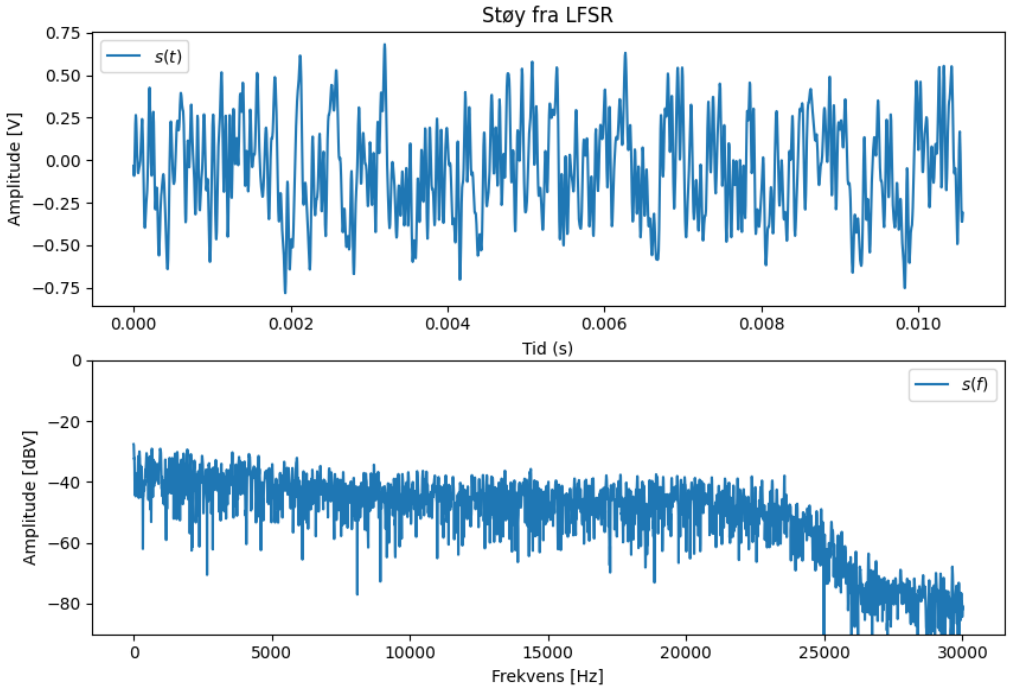
Top-level representasjon av realisert FIR-filter er vist i Figur 10.



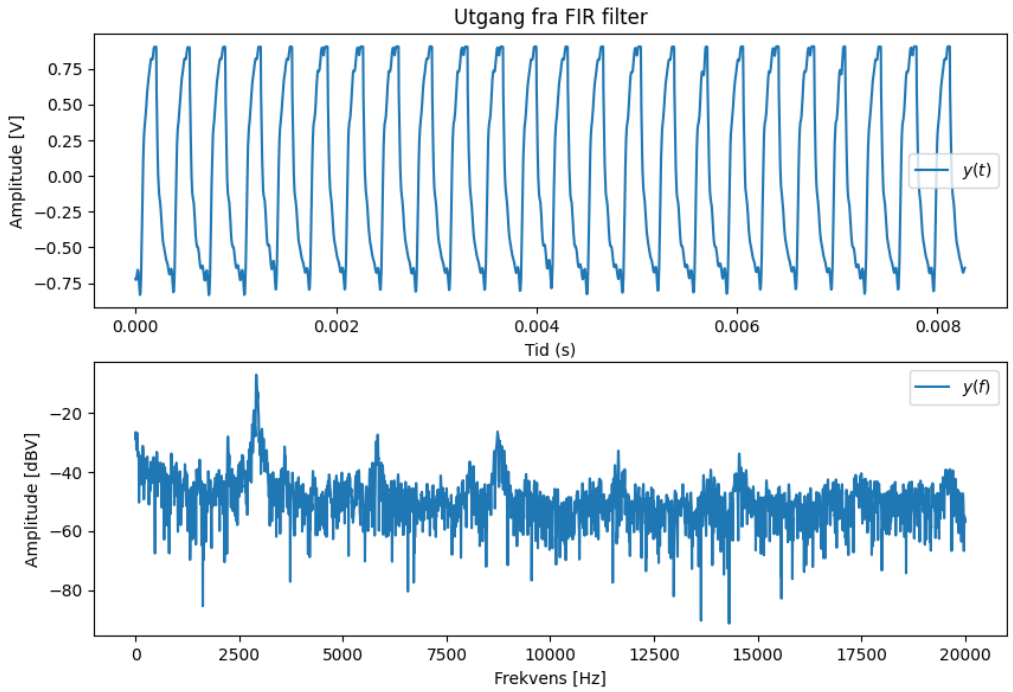
Figur 10: Top-level av realisert FIR-filter

5 Resultat

Figur 11 og 12 viser støysignalet s ut i fra utgangen på LFSR, og hvordan signalet y kommer igjennom FIR-filteret. Figur 13 i appendix, viser oppkobling av for å utgjøre målingene.



Figur 11: Støysignal ut fra LFSR



Figur 12: Signal ut i fra FIR-filter

6 Diskusjon

6.1 Støygenerator

Støygeneratoren endte opp med å funke som forventet, ved å høre på signalet så er det kun støy som er hørbart, det er ikke mulig å tyde noen toner som skiller seg ut. I Figur 11 så er det mulig å se at støyen forsvinner ved rundt 24 kHz. Grunnen til at støyen avtar kommer i fra det som ble nevnt i seksjon 4.2, den høyeste frekvenskomponenten i støyet vil være gitt ved halvparten av klokkefrekvensen inn til støygeneratoren.

6.2 Båndpass filter

Ut i fra de teoretiske beregningene og simuleringen gjort i python så skal filteret ende opp med å ha en Q-faktor på rundt 206. Ved realisering av filteret endte senterfrekvensen f_0 på 2910 Hz med knekkfrekvenser på $f = 2903$ Hz og $f = 2923$ Hz. Ut i fra målingene endes det da opp med en Q-faktor på $Q = \frac{2910}{2923-2903} = 145.5$.

Filteret endte opp med en senterfrekvens som avviker med $2910\,\text{Hz} - 2880\,\text{Hz} = 30\,\text{Hz}$ i fra det som er blitt designet. En årsak til avviket kan komme fra at koeffisientene ikke er spredd noe langt utover impulsresponsen (se Figur 8). Det finnes flere måter å få koeffisientene til å

utnytte mere av impulsresponsen;

1. Øke antall koeffisienter.
2. Senke samplingsraten.
3. Optimalisere systemet slik at utregninger skjer parallellt i stedet for serielt, dette gjør at antall koeffisienter kan økes drastisk.

Koeffisientene er veldig små tall, i systemet som er blitt designet så er det bare 6 av 24 bits som er gjeldende verdier i koeffisientene. Unøyaktigheter i koeffisienter kan lede til at systemet ikke lengre er lineært, som videre fører til at harmoniske kan oppstå. I Figur 12 kan man se at det er kommet harmoniske signal, slik at sinusbølgen på utgangen inneholder flere frekvenskomponenter enn ønskelig.

Ved å høre på signalet som kommer ut, så kan man høre en tydelig tone, blandet sammen med noen litt lavere toner.

7 Konklusjon

Systemet endte opp med å kunne filtrere ut et tone fra hvit støy, slik at tonen var hørbar. Båndpassfilteret hadde et avvik på senterfrekvensen f_0 med 30 Hz. Båndbredden til filteret er på 20 Hz. Q-verdien til filteret er 145.5.

8 Takk

Tusen takk til Christian Cartfjord fra Zolve AS for gode diskusjoner, gode forklaringer og for å ha sittet timesvis for å være tilgjengelig for å hjelpe.

Referanser

[1] Wikipedia, *Hvit støy*, https://no.wikipedia.org/wiki/Hvit_stÃÿy.

[2] Wikipedia, *Linear-feedback shift register*, https://en.wikipedia.org/wiki/Linear-feedback_shift_register.

[3] Xilinx, *Efficient Shift Registers*, https://docs.xilinx.com/v/u/en-US/xapp052.

[4] Terasic, *DE1-SoC Board*, https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=836.

[5] Wolfson, *WM8731 datasheet*, http://cdn.sparkfun.com/datasheets/Dev/Arduino/Shields/WolfsonWM8731.pdf

[6] Sindre Mandelid Kvam, *FIR filter*, https://gitlab.com/sindrekvam/fir-filter.

[7] The University of Texas, *Fourier transform tables*, https://www.yumpu.com/en/document/view/43107106/fourier-transform-tables.

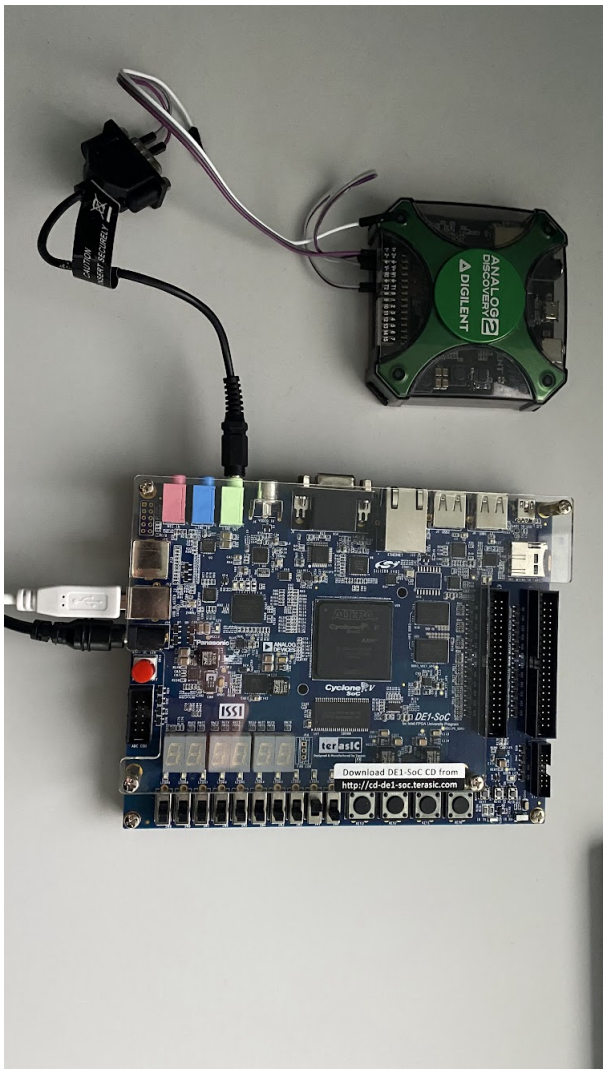
[8] Wikipedia, *Rectangular function*, https://en.wikipedia.org/wiki/Rectangular_function.

[9] *Critical Bandwidths and Just-Noticeable Differences* https://www.phys.uconn.edu/~gibson/Notes/Section7_2/Sec7_2.htm.

[10] Wikipedia, *Finite Impule Response*, https://en.wikipedia.org/wiki/Finite_impulse_response.

A Appendix

A.1 Oppkobling



Figur 13: Oppkobling av måleinstrument sammen med FPGA utviklingskort.