Forord

Denne rapporten beskriver en mulig anvendelse av analog og digital VLSI til å realisere en ASIC-krets brukt i et kommersielt system. Problemstillingen sirkler mye om pris, og streben mot enkel konstruksjon. Mest interessant er likevel det faktum at alle de komponenter som brukes i tilsvarende systemer, her er integrert i én enkelt krets. Enkelte kretsløsninger og VLSI-strukturer er derfor svært spesielle, og ikke tidligere brukt i tilsvarende systemer.

Oppgaven er gitt i samarbeid mellom NorskElektroOptikk A/S, **NEO**, ved Jon Kristian Hagene, og Institutt for Informatikk, ved Tor Sverre Lande. Den inkluderer analyse av problemstillingen, spesifikasjoner for et praktisk system, konstruksjonsløsning for systemet og endelig implementasjon i VLSI. Alt praktisk arbeid er utført på utstyr ved Institutt for Informatikk. Testing og målinger er utført ved IfI's VLSI-LAB.

Rapporten forutsetter at leseren er kjent med begreper fra VLSI-konstruksjon, og har innsikt i digital og analog elektronikk. Begreper hentet fra spesielle områder i disse fagfeltene og brukt i rapporten, blir forklart etter hvert som de introduseres. Jeg har prøvd å unngå referanser til andre arbeider og henvisninger til litteratur. Alle deler er forklart såvidt forståelig at leseren skal kunne sette seg inn løsningen uten å måtte lese ekstra litteratur. Et vedlegg tar for seg selve realiseringen av systemet i VLSI. Realiseringen er teknologi-avhengig; parametere og teknikker endres etter kort tid, derfor er ikke denne delen av oppgaven berørt i selve rapporten.

Jeg ønsker å takke Jon Kr.Hagene og Tor S.Lande for all den hjelp de har gitt underveis i arbeidet med oppgaven. Dessuten Yngvar Berg ved IfI for kommentarer og tilbakemelding på rapportskriving og punkter i oppgaveløsningen, og Terje Knudsen[11][12] for hjelp med utlegget og kretstesting.

Morten Larsen
IfI, november 1992

Innhold

Figurer

Tabeller

Kapittel 1

Innledning

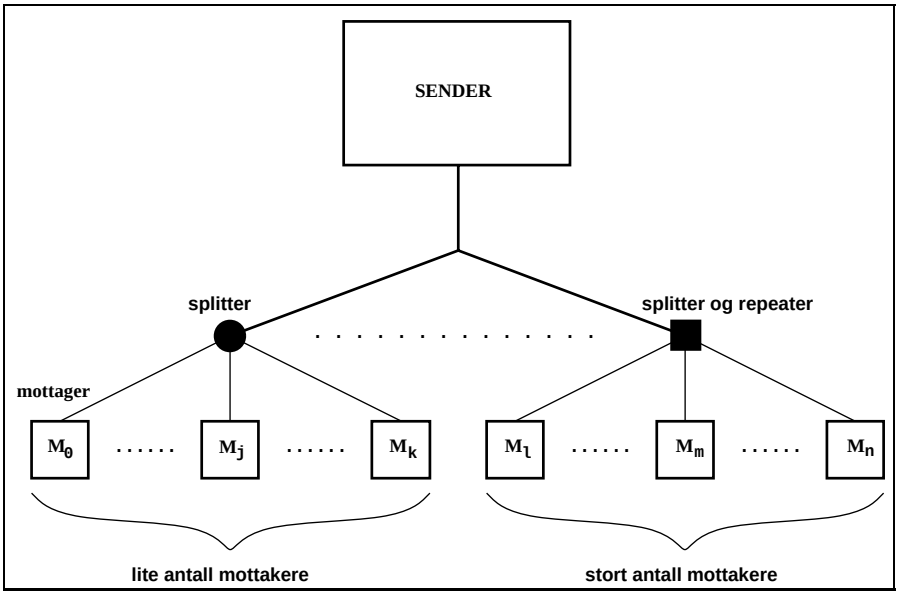
1.1 Bakgrunn og intensjoner

Bakgrunnen for oppgaven er at det per i dag er mulig å splitte lyset fra én optisk fiber ut i flere fibre *direkte*, uten å benytte elektrooptiske komponenter som forsterkere eller "repeatere". Med en høyeffekt-laser som lyskilde, er det mulig å sende data til mange mottakere *samtidig*. Dette vil bli en *billig* løsning, fordi dyre koblinger og elektroniske komponenter elimineres. Men det er bare mulig å *sende* data til mange passive mottakere med dette systemet. Det er adskillig vanskeligere å multiplekse signalet fra flere fibre over i én enkelt fiber.

Det er ønskelig å bruke denne teknologien til å bygge et fibernett bestående av én sender og mange mottakernoder. Senderutstyret kan tenkes å være relativt avansert, mens mottakerustyret til hver enkelt node må være billig og enkelt. Mottakernodene skal være adressérbare fra sendersiden, og de skal utføre lokale funksjoner ut i fra hvilke kommandoer senderen gir. Disse funksjonene kan være å styre sikkerhetsbrytere i et elektrisk støyfullt miljø, eller slå måleinstrumenter av og på. Andre bruksområder er typisk énveis kommunikasjon, som for eksempel nedlasting av et program til en mikrokontroller.

Mottagerutstyrets kraftforsyning skal være batteri(er). Fordi utstyret skal kunne stå lenge uten vedlikehold, må utstyret kunne slås av og på etter behov for å spare batteriet. Dette gjøres i praksis ved å stanse mottagernodens klokke.

En annen detalj ved systemet,er at alle komponentene i mottakerutstyret er integrert i VLSI på én og samme brikke. En hybrid analog/digital krets med fotodetektor, demodulator, oscillator som representerer lokal klokke, registre, tellere og kontroll-logikk er lagt ut på én silisiumbit. Videre festes den optiske fiberen *direkte* på overflaten til brikken, uten noen form for spesialtilkobling. Dette vil gjøre systemet billig, og enkelt å montere.



Figur 1.1: Systemarkitektur

1.2 Systemoversikt

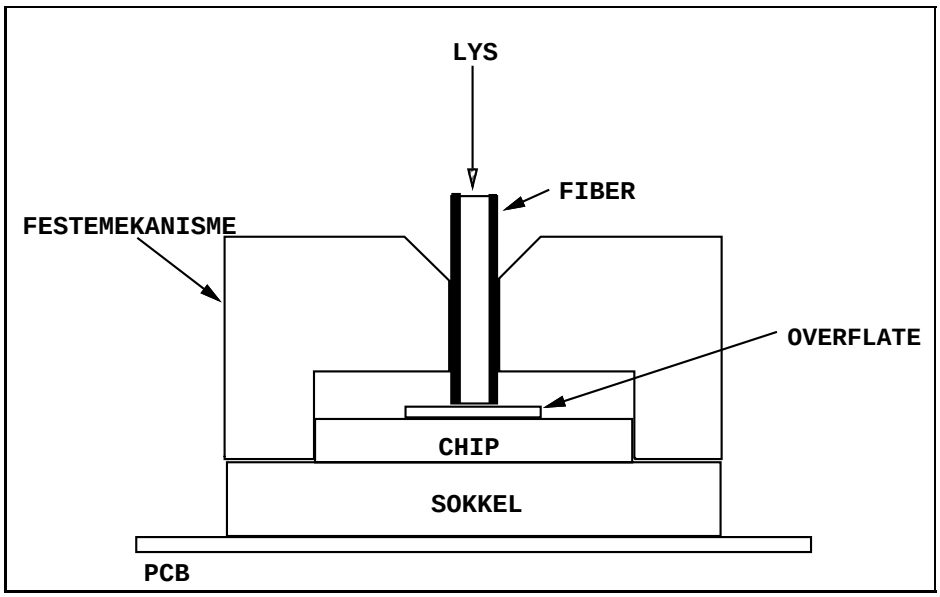
1.2.1 Nett-topologi

Topologien, eller oppbygningen til nettet er type "én-til-alle". Det består av én enkelt sender, og et større antall mottagere som er forbundet med fiberoptikk. Dette er vist i figur 1.1. Maksimalt antall mottagere er begrenset av adresse-lengden. Adressen til mottagerne er fast, og kan ikke endres av senderen. Effekten fra senderen fordeles tilnærmet likt på hver mottager. Senderen bør derfor være kraftig, fortrinnsvis en laser som opererer i området fra infrarødt til synlig, rødt lys. Men er antall mottagere lite, kan en høyeffekt-LED være tilstrekkelig.

Denne nett-arkitekturen egner seg for eksempel i et system med mange instrumenter som skal slås av og på til forskjellige tider.

1.2.2 Fiber direkte på chip

For å gjøre systemet billig og enkelt, skal fiberen kunne festes *direkte* på overglasset til den integrerte mottageren. Dette kan gjøres med for eksempel epoxy-lim, eller en enkel adapter-del som vist i figur 1.2. Overgangen vil gi et visst tap, men dette skal ikke påvirke påliteligheten.



Figur 1.2: Mekanisk oppbygning

1.2.3 "On-chip" fotomottager

Foto-detektorer i optisk kommunikasjon er vanligvis dyre spesialkomponenter som kan trenge egen, isolert spennings-forsyning. Argumentet som brukes til forsvar av disse omkostningene, er den store båndbredden disse systemene har, det vil si høy datarate. Relativt mye *lavere* datarate indikerer at det kan være mulig å lage en integrert foto-detektor i standard CMOS-prossess, på samme brikke som inneholder resten av mottageren. Hvilket betyr et billig, enkelt og ikke minst kompakt system.

1.2.4 "On-chip" klokke

En av de grunnleggende forutsetninger for konstruksjonen av systemet, er at synkron dataoverføring skal brukes. Mottageren skal derfor ha en synkroniseringskrets, en *demodulator*, til dette formålet. Demodulatoren trenger en lokal referanse, en mottager-klokke, for å synkronisere datamottaket. Denne klokka er en *spenningsstyrt* oscillator, en VCO (Voltage Controlled Oscillator), hvis frekvens varierer med en kontrollspenning, V_c. Mottagerenheten trenger dessuten en klokke for å klokke registre, vipper og tellere i digitaldelen.

1.2.5 Spenningsforsyning

Et annet moment ved å integrere hele systemet på én enkelt chip, er lavt effektforbruk. Lavt effektforbruk betyr at batterier kan benyttes som spenningsforsyning, hvilket vil si fullstendig selvbergede enheter. Dersom effektforbruket er tilstrekkelig lite, kan en mottager være i full operasjon hele den beregnede levetiden uten tilsyn eller service. For å

oppnå dette, bør kretsen operere med lavere V_{dd} enn +5V. Et Lithium-batteri kan gi en forsyningsspenning på 3.2V, som er anvendelig for vanlig *statisk* digital logikk forutsatt lav klokkefrekvens. De analoge kretsene kan få større problemer med et lavt spenningsnivå, men dette kan jeg ikke si noe sikkert om på forhånd.

1.2.6 Adresséring

Setting av mottager-adresse kan gjøres via pinner på kretsen med brytere, eksempelvis DIP-brytere, for enkelhets skyld. Dette bør først og fremst gjelde prototyp-versjonen. Endelig versjon bør ha statisk RAM, eller et register der adresse kan settes via programmeringspinner.

1.2.7 System og omgivelser

På grunn av lav datarate er en billig, multi-modus plastfiber tilstrekkelig som transmisjonsmedium. Fiberlengden vil ikke være begrenset av forskjeller i ganglengde for lyset i fiberen, men heller av dempningen eller effekttapet i fiberen. Batteridrift betyr at selve spenningsforsyningen er robust overfor støy, men kretskortet bør likevel plasseres i en skjermet boks. Selve kretskortet vil bare inneholde mottaker-brikken, batteri(er), avkoblingskondensatorer, adresse-brytere og eventuelt et par potensiometere for å justere forspenning til enkelte kretselementer slik at marginer og grenseverdier kan undersøkes. De to siste detaljene vil bare gjelde for prototypen.

Festemekanismen for fiberen bør utformes slik at den lett kan festes på mottager-brikken, og såpass nøyaktig at etterjustering er unødvendig. Som et eksempel kan vi tenke oss en tykk plastkabel med 1mm tversnitt brukt som lysleder. Lysstrålen som faller på fotodetektor-strukturen er da minst 1mm i diameter. Dersom detektor-strukturen er sirkulær og 0.3mm i diameter, har vi en plasserings-margin på vel 0.35mm i hver retning hvis vi forutsetter at hele strukturen skal belyses.

1.3 Systemfunksjon

1.3.1 Kommunikasjon

Kommunikasjon mellom sender og mottagere foregår ved hjelp av datarammer. Datarammene er avgrensede sekvenser med data. De er oppdelt i forskjellige felter, og oppbygningen er definert som fast. Rammene er derfor like med hensyn på lengde, der lengden er lik et helt antall ord. Rammene utgjør både en innpakning av data for oversending, og en styring av kommunikasjonen. Alle rammer leses av alle mottagere, men bare den enheten som rammen er adressert til, foretar seg noe aktivt.

1.3.2 Kommandoer

Den digitale logikken i mottageren skal realisere kommandoer som styrer klokkefunksjonen og timeren. Klokka skal bare kunne slås *av* med en kommando, mens timeren skal kunne slås både av og på via kommandoer.

1.3.3 Feilsjekking

Feilretting eller feilkorreksjon er komplisert, og krever mye ekstra logikk som gir stort arealforbruk. Dessuten er det nødvendig å legge til ekstra bit-informasjon, som vil gi "overhead" i data-strømmen. Derfor er det bedre å lage et pålitelig system, der feil ikke får katastrofale følger. Systemet vil derfor bare utføre feilsjekking. Dette kan være en CRC (CyclicRedundancyCheck), eller en paritets-sjekk.

En seriell CRC vil si at et ord, en såkalt *sjekksum*, genereres ut av mottatte, serielle data av gitt lengde. Generert sjekksum sammenliknes så med et ord som blir sendt etter vanlige data-ord. Dette ordet blir generert av originale data i senderen ved hjelp av den samme kretsen som i mottakeren. Derfor skal de to ordene være identiske, og et avvik vil indikere transmisjonsfeil. En enkel paritets-sjekk innebærer at et paritets-bit blir satt eller resatt ettersom mottatte data av en gitt lengde inneholder et like eller et odde antall bit av verdi "1". Dette bitet sammenliknes så med et tilsvarende oversendt bit.

Mens sjekksum-generering bruker et helt ord til lagring av verdien og mye logikk for selve genereringen, er paritetsbit-generering enklere og trenger kún en vippe for å holde verdien. Men til gjengjeld dekker en CRC-sjekk flere feil enn en paritets-sjekk, og den detekterer flere etterfølgende bitfeil som for eksempel ved "burst errors". Prisen å betale er mer logikk enn for paritets-sjekk.

Fordi jeg antok at systemet ville bli for komplekst med feilkorrigering, utelot jeg dette. Det ville blitt vanskelig å lage, og ville desuten tatt opp uforholdsmessig mye plass på brikken. I stedet ønsket jeg å gjøre feilsjekkingen pålitelig. En én-bits paritets-sjekk gir kun 50% feildekning, og dette var argument godt nok for å bruke CRC istedet.

Kapittel 2

Dataoverføring

I dette kapitlet tar jeg for meg de viktigste aspektene ved selve datatransporten. Det vil si selve kodingsformatet, altså hvordan enkeltbit representeres på linjen, og rammeformatet. Rammeformatet bestemmer følgende:

- Hvordan mottageren skal kunne synkronisere seg til senderen
- Hvordan mottageren skal vite at det virkelig kommer en ny sending med data
- Hvordan mottager skal verifisere om dataene er korrekt mottatt
- Hvordan mottageren skal adresséres
- Hva mottageren skal foreta seg på grunnlag av oversendte data

Det første punktet introduserer begrepene *koherente* og *ikke-koherente* mottager-systemer. Disse begrepene forklares nærmere i kapittel 3, men foreløpig kan koherente systemer oversettes med mottager *med* lokal klokke, og ikke-koherente med mottager *uten* lokal klokke. Øvrige begreper introdusert i dette kapitlet er kort og punktvis forklart i neste underkapittel.

2.1 Begreper

- L_r = lengden av en dataramme
- L_f = lengden av et felt
- T_B = bitperioden
- f_B = baudraten eller bitfrekvensen = $\frac{1}{T_B}$
- f_t = transmisjonsfrekvensen eller klokkerate til T_x
- T_x = senderklokke
- R_x = mottagerklokke

2.2 Modulasjon/koding

Forutsetningen for systemet, er at digital overføring benyttes. Derfor vil jeg bare gjennomgå *digitale* modulerings-teknikker. Dette kalles også koding, fordi bits representeres under sending som en kombinasjon av digitale nivåer i tid. Det er mange forskjellige typer digital modulasjon eller koding; noen innebærer koherente systemer, mens andre gir tilstrekkelig klokkeinformasjon til at data kan "klokke seg selv". Dersom en ikke-koherent teknikk brukes for demodulasjon, kan systemet kalles en dekoder heller enn en demodulator.

De hensyn som spiller inn ved valg av kodingsform, er som følger:

- Oppnåelig hastighet (datarate)
- Båndbredde kontra datarate
- Egnethet for ikke-koherent eller koherent mottagning
- Feilsannsynlighet
- Kompleksitet

Forskjellige kodingsformer har forskjellige begrensninger med hensyn på datarate. Dataraten benevnes [bit/s], eller "baud".

De fleste teknikker for digital koding krever 2 eller flere nivåskift for å representere et bit. Dette betyr at kodet signal vil ha en frekvens som er det dobbelte eller mer enn hva en ukodet bitstrøm maksimalt vil ha. Maksimal signalfrekvens for en ukodet bitstrøm fåes dersom fire etterfølgende bit er forskjellige fra det forrige, altså bitmønsteret "0101". Dette blir et firkantpulstog med signalfrekvens lik halvparten av dataraten, fordi to bitperioder utgjør signalperioden; $f_{t,max} = \frac{1}{2_B}$. Dobbelt så mange nivåskift i kodet signal som i datasignalet betyr at båndbredden til sender, transmisjonslinje og mottager må dobles. Det vil si en dobbelt så rask laserdiode for sending og dobbelt så rask fotodiode for mottagning, vanskeligere og dyrere konstruksjon av sender og mottager. Kanskje blir også frekvensinnholdet i signalet flyttet opp til et område med mer støy.

Som før nevnt er enkelte kodingsformer ubrukelige for ikke-koherent mottagning. Dette gjelder alle kodingsformer som ikke har minst én transisjon i hver bitperiode. Også for en koherent mottager er det ønskelig med mange transisjoner. Det sikrer god synkronisering, og muligheten for feil er ikke avhengig av datainnholdet. Også koherente systemer er avhengige av synkroniseringspunkter, eller transisjoner, med et visst mellomrom. Det er derfor ikke mulig å sende et enormt antall nuller representert med ett nivå, altså ukodet, og deretter forvente at en koherent mottaker skal kunne klokke etterfølgende data riktig inn.

Det er lettere å oppdage feil i overføringen dersom man vet at hvert bit inneholder for eksempel en transisjon i midten av bitet. En vippe kan da brukes til å oppdage en manglende transisjon. Dersom overføringskanalen er beheftet med støy, foretrekkes kodingsformer som gir minst én transisjon i hvert bit.

Hvor komplisert sender og mottaker behøver å være, avhenger slett ikke bare av kodingsformen. Et ikke-koherent system er vanligvis konstruksjonsmessig enklere enn et koherent, og enkelte blandede analoge/digitale teknikker har en mengde finesser som gjør systemet stort i utstrekning, og komplisert i virkemåte. Likevel er det en sammenheng mellom hvor effektiv en kodingsform er med hensyn på bitrate, eller båndbredde, og hvor kompleks sender og mottager må være.

Jeg vil begrense diskusjonen til rent digitale moduleringsteknikker, der bare to nivåer i signalet er tillatt, og det ikke er noen form for modulasjon av pulsbredden slik som i PWM (PulseWidthModulation) og teknikker avledet av denne. De mest vanlige formene for digital modulasjon eller linjekoding er som følger:

- NRZ - "Non Return to Zero"
- "Mark-Space"-koding
- RZ - "Return to Zero"
- Manchester-koding
- Delay-koding, også kalt Miller-koding

Ren NRZ-koding (se figur 2.1) er egentlig ingen koding av dataene overhodet. Et "1"-symbol er representert ved et høyt nivå av varighet lik en periode av overføringsfrekvensen eller klokkeraten. Et "0"-symbol er representert med et lavt nivå av samme lengde. Det betyr at mottagningen må skje koherent. Synkroniseringen er sterkt avhengig av dataene; finnes ingen overganger fra "0" til "1" i bitstrømmen, finnes heller ingen synkroniseringspunkter. Derfor er det nødvendig med synkroniserings-sekvenser med jevne mellomrom som består av bitmønsteret {...0101010101...} for å holde lokal klokke låst på referansen.

Utenom synkroniserings-sekvensene kan ikke NRZ-dataene brukes som referanse for en koherent mottager, fordi frekvensinnholdet er spredt utover hele spekteret fra 0Hz (DC) til $f_{t,max} = \frac{1}{2T_B} = \frac{1}{2} \cdot bitraten$. Vanligvis gjøres derfor NRZ-data om til kvasi-RZ-data ved å generere en puls ved hver transisjon, både for positiv og negativ flanke. Dermed *dobles* antall synkroniseringspunkter, og i tillegg har man da en frekvenskomponent lik $f_B = \frac{1}{T_B}$=bitraten, som lokal klokke kan bruke som referanse.

"Space-Mark", er egentlig bare en variant av enten NRZ- eller bifase-koding. En transisjon fra lav til høy i dataene, blir kodet som et såkalt "Space", mens den omvendte transisjonen kodes som et "Mark". Denne teknikken gir færre transisjoner enn grunnformen den bygger på, og sparer båndbredde.

RZ-koding er nesten like enkel som NRZ: Bitverdien "1" representeres med en puls av varighet $\frac{T_B}{2}$, der T_B er bitperioden, og bitverdien "0" representeres ved fravær av en puls (se figur 2.1). Dette betyr igjen at en ikke-koherent teknikk for mottagning ikke er brukbar. Kodingsteknikken kalles "Return-to-Zero" fordi signalet *alltid* avslutter et bit med lavt nivå. Mens NRZ-koding maksimalt tar opp en båndbredde for sin 1.harmoniske tilsvarende $f_{t,max} = \frac{1}{2}_B = \frac{1}{2T_B}$, opptar RZ-koding det dobbelte; $f_{t,max} = f_B = \frac{1}{T_B}$. En annen måte å betrakte denne type koding på, er å se på en "1" som tilstedeværelse av klokkefrekvensen, og en "0" som fravær av den.

Enkel Manchester-koding kalles også bifase-koding eller tofase-koding. Prinsippet er at et pulstog skiftes 180° i fase, altså inverteres, etter hvilken bitverdi som skal representeres. I Manchester1-koding (M1) er "1" representert ved 0° faseforskjell mellom kodet signal og ikke-skiftet pulstog, mens "0" representeres av ±180° faseforskjell relativt til ikke-skiftet pulstog. Manchester2-koding (M2) har motsatt representasjon.

Dette kan også betraktes ut i fra transisjonene i midten av hvert bit. Negativ transisjon gir "1", mens positiv transisjon gir "0" (M1). Alternativt kan vi si at logisk høyt nivå representeres av de to bitene "10", mens logisk lavt nivå representeres av bitene "01". Metoden er avledet av generell faseskift-koding, PSK (PhaseShiftKeying), der fasen kan skiftes i sprang på $\frac{360°}{n}$, der $n = 2^k$ (k er et heltall) er antall kvantiseringsnivåer. Eksempelvis dekoder n=8 tre bit i hver periode, og fasen skiftes i sprang på 45°. Med et diskontinuerlig (digitalt) signal, er det bare mulig å skifte fasen i sprang på 180°.

Miller-koding kalles også delay-koding, altså forsinkelseskoding. En transisjon i ukodet datastrøm fra logisk lav til høy, blir sendt som en puls forsinket en halv bitperiode, $\frac{1}{2}_B$, i forhold til nivåskiftet. Denne teknikken gir flere transisjoner enn NRZ-koding, men færre enn RZ-koding.

Fordeler og ulemper kan trekkes ut av beskrivelsen for de forskjellige kodeteknikkene. *Fasemarginen* angir hvor stor andel av bitperioden klokking eller sampling av bitet kan skje i. En bitperiode tilsvarer 2π radianer, eller 360°, og en tilsvarende fasemargin betyr at sampling av bitverdien kan skje hvor som helst i bitperioden. Parametere for "Mark/Space"-koding forutsetter her NRZ-L -koding som utgangspunkt. Disse variantene kalles NRZ-M(ark) og NRZ-S(pace). Brukes en annen teknikk som utgangsform, fåes de samme parameterne for Mark/Space-varianten som for utgangspunktet. Sammenlikningene er vist i tabell 2.1.

Jeg forkastet bifase-modulasjon (M1/M2) som linjekode for data. Årsaken til dette var den ekstra kompleksiteten den ga i mottakeren spesielt, men også fordi det er vanskeligere å implementere senderen. Det er enklere å imitere NRZ- og RZ-data med en funksjonsgenerator enn bifase-modulerte data. Bifase-modulasjon egner seg dessuten ikke så godt dersom pulsbredden er av vital betydning.

Egenskap	**NRZ**	**RZ0/RZ1**	**M1/M2**	**Space/Mark**
transisjoner per bit	0 eller 1	0, eller 2	1 eller 2	0, eller 1
DC-komponent	dataavhengig	dataavhengig	ingen	dataavhengig
koherent mottager	nødvendig	nødvendig	ikke nødvendig	nødvendig
feil/støy-resistens	liten	middels	stor	middels
fasemargin i mottager	2π	π	π	2π
maksimal datarate	lav	middels	høy	middels
minimum båndbredde	$f_{t,min} = \frac{f_B}{2}$	$f_{t,min} = f_B$	$f_{t,min} = f_B$	$f_{t,min} = \frac{f_B}{2}$
senderkompleksitet	liten	middels	middels	middels
mottagerkompleksitet	middels/stor	middels	liten/stor(*)	middels/stor

Tabell 2.1: Sammenligning av forskjellige kodingstyper

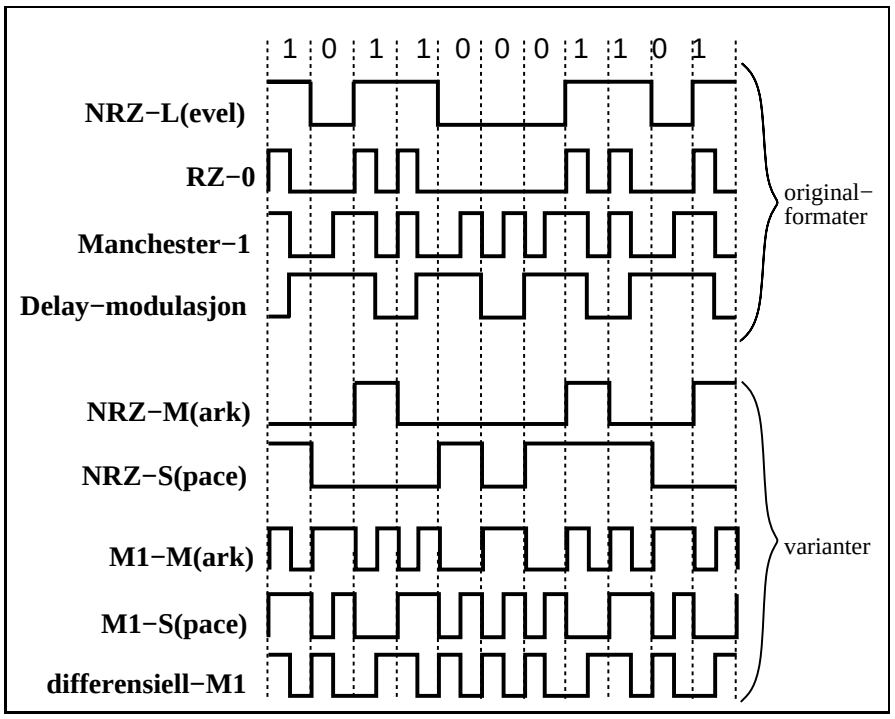
"Mark-Space" og delay-modulasjon er varianter av andre teknikker som ikke gir bedre synkronisering enn utgangspunktet. Riktignok gir de som regel større båndbredde, men dette er ikke en kritisk faktor i det systemet jeg skal konstruere. Antallet transisjoner per bit, og regelmessigheten av transisjoner er her viktigere faktorer. "Space-Mark"-koding med utgangspunkt i RZ- eller NRZ-koding, har høyere støy-immunitet enn grunnformene, og dermed lavere feilrate. Men til gjengjeld må en tilstandsmaskin dekode dataene i mottager, og denne ekstra kompleksiteten gjorde at jeg forkastet også disse teknikkene.

Konklusjonen er at RZ-koding er enkleste løsning, og dessuten var demodulasjon av RZ-signal omtalt flere steder i tilgjengelig litteratur, slik at jeg hadde eksisterende løsninger å bygge på. Jeg har hele tiden valgt den enkleste løsningen, fordi den har størst mulighet for å gi resultater ved første forsøk på implementasjon. Selv om løsningen ikke fungerer godt nok, kan den gi verdifulle data å bygge videre på.

2.3 Overføringsprotokoll

2.3.1 Rammeformat

Fordi *jeg* har definert systemets oppbygning, står jeg også fritt til å definere protokollen for oversending av data til enheter. Vanligvis defineres en protokoll for data-transport uten at transport-systemets oppbygning er gitt på forhånd. Protokollen defineres ut i fra kriterier som sikkerhet, fleksibilitet og spesielt effektivitet gitt ved stor effektiv båndbredde. Realiseringen av systemet må da gjøres med basis i protokollen. For min del har det vært nesten motsatt. Protokollen har delvis blitt definert slik at den skal passe godt for analogdelen, og dessuten slik at digitaldelen skal kunne realiseres relativt enkelt. Rammeformatet er gitt av figur 2.2, der både inndeling i data-felter og bit-innhold er vist.

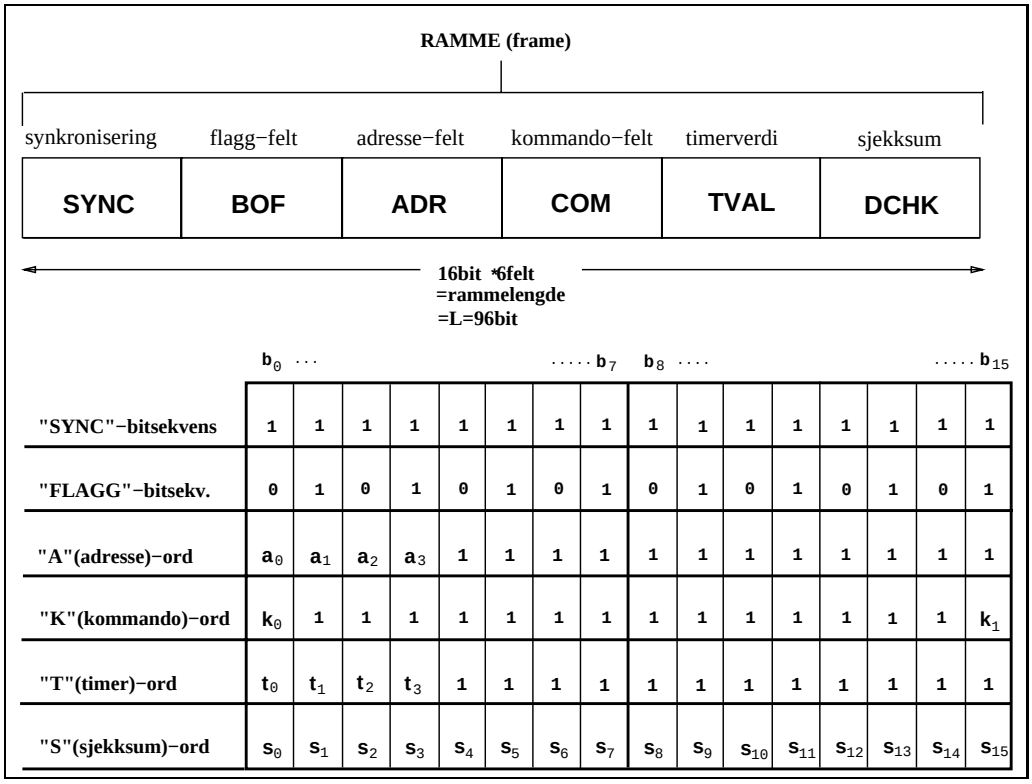


Figur 2.1: Forskjellige digitale kodeteknikker

Data-rammene er av fast lengde, L_r = 80 bit + *SYNC*, og alle rammer er like av oppbygning, men ikke like med hensyn på *bit-innhold* selvsagt. En ramme er satt sammen av seks deler. Første del er et synkroniserings-felt: *SYNC*. Andre del er et flagg-felt: *BOF*. Tredje del er et adresse-felt: *ADR*. Fjerde del er et kommando-felt: *COM*. Femte del er et felt for timer-verdi: *TVAL*, og siste del er et sjekksum-felt: *DCHK*.

Med unntak av synkroniserings-feltet, er lengden av hvert felt, L_f, lik 1ord=2byte=16bit. Lengden av en ramme, L_r, er oppgitt med uspesifisert *SYNC*-lengde, fordi denne teoretisk kan være uendelig lang. I praksis kan lengden settes etter som hvor lang tid mottager trenger for å synkronisere sin klokke, R_x, til senderens referanse, T_x. For å gjøre ramme-definisjonen mest mulig konsistent, er ideéll lengde på *SYNC*-feltet lik 16bit.

Det er ikke strengt nødvendig med synkroniserings-felt mellom påfølgende rammer uten dødtid imellom dem. Men et minimum på 16 synkroniserings-bits er fornuftig, igjen for å få konsistens, men også for at mottager-klokke skal kunne justere seg bedre inn. Dette kan være kritisk dersom for eksempel siste felt (=sjekksum) i en ramme inneholder lite synkroniserings-punkter, det vil si lite "1"-ere.



	b_0 ...							b_7	b_8							b_{15}
"SYNC"−bitsekvens	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
"FLAGG"−bitsekv.	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
"A"(adresse)−ord	a_0	a_1	a_2	a_3	1	1	1	1	1	1	1	1	1	1	1	1
"K"(kommando)−ord	k_0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	k_1
"T"(timer)−ord	t_0	t_1	t_2	t_3	1	1	1	1	1	1	1	1	1	1	1	1
"S"(sjekksum)−ord	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	s_{12}	s_{13}	s_{14}	s_{15}

Figur 2.2: Protokollens rammeformat

2.3.2 Synkroniserings-felt

Synkroniserings-feltet, *SYNC*, skal gi mottager informasjon om klokketakten til senderen. Feltet må inneholde maksimalt med synkroniserings-punkter, og dette betyr for RZ-koding at bit-innholdet i feltet kún er "1"-ere. Lengden av feltet er som før nevnt avhengig av mottakerens egenskaper. Teoretisk er optimal løsning en uendelig lang sekvens, men fasefeilen i en PLL-demodulator er eksponensielt fallende med tid. Derfor bør *SYNC*-feltets lengde avpasses etter hvor lang tid det tar før mottagerens *statiske* fasefeil er dominerende, det vil si den *dynamiske* fasefeil $\leq$ statisk fasefeil.

2.3.3 Flagg-felt

Flagg-feltet er en spesiell kombinasjon av bit som detekteres av fast logikk i mottageren, og starter innlesing av en ramme. Den første "0"-eren i en ramme er alltid første bit, eller laveste bit =LSB, i *BOF*. Bit-innholdet i startflagget er

$$BOF=\underline{\{0101010101010101\}}$$

der bitet lengst til høyre er MSB. Denne kombinasjonen er motstandsdyktig mot feil fordi den har et likt forhold mellom "0"-ere og "1"-ere og fordi det er en regelmessig/repeterende sekvens. Dette betyr at det er meget liten sannsynlighet for at en annen bit-sekvens ødelegges av støy på en slik måte at den blir identisk med *BOF*. Spesielt er det viktig at synkroniseringsfeltet ikke forveksles med *BOF*. Dersom $SYNC=\{11...11\}$ skal bli til $BOF=\{01...01\}$, betyr dette at annethvert bit er klokket inn på feil nivå og/eller at støy har invertert annethvert bit. Sannsynligheten for dette er meget liten.

2.3.4 Adresse-felt

Adresse-feltet, *ADR*, inneholder adresse til én spesifikk enhet blant mange som står i nettet, som senderen vil opprette forbindelse med. Adressefeltet er på 16bit, og totalt er det derfor $2^{16} = \underline{65536}$ mulige adresser. Hvilket igjen vil si 65536 enheter, dersom man forutsetter at alle skal ha unike adresser. Jeg har bare brukt 4bit av disse 16 fordi full utnyttelse ville gi både større arealforbuk på chipen, og fordi adressen settes eksternt via pinner. Med 35 pinner tilgjengelige for analog/digital-I/O, ville det vært sløseri å bruke 16 pinner til å sette adressen. Med 4bits adresse er det mulig å sette opp et system med maksimalt 16 enheter, men dette mener jeg er tilstrekkelig for demonstrasjonsformål.

2.3.5 Kommando-felt

Kommmando-feltet har, analogt med adresse-feltets 65536 unike adresser, mulighet for å kode 65536 ulike kommandoer. Systemet slik det er definert, trenger bare *tre* kommandoer for å utføre de ønskede funksjoner. Ergo brukes bare to bit av totalt 16 til å kode kommandoene. Det kan derfor se ut som sløsing å bruke et eget felt til dette, men dette gir mulighet for å lage utvidelser senere, og det gjør ramme-formatet enkelt. Hastighet eller båndbredde er uansett ikke prioriterte mål med dette systemet, derfor kan slike "luksuriøse" kompromisser foretas.

2.3.6 Timer-felt

En av spesifikasjonene for systemet, var at forsinkelsen på utgangssvitsjen skulle kunne settes med en kommando. Denne kommandoen er kalt "Timer på". Kommandoen trenger et argument; verdien for forsinkelsen, *TVAL*. Dette feltet er som for de andre på 16bit. Det gir en maksimal oppløsning på $\frac{1}{65536}$-del = $\underline{0.00001528}$ av timerens telle-område. Men også her har jeg gjort en avveining, slik at bare fire av bitene brukes. Nok en gang for å spare logikk (mindre kompleksitet) og plass (areal). Dette gir dårligere oppløsning i timeren, slik at programmering av intervall foretas i grovere steg.

2.3.7 Sjekksum-felt

Det siste feltet i rammen er sjekksum-feltet, naturlig nok, siden dette feltet verifiserer resten av rammen som kommer foran dette feltet. Dette feltet brukes i sin fulle bredde, fordi feildekningsgraden er avhengig av lengden på sjekksummen, som er identisk med lengden av det tilbakekoblede skiftregisteret som beregner signatur for de mottatte data. Signaturen skal være identisk med mottatt sjekksum. Kommandoen "Stopp klokke" er deaktivert dersom signatur og sjekksum *ikke* er like.

Kapittel 3

Betraktninger og avveininger

3.1 Mottagning

Første og viktigste komponent i mottageren er fotodetektoren; den lysfølsomme strukturen på Silisium-brikken. Karakteristikken til denne delen bestemmer som regel båndbredde og signal/støyforhold for hele mottageren. I en konvensjonell CMOS-prossess er det mulig å lage to forskjellige typer lysfølsomme komponenter:

- Fotodioder
- Fototransistorer

Fotodioder er sterkt ulineære, men har desto bedre frekvensrespons. Spesielle fotodioder, såkalte PIN-dioder, bygd av egnede materialer som GaInP, GaAsIn og lignende, benyttes til digital kommunikasjon med klokkefrekvenser over 1GHz. De bygges opp med en tredimensjonal struktur, og dette krever svært spesialiserte framstillings-prossesser. Slike hastigheter er ikke mulig å oppnå dersom fotodioden realiseres i standard CMOS, men forsøk [1] tyder på at de kan fungere i praksis opp til 1MHz.

Fototransistorer kan lages i flere varianter i CMOS. De har god linearitet innenfor et begrenset område og meget god dynamikk. De er likevel lite brukt i optoelektronisk kommunikasjon fordi virkelig stor hastighet krever andre materialer enn Silisium, spesielle prossesser og dyr framstilling. Dessuten kan ikke hastigheten til fototransistorer generelt måle seg med hastigheten til fotodioder. Lineariteten er til gjengjeld mye bedre.

Den andre delen av en typisk fotomottaker består av en forsterker-krets med "gain"-kontroll (AGC) for analoge signaler, eller en terskel-krets for digitale signaler, slik at utslaget til signalet er tilstrekkelig. Denne delkretsen kaller jeg heretter en *pulsformer*.

3.2 Demodulasjon

Demodulasjon av et signal sammensatt av en bærebølge (klokke), og modulerings-signalet (data), betyr egentlig å skille de to fra hverandre. Demodulatoren gjør dette enten indirekte ved først å *synkronisere* seg til klokke, eller direkte ved å synkronisere seg til bit. Demodulasjon i digital kommunikasjon kan med andre ord ha en dobbel effekt:

- Bitsynkronisering for datagjenvinning
- Klokkesynkronisering for klokkegjenvinning

Det første punktet er det primære, fordi enkelte systemer ikke er avhengige av å gjenvinne klokke for å gjenvinne data. Bitsynkronisering kalles også symbolsynkronisering, fordi et signalnivå ikke nødvendigvis koder bare ett, men flere bit (et symbol). Klokkesynkronisering kalles også bærersynkronisering fordi klokkefrekvensen sees på som en bærebølge akkurat som i analog kommunikasjon. Jeg vil ikke komme inn på systemer som ikke krever klokkesynkronisering/gjenvinning, fordi oppgaven forutsetter nettopp det. Men jeg kan kort nevne at disse systemene stort sett benytter seg av det man kaller blokk-synkronisering, eller ramme-synkronisering. Av de systemene som gjenvinner klokke, har vi to klasser:

- Koherente
- Nonkoherente eller ikke-koherente

Koherente demodulatorer gjenvinner klokke ved å generere en tilnærmet lik klokke med en lokal oscillator, som sammenliknes med referansen i signalet og korrigeres fortløpende. Dette kan være en rent digital prossess, en rent analog, eller en blanding. Utformingen avhenger av modulasjonsformen, men alle typer modulasjon eller koding kan benyttes.

Nonkoherente demodulatorer gjenvinner klokke *direkte* fra modulert signal. Dersom modulert signal faller bort, faller også klokke bort. Denne teknikken krever ingen lokal oscillator, men til gjengjeld er den avhengig av en transisjon i hvert bit for digital modulasjon.

Langt de fleste kommunikasjons-systemer bruker koherente demodulatorer. En hovedkomponent i slike demodulatorer er en PLL. Elementært blokkskjema for en PLL-krets er gitt i figur 3.1. Skjemaet inkluderer bare de blokkene som alltid må inngå i en PLL.

Figur 3.1: Blokkskjema for PLL-krets

3.3 PLL-demodulator

3.3.1 PLL-betegnelser, variabler, konstanter og begreper

Betegnelser:

- PLL = PhaseLockedLoop; faselåst sløyfe, eller fasestyrt reguleringssløyfe
- PD = PhaseDetector; fasedetektor eller fasekomparator
- PHD = samme som PD, brukt i eldre litteratur om emnet
- VCO = VoltageControlledOscillator; spenningsstyrt oscillator

Variable størrelser:

- $v_i(t)$ og $V_o(t)$ = inngangs-signal; sender-referanse eller kodet signal, og utgangs-signal; mottager-referanse eller synkronisert klokke
- ω_i og ω_o = inngangs-vinkelfrekvens, $2\pi f_i$, og utgangs-vinkelfrekvens, $2\pi f_o$
- θ_i og θ_o = inngangsfase og utgangsfase i radianer
- $v_d(t)$ = differanse-signal fra fasekomparator
- θ_d = differansefase; utgangsfasens avvik fra inngangsfasen

- $v_c(t)$ = kontroll-signal; middelverdi av differanse-signalet over en bitperiode

Konstanter:

- K_d = PD-konstant; forholdet mellom differansespenning og differansefase i [Volt/radian]
- K_o = VCO-konstant; forholdet mellom utgangsfrekvens og kontrollspenning i [(rad/s)/Volt]
- K_h = endelig dempning (AC-forsterkning) i sløyfefilter; dimensjonsløs
- K_p = dempnings- eller forsterknings-faktor i eventuelt forsterker-ledd (spennings-tilpasning); dimensjonsløs
- K = PLL-båndbredde i [rad/s]
- ω_{o0} = VCO-senterfrekvens i [rad/s]

Begreper:

- "Lock" : tilstand der utgangsfrekvensen er synkronisert med inngangsfrekvensen, eller $(\overline{f_i - f_o}) = 0$
- "lock-in range") : frekvensområdet utgangsfrekvensen kan følge inngangsfrekvensen
- statisk fasefeil : fasedifferansen når utgangsfrekvens er lik inngangsfrekvens
- faseområde : differansen mellom den største negative og største positive differansefase som PD kan representere ved en proporsjonal spenning
- senterfrekvens : den tilnærmet stabile frekvens, ω_{o0}, som utgangs-signalet får dersom $v_i(t) = 0$
- PLL-båndbredde : uttrykk for hvor hurtige endringer i inngangsfrekvensen PLL kan følge uten å miste synkronisasjon

3.3.2 Oversikt over PLL-komponentene

En *enkel* PLL består av 3 hoved-komponenter:

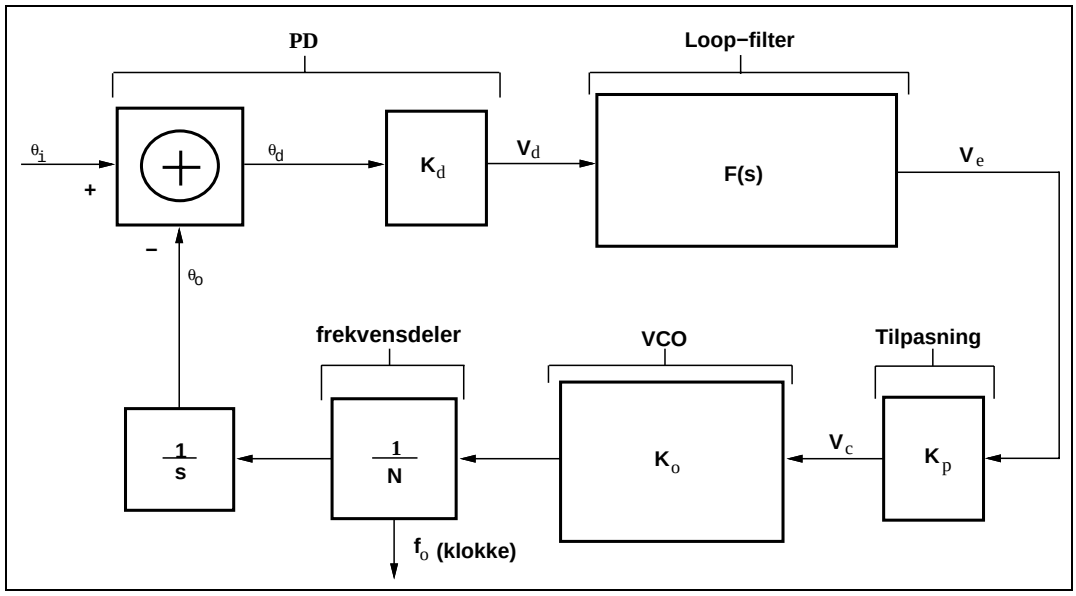
1. *Fasekomparator.* Kalles også fasedetektor. Den tar differansen mellom fasen til to inngangs-signaler, og sender den ut som en differanse-spenning, $v_d(t)$.

2. *Sløyfefilter.* Dette er gjerne en enkel integrator, som tar ut den løpende middelverdien, eller DC-verdien til differanse-spenningen (feilspenningen), og denne middelverdien sendes ut som $v_c(t)$.

3. *Spenningsstyrt oscillator.* Denne komponenten kalles oftest en VCO. Oscillatoren styres av spenningen fra sløyfefilteret, $v_c(t)$, og gir ut en spenning $v_o(t)$, som er den ene inngangsverdien til fasekomparatoren. Oscillatorspenningens vinkelfrekvens, ω_o, er en lineær funksjon av kontrollspenningen.

Dette er en grov forklaring som ikke gjelder for alle PLL-systemer, og utelater alle komponenter som ikke er absolutt nødvendige i et *analogt* PLL-system. Plasseringen av de forskjellige komponentene i en PLL er vist i figur 3.2, med inngangs- og utgangsvariabler. Tilpasnings-leddet er bare nødvendig dersom spennings-nivået fra sløyfe-filteret ikke samsvarer med det nivået som VCO krever. Denne konstanten kan også inkluderes i overføringsfunksjonen til filteret, men er her vist separat fordi jeg bruker én delkrets til filter og én til spennings-tilpasning i kretsløsningen.

3.4 Teoretiske betraktninger for PLL



Figur 3.2: Blokkskjema over PLL for systemanalyse

Hensikten med en PLL er å opprettholde en utgangsfrekvens som er eksakt lik inngangsfrekvensen. Endringen i utgangsfrekvensen betraktes derfor som et avvik fra referansen, altså inngangsfrekvensen. Dersom utgangsfrekvens er *lavere* enn inngangsfrekvens, får vi en negativ endring i utgangsfrekvensen. Med utgangsfrekvens *høyere* enn inngangsfrekvensen, får vi en positiv endring.

Over tid vil utgangsfrekvensen nærme seg inngangsfrekvensen slik at frekvensendringen blir et periodisk signal med like stor del under som over verdien til inngangsfrekvensen. Med andre ord er endringen i utgangsfrekvens like mye negativ som positiv under stabile forhold; når alle transienter har dødd ut, og den *midlere* endringen er lik null.

Den dynamiske fasefeilen er proporsjonal med endringen av utgangsfrekvens, og vil derfor under stabile forhold også midles til null. Dette leder oss til å tro at den gjennomsnittlige fasefeilen da er null, men nei! Av forskjellige årsaker vil utgangsfasen *alltid* ende litt foran eller litt bak inngangsfasen. Dette er en konsekvens av naturens innebygde trang til å skape kaos, og her manifisterer det seg ved elektronikkdesignerens pest; **offset**. Dersom vi kunne skaffe en perfekt sender med null avvik i senderfrekvens, og null fasefeil i mottakeren var oppnåelig, kunne mottakeren synkronisere seg inn på senderfrekvensen for deretter å motta data i det uendelige uten ny synkronisering. Men siden dette ikke er noen realitet, er vi nødt til å ta hensyn til den konstante fasefeilen som kalles *statisk* fasefeil.

Vi ser av figur 3.2 at en PLL er et tilbakekoblet system, med *negativ* tilbakekobling, akkurat som en klassisk regulerings-sløyfe. På samme måte kan sløyfens egenskaper analyseres ved å finne overføringsfunksjonen. Inngangsvariabelen er fasefeilen, mens utgangsvariabelen er utgangsfasen. Sløyfens overføringsfunksjon er gitt som utgangsfasen delt på fasefeilen. Dette reduseres til et uttrykk som består av overføringsfunksjonen til hvert ledd i sløyfen multiplisert med hverandre.

Overføringsfunksjon for sløyfen dersom sløyfe-filterets frekvensrespons har en endelig verdi:

$$G(s) = (K_d K_h K_o F(s))/s \tag{3.1}$$

Sløyfe-*forsterkningen* er gitt av karakteristikken til PD, og VCO:

$$K \equiv K_d K_h K_o \tag{3.2}$$

Disse to ligningene forutsetter at AC-responsen til filteret har en endelig verdi, altså en verdi forskjellig fra null for uendelig høy frekvens. Brukes en ordinær integrator, eller et RC-lavpassfilter, er dette ikke lenger tilfelle og tilnærminger må brukes.

Sløyfens *båndbredde* er et uttrykk for hvor *hurtig* utgangsfrekvensen kan følge endringer i inngangsfrekvensen. Sagt på en annen måte er det et uttrykk for hvor *hurtige* variasjoner referansefrekvensen kan ha, *uten* at PLL taper synkroniseringen. Den er gitt av PD- og

VCO-karakteristikken alene dersom sløyfen ikke inneholder noe sløyfefilter. Da fåes en svært stor båndbredde. Et sløyfefilter og eventuelt et dempeledd (ofte integrert i samme krets), må som regel legges inn for å forhindre at PLL ikke bare følger referansefrekvensen, men også støy og harmoniske frekvenser til ønsket senterfrekvens.

Sløyfens 3dB-båndbredde med dempeledd fra sløyfefilter:

$$\omega_{3dB} = K_d K_h K_o \tag{3.3}$$

Overføringsfunksjonen for hele systemet, er gitt som forholdet mellom utgangsfase og inngangsfase. Legg merke til at dersom fasefeilen går mot null, går $G(s)$ mot uendelig. Overføringsfunksjonen for systemet går i sin tur mot én, hvilket betyr at inn- og utgangsfase er tilnærmet lik.

Overføringsfunksjon for hele systemet, $H(s)$:

$$\frac{\theta_o(s)}{\theta_i(s)} = \frac{G(s)}{1 + G(s)} = \frac{G(j\omega)}{1 + G(j\omega)} \tag{3.4}$$

Forenklet overføringsfunksjon til system, $H(s)$:

$$\frac{\theta_o(s)}{\theta_i(s)} = \frac{K}{s + K} \tag{3.5}$$

Gjennomsnittlig endring i utgangsfrekvens er avhengig av filterets DC-respons, $F(0)$, i tillegg til parameterne for PD, dempeledd og VCO:

$$\overline{\Delta\omega_o} = \overline{\theta_e} K_d F(0) K_o + V_{do} F(0) K_o - V_{co} K_o \tag{3.6}$$

Denne siste ligningen er et uttrykk for hvor mye ω_o svinger rundt senterfrekvensen. Denne svingningen ser man ved stabil tilstand som en sinus med konstant amplitude oppå kontrollspenningen dersom en PD med sinusoidal karakteristikk benyttes. En PD med triangulær karakteristikk, slik som de aller fleste *digitale PD'er*, gir en triangulær svingning. Amplituden på denne svingningen bør helt klart begrenses. Dersom den blir for stor, blir systemet heller aldri stabilt og synkronisering oppnås ikke. Kriteriet for stabilitet er at forholdet mellom endringen av utgangsfrekvensen og senterfrekvensen er mindre eller lik en halv:

$$\textit{Stabilitetskriterium: } \frac{\Delta\omega_o}{\omega_{o0}} \leq \frac{1}{2}$$

I praksis må forholdet være godt under en halv for at systemet skal kunne fungere.

Statisk fasefeil er avhengig av sløyfefilterets DC-forsterkning $F(0)$:

$$\theta_{eo} = \frac{-V_{do}}{K_d + K_d F(0)} = \frac{-V_{do}}{K_d \cdot (1 + F(0))} \tag{3.7}$$

Av ligning 3.7 ser vi at offset i PD, V_{do}, bør gjøres minst mulig. Videre bør karakteristikken til PD gjøres så bratt som mulig, altså *stor* K_d. DC-responsen (forsterkningen) til filteret bør også gjøres stor. Dette er logisk; anta at PD-offset er konstant=10mV, mens $K_d = 10V/rad$ og DC-forsterkning, $F(0)$=1000. Da får vi følgende verdi for den *konstante* (statiske) fasefeilen under synkronisert, stabil tilstand:

$$\theta_{eo} = \frac{0.01V}{10V/rad \cdot (1+1000)} = \text{cirka } \underline{10^{-6} rad}$$

Med andre ord en ganske marginal verdi, spesielt ved lave datarater og stor synkroniseringsmargin.

3.4.1 Parametervurderinger

En punktvis oppsummering av hvilke parametere i PLL-systemet som fortjener oppmerksomhet, og aktuelle verdier for disse parameterne kan settes opp etter den teorien som er gjennomgått. Den antagelsen jeg på forhånd gjør for senderen, er god stabilitet og derfor små variasjoner i referansesignalet V_i.

- Båndbredde $\omega_{3dB} \leq 10kHz$
- Dynamisk fasefeil ikke kritisk
- Statisk fasefeil, $\theta_{e0} \leq \pm\frac{\pi}{4}$ ved senterfrekvens
- Modulasjonsområde $\omega_L \leq \pm 5kHz$ rundt senterfrekvens
- "Lock-in"-tid $\leq 10^{-4}s$
- Variasjon i senterfrekvens uten referanse, $\Delta\omega_{o0} \leq 1kHz$

3.5 Gjensyn med PLL-komponentene

3.5.1 Fasedetektor: PD

Forkortelsen PD står for PhaseDetector, greit oversatt til fasedetektor. Av og til brukes også forkortelsen PHD, som står for det samme. Blokkskjema for PD er vist i figur 3.3. Matematisk funksjon er som følger: utgangsfasen trekkes fra inngangsfasen, og differansen

Figur 3.3: Blokkskjema over PD

multipliseres med en *konstant* som representerer karakteristikken til PD. Produktet er en differansespenning, V_d. I figur 3.3 er blokkskjema for en PD vist, med utgangspunkt i en analog multiplikator. Denne kan analyseres med hensyn på spenningsvariabler eller fasevariabler.

PD-karakteristikken er forholdet mellom differanse-spenningen ut av PD og fase-differansen inn. Forholdet er gitt av operasjons-området til V_d over operasjons-området til θ_e, og kalles K_d:

$$K_d = \frac{V_{d,max} - V_{d,min}}{\theta_{e,max} - \theta_{e,min}}$$

3.5.2 Digitale fasedetektorer

En analog fasedetektor kan også brukes til å beregne fasedifferansen mellom to *digitale*. For eksempel vil en analog multiplikator utføre samme funksjon som en XOR-port for to digitale inngangsvariabler, $v_i(t)$ og $v_o(t)$. Men den kombinatoriske kretsen er enklere i oppbygningen enn multiplikatoren, den har større båndbredde og liten offset dersom den er riktig konstruert.

Grunnet disse faktorene så jeg ingen grunn til å bruke en analog fasedetektor, og ytterligere avveininger er foretatt kún mellom forskjellige digitale fasedetektorer.

Digital, kombinatorisk PD

Idéellt sett er fasedetektoren en multiplikator, med sinus-spenninger som inngangsverdier. Dette er for å forenkle lineær analyse av systemet. Med digitale inngangs-signaler, brukes i praksis andre typer PD'er. En XOR-port utfører den samme funksjonen som en multiplikator med digitale inngangsverdier. XOR-porten er et eksempel på en kombinatorisk

Figur 3.4: Karakteristikk for XOR-PD med -90 grader faserelasjon

PD. Fase-karakteristikken er vist i figur 3.4. Legg merke til at fremstilt karakteristikk for XOR-porten her antar en faseforskjell mellom θ_i og θ_0 på $-90°$, og *ikke* nullfase. Også OR/NOR-porter kan brukes som fasedetektorer. Kombinatoriske porter er raske, og de fungerer selv om pulser går tapt i et pulstog. Deres viktigste begrensninger er at de har begrenset operasjons-område, og at de er avhengige av pulsbredden ("duty-cycle") på inngangssignalene.

Også AND/NAND-porter har egenskaper som gjør at de kan brukes som fasedetektorer. Både OR/NOR og AND/NAND har samme operasjonsområde som XOR/XNOR-porter, $(-\pi/2, +\pi/2)$. Men de er likevel mindre egnet fordi differanse-spenningen er halvert i forhold til XOR/XNOR. OR- og NAND-porter utnytter bare området $V_d \in [V_{dd}/2, V_{dd}]$, NOR- og AND-porter utnytter tilsvarende området $V_{dd} \in [0, V_{dd}/2]$, mens XOR/XNOR-porter utnytter hele området fra 0V til V_{dd}.

Dette er illustrert i figur 3.5. Karakteristikk for NAND-port og NOR-port er ikke tatt med fordi NAND er ekvivalent med OR, og NOR er ekvivalent med AND i PD-sammenheng. Riktignok forskjøvet 180°, altså invertert. Her er fasefeilen θ_e relativt til null fasedifferanse, slik at karakteristikken til portene er forskjøvet langs x-aksen i forhold til figur 3.4. Karakteristikkene er repetérende utenfor området $\pm\pi$ radianer.

En XOR/XNOR-port har en større variasjon i $v_d(t)$ enn de andre typene kombinatoriske porter. Dette gir seg uttrykk i PD-karakteristikken. En standard CMOS XOR-port vil ha en PD-karakteristikk:

Figur 3.5: (a-c) PD-karakteristikk for kombinatoriske porter

$K_{d,XOR} = V_{dd}/\pi = 5V/\pi(rad) = \underline{1.59V/rad}$

Til sammenlikning vil de andre kombinatoriske portene ha en PD-karakteristikk:
$K_{d,AND/OR} = (V_{dd} - V_{dd}/2/\pi = 2.5V/\pi(rad) = \underline{0.795V/rad}$

En stor K_d er ettertraktet, og i en slik henseende er AND/OR-porter underlegne XOR-porten som PD.

AV PD-karakteristikken til AND-, OR- og XOR-porten ser vi at midtpunktet i operasjonsområdet til V_d tilsvarer en absolutt fasedifferanse $\theta_{e0} = \frac{\pi}{2}$, eller 90° faseforskjell. Derfor er det best med tanke på symmetrisk respons å sammenlikne referansen med et 90° faseforskjøvet VCO-signal. Men dersom referansen faller bort, hvilket betyr ingen innkommende pulser, vil den effektive faseforskjellen bli lik null.

Resultatet er at differanse-spenningen fra AND- og OR-porten vil falle til $V_{d,min}$, henholdsvis 0V for AND-porten og 2.5V for OR-porten. VCO vil da ende på en helt annen frittløpende frekvens enn referansefrekvensen. XOR-porten derimot, vil fremdeles ligge på $\frac{V_{d,max}}{2}$ på grunn av sin kombinatoriske funksjon. Konklusjonen blir derfor at XOR-porten er et bedre valg som PD.

Dersom ett eller begge inngangs-signalene til en kombinatorisk PD har en duty-cycle forskjellig fra 50%, endres karakteristikken. Dette kan sammenlignes med offset i et idéellt sinus-signal, slik at DC-innholdet er forskjellig fra 0. I en idéell multiplikator, vil utgangen inneholde produktet av offset-verdiene. Dersom bare det ene inngangs-signalet inneholder en offset, blir offset på utgangen lik null, men utgangsverdien *faseforskyves* proporsjonalt med offset-verdien. Hvilket betyr at verdien for feilsignalet ved null *relativ* faseforskjell, eller operasjonspunktet V_{d0}, forskyves proporsjonalt med offset.

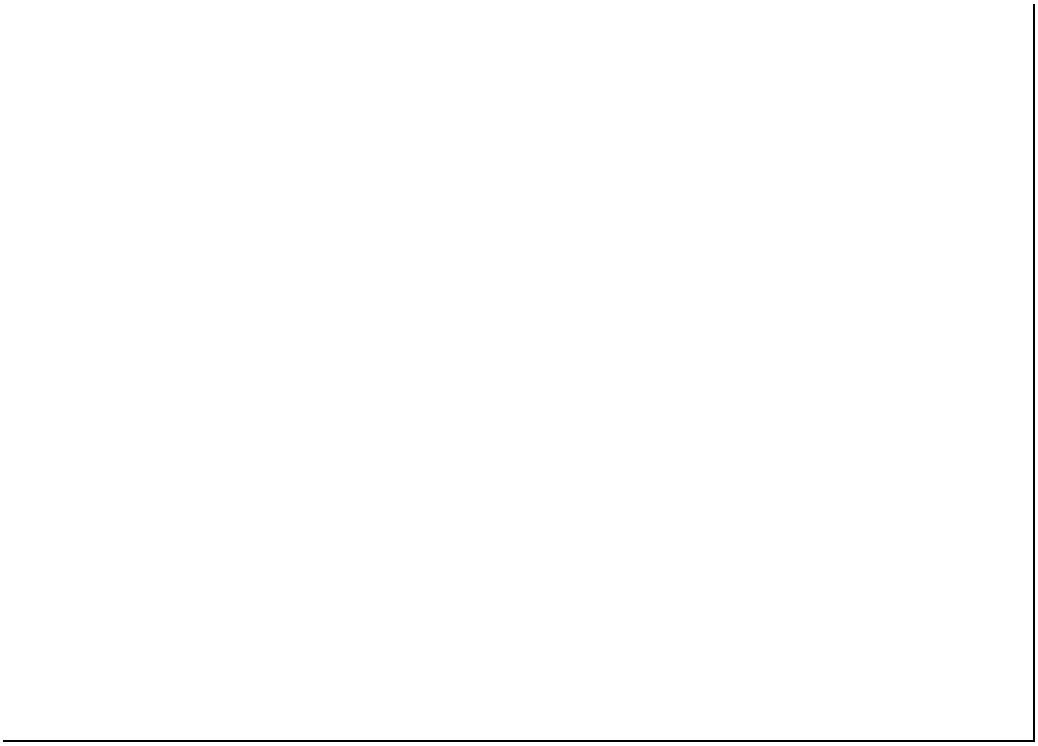
I en XOR-PD skjer det samme, men også selve *karakteristikken* komprimeres rundt V_{d0}. Konsekvensen er at effektiv K_d blir mindre. I figur 3.6 er karakteristikken for en XOR-PD vist med DutyCycle på ett eller begge signalene forskjellig fra 50%.

En annen kilde til offset er porten selv. Dersom omslagskarakteristikken ikke er symmetrisk om $\frac{V_{dd}}{2}$, eller porten har hysterese, forskyves karakteristikken langs y-aksen. Kriteriet for null offset introdusert i en kombinatorisk PD, er at:

$$V_{TH,high} = V_{TH,low} = \underline{2.5V}$$

Digital, sekvensiell PD

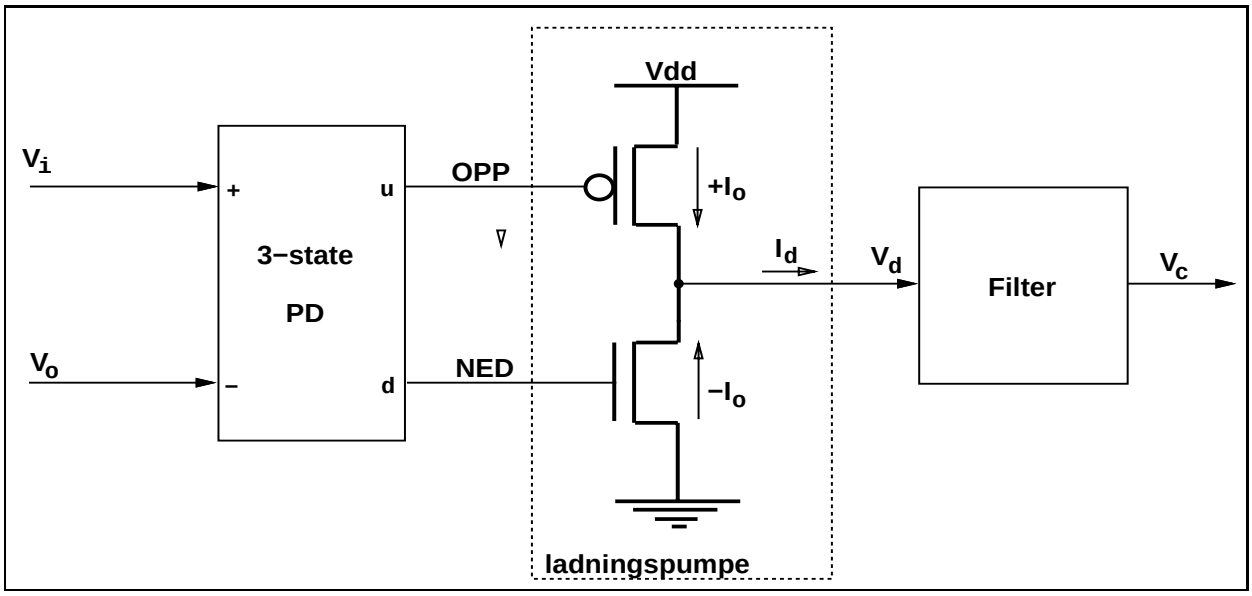
Sekvensielle PD'er for digitale signaler har større operasjons-område enn porter. Men de fleste fungerer ikke dersom pulser mangler i et inngangs-signal, det betyr *ikke-kontinuerlig* operasjon. I tillegg er støymarginen dårligere enn for porter, og de er langsommere slik at



Figur 3.6: Karakteristikk for XOR-PD med varierende “duty-cycle” på input

de ikke kan brukes ved så høye frekvenser som porter. Men til gjengjeld kan sekvensielle fasedetektorer med to utganger, kalt "3-state" PD'er, brukes til å styre to CMOS transistor-svitsjer (se figur 3.7).

Denne kretsen kalles en "ladningspumpe", fordi den enten forsyner et påfølgende filter med ladning, subsidiært strøm, dersom Q1 er på og Q2 er av, eller trekker ladning ut av det dersom forholdet er motsatt. I tillegg kan begge transistorene være avstengt, slik at nesten ingen ladning flyter hverken ut eller inn av kretsen. Altså en høyimpedans-tilstand, kalt "Z"-tilstand, både til jord og V_{dd}.Kretsskjema for PD og ladningspumpe er vist i figur 3.7. Funksjonen til kretsen er gitt av tabell 3.1.

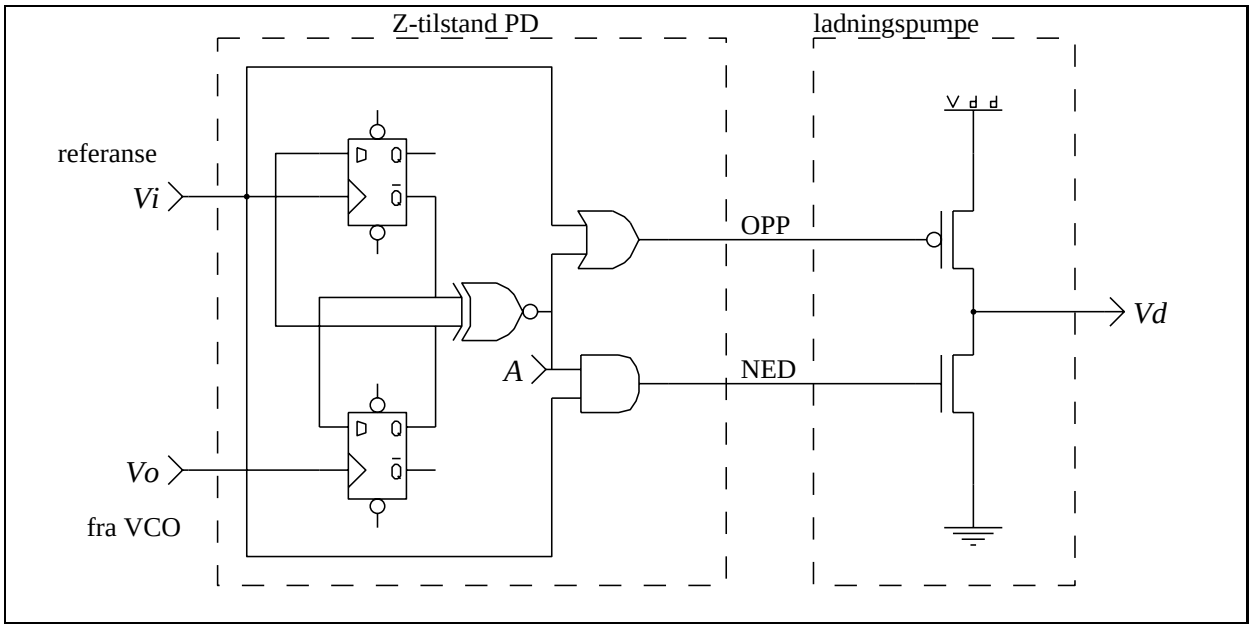


Figur 3.7: "3-state" PD og ladningspumpe

"Z"-tilstanden er viktig å merke seg. Denne tilstanden inntreffer når faseforskjellen er lik null. Begge transistorene i ladningspumpa er da stengt av. Spenningen ut fra filteret forblir den samme, og frekvensen ut fra VCO er konstant. Dette gjelder bare dersom en kapasitiv last er koblet til utgangen av ladningspumpa, slik at ladningen bevares tilstrekkelig lenge. Ellers vil ladningen forsvinne på grunn av lekkasjestrømmer. Siden denne kapasitansen alltid er inkludert i filteret, har jeg ikke tatt den med i figur 3.7.

For en "3-state PD", brukes som oftest flankene til V_o for sammenligning med flankene til V_i, mens flankene til samme signal faseforskjøvet med $-90°$, brukes til å sample bitverdien. En type realisering av en slik "3-state PD" er vist i figur 3.8. Denne realiseringen er avhengig av "duty-cycle" til V_o, men fungerer også uten innkommende pulser i V_i. Dette må tas med noen forbehold, i og med at jeg selv kom fram til denne kretsen.

Det er verdt å merke seg her at operasjonen til kretsen *ikke* er uavhengig av pulsbredden til inngangs-signalene. Operasjonsområdet stiger lineært med økende "duty-cycle" på V_i, inntil idéellt forhold oppnås med et symmetrisk signal (50% duty-cycle). Operasjonsområdet er da fra $-180°$ til $+180°$, altså det dobbelte av hva en PD konstruert med kombinatoriske porter oppviser under samme forhold.



Figur 3.8: Kretsskjema for pseudo-"3-state" fasedetektor

Sekvensielle PD'er har samme operasjonsområde for differanse-spenningen som XOR-porten. Men fordi operasjonsområdet med hensyn på fasedifferansen er det dobbelte, blir PD-karakteristikken halvparten:

$$K_{d,SPD} = \frac{V_{dd}}{2\pi} = \frac{5V}{2\pi(rad)} = \underline{0.795V/rad}$$

Karakteristikken til en sekvensiell PD er vist i figur 3.9. Akkurat som for en kombinatorisk, digital PD er det en diskontinuitet ved yttergrensen til operasjonsområdet. Men i motsetning til for eksempel en XOR-PD, bruker ikke en sekvensiell PD halve bitperioden til korreksjon for neste periode. En sekvensiell PD følger innkommende flanker gjennom hele bitperioden.

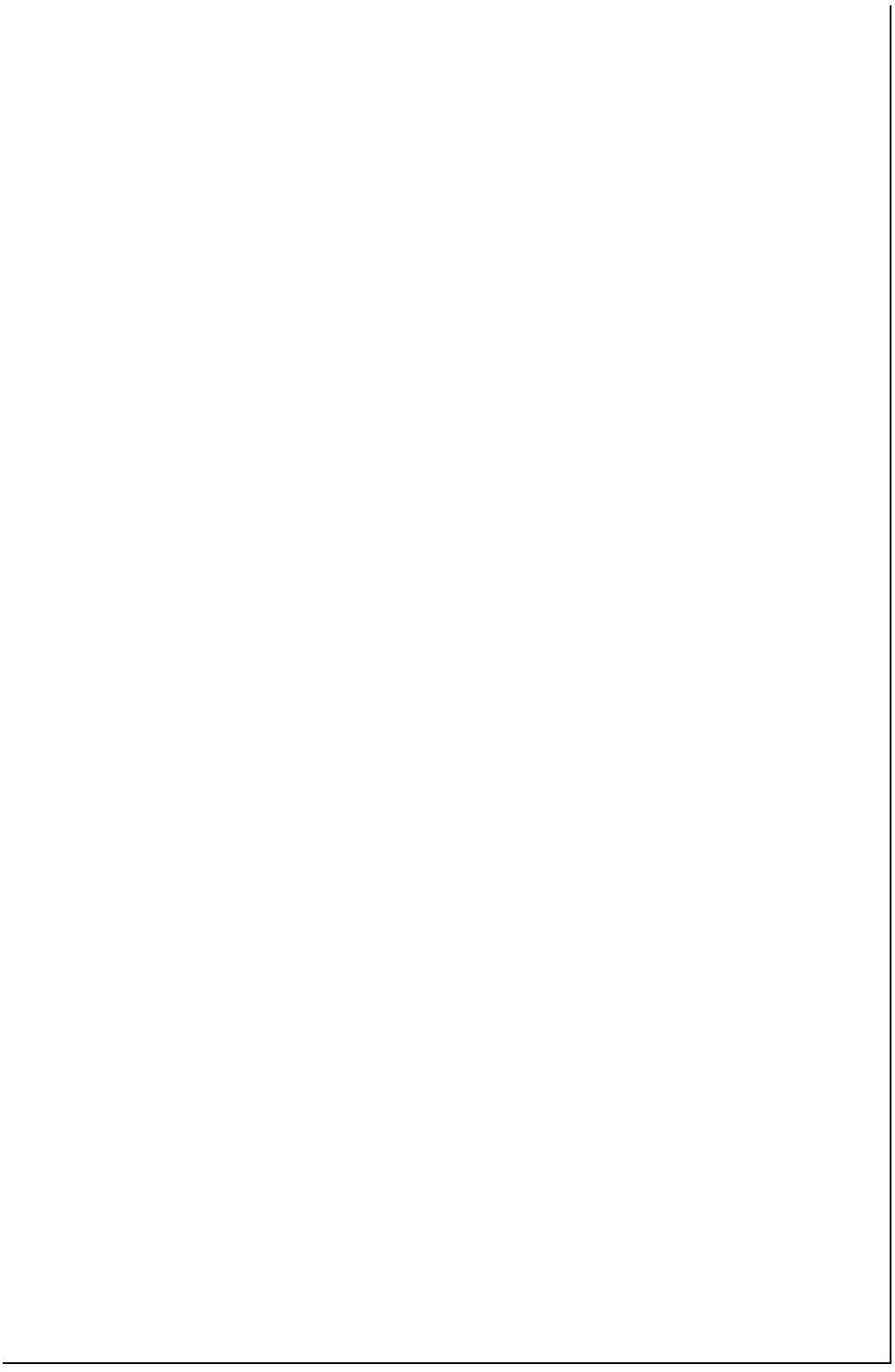
Sett at innkommende flanke faller utenfor en bitperiode sett fra mottageren. Da "antar" PD at neste innkommende flanke faller helt i begynnelsen eller helt i slutten av neste

Figur 3.9: Relasjon mellom fasefeil og differansesignal for sekvensiell PD

bitperiode. Dersom nest siste flanke lå helt i begynnelsen av en bitperiode, er differansespenningen det maksimale som PD kan gi ut. Men faller siste flanke utenfor faseområdet (en bitperiode), antar PD at flanken vil komme helt i slutten av neste bitperiode. Konsekvensen er at den gir ut lavest mulig spenning i håp om å treffe denne flanken. Dersom den manglende flanken bare skyldtes en manglende puls, er dette et feilgrep som kan medføre at PD bommer på neste innkommende flanke med mer enn en halv bitperiode, og bit'et klokkes feil.

Dette er en ulempe som mer enn oppveies av det store operasjonsområdet for fasedifferansen. Flere sekvensielle PD'er kan dessuten brukes som både fase- og frekvenskomparatorer. En type sekvensiell PD, kalt "2-state PD", er vist i figur 3.10 sammen med en "3-state" PD. Den fungerer ikke dersom pulser mangler i V_i, og kan derfor ikke brukes til RZ-demodulering. Til gjengjeld er operasjonen uavhengig av pulsbredde, eller "duty-cycle".

Dersom frekvensdelere (tellere) plasseres foran inngangene på en 2-state PD, kan faseområdet økes med en faktor gitt av tellerens størrelse. Dersom frekvensen deles ned med 3, økes faseområdet til det tredobbelte. Men en kan sjelden få noe uten å gi noe tilbake. I dette tilfellet blir tidsresponsen dårligere fordi de ned-delte frekvensene bare representerer et *gjennomsnitt* av de opprinnelige frekvensene, og variasjonene i $f_i^{'}$ og $f_o^{'}$ er bare gjennomsnittet av variasjonene i f_i og f_o. Tidsresponsen blir dårligere, og jeg vurderte aldri å bruke en slik teknikk.



Figur 3.10: Kretsskjema for “2-state” og “3-state” sekvensiell PD

Konseptet for en "3-state" PD kan brukes til å lage en "n-state" PD. Faseområdet utvides dersom flere tilstander er mulige. Som en oppsummering har jeg i tabell 3.2 listet opp sterke og svake sider ved de to hovedtypene digitale fasedetektorer.

3.5.3 Spenningsstyrt oscillator: VCO

Som navnet tilsier er en VCO styrt av en spenning. I PLL-sammenheng kalles denne spenningen *kontrollspenningen*, som benevnes V_c. Kontrollspenningen er som regel en tidsvarierende funksjon, $v_c(t)$, men for system-analyse brukes øyeblikksverdien (momentanverdien) V_c.

Spenningsstyrte oscillatorer kan i utgangspunktet lages av alle typer oscillatorer. Som regel er en oscillator bygd opp av et forsterkende element med *positiv* tilbakekobling gjennom en forsinkelse. Tidskonstanten, eller perioden til oscillasjonsfrekvensen, er gitt av forsterkningen og verdien av forsinkelsen. *Stor* forsterkning, og *liten* forsinkelse gir en høy oscillasjonsfrekvens. Maksimalfrekvensen er gitt av minimums-forsinkelsen og båndbredden til forsterkerelementet. En tilbakekoblet CMOS-inverter brukt som en spennings-referanse, er et eksempel på en tilsynelatende ustabil krets som likevel er stabil på grunn av inverterens begrensede båndbredde.

Dersom forsterkningen og/eller forsinkelsen kan kontrolleres med en spenning, blir oscillatoren en VCO. Begge deler er ofte mulig. En forsterker inneholder for eksempel en spenningsstyrt strømkilde. Økende spenning gir økende strøm, og derfor større forsterkning = høyere frekvens. Forsinkelsen er ofte et enkelt RC-ledd. Dersom resistans eller kapasitans kan styres med en spenning, oppnås (nesten) det samme som ved å øke forsterkningen. En CMOS N-transistor kan for eksempel brukes som en styrt resistans. Stigende spenning på "gate" til transistoren gir synkende resistans. Dette betyr igjen mindre tidskonstant = høyere frekvens.

Den viktigste parameteren for en VCO er forholdet mellom kontrollspenning, V_c, og oscillatorfrekvens, ω_o. Denne parameteren kalles K_o, og regnes som en konstant rundt arbeidspunktet ω_{o0}.

Karakteristikk for VCO:

$$K_o \equiv d\omega_o/dv_c = d\Delta\omega_o/dv_c \tag{3.8}$$

3.5.4 Sløyfefilter

Sløyfefilterets oppgave er å fjerne AC-komponenten fra feilspenningen, $v_d(t)$, og sende ut den varierende DC-komponenten som en kontrollspenning, $v_c(t)$ til VCO. Altså er dette en integrator, eller et lavpassfilter. Dette gjelder ikke for *alle* PLL-systemer, men unntakene er sjeldne og av liten interesse her.

Sløyfefilteret kan være et aktivt eller passivt filter. Hvilken type som velges, er avhengig av applikasjonen. Tabell 3.3 viser en sammenligning mellom aktivt og passivt filter. At et aktivt filter har egne poler, betyr at frekvenskarakteristikken til filteret er avhengig av ikke bare verdien til de passive komponentene, men også båndbredden til de aktive komponentene. Også passive komponenter har parasitt-kapasitanser og forbundet med dem, men disse er forholdsmessig mye mindre.

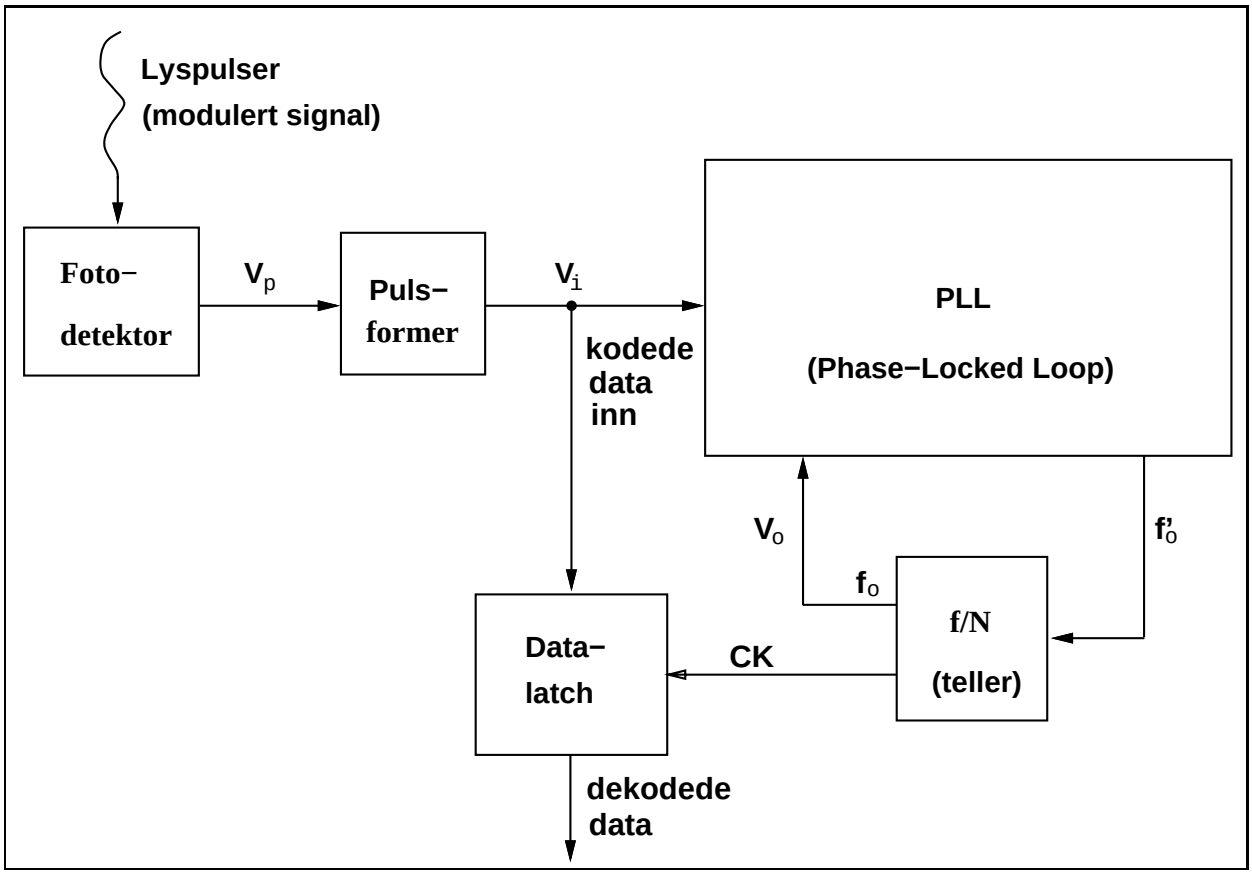
Drift og egenstøy er sammenliknet med forbehold om samme toleranser for de passive komponentene. Amplitudeavhengig kan også oversettes med mulighet for forvrengning. Forvrengning er synonymt med introduksjon av høyfrekvente komponenter, eller harmoniske frekvenser, og disse kan sees på som støy.

For PLL-kretser er den vanligste problemstillingen hva som er enklest. Dernest kommer hvilken DC-forsterkning som er nødvendig. Pris er en faktor som her står i forhold til kompleksitet. Stor kompleksitet betyr flere komponenter, hvilket igjen betyr større arealforbruk i VLSI-implementasjonen. Dessuten er komponentene ofte vanskeligere å lage på grunn av spesielle geometrier og lavere toleranser. Kort sagt er det vanskeligere å få kretsene til å fungere som forutsatt.

3.6 Oppsummering

Et foreløpig blokksjema over mottagerdelen, kan settes opp på grunnlag av de punktene jeg hittil har gjennomgått. Dette blokkskjemaet er vist i figur 3.11. Det er tre distinkte blokker; fotodetektor, fotodetektor-krets (pulsformer), og PLL-demodulator. Frekvensdeleren hører naturlig til PLL, men er vist som egen blokk fordi den er sammensatt av digitale kretser. Det samme gjelder data-latchen. Disse tre blokkene; foto-detektor, pulsformeren og PLL, utgjør hovedparten av de analoge delkretsene.

I det neste kapitlet vil jeg gjennomgå konstruksjonen av blokkene omtalt i dette kapitlet, pluss deler som er berørt i tidligere kapitler. Det aller meste av teorien som ble introdusert i dette kapitlet er tatt fra [4].



Figur 3.11: Beskrivelse av system uten digitaldel

OPP	NED	kommentar	innvirkning på V_d
0	0	$\theta_e < 0$	stiger
0	1	ugyldig	—
1	0	$\theta_e = 0$	uforandret
1	1	$\theta_e > 0$	synker

Tabell 3.1: Virkemåte til ladningspumpe styrt av "3-state" PD

Egenskap	**Kombinatorisk PD**	**Sekvensiell PD**
faseområde	$\pi/2$	π
linearitet over faseområdet	hele	hele
pulsbredde-avhengig	ja	nei
støymargin	middels	liten
båndbredde	stor	middels
karakteristikk ($V_{dd} = 5V$)	1.6V/rad	0.8V/rad
oppbygning	enkel	kompleks

Tabell 3.2: Sammenligning av kombinatorisk og sekvensiell digital PD

parameter	**aktivt filter**	**passivt filter**
DC-forsterkning	over 60dB	maksimalt=1
stabilitet	systemavhengig	alltid stabil
egenstøy	middels	liten
offset	ja	nei
drift	middels	liten
fasekarakteristikk	god	dårlig
amplitudeavhengig	ja	nei
egne poler	ja	nei
oppnåelig dempning	stor	middels
kompleksitet	stor	liten

Tabell 3.3: Sammenligning av aktivt og passivt filter

Kapittel 4

Konstruksjon av analogdel

4.1 Innledning om delkretsene

Alle forsterkerelementer er implementert som "wide-range" transkonduktans-forsterkere. Blant annet fordi disse er enkle og effektive, men også fordi man kan kontrollere utgangsstrømmen deres eksternt med en bias-spenning, V_b. Dette i motsetning til operasjonsforsterkere som opererer med spenningsforsterkning, og krever *eksterne* komponenter (resistorer) for å sette en gitt forsterkning. Dessuten er oppbygningen til operasjonsforsterkere mer kompleks, og de tar derfor opp mer areal.

Transkonduktans-forsterkerne tar opp lite silisiumareal sammenliknet med andre typer CMOS-forsterkere. Dessuten må transistorene i liten grad skaleres individuelt for at de skal fungere. Begrunnelsen for å bruke den gjennomført i design av hele analogdelen, er at dette medfører enkel konstruksjon, men spesielt fordi det er plassbesparende. Enkel oppbygning betyr at det er mindre risiko for konstruksjonsfeil eller utleggsfeil. Derfor er denne kretsen idéell for nybegynnere innen analog VLSI. Transkonduktans-forsterkere er grundig beskrevet i litteraturen, blant annet i boka til Carver Mead [3].

Passive komponenter jeg har brukt, omfatter kún kondensatorer. Disse er uten unntak realisert som poly1-poly2 kondensatorer for best mulig linearitet. Alle blokkene i analogdelen har både en avgrenset funksjon, i betydningen virkemåte, og en systemfunksjon. For eksempel har en XOR-port som statisk funksjon at den utfører logisk "xor" mellom to binære operander, mens dynamisk funksjon, eller systemfunksjon i PLL-systemet, er som fasedetektor.

Jeg går ut i fra at den som leser dette er kjent med virkemåten til de fleste elementære delene i systemet. Derfor vil jeg bare forklare funksjonen til systemblokker i detalj, mens jeg utelater triviell prat om forsterkere og lignende, unntatt der det er nødvendig med en oppklaring.

4.2 Fotodetektor

Fotodetektoren er ikke min fortjeneste. Her må jeg kreditere Morten Salamonsen for konstruksjon og utlegg. Idéen til "diff1"-kretsen og fotodioden kom fra veileder Tor Landes hånd. Jeg har gitt en enkel beskrivelse her, men for detaljer kan artikkelen til Salamonsen med flere [1] studeres.

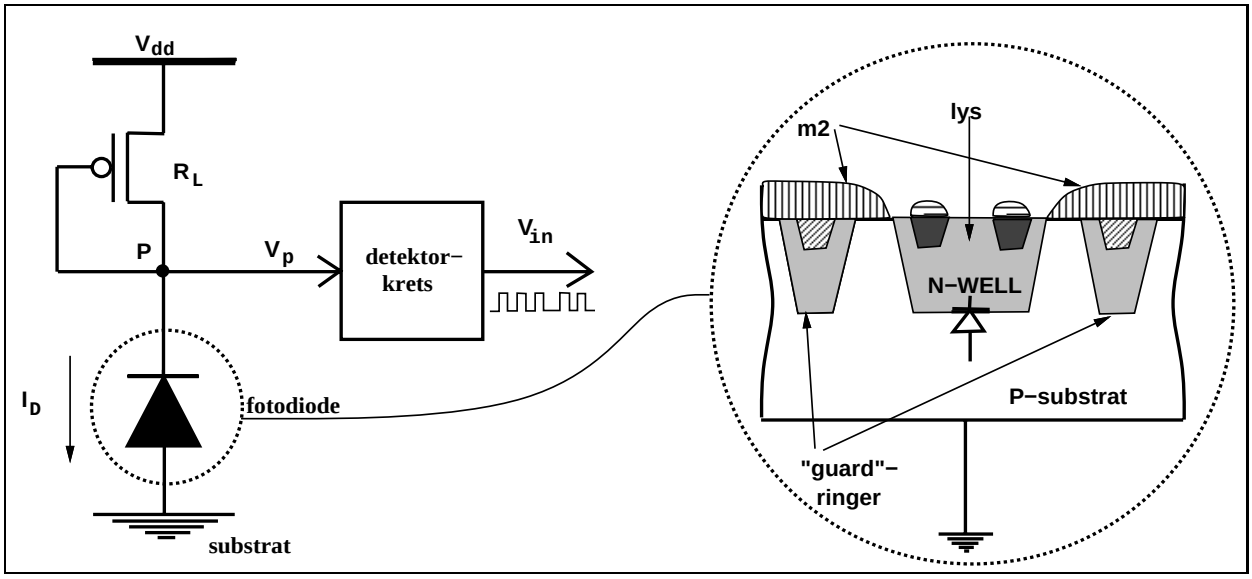
Denne kretsen er sammensatt av to delkretser: en fotoelektrisk sensor, og en pulsformer som gir fullt sving fra GND til V_{dd} på mottatt signal. Fotoelektrisk sensor er i hovedsak en fotodiode. Spenningen over dioden, kalt $v_p(t)$, moduleres av lyspulser. Pulsformeren, eller detektoren, er en analog differensiator-krets. $v_p(t)$ er inngangs-signal til differensiatoren. Utgangs-signalet, $v_i(t)$, er inngangsvariabelen til PLL-demodulatoren.

4.2.1 Fotoelektrisk sensor

Fotodioden er i hovedsak en N-type brønn med kontakt til substratet av P-type. Overgangen mellom substratet og brønnen blir PN-overgangen i dioden. Over den lysfølsomme overflaten ligger det tynne diffusjonslinjer (N-type diffusjon, samme type som brønnen), som har kontakt med brønnen. Denne diffusjonen samler opp frie ladningsbærere, løsrevet av fotoner, før de rekombinerer i brønnen. Hvor *mange* ladningsbærere som plukkes opp, og hvor *raskt* de samles opp, bestemmer responstiden til fotodioden. Antall ladningsbærere begrenses blant annet av brønn-materialets lave dopingsgrad.

Fotodioden er "reverse biased", altså forspent i sperreretningen. Over diffusjonslinjene ligger metall1-linjer som har kontakt til diffusjonen via diffusjonskontakter. Spenningen påsettes dioden gjennom disse metall-lederne. For å erstatte tapet av ladningsbærere i brønnen, strømmer elektroner inn. Vi har derfor en revers-strøm som er en funksjon av antall ladningsbærere som frigjøres, hvilket med en grov tilnærmelse er proporsjonalt med antall fotoner som treffer det lysfølsomme arealet. Antall fotoner per arealenhet tilsvarer lysintensiteten. Vi kan se på dioden som en styrt motstand. Denne motstanden er i serie med en fast lastmotstand, slik at spenningen i punktet mellom dem er avhengig av den styrte motstanden alene. Faller motstanden i dioden, altså fotodioden truffet av en lyspuls, faller også spenningen til V_{pL}. Stiger motstanden, altså fravær av lys eller ingen lyspuls, stiger også spenningen til V_{pH}.

Ser man på *strømmene* i punktet P i figur 4.1, blir det omvendt. Dersom motstanden i dioden faller, stiger strømmen I_D i verdi. Stiger motstanden, vil I_D synke. Størrelsen på lastmotstanden bestemmes ut i fra et kompromiss. Dersom den er liten, blir strømmen, I_D, *stor*, men spenningsdifferansen, $V_{ph} - V_{pl}$, blir liten. Motsvarende blir I_D liten mens $V_{ph} - V_{pl}$ blir stor dersom R_L velges stor. Lastmotstanden utgjøres av en diodekoblet P-transistor med W/L=8, altså aktiv last. Last-transistoren har en "Drain-Source" kapasitans, C_{ds}, som påvirker frekvensresponsen til fotodetektoren, og stige- og fall-tider.



Figur 4.1: Skjema for fotodiode-krets

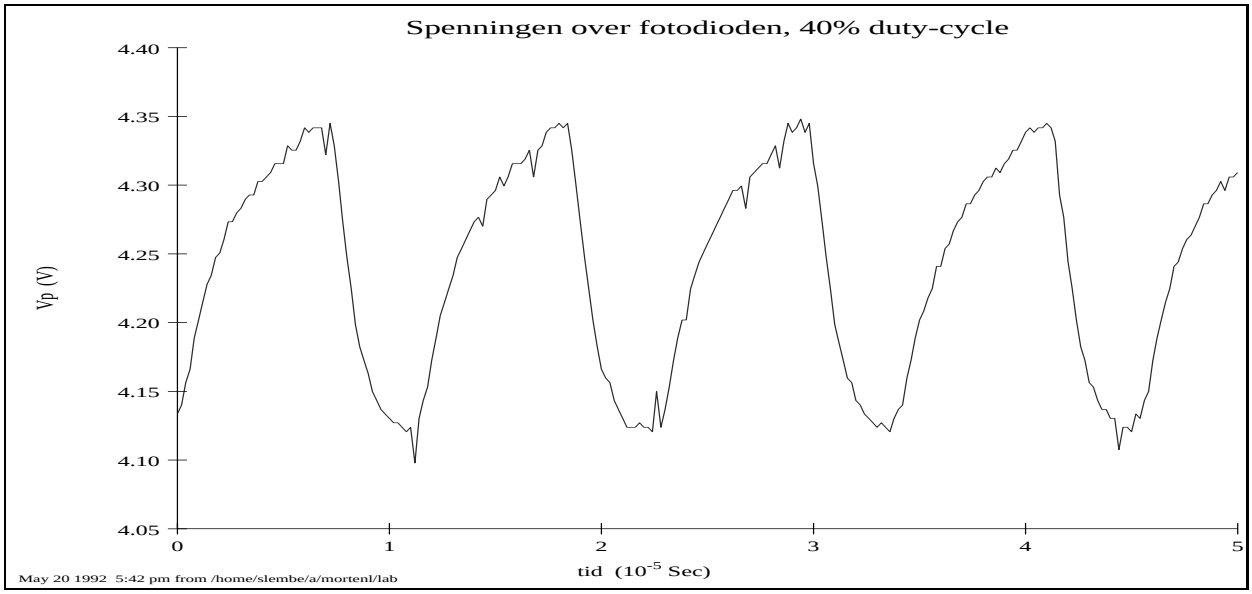
Målinger på realisert krets (se figur 4.2) viser at variasjonen i $v_p(t)$, som tilsvarer AC-komponenten, er stor nok ved 100kHz til å drive komparatorens utgang høy. Legg merke til støyspikerene i V_p. Støyspikerne er uttalte selv om denne målingen ble foretatt uten aktiv digitaldel. Målingene er foretatt med en lysemitterende diode (LED) som fotoelektrisk transmitter, med en 150 motstand i serie med signalkilden slik at pulset strøm er opp i mot det maksimale av det dioden kan lede. Altså med størst mulig lysstyrke på pulsene.

4.2.2 Detektorkrets

"Diff1"-kretsen er et eksempel på en "smart sensor". Utgangssignalet fra denne kretsen skal i teorien være uavhengig av inngangssignalets amplitude. Kretsen er bygd opp av en integrator og en komparator som vist i figur 4.3.

Komparatoren sammenligner det integrerte inngangssignalet med det opprinnelige inngangssignalet, og forsterker denne differansen. I klartekst betyr dette at dersom inngangssignalet er noen få millivolt større enn det integrerte signalet, går utgangen av komparatoren til V_{dd}. Omvendt vil utgangen av komparatoren gå til GND dersom inngangssignalet synker noen få millivolt under det integrerte signalet. Kretsen forsterker *differansen* mellom de to verdiene, og operasjonen til kretsen er tilnærmet lik en *differensiator*.

Poenget med kretsen er at AC-delen av inngangs-signalet $v_p(t)$,dempes av integratoren, slik at AC-komponenten inn på "+"-inngangen til komparatoren *alltid* er mindre enn på



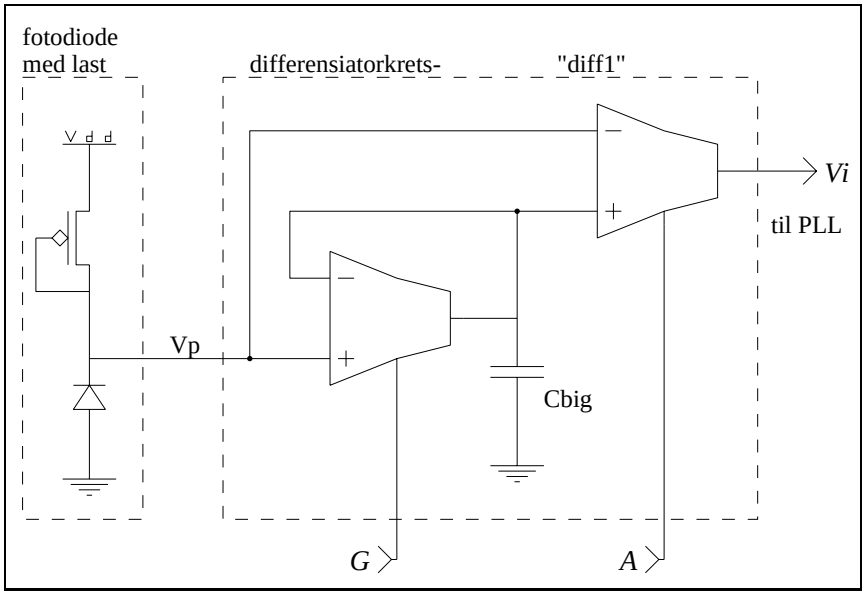
Figur 4.2: Respons til fotodiode for LED-pulser med 40% DutyCycle

"-"-inngangen. Dette betyr at utgangen av integratoren virker som en referanse til komparatoren. En referanse som omstiller seg etter størrelsen på DC-komponenten til $v_p(t)$. Med andre ord er denne kretsen uavhengig av DC-nivået til inngangs-signalet. Men det er verdt å merke seg at spenningen ut fra integratoren alltid er litt lavere (cirka 50mV dersom V_p ligger rundt 4V) enn V_p, på grunn av negativ offset i transkonduktansforsterkeren. Dette er imidlertid en *fordel*, fordi utgangen fra kretsen alltid vil ligge til *GND* dersom inngangsspenningen til komparatoren er negativ, $V_+ - V_- < 0$.

Når en positiv lyspuls treffer fotodioden, oppstår en negativ puls på V_p. Dersom denne er stor nok (absoluttverdi større enn 50mV), vil komparatorens utgang slå om fra GND til V_{dd}, fra logisk lavt nivå til logisk høyt nivå. Utgangen vil ligge høy inntil integrert spenning har nådd $V_p - V_{offset}$. Da vil den igjen falle til *GND*. Dette gir en puls ut, der pulsbredden er avhengig av integrasjons-konstanten. Denne settes med bias-spenningen til integratoren, kalt G.

Dessverre er det ikke bare lyspulser som bidrar til AC-komponenten i $v_p(t)$. Også *støy* gir et uønsket bidrag som kan gi problemer. En støyspiker inn på komparatoren gir akkurat samme resultat som en lysinitiert puls dersom den har tilstrekkelig amplitude. Det som kan gjøres for å minske kretsens følsomhet for støy, er å senke bias-spenningen til komparatoren. Denne spenningen benevnes A.

Dette har en dobbeltsidig effekt: for det første kreves det en større inngangs-spenning på komparatoren (større differanse mellom V_+ og V_-) før den slår om. Støyens amplitude vil



Figur 4.3: Fullstendig fotodetektor

som regel være mindre enn puls-amplituden fra fotodioden, og slik kan komparatoren terskles til bare å slå om for signal-amplituder såvidt over støyen. Er støyens gjennomsnittlige amplitude over utslaget til $v_p(t)$, kan ingenting av denne verden hjelpe denne kretsen.

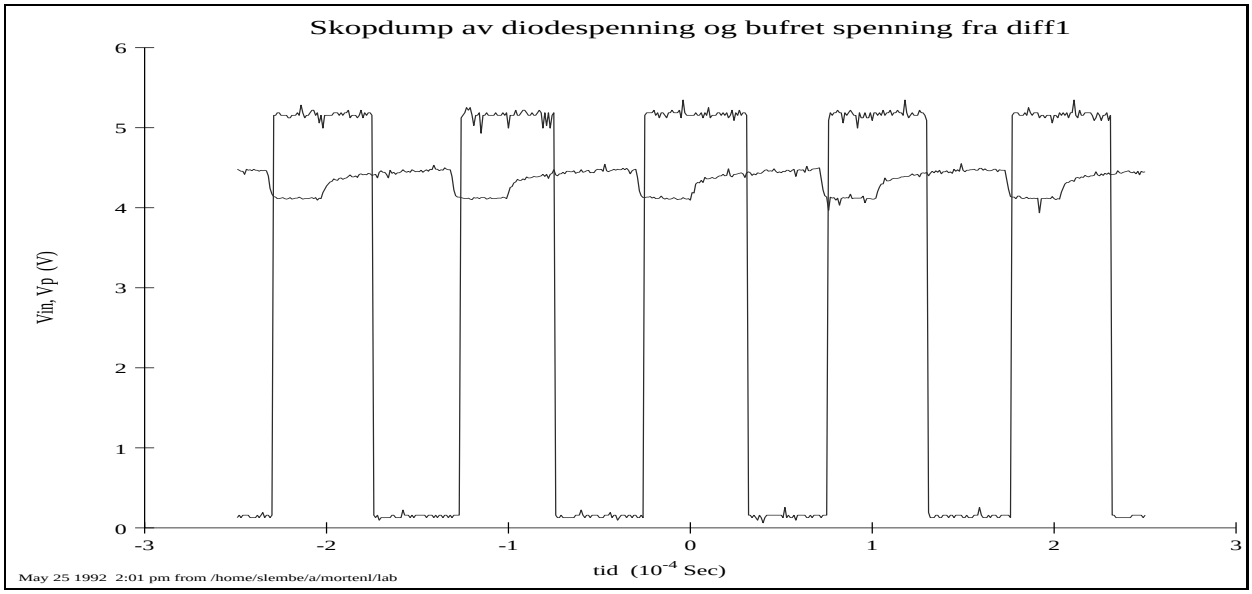
Den andre effekten av lavere verdi på A, er at komparatoren slår om langsommere fordi utgangs-strømmen senkes. Lav forsterkning i komparatoren kan bety at den ikke rekker å slå om for kortvarige støyspikere, med andre ord en slags lavpass-filtrering. Denne effekten er av mindre betydning enn den som er beskrevet over.

Praktiske målinger viser at kretsen fungerer som forutsatt dersom VCO og digitaldel ikke opererer samtidig med den. Resultat av måling *med* aktiv digitaldel og VCO, men med pulsfrekvens senket til 10kHz, er vist i figur 4.4. Ved en så lav frekvens er utslaget i $v_p(t)$ så stort (nesten 0.3V) at den internt genererte støyen ikke ødelegger kretsens funksjon.

4.3 Pulsdetektor

Pulsdetektoren reagerer bare på pulser med T_{high} større enn en viss verdi. Denne verdien er som før nevnt satt til en mye større verdi enn T_{high} for vanlige signaleringspulser, det vil si data modulert med 100kHz-firkantpuls, 50% duty-cycle, altså $T_H = T_L = 5 \cdot 10^{-6}$s.

Dersom en puls av tilstrekkelig varighet sendes inn i kretsen, går utgangen etter en viss tid høy. Pulsdetektoren er bygd opp av to elementer: en integrator og en komparator. Begge disse komponentene er i sin tur bygd opp av "wide-range" transkonduktans-



Figur 4.4: Relasjon mellom diodespenning og detektor-utgang

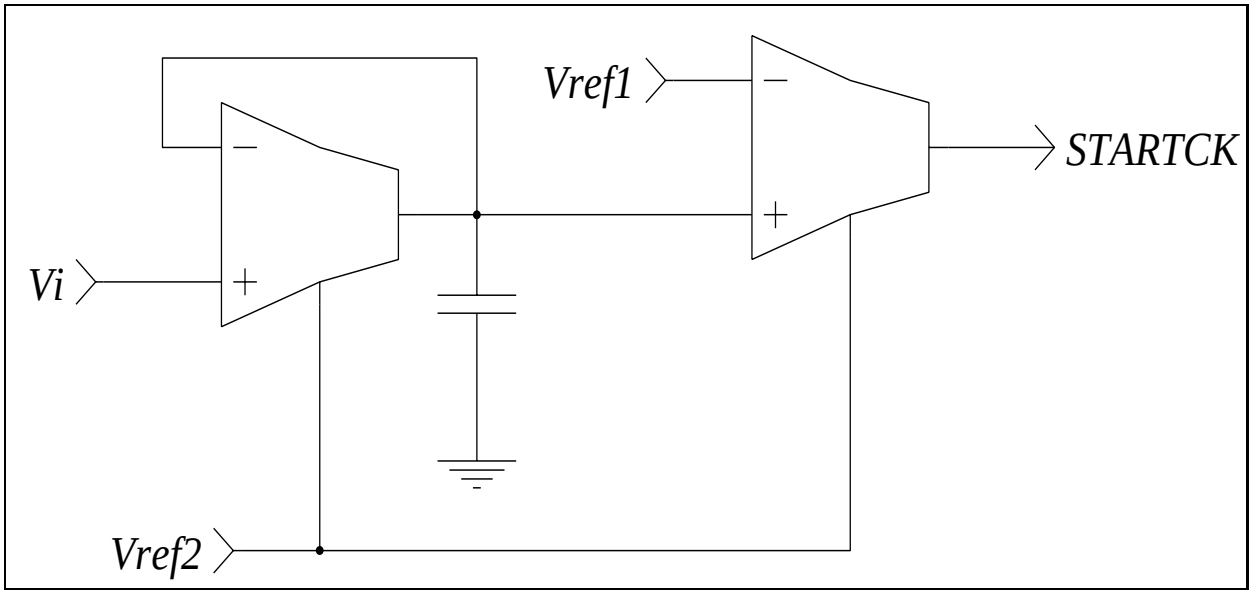
forsterkere. Referansen til komparatoren, V_{ref1}, kommer fra en spennings-referanse lagd med diodekoblede P- og N-transistorer, der referanse-spenningen er gitt av relativ konduktans i hver transistor. Ønsket spenning fåes ved individuell skalering. Kretsskjema for pulsdetektoren er vist i figur 4.5.

4.4 Delkretser i PLL

4.4.1 Fasedetektor - PD

Fasedetektoren er en enkel 8-transistor XOR-port, *ikke* bygd opp av transmisjonsporter. Som før forklart kan den sees på som en multiplikator med overdrevne innganger, og utgangsverdien beregnes som en *middelverdi* av en periode av utgangssignalet. Dette utgangssignalets periode er lik $T_d = \frac{T_i T_o}{T_i + T_o}$, der T_i og T_o er perioden til de to inngangssignalene. Dersom $\omega_o = \omega_i$ ($T_i = T_o$), blir derfor $T_d = \frac{1}{2}T_o$.

Dette er en meget enkel, men effektiv fasekomparator. Den kan brukes ved relativt høye frekvenser, og den er ikke utsatt for "cycle-slip" slik som sekvensielle fasedetektorer. Til gjengjeld har den et begrenset fase-operasjonsområde som går fra $-90°$ til $+90°$ ($-\frac{\pi}{2}$ radianer til $+\frac{\pi}{2}$ radianer). Dette betyr at systemet ikke vil være "låst" på referansen dersom den har mer enn $\pm 90°$ momentan faseendring, som for eksempel ved bifase-modulasjon (M1/M2-koding).



Figur 4.5: Kretsskjema for pulsdetektor

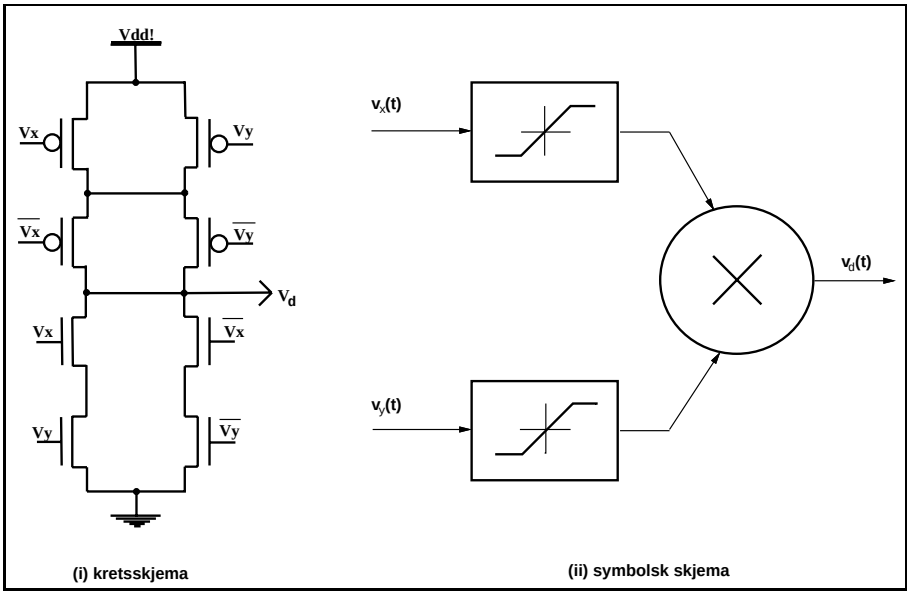
En annen bemerkning til XOR-porten, er at den kún er en *fase*-komparator, og ikke en *frekvens*-komparator. Dersom oscillatorens utgangsfrekvens, ω_o, er lik $n \cdot \omega_i$ eller $\frac{1}{n} \cdot \omega_i$, der n=partall, vil middelverdien av utgangssignalet fra XOR-porten være den samme som for $\omega_o = \omega_i$. Derfor er fasedetektoren det nest mest kritiske leddet i en PLL.

XOR-porten ble valgt som PD spesielt fordi den var enkel, men også på grunn av sløyfefilteret. Dersom jeg hadde valgt å bruke en "charge-pump"-teknikk, måtte jeg også ha brukt en sekvensiell PD. Men jeg vurderte det slik at jo flere deler det er i et puslespill, jo vanskeligere er det å sette det feilfritt sammen.

4.4.2 Sløyfefilter

Sløyfefilteret har som oppgave å gjenvinne DC-verdien, eller middelverdien, av utgangssignalet fra fasedetektoren. Dette betyr ikke at konstruksjonen er triviell. Parametere som steg- og impuls-repons, drift, offset, støyresistens og båndbredde er i *enkelte* tilfeller svært kritiske. Disse faktorene har jeg ikke gått spesielt inn for å optimalisere, fordi jeg anså andre komponenter for å ha større innvirkning på kretsens operasjon.

Det viser seg ved studier av litteratur, at til de fleste anvendelser er en vanlig integrator bygd opp av en operasjonsforsterker og noen få passive komponenter godt nok dersom inngangsfrekvensen er tilnærmet fast. Et enkelt RC-lavpassfilter kan også brukes. Men her må det tas med i beregningen hvor mye strøm fasedetektoren kan gi til filteret.



Figur 4.6: Kretsskjema for XOR-fasedetektor

Som før nevnt, har det systemet jeg skal lage, store marginer med hensyn på hastighet og synkronisering. Responstiden eller fasevridningen til filteret er derfor ikke kritisk. En "wide-range" transkonduktans-forsterker i integrator-kobling, bør være tilstrekkelig i denne applikasjonen. Dessverre er ikke impulsresponsen til integratoren symmetrisk, men individuell skalering av transistorene kan redusere dette problemet noe.

Idéelt sett skiller integratoren ut middelverdien av inngangssignalet. Men denne middelverdien beregnes over et intervall gitt av integrasjons-konstanten τ_c, som er gitt av verdien til transkonduktansen i forsterkeren og kapasitansen på utgangen. Integrator-kapasitansen (utgangs-kapasitansen) C_{ig} er en fast verdi, og derfor varieres integrasjons-konstanten bare med bias-spenningen til forsterkeren.

Integrasjons-konstanten settes som et kompromiss mellom responstid og dempning. Dersom τ er liten, blir responstiden liten, men dempningen av AC-komponenten reduseres. Dersom τ_c er stor, blir AC-komponenten i utgangsspenningen liten, men responstiden blir stor fordi forsterkeren bruker lang tid på å lade opp C_{ig}.

En liten τ_c betyr derfor at $v_o(t)$ bedre følger variasjoner i signalet fra fasekomparatoren, men det betyr også at $v_c(t)$ får større oversving og får et oscillatorisk innsvingningsforløp for stegvariasjoner i inngangssignalet. Etter at *systemets* egenrespons etter en tid har dødd ut, vil det likevel være en periodisk variasjon i $v_c(t)$. Det er fordi signalet fra fasekomparatoren er periodisk. Dette bidraget vil dempes i samsvar med integrasjons-konstanten. Er den for liten, kan $v_c(t)$ variere så mye at enkelte innkommende klokkepulser mistes. I verste fall

vil systemet ikke klare å låse seg til innkommende frekvens.

En stor τ_c betyr at systemet bruker lengre tid på å synkronisere seg inn på f_{in}. Er τ_c *for* stor, kan hurtige endringer i f_{in} føre til at systemet mister "lock". Dersom man antar slike gunstige betingelser som liten variasjon i f_i, og lang synkroniserings-sekvens, er det ikke noe i veien for å bruke en stor integrasjons-konstant. Faktisk kan det være gunstig med tanke på støy-egenskaper, fordi støy i fasekomparator-signalet dempes bedre.

For RZ-demodulasjon er dette også gunstig. Når det er mange "0"-symboler i dataene, vil det komme få synkroniseringspulser. Da vil systemet låse seg til oscillatorens frittløpende frekvens, f_{o0}. Signalet fra fasekomparatoren vil da ha en periode tilsvarende perioden til oscillatorsignalet. Denne perioden er tilnærmet det dobbelte av perioden til den spenningen fasekomparatoren gir ut dersom det kommer en strøm av synkroniseringspulser. AC-komponenten i fasekomparator-signalet vil dermed bli flyttet nedover i frekvens, og dette vil gi større variasjoner i integratorsignalet. En stor τ_c vil gjøre forskjellen mellom disse to tilstandene mindre, og dermed lettere beholde synkroniseringen.

Forspenningen til integratoren tilføres eksternt via en analog "input-pad" i min kretsløsning, slik at det er mulig å stille τ til en passende verdi. Ut i fra simuleringer er omtrentlige verdier $\omega_c = \frac{1}{\tau_c} = 6krad/s$ for $V_c = 0.75V$, og $\omega_c = 18.5krad/s$ for $V_c = 0.8V$. Kondensatoren C_{ig}, er på cirka 2.5pF. Poly1-poly2 kapasitansen er oppgitt til cirka $0.55fF/um^2$ av fabrikanten for den aktuelle teknologien.

4.4.3 Tilpasningsledd

Middelverdien av integratorspenningen skal under synkronisert tilstand ligge rundt $V_{e0} = 2.5V$. Dette er for høy verdi til å kunne brukes direkte som kontrollspenning, V_c, til VCO. Et tilpasningsledd er derfor påkrevet. Det består av et strømspeil (transistor Q1 og Q2), og et differensialpar (transistor Q3 og Q4). Fordi Q4 har "gate" koblet til utgangen, er utgangsspenningen V_c bare gitt av strømmen gjennom Q2 og Q4, og dimensjoneringen av de to transistorene. Strømmen gjennom Q2 og Q4 er avhengig av konduktansen i Q3 som varierer med inngangs-spenningen på "gate" til Q3, altså feilspenningen V_e.

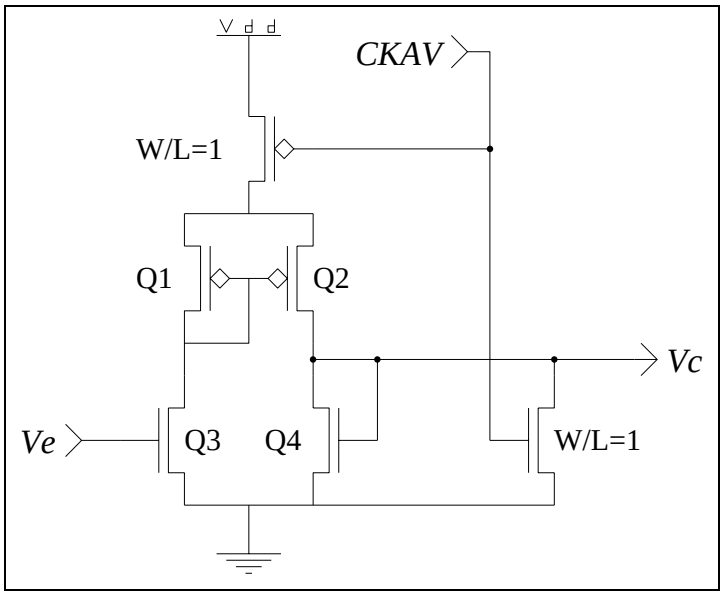
For å kunne stanse oscillasjonen i VCO, må $V_c = 0V$. Dette er gjort med en ekstra N-transistor som senker V_c til GND dersom kontrollsignalet #STARTCK(=#CKAV)=1. For ikke å trekke uforholdsmessig mye strøm gjennom Q2 og denne transistoren, er også en ekstra P-transistor koblet mellom kretsen og V_{dd}. Kretsen er derfor operativ bare dersom #STARTCK(=#CKAV)=0.

Skaleringen av transistorene er slik:

- W/L(Q1)=2
- W/L(Q2)=1/2

- W/L(Q3)=1/5
- W/L(Q4)=1

V_c vil rundt operasjonspunktet $V_{c0} = 1.25V$, være lik $V_e K_p$, der $K_p = 0.5$.



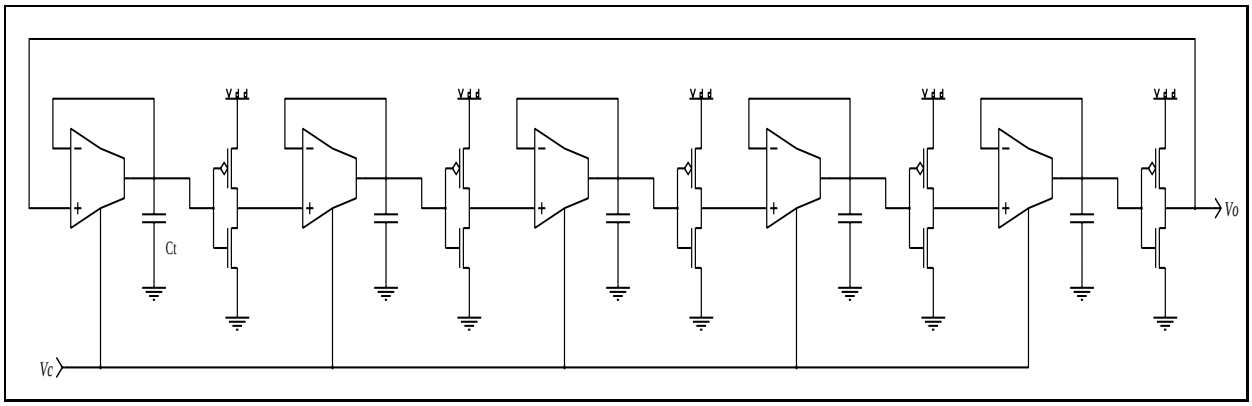
Figur 4.7: kretsskjema for tilpasning og oscillator-kontroll

Dessverre demper denne koblingen også AC-delen av V_c. Operasjonsområdet til V_c begrenses til mellom 0.9V og 1.6V av denne kretsen. Dette kalles "range-restriction", eller område-begrensning, og er ofte nødvendig av nettopp den årsak at VCO har et begrenset lineært område.

4.4.4 Spenningsstyrt oscillator - VCO

Ethvert PLL-system trenger en VCO. En enkel konstruksjon i CMOS er med ringkoblede invertere. Med et odde antall invertere koblet i ring (se figur 4.8), er man nesten garantert oscillasjon. Koblingen er inherent ustabil, eller teoretisk ustabil dersom sløyfe-forsterkningen > 1. Kort forklart betyr dette at dersom tilbakekoblings-signalet dempes så mye at spennings-svinget på inngangene til inverterne blir for lite til at de slår om, blir de liggende enten til GND eller V_{dd}, og oscillasjonen stanser. Den muligheten foreligger også at N- og P-transistoren i inverteren blir liggende i lineært område, og spenningen ved hvert trinn blir liggende rundt omslagspunktet til inverterne.

Dersom en spenningsstyrt forsinkelse settes inn mellom hvert ledd i koblingen, blir ringoscillatoren en VCO. Min løsning er å bruke integratorer som forsinkelsesledd. Integratorkapasitansen C_t er en poly1-poly2 kondensator på cirka 675fF. Den spenningsstyrte motstanden er transkonduktans-forsterkeren, fordi utgangs-strømmen er avhengig av biasspenningen, $V_b = V_c$.



Figur 4.8: Kretsskjema for spenningsstyrt oscillator, VCO

Én inverter er tilstrekkelig for å lage en oscillator, men spenningen fra siste integratortrinn vil variere svært lite rundt omslagspunktet til inverteren. Offset og støy kan da forårsake store problemer Disse faktorenes innvirkning reduseres til et minimum med en inverter mellom hvert ledd, fordi utgangs-spenningen fra hver integrator svinger over et stort område. Sagt på en annen måte, økes sløyfe-forsterkningen med flere invertere.

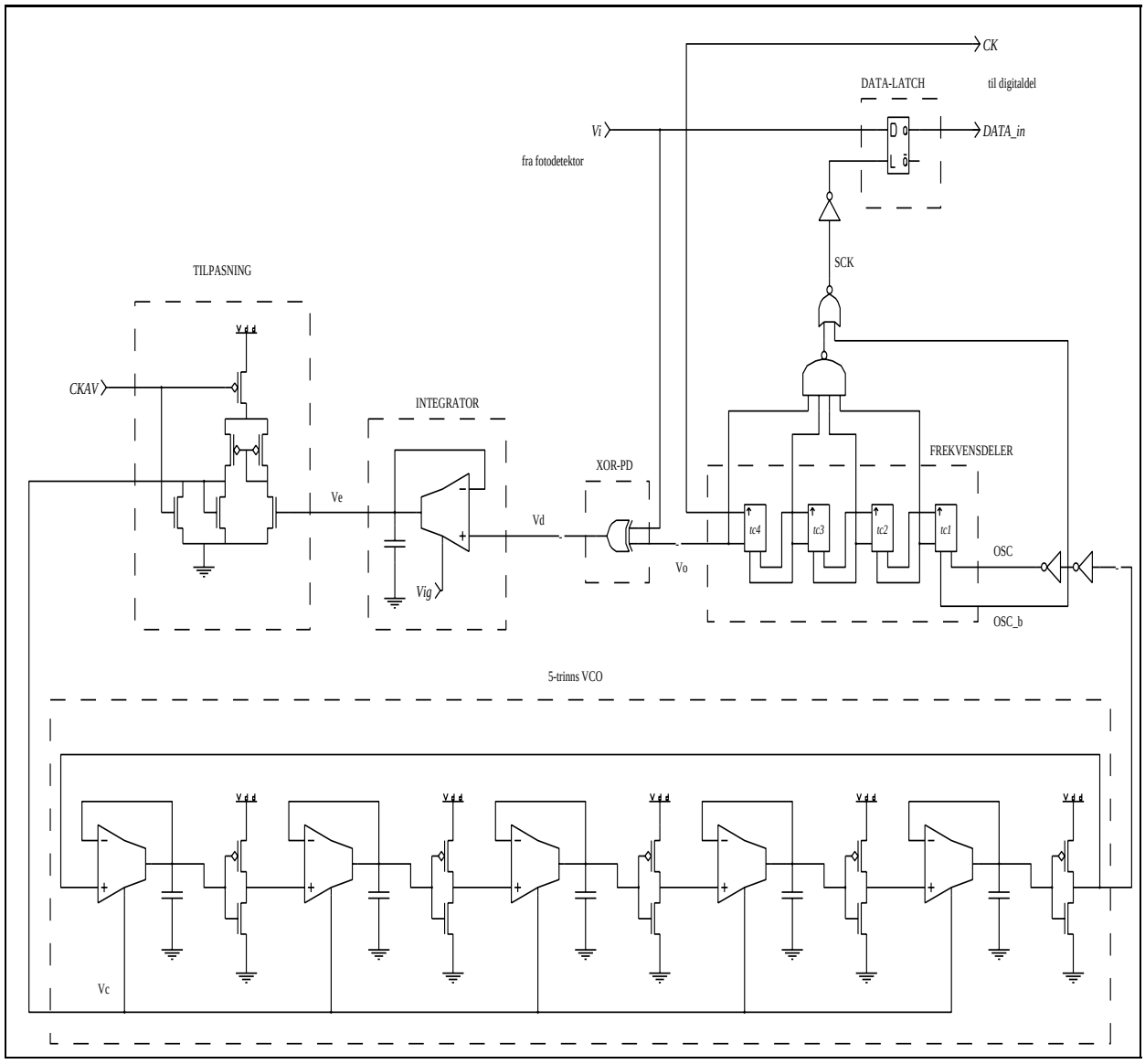
4.5 Sammenkobling av analoge delkretser

4.5.1 Sammenkobling av fotomottager, pulsdetektor og PLL

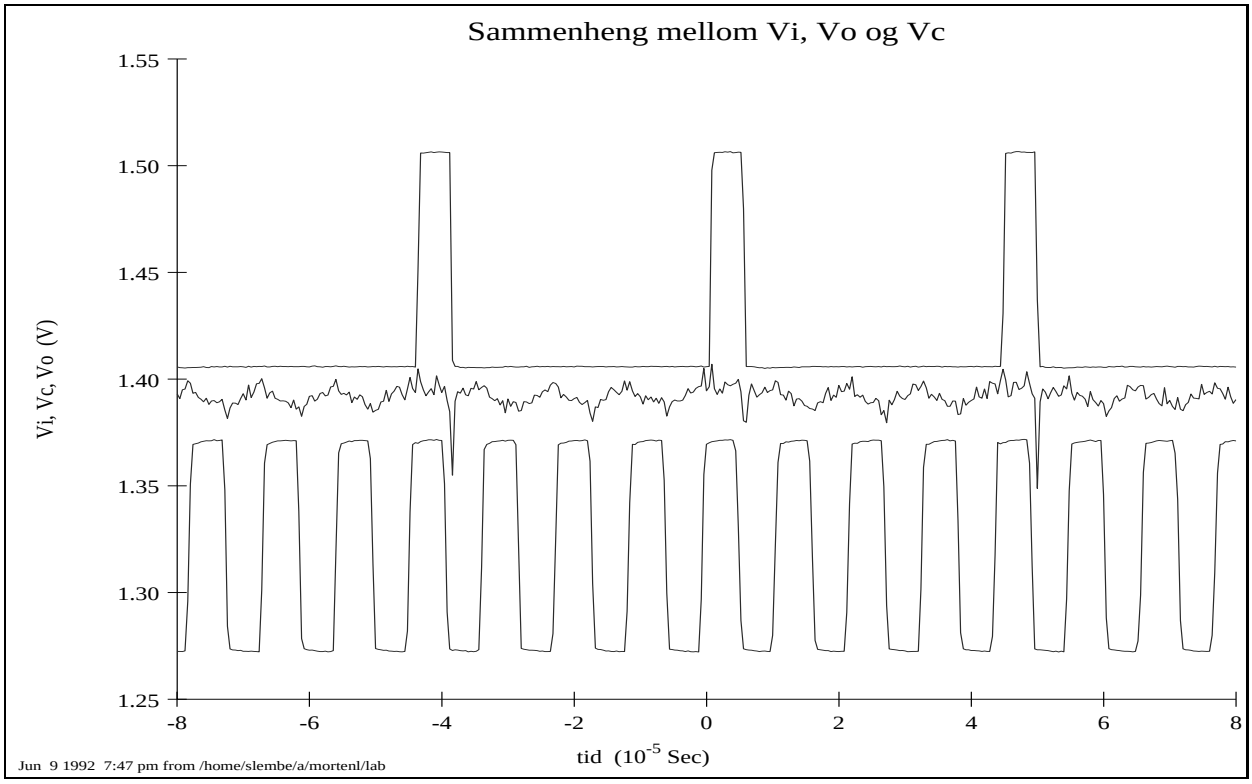
Utgangs-signalet fra fotomottager bufres med to invertere, og går henholdsvis til PLL og pulsdetektor. Bare selve fotodioden er skjermet mot elektrisk støy, mens de andre analoge delkretsene er skjermet for lys. Teknikkene for dette inngår i VLSI-implementasjonen, og vedlegg A kan leses om dette har interesse. Utgangen fra pulsdetektoren går til logikk som dekoder styresignalet #STARTCK (#CKAV) til VCO.

4.5.2 PLL-krets

Hele PLL-systemet er vist krets-skjematisk i figur 4.9. Signalet fra fotomottakeren bufres via en inverter for å få steile flanker, og går deretter inn på fasedetektoren sammen med referansesignalet; V_o. Differansesignalet V_d fra PD er inngangsverdien til integratoren, som midler $v_d(t)$ over tidskonstanten τ_c.



Figur 4.9: Kretsskjema for fullstendig PLL



Figur 4.10: PLL-funksjon illustrert ved inn- og ut-signal, og kontrollspenning

Det filtrerte signalet, $v_e(t)$, dempes til en passende verdi i "tpl" (tilpasningsledd). Utgangsverdien fra dette leddet, V_c, er kontrollspenning for VCO. I VCO'en omsettes V_c til en frekvens ω_{osc} lineært i operasjonsområdet. Denne frekvensen deles ned i frekvensdeleren med en faktor N=16, og signalet ut er frekvens- eller fase-referanse, f_o eller θ_o, til inngangssignalet. Operasjonen til PLL er vist i figur 4.10. Inngangs-frekvensen er cirka 90kHz, men 3 av 4 pulser er utelatt i RZ-signalet. Legg merke til at klokkesignalet generert av VCO er invertert i forhold til signalet vist her, slik at positive flanker i klokke treffer midt i hver innkommende puls.

Kapittel 5

Vurdering av analogdel

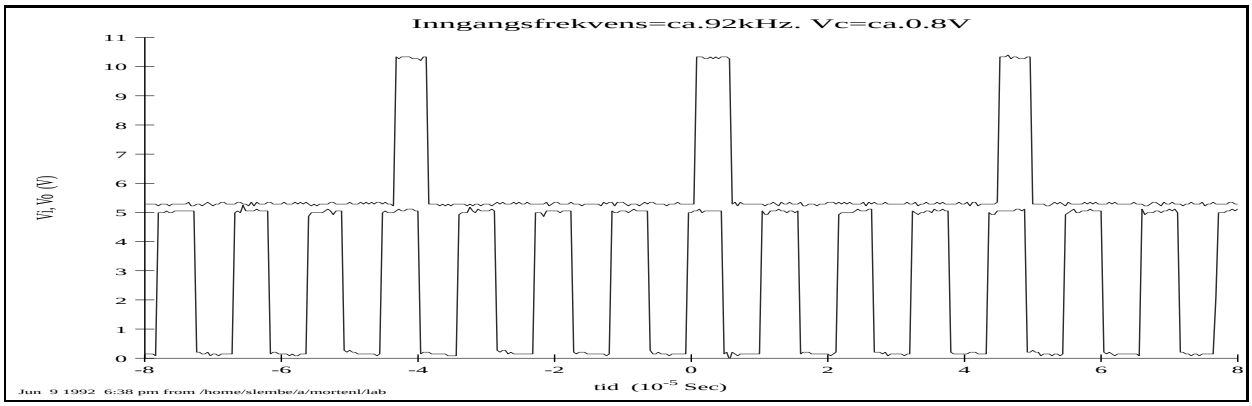
5.1 Evaluering av PLL

5.1.1 Synkronisering

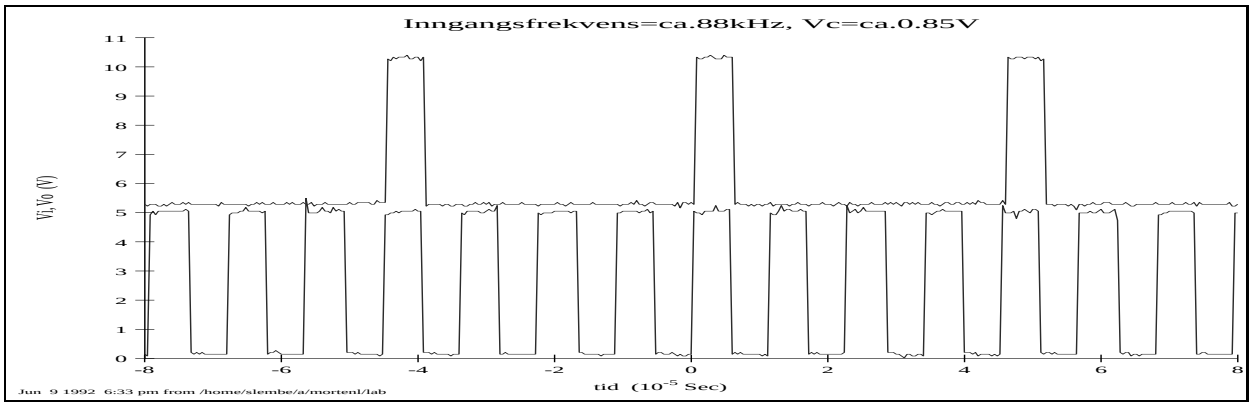
Jeg utforsket synkroniserings- og timing-marginen til systemet ved å sende pulser tilsvarende et repetérende, RZ-kodet bitmønster TP={"1000"}={**HLLL**} inn i PLL, med varierende klokkefrekvens. Dette er bare en primitiv test av systemet. For å finne tiden fra bortfall av synkroniseringspunkter i data til PLL driver ut av "lock", bør testen foregå rundt senterfrekvensen til VCO, og inngangs-signalet bør være et bitmønster med mange flere "0"-ere mellom hvert synkroniseringspunkt.

Eksempelvis kan man legge til et nytt bit uten synkroniseringspunkt, altså en "0", for hver testrunde til man ser at PLL taper synkronisasjonen. En annen variant som har spesiell verdi for den protokollen jeg har fastlagt, er å sende et bitmønster som innledes med 16 synkroniseringsbit, og som etterfølges av 16 bit, med synkroniseringspunkt i annenhvert bit. Deretter sendes 63 bit uten synkroniserinspunkter, med et synkroniseringspunkt helt til sist i testønsteret. Dekodede bit fra PLL sammenlignes til slutt med testmønsteret for å finne eventuelle bitfeil. Dette gjøres med flere repetisjoner av testmønsteret.

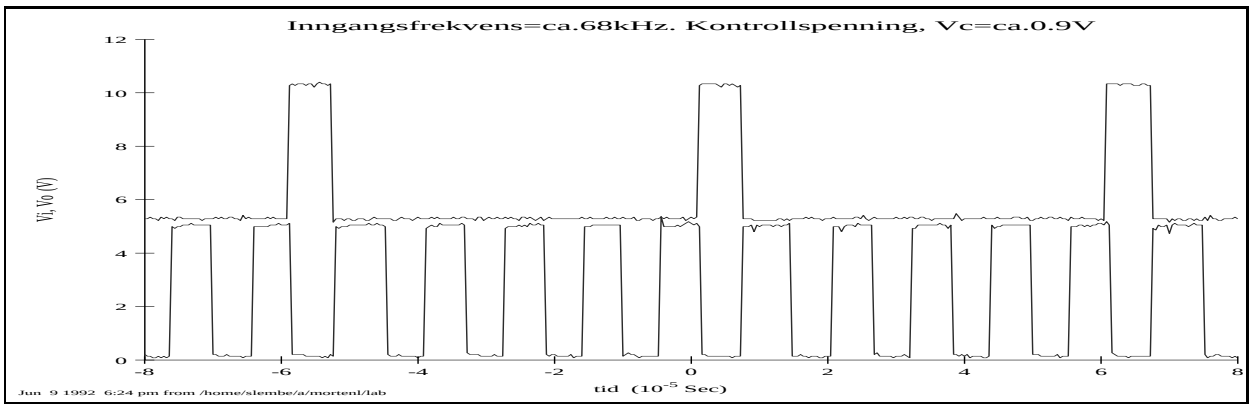
Dersom denne testen viser at bitfeil bare opptrer i synkroniseringsfeltet, er ingen ytterligere testing nødvendig for å bestemme timing-margin ved mottager-systemets senterfrekvens, forutsatt bruk av den gitte protokoll. Men opptrer bitfeil også utenfor dette feltet, må testen kjøres svært mange ganger for å opparbeide statistikk. Er sannsynligheten for feil (i [feil/bit]) fordelt noenlunde likt i over hele testmønsteret, er det usannsynlig at synkroniserings-marginen er årsaken til disse feilene. Dersom midlere feilrate er over en viss verdi, for eksempel mer enn 1 bitfeil per 2 rammer, må hele kretsen revurderes. Viser det seg derimot at sannsynligheten for feil øker med antall mottatte bit uten synkroniseringspunkter, mot slutten av en ramme, er timing-marginen ganske sikkert for liten. En endring av komponentverdiene i PLL kan være nok til å korrigere dette.



Figur 5.1: Synkronisering ved øvre grensefrekvens



Figur 5.2: Synkronisering rundt senterfrekvensen



Figur 5.3: Synkronisering ved nedre grensefrekvens

Målinger indikerer at modulasjonsområdet til PLL er asymmetrisk om senterfrekvensen, ω_{o0}. For den aktuelle kretsen som disse testene ble foretatt på, fikk jeg at øvre grensefrekvens er omtrent 95kHz. Senterfrekvensen ligger rundt 89kHz, og nedre grensefrekvens er omtrent 65kHz. Dette betyr at klokkefrekvensen kan gå 25kHz *under* senterfrekvensen, men bare 5kHz over. Dette skyldes dels den ulineære karakteristikken til tilpasnings-leddet, men spesielt begrensningen i modulasjonsområdet til VCO.

Figur 5.1 viser at testmønsteret gir god nok synkronisering til at mottagning kan skje ved 92kHz. Kontrollspenningen som er omtalt i figuren, er *ikke* styrespenningen til VCO, V_c, men integratorens bias-spenning V_{ig}. Med en klokkefrekvens i sender rundt senterfrekvensen, ser vi av figur 5.2 at timing-marginen er god. De negative transisjonene i invertert, lokal klokke er plassert nesten midt i hvert synkroniseringsbit. Men testing ved nedre grensefrekvens (figur 5.3) avslører at mottager klokker inn et ekstra bit, den taper altså synkronisering. Denne "misforståelsen" kalles et "cycle slip" fordi PLL introduserer en ekstra puls, eller en ekstra sykel, i forhold til referansen. Egentlig betyr "cycle slip" at en eller flere perioder av referansen mistes i mottageren, men problemet er som vist her tosidig. Legg merke til at PLL forsøker å senke ω_o til minimum etter hver innkommende puls.

Hvorfor fikk vi "cycle slip" i testen vist i fig 5.3, men ikke i figur 5.1? Et regnestykke basert på resultatene gir en pekepinn:

$$\text{Periode til senterfrekvens=90kHz: } T_{o0} = \frac{1}{9*10^4 s^{-1}} = 1.1 * 10^{-5}$$
$$\text{Periode til øvre grensefrekvens=95kHz: } T_{o,min} = \frac{1}{9.5*10^4 s^{-1}} = 1.05 * 10^{-5}$$
$$\text{Periode til nedre grensefrekvens=65kHz: } T_{o,max} = \frac{1}{6.5*10^4 s^{-1}} = 1.54 * 10^{-5}$$

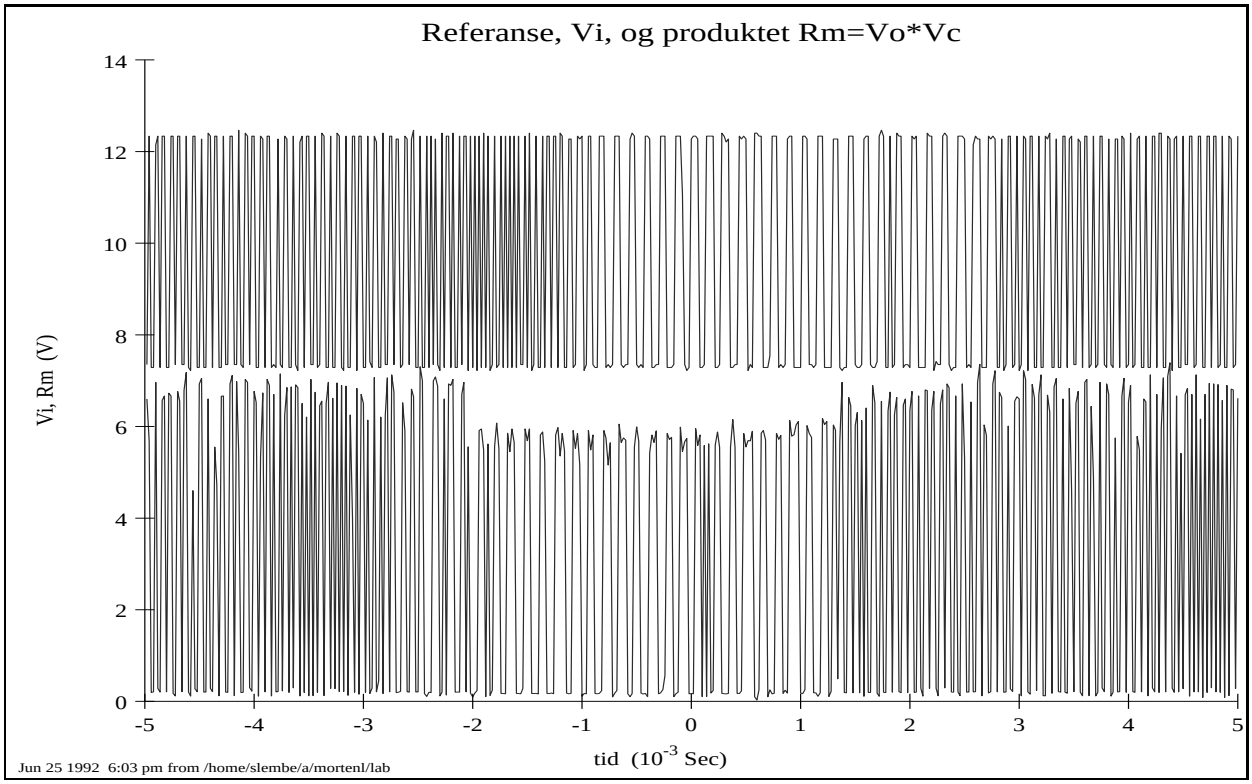
Tiden det tar å sende 3 bit blir:

$$T_{send,0} = 3T_{o0} = 3.3 \cdot 10^{e-5}$$
$$\mathrm{T}_{send,min} = 3T_{o,min} = 3.15 \cdot 10^{e-5}$$
$$\mathrm{T}_{send,max} = 3T_{o,max} = 4.62 \cdot 10^{e-5}$$

Og forholdet mellom sendetid ved senterfrekvens og henhodsvis øvre og nedre grensefrekvens:

$$\text{1) Periodeforhold, } \frac{T_{send,min}}{T_{o0}} = \underline{2.86}$$
$$\text{2) Periodeforhold, } \frac{T_{send,max}}{T_{o0}} = \underline{4.20}$$

Her kommer det forholdet fram, at med tre etterfølgende bit uten referansepunkter,vil tiden for tre bitperioder i sender representere et annet antall perioder i mottager ved dens senterfrekvens. Dersom denne differansen er på mer enn én hel bitperiode, T_{o0}, er timing-marginen over kritisk grense. Da vil synkroniseringen avhenge av systemets tidsrespons alene, eller båndbredden til PLL. Forholdet i 1) er ikke kritisk. Men dersom dette forholdet var ≤ 2, ville antakelig "cycle slip" oppstå.



Figur 5.4: Frekvensmodulering av sender, og respons i mottager

I 2) er forholdet helt klart kritisk, fordi marginen er overskredet med 0.2 perioder. Et "cycle-slip" oppstår, og en kodingstype må derfor velges som gir et synkroniseringspunkt i hvert bit, dersom hele "lock in"-området til mottager skal kunne utnyttes. Det andre alternativet er å redusere båndbredden til PLL. Nettopp på grunn av forholdet beskrevet over, er ofte båndbredden til PLL-demodulatorer i digital kommunikasjon satt svært liten. Som tidligere forklart er ikke dette bare gunstig, derfor må de vanlige kompromissene gjøres.

5.1.2 Respons i tidsplanet og "lock in"

Responsen til PLL henger nøye sammen med responsen til filteret. En test av systemet med FM-modulert, kontinuerlig referanse (V_i), gir informasjon om tidsresponsen til systemet. Tidsresponsen er i sin tur et mål på "lock in"-tid og båndbredde til systemet. Figur 5.4 viser demodulasjon av FM-modulert inngangs-signal. Modulasjonsområdet er 10kHz, $f_{i,min} = 80kHz$ og $f_{i,max} = 90kHz$, og modulasjonsfrekvensen er 1kHz firkantpuls. Det nederste signalet er produktet mellom kontrollspenning og utgangs-signal, $V_c V_o$. Dette produktet er vist bare for å få fram sammenhengen mellom kontrollspenning og utgangsfrekvens.

Her kan vi se at PLL er i stand til å følge inngangsfrekvensen, men at responsen er dempet i forhold til V_i. Jeg har vanskelig for å forklare at V_o ser ut til å endres *før* V_i.

Antakelig er effektiv båndbredde lavere enn beregnet av forskjellige årsaker, uten at det gjør meg noe klokere. En faktor som ikke er tatt med i selve beregningen, er den begrensede båndbredden til VCO. Den idéelle modellen som er brukt som utgangspunkt, forutsetter at VCO kan endre verdi på utgangsfrekvensen momentant. Dette er selvsagt ikke tilfelle. For å kunne si noe eksakt om VCO-båndbredde, må målinger foretas på en frittstående VCO.

5.1.3 Modulasjonsområde

Det lineære området til VCO er begrenset til en kontrollspenning på cirka $V_{c1} = 1.0V$ nedad, og $V_{c2} = 1.5V$ oppad. VCO-konstanten er i operasjonsområdet, altså ved senterfrekvensen ω_{o0}, beregnet til omtrent 80kHz/V=500krad/s/V. Med disse verdiene fåes at:

$$\omega_{o1} = K_o V_{c1} = 500krad/s/V * 1V = \underline{500krad/s = 80kHz}$$
$$\omega_{o2} = K_o V_{c2} = 500krad/s/V * 1.5V = \underline{750krad/s = 120kHz}$$

Modulasjonsområdet er gitt av differansen mellom høyeste og laveste frekvens PLL kan gi ut, eller også det største området V_c kan sveipe over, benevnt med $\delta V_{c,max}$, forutsatt at kontrollspenningen hele tiden er i det lineære området til VCO. Modulasjonsområdet kalles også "lock in - range", ω_L, fordi det angir over hvilket område PLL kan oppnå "lock". Beregnet modulasjonsområde blir:

$$\omega_L = \delta V_{c,max} K_o = \omega_{o2} - \omega_{o1} = \underline{250krad/s = 40kHz}$$

Modulasjonsområdet er symmetrisk om senterfrekvensen=100kHz i teorien. Men målinger viser at ω_{o1} = cirka 65kHz, og ω_{o2} = cirka 95kHz, for den kretsen som lå nærmest de oppsatte spesifikasjoner. Altså et praktisk modulasjonsområde mellom 25kHz og 30kHz. Årsaken til denne reduksjonen er mest sannsynlig en for stor reduksjon av modulasjonsområdet til VCO på grunn av frekvensdeleren.

5.1.4 Båndbredde

Dersom sløyfe-filteret har en endelig dempning forskjellig fra null ved frekvenser godt over knekkfrekvensen til filteret, $\omega \ll \omega_c$, er 3dB-båndbredden til PLL gitt av ligning 3.3 fra kapittel 3. Konstanten K_h er den endelige, eller den maksimale dempningen til filteret. Integratoren i mitt system har ikke endelig dempning. Derfor må dempningen rundt den *dobbelte* verdien av senterfrekvensen benyttes som en erstatning for K_h fra ligning 3.3, fordi dette er frekvensen til $v_d(t)$. Beregnet knekkfrekvens (3dB dempning) for sløyfefilteret, er 3kHz=18.84krad/s for $V_{ig} = 0.8V$ som forspenning til integratoren. Dette gir en teoretisk båndbredde lik:

$$\omega_{3dB} = K_d |F(j2\omega_{o0})| K_p K_o = 1.6V/rad \cdot |(1 + (\tfrac{1.256 \cdot 10^5}{1.884 \cdot 10^4})^2)^{-1/2}| \cdot 0.5 \cdot 500krad/s =$$
$$1.6V/rad \cdot 0.015 \cdot 0.5 \cdot \cdot 10^5 rad/s = \underline{6 \cdot 10^3 s^{-1} = 6kHz}$$

PLL-båndbredden er vanskelig å måle med enkle midler, så dette tallet har jeg ikke vært i stand til å verifisere. Helt sikkert er det at det i praksis er langt lavere. Derimot har jeg skrudd på forspenningen til integratoren, V_{ig}, og sett at økende forspenning = høyere knekkfrekvens, gjør PLL bedre i stand til å følge variasjoner i ω_i. Men dette gjelder bare inntil variasjonen i V_c blir så store at PLL faller ut av "lock".

Uten referanse, $V_i = 0$, går kontrollspenningen så lavt ved $V_{ig} \geq 1.0V$, at VCO ikke lenger greier å produsere noe utgangssignal. V_o blir liggende til jord, og $\omega_o = 0$. Med referanse, svinger V_c over hele det mulige området, også utenfor lineært område til VCO. resultatet blir et frekvensmodulert utgangs-signal som moduleres av V_i. Ingen av disse to situasjonene er ønskelige selvsagt, men mest alvorlig er det at uten referanse stanser PLL nesten helt opp.

Resultatene indikerer at båndbredden bør være *stor* med et referansesignal til PLL, og *liten* uten referanse. Blant annet kunne jeg skru forspenningen til integratoren helt av, $V_{ig} = 0$, uten annet enn små endringer i ω_o, når PLL ikke hadde noen innkommende referanse. Spenningen på integrator-kondensatoren, tilsvarende V_c, ble liggende der når forspenningen ble skrudd av. Kontrollspenningen holdt seg derfor nesten konstant, og tilsvarende ble også utgangsfrekvensen, ω_{o0}, svært jevn. Men uten forspenning til integratoren kunne heller ikke PLL fungere når den igjen fikk et referansesignal. Båndbredden ble tilnærmet lik null, og utgangsfrekvensen holdt seg konstant uavhengig av endringer i ω_i.

Ideéllt sett bør forspenningen være rundt 0.8V når referansen er lik null; $V_i = 0$, og mellom 0.9V og 1.0V med referanse. Men dette vet jeg ikke helt hvordan skal gjøres i praksis. Derfor er det nærliggende å sette begrensninger på variasjonen i ω_i, $\Delta\omega_i$, og heller sette båndbredden til PLL relativt lav med for eksempel $V_{ig} = 0.85V$. Men dersom ω_{o0} avviker mye fra krets til krets, er dette ugunstig fordi synkronisering tar lengre tid.

5.1.5 Statisk fasefeil

Den statiske fasefeilen fører til at de positive flankene til klokka generert i PLL ikke ender midt i innkommende pulser, eller midt i biverdien til RZ-kodet signal. Dette er selvsagt alvorlig dersom flankeavstanden, eller timing-marginen, blir liten. I mitt system er statisk fasefeil vanskelig å redusere fordi jeg ikke har DC-forsterkning i integratoren, men tvert i mot en liten dempning. Mottiltak kan være å lage K_d eller K_o større. Det er vanskelig å lage PD-konstanten større, derfor må heller VCO konstrueres annerledes. For eksempel vil K_o økes dersom neddelings-faktoren i frekvensdeleren gjøres mindre.

Dempningen i integratoren introduseres av den begrensede forsterkningentil en transkonduktans-forsterker. Typisk vil forsterkningen være maksimalt 1000 til 2000. Jeg gjør derfor en forsiktig tilnærming ved å sette differensial-forsterkningen i mine forsterkerkretser lik

A_v=250. Integratoren reduseres til en spenningsfølger ved DC-analyse. Utgangsspenningen fra følgeren dempes med faktoren $F(0)$.

$$\text{DC-forsterkning: } F(0) = \frac{A_v}{1+A_v} = \frac{250}{1+250} = \underline{0.96}$$

Dette medfører at sløyfeforsterkningen reduseres, og dermed blir statisk fasefeil større. Men denne verdien er så liten at andre faktorer, som offset i differansespenningen fra PD og i kontrollspenningen, gir et større bidrag til den totale, statiske fasefeilen. Som en tilnærming kan variasjonen i "duty-cycle" på V_i være inntil $\pm 10\%$. Dette vil gi en offset rundt operasjonspunktet V_{d0} på $V_{do} = \pm 0,25V$. Beregnet fasefeil blir ifølge ligning 3.7 med *positiv* offset:

$$\theta_{e0} = \frac{-V_{do}}{K_d \cdot (1+F(0))} = \frac{-0.25V}{1.6V/rad \cdot (1+0.96)} = \underline{-0.0797rad} = \underline{-4.58°}$$

Denne verdien er ikke avskrekkende stor; effektivt faseområde er redusert med 4.58°, eller rundt 2.5%. Operasjonsområdet til V_d er derimot redusert med $\frac{V_{do}}{V_{dm}} = \frac{0.25V}{5V} = 0.05$, altså 5%.

5.2 Evaluering av analogdel

5.2.1 Evaluering av fotomottaker

Fotomottakeren er den mest kritiske komponenten i hele analogdelen på grunn av de lave signalnivåene. Dette gjør den følsom for støy og temperaturdrift. Med den eksisterende konstruksjonen er det mulig å benytte den opp til 25kHz, men dette utnytter langt i fra dens fulle potensiale.

Det er uklart om det er selve fotodioden eller diff1-kretsen som fanger opp mest støy. Sannsynligvis er det fotodioden, fordi støyspikere i V_p forårsaker pulser med fullt sving på utgangen av komparatoren i diff1-kretsen. Støyspikere på V_{dd}-linjene til diff1-kretsen vil bare gi tilsvarende støyspikere, men kanskje med noe mindre størrelse på komparator-utgangen.

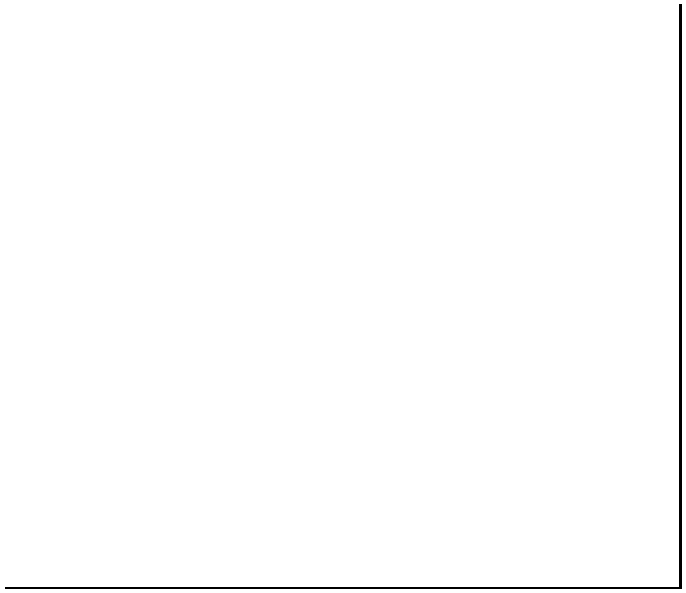
5.2.2 Fasedetektor

XOR-porten var et godt valg som PD. Den store båndbredden gjorde det mulig å utnytte nesten hele det teoretiske faseområdet. Flankene til V_o krøp nesten helt inntil positiv flanke på V_i for $f_{i,min}$, og nesten helt inntil negativ flanke for $f_{i,max}$.

Men et minus er at V_d ikke holdes konstant under stabil, eller låst tilstand, men er et periodisk signal. Dette medførte den relativt store rippelen i V_c. Jeg ble her reddet av at jeg for det første kunne justere filterets avkuttingsfrekvens, og dermed dempning ved ω_{o0}, med V_{ig}. For det andre reduserer lavpass-karakteristikken til VCO, innvirkningen av denne rippelen på ω_o.

Figur 5.5: Korreksjon av VCO-senterfrekvens med ekstra oscillator-ledd

Dette vil bety enda et avvik fra målsetningen om ingen eksterne signaler og komponenter. Men det kan også være den eneste praktiske metoden for å korrigere senterfrekvensen til VCO. Muligens vil dette problemet reduseres ettersom bedre CMOS-prossesser tas i bruk, og prossessvariasjoner reduseres. Men per i dag er dette kanskje en jobb for et analogt, nevralt nettverk med opplæring på brikken slik at korreksjonenkan foretas automatisk?



Figur 5.6: Simulering av VCO

Simuleringen ble utført med en 5-trinns ringoscillator-VCO av samme type som jeg benyttet i kretsen, men med $C_t = 80fF$ i integratorleddene. Derfor er senterfrekvens langt høyere her enn for det realiserte system. Tykk linje representerer senterfrekvens, f_{o0}, som funksjon av kontrollspenning, V_c. Det tilnærmet lineære område er avmerket fra $V_{c,min}$ til $V_{c,max}$. Karakteristikken er også lineær et stykke over $V_{c,max}$, men tilpasnings-kretsen kan ikke gi ut en høyere spenning enn cirka 1.6V, på grunn av områdebegrensningen til spennings-tilpasningen. Karakteristikken for realisert VCO blir flyttet omtrent én orden ned i frekvens, fordi C_t ble lagd 8-9 ganger større, og bias-transistorene i transkonduktans-forsterkerne ble skalert ned..

Modulasjonsområde: Av figur 5.6 kan den konklusjonen trekkes at modulasjonsområdet er stort. Simuleringene viser ingen nedre grense for f_o, og oppad er det først og fremst slew-rate til integratorene som utgjør begrensningen.

Simuleringene tilsier en $f_{o,max} = 225MHz$, men dersom integratorspenningen i hvert ledd bare varierer med noen få millivolt rundt svitsjepunktet til inverterne, vil støy (i integratorspenningen) enten motvirke oscillasjonen eller generere tilfeldige frekvenser. Antakeligvis det siste. Støyen vil ganske sikkert være der, derfor vil et nøkternt estimat av $f_{o,max}$ være cirka 10MHz.

Båndbredde: I praksis har VCO en begrensning i hvor hurtig utgangsfrekvensen kan moduleres. Systemfunksjonen er derfor frekvensavhengig, og kan tilnærmes med:

$$\textit{Tilnærmet frekvensrespons for VCO: } K_o(s) = K_{o0} \cdot \frac{\omega_1}{s+\omega_1}$$

Eksakt verdi for ω_1 i VCO har jeg liten formening om, men simuleringer tilsier at den ligger over 50krad/s.

Fasedrift og frekvensdrift: Ut i fra målinger er frekvensdrift maksimalt 300Hz ved 90kHz senterfrekvens og romtemperatur. Altså omtrent 0.33 prosent, eller 0.0033Hz/Hz. Antakelig er gjennomsnittsverdien mye lavere, men dette kan jeg ikke verifisere. Båndbredde til PLL på 6kHz bør være tilstrekkelig til å korrigere for dette frekvensavviket.

Som tidligere nevnt er senterfrekvensen tildels svært forskjellig fra krets til krets. Hver brikke ble testet under tilnærmet like forhold, og temperaturdrift kan derfor ikke være årsaken til de store avvikene. Det er også en sammenheng mellom integratorbias, V_{ig}, og senterfrekvens. Når V_{ig} kommer over terskelspenningen til bias-transistoren, faller senterfrekvensen. Dette er ganske sikkert fordi endringene i kontrollspenningen over tid, $\frac{\Delta v_c(t)}{\Delta t}$, blir for store til at VCO greier å følge med.

Jeg noterte også at DutyCycle på V_o ble forskjellig fra 50% rett før båndbredden til VCO ble overskredet, altså et asymmetrisk utgangs-signal. DutyCycle ble opptil 65%, og dette forholdet tror jeg skyldes asymmetriske stige- og fall-tider for integratorene i VCO. Sammenhengen mellom senterfrekvens og integratorbias er fremstilt grafisk i figur 5.7. Disse måleresultatene ble innhentet manuelt, og verdiene er derfor ikke helt eksakte. Figur 5.7 viser resultatene for de kretsene som viste enten størst avvik, eller minst avvik. Tykk linje viser gjennomsnittet for alle kretsene.

Kostnad: Arealet som VCO opptar på brikken representerer her kostnadsfaktoren. Kostnaden ved denne type VCO avhenger lineært av antall ledd, og størrelsen av hvert ledd. Dersom man velger å bruke få ledd, men istedet lage kondensatoren større, tjener man noe inn på arealet. Dessuten blir det lettere å beregne senterfrekvensen nøyaktig. Men til gjengjeld blir kretsen mer utsatt for støy, og utgangsspenningen blir mer asymmetrisk enn med mange ledd.

Figur 5.7: Relasjon mellom integratorbias og utgangsfrekvens

Kapittel 6

Digitaldel

Enkelte notasjoner i dette kapittelet behøver omtale: Signaler er skrives som #[signalnavn]. Bitfelter fra datarammer er som før omtalt i uthevet skrift, slik som for eksempel *SYNC*. Bits som utgjør hele ord eller deler av dem, skrives for eksempel som {b0,b1,..,b15}.

All skjemategning og digital simulering ble utført med programmet DigLOG [7]. Et relativt ustødig verktøy som kjempet i mot mine ønsker hele veien.

6.1 Overblikk

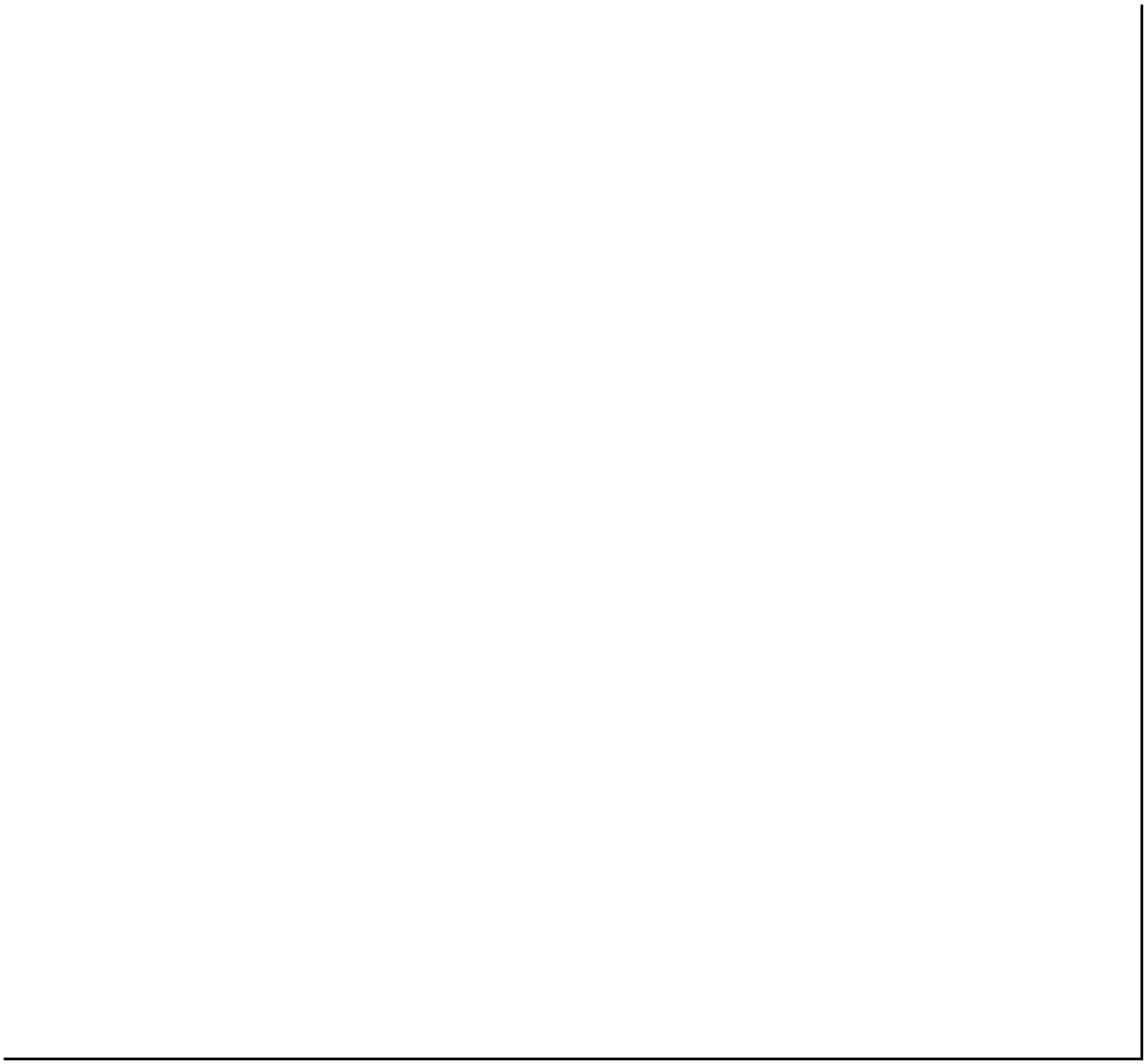
Digitaldelen består av 6 hovedblokker som vist i figur 6.1. Disse blokkene er:

1. Et 16-bits skiftregister
2. En 6-bits synkron teller, **T1**
3. En registerblokk
4. Sekvensiell kontroll-logikk
5. En timer hovedsakelig bygd rundt en rippelteller, **T2**
6. Et 16-bits tilbakekoblet skiftregister for sjekksum-generering

Skiftregisteret er 16 bit langt, og laget med seriekoblede latcher. Som systemblokk er det benevnt "16bsr".

Telleren **T1** har til oppgave å synkronisere rammelesingen. Operasjonen er synkron, og lengden er 6 bit. Med andre ord kan den telle fra 0 til 65 binært, som er bitnummereringen i en ramme. En positiv puls, eller et synkroniseringspunkt, genereres med et intervall på 16 klokkepulser med dekoder-logikk. Telleren kalles "6bsynkrt" i systemet.

Registerne inneholder de aktuelle bit fra hvert felt i rammen. Nye verdier lastes inn med synkroniserings-signaler fra telleren **T1**. Registernes navn er "AR", "BR", "CR" og "DR".



Figur 6.1: Blokkskjema over digitaldelen

Kontroll-logikken styrer både rammelesing, kommando-utførelse og sjekksum-generering. Den er laget som en konvensjonell tilstandsmaskin. Denne blokka kalles "skntrl".

Timeren, eller teller **T2**, brukes for å sette en utgang lav for et tidsrom gitt av verdien til rammefeltet *TVAL*. Den er laget som en rippel-teller, altså asynkron operasjon, og lengden er 20 bit. De fire mest signifikante bit av disse, inngår i en prescaler. Prescaleren sammenligner de 4 bitene med gitt timerverdi, og setter et signal høyt dersom verdiene stemmer overens. Systemblokk-benevningen er "timer".

Sjekksum-generatoren er et såkalt "LFSR", det står for LinearFeedbackShiftRegister. Dette er det samme skiftregisteret som brukes til å skifte rammebits fra serieform til parallellform med hensyn på oppbygning. I tillegg er enkelte bits tatt ut og XOR'et med hverandre, for deretter å tilbakekobles til en XOR-port der rammebits er input sammen med tilbakekoblet verdi. Utgangen fra denne XOR-porten er input til skiftregisteret igjen. System-navnet er "SAR", men i flytskjemaet kalles den SR (SjekksumRegister).

Sammenkoblingen av blokkene er såvidt skissert i figur 6.1. Tykk linje mellom blokker representerer data-linjene fra skiftregisteret. Blokker med kombinatorisk logikk som utfører dekoding og sammenligning mellom bitverdier fra forskjellige blokker, er ikke tatt med. Funksjonen til digitaldelen er fremstilt i et flytskjema i figur 6.3.

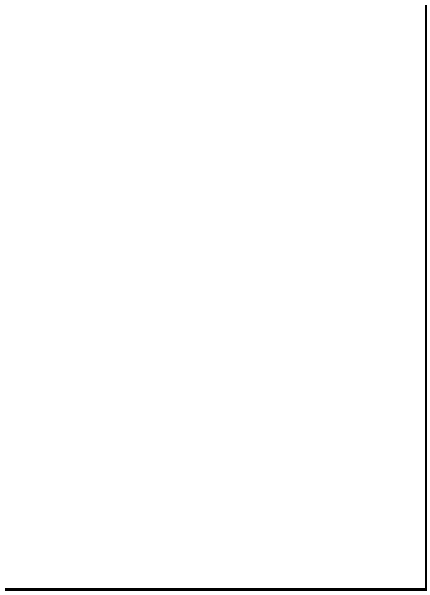
6.2 Spesifikasjoner for systemet

6.2.1 Rammelesing

En rammelesings-sekvens tar 64+1 = 65 klokkesykler å fullføre. Selve lesingen av aktuell rammeinformasjon tar 64 klokkesykler, fordi lengden av *ADR*-felt + *COM*-felt + *TVAL* + *CHK* = 4·16 = 64bit. I tillegg overlapper kontrollsignalet #EN (ENable), disse 64 bit med én klokkesykel.

I alt seks signaler brukes til rammelesing og synkronisering:

- #FLAG: signaliserer starten på en ramme
- #EN: aktiveres etter et flagg, starter en rammelesings-sekvens
- #AL: laster inn adresseverdi fra *ADR*-felt når rammeteller=16
- #BL: laster inn kommando fra *COM*-felt når rammeteller=32
- #CL: laster inn timerverdi fra *TVAL*-felt når rammeteller=48
- #DL: laster inn sjekksum fra *CHK*-felt når rammeteller=64, og deaktiverer #EN, som avslutter en rammelesings-sekvens



Figur 6.2: Flytskjema for ramme-lesing og -synkronisering

Flytskjemaet i figur 6.2 beskriver synkroniseringen av rammelesing. "Decision"-boksene viser hvilke signaler som testes på ved hvert punkt, og utgangsverdiene ("0"/"1") er også vist. Rektangulære bokser representerer start- og stopp-tilstandene i en sekvens, mens avrundede bokser angir mellomtilstander.

6.2.2 Kommandofunksjoner

Verdien av kommandofeltet, *COM*={#b0,#b15}, i data-rammen styrer funksjonen til systemet. Det er kún tre kommandoer å ta hensyn til:

1. "Timer på": *COM*={01} dekoder kommandolinjen #K1=0
2. "Timer av": *COM*={10} dekoder kommandolinjen #K2=0
3. "Stopp klokke": *COM*={11} dekoder kommandolinjen #K3=0

"Timer på" *starter* timeren slik at den teller opp til den verdien som feltet *TVAL* i rammen inneholder. Nye rammer kan mottas Når denne verdien er nådd, blir signalet #TO(TimeOut)=1, for å markere at timerintervallet er utløpt. Dette gjelder ikke dersom en ny ramme inneholdende en av de to andre kommandoene allerede har blitt mottatt og dekodet.

"Timer av" *stanser* timeren dersom den er i ferd med å telle opp, og resetter den til null. Dette utføres først når timing-signalet #CL=1. Verdien i *TVAL*-feltet brukes ikke.

Figur 6.3: Flytskjema for digitaldel

"Stopp klokke" stopper lokal klokke på mottagerenheten. Den stanser også timeren, og resetter den til null. Merk her at bare klokka på den *adresserte* enheten stanses. Bare én mottagerklokke av gangen kan derfor slås av, dersom alle enheter har unike adresser. Denne funksjonen utføres først når #DL=1, altså når hele rammen har blitt mottatt. Men dersom sjekksummen indikerer en feil, utføres kommandoen ikke. Verdien i timerfeltet *TVAL* brukes ikke i dette tilfellet heller.

6.2.3 Synkronisering av kommandoer

Jeg har gjort to endringer på digitaldelen i forhold til flytskjemaet i figur 6.3. Den ene endringen består i at ikke-matchende adresse, #AI=0, *ikke* resetter kontroll-logikken. Det som i stedet skjer, er at registerne "BR", "CR" og "DR" ikke latches, men beholder gammel verdi. Hele rammen leses, men ingen kommando utføres. Dette gjorde jeg fordi det var enkelt, men spesielt fordi jeg ville ha like rammelesings-sekvenser. Dette er mest på grunn av testmulighetene; dersom en feil gjorde at signalet #AI=0 bestandig, ville jeg ikke kunne se om rammelesings-logikken fungerte for en hel sekvens.

Den andre endringen går ut på at timeren ikke startes og stoppes på samme tidspunkt i rammelesings-sekvensen. Egentlig skal timeren startes og stoppes ved synkroniseringspunktet #CL=1, men for å forvisse meg om at ikke #CL styrer timeren alene, stoppes timeren dersom #K2=0 ved synkroniseringspunktet #BL=1. Husk at kommandoen "Timer av", #K2=0, ikke trenger noe timerverdi-argument fra feltet *TVAL* slik at dette ikke bryter med andre spesifikasjoner. På denne måten utføres hver kommando ved hvert sitt synkroniseringspunkt; "Timer på" ved CL=1, "Timer av" ved #BL=1, og "Stopp klokke" ved #DL=1. For mest mulig énhetlig kommando-synkronisering, er det kanskje best at kommando-utføringen skjer ved slutten av en rammelesings-sekvens, når #DL=1, men jeg mener at testing blir enklere ved å kunne identifisere kommandosekvensen ut i fra distinkte tidspunkter.

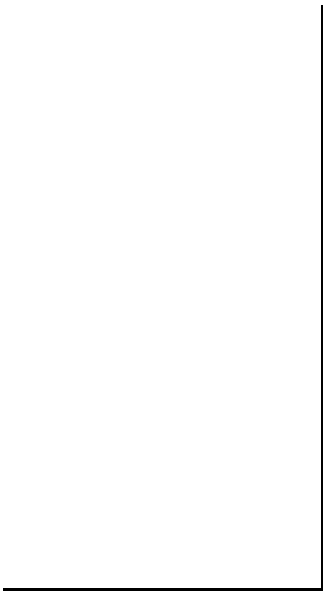
6.2.4 Klokkestyring

En fjerde kommando, "Start klokke", er egentlig bare et signal, #STARTCK, og ikke en kommando. Dette signalet dekodes ikke fra kommando-feltet i datarammen, men kommer fra pulsdetektor-kretsen i analogdelen av brikken. Signalet har ingen funksjon dersom ikke klokke, altså oscillatoren, er avslått. Men er dette tilfellet, vil #STARTCK=1 slå klokke på igjen. Denne sammenhengen er vist i figur 6.4. Overgangen fra tilstand der #STOPCK=1, til #STOPCK=0, skjer uten klokking i og med at VCO er deaktivert.

6.2.5 Feilsjekking

Feilsjekkingen utføres med et tilbakekoblet register. Tre signaler er involvert bare i feilsjekkingslogikken:

- #SAREN : *SignaturAnalyseRegisterENable*, starter sjekksum-generering



Figur 6.4: Flytskjema for spesialtilfelle der klokke slås av

- #RSAR : *ResetSignaturAnalyseRegister*, nullstiller sjekksum-generator
- #EI : *ErrorIndikasjon*, signaliserer én eller flere bitfeil i ramme

6.3 Logiske blokker

6.3.1 Kontroll-logikk

Kontroll-logikken er bygd opp av to blokker: en tilstandsmaskin, "tm", og en 6-bits synkron teller, "6bsynkrt". Tilstandsmaskinen besørger sekvensiell kontroll, mens telleren foretar rammelesings-synkronisering.

Funksjon til 6-bits, synkron teller:

$$t0(n) = \overline{t0(n-1)}$$

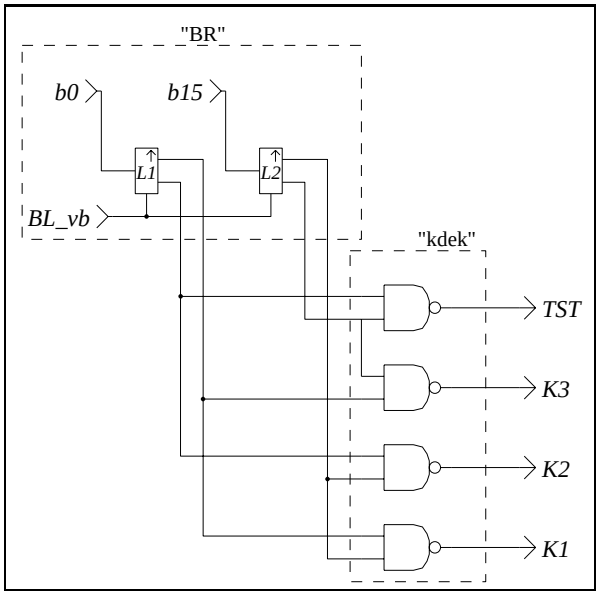
$$t1(n) = t0(n-1) \oplus t1(n-1)$$

$$t2(n) = (t0(n-1) \cdot t1(n-1)) \oplus t2(n-1)$$

$$t3(n) = (t1(n-1) \cdot t2(n-1)) \oplus t3(n-1)$$

$$t4(n) = (t2(n-1) \cdot t3(n-1)) \oplus t4(n-1)$$

$$t5(n) = (t3(n-1) \cdot t4(n-1)) \oplus \overline{t5(n-1)}$$



Figur 6.5: kommando-dekoder og -register

Uttrykket for laveste bit kan også skrives som:

$$t0(n) = 1 \oplus t0(n-1)$$

Og tilsvarende for nest laveste bit:

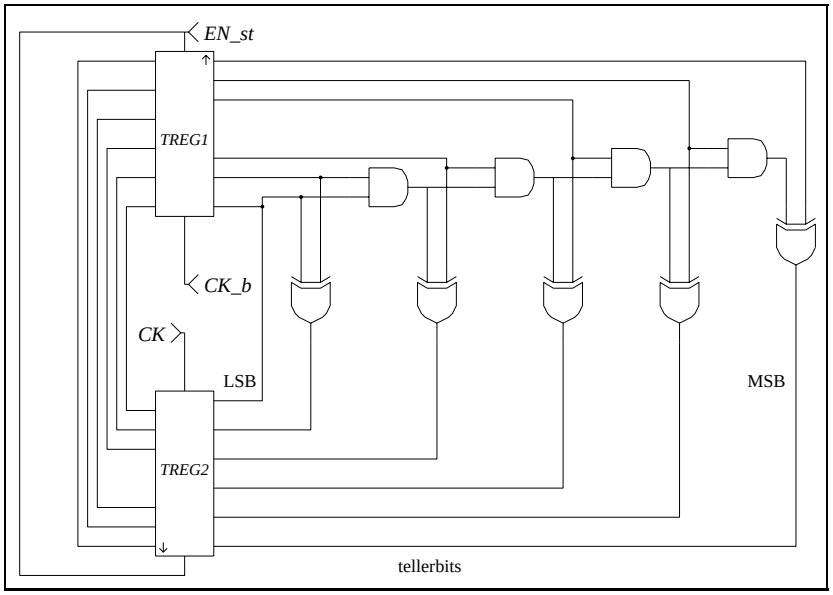
$$t1(n) = (1 \cdot t0(n-1)) \oplus t1(n-1)$$

Generelt uttrykk blir derfor:

$$t_m(n) = (t_{m-2}(n-1) \cdot t_{m-1}(n-1)) \oplus t_m(n-1)$$

der $t_{m-p}(n-1) = 1$, dersom $(m-p) < 0$

Det høyeste bitet bestemmer hvilken vei telleren skal telle. Fordi telleren skal telle i retning *oppover*, brukes invertert MSB, $\overline{\#t5(n-1)}$, i uttrykket for $\#t5(n)$. Figur 6.6 viser kretsskjema for synkron teller.



Figur 6.6: Synkron teller for rammelesing-kontroll

Tellerbitene dekodes slik at et omløp på 64 klokkesykler gir en puls på 4 forskjellige kontroll-linjer med en avstand lik 16 klokkesykler. Disse 4 linjene, kalt #AL, #BL, #CL og #DL, latcher hvert sitt 16-bit rammefelt. De brukes med andre ord til å lese inn de 4 feltene med informasjon som er forskjellig fra ramme til ramme. Samtidig gir de synkroniseringspunkter til sekvens-kontrollen. Logisk uttrykk for signalene fra rammelesing-kontroll:

$$AL = (t0 \cdot t1 \cdot t2 \cdot t3) \cdot \overline{t4} \cdot t5$$

$$BL = (t0 \cdot t1 \cdot t2 \cdot t3) \cdot t4 \cdot t5$$

$$CL = (t0 \cdot t1 \cdot t2 \cdot t3) \cdot \overline{t4} \cdot \overline{t5}$$

$$DL = (t0 \cdot t1 \cdot t2 \cdot t3) \cdot t4 \cdot \overline{t5}$$

Disse signalene brukes ikke direkte til å latche registere, men føres som input til NAND-porter sammen med signalet #AI fra adresse-komparatoren, og $\overline{\#CK}$ (se figur 6.8. Dette betyr at ingen av utgangene er aktive, det vil si ingen felt leses inn i tilhørende registere, dersom adresseringen er ugyldig. Unntaket er #ALvb som ikke er avhengig av #AI, fordi dette signalet brukes til å lese inn adresse.

$$ALv = \overline{AL} + CK$$

$$BLv = \overline{BL} + \overline{AI} + CK$$

$$CLv = \overline{CL} + \overline{AI} + CK$$

$$DLv = \overline{DL} + \overline{AI} + CK$$

Tilstandsmaskinen får input vesentlig fra telleren *og* fra kommando-dekoderen, "kdek". Dekoderen er en enkel 2:4 NAND-dekoder (se figur 6.5). De to inngangene på dekoderen kommer fra et 2-bits kommando-register, "BR". Registeret latcher igjennom data-bit #b0 og #b15 fra kommando-feltet, *COM*. Ut fra registeret kalles disse verdiene #k0 og #k1. Logisk funksjon til kommando-dekoderen er illustrert gjennom tabell 6.1.

#k1	**#k0**	**#K1**	**#K2**	**#K3**	**#TST**
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	1

Tabell 6.1: Sannhetstabell for kommando-dekoder, "kdek"

Bit #k0 og #k1 dekodes til kommandoene #K1, #K2, #K3 og #TST, etter funksjonstabell gitt i tabell 6.1. Kommandolinjene er aktive ved lavt nivå, og bare én av gangen. #K1, #K2 og #K3 er input til tilstandsmaskinen. #K1=0 angir: "timer på", #K2=0 angir: "timer av", #K3=0 angir: "klokke av". En "logisk feil", er at *to* kommandoer #K3=0="timer på" *etter hverandre*, likevel slår timer av. Det må derfor settes som en forutsetning at to påfølgende "timer på"-kommandoer er ulovlig. Bortsett fra et mulig ønske om å forandre timer-intervall, er dette dessuten også en selvmotsigende kommandosekvens. Dekodingen av bitene i BR til kommandosignaler, er som følger:

$$K1 = \overline{k0} + \overline{k1}$$

$$K2 = k0 + \overline{k1}$$

$$K3 = \overline{k0} + k1$$

$$TST = k0 + k1$$

I tillegg til kommando-input, får tilstandsmaskinen input-verdiene #STARTCK fra puls-detektoren ("pdtct"), #EI fra sjekksum-komparatoren ("scmp"), #FLAG fra flagg-detektoren ("fd"), #CLuv og #DLuv fra teller-dekoderen ("tdek"). Utgangsverdier fra tilstandsmaskinen er kontrollsignaler som styrer kretsens funksjon. Kontrollsignalene er: #STOPCK, #EN, #STT, #SAREN og #RSAR. #STOPCK slår oscillator, altså VCO, av og på. #EN er enable-signal for rammelesings-logikk. #SAREN er enable-signal for signatur-generator ("SAR"). #RSAR er reset-signal for ditto.

#FLAG	#DL	#EN(n)	#EN(n+1)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Tabell 6.2: Sannhetstabell for kontrollsignal #EN

Det er mulig å utvide uttrykkene som inkluderer input fra rammelesings-kontroll, det vil si #CL og #DL, ved å ekspandere dem slik at tellerbitene tas med. Men dette blir så rotete og uoversiktlig at det er bedre å betrakte rammelesings-kontrollen som en "svart boks", og bare bruke de endelige utgangs-signalene #CL og #DL. Logisk uttrykk for kontrollsignalene er som følger:

- ENable-signal for rammelesing/kontroll:

$$EN(n) = \overline{DL} \cdot (EN(n-1) + FLAGG)$$

- SARENable-signal for signaturgenerator-kontroll:

$$SAREN(n) = \overline{CL} \cdot (EN(n-1) + FLAGG)$$

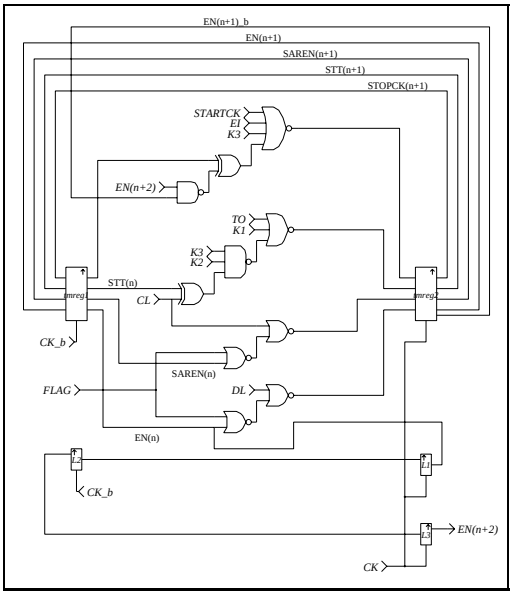
- STT(SeTTimer)-signal for timerkontroll:

$$STT(n) = \overline{K1} \cdot K2 \cdot K3 \cdot \overline{TO} \cdot (STT(n-1) \oplus CL)$$

- STOPCK-signal for klokke-kontroll:

$$STOPCK(n) = \overline{STARTCK} + \overline{EI} + \overline{K3} + ((EN(n-1) \cdot \overline{EN(n)}) \oplus \overline{STOPCK(n-1)}$$

Tilstandsmaskinen er sammensatt av en blokk kombinatorisk logikk (se figur 6.7), og to 5-bits registre. De to registerne, "tmreg1" og "tmreg2" klokkes av hver sin klokkefase, "tmreg1" av #CK, og "tmreg2" av #CKb.



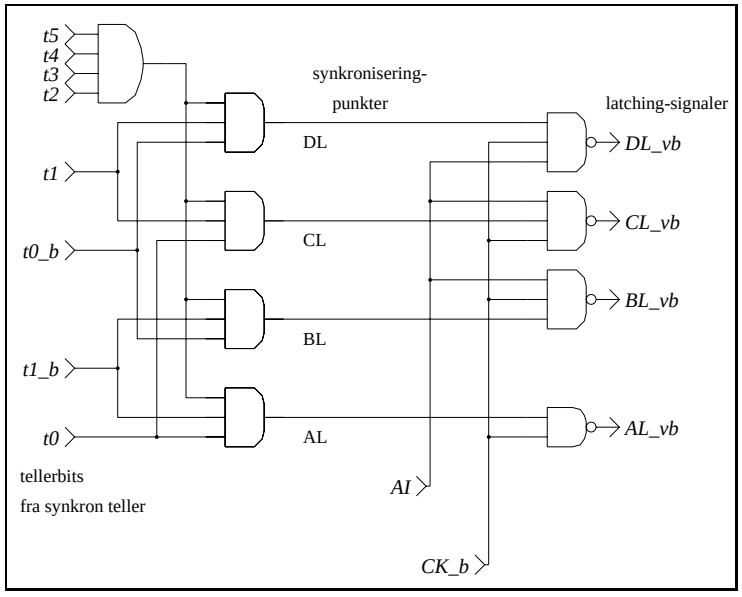
Figur 6.7: Tilstandsmaskin

Denne tabellen er indentisk med den for signalet #EN, dersom synkroniseringspunktet #CL byttes ut med #DL. Derfor blir også logikken den samme, med to stykk 2-input NOR-porter (se figur 6.7).

Alle input-verdier som ikke er fylt inn i tabellen er "x"-verdier ("Don't Care"). For kommandoen #K1=0="Timer på", har ikke synkroniseringspunktet #BL=1 noen innvirkning. Men for kommandoen #K2=0="Timer av", er det CL=1 som ikke har noen virkning. Kommandolinjen #K3 resetter umiddelbart #STT når den går lav for å signalisere kommandoen "Stopp klokke", hvilket skjer synkront med synkroniseringspunktet #BL=1 fordi

#FLAG	**#CL**	**#SAREN(n)**	**#SAREN(n+1)**
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Tabell 6.3: Sannhetstabell for kontrollsignal #SAREN



Figur 6.8: rammelesings-dekoder

dette latcher kommandoverdien.

6.3.2 Skiftregister

Data blir skiftet fra serieform til parallell av et 16-bits skiftregister, "16bsr". Samtlige 16 bit blir bare brukt til sjekksum, mens 4 laveste (#b0..#b3) blir brukt til adresse og timerverdi, og laveste samt høyeste bit, #b0 og #b15, brukes til kommando. Den store redundansen kan forsvares med at jo lengre sjekksummen er, jo færre feil passerer uoppdaget.

Et flagg i data-strømmen oppdages med en flagg-detektor, "fd". Kretsskjema er vist i figur 6.10. Dette er en enkel dekoder med logisk utgangsverdi #FLAG=1 for et bitmønster tilsvarende et flagg, det vil si {1010..1010}. Logisk uttrykk for FLAG-bitet:

#K1	**#K2**	**#K3**	**#BL**	**#CL**	**#TO**	**#STT(n+1)**
0				0	0	STT(n)
0				1	0	1
0				x	1	0
	0		0			STT(n)
	0		1			0
		0	0			STT(n)
		0	1			0

Tabell 6.4: Sannhetstabell for kontrollsignal #STT

Figur 6.9: fullstendig sekvenskontroll

$$FLAG = \overline{(b0 \cdot b2 \cdot b4 \cdot b6)} \cdot \overline{(b8 \cdot b10 \cdot b12 \cdot b14)} \cdot (b1 \cdot b3 \cdot b5 \cdot b7) \cdot (b9 \cdot b11 \cdot b13 \cdot b15)$$

Evaluering av adresse skjer med en adresse-komparator, "acmp". Dersom adressfelt ADR=mottageradresse, er utgangen fra "acmp" logisk 1. Denne verdien lastes gjennom registeret AR dersom #ALv=0. Ut fra AR kalles denne verdien #AI.

Funksjonsuttrykk for adresse-komparator:

$$AI = \overline{((a0 \oplus b0) + (a1 \oplus b1) + (a2 \oplus b2) + (a3 \oplus b3))}$$

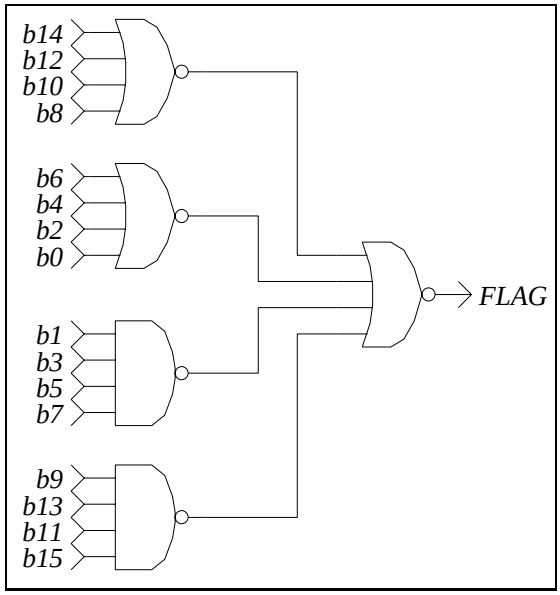
Formelt kan funksjonen beskrives som:

$$a_n = b_n \rightarrow \underline{AI = 1} \qquad ; n \in 0, .., 3$$

$$a_n \neq b_n \rightarrow \underline{AI = 0} \qquad ; n \in 0, .., 3$$

6.3.3 Sjekksum-register

Sjekksum-generatoren er et 16-bits skiftregister der enkelte av bit-linjene er input til XOR-porter. Disse er koblet i serie, og utgangen fra *siste* XOR-port XOR'es med dekodede data fra PLL. Verdien fra denne XOR-porten, kalt polynomverdien, er inngangsverdi til skiftregisteret.



Figur 6.10: Flagg-detektor

Sjekksum-polynomet er bare gitt av de tilbakekoblede verdiene. Her blir det lik:

$$P_s = (s0 \oplus s5) \oplus s12$$

Dette kan også skrives slik:

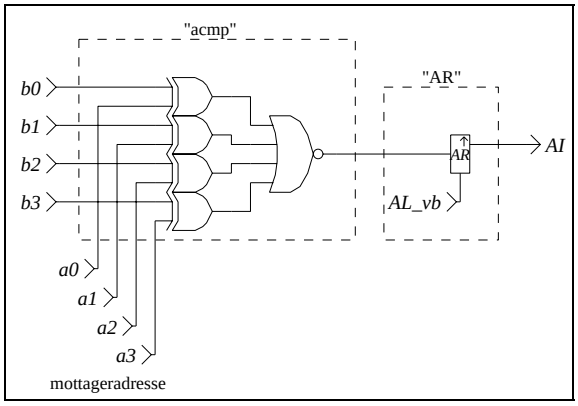
$$S(n) = (S(n - p_1 - 1) \oplus S(n - p_2 - 1)) \oplus S(n - p_3 - 1)$$

Som her gir:

$$S(n) = (S(n - 1) \oplus S(n - 6)) \oplus S(n - 13)$$

Der S(n) er ny verdi inn i registeret.

Signaturen generert fra overførte data sammenlignes med oversendt sjekksum/signatur. Dette blir gjort i "scmp", en XOR-komparator. Input til komparatoren er derfor signatur-bits, {#s0...#s15}, og data-bits, {#b0...#b15}. Dersom signaturen er overenstemmende med sjekksum, er utgangsverdien til "scmp" lik logisk 0. Denne verdien lastes gjennom én-bits registeret "DR" dersom #DLvb=0. Utgangen til register "DR" er #EI.



Figur 6.11: adresse-sjekker og A-register

Funksjonsuttrykk for signatur-komparatoren:

$$EI = (s0 \oplus b0) + (s1 \oplus b1) + (s2 \oplus b2) + ... + (s15 \oplus b15)$$

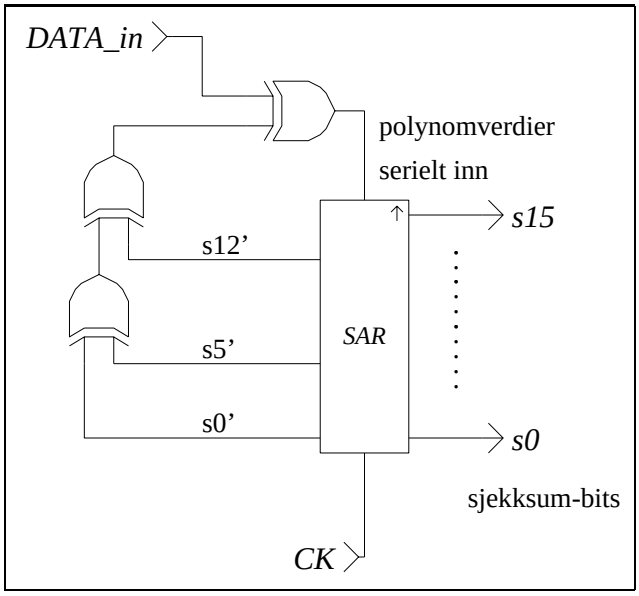
$$s_n = b_n \rightarrow \underline{EI = 0} \qquad ; n \in 0, .., 15$$

$$s_n \neq b_n \rightarrow \underline{EI = 1} \qquad ; n \in 0, .., 15$$

6.3.4 Timer

Timeren er i hovedsak en 20-bits rippel-teller, der de 4 mest signifikante bit utgjør en prescaler. Dette betyr at timeren kan telle i intervaller på $n \cdot 2^{16}$ klokkeperioder, der n er et tall mellom 0 og 15. Prescaler-verdien, n, settes av timerverdi-feltet, $TVAL$, og bestemmer timer-intervallet. Mulige intervaller er 0 klokkeperioder, 65536 klokkeperioder, $2 \cdot 65536 = 131072$ klokkeperioder, og så videre opp til $16 \cdot 65536 = \underline{1048576}$ klokkeperioder maksimalt. Dette gir en forsinkelse i tid på cirka 10.5 sekunder med en 100kHz klokke.

Timeren er vist skjematisk i figur 6.14. Den er bygd opp av en teller, et register, og en komparator. Telleren er som nevnt over en 20-bits rippelteller. At den er en rippelteller, betyr at den ikke teller synkront, men vil ha flanker på teller-utgangene forskjøvet med forsinkelsen gitt av hver port rippel-signalet har passert. Jo høyere teller-bit, jo flere porter i serie, og desto større forsinkelse. Men dette er ikke av betydning fordi utgangen fra timeren, #TO, ikke brukes til noen form for synkronisering. Ulempen her er bare at vi ikke får et eksakt timer-intervall i henhold til en klokkeperiode på 10us, men et noe *lengre* intervall. Dette gjelder også dersom oscillatoren driver mot lavere frekvenser. Dersom VCO driver mot høyere frekvenser fås kortere intervall.



Figur 6.12: signatur-register

Når 65536 klokkeperioder er passert, endrer det 16.trinnet i telleren verdi. Dette går videre som klokkesignal til det 1.trinnet i prescaler. Dersom de fire utgangsverdiene fra prescaler (#p0,#p1,#p2,#p3) er sammenfallende med verdiene i *TVAL*, vil utgangen på XOR-komparatoren, #TO, gå høy. #TO=1 betyr med andre ord utløpt timerintervall. Verdien av *TVAL* holdes av et 4-bits register, "CR". Dette registeret laster igjennom de 4 minst signifikante bit (#bo,#b1,#b2,#b3) dersom #CLv=0.

Funksjonsuttrykk for timer-komparator:

$$TO = \overline{((c0 \oplus p0) + (c1 \oplus p1) + (c2 \oplus p2) + (c3 \oplus p3))}$$

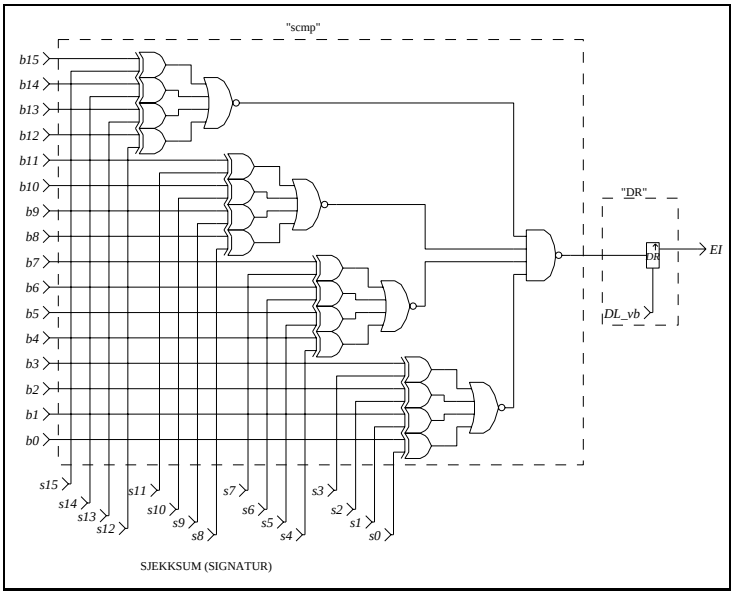
Formelt kan funksjonen beskrives som:

$$c_n = p_n \rightarrow \underline{TO = 1} \qquad ; n \in 0, .., 3$$

$$c_n \neq p_n \rightarrow \underline{TO = 0} \qquad ; n \in 0, .., 3$$

6.4 Simuleringer

Funksjonell testing av digitaldelen ga lite analysérbare resultater. Ingen av utgangene ble endret for forskjellige input. De lå alle til *GND* unntatt #SARCK, dersom jeg satte testsignalet #TSTSAR=1. I normalmodus, der PLL forsyner digitaldelen med klokkesignal, skiftet skiftregisteret i sjekksum-generatoren bitverdier videre. Bitmønsteret endret seg tilsynelatende tilfeldig, slik at det ser ut som om den genererte en tilfeldig bitsekvens basert på sjekksum-polynomet.



Figur 6.13: signatur-komparator og D-register

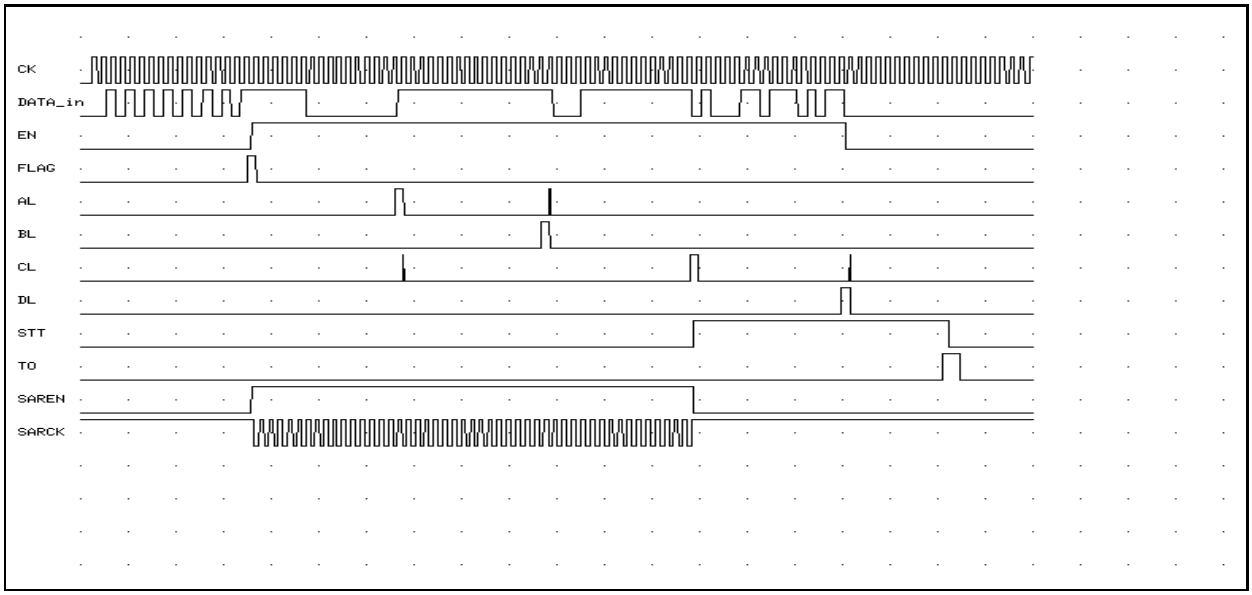
Jeg har tatt med noen simuleringer av systemet for å belyse digitaldelens funksjon. Signalene #AI og #EI er ikke tatt med fordi de er statiske (faste) for disse sekvensene. Dette gjelder ikke dersom adresse eller sjekksum er ugyldig.

Første ramme inneholder kommandoen "Timer på". Timer-intervallet er $TVAL=\{0001\}$, eller timerverdi=1·16=16 klokkesykler. For denne lave timerverdien kan vi se i figur 6.15 at timer resettes; #STT=0, når timer-intervallet utløper; #TO=1. Synkroniseringspunktene #AL=1, #BL=1, #CL=1 og #DL=1 er adskilt med 16 klokkesykler, eller en rammefeltlengde. Signalene #EN og #SAREN går høye når #FLAG=1. #EN forblir høy inntil #DL=1, når hele rammen er lest inn. #SAREN går lav før sjekksummen leses ved #CL=1, slik at ikke mottager-generert sjekksum beregnes på grunnlag av bitverdiene i feltet *CHK*.

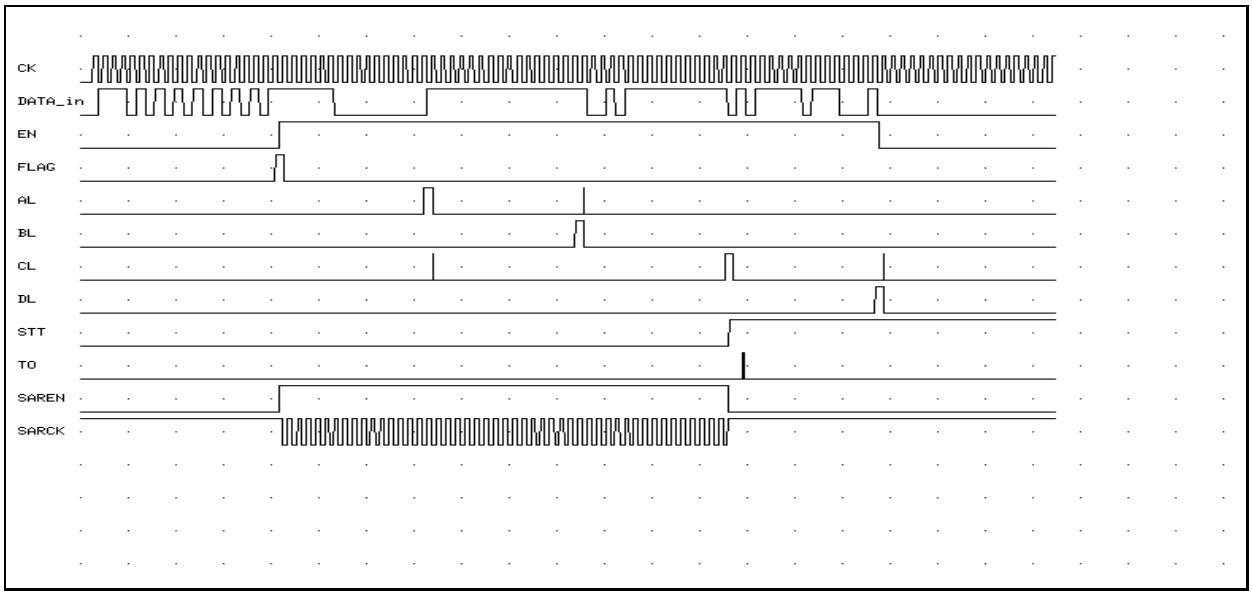
Legg merke til signalet #SARCK nederst. Det er klokkepulser til signaturgeneratoren. Altså leses bare de 48 bits i de tre rammefeltene *ADR*, *COM* og fra rammen inn i signaturgenerator. Bitverdien i *SYNC* og *BOF* er den samme for hver ramme og brukes følgelig ikke i sjekksum-genereringen.

Simuleringen vist i figur 6.17 er gjort i sekvens etter forrige simulering med kommandoen "Timer på" vist i figur 6.16. Altså har timeren startet nedtellingen av timerintervallet=64 klokkesykler når neste ramme sendes. Den inneholder kommandoen "Timer av". Ved synkroniseringspunktet #BL=1, går #STT lav igjen og timeren stanses og resettes. Det er med andre ord signalet #K2=0 som her resetter timer, og ikke #TO=1.

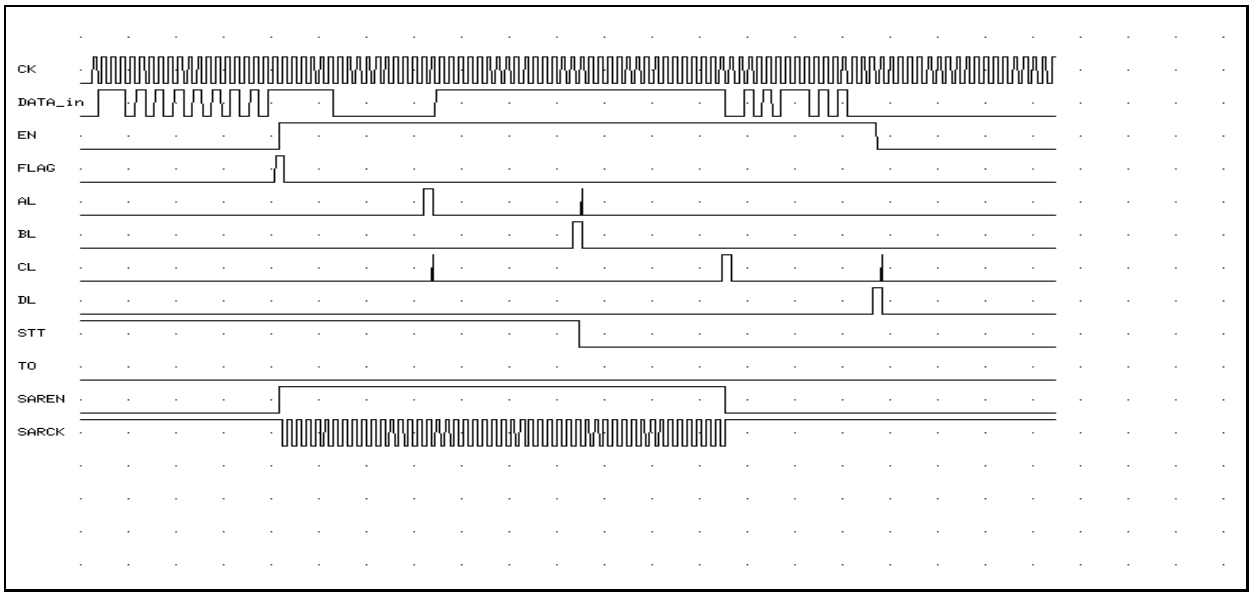
Figur 6.14: Timer-register og delay-timer



Figur 6.15: Simulering av digitalsystem med kommando “Timer på”



Figur 6.16: Simulering med kommando "Timer på", og timerverdi=4



Figur 6.17: Simulering med kommando "Timer av"

Kapittel 7

Fra delblokker til system

Sammenkobling av digitale delkretser er allerede omtalt, og sammenkoblingen av analoge delkretser er gitt av hva som er inngangs-signal og utgangs-signal til hver delkrets. I dette kapittelet vil jeg først og fremst ta for meg de blokkene som utgjør forbindelsen mellom analog og digital del(krets). Disse delkretsene ble ikke omtalt i kapittelet om analoge delkretser, fordi de ikke hadde noen direkte innflytelse på mottagerens analoge funksjoner, og fordi deres egen funksjon er av digital natur.

7.1 Separate komponenter til PLL

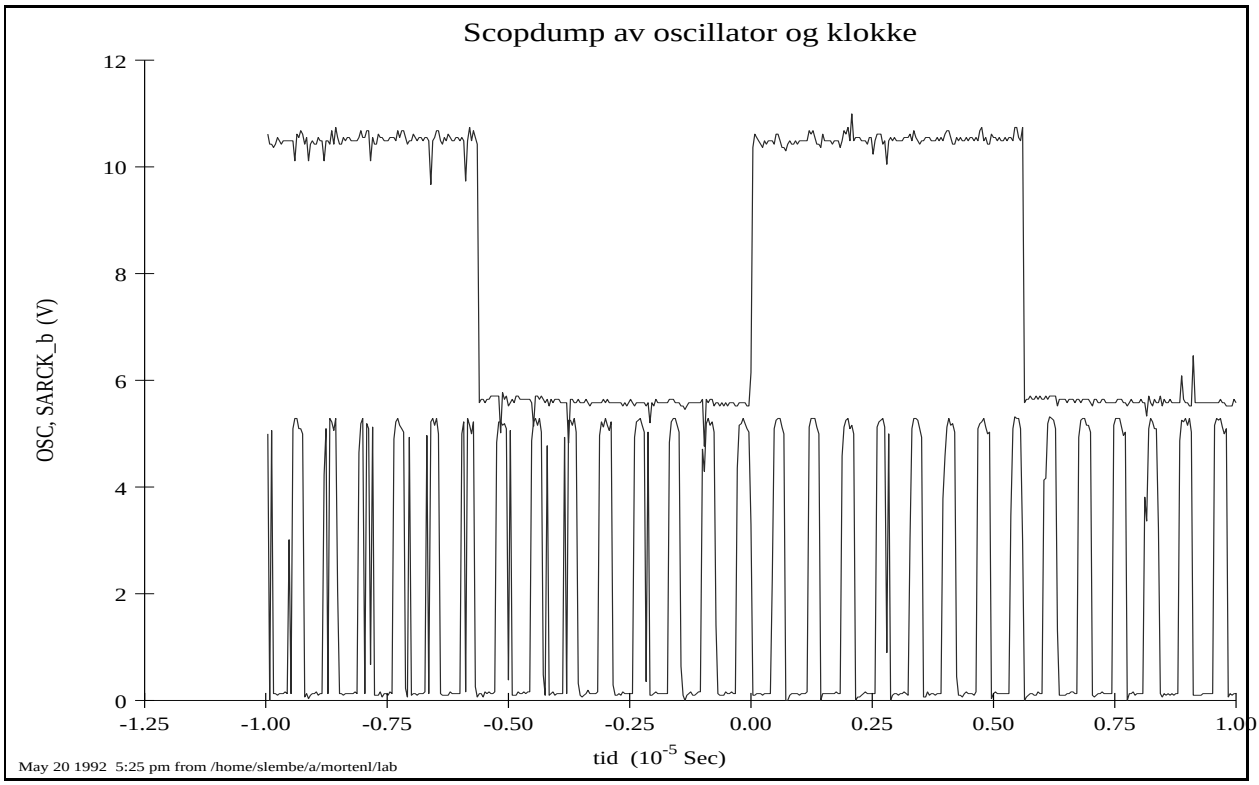
7.1.1 Frekvensdeler

Frekvensdeleren er simpelthen en binær teller. Denne 4-bits telleren er bygd opp av latcher, og deler ned oscillatorfrekvensen med faktoren N=16. Fordi $f_0^{'}$ er 1.6MHz (ω_{o0}= cirka 10Mrad/s), blir frekvensen ut av frekvensdeleren, f_o= 100kHz (ω_0= 628Krad/s). Dette er en rippel-teller, og derfor vil klokkeflankene ut fra telleren være forsinket med summen av forsinkelsen i hvert ledd i telleren, i forhold til klokkeflanker inn fra oscillator.

Telleren har 4 ledd, og i alt 8 latcher. Men disse latchene klokkes av hver sin klokkefase i hvert ledd, og derfor blir totalforsinkelsen bare gitt av forsinkelsen igjennom 4 latcher. En latch har en maksimal forsinkelse (simulert med utgangskapasitans $C_L = 1pF$) på omtrent 12ns. $4 \cdot 12ns = 48ns$, altså til sammen rundt 50ns. En klokkeperiode er på 10us (ved $f_o = 100kHz$), og delay introdusert av porter/logikk/teller utgjør $\frac{5 \cdot 10^{-8}}{10^{-5}} = 5 \cdot 10^{-3} = \underline{0.005}$ av en periode. Perioden er på 360°, og faseforskyvningen forårsaket av porter er derfor:

$\theta_p = 0.005 \cdot 360° = \underline{1.8°}$

Fasemarginen er 90°, og faseforskyvningen utgjør $\frac{1.8°}{90°} = 0.02 = 2\%$ maksimalt av fasemarginen. Dette er *ikke* en betydelig del av fasemarginen, og vil sjelden kunne gi problemer.

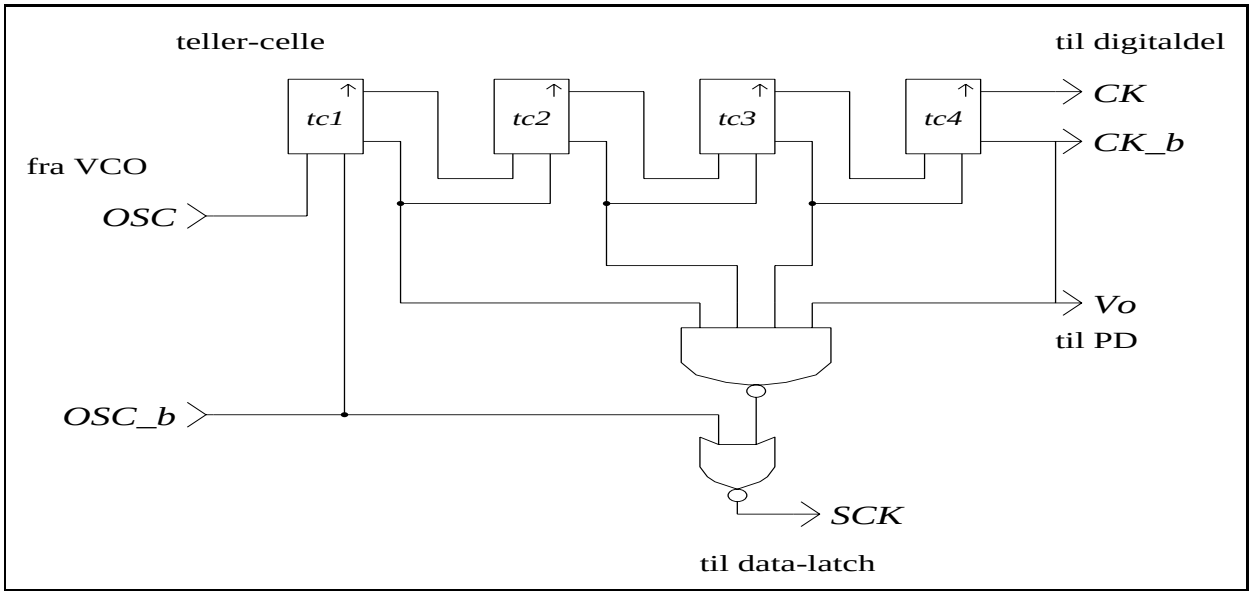


Figur 7.1: VCO-utgang og tilsvarende invertert klokke

Figur 7.1 viser VCO-signalet, #OSC, og utgangen #CK, eller V_o, av frekvensdeleren. Systemklokken er høy for 8 perioder av oscillatorsignalet, og lav et tilsvarende tidsrom. Legg merke til at klokkingen i frekvensdeleren skjer på negativ (fallende) flanke, fordi latchene klokkes med lavt nivå.

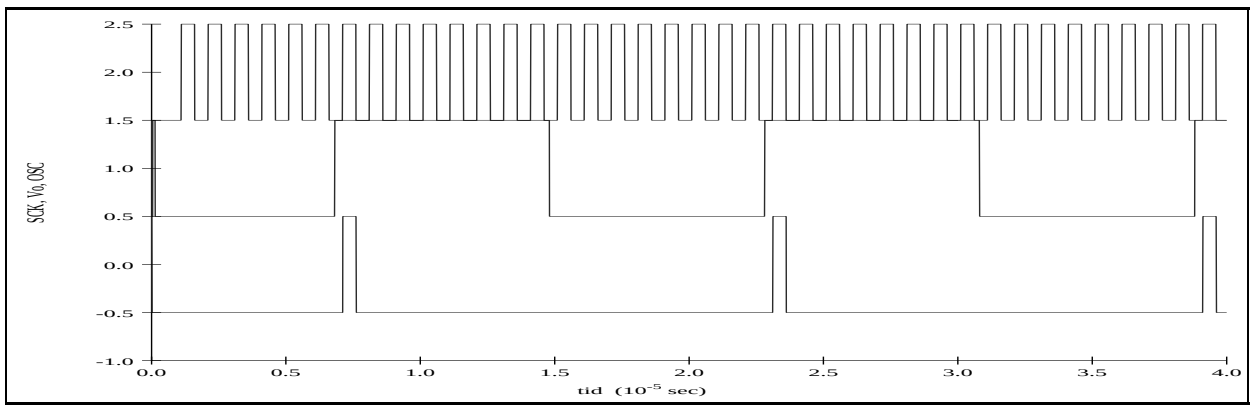
7.1.2 Dekoder for datagjenvinning

Datagjenvinning foretas med en negativ-klokket latch og to logiske porter; en 4-input NAND-port, og en 2-input NOR-port, som vist i kretsskjemaet i figur 7.2. Det RZ-kodede signalet klokkes gjennom latchen av signalet #OSCK, og fra utgangen på latchen går dekodet datastrøm til den digitale systemdelen. Data-latchen klokkes av utgangssignalet fra NOR-porten. Input til NAND-porten er de fire utgangene fra hvert ledd i frekvensdeleren (telleren), slik at porten går lav når alle disse utgangene får en positiv flanke. Dette er en tilstand som bare inntreffer i hver *sekstende* periode av oscillatorsignalet, som er synkroniseringspunktet.



Figur 7.2: data-latch og klokkingslogikk

Signalet fra NAND-porten kunne jeg ha brukt som klokkingssignal direkte, fordi data-latchen klokkes med lavt nivå (#OSCK=0). Men fordi frekvensdeleren er en rippelteller, vil det være glitcher i signalet. Dette eliminerer jeg ved å sende signalet inn på NOR-porten sammen med det *inverterte* oscillator-signalet (#OSCb). Frekvensdeleren klokkes på oscillator-signalets *negative* flanke, og derfor vil eventuelle glitcher komme like etter denne flanken. Men i det *negativ* flanke på oscillator-signalet kommer, er tellerutgangene stabile, og signalet ut fra NOR-porten (#OSCK) er også stabilt.



Figur 7.3: Signalforløp for VCO, klokke og pulser til datalatch

Dette betyr at en forsinkelse på en *halv periode* av oscillator-signalet er blitt introdusert, hvilket igjen er 1/32-del av en klokkeperiode. Men dette betyr bare at timing-marginen for selve *klokkingen* av data er blitt dårligere, selve synkroniseringen skjer fortsatt i henhold til positiv flanke, altså uendret fasemargin. Signalforløp og timing i dekoder-logikken er illustrert i simuleringen i figur 7.3.

Testing av kretsen ga indikasjoner på at data-bits ikke ble klokket inn korrekt. Dobbelsjekking av utlegget ga forklaringen: Klokke fra VCO var forbyttet med invertert klokke i hele dekoder-logikken. Samplings-pulser til data-latchen kommer derfor antakelig ved negativ transisjon i V_o, dersom nye simuleringer stemmer overens med faktisk dekoder-funksjon, og bitverdien "0" klokkes kontinuerlig inn til digitaldelen.

Denne feilen har vanskeliggjort all digital testing på kretsen, og dette illustrerer hvor viktig sammenkoblingen mellom digitaldelen og analogdelen er. Delkretsene i dekoderen er enkle og oversiktlige, men desto mer vitale.

7.2 Sammenkobling av analogdel og digitaldel

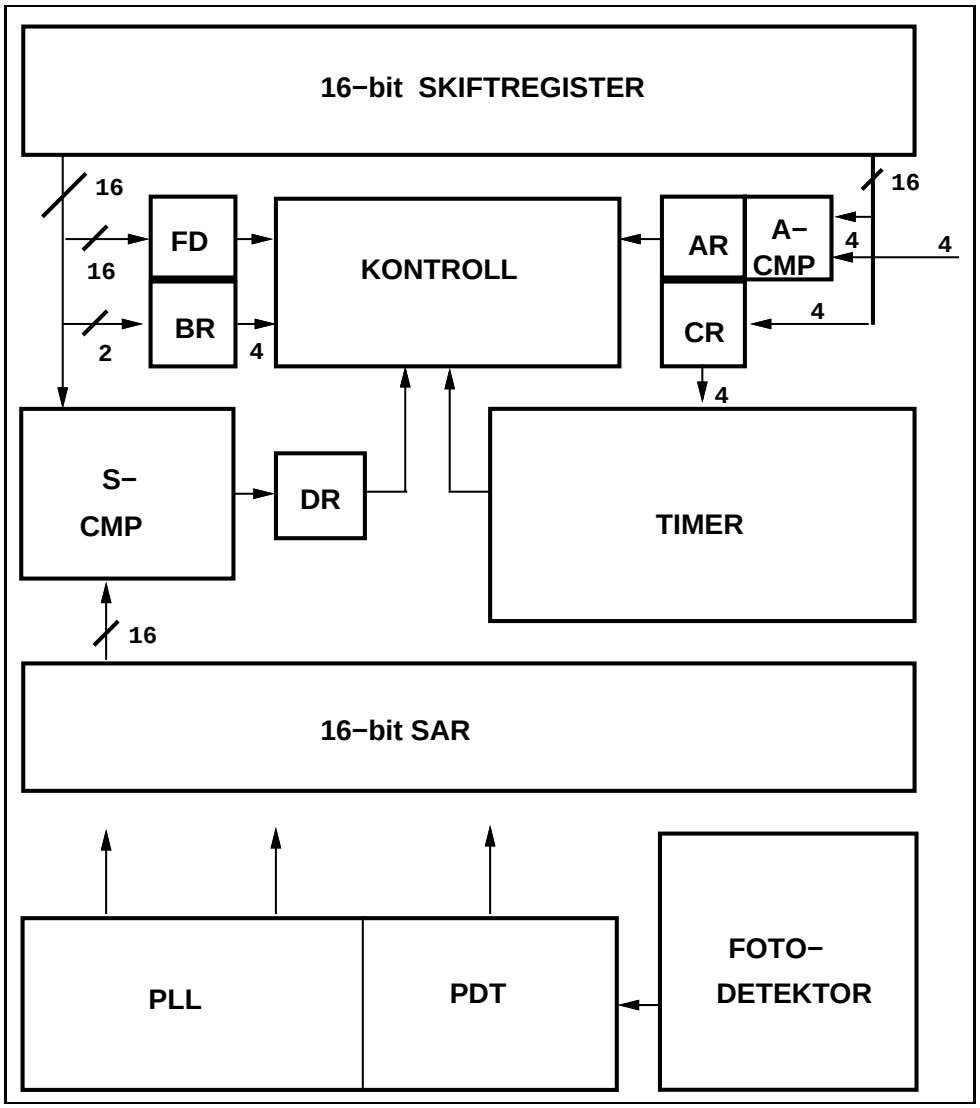
Relativt få signaler går mellom analogdel og digitaldel. Kommando-linjen #STOPCK er eneste signal fra digitaldel til analogdel, mens andre veien går generert klokke og dekodede data fra PLL. I tilleg er signalet #STARTCK også ført fra puls-detektor til digitaldelens tilstandsmaskin. Dette er for å sette #STOPCK=0 ved første klokkepuls, slik at vi ikke risikerer å stanse klokke igjen umiddelbart. Grovskisse av sammenkoblingen er vist i figur 7.4.

7.3 Oppsummering

Dette kapittelet har forklart hvordan analoge delkretser settes sammen til komplett mottagersystem. Videre er sammenkoblingen mellom analogdel og digitaldel blitt vist. Et

Figur 7.4: Sammenkobling av analog og digital delkrets

blokkskjema for hele systemet er vist i figur 7.5. I neste kapittel vil evaluering av kretsløsninger og konstruksjons-strategi være tema.



Figur 7.5: Blokkskjema-beskrivelse av fullstendig system

Kapittel 8

Konstruksjons-alternativer

I underkapitlene som følger, har jeg kommet med forslag til forbedringer for de deler som jeg mener trenger dette, eller alternative løsninger som muligens kan være bedre.

8.1 Alternative løsninger for analogdel

8.1.1 Redesign av fotomottaker og alternativer

Fotodetektoren bør legges adskilt fra de andre kretsdelene, så langt fra dem som arealet tillater. Spesielt er det viktig at digitale kretser ligger lengst mulig unna. Spenningsforsyningen bør eventuelt utstyres med kretser for begrensning av transienter og støy. Hele fotomottaker-delen kan skjermes mot støypåvirkning gjennom substratet, såkalt "crosstalk", ved å legge en ekstra "guard-ring" hele veien rundt i tillegg til den som ligger rundt fotodioden.

8.1.2 Redesign av demodulator og alternativer

En sekvensiell PD er bedre egnet enn en XOR-fasedetektor ved lave datarater. Både fordi den har et dobbelt så stort "Lock-in"-område (faseområde), og fordi den er tilnærmet uavhengig av pulsbredden til inngangssignalet. Dersom utgangen fra PD styrer en ladningspumpe, kan frekvenskomponenten som tilsvarer inngangsfrekvensen pluss utgangsfrekvensen unngås. Dette gjelder bare når utgangsfrekvensen er låst til inngangsfrekvensen (under "lock"), men er en stor forbedring i forhold til XOR-porten.

Fase/frekvens-variasjonen blir også mindre; og dersom vi ser bort i fra at kontrollspenningen vil variere noe på grunn av lekkasje ut av ladningspumpe og fra filterkomponentene, går den til null ved låst tilstand. Dette i kontrast til en XOR-PD, som må etterfølges av et filter med nesten uendelig stor dempning over knekkfrekvensen for å oppfylle samme kriterium.

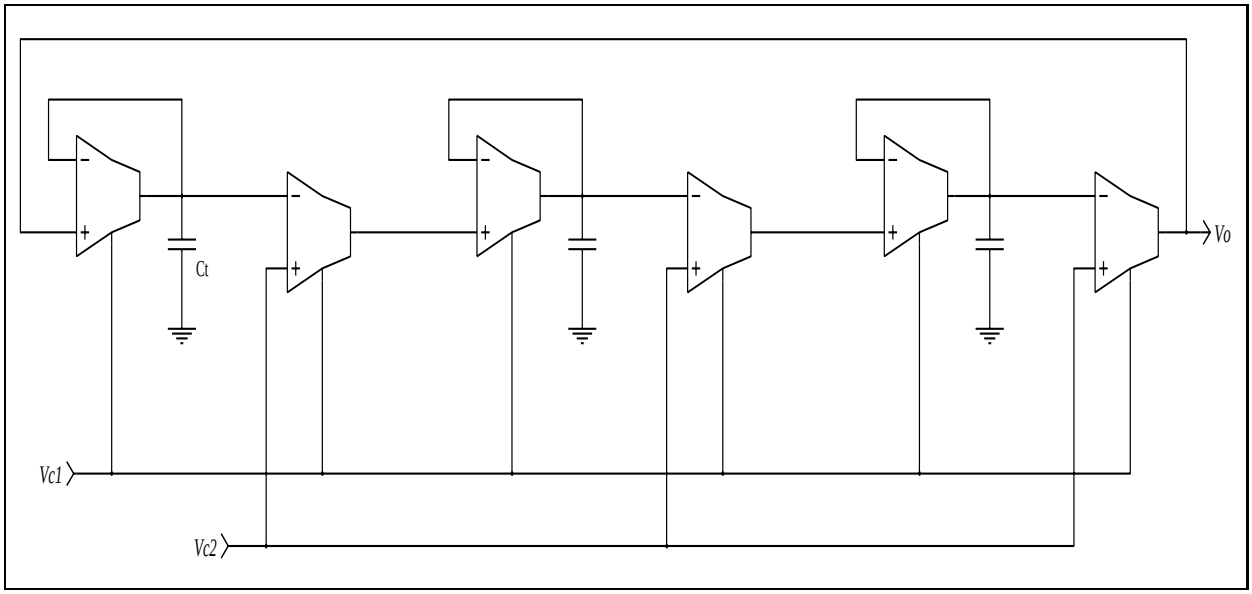
Figur 8.1: PLL med flankestyrt PD, ladningspumpe og S/H-filter

I figur 8.1 har jeg vist et eksempel på en implementasjon som benytter en sekvensiell PD, en ladningspumpe og et "Sample/Hold"-filter. Lagringskondensatoren, C_f, er lagd med en N-type transistor med stort "gate"-areal for å oppnå stor "gate"-kapasitans. Størrelsen på C_f avveies mot hvor mye strøm ladningspumpa kan gi og ta. Dette bestemmes igjen av størrelsen på N- og P-transistoren i pumpekoblingen.

Et stort problem med den realiserte kretsen, er støyen som produseres av VCO. Noe av dette kunne vært unngått dersom VCO ble designet slik at den frittløpende frekvensen lå rundt $4\omega_i$, eller kanskje bare rundt det dobbelte av inngangsfrekvensen. Ved å utvide oscillatoren med flere integrator-inverter-ledd, og øke størrelsen på polykondensatorene kan dette oppnås, selv med det begrensede silisiumareal til rådighet.

La oss si at frittløpende frekvens ligger rundt 1.5MHz for realisert system. Utvides antall ledd fra 5 til 15, vil den ligge rundt 500kHz. Dersom kapasitansverdiene til hvert integratorledd økes fra 675fF til cirka 2pF, skal frittløpende frekvens fra VCO ligge rundt 200kHz. En én-bits teller deler så ned til ønsket frekvens som er halvparten; 100kHz. Som en bieffekt av dette, vil også tilfeldige prossessvariasjoner få mindre betydning, fordi større kondensatorer har mindre prosentvis avvik fra eksakt verdi enn små. Til gjengjeld vil sannsynligheten for en defekt i VCO øke ettersom et større areal forbrukes.

En annen måte å redusere støypåvirkningen fra VCO, er å legge VCO isolert fra andre kretser, eventuelt med skjermende guard-ringer rundt, og skjermet V_{dd}- og GND-tilførsel. Dette kan meget vel kombineres med metoden over, men arealforbruket må avveies mot



Figur 8.2: Alternativ VCO bygd med integratorer og komparatorer

effekten. Et alternativ til inverterne i ringoscillatoren, er å bruke *komparatorer*. Dette er vist i figur 8.2. Både V_{c1} og V_{c2} kan brukes til å modulere utgangs-frekvensen, men det er vanskeligere å få 50% "duty-cycle" på V_o med denne løsningen. Denne løsningen har jeg lagd selv, men lignende løsninger omtales i litteraturen, så den burde fungere i praksis.

8.2 Alternative løsninger for digitaldel

8.2.1 Alternativ klokkestrategi

Problemet med overlappende tofase-klokke, kunne vært løst ved å lage to ikke-overlappende klokker; #phi1 og #phi2 av utgangene fra telleren i PLL-systemet. Andre alternativer er å bruke signaler fra mellomstadier i en ringoscillator-VCO til å generere ikke-overlappende klokkefaser, eller å bruke tre D-vipper med tilbakekobling. Disse teknikkene er vist i figur 8.3 og er tatt fra [2].

8.2.2 Bruk av flanketriggede D-vipper

Realiseringen av logikken ville blitt enklere dersom lagringselementene var flanketriggede D-vipper heller enn latcher. Et lagringselement består av to latcher, klokket av de overlappende klokkefasene #CK og #CKb, i den realiserte kretsen. En overlappende klokke indikerer at å bruke flankene er bedre enn å bruke nivåene.

Figur 8.3: Generering av ikke-overlappende klokkefaser

Med D-vipper ville jeg trengt kún én vippe per lagringselement. Dessuten ville ikke klokkefasen CKb lenger være nødvendig, fordi positiv og negativ flanke i #CK er tilstrekkelig for klokking, forutsatt bruk av både positiv og negativ triggede D-vipper. Generering og distribusjon av invertert klokke ville falle bort, og oversikten over systemet ville bli bedre.

Jeg implementerte digitaldelen utelukkende med standardceller fra et bibliotek[6]. Blant annet for å spare utviklingstid, men også for å redusere konstruksjonsfeil i selve cellene. Jeg fant ingen flanketriggede D-vipper i dette biblioteket med "CLEAR"- eller "RESET"-inngang. Enkle SR-vipper og latcher var det derimot flere av, og derfor valgte jeg å bruke latcher med egen klokkebufring.

8.2.3 Alternativ kontroll- logikk

Tilstandsmaskinen ble realisert som en tilstandsmaskin, men med ett lagringselement (=to latcher) per kontrollsignal. Vanligvis brukes $log_2 N$ stykk lagringselementer for å produsere N kontrollsignaler. For eksempel kunne jeg brukt 3 lagringselementer og en 3:8-dekoder for å lage de 5 kontrollsignalene, og implementert kontroll-logikken som en ASM (AlgorithmicStateMachine).

Dette ville kún gitt en besparelse på to lagringselementer, eller 4 latcher, men testing ville blitt enklere. Eksempelvis kunne jeg plassert MUX'er foran hvert lagringselement, og

dirigert input-verdiene eller tilbakekoblings-verdiene gjennom dem sammen med testlinjer. To test-innganger og en 2:4-dekoder ville ha vært tilstrekkelig for å tvinge tilstandsmaskinen til en hvilken som helst verdi.

Rammelesings-kontrollen kunne like gjerne blitt utstyrt med en rippel-teller, uten at operasjonen ville blitt dårligere. Dette ville spart areal, og gitt en enklere og mer oversiktlig løsning. Dersom glitch'er likevel skulle oppstå, kunne disse likevel blitt fjernet fra rammelesings-signalene, #AL,#BL,#CL og #DL ved å forsinke positiv flanke på #CKb.

8.2.4 "On-chip" adresse

Et framtidig mål er å kunne forhånds-programmere mottager-adressen. En seriell inngang for adressen og en programmerings-linje er tilstrekkelig for å legge adressen i et register eller statisk RAM via et skiftregister. Dette eliminerer behovet for eksterne brytere og mange pinner til adresse. Det burde være uproblematisk å inkludere den ekstra logikken på kretsen, men for å minimalisere løsningen kan man benytte det eksisterende skiftregisteret, "16bsr", og legge til noen ekstra porter for å lede skiftede bits til adresseregisteret eller RAM.

Kapittel 9

Oppsummering

9.1 Tilbakeblikk på kretsløsningen

Jeg er fornøyd med resultatene oppnådd for analogdelen. De har vist at det er mulig å lage en praktisk brukbar løsning med enkelte mindre modifikasjoner på kretsen. Digitaldelen er dessverre ikke blitt testet grundig nok. Jeg vet nå at det er oppstått feil ved selve utlegget av kretsene, men ikke om det er konstruksjonsfeil ved tilføyelse av ekstra logikk for testing som gjør det vanskelig å få fornuftige resultater. Uansett burde jeg ha gjort mer arbeid med digitaldelen med tanke på enkel og oversiktlig testing. Jeg har også en følelse av at selve logikken kunne blitt implementert enklere.

En spesielt verdifull erfaring fra arbeidet med oppgaven, er hvor viktig det er å passe ekstra godt på ved sammenkobling mellom analogdel og digitaldel. På grunn av én feil signaltilkobling i data-gjenvinningsdelen, eller dekoderen, kan ikke hele systemet fungere sammen i operasjonsmodus. En slik simpel feil har her ødelagt utrolig mye. En annen konklusjon som kan trekkes av dette, er at den enkleste løsningen alltid bør prøves først, med mindre man har tungtveiende argumenter i mot dette. Jeg hadde egentlig bare antagelser å bygge på da jeg besluttet ikke å bruke en enkel flanketrigget D-vippe i dekoderen.

9.2 Erfaringer

Arbeidet med oppgaven har vært morsomt, men krevende. Det har gitt meg erfaring med flere disipliner: analog elektronikk, optoelektronikk, digital elektronikk og datakommunikasjon. Jeg hadde stort sett bare innsikt i digital elektronikk på forhånd. Analog elektronikk kjente jeg bare som kretselementer, som for eksempel D/A-konvertere og operasjonsforsterkere, ikke på transistornivå som analog VLSI.

Etter en stund syntes jeg konstruksjon av de analoge delene var mer givende og interessant enn design av de digitale delene. Digital design er i mye større grad enn analog design en automatisert oppgave, der programmer for simulering, automatisk syntese, automatisk testing og lignende reduserer designerens oppgave til å taste input til disse pro-

grammene. Digitale systemer er som oftest mye mer omfattende enn analoge, både i fysisk størrelse/utstrekning og i kompleksitet. I digital design går derfor mye tid med til dobbel- og trippel-sjekking av de delene man til enhver tid har fullført.

Analoge kretser er utsatt for mange feilkilder; de ytre er blant annet begrensninger i måleinstrumenter og støy fra omgivelsene, mens indre feilkilder kan være drift og offset samt intern støy. Men til gjengjeld kan analoge kretser gi langt mer informasjon enn digitale kretser dersom noe går galt. Dette kommer klart frem i evalueringen av PLL-delen, der operasjonen til demodulatoren kan *graderes* ut i fra kvalitative mål. Selv om operasjonen ikke er akkurat som forutsatt eller ønsket, kan vi hente ut informasjon om hvor feilen ligger, hvorfor den ligger der, og eventuelt hvordan den kan korrigeres.

9.3 Verktøy

Jeg har prøvd å holde meg til de verktøy/programmer som brukes regelmessig av kursene in241: "VLSI-konstruksjon", in342: "testing og verifikajon av VLSI", og in347 (in346): "analog VLSI og nevrale nettverk". Disse programmene kan kjøres på de fleste typer maskiner ved IfI, er stabile og oppgraderes/vedlikeholdes jevnlig. Samtidig er det en god del oppdaterte brukerveiledninger som ligger fritt tilgjengelige til disse programverktøyene; se blant annet [7, 8, 9, 10, 11, 12].

Vedlegg A

VLSI-implementasjon i CMOS

Dette vedlegget beskriver strategien bak og arbeidet med selve VLSI-utlegget av kretsen. Dette har kanskje liten interesse for uinvidde innen CMOS-VLSI, men dersom man likevel skulle ønske å sette seg inn i det, kan [11] brukes for å gjøre seg kjent med spesielle ord og uttrykk som jeg ikke har forklart nærmere her.

Delene A.1 til A.5 dreier seg stort sett om arbeidet med utlegg av kretsen og verktøyene som ble brukt. De tjener kun som en referanse dersom man ønsker å gå nærmere inn på selve utleggs-løsningen. Forøvrig kan man hoppe direkte til A.6 som gir en oversikt over kretsorganiseringen.

A.1 Verktøy

Med unntak av enkelte analoge celler, har utlegg blitt foretatt med utleggseditoren MAGIC (se [8]). All *ruting* mellom celler og blokker har blitt foretatt med MAGIC. Programmet har eget kommandovindu ved siden av editeringsvinduet. Kommando-makroer kan bindes til vilkårlige taster. Ekstrahering av utlegg, generering av nettliste og digital simulering kan foregå inne i MAGIC. Generering av nettliste, som blant annet brukes til automatisk ruting, skjer med eget hjelpevindu. Editering av utlegg samt komposisjon og manipulering av celler kan skje i *samme vindu*.

Enkelte celler har blitt lagt ut med "cif"-editoren WOL (se [10]). Dermed har jeg også måttet ta celler ut i "cif"-format fra WOL, for deretter å hente dem inn i MAGIC. Dette innebar en del redigering av "cif"-fila fra WOL, fordi cif-formatet til WOL avviker en del fra cif-formatet til MAGIC. Dessuten er cellene fra WOL laget med *generisk* teknologi, hvilket betyr *generiske lag* som det er opp til prossesshuset å bestemme endelig definisjon til.

A.2 Spesielle problemer

Forskjellen mellom formatene og teknologiene gir problemer med hensyn på brønner og diffusjon, fordi jeg benyttet N-brønn SCMOS-teknologi i MAGIC. Denne teknologien er ikke generisk, men definerer brønner eksplisitt og har egne N- og P-selekt lag for å bestemme diffusjons-type.

Enda større problemer blir det dersom en "cif"-fil fra MAGIC skal leses av WOL. Ikke bare blir det problemer med hensyn på oversetting av N- og P-selekteringslag til generisk selektlag, men WOL er også mer nøye på det tekstmessige oppsettet, eller typesettingen, av "cif"-fila. Blant annet godtas ikke blank/mellomrom mellom "L" og lagnavnet. For eksempel må "L CAA" i MAGIC stå som "LCAA" i WOL. MAGIC ignorerer denne forskjellen.

A.3 "cif"-format i MAGIC og XWOL

Den logiske definisjonen av lag i MAGIC er som følger; all diffusjon er i utgangspunktet "CAA" i cif-kode. Dersom diffusjonen ligger under en boks med cif-kode "SND", eller N-selekt, blir diffusjonen definert som ndiffusjon("ndiff") i MAGIC. Dersom diffusjonen ligger under en "CWN"-boks, eller N-brønn, *og* en "SND"-boks, blir den definert som "nsubstratendiff", eller "nohmic". "nohmic" og "pohmic", eller "nsubstratendiff" og "psubstratepdiff", er materiale for henholdsvis brønn-tilkobling og substrat-tilkobling.

Omvendt forhold får vi dersom diffusjonen ligger under en "SPD"-boks, eller P-selekt, *og* N-brønn ("CWN"). Denne diffusjonen blir definert som "pdiff" i MAGIC. Dersom *bare* "SPD" ligger over diffusjonen, blir den definert som "pohmic".

A.4 Konversjon fra WOL-format til MAGIC-format

Fra XWOL hadde jeg en god del analoge subceller og "pader" på "cif"-format. Disse cif-filene har generisk diffusjon og brønn-lag. Derfor startet jeg opp MAGIC med *generisk* teknologi. Deretter kunne jeg lese inn cif-filene fra XWOL med magic-kommandoen "cif read ⟨.cif-fil⟩". Dermed leses cif-koden inn i MAGIC og omgjøres til *.mag*-format. Lagring av disse filene på .mag-format, gjøres enkelt ved å gi kommandoen "save ⟨.mag-filnavn⟩".

Disse .mag-filene hadde en liten hake: "scgen"-teknologien er konfigurert for *P-brønn* prossess. Hvilket betød at alle P-type lag ble N-type og omvendt. Dette korrigerte jeg i teksteditoren, ved å gå inn på samtlige .mag-filer og bytte om bokstavene p og n i alle lagnavn der disse anga material-type. Deretter startet jeg opp MAGIC med vanlig "scmos"-teknologi (N-brønn), og hentet inn de korrigerte cellene. Ved visuell sjekk kunne jeg fastslå at lagene var kommet riktig, og ingen designregel-feil var blitt introdusert.

I den konfigurasjonen av MAGIC som jeg har brukt, anvendes bare bokser; altså Manhattan-geometri. Alle lag skrives ut på cif-format som bokser fra MAGIC, men cif-polygoner kan leses *inn* i MAGIC. Disse blir omdefinert til mange, mindre bokser som tilnærmer polygon-geomtrien. Resultatet blir et mer "hakkete" utseende på f.eks. bokstaver, slik man kan se det på krets-indentifikasjonen til min chip. En cif-fil med mange og/eller store polygoner, vil derfor resultere i en *større* cif-fil dersom den først leses inn i MAGIC og så skrives ut igjen.

Dersom hele utlegget skulle blitt prossessert i 2um-teknologi, ville dimensjonene til utlegget blitt så store at en tinychip-ramme ville blitt for liten. Derfor ble dette utlegget prossessert i 1.5um-teknologi. Arealmessig fikk jeg med denne løsningen godt armslag. Problemene ble introdusert med ramma. Opprinnelig en "tinychip"-ramme for 2um-prossess og med kun *digitale* I/O-"pader", skalert opp til 1.5um-teknologi.

A.5 Kretsramme og tilkoblingskretser - "pader"

Alle de digitale "padene" i ramma like. Om de skal fungere som inngangs- eller utgangs-buffer, bestemmes av konfigureringen. Dersom funksjons-styreingen *Enable* på "paden" kobles til V_{dd}, fungerer den som output-"pad", der output bufres via et tristate-buffer. Dersom *ENABLE* kobles til GND, fungerer den som input-"pad", der man inne på brikken kan ta input enten fra bufret inngang,eller ubufret. Ubufret input kommer direkte fra pad, med en 150Ohms diffusjons-resistor i mellom som ESD-beskyttelse.

Et utlegg med bare digitale kretser, ville kunne blitt plassert i denne ramma uten noe ekstra arbeid. Men fordi mitt utlegg er en hybrid analog/digital-krets, måtte jeg bytte ut enkelte digitale pader med analoge. Behovet for analoge pader var rundt 10 pinner. "Tinychip"-ramma inneholder 40 "pader"; 4 hjørne-"pader", vanligvis til Vdd og GND, og 36 øvrige "pader" der de fleste er I/O-"pader". Ramma er kvadratisk, og disse 36 er derfor fordelt med 9 stykker langs hver side. Midterste "pad" på hver side er opprinnelig avsatt til spesielle V_{dd}- og GND-"pader", som skal forsynne kretsene *inne på brikken* med +5V og jord. Altså faller 4 av de 36 bort til dette formålet; 2 stykk V_{dd} og 2 stykk GND, og igjen står det 32 "pader" til digital I/O.

Jeg gikk ut i fra at én V_{dd}- og GND-"pad" er nok til å forsyne kretsene inne på brikken, og byttet derfor ut en "$chip V_{dd}$-pad" ut med en digital I/O-"pad". Fordi analogdelen ligger adskilt fra digitaldelen, langs den ene siden av brikken, fant jeg ut at jeg like gjerne kunne bytte ut *alle* "padene" langs én side med analoge "pader". Endelig rammekonfigurasjon er vist i figur A.1.

De analoge bufrings-kretsene er lagt mellom to metall-"pader". Den ene er utilkoblet, mens den andre er tilkoblet integrator-utgangen i pulsformeren, eller "diff1"-kretsen. En ekstern kondensator kan tilkobles denne "paden" for å oppnå en tilstrekkelig stor integrasjonskonstant. Analoge utgangs-buffere er "wide-range" transkonduktans-forsterkere

koblet som spenningsfølgere, med store transistorer i utgangstrinnet. Analoge inngangs-"pader" består av en polysilisium-resistor med dioder til V_{dd} og GND som ESD-beskyttelse.

A.5.1 Skalering av analoge "pader"

For å få de analoge "padene" til å passe inn i ramma, samt overholde gitte betingelser satt av prossess-huset, måtte jeg skalere dem opp. I 2um-teknologi er *lambda*=1um. I 1.5um-teknologi er *lambda*=0.8um.

$$\text{Skaleringsfaktor: } \frac{1um}{0.8um} = \underline{1.25}$$

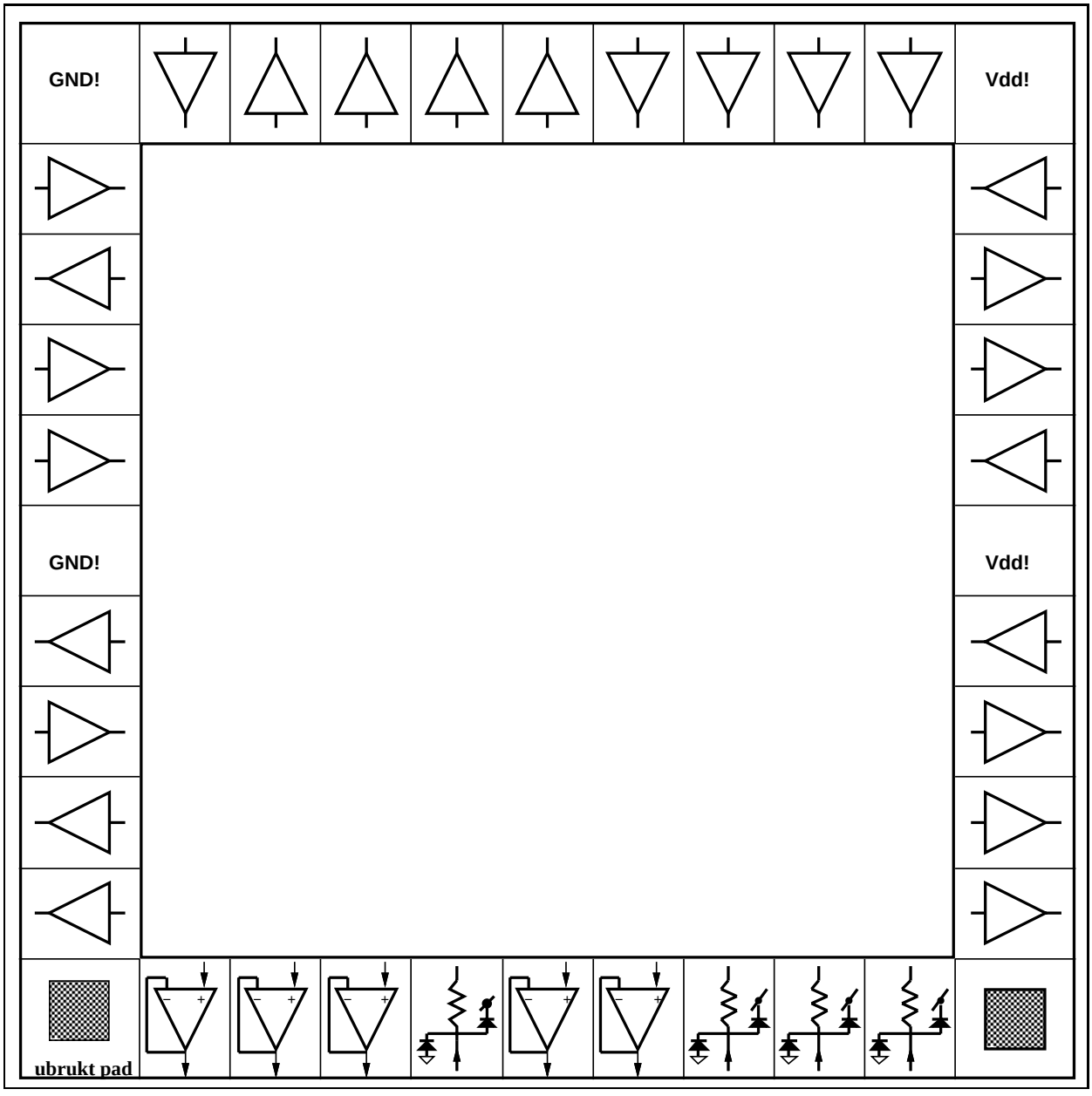
En oppnår altså en 25% større integrasjonsgrad dersom utleggsreglene forblir de samme. Men for at selve krets-ramma skal ha de samme dimensjonene som i utgangspunktet, må den skaleres opp med en faktor på 1.25. Som et eksempel kan vi ta "bonde-padene". De skal i følge fabrikanten være $100 \cdot 100um$ $(0.1 \cdot 0.1mm)$. I 1.5um-teknologi betyr dette at utvendige mål skal være $125 \cdot 125lambda$.

Etter å ha lest inn .mag-filene for de analoge "padene" i MAGIC og lagret dem i "cif"-format igjen, leste jeg cif-koden inn igjen med "cif-inputstyle" satt til 0.8. Dette betyr at $1lambda$ i "cif"-fila leses inn som 0.8 *lambda*. Opprinnelig er *lambda* i "cif-inputstyle" satt til 1.0, slik at et utlegg som leses som en "cif"-fil får uforandrede dimensjoner i MAGIC. Med *lambda*-inn=0.8, blir utlegget krympet med en faktor på 0.8.

Neste skritt er at jeg setter lambda i "cif-outputstyle" til 1.5, altså *lambda*-ut=1.5, og skriver utlegget ut på nytt som cif-fil uten å ha gjort noe med det. $1.5 \cdot 0.8 = 1.2$,som er det nærmeste jeg kommer faktoren 1.25.

Når jeg så leste inn "cif"-koden for siste gang, var lambda i "cif-inputstyle" resatt til 1.0, slik at geometrien ikke ble endret. Cellene er med denne operasjonen skalert opp med en faktor på $\underline{1.2}$. For å få eksakt riktige mål i *lambda*, måtte jeg editere resten for hånd. Det vil si "klatte" på litt metall i ytterkantene av cellene. Samtidig rettet jeg opp de bruddene på designreglene som eventuelt hadde oppstått i løpet av konversjonen og skaleringen. For eksempel geometrier som hadde kommet for nær hverandre.

Under denne editeringen kan jeg selvsagt ha gjort feil, men jeg sjekket de ferdig editerte cellene mot de opprinnelige cellene i WOL, og er temmelig sikker på at dette ikke er tilfelle. Dersom både skalerte celler i MAGIC og opprinnelige celler i WOL hadde blitt ekstrahert og sammenlignet med programmet NETCMP, ville jeg ha hatt ytterligere ryggdekning. Men på grunn av den relativt lave kompleksiteten i cellene, mente jeg at visuell sjekking var tilstrekkelig.



Figur A.1: Skjema for pad-konfigurering

A.6 Floorplan-løsning

Floorplan-løsning for systemet er relativt konvensjonell. Alle celler og blokker er lagt ut med samme orientering i planet, og sammenfallende med orienteringen til ramma. System-blokker er i hovedsak satt sammen *vertikalt*. Det vil si at jeg har bygd opp blokkene så *brede* som jeg fant forsvarlig ut i fra lengder på signal-linjer. Blokker er satt sammen til så brede *moduler* som er mulig ut i fra begrensninger som tilgjengelig areal og funksjonen til modulen. Alle standard-cellene fra biblioteket (se [6]) har lik høyde, $H = 75lambda$, slik at de fleste systemblokkene også har samme høyde.

Data-registeret,"16bsr", og signatur(sjekksum)-registeret,"SAR", er plassert i hver sin ende av den digitale delen. Dette har jeg gjort fordi data-linjene fra signatur-registeret ikke brukes av kontroll-logikken. Med denne plasseringen unngår jeg at signal-linjer fra "SAR" blandes sammen med signal-linjer fra "16bsr", og utlegget blir mer oversiktlig. Dersom jeg ønsker å gjøre en endring av feilsjekk-logikken, for eksempel endre sjekksum-polynomet, er dette lettere dersom den ligger i periferien av utlegget.

Kontrolldelen er lagt sentralt i utlegget for å gi kort gjennomsnittlig veilengde for kontrollsignalene. Kombinatorisk logikk som forsyner kontrolldelen med inngangsverdier, for eksempel adresse-sjekker og kommando-dekoder, er lagt rundt kontrolldelen. Disse blokkene henter sin input fra databus'en, og er plassert kompakt for å krympe arealet. Her fikk jeg de største problemene med hensyn på ruting, fordi det ble vanskelig å rute igjennom små blokker plassert over hverandre. Det ville vært en fordel å sette flere mindre blokker sammen til større moduler dersom ruting var absolutt kritisk. Men ut i fra blokkenes funksjon og tilgjengelig plass, fant jeg det lønnsomt å plassere dem desentralisert.

Timer-modulen er akkurat som signatur-registeret plassert ute på siden av utlegget. Tilgjengelig plass rundt denne modulen er det lite av, og derfor er det ikke store endringer til på den før andre moduler/blokker må flyttes. Dersom bare signatur-registeret, "SAR", flyttes nedover blir dette noe enklere.

De analoge delkretsene ble lagt så langt i fra digitale kretser og I/O-"pader" som mulig, men uten ekstra beskyttelse som for eksempel "guard"-ringer. VCO plasserte jeg i motsatt ende av analogdelen i forhold til fotodioden. Analogdelen ble beskyttet mot lyspåvirkning ved å legge et metall2-lag over hele delkretsen. Selvsagt med unntak av selve fotodioden. Fotodioden ble laget kvadratisk, med sidelengder på $360 \times 360\lambda$.

A.7 Rutings-løsning

Den vertikale floorplan-løsningen betyr at V_{dd} - og GND -linjer blir lange. Dette kan igjen bety at spenningsdropp kan bli et problem. Dette er i tilfelle et problem som vil bli mer framtredende dersom klokke-hastigheten økes. Ved en klokkefrekvens under 1MHz, anser

jeg problemet som lite. Dersom alternativet er å gjøre signal-linjer lengre, blir løsningen dårligere. Lengre signal-linjer fanger opp mer støy.

Det første punktet i rutings-strategien, er hvordan V_{dd} - og GND -forsyning skal legges. En vanlig løsning er en trestruktur, der greinene griper inn i hverandre (se figur under). Jeg har lagt opp rutingen slik at utlegget bare har ett V_{dd} -tre og ett GND -tre. "Stammen" til V_{dd} -treet er en bred, vertikal metall2-leder langs den siden av utlegget der V_{dd}-"padene" ligger. Tilsvarende ligger GND -forsyningen langs den andre siden av utlegget. Denne løsningen gir få krysninger mellom V_{dd} - og GND -linjer. Helst burde det ikke forekommet krysninger inne på substratet overhodet, men fordi jeg måtte flytte blokker *etter* at jeg påbegynte rutingen, oppsto det problemer som enklest ble løst ved å la V_{dd} - og GND -linjer krysse hverandre.

Lærdom som kan trekkes av dette, er at det er lurt å sette av god plass mellom blokkene på forhånd, deretter legge ut rutingen med minst mulig vinkler, og *så* vurdere om det er nødvendig, enn si mulig, å optimalisere blokkplasseringen og rutingen. Dette gjelder ikke nødvendigvis meget regulære utlegg, der rutingen for eksempel kan integreres i makrocellene.

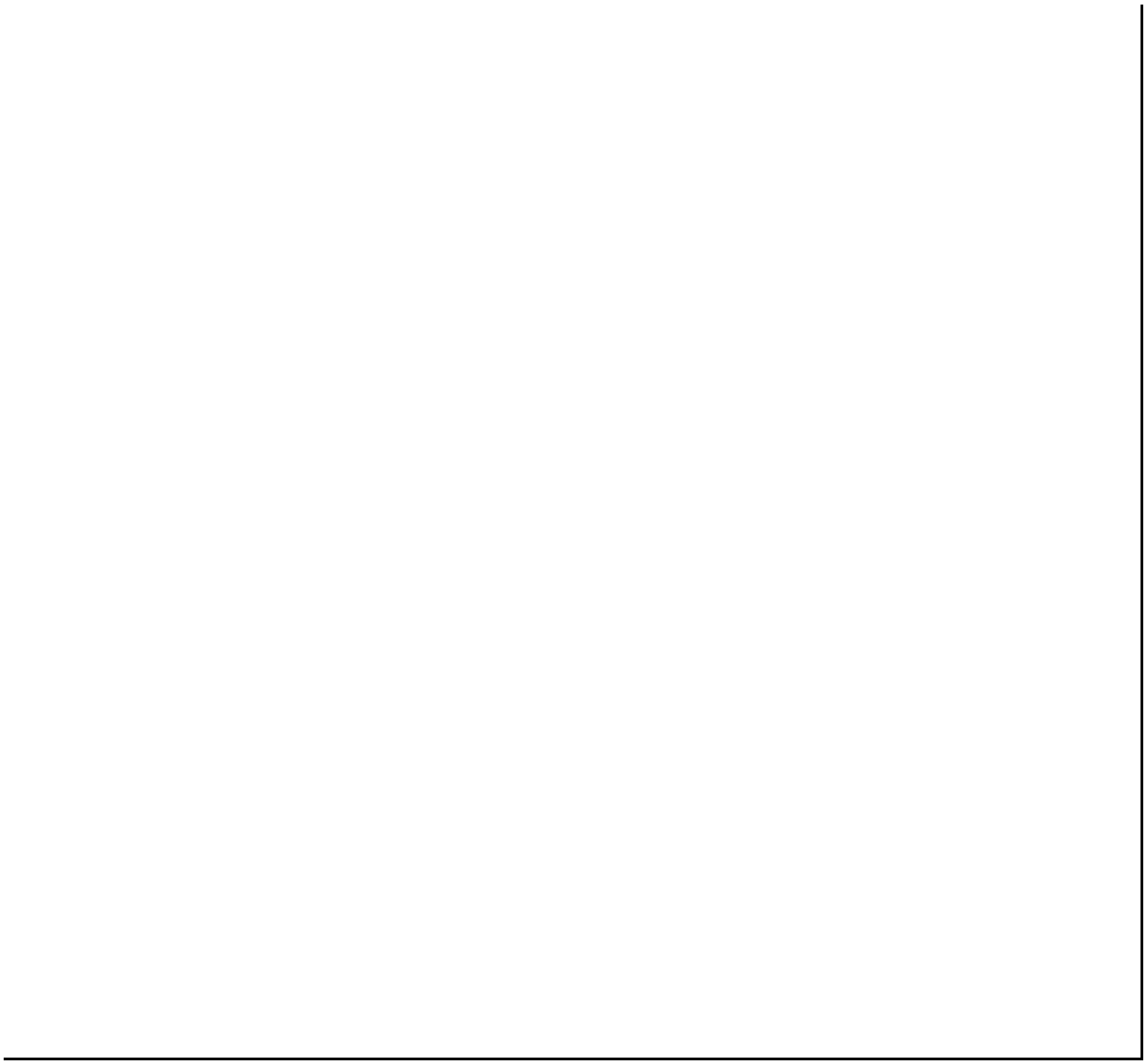
A.8 Testing og måling

Ekstra logikk er plassert på chip'en for å gi mulighet for testing av kretsen. MUX'er plassert i signalveiene gjør det mulig å påtrykke egne testvektorer og testsekvenser for testing av digitaldelen. Også klokkelinjene går via MUX'er, og det er derfor mulig å teste analogdel og digitaldel separat ved å påtrykke en ekstern klokke. Testlogikk og testlinjer er tegnet i figur A.2. Signaler fra analogdelen ut til pinner som bufres via digital I/O-"pad", er ikke tatt med i figur A.2.

Analogdelen kan i liten grad styres eksternt, men til gjengjeld er mulighetene for å observere signalveien flere. Målepunkter er tatt ut for V_p, rett fra fotodioden (pinne#11), og fra utgangen på "diff1"-kretsen (pinne#10). Utgangen på V_i-bufferet kan i tillegg observeres via en digital "output-pad" (pinne#4). Dette kan godt sies å være overflødig, men min filosofi er at jo tidligere en feil oppstår i signalveien, jo vanskeligere er den å oppdage. For ikke å si hvor ødeleggende den er for systemets funksjon.

Utgangen på pulsdetektoren, #STARTCK, kan observeres på pinne#6, og dessuten kan både utgangsspenningen fra integratoren og kontrollspenning, Vc, ut fra tilpasning til VCO, observeres på henholdsvis pinne#8 og pinne#7. Signalet fra VCO tas ut til observasjon *før* det deles ned til riktig frekvens. Dette gir bedre oppløsning og gjør det lettere å sammenlikne flankene på innkommende signal med oscillatorens. VCO-output er tatt ut med en digital output-pad (pinne#3).

Figur A.2: Testlogikk og testsignaler for digitaldelen



Figur A.3: Testlinjer til og fra analogdel

Analoge innganger for testing er det 3 stykker av. Dette er eksklusive en "bias-pad" for forspenning til "wide-range" bufferne (pinne#12). Det er analoge input-pader for innstilling av faktorene A (pinne#14) og G (pinne#13) i "diff1"-kretsen. Spenningen benevnt A er forsterkning i komparatoren, og bestemmer hvor fort utgangen kan slå om. Spenningen G styrer transkonduktansen til integratoren, og bestemmer hvor lang en puls blir, altså "duty-cycle" på pulsene.

Den siste analoge "input-paden" (pinne#9), gir bias-spenning til transkonduktans-forsterkeren i integratoren, for å stille knekkfrekvensen eksternt. Egentlig ønsket jeg at knekkfrekvensen skulle være satt på forhånd med en spenningsreferanse, men på grunn av mange usikre faktorer bestemte jeg at dette foreløpig skulle være en "skrue", altså en ekstern spenning, for å gi større fleksibilitet.

A.9 Evaluering og forslag til forbedringer

Etter at jeg begynte å måle på kretsen, frant jeg ut at det var flere detaljer som kunne forenklet testing og måling samt gjort dette mer nøyaktig. En tommelfinger-regel er å legge til logikk eller innganger som setter kretsen i en gitt (kjent) oppstart-tilstand. Dersom intern logikk ikke legges til kretsen, bør et eller flere eksterne signaler føres inn til logikken. Dette er typisk en *RESET*-pinne, som blanker alle transparente registre, resetter eller setter vipper og tvinger tilstandsmaskiner og sekvensgeneratorer til en gitt tilstand. En slik inngang bør være essensiell på enhver prototyp-chip, men dessverre tok jeg det ikke med i systemet.

Analogdelen er skilt fra digitaldelen dersom testpinnen #TSEL=1. Klokke og data generert i PLL er da uten betydning for digitaldelen. Men dette betyr ikke at analogdelen ikke kan testes samtidig. En mangel ved kretsen, er at PLL ikke kan tilføres et RZ-kodet inngangssignal eksternt. Et slikt signal, enten indentisk med testklokke-signalet eller tilført via en digital inngangs-"pad", kan manipuleres langt friere enn signalet fra fotodetektoren, som med den eksisterende løsningen er det eneste referanse-inngangssignalet til PLL.

Uten lyspulser vil pulsdetektorens utgang ligge til jord, og oscillatorens operasjon er *bare* avhengig av de digitale signalene #STARCK og #STOPCK. Av disse er det bare #STARTCK som kan kontrolleres, fordi det kan erstattes med testsignalet #TSTARTCK, og dermed har bare #STOPCK betydning. Det viser seg at dette signalet ligger lavt ved oppstart, og derfor igangsettes oscillatoren ved hver oppstart. Ideélt sett, burde oscillatoren være avslått ved oppstart, slik at det var mulig å teste pulsdetektoren enklest mulig. Dersom oscillatoren startet etter en lang lyspuls (med romslig margin), ville denne delen være verifisert omgående. Hvis ikke, kunne den avskrives og start/stopp av VCO kunne kontrolleres av #TSTARTCK. Som før nevnt er derfor en *RESET*-pinne essensiell i en prototyp, for å sette kretsen i en gitt utgangstilstand før testing.

Analog og digital V_{dd} og GND burde vært fysisk skilt fra hverandre på brikken, på grunn av problemer med "crosstalk" og spenningstransienter. Dette er problemer som jeg var klar over på forhånd. Men fordi dette ville ta opp minst to pinner til, lot jeg likevel digitaldel og analogdel ha felles V_{dd} og GND. Min forhåndsvurdering var at klokking av digitalelementene *ikke* ville gi store transienter eller andre problemer fordi klokkefrekvensen er lav. Dette holdt bare delvis, og ulempene med hensyn på analogdelens funksjon, og ikke minst målinger på den, ville rettferdiggjort de to ekstra pinnene. Alternativ "floorplan", med adskilt spenningsforsyning og "guard"-ringer rundt analogdelen, er vist i figur A.4.

Disse pinnene kunne blitt frigjort ved å gjøre et mer kresent utvalg av signaler for uttak. Uttak av hele *fem* ledd i skiftregisteret er dårlig pinneøkonomi, selv om dette er de viktigste databit. En riktigere fordeling ville vært det laveste og det høyeste bitet i skiftregisteret, b0 og b15. Dette ville frigjort tre pinner, hvorav to kan brukes til spenningsforsyning og jord til analogdel, og siste blir en *RESET*-pinne.

Med en slik omfordeling ville digitaldelen alene bruke 4 "pader" eller pinner til spenningsforsyning og jord. Ved å omkonstruere hjørnepadene, til pinne#25 og pinne#35, slik at de forsynte både pader og brikke med V_{dd} og GND, kunne pinne#20 og pinne#40 frigjøres til andre formål. For eksempel ville det være fornuftig å ta ut signalet fra flagg-detektoren, "fd", og rammeslutt-markøren #DL til observasjon på disse to pinnene.

Et alternativ til faste inn/ut-pader, ved å sette *ENABLE* til en fast verdi, er å bruke en alternerende konfigurasjon. Hvilket betyr at *ENABLE* brukes som et styringssignal. Slik kan én enkelt digital I/O-"pad" brukes *både* som utgang og som inngang, avhengig av hvilken modus kretsen er i.

Dersom brikken er i observasjonsmodus, eller *målemodus*, er uttak av interne signaler til observasjon det viktigste, og alle styrbare pader settes opp som utganger. Samtidig settes en MUX etter hver pad til å velge kanal M, slik at de observerte signalene går videre på brikken til de blokkene de skal styre.

I testmodus settes hver pad opp som inngang, fordi testvektorer skal påtrykkes kretsen. Samtidig settes MUX'en til å velge kanal T, slik at testsignalene erstatter de interne signalene. *ENABLE* brukes da til å styre både MUX'er og "pader". Altså tilsvarer ENABLE-signalet #TSEL i mitt opprinnelige design. OBS! Det er verdt å merke seg at noen pader likevel settes opp som fast inngang eller utgang. Faste innganger er for eksempel adressepinnene, og faste utganger tar ut signaler som bare skal observeres. Dette gjelder spesielt signaler fra sluttrinn i signalveien, eksempelvis #STT.

A.10 Oppsummering

I ettertid ser jeg mange punkter i VLSI-realiseringen som kunne blitt gjort bedre. Dette gjelder både arbeidet med utlegg av kretsene, og strategien bak konstruksjon av systemet

Figur A.4: Alternativt “floorplan” med skilte spenningsforsyninger

med tanke på testing og verifikasjon. Slik vil det nesten alltid være, men enkelte ting bør nevnes for eventuelt videre arbeid med systemet:

Utleggene av alle delkretsene bør ekstraheres og simuleres. Digitaldelen bør ekstraheres og simuleres med en digitalsimulator, analog simulering er antagelig ikke strengt nødvendig. Digital simulering på ekstraherte kretser ble ikke foretatt, fordi nytt simulator-program ble introdusert samtidig med utleggs-arbeidet, og eksisterende simulator kunne ikke brukes sammen med ny versjon av utleggs-programmet.

Nettliste bør ekstraheres for alle delkretser, og sammenlignes med nettliste fra skjematisk krets, helt opp til øverste nivå i hierarkiet. Verifikasjon av utlegg på denne måten ble gjort med analogdelen, men ikke på digitaldel.

Testlinjene er ikke tatt ut med stor nok omhu, derfor bør de revurderes. Videre bør testlogikken konstrueres annerledes. Analoge og digitale testlinjer bør isoleres fra hverandre. Øvrige svake punkter i konstruksjonen har jeg omtalt tidligere, og berører først og fremst selve konstruksjonen av delkretsene. Det gjenstår også mye arbeid med måling på kretsen, men gode nok resultater foreligger til å konstruere et bedre system på grunnlag av disse.

Referanser

[1] Tor Sverre Lande, Morten Salamonsen, Yngvar Berg.
Universitetet i Oslo.
Per Olaf Pahr.
Tandberg Data A/S.
Photosensors for CMOS Pulsed Light Detectors
Proceedings - Eighteenth European Solid State Circuits Conference ESSIRC'92.

[2] M. R. Haskard I. C. May.
Editor: K. Eshraghian.
Analog VLSI Design
Prentice Hall 1988.

[3] C. Mead.
Analog VLSI and Neural Systems.
Addison Wesley 1989.

[4] D. H. Wolaver.
Phase-Locked Loop Circuit Design.
Prentice Hall 1991.

[5] G. Russell I. L. Sayers.
Advanced Simulation and Test Methodologies.
Van Nostrand Reinhold International 1989.

[6] Institute for Technology Development,
Advanced Microelectronics Division 1990.
Scalable CMOS (SCMOS) Standard Cell Library.

[7] Brukermanual for LOG (AnaLOG og DigLOG):
LOG manual.

[8] J. O. Ousterhout (kp.7)
University of California Berkeley.
M. Chow M. Horowitz (kp.8-9)
Stanford University.
W. Scott M. Arnold (kp.10-11)
Lawrence Livermore National Laboratory.
Magic version 6 Tutorial.

[9] A. Vladimirescu K. Zhang A. R. Newton D. O. Pederson A. Sangiovanni
University of California Berkeley.
SPICE User's Guide.

[10] M. A. Silviotti 1990.
NEWOL User's Guide.

[11] Terje Knudsen,
Institutt for informatikk, UiO, 1991.
Programverktøy for simulering og utlegg av VLSI-kretser.

[12] Terje Knudsen,
Institutt for informatikk, UiO, 1991.
Laboratorieinstrumenter og omgivelsesprogrammer.