Sensur av hovedoppgaver
Høgskolen i Buskerud
Avdeling for Teknologi

Prosjektnummer: 2011-09
For studieåret: 2010/2011
Emnekode: SFHO-3200

Prosjektnavn:
Multimikrofonsystem
Multi Microphone System

Utført i samarbeid med: Posicom AS

Ekstern veileder: Ragnvald Otterlei

Sammendrag: Denne prosjektoppgaven har gått ut på å lage en hardwareprototype bestående av et AD-kort for digitalisering av data og et FPGA-system for videresending over Ethernet. Vi lagde et kretskort med åtte kanaler. Mikrofonsignalene digitaliseres med en samplingfrekvens på 96 kHz og 24 bits oppløsning. Etter digitaliseringen blir dataene sendt til FPGA-kortet over en I^2S-buss. FPGA-en pakker dataene i UDP-pakker, sender de til en mottaks-PC og et PC-program vi har skrevet lagrer dataene til fil.

Stikkord:
- AD-konvertering
- Kretskortutlegg
- FPGA-utvikling

Tilgjengelig: Ja

Prosjekt deltagere og karakter:

Navn	Karakter
Kathrine Kaasa	
Knut Olav Skyttemyr	
Jardar Hvidsten	
Fredrik Bakke	
Erlend Rauk Aasland	

Dato: 9. Juni 2011

Sigmund Gudvangen	Olaf Hallan Graven	Ragnvald Otterlei
Intern Veileder	Intern Sensor	Ekstern Sensor

Prosjektgruppe 2011-09 bekrefter herved at arbeidet som beskrives i disse dokumentene er helt og fullt vårt eget verk.

Erlend Rauk Aasland	Fredrik Bakke
Jardar Hvidsten	Kathrine Kaasa

Knut Olav Skyttemyr

I denne permen finnes følgende dokumenter:

Fane	Nr.	Navn	Versjon	Sideantall
1	P.1	Visjonsdokument	1.b	6
2	P.2	Forstudie	1.c	16
3	P.3	Kravspesifikasjon	1.d	14
4	P.4	Testspesifikasjon	1.f	13
5	P.5	Prosjektplan	1.d	20
6	S.2	Brukermanual	1.a	20
7	S.3	Hovedrapport	1.a	62
8	S.5	Mottaksprogram	1.a	13
9	S.6	Programvare på NIOS	1.a	7
10	S.7	Mottaksmodul	1.a	10
11	S.8	Etteranalyse	1.a	9
12	T.3	Systemtest	1.a	12
12	T.4	Testrapport – AD-kort	1.a	19
12	T.5	Testrapport – FPGA/PV	1.a	21
12	T.6	Testrapport – ytelse	1.a	12

På CD-en finnes følgende:

- P.6 Risikoanalyse
- P.7 Endringer
- S.1 Designdokument
- S.4 Aktivitetsrapporter
- T.1 Simuleringsrapport
- T.2 Test av spenningsforsyning
- P.1 – P.7 Planleggingsdokumenter i .pdf og .odt
- S.1 – S.8 Systembeskrivelsesdokumenter i .pdf og .odt
- T.1 – T.6 Testrapporter i .pdf og.odt
- VHDL-moduler
- Mottaksprogram
- SoPC
- Visualiseringsscript for MATLAB
- Kretsskjema
- Artwork-filer AD-kort
- OrCAD-prosjekt
- Regnskap
- Timelister
- Møteinnkallinger og referater
- Oppfølgingsdokumenter
- Plakat

POSICOM

HØGSKOLEN i Buskerud

Visjonsdokument		
Dokumenttype	Dokumentnummer	Versjonsnummer
Planlegging	P.1	1.b
Dato sist endret	Frigitt dokument?	Dokumentansvarlig
27.05.2011	Ja	Knut Olav Skyttemyr
Gruppemedlemmer:		
Kathrine Kaasa, Knut Olav Skyttemyr, Jardar Hvidsten, Fredrik Bakke, Erlend Rauk Aasland		

Innholdsliste

1 Kontaktinfo....3
2 Introduksjon....4
3 Om prosjektoppgaven....5
3.1 Hvorfor?....5
3.2 Hva?....5

1 Kontaktinfo

Navn	Alder	Linje	E-post	Telefon
Kathrine Kaasa	21	Audioteknologi	kaasa3@live.no	48126516
Erlend Rauk Aasland	25	Embedded systems	erlend.aasland@gmail.com	41203446
Fredrik Bakke	25	Embedded systems	bakkefredrik@gmail.com	97591496
Jardar Hvidsten	22	Mekatronikk	jardar_h@hotmail.com	40479488
Knut Olav Skyttemyr	27	Audioteknologi	ko.skyttemyr@gmail.com	91634894

Oppdragsgiver	Ekstern sensor	Ekstern veileder
Posicom AS	Ragnvald Otterlei	Ragnvald Otterlei
Dyrmyrgata 35	ragnvald@posicom.no	
3611 Kongsberg	92030141	

2 Introduksjon

Dette dokumentet har som mål å gi en overordnet beskrivelse av systemet som skal lages i løpet av prosjektet. Det tar for seg hvorfor oppdragsgiver ønsker å gi denne oppgaven og hva systemet skal gjøre. Endelig godkjenning av prosjektet skjer på bakgrunn av dette dokumentet og dokumentet danner grunnlag for en mer detaljert beskrivelse av systemet.

3 Om prosjektoppgaven

3.1 Hvorfor?

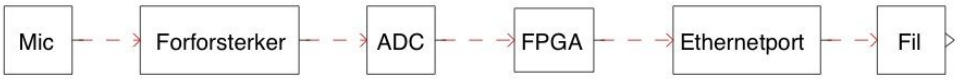
Snipos (Sniper Positioning System) er et system som bruker mikrofoner for å detektere retning og avstand til avfyrte våpen. Våpenprosjektil beveger seg med overlydshastighet og avgir en kraftig lydbølge (sjokkpuls). Selve våpenet avgir også en kraftig lydbølge (munningspuls), men styrken på denne avtar ettersom avstanden øker. På lange avstander kan det derfor være vanskelig å fange opp munningspulsen. Andre forhold som lokalt støy kan også føre til at munningspulsen blir vanskelig å fange opp.

Oppdragsgiver er interessert i å øke rekkevidden til systemet. Mer konkret vil det da si å kunne fange opp munningspulsen fra lengre hold enn dagens system, og/eller ved høye støynivå.

Ph.d. stipendiat ved HiBu, Antonio Ramos, jobber deltid for Snipos og en av tingene han jobber med er beamforming. Oppdragsgiver ønsker å få laget hardwaren til et høyere ordens mikrofonarray for å ta i bruk arbeidet til Antonio.

3.2 Hva?

Prosjektet vil da gå ut på å designe en hardwareprototype for beamforming ved hjelp av høyere ordens mikrofonarray. Signalkjeden vil bli som følgende:



Figur 1: Signalkjeden til systemet som skal lages.

Beamforming krever at mikrofonene festes i et bestemt mønster på et sfærisk array. For at prosjektet ikke skal bli for stort og sprikende faglig sett vil vi ikke ta for oss designet av selve arrayet, men konsentrere oss om elektronikken og databehandling. Oppdragsgiver vil ta seg av designet av dette arrayet.

Dette systemet stiller enorme krav til mikrofonene. De skal tåle høye lydtrykk (opp mot 160-170 dB SPL) og samtidig må støygulvet være lavt nok for å fange opp munningspulsen under vanskelige forhold. Det er urealistisk å tro at vi kan finne billige mikrofoner som er gode nok til å ta opp både sjokkpulsen og munningspulsen.

Det høye antall mikrofoner gjør at prisen per mikrofon er viktig, derfor vil systemet vi lager konsentrere seg om å behandle data fra mange mikrofoner samtidig. Testing av systemet kan gjøres med billige mikrofoner. Vi tar sikte på å lage en prototype, funksjonaliteten til mikrofonene

trenger derfor ikke være optimal. Fortsatt er det en utfordring å finne billige mikrofoner, for de må tåle de høye lydtrykkene fra sjokkpulsen uten å bli ødelagt rent mekanisk. I teorien er justering av forforsterkeren det eneste som må til ved bytte av mikrofontype.

Forforsterkeren skalerer signalene slik at utgangssignalet fra mikrofonene passer til inngangen på ADC-en. De høye lydtrykkene systemet blir utsatt for gjør at systemet må ha et veldig stort dynamisk omfang. Oppløsningen på ADC-en må være stor nok til at kvaliteten på opptakene blir god nok, samtidig må ikke den totale datamengden overskride det resten av systemet kan håndtere. Elektronikken må designes med EMC i bakhodet, Snipos skal i utgangspunktet være milspec. Beamforming stiller høye krav til synkronisering mellom kanalene i og med at algoritmen baserer seg på helt synkrone signaler. FPGA-en må programmeres til å styre systemet og sende data videre over Ethernet. Dersom det er behov for å utvide oppgaven kunne vi tenke oss å se litt på signalbehandling på FPGA-en.

POSICOM

HØGSKOLEN i Buskerud

Forstudie		
Dokumenttype	Dokumentnummer	Versjonsnummer
Planlegging	P.2	1.c
Dato sist endret	Frigitt dokument?	Dokumentansvarlig
27.05.2011	Ja	Knut Olav Skyttemyr
Gruppemedlemmer:		
Kathrine Kaasa, Knut Olav Skyttemyr, Jardar Hvidsten, Fredrik Bakke, Erlend Rauk Aasland		

Innholdsliste

1 Versjonhistorie....4
2 Innledning....5
3 Prosjektomgivelse....6
4 Prosjektmål....7
5 Prosjektmodell....8
6 Grovinndeling av aktiviteter....9
6.1 Mikrofoner....9
6.2 Forforsterker....10
6.3 Analog til digital konvertering....10
6.4 FPGA....11
6.4.1 Krav til FPGA:....11
6.4.2 FPGA – mot ADC....11
6.4.3 FPGA – mot Ethernet....12
6.5 Ethernetkommunikasjon....12
7 Løsningsalternativer....13
7.1 Mikrofoner....13
7.2 Forforsterker....13
7.3 ADC....14
7.4 Kretskort....14
7.5 FPGA....15
7.6 Ethernet....15
8 Fremdriftsplan....16

Forkortelser og akronymer

FPGA	-	Field Programmable Gate Array
ADC	-	Analog-to-digital converter
GPS	-	Global Positioning System
SNR	-	Signal-to-noise ratio
SP	-	Sjokkpuls
MP	-	Munningspuls
IP-core	-	Intellectual Property core
LB	-	Logic Block
UDP	-	User Datagram Protocol
EMC	-	Electromagnetic compatibility
I^2S	-	Inter-IC sound (Integrated Interchip Sound)
MSB	-	Mest signifikante bit

SRAM	-	Static Random Access Memory
I/O	-	Input/Output
I^2C	-	Inter-Integrated Circuit
TCP	-	Transmission Control Protocol
MEMS	-	Micro-electro-mechanical Systems

1 Versjonhistorie

Versjon	Endringer
0.a	Grov skisse
0.b	Ny layout, prosjektomgivelse
0.c	FPGA, ethernet og løsningsalternativer
0.d	Nytt kapittel om prosjektmodell
1.a	Første frigitte versjon
1.b	Første innleverte versjon. Endret datoformater, korrekturlest.
1.c	Endelig versjon.

2 Innledning

Dette dokumentet har til hensikt å gi en mer teknologisk detaljert beskrivelse av prosjektet. Det tar også for seg prosjektmål, forskjellige løsningsalternativer og eventuelle problemer og begrensninger som kan oppstå. Dokumentet inneholder også en fremdriftsplan med tidsfrister frem til første fremføring i starten av januar 2011. For de overordnede linjene i prosjektet viser vi til visjonsdokumentet (P.1).

3 Prosjektomgivelse

Snipos (Sniper Positioning System) er et system som bruker mikrofoner for å detektere retning og avstand til avfyrte våpen. Dette prosjektet går ut på å lage et nytt multikanals mikrofon- og hardware-system som skal forbedre rekkevidden til systemet. Dette nye systemet skal sannsynligvis konsentrere seg om å fange opp lave lyder (MP - Munningspuls), mens det nåværende mikrofonsystemet skal ta seg av høye lyder (SP - Sjokkpuls).

Per dags dato bruker Snipos et 4-kanalssystem. Dette er nok til å finne retning både vertikalt og horisontalt. Mikrofonene som brukes er av medium kvalitet, de sliter litt med høye lydtrykk og løsningen med beskyttelse mot vær, spesielt regn, er ikke optimal. Mikrofonsignalene blir digitalisert med en oppløsning på 16 bit og samplingsfrekvens på 96 kHz. ADC (Analog-to-digital converter) er av type Texas Instruments PCM1804. De fire ADC-ene er satt opp slik at den ene er master og de tre andre er slave. Det er kun master som mottar klokke fra FPGA (Field Programmable Gate Array).

Snipos har selv utviklet et forforsterkerkort, dette kortet sitter relativt nærme mikrofonene. De har også utviklet et FPGA-kort der både FPGA-en og ADC-ene sitter. FPGA-en er en Altera Cyclone II. Dette FPGA-kortet sitter i en milspec datamaskin. Data fra FPGA overføres gjennom Ethernet (UDP - User Datagram Protocol) til selve datamaskinen. Bitraten til dagens system er ca. 13-14 Mbit. Algoritmen for gjenkjenning og deteksjon utføres i denne maskinen og resultatene sendes videre over Ethernet igjen til et grensesnitt som georefererer resultatene via GPS (Global Positioning System) og plotter de på et kart.

4 Prosjektmål

Hovedmålet med prosjektet er å lage en hardwareprototype som henter data fra et høyere ordens mikrofonarray og digitaliserer disse dataene. Dette systemet skal gå i sanntid hvor kravet til synkroni mellom kanalene er veldig høyt. Vi skal kunne lagre x antall sekunder med lydopptak til en fil som vi kan se på i for eksempel MATLAB eller sette opp en UDP-forbindelse i et program som Labview for å verifisere at systemet fungerer tilfredsstillende. Systemet skal også følge de krav som ellers er satt av oppdragsgiveren.

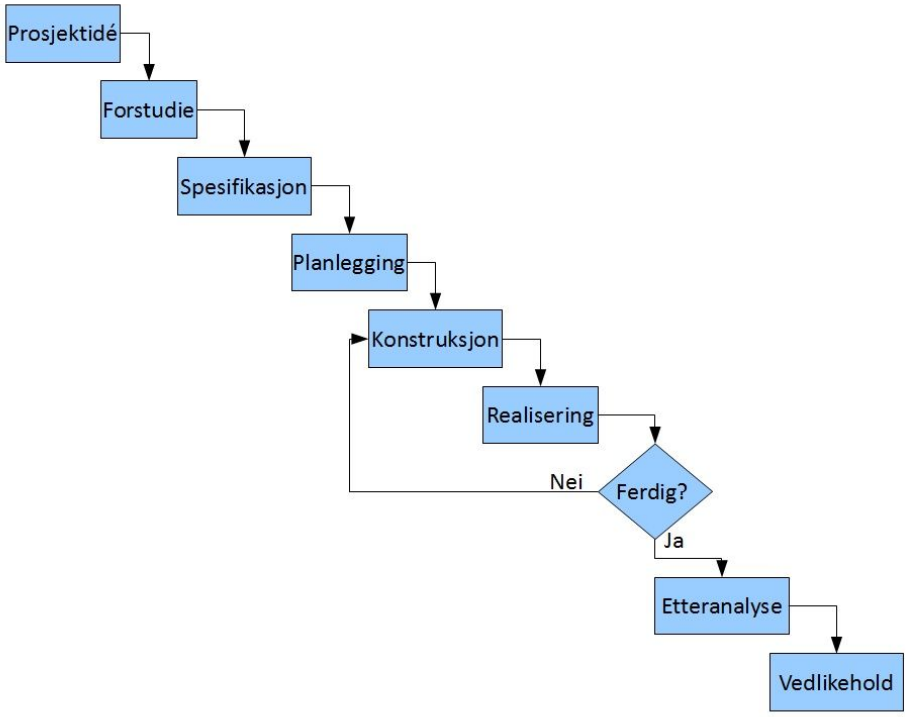
Et annet mål er å kunne følge og jobbe etter den prosjektmodellen vi har valgt for dette prosjektet.

5 Prosjektmodell

Som en del av forstudiet har vi valgt å se på hvilken prosjektmodell vi skal følge i dette prosjektet. Siden det er en prototype vi skal lage, altså en enkel versjon av produktet, vil evolusjonær utvikling passe oss best. Vi har valgt å ta utgangspunkt i inkrementell utvikling framfor prototyping som er de to undergruppene av evolusjonær utvikling. Den inkrementelle utviklingen tar utgangspunkt i at systemet bygges som separate komponenter, hvor man bygger og tester de viktigste komponentene først. Hva som er de viktigste komponentene/kravene kommer fram i kravspesifikasjonen. Når man har gjort dette går man videre med noen litt mindre viktige krav og bygger og tester disse, og slik fortsetter man til man er ferdig, se Figur 1.

Ved bruk av denne prosjektmodellen vil vi kunne få ferdig de viktige delene for systemet slik at vi kan vise at hovedprinsippet med systemet fungerer. Slik er vi nesten garantert å få noe som vi kan vise fram ved slutten av prosjektet.

En av grunnene til at inkrementell utvikling vil passe dette prosjektet er fordi vi har ganske god oversikt over hvordan systemet skal bli. Dette gjør at vi skal kunne lage en kravspesifikasjon som ikke trengs å endres mye på i løpet av prosjekttiden.



Figur 1: Flytdiagram for inkrementell utvikling.

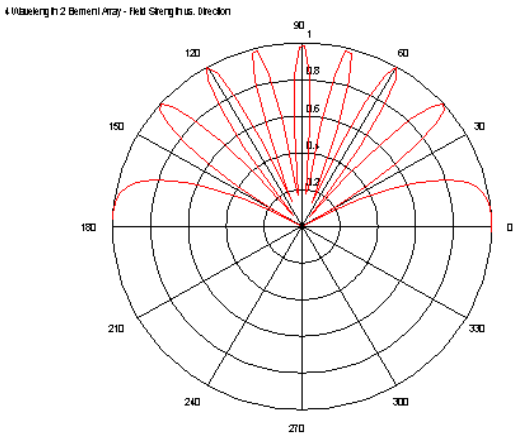
6 Grovinndeling av aktiviteter

Vi har gjort en grovinndeling av aktiviteter i prosjektet og har gjort en mer detaljert studie av de forskjellige aktivitetene.

6.1 Mikrofoner

Det finnes flere mikrofontyper, men til dette prosjektet er det sannsynligvis prepolariserte kondensatormikrofoner som egner seg best. Av de mikrofonene som finnes på markedet i dag er det blant kondensatormikrofoner vi finner de med størst SNR (Signal-to-noise ratio). I og med at mikrofonene må tåle ekstremt høye lydtrykk fra SP uten å bli ødelagt og samtidig kunne fange opp MP fra lange avstander er SNR viktig. Prepolariseringen gjør at vi slipper å tenke på polariseringsspenningen i designet av elektronikken. Samtidig er de mer robuste med tanke på å kunne fungere utendørs i regn og kulde i forhold til mikrofoner uten denne prepolariseringen.

Mikrofonene fungerer slik at de gir ut en spenning [V] når mikrofonmembranen blir utsatt for en lydbølge med et gitt trykk [Pa]. Mikrofoner blir i databladene oppgitt til å ha en viss sensitivitet [mV/Pa]. For normale lydtrykk er da spenningen ut av mikrofonen veldig lav, derfor har de fleste mikrofoner (bortsett fra dyre laboratoriemikrofoner) en transistor rett etter membranen som forsterker opp signalene. Dette er viktig fordi veldig lave spenninger er følsomme for støy, for eksempel støy fra elektriske komponenter eller elektromagnetiske forstyrrelser. Denne transistoren må forspennes for å fungere, så det må tas med i designet av elektronikken.



Figur 2: Beamforming

Mikrofoner kan lages med forskjellig retningskarakteristikk. Til beamforming er det omnidireksjonelle mikrofoner som brukes. Omnidireksjonelle mikrofoner har lik sensitivitet i alle retninger. Beamforming oppnås ved å summere bidragene fra 2^{m-1} omnidireksjonelle mikrofoner, der m er ordenen. Høyere orden fører til flere og smalere lober, og hver lobe har høyere sensitivitet, det vil i praksis si at man kan fange opp lavere lyder enn tidligere. En annen ting som

kan utnyttes er lobene. Man kan velge ut hvilke lober man vil lytte på og dermed fjerne uønskede lyder fra opptaket. Det kan være ekko fra en husvegg, eller skudd fra egne styrker som står ved siden av deg. Beamformingteknikken er aktuell i andre sammenhenger også; For eksempel under idrettsarrangement der man vil fange opp hva noen av idrettsutøverne sier og stenge ute støy fra for eksempel publikum. Et annet bruksområde er i videokonferansesystem.

6.2 Forforsterker

Forforsterkeren befinner seg i mellom mikrofonen og ADC. For det første skal forforsterkeren sette arbeidsområde for mikrofonen. ADC-en har en spesifisert maks inngangsspenning, for eksempel ±5 V. Det vil si at spenninger over ±5 V fører til at ADC-en går i metning (klipping). Da vil vi oppleve forvrengning av signalet. Forforsterkeren kan justere forsterkningen av mikrofonsignalene slik at ADC-en går i metning ved ønsket lydtrykk, som oftest vil vi at systemet skal ha litt å gå på (headroom) før ADC-en går i metning. Som nevnt tidligere kan vi oppleve enorme lydtrykk når våpenprosjektil passerer nærme. Dersom vi skal konsentrere oss om å fange opp lave lyder må vi leve med at høye lyder fører til at ADC går i metning. Det er viktig at det kritiske spenningsnivå ikke overstiges, da elektronikk kan bli ødelagt.

En annen viktig oppgave for forforsterkeren er å videreformidle spenning til mikrofonene slik at vi kan forspenne transistoren i mikrofonen.

Det finnes flere typer forforsterkerløsninger, både analoge og digitale. Disse kan igjen være inndelt i lavt støynivå og lavt forvrengningsnivå.

6.3 Analog til digital konvertering

For at FPGA-en skal kunne gjøre noe med signalet vi får fra mikrofonene trenger vi en ADC som gjør om signalet fra analogt til digitalt. For dette prosjektet vil en ADC av sigma-delta-typen være mest aktuell. Disse er vanligst til digitalisering av audiodata. Denne type konvertere tar blant annet i bruk oversampling for å unngå problemer med veldig bratte analoge anti-aliasing filtere (høy orden), da disse er vanskelige og dyre å implementere.

Antall intervaller amplituden blir delt inn i er gitt ved:

$$N = \pm 2^{B-1} \tag{1}$$

hvor B er oppløsningen på ADC gitt i bit. For en ADC med oppløsning på 16 bit betyr det at den vil ha en maksimum amplitude på ±32768 intervaller.

For mye støy er noe vi helst til unngå i stor grad. Ideelt sett er SNR gitt ved:

$$SNR \approx 6{,}02B \tag{2}$$

Til tross for dette er det vanskelig å få noe bedre SNR enn 110-120 dB selv om ADC har en høy

oppløsning (for eksempel 24 bit), grunnet støy fra analog elektronikk. Oversamplingen som blir brukt i sigma-delta konvertere kan forbedre SNR med ca. 3 dB hver gang samplefrekvensen blir doblet. Slik kan man ta i bruk lavere oppløsning på en ADC, men allikevel få god SNR.

Grensesnittet mellom ADC og FPGA-en vil være en standard som heter I^2S (Integrated Interchip Sound) og den består av en kontroller, en sender og en mottaker. Hver I^2S-buss kan behandle 2 kanaler. Når vi går til innkjøp av ADC-er vil senderen være ferdig implementert i ADC-ene. I^2S-bussen består av 3 linjer. En datalinje (SD) som sender bit-ene serielt, MSB (Mest signifikante bit) først. Så en ordvelgerlinje (WS) som skiller mellom de to kanalene. Og den siste linjen (SCLK) tar seg av klokken. Både WS og SCLK er toveis, det vil si at man kan velge fritt om sender eller mottaker skal være master. Klokkehastigheten bestemmes av oppløsning på ADC-en og samplingfrekvens. For eksempel vil en samplingfrekvens på 48 kHz og en oppløsning på 24 bit gi en klokkefrekvens på

$$\mathrm{CLK} = 48000\,\mathrm{Hz} \cdot 24 \cdot 2 = 2{,}304\ \mathrm{MHz} \tag{3}$$

Årsaken til at man må gange med 2 er at de to kanalene sendes serielt.

6.4 FPGA

FPGA-en er en sentral del av systemet vi skal bygge. Det er viktig at den har kapasitet (hastighet, LB-er (Logic Block), user I/O) til å hente data fra ADC-ene, og deretter videresende over Ethernet. Størrelsen på den interne klokkehastigheten til FPGA-en er viktig for å kunne sende videre data med høy bitrate, for eksempel Gbit. Bortsett fra det trenger vi trolig ikke mye ekstra funksjoner. Programmeringsspråket vi skal bruke er VHDL.

6.4.1 Krav til FPGA:

- Tilstrekkelig kapasitet i form av LB-er
- Klokkehastighet til å behandle data raskt nok
- Nok user I/O-pinner (Input/Output-pinner)
- Mulighet til å programmere seg selv ved hver oppstart
- Internt eller ekstern non-volatile minne for henting av program ved oppstart, og mellomlagre innsamlede data
- Tilgang til IP-cores (Intellectual Property core).

6.4.2 FPGA – mot ADC

FPGA-en tar i mot data fra ADC-ene. Det er viktig at det ikke går klokke og styresignaler over de

samme linjene som data går i, dette for å unngå jitter. I^2S-standarden har denne funksjonaliteten, i motsetning til I^2C (Inter-Integrated Circuit). Man kan velge mellom å bruke en ekstern klokke som videresendes fra ADC til FPGA. Eller så kan man sende klokke fra FPGA-kortet over I^2S-bussen til ADC-ene. Siden det er viktig at samplene fra ADC er synkroniserte, må de komme synkront inn til FPGA over separate I^2S-busser. I og med at systemet skal ha mange kanaler vil det si én I^2S-buss for hver ADC, og dette krever mange user I/O-pinner på FPGA. Mottakerdelen av I^2S-standarden implementeres ved hjelp av VHDL-kode på FPGA-brikken. Det finnes både open-source kode og kommersiell kode som har denne funksjonaliteten.

6.4.3 FPGA – mot Ethernet

Ved overføring av data fra FPGA over Ethernet brukes protokollen UDP i dagens Snipos-system. For å implementere UDP i FPGA finnes det IP-cores som er utviklet for dette. IP-core er en gjenbrukbar logisk enhet som kan implementere ulike funksjoner. På samme måte som et bibliotek i programmeringsverdenen tilbyr tjenester for et program. Selve FPGA-kortet må ha mulighet for ethernetforbindelse med en RJ45-plugg (inngang for nettverkskabel).

6.5 Ethernetkommunikasjon

Fra FPGA-kortet har vi muligheten til å sende ut de digitale signalene over Ethernet. Hvor mye data som blir generert fra lydsignalet avhenger av antall mikrofoner vi vil bruke, samt oppløsning på ADC. Dette må vi ta hensyn til i overføringen i forhold til utstyret i begge ender av nettverksforbindelsen. En annen ting som er viktig er klokkehastigheten til FPGA-en. Dersom den er for lav vil FPGA-en bli en flaskehals. Det vil være hensiktsmessig med UDP som overføringsprotokoll istedenfor TCP (Transmission Control Protocol). UDP sender uten garanti for at alle datapakker kommer fram, men ulempen for tap av pakker vil sannsynligvis være mindre enn en treg forbindelse når store datamengder overføres.

7 Løsningsalternativer

Det er viktig å ikke snevre seg inn på hvordan man skal løse problemer for tidlig. Men likevel er mye av fremgangsmåtene ganske satt allerede. Vi har gått igjennom de forskjellige aktivitetene og sett på om det finnes forskjellige måter og løse dette på.

7.1 Mikrofoner

Når det gjelder mikrofoner kan vi velge mellom å bruke så billige mikrofoner som mulig, eller prøve å finne noen som kan brukes i det endelige systemet. Det som taler for å bruke billige mikrofoner er at totalprisen på prosjektet vil synke betraktelig, avhengig av hvor mange mikrofoner vi skal bruke. For å sjekke at dataoverføringen fungerer og at kanalene er synkroniserte holder det å bruke billige mikrofoner. Uansett bør vi gjøre en markedsundersøkelse der vi leter etter aktuelle mikrofoner. Blant annet bør vi se etter MEMS-mikrofoner (Micro-electro-mechanical Systems), dette er veldig billige og robuste mikrofoner. Til nå har SNR ikke vært bra nok. MEMS-mikrofoner er mest brukt i mobiltelefoner og datautstyr. De teknologiske løsningene er der, men foreløpig er ikke MEMS-mikrofoner med høy SNR kommersielt tilgjengelig. Dette må vi sjekke før vi bestemmer oss.

7.2 Forforsterker

Det greieste hadde vært å implementere forforsterkeren sammen med ADC på et felles kretskort. Men på grunn av langt kabelstrekk mellom mikrofoner og ADC og med tanke på EMC (Electromagnetic compatibility) må sannsynligvis forforsterkeren lages på et eget kretskort og plasseres så nærme mikrofonene som mulig.

Vi må også velge mellom digital eller analog forforsterker. Fordelen med digital forforsterker er at det er raskt og enkelt å justere forsterkningen. Billige mikrofoner har grove toleranser og 2 forskjellige mikrofoner av samme type vil høyst sannsynlig kreve litt forskjellig forsterkning for å gi lik spenning ut. Analoge forforsterkere må justeres ved å bytte ut, og lodde motstander på kretskortet. Digitale forforsterkere justeres ved koding, bakdelen er at digitale forforsterkere er dyrere.

For at oppgaven skal bli gjennomførbar bør ikke dette prioriteres først. Forforsterkeren er ikke avgjørende for å sjekke at systemet fungerer tilfredsstillende med tanke på synkronisering og datafangst. Men det er viktig å lage blant annet overspenningsvern før ADC slik at eventuelt høye spenninger ut fra mikrofoner ikke skader ADC-ene.

7.3 ADC

Dersom vi skal ha 32 kanaler krever det 16 ADC-er. Dette vil ta stor plass og kreve mye av spenningsforsyningen. Derfor må vi undersøke om det finnes ADC-er med flere kanaler. Hvilken samplingfrekvens man skal bruke avhenger av om vi kun er interesserte i å kunne oppfatte MP, eller om vi også skal kunne lese av SP. Hvis systemet bare skal ta seg av MP, så kan man muligens senke samplingfrekvensen ned fra 96 kHz som Snipos bruker i dag. SP krever så høy samplingfrekvens fordi stigningstallet til SPs første positive flanke er veldig stor. Ved høy samplingfrekvens vil vi få en stor datamengde å behandle, vi kan da få problemer med forsinkelser på systemet og tap av data. Valg av oppløsning er også et viktig spørsmål. I og med at MP ofte har lav amplitude kan 16 bit bli i minste laget. Vi er interessert i så lavt støynivå fra systemet som mulig for å ikke drukne MP i støy. Derfor bør høyere oppløsning vurderes. Bakdelen er at total datamengde øker ved økt oppløsning.

For å beholde den SNR ADC-produsenten oppgir krever det en stabil og ren analog spenningsforsyning. Dersom den lager mye støy vil total SNR synke.

7.4 Kretskort

Når det gjelder design av kretskort har vi flere muligheter: Vi kan enten designe et kort med FPGA og ADC-er fra bunnen av, vi kan kjøpe FPGA-utviklingskort eller bruke FPGA-kortet som allerede blir brukt i Snipos-system. For det første er vanskelighetsgraden enorm på å designe et FPGA-kort fra bunnen av med de krav til hastighet vi har. Det er snakk om veldig mange lag på kortet og det vil ta veldig lang tid å få prosessert det. Kapasiteten til FPGA-en Snipos bruker nå er trolig for liten. Så det beste alternativet er nok å kjøpe et utviklingskort fra produsent. Antall I/O-pinner bør ikke bli noe problem uansett. Da må vi designe et kort med ADC-er og eventuelt et eget kort med forforsterkere. Vanskelighetsgraden og prosesseringstiden på disse er trolig mer passende til et studentprosjekt.

Vi må også velge mellom å få et eksternt firma til å prosessere kortene med komponenter og alt eller å bare få prosessert selve kortet og lodde på komponenter selv. Økonomi, tidsbruk kompleksitet på kortet blir avgjørende her. Men det enkleste er å få kortet ferdig laget med komponenter. Komponentene er ofte veldig små og vanskelige å lodde på. Vi ser allerede nå at det blir en utfordring med tanke på kretskortutlegg og prosesseringstider. Vi må komme kjapt i gang med design etter jul. Et av problemene er at vi som skal ha valgfag i dette begynner med undervisningen først etter jul. Det første vi må bestemme oss for er hvilken software vi skal bruke. På skolen lærer vi OrCAD. Dette programmet er ikke veldig godt. Dersom Posicom kan låne oss lisenser til programmet de bruker er dette et alternativ. En annen positiv ting med dette er at det blir lett for Posicom å modifisere kretskortutlegget vi har gjort ved en senere anledning.

7.5 FPGA

Ved valg av produsent står det hovedsakelig mellom Altera og Xilinx som begge har fordeler og ulemper. Det er en Altera FPGA som allerede sitter i Snipos-systemet, så det er muligheter for at oppdragsgiver foretrekker dette på grunn av hardwarekompatibilitet, lisenser på utviklingsverktøy, IP-cores eller andre ting. Hvis dette ikke er et ønske vil en FPGA fra Xilinx være det mest naturlige valget i og med at det er Xilinx FPGA og utviklingsverktøy vi har brukt på skolen og gjort oss erfaringer med. Kapasitet på FPGA kommer an på antall kanaler og oppløsning. Dersom datamengdene blir veldig store kan det hende at det trengs en ekstra buffer i form av for eksempel en SRAM (Static Random Access Memory).

Hvilken FPGA vi skal bruke har vi ikke kommet langt nok i prosessen til å kunne ta et valg på, vi må se om det finnes IP-cores til aktuelle kandidater hvis vi velger å gå den veien. Vi vil også trenge internt eller eksternt non-volatile minne. På grunn av at det er store datamengder som skal håndteres av systemet, vil vi trolig trenge en high-end FPGA, Altera Stratix eller lignende, med stor kapasitet og høy klokkefrekvens. Men disse er veldig dyre, så det må gjøres en avveining opp mot økonomi. I tillegg trenger vi en ekstern klokkepulsgenerator.

7.6 Ethernet

Når det kommer til Ethernet-implementering på FPGA-en, må vi se på om vi skal lage egen implementasjon for protokollene vi skal bruke. Det finnes IP-cores som er open source og er dermed til fri bruk. IP-cores vi må se nærmere på er UDP/IP core, og om denne kan brukes uten IP om tilkoblingen mellom FPGA og datamaskinen er punkt til punkt. Et annet alternativ er også å sende rå Ethernet-rammer, uten å tenke på UDP-protokollen.

På grunn av store datamengder vil 10BASE-T ikke være raskt nok. 100BASE-T eller 1000BASE-T må vurderes. Dette må undersøkes i forhold til antall kanaler og oppløsning på kanalene.

Ved overføringen av data over Ethernet til en PC, trenger vi et program som tar imot disse. Et alternativ er å lagre data på fil. Oppdragsgiver har utviklet et program som tar imot og lagrer dataene fra FPGA-en. Hvis dette programmet oppfyller våre krav kan vi bruke dette, eventuelt kan vi utvikle et nytt eller endre på oppdragsgiver sitt program.

8 Fremdriftsplan

Frist	Oppgave	Ansvarlig
11.10.10	Forstudie	Knut Olav Skyttemyr
05.11.10	Kravspesifikasjon	Knut Olav/Fredrik
15.11.10	Testspesifikasjon	Jardar Hvidsten/Erlend Aasland
15.11.10	Prosjektplan	Kathrine Kaasa
23.12.10	Web	Fredrik Bakke
23.12.10	Økonomi	Jardar Hvidsten
23.12.10	Risikoanalyse	Erlend Aasland
23.12.10	Belastningsdiagram	Jardar Hvidsten
10.01.11	Presentasjon 1	Kathrine Kaasa

POSICOM

HØGSKOLEN i Buskerud

Kravspesifikasjon		
Dokumenttype	Dokumentnummer	Versjonsnummer
Planlegging	P.3	1.d
Dato sist endret	Frigitt dokument?	Dokumentansvarlig
27.05.2011	Ja	Knut Olav/Fredrik
Gruppemedlemmer:		
Kathrine Kaasa, Knut Olav Skyttemyr, Jardar Hvidsten, Fredrik Bakke, Erlend Rauk Aasland		

Innholdsliste

1 Versjonhistorie....3
2 Introduksjon....4
3 Definisjoner....5
4 Sammendrag av krav....6
4.1 Prosjektkrav....6
4.2 Analoge systemkrav....6
4.3 Digitale systemkrav....6
5 Krav....7
5.1 Prosjektkrav....7
5.2 Systemkrav....8
5.2.1 Analog del....8
5.2.2 Digital del....11
6 Referanser....14

Forkortelser og akronymer

ADC	-	Analog-to-digital converter
AD	-	Analog-til-digital
Snipos	-	Sniper Positioning System
FPGA	-	Field Programmable Gate Array
UDP	-	User Datagram Protocol
HMS	-	Helse, miljø, sikkerhet
EMC	-	Electromagnetic compatibility
ESD	-	Electrostatic discharge

1 Versjonhistorie

Versjon	Endringer
0.a	Oppretting av dokument
0.b	Innføring av krav fra første kravspec møte
1.a	Første frigitte versjon av dokumentet.
1.b	Første innleverte versjon
1.c	Tatt vekk krav A.I.A.6. Endret kravprioritet på tidligere krav A.I.A.1, B.I.A.3, nå B.I.A.1 og C.I.A.3. Endret krav A.I.A.2. (I henhold til samtale med oppdragsgiver) Slettet krav A.I.A.6. Slettet krav A.F.A.10, lagt til rel.dok. Lagt til de nye test-ID-ene. Korrekturlest.
1.d	Endelig versjon.

2 Introduksjon

Denne oppgaven er gitt av Posicom AS i Kongsberg. Oppgaven er gitt i forbindelse med et system kalt Snipos (Sniper Positioning System). Dette systemet bruker mikrofoner for å fange opp skuddlyder, og på bakgrunn av det regner de ut avstand og retning til våpenet. Dagens system har fire mikrofoner. En av de viktigste utfordringene Snipos har i dag er å få økt rekkevidden (signal-støy forhold). Oppgaven går ut på å lage en hardwareprototype for åtte mikrofoner. Med åtte mikrofoner kan rekkevidden til systemet i teorien økes til det dobbelte. Prinsippet som brukes til dette er beamforming.

Dette dokumentet inneholder en oversikt over krav som er satt til systemet vi skal lage. Denne kravspesifikasjonen danner grunnlaget for videre planlegging og utføring av oppgaven [1][2].

I dette prosjektet vil vi lage en hardwareprototype for sampling og levering av lyddata. Denne skal gjøre det mulig for en mottakerdatamaskin å demonstrere beamforming-prinsippet. Vi vil derfor ikke se så nøye på ytre utfordringer som HMS (Helse, miljø, sikkerhet), ESD (Electrostatic discharge), utendørsbruk og så videre. Vi vil tenke på EMC (Electromagnetic compatibility) ved design av kretskortet, men vil ikke teste dette så nøye underveis.

3 Definisjoner

For å gjøre kravspesifikasjonen oversiktlig er det hensiktsmessig å gruppere kravene. Krav gitt av oppdragsgiver er primære krav og disse er endelige. Krav som gruppa har avledet av de primære kravene og krav gruppa setter selv kan forandres i løpet av prosjektet dersom det er nødvendig.

Vi deler kravene inn i tre grupper:

- **Funksjonelle (F):** Beskriver hva systemet skal kunne utføre.
- **Ikke-funksjonelle (I):** Beskriver egenskapene og utformingen til systemet.
- **Prosjektkrav (P):** Beskriver selve arbeidsprosessen og krav knyttet til dokumentasjon frem mot sluttproduktet.

Vi har også valgt å dele inn de funksjonelle kravene i analog del og digital del. Analog del er gruppe **A** og digital del er gruppe **D**.

Kravene vil bli tildelt en prioritet: **A**, **B** eller **C**.

- **Prioritet A** er krav som helt nødvendige å oppfylle for at systemet skal fungere.
- **Prioritet B** er krav som er viktige, men som nedprioriteres i forhold til A-krav.
- **Prioritet C** er krav som er mindre viktige. Dersom vi får tid ser vi på disse.

Ut i fra denne inndelingen vil hvert krav få en unik ID bestående av

[prioritet].[kravtype].[gruppe].[nummer], for eksempel A.F.1.1

De vil også få en test-ID som kobler kravet til testspesifikasjonen [1].

Alle detaljene rundt et krav vil bli fylt inn i et skjema som dette:

Navn:	**Prioritet:**
Kilde:	**Opprettet (dato):**
Beskrivelse:	
Kravtype:	**Endret:**
Krav-ID:	**Relatert test:**

4 Sammendrag av krav

Nedenfor følger et sammendrag av alle kravene med ID og navn. For kravdetaljer viser vi til neste kapittel.

4.1 Prosjektkrav

A.P.1 – Språk

A.P.2 – Offentlighet

A.P.3 – Plassering av hjemmeside

4.2 Analoge systemkrav

B.I.A.1 – PC104

A.F.A.2 – Antall kanaler

C.I.A.3 – Stabling

A.F.A.4 – Oppløsning

A.F.A.5 – Samplingfrekvens

A.I.A.7 – Klokke

A.F.A.8 – Mikrofonspenning

A.F.A.9 – Mikrofonforsterkning

A.F.A.11 – Spenningsforsyning AD-kort

4.3 Digitale systemkrav

A.I.D.1 – FPGA (Field Programmable Gate Array)

A.F.D.2 – Ethernetoverføring

A.I.D.3 – Grensesnitt ADC-FPGA

A.I.D.4 – Utviklingsspråk

A.F.D.5 – Lagring av data

C.F.D.6 – Visualisering av data

C.F.D.7 – Beamforming

5 Krav

5.1 Prosjektkrav

Navn: Språk	**Prioritet:** A
Kilde: Prosjektgruppa	**Opprettet (dato):** 26.10.10
Beskrivelse: Dokumentasjon skal skrives på norsk.	
Kravtype: Prosjektkrav	**Endret:**
Krav-ID: A.P.1	**Relatert test:**

Navn: Offentlighet	**Prioritet:** A
Kilde: Ragnvald Otterlei	**Opprettet (dato):** 10.11.10
Beskrivelse: Dokumenter som inneholder tekniske beskrivelser av Snipossystemet skal ikke offentliggjøres.	
Kravtype: Prosjektkrav	**Endret:** 22.05.2011
Krav-ID: A.P.2	**Relatert test:**

Navn: Plassering av hjemmeside	**Prioritet:** A
Kilde: Sigmund Gudvangen	**Opprettet (dato):** 29.10.10
Beskrivelse: Hjemmeside skal ligge på skolens server.	
Kravtype: Prosjektkrav	**Endret:**
Krav-ID: A.P.3	**Relatert test:**

5.2 Systemkrav

5.2.1 Analog del

Navn: PC104	**Prioritet:** B
Kilde: Ragnvald Otterlei	**Opprettet (dato):** 26.10.10
Beskrivelse: AD-kortet (Analog-til-digital) må ha PC104 størrelse, 3,6''x3,8''.	
Kravtype: Ikke-funksjonell	**Endret**: 21.01.11, Ragnvald Otterlei
Krav-ID: B.I.A.1	**Relatert test:**

Navn: Antall kanaler	**Prioritet:** A
Kilde: Ragnvald Otterlei	**Opprettet (dato):** 26.10.10
Beskrivelse: AD-kortet må ha åtte kanaler.	
Kravtype: Funksjonell	**Endret**: 21.01.11, Ragnvald Otterlei
Krav-ID: A.F.A.2	**Relatert test:**

Navn: Stabling	**Prioritet:** C
Kilde: Ragnvald Otterlei	**Opprettet (dato):** 26.10.10
Beskrivelse: AD-kortet designes med mulighet for stabling. Det vil si at det må implementeres en distribusjon av masterklokke til et AD-kort nummer 2.	
Kravtype: Ikke-funksjonell	**Endret:** 21.01.11, Ragnvald Otterlei
Krav-ID: C.I.A.3	**Relatert test:**

Navn: Oppløsning	**Prioritet:** A
Kilde: Ragnvald Otterlei	**Opprettet (dato):** 26.10.10
Beskrivelse: ADC-ene (Analog-to-digital converter) må ha en oppløsning på 24 bit.	
Kravtype: Funksjonell	**Endret:**
Krav-ID: A.F.A.4	**Relatert test:**

Navn: Samplingfrekvens	**Prioritet:** A
Kilde: Ragnvald Otterlei	**Opprettet (dato):** 26.10.10
Beskrivelse: ADC-ene må ha mulighet for en samplingfrekvens på 96 kHz.	
Kravtype: Funksjonell	**Endret:**
Krav-ID: A.F.A.5	**Relatert test:** T.H.3

Navn: Klokke	**Prioritet:** A
Kilde: Geir Myklatun	**Opprettet (dato):** 11.11.10
Beskrivelse: Masterklokke implementeres på AD-kortet.	
Kravtype: Ikke-funksjonell	**Endret:**
Krav-ID: A.I.A.7	**Relatert test:** T.H.3

Navn: Mikrofonspenning	**Prioritet:** A
Kilde: Gruppa	**Opprettet (dato):** 26.10.10
Beskrivelse: AD-kortet må gi spenning til transistoren i mikrofonene. Antall volt avhenger valgte komponenter.	
Kravtype: Funksjonell	**Endret:** 26.01.11, gruppa.
Krav-ID: A.F.A.8	**Relatert test:** T.H.2

Navn: Mikrofonforsterkning	**Prioritet:** A
Kilde: Ragnvald Otterlei	**Opprettet (dato):** 10.11.10
Beskrivelse: Det skal implementeres variabel forsterkning på AD-kortet.	
Kravtype: Funksjonell	**Endret:**
Krav-ID: A.F.A.9	**Relatert test:** T.H.5

Navn: Spenningsforsyning AD-kort	**Prioritet:** A
Kilde: Ragnvald Otterlei	**Opprettet (dato):** 14.11.10
Beskrivelse: AD-kortet må forsynes med spenning ut i fra valgte komponenter.	
Kravtype: Funksjonell	**Endret:** 21.01.11, Ragnvald Otterlei
Krav-ID: A.F.A.11	**Relatert test:** T.H.2

5.2.2 Digital del

Navn: FPGA	**Prioritet:** A
Kilde: Ragnvald Otterlei	**Opprettet (dato):** 26.10.10
Beskrivelse: Det skal brukes et Altera Cyclone utviklingssett.	
Kravtype: Ikke-funksjonell	**Endret:**
Krav-ID: A.I.D.1	**Relatert test:**

Navn: Ethernetoverføring	**Prioritet:** A
Kilde: Gruppa	**Opprettet (dato):** 26.10.10
Beskrivelse: Overføring fra FPGA til mottaker-PC skal skje ved hjelp av UDP (User Datagram Protocol) over Ethernet.	
Kravtype: Funksjonell	**Endret:**
Krav-ID: A.F.D.2	**Relatert test:** T.P.1

Navn: Grensesnitt ADC - FPGA	**Prioritet:** A
Kilde: Gruppa	**Opprettet (dato):** 26.10.10
Beskrivelse: I^2S brukes som grensesnitt mellom ADC og FPGA.	
Kravtype: Ikke-funksjonell	**Endret:**
Krav-ID: A.I.D.3	**Relatert test:** T.P.3

Navn: Utviklings-språk	**Prioritet:** A
Kilde: Gruppa	**Opprettet (dato):** 29.10.10
Beskrivelse: Det skal brukes VHDL for å programmere FPGA-en og C i Nios II-prosessoren.	
Kravtype: Ikke-funksjonell	**Endret:**
Krav-ID: A.I.D.4	**Relatert test:**

Navn: Lagring av data	**Prioritet:** A
Kilde: Gruppa	**Opprettet (dato):** 29.10.10
Beskrivelse: Data som samles inn skal lagres binært til fil på en mottaker-PC.	
Kravtype: Funksjonell	**Endret:**
Krav-ID: A.F.D.5	**Relatert test:** T.P.2

Navn: Visualisering av data	**Prioritet:** C
Kilde: Gruppa	**Opprettet (dato):** 29.10.10
Beskrivelse: Data som samles inn skal kunne visualiseres grafisk på mottaker-PC.	
Kravtype: Ikke-funksjonell	**Endret:**
Krav-ID: C.I.D.6	**Relatert test:**

Navn: Beamforming	**Prioritet:** C
Kilde: Gruppa	**Opprettet (dato):** 26.10.10
Beskrivelse: En enkel beamformingsalgoritme skal implementeres på mottaker-PC for å demonstrere beamforming.	
Kravtype: Ikke-funksjonell	**Endret:**
Krav-ID: C.I.D.7	**Relatert test:**

6 Referanser

[1] Prosjektgruppe 2011-9, *P.4 Testspesifikasjon.* MMS, 2011.

[2] Prosjektgruppe 2011-9, *P.5 Prosjektplan.* MMS, 2011

POSICOM

HØGSKOLEN i Buskerud

Testspesifikasjon		
Dokumenttype	Dokumentnummer	Versjonsnummer
Planlegging	P.4	1.f
Dato sist endret	Frigitt dokument?	Dokumentansvarlig
25.05.2011	Ja	Jardar, Erlend
Gruppemedlemmer:		
Kathrine Kaasa, Knut Olav Skyttemyr, Jardar Hvidsten, Fredrik Bakke, Erlend Rauk Aasland		

Innholdsliste

1 Versjonhistorie....3
2 Introduksjon....4
3 Definisjoner....5
4 Teststrategier....6
4.1 Inkrementell testing....6
4.2 Stresstesting....6
5 Testmetoder....7
5.1 Black-box....7
5.2 White-box....7
5.3 Grensesnitt-test....7
5.4 Systemtest....7
6 Tester....8
6.1 Test av digitale systemkrav....8
6.2 Hardwaretester....9
6.3 Ytelse 11
6.4 Systemtest....12
7 Referanser....13

Forkortelser og akronymer

ADC	-	Analog-to-digital converter
I^2S	-	Inter-IC Sound (Integrated Interchip Sound)
FPGA	-	Field Programmable Gate Array
UDP	-	User Datagram Protocol
AD	-	Analog-til-digital
FIFO	-	First In, First Out
DMA	-	Direct Memory Access
SDRAM	-	Synchronous dynamic random access memory
SGDMA	-	Scatter-Gather Direct Memory Access
RTL	-	Register Transfer Level

1 Versjonhistorie

Versjon	Endringer
0.a	Lagt til teststrategier
0.b	Lagt til test av digitale krav
0.c	Lagt til referanser og systemtest
1.a	Første frigitte versjon
1.b	Første innleverte versjon
1.c	Tatt vekk test T.A.6
1.d	Lagt til mikrofontest og forforsterkertest.
1.e	Endret, flyttet og slått sammen flere tester. Endret analoge tester til hardwaretester og digitale tester til FPGA-tester. Tatt vekk kapittel om generelle tester, lagt til kapittel om ytelse.
1.f	Endelig versjon.

2 Introduksjon

Dette dokumentet inneholder tester som vi vil kjøre i prosjektet hvor vi utvikler et multimikrofonsystem for Posicom AS. Testene er utformet for å dekke alle systemkrav. I tillegg har vi utviklingstester som vi kjører underveis i prosjektet. Dette er viktig for verifisering av enkeltmoduler.

Når vi utfører tester vil vi skrive en testrapport. Dette vil være en detaljert beskrivelse av testprosedyren, hva som blir testet, forventet resultat og faktisk resultat. Dersom feil blir avdekket vil det bli foreslått en mulig løsning på problemet.

Testrapportene vil bli samlet for hver av de forskjellige gruppene beskrevet i kapittel 3 Definisjoner.

3 Definisjoner

For å gjøre testspesifikasjonen oversiktlig er det hensiktsmessig å gruppere testene. Vi har valgt å dele inn testene i grupper:

- **Programvare/VHDL-tester (P)**
- **Hardwaretester (H)**
- **Ytelsestester (Y)**
- **Systemtest (S)**

Ut i fra denne inndelingen vil hver test få en unik ID bestående av følgende:

[T for test].[gruppe].[nummer], for eksempel T.F.1

Alle detaljene rundt en test vil bli fylt ut i en tabell som dette:

Test-ID:	**Relaterte krav:**
Beskrivelse:	
Testbeskrivelse:	

Der testen kan relateres til et eller flere krav i [2] vil dette bli spesifisert.

4 Teststrategier

En teststrategi beskriver på et overordnet nivå hvordan testingen skal utføres. I vårt prosjekt har vi valgt å benytte oss av inkrementell testing og stresstesting, begge nevnt i [1]. Vi bruker to ulike strategier da dette passer best med systemet vi skal teste.

4.1 Inkrementell testing

Her starter vi med å teste de mest kritiske komponentene hver for seg. Hvis dette virker tilfredsstillende kan vi sette komponentene sammen til en modul. Slik fortsetter vi til modulene er satt sammen til et komplett system. Vi er da sikre på at hver enkelt modul fungerer som det skal. Dette reduserer antall mulige feilkilder.

For AD-kortet (Analog-til-digital) vil det være vanskelig å teste de forskjellige modulene før vi får AD-kortet ferdig produsert. Under konstruering av kretsen vil vi derfor simulere de forskjellige modulene for å verifisere at de fungerer som planlagt. Disse simuleringene vil ikke være knyttet til noen tester, men vil bli rapportert i en egen simuleringsrapport.

4.2 Stresstesting

En stresstest foregår ved at vi utsetter systemet for maksimal belastning eller en belastning som er høyere enn det systemet er bygd for. Vi observerer hvordan systemet takler belastningen. Målet med testene er å se om systemet feiler, og framprovosere feil som vi ikke har oppdaget tidligere. Feil må rapporteres og det gjøres en vurdering på om feilen er av kritisk karakter og må rettes.

5 Testmetoder

Innenfor hver teststrategi bruker vi ulike typer testmetoder. Hvilken testmetode vi bruker kommer for eksempel an på om det er utvikleren av enheten som tester sitt eget system, eller om det er en person som ikke kjenner systemet. Under har vi beskrevet metodene vi vil benytte oss av [1].

5.1 Black-box

Under en black-box-test ser vi på systemet som en boks med et visst antall innganger og utganger. Han eller hun som utfører testen behøver ingen forkunnskaper om innsiden av systemet som testes. Testeren tar kun for seg inndata, og registrer om utdata samsvarer med forventet eller «riktig» verdi.

Systemtesten vil bestå av en black-box-test.

5.2 White-box

Under en slik testmetode er det nødvendig at testeren både kjenner til og har kunnskaper om systemet. Testeren kan da sjekke om inndata blir behandlet på forventet måte, og om utdata er av forventet verdi.

Hardwaretestene og programvare/VHDL-testene vil i stor grad være white-box-tester.

5.3 Grensesnitt-test

Grensesnitt-test brukes når man slår sammen enkeltmoduler av et system til større moduler eller et komplett system. Målet med en slik test er å finne feil som oppstår når man slår enkeltmoduler sammen. Dette kan være feil i utviklingen eller feiloppfatninger som utvikleren har gjort seg når modulen var under konstruksjon. Eller rett og slett koblingsfeil.

5.4 Systemtest

Systemtesten er en omfattende test som skal kjøres etter at alle A-krav er oppfylt. Systemet vil her bli testet som en helhet. Alle grensesnitt og all funksjonalitet skal dekkes av testbeskrivelsen. Systemtesten skal først kjøres internt, deretter med oppdragsgiver.

6 Tester

6.1 Test av digitale systemkrav

Test-ID: T.P.1	**Relaterte krav:** A.F.D.2
Beskrivelse: Overføring fra FPGA (Field Programmable Gate Array) til mottaker skal skje ved hjelp av UDP (User Datagram Protocol) over Ethernet.	
Testbeskrivelse: Send data fra FPGA og verifiser at overføring skjer ved hjelp av UDP. Bruk et pakkesniffeprogram til dette.	

Test-ID: T.P.2	**Relaterte krav:** A.F.D.5
Beskrivelse: Lagring til fil.	
Testbeskrivelse: Innsamlet data på mottaker-PC skal være korrekt og brukbar til videre behandling. Kontrollert eventuelle pakketap og at data ikke er korrupt.	

Test-ID: T.P.3	**Relaterte krav:** A.I.D.3
Beskrivelse: Mottak: Lese fra I^2S-buss (Integrated Interchip Sound).	
Testbeskrivelse: Testen utføres som RTL-simulering (Register Transfer Level--simulering) i Modelsim. Data kommer inn over I^2S-bussen som spesifisert i databladet til aktuell ADC (Analog-to-digital converter). Se at riktig data leses av bussen og at overflødige bits ignoreres.	

Test-ID: T.P.4	**Relaterte krav:**
Beskrivelse: FIFO (First In, First Out) og DMA (Direct Memory Access)	
Testbeskrivelse: Skriv til FIFO i samme hastighet som beskrevet i [2]. Kontroller at DMA henter ut forventede data uten at FIFO flyter over. FIFO-portene skal være fungerende Avalon-MM-grensesnitt med 32 bits ordbredde. Og DMA skal være aktiv på maks 50 % av klokkesyklusene.	

Test-ID: T.P.5	**Relaterte krav:** A.F.D.2
Beskrivelse: Sammenslåing av moduler (unntatt mottak/I^2S)	
Testbeskrivelse: Etter sammenslåing av FIFO-, SGDMA- og SDRAM-, DMA- og Ethernetmoduler, vil vi forsikre oss om at data blir overført korrekt mellom disse og data kan avleses på mottaker-PC. Testdata som sendes må være av et format som kan avdekke pakketap, feil rekkefølge og korrupsjon. Testen utføres ved hjelp av en trafikkgenerator istedenfor I^2S-grensesnitt og mottaksmodul.	

Test-ID: T.P.6	**Relaterte krav:**
Beskrivelse: Dataflyt fra mottaksmodul til FIFO.	
Testbeskrivelse: Testen uføres med RTL simuleringsverktøy. Mat mottaksmodulen med data, og se at den genererer de data og kontrollsignaler som trengs for å kommunisere med FIFO igjennom Avalon-bussen.	

Test-ID: T.P.7	**Relaterte krav:** A.F.D.2
Beskrivelse: Pakking og videresending	
Testbeskrivelse: Kontroller at data fra SDRAM (Synchronous dynamic random access memory) blir hentet og videresendt over Ethernet. Dette må skje i en høyere hastighet enn overføringshastigheten av data fra AD-kortet. Data skal overføres ved hjelp av UDP-/IP-protokollene.	

6.2 Hardwaretester

Test-ID: T.H.1	**Relaterte krav:**
Beskrivelse: Kalibreringstest	
Testbeskrivelse: Forsyn mikrofonene med 5 VDC, bruk en kalibrator for å sette et kjent lydtrykk på mikrofonen. Mål signalsvingen på mikrofonutgangene. Hvis mikrofonene har differensiell konfigurering, skal signalet på pluss-/minuslinjene ha lik amplitude, men være i 180° faseforskjøvet. Hvis mikrofonene har single ended konfigurering skal vi bare ha et signal på plusslinjen, minuslinjen skal være koblet til jord. Verifiser at positiv og negativ linje ut fra mikrofonene stemmer med konnektorene.	

Test-ID: T.H.2	**Relaterte krav:** A.F.A.8, A.F.A.11
Beskrivelse: Spenningsforsyning	
Testbeskrivelse: Mål spenninger inn på AD-kortet med et voltmeter. Verifiser at spenningen er 5 VDC på analoge komponenter og 3,3 VDC på digitale komponenter. Verifiser at spenningen etter filtrene er mellom 4,75 - 5,25 VDC og 3,0 - 3,6 VDC.	

Test-ID: T.H.3	**Relaterte krav:** A.F.A.5, A.I.A.7
Beskrivelse: Klokkelinjer	
Testbeskrivelse: Verifiser at frekvensen på masterklokken = 24,576 MHz, BCK = 6,144 MHz, LRCK = 96 kHz ved hjelp av oscilloskop.	

Test-ID: T.H.4	**Relaterte krav:** A.F.A.2
Beskrivelse: Signalflyt	
Testbeskrivelse: Bruk en signalgenerator og send et signal på 1 kHz på en mikrofoninngang av gangen. **Husk at det er 5 VDC på pluss-linjen!** Bruk et oscilloskop til å måle på den tilhørende forforsterkerutgangen. Verifiser at signalsvinget er flyttet opp til 2,5 VDC. Følg signalet helt frem til 40-pins konnektoren. Gjør dette for alle de åtte kanalene. Verifiser at alle kanalene har signalflyt helt fra mikrofontilkoblingene til 40-pins konnektoren.	

Test-ID: T.H.5	**Relaterte krav:**
Beskrivelse: Forsterkningstest	
Testbeskrivelse: Koble til alle mikrofonene. Bruk en kalibrator på en mikrofon av gangen. Mål signalet på utgangen til forforsterkeren. Verifiser av signalet har blitt forsterket som forventet. Merk at signalsvinget skal være flyttet fra 0 V til 2,5 V. Med en inngangsamplitude på 141 mV_P skal utgangsamplituden være 2,5 V + 141 $mV_P \cdot 1{,}75\ V_{Put}/V_{Pinn} \approx 2{,}75V_P$. Verifiser også at frekvensresponsen er flat i arbeidsområdet, 20 Hz – 20 kHz.	

Test-ID: T.H.6	**Relaterte krav:**
Beskrivelse: Synkronisering	
Testbeskrivelse: Koble til en signalgenerator i parallell til alle mikrofoninngangene. Spill av et kjent signal. Sammenlign dataene fra de 8 kanalene og verifiser at signalene er synkroniserte langs tidsaksen.	

6.3 Ytelse

Test-ID: T.Y.1	**Relaterte krav:**
Beskrivelse: Filtertest	
Testbeskrivelse: Mål signalstøyen på 5 VDC- og 3,3 VDC-linjene før filteret mens FPGA-kortet belastes. Mål signalstøyen på de samme linjene etter filteret. Sammenlign resultatene og verifiser at filteret har en dempende effekt på signalstøyen.	

Test-ID: T.Y.2	**Relaterte krav:**
Beskrivelse: Langtidstest.	
Testbeskrivelse: Systemet skal kjøres på maksimal belastning over 12 timer uten å krasje. Sett på logging av antall pakketap.	

Test-ID: T.Y.3	**Relaterte krav:**
Beskrivelse: Støymåling**.**	
Testbeskrivelse: Sammenligning av støy med de to forskjellige spenningsforsyningsløsningene. Gjør opptak uten mikrofoner koblet til, altså måling av støy internt på kortet. Sammenlign maksutslaget på støyen mellom de to forsyningsløsningene.	

6.4 Systemtest

Test-ID: T.S.1
Forventning: Data skal sendes på alle kanaler, og lagres i en fil. Filen skal kunne leses ut på ekstern PC. Dataene skal samsvare med lydsignalet som sendes ut.
Testbeskrivelse: Sett opp L-MS-01, en halvkulehøyttaler, og mål opp en halvsirkel som mikrofonene plasseres på. Vi sikrer da at lyden treffer alle mikrofonene samtidig. Koble mikrofonene til AD-kortet. Send et signal med en kjent frekvens til høyttaleren. Les av filen med de lagrede lyddataene. Verifiser at dataene er like som det avspilte lydsignalet. Det vil si at kanalene skal være synkroniserte og at forsterkningen er lik på alle kanalene.

7 Referanser

[1] Strøm, T., Graven, O.H., *Prosjekthåndbok.*, Høgskolen i Buskerud, Avdeling for teknologi, 2009.

[2] Prosjektgruppe 2011-9 , *P.3 Kravspesifikasjon.* MMS, 2011.

POSICOM

HØGSKOLEN i Buskerud

Prosjektplan		
Dokumenttype	Dokumentnummer	Versjonsnummer
Planlegging	P.5	1.d
Dato sist endret	Frigitt dokument?	Dokumentansvarlig
27.05.2011	Ja	Kathrine Kaasa
Gruppemedlemmer:		
Kathrine Kaasa, Knut Olav Skyttemyr, Jardar Hvidsten, Fredrik Bakke, Erlend Rauk Aasland		

Innholdsliste

1 Versjonhistorie....3
2 Introduksjon....4
2.1 Forkortelser og akronymer....4
3 Målsetting....5
4 Prosjektmodell....6
5 Avgrensninger og forutsetninger....7
6 Aktivitetsbeskrivelser....8
7 Aktivitetskart....11
8 Hovedplan....15
8.1 Tidsplan....15
8.2 Budsjett....17
9 Om oss....18
10 Dokumentoversikt....19
11 Referanser....20

Figurliste

Flytdiagram for inkrementell utvikling....6
Aktivitetskart for administrasjon....11
Aktivitetskart for dokumentasjon....12
Aktivitetskart for hardware....13
Aktivitetskart for FPGA/programvare....14
Aktivitetskart for test og montering....14
Gantt-diagram....16

1 Versjonhistorie

Versjon	Endringer
0.a	Opprettet dokument, skrevet inn forslag på kapittelinndeling
0.b	Lagt til kapittel om prosjektmodell, endret kapitteloppsett.
1.a	Første frigitte versjon.
1.b	Første innleverte versjon.
1.c	Endret tidsplan og tidsbruk på flere aktiviteter. Se [3] for alle endringsdetaljer.
1.d	Endelig versjon.

2 Introduksjon

Denne oppgaven er gitt av Posicom AS i Kongsberg. Oppgaven er gitt i forbindelse med et system kalt Snipos. Dette systemet bruker mikrofoner for å fange opp skuddlyder, og på bakgrunn av det regner de ut avstand og retning til våpenet. Dagens system har fire mikrofoner. En av de viktigste utfordringene Snipos har i dag er å få økt rekkevidden (signal-støy forhold). Oppgaven går ut på å lage en hardwareprototype for åtte mikrofoner. Med åtte mikrofoner kan rekkevidden til systemet i teorien økes til det dobbelte. Prinsippet som brukes til dette er beamforming.

I dette dokumentet vil vi definere aktivitetene til dette prosjektet ut i fra krav- og testspesifikasjon [1] [2]. Aktivitetene vil så bli satt inn i prosjektplanen der vi estimerer tidsbruk og definerer ansvarsfordeling.

2.1 Forkortelser og akronymer

Snipos	-	Sniper Positioning System
HMS	-	Helse, miljø, sikkerhet
EMC	-	Electromagnetic compability
ESD	-	Electrostatic discharge
ADC	-	Analog-to-digital converter
AD-kort	-	Analog-til-digital-kretskort
FPGA	-	Field Programmable Gate Array
I^2S	-	Inter-IC Sound (Integrated Interchip Sound)
IP-core	-	Intellectual Property core
FIFO	-	First In, First Out
DMA	-	Direct Memory Access
UDP	-	User Datagram Protocol

3 Målsetting

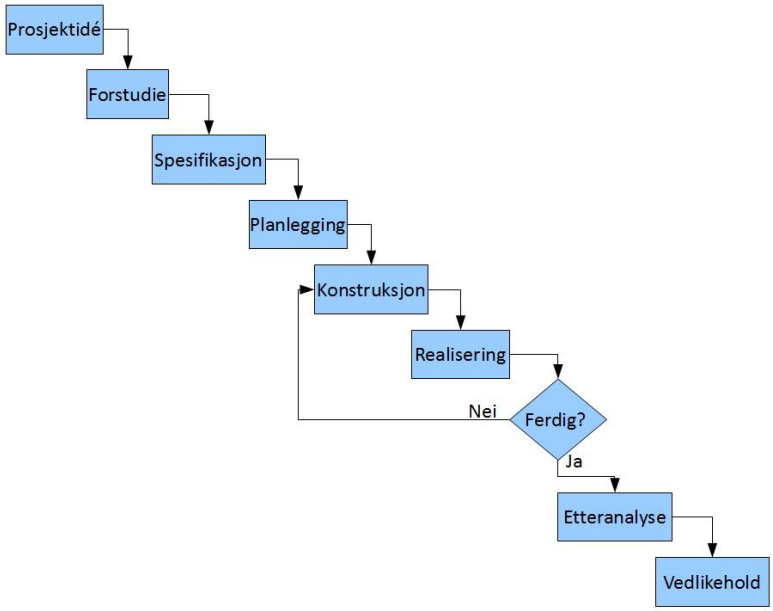
Målsetningene kan deles inn i to hovedgrupper: Tekniske mål og mål for hva vi skal lære av å jobbe med et slikt prosjekt.

De tekniske målene vil være å lage et hardwaresystem som fungerer etter kravene som er satt i kravspesifikasjonen. Systemet skal gå i sanntid og vi skal kunne lagre x-antall sekunder med lydopptak til en fil. Det skal også være mulig å utvide systemet vi lager til å kunne ta i bruk flere kanaler om det skulle være ønskelig ved en senere anledning.

Mål for hva vi skal lære av prosjektet er hvordan man skal jobbe mest effektivt sammen som en gruppe, hvordan man skal organisere et slikt prosjekt og få en viss følelse av hvordan det vil være når man kommer ut i arbeidslivet.

4 Prosjektmodell

For dette prosjektet har vi valgt å følge en prosjektmodell som heter inkrementell utvikling. Dette er en undergruppe av evolusjonære utviklingsmodeller. Den inkrementelle utviklingen tar utgangspunkt i at systemet (både hardware og FPGA/programvare) skal bygges i separate komponenter hvor man bygger og tester de viktigste komponentene først. Et eksempel på en slik viktig komponent i vårt system er AD-kortet. Når de viktigste komponentene er bygget og testet kan man se på de mindre viktige komponentene og slik fortsetter man til man er ferdig, eller har gått tom for tid. Som vi ser fra Figur 1 er de mindre viktige delene planlagt, men det tas med i tidsplanleggingen at man potensielt ikke har tid til å utføre de.



Figur 1: Flytdiagram for inkrementell utvikling.

Denne prosjektmodellen passer dette prosjektet godt da vi har god oversikt over hvordan det ferdige produktet vil bli allerede i planleggingsfasen. Med denne modellen kan vi også være sikre på å få ferdig de viktigste delene slik at vi ved slutten av prosjekttiden har et fungerende system.

5 Avgrensninger og forutsetninger

I dette prosjektet vil vi konsentrere oss om å lage en prototype for å demonstrere beamforming-prinsippet. Derfor vil vi ikke se så nøye på ytre utfordringer som HMS, ESD, utendørsbruk og så videre. Vi vil tenke på EMC ved design av AD-kortet, men vil ikke teste så nøye hvor robust AD-kortet er underveis. Vi vil heller ikke se på det mekaniske designet av selve mikrofonarrayet. Når det gjelder mikrofoner er det ikke så viktig hvilke mikrofoner vi tester med, så lenge tilkoblingsmulighetene er relativt universelle slik at man kan bytte mellom forskjellige mikrofontyper.

Følgende forutsetninger må oppfylles for at dette prosjektet skal bli vellykket:

- Veiledere fra HiBu og Posicom kan hjelpe oss med tekniske utfordringer.
- Posicom betaler for komponenter og arbeid som trengs i forbindelse med bygging av prototypen.
- Komponenter og utstyr vi bestiller ikke kommer for sent og ikke er ødelagt når vi får dem.

6 Aktivitetsbeskrivelser

I Tabell 1 er lista over aktivitetene dette prosjektet er delt inn i. Disse aktivitetene er kun de som jobbes med etter jul. I tillegg har vi valgt å ta med estimert tidsbruk på hver aktivitet samlet på hele gruppa for å få en generell oversikt. Dette prosjektet gir 20 studiepoeng og vi tar utgangspunkt i at hver person skal jobbe minimum 25 timer per studiepoeng. Dette betyr at hver person skal jobbe 500 timer. Tilsammen for prosjektgruppa blir dette 2500 timer. 250 av disse timene er beregnet på det som skal gjøres før jul. Det gir minimum 2250 timer til arbeid med prosjektet etter jul.

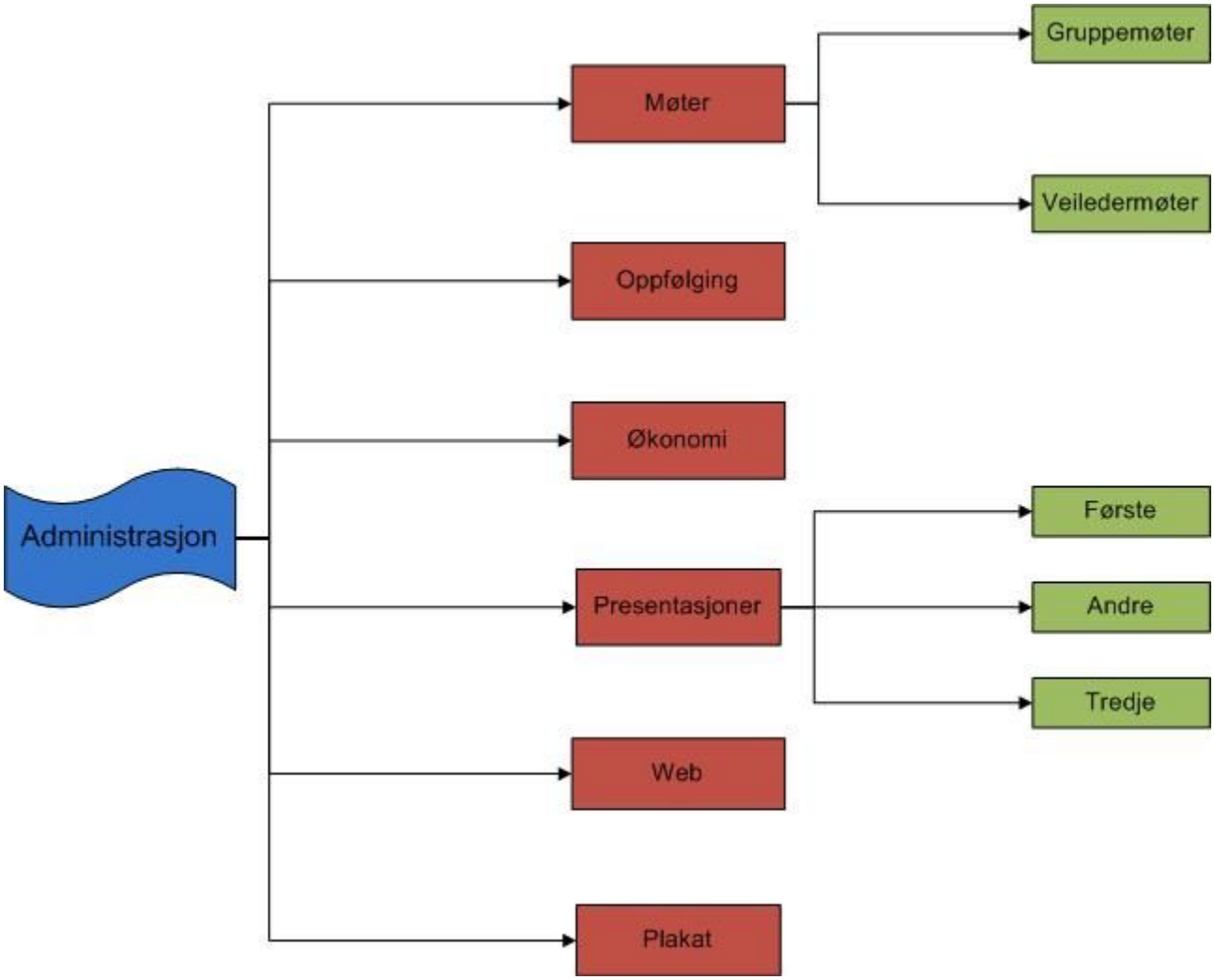
Tabell 1: Aktivitetsbeskrivelser med tidsberegning

Nr.	Aktivitet	Beskrivelse	KravID	Tidsbruk
	Administrasjon			
1	Gruppemøter	Møter holdt internt i gruppa og eventuelle møtereferater skrevet i etterkant.		60
2	Veiledermøter	Møter holdt med ekstern og intern veileder og møtereferater skrevet etterkant.		110
3	Presentasjoner	3 presentasjoner i løpet av prosjektet skal forberedes og holdes for veiledere, sensorer og andre som vil høre på.		300
4	Økonomi	Budsjett og regnskap skal holdes både for tidsbruk og pengebruk.		10
5	Web	Webside skal opprettes og oppdateres i løpet av prosjektet med hvordan det ligger an.	A.I.2 A.I.3	10
6	Plakat	Lage en enkel plakat hvor prosjektet skal vises frem.		30
7	Oppfølging	Følge opp tidsbruken i prosjektet og skrive rapport hver uke om hva som er gjort den gjeldende uka og hva som skal gjøres neste uke.	A.I.1	30
	Dokumentasjon			
8	Planleggings-dokumenter	Vedlikeholde dokumenter som kravspesifikasjon, prosjektplan og testspesifikasjon.	A.I.1	140
9	Systembeskrivelses-dokumenter	Skrive et dokument der vi forklarer hvorfor vi velger de forskjellige løsningene både for hardware (AD-kortet) og FPGA/programvare	A.I.1	150
10	Teknisk dokument	Dokument over hvordan systemet fungerer og skal brukes.	A.I.1	70
11	Etteranalyse	Skrive en sluttrapport for prosjektet og om	A.I.1	60

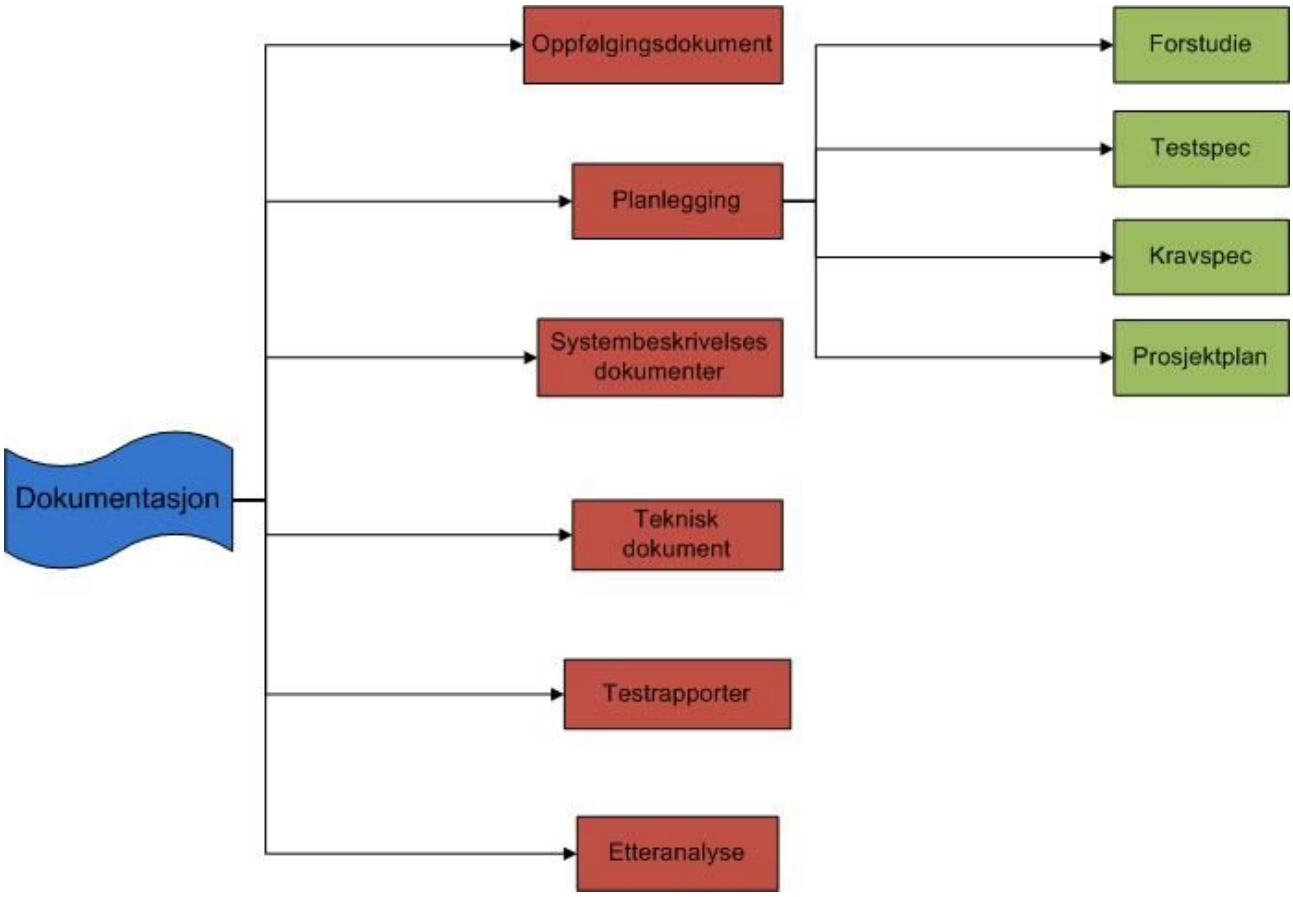
		hvordan systemet fungerte.		
12	Testrapporter	Kvittering for testene skal skrives i en rapport.	A.I.1	55
	Hardware			
13	Undersøkelse av komponenter	Finne kandidater til komponenter.	B.I.A.1 A.I.A.2	52
14	Valg av komponenter	Velge komponenter og bestille de.		18
15	Lære seg kretskort	Lære seg hvordan man best designer et kretskort og hvordan ta hensyn til EMC.		60
16	Lære seg OrCad	Lære seg det vi trenger av OrCAD utover det vi lærer i valgfaget.	B.I.A.1 A.I.A.2 C.I.A.3	60
17	Kretskortskjema	Tegne et krets-skjema for AD-kortet.		40
18	Kretskortutlegg	Tegne kretskortutlegget i OrCAD.		110
19	ADC-modul	Finne ut hvordan skal ADC-ene integreres i kretsen.	A.F.A.4 A.F.A.5	60
20	Forforsterker-modul	Mikrofonsignalene må kunne skaleres slik at ADC-ene klipper ved valgt lydtrykk.	A.F.A.9	60
21	Grensesnitt og spenningsløsning	Sette opp grensesnittet mellom AD-kortet og FPGA-en. Kontakter og I^2S-buss. Sørge for at AD-kortet får analog og digital spenning fra FPGA-en.		30
22	Overspenningsvern	Sørge for at AD-kortet er beskyttet mot overspenning og blokkerer likestrøm.	A.F.A.10	30
23	Mikrofongrensesnitt	Sette opp grensesnittet mellom mikrofonene og AD-kortet.	A.F.A.11 A.F.A.8	30
24	Klokke	Implementere klokke (oscillator) og distribusjon til alle ADC-ene og til FPGA-en over I^2S-bussen.	A.I.A.7	5
	FPGA/Programvare			
25	Opplæring	Få kunnskap om programvare, verktøy, IP-cores og komponenter vi kommer til å bruke.		35
26	Planlegging av arkitektur	Planlegge hvordan nødvendige moduler i FPGA skal implementeres,samt undersøkende arbeide. Planlegge hvordan vi skal bygge systemet. Nødvendige moduler og IP-cores / referansedesign		30
27	Referansedesign	Sette opp fungerende webserver-referansedesign som skal modifisere til vår	A.I.D.3	30

		bruk.		
28	Mottak fra AD-kort	Lage modul for mottak av lyddata fra AD-kort (I^2S), og videresending til FIFO.		50
29	DMA og FIFO	Overføring av data fra mottaks-modul over til systemets klokkedomene slik at det kan behandles videre.	A.I.D.4	60
30	Pakking og videresending av data	Overføring av datastrøm som UDP-pakker over Ethernet.	A.F.D.2	80
31	Sammenslåing av moduler	Sette alt sammen til et system som oppfyller kravene.		105
32	Lagring av lyddata	Lage PC-program som kan motta lyddata fra systemet og lagre disse til fil.	A.F.D.5	80
33	Visualisering	Visualisere lyddata ved hjelp av Labview eller et egenlaget PC-program.	C.F.D.6 C.F.D.7	55
	Montering og testing			
34	Utviklingstester hardware	Tester og simulering av enkeltdeler av hardware underveis i utviklingsprosessen.		55
35	Utviklingstester FPGA/programvare	Tester av programvare/VHDL underveis i utviklingsprosessen		120
37	Montering og testing	Det ferdig-monterte AD-kortet skal testes og eventuelle endringer gjøres og deretter settes sammen med resten av systemet.		130
38	Systemtest	Det ferdige systemet testes internt i gruppa, deretter med oppdragsgiver.		65
	Total sum timer			**2475**

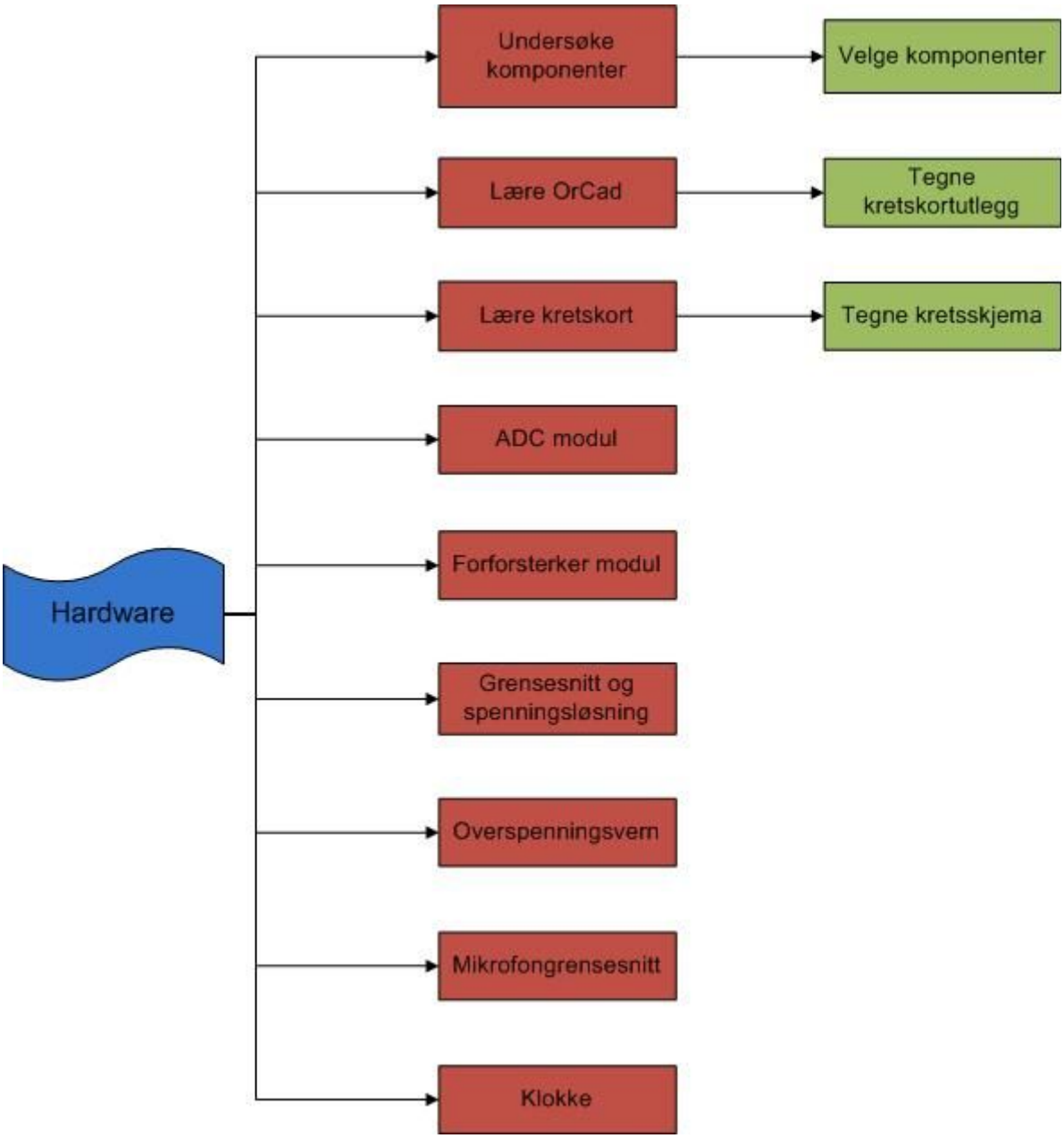
7 Aktivitetskart



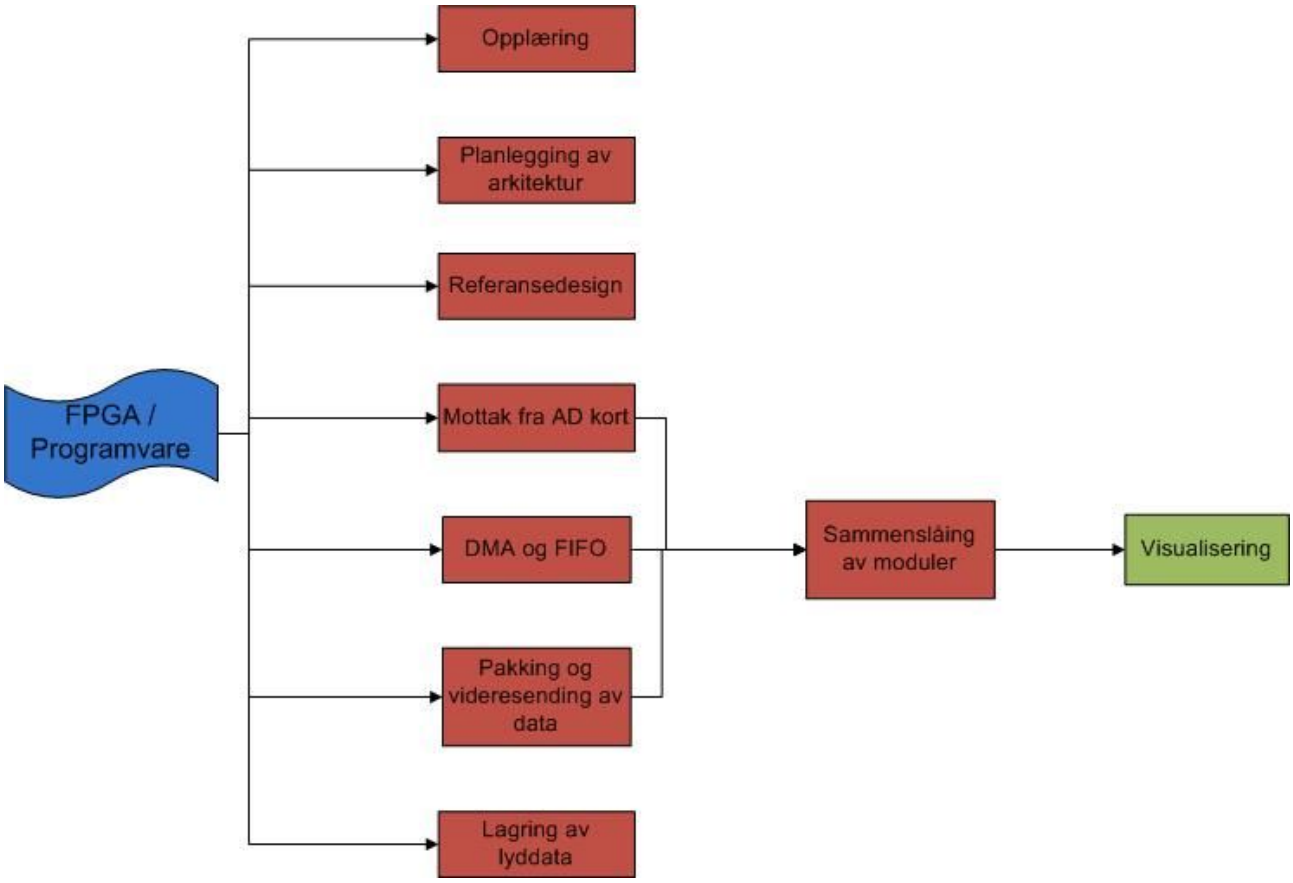
Figur 2: Aktivitetskart for administrasjon.



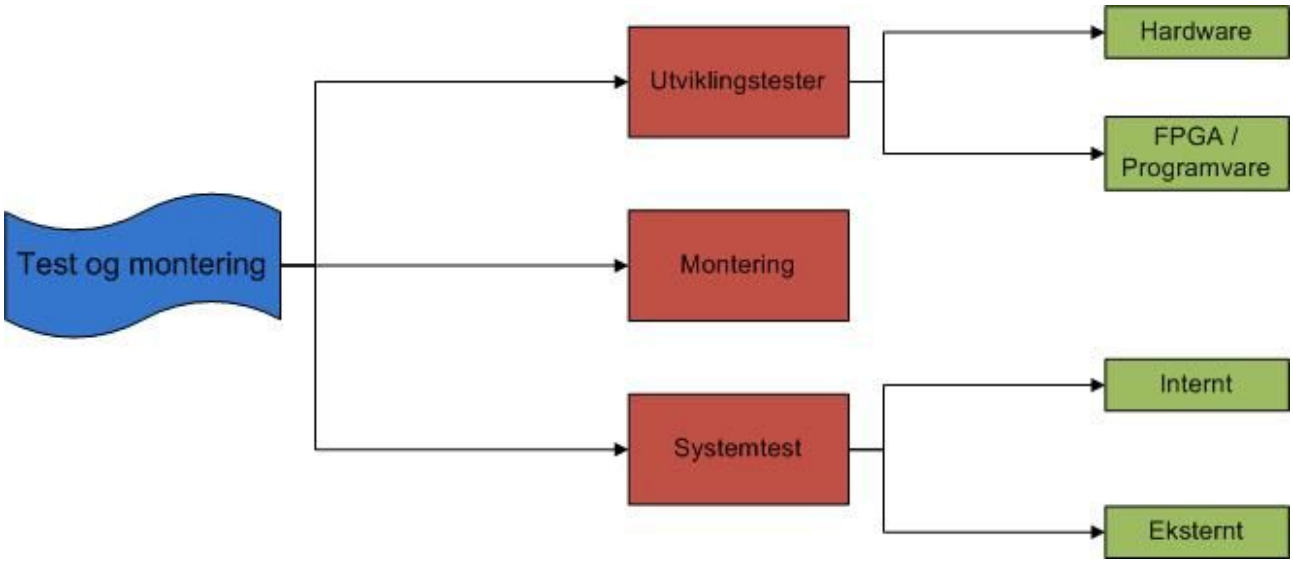
Figur 3: Aktivitetskart for dokumentasjon.



Figur 4: Aktivitetskart for hardware.



Figur 5: Aktivitetskart for FPGA/programvare



Figur 6: Aktivitetskart for test og montering.

8 Hovedplan

Vi har valgt følgende milepæler for vårt prosjekt:

Milepæl 1: Første presentasjon, 11.01.2011.

Milepæl 2: Andre presentasjon, 22.03.2011.

Milepæl 3: Endelig innlevering, 30.05.2011.

Milepæl 4: Tredje presentasjon, 09.06.2011.

8.1 Tidsplan

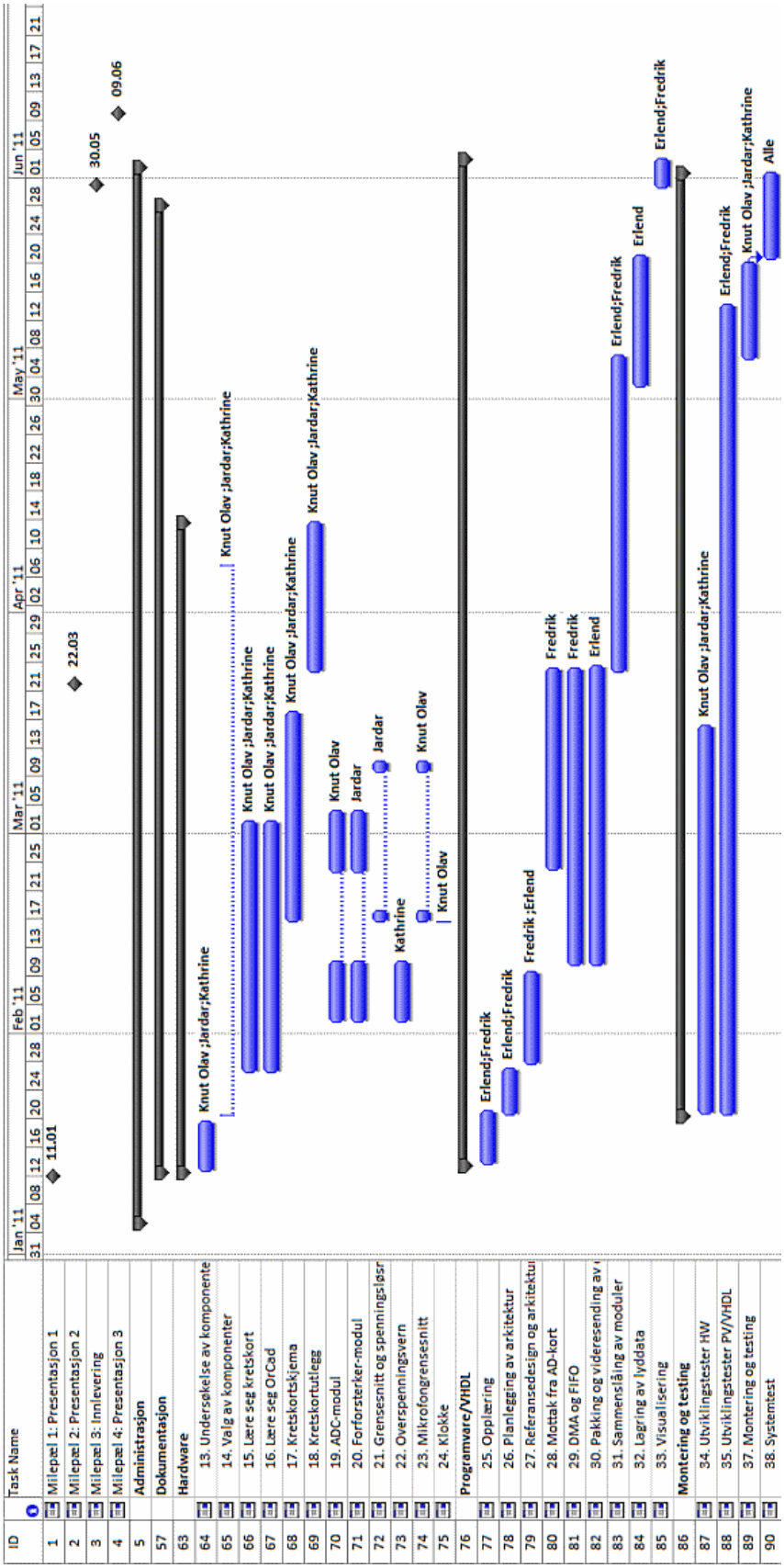
Figur 7 viser gantt-diagrammet for hele prosjektperioden etter jul. Dette er en generell og foreløpig tidsplan for når de forskjellige aktivitetene skal jobbes med og være ferdige. Før påske er det tatt utgangspunkt i at vi bruker to arbeidsdager à 7,5 timer pr uke, mens etter påske bruker vi fem arbeidsdager à 7,5 timer pr uke. Dette samsvarer ikke helt med å jobbe 2250 timer etter jul, så i enkelte perioder, sannsynligvis rundt presentasjoner og store frister, vil det måtte jobbes ekstra.

Alle administrasjonsaktivitetene bortsett fra nr. 6 (plakat) er aktiviteter som vil gå over hele prosjektperioden og er derfor ikke spesifisert nærmere. Aktivitet nr. 6 vil gjøres mot slutten av prosjektet. Dokumentasjonsaktivitetene vil også gå over større perioder og vil derfor ikke være mer spesifisert enn dette.

Det vil bli holdt møter med veileder hver uke om prosjektets framgang og gruppa vil også ha en dag i uka hvor det blir holdt et slikt kort møte internt. I tillegg vil vi ha noen større møter hvor vi går gjennom prosjektets framgang internt for å få en god oversikt over hvor vi er og om det er noen tiltak som må gjøres i forhold til planen. Disse gjennomgangene vil tas i begynnelsen av hver måned. På disse tidspunktene vil det være naturlig å oppdatere tidsplanen.

I forbindelse med veiledermøtene vil prosjektleder gå i gjennom tidsbruken per uke for hvert gruppemedlem og total tidsbruk gjennom prosjektet. På denne måten vil det hele tiden holdes kontroll over hvor mye som har blitt jobbet med hver aktivitet og om det trengs noen umiddelbare endringer i prosjektplanen.

Flere av aktivitetene vil det være naturlig å jobbe med parallellt med andre aktiviteter. Dette er grunnen til at det er flere aktiviteter som går simultant. Disse aktivitetene har da blitt utvidet i tidsplanen slik at det er tatt høyde for at man ikke kommer til å jobbe sammenhengende med aktiviteten i det tidsrommet aktiviteten har fått.



Figur 7: Gantt-diagram

8.2 Budsjett

Utgifter	Pris
AD-kort	10 000,00 NOK
FPGA-kort	0,00 NOK
Komponenter	0,00 NOK
Lisenser	0,00 NOK
Kontorrekvisita	500,00 NOK
Mikrofoner	800,00 NOK
Diverse	700,00 NOK
Sum	12 000,00 NOK

FPGA-kortet og nødvendige lisenser får vi gratis av ARROW. Det samme gjelder transistorer, spenningsforsyning, ADC-er og overspenningsvern. Her vil vi få et mindre antall av de ulike komponentene som produktprøver. Forhåpentligvis får vi nok prøver av komponentene til å dekke vårt behov. Vi må allikevel ta høyde for at vi kan måtte kjøpe noen komponenter. Dette vil i så fall gå under posten «diverse».

Den absolutt største utgiften vil være produksjon av AD-kortet. Vi vet enda ikke hvor komplisert oppbygningen av kortet vil være. En mulighet er å produsere et kort hos Tinius Olsen i Kongsberg, som er en skole. De vil ikke kreve profitt på produksjonen av AD-kort, og vil med høy sannsynlighet være det billigste alternativet. De avgjørende faktorene for hvem vi velger å sette produksjonen bort til er kortets kompleksitet, produksjonspris og produksjonstid. Vi har ikke mulighet til å vente flere måneder på å få AD-kortet produsert etter at vi er ferdige med kretskortutlegget.

Utviklingverktøy som PC har vi ikke regnet som noen utgiftpost i prosjektet. Vi har mulighet til å benytte våre private PC-er i utviklingsarbeidet. I tillegg vil vi ha mulighet til å benytte skolens utstyr. En ting vi må ta høyde for er å forsikre oss om at alle dokumenter som inneholder en teknisk beskrivelse av systemet ikke lekker ut til Posicoms konkurrenter. Dette løser vi ved å publisere dokumentene på en passordbeskyttet nettside. Skolen stiller med et domene, vi har derfor kontroll over hvor serveren for nettsiden befinner seg samtidig som nettsiden er gratis.

Verktøy kan vi låne av HiBu og Posicom, så sant de har det nødvendige utstyret. Noen av gruppemedlemmene har også tilgang til en del verktøy privat. Det er derfor lite trolig at vi må kjøpe inn verktøy.

Motstander og andre enkle komponenter vi trenger i forbindelse med sammensetting og test av systemet kan lånes av HiBu.

9 Om oss

Navn	Initialer	Linje	E-post	Telefon
Kathrine Kaasa	KK	Audioteknologi	kaasa3@live.no	48126516
Erlend Rauk Aasland	ERA	Embedded systems	erlend.aasland@gmail.com	41203446
Fredrik Bakke	FB	Embedded systems	bakkefredrik@gmail.com	97591496
Jardar Hvidsten	JH	Mekatronikk	jardar_h@hotmail.com	40479488
Knut Olav Skyttemyr	KOS	Audioteknologi	ko.skyttemyr@gmail.com	91634894

Gruppen har en fin fordeling mellom elektronikk og datastudenter. Dette er spennende i og med at mye av oppgaven ligger i overgangen mellom den analoge verden og den digitale verden. Dermed har vi muligheten til å lære av hverandre.

Ansvarfordeling:
I og med at dette prosjektet kommer til å bli mye større enn det vi er vant med fra tidligere i studiet, er det hensiktsmessig å fordele ansvar. Den ansvarlige personen skal ikke nødvendigvis gjøre alt arbeidet selv, men har det overordnede ansvar for at ting blir gjort i henhold til fremdriftsplanen. Han/hun har også ansvar for å delegere ut oppgaver i forbindelse med ansvarsområdet sitt.

Kathrine: Gruppeleder, oppfølgingsdokument, presentasjonsansvarlig, prosjektplan.

Erlend: Dokumentansvarlig, risikoanalyse, test FPGA/programvare.

Fredrik: Design og implementasjon FPGA/programvare, web.

Jardar: Design hardware, økonomi, belastningsdiagram, test hardware.

Knut Olav: Kontaktperson oppdragsgiver, forstudie, kravspesifikasjon, hardwarekomponenter.

10 Dokumentoversikt

Planlegging. [P]

Dokumentnavn	Dokumentnummer	Ansvarlig	Første innlevering
Visjonsdokument	P.1	KOS	07.01.2011
Forstudie	P.2	KOS	07.01.2011
Kravspesifikasjon	P.3	KOS, FB	07.01.2011
Testspesifikasjon	P.4	ERA, JH	07.01.2011
Prosjektplan	P.5	KK	07.01.2011
Risikoanalyse	P.6	ERA	07.01.2011
Endringer	P.7	KK	18.03.2011

Systembeskrivelse. [S]

Dokumentnavn	Dokumentnummer	Ansvarlig	Første innlevering
Designdokument	S.1	KOS	18.03.2011
Brukermanual	S.2	KK	30.05.2011
Hovedrapport	S.3	KK	30.05.2011
Aktivitetsrapporter	S.4	ERA	18.03.2011
Mottaksprogram	S.5	ERA	30.05.2011
Programvare på NIOS	S.6	ERA	30.05.2011
Mottaksmodul	S.7	FB	30.05.2011
Etteranalyse	S.8	KK	30.05.2011

Testdokumenter. [T]

Dokumentnavn	Dokumentnummer	Ansvarlig	Første innlevering
Simuleringsrapport	T.1	KK	18.03.2011
Spenningsforsyning	T.2	KOS	18.03.2011
Systemtestrapport	T.3	FB	30.05.2011
Testrapport - AD-kort	T.4	JH	30.05.2011
Testrapport - FPGA/Programvare	T.5	ERA	30.05.2011
Testrapport - ytelse	T.6	KOS	30.05.2011

11 Referanser

[1] Prosjektgruppe 2011-9, *P.3 Kravspesifikasjon.* MMS, 2011.

[2] Prosjektgruppe 2011-9, *P.4 Testspesifikasjon.* MMS, 2011.

[3] Prosjektgruppe 2011-9, *P.7 Endringer.* MMS, 2011.

POSICOM

HØGSKOLEN i Buskerud

Brukermanual		
Dokumenttype	Dokumentnummer	Versjonsnummer
Systembeskrivelse	S.2	1.a
Dato sist endret	Frigitt dokument?	Dokumentansvarlig
27.05.2011	Ja	Kathrine Kaasa
Gruppemedlemmer:		
Kathrine Kaasa, Knut Olav Skyttemyr, Jardar Hvidsten, Fredrik Bakke, Erlend Rauk Aasland		

Innholdsliste

1 Introduksjon....3
1.1 Forkortelser og akronymer....3
2 Deleliste....4
3 Oppkobling....5
4 Konfigurasjonsmuligheter for AD-kort....7
4.1 Single ended eller differensiell konfigurasjon....7
4.2 Spenningsforsyning....8
4.3 Endring av forsterkning....10
5 FPGA-kort....12
5.1 Oversikt....12
6 Mottaksprogramvare....13
6.1 Brukergrensesnitt....13
6.2 Amplitudekorrigering....14
6.3 Filformat....15
6.4 Visualisering....15
7 Feildeteksjon og rapportering....17
7.1 Feiltoleranse....17
7.2 Aktivitet på signallinjer....17
7.3 Testsignaler....18
7.4 Feilrapportering....19
8 Referanser....20

Figurliste

SMA-kabel....4
40-pins-kabel....4
Oppkobling av systemet....5
Skjemategning over modifisert 40-pins kabel....6
Forsterkningsnettverk single-ended konfigurasjon....7
Forsterkningsnettverk differensiell konfigurasjon....8
Spenningsplanet....8
AD-kort med markering av jumpere og skruekonnektorene....9
Forforsterkerkrets....10
Kalkulator fra Texas Instruments. Beregning av motstandsverdier single-ended....11
oversikt – FPGA-kort. Bildet er hentet fra DE2-115 brukermanual....12
Mottaksprogram....13
Dialogboks for amplitudekorrigering....14
Velge ønsket fil å visualisere....15
Eksempel på visualisering i Matlab....16
40-pins kontakt på FPGA-kort. Bildet er hentet fra DE2-115 datablad....18
Hvordan åpne NIOSII-terminal....19

1 Introduksjon

Dette dokumentet er en brukerveiledning for Multimikrofonsystemet. Systemet består av åtte mikrofoner, et AD-kort med åtte kanaler, et FPGA-kort og et mottaksprogram på en ekstern mottaks-PC. Mikrofonene plasseres i et mikrofonarray og kobles til kretskortet med kabler. På kretskortet vil signalene fra mikrofonene bli forsterket av forforsterkerkretsene. De forsterkede signalene blir sendt inn til ADC-en, her blir signalene konvertert fra analoge signaler til en digital bitstrøm. Denne bitstrømmen sendes videre til FPGA-en over en I^2S-buss. Ut fra FPGA-kortet får man UDP-pakker gjennom en Ethernet-kabel som lagres på mottaks-PC-en.

1.1 Forkortelser og akronymer

ADC	-	Analog-to-digital converter
FPGA	-	Field Programmable Gate Array
SMA	-	SubMiniature version A
LED	-	Light Emitting Diode
AD-kort	-	Analog-til-digital-kretskort
USB	-	Universal Serial Bus
GUI	-	Graphical User Interface
IP	-	Internet Protocol
PCM	-	Pulse-code modulation
L/R-klokken	-	Left/Right-klokken
UART	-	Universal Asynchronous Receiver/Transmitter

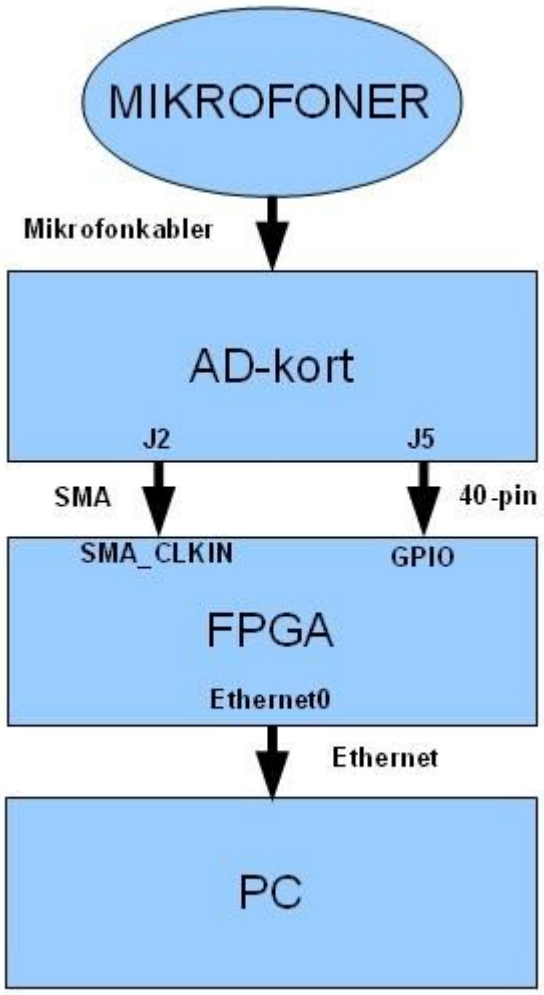
2 Deleliste

- Åtte mikrofoner
- AD-kort
- FPGA-kort med spenningsforsyning
- 40-pins-kabel. Se Figur 4 for skjemategning.
- SMA-kabel, 50 Ω
- Ethernet-kabel, CAT5E eller bedre
- Ekstern mottaks-PC
- USB-kabel for feilrapportering

Figur 1: SMA-kabel

Figur 2: 40-pins-kabel

3 Oppkobling



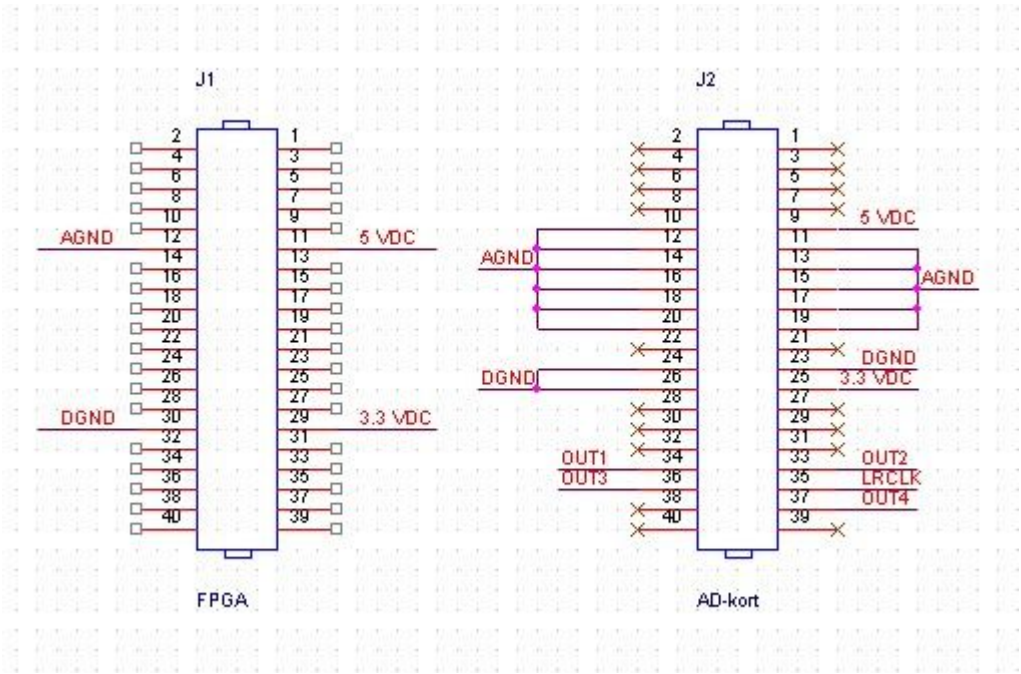
Figur 3: Oppkobling av systemet.

- Koble mikrofonene til AD-kortet.
- Fest 40-pins-kabelen på AD-kortet og FPGA-kortet. **Merk:** På grunn av en designfeil kan ikke en vanlig 40-pins kabel brukes. Se Figur 4 for skjemategning.
- Fest SMA-kabelen på AD-kortet og FPGA-kortet (J2 - SMA_CLKIN).
- Koble på Ethernet-kabelen mellom FPGA ETHERNET0 og PC.
- Sett i spenningsforsyningen til FPGA-kortet.

Systemet er nå koblet opp og klart til å brukes. Se kapittel 8 for hvordan man tar et lydopptak. For oversikt over tilkoblinger på FPGA-kortet – se Figur 11.

Oppkoblingen over forutsetter at man bruker spenningsforsyningen fra FPGA-en. Se kapittel 6.2 for veiledning på hvordan man bruker eksterne forsyninger og for dobbelsjekking at AD-kortet er

konfigurert riktig.



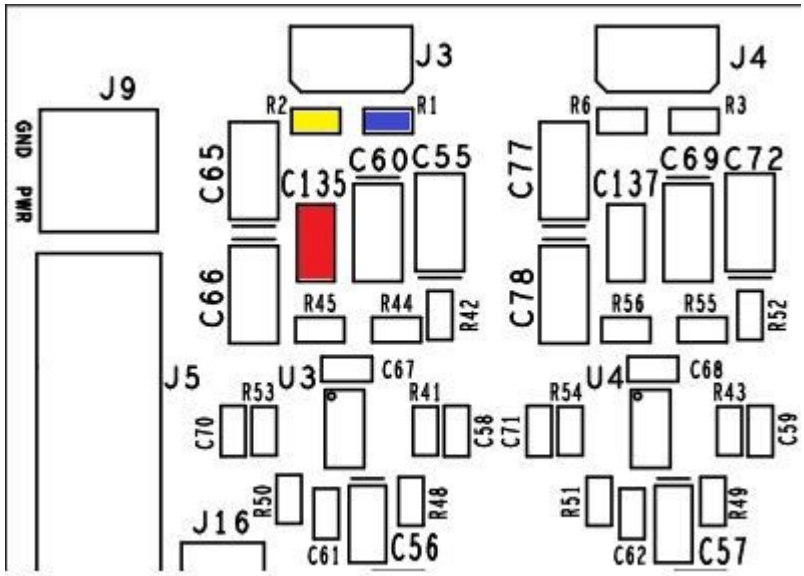
Figur 4: Skjemategning over modifisert 40-pins kabel

4 Konfigurasjonsmuligheter for AD-kort

Dette kortet er designet med tanke på å ha noen valgmuligheter når det gjelder mikrofonkonfigurasjon, bruk av forskjellige spenningsforsyninger og forsterkning.

4.1 Single ended eller differensiell konfigurasjon

Kretskortet er laget for å kunne velge mellom single ended og differensiell konfigurasjon. Å bytte mellom disse to konfigurasjonene krever at vi lodder om noen motstander og kondensatorer i forsterkningsnettverket rundt forforsterkeren.

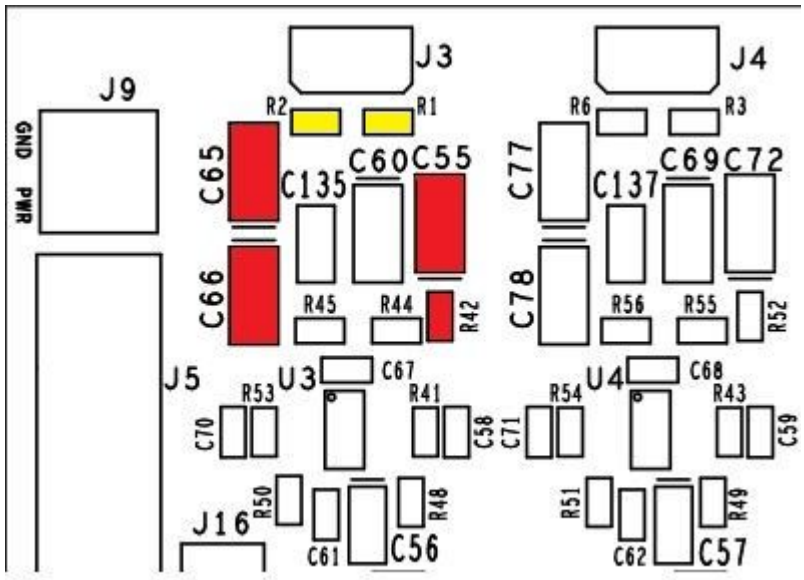


Figur 5: Forsterkningsnettverk single-ended konfigurasjon.

Figur 5 viser hvilke motstander og hvilke kondensator som må tas hensyn til i single ended konfigurasjon. Kondensatoren som er farget rød skal ikke loddes på ved bruk av single ended konfigurasjon. Den gule motstanden skal være 0 Ω, den blå skal være på 3 kΩ. Dette gjelder for alle de åtte kanalene.

Figur 6 viser hvilke motstander og kondensatorer som må ta hensyn til ved bruk av differensiell konfigurasjon. Kondensatorene og motstanden som er farget røde skal ikke være tilkoblet ved differensiell konfigurasjon. De to gule motstandene skal være like og ha verdien 1,5 kΩ. Merk at kondensatorene C135 og C60 skal ha samme verdi. Dette gjelder for alle de åtte kanalene.

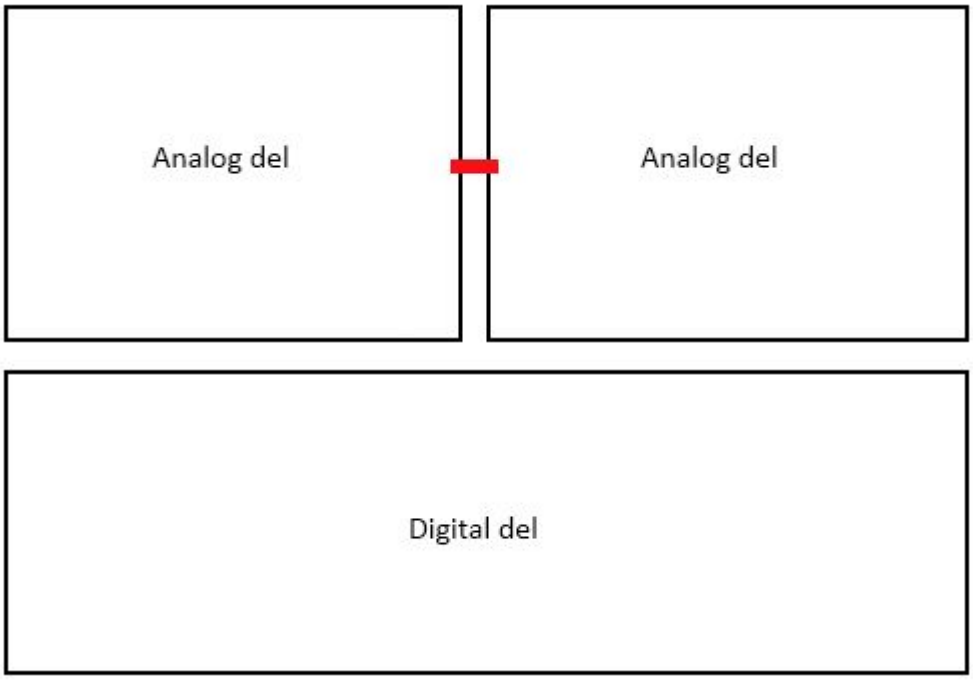
Merk at mikrofonene og forsterkningskretsen må være konfigurert likt!



Figur 6: Forsterkningsnettverk differensiell konfigurasjon.

4.2 Spenningsforsyning

Spenningsplanet er delt opp i tre deler slik som Figur 7 viser.



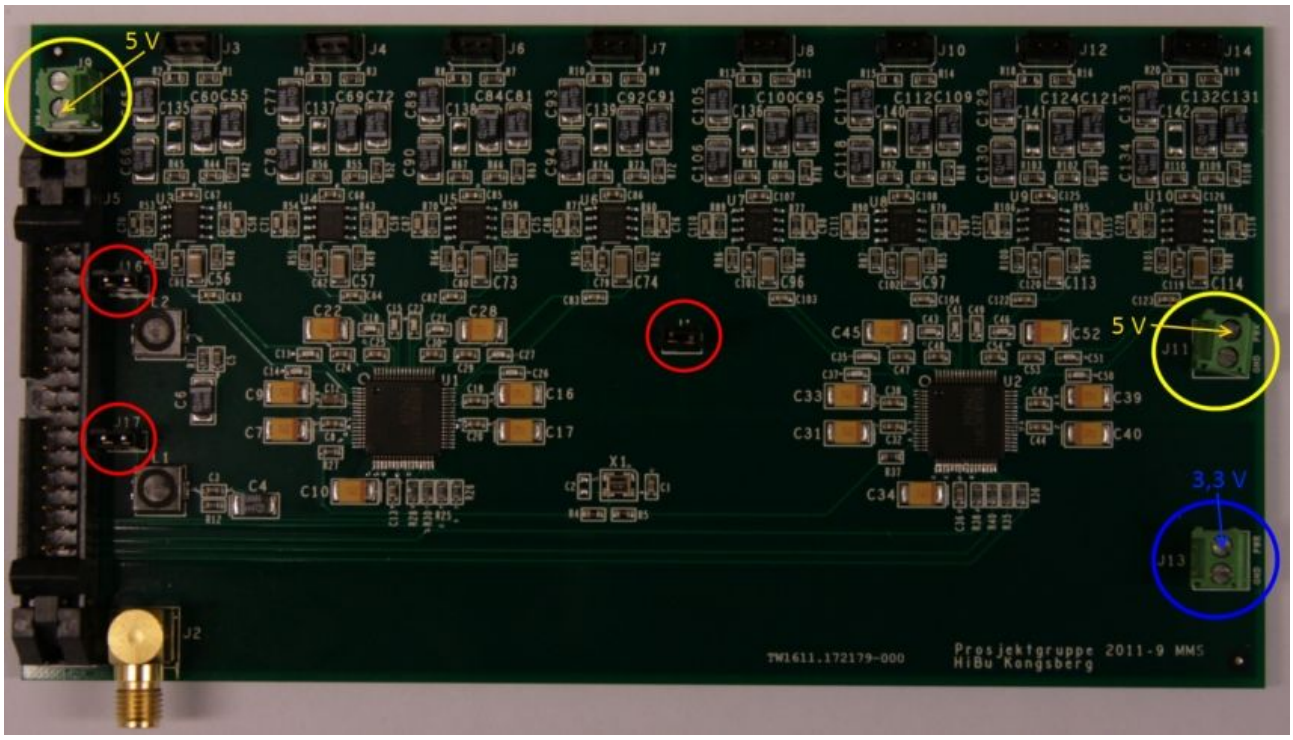
Figur 7: Spenningsplanet

Dette designet gir mulighet til å velge to ulike konfigurasjoner:

- Hente 5 V og 3,3 V fra FPGA-kortet.
- Bruke galvanisk isolerte spenningsforsyninger; to stykk 5 V og en 3,3 V.

De eksterne spenningsforsyningene må være galvanisk isolerte for at vi ikke skal få jordsløyfer inne på kortet.

Vi har valgt denne løsningen for å ha en mulighet til å teste om støyen på spenningsforsyningene fra FPGA-en har negativ innvirking på systemet, og hvor omfattende innvirkningen er.



Figur 8: AD-kort med markering av jumpere og skruekonnektorene.

Ved bruk av spenningsforsyning fra FPGA-kortet:

- Alle de tre jumperene, markert med røde sirkler i Figur 8, skal være tilkoblet.

De to analoge delene blir koblet sammen med en jumper (J1), markert med den rød sirkelen lengst til høyre. Jumperne som sørger for kontakt mellom 40-pins-konnektoren og spenningsplanet (J16 og J17) skal være tilkoblet, de to røde sirklene til venstre.

Ved bruk av eksterne spenningsforsyninger:

- Alle de tre jumperene skal være frakoblet.
- Eksterne spenningsforsyninger kobles til via de grønne skruekonnektorene
  - 5 V kobles til J9 og J11, markert med gule sirkler i Figur 8.
  - 3,3 V kobles til J13, markert med blå sirkel i Figur 8.

Husk: Positiv linje på eksterne spenningsforsyninger skal til «PWR» på skruekonnektorene, mens den negative skal til «GND» (merket på AD-kortet).

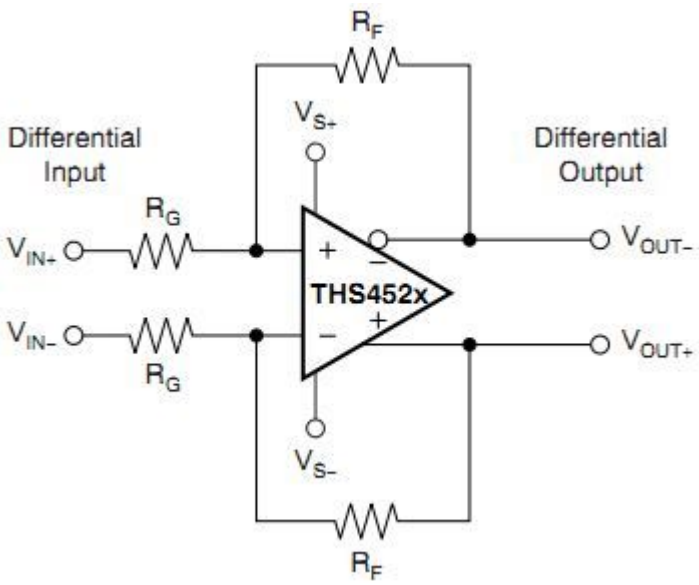
Både 40-pins-kabelen og SMA-kabelen skal fortsatt være tilkoblet.

Merk: Eksterne spenningsforsyninger må **ikke** brukes sammen med spenningsforsyninger fra FPGA-kortet!

4.3 Endring av forsterkning

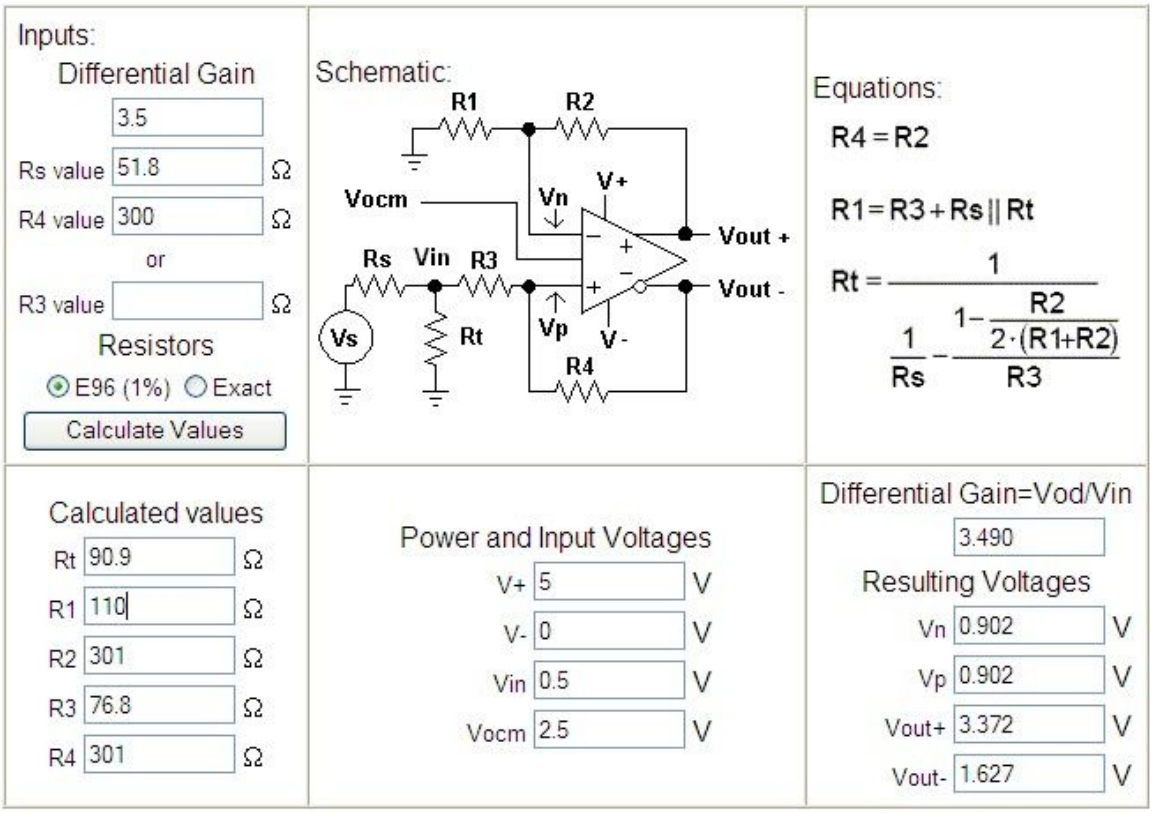
Det er mulig å endre signalforsterkningen ved å endre noen motstandsverdier i forforsterkerkretsen. Dette er en lett beregning å gjøre ved bruk av differensiell konfigurasjon, da er det R_f og R_g som har en innvirkning på forsterkningen, se Figur 9. Forsterkningen blir da:

$$G = \frac{R_f}{R_g} \tag{1}$$



Figur 9: Forforsterkerkrets.

Ved bruk av single ended konfigurasjon er regnestykket noe vanskeligere. Det må da foretas en impedanstilpasning, men det finnes en løsning på dette også. Man kan bruke en ferdiglaget kalkulator fra Texas Instruments, se Figur 10. Link til denne finnes vedlagt [1].

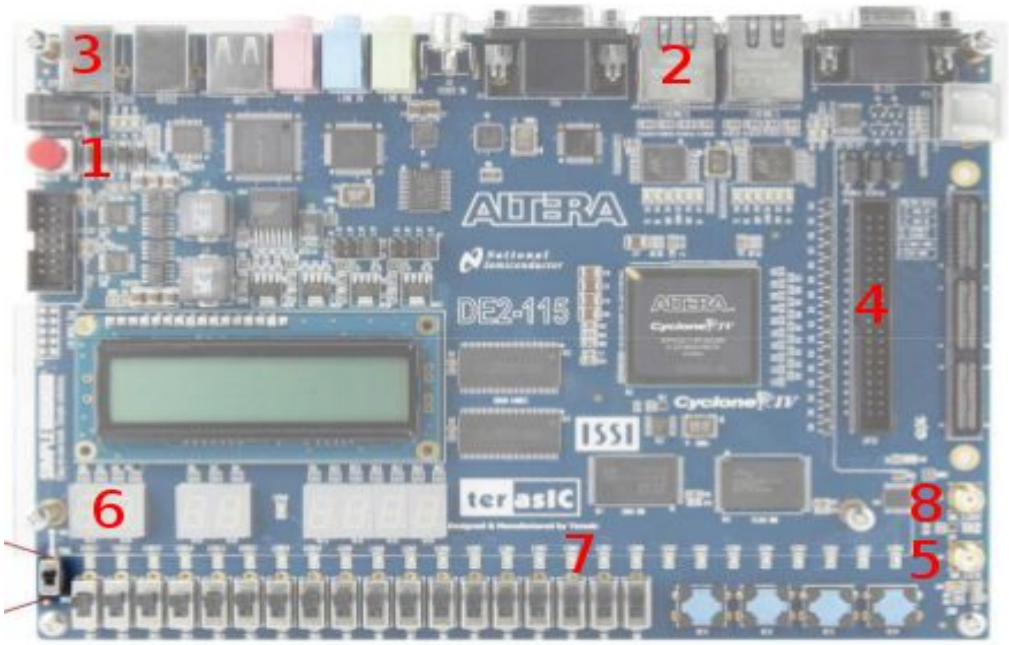


Figur 10: Kalkulator fra Texas Instruments. Beregning av motstandsverdier single-ended.

5 FPGA-kort

På FPGA-en og FPGA-kortet er alt som trengs for å overføre data mellom ADC-er og datamaskin. Etter at alt er koblet opp og FPGA-kortet skrudd på, skal styringen skje fra mottaksprogrammet på mottaker-PC. Kortet har likevel fått ulike muligheter for feildetektering og rapportering. Alle kontakter og grensesnittelementer vi bruker er markert i Figur 11. For en generell brukermanual for FPGA-kortet se referanse [2].

5.1 Oversikt



Figur 11: oversikt – FPGA-kort. Bildet er hentet fra DE2-115 brukermanual

1. Av/På-bryter.
2. ETHERNET0 – for dataoverføring til PC.
3. USB Blaster port – for debugging.
4. 40-pins-konnektor. Kobles til AD-kort – har også hjelpemidler for feilsjekking.
5. SMA klokkeinngang, for overføring av bitklokke.
6. Syv-segment. Viser feilkode.
7. LEDR indikerer om det ikke er signal på inn-portene.
8. SMA klokkeutgang. For feilsjekking.

6 Mottaksprogramvare

For å koble FPGA til mottaks-PC er det nødvendig at mottaks-PC er satt opp med fast IP-adresse og ikke DHCP (å motta IP automatisk). FPGA-kortet er satt opp med IP-adressen 192.168.21.171. Mottaks-PC må derfor settes opp med en IP-adresse innenfor det samme segmentet.

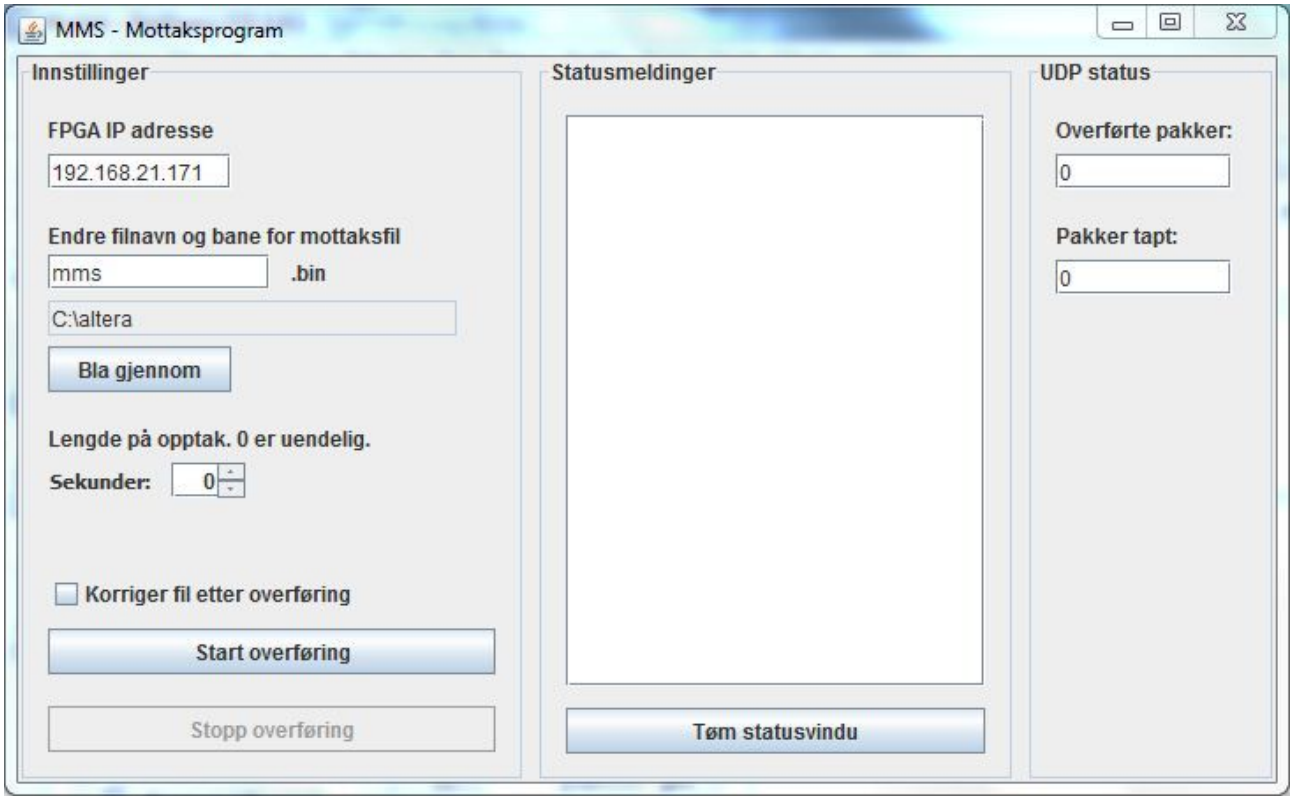
På mottaker-PC velg IP-adressen: 192.168.21.[0-255] og Subnet mask: 255.255.255.0.

6.1 Brukergrensesnitt

Figur 12 viser det grafiske grensesnittet for mottaksprogrammet. Her kan det gjøres følgende innstillinger:

- Endre IP-adressen til FPGA-kortet.
- Endre filnavnet og filplassering for hvor mottatt data lagres.
  - Velg «bla gjennom» for å endre målmappe.
- Angi antall sekunder opptaket skal vare.
- Slå på korrigering av amplitude, og endre skaleringkonstant.

Når ønskede endringer er utført, trykkes «start overføring» knappen.



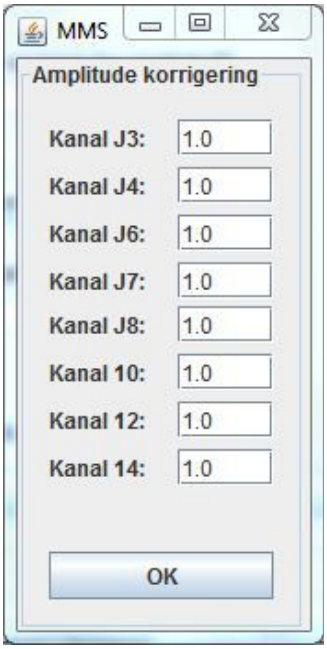
Figur 12: Mottaksprogram

Ved trykk på «start overføring» knappen vil et startsignal vil sendes til FPGA-kortet som forteller at dataoverføringen kan settes igang. Når ønskelig mengde data er mottatt stanses overføringen ved å trykke «stopp overføring» knappen og et stoppsignal vil sendes til FPGA (*Dersom et antall sekund med dataoverføring er valgt, behøver ikke stopp-knappen trykkes. Da dette skjer automatisk*). Overført data er nå lagret i filen som spesifisert i innstillingene.

Nå er FPGA-kortet og mottaks-PC klar for en ny overføring om dette er ønskelig. Før en ny overføring er det mulig å endre innstillingene. I midtre rute skrives det ut statusmeldinger som beskriver hvilke operasjoner som blir utført. I høyre rute skrives antall overførte og antall tapte datapakker.

6.2 Amplitudekorrigering

Dersom valget for korrigering av fil ble aktivert, er det mulig å stille inn hvilken konstant amplituden skal skaleres med, se Figur 13. Standardverdi, uten korrigering, er '1.0'.



Figur 13: Dialogboks for amplitudekorrigering

Man må man beregne noe tid etter overføring før denne korrigeringsprosessen blir ferdig. Melding om dette vil bli gitt i statusvinduet. Denne prosessen vil generere en ny fil, med navn: «[*filnavn*].bin_corr». Filen vil inneholde de gitte endringene.

6.3 Filformat

Filen som lagres består av rådata fra AD-kortet på PCM-format – og inneholder ingen header eller info om lengde, frekvens eller annen informasjon. Samplefrekvens er 96 kHz, og oppløsning er 24 bit. Data lagres på toerkomplementform i kanalrekkefølge som er vist i tabellen under – 24 bit for hver kanal – little endian.

Tabell 1: Kanalrekkefølge i fil, kanalnavn henviser til mikrofontilkobling på AD-kort.

Kanalrekkefølge i fil							
J3	J4	J6	J7	J8	J10	J12	J14

6.4 Visualisering

Filen som blir generert av mottaksprogrammet kan visualiseres i Matlab. Visualiseringen vil vise signalene i hver kanal. Dette gjør det lettere å feilsøke, da man kan lese ut amplitude, frekvens og eventuell støy. Matlabscriptet leser 24 bit av gangen som et heltall med fortegn, og plotter disse heltallene i en graf.

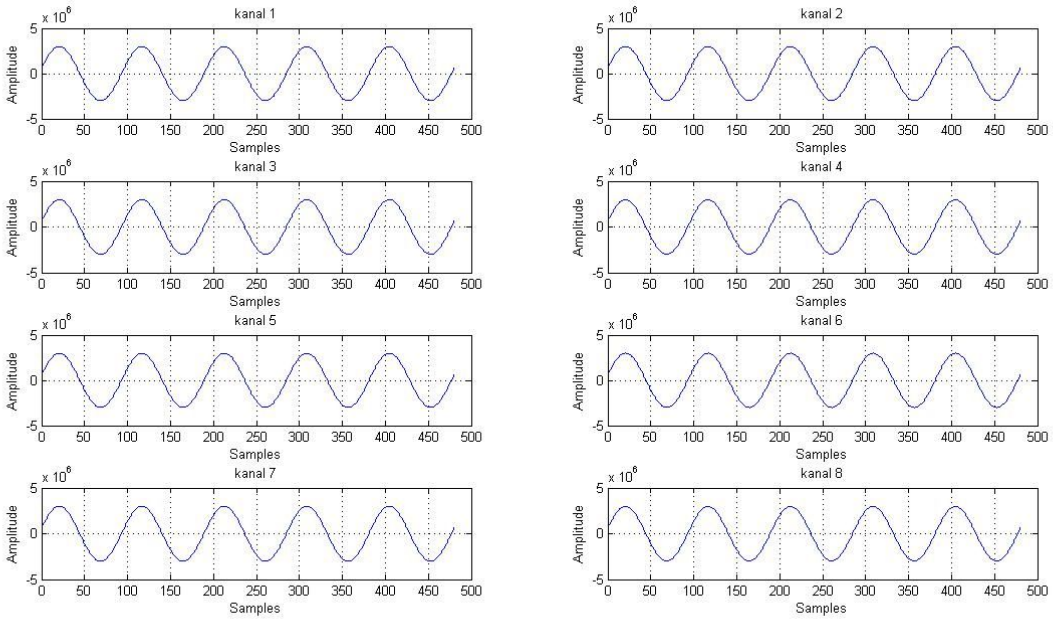
For å få opp grafene må en sørge for at scriptet leser av den ønskede filen, dette gjøres ved å skrive inn filnavnet i linje 11 i scriptet, se Figur 14.

```
11 -    file = '8micJ10_corr.log'; % Skriv inn ønsket filnavn her
12 -    fid = fopen(file);
```

Figur 14: Velge ønsket fil å visualisere

Når man trykker «kjør» (F5), er det mulig man får opp en dialogboks som spør om man vil endre mappen Matlab leter etter filer i. Trykk «Change directory» om dette kommer opp.

Et eksempel på visualisering av et sinussignal er vist i Figur 15. Her er det sendt inn et sinussignal på alle kanalene, gjort et opptak og filen er lastet inn i Matlab for visualisering.



Figur 15: Eksempel på visualisering i Matlab

7 Feildeteksjon og rapportering

For å avdekke feil med tilkoblinger og systemfeil under kjøring har FPGA-systemet ulike feilrapporteringsmekanismer. I tillegg til å mate inngangspinnene med kompatible signaler.

7.1 Feiltoleranse

Flere ting kan gå galt med et så kompleks system som dette. Hvis komponenter på FPGA-kortet skulle feile, er det stor sannsynlighet for at systemet blir ubrukelig. Dette kan det gjøres lite med, men det kan rapporteres for eksempel gjennom NIOSII-terminal. Hvis derimot noe skulle feile før FPGA-kortet i signalflyten, kan det hende samplingen kan fortsette på redusert kapasitet.

Feil med bitklokke eller ordklokke vil gjøre systemet ubrukelig, siden de serielle datalinjene er synkronisert mot disse. Hvis det skulle være feil med en eller flere serielle datalinjer vil systemet fortsatt overføre data fra alle linjene. Signalindikatorene på linjene kan da indikere hvilken linje som som ikke er i orden, og samples må ignoreres fra disse når filen med rådata brukes.

7.2 Aktivitet på signallinjer

For å lett kunne fastslå om det kommer signal inn på L/R-klokken (LRCK), og serielle datalinjer – vil LEDR1-5 indikere om dette ikke er tilfelle. Disse varsler oss når den assosierte linjen står konstant høy eller lav, og er skrudd av når alle linjene får signal inn som normalt. Hvor lang tid en linje kan ha konstant signal uten at LED-en indikerer er avhengig av signalet som er forventet inn på linjen.

Tabell 2: Aktivitet på signallinjer

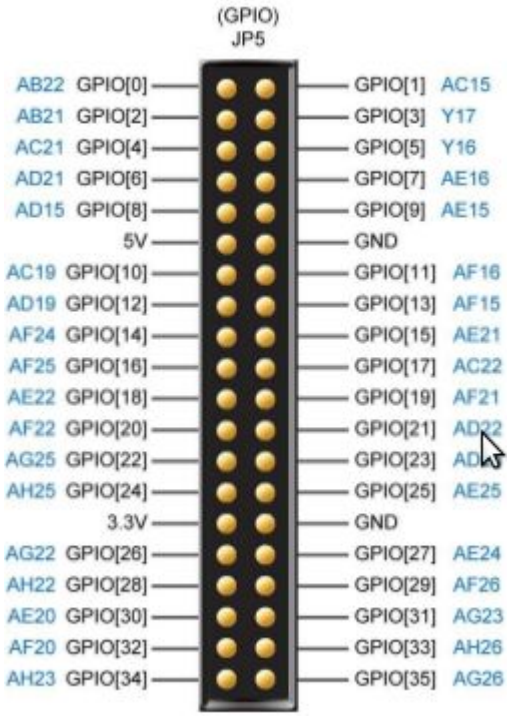
LEDR	Beskrivelse	Kontakt FPGA-kort	Kontakt AD-kort
1	Left-right clock	GPIO30	
2	kanal J3 og J4	GPIO29	SDOUT1, ADC_master
3	kanal J6 og J7	GPI28	SDOUT2 , ADC_master
4	kanal J8 og J10	GPIO31	SDOUT1, ADC_slave
5	kanal J12 og J14	GPIO32	SDOUT2, ADC_slave

7.3 Testsignaler

For å gjøre det enklere å feilsøke på FPGA-kortet, har den en innbygd signalgenerator. Denne lager signaler som er kompatible med inngangsgrensesnittet på FPGA-en. Signalene er tilgjengelige på GPIO-pinner på 40-pins-konnektoren, og på SMA-klokkeutgang.

Tabell 3: Kobling av testsignaler

Signal	Beskrivelse av signal.	Hentes fra pinne	Kobles til (forslag)
Bitklokke	6,25 Mhz klokkesignal	SMA clock out	SMA clock in
Left-right clock	Periodisk klokke på (6,25/64) MHz	GPIO4	GPIO30
SD0	Sekvens bestående av tallene 0-255 (teller rundt) formatert som 32 bits unsigned integer (MSB først)	GPIO0	GPIO29
SD1	Annenhver bit 0 og 1	GPIO1	GPIO28
SD2	Annenhver bit 0 og 1	GPIO2	GPIO31
SD3	Annenhver bit 0 og 1	GPIO3	GPIO32



Figur 16: 40-pins kontakt på FPGA-kort. Bildet er hentet fra DE2-115 datablad.

7.4 Feilrapportering

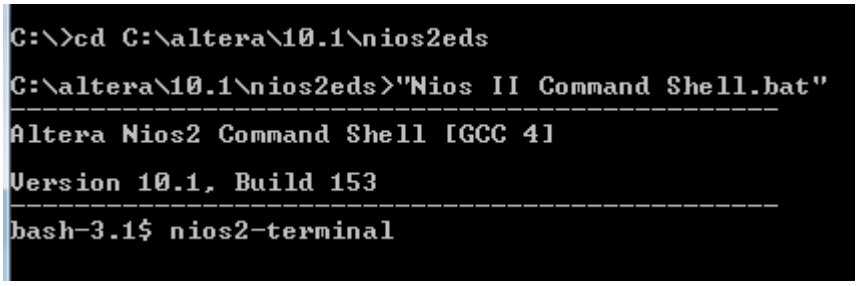
Hvis en feil skulle oppdages mens systemet kjører, vil feilkoden vises på syv-segmentet HEX7 (oppføring 7 i Figur 11).

Tabell 4: Feilkoder

Feilkode	Beskrivelse
1	Opprettelse av socket mislyktes.
2	Bind socket mislyktes.
3	Feil i mottak av startbit nummer 1.
4	Feil i sending av data til mottaks-PC.
5	Initialisering av FIFO buffer feilet.
6	Initialisering av DMA feilet.

Løsningen dersom en av disse feilkodene vises, er stort sett å restarte FPGA-kortet. Feilkodene vil være til hjelp dersom et problem oppstår ofte, og kildekoden må endres.

En mer detaljert feilmelding/statusmelding skrives ut til NIOSII-terminal. NIOSII-terminalen monitorerer meldinger som rutes gjennom JTAG UART kanalen fra FPGA-kortet. Dette forutsetter at USB-kabelen til FPGA-kortet er koblet til. Denne kobles fra en USB-port på mottaks-PC til USB Blaster porten (nummer 3 i Figur 11). NIOSII-terminalen åpnes ved å først åpne «NIOS II command shell» fra programmenyen. Eller kjøre «NIOS II Command Shell.bat» fra C:\altera\10.1nios2eds\ som vist i Figur 17. Fra command shell åpnes NIOSII-terminal ved å taste inn «nios2-terminal». Altera sin USB-blaster driver må på forhånd være installert.



Figur 17: Hvordan åpne NIOSII-terminal

8 Referanser

[1] Texas Instruments online kalkulator, sist besøkt 27.04.11
http://www-k.ext.ti.com/srvs/cgi-bin/webcgi.exe?Company={5761bcd8-11f5-4e08-84e0-8167176a4ed9},kb=analog,case=obj%2812223%29,new

[2] *DE2-115 User manual V1.0.,* Terasic Technologies Inc, 2010 [DE2_115_User_manual.pdf]

POSICOM

HØGSKOLEN i Buskerud

Hovedrapport		
Dokumenttype	Dokumentnummer	Versjonsnummer
Systembeskrivelse	S.3	1.a
Dato sist endret	Frigitt dokument?	Dokumentansvarlig
30.05.2011	Ja	KK, KOS, JH, FB, ERA
Gruppemedlemmer:		
Kathrine Kaasa, Knut Olav Skyttemyr, Jardar Hvidsten, Fredrik Bakke, Erlend Rauk Aasland		

Sammendrag

Denne prosjektoppgaven, gitt av Posicom AS, har gått ut på å lage en hardwareprototype bestående av et AD-kort for digitalisering av data og et FPGA-system for videresending over Ethernet. Vi lagde et kretskort med åtte kanaler. Mikrofonsignalene digitaliseres med en samplingfrekvens på 96 kHz og 24 bits oppløsning. Etter digitaliseringen blir dataene sendt til FPGA-kortet over en I^2S-buss. FPGA-en pakker dataene i UDP-pakker, sender de til en mottaks-PC og et PC-program vi har laget lagrer dataene til fil.

Biasnettverket til mikrofonene ble laget inne på kretskortet først i signalkjeden. Neste komponent i signalkjeden er forforsterkeren. Denne har som oppgave å forsterke signalene, i vårt tilfelle med en faktor på 1,75. I tillegg flytter den referansespenningen fra 0 til 2,5 VDC, fordi ADC-en trenger dette. Vi har brukt to stykk ADC-er med fire kanaler i hver pakke. ADC-en mottar systemklokke fra en krystall oscillator og sender videre en seriell digital bitstrøm over I^2S-bussen.

Kretskortet ble laget som et firelagskort, med signalbaner på de to ytterste lagene. Komponenter ble kun plassert på topplaget. Vi benyttet oss av et jordplan og spenningsplan. Disse ble delt opp i seksjoner, to analoge 5 volt seksjoner og en digitalt 3,3 volt seksjon. Seksjonene på jordplanet ble koblet sammen rett under ADC-en, fordi ADC-ene må ha felles jordreferanse på analog og digital side av komponenten.

Kretskortet blir forsynt med spenning fra FPGA-kortet. Vi har også lagt inn muligheten for å bruke eksterne spenningsforsyninger, fordi det er en del støy på spenningsforsyningen fra FPGA-kortet.

FPGA-en gjør jobben med å lese data fra AD-kortet og videresende til mottaker-PC. Kortet vi bruker er et utviklingskort fra Terasic, og på det har vi hatt tilgjengelig det vi har trengt av RAM og fysisk grensesnitt mot Ethernet.

Inne på FPGA-en har vi laget et system som leser data fra AD-kortet og legger dette i et ringbuffer. Videre bruker vi en «soft-core» prosessor med UDP/IP protokollstakk for å pakke dataene i UDP/IP pakker. Pakkene sendes så til en Ethernet MAC på FPGA-brikken – og går derfra over Ethernet-kabel til mottaker-PC.

På mottaker-PC-en kjører et program som lar brukeren lagre innkommende data til fil og starte og stoppe datastrømmen. Her kan også amplitudene etterkorrigeres for å kompensere for ulikheter mellom kanalene. Etter at dataene er lagret kan de visualiseres i MATLAB.

Testresultater viser at systemet fungerer tilfredsstillende. Det er likevel rom for forbedringer og mulighet for utvidelse. Systemet har et signal-støy-forhold på ca. 14 bit. Ved bruk av eksterne spenningsforsyninger øker dette forholdet til ca. 16 bit. Lokale spenningsregulatorer vil kunne bedre dette ytterligere. Ellers har vi oppnådd en tilfredsstillende synkronisering mellom kanalene langs tidsaksen. Amplitudemessig er det litt forskjeller på kanalene på grunn av toleranser på de analoge komponentene. Dette kan fininnstilles ved hjelp av digital skalering i etterkant.

Systemet har ingen problemer med å takle den store datastrømmen. Ved å gå ned til 16 bits oppløsning er det rom for å doble antall kanaler. Systemet oppfører seg stabilt over tid, uten problemer med pakketap eller lignende.

Abstract

In this bachelor project our goal was to make a hardware prototype. The prototype consists of an analog-to-digital conversion PCB and a FPGA-system that use Ethernet to transmit data. We made a PCB with eight microphone channels. The signals from the microphones are digitalized using a sampling frequency of 96 kHz and 24 bit resolution. The digitalized data are transmitted to the FPGA-system using an I^2S-bus. Further, the FPGA-system uses the UDP protocol to transmit data to the connected computer where it is stored by a receiver-program.

The microphones need to be powered with 5 VDC. To amplify the signals we use an operational amplifier. In our system the gain factor is 1,75. In addition the amplifier raise the common voltage from 0 V to 2,5 V. The ADC in our system needs a common voltage of 2,5 V. We use two ADC's with four channels in each chip. The ADC needs a system clock, that comes from the crystal oscillator.

The PCB we made is a four layer board. The top and bottom layer are used as signal layers. Due to production costs, components are placed on the top layer only. The two inner layers are used as ground plane and power plane. These planes are divided into two analog 5 V sections and one 3,3 V digital section. The ADC need common ground on both the analog and digital side of the chip. Because of this, the ground plane is connected together below the ADC chip.

The PCB is powered from the FPGA. In addition we have decided to add connectors for external power supplies. The reason for this is the amount of noise from the FPGA power supply.

For the digital part of the system, we have used a development kit from Terasic with a FPGA from Altera. The kit contains all we need of RAM and Ethernet interfaces. On the FPGA-chip, we have made a system for receiving data from the I^2S-bus and placing it in a ring-buffer. We use a soft-core processor with an UDP/IP protocol stack to embed data in UDP packets. The packets are sent to an Ethernet MAC and further on through an Ethernet cable to the receiver.

On the receiver-end, the user can store the incoming data to a file. The software also lets the user scale the amplitude of individual channels, to compensate for differences between the channels due to component tolerances. The stored data can be viewed in MATLAB with a provided script.

The test results confirm that our system works satisfactory, but there is still room for improvement. We have measured a signal to noise ratio of approx. 14 bit. When using external power supplies, the ratio increases to approx. 16 bit. We believe that voltage regulators located on the PCB would improve this further. Other important things we have tested is how much the channels differ from each other. In the time domain, we have seen that the channels are in satisfactory synchronization. But we have seen some differences on the amplitude of the different channels. This can be compensated for in the software on the receiver.

The system has no trouble with the huge data rate, and there are no troubles with loss of data. We have also seen that if we reduce the resolution to 16 bit, the number of channels can be doubled with only small modifications to the receiver-module on the FPGA.

Innholdsliste

Sammendrag 2
Abstract 3
1 Introduksjon....8
2 Moduler AD-kort....11
2.1 Komponenter....11
2.2 Mikrofon-grensesnitt....11
2.3 Overspenningsvern....12
2.4 Forforsterker....13
2.4.1 Differensiell konfigurasjon....14
2.4.2 Single ended konfigurasjon....16
2.5 Klokke....19
2.6 ADC....19
2.7 Grensesnitt mot FPGA og spenningsforsyning....21
3 PCB-design 25
3.1 Jordplan 25
3.2 Spenningsplan....27
3.3 Signalplan....28
3.4 Annet....31
4 FPGA-design....33
4.1 Forklaring av begreper....33
4.2 Arkitektur....34
4.2.1 Modifikasjon av referansedesign....35
4.2.2 Systembuss....36
4.2.3 Operativsystem....36
4.3 Moduldesign....36
4.3.1 Triple-speed Ethernet....37
4.3.2 Mottak....38
4.3.3 FIFO....39
4.3.4 DMA....40
4.3.5 FIFO og DMA buss-grensesnitt....42
4.3.6 Fra SDRAM til Ethernet....42
4.3.7 Prosessor og RAM....42
4.3.8 Ressursbruk og kompileringsrapport....42
4.4 Mottak og lagring på PC....44
4.4.1 Mottaksprogram....44
4.4.2 Filformat....45
4.4.3 Visualisering....46
4.5 Systemfeil....46
4.5.1 Toleranse....46
4.5.2 NIOS-rapportering....46
4.5.3 Sjekk av linjeaktivitet....47
4.5.4 AD-signalgenerator....47
5 Testresultater....49

5.1 Simulering av analog krets....49
5.2 Test av AD-kort....49
5.3 FPGA-test....50
5.4 Ytelsetester....50
5.5 Systemtest....50
6 Forslag til forbedringer....51
7 Konklusjon....53
8 Referanser....54
9 Vedlegg....56
9.1 Vedlegg I – Matlabprogram for visualisering....56
9.2 Vedlegg II – Fullstendig kretsskjema....57

Figurliste

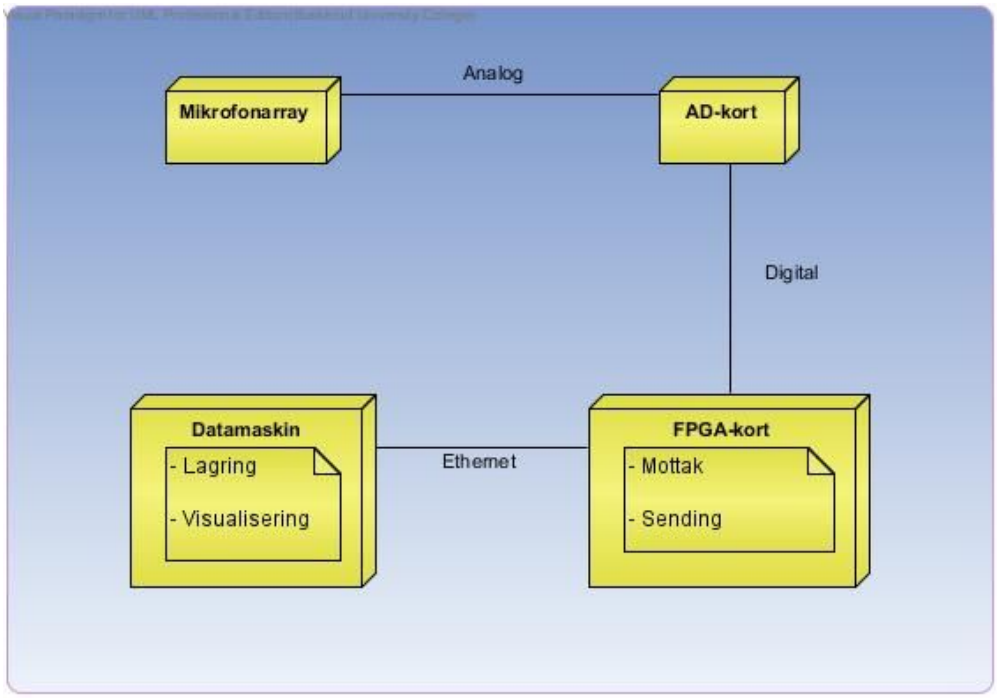
Figur 1: Blokkdiagram over oppgaven....8
Figur 2: Arkitektur AD-kort....9
Figur 3: Arkitektur FPGA....10
Figur 4: Mikrofonkonfigurasjoner....12
Figur 5: Skjemategning av forforsterker konfigurert for differensiell inngang....14
Figur 6: Filterkarakteristikk for båndpassfilter med 1,75 VPut/VPinn i forsterkning....15
Figur 7: Utgangssignal ved differensiellt inngangssignal på 1 Vp og 1,75 VPut/VPinn i forsterkning. 15
Figur 8: Utgangssignal med 1,75 VPut/VPinn i forsterkning og inngangssignal på 1,41 VP....16
Figur 9: Skjemategning for forforsterkerkretsen med single ended konfigurasjon....16
Figur 10: Online impedanskalkulator fra Texas Instruments....17
Figur 11: Frekvenskarakteristikk forforsterker....18
Figur 12: Utgangssignal med single ended konfigurasjon og 1 Vp inngangssignal. 3,5 i differensiell forsterkning....18
Figur 13: Støymåling av spenningsforsyning fra FPGA....22
Figur 14: Utgangspunkt for filterdesign....23
Figur 15: Filterkarakteristikk for spenningsforsyningsfilter....24
Figur 16: Oppdeling 4-lags kort....25
Figur 17: Jordplan med inndeling i en digital del og to analoge deler. 26
Figur 18: Jordplan med inndeling i en analog og en digital del....27
Figur 19: Spenningsplan med jumper mellom analoge deler....28
Figur 20: Vollgrav rundt klokkekrets, bilde hentet fra [18]....29
Figur 21: Avkoblingskondensator analoge kretser, bilde hentet fra [18]....30
Figur 22: Avkoblingskondensator digitale kretser, bilde hentet fra [18]....30
Figur 23: Thermal relief, bilde hentet fra [18]....31
Figur 24: Oversikt over moduler i SOPCbuilder....33
Figur 25: Dataflyt og utvalgte moduler....35
Figur 26: Modulhierarki....37
Figur 27:Blokkdiagram, Mottak....38
Figur 28: Fra 24 bits sampler på I2S buss, til 32 bits word i systemet....39
Figur 29: Parametrisering av FIFO....40
Figur 30: DMA-parametrisering....41
Figur 31: GUI for mottaksprogram....45
Figur 32: Linjesjekkmodul....47
Figur 33: Signalgenerator....48

Forkortelser og akronymer

AD	-	Analog til Digital
ADC	-	Analog-to-digital converter
PCB	-	Printed Circuit Board
SNR	-	Signal-to-noise ratio
IC	-	Integrated Chip
I^2S	-	Inter-IC sound (Integrated Interchip Sound)
XO	-	Krystalloscillator
SMD	-	Surface mount devices.
EMC	-	Electromagnetic compatibility
SPL	-	Sound Pressure Level
mil	-	milli-inch
FIFO	-	First In, First Out
DMA	-	Direct Memory Access
FPGA	-	Field Programmable Gate Array
LE	-	Logic Elements
RTOS	-	Real Time Operating System
SoC	-	System on a Chip
MAC	-	Medium Access Controller
PCS	-	Physical Coding Sublayer
PCM	-	Pulse Code Modulation
AMT	-	Avalon Master Template
M9k	-	Minneblokker på FPGA
SOPCBuilder	-	System On a Programmable Chip Builder
JFET	-	Junction Field Effect Transistor

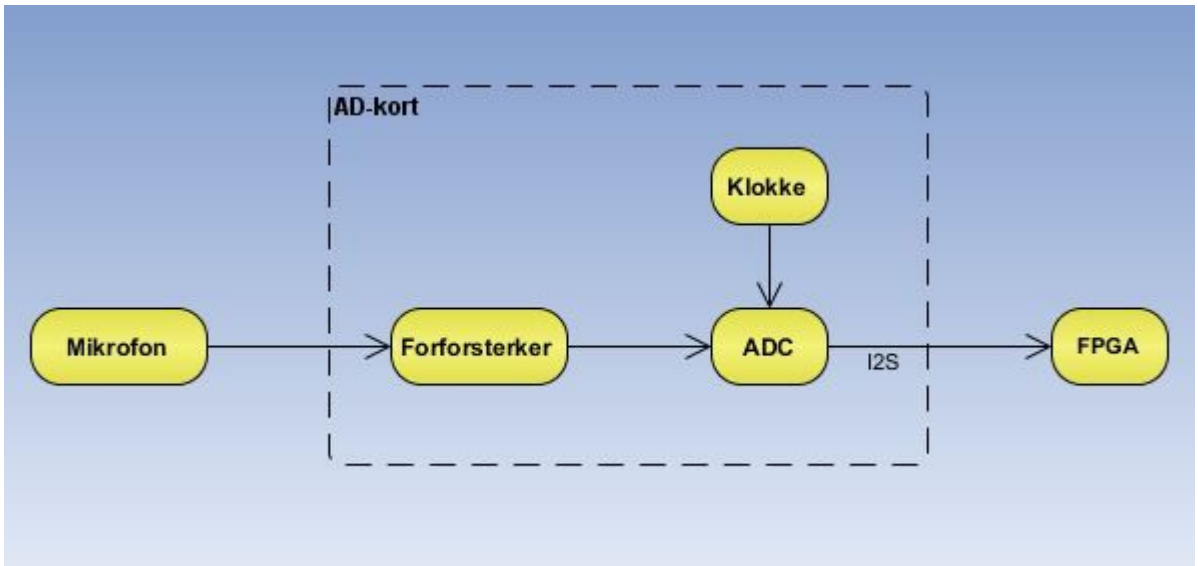
1 Introduksjon

Denne oppgaven ble gitt av Posicom AS i Kongsberg. Oppgaven er gitt i forbindelse med et system kalt Snipos. Dette systemet bruker mikrofoner for å fange opp skuddlyder, og på bakgrunn av det regner de ut avstand og retning til våpenet. Dagens system har fire mikrofoner. En av de viktigste utfordringene Snipos har i dag er å få økt rekkevidden (signal-støy forhold). Oppgaven gikk ut på å lage en hardwareprototype for åtte mikrofoner. Med åtte mikrofoner kan rekkevidden til systemet i teorien økes til det dobbelte. Prinsippet som brukes til dette er beamforming. Se Figur 1 for et blokkdiagram over oppgaven. Det mekaniske designet av mikrofonarrayet er viktig for å få til beamforming. Dette er utenfor vårt fagfelt og var ikke en del av vår oppgave.



Figur 1: Blokkdiagram over oppgaven

AD-kortet (Analog til Digital) skal omforme analoge mikrofonsignaler om til digitale signaler. Dette kretskortet har vi laget fra bunnen av. Vi har valgt å bruke delta-sigma konvertere i designet vårt. Delta-sigma konvertere er ofte brukt til digitalisering av lyddata. De har gode egenskaper når det kommer til signal-støy-forhold [48]. Resten av komponentene er valgt for å passe sammen med ADC-en (Analog-to-digital converter). Se Figur 2 for oversikt over AD-kortet.



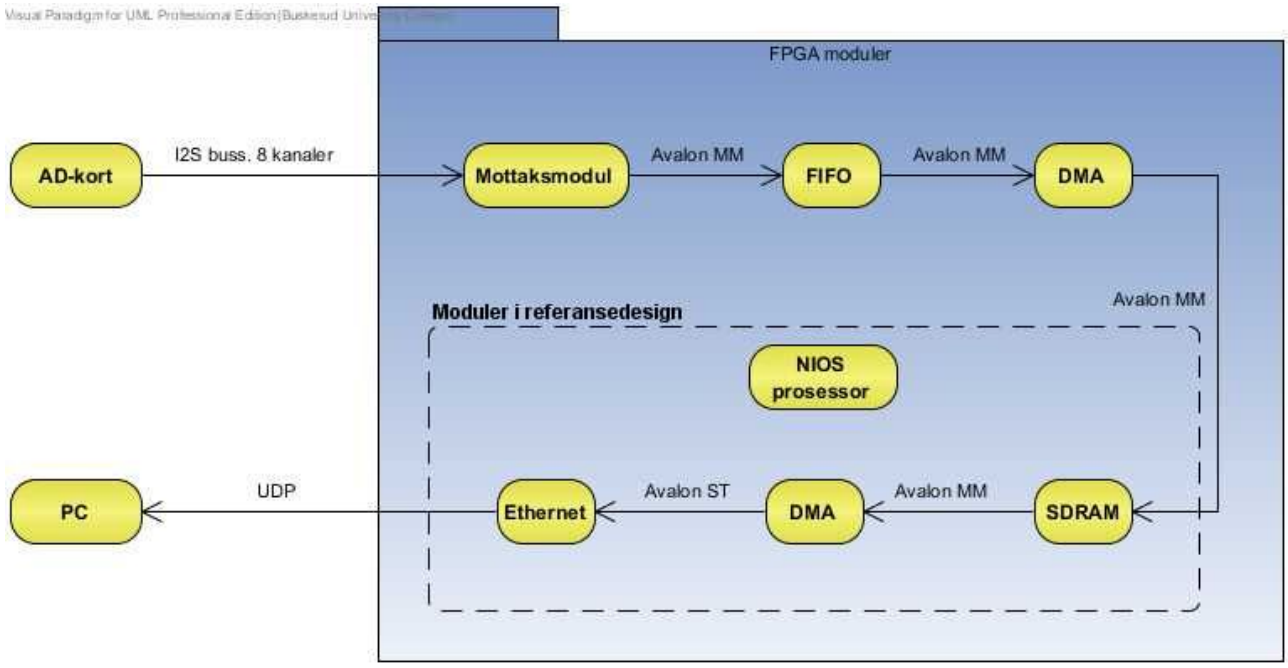
Figur 2: Arkitektur AD-kort

Vi benytter et FPGA-kort (Field Programmable Gate Array), hovedsakelig da oppdragsgiver har dette i sitt Snipos-system. En FPGA er svært anvendelig for å implementere et system som vi gjør. Det finnes såkalte IP-cores, som er ferdiglagde moduler for blant annet CPU (Central Processing Unit), FIFO (First In, First Out) og minne. Slike moduler gjør det raskere å utvikle et stort og komplekst system uten å utvikle alt fra bunnen av. Vi har brukt IP-cores (Intellectual Property) aktivt i vår utvikling, men også lagd våre egne moduler der det var behov for det.

Vi har fått et FPGA-kort fra Terasic som vi har brukt til mottak, UDP-pakking (User Datagram Protocol), og sending over Ethernet. Ethernet benyttes da det er et hendig og robust dataoverføringsmedium, med støtte for høye hastigheter. I forhold til USB- og Firewire-protokollene er Ethernet lett å implementere og bruke. Vi implementerte vårt system ved å skrive kode i C og VHDL, og ved hjelp av utviklingsverktøy fra Altera. For at ikke oppgaven skulle bli for stor tok vi utgangspunkt i et referansedesign for FPGA-kortet som vi tilpasset til vår bruk. Et referansedesign er en slags demo av et tenkt system som kan implementeres på FPGA-kortet. I vårt tilfelle; en webserver. Se Figur 3 for et blokkdiagram over arkitekturen på FPGA-systemet.

Lyddata lagres binært til fil på en ekstern mottaks-PC ved hjelp av et program vi har laget. Dette for å kunne visualisere lyddataene. Programmet ble skrevet i java.

Alle valg av komponenter og tekniske løsninger er valgt på bakgrunn av kravspesifikasjonen [34].



Figur 3: Arkitektur FPGA

2 Moduler AD-kort

Når det gjelder kretskortdesign, så er det lurt å dele kortet opp i forskjellige moduler ut i fra blant annet funksjonalitet og hastigheter. Dette for best mulig å hindre forstyrrelser fra en komponent til en annen [1]. Dette kapittelet tar for seg modul for modul, og beskriver hvordan komponentene er satt opp i forhold til kravspesifikasjonen og hvordan de er integrert i forhold til resten av kretsen. Først kommer litt om komponentene vi bruker.

2.1 Komponenter

Valgte hardwarekomponenter er listet i Tabell 1. Alle komponenter er valgt på bakgrunn av kravspesifikasjonen [34].

Tabell 1: Hovedkomponenter på AD-kortet

Komponent	Merke
Mikrofon	Josephsson C550H [23]
Forforsterker	Texas Instruments THS4521 [4]
ADC	Texas Instruments PCM4204 [3]
Krystall Oscillator	IQD CFPS-39 [35]

Mikrofonen ble valgt ut fordi oppdragsgiver hadde slike mikrofoner liggende som vi kunne få til vårt system. Forforsterkeren ble valgt ut fordi den kan forsynes med kun positiv spenning, 5 VDC, som forenklet AD-kortet noe [4]. ADC-en ble valgt på grunn av at den har fire kanaler i en pakke, og har gode spesifikasjoner. Krystall oscillator ble valgt ut i fra ADC-ens krav til systemklokke [3].

Til vårt system finnes det to aktuelle komponenttyper: Overflatemonterte (SM) og gjennomgående hull (TH). Det er fordeler og ulemper ved begge. De overflatemonterte er bedre når det kommer til EMC (Electromagnetic compatibility) for de fleste komponenter, men kan være vanskeligere å montere på kortet. De kan også være litt dyrere. TH er derimot dårligere når det kommer til EMC, men enklere å montere på kortet. Det finnes flere teknikker for å redusere EMC-problemet for TH, men selv når disse er tatt i bruk vil SM føre til et mye mindre EMC-problem [1]. I samråd med oppdragsgiver bestemte vi oss for følgende: SM 0603 størrelse på passive komponenter. Store elektrolyttkondensatorer og spoler finnes ikke i 0603, så vi har gått opp i størrelse på de komponenetene. Vi har brukt TH type konnektorer, i og med at de er mer robuste med tanke på mekanisk slitasje [1].

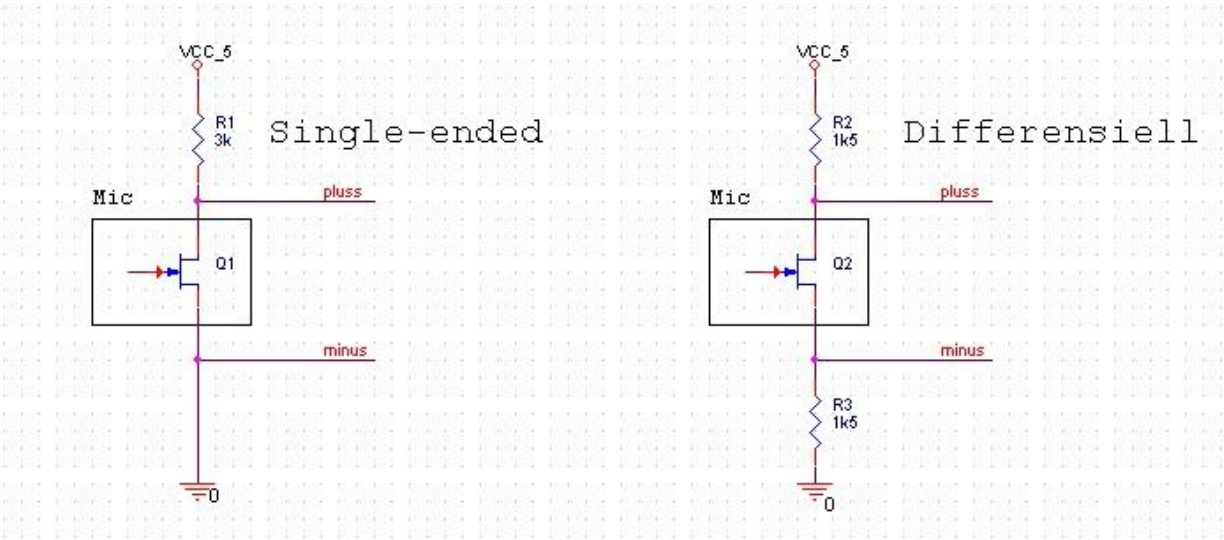
2.2 Mikrofon-grensesnitt

Josephsson C550H har balanserte utganger og en JFET (Junction Field Effect Transistor) transistor inne i kapsulen. Josephsson anbefaler et biasnettverk med 5 VDC gjennom en 3 kΩ til drain. Dette gjelder single ended konfigurasjon. Ved differensiell konfigurasjon trenger vi en 1,5 kΩ til drain, og en 1,5 kΩ til source, se Figur 4. Denne biasspenningen blokkeres ved hjelp av kondensatorer på forforsterkerinngangen for å ikke få DC-offset.

C550H har en sensitivitet på 10 mV/Pa. Den er designet til å fungere normalt opp til 135 dB SPL (Sound Pressure Level). SPL er definert slik at 134 dB tilsvarer 100 Pa. 134 dB lydtrykk på mikrofonen gir ut en spenning på 1 V_{rms}, det tilsvarer $\sqrt{2}$ V_p. Vi har satt 135 dB som øvre grense for arbeidsområdet til systemet. Arbeidsområdet kan endres ved å justere forsterkningen på forforsterkeren.

Josephssonmikrofonene er i utgangspunktet kalibrert fra fabrikken, men vi har klippet vekk den delen av mikrofonen. Derfor er det forskjeller på mikrofonene, i form av forskjellig utslag ved samme lydtrykk. I tillegg har tre av åtte mikrofoner negativ leder koblet til chassi. Ved bruk av differensiell konfigurasjon kan dette bli et problem i og med at chassi på hardware ofte er koblet til jord. Derfor har vi etter anbefaling fra oppdragsgiver valgt å bruke single ended konfigurasjon.

Vi har satt inn ekstra footprints som gjør det mulig å gå over til differensiell konfigurasjon ved behov.



Figur 4: Mikrofonkonfigurasjoner

2.3 Overspenningsvern

Overspenningsvernet skal sørge for at inngangssignalet ikke overstiger det pinnene på forforsterkeren tåler. Vi har sett på dioder som kan klippe signalet ved ønsket verdi.

Når det gjelder dioder har vi i hovedsak sett på tre forskjellige typer:

- Zenerdioder
- Schottkydioder
- Transient Suppression Voltage (TVS) dioder.

TVS-dioder er i praksis zenerdioder som er designet for å beskytte mot overspenninger. Disse diodene er veldig gode mot transienter, det vil si plutselige endringer i spenningen. TVS-dioder og zenerdioder kan ha mye lekkasjestrøm, noe som gjør dem lite attraktive i vårt tilfelle. I tillegg har disse diodene høyere spenningsfall enn schottkydioder. Generelt spenningsfall (V_F) for zenerdioder ligger rundt 0,7 V, det ligger rundt 0,3 V for schottkydioder.

Forforsterkeren i systemet vårt, THS4521, har en maksimum inngangs-spenning på[4]:

$$V_{inn\pm.} = (V_{S-.}) - 0{,}7 \;\; til \;\; (V_{S+.}) + 0{,}7 \tag{1}$$

V_{S-} er 0 V, V_{S+} er 5 V [4]. Det resulterer i at V_{INN} går fra -0,7 V til 5,7 V, altså ± 3,2 V_p. Vi begrenser inngangsspenningen til ± 2,9 V_P for p ha en sikkerhetsmargin. For å få til dette må spenningsfallet over dioden være på ca. 0,3 V.

På bakgrunn av forforsterkerens egenskaper har vi valgt en schottky-diode som har ca. V_F = 0,3 V ved ca. 1 mA. Vi funnet en som passer vår bruk: SD103AW.

For at diodekretsen skal klippe likt på begge sider av signalet må det kobles to dioder til hver inngangspinne. Disse diodene kobles også til V_{cc} (5 V) og V_{ee} (0 V), utifra disse biasspenningene kan vi bestemme hvor diodene skal klippe[22]. Schottkydiodene vil klippe ved:

$$\begin{aligned} +V_{klipp} &= +V_{BIAS} + V_F \\ &\quad og \\ -V_{klipp} &= -V_{BIAS} - V_F \end{aligned} \tag{2}$$

I Vårt system forsynes mikrofonene med 5 VDC, maksimum spenning ut fra mikrofonene er da 5 VDC. Dette betyr at overspenningsvernet slik vi har sett det for oss er unødvendig og det er ikke tatt med i systemet.

Overspenningsvernet ville fortsatt hatt en funksjon som ESD-beskyttelse (Electrostatic discharge), men siden dette er en prototype er dette ikke prioritert. ESD i form av elektriske støt kan være et potensielt problem. Håndteringen av AD-kortet kan forebygge risikoen for ødeleggende støt. Vi har brukt et bånd som festes i pulten, håndleddet og et jordingspunkt når AD-kortet håndteres.

2.4 Forforsterker

Vi bruker åtte THS4521 pakker. Forforsterkerene krever en spenningsforsyning på 5 VDC [4]. Spenningene på utgangspinnene vil da ligge mellom 0-5 V.

ADC-en krever en referansespenning på 2,5 VDC på inngangene [3]. En mulighet har vært å hente denne referansespenningen fra V_{OCM}-utgangen på ADC-en. Vi bruker en spenningsforsyning på 5

VDC til forforsterkeren, da settes V_{OCM} til 2,5 V ved hjelp av en spenningsdeler inne i forforsterkeren [21]. I følge tester Texas Instruments har gjort er det ingen forskjell på disse to løsningene. Vi har koblet V_{OCM} via en 0,22 µF kondensator til jord. Denne kondensatoren trengs for å fjerne høyfrekvent støy [4].

Det er en motstand på 40,2 Ω på hver utgang fra forforsterkeren. ADC-en representerer en ustabil last for forforsterkeren. Denne 40,2 Ω motstanden vil fungere som isolasjon mellom forforsterker og ADC, og vil hindre faseforskyvninger i forforsterkerkretsen [2].

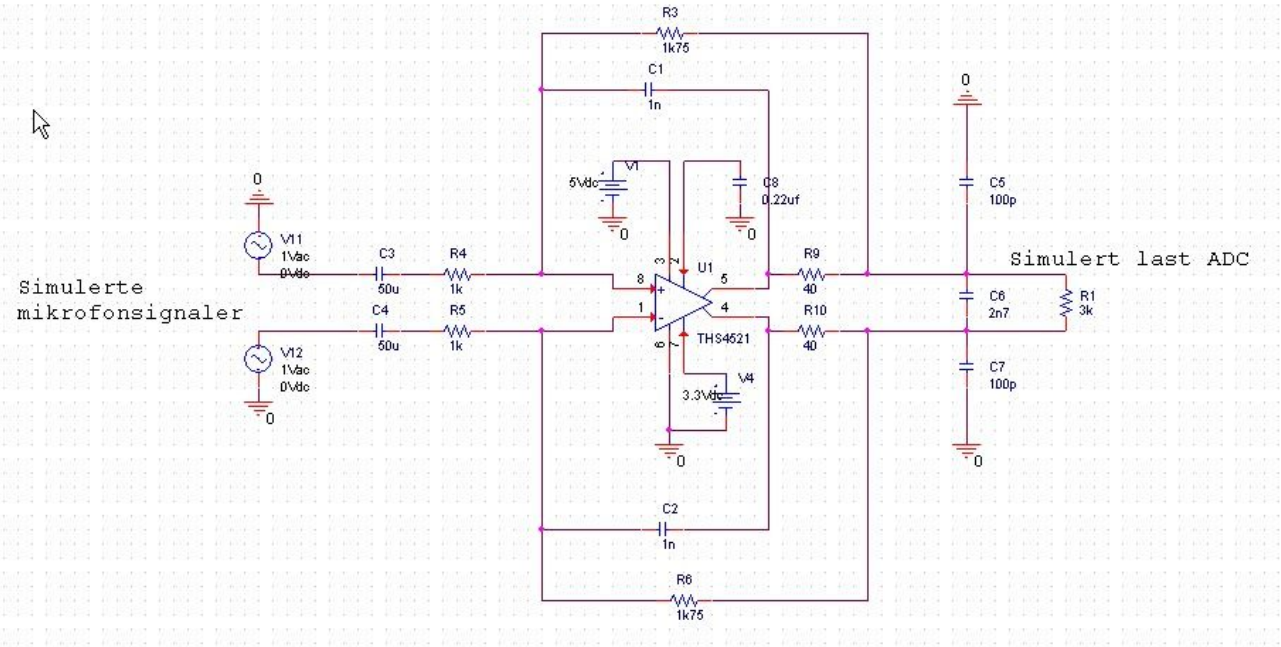
Vi har gjort beregninger på forforsterkeren i både differensiell og single ended konfigurasjon.

2.4.1 Differensiell konfigurasjon

Vi har tidligere regnet ut at et lydtrykk på 134 dB SPL vil gi et signalsving på √2 V_p. Maksimum signalsving inn på forforsterkeren er 2,5 V_p. En forsterkningsfaktor på 1,75 $V_{Put/VPinn}$ gir ønsket arbeidsområde: √2 V · 1,75 ≈ 2,47 V_{Put}. Signalforsterkning bestemmes av to motstander. Formelen for beregning av gain er:

$$G = \frac{R_f}{R_g} \tag{3}$$

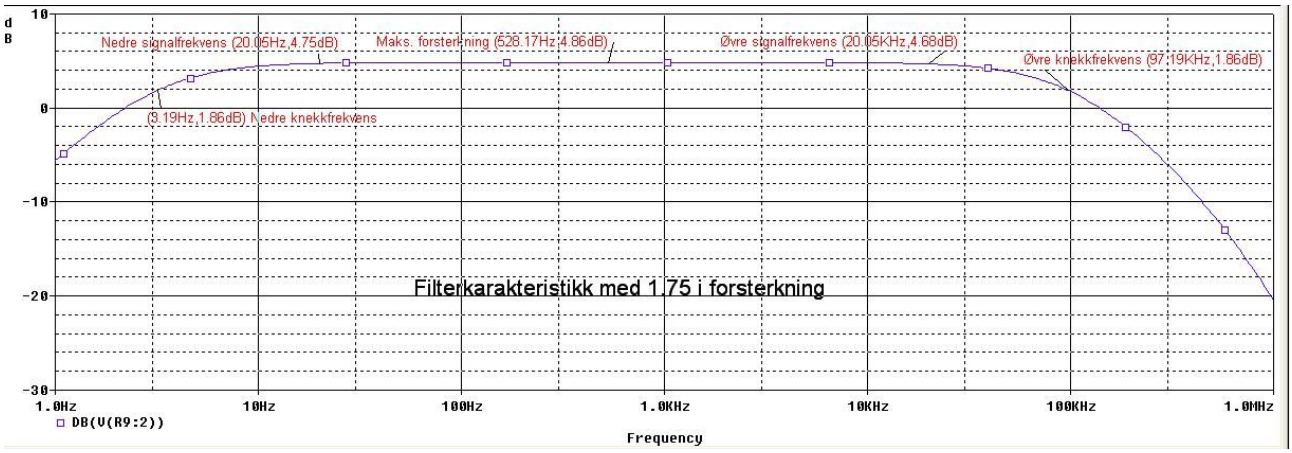
Det vil si at hvis R_G har en verdi på 1 kΩ, så må R_F ha en verdi på 1750 Ω.



Figur 5: Skjemategning av forforsterker konfigurert for differensiell inngang.

Forforsterkerkretsen inneholder et båndpassfilter. Før vi kan beregne nedre og øvre knekkfrekvens må vi avklare hvor stor signalforsterkning vi skal ha. Nedre knekkfrekvens bestemmes av høypassfilteret, C3 i serie med R4. Øvre knekkfrekvens bestemmes av lavpassfilteret C1 i parallell med R3, se Figur 5. Dette gjelder både positiv og negativ linje. Formelen for å beregne knekkfrekvens er:

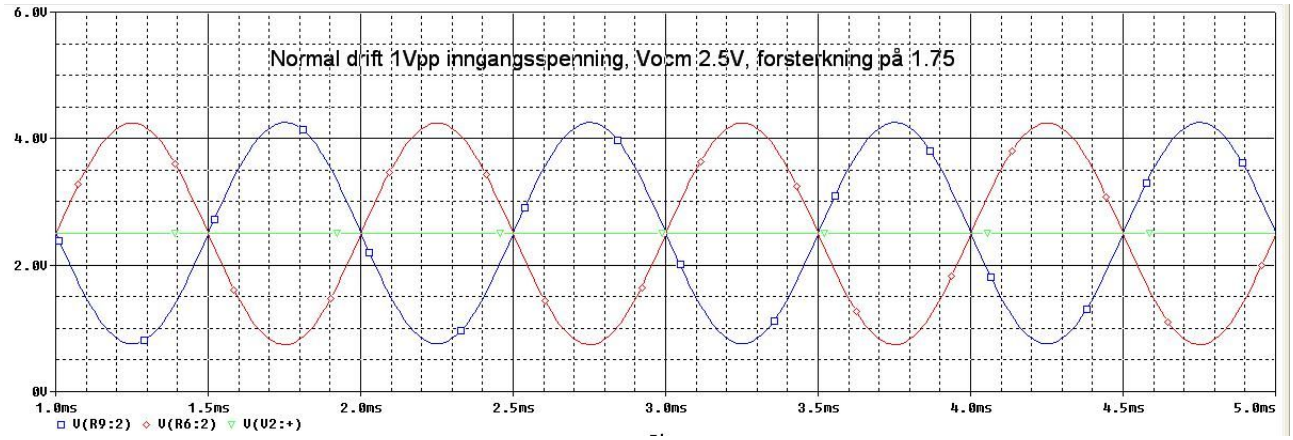
$$f_c = \frac{1}{2\Pi RC} \tag{4}$$



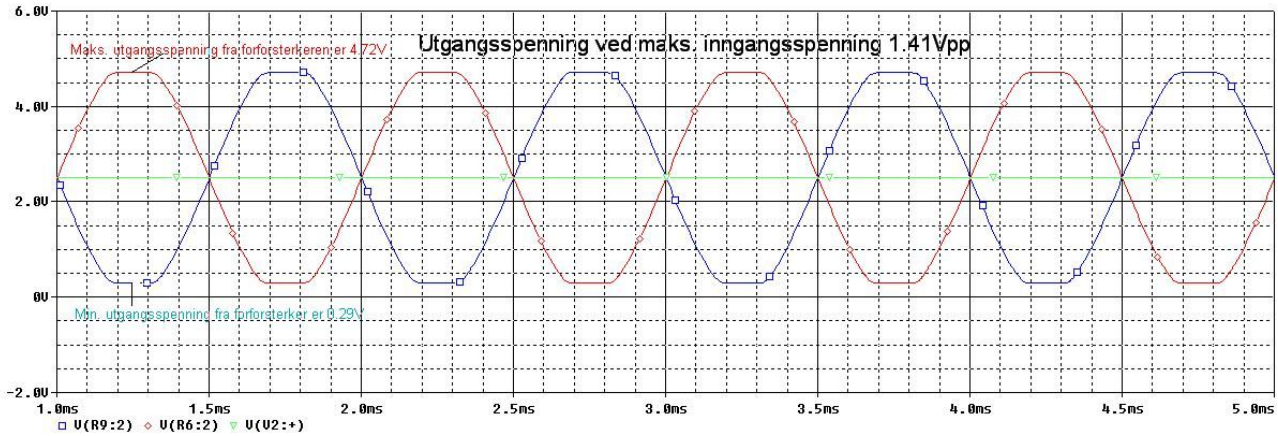
Figur 6: Filterkarakteristikk for båndpassfilter med 1,75 V_{Put}/V_{Pinn} i forsterkning.

Simuleringen i Figur 6 viser at forsterkningen er tilnærmet lineær i området 20 Hz-20 kHz, dette er frekvensområdet vi er interesserte i å forsterke.

Forventet amplitude på utgangssignal med 1,75 V_{Put}/V_{Pinn} i forsterkning og 1 V_P inngangssignal er 1,75 $V_{Put.}$ 2,5 V + 1,75 = 4,25 V, dette stemmer overens med det vi ser i Figur 7.



Figur 7: Utgangssignal ved differensiellt inngangssignal på 1 Vp og 1,75 V_{Put}/V_{Pinn} i forsterkning.

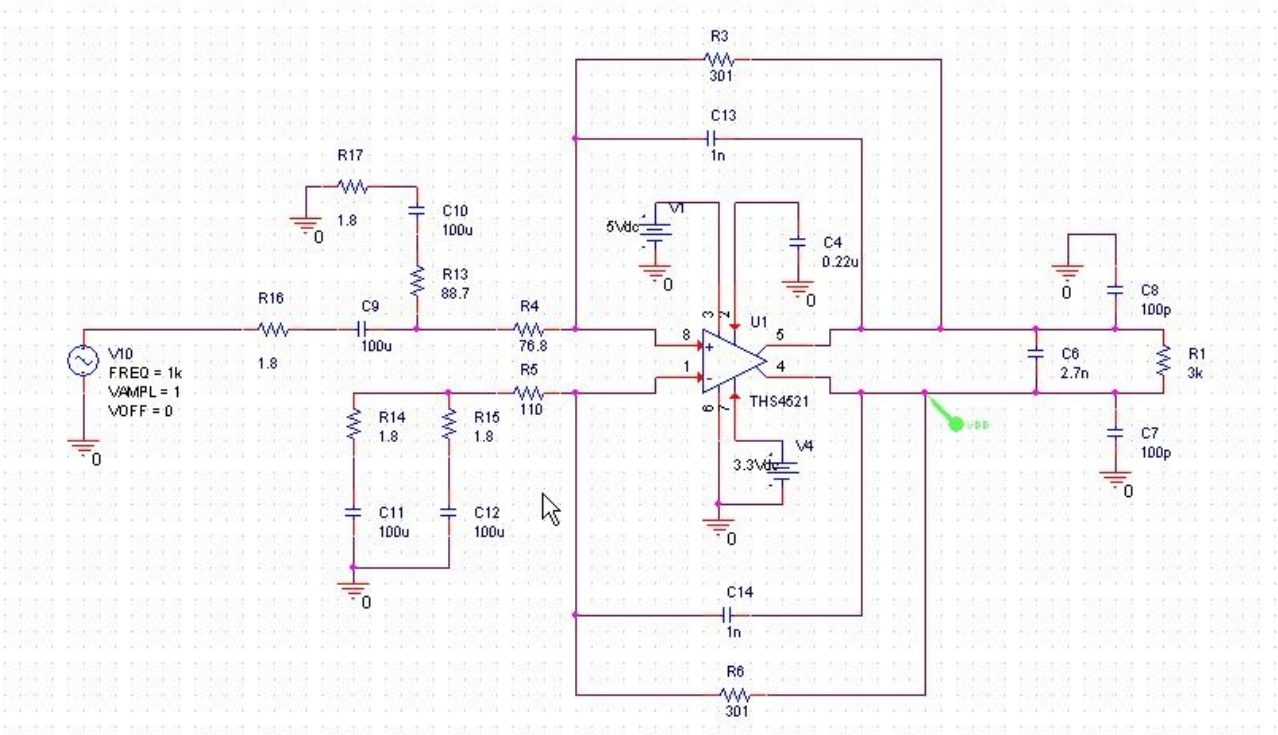


Figur 8: Utgangssignal med 1,75 V_{Put}/V_{Pinn} i forsterkning og inngangssignal på 1,41 V_P

I Figur 8 ser vi at forforsterkeren går i metning ved 4,72 V. Dette er fordi forforsterkeren ikke er ideell, og kan derfor ikke gi helt opp til 5 V på utgangen. Dette er normalt og som forventet. Det vil heller ikke ha noen store konsekvenser for dette gjelder bare de høyeste lydene. For flere detaljer rundt simuleringene se [49].

2.4.2 Single ended konfigurasjon

Ved single ended konfigurasjon må vi gjøre en impedanstilpasning for å få et stabilt utgangssignal som svinger rundt 2,5 VDC.

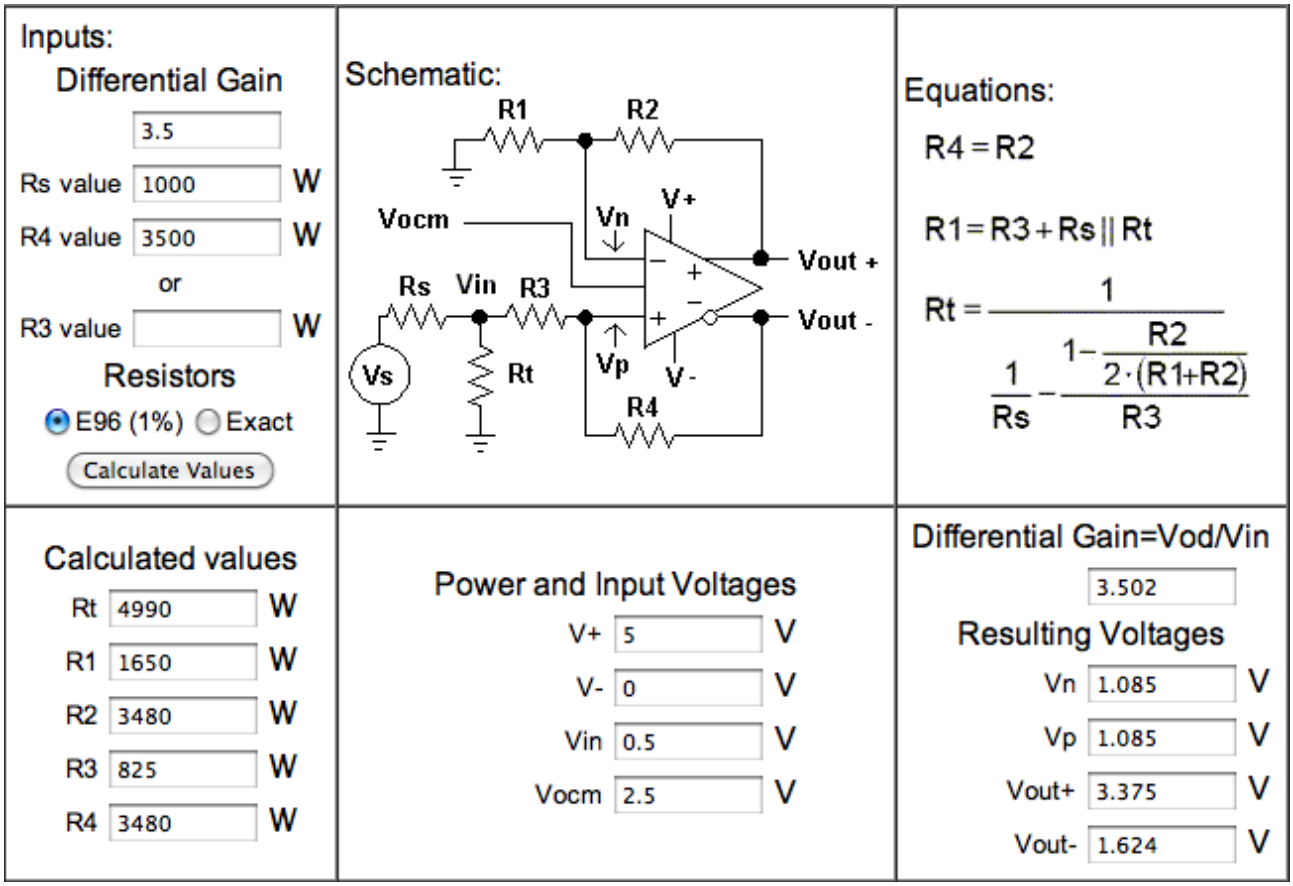


Figur 9: Skjemategning for forforsterkerkretsen med single ended konfigurasjon.

Figur 9 viser kretsen bygget med tanke på impedanstilpasning. Kondensatorene på 100 µF brukes for å blokkere biasspenningen fra mikrofonene. Dette er viktig for å hindre forskyvning av V_{OCM}. Motstandsverdiene er beregnet ut i fra seks likninger [11]. Texas Instuments har laget en online kalkulator der vi kan regne ut disse komponentverdiene [19]. Se Figur 10 for verdiene vi bruker.

Når vi bruker single ended konfigurasjon må vi doble den differensielle forsterkningen. Dette fordi forforsterkeren forsterker differensen mellom de to kanalene. Når vi nå setter den ene kanalen til 0, så har vi halvert differensen mellom de to kanalene. Når vi ønsker en total forsterkning på 1,75 V_{Put}/V_{Pinn} må vi sette den differensielle forsterkningen til 2 · 1,75 = 3,5. Vi synes ordbruken differensiell forsterkningen kan være litt misvisende, men Texas Instruments ordlegger seg slik. Dette henspiller på at vi omgjør single ended inngangssignal om til differensielle signaler på utgangen.

Vi bruker standardverdier på komponentene (E96). Dette gjør at forsterkningen blir 3,502 i stedet for 3,5. Dette er så liten forskjell at det ikke har betydning for vår del.

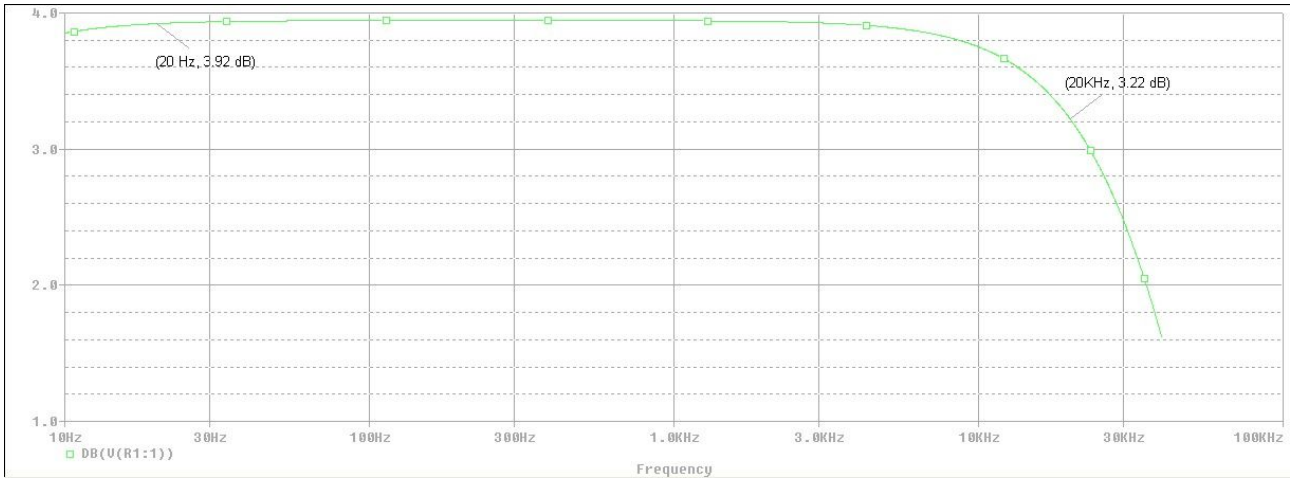


Figur 10: Online impedanskalkulator fra Texas Instruments

Vi har satt Rs til 1 kΩ fordi anbefalt mikrofonlast er 1 kΩ [23]. JFET-forsterkningen inne i mikrofonkapsulen er proporsjonal med lasten mikrofonen ser [22]. Med disse motstandsverdiene får vi en litt høyere spenning ut fra mikrofonene enn forventet. Ved 1 kHz 114 dB lydsignal inn forventer vi 141 mV_P ut fra mikrofonene, vi får ca. 180 mV_P ut. Årsaken til dette er mest sannsynlig at vi har klippet vekk en del av mikrofonen. I samråd med oppdragsgiver har vi bestemt at dette er et akseptabelt avvik. Avviket i forventet spenning og faktisk spenning fører til at systemet vil få en

mindre rekkevidde med tanke på lydtrykk. I utgangspunktet ville ikke systemet gått i metning før ved 134 dB, nå går systemet i metning ved ca. 128 dB [38].

100 µF kondensatorene i serie med motstandene utgjør et høypassfilter, se Figur 9. Sammen med kondensatorene på 1 nF i parallell med tilbakekoblingsmotstanden utgjør dette et båndpassfilter. Figur 12 viser simulering av filterkarakteristikken til båndpassfilteret. Vi har kun tatt med karakteristikken fra 20 Hz - 20 kHz. For audiobruk vil man ha flat frekvensrespons fra 20 Hz til 20 kHz. Som vi ser er faller responsen litt av mellom 10 kHz og 20 kHz. Dette kan bedres ved å minske verdien på tilbakekoblingskondensatorene, fra 1 nF til 0,1 nF for eksempel. Men i og med at Snipos filtrerer bort disse frekvensene, har vi ikke tatt oss tid til å bytte disse kondensatorene.



Figur 11: Frekvenskarakteristikk forforsterker

Figur 12: Utgangssignal med single ended konfigurasjon og 1 V_p inngangssignal. 3,5 i differensiell forsterkning.

Figur 15 viser signalsvinget på de to utgangene fra forforsterkeren med en differensiell forsterkning satt til 3,5. Vi ser at utgangene er balanserte, det vil si lik amplitude og 180° faseforskjøvet. Vi ser

også at vi får differensiell utgang som svinger rundt 2,5 V selv om inngangen er single ended.

2.5 Klokke

Høyfrekvente oscillatorer er en av de største kildene til problemer med tanke på EMC. Derfor er integreringen av denne modulen veldig viktig. En tommelfingerregel er å bruke lavest mulig frekvens på oscillatoren [1]. I henhold til kravspesifikasjonen [34] og databladet til ADC-en [3], så er lavest mulig frekvens på oscillatoren 24,576 MHz. Vi har valgt en krystalloscillator (XO) med denne frekvensen. Når det gjelder passive komponenter, så har vi brukt anbefalt verdi på avkoblingskondensatoren på spenningsinngangen [35].

Etter diskusjon med produsenten av oscillatoren har vi droppet buffer på utgangen til oscillatoren. Dette fordi vi kun splitter signalet i to, og lasten i enden av hver splitt er identisk (to like ADC-er).

Oscillatoren er kalibrert til å se en kapasitiv last på 15 pF [35]. For å sikre mest mulig nøyaktig klokkesignal bør den kapasitive lasten i kretsen være omtrent den samme. Vi har satt inn en footprint for å kunne tilpasse denne lasten. Vi har ikke opplevd problemer i forhold til dette, og har dermed ikke sett nærmere på tilpasning av kapasitiv last.

Firkantpulser har veldig mange overharmoniske komponenter. Alt for raske stigningstider på klokka kan føre til oversving og undersving. Dempemotstander rett etter utgangen på XO vil runde av firkantpulsene litt. Dette vil øke stigningstiden litt og det kan også være gunstig med tanke på EMC [1]. Vi har satt inn dempemotstander på 47 Ω rett etter utgangen på XO.

2.6 ADC

Vi bruker to stk PCM4204 AD-konvertere. For å kunne omforme data trenger ADC-en en systemklokke (SCKl) [3]. Denne hentes fra krystalloscillatoren.

Det mest hensiktsmessige oppsettet er å sette den ene ADC-en til å være master og den andre til å være slave. Begge får systemklokke fra XO. Vi sender data videre over en I^2S-buss. I^2S-bussen består av to klokkesignaler og et datasignal. Datasignalet hentes fra SDOUT-pinnene. Datasignalet består av to kanaler, left og right, som sendes serielt. Disse navene, left og right, henspiller på stereo-oppsett som er vanlig i audioverdenen. De to klokkesignalene kalles bitklokke (BCK) og leftright-klokke (LRCK). Master-ADC sender BCK og LRCK videre til både I^2S-bussen og ADC-en som er slave. Med åtte kanaler har vi da totalt seks linjer ut fra AD-kortet. BCK, LRCK, SDOUT1 og SDOUT2 fra både master og slave [3]. FPGA-kortet har en egen inngang for klokkesignaler. Vi ble anbefalt å bruke denne. Derfor sender vi BCK på en SMA-kabel mellom AD-kort og FPGA-kortet.

ADC-ene oversampler med en faktor på 256 [3]. For å opprettholde en stabil klokke når vi dividerer ned klokken, må den divideres ned med et heltall. Det vil si at SCKl blir dividert ned med en faktor på fire, slik at BCK gir 32 bit per sample istedenfor 24. De åtte minst signifikante bit-ene blir da behandlet som «don't care» bit. Dette må det tas høyde for i mottaket på FPGA-kortet. Frekvensen

på klokkene ut av ADC blir da: BCK = 6,144 MHz og LRCK = 96 Khz [3].

Tabell 2: Tabell for å sette pinnene på ADC til riktig datahastighet[3].

Pinne	Master	Slave
FS0	1	1
FS1	0	0
FS2	1	0
S/M	0	1
FMT0	1	1
FMT1	0	0
FMT2	0	0

I følge databladet til ADC-en må pinnene settes som i Tabell 2 for å oppnå riktig datahastighet (96 kHz, I^2S) [3]. Høye pinner (1), er koblet til 3,3 VDC, og lave pinner (0) er koblet til jord.

Vi har en del pinner som ikke er i bruk på ADC-pakken. Vi har brukt følgende prinsipp for disse: Innganger kobles til jord, utganger lar vi være utilkoblet. Utganger er loddet fast til PCB pad-en for å sikre mest mulig stabilitet på kortet.

Clip: Vi har ikke bruk for denne funksjonaliteten. De er utganger, derfor forblir de utilkoblet.

Sub: Denne pinnen brukes til TDM (time division multiplexed) modus. Vi bruker ikke TDM-modus. I følge databladet kan vi sette denne til logisk lav [3].

Test: Denne pinnen skal legges til jord [3].

RST: Denne brukes til å resette ADC-en, for eksempel i forbindelse med forandring av parametre. Vi skal kjøre med faste parametre hele tiden, derfor er denne pinnen satt til høy (aktiv lav). Det står heller ikke noe om rekkefølge på oppstartssekvens i databladet. ADC-en vil tvinge RST til lav under oppstart og venter på stabile spenninger og klokkesignal før den begynner å konvertere [3].

HPFD: ADC-en har innebygd et høypassfilter for å fjerne DC-spenninger. Knekkfrekvensen er gitt ved f_s / 48k = 96 kHz / 48k = 2 Hz [3]. Vi har satt dette filteret aktivt.

DSD: Vi bruker ikke denne funksjonaliteten. Det har vært litt vanskelig å finne ut av hvordan disse skal behandles. Det står ikke forklart i databladet hva vi skal gjøre med de i vår situasjon. Vi har hatt kontakt med Texas Instruments, og de har sagt at vi kan koble DSD-pinnene rett til jord.

I og med at maksimum signalsving ut fra forforsterkeren ligger innenfor de absolutte grenseverdiene til ADC-en (0-5 V) trengte vi ikke ekstra beskyttelse på inngangene til ADC-en [3].

Når det gjelder passive komponenter i forbindelse med ADC-en, så har vi fulgt anbefalingene som står listet opp i databladet. De passive komponentene som er brukt er avkoblingskondensatorer. Avkoblingskondensatorer brukes til å filtrere vekk støy slik at støyen ikke forplanter seg videre i kretsen [3].

I databladet bruker de en 100 Ω motstand på klokke-innganger/utganger og digitale

datautganger [3]. Dette for å dempe signalene litt med tanke på EMC. Vi satt i utgangspunktet disse inn i kretsen, men etter tester viste det seg at vi måtte fjerne de [38].

Vi har satt en kondensator på 100 pF koblet til jord så tett som mulig på de analoge inngangene. Disse fungerer som en «batteripakke», og vil forhindre spenningsfall på utgangene fra forforsterkeren i perioder med stort strømtrekk fra ADC-en [2].

For å sikre god håndtering av overskuddsvarme, har Texas Instruments utviklet en standard kalt «PowerPad Package». Dette er ikke en egen komponent, men en teknikk som utføres i selve kretskortutlegget. Denne teknikken går ut på å lage et område i kobber rett under pakken, kalt «thermal pad». Dette området må være litt mindre enn pakka, slik at det ikke er i veien for pinnene. For vår ADC-pakke anbefales en størrelse på 8x8 mm. På undersiden av pakken er det også et tilsvarende metallområde. I tillegg til å lodde på pinnene, så loddes også disse to metallområdene sammen. På «thermal pad» skal det legges inn 3x3 via med en størrelse på maksimum 13 mil (milli-inch) (0,33 mm), alle ni viaene kobles til jordplanet. Det vil si at overskuddsvarme ledes fra pakken og ned til jordplanet. For å sikre lav termisk resistans er det viktig med heldekkende tilkobling (ikke thermal relief) rundt viaene på jordplanet. På det underste kretskortlaget lages det en tilsvarende «thermal pad», men den loddes ikke til noe [3]. Vi har fulgt denne oppskriften i designet vårt.

2.7 Grensesnitt mot FPGA og spenningsforsyning

De komponentene vi har valgt trenger +5 VDC og +3,3 VDC. Oppdragsgiver har bestemt at vi skal hente spenning fra FPGA-kortet, selv om dette ikke er en optimal løsning med tanke på støy.

I og med at det er en begrensning på strømtrekk fra spenningsforsyningen på FPGA-kortet, så har vi gjort et lite overslag over hva komponentene bruker av strøm, se Tabell 7 og Tabell 8.

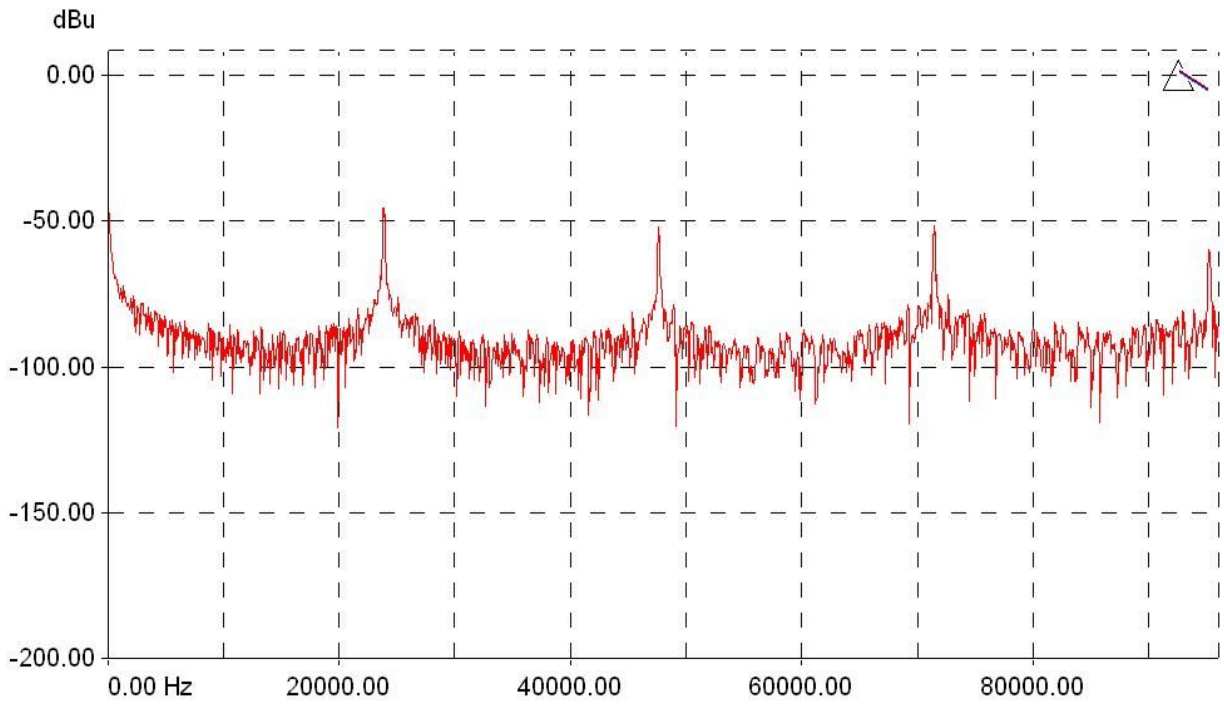
Tabell 7: Beregning av strømtrekk for analoge kretser

Type	Antall	Maks strømtrekk (5 VDC, maks 1 A)
PCM4204	2	130 · 2 = 260 mA (max quiescent current) [3]
THS4521	8	1,25 · 8 = 10 mA (quiescent operating current [4]
C550H	8	5 · 8 = 40 mA (power requirement) [23]
		Totalt: 310 mA

Tabell 8: Beregning av strømtrekket til digitale kretser

Type	Antall	Maks strømtrekk (3,3 VDC, maks 1,5 A)
PCM4204	2	44 · 2 = 88 mA (max quiescent current) [3]
XO	1	12 mA (supply current) [35]
		Totalt: 100 mA

Vi har valgt å filtrere spenningsforsyningen fra FPGA-kortet ved å bruke et passivt filter. For å kunne si noe om hvilke frekvensområde som må filtreres har vi gjort en støymåling på 5 V-linjen [39]. Figur 14 viser støyen på 5 V-linjen da vi kjørte et program på FPGA-en. Vi ser at den første kritiske støyen kommer ved 22 kHz i frekvens. Filteret har vi laget med tanke på størst mulig støydemping ved 22 kHz.



Figur 13: Støymåling av spenningsforsyning fra FPGA

For å kunne ta en avgjørelse på hva slags filtreringsmetode vi vil ta i bruk, må vi se på egenskapene til komponentene som skal bruke spenningsforsyningen.

ADC-en kan opereres med spenninger fra 4,75 V til 5,25 V på 5 V-linjen, og fra 3,0 V til 3,6 V på 3,3 V-linjen [3]. Forforsterkeren kan opereres med enda mindre spenninger [4], så vi har tatt utgangspunkt i et maksimalt tillatt spenningsfall på 0,25 V.

For å beregne totallasten på 5 V-linjen bruker vi maksimum strømtrekk som referanse. Dette gjør vi fordi komponentene representerer minst motstand ved maksimum strømtrekk (U = R · I). Totalt forsynes to ADC-er, to forforsterkere og åtte mikrofoner av 5 V-linjen. Disse komponentene står i parallell. Av Tabell 7 og Tabell 8 kan vi lese at maksimalt strømtrekk fra komponentene er 310 mA på 5 V-linjen. Totallasten komponentene representerer blir da:

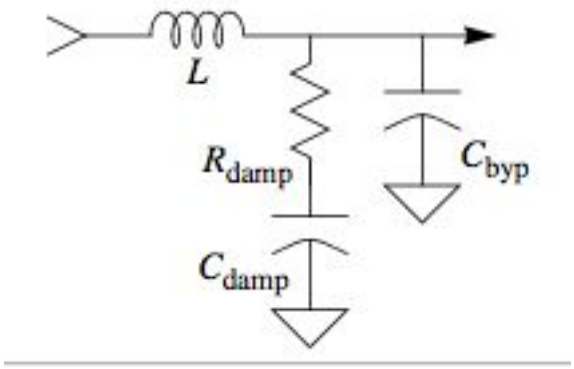
$$R = \frac{5\,V}{310\,mA} = 16{,}13\,\Omega \tag{5}$$

Maksimalt tillatte spenningsfall på 0,25 V betyr at DC-motstanden til filteret må være under 0,25 V / 5 V = 5 % av totallasten komponentene representerer. Maksimum DC-motstand i filteret blir da

$$R \leq 16{,}13\,\Omega \cdot 0{,}05 = 0{,}80\,\Omega \tag{6}$$

Denne forutsetningen gjør et RC-filter uegnet på grunn av at spenningsfallet over motstanden vil være for stort.

Et LC-filter kan designes med en DC-motstand som er innenfor toleransene. Ulempen med et LC-filter er at spolen vi bruker kan påvirke kretsen negativt. Den har en resonansfrekvens og vil forsteke signaler som ligger i dette området. For å dempe denne resonansen har vi brukt en dempningsmotstand og en dempningskondensator i parallell med en avkoblingskondensator [10].



Figur 14: Utgangspunkt for filterdesign.

Verdien til dempingsmotstanden og dempingskondensatoren beregnes ved å bruke tre likninger [10]:

$$C_{damp} > 16C_{byp} \tag{7}$$

$$R_{damp} > 2\sqrt{\left(\frac{L}{C_{damp}}\right)} \tag{8}$$

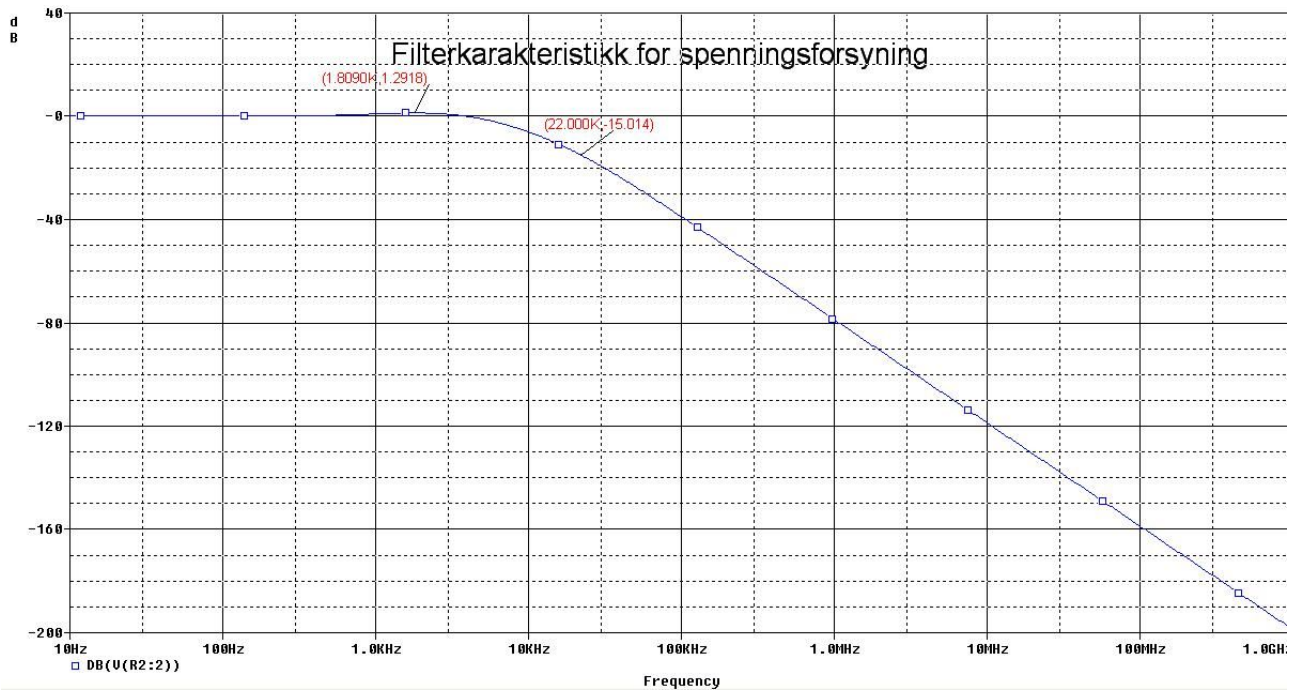
$$R_{damp} < \frac{1}{2}\sqrt{\left(\frac{L}{C_{byp}}\right)} \tag{9}$$

Ved å se på standardkomponentverdier og pris, samt passe på at DC-motstanden til spolen er under 0,80 Ω, har vi kommet fram til et design som passer til vårt formål: LC-filter med L = 47 µH, C_{byp} = 4,7 µF, C_{damp} = 100 µF og R_{damp} = 1,5 Ω.

Vi ser av Figur 16 at vi fortsatt har en liten forsterkning ved frekvenser rundt 1,8 kHz. Maksimum forsterkning er ved 1,8 kHz og er da 1,29 dB. Dette har vi vurdert til å være lavt nok, det vil ikke ha for store negative konsekvenser for vårt system. Ellers ser vi av Figur 16 at vi har klart å oppnå en støydempning på -15 dB ved 22 kHz.

Støyfiltrering på 3,3 V-linjen er ikke like kritisk som for 5 V-linja. Grunnen til det er at 3,3 V kun

brukes av digitale komponenter som tåler støy mye bedre enn analoge. Vi har ikke målt støynivået på 3,3 V-linjen. I kretskortdesignet har vi lagt opp til å bruke det samme filteret på 3,3 V-linjen som vi bruker på 5 V-linjen.



Figur 15: Filterkarakteristikk for spenningsforsyningsfilter.

Reserveløsning spenningsforsyning:

Vi har vært fullt klar over at vi kan få problemer med å oppnå godt signal-støy forhold ved å hente spenning fra FPGA. Derfor valgte vi å legge inn en reserveløsning. Vi har lagt til tre skruekonnektorer der vi kan koble til eksterne spenningsforsyninger, to 5 VDC og en 3,3 VDC.

Vi har også valgt å legge inn jumpere for å lett kunne koble vekk spenningsforsyningen fra FPGA-kortet.

3 PCB-design

I vårt design har vi valgt å lage et firelags kort. Dette fordi det er anbefalt å ha minst ett jordplan og ett spenningsplan når vi bruker kompliserte komponenter som ADC-er. Dette vil gjøre kortet mer robust mot støy og EMC-problemer. Fordelingen av planene er to signalplan som topp- og bunnlag. Jord og spenningsplan ligger i midten med jordplanet nærmest topplaget. Jordplanet bør være mellom spenningsplanet og det planet med de mest sensitive kretsene, altså topplaget [1]. På grunn av valgt monteringsmetode har vi kun komponenter på topplaget. Det er mulig å ha komponenter på bunnplanet i tillegg, men det ville gjort monteringen betraktelig dyrere.

Man kan tenke seg å ha jord og spenningsplan ytterst for å skjerme kretsen bedre, men det vil da blitt vanskelig å feilsøke kretsen. Dessuten vil alle viaene redusere mye av denne skjermingseffekten. Se Figur 17 for oversikt over planene på kortet vårt. Dielectric FR-4 er standard isolasjonsmateriale mellom lagene. Tykkelsen på lagene er 1 oz kobber, som også er standard.

	Subclass Name	Type	Material
1		SURFACE	AIR
2	TOP	CONDUCTOR	COPPER
3		DIELECTRIC	FR-4
4	GND	PLANE	COPPER
5		DIELECTRIC	FR-4
6	PWR	PLANE	COPPER
7		DIELECTRIC	FR-4
8	BOTTOM	CONDUCTOR	COPPER
9		SURFACE	AIR

Figur 16: Oppdeling 4-lags kort

3.1 Jordplan

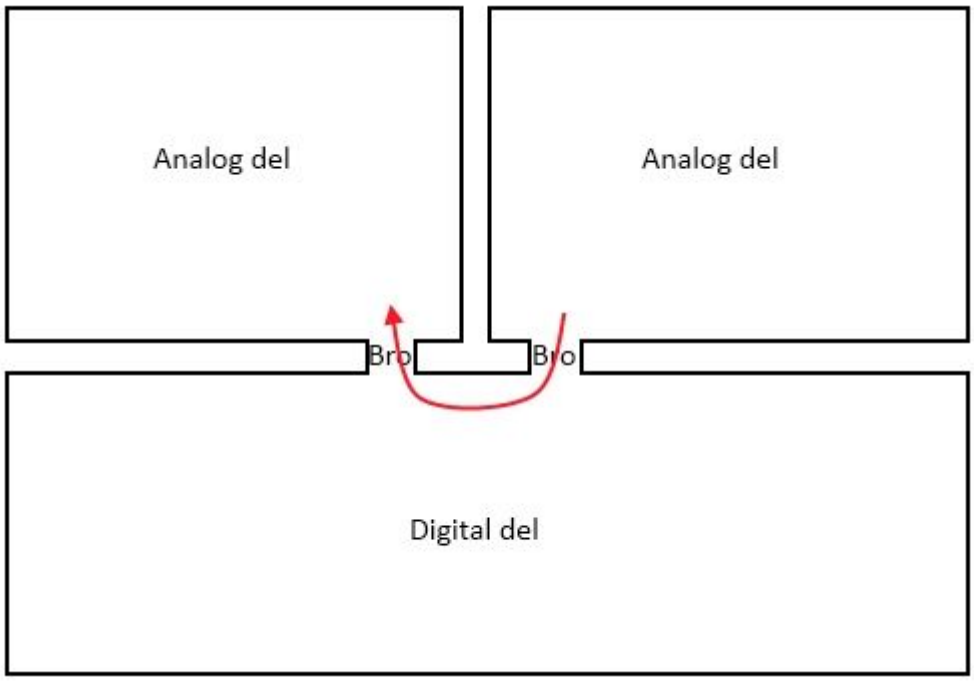
Det finnes mange forskjellige løsninger for hvordan man bør designe et jordplan. Når man har både analoge og digitale signaler blir dette enda mer vanskelig fordi ADC-en må ha felles jordreferanse [3]. Det vi har valgt å ta mest hensyn til er at vi bruker en høyoppløselig ADC. I vårt design har vi sett på tre forskjellige løsninger:

1: Dele opp jordplanet i to analoge deler og en digital del, Figur 18.

Her vil det bli et problem i forhold til konnektoren til FPGA-en. Vi henter 5 V og 3,3 V gjennom denne konnektoren. Konnektoren må plasseres på kanten av kretskortet og den må ligge over både en analog og en digital del. Det betyr at den ene analoge delen ikke vil være direkte koblet til konnektoren. Dette fører igjen til at returstrømmer må ta turen innom det digitale planet for å komme tilbake til konnektoren. Dette er ikke gunstig siden returstrømmer bør ha så kort og direkte bane som mulig [1].

Denne løsningen ville derimot vært veldig bra om spenningsforsyningen hadde sett annerledes ut.

Om vi hadde satt en spenningsregulator på hver av de analoge delene ville det ikke blitt noe problemer med returstrømmen.



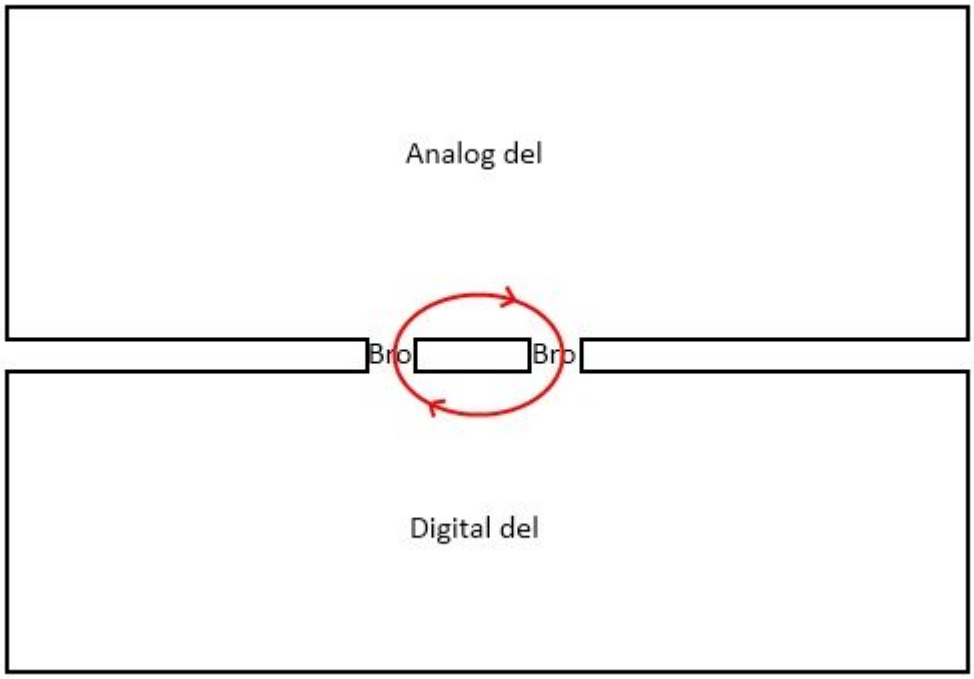
Figur 17: Jordplan med inndeling i en digital del og to analoge deler.

2: Dele opp jordplanet i en analog del og en digital del, Figur 19.

Problemet her blir at digitale strømmer kan ta turen innom den analoge delen. Dette er veldig ugunstig siden den analoge siden er veldig følsom for støy, og digitale strømmer er som regel ganske støyete. Denne støyen vil forplante seg gjennom hele signalkjeden og føre til lavere signal-støy forhold. I praksis vil denne løsningen være veldig lik løsning nr. 3, og løsning nr. 3 er en mer ryddig løsning.

3: Kun ha ett jordplan som ikke er delt opp.

Det er mange som mener at det å ikke dele opp jordplanet er den beste løsningen, spesielt for ADC-er med lav oppløsning. Siden ADC-ene som skal brukes har en høy oppløsning kan det å bruke denne løsningen føre til problemer når det gjelder støy [17].



Figur 18: Jordplan med inndeling i en analog og en digital del.

Vårt valg:

Det er tydelig at ingen av løsningene er helt optimale, men det er vanskelig å få til siden vi ikke har valgt en optimal løsning for spenningsforsyningen. Vi har valgt å gå for løsning nr. 1, siden denne hadde vært den beste med optimal spenningsforsyning. Med dette valget prioriterer vi å hindre digitale strømmer å komme inn på analog side, fremfor å ha korte og direkte returbaner på analog side.

Vi har i tillegg valgt å lage en mulighet for å bruke eksterne spenningsforsyninger. Vi har sammenlignet disse to mulighetene, og det er en viss forskjell på ytelsen til systemet ved de forskjellige løsningene [40].

ADC-ene ligger rett over broa, slik at de får felles jordreferanse.

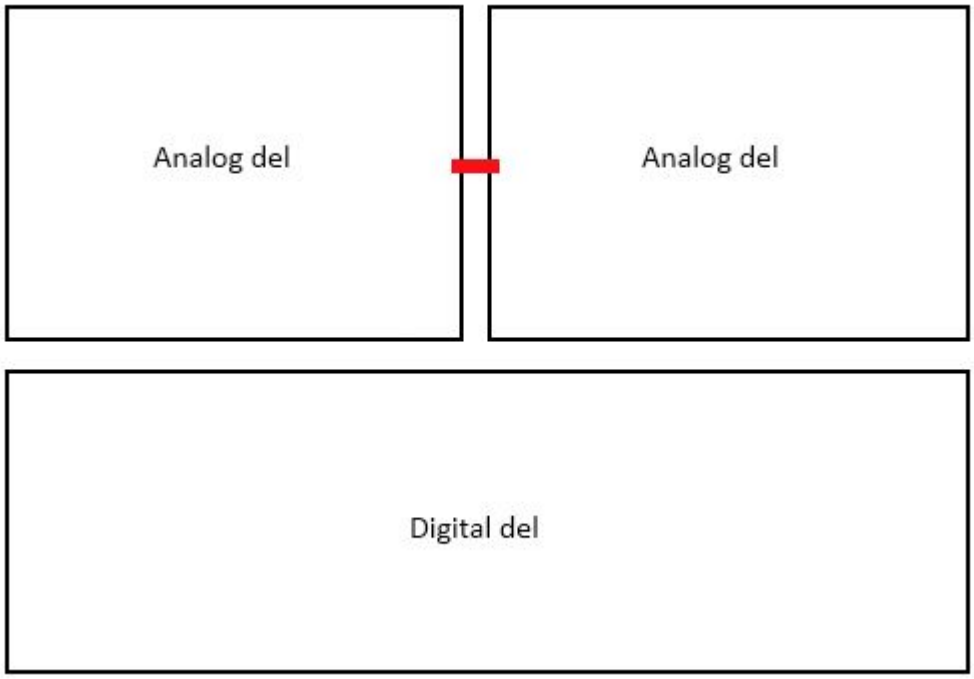
Når vi har valgt å splitte jordplanet er det viktig å sørge for at det ikke går noen signallinjer over selve splitten. Det er også viktig at sløyfene til strømmene har minst mulig areal. Dette vil også minske problemene ved at de analoge strømmene går over på digital del.

3.2 Spenningsplan

De analoge kretsene trenger 5 VDC, og de digitale kretsene trenger 3,3 VDC. Dette krever at spenningsplanet er oppdelt. Utfordringen har vært å få til 5 V-delen av planet. Når vi bruker forsyningen fra FPGA-kortet trenger vi et kontinuerlig 5 V-plan over begge de analoge seksjonene. Når vi bruker de eksterne forsyningene må 5 V-planet deles i to. Vi har valgt å bruke en

jumperløsning for å raskt kunne velge mellom disse to løsningene, se Figur 20. Impedansen blir relativt høy i tilkoblingspunktet, men strømmene i kretsen vår er såpass lave at går greit. Vi har også brukt jumpere for å kunne koble vekk spenningsforsyningen fra FPGA-en fordi 40-pins konnektoren må være tilkoblet til FPGA-kortet uansett. Bakdelen med disse jumperene er at de fungerer som antenner i høyfrekvensområdet [1]. Disse frekvensene blir filtrert bort, men de kan påvirke systemet på en negativ måte. Disse jumperløsningene egner seg kun for en prototype og må fjernes i nye versjoner av systemet.

Felles for både jord og spenningsplan er at vi må ha en klarering til kanten av kortet for å hindre kortslutning mellom planene på utsiden av kortet. Vi må også ha en klarering mellom de forskjellige seksjonene av planene. Vi har fulgt anbefalingen på 50 mils vi fant i [18].



Figur 19: Spenningsplan med jumper mellom analoge deler

3.3 Signalplan

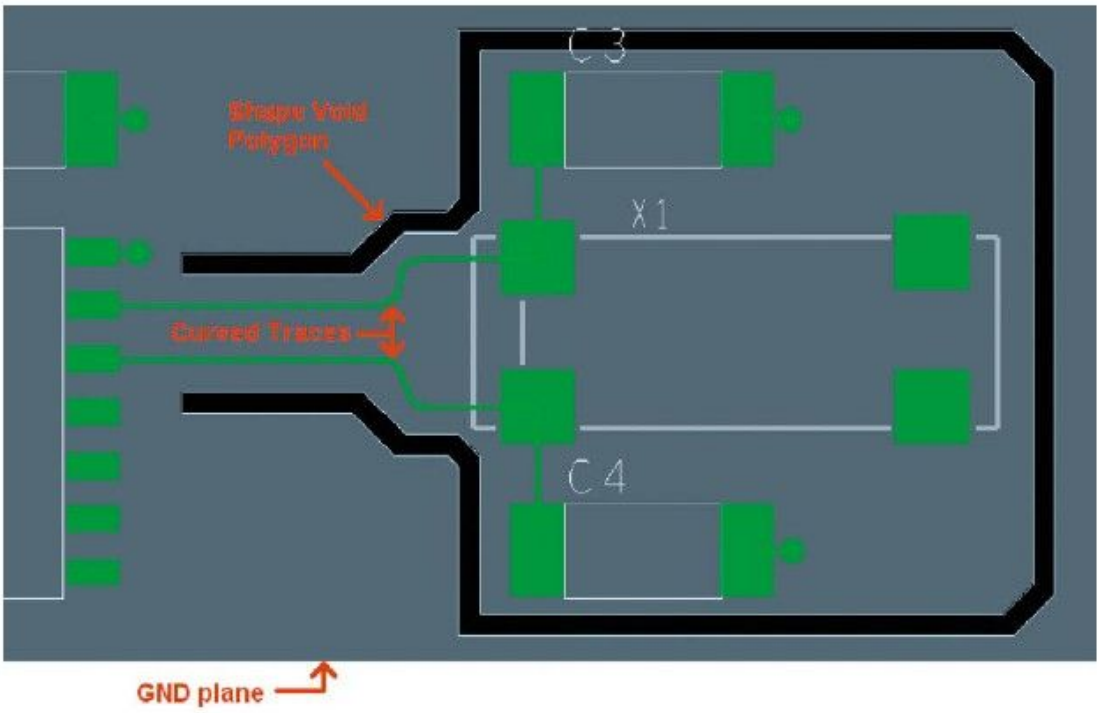
Som nevnt tidligere har vi to signalplan, topp og bunnlaget. Vi har kun komponenter på topplaget, signalbanene frem til ADC-en kun på topplaget. Dette gjør at det er lett å følge signalflyten med tanke på feilsøking.

For å sikre best mulig elektromagnetisk kompabilitet har vi lest en del om emnet og vi har valgt å følge en del anbefalinger. En generell anbefaling vi har fulgt er å lage kortet så kompakt som mulig med korte baner og små strømsløyfer.

Klokke:

Det er de høye frekvensene som ofte skaper mest trøbbel i et design med tanke på EMC. Den høyeste frekvensen i kretsen vår kommer fra systemklokken (24,576 MHz). Dette er ikke en veldig høy frekvens sammenlignet med GHz-kretser, men vi har likevel tatt hensyn til dette i utlegget vårt. Klokkekretsen ble lagt til et topplaget for å unngå via. Et via innfører 90 graders svinger i signalbanen, og det kan bli problemer med refleksjoner. Vi holder banene så korte som mulig, og vi lagde banene fra klokka til de to ADC-ene omtrent like lange [1].

Vi har sett anbefalinger om en «vollgrav» rundt klokkekretsen på jordplanet, se Figur 21. Vi har diskutert dette med intern veileder og kommet frem til at vi ikke lager dette. Dette ville kanskje vært viktigere hvis vi hadde hatt et jordplan som ikke var oppdelt. Det er vanskelig å si noe på forhånd om effekten av dette.



Figur 20: Vollgrav rundt klokkekrets, bilde hentet fra [18]

Banebredder:

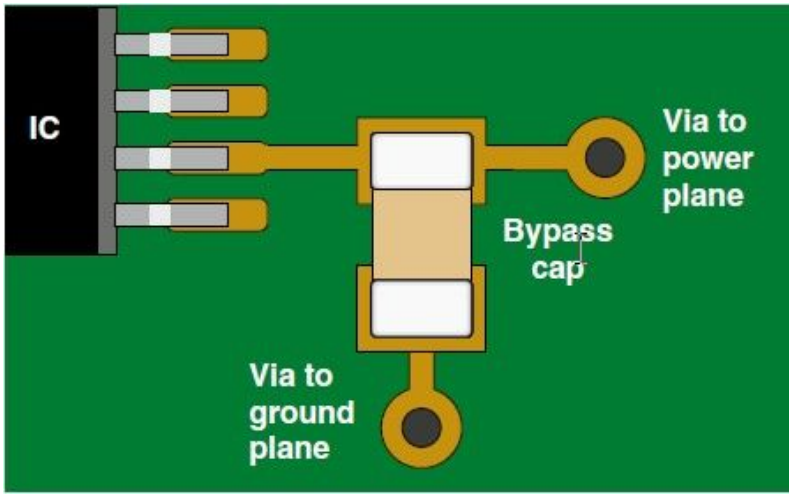
Produsenten har mulighet for å produsere helt ned til seks mils baner uten ekstra kostnad. Men det er ikke noe poeng i å presse marginene. Vi har valgt en banebredde på ti mils. Dette skulle være mer enn bredt nok til de strømmene (maks 1 A) vi har i kretsen vår. Vi har fjernet alle 90 graders svinger for å unngå problemer med refleksjoner [1].

Banebredden påvirker impedansen mellom mikrofon og forforsterker. Vi har regnet ut at den er så liten at den har ikke noe å si for impedanstilpasningen når vi bruker single ended konfigurasjon.

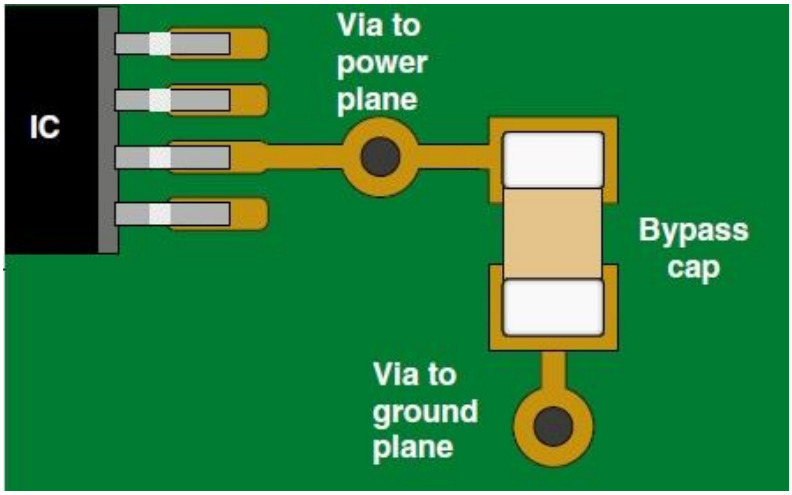
Avkoblingskondensatorer:

Vi bruker stort sett par av avkoblingskondensatorer i kretsen vår, en liten keramisk kondensator og en stor elektrolytt. Det er viktig at avkoblingskondensatoren blir plassert så tett som mulig inntil pinnen på pakka. Derfor har vi lagt den lille keramiske nærmest pinnen og elektrolytten utenfor der igjen. Dersom vi kunne lagt komponenter på undersiden ville det vært naturlig å legge elektrolytten på undersiden.

Det anbefales to forskjellige plasseringer av viaene i forbindelse med avkoblingen, en for analoge kretser og en for digitale kretser. Se Figur 22 og Figur 23. Dette er viktigst ved høye frekvenser og raske stigningstider, men vi har fulgt denne anbefalingen uansett [18].



Figur 21: Avkoblingskondensator analoge kretser, bilde hentet fra [18]



Figur 22: Avkoblingskondensator digitale kretser, bilde hentet fra [18]

3.4 Annet

Andre viktige momenter ved utlegget:

Footprints:

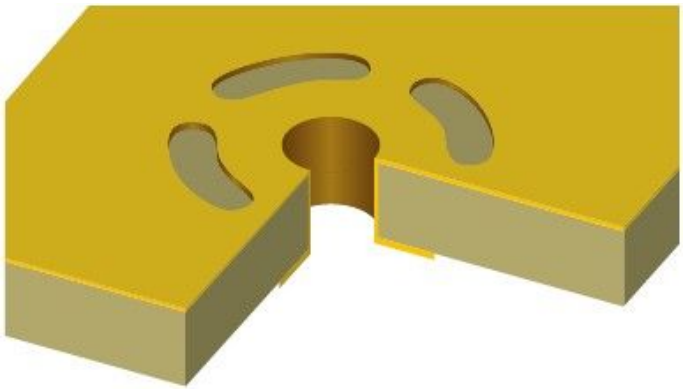
Vi har laget de fleste footprint selv. Mange av komponentene har spesifisert footprint i databladet. Komponenter uten spesifiserte footprints i datablad har vi laget etter anbefalinger i [18]. Kort forklart er borrehull ca. 1,5 ganger større enn selve pinnen. Dette for å ta høyde for utvidelse av metall som blir varmt og opptrekk av loddetinn. Loddeører er ca. 20 mil større enn hullet og antipad er ca. 20 mil større enn loddeørene igjen. Vi opplevde ikke problemer med feil dimensjoner på footprints i forhold til de fysiske komponentene.

Via:

Hele poenget med jordplan er å ha lav impedans på returbanene. Derfor er det viktig å unngå lange baner mellom komponent og via. Via plasseres så tett som mulig på komponenten. Tette ansamlinger av via vil føre til høyere impedans i det området på planene. Dette vil vi unngå for å sikre så korte og direkte returbaner som mulig [1]. Dessuten vil kortet kunne sprekke om det bores hull for tett eller for nærme kanten. Derfor unngår vi dette. Vi har fjernet loddestoppen på det ene laget, dette for å unngå at gasser fanges inne i viaet. Disse gassene kan ekspandere og ødelegge deler av kortet.

Thermal relief:

Alle through-hole komponenter trenger thermal relief for å sikre høy nok oppvarming ved lodding. Vi har brukt standard thermal relief med 20 mils åpninger (spokes width). Se Figur 5.



Figur 23: Thermal relief, bilde hentet fra [18]

Skjerming på 40-pins konnektor:

Vi har en del pinner til overs som vi ikke har bruk for. Vi har valgt å bruke en del av de til skjerming mellom analog og digital del. Alle pinnene mellom 5 V og 3,3 V fra FPGA er koblet til jord.

Loddeører:

Vi ble anbefalt av han som skal montere kortet vårt å øke størrelsen på loddeørene med 20 mils på bunnlaget i forhold til topplaget. Dette for å sikre nok varme til opptrekk av nok loddetinn.

Loddestopp:

Det anbefales at det er litt avstand mellom loddestoppen og loddeørene. Hvor mye avhenger av produsenten. Vi har ikke fått noe tydelig svar fra produsent på dette og har laget en margin på 0,1 mm mellom loddestopp og loddeører. På grunn av små feilmarginer kan loddestoppen bli plassert delvis inne på loddeørene, dette vil føre til en dårligere lodding. Marginen på 0,1 mm vil sikre at loddestoppen kommer utenfor loddeørene.

Powerpad Package:

Denne ble diskutert tidligere under kapittelet om ADC-modulen. I utlegget ble den laget nøyaktig ut i fra anbefalingene i databladet.

Fidusial merker:

Montøren ville ha fidusial-merker på kortet, to merker plassert i to diagonale hjørner. Disse merkene brukes til å plassere loddepastastensilen korrekt slik at ingen komponenter blir forskjøvet fra sin tiltenkte posisjon.

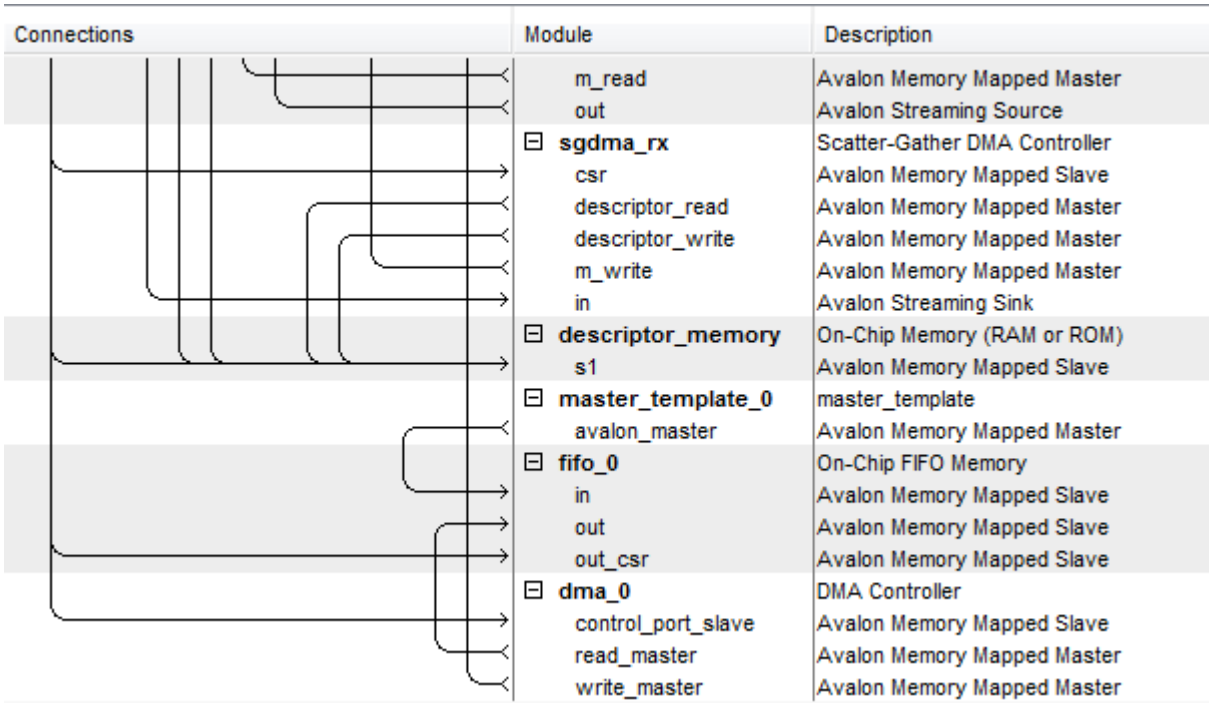
4 FPGA-design

Vi vil i dette kapittelet gjøre rede for hvordan vi har implementert SoC (System on Chip) på FPGA-en. Valg av arkitektur og kommunikasjon mellom moduler er sentrale temaer – i tillegg til design av enkeltmoduler. Data som sendes ut fra FPGA-kortet mottas på en mottaker-PC. Programvaren vi har utviklet for dette vil også gjøres rede for i dette kapittelet.

4.1 Forklaring av begreper

For at leseren lettere skal forstå de neste kapittelene, er det nødvendig å beskrive noen begreper og teknologier som benyttes.

SOPCBuilder [24]: (System On a Programmable Chip Builder) Programvare fra Altera som gjør det lett å implementere et helt system. Dette gjøres ved å velge de moduler som ønskes, og "dra" de inn i en samlet oversikt. Her kan man trekke streker mellom modulene for å koble de sammen på Avalonbussen, se Figur 24. Modulene er såkalte IP-cores. Dette er ferdiglagde, parametriserbare, implementasjoner av for eksempel NIOS-prosessor, FIFO-buffer og Ethernet MAC (Medium Access Controller).



Figur 24: Oversikt over moduler i SOPCbuilder

Avalon-buss [25]: Alteras systembuss, som binder sammen modulene i SOPCBuilder. Det er flere utgaver av Avalon grensesnittet, men det er to utgaver som bør nevnes da disse er brukt aktivt i planleggingen.

- Avalon MM (Memory Mapped): Toveis multimaster-grensesnitt som gir masteren tilgang til slaven som om det var et minneområde.
- Avalon ST (Streaming): Enveis en-til-en kommunikasjon mellom komponenter som krever høy båndbredde og lav tidsforsinkelse. [5]

Komponenter som DMA (Direct Memory Access), ikke SGDMA, og onchip FIFO har ikke støtte for Avalon ST og vi er derfor avhengig av å bruke Avalon MM til disse.

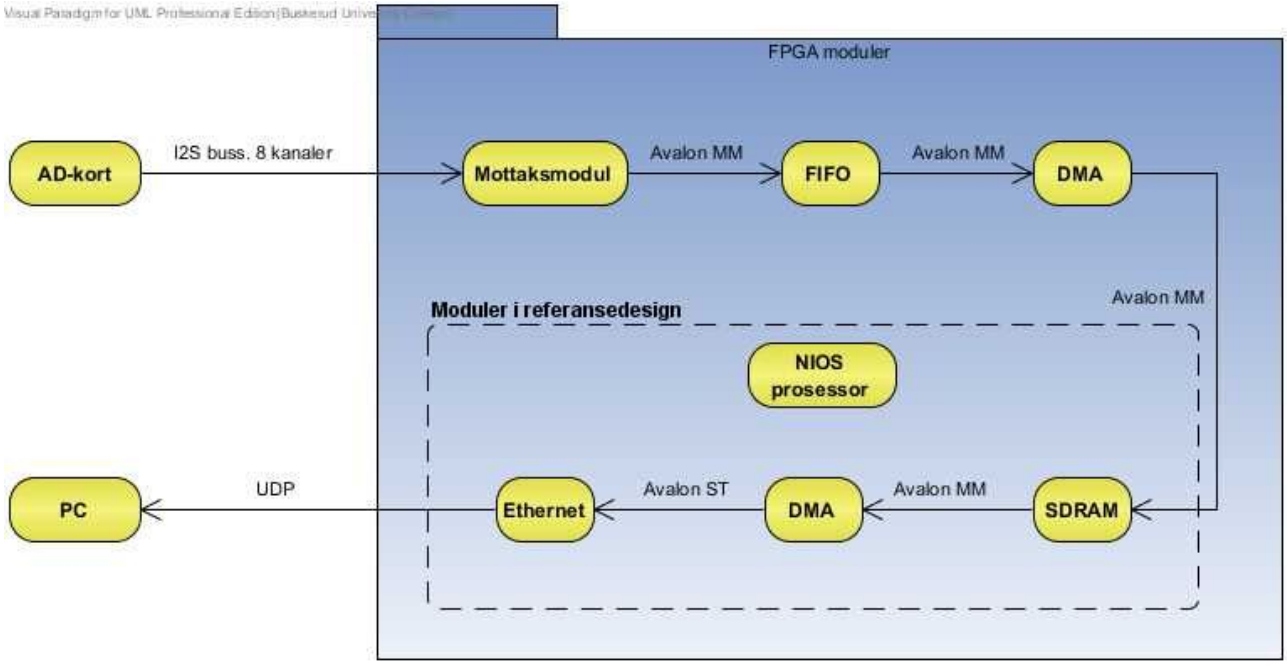
NIOS II-prosessor [26]: Soft-core prosessor fra Altera som kan implementeres i programmerbar logikk og minneblokker på FPGA-en. På denne prosessoren kjøres operativsystemet **MicroC/OS-II [27]**. Dette er et sanntids operativsystem som blir brukt i referansedesignet vi tar utgangspunkt i.

NicheStack [28]: En protokollsamling spesielt utviklet for embedded (innbakte) enheter. Den implementerer protokoller som TCP, UDP, IP, FTP, Telnet, DHCP osv. NicheStack bruker standard Berkley socket API.

SGDMA [29]: Altera sin “scatter gather” DMA kontroller. Denne leser sine egne descriptorer fremfor at transaksjoner blir initiert av CPU. SGDMA kan lese fra og skrive til usammenhengende minneområder.

4.2 Arkitektur

Systemet vårt tar utgangspunkt i et referansedesign for FPGA-kortet vi bruker. Referansedesignet er utviklet for en TCP/IP webserver og har mye av det vi trenger for å flytte data fra FPGA og ut på Ethernet. Vi beholder brukbar funksjonalitet og fjerne resterende. Vi har også laget en mottaksmodul som skriver data fra AD-kortet til et FIFO. En DMA-kontroller avlaster CPU-en ved å flytte data fra FIFO over til SDRAM. Deretter pakkes dataene i UDP/IP-Ethernet pakker og sendes videre over nettverkskabelen. Altera sitt Megacore bibliotek har de nødvendige IP-cores for FIFO og DMA.



Figur 25: Dataflyt og utvalgte moduler

4.2.1 Modifikasjon av referansedesign

Referansedesignet gir oss mye funksjonalitet, og vi sparte derfor en del tid på å tilpasse dette til vår bruk fremfor å lage alt fra bunn. Som standard bruker referansedesignet SDRAM som systemminne. Dette tillater stor nok gjennomstrømning til vår bruk. I følge [7, kap. 7] kan gjennomstrømning på SDRAM generelt beregnes som:

$$T = \frac{clock \cdot Bytes\ per\ transfer}{(Read\ ratio \cdot Cycles\ per\ read + Write\ ratio \cdot Cycles\ per\ write)} \tag{10}$$

Vi har ikke funnet data på antall «burst» SDRAM-kontrolleren tillater eller hvilken hastighet SDRAM-en vår kjører på siden brikken gir flere muligheter for dette. Ved å ikke «burste», bruke 50/50 lese-skrive forhold, og være klokket på laveste mulig frekvens vil i følge datablad [8] gi en estimert gjennomstrømning på 266 MB/s. Vårt system har totalt en gjennomstrømning på 2,3 MB/s. Vi anslo at denne RAM-en er rask nok.

Referansedesignet benytter også den mest avanserte av NIOS-prosessorene (/f) [26 kapittel 5-4]. Denne bruker noe flere LE-er (Logic Elements) enn de andre, har høyest ytelse og fordeler som for eksempel Dynamisk branch prediction [13]. NIOSII/f har også data og instruksjonscache, noe som øker ytelsen betraktelig. Bruk av nettverksprotokoll innebærer mye behandling av de samme dataene. Siden FPGA-kortet har nok LE-er, og ytelse er avgjørende, beholdt vi NIOSII/f i vårt system. Se kapittel 4.3.8.

Referansedesignet bruker en SGDMA-kontroller til å hente data fra SDRAM til Ethernet-modulen. Vi vurderte å la SGDMA-kontrolleren hente data direkte til FIFO. På denne måten ville ikke datastrømmen hatt behov å gå veien om SDRAM. Enda viktigere ville lesingen fra FIFO være

jevnere og vi kunne benyttet en mindre FIFO. Ulempene med denne løsningen er at det avviker en del fra referansedesignet vi tok utgangspunkt i. Ombygging av hardwaredesignet ville vært fort gjort, men NicheStacken som vi er sterkt avhengig av (Ethernet/UDP) krever at data ligger tilgjengelig på sammenhengende minne. Dette er ikke tilfelle med et FIFO buffer, og vi måtte da valgt en annen løsning enn NicheStacken [9].

4.2.2 Systembuss

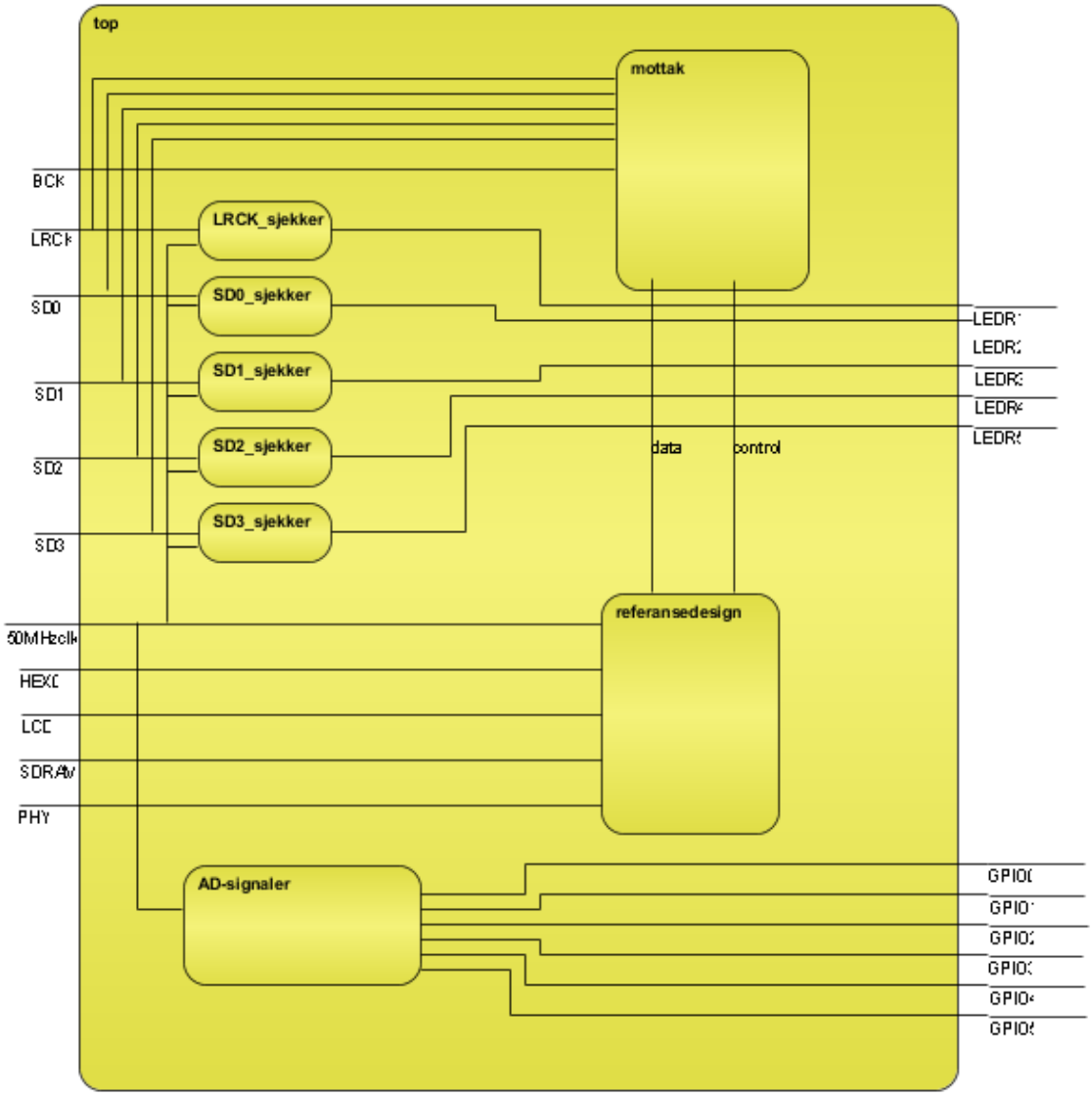
Altera sine designverktøy og referansedesignet er begge tungt forankret i Altera sin systembuss Avalon [25]. Siden tradisjonelle systembusser krever at bussmastere må kjempe om tilgang til bussen har vi vurdert om Avalonbussen kunne bli en flaskehals. Det vil den ikke bli, siden den bruker et konsept Altera kaller «slave side arbitration». Slave side arbitration går ut på at hver master har individuelle linjer til hver slave den aksesserer. Hver master kjemper da bare om tilgang til den slaven den vil aksessere – i kontrast til å kjempe om å aksessere bussen. Dette tillater at flere mastere kan være aktive på bussen samtidig, så lenge de aksesserer ulike slaver [12].

4.2.3 Operativsystem

For å benytte NicheStack på en måte som er støttet av Altera var vi avhengige av å kjøre RTOS-et (Real Time Operating System) MicroC/OS-II på NIOS II prosessoren [27]. For å minske belastning på CPU kunne vi omgått dette ved å kjøre NicheStacken i såkalt Super Loop-modus [14]. Vi tok utgangspunkt i å bruke MicroC/OS-II, siden å kjøre i Super Loop-modus kan gi uheldige bieffekter som blant annet minnelekkasje [14]. NicheStack i Super Loop-modus er i tillegg enda ikke støttet av Altera.

4.3 Moduldesign

I dette kapittelet vil vi beskrive hvordan vi designet enkeltmoduler. Hvilke ferdigmoduler vi har brukt, hvordan vi har parametrisert de og hvordan de brukes fra NIOSII. Samt hvordan vi har designet mottaksmodulen. I tillegg til modulene i dette kapittelet har vi laget noen moduler til bruk i feilsøking og rapportering. Disse beskrives under kapittel 4.5.3. Figur 26 viser en hierarkisk oversikt over de relevante delene av systemet. Merk at mye av designet ligger under «referansedesign» i figurene.



Figur 26: Modulhierarki

4.3.1 Triple-speed Ethernet

Implementasjonen av Ethernet-modulen er en forhåndsdefinert IP-core, Triple-Speed Ethernet megacore (TSE). Denne tilfredsstilte vårt krav til overføring av data over Ethernet. TSE-modulen implementerer et 10/100/1000 Mbps Ethernet grensesnitt (MAC og PCS - Physical Coding Sublayer) og støtter IEEE 802.3 standarden [6]. Denne modulen er en del av referansedesignet vi tok utgangspunkt i. Båndbredden vi har greid å oppnå er på 36Mb/s (for mulige forbedringer se 6) , til sammenlikning kommer data fra AD-kortet med en hastighet på 18,4 Mb/s (megabit per sekund).

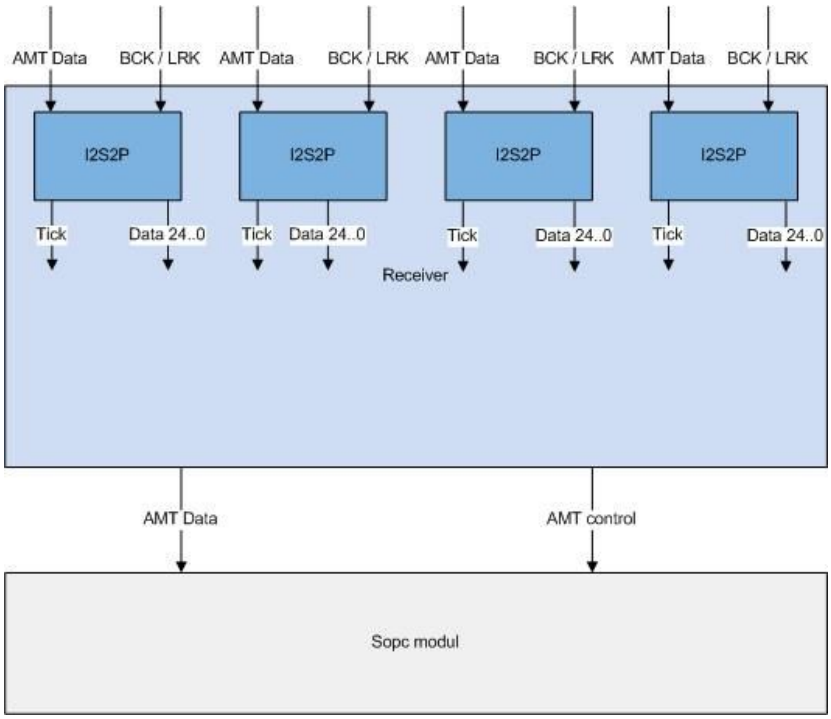
Da samplingsraten er 96 kHz og oppløsningen er 24 bit i åtte kanaler.

$$96\,kHz \cdot 24\,bit \cdot 8\,kanaler = 18{,}4\,Mbit/s \tag{11}$$

4.3.2 Mottak

Data kommer inn fra AD-konvertere i gjennom I²S-busser. Altera hadde ikke tilgjengelige IP-cores som kan lese fra I²S. Vi vurderte å bruke åpne IP-cores fra f.eks. Opencores.com [37]. Disse var imidlertidig ikke tilgjengelig under en lisens som gjør de brukelige i vårt system, så vi lagde denne modulen selv. Siden I²S-spesifikasjonen ikke er entydig, utviklet vi den ut i fra spesifikasjonene i databladet til AD-konverteren [3].

Fra AD-kortet kommer lyddata inn på FPGA-en over fire stk I²S-datalinjer med felles bitklokke BCK (6,144 MHz) og Left-Right-klokke LRCK (96 kHz). To lydkanaler kommer inn over hver datalinje. Mottaksmodulen implementerer derfor en I²S-mottaker for hver av datalinjene. Dette er en seriell til parallell konvertering.



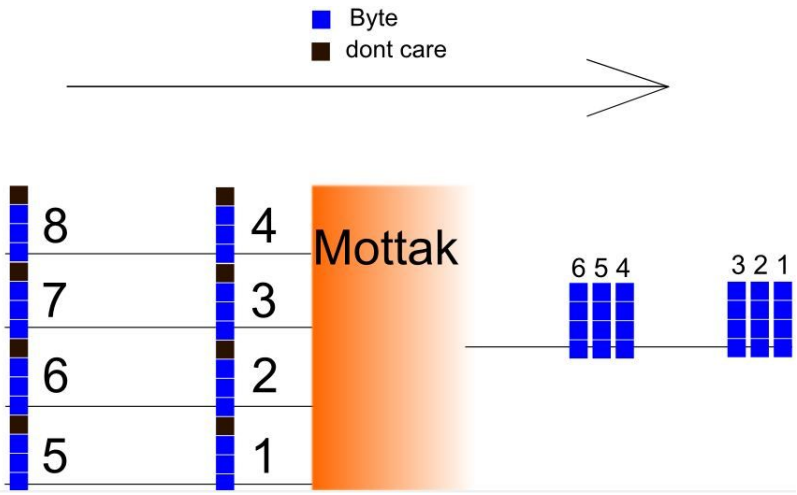
Figur 27:Blokkdiagram, Mottak

Mottaksmodulen tar i mot data som er synkronisert mot BCK. Vi har måttet ta et valg om vi skal la mottaksmodulen være klokket av BCK og dermed introdusere et nytt klokkedomene, eller være klokket av systemklokken og istedet bruke en «edge-detector» på BCK for å bestemme når det skal leses. Til tross for at vi introduserte et nytt klokkedomene gikk vi for førstnevnte løsning. Ved å la mottakslogikken være klokket av BCK-klokken vil implementasjon av mottaksmodulen bli mindre kompleks. Dataene sendes gjennom et FIFO-buffer på grunn av «rykkvis» pakking og viderebehandling. FIFO-bufferet har to porter som trygt kan aksesseres fra ulike klokkedomener – dette sørger for synkronisering. FIFO-bufferet fra Altera har nødvendige innebygde synkroniseringsbuffere. Vi tar inn BCK på en dedikert klokkepinne (SMA) for å få den på internt klokkenett.

Mottaksmodulen sender dataene videre til FIFO-bufferet. For å oppnå dette kunne vi enten laget vår egen implementasjon av et Avalongrensesnitt, eller bruke en modul fra Altera som

implementerer MM-master grensesnitt, og eksporterer kontroll og datasignaler til vår egenlagde komponent [15]. Vi gikk for sistnevnte da vi gjennom prototyping fant ut at det var det letteste.

For ikke å påføre systemet ekstra belastning valgte vi å slå sammen 4 - 24 bits sampler til 3 - 32 bits word allerede i mottaks-modulen. Dette kan vi gjøre fordi vi får inn 4 byte for hver kanal, men bare 3 av disse inneholder data. Bytene sender vi i little-endian byte-rekkefølge i hver kanal, siden de vanligste operativsystemene bruker dette (Linux x86, Windows, OS-X). Modulen videresender data i en kanalrekkefølge som samsvarer med plasseringer av mikrofon-konnektorene på AD-kortet.



Figur 28: Fra 24 bits sampler på I²S buss, til 32 bits word i systemet

I tilfelle man senere vil endre byterekkefølge til big-endian – er kode til dette kommentert ut i VHDL-filen «receiver.vhd».

For å ha mulighet til å pause mottaksmodulens fylling av FIFO-bufferet (via AMT - Avalon Master Template), implementerte vi mottaksmodulen med et stopp-signal. Når dette signalet går høyt vil mottaksmodulen gjøre ferdig sampelet modulen holder på med, initiere tømming av AMT, for så og vente til stoppsignalet går lavt. Deretter synkroniserer den seg mot LRCK før den fortsetter som normalt. På denne måten kan brukeren av mottaksprogrammet styre datastrømmen fra AD-kortet. For en mer teknisk beskrivelse av mottaksmodulen se [43].

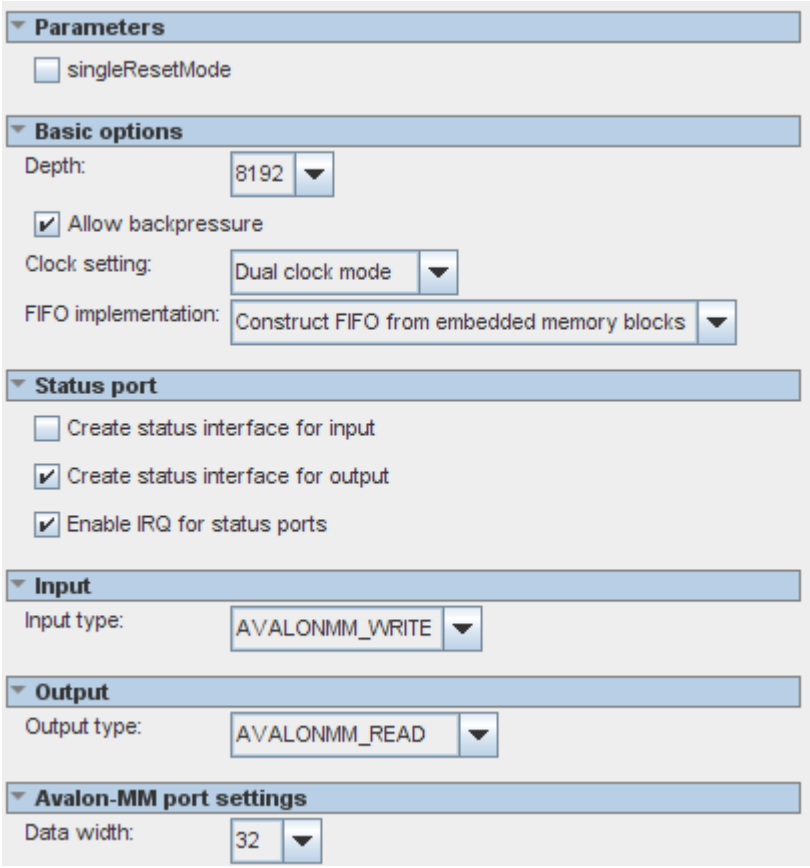
Både «I2S2P» og «Receiver» modulen er laget som en FSMD med asynkron reset. AMT-en vi bruker befinner seg i SOPC-modulen.

4.3.3 FIFO

Som mellomlager brukes det et dual-port FIFO-buffer [30]. I den ene porten skriver mottaksmodulen data i jevnt tempo, mens en DMA henter fra den andre porten så fort som pakking og videresendingen tillater det. SDRAM-en vi skriver til senere har kun en port, og en annen DMA leser ut like ofte, og mye, som det skrives inn. Dette gjør at FIFO-en må være i stand til å holde på data i lesepausene.

I mellomlageret går lyddataene fra BCK-domenet over til resten av systemet sitt 50MHz

klokkedomene. Den andre hensikten med FIFO-en er å sørge for synkronisering i mellom disse klokkedomenene. Den enkleste og raskeste måten å implementere en to ports FIFO var å benytte en ferdiglaget IP-core som bruker FPGA-en sine interne minneblokker. Denne har også alle nødvendige synkroniseringsregistre. Det har vist seg at vi har mer enn nok on-chip ram på FPGA-brikken. Vi har endt opp med en FIFO på 32 kB, til sammenlikning sender vi 1008(4.5.1) byte per UDP-pakke. Vår erfaring igjennom prosjektet er at FIFO-en kunne vært betydelig mindre (under 2 kB) uten at dette ville hatt noen negativ virkning, likevel velger vi å ha den så stor siden vi har så mye minneblokker.



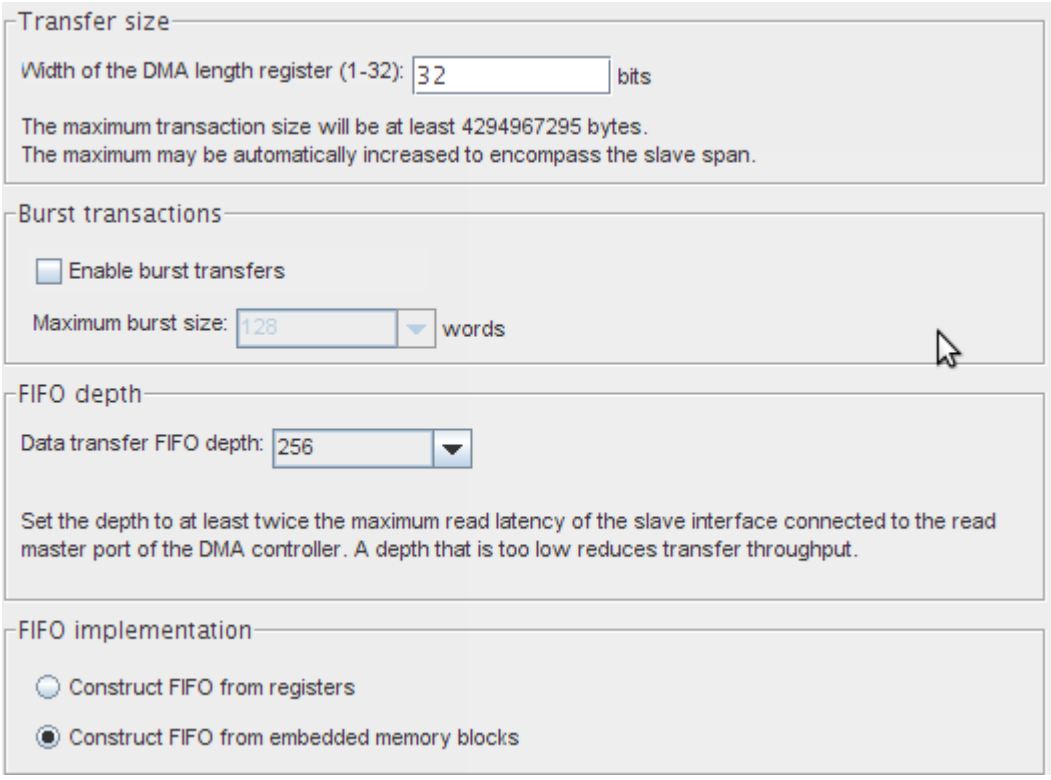
Figur 29: Parametrisering av FIFO

Figuren over viser våre valg for parametrisering av FIFO modulen. Den er gjort romslig, to porter og en statusport mot NIOSII. Den har også Avalon-MM-grensesnitt. Siden vi opererer med to klokkedomener og bruk av statusport for toports FIFO-buffere kan dette øke faren for metastabilitet [30]. I kompileringsrapporten har vi sett at den ikke er avskrekkende (Mean Time Before Failure > 1 milliard år) [44]. for mer informasjon om størrelsen til bufferet, se kap 4.3.8.

4.3.4 DMA

Istedet for at CPU flytter data fra FIFO til SDRAM valgte vi å gå for en DMA-løsning. DMA-kontrolleren jobber parallelt med CPU-en, og sørger på den måten for at CPU kan gjøre andre ting. Dette frigjør ressurser hos CPU til operativsystem og nettverksaktiviteter samt arbitrering av systembussen. Vi har to ulike DMA-kontrollere tilgjengelig: «Vanlig» DMA og SGDMA. Ved bruk av «vanlig» DMA må CPU initiere hver overføring og det kan kun skrives til en sammenhengende datablokk. SGDMA starter overføringer på egenhånd ved å lese i et "descriptor-minne", og kan skrive til usammenhengende adresser [29].

Vi valgte å bruke "vanlig DMA" [31] siden vi kun trenger å skrive til en sammenhengende minneblokk. Samt at en kontroller som opererer helt på egenhånd introduserer synkroniseringsproblematikk, og vi vil holde designet så enkelt som mulig.



Figur 30: DMA-parametrisering

DMA-kontrolleren har vi parametrisert som i bildet over. Den har et innebygd FIFO-buffer som vi lager av M9K minneblokker, og gjort romslig. Vi har skrudd av bursting siden det uansett ikke kommer til å være noen å konkurrere om tilgang til FIFO eller SDRAM, bursting krever også mer ressurser. Lengderegisteret er økt fra 13 til 32 bit – etter vi så systemet ikke klarte å overføre ønsket mengde data av gangen. I tillegg kunne vi velge hvor bred datastrøm DMA-en skal kunne takle. Denne settes til 32 bit, siden det er dette vi bruker gjennom hele systemet.

På Figur 30 ser vi at DMA-en har en innebygd FIFO, denne kunne ikke gjøres så stor som det dedikerte FIFO-bufferet. Vi ser likevel at vi kunne brukt denne istedenfor en egen FIFO-Modul, men

da måtte vi også laget vårt eget Avalon-slavegrensesnitt til mottaksmodulen istenfor å bruke Altera sin AMT. Vi ville heller ikke oppnådd synkroniserering mellom de to klokkedomene, så vi måtte klokket avlesing fra I^2S busser på systemklokken 38.

4.3.5 FIFO og DMA buss-grensesnitt

Siden det er en del data som fraktes fra I^2S-buss til RAM har vi vurdert å bruke Avalon sitt ST grensesnitt. Denne er raskere enn MM-grensesnittet og ville tillatt å skrive hele sample-bredden til FIFO på en enkelt klokkepuls. Et bussbredde-adapter ville ha delt strømmen i 32 bit word før den ble skrevet til SDRAM. Dette oppsettet ville gjort mottakskomponenten noe enklere å implementere, til gjengjeld måtte vi brukt Altera sin SGDMA kontroller som initierer overføringer på egenhånd, forklart i kapittel 4.3.4. Vi gikk for løsningen med Avalon MM-grensesnitt og den «vanlige DMA-en» [31]. Vi kan ikke oppnå større båndbredde med ST-grensesnitt siden skriving til SDRAM likevel må skje 32 bit av gangen.

4.3.6 Fra SDRAM til Ethernet

NIOSII prosessoren har både data og instruksjons-cache. Vi har gitt vår implementasjon 64KB hver etter at vi så nettverksytelsen økte betraktelig med dette. Bruk av nettverksprotokoll innebærer mye behandling av de samme dataene.

Det at vi bruker data-cache førte i starten til at prosessoren jobbet med «dirty cache» siden DMA-kontrolleren aksesserte det samme minnet utenom cache. Dette oppdaget vi under prototyping, da vi hver gang oppdaget at de fem første overførte ordene ikke ble funnet. Løsningen på problemet var å bruke «alt_uncached_malloc»-funksjonen i applikasjonsprogrammet. Denne funksjonen sørger for at data fra valgt allokerte minneområdet ikke legges i cache, og all tilgang til området skjer utenfor cache [20, side 9-7]. Applikasjonsprogrammet som kjøres på NIOSII prosessoren fungerer som en socket-server mot mottaker-PC-en. Det er her pakking og videresending av data skjer, samt kontroll av mottaksmodulen. For en utfyllende beskrivelse av applikasjonsprogrammet se [46].

4.3.7 Prosessor og RAM

Vi har brukt den mest avanserte konfigurasjonen av NIOSII prosessoren vi har hatt tilgang til – NIOSII /f. Til forskjell fra konfigurasjonene under har den 6-trinns instruksjonspipeline, dynamisk branch-prediction og separat data og instruksjons-cache. På grunn av dette bruker den 25% flere LE'er – noe vi har nok av 4.3.8. Spesielt cache-en har vi sett betyr mye for nettverksytelsen til systemet 4.2.1. Vi har gitt prosessoren 64kB data og Instruksjons-cache.

Vi har 128MB SDRAM, 8MB FLASH-minne, 2MB SRAM [47] å velge mellom som RAM for prosessoren. Vi tok utgangspunkt i å bruke SDRAM-en 4.2.1 - men har undersøkt om vi kunne brukt en annen type siden SDRAM-en er den tregeste å aksessere. Vi har derfor undersøkt hvor

mye av SDRAM vi bruker. Den brukes til applikasjonsprogrammet med stack og heap inkludert 1 MB buffer. Størrelse på stack og heap er dynamisk, så de vet vi ikke. Det vi derimot vet er at applikasjonsprogrammet med RTOS tar 6,45 MB, og at vi lager oss et buffer på 1 MB. Altså trenger NIOSII prosessoren 7,45 MB RAM pluss stacken og deler av heapen. Mesteparten av dette går trolig til RTOS og protokollstakk. Vi er derfor trolig avhengige av SDRAM-en.

4.3.8 Ressursbruk og kompileringsrapport

Kompileringsrapporten generert av Quartus programvaren viser at systemet vårt bruker følgende ressurser, se Tabell 9 [45]:

Tabell 9: Ressursbruk

Type	Forbruk i andel	Forbruk	Tilgjengelig
Logiske elementer	15,00%	17262	114448
Onchip minne-bits	67,00%	2667290	3981312
PLL-er	50,00%	2	4
Multipliers	<1%	4	532

Vi ser at ressursbruken ikke er avskrekkende, bortsett kanskje fra minne-bits. Referansedesignet vi tok utgangspunkt i bruker PLL-er som klokke til ulike deler av systemet. Den ene brukes i størsteparten av systemet, mens nummer to brukes til en del IO-moduler. I tillegg ser vi at to Multipliers blir brukt av noen av IP-corene vi bruker. Antall logiske elementer er lavt som forventet, siden FPGA-brikken vi har er av det romslige slaget. On-chip minne er det vi bruker størst andel av. FIFO-modulen benytter disse, og for sikkerhets skyld har vi gjort den så stor stor som det går an. Det er også andre deler av vårt design som bruker minnebits. Vi har sett nærmere på hvilke, se Tabell 10.

Tabell 10: Største forbrukere av M9k minnebit

Submodul	Forbruk i andel (av tilgjenelig)	Forbruk av minnebits	Kommentar
NIOSII	28,00%	1129216	Over 99% er cache for instruksjoner og data, mye kan trolig optimaliseres bort.
onchip_memory_2	26,00%	1048576	Minneområde som var en del av refereansedesiget, og som ikke brukes. Alt kan frigjøres.
FIFO_0	6,50%	262144	FIFO-modulen vår, kan trolig gjøres mye mindre. Se kap. 4.3.3
TSE_mac	3,80%	152240	Mesteparten brukes til et innebygd buffer, uendret fra referansedesignet.
Descriptor memory	0,80%	32768	Minne som brukes av RX og TX – buffer i forbindelse med TSE. Uendret fra referansedesignet.

Våre erfaringer igjennom prosjektet er at FIFO-en uten problemer kunne vært en del mindre, eller at en like stor FIFO hadde vært nok for flere kanaler. Likevel vil det være mer å hente på å minske CPU-cache eller bruke onchip_memory_2 dersom man skulle trenge å frigjøre minnebits ved senere anledning. Vi ser også at vi var for pessimistisk i starten når vi anslo hvor stort FIFO-buffer vi trengte.

Grunnen til at vi har beholdt deler av referansedesignet som vi ikke bruker - er at de enkelt kan tas i bruk hvis det skulle bli nødvendig.

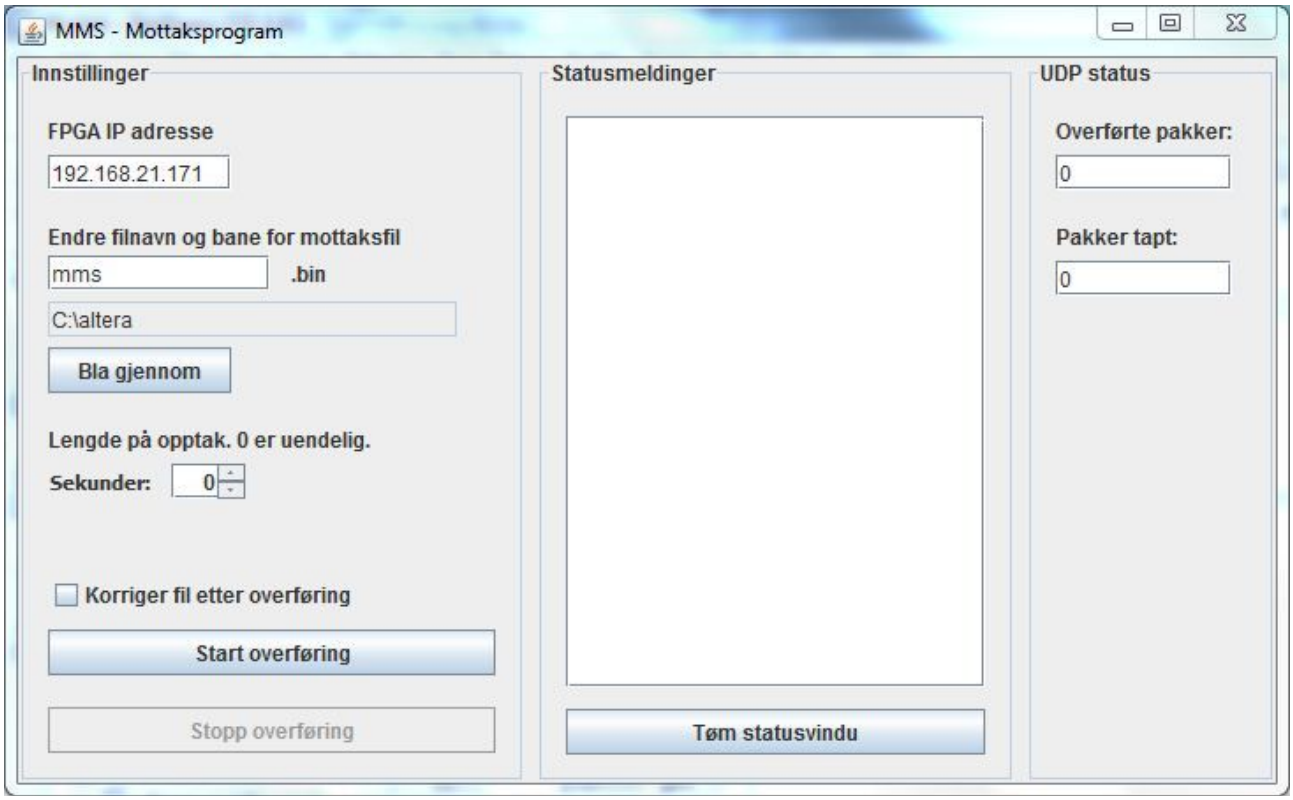
Vi har arvet en del advarsler fra referansedesignet vi tok utgangspunkt i. Disse har vi gått igjennom, og bestemt oss for ikke å gjøre noe med de siden tester viser at systemet fungerer [41]. En del av advarslene går på at inngangspinner ikke har oppgitt setup og hold-tid, og at utgangspinner ikke har «drive-strength». Pinnene det er snakk om er hovedsakelig til moduler eksternt på FPGA-kortet – som for eksempel RAM-brikker. Det vi har sett nøyere på er metastabilitet og «Mean time before failure» siden vi opererer med to klokkedomener. Denne oppgis i rapporten som «Greater than 1 billion years», noe som er akseptabelt [45]. En annen ting vi har gjort er å måle hvor lang tid inngangssignalene bruker på å gå fra 0-90 og 90-0, og registrere dette i Quartus. Denne informasjonen brukes under kompilering til å beregne hvor lang tid signalet bruker på å forplante seg inn fra inngangen.

4.4 Mottak og lagring på PC

Mottaks-PC-en er endepunktet for dataene som blir generert av mikrofonene. Mottaks-PC-en trenger et mottaksprogram og en funksjon for å lagre dette til en fil for senere bruk. For å gjøre

dette mest mulig brukervennlig har vi lagd et grafisk brukergrensesnitt i Java.

4.4.1 Mottaksprogram



Figur 31: GUI for mottaksprogram

Figur 31 viser det grafiske grensesnittet for mottaksprogrammet. Dette har vi utviklet for å ta i mot data fra FPGA-kortet. Før overføringen starter har man mulighet til å endre IP-adressen til FPGA-kortet og hvilken bane og filnavn som dataene skal lagres i. Brukeren kan også stille inn antall sekunder et opptak skal lagres. I tillegg har vi laget en funksjon for å endre amplituden på hver kanal. Dette gjøres for å kompensere for de individuelle forskjellene på mikrofonene. Disse innstillingene gjøres før vi trykker på «start overføring» og dataoverføringen kan begynne. I midtre rute av programmet skrives det ut statusmeldinger for å gi brukeren beskjed om valgene som tas og operasjoner som gjennomføres. I ruten til høyre vil antall datapakker og eventuelt antall tapte pakker bli skrevet ut. Etter hver gang ønsket mengde data er overført og overføringen er stoppet, kan brukeren når som helst starte en ny overføring uten å starte mottaksprogrammet eller NIOS-prosessoren på nytt. Dette gjør hele systemet veldig brukervennlig og enkelt å bruke. For en mer utfyllende beskrivelse av mottaksprogrammet se referanse [32].

4.4.2 Filformat

Filen som lagres består av rådata fra AD-kortet på PCM-format (Pulse Code Modulation) – og

inneholder ingen header eller info om lengde, frekvens eller annen informasjon. Samplefrekvens er 96Khz, og oppløsning er 24 bit. Data lagres på toer-komplementform i kanalrekkefølge som er vist i tabellen under. 24 bit for hver kanal. Byteorder er little endian, se kap 4.3.2. kanalrekkefølgen samsvarer med plassering av mikrofonkonnektorer på AD-kortet. Vi har valgt filendelsen *.bin* og *.bin_corr* for filene som er korrigert av mottaksprogrammet.

Tabell 11: Kanalrekkefølge

Kanalrekkefølge i fil							
J3	J4	J6	J7	J8	J10	J12	J14

4.4.3 Visualisering

For å visualisere data som kommer inn på mottaker-PCen bruker vi Matlab, se Vedlegg I – Matlabprogram for visualisering , og et script som viser signalene på hver kanal. 24 bit av gangen leses som et 24 bits heltall med fortegn, og plottes deretter i en graf. Dette har også vært til god hjelp under testing og feilsøking. Vi har ikke prioritert sanntids-visualisering.

4.5 Systemfeil

Feil kan skje med systemet. Noen feil vil tillate at man fortsetter på redusert kapasitet, andre kan bare rapporteres. Her vil vi beskrive systemets feiltoleranse – samt feilrapportering og innebygde verktøy for feilsøking.

4.5.1 Toleranse

Flere ting kan gå galt med et så kompleks system som vi har laget, og vi har hatt et ønske om å lage systemet så tolerant som mulig. Hvis komponenter som brukes på FPGA-kortet skulle feile, er det stor sannsynlighet for at systemet blir ubrukelig. Dette kan vi gjøre lite med, bortsett fra rapportering. Hvis derimot noe skulle feile før FPGA-kortet i signalflyten, kan det hende vi kan fortsette samplingen på redusert kapasitet.

Feil med BCK eller LRCK vil gjøre systemet ubrukelig – siden de serielle datalinjene er synkronisert mot disse. Hvis det skulle være feil med en eller flere serielle datalinjer vil systemet fortsatt overføre data fra alle linjene. Signalindikatorene på linjene vil da indikere hvilken linje som har aktivitet, og brukeren må da ignorere data fra disse kanalene.

Hvis mottaks-PC-en ikke har god nok ytelse til å behandle og lagre data like fort som det kommer inn – vil pakker forkastes i protokollstakken på mottakerside og gå tapt. Dette ligger i naturen til UDP-protokollen. Vi har tatt hensyn til dette ved at brukeren varsles om pakketap igjennom GUI, og at pakker har en slik størrelse at kanalene i mottakerfil ikke blir stokket om dersom en pakke går tapt.

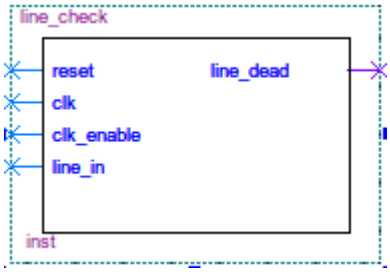
4.5.2 NIOS-rapportering

I applikasjonsprogrammet som kjøres på NIOSII-prosessoren [46] kan det også oppstå feil. Rapportering fra NIOS vil være mest utfyllende ved hjelp av NIOSII-terminalen. Her vil både statusmeldinger og feilmeldinger skrives ut underveis som koden kjøres. Terminalen behøver ikke være i bruk under kjøring av systemet, så dersom en feil oppstår vil vi at dette skal vises i andre former enn kun som en melding i terminal. Derfor har vi tatt i bruk syv-segmentene på FPGA-kortet. Dersom API-funksjoner feiler, som for eksempel at FIFO er full ved oppstart vil syv-segment nummer 7 («HEX7» på FPGA-kortet) vise et siffer. Hvilket siffer som samsvarer med hvilket problem vil beskrives i [33]. På denne måten vil brukeren av systemet få flere muligheter til å oppdage feil under bruk. Dette øker brukervennligheten i systemet vårt.

4.5.3 Sjekk av linjeaktivitet

Vi har laget en mekanisme for å varsle dersom det ikke kommer inn forventet aktivitet på LRCK og de serielle datalinjene. Det var i utgangspunktet kun ment til bruk under utvikling og testing. Vi så at dette kunne være nyttig for å oppdage feil i det ferdige systemet også. For eksempel dersom kabelen mellom AD-kort og FPGA-kort ikke sitter godt nok i kontaktene.

Feilsjekkingsmodulen er laget som et shiftregister med valgfri størrelse. For hver klokkepuls vil det samples fra *line_in*, og når shiftregisteret inneholder bare enere eller bare nullere – vil *line_dead* gå høy.



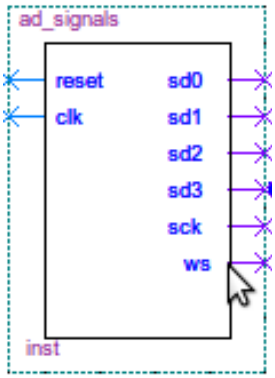
Figur 32: Linjesjekkmodul

En slik modul er koblet på hver inngang bortsett fra BCK (se Figur 26), og gitt de så lange registre at de ikke varsler for tidlig. *line_dead*-signalene har vi koblet på røde LED-lamper på FPGA-kortet. Se [33] for oversikt over hvilke.

4.5.4 AD-signalgenerator

Dette er en modul som er laget for å teste systemet fra inngangspinner til mottaks-PC. Den kjører på systemklokken – og genererer signaler som er kompatible med inngangs-grensesnittet på FPGA-kortet. Som med linjesjekkermodulen har vi også valgt å ta med denne til det ferdige designet. Vi har sett at den er et nyttig verktøy for feilsøking. Den kan brukes til å finne ut om eventuelle feil

ligger før eller etter inngangspinnene på FPGA-en i signalflyten.



Figur 33: Signalgenerator

Signalene fra modulen har vi gjort tilgjengelig på GPIO-pinner på 40-pins konnektoren og SMA-konnektor som står utilkoblet under vanlig bruk. Se [33] for instruksjon. Signalene er ikke ment til å emulere lyd, bare teste dataflyt fra inngangspinner på FPGA-kort til mottaksprogram på PC. For å feilsøke med signalgeneratoren benyttes ledninger mellom utgangspinner og inngangspinner.

Tabell 12: Testsignaler

Signal	Beskrivelse av signal
BCK	50MHz / 8 = 6,25 Mhz
LRCK	6,25/64 MHz
SD0	Sekvens bestående av tallene 0-255 (teller rundt) formatert som 32 bits unsigned integer (MSb først)
SD1	Annenhver bit 0 og 1
SD2	Annenhver bit 0 og 1
SD3	Annenhver bit 0 og 1

5 Testresultater

Vi har gjort en del tester av systemet vårt, både modulvis og på hele systemet sammensatt. Her følger en oppsummering av resultatene. For utførende informasjon om testene og resultatene viser vi til de respektive testrapportene.

5.1 Simulering av analog krets

Vi gjorde en del simuleringer i PSpice av den analoge kretsen før vi produserte AD-kortet. Resultatene viser at signalene ut fra forforsterkeren er som forventet. Impedanstilpasningen fungerer tilfredsstillende. Vi får gjort om signalene fra single ended til differensiell og flyttet signalsvinget opp til 2,5 V. Forsterkningen ligger rett i underkant av 1,75 V/V [49].

5.2 Test av AD-kort

Først sjekket vi frekvensen på de tre klokkelinjene. Frekvensene stemte, men FPGA-kortet klarte ikke å lese av data på klokkepulsene. Det viste seg at stigningstiden til bitklokken var for lang. Løsningen var å fjerne noen dempemotstander i kretsen.

Når det gjelder likheten mellom de åtte mikrofonene, så vi at det var noen små forskjeller fra mikrofon til mikrofon, men dette var vi klar over på forhånd og det er også oppdragsgiver. Vi fant ut at tre av mikrofonene hadde negativ leder koblet til chassis. Det oppstod et problem med at mikrofonene ikke gav det forventede signalsvinget. Det var mye lavere. Vi fant etter hvert ut at lasten mikrofonen så måtte økes. Det krevde ny beregning av motstandsverdier rundt forforsterkeren, og omlodding av AD-kortet.

Til slutt testet vi på forsterkningen og synkroniseringen mellom de åtte kanalene. Synkroniseringen langs tidsaksen var god, men som forventet fant vi litt forskjeller på forsterkningen. Dette kommer i hovedsak av toleranser på komponentverdier. Med mindre man bruker masse tid på å tilpasse verdier på komponenter vil det alltid være forskjeller på analoge kretser. Vi så også, av ukjent grunn, at fire av mikrofonene vi bruker ligger omtrent en tredjedel under de fire andre mikrofonene. Vi valgte å se litt på digital skalering av signalene i etterkant av opptaket.

Frekvensresponsen til AD-kortet var omtrent som forventet, og responsen er jevn i området fra 20 Hz opp til 10 kHz.

Ellers så fant vi to små avvik som sannsynligvis skyldes spenningsforsyningen. Vi har en liten negativ DC-offset på ca. 50 mV. Og det er små variasjoner på amplituden innad i hver kanal langs tidsaksen. Disse får vi ikke gjort noe med i første omgang, men vil sannsynligvis ha lite å si for den bruken systemet er tenkt til [38].

5.3 FPGA-test

Etter at alle modulene var satt sammen til et samlet system, hadde vi ikke mulighet til å teste med reelle data fra AD-kortet (da AD-kortet enda ikke var produsert). Vi lagde da en trafikkgenerator som sendte ut blant annet et sekvensnummer. Vi kunne da se om alle data kom fram til mottaker-PC uten tap, duplisering eller omfordeling.

Testene vi har utført har vist oss at alle modulene, enkeltvis og koblet sammen, fungerer som planlagt.

Vi har blant annet testet Ethernet-overføringshastighet, pakketap mellom FPGA og PC, hastighet og datatap fra mottaksmodul til resten av systemet [41].

5.4 Ytelsetester

Selv om vi ikke hadde spesifikke krav til ytelse gjorde vi noen tester på dette. Først sjekket vi hvor godt spenningsfiltreringen fungerte. Den fungerte godt, men som forventet burde knekkfrekvensen vært lavere for enda bedre effekt på systemet.

Vi gjorde en langtidstest for å se på stabiliteten over tid. Ved kjøring av systemet i ca. 40 timer opplevde vi ingen pakketap eller andre problemer.

Vi sammenlignet støyen ved bruk av FPGA-forsyning og eksterne spenningsforsyninger. Testene ble gjort uten mikrofoner tilkoblet, så målingene gikk kun på støy internt på AD-kortet. Som forventet var det mer støy ved bruk av FPGA-forsyningen. Vi fant ut at systemet hadde støy i de åtte laveste bitene ved ekstern spenningsforsyning og i de ti laveste bitene ved FPGA-forsyning. Dette synes vi likevel er akseptabelt. ADC-en kan maksimalt gi 19 bits oppløsning, så det vil være støy i de fem laveste bitene uansett [40].

5.5 Systemtest

Systemtesten ble gjennomført uten at nye problemstillinger oppstod. Vi fikk verifisert at arbeidsområdet, med spenningsforsyning fra FPGA-kortet, er omtrent 14 bit stort, fra ca 52 – 128 dB SPL [42].

6 Forslag til forbedringer

Den viktigste forbedringen vi kan foreslå på AD-kortet går på spenningsforsyningen. For optimal ytelse er det helt nødvendig med egne regulatorer for hver seksjon på kortet. Men det krever tilgang på en forsyningsspenning på mer enn 5 VDC på grunn av spenningsfallet over regulatorene.

Vi vil også foreslå andre mikrofoner, eventuelt sikre at det er åtte identiske C550H-mikrofoner som ikke har ledere koblet til chassi. Dette for å kunne bruke differensiell konfigurasjon. Da kan vi bruke færre kondensatorer, som i tillegg tar mindre plass og er mye billigere.

For å oppnå helt flat frekvensrespons opp til 20 kHz, må tilbakekoblingskondensatorene rundt forforsterkeren bytte verdi fra 1 nF til 0,1 nF.

Vi ser også at vi kunne laget en smartere kabel-løsning. 40-pins konnektoren tar mye plass, og vi kunne fint klart oss med 10-20 pinner. En annen ting er at det nåværende kortet har en liten koblingsfeil som gjør at 40-pins kabelen må modifiseres litt.

En ting vi har savnet har vært lysdioder som indikerer at kortet har korrekt spenning. Dette er veldig enkelt å legge inn, og gjør det enkelt å se at spenningsforsyningen fungerer korrekt.

Et krav vi ikke har oppfylt går på stabling, det vil si koble sammen flere AD-kort. Det som trengs for å gjøre dette er et buffer på systemklokka. Signalet rutes til en SMA-konnektor tilsvarende løsningen vi har valgt for bitklokka. Dette er enkelt å få til.

Hva gjelder forbedringer på FPGA-systemet, så har vi gjort noen tester av hvordan systemet ville oppføre seg med flere kanaler. Dette kan være hendig å vite noe om dersom det blir aktuelt å utvide systemet. Ved å doble antall kanaler vil man også doble mengden data som går gjennom systemet. Under en test der vi så hvor stor båndbredde vi kan få over Ethernet, oppnådde vi 34,5 Mbit/s. En utvidelse til 16 kanaler med 24 bits oppløsning vil kreve en overføringshastighet på 36,8 Mbit/s.

$$96000\,Hz \cdot 24\,bit \cdot 16\,kanaler = 36{,}8\,Mbit/s \tag{12}$$

Dette ville dermed ikke vært mulig med den båndbredden vi får til. Hvis vi derimot setter ned oppløsningen til 16 bit, ville nødvendig overføringshastighet være 24,5 Mbit/s.

$$96000\,Hz \cdot 16\,bit \cdot 16\,kanaler = 24{,}5\,Mbit/s \tag{13}$$

Dette kunne blitt gjort fordi vi får ti bit med støy fra AD-kortet (åtte bit med ekstern strømforsyning).

Vi gjorde en test ved å sende denne datamengden igjennom systemet vårt fra mottaksmodul til mottaksprogram. Det gikk uten problemer. Vi kunne altså hatt 16 kanaler med 16 bits oppløsning med bare små endringer i mottaksmodulen.

Større overføringshastighet kan også oppnås ved å maskinvare-akselerere deler av protokollstakken, eller komprimere data i maskinvare før det sendes ut. Siden applikasjonsprogrammet vårt aksesserer RAM-en flittig – kan det også hjelpe å bruke en raskere RAM-type. Vi har ikke profilert applikasjonsprogrammet, men dette er mulig med Altera sine

verktøy og kunne også økt mulig overføringshastighet.

Vi har sett på om vi kan komme utenom å bruke SDRAM-en, og istedet bare bruke on-chip minne eller SRAM brikke som er raskere. Det lar seg ikke gjøre siden applikasjonsprogrammet på NIOSII prosessoren trenger minst 7,45 MB RAM se kapittel 4.3.8. Vi kan bruke maks ~3 KB On-chip RAM (ledig + onchip_memory2 + optimaliserer cache) se Tabell 10. Av ekstern SRAM har vi 2 MB. FPGA-kortet har derimot en 8 MB flashbrikke som kunne holdt – avhengig av størrelse på stack og heap.

Vi kjører applikasjonsprogrammet vårt i et RTOS som vi ikke benytter noe særlig funksjonalitet fra, og som trolig gjør programmet mer tungkjørt enn det trenger å være. Det går an å bruke Nichestacken i en såkalt «superloop» uten RTOS'et [14]. Dette kan være raskere, men er enda ikke støttet av Altera.

Mottaksprogrammet er skrevet i Java, da vi var mest kjent med GUI funksjonene i dette programmeringsspråket. Vi vurderte å lære oss «Qt» rammeverket til dette formålet, og dermed skrive mottaksprogrammet i C/C++. På grunn av tidsbegrensninger gjorde vi ikke dette. Som beskrevet tidligere har forskjellene på mikronene gjort at vi måtte korrigere amplituden til signalene i programvare. Dette er en tilleggsfunksjon til mottaksprogrammet, og må utføres på den genererte filen etter dataoverføringen er ferdig. Java er ikke spesielt effektiv på formatering av data på bit-nivå, så å utføre dette i sanntid under mottak ville ikke vært mulig i dagens program. Dette ville tatt for lang tid, og programmet ville stagnert. Et forbedringspunkt hadde dermed vært å implementert mottaksprogrammet og korrigeringsfunksjonen i C/C++. Prototyping har vist at dette ville økt hastigheten på korrigeringsfunksjonen betraktelig.

7 Konklusjon

Hovedmålet med prosjektet var å lage en hardwareprototype som overfører digitaliserte signaler fra åtte mikrofoner til en ekstern PC, samt et mottaksprogram for lagring. Avlesningen fra de åtte mikrofonene skulle være synkron. Systemet skulle heller ikke innføre noen forskyvninger mellom kanalene etter digitalisering. Tester viser at alle kanalene er synkrone. Tidsforskjellen mellom kanalene er mindre enn ett sampel.

Overføringen av data fra mikrofonene gjennom systemet og ut til ekstern PC fungerer også tilfredsstillende. Under testene har systemet kjørt uten tap av data, fra inngangpinner på FPGA-en til filen som lagres på mottaker-PC. Vi har blant annet kjørt en langttidstest av systemet i 40 timer hvor vi har logget antall pakketap. I løpet av denne perioden mistet vi ingen datapakker.

FPGA-systemet vi har laget fungerer som det skal og oppfyller alle krav. Det gjør mottaksprogrammet også. Vi synes begge deler er brukervennlige – og opptak kan gjøres av en bruker uten inngående kjennskap til systemet.

Mottaksprogrammet fungerer også og vi har prioritert brukervennlighet. Algoritmen som etterkorrigerer amplitudene bruker lengre tid enn ønskelig, dette har sammenheng med programmeringsspråket som blir brukt. En forbedring vil være å skrive mottaksprogrammet i C++. FPGA-systemet kan utvides til å takle 16 kanaler med bare små modifikasjoner av mottaksmodulen, det vil si å gå ned til 16 bits oppløsning.

AD-kortet er for stort i forhold til kravet om fysisk størrelse. Vurderingen av om vi skulle få AD-kortet minst mulig, eller om vi skulle nedprioritere kravet om fysisk størrelse ble en avveining mellom viktigheten av fysisk størrelse og produksjonspris. På grunn høyere pris på lodding av kort med komponenter både på topp- og bunnplan, ble kravet om fysisk størrelse nedprioritert. Denne avgjørelsen hadde ingen direkte innvirkning på funksjonaliteten til kortet, men resulterte i at vi bare plasserte komponenter på topplanet.

En målsetning som ikke har vært forankret i kravspesifikasjonen er et mål om å lage et system med best mulig signal-støy-forhold. På grunn av valgt spenningsforsyning har vi ikke oppnådd så gode resultater som vi kunne ønske oss. Det er nå støy i de ti laveste bitene. Vi gjorde et forsøk på å prøve og begrense støyen i fra spenningsforsyningene ved å lage et LC-filter. Filteret i seg selv fungerer, men vi har ikke klart å få knekkfrekvensen lav nok. Grunnen til det er at vi ikke tåler noe spenningsfall over filteret. En bedre løsning ville vært å forsyne AD-kortet med en høyere spenning enn 5 V, og legge inn egne spenningsregulatorer lokalt på AD-kortet. Vi tror dette ville gitt et bedre signal-støy-forhold.

Ved bruk av eksterne spenningsforsyninger har vi klart å oppnå et signal-støy-forhold hvor de åtte laveste bitene er støy. Dette viser at den største kilden til støy er spenningsforsyningene, ikke selve AD-kortet.

8 Referanser

[1] O'Hara, M., *EMC at component and PCB level.* Newnes, 1998.

[2] Karki, J., *Fully differentialamplifier design in high-speed data acquisition systems (slyt119).* Texas Instruments, 2005.

[3] *PCM4204 datasheet*, Texas Instruments, 2004.

[4] *THS4521 datasheet*, Texas Instruments, 2008.

[5] *Avalon Interface Specification.* Altera Corporation, August, 2010.

[6] *Triple-speed Ethernet Megacore.* Altera Corporation, Desember, 2010.

[7] *Using SDRAM on AT91SAM9 Microcontrollers.* Atmel, Mars, 2011, Sist besøkt 15.3.11. http://www.atmel.com/dyn/resources/prod_documents/doc6256.pdf

[8] *IS42S16320B SDRAM datablad.* ISSI, Desember, 2009.

[9] *TSE, nichestack tx from fixed location*, Alteraforum, Mars, 2011, Sist besøkt 15.3.11. http://www.alteraforum.com/forum/showthread.php?t=28379

[10] Kundert, K., *Power Supply Noise Reduction. Designer`s Guide Consulting.* Inc. Version 4, jan. 2004.

[11] Karki, J., *Input impedance matching with fully differential amplifiers (slyt310).* Texas Instruments, 2008.

[12] *Avalon switch fabric.* Altera Corporation, August 2005. Sist besøkt 17.03.11. http://www.cas.mcmaster.ca/~lawford/3TB4/ref/avalonswfabric.pdf

[13] *Nios II/f Core: Fast for Performance-Critical Applications,* Altera Corporation, Sist besøkt 17.03.11 http://www.altera.com/products/ip/processors/nios2/cores/fast/ni2-fast-core.html

[14] *Super Loop NicheStack*, AlteraWiki.com. Sist besøkt 17.03.11. http://www.alterawiki.com/wiki/%22Super_Loop%22_NicheStack

[15] *exm-avalon-mm*. Altera.com, Sist besøkt 18.03.11 http://www.altera.com/support/examples/nios2/exm-avalon-mm.html

[16] Karki, J., *Analysis of fully differential amplifiers (slyt157).* Texas Instruments, 2000.

[17] Ott H. W., *Grounding of mixed signal systems.*, mars 2004.

[18] Mitzner, K., *Complete PCB design using OrCAD capture and PCB editor.* Newnes, 2010

[19] Texas instruments online kalkulator, Sist besøkt 27.04.11 http://www-k.ext.ti.com/srvs/cgi-bin/webcgi.exe?Company={5761bcd8-11f5-4e08-84e0-8167176a4ed9},kb=analog,case=obj%2812223%29,new

[20] *NIOS II Software Developer's Handbook.* Altera Corporation, Juli, 2010. [Nios2 software developers handbook.pdf]

[21] Karki, J., *Fully-Differential Amplifiers (SLOA054).* Texas Instruments, september 2000.

[22] Floyd, T. L., *Electronic Devices,* 8th ed., Pearson education, 2008.

[23] *Josephson C550.*, Josephson Engineering.

[24] *Sopc builder: User guide.* Altera Corporation, Desember, 2010.

[25] *Avalon bus specification, reference manual*, Altera Corporation Januar 2003.

[26] *Nios II Processor Reference Handbook*. Altera Corporation Mai 2011.

[27] *µC/OS-II Kernel.* Micriµm, Inc. http://www.micrium.com/page/products/rtos/os-ii . Sist besøkt 25.05.11.

[28] *Ethernet and the NicheStack TCP/IP Stack - Nios II Edition.* Altera Corporation, Mai, 2011.

[29] *Scatter-Gather DMA Controller Core.* Altera Corporation, Mai, 2007.

[30] *On-Chip FIFO memory core.* Altera Corporation, November, 2009.

[31] *Nios DMA.* Altera Corporation, 2003.

[32] Prosjektgruppe 2011-09, *S.5 Mottaksprogram*, MMS, 2011.

[33] Prosjektgruppe 2011-09, *S.2 Brukermanual,* MMS, 2011.

[34] Prosjektgruppe 2011-09, *P.3 Kravspesifikasjon*, MMS, 2011.

[35] *CFPS-39 datasheet.* IQD engineering, 2010.

[36] Prosjektgruppe 2011-09, *S.4 Aktivitetsrapporter FPGA,* MMS, 2011.

[37] *I2S to parallel interface.* OpenCores, http://opencores.org/project,i2s_to_parallel . Sist besøkt 25.05.2011.

[38] Prosjektgruppe 2011-09, *T.4 Test av AD-kort*, MMS, 2011.

[39] Prosjektgruppe 2011-09, *T.2 Test av spenningsforsyning*, MMS, 2011.

[40] Prosjektgruppe 2011-09, *T.6 Test ytelse system*, MMS, 2011.

[41] Prosjektgruppe 2011-09, *T.5 Test FPGA/programvare* , MMS, 2011.

[42] Prosjektgruppe 2011-09, *T.3 Systemtest*, MMS, 2011.

[43] Prosjektgruppe 2011-09, S.7 *Mottaksmodul*, MMS, 2011.

[44] *Kompileringsrapport-Metastability Report,* Quartus, MMS, 2011.

[45] *Kompileringsrapport_Resource Utilization by Entity,* Quartus, MMS, 2011.

[46] Prosjektgruppe 2011-09, S.6 *Applikasjonsprogram NIOS*, MMS, 2011.

[47] *DE2-115 user manual*, Altera Corporation, 2010.

[48] Gudvangen Sigmund, *Introduction to delta-sigma converters*, HiBu 2010

[49] Prosjektgruppe 2011-09, T.1 *Simuleringsrapporter*, MMS, 2011.

9 Vedlegg

9.1 Vedlegg I – Matlabprogram for visualisering

```
%Load and plot binary files
%Projectgroup 2011-09 MMS
%HiBu Kongsberg

clc
clear all
close all

sek = 1; %antall sek opptak.
fs = 96000; % samplingfrekvens.

file = 'filnavn.bin'; %filnavn
fid = fopen(file);

data = fread(fid,[8,1*sek*fs],'bit24', 0, 'l'); %b = big, l = little
fclose(fid);

%tidsdomene
figure(1)

subplot(4,2,1)
plot(data(1,:)), grid on
title('kanal J3'), xlabel('Samples'), ylabel('Amplitude')

subplot(4,2,2)
plot(data(2,:)),grid on
title('kanal J4'), xlabel('Samples'), ylabel('Amplitude')

subplot(4,2,3)
plot(data(3,:)),grid on
title('kanal J6'), xlabel('Samples'), ylabel('Amplitude')

subplot(4,2,4)
plot(data(4,:)),grid on
title('kanal J7'), xlabel('Samples'), ylabel('Amplitude')

subplot(4,2,5)
plot(data(5,:)), grid on
title('kanal J8'), xlabel('Samples'), ylabel('Amplitude')

subplot(4,2,6)
plot(data(6,:)),grid on
title('kanal J10'), xlabel('Samples'), ylabel('Amplitude')

subplot(4,2,7)
plot(data(7,:)),grid on
title('kanal J12'), xlabel('Samples'), ylabel('Amplitude')

subplot(4,2,8)
plot(data(8,:)),grid on
title('kanal J14'), xlabel('Samples'), ylabel('Amplitude')
```

9.2 Vedlegg II – Fullstendig kretsskjema

De neste fem sidene viser det fullstendige kretsskjema for AD-kortet.

POSICOM

HØGSKOLEN i Buskerud

Mottaksprogram		
Dokumenttype	Dokumentnummer	Versjonsnummer
Systembeskrivelse	S.5	1.a
Dato sist endret	Frigitt dokument?	Dokumentansvarlig
27.05.2011	Ja	Erlend
Gruppemedlemmer:		
Kathrine Kaasa, Knut Olav Skyttemyr, Jardar Hvidsten, Fredrik Bakke, Erlend Rauk Aasland		

Innholdsliste

1 Introduksjon....3
2 Mottaksprogram....4
2.1 Klassene....6
2.2 Scenario: Innstillinger....7
2.3 Scenario: Start....9
2.4 Scenario: Stopp....11
2.5 Scenario: Korrigere amplitude....12

Illustrasjonsliste

Use case diagram....4
Klassediagram mottaksprogram....5
Grafisk brukergrensesnitt....6
Sekvensdiagram for innstillinger....8
Sekvensdiagram for start....10
Sekvensdiagram for stopp....11
Grensesnitt for korrigering av amplitude....12
Sekvensdiagram for korrigere amplitude....13

Forkortelser og akronymer

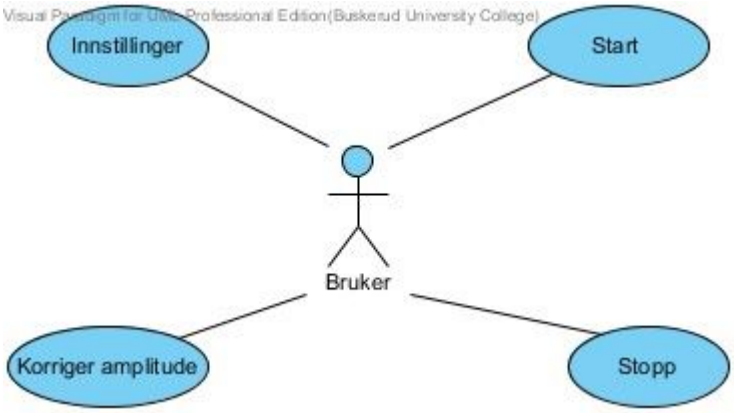
UML	-	Unified Modelling Language
FPGA	-	Field Programmable Gate Array
AD	-	Analog-til-digital
GUI	-	Graphical User Interface
UDP	-	User Datagram Protocol
FIFO	-	First In, First Out

1 Introduksjon

Mottaksprogrammet har som hensikt å styre dataoverføringen mellom FPGA (Field Programmable Gate Array) og mottaks-PC.
Dette dokumentet er en teknisk og funksjonell beskrivelse av programmet, og UML-diagrammer (Unified Modelling Language) vil bli vist som en visuell beskrivelse.

2 Mottaksprogram

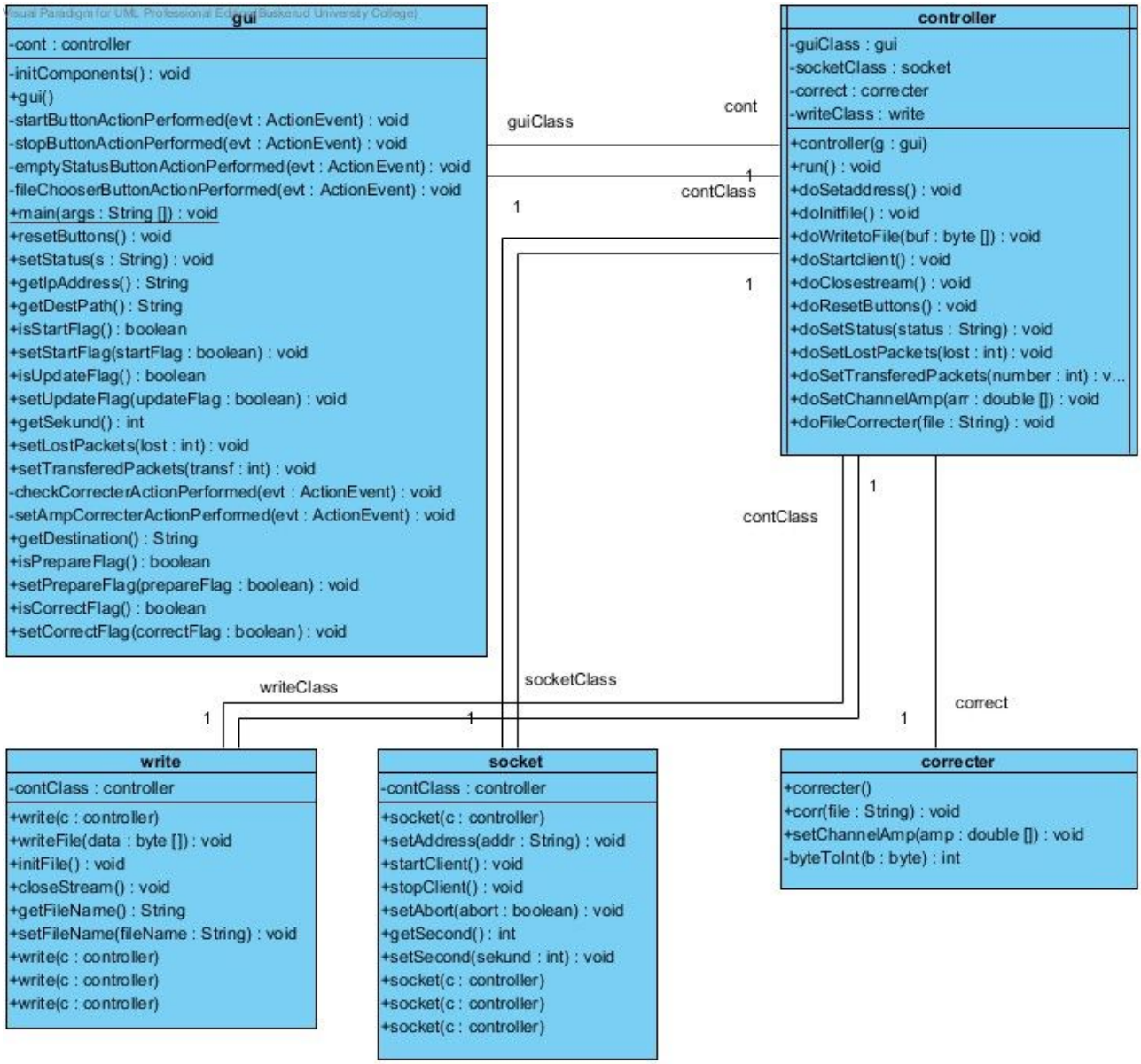
Målet med mottaksprogrammet var å lage et brukervennlig program som kan styre datastrømmen fra AD-kortet (Analog-til-digital), samtidig som det skal kunne lagre data det mottar til fil. Datastrømmen skal kunne startes og stoppes av brukeren uten at hele systemet må restartes mellom hver gang. Vi valgte å lage dette programmet i Java, da det gjør det enkelt å lage et grafisk grensesnitt. Se Figur 1 for use-caser som bekriver ulike scenarioer i programmet.



Figur 1: Use case diagram

De ulike use-casene vil bli forklart nærmere i de neste kapitlene.

Figur 2 er et klassediagram over alle klassene i mottaksprogrammet.



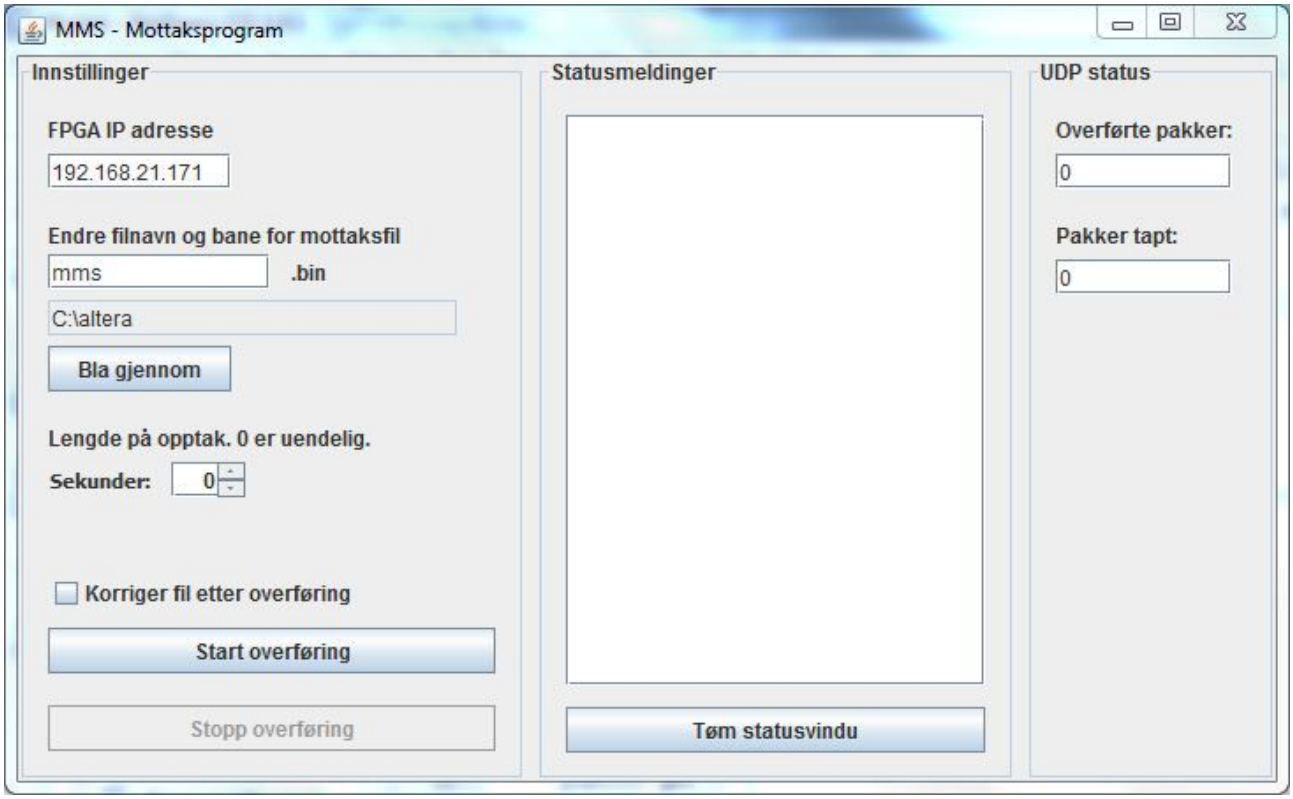
Figur 2: Klassediagram mottaksprogram

Ved oppstart av programmet vil et GUI-objekt (Graphical User Interface) starte opp og vise det grafiske grensesnittet. Et objekt av controller-klassen opprettes samtidig, denne har ansvar for kommunikasjonen mellom GUI og de andre klassene. For at GUI skal være mottagelig for interaksjon fra brukeren vil vi opprette controller-objektet som et aktivt objekt (kjøres i egen tråd). I controlleren opprettes objekter av write-, socket- og correcter-klassene.

2.1 Klassene

Som nevnt er controller-klassen et bindeledd mellom GUI og write-/socket-klassene. GUI og controller kjøres i hver sin tråd, dette for blant annet at brukeren fortsatt skal ha mulighet til å trykke på stopp-knappen selv om det skrives data til fil. Tråden i controller-klassen kjører i en evig løkke og sjekker om brukeren har trykket start-knappen. Om brukeren trykker stopp-knappen vil «doClosestream» metoden kjøres. Socket- og writeklassene representerer de to viktigste delene av programmet. Socket-klassen oppretter socketkommunikasjon mellom mottaker-PC og FPGA-kortet. Det er i denne klassen data mottas og sendes til FPGA-kortet.

Write-klassen brukes når data som er mottatt skal skrives til fil. Den oppretter da filen som er spesifisert av brukeren i GUI-vinduet, eller overskriver dersom den finnes fra før. Før hver datapakke sendes ut fra FPGA-kortet legges det til et sekvensnummer. Dette bruker vi for å kontroller antall pakker som mistes underveis. Før data skrives til fil, sjekkes derfor dette nummeret opp mot forrige nummer. Slik vet vi om det har oppstått et pakketap.



Figur 3: Grafisk brukergrensesnitt

I de neste kapitlene vil vi presentere tre scenarioer som vil beskrive all funksjonalitet i mottaksprogrammet.

2.2 Scenario: Innstillinger

Det første brukeren må gjøre etter oppstart av programmet er å endre filnavn og bane for filen som data skal lagres i. Filbanen endres ved å trykke på «bla gjennom» knappen og et filutforskervindu dukker opp. IP-adressen til FPGA-kortet vil sjelden endres, så denne kan stort sett forbli uendret. Brukeren kan også stille inn antall sekunder et opptak skal bestå av. Dersom amplituden på signalet fra en eller flere mikrofoner må korrigeres trykkes knappen for dette. Denne funksjonen beskrives nærmere i kapittel 4.5.

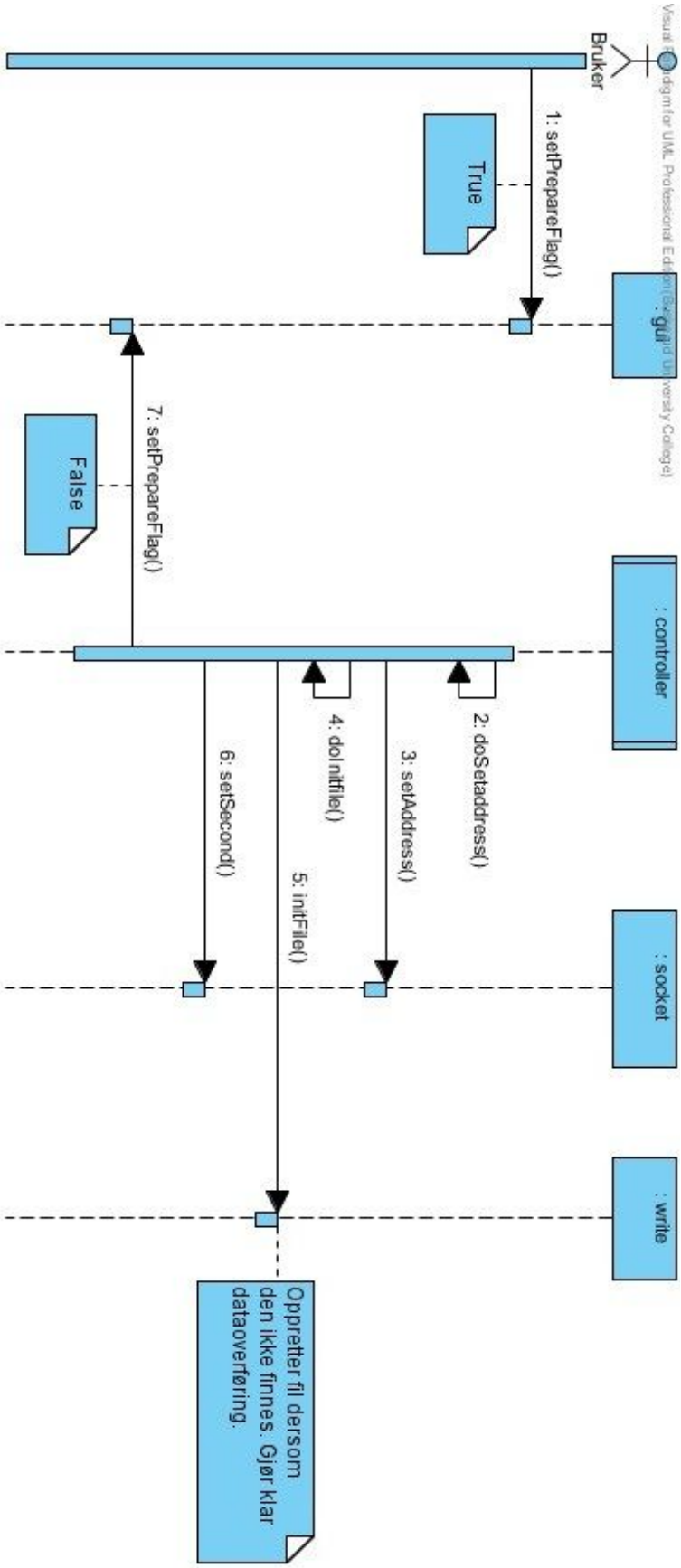
Når brukeren er klar kan start-knappen trykkes, og et prepare-flagg blir satt. Dette blir oppdaget av controller-klassen, og følgende metoder kjøres.

DoSetaddress: Setter IP-adressen spesifisert av brukeren. Denne adressen brukes når en UDP pakke skal sendes til FPGA-kortet.

DoInitfile: Oppretter filen spesifisert av brukeren.

SetSeconds: Antall sekunder bruker har valgt for overføring av data vil bli satt i en variabel i socket klassen. Denne verdien brukes i scenario: start.

Til slutt vil prepare-flagget settes tilbake. Mottaksprogrammet er nå klar for å motta data. Et sekvensdiagram for dette scenarioet vises i Figur 4.



Figur 4: Sekvensdiagram for innstillinger

2.3 Scenario: Start

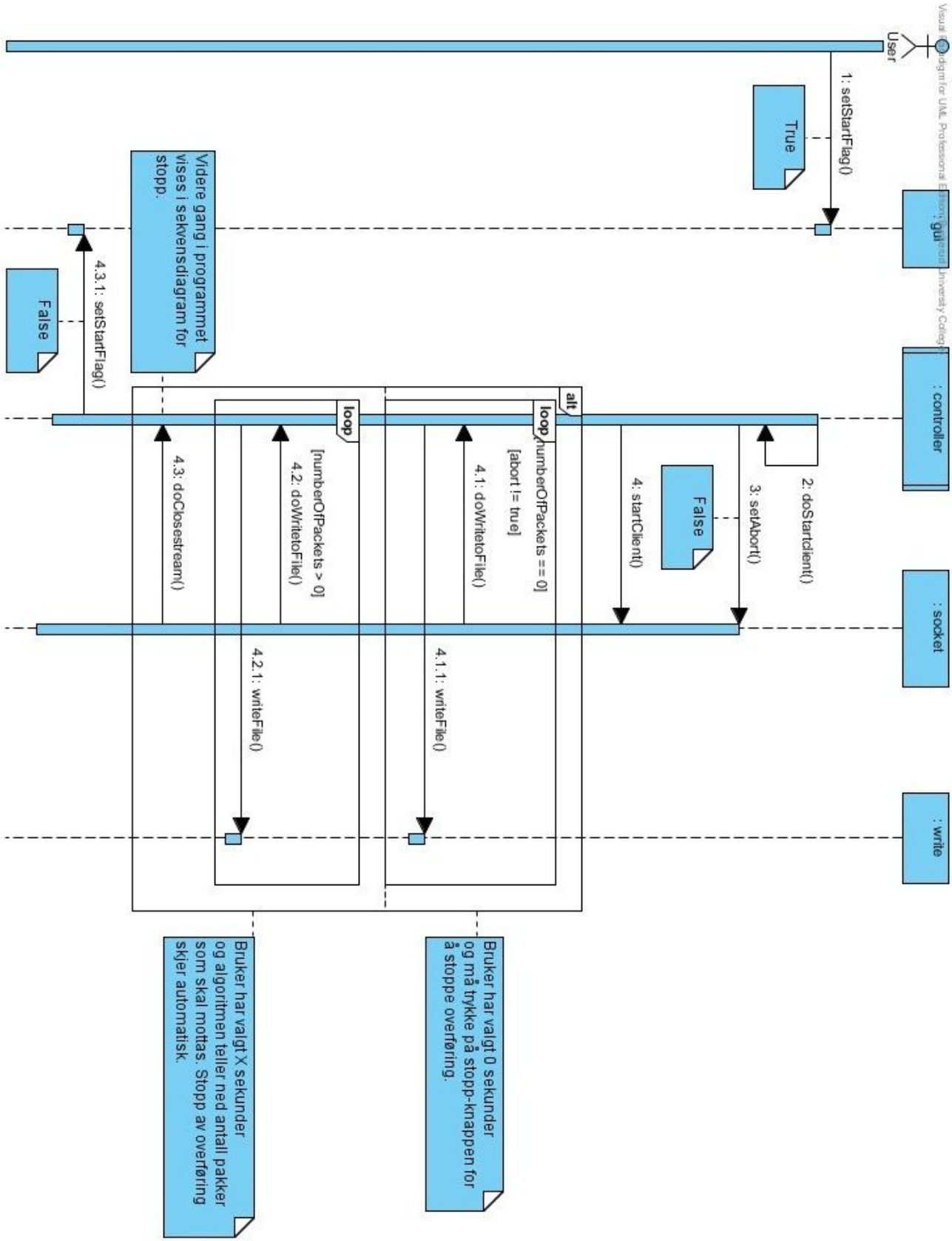
Når start-knappen har blitt trykt og innstillinger-sekvensen har blitt utført, vil et start-flagg blir satt. Dette blir oppdaget av controller-klassen, og følgende metoder kjøres.

DoStartClient: Denne metoden setter først abort-flagget til «false». Som ikke blir satt til «true» før dataoverføringen skal stoppes. Deretter kjøres selve startClient metoden i socket-klassen. Her blir selve socket kommunikasjonen opprettet. Etter dette sjekkes det om brukeren har valgt et visst antall sekunder under innstillinger-scenarioet. Dersom brukeren har valgt 0 sekunder, betyr dette at data kan mottas helt til brukeren trykker på stopp-knappen. Dersom brukeren har valgt et tall større enn 0, vil antall pakker som skal mottas beregnes. Siden vi vet at ett sample består av 2304000 byte/sekund vil ett sekund bestå av ca. 2285 datapakker på størrelsen 1008 byte. Størrelsen på datapakkene går ikke nøyaktig opp i ett sekund, vi fant det viktigere at størrelsen på datapakkene som overføres er delelig på antall bits per kanal (24 bit). Dette vil ikke ha noe innvirkning på en visualisert visning av filen.

For at NIOS-prosessoren på FPGA-kortet skal sette igang overføring av data må den motta en startpakke. Denne pakken består av en byte med verdien '1' eller '0'. Verdien '0' tolkes hos FPGA-kortet at den skal begynne dataoverføringen, og den stopper ved verdien '1'. På dette tidspunkt i koden vil vi at overføringen skal starte og setter derfor verdien til '0'. Dersom brukeren valgte 0 sekunder vil overføringen av data kun avbrytes av abort-flagget. Dersom brukeren valgte et visst antall sekunder vil overføringen avbrytes når variabelen med antall pakker har dekrementert til 0. Mottaksprogrammet mottar nå data kontinuerlig og sender overført data til doWritetoFile metoden.

DoWritetoFile: Siden all kommunikasjon mellom klassene skjer i controller-klassen vil data som skal skrives til fil gå gjennom controller-klassen til write-klassen. For hver gang en ny UDP-pakke (User Datagram Protocol) med data er mottatt av programmet blir denne metoden kalt. Derfor er det naturlig å sjekke sekvensnummeret til UDP-pakken her. Sekvensnummereringen er tall fra 0 til 254. Forrige og nåværende sekvensnummer blir sjekket opp mot hverandre. Dersom nåværende tall ikke er lik forrige nummer + 1 vil det ha oppstått pakketap og dette blir registrert og vist for brukeren. Når denne beregningen er gjort vil data skrives til fil.

Til slutt vil start-flagget settes tilbake og datastrømmen kan stanses. Sekvensdiagram for startscenarioet vises i Figur 5.



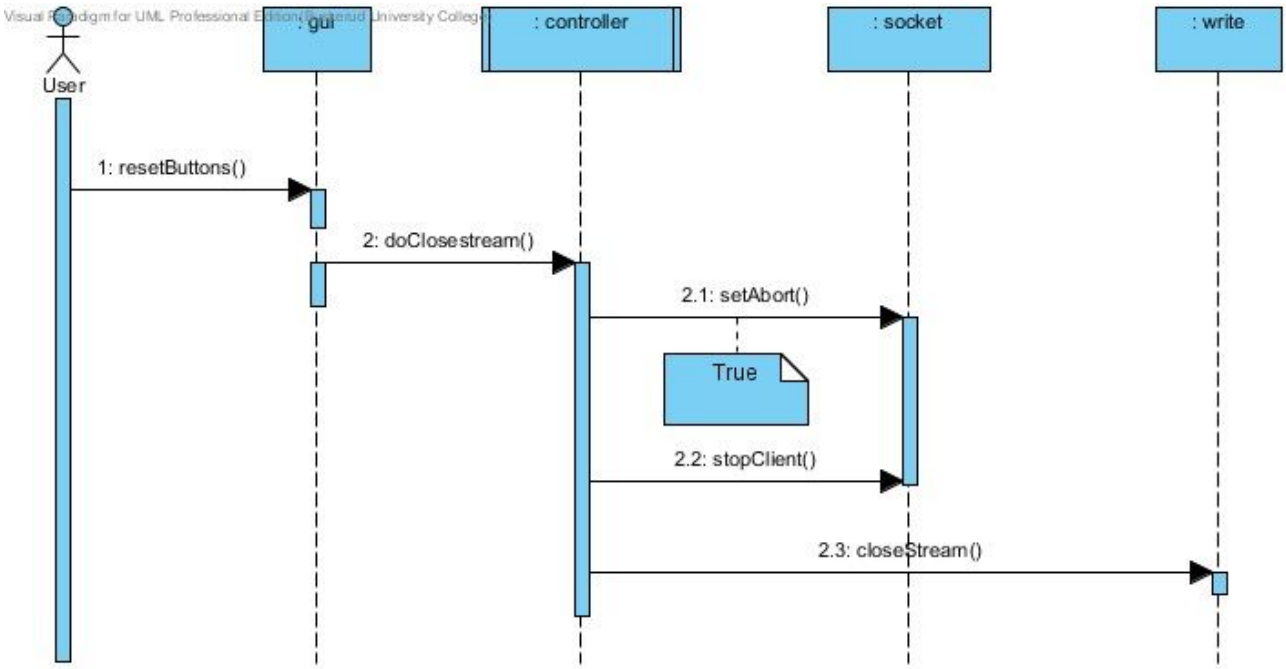
Figur 5: Sekvensdiagram for start

2.4 Scenario: Stopp

Stopp scenarioet tiltrer når brukeren ønsker å avslutte dataoverføringen med stopp-knappen, eller antall sekunder med byte er overført og programmet stanser overføringen automatisk. Uansett hvilken måte dataoverføringen stopper på skjer dette ved et kall til «doClosestream» metoden i controller-klassen.

DoClosestream: Først settes abort-flagget til «true», slik at overføringen som er avhengig av denne avbrytes. Deretter kjøres «stopClient» metoden i socket-klassen. Her settes startpakken, som tidligere beskrevet i doStartclient-metoden, til verdien '1' og sendes over til FPGA-kortet. Dette tolkes i NIOS-prosessoren at FIFO-bufferet (First In, First Out) skal slutte å fylles opp og at det skal leses til det er tomt, slik at ikke gammel data blir overført ved neste dataoverføring. Samtidig kjøres «closeStream» metoden i write-klassen for å frigi ressursen som holder på datafilen.

Se Figur 6 for sekvensdiagram for dette scenarioet.



Figur 6: Sekvensdiagram for stopp

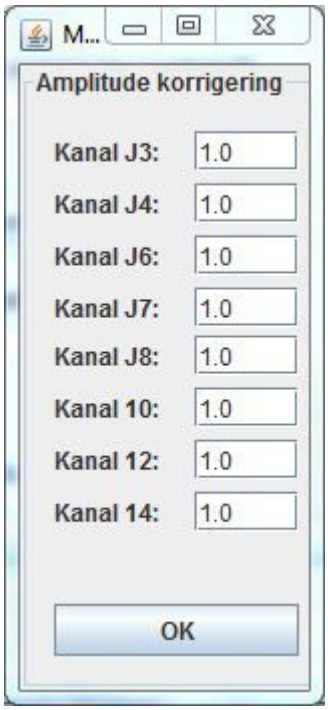
2.5 Scenario: Korrigere amplitude

På grunn av forskjeller på mikrofonene som er koblet til systemet, har vi gitt brukeren muligheten til å korrigere amplituden for hver kanal individuelt. Dette gjøres ved å huke av «korriger fil etter overføring» før overføringen starter. Brukeren vil da få presentert vinduet i Figur 7, hvor han/hun kan gjøre de nødvendige justeringer.

SetChannelAmp(): Verdiene som blir satt i Figur 7 blir plassert i et array med 8 plasser. Plasseringen til verdiene i arrayet bestemmer hvilken kanal den tilhører. Array[0] tilhører kanal J3 og Array[7] tilhører kanal J14. Dette arrayet blir sendt til correcter-klassen.

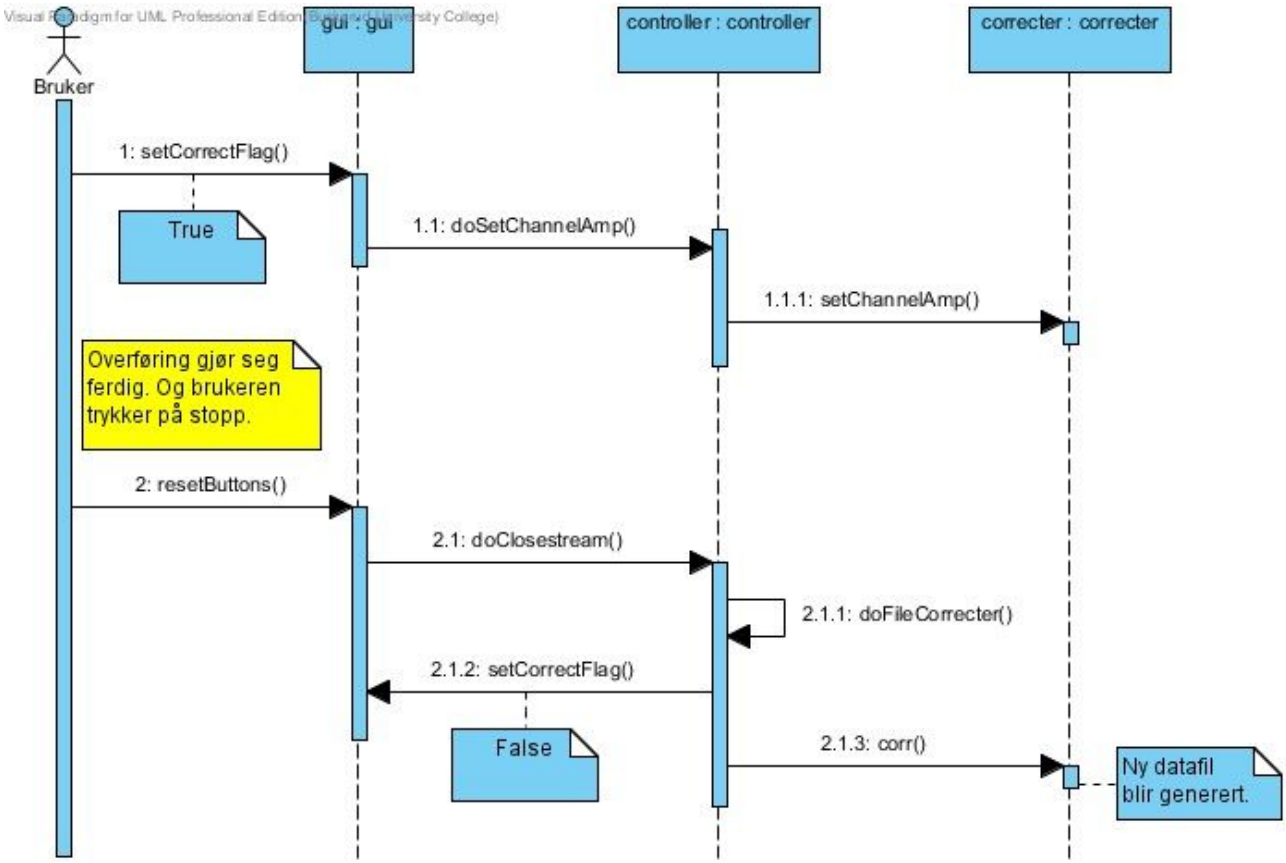
Corr(): Etter hver overføring som blir gjort mens «korriger fil etter overføring» er huket av – vil det kjøres en algoritme som går igjennom den lagrede filen og lager en kopi med justerte amplituder. Denne algoritmen bruker typisk ≈15 sekunder per sekund med lyddata.

Algoritmen er i correcter-klassen. Den leser maks 240 byte av filen av gangen – for å ikke bruke for mye RAM. Siden vi ikke har noen datatype på 24 bit tilgjengelig – shiftes de tre bytene som hører til et sample inn i et 32 bits heltall. Nødvendige justeringer gjøres for at ikke java sin casting skal korrumpere dataene. Siden utføres multiplikasjonen på heltallet, og det shiftes ut til ny fil.



Figur 7: Grensesnitt for korrigering av amplitude

Som beskrevet, må korrigering av fil aktiveres før dataoverføringen skjer. Selve korrigeringen kjøres etter dataoverføringen er ferdig. Derfor er sekvensdiagrammet delt opp i to «sekvenser». Først aktiveres korrigering, og korrigering settes igang når brukeren avslutter dataoverføring. Se sekvensdiagram i Figur 8.



Figur 8: Sekvensdiagram for korrigere amplitude

POSICOM

HØGSKOLEN i Buskerud

Applikasjonsprogram, NIOS II		
Dokumenttype	Dokumentnummer	Versjonsnummer
Systembeskrivelse	S.6	1.a
Dato sist endret	Frigitt dokument?	Dokumentansvarlig
19.05.2011	Ja	Erlend Rauk Aasland
Gruppemedlemmer:		
Kathrine Kaasa, Knut Olav Skyttemyr, Jardar Hvidsten, Fredrik Bakke, Erlend Rauk Aasland		

Innholdsliste

1 Introduksjon....3
2 Oversikt....4
3 Funksjoner....5
3.1 Main....5
3.2 Socket og receive....5

Figurliste

Diagram over C-filer....4
Sekvensdiagram for Applikasjonsprogram....7

Forkortelser og akronymer

MicroC/OS II	-	Operativsystemet som applikasjonsprogrammet kjører på
DMA	-	Direct Memory Access
FIFO	-	First In, First Out
FPGA	-	Field Programmable Gate Array
UDP	-	User Datagram Protocol
UML	-	Unified Modelling Language

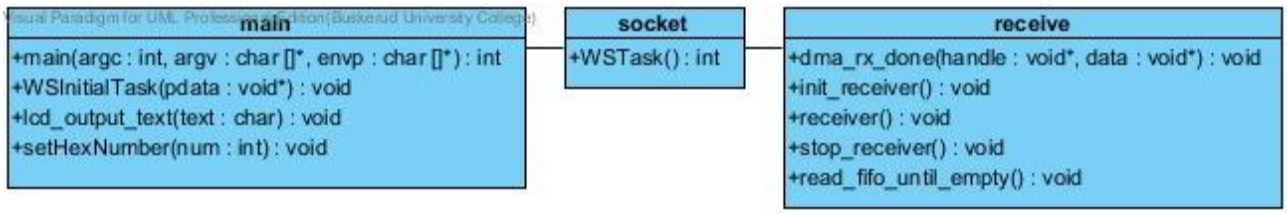
1 Introduksjon

Dette dokumentet tar for seg programvaren som vi har endret og utviklet i NIOSII-prosessoren på FPGA-kortet (Field Programmable Gate Array). Dokumentet vil gi en funksjonsbeskrivelse av hvordan programvaren fungerer. UML-diagrammer (Unified Modelling Language) er vedlagt.

2 Oversikt

Programvaren som kjøres på NIOSII-prosessoren var i utgangspunktet skrevet som en webserver. Det betød at vi fikk mye ferdig kode, og at vi kunne strippe vekk unødvendig funksjonalitet og beholde det viktige. Det viktige for oss i denne sammenheng var kode som starter opp operativsystemet MicroC/OS II og Nichestacken. MicroC/OS II er nødvendig for å bruke Nichestacken, og denne er vi avhengig av for å benytte oss av Ethernet-, UDP- og IP-protokollene (Int). MicroC/OS II er et sanntidsoperativsystem og oppretter oppgavene sine som egne «tasker». Disse kjører i hver sin uendelige løkke med hver sin prioritet.

For at operativsystemet og prosessoren skulle ha minst mulig å gjøre bortsett fra å oversende data til mottaker-PC, fjernet vi alle oppgavene bortsett fra «WSInitialTask» i filen *main.c*. I denne oppgaven starter Nichestacken, oppretter en ny oppgave «WSTask», og «WSInitialTask» sletter seg selv. Fra denne stund er det kun «WSTask» som er aktiv oppgave i operativsystemet. «WSTask» befinner seg i filen *socket.c*. Vi har fjernet flere filer, og de som er igjen er avgjørende for funksjonaliteten vi behøver. Disse filene inneholder mye kode skrevet av Altera, og vi har utført endringer tilpasset vårt behov der det var nødvendig. Dette gjelder spesielt *socket.c*. *Receive.c* har vi skrevet selv. Figur 1 viser et diagram over de viktigste kodefilene i applikasjonsprogrammet. Det er kun disse filene vi har endret og brukt underveis i utviklingen.



Figur 1: Diagram over C-filer.

3 Funksjoner

Vi vil nå beskrive funksjonaliteten til programmet. Som nevnt er det kun de tre kodefilene i diagrammet over som er viktige i denne sammenheng. Det vil si hvordan NIOS styrer dataoverføringen helt fra FIFO (First In, First Out) til Ethernet.

3.1 Main

I main funksjonen opprettes det en oppgave som operativsystemet, MicroC/OS II, kan utføre. Denne oppgaven kalles «WSInitialTask» og må opprettes før operativsystemet starter. Når denne er opprettet kan operativsystemet starte og oppgaven utføres. Programflyten flytter seg derfor til «WSInitialTask» og Nichestack TCP/IP stakken, herunder UDP, vil initialiseres. Når dette er utført vil en ny oppgave for operativsystemet opprettes og gjeldende oppgave vil slette seg selv, da den ikke skal brukes mer. Denne oppgaven kjøres og programflyten endrer seg til «WSTask» i socket.

3.2 Socket og receive

I dette kapittelet beskriver vi både socket- og receive-filene samtidig, da socket bruker receiver underveis i programflyten. Hensikten med disse to er å la DMA-kontrolleren (Direct Memory Access) hente data fra FIFO-bufferet, plassere dette i minnet slik at dataene kan sendes over Ethernet til mottaker-PC-en. Samtidig skal de kunne ta i mot stopp- og start-bit fra mottaker-PC. For å kunne styre hardware-modulene (mottaksmodul, linecheck og signalgenerator) fra NIOSII, har vi koblet om signaler som før gikk til IO-pinner (In Out) – til hardware-moduler på FPGA:

Tabell 1: Signaler koblet til IO-innganger

Signal på FPGA	Navn i NIOS	Forklaring
reset_from_nios	LEDG0	Reset for mottaksmodul og AD-signalgenerator.
stop_filling_fifo	LEDG1	Koblet til stopp signalet på mottaksmodul. Avbryter fylling av FIFO.
reset_line_check_from_nios	LEDG2	Reset signal til linjesjekker. Brukes for å slå av lampene ved oppstart.
For å bruke disse signalene sender vi 1 eller 0 til tilhørende LEDR inngang.		

Det første som skjer i WSTask er et kall til «init_receiver» funksjonen. Her opprettes en DMA-kanal som brukes for å hente data fra FIFO-bufferet. I tillegg initialiseres FIFO-bufferet med korrekt minneadresse. Deretter opprettes en socket deskriptor, som gjør at vi kan sende/motta data over

Ethernet. Socket-deskriptoren assosieres så med IP-adresse og portnummer slik at andre Ethernet-enheter kan koble seg til denne socketen.

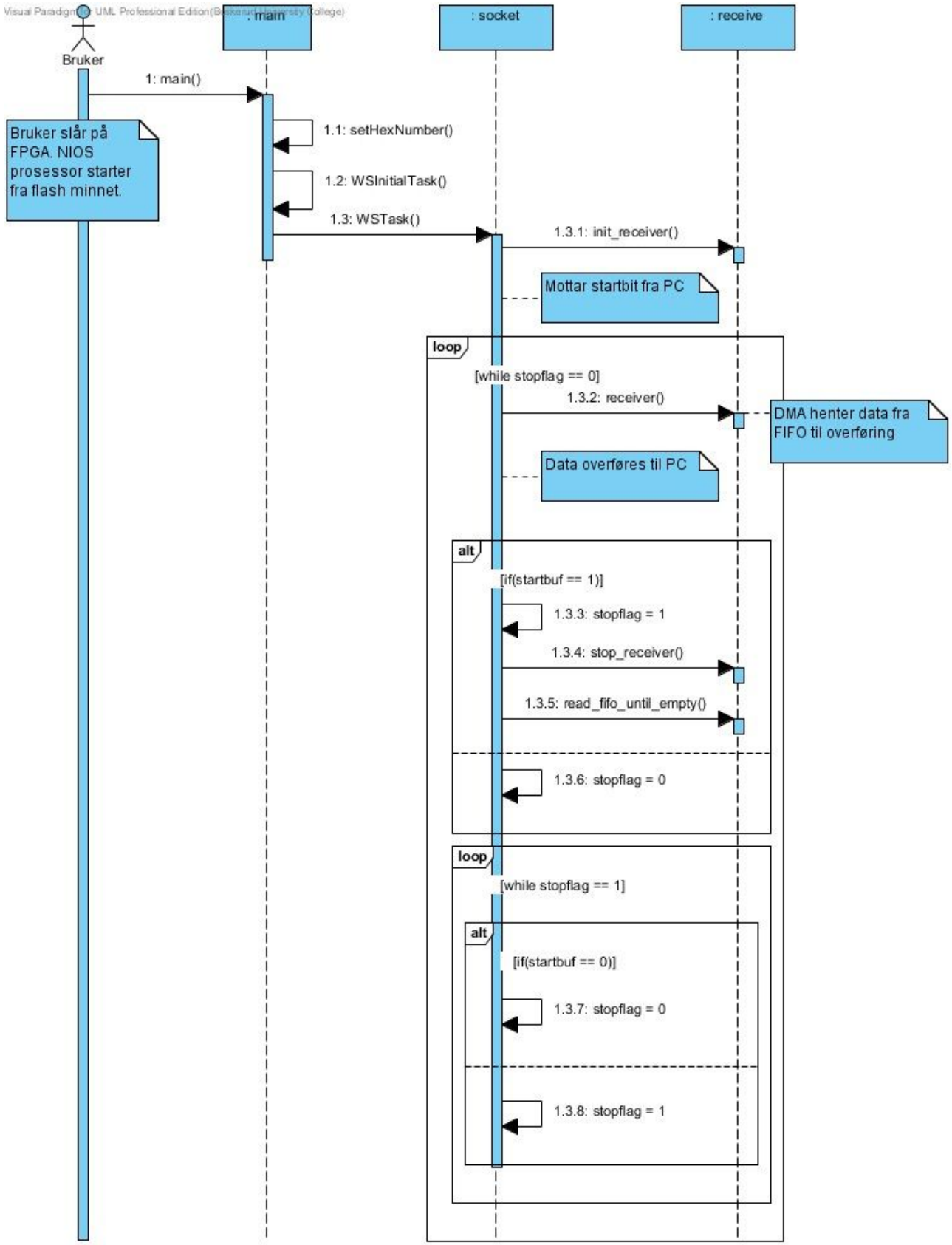
Nå er FPGA-kortet og NIOSII klar for å sende data til mottaker-PC, og programmet venter nå på å motta startbit fra mottaker-PC. Sending av data til mottaker-PC er avhengig av et flagg, «stopFlag». Om dette flagget har verdien '0' kan data sendes til mottaker-PC. Dette pågår helt til det mottas et stoppbit. Hvis stoppbit er mottatt settes stopFlag til '1'. Da skal programflyten vente til et nytt startbit mottas. Se tabell nedenfor og sekvensdiagram i Figur 2.

Tabell 2: Start og stopp signal

Mottatt bit:	StopFlag settes til:	Forklaring
'1' = startbit	'1'	Dataoverføring stanses. Programflyten lytter nå etter mottak av startbit.
'0' = stoppbit	'0'	Dataoverføring kan nå starte. Mottaker-PC er klar.

Når stopFlagget har verdien '0' sendes data kontinuerlig i en løkke. Hver gang løkken begynner kjøres et funksjonskall til «receiver». Her starter DMA med dataoverføring fra FIFO-bufferet og plasserer det i SDRAM-minnet. Dette vil pågå helt til DMA-kontrolleren er ferdig. Når korrekt mengde data er plassert i minnet legger vi til et ekstra bit med et sekvensnummer på databufferet som skal oversendes. Slik kan vi sjekke på mottaker-PC om det har oppstått pakketap.

Slik går programflyten i en løkke og dataoverføringen styres fullt og helt fra mottaker-PC-en.



Figur 2: Sekvensdiagram for Applikasjonsprogram

POSICOM

HØGSKOLEN i Buskerud

Modulbeskrivelse: Mottak.		
Dokumenttype	Dokumentnummer	Versjonsnummer
Systembeskrivelse	S.7	0.a
Dato sist endret	Frigitt dokument?	Dokumentansvarlig
27.05.2011	Nei	Fredrik Bakke
Gruppemedlemmer:		
Kathrine Kaasa, Knut Olav Skyttemyr, Jardar Hvidsten, Fredrik Bakke, Erlend Rauk Aasland		

Innholdsliste

1 Introduksjon....3
2 Om modulen....4
3 AD-grensesnitt....5
4 Oppbygning....6
5 Referanser....10

Figurliste

Fra 24 bits sampler på I2S-buss, til 32 bits word i systemet....4
ADC – timingdiagram [2]....5
Blokkdiagram for mottaksmodul, AMT-en er en del av SOPC modulen....6
Tilstandsdiagram, I2S2P....7
Tilstandsdiagram, Receiver....8

Forkortelser og akronymer

FPGA	-	Field Programmable Gate Array
FSMD	-	Finite State Machine with Datapath
AMT	-	Avalon Master Template, for kommunikasjon over systembuss
LRCK	-	Ordklokke fra ADC-er
WS	-	Navnet til LRCK i I^2S-spesifikasjonen
BCK	-	Bitklokke fra ADC-er
FIFO	-	First In, First Out-mellomlager
I^2S	-	Inter-IC sound (Integrated Interchip Sound)
AD	-	Analog til digital

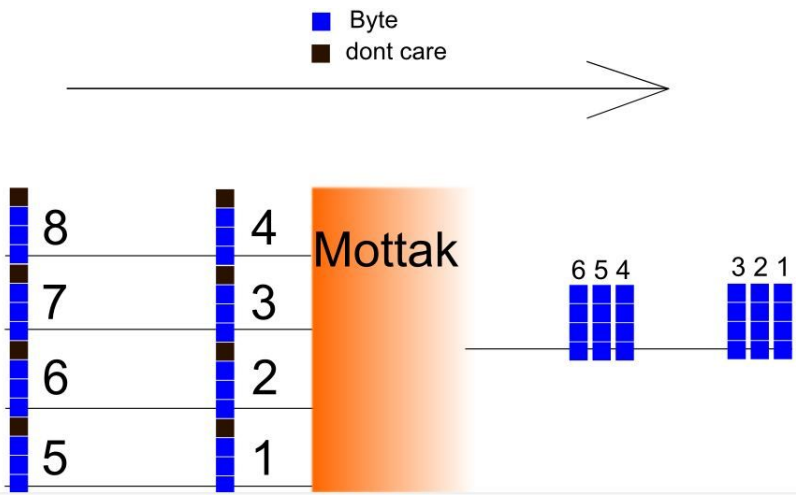
1 Introduksjon

Dette dokumentet er en teknisk beskrivelse av mottaksmodulen på FPGA-en.

2 Om modulen

I²S-spesifikasjonen (Inter-IC Sound) er ikke entydig, derfor har vi laget den utifra spesifikasjonene i databladet til ADC-en (Analog-to-digital converter) [2] som vi bruker. Fra AD-kortet (Analog til Digital) vil lyddata komme inn på FPGA-en over fire stk I²S-datalinjer med felles bitklokke BCK (≈6,1 MHz) og Left-Right-klokke, LRCK. To lydkanaler kommer inn over hver datalinje [3]. Mottakslogikken tar i mot data som er synkronisert mot BCK. Vi har valgt å klokke den på BCK. For å sende data videre til FIFO-bufferet (First In, First Out) har vi valgt å bruke en modul fra Altera som kommuniserer med systembussen [1]. Dette forenkler kontrollsignalene vi må generere.

For ikke påføre systemet ekstra belastning velger vi å slå sammen 4 - 24 bits sampler til 3 - 32 bits word allerede i mottaks-modulen. Dette kan vi gjøre fordi vi får inn 4 byte for hver kanal, men bare 3 av de inneholder data. Vi har valgt å sende bytene i hver kanal i little-endian byte-rekkefølge både innad i hver kanal, og innad i hvert ord som sendes videre. De vanligste operativsystemene benytter denne rekkefølgen (Linux x86, Windows, OS-X, NIOSII).



Figur 1: Fra 24 bits sampler på I²S-buss, til 32 bits word i systemet

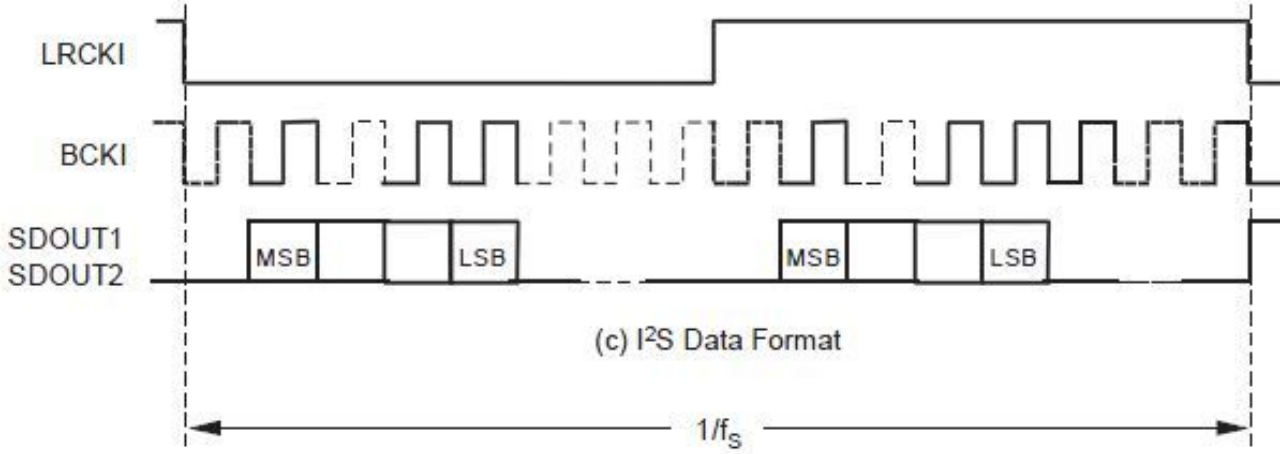
For å endre byterekkefølge innad i kanalene til big-endian, er kode til dette i kommentarer i VHDL-filen «receiver.vhd».

Under sammenslåingen av modulene vil vi ha mulighet til å pause mottaksmodulens fylling av FIFO-bufferet, via AMT (Avalon Master Template). Dette implementerte vi ved å gi mottaksmodulen et stopp-signal. Når dette signalet går høyt vil mottaksmodulen gjøre ferdig sampelet modulen holder på med, initiere tømming av AMT, for så og vente til stoppsignalet går lavt. Deretter synkroniserer den seg mot LRCK før den fortsetter som normalt.

3 AD-grensesnitt

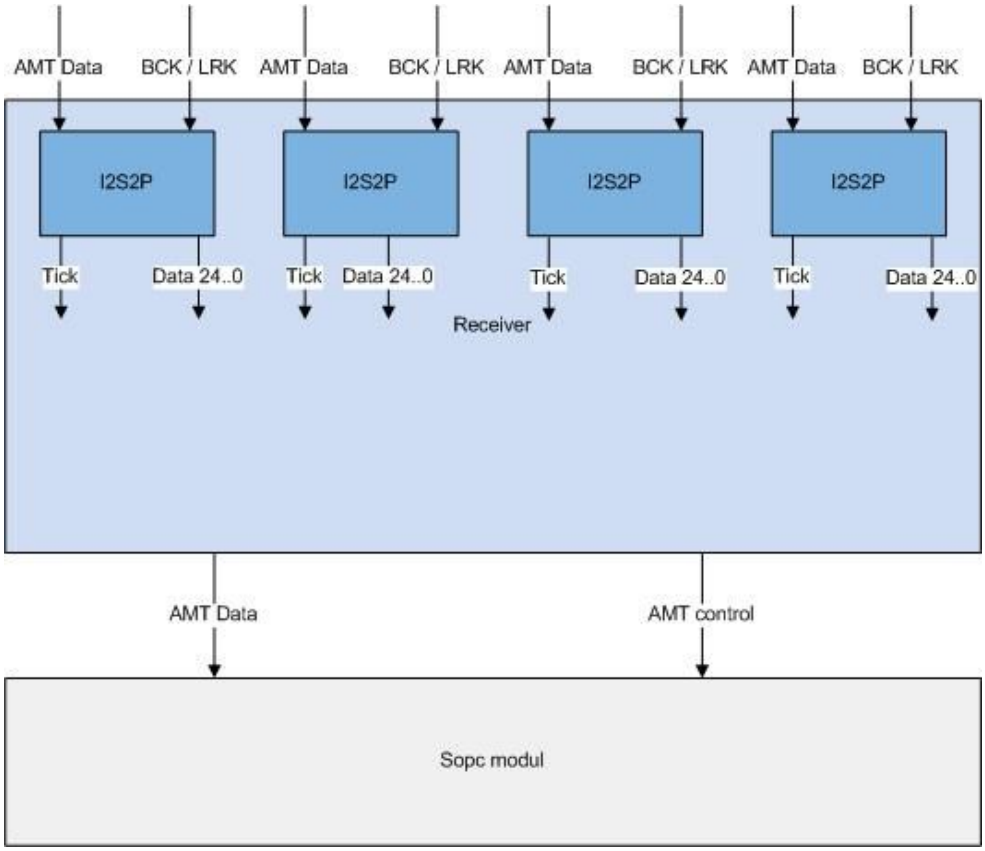
ADC-ene sender data over I²S-busser. Siden I²S-spesifikasjonen ikke er entydig, har vi laget modulen utifra databladet til ADC-ene. Designet av mottaksmodulen er laget med utgangspunkt i følgende:

- Sender-ADC-er er av type TI PCM4204
- Alle bussene er klokket av samme eksterne oscillator og har felles LRCK.
- MSB blir sendt først, og LRCK bytter før LSB. [3]
- Det overføres 32 bit for hver kanal. LRCK bytter på hvert 32. bit. [2]
- Data er synkronisert på negativ flanke i sender. [2]
- LRCK er synkronisert på negativ flanke [2]



Figur 2: ADC – timingdiagram [2]

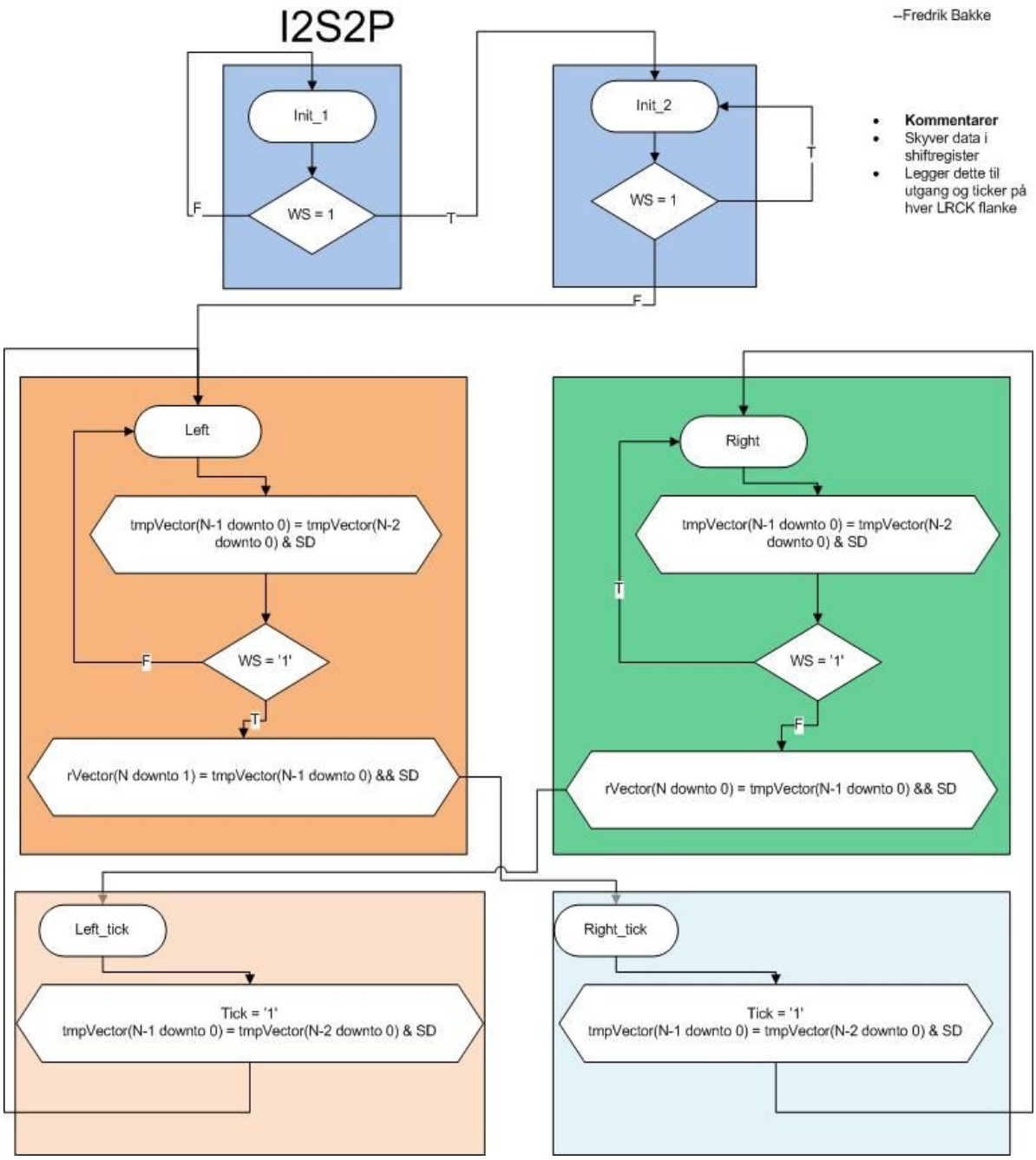
4 Oppbygning



Figur 3: Blokkdiagram for mottaksmodul, AMT-en er en del av SOPC modulen.

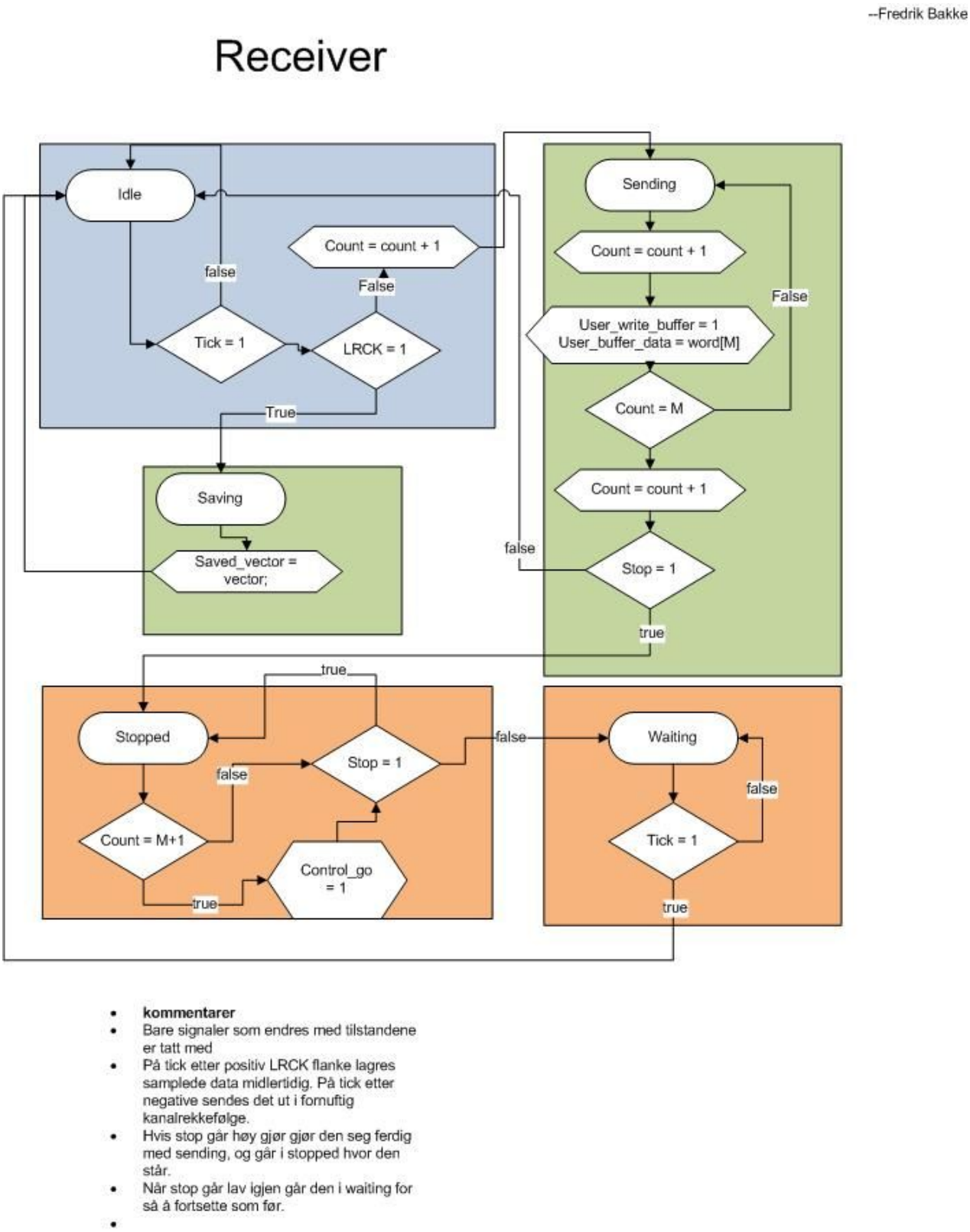
Mottaksmodulen har 4 grensesnitt (I2S2P) som hver gjør om seriell data fra en buss til parallelle data. Hvert grensesnitt er en FSMD (Finite State Machine with Datapath) som shifter data inn i et shiftregister, og produserer et tick når nytt data er klart. En overordnet modul (Receiver) lager de data og kontrollsignaler som er nødvendig for å få sendt data videre.

Tilstandsdiagrammet i Figur 4 beskriver grensesnittene.



Figur 4: Tilstandsdiagram, I2S2P

I2S2P-modulen har synkron reset. Ved en reset vil grensesnittene bruke to eller tre endringer i Word-select på å synkronisere seg til å starte mottaket på de venstre kanalene. Slik vet man at de første dataene etter en reset alltid er fra riktige kanaler.



Figur 5: Tilstandsdiagram, Receiver

Receiver-modulen får data fra I2S2P-modulene og genererer nødvendige data og kontrollsignaler

for at AMT-en kan sende data videre til FIFO. Siden hver buss sender 8 overflødige bit etter hver sample blir disse forkastet av den overordnede modulen. Kanalrekkefølge og byte-order gjøres om riktig. Receiver-modulen er også en FSMD med asynkron reset – og er beskrevet med tilstandsdiagrammet i Figur 5.

5 Referanser

[1] *exm-avalon-mm*. Altera.com, Sist besøkt 18.03.11
http://www.altera.com/support/examples/nios2/exm-avalon-mm.html

[2] *PCM4204 datasheet*, Texas Instruments, 2004.

[3] *I^2S bus specification.* Philips Semiconductors, Juni, 1996.

POSICOM

HØGSKOLEN i Buskerud

Etteranalyse		
Dokumenttype	Dokumentnummer	Versjonsnummer
Systembeskrivelse	S.8	1.a
Dato sist endret	Frigitt dokument?	Dokumentansvarlig
25.05.2011	Ja	Kathrine Kaasa
Gruppemedlemmer:		
Kathrine Kaasa, Knut Olav Skyttemyr, Jardar Hvidsten, Fredrik Bakke, Erlend Rauk Aasland		

Innholdsliste

1 Innledning....3
2 Prosjektgjennomføring....4
2.1 Arbeidsprosess....4
2.2 Prosjektplan....4
2.3 Kravspesifikasjon....5
2.4 Testspesifikasjonen....6
3 Økonomi....7
4 Subjektiv evaluering....7
5 Referanser....9

Forkortelser og Akronymer

Snipos	-	Sniper Positioning System
AD	-	Analog til Digital
FPGA	-	Field Programmable Gate Array
ADC	-	Analog-to-digital converter

1 Innledning

Denne oppgaven ble gitt av Posicom AS i Kongsberg. Gruppemedlem Knut Olav Skyttemyr hadde sommerjobb der sommeren 2010. Han ble spurt om å gjøre en bacheloroppgave for Posicom i forbindelse med systemet Snipos. Oppgaven var løst definert i starten, men det ville dreie seg om et nytt hardwaresystem som kunne ta i bruk arbeidet Antonio Ramos gjør i forbindelse med beamforming. Resten av medlemmene ble spurt i starten av august om å være med på dette prosjektet.

I dette dokumentet vil vi vurdere prosjektet vi har gjort. Vi vil ta en gjennomgang av prosjektet og se på hvordan vi har arbeidet ut i fra prosjektplan, kravspek og testspek. Vi vil se på den økonomiske delen av prosjektet. Til slutt vil vi gjøre en subjektiv evaluering av prosjektet i forhold til planleggingen.

2 Prosjektgjennomføring

2.1 Arbeidsprosess

Vi har valgt å jobbe mest mulig sammen på skolen. Det ble innført fleksitid de dagene som ble satt av til prosjektarbeid, hvor kjernetiden var mellom 9:00 og 14:00. Dette sikret minst 10 timers arbeid per uke og de resterende fem var hver enkelt ansvarlig for selv.

Vi har ført timer hver dag og skrevet hva vi har jobbet med. På denne måten har det vært enkelt å holde oversikt hvor mye tid som er brukt på hver enkelt aktivitet, og hvem som har jobbet på de forskjellige aktivitetene. Vi har ikke ført timer for pauser/lunsj, eller de timene vi har brukt på tankearbeid etter at vi har gått hjem for dagen. De timene som fremkommer av timelistene har utelukkende blitt brukt til jobbing på prosjektet.

Det at alle har tilbrakt mye tid sammen på prosjektrommet har ført til at det har vært enkelt å diskutere seg fram til løsninger når det har vært nødvendig og det har også vært enkelt å få med seg hva de andre holder på med. Når valg har blitt gjort har alle vært inneforstått med hva valgmulighetene er og hva det endelige valget ble.

Utviklingsmodellen som ble valgt i begynnelsen av prosjektet har blitt fulgt så langt det lot seg gjøre. Det har vært nødvendig å tilpasse den til prosjektet da det var umulig å fysisk teste modulene på AD-kortet hver for seg før vi produserte kortet. Det har derimot blitt tatt i bruk simuleringer til å verifisere at kretsdesignet vi laget fungerte. Disse simuleringene erstatter dermed testingen som skal gjøres etter at hver modul er laget. Dette er i samsvar med utviklingsmodellen inkrementell utvikling.

2.2 Prosjektplan

Før prosjektet startet i januar ble det skrevet en prosjektplan [1]. Dette var en foreløpig plan som det har blitt gjort flere endringer i underveis i prosjektet. Det ble fort lagt opp til at en gang i uken skulle det oppdateres hvor mye tid som tilsammen hadde blitt brukt på hver aktivitet på bakgrunn av det som sto i timelistene. Dette viste seg å være en fornuftig måte å holde orden på hvordan vi lå an i forhold til prosjektplanen.

I prosjektplanen la vi, etter en kraftig revidering rett etter første presentasjon, opp til at prosjektet skulle være ferdig senest 20. mai. Nå i slutten av prosjektet har dette ført til at vi har hatt et par «ekstra» dager som har kommet godt med.

Dokumentasjon

Noen aktiviteter har vært vanskelige å beregne hvor mye tid som går med. Dette gjelder spesielt aktivitetene som går på å skrive et eller flere dokumenter. Rent tidsmessig har det allikevel ikke vært et problem. Det har blitt lagt opp til at i tiden før en presentasjon skulle det brukes mest tid på dokumentasjon. Også tiden før siste innlevering har ikke dette vist seg å bli et problem fordi systemet var ferdig en uke før, slik at det var mulig å bruke de siste dagene på å gjøre ferdig

dokumentene.

Generelt sett har disse aktivitetene blitt økt i takt med at dokumentasjonsmengden har økt. Når det har vært mulig har det blitt flyttet timer fra andre aktiviteter over på disse.

Hardware

På enkelte tekniske aktiviteter har det vært vanskelig å beregne hvor mye tid man skal bruke. Dette går begge veier. Aktiviteten som gikk på ADC-en brukte vi mye mindre tid enn planlagt, mens det å lage kretskort tok mer tid enn planlagt. Selv om aktiviteter ble sprengt har ikke dette ført til at det har vært fare for at prosjektet ikke ble ferdig. Dette har man sikret ved å jobbe ekstra i den tiden man trenger det.

I ettertid ser vi at noen av aktivitetene kunne bli delt i to på en naturlig måte. Det ville derimot blitt vanskelig å gjøre etter at arbeidet med aktiviteten startet på grunn av at det i timelistene ble ført opp timer på begge de to «naturlige delene».

Framfor å legge til ekstra timer har det blitt flyttet timer fra andre hardwareaktiviteter som man vet man har satt opp flere timer enn det som har blitt brukt. Dette førte til at den totale arbeidsmengden innenfor hardware ikke har økt.

FPGA/Programvare

Underveis i prosjektet har det skjedd noen endringer både i forhold til tidsbruk og navngivning og innhold av aktiviteter. Etter å ha presentert ett forslag til arkitektur for oppdragsgiver tidlig i februar, ble mange av aktivitetene endret. De første aktivitetene under FPGA/Programvare-delen gikk fortere enn planlagt. Dette gjorde at vi hadde ekstra timer til overs som kunne settes på andre FPGA/Programvare-aktiviteter dersom det skulle bli nødvendig. Dette ble også nødvendig etterhvert, da det å slå sammen modulene og lagring til fil tok lenger tid en planlagt. Også under FPGA/Programvare-delen har timene kunnet bli flyttet innad. Unntaket er at det også ble flyttet noen timer over på testing da dette også tok lenger tid enn planlagt.

Testing

Det har skjedd endel endringer under denne delen av prosjektet. Utviklingstestene for FPGA/Programvare tok som nevnt mer tid enn opprinnelig planlagt. Utviklingstestene for hardware ble endret på da vi valgte å ha en aktivitet som gikk på simuleringene som ble gjort før vi produserte AD-kortet og en aktivitet for testene som ble gjort etter at AD-kortet var produsert. Sistnevnte aktivitet gikk også på eventuell omlodding av kortet. Denne aktiviteten burde også vært delt opp i to, men timelistene gjorde det igjen vanskelig å dele opp. Det ble loddet og testet om hverandre, så det enkleste var å kun overføre og legge til flere timer. Systemtesten har derimot tatt kortere tid enn forventet.

2.3 Kravspesifikasjon

Arbeidet i hele prosjektperioden har tatt utgangspunkt i kravene i kravspesifikasjonen [2]. Noen krav har blitt endret underveis, andre har ikke blitt oppfylt, men alle A-kravene har blitt oppfylt.

A.P.2 – Offentlighet

Kravet om offentlighet har endret seg i løpet av prosjektperioden. I utgangspunktet skulle ingen tekniske dokumenter offentliggjøres. Senere har oppdragsgiver bestemt at vi kan offentliggjøre alle dokumenter som ikke inneholder teknisk beskrivelse av Snipos.

B.I.A.1 – PC104

Kravet om maks størrelse på kort ble ikke oppfylt grunnet en begrensning på hvor vi kunne plassere komponentene på kortet. Oppdragsgiver foreslo tidlig å droppe dette kravet. Dette førte til billigere montering og et enklere enklere utlegg for oss.

C.I.A.3 – Stabling

Dette har vært et C-krav og vi har ikke prioritert å få til dette.

A.F.A.9 – Mikrofonforsterkning

Dette kravet er oppfylt. Men vi vil presisere at man må lodde på nye komponenter for å forandre forsterkningen. Vi har ikke prioritert å legge inn mulighet for enklere justeringsmåter.

A.F.A.11 – Spenningsforsyning AD-kort

AD-kortet forsynes med 5 V og 3,3 V fra FPGA-kortet. Vi har i tillegg valgt å legge inn mulighet for å koble til eksterne spenningsforsyninger.

C.F.D.7 – Beamforming

Dette kravet har ikke vært mulig å gjennomføre. Design og produksjon av mikrofonarrayet måtte gjøres av noen utenfra. Av forskjellige årsaker har ikke dette blitt gjort.

2.4 Testspesifikasjonen

Vi lagde en testspesifikasjon i starten av prosjektet som veldig skjematisk tok for seg hvert krav. Vi så etter hvert at dette var en uhensiktsmessig måte å teste på. Derfor gjorde vi noen ganske store endringer på testspesifikasjonen [3].

3 Økonomi

De største utgiftene vi har hatt har vært i forbindelse med AD-kortet. En viktig årsak til de avvikene vi har er at vi bestemte oss for å montere to kort i stedet for ett, for å ha ett i reserve. Opprinnelig budsjett tok bare høyde for montering av ett kort. Mikrofoner slapp vi å kjøpe inn fordi oppdragsgiver hadde nok mikrofoner liggende. Fargekopier og plakattrykking går under diverse-posten. Derfor ble den så stor.

Utgifter	**Beregnet pris**	**Endelig pris**
AD-kort	10 000,00 NOK	6 905,63 NOK
FPGA-kort	0,00 NOK	0,00 NOK
Komponenter	0,00 NOK	2 333,14 NOK
Lisenser	0,00 NOK	0,00 NOK
Kontorrekvisita	500,00 NOK	211,00 NOK
Mikrofoner	800,00 NOK	0,00 NOK
Diverse	700,00 NOK	984,00 NOK
Sum	12 000,00 NOK	10 433,77 NOK

Til tross for den høye utgiften på komponenter i forhold til budsjetterte utgifter, vil ikke dette påvirke sluttsummen på prosjektet i forhold til det som er budsjettert. Vi hadde et inntrykk av at vi kom til å få nok komponentprøver av de passive komponentene til å dekke komponentbehovet. Vi var ikke klar over at vi kom til å trenge ca. 240 passive komponenter til hvert kort. Dette var dårlig beregning fra vår side. Forbruket totalt blir allikevel mindre enn budsjettert, grunnet at vi brukte 3000 kr mindre på produksjon av AD-kortet enn det som ble satt opp.

4 Subjektiv evaluering

I løpet av det siste halvåret har vi lært mye om prosjektarbeid, jobbe i gruppe og det å prøve og jobbe etter en tidsplan. Det sistnevnte har kanskje vært det vanskeligste siden det er vanskelig å planlegge hvor lang tid en aktivitet vil ta. Til tross for dette synes vi det har gått veldig bra. Det har vært en del endringer i tidsplanen, men de overordnede tidsfristene har alltid blitt holdt.

Vi har vært heldige i forhold til at det vi har bestilt ikke har blitt forsinket, men har heller kommet før vi har forventet. Det har heller ikke vært lang ventetid på det vi har bestilt. Komponenter har kommet i løpet av to-tre dager og kretskortet kom i løpet av en drøy uke. FPGA-utviklingskortet fikk vi tak i før jul, så dette har vi kunnet ta i bruk fra starten av. Dette har gjort at vi ikke har hatt noen problemer med forsinkelser annet enn de vi eventuelt har skapt selv. Det var en liten periode hvor videre arbeid med FPGA-en og programvaren krevde at AD-kortet var produsert og at det var verifisert at det fungerte. Det ble en liten ventetid på et par dager, men denne tiden ble brukt på dokumentasjon som var nødvendig å gjøre. Når vi i tillegg valgte å få montere to kort kunne begge «avdelingene» bruke et AD-kort og dette har vært veldig nyttig.

Når vi har jobbet med AD-kortet har vi hele veien hatt god kontakt med oppdragsgiver, Posicom AS, og fått god hjelp og veiledning av dem. Vi har også fått god veiledning av lærere på skolen og intern veileder. Vi har innimellom blitt litt «presset» mellom forskjellige ønsker og løsninger, men vi føler at dette har løst seg greit ved at vi hele tiden har god dokumentasjon for de løsninger vi har valgt.

Det har gått med mye tid på å bli kjent med Altera sine utviklingsverktøy og ferdigmoduler – og på å planlegge designet. Vi fikk god starthjelp av blant annet Posicom, men etter det har designavgjørelser i hovedsak blitt tatt på grunnlag av studering av datablader, prototyping og testing. Ellers har vi oppsøkt hjelp fra skolen når vi har trengt det. Selv om vi ikke har brukt ressurser utenfor gruppen i like stor grad som de på hardware, har det vært betryggende å ha ansatte ved både Posicom, skolen og Arrow å kunne henvende seg til når det har vært nødvendig. Vi har også vært heldige med å ha god kontakt med Altera. Disse har gitt oss både FPGA-kortet og lisenser vi har hatt bruk for i utviklingen.

Som gruppe synes vi at vi har fungert veldig bra. Oppgaven har i stor grad krevd at vi har måttet jobbe mye sammen og diskutere oss fram til de rette løsningene. Dette har gått fint og det har ikke vært noen interne problemer i gruppa.

Til slutt vil vi takke intern veileder Sigmund Gudvangen, ekstern sensor og veileder Rangvald Otterlei, ekstern veileder Geir Myklatun, høyskolelektor Dag Samuelsen, Arrow Nordic, Altera, Texas Instruments og Josephson for hjelp og støtte gjennom hele prosjektet. I tillegg vil vi takke Monica Olsen for å lage logoen og venner og familie for støtte.

5 Referanser

[1] Prosjektgruppe 2011-09, *P.5 Prosjektplan*, MMS, 2011.

[2] Prosjektgruppe 2011-09, *P.3 Kravspesifikasjon*, MMS, 2011.

[3] Prosjektgruppe 2011-09, *P.4 Testspesifikasjon*, MMS, 2011.

POSICOM

HØGSKOLEN i Buskerud

MMS Multimikrofonsystem

Systemtest		
Dokumenttype	Dokumentnummer	Versjonsnummer
Test	T.3	1.a
Dato sist endret	Frigitt dokument?	Dokumentansvarlig
27.05.2011	Ja	Fredrik Bakke
Gruppemedlemmer:		
Kathrine Kaasa, Knut Olav Skyttemyr, Jardar Hvidsten, Fredrik Bakke, Erlend Rauk Aasland		

Innholdsliste

1 Introduksjon....3
2 Testforberedelse....4
2.1 Testutstyr....4
3 Testbeskrivelse....5
3.1 Forventet resultat....5
3.2 Resultat....5
3.3 Oppståtte feil....9
4 Konklusjon....10
5 Referanser....11

Figurliste

Figur 1: Testoppsett....5
Figur 2: 1 kHz, 100 dB akustisk måling. FS = 96 kHz, 24 bits oppløsning....6
Figur 3: 1 kHz, 100 dB akustisk måling. FS = 96 kHz, 24 bits oppløsning. De fire siste kanalene (J8, J10, J12, J14) skalert med en faktor på 3....6
Figur 4: 1 kHz, 50 dB akustisk måling. FS = 96 kHz, 24 bits oppløsning. De fire siste kanalene (J8, J10, J12, J14) skalert med en faktor på 3....7
Figur 5: 1 kHz, 122 dB akustisk måling. FS = 96 kHz, 24 bits oppløsning. De fire siste kanalene (J8, J10, J12, J14) skalert med en faktor på 3....8
Figur 6: 1 kHz, 126 dB akustisk måling. FS = 96 kHz, 24 bits oppløsning....8

Forkortelser og akronymer

FPGA	-	Field Programmable Gate Array
AD	-	Analog til digital

1 Introduksjon

Systemtesten tester systemet som en helhet [1]. Dersom denne testen bestås er systemet ferdig.

2 Testforberedelse

2.1 Testutstyr

- L-MS-01, halvkulehøyttaler
- Signalgenerator, Hewlett Packard 33120A [2]
- FPGA-kort [3]
- AD-kort [4]
- Mikrofoner [6]
- Kabler
- Mottaks-PC
- Lydtrykkmåler, Clas Ohlson ST-805

3 Testbeskrivelse

Sett opp L-MS-01, en halvkulehøyttaler, og mål opp en halvsirkel som mikrofonene plasseres på, se Figur 1. Vi sikrer da at lyden treffer alle mikrofonene samtidig. Koble mikrofonene til AD-kortet (Analog til Digital). Send et signal med en kjent frekvens til høyttaleren. Bruk lydtrykksmåleren til å se hvilket lydtrykk mikrofonene blir utsatt for. Gjør et opptak med ca. 100 dB, et med ca. 125 dB og et med ca. 50 dB. Les av filen med de lagrede lyddataene. Verifiser at dataene er like som det avspilte lydsignalet. Det vil si at kanalene skal være synkroniserte og at forsterkningen er lik på alle kanalene.

Vi kjenner ikke til frekvensresponsen til signalkilden vår, derfor gjør vi ikke noen frekvensmålinger i denne testen. Frekvensresponsen til systemet uten mikrofonene tilkoblet ble testet tidligere og funnet tilfredsstillende [5].

Figur 1: Testoppsett

3.1 Forventet resultat

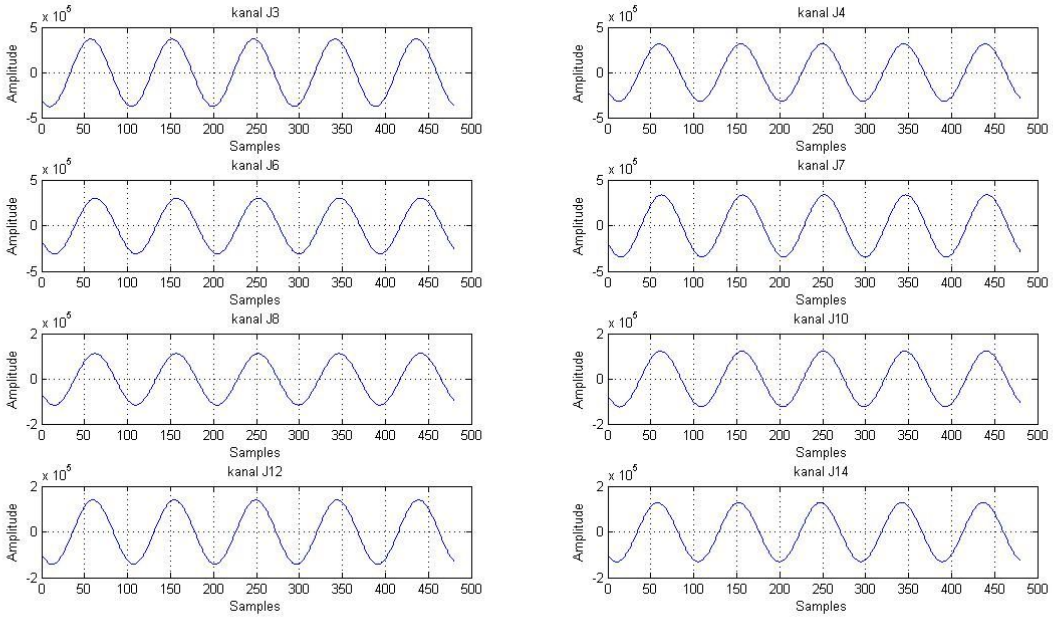
Etter testene vi har gjort av AD-kortet forventer vi god synkronisering i tidsdomenet. Men vi forventer litt forskjell på amplitudene til de åtte kanalene. Vi forventer også at fire av kanalene skal gi et utslag på ca. 1/3 sammenlignet med de fire andre [5].

125 dB skal være i toppen av arbeidsområdet. Det vil si en amplitude som nærmer seg $\pm 2^{23}$. 50 dB skal være i bunnen av arbeidsområdet. Det vil si at vi akkurat skal kunne skille ut signalet fra bakgrunnsstøyen.

3.2 Resultat

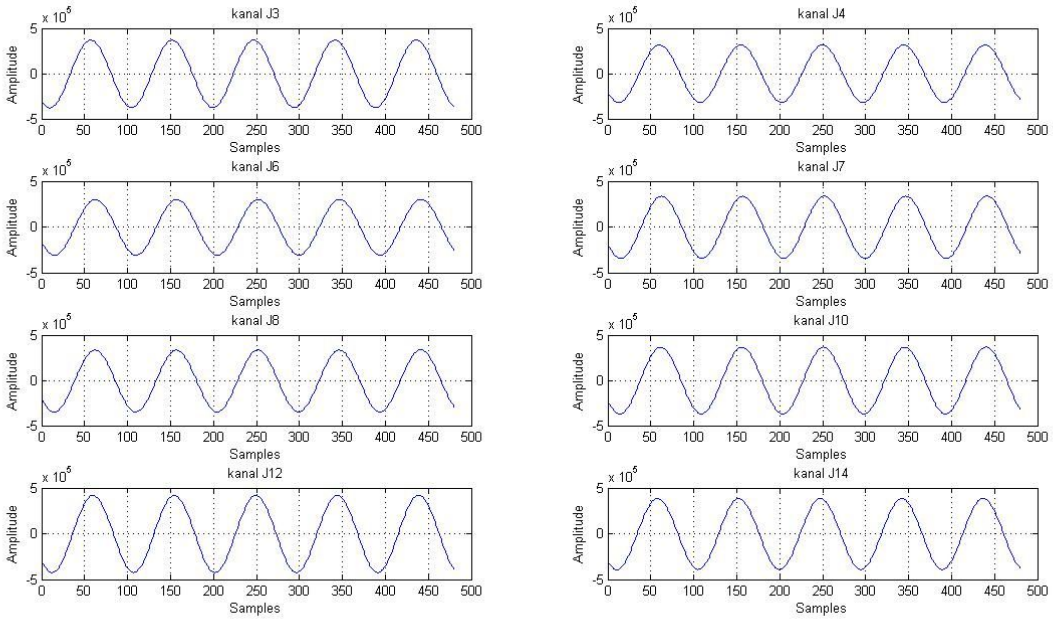
Først ser vi på resultatet av 100 dB målingen vår, se Figur 2. Som vi ser får vi en fin sinuskurve på

alle kanalene. Vi ser også at amplituden på de fire siste kanalene er ca. 1/3 av de fire første.



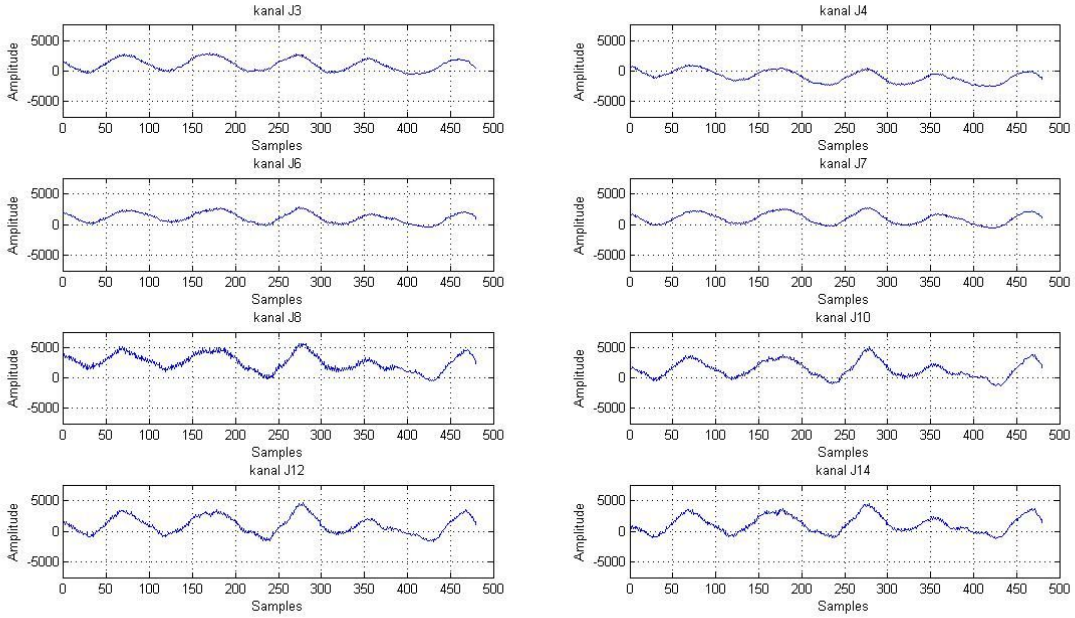
Figur 2: 1 kHz, 100 dB akustisk måling. F_S = 96 kHz, 24 bits oppløsning.

Denne forskjellen kompenserer vi for digitalt. Se Figur 3 for bilde av signalene etter denne skaleringen. Det er fortsatt litt forskjeller. Ved behov kan man fininnstille hver enkelt kanal ved hjelp av digital skalering.



Figur 3: 1 kHz, 100 dB akustisk måling. F_S = 96 kHz, 24 bits oppløsning. De fire siste kanalene (J8, J10, J12, J14) skalert med en faktor på 3.

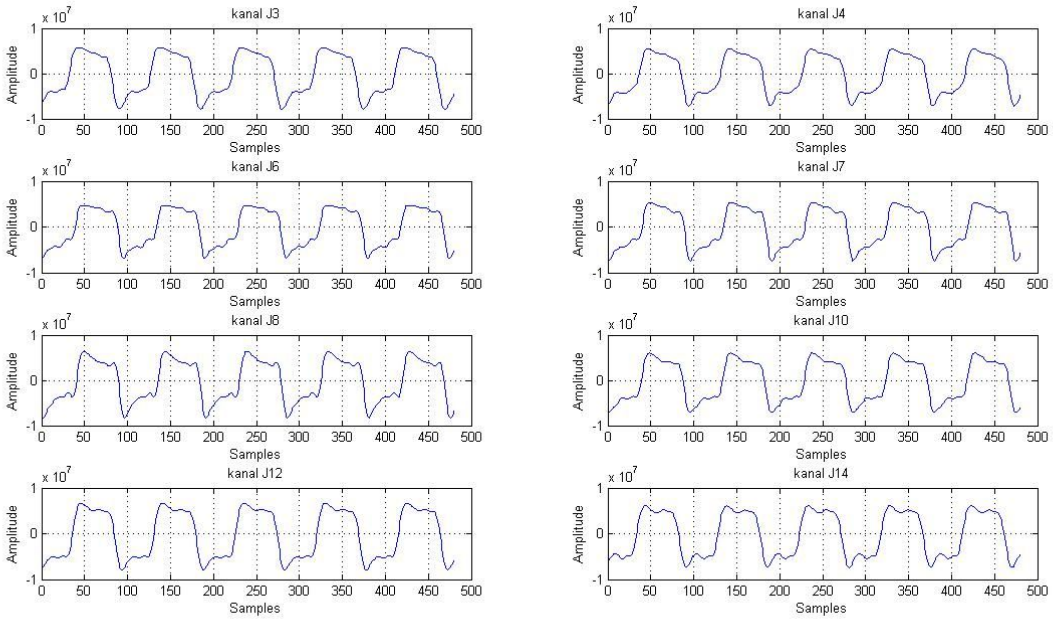
Neste måling var ved 50 dB SPL. Dette er i nedre grenseområde for hva systemet skal kunne fange opp. Se Figur 4 for bilde av opptaket. Som vi ser kan vi skille ut sinuskurven, men det er betraktelig mer støy sammenlignet med 100 dB målingen. Denne støyen stammer fra de akustiske omgivelsene, fra selve mikrofonene og AD-kortet.



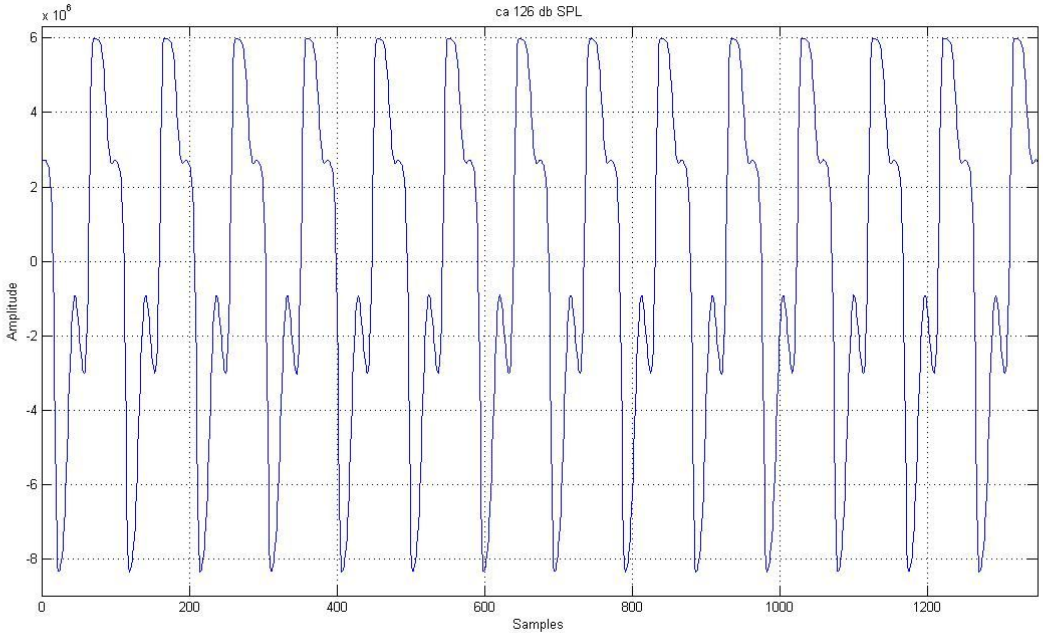
Figur 4: 1 kHz, 50 dB akustisk måling. F_S = 96 kHz, 24 bits oppløsning. De fire siste kanalene (J8, J10, J12, J14) skalert med en faktor på 3.

Den siste målingen tok vi på ca. 122 dB. Det var det høyeste vi klarte å oppnå med testutstyret vi hadde tilgjengelig. Se Figur 5 for bilde av opptaket. Som vi kan se så er sinussignalet kraftig forvrengt. Dette skyldes forvrenging i høyttaleren brukt til å spille av det akustiske signalet. Ellers så ser vi at maks amplitude er innenfor maks tillatte (2^{23}).

Vi fikk gjort en måling med ca. 126 dB, men kun på en mikrofon (holdt den tett inntil høyttaleren). Se Figur 6 for bilde av opptaket. Vi ser at ved 126 dB er vi veldig nærme maksimum amplitude. Som vi så i [5], så har vi en DC-offset på ca. -50 mV.



Figur 5: 1 kHz, 122 dB akustisk måling. F_S = 96 kHz, 24 bits oppløsning. De fire siste kanalene (J8, J10, J12, J14) skalert med en faktor på 3.



Figur 6: 1 kHz, 126 dB akustisk måling. F_S = 96 kHz, 24 bits oppløsning.

3.3 Oppståtte feil

Høyttalerforvregning førte til deformert sinuskurve ved 122 dB- og 126 dB-målingen.

4 Konklusjon

Vi har ikke oppdaget noen nye feil i denne systemtesten. Det vi oppdaget av avvik i løpet av tidligere tester var fortsatt til stedet i systemtesten [5]. Det vil si små variasjoner på amplituden til sinussignalet, og en liten negativ DC-offset. Disse er diskutert med intern veileder og vil sannsynligvis ha lite å si for den bruken systemet er tenkt til.

Vi klarte å skille ut sinussignalet fra støyen ved 50 dB, og 122 dB var greit innenfor maksverdi. Så beregningen vi gjorde på at arbeidsområdet vårt er fra 52 dB til 128 dB ser ut til å stemme ganske godt.

5 Referanser

[1] Prosjektgruppe 2011-09, *P.4 Testspesifikasjon.*, MMS, 2011

[2] *User's Guide HP33120a.*, Hewlett Packard, 1997.

[3] *DE2-115 User Manual.*, Terasic, 2010

[4] Prosjektgruppe 2011-09, *S.2 Brukermanual.*, MMS, 2011.

[5] Prosjektgruppe 2011-09, *T.4 Test av AD-kort.*, MMS, 2011.

[6] *Josephson C550.*, Josephson Engineering.

POSICOM

HØGSKOLEN i Buskerud

MMS Multimikrofonsystem

Testrapport - AD-kort		
Dokumenttype	Dokumentnummer	Versjonsnummer
Test	T.4	1.a
Dato sist endret	Frigitt dokument?	Dokumentansvarlig
27.05.2011	Ja	Jardar Hvidsten
Gruppemedlemmer:		
Kathrine Kaasa, Knut Olav Skyttemyr, Jardar Hvidsten, Fredrik Bakke, Erlend Rauk Aasland		

Innholdsliste

1 Introduksjon....4
1.1 Relevante tester....4
2 Testforberedelse....5
2.1 Testutstyr....5
3 Kjøring av T.H.1....6
3.1 Forventet resultat....6
3.2 Resultat....6
3.3 Konklusjon....6
4 Kjøring av T.H.2....7
4.1 Forventet resultat....7
4.2 Resultat....7
4.3 Konklusjon....7
5 Kjøring av T.H.3....8
5.1 Forventet resultat....8
5.2 Resultat....9
5.3 Oppståtte feil....9
5.4 Konklusjon....9
6 Kjøring av T.H.4....10
6.1 Forventet resultat....10
6.2 Resultat....10
6.3 Konklusjon....10
7 Kjøring av T.H.5....11
7.1 Forventet resultat....11
7.2 Resultat....11
7.3 Oppståtte feil....11
7.4 Konklusjon....11
8 Andre kjøring av T.H.5....12
8.1 Forventet resultat....12
8.2 Resultat....12
8.3 Konklusjon....15
9 Kjøring av T.H.6....16
9.1 Forventet resultat....16
9.2 Resultat....16
9.3 Konklusjon....17
10 Oppsummering....18
11 Referanser....19

Figurliste

Figur 1: Bitklokke med og uten dempemotstand....8
Figur 2: Forskjell på amplituder. FS = 96 kHz , 24 bits oppløsning....13
Figur 3: Amplitude som funksjon av frekvens. Signalgeneratorsweep fra 20 Hz til 10 kHz. Fs = 96 kHz, 24 bits oppløsning....14
Figur 4: Amplitude som funksjon av frekvens. Signalgeneratorsweep fra 20 Hz til 20 kHz. Fs = 96 kHz, 24 bits oppløsning....15
Figur 5: Samme signal på alle kanalene samtidig. FS = 96 kHz, 24 bit oppløsning....16
Figur 6: Synkronisering mellom de åtte kanalene. FS = 96 kHz, 24 bit oppløsning....17

Forkortelser og akronymer

Rms	-	Root Mean Square
JFET	-	Junction Field Effect Transistor
ADC	-	Analog-to-digital converter
FPGA	-	Field Programmable Gate Array
AD	-	Analog til digital

1 Introduksjon

Dette dokumentet inneholder hardwaretestene av AD-kortet (Analog til Digital) [5]. Testene som skal kjøres er ment for å avdekke eventuelle feil med komponenter, montasje eller design. Disse testene må bestås før AD-kortet er klart til systemtesten.

1.1 Relevante tester

Test-ID:
T.H.1
T.H.2
T.H.3
T.H.4
T.H.5
T.H.6

2 Testforberedelse

2.1 Testutstyr

- Oscilloskop, Hewlett Packard 54610A [2]
- Voltmeter, Escort EDM168
- Signalgenerator, Hewlett Packard 33120A [3]
- Kalibrator, Norsonic: 1 kHz, 10 Pa (114 dB SPL)
- AD-kort [4]
- Spenningsforsyninger på Delta Elektronika E030-3 og GW GPS-3030
- Mikrofoner [8]

3 Kjøring av T.H.1

Kalibreringstest av mikrofonene. Denne testen har som mål å avdekke eventuelle forskjeller mellom de åtte mikrofonene vi skal bruke. Vi bruker kalibratoren for å få et kjent signal ut av mikrofonen, og et oscilloskop til å lese av signalene. Vi sjekker også om noen av mikrofonlederne er koblet til chassis.

3.1 Forventet resultat

Et lydtrykk på 10 Pa satt på en mikrofon med en sensitivitet på 10 mV/Pa skal gi en spenning ut på 10 Pa · 10 mV = 100 mV_{rms}. Dette tilsvarer 141 mV_p.

3.2 Resultat

Spenningen ut var avhengig av lasten. Ut i fra databladet til mikrofonen er anbefalt last 1 kΩ. Når vi målte med en last på 1 kΩ fikk vi en spenning på ca. 120-130 mV_p. Dette er veldig nærme det vi forventer. Vi må også ta høyde for en viss usikkerhet på grunn av unøyaktige avlesninger på oscilloskopet og unøyaktige motstander. Disse forskjellene forsterkes gjennom forforsterkeren. Forskjellene i amplitude er relativt enkelt å kompensere for digitalt inne på mottaks-PC-en.

Endret vi lasten endret vi også amplituden på signalet. Høyere last gir høyere amplitude. Dette stemmer også med teorien for JFET (Junction Field Effect Transistor) der forsterkningen er proporsjonal med lasten [1].

Tre av åtte mikrofoner har negativ leder koblet til chassis.

3.3 Konklusjon

Vi må ta hensyn til at mikrofonen ikke klarer å gi forventet spenning ut uavhengig hvilken last den ser. Fordi vi har valgt å bruke single ended konfigurasjon vil det ikke by på problemer at tre av mikrofonene har negativ leder koblet til chassis. Det kan tenkes at mikrofonstativet som skal lages i fremtiden er koblet til jord på en eller annen måte.

4 Kjøring av T.H.2

Denne testen har som mål å verifisere at spenningsforsyningene fra FPGA-kortet (Field Programmable Gate Array) til AD-kortet fungerer korrekt. Vi bruker voltmeter til å måle spenningene.

4.1 Forventet resultat

Spenningen etter filtrene skal være mellom 4,75 - 5,25 VDC og 3,0 - 3,6 VDC.

4.2 Resultat

Spenningen etter filtrene var 4,91 VDC og 3,26 VDC.

4.3 Konklusjon

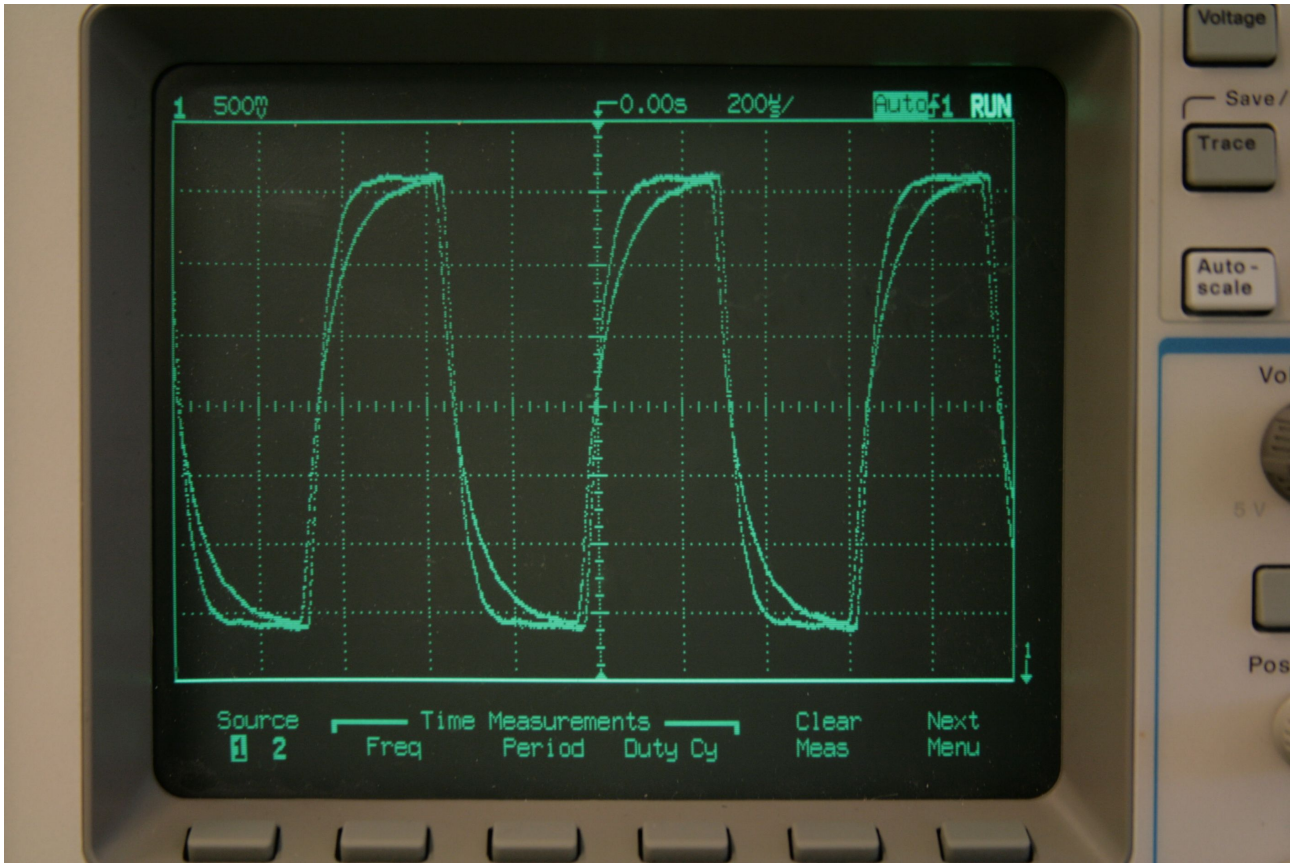
Kortet fungerer som forventet med tanke på spenningsforsyning og filtrering.

5 Kjøring av T.H.3

I denne testen skal vi verifisere at frekvensen på klokkelinjene er som forventet. Når spenningsforsyningen fungerer korrekt skal klokkelinjene gå hele tiden. Frekvensen måles med oscilloskop.

5.1 Forventet resultat

L/R-klokken skal ha en frekvens på 96 kHz. Bitklokken skal ha en frekvens på 6,144 MHz. Systemklokken skal ha en frekvens på 24,576 MHz.



Figur 1: Bitklokke med og uten dempemotstand

5.2 Resultat

Frekvensene var som forventet på alle tre klokkene. Se Figur 1 for bilde av bitklokken. Som vi ser er stigningstiden ganske lang sammenlignet med klokkeperioden. I samme bilde ser vi bitklokken uten dempemotstanden. Stigningstiden er betraktelig kortere.

5.3 Oppståtte feil

Vi har satt inn dempemotstander flere steder i kretsen for blant annet å bedre den elektromagnetiske kompabiliteten (EMC) til kortet. Men vi ser at den totale dempingen har blitt for stor på bitklokken. Dette fører igjen til høy stigningstid i forhold til klokkeperioden. Anbefalt verdi på stigningstiden er maksimum 10 % av klokkeperioden. I vårt system var stigningstiden 20-25 % av klokkeperioden. På grunn av dette fungerte ikke mottaksmodulen på FPGA-kortet. Da vi fjernet noen av dempemotstandene gikk stigningstiden ned til ca. 12 % og mottaket på FPGA-kortet fungerte.

5.4 Konklusjon

Klokkelinjene fungerte, men bitklokken hadde for lang stigningstid. Løsningen er å fjerne dempemotstanden rett etter utgangspinnen på ADC-pakken (Analog-to-digital converter). Klokken blir dempet nok av motstandene som befinner seg mellom krystalloscillatoren og inngangen på ADC-pakken.

6 Kjøring av T.H.4

Denne testen skal verifisere at signalflyten er korrekt fra mikrofoninngangene helt til 40-pins-konnektoren. Bruk en signalgenerator og send et signal på 1 kHz på en mikrofoninngang av gangen. 5 VDC-forsyningen må blokkeres med en kondensator. Bruk et oscilloskop til å måle på den tilhørende forforsterkerutgangen, og I^2S-bussene.

6.1 Forventet resultat

Signalflyt helt fra mikrofonkonnektoren og til 40-pins-konnektoren på alle de åtte kanalene. Signalsvinget skal være flyttet opp til 2,5 VDC etter forforsterkeren. Det skal gå en bitstrøm på I^2S-bussene.

6.2 Resultat

Det er signalflyt fra mikrofonkonnektoren og helt til 40-pins-konnektoren på alle de åtte kanalene. Signalsvinget er også flyttet opp til 2,5 VDC på alle kanalene.

6.3 Konklusjon

AD-kortet fungerer tilfredsstillende med tanke på signalflyt på alle kanalene.

7 Kjøring av T.H.5

Forsterkningstest. Bruk samme mikrofon på alle åtte inngangene. Bruk kalibratoren og mål signalet på utgangen til forforsterkeren. Verifiser av signalet har blitt forsterket som forventet. Sjekk også at frekvensresponsen er som forventet.

7.1 Forventet resultat

Med en inngangsamplitude på 141 mV_P skal utgangsamplituden være:
2,5 V + 141 $mV_P \cdot 1{,}75\ V_{Put}/V_{Pinn} \approx 2{,}75V_P$.. Ut i fra simuleringene skal frekvensresponsen være tilnærmet flat fra 20 Hz – 20 kHz.

7.2 Resultat

Spenningen ut er avhengig av lasten mikrofonen ser. Da vi koblet til mikrofonen til AD-kortet fikk vi bare et signalsving på ca. 10% av det vi forventet. Men selve AD-kortet fungerte som forventet. Vi fikk en forsterkning på 1,75 V_{Put}/V_{Pinn} selv om signalene inn var mye mindre enn forventet.

7.3 Oppståtte feil

Vi har tatt høyde for at mikrofonene klarer å gi ut 10 mV/Pa uansett last. Vi har for liten last, noe som fører til et alt for lavt signalsving.

7.4 Konklusjon

Vi må øke lasten mikrofonen ser. Men samtidig må vi passe på at impedanstilpasning blir riktig og at den totale forsterkningen fortsatt blir 1,75 V_{Put}/V_{Pinn}.

8 Andre kjøring av T.H.5

Ny kjøring av forsterkningstesten med nye verdier på motstandene rundt forforsterkeren. Se [7] for mer informasjon om hvordan vi kom frem til de nye verdiene.

8.1 Forventet resultat

På grunn av dårlig tid fikk vi ikke fininnstilt motstandsverdiene slik at arbeidsområdet går nøyaktig til 135 dB. Vi forventer at arbeidsområdet blir litt lavere enn først antatt. Vi forventer også at frekvensresponsen faller litt av mellom 10 kHz og 20 kHz, dette er enkelt å bedre, men vi bruker ikke tid på det fordi disse frekvensene blir filtrert bort av Snipos uansett. Vi forventer litt forskjell mellom kanalene på grunn av komponenttoleranser.

8.2 Resultat

Som vi ser av Tabell 1 gir 114 dB SPL inn på AD-kortet omtrent 20 % av maks amplitude ($1{,}6 \cdot 10^6 / 2^{23} \approx 20$ %). Det vil si at 100 % utslag tilsier en økning på $20 \cdot \log(100 / 20) \approx 14$ dB. Altså vil arbeidsområdet være opp til ca. 128 dB. Da har vi ikke tatt med i beregningen at forforsterkeren går i metning litt før 100 %. I ytelsestestene fant vi ut at systemet har en SNR på ca 75,5 dB [6]. Arbeidsområdet til systemet vårt blir da fra ca. 52 dB til 128 dB SPL.

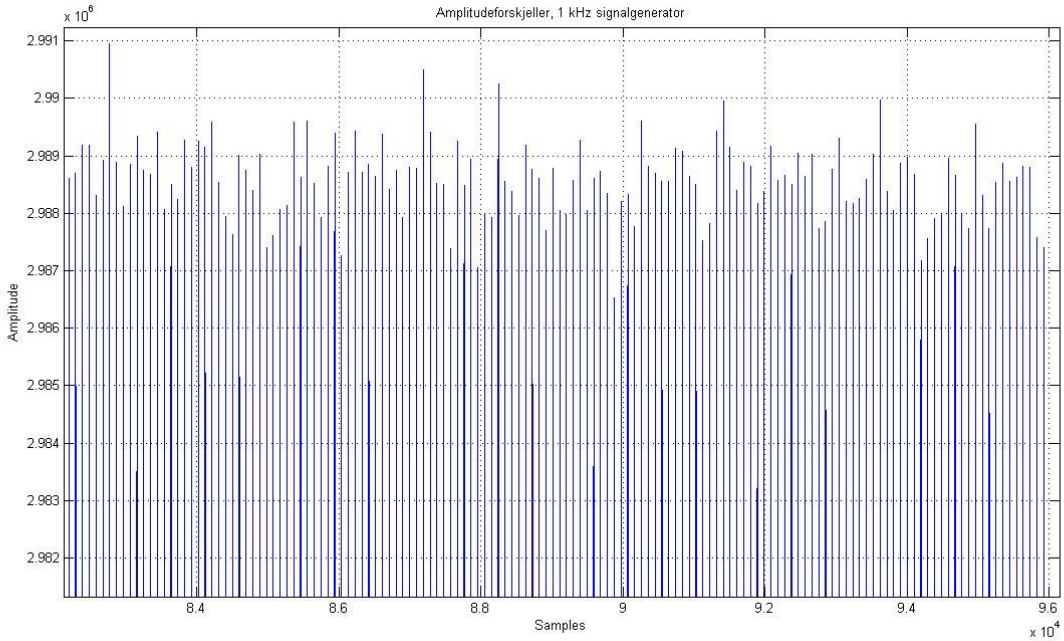
Vi så også at det var litt forskjell på kanalene når det kommer til amplitude. Selv om mikrofonene på papiret i utgangspunktet skal være de samme, ser vi at det er tydelige forskjeller på mikrofonene. Det ser ut som de fire nyeste mikrofonene (J8, J10, J12, J14) bare har en sensitivitet på 1/3 sammenlignet med de fire eldste. Dette kan vi kompensere for inne på mottaks-PC-en. Det er også litt forskjell på amplituden mellom mikrofonene som er helt like. Her spiller også toleranser på motstandsverdier i det passive nettverket rundt forforsterkeren inn.

En annen ting vi ser av tabellen er at vi har en liten negativ DC-offset. Denne offseten er på ca. 50 mV. ($(2{,}5\text{V} / 2^{23}) \cdot (1{,}77 \cdot 10^6 - 1{,}61 \cdot 10^6) \approx 48$ mV), se Tabell 1. Dette skyldes mest sannsynlig spenningsforsyningen. Ved måling ser vi at vi har ca. 4,9 VDC inn på AD-kortet, ikke 5 VDC. Forforsterkeren bruker differansen mellom forsyningsspenningen og jord til å sette referansespenningen. Så med 4,9 VDC blir referansespenningen 2,45 V ikke 2,5 V som AD-konverteren vil ha. Denne offseten forplanter seg og ut fra AD-konverteren og vi kan da forvente 2,45 V – 2,5 V = - 50 mV. Akkurat som vi ser i målingene våre.

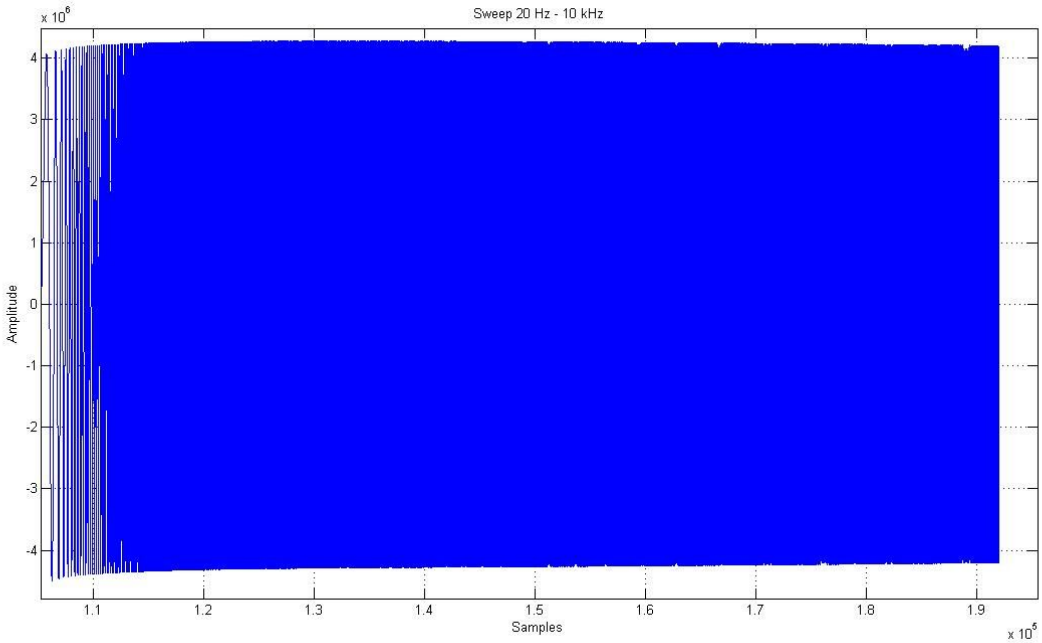
Tabell 1: Amplitudeforskjeller på de åtte kanalene (24 bits verdier: $\pm 2^{23}$).

Kanal	Maks amplitude ved 114 dB	Min amplitude ved 114 dB
J3	1611314	-1770570
J4	1523903	-1722655
J6	1546791	-1701139
J7	1599070	-1770565
J8	505595	-538847
J10	532825	-569751
J12	533325	-576176
J14	586133	-629648

Vi ser av Figur 2 at det er små forskjeller på amplitudene til sinuskurven innad i samme kanal. Det kan være flere årsaker til dette. Det kan være unøyaktigheter på signalgeneratoren, eller det kan være støy fra spenningsforsyningen som skaper disse forskjellene. Uansett er disse forskjellene i størrelsen mindre enn ca. 2 mV. Dette har sannsynligvis lite å si til den bruken som er tenkt. Snipos ser kun på enkeltpulser, ikke flere like rett etter hverandre.

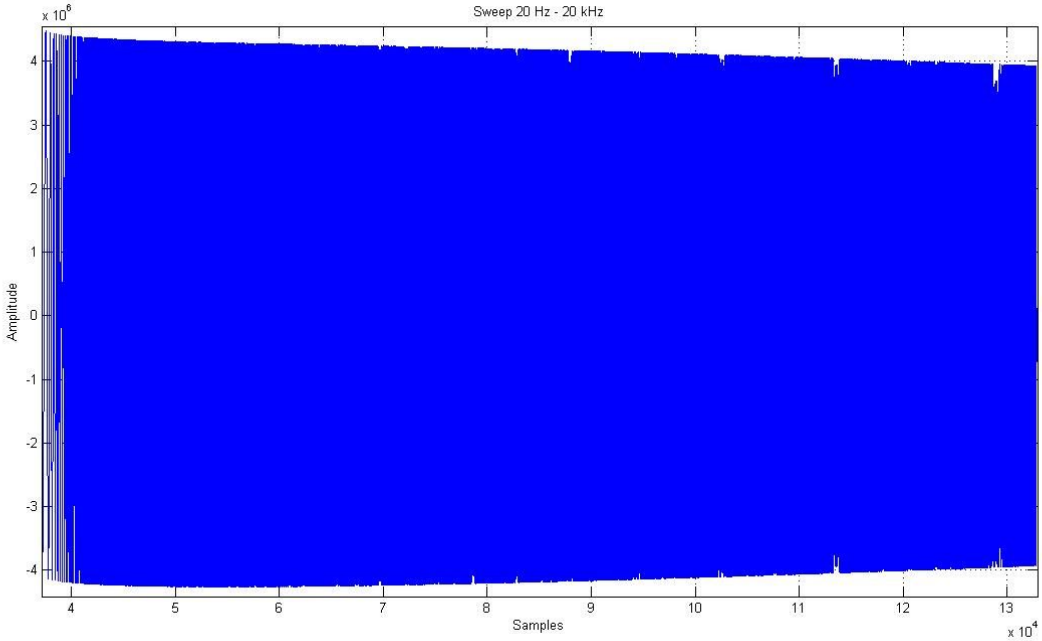


Figur 2: Forskjell på amplituder. F_S = 96 kHz , 24 bits oppløsning.



Figur 3: Amplitude som funksjon av frekvens. Signalgeneratorsweep fra 20 Hz til 10 kHz. Fs = 96 kHz, 24 bits oppløsning

Av Figur 3 kan vi antyde frekvensresponsen til systemet. Vi ser at amplitudene er tilnærmet like mellom 20 Hz og 10 kHz.



Figur 4: Amplitude som funksjon av frekvens. Signalgeneratorsweep fra 20 Hz til 20 kHz. Fs = 96 kHz, 24 bits oppløsning

Vi ser av Figur 4 at responsen faller litt av mellom ca. 10 kHz og 20 kHz. Dette er enkelt å forbedre, men disse frekvensene blir filtrert bort av Snipos uansett.

8.3 Konklusjon

Vi hadde egentlig bestemt oss for et arbeidsområde opp til 135 dB SPL. Vi ser at vi har tatt litt for godt i og har nå et arbeidsområde opp til ca. 128 dB SPL. Dette er diskutert med og godkjent av oppdragsgiver. På grunn av dårlig tid gjør vi ikke nye beregninger for å fininnstille dette enda bedre.

Det er små forskjeller på amplituden til de forskjellige kanalene. Disse forskjellene er så små, og de er ikke kritiske for Snipos. Derfor gjør vi ikke noe mer med de.

DC-offseten på utgangen av AD-konverteren er heller ikke kritisk. Snipos ser kun på enkeltpulser og er interessert i maksverdiene.

Frekvensresponsen er også god nok til den bruken Snipos har.

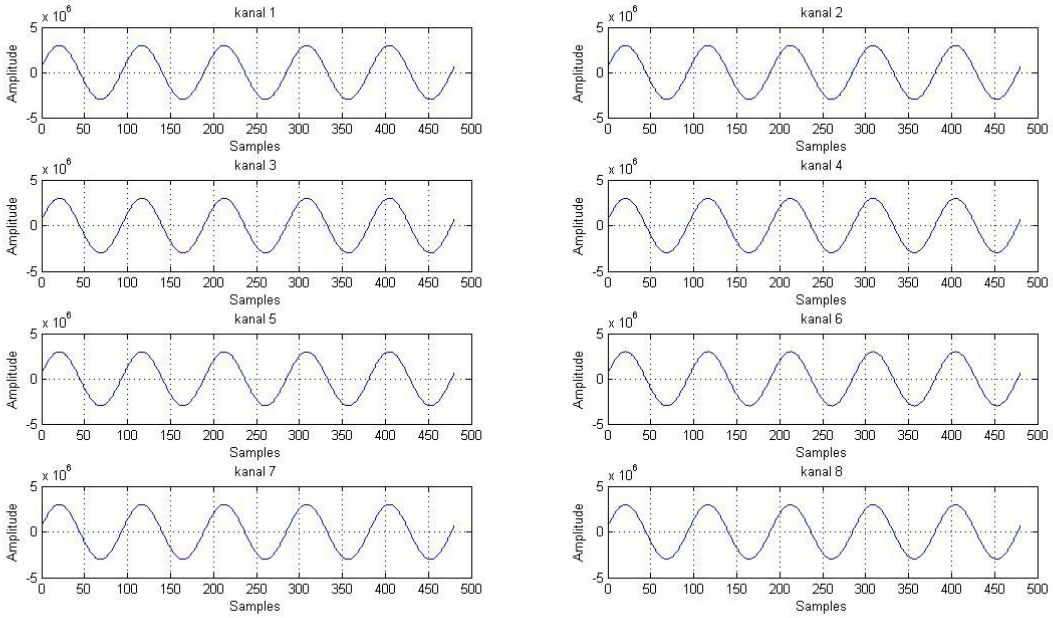
9 Kjøring av T.H.6

Testing av synkronisering mellom de 8 kanalene. Vi spiller av et 1 kHz signal fra signalgeneratoren på alle mikrofoninngangene samtidig.

9.1 Forventet resultat

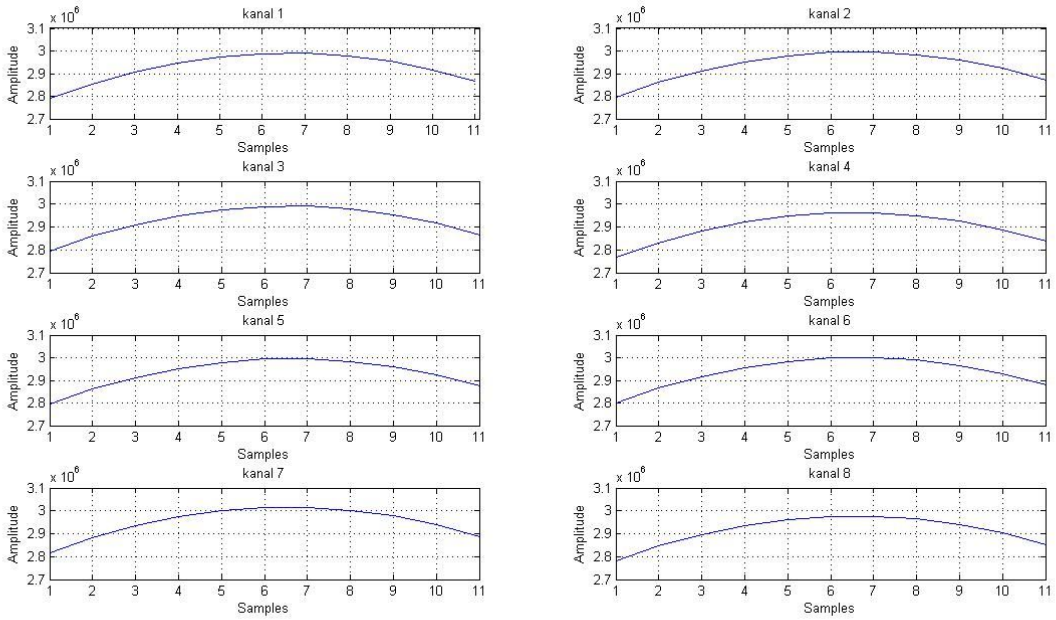
Forventet resultat er at alle kanalene er helt synkroniserte.

9.2 Resultat



Figur 5: Samme signal på alle kanalene samtidig. F_S = 96 kHz, 24 bit oppløsning.

I Figur 5 ser vi at alle kanalene tilsynelatende er synkroniserte i forhold til hverandre.



Figur 6: Synkronisering mellom de åtte kanalene. F_S = 96 kHz, 24 bit oppløsning.

I Figur 6 har vi zoomet inn på signalene vi så i Figur 5. Vi ser at toppunktet på kurven kommer mellom sample 6 og 7 i alle kanalene. Det vil si at forskjellen mellom kanalene er en god del mindre enn et sample.

9.3 Konklusjon

Vi har sett at avviket mellom kanalene i tidsaksen er mindre enn ett sample. Omsatt til tid blir denne forskjellen mindre enn 1 / 96000 ≈ 10 µs. Med en lydhastighet på ca. 340 m/s, blir denne forskjellen $10 \cdot 10^{-6}$ s · 340 m/s = 3,5 mm. Dette er såpass lite at vi i praksis kan vi se bort i fra dette lille avviket.

10 Oppsummering

Mikrofonene er som forventet litt forskjellige. Det er det lite å gjøre med for oss.

Spenningsforsyningene fungerte som forventet.

Vi måtte fjerne noen dempemotstander på klokkelinjene for at de skulle fungere korrekt.

Signalflyten på AD-kortet var som forventet.

Lasten mikrofonene ser gjorde at vi fikk gale verdier på signalene. Etter nye beregninger kom vi mye nærmere det arbeidsområdet vi hadde satt, ca. 128 dB mot planlagt 134 dB. Totalt arbeidsområdet for systemet vårt er ca. 52-128 dB.

Amplitudene til de forskjellige kanalene avviker litt fra hverandre. Dette får vi ikke gjort noe med i første omgang. Det eneste vi skal prøve å gjøre er å digitalt skalere opp de fire mikrofonene som bare gav 1/3 amplituder.

Langs tidsaksen er synkroniseringen av kanalene tilfredsstillende.

11 Referanser

[1] Floyd, T., *Electronic Devices.,* 8th ed., Pearson education, 2008.

[2] *User and Service Guide HP 54610 Oscilloscope.*, Hewlett Packard, 1994.

[3] *User's Guide HP33120a.*, Hewlett Packard, 1997.

[4] Prosjektgruppe 2011-09, *S.2 Brukermanual.*, MMS, 2011.

[5] Prosjektgruppe 2011-09, *T.4 Testspesifikasjon.*, MMS, 2011.

[6] Prosjektgruppe 2011-09, *T.6 Ytelse system.*, MMS, 2011.

[7] Prosjektgruppe 2011-09, *S.1 Designdokument.*, MMS, 2011.

[8] *Josephson C550.*, Josephson Engineering.

POSICOM

HØGSKOLEN i Buskerud

Testrapport - FPGA/Programvare

Dokumenttype	Dokumentnummer	Versjonsnummer
Test	T.5	1.a
Dato sist endret	Frigitt dokument?	Dokumentansvarlig
27.05.2011	Ja	Erlend Rauk Aasland

Gruppemedlemmer:

Kathrine Kaasa, Knut Olav Skyttemyr, Jardar Hvidsten, Fredrik Bakke, Erlend Rauk Aasland

Innholdsliste

1 Introduksjon....4
1.1 Utførte tester....4
2 Kjøring av T.P.3....5
2.1 Testutstyr....5
2.2 Visuell testoversikt....5
2.3 Testbeskrivelse....6
2.3.1 Forventet resultat....6
2.3.2 Resultat....7
2.3.3 Oppståtte feil....7
2.4 Konklusjon....7
3 Kjøring av T.P.4....8
3.1 Testutstyr....8
3.2 Visuell testoversikt....8
3.3 Testbeskrivelse....8
3.3.1 Forventet resultat....9
3.3.2 Resultat....9
3.3.3 Oppståtte feil....9
3.4 Konklusjon....9
4 Kjøring av T.P.7, T.P.1....10
4.1 Testutstyr....10
4.2 Visuell testoversikt....10
4.3 Testbeskrivelse....10
4.3.1 Forventet resultat....11
4.3.2 Resultat....11
4.3.3 Oppståtte feil....13
4.4 Konklusjon....13
5 Kjøring av T.P.5, T.P.2....14
5.1 Testutstyr....14
5.2 Visuell testoversikt....14
5.3 Testbeskrivelse....14
5.3.1 Forventet resultat....15
5.3.2 Resultat....15
5.3.3 Oppståtte feil....15
5.4 Konklusjon....15
6 Kjøring av T.P.6....17
6.1 Testutstyr....17
6.2 Visuell testoversikt....17
6.3 Testbeskrivelse....17
6.3.1 Forventet resultat....18
6.3.2 Resultat....19
6.3.3 Oppståtte feil....20
6.4 Konklusjon....20
7 Referanser....21

Figurliste

Figur 1: Busslinjer....5
Figur 2: Timingdiagram fra ADC, SDOUTX gir 2x2 kanaler....6
Figur 3: Inndata fra 4 kanaler....6
Figur 4: Forventede verdier, her på heksadesimalform....7
Figur 5: Utklipp som viser hvordan data fra 4 kanaler blir tilgjengelig....7
Figur 6: Oppsett....8
Figur 7: Testoversikt....10
Figur 8: Hastighetsmåleren under test....11
Figur 9: Graf over hastigheten....12
Figur 10: Wireshark....12
Figur 11: Testoversikt og sammenslåtte moduler....14
Figur 12: Avlesning av fil....16
Figur 13: Testoppsett....17

Forkortelser og akronymer

FPGA	-	Field Programmable Gate Array
FIFO	-	First In, First Out
TP	-	Twisted Pair
AMT	-	Avalon Master Template
I^2S	-	Inter-IC Sound
LRCK	-	Left/Right clock
MSB	-	Mest signifikante bit
DMA	-	Direct Memory Access
MM	-	Master Template
UART	-	Universal asynchronous receiver/transmitter
CPU	-	Central Processing Unit
UDP	-	User Datagram Protocol
RTL	-	Register Transfer Level
AD	-	Analog til digital
SDRAM	-	Synchronous dynamic random access memory

1 Introduksjon

Dette dokumentet er en samling av testrapporter som omhandler moduler på FPGA-kortet (Field Programmable Gate Array) [5]. Hver testrapport kan utføre en eller flere tester samtidig om dette er naturlig. Dette er beskrevet ut fra test-ID-ene i overskriften for hver testrapport.

1.1 Utførte tester

Test-ID:
T.P.3
T.P.4
T.P.7 og T.P.1
T.P.5 og T.P.2
T.P.6

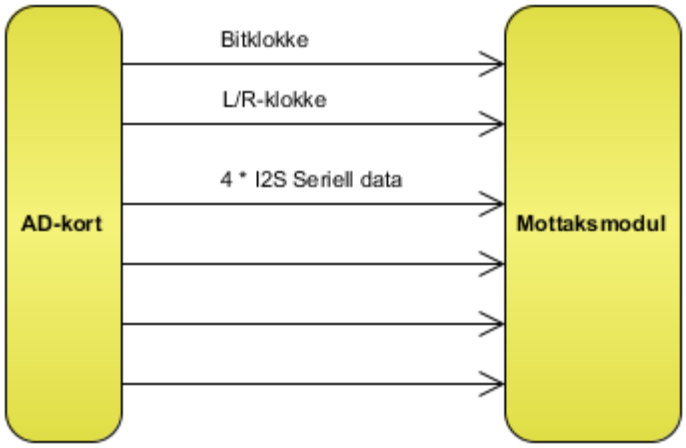
2 Kjøring av T.P.3

Dette dokumentet beskriver testprosedyren for hvordan mottaksmodulen på FPGA-en leser fra I^2S-bussene (Inter-IC Sound). Inn til modulene kommer det 4 I^2S-datalinjer (2 busser hver), en bitklokke-linje og en LRCK-linje (Left/Right clock). Testen skal skal vise at modulen greier å lese korrekt fra I^2S-linjene. Vi tar ikke for oss hvordan avlest data skal behandles videre.

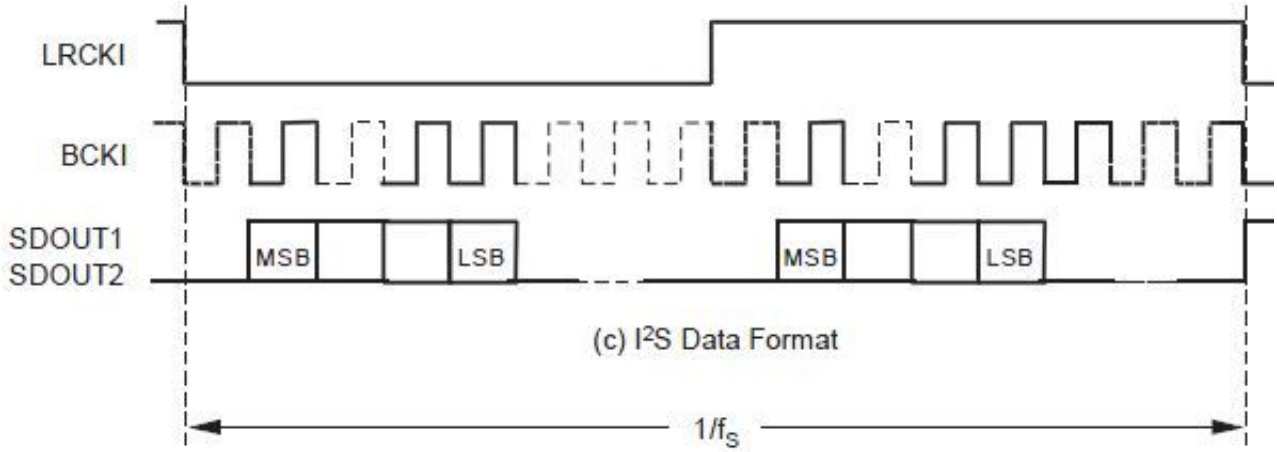
2.1 Testutstyr

For å utføre denne testen bruker vi en prototype av modulen som skal testes, skrevet i VHDL, simuleringsverktøyet Modelsim, en testbenk (VHDL) som genererer de inngangssignalene vi trenger for å verifisere funksjonaliteten, og databladet til AD-konverteren (Analog til digital) TI PCM4204 [1]. Modulprototypen inneholder koden vi bruker for å lese bussen, samt en utgangsvektor som lar oss se at den leser av korrekt.

2.2 Visuell testoversikt



Figur 1: Busslinjer



Figur 2: Timingdiagram fra ADC, SDOUTX gir 2x2 kanaler.

2.3 Testbeskrivelse

Vi utførte testen ved å sende databit serielt inn på alle busslinjene. LRCK-linje og datalinje synkroniserte vi mot negativ bitklokke-flanke som spesifisert i timingdiagrammet. Vi genererte nødvendige signaler med testbenken, og simulerte/visualiserte med Modelsim.

Dataene vi har sender inn på de serielle datalinjene vises i diagrammet under. Vi sender slik at vi kan se om modulen reagerer korrekt på LRCK-linja. Overflødige bits skal ignoreres.

Inndata	Nibble 1				Nibble 2				Nibble 3				Nibble 4			
bit:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
bus1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1
bus2	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1
bus3	1	1	1	1	0	0	1	1	0	0	1	1	1	1	0	0
bus4	1	1	0	0	1	1	0	0	0	0	1	1	0	0	0	0

Nibble 5				Nibble 6				Nibble 7				Nibble 8			
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	1	1	1	1	1	1	X	X	X	X	X	X	X	X
1	1	0	0	0	0	1	1	X	X	X	X	X	X	X	X
1	1	0	0	1	1	0	0	X	X	X	X	X	X	X	X
1	1	1	1	1	1	1	1	X	X	X	X	X	X	X	X

Figur 3: Inndata fra 4 kanaler

2.3.1 Forventet resultat

Forventet resultat er en utgangsvektor som består av en sample (24 bit) for en kanal på hver datalinje. Vektoren skal ha de verdiene vi sender serielt inn, og vise at modulen leser data korrekt og kontinuerlig. Unødvendige bits skal være ignorert.

Forventede verdier er vist i tabellen under. For å gjøre tabellen og simuleringen lesbare er

utgangsverdiene vist på heksadesimal form.

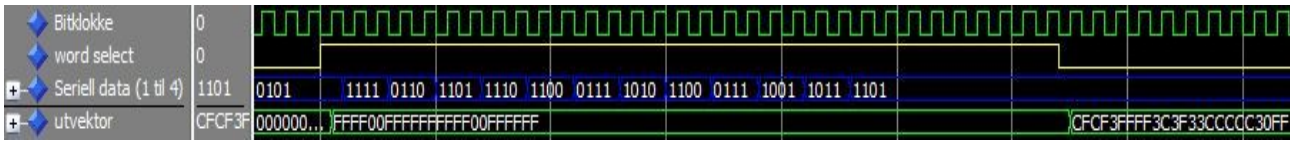
Utdata								
Nibble (4 bit)	1	2	3	4	5	6	7	8
bus1	C	F	C	F	3	F	X	X
bus2	F	F	F	3	C	3	X	X
bus3	F	3	3	C	C	C	X	X
bus4	C	C	3	0	F	F	X	X

Figur 4: Forventede verdier, her på heksadesimalform

Utgangsvektoren skal tilsvare disse verdiene i en vektor, lest fra venstre til høyre og linje for linje. X-er skal ikke være med.

2.3.2 Resultat

Resultatet av simuleringen viser at modulen leser fra I^{2}S-bussen som spesifisert i databladet til ADC-en [1]. Man ser av bølgediagrammet under at hver MSB (Mest signifikante bit) (første i rekka) leses en klokkepuls etter at LRCK har hatt flanke. Dette er korrekt oppførsel. Simuleringen viser også at avlesing skjer kontinuerlig, og at utgangsvektor er klar rett etter hver lesing av bussen.



Figur 5: Utklipp som viser hvordan data fra 4 kanaler blir tilgjengelig.

2.3.3 Oppståtte feil

Testen har ikke avdekket noen feil eller mangler.

2.4 Konklusjon

Vi kan konkludere med at avlesing fra I^{2}S-bussene skjer i henhold til databladet [1] til ADC-en vi bruker (I^{2}S-modus). I tillegg ignoreres overflødige bits som kommer inn. Testen tar ikke for seg hvordan avlest data behandles videre.

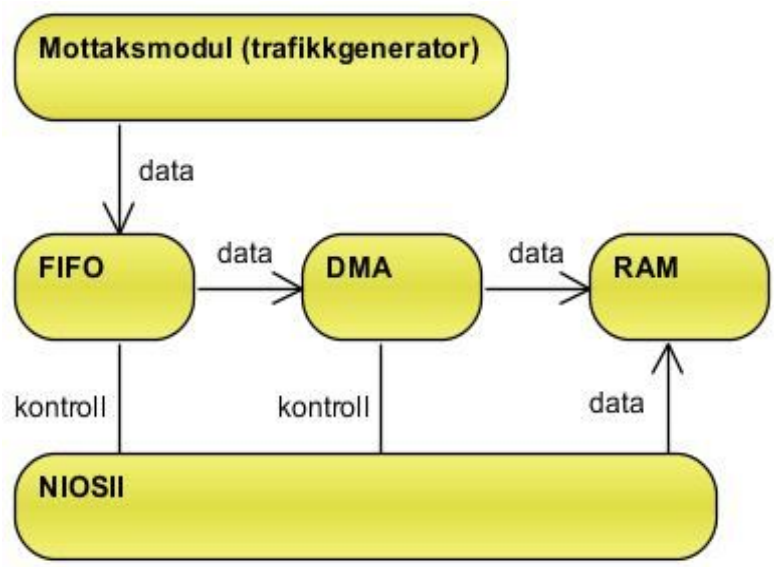
3 Kjøring av T.P.4

Dette dokumentet beskriver testprosedyren for FIFO (First In, First Out) og DMA-løsningen (Direct Memory Access) som befinner seg imellom mottaksmodulen og RAM-en i systemet vårt. Vi bruker ferdigmodulene «On-chip FIFO» og «DMA controller». Begge kommuniserer over Avalon-bussen med 32 bits bredde MM (Master Template) grensesnitt. Hensikten med testen er å verifisere at vi greier å sende krevet gjennomstrømming igjennom FIFO/DMA-oppsettet.

3.1 Testutstyr

Selvlaget trafikkgenerator, UART-kommunikasjon (Universal asynchronous receiver/transmitter) til NIOSII CPU (Central Processing Unit).

3.2 Visuell testoversikt



Figur 6: Oppsett

3.3 Testbeskrivelse

For å utføre testen setter vi trafikkgeneratoren opp til å skape den trafikken den det ferdige systemet kommer til å påføres (96 kHz · 24 bit · 8 kanaler). Vi kjører samtidig et program på CPU-en

som veksler på å initiere DMA-transaksjoner og gjøre andre ting. Vi er enda ikke sikker på hvor lange pauser CPU-en vil måtte ta, men vi har som minstekrav at DMA bare skal kunne være aktiv på 50 % av klokkesyklusene. For å vite om FIFO-nivå når for høye verdier, sjekker vi nivået for hver transaksjon, og skriver ut (hvis for høyt) igjennom UART.

DMA-en skriver til et innbuffer av minneblokker på FPGA-brikken. I det ferdige systemet vil vi trolig skrive til SDRAM (Synchronous dynamic random access memory), men denne testen omfatter bare FIFO/DMA oppsettet. I de periodene CPU skal gjøre «andre ting» bruker vi i denne testen C-funksjonen memset() til å skrive nye verdier til innbufferet. DMA er satt opp til å overføre 1 kB pr overføring.

3.3.1 Forventet resultat

Forventet resultat er at CPU skal kunne kjøre memset() mer enn et par ganger for hver DMA-overføring uten at FIFO flyter over.

3.3.2 Resultat

Kjøring av testen viser at for hver DMA overføring (1 kB) kan vi overskrive innbufferet 33 ganger (33 kB) uten at FIFO flyter over. I tillegg kommer også sjekking av fyllnivå.

3.3.3 Oppståtte feil

Testen har ikke avdekket noen feil.

3.4 Konklusjon

Kjøring av testen viser at vi kan ha ønsket gjennomstrømming til tross for at DMA bare er aktiv på maks 50 % av klokkesyklene (som spesifisert i testspesifikasjonen). Testen viser at den bare trenger å være aktiv på ≈3 %. Testen må derfor sies å være vellykket.

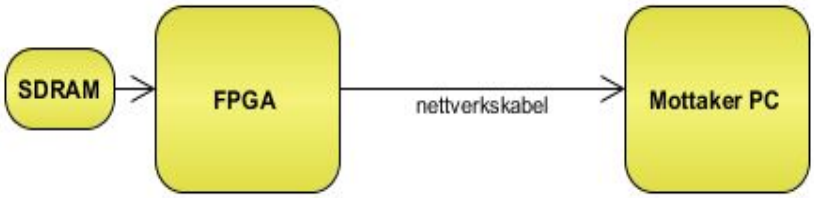
4 Kjøring av T.P.7, T.P.1

Dette rapporten beskriver testen av FPGA-modulen for pakking og videresending av data. Samtidig vil testen vise at overføringen over Ethernet skjer ved hjelp av UDP-protokollen (User Datagram Protocol).

4.1 Testutstyr

FPGA-kort, TP-kabel (Twisted Pair), PC med mottakerprogram, NetWorx [3] og Wireshark [4].

4.2 Visuell testoversikt



Figur 7: Testoversikt

4.3 Testbeskrivelse

Målet med denne testen er å overvære at overføring av data fra SDRAM til mottaker-PC skjer i henhold til krav vi har satt til hastighet. På mottaker-PC har vi laget et C-program som kobler seg til FPGA-kortet via socket-kommunikasjon. Når FPGA og PC er koblet sammen er programmet klart til å ta i mot UDP-pakker fra FPGA-kortet. For å måle hastigheten på dataene som blir overført bruker vi et gratis program fra SoftPerfect Research, som heter NetWorx. Dette programmet kan måle nettverkstrafikk i sanntid og passer derfor godt til dette formålet. Samtidig vil vi kjøre et pakkesnifferprogram (Wireshark) som kan fortelle oss at overføringen skjer ved hjelp av UDP-protokollen.

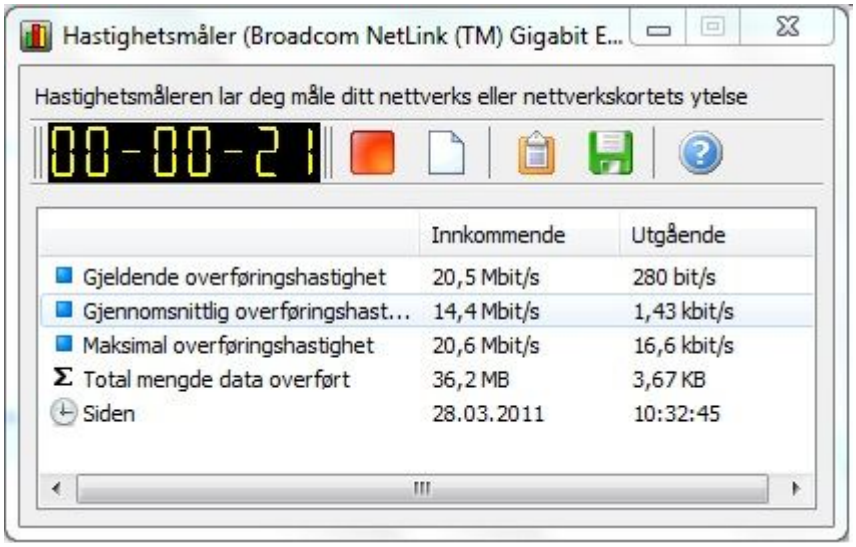
På forhånd har vi bekreftet at data som blir oversendt faktisk er plassert på SDRAM-brikken. I BSP-editoren i NIOSII-programvaren er både stack og heap til NIOSII-prosessoren plassert på SDRAM. I tillegg ser vi med mottaksprogrammet at det vi mottar er det samme som vi sender ut.

4.3.1 Forventet resultat

Innkommende data fra AD-kortet har følgende spesifikasjoner: Frekvens 96 kHz, Oppløsning 24 bit og 8 kanaler. Hvis man multipliserer disse tallene får man antall bit per sekund data som blir overført (96000 Hz · 24 bit · 8 = 18 432 000 bit/s). Vi ønsker at datastrøm ut fra FPGA-kortet skal være minst like stor som datastrøm inn. For at testen skal godkjennes må hastigheten på dataoverføringen derfor være over 18,4 Mb/s. Under kjøring av testen sender vi et buffer på 1024 byte, hundre tusen antall ganger.

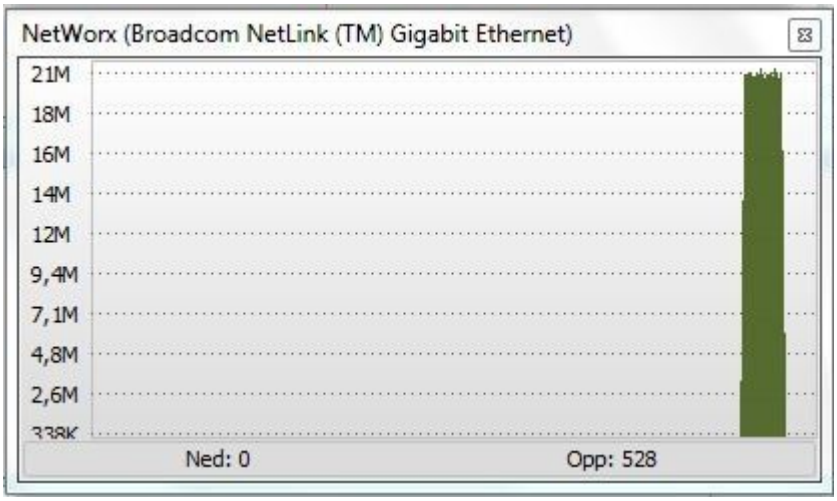
4.3.2 Resultat

Som bildet under viser er målt gjennomstrømming større enn kravet vårt. Det ble fort funnet ut at gjentatte kall til «printf» i C-koden holdt overføringshastigheten nede. Disse kallene ble derfor tatt bort etter at vi visste at overføringen skjedde korrekt.

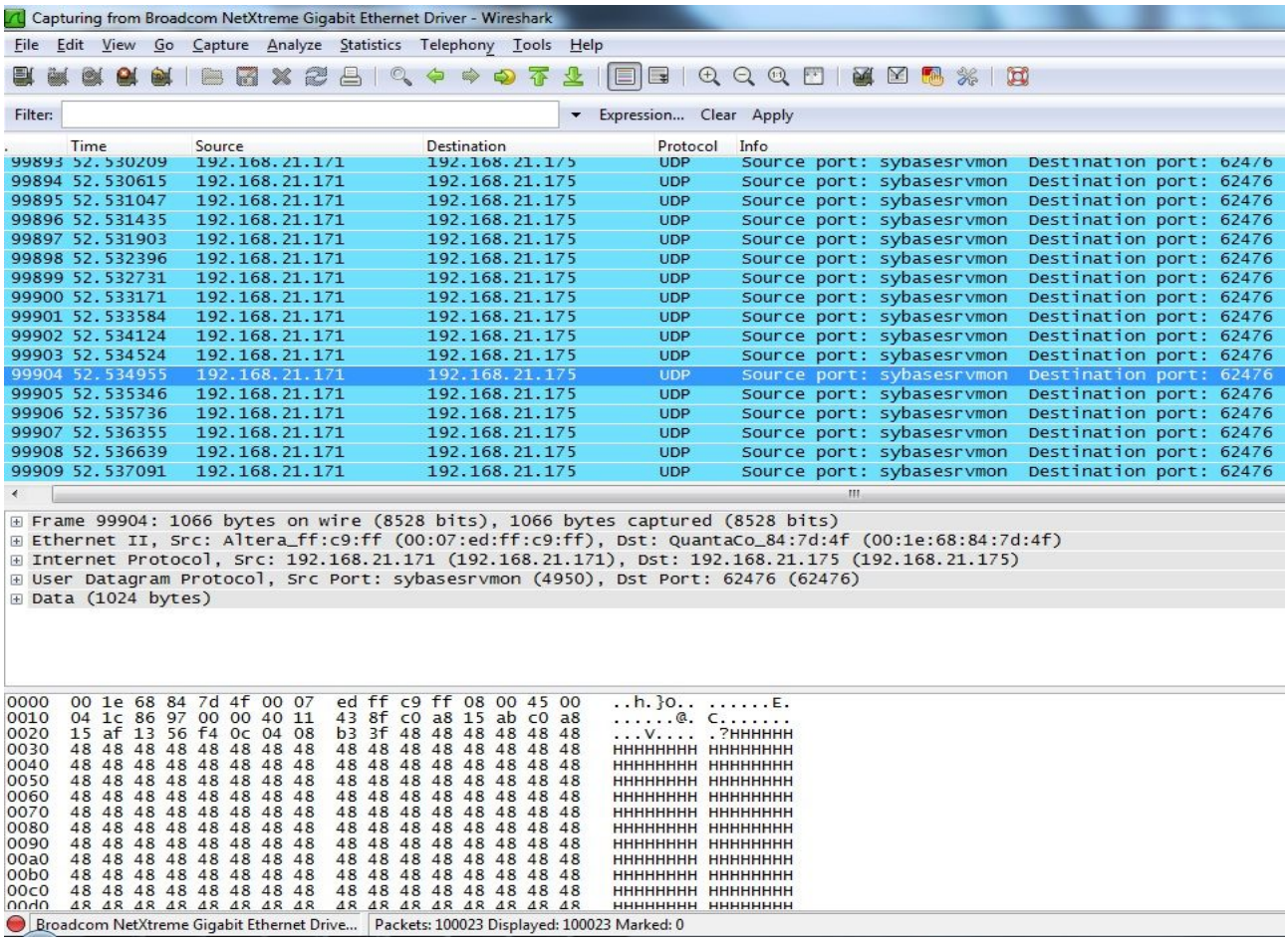


Figur 8: Hastighetsmåleren under test

Hastigheten på overføringen holdt seg jevnt rundt 20,3 til 20,5 Mbit/s under hele testen. Som også grafen under viser er det ingen store hastighetsforskjeller under overføringen.



Figur 9: Graf over hastigheten



Figur 10: Wireshark

Pakkesnifferprogrammet overvåket alle pakker som ble sendt under hele testen, her kunne vi se klart at alle mottatte pakker var av typen UDP. Bildet over er fra programmet Wireshark og viser noen av de mottatte pakkene.

4.3.3 Oppståtte feil

Testen har ikke avdekket noen feil.

4.4 Konklusjon

Denne testen har vist at prosjektets krav til overføring av UDP-pakker til mottaker-PC er tilfredsstillende. Hastighetsmåleren viser at overføringen av data skjer fortere enn innkommende pakker fra AD-kortet. Wireshark viser at alle mottatte datapakker fra FPGA-kortet bruker UDP-protokollen. Begge besvarte tester er gjennomført som forventet.

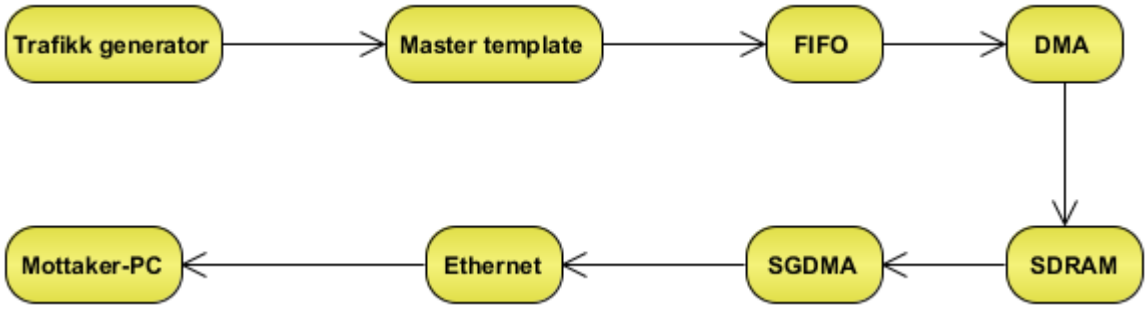
5 Kjøring av T.P.5, T.P.2

Denne rapporten beskriver test av sammenslåtte moduler på FPGA-kortet. Hvilke moduler vises best i visuell testoversikt. Samtidig vil vi teste at dataene blir lagret til fil på en mottaker-PC. Vi har fra før testet alle moduler hver for seg. Denne testen vil avdekke eventuelle feil etter sammenslåingen. Mottaksmodulen er den eneste modulen som ikke er representert i sammenheng med denne testen. Denne vil vi slå sammen med resten av systemet når vi har mulighet til å teste med AD-kortet. Selve mottaksmodulen isolert sett er testet i T.P.4.

5.1 Testutstyr

FPGA-kort, mottaker-PC, TP-kabel.

5.2 Visuell testoversikt



Figur 11: Testoversikt og sammenslåtte moduler

5.3 Testbeskrivelse

For at vi skal kunne bevise at all data blir overført uten korrupsjon eller tap gjennom alle moduler til fil, har vi laget en trafikkgenerator. Denne sender ut seks 32 bits pakker som hver teller oppover fra 0 til 233 gjentatte ganger. Når vi så leser av filen er det lett å se om vi mangler noen tall og dermed har mistet data.

5.3.1 Forventet resultat

Innholdet i filen på mottaker-PC skal være en sammenhengende tallrekke fra 0 til 233.

5.3.2 Resultat

Filen som inneholder data fra FPGA-kortet er ikke direkte avlesbart i en tekstbehandler, da innholdet er lagret i binærformat. Vi benytter oss derfor av «Od», et program i Cygwin som formaterer innholdet for oss. Siden vi sender hvert tall som 32 bit, vil vi at hver 32. ordbredde skal gjøres om til desimaltall. Resultatet vises i Figur 12. Som man ser er det ingen tall som mangler eller rekkefølgen er stokket om.

5.3.3 Oppståtte feil

Det ble avdekket en feil i trafikkgeneratoren før denne testen ble gjennomført. Vi kunne lese av filen at noen tall manglet. Det viste seg at dette kun var kodefeil. Feilen gikk på at tellingen fortsatte selv når trafikkgeneratoren ikke skulle sende data. Trafikkgeneratoren hadde da telt for langt da den fikk mulighet til å sende på nytt. Etter en liten endring i VHDL-koden, ble problemet løst og resultatet sees i Figur 12.

5.4 Konklusjon

Vi konkluderer her med at alle de sammenslåtte modulene og lagring til fil fungerer tilfredsstillende, og er i henhold til våre krav.

/cygdrive/c/altera

0000000	0	1	2	3
0000020	4	5	6	7
0000040	8	9	10	11
0000060	12	13	14	15
0000100	16	17	18	19
0000120	20	21	22	23
0000140	24	25	26	27
0000160	28	29	30	31
0000200	32	33	34	35
0000220	36	37	38	39
0000240	40	41	42	43
0000260	44	45	46	47
0000300	48	49	50	51
0000320	52	53	54	55
0000340	56	57	58	59
0000360	60	61	62	63
0000400	64	65	66	67
0000420	68	69	70	71
0000440	72	73	74	75
0000460	76	77	78	79
0000500	80	81	82	83
0000520	84	85	86	87
0000540	88	89	90	91
0000560	92	93	94	95
0000600	96	97	98	99
0000620	100	101	102	103
0000640	104	105	106	107
0000660	108	109	110	111
0000700	112	113	114	115
0000720	116	117	118	119
0000740	120	121	122	123
0000760	124	125	126	127
0001000	128	129	130	131
0001020	132	133	134	135
0001040	136	137	138	139
0001060	140	141	142	143
0001100	144	145	146	147
0001120	148	149	150	151
0001140	152	153	154	155
0001160	156	157	158	159
0001200	160	161	162	163
0001220	164	165	166	167
0001240	168	169	170	171
0001260	172	173	174	175
0001300	176	177	178	179
0001320	180	181	182	183
0001340	184	185	186	187
0001360	188	189	190	191
0001400	192	193	194	195
0001420	196	197	198	199
0001440	200	201	202	203
0001460	204	205	206	207
0001500	208	209	210	211
0001520	212	213	214	215
0001540	216	217	218	219
0001560	220	221	222	223
0001600	224	225	226	227
0001620	228	229	230	231
0001640	232	233	0	1

Figur 12: Avlesning av fil

6 Kjøring av T.P.6

Vi tester om mottaksmodulen er i stand til å generere de data og kontrollsignaler som trengs for å sende mottatt data videre til FIFO. Siden FIFO er på Avalon-bussen, bruker vi en «Avalon master template», eller AMT (Avalon Master Template), for å skrive til bussen. Mottaksmodulen må derfor generere signaler for å kommunisere med denne.

6.1 Testutstyr

RTL-simuleringsverktøyet (Register Transfer Level) Modelsim, og egenlaget testbenk skrevet i VHDL. Mottaksmodulen konfigureres slik den skal være i ferdige systemet.

6.2 Visuell testoversikt

Figur 13: Testoppsett

6.3 Testbeskrivelse

Vi satt opp testbenkene til å generere klokkesekvens og data som det kan komme inn fra I^2S-bussen. For å sende til AMT må vi generere et sett med kontrollsignaler. Disse er spesifisert i databladet til AMT [2] og vi bruker de som vist i Tabell 1

Tabell 1: Signaler til og fra Avalon Master template

user_write_buffer	Holdes høy for å skrive til AMT sitt interne buffer. Vi holder denne høy i tre perioder etter hver flanke på Word Select, og skriver med det 3x4 byte (4 kanaler á 3 byte).
user_buffer_data	Data som skrives når uwb er høy. Se rad over.
control_base	Avalonadressen det skal skrives til. For AMT har FIFO adressen 00000000.
control_done	Signal fra AMT om at den har sendt alt som ligger i internt buffer. Ignoreres av oss.
control_fixed_location	Indikerer at vi vil sende til fast adresse i kontrast til å inkrementere den.
control_go	På dette signalet starter AMT å sende fra det interne bufferet.
control_length	Hvor mange bytes som skal sendes. Vi sender 3 x 4 = 12.
user_buffer_full	Indikerer at det interne bufferet er fullt. Vi sjekker ikke dette siden vi sender i fast tempo med nok tid til at AMT får sendt ferdig.

Mottaksmodulen mottar 3 gyldige og en ugyldig byte for hver kanal (4 kanaler av gangen), og sender disse som tre sekvensielle ord.

Tabell 2: Data inn på serielle datalinjer. Verdier er i heksadesimal.
Fargekoder for sammenlikning med Tabell 3.

	Første byte	Andre byte	Tredje byte	Fjerde byte (dont care)
Seriell Data 1	0 2	0 0	0 0	0 0
Seriell Data 2	0 0	0 0	0 0	0 0
Seriell Data 3	0 0	0 0	0 0	0 0
Seriell Data 4	0 4	0 0	0 0	0 0

6.3.1 Forventet resultat

Vi forventer at data ut skal være det samme som data inn – trukket sammen til tre ord. Fargekodene i tabellene viser hvordan bytene skal arrangeres for 4 kanaler / tre ord av gangen.

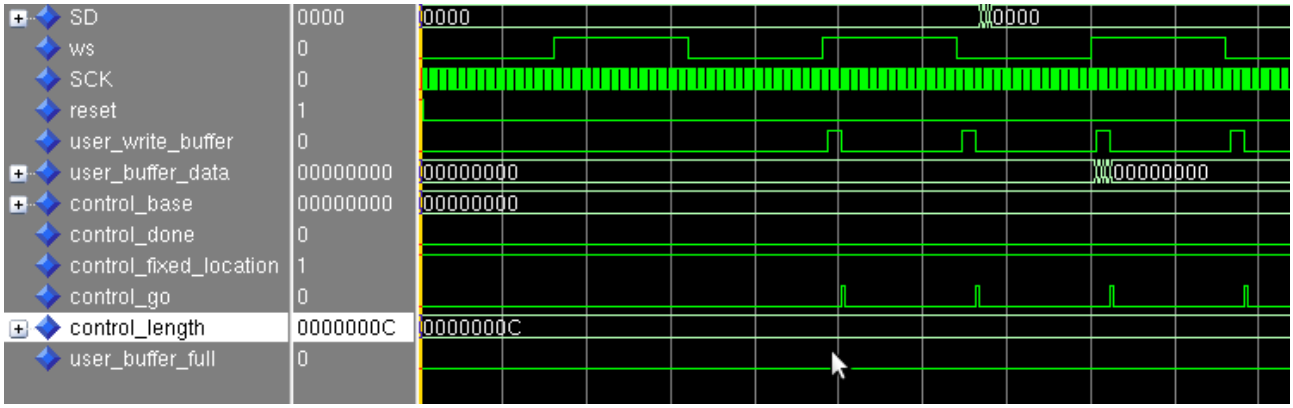
Tabell 3: Videre til systemet, fargekoder for sammenlikning med Tabell 2.

	Første byte	Andre byte	Tredje byte	Fjerde byte
Første ord	0 2	0 0	0 0	0 0
Andre ord	0 0	0 0	0 0	0 0
Tredje ord	0 0	0 4	0 0	0 0

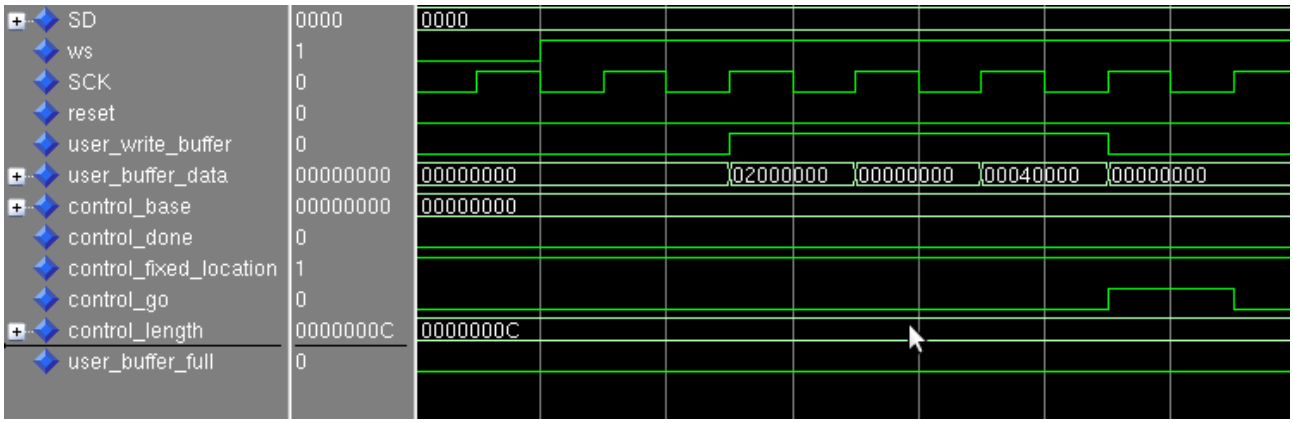
Vi forventer også at user_buffer_write skal være høy og at user_write_data skal ha riktige words i tre perioder kort tid etter hver flanke på LRCK. Deretter skal control_go være høy i en periode. Resten av signalene er statiske og beskrevet i Tabell 1. Mottak fra buss er allerede testet og beskrevet i testrapport T.P.3

6.3.2 Resultat

RTL simulering av modulen viser at data som sendes til AMT er det samme som kommer inn over I²S-bussene. Data samles korrekt og de rette kontrollsignalene genereres korrekt og til riktig tid.



Figur 14: Mottak starter etter første negative WS flanke, og fortsetter deretter. Write_buffer går høy for hver WS (LRCK) flanke, og deretter control_go.



Figur 15: Samme simulering - viser her signalene som videresender mottatte data til AMT.

6.3.3 Oppståtte feil

Ingen.

6.4 Konklusjon

Testen har ikke avdekket noen feil, og mottaksmodulen greier tilsynelatende å sende data til FIFO.

7 Referanser

[1] *PCM4204 datasheet.* Texas Instrument, 2004.

[2] *Avalon-MM Master Templates readme.* Altera, 2008.

[3] Softperfect. Networx - Free Bandwidth Monitoring and Usage Reporting. http://www.softperfect.com/products/networx/ . Sist besøkt: 25.05.11.

[4] Wireshark protocol analyzer. http://www.wireshark.org . Sist besøkt: 25.05.11.

[5] Prosjektgruppe 2011-09, *P.4 Testspesifikasjon*, MMS, 2011

POSICOM

HØGSKOLEN i Buskerud

MMS Multimikrofonsystem

Testrapport - ytelse		
Dokumenttype	Dokumentnummer	Versjonsnummer
Test	T.6	1.a
Dato sist endret	Frigitt dokument?	Dokumentansvarlig
27.05.2011	Ja	Knut Olav Skyttemyr
Gruppemedlemmer:		
Kathrine Kaasa, Knut Olav Skyttemyr, Jardar Hvidsten, Fredrik Bakke, Erlend Rauk Aasland		

Innholdsliste

1 Introduksjon....3
1.1 Relevante tester....3
2 Testutstyr....4
3 Kjøring av T.Y.1....5
3.1 Forventet resultat....5
3.2 Resultat....5
3.3 Konklusjon....6
4 Kjøring av T.Y.2....7
4.1 Forventet resultat....7
4.2 Resultat....7
4.3 Konklusjon....8
5 Kjøring av T.Y.3....9
5.1 Forventet resultat....9
5.2 Resultat....9
5.3 Konklusjon....10
6 Oppsummering....11
7 Referanser....12

Figurliste

Figur 1: Støy internt på AD-kortet uten filter. FS = 96 kHz, 24 bits oppløsning....5
Figur 2: Støy internt på AD-kort med filtrering. FS = 96 kHz, 24 bits oppløsning....6
Figur 3: Mottaksprogram etter langtidstest....7
Figur 4: Støy internt på AD-kortet med spenningsforsyning fra FPGA. FS = 96 kHz, 24 bits oppløsning....9
Figur 5: Støy internt på AD-kortet med eksterne spenningsforsyninger. FS = 96 kHz, 24 bits oppløsning....10

Forkortelser og akronymer

FPGA	-	Field Programmable Gate Array
SNR	-	Signal-Noise-Ratio
AD	-	Analog til digital

1 Introduksjon

Dette dokumentet tar for seg testene som går på systemets ytelse [1]. Vi har ikke fått noen krav på dette, men det er uansett interessant å se hvor godt systemet klarer disse testene.

1.1 Relevante tester

Test-ID:
T.Y.1
T.Y.2
T.Y.3

2 Testutstyr

- FPGA-kort [2]
- AD-kort [3]
- Spenningsforsyninger: Delta Elektronika E030-3 og GW GPS-3030
- Kabler

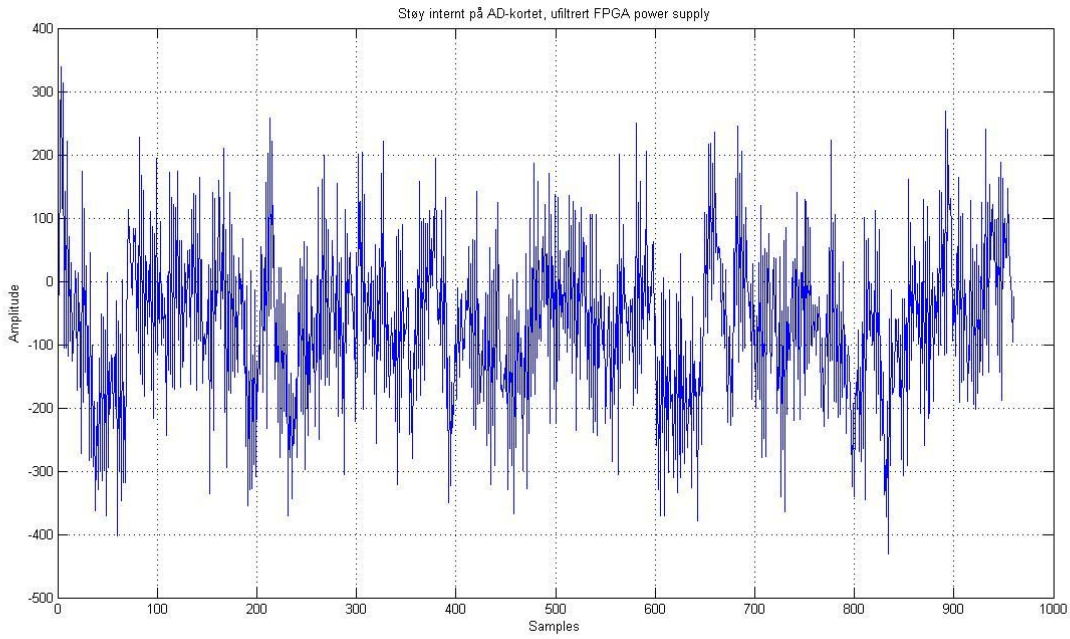
3 Kjøring av T.Y.1

Mål signalstøyen på 5 VDC- og 3,3 VDC-linjene før filteret mens FPGA-kortet belastes. Mål signalstøyen på de samme linjene etter filteret. Sammenlign resultatene og verifiser at filteret har en dempende effekt på signalstøyen.

3.1 Forventet resultat

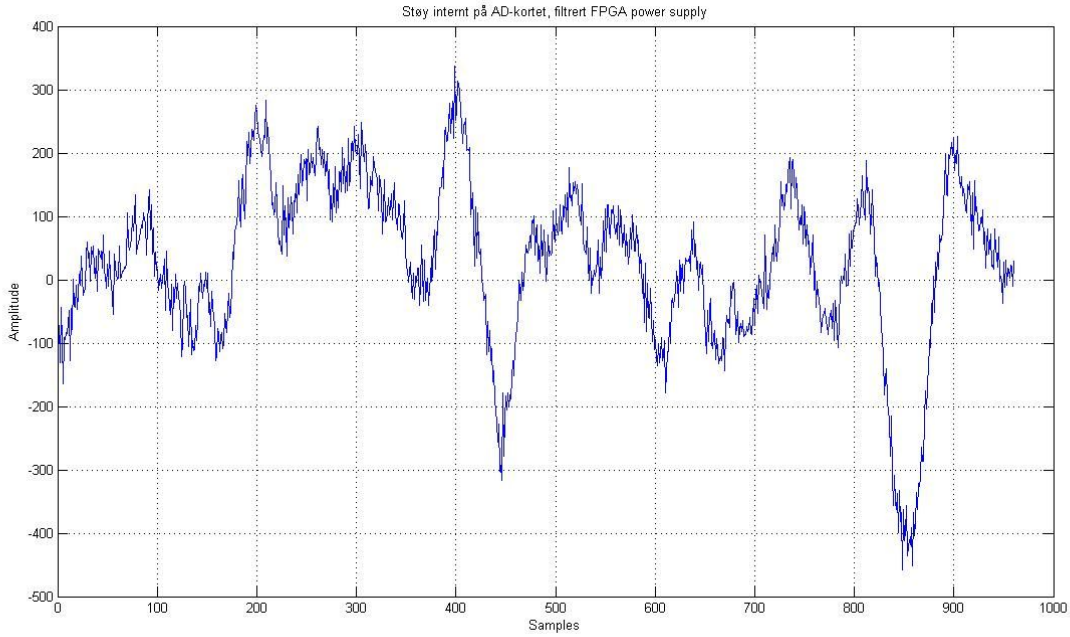
Vi forventer at høyfrekvent støy skal bli lavere ved bruk av filteret.

3.2 Resultat



Figur 1: Støy internt på AD-kortet uten filter. F_S = 96 kHz, 24 bits oppløsning.

Se Figur 1 for et bilde av støyen på AD-kortet uten støydempingsfilteret.



Figur 2: Støy internt på AD-kort med filtrering. F_S = 96 kHz, 24 bits oppløsning.

Se Figur 2 for et bilde av støyen etter at spenningsforsyningen fra FPGA (Field Programmable Gate Array) er filtrert. Som vi ser er høyfrekvent støy redusert betraktelig. Men vi ser samtidig at støyen i lavere frekvenser fortsatt ligger i de 10 laveste bitene (± 512).

3.3 Konklusjon

Støydempingsfilteret fungerer som det skal. Høyfrekvent støy fra FPGA er merkbart lavere ved bruk av filteret. Men skulle vi oppnådd bedre SNR (Signal-Noise-Ratio) i systemet vårt, så må knekkfrekvensen til filteret presses lengre ned. Dette hadde vi problemer med å få til på grunn av størrelser på komponenter og så videre. Se designdokumentet for flere detaljer angående dette [4].

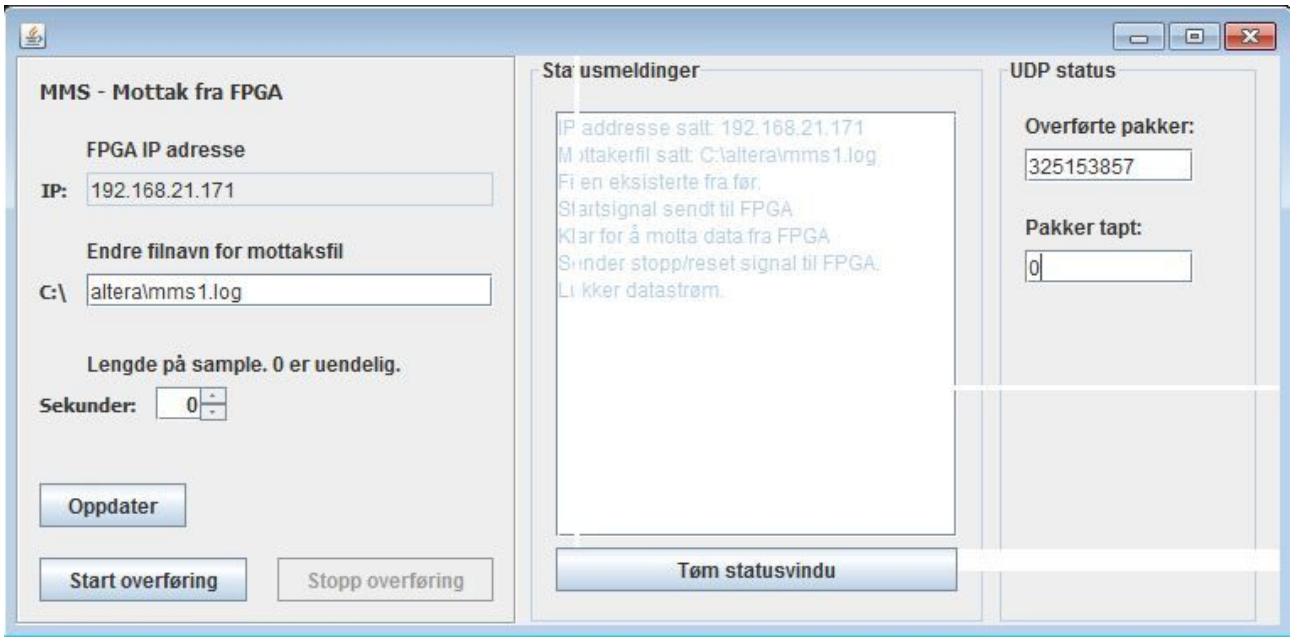
4 Kjøring av T.Y.2

I denne testen kjører vi en langtidstest. Systemet skal kjøres på maksimal belastning over 12 timer uten å krasje. Vi ønsker å vite hvor mye data vi mister ved å la systemet kjøre over lang tid, og registrerer derfor antall datapakker vi mister underveis. Testen starter vi på ettermiddagen den 16. mai og lar systemet kjøre gjennom nasjonaldagen før vi ser resultatet av testen morgenen den 18. mai.

4.1 Forventet resultat

Under korte tester vi har gjort underveis i utviklingen, har vi noen ganger oppdaget en til to pakketap på korte kjøringer av systemet. Stort sett skjer overføringer uten pakketap, og dette håper vi også holder seg over lengre tidsperioder. Noen få tap vil være akseptabelt da oddsene for et pakketap skal oppstå under opptak av en sjokkpuls/munningspuls fra et våpen, vil være svært liten.

4.2 Resultat



Figur 3: Mottaksprogram etter langtidstest

Etter 40,1 timer konstant kjøring av systemet var det overført 325153857 antall pakker på

størrelsen 1008 byte. Det var registrert ingen pakketap. Figur 3 viser mottaksprogrammet etter endt test.

4.3 Konklusjon

Resultatet av testen betyr at systemet vårt er robust og tåler jevn belastning over lengre tidsperioder. Dersom systemet skulle bli brukt operativt er en slik belastning regnet som normal belastning og derfor svært viktig at systemet er stabilt.

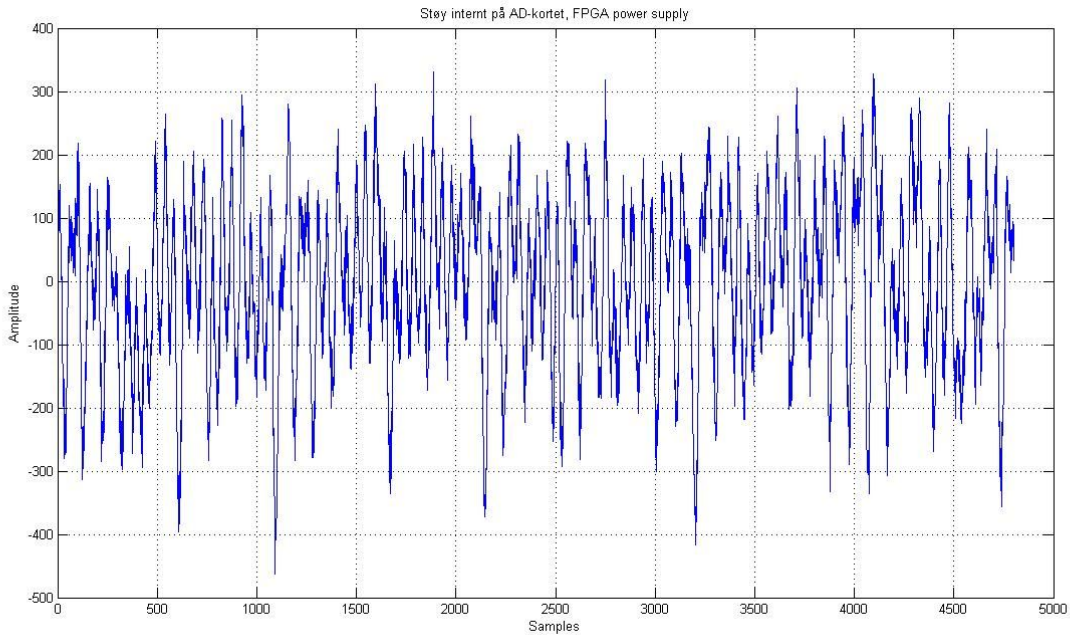
5 Kjøring av T.Y.3

Sammenligning av støy med de to forskjellige spenningsforsyningsløsningene. Gjør opptak uten mikrofoner koblet til, altså måling av støy internt på kortet. Sammenlign maksutslaget på støyen mellom de to forsyningsløsningene. Denne testen tar ikke med støy fra mikrofonene.

5.1 Forventet resultat

Vi forventer at det er mer støy på forsyningen fra FPGA-kortet. Dessuten er jordplanet på AD-kortet (Analog til Digital) mest tilpasset den eksterne spenningsforsyningen. I følge databladet til ADC-en er maksimum dynamisk rekkevidde 118 dB (ca. 20 bit). Derfor vil det ligge støy i de 4-5 laveste bitene uansett hvor godt eller dårlig AD-kortet er laget.

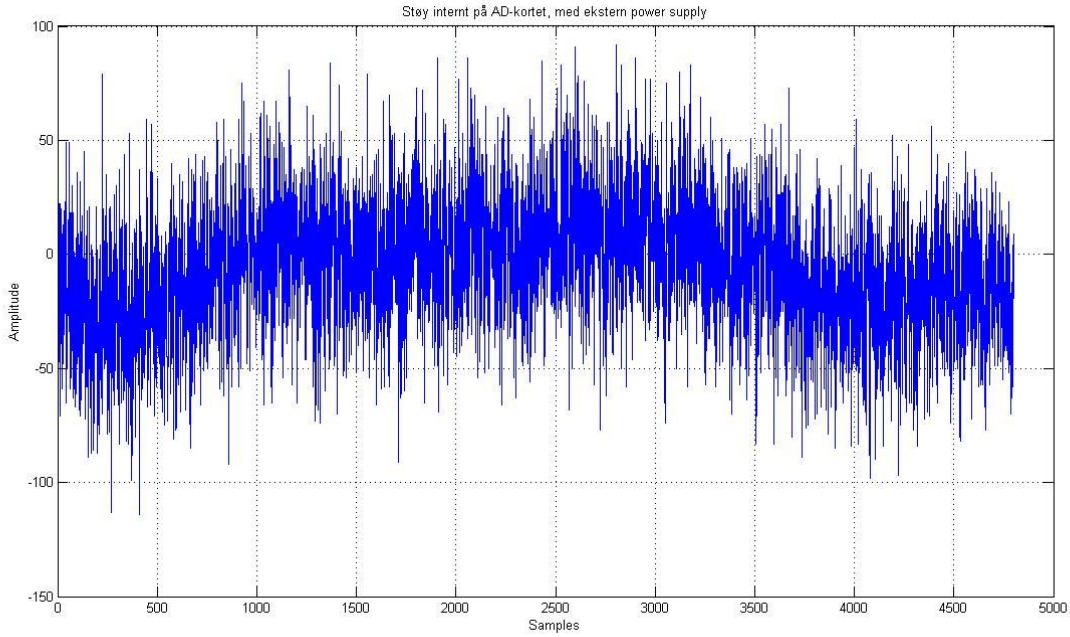
5.2 Resultat



Figur 4: Støy internt på AD-kortet med spenningsforsyning fra FPGA. F_S = 96 kHz, 24 bits oppløsning

Som vi ser av Figur 4 er utslaget på støyen med spenningsforsyning fra FPGA godt innenfor ± 512 (2^{10} = 1024). Dette tilsier at det ligger støy i de 10 laveste bitene. Totalt har vi da en dynamisk

rekkevidde på 24 bit – 10 bit = 14 bit. Vi antar gaussisk amplitudefordeling på støyen, og kan da regne ut det dynamiske området til systemet [5]. SNR = 14 bit · 6 dB/bit – 8,5 dB = 75,5 dB.



Figur 5: Støy internt på AD-kortet med eksterne spenningsforsyninger. F_S = 96 kHz, 24 bits oppløsning.

Vi ser av Figur 5 at utslaget på støyen med ekstern spenningsforsyning ligger innenfor ± 128 (2^8 = 256). Det vil si en forbedring på 2 bit med eksterne spenningsforsyninger. Med gaussisk amplitudefordeling får vi da SNR = 16 bit · 6 dB/bit - 8,5 dB = 87,5 dB.

5.3 Konklusjon

Som forventet er det mindre støy og bedre SNR ved bruk av godt regulerte spenningsforsyninger. I databladet til ADC-en har de gjort målinger der støy pluss forvregning (THD+N) ligger på -103 dB. Vårt system ligger noe under dette. Men vi synes dette er akseptabelt med tanke på at det er det første kretskortet vi lager. Merk at dette er målinger uten støy fra mikrofon, men den burde være lavere en det vi finner internt på AD-kortet.

6 Oppsummering

Støydempingsfilteret fungerer som antatt, men for å forbedre SNR må knekkfrekvensen lengre ned.

Systemet fungerer som det skal over lengre tid. Vi opplevde ingen pakketap i løpet av 40 timers drift.

Som forventet senker støy fra FPGA-kortet SNR til systemet vårt. I denne testen oppnådde vi omtrent 14 bit oppløsning med spenningsforsyning fra FPGA og 16 bit oppløsning med eksterne spenningsforsyninger.

7 Referanser

[1] Prosjektgruppe 2011-09, *P.4 Testspesifikasjon.*, 2011

[2] *DE2-115 User Manual.*, Terasic, 2010

[3] Prosjektgruppe 2011-09, *S.2 Brukermanua.l*, 2011.

[4] Prosjektgruppe 2011-09, *S.1 Designdokument.*, 2011.

[5] Gudvangen, S., *Introduction to quantization.*, ATEK HiBu, 2008.