FFI RAPPORT

DIGITAL MULTISTATISK RADAR, SENDER OG MOTTAKER

JOHNSRUD Steinar, TANSEM Ivar

FFI/RAPPORT-2002/01931

FFIE/726/170

Godkjent
Kjeller 22 Mai 2002

John-Mikal Størdal
Forskningssjef

DIGITAL MULTISTATISK RADAR, SENDER OG MOTTAKER

JOHNSRUD Steinar, TANSEM Ivar

FFI/RAPPORT-2002/01931

FORSVARETS FORSKNINGSINSTITUTT
Norwegian Defence Research Establishment
Postboks 25, 2027 Kjeller, Norge

FORSVARETS FORSKNINGSINSTITUTT (FFI)
Norwegian Defence Research Establishment

P O BOX 25
NO-2027 KJELLER, NORWAY

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE
(when data entered)

REPORT DOCUMENTATION PAGE

1) PUBL/REPORT NUMBER	2) SECURITY CLASSIFICATION	3) NUMBER OF PAGES
FFI/RAPPORT-2002/01931	UNCLASSIFIED	27
1a) PROJECT REFERENCE	**2a) DECLASSIFICATION/DOWNGRADING SCHEDULE**	
FFIE/726/170	-	

4) TITLE

DIGITAL MULTISTATISK RADAR, SENDER OG MOTTAKER

(DIGITAL MULTISTATIC RADAR, TRANSMITTER AND RECEIVER)

5) NAMES OF AUTHOR(S) IN FULL (surname first)

JOHNSRUD Steinar, TANSEM Ivar

6) DISTRIBUTION STATEMENT

Approved for public release. Distribution unlimited. (Offentlig tilgjengelig)

7) INDEXING TERMS

IN ENGLISH:	IN NORWEGIAN:
a) Radar	a) Radar
b) Radar receivers	b) Radar mottaker
c) Radar transmitters	c) Radar sender
d) CW radar	d) CW radar
e)	e)

THESAURUS REFERENCE: **INSPEC 1999**

8) ABSTRACT

This report describes the receiver and the transmitter designed for verifying a multistatitic radar concept. The digital part of the receiver and the transmitter are based on FPGAs. The report also describes the software for controlling the system.

9) DATE	AUTHORIZED BY	POSITION
	This page only	
22 Mai 2002	John-Mikal Størdal	Director of Research

ISBN-82-464-0626-4

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE
(when data entered)

INNHOLD

DIGITAL MULTISTATISK RADAR, SENDER OG MOTTAKER

1 INNLEDNING

I prosjekt 726 Digital multistatisk radar er det utviklet en radarsender og -mottaker for å verifisere det multistatiske konseptet. Det er utviklet en sender og en mottaker slik at det pr dato er et bistatisk oppsett. I et bistatisk radaroppsett plasseres sender og mottaker atskilt med en fysisk avstand. Senderenheten som er utviklet i prosjektet sender kontinuerlig signal (CW) og mottakeren mottar eventuelt reflektert signal. All nødvendig signalprosessering er tenkt utført på mottakerenheten i sann tid. Se (1).

Dette dokument beskriver sender- og mottakersystemet, hardware og software. Det beskriver hvordan systemet er bygd opp og hvordan sender og mottaker virker på blokknivå. Det er utviklet et brukerprogram i prosjektet. Dette dokument beskriver også brukerinterfacet og de forskjellige funksjoner i programmet.

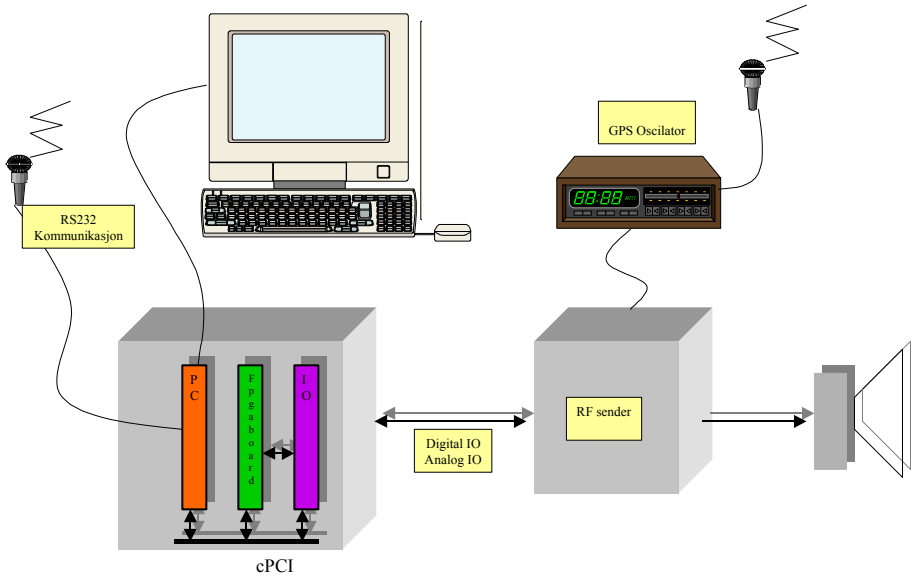
2 RADARSENDER OG -MOTTAKER EN SYSTEMBESKRIVELSE

Sender og mottaker er to selvstendige enheter, se Figur 2.1, men de er skjematisk relativt like, se Figur 2.2 og Figur 2.3. Sender og mottaker består av flere like moduler. Både sender og mottaker er bygd rundt et compact PCI system. I dette cPCI systemet har vi plassert et egenutviklet kretskort kalt FPGA-kort (Field Programmable Gate Array), se (2). Det er to slike kort som utgjør digital kodeutsendelse og digital datainnsamling. Det er i prosjektet også utviklet et kretskort som er kalt IO-kort, se (3). Dette kortet inneholder en D/A- og en A/D-modul. I både sender og mottaker er det plassert et FPGA-kort og et IO-kort i cPCI systemet. Disse kortene styres av en PC som er koblet til cPCI systemet igjennom en PCI bridge. Den analoge delen består av en RF-sender og en RF-mottaker, se henholdsvis Figur 2.2 og Figur 2.3. I senderenheten har vi en dataflyt fra digital kodeutsendelse gjennom IO-kortet og RF-sender før koden sendes ut på antennen. I mottakeren er dataflyten motsatt vei, fra antennen gjennom RF-mottaker videre til IO-kortet, før det innkomne radarsignal lagres i digital datainnsamlingsenhet. I forbindelse med parameteroppsett i systemet kommuniserer mottaker og sender ved hjelp av en RS232 link, se kapitel 2.3.
Synkronisering av datainnsamling med kodeutsending skjer via GPS-styrte rubidiumoscilatorer, se kapitel 2.4.

Figur 2.1 Bilde av sender- og mottakerrack. Sender til venstre.

2.1 Sender

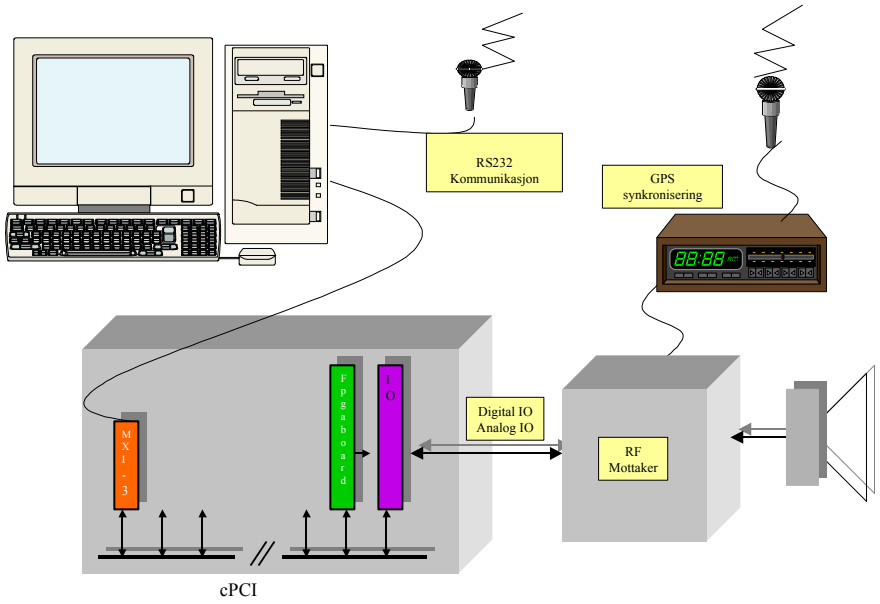
Med referanse til Figur 2.2 deler vi sender opp i følgende moduler. Det er en PC i systemet hvor all kontrollsoftware kjøres. Dette er en cPCI PC, det vil si et PC kort som kobles direkte i cPCI bakplanet. Videre er det en modul som tar seg av den digital kodeutsendingen. Denne modulen er lokalisert på FPGA-kortet. Digital kode sendes til I/O-kortet. Her rutes koden igjennom en D/A-modul før den, nå analoge koden, sendes til RF-sender, se (4), som legger koden i rett frekvensområde og forsterker signalet før den sendes ut på radarantennen.



Figur 2.2 Blokkskjema over senderenheten i multistatisk radar.

2.2 Mottaker

Med referanse til Figur 2.3 deler vi mottaker opp i følgende moduler. Det er en PC i systemet hvor all kontrollsoftware kjøres. Det er en RF-mottaker, se (4), der det innkommende signalet forsterkes og mixes ned til grunnfrekvens. Videre består mottakeren av et IO-kort. Dette er samme type kort som i sender men i mottaker benyttes A/D-modulen. Det er også en datainnsamlingsmodul der det digitaliserte mottatte signalet lagres. Denne datainnsamlingsmodulen er lokalisert på et FPGA-kort fysisk likt det som brukes i sender enheten men hvor man nå bruker datainnsamlingslogikken på kortet.
Dataflyten i mottakerenhet er fra antenne inn på RF-mottakermodul videre gjennom IO-kortet før det samplede signalet lagres på datainnsamlingsmodulen.
I mottakerenheten bruker vi et cPCI system som består av to cPCI bakplan koblet sammen med en PCI bridge. Compact PCI systemet i mottaker er dimensjonert for sanntids signalbehandlingshardware slik at det er 12 slotter i bakplanet. Dette gir følgelig mange ledige slotter når systemet brukes som datainnsamlingssystem. Men det viser fleksibiliteten i systemet ved at hardware og system som er designet for sanntids signalbehandling, med en rekonfigurering av FPGA innholdet på FPGA-kortet, brukes som datainnsamlingssystem. PCen i mottakerenheten er en rackmontert PC som er koblet til cPCI systemet via en PCI bridge.



Figur 2.3 Blokkskjema over mottakerenheten i multistatisk radar.

2.3 RS232 link

Parameteroppsetting i sender og mottaker styres av de respektive PCene i sender og mottaker rackene. Via en trådløs RS232 link kan man fra mottakeren sende parameteroppsettet til PCen i sender. Slik at man fra brukerinterfacet som kjører på konsollet på mottakeren også kan styre parameteroppsettet i sender. Kommunikasjonen mellom mottaker og sender går på en trådløs RS232 link. Denne linken består av en sender og en mottaker som kobles på RS232 inngangen på PCen i henholdsvis mottaker og sender.

2.4 GPS synkronisering

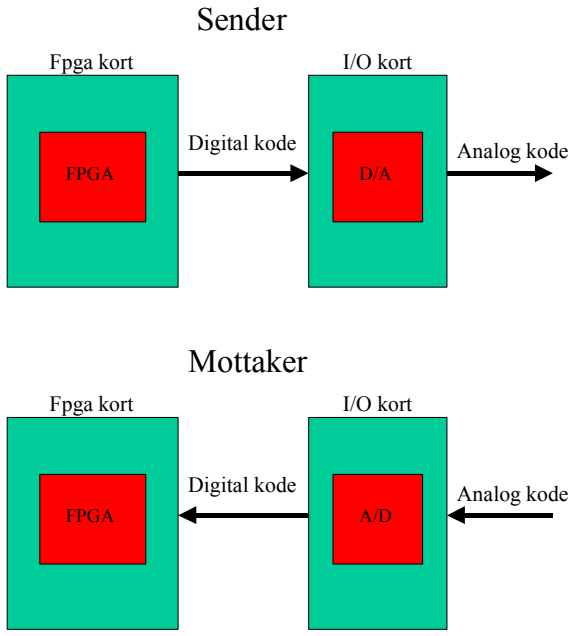
Datainnsamlingen er synkronisert med kodeutsendelse, slik at datainnsamling starter når første bit i koden sendes ut. Denne synkroniseringen skjer ved hjelp av sekundpuls fra GPS-mottaker. I både sender og mottaker er det også en GPS-styrt rubidiumoscilator som sender ut 10MHz. Disse oscilatorene disiplineres mot GPS klokkesignaler. Se for øvrig kapitel 3.8 og (5).

3 FPGA INNHOLD, SENDER OG MOTTAKER

I prosjekt 726 Digital multistatisk radar er det utviklet et HW system for sanntids signalbehandling. Dette systemet er bygd rundt egenutviklete kretskort med FPGA kretser. Dette gjør systemet rekonfigurerbart. Systemet er brukt for kodeutsendelse og datainnsamling i den digitale multistatiske radaren. Dette kapitelet beskriver det designet som implementerer kodeutsendelse og datainnsamling implementert i FPGA på våre egenutviklede kretskort.

3.1 Kort beskrivelse av konseptet

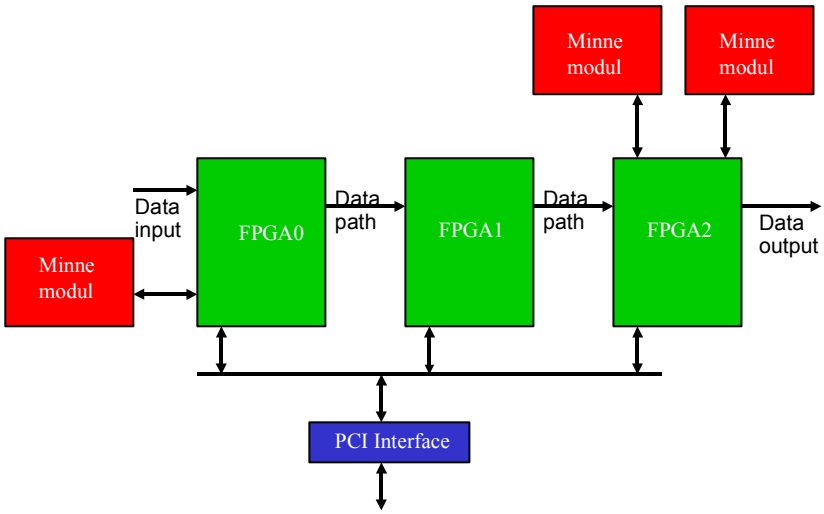
For å verifisere et konsept, digital multistatisk radar, er det bygd en sender og en mottaker. Det tidligere nevnte sanntids signalbehandlingssystemet består, blant annet, av en rekke like kretskort, FPGA-kort. Kjernen på disse kortene består av FPGA kretser. Disse kortene er brukt for å designe digital sender og digital mottaker. Som figuren under viser, består sender av et FPGA kretskort som sender digital kode ut til et I/O kort med en D/A. Her genererer vi det analoge utgangssignalet som sendes videre til RF-sender. Videre med henvisning til Figur 3.1 så har mottaker samme konfigurasjon men med dataflyt motsatt vei. Det kommer analoge data fra RF-mottaker inn på I/O kortet med en A/D. Digitale data sendes så til et FPGA-kort hvor det mottatte signal lagres.



Figur 3.1 FPGA-kortet i konseptuel sammenheng.

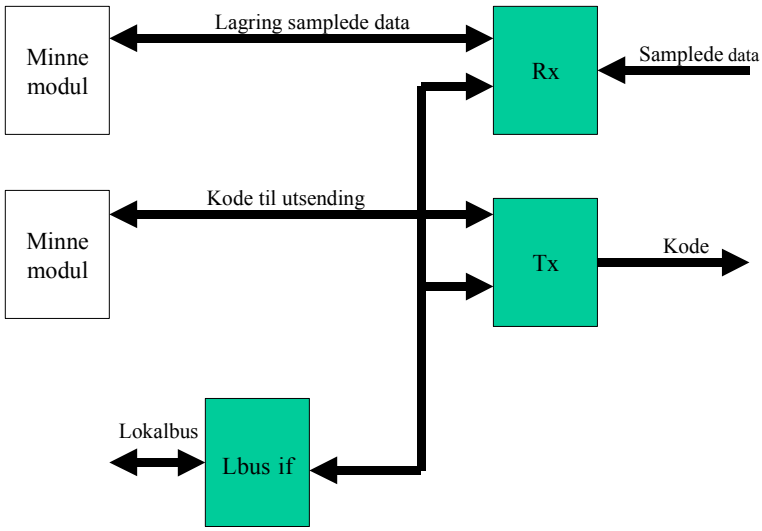
3.2 Kort beskrivelse av FPGA kretskort

Kretskortet er, som vist i Figur 3.2, bygd opp med tre FPGA kretser og tre minnemoduler. I/O-kortet, se Figur 3.1, er koblet til bussen merket Data output i Figur 3.2. Slik at det er FPGA krets merket FPGA2 som er koblet mot I/O kortet og det er i den kretsen sender og mottaker HW er implementert. Det er følgelig i de minnemodulene som er koblet til denne kretsen den digitale koden for utsendelse og mottatte data er lagret i hver sin respektive minnemodul. FPGA kretsene er koblet til en lokalbuss som sørger for at man kan kommunisere med kretsene fra et bruker program. Typiske operasjoner på lokalbussen mot FPGAene er lesing og skriving i minnemoduler og lesing og skriving av registre etc i FPGAene. Aksess til minnemodulene for bruker skjer via lokalbussen. Ellers er minnemodulene integrert i dataveiene i systemet som lagringsplass for data, enten som mellomlagring eller sluttlagring av resultater.



Figur 3.2 Skjematisk oversikt over kortet.

3.3 Overordnet struktur



Figur 3.3 Figuren viser chipens overordnede struktur.

Figuren over viser en overordnet struktur på FPGA chip 2 på et FPGA-kort. Det er samme innhold på FPGA2 på det kortet som sitter i sender og det som sitter i mottaker. Som figuren viser inneholder altså chipen både logikk for datainnsamling, Rx, og for kodeutsending, Tx. Dette vil si at FPGA-kort i sender er identisk med FPGA-kort i mottaker. I tillegg til en

kodeutsendingsblokk og en datainnsamlingsblokk er det et interface mot lokalbussen på chipen.

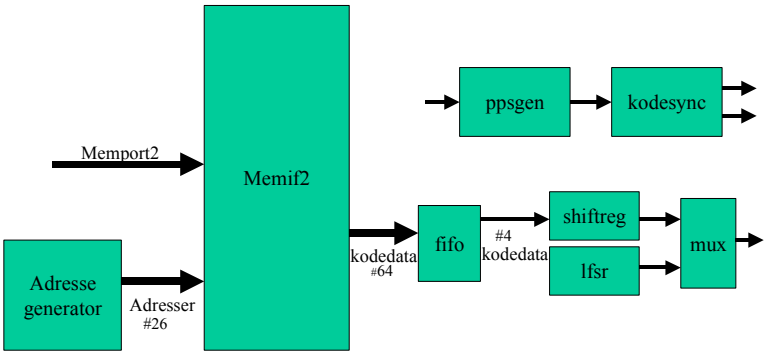
3.4 Lagring av samplede data

Mottatte radardata rutes inn på datainnsamlingsblokk, Rx, se Figur 3.3. Datainnsamlingsblokken ordner og skriver data i rett rekkefølge til minnemodul for lagring. Lagring skjer ved at vi begynner å fylle minne fra adresse 0 og til minne er fylt opp. Det er lagringsplass til 32 Msamples.

3.5 Lagring av kode i minne

Kode for utsendelse er på forhånd lagret i minnemodul for kodeutsendelse, se Figur 3.3. Kodeutsendingsblokka, Tx, leser minnemodulen og sender dataene ut mot I/O kortet. Startadresse og lengde er brukerstyrt.

3.6 Kodeutsendingsblokk

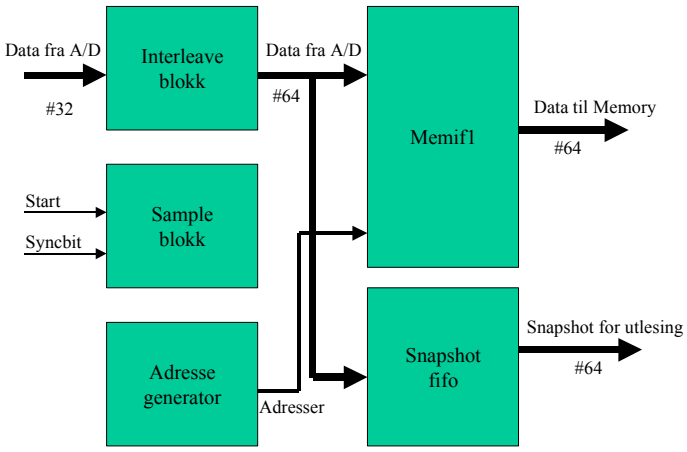


Figur 3.4 Figuren viser en blokknivå beskrivelse av kodeutsending.

Kodeutsendingsblokka består av en adressegenerator som generer adresser til et minneinterface, (Memif2) se kapitel 3.9. Minneinterfacet henter data fra minnemodulen hvor koden er lagret for så å legge kodeordene i en fifo. Adressegeneratoren kan programmeres med startadresse og lengde. Generatoren generer adresser, fra startadressen og med angitt lengde, helt til den restartes. Ut i fra minneinterfacet lagres kodedataene i en fifo. Her lagres de på 64 bits format som er det samme som leses fra minnemodulen. Fifo brukes for synkronisering. Det skrives til fifoen med maksimum minne-båndbredde for så å leses ut med en frekvens bestemt av kodefrekvensen. En kode består av et antall bit som sendes ut sekvensielt. I minne lagres koden i 64 bits ord. I 64 bits ordene brukes 2 bit, bit 0 og bit 32. I tillegg brukes et synkroniseringsbit. Dette bitet ligger som bit 15 i det første 64 bits ordet. Bit 0 og bit 32 legges inn i et shiftregister, parallell inn og seriell ut, for å sekvensiere bitene. Opplasting av shiftregisteret styres av en

State Machine, ikke vist i Figur 3.4.
Datainnsamling må synkroniseres med kodeutsendelse. Dette gjøres med det før omtalte synkroniseringsbitet. Mer om dette i kapitel 3.8. Kodeutsendelse synkroniseres til GPS klokkesignaler. Dette gjøres ved hjelp av en sekundpuls fra GPS-mottakerene. Dette er et signal som sendes fra GPS-mottakerene hvert sekund med basis i GPS-klokken. Hver gang vi mottar en sekundpuls restarter vi kodeutsendelse. For å unngå et avbrudd i kodeutsending i den tidsluken det tar å restarte kodepipen sender vi pseudo random kode. Denne koden genereres i et linear feedback shift register, lfshr. Det velges kode eller pseudo random kode med muxen i slutten av data pathen i Figur 3.4. Årsaken til at man sender pseudo random kode istedenfor ingen utsendelse er ønsket om å oppnå gode LPI egenskaper.

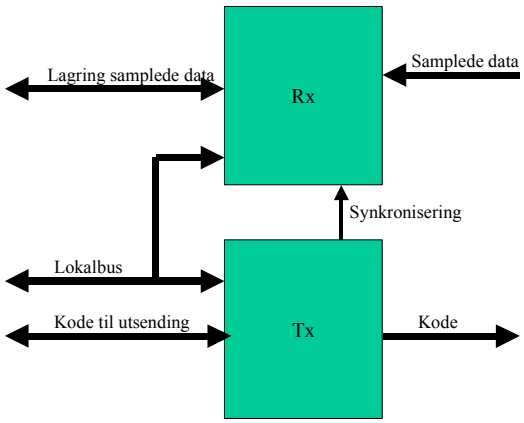
3.7 Datainnsamlingsblokk



Figur 3.5 Figuren viser blokknivå på datainnsamling.

Datainnsamlingsblokken består hovedsakelig av to blokker. Interleave blokka og Memif1, minneinterface(Memory interface). Det er dessuten en blokk kalt sampleblokk som sørger for oppstart av sampling synkronisert med kodeutsendelse i sender. I tillegg er det en snapshot FIFO som viser øyeblikksverdier av samplede data og en adressegenerator.
Data mottatt fra A/D blokken på I/O-kortet, se Figur 3.1, består av et ord som inneholder en real del og en imaginær del, det foregår en IQ demodulasjon i RF mottaker, se Figur 2.3 og (4). I Interleaveblokka ordnes disse i et 32 bits ord. Videre ordnes to etterfølgende 32 bits ord i et 64 bits ord før det sendes til Memif1 for lagring i den dedikerte minnemodulen.
Adressegeneratoren genererer adresser fra adresse 0 og opp til høyeste adresse i adressefeltet, 25 bit.

3.8 Synkronisering



Figur 3.6 Figuren viser overordnet blokkskjema av kretsen med inntegnet synkroniseringslinje mellom kodetutsendingsmodul og datainnsamlingsmodul.

Sender og mottaker synkroniseres på grunnlag av sekundpulsen fra GPS-mottakeren. Som nevnt i kapitel 3.3 er kortene i sender og mottaker identiske. FPGA chip2 inneholder logikk for både kodeutsendelse og datainnsamling. Vi benytter oss av kodeutsendelseslogikken i mottakerenheten for å synkronisere datainnsamling med kodeutsendelse. Når vi starter datainnsamling bruker vi det før omtalte synkbitet, se kapitel 3.6, for å synkronisere innsamling av data med utsending av kode. Dette skjer ved at vi ikke starter sampling av data før vi har mottatt synkbit. Denne prosedyren skjer når datainnsamlingslogikken mottar synkbit etter at det er gitt en kommando for å starte datainnsamling. Synkbit er signalet merket synkronisering i Figur 3.6.

3.9 Minneinterface

Det er to minneinterface i chipen, Memif1 og Memif2. Deres funksjon er å interface den øvrige hardwaren mot minnemodulene. Minnemodulene er PC133 SRAM moduler, minneinterfacet er ingen generelt interface men spesifikt mot denne type moduler. Det er to måter å aksessere minne på igjennom minneinterfacet. En ved hjelp av singelaksesser over lokalbusen, Lbus, som vi kan kalle kontrollaksesser og igjennom ”datapathen” som vi kan kalle datapathaksesser. Kontrollaksessene er bidireksjonale mens datapathaksessene er unidireksjonale. Vi har følgelig to minneinterface Memif1, skriv interfacet, og Memif2, les interfacet. Skriv og les notasjonen er med ståsted i minnemodulen.
Et minne interface har følgende hovedblokker: For kontroll aksesser er det en adresse og en datafifo. For datapathaksesser er det en adressegenerator og en datafifo. I tillegg er det en statemachine som håndterer signalering til minne på cyclenivå. Når det gjelder adressegenerering i datapathen i minneinterfacet ligger selve genereringen, som er en programmerbar teller, hierarkisk på utsiden av selve minneinterface blokka. Den hierarkiske

interne delen av adressegeneringen består av sjekking på kryssing av ”pager” i minnemodulen. I tillegg er datapathfifoen i Memif2, les interfacet, plassert hierarkisk på utsiden av minneinterface blokka.

3.10 Lokalbus interface

På lokalbussen finnes det 4 chip selecter cs0-3. Disse genereres på grunnlag av PCI adresser og settes ut av PCI interfacet på kortet. Disse chip selectene brukes direkte til å definere adresseområder inne på kortet.

- cs0 er registeraksess i Xilinx kretser. I tillegg er det to adressebit, 13 og 12, som definerer chip nummer.
- cs1 er memoryaksesser i minnemodul 1, memif1
- cs2 er memoryaksesser i minnenmodul 2, memif2

Lokalbuss interfacet, Lbusif, interfacer lokalbussen mot register aksesser i en Xilinx krets. Hver adresse i Lbusif aktiviserer en strobe som gir chip enable, ce, til det valgte register. Data på lokalbussen skrives så i registeret med system klokken.

4 BRUKERINTERFACE, SW FOR STYRING AV FPGA-KORT

For å styre FPGA-kortene som er utviklet i prosjekt 726, ”Digital Multistatisk Radar”, er det skrevet en del programmer i C++ for windowsNT. Det viktigste programmet er en monitor, fpgamon, som gir brukeren aksess til de fleste funksjonene på FPGA-kortet.

Lavnivåkommunikasjonen med FPGA-kortene skjer via bibloteksfunksjoner som er kjøpt inn som en del av et ”utviklings-kit” til PLX-Technologys businterfacechip PLX9030, se (6). PLX9030 brukes for å forbinde kortene med PCI-bussen i bakplanet.

4.1 Oppstart

Ved oppstart av fpgamon scannes PCI bussen for å finne alle kort i systemet med PLX9030 interfacechip. Hvert kort tilordnes et logisk nummer og kortets fysiske adresse (plassering i bakplanet) vises. Brukeren velger kort ved å angi logisk nummer. Det er ikke mulig å styre to kort samtidig fra et fpgamon-program, med flere fpgamon-programmer kan kjøres samtidig mot samme eller forskjellige FPGA-kort.

4.2 Hovedmeny

Etter at brukeren har valgt FPGA-kort (egentlig PLX9030 chip), viser programmet en hjelpemeny med mulige valg videre:

```
a = Applications (Code generation, Data acquisition etc)
c = Compute (Writes filter and input, compares output)
d = Diagnose and write to controller
f = fpga reload
```

```
h = Shows this list
i = Reading IDCODE
m = Memory access
p = Program device(s)
q = quit
r = Reset FpgaBoard and eTBC
w = Write register
l = Read register
```

”Applications” gir tilgang til en ny meny med applikasjonsavhengige valg.

”Compute” leser to filer fra harddisk og kopierer disse til henholdsvis minnebank1 og minnebank3 på fpga-kortet. Deretter blir innholdet i minnebank2 sammenlignet med fasitfil på harddisk. Filnavnene er satt fast i programmet. Denne funksjonen brukes for å verifisere regneoperasjoner i fpga-ene.

”Diagnose and write to controller” åpner en meny med funksjoner for å lese og skrive parametere i JTAG-kontrolleren på FPGA-kortet.

”fpga reload” laster fpga-ene fra flash-minne. Dette må gjøres for å få nytt innhold i fpga-ene etter at flash-minnet er reprogramert.

”shows this list” skriver ut hjelp-menyen.

”Reading IDCODE” leser id-koden til komponentene i JTAG-kjeden og viser idkodene på skjermen.

”Memory access” gir tilgang til et sett med funksjoner for skriving og lesing av minne.

”Program device(s)” gir mulighet til å programmere flash-minnet med nytt FPGA-innhold. Dette skjer fra fil som må ligge klar på harddisk. ”Program device(s)” skriver ut liste med filnavn som ligger inne i programmet. Brukeren velger fil etter først å ha laget ei fil med ønsket innhold og lagt denne på harddisken.

”quit” går ut til forrige nivå, dvs lar brukeren velge et nytt FPGA-kort.

”Reset FpgaBoard and eTBC” gir reset signal til FPGA-kretser og annen “brukerstyrt” logikk, samt til JTAG kontrolleren (eTBC= enhanched TestBus Controller)

”Write register” skriver til register. Denne funksjonen er overflødig etter at ”Memory access”-funksjonen ble innført.

”Read register” leser fra register. Denne funksjonen er overflødig etter at ”Memory access”-funksjonen ble innført.

4.2.1 Applikasjonsmeny

Funksjoner for å styre FPGA-kortet når det er programmert som sender/mottaker/datainnsamlingskort. Menyen gir følgende valg:

```
D,d   : Data acquisition
G,g   : GPS
H,h,? : Help. Displays this message
I,i   : Initializing radar system
L,l   : Live parameters
R,r   : Receiver
T,t   : Transmitter
W,w   : Wireless Transmitter control
q     : quit
```

“Data acquisition” styring av datainnsamlingsfunksjonene.

“GPS” lese/skrive parametere fra GPS. Denne funksjonen bruker ikke FPGA-kortet, men aksesserer GPS-modulen via en av PC’ens serieporter. Kommandoen returnerer posisjon fra GPS.

“Help. Displays this message” skriver ut hjelp-menyen.

”Initializing radar system” setter default-parametere i sender og mottaker. Kommandoen må kjøres på mottaker etter at både sender og mottaker er startet. Operasjonene som utføres er:
Kodefil ”sb456” lastes inn i minnet på sender.
Demping i sender settes til 30 dB. Dvs at utgangssignalet er på 0 dBm (1mW).
Senderfrekvens settes til 5.9 GHz.
Senderen slås på.
Demping i mottager settes til 0dB.
Mottagerfrekvens settes til 5.695 GHz.
Liveloop sleeptime settes til 1000 ms.
Liveloop filstørrelse settes til 120000 sampels.
Minneområdet som brukes til synkronisering i mottageren initialiseres.

“Live parameters” gir mulighet for å sette/forandre parametere til “dimura_loop” programmet som brukes for å samle inn data for “live” prosessering i matlab.

“Receiver“ styrer lokal mottager-modul.

“Transmitter“ styrer lokal sender-modul.

“Wireless Transmitter control“ kjøres på mottager for å fjernstyre senderen via radiomodem.

“quit” går tilbake til hovedmenyen.

4.2.1.1 Datainnsamling (data acquisition)

Denne programbiten gir mulighet for å styre datainnsamlingsfunksjonaliteten ved hjelp av noen enkle kommandoer:

```
A,a   : Set attenuator value
D,d   : Dump data to file <filename> <size(samples, decimal)>
G,g   : Go!
H,h,? : Help. Displays this message.
O,o   : Change outputfilename
R,r   : Reset
S,s   : Snapshot
V,v   : View sampled data
q     : quit
```

“Set attenuator value” skriver ut verdien på demperen i mottageren. Brukeren kan så skrive inn ny verdi, 0-79 dB med oppløsning 1 dB. Programmet sjekker at verdien er lovlig, skriver til registeret som styrer demperen, leser tilbake fra registeret og viser den nye verdien.

“Dump data to file” legger innsamlede data på fil. Brukeren blir spurt om antall sampler som skal dumpes og får mulighet til å legge inn kommentar i log-fila som lagres samtidig med data-fila.

“Go!” starter datainnsamling. Innsamlingen starter umiddelbart og fortsetter til minnet er fullt (128 Mbyte = 32 Msamples = 1.79s med samplingshastighet 18.75 Msamples/s)

“Help. Displays this message.” skriver ut hjelp-menyen.

“Change outputfilename” forandrer navn på filene som lagres.
“Reset” resetter FPGA-kortet. FPGA-kortet blir også resatt etter hver datainnsamling.

“Snapshot” leser og viser øyeblikksverdier fra innsamlingskanalene.

“View sampled data“ leser og viser innsamlede data.

“quit” går tilbake til hovedmenyen.

4.2.1.2 Live parametere

Det er skrevet et C++-program (”dimura_loop”) som kjører i løkke og gjør datainnsamling og skriver til fil for at vi skal kunne gjøre ”live” prosessering. Dette programmet leser to parametere fra datainnsamlingskortet: ”filesize” og ”sleeptime”. ”Filesize” sier hvor mange ord som skal samles inn og legges på fil hver gang løkka kjøres. ”Sleeptime” sier hvor mange millisekunder programmet skal vente fra fil er skrevet til datainnsamling startes på nytt. Disse parameterene kan settes fra ”LiveLoop”-menyen i fpgamon:

```
F,f   : Set filesize (number of samples)
H,h,? : Help. Displays this message.
S,s   : Set sleep value (milliseconds)
q     : quit
```

“Set filesize” viser verdien på “Filesize” og gir brukeren mulighet til å sette ny verdi.

“Help. Displays this message” skriver ut hjelp-menyen.

“Set sleep value” viser verdien på “Sleeptime” og gir brukeren mulighet til å sette ny verdi.

“quit” går tilbake til hovedmenyen.

4.2.1.3 Receiver (mottaker)

Styring av mottager-parametere:

```
A,a   : Set attenuator value
F,f   : Set 1.LO frequency
H,h,? : Help. Displays this message.
q     : quit
```

“Set attenuator value” skriver ut verdien på demperen i mottageren. Brukeren kan så skrive inn ny verdi, 0-79 dB med oppløsning 1 dB. Programmet sjekker at verdien er lovlig, skriver til registeret som styrer demperen, leser tilbake fra registeret og viser den nye verdien.

“Set 1.LO frequency“ skriver ut frekvensen på 1.LO i mottageren (”mottagerfrekvensen”). Brukeren kan så skrive inn ny verdi, 5.0 – 6.0 GHz. Programmet sjekker at verdien er lovlig, runder eventuelt av til nærmeste 125 kHz, skriver til registeret som styrer syntesizeren, leser tilbake fra registeret og viser den nye verdien.

“Help. Displays this message” skriver ut hjelp-menyen.

“quit” går tilbake til hovedmenyen.

4.2.1.4 Transmitter (sender)

Styring av senderparametere lokalt, dvs når fpgamon kjøres på senderens PC.

```
A,a   : Set attenuator value
B,b   : Set begin address for code
C,c   : Set codegenerator frequency
F,f   : Set 1.LO frequency
H,h,? : Help. Displays this message.
K,k   : Copy code from codefile to memory
L,l   : Set length of code
O,o   : On/off toggel.
q     : quit
```

"Set attenuator value" skriver ut verdien på demperen i senderen. Brukeren kan så skrive inn ny verdi, 0-70 dB med oppløsning 10 dB. Programmet sjekker at verdien er lovlig, skriver til registeret som styrer demperen, leser tilbake fra registeret og viser den nye verdien.

"Set begin address for code" gir brukeren mulighet til å forandre startadressen for lesing av kode fra minnet. Kan brukes for å bytte kode med flere koder liggende i minnet samtidig.

"Set 1.LO frequency" skriver ut frekvensen på 1.LO i senderen ("senderfrekvensen"). Brukeren kan så skrive inn ny verdi, 5.0 – 6.0 GHz. Programmet sjekker at verdien er lovlig, runder eventuelt av til nærmeste 125 kHz, skriver til registeret som styrer syntesizeren, leser tilbake fra registeret og viser den nye verdien.

"Help. Displays this message" skriver ut hjelp-menyen.

"Copy code from codefile to memory" kopierer kodefil fra harddisk til minnet på FPGA-kortet. Brukeren skriver inn filnavnet. Koden legges inn fra adresse 0 i kodeminnet. Startadresse for lesing av kode settes til 0 og kodelengde-parameteren settes.

"Set length of code" gir brukeren mulighet til å forandre kodelengde-parameteren for lesing av kode fra minnet. Ved normal bruk settes denne automatisk ved kopiering av kode fra fil til minne.

"On/off toggel" slår senderen av og på.

"quit" går tilbake til hovedmenyen.

4.2.1.5 Wireless Transmitter (trådløs sender)

Fjernstyring av senderparametere, dvs når fpgamon kjøres på mottagerens PC. Den trådløse overføringen skjer via trådløs RS232-kommunikasjon (radiomodem).

```
A,a   : Set attenuator value
F,f   : Set 1.LO frequency
H,h,? : Help. Displays this message
K,k   : Copy code from codefile to memory
O,o   : On/off toggel
S,s   : Display current transmitter settings
q     : quit
```

“Set attenuator value” skriver ut verdien på demperen i senderen. Brukeren kan så skrive inn ny verdi, 0-70 dB med oppløsning 10 dB. Programmet sjekker at verdien er lovlig, skriver til registeret som styrer demperen, leser tilbake fra registeret og viser den nye verdien. Samtidig vises også verdien for senderfrekvens og om senderen står av eller på.

“Set 1.LO frequency“ skriver ut frekvensen på 1.LO i senderen (”senderfrekvensen”). Brukeren kan så skrive inn ny verdi, 5.0 – 6.0 GHz. Programmet sjekker at verdien er lovlig, runder eventuelt av til nærmeste 125 kHz, skriver til registeret som styrer syntesizeren, leser tilbake fra registeret og viser den nye verdien. Samtidig vises også verdien for demperen i senderen og om senderen står av eller på.

“Help. Displays this message” skriver ut hjelp-menyen.

“Copy code from codefile to memory” kopierer kodefil fra harddisk til minnet på FPGA-kortet. Brukeren skriver inn filnavnet. Koden legges inn fra adresse 0 i kodeminnet. Startadresse for lesing av kode settes til 0 og kodelengde-parameteren settes. Filnavn og kodelengde skrives ut.

“On/off toggel” slår senderen av og på, ny tilstand vises sammen med verdiene for demping og senderfrekvens.

“Display current transmitter settings” viser status for senderen.

“quit” går tilbake til hovedmenyen.

4.3 Kontroll av JTAG-kontrolleren

Konfigurasjonsregisterene i JTAG-kontrolleren leses og verdiene skrives ut sammen med en liste over lovlige kommandoer:

```
0 ConfigurationA : xx
1 ConfigurationB : xx
2 status         : xx
3 Command        : xx
4 TDObuffer      : xx
5 TDIbuffer      : xx
6 counterval     : xxxxxxxx
```

```
7 Discrete ctrl  : xx
8 IDReg          : xx

0-7 = write register
r   = reset eTBC
q   = quit
any other key = update
```

"0-7" ber brukeren om nytt innhold til gjeldende register hvis registeret er skrivbart. Hvis registeret ikke er skrivbart får brukeren beskjed om det.

"r" resetter kontrolleren.

"quit" går tilbake til hovedmenyen.

Alle andre tastetrykk vil lese og vise registerverdiene på nytt.

Denne funksjonen brukes bare for debugging. Ved normal bruk av systemet blir nødvendige kontrollerparameterene satt automatisk av programmet som styrer FPGA-kortet.

4.4 Skriving og lesing av minne

Adressene angis som logiske 32bits byte-adresser. Mapping fra logisk til fysisk adresse skjer i flere trinn i C ++ programmet, i det innkjøpte bibloteket, i PC'ens bios og på FPGA-kortet.

Den logiske adressen bygges opp av flere felt:
Bit 31 : 28 : Adressespace på FPGA-kortet (0-3)
Bit 27 : 20 : "Remappes" i PLX9030
Bit 19 : 16 : "Remappes" i FPGA
Bit 15 : 2 : Fysiske adressbit 15:2
Bit 1 : 0 : Brukes ikke. Alle aksesser skal være 32 bit. (a1=a0=0)

Hver adressespace har fått et område i den fysiske adressemapen i PC'en.
"Remappes" betyr at verdien lagres i et register og settes ut mot minnet sammen med resten av adressen.
Tilgengelige kommandoer ved skriving og lesing av minnet:

```
A,a   : Address in address, <start address>, <size(bytes)>
B,b   : Begin/start address for code, <start address
C,c   : Copy data
<Source startaddress>, <Destination startaddress>, <Size(Bytes, hex)>
D,d   : Data compare <Memoryspace (0,1,2,3)>
E,e   : End address for code, <end address>
F,f   : Fill memory, <start address>, <size(bytes)>, <datavalue>
```

```
G,g   : Gather, Starts sampling
H,h,? : Help. Displays this message.
L,l   : Load data from file to memory,  <Memoryspace (0,1,2,3)>
M,m   : Memory read and write
S,s   : Sequencial datafill.
<start address>, <size(bytes)>, <start value>, <stepsize>
V,v   : Verify.
      <start address0>, <start address1>, <size(bytes)> <no of errors>
q     : quit
```

"Address in address" skriver logisk adresse for lokasjonen inn i minnet. Parametre angir startadresse og størrelsen på dataområdet.

"Begin/start address for code" er addresse for start av kodeinnhold. Kan brukes for å skifte kode raskt. Ved å benytte dette registeret kan flere koder ligge i minnet samtidig.

"Copy data" kopierer data fra et område til et annet. Parametre: kildestartadresse, destinasjonstartaddresse og antall byte.

"Data compare" sammenligner innholdet i et dataområde med data i ei fil. Minneområde(addressespace) angis som en parameter. Dataenes plassering i minneområdet angis av addressefeltet i datafila. Datafilas navn og plassering ligger fast i C ++ programmet.

"End address for code" er kodelengden.

"Fill memory" fyller et minneområde med en dataverdi. Startaddresse, størrelse på dataområdet og dataverdi angis.

"Gather, Starts sampling" starter datainnsamling ved å skrive til en bestemt adresse. Dette gjøres vanligvis fra datainnsamlingsmenyen.

"Help. Displays this message" skriver ut hjelp-menyen.

"Load data from file to memory" kopierer data fra fil til minne. Minneområde(addressespace) angis som en parameter. Dataenes plassering i minneområdet angis av addressefeltet i datafila. Datafilas navn og plassering ligger fast i C ++ programmet.

"Memory read and write" leser og skriver enkeltord i minnet

"Sequencial datafill" legger inn inkrementelle dataverdier i et minneområde. Startadresse, størrelse på dataområdet, startverdi og inkrement angis som parametre.

"Verify" sammenligner innholdet i to dataområder. Parametere: Startadresser for de to

områdene, størrelsen på dataområdene og hvor mange feil som tillates før sammenligningen stoppes.

“quit” går tilbake til hovedmenyen.

Litteratur

(1) Johnsen Terje, Olsen Karl Erik, Johnsrud Steinar, Gundersen Rune, Bjordal Halvor, Tansem Ivar, Sørnes Per (2002): Multistatisk CW radar – konsept, FFI/RAPPORT-2002/01767

(2) Sørnes, Per (2002): FPGAkort2, FFI/RAPPORT-2002/02364

(3) Sørnes, Per (2002): IO kort, FFI/RAPPORT-2002/02365

(4) Gundersen Rune, Johnsen Terje, Johnsrud Steinar, Sørnes Per (2002): RF-maskinvare for digital radar,FFI/RAPPORT-2002/01751, under utarbeidelse

(5) Johnsen Terje (2002): Frekvens- og tidssynkronisering ved bruk av GPS disiplinerte referanser, FFI/RAPPORT-2002/00701, Begrenset

(6) PLX Technology (2000): PLX PCI Host SDK Software Development Kit

FORDELINGSLISTE

FFIE **Dato:** 22 Mai 2002

RAPPORTTYPE (KRYSS AV)	RAPPORT NR.	REFERANSE	RAPPORTENS DATO
X RAPP ☐ NOTAT ☐ RR	2002/01931	FFIE/726/170	22 Mai 2002
RAPPORTENS BESKYTTELSESGRAD		ANTALL EKS UTSTEDT	ANTALL SIDER
UGRADERT		45	27
RAPPORTENS TITTEL		FORFATTER(E)	
DIGITAL MULTISTATISK RADAR, SENDER OG MOTTAKER		JOHNSRUD Steinar, TANSEM Ivar	
FORDELING GODKJENT AV FORSKNINGSSJEF		FORDELING GODKJENT AV AVDELINGSSJEF:	
John-Mikal Størdal		Johnny Bardal	

EKSTERN FORDELING

ANTALL	EKS NR	TIL
1		Tore Belsnes, FO/E
1		Major Sverre Vestad, LVI
1		FO/SST
1		Bjørn Dyrøy, FLO/Sjø/Teknisk avdeling

INTERN FORDELING

ANTALL	EKS NR	TIL
14		FFI-Bibl
1		Adm direktør/stabssjef
1		FFIE
1		FFISYS
1		FFIBM
1		FFIN
1		Ivar Tansem, FFIE
1		Steinar Johnsrud, FFIE
1		Rune Gundersen, FFIE
1		Halvor Bjordal, FFIE
1		Per Sørnes, FFIE
1		Terje Johnsen, FFIE
1		Karl Erik Olsen, FFIE
1		John-Mikal Størdal, FFIE
1		Trond Hellum, FFIE
1		Øyvind Thingsrud, FFIE
1		Tor Holmboe, FFIE
1		Arne Petter Bartholdsen, FFIE
1		Leif Hanssen, FFIE
1		Svein Erik Hamran, FFIE
1		Tore Smedstad, FFIE
1		Hans Øhra, FFIE
1		Kirsten Kvernsveen , FFIE
1		Harald Mathisen, FFISYS
1		Stein Malerud, FFISYS
1		Arne Skogstad, FFIN
2		Arkiv, FFIE
		FFI-veven

FFI-K1 Retningslinjer for fordeling og forsendelse er gitt i Oraklet, Bind I, Bestemmelser om publikasjoner for Forsvarets forskningsinstitutt, pkt 2 og 5. Benytt ny side om nødvendig.