# Ekkoloddsystem realisert med System-on-Chip på FPGA

**Mats Randgaard**


**Masteroppgave ved Fysisk institutt**

UNIVERSITETET I OSLO

Januar 2010

# Forord

Noen lærerike år ved Universitetet i Oslo går nå mot slutten. I løpet av disse årene har jeg hatt tid og mulighet til å sette meg grundig inn i elektronikken, og gå dypere inn i emner som interesserer meg spesielt. Fagvalgene jeg har gjort i masterutdanningen har vært basert på hva jeg har hatt lyst til å lære mer om, og derfor er det interessant å se at jeg har fått bruk for alle fagene i arbeidet med masteroppgaven også. Jeg ønsker å takke førsteamanuensis Torfinn Lindem for at jeg fikk mulighet til å jobbe med denne oppgaven som har gitt meg en bred praktisk bakgrunn i elektronikk, og særlig i FPGA-teknologi som jeg ønsket å fordype meg i.

Siden jeg har vært den eneste på Fysisk institutt som har jobbet med en masteroppgave med FPGA-teknologi, har jeg stort sett måttet finne ut av ting på egenhånd. Det har vært krevende og tatt mye tid. Derfor har det vært inspirerende at mine erfaringer og kildekoder for bruk av IP-stakken lightweight IP (lwIP) allerede har blitt tatt i bruk. Trygve Ødegaard ved Elektronikklaboratoriet ved Fysisk institutt (ELAB) har laget en laboppgave for professor Bernhard Skaalis fag, FYS 4220, basert på mine kildekoder. De kommer også til å bli brukt i en fotonteller på Andøya som Halvor Strøm ved ELAB holder på med å utvikle.

Takk til Johan Tresvig, Markus Grønstad, Henning Vangli og Knut Skumsvoll, som jeg har delt kontor med i løpet av disse årene, for en fin tid ved Universitetet i Oslo.

Jeg vil også takke min far, Frode Randgaard, for at han har lest korrektur på oppgaven selv om han ikke har grunnlag for å forstå noe særlig av elektronikken!

Aller mest vil jeg få takke min kjære Åshild og våre kjære små, Markus og Miriam, som har latt meg ta denne masteroppgaven, selv om det har tatt mye tid fra kvelder og helger vi heller skulle vært sammen. Du har vært veldig hjelpsom og tålmodig, Åshild!

Oslo, 29. januar 2010

Mats Randgaard

# Innholdsfortegnelse


**1. INNLEDNING ..... 1**

1.1 BAKGRUNN ..... 2

1.2 PROBLEMSTILLINGER ..... 2

1.3 SYSTEMOVERSIKT ..... 5

1.4 HYDROAKUSTIKK ..... 8

**2. ANALOG ELEKTRONIKK ..... 13**

2.1 BINDELEDD MELLOM KASSETTSPILLER OG FPGA ..... 14

2.2 KOMPONENTVALG ..... 14

2.3 KRETSBESKRIVELSE ..... 15

**3. DIGITAL ELEKTRONIKK ..... 39**

3.1 PROGRAMMERBAR LOGIKK ..... 42

3.2 HVA ER EN FPGA? ..... 46

3.3 HVORFOR BRUKE EN FPGA? ..... 53

3.4 FPGA FOR EKKOLODDPROSJEKTET ..... 56

3.5 PROGRAMVARE FRA XILINX ..... 60

3.6 MASKINVAREBESKRIVENDE SPRÅK ..... 69

3.7 SYSTEMOVERSIKT ..... 72

3.8 ENHETER ..... 73

3.9 MODULER I XILINX PLATFORM STUDIO ..... 101

**4. PROGRAMVARE ..... 119**

4.1 TCP/IP ..... 120

4.2 PROGRAMKODE C ..... 138

4.3 DATAINNSAMLINGSPROGRAM FOR PC ..... 142

**5. RESULTATER ..... 147**

5.1 FREKVENSGENERATOR ..... 147

5.2 KASSETTSPILLER ..... 152

**6. KONKLUSJON ..... 155**

**KILDELISTE ..... 157**

**PROGRAMLISTE ..... 166**

**VEDLEGG ..... 171**

**A. KRETSSKJEMA FOR EKSISTERENDE SYSTEM ..... 172**

**B. BEREGNING AV OVERSAMPLING FOR AD-OMFORMEREN ..... 176**

**C. KRETSSKJEMA ..... 177**

**D. KRETSKORTUTLEGG ..... 184**

**E. PARAMETRE FOR BÅNDPASSFILTRENE ..... 189**

**F. DOXYGEN – DOKUMENTASJON AV KILDEKODE ..... 191**

**G. LIGHT WEIGHT IP (LWIP) FOR XILINX FPGA-ER ..... 193**

**H. TESTUTSTYR ..... 205**

**I. VEDLEGG PÅ CD OG INTERNETT ..... 206**

# Språk

Fagområdet elektronikk er dominert av engelsk språkbruk, men vi har så langt det er mulig prøvd å bruke norske ord der det finnes. Noen ord lar seg ikke oversette til norsk uten at meningen blir upresis, og andre ord er så innarbeidet på engelsk at de vil være vanskelige å forstå i en norsk oversettelse. Ord som ofte brukes på engelsk, men som her er skrevet på norsk, etterfølges av en parentes med det engelske ordet. Oversettelsene er stort sett hentet fra teknisk engelsk-norsk ordbok på ordnett.no. Forkortelser er forklart første gang de brukes.

# 1. Innledning

1.1 BAKGRUNN ..... 2

1.2 PROBLEMSTILLINGER ..... 2

*1.2.1 Ekkolodd for Naturhistorisk museum* ..... *3*

*1.2.2 Digitalisering av kassetter* ..... *3*

*1.2.3 Uthenting av bærebølgens omhyllingskurve* ..... *4*

1.3 SYSTEMOVERSIKT ..... 5

*1.3.1 HADAS 2010* ..... *6*

1.4 HYDROAKUSTIKK ..... 8

## 1.1 Bakgrunn

Sonargruppen ved Fysisk institutt ved Universitetet i Oslo har gjennom mange år jobbet med problemstillinger innen hydroakustikk. Blant annet utvikles analyseprogrammet SONAR 5 av førsteamanuensis Helge Balk [Balk, Sonar 5], og det har blitt utviklet flere ekkoloddsystemer for forskningsanvendelser. Det som skiller disse systemene fra ekkolodd og sonarer som brukes av fiskere, er at brukeren har full kontroll over signalbehandlingen slik at målingene blir mer korrekte. De mottatte dataene kan også lagres for videre analyse.

Et av disse ekkoloddsystemene er SIMRAD EY-M [SIMRAD, 1979] som ble utviklet av førsteamanuensis Torfinn Lindem på midten av 70-tallet for SIMRAD. Det fantes flere ekkolodd for vitenskapelige anvendelser, men dette var også portabelt og derfor godt egnet til ferskvannsundersøkelser. Ekkoloddet var analogt og de mottatte signalene ble tatt opp på kassett og plottet på papir.

Da det etter hvert ble behov for å digitalisere opptakene for videre analyse, utviklet firmaet Lindem Data Acquisition, som var eid av førsteamanuensis Torfinn Lindem, digitaliseringssystemet HADAS (Hydro Acoustic Data Acquisition System) [Lindem data Acquisition, 1992]. Dette besto av en boks for justering av det analoge signalet fra kassettspilleren som var koblet til et PCI-kort for AD-omforming av signalet. I tillegg fulgte det med programvare for å lagre og presentre dataene.

## 1.2 Problemstillinger

Oppgavens tittel er “Ekkoloddsystem realisert med System-on-Chip på FPGA”. Begrepet System-on-Chip (SoC) brukes om systemer der flere funksjoner er samlet på samme brikke. Noen bruker begrepet om systemer som inneholder både digitale og analoge kretser, men begrepet kan også brukes om digitale kretser som består av flere elementer. En mikrokontroller er egentlig ”et system på en brikke”, men begrepet

brukes om større systemer med kraftigere prosessorer og mer minne. I forbindelse med FPGA-er (se kapittel 3.2) brukes begrepet når en prosessor er implementert på FPGA-en sammen med spesiallaget digital logikk, nettverksgrensesnitt, tellere og lignende [Wikipedia, System-on-a-chip].

### 1.2.1 Ekkolodd for Naturhistorisk museum

SIMRAD EY-M og HADAS ble brukt av mange forskningsinstitusjoner over hele landet og internasjonalt. Laboratorium for Ferskvannsøkologi og Innlandsfiske (LFI) ved Naturhistorisk museum ved Universitetet i Oslo [Naturhistorisk museum, LFI] er en av institusjonene som fortsatt bruker systemet til sine forskningsprosjekter.

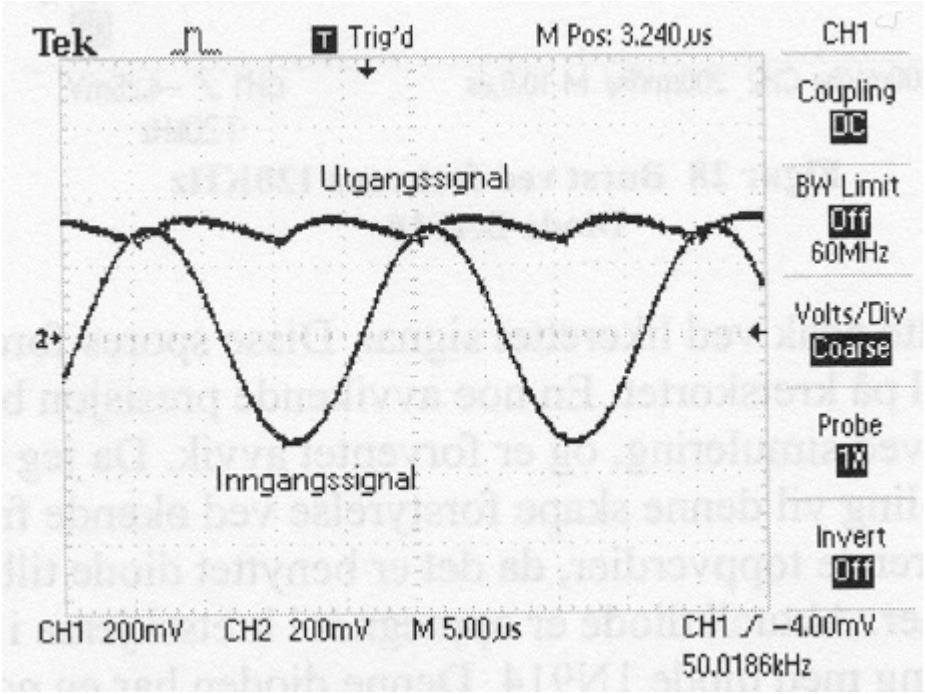
Systemet begynner nå å bli gammelt og de ønsker at dagens analoge opptak med påfølgende digitalisering skal erstattes med en løsning der sonardata sendes direkte fra ekkoloddet til en PC via Ethernet for videre behandling og presentasjon. Hvis det lar seg gjøre, er det ønskelig at elektronikken er pakket i en kompakt, bærbar boks som kan kobles til en bærbar PC.

### 1.2.2 Digitalisering av kassetter

Flere av forskningsinstitusjonene har behov for å se på de gamle opptakene på nytt for å se endringene som har skjedd over tid. Dessverre er det mange kassetter med data som ikke er konvertert til et format som kan analyseres i dag. HADAS-systemet er utdatert og kan ikke brukes på moderne PC-er. Det har derfor blitt et behov for å digitalisere gamle kassetter med historiske data. Dette er opptak av rådata fra en 70 kHz sonar som er mikset ned til 10 kHz og spilt inn på analoge kassetter. Signalene svinger om 0 V og amplitudeutslaget er maksimalt ±1 V. Ekkoloddsignalet ligger på R-utgangen av kassettspilleren. På L-utgangen ligger et triggersignal som markerer starten på hvert ping.

### 1.2.3 Uthenting av bærebølgens omhyllingskurve

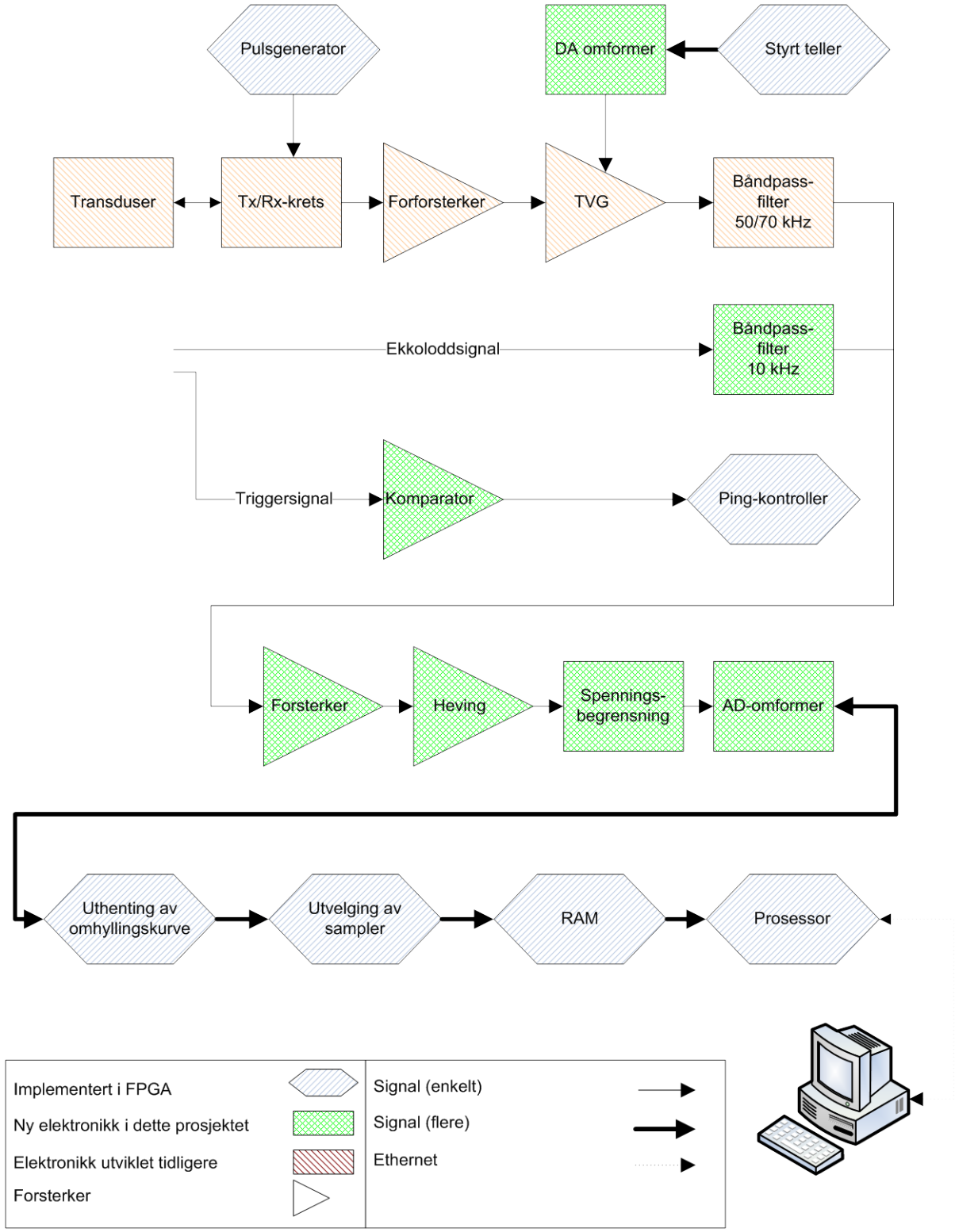
Det er omhyllingskurven av signalet vi er interessert i å ta vare på i det mottatte signalet (Innføring i hydroakustikk i kapittel 1.4). Både HADAS-systemet, Merdøye og andre ekkoloddsystemer som har blitt utviklet ved sonargruppa (for eksempel [Skumsvoll, 2008]) har brukt analog signalbehandling for å hente ut omhyllingskruven av signalet, før denne er samplet og konvertert til digitale verdier. Som vist i [Skumsvoll, 2008, s. 29 og 37-40] er den analoge signalbehandlingen noe unøyaktig og forvrenger utgangssignalet mer enn ønskelig.




**Figur 1-1:** Inngangs- og utgangssignaler fra tidligere utviklet kretskort for uthenting av omhyllingskurve [Skumsvoll, 2008, figur 30, s. 40].

Nå som både AD-omformere og digitale signalprosesseringsenheter har blitt raskere, var det derfor ønskelig å se om det var mulig å sample bærebølgen direkte for så å hente ut omhyllingskurven ved hjelp av digital signalbehandling.

## 1.3 Systemoversikt




**Figur 1-2:** Forenklet skisse av systemet (Tegnet i Microsoft Visio)

Systemet er designet for å løse problemstillingene i kapittel 1.2: enhetene skal kunne brukes til å digitalisere gamle kassettopptak og å inngå i et ekkolodd for Naturhistorisk museum. Skissen i figur 1-2 viser hvordan dette kan implementeres.

Anvendelsen av systemet som ekkolodd er ikke ferdig utviklet i dette prosjektet. Som figur 1-2 viser er tanken å benytte elektronikk som er utviklet i Sonargruppen til tidligere prosjekter. Kretstegningene som er utviklet av Stein Lyng Nielsen ved ELAB er vist i vedlegg A. Dette er et ekkolodd med sendefrekvens på 50 kHz eller 70 kHz. Mange av kretsene kan brukes som de er, mens andre må tilpasses for det nye systemet.

Systemet er klart for å digitalisere kassetter, men elektronikken og innholdet i FPGA-en er designet med tanke på at ekkoloddet skal ha to anvendelser, og kan derfor implementeres direkte i et ekkolodd.

Den videre beskrivelsen av systemet er delt opp i følgende hoveddeler:

- analog elektronikk (kapittel 2) som tar for seg oppbygging av kretskortet
- digital elektronikk (kapittel 3) som omhandler digital logikk på FPGA-en
- programvare (kapittel 4) som beskriver programvare for både MicroBlaze-prosessoren på FPGA-en (C) og datainnsamlingsprogrammet for datamaskiner (Python).

### 1.3.1 HADAS 2010

Det første systemet som ble utviklet ved sonargruppa for å digitalisere ekkoloddsignal het HADAS (Hydro Acoustic Data Acquisition System) [Lindem data Acquisition, 1992]. Senere ble programmet HADAS 2000 laget [Tor Anstein Olsen, 2002], så i dette prosjektet har vi brukt betegnelsen HADAS 2010 på det nye systemet. Systemet består av flere deler, så i kildekode og andre dokumenter er følgende betegnelser brukt for å skille de ulike delene fra hverandre:

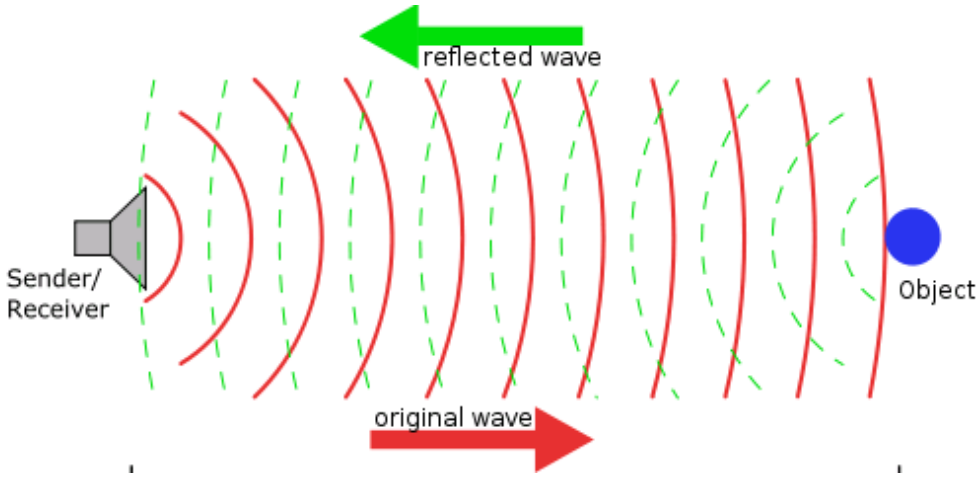
| | |
|---|---|
| HADAS TR | Tape Recorder (Modus for innspilling av kassetter) |
| HADAS ES | Echo Sounder (Modus som ekkolodd) |
| HADAS HW | Hardware (Kretskort) |
| HADAS SoC | System-on-Chip (VHDL + EDK i FPGA-en) |
| HADAS FW | Firmware (C-kode i MicroBlaze-prosessoren) |
| HADAS SW | Software (Datainnsamlingsprogrammet) |

**Tabell 1-1:** Navn som brukes på de ulike delene av prosjektet

## 1.4 Hydroakustikk

Hvaler og delfiner sender lyder under vann for å kommunisere og for å vurdere avstander, men det første mennesket som noterte sine observasjoner på området var Leonardo da Vinci som i 1490 hørte skip på lang avstand ved å stikke et rør i vannet og lytte ved den andre enden [MacLennan og Simmonds, 1992, s. 2]. På begynnelsen av 1900-tallet førte Titanic-ulykken og første verdenskrig til at flere forskningsmiljøer utviklet utstyr for å lytte etter isfjell og ubåter under vann [Wikipedia, Sonar].

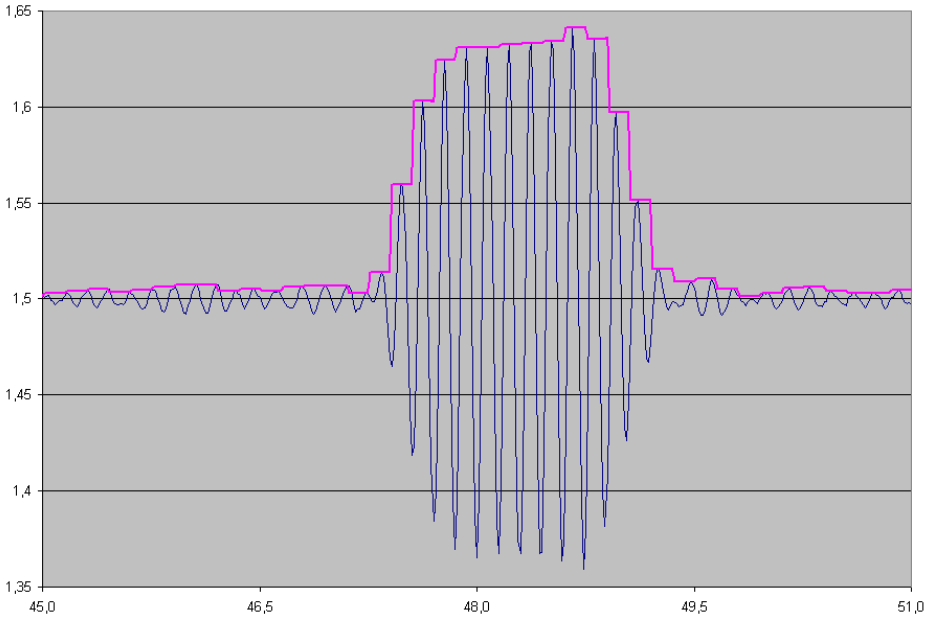
Under andre verdenskrig tok amerikanerne i bruk betegnelsen SONAR (Sound Navigation And Ranging) [Wikipedia, Sonar]. Begrepet sonar brukes i dag gjerne om instrumenter som måler horisontalt i vannet og tegner opp et sirkulært bilde i likhet med radarer, mens ekkolodd normalt måler vertikalt og tegner opp bildet som vertikale streker. I denne oppgaven bruker vi begrepet ekkolodd fordi bildet dannes av vertikale streker, selv om instrumentet brukes både vertikalt og horisontalt.




**Figur 1-3** Prinsippet for et ekkolodd [Wikimedia, Sonar Principle].

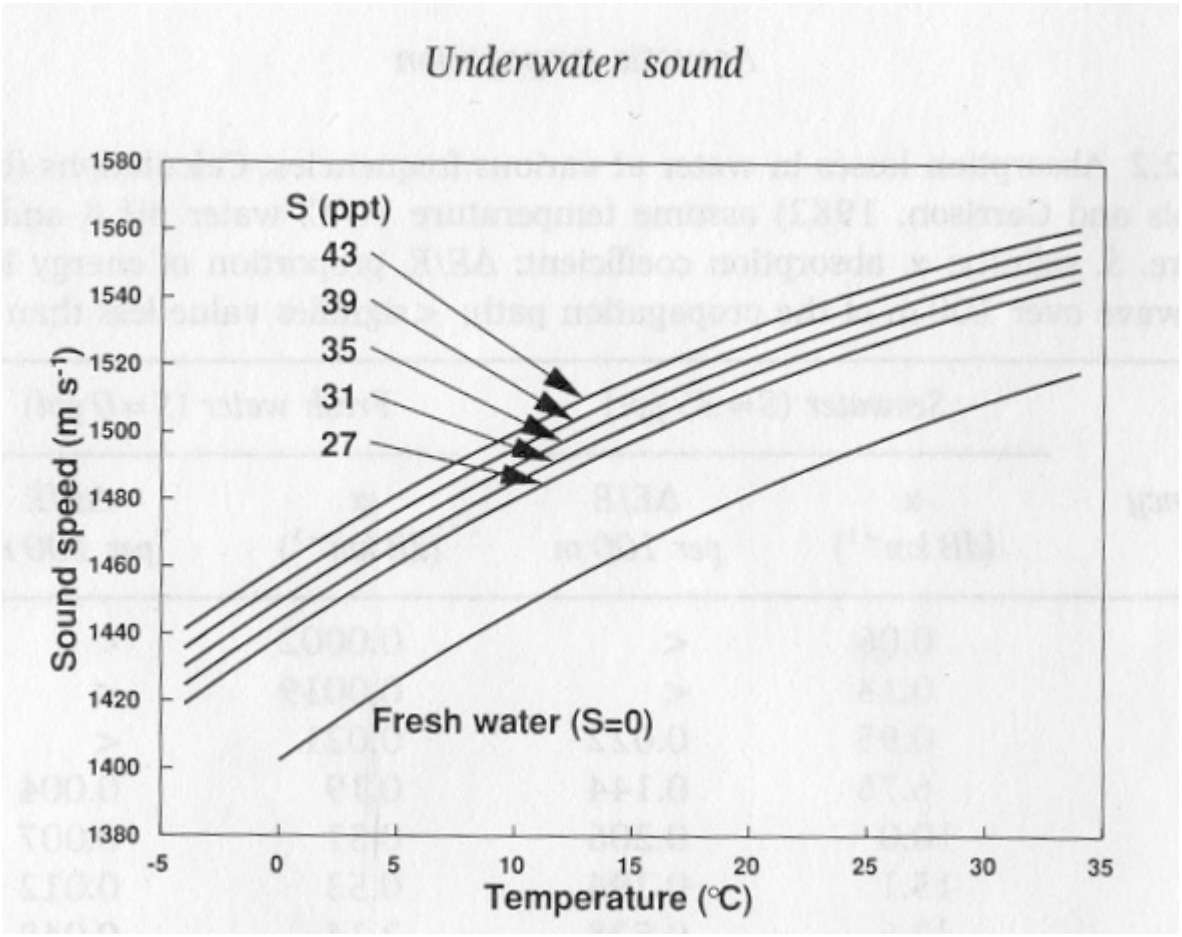
Ekkoloddet består gjerne av en transduser som sender og mottar lydbølger, elektronikk for sending og mottak av lydbølger og en skjerm for å vise de mottatte signalene. Transduseren fungerer først som en slags høyttaler og sender en lydbølge ut i vannet. Utsendingen kalles gjerne et ”ping”. Når lydbølgen treffer et objekt, som for eksempel en fisk eller bunn, reflekteres noen av lydbølgen tilbake til transduseren. Transduseren

brukes da som en slags mikrofon for å motta de reflekterte lydbølgene. Ved å forsterke og behandle de mottatte signalene kan deretter en gjengivelse av objektet vises på skjermen.




**Figur 1-4:** Eksempel på reflektert bølge fra et objekt (tynn blå linje). Den tykke lilla linjen er omhyllingskurven til signalet. (Sonardata behandlet og presentert i Microsoft Excel).

Lydbølgene følger fysikkens lover, og her er en kort oppsummering av de som har betydning for dette prosjektet:




**Figur 1-5:** Lydhastighet i vann. S står for salinitet (oppløst saltinnhold i vann) [MacLennan og Simmonds, 1992, s. 24]

Lydhastigheten i vann er avhengig av dybde, temperatur og saltinnholdet i vannet. Forholdet mellom lydhastighet, temperatur og saltinnhold er vist i figur 1-5. Lydhastigheten avtar med dybde, men på dypere vann er gjerne også temperaturen lavere i tillegg til at saltinnholdet endrer seg [MacLennan og Simmonds, 1992, s.23-27]. Dette prosjektet er rettet mot ekkoloddsundersøkelser i ferskvann, så i de videre utregningene har vi brukt c = 1450 m/s som lydhastighet.

Tiden lydbølgen bruker på å tilbakelegge en strekning, s, i vannet er gitt ved formelen s = c*t, der c er lydhastigheten, og t er tiden det tar. Et ekkolodd som skal kunne måle avtander inntil 100 meter må ta imot signaler så lang tid som lydbølgen bruker på å

tilbakelegge 200 meter, fordi lydbølgen skal gå fram og tilbake. Det tar t = (2*s)/c = 200 m/1450 m/s = 138 ms.

Lavere frekvenser dempes mindre og kan sendes over lengre avstander, mens høyere frekvenser har kortere bølgelengde som gir høyere romlig oppløsning [MacLennan og Simmonds, 1992, s. 41-42]. Bølgelengden i meter, $\lambda$, er gitt ved: $\lambda = c/f$, der c er lydhastighet i m/s og f er lydbølgens frekvens i hertz [MacLennan og Simmonds, 1992, s. 9-11]. Bølgelengden for de frekvensene som kan være aktuelle for dette systemet er:

- 50 kHz: 29 mm
- 70 kHz: 20,7 mm
- 200 kHz: 7,25 mm

Lydbølgen sprer seg sfærisk i vannet. Det medfører at energien avtar med kvadratet av avstanden. Når lydbølgen treffer et objekt i vannet vil de reflekterte lydbølgene også spre seg sfærisk ut fra objektet. Intensiteten i signalet som til slutt mottas av transduseren er dermed gitt ved $I_{inn} = I_{ut}*1/r^4$ [MacLennan og Simmonds, 1992, s. 20-21]. For å justere for dette forsterkes gjerne signalet med en tidsvariabel forsterker (TVG).

Med de transduserne som brukes av sonargruppa har lydbølgen som sendes ut en varighet på 0,5-1 ms og består av flere perioder av sendefrekvensen. For eksempel vil en lydbølge på 50 kHz som sendes ut med en varighet på 0,5 ms bestå av 25 perioder. Varigheten må være så lang for at transduseren skal rekke å begynne å svinge med fullt utslag [MacLennan og Simmonds, 1992, s. 27-29]. Mottatt signal som er reflektert fra en fisk har en varighet på 1-1,2 ms. Toppverdien av dette signalet sier noe om fiskens størrelse.

# 2. Analog elektronikk

2.1 BINDELEDD MELLOM KASSETTSPILLER OG FPGA ..... 14

2.2 KOMPONENTVALG ..... 14

2.3 KRETSBESKRIVELSE ..... 15

*2.3.1 Båndpassfiltrene ..... 16*

2.3.1.1 Båndpassfilter 10 kHz ..... 17

2.3.1.2 Båndpassfilter 70 kHz ..... 20

*2.3.2 Forsterker ..... 22*

*2.3.3 AD-omformer ..... 25*

2.3.3.1 Oppløsning: ..... 25

2.3.3.2 Samplingshastighet: ..... 25

2.3.3.3 Analog inngang: ..... 27

2.3.3.4 Digital utgang: ..... 27

2.3.3.5 LTC2356-14 ..... 27

*2.3.4 DA-omformer ..... 30*

*2.3.5 Trigger ..... 32*

*2.3.6 Strømforsyning ..... 33*

*2.3.7 Kontakter, lysdioder, trykknapper og lignende ..... 35*

*2.3.8 Kretskortet ..... 36*

## 2.1 Bindeledd mellom kassettspiller og FPGA

Mellom kassettspilleren og FPGA-en er det et kretskort som behandler ekkoloddsignalene før de samples og behandles av FPGA-en. Kretskortet inneholder all nødvendig elektronikk for å digitalisere kassetter, men har i tillegg noen kretser som kan brukes for å teste systemet under utvikling.

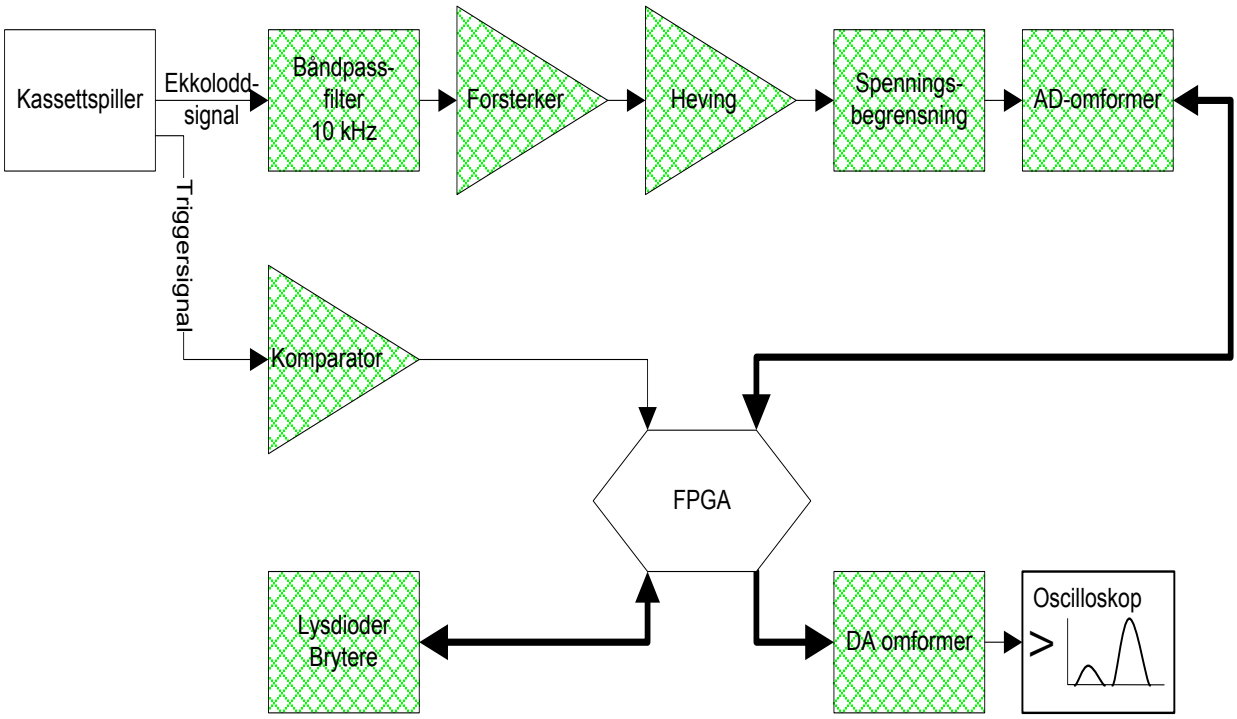
## 2.2 Komponentvalg

Selv om kretskortet i utgangspunktet skulle testes ved hjelp av opptak fra en kassettspiller, ble hovedelementene som ble benyttet valgt med utgangspunkt i at de skulle kunne brukes i det endelige designet. Elektronikken måtte derfor kunne brukes på signaler fra kassetter, dagens ekkolodd transduser og en eventuell ny transduser i framtiden. Flere av komponentene som ble valgt har derfor et mye bredere anvendelsesområde enn det som ville være nødvendig hvis vi kun skulle brukt dem til for eksempel en 70 kHz sonar.

For å unngå lange leveringstider og mange små bestillinger, ble komponenter som var tilgjengelige på Elektronikklaboratoriet ved Fysisk institutt (ELAB) benyttet i størst mulig grad. Det er både økonomisk, siden de kjøper inn i store kvanta, og fleksibelt, siden de har mange komponentverdier tilgjengelig. Dette er komponenter som de ansatte på ELAB har testet og kjenner godt fra før, så det ville også være lett å få hjelp hvis noe uventet skulle oppstå.

Vi har prøvd å bruke overflatemonterbare komponenter (SMD) der det finnes fordi det er enklere å montere og gir et mer kompakt kretskort enn hullmonterte komponenter, i tillegg til at de korte beina plukker mindre støy. Samtidig var det viktig at komponentene ikke var for små, slik at ikke kretskortet kunne produseres ved ELAB. Les mer om produksjon av kretskort ved ELAB i kapittel 2.3.8.

## 2.3 Kretsbeskrivelse



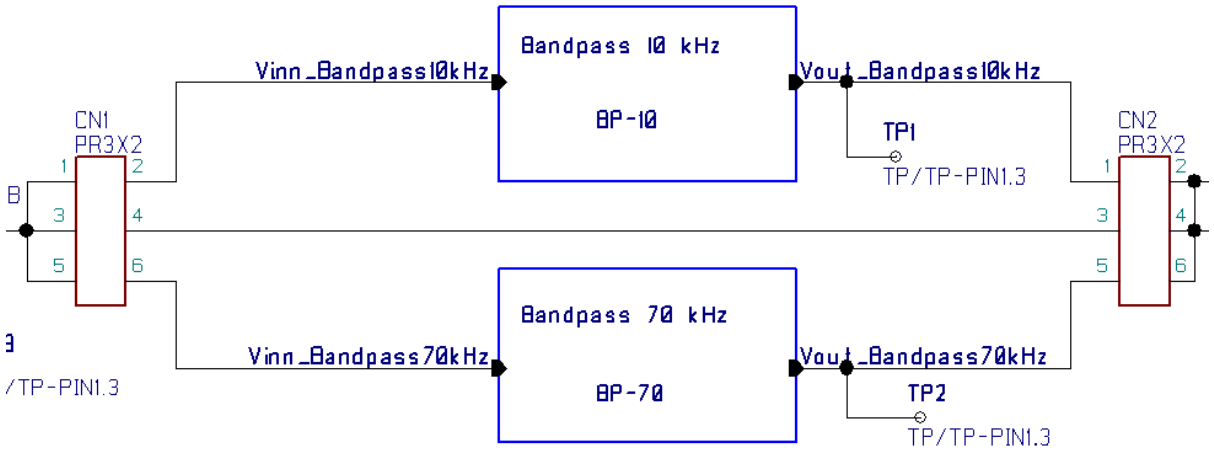

**Figur 2-1:** *Logisk skisse av modulene på kretskortet (Tegnet i Microsoft Visio).*

Signalene fra kassettspilleren filtreres, forsterkes og løftes før det samples i AD-omformeren. I tillegg er det en krets som gir en signalpuls til FPGA-en hver gang et nytt ping starter (se kapittel 1.2.2).

Kretskortet har flere funksjoner enn det som strengt tatt trengs for å digitalisere kassettene fordi det også er laget for å kunne teste funksjonaliteten mens systemet ble utviklet. Kretskortet har derfor noen komponenter som brukes for testing:

- et båndpassfilter med senterfrekvens på 70 kHz for å kunne teste systemet med reelle ekkoloddsignaler. Dette har ikke blitt brukt i prosjektet, men ligger klar for tilkobling. Vi hadde ingen transdusere på 200 kHz tilgjengelig, så derfor er det ingen båndpassfiltre for dette på kretskortet.
- DA-omformer som er koblet til FPGA-en, så signaler som er samplet og prosessert av FPGA-en kan legges ut på DA-omformeren og vises i oscilloskop.
- Lysdioder og brytere som er koblet til FPGA-en for status og kontroll av systemet.

### 2.3.1 Båndpassfiltrene




**Figur 2-2:** To båndpassfiltre og mulighet for å koble forbi (Skjermategning fra Zuken Cadstar).

Kretskortet inneholder to båndpassfiltre som er montert parallelt. Det ene har senterfrekvens på 10 kHz, mens det andre har 70 kHz. Hvilket filter som skal benyttes velges ved å montere jumpere på kontaktene CN1 og CN2. I tillegg er det mulig å koble seg forbi hvis man ikke ønsker å filtrere signalet, for eksempel ved digital filtrering på FPGA-en. Operasjonsforsterkerne er av typen NE5534AD [Philips Semiconductors, 1994] og er blant komponentene som er tilgjengelige på ELAB.

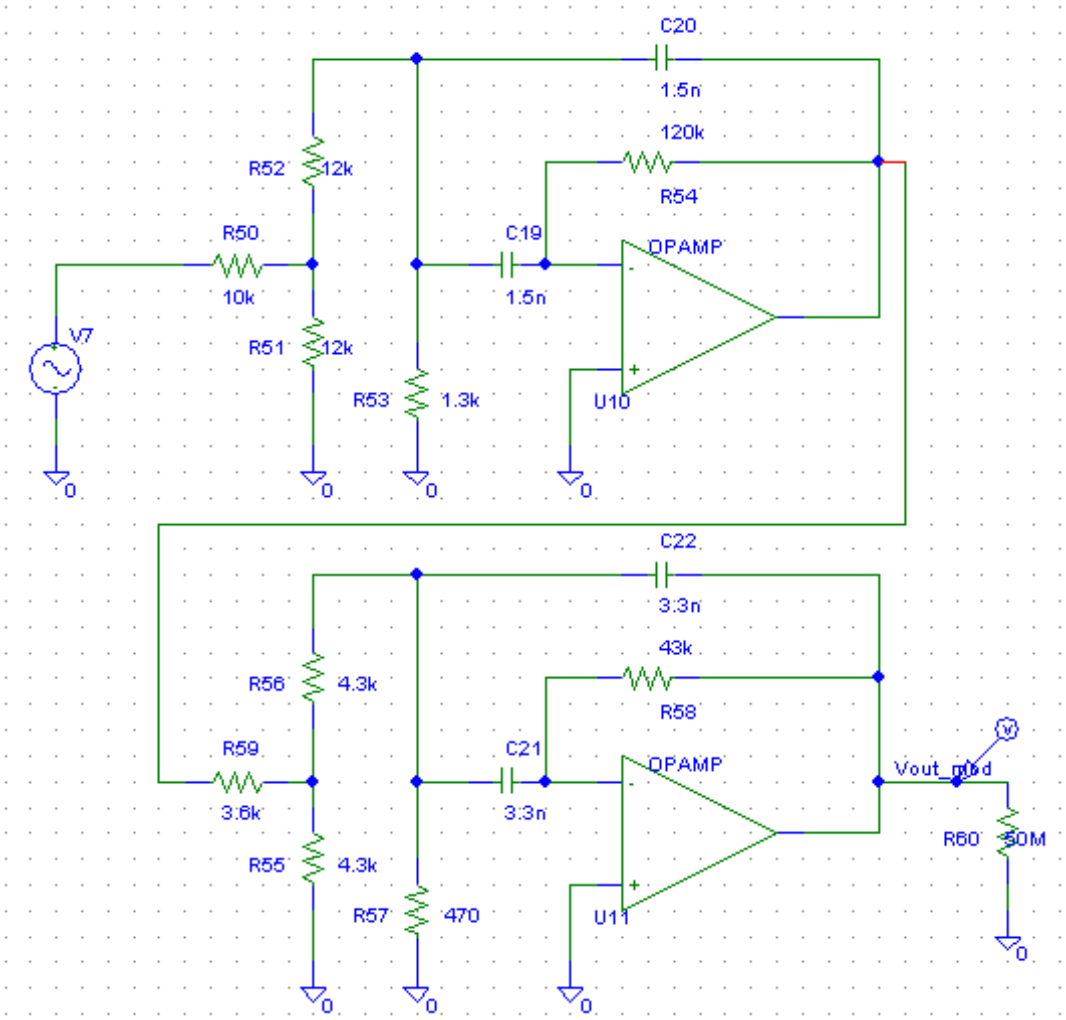
Begge filtrene består av to multiple-feedback filtre [Franco, 2002, s. 141-142] i kaskade, og det er kun komponentverdiene som skiller dem fra hverandre. Det var ikke ønskelig med større eller mer avanserte filtre fordi det krever flere komponenter og tar mer plass på kretskortet.

Filtrene ble designet i dataprogrammet FilterPro fra Texas Instruments. Ved å legge inn ønskede parametre, fikk vi beregnet de riktige komponentverdiene. For å fordele den totale forsterkningen utover flere operasjonsforsterkere ble det lagt litt forsterkning i båndpassfiltrene. Skjermbilder av FilterPro med de valgte verdiene er gjengitt i vedlegg E.

Etter at komponentene var valgt ble kretsforslaget simulert i PSpice for verifisering av resultatet. Begge filtrene måtte endres noe i forhold til resultatet fra FilterPro fordi ikke

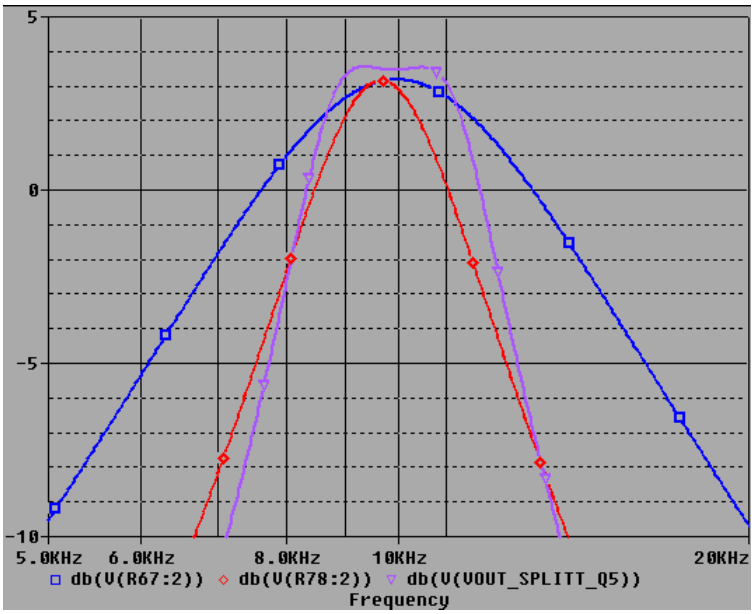
alle komponentverdiene var tilgjengelige på ELAB, og fordi justering av komponentverdiene ga bedre resultat.

#### 2.3.1.1 Båndpassfilter 10 kHz



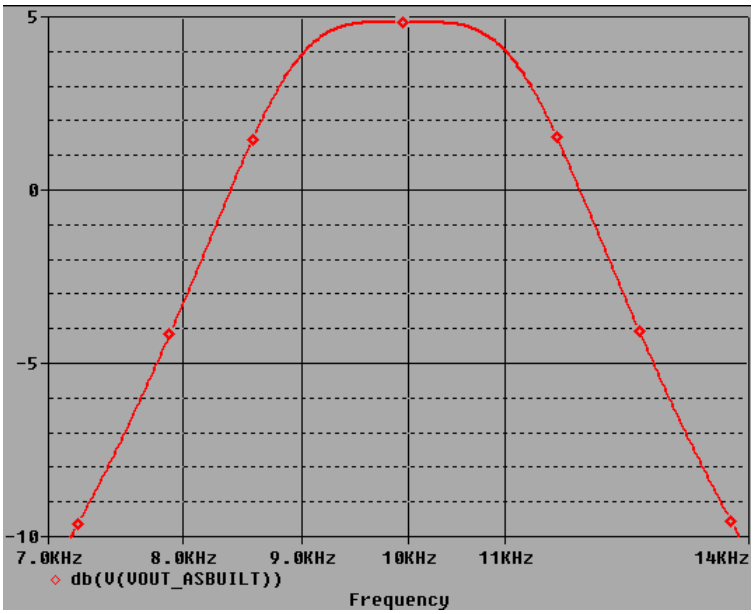

**Figur 2-3:** Båndpassfilter med senterfrekvens på 10 kHz (Skjermategning fra PSpice Schematics 9.1)

Båndpassfiltret med senterfrekvens på 10 kHz er designet slik at det har en bred og flat topp. Dette er ønskelig fordi frekvensen på signalet fra kassettspilleren kan drive vekk fra 10 kHz ved for eksempel hastighetsendring på kassettavspillingen. Filteret kan også bomme på senterfrekvensen på grunn av unøyaktige komponentverdier og temperatur (se kurven merket med ◊ (rød) i figur 2-4). Ved å ha en bred og flat topp unngår vi dermed at nyttesignalet vårt ligger på en av flankene.




**Figur 2-4:** Eksempel på AC-sweep for tre ulike båndpassfiltre. Filteret merket med □ (blå) har Q-faktor 1,9. ◇ (rød) har Q-faktor 2,8. ▼ (lilla) er stagger-tuned og har en Q-faktor på 3,1. (Skjermbilde fra Orcad PSpice 9.1)

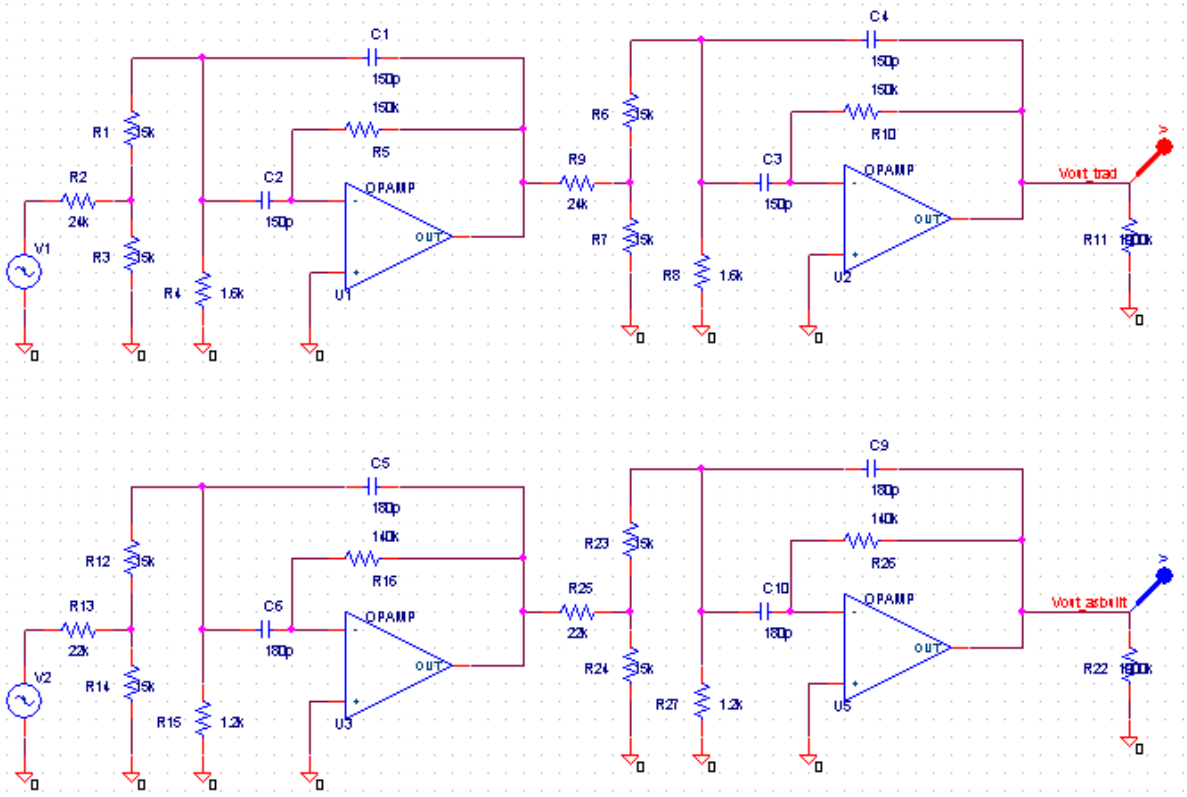
Dette kan løses ved å ha et slakt båndpassfilter med lite dempning til side for senterfrekvensen, men vi valgte å lage et stagger-tuned filter. Det lages ved å sette to båndpassfiltre etter hverandre, der det ene filteret har en senterfrekvens som er lavere enn den ønskede senterfrekvensen, mens det andre har høyere senterfrekvens. Hvis filtrene settes riktig i forhold til hverandre, får man et forholdsvis skarpt båndpassfilter med en flate ved senterfrekvensen. I figur 2-5 ser vi tydelig at filteret som er stagger-tuned er flatt i et bredt område rundt senterfrekvensen, men har likevel den høyeste Q-faktoren.




**Figur 2-5** Simulering av AC-sweep på båndpassfilteret med senterfrekvens på 10 kHz. Legg merke til den flate toppen som oppnås ved å sette to smale båndpassfiltre ved siden av hverandre. (Skjermbilde fra Orcad PSpice 9.1)

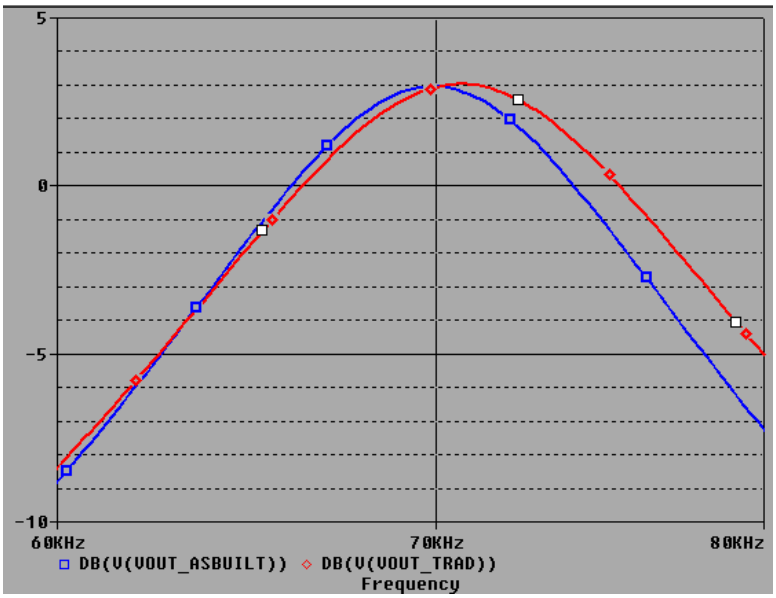
Etter at komponentverdiene var valgt i FilterPro måtte kretsen endres og simuleres før vi fikk ønsket respons med tilgjengelige standardkomponenter. Simulering av den endelige kretsløsningen vises i figur 2-5.

### 2.3.1.2 Båndpassfilter 70 kHz



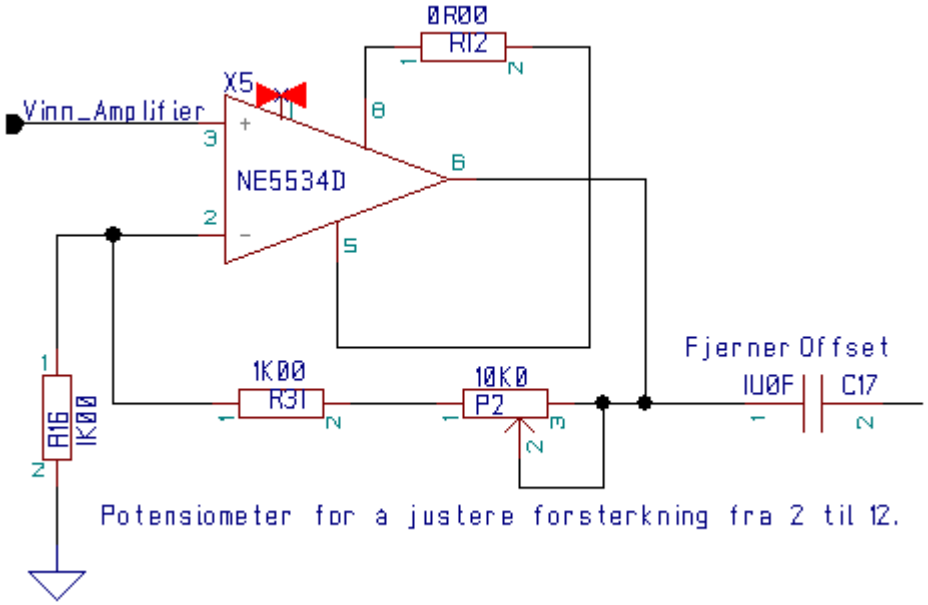

**Figur 2-6** Kretsskjema for båndpassfilter 70 kHz med komponentverdier fra FilterPro øverst og de komponentverdiene som ble valgt nederst. (Skjermbilde fra Orcad Capture CIS 10.3)

Båndpassfilteret med senterfrekvens på 70 kHz består av to identiske multiple-feedback filtre i kaskade. Etter at komponentverdiene var valgt i FilterPro måtte kretsen endres og simuleres før vi fikk ønsket respons med tilgjengelige standardkomponenter. Simulering av den endelige kretsløsningen vises i figur 2-7.



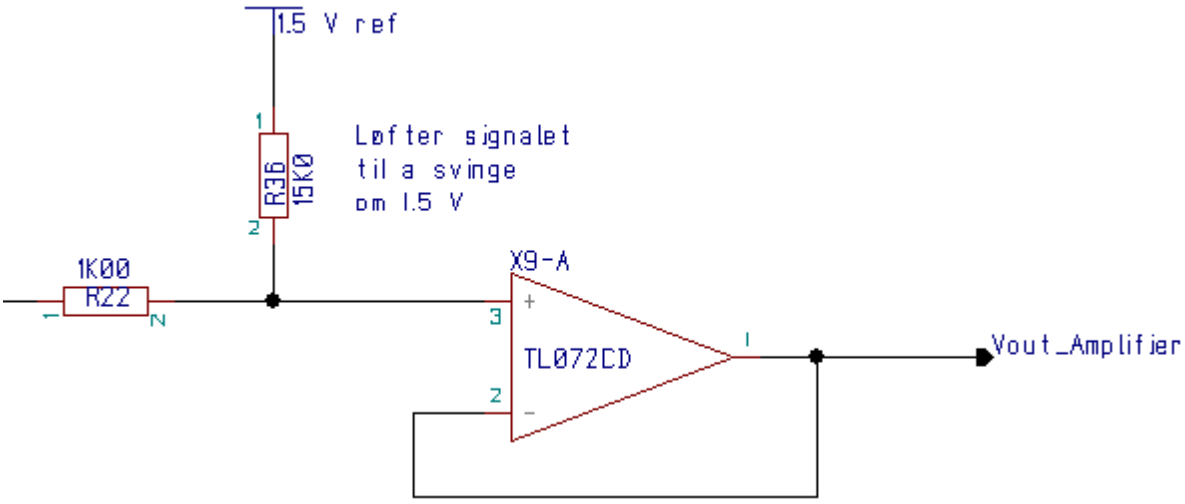

**Figur 2-7** Sammenligning av kurve for kretsen fra FilterPro ◊(rød) og kretsen slik den ble bygget □(blå). (Skjermbilde fra Orcad PSpice 10.3)

## 2.3.2 Forsterker




**Figur 2-8:** Forsterker signalet 2-12 ganger (Skjermategning fra Zuken Cadstar).

Signalet må forsterkes før det samples, slik at signalene man ønsker å se på blir store nok til at de gir utslag på AD-omformeren. Båndpassfiltrene forsterker nyttesignalene noe, mens den resterende forsterkningen gjøres av en ikke-inverterende forsterker [Franco, 2002, s. 8-11]. For en ideell ikke-inverterende forsterker er forsterkningen gitt ved $A = 1 + \frac{R_2}{R_1}$, der potensiometeret P2 og motstanden R31 tilsvarer $R_2$ og motstanden R16 tilsvarer $R_1$. Hvis potensiometeret P2 settes til 0 Ω blir forsterkningen $A = 1 + \frac{0\Omega + 1k\Omega}{1k\Omega} = 2$. Hvis potensiometeret P2 settes til høyeste verdi som er 10 kΩ, blir forsterkningen $A = 1 + \frac{10k\Omega + 1k\Omega}{1k\Omega} = 12$.




**Figur 2-9:** Løfter signalet til å svinge om 1,5 V før det kan samples (Skjermategning fra Zuken Cadstar).

AD-omformeren krever at signalet ligger mellom 0 V og 3 V (kapittel 2.3.3.5). Signalet må derfor løftes fra å svinge om 0 V til å svinge om 1,5 V.

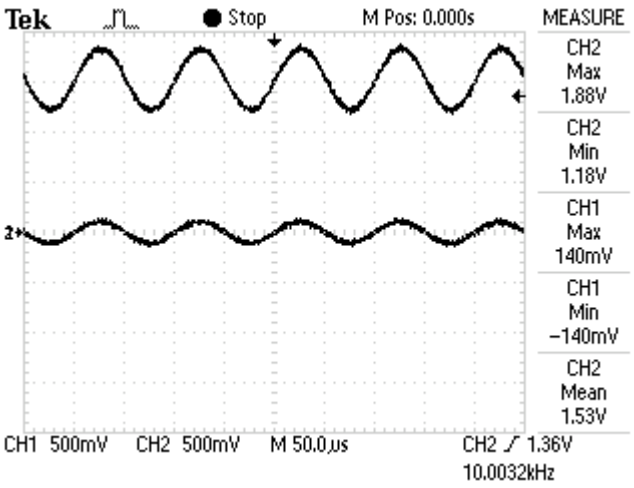
Kondensatorer i signalbanen mellom komponentene stopper likespenning. Ved å sette inn kondensatorer og la signalet svinge om 0 V unngår man at offsetspenninger fra de ulike komponentene blir forsterket gjennom kretsen. Figur 2-8 viser at all eventuell likespenning etter forsterkertrinnet fjernes med en kondensator. Det er viktig å ha kontroll med offset så signalet ikke drifter utenfor AD-omformerens arbeidsområde.

Etter at signalet er løftet til å svinge om 1,5 V, kan man ikke ha flere kondensatorer i signalbanen. Dersom det er et problem at det legges til offset i resten av kretsen fram til AD-omformeren, kan dette fjernes etter at signalet er samplet. Det kan gjøres ved at FPGA-en regner ut gjennomsnittsverdien av signalet og deretter trekker denne verdien fra hvert sampel.

For å løfte signalet brukes en summasjonsforsterker [Franco, 2002, s. 17-19] som vist i figur 2-9. Det er ingen motstand i tilbakekoblingssløyfen for at offsetspenningen (og signalet) ikke skal bli forsterket.

Referansespenningen på 1,5 V er laget ved spenningsdeling av den regulerte spenningen på 3,3 V. Eventuelle spenningsavvik får ikke betydning fordi den samme

spenningen brukes som referansespenning for AD-omformeren, og har derfor samme nivå.




**Figur 2-10:** Kanal 1: inngang, kanal 2: utgang etter forsterkning og heving av signalet (Bilde fra oscilloskop med ScopePrinterBMP)

### 2.3.3 AD-omformer

For å kunne behandle ekkoloddsignalene i FPGA-en trenger vi en AD-omformer for å konvertere det analoge signalet til digitale verdier. For at AD-omformeren skulle være i stand til å sample bærebølgen måtte den oppfylle en del krav.

#### *2.3.3.1 Oppløsning:*

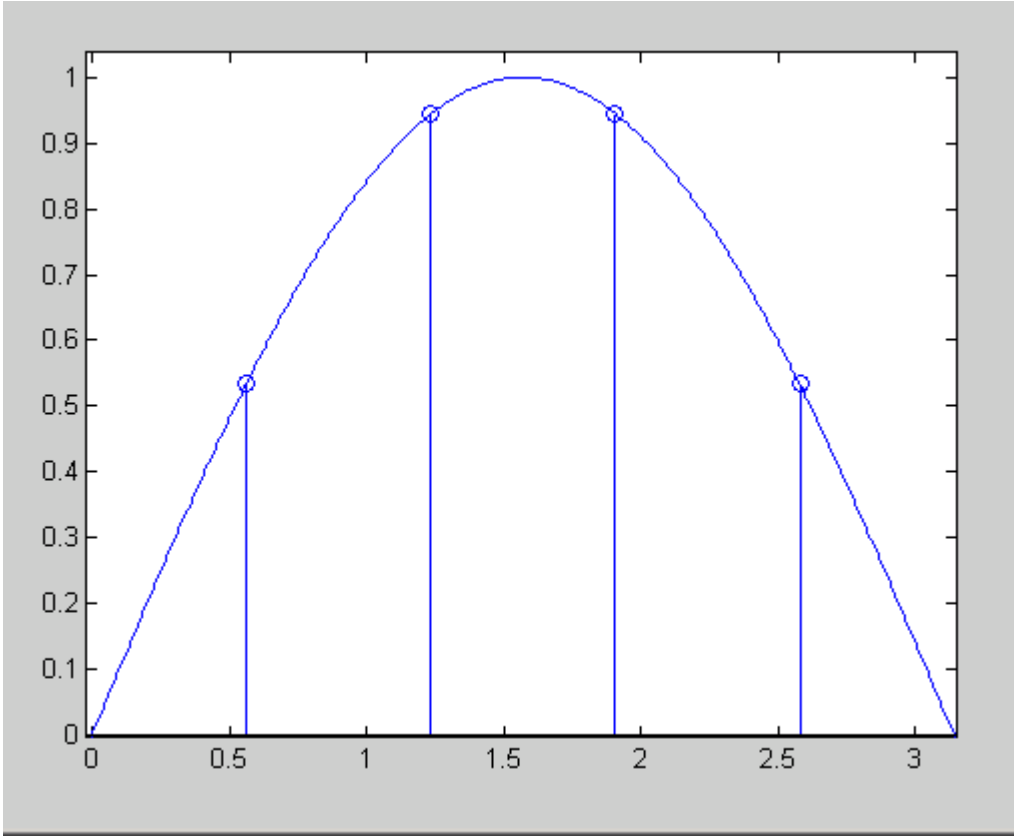
Ekkoloddsystemet som brukes i dag har en AD-omformer med 8 bit oppløsning. Oppløsningen på det nye system skulle minimum være 8 bit, men gjerne høyere. Samtidig er det ikke nødvendig å kunne skille spenningsnivåer som er mindre enn støyen i kretsen.

Fordi det måtte lages et grensesnitt mot AD-omformeren i FPGA-en (Se kapittel 3.8.1) ble AD-omformeren valgt ut før kretskortet var ferdig designet. Det er også ønskelig at den samme AD-omformeren brukes ved en senere utvikling av kretskort for ekkoloddet fordi den nå er testet.

Fordi det ikke var mulig å beregne støynivået i kretsene på forhånd, ble kravet kun at AD-omformeren skulle ha høyere oppløsning enn 8 bit. Dersom det senere viser seg at støyen i kretsen er større enn 1 LSB, kan man se bort fra de bittene som påvirkes av støyen.

#### *2.3.3.2 Samplingshastighet:*

Ved sampling av analoge signaler må samplingshastigheten minimum være $2*f_{max}$, der $f_{max}$ er den høyeste frekvenskomponenten i signalet som skal samples, for at signalet skal kunne rekonstrueres. Denne grensen kalles Nyquist samplingshastighet [Natås, 2005, s. 126].




**Figur 2-11**: Samplene treffer lengst fra toppen når to sampler er like langt fra toppunktet (Plot fra MATLAB).

Fordi vi skal hente ut den høyeste verdien i perioden må vi sample mye oftere enn Nyquist samplingshastighet for å være sikker på at et av samplene treffer så nærme toppen som mulig. Som et utgangspunkt ble det bestemt at AD-omformeren måtte kunne sample signalet så ofte at den største differansen mellom signalets toppunkt og høyeste samplingspunkt er 0,5 dB.

Største differanse mellom toppunkt og samplingspunkt er når to samplingspunkt er like langt fra toppunktet. Hvis denne avstanden skal være mindre enn 0,5 dB, må samplingshastigheten være minst 9,3483 ganger raskere enn frekvensen som skal samples (se utregning i vedlegg B). Vi tok derfor utgangspunkt i at AD-omformeren måtte ha en samplingshastighet på minimum 10 * $f_{max}$. Den høyeste frekvensen dette systemet skal kunne motta er 200 kHz, så minimum samplingshastighet ble satt til 2 MSPS.

AD-omformeren skal også kunne sample bærebølger som er mikset ned til 10 kHz på kassetter. En oversampling på 10 ganger vi i så fall gi 100 kSPS. Det finnes ikke så

mange AD-omformere som både kan sample raskere enn 2 MSPS, og som også kan sample ved 100 kSPS. Dersom AD-omformeren ikke kan sample ved 100 kSPS kan dette løses ved at den sampler ved en høyere hastighet, mens FPGA-en leser ut samplene med 100 kSPS.

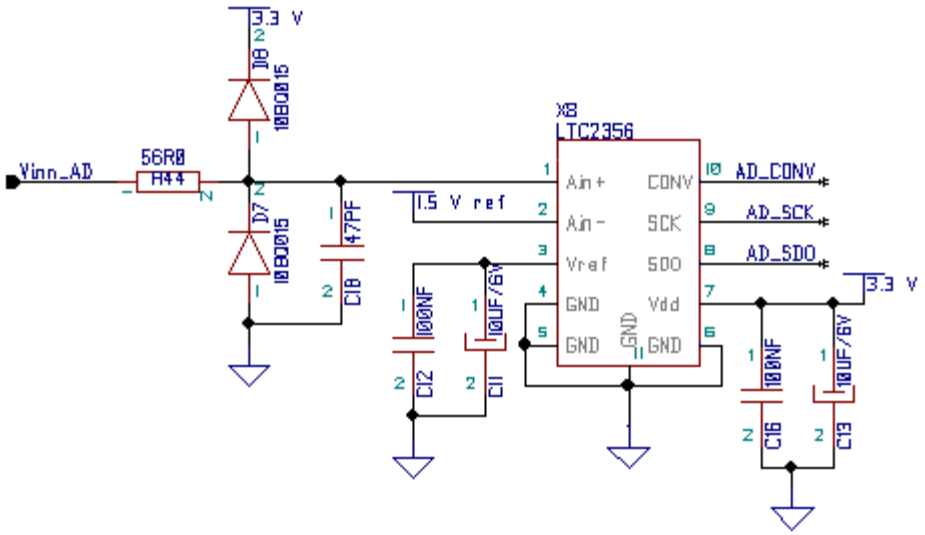
### 2.3.3.3 Analog inngang:

For å få et godt signal-støy-forhold er det ønskelig at det analoge signalet inn på AD-omformeren kan være størst mulig. Det er også ønskelig at AD-omformeren kan sample bipolare signaler, slik at offset kan fjernes i forkant. Dessverre finnes det få AD-omformere med høy samplingshastighet som oppfyller disse kravene.

### 2.3.3.4 Digital utgang:

Det digitale grensesnittet på AD-omformere er enten parallelle eller serielle. Fordi det uansett måtte lages en VHDL-driver for AD-omformeren i FPGA-en, og fordi FPGA-en har så mange ubrukte pinner, kan begge overføringsmetodene brukes. Spenningsnivået på FPGA-ens digitale innganger og utganger er 2,5 V eller 3,3 V [Xilinx, UG454, s. 21], så spenningsnivået på AD-omformerens I/O må være kompatibel med dette.

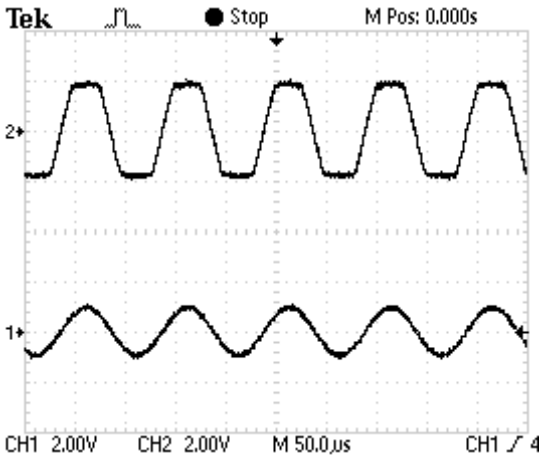
### 2.3.3.5 LTC2356-14




**Figur 2-12:** Skjemategning av AD-omformeren LTC2356 (Skjermategning fra Zuken Cadstar).

Det er ikke lett å finne en AD-omformer som oppfyller alle kravene og som også er tilgjengelig i Norge. Valget falt til slutt på LTC2356-14 fra Linear Technology [Linear, LTC2356]. Det er en 14-bits AD-omformer med samplingshastighet på opptil 3,5 MSPS. Det digitale grensesnittet er serielt og basert på SPI. Grensesnittet er beskrevet nærmere i kapittel 3.8.1. De digitale signalene har et spenningsnivå på 0 - 2,5 V, og er dermed kompatible med FPGA-en.

I databladet [Linear, LTC2356] oppgir de at den analoge inngangsspenningen kan være bipolar ±1,25 V. Ved nærmere studering av databladet, etter at AD-omformeren var kjøpt inn, viste det seg at signalet kan svinge ±1,25 V rundt et spenningsnivå på 1,5 V [Linear, LTC2356-14, s. 11]. Vi ble derfor nødt til å løfte signalet til å svinge rundt 1,5 V i forkant av AD-omformeren (se kapittel 2.3.2).




**Figur 2-13:** Spenningsbegrensning beskytter inngangen på AD-omformeren. Kanal 1: inngang, kanal 2: Forsterket og klippet signal før AD-omformeren. (Bilde fra oscilloskop med ScopePrinterBMP)

AD-omformeren er koblet opp som vist i figur 2-12. Det analoge signalet, Vinn_AD, går gjennom et lavpassfilter og spenningsbegrensning før det kobles til inngangen Ain+.

Spenningsbegrensningen består av to schottkydioder av typen 10BQ015 [International Rectifier, 10BQ015] som er koblet mot jord og 3,3 V. De hindrer at signalet til den analoge inngangen på AD-omformeren er lavere enn -0,3 V og høyere enn 3,6 V som er de maksimale spenningene AD-omformeren tåler [Linear, LTC2356, s. 2]. AD-omformeren har også innebygde dioder for spenningsbegrensning som kan lede inntil

100 mA uten at den ødelegges [Linear, LTC2356, s. 4], men siden AD-omformeren koster 200-300 kr mener vi det er fornuftig å beskytte den ekstra godt.

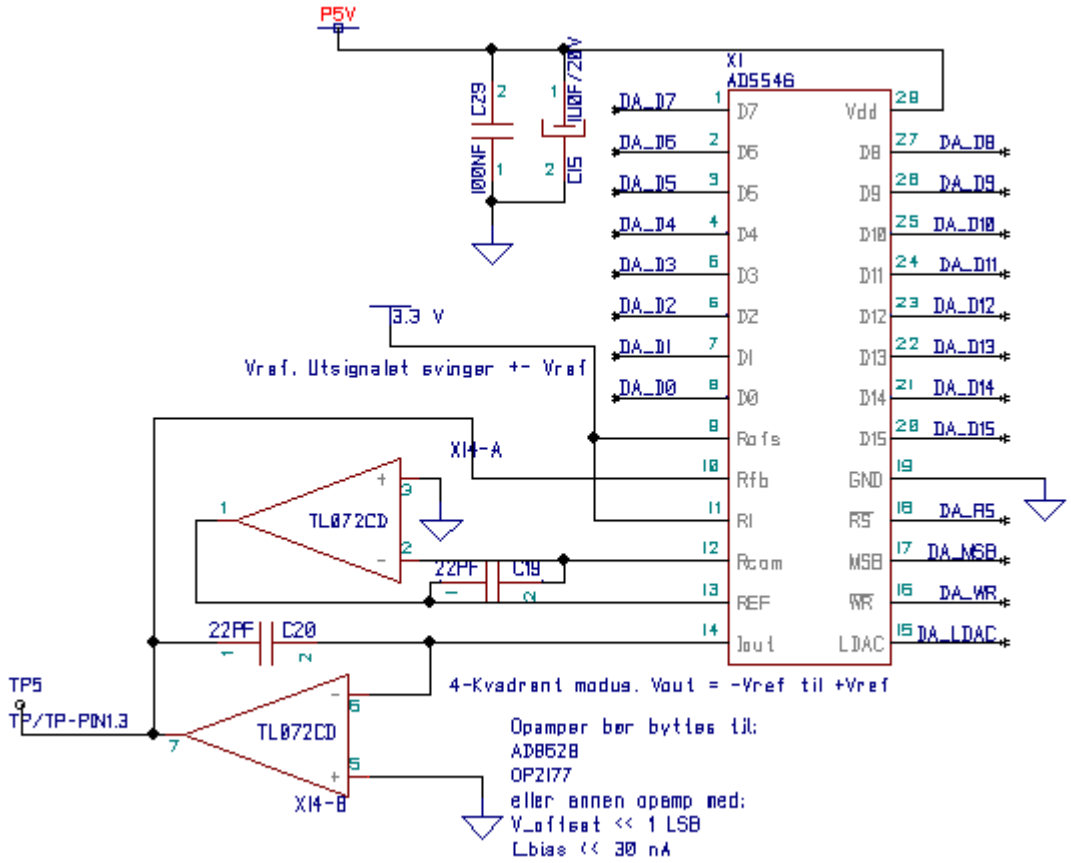
Lavpassfilteret består av en motstand på 56 Ω (R44 i figur 2-12) og en kondensator (C18) på 44 pF, og er satt inn etter anbefaling fra databladet [Linear, LTC2356, s. 11]. Det har flere hensikter ved siden av å filtrere bort støy. Kondensatoren fungerer også som et ladningsreservoar for AD-omformerens sample-and-hold-krets [Linear, LTC2356, s. 11], og motstanden i filteret fungerer også som last når schottkydiodene i spenningsbegrensningen leder, så utgangsmotstandene på operasjonsforsterkeren ikke belastes.

Inngangen Ain- kobles til en spenningsreferanse på 1,5 V. Dette er den samme spenningsreferansen som brukes for å løfte signalet til å svinge om 1,5 V (se kapittel 2.3.2), så eventuelle avvik i spenningsreferansen vil bli kansellert av den differensielle inngangen på AD-omformeren.

AD-omformeren trenger en spenning på 3,3 V for å fungere. Det kunne være mulig å bruke 3,3 V spenningen fra utviklingskortet for FPGA-en [Xilinx, UG454], men fordi vi ikke vet om den er godt nok regulert og inneholder lite støy, har vi en egen 3,3 V spenningsforsyning på kortet (se kapittel 2.3.6). Denne er avkoblet med en keramisk- og en elektrolyttkondensator for å fjerne støy.

Fordi AD-omformeren ikke hadde vært brukt ved ELAB før måtte komponenten tegnes i Zuken Cadstar. Komponenten ligger vedlagt på CD-en.

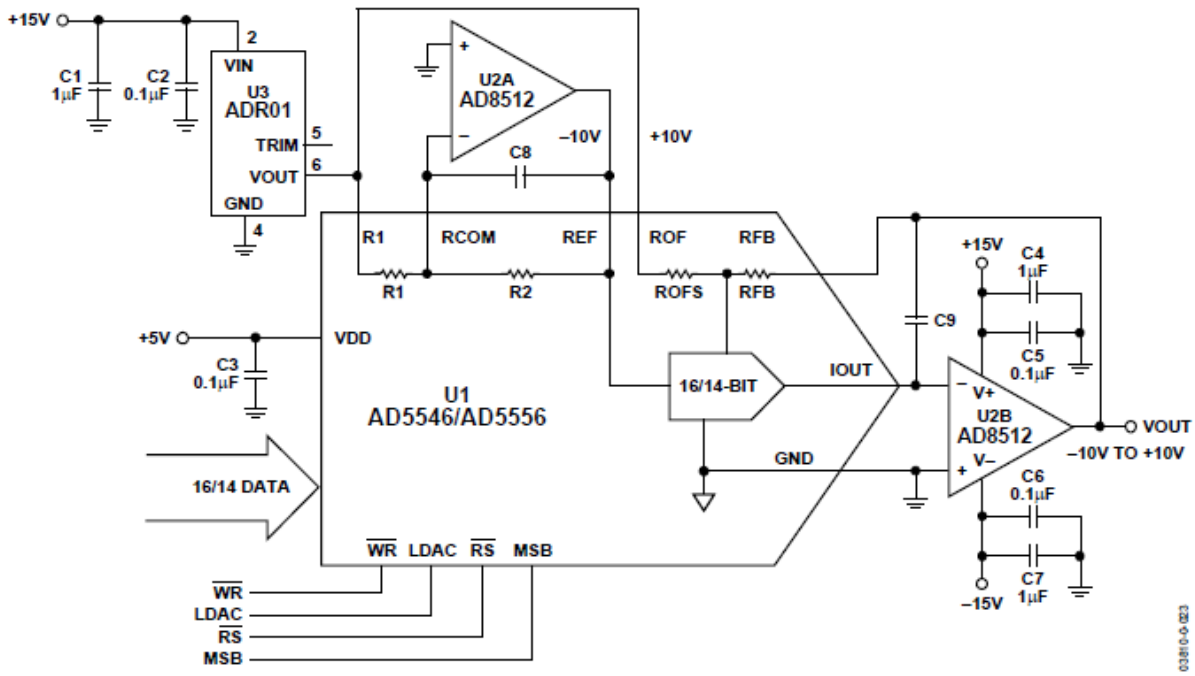
### 2.3.4 DA-omformer




**Figur 2-14:** Skjemategning av DA-omformeren AD5546 (Skjermategning fra Zuken Cadstar).

En DA-omformer ble montert på kretskortet slik at verdier fra FPGA-en kunne konverteres til analoge signaler som kunne vurderes på oscilloskopet. Dermed kunne vi teste algoritmene som ble utviklet for FPGA-en før prosessorsystemet med Ethernet-kommunikasjon var på plass. DA-omformeren måtte kunne gjengi verdiene som AD-omformeren samplet, så den måtte ha minst 14-bits oppløsning og en konverteringshastighet på minst 2 MSPS.

Vi valgte DA-omformeren AD5546 fra Analog Devices [Analog Devices, AD5546]. Denne har 16-bits oppløsning og en innsvingningstid (eng: settling time) på 0,5 µs, som gjør at den er rask nok til å konvertere dataene FPGA-en mottar fra AD-omformeren. Den er også godt egent for å styre kretser for tidsvariabel forsterkning (TVG) i ekkolodd. Det digitale grensesnittet er parallelt og er beskrevet i kapittel 3.8.9.



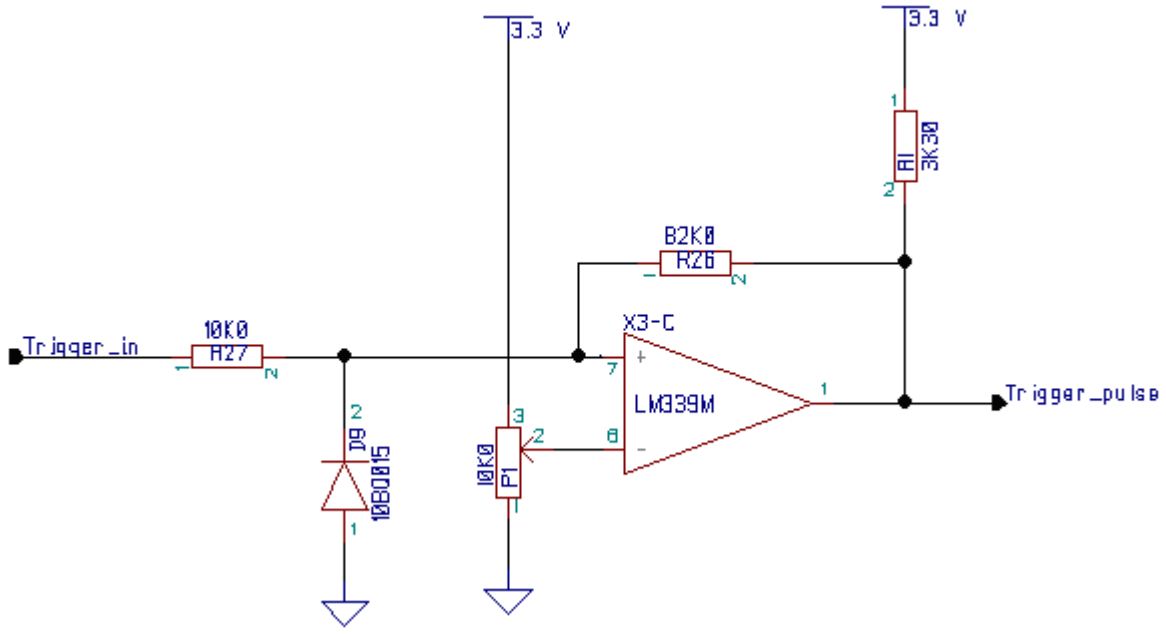

**Figur 2-15:** Oppsett av DA-omformeren AD5546 [Analog Devices, AD5546, s. 14].

DA-omformeren kan konfigureres på ulike måter, men i dette prosjekter er den satt i henhold til skissen fra databladet som er gjengitt i figur 2-15. Kretsskjemategningen er vist i figur 2-14. Dette oppsettet gjør at utgangen blir bipolar og svinger om 0 V, i likhet med signalene fra kassettspilleren. Utspenning svinger innenfor ±Vref, som i dette tilfellet blir ±3,3 V.

Operasjonsforsterkerne som er koblet til DA-omformeren er av typen TL072CD [Texas Instruments, TL072]. Denne ble valgt fordi den var blant komponentene som var tilgjengelige på ELAB og har gode nok parametre for testformål. [Analog Devices, AD5546, s. 12] anbefaler spesifikt operasjonsforsterkerne OP1177 og DA8628, men enhver operasjonsforsterker med lavere bias-strøm enn 30 nA og offsetspenning mindre enn 1 LSB er egnet. TL072CD har lav bias-strøm (typisk 65 pA), men offsetspenningen på typisk 3 mV [Texas Instruments, TL072, s. 7] er for høy i forhold til 1 LSB ($\frac{6{,}6V}{2^{16}\,bit} = 0{,}1\,{}^{mV}\!/_{bit}$).

Fordi DA-omformeren ikke hadde vært brukt ved ELAB, før måtte komponenten tegnes i Zuken Cadstar. Komponenten ligger vedlagt på CD-en.

### 2.3.5 Trigger




**Figur 2-16:** Skjemategning av triggerkrets (Skjermategning fra Zuken Cadstar).

På kassettenes L-kanal er det spilt inn et triggersignal som indikerer at pinget sendes ut. Signalet fremstår som en ujevn puls med utslag på ±1 V. For at FPGA-en skal registrere triggersignalet, må den motta en puls på 3,3 V. For at ikke pulsen fra komparatoren skal prelle hvis pulsen på kassetten er ustabil, ønsket vi å ha noe hysterese i kretsen. Vi lagde derfor en Schmitt-trigger basert på [Franco, 2002, s. 416-422] og kretstegningen for en ikke-inverterende komparator med hysterese i [National, LM339, s.10].

Komparatoren som er brukt er LM339 fra National [National, LM339]. Ved å endre på potensiometeret (P1), endres referansespenningen for hvor komparatoren skal slå inn fra mellom 0 V og 3,3 V.

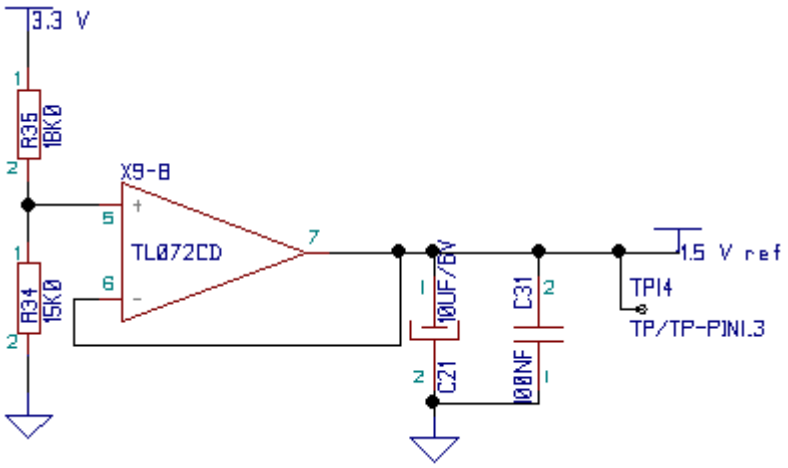
Zenerdioden (D9) er satt inn for å beskytte komparatoren mot negative spenninger, og alle pinner på de ubrukte komparatorene i pakken er koblet til den negative spenningsforsyningen, i henhold til [National, LM339, s.6].

### 2.3.6 Strømforsyning

Ekkoloddene drives normalt av en 12 V spenningskilde (for eksempel bilbatteri) og svinger rundt 6 V, men utgangssignalet fra kassettspilleren svinger om 0 V med maksimalt amplitudeutslag på ±1 V. Det var derfor nødvendig med både positiv og negativ forsyningsspenning til kretskortet. Det derfor koblet en ekstern spenningsforsyning til kretskortet som gir +12 V og -12 V.

DA-omformeren trenger +5 V, og denne genereres fra +12 V spenning av LM78L05 [National Semiconductor, LM78LXX]. Denne er satt opp i henhold til skjemategning i [National Semiconductor, LM78LXX, s. 9]. Vi har for sikkerhetsskyld lagt en schottkydiode mellom inngangen og utgangen, så den leder strøm og tømmer kondensatorene på utgangen når spenningen på inngangen slås av, slik at utgangen på spenningsregulatoren ikke blir ødelagt.

Kretskortet trenger også +3,3 V spenning til AD-omformeren og referansespenninger på kortet. XC6204B33 [Torex Semiconductor, XC6204] genererer 3,3 V fra 5 V, og er satt opp i henhold til [Torex Semiconductor, XC6204, s. 17]. Det er kun én spenningsregulator på 3.3 V for både referansespenning og drift av AD-omformeren, fordi AD-omformerens strømforbruk er så lavt at kondensatorene kompenserer for støy den genererer.




**Figur 2-17:** Spenningsreferanse 1,5 V (Skjemategning fra Zuken Cadstar).

AD-omformeren trenger en spenningsreferanse på 1,5 V, og den er laget ved spenningsdeling av 3,3 V. Motstandene er etterfulgt av en spenningsfølger [Franco, 2002, s. 11-12] for at ikke lasten som kobles til skal påvirke spenningsnivået.

Alle strømforsyningene er etterfulgt av kondensatorer for å fjerne støy og sørge for stabil strømtilførsel. Det er plassert en 10 uF elektrolytt og 100 nF keramisk i parallell etter spenningsforsyningene for ±12 V og 1,5 V fordi den keramiske kondensatoren fjerner høyfrekvent støy mer effektivt enn elektrolyttkondensatoren. For spenningsforsyningene på 3,3 V og 5 V har vi fulgt anbefalingene fra databladet.

En keramisk kondensator på 100 nF er plassert tett inntil alle IC-er, der databladet ikke spesifiserer noe annet. Disse sørger for en jevn og støyfri spenningsforsyning til IC-ene, i tillegg til at de leder strømmen fra bunnlaget til topplaget som beskrevet i kapittel 2.3.8.

### 2.3.7 Kontakter, lysdioder og trykknapper

I tillegg til kretsene som allerede er beskrevet er følgende komponenter montert på kretskortet:

- **EXP-kontakt**
  - To 120-pinners kontakter av typen Samtec QTE-060-09-F-D-A [Samtec, QTE] er montert på utviklingskortet for FPGA-en og er koblet til beina på FPGA-en [Xilinx, UG454]. For å koble kretskortet til utviklingskortet for FPGA-en er det montert en kontakt av typen Samtec QSE-060-01-F-D-A [Samtec, QSE] på kretskortet, og disse er sammenkoblet med en flatkabel av typen Samtec HFEM2-060-T-03.00-SE [Samtec, HFEM2-SE]. Flatkablen kobles til JX2 på utviklingskortet for FPGA-en (Xilinx Spartan 3A DSP Starter Platform) [Xilinx, UG454].
  - Fordi kontakten som er montert på utviklingskortet (Samtec QSE-060-01-F-D-A) ikke hadde vært brukt ved ELAB før, måtte komponenten tegnes i Zuken Cadstar. Komponenten ligger vedlagt på CD-en.
- **Phonokontakter**
  - Kassettspilleren kobles til kretskortet med en phonokontakt for hver av de to stereokanalene. Den sorte er for triggersignalet på L-kanalen, mens den røde er for ekkoloddsignalet på R-kanalen.
- **Lysdioder og trykknapper**
  - Det er montert to lysdioder som lyser når det er spenning på kretsen.
  - EXP-kontakten er koblet til seks lysdioder og fire trykknapper som kan brukes av FPGA-en.

### 2.3.8 Kretskortet

Kretskortet ble produsert ved Elektronikklaboratoriet ved Fysisk institutt (ELAB) da det er billigere og raskere enn å produsere eksternt. Der er det utstyr for å etse tolags kretskort uten gjennomplettering og montere overflatemonterte komponenter (SMD) ved ovnslodding. Overflatemonterte komponenter ble plassert manuelt og loddet med "reflow"-lodding i IR-ovn. Hullmonterte komponenter ble loddet for hånd til slutt.

Kretskortutlegget er laget etter følgende designregler og prinsipper:

- Etseprosessen som benyttes krever at banebredden minimum må være 12 mills (0,305 mm) og mellomrom mellom baner minimum må være 9 mills (0,229 mm). Beina til AD-omformeren står så tett at vi måtte endre designreglene "Pad to pad spacing" og "Route to pad spacing" i Zuken Cadstar fra 0,229 mm til 0,195 mm.
- Komponentene er gruppert på kretskortet etter hvilken funksjon de hører til (båndpassfilter, forsterker, AD-omformer osv.). Det gjør kretskortet oversiktlig ved feilsøking.
- Komponentene er plassert strukturert og tett for å utnytte plassen på kortet på en best mulig måte. Det gjør det også mye lettere å rute designet, og man får kortere ledningsbaner. Korte ledningsbaner er en fordel både for å få så lavt spenningsfall som mulig og for å få mindre strømsløyfer som kan generere og plukke opp støy (EMC). I tillegg blir det lettere å foreta visuelle inspeksjoner.
- Alle komponentene er plassert med samme orientering, med kun noen få unntak. Det gjør sannsynligheten for feilplasseringer mindre ved en eventuell automatisert komponentplasseringsprosess. Fordi vi skulle plassere komponentene for hånd, ble det likevel gjort andre prioriteringer ved komponentplasseringen i enkelte tilfeller.
- For best mulig jording over hele kortet er jordlederen rutet i et stjernemønster fra et felles jordingspunkt, slik at ledningsbanen til hver enkelt komponent er kortest mulig.
- All ruting er gjort med optimal bredde på ledningsbanene for å få lavest mulig motstand i ledningsbanene ($R_{ledningsbane} = R_{sq}$*Lengde/Bredde, der $R_{sq}$ er motstanden for en kvadratmeter av ledningsmaterialet (kobber)) [Halbo og Ohlckers, 1995, s. 6.5]. Inn på små kontaktflater er det brukt redusert bredde på ledningsbanen for å unngå at loddetinnet flyter ut.
- For å få kortest mulig ledningsbaner er rutingen optimalisert ved at ledningsbanene går skrått med en vinkel på $45^o$ der det lar seg gjøre.
- Fordi kretskortet ikke er gjennomplettert, kan man ikke bruke via-hull for å lede strømmen mellom oversiden og undersiden av kortet.
    - I signalbanen skifter signallederen side der det også er naturlig å ha testpunkter i form av hullmonterte testpinner.

    - For å lede spenningsforsyningen fra undersiden av kretskortet til overflatemonterte komponenter på oversiden, er det brukt hullmonterte avkoblingskondensatorer. Under produksjonen av kortet monterte vi avkoblingskondensatorene til slutt. Ved å montere en og en avkoblingskondensator kunne vi teste deler av kretskortet, mens de umonterte kretsdelene var spenningsløse. Dermed unngikk vi å ødelegge komponenter på grunn av feil andre steder i kretsen. Det er også enklere å finne feil fordi man får testet hver enkelt kretsdel for seg. Teknikken med å bruke hullmonterte avkoblingskondensatorer på tolags kretskort er etter råd fra Stein Lyng Nielsen ved ELAB:
- Hullmonterte komponenter som for eksempel phonokontaktene kan ikke loddes på oversiden, så da går ledningsbanene til komponenten på undersiden av kretskortet.
- Operasjonsforsterkeren NE5534 har pinner for kompensasjonskondensatorer [Philips Semiconductors, 1994]. Disse er ikke montert. For å sette av plass på kretskortet til eventuell senere montering, er det tegnet nullohmsmotstander i størrelse 1206 der de skal være. Ingen nullohmsmotstander skal monteres!

Kretskortutlegget er vist i vedlegg D, og ligger på den vedlagte CD-en.

Under produksjonen av kretskortet oppdaget vi en del feil ved det opprinnelige utlegget:

- QSE-kontakten manglet styrepinner, og paddene stakk for lite ut.
- QSE-kontakten var tegnet feil, så pin 1 var der pin 119 skulle være osv. Rettingen har ført til at andre pinner på FPGA-en må brukes enn de som brukes i dag. Se oversikt i [Xilinx, UG454].
- Inngangene + og – på operasjonsforsterkerne for båndpassfilteret og ved DA-omformeren var byttet om.
- Schottkydiode fra 3,3 V på inngangen til AD-omformeren var tegnet og montert feil vei.
-

Feilene er rettet manuelt på kretskortet, og alle komponentbeskrivelser, skjemategninger og kretskortutlegg er rettet.

# 3. Digital elektronikk

3.1 PROGRAMMERBAR LOGIKK ..... 42

3.2 HVA ER EN FPGA? ..... 46

*3.2.1 Oppslagstabell ..... 47*

*3.2.2 Arkitektur ..... 48*

*3.2.3 Andre elementer ..... 49*

3.2.3.1 MAC - DSP48A ..... 52

3.3 HVORFOR BRUKE EN FPGA? ..... 53

*3.3.1 FPGA vs. Mikrokontroller ..... 54*

3.4 FPGA FOR EKKOLODDPROSJEKTET ..... 56

*3.4.1 Valg av leverandør - Xilinx ..... 56*

*3.4.2 Valg av FPGA - Xilinx Spartan-3A DSP 1800 ..... 56*

3.4.2.1 Alternativer ..... 57

*3.4.3 Valg av utviklingskort - Spartan-3A DSP Starter Platform ..... 58*

3.5 PROGRAMVARE FRA XILINX ..... 60

*3.5.1 Xilinx (ISE) ..... 61*

*3.5.2 Xilinx Embedded Development Kit (EDK) ..... 63*

3.5.2.1 Xilinx Platform Studio (XPS) ..... 63

3.5.2.2 Software Development Kit (SDK) ..... 65

3.5.2.3 Andre Xilinx-programmer ..... 65

*3.5.3 Erfaringer med Xilinx-programvaren ..... 66*

3.5.3.1 Tips for problemløsning ..... 67

3.6 MASKINVAREBESKRIVENDE SPRÅK ..... 69

*3.6.1 Navneregler for VHDL ..... 70*

3.6.1.1 Sammenkobling av enheter ..... 70

3.7 SYSTEMOVERSIKT ..... 72

3.8 ENHETER ..... 73

*3.8.1 Grensesnitt mot AD-omformer – AD_LTC2356.vhdl ..... 75*

3.8.1.1 SPI-grensesnittet ..... 75

3.8.1.2 Timing ..... 75

3.8.1.3 Samplings- og konverteringsfrekvens ..... 76

*3.8.2 Kryssing av klokkedomener – cross_clock_domain.vhdl ..... 80*

*3.8.3 Uthenting av omhyllingskurven - envelope.vhdl ..... 83*

3.8.3.1 Problem med dopplereffekt? ..... 87

*3.8.4 Multiplexer - mux_4_to_1.vhdl ..... 88*

*3.8.5 Velge ut data – sample_data.vhdl ..... 89*

*3.8.6 Sette sammen sekvensielle data - concatenate_sequential_data.vhdl ..... 91*

*3.8.7 Skrive data til minneområde - RAM_RW_sequence.vhdl ..... 94*

*3.8.8 Minneområde - BRAM ..... 95*

*3.8.9 Pulskontroll – delay_and_hold.vhdl ..... 98*

*3.8.10 Grensesnitt mot DA-omformer - DAC_AD55X6.vhdl ..... 99*

3.9 MODULER I XILINX PLATFORM STUDIO ..... 101

*3.9.1 Prosessor - MicroBlaze ..... 102*

*3.9.2 BRAM-kontroller – lmb_bram_if_cntrl ..... 104*

*3.9.3 Avbruddskontroller – xps_intc ..... 105*

*3.9.4 Timer – xps_tmr ..... 106*

*3.9.5 UART – xps_uartlite ..... 107*

*3.9.6 Ethernet MAC – xps_ethernetlite ..... 109*

3.9.6.1 Ethernet ..... 109

3.9.6.2 PHY ..... 109

3.9.6.3 Ethernet MAC ..... 110

3.9.6.4 XPS 10/100 Ethernet MAC Lite ..... 111

*3.9.7 DDR-RAM grensesnitt - mpmc ..... 112*

3.9.7.1 MPMC ..... 112

3.9.7.2 Hurtigminne ..... 113

*3.9.8 Sanntidsklokke – real_time_clock.vhdl ..... 114*

*3.9.9 Grensesnitt mot prosessoren – module_interface.vhdl ..... 116*

*3.9.10 Øvrige moduler i XPS ..... 117*

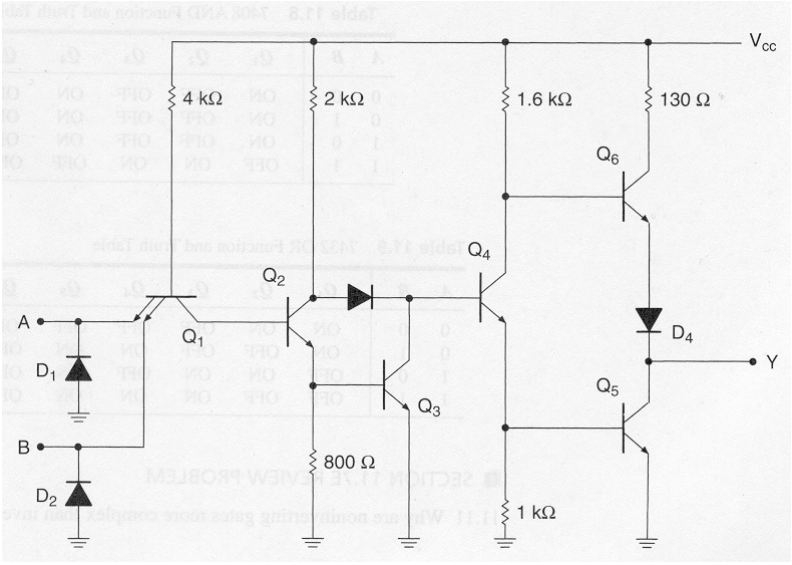
## 3.1 Programmerbar logikk

Fordi FPGA-teknologi er ukjent for mange potensielle lesere av denne oppgaven, har vi valgt å starte med en grundig introduksjon til emnet. Vi trekker linjene tilbake til de første digitale logiske kretsene for å gradvis forklare kompleksiteten i takt med utviklingen. Lesere som er kjent med FPGA-teknologi kan trygt hoppe til kapittel 3.3.

I 1947 ble transistoren oppfunnet ved Bell Laboratorium i USA. Dette danner grunnlaget for de digitale logiske kretsene vi har i dag. Ved siden av å kunne brukes som forsterkerelementer, kan man lage logiske funksjoner med dem. I starten brukte man diskrete komponenter for å bygge de logiske funksjonene man ønsket. Etter hvert kom ideen om å sette sammen hele kretser i én komponent, og på slutten av 50-tallet så de første integrerte kretsene dagens lys. [Maxfield, 2004, s. 25-26]

De første kretsene var basert på bipolare transistorer (BJT) og satt sammen til TTL eller ECL logikk. I 1962 ble MOSFET-transistoren (metal-oxide semiconductor field-effect transistors) oppfunnet. MOSFET-transistorene brukes i CMOS-kretser, som er det mest brukte i dag. Den viktigste fordelen med CMOS- framfor TTL-logikk er at CMOS bruker veldig lite strøm. [Maxfield, 2004, s.26-27]

På midten av 60-tallet introduserte Texas Instruments **74xx-serien** med logiske byggeblokker. Hver blokk inneholdt noen enkle logiske elementer (7400 – fire NAND-porter, 7402 – fire NOR-porter, 7404 – seks NOT-porter, osv.) som kunne settes sammen til større kretser. [Maxfield, 2004, s. 27]

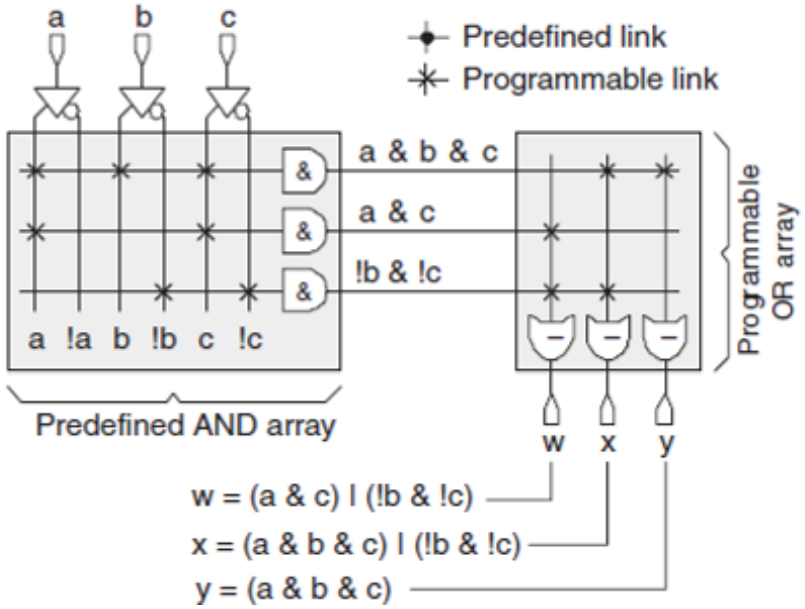



**Figur 3-1:** Kretsen i en 7408 AND port [Dueck, 2001, figure 11.43]

Etter hvert ønsket man å sette sammen flere elementer i større programmerbare kretser. Kretsene ble programmerbare ved at bindeleddene internt mellom elementene kunne åpnes eller lukkes. Dette ble gjort ved metoder som fuse, anti-fuse, EPROM-transistorer og EEPROM-celler [Maxfield, 2004, s. 9-23].

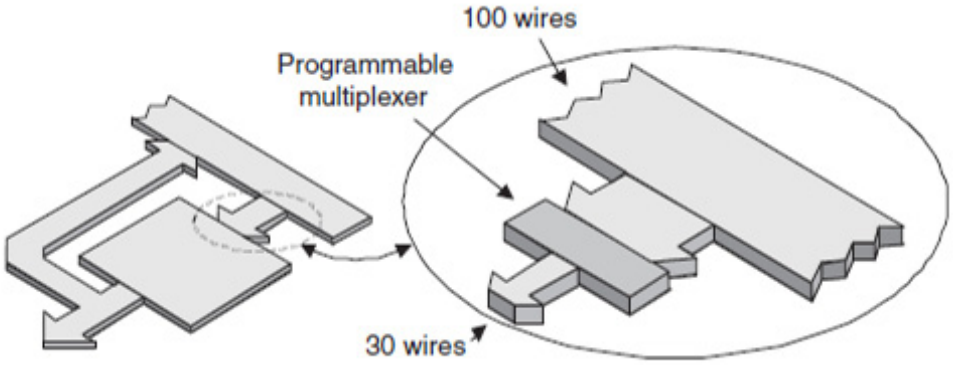
Den første programmerbare logiske kretsen (PLD – Programmable Logic Device) var **PROM** (Programmable Read-Only Memory) som ble introdusert i 1970. Disse bestod av en fast tabell med OG-funksjoner som kunne leses ut gjennom en programmerbar tabell med ELLER-funksjoner. Relativt komplekse logiske funksjoner, som tidligere var implementert med mange 74xx-kretser kunne implementeres i én PLD. Dermed fikk man mindre og billigere kretskort. [Maxfield, 2004, s. 30-33]

En av de store ulempene med PROM var at utgangene på kretsene var avhengig av alle inngangene. I praksis måtte man derfor bruke få innganger på hver krets. På midten av 70-tallet ble derfor **PLA** (Programmable Logic Array) introdusert, der både OG- og ELLER-tabellen var programmerbar. Ulempen med disse er at signalet bruker forholdsvis lang tid gjennom de programmerbare koblingene. Dermed ble **PAL** (Programmable Array Logic) og **GAL** (Generic Array Logic) som er omtrent det motsatte av en PROM (fast ELLER-tabell og programmerbar OG-tabell) introdusert på slutten av 70-tallet.




**Figur 3-2** En programmert PLA. [Maxfield, 2004, figure 3-5].

På begynnelsen av 80-tallet kom de første **CPLD**-ene (Complex PLD). Altera var de første som virkelig lyktes med dette. Dette er integrerte kretser der flere logikkblokker (ofte PAL) er sammenkoblet internt på brikken. Både logikkblokkene og sammenkoblingene er programmerbare [Maxfield, 2004, s. 37-39].




**Figur 3-3:** Programmerbare sammenkoblinger i FPGA-en [Maxfield, 2004, figure 3-9]

Parallelt med disse programmerbare kretsene har vi hatt **ASIC** (Application-Spesific Integrated Circuit) som er integrerte kretser som produseres etter beskrivelse fra firmaet som bestiller dem. ASIC-produsentene kan la kundene bestemme nøyaktig hvordan kretsen skal lages, eller de kan lage ferdig et lag med enheter som kobles sammen med et lag med ledningsbaner som kunden konfigurerer. Enhetene som kobles

sammen kan enten være transistorer og motstander, eller ferdiglagde moduler som multipleksere, porter, oppslagstabeller (LUT), flipp-flopper osv. ASIC-er gjør det mulig å lage store, avanserte og raske funksjoner. De er dyre og tidkrevende og designe, men ved store salgsvolum blir hver IC forholdsvis billig. Det er heller ikke mulig å endre dem etter at de er produsert. [Maxfield, 2004, s. 42-49]

I 1984 introduserte Xilinx den første **FPGA**-en (Field Programmable Gate Array) som er en kombinasjon av ASIC-ens avanserte funksjoner og CPLD-ens programmerbarhet.

[Maxfield, 2004, s. 49-53]

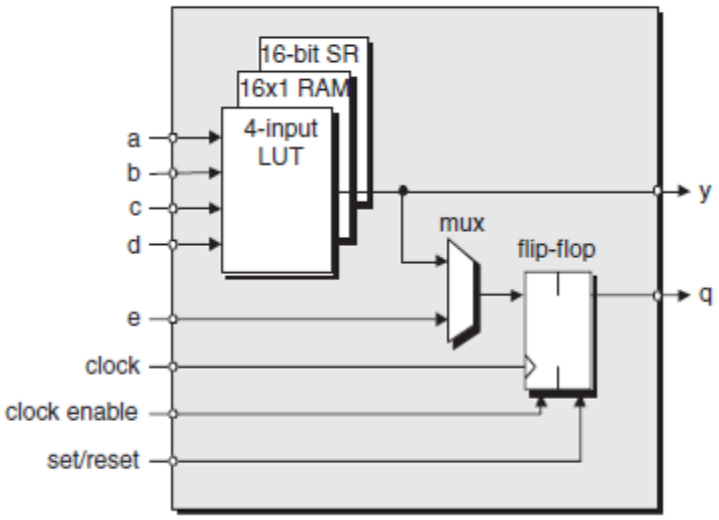
## 3.2 Hva er en FPGA?

FPGA står for *Field Programmable Gate Array.* Det er en integrert krets (IC) som kan programmeres til å inneholde digital logikk.

Det finnes flere firmaer som produserer FPGA-er, for eksempel Xilinx, Altera, Lattice Semiconductor, Actel og QuickLogic. Xilinx og Altera er de desidert største produsentene med 86 % markedsandel [Altera, 2008 Annual Report (Form 10-K)]. Av disse har Xilinx hatt 50 – 51 % av markedet de siste fem årene, mens Altera har hatt 32 – 35 % av markedet i samme periode [Xilinx, Xilinx Investor Factsheet Second Quarter Fiscal Year 2010].

Den fysiske oppbyggingen av kretsene varierer veldig fra produsent til produsent og innad blant de ulike modellene, men den logiske oppbyggingen av kretsen er derimot basert på de samme prinsippene. FPGA-er består av programmerbare logiske celler som er knyttet sammen ved hjelp av programmerbare sammenkoblinger. Den videre beskrivelsen tar utgangspunkt i oppbygningen av FPGA-er fra Xilinx, men den er stort sett lik oppbygningen Altera og andre produsenter bruker.

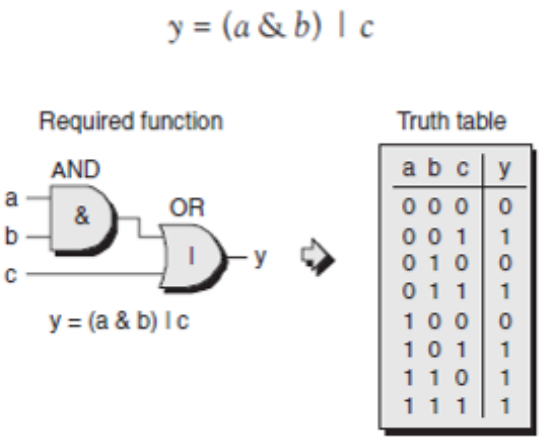
De minste byggeblokkene er logikkceller (Xilinx: Logic Cell, Altera: Logic Element). Disse består av en oppslagstabell (LUT - Look-Up Table), en multiplekser og et register. En meget forenklet framstilling av en slik logikkcelle er vist i figur 3-4.




**Figur 3-4:** Forenklet skisse av innholdet i en oppslagstabell [Maxfield, 2004, figure 4-7]

### 3.2.1 Oppslagstabell

Oppslagstabellen (LUT - Look-Up Table) har oftest fire innganger, og basert på verdien av disse settes utgangen. En hvilken som helst kombinatorisk funksjon med inntil fire innganger kan programmeres til oppslagstabellen. Fordi oppslagstabellen er implementert som SRAM, kan den også brukes som små minneblokker (f.eks. 16 x 1 RAM), som gjerne kalles distribuert RAM. Ved å bruke flere oppslagstabeller fra flere celler, kan man lage større minneblokker. Oppslagstabellene kan også brukes som skiftregistre (for eksempel 16-bit skiftregister) ved at alle minnecellene i oppslagstabellen kobles sammen i en kjede. [Maxfield, 2004, s. 69-75]

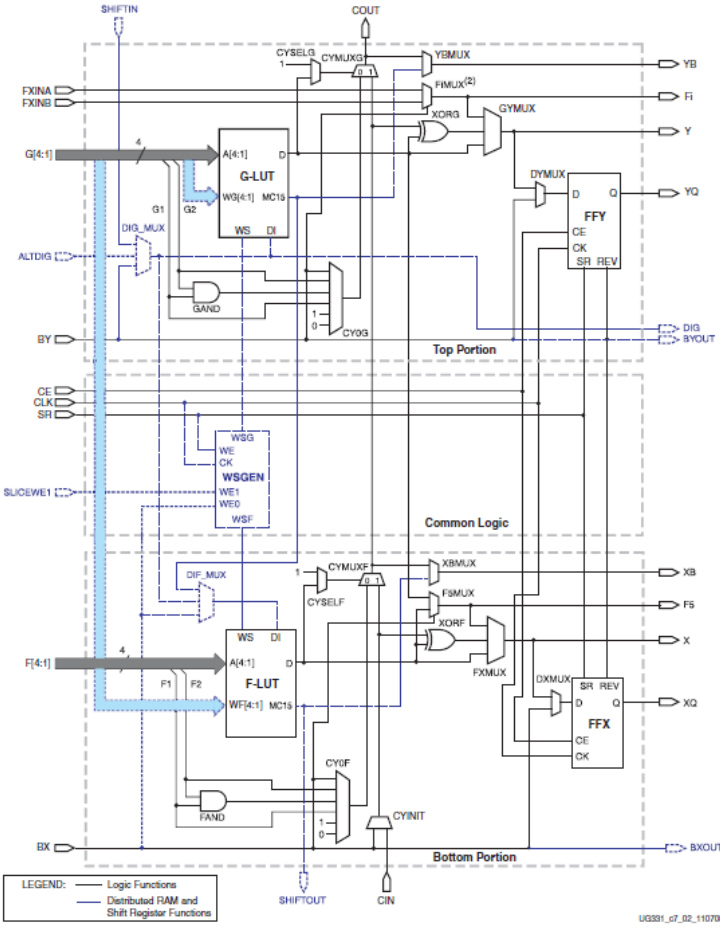



**Figur 3-5:** Innholdet i en oppslagstabell etter programmering [Maxfield, 2004, figure 4-3]

Hver logisk celle kan brukes som et minneelement, et skiftregister, en multiplekser, et register eller til å utføre en kombinatorisk funksjon eller en kombinasjon av disse. Når FPGA-en programmeres, fylles oppslagstabellene med verdier. Multiplekseren og registeret konfigureres, og polariteten til klokkesignaler (positiv eller negativ flanke) og nullstillingssignalet (aktiv høy eller lav) settes.

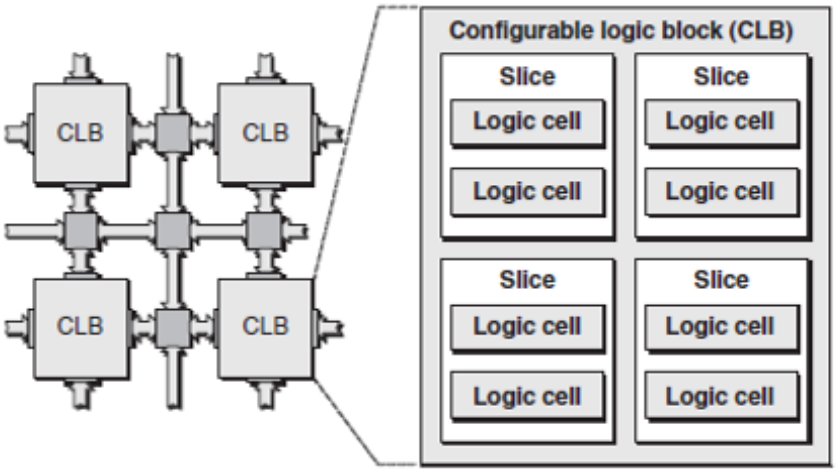
### 3.2.2 Arkitektur

Xilinx setter to logiske celler sammen til det de kaller en **slice**. Hver slice deler blant annet klokkesignal, klokke-aktivering (CE – clock enable) og nullstillingssignal. En forenklet framstilling av hvordan dette er implementert i Xilinx Spartan 3-FPGA-er er vist i figuren under:




**Figur 3-6** Forenklet framstilling av en slice i en Xilinx Spartan-3 FPGA. [Xilinx, UG331, s. 154]

Flere slicer settes sammen til en **konfigurerbar logisk blokk** (Xilinx: CLB – Configurable Logic Block, Altera: LAB – Logic Array Block). Disse kobles igjen sammen ved hjelp av programmerbare sammenkoblinger. Ved å settes sammen de logiske cellene i slicer og CLB-er, kan man ha raske sammenkoblinger mellom CLB-ene, raskere sammenkoblinger internt i CLB-ene, og enda raskere internt i slicene, samtidig som det er enkelt å koble logikken sammen i en helhet. Hvis alle de logiske elementene skulle kunne kobles direkte sammen, ville man fått trege sammenkoblinger mellom alle elementene. [Maxfield, 2004, s. 75-77]




**Figur 3-7:** Den interne arkitekturen i en FPGA [Maxfield, 2004, figure 4-9]

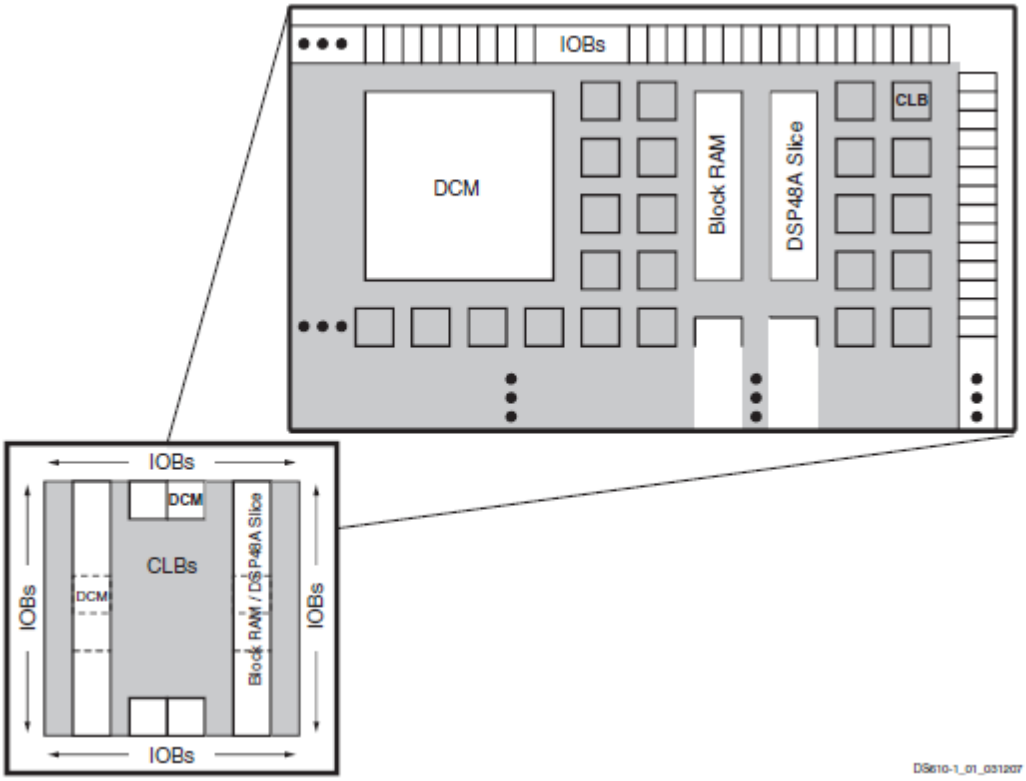
### 3.2.3 Andre elementer

I tillegg til de konfigurerbare logiske blokkene legges det ofte inn noen større elementer som for eksempel:

- Klokketrær (eng: clock tree) er spesiallagde signalbaner for å kunne overføre klokkesignaler over deler eller hele kretsen uten at signalene blir for mye forsinket i forhold til hverandre i de ulike delene av kretsen. Klokketrærene er koblet til egne pinner på FPGA-en som brukes som klokkeinnganger. [Maxfield, 2004, s.84-85]
- Klokkekontrollere (Xilinx: DCM – Digital Clock Manager) som brukes til å generere nye klokkesignaler basert på ett klokkesignal. Hvilke funksjoner som er implementert varierer, men de kan ofte justere for ujevnheter i klokkesignalet, generere klokker med annen frekvens, legge til faseskift, eller kombinasjoner av disse. [Maxfield, 2004, s. 85-89]

- Minneområder (Xilinx: BRAM - block RAM) som gir mulighet til å lagre og lese større mengder data enn det er mulig med LUT-er konfigurert som distribuert RAM. Minneområdene kan brukes som RAM, FIFO (First In – First Out), tilstandsmaskiner ol. [Maxfield, 2004, s.78-79]
- Multiplikatorer, adderingsenheter og MAC-er (Multiply-and-Accumulate) som er veldig nyttig ved digital signalprosessering, og opererer mye raskere enn det ville vært mulig å gjøre ved å implementere funksjonene med logiske celler. [Maxfield, 2004, s.79-80]
- Prosessorkjerner som kan programmeres på samme måte som andre prosessorer, men som har direkte tilkobling til FPGA-ens indre signalbaner. Antall prosessorkjerner som implementeres varierer fra FPGA til FPGA. Prosessorkjerner implementert i samme silisiumbrikke som FPGA-en kalles gjerne *hard core,* i motsetning til prosessorkjerner i FPGA-ens logikk (*soft core)* (Se kapittel 3.9.1). [Maxfield, 2004, s.80-84]
- Høyhastighets transceiverkretser for å overføre store mengder data med høy hastighet til andre enheter. [Maxfield, 2004, s. 92-93]
- Konfigureringsenheter som for eksempel JTAG (Joint Test Action Group) som brukes til både å programmere FPGA-en, og for å lese ut verdier under feilsøking. [Maxfield, 2004, kapittel 5]

Fordi FPGA-er lages med ulik sammensetning av de logiske elementene og med ulike spesialfunksjoner er det ofte veldig vanskelig å sammenligne FPGA-er fra forskjellige leverandører.




**Figur 3-8** Oversikt over elementene i Xilinx Spartan 3A DSP [Xilinx, DS610, s. 5].

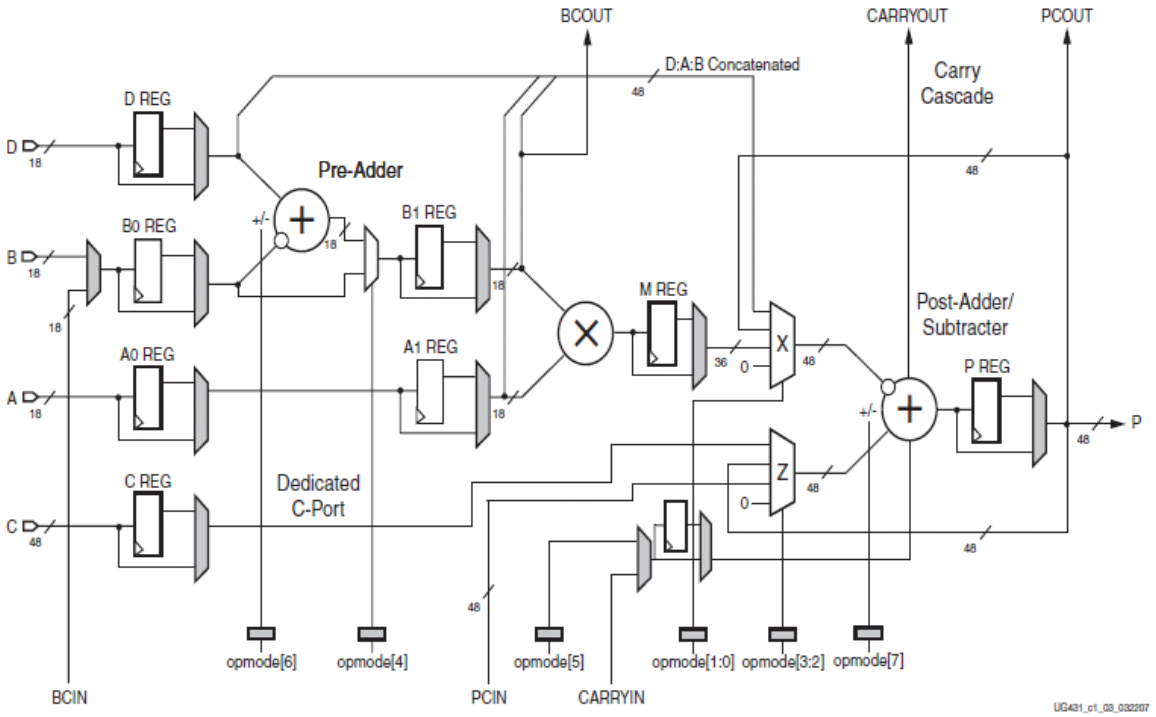
Innholdet i en Xilinx Spartan 3A DSP FPGA vises i figur 3-8. Forkortelsene som brukes har følgende betydning:

- CLB (Configurable Logic Block) – Konfigurerbare logiske blokker.
- Block RAM – Minneområder. Hvert område er på 18 kbit.
- IOB (Input/Output Blocks) – Styrer signalflyten mellom I/O-pinner og den interne logikken. De fleste I/O-pinnene er for generell bruk, men noen er spesielt satt av til klokkesignaler ol.
- DCM (Digital Clock Manager) – Klokkekontroller.
- DSP48A Slice (XtremeDSP™ DSP48A Slice) – Spesiallaget multiplikator, adderingsenhet og akkumulator for digital signalprosessering. Dette er den viktigste forskjellen mellom DSP-variantene og de øvrige Spartan 3A FPGA-ene.

### 3.2.3.1 MAC - DSP48A

En MAC-celle består, som navnet tilsier, av en multiplikator (B*C) og en akkumulator (A = A+B). Til sammen utfører de funksjonen A = A + B * C, som er veldig vanlig i digital signalbehandling, blant annet diskret fourier-transformasjon og digital filtrering [Wikipedia, Multiply-accumulate]. Det er fullt mulig å gjøre disse operasjonene i generelle FPGA-er også, men det vil kreve mange logikkblokker som vil medføre høyere strømforbruk, dårligere utnyttelse av brikkens kapasitet, og lavere hastighet [Xilinx, UG431].

I denne FPGA-en kalles MAC-cellene DSP48A, og de er komplekse enheter med mange anvendelser. Enkle funksjoner lages ved at man bruker enkeltelementer fra MAC-cellene (multiplikasjon, addisjon, subtraksjon, sammenligning, multipleksing), og ved å sette sammen flere MAC-celler kan man lage avanserte funksjoner som FIR-filtre og lignende [Xilinx, UG431].




**Figur 3-9** En DSP48A-enhet. [Xilinx, UG431]

## 3.3 Hvorfor bruke en FPGA?

Vi trengte en sentral enhet for å styre ekkoloddsystemet og ta seg av blant annet følgende oppgaver:

- Styre sending av ping eller registrere triggersignal fra kassettspilleren.
- Styre AD-omformeren og ta imot data fra denne.
- Hente ut omhyllingskurven basert på dataene fra AD-omformeren.
- Sende verdiene for omhyllingskurven over Ethernet til en PC for lagring og analysering.
- Motta kommandoer for konfigurering og kontroll via Ethernet.

Disse oppgavene kan løses med for eksempel en prosessor (CPU), en CPLD eller en FPGA. Prosessorer trenger flere komponenter rundt seg, som f.eks. RAM, ROM og diverse periferienheter. Derfor brukes gjerne mikrokontrollere som er IC-er bestående av en prosessorkjerne, RAM, ROM, timere, kommunikasjonsenheter og lignende på samme brikke.

Selv om det er fullt mulig å implementere komplette IP-stakker i hardware, som for eksempel [Adescom Polska, Wire-speed Internet] og [RealFast, UDP/IP cores], er det mye enklere å implementere dette for en prosessorkjerne. Utvalget blir også mye større. Det er derfor ønskelig at kontrollenheten har en prosessorkjerne til å ta seg av Ethernet-trafikk.

Det er mulig å implementere soft core-prosessorer på CPLD-er [Opencores, MCPU], men det er lite utbredt. FPGA-er kan derimot inneholde både hard core- og soft core-prosessorer. Derfor står valget i praksis mellom en FPGA eller en mikrokontroller.

### 3.3.1 FPGA vs. Mikrokontroller

- **Ytelse:** I en prosessor utføres oppgavene sekvensielt. I en FPGA derimot kan oppgavene utføres parallelt på flere datasett samtidig. Dermed kan man oppnå betydelig høyere ytelse med en FPGA enn med en prosessor til visse applikasjoner.
    - I dette prosjektet prosesseres samplingsverdiene etter at de er lest inn fra AD-omformeren, og før de lagres i minnet. Slik det er implementert i dag, bruker FPGA-en tre klokkeslag på å detektere om det er et av bærebølgens toppunkt som er samplet, avgjøre om verdien skal lagres og eventuelt lagre det til minneområdet.
    - Sekvensen ville sett veldig annerledes ut i et prosessorsystem, men den ville garantert brukt mer enn tre klokkeslag på prosessen.
    - Signalprosessering av ekkoloddsignaler er ikke spesielt krevende for maskinvaren, så det er nok først ved høyere hastigheter og store datamengder at man virkelig får utnyttet FPGA-ens potensial.
- **Konfigurerbarhet og fleksibilitet:** Arkitekturen i en mikrokontroller er gitt når man kjøper den. Man må betale for alt som følger med, selv om man kanskje ikke trenger det. Det logiske innholdet i en FPGA konfigureres derimot etter at den er kjøpt. Hvis man trenger én eller flere prosessorkjerner, ulike IO-enheter og busser, DSP-funksjonalitet og lignende, kan man legge det til ved behov. Den virkelig store fordelen er likevel at man kan lage egne logikkmoduler for å utføre spesielle oppgaver som ofte ellers ville tatt mye tid i en prosessorkjerne. Slik kan ny funksjonalitet legges til uten at man behøver å bytte til en kraftigere prosessor og endre kretskortutlegget.
- **Kompleksitet:** På grunn av fleksibiliteten i en FPGA kan man lage svært omfattende systemer, sammenlignet med en ferdiglaget mikrokontroller. Dette gir store fordeler, men det gjør også at terskelen for å ta i bruk en FPGA er betydelig høyere.
- **Strømforbruk:** FPGA-er bruker som oftest mer strøm enn mikrokontrollere, selv om strømforbruket stadig går ned. Høyt strømforbruk er en ulempe siden ekkoloddsystemet som brikken skal inngå i skal være mobilt. Da bilbatterier har blitt brukt som strømkilde for ekkoloddsystemene som er utviklet ved Fysisk institutt til nå, er strømforbruk ikke en avgjørende faktor.
- **Pålitelighet:** Mikrokontrollere og FPGA-er som selges på det kommersielle markedet er gjennomtestet før salg av produsenten. Innmaten på mikrokontrolleren (Programkode) og FPGA-en (maskinvarebeskrivelse og evt. programkode) må brukeren teste selv, noe som er en tidkrevende prosess. Fordi en FPGA inneholder maskinvarebeskrivelser i tillegg til programkode, er det også mer komplekst og tidkrevende å teste.
- **Pris:** Mikrokontrollere er som oftest mye billigere enn FPGA-er.
    - Prisen for en Xilinx Spartan 3A DSP 1800A FPGA hos Farnell var i desember 2009 ca. 600 kr ekskludert merverdiavgift, avhengig av antallet som kjøpes inn [Farnell, Xilinx Spartan 3A DSP 1800A]. I

tillegg trenger man minnebrikker, Ethernet PHY, RS232-brikke m.m. for å få et kjørbart system.
- Prisen for mikrokontrolleren Atmel ATXMEGA 128 som brukes mye på Fysisk institutt for tiden, var hos Farnell i desember 2009 ca. 50 kr ekskludert merverdiavgift, avhengig av antallet som kjøpes inn [Farnell, ATXMEGA 128].
- Siden dette systemet uansett ikke skal produseres i et stort antall, er komponentkostnadene av liten betydning. Komponentenes egenskaper og ytelse er mye viktigere faktorer ved innkjøp av komponenter.

Ragnar Holm og Geir Frode Raanes Sørensen ved Elektronikklaboratoriet på Fysisk institutt (ELAB) har den siste tiden forsøkt å sample bærebølgen direkte med en ATXMEGA 128. I skrivende stund fungerer algoritmen for uthenting av omhyllingskurven som den skal, men systemet er ikke raskt nok til hente ut omhyllingskurven av et 50 kHz signal.

Fordi ekkoloddsystemet skal brukes til forskning, er kanskje konfigurerbarhet og ytelse de viktigste faktorene. Det gir mulighet til å gjøre mange forskjellige undersøkelser basert på samme system. Innmaten i FPGA-en kan enkelt endres eller erstattes, og nye IP-moduler kan legges til, alt etter hva systemet skal brukes til. Man kan derfor basere seg på ett system i stedet for lage nytt design for hvert prosjekt som skal gjennomføres.

Et mikrokontrollersystem er nok tilstrekkelig for å utføre oppgavene i dette prosjektet, men det gir færre muligheter for utvidelser i framtiden, så vi mener at det kan være fornuftig å velge en FPGA til dette systemet. Allerede før prosjektet ble startet, ble det bestemt å bruke en FPGA i masteroppgaven fordi vi ønsket å heve vår kompetanse på området. Det tas noen andre hensyn når systemutviklingen skjer i forbindelse med utdanning.

## 3.4 FPGA for ekkoloddprosjektet

### 3.4.1 Valg av leverandør - Xilinx

Som nevnt i kapittel 3.2, er det flere firmaer som produserer FPGA-er, men vi har valgt å bruke en FPGA fra Xilinx. FPGA-er fra Xilinx brukes allerede ved Universitet i Oslo i blant annet faget "Digital systemkonstruksjon" (INF3430/4430), ved Elektronikklaboratoriet ved Fysisk institutt og i forskningsgruppen ROBIN (Robotics and intelligent systems) ved Institutt for informatikk. I tillegg er Xilinx markedsleder på FPGA-er [Xilinx, Xilinx Investor Factsheet Second Quarter Fiscal Year 2010].

### 3.4.2 Valg av FPGA - Xilinx Spartan-3A DSP 1800

Xilinx lager to familier med FPGA-er, Spartan og Virtex. Spartan er best på pris, mens Virtex har høyest ytelse. For vårt prosjekt var det tilstrekkelig med en FPGA fra Spartan-familien.

FPGA-en som benyttes ble valgt helt i starten av prosjektet. Vi valgte en større FPGA enn det vi regnet med at den endelige kretsen ville kreve for å være sikre på å ha nødvendig kapasitet til det vi skulle implementere, og ha mulighet til å teste ulike løsninger. Det er ledig plass til fremtidige utvidelser av designet, slik at man kan bygge videre på det arbeidet som er gjort.

Den valgte FPGA-en er laget spesielt for digital signalprosessering (DSP), noe som er nyttig i sonaranvendelser. Det for eksempel enkelt å implementere digitale filtre og lignende. Det som skiller denne FPGA-en fra generelle FPGA-er, er at den inneholder mer og raskere minne (BRAM) og spesiallagde multipliser-og-akkumuler-celler (MAC) [Xilinx, DS610].

Xilinx lager to Spartan-FPGA-er med DSP-funksjonalitet, Xilinx Spartan-3A DSP 1800A og Xilinx Spartan-3A DSP 3400A. Også her valgte vi den enkleste og billigste utgaven, da det gav oss mer enn nok ytelse. FPGA-en vi valgte er en Xilinx Spartan-3A DSP 1800-FG676 montert på et utviklingskort (Starter Platform).

#### 3.4.2.1 Alternativer

| | Implementert | Spartan 3A DSP 1800A (XC3SD1800A) | Utnyttelsesgrad |
|---|---|---|---|
| **IO-pinner** | ca. 125 | 519 | 23 % |
| **Registre** | ca. 9000 | 33280 | 26 % |
| **Look-up tables (LUT)** | ca. 13500 | 33280 | 40 % |
| **Slicer** | ca. 10500 | 16640 | 61 % |
| **Blokk RAM (á 18 kbit)** | ca. 55 | 84 | 65 % |
| **DSP48A-blokker** | ca. 10, men disse kunne vært vanlige Spartan-multiplikatorer. | 84 | 9 % |

**Tabell 3-1** Oversikt over utnyttelsesgraden av FPGA-en i dette prosjektet

I tabell 3-1 vises hvor mye av FPGA-en som har blitt brukt i prosjektet. Tallene i kolonnen ”Implementert” er omtrentlige. De kan variere for hver gang prosjektet syntetiseres, avhengig av hvordan det optimaliseres.

Over 60 % av slicene er i bruk, men dette kan nok reduseres noe ved å optimalisere prosessorsystemet i EDK, skrive VHDL-kode som er mer optimalisert for denne FPGA-arkitekturen, og ved å sette Xilinx ISE til å optimalisere designet med hensyn til størrelse i stedet for hastighet. Antall blokk-RAM som brukes kan også reduseres ved å flytte noe av MicroBlaze-prosessorens minne fra blokk-RAM og over i DDR2 SDRAM.

Hvis man reduserer antall blokk-RAM som brukes, vil det trolig være mulig å implementere designet i Xilinx Spartan 3A 1400A [Xilinx, Extended Spartan-3A Family] som er hakket mindre og billigere (375 kr hos Farnell i desember 2009 (http://no.farnell.com/xilinx/xc3s1400a-4ftg256c/fpga-spartan-3a-1400k-ele-256ftbga/dp/1671096)). Prisdifferansen er ca. 200 kr + mva, så det er en ubetydelig forskjell når kapasiteten blir mindre og man mister DSP-funksjonaliteten.

De nyeste ”billig-FPGA-ene” fra Xilinx, Spartan 6 [Xilinx, Spartan-6 Family], har mye bedre ytelse enn forrige generasjon, og alle har DSP-funksjonalitet. Man får mer ytelse for pengene med disse, og slik vil det nok bare fortsette. Arkitekturen er endret til LUT-er med 6 innganger, og flere LUT-er og flip-flop-er pr slice, så de kan ikke sammenlignes direkte med Spartan 3, men må eventuelt testsyntetiseres i ISE 11.

### 3.4.3 Valg av utviklingskort - Spartan-3A DSP Starter Platform

Å lage utlegg for FPGA-er tar mye tid og er veldig krevende. Slike høyhastighetskomponenter er følsomme for strøinduktanser og signalenes tidskrav er kritiske. IC-brikken har mange bein (676 for denne FPGA-en), så det vil kreve mange lag for å rute ut beina som trengs. Utviklingskortet vi bruker har et 12-lags kretskort [Xilinx, UG454, s. 33], men det skal være mulig å bruke 4-6 lag for å rute ut FPGA-en [Xilinx, UG331, s. 457].

For å utelukke feil ved PCB-utlegg, og for å få en rask oppstart av prosjektet, var det naturlig å starte med et utviklingskort. Mange bedrifter velger å selge produktene sine med utviklingskort montert i produktet den første tiden. Vi har også valgt å ikke utvikle et eget kretskort for FPGA-en, da det ville være både tidkrevende og dyrt i forhold til hvor mange kretskort som eventuelt ville blitt produsert.

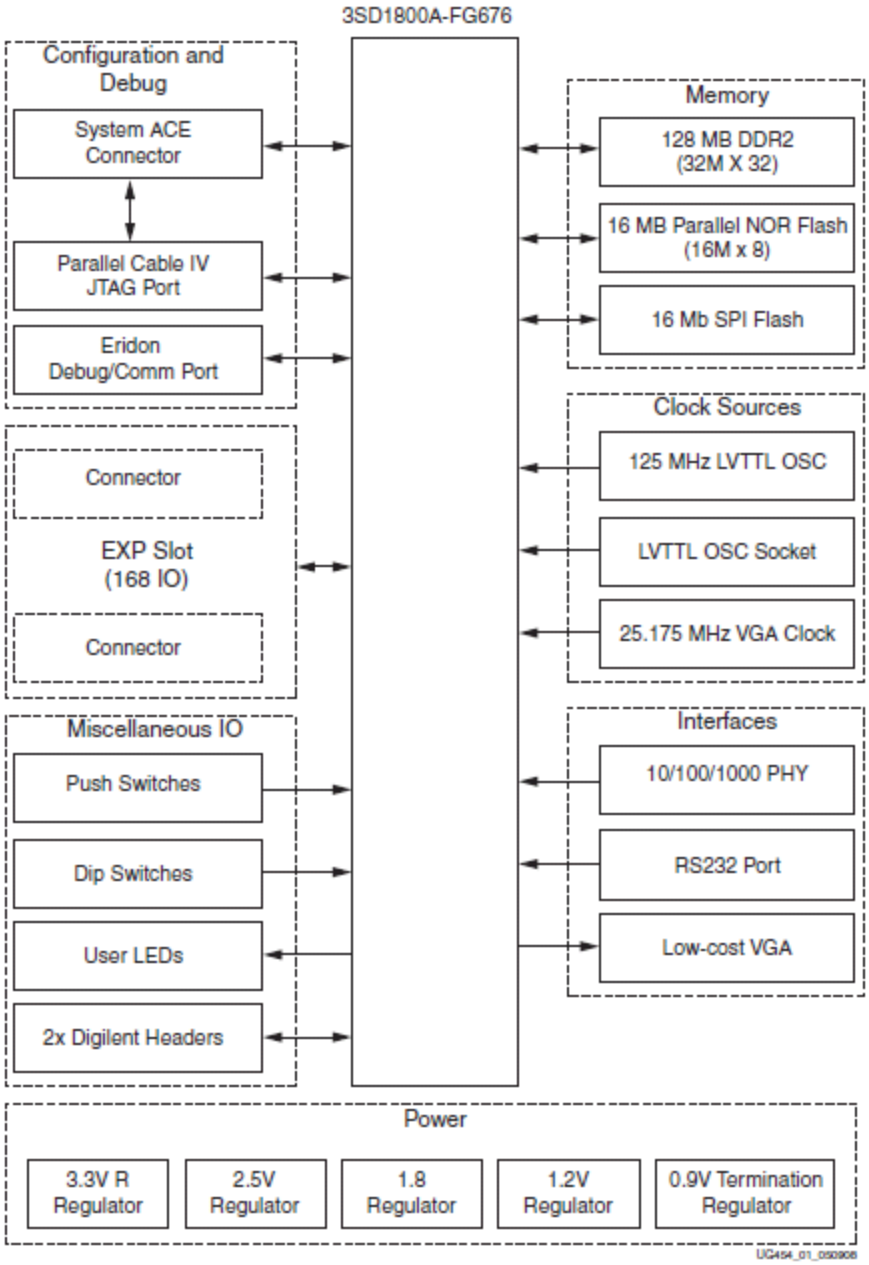
Da utviklingskortet ble kjøpt inn, fantes det kun ett utviklingskort med en Xilinx Spartan DSP-FPGA (det har kommet flere senere). Det heter Spartan-3A DSP Starter Platform (HW-SD1800A-DSP-SB-UNI-G).

I tillegg til FPGA-en og nødvendig elektronikk rundt den, inneholder kortet blant annet [Xilinx, UG454]:

- Eksterne minnebrikker
    - DDR2 SDRAM for midlertidig lagring av data.
    - Flash-minne (SPI og parallell) for konfigurering og lagring av data.
- Ulike grensesnitt
    - Ethernet PHY (10/100/1000 Mbit/s)
    - JTAG-port for programmering av FPGA-en.
    - RS232-port for seriell kommunikasjon.

- VGA-kontakt for tilkobling av skjerm.
- Ekspansjonskontakter for I/O (Digilent og EXP) for tilkobling til andre enheter og kretskort.

- Brytere, knapper og lysdioder til valgfrie anvendelser.




**Figur 3-10** Oversikt over innholdet på Spartan-3A DSP Starter Platform [Xilinx, UG454]

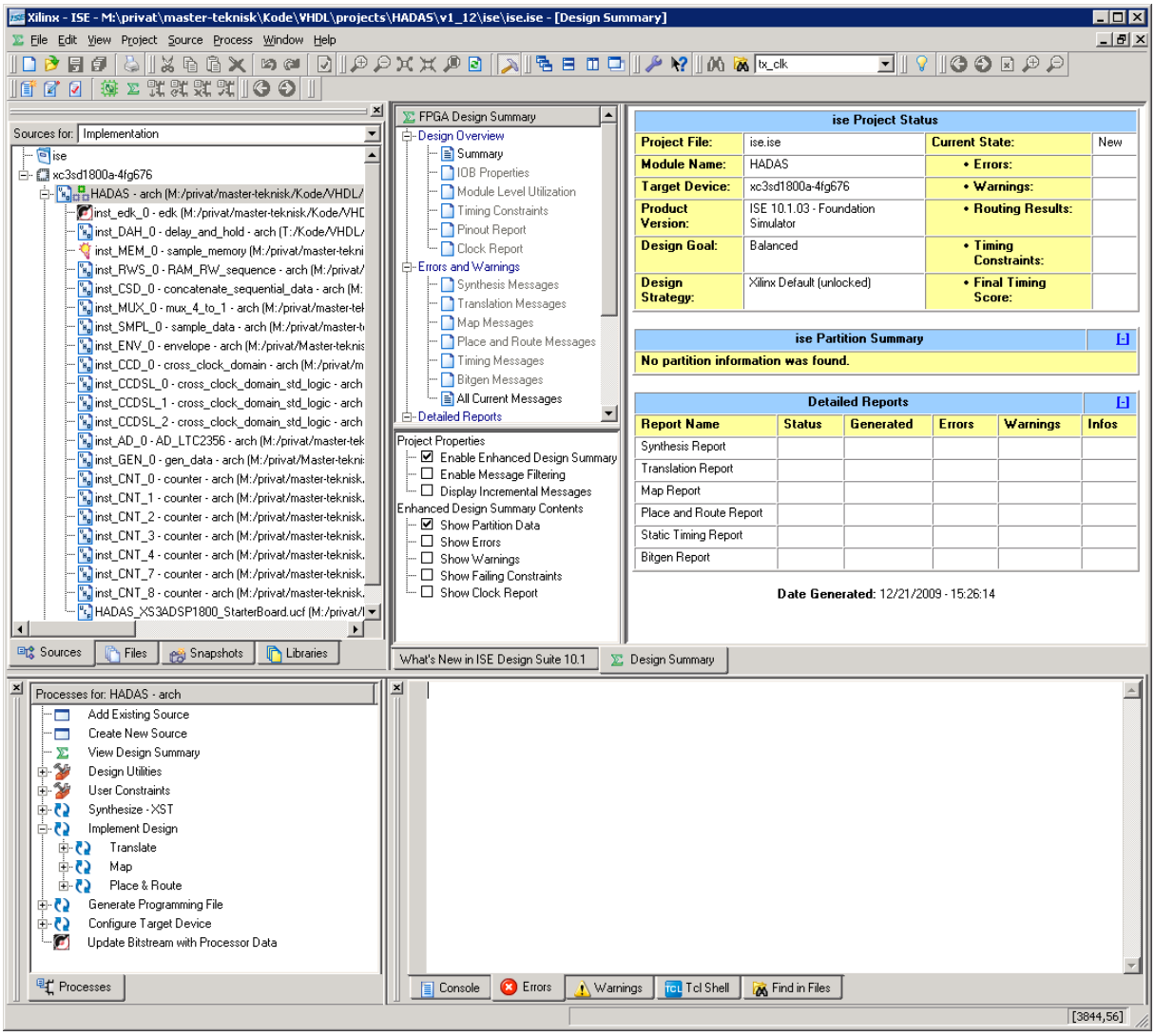
Fordi utviklingskortet har en ekstern DDR2 minnebrikke der både data og programinstruksjoner kan lagres, er det også godt egnet for å kjøre større applikasjoner og operativsystemer på en MicroBlaze-prosessor i FPGA-en. Dette gir utvikleren stor frihet til å teste ut ulike løsningsmetoder.

## 3.5 Programvare fra Xilinx

Man trenger flere programmer for å konfigurere FPGA-en. Disse må være tilpasset den FPGA-en man bruker. Xilinx tilbyr ISE WebPACK Design Software [Xilinx, ISE WebPACK] for syntese og implementering av digital logikk gratis, men det er begrenset hvilke FPGA-er den kan syntetisere til [Xilinx, Design Tools]. Hvis man skal bruke MicroBlaze-prosessoren og modulene rundt denne, må man kjøpe fullverdige versjoner av Xilinx ISE og EDK.

Universitetet i Oslo er medlem i Europractice, og har dermed tilgang til Xilinx-programvare til gunstige priser [Europractice, Xilinx]. Da prosjektet begynte, benyttet vi versjon 9.2 av programvaren, men vi fikk etter hvert oppdatert til versjon 10.1. I mellomtiden har Xilinx lansert versjon 11.3, og Fysisk institutt planlegger å oppgradere til denne versjonen i nærmeste framtid.

## 3.5.1 Xilinx (ISE)




**Figur 3-11** Skjermbilde av Xilinx ISE

ISE er et komplett verktøy for design av digitale logiske kretser. Man kan opprette og endre VHDL/Verilog-filer eller bruke innebygde programmer for å tegne skjemategninger, designe tilstandsmaskiner (State Diagram) eller legg til spesielle elementer som for eksempel minneblokker (CORE Generator & Architecture Wizard) og prosessorkjerner (EDK (ikke inkludert i ISE)).

Når den logiske kretsen er ferdig beskrevet, blir den syntetisert til nettlister, som er en fysisk beskrivelse av kretsen. Nettlister inneholder beskrivelser av logiske elementer og sammenkoblingene mellom disse. Den logiske kretsen implementeres deretter i henhold til den valgte FPGA-arkitekturen.

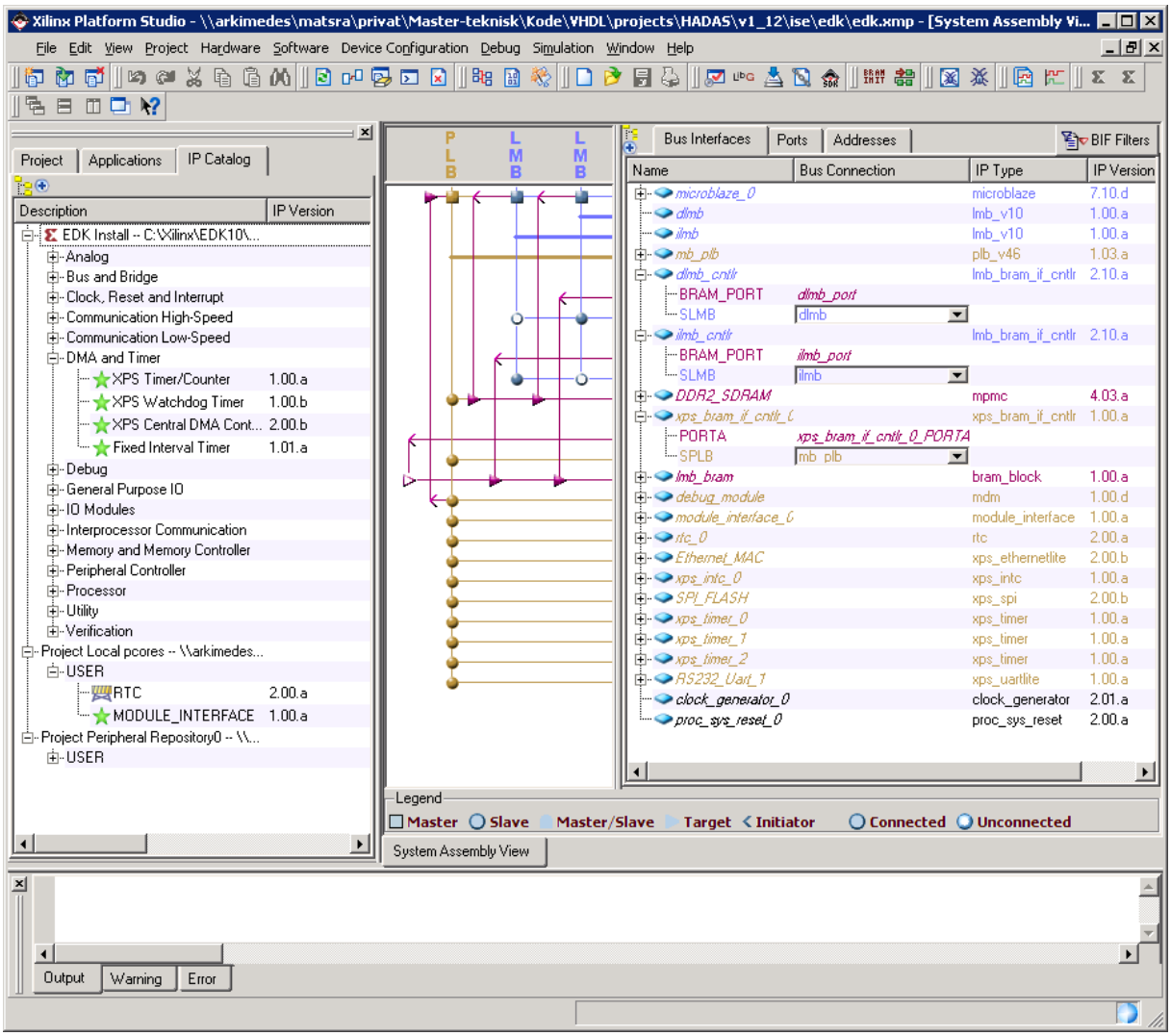
Ved å endre rammebetingelsene den logiske kretsen skal syntetiseres og implementeres etter (eng: constraints), kan den samme logiske kretsbeskrivelsen (VHDL) implementeres forskjellig med hensyn til pinneplassering, strømforbruk, ressursforbruk, klokkehastighet og lignende. Rammebetingelsene skrives som tekst i VHDL-koden (attributes) eller en egen fil (Xilinx: User Constraints File), og spesifiseres i syntheseverktøyet.

Kretsen verifiseres mot rammebetingelsene. Hvis programmet klarer å implementere den logiske kretsen, lages det til slutt en fil som kan brukes for å programmere FPGA-en. På alle nivåer kan designet simuleres i den innebygde simulatoren eller i eksterne simuleringsprogrammer som for eksempel ModelSim [Mentor Graphics, ModelSim].

## 3.5.2 Xilinx Embedded Development Kit (EDK)

EDK er en pakke med programmer for å kunne bruke PowerPC-prosessoren som er implementert i noen Virtex-FPGA-er, eller implementere soft-core-prosessoren MicroBlaze på alle Xilinx-FPGA-er. Lisens for MicroBlaze-prosessoren (se kapittel 3.9.1) er den del av pakken.

### *3.5.2.1 Xilinx Platform Studio (XPS)*



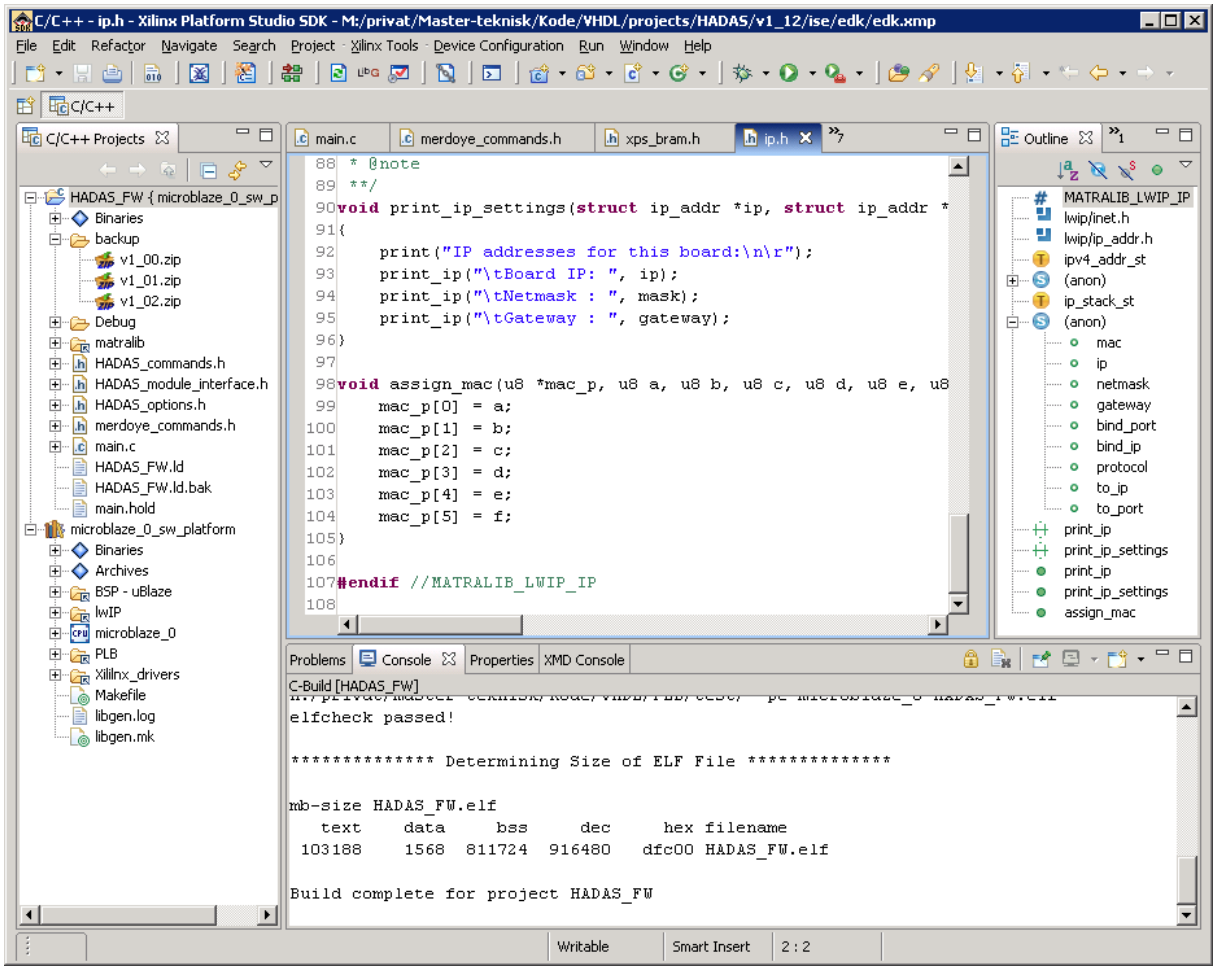

**Figur 3-12** Skjermbilde av Xilinx Platform Studio (XPS)

XPS er en databuss-sentrert framstilling av prosessorkjerner og andre moduler som kan integreres på en FPGA. Det gir et grafisk brukergrensesnitt for å opprette og konfigurere prosessorsystemer.

Prosessorsystemene kan enten bygges opp manuelt ved å legge til moduler og konfigurere og koble sammen disse, eller man kan benytte programmet Base System Builder (BSB) som veileder brukeren gjennom designprosessen.

I dette prosjektet har vi brukt Base System Builder for å generere en basisplattform for systemet med prosessorkjerne og grensesnitt mot DDR2-RAM, RS-232, Ethernet PHY og lignende. Fordi programmet kjenner testkortet vi har benyttet, sørget det for å rute signalene til riktige pinner og opprettet andre nødvendige parametre. Deretter har vi lagt til flere moduler som timere og selvlage enheter manuelt og konfigurert de eksisterende, slik at systemet passer til oppgaven som skal løses.

### 3.5.2.2 Software Development Kit (SDK)




**Figur 3-13** Skjermbilde av Xilinx Software Development Kit

SDK er basert på programvareutviklingsverktøyet Eclipse [The Eclipse Foundation, Eclipse] og utvidet med tilleggsmoduler fra Xilinx. Programmet brukes til å programmere, feilsøke og kompilere c-kode for integrerte prosessorer på FPGA-en.

### 3.5.2.3 Andre Xilinx-programmer

Andre programmer fra Xilinx som også kan være nyttige i utviklingsprosessen er

- **ChipScope** [Xilinx, ChipScope] brukes til å overvåke interne signaler på FPGA-en for sanntidsverifikasjon av FPGA-design.
- **System Generator** [Xilinx, System Generator] kan generere syntetiserbar kode fra algoritmer for digital signalbehandling som er modulert i det grafiske verktøyet SIMULINK for MATLAB[Mathworks, MATLAB & SIMULINK].

### 3.5.3 Erfaringer med Xilinx-programvaren

Programvaren fra Xilinx er spesiallaget for å konfigurere deres egne FPGA-er, men dessverre har vi opplevd en rekke problemer i løpet av prosjektet. Programmene er store og komplekse. ISE består av ca 130 000 filer og bruker ca 8 GB på harddisken. EDK består av ca 27 000 filer og bruker ca. 2 GB. Dette inkluderer flere biblioteksfiler for syntese, implementering og programvare. Dessverre er det noen av filene som inneholder feil som kan gjøre at programmet bråstopper, eller at uforutsette ting skjer.

Det som særlig har vist seg å være sårbart, er sammenkoblingen mellom ISE, XPS og SDK. Slik prosjektet er satt opp, er ISE basis for hele prosjektet, mens XPS brukes til å konfigurere prosessorsystemet. I tillegg brukes SDK for å programmere prosessorsystemet. Alle programmene er derfor avhengig av informasjon fra de andre programmene. De leser filer som de andre programmene genererer. Derfor er det innenfor en prosjektkatalog flere filer med samme navn og innehold. Dessverre skjer det litt for ofte at et av programmene utelater å oppdatere én av filene. Dermed har de andre programmene feil informasjon. Dette kan føre til samme type feilmeldinger som når det er noe galt med det brukeren har gjort.

I tillegg til filene som opprettes for at programmene skal kommunisere med hverandre, genereres det en rekke andre filer i prosjektmappen for at prosesser som kjøres skal gå raskere neste gang. Et ny prosjektmappe med komplett system i ISE og EDK inneholder 230 filer og bruker ca 2 MB på harddisken (VHDL-, c- og h-filer vi har utviklet er ikke medregnet). Etter at systemet er syntetisert inneholder mappen over 3500 filer og bruker over 400 MB på harddisken.

Når man gjør endringer på systemet, er det ikke alltid programmet oppdaterer alle filene. Den bruker filer som er generert tidligere og ikke inneholder endringene man har gjort. Dermed kan man prøve å endre filene i dagevis uten å få det til å fungere, fordi en gammel versjon brukes hver gang man tester det.

Fordi filer med samme navn og innhold finnes flere steder, gjelder det å redigere på riktig fil. For eksempel finnes h-filer for drivere man har laget til egne EDK-moduler

på tre forkjellige steder. Hele modulen ligger i mappen ”pcores”. Det er dette som er den originale filen man må endre på hvis endringen skal tas vare på. EDK kopierer driverfilene til ”SDK_projects\microblaze_0_sw_platform\microblaze_0\libsrc” for å kompilere dem, mens h-filene også legges i ”SDK_projects\microblaze_0_sw_platform\microblaze_0\include” som er tilgjengelig fra SDK. Endringer på den sistnevnte filen gir endringer ved kompilering, men hvis man velger ”Generate Libraries and BSPs” i SDK, skrives filene over av den som ligger i ”pcores”. Altså er det filen i ”pcores” som må endres!

Xilinx har nå kommet med versjon 11.3 av ISE og EDK. I denne versjonen er grensesnittene mellom de forskjellige programmene endret. De har også rettet opp mange feil fra de tidligere versjonene. Vi har dessverre ikke hatt anledning til å teste denne versjonen, men beskrivelsene virker lovende.

#### *3.5.3.1 Tips for problemløsning*

I løpet av prosjektet har til sammen mer enn én måned gått med til feilsøking og gjenoppbygging av prosjekter etter feil i programvaren fra Xilinx - i realiteten enda flere dager fordi utviklingsprosessen har gått tregere. Årsaken til at det går så mye tid, er at feilene i programvaren resulterer i samme type feil som når brukeren har programmert noe feil. Fordi man ikke vet hva feilen skyldes, kan man bruke flere dager på å lete etter feil og prøve å rette på systemer og kode en selv har skrevet. Når man til slutt får mistanke om at feilen kan skyldes programvaren, og oppretter et nytt prosjekt, viser det seg at alt en selv har gjort har fungert hele tiden.

Feilmeldingene inneholder et feilnummer og en generell tekst. Ved å trykke på feilnummeret utføres et søk i feildatabasen til Xilinx på internett (også tilgjengelig fra [Xilinx, Answer Browser]), og resultatet vises direkte i programmet. Av og til får man flere treff, men siden samme feilkode brukes på flere forskjellige programmer fra Xilinx, og på de ulike versjonene, man må lete fram til et løsningsforslag som kan passe med programmet man selv bruker.

Det vanligste er, dessverre, at feilmeldingene ikke er dokumentert. Løsningen blir da å søke på Xilinx brukerforum[Xilinx, User Community Forums] eller generelt på internett og håpe at noen andre har fått svar på samme problem tidligere.

Det hjelper ofte å kjøre de innebygde funksjonene for å rense bort alle genererte prosjektfiler (ISE: ”Project” → ”Cleanup Project Files”, XPS: ”Project” → ”Clean All Generated Files”, SDK: ”Project” → ”Clean…”). Hvis noe ulogisk skjer er dette det første man må gjøre, selv om syntetiseringen tar lengre tid neste gang den kjøres. Av og til må man bygge opp hele prosjektet på nytt. Da kan det være greit å ha et utgangspunkt å begynne med. Da vi hadde satt opp hele systemet i ISE og EDK, tok vi kopi av hele prosjektmappen før vi syntetiserte noe som helst. Dermed måtte bare systemendringer siden forrige opprettelse legges inn før prosjektet var i gang igjen. Egne VHDL, c- og h- filer ligger utenom prosjektmappen, så referanser til disse er identiske fra prosjekt til prosjekt.

Noen tips ved bruk av Xilinx ISE og EDK versjon 10 er samlet her:

- Det må aldri være mellomrom i mappestien (eng: path) til prosjektet.
- Hvis det lages en instans av et EDK-prosjekt i en VHDL-fil (se HADAS.vhdl), må instansnavnet skrives med små bokstaver [Xilinx, AR #21458].
- Hvis det er lagt til en testbenk i ISE for å simulere kretsen, må denne fjernes igjen før kretsen syntetiseres [Xilinx, AR #21458].
- Når man bruker ”Create or Import Peripheral” i Xilinx XPS for å opprette egne moduler for PLB-bussen, lages det også c- og h-filer for modulen. Disse kan også inneholde referanser til registre som ikke finnes i modulen.
- Dersom MicroBlaze-prosessoren låser seg på grunn av feil i koden, er det ofte ikke mulig å laste ned ny kode til den. Dette løses ved å først skrive kommandoen ”rst” i ”XMD Console” i SDK for å nullstille prosessoren.
- Fordi det kan være vanskelig å vite hvor feilen har oppstått, vil det ofte lønne seg å ikke gjøre for store endringer mellom hver gang designet syntetiseres.

## 3.6 Maskinvarebeskrivende språk

For å beskrive hvordan den digitale logikken skal være, brukes et maskinvarebeskrivende språk (eng: Hardware Description Language (HDL)). Når en krets er beskrevet i et slikt språk, kan den simuleres eller implementeres i for eksempel en CPLD eller FPGA.

De første digitale kretsene ble beskrevet ved kretsskjemaer som viste hvordan de forskjellige komponentene var koblet sammen. Skjemategninger kan gi god oversikt over kretsen, og brukes også til en hvis grad i dag, men det kan være tungvint å gjøre endringer og utvide kretsen. [Maxfield, 2004, kapittel 8]

Etter hvert fant man ut at det ville være mer effektivt å beskrive kretsene med tekst, og man utviklet ulike maskinvarebeskrivende språk. I begynnelsen lagde gjerne hver produsent sitt eget språk, men etter hvert har man fått standardiserte språk. De mest brukte maskinvarebeskrivende språkene i dag er Verilog (IEEE 1364) og VHDL (IEEE 1076), mens SystemVerilog (IEEE 1800) og SystemC (IEEE 1666) er stadig mer populære [Maxfield, 2004, kapittel 9]. Ellers finnes det mange andre språk [Wikipedia, Hardware Description Language] som f.eks. MyHDL som er et Python-basert språk [MyHDL, From Python to silicon]. Modulene vi har utviklet til dette prosjektet er skrevet i VHDL.

### 3.6.1 Navneregler for VHDL

I all VHDL-kode som er skrevet i prosjektet er følgene koderegler brukt for at koden skal være helhetlig og lettere å forstå:

| | |
|---|---|
| Interne signaler | *_i |
| Aktivt lavt signal | *_n |
| Signaler som går ut av FPGA-en. Brukes kun på toppnivået | *_pin |
| Signaltype | *_type |
| Variabler | *_var |
| Bussbredder | *_width |
| Sammenkobling av enheter (IP-er) | <fra instans>_* (Se kapittel 3.6.1.1) |
| Klokkesignal | clk_* |
| Teller | cnt_* |
| Initialiseringsverdi | init_* |
| Instans av enhet (entity eller component) | inst_* |
| Avbrudd | irpt_* |
| Prosesser | proc_* |
| Tilstandssignal | s_* |
| Asynkront signal | a* |

**Tabell 3-2** Oversikt over navnereglene for VHDL som er brukt i prosjektet. Forklaringer: * erstattes med variabelnavnet.

Det er kun brukt engelske variabelnavn for å unngå æ, ø, å, som ikke er tillatte tegn i VHDL.

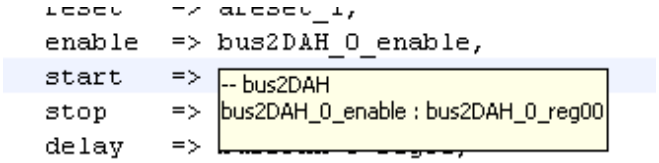
#### *3.6.1.1 Sammenkobling av enheter*

Designet består av en rekke filer som er skrevet i VHDL, og noen komponenter som er generert i XPS og ISE. Hver VHDL-fil inneholder kun én enhet, og hver enhet har kun én oppgave. Det er strukturert slik fordi det gjør det enkelt å bruke enhetene flere steder i samme prosjekt, og det er enklere å teste hver enhet for seg.

Sammenkobling av enhetene i designet skjer kun på toppnivå (filen HADAS.vhdl). Det gir en ryddig struktur, og det er lett å bytte ut enheter eller legge til nye. Enhetene kan dermed kombineres på nye måter, og man kan enkelt lage nye prosjekter med dem.

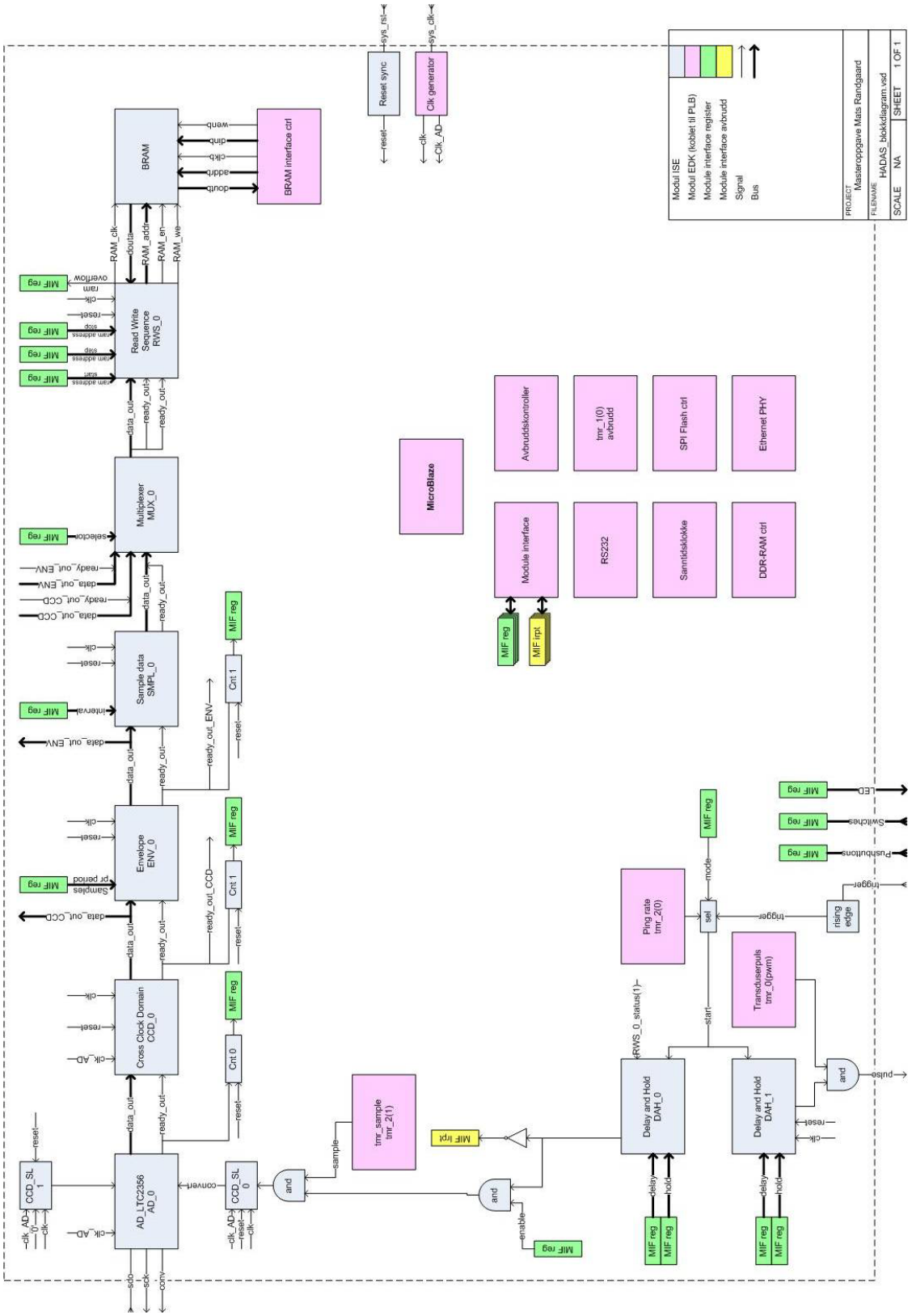
Signalene som binder enhetene sammen, har følgende navngiving: <fra instans>_<utgangsport>. En utgang kan gå til flere innganger. Derfor er det viktig å vite hvor signalet kommer fra. Instansnavnet står først, slik at man kun trenger å huske instansnavnet signalene skal komme fra for å få opp alle utgangsportene med auto-complete i programmeringsverktøyet (f.eks. ctrl + mellomrom i Eclipse).

Buss-signaler som deles opp ved hjelp av alias, navngis på samme måte. F.eks. er bus2HAD_0_reset_all et signal som er hentet ut av bussen bus2HAD_0_reg00. Legg merke til at reg_00 ikke er med i signalnavnet, da det gjør det enklere å flytte signalet til det bus2HAD-registeret som har ledig kapasitet uten å måtte endre signalnavnet.




**Figur 3-14** Ved å holde musen over et signalnavn vises også hvilken bus det kommer fra.

## 3.7 Systemoversikt




**Figur 3-15:** Innholdet i FPGA-en på toppnivå (HADAS.vhdl). (Tegnet i Microsoft Visio).

## 3.8 Enheter

Enheten ”AD_LTC2356” styrer AD-omformeren (se kapittel 2.3.3) og mottar serielle data fra denne. ”Cross Clock Domain” er et bindeledd mellom kretser med ulik klokkefrekvens, slik at dataene kan overføres fra ”AD_LTC2356” til ”Envelope” selv om disse går på forkjellig klokkefrekvens. ”Envelope” henter ut omhyllingskurven til signalet, og gir kun toppverdiene videre til enheten ”Sample data”. Denne brukes til å redusere datamengden hvis dette er ønskelig.

Den neste kretsen i skjemaet er ”Multiplexer”, som styrer hvilke data som skal sendes videre. Data fra ”AD_LTC2356” (via ”Cross Clock Domain”), ”Envelope” eller ”Sample data” kan sendes videre. Enheten ”Read Write Sequence” sørger for å lagre dataene i minneområdet på FPGA-en, ”BRAM”. Dataene i ”BRAM” kan leses av prosessoren gjennom enheten ”BRAM interface ctrl”. Prosessoren, ”MicroBlaze”, styrer alle enheter på PLB-bussen.

Enheten ”Module interface” fungerer som et grensesnitt mellom prosessorsystemet og enhetene som er skrevet i VHDL. Den har registre for å lese inn statusverdier, og for å endre parametre i enhetene. I tillegg har den en inngang for avbruddssignaler. I figur 3-15 er registrene markert med ”MIF reg” og avbrudd med ”MIF irpt”.

Telleren ”Ping rate” genererer en puls hver gang kretsen skal sende ut et ping og begynne å sample data. Kretsen begynner også å sample data hvis det kommer en puls på trigger-inngangen fra kassettspilleren. Hvilken puls som skal detekteres kan velges. Pulsen går videre til ”Delay and hold”-enhetene, ”DAH_0” og ”DAH_1”.

”DAH_1” er ikke implementert i kretsen, fordi det ikke er koblet noen transduser til systemet. Telleren ”Transduserpuls” er likevel implementert i XPS og SDK. Tanken er at ”Transduserpuls” skal generere en kontinuerlig frekvens (50 kHz, 70 kHz, 200 kHz) med ønsket puls/pause-forhold (eng: duty cycle), mens ”DAH_1” styrer lengden på pulsen (0,5-1 ms) som skal sendes ut i vannet.

”DAH_0” brukes til å styre når AD-omformeren skal sample signalet. Etter at den har mottatt pulsen fra ”Ping rate”-telleren kan den vente med å aktivere AD-omformeren, så den ikke sampler pulsen som sendes ut fra transduseren. Deretter holder den utgangen høy så lenge som AD-omformeren skal sample signalet, slik at man kan regulere samplingstiden etter dybden i vannet som skal undersøkes.

”Enable”-signalet ut fra ”DAH_0” går også til avbruddsinngangen på ”Module interface”. Når dette signalet går lavt varsles prosessoren om at pinget er ferdig, og begynner å lese ut data fra BRAM og sender det som UDP-pakker over Ethernet.

Telleren ”Sample rate” genererer en puls hver gang AD-omformeren skal sample signalet. Hvor ofte disse pulsene skal genereres kan settes av prosessoren, slik at signaler med ulik frekvens kan oversamples med et visst antall samplinger pr periode.

Tellerne som er skrevet i VHDL, ”CNT_*”, teller opp hver gang det kommer en puls på inngangen. Disse har blitt brukt under utviklingen av prosjektet for å kontrollere at enhetene fungerer som de skal.

Alle enhetene, testbenker, simuleringsoppsett for ModelSim og simuleringsskript ligger vedlagt på CD-en og på internett. Enheter som ligger i mappen ”Kode\VHDL\matralib” er generelle enheter som også kan brukes i andre prosjekter, mens de som ligger i mappen ”Kode\VHDL\projects\HADAS\VHDL\” er kun for dette prosjektet.

Det er fullt mulig å simulere systemet med prosessor og databusser også, men dette går ganske tregt. Vi har derfor laget en enhet, ”edk_sim.vhdl” som simulerer prosessorens kommunikasjon med de øvrige enhetene som er skrevet i VHDL, slik at samspillet mellom disse kan testes.

### 3.8.1 Grensesnitt mot AD-omformer – AD_LTC2356.vhdl

AD-omformeren som benyttes i prosjektet er LTC2356-14 fra Linear Technology. I dette kapittelet beskrives det serielle grensesnittet, mens den øvrige funksjonaliteten er beskrevet i kapittel 2.3.3 og i databladet for komponenten [Linear, LTC2356].

#### *3.8.1.1 SPI-grensesnittet*

LTC2356-14 har et SPI-kompatibelt grensesnitt for kontroll og utlesing av data. SPI (Serial Peripheral Interface bus) er en synkron seriell bus med 3 eller 4 signallinjer, klokke, inndata, utdata og slavevelger (eng: slave select) [Wikipedia, SPI].
Grensesnittet for LTC2356-14 består av følgende signallinjer:

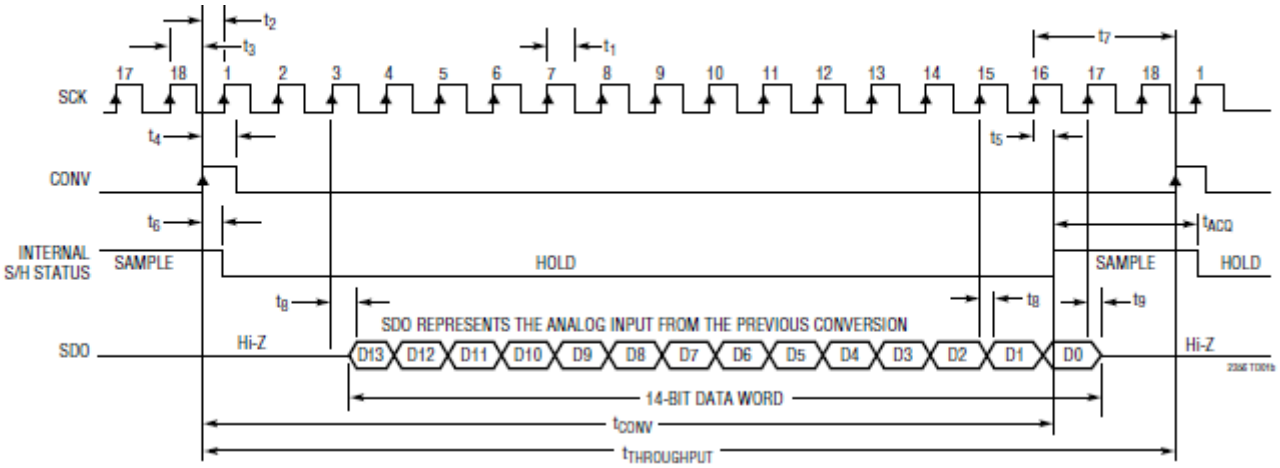
- SCK – klokkesignal
- CONV – Starter konvertering. Tilsvarer linjen for inndata i SPI.
- SDO – Serielle data basert på forrige konvertering.

Signallinjen for slavevelger er ikke implementert.

Det finnes SPI-grensesnitt til Xilinx XPS [Xilinx, DS570]. Denne er koblet til PLB-bussen, og kan ikke brukes i dette prosjektet siden dataene skal prosesseres i VHDL-logikk etter at de er samplet. Vi måtte derfor lage et eget VHDL-grensesnitt mot AD-omformeren.

#### *3.8.1.2 Timing*

Timingdiagrammet i figur 3-16 viser hva som sendes og mottas av AD-omformeren. Når CONV er høy startes konverteringen. Fra og med den tredje stigende klokkeflanken legges verdien av forrige konvertering ut på SDO med MSB først.




**Figur 3-16** Timingdiagram for LTC2356. [Linear, LTC2356, s. 9]

Etter at siste bit er lagt ut, må det gå minst 39 ns før neste konvertering starter, slik at AD-omformeren får samplet nye data på inngangen. Så lenge frekvensen på SCK er lavere enn 1/39 ns = 25,6 MHz, kan ny konvertering starte umiddelbart etter at forrige ble avsluttet. Hele konverteringen skjer i løpet av 16 klokkepulser [Linear, LTC2356, s. 14].

Dersom AD-omformeren skal brukes opp mot maksimal samplingsrate på 3,5 MSPS (SCK = 63 MHz), må man legge til to klokkepulser etter at konverteringen er avsluttet, slik at samplingspausen er ivaretatt [Linear, LTC2356, s. 14]. VHDL-driveren vi har laget bruker alltid 18 klokkepulser pr konvertering, så brukeren kan bruke den på alle frekvenser mellom 100 kHz og 63 MHz [Linear, LTC2356, s. 4].

AD-omformeren leser CONV og legger data ut på SDO på stigende klokkeflanke. VHDL-grensesnittet må derfor sette CONV og lese data fra SDO på fallende klokkeflanke

### *3.8.1.3 Samplings- og konverteringsfrekvens*

Som beskrevet i kapittel 2.3.3.2 er det valgt å sample inngangssignalets bærebølge ti ganger pr periode, slik at samplingspunktene treffer så nær toppunktene i bærebølgen som mulig. Det er CONV-linjen som starter konvertering av inngangssignalet, så det er denne linjen som må settes høy ti ganger pr periode. Mellom hver konvertering må

signalet leses ut av AD-omformeren. Det tar 16 eller 18 klokkeslag på SCK-linjen (se kapittel 3.8.1.2). Figur 3-16 viser hvor ofte CONV-linjen må settes høy, og hvor høy frekvens SCK-linjen minimum må ha for å rekke å konvertere signalet mellom hver konvertering. SCK-linjen kan gjerne ha høyere frekvens enn det som er oppgitt i tabellen. I så fall konverteres dataene raskere, og det blir en pause mellom hver konvertering.

| **Anvendelse** | **Bærebølgens frekvens** | **Konverterings-frekvens CONV (10x oversampling)** | **SCK-frekvens (minimum) 16 perioder pr konvertering** | **SCK-frekvens (minimum) 18 perioder pr konvertering** |
|---|---|---|---|---|
| Ekkolodd | 200 kHz | 2 MSPS | NA | 36 MHz |
| Ekkolodd | 70 kHz | 0,7 MSPS | 11,2 MHz | 12,6 MHz |
| Ekkolodd | 50 kHz | 0,5 MSPS | 8 MHz | 9 MHz |
| Kassettopptak | 10 kHz | 0,1 MSPS | 1,6 MHz | 1,8 MHz |

**Tabell 3-3** Beregning av CONV-frekvens og minste SCK-frekvens. Ved SCK-frekvens høyere enn 25,6 MHz må AD-omformeren bruke minimum 18 klokkeperioder pr konvertering (Se kapittel 3.8.1.2).

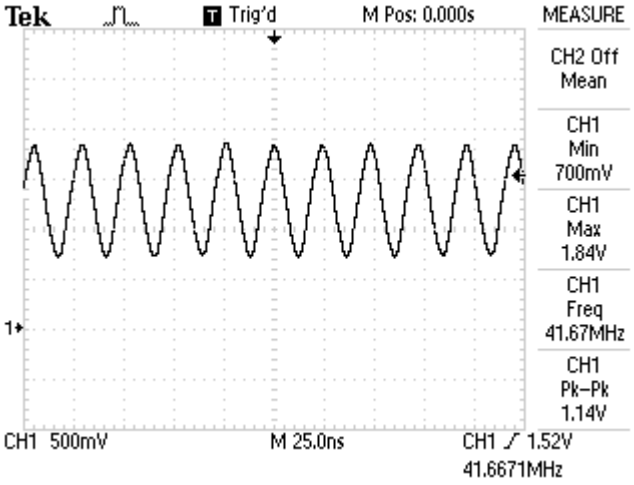
SCK-frekvensen genereres av Clock Generator [Xilinx, DS614] i Xilinx EDK, som automatisk setter opp en DCM (Digital Clock Manager) [Xilinx, UG331, s. 61-150] og andre nødvendige komponenter for å kunne generere de ønskede klokkefrekvensene.

For å generere klokkefrekvenser med lavere frekvens enn inngangsklokken, kan man bruke DCM-ens CLKDV-utgang. Denne utgangen kan dele inngangsklokken med dividender fra 1,5 til 7,5 med steg på 0,5 og fra 8 til 16 med steg på 1 [Xilinx, UG331, s. 62].

SCK-frekvensen kan gjerne være høyere enn tallene som er oppgitt i tabell 3-3. Siden AD-omformeren kan operere med en maksimal SCK-frekvens på 63 MHz, kunne man valgt å dele inngangsklokken med to i DCM-en som gir 62,5 MHz. Dette er frekvens prosessoren går på, og den øvrige logikken er derfor satt til å gå på denne klokkefrekvensen for å ha så få klokkedomener som mulig.

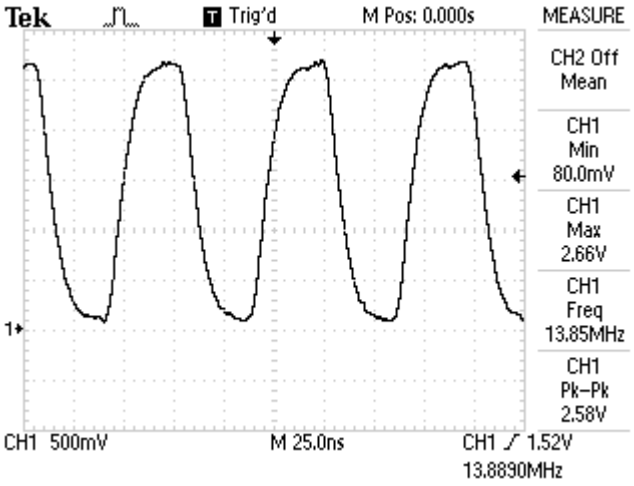
Dessverre er testkortet ikke designet for så høye frekvenser, noe testen med en SCK-frekvens på 41,67 MHz viser. Denne frekvensen ble testet fordi det er den laveste frekvensen som kan genereres av DCM-en som samtidig er høy nok til å kunne sample en bærebølge på 200 kHz (se tabell 3-3). DCM-en deler da inngangsklokken med 3.

Som figur 3-17 viser, klarer ikke FPGA-en å drive signalet lavere enn 700 mV og høyere enn 1,84 V på dette testkortet. Det er ikke tilstrekkelig, da AD-omformeren krever at de digitale signalene må være høyere enn 2,4 V for å registreres som en ener, og lavere enn 0,6 V for å registreres som null [Linear, LTC2356, s. 3]. SCK-signalet oppfyller ingen av kravene ved denne frekvensen, så en lavere frekvens må benyttes.




**Figur 3-17** SCK-linjen fra FPGA til AD-omformer ved 41.67 MHz (Oscilloskopbilde fra ScopePrinterBMP)

Vi har valgt å bruke en frekvens til SCK-linjen på 13,89 MHz, da den er høy nok til at et 70 kHz ekkoloddsignal kan oversamples 10 ganger (se tabell 3-3), samtidig som signalspenningene holder seg innenfor AD-omformerens spesifikasjoner (se figur 3-18).

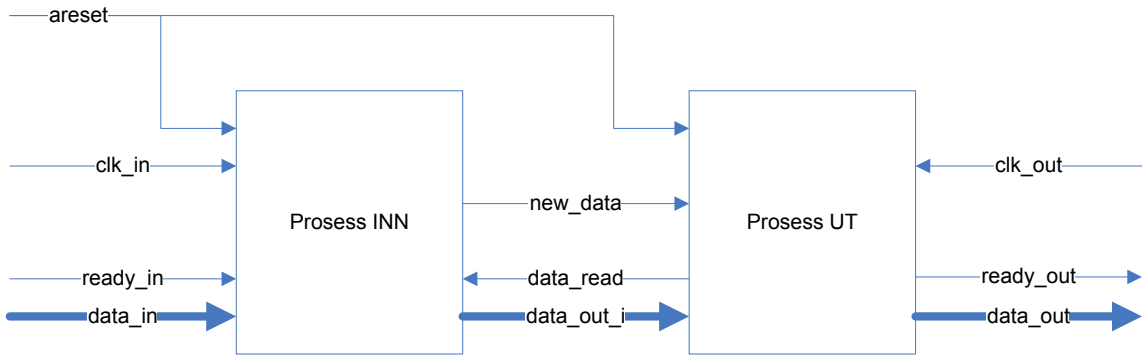



**Figur 3-18** SCK-linjen fra FPGA til AD-omformer ved 13,89 MHz (Oscilloskopbilde fra ScopePrinterBMP)

### 3.8.2 Kryssing av klokkedomener – cross_clock_domain.vhdl

Enhetene som styrer AD-omformeren (kapittel 3.8.1) og DA-omformeren (kapittel 3.8.9) må bruke en lavere klokkefrekvens enn de øvrige enhetene på FPGA-en fordi de kommuniserer med eksterne IC-er. MicroBlaze-prosessoren (kapittel 3.9.1) går på 62,5 MHz. De andre enhetene går derfor på samme frekvens for å ikke få for mange klokkedomener på FPGA-en. AD-omformeren går på 13,89 MHz (se kapittel 3.8.1.3), og DA-omformeren går på maksimum 50 MHz (se kapittel 3.8.9).

For at data fra en enhet skal kunne mottas av en enhet med en annen klokkefrekvens, må overføringen av dataene synkroniseres. Hvis overføringen ikke synkroniseres, vil data kunne forsvinne fordi dataene ikke er tilgjengelige på inngangen til neste enhet, når dennes klokkesignal går høyt [Cummings, 2008, s. 13-15].

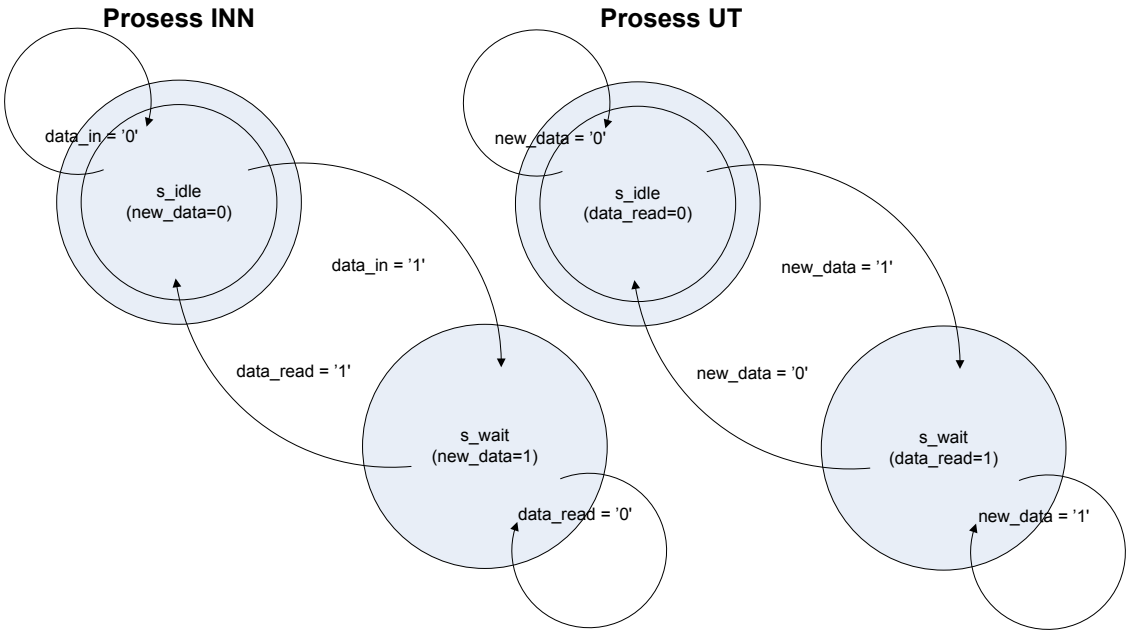



**Figur 3-19:** Interne signaler i cross_clock_domain.vhdl (Tegnet i Microsoft Visio).

Vi har laget egne enheter for å synkronisere overføring av data mellom enheter med ulik klokkefrekvens. Cross_clock_domain.vhdl er for overføring av flere signaler (std_logic_vector), mens cross_clock_domain_std_logic.vhdl er for enkle signaler (std_logic). Disse kan brukes med valgfrie klokkefrekvenser, og kan derfor brukes flere steder i kretsen.

For å sikre korrekt overføring, er kretsene laget med tilbakekobling, slik at mottakerprosessen varsler at data er mottatt [Cummings, 2008, s. 17, 23-28]. Ulempen med å gjøre det på denne måten er at signalet kan bli noe mer forsinket enn ved en

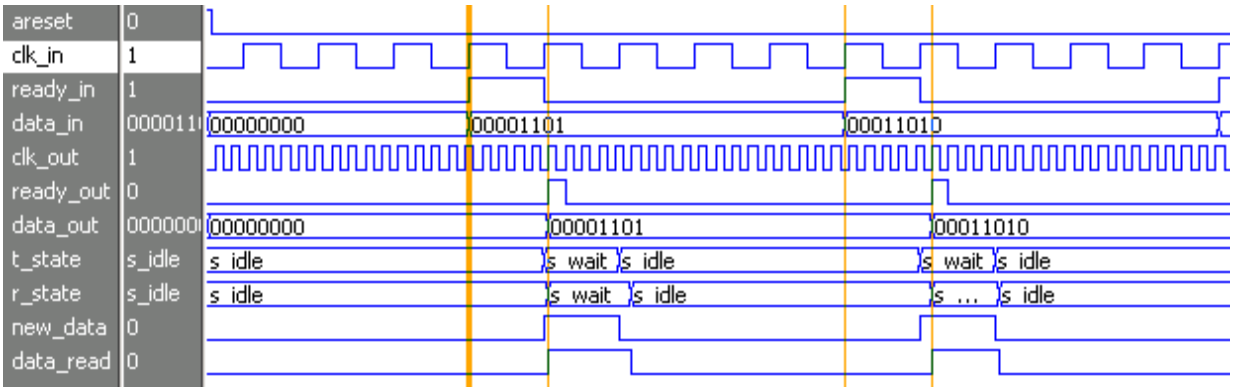
spesialtilpasset løsning. Avstand mellom to etterfølgende pulser må være minimum to perioder av klokken som tar imot. I dette prosjektet samples signalet med maksimalt 2 MSPS, så det er med mye større intervall enn klokkeperiodene i systemet.



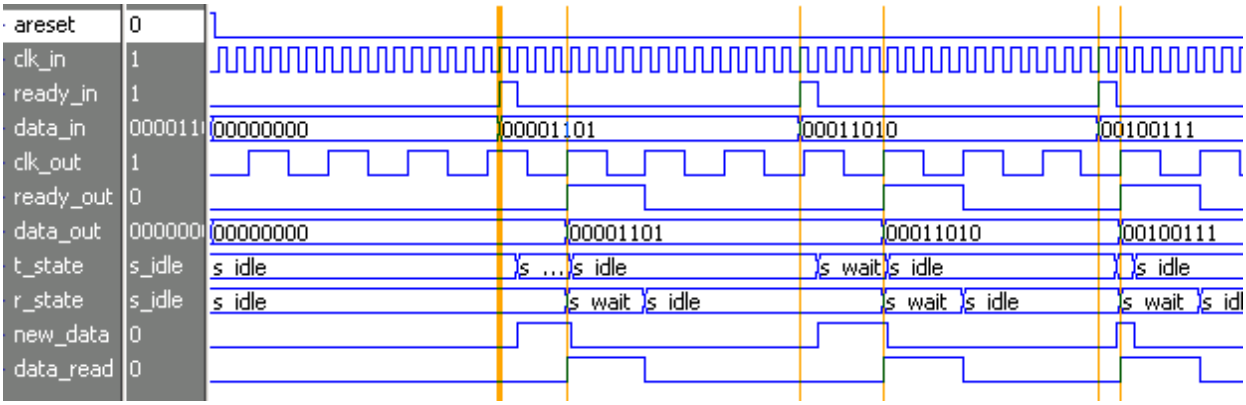

**Figur 3-20:** Flytskjema for cross_clock_domain.vhdl (Tegnet i Microsoft Visio).

Som figur 3-19 og figur 3-20 viser, består enheten av to prosesser, som begge inneholder en tilstandsmaskin. Etter at nye data er tilgjengelige på inngangen skifter tilstandsmaskinene tilstand ved endringer i de interne signalene new_data og data_read. Som simuleringene i figur 3-21, figur 3-22 og figur 3-23 viser, varierer tiden mellom ready_in og ready_out for hver gang data overføres. Klokkene som er brukt (systemklokke 62,5 MHz og AD-klokke 13,89 MHz) er ikke synkrone og forskyver seg derfor i forhold til hverandre.

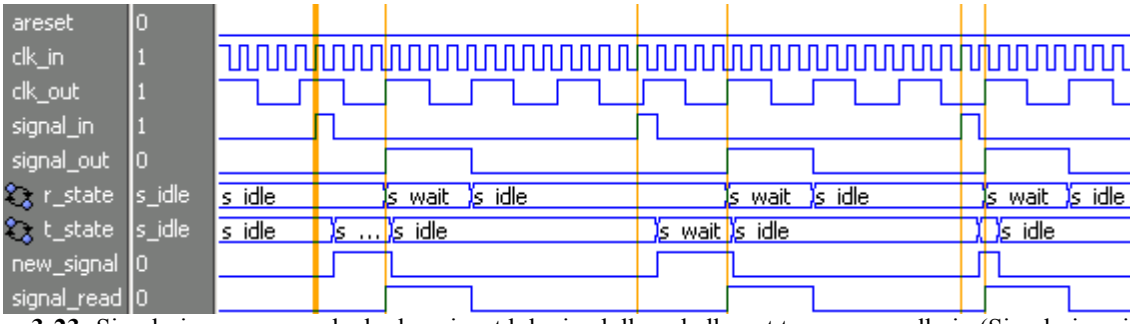
Enheten har asynkron nullstilling for at begge prosessene skal nullstilles samtidig, uavhengig av klokkene.




**Figur 3-21:** Simulering av cross_clock_domain.vhdl med clk_out raskere enn clk_in (Simulering i ModelSim).

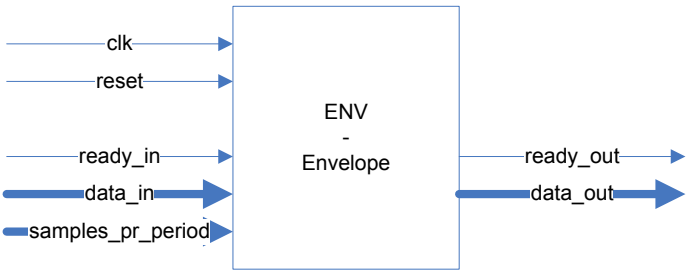



**Figur 3-22:** Simulering av cross_clock_domain.vhdl med clk_out tregere enn clk_in (Simulering i ModelSim).




**Figur 3-23:** Simulering av cross_clock_domain_std_logic.vhdl med clk_out tregere enn clk_in (Simulerings i ModelSim SE 6.5c).

### 3.8.3 Uthenting av omhyllingskurven - envelope.vhdl




**Figur 3-24:** Symbol for envelope.vhdl (Tegnet i Microsoft Visio)

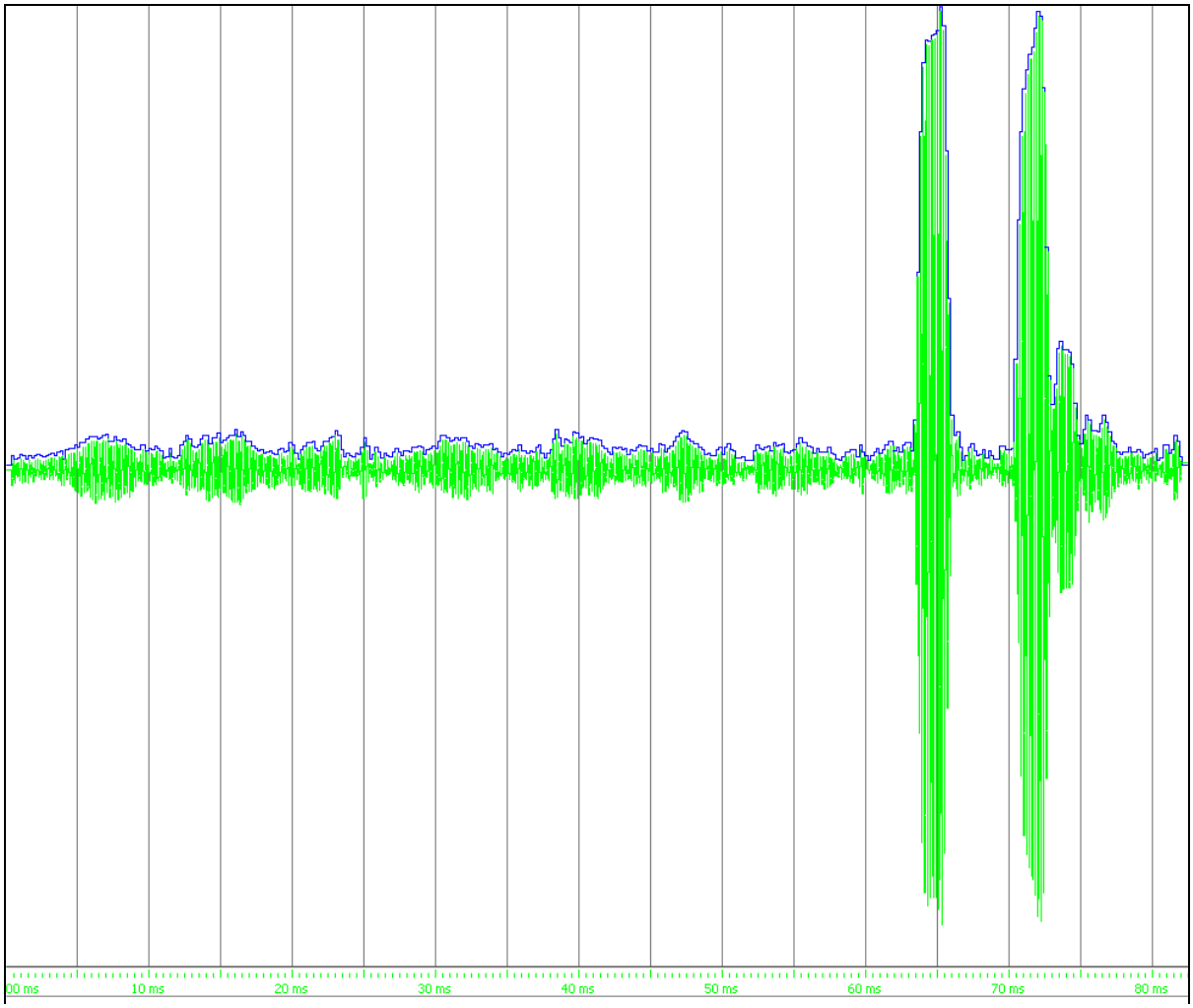
Signalet som er samplet har samme frekvens som pinget som ble sendt ut, men det er omhyllingskurven til dette signalet som er interessant. Uthenting av omhyllingskurven har tidligere blitt gjort med analog signalbehandling før signalet samples. Denne metoden gir et unøyaktig og forvrengt signal [Skumsvoll, 2008, s. 29 og 37-40]. I dette prosjektet samples bærebølgen direkte, mens omhyllingskurven hentes ut av disse dataene i FPGA-en.

Den klassiske metoden for å hente ut omhyllingskurven av et signal er ved hjelp av Hilbert-transform [DSPRelated, Analytic Signals and Hilbert Transform Filters]. Den digitale signalbehandlingen krever mye ressurser, men en DSP FPGA som den som er benyttet i prosjektet kan være egnet for dette.

Fordi bærebølgen i signalet svinger med en fast frekvens, har vi valgt en enklere framgangsmåte for å hente ut omhyllingskurven. Ved å sample bærebølgen flere ganger pr periode, vil et av samplene treffe i nærheten av toppunktet. FPGA-ens oppgave blir deretter å finne dette toppunktet. Toppunktene velges ut ved å sammenligne verdiene fortløpende og ta vare på den største verdien fra hver periode.

Den største utfordringen er å avgjøre hvilke verdier som tilhører hver periode. En metode er å detektere når signalet skifter fra å være positivt til negativt og motsatt. Verdiene fra AD-omformeren er toerkomplemente tall, så når det mest signifikante bittet (MSB) er tallets fortegn.

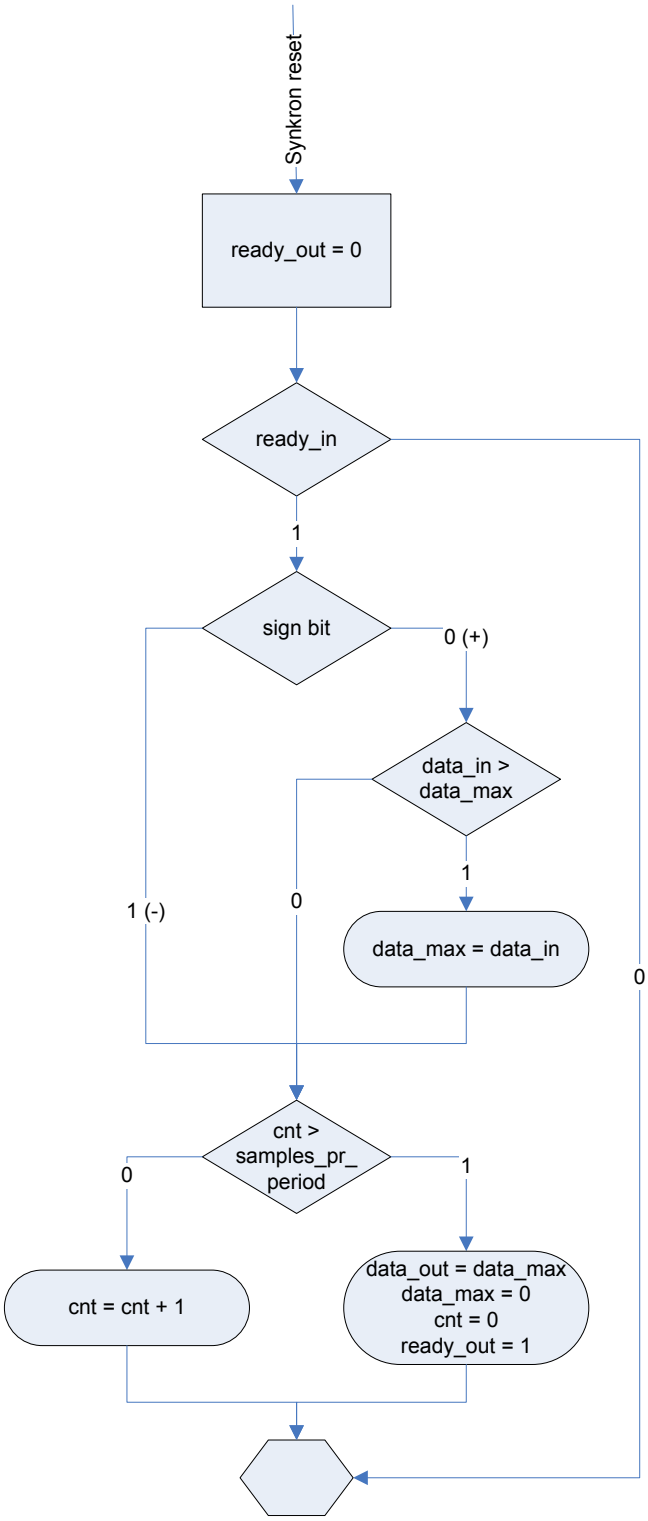
Dette fungerte stort sett veldig bra da vi testet det, men det ble problemer når signalet var så svakt at det ikke skiftet fortegn i løpet av perioden. Fordi det ikke ble tatt vare på noen toppverdi for perioden ble de etterfølgende verdiene forskjøvet en plass fram i forhold til det originale signalet. Et annet problem var at svake signaler kunne skifte fortegn flere ganger innen samme periode på grunn av støy. Dermed ble det noen ganger registrert flere toppverdier enn det var perioder i det opprinnelige signalet.



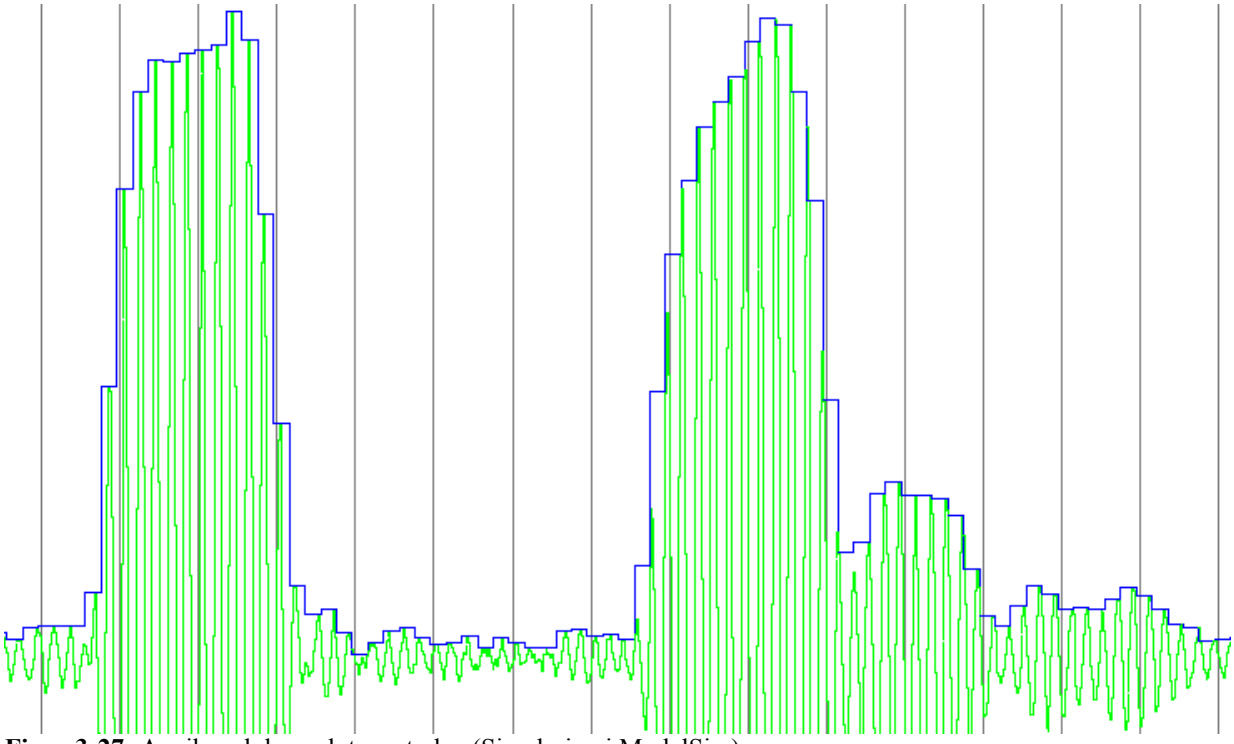

**Figur 3-25:** Simulering av uthenting av omhyllingskurven med virkelige ekkoloddata (Simulering i ModelSim)

Metoden som ble valgt til slutt baserer seg på at signalet er oversamplet et fast antall ganger. Som et utgangspunkt samples signalet med en frekvens som er ti ganger så høy som signalets frekvens. Det betyr at hver periode i signalet samples ti ganger. Blant disse ti verdiene er også toppverdien. Enheten sammenligner derfor ti etterfølgende

verdier og sender videre den høyeste verdien blant disse. Dermed blir én verdi fra hver periode sendt videre, og heller ikke flere.




**Figur 3-26:** Flytskjema for envelope.vhdl (Tegnet i Microsoft Visio)



**Figur 3-27:** Avvik ved den valgte metoden (Simulering i ModelSim)

Dessverre er heller ikke denne metoden helt feilfri. I figur 3-27 ser vi at omhyllingskurven stort sett består av toppunkter, men ved den andre store toppen ser vi at omhyllingskurven dannes av punkter fra siden av bærebølgen. Dette skjer fordi signalet er forskjøvet slik at perioden som sammenlignes ikke går fra nullpunkt til nullpunkt, men fra et stykke opp på toppen av hver bølge. Ved store endringer i amplituden vil derfor punkter i siden på neste bølgetopp være større enn toppunktet i perioden.

Konsekvenser blir at omhyllingskurven forskyves inntil én periode fram i forhold til det opprinnelige signalet. I vannet utgjør denne forskyvningen at usikkerheten ved posisjonsbestemmelse av objekter blir (s = c/f, c = lydhastighet i vann: 1500 m/s) 15 cm ved en bærebølge på 10 kHz (kassett) og 1,05 cm ved en bærebølge på 70 kHz (ekkolodd). Disse avvikene er små, og som man ser av figur 3-27 er bølgeformen av omhyllingskurven godt bevart.

### 3.8.3.1 Problem med dopplereffekt?

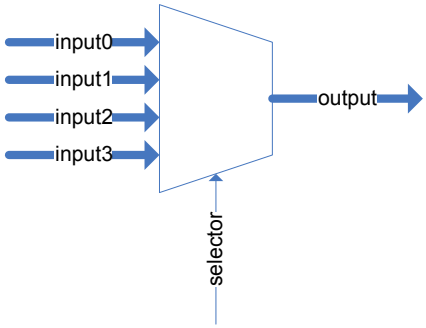
Metoden baserer seg på at frekvensen til bærebølgen er konstant, slik at oversamplingen av signalet gir et fast antall sampler pr periode. Når lydbølgen treffer noe som beveger seg i forhold til transduseren, vil den returnerte frekvensen være endret på grunn av dopplereffekten. Men mottatte frekvensen $f_r$ er gitt av formelen $f_r = f_o(1-2\frac{v}{c})$, der $f_0$ er frekvensen til signalet som ble sendt ut, v er hastigheten til objektet som reflekterer lydbølgen, og c er lydhastigheten i vann [Chapman og Hall, 1992, s. 65].

Hastigheten til objektet kan både komme av at objektet beveger seg i forhold til transduseren, og at transduseren beveger seg i forhold til objektet. Ekkoloddutstyret brukes vanligvis montert på et stativ i vannet eller ved rolig overseiling av området som skal undersøkes. Hvis utstyret er montert på en båt er det kun hastighetskomponenten som er parallell med transduserens retning som har betydning. Ved opptak er det vanlig å holde en fart på 6-8 knop (3-4 m/s), så hastighetskomponenten i transduserens retning er ubetydelig.

I følge [Fish Base, Salmo trutta trutta] svømmer en 38 cm lang sjøørret med en hastighet på 3,26 m/s i et utras. Hvis ørreten svømmer rett mot transduseren i denne hastigheten vil den returnerte frekvensen endres med 225 Hz hvis $f_o$ er 50 kHz, v er 3,26 m/s og c er 1450 m/s. Det medfører at de etterfølgende periodene samples ($1/f_0$-$1/f_r$) 90 ns feil. Avstanden blir dermed (90e-9*100/(1/f0)) 0,45 % feil. I og med at det ikke vil være fisk som svømmer mot transduseren hele tiden vil dette ikke ha noen betydning.

### 3.8.4 Multiplexer - mux_4_to_1.vhdl

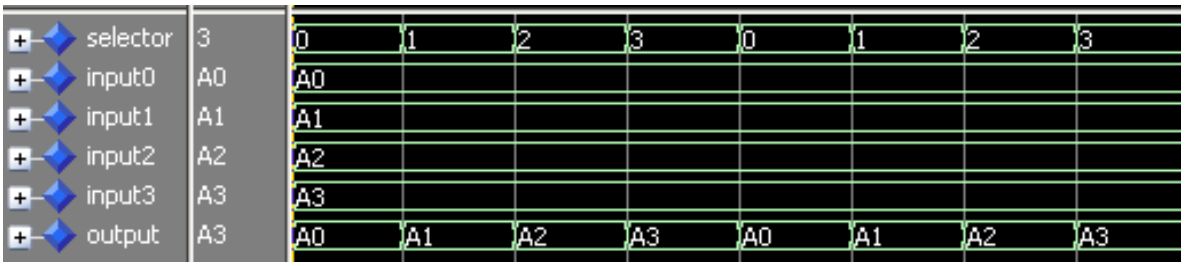
En multiplekser (ofte kalt MUX) brukes til å velge hvilket signal som skal går videre. Vi har laget en multiplekser med fire signaler inn og ett ut. Selector-signalet brukes til å velge hvilket av de fire signalene som skal legges på utgangen.




**Figur 3-28:** Symbol for multiplekseren (Tegnet i Microsoft Visio)

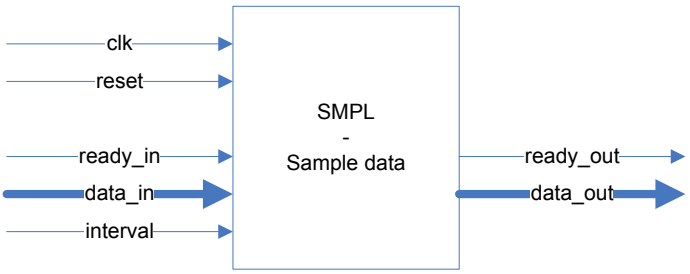
Inngangs- og utgangssignalene er av typen std_logic_vector, men de er ikke begrenset til noen bestemt bussbredde. Dermed er det bredden av signalene som kobles til som avgjør hvor bred signalene gjennom multiplekseren skal være, men alle inngang- og utgangssignaler må være like brede!

Testbenken tb_mux_4_to_1.vhdl legger forskjellige verdier på de fire inngangene og endrer selector-signalet hvert tiende nanosekund. Dette gjøres to ganger. Resultatet demonstrerer kretsens virkemåte og vises i figur 3-29.




**Figur 3-29** Simulering av multiplekseren (Skjermbilde fra ModelSim SE 6.5c)

### 3.8.5 Velge ut data – sample_data.vhdl

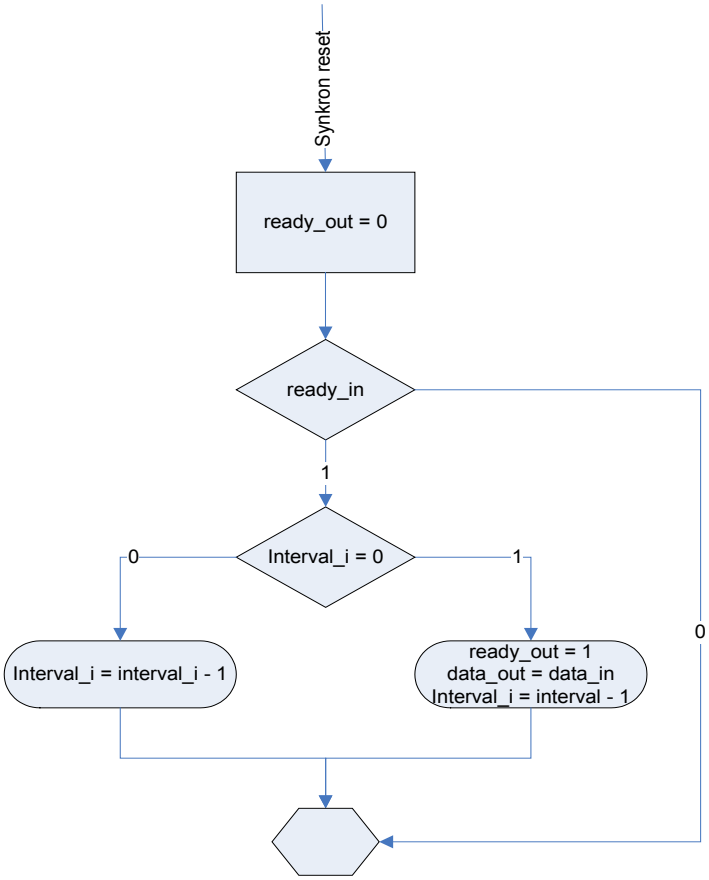



**Figur 3-30:** Symbol for Sample data (Tegnet i Microsoft Visio)

På grunn av begrenset lagringskapasitet i både FPGA-ens BRAM (kapittel 3.8.8) og på datamaskinen som tar imot og lagrer dataene kan det være nødvendig å begrense datamengden. For å kunne velge hvor mange av måleverdiene som skal tas vare på er det laget en enhet som sender videre måleverdier med et konfigurerbart intervall, mens de øvrige verdiene forkastes.

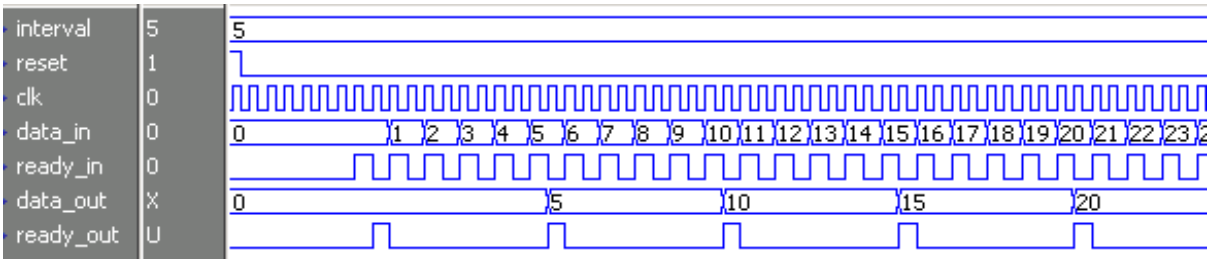
Hvis dataene som behandles er måleverdier direkte fra AD-omformeren tilsvarer dette å sample signalet med en lavere samplingshastighet. Når toppunkter som utgjør omhylligskurven til signalet skal velges ut må samplingshastigheten til AD-omformeren opprettholdes, men etter at toppunktene er valgt ut kan verdiene samples på nytt for å få færre toppunkter.

Ekkoloddsystemene som er utviklet ved Sonargruppen tidligere sampler omhyllingskurven med 10 kSPS. Hvis man bruker samme oppløsning i dette systemet betyr det at man lagrer hvert 5. sampel når bærebølgen er 50 kHz (50 kHz/10 kSPS = 5), hvert 7. sampel når den er 70 kHz og hvert 20. sampel når den er 200 kHz.




**Figur 3-31:** Flytskjema for Sample Data (Tegnet i Microsoft Visio)

Som figur 3-31 viser har enheten en intern teller, Interval_i, som telles ned hver gang et sample er tilgjengelig på inngangen. Når telleren er null legges samplet ut på utgangen mens ready_out går høy for å varsle at nye data er klare. Samtidig tilbakestilles telleren til verdien på Interval-inngangen. Hvis Interval-inngangen har verdien 1 går hvert eneste sample videre. Hvis den er for eksempel 7 går hvert syvende sample videre. Interval_i settes til 0 når enheten nullstilles slik at første sample alltid går videre.




**Figur 3-32:** Simulering av sample_data.vhdl (Simulering fra ModelSim)

### 3.8.6 Sette sammen sekvensielle data - concatenate_sequential_data.vhdl

Da blokkminnet (se kapittel 3.8.8) ble opprettet ble dataene skrevet som 16 bits verdier til minnets port A og lest ut fra port B som 32 bits verdier. Adresseringen i minnet var da som vist i tabell 3-4.

| **Byteadresse Port B + 3** | **Byteadresse Port B + 2** | **Byteadresse Port B + 1** | **Byteadresse Port B + 0** | **Adresse Port A (16 bit)** | **Adresse Port B (32 bit)** |
|---|---|---|---|---|---|
| Sampel 2 (Port A: 0x1) | | Sampel 1 (Port A: 0x0) | | 0x0 | 0x0 |
| Sampel 4 (Port A: 0x3) | | Sampel 3 (Port A: 0x2) | | 0x2 | 0x4 |
| Sampel 6 (Port A: 0x5) | | Sampel 5 (Port A: 0x4) | | 0x4 | 0x8 |
| Sampel 8 (Port A: 0x7) | | Sampel 7 (Port A: 0x6) | | 0x6 | 0xC |

**Tabell 3-4** Minneadressering med 16 bit port A og 32 bit port B

Som vist i tabell 3-4 er det byteadresser på port B, så det er fullt mulig å lese ut data som 8 bits, 16 bits og 32 bits verdier. Vi har valgt å lese ut dataene fra port B som 32 bits verdier fordi det gjør at datautlesningen går dobbelt så fort i forhold til å lese ut to 16 bits verdier.

Når dataene skal sendes som UDP-pakker er det IP-stakken lightweight IP (se kapittel 4.2.1) som leser ut dataene og sender dem. Dermed er det ikke mulig å manipulere på de utleste dataene før de sendes. Det medførte at dataene ble lagt i UDP-pakkene i samme rekkefølge som de har i minnet som 32-bits verdier:

Sampel 2 | Sampel 1 | Sampel 4 | Sampel 3 | Sampel 6 | Sampel 5 osv.

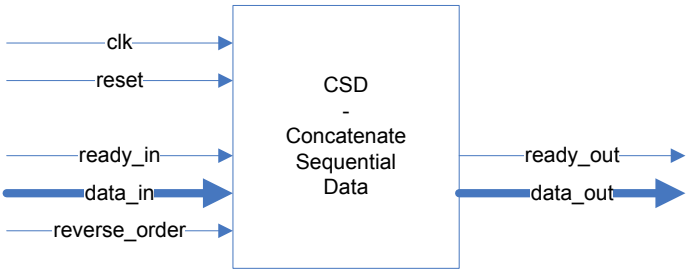
Det mest logiske hadde vært om det ble sendt i følgende rekkefølge:

Sampel 1 | Sampel 2 | Sampel 3 | Sampel 4 | Sampel 5 | Sampel 6 osv.

Vi endret derfor blokkminnet til å ha 32 bits databredde på både port A og B, og lagde denne modulen for å sette to etterfølgende databuss-verdier ved siden av hverandre på en databuss som er dobbelt så bred. Plasseringen av verdiene på utgangsbussen er valgfri slik at samplene kan lagres i minnet i ønsket rekkefølge.

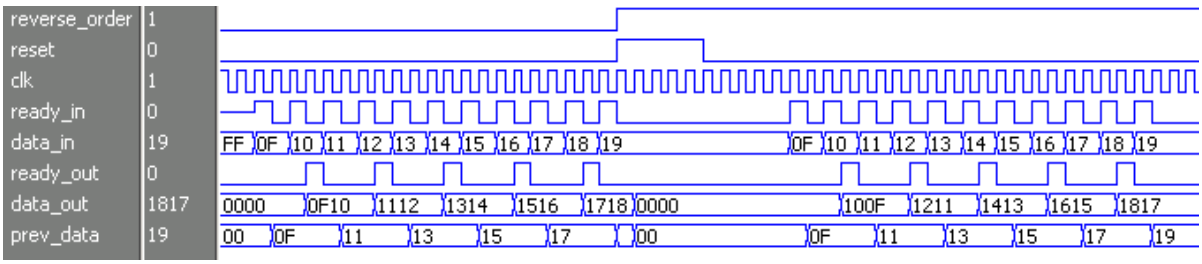
| Byteadresse Port B + 3 | Byteadresse Port B + 2 | Byteadresse Port B + 1 | Byteadresse Port B + 0 | Adresse Port A (32 bit) | Adresse Port B (32 bit) |
|---|---|---|---|---|---|
| Sampel 1 | | Sampel 2 | | 0x0 | 0x0 |
| Sampel 3 | | Sampel 4 | | 0x1 | 0x4 |
| Sampel 5 | | Sampel 6 | | 0x2 | 0x8 |
| Sampel 7 | | Sampel 8 | | 0x3 | 0xC |

**Tabell 3-5** Minneadressering med 32 bit port A og B

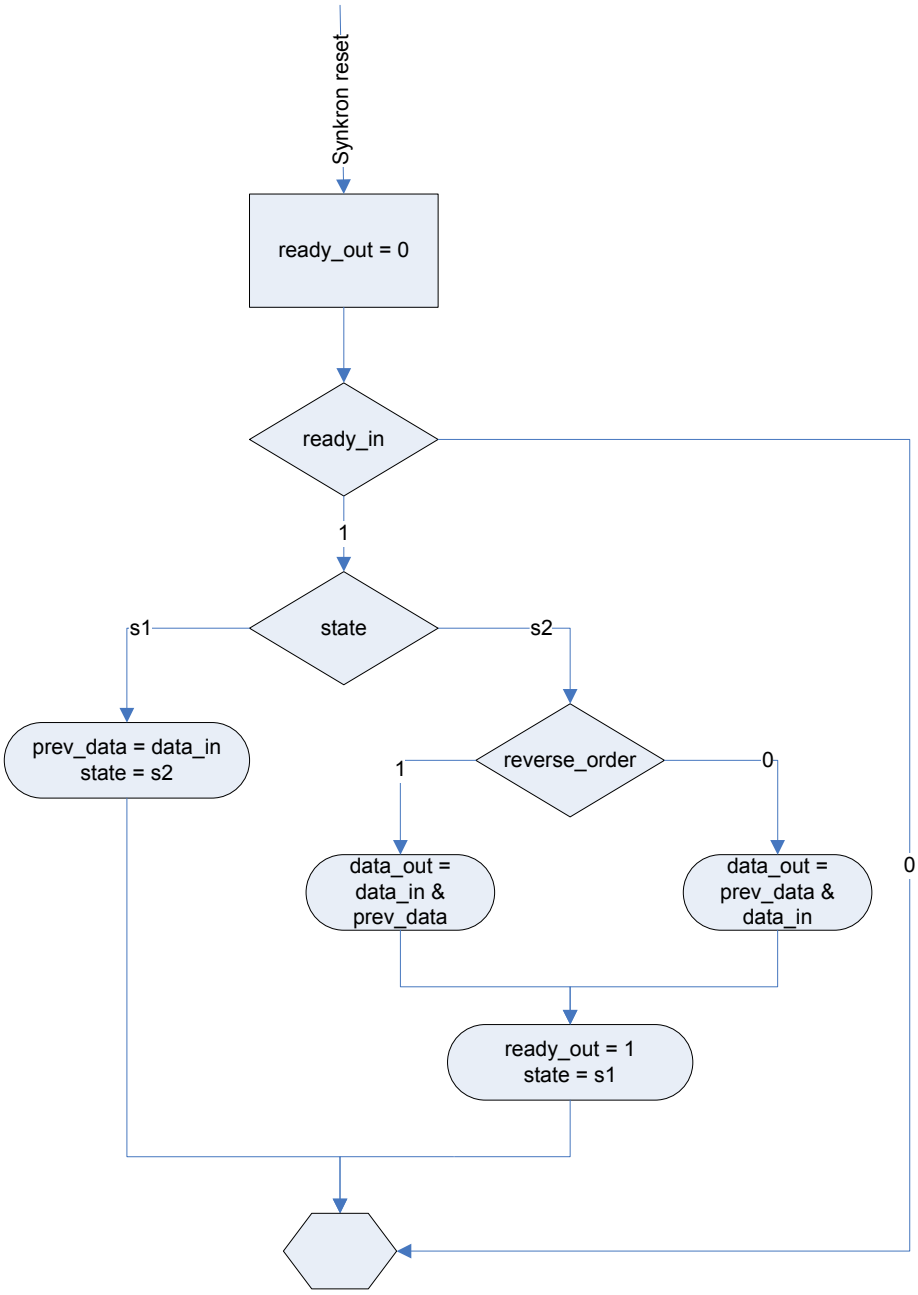



**Figur 3-33** Symbol for concatenate_sequential_data.vhdl

Første gang data mottas på inngangen lagres dette, så det kan legges ut på utgangen neste gang data mottas. Som standard legges de første dataene som mottas fra MSB og til midten av utgangssignalet, og de neste dataene fra midten og til LSB. Dersom reverse_order er én, legges de sist mottatte dataene fra MSB og til midten, mens de første dataene legges fra midten og til LSB. I dette prosjektet er reverse_order satt til null.




**Figur 3-34:** Simulering av concatenate_sequential_data.vhdl (Simulering i ModelSim)




**Figur 3-35** Flytskjema for concatenate_sequential_data.vhdl

### 3.8.7 Skrive data til minneområde - RAM_RW_sequence.vhdl

Sampler som er ferdigbehandlet i maskinvaren skrives til et minneområde slik at prosessoren kan lese og sende dem som UDP-pakker når pinget er ferdig. Vi har derfor laget en VHDL-modul som skriver (og leser) data sekvensielt til minnet.

En lese-/skrivesekvens startes ved at enable-signalet settes høyt i én klokkeperiode. Hvis data skal skrives til minneområdet må write_enable-signalet være høyt samtidig. I så fall skrives dataene som ligger på data_in-porten til minnet. Verdien som minnet returnerer legges til data_out-porten, mens ready_out-porten går høy én klokkeperiode for å varsle at nye data er lagt på utgangen.

Verdiene på portene ram_address_start, ram_address_step og ram_address_stop brukes for å styre hvilke adresser samplene skrives til.

- **ram_address_start** definerer hvilken adresse modulen skal begynne å skrive til. Det gjør det mulig å bruke minneområdet til flere ting. I dette prosjektet starter modulen å skrive til adressen som tilsvarer datafeltet i UDP-pakkene (se kapittel 4.1.3.4), slik at prosessoren kan fylle inn de første adressene med tidsstempling, antall sampler i pakken osv. Dermed kan adressen til minneområdets første adresse overføres til IP-stakken lwIP (se kapittel 4.2.1) som sender alt som ligger i minnet som en UDP-pakke.
- **ram_address_step** avgjør hvor mye adressen skal økes med for hver bolk med data som skal skrives til minnet.
- **ram_address_stop** fastsetter hva som skal være siste adresse som skal kunne skrives til. Når adressetelleren når denne adressen går et varslingsbit i statusregisteret høyt, og modulen slutter å skrive til minnet.

### 3.8.8 Minneområde - BRAM

I FPGA-en er det laget flere minneområder (blokk-RAM (BRAM)), for hurtig lagring av større datamengder enn det som lar seg lagre i distribuert RAM. Minneområdene har to porter som kan brukes til å lese og skrive til minnet samtidig. I Xilinx Spartan 3A DSP 1800A er det 88 blokk-RAM som hver er på 18 kbit, hvorav 16 kbit data og 2 kbit paritetsbit. Hvert minneområde kan konfigureres til å ha alt fra 16000 adresser og 1 bit data, til 512 adresser og 32 bit data. Ved å sette flere minneområder sammen kan man lage større minnestrukturer. [Xilinx, HDL-library, s. 229]

I VHDL lages minneområdet ved å lage flerdimensjonale tabeller og prosesser som leser og skriver til disse [Sjoholm og Lindh, 1997, s. 158-162. ]. Hvis dette gjøres riktig vil syntеseverktøyet skjønne at det er et minneområde som er beskrevet, og bruke et minneområde i FPGA-en. Fordelen med å gjøre det slik er at koden kan brukes på FPGA-er fra forskjellige leverandører uten å måtte endres.

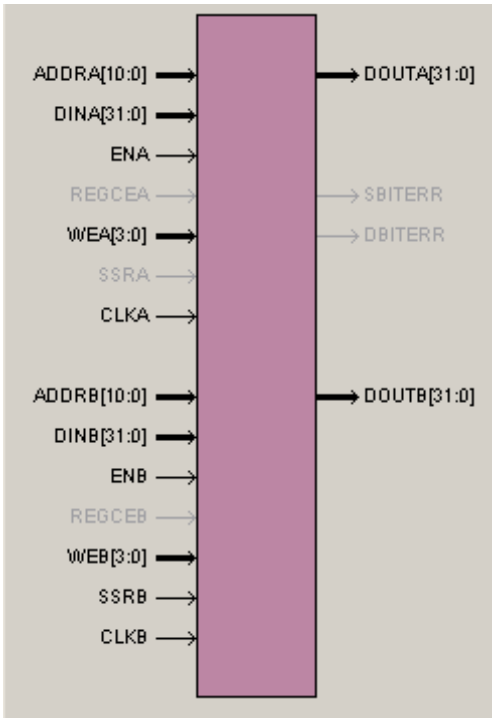
Dessverre er det noe av FPGA-ens funksjonalitet som ikke støttes hvis man gjør det slik. Syntеseverktøyet til Xilinx, XST, klarer blant annet ikke å opprette minneområder med ulik databredde på de to portene [Xilinx, XST, s. 160]. Da BRAM-en ble opprettet var tanken å skrive samplene som 16 bit data til BRAM og lese to og to sampler ut på prosessorsiden som 32 bit data. For å få til dette måtte RAM-blokkene opprettes eksplisitt.

BRAM-blokkene kan opprettes ved strukturell VHDL, men det finnes også verktøy for å opprette disse i både XPS [Xilinx, DS444] og ISE [Xilinx, DS512]. Vi har valgt å opprette BRAM-en i ISE med ”Block Memory Generator” [Xilinx, DS512] i ”CORE Generator & Architecture Wizard”. Da har man mye større valgfrihet enn når man oppretter BRAM i XPS. ”Block Memory Generator” gir full kontroll over hvordan minneområdet skal være, og det er mulig å simulere VHDL-koden sammen med minneområdet, uten å måtte simulere hele prosessorsystemet.

I dette prosjekter er BRAM-en opprettet som en dualport-RAM med portbredde på 32 bits både på port A og port B. Skrivedybden er satt til 2048. Minneområdet er på til sammen 32 *bit* * 2048 = 65536 *bit* fordelt på fire BRAM-blokker á 16384 bit (paritetsbittene brukes ikke).

Altså er det mulig å lagre inntil $\frac{32\ bit}{8\ bit/byte} * 2048 = 8192\ byte$ med data. Hvert sampel er 14 bit, men lagres som 16 bit i BRAM, så det er altså mulig å lagre inntil $\frac{32\ bit}{16\ bit/sample} * 2048 = 4096\ sampler$.

Hvis signalet samples med 10 kSPS (se kapittel 3.8.5), som har vært vanlig i tidligere ekkoloddsystemer, vil det være tilstrekkelig lagringskapasitet for sampler fra 4096 sampler/10 kSPS ≈ 410 ms. I henhold til teorien i kapittel 1.4 tilsvarer det refleksjoner fra en avstand på s = (v*t)/2 = (1450 m/s *410 ms)/2 ≈ 300 m. Hvis signalet samples dobbelt så ofte (20 kSPS), kan man lagre data fra en avstand på ca. 150 m.




**Figur 3-36** Symbol for blokkminnet [Skjermbilde fra "Block Memory Generator" i "Xilinx CORE generator"]

En BRAM-kontroller som kan kommunisere med MicroBlaze-prosessoren er koblet til port B, så en del av valgene for BRAM-en måtte settes etter hvordan den fungerer (se kapittel 3.9.2). Følgende innstillinger er valgt:

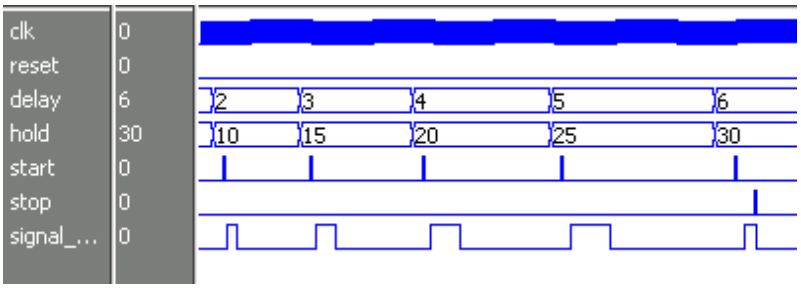
- Port A:
  - Databredde inn og ut er 32 bit.
  - Minnedybden er 2048.
  - Operasjonsmodus er ”read first”.
  - Pin for aktivering av port A (ENA) er tilgjengelig.
  - Pin for nullstilling av port A (SSRA) er ikke tilgjengelig.
- Port B:
  - Databredde inn og ut er 32 bit
  - Minnedybden er 2048.
  - Operasjonsmodus er ”read first”
  - Pin for aktivering av port B (ENB) er tilgjengelig
  - Pin for nullstilling av port B (SSRB) er tilgjengelig fordi BRAM-kontrolleren i EDK har port for dette.
- Ingen registre, verken for hver minneblokk eller for hele minnemodulen.

Alle plasser i minnet settes til verdien null ved oppstart.

### 3.8.9 Pulskontroll – delay_and_hold.vhdl

Som beskrevet i kapittel 3.7 og begynnelsen av 3.8, genereres det en puls fra en prosessorstyrt teller eller på triggerpinnen hver gang et ping skal sendes ut og samplingen skal begynne. For å styre hvor langt pulssekvens som skal sendes ut i vannet, og hvor lenge AD-omformeren skal sample signalet har vi laget enheten delay_and_hold for å styre dette.

Når enheten mottar en puls går den inn i ”delay”-tilstanden (delay). Når telleren for denne tilstanden er utløpt, går den til ”hold”-tilstanden. Utgangen settes da høy, og holdes høy inn til telleren for denne tilstanden er utløpt. Hvor lenge den skal være i de forkjellige tilstandene styres av tellerverdien som settes på ”delay”- og ”hold”-inngangene. En klokkepuls på inngangen blir dermed forsinket og/eller utvidet.




**Figur 3-37:** Simulering av delay_and_hold.vhdl med stadig økende forsinkelse og holdetid. Den siste pulsen stoppes av en puls på stop-inngangen (Simulering i ModelSim)

### 3.8.10 Grensesnitt mot DA-omformer - DAC_AD55X6.vhdl

På testkortet er det montert en DA-omformer av typen AD5546 fra Analog Devices [Analog Devices, AD5546]. Denne ble i starten av prosjektet brukt til å konvertere data fra FPGA-en slik at dette kunne vises på et oscilloskop. Dermed var det mulig å teste de ulike signalbehandlingsmodulene i FPGA-en før UDP-kommunikasjonen var implementert. DA-omformeren kan også brukes til å styre en TVG-krets i ekkoloddet.

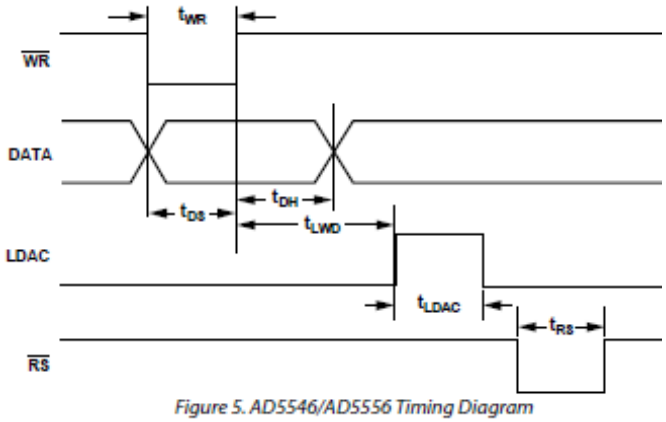



*Figure 5. AD5546/AD5556 Timing Diagram*

**Figur 3-38** [Analog Devices, AD5546, s. 7]

DA-omformeren har 16 parallelle innganger for data. Det er doble registre i AD-omformeren, så når WR-pinnen legges lav lastes dataene inn i inngangsregistrene, men først når LDAC settes høy lastes dataene fra inngangsregistrene til omformingsregistrene som oppdaterer den analoge utgangen. [Analog Devices, AD5546, s. 7 og 12]

**Table 7. Control Inputs**

| RS | WR | LDAC | Register Operation |
|---|---|---|---|
| 0 | X | X | Reset output to 0, with MSB pin = 0. Midscale with MSB pin = 1. |
| 1 | 0 | 0 | Load input register with data bits. |
| 1 | 1 | 1 | Load DAC register with the contents of the input register. |
| 1 | 0 | 1 | Input and DAC registers are transparent. |
| 1 | ⊓ | ⊓ | When LDAC and WR are tied together and programmed as a pulse, the data bits are loaded into the input register on the falling edge of the pulse, and then loaded into the DAC register on the rising edge of the pulse. |
| 1 | 1 | 0 | No register operation. |

**Figur 3-39** [Analog Devices, AD5546, s. 7]

I følge figur 3-39 og [Analog Devices, AD5546, s. 12] er det mulig å oppdatere utgangen på DA-omformeren på ulike måter. Hvis WR er 0 og LDAC er 1 vil DA-omformeren være transparent, slik at dataene som kommer på inngangen til DA-omformeren vil endre det analoge signalet umiddelbart. Dette kan være en uheldig framgangsmåte fordi det ikke er garantert at alle 16 bit endres nøyaktig samtidig. Det vil i så fall medføre at den analoge utgangen vil endre seg inntil inngangen er stabil.

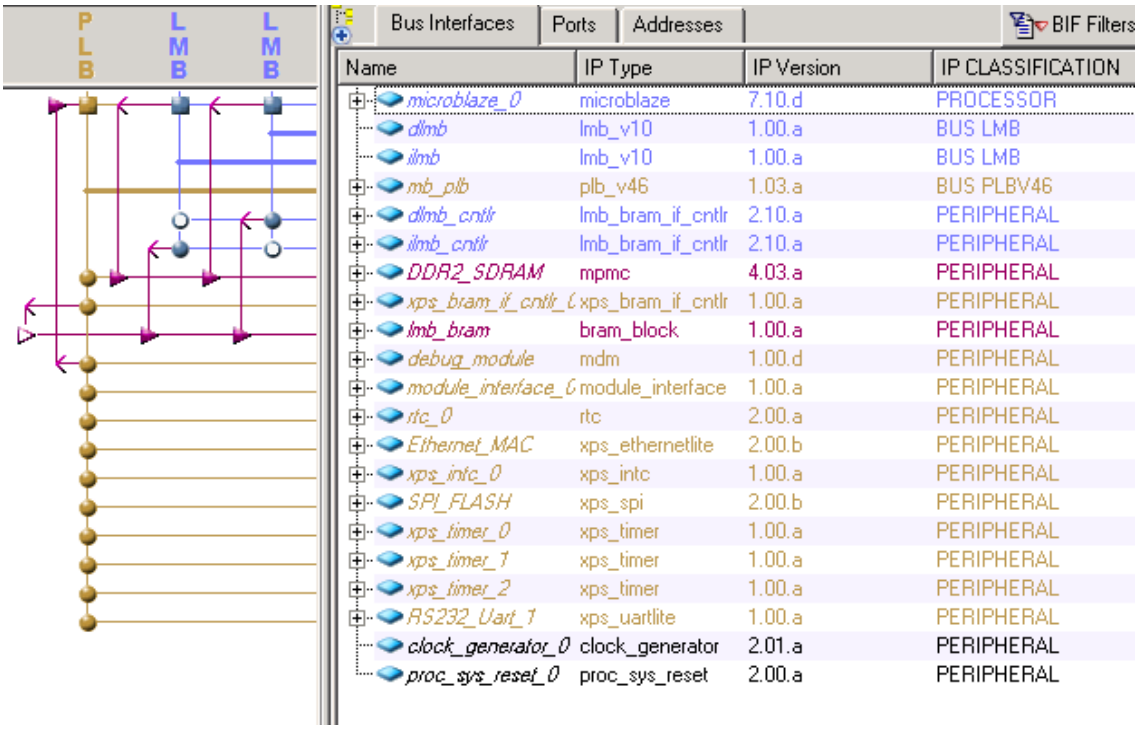
Ved å først laste inngangsverdiene inn i DA-omformeren, for så å konvertere dem, er man sikret at kun ønskede verdier legges på utgangen. Dette kan i følge figur 3-39 enten gjøres ved å først sette WR til 0 og deretter sette LDAC til 1, eller ved å koble WR og LDAC sammen og la disse gå i en negativ puls.

## 3.9 Moduler i Xilinx Platform Studio

Xilinx Platform Studio (XPS) (se kapittel 3.5.2.1) brukes til å sette sammen prosessorsystemet på FPGA-en. Ved hjelp av det grafiske brukergrensesnittet kan moduler legges til, endres og fjernes, mens XPS sørger for å oppdatere de underliggende filene. Felles for alle enhetene som konfigureres i XPS er at de er koblet sammen med databusser.

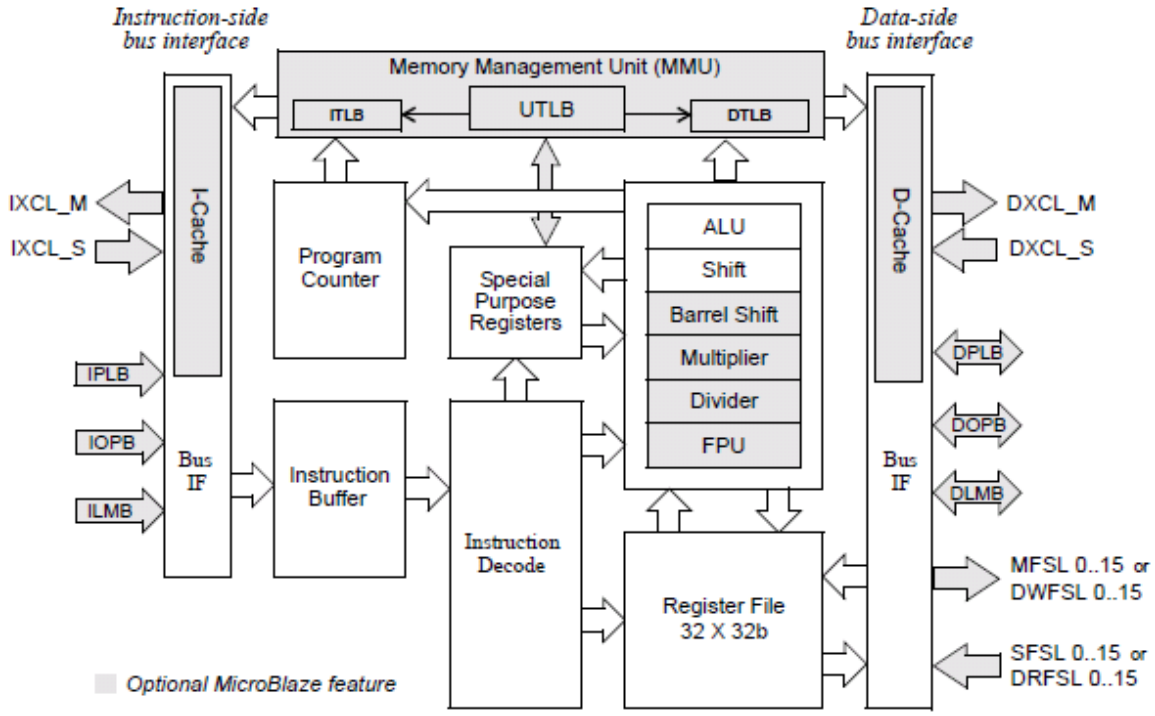
De fleste enhetene henger på Processor Local Bus (PLB). Databussen er basert på CoreConnect fra IBM. Den kan brukes med 32-bit, 64-bit eller 128-bit mastere og slaver [Xilinx, DS531]. I dette systemet er det kun én master, MicroBlaze-prosessoren, og alle enhetene som er koblet til bussen er 32-bit.

Lokale minneområder på FPGA-en, BRAM (se kapittel 3.8.8), kan kobles til prosessoren med Local Memory Bus (LMB). Dette er en 32-bits databuss som kun tillater én master. Inntil 16 slaver kan kobles til hver buss, men ofte opprettes det en egen buss for hver slave. Fordelen med LMB framfor PLB, er at LMB bruker færre klokkesykler pr dataoverføring på bussen [Xilinx, DC445].




**Figur 3-40:** Enheter i Xilinx Platform Studio med oversikt over hvilke busser de er tilkoblet (Skjermbilde av XPS)

### 3.9.1 Prosessor - MicroBlaze




**Figur 3-41:** Enheter i MicroBlaze-prosessoren [Xilinx, UG081, s. 10]

Prosessoren er en MicroBlaze-prosessor fra Xilinx. Dette er en RISC-prosessor (Reduced Instruction Set Computer) med 32-bits instruksjoner.

Dette er en soft-core-prosessor. Det vil si at den er implementert i FPGA-ens logikkelementer. Prosessoren kan dermed konfigureres til oppgavene den skal utføre. Som vist i figur 3-41 er flere av prosessorkjernens elementer valgfrie. Innholdet i disse kan også konfigureres. Vi har blant annet valgt å inkludere hurtigminne (eng: cache) for å få raskere tilgang til instruksjoner i DDR-RAM (se kapittel 3.9.7), Barrel Shifter for å kunne utføre bitskifting på én klokkeperiode og 32-bits multiplikator for raskere multiplikasjon.

Hvis noen instruksjoner i programvaren tar for lang tid å utføre kan man lage digital logikk som utfører den samme oppgaven på kortere tid og koble den til prosessoren som en coprosessor med FSL-tilkobling [Xilinx, Embedded System Tools Reference Manual, s.26].

Prosessorsystemet kan også utvides med å koble andre enheter til en felles databuss. Dermed har prosessoren tilgang på spesiallagde enheter, og kan også brukes til å styre og konfigurere enhetene på FPGA-en.

### 3.9.2 BRAM-kontroller – lmb_bram_if_cntrl

For at prosessoren skal kunne skrive og lese til minneområdet der dataene fra ekkoloddet er lagret (se kapittel 3.8.8) brukes en minnekontroller. Vi har valgt å bruke lmb_bram_if_cntrl som er en ferdiglaget enhet fra Xilinx. Denne er koblet til Local Memory Bus (LMB) (se kapittel 3.9) for raskest mulig overføring av data.

Xilinx har også laget en minnekontroller som kobles til Processor Local Bus (PLB) (se kapittel 3.9). Denne heter xps_bram_if_cntrl. Hvis vi hadde brukt en annen Ethernet MAC (se kapittel 3.9.6) som hadde støttet direkte minnetilgang (DMA - Direct Memory Access) ville det vært en fordel å bruke denne i stedet. Da kunne Ethernet MAC-en hentet data fra BRAM uten at prosessoren ville blitt belastet under overføringen.

BRAM-kontrolleren har 32-bits adresser og 32-bits databredde, men den kan også overføre enkeltbyte. For hvert dataområde på 32-bit er det altså fire adresser, en til hver byte. Adressestørrelsen man oppgir i Xilinx Platform Studio er derfor antall byte kontrolleren skal kunne adressere. I dette tilfellet er det 8k.

### 3.9.3 Avbruddskontroller – xps_intc

Avbruddskontrolleren er en ferdiglaget enhet fra Xilinx (xps_intc) [Xilinx, DS572]. Den samler alle avbruddlinjer fra hele kretsen og gir et avbrudd til MicroBlaze-prosessoren dersom det kommer et avbrudd på en av disse. Når MicroBlaze-prosessoren får et avbrudd kontakter den avbruddskontrolleren via PLB-bussen for å få informasjon om hvilken enhet som hadde generert avbruddet.

Avbruddskontrolleren kan ha inntil 32 avbruddslinjer tilkoblet, og kan prioritere disse i forhold til hverandre. I dette prosjektet er avbruddslinjer fra Ethernet MAC-en (kapittel 3.9.6.3), den ene timeren (kapittel 3.9.4) og Module Interface (kapittel 3.9.9) koblet til. Ved kjøring av kode som er prioritert kan avbruddsvarslingen til prosessoren skrus av slik at den ikke trenger å bruke tid på å behandle avbruddet.

### 3.9.4 Timer – xps_tmr

I prosjektet brukes flere timere av typen xps_tmr fra Xilinx [Xilinx, DS573]. Hver timer inneholder to tellere og kan konfigureres til å gi avbrudd, registrere tidspunkt (tellerverdi) for hendelser og generere vanlige og pulsbreddemodulerte pulser. Andre egenskaper for tellerne - som antall bit, om de skal telle oppover eller nedover og automatisk omstart kan også konfigureres.

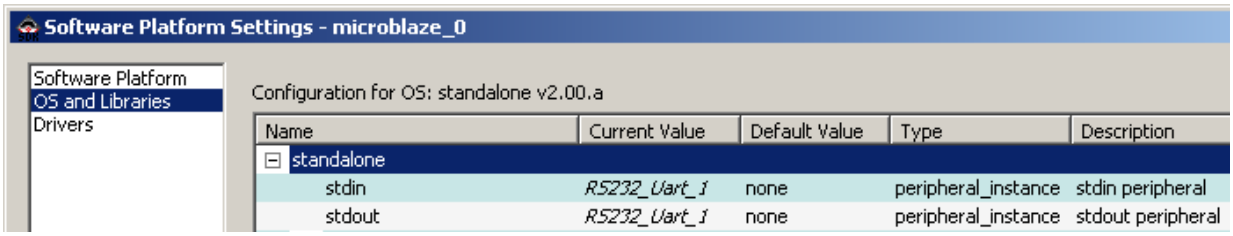
I prosjektet er følgende timere satt opp:

- timer 0:
  - Begge tellerne brukes for å generere et pulsbreddemodulert signal til transduseren. Teller 0 brukes til å bestemme avstanden mellom pulsene, mens teller 1 styrer lengden på hver puls. Verdiene for det pulsbreddemodulerte signalet er gitt ved følgende formler:
    - PWM_PERIOD = (TLR0 + 2) x PLB_CLOCK_PERIOD
    - PWM_HIGH_TIME = (TLR1 + 2) x PLB_CLOCK_PERIOD
    - TLR0 og TLR1 er tellerverdiene for henholdsvis teller 0 og 1, og settes i C-koden. PLB_CLOCK_PERIOD er klokkeperioden på PLB-bussen som er (1/62,5 MHz) 16 ns.
- timer 1
  - Teller 0 genererer avbrudd for MicroBlaze-prosessoren. Dette avbruddet brukes for å kalle funksjoner som må kalles med gitte mellomrom, blant annet i IP-stakken Lightweight IP (se kapittel 4.2.1). Tiden mellom hvert avbrudd er gitt ved følgende formel:
    - TIMING_INTERVAL = (TLR0 + 2) x PLB_CLOCK_PERIOD
    - TLR0 er tellerverdien for teller 0, og settes i C-koden. PLB_CLOCK_PERIOD er klokkeperioden på PLB-bussen som er (1/62,5 MHz) 16 ns.
  - Teller 1 er ikke i bruk.
- timer 2
  - Teller 0 genererer en puls når transduseren skal sende ut et ping. Avstanden mellom hvert ping er gitt ved samme formel som for timer 1.
  - Teller 1 genererer en puls hver gang AD-omformeren skal sample en ny verdi (se kapittel 3.8.1). Avstanden mellom hver sampling er gitt av samme formel som for timer 1.

### 3.9.5 UART – xps_uartlite

På FPGA-testkortet er det en RS232-transceiver fra Texas Instruments av typen MAX3221 [TI, MAX3221]. Denne er koblet til en DB9-kontakt som kan kobles med en hann-til-hunn nullmodemkabel (krysskoblet) til en standard seriell PC-port. RS232-transcieverens Rx- og Tx-porter er koblet til FPGA-ens pinnummer N21 (Rx) og P22 (Tx) [Xilinx, UG454, s. 17].

Ved å sette opp prosessorsystemet med en UART (Universal Asynchronous Receiver Transmitter), kan man sende og motta data via dette grensesnittet. I dette systemet brukes XPS UART lite som er en IP-modul fra Xilinx [Xilinx, DS571]. Den er koblet til PLB-bussen og har ellers utganger for Rx, Tx og avbrudd. Rx og Tx er koblet til de respektive portene på RS232-transcieveren MAX3221, mens avbruddslinjen er åpen. Det er mulig å bruke de medfølgende API-filene, men ved å sette stdin og stdout til denne modulen i Xilinx SDK blir grensesnittet benyttet ved for eksempel print()-, printf()-, scanf()-kommandoene uten at man behøver å konfigurere systemet ytterligere. Dette gjøres i Xilinx XPS med menyvalgene ”Xilinx Tools” → ”Software platform settings…” → ”OS and Libraries”. Se figur 3-42.




**Figur 3-42:** print-kommandoer kan settes til å skrive til UART (Skjermbilde av Xilinx SDK).

Egenskapene for UART-en kan justeres i Xilinx XPS ved å høyreklikke på enheten i ”System Assembly View” og velge ”Configure IP…”. UART-en er i dette prosjektet satt opp med følgende innstillinger:

| | |
|---|---|
| **Baudrate** | 115200 bps |
| **Databit pr ramme** | 8 |
| **Paritetsbit** | Ingen |

**Tabell 3-6**

Legg merke til at MAX3221 maksimalt overføre data med en baudrate på 250 kbps [Texas Instruments, MAX3221, s. 1], som i praksis vil si 230400 bps når den er koblet til UART-en.

## 3.9.6 Ethernet MAC – xps_ethernetlite

### *3.9.6.1 Ethernet*

I 1972 utviklet Xerox den første Ethernet-standarden, men den har siden den gang blitt endret og utvidet flere ganger. Den er nå standardisert av IEEE (The Institute of Electrical and Electronics Engineers, Inc.) som IEEE 802.3. Standarden består av flere understandarder som beskriver kommunikasjon over ulike medium (tvunnet kobberkabel, koaksialkabel, fiber) med forskjellig hastighet (10 Mbit/s, 100 Mbit/s, 1 Gbit/s osv.) på det fysiske laget i OSI-modellen. I tillegg defineres Media Access Control (MAC) som styrer tilgangen til mediet, og formatet på adresserammen (tabell 3-7) [Wikipedia, Ethernet].

| Blokkstart (eng: Preamble) | Rammeavgrenser (eng: Start-of-Frame-Delimiter) | MAC-adresse for destinasjon | MAC-adresse for avsender | Type eller lengde | Nyttelast (eng: Payload ) | Integritetssjekk (CRC32) | Ventetid før neste ramme (eng: Interframe gap) |
|---|---|---|---|---|---|---|---|
| 7 byte med 10101010 | 1 byte med 10101011 | 6 byte | 6 byte | 2 byte | 46-1500 byte | 4 byte | Tiden det tar å sende 12 byte (960 ns ved 100 Mbit/s) |
| | | 64-1518 byte | | | | | |

**Tabell 3-7:** Ethernet-ramme [Wikipedia, Ethernet]

### *3.9.6.2 PHY*

Testkortet er utstyr med en Ethernet PHY fra National Semiconductor med betegnelsen DP83865DVH Gig PHYTER V [National, DP83865]. Denne tar seg av det fysiske laget i OSI-modellen og støtter IEEE 802.3 10BASE-T, 100BASE-TX og 1000BASE-T. De øvrige lagene i OSI-modellen må eventuelt implementeres i FPGA-en. Den kan overføre data med en hastighet på 10, 100 eller 1000 Mb/s, men det krever at de øvrige lagene også gjør det [Xilinx, UG454, s. 16-19].

PHY-en er koblet til en Tyco AMP RJ-45 kontakt [Tyco, Mag45] for tilkobling av Ethernetkabel. Den driver et sett med lysdioder for å indikere linkens hastighet og om

kommunikasjonen er fullduplekс eller ikke. PHY-ens grensesnitt mot FPGA-en er standard GMII (Gigabit Media Independent Interface). Pinnene som er brukt er listet opp i [Xilinx, UG454, tabell 7].

Enheten kan konfigureres ved å sette jumpere i henhold til [Xilinx, UG454, tabell 8], men blant standardinnstillingene kan det nevnes at den støtter automatisk forhandling om linkhastighet og autodeteksjon av nettverkskabel, så man ikke trenger kryssede nettverkskabler for å koble systemet direkte til en PC.

#### *3.9.6.3 Ethernet MAC*

For å drive PHY-en må en Ethernet MAC (Media Access Control) implementeres i FPGA-en. Denne kommuniserer med PHY-en i henhold til Ethernet standarden for MII (Media Independent Interface). MII finnes i ulike versjoner, som for eksempel GMII for datarater opptil 1 Gbit/s. MII-standarden sørger for at alle enheter som støtter denne kan brukes sammen. Noen av FPGA-ene fra Xilinx leveres med en MAC integrert på IC-en, men det er ikke tilfelle for Spartan-3A DSP.

IP-er for Ethernet MAC lages av flere leverandører, og det finnes flere åpen-kildekode-prosjekter hos OpenCores.org som for eksempel [OpenCores.org, tir-mode Ethernet MAC]. Vi ønsket å bruke en Ethernet MAC som er levert av Xilinx så vi ikke skulle få unødvendige problemer med å integrere den mot MicroBlaze-prosessoren, og fordi Xilinx har tilpasset IP-stakken lightweight IP (lwIP) (se kapittel 4.2.1) til disse. Xilinx leverer to IP-er for Ethernet MAC, nemlig XPS 10/100 Ethernet MAC Lite og XPS LocalLink Tri-mode Ethernet MAC.

**XPS LocalLink Tri-mode Ethernet MAC** [Xilinx, DS537] er kun gratis for FPGA-er med integrert MAC. På FPGA-er som ikke har det vil den fungere som en soft-MAC, og koster da penger å bruke. Dette er en meget avansert Ethernet MAC som blant annet støtter overføringshastigheter opptil 1 Gbit/s, DMA (Direct Memory Access), Jumbo Frames (pakker opptil 9 kB) og kalkulering av CRC for TCP og UDP i maskinvaren.

#### 3.9.6.4 XPS 10/100 Ethernet MAC Lite

XPS 10/100 Ethernet MAC Lite [Xilinx, DS580] er en soft-MAC som kun tilbyr de mest grunnleggende funksjonene for Ethernet-kommunikasjon, men den er i motsetning gratis. Vi valgte å bruke denne i prosjektet vårt, fordi datamengdene som skal overføres er små, og vi har ikke bruk for de mest avanserte funksjonene.

Enheten er koblet til PLB-bussen og har MII (Media Independent Interface) grensesnitt mot PHY-transceiveren. Bussfrekvensen på PLB-bussen må være minst 50 MHz hvis Ethernet-linken skal overføre data med inntil 100 Mbit/s, og mist 5 MHz hvis overføringshastigheten skal være inntil 10 Mbit/s [Xilinx, DS580, s. 17]. I dette systemet er bussfrekvensen 62,5 MHz, men det er flere faktorer som gjør at det likevel ikke er mulig å oppnå 100 Mbit/s i praksis. Blant annet er lwIP-stakken som benyttes i følge Xilinx' egne tester ikke er i stand til å oppnå høyere overføringshastighet enn 15 Mbit/s med MicroBlaze-prosessor og Ethernet MAC Lite på en Spartan-3-FPGA [Xilinx, lwIP, s. 12].

I standardkonfigurasjon har enheten et buffer for sending og et for mottak av pakker som hver er på 2 kB, og altså rommer én pakke. For å øke ytelsen er det mulig å legge til et ekstra buffer for hver retning, slik at det ene kan leses mens det andre skrives til (ping-pong). Dette gjøres ved å høyreklikke på enheten i "System Assembly View" og velge "Configure IP…". Der velges "Include Second Receiver Buffer" og "Include Second Transmitter Buffer". Doble buffer er lagt til i dette prosjektet [Xilinx, DS580].

Data overføres til og fra bufferne som skrive- og leseoperasjoner til bufferminnet som 32-bit-verdier, og det er ingen støtte for DMA. Burst-overføring av data er heller ikke støttet [Xilinx, DS580, s. 17], så hver verdi må skrives/leses direkte til/fra en adresse. Prosessoren må derfor brukes til overføre dataene, og dette er en tidkrevende prosess og dårlig utnyttelse av ressursene. I dette prosjektet har det likevel vist seg å være tilstrekkelig ytelse, men hvis det i framtidige prosjekter blir problemer med dette, bør man bytte ut Ethernet MAC-en med en som støtter DMA.

### 3.9.7 DDR-RAM grensesnitt - mpmc

FPGA-kortet har to DDR2 RAM-brikker på til sammen 128 MB. Hver brikke er på 64 MB og har 32 bits databredde, og dermed 16 millioner adresser. Brikkene er laget av Micron og har delenummeret MT47H32M16BM. [Xilinx, UG454, s. 29, 33] Datablad skal egentlig være tilgjengelig ved å taste inn identifikasjonen på pakken (D9GMF) eller delenummeret (MT47H32M16BM) i FBGA-dekoderen på www.micron.com, men det har ikke gitt resultater i løpet av dette prosjektet.

Brikkene kan brukes med en klokkefrekvens på maksimalt 133 MHz [Xilinx, UG454, s. 33]. For at MicroBlaze-prosessoren og RAM-brikkene skal være synkrone, må frekvensene vi velger være delelige med $2^n$. I dette prosjektet drives de via FPGA-en med en klokkefrekvens på 125 MHz, som er kortets klokkefrekvens, mens MicroBlaze-prosessoren går på 62,5 MHz, som er halvparten. Hvis man ikke trenger RAM-brikkene, kan MicroBlaze-prosessoren gå på over 90 MHz på dette utviklingskortet.

Hastigheten på hele designet de dermed styrt av minnebrikkenes hastighet. Hvis man designer et eget kretskort for FPGA-en og velger raskere minnebrikker (200 MHz eller 266 MHz), kan man ha maksimal frekvens på MicroBlaze-prosessoren (over 90 MHz) og det dobbelte på minnebrikkene (over 180 MHz). FPGA-en som er brukt i prosjektet (Xilinx Spartan-3A DSP 1800) kan brukes med klokkefrekvenser på opptil 250 MHz [Xilinx, UG331], men systemet som helhet vil ikke kunne takle det, blant annet på grunn av MicroBlaze-prosessoren.

#### *3.9.7.1 MPMC*

Internt på FPGA-en brukes IP-modulen Multi-Port Memory Controller (MPMC) fra Xilinx for å lese og skrive til DDR, DDR2 og SDRAM [Xilinx, DS643]. IP-modulen er tilgjengelig i Xilinx EDK, og kan legges til designet manuelt eller med Wizard-en Base System Builder.

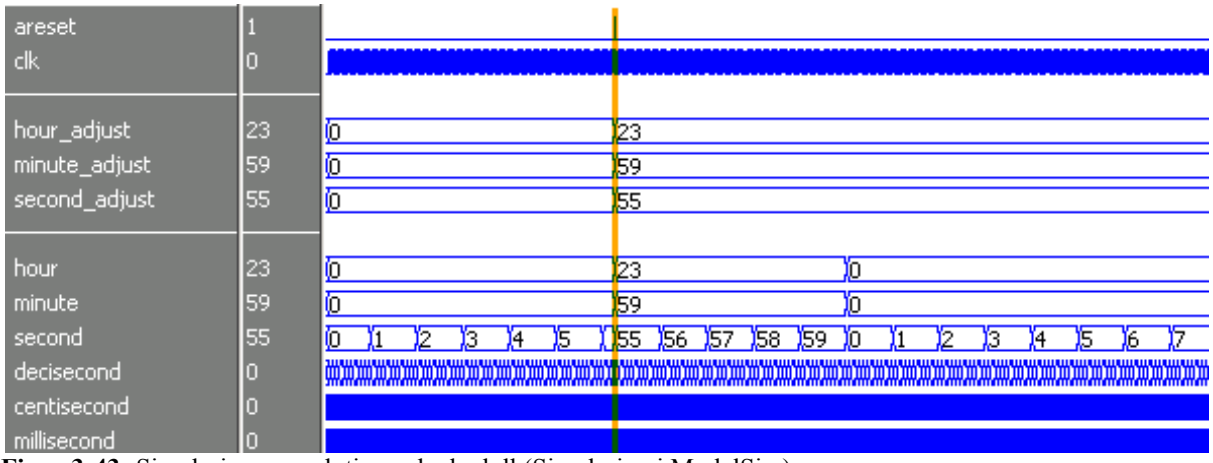
#### 3.9.7.2 Hurtigminne

Fordi lesing og skriving til DDR-RAM over PLB-bussen tar tid, brukes internt hurtigminne (eng: cache) for data som brukes ofte. Hurtigminnet består av BRAM-blokker på FPGA-en, så størrelsen på hurtigminnet kan konfigureres. For å bruke hurtigminne må dette gjøres tilgjengelig ved konfigurering av MicroBlaze-prosessoren (se kapittel 3.9.1) og i C-koden.

### 3.9.8 Sanntidsklokke – real_time_clock.vhdl

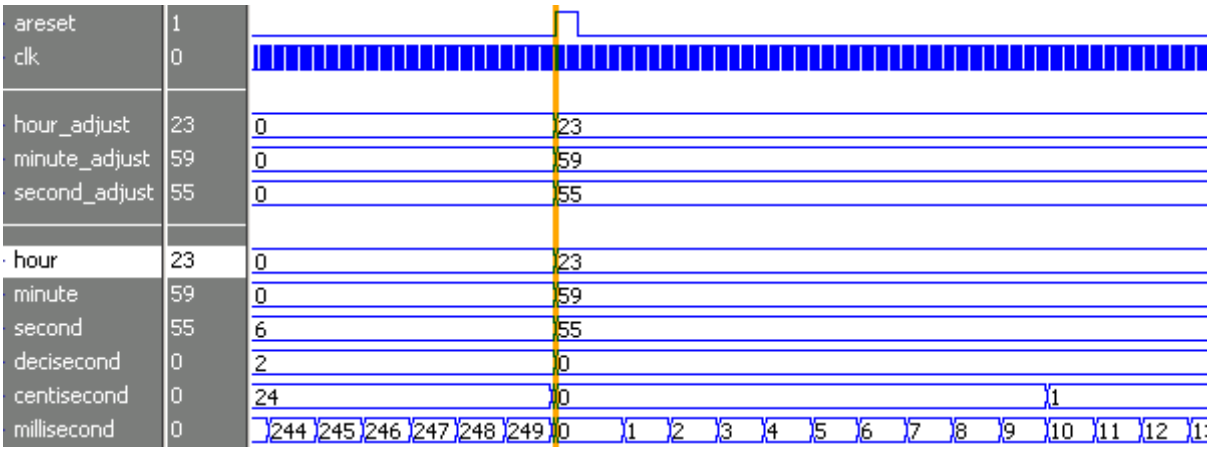
For å kunne tidsstemple alle UDP-pakkene som sendes fra FPGA-en til PC-en trenger vi en sanntidsklokke (eng: Real Time Clock – RTC) som holder kontroll på timer, minutter og sekunder. I mange systemer brukes en ekstern IC for dette. Fordelen med å bruke en ekstern IC er at den gjerne er koblet til en egen spenningskilde slik at den kan gå og være korrekt mens utstyret ikke er i bruk. Det er ikke implementert en slik IC på kretskortet, så vi måtte velge en annen løsning.

En mulighet kunne være å bruke en intern timer i prosessorsystemet og la prosessoren telle opp antall klokkepulser (ticks) og bruke verdien til å beregne hva tiden er. For å avlaste prosessoren har vi i stedet laget en egen enhet i VHDL for å holde kontroll på tiden. Denne kan brukes både selvstendig og på PLB-bussen.




**Figur 3-43:** Simulering av real_time_clock.vhdl (Simulering i ModelSim)

Sanntidsklokken består av en prosess som deler klokkefrekvensen inn til enheten (PLB-klokken ved tilkobling til PLB-bussen) med til en klokkefrekvens på 1 kHz. Denne brukes for å oppdatere millisecond-telleren som videre oppdaterer centisecond-telleren og så videre. Når enheten nullstilles lastes verdiene som ligger i *_adjust-registrene inn tellerne for time, minutt og sekund, mens de øvrige tellerne settes til null, som vist i figur 3-44.




**Figur 3-44:** Ved justering av klokken oppdateres registrene for time, minutt og sekund, mens resten nullstilles (Simulering i ModelSim).

### 3.9.9 Grensesnitt mot prosessoren – module_interface.vhdl

I stedet for å la alle enheter som er skrevet i VHDL ha et eget grensesnitt mot PLB-bussen for overvåking og konfigurering har vi laget en egen modul for dette. Det gjør at VHDL-modulene også kan brukes uten at et prosessorsystem er implementert, i tillegg til at systemet totalt sett blir mindre komplekst.

Enheten har en egen avbruddskontroller som samler alle avbruddslinjer fra VHDL-modulene til ett avbruddssignal som ledes videre til prosessorsystemets avbruddskontroller (se kapittel 3.9.3). Den integrerte avbruddskontrolleren kan ta imot inntil 32 avbrudd.

Vi har valgt å implementere avbruddskontrolleren uten "Device ISC" som gir mer avansert funksjonalitet som aktivering og prioritering av hvert enkelt avbrudd [Xilinx, DS516, s. 3-4]. Dette gjør at systemet bruker mindre ressurser på FPGA-en, men vi ser i ettertid at den gjerne kunne vært inkludert. Dersom prioritering av avbruddene trengs, kan dette utføres av c-koden ved å sjekke visse avbrudd før andre.

Enheten har i tillegg en rekke registre som brukes til kommunikasjon mellom VHDL-modulene og prosessoren. Registrene som heter ip2bus_reg## kan VHDL-modulene skrive til, mens prosessoren bare kan lese dem. Disse brukes for å hente inn verdier fra VHDL-modulene. For registrene som heter bus2ip_reg## har prosessoren skrive- og lesetilgang, mens VHDL-modulene kun kan lese dem. Disse brukes for å konfigurere VHDL-modulene. Det er opprettet 50 registre av hver type, men de som ikke brukes blir optimalisert bort under syntetisering.

| BUS2IP | | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reg00 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| reg01 | LED | | | | | | | | | | | | | | | | | | | | | | | | | | | Led5 | Led4 | Led3 | Led2 | Led1 | Led0 |
| reg02 | HAD | | | | | | | | | | | | | | | | | | | | | | | | | | | | | enable | mode | | reset |
| reg03 | MUX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | selector | |
| reg04 | RWS | | | | | | | | | | | | | | | | | | | | | | start | | | | | | | | | | |
| reg05 | RWS | | | | | | | | | | | | | | | | | | | | | | stop | | | | | | | | | | |
| reg06 | RWS | | | | | | | | | | | | | | | | | | | | | | | | | | step | | | | | | |
| reg07 | SMPL | | | | | | | | | | | | | | | | | | | | | | | | | | interval | | | | | | |
| reg08 | ENV | | | | | | | | | | | | | | | | | | | | | | | | | | samples_pr_period | | | | | | |
| reg09 | DAH | delay | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| reg10 | DAH | hold | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Figur 3-45:** Oversikt over BUS2IP-registerne.

### 3.9.10 Øvrige moduler i XPS

I tillegg til modulene som tidligere er nevnt i dette kapittelet er følgende moduler implementert:

- **Feilsøkingsmodul** (MicroBlaze Debug Module (mdm)) brukes for å styre prosessoren og lese ut registerverdier under feilsøking i c-kode på FPGA-en.
- **SPI Flash-RAM kontroller** (XPS Serial Peripheral Interface (xps_spi)) som kan brukes for å lese og skrive verdier til Flash-minnet på testkortet [Xilinx, UG454, s. 12]. Denne er ikke i bruk, men kan for eksempel brukes til å lagre konfigurasjonsinnstillinger som skal brukes igjen neste gang systemet skal brukes.
- **Klokkegenerator** (Clock generator (clock_generator)) genererer klokkefrekvenser for prosessoren, DDR-RAM og AD-omformeren fra klokkefrekvensen på 125 MHz til kortet. Den benytter Digital Clock Managers (DCM) på FPGA-en. Den kan med fordel byttes ut med en DCM som opprettes i ISE, fordi det gir bedre kontroll, i tillegg til at den da kan simuleres sammen med resten av logikken.
- **Nullstillingsmodul** (Processor System Reset Module (proc_sys_reset)) synkroniserer asynkrone nullstillingssignaler og sørger for at klokkesignalene fra DCM-en er stabile før resten av prosessorsystemet startes opp.

# 4. Programvare

4.1 TCP/IP ..... 120

*4.1.1 Internettlaget - IP (Internet Protocol) ..... 121*

*4.1.2 Transportlaget - UDP ..... 123*

*4.1.3 Applikasjonslaget - Innholdet i UDP-pakkene ..... 124*

4.1.3.1 Kommandosett og pakkestruktur for Merdøye MKII ..... 125

4.1.3.2 Kommandosett og pakkestruktur for SIMRAD EK 500 ..... 130

4.1.3.3 Kommandosett for HADAS 2010 ..... 134

4.1.3.4 Pakkestruktur for HADAS 2010 ..... 137

4.2 PROGRAMKODE C ..... 138

*4.2.1 lightweight IP (lwIP) ..... 139*

4.3 DATAINNSAMLINGSPROGRAM FOR PC ..... 142

*4.3.1 Filformat - EYM ..... 144*

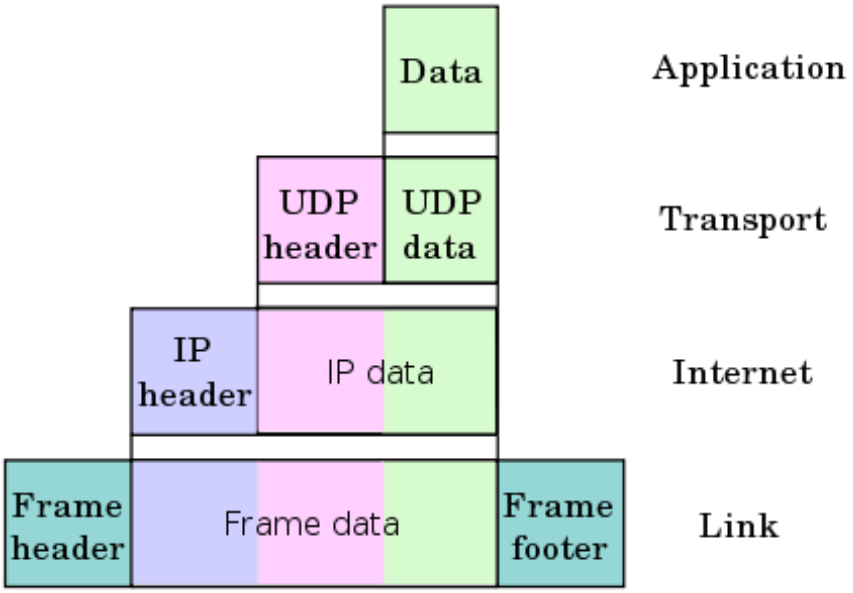
I prosjektet er det skrevet et program som installeres på en datamaskin for å sende kommandoer til ekkoloddsystemet og motta og lagre måleverdiene ekkoloddet sender tilbake. I FPGA-en i ekkoloddsystemet er det implementert en MicroBlaze-prosessor (kapittel 3.9.1) der et program mottar kommandoene, styrer ekkoloddet og sender måleverdier til datamaskinen. Kommunikasjonen mellom datamaskinen og ekkoloddsystemet skjer ved hjelp av UDP-pakker over Ethernet.

## 4.1 TCP/IP

| | |
|---|---|
| **Applikasjonslaget** | HTTP, FTP, IMAP, POP, DHCP, DNS, SNMP |
| **Transportlaget** | TCP, UDP |
| **Internettlaget** | IP, ICMP, IGMP |
| **Linklaget** | ARP, MAC(Ethernet) |

**Tabell 4-1:** TCP/IP-modellen med noen eksempler på protokoller som hører til hvert lag.

For å kommunisere over Ethernet brukes vanligvis protokollene i TCP/IP-stakken. Tabell 4-1 viser lagene i TCP/IP-modellen og noen av protokollene som hører til de forskjellige lagene. I dette prosjektet bruker vi UDP/IP for å sende data og kommandoer mellom datamaskinen og ekkoloddsystemet, mens IP-stakken lightweight IP (lwIP) (kapittel 4.2.1) tar seg av administrative protokoller som ICMP og ARP. Modulene som tar seg av kommunikasjonen på linklaget er beskrevet i kapittel 3.9.6.1.




**Figur 4-1:** Innpakking av data i en UDP/IP-pakke [Wikipedia, UDP encapsulation]

Når data skal sendes fra en applikasjon med UDP/IP, pakkes dataene inn med en ny ramme for hvert lag den går nedover lagene i TCP/IP-modellen. Når Ethernet-pakken mottas av den andre enheten fjernes lagene i motsatt rekkefølge, slik at dataene er tilgjengelig på applikasjonslaget. Ethernet-pakken på linklaget er beskrevet i 3.9.6.1.

### 4.1.1 Internettlaget - IP (Internet Protocol)

IP brukes i dag i versjon 4 (IPv4) og versjon 6 (IPv6), men det er så store forskjeller mellom dem at de ikke er kompatible. IPv6 er foreløpig så ny at det er mye nettverksutstyr som ikke støtter den. Selv om ekkoloddsystemet kan støtte IPv6, har vi valgt å bruke IPv4 for å unngå kompatibilitetsproblemer.

| bit offset | 0–3 | 4–7 | 8–15 | 16–18 | 19–31 |
|---|---|---|---|---|---|
| 0 | Version | Header length | Differentiated Services | Total Length | |
| 32 | Identification | | | Flags | Fragment Offset |
| 64 | Time to Live | | Protocol | Header Checksum | |
| 96 | Source Address | | | | |
| 128 | Destination Address | | | | |
| 160 | Options | | | | |
| 160 or 192+ | Data | | | | |

**Figur 4-2:** IP-pakke [Wikipedia, IPv4]

Dataene fra laget over pakkes inn i en IP-pakke som vist i figur 4-2. Pakken består av mange felt, men kun de med størst betydning i dette prosjektet er beskrevet her (se [Wikipedia, IPv4] for beskrivelse av de øvrige feltene).

A. **Total Length, Identification, Flags og Fragment Offset** brukes for å dele opp data som er større enn det nettverket tillater å overføre. Denne størrelsen kalles MTU (Maximum transmission unit) og er vanligvis 1500 byte for Ethernet [Wikipedia, MTU]. Det er mulig å sende større pakker med jumbo frames [Wikipedia, Jumbo frames], men støtten for dette er lite utbredt. Hvis dataene er større enn MTU, deles de opp i inntil 1480 byte pr pakke og sendes som flere pakker etter hverandre. Når siste pakke er mottatt av mottageren settes dataene sammen igjen og behandles videre av neste lag i stakken [Wikipedia, IPv4].
B. **Protocol** brukes for å identifisere hva slags protokoll som brukes på dataene som er i datafeltet. Protokollnummer 1 brukes for ICMP, mens UDP har protokollnummer 17 [Wikipedia, IPv4].
C. **Source Address og Destination Address** brukes for å identifisere henholdsvis avsender og mottaker på IP-nivå. IP-adressene gjør det mulig å sende pakker mellom flere nettverk, blant annet over Internett. Adressen til hver enhet på nettverket kan enten fastsettes i enheten (automatisk eller manuelt) eller tildeles fra en DHCP-server (Dynamic Host Configuration Protocol).

### 4.1.2 Transportlaget - UDP

UDP (User Datagram Protocol) er en av protokollene som kan brukes for å sende meldinger over Ethernet. I motsetning til TCP kan UDP-pakker sendes uten at det er opprettet kontakt med mottakeren først. Det gjør at UDP er enklere å implementere, og den er rask fordi den ikke bruker tid på å holde forbindelsen ved like. Ulempen med at UDP er forbindelsesløs er at det ikke er noen garanti for rekkefølgen pakkene kommer frem i, eller at de kommer frem i det hele tatt [Wikipedia, UDP].

| bits | 0 - 15 | 16 - 31 |
|---|---|---|
| 0 | Source Port | Destination Port |
| 32 | Length | Checksum |
| 64 | Data | |

**Figur 4-3:** UDP-pakke [Wikipedia, UDP]

Data som sendes med UDP pakkes inn i en pakke med fire felt:

- **Source Port og Destination Port** angir hvilke portnummer pakken er sendt fra og hvor den skal sendes. Det finnes 65535 porter for UDP som programmer kan binde seg til. Dersom dataene som mottas har riktig IP-adresse og portnummer, vil dataene bli gjort tilgjenglige for programmet. På denne måten kan flere programmer som venter på UDP-pakker kjøre samtidig.
- **Length** innholder størrelsen på datafeltet.
- **Checksum** brukes for å kontrollere at innholdet i datafeltet ikke er endret etter at pakken ble sendt.

### 4.1.3 Applikasjonslaget - Innholdet i UDP-pakkene

For å styre ekkoloddet må kommandoer sendes fra PC-en til ekkoloddsystemet. Det bør også sendes en respons fra ekkoloddsystemet på at kommandoen er mottatt og utført, særlig fordi systemet er basert på UDP som ikke garanterer at pakkene kommer fram. Kommandoene må selvfølgelig være kjent av både datainnsamlingsprogrammet og ekkoloddsystemet for at det skal fungere.

I UDP-pakkene med ekkoloddata må det også være en fast struktur på innholdet så dataprogrammene som skal tolke dem er i stand til å finne fram til informasjonen. Hver enkelt UDP-pakke må kunne identifiseres og sorteres av datainnsamlingsprogrammet fordi de kan bruke ulik tid på å komme fram gjennom nettverket. For å løse dette er det fornuftig å legge inn et hode (eng. header) som inneholder denne informasjonen før meldingskroppen med datainnholdet.

Det er to ulike kommandosett som har vært i bruk i sonargruppa ved UDP-kommunikasjon med ekkoloddsystemer. Det ene er kommandosettet som benyttes til den kommersielle forskningsekkoloddet SIMRAD EK 500 [Mork 2000], mens det andre er et kommandosett som er utviklet ved Elektronikklaboratoriet ved Fysisk institutt for ekkoloddsystemet Merdøye MKII.

For at man ikke skal ha alt for mange standarder i bruk på samme institutt, var det ønskelig å følge en av disse standardene i dette prosjektet. Ved å standardisere pakkestrukturene slik at datainnsamlingsprogrammer og ekkolodd kan brukes om hverandre, blir det lettere for framtidige prosjekter å videreutvikle tidligere prosjekter i stedet for å lage alt på nytt.

#### 4.1.3.1 Kommandosett og pakkestruktur for Merdøye MKII

Merdøye MKII er et ekkoloddsystem for overvåking av oppdrettsmerder, og systemet har kapasitet til å overvåke flere merder samtidig. Mikrokontrolleren som styrer systemet er programmert i C, mens datainnsamlingsprogrammet er laget i LabView.

Kommandoene som sendes fra PC-en til ekkoloddsystemet er beskrevet i tabell 4-2, mens pakkestrukturen for ekkoloddata er beskrevet i tabell 4-3.

**Kommandosett for Merdøye MKII**

| Merdøye MKII | Kommando | Respons | Forklaring |
|---|---|---|---|
| **Koble til enheten** | b | 'Ethernet broadcast, START' | Enheten som sendte denne kommandoen får kontrollen over Merdøye MKII. Denne enhetens IP- og MAC-adresse settes. |
| **Konfigurere enheten for broadcasting av UDP-pakker** | broadcast | 'Ethernet broadcast' | IP- og MAC-adresse for Ethernet broadcast settes. Meldingene er deretter tilgjengelig for alle enheter på LAN. |
| **Sjekke statusregisteret** | status | 'Status' + S<br>*S er innholdet i statusregisteret (1 byte).* | Bit 1-6: Om svingeren (1-6) er aktiv (1) eller ikke (0).<br>Bit 7: Pingraten er 1 pr. sek. (1) eller 3 pr sek (0).<br>Bit 8: Broadcast er på (1) eller av (0). |
| **Starte og stoppe ekkoloddet** | srX<br>*X er svingernummer 1-6.* | 'OK'<br>*Deretter ekkoloddata som beskrevet i tabell 4-3.*<br>*Ved feil:*<br>'Command not valid'<br>'Argument not valid'<br>'Error' | Veksler svinger nummer X av eller på. |
| **Sjekke Ethernet-kontrollerens versjonsnummer.** | v | 'ver=' + versjonsnummer | |
| **Sette skuddraten** | p | 'Pingrate satt til X pr.sek'<br>X er 1 eller 3. | Veksler skuddraten mellom 1 og 3 skudd pr sekund. |
| **Stille sanntidsklokken** | RHHMMSS | 'Clock is set' | |
| **Motta hjelpemelding** | h | 'Ethernet controller, ELAB RD. Send srX to set transducer X. X is a number (1-6)' | |
| **Teste systemet med ett ping** | c | *Mottar én pakke med sampler.* | Tester svinger nummer 1. |
| **Gi en triggerpuls** | trig | 'Trigger pulse' | |

**Tabell 4-2** Kommandoer for Merdøye MKII. Alle kommandoer avsluttes med carriage return (\r i C og andre lignende programmeringsspråk).

**Styring av Merdøye MKII**

For å bruke Merdøye MKII må man først sende kommandoen for å koble til enheten (’b\r’) og vente på respons (‘Ethernet broadcast, START’). Deretter sender man for eksempel ’sr1\r’ for å starte svinger nummer 1. For å stoppe den igjen, sender man samme kommando en gang til (’sr1\r’).

Legg merke til at kommandoene avsluttes med ”carriage return” (CR) som er \r i C og flere andre lignende programmeringsspråk (for eksempel Python). Hvis man bruker et annet programmeringsspråk, må man bruke riktig symbol for ”carriage return” for dette programmeringsspråket.

**Styrker og svakheter ved kommandosettet for Merdøye MKII**

Fordelen med kommandosettet for Merdøye MKII er at det har en enkel struktur som kun inneholder de kommandoene man trenger for å styre ekkoloddsystemet. Ved hjelp av tabell 4-2 og forklaringen i avsnittet over har man alt man trenger for å kunne styre ekkoloddsystemet. Siden kommandoene kun er bokstaver eller korte ord, krever det lite tekstprosessering av ekkoloddsystemet og datainnsamlingsprogrammet.

Kommandosettet har en del svakheter som burde rettes opp hvis det skal brukes som en standard:

- Kommandoene som sendes er både enkle bokstaver og hele ord. Siden det ikke er noen fast struktur på kommandoene, kan det på sikt bli vanskelig å utvide, fordi man går tom for bokstaver og ord med logisk betydning.
- Kommandoene er skrevet med ASCII-tegn, men de er likevel ikke umiddelbart forståelige for mennesker.
- Kommandoene som tar variabler skrevet i ett ord (for eksempel srX og RHHMMSS). Ved å skille kommando og variabel er det enklere for mennesker å lese det, og programmene som skal tolke det kan lett skille kommando og argument ved å splitte på mellomrom, likhetstegn eller lignende.
- Responsen fra ekkoloddsystemet når en kommando er mottatt er noen ganger på norsk og andre ganger på engelsk.
- Responsen er ikke alltid så lett å forstå, som for eksempel ”Ethernet broadcast” og ”Ethernet broadcast, START”.

**Pakkestruktur for Merdøye MKII**

Headeren i UDP-pakkene med ekkoloddata tar utgangspunkt i W-telegram som SIMRAD EK 500 bruker, men de er ikke identiske og kan derfor ikke brukes om hverandre.

| **Bytenummer** | | **Eksempel** | **Forklaring** |
|---|---|---|---|
| 1 | W | W | Telegramtype W [SIMRAD, 1992] |
| 2 | # | 1 | Svingernummer |
| 3 | , | , | Separator |
| 4 | # | 1 | Time (X0) |
| 5 | # | 3 | Time (0X) |
| 6 | # | 5 | Minutt (X0) |
| 7 | # | 3 | Minutt (0X) |
| 8 | # | 2 | Sekund (X0) |
| 9 | # | 8 | Sekund (0X) |
| 10 | 0 | 0 | Separator |
| 11 | # | 0 | Antall byte med sampler i pakken (X000) |
| 12 | # | 7 | Antall byte med sampler i pakken (0X00) |
| 13 | # | 2 | Antall byte med sampler i pakken (00X0) |
| 14 | # | 0 | Antall byte med sampler i pakken (000X) |
| 15 | # | 0 | Holdt av for framtidig bruk |
| 16 | # | 0 | Holdt av for framtidig bruk |
| 17 | # | 0 | Holdt av for framtidig bruk |
| 18 | # | 0 | Holdt av for framtidig bruk |
| 19 | # | 0 | Holdt av for framtidig bruk |
| 20 | # | 0 | Holdt av for framtidig bruk |
| 21 – | | | Ekkoloddata. 8-bits binær verdi. (Ekkoloddataene fra Merdøye MKII er 8-bits ord (0-255) fra en 8-bits AD-omformer i ATmega64 med Vref = 4,096V). |

**Tabell 4-3** Pakkestruktur for Merdøye MKII. # erstattes med tall i ASCII-format.

**Styrker og svakheter ved pakkestrukturen for Merdøye MKII**

Pakkestrukturen for Merdøye MKII gir mye av den informasjonen som trengs for å identifisere dataene som sendes, med svingernummer, tidspunkt og antall byte med data.

Den har likevel en del svakheter:

- Som separator er det et sted brukt komma og et annet sted tallet null. Bruk av komma gjør det enklere for mennesker å lese telegrammet, og det er også dette som brukes i SIMRAD EK 500.
- Det er ingen separator mellom tabellstørrelsen og dataene. Hvis det hadde vært et komma her også, ville det vært lettere for mennesker å lese pakkene. I tillegg kunne dataprogrammet splittet den mottatte pakken på komma for å skille feltene fra hverandre, i stedet for å lese ut et oppgitt byteintervall.
- Tidsangivelsen mangler hundredeler, så hvis man sender flere ping hvert sekund mottar man flere pakker med identisk header. Hvis klokken i systemet ikke tilbyr hundredeler, kunne man brukt en pakketeller som nullstilles hvert sekund, så rekkefølgen på pakkene kan avgjøres av mottakerprogrammet.
- Verdien for tabellstørrelse er statisk i dagens versjon av Merdøye MKII, og den viser feil verdi i forhold til hvor mye data som sendes. Hvis denne pakkestrukturen skal brukes, bør verdien genereres for hver pakke som sendes. Hvis ikke har den ingen nytteverdi.
- Hvis ekkoloddataene må sendes i flere UDP-pakker fordi det ikke er plass til alle dataene fra et ping i én standard UDP-pakke (maks 1500 byte totalt), finnes det ikke noe felt som viser rekkefølgen på pakkene. Dette er ikke noe problem hvis man kun sender én stor UDP-pakke som fragmenteres på IP-nivå av IP-stakken i ekkoloddsystemet og settes sammen igjen av IP-stakken i PC-en. Fordi det ikke finnes ferdiglagde IP-stakker med IP-fragmentering til alle mikrokontrollere og prosessorer som kan være aktuelle i framtiden, er det fornuftig å legge til et felt for oppdeling av ekkoloddata.

### 4.1.3.2 Kommandosett og pakkestruktur for SIMRAD EK 500

Det er laget flere datainnsamlingsprogrammer for SIMRAD EK 500, blant annet BI500 (UNIX), SIMRAD EP 500 (MS-DOS) og Echo Receiver (Windows). Den siste ble utviklet av Olav Håkon Mork som et hovedfagsprosjekt ved sonargruppa [Mork, 2000]. Ved å basere kommandoene og pakkestrukturen på det som er brukt for SIMRAD EK 500, kan altså denne programvaren brukes til å styre ekkoloddet.

**Kommandosett for SIMRAD EK 500**

Innholdet i UDP-pakkene til og fra SIMRAD EK 500 er bygd opp som telegrammer. Det er egentlig laget for seriell kommunikasjon (RS232), men er også utvidet for Ethernet. De to første tegnene brukes til å skille de ulike telegramtypene fra hverandre. For eksempel er W1 en telegramtype som inneholder ekkoloddata, mens PE brukes for telegrammer som skal endre et av ekkoloddsystemets parametre, og fungerer derfor som en kommando. Telegrammene for ekkoloddata har en sammensatt struktur, og kommandotelegrammene inneholder en mappesti for å sette riktig parameter. Alt dette er nøyaktig beskrevet i [SIMRAD, 1992].

Eksempel på kommandoene som trengs for å koble seg til ekkoloddsystemet:

PE, 10385212,/ETHERNET COM. MENU/Remote ETH Addr.=08:13:14:F3:98:10\r

PE, 10385336,/ETHERNET COM. MENU/Remote IP Addr.=131.051.171.087\r

**Styrker og svakheter ved kommandosettet for SIMRAD EK 500**

Den store fordelen med å bruke samme kommandosett som SIMRAD EK 500 er at datainnsamlingsprogrammer for systemene kan brukes om hverandre. SIMRAD EK 500 er ikke lenger i produksjon, men brukes fortsatt av flere forskningsinstitusjoner. Ved å velge kommandosettet for SIMRAD EK 500 ser man altså bakover i stedet for framover.

Telegrammene har en sammensatt struktur som gjør det enkelt å utvide med flere kommandoer på en logisk og oversiktlig måte hvis det er nødvendig i framtiden. Det er

mange kommandoer som er implementert for SIMRAD EK 500, så sannsynligvis er alle kommandoer man trenger tilgjengelig. Eventuelt kan man også implementere flere funksjoner etter samme struktur for nye systemer, men de er da ikke kompatible med SIMRAD EK 500.

Den komplekse kommandostrukturen er også en ulempe da den krever en del tekstprosessering av enhetene for å tolke kommandoene, og det kan være utfordrende å finne fram til hvilken kommando som skal sendes.

Programmet Echo Receiver kan ikke sende flere kommandoer enn det programmet selv konfigurerer via GUI, da prf-filen overskrives av programmet. Hvis man ønsker å bruke programmet for andre ekkolodd basert på et utvalg av EK 500-kommandoene, må man skrive om programmet for å støtte det, og det er ikke ønskelig.

**Pakkestruktur for SIMRAD EK 500's W-telegrammer**

Eksempel på mottatt telegram med ekkoloddata:

W1,10441352,000214540003015608A6AB12

| Bytenummer | | Eksempel | Forklaring |
|---|---|---|---|
| 1 | W | W | Telegramtype W |
| 2 | X | 1 | Svingernummer |
| 3 | , | , | Separator |
| 4 | H | 1 | Time (X0) |
| 5 | H | 0 | Time (0X) |
| 6 | M | 4 | Minutt (X0) |
| 7 | M | 4 | Minutt (0X) |
| 8 | S | 1 | Sekund (X0) |
| 9 | S | 3 | Sekund (0X) |
| 10 | C | 5 | Hundredel (X0) |
| 11 | C | 2 | Hundredel (0X) |
| 12 | , | , | Separator |
| 13 | B | 0 | Blokknummer (X000) |
| 14 | B | 0 | Blokknummer (0X00) |
| 15 | B | 0 | Blokknummer (00X0) |
| 16 | B | 2 | Blokknummer (000X) |
| 17 | F | 1 | Fragment offset i byte (X000) |
| 18 | F | 4 | Fragment offset i byte (0X00) |
| 19 | F | 5 | Fragment offset i byte (00X0) |
| 20 | F | 4 | Fragment offset i byte (000X) |
| 21 | T | 0 | Tabellstørrelse (X000) |
| 22 | T | 3 | Tabellstørrelse (0X00) |
| 23 | T | 0 | Tabellstørrelse (00X0) |
| 24 | T | 0 | Tabellstørrelse (000X) |
| 25 – | | | Ekkoloddata (16 bit pr verdi). Effekt. |

**Tabell 4-4:** [SIMRAD, 1992]

**Styrker og svakheter ved pakkestrukturen for SIMRAD EK 500**

Pakkestrukturen for SIMRAD EK 500 gir all den informasjonen som trengs for å identifisere dataene som sendes, med svingernummer, tidspunkt og antall byte med data, i tillegg til fragment offset og blokknummer som gjør det mulig å dele opp data i flere pakker på systemer som ikke støtter IP-fragmentering (se kapittel 4.1.1).

Det er ingen separator mellom blokknummer, fragment offset, tabellstørrelse og ekkoloddata. Hvis det hadde vært et komma her også, ville det vært lettere for

mennesker å lese pakkene, i tillegg til at dataprogrammet kunne splittet den mottatte pakken på komma for å skille informasjonen fra hverandre i stedet for å lese ut et oppgitt byteintervall.

### 4.1.3.3 Kommandosett for HADAS 2010

Kommandostrukturen for SIMRAD EK 500 er beskrevet i kapittel 4.1.3.2. Kommandosettet er til et ekkoloddsystem som er tatt ut av produksjon, så selv om en del forskningsinstitusjoner bruker det fortsatt, er det ikke et kommandosett for framtiden. Støtten for kommandosettet i programmet Echo Receiver er også så begrenset at systemene ikke kan brukes om hverandre. Vi velger derfor å ikke bruke kommandosettet for SIMRAD EK 500.

Kommandosettet for Merdøye MKII er beskrevet i kapittel 4.1.3.1. Det er et enkelt kommandosett med få kommandoer, men det er tilstrekkelig til å kunne styre systemets hovedfunksjoner. Som vi har påpekt i kapittel 4.1.3.1 har det en rekke ulemper, men vi velger å implementere kommandosettet fordi det da er kompatibelt med programvare og ekkolodd for Merdøye MKII.

Responsen fra ekkoloddsystemet ved bruk av Merdøye MKII-kommandoer er endret i forhold til hvordan det er implementert i Merdøye MKII. Ved bruk av Merdøye MKII-kommandoer brukes automatisk pakkestrukturen for Merdøye MKII. Legg merke til at dataene som da sendes er 16-bit-verdier i stedet for 8-bit som er standard for Merdøye.

Fordi kommandosettet for Merdøye MKII ikke er tilstrekkelig til å kunne styre alle funksjoner på det nye ekkoloddsystemet, er det i tillegg implementere et eget kommandosett. Merdøye MKII-kommandoer kan derfor brukes til å styre enkle funksjoner, mens HADAS 2010-kommandoene gir full kontroll over systemet.

Det nye kommandosettet for HADAS 2010 er laget til dette prosjektet og brukes ikke av andre systemer. Dermed er det ingen eksisterende programmer eller ekkolodd som kan brukes, men denne kompatibiliteten er ivaretatt ved at ekkoloddsystemet også støtter kommandosettet for Merdøye MKII.

Kommandosettet er basert på erfaringene fra kommandosettene for Merdøye MKII og SIMRAD EK 500. Strukturen er enkel å forstå for både mennesker og maskiner og kan enkelt utvides til nye anvendelser. Kommandoene har en logisk oppbygging, og det

brukes hele ord skrevet i ASCII. Fordi systemet skal brukes i forskningsanvendelser er det en fordel at kommandoene ikke er binære koder, men lesbare, slik at et er lett å gjøre de nødvendige innstillingene av systemet.

Den største ulempen med kommandosettet er at det er tilpasset ekkoloddets systemarkitektur, så kommandoene kan ikke uten videre brukes på andre systemer. Prinsippene som ligger til grunn er derimot generelle. Ved å lage nye systemer basert på samme struktur, kan de enkelt utvides og brukes om hverandre hvis det er ønskelig. Kommandoene har en logisk oppbygning, men for å endre avanserte parametre kreves det at man har kjennskap til systemarkitekturen.

Det nye kommandosettet for HADAS 2010 har kun to kommandoer, set og get. Kommandoen etterfølges av et variabelnavn som er basert på strukturen hw_soc_st i HADAS_options.h. Ved set-kommandoer etterfølges variabelnavnet av et er-lik-tegn (=), mens ved get-kommandoer blir dette eventuelt ignorert av systemet. Etter er-lik-tegnet kommer verdien variabelen skal settes til.

Kommandoene sendes som ASCII-tegn og avsluttes med et nullterminering stegn. Nullterminering stegnet er valgt fordi det brukes for å avslutte strenger i C og lignende språk, og skrives som \0. Hvis for eksempel sanntidsklokken i systemet skal stilles sendes: set rtc_0.adjust=124316\0

| **Kommando** | **Merdøye MKII** | **HADAS 2010**<br>*(Alle kommandoer avsluttes med \0)* |
|---|---|---|
| **Koble til enheten** | b\r | set udp.send_to_ip=connect |
| **Konfigurere enheten for kringkasting av UDP-pakker** | broadcast\r | set udp.send_to_ip=broadcast |
| **Sjekke statusregisteret** | status\r | get had.status<br><br>*Ulike meldinger avhengig av hvilket statusregister som skal sjekkes, for eksempel:*<br>get ping_rate.period<br>get rtc_0.clock |
| **Starte ekkoloddet** | srX\r | set had.enable=on |
| **Stoppe ekkoloddet** | srX\r | set had.enable=off |
| **Sette pingraten** | p\r *(skifter mellom 1 og 3 pr sekund)* | set ping_rate.period=*heltallsverdi* |
| **Stille sanntidsklokken** | RHHMMSS\r | set rtc_0. adjust=hhmmss *(heltallsverdi)*<br>*Tideler, hundredeler og tusendeler nulles ut.* |
| **Motta hjelpemelding** | h\r | *Foreløpig ikke implementert* |
| **Teste systemet med ett ping** | c\r | set had.enable=test |

**Tabell 4-5:** Kommandoer for Merdøye MKII og HADAS 2010 som kan brukes med systemet. Øvrige HADAS 2010-kommandoer kan leses ut av HADAS_commands.h
(\r = "carriage return", \n = "new line"/"line feed", \0 = "Terminering av strenger", NA = ikke tilgjengelig, HHMMSS = tidspunkt i timer, minutter og sekunder.).

#### 4.1.3.4 Pakkestruktur for HADAS 2010

Pakkestrukturen for Merdøye MKII er beskrevet i kapittel 4.1.3.1. Som nevnt der er pakkestrukturen ikke god nok fordi det ikke er mulig å skille pakker som er sendt innen samme sekund, og det er ikke mulig å sende ekkoloddata fra samme ping i flere pakker hvis ikke IP-stakken støtter fragmentering av IP-pakker.

Fordi pakkestrukturen for Merdøye MKII er mangelfull bruker vi heller pakkestrukturen som denne egentlig er basert på, nemlig W-telegrammene for SIMRAD EK 500. W-telegrammene er godt egnet for overføring av ekkoloddata (se kapittel 4.1.3.2), men ekkoloddataene skal være effektverdier i henhold til spesifikasjonen for SIMRAD EK 500. Det betyr at ekkoloddet må trekke fra TVG og beregne effektverdier før de oversendes.

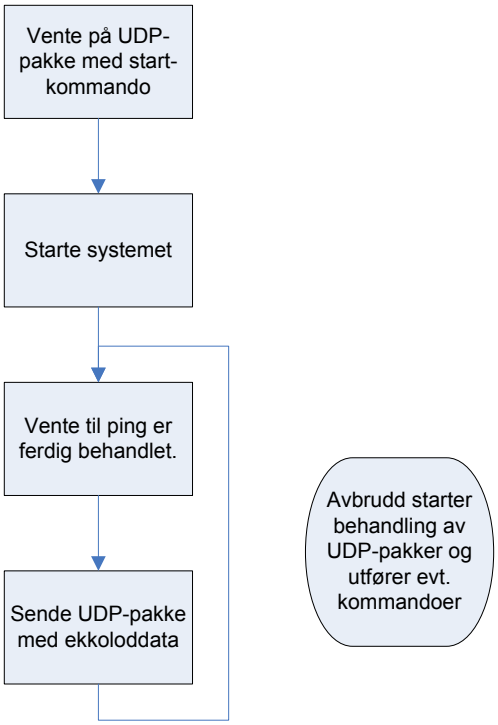
Ifølge førsteamanuensis Helge Balk er det bedre å motta og lagre rådata, da det gir bedre resultater ved etterjustering av signalene i programmer som Sonar 5. Pakkene fra det nye ekkoloddsystemet starter med bokstaven ”H” (for HADAS) for å skille dem fra SIMRAD-pakker, men er ellers identiske med W-telegrammene. Feltene for ”Fragment offset i byte” og ”Blokknummer” er fylt med nuller fordi pakkene fragmenteres på IP-nivå (se kapittel 4.1.1).

## 4.2 Programkode C

MicroBlaze prosessoren kan kjøre flere forskjellige operativsystemer. Et operativsystem sørger for et generelt og enkelt grensesnitt mot ressursene i systemet og fordeler dem mellom programmene som kjører. Slik kan operativsystemet la flere programmer kjøre samtidig uten at minneområder blir overskrevet eller at to programmer bruker samme enhet samtidig.

Det er utviklet flere distribusjoner av Linux for MicroBlaze, som [Petalogix, PetaLinux] og [Lynuxworks, Xilinx], og fra versjon 2.6.31 av Linux-kjernen er MicroBlaze-arkitekturen inkludert [Xilinx Open Source Wiki, MicroBlaze-Linux og kernelnewbies, Linux 2.6.31]. Flere andre større operativsystemer er også tilrettelagt for MicroBlaze-prosessoren [Xilinx, Third Party Embedded Solutions Providers]. Xilinx har utviklet et enkelt operativsystem, Xilkernel, som blant annet tilbyr kjøring av flere tråder, styring av delt minne og programvare-tidtakere [Xilinx, Xilkernel].

I dette prosjektet har vi valgt å ikke bruke noe operativsystem fordi programmet har en sekvensiell programflyt, slik at det ikke er nødvendig at flere programmer kjører samtidig for å få oppgavene utført i tide. I stedet brukes avbrudd for å varsle systemet når handlinger som avviker fra den vanlige programflyten skal utføres. Ved å skru av og på avbrudd forskjellige steder i koden har man dermed full kontroll på hvilke funksjoner som utføres til en hver tid, og kan prioritere enkelte handlinger foran andre. Programmet er optimalisert for hovedoppgaven, nemlig å sende UDP-pakker med sonardata, mens konfigurasjon av systemet skjer når systemet er ledig.




**Figur 4-4:** Programflyt (Tegnet i Microsoft Visio)

Fordi systemet er satt opp uten operativsystem gjør EDK automatisk biblioteket ”Standalone Board Support Package” [Xilinx, Standalone BSP] tilgjengelig. Biblioteket gir tilgang på prosessorrelaterte funksjoner som behandling av avbrudd og styring av cache.

### 4.2.1 lightweight IP (lwIP)

De fleste operativsystemer har innebygde TCP/IP-stakker for å sende og motta data over Ethernet. Fordi vi ikke bruker noe operativsystem må vi enten skrive en enkel IP-stakk selv for sending og mottak av UDP-pakker eller bruke en som er ferdiglaget. Fordelen med å bruke en ferdiglaget IP-stakk er at den er enklere å ta i bruk, og den fungerer som den skal i henhold til standardene den er skrevet for. En ferdiglaget IP-stakk gir gjerne også tilgang på flere protokoller og sørger for å svare på administrativ Ethernet-trafikk uten at brukeren behøver å konfigurere dette.

Det finnes flere IP-stakker for MicroBlaze-prosessoren, som for eksempel [Sourceforge, cipsuite] og [Treck, Xilinx]. Vi har valgt å bruk lightweight IP (lwIP)

[Nongnu, lwIP], som følger med som et programvarebibliotek i Xilinx SDK [Xilinx, lwIP].

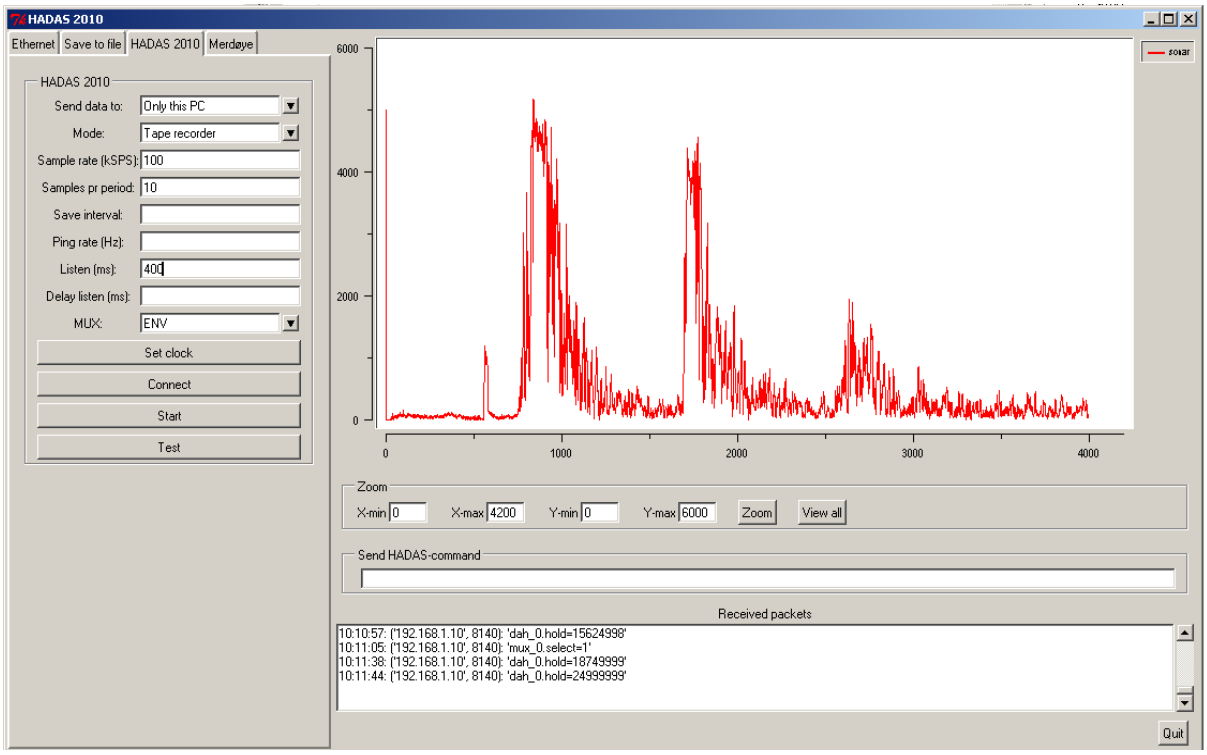
lwIP er laget for å bruke så lite ressurser som mulig samtidig som den tilbyr en rekke protokoller som IP, TCP, UDP, ICMP, IGMP, DHCP, ARP, SNMP, AUTOIP, PPP og DNS [wikia, lwIP wiki]. I Xilinx SDK kan man velge hvilke protokoller som skal inkluderes, og egenskapene for hver enkelt protokoll kan. I dette prosjektet er standardinnstillingene beholdt, unntatt at TCP er fjernet fra stakken.

Egenskaper for lwIP i dette prosjektet:

- Se vedlegg G for informasjon om hvordan systemet er satt opp.
- Fordi systemet er satt opp uten operativsystem kan ikke det forenklede brukergrensesnittet for lwIP benyttes. Dette brukergrensesnittet har også lavere ytelse enn ”raw”-grensesnittet som er benyttet i dette prosjektet [Xilinx, lwIP].
- lwIP er satt opp til å bruke operativsystemets tidtakere for å kalle enkelte funksjoner med faste mellomrom. Fordi vi ikke bruker noe operativsystem må vi kalle disse funksjonene fra vår egen kode. Funksjoner for å tømme ARP-tabellen og buffer for gjenoppbygging av IP-pakker er implementert i filen hardware.h. Dersom TCP, DHCP, AutoIP eller IGMP skal benyttes, må det legges til funksjoner for å kalle tcp_tmr(), dhcp_fine_tmr(), dhcp_coarse_tmr(), autoip_tmr() og igmp_tmr() med jevne mellomrom [Nongnu, lwIP documentation].
- Fragmentering av pakker på IP-nivå er aktivert, slik at data som er større enn MTU blir delt opp og sendt som flere pakker (se kapittel 4.1.1). Dataene som ligger i BRAM (kapittel 3.8.8) kan være inntil 8192 byte, så disse deles automatisk opp i flere pakker før de sendes. Ekkoloddsystemet skal ikke motta pakker som er større enn MTU, så sammensetting av IP-pakker er ikke aktivert.
- Fordi IP-adressen må være kjent og systemet i første omgang kun skal brukes direkte koblet til en PC, er det valgt å ikke bruke DHCP eller AutoIP for å sette IP-adressen. Den statiske IP-adressen er skrevet inn i C-koden, som derfor må endres og kompileres på nytt dersom IP-adressen skal endres Hvis det senere er ønskelig at systemet skal være koblet til et større nettverk, kan det være fornuftig å aktivere DHCP eller AutoIP så systemet kobler seg opp som det skal.
- Ekkoloddsystemet er satt til å lytte etter UDP-pakker til port 8140 fra enhver IP-adresse. Portnummeret er valgt basert på bokstavnummeret til de tre første bokstavene i navnet HADAS (H = 8, A = 1, D = 4) og at det er første UDP-port som er implementert i systemet (0). I følge [IANA, port numbers] er port 8140 ikke i bruk av noen registrerte programmer, og kommer derfor ikke i konflikt med øvrig Ethernet-trafikk.

All kildekode ligger vedlagt på CD-en og på internett. Det som ligger under mappen ”Kode\C\matralib” er generelle funksjoner som også brukes i andre prosjekter, mens de som ligger i ”Kode\C\projects\HADAS_firmware\versioner” er laget kun for dette prosjektet.

## 4.3 Datainnsamlingsprogram for PC




**Figur 4-5:** Skjermbilde av datainnsamlingsprogrammet.

For å styre ekkoloddsystemet og motta og lagre ekkoloddata, er det laget et program som kan installeres på en datamaskin. Programmet er skrevet i Python [Python community, Python], som er et objektorientert programmeringsspråk som kan brukes på forskjellige operativsystemer. Python har en lettlest og enkel syntaks med kraftige funksjoner, slik at det er forholdsvis enkelt å utvikle nye programmer.

I motsetning til kompilerte programmeringsspråk som for eksempel C, blir ikke Python kompilert før det kjøres. Kildekoden oversettes til plattformuavhengig bytekode (eng.: byte code) som deretter kjøres av Python [Langtangen, 2008, s. 8]. Funksjonene som faktisk utføres er skrevet i C, så Python kan enkelt utvides med nye C-moduler hvis det er funksjonalitet som mangler.

Programmeringsspråk som kjøres på denne måten kalles gjerne skripsspråk. Når programmer kjøres på denne måten, er ytelsen noe lavere enn det man kan oppnå med

optimalisert, kompilert kode. I følge [Langtangen, 2008, s. 8-9 og s. 42-43] er forskjellen veldig liten for generelle programmeringsoppgaver. Spesielt krevende sekvenser kan eventuelt skrives i C og brukes av Python-programmet, i tillegg til at Python-koden kan optimaliseres for hastighet [Langtangen, 2008, s. 439-449]. Det er mulig å kompilere de fleste Python-programmer med Py2exe [Py2exe, frontpage] til en kjørbar exe-fil, men dette har mindre å si for hastigheten til programmet.

Programmet som er skrevet i dette prosjektet har følgende hensikter:

- Sende kommandoer til ekkoloddsystemet for å konfigurere og styre det.
- Motta UDP-pakker med data fra ekkoloddsystemet over Ethernet.
- Plotte de mottatte dataene så operatøren kan følge med på at systemet fungerer som det skal.
- Lagre de mottatte dataene til fil så de kan analyseres og arkiveres.

Programmet kan brukes med ekkoloddsystemene Merdøye MKII (se kapittel 4.1.3.1) og HADAS 2010.

Det grafiske brukergrensesnittet er laget i med Tkinter [Python, Tkinter] som er et grensesnitt mot Tcl/Tk [Sourceforge, tcl]. For å kunne bruke større ferdiglagde elementer i brukergrensesnittet er Tkinter utvidet med Python megawidgets [Sourceforge, pmw]. Plottet er tegnet med BLT [Sourceforge, blt]. Tkinter er benyttet fordi det følger med Python, men de øvrige bibliotekene må installeres separat

Det grafiske brukergrensesnittet har et ganske gammeldags utseende, så hvis det skal gjøres en større oppdatering av programmet anbefales det å bytte til GUI-bibliotek (PyQt, PyGTK eller wxPython) Plottet kan gjerne tegnes med MathPlotLib[Sourceforge, matplotlib] som også støtter 3D-fargeplott for å tegne ekkogram.

All kildekode ligger vedlagt på CD-en og på internett under ”Kode\Python\projects\HADAS2010\”.

### 4.3.1 Filformat - EYM

Ekkoloddataene må lagres til et filformat som enkelt kan leses av programmene som senere skal brukes til å tolke dataene. I første omgang er det analyseprogrammene Sonar 4 og Sonar 5 som utvikles av førsteamanuensis Helge Balk ved Sonargruppa som skal benyttes [Balk, Sonar 5]. Filformatet EYM er utviklet av førsteamanuensis Helge Balk for at data fra nye hydroakustikkapplikasjoner skal kunne tolkes av Sonar.

Dataene som lagres er tellerverdier fra AD-omformeren. Når programmet brukes med Merdøye MKII er dette 8-bit pr sample (se kapittel 4.1.3.1). Med HADAS 2010 er det 16-bit pr sample, der det mest signifikante bittet er fortegnsbit og tellerverdien er de 13 minst signifikante bittene.

Filer i filformatet EYM har filetternavnet ".eym" og har følgende struktur:

FileHeader

PingHeader

Data

PingHeader

Data

…

**FileHeader**

| | | |
|---|---|---|
| Place | Array of 40 char | 40 byte |
| Survey area | Array of 40 char | 40 byte |
| Transect type | Array of 40 char | 40 byte |
| Transducer Nr | Integer | 4 byte |
| Start date, ddmmyyyy | Array of 8 char | 8 char |
| Start time, hhmmssnn | Array of 8 char | 8 char |
| Ping rate | Float – single | 4 byte |
| TVG | Integer | 4 byte |
| Calibration in file | Integer | 4 byte, 0=No 1=Yes |
| File name if not on this file | Array of 40 char | Other file name |
| Calibration sphere TS | Float – single | 4 byte ex. -40.3 dB |
| Max depth* | Float – single | 4 byte |
| Sample frequency (kHz) | Integer | 4 byte |
| Water temperature (deg) | Float – single | 4 byte |
| Sound frequency (kHz) | Float – single | 4 byte |
| Opening angle (deg) | Float – single | 4 byte |
| Pulse length (ms) | Float – single | 4 byte |
| Bandwidth (kHz) | Float – single | 4 byte |
| 2WayEqBeamAngle | Float – single | 4 byte |
| Dummy | Array [255] | 255 byte, Set all to 0 |
| Dummy | Array [255] | 255 byte, Set all to 0 |

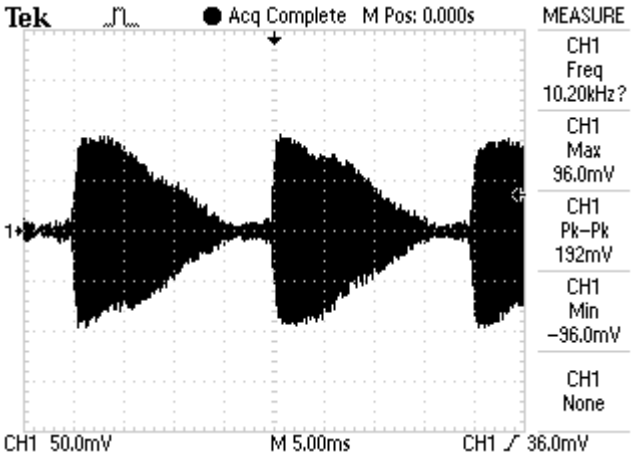
**PingHeader**

| | | |
|---|---|---|
| Code | Array of 8 byte | $1F2F3F4F5F6F7F8F |
| Nrsample | 4 byte | Integer |

# 5. Resultater

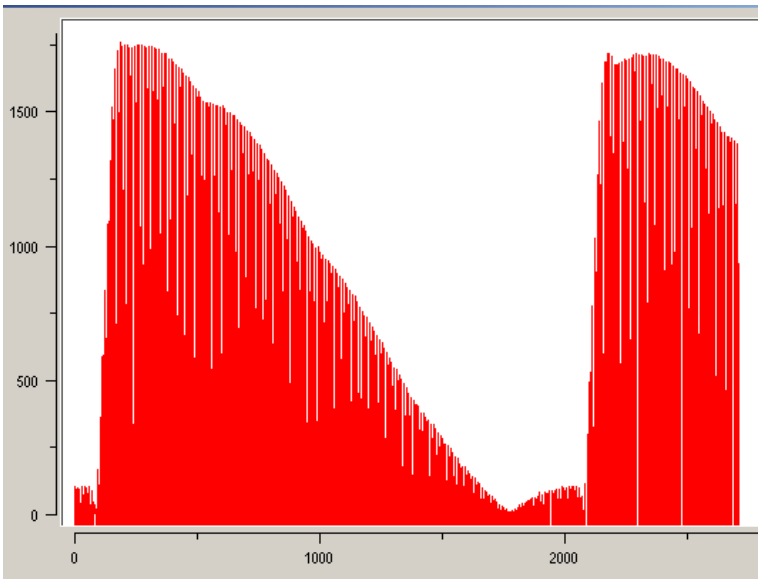
Ekkoloddsystemet er testet med frekvenser fra en frekvensgenerator (se vedlegg G) og med ekkoloddsignaler fra kassettspiller. Problemstillingen i kapittel 1.2 sier at systemet skal kunne behandle bærebølger med frekvens på 10 kHZ, 50 kHz, 70 kHz og 200 kHz. Bærebølger på 200 kHz er ikke mulig å behandle fordi en klokkefrekvens til AD-omformeren som er høy nok til å sample en slik bærebølge, dempes for mye av kretskortet (se kapittel 2.3.3.2). De øvrige frekvensene er testet ved å legge et amplitudemodulert signal fra frekvensgeneratoren på inngangen. Programmet som er utviklet i dette prosjektet, HADAS 2010 (kapittel 4.3), brukes til å styre FPGA-en og ta i mot og vise dataene.

## 5.1 Frekvensgenerator



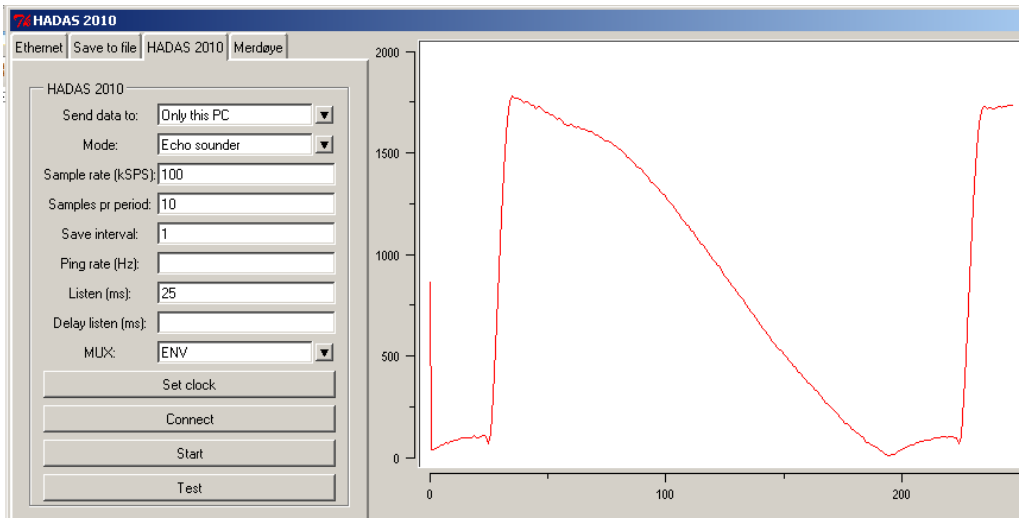

**Figur 5-1:** Testsignal fra frekvensgenerator. Bærebølge 10 kHz (Oscilloskopbilde med ScopePrinterBMP).

Testsignalet fra frekvensgeneratoren hadde en frekvens på 10 kHz, mens amplituden endret seg fra ca 100 mV mot null i løpet av ca 20 ms. Se figur 5-1.




**Figur 5-2:** Verdiene fra AD-omformeren. Bærebølgen på 10 kHz er samplet med 100 kSPS (Skjermbilde av HADAS 2010).

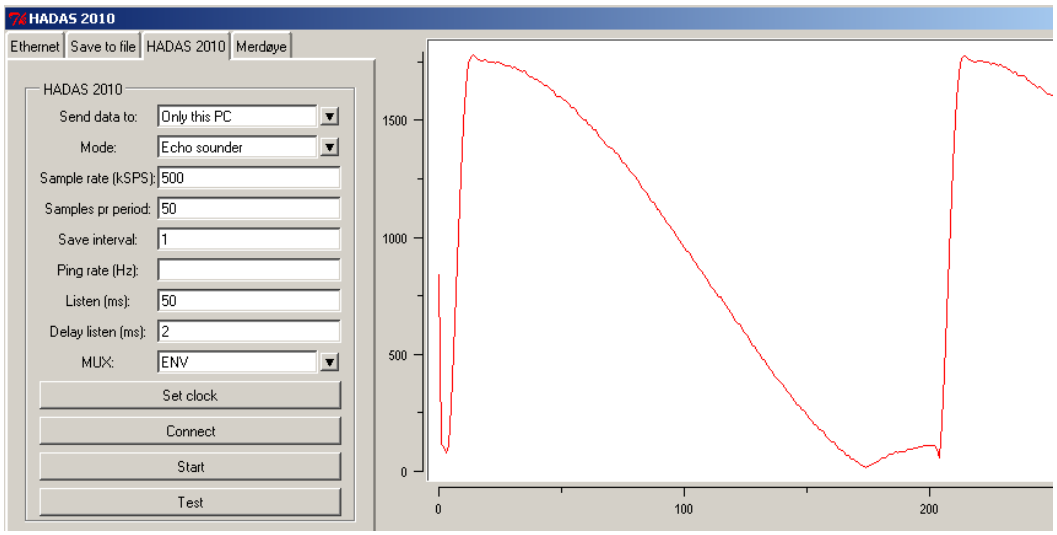
I figur 5-2 ser vi dataene som FPGA-en har hentet fra AD-omformeren. Bærebølgen på 10 kHz er samplet med 100 kSPS, så plottet viser ti sampler pr periode. På x-aksen i plottet ser vi at hver amplitudeendring skjer i løpet av ca 2000 sampler, som tilsvarer 20 ms (2000/100 kSPS).




**Figur 5-3:** Omhyllingskurven til signalet. Bærebølgen på 10 kHz er samplet med 100 kSPS (Skjermbilde av HADAS 2010).

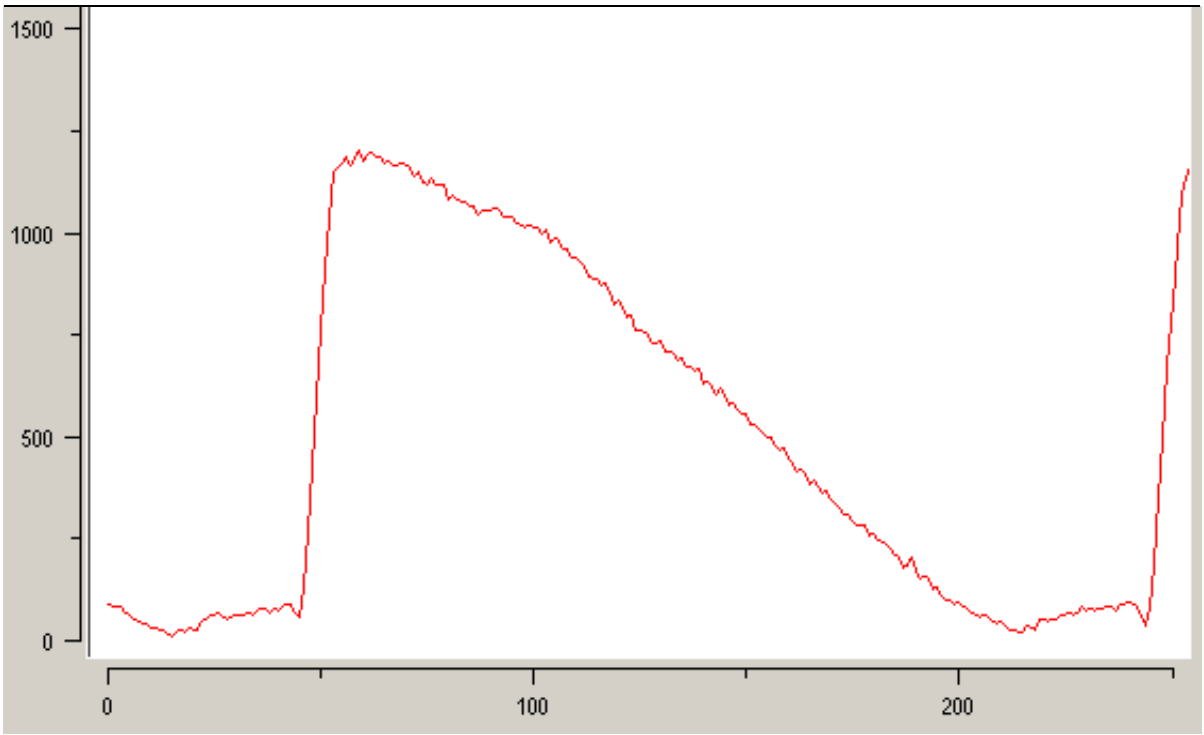
Figur 5-3 viser omhyllingskurven til signalet. Den er hentet ut med enheten envelope.vhdl (kapittel 3.8.3). Bærebølgen er samplet med 100 kSPS, så hver periode er samplet 10 ganger. Envelope.vhdl sender derfor videre den høyeste verdien av 10

etterfølgende samplingsverdier. I løpet av amplitudeendringen er det ca 200 perioder (20 ms / (1/10 kHz)), som vi kan se på x-aksen i plottet.



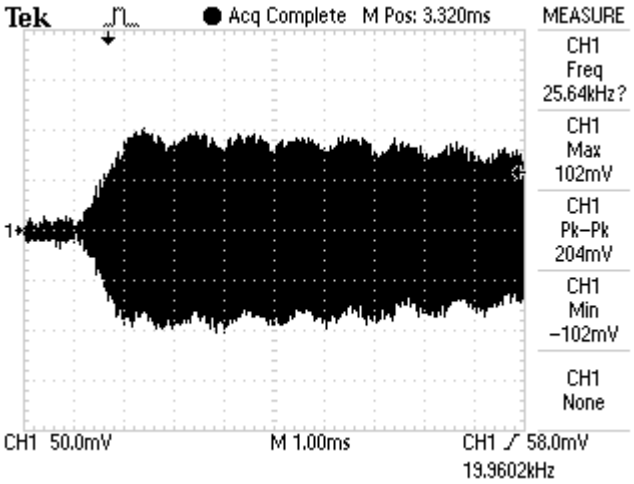

**Figur 5-4:** Omhyllingskurven til signalet. Bærebølgen på 10 kHz er samplet med 500 kSPS (Skjermbilde av HADAS 2010).

I figur 5-4 vises omhyllingskurven til signalet når den er samplet med 500 kSPS, mens envelope.vhdl sender den høyeste verdien blant 50 videre. Ved å sample flere ganger pr periode, vil noen av samplene treffe nærmere toppunktet i bærebølgen (se kapittel 2.3.3.2). Dette gir en mer nøyaktig gjengivelse av omhyllingskurven.

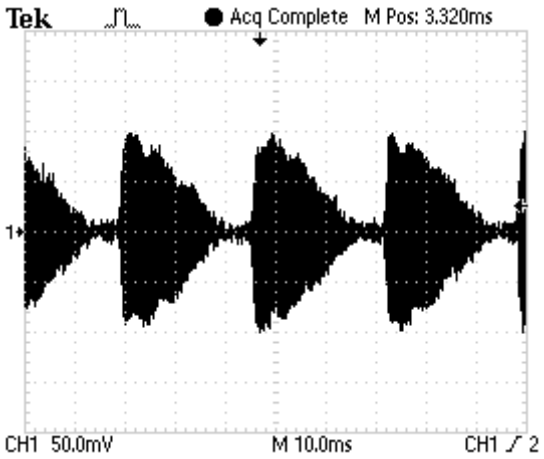


**Figur 5-5:** Omhyllingskurven til signalet uten båndpassfiltrering. Bærebølgen på 10 kHz er samplet med 100 kSPS (Skjermbilde av HADAS 2010).

Omhyllingskurven i figur 5-5 er hentet ut på samme måte som i figur 5-3, men signalet er koblet utenom båndpassfilteret på kretskortet. Ved å sammenligne figur 5-5 og figur 5-3 ser vi at omhyllingskurven er mer ujevn i figur 5-5. Dette skyldes støy som båndpassfilteret har fjernet i figur 5-3. Det er også noe lavere amplitudeverdier i figur 5-5 fordi båndpassfilteret forsterker signalet.



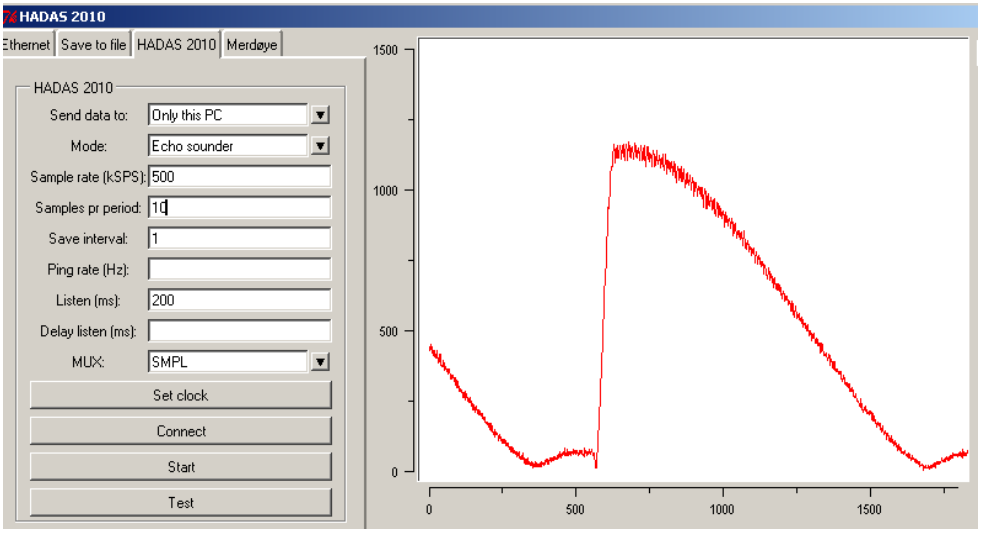

**Figur 5-6:** Testsignal fra frekvensgenerator. Bærebølge 50 kHz (Oscilloskopbilde med ScopePrinterBMP)




**Figur 5-7:** Testsignal fra frekvensgenerator. Bærebølge 70 kHz (Oscilloskopbilde med ScopePrinterBMP)

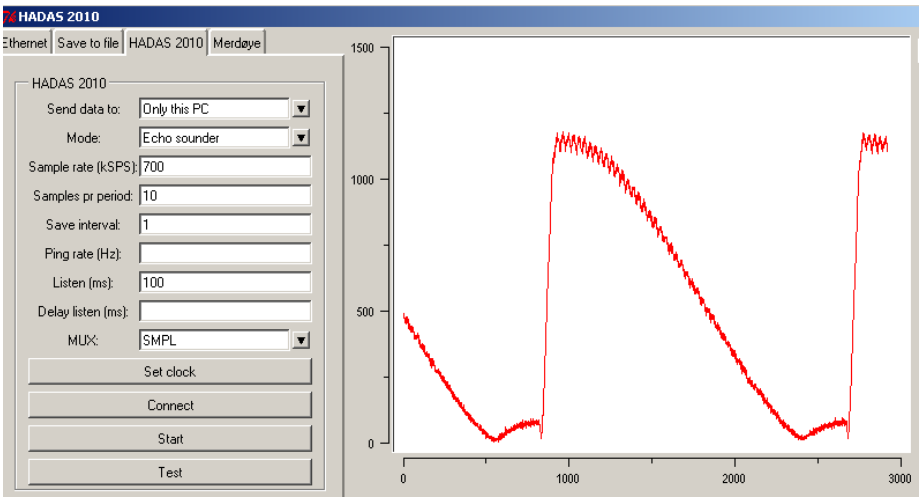
Vi testet deretter systemet ved frekvenser på 50 kHz og 70 kHz. Signalet hadde samme form som ved 10 kHz, men amplitudeendringen skjedde i løpet av ca. 25 ms.

Oscilloskopbildet i figur 5-6 viser signalet på 50 kHz med 1 ms tidsoppløsning, mens figur 5-7 viser signalet på 70 kHZ med 10 ms tidsoppløsning.

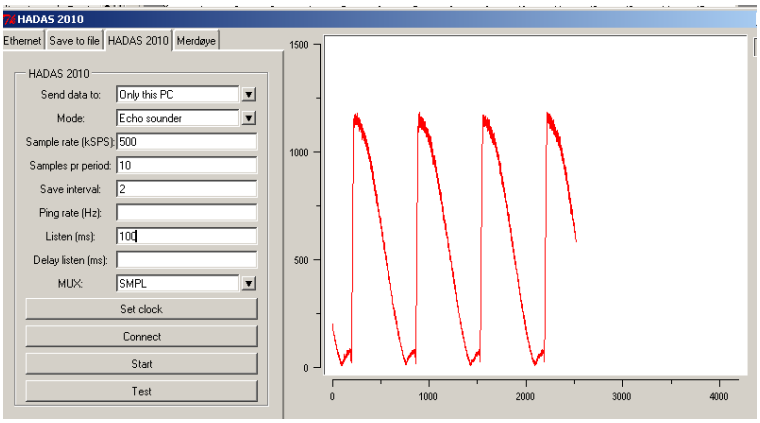



**Figur 5-8:** Omhyllingskurven til signal med bærebølge på 50 kHz. Det er samplet med 500 kSPS (Skjermbilde av HADAS 2010).

Vi samplet signalet med 10 samplinger pr periode. Samplingshastigheten var derfor 500 kSPS og 700 kSPS for henholdsvis 50 kHz og 70 kHz signal. Envelope.vhdl hentet ut den høyeste verdien av ti etterfølgende sampler i begge tilfellene. Som vist i figur 5-8 og figur 5-9 klarer systemet å hente ut omhyllingskurven til signalet også ved disse frekvensene. Signalene er ikke filtrert, så vi ser at det er en del støy som omgir omhyllingskurvene.



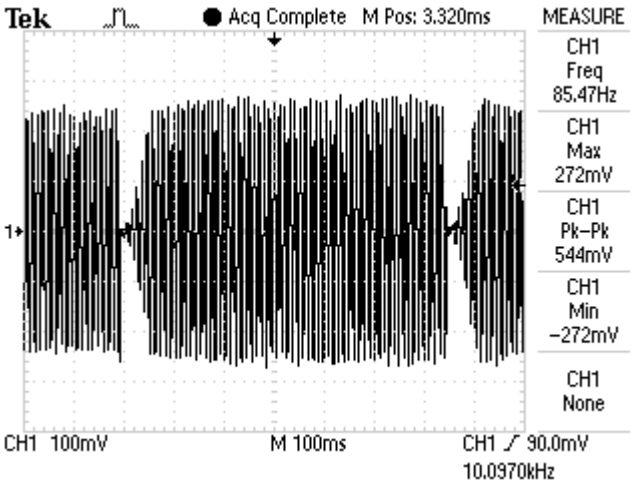

**Figur 5-9:** Omhyllingskurven til signal med bærebølge på 70 kHz. Det er samplet med 700 kSPS (Skjermbilde av HADAS 2010).




**Figur 5-10:** Nedsampling av omhyllingskurven (Skjermbilde av HADAS 2010).

Ved uthenting av omhyllingskurven fra signaler med høy frekvens, kan det bli for mange toppverdier i forhold til hva som er praktisk å lagre. Med enheten sample_data.vhdl kan verdiene i omhyllingskurven samples ned til en håndterlig mengde. I figur 5-10 er ”Save interval” satt til 2. Det betyr at annenhver toppverdi fra omhyllingskurven sendes videre. Det tilsvarer å sample omhyllingskurven med 25 kSPS i stedet for 50 kSPS.

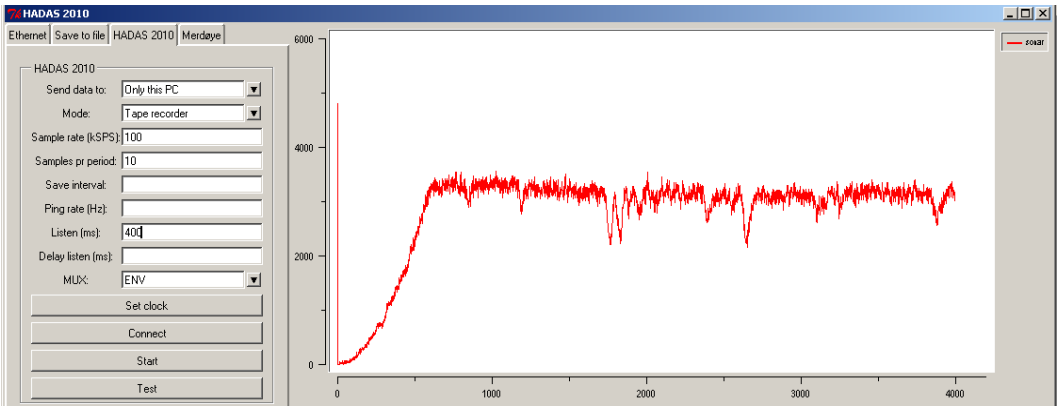
## 5.2 Kassettspiller



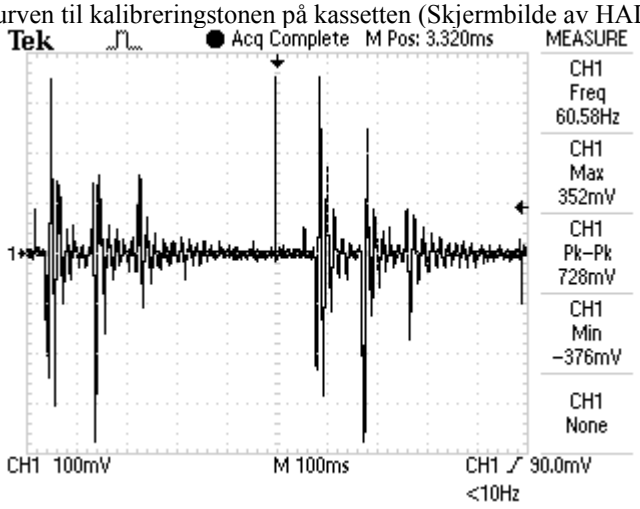

**Figur 5-11:** Kalibreringstone fra kassettspilleren (Oscilloskopbilde med ScopePrinterBMP)

Ved å endre ”Mode” til ”Tape recorder”, kunne vi teste systemet med ekkoloddsignaler fra kassettspilleren. Signalet fra kassettspilleren har en frekvens på 10 kHz, så vi

samplet det med 100 kSPS. Det var en kalibreringstone på kassetten som vist i figur 5-11. Omhyllingskurven til denne er vist i figur 5-12.

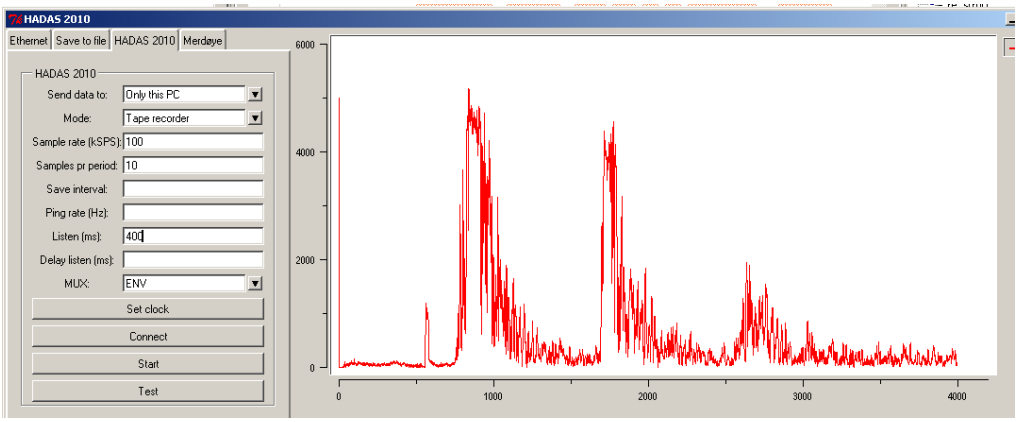



**Figur 5-12:** Omhyllingskurven til kalibreringstonen på kassetten (Skjermbilde av HADAS 2010).




**Figur 5-13:** Ekkoloddsignal fra kassettspilleren (Oscilloskopbilde fra ScopePrinterBMP)

I figur 5-13 vises oscilloskopbildet av et ping fra midten og mot høyre. Omhyllingskurven til et ping fra samme sekvens vises i figur 5-14. Vi kan se at omhyllingskurven til signalet er bevart slik at informasjonen kan hentes ut. Den første toppen er et objekt i vannet på ca 40 meters avstand ((1/10 kHz * 550 perioder * 1500 m/s)/2). Den største toppen er havbunnen på ca ((1/10 kHz * 800 perioder * 1500 m/s)/2) 60 meters avstand. De to etterfølgende toppene er refleksjoner av signalet mellom vannflaten og havbunnen.




**Figur 5-14:** Omhyllingskurven til et ekkoloddsignal fra kassettspilleren (Skjermbilde av HADAS 2010).

# 6. Konklusjon

Målsettingen med denne oppgaven var å lage et ekkoloddsystem basert på FPGA-teknologi. I henhold til problemstillingene som er beskrevet i kapittel 1.2, skulle arbeidet med oppgaven utføres med tre hensikter for øyet:

1. Ekkolodd for Laboratorium for Ferskvannsøkologi og Innlandsfiske (LFI) ved Naturhistorisk museum ved Universitetet i Oslo

2. System for å digitalisere gamle kassettopptak med sonardata.

3. Utvikle en forbedret metode for å hente ut omhyllingskurven til signalet.

Vi har ikke klart å få ferdig et ekkoloddsystem for LFI i løpet av arbeidet med denne oppgaven, men vi er flere skritt nærmere. Arbeidet som er gjort med den analoge kretsen, innholdet i FPGA-en, programmet på MicroBlaze-prosessoren og datainnsamlingsprogrammet for PC er gjort med tanke på at det skal inngå i et ekkolodd.

For at systemet skal kunne brukes som et ekkolodd må det lages et kretskort basert på arbeidet som er gjort i denne oppgaven og et eksisterende ekkoloddsystem (vedlegg A). Hvis det er ønskelig kan den eksisterende løsningen for tidsvariabel forsterkning (TVG) erstattes av en styrt teller på FPGA-en som er koblet til DA-omformeren. FPGA-en kan også generere transduserpulsen som skissert i kapittel 3.7 og 3.8. Andre ting som kan utbedres er merket med @TODO i kildekoden. Det gjenstående arbeidet vil kunne bli utført som en fremtidig masteroppgave eller av andre ved sonargruppen.

Som vist i kapittel 5.2 kan systemet brukes til å digitalisere kassetter med gamle opptak av ekkoloddsignaler. Testene viser at digitaliseringen er av god kvalitet, slik at jobben med å digitalisere disse opptakene kan påbegynnes.

Uthenting av signalets omhyllingskurve har tidligere blitt gjort med analog signalbehandling. Vi har i arbeidet med denne oppgaven vist at det med fordel kan

gjøres i en FPGA. Fordi vi kan anta at bærebølgen i signalet har en fast frekvens, har vi utviklet en enkel algoritme for å hente ut toppunktene fra hver periode i bærebølgen. Enheten ”envelope.vhdl”, som har blitt utviklet i prosjektet, henter ut signalets omhyllingskurve med stor nøyaktighet og ubetydelig forvrengning, som vist i kapittel 3.8.3 og kapittel 5.

På FPGA-en er det også implementert en prosessor som brukes for å styre ekkoloddsystemet. Prosessoren skriver og leser fra registre for å overvåke og konfigurere de øvrige enhetene som er utviklet på FPGA-en. Til sammen utgjør dette et System-on-Chip.

Prosessoren brukes også for å kommunisere via Ethernet med datainnsamlingsprogrammet som er utviklet i dette prosjektet. Ved å sende kommandoer i UDP-pakker fra datainnsamlingsprogrammet til FPGA-en, konfigureres systemet på FPGA-en. Brukeren har dermed full kontroll over hvordan signalbehandlingen i FPGA-en utføres, noe som er viktig ved ekkoloddsystemer for forskningsanvendelser.

# Kildeliste


| | |
|---|---|
| [Adescom Polska, Wire-speed Internet] | ADESCOM POLSKA, *Wire-speed, Internet: TCP/IP stack*, [online]. Tilgjengelig fra: http://www.adescom.com/ipac1.htm (Besøkt 7. desember 2009). |
| [Altera, 2008 Annual Report (Form 10-K)] | ALTERA CORPORATION, *2008 Annual Report (Form 10-K)* [online]. Tilgjengelig fra: http://phx.corporate-ir.net/External.File?item=UGFyZW50SUQ9MzMwMzg2fENoaWxkSUQ9MzEyMTQxfFR5cGU9MQ==&t=1 (Besøkt 30. november 2009) |
| [Analog Devices, AD5546] | ANALOG DEVICES, *AD5546/AD5556: Current Output, Parallel Input, 16-/14-Bit Multiplying DACs with 4-Quadrant Resistors,* D03810-0-9/09(A), September 2009 (Rev A), [online]. Tilgjengelig fra: http://www.analog.com/static/imported-files/data_sheets/AD5546_5556.pdf (Besøkt 14. desember 2009). |
| [Balk, Sonar 5] | BALK, HELGE, *Sonar4 and Sonar5-Pro post processing tools,* [online]. Tilgjengelig fra: http://www.fys.uio.no/~hbalk/sonar4_5/index.htm (Besøkt 28. januar 2010). |
| [Dimitri van Heesch, Documenting the code] | DIMITRI VAN HEESCH, *Doxygen: Documenting the code*, [online]. Tilgjengelig fra: *http://www.stack.nl/~dimitri/doxygen/docblocks.html* (Besøkt 8. desember 2009) |
| [Dimitri van Heesch, Doxygen] | DIMITRI VAN HEESCH, *Doxygen: Source code documentation generator tool*, [online]. Tilgjengelig fra: http://www.doxygen.org (Besøkt 8. desember 2009) |
| [Dimitri van Heesch, Special commands] | DIMITRI VAN HEESCH, *Doxygen: Special commands*, [online]. Tilgjengelig fra: *http://www.stack.nl/~dimitri/doxygen/commands.html* (Besøkt 8. desember 2009) |
| [DSPRelated, Analytic Signals and Hilbert Transform Filters] | DSPRELATED, *Analytic Signals and Hilbert Transform Filters*, [online]. Tilgjengelig fra: http://www.dsprelated.com/dspbooks/mdft/Analytic_Signals_Hilbert_Transform.html (Besøkt 28. januar 2010) |
| [Europractice, Xilinx] | EUROPRACTICE, Xilinx, [online]. Tilgjengelig fra: http://www.europractice.stfc.ac.uk/software/xilinx.html (Besøkt 28. januar 2010) |

| | |
|---|---|
| [Farnell, ATXMEGA 128] | FARNELL, *ATXMEGA 128*, [online]. Tilgjengelig fra: http://no.farnell.com/jsp/search/browse.jsp?N=500013+1004406&Ntk=gensearch_001&Ntt=xmega+128&Ntx=mode+matchallpartial (Besøkt 7. desember 2009). |
| [Farnell, Xilinx Spartan 3A DSP 1800 A] | FARNELL, *Xilinx Spartan 3A DSP 1800 A*, [online]. Tilgjengelig fra: http://no.farnell.com/xilinx/xc3sd1800a-4csg484c/fpga-spartan-3a-dsp-37k-cells-484bga/dp/1605855 (Besøkt 7. desember 2009). |
| [Fish Base, Salmo trutta trutta] | FISH BASE, *Swimming speeds of Salmo trutta trutta*, [online]. Tilgjengelig fra: http://filaman.uni-kiel.de/Physiology/SpeedList.php?ID=238&SortBy=Speedms (Besøkt 8. januar 2010)] |
| [FRANCO, 2002] | FRANCO, SERGIO, 2002. Design with operational amplifiers and analog integrated circuits. 3rd ed. Boston: McGraw-Hill |
| [Halbo og Ohlckers, 1995] | HALBO, LEIF OG OHLCKERS, PER, 1995, *Electronic components, packaging and production,* Oslo: L. Halbo. |
| [IANA, port numbers] | IANA, *Port numbers*, [online]., Tilgjengelig fra: http://www.iana.org/assignments/port-numbers (Besøkt 28. januar 2010) |
| [International Rectifier, 10BQ015] | INTERNATIONAL RECTIFIER, *10BQ015: Schottky rectifier,* rev. H [online]. Tilgjengelig fra: http://www.datasheetcatalog.org/datasheet/irf/10bq15.pdf (Besøkt 13. januar 2010). |
| [Kernelnewbies, Linux 2.6.31] | KERNELNEWBIES, *Linux 2.6.31*, [online]. Tilgjengelig fra: http://kernelnewbies.org/Linux_2_6_31 (Besøkt 14. januar 2010). |
| [Langtangen, 2008] | LANGTANGEN, HANS PETTER, 2008. *Python Scripting for Computational Science*. 3rd ed. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg. |
| [Lindem data Acquisition, 1992] | LINDEM DATA ACQUISITION, 1992, *Hydro Acoustic Data Acquisition System: HADAS: Instruction Manual: (Version 4.01, supporting the SIMRAD EY-M and EY-200 echosounders).* |
| [Linear, LTC2356] | LINEAR TECHNOLOGY CORPORATION, *LTC2356-12/LTC2356-14 - Serial 12-Bit/14-Bit, 3.5Msps Sampling ADCs with Shutdown,* [online]. Tilgjengelig fra: http://cds.linear.com/docs/Datasheet/2356f.pdf (Besøkt 2. desember 2009). |
| [Lynuxworks, Xilinx] | LYNUXWORKS, *Xilinx*, [online]. Tilgjengelig fra: http://www.lynuxworks.com/board-support/xilinx.php (Besøkt 14. januar 2010). |
| [MacLennan og Simmonds, 1992] | MACLENNAN, D. N. OG SIMMONDS, E. J., *Fisheries Acoustics,* London: Chapman & Hall. |

| | |
|---|---|
| [Mathworks, MATLAB & SIMULINK] | MATHWORKS, *MATLAB & SIMULINK*, [online] Tilgjengelig fra: http://www.mathworks.com/products/ (Besøkt 28. januar 2010) |
| [Maxfield, 2004] | MAXFIELD, CLIVE, 2004. The Design Warrior's Guide to FPGAs: Devices, Tools and Flows. Boston: Newnes/Elsevier. |
| [Mentor Graphics, ModelSim] | MENTOR GRAPHICS, ModelSim, [online]. Tilgjengelig fra: http://www.modell.com (Besøkt 28. januar 2010) |
| [Mork, 2000] | MORK, OLAV HÅKON, 2000. "UDP-basert datainnsamling fra Simrad EK 500-ekkolodd: flertrådsprogrammering med Borland Delphi for Microsoft Windows", hovedfagsoppgave. Oslo: O.H. Mork. |
| [MyHDL, From Python to silicon] | MyHDL, *From Python to silicon*, [online]. Tilgjengelig fra: http://www.myhdl.org (Besøkt 1. desember 2009) |
| [National Semiconductor, LM78LXX] | NATIONAL SEMICONDUCTOR, 1995, *LM78LXX Series 3-Terminal Positive Regulators*, februar 1995, [online]. Tilgjengelig fra: http://www.national.com/ds/LM/LM78L05.pdf (Besøkt 13. januar 2010). |
| [National, DP83865] | NATIONAL SEMICONDUCTOR (copyright 2004), *DP83865 Gig PHYTER® V 10/100/1000 Ethernet Physical Layer,* oktober 2004, [online]. Tilgjengelig fra: http://cache.national.com/ds/DP/DP83865.pdf (Besøkt 4. desember 2009). |
| [National, LM339] | NATIONAL SEMICONDUCTOR (copyright 2000), *LM139/LM239/LM339/LM2901/LM3302,* august 2000. |
| [Naturhistorisk museum, LFI] | NATURHISTORISK MUSEUM, UNIVERSITETET I OSLO, *Laboratorium for Ferskvannsøkologi og Innlandsfiske (LFI)*, [online]. Tilgjengelig fra http://www.nhm.uio.no/forskning-samlinger/forskning/oppdragsforskning/lfi/ (Besøkt 28. januar 2010). |
| [Natås, 2005] | NATÅS, TERJE M., 2005. *Signalbehandling for ingeniører*. Bergen: Høgskolen i Bergen. |
| [Nongnu, lwIP documentation] | NONGNU, *lwIP Documentation: lwip 1.3.0*, [online]., Tilgjengelig fra: http://www.nongnu.org/lwip (Besøkt 28. januar 2010) |
| [Nongnu, lwIP] | NONGNU, *lwIP: A Lightweight TCP/IP stack*, [online]., Tilgjengelig fra: http://savannah.nongnu.org/projects/lwip/ (Besøkt 28. januar 2010) |
| [Olsen, 2002] | OLSEN, TOR ANSTEIN, 2002. "Sanntids innsamling av sonardata under Windows -98: eksternt innsamlingskort bygget rundt en CPLD, Windows-applikasjon basert på flertrådkjøring", hovedfagsoppgave. Oslo: T. A. Olsen. |

| | |
|---|---|
| [Opencores, MCPU] | OPENCORES, *MCPU: A minimal CPU for a CPLD*, [online]. Tilgjengelig fra: http://www.opencores.org/project,mcpu,overview (Besøkt 15. januar 2010). |
| [OpenCores.org, tir-mode Ethernet MAC] | OPENCORES.ORG, *10_100_1000 Mbps tri-mode Ethernet MAC,* [online]. Tilgjengelig fra: http://www.opencores.org/project,ethernet_tri_mode (Besøkt 4. desember 2009). |
| [Petalogix, PetaLinux] | PETALOGIX, *PetaLinux*, [online]. Tilgjengelig fra: http://www.petalogix.com/products/petalinux (Besøkt 14. januar 2010). |
| [Philips Semiconductors, 1994] | PHILIPS SEMICONDUCTORS, 1994, *Dual and single low noise op amp NE5533/5533A/NE/SA/SE5534/5534A,* [online]. Tilgjengelig fra: http://pdf1.alldatasheet.com/datasheet-pdf/view/17973/PHILIPS/NE5534.html (Besøkt 13. januar 2010). |
| [Py2exe, fontpage] | PY2EXE, *Frontpage*, [online]. Tilgjengelig fra: http://www.py2exe.org/ (Besøkt 28. januar 2010). |
| [Python community, Python] | PYTHON COMMUNITY, *Python*, [online]. Tilgjengelig fra: http://www.python.org/ (Besøkt 28. januar 2010) |
| [Python, TkInter] | PYTHON, TkInter, [online]. Tilgjengelig fra: http://wiki.python.org/moin/TkInter (Besøkt 28. januar 2010). |
| [RealFast, UDP/IP cores] | REALFAST, *UDP/IP cores*, [online]. Tilgjengelig fra: http://www.realfast.se/rfipp/products/udpip/udpip.shtml (Besøkt 7. desember 2009) |
| [Samtec, HFEM2-SE] | SAMTEC, *HFEM2-SE series overview,* [online]. Tilgjengelig fra: http://www.samtec.com/ProductInformation/TechnicalSpecifications/Overview.aspx?series=HFEM2-SE (Besøkt 28. januar 2010). |
| [Samtec, QSE] | SAMTEC, *QSE series overview,* [online]. Tilgjengelig fra: http://www.samtec.com/ProductInformation/TechnicalSpecifications/Overview.aspx?series=QSE (Besøkt 28. januar 2010). |
| [Samtec, QSE] | SAMTEC, *QSE series,* [online]. Tilgjengelig fra: http://www.samtec.com/ProductInformation/TechnicalSpecifications/Overview.aspx?series=QSE (Besøkt 28. januar 2010). |
| [Samtec, QTE | SAMTEC, *QTE series overview,* [online]. Tilgjengelig fra: http://www.samtec.com/ProductInformation/TechnicalSpecifications/Overview.aspx?series=QTE (Besøkt 28. januar 2010). |
| [SIMRAD, 1979] | SIMRAD, *EY-M Echo Sounder Instruction Manual*, Publ. no.: P1136E. Mars 1979. |
| [SIMRAD, 1992] | SIMRAD, 1992. "Operator manual: SIMRAD EK 500 Scientific Echo Sounder". Horten: SIMRAD. |

| | |
|---|---|
| [Sjoholm og Lindh,1997] | SJÖHOLM, STEFAN og LINDH, LENNART, 1997. *VHDL for designers.* London: Prentice Hall. |
| [Skumsvoll, 2008] | SKUMSVOLL, KNUT, 2008. ”Utvikling av sonarsystem”, masteroppgave. Oslo: K. Skumsvoll. |
| [Sourceforge, blt] | SOURCEFORGE, *blt*, [online]. Tilgjengelig fra: http://blt.sourceforge.net/ http://www.py2exe.org/(Besøkt 28. januar 2010). |
| [Sourceforge, cipsuite] | SOURCEFORGE, *Compact Internet Protocol Suite*, [online]. Tilgjengelig fra: http://cipsuite.sourceforge.net (Besøkt 28. januar 2010) |
| [Sourceforge, matplotlib] | SOURCEFORGE, *matplotlib*, [online]. Tilgjengelig fra: http://matplotlib.sourceforge.net/ http://www.py2exe.org/ (Besøkt 28. januar 2010). |
| [Sourceforge, pmw] | SOURCEFORGE, *pmw*, [online]. Tilgjengelig fra: http://pwm.sourceforge.net/ http://www.py2exe.org/(Besøkt 28. januar 2010). |
| [Sourceforge, tcl] | SOURCEFORGE, *tcl*, [online]. Tilgjengelig fra: http://tcl.sourceforge.net/ (Besøkt 28. januar 2010). |
| [Texas Instruments, MAX3221] | TEXAS INSTRUMENTS (copyright 2004), *MAX3221: 3-V to 5.5-V single-channel RS-232 line driver/receiver with +-15-kV ESD protection*, [online]. Tilgjengelig fra: http://focus.ti.com/lit/ds/symlink/max3221.pdf (Besøkt 3. desember 2009). |
| [Texas Instruments, TL072] | TEXAS INSTRUMENTS, *TL071, TL071A, TL071B, TL072, TL072A, TL072B, TL074, TL074A, TL074B, Low-noise JFET-input operational amplifiers,* September 2003, [online]. Tilgjengelig fra: http://focus.ti.com/lit/ds/symlink/tl072.pdf (Besøkt 13. januar 2010). |
| [The Eclipse Foundation, Eclipse] | THE ECLIPSE FOUNDATION, *Eclipse*, [online]. Tilgjengelig fra: http://www.eclipse.org/ (Besøkt 28. januar 2010). |
| [Torex Semiconductor, XC6204] | TOREX SEMICONDUCTOR, *XC6204/XC6205Series*, ETR0304_005a, [online]. Tilgjengelig fra: http://www.torex.co.jp/english/products/pro02/pdf/6-XC6204~05.pdf (Besøkt 13. januar 2010). |
| [Treck, Xilinx] | TRECK, Inc., Xilinx, [online]. Tilgjengelig fra: http://www.treck.com/xilinx.html (Besøkt 28. januar 2010) |
| [Tyco, Mag45] | TYCO ELECTRONICS, *Mag45 – RJ45 with integrated magnetics*, [online]. Tilgjengelig fra: http://www.tycoelectronics.com/catalog/bin/TE.Connect?C=1&M=BYPN&PID=407634&PN=1-6605833-1&I=13 (Besøkt 4. desember 2009) |

| | |
|---|---|
| [VB helper, Twos Complement] | VB helper, Tutorial: Twos Complement Numbers [online]. Tilgjengelig fra: http://www.vb-helper.com/tutorial_twos_complement.html (Besøkt 1. april 2009) |
| [wikia, lwIP wiki] | WIKIA, *LwIP Wiki*, [online].Tilgjengelig fra: http://lwip.wikia.com/wiki/LwIP_Wiki (Besøkt 28. januar 2010) |
| [Wikimedia, Sonar Principle] | WIKIMEDIA, *Sonar Prinsiple EN*, [online]. Tilgjengelig fra: http://en.wikipedia.org/wiki/File:Sonar_Principle_EN.svg (Besøkt 8. januar 2010). |
| [Wikipedia, 2's complement] | Wikipedia, 2's complement [online]. Tilgjengelig fra: http://en.wikipedia.org/wiki/2%27s_complement (Besøkt 1. april 2009) |
| [Wikipedia, Ethernet] | Wikipedia, Ethernet, [online]. Tilgjengelig fra: http://en.wikipedia.org/wiki/Ethernet (Besøkt 11. januar 2010). |
| [Wikipedia, FPGA] | Wikipedia, FPGA [online]. Tilgjengelig fra: http://en.wikipedia.org/wiki/FPGA (Besøkt 12. mars 2009) |
| [Wikipedia, Hardware Description Language] | Wikipedia, Hardware Description Language, [online]. Tilgjengelig fra: http://en.wikipedia.org/wiki/Hardware_Description_Language (Besøkt 8. desember 2009) |
| [Wikipedia, IPv4] | Wikipedia, IPv4, [online]. Tilgjengelig fra http://en.wikipedia.org/wiki/IPv4 (Besøkt 12. januar 2010). |
| [Wikipedia, Jumbo frames] | Wikipedia, Jumbo Frames, [online]. Tilgjengelig fra http://en.wikipedia.org/wiki/Jumbo_Frames (Besøkt 12. januar 2010). |
| [Wikipedia, MTU] | Wikipedia, MTU, [online]. Tilgjengelig fra http://en.wikipedia.org/wiki/Maximum_transmission_unit (Besøkt 12. januar 2010). |
| [Wikipedia, Multiply-accumulate] | Wikipedia, Multiply-accumulate [online]. Tilgjengelig fra: http://en.wikipedia.org/wiki/Multiply-accumulate (Besøkt 24. februar 2009) |
| [Wikipedia, Sonar] | WIKIPEDIA, *Sonar*, [online]. Tilgjengelig fra: http://en.wikipedia.org/wiki/Sonar (Besøkt 8. januar 2010). |
| [Wikipedia, SPI] | Wikipedia, Serial Peripheral Interface Bus, [online]. Tilgjengelig fra: http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus (Besøkt 2. desember 2009) |
| [Wikipedia, System-on-a-chip] | WIKIPEDIA, *System-on –a-chip*, [online]. Tilgjengelig fra http://en.wikipedia.org/wiki/System-on-a-chip (Besøkt 16. januar 2010) |
| [Wikipedia, UDP encapsulation] | Wikipedia, *UDP encapsulation*, bilde, [online]. Tilgjengelig fra http://en.wikipedia.org/wiki/File:UDP_encapsulation.svg (Besøkt 12. januar 2010). |

| | |
|---|---|
| [Wikipedia, UDP] | WIKIPEDIA, *User Datagram Protocol*, [online]. Tilgjengelig fra: http://en.wikipedia.org/wiki/User_Datagram_Protocol (Besøkt 8. januar 2010) |
| [Xilinx Open Source Wiki, MicroBlaze-Linux] | XILINX OPEN SOURCE WIKI, *MicroBlaze Linux*, [online]. Tilgjengelig fra: http://xilinx.wikidot.com/microblaze-linux (Besøkt 14. januar 2010). |
| [Xilinx, Answer Browser] | XILINX, *Answer Browser,* [online]. Tilgjengelig fra: http://www.xilinx.com/support/answers/ (Besøkt 28. januar 2010) |
| [Xilinx, ChipScope] | XILINX, *ChipScope Pro and the Serial I/O Toolkit*, [online]. Tilgjengelig fra: http://www.xilinx.com/tools/cspro.htm (Besøkt 28. januar 2010) |
| [Xilinx, DC445] | XILINX, Inc., *DS445 - Local Memory Bus (LMB) V1.0 (v1.00a)*, 28. juli 2008. |
| [Xilinx, Design Tools] | XILINX; *XILINX DESIGN TOOLS-ISE 11.4,* [online]. Tilgjengelig fra: http://www.xilinx.com/publications/matrix/Software_matrix.pdf (Besøkt 28. januar 2010). |
| [Xilinx, DS444] | XILINX, Inc., *DS444 - Block RAM (BRAM) Block (v1.00a)*. 12. mars 2007. |
| [Xilinx, DS512] | XILINX, Inc. *DS512 - Block Memory Generator v2.8*. 19. September 2008. |
| [Xilinx, DS516] | XILINX, Inc. *DS516 - Interrupt Control (v2.00a),* 27. februar 2007. |
| [Xilinx, DS531] | XILINX, Inc., *DS531 - Processor Local Bus (PLB) v4.6 (v1.03a), 28. juli 2008.* |
| [Xilinx, DS537] | XILINX, Inc., *DS537 - XPS LL TEMAC (v1.01b)*. 22. juli 2008. |
| [Xilinx, DS570] | XILINX, Inc., *DS570 – XPS Serial Peripheral Interface (SPI) (v2.00b)*. 24. juli 2008. |
| [Xilinx, DS571] | XILINX, Inc., *DS571 - XPS UART Lite (v1.00a).* 18. juli 2008 |
| [Xilinx, DS572] | XILINX, Inc., *DS572 - XPS Interrupt Controller (v1.00.a)*, 22. juli 2008. |
| [Xilinx, DS573] | XILINX, Inc., *DS573 - XPS Timer/Counter (v1.00a),* 21. april 2008. |
| [Xilinx, DS580] | XILINX, Inc., *DS580 - XPS Ethernet Lite Media Access Controller (v2.00b)*. 29. juli 2008. |
| [Xilinx, DS610] | XILINX, Inc., *DS610 – Spartan-3A DSP FPGA Family: Data Sheet.* 11. mars 2009. |
| [Xilinx, DS614] | XILINX, Inc., *DS614 – Clock Generator (v2.01a)*. 28. juli 2008 |
| [Xilinx, DS643] | XILINX, Inc., *DS643 - Multi-Port Memory Controller (MPMC) (v4.03.a)*. 30. juli 2008. |
| [Xilinx, DS643] | XILINX, Inc., *DS643 – Multi-Port Memory Controller (MPMC) (v3.00.b)*. 8. november 2007. |

| | |
|---|---|
| [Xilinx, Embedded System Tools Reference Manual] | XILINX, Inc., Embedded System Tools Reference Manual: Embedded Development Kit, EDK 10.1, Service Pack 3, [online]. Tilgjengelig fra: http://www.xilinx.com/support/documentation/sw_manuals/edk10_est_rm.pdf (Besøkt 15. januar 2010). |
| [Xilinx, Extended Spartan-3A Family] | XILINX, Inc., Extended Spartan-3A Family, [online]. Tilgjengelig fra: http://www.xilinx.com/publications/prod_mktg/SP3A_Ext_Family_psm_table.pdf (Besøkt 7. desember 2009) |
| [Xilinx, HDL-library] | XILINX, Inc., *Spartan-3A and Spartan-3A DSP libraries guide for HDL designs: ISE 10.1*, [online]. Tilgjengelig fra: http://www.xilinx.com/itp/xilinx10/books/docs/spartan3a_hdl/spartan3a_hdl.pdf (Besøkt 9. desember 2009). |
| [Xilinx, ISE WebPACK] | XILINX; *ISE WebPACK Design Software,* [online]. Tilgjengelig fra: http://www.xilinx.com/tools/webpack.htm (Besøkt 28. januar 2010). |
| [Xilinx, lwIP] | XILINX, Inc., "lwIP 1.3.0 Library (v1.00.a)" i XILINX, Inc., *OS and Libraries Document Collection: EDK 10.1 Service Pack 2*. 19. September 2008. |
| [Xilinx, Spartan-6 Family] | XILINX, Inc., Spartan-6 Family FPGAs, [online]. Tilgjengelig fra: http://www.xilinx.com/publications/prod_mktg/Spartan6_Product_Table.pdf (Besøkt 7. desember 2009) |
| [Xilinx, System Generator] | XILINX, *System Generator for DSP*, [online] Tilgjengelig fra: http://www.xilinx.com/tools/sysgen.htm (Besøkt 28. januar 2010) |
| [Xilinx, Third Party Embedded Solutions Providers] | XILINX, Inc., *Third Party Embedded Solutions Providers*, [online]. Tilgjengelig fra http://www.xilinx.com/ise/embedded/epartners/listing.htm (Besøkt 14. januar 2010): |
| [Xilinx, UG081] | XILINX, Inc., *UG081 (v.9.3) – MicroBlaze Processor Reference Guide*, 14. juli 2008. |
| [Xilinx, UG331] | XILINX, Inc., *UG331 - Spartan-3 Generation FPGA User Guide: Extended Spartan-3A, Spartan-3E, and Spartan-3FPGA Families.* 21. januar 2009 |
| [Xilinx, UG431] | XILINX, Inc., *UG431 – XtremeDSP DSP48A for Spartan-3A DSP FPGAs User Guide*. 15. juli 2008. |
| [Xilinx, UG454] | XILINX, Inc., *UG454 – Spartan-3A DSP Starter Platform User Guide.* 30. januar 2009. |
| [Xilinx, User Community Forums] | XILINX, *User Community Forums*,[online]. Tilgjengelig fra: http://forums.xilinx.com/xlnx/ (Besøkt 28. januar 2010). |

| | |
|---|---|
| [Xilinx, Xilinx Investor Factsheet Second Quarter Fiscal Year 2010] | XILINX, INC., *Xilinx Investor Factsheet Second Quarter Fiscal Year 2010* [online]. Tilgjengelig fra: http://phx.corporate-ir.net/External.File?item=UGFyZW50SUQ9MTc1MTh8Q2hpbGRJRD0tMXxUeXBlPTM=&t=1 (Besøkt 30. november 2009) |
| [Xilinx, Xilkernel] | XILINX, Inc., *Xilkernel (v4.00.a),* [online]. Tilgjengelig fra http://www.xilinx.com/support/documentation/sw_manuals/edk10_oslib_rm.pdf (Besøkt 14. januar 2010). |
| [Xilinx, XST] | XILINX, Inc., *XST User Guide: 10.1*, [online]. Tilgjengelig fra: http://www.xilinx.com/itp/xilinx10/books/docs/xst/xst.pdf (Besøkt 9. desember 2009). |
| [Xilix, AR #21458] | XILINX, Inc., *AR #21458 - 10.1 NGDBuild - Adding "system.xmp" file to my project as a sub-module source results in: "Failed to process BMM file edkBmmFile.bmm"*, [online]. Tilgjengelig fra: http://www.xilinx.com/support/answers/21458.htm (Besøkt 15. januar 2010). |

# Programliste

Listen viser tekniske dataprogrammer som ble benyttet i arbeidet med masteroppgaven. Alle programmene ble kjørt under Microsoft Windows XP Professional Service Pack 3. Datamaskinen hadde en Intel Pentium 4 (3,20 GHz) prosessor og 2 GB RAM.

## Analog elektronikk

### Cadstar Design Editor

- Versjon 10.0 - Patch 4 og versjon 11.0
- Zuken Ltd.
- www.zuken.com
- Kretskortutlegg.

### FilterPro

- Versjon 2.00.0017
- Texas Instruments Incorporated
- www.ti.com
- Designe analoge filtre

### PSpice Schematics

- Evaluation version 9.1 – Web Update 1
- Cadence Design Systems
- www.cadence.com
- Simulering av analoge kretser.

### Orcad

- Versjon 10.3
- Cadence Design Systems
- www.cadence.com
- Simulering av analoge kretser.

### ScopePrinterBMP

- Versjon 1.0D
- Halvor Strøm
- Lagring av skjermbilder fra Tektronix TDS 1002 oscilloskop

# Programvareutvikling (Python, C)

## Python

- CPython versjon2.4.4 fra ActivePython
- Python Software Foundation
- www.python.org
- Interpreteringsprogram for all Python-kode.

## BLT

- Versjon 2.4z
- George A. Howlett
- http://blt.sourceforge.net/
- Tegning av plot i pythonprogrammer

## PMW

- Versjon 1.3
- http://pmw.sourceforge.net/
- Utvidelse av GUI-biblioteket Tkinter som følger med Python

## Resource Hacker

- Versjon 3.4.0.79
- Angus Johnson
- http://www.angusj.com/resourcehacker/
- Endre ikon på exe-filer

## Eclipse SDK

- Versjon 3.5.1
- Eclipse Foundation
- www.eclipse.org
- Programvareutviklingsverktøy. Brukt til å skrive VHDL- og Python-kode.
- Følgende moduler er installert:
    - PyDev
        - Versjon 1.4.4
        - Aptana
        - www.pydev.sourceforge.net
        - Verktøyboks for utvikling av Python-kode
    - Veditor
        - Versjon 0.7.0
        - http://sourceforge.net/projects/veditor/
        - Verktøyboks for utvikling av Verilog- og VHDL-kode. Gratis og bedre enn SimplifIDE.

- SimplifIDE
  - Versjon 1.1.0
  - SimplifIDE
  - www.simplifide.com
  - Verktøyboks for utvikling av Verilog- og VHDL-kode. Kommersiell modul som foreløpig er gratis betaversjon.

# Monitorering

## WireShark

- Versjon 1.0.6
- Gerald Combs and contributors
- www.wireshark.org
- Viser alle Ethernet-pakker som mottas av nettverkskortet. Brukt ved utvikling av nettverksapplikasjoner.

## Terra Term

- Versjon 4.6.1
- Tera Term Project
- http://ttssh2.sourceforge.jp/
- Terminal for seriell kommunikasjon via RS-232. Bedre enn Microsoft HyperTerminal. Brukt ved testing av FPGA-en.

# VHDL

## Xilinx ISE Design Suite

- Versjon 9.2.04i og 10.1.03
- Xilinx
- www.xilinx.com
- Syntese, implementering og programmering av Xilinx-FPGA-er.

## Xilinx Embedded Development Kit (EDK)

- Versjon 9.2.02i og versjon 10.1.03
- Xilinx
- www.xilinx.com
- **Xilinx Platform Studio (XPS):** Databuss-sentrert framstilling av integrerte systemer på FPGA-en. For integrering av prosessorer og andre IP-er (Intellectual Property) på Xilinx-FPGA-er.

- **Software Development Kit (SDK):** Eclipse-basert utviklingsverktøy for utvikling og kompilering av c-kode for integrerte prosessorer.

### ModelSim

- Versjon SE 6.2d og SE 6.5c
- Mentor Graphics
- http://www.mentor.com
- Simulering av HDL-kode.

## Dokumentasjon

### Doxygen

- Versjon 1.5.8
- Dimitri van Heesch
- http://www.doxygen.org
- Autogenerering av dokumentasjon (HTML, RTF m.m.) for kildekode (Brukt til VHDL og C).

### Epydoc

- Versjon 3.0
- Edward Loper
- http://epydoc.sourceforge.net
- Autogenerering av dokumentasjon (HTML, RTF m.m.) for kildekode (Brukt til Python).

### MATLAB

- Version 7.5.0.342 (R2007b)
- The MathWorks Inc.
- www.mathworks.com
- Matematiske beregninger

# Vedlegg

**A. KRETSSKJEMA FOR EKSISTERENDE SYSTEM ..... 172**

**B. BEREGNING AV OVERSAMPLING FOR AD-OMFORMEREN ..... 176**

**C. KRETSSKJEMA ..... 177**

**D. KRETSKORTUTLEGG ..... 184**

**E. PARAMETRE FOR BÅNDPASSFILTRENE ..... 189**

**F. DOXYGEN – DOKUMENTASJON AV KILDEKODE ..... 191**

F.1 SPESIALKOMMANDOER ..... 192

**G. LIGHT WEIGHT IP (LWIP) FOR XILINX FPGA-ER ..... 193**

G.1 BEGRENSNINGER I DENNE GUIDEN ..... 193

G.2 RESSURSER PÅ INTERNETT ..... 194

G.3 RESSURSER PÅ HARDDISK NÅR XILINX EDK ER INSTALLERT ..... 195

G.4 OPPSETT AV PROSESSORSYSTEMET I XPS ..... 196

G.5 ENDRINGER I SDK ..... 198

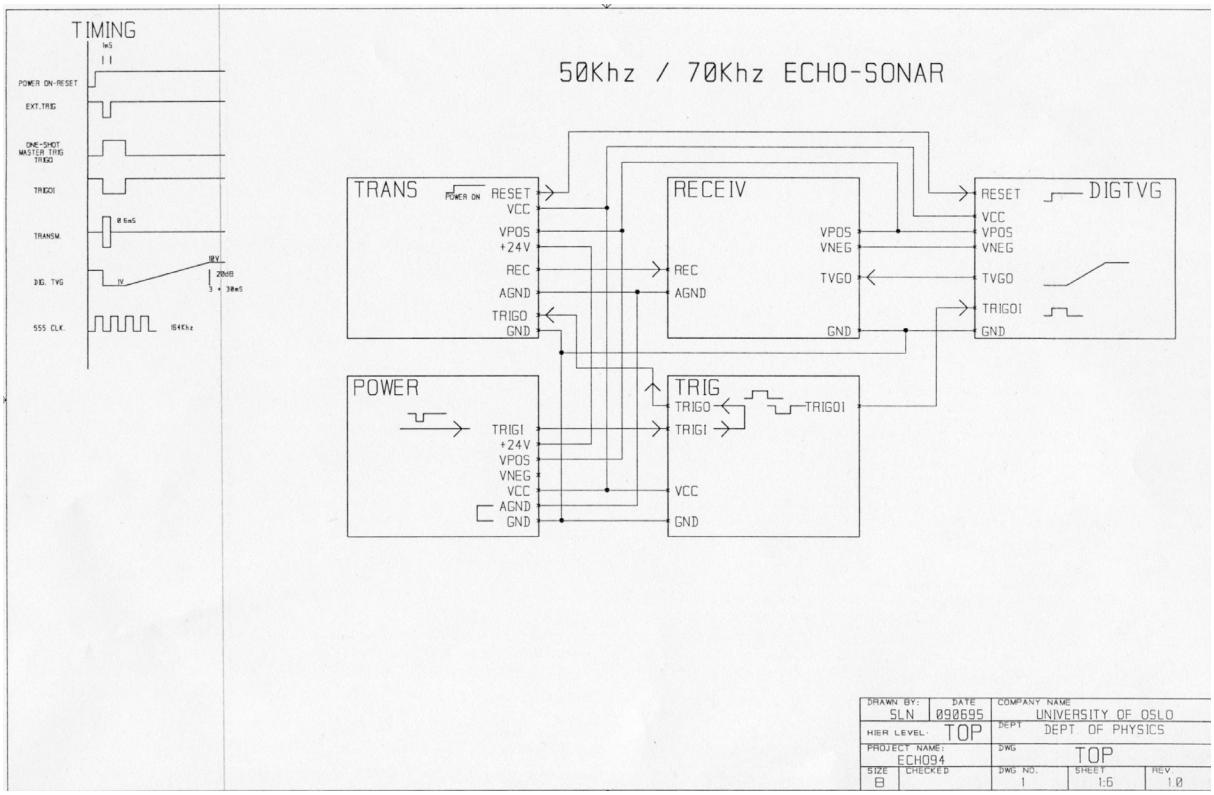
G.6 KONFIGURERE LWIP ..... 198

**H. TESTUTSTYR ..... 205**

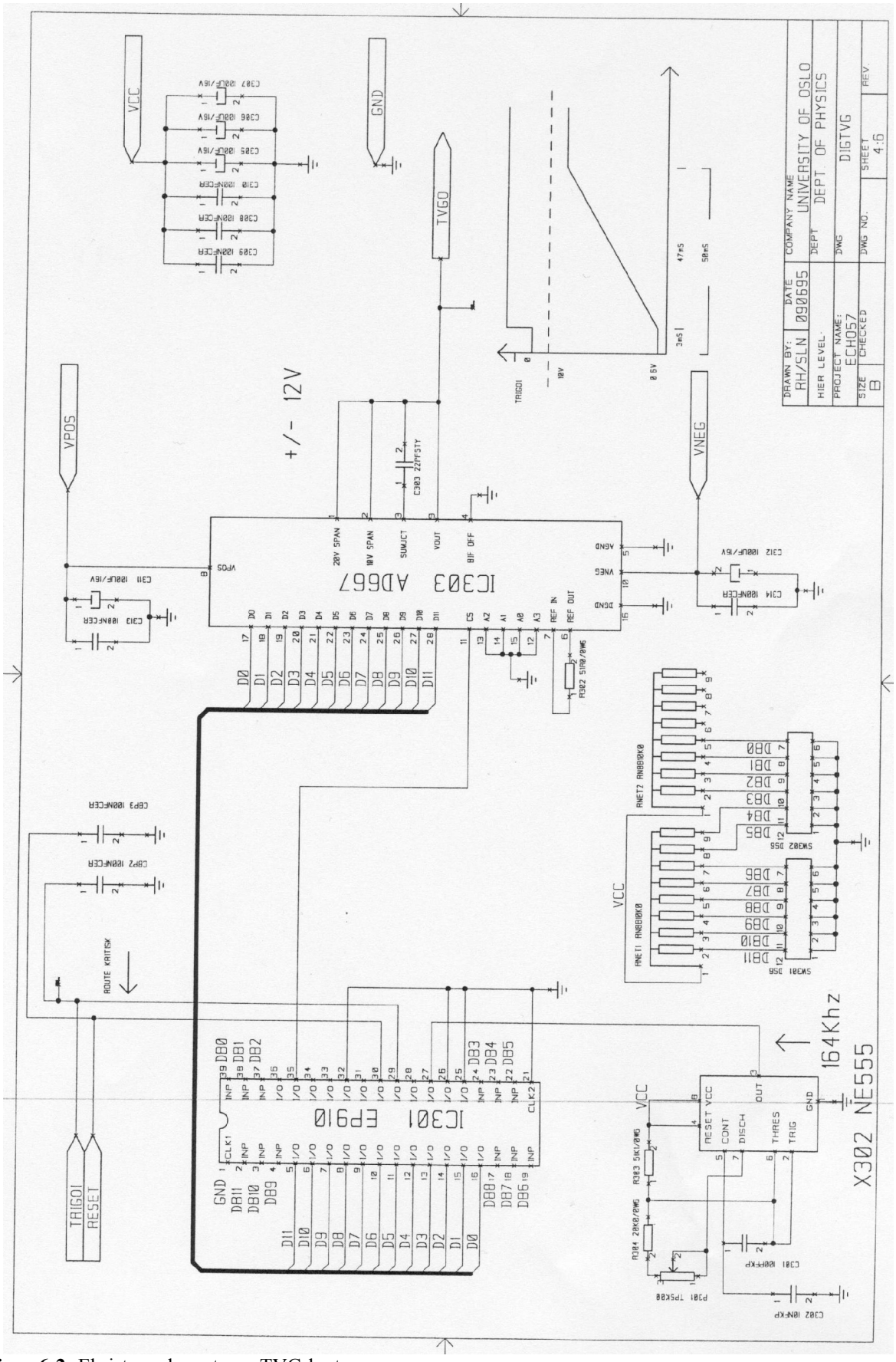
**I. VEDLEGG PÅ CD OG INTERNETT ..... 206**

# A. Kretsskjema for eksisterende system

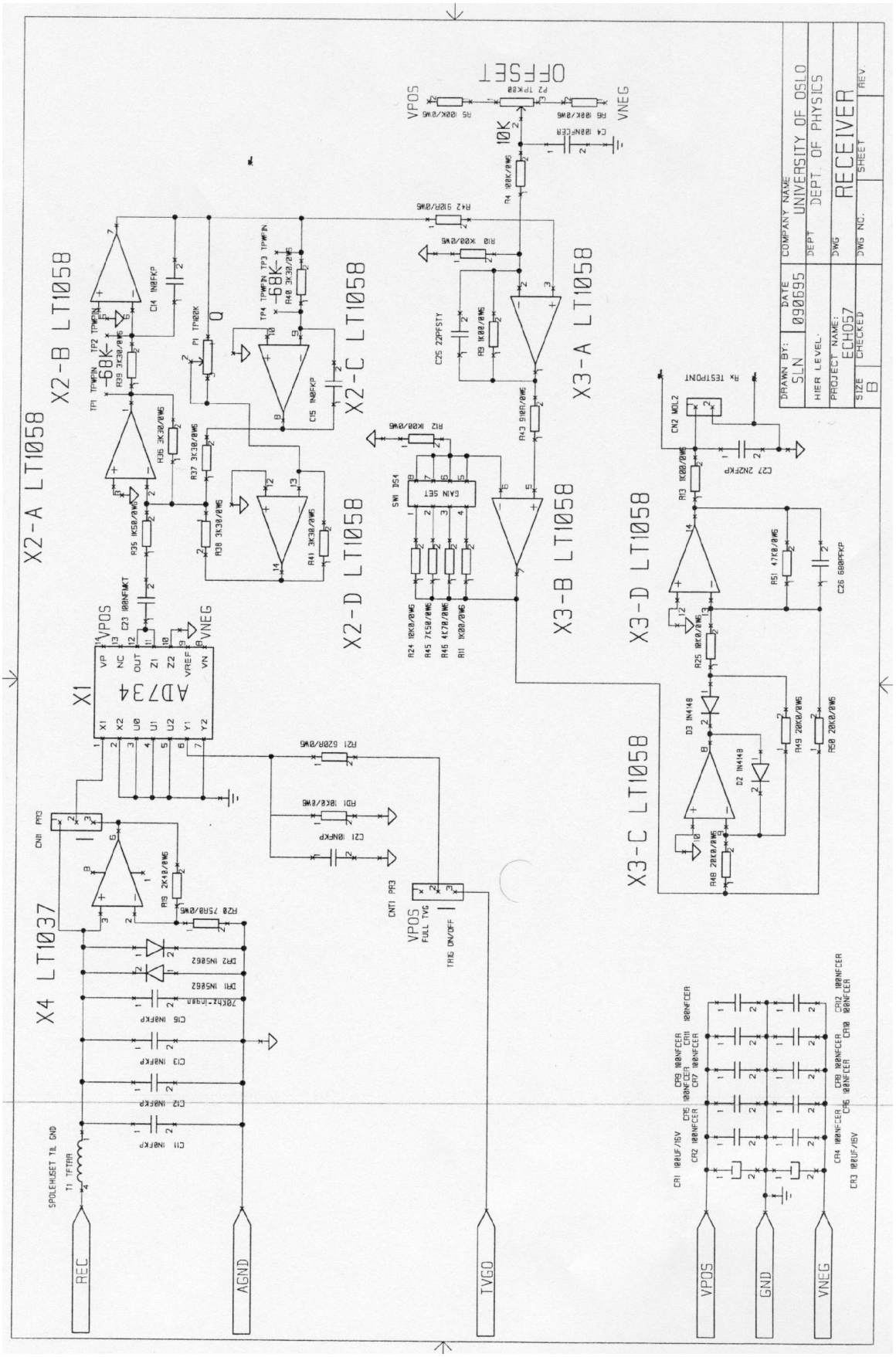
Kretsskjemaet er tegnet av Stein Lyng Nielsen ved Elektronikklaboratoriet ved Fysisk institutt.



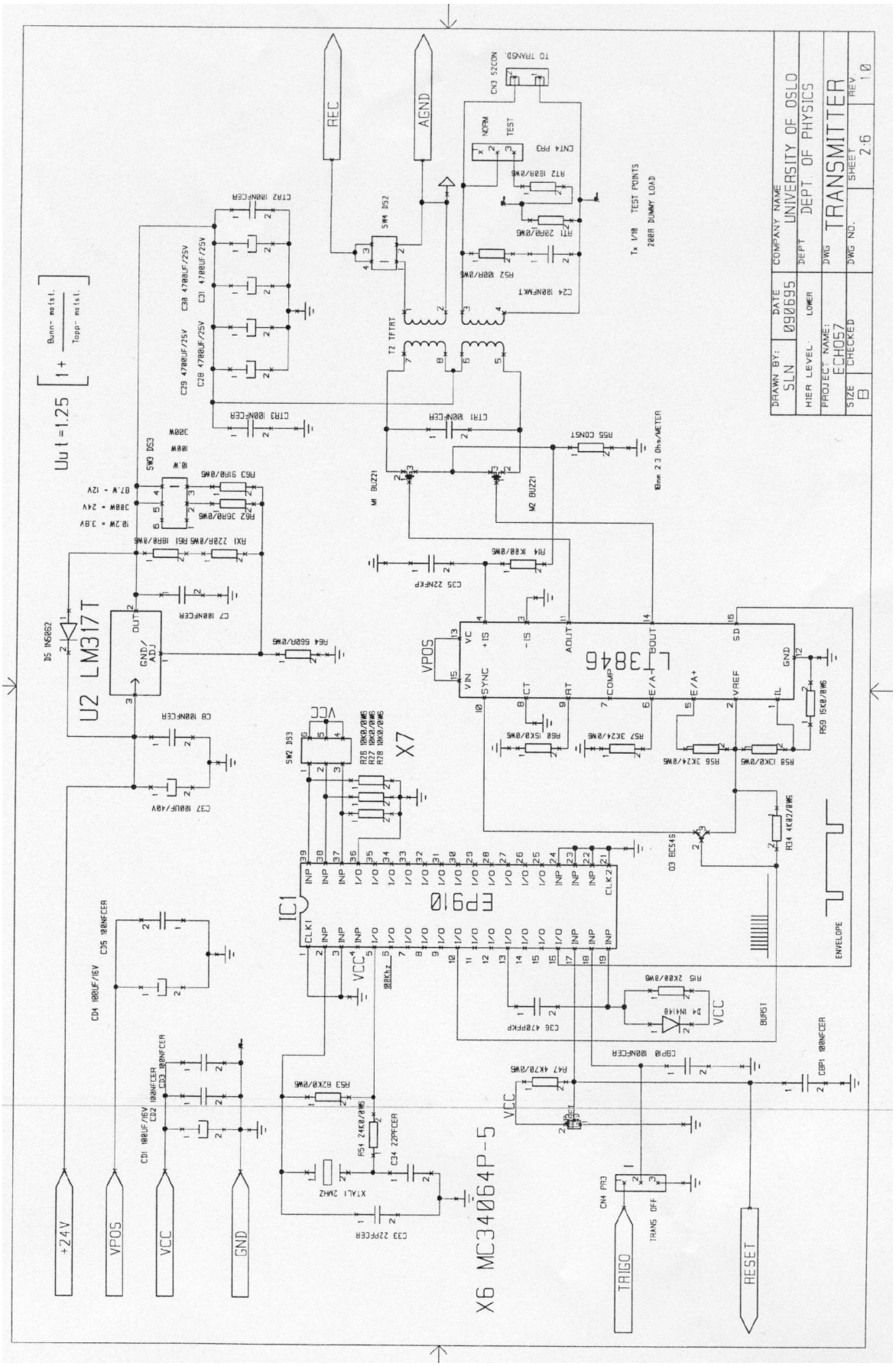

**Figur 6-1:** Eksisterende system - Toppskjema




**Figur 6-2:** Eksisterende system - TVG-krets




**Figur 6-3:** Eksisterende system – Receiver




**Figur 6-4:** Eksisterende system - transmitter

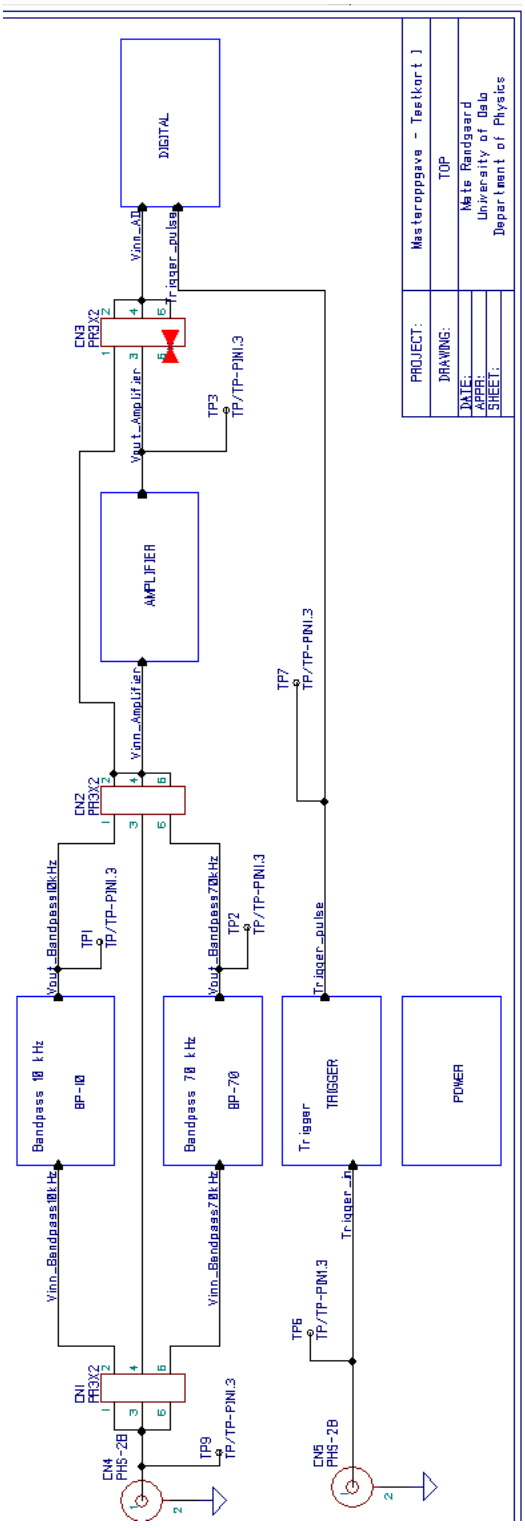
# B. Beregning av oversampling for AD-omformeren

**MATLAB-filen beregn_oversampling.m:**

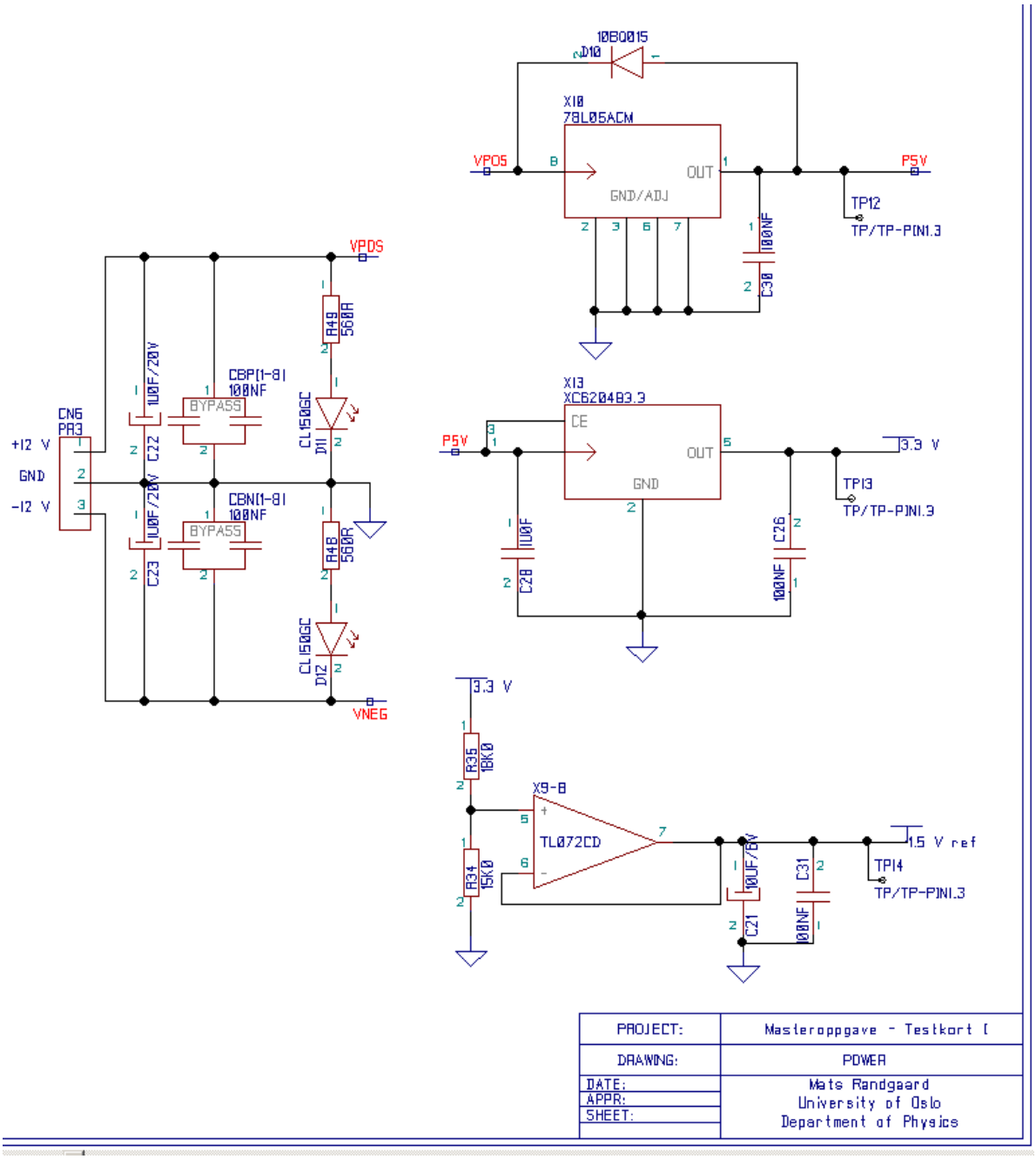
```
%Beregning av hvor mye et signal minimum må oversamples for at differansen
%mellom toppunktet og den største samplingsverdien ikke må være større enn
%en gitt verdi.
%
%Største differanse mellom toppunkt og samplingspunkt er når to
%samplingspunkt er like langt fra toppunktet.

%differanse mellom toppunkt og den største samplingsverdien i desibel
dB = 0.50;

%dB = 20*log10(1/x)
x = 1/(10^(dB/20));

%Sampel som treffer på siden av toppen
t_siden = asin(x);

%Avstand i tid fra toppen (sin(pi/2)=1) til sampel som treffer på siden av
toppen
t_diff_topp_siden = (pi/2)-t_siden;

%Differanse mellom to sampel som treffer på hver sin side av toppen
t_diff = 2*t_diff_topp_siden;

%Det andre sampelet som treffer like langt fra toppen
t_andre_siden = t_siden + t_diff;

%Beregner samplingspunkter utfra at to punkter skal ligge like langt fra
%toppunktet.
t_sample = t_siden-(t_diff):t_diff:4*pi;

%Beregner sinuskurven.
t = 0:2*pi/1000:4*pi;
f = sin(t);
plot(t,f)%Tegner sinuskurven
hold
stem(t_sample, sin(t_sample))%Tegner samplingspunktene
pause
close

%Beregner hvor mye frekvensen må oversamples.
oversampling = (2*pi)/t_diff;

disp(['Største differanse mellom toppunkt og samplingspunkt er når to'])
disp(['samplingspunkt er like langt fra toppunktet. Hvis denne avstanden'])
disp(['skal være mindre enn ', num2str(dB),' dB må samplingshastigheten'])
disp(['være minst ', num2str(oversampling),' ganger raskere enn
frekvensen'])
disp(['som skal samples'])
```

# C. Kretsskjema

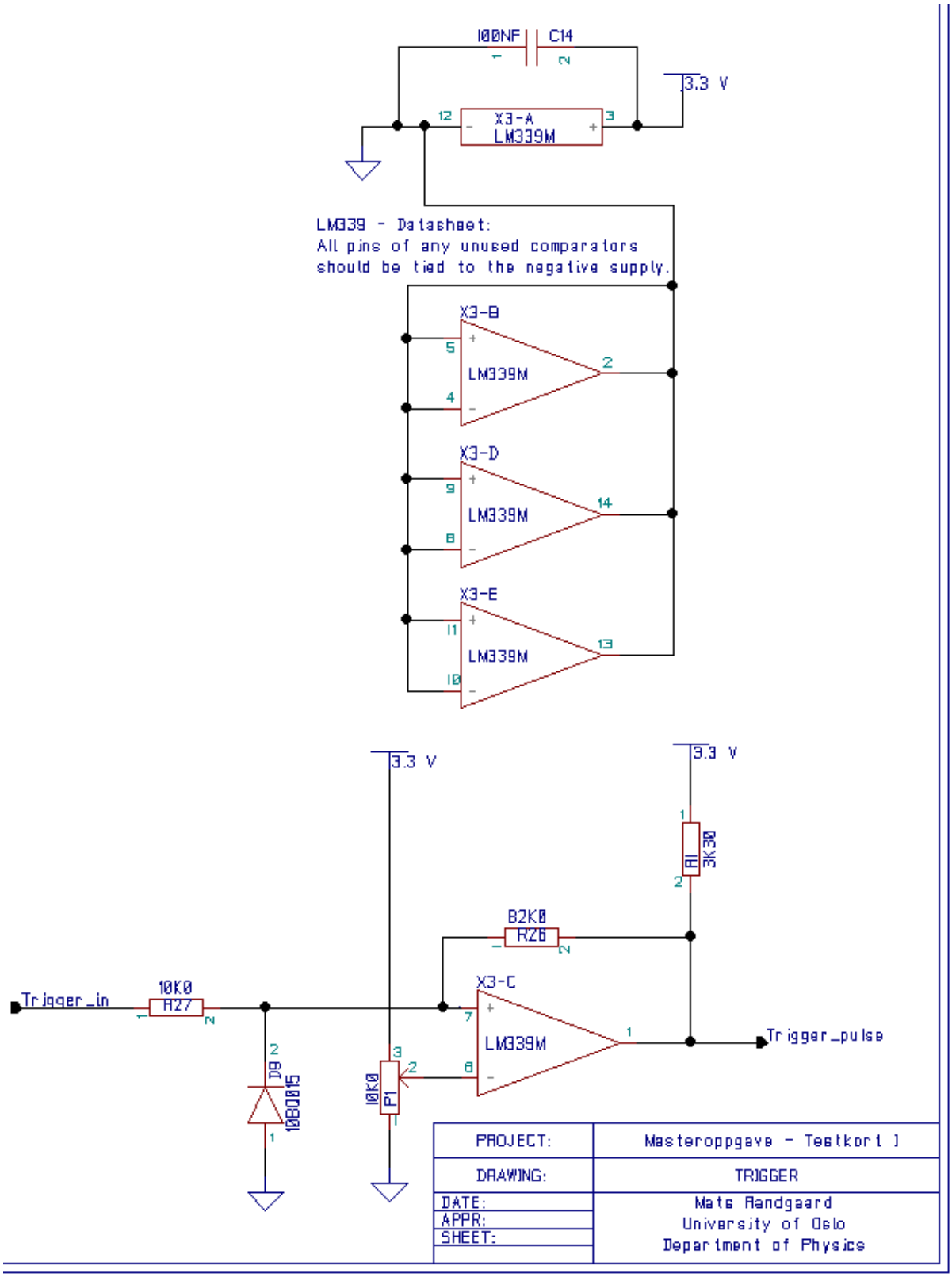
Kretsskjema for Zuken Cadstar 11 ligger på vedlagt CD.



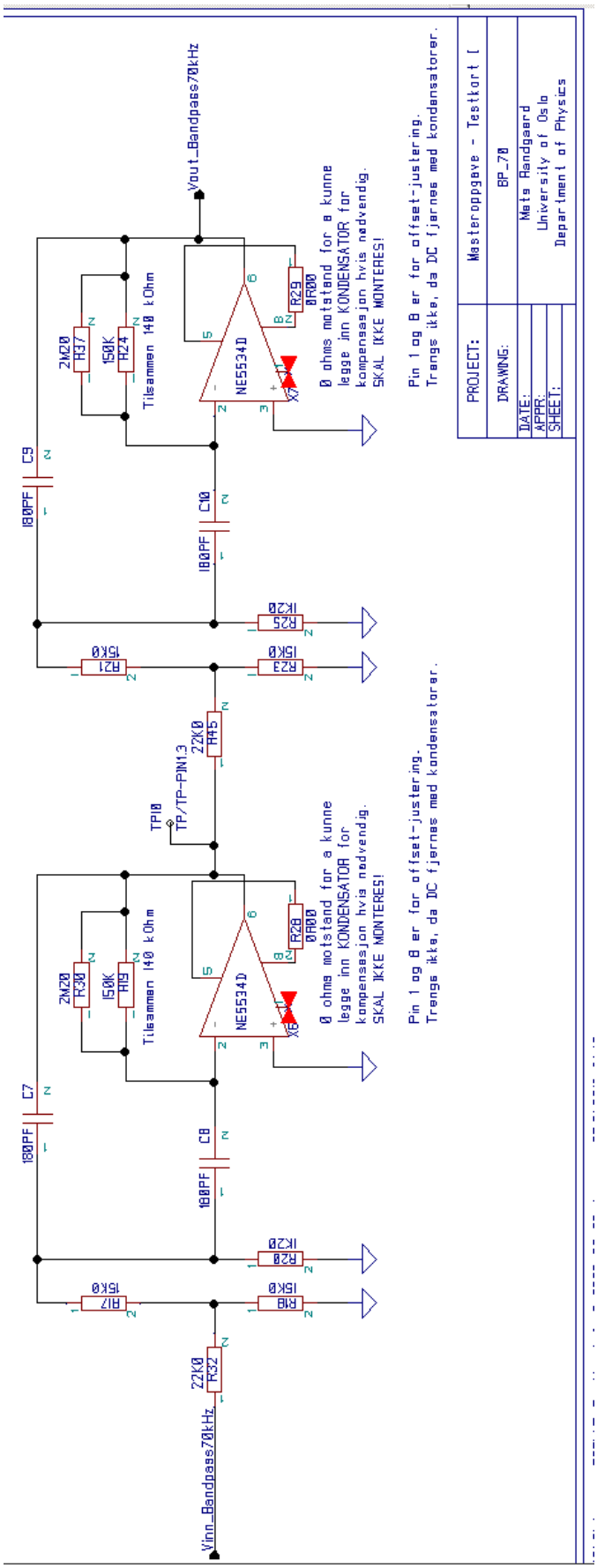

**Figur 6-5:** Toppnivå



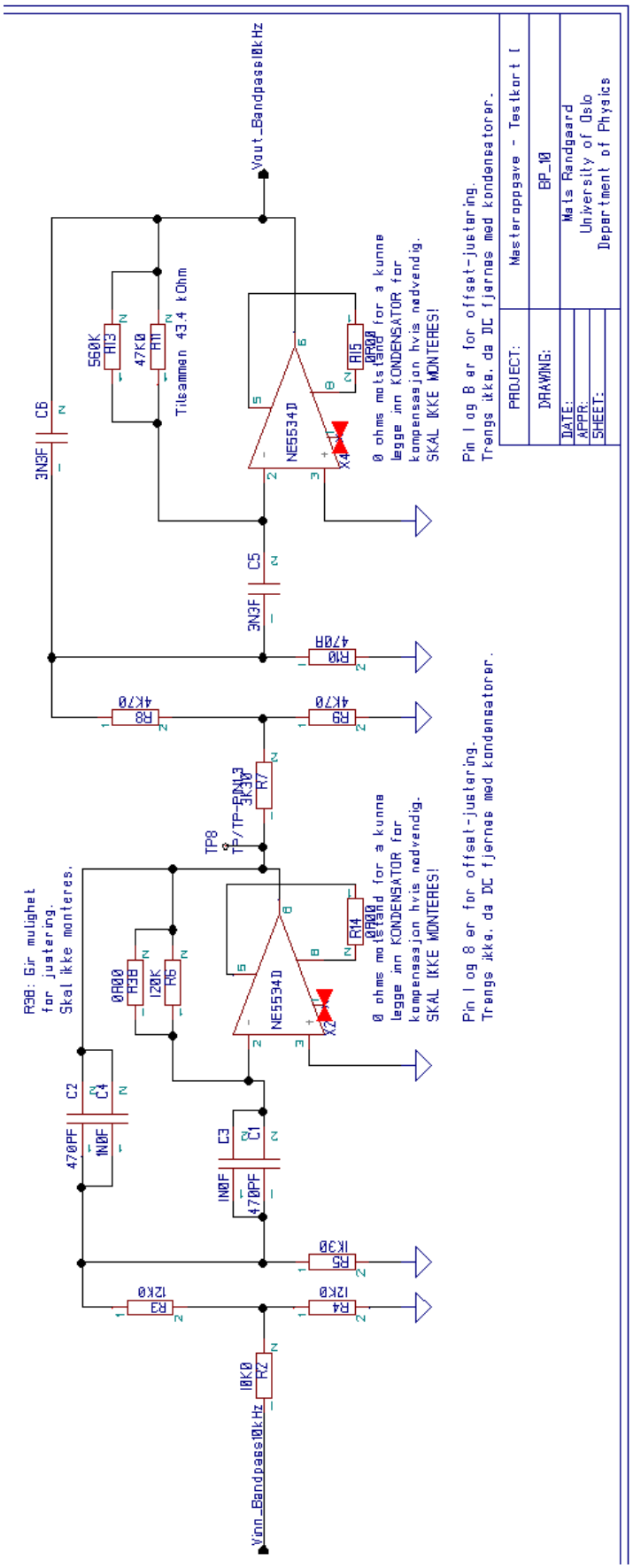

**Figur 6-6:** Strømforsyning



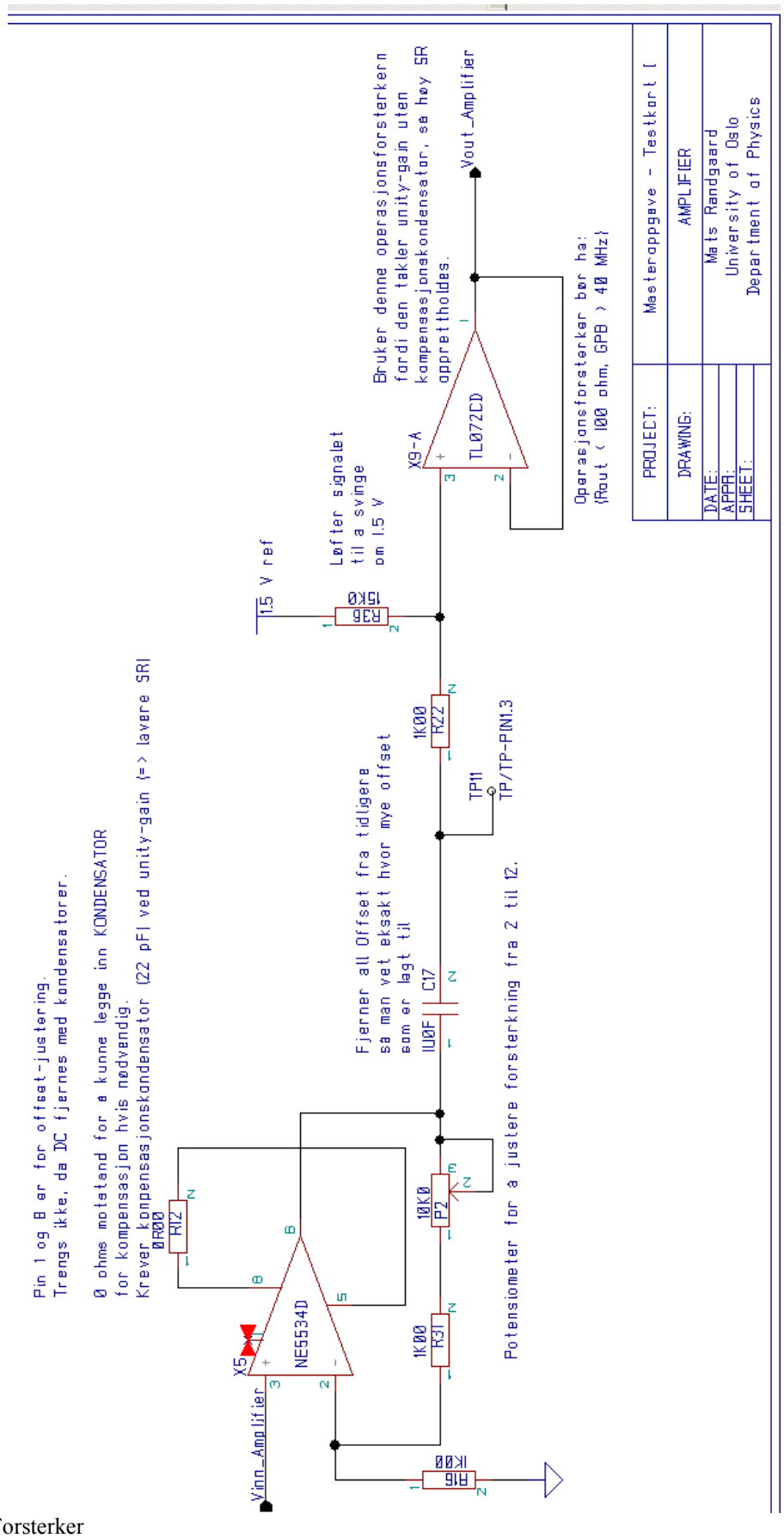

**Figur 6-7:** Trigger



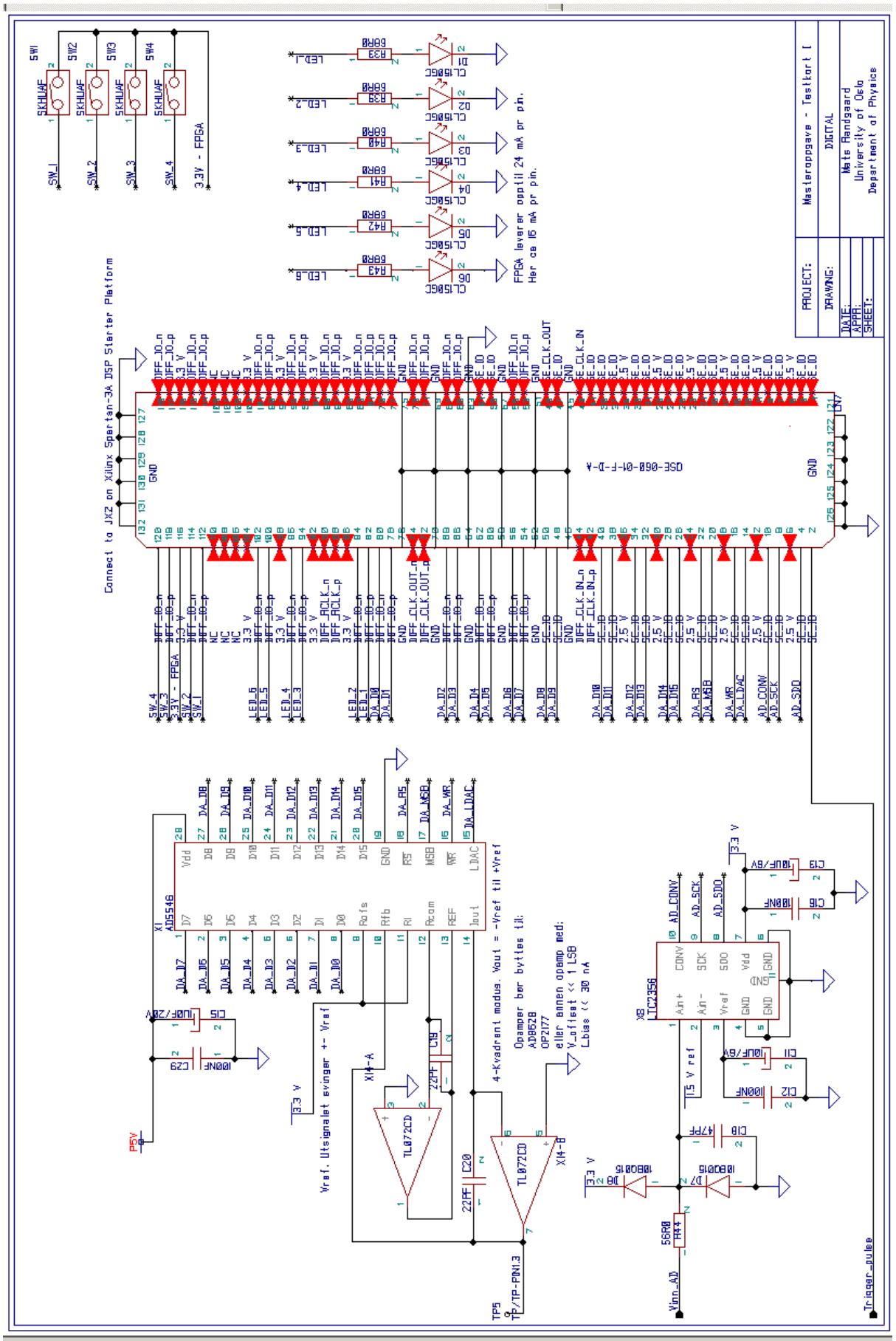

**Figur 6-8:** Båndpassfilter 70 kHz




**Figur 6-9:** Båndpassfilter 10 kHz



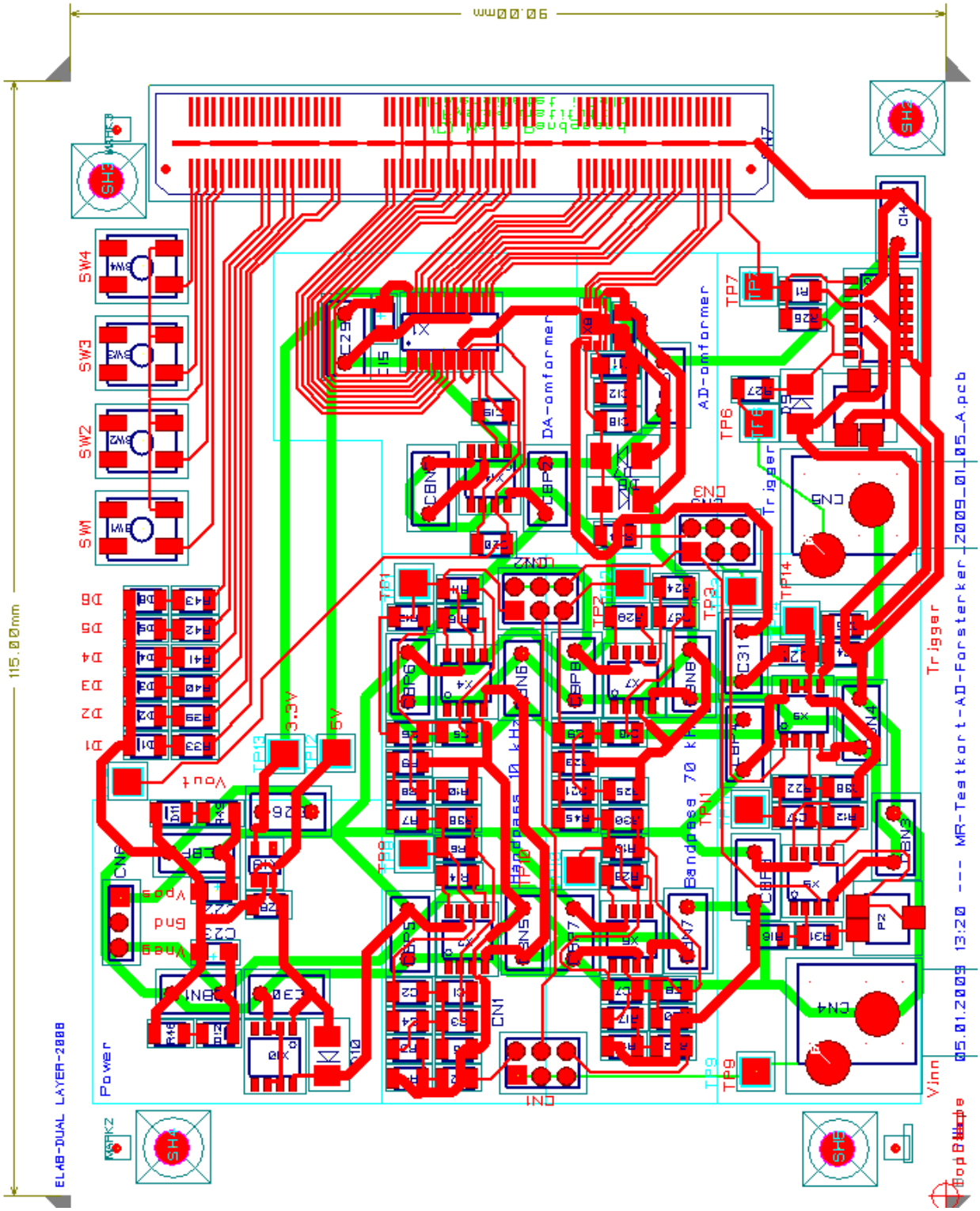

**Figur 6-10:** Forsterker



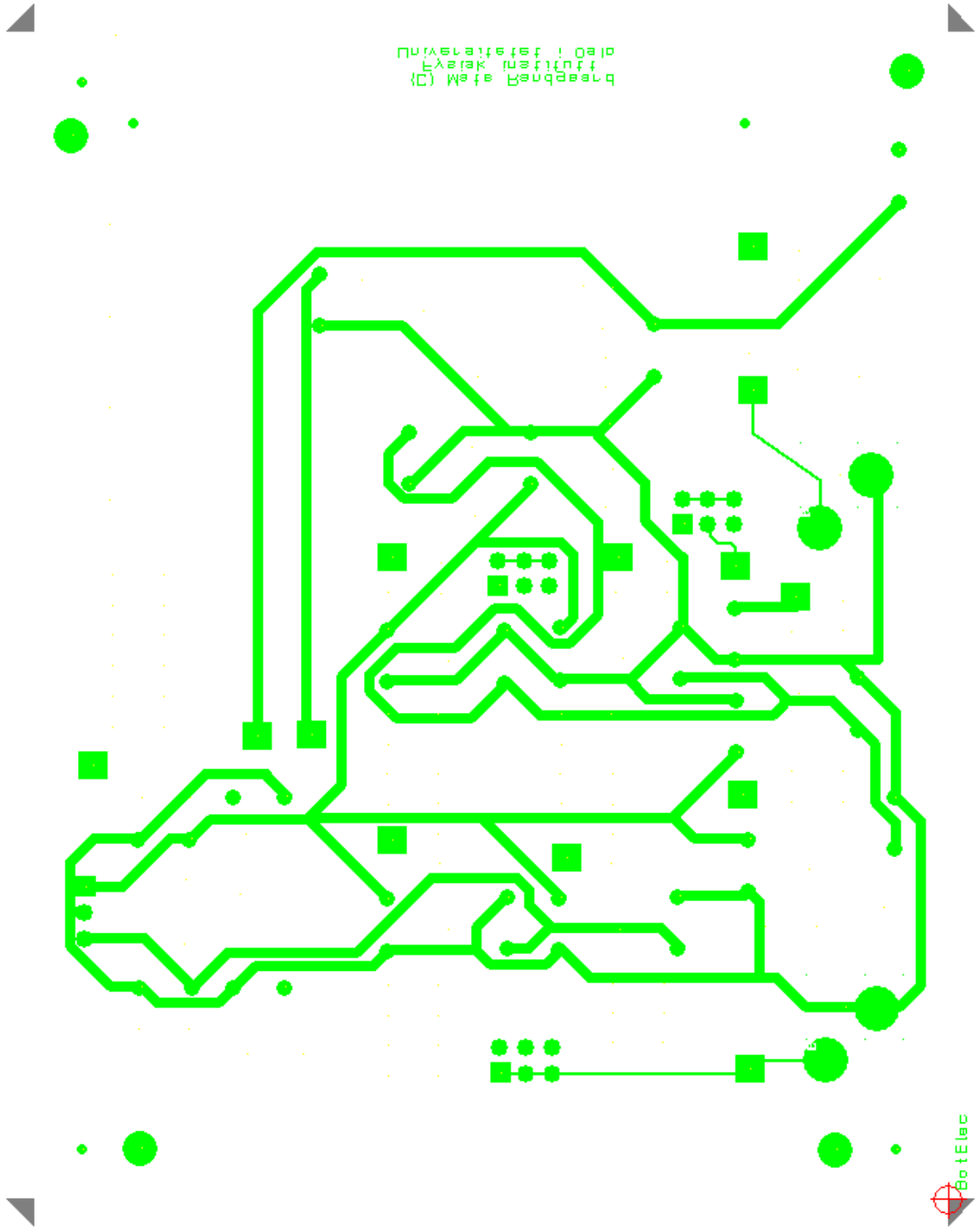
**Figur 6-11:** AD-omformer, DA-omformer og kontakt til FPGA-en.

# D. Kretskortutlegg

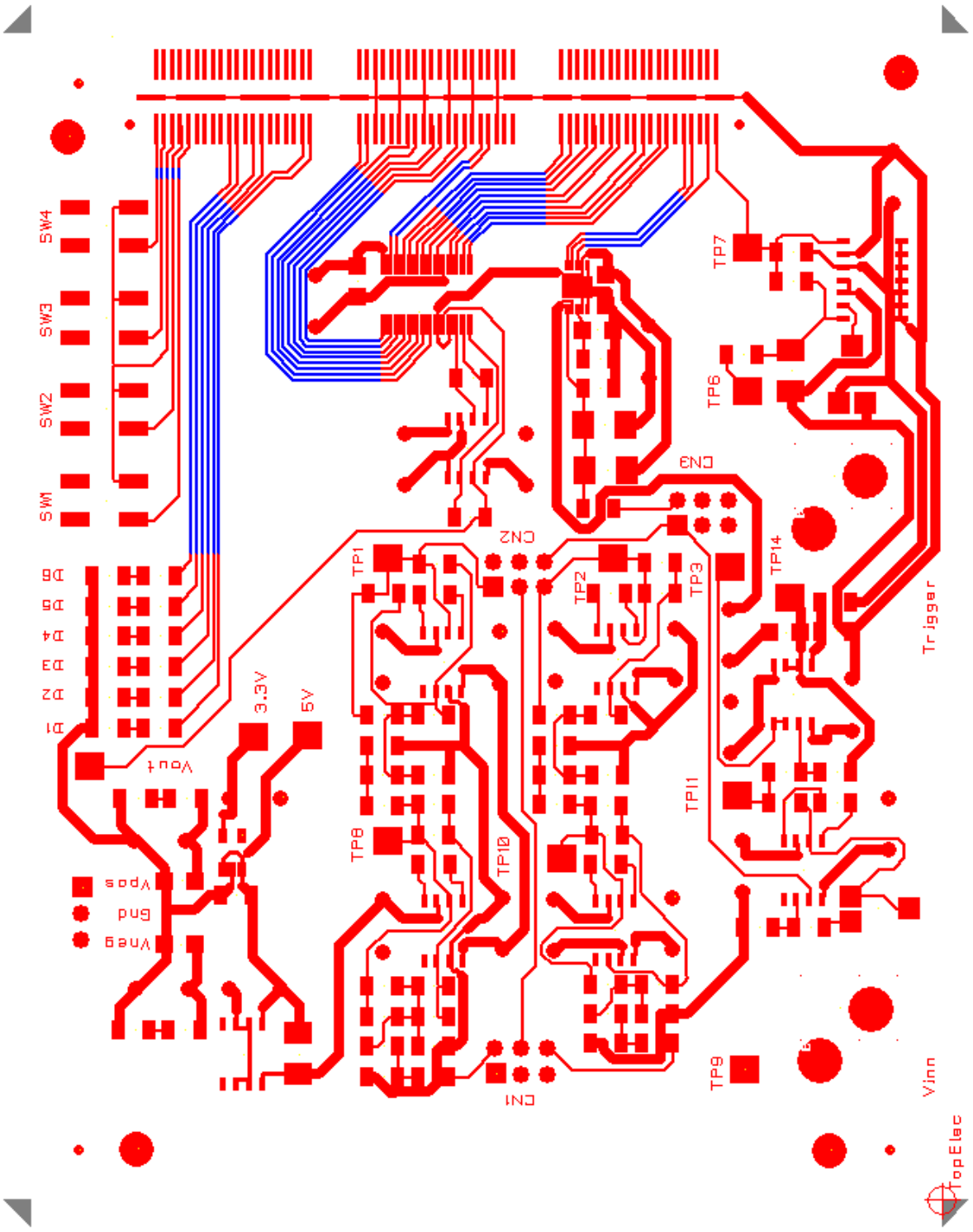
Kretskortutlegg for Zuken Cadstar 11 ligger på vedlagt CD.



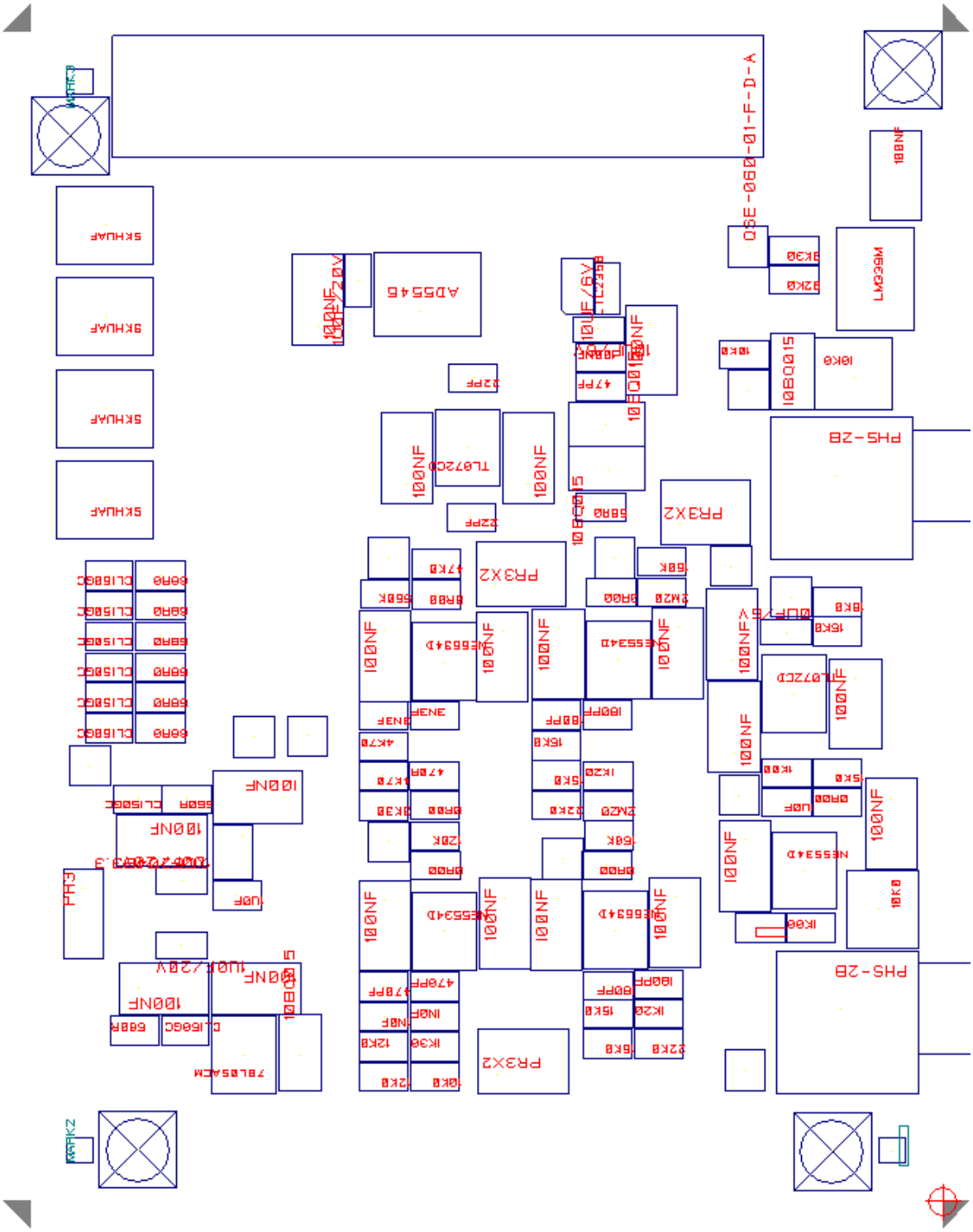

**Figur 6-12:** Alle lagene



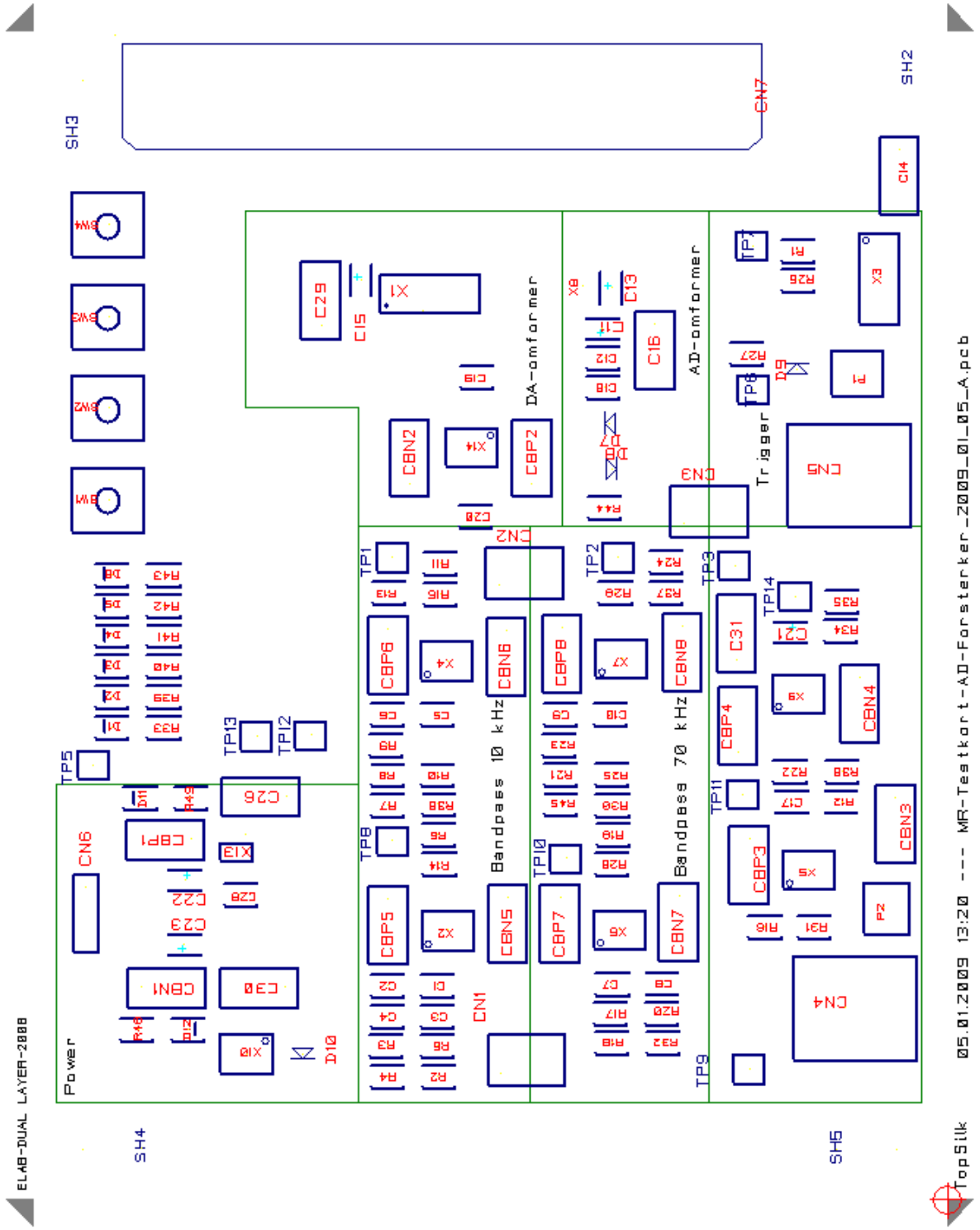
**Figur 6-13:** Ledningsbaner i bunnlaget




**Figur 6-14:** Ledningsbaner i topplaget

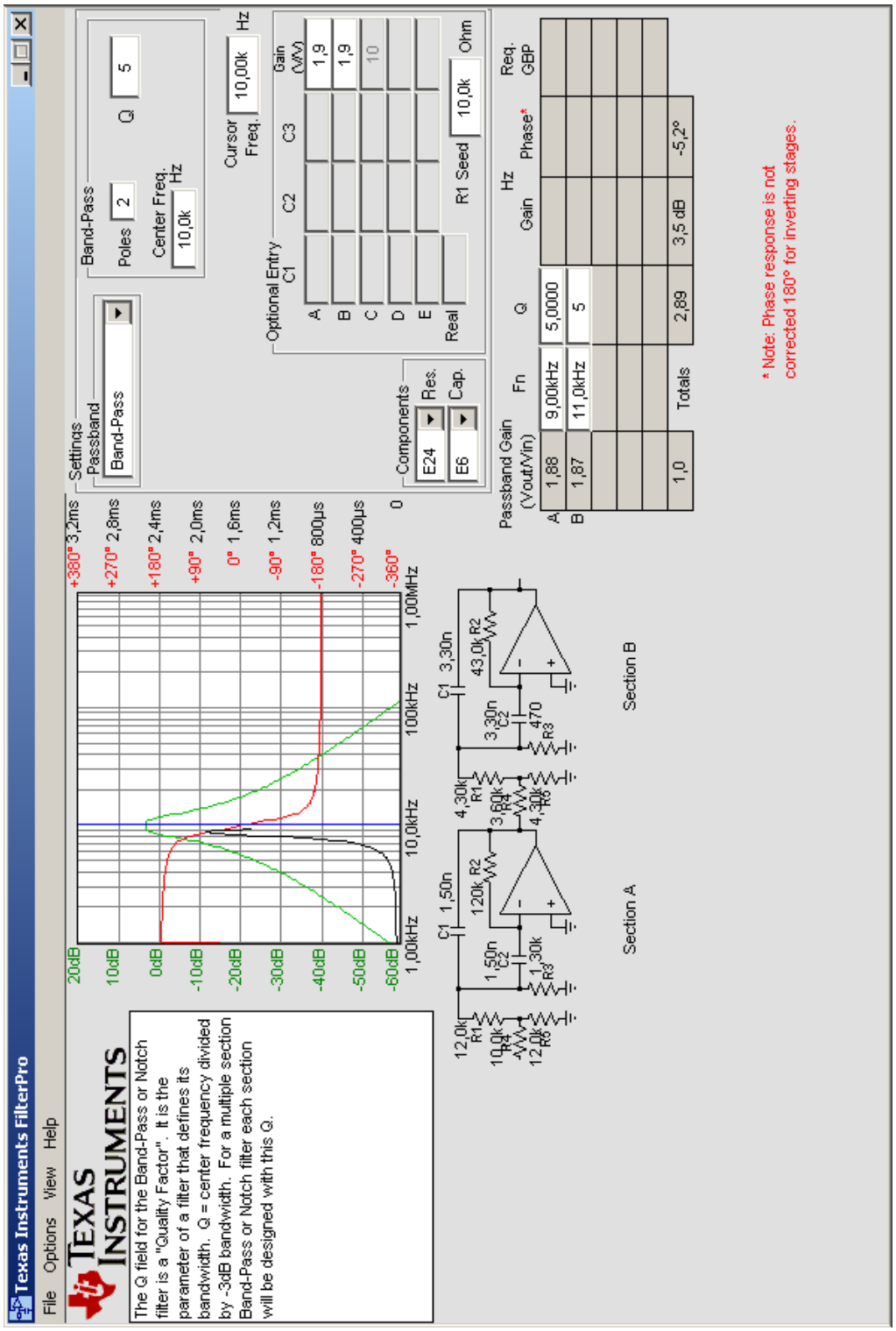



**Figur 6-15:** Komponentplassering

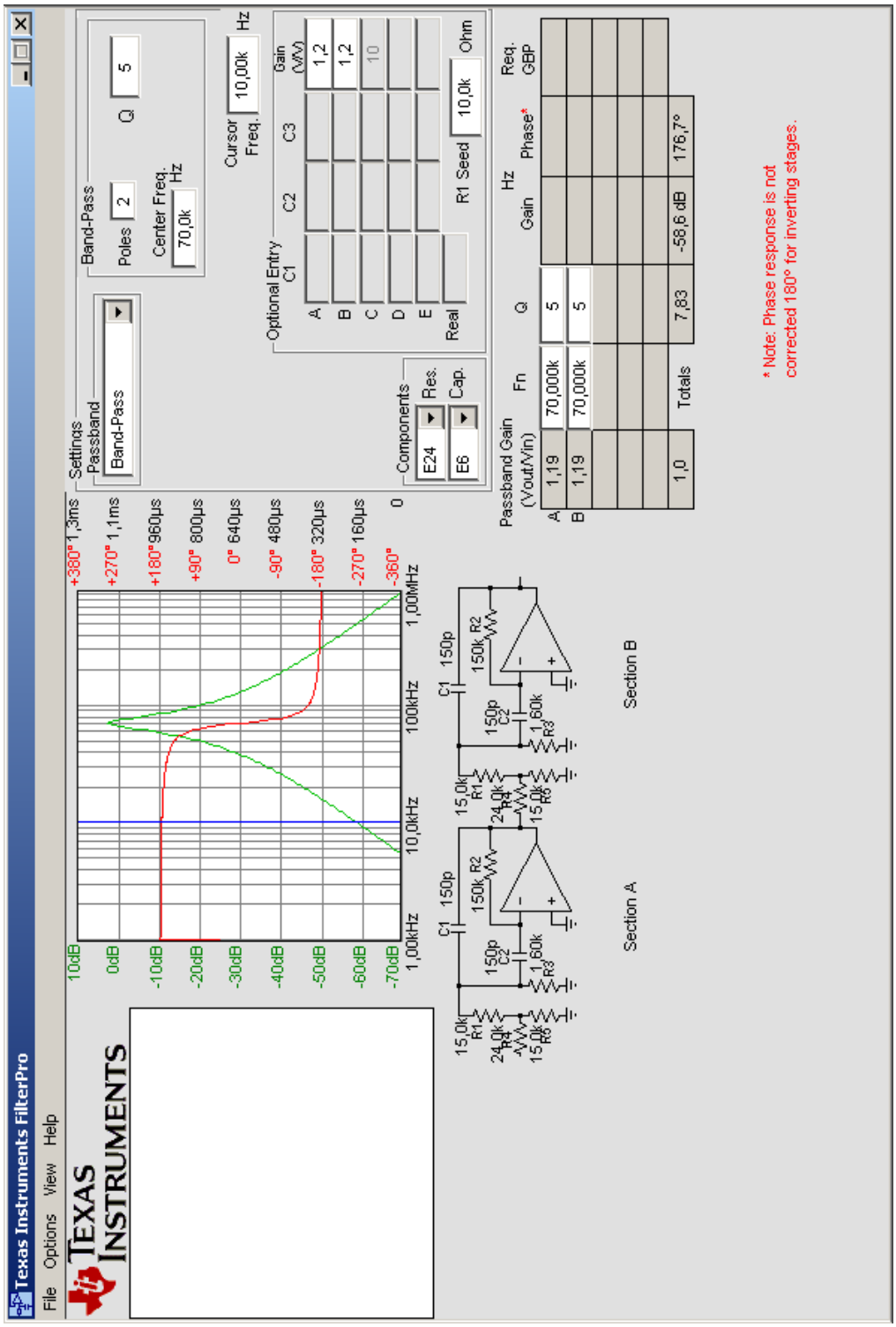



**Figur 6-16:** Silketrykk

# E. Parametre for båndpassfiltrene




**Figur 6-17:** Valgte parametre for båndpassfilter med senterfrekvens på 10 kHz (Skjermbilde fra Texas Instruments FilterPro 2.0)




**Figur 6-18:** Valgte parametre for båndpassfilter med senterfrekvens på 70 kHz (Skjermbilde fra Texas Instruments FilterPro 2.0)

# F. Doxygen – dokumentasjon av kildekode

Kommentarer som er skrevet i kildekodene er skrevet slik at det kan genereres dokumentasjon med Doxygen [Dimitri van Heesch, Doxygen]. Kommentarer for å dokumentere funksjoner, variabelstrukturer og lignende blir hentet ut av Doxygen som deretter lager dokumentasjon basert på dette. Det er dermed enkelt å holde dokumentasjonen oppdatert, og spesialkommandoene som brukes (se kapittel F.1) hjelper til med å strukturere dokumentasjonen i kildekoden også. Doxygen kan generere dokumenter i blant annet HTML, RTF, Latex og PDF.

For at Doxygen skal kunne hente ut kommentarer må de starte med en spesiell sekvens. Disse varierer etter hvilket språk som benyttes [Dimitri van Heesch, Documenting the code].

- C
  - C-kode kan dokumenteres på flere måter, men vi har valgt å starte linjekommentarer med //!, mens blokkommentarer startes med /** og avsluttes med */ (merk utropstegnet og doble stjerner).
- VHDL
  - Kommentarer starter med --! og varer ut linja (merk utropstegnet). Hvis sekvensen repeteres flere linjer under hverandre regner Doxygen det som en kommentarblokk.
- Python
  - Kommentarer kan være i docstring-sekvenser, altså det som står med tre doble anførselstegn på hver side, for eksempel ”””Kommentar”””. Docstringer er en del av programmeringsspråket Python, og støttes av mange dokumentasjonsverktøy. Dessverre støttes ikke spesialkommandoer (se kapittel F.1) hvis man bruker docstring.
  - Kommentarer kan også starte med ## og varer ut linja (merk dobbel skigard). Spesialkommandoer (se kapittel F.1) fungerer ved slike kommentarer.

## F.1 Spesialkommandoer

Det finnes en rekke spesialkommandoer som kan brukes til å formaterer utseendet. Vi har valgt å starte alle spesialkommandoer med @, men \ kan også brukes. Eksempler på spesialkommandoer kan være @param og @return som gjør at ordet som kommer etter tolkes som henholdsvis et parameter og en returverdi fra en funksjon, og formateres deretter [Dimitri van Heesch, Special commands].

Spesialkommandoene skrives med må bokstaver, men for å få både Eclipse og Doxygen til å liste opp elementer på gjørelista startes de med @TODO. Eclipse krever store bokstaver, så for å få Doxygen til å tolke det riktig må man legge til følgende i Doxyfile:

ALIASES = TODO=@todo

# G. Light Weight IP (lwIP) for Xilinx FPGA-er

Light Weight IP (lwIP) er en TCP/IP-stack som er laget spesielt for innebygde systemer (eng. embedded systems). Under utviklingen er det derfor lagt vekt på at den skal bruke så lite ressurser som mulig, samtidig som den tilbyr blant annet protokollene IP, TCP, UDP, ICMP, IGMP, DHCP, ARP m.m.

lwIP ble opprinnelig laget av Adam Dunkels ved Swedish Institute of Computer Science, men blir i dag vedlikeholdt og videreutviklet som et fri kildekode-prosjekt (BSD-lisens).

## G.1 Begrensninger i denne guiden

- lwIP er tilpasset en rekke ulike systemer, men denne guiden tar for seg hvordan man bruker lwIP for **Xilinx** FPGA-er.
- Xilinx sine MicroBlaze- og PowerPC (405 og 440)-prosessorer kan brukes med lwIP, men her gjennomgås kun hvordan det gjøres for **MicroBlaze**-prosessorer.
- Det er mulig å bruke lwIP med både Ethernetlite (xps_ethernetlite) og TEMAC (xps_ll_temac) Ethernet MAC-kjernene fra Xilinx, men kun **Ethernetlite** benyttes her.
- I denne guiden er **ISE 10.1.03, XPS 10.1.03 og SDK 10.1.03** benyttet for å konfigurere FPGA-en.
- Det oppsatte systemet er testet på Xilinx Spartan 3A DSP XtremeDSP Starter Platform.

## G.2 Ressurser på internett

| | | |
|---|---|---|
| **lwIP's offisielle hjemmeside** | http://savannah.nongnu.org/projects/lwip/ | |
| **lwIP's opprinnelige hjemmeside** | http://www.sics.se/~adam/lwip/ | Blir IKKE vedlikeholdt lenger. |
| **lwIP Wiki** | http://lwip.wikia.com/wiki/LwIP_Wiki | Beskrivelse av lwIP og hvordan den brukes |
| **lwIP dokumentasjon** | http://www.nongnu.org/lwip/ | Dokumentasjon av alle kildefiler for siste versjon av lwIP. |
| **Dokumentasjon av den første utgaven av lwIP.** | http://www.sics.se/~adam/thesis.pdf eller ftp://ftp.sics.se/pub/SICS-reports/Reports/SICS-T--2001-20--SE.pdf | Noe av innholdet er utdatert, men den forklarer hovedprinsippene på en grundig måte. |
| **E-postliste for lwIP** | http://www.mail-archive.com/lwip-users@nongnu.org/ | Søkbart arkiv over meldinger som er sendt på e-postlisten. |
| **Xilinx brukerforum** | http://forums.xilinx.com/xlnx/ | Brukerforum for alt som har med Xilinx sine produkter å gjøre, og har derfor mange poster om lwIP for Xilinx. |
| **Xilinx lwIP eksempel**<br><br>**xapp1026** | http://www.xilinx.com/support/documentation/application_notes/xapp1026.pdf | Eksempel for ML505, ML403 eller Spartan-3AN Starter Development Board med eksempelfiler |
| **AVNET lwIP eksempler** | https://www.em.avnet.com/common/filetree/0,2740,RID%253D0%2526CID%253D42106%2526CAT%253D0%2526CCD%253DUSA%2526SID%253D32214%2526DID%253DDF2%2526SRT%253D1%2526LID%253D32232%2526PRT%253D0%2526PVW%253D%2526PNT%253D%2526BID%253DDF2%2526CTP%253DEVK,00.html | Krever registrering hos AVNET. Denne lenken peker på eksempelfiler for Xilinx Spartan 3A DSP XtremeDSP Starter Platform. |

## G.3 Ressurser på harddisk når Xilinx EDK er installert

| | | |
|---|---|---|
| **lwIP på lokal harddisk**<br><br>*<lwIP på lokal harddisk>* | *<EDK's plassering på harddisken>*\sw\ThirdParty\sw_services*<lwIP versjon>*<br><br>f.eks.<br>C:\Xilinx\EDK\sw\ThirdParty\sw_services\lwip130_v1_00_a | Ligger i denne mappen når Xilinx EDK er installert på maskinen. |
| **Dokumentasjon av lwIP**<br><br>*<dokumentasjon av lwIP>* | *<lwIP på lokal harddisk>*\src*<lwip versjon>*\doc<br><br>f.eks.<br>C:\Xilinx\EDK\sw\ThirdParty\sw_services\lwip130_v1_00_a\src\lwip-1.3.0\doc | |
| **Dokumentasjon av lwIP for Xilinx** | *<lwIP på lokal harddisk>*\doc*<lwip versjon>.pdf*<br><br>f.eks.<br>C:\Xilinx\EDK\sw\ThirdParty\sw_services\lwip130_v1_00_a\doc\lwip130_v1_00_a.pdf | Inneholder oppsett av maskinvare og programvare, konfigureringsalternativer, Xilinx-funksjoner og eksempler. |
| **Dokumenasjon av raw-API-et** | *<dokumentasjon av lwIP>\rawapi.txt*<br><br>rawapi.html er mitt forsøk på å gjøre rawapi.txt (for versjon 1.3.0) leselig (vedlagt). | |
| **Kildefiler for lwIP** | *<lwIP på lokal harddisk>*\src*<lwip versjon>*\src<br><br>f.eks.<br>C:\Xilinx\EDK\sw\ThirdParty\sw_services\lwip130_v1_00_a\src\lwip-1.3.0\src | .h og .c filer ligger her og i undermappene. |
| **Kildefiler for Xilinx' tilpasning av lwIP** | *<lwIP på lokal harddisk>*\src\contrib\ports\xilinx\<br><br>f.eks.<br>C:\Xilinx\EDK\sw\ThirdParty\sw_services\lwip130_v1_00_a\src\contrib\ports\xilinx | .h og .c filer ligger her og i undermappene. |

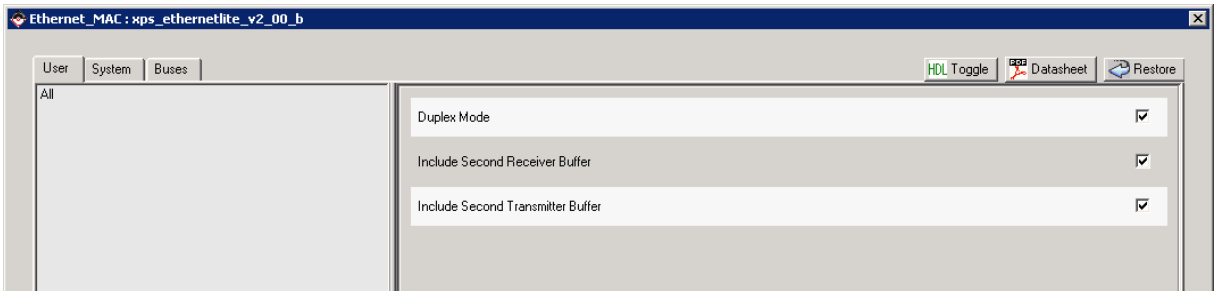
## G.4 Oppsett av prosessorsystemet i XPS

For at lwIP skal kunne brukes må man sette opp FPGA-systemet med minimum følgende komponenter:

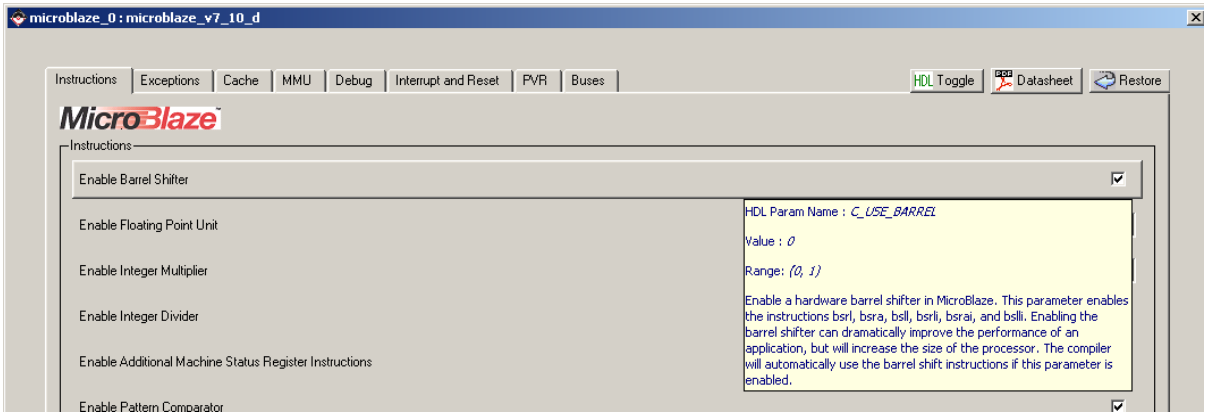
- Prosessor (MicroBlaze eller PowerPC)
- **Ethernet MAC med avbrudd** (xps_ethernetlite eller xps_ll_temac)
- **Timer med avbrudd** (xps_timer).
- **Avbruddskontroller** (xps_intc). Støttefunksjonene fra Xilinx er basert på avbrudd, så for å få koden til å fungere er man avhengig av en avbruddskontroller.

Hvis man starter et nytt design, kan alt dette settes opp i Base System Builder (BSB). Hvis man allerede har et prosessorsystem, må man legge til de komponentene man mangler og koble dem til riktig.

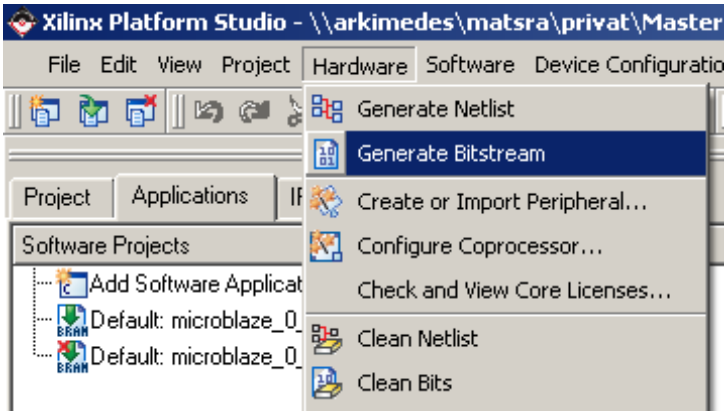
# Konfigurere enheter




**Figur 6-19** Dobbeltklikk på Ethernetlite og velg å inkludere ekstra buffere for sending og mottak av pakker. Dette øker hastigheten på applikasjonen.



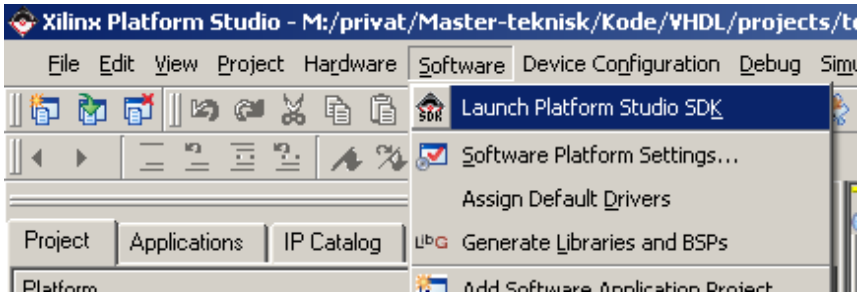

**Figur 6-20** Dobbelklikk på MicroBlaze-prosessoren og velg å inkludere ”Barrel Shifter”. Dette øker hastigheten på applikasjonen.




**Figur 6-21** Når hele prosessorsystemet er satt opp velges ”Generate Bitstream” hvis det kun er et XPS-prosjekt. Hvis ISE ble benyttet for å sette opp systemet må man velge ”Genereate Programming File” i ISE. Disse stegene tar lang tid (ca. 20-30 minutter).

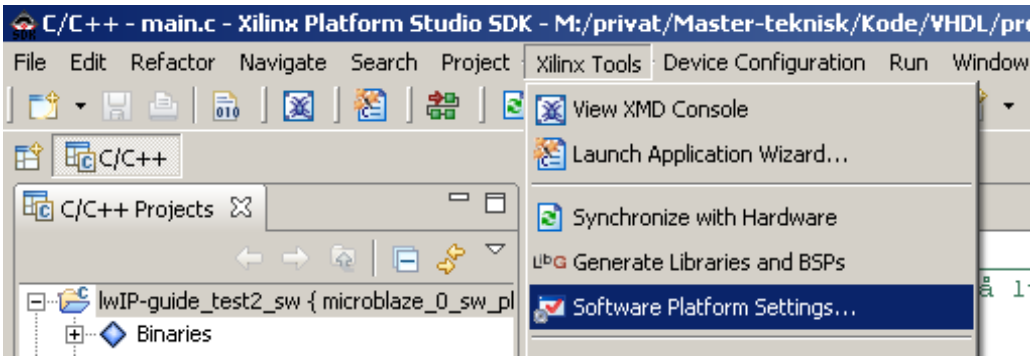
## G.5 Endringer i SDK

Prosessorsystemet er nå ferdig generert. Det neste som må gjøres er å endre noen innstillinger i SDK.

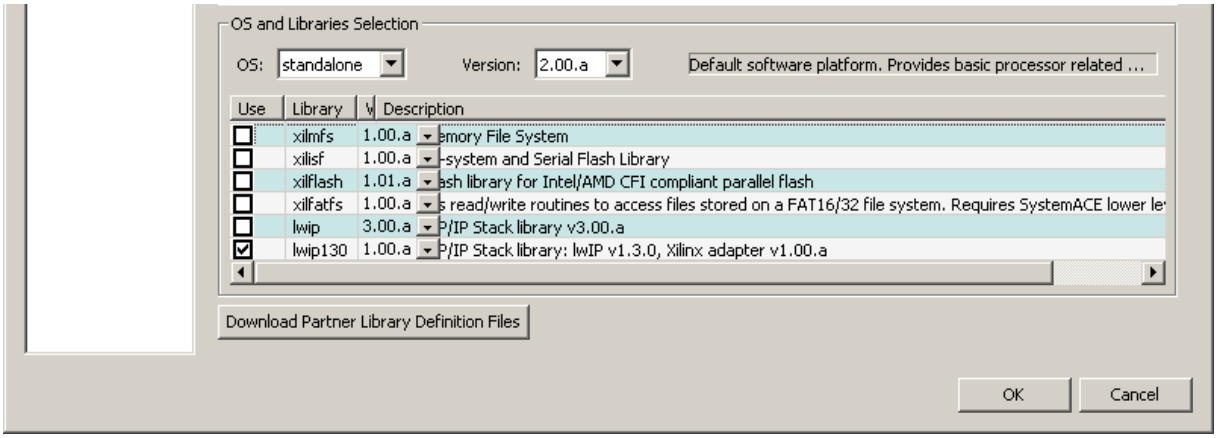



**Figur 6-22** Start SDK og opprett et nytt programvareprosjekt.

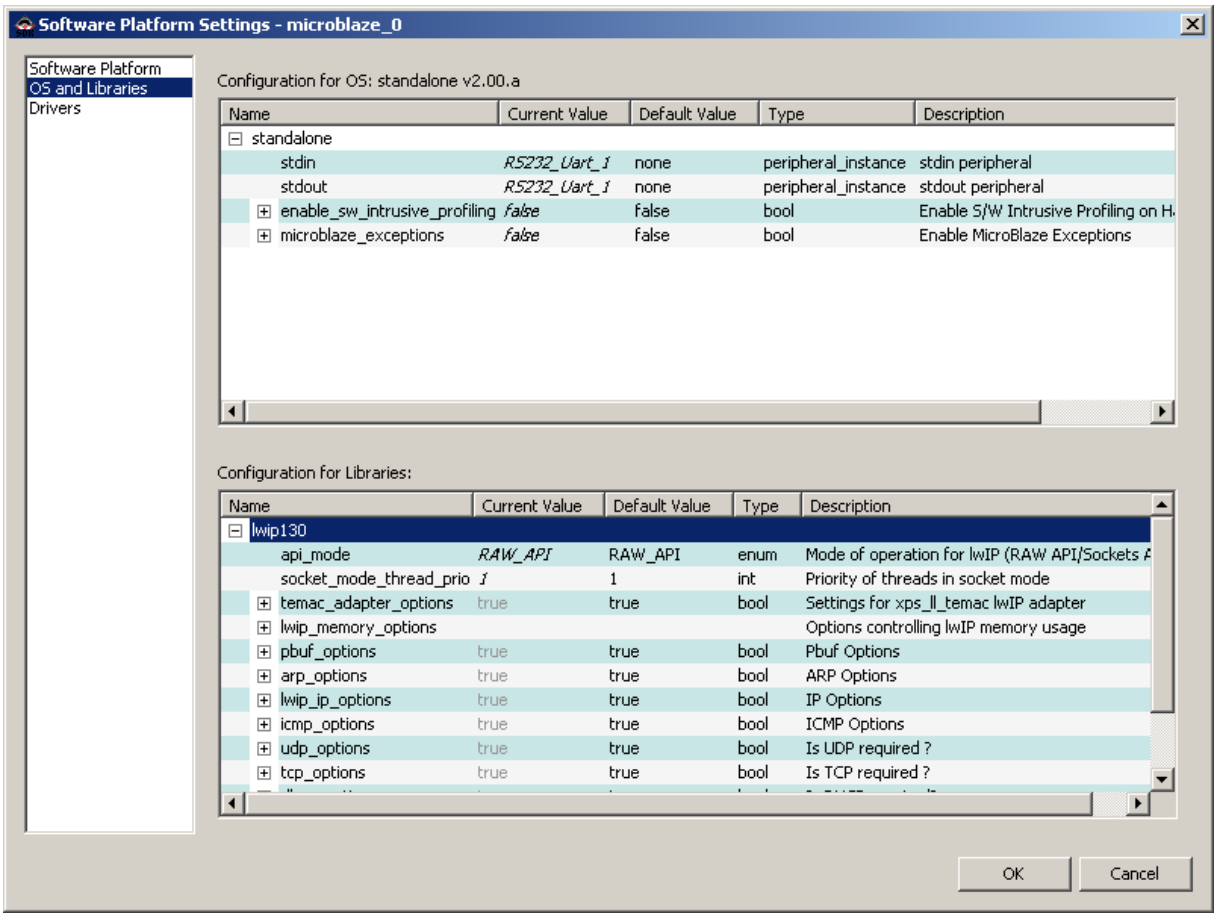
## G.6 Konfigurere lwIP




**Figur 6-23** Velg "Software Platform Settings…"

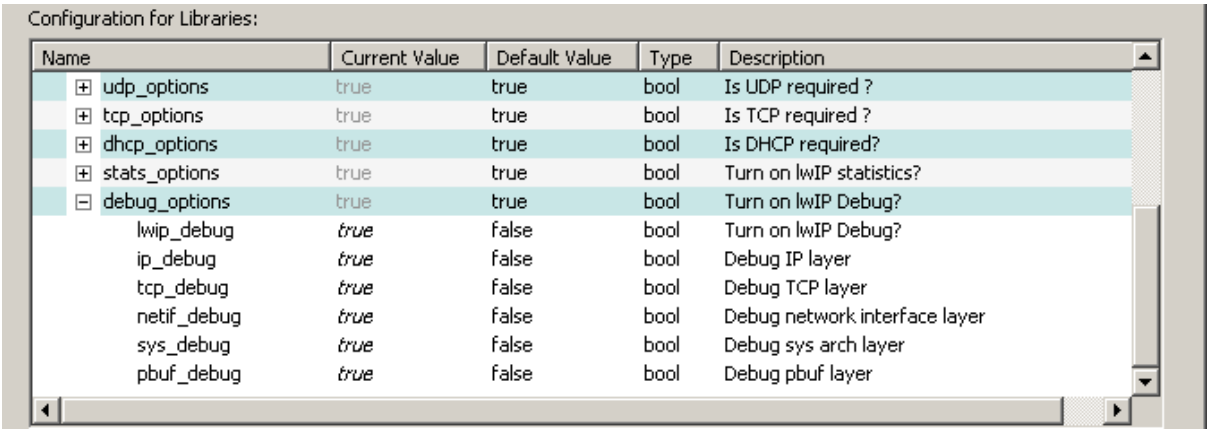



**Figur 6-24** Velg "Software Platform" i menyen på venstre side, og huk av for den siste versjonen av lwIP under "OS and Libraries Selection".




**Figur 6-25** Under ”OS and Libraries” vil nå den valgte versjonen av lwIP vises i vinduet for ”Configuration for Libraries”.

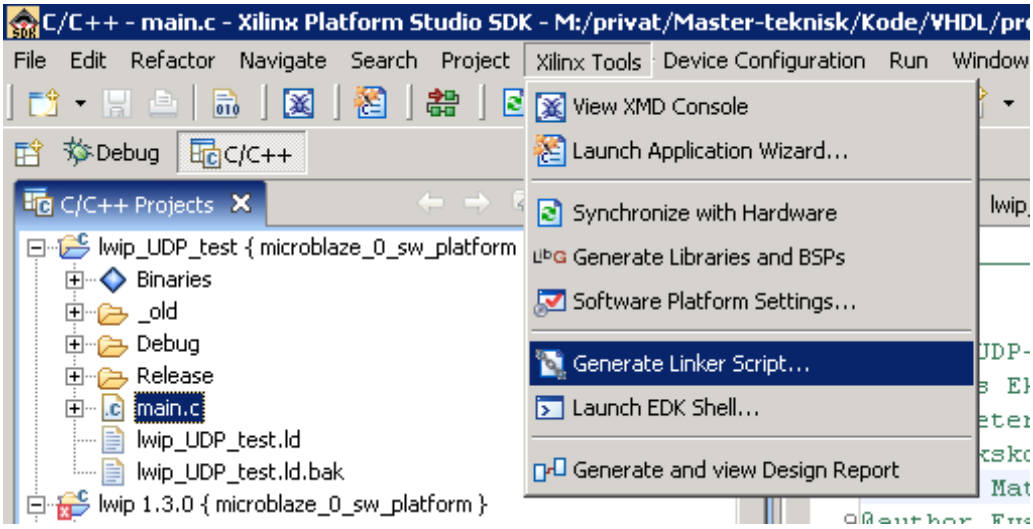
Mange av innstillingene som her kan velges er beskrevet i dokumentet som er kalt ”Dokumentasjon av lwIP for Xilinx” i ressurslisten i kapittel G.3. Standardinnstillingene kan brukes i første omgang, så kan man heller finjustere systemet etter hvert.



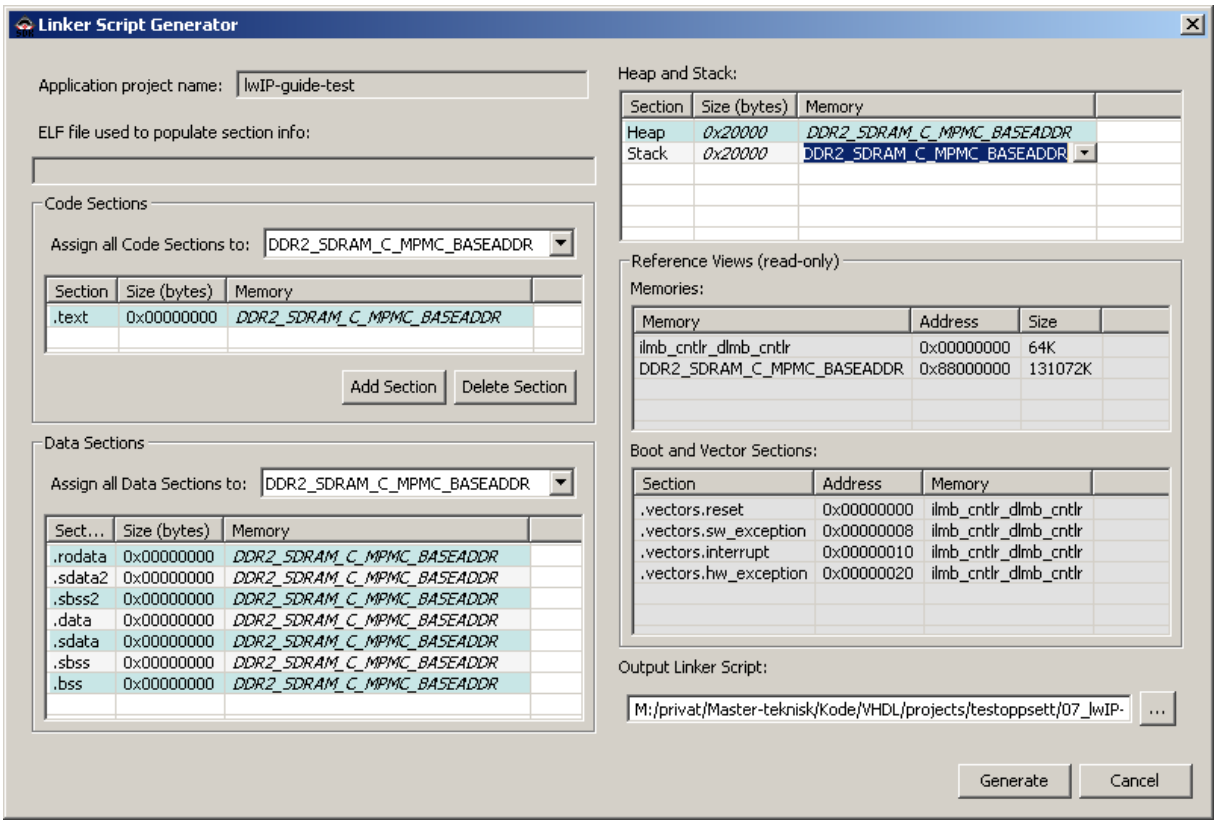

**Figur 6-26** Under utvikling av applikasjoner anbefales det å sette alle valgene under ”debug_options” til true, så lwIP skriver ut status- og feilmeldinger til standard ut (vanligvis UART RS232). Utskrift skjer kun når kompilatorflagget –g (Generate Debug Symbols) er satt (Standard for Configuration: Debug i SDK).

lwIP-biblioteket genereres automatisk etter at du har trykket på OK-knappen. Hvis ikke dette skjer må du velge ”Xilinx Tools” og ”Generate Libraries and BSPs”. Endringene som er gjort skrives til filen lwipopts.h, og biblioteket kompileres med de nye innstillingene.

## Endre linkerskriptet

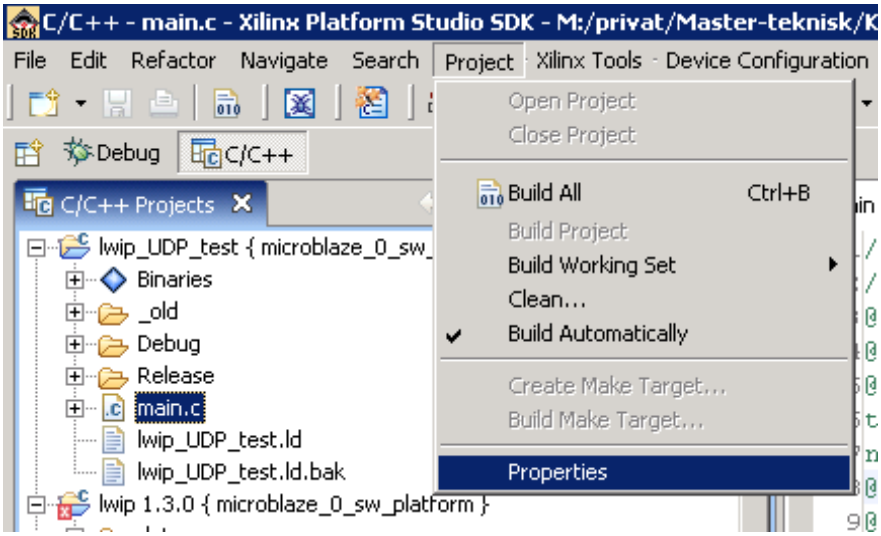



**Figur 6-27** Velg ”Generate Linker Script…”. Hvis man har flere programvareprosjekter må man deretter velge hvilket prosjekt man nå skal lage linkerskript for.

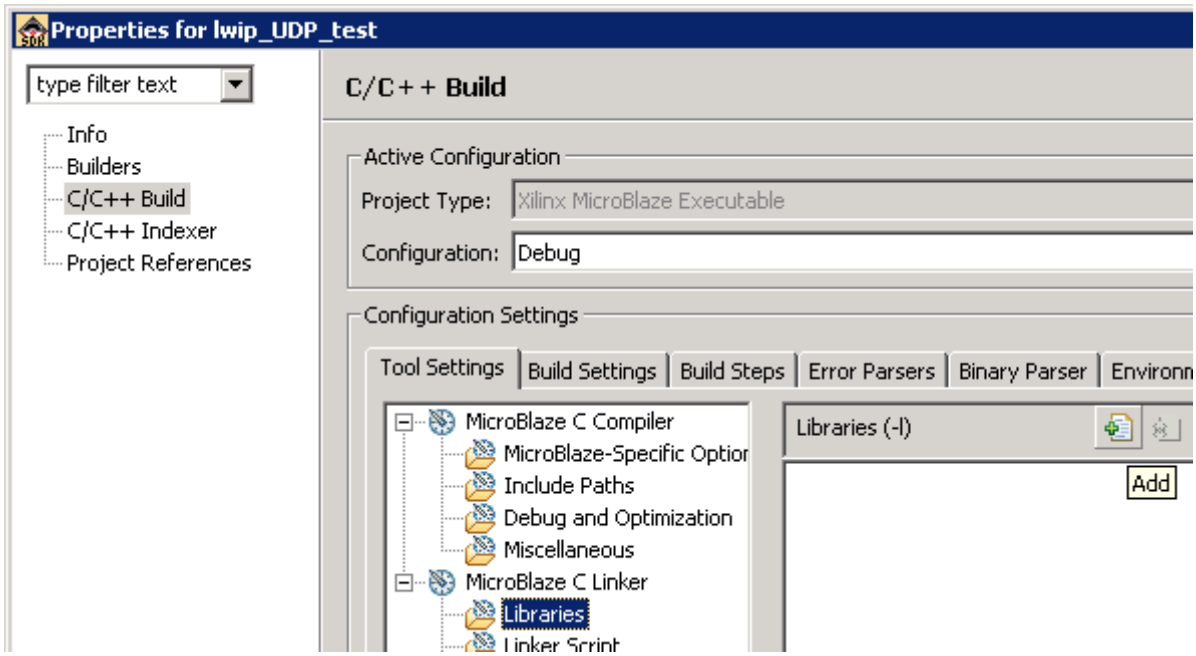



**Figur 6-28** For at lwIP skal kjøre riktig er det veldig viktig at heap og stack er store nok. Standardinnstillingen på 0x400 er alt for lite. I eksemplene fra Xilinx og AVNET er begge satt til 0x20000.

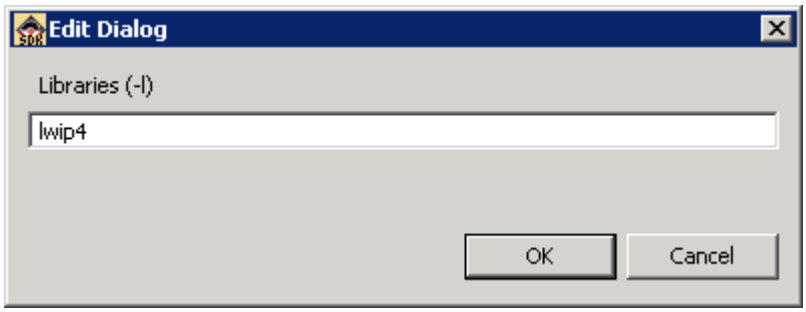
## Sette linkerflagg for lwIP



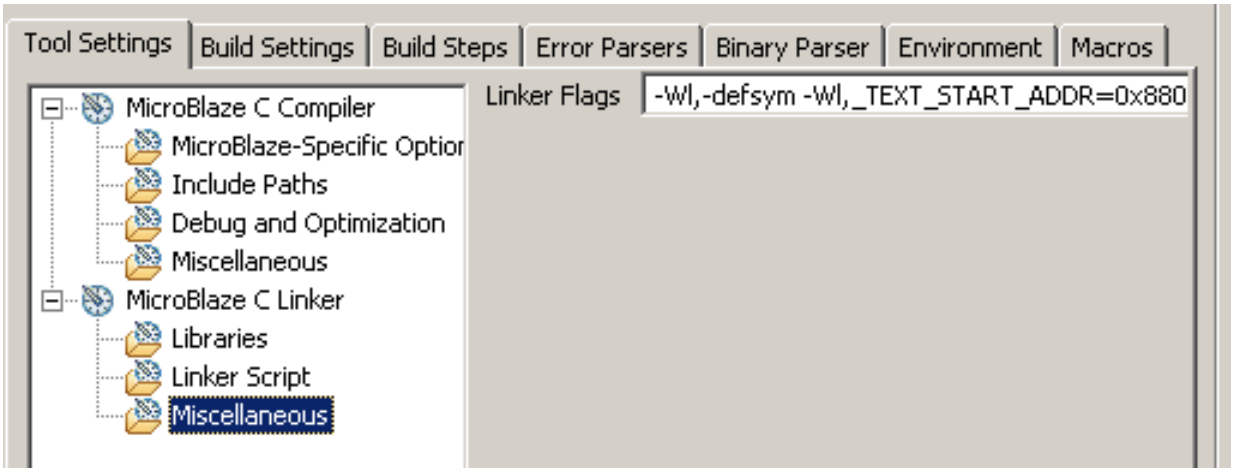

**Figur 6-29** Velg ”Properties” for det aktuelle prosjektet.




**Figur 6-30** Under ”C/C++-build”, “MicroBlaze C Linker” og “Libraries” velger man å legge til et nytt bibliotek.




**Figur 6-31** Skriv inn lwip4 og trykk ”OK”.

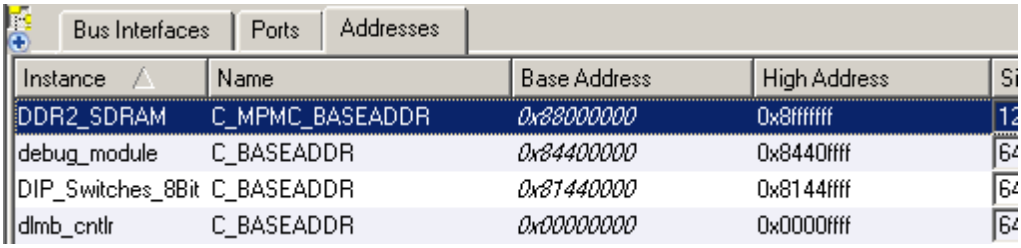



**Figur 6-32** Under ”Miscellaneous” må man legge til et linkerflagg for å sette startadressen for programmet til eksternt minne, fordi programmene fort er for store til å kunne kjøres fra BRAM.

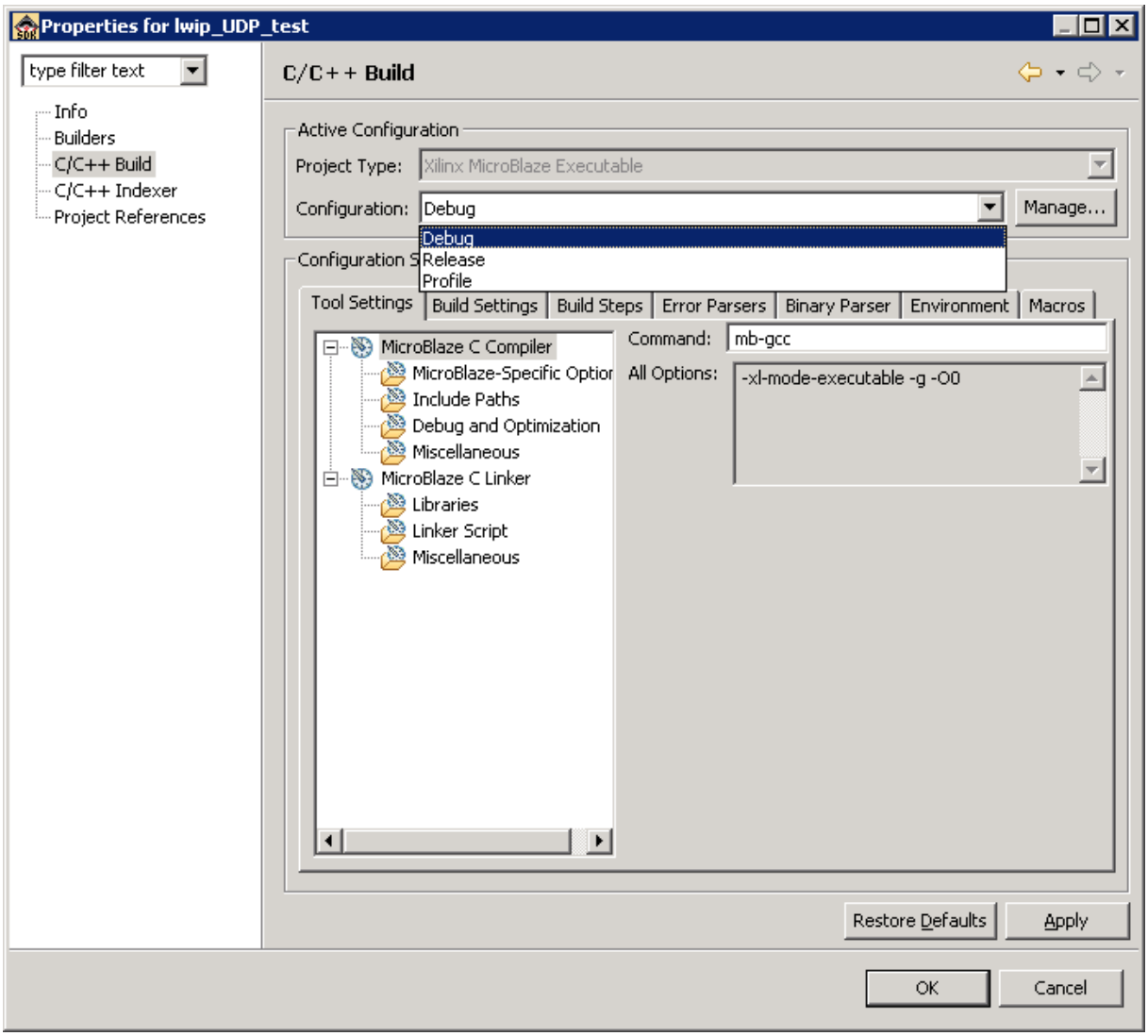
Følgende setning limes inn i tekstboksen for “Linker Flags”:

-Wl,-defsym -Wl,_TEXT_START_ADDR=0x88000000

Det siste tallet (0x88000000) er adressen til det eksterne minnet. Denne er avhengig av hvordan systemet er satt opp og må byttes ut med adressen man finner under ”Addresses” i XPS.




**Figur 6-33** Adressen til det eksterne minnet finner man under ”Addresses” i XPS.




**Figur 6-34** Hvis man har flere konfigurasjoner må man repetere de siste stegene for alle konfigurasjonene så det er riktig når man senere skifter konfigurasjon.

Når filen kompilerer uten feil, er man klare for å teste med det vedlagte programmet.

# H. Testutstyr

For å teste kretsene er følgende utstyr benyttet:

- **Funksjonsgenerator:** Good Will Instruments, Inc. GFG-81 Function Generator.
- **Oscilloskop:** Tektronix TDS 1002. Tokanals digitalt oscilloskop. 60 MHz. 1 GS/s.
- **Strømforsyning:** ISO-TECH IPS1125. To utganger med 0-20 V DC (1/4 A).
- **Kassettspiller:** SONY TC-D5M

# I. Vedlegg på CD og Internett

Den vedlagte CD-en inneholder følgende:

- Installasjonsfiler for Pyton, Blt, Pmw, filterprogrammer og annen fritt tilgjengelig programvare.
- Hele oppgaven som word-dokument
- All kildekode (VHDL, C, Python)
- Komponenter som er laget for CADSTAR
- Kretskortutlegg og skjemategninger for CADSTAR

Mesteparten av dette er også tilgjengelig på http://www.randgaard.org/HADAS2010 slik at lesere som finner oppgaven via www.duo.uio.no kan få tilgang til filene.