Spesifikasjon og design av et utlesningssystem for et LHC-eksperiment ved CERN.

Pål Even Gaarder

11. november 1994

Hovedfagsoppgave til Cand. Scient. graden ved Universitet i Oslo, Fysisk Institutt.

Forord.

Denne avhandlingen er en besvarelse til graden *Cand. Scient.* ved Universitetet i Oslo. Mine veiledere har vært Professor Torleiv Buran og NFR forsker Dr. Steinar Stapnes.

Jeg har iløpet av dette studiet verdsatt verdien av Universiteter som grunnforskningsinstitusjoner der politisk eller økonomisk påvikning minimaliseres. Jeg håper denne situasjonen kan opprettеholdes slik at samfunnet i fremtiden også kan utvikles fra denne kreative grunnstein, grunnforskning, som velstanden i dag bygger på.

Jeg vil også, til slutt, nevne at deler av avhandlingen er gjort på bakgrunn av informasjon fra, eller i sammarbeid med, andre studenter og ansatte ved UiO som er knyttet til LHC, da spessielt Steinar Stapnes.

Pål Even Gaarder

Sammendrag

Avhandlingen er i sin helhet å betrakte som et forprosjekt. Dette pga. at LHC prosjektet har vært under utvikling parallelt med denne avhandlingen og at det fortsatt kommer til å bli foretatt flere iterasjoner før den endelige løsning foreligger. Dette er også den første avhandlingen skrevet ved elementærpartikkel-fysikk gruppen ved UiO om LHC prosjektet. Det gjør nødvendigvis denne avhandlingen noe mere omstendelig enn den ville vært om det på forhånd hadde eksistert beskrivelser av problemstillingen.

Hensikten med avhandlingen er å beskrive og gi en forståelse av hva som kreves av et datainnsamlingssystem for et LHC eksperiment, spesielt ATLAS eksperimentet.

Avhandlingen består av to deler:

1. Betraktninger om problemet og hvordan dette kan løses. I avhandlingen er innerdetektoren og dennes silisiumstripe-baserte moduler brukt som en representativ del for DAQ systemet. Dette er gjort for å begrense omfanget og fordi det er denne detektordelen utviklingen i Oslo er rettet mot.

2. Beregninger og simuleringer med etterfølgende diskusjon av hvordan noen av løsningene vil kunne fungere.

Konklusjonen i avhandlingen er at en har en god forståelse av eksperimentets synkrone operasjon, men det vil være viktig å ha en bedre forståelse av de asynkrone operasjonene og korrelasjoner mhp. tap av data. En bedre forståelse av den asynkrone delen vil gjøre at en har større forståelse av eksperimentets problem med å lese ut all data komplett.

Vedlagt er :

A En ordliste som knytter den norske ordbruken i denne avhandlingen til den mere brukte internasjonale, samtidig som det er en forklaring av forkortelser som er brukt.

B Et utlegg av den siste versjonen av innerdetektoren til ATLAS før denne avhandlingen ble trykt.

D Brukerbeskrivelse av programmene som er laget i forbindelse med denne avhandlingen, og hvordan de er brukt i denne avhandlingen. Jeg vil tro det vil være av stor nytte å lese dette vedlegget parallelt med kapittel 4 og 5.

– Til slutt er det en liste over literatur som inneholder bakgrunnsstoff og tilleggsmateriale.

Innhold

Kapittel 1

The Large Hadron Collider, LHC.

The Large Hadron Collider (LHC), vil bli den neste proton-proton og tungjone akseleratoren ved CERN.
For å kunne identifisere de minste bestanddeler som naturen er bygd opp av og hvordan de vekselvirker, trenges høy konsentrasjon av energi. Dette gjøres ved å bruke partikkel-akseleratoren. Ved LHC vil hver av de kolliderende protonene ha en energi på 7 TeV[1]. 14 TeV vil derfor konsentreres innen et volum på $\approx 10^{-45} m^3$. Protonene har en indre struktur av kvarker og gluoner, og energien til protonet er fordelt på kvarkene og gluonene.
De kraftigste akseleratorer i dag er:

- CERN LEP elektron-positron akselerator som i 1995-96 blir oppgradert til 170-180 GeV i massesenter systemet.
- Tevatron proton-antiproton akseleratoren til Fermilab med 1.8 TeV energi i massesenter systemet som gir ca. 200 GeV på kvark/gluon nivå.

De reaksjoner en ser etter ved disse energiene, vil være meget sjeldne, dvs. ha lite virkningstverrsnitt σ, for produksjon av f.eks. Higgs partikler, tunge vektor bosoner osv. Det er derfor viktig å ha høy luminositet $\mathcal{L}$:

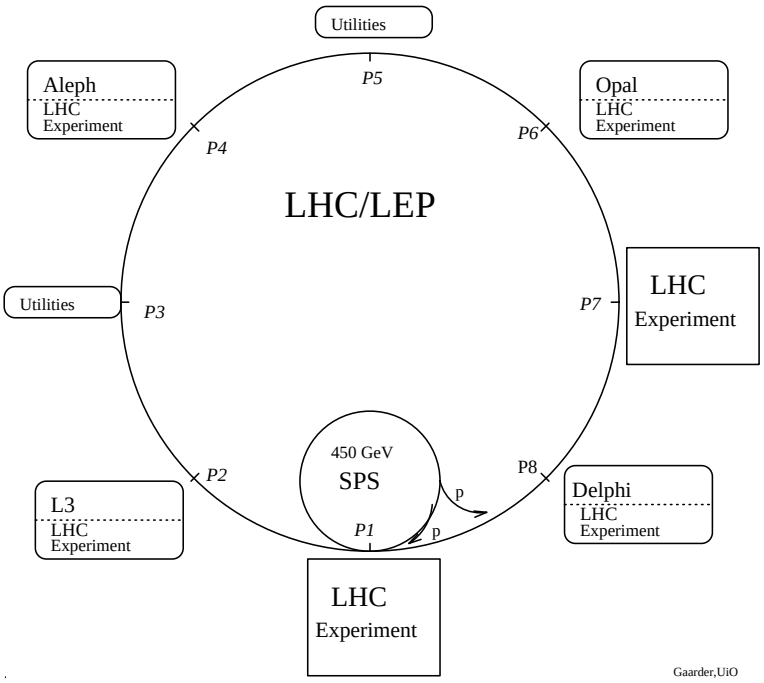
$$\mathcal{L} = f\frac{N_1 N_2}{A}, \tag{1.1}$$

hvor f er frekvensen, N partikler i hver pakke og $A = 4\pi\sigma_1\sigma_2$ tverrsnittet for partikkelstrålene i kollisjonspunktet. Noen av verdiene er gitt i tabell 1.1. Disse ekstreme kravene gjør at det kreves utstrakt teknologisk forskning og nyutvikling for at LHC skal kunne settes i drift rett etter år 2000 slik som planlagt, [31], [3], [48].

1.1 Akselerator-spesifikasjonene.

Ved utgravingen av tunnelen for LEP CERN Genev, ble det planlagt at det senere skulle være mulig å installere en proton-proton akselerator i samme tunnelen. LHC vil derfor bli installert i samme tunnel som LEP. Figur 1.1 viser akseleratoren og de forskjellige eksperimenter som er foreslått. Ved $p2, p4, p6, p8$ er det allerede eksisterende LEP eksperimenter.

[1] $1 TeV = 1.6 \cdot 10^{-7} J$.



Figur 1.1: *De forskjellige eksperimenter ved LEP/LHC tunnelen.*

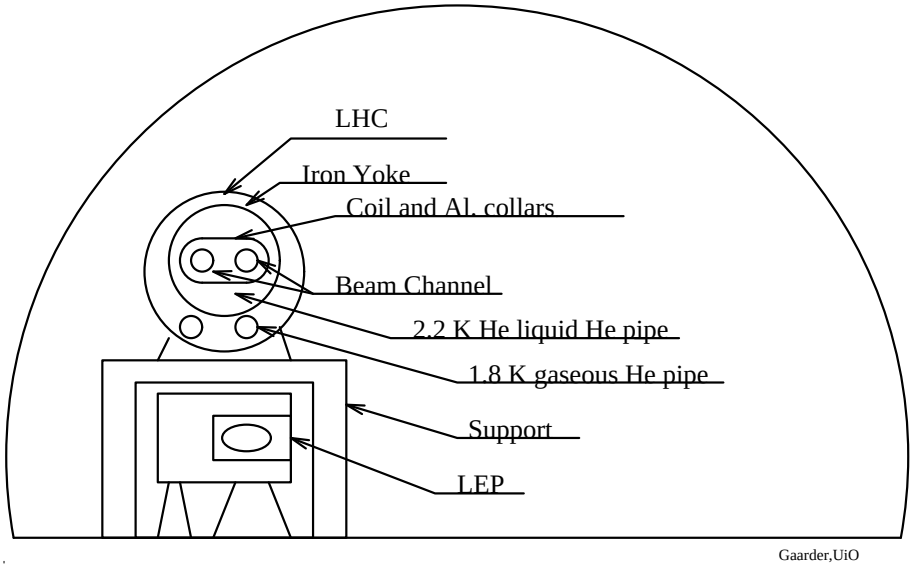
Protonene vil i to adskilte rør bli akselerert i motsatt retning til hverandre. For å avbøye og holde protonene i banen benyttes supraledende dipolmagneter. Figur 1.2 viser magnetene og de to strålerørene montert på topppen av LEP-magnetene. Avbøyningsmagnetene vil gi et 9 Tesla magnetfelt som gjør at en kan holde på protoner med en relativistisk masse som vil gi en energi på opptil 14 TeV i p-p kollisjonene.

LHC vil ha en forholdsvis høy luminositet sammenlignet med andre foreslåtte akseleratorer av samme typen. Dette for å øke sannsynligheten for å kunne produsere de ønskede hendelser ved den allerede gitte radius for LHC og den begrensningen en har i styrken på magnetfeltet. Tabell 1.1 gir noen hovedparametere for LHC maskinen [27].

1.2 Fysikk og den eksperimentelle målsetningen.

Noen av grunnene til at en velger å akselerere protoner istedenfor elektroner er:

- strålingstapet i akseleratoren øker omvendt proporsjonalt med massen av den akselererte partikkel opphøyd i 4. potens,
- det gjør det mulig å studere *parton-parton* kollisjoner,
- det gjør at en vil få en simultan studie over et bredt energibånd, som gjør at en ikke trenger å tune akseleratoren til en bestemt resonansenergi,
- en vil ofte ved hadron-hadron reaksjoner få et økt tverrsnitt pga. QCD produksjon, som gjør at hendelsesraten for en interessant reaksjon vil øke,



Figur 1.2: *LHC akseleratoren i LEP tunnelen.*

		p-p	**pb-Ions**
Expected operational energy	TeV	14.0	1142
Dipole field (max)	T	8.65(9.0)	8.6
Luminosity	$cm^{-2}s^{-1}$	$\sim 10^{34}$	$\sim 10^{27}$
Number of bunches		2835	560
Bunch spacing	m/ ns	7.5/25	40.5/135
Particles per bunch		10^{11}	$9.4 * 10^{10}$
Particles per beam		$2.8 * 10^{14}$	$5.3 * 10^{10}$
Number of experiments (detectors)		2	1(+1)
β^* at interaction point	m	0.5	0.5
r.m.s. radius at interaction point	μm	16	15
r.m.s. collision lenght	mm	53	53
Crossing angle	μrad	200	< 100
Interbeam distance	mm	180	
Max. cable current	A	8800	
Synchrotron radiation (2 beams)	kW	7.4	
Luminosity at beam-beam limit	$cm^{-2}s^{-1}$	$2.5 * 10^{34}$	

Tabell 1.1: *LHC Main Parameters (May 1993), 40 MHz.*

- en vil få et større antall initial tilstander, f.eks. direkte produksjon av W.

Hadron maskiner har også enkelte ulemper som,:

- andelen av massefart (*scaling variabelen* $x = \frac{partonmassefart}{protonmassefart}$) for partonene som vekselvirker vil generelt være ulike ($x_1 \neq x_2$). Dette gjør at massesenteret ikke vil være i ro i laboratoriesystemet, og en må beregne massen for resonansen ved invariant masseberegning som stiller større krav til detektorens oppløsning,

- en vil få store tverrsnitt for uinteressante hendelser (bakgrunn),

- i hadron akseleratorer vil en ofte ha flere kollisjoner for hver gang partikkelbuntene møtes inne i detektoren,

  (1.2) $$\bar{n} = \mathcal{L} * \sigma_{inel} * \Delta t$$

  og gir ca. 18 hendelser pr. kollisjon, ved $\Delta t = 25ns, \sigma_{inel} = 70mb, \mathcal{L} = 10^{34} cm^{-2} s^{-1}$.

LHC vil bli det første instrumentet som gir masser på over $1TeV/c^2$ direkte tilgjengelig for eksperimenter. Dette vil åpne muligheten for å utforske dagens fysiske teorier og gi svar på mere spekulative teorier. Et viktig mål er derfor å bygge en fleksibel maskin som kan gi svar på dagens problemer og som kan tilpasses fysiske problemstillinger i årene fremover, [21] [22].

1.2.1 Standardmodell studier.

Det er fortsatt mange ubesvarte spøsmål i Standard Modellen, SM, som:

- opphavet til masse,

- symmetribrudd,

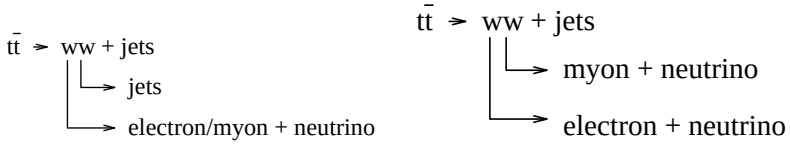
- forståelse av generasjons-hierarkiet.

Søk etter Higgs boson og undersøkelse av Higgs-mekanismen, vil gi svar på mange av disse spørsmålene. Andre viktige områder for å kunne utdype SM er test av elektrosvak gauge-koblinger og studier av topp-kvark systemer. En vil også gå utover SM som å forsøke å finne kvarksammensetninger, supersymmetri og evt. nye eksotiske fysiske fenomen.

1.2.2 Topp-kvark produksjon.

I Standard Modellen er t kvarken krevd som den svake isospinn partneren til b kvarken. Ved LHC vil en søke etter m_{topp} og ikke-standard topp henfall. Ved Fermilab Tevatron er det funnet 12 kandidater til $t\bar{t}$ som har gitt en t masse på $174 \pm 10^{+13}_{-12} GeV/c^2$ [25]. Tidligere eksperimenter begrennser m_{topp} til $169 \pm 20 GeV/c^2$ [46]. For $m_{topp} = 130 GeV/c^2$ og $\sigma_{t\bar{t}} = 3$ nb, finner en fra formel 1.2 at LHC vil kunne produsere $\approx 3 * 10^8$ t henfall i løpet av 10^7s. Typisk topp kvark henfall vil være $t\bar{t} \rightarrow wwqq$, der kvarkene q normalt vil være b.

Mulige t henfall:



En vil også se etter $t \to bH^+$ henfall (kapittel 1.2.4). Rekonstruksjon av hadronisk topp kvark henfall baserer seg på følgende seleksjonskriteria:

- Identifikasjon av leptoner med $p_T > 40$ GeV,
- Minst 3 jet-strukturer innenfor $\eta < 2$, $p_T^1 > 50$ GeV og $p_T^{2,3} > 40$ GeV i hemisfæren motsatt til leptonet,
- b-jet identifikasjon, der minst ett ladet spor har $P_T > 2$GeV og impact parameteren $> 200\mu m$.

Beregning av massen til topp kvarken.
Ved $t \to jjb$ vil en først rekonstruere at det var en b kvark. Identifikasjon av b-kvarken vil føre til en kraftig reduksjon i bakgrunn som igjen vil føre til reduksjon av systematiske feil og dermed en reduksjon i usikkerheten ved bestemmelse av t kvarkens masse. Ved $t \to bl\nu, b \to c\mu\nu$, vil en få meget rene signaler og liten eksperimental feil ved beregning av m_t.

1.2.3 Supersymmetri (SUSY).

Teoretisk er det foreslått at det er en symmetri mellom fermioner og bosoner, som gjør at vi vil ha et sett av partikler i tillegg til de kjente idag, men der dagens spinn 1/2 partikler vil være spinn 0 spartikler og spinn 1 partikler vil være spinn 1/2 spartikler. En har enda ikke observert slike spartikler slik at en antar deres masse er større enn for partikler. Ved LEP har en funnet en nedre grense på $m(\tilde{e})$ til å være $> m_z/2$. Typiske signaturer vil være:

Spartikkel type	**Signatur**
gluino/skvark	$E_T^{miss} + jets$
	to leptoner med samme ladning
	multiple $Z + jets + E_T^{miss}$
sleptons	isolerte leptoner med høy p_T
chargino/neutralino	isolerte leptoner med høy p_T

1.2.4 Higgs-partikkelen.

En av hovedmotivasjonene for LHC er søk etter Higgs boson i masseområdet $80GeV$ til $1TeV$.

Henfallskanal	m_H (GeV)	Karakteristikk
SM Higgs		
$\gamma\gamma$	$80-150$	γ-identifikasjon mot π^0 og e masseoppløsning.
$ZZ^* \to llll$	$130-2m_z$	Masseoppløsning, leptonisolasjon ved høy $\mathcal{L}$
$ZZ \to ll\nu\bar{\nu}$	$400-800$	Full romoppløsning i kalorimeteret for identifikasjon av jet-strukturer i foroverretningen.
$WW \to l\nu jj$ og $ZZ \to lljj$	> 600	Høy$-p_t$ W, $Z \to jj$ jet-rekonstruksjon i foroverretningen.
SUSY Higgs		
$\to \gamma\gamma$		Høy $\mathcal{L}$
$A, H \to \tau\tau$		Lav $\mathcal{L}$
		tau identifikasjon
		b *tagging* til *veto* $t\bar{t}$*-backg.*
		kalorimeter *hermiticity*
$t \to bH^+$ med $H^+ \to \tau\nu$		Som ovenfor.
med $H^+ \to c\bar{s}$		*jet* masseoppløsning.

Tabell 1.2: *Typiske Higgs-henfall.*

- I minimal SM vil en ha en Higgs dublett og 4 felt, tre av dem gir masse til $W^{\pm}Z$ og en gir opphav til en ny partikkel H^0.
- I en minimal SUSY modell vil en ha to Higgs dubletter og 8 felt, 3 av dem gir masse til $W^{\pm}Z$ og de siste 5 gir opphav til 5 nye partikler, H^0,h^0, A^0 og $H^{\pm}$.

Ved $m_H < 200$ GeV vil bredden Γ_H være smal ($\Gamma_H^{tot} < 0.01$ GeV) og stille strenge krav til eksperimentell oppløsning. Ved $m_H > 400$ GeV ($2m_z$) vil bredden Γ_H ha økt betydelig ($\Gamma_H^{tot} > 100$ GeV). Dette gjør at en må stille store krav til eksperimentets evne til å kunne observere m_H over en stor bredde. Tabell 1.2 viser de forskjellige henfallskanalene for Higgs og hva som er karakteristisk for dem.

1.2.5 Kvark-sammensetninger.

Det er nærliggende å spekulere i om dagens elementærpartikler er sammensatt av et underliggende sett av mer fundamentale enheter enn kvarker og leptoner. Dette pga. at vi har tre liknende familier uten en skikkelig forklaring for dette. Ved høye impuls overføringer ved qq spredning vil en kunne ha muligheten for å se kvark substrukturer. Måling av dette kan bli gjort ved å måle tverrsnittet på jet-strukturen og vinkelfordeling. For en slik måling kreves gode kalorimetere [35].

1.2.6 LHC som en b-kvark fabrikk.

En kan tenke seg å produsere b kvarker på følgende måter:

- Ordinær p-p kollisjon, der b kvarkene er produsert i foroverretning.

- En kan ekstrahere protoner ut fra LHC primærstrålen og sende de inn mot et stasjonært mål.
- En kan blåse en stråle igjennom stråle røret for protonene. Denne strålen vil være i tilnærmet ro i forhold til proton strålen, en vil dermed få et stasjonært eksperiment.

Hovedmålet vil være å se på CP-brudd i b-henfall som før bare er observert i K henfall.

1.2.7 Lav luminositet fysikk.

Ved oppstart av LHC vil en ha en noe lavere luminositet en den maksimale ca. $10^{32} cm^{-2} s^{-1}$. En vil da kunne studere:

- Reaksjoner som krever god b-kvark gjenkjenning ved den reduserte bakgrunn. Dette vil gjelde ved f.eks beregning av m_t og søk etter sjeldne nye topp kvark henfall som $t \longrightarrow bH^+$.
- Identifikasjon av hadronisk τ henfall i søk etter ladet Higgs eller pseudoscalar Higgs henfall i SUSY.
- Rekonstruksjon av CP-brytende B-henfall.

1.3 Rater.

Hendelseraten og luminositeten for LHC er blitt justert ettersom nye aspekter har kommet frem. 1992/1993 ble det foreslått å redusere kollisjonsfrekvensen av protonbuntene fra 66 MHz til 40 MHz. En del beregninger i denne oppgaven er fortsatt med de eldre ratene, mens en del beregninger er gjort på nytt. En må være oppmerksom på at de tallene som er brukt som grunnlag for beregninger av rater ved LHC er foreløpige estimater og ikke endelige.

1.3.1 Tverrsnitt σ og log(s) fysikk.

For å kunne finne hvilken partikkel-tetthet en vil ha i kollisjonspunktene ved LHC må en først estimere det *non single* diffraktive tverrsnittet σ_{nsd} som kan uttrykkes ved:

$$\sigma_{nsd} = (1 - \frac{\sigma_{el}}{\sigma_{tot}})(1 - \frac{\sigma_{sd}}{\sigma_{tot}})\sigma_{tot} \tag{1.3}$$

hvor σ_{el} er det elastiske tverrsnittet, σ_{sd} er det *single* diffraktive tverrsnittet og σ_{tot} er det totale tverrsnittet, [47], [10],[2].

- **Totalt tverrsnitt.** Ved å ekstrapolere σ_{tot} som en kjenner fra andre p-p eksperimenter vil en kunne estimere den forventede σ_{tot} for den høyere energien til LHC. Om $\sigma_{tot} \propto \log^2 s$, vil en forvente $\sigma_{tot} \approx 130mb$ ved LHC, men om σ_{tot} går mot en konstant vil $\sigma_{tot} \approx 90mb$. Vi estimerer derfor at $\sigma_{tot} \approx 110 \pm 20mb$ som foreslått i [18].
- **Elastisk tverrsnitt.** Ved å ekstrapolere resultater fra tidligere eksperimenter vil en finne $\sigma_{el}/\sigma_{tot} \approx 0.26$ ved LHC, som medfører $\sigma_{el} \approx 30mb$.
- ***Single* diffraktivt tverrsnitt**. Om en regner forholdet σ_{sd}/σ_{tot} som uavhengig av energien kan en fra $\sqrt{s} = 1.8$ TeV finne $\sigma_{sd}/\sigma_{tot} = 0.129$ ved LHC, som gir $\sigma_{sd} \approx 14mb$.

En vil da fra formel 1.3 få $\sigma_{nsd} \approx 70mb$ som er brukt i det følgende.

1.3.2 Minimum bias-hendelser.

En *non-single* diffraktiv hendelse vil gi opphav til en minimum bias kollisjon. Det er viktig å kunne vite hvor mange minimum bias-hendelser en vil ha ved LHC. Dette for å kunne si noe om hvor mange partikler som kommer ut av kollisjonene og bidra til bakgrunnen. Egenskapene til minimum bias-hendelser er ikke mulig å regne ut i QCD, så en må bruke ekstrapoleringer fra SppS og Tevatronen ved FERMI-lab og Monte Carlo (MC) simuleringer som gir $dn^{\pm}/d\eta \approx 6$ ladete partikler/minimum bias-hendelse. Dette gir oss 36 ladete spor/minimum bias-hendelse for $\mid \eta \mid = 3$.

Fra formel 1.2 ($\mathcal{L} = 1.7 * 10^{34} cm^{-2} s^{-1}, \sigma_{nsd} = 70mb$ og $\Delta t = 25$ ns.) kan en finne at en vil ha ca. 30 minimum bias-hendelser. Dette vil føre til:

- ca. 1080 ladede spor i innerdetektoren, dominert av $\pi^+\pi^-$.
- ca. 2160 partikler i kalorimeterene, dominert av $\pi^0 \longrightarrow \gamma\gamma$.
- < 0.1 muon spor.

Et problem ved den høye luminositeten en har ved LHC vil være at en får overlapping, *pileup*. Dette vil være:

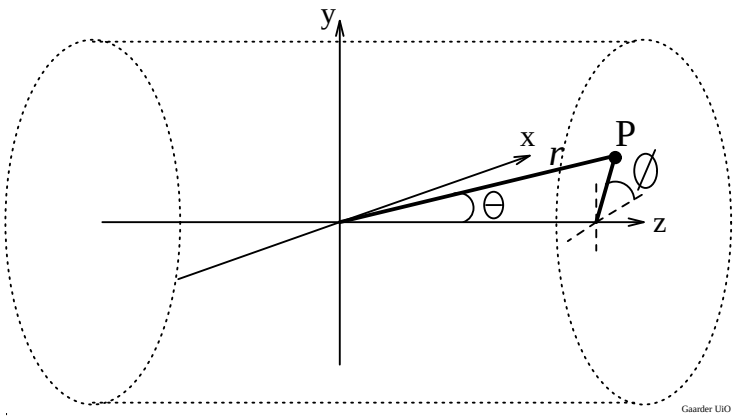
- Overlapping av den interessante hendelsen i rom pga. at en får flere hendelser samtidig (m.bias> 1).
- Overlapping av den interessante hendelse pga. at hendelser fra tidligere kollisjoner fortsatt vil være til stede i detektoren (*loopers* etc.).
- Overlapping i tid pga. detektor-responsen er lengre ennΔt.

En bør imidlertid prøve å redusere denne overlappingen, slik at usikkerheten i gjenkjennelsen av en hendelse blir minst mulig.

Kapittel 2

Et LHC-eksperiment.

Detektorer ved kollisjons-akseleratorer består av en tønnedel og en endcap-del. Den indre delen av detektoren vil rekonstruere spor, vekselvirkningspunkt-parametre og massefart, mens kalorimetrene sørger for energi-målinger av individuelle partikler. Utenfor det hele vil muon-kamre verifisere og måle avbøyninger (i.e. massefart) av muonet i et magnetfelt. Siden kollisjonene er symmetrisk fordelt rundt vekselvirkningspunktet (i ϕ) brukes ofte sfæriske koordinater som vist i figur 2.1, samtidig som en utformer eksperimentet som en tønne med tilnærmet 4π romvinkeldekning. I hadroniske kollisjoner er partikkel-tettheten i θ-retningen noenlunde flat i pseudora-



Figur 2.1: *De sfæriske koordinatene til et eksperiment.*

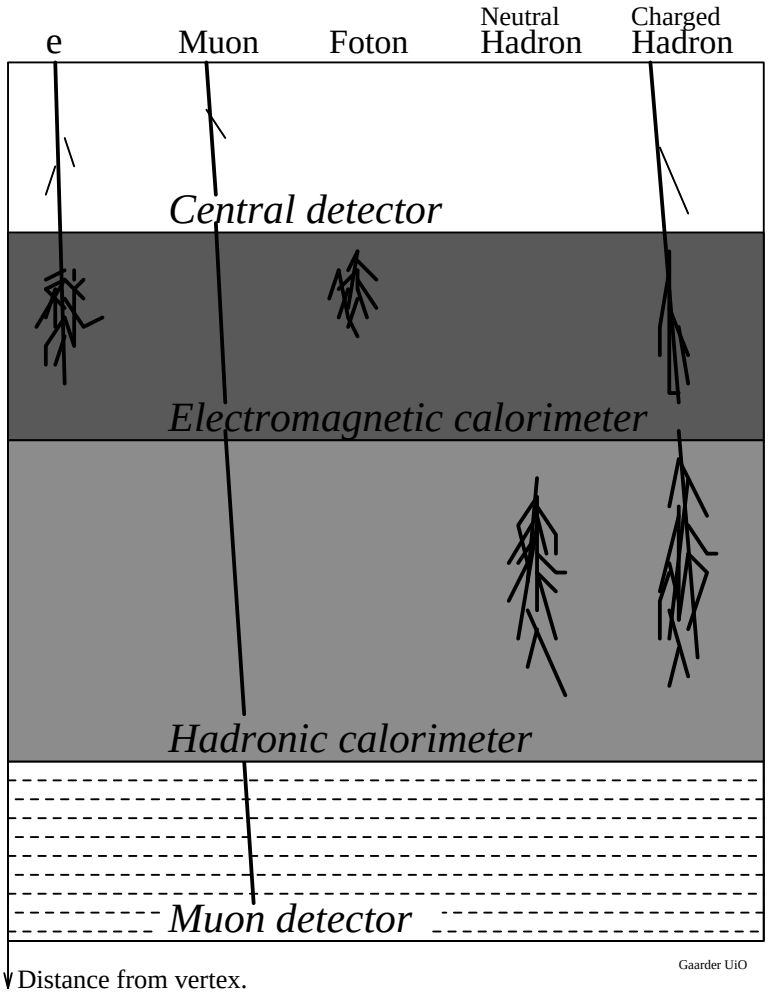
piditeten

$$\eta = -ln(tan(\frac{\theta}{2})) \tag{2.1}$$

og en bruker derfor ofte η til å beskrive dekning i θ-retning.

I tillegg til måling av retning, posisjon og energi av partiklene er identifikasjon et viktig punkt. Figur 2.2 gjengir hvordan en del viktige partikler vil avsette sine signaturer i spordetektorer, kalorimetere og muon-kammerene.

En vil i praksis ikke klare å dekke hele området (4π) rundt vekselvirkningspunktet i eksperimentet, som fører til at akseptansen , ε_a, blir mindre enn 1. Akseptansen for en spesifikk

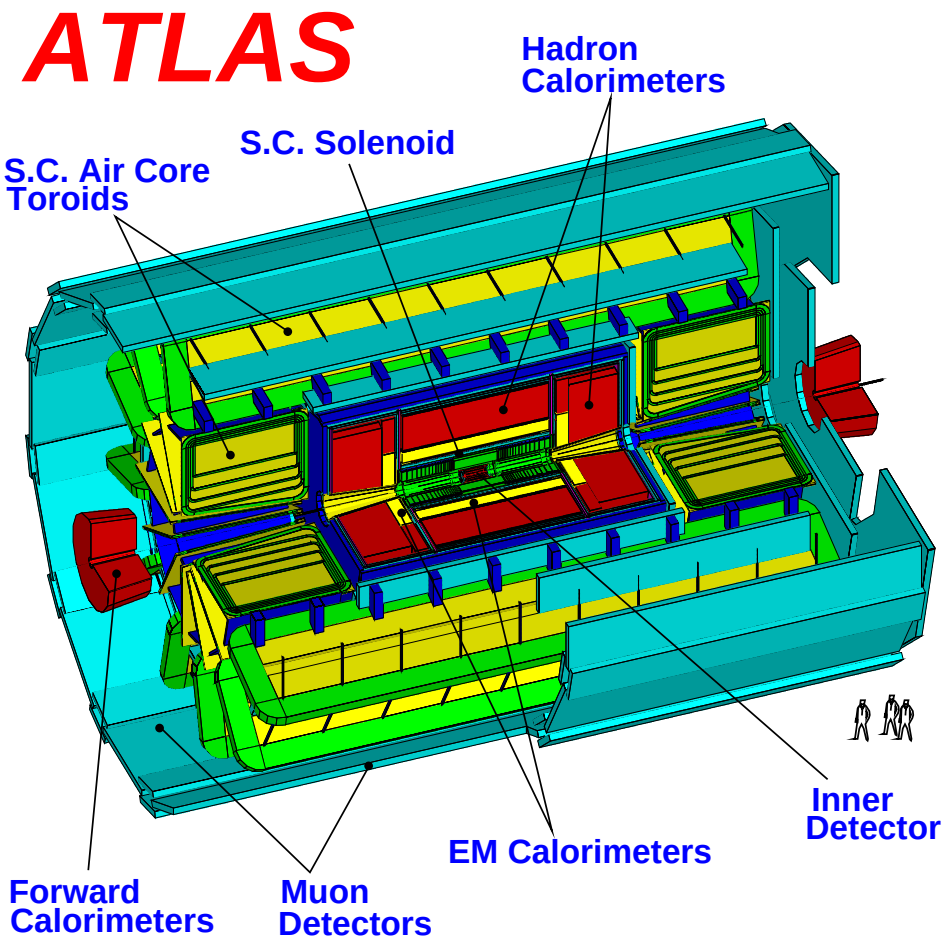


Figur 2.2: *Kriterier for partikkeldeteksjon.*

fysikk-kanal vil være en funksjon av hvor stort område eksperimentet geometrisk dekker, av effektiviteten til detektorene og av vinkelfordelingen til sluttproduktene i denne kanalen.

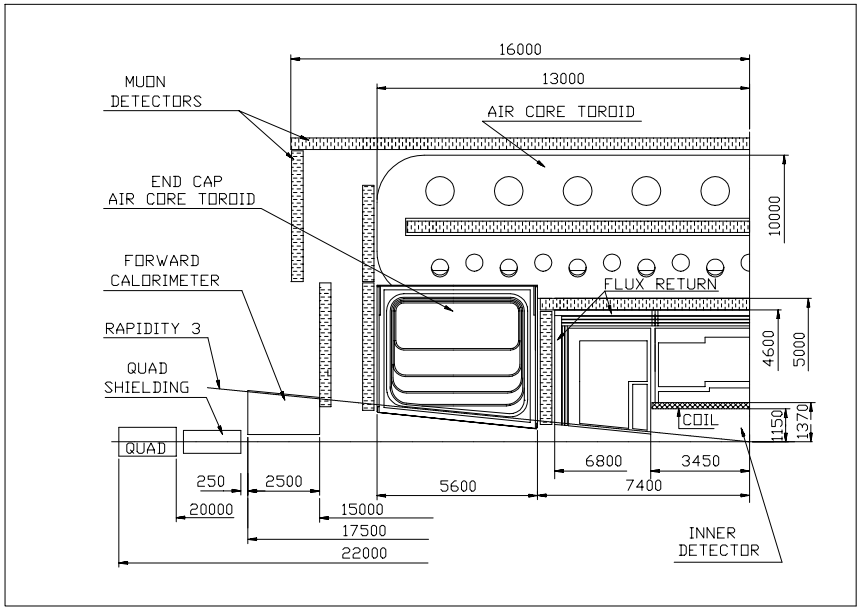
2.1 ATLAS eksperimentet

ATLAS [14] [16] vil være et *general-purpose* eksperiment for LHC. Hovedmålet for eksperimentet er å kunne gjøre observasjoner ved den høyeste luminositeten til LHC med en detektor som tar hånd om så mange signaturer som mulig, ved å bruke elektron, gamma, muon, jet-struktur og energi-balanse målinger (E_T). Det er også lagt vekt på å kunne foreta målinger ved lav luminositet. Figur 2.3, 2.4 og 2.5 viser utlegg for ATLAS detektoren.

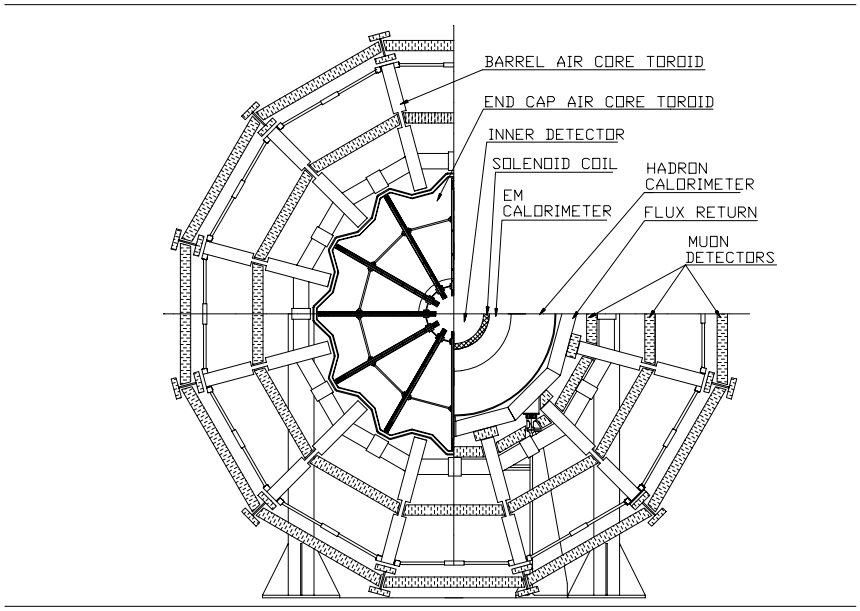


Figur 2.3: *ATLAS detektoren.*

Tabell 2.1 viser krav til detektoren, og tabell 2.2 viser oppsummert de forskjellige karakteristikker til subdetektorene.



Figur 2.4: *ATLAS detektorens dimensjoner.*



Figur 2.5: *ATLAS detektoren sett fra enden.*

Detector component	Resolution, characteristics	η coverage	
		Measurement	Trigger
EM calorimetry	$10\%/\sqrt{E} \oplus 1\%$	± 3	± 2.5
Preshower detection	Enhance $\gamma - \pi^0$ separation	± 2.5	
Jet and missing E_T Calorimetry			
- barrel and end-cap	$50\%/\sqrt{E} \oplus 3\%$	± 3	± 3
- forward	$100\%/\sqrt{E} \oplus 7\%$	$3 < \lvert\eta\rvert < 5$	$3 < \lvert\eta\rvert < 5$
Inner detector	$5 \cdot 10^{-4} p_T \oplus 1\%$	± 2.5	
	Enhance electron identification	± 2.5	
	τ and b tagging	± 1.5	
Muon detection	20% or better at $p_T = 1$ TeV	± 3	± 2.5
	Stand-alone capability at highest luminosity		

Tabell 2.1: *Detektor spesifikasjoner.*

2.2 Trigger.

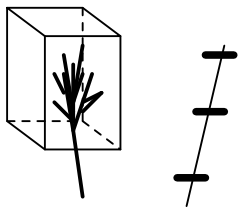
For å kunne sile ut de hendelsene som er interessante må en se etter det som er de karakteristiske signaturer for dem. En vil da *trigge* når disse oppstår. Dette gjør at de store hendelseratene som er beskrevet ovenfor blir betraktelig redusert og lettere å håndtere ved utlesning. En forutsetter at triggerene kommer ukorrelert i tid. En deler opp triggingen i flere nivåer, der en legger til flere og flere kriterier ettersom en går lenger ned i nivåene, dette er nærmere beskrevet i kapittel 2.4.

Subdetector	η coverage	Baseline design	Alternatives	Comments
Inner detector	± 2.5			
- vertexing	± 1.5	Si pixels	Si micro-strips	Removable at high $\mathcal{L}$
- innermost tracking	± 1.5	Si micro-strips		if necessary
	> 1.5	GaAs micro-strips		
- outer forward tracking	> 1	MSGC and TRD straws		
- outer central tracking	± 1	Si strips and pads	MSGC or	Further studies needed
		TRD straws	scint. fibres	to optimize layout
Superconducting solenoid		Integrated in LAr cryostat	Separate cryostat	2 T
Calorimetry	± 5			
- em with preshower				
- barrel	± 1.5	LAr Accordion	LAr TGT	
- end-caps	1.5–3	LAr Accordion or TGT	or scint. fibres	
- hadronic				Possibly a tail
- barrel and end-caps	± 3	LAr, or scintillating fibres		catcher calorimeter
		or scintillator tiles		in case of LAr
- forward	3–5	Liquid scintillator		
		or high pressure gas		
Muon system	± 3			
- magnet	± 3	Superconducting air-core		
		or warm iron-core toroids		
- tracking detectors	± 3	High pressure drift tubes		
		or honeycomb strip chamb.		
		or jet cell drift chambers		
- trigger	± 2.5	Resistive plate chambers	Combined with	
			tracking detector	

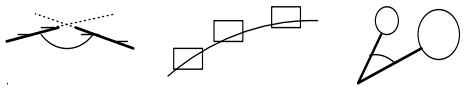
Tabell 2.2: *Egenskapene til subdetektorene.*

Typiske triggerrater vil være:

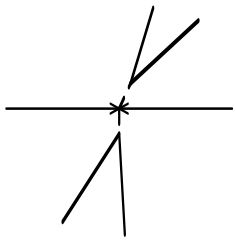
- **LVL1** En vil ut fra LVL1 ha gjort en reduksjon ned til ca. 10^5 Hz. Dette vil bli gjort vha. en trigger, T1, som i snitt vil bli gitt 1 av 400 ganger og vi vil ha en reduksjonsfaktor $a_{LVL1} = 400$. T1 avgjørelsen vil bli avgjort vha. partikkelidentifikasjoner, totale energi kutt, høye p_T elektroner, muoner, jet-struktur identifikasjon, manglende E_T, lokale spor rekonstruksjoner og energi rekonstruksjon av hurtig tilgjengelig makro-granular informasjon.



- **LVL2** En vil ut fra LVL2 ha gjort en reduksjon ned til ca.10^3 Hz. Dette vil bli gjort vha. en trigger, T2, som i snitt vil bli gitt 1 av 100 ganger og vi vil ha en reduksjonsfaktor $a_{LVL2} = 100$. T2 avgjørelsen vil bli avgjort vha. rene partikkel og kinematiske signaturer, finere granulare transvers og langsgående profiler, kinematiske kutt, spor rekonstruksjon og topologi.



- **LVL3** Reduksjon ned til ca. 100 Hz. Denne reduksjonsfaktoren $a_{LVL3} = 10$ vil komme fra mere kompliserte og tidkrevende fullstendige prosesser som rekonstruksjon, analyse og identifikasjon av fysisk hendelse.

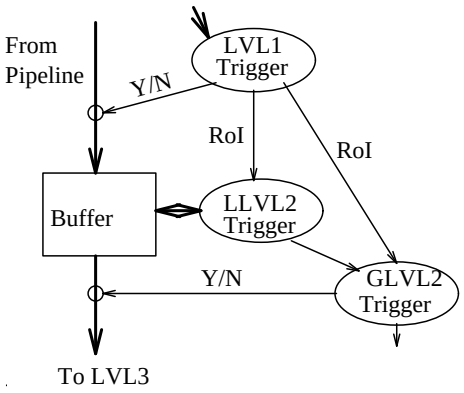


2.3 Region of Interest.

Ved første nivå vil en finne og merke *Region of Interest* (RoI). RoI definerer området til interessante signaturer i detektoren, f.eks. elektroner i kalorimetrene, muon i muon-kammere etc. De høyere nivåene bruker RoI informasjonen for å avgjøre om det skal gis en positiv trigger.

Figur 2.6 viser generelt konseptet for hvordan RoI blir brukt. LVL1 triggeren identifiserer RoI i $\eta - \phi$ rommet for den lokale og globale LVL2 triggeren, typisk $\Delta\eta \cdot \Delta\phi = 0.2 \cdot 0.2$ i kalorimeteret. Usikkerheten i vekselvirkningspunktet og avbøyningen av ladede partikler gjør at en inn mot sentrum må merke av en kjegle som er videre enn klusteret i kalorimeteret [40].

LVL2 triggeren vil jobbe mot de identifiserte dataene i LVL2 bufferet som tilhører RoI. Det er derfor viktig at RoI dataene er raskt og enkelt tilgjengelige for LVL2 triggeren. Det bør ikke være nødvendig å lese ut mere data en RoI til LVL2 før en trigger 2 gis.



Figur 2.6: *Prinsippet for RoI.*

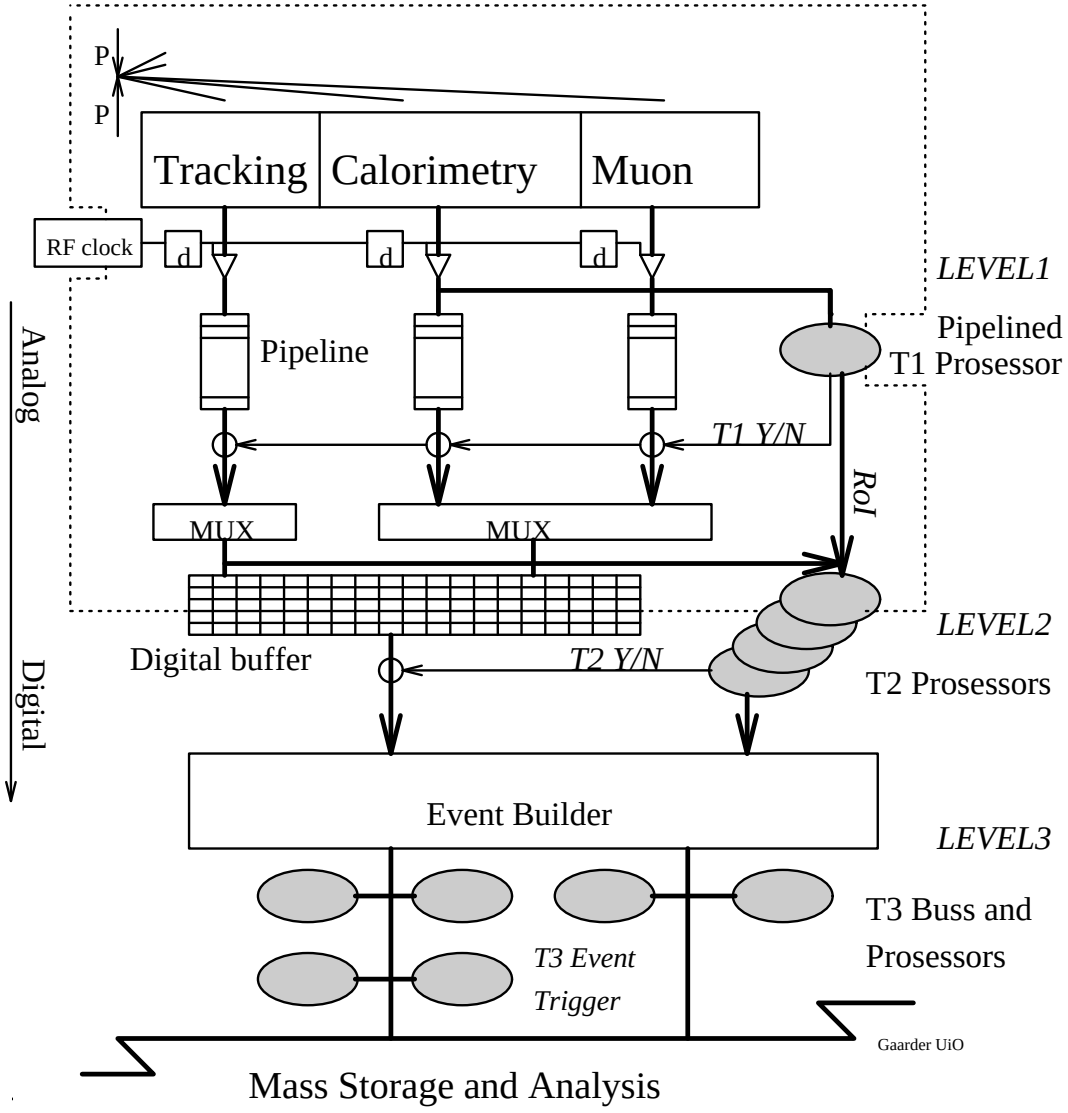
2.4 Et data-innsamlings-system (DAQ) for et eksperiment.

For å kunne samle inn og strukturere det som måles i et eksperiment må en ha et data-innsamlings-system (DAQ). Figur 2.7 viser en skisse av et datainnsamlingssystem for LHC.

DAQ systemet vil typisk bestå av følgende deler. Først Frontend Elektronikk (FE) som vil tilhøre nivå 1 og være montert ved selve detektoren. Dette kan være forforsterkere og signal-*shapers*, enheter for lagring som pipeline og buffere, filterprosessorer, diskriminatorer, multipleksere og analog til digital konvertere (ADC). I tillegg vil en ha en kontroll og feilbehandlingslogikk som styrer utlesningen av nivå 1, samt en T1 prosessor som vil ha sin logikk både inne i selve detektoren og utenfor. En vil ha en flytende overgang mellom FE og nivå 2 som vil bestå av multipleksere, trigger prosessorer, digitale og evt. analoge buffere, data busser og signal kabler. Nivå 2 vil være sterkt knyttet til nivå 3 som vil bestå av et fullstendig datahåndteringssystem med konvensjonelle databusser og prosessorer. Her vil en fullstendig hendelsesoppbygning foregå og en utvelgelse (T3) av interessante hendelser, som igjen vil bli lagt inn på konvensjonelle lagringsenheter. Overordnet hele DAQ systemet vil det være et kontrollsystem som styrer og monitorerer foreløpet av hele eksperimentet.

For LHC må en designe et DAQ system med en ekstrem yteevne etter følgende hoved-prinsipper:

- Hierarkisk data innsamling. Dette vil si at en gjør en utsiling av hendelser ut fra strengere og strengere kriterier ettersom dataene beveger seg igjennom DAQ systemet.
- Data-buffere i FE, *pipeline*. Dette for å allerede i fronten av DAQ systemet kunne gjøre



Figur 2.7: *Et generelt DAQ system for et LHC eksperiment.*

en utsiling av hendelser. For å kunne gjøre dette må en lagre dataene inntil hendelsen er vurdert.

- Parallell datainnsamling og prosessering.

Et utlegg for ATLAS eksperimentets datainnsamlingssystem er illustrert i figur 2.8. Mange grunnleggende antagelser er tatt med i denne modellen. Følgende forkortelser er brukt i figur 2.8:

- LVL1 (LVL2) ... nivå 1 (2).
- LLVL1 (LLVL2) ... lokal trigger prosessor på nivå 1 (2).
- LROC1 (LROC2) ... lokal *readout* kontroller på nivå 1 (2).
- GLVL1 (GLVL2) ... global trigger prosessor på nivå 1 (2).

Det vil generelt være en fordel å lagre så lite som mulig data inne i detektorene, pga. plassmangel, varmeutvikling osv. Dette må da vurderes mot at det er en klar begrensning på overføringshastigheten for data ut av detektorene.

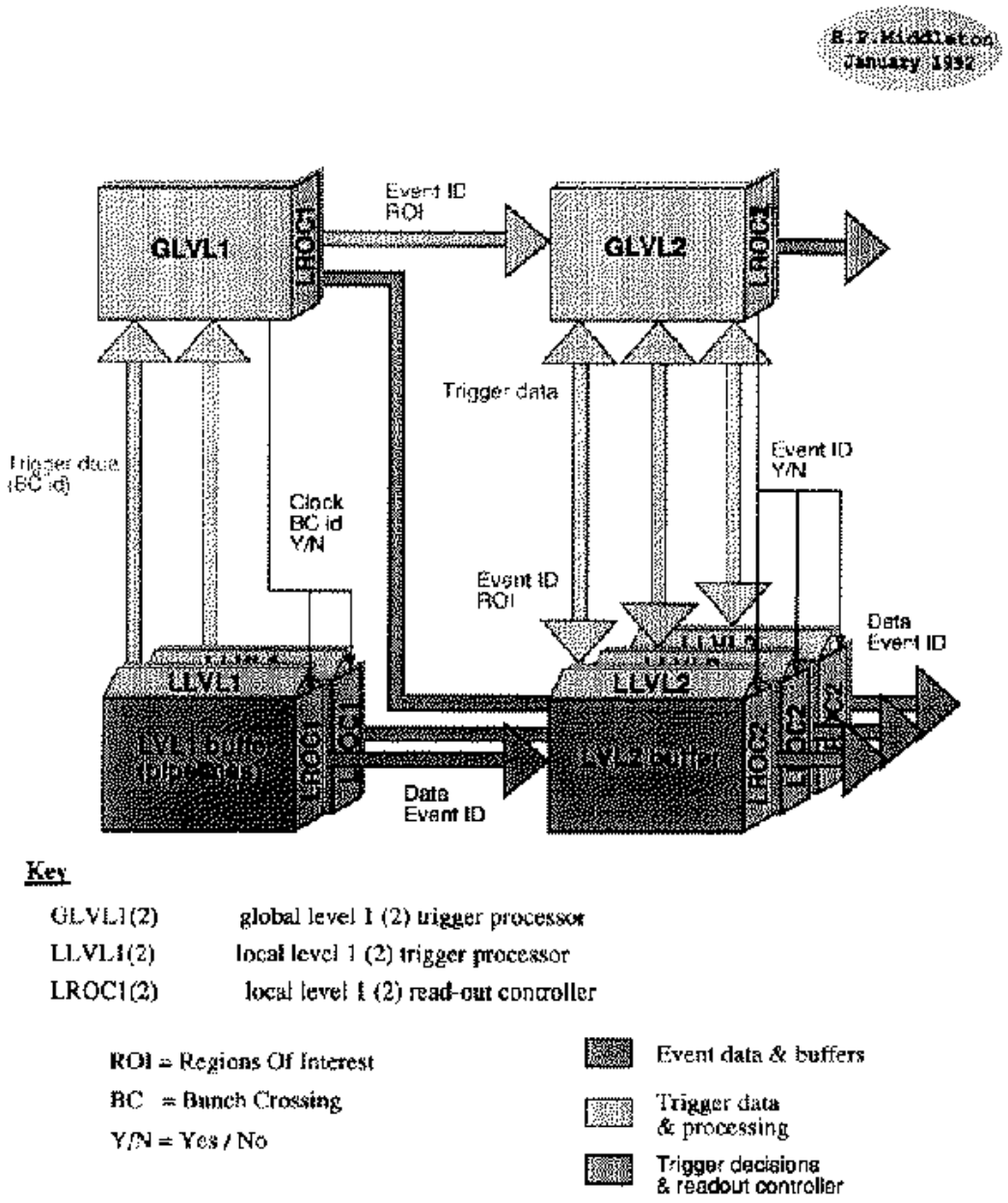
Et eksperiment er delt opp i flere deldetektorer[1] som hver for seg har forskjellige oppgaver. Hver deldetektor må ha deres egen tilkobling til resten av DAQ systemet, og må kunne selvstendig kjøre tester og kalibrering. En deldetektor er igjen delt opp i funksjonelt like subdeler. En subdel har hver for seg sin egen LROC1 og mottar en klokke-linje. Subdelene må selv kunne håndtere sin interne tid. En subdel vil typisk bestå av et antall front-end sensorer, en eller flere LLVL1 trigger prosessorer som behandler og lagrer informasjon til det evt. blir gitt en global trigger (T1). En del retningslinjer kan bli gitt:

- Den interne synkronisering av pipelinene gjøres innen subdelen.
- Deldetektoren må selv håndtere data overføringen til LVL2 bufferne.
- Det blir gjort feil-sjekking inne i hver subdel.
- Subdelene må være koblet til en databuss/linje for kontroll, debugging, kalibrering osv.
- Enhver detektor skal kunne bli kjørt alene eller bli koblet ut.
- Hele eksperimentet, alle detektorer og submoduler har den samme pipeline tidsforsinkelse ($t_{T1delay}$).

Nivå 2 bufferet, LROC2 og LLVL2 trigger prosessoren vil enten være montert på subdelen eller utenfor detektoren. LROC2 og LLVL2 vil ofte være bygd sammen til en prosessor. Generelt er det viktig at LROC2 og LLVL2 begge kan kommunisere med GLVL2 prosessoren. Dette for at en skal kunne lese ut og behandle RoI på en enkel og rask måte uten at all data fra hele eksperimentet må leses ut. Data som ikke tilhører RoI vil da ligge i nivå 1 eller 2 buffere til det gis signal om at disse enten skal forkastes eller leses ut.

Hver subdel vil få et klokkesignal, en trigger-bestemmelse og muligens en BC identifikasjon fra GLVL1. Subdelene må kunne oppdage og rapportere feil som fulle buffere, asynkronasjon osv. GLVL2 (GLVL3) vil være koblet til en generell database for eksperimentet. Dataene ut av subdelen vil typisk være adresser (koordinater), verdier og et hendelsesnummer som tilhører

[1] F.eks barrel-EM, silisium-tracker, barrel-muon, etc.



Figur 2.8: *Konseptuelt skjema.*

verdiene. Om en verdi går tapt, eller at det har oppstått en feil i et område, må dette registreres i databasen. Hendelses-nummeret vil normalt bli generert lokalt inne i LROC, evt. globalt og sendt til subdelen.

Kravene og spesifikasjonene til LHC gjør at en får mange viktige og ofte motstridende krav til detektorene. En konsekvens av dette blir at mange parametere må optimaliseres sett fra et globalt synspunkt og prioriteres etter hvor viktige de er. Noen av disse er summert nedenfor:

1. Lavt effektforbruk fører til enklere kjøling, mindre geometrisk unøyaktighet pga. termiske fluktuasjoner, mindre kabling, enklere bærestruktur, mindre materiale i detektoren osv.

2. Peaking time amp. går som $\propto \frac{1}{effekt}$. Dette fører til at om effekten reduseres må en filtrere/prosessere et diskret signal, tilpasset rett BC, ut fra forsterkerens signal som vil gå over flere BC. En vil også få større problemer med overlapping av signaler fra forskjellige hendelser.

3. Signal/støy går som $\propto \frac{1}{effekt}$, som fører til at en får problemer med mye støy om effekten reduseres. Ved en gitt luminositet vil det være klar grense for hvor mye støy en kan tillate før det vil være umulig å rekonstruere en hendelse.

   Mer direkte vil det også være en klar grense for hvor mye støy en tolererer i FE elektronikken før signalet tapes.

4. En må ha elektronikk som er strålings-hard.

5. En må minimalisere effekten av fluktuasjoner mellom subdetektorer. Dette kan være fluktuasjoner pga. produksjon, håndtering, geometrisk-posisjon (opp/ned, B-felt), termiske svingninger i elektronikk, strålingsrate, strålingskader osv. Mange av disse fluktuasjonene vil en fjerne ved deldetektorens utforming iforhold til hvor den er plassert i eksperimentet, kalibrering osv.

6. En må ha rask elektronikk med presis tidsoppløsning.

2.4.1 Effektiviteten til DAQ systemet.

En vil modellere DAQ systemet etter triggerstrategien og de data-rater en trenger å lese ut, som igjen er gitt av de fysiske hendelser en leter etter. En viktig parameter for DAQ systemet vil være å få effektiviteten, ε_{DAQ} så høy som mulig. Dessuten er det viktig at eventuelle tap av informasjon er tilfeldig og **ikke** er korrelert med spesifikke fysikk-kanaler.
Antallet hendelser ut av DAQ systemet og inn på *tape* er[2]:

$$N_{tape} = N_{fys} * \varepsilon_{tot} = N_{fys} * \varepsilon_a * \varepsilon_{trigger} * \varepsilon_{DAQ}, \tag{2.2}$$

hvor N_{fys} vil være ved et ideelt eksperiment, ε_a er akseptansen, $\varepsilon_{trigger}$ er trigger effektiviteten og ε_{DAQ} er effektiviteten for DAQ systemet. $1-\varepsilon_{DAQ}$ vil da være andelen av tap pga. bufferoverflyt, overføringsfeil, *time-out* osv. En ønsker selvfølgelig å lage eksperimentet slik at ε_{tot} nærmer seg 1, samtidig som en vurderer dette mot påliteligheten for eksperimentet, levetid på eksperimentet, kostnader osv. En må vurdere $\varepsilon_{trigger}$ mot ε_{DAQ}. En økning av $\varepsilon_{trigger}$ vil føre til en økning av bakgrunn og dermed en kraftig økning av datamengden som DAQ systemet skal håndtere.
En kan skille mellom to typer ineffektivitet i DAQ systemet:

[2]Med hendelser menes her de hendelser vi ser etter.

- Ineffektivitet ved full korrelasjon, $1-\varepsilon_{sync}$. Dette vil være tap av en hel hendelse og tilsvare en reduksjon i den effektive tiden eksperimentet er i drift. En kan akseptere $1-2\%$ av denne typen ineffektivitet.

- Lokale ukorrelerte ineffektiviteter, $1-\varepsilon_{async}$. Dette vil da være hvor **stor del** av innerdetektoren DAQ systemet ikke klarer å dekke. Normalt vil dette si hvor stor sannsynlighet en har for å miste alle data tilhørende en hendelse i en submodul . Dette vil redusere presisjonen til eksperimentet og muligheten for å rekonstruere hendelser. Dette vil være en mere kritisk ineffektivitet og må vurderes ved fullstendige simuleringer av DAQ systemet basert på MC fysikk-simuleringer.

En vil ha:

$$\varepsilon_{DAQ} = \varepsilon_{sync} \cdot \varepsilon_{async} \tag{2.3}$$

der $\varepsilon_{sync} = 1 - P_{dead}(T1)$ vil være effektiviteten gitt av tapet av hendelser i den synkrone delen av DAQ systemet og $P_{dead}(T1) = \frac{T1_{dead}}{T1}$ er dødtiden for T1. ε_{async} vil være effektivitet funnet fra asynkrone tap av deler av en hendelse i DAQ systemet. Det kan være et definisjons-spørsmål om hva som er ε_{sync} eller ε_{async}. Dette pga. at en detektor internt vil være synkron selv om den globalt mot andre detektorer er asynkron.

En bør forsøke å lage eksperimentet slik at tap av data er mest mulig synkront. I tillegg til en økt evne til å lese ut hele hendelser komplett som nevnt ovenfor, vil en ved at tap av data er mest mulig korrelert ofte få en bedre effektivitet , ε_{DAQ}, og en bedre oversikt over tilstanden inne DAQ systemet. Forbedringen av ε_{DAQ} kan forklares på følgende måte: Anta 2 deldetektorer A og B, med en effektivitet på ε_A og ε_B[3].

- Når dødtiden for detektorene er asynkrone, dvs. at det ikke er noen korrelasjon mellom når detektor A og detektor B mister data, vil en ha en total effektivitet:

$$\varepsilon_{async} = 1 - P(A_{dead}) - P(B_{dead}) + P(A_{dead}) * P(B_{dead}) = \varepsilon_A * \varepsilon_B \tag{2.4}$$

- Når detektorene er synkrone vil en ha en total effektivitet:

$$\varepsilon_{sync} = 1 - P(A_{dead}) = 1 - P(B_{dead}) = \varepsilon_A = \varepsilon_B \tag{2.5}$$

En kan se at, når sannsynligheten for tap i hver enkelt detektor holdes konstant, vil ε øke når detektorene gjøres mere korrelerte.

En vil ha at sannsynligheten for å ikke miste noen submoduler i en detektor vil være:

$$P = (\varepsilon_{async})^n \tag{2.6}$$

hvor n er antallet submoduler. En kan se at ved stor n (typisk $n \approx$1000-3000), må $\varepsilon_{async} \approx 1$ for at en ikke skal miste noe av informasjonen tilhørende en hendelse.

En kan i de fleste tilfeller anta at ε_{async} vil være konstant om prosessene i submodulen varieres proporsjonalt med detektorarealet submodulen dekker. Dette diskuteres nærmere i kapittel 5. Dette vil si at om raten av signaler inn til en submodul økes[4] med en faktor k må en redusere

[3]Her forutsettes det at ved ineffektivitet i en av detektorene forkastes hele eksperimentets hendelse.
[4]Inkl. støy etc.

tiden de prosesser[5] som er knyttet til den asynkrone delen bruker med en faktor k, om ε_{async} skal forbli uforandret.

Vi vil få trigger-dødtid pga. at vi har begrensninger i den synkrone delen av DAQ systemet. Denne dødtiden vil være en funksjon av den gjennomsnittlige trigger hastigheten, hvor tett etter hverandre triggerne kommer, bufferstørrelser og konstruksjonen av elektronikken i den synkrone delen av DAQ systemet.

Når DAQ systemet ikke er synkront lenger vil en kunne miste deler av hendelsen, uten at dette vil gi noe direkte bidrag til økt trigger dødtid, men en redusert mulighet til å rekonstruere hendelsen.

For å forebygge dødtid og tap av deler av hendelser må flere punkter vurderes:

- De globale nivåene bør i størst mulig grad holde rede på tilstanden i DAQ systemet. Ved at en på globalt nivå kjenner tilstanden til de lokale nivåene vil en kunne forhindre at det blir sendt triggere ol. som en allikevel ikke kan håndtere. En kan tenke seg at det ved stor belastning på et høyere nivå (nivå 2-3) gis negativ tilbakemelding til nivåene foran, for å redusere datamengden i DAQ systemet.

- En regulering av datastrømmen kan f.eks. bestå av at GLVL1 utelater enkelte triggere for å avlaste bufferne i fronten av submodulene. Eventuelt at GLVL1 blir kontrolert av en statistikk prosess som undertrykker triggere om de i perioder kommer med en for intens rate.

- En kan også tenke seg at en har forskjellig prioritet ved trigging på forskjellige fysikk-kanaler. En kan f.eks gi signaturen til en Higgs partikkel maksimal prioritet, og heller miste en kandidat til topp kvark henfall om DAQ systemet er hardt belastet.

- LVL1 kan gi signal tilbake til GLVL1 rett før det f.eks. er bufferoverflyt slik at GLVL1 kan holde igjen triggere til LVL1 igjen gir beskjed om at den har fri kapasitet. En vil da ha $\varepsilon_{async} = 1$ og et 100% korrelert DAQ system mhp. tap av data, slik at de hendelsene som leses ut vil være fullstendige og en vil heller ikke belaste DAQ system med ufullstendige hendelser. Det vil imidlertid trolig være vanskelig å gjøre dette. Spesielt pga. at mange subdeler ikke har en kobling direkte til GLVL1 pga. at de ikke deltar i avgjørelsen av en trigger.

- DAQ systemet kan ha forskjellige grader av hvor presist det leser ut og behandler data. Ved stor belastning av enkelte subdeler kan en da tenke seg at en istedenfor å miste data fullstendig, heller reduserer oppløsningen i disse subdelene. Prosessene i subdelene må da være slik at de i tide kan omstille seg før tap av data oppstår.

- En kan tenke seg å dimensjonere DAQ systemet slik at det har kapasitet til å håndtere den største dataraten som det er mulig kan oppstå.

Bruk av noen av disse metodene vil, i sin ytterste grad, føre til fullstendige korrelerte av tap i de forskjellige detektorer[6].

[5] Med prosesser mener vi her prosesser knyttet til tid som f.eks ADC, pakking ol. og ikke buffere ol.

[6] Eksperimentet er ikke lenger datadrevet og vil derfor ha en karakter mere lik de eksperimenter en har tidligere erfaring fra.

2.4.2 Den synkrone delen.

Den synkrone delen av systemet er generelt sekvensen fra hendelses-tidspunktet til det blir mottatt en første nivå trigger bestemmelse. Sekvensen vil ha for hver avgjørelse likt antall klokkepulser. Innen deler av DAQ systemet, f.eks. innen en deldetektor, vil en også ofte ha synkronitet etter at det er gitt en første nivå avgjørelse. De delene som er synkrone vil ha en 100 % korrelasjon når det gjelder tap av data.

De internt synkrone individuelle subdelene vil gi informasjon til sin lokale trigger prosessor, LLVL1. Alle LLVL1 vil så synkront oversende informasjon til den globale trigger prosessoren, GLVL1. Siden subenhetene opererer ved forskjellig lokal tid må det være forsinkelsesenheter eller annen synkronisasjonslogikk som sørger for at GLVL1 mottar dataene riktig. Etter et fast antall klokke pulser vil GLVL1 sende en trigger avgjørelse, muligens inkludert en BC identifikasjon, tilbake til subdelene.

Subdelene vil i mellomtiden ha lagret dataene i buffere, og vil ved en positiv trigger initialisere en overføring av data til LVL2 bufferne. På dette nivået vil DAQ systemet generelt (globalt) ikke være synkront lenger.

En del viktige parametere som avgjør konstruksjonen av den synkrone delen av DAQ systemet vil være:

- Den gjennomsnittlige raten av positive trigger 1 avgjørelser.
- Tiden fra detektorene får data inn i FIFO'ene[7] til det blir gitt en trigger 1 avgjørelse.
- Typen elektronikk i fronten og deres mulighet til å kunne håndtere dataene. F.eks. hvor lite intervallet mellom to triggere kan være.
- Hastigheten på overføringen av data ut av LVL1 til LVL2 etter at en positiv trigger 1 avgjørelse er tatt.

2.4.3 Den asynkrone delen.

Den asynkrone delen av DAQ systemet begynner generelt når LROC1 til subdelene har mottatt en trigger 1 avgjørelse, og går helt til prosedyren for hendelsesoppbygningen er ferdig. En kan allikevel dele opp den asynkrone delen i mindre deler og finne at disse mindre delene ofte har interne synkrone deler som globalt er asynkrone. Etter en positiv trigger 1 avgjørelse vil subdelene begynne å håndtere og forberede data for oversendelse til LVL2 bufferne. GLVL2 prosessoren vil bli gjort oppmerksom på en ny hendelse fra GLVL1, og vil i samarbeid med LLVL2 prosessorene begynne å jobbe med dataene ettersom de asynkront mottar dem.

En må vurdere flere forskjellige arkitekturer for kommunikasjonen mellom GLVL2 og LLVL2 prosessorene. Først og fremst må en ha en slik kommunikasjon som utnytter RoI mekanismen, slik at LLVL2 prosessorene fritt kan ha tilgang til dataene i LVL2 bufferne som er knyttet til RoI.

En kan tenke seg flere forskjellige arrangementer av de lokale og globale LVL2 prosessoringsdelene. Med en RoI definisjon kan en tenke seg å overføre all data fra en hendelse og en detektor til en GLVL2 prosessor. Dette fører til at en vil trenge en farm av GLVL2 prosessorer, men

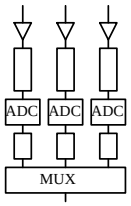
[7] FIFO, først inn først ut.

redusere behovet for lokal prosessering gjort av LLVL2 prosessorer. Dette vil føre til at overlappingsproblemet i kalorimetrene forsvinner[8], men beregningen av nivå 2 triggeren vil trolig ta lengre tid.

En annen løsning vil være å foreta så mye som mulig av nivå 2 trigger beregningene lokalt, og prøve å behandle problemet med overlapping lokalt. Kun resultatene av den lokale prosesseringen vil da bli oversendt til noen få GLVL2 prosessorer.

2.4.4 Analog kontra digital datahåndtering.

Signalet fra detektorene vil alltid være analoge. Et sted i DAQ systemet må de analoge signalene transformeres over til digitale (evt. forkastes). Denne transformeringen skjer ved bruk av en enkel diskriminasjon eller ved en ADC. Diskriminatoren vil vha. et grensenivå avgjøre om det er et signal tilstede og evt. sette en binær bit som ja eller nei. Denne binære verdien kan da behandles som en fullverdig informasjon av DAQ systemet eller en kan i tillegg utføre en analog til digital transformasjon, som da vil gi oss en mere fullstendig informasjon om signalet. En må vurdere for hver enkelt detektor hvor en bør gjøre denne transformasjonen fra analog til digital. En transformasjon tidlig i DAQ systemet vil generelt gi fordeler og ulemper som:



+ En kan tidlig gjøre digitale signalberegninger og analyser.

+ En får en enklere timing av signalene.

+ Støy og strålingssterk dataoverføring.

- Større datamengde å transportere.

- Data fra en hendelse vil ofte bli tidligere asynkrone i DAQ systemet.

- Høyere effekt forbruk, pga. flere ADC'er og evt. pga en større datamengde som må transformeres.

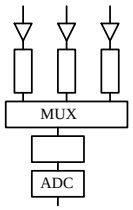
- En trenger flere elektriske forbindelser, dette om signalet blir transformert til en digital verdi med flere bit. En kan evt. bruke en seriell overføring ved høyere hastighet.

En transformasjon sent i DAQ systemet vil gi fordeler og ulemper som:

+ Lavt effekt forbruk.

+ Plassbesparelse, færre integrerte kretser i FE.

[8] Dette vil f.eks være energiavsetning i et kalorimeter mellom to subdeler.



Trigger	Rate
≥ 1 isolated em cluster with $P_T > 40$ GeV (isolation not required for clusters with $P_T > 65$ GeV).	31 kHz
≥ 2 isolated em clusters, each with $P_T > 20$ GeV (loose isolation cut)	16 kHz
$\geq 1\mu$ with $P_T > 20$ GeV	8 kHz
$\geq 2\mu's$, each with $P_T > 20$ GeV	67 Hz
≥ 2 jets, each with $P_T > 200$ GeV	5 kHz

Tabell 2.3: *Nivå-1 trigger frekvens ved* $\mathcal{L} = 1.7 \cdot 10^{34} cm^{-2} s^{-1}$.

\+ Mindre kompleks og mere pålitelig FE.

\- Dynamisk område kan bli mindre.

\- Vanskeligere kalibrering, men evt. mindre kalibrering pga. færre ADC.

\- Følsomt for pickup og støy.

\- Analogt signal krever større krav til timing enn en digital puls ved samme frekvens. Dette pga. at de digitale signalene kun er avhengig av en grenseverdi, mens en må lese av den presise verdien for et analogt signal.

ADC'ene vil ofte ikke være lineære, men ha en logaritmisk respons, som vil øke den dynamiske bredden. Typisk vil en ha en dynamisk bredde på 15-16 bit og oppløsning på 9-10 bit i kalorimetrene. Innerdetektoren vil ha en noe lavere dynamisk bredde og oppløsning (typisk 7 bit).

2.4.5 Global nivå 1 trigger.

Det meste av informasjonen for nivå 1 triggeren vil komme fra kalorimetrene (*em* og hadron) og muon detektoren. Estimater viser at overføring av informasjon til triggerprosessoren, beregningene og overføring av trigger avgjørelsen frem til pipelinene vil kunne bli gjort på under 2μs. Tabell 2.3 viser trigger 1 ratene for de forskjellige signaturer. En kan pessimistisk totalt regne en trigger 1 rate på $\approx$ 100 KHz. Noe som tilsvarer at det i snitt gis en trigger 1 for hver 400 BC eller $< t_{T1delay} >= 10\mu s$.

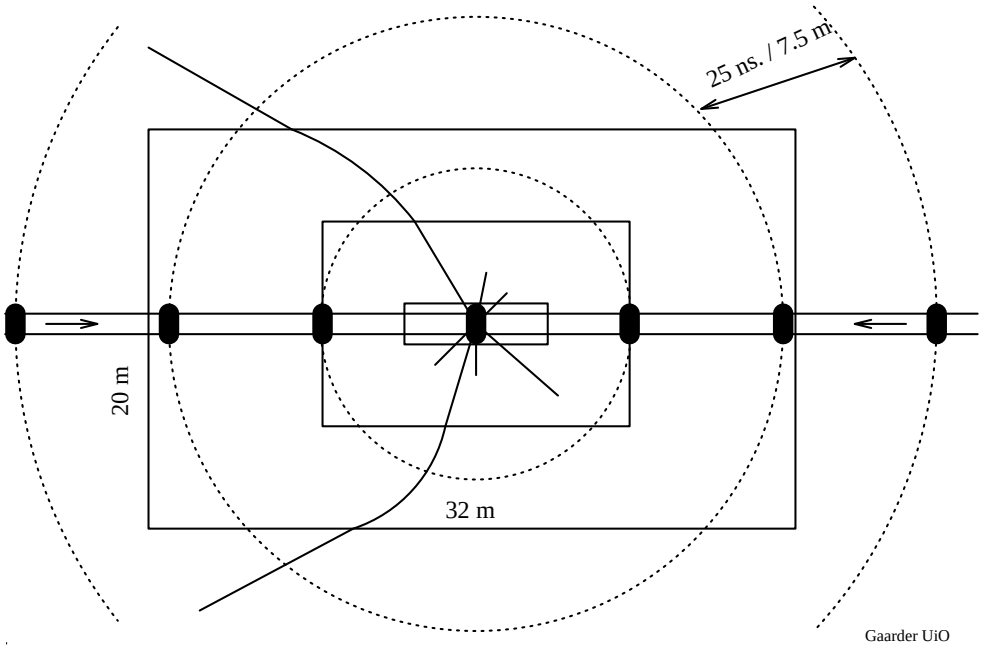
Triggeravgjørelsen vil bli sendt til alle subdeler i eksperimentet vha. individuelle forsinkelsesenheter. En vil sammen med denne kunne sende en BC identifikasjon, de individuelle LROC1 vil

så teste om denne er i samsvar med deres lokale klokke. En tilsvarende datapakke med også RoI inkludert, vil bli sendt til GLVL2 enheten.

En vil ha en trigger dødtid, $T1_{dead}$, som hovedsaklig vil komme av at en har overflyt i buffere foran prosesser i fronten av DAQ systemet som tar lengre tid enn en BC. En vil også ha bidrag til $T1_{dead}$ pga. at en del prosesser krever et gitt minste interval imellom to triggere.

2.4.6 Synkronisering.

Ved den høye kollisjonsfrekvensen til LHC, hvert 25 ns, vil tidssynkroniseringen i eksperimentet være en stor utfordring. Hver gang det skjer en ny kollisjon vil de kollisjoner som har skjedd før, 25, 50, 75 ns. osv. fortsatt ha innvirkning på eksperimentet. Partikler vil fortsatt være inne i detektoren, og ved *loopers* inne i innerdetektoren vil det ta opptil 5-8 BC før de ikke gir signaler i detektoren lenger. Detektorene vil fortsatt jobbe med å behandle pulsene fra forrige kollisjon når neste kommer, og signalene vil fortsatt være på vei i kablene ut av detektorene når det påtrykkes nye signaler, figur 2.9 kan illustrere denne overlappingen [1].



Figur 2.9: *Tidssynkronisering i et LHC eksperiment.*

Et annet stort problem vil være å koble rett hendelsesnummer til rett signal. Dette kan bli gjort ved å avpasse lengden på signalkablene slik at tidsforsinkelsene blir like for hele eksperimentet og ved å bruke tidsforsinkelsesenheter. En må ha mulighet for å teste og kalibrere tidssynkroniseringen i DAQ systemet. Dette kan tenkes å gjøres ved å bruke test pulser påtrykt DAQ systemet, eller ved å foreta kalibrering ved lav luominisitet og kontinuerlig under kjøring av selve eksperimentet. Denne kalibreringen krever at en har mulighet for å justere tiden globalt mellom hver detektor og finjustere tiden inne i FE elektronikken på deldetektor nivå.

Det vil være viktig, i størst mulig grad, å la tidspunktet for når data ble registrert av en deldetektor følge dataene igjennom hele DAQ systemet. En vil på denne måten ha et tidspunkt

knyttet til dataene og mulighet til å kontrollere synkroniseringen i DAQ systemet.

For lagene ved r=200mm og r=300mm vil den største tidsforskjellen innvendig være ca. 7 ns (2m/c). Den største tidsforskjellen innen en detektormodul vil være ca. 0.2 ns. (0.06m/c). Dette fører til at en bør ha et tidsjusteringssystem innen hver DM som kan gi en forsinkelse på 0-7 ns. og en oppløsning ned mot 0.2 ns. En kan da beregne tidsavviket ved å justere tiden i små step over en gitt tid, og lage en statistikk over signalverdien som en funksjon av tidsjusteringen. En vil da ha muligheten til å finne tidspunktet der FE-elektronikken bør sample for å få det beste signalet.

2.4.7 Feilbehandling.

Den mest fleksible løsningen er å introdusere lokal dødtid og lokale data-tap. Dette betyr at de enkelte subdeler vil rapportere feil, men sende ut data med status i størst mulig grad det er mulig. GLVL1 blir kun undertrykt etter en nøye overveid statusvurdering av en intelligent prosessor høyere opp i DAQ systemet. Dette kan f.eks. oppstå når det er for lite intervall mellom to triggere. En må da vurdere fordelene ved lokal dødtid mot et mere synkront system som diskutert tidligere.

Ved feil eller fulle buffere ut av LVL1 må LROC1 ha mulighet til å kunne rapportere dette til nivå 2 bufferets LROC2. Dette gjøres enklest ved at LROC1 sender ut en tom datapakke med feilmelding.

Prosesser som er avhengige eller venter på data fra en annen prosess må ha en avbruddsprosedyre som tvinger prosessen til å forstsette etter en gitt tid selv om den mangler data.

Undertrykkingen av GLVL1 vil komme fra GLVL2 eller spesielle avbruddsprosessorer koblet til samme nettverket som GLVL1 og GLVL2. Bestemmelsen bør bli basert på oppsamlet informasjon om status for hver enkelt subdel. På denne måten vil den synkrone delen av eksperimentet være skilt fra den asynkrone og individuelle tilfeldige feil vil ikke automatisk føre til dødtid for hele eksperimentet. Det vil også gi, og kreve, en større grad av mulighet for monitoring og kontroll av feilbehandlingssystemet.

En annen feil oppstår når en LLVL1 prosessor ikke kan gi brukbar informasjon til trigger avgjørelsen. Og i noen tilfeller vil ikke en subdel kunne gi informasjon til 2 og 3 nivå triggerne. I det siste tilfellet bør nivå 1 informasjon bli brukt og nivå 1 bestemmelsen bør bli sendt ut.

2.4.8 Hendelsesoppbygning

Hendelsesoppbygningen er meget arkitekturavhengig og flere løsninger blir vurdert. Systemet som foretar hendelsesoppbygningen vil være i en kontinuerlig utvikling samtidig som nye krav kommer frem og ny teknologi blir utviklet. De fleste løsningene er bassert på kommersiell hardware eller en kombinasjon av kjente systemer. Detaljene til delene av DAQ-systemet som omfatter hendelsesoppbygningen er derfor ikke beskrevet i denne oppgaven.

2.5 Innerdetektoren.

Figur 2.10 viser ATLAS inner-detektor [17] [12] som vil være 6.8 m lang og ha en ytre radius på 1.15 m. Hovedhensikten med innerdetektoren er å kunne rekonstruere ladede spor, gjøre en presis måling av massefart (og ladning) for leptoner, og identifikasjon av elektroner. For 2 nivå triggeren vil det være viktig med en god spor-rekonstruksjon. Ved høy luminositet vil en ha følgende mål:

- Rekonstruksjon av ladede isolerte høy P_T spor. Lepton identifikasjon ned til ≈7 GeV og i området $\mid \eta \mid < 2.5$ er målet.

- De isolerte elektronene som en ser etter ved LHC vil være få i forhold til QCD jet-strukturer (forhold ca. 10^{-5}). En vil for hvert isolert elektron ha ca. 10^3 triggere pga. jet-strukturer (bakgrunn $\approx \pi^0$) etter en kalorimeter-trigger. Vha. innerdetektoren kan bakgrunnen reduseres kraftig. En kan gjøre det meste av denne reduksjonen ved å se om partikkelsporet og massefart i innerdetektoren faller sammen med energiavsetning og posisjon til klusteret i kalorimeteret. Dette vil bli gjort allerede i trigger 2 algoritmen.

- Måle massefart for leptoner. Ved par-produksjon av tunge W bosoner vil en ha henfall til likt ladede lepton par. Innerdetektoren må kunne detektere ladningen til elektronet med $P_T \approx 500 GeV/c$ (som tilsvarer P_T oppløsning ≈ 30% ved $500 GeV/c$) for å skille disse fra $t\bar{t}$ henfall til motsatt ladete lepton par.

- Rekonstruksjon av lave P_T ($P_T > 2 GeV/c$) tett ved en høy P_T lepton kandidat.

  Leptonkandidater som kommer fra b henfall bakgrunn, kan reduseres ved å rekonstruere ladede spor i nærheten av leptonkandidaten.

  Bakgrunn fra lav P_T elektronpar fra Dalitzhenfall ($\pi^0 \longrightarrow (e^+e^-\gamma)$) kan reduseres.

Ved lav luminositet ($\approx 10^{32} cm^{-2} s^{-1}$) vil målet for innerdetektoren være å gi gode målinger av støt parameteren, se kapittel 2.5.1, for b, τ identifikasjon og generelt maksimal presisjon og oppløsning.

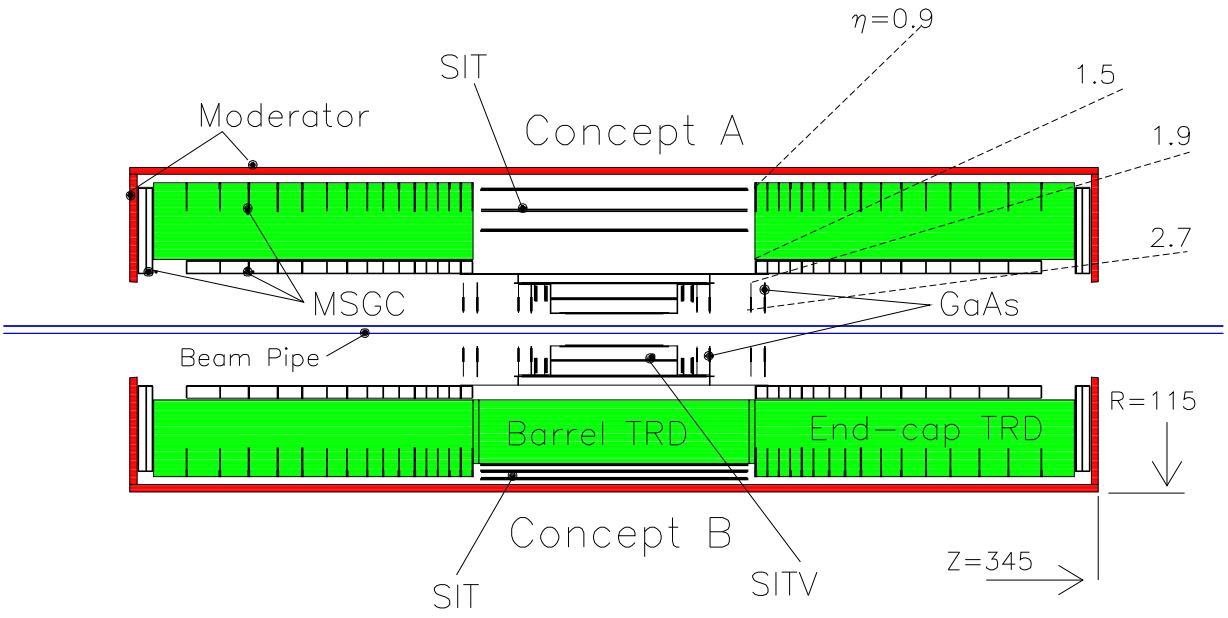
Kravet til nøyaktighet ved måling av leptoner, gjør at en må ha en presisjon på $< 20 \mu m$ for detektoren nærmest partikkelstrålen og ca. $< 60 \mu m$ der kalorimeterene begynner.

Innerdetektoren må også være laget slik at den kan tåle den høye strålingsintensiteten som vil være tilstede rundt kollisjonspunktet. Fra kalorimetrene vil en ha en neutron flux som i stor grad vil gi strålingsskader i sensorerene og elektronikken til innerdetektoren. En vil derfor legge et ca. 5 cm tykt lag av polythylen som moderator rundt innerdetektoren. Denne moderatoren vil redusere mengden neutroner tilbake til innerdetektoren.

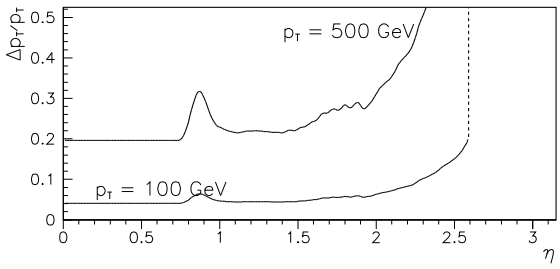
Målsetningene ovenfor vil bli best oppfylt med en kombinasjon av diskrete og kontinuerlige lag av spordetektorer. En må ha presis diskret sporgjenkjenning for å oppnå en massefart oppløsning på 30% ved $P_T = 500 GeV/c$. For rekonstruksjon av lave P_T spor vil kontinuerlig sporgjenkjenning være nødvendig. Figur 2.11 viser en simulering av massefart oppløsningen for $100 GeV/c$ og $500 GeV/c$ spor gjort for innerdetektoren.

Innenfra vil detektoren grovt beskrevet bli bygd opp av lag med pixel detektorer, dobbeltsidige silisiumstripe-detektorer eller enkeltsidige silisiumstripe-detektorer. En av grunnene til at en bruker pixeldetektorer er at disse vil være strålingssterke og ha en god punktoppløsning. Silisiumstripe-detektorene er noe rimeligere og vil gi tilstrekkelig oppløsning ved en større radius. En vil trolig bruke ladningsdeling mellom stripene for å oppnå en høyere oppløsning enn utlesnings oppløsningen. Hensikten til stripene og pixelene vil være å finne nøyaktig partikkel treffpunkt og start på partikkelspor, gi grunnlag for målinger av støt parameteren og evt. posisjon for andre partikkel-henfall (*secondary vertex*).

Mellom silisiumdetektoren og kalorimeteret vil det ligge Transition Radiation Tracker (TRT). TRT er flerlags strå-kammer som vil øke presisjonen og forbedre muligheten til elektron identifikasjon. TRT vil typisk bestå av 2*950 mm lange strå med en diameter på $40 mm$ for rekonstruksjon i r-ϕ. TRT vil gi et økt antall lag, og tetthet, av detekterende materiale i innerdetektoren,



Figur 2.10: *Utlegg av Innerdetektoren.*

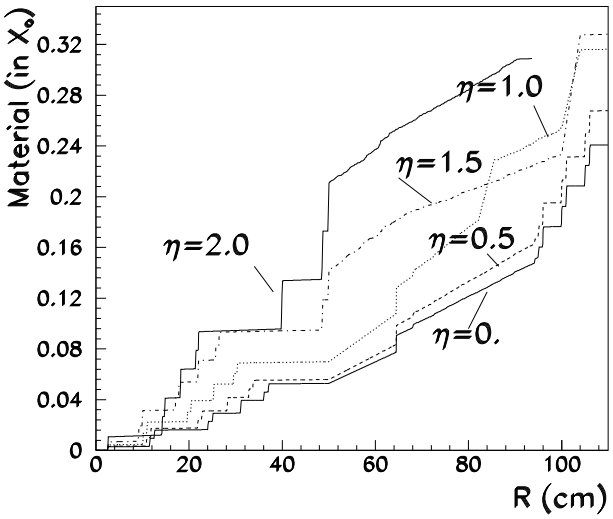


Figur 2.11: *Massefart oppløsning i innerdetektoren.*

som vil føre til en økt undertrykking av bakgrunn og forbedret sannsynlighet for å finne partikkelspor.

Tabell 2.4 viser de tekniske spesifikasjonene for innerdetektoren.

En vil prøve å minimalisere strålingslengden. Figur 2.12 viser et estimat for hvordan den radielle materialmengden vil være, vist som strålingslengde for forskjellige η. Denne fordeling



Figur 2.12: *Radiell fordeling av materialtetthet, målt i strålingslengde, for innerdetektoren.*

er gjort med de nevnte detektorer, deres bærestruktur, kabling og kjøling. Type og konstruksjon av hver enkelt detektor er gjort med tanke på å minimalisere strålingslengden tett ved kollisjonspunktet, samtidig som en har beholdt en geometrisk stabilitet i konstruksjonen.

Materialet i detektoren vil føre til bremsestrålingen som vil gi en redusert oppløsningen av massefarten til partikkelen. Eksperimentets effektivitet til å skille elektroner og fotoner vil derfor også bli redusert.

Bremsestrålingen vil også gjøre det vanskeligere å ekstrapolere partikkelspor utenfra og innover mot sentrum av innerdetektoren.

For å redusere *pile-up* ved rekonstruksjon av spor ved høy luominisitet vil en grovt bruke følgende prosedyre:

- Strådetektorene vil gi et sett av partikkelspor-kandidater i r-ϕ planet.
- Disse partikkelspor kandidatene blir så ekstrapolert til de nærliggende diskrete detektorene som har 3-D oppløsning. Ved høye P_T spor vil det ved denne ekstrapoleringen være mulig å finne alle treff i de diskrete detektorene. Dette kan gjøres ved å følge et $\approx 500\mu m$ bredt bånd i r-ϕ planet (r-ϕ for tønna og Z-ϕ for ende stykkene). De diskrete detektorene vil da gi oss en presis 3-D partikkelbane.
- Ved å ekstrapolere sporet mot de innerste silisiumlagene og gjøre en sportilpasning kan eventuelle sekundære henfallspunkter påvises.

Detector	Type	Coord.	Area/Vol	σ	No.	Costs			Channels		
						unit	total	detector	unit	total	detector
			m^2, m^3	μm		MCHF	MCHF	MCHF			
Forward											
MSGC	Full wheel	ϕ, u, v	2.64	60	6	0.83	4.98		274,310	1,645,860	
	Hollow wheel	ϕ, u, v	1.21	60	26	0.34	8.84	13.82	113,660	2,955,160	4,601,020
TRT	7cm high dens.	ϕ	0.16	150	6	0.27	1.62		6,825	40,950	
	10.2cm high dens.	ϕ	0.24	150	8	0.39	3.15		9,945	79,560	
	7cm low dens.	ϕ	0.16	150	16	0.18	2.88		4,550	72,800	
	10.2cm low dens.	ϕ	0.24	150	10	0.26	2.60	10.25	6,630	66,300	259,610
SCT	small wheels	u, v	0.09	20	4	0.37	1.48		75,000	300,000	
	large wheels	u, v	0.13	20	4	0.50	2.00	3.48	100,000	400,000	700,000
Barrel											
SCT	11.5cm pixel	ϕ, z	0.52	20	1	2.78	2.78		50 M	50 M	
	14.5cm pixel	ϕ, z	0.77	20	1	3.62	3.62	6.40	50 M	50 M	100 M
	20cm 2-sided	ϕ, z	1.36	20	1	3.36	3.36		285,000	285,000	
	30cm 2-sided	ϕ, z	2.73	20	1	6.77	6.77	10.13	571,650	571,650	856,650
	52cm 1-sided	ϕ	6.21	60	1	2.44	2.44		274,300	274,300	
	53cm 1-sided	z	6.33	300	1	2.48	2.48		279,600	279,600	
	79cm 1-sided	ϕ	9.43	60	1	3.68	3.68		416,540	416,540	
	81cm 1-sided	z	9.67	300	1	3.98	3.98		427,140	427,140	
	105cm 1-sided	ϕ	12.53	60	1	4.91	4.91	17.49	553,470	553,470	1,951,050
TRT	outer	ϕ	2.33	150	1	1.45	1.45		62,170	62,170	
	inner	ϕ	1.65	150	1	1.12	1.12		47,830	47,830	
	transition	ϕ	0.12	150	2	0.07	0.14	2.71	8,075	16,150	126,150
TOTAL								64.28			

Tabell 2.4: *ATLAS Inner Detektor Spesifikasjoner.*

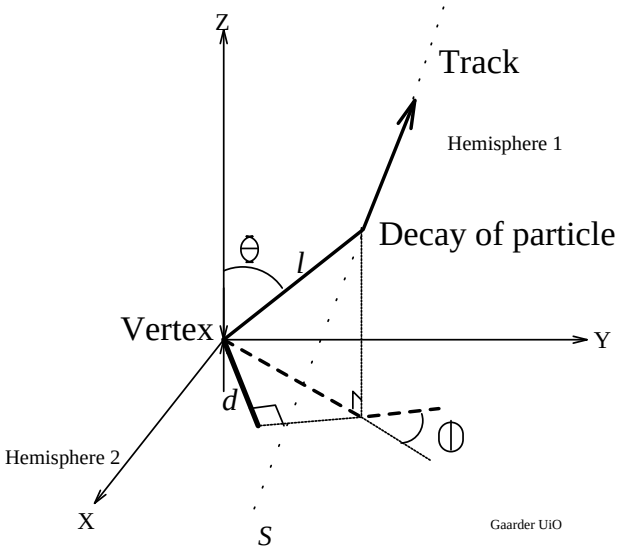
2.5.1 Impactparameter.

Impactparameter-måling brukes til å identifisere partikler som henfaller i et sekundært vekselvirkningspunkt. F.eks. vil en b kvark etter kort tid $\tau \sim 1.2ps$, og en lengde l, henfalle til andre partikler. Ved å projeksere partikkel sporene inn i X-Y planet vil en fjerne problemet med at scaling variablene (x) i kollisjonen er forskjellige langs Z-aksen. Den korteste avstanden fra partikkelsporet til *vertex* i X-Y planet kalles da impact parameteren d, se figur 2.13.

$$d = l * sin\Theta * \cos\Phi \tag{2.7}$$

der $l = \beta\gamma c\tau$.

Ved rekonstruksjon av en hendelse vil en ikke kjenne l eller punktet hvor partikkelen henfalt, dvs. vinkelen Θ er ukjent. En må derfor rekonstruere, ved hjelp av linjen S, dens impactparameter. Denne eksperimentalt målte impact parameter kan så sammenlignes med en forventet impact parameter fordeling som f.eks. er beregnet vha. MC simuleringer.



Figur 2.13: *Impactparameteren d.*

2.6 Elektromagnetisk og hadronkalorimeter.

En har kommet frem til følgende kriterier for et design av et kalorimetersystem for LHC:

- God elektromagnetisk (EM) kalorimetri for identifikasjon og måling av energien til fotoner og elektroner i energi området fra 7-10 GeV opp til noen TeV. Effektiviteten for å detektere $H \longrightarrow ZZ^* \longrightarrow 4e$ øker raskt for en lett Higgs når E_T-kuttet reduseres. På den andre siden ved tunge $Z' \longrightarrow ee$ må en kunne måle elektroner opp mot 3 TeV.
- Et hermetisk lukket system for måling av energibalansen i hendelsen.
- Systemet må kunne fungere og trigge ved luminositeter over $10^{34}cm^{-2}s^{-1}$.

- Systemet må kunne tåle den oppsamlete strålingsdosen over 10 års drift.

Sentraltønne og *end-cap* delene ($\mid \eta \mid < 3$) består av et LAr [24] elektromagnetisk (EM) kalorimeter. Utenpå EM kalorimeteret ligger et hadron kalorimeter, tykt nok til å fange opp de høy energetiske jet-strukturene produsert ved LHC. I forover retningen ($3 < \mid \eta \mid < 5$) hvor fart og strålings hardhet er kritisk, vil en separat detektor med noe lavere yteevne bli brukt.

Det vil i *em* kalorimeteret være integrert en *preshower* [15] hovedsaklig for å kunne skille e, π^0.

For hadron kalorimeteret er det foreslått 3 forskjellige løsninger, jern-LAr , jern-scintilerende fibrer eller jern-scintilerende plater.

Kalorimeteret vil ved en elektron kandidat gi en nivå 1 trigger. Dette vil føre til at det blir merket av et RoI område, i $\eta - \phi$ rom. Størelsen på dette området vil typisk være $\Delta\eta * \Delta\phi = 0.2 * 0.2$.

2.7 Muon-detektor.

Muon detektoren vil ligge ytterst og ha som oppgave å detektere og måle massefart for tunge leptoner. Noen av målene for muon detektoren vil være:

- Kunne gi en nivå 1 trigger.
- God massefart oppløsning i området 10-3000 GeV.
- Dekke et område opp til $\mid \eta \mid = 3$.
- Kunne fungere selvstendig ved en luminosistet på $\approx 2 * 10^{34} cm^{-2} s^{-1}$.
- Sikker spor-rekonstruksjon.

Muon detektoren er plassert inne i en superledende toroidmagnet. Magnetfeltet fra denne vil føre til en avbøyning av ladede partikler i r-Z.

Inne i magnetfeltet og utenfor vil det ligge flere lag med detekterende materiale for rekonstruksjon av partikkelspor.

2.8 Magnetiske felt.

Ved å finne den krumning en ladet partikkel beveger seg med i et hommogent magnetfelt, vil en kunne beregne dennes massefart.

$$P_{-} = mv = rqB[kgm/s], eller P_T = \frac{r}{0.3B}[GeV/c] \tag{2.8}$$

Magnet konfigurasjonen i eksperimentet har en stor innvirkning på hele detektor konseptet. Ved ATLAS er det valgt å ha en superledende solenoid i innerdetektoren, og en toroid magnet i muon detektoren. Dette fører til at en vil ha:

- Et kraftig homogent felt i innerdetektoren
- Et robust selvstendig muon detektor system med stor akseptanse.

Solonoiden vil være en ett-lags superledende spole, 1230 mm i radius med 6300 mm lengde, som vil gi et tilnærmet homogent felt på 2T for avbøyning av ladede partikler i r-ϕ i innerdetektoren. Strømmen vil være 7500 A og en lagret energi på 50 MJ. Det 2T sterke magnetfeltet vil og føre til at ladede partikler, *loopers* $P_T \widetilde{<} 0.8 GeV$, blir fanget inne i innerdetektoren. Dette vil føre til en økt flux av partikler i innerdetektoren, figur 2.15 viser dette ved forskjellig magnetfelt.

Den ytre toroiden vil bli satt sammen av et antall superledende spoler. Figur 2.5 viser hvordan 12 spoler kan bli lagt rundt tønnen slik at de tilsammen danner en hel toroid. Magnetfeltet vil da ligge inne i de 12 spolene uten å gi en forstyrrende effekt inne i detektoren. Antallet spoler vil trolig bli redusert til 8 for å minke kostnadene.

2.9 Partikkelflux og mengden av detektortreff.

Partikkelfluxen i detektorene vil ha to viktige konsekvenser for eksperimentet:

- Partikler som skader detektorene og deres tilhørende elektronikk, gjør at en må bruke spesielle konstruksjoner og elektronikk for å oppnå en akseptabel levetid for eksperimentet ($\approx$ 10 LHC år).
- Høy datarate som fører til stor belastning på utlesning og trigger-systemet.

Den ladede fluxen for LHC ved en luminositet $\approx 1.7 * 10^{34} cm^{-2} s^{-1}$ vil være $N \approx 2 * 10^9 / r^2 cm^{-2} s^{-1}$ ved en radius r fra stråleaksen (Z).

Fra formel 2.8 har vi at ladede partikler med lav P_T vil være fanget inne i magnetfeltet. Partiklene vil gå i bane rundt Z aksen og øke mengden av treff i detektorene.

Dataraten avgjøres av:

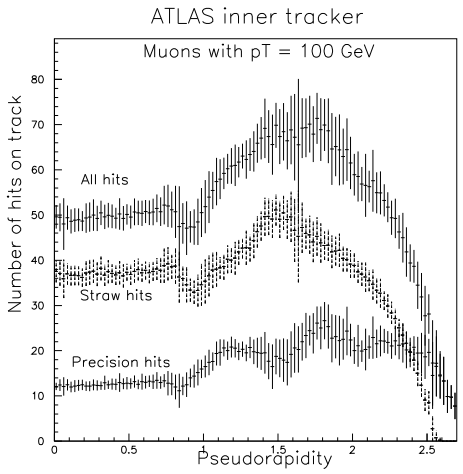
- Frekvensen av partikkel-treff i detektorene.
- Datamengde/treff.
- Signal/støy forholdet.

Bidraget av signaler fra støy er lite kjent og det må gjøres videre forskning på dette. Dette kan først gjøres presist når prototyper av utlesningselektronikken er ferdig, og en kan måle på denne i testoppsett. Inntil dette er gjort må en estimere støyen best mulig og pessimistisk forvente at den gir en betydelig mengde data som DAQ systemet må lese ut .

2.9.1 Antall treff i innerdetektoren.

Konstruksjonen av innerdetektoren er optimalisert slik at det alltid er mulig å beregne banen til en partikkel, ved max luminositet, fra minst 4 punkter vha. de diskrete detektorene uten å ta i bruk detektorene innenfor r=200 mm. En vil også ha ca. 40 signaler (minus ineffektivitet) fra de kontinuerlige detektorene for hvert partikkel spor [16]. Figur 2.14 viser gjennomsnittlig antall treff i innerdetektoren fra simulering av et muon spor på 100 GeV.

Figur 2.15 [13] viser simuleringer gjort for en sentraldetektor ved et LHC eksperiment. Simuleringene er gjort med GEANT og 15 ns intervall mellom partikkel-kollisjonene og ved forskjellige magnetfelt. Lagene dekker området $\mid \eta \mid = 2$. En kan gjøre en del betraktninger og estimater av dataratene for de innerste lagene med silisium-striper:



Figur 2.14: *Gjennomsnittlig antall treff fra et rekonstruert muon spor på 100 GeV i innerdetektoren. Hver måling (R-ϕ, z eller stereo) er regnet som et treff.*

- Figur 2.15a viser gjennomsnittlig antall treff/lag for en minimum bias hendelse. En kan se at *loopers* øker antallet treff ved økende magnetfelt, spesielt ved en radius mindre enn 0.3 m.

- Fra figur 2.15b kan se at sannsynligheten for signal er størst ved $\eta \approx 0$, og vil være ca. 0.25% ved 20 minimum bias hendelser og 2 T magnetfelt.

- Ved r=0.25 m vil en silisiumdetektor som dekker $\mid \eta \mid < 2$ ha et areal på $2.8m^2$, 1.1 mill kanaler (pitch $50\mu m$, lengde $5cm$). Fra 2.15 a) kan en se at en vil ha ca. 50 treff pr. lag for hver minimum bias hendelse, som gir oss ca. 1500 treff ved 30 minimum bias hendelser. Om en regner med at et treff vil gi signal i to kanaler, vil sannsynligheten for signal i en kanal bli ca. 0.27% i snitt over hele laget ved 2 T magnetfelt (0.18% ved 20 mbias).

- Tilsvarende kan en fra kapittel 1.3.2 finne at en vil ha ca. **720** ladede spor i innerdetektoren ved $dn^{\pm}/d\eta \approx 6$, $\mid \eta \mid = 2$ og 30 minimum bias hendelser. Ved å sette magnetfeltet til 0 kan en sammenligne med figur 2.15a, der en finner at det vil være ca. 33 treff pr. minimum bias hendelse ved r=0.25 m, eller ca. **990** treff ved 30 minimum bias hendelser. Ved å ekstrapolere 2.15a mot en mindre radius kan en se at disse resultatene stemmer bra overens. Evt. viser dette at $dn^{\pm}/d\eta \approx 6$ kan være noe for lavt og optimistisk ved voksende radius.

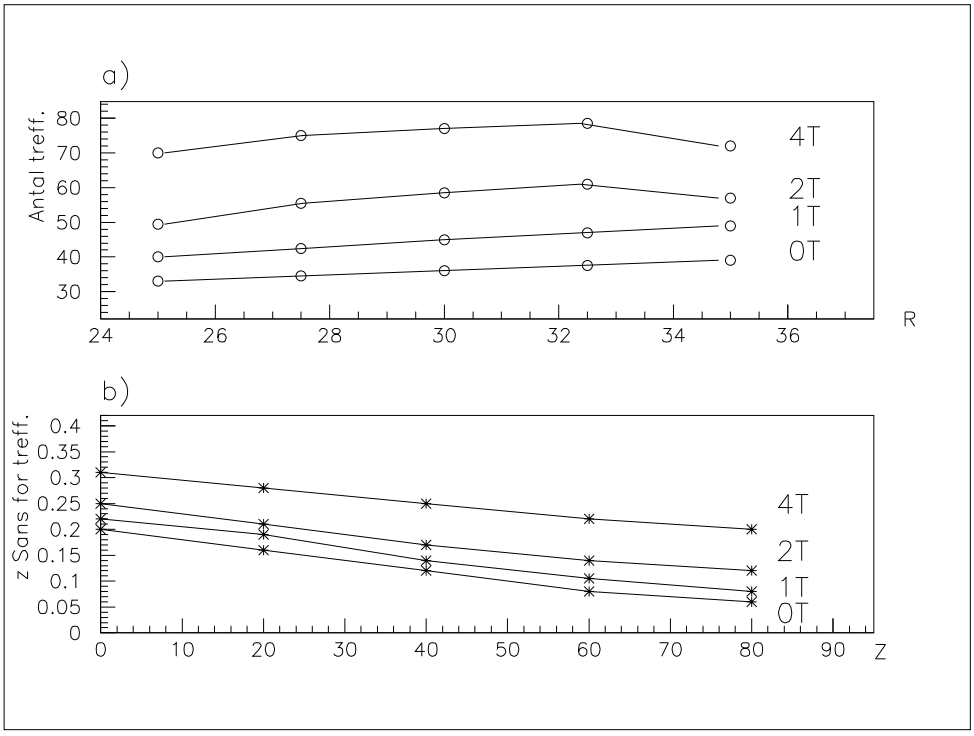
- Fra figur 2.10 ser en at lag r=0.2 m og r=0.3 m vil dekke området $\mid \eta \mid \approx 1.5$ og fra tabell 2.4 ser en at det vil være 285000 striper ved r=0.2 m og 571650 striper ved r=0.3 m, ved $100\mu m$ readout pitch og 50 mm lange striper. Om $dn^{\pm}/d\eta \approx 6$ vil det gå 540 partikkel spor gjennom hvert av lagene ved 30 minimum bias. Dette vil gi oss en sannsynlighet for signal i en kanal, om et treff dekker en stripe, på 0.2% for r=0.2 m og 0.1% for r=0.3 m. Dette vil være noe optimistisk, spesielt for r=0.3 m, pga. at en ikke har regnet med magnetfeltet eller økningen av treff pga. radius >0 og regner med at et treff kun gir signal i en stripe.

Det vil være rimelig å anta sannsynligheten for signal i en kanal pga. en ioniserende partikkel, ved $\eta = 0$, til å pessimistisk være som tabell 2.5 viser. Det er her tatt med både gamle og nye

LHC parametere. En antar her at et partikkelspor i snitt vil dekke et område på $100\mu m$, dvs. 2 stk. $50\mu m * 50mm$ striper. Geometrien på lagene er noe forandret ved de nye kollisjonsratene. En kan anta at økningen i sannsynlighet for treff, når en går fra gamle til nye kollisjonsrater og luominisiteten holdes konstant, vil være tilnærmet $\frac{25ns}{15ns}$.

Radius	66 MHz	40 MHz
0.2 m	0.3%	0.45%
0.3 m	0.2%	0.30%

Tabell 2.5: *Antatt sannsynlighet for signal i en Si stripe.*



Figur 2.15: a) *Gjennomsnittlig antall treff pr. sylindrisk lag med en rapidity dekning på* $\mid \eta \mid = 2$ *som funksjon av radius ved et magnetfelt på 0, 1, 2, 4 T.* b) *Sannsynligheten for signal i en kanal ved r=0.25 m som funksjon av z og et treff dekker to striper (*$50\mu m$ *pitch* $*50mm$*) ved 20 minimum bias hendelser.*

Pga. jet-strukturer vil ikke sannsynligheten for signal i en kanal være jevnt fordelt i detektoren, som antatt for tabell 2.5. Dette gjør at den lokale sannsynligheten (innenfor klusteret) vil være høyere enn 2.5 viser.

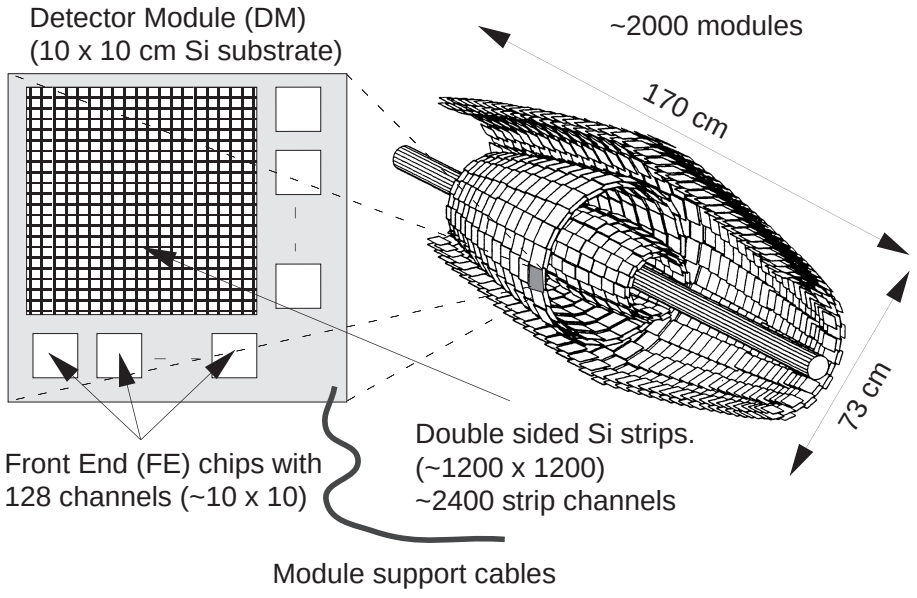
Kapittel 3

Silisium-submodul.

Som beskrevet i kapittel 2.4 vil en deldetektor bli bygd opp av et antall funksjonelt like submoduler. Disse submodulene vil ofte basere seg på forskningsarbeid fra forskjellige forsknings og utviklings grupper, RD-19 for pixel detektorer, RD-2 og RD-20 for silisium detektorene. En vil her i hovedsak behandle de silisiumstripe baserte submodulene. Disse subdelene, heretter kalt Detektormoduler (DM), er basert på silisiumstripe detektorer og *Front End* (FE) elektronikk utviklet innen RD-20.

3.1 Detektormodulen.

Lagene vil kunne bli bygd opp av detektormoduler lagt som et skall, med en liten overlapping, rundt kollisjonspunktet som figur 3.1 viser.



Figur 3.1: *Detektor Moduler lagt i skall rundt strålerøret (her en gammel lay-out).*

DM vil være en selvbærende enhet som på en forholdsvis enkel måte kan monteres/demonteres

fra bærestrukturen. En kan evt. montere sammen flere DM i lengden til en enhet, dette kan f.eks. gjøre det enklere å tilkoble kjølingen. Detektorarealet til DM vil være gitt av størrelsen på den silisium skiven som brukes, typisk en 4" *wafer* som gir$(60 - 70mm^2)$. Utlesningselektronikken vil bli montert rundt kantene av selve silisium detektorene. Plasseringen av elektronikken avhenger av bærestruktur, kabling, båding, kjøling og andre faktorer som f.eks. geometrien på deldetektoren. Antallet koblinger/signal overføringer og deres følsomhet for støy vil være kritiske faktorer. Mulige plasseringer av elektronikken vil være:

- I plan med silisiumstripene og ca. 3 cm rett ut på 2 sider som figur 3.4 viser. Dette fører til at en får en oversiktlig DM med enkel tilkobling av kabler, men noe vanskeligere sammenkobling av modulene pga. at det blir 2 sider som stikker utenfor selve detektor flaten.

- En kan legge aluminiumstriper vinkelrett på silisiumstripene i den ene retningen, *fan-in*, som figur 3.2d viser. Dette gjør at en kan lese ut silisiumstripen fra siden, og muliggjør plassering av all FE elektronikk på samme side, evt. rett ovenfor på motstående side.

- Plassere elektronikken vinkelrett på sidene av DM, som figur 3.2a viser et eksempel av. Dette vil gi en mere kompakt og selvbærende løsning enn å la elektronikken være i plan. Ved å legge databussene, ledningsføringer og kontrollogikken på baksiden vil en få kort signalvei og god plass til signalføringene. Ulemper ved løsningen vil være problemer med å koble kjøling til brikkene og bånde elektronikken til silisiumstripene. En kan gjøre kjølingen noe enklere ved å ta ut signalene vinkelrett på silisiumstripene i en retning som figur 3.2b viser.

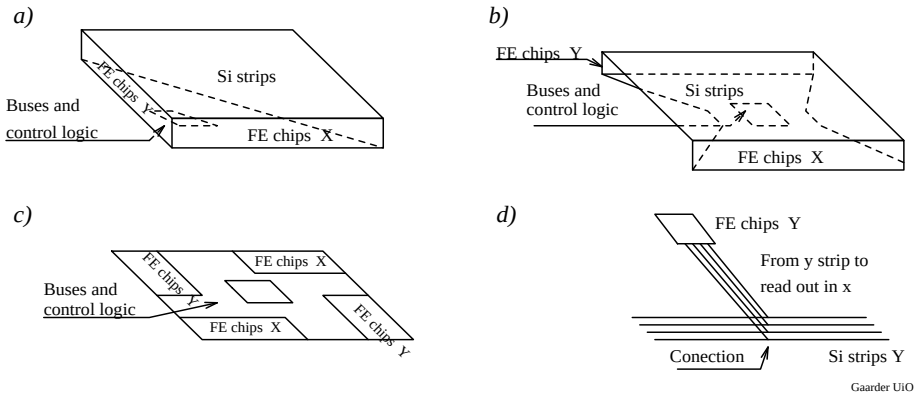
- Plassere elektronikken ovenpå silisiumstripene, ved f.eks. *flip chip* metoder. Dette vil gi en meget kompakt DM med lav masse og mulighet for kort kabling til utlesningslogikken som kan plasseres på eget substrat i sentrum. Figur 3.2c viser en mulig plassering av FE brikkene.

- La elektronikken, eller deler av denne, være på eget substrat som er montert uavhengig av silisiumstripene. Dette kan f.eks. gjøres vha. kapton folie. En vil muligens få lang kabling og problemer med støy ved denne metoden.

- En kombinasjon av disse løsningene.

Detektormodulen vil ha silisiumstriper på begge sider. Dette kan oppnås med to enkeltsidige detektorer rygg mot rygg eller med en dobbeltsidig detektor. En vurderer også å ha forskjellig utlesnings pitch i de to retningene, dette er nærmere vurdert senere. Figur 3.3 viser en mulig løsning med kevlar bærestruktur. DM består her av to adskilte silisiumskiver med evt. kjølemedium i det lukkede rommet mellom platene. Den ene silisiumskiven har ledere vinkelrett på silisiumstripene, *fan-in*, slik at stripene kan leses ut på siden av stripene og dermed muligjøre kjøling av alle brikkene på den ene siden. Utlesningslogikken for FE brikkene bør legges midt på side kanalen for å gi kortest mulig signaloverføringer. Denne løsningen vil gi en stiv selvbærende konstruksjon av DM. Et problem her vil være at trykk forandringer (f.eks. pga. kjøling) inne i hulrommet og utenfor DM vil gi påkjenninger og evt. geometrisk unøyiaktighet. En bør derfor ha avstivere imellom silisiumskivene for å forhindre denne effekten og få en maksimalt stiv konstruksjon.

Figur 3.4 viser et mulig utlegg av detektormodulen og hva denne evt. vil bestå av:



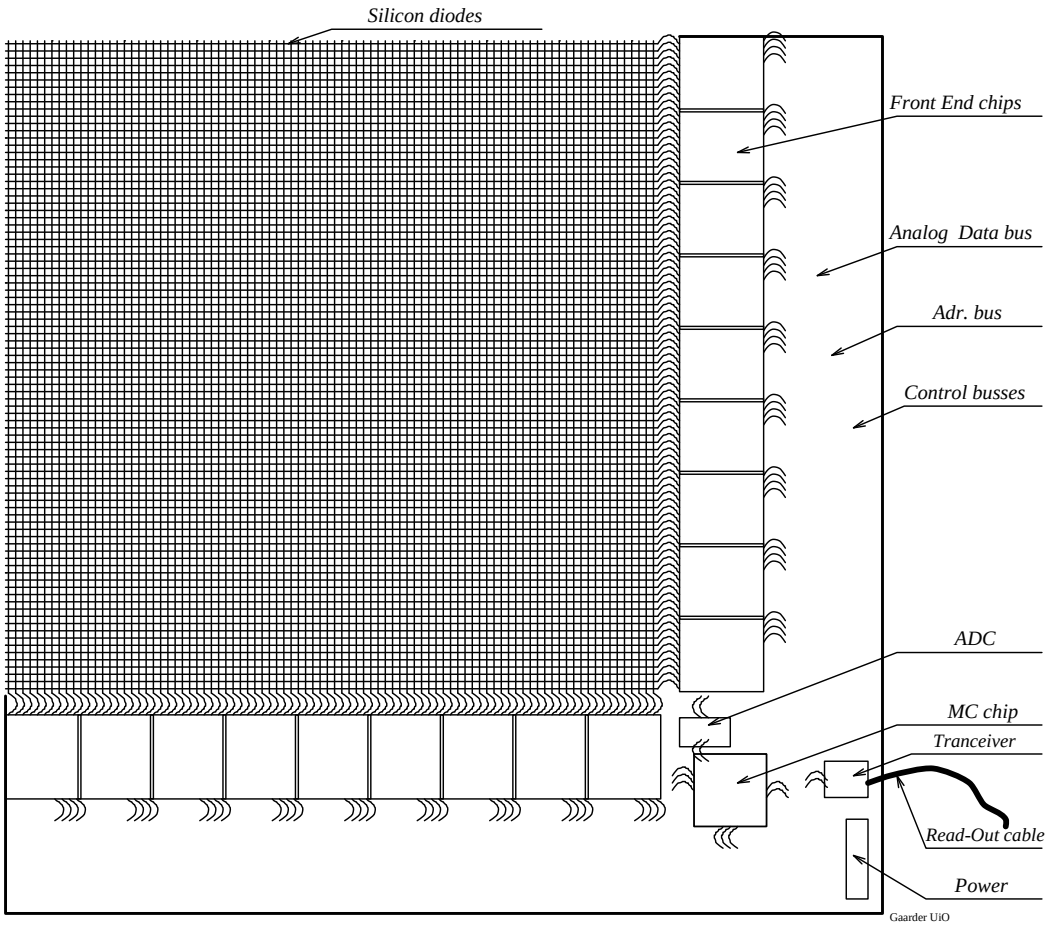
Figur 3.2: *Forskjellige utlegg av detektormodulen og signalføring på denne.*

- Silisium dioder (striper/kanaler) i to lag vridd 90°til hverandre.
- *Front End chip* (FE chip) som består av forsterker, *shaper*, ADB, APSP, diskriminator og DSR.
- Analog databuss til å overføre signalene til ADC'en.
- Digital buss til å overføre adresser, og evt. et antall kontrollsignaler.
- ADC.
- En *Module Control chip* (MC chip) som består av *Protocol Engine* (PE) med databuffer, og en *Control and Monitoring* (CM) modul.
- Optisk seriell tranceiver/receiver, og evt. en egen receiver til å ta imot klokke og trigger signaler.
- Tilkobling for strøm og spennings-referanser.

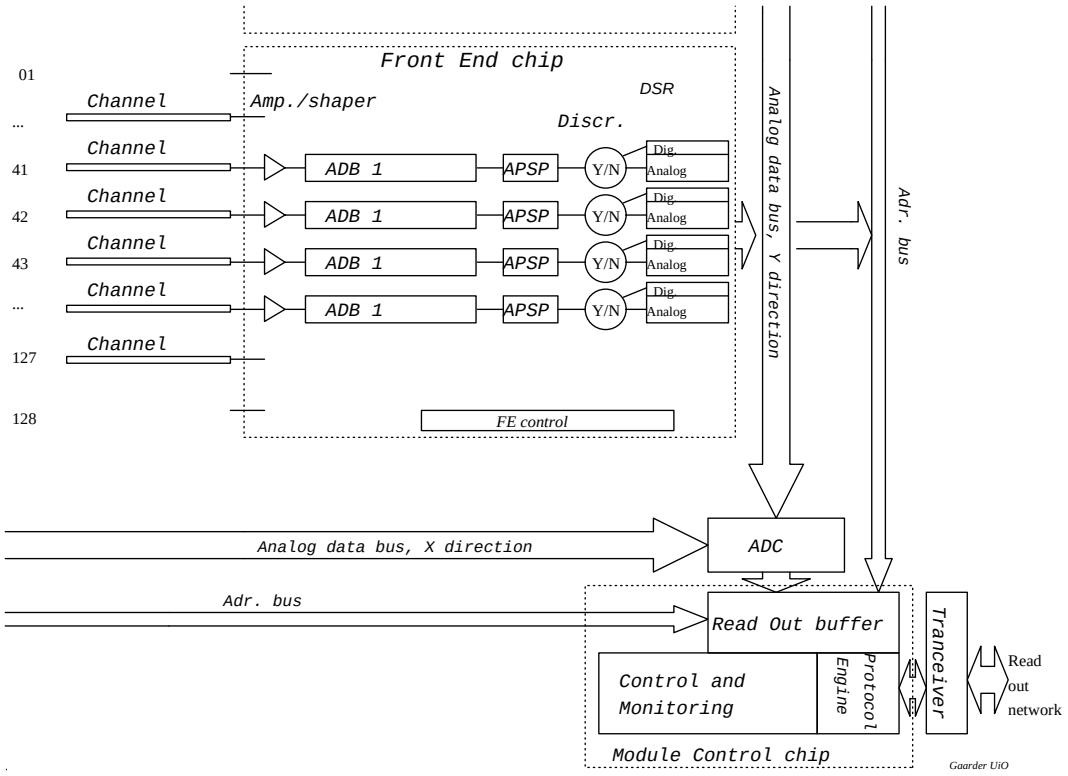
Figur 3.5 viser et forslag til arbeidsprinsippet for DM. Selve sensorene på DM består av silisium striper i X og Y retningen, f.eks. $60mm$ lange med $50\mu m$ detektor pitch. Til sensorene er det koblet et antall integrerte kretser, som hver fanger opp signalene fra 128 individuelle silisium striper. Når en partikkel går igjennom en silisium stripe får vi et elektrisk signal ut av kanalen. En forsterker koblet til kanalen hever signalet før det blir avlest for hver Δt og lagt inn i et *Analog Delay and Buffer* (ADB). Pga. at den globale første nivå triggeren vil ha en tidsforsinkelse ($t_{T1delay}$) trenger vi en pipeline (FIFO), slik at vi kan mellomlagre verdiene før vi får en avgjørelse (T1) om vi skal lagre verdien for senere utlesning. ADB fungerer både som en pipeline før T1 avgjørelsen og som et buffer for neste prosess. ADB har parallelt en kompleks digital logikk av registre som peker på de lagrete analoge verdiene i bufferet.

Hvis vi får en første nivå trigger så blir de tilhørende verdiene merket og lagret i ADB for senere å bli behandlet av *Analog Pulse Shape Processor* (APSP). APSP'en er basert på diskret tidsfiltrering og vektlegging av de merkete verdiene i ADB. APSP'en dekoder signalet og reduserer støy og pileup. Hele dekodingen tar en gitt APSP tidsforsinkelse ($\Delta t < t_{APSP} < t_{T1}$), som gjør at det må være et derandomizerende buffer (inkludert i ADB) foran.

Figur 3.3: *Utlegg av DM i to lag med bærestruktur. De forskjellige deler er angitt.*



Figur 3.4: *Detektormodulen og dennes komponenter.*



Figur 3.5: *Arbeidsprinsippet for DM.*

Signalverdien fra APSP'en vil bli lagt inn i *Derandomising and Sparsifying Readoutbuffer* (DSR). Dette er et kombinert buffer og en utlesningsmultiplekser ut av FE brikken.

Etter APSP'en er det en diskriminator som, etter en gitt referanseverdi, vil avgjøre om det var treff eller ikke i denne spesifikke kanalen. Om signalverdien er over den gitte grenseverdien vil en bit, både i kanalen med treff og nabokanalene, bli satt for å indikere at data der senere skal leses ut. Dette digitale bufferet vil operere parallelt med det analoge bufferet i DSR'en.

Nabokanalene av hovedtreffet tas med slik at en senere kan gjøre en beregning av tyngdepunktet av ladningen fra silisium stripene. DSR starter på det første bufferet med data og leser ut dataene videre syklisk fra kanal 1, brikke 1, og videre til siste kanal og brikke. Dette fører til at en leser ut hele hendelsen fullstendig før en begynner på neste. DSR vil bruke en DSR tidsforsinkelse, t_{DSR}, på å lese ut en verdi. Fra DSR'en overføres signalet til en *Analog Digital Converter* (ADC). Samtidig som den analoge verdien blir lest ut, vil digital informasjon f.eks. senterkanalnummer og brikkenummer bli lagt ut på en databuss, evt. holdt rede på av MC. ADC'en får data fra en og en kanal og bruker en ADC tidsforsinkelse, t_{ADC}, på å konvertere den analoge verdien til en digital representasjon. De digitale dataene blir lagt inn til *Protocol Engine* (PE) som både er et digitalt buffer og en generell data håndtering og pakkeenhet. DM vil også ha *Bus Snooker* (BS) som tar seg av den overordnede kontrollen av DSR'ene og dataflyten ut av brikkene. Parallellt til de primære funksjonene til DM vil en ha en *Control Monitoring* (CM) enhet som inneholder *slow control*, monitoring, test og kalibrering, initialisering etc. På utgangen av DM vil det være et nettverksinterface som gjør at en kan sende data inn og ut av detektor modulen.

3.2 De enkelte funksjoner på detektormodulen.

I de følgende underkapittler vil de enkelte deler til DM bli nærmere beskrevet.

3.2.1 Sensoren.

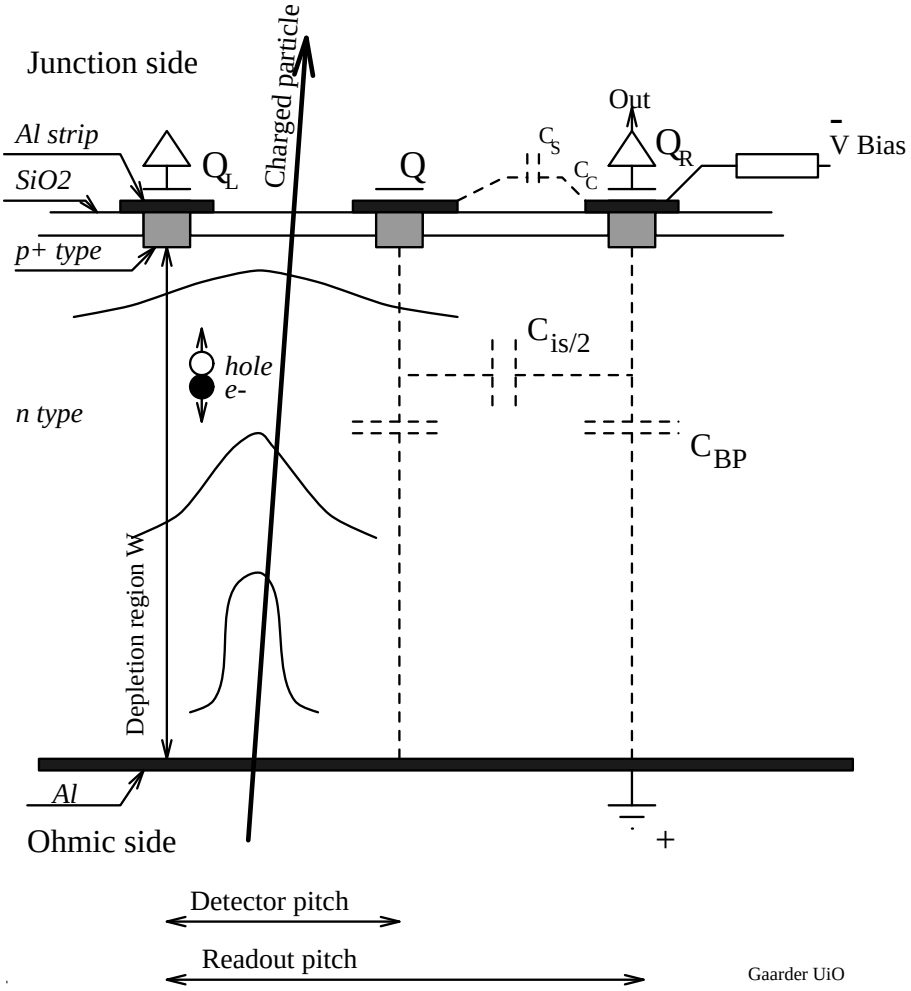
Det er vurdert å bruke dobbeltsidig eller enkeltsidige silisiumstripedetektorer som sensorer for presisjonsdelen av ATLAS indre detektor. Silisium stripedetektorer kombinert med pixel lag nærmere kollisjonspunktet vil gi en tilstrekkelig god undertrykkelse av falske spor og tilstrekkelig P_T oppløsning.

Figur 3.6 viser en enkeltsidig silisium detektor i snitt, med ladningsdeling. Detektoren består av n dopete silisium med sterkt p+ dopet striper. Figur 3.7 viser prinsippet for hvordan dioden virker; 3.7a p og n dopet silisium kobles sammen, 3.7b hullene og elektronene, dvs. majoritetsbærerne, vil diffundere til respektivt n og p siden og vi får en deplesjonssone W.

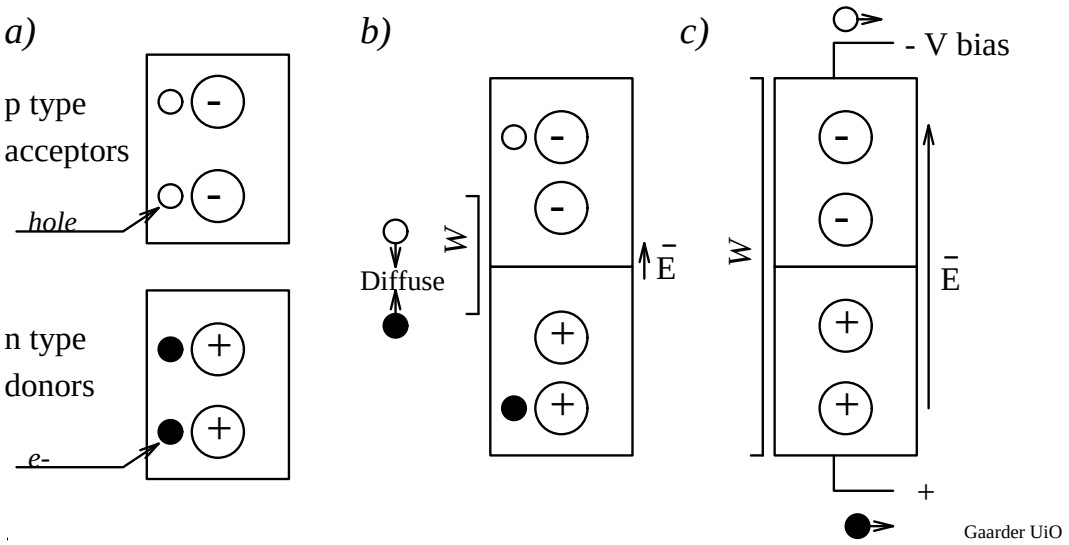
Ved å legge en reverskoblet forspenning over $p-n$ koblingen, 3.7c, vil de resterende majoritetsbærerene bli ledet ut av dioden og vi vil ikke lenger ha frie ladningsbærere inne i dioden, dvs. deplesjonssonen W vil gå over hele dioden.

Den eneste strømmen som da vil gå vil være termisk generasjonsstrøm I_s fra frie minoritetsbærere. Det er da forutsatt at feltet er under grensen for sammenbrudd og en ser bort fra effekter som forandringen i dopingen pga. stråling over tid osv.

En ladet partikkel som går igjennom silisiumet vil avsette energi pga. ionisasjon eller eksitasjon av silisiumatomene, og det kan bli dannet elektron-hullpar, dvs. frie minoritetsbærere. Det elektriske feltet som er påtrykt silisiumet fra forspenningen gjør at hullene og elektronene driver mot stripesiden (p) og den ohmske siden (n) respektivt.



Figur 3.6: *Enkeltsidig silisiumstripedetektor med ladningsdeling.*



Figur 3.7: *Prinsippet for silisium diodene, a) p og n dopet silisium, b) p og n silisiumet i likevekt etter sammenkobling, c) dioden med revers forspenning.*

Disse ladningene, Q_L, Q og Q_R, som er kapasitivt koblet til aluminiumstripen vil gi signal i forsterkerne[1]. Stripene vil ligge vinkelrett i forhold til hverandre på hver side av DM, evt. med en liten stereo vinkel ($\sim 3°$).

Pga. den tiden det tar å samle opp ladningen fra stripene vil signalet ut av diodene ha en hale på opp til 20 - 30 ns.

***Minimum Ionizing Particle* (MIP).**

En ladet partikkel vil avsette en energi i det medium den går igjennom som følger Bethe-Block formelen [29]. Den minste ladning en ladet partikkel i kan avsette i detektoren er kalt en *Minimum Ionizing Particle* (MIP). Ladningen vil variere fra 1 MIP til ca. 1.6 MIP avhengig av massefarten til partikkelen. Ladningen en partikkel avsetter vil øke proporsjonalt med strekningen partikkelen går i igjennom silisiumdetektoren. Den mest sannsynlige ladning vil være ca. 22400 elektroner for en ladet partikkel ved en detektortykkelse på $300\mu m$ (90°). Pga. effekter som Landau fordeling av energien, ladningsdeling på 2 striper, ballistisk underskudd, *trapping* etter store doser og tap pga. kapasitiv kobling til baksiden vil ladningen som må kunne observeres være mye lavere ($\approx 5000e^-$) [37].

Detektorens pitch.

Stripeavstanden (detektorpitchen) for en DM vil være senteravstanden mellom to striper. Avstanden mellom utlesningsstripene (utlesningspitchen) for en DM vil være senteravstanden mellom stripene som leses ut. Figur 3.6 viser en enkeltsidig silisiumstripe detektor med en mellomliggende stripe for hver stripe som leses ut og dennes stripeavstander.

Det er i denne oppgaven for det meste regnet med en avtand på $50\mu m$ mellom utlesningsstripene, selv om en trolig vil øke denne til nærmere $100\mu m$ eller mere for å redusere antall utlesnings kanaler.

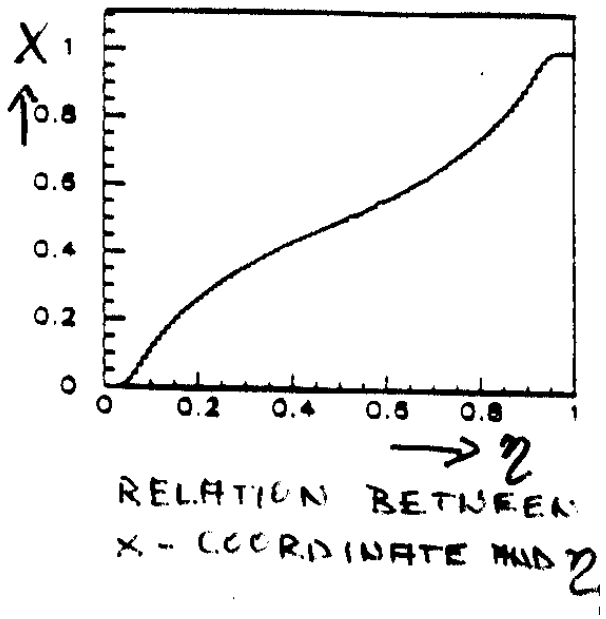
[1] Q fra mellomliggende stripe indirekte pga. $C_{is/2}$.

Kapasitiv kobling.
For å forhindre termisk generasjonsstøy I_s, og forspenningen i å gå direkte til forsterkerne vil en legge et tynt oksydlag mellom aluminiumsstripene og de p dopete stripene. Dette vil gi en kapasitiv kobling, C_C, mellom stripene og tilkoblingspunktene, og resultere i en AC kobling mellom stripene og forspenningen/forsterkeren. Referanse [26] viser produksjonsmetode og tester av en detektor av denne typen.

Kapasitiv ladningsdeling og mellomliggende striper.
Ved å gjøre en ladnings interpolasjon mellom flere striper kan en beregne hvor partikkelbanen var med en støre nøyaktighet enn avstanden mellom utlesningsstripene. Ligning 3.1 viser hvordan en kan beregne tyngdepunktet for ladningen:

$$\eta \equiv \frac{PH_r}{PH_l + PH_r} \tag{3.1}$$

hvor PH_r, PH_l er signalet i høyre og venstre kanal respektivt. Figur 3.8 viser hvordan en kan finne treffpunktet x for partikkelen fra den venstre stripen som funksjon av η, her funnet empirisk fra en detektor kjørt i et testoppsett [38].

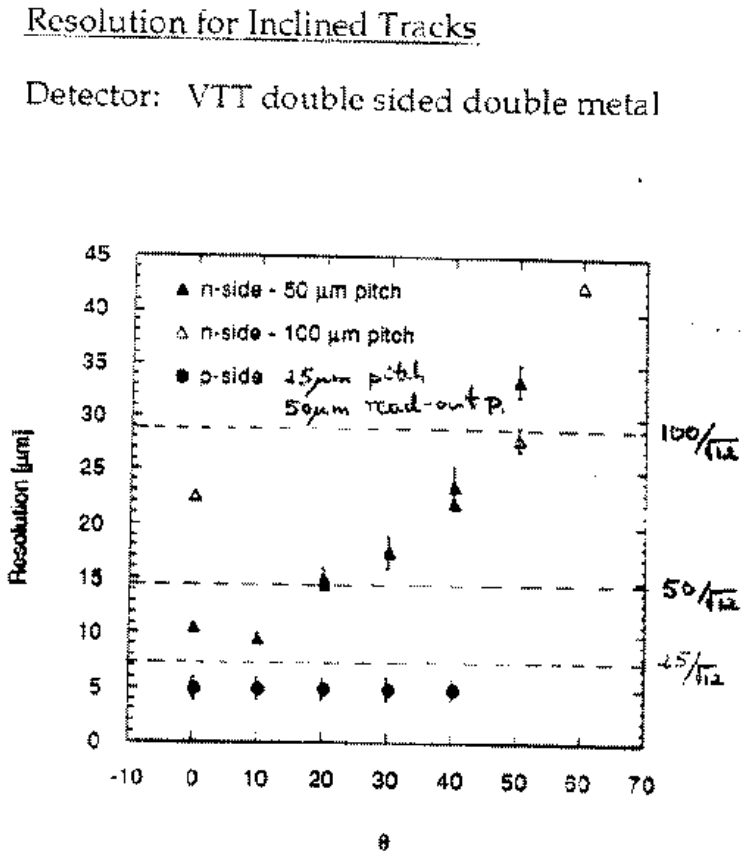


Figur 3.8: *Avstanden x fra venstre stripe (normert til 1) som funksjon av η ved analog utlesning og ladningsinterpolisasjon.*

Ved ladningsinterpolasjon er en avhengig av at partikkelen går tilnærmet vinkelrett igjennom detektoren. Dette fordi en ikke vil være i stand til å gjøre en nøyaktig ladningsinterpolasjon om ladningen gir et jevnt signal over flere striper. Figur 3.9 viser oppløsningen for et partikkelspor som funksjon av vinkelen til sporet for forskjellige testdetektorer. En kan se at reduksjonen av oppløsningen først gjør seg gjeldende når vinkelen blir så stor ($\theta > 10°$ for $50\mu m pitch$) at sporet i detektoren dekker flere striper. En må også være oppmerksom på at den veien en partikkel tilbakelegger innenfor en stripes bredde blir mindre ved at partikkelen går på skrå. Dette pga.

at utlesningsavstanden er mindre enn tykkelsen på detektoren, og en vil dermed få færre e^-/h par i hver enkelt stripes område.

En kan også se en effekt av dårlig oppløsning ved $\eta \approx 0$ pga. at ladningen hovedsakelig går til en stripe.



Figur 3.9: *Oppløsningen av en dobbeltsidig detektor som funksjon av vinkelen til de innkommende partikler. (på p-side er $\theta = 0$ og $25\mu m$ detektor pitch og $50\mu m$ utlesnings-pitch).*

En kan i tillegg ha mellomliggende striper, som figur 3.6 viser, som vil fungere på samme måte som de som leses ut, bortsett fra at de ikke er koblet til FE brikken. De ekstra mellomliggende stripene gjør at en får en økt kapasitiv kobling og mulighet for å gjøre en beregning av treffet til partikkelen uten å lese ut alle stripene.

En kan tenke seg at en får et δ formet strømsignal $Q\delta(t)$ på den midterste stripen pga. et partikkelspor her[2]. En vil da få påtrykt en spenning V_m på den midterste stripen. $Q\delta(t)$ signalet vil se en kapasitans $C_{is/2}$ påvenstre side og en kapasitans $C_{is/2}$ på høyre side. Når $dV = \frac{1}{C}dQ$ vil en pga. den kapasitive koblingen $C_{is/2}$ få et signal i nabostripene. Disse signalene i L og R vil da være like store og en kan da beregne at det var et spor i midten [9][43] [38].

Ulemper ved mellomliggende striper vil være at presisjonen til detektoren vil være meget avhengig av signal/støyforholdet og tap til den ohmske siden (C_{BP}). En annen ulempe er redusert

[2]Ser bort fra diffusjon til sidestripene.

to-spor oppløsning.

Figur 3.10 viser kvalitativt hvordan readout-pitchen innvirker på antall kanaler som leses ut[3]. Senter av ladningen er der hvor partikkelen gikk gjennom detektoren. Ladningsfordelingen er det området av detektoren det har blitt frigjort elektron hull-par. Signalklusteret er der en kan forvente å få signal i detektoren. En har her økt den kapasitive koblingen mellom stripene ettersom avstanden mellom stripene som leses ut økes. En kan av figuren se at ved spor som ikke går vinkelrett igjennom detektoren vil en ved liten utlesningsavstand ha en kraftig økning av antallet kanaler som må leses ut. Denne effekten av økt sannsynlighet for signal i en kanal vil være viktigst for striper vinkelrett på z ved store rapiditeter. En bør derfor, ved store Z, ha større avstand mellom stripene som leses ut. En må og ta hensyn til at antallet e/h par som blir løsrevet og kan drifte til hver enkelt stripe blir lavere når partikkelen går på skrå. Dette gjør at en får et svakere signal og evt. en reduksjon i sannsynligheten for at signalene er over grenseverdien til diskriminatoren.

Sannsynligheten for signal i en stripe som er funnet i tabell 2.5 vil derfor være avhengig av den avstand en velger å bruke mellom stripene som leses ut. Det vil være rimelig å tro at tabell 2.5 er riktig om en forutsetter at en justerer avstanden mellom de silisiumstripene som leses ut ettersom hvor og hvordan de ligger i detektoren. Verdiene i tabell 2.5 vil ellers trolig være noe lave og optimistiske.

For en 4" silisiumplate vil en typisk ha 1280 utlesningskanaler og 10 FE brikker eller 384 kanaler og 3 FE brikker, ved en utlesningsavstand på $50\mu m$ eller $150\mu m$ respektivt.

3.2.2 Et frontend elektronikk konsept for LHC.

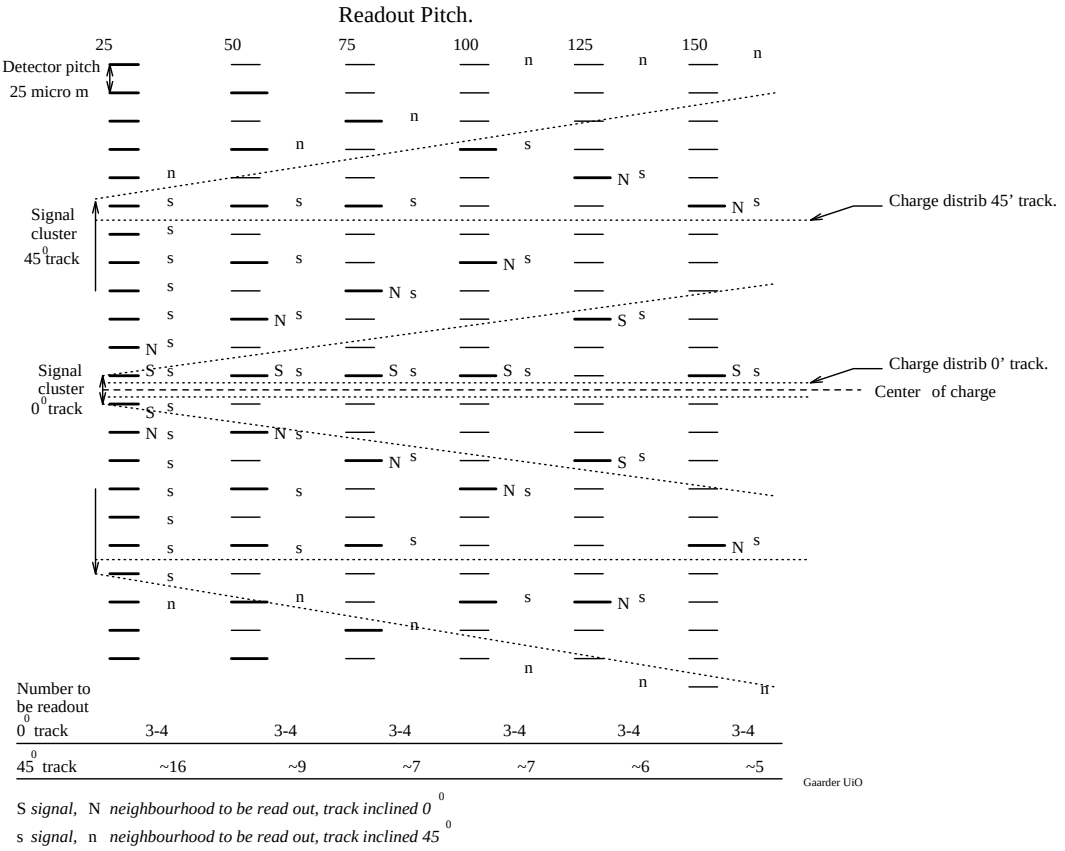
Den fremste elektronikken vil være meget viktig for konstruksjon av en silisium stripe detektor for LHC. Valget av løsninger vil være avhengige av de krav som er gitt for hele ATLAS detektoren. De viktigste og mest kritiske parametere er:

- Høyt signal/støy forhold.
- Lavt effektforbruk.
- Tilstrekkelig hastighet.
- Strålingsmotstandsdyktighet.
- Partikkelflux, datarate og trigger hastighet

For å kunne tilfredsstille disse kravene best mulig har RD-20 kommet frem til disse generelle løsningene:

- Enkel ladningsfølsom forsterker med enkel *shaping* for å kunne gjøre støyen minimal.
- Forsterker med lengre *peaking time* enn tiden mellom to hendelser, dette vil reduserer effektforbruk og støy.
- Analog pipeline med spenningsavlesning for å lagre signalene til det mottas en første nivå trigger.
- *Analog Pulse Shape Processor* (APSP).

[3]Det er her ikke tatt hensyn til at signalet kan bli for svakt til at det registreres av DAQ systemet om vinkelen blir for stor.



Figur 3.10: *Kvalitativt hvordan forskjellig avstand mellom de stripene som leses ut innvirker på antallet kanaler som har et signal. En ser her 6 forskjellige detektormoduler ovenfra, alle med forskjellig utlesningsavstand. Her med 2 partikkelspor henholdsvis* 0^0 *og* 45^0 *på normalen til detektoren. Den kapasitive koblingen blir justert opp ettersom utlesningsavstanden økes. Center of charge viser hvor partikkelen har gått (dvs. normalt på denne figur), charge distribution viser innenfor hvilke område det er avsatt e/h par og en kan forvente å få signal innenfor signal cluster.*

Figur 3.11: *Det originale RD-20 FEE konseptet.*

opererer synkront[4] og bruker samme klokkefrekvens som BC klokken, tilpasset den lokale tid for hver DM.

3.2.3 Forforsterker og shaper.

Forsterkeren og shaper er en ladnings følsom tidskontinuerlig forsterker fulgt av en CR-RC krets [23]. Den har en peaking tid og en forsterkning som er tilstrekkelig stor til å ungå store støybidrag

[4]Uten hand-shake fra prosesser etter APSP.

og ikke-uniform fordeling, men så lav som mulig for å redusere effektforbruk. Det er foreslått en forsterkning på 40mV/MIP. Med denne forsterkning og et støy bidrag på mindre enn $1500e^-$, vil en ha et effekt forbruk på ca. 1mW/kanal. Det er også foreslått en peaking tid på $t_p = 75$ns, dvs. over flere BC.

3.2.4 Analog Delay Buffer.

Analog Delay and Buffer, ADB[19][11] består av to deler, en analog del og en digital kontroll logikk. Den analoge delen består av 128 like, men individuelle parallelle kanaler. Hver kanal vil bestå av et antall analoge lagringsceller som både fungerer som pipeline og buffer. Antall lagringsceller N_{celler} er avhengig av antall avlesninger som APSP'en bruker N_{APSP}, lengden på pipeline N_{pip}, pga. trigger forsinkelsen $t_{T1delay}$, og buffer dybden N_{ADB}. En vil også i tillegg ha en ekstra celle av initialiseringsgrunner. N_{ADB} er avhengig av hvor stor andel av T1 en kan akseptere å miste pga. ADB bufferoverflyt. Om tiden mellom to hendelser er $\triangle$t vil lengden av ADB bli :

$$N_{celler} = N_{APSP} * N_{ADB} + t_{T1delay}/\triangle t + 1 \tag{3.2}$$

N_{APSP} er antall dataavlesninger som APSP bruker.

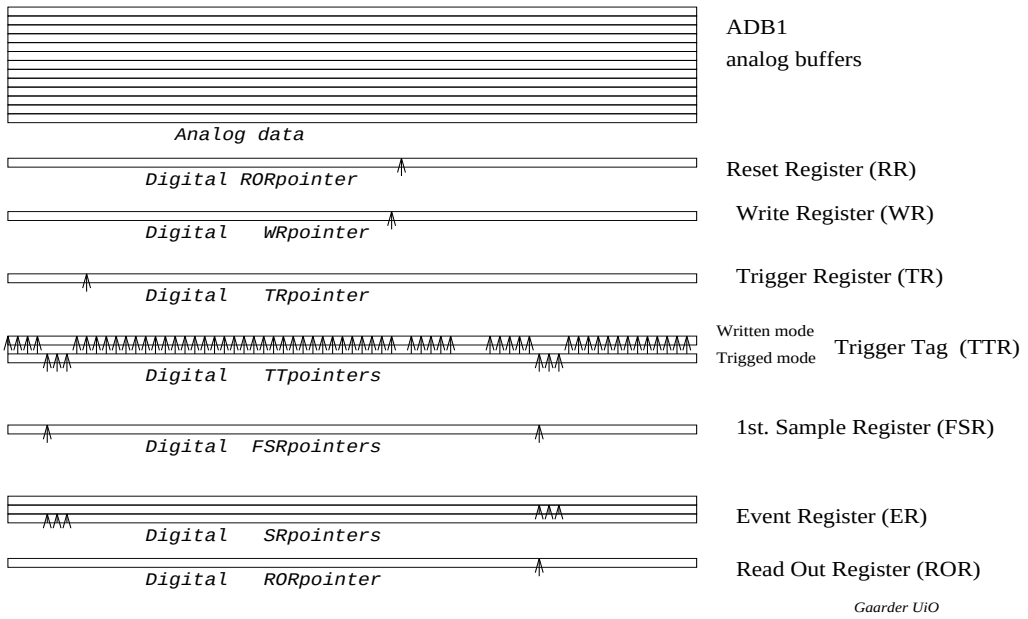
Registrene i den digitale kontroll logikken peker på hvor data fra silisium stripene blir lagt i ADB, hvilke celler som er opptatt av pipeline, hvilke celler som er lagret pga. at de er trigget, hvilke celler som tilhører en bestemt trigger og hvilke celle som blir eller skal leses ut. Klokka til denne digitale logikken blir styrt av BC klokken. Kontrollogikken er bygd opp av registre og en kontroll del. Hele logikken ligger parallelt på siden av lagringscellene, og det er en felles logikk som styrer alle kanalene i brikken synkront. Logikken er bygd opp av registre, eller pekere, med samme lengde som antall lagringsceller. Figur 3.12 viser registrene.

Reset Register (RR) ligger rett foran Write Register og nullstiller cellene slik at de er klare til å ta imot en ny verdi fra forsterkeren. RR flytter seg i takt med BC klokken og hopper kun til de celler som er ledige. RR vil se hvilke celler som er ledige ved å se på Trigger Tag Register og bruke en *by-pass* funksjon for å hoppe over de merkede. De celler som RR går til vil da være de som ikke er i Trigged mode av TTR. Når RR er kommet til enden vil den hoppe til den første celle som er ledig fra begynnelsen igjen, dvs. sirkulært. Det er og vurdert å la RR hoppe over f.eks annenhver celle og la den gå bakover igjen i de andre cellene når den er kommet til enden. Dette for å få en mere jevn overgang i endepunktene.

Write Register (WR) peker på den cellen der den nye verdien skal leses inn. WR vil gå i "fotsporene" til RR Δt bak. De celler som WR går til vil da være de som ikke er i Trigged mod. av Trigger Tag Register.

Trigger Register (TR) peker på hvilke celle som skal lagres om en får en T1. TR vil ligge i en avstand av nøyaktig T1's tidsforsinkelse etter WR og gå i "fotsporene" til WR. De celler som TR går til vil da være de som er i Written mode av Trigger Tag Register.

Trigger Tag Register (TTR) peker på de celler som har innlest data fra WR og de av disse som er merket pga. at det har vært triggere. Om det kommer en T1 vil TTR sette N_{APSP} celler i Written mode, der TR peker. TTR vil ikke sette Written mode før TR peker på cellen, dvs. det vil ta $N_{APSP} * \Delta t$ tid å merke av hele triggeren. En kan si at hver celle i TTR vil være i



Figur 3.12: *ADB logikkens registre med buffer for 3 hendelser, og en APSP som bruker 3 dataavlesninger ($N_{APSP} = SAPSP = 3$). Her med en hendelse lest ut og 2 hendelser klare til å leses ut.*

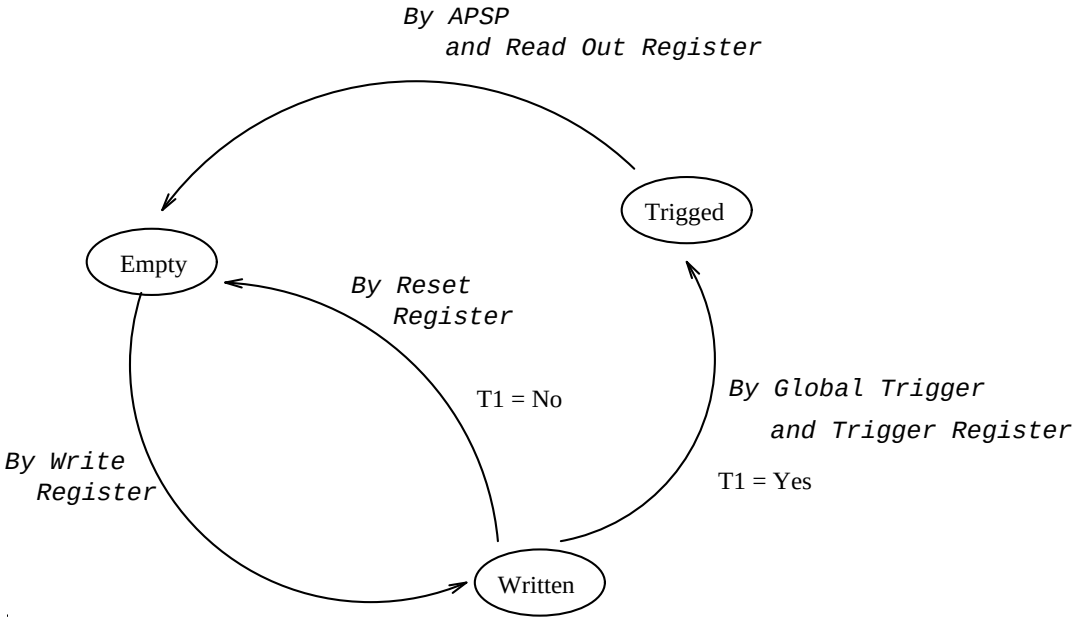
1 av 3 tilstander, empty, written eller trigged (figur 3.13). Er TTR empty betyr det at RR har nullstilt cellen eller at APSP har lest den ut, TTR vil da ikke ha noen pekere på disse cellene. Er TTR written betyr det at WR har lest inn en verdi. Er TTR trigged betyr det at vi har merket av cellen pga. en T1.

First Sample Register (FSR) peker på den første cellen av de som er lagret for en hendelse. FSR vil bli styrt av TR, og satt når en får en ny trigger. FSR gjør at en vet hvilke av de merkete cellene i en hendelse som er den første, resten av cellene som tilhører hendelsen vil da ligge syklisk etter denne.

Event Register (ER) er flere registre, ett for hver hendelse som skal lagres. ER peker på de celler som er lagret for en spesifikk hendelse. Antallet ER vil da være det samme som buffer dybden N_{ADB}av ADB. ER vil bli satt samtidig som TTR. ROR nullstiller ER. De forskjellige hendelser blir lest og nullstilt syklisk (FIFO).

Read Out Register (ROR) peker på den cellen som skal leses ut av APSP. ROR vil starte på den første cellen i den hendelse som har ligget lengst, og deretter vha. ER lese ut resten av hendelsen.

Datainnlesningen er uavhengig av utlesningen, og en vil derfor kunne kontinuerlig lese inn nye verdier i takt med BC klokken. For å redusere plass og effektforbruk er ADB cellene bygd opp av rene kondensatorer, som gjør at vi får en destruktiv utlesning, dvs. den analoge verdien kan bare brukes en gang. En vil der for ikke kunne lese ut mere enn en hendelse om det er flere som overlapper hverandre. Med overlapping menes det her at kriteriene for en ny trigger er oppfylt



Figur 3.13: *Tilstandene til Trigger Tag Registeret, og når TTR skifter tilstand.*

før S_{APSP} hendelser. Triggerlogikken vil da globalt undertrykke denne triggeravgjørelsen.

3.2.5 Analog Pulse Shape Processor.

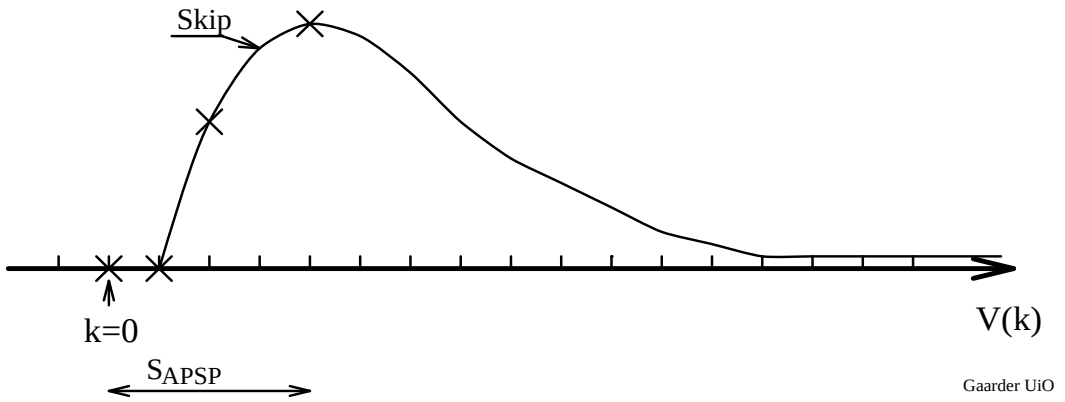
Signalet ut av forsterkeren bruker en tid som går ut over BC tiden før signalet har nådd sitt maksimum, $T_p = 75ns$ for RD-20's løsninger. Signalet har også en hale som også gjør at en kan få overlapping av flere signaler. Disse effektene gjør at en har valgt å ta flere avlesninger av signalet, en tids-diskret integrasjon og vektlegging for å best mulig kunne gjengi det opprinnelige signalet. En vil derfor etter forsterkeren (og ADB) bruke et filter, *Analog Pulse Shape Processor*(APSP)[39], for å kunne dekode avlesningene og beregne en spesifikk verdi av disse, dvs. et diskret analogt signal som på best mulig måte representerer partikkeltreffet i tilhørende stripe. APSP'en vil også ha en effekt av å redusere den parallelle støyen. Figur 3.14 viser de avleste signalene ut fra forsterkeren. kryssene viser de som blir benyttet av APSP'en.

Det vil være en APSP for hver kanal, dette pga. at APSP prosessen tar forholdsvis lang tid ($\sim 5\mu s$) og pga. at verdiene ut av ADB (foran diskriminatoren) er for upresis til å foreta en undertrykking av kanalene uten treff. Dette gjør at en ser bort fra muligheten til å multiplexe flere kanaler til en APSP.

APSP vil lese av og nullstille N_{APSP} celler fra ADB'en. Den destruktive lesingen av cellene fører til at f.eks om to T1 kommer rett etter hverandre, vil APSP når den leser den første T1 også slette celler som hører den andre til, vi får en T1 dødtid,

$$T1_{sample} = (S_{APSP} - 1) * \triangle t. \tag{3.3}$$

der S_{APSP} er antallet celler i ADB fra og med den første til den siste som er merket for en T1 inkludert de som APSP evt. ikke leser av.



Figur 3.14: *De avleste signalene fra forsterkeren, med avlest intervall som abscisse.*

APSP starter på den første cellen som hører til en hendelse og bruker ca. T_{APSP}/N_{APSP} på å lese og prosessere denne verdien, og vil så syklisk gjøre dette for hver celle som hører til hendelsen. APSP vil ikke frigi bufferne føre den er ferdig med å lese ut alle N_{APSP} cellene fra hendelsen.

3.2.6 Diskriminator

Det er bare en liten del av kanalene som vil ha et treff for hver hendelse. Det vil likevel kontinuerlig komme svake signaler ut av forsterkerne pga. støy selv om det ikke er partikkeltreff i kanalen. Dette argumenterer for å ha en diskriminator som på best mulig måte skiller signaler pga. partikkeltreff i stripene fra signaler pga. støy. Denne diskriminatoren kan da f.eks. etter å ha sammenlignet med en gitt grenseverdi, avgjøre om vi skal ta vare på verdien i kanalen eller forkaste den. En vil da etter diskriminatoren ha muligheten til å foreta en kraftig datareduksjon.

En kan si at dataraten inn til diskriminatoren er gitt av T1 frekvensen, og vil derfor være synkron for hele detektoren. Datamengden ut av brikken etter diskriminatoren vil være avhengig av antall treff i de silisium striper som tilhører FE brikken og er derfor asynkron. Diskriminatoren vil sette en *Threshold-bit* i et en-bit digitalt buffer som går parallelt med det analoge bufferet avhengig av om signalet er større/mindre enn diskriminator-nivået. Flere løsninger for diskriminatoren vil være mulig, her er noen eksempler:

1. Diskriminatoren sammenligner verdien fra APSP med en referanseverdi og setter den binære biten om verdien er over. Diskriminatoren gir også signal til kanalene ved siden at disse også skal sette den binære biten. Dette systemet er vist i figur 3.24 og vil være den løsningen som denne oppgaven baserer seg på.

2. En kan vurdere å samle opp ladningen fra side kanaler og gjøre en beregning av total ladning "Interstrip Intelligence" (II) [30]. Dette gjøres for at en skal kunne registrere treff som har ladning over den gitte grenseverdi, men har fordelt ladningen over flere striper. Dette vil trolig gjøre at en kan heve grenseverdien for diskriminatoren noe, og en vil få et noe bedre signal/støy forhold. Sannsynligheten for utlest signal i en kanal pga. støy vil bli redusert samtidig som sannsynligheten for å lese ut reelle treff av partikler trolig vil øke noe.

3. Ved et signal i en kanal kan en også forvente at det er et svakt signal i nabokanalene. En kan derfor undertrykke et svakt signal om det ikke er signal i nabokanalene. Dette vil gi noenlunde de samme effektene som alternativet rett ovenfor gir.

4. Det kan i visse tilfeller være mulig å gjøre en ladningsinterpolasjon med kun to signaler utlest. En kan derfor ha en mere komplisert logikk som justerer og forbereder signalene for en ladningsinterpolasjon. Dette vil da bli gjort etter en sammenligning av signalene fra kanalene som ligger ved siden av hverandre. Dette kan f.eks. gjøres med et nevralt nettverk. Med et slik system kan en etter utlesning fra DM beregne senter for treffpunktet vha. f.eks. kun to kanaler istedenfor 5-6 ved treff av en partikkel som har gått på skrå igjennom detektoren.

   En vil også ha de muligheter som er nevnt i alt. 2 og 3 ovenfor.

   Den vesentlige fordelen med et slik system vil være at det muligjør en ytterligere reduksjon av mengden data som skal leses ut av DAQ systemet.

Uansett hvilke system en velger så vil alle kreve en kalibreringsrutine og trolig et eget referansesignal. Denne kalibreringsrutinen kan enten bli gjort lokalt inne i FE brikken, av MC brikken eller globalt.

3.2.7 Sparse utlesning.

Etter diskriminatoren har vi fått en data reduksjon, noe som argumenterer for at en kan redusere antall kanaler, dvs. bruke en multiplekser og la flere kanaler gå sammen til en som leses serielt ut. Det er foreslått å la alle kanaler på en DM bli multiplekset inn til en ADC. Evt. 2 ADC, en for X retning og en for Y retning. I front av denne multiplekseren vil vi trenge et buffer pga. at all data fra en hendelse kommer parallelt og må mellomlagres til multiplekseren er ferdig. Etter multiplekseren, og foran ADC, vil det ikke være et buffer. Dette gjør at den maksimale utlesningshastigheten til multiplekseren vil være gitt av hastigheten til ADC, eller vv. I denne oppgaven er dette system kalt *Derandomising and Sparse Readout* (DSR) [45] og er under utvikling for testkjøring i løpet av 1995.

Et løsningsforslag til systemet er som følger:
Innebygd i hver FE brikke er et system som er bygd opp av parallelle buffere og pekere som kontrollerer inn og utlesningen. I tillegg har en et felles kontrollsystem for hele DM, kalt *Bus Snooker* (BS), som vil være sentralt plassert på DM. BS vil starte/stoppe utlesningen av en spesifikk brikke og samtidig holde orden på adresser. Figur 3.15 viser hvordan dette er tenkt gjort.

Inne i hver brikke vil det være pekere som styrer hvilke hendelse som skal eller blir lest ut. Utpekerene peker på den hendelsen som blir lest ut. Innpekeren peker på den siste hendelse som kom inn. Logikkpekeren, inne i hver brikke velger den kanal som skal leses ut, med start på kanal 1. Analog-pekeren vil bruke en t_{DSR} tid på finne og lese ut en kanal med data. Når både analog-pekeren og utpekeren peker på en celle i bufferet samtidig som BS gir klarsignal for denne brikken vil dennes analoge verdi bli lagt ut på en analog utlinje og threshold-bit blir resatt. Utlesningslogikken bruker threshold-bit'en som referanse for å finne bufferne som skal leses ut. Logikken fortsetter til alle buffere med data og hendelser er lest ut av brikkene. Utvelgelsen og styringen av hvilken brikke som leses ut krever et system som er både raskt og pålitelig, slik at ikke en feil i en kanal eller brikke gjør at hele DM blir ute av drift. BS må også holde kontroll

Figur 3.15: *Bus Snooker (BS) systemet til DM.*

på når en hel hendelse er lest ut av FE brikkene. Dette for at PE så hurtig som mulig skal få beskjed om at den kan avslutte pakken som tilhører hendelsen.

Dette systemet vil trolig trenge følgende signal linjer:

- Klokkelinje.
- En adresselinje som FE brikken kan sende ut kanaladressen. Denne kanal adressen vil trolig bli sendt serielt.
- En analoglinje der den FE brikken kan sende ut den analoge verdien fra kanalen til ADC'en.
- En linje som forteller om det ligger data til å leses ut av FE brikken.
- Linjer som forteller at en spesifikk FE brikke kan begynne å lese ut. På figur 3.24 er dette gjort ved at en har en teller innvendig i FE brikken som holder rede på hvilken FE brikke som blir lest ut. Denne telleren blir så styrt vha. to linjer, en som øker telleren og en som nullstiller den. Istedenfor linjen som nullstiller telleren kan en evt. la telleren kun løpe til antallet FE brikker og så starte på null igjen. En kan da la nullstillingslinjen være en mere generell linje for nullstilling av hele FE brikken.

I tillegg til disse linjene er det også trolig at en vil kunne trenge andre kontrollsignaler.

Tiden en vil bruke på å lese ut en kanal vil være noe avhengig av om det er den første kanalen i en hendelse, om det er den første kanalen i en FE brikke eller en vilkårlig kanal som skal leses ut. Ved start på utlesning av en ny hendelse må en gi analog-pekerene tid til å søke igjennom alle kanalene og tid til å sette linjene som forteller om det er data i brikkene eller ikke. En vil og trenge noe tid på skifte utlesning fra en FE brikke til en annen. Disse operasjonene vil trolig føre til at den gjennomsnittlige tiden det tar å lese ut en kanal vil variere noe fra t_{DSR}. En kan finne at ca. 40 kanaler blir lest ut for hver T1 fra en DM ved 0.6% sannsynlighet for signal, 2 nabokanaler og 18 FE brikker med 128 kanaler. Om en regner at den første kanalen bruker dobbelt så lang tid som de andre vil dette føre til en gjennomsnittlig økning av utlesningstiden på ca. 2%. I denne oppgaven er det derfor regnet med at den gjennomsnittlige tiden en vil trenge for å lese ut en kanal fra FE brikkene vil våre t_{DSR} uavhengig av hvilken kanal dette er.

3.2.8 Analog til Digital Konvertering.

Det analoge signalet vil bli konvertert til en digital representasjon. Hvordan og hvor i utlesningssystemet dette skal gjøres vil være en vurdering avhengig av datamengde/frekvens, støy forhold, dynamisk bredde, kalibrering, strålingshardhet, plass, varmeutvikling, hvor datareduksjon kan gjøres, robusthet, kompleksitet, kostnader etc. En digital representasjon av verdien vil gi en sikrere overføring av data, noe som gjør at det vil være en fordel å gjøre konvertering så tidlig som mulig i systemet. I utlegget av DM som blir behandlet her vil en vurdere mellom 3 generelle løsninger:

- En ADC for hver brikke etter DSR og multipleksing til et felles digitalt buffer. Dette gir muligheten til å koble hver FE brikke direkte til en ADC, evt. integrere en ADC i hver FE brikke. Dette vil føre til kort overførsel av de analoge verdiene. En fordel vil være at om en ADC slutter å virke vil det ikke sette hele DM ut av funksjon. Ved å legge ADC'ene sammen med resten av FE-elektronikken vil det også med visse kjølesystemer være enklere å kjøle den.

- En ADC for hver side av DM. Ved å multiplekse flere FE brikker inn til en ADC vil en trolig benytte kapasiteten til ADC'en bedre og en vil få en mere jevn drift av ADC'en. Denne løsningen er også bedre egnet for bruk av kommersielt tilgjengelige ADC'er.
- En ADC for hele DM. Denne løsningen fører til at en må føre signalene fra to sider inn til et punkt.

Konverteringen kan være logaritmisk. Dette reduserer antall bit samtidig som det gir en bedre oppløsning av de laveste verdiene.

3.2.9 Modulekontroller.

For å styre og kontrollere utlesningen av DM må en ha en enhet som er kalt *Module Controller* (MC). Figur 3.16 viser en oversikt over hva MC vil bestå av.

Modulekontrolleren vil inneholde en kontroll og monitorings enhet, protokoll-maskin, bus-snooker, tilkobling til utlesningsnettverket og evt. en ADC. Alle disse komponentene vil i mere eller mindre grad bli integrert i en brikke. Figur 3.17 og figur 3.18 viser i mere detalj hvordan kommunikasjon innad på MC vil foregå.

På MC bør det være et system som går synkront med FE brikkene. En kan grovt si at dette vil være en kopi av en FE-brikke og dennes logikk. Med dette systemet kan MC hele tiden skal holde rede på tilstanden til FE brikkene uten å ha direkte kommunikasjon med dem. En vil på denne måten ha et raskt kontrollsystem med få linjer. I øvre høyre hjørne av figur 3.17 kan en se et eksempel på hvordan et slikt system er tenkt. Dette systemet vil også kunne gi feilmeldinger til protokoll-maskinen ved f.eks. tap av data.

3.2.10 Protokoll-maskin.

Inne i MC vil det være en *Protocol Engine* (PE) [33]. PE vil ta seg av den generelle datapakkingen, bufferingen og hånteringen inn og ut av DM. Den PE som blir beskrevet her er rettet spesielt mot et datahåndteringsystem kalt SWIPP [32]. Figur 3.19 viser hvordan en har tenkt seg PE. PE vil gjøre det mulig å koble DM som en generell node til SWIPP systemet vha. en seriell dataforbindelse. PE vil organisere data som serielle pakker av data som vil bestå av adresse, data, kontrollinformasjon osv. Normalt vil all data fra en hendelse bli organisert som en slik pakke.

3.2.11 Kontroll og testing av DM under drift.

En må ha mulighet for å kunne gå inn og utføre visse rutiner og avlesninger av DM samtidig som eksperimentet er i drift. Dette kapittelet vil beskrive noen av mulighetene for hvordan dette er tenkt å kunne bli integrert inn i DM.

Fordelen med å kunne utføre en kontroll direkte på DM vil være at en kan redusere antallet kabler og referanselinjer koblet til DM. En kan da ha en DM som kun har noen få elektriske tilkoblinger i tillegg til den optiske koblingen og som i størst mulig grad *henger fritt* i forhold til omgivelsene.

Det er noe uklart hvilke rutiner som er nødvendige for DM. Generelt bør en forsøke å lage et system som er mest mulig fleksibelt slik at en kan legge til ekstra rutiner ettersom en finner det nødvendig. En bør da lage et standardisert system med en standard intern databuss koblet til PE inne i MC. Det må også være en form for prioriterings logikk som styrer denne databussens

Figur 3.16: *Utlegg av Modul-kontrolleren og dennes logiske inndeling.*

Figur 3.17: *Utkast av modul-kontrolleren med kommunikasjon mellom FE-brikken og Protokoll-maskinen.*

Figur 3.18: *Utkast av modul-kontrolleren og dennes kommunikasjonslinjer.*

Figur 3.19: *Utlegg av protokoll-maskinen.*

kommunikasjon med PE. Dette gjør at forskjellige rutiner kan ha forskjellig prioritering, dette for å forhindre at de primære oppgave til DM ikke må vente på at mindre viktige, og ofte sene, rutiner blir utført. Rutinene må også i størst mulig grad være slik at de kan bli utført parallelt. Figur 3.20 viser hvordan en kan tenke seg at et slik system bør se ut, mens figur 3.18 viser dette systemet integrert i MC.

Følgende rutiner kan tenkes å være mulig å utføre på hver DM:

- Kontroll og justering av forspenning og strøm. Dette vil f.eks være forspenningene som ligger over silisiumdetektorene. Om denne spenningen er feil vil verdiene ut få en målefeil. Ved å måle strømforbruket til DM vil en også kunne ha en viss kontroll over tilstanden til DM. En kan også tenke seg å justere spenninger ettersom DM forandrer karakter under lengre drift.

- Monitorering av sensor-lekasjestrømmer.

- Kontroll av grenseverdier.

- Kalibrering av FE. En må ha mulighet for å kalibrere, internt eller eksternt, grenseverdier og referansespenningene for hver enkelt DM.

- Statistikk over signalverdier for en spesifikk kanal, brikke eller modul. Dette kan gi en indikasjon på den enkelte stripes, brikkes eller moduls tilstand og evt. gi grunnlag for kalibreringer. Dette kan tenkes å gjøres ved å inne i MC summere opp verdien ut fra ADC'en hver gang en spesifikk kanal eller brikke leses ut. Et register på 32 bit vil være tilstrekkelig til å kunne summere opp verdiene fra ADC'en for alle treff i DM for hver T1 over et helt døgns drift av eksperimentet. Denne statistikken kan også gjøres globalt. En global statistikk vil føre til en enklere MC, men noe mindre fullstendig monitoring og kontroll av DM.

- Egen sensor som måler den radioaktive dosen DM har fått.

- Klokkefasejusteringer. Hver DM vil ha sin egen lokal tid, det må derfor være mulig å finjustere de interne klokkepulsene på DM.

- Temperaturmålinger. Ved å måle temperaturen på DM og evt. enkeltkomponenter innad på DM vil en kunne ha kontroll på tilstand til DM samtidig som en kan korrigere signalprosessering som er temperaturavhengig.

- Sjekk av interne trigger og hendelsesnummer mot globale verdier.

- Påtrykke signaler. En kan f.eks tenke seg å påtrykke ADC'en en gitt spenning for så å kunne sjekke denne etter utlesning eller for kalibrering.

- Nullstilling av klokker og tellere.

- Slå av og sperre deler av DM. En kan f.eks tenke seg å koble ut alle FE brikkene i en retning på DM.

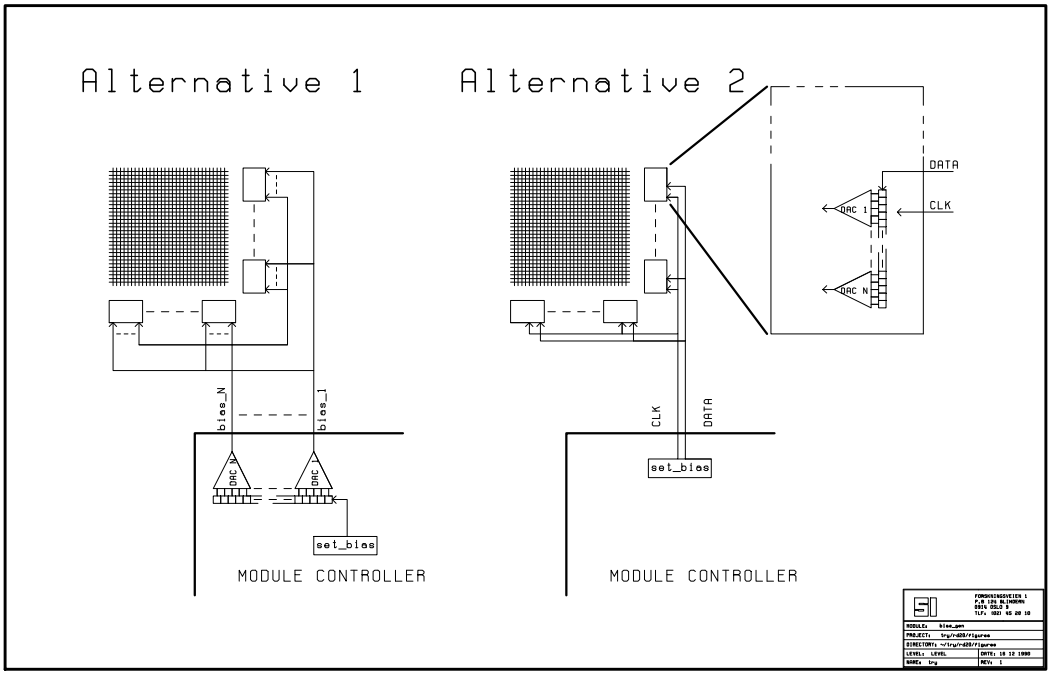
- Nullstilling og start av hele DM. DM må ha en oppstart rutine slik at DM i størst mulig grad kan resette og sette seg selv i drift med minimal kommunikasjon mot den eksterne monitoring utenfor DM.

Figur 3.20: *Utlegg for kontroll og monitor delens bussarkitektur.*

Noen av disse rutinene vil være noe overflødige eller konstruksjonsmessig vanskelige å inkludere og vil derfor etterhvert bli vurdert om de er nødvendige.

De analoge delene av FE brikken og ADC'en vil trenge 8-12 referanse/forspenning spenninger/strømmer. Trolig vil en kunne bruke de sammereferansene for alle FE brikkene over hele DM [42]. To alternativer er vurdert for generering av referansespenninger:

1. Nødvendige referanser kan bli generert i MC og fordelt parallelt til FE brikkene.
2. Referansene kan bli generert lokalt inne i hver FE-brikke.



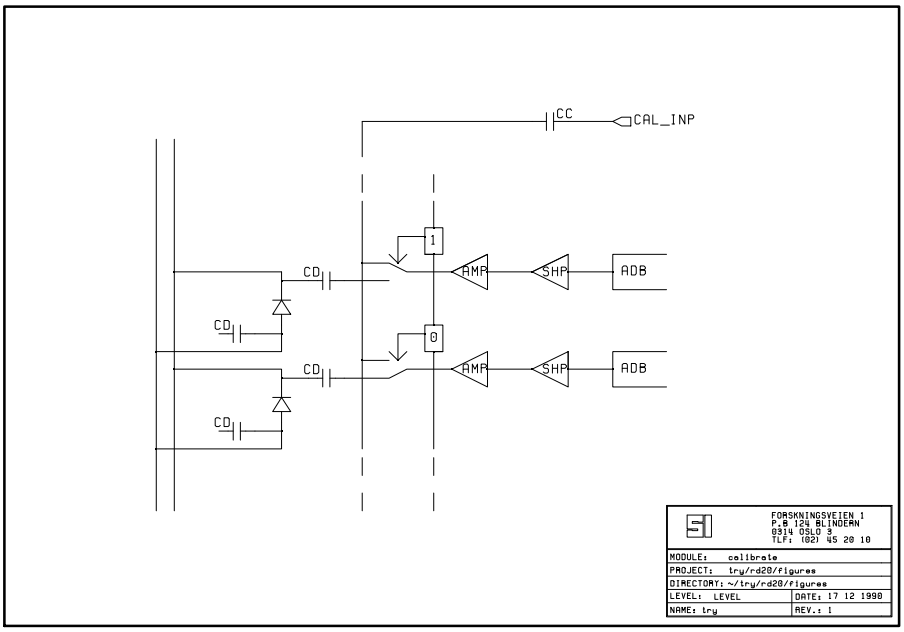
Figur 3.21: *Generering av forspenninger spenninger og strøm innad på DM.*

Figure 3.21 viser prinsippet for disse to løsningene. Pga. plassmangel innad i FE brikken vil trolig alt. 1 være best egnet, selv om denne løsningen stiller større krav til signal overføring innad på DM. Alternativ 2 vil imidlertid være en mere robust løsning.

FE kalibrering

FE brikken vil bli kalibrert ved at det settes en en-bit. Alt ettersom denne bitten er satt vil forsterkeren bli styrt til å få signal fra silisiumstripen eller fra et testsignal CAL_{INP}.

Figure 3.22 viser prinsippet for kalibreringen, her med en bit satt for en kanal.



Figur 3.22: *Skisse for hvordan et testsignal CAL_{INP} kan på trykkes en kanal.*

En må vurdere om en skal ha mulighet for å individuelt sette en og en kanal i kalibreringstilstand eller hele FE brikken[5] samtidig. Det ideelle vil være å kunne påtrykke et spesifikt CAL_{INP} for hver kanal. Dette vil da gi mulighet for å kunne påtrykke detektoren en allerede simulert hendelse.

En kan også slå av spesifikke deler av FE elektronikken ved å sette elektronikken i kalibreringstilstand uten å påtrykke noe signal.

Justering av BC klokken.

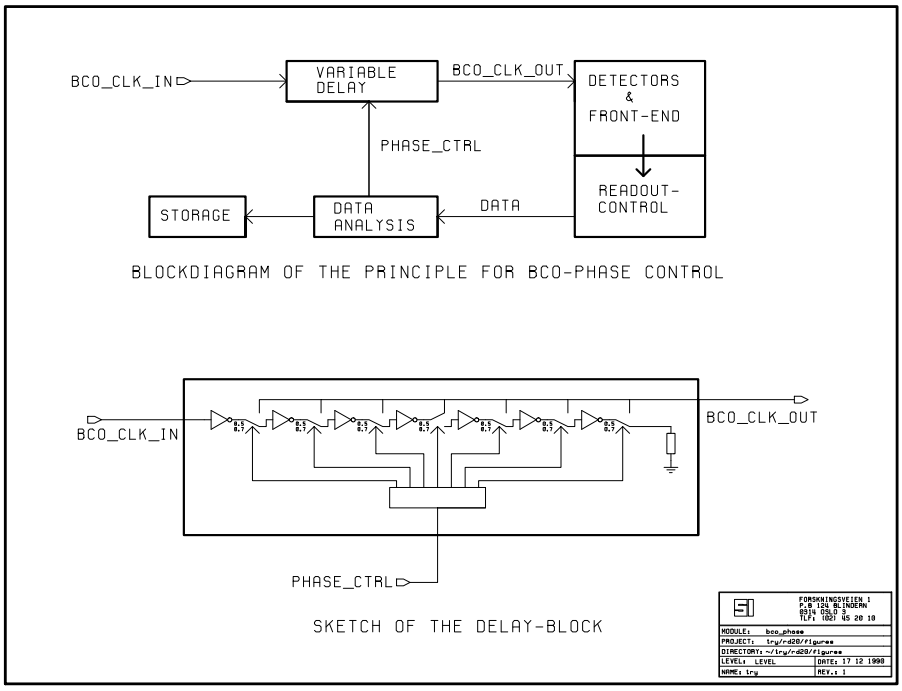
Som beskrevet i kapittel 2.4.6 vil en ha behov for å justere de lokale klokke-signalene, i forhold til den globale BC klokken. Figur 3.23 viser hvordan dette kan bli gjort. Hver inverterer vil her gi en viss tidsforsinkelse av BC klokkepulsen. En kan variere antallet invertere som legges i serie med BC klokkesignalet og samtidig gjøre en statistikk over signalet. Denne statistikken vil vise det ideelle avlesningspunktet for forsterkeren.

Måling av temperatur.

Det vil være viktig å holde detektoren ved konstant temperatur. Det vil derfor trolig være nødvendig å legge inn temperatursensorer inne i detektoren for å kunne kontrollere temperaturen under drift. De lokale temperaturene vil også kunne gi indikasjoner på tilstanden til elektronikken. En kan enten montere inn standard temperatursensorer eller integrere dioder/transistorer direkte i VLSI'en.

Stråling og parameter drift.

[5]Evt. hele DM.



Figur 3.23: *Mulig metode for å justere BC klokken internt på DM.*

Selv om en lager silisiumdetektorene og FE elektronikken strålingshard vil det etter en tids drift bli variasjoner i detektoren pga. stråling. Det er ikke klart om det vil være nødvendig å måle den oppsamlete dosen for DM. Men ved å måle dosen som elektronikken er blitt utsatt for vil en ha et bedre grunnlag for å kunne justere og kalibrere parametrene til detektoren.

3.3 Støy.

For en $300\mu m$ silikondetektor er signal til støyforholdet (SNR) definert som [44]:

$$SNR = MIP/ENC = 22400e^-/ENC \tag{3.4}$$

hvor ekvivalent støy ladning (ENC) [rms e^-] uttrykker støyen i systemet. Støyen vil bestå av seriell støy hvor:

$$ENC_s \propto C_{in} \quad og \quad 1/\sqrt{T_p} \tag{3.5}$$

$C_{in} = C_{det} + C_{amp}$ er kapasitansen over inngangen og T_p er stigetiden for forsterkeren. Denne støyen vil hovedsakelig komme pga. termisk støy i stripene. En vil og ha parallell støy hvor

$$ENC_p \propto \sqrt{T_p} \tag{3.6}$$

hovedsakelig fra lekkasjestrøm (I_d) (*shot noise*).

Om en krever en 4σ grense på støyen i forhold til diskriminatorgrensen på $4750e^-$, finner en at 1σ støy tilsvarer $1200e^-$ [36]. Dette vil videre gi en sannsynlighet for signal i en kanal

pga. støy på 0.003%. En mindre optimistisk og realistisk støygrense vil være 3σ. Dette vil gi en sannsynlighet for signal pga. støy tilsvarende 0.15%.

I denne oppgaven forutsettes det at dette bidraget av støy er jevnt fordelt og kan adderes direkte til verdiene i tabell 2.5. En må allikevel forvente at en vil ha variasjoner i støy lokalt pga. f.eks. produksjonsvariasjoner av elektronikken og variasjoner i støy over tid pga. f.eks. ujevne referansespenninger, bestråling av elektronikk osv. Det kreves videre prototype studier for å finne disse variasjonene.

En har her regnet med flere forskjellige støyforhold, og får en sannsynlighet for at diskriminatoren definerer kanalen som truffet, som tabell 3.1 viser for $r = 200mm$.

Fysikk bidrag	Støy	Totalt
0.45	0.15 %	0.60%
0.45	0.40 %	0.85%
0.45	0.80 %	1.25%
0.45	1.3 %	1.75%
0.45	2.0 %	2.45%

Tabell 3.1: *Antatt sannsynlighet for at det skal være et signal i en kanal over grensen til diskriminatoren.*

3.4 Datahåndtering og dataflyt innad på detektormodulen.

Som nevnt tidligere vil en av forskjellige årsaker ikke klare å handtere interessante hendelser eller en vil miste hele hendelser eller deler av denne i DAQ systemet. En må derfor vurdere hvordan de forskjellige prosesser skal kommunisere, når data skal overføres, når en skal akseptere å miste data osv.

Det er viktig at det ikke er noen korrelasjon mellom når en mister data og denne hendelsens datamengde eller fysikk-kanal den kommer fra. Dette for å hindre at sannsynligheten for tap av data skal være større for interessante hendelser enn for bakgrunn. I denne oppgaven er det lagt vekt på at det ikke skal være noen korrelasjon mellom datamengde, konsentrasjonen av datamengden i detektoren og tap av data [6].

Hovedprinsippet må da bli at en prioriterer de hendelser som allerede er i systemet. **Dette vil si at en heller forkaster nye hendelser enn deler av en hendelse som allerede er i systemet**. En må også lage bufferne og systemet slik at en ikke mister data fra en hendelse pga. at den inneholder mye data.

Figur 3.24 viser en mulig løsning av hvordan Felix FE brikken kan bli utlest. Signalet ut av forsterkeren vil kontinuerlig bli avlest for hver Δt og lagt inn i ADB. Forsterkeren og pipeline funksjonen til ADB er laget slik at de kontinuerlig skal kunne håndtere de nye verdiene uten tap av data. Dette pipelinesystemet er derfor optimalt og vil ikke kunne bli forbedret mhp. tap av data.

APSP'en vil starte med å prosessere dataene fra en hendelse når den får signal fra ADB-kontrollen. ADB-kontrollen vil være en del av den digitale logikken for ADB som beskrevet i

[6] Dvs. de forskjellige fysikk-kanalene er ikke vurdert.

Figur 3.24: *Forslag for utlesning av Felix brikken med tilhørende signaler.*

kapittel 3.2.4. ADB-kontrollen vil kontrollere, hvilke hendelser som er lagret, overføringen av data fra ADB til APSP og evt. når data fra APSP skal legges ut til DSR.

ADB kontrollen vil igjen bli styrt av T1, BC klokken og evt. andre kontrollsignaler internt i brikken eller på DM.

Pga. at APSP benytter seg av flere avlesninger fra ADB må en ha $S_{APSP} - 1$ BC mellom to T1, der S_{APSP} er som gitt i formel 3.3. En antar her at GLVL1 holder kontroll på denne dødtiden og ikke sender ut en ny T1 før det har gått $S_{APSP} - 1$ BC.

En har flere alternativer for når APSP skal begynne å prosessere verdiene fra ADB.

1. APSP begynner å prosessere data fra ADB så fort ADB-kontrollen har merket av data for en ny T1, eller om det er tidligere avmerkede T1 i ADB.

   Systemet vil da være synkront til etter APSP, og GLVL1 kan holde rede på om det er plass i ADB. Bufferoverflyten i ADB blir dermed håndtert globalt. Det vil være plass i ADB når $N_{ADB} > S_{ADB}$ der N_{ADB} er dybden til ADB og S_{ADB}er antallet hendelser som er i ADB.

   ADB-kontrollen vil overføre data fra APSP om det er ledig plass i DSR bufferene. Det vil være ledig plass i DSR bufferet når $N_{DSR} > S_{DSR}$ der N_{DSR} er dybden til DSR bufferet og S_{ADB} er antallet hendelser som er i DSR bufferet. Antallet hendelser som vil ligge i DSR bufferet vil være avhengig av antallet kanaler som skal leses ut for hver hendelse som er i bufferet[7] og tiden utlesningslogikken bruker på lese ut en kanal. Inn pekerene og utpekerene for DSR bufferet vil bli flyttet samtidig for hele DM. Dette gjør at en vil ha synkronitet innad på DM, men global asynkronitet mellom DM'ene etter diskriminatoren[8].

   MC vil parallelt holde rede på S_{DSR} og rapportere overflyt i DSR bufferet når dette oppstår.

   Det forventes at ADC'en etter DSR har tilstrekkelig hastighet til å prosessere data kontinuerlig uten buffere i front.

2. En kan la DSR-kontrollen rapportere fulle DSR buffere tilbake til ADB-kontrollen, ADB-kontrollen kan så vente med å starte APSP til det er ledig plass i DSR bufferet. Dette fører til at en flytter overflyt i systemet fremover fra DSR til ADB. Det vil også her være mulig for MC å holde rede på tilstanden til FE brikkene.

   Fordelen med dette systemet vil være at en trolig reduserer tapet av data og APSP slipper å prosessere en hendelse som likevel blir forkastet pga. DSR buffer overflyt.

   Ulempen vil være at DM modulene ikke lenger vil være synkrone etter pipeline.

3. En kan ha samme løsning som alt. 2, men i tillegg også undertrykke APSP fra prosesser etter DSR. En kan tenke seg at en prosess, når den ikke har kapasitet til å håndtere mere data, gir beskjed til prosessen foran at denne skal stoppe. Hele systemet vil da fungere som en kjede av prosesser der overbelastning blir ført tilbake til APSP og dermed ADB. En vil da kun miste data i ADB bufferet. Det er vanskelig å generelt si noe om et slik system pga. at det vil være avhengig av prosessene etter DSR. Fordelen vil være at en får en jevnere strøm av data i systemet og dermed kan dimensjonere resten av DAQ systemet mindre. En vil også være sikker på at det ikke er noen korrelasjon mellom datamengden i en hendelse og tap av data.

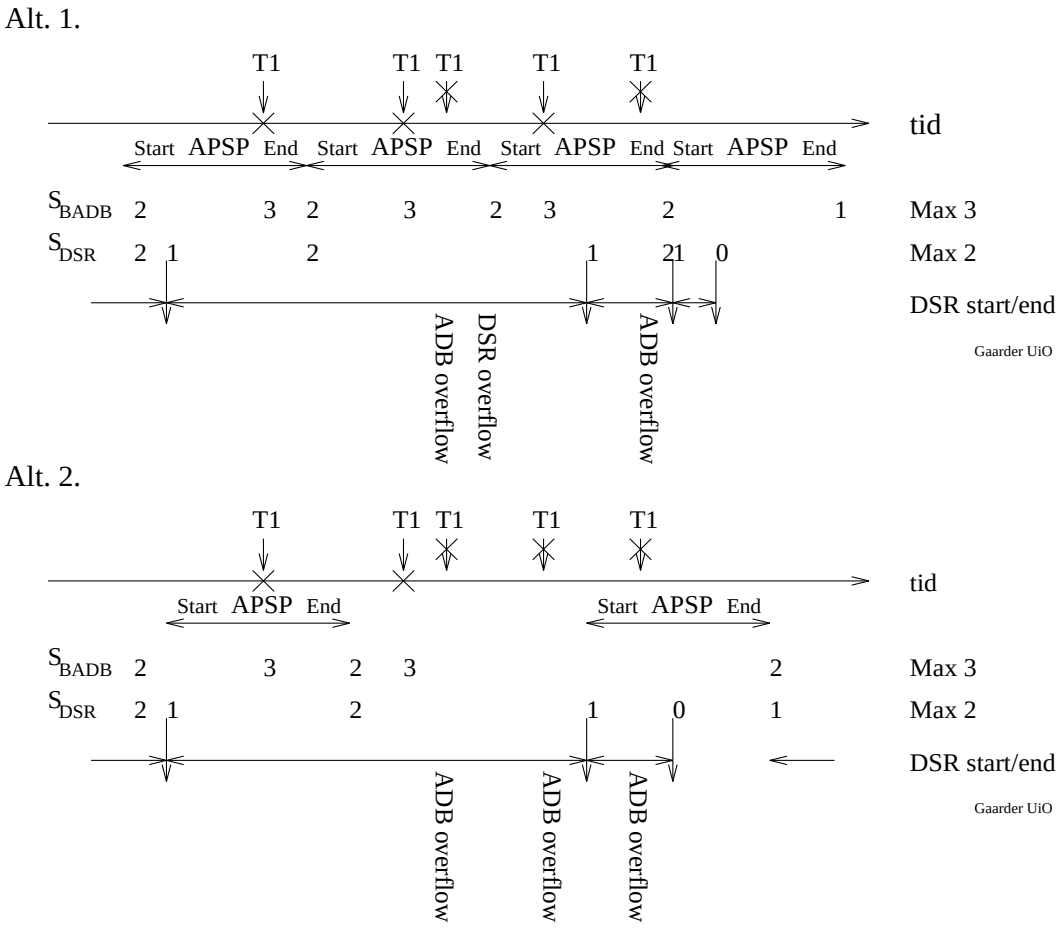
   Ulempen med et slik system kan være at det blir stående mere data i DAQ systemet og tidsforsinkelsen for utlesningen av en hendelse kan bli større. En må simulere systemet

[7] Dette pga. prosesseringstiden for en hendelse da blir lengre.
[8] Hver DM vil ha et forskjellig antall kanaler å lese ut.

for å finne svar på dette. Det vil være meget komplisert å gi tilbakemeldinger hele veien tilbake i DAQ systemet. Det komplekse systemet, der prosessene mer eller mindre blir flettet sammen, vil også føre til en større sannsynlighet for ineffektivitet og vranglåser i DAQ systemet.

4. En kan ha samme løsning som alt. 2, men la APSP holde på verdien til DSR er klar. Dette vil gi samme resultat som alt. 2, bortsett fra at systemet vil fungere som om en hadde et noe dypere buffer.

Figur 3.25 viser hvordan ADB, APSP og DSR vil fungere for noen av alternativene. Her symboliserer tallene antallet buffere som er i bruk for ADB og DSR. Bufferdybden er her $N_{ADB} = 3$ og $N_{DSR} = 2$. Et kryss på tidsaksen under en T1 symboliserer at en trigger er godkjent, mens et kryss over en T1 symboliserer en underkjent T1 eller ineffektivitet. Tiden APSP bruker er fast, mens tiden DSR bruker er tilfeldig for hver hendelse. En kan se at alt. 2 her har gitt et tap av flere triggere [9].



Figur 3.25: *Alternativer for datahåndtering mellom ADB, APSP og DSR.*

[9] Dette er et ekstremt tilfelle med liten bufferdybde og et stort $\frac{t_{APSP}}{t_{T1}}$ forhold

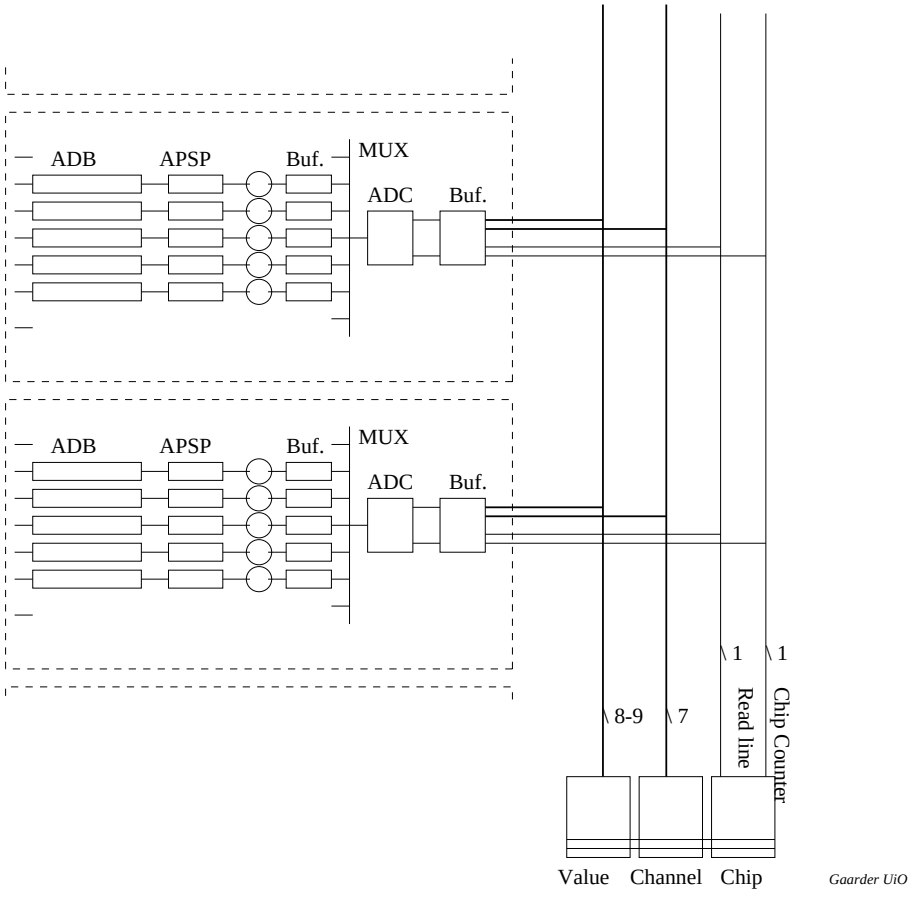
Diskrimineringen som blir gjort av de analoge verdiene fra APSP vil være en parallell prosess og vil derfor i liten grad påvirke flyten av data. Diskriminator-alternativ 3 vil gi mulighet for reduksjon av antallet kanaler, f.eks 4:1, som igjen vil åpne muligheten til å ha et annet utlesningssystem fra FE-brikken. Ved diskriminator alternativ 2 og 3 vil en trolig trenge en viss prosesseringstid før data er klare for DSR. Diskriminatorer er diskutert i kapittel 3.2.6

Multipleksingen (DSR) etter diskriminatoren starter så fort verdien fra APSP er lagt over i DSR bufferet.

ADC etter DSR vil fungere som en slave. ADC vil derfor ikke påvirke dataflyten på annen måte enn at den vil virke som en forsinkelse.

Om det er en ADC for hver FE brikke, eller flere ADC på DM enn det er utlesningsbuffere (PE) på DM vil en måtte ha et multiplexer-system etter ADC.

Figur 3.26 viser et alternativ med en ADC for hver brikke. En vil da ha et digitalt buffer innad i hver brikke etter ADC. Virkemåten kan være som følger:



Figur 3.26: *Utlesningssystem med en ADC for hver brikke.*

ADC'en i en brikke omformer all data som tilhører en hendelse og legger dette i det digitale bufferet med tilhørende kanaladresse.

Brikkelinjen gir en puls for å øke en teller inne i brikken. Når telleren tilsvarer brikken går leselinjen høy om ADC er ferdig med alle kanalene i brikken. De digitale dataene og kanal-

adressene klokkes så ut. Etter at alle dataene tilhørende en hendelse er lest ut går leselinjen lav og brikkelinjen gir en puls for å starte utlesning av neste brikke. Med det samme det digitale bufferet er tomt starter ADC på neste hendelse.

Systemet vil på denne måten ikke blande rekkefølgen på hendelsene som leses ut.

Om bufferet etter ADC har kapasitet til å lagre alle kanalene i en brikke vil en trolig ha et system med minst like bra kapasitet som et system med multipleksing inn til en ADC for alle brikkene. Dette gjelder kun om ADC'ene i de to forskjellige løsningene har en hastighet som tilsvarer antallet ADC multiplisert med hastigheten til hver enkelt ADC. For å finne kapasiteten og tidsforsinkelsen i systemet nøyaktig bør det simuleres.

Fordelen med denne løsningen vil ligge i en redusert hastighet for ADC'ene og en mindre støy ømfintelig overføring av data innad på DM.

Ulempene vil være en mere komplisert og strømkrevende FE brikke, evt. en ulik konvertertering for hver FE brikke, flere referanselinjer og datalinjer på DM.

Fra ADC'en blir de digitale dataene overført til PE som lagrer og pakker dataene. Her vil en ha to muligheter for håndtering av data:

1. La data bli overført fra ADC til PE uavhengig av om PE er klar til å ta imot eller har kapasitet til å håndtere dataene som kommer. En vil da kunne miste data i PE.

   Fordelen med dette systemet er at en får en enklere kommunikasjon og slipper å kommunisere fra PE tilbake til FE brikkene.

   Ulempen vil være at en kan miste deler av en hendelse og at det vil være vanskeligere å holde kontroll på hvilke data en har mistet.

2. La PE gi beskjed til FE brikkene at den ikke har kapasitet til å håndtere mere data. Dette kan enten gjøres ved at en stopper utlesningen etter siste kanal eller en stanser utlesningen etter siste FE brikke. Utlesningen av FE brikkene vil stå i samme posisjon inntil den igjen er klar til å håndtere mere data. Det er uansett viktig at PE har bufferkapasitet slik at den kan ta imot de dataene som fortsatt er i systemet etter at utlesningen er stanset.

   Fordelen med dette systemet er at vil ha bedre orden på data som leses ut og en vil ikke miste deler av en hendelse.

I PE blir data pakket og deretter lest ut av DM. En må regne med at en får en tidsforsinkelse i systemet under denne pakkeprosessen. En kan regne med at det ikke vil være noe tap av data i denne prosessen.

3.4.1 Pakking og reduksjon av data.

En vil i størst mulig grad pakke data slik at mengden data blir minst mulig uten at dette går ut over informasjonen som kommer utav detektor- systemet. Pakkingen av data fører også til at en har bedre kontroll over hendelsene som beveger seg i DAQ systemet.

En har allerede i selve silisiumdetektoren en viss reduksjon ved at en bruker kapasitivt koblede mellomliggende striper som forklart i kapittel 3.2.1.

I diskriminatoren kan en foreta en pakking og reduksjon av data ved at signal fra flere striper slås sammen.

Ved at diskriminatoren har merket de kanaler som har data vil det ved utlesningen av DSR bli gjort en *zero suppression*, dvs. kanaler uten data blir ikke lest ut. Dette vil føre til en kraftig reduksjon av data pga. den lave sannsynligheten for signal.

Det vil ikke være nødvendig å tilegne alle kanalene fra en FE brikke en FE-brikkeadresse. En kan f.eks kun tilegne den første kanal som leses ut av en brikke en FE-brikkeadresse. En kan også foreta pakking av kanaladressene ved at nabokanaler som leses ut kun blir tillagt den første kanalens adresse. Dette vil redusere antallet kanaladresser en leser ut til mindre enn 1/3 pga. at en leser ut nabokanalene til den med signal. Denne pakkingen forutsettes gjort i MC.

En vil da i PE-bufferet lagre 7 bit pga. kanaladresse, 5 bit pga. FE-brikkeadresse og ≈ 8 bit for signalverdi. Totalt 20 byte/kanal. Dette gjelder da uten pakking av kanaladresse og FE-brikkeadresse. For å avgjøre det virkelige gjennomsnittlige antall bit som må lagres i bufferet for hver kanal som leses ut må en simulere systemet. PE vil ved utsendelse fra DM tilegne hendelsen en del protokollsignaler. I denne oppgaven antar vi dette til å flatt være 20 byte/hendelse.

3.5 Utlesningssystemet etter DM.

Etter DM vil en lese dataene mest mulig direkte ut, med minst mulig mellomliggende buffere og prosesser føre DAQ systemet på utsiden av detektortønnen. Dette vil da trolig bli gjort med optiske kabler.

Vi har studert en løsning rettet mot et datahåndtering og utlesningssystem kalt SWIPP , *Switched Interconnection of Parallel Processors*[32] [4]. Figur 3.27 viser hvordan datatransport mellom de forskjellige nodene vil foregå.

Hver node, *Compute Engine* (CE), er koblet til SWIPP nettverket med en *Protocol Engine* (PE) som grensesnitt. Hver PE er igjen koblet til en *Switch* (SW). Hver SW kan typisk ha 16 serielle innganger/utganger som igjen er koblet til nye SW eller PE. Hver DM vil da være en CE med sin egen PE koblet til SWIPP nettverket.

Dataene fra CE blir direkte overført til PE som tar seg av kommunikasjonen med SWIPP nettverket. Internt på DM bør en ha muligheten for å stoppe overføringen av data fra CE om ikke PE har kapasitet til å håndtere mere data, dette hovedsakelig for å ikke dele opp hendelser. Dette er diskutert nærmere i senere kapitler.

PE vil ordne dataene i pakker med adresse og annen nødvendig informasjon innebygget i pakken. Figur 3.28 viser et typisk eksempel på en slik pakke. Pakken blir så serielt sendt ut til den SW som PE er koblet til. SW vil vha. den adresse som ligger i pakken transportere pakken direkte igjennom SW og ut på rett utgang. Pakken vil så bli transportert gjennom systemet til den CE som adressen tilsvarer.

SWIPP vil være et fleksibelt system som kan sende data begge veier, samtidig som det enkelt kan modifiseres og utvides der en trenger nye prosesser eller større kapasitet.

3.6 Effektforbruk til detektormodulen.

Figur 3.29 [34] viser et overslag over effekt forbruket for dagens FELIX FE-brikke.

Om en antar $I_d = 270\mu A$ vil den avgitte effekten for hver stripe være $P_{kanal} = 1.74mW$ og totalt for en FE brikke $P_{brikke} \approx 0.22W$ (128 kanaler).

3.6.1 Kjøling av submodulene.

Innerdetektoren vil totalt utvikle flere kW som må transporteres ut av detektoren. I tillegg til at varmen må transporteres ut for å forhindre overoppheting av elektronikken, er det også viktig at en har en stabil temperatur (ned mot $0°C$) uten temperaturgradienter inne i detektoren. Dette

Figur 3.27: *SWIPP nettverkets kobling mot detektorene og DAQ systemet.*

Figur 3.29: *Effekt forbruk for dagens FELIX brikke.*

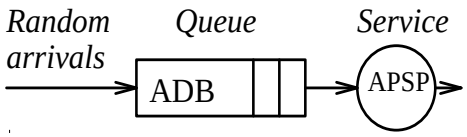
for at målingene skal være presise både med hensyn på signal og geometri. Det vil også være helt essensielt å holde silisiumdetektorene ved en lav temperatur med hensyn på strålingsskader. Valg av kjølemetode vil ha avgjørende rolle for utlegget av DM. Graden av effektiv kjøling vil også ha sterk innvirkning på den effekt en kan tillate forsterkerne å avgi og dermed signal/støy forholdet.

Kapittel 4

Resultater for ADB.

APSP operasjons-tiden er lengre enn tiden mellom hver BC. Dette gjør at en statistisk alltid vil få en sannsynlighet for overflyt i bufferet (ADB) foran APSP, når dette har en endelig lengde. Denne overflyten i ADB vil da være en funksjon av den gjennomsnittlige tiden mellom to triggere (t_{T1}), tiden APSP bruker (t_{APSP}) og dybden på ADB (N_{ADB}).

En kan tenke seg at ADB er en kø med T1 som kunde og APSP som service som figur 4.1 viser. Begrensningen på en APSP for hver utlesningskanal gjør at det er et enkelt-service køproblem.



Figur 4.1: *ADB og APSP som et køproblem.*

En har at den gjennomsnittlige triggerraten er:

$$\alpha = \frac{1}{t_{T1}} = \frac{1}{a_L \Delta t} \tag{4.1}$$

hvor $a_L = \frac{t_{T1}}{\Delta t}$ er gjennomsnittlig antall BC mellom hver T1 som blir gitt. En kan videre anta at kundene, T1, ankommer tilfeldig etter en Poisson fordeling med en sannsynlighet

$$P_r(t) = \frac{(\alpha t)^r}{r!} e^{-\alpha t} \tag{4.2}$$

for at r T1 skal komme i løpet av en tid t. Dette vil være korrekt ved stabil kjøring av eksperimentet.

I alternativ 1 fra kapittel 3.2.5 vil tiden APSP bruker være fast slik at en vil ha et system med konstant service-tid. I alternativ 2 og 3 i kapittel 3.2.5 vil service-tiden APSP bruker være korrelert med aspekter av systemet etter APSP og dermed ikke konstant. Dette kapittel vil kun betrakte systemet frem til APSP og ikke ta hensyn til senere prosesser, dvs. konstant servicetid.

4.1 Analytisk utregning av ADB.

Dødtid pga. destruktiv utlesning av ADB.
Formel 3.3 tilsier at en vil ha en dødtid. Sannsynligheten for at det blir gitt en trigger i løpet av dødtiden vil være:

$$P_{sample} = (1 - (1 - \frac{1}{\frac{t_{T1}}{\Delta t}})^{(S_{APSP}-1)}) \tag{4.3}$$

Tabell 4.1 viser sannsynligheten for dødtid ved gamle og nye LHC parametere for en APSP som hopper over en dataavlesning ($S_{APSP} = N_{APSP} + 1$) og bruker henholdsvis $N_{APSP} = 2$, 3 og 4 dataavlesninger.

Δt	S_{APSP} 3	S_{APSP} 4	S_{APSP} 5
$15ns$	0.30%	0.45%	0.60%
$25ns$	0.50%	0.75%	1.00%

Tabell 4.1: *Beregnet sannsynligheten for å miste en trigger pga. at de kommer for tett.*

Dette tapet av triggere vil gjøre at ineffektivitet pga. bufferoverflyt i ADB blir noe lavere. Dette pga. at en globalt vil undertrykkede triggere som kommer for tett etter hverandre og en får en jevnere og mindre intens triggerrate.

Beregning av bufferet.
Sannsynligheten for å få 0 T1 i løpet av B BC vil være:

$$\beta_0(B) = (1 - \frac{1}{a_L})^B \tag{4.4}$$

når sannsynligheten for å få en T1 ved en BC er $\frac{1}{a_L}$.

Sannsynligheten for å få eksakt A T1 i løpet av B BC vil være:

$$\beta_{\equiv A}(B) = (\frac{1}{a_L})^A \cdot (1 - \frac{1}{a_L})^{B-A} \cdot \binom{B}{A} \tag{4.5}$$

Sannsynligheten for å få A eller flere T1 i løpet av B BC vil være:

$$\beta_{\geq A}(B) = 1 - \sum_{k=0}^{A-1} \beta_{\equiv k}(B) \tag{4.6}$$

Pga. at køen er tidsdiskret og APSP bruker en eksakt tid løses køproblemet ved å beregne sannsynligheten for en forandring i bufferet over en tid t_{APSP}. Antallet muligheter[1] der det kan skje en forandring i løpet av denne tiden vil være $a_A = \frac{t_{APSP}}{\Delta t}$. Om det var 1 eller flere hendelser i bufferet ved t_0 vil da sannsynligheten for at APSP er ferdig, og har fjernet en hendelse fra bufferet, være lik 1 etter t_{APSP}.

[1] Dvs. en har en mulighet for hver klokkepuls, $25ns$ eller Δt.

Sannsynligheten, P_0, for at en har 0 hendelser i bufferet etter a_A BC vil være:

$$P_0 = P'_0 \cdot \beta_0 + P'_1 \cdot \beta_0 \tag{4.7}$$

der P'_0 og P'_1 er sannsynligheten for at det var 0 og 1 hendelse i bufferet respektivt. Det siste leddet gir et bidrag til sannsynligheten for å få et tomt buffer pga. at APSP vil ha fjernet en hendelse i løpet av a_A BC. Forutsetter at systemet er stabilt (P'=P eller $t \to \infty$) og får:

$$P_1 = P_0 \cdot \frac{1-\beta_0}{\beta_0} \tag{4.8}$$

Ved et uendelig langt buffer vil en generelt ha at sannsynligheten for for å ha n hendelser i bufferet:

$$P_n = P_0 \cdot \beta_{\equiv n} + \sum_{k=1}^{n} P_k \cdot \beta_{\equiv n-k+1} + P_{n+1} \cdot \beta_0 \tag{4.9}$$

Som gir:

$$P_{n+1} = \frac{P_n - P_0 \cdot \beta_{\equiv n} - \sum_{k=1}^{n} P_k \cdot \beta_{\equiv n-k+1}}{\beta_0} \tag{4.10}$$

Formel 4.10 vil gi oss sannsynligheten for å ha n hendelser i bufferet når $N_{ADB} = \infty$. En har en maksimum lengde på bufferet som gjør at en må beregne $n = N_{ADB} - 1$ og $n = N_{ADB}$ annerledes. Dette pga. at en ikke har noen sannsynlighet for å få en overgang fra $n = N_{ADB} + 1$ til $n = N_{ADB}$ og at en må legge til sannsynligheten for å få flere T1 enn bufferet har kapasitet til [2]. Denne siste effekten vil påvirke både $P_{N_{ADB}-1}$ og $P_{N_{ADB}}$.

En kan gjøre en tilnærmelse:

$$P_{N_{ADB}} = P'_{N_{ADB}} - P'_{N_{ADB}+1} \cdot \beta_0 \tag{4.11}$$

der $P'_{N_{ADB}}$ og $P'_{N_{ADB}+1}$ er sannsynligheten funnet fra 4.9. En har da tatt hensyn til at en ikke kan oppnå $n = N_{ADB}$ fra $n = N_{ADB}+1$. Etter normering vil en da få en sannsynlighetsfordelning av hvor mange hendelser som er i bufferet som figur 4.2 viser.

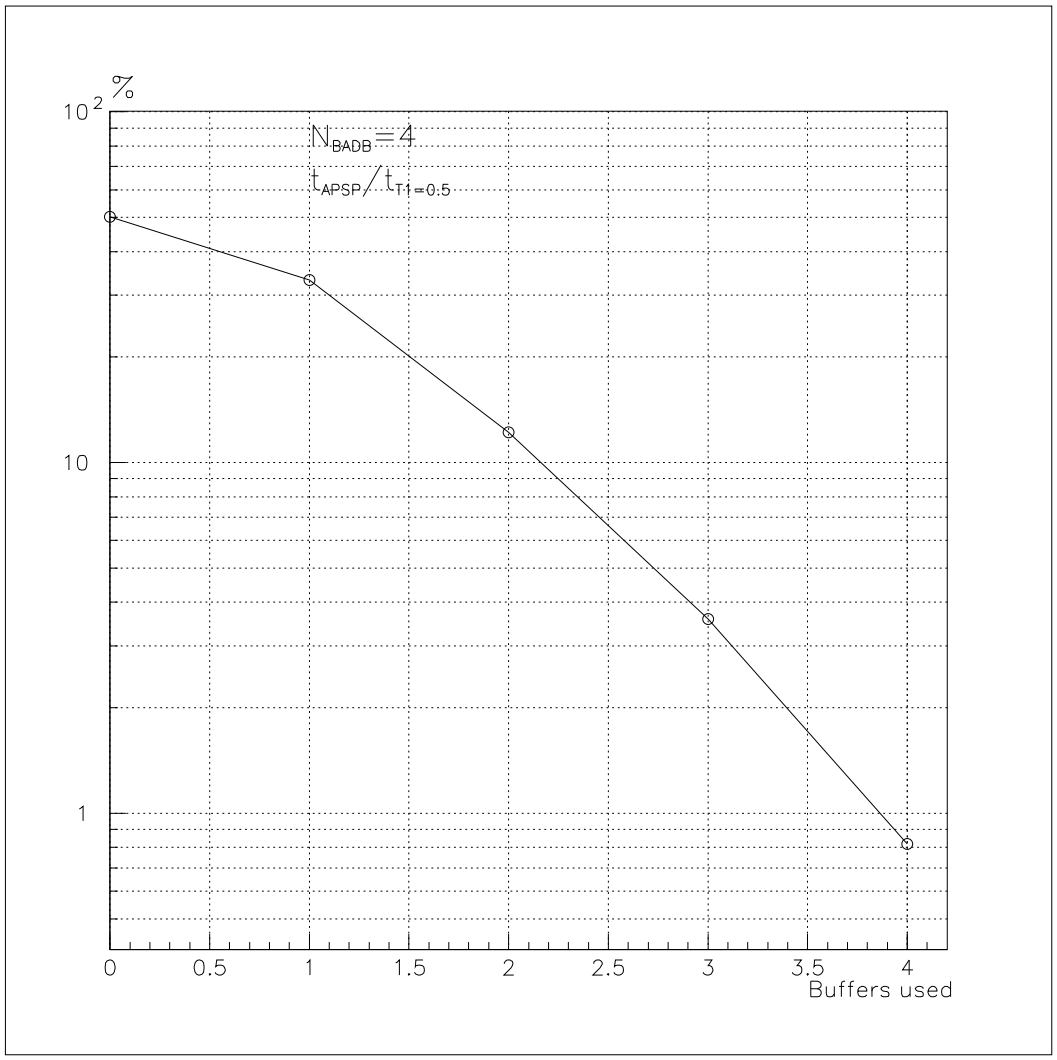
Pga. at triggerene kommer helt tilfeldig vil sannsynligheten for å miste en trigger være den samme som sannsynligheten for å finne bufferet fullt. Ineffektiviteten er gitt ved:

$$1 - \varepsilon_{ADB} = P_{n=N_{ADB}} \tag{4.12}$$

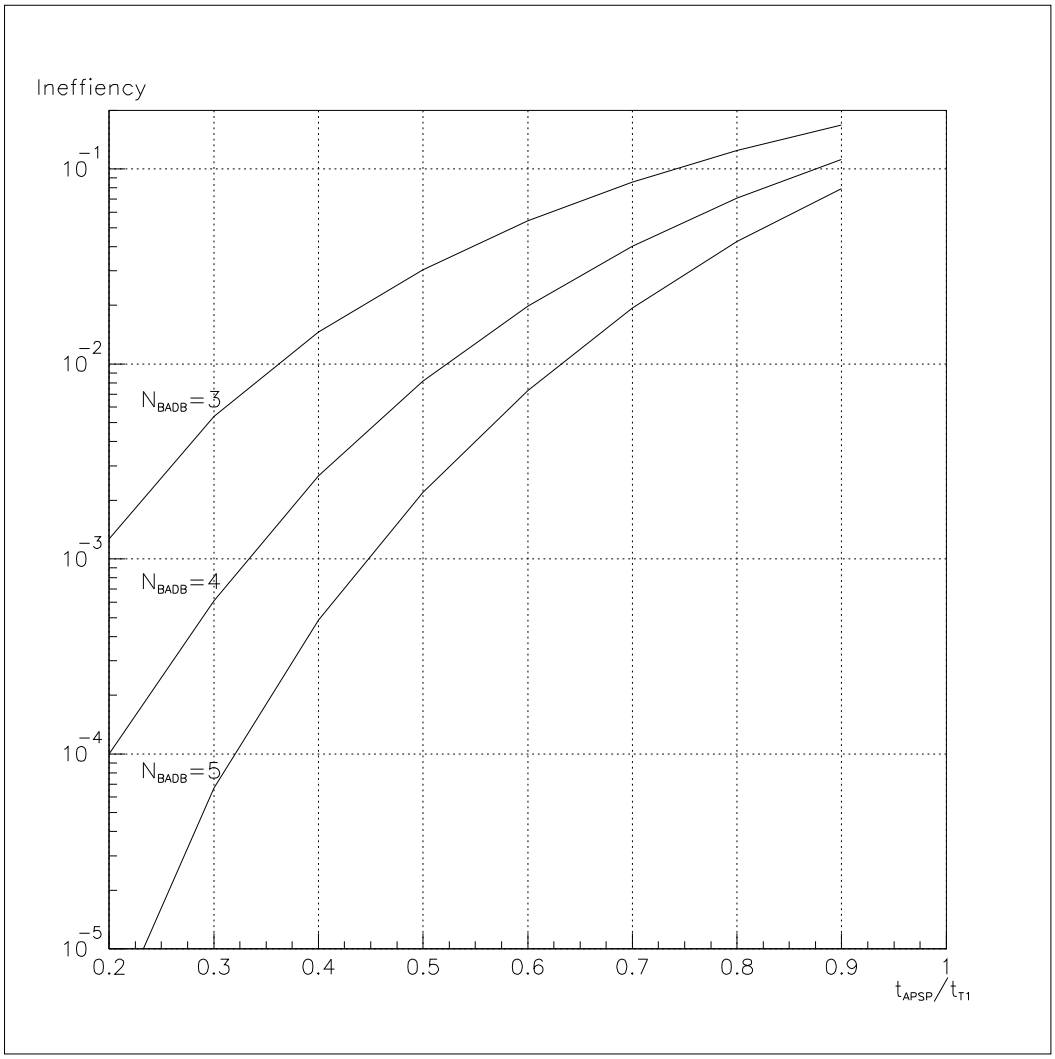
Figur 4.3 viser ineffektivitet til ADB som funksjon av hastigheten til APSP med en bufferdybde på 3,4 og 5 som gitt av formelene ovenfor.

I [28] er sannsynligheten for tap av triggere beregnet på en noe annen måte og det er konkludert med at en vil ha et tap på mellom 0.8% og 1.5% ved en bufferdybde på 4 og $\frac{t_{APSP}}{t_{T1}} = 0.5$. Ved bruk av den nedre grensen i [28], *asynchronous*, vil disse stemme best overens med beregninger og simuleringer foretatt i denne oppgaven. *Asynchronous* vil si at APSP starter å prosessere asynkront med at triggerene gis.

[2] En havner i tilstanden med $n = N_{ADB} - 1$ eller $n = N_{ADB}$ når en har overflyt



Figur 4.2: *Beregnet sannsynlighetsfordelning av hvor mange hendelser som er i bufferet, her med bufferdybde 4 og* $\frac{t_{APSP}}{t_{T1}} = 0.5$.



Figur 4.3: *Beregnet ineffektivitet i ADB bufferet som funksjon av* $\frac{t_{APSP}}{t_{T1}}$*, med en bufferdybde på henholdsvis 3, 4 og 5.*

4.2 Simulering av ADB.

En kan lage modeller av systemet og simulere det. Dette er her gjort vha. programmeringsspråket MODSIM II , (appendix C) og [5][6][7][8], på en standard arbeidsstasjon.

Først er det gjort en simulering av ADB som et standard køproblem, og deretter en fullstendig simulering av virkemåten til ADB med registre som beskrevet i kapittel 3.2.4.

4.2.1 Forenklet simulering.

Programmet og simuleringene som her er brukt er beskrevet i appendix D.1. Tabell 4.2 viser sannsynligheten for å miste en trigger pga. at den kommer for tett etter forrige.

Δt	S_{APSP} 4	S_{APSP} 5
$15ns$	0.45%	0.60%
$25ns$	0.75%	1.00%

Tabell 4.2: *Simulert sannsynligheten for å miste en T1 trigger pga. at de kommer for tett.*

En kan se at de analytiske beregningene og simuleringene stemmer bra overens. Dette er brukt som kryss-sjekk av simuleringsprogrammet.

Figur 4.4 og tabell 4.3 viser ineffektiviteten til ADB som funksjon av hastigheten til APSP. En har her ikke fjernet triggere pga. at de kommer for tett. En kan se at det er noenlunde bra samsvar mellom den simulerte ineffektivitet og beregnede ineffektivitet som sirklene viser.

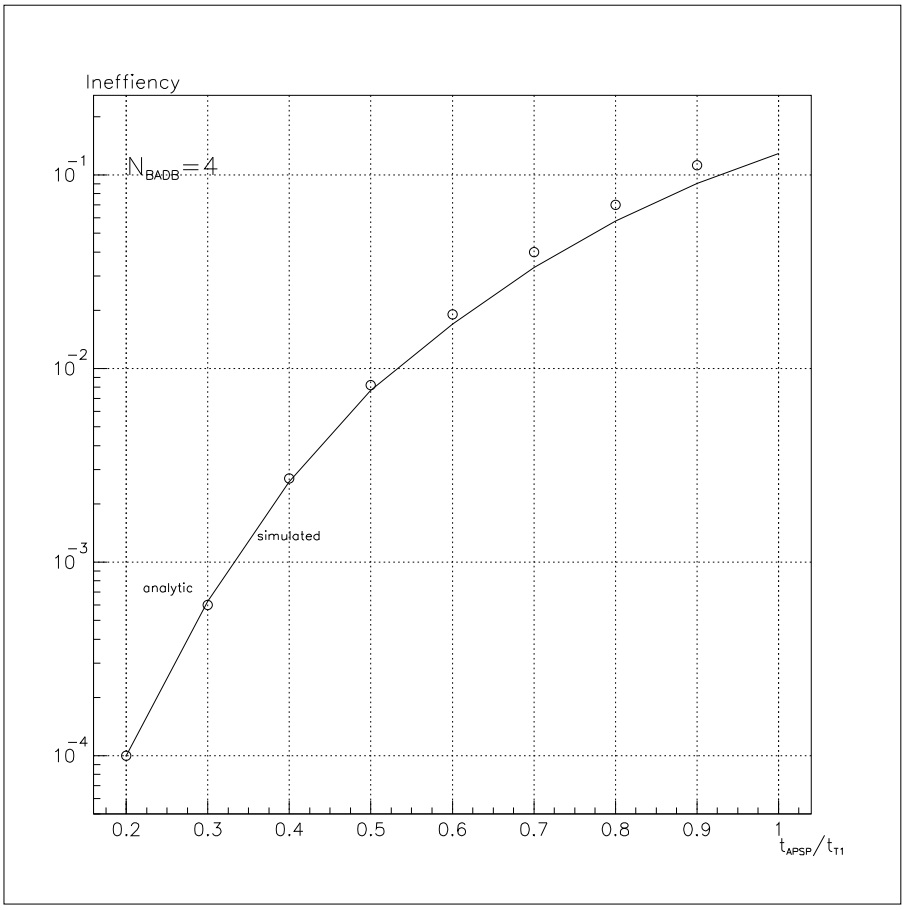
$\frac{t_{APSP}}{t_{T1}}$	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Beregnet tap %	0.01	0.06	0.27	0.82	1.9	4.0	7.0	11.2	-
Sim. tap av T1 %	0.01	0.06	0.26	0.77	1.7	3.3	5.8	9.1	12.9

Tabell 4.3: *Sammenligning av ineffektivitet pga. bufferoverflyt ved analytisk utregning og ved simulering.*

En ser at den beregnede ineffektivitet stemmer bra overens sålenge $\frac{t_{APSP}}{t_{T1}}$ ikke blir stor. Feilen ligger i den tilnærmelsen som er foretatt i ligning 4.11.

En kan også finne ineffektiviteten ved å se på hvor stor andel av tiden bufferet er fullt som ligning 4.12 tilsier. Tabell 4.4 viser dette ved en bufferdybde på 4 og ved $N_{APSP} = 4$. Den nederste rekken viser for sammenligning tapet av triggere ved samme simulering. En har her tatt hensyn til at en vil miste triggere pga. at de kommer for tett og disse er på forhånd fjernet ($N_{APSP} = 4$,$S_{APSP} = 5$). En kan se at det er et bra samsvar.

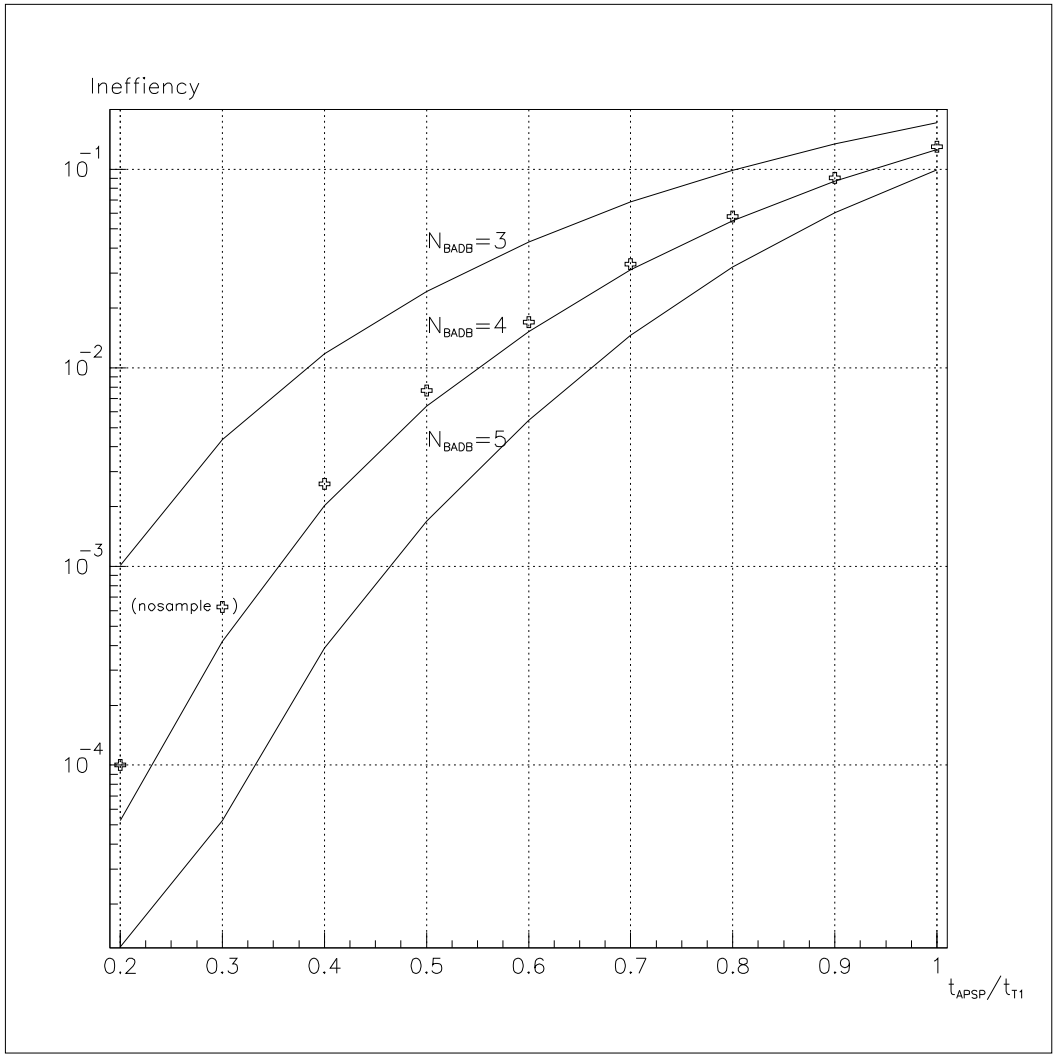
Figur 4.5 viser ineffektivitet til ADB som funksjon av hastigheten til APSP, med en bufferdybde 3,4 og 5. Korsene viser for sammenligning hva ineffektiviteten ville vært om triggerene som kommer for tett ikke var fjernet.



Figur 4.4: *Simulert ineffektivitet i ADB bufferet som funksjon av* $\frac{t_{APSP}}{t_{T1}}$*, med bufferdybde 4 og* $S_{APSP} = 0$.

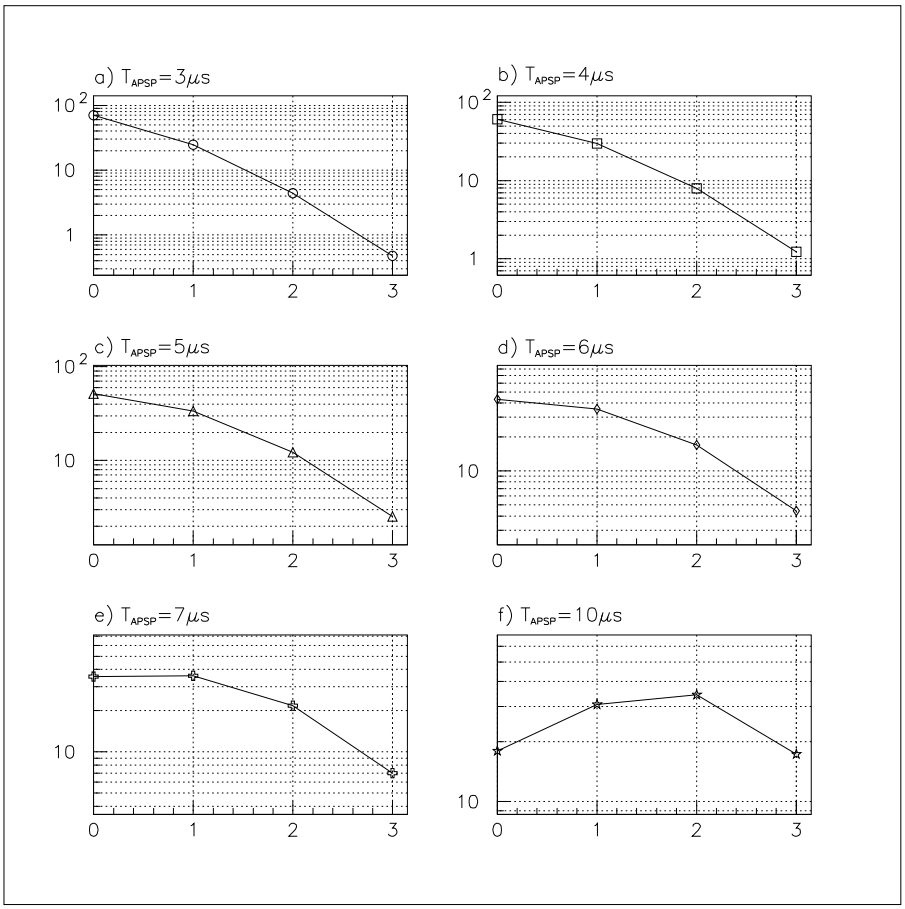
$\frac{t_{APSP}}{t_{T1}}$	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
% tid m. $S_{ADB} = 4$	0.0066	0.051	0.23	0.67	1.59	3.16	5.55	8.81	12.63
% sim. tap av T1	0.0064	0.046	0.22	0.67	1.57	3.19	5.60	8.82	12.72

Tabell 4.4: *Sammenligning av ineffektivitet pga. bufferoverflyt funnet ved tiden bufferet er fullt og direkte ved andelen T1 mistet.*



Figur 4.5: *Simulert ineffektivitet i ADB bufferet som funksjon av* $\frac{t_{APSP}}{t_{T1}}$*, med bufferdybde på 3, 4 og 5. Triggere som kommer for tett er her fjernet. Korsene viser for sammenligning ineffektivitet ved bufferdybde 4 uten at det er tatt hensyn til at triggere som kommer for tett vil bli fjernet.*

Figur 4.6, 4.7 og 4.8 viser fordelingen av hvor mange T1 som er i bufferet med en bufferdybde på henholdsvis 3,4 og 5. Her med $t_{APSP} = 3,4,5,6,7$ eller $10\mu s$ og $S_{APSP} = 5$ BC.

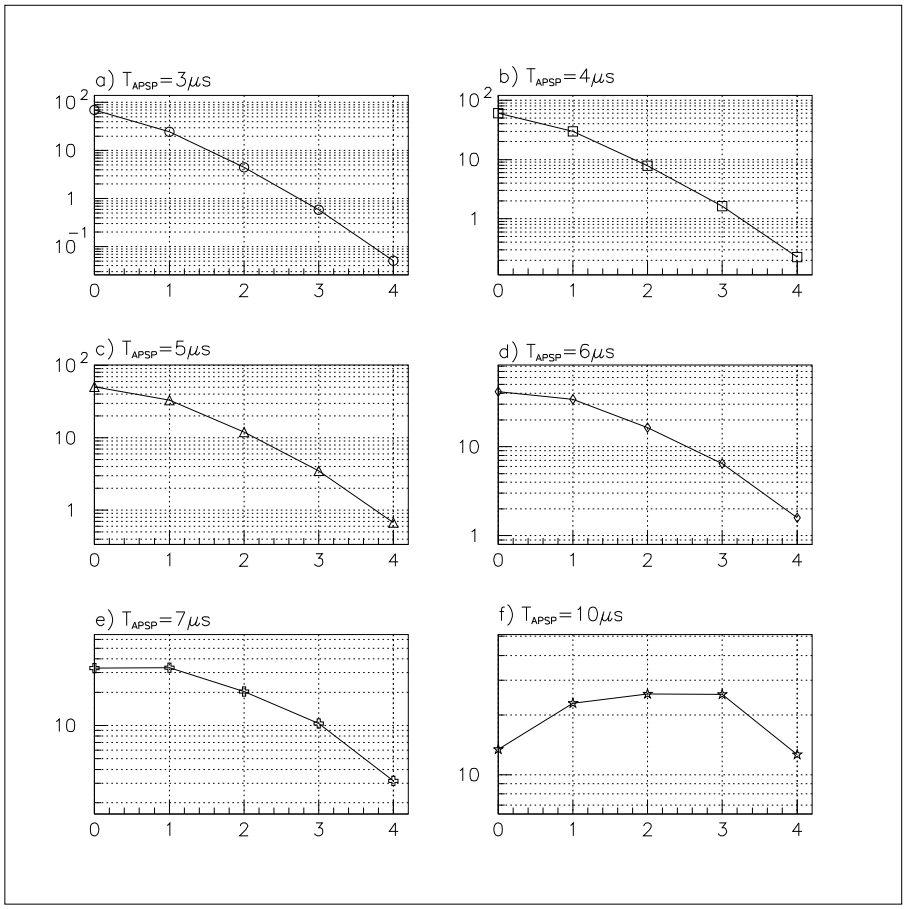


Figur 4.6: *Simulert fordeling av hvor mange hendelser som er i bufferet ved forskjellig APSP hastighet. Her ved en bufferdybde på 3.*

Tabell 4.5 viser den totale ineffektivitet pga. tap av triggere som kommer for tett. Dette for en bufferdybde på 4 og $S_{APSP} = 5$. Den øverste rekken er den totale ineffektivitet, den andre rekken er ineffektivitet pga. bufferoverflyt alene, den tredje er ineffektivitet pga. tap av triggere som kommer for tett og den nederste er andelen av ineffektivitet pga. tap som er fra både bufferoverflyt og triggere som kommer for tett. Alle ineffektivitetene er regnet i forhold til totalt antall triggere.

4.2.2 Fullstendig simulering av ADB.

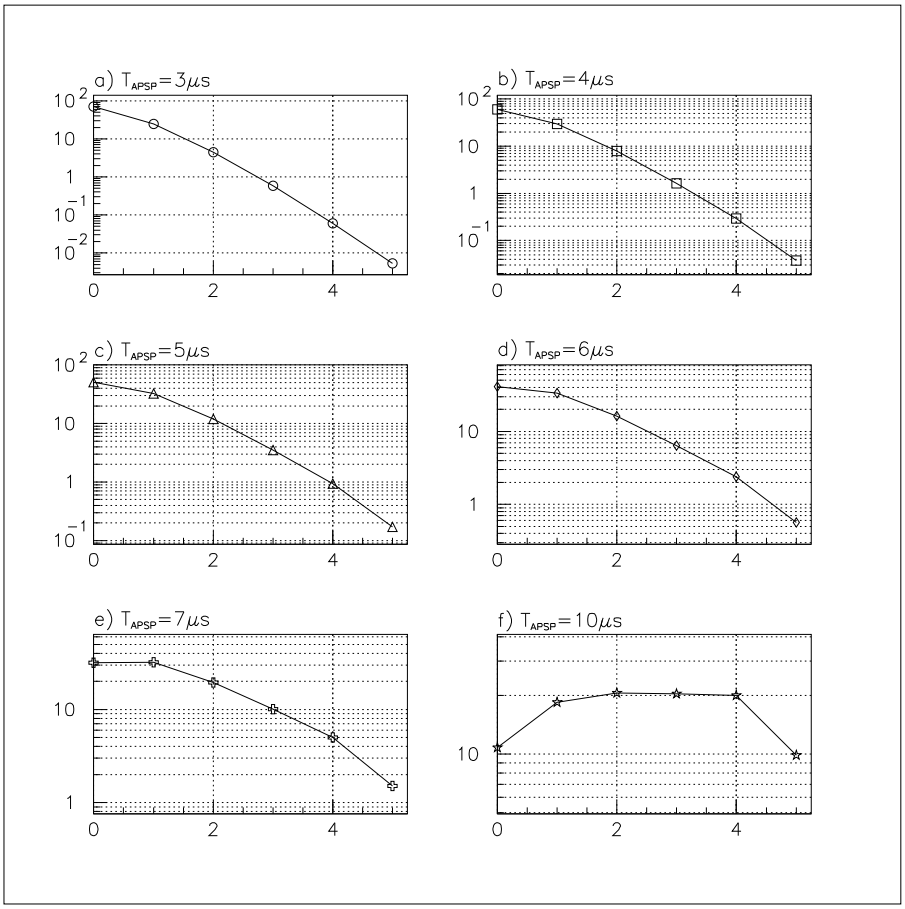
En har foretatt en fullstendig simulering av ADB med registre og pekere. Dette for å kunne gi en pedagogisk forståelse av hvordan den kompliserte logikken virker og for å evt. teste om det er noe avvik fra standard kø modellen.



Figur 4.7: *Simulert fordeling av hvor mange hendelser som er i bufferet ved forskjellig APSP hastighet. Her ved en bufferdybde på 4.*

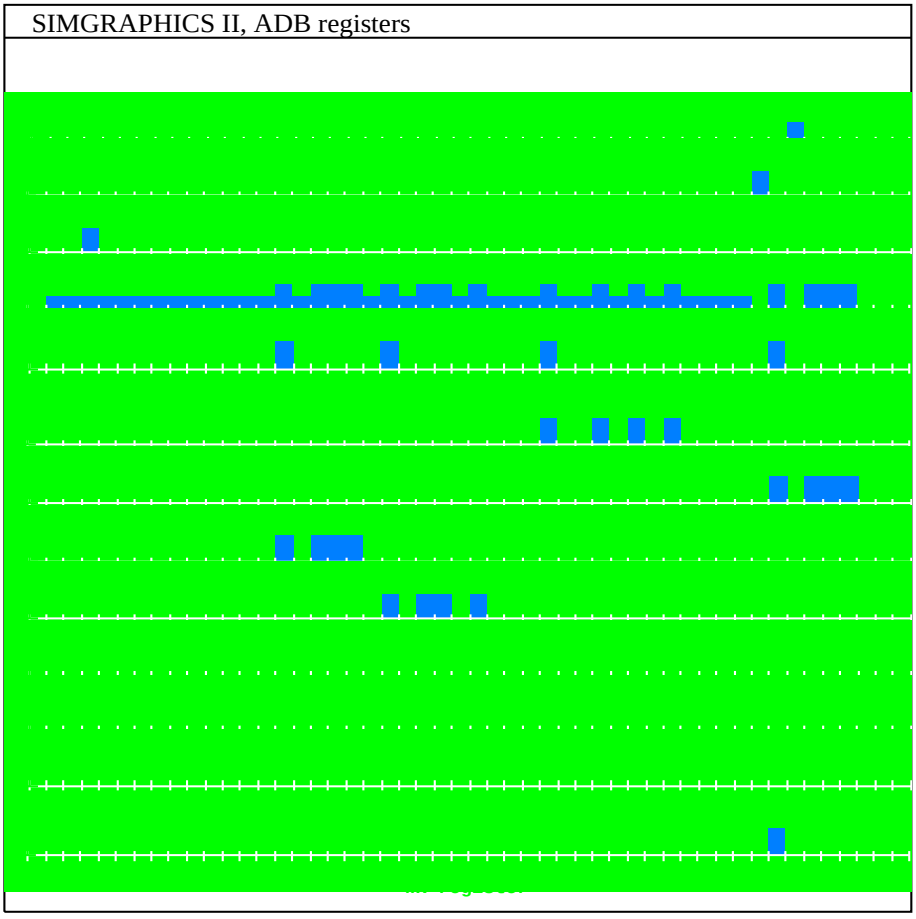
$\frac{t_{APSP}}{T1}$	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$Total\%$	0.995	1.031	1.189	1.620	2.479	4.042	6.385	9.521	13.339
$APSP\%$	0.006	0.042	0.201	0.637	1.505	3.080	5.449	8.622	12.476
$Sample\%$	0.988	0.984	0.973	0.951	0.911	0.856	0.783	0.702	0.617
$Comb.\%$	0.001	0.005	0.015	0.032	0.063	0.106	0.153	0.197	0.246

Tabell 4.5: *Simulert bidrag av ineffektivitet i forhold til totalt antall triggere. Her ved bufferdybde på 4.*



Figur 4.8: *Simulert fordeling av hvor mange hendelser som er i bufferet ved forskjellig APSP hastighet. Her ved en bufferdybde på 5.*

Ved simulering får en opp et grafisk vindu som figur 4.9 viser. Her med $N_{ADB} = 4$, $\frac{t_{APSP}}{t_{T1}} \approx 1$ og 30 pipeline registere.



Figur 4.9: *Grafisk vindu som viser tilstanden til registrene i ADB.*

Registrene i det grafiske vinduet vil tilsvare registrene i figur 3.12. Trigger Tag registeret er representert ved et register der halv høyde tilsvarer *written mode* og hel høyde tilsvarer *trigged mode*.

En kan ikke se at simuleringer med dette programmet gir noen andre resultater enn de forenklete simuleringene av ADB.

4.3 Kommentarer til resultatene.

Det er en god forståelse av dataflyten og tapet av data beskrevet i dette kapittel. En vil ha et synkronisert tap av data og en akseptabel ineffektivitet på godt under 2%. Det er også god overenstemmelse mellom simuleringer og beregninger. En må, ved bruk av den metode som er brukt i denne oppgave, ved beregninger av ineffektivitet være klar over at det er gjort en tilnærmelse ved formel 4.11.

Kapittel 5

Resultater for hele DM.

Som nevnt ovenfor vil det være flere mulige løsninger for DM. I tillegg til å finne karakteristikken til en spesifikk løsning er det også av interesse å få en forståelse for hvordan karakteristikken forandrer seg som funksjon av forskjellige parametere og utlegg av DM. Det er også av interesse å finne hvordan forskjellig grad av støy påvirker DAQ systemet, som f.eks. ineffektivitet, størrelsen på datapakkene som kommer fra DM og fordelingen i tid av disse. En kan gjøre analytiske overslag over karakteristikken til DM, men for å få et fullstendig og mere presist bilde vil det være nødvendig å lage modeller av DM og implementere disse som simuleringsprogrammer.

5.1 Analytiske overslag.

5.1.1 Raten av data ut av DM.

Antallet kanaler som har signal (N_{sig}) for en DM ved en positiv T1 er tilnærmet:

$$N_{sig} \approx (occu + occu_{noise}) \cdot N_{channels} \cdot N_{chips} \tag{5.1}$$

hvor $occu$ og $occu_{noise}$ er sannsynligheten for å ha signal i en kanal pga. partikkeltreff og støy.

Inkludert i dette vil en ha et tilfeldig bidrag av signal pga. jet-strukturer. Fra [22] har en at en $0.5 TeV/c$ jet-struktur vil ha en $20 mrad$ kon og det vil være 30-70 ladete partikler i jet-strukturen.

Typisk vil en ha 2 jet-strukturer for hver interesang hendelse som hver vil dekke 2-3 brikker (dobbeltsidig DM), en vil da ha 4-6 brikker truffet av en jet-struktur for hver hendelse. En kan regne at det er ca. 400 DM med hver ca. 18 brikker ved $r = 20cm$. En vil da få at sannsynligheten for en jet-struktur i en brikke vil være $\approx 0.07\%$.

Om en regner med at det er ≈ 40 partikler for hver jet-struktur vil en ha en sannsynelighet for signal i en kanal tilsvarende $\approx 40\%$ i en brikke der det er en jet-struktur.

En kan da regne med at bidraget til N_{sig} pga. jet-strukturer vil være ≈ 0.6 En må være klar over at dette er grove overslag.

N_{sig} vil da typisk være som tabell 5.1 viser. Dette ved $occu = 0.0045$, $N_{channels} = 128$ og $N_{chips} = 18$.

Om en for hver kanal med signal leser ut nabokanalene på hver side, regner en byte for hver verdi som leses ut, legger til en byte som kanaladresse for hver ny kanal som ikke er nabokanal av forrige, regner en byte som brikkeadresse og legger til 20 byte flatt for diverse informasjon vil

Støy	0.0	0.0015	0.0040	0.0080	0.013	0.02
N_{sig}	10.5	14	19.6	28.8	40.3	56.4

Tabell 5.1: *Beregnet gjennomsnittlig antall kanaler med signal i en DM ved forskjellig støy.*

en få en gjennomsnittlig pakkestørrelse ut av DM på:

$$N_{byte} \approx N_{sig} \cdot 3 + N_{sig} \cdot 1 + N_{chips} \cdot (1 - (1 - (occu + occu_{noise}))^{N_{channels}}) \cdot 1 + 20 \tag{5.2}$$

som fra ligning 5.1 og ovenfor verdier vil gi oss $\approx$ 86byte i hver pakke ved 0.15% støy.

Den gjennomsnittlige raten i byte/s ut av DM vil være:

$$\frac{N_{byte}}{t_{T1}}. \tag{5.3}$$

Figur 5.1 viser raten ut av DM ved forskjellig støy.

5.1.2 Overslag over DSR bufferet.

En kan som en tilnærmelse anta at hendelsene fra APSP fortsatt kommer med en gjennomsnittlig rate α som gitt i ligning 4.1. Videre kan en som en tilnærmelse anta at prosesseringstiden for en hel hendelse av DSR følger en Poissonfordeling med en gjennomsnittlig utlesningsrate:

$$\sigma = \frac{1}{N_{sig} \cdot 3 \cdot t_{DSR}}. \tag{5.4}$$

Fra [20] har en:

$$\rho \equiv \frac{\alpha}{\sigma} = \frac{N_{sig} \cdot 3 \cdot t_{DSR}}{t_{T1}} \tag{5.5}$$

og videre at sannsynligheten for å ha n hendelser i DSR bufferet:

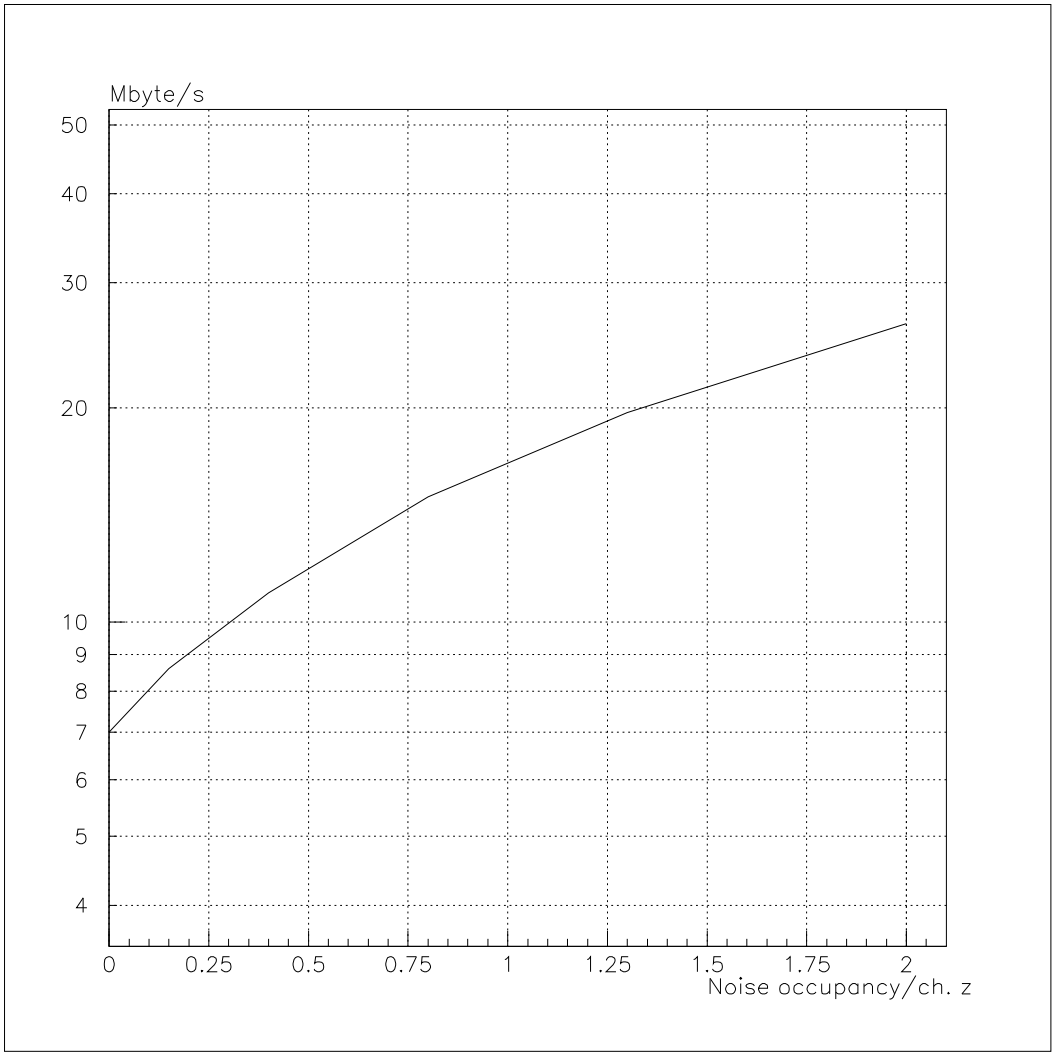
$$P_n = \frac{\rho^n \cdot (1 - \rho)}{1 - \rho^{(N_{DSR}+1)}} \tag{5.6}$$

ved en maksimal dybde N_{DSR} på DSR bufferet.

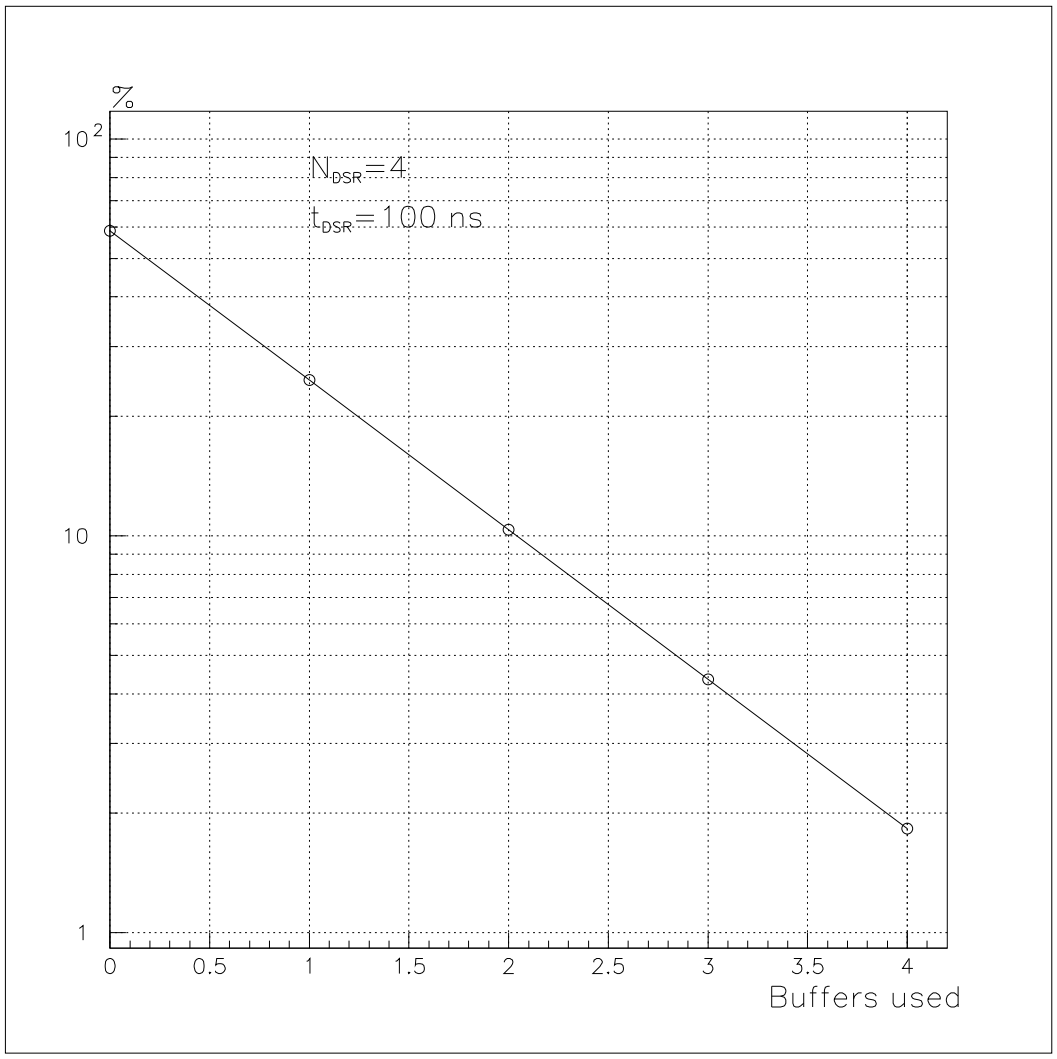
Ved $t_{DSR} = 100ns$, $N_{DSR} = 4$ og $t_{T1} = 10\mu s$ vil dette gi oss en sannsynlighetsfordelning av hvor mange hendelser som er i bufferet som figur 5.2 viser.

En annen metode vil være å grovt anta at prosesseringstiden er konstant og bruke den samme metoden som ble brukt for å beregne ADB bufferet. Med ligning 5.4 innsatt som prosesseringstid vil en få en fordeling som figur 5.3 viser.

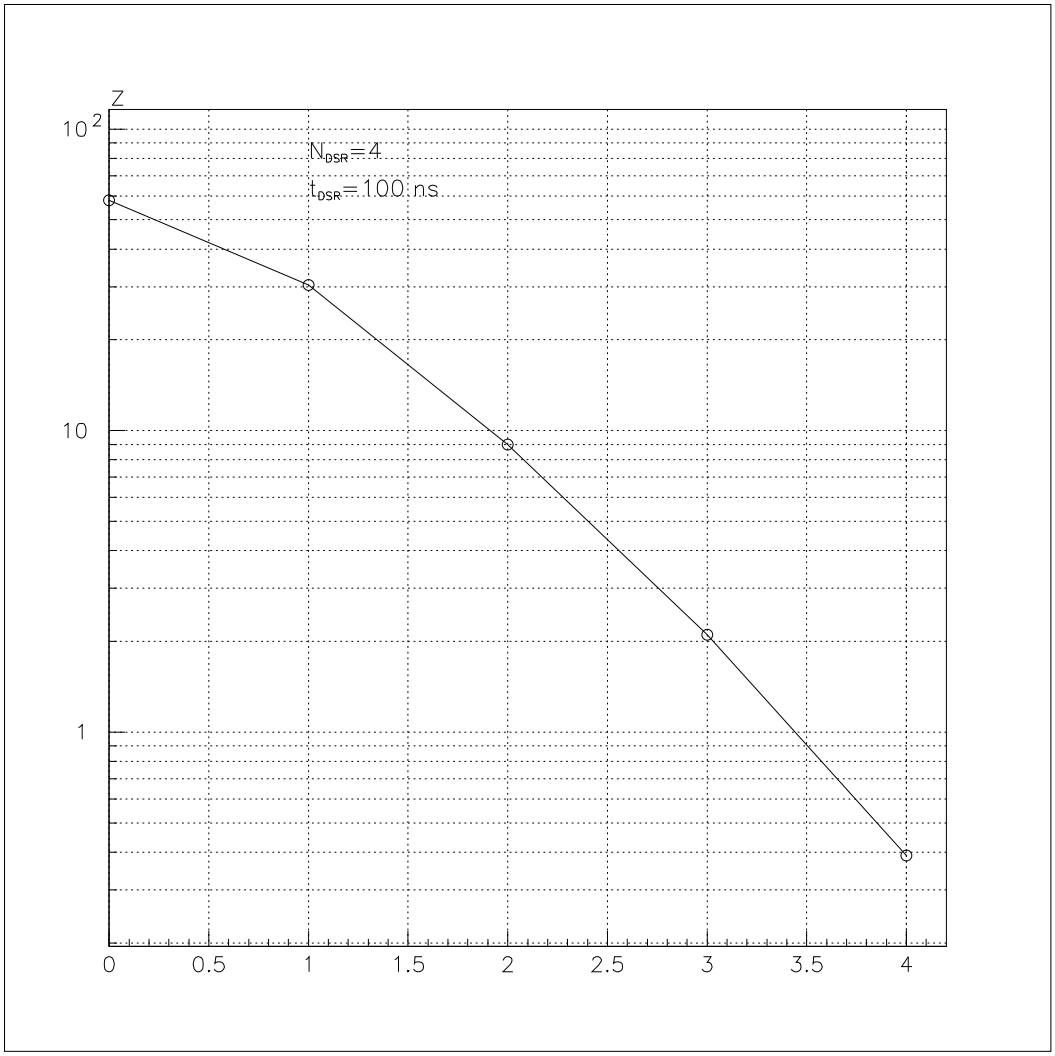
En kan videre finne sannsynligheten for å miste en hendelse i DSR bufferet vha. samme resonnement som i ligning 4.12. Figur 5.4 viser dette for en DSR og ADC hastighet på $10MHz$ og $N_{DSR} = 4$. Og figur 5.5 viser dette ved DSR og ADC hastighet på $13.3MHz$ og $N_{DSR} = 5$. Den siste av de to ovenfor metoder er her benyttet.



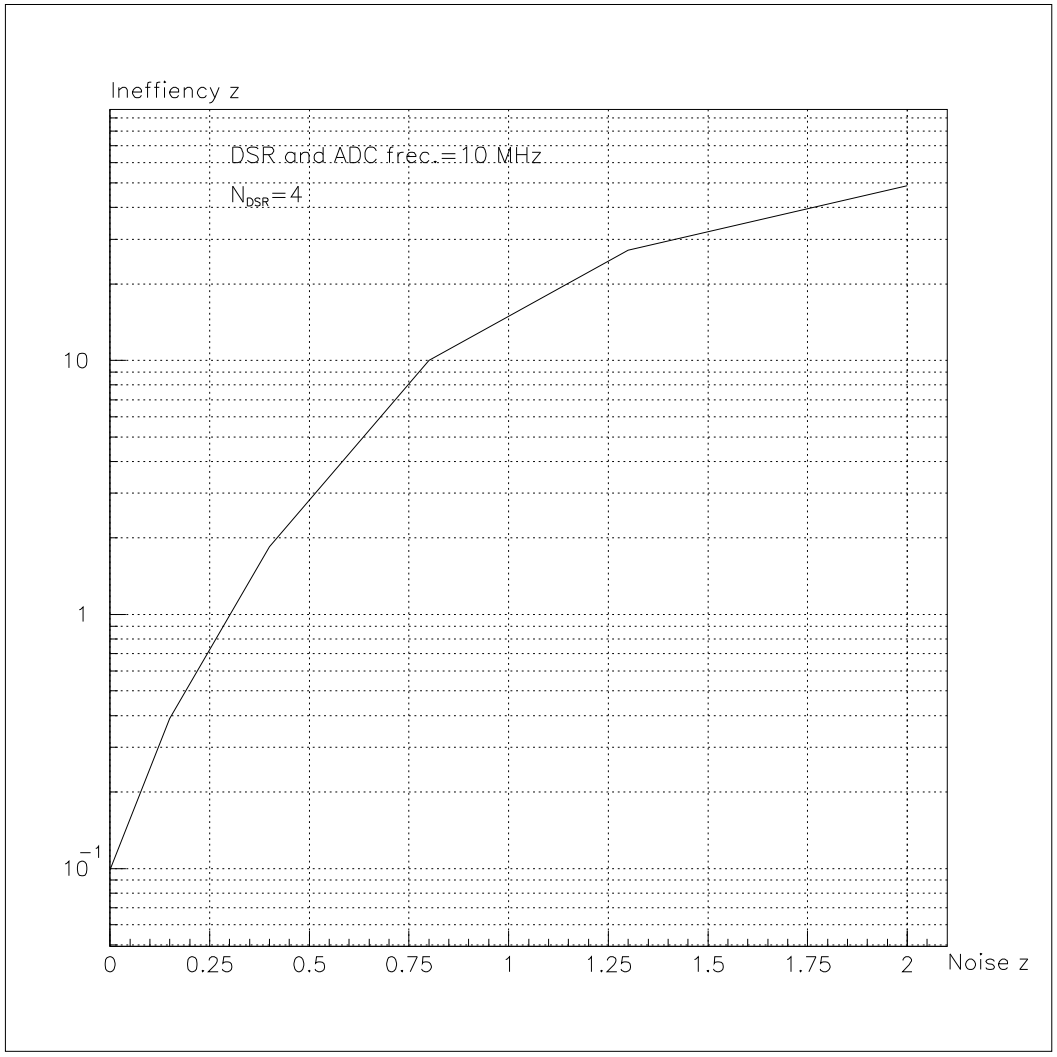
Figur 5.1: *Beregnet Mbyte/s ut av DM som funksjon av støy. Her med en sannsynlighet på* 0.45% *for signal pga. partikkeltreff.*



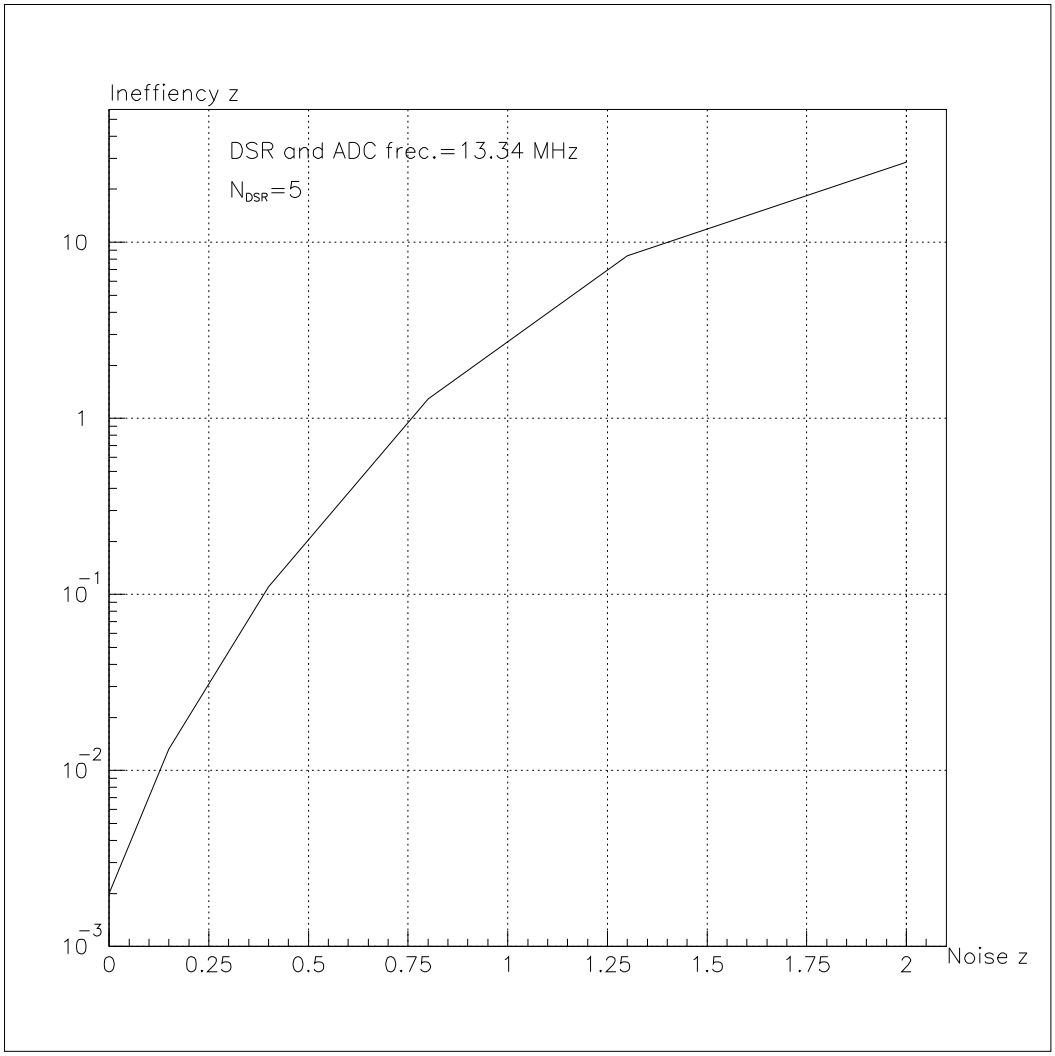
Figur 5.2: *Beregnet sannsynlighetsfordelning av hvor mange hendelser som er i DSR bufferet, her med bufferdybde 4, 0.15% støy og* $t_{DSR} = 100ns$. *Beregnet med metode fra D.R.Cox and W.L.Smith [].*



Figur 5.3: *Beregnet sannsynlighetsfordelning av hvor mange hendelser som er i DSR-bufferet, her med bufferdybde 4, 0.15% støy og* $t_{DSR} = 100ns$*. Beregnet ved egen metode.*



Figur 5.4: *Beregnet sannsynlighet for å miste en hendelse i DSR-bufferet som funksjon av støyen. Her med* $N_{DSR} = 4$ *og* $t_{DSR} = 100ns$*. Beregnet ved egen metode.*



Figur 5.5: *Beregnet sannsynlighet for å miste en hendelse i DSR-bufferet som funksjon av støyen. Her med* $N_{DSR} = 5$ *og* $t_{DSR} = 75ns$*. Beregnet ved egen metode.*

5.1.3 Overslag over prosesseringstid til DM av en hendelse.

En kan gjøre et overslag for tiden fra DM mottar en positiv T1-avgjørelse til hele pakken som tilhører hendelsen er lest ut av DM. Dette vil være tiden hendelsen ligger i ADB, tiden APSP bruker, tiden hendelsen ligger i DSR bufferet, tiden det tar å lese ut av DSR, ADC tiden, pakking og tiden det tar å lese ut av utbufferet. En kan anta at ADC går parallelt med DSR og ikke gir noen forsinkelse, pakkingen går parallelt og ikke tar noe tid og at utbufferet er noenlunde tomt slik at utlesningen av DM starter rett etter at ADC gir ut data fra hendelsen, slik at utlesningen ikke gir noe tillegg i tiden. Et overslag vil da være:

$$t_{DMdelay} \approx \sum_{k=2}^{k=N_{ADB}} P_{S_{ADB}=k} \cdot (k-1) \cdot t_{APSP} + t_{APSP} + \sum_{k=2}^{k=N_{DSR}} P_{S_{DSR}=k} \cdot (k-1) N_{sig} \cdot 3 \cdot t_{DSR} + N_{sig} \cdot 3 \cdot t_{DSR} \tag{5.7}$$

Om en fra kapittel 4.1 antar sannsynligheten ($P_{S_{ADB}=k}$) for å ha 2,3 og 4 hendelser i ADB til å være respektivt 0.122, 0.036 og 0.008. Og en antar sannsynligheten ($P_{S_{DSR}=k}$) for å ha 2,3 og 4 hendelser i DSR bufferet til å være respektivt 0.090, 0.021 og 0.004 vil en få en gjennomsnittlig utlesningstid på $10.9\mu s$. Dette ved $t_{APSP} = 5\mu s$, $t_{DSR} = 100ns$ og 0.15% støy. En kan i tillegg anta at en bruker noe ekstra tid pga. ADC, pakking og utlesning av utbufferet slik at en kan legge til noe tid og får totalt $t_{DMdelay} \approx 12\mu s$.

Figur 5.6 viser forsinkelsen fra en trigger gis til hele hendelsen er lest inn i utbufferet[1,2], dvs. hendelsen er her ikke lest helt ut av DM.

5.2 Simulering av DM.

Tillegg D.3 forklarer hvordan programmene er bygd opp og hvordan de er brukt. Verdiene som normalt er satt for parameterene til DM er også listet opp her. Verdiene brukt for sannsynligheten for jet-strukturer i en brikke avviker noe fra de beregnede, en kan anse verdiene i kappittel 5.1.1 som mest korrekte.

For å få en generell forståelse av DM og hvor tapene er når parametre varieres er det først gjort simuleringer med program som genererer fysikken selv og med en DM med et utlegg som skissert i alternativ 1 i kapittel 3.4, dvs. tap av data i et gitt buffer forskyves ikke til andre prosesser i DM. Senere er det vist noen av resultatene etter simuleringer av andre alternativer av DM for sammenligning. Til slutt er det vist resultater av simuleringer som er basert på Monte Carlo simuleringer av data-mengden fra detektoren.

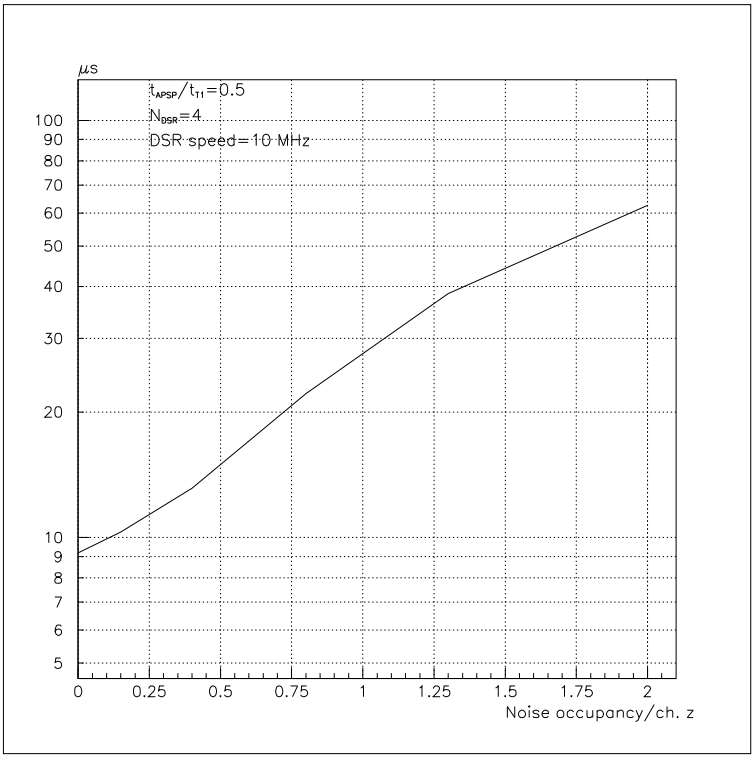
5.2.1 Simulering ved bruk av program som generer fysikken selv.

Generelle resultater.

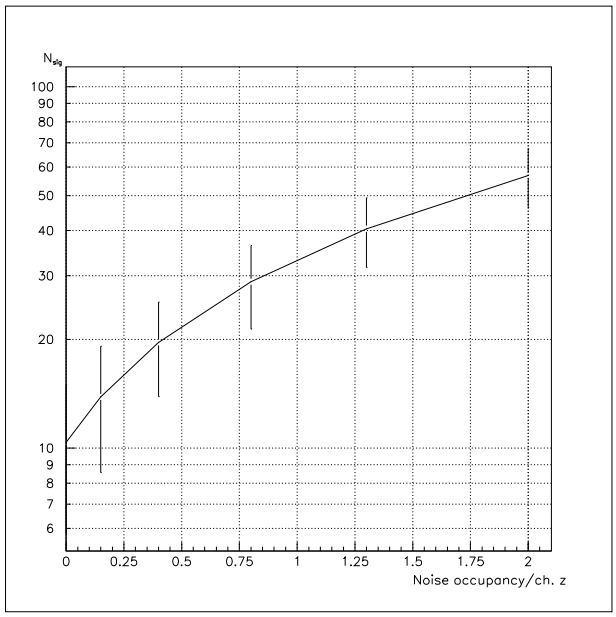
Først er det vist noen generelle resultater uavhengig av alternativet til DM og der parameterene etter ADB er justert slik at det ikke er tap av data i de etterfølgende prosessene til DM. Figur 5.7 viser gjennomsnittlig antall signal i en DM som funksjon av støy. Figur 5.8 viser den gjennom-

[1] Det er ikke lagt til noe tid til pakking

[2] Ligning 5.7 er benyttet med fordeling av hvor mange hendelser som er i DSR bufferet fra simuleringer



Figur 5.6: *Beregnet forsinkelse fra en trigger gis til hele hendelsener lest ut av DM som funksjon av støyen med program som behandler DM som alternativ 1. Her med* $N_{DSR} = 4$ *og* $t_{DSR} = 100ns$.

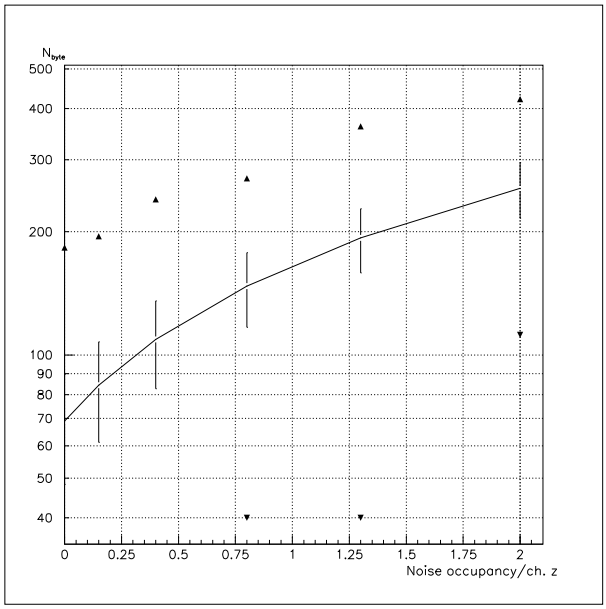


Figur 5.7: *Simulert gjennomsnittlig antall signal i DM som funksjon av støy ved bruk av program som genererer fysikken selv. Her med en sannsynlighet på* 0.45% *for signal pga. partikkeltreff. Strekene tilsvarer* $\pm 1\sigma$.

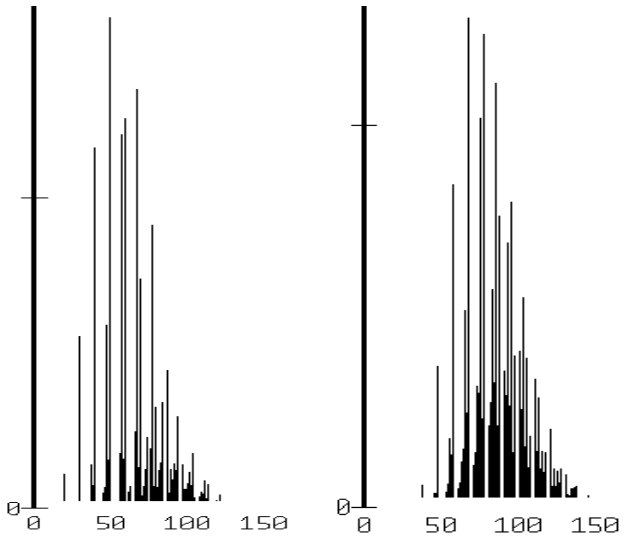
snittlige størrelsen på pakkene ut av DM som funksjon av støy. Den maksimale og minimale størrelsen på datapakken er også vist.

Figur 5.9 og tabell 5.2- 5.4 viser fordelingen av pakkestørrelsen ved henholdsvis 0.0%, 0.15% 0.4%, 0.8%, 1.3% og 2.0% støy.

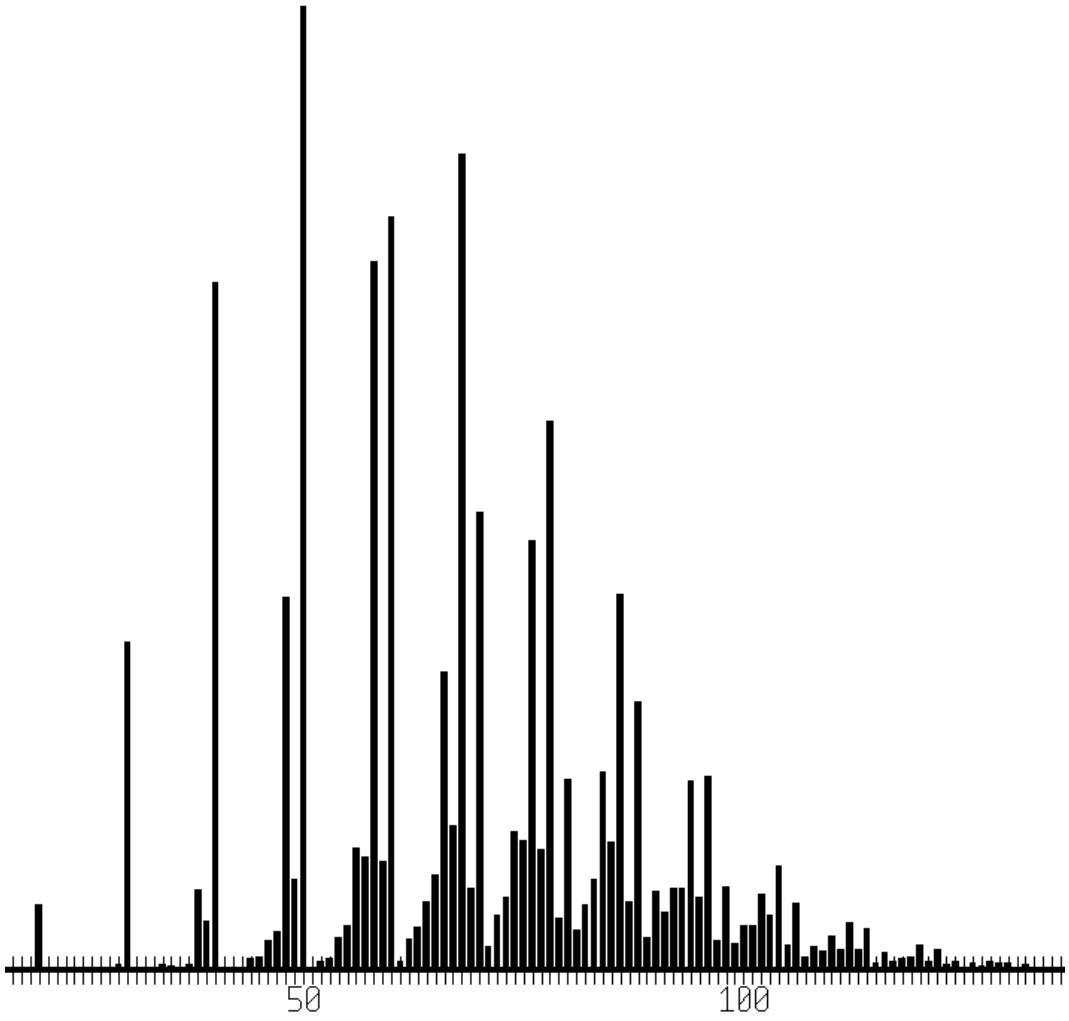
Figur 5.10 viser Mbyte/s ut av DM som funksjon av støy.



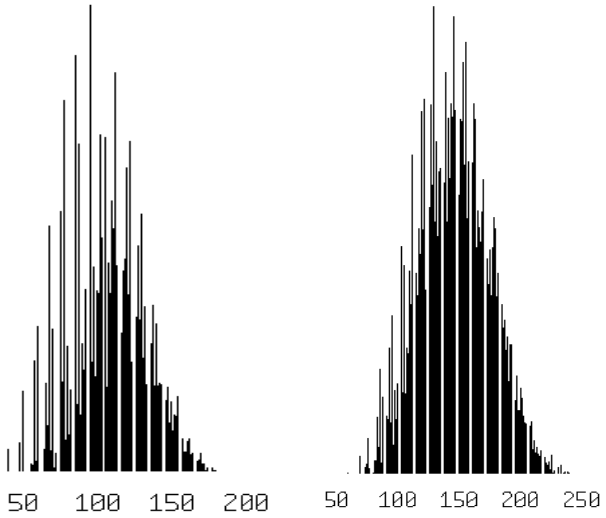
Figur 5.8: *Simulert gjennomsnittlig størrelse på datapakkene ut av DM som funksjon av støy ved bruk av program som genererer fysikken selv. Strekene tilsvarer* $\pm 1\sigma$ *og merkene viser den maksimale og minimale verdien som simuleringen har gitt.*



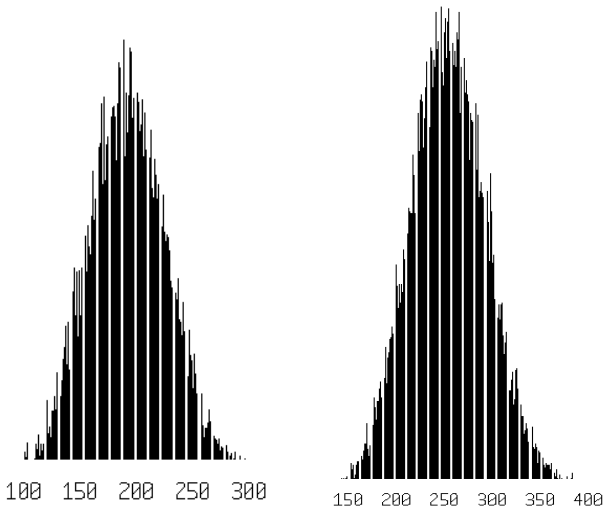
Tabell 5.2: *Simulert fordeling av størrelsen på datapakkene i byte ut av DM ved bruk av program som genererer fysikken selv. Her uten støy og med støy* $= 0.15\%$*. Midlet er her* 68.97 *byte/pakke og* 84.38 *byte/pakke.*



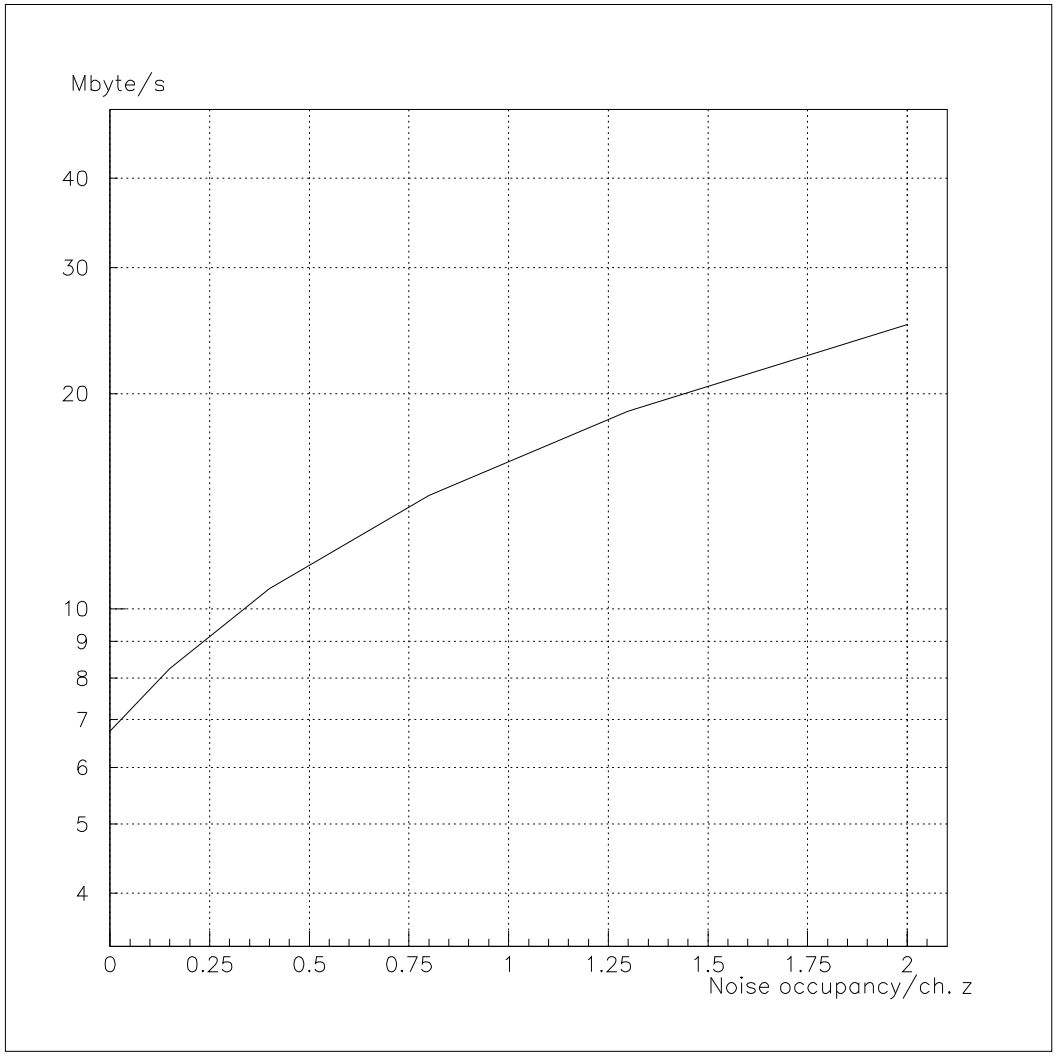
Figur 5.9: *Simulert fordeling av størrelsen på datapakkene i byte ut av DM ved bruk av program som genererer fysikken selv. Her uten støy og uten å generere jet-strukturer. Midlet er her* 68.87 *byte/pakke.*



Tabell 5.3: *Simulert fordeling av størrelsen på datapakkene i byte ut av DM ved bruk av program som genererer fysikken selv. Her med støy*= 0.4% *og* 0.8%. *Midlet er her* 109.14 *byte/pakke og* 147.35 *byte/pakke.*



Tabell 5.4: *Simulert fordeling av størrelsen på datapakkene i byte ut av DM ved bruk av program som genererer fysikken selv. Her med støy*= 1.3% *og* 2.0%. *Midlet er her* 193.2 *byte/pakke og* 255.4 *byte/pakke.*

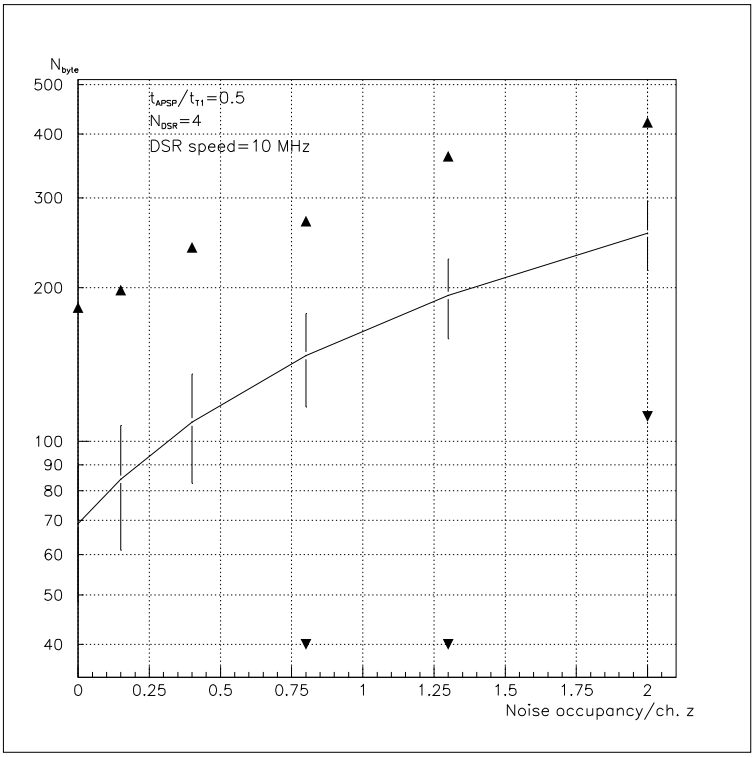


Figur 5.10: *Simulert Mbyte/s ut av DM som funksjon av støy ved bruk av program som genererer fysikken selv.*

Alternativ 1.

Alternativ 1 er simulert som beskrevet i appendix D og håndterer DM som beskrevet i alternativ 1 i kapittel 3.4.

Figur 5.11 viser den gjennomsnittlige størrelsen på datapakkene ut av DM som funksjon av støy ved forskjellige DSR parametere. Parameterene til DM er justert slik at det ikke er tap av data etter ADC. Ved å sammenligne figur 5.8 og 5.11, eller eventuelt 5.12 og 5.4, kan en se at disse er like. Dette viser indirekte at det ikke er noen sammenheng mellom tap av data og størrelsen på hendelsen frem til og med ADC[3].



Figur 5.11: *Simulert gjennomsnittlig størrelse på datapakkene ut av DM som funksjon av støy med program som behandler DM som alternativ 1. Her med* $N_{DSR} = 4$ *og* $t_{DSR} = 100ns$..

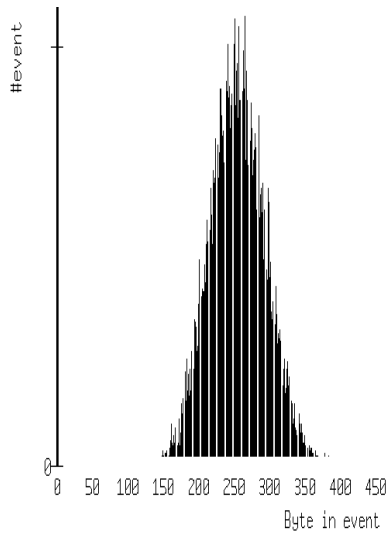
Figur 5.13, 5.14 og 5.15 viser typiske eksempler på strømmen av data ut av DM ved forskjellige parametere.

Tabell 5.5 viser raten ut av DM som funksjon av støy ved forskjellige DSR parametere. En kan se at begrensningen av raten og tapet av data ved stor støy avtar ved å optimalisere DSR parameterene.

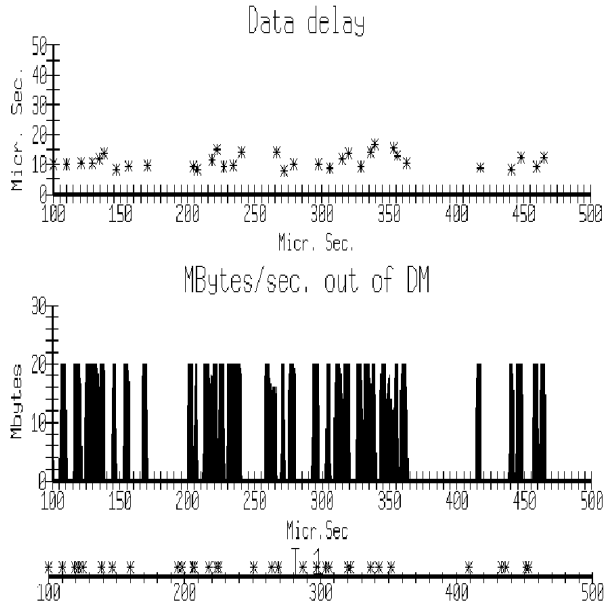
Figur 5.16, 5.17 og 5.18 viser fordelingen av hvor mange hendelser som er i DSR bufferet ved forskjellig støy og DSR parametere.

Figur 5.19, 5.20 og 5.21 viser sannsynligheten for å miste en hendelse pga. DSR bufferoverflyt

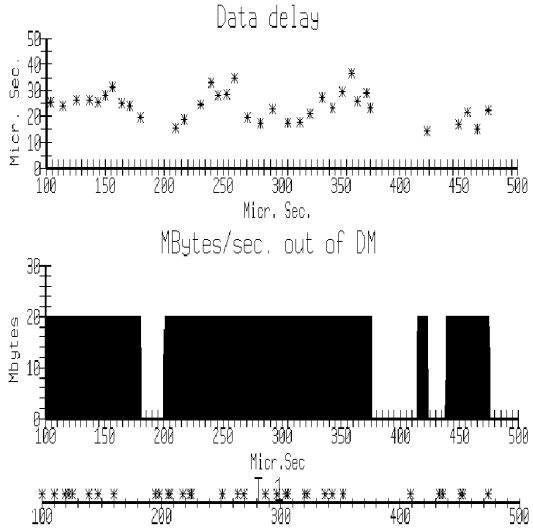
[3]Figur 5.11 har betydelig tap av data ved stor støy



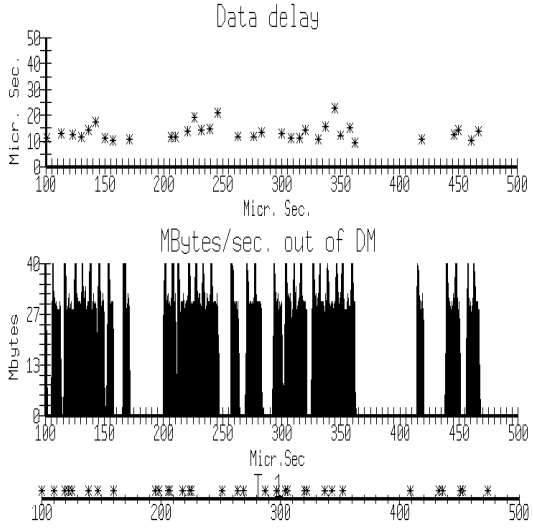
Figur 5.12: *Simulert fordelingav størrelsen på datapakkene i byte ut av DM ved bruk av program som genererer fysikken selv. Her med støy*= 2.0%, $N_{DSR} = 4$ *og* $t_{DSR} = 100ns$. *Midlet er her* 255.8 *byte/pakke.*



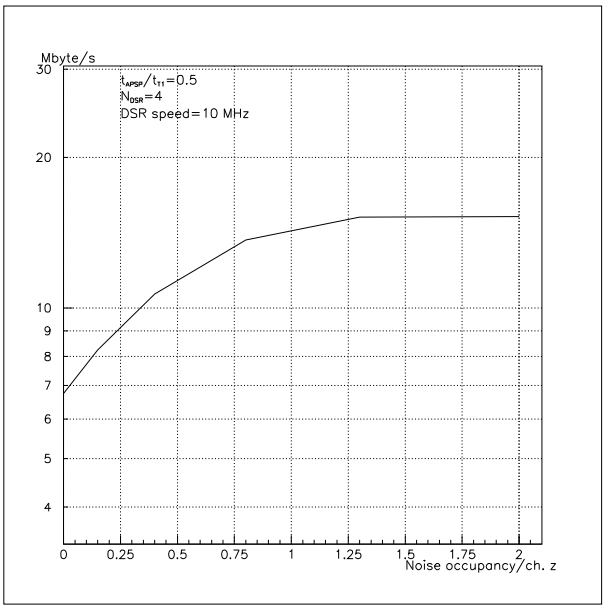
Figur 5.13: *Simulert strøm av data ut av DM med tiden i μs som abscisse og program som behandler DM som alternativ 1. Nederst vises triggerene og øverst forsinkelsen fra triggeren er gitt til hele hendelsen er lest ut av DM. Her med* $N_{DSR} = 4$, $t_{DSR} = 100ns$, *støy*= 0.15% *og utlesningshastighet fra DM på 20Mbyte/s.*

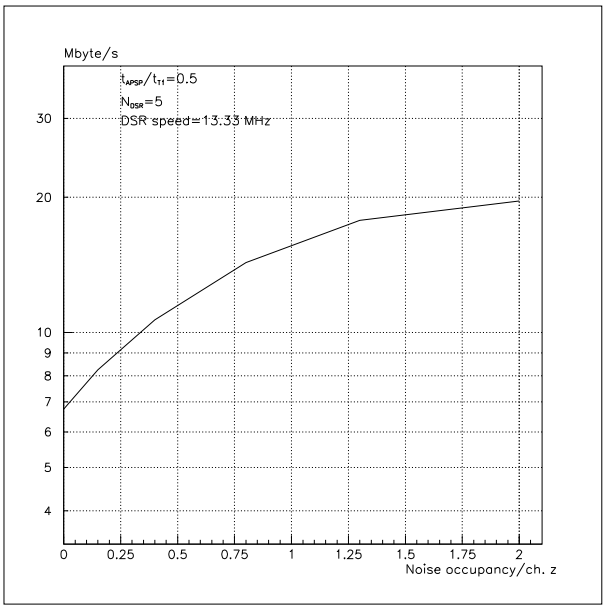


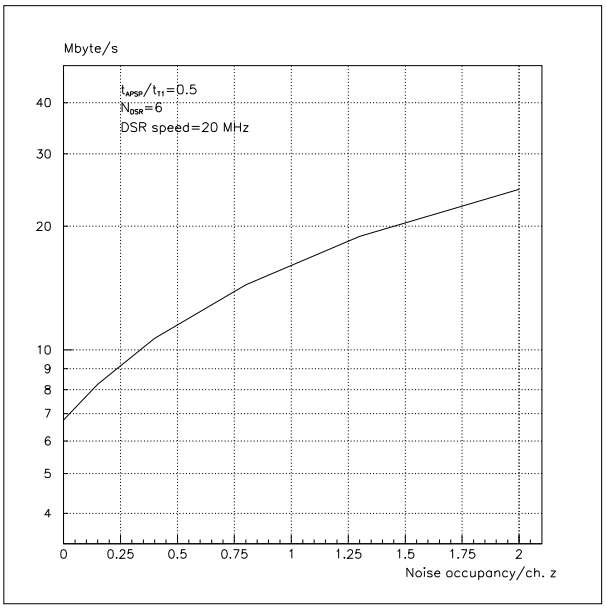
Figur 5.14: *Simulert strøm av data ut av DM med tiden i µs som abscisse og program som behandler DM som alternativ 1. Nederst vises triggerene og øverst forsinkelsen fra triggeren er gitt til hele hendelsen er lest ut av DM. Her med* $N_{DSR} = 6$, $t_{DSR} = 50ns$, *støy*= 1.3% *og utlesningshastighet fra DM på 20 Mbyte/s.*



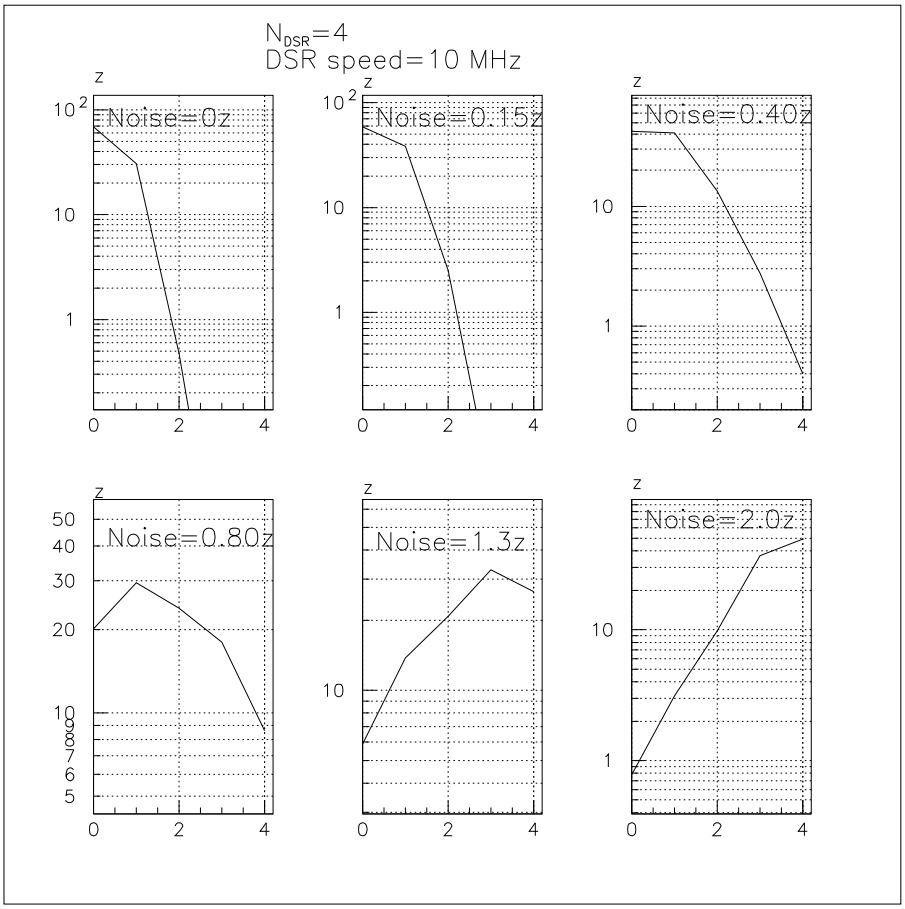
Figur 5.15: *Simulert strøm av data ut av DM med tiden i µs som abscisse og program som behandler DM som alternativ 1. Nederst vises triggerene og øverst forsinkelsen fra triggeren er gitt til hele hendelsen er lest utav DM. Her med* $N_{DSR} = 6$, $t_{DSR} = 50ns$, *støy*= 1.3% *og utlesningshastighet fra DM på 40 Mbyte/s.*



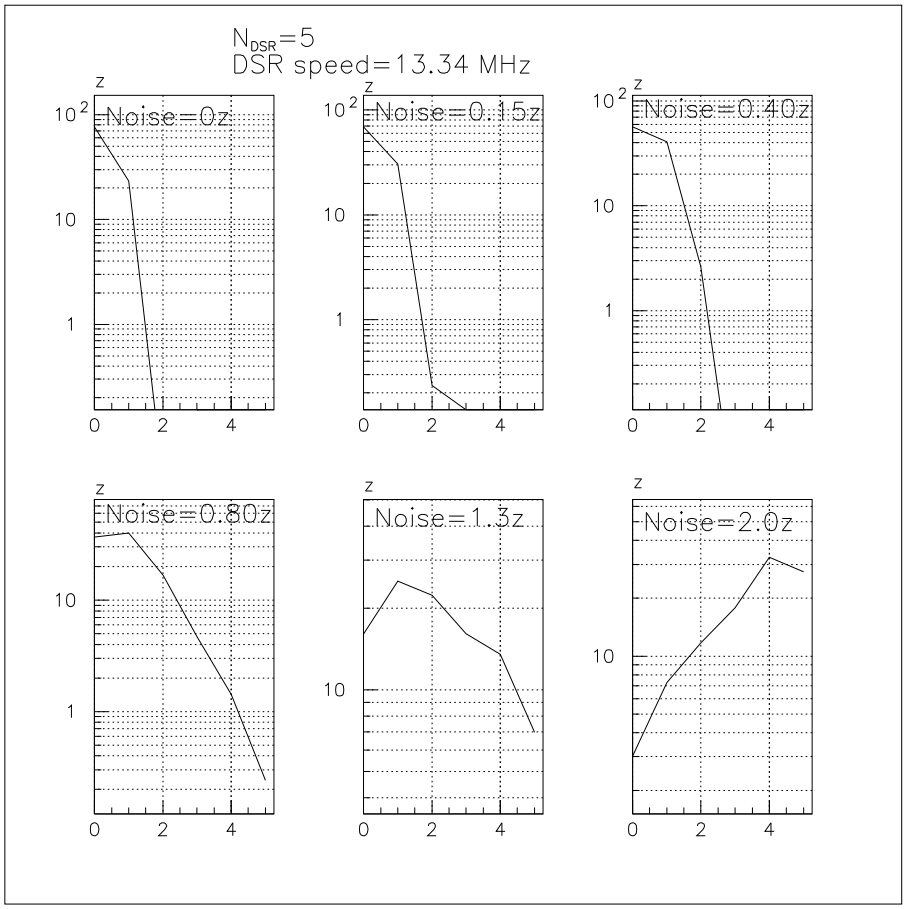




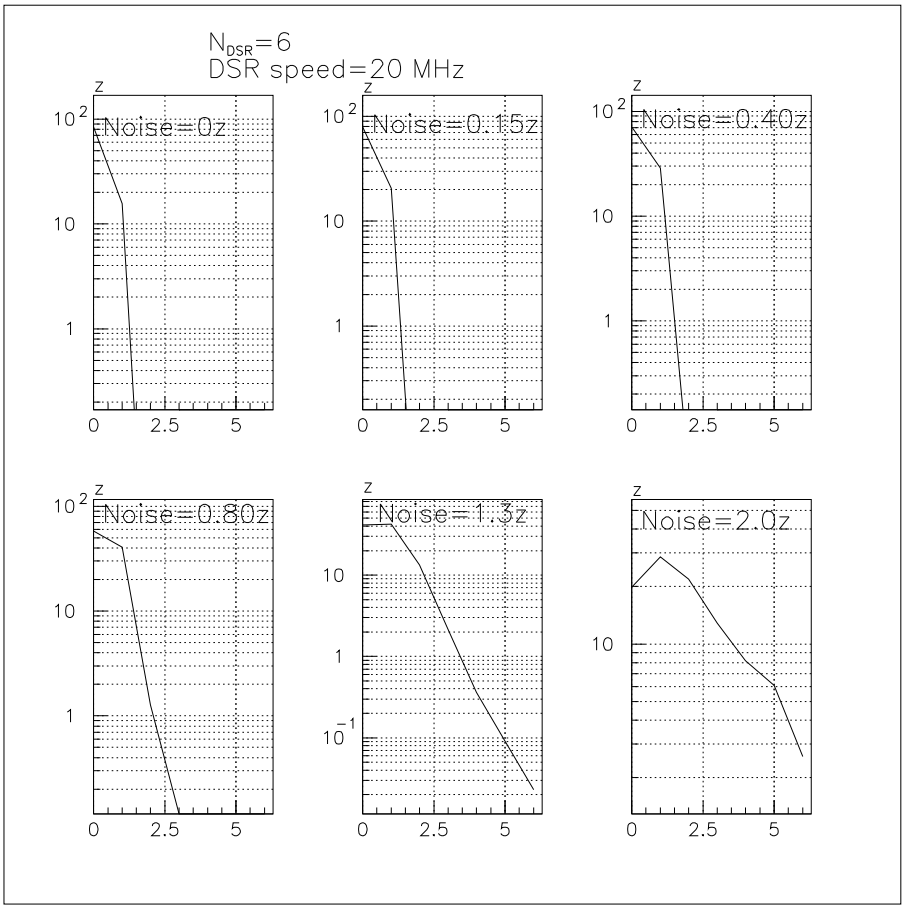
Tabell 5.5: *Simulert Mbyte/s ut av DM som funksjon av støy med program som behandler DM som alternativ 1. Her med* $N_{DSR} = 4$ *, 5 eller 6 og* $t_{DSR} = 100ns$*,* $75ns$ *eller* $50ns$*.*



Figur 5.16: *Simulert sannsynlighetsfordeling av hvor mange hendelser som er i DSR bufferet med program som behandler DM som alternativ 1. Her med* $N_{DSR} = 4$, $t_{DSR} = 100ns$ *og forskjellig støy. Abscissene viser hvor mange hendelser det er i bufferene.*

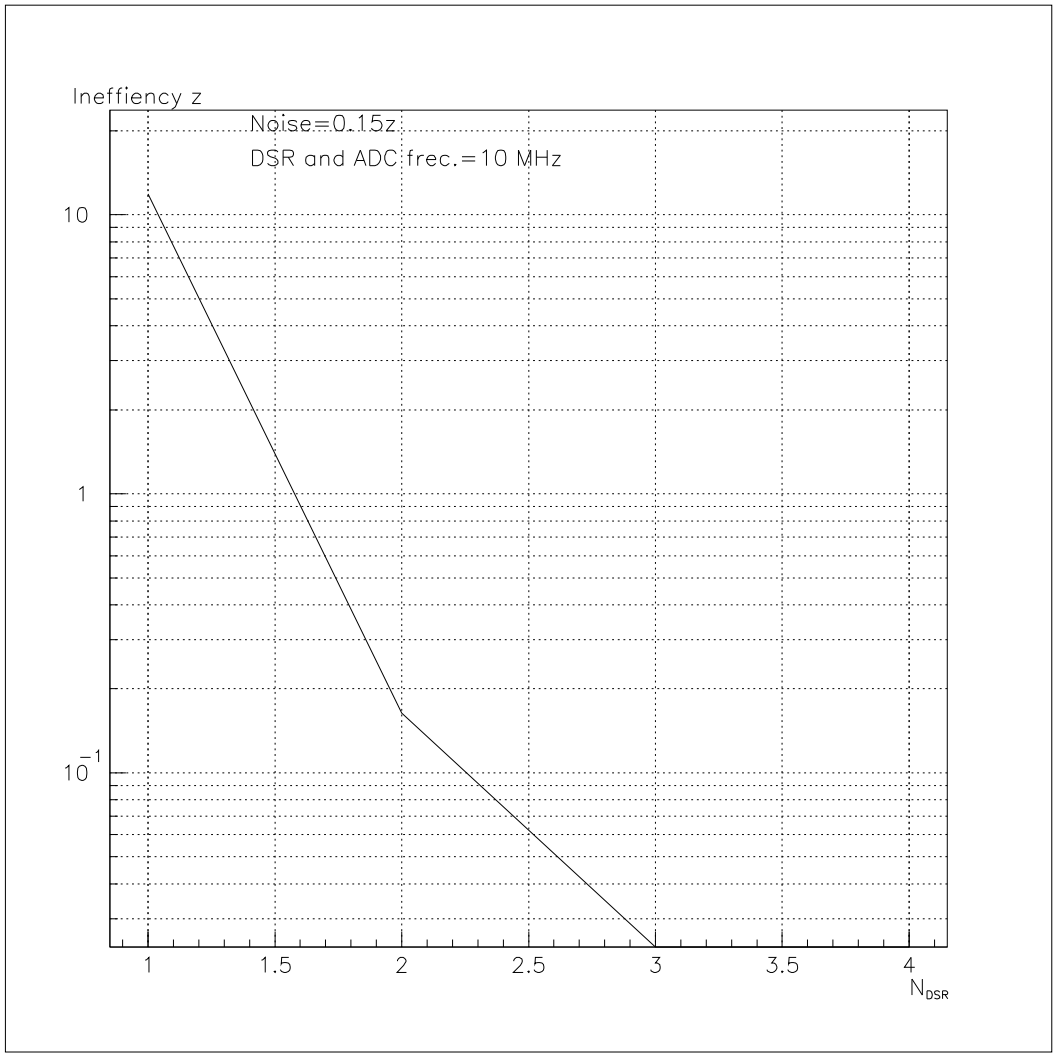


Figur 5.17: *Simulert sannsynlighetsfordeling av hvor mange hendelser som er i DSR bufferet med program som behandler DM som alternativ 1. Her med* $N_{DSR} = 5$, $t_{DSR} = 75ns$ *og forskjellig støy. Abscissene viser hvor mange hendelser det er i bufferen.*



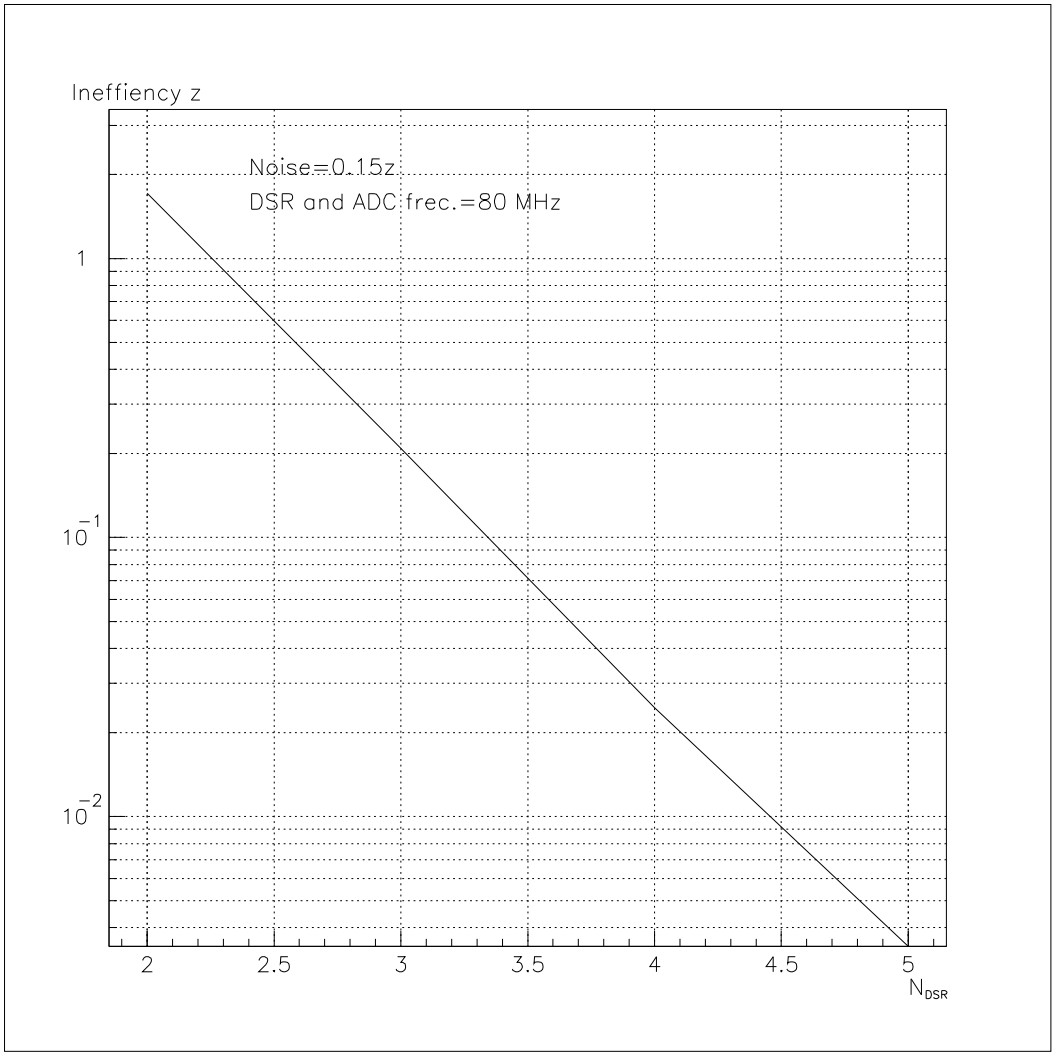
Figur 5.18: *Simulert sannsynlighetsfordeling av hvor mange hendelsersom er i DSR bufferet med program som behandler DM som alternativ 1. Her med* $N_{DSR} = 6$, $t_{DSR} = 50ns$ *og forskjellig støy. Abscissene viser hvor mange hendelser det er i bufferen.*

som funksjon av DSR bufferdybden ved 0.15% støy og forskjellige hastigheter på utlesningen av DSR.

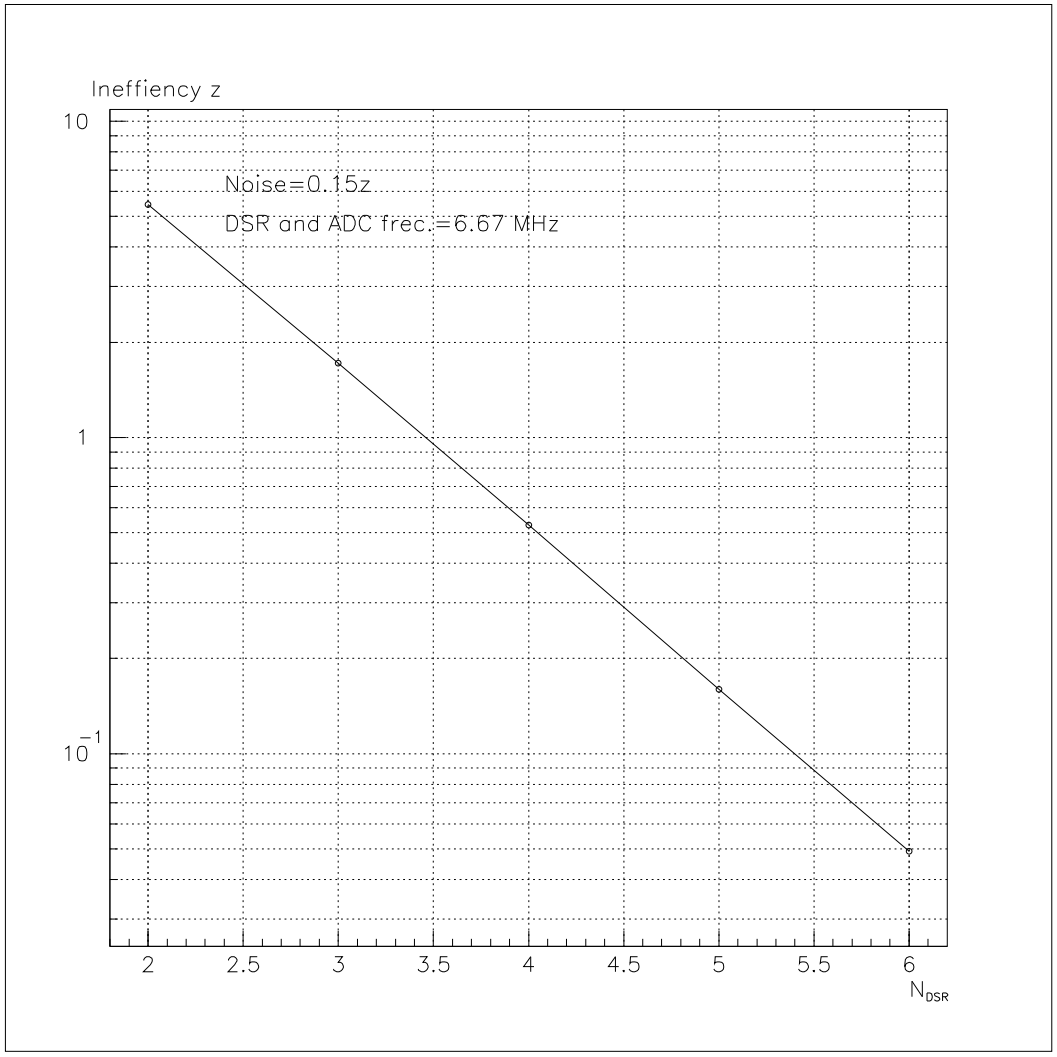


Figur 5.19: *Simulert sannsynlighet for å miste en hendelse i DSR bufferet som funksjon av dybden på DSR bufferet med program som behandler DM som alternativ 1. Her med* $t_{DSR} = 100ns$.

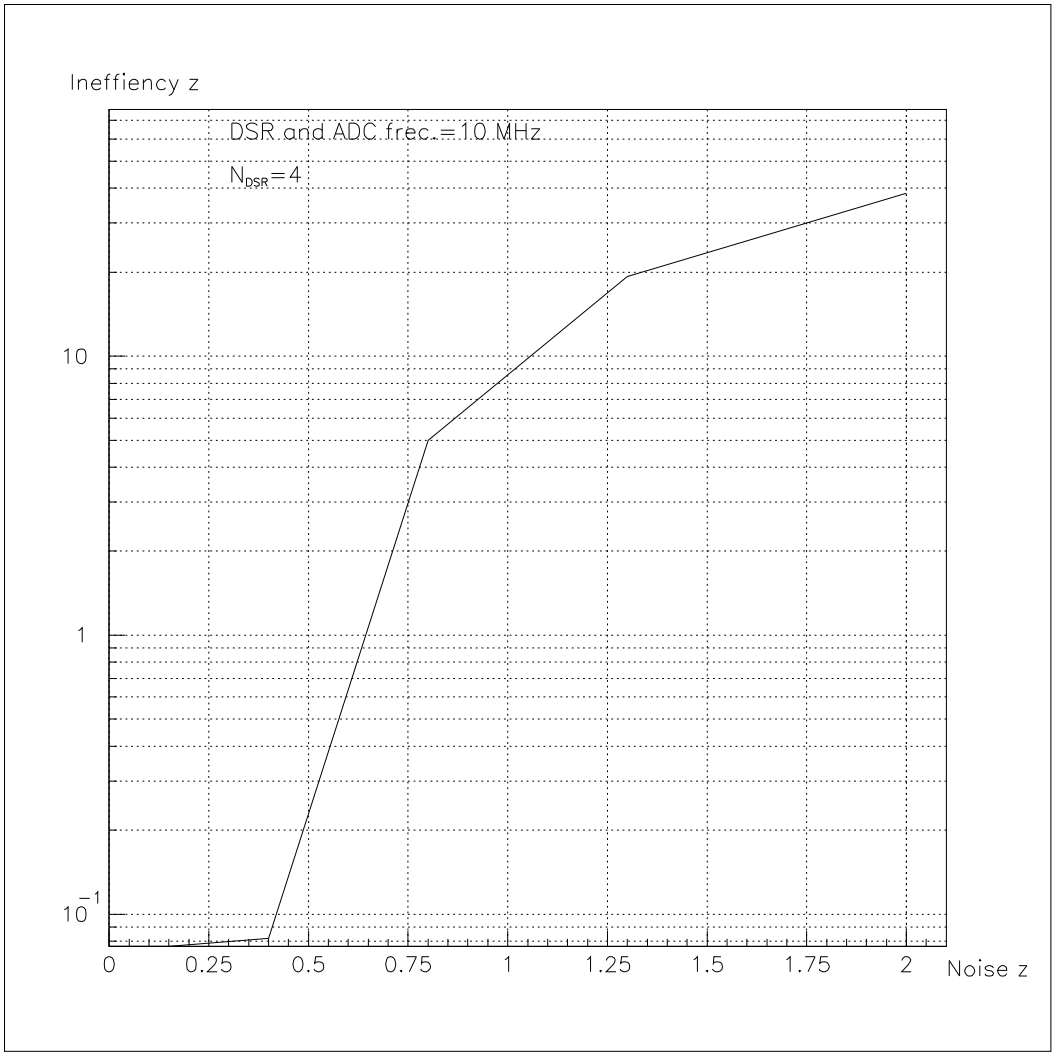
Figur 5.22, 5.23 og 5.24 viser sannsynligheten for å miste en hendelse pga. DSR bufferoverflyt som funksjon av støyen ved forskjellige DSR parametere.



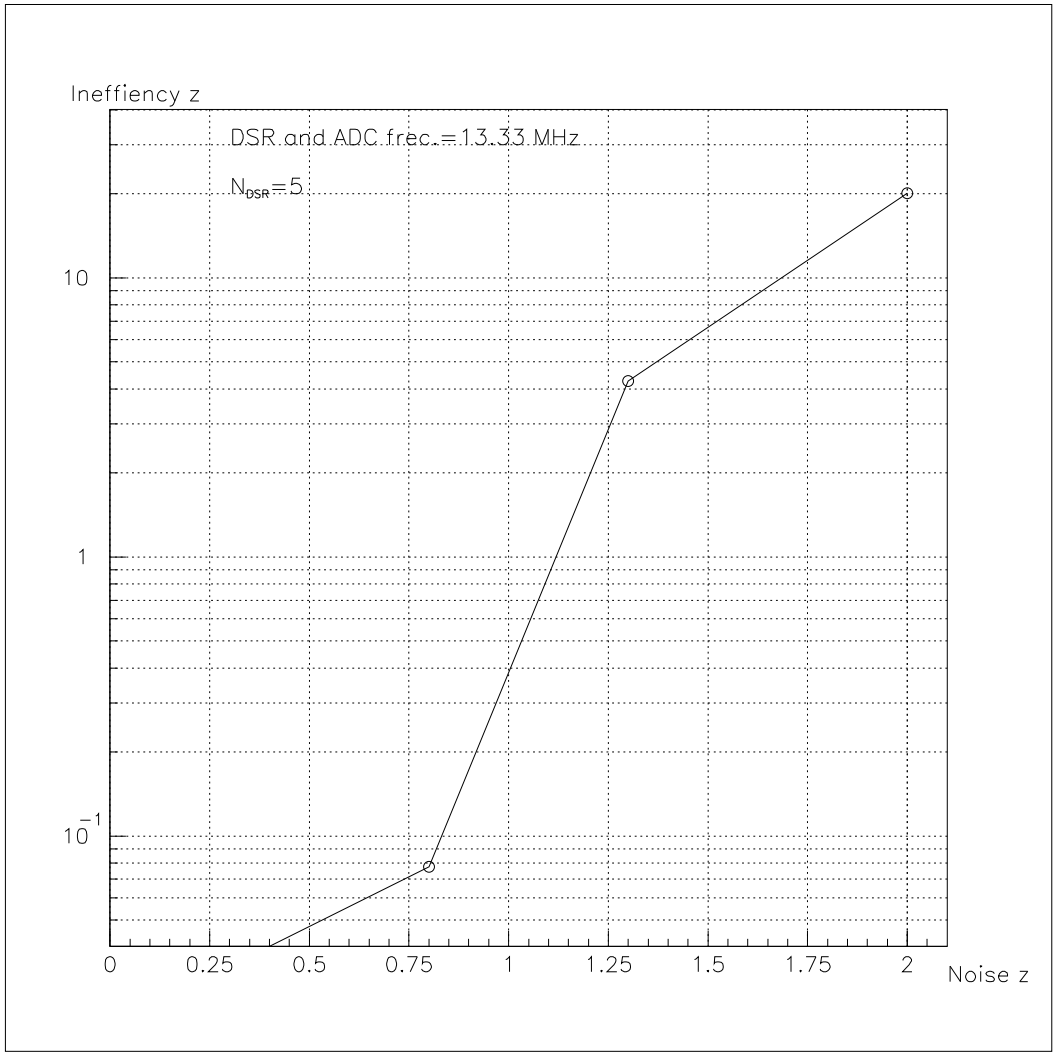
Figur 5.20: *Simulert sannsynlighet for å miste en hendelse i DSR bufferet som funksjon av dybden på DSR bufferet med program som behandler DM som alternativ 1. Her med* $t_{DSR} = 125ns$.



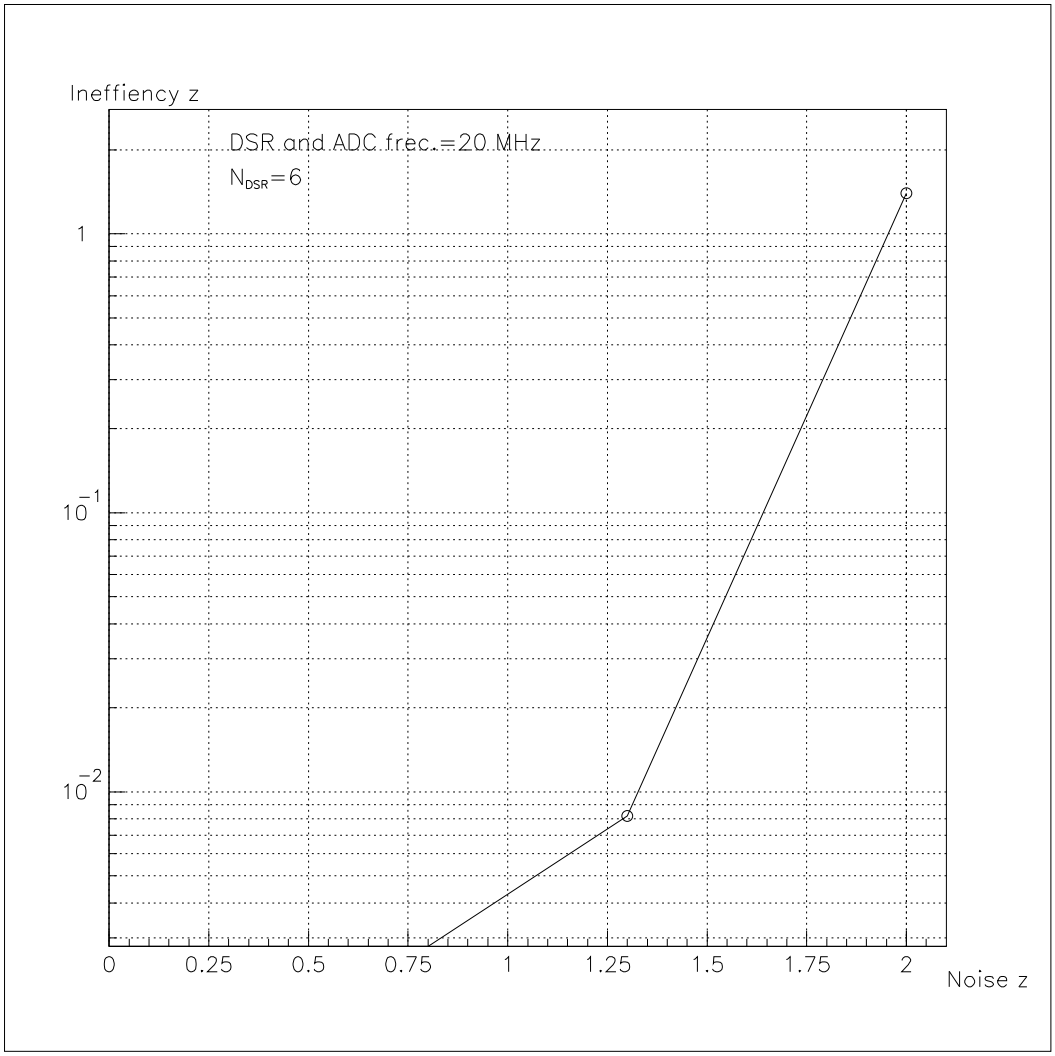
Figur 5.21: *Simulert sannsynlighet for å miste en hendelse i DSR bufferet som funksjon av dybden på DSR bufferet med program som behandler DM som alternativ 1. Her med* $t_{DSR} = 150ns$.



Figur 5.22: *Simulert sannsynlighet for å miste en hendelse i DSR bufferet som funksjon av støyen med program som behandler DM som alternativ 1. Her med* $N_{DSR} = 4$ *og* $t_{DSR} = 100ns$.



Figur 5.23: *Simulert sannsynelighet for å miste en hendelse i DSR bufferet som funksjon av støyen med program som behandler DM som alternativ 1. Her med* $N_{DSR} = 5$ *og* $t_{DSR} = 75ns$.



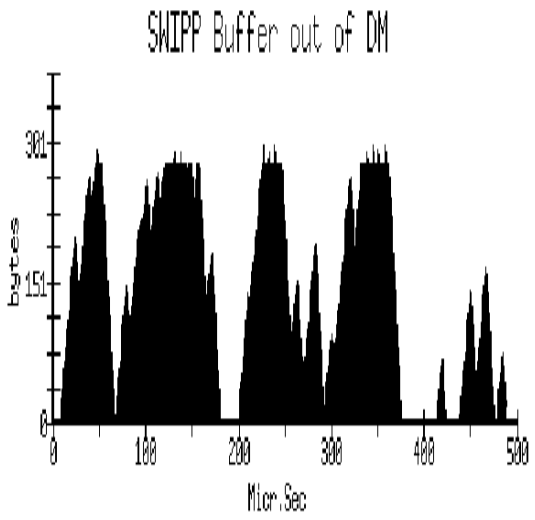
Figur 5.24: *Simulert sannsynelighet for å miste en hendelse i DSR bufferet som funksjon av støyen med program som behandler DM som alternativ 1. Her med* $N_{DSR} = 6$ *og* $t_{DSR} = 50ns$.

Tabell 5.6 viser tapet i utbufferet ved forskjellige parametere.

Noise % / Out speed		Buffersize N_{out}	0.0%	0.15%	0.4%	0.8%	1.3%	2.0%
$t_{DSR} = 100ns$	20 Mbyte/s	300byte	0.0	0.0	0.0	0.0	0.0	0.0
		500byte	0.0	0.0	0.0	0.0	0.0	0.0
	40 Mbyte/s	300byte	0.0	0.0	0.0	0.0	0.0	0.0
		500byte	0.0	0.0	0.0	0.0	0.0	0.0
$t_{DSR} = 75ns$	20 Mbyte/s	300byte	0.0	0.0	0.002	0.7	1.8	1.3
		500byte	0.0	0.0	0.0	0.1	0.9	0.9
	40 Mbyte/s	300byte	0.0	0.0	0.0	0.0	0.0	0.0
		500byte	0.0	0.0	0.0	0.0	0.0	0.0
$t_{DSR} = 50ns$	20 Mbyte/s	300byte	0.0	0.0	0.03	4.3	14.4	27.1
		500byte	0.0	0.0	0.0	1.1	9.9	24.7
	40 Mbyte/s	300byte	0.0	0.0	0.0	0.0	0.0	0.0
		500byte	0.0	0.0	0.0	0.0	0.0	0.0

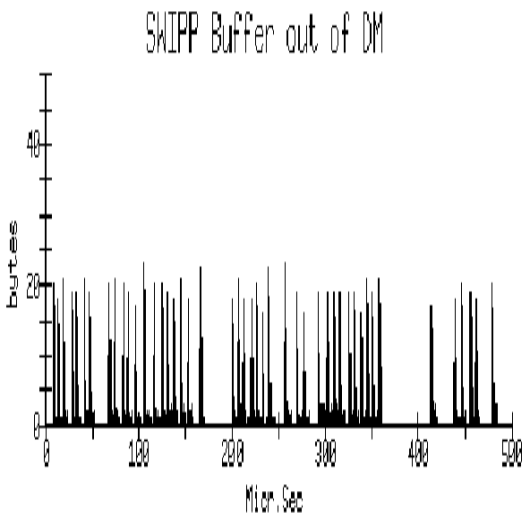
Tabell 5.6: *Simulert tap i % av data i utbufferet ved forskjellige parametere med program som behandler DM som alternativ 1.*

Figur 5.25 og 5.26 viser antallet byte i utbufferet ved forskjellige parametere [4]. En kan i figur 5.25 se at en i perioder har overflyt i utbufferet. I figur 5.26 ser en at utbufferet blir tømt kontinuerlig uten noen opphopning.



Figur 5.25: *Simulert antall byte i utbufferet med program som behandler DM som alternativ 1. Her med $N_{DSR} = 6$, $t_{DSR} = 50ns$, støy= 1.3% og utlesningshastighet fra DM på 20 Mbyte/s.*

[4]Fra samme simulering som figur 5.14 og 5.15



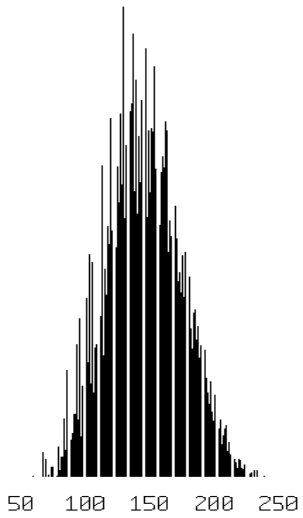
Figur 5.26: *Simulert antall byte i utbufferet med program som behandler DM som alternativ 1. Her med* $N_{DSR} = 6$, $t_{DSR} = 50ns$, *støy*= 1.3% *og utlesningshastighet fra DM på 40 Mbyte/s.*

Figur 5.27 og 5.28 viser fordelingen av størrelsen på datapakkene ut av DM. Ved å sammenligne disse med figur 5.3 og 5.4 respektivt ser en at det er en reduksjon i hendelser med mange byte[5]. Figur 5.27 har et tap i utbufferet på 0.67% og figur 5.28 har et tap i utbufferet på 27.08%. En kan av dette se at **en ineffektivitet etter ADC fører til et tap av hendelser med store datamengder**.

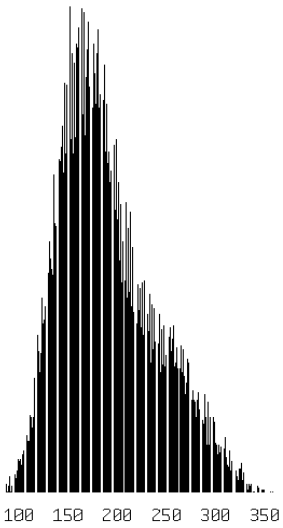
Figur 5.29, 5.30 og 5.31 viser gjennomsnittelige forsinkelse en hendelse bruker på å bli lest ut av DM fra en trigger blir gitt til hele hendelsen er ute av DM. Merkene viser maksimal og minimal tidene som er funnet fra simuleringene.

Figur 5.32 viser gjennomsnittelige forsinkelse en hendelse bruker på å bli lest ut av DM fra en trigger blir gitt til hele hendelsen er ute av DM når utlesningshastigheten økes til 40Mbyte/s.

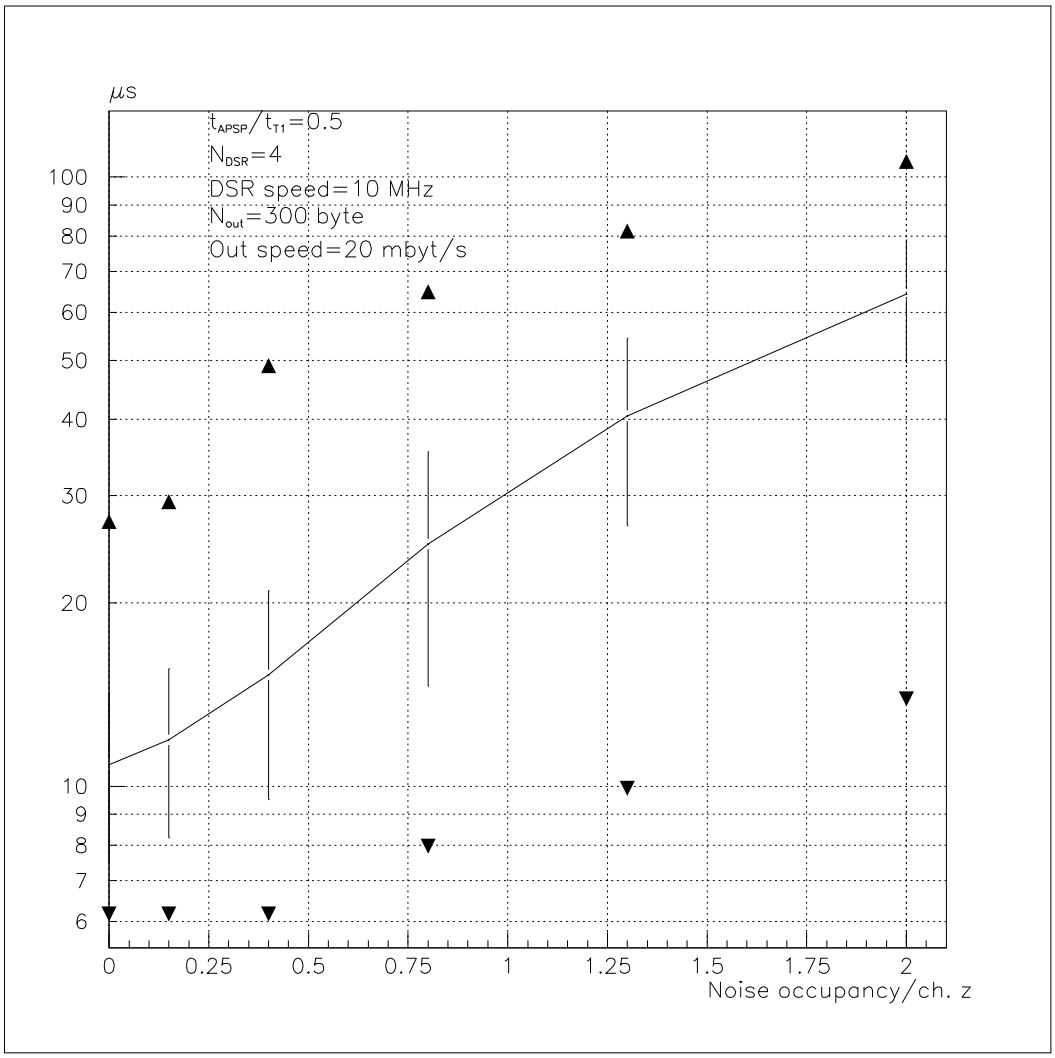
[5]Figur 5.3 og 5.27 har tilsynelatende lik fordeling, men midlet er noe forskjøvet mot venstre for 5.27



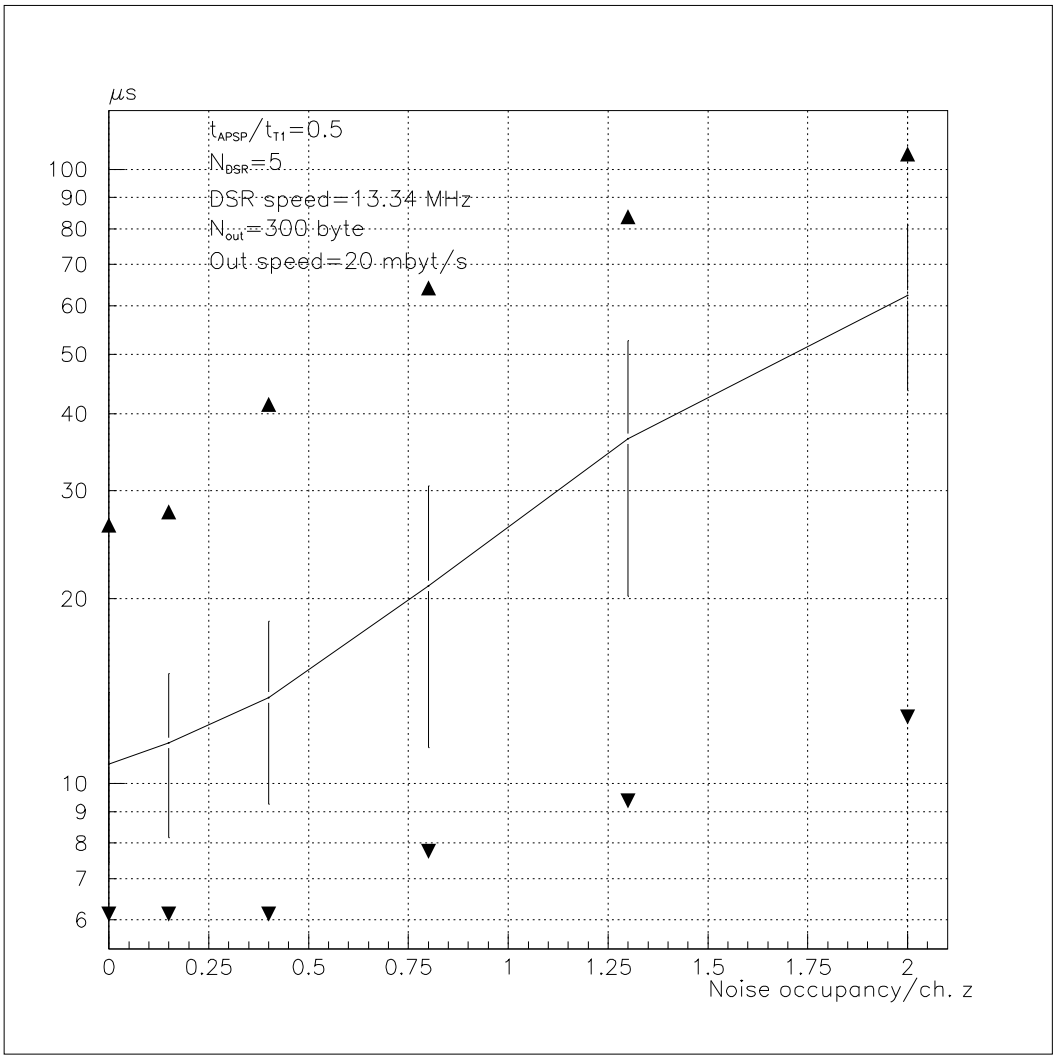
Figur 5.27: *Simulert fordeling av størrelsen på datapakkene i byte ut av DM ved bruk av program som genererer fysikken selv. Her med støy*= 0.8%, $N_{DSR} = 5$, $t_{DSR} = 75ns$, $N_{out} = 300$ *og en utlesningshastighet på 20 Mbyte/s. Midlet er her* 146.52 *byte/pakke*



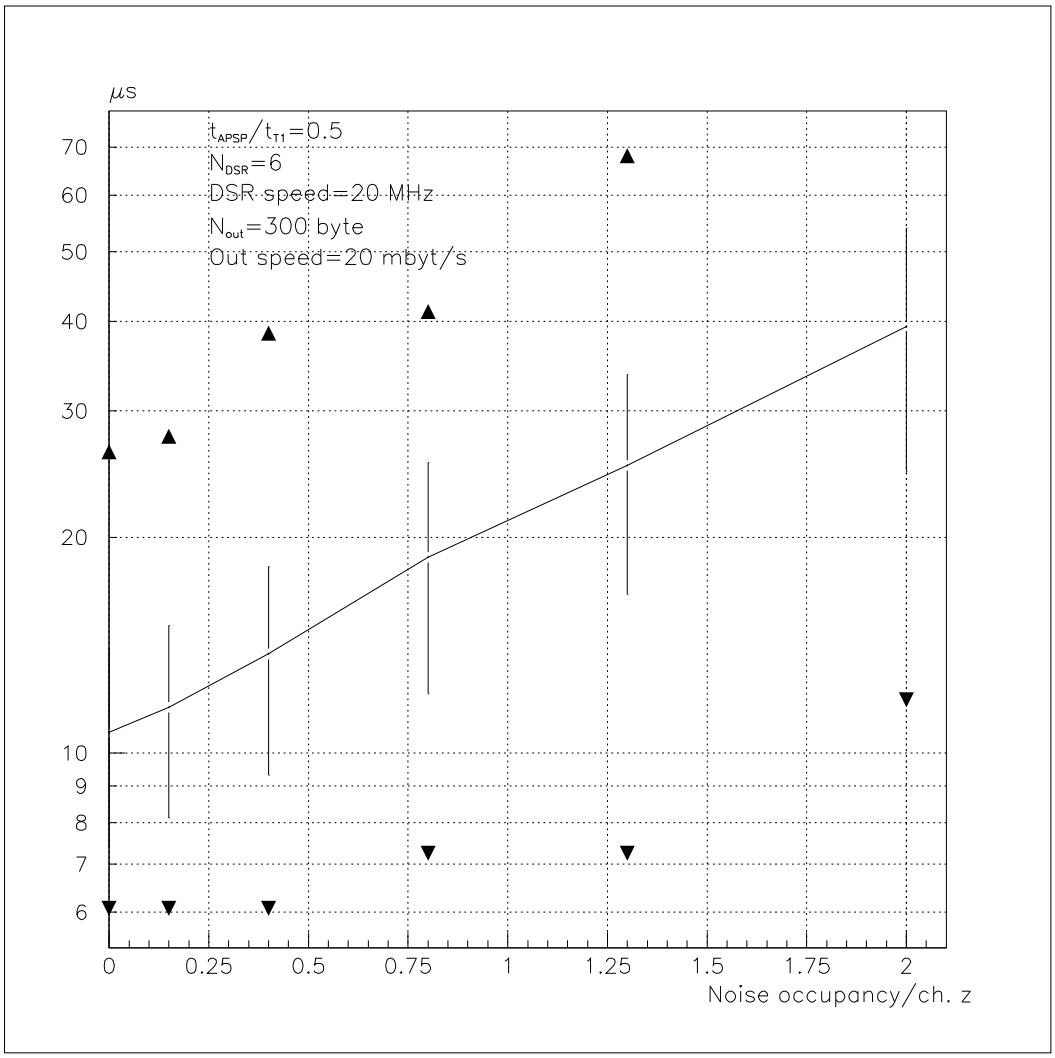
Figur 5.28: *Simulert fordeling av størrelsen på datapakkene i byte ut av DM ved bruk av program som genererer fysikken selv. Her med støy*= 2.0%, $N_{DSR} = 6$, $t_{DSR} = 50ns$, $N_{out} = 300$ *og en utlesningshastighet på 20 Mbyte/s. Midlet er her* 252.34 *byte/pakke*



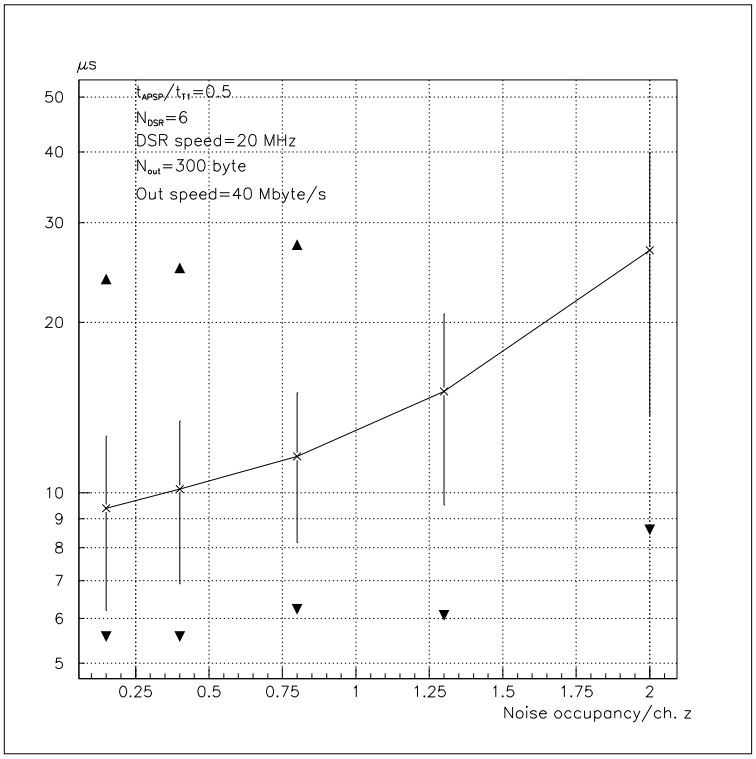
Figur 5.29: *Simulert forsinkelse fra en trigger gis til hele hendelsen er lest ut av DM som funksjon av støyen med program som behandler DM som alternativ 1. Her med* $N_{DSR} = 4$ *og* $t_{DSR} = 100ns$*. Strekene tilsvarer* $\pm 1\sigma$ *og merkene viser den maksimale og minimale verdien som simuleringen har gitt.*



Figur 5.30: *Simulert forsinkelse fra en trigger gis til hele hendelsen er lest ut av DM som funksjon av støyen med program som behandler DM som alternativ 1. Her med* $N_{DSR} = 5$ *og* $t_{DSR} = 75ns$. *Strekene tilsvarer* $\pm 1\sigma$ *og merkene viser den maksimale og minimale verdien som simuleringen har gitt.*

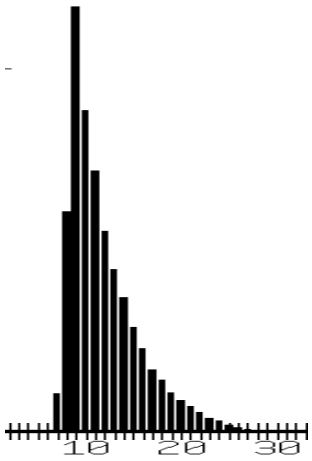


Figur 5.31: *Simulert forsinkelse fra en trigger gis til hele hendelsen er lest ut av DM som funksjon av støyen med program som behandler DM som alternativ 1. Her med* $N_{DSR} = 6$ *og* $t_{DSR} = 50ns$. *Strekene tilsvarer* $\pm 1\sigma$ *og merkene viser den maksimale og minimale verdien som simuleringen har gitt.*

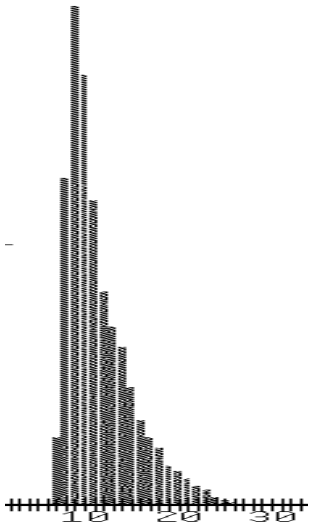


Figur 5.32: *Simulert forsinkelse fra en trigger gis til hele hendelsen er lest ut av DM som funksjon av støyen med program som behandler DM som alternativ 1. Her med* $N_{DSR} = 6$ $t_{DSR} = 50ns$ *og en utlesningshastighet fra utbufferet på 40 Mbyte/s.*

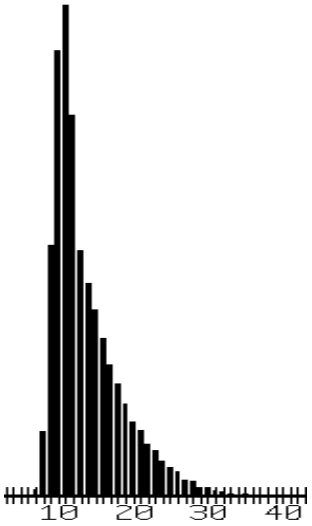
Figur 5.33 - 5.40 viser fordelingen av hvor lang tid en hendelse bruker på å bli lest ut av DM fra en trigger blir gitt til hele hendelsen er ute av DM ved forskjellige parametere.



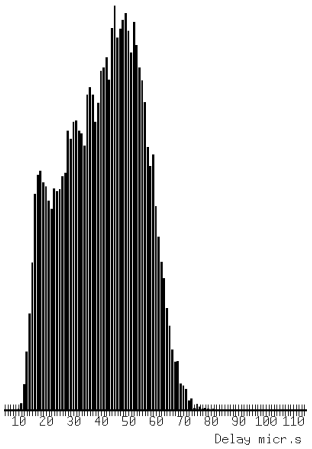
Figur 5.33: *Simulert fordeling av forsinkelsen en hendelse bruker før den er lest ut av DM med program som behandler DM som alternativ 1. Her med støy*= 0.15%, $N_{DSR} = 4$, $t_{DSR} = 100ns$. *Midlet er her* $11.91\mu s$.



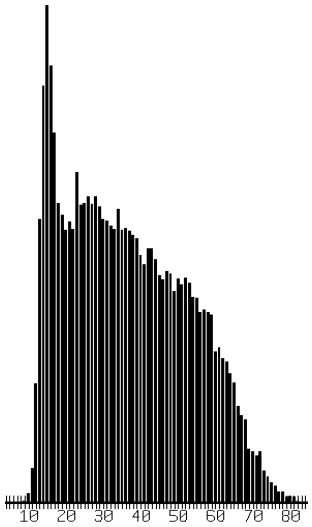
Figur 5.34: *Simulert fordeling av forsinkelsen en hendelse bruker før den er lest ut av DM med program som behandler DM som alternativ 1. Her med støy*= 0.15%, $N_{DSR} = 6$, $t_{DSR} = 50ns$. *Midlet er her* $11.59\mu s$.



Figur 5.35: *Simulert fordeling av forsinkelsen en hendelse bruker før den er lest ut av DM med program som behandler DM som alternativ 1. Her med støy*= 0.4%, $N_{DSR} = 5$, $t_{DSR} = 75ns$. *Midlet er her* $13.81\mu s$.

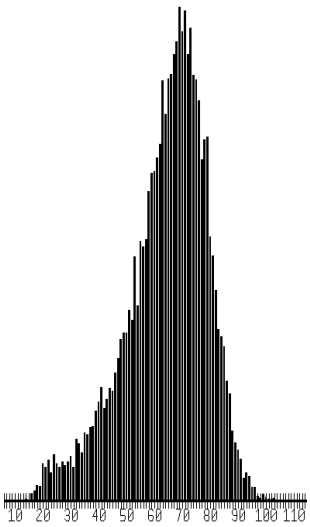


Figur 5.36: *Simulertfordeling av forsinkelsen en hendelse bruker før den er lest ut av DM med program som behandler DM som alternativ 1. Her med støy*= 1.3%, $N_{DSR} = 4$, $t_{DSR} = 100ns$. *Midlet er her* $40.56\mu s$.

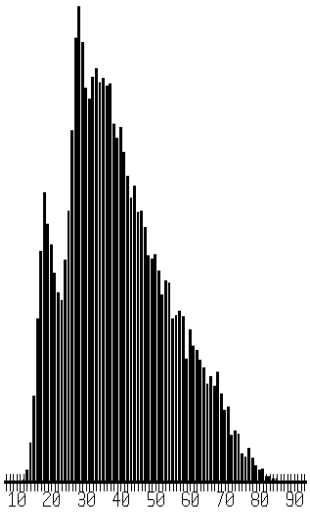


Figur 5.37: *Simulert fordeling av forsinkelsen en hendelse bruker før den er lest ut av DM med program som behandler DM som alternativ 1. Her med støy*= 1.3%, $N_{DSR} = 5$, $t_{DSR} = 75ns$. *Midlet er her* $36.40\mu s$.

Figur 5.38: *Simulert fordeling av forsinkelsen en hendelse bruker før den er lest ut av DM med program som behandler DM som alternativ 1. Her med støy*= 1.3%, $N_{DSR} = 5$, $t_{DSR} = 75ns$ *og utlesningshastigheten fra utbufferet er 40 Mbyte/s. Midlet er her* $29.23\mu s$.

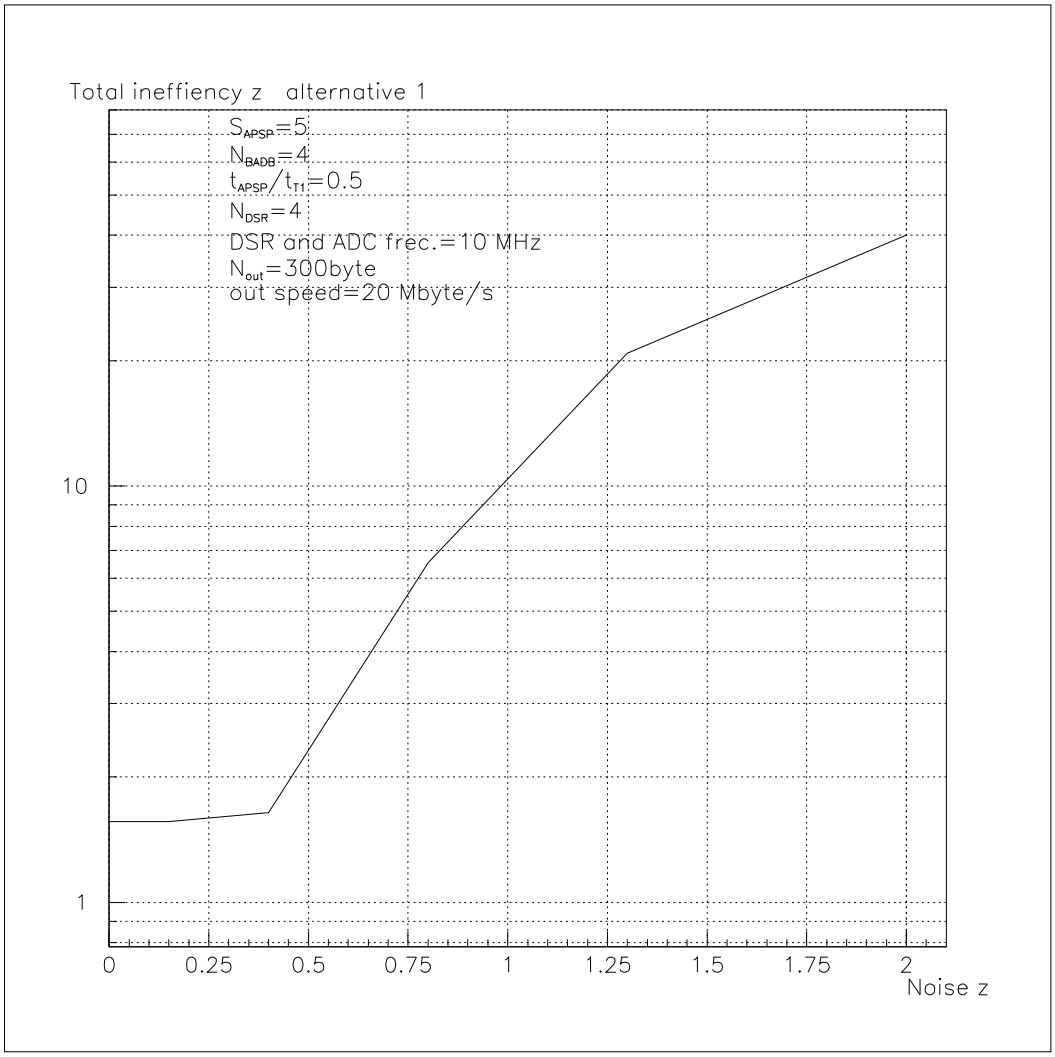


Figur 5.39: *Simulert fordeling av forsinkelsen en hendelse bruker før den er lest ut av DM med program som behandler DM som alternativ 1. Her med støy*= 2%, $N_{DSR} = 4$, $t_{DSR} = 100ns$. *Midlet er her* $64.2\mu s$.



Figur 5.40: *Simulert fordeling av forsinkelsen en hendelse bruker før den er lest ut av DM med program som behandler DM som alternativ 1. Her med støy*= 2%, $N_{DSR} = 6$, $t_{DSR} = 50ns$. *Midlet er her* $39.31\mu s$.

Figur 5.41 viser den totale ineffektivitet for DM inkludert ineffektivitet pga. triggere som kommer for tett og ineffektivitet i ADB[6].



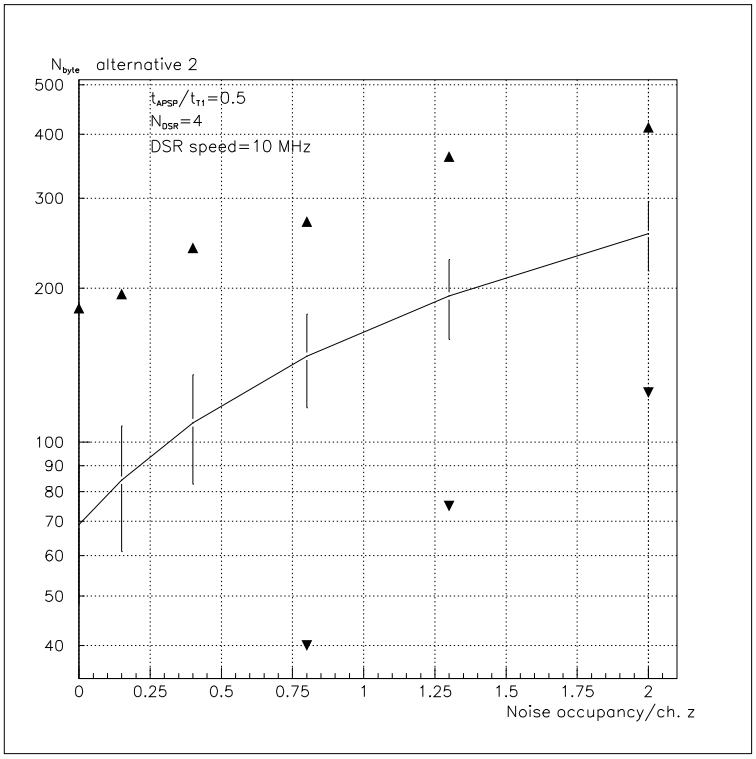
Figur 5.41: *Simulert total ineffektivitet for DM som funksjon av støyen med program som behandler DM som alternativ 1. Her med* $S_{APSP} = 5$, $N_{ADB} = 4$, $\frac{t_{APSP}}{t_{T1}} = 0.5$, $N_{DSR} = 4$ *og* $t_{DSR} = 100ns$.

[6]Pga. den lave DSR hastigheten er ineffektivitet i utbufferet≈ 0.0

Alternativ 2.

Alternativ 2 er simulert som beskrevet i appendix D og håndterer DM som beskrevet i alternativ 2 ovenfor. APSP starter ikke å prosessere en hendelse før det er ledig plass i DSR bufferet.

Figur 5.42 viser den gjennomsnittlige størrelsen på datapakkene ut av DM som funksjon av støy ved forskjellige DSR parametere. Parameterene til DM er justert slik at det ikke er tap av data etter ADC. Ved å sammenligne figur 5.8 og 5.42 , eller eventuelt 5.43 og figur 5.4, kan en se at disse er like. Dette viser indirekte at det ikke er noen sammenheng mellom tap av data og størelsen på hendelsen frem til og med ADC[7].



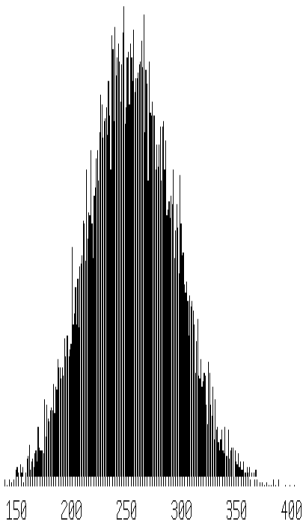
Figur 5.42: *Simulert gjennomsnittlig størelse på datapakkene ut av DM som funksjon av støy med program som behandler DM som alternativ 2. Her med* $N_{DSR} = 4$ *og* $t_{DSR} = 100ns$*. Strekene tilsvarer* $\pm 1\sigma$ *og merkene viser den maksimale og minimale verdien som simuleringen har gitt.*

En kan ikke se noen vesentlig forskjell på strømmen av data ut av DM ved forskjellige DSR parametere ved sammenligning av alternativ 1 og 2. Den gjennomsnittlige raten av data ut av DM vil heller ikke variere fra alternativ 1 når en ser bort fra et evt. tap i DSR bufferet.

Figur 5.44 og 5.45 viser fordelingen av hvor mange hendelser som er i DSR bufferet ved forskjellig støy. En kan se at det er først ved høy støy en har en vesentlig forskjell fra fordelingen fra alternativ 1.

Figur 5.46 og 5.47 viser tiden DSR bufferet er fullt som funksjon av støyen ved forskjellige DSR parametere. For å kunne danne seg et bilde av ineffektiviteten til DSR kan en studere disse

[7]Figur 5.42 har betydelig tap av data ved stor støy.



Figur 5.43: *Simulert fordeling av størrelsen på datapakkene i byte ut av DM ved bruk av program som genererer fysikken selv og behandler DM som alternativ 2. Her med støy*= 2.0%, $N_{DSR} = 4$ *og* $t_{DSR} = 100ns$. *Midlet er her* 255.6 *byte/pakke.*

grafene[8].

Ved å sammenligne figurene som viser antallet byte i utbufferet med alternativ 1 kan en ikke se noen vesentlig forskjell på hvor mange byte det er i utbufferet under simuleringene.

Tabell 5.7 viser tapet i utbufferet ved forskjellige parametere. En kan se at det ikke er noen vesentlig forskjell fra alternativ 1.

Noise % ——— Out speed	N_{out}	0.0%	0.15%	0.4%	0.8%	1.3%	2.0%
$t_{DSR} = 100ns$ 20 Mbyte/s	300byte	0.0	0.0	0.0	0.0	0.0	0.0
$t_{DSR} = 50ns$ 20 Mbyte/s	300byte	0.0	0.0	0.3	4.3	14.4	27.5

Tabell 5.7: *Simulert tap i % av data i utbufferet ved forskjellige DSR parametere og støy med program som behandler DM som alternativ 2.*

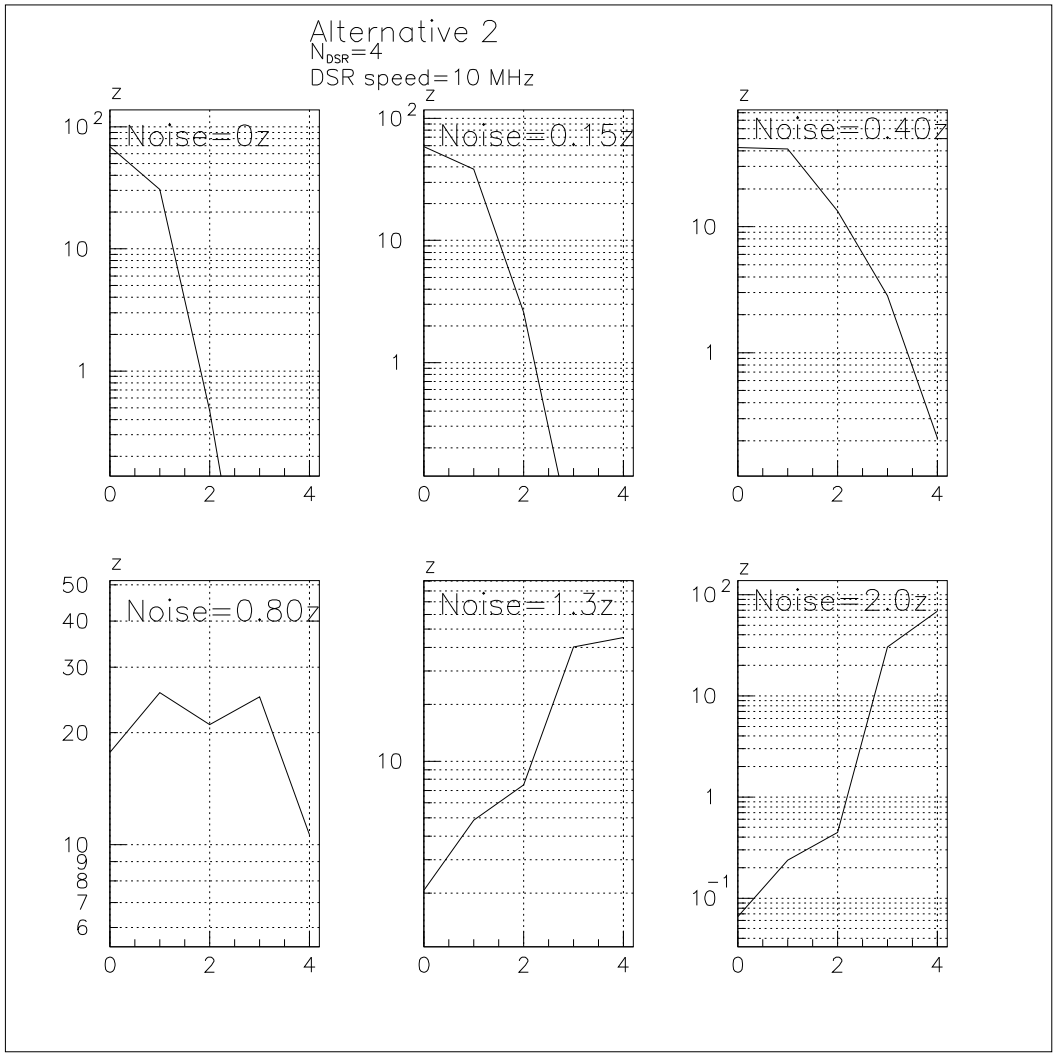
Figur 5.48 viser gjennomsnittlige forsinkelse en hendelse bruker på å bli lest ut av DM fra en trigger blir gitt til hele hendelsen er ute av DM. En kan også se maksimal og minimal tidene som er funnet fra simuleringene. En kan se at en har en betydelig økning i prosesseringstiden for en hendelse ved økende støy i forhold til alternativ 1.

Figur 5.49 viser et eksempel på fordelingen av hvor lang tid en hendelse bruker på å bli lest ut av DM fra en trigger blir gitt til hele hendelsen er ute av DM.

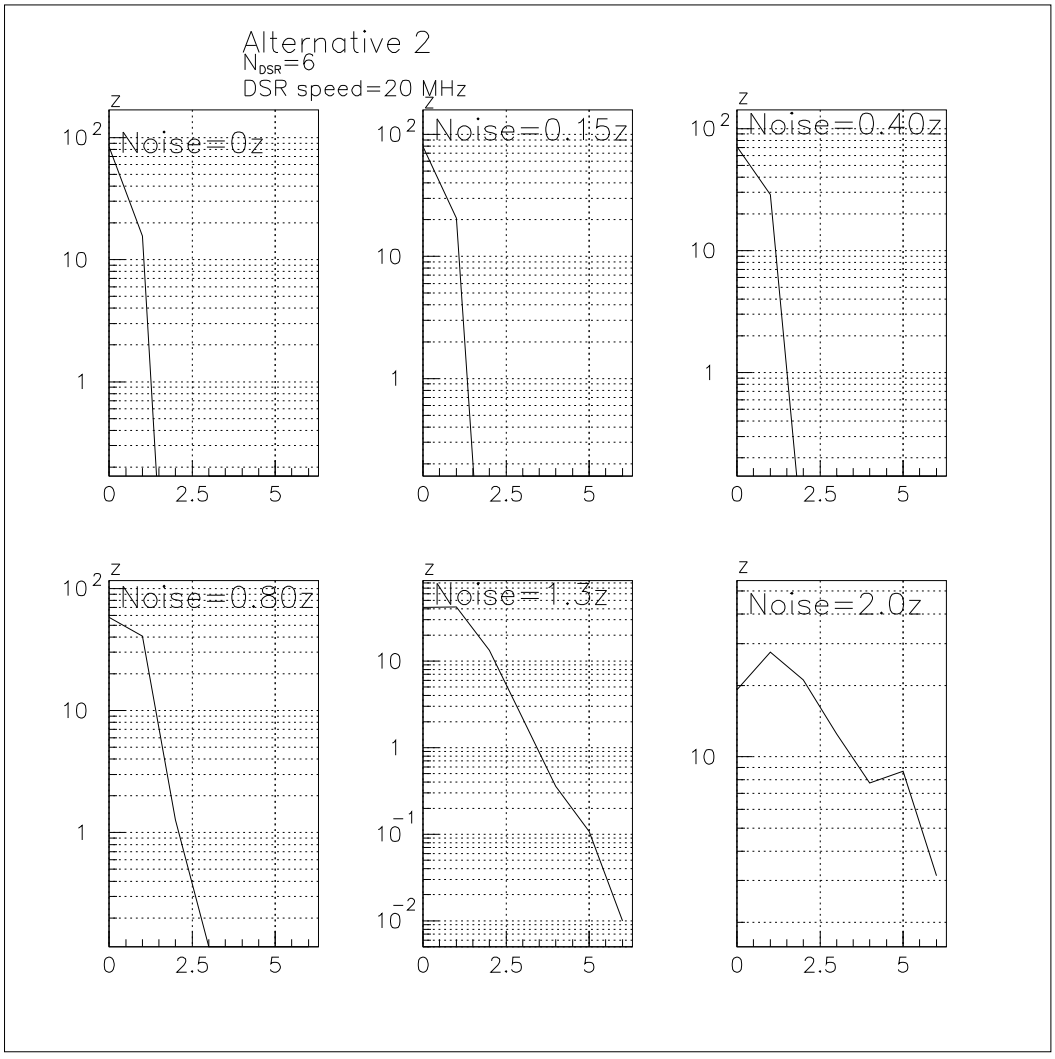
Figur 5.50 viser den totale ineffektivitet for DM inkludert ineffektivitet pga. triggere som kommer for tett og ineffektivitet i ADB[9]. En kan se at en har en reduksjon i ineffektivitet ,

[8]Dette er noe ukritisk og bør ikke brukes ved sammenligning med andre alternativers ineffektivitet.

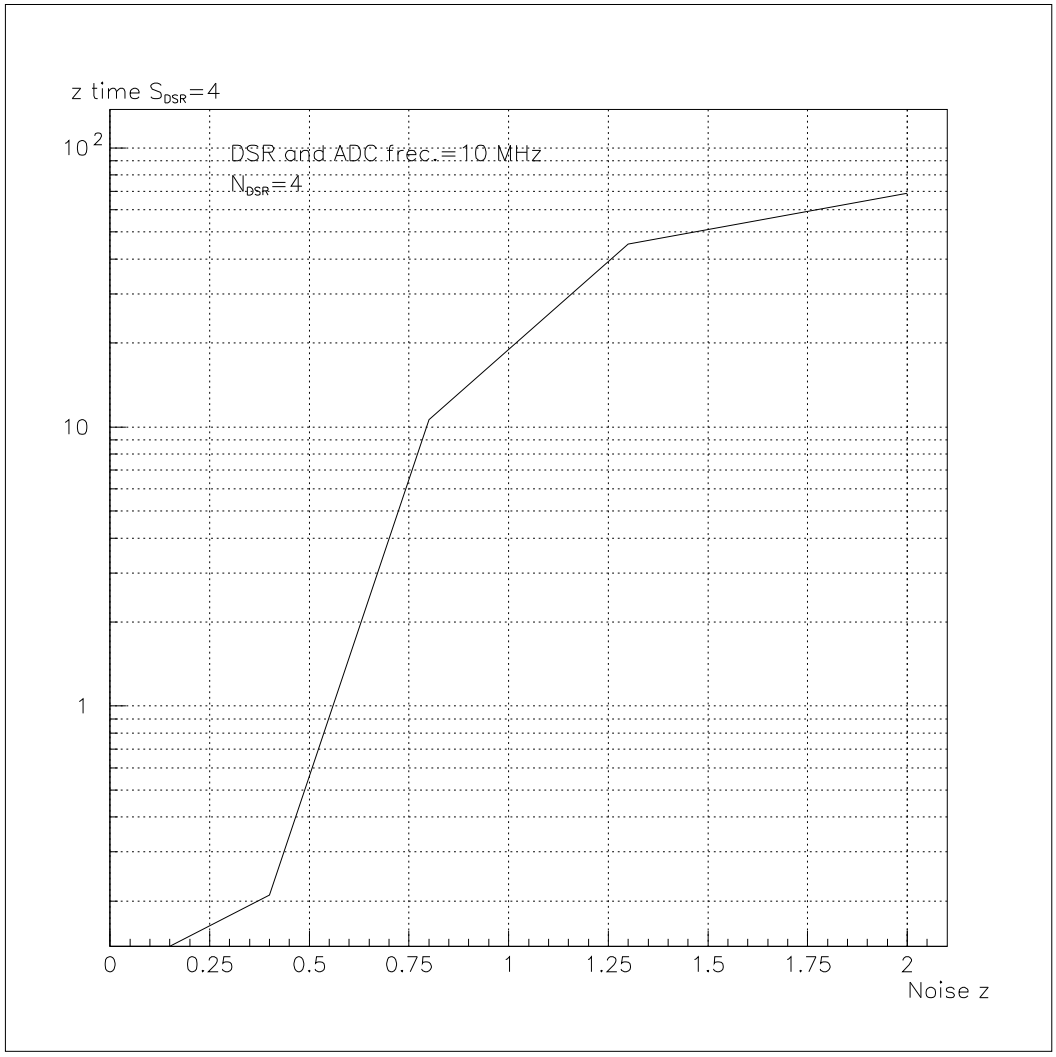
[9]Pga. den lave DSR hastigheten er ineffektivitet i utbufferet nær null.



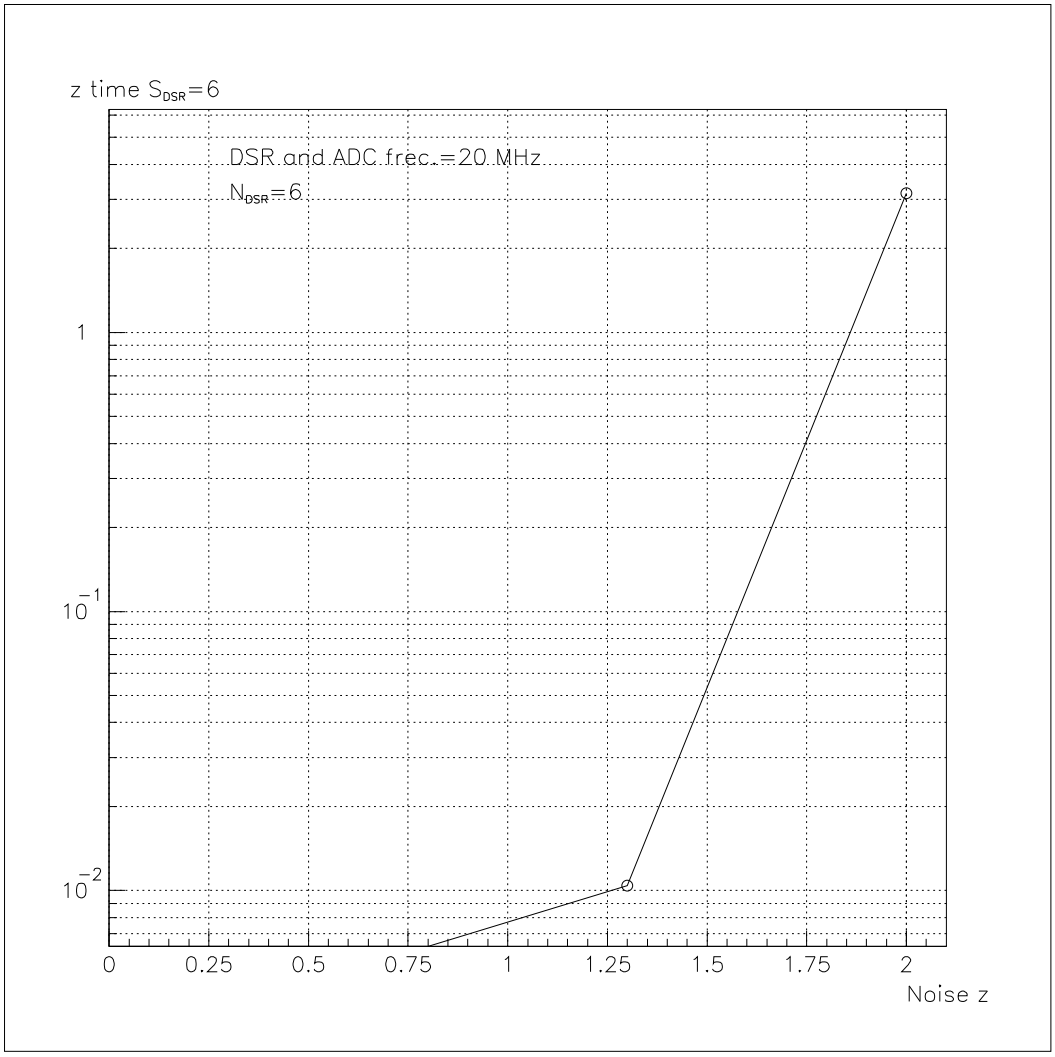
Figur 5.44: *Simulert sannsynlighetsfordeling av hvor mange hendelser som er i DSR bufferet med program som behandler DM som alternativ 2. Her med* $N_{DSR} = 4$, $t_{DSR} = 100ns$ *og forskjellig støy. Abscissene viser hvor mange hendelser det er i bufferene.*



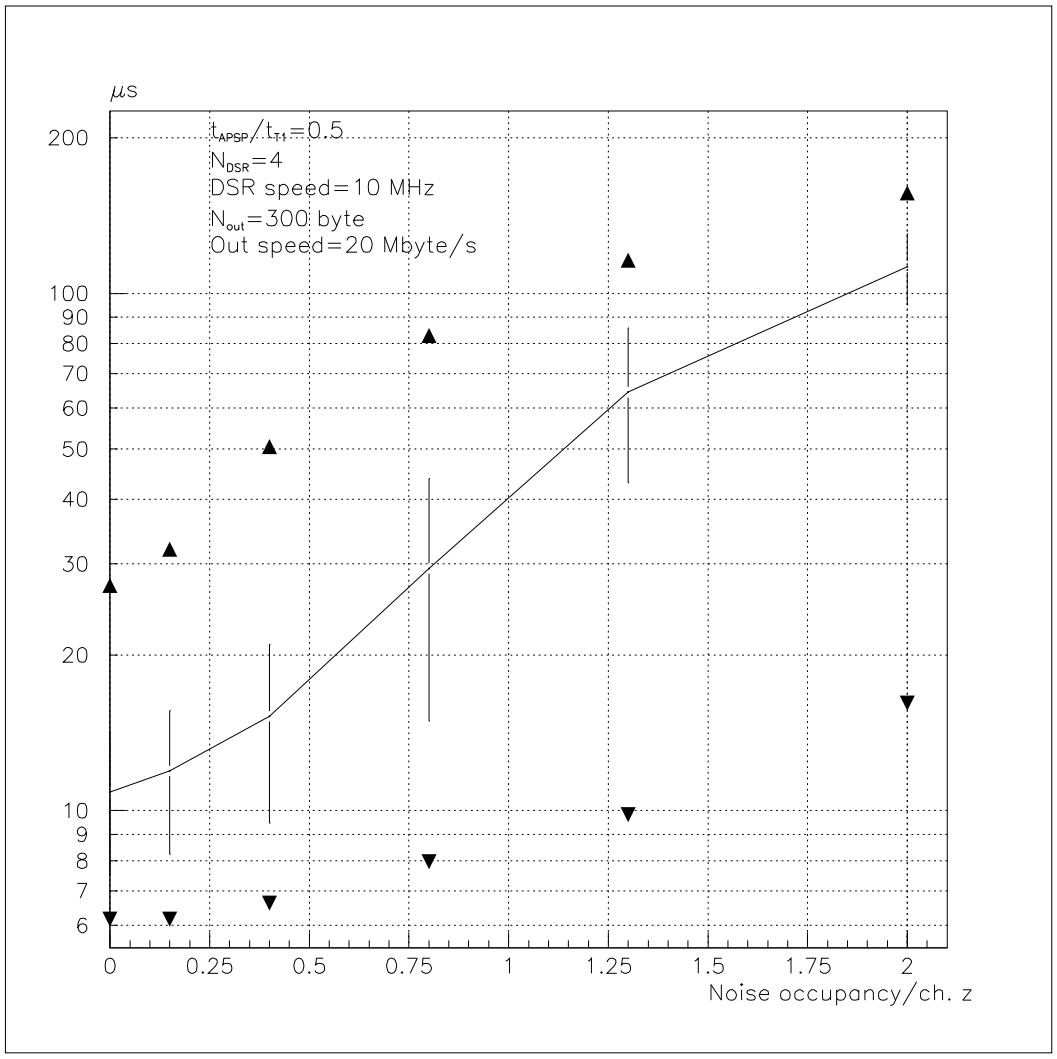
Figur 5.45: *Simulert sannsynlighetsfordeling av hvor mange hendelser som er i DSR bufferet med program som behandler DM som alternativ 2. Her med* $N_{DSR} = 6$, $t_{DSR} = 50ns$ *og forskjellig støy. Abscissene viser hvor mange hendelser det er i bufferene.*



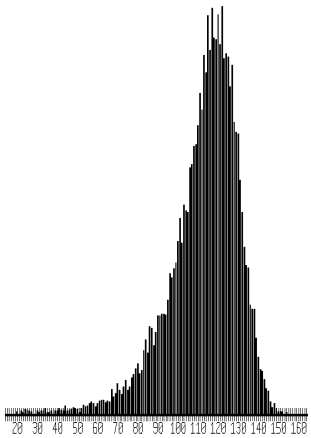
Figur 5.46: *Simulert tid DSR bufferet er fullt som funksjon av støyen med program som behandler DM som alternativ 2. Her med* $N_{DSR} = 4$ *og* $t_{DSR} = 100ns$.



Figur 5.47: *Simulert tid DSR bufferet er fullt som funksjon av støyen med program som behandler DM som alternativ 2. Her med* $N_{DSR} = 6$ *og* $t_{DSR} = 50ns$.

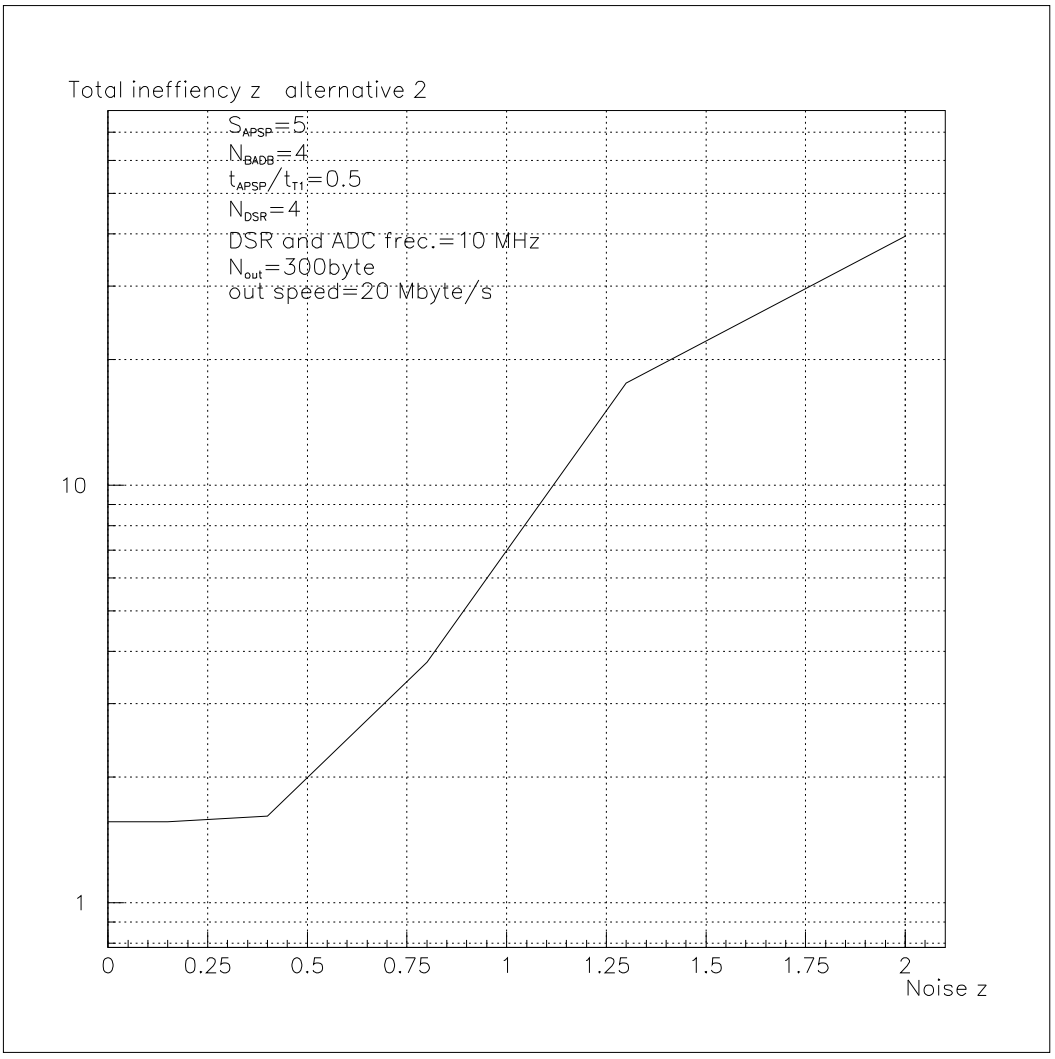


Figur 5.48: *Simulert forsinkelse fra en trigger gis til hele hendelsen er lest ut av DM som funksjon av støyen med program som behandler DM som alternativ 2. Her med* $N_{DSR} = 4$ *og* $t_{DSR} = 100ns$. *Strekene tilsvarer* $\pm 1\sigma$ *og merkene viser den maksimale og minimale verdien som simuleringen har gitt.*



Figur 5.49: *Simulertfordeling av forsinkelsen en hendelse bruker før den er lest ut av DM med program som behandler DM som alternativ 2. Her med støy*= 2.0%, $N_{DSR} = 4$, $t_{DSR} = 100ns$. *Midlet er her* $112.7\mu s$.

sammenlignet med alternativ 1, fra $\approx 0.5\%$ til $\approx 1.5\%$ støy. Grunnen til dette er at det er først fra $\approx 0.5\%$ støy ineffektivitet i DSR bufferet gjør seg gjeldende og ved stor støy er ineffektiviteten til DSR for de to alternativene utjevnet.



Figur 5.50: *Simulert total ineffektivitet for DM som funksjon av støyen med program som behandler DM som alternativ 2. Her med* $S_{APSP} = 5$, $N_{ADB} = 4$, $\frac{t_{APSP}}{t_{T1}} = 0.5$, $N_{DSR} = 4$ *og* $t_{DSR} = 100ns$.

Alternativ 3.

I dette tilfellet vil tap av data i MC forskyves frem til DSR utlesningen som videre forskyver sitt tap til ADB. Den totale ineffektivitet i MC vil være av en mere kompleks type enn den en får direkte av prosesstidene for de primære funksjonene til MC[10]. En vil f.eks. ha ineffektivitet pga. funksjonsfeil, kalibrering osv. dersom dette gjøres under data-taking. Det vil derfor være svært vanskelig å kunne finne denne ineffektiviteten før en vet hvordan MC vil bli konstruert og fungere.

Ovenfor er det vist noen resultater for ineffektivitet til utbufferet. En må kreve at denne skal være svært liten og derfor ha liten innvirkning på flyten av data foran utbufferet.

Om en krever at en ikke mister data i utbufferet kan en derfor simulere dette alternativet som alternativ 2 og dermed oppnå de samme resultatene.

[10]Med primære funksjonene til MC menes her å videreformidle verdiene fra ADC ut av DM.

Alternativ 4.

Alternativ 4 er simulert som beskrevet i vedlegg D. Alternativ 4 er tatt med for å få en forståelse for hva som skjer om prosesseringstiden til en prosess gjøres avhengig av prosessen etter. Det er her gjort ved at APSP venter med å gi ut sin ferdig prosesserte utverdi til det er ledig plass i DSR bufferet etter.

Simuleringene viser ingen korrelasjon mellom tap og størrelsen på hendelsen frem til ADC.

En kan ikke se noen vesentlig forskjell på strømmen av data ut av DM ved forskjellige DSR parametere ved sammenligning av alternativ 1 og 4. Den gjennomsnittlige raten av data ut av DM vil heller ikke variere fra alternativ 1 når en ser bort fra et evt. tap i DSR bufferet.

Figur 5.51 viser fordelingen av hvor mange hendelser som er i DSR bufferet ved forskjellig støy og DSR parametere. En kan finne at sannsynligheten for å finne $S_{DSR} < N_{DSR} - 1$ er noenlunde den samme som i alternativ 2. Ved $S_{DSR} = N_{DSR} - 1$ går den kraftig ned og ved fullt buffer går den kraftig opp. Grunnen til dette er at når DSR har fullt buffer og den er ferdig med å prosessere en hendelse vil APSP med stor sannsynlighet gi en ny hendelse straks etter.

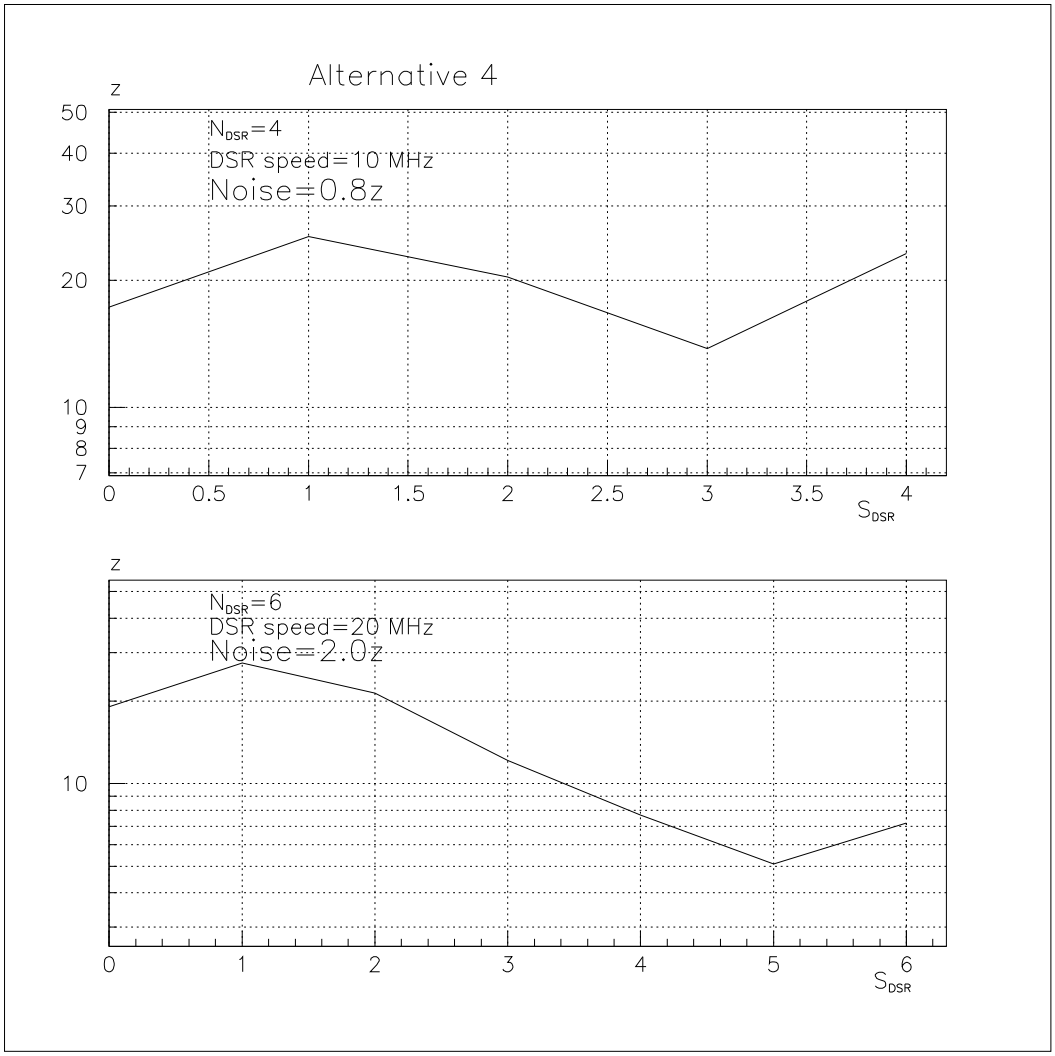
Figur 5.52 viser tiden DSR bufferet er fullt som funksjon av støyen. En kan se en kraftig økning i tiden DSR bufferet er fullt sammenlignet med alternativ 2.

Figur 5.53 viser gjennomsnittlig forsinkelse en hendelse bruker på å bli lest ut av DM fra en trigger blir gitt til hele hendelsen er ute av DM. En kan se en liten økning i prosesseringstiden ved stor støy.

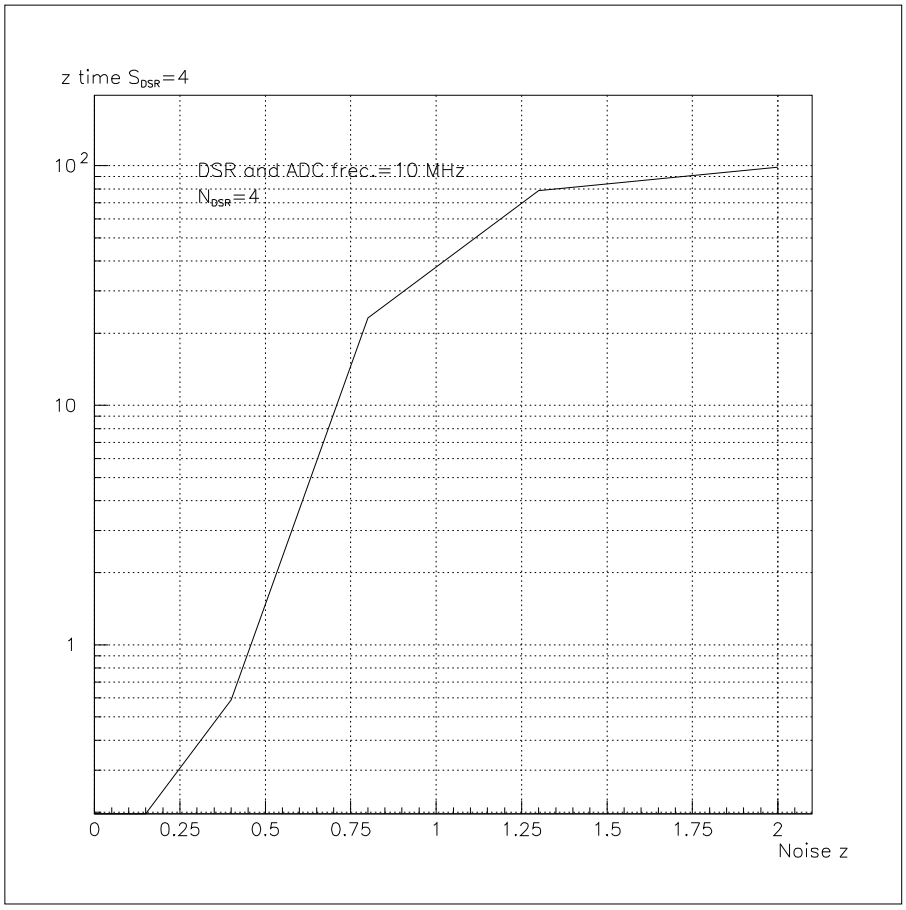
Figur 5.54 viser et eksempel på fordelingen av hvor lang tid en hendelse bruker på å bli lest ut av DM fra en trigger blir gitt til hele hendelsen er ute av DM.

Figur 5.55 viser den totale ineffektivitet for DM inkludert ineffektivitet pga. triggere som kommer for tett og ineffektivitet i ADB[11]. En kan ikke se noen vesentlig forskjell i ineffektiviteten, evt. en svak reduksjon ved 0.05% støy der ineffektivitet til DSR begynner å gjøre seg gjeldende.

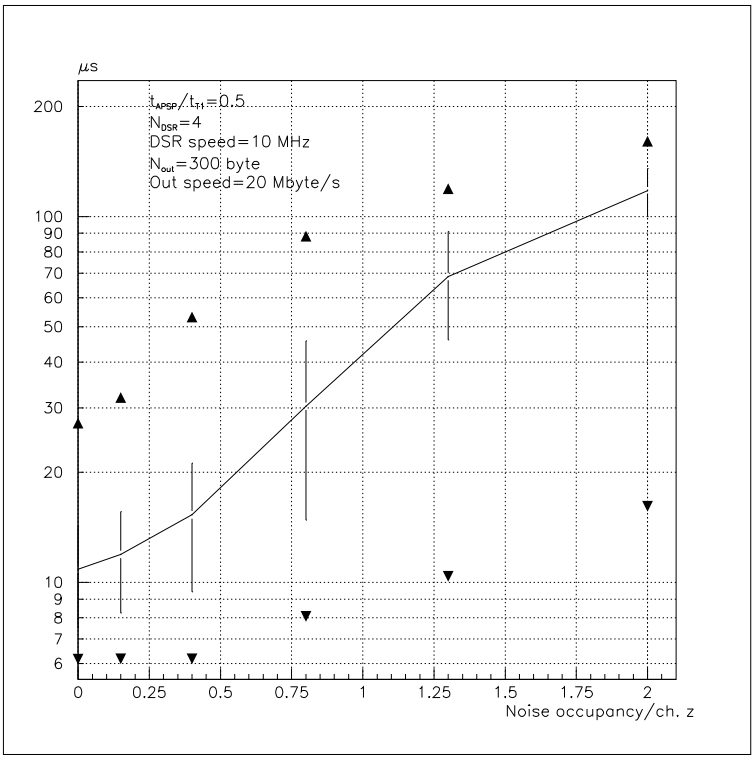
[11] Pga. den lave DSR hastigheten er ineffektivitet i utbufferet nær null.



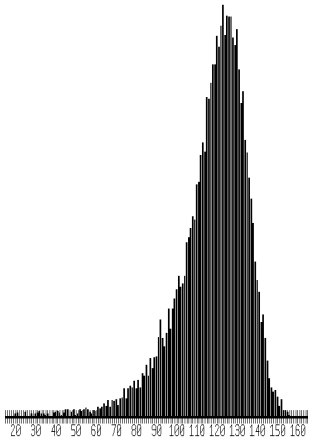
Figur 5.51: *Simulert sannsynlighetsfordelning av hvor mange hendelser som er i DSR bufferet med program som behandler DM som alternativ 4. Her med støy= 0.8% og støy= 2.0%.*



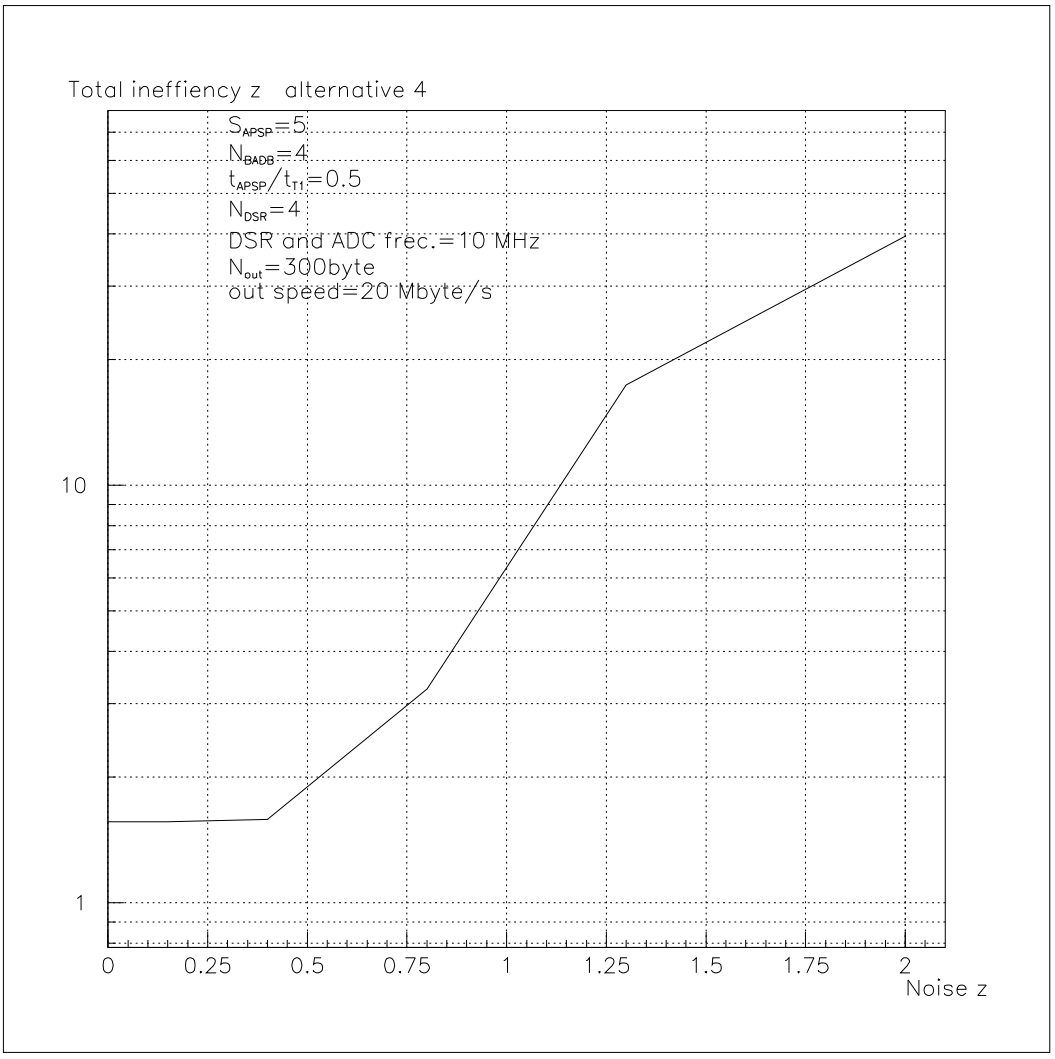
Figur 5.52: *Simulert tid DSR bufferet er fullt som funksjon av støyen med program som behandler DM som alternativ 4. Her med* $N_{DSR} = 4$ *og* $t_{DSR} = 100ns$.



Figur 5.53: *Simulert forsinkelse fra en trigger gis til hele hendelsen er lest ut av DM som funksjon av støyen med program som behandler DM som alternativ 4. Her med* $N_{DSR} = 4$ *og* $t_{DSR} = 100ns$. *Strekene tilsvarer* $\pm 1\sigma$ *og merkene viser den maksimale og minimale verdien som simuleringen har gitt.*



Figur 5.54: *Simulert fordeling av forsinkelsen en hendelse bruker før den er lest ut av DM med program som behandler DM som alternativ 4. Her med støy*= 2.0%, $N_{DSR} = 4$, $t_{DSR} = 100ns$. *Midlet er her* $117.7\mu s$.



Figur 5.55: *Simulert total ineffektivitet for DM som funksjon av støyen med program som behandler DM som alternativ 4. Her med* $S_{APSP} = 5$, $N_{BADB} = 4$, $\frac{t_{APSP}}{t_{T1}} = 0.5$, $N_{DSR} = 4$ *og* $t_{DSR} = 100ns$.

5.2.2 Ineffektivitet ved variasjon av arealet til DM.

Fra kapittel 2.4 har en antatt at ineffektivitet ε_{async} vil være konstant om tiden til prosessene varieres omvendt proporsjonalt med raten av signaler inn. For DSR vil dette være at tapet er konstant om en reduserer t_{DSR} proporsjonalt med N_{sig}. Fra simuleringer har en som tabell 5.8 viser. Resultatene viser at ε_{DSR} avviker noe fra å være konstant.

N_{sig}	Occupancy	Noise	Trigger	N_{DSR}	t_{DSR}	t_{out}	$t_{DMdelay}$	Efficiency DSR
14.40	0.225%	0.4%	400 RND	4	200 ns.	100 ns.	$26.5\mu s$	93.4%
28.88	0.45%	0.8%	400 RND	4	100 ns.	50 ns.	$25.0\mu s$	95.0%
58.03	0.90%	1.6%	400 RND	4	50 ns.	25 ns.	$24.0\mu s$	95.2%

Tabell 5.8: *Simulert forsinkelse av en hendelse og effektiviteten til DSR ved forskjellige parametere med program som behandler DM som alternativ 1 og genererer fysikken selv.*

5.2.3 Simulering ved bruk av program med fysikk-input fra en fysikk Monte Carlo simulering.

I appendix D.3 er det forklart hvordan programmet er bygd opp og bruken av det.

Monte Carlo (MC) simuleringen er gjort ved 16 minimum bias hendelser pr. BC. For å øke antallet minimum bias hendelser i MC simuleringen fra 16 til 30 er det lagt til 0.1 prosentpoeng støy i tillegg til den reelle støyen. Valget av å legge til 0.1% er gjort pga. at dette gir noenlunde samme gjennomsnittlig rate som simuleringene med programmet som genererer fysikken selv [12]. En kan derfor sammenligne resultatene fra de forskjellige måtene å generere fysikk. Differansen, 0.1%, en legger til viser at den MC genererte fysikken gir en noe høy sannsynlighet for treff for hver minimum bias hendelse. Hovedgrunnen til dette er at dataene fra MC simuleringene er hentet fra området i eller rundt *Region of Interest*[13].

Alle simuleringene her er gjort med et utlegg av DM etter alternativ 1. Figur 5.56 viser gjennomsnittlig antall signal i en DM som funksjon av støy. En kan legge merke til at antallet signal for hver hendelse har en større variasjon en ved simuleringen der programmet generer fysikken selv, figur 5.7.

Figur 5.57 viser den gjennomsnittlige størrelsen på datapakkene ut av DM som funksjon av støy. Parameterene til DM er justert slik at det ikke er tap av data etter ADC. Den maksimale og minimale størrelsen på datapakken er også vist. En kan ikke se noen forskjell på den gjennomsnittlige størrelsen på datapakkene i forhold til simuleringer gjort med program som genererer fysikken selv. Imidlertid er det en økning i variasjonen til datapakkene i forhold til program som genererer fysikken selv. Maksimal-verdiene funnet fra simuleringene har også økt kraftig i forhold til program som genererer fysikken selv. Disse variasjonene skyldes at en QCD jet-struktur i et gitt område vil gi store lokale multiplitets-fluktuasjoner.

Figur 5.9 og 5.10 viser fordelingen av pakkestørrelsen ved henholdsvis 0.0%, 0.15%, 0.8% og 2.0% støy. En kan se at en har fått en hale ut til høyre i forhold til simuleringer gjort med program som generer fysikken selv.

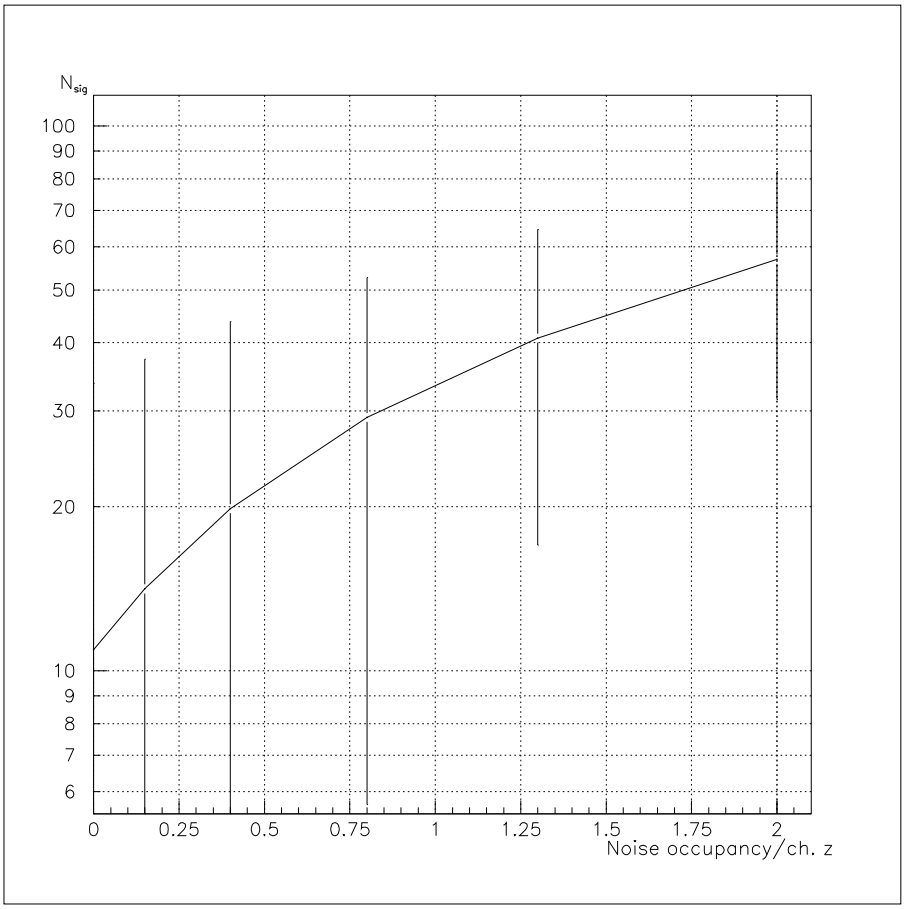
Figur 5.58 viser Mbyte/s ut DM som funksjon av støy. En kan finne at avviket i forhold til simulering gjort med program som genererer fysikken selv er under 3%.

Figur 5.59 viser et typisk eksempel på strømen av data ut av DM. En kan se en svak tendens til en større grad av opphopning sammenlignet med simulering gjort med program som genererer fysikken selv.

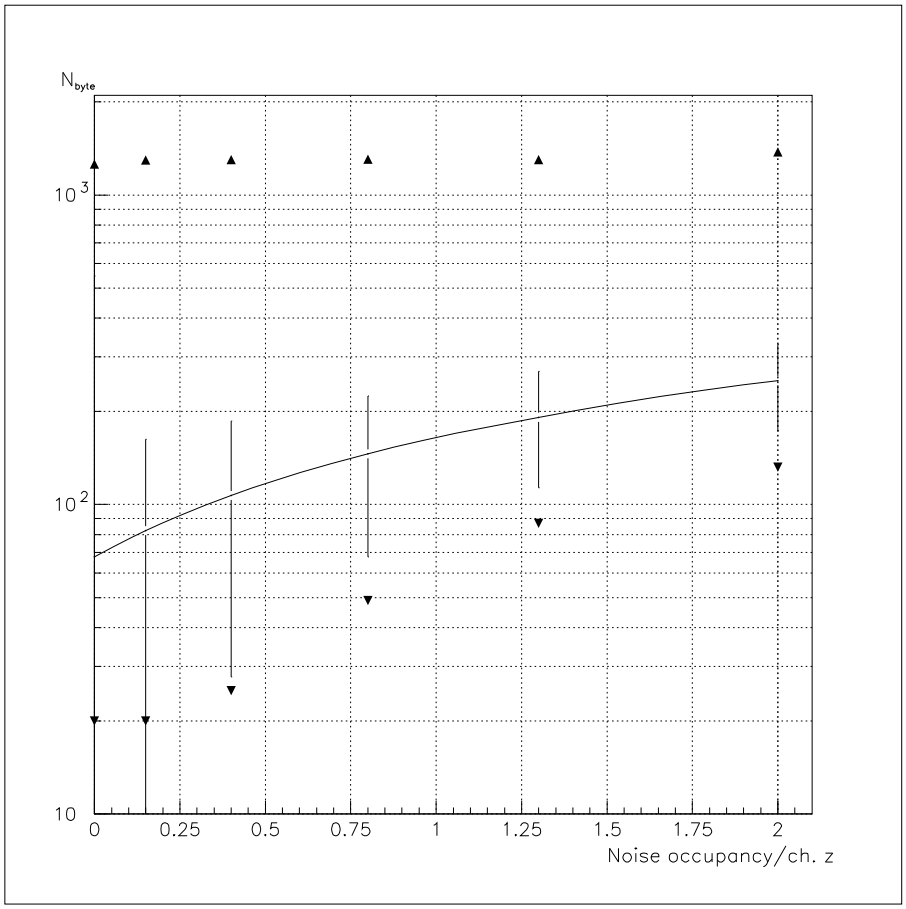
Figur 5.60 viser fordelingen av hvor mange hendelser som er i DSR bufferet ved forskjellig støy. En kan tydelig se en økning av tiden DSR bufferet er fullt når en sammenligner med simuleringer gjort med program som genererer fysikken selv. Dette gjelder spesielt når støyen er lav slik at en gjennomsnittlig har en lavere belastning på DSR bufferet.

[12] En kunne ellers tenkt seg å legge til $\approx \frac{0.45\%}{30mbias} \cdot (30mbias - 16mbias) = 0.21$ prosentpoeng for å kompensere for differansen på 14 mbias.

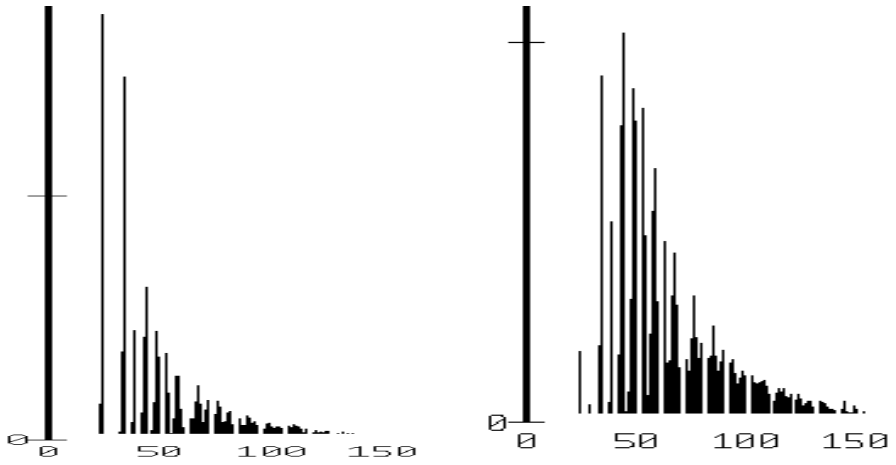
[13] Må også, ved sammenligning, huske på at sannsynligheten for treff funnet for program som generer fysikken selv ble pessimistisk satt noe høyt i kapittel 2.9.1.



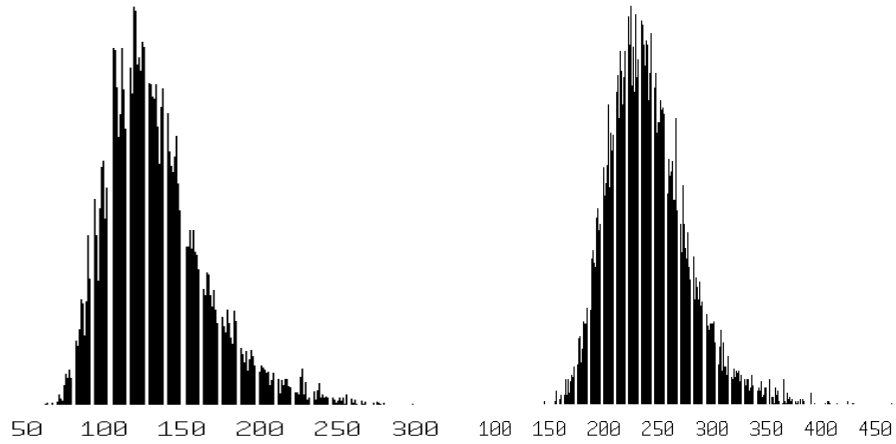
Figur 5.56: *Simulert gjennomsnittlig antall signal i DM som funksjon av støy ved bruk av MC generert fysikk. Strekene tilsvarer* $\pm 1\sigma$.



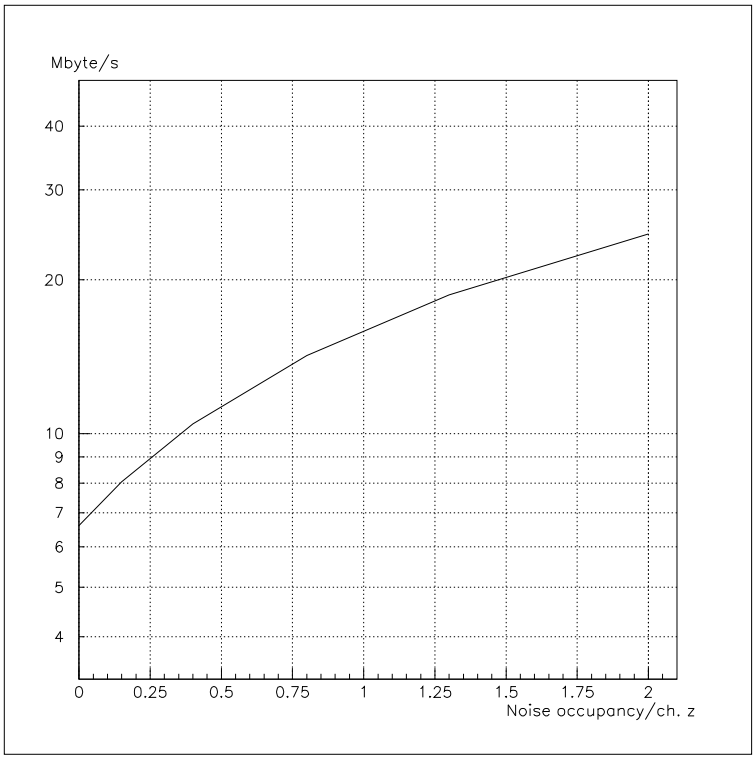
Figur 5.57: *Simulert gjennomsnittlig størrelse på datapakkene ut av DM som funksjon av støy ved bruk av program som bruker fysikk fra MC generert fysikk. Strekene tilsvarer* $\pm 1\sigma$ *og merkene viser den maksimale og minimale verdien som simuleringen har gitt.*



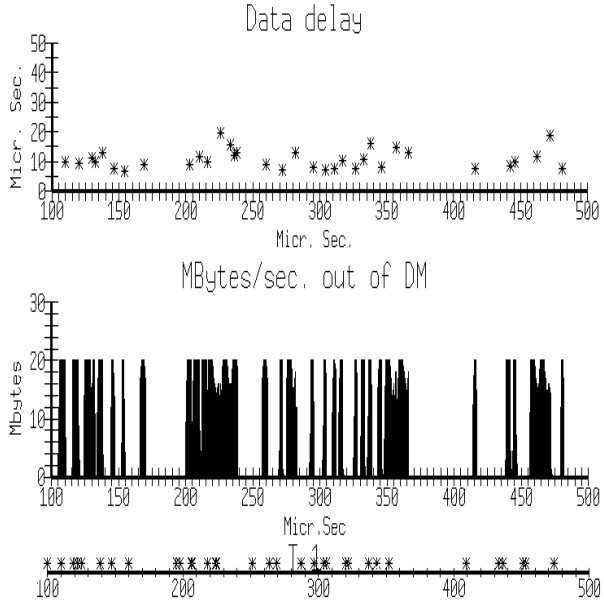
Tabell 5.9: *Simulert fordeling av størrelsen på datapakkene i byte ut av DM ved bruk MC generert fysikk. Her uten støy og med støy*$= 0.15\%$*. Midlet er her* 67.6 *byte/pakke og* 82.4 *byte/pakke.*



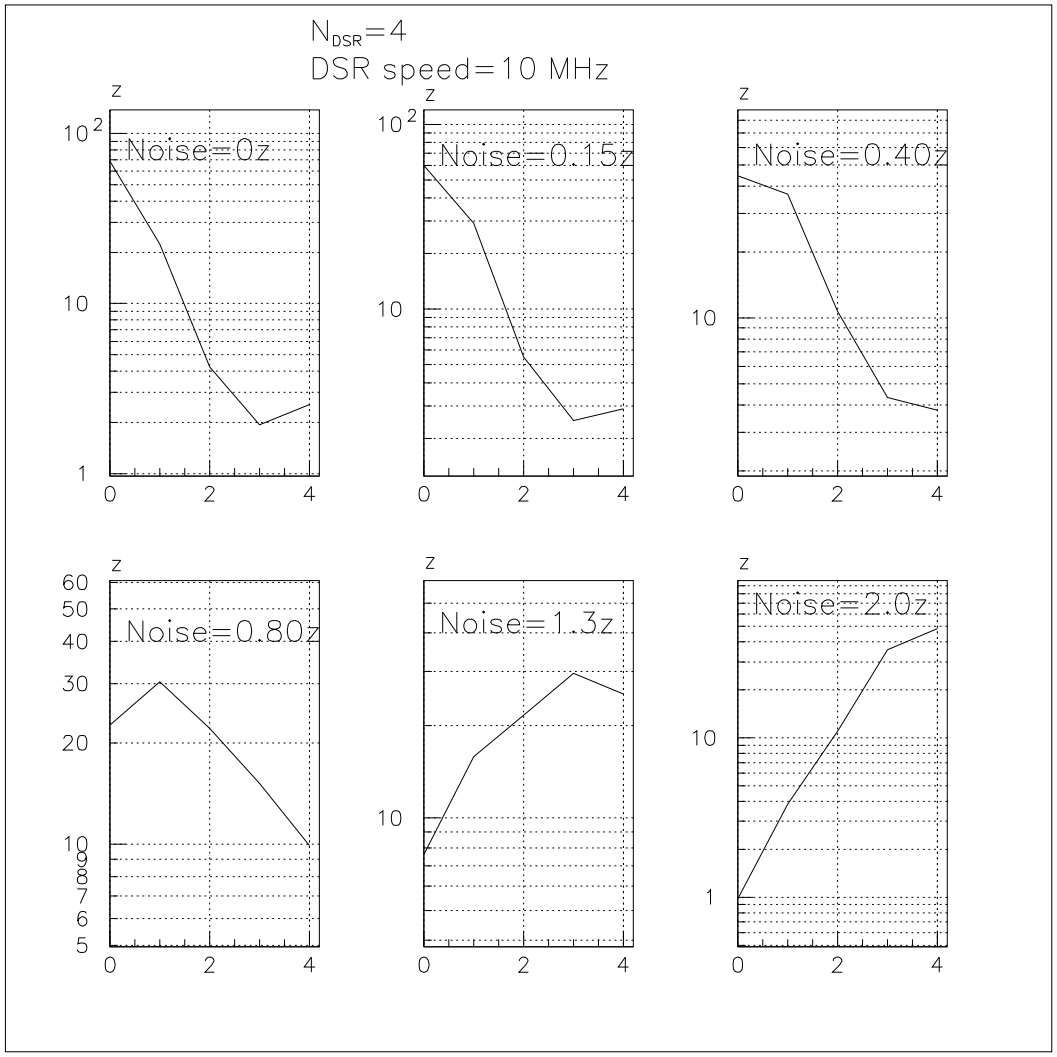
Tabell 5.10: *Simulert fordeling av størrelsen på datapakkene i byte ut av DM ved bruk MC generert fysikk. Her med støy*$= 0.8\%$ *og* $= 2.0\%$*. Midlet er her* 145.5 *byte/pakke og* 251.3 *byte/pakke.*



Figur 5.58: *Simulert Mbyte/s ut av DM som funksjon av støy ved bruk MC generert fysikk.*

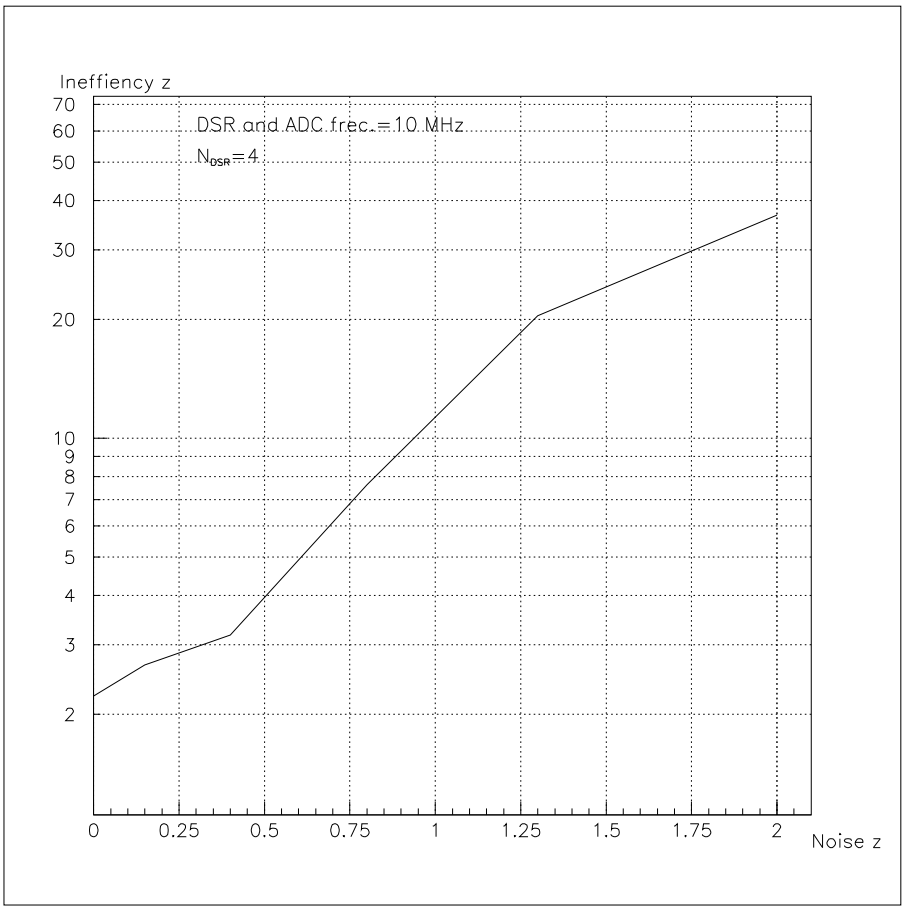


Figur 5.59: *Simulert strøm av data ut av DM med tiden i μs som abscisse og program som behandler DM som alternativ 1 og MC generert fysikk. Nederst sees triggerene og øverst forsinkelsen fra triggeren er gitt til hele hendelsen er lest ut av DM. Her med $N_{DSR} = 4$, $t_{DSR} = 100ns$, støy= 0.15% og utlesningshastighet fra DM på 20 Mbyte/s.*

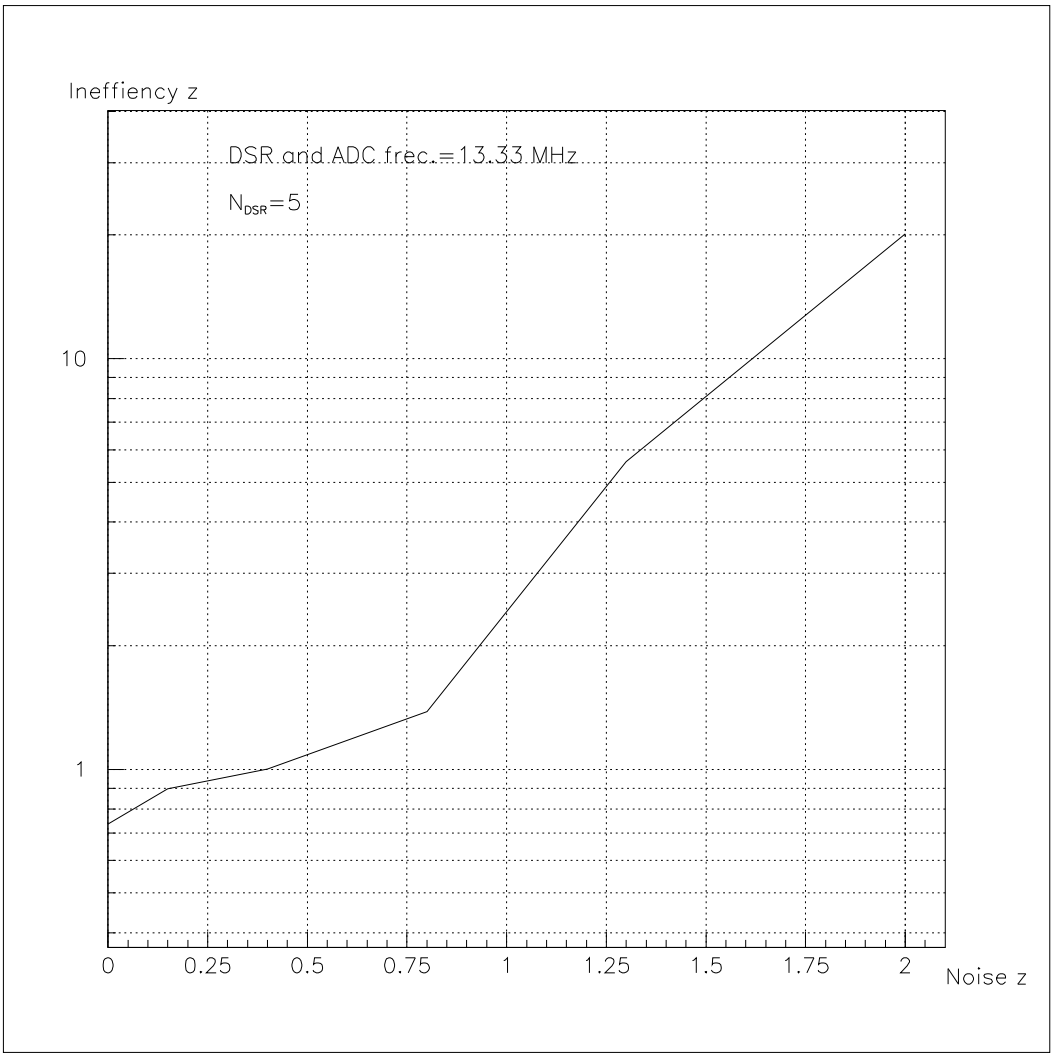


Figur 5.60: *Simulert sannsynlighetsfordeling av hvor mange hendelser som er i DSR bufferet med program som behandler DM som alternativ 1 og MC generert fysikk. Her med* $N_{DSR} = 4$ *og* $t_{DSR} = 100ns$*. Abscissen viser antallet hendelser i bufferet.*

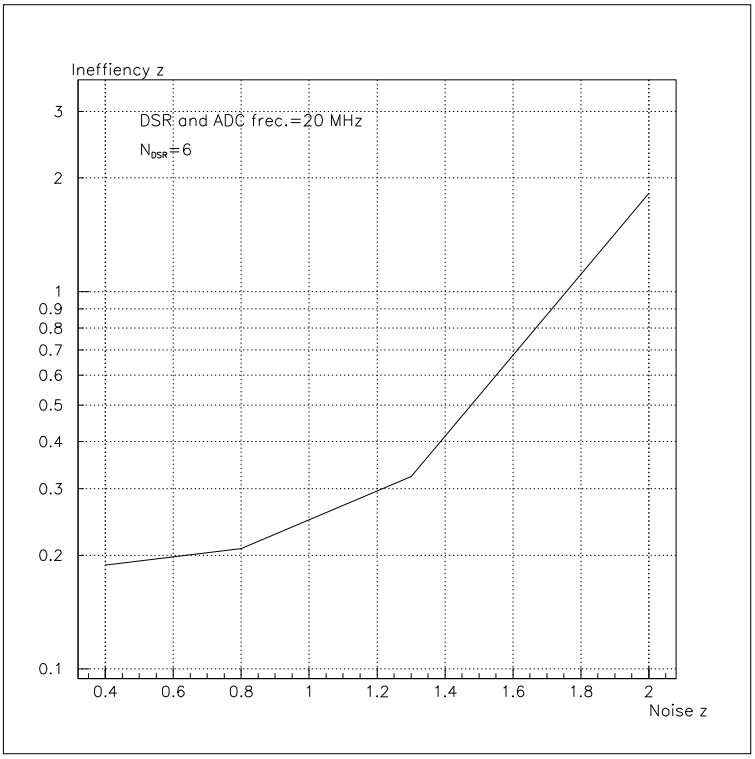
Figur 5.61, 5.62 og 5.63 viser sannsynligheten for å miste en hendelse pga. DSR bufferoverflyt som funksjon av støyen ved forskjellige DSR parametere. En kan se en tydelig økning av ineffektivitet ved lav støy i forhold til simulering med program som genererer fysikken selv.



Figur 5.61: *Simulert sannsynlighet for å miste en hendelse i DSR bufferet som funksjon av støyen med program som behandler DM som alternativ 1 og MC generert fysikk. Her med* $N_{DSR} = 4$ *og* $t_{DSR} = 100ns$.



Figur 5.62: *Simulert sannsynlighet for å miste en hendelse i DSR bufferet som funksjon av støyen med program som behandler DM som alternativ 1 og MC generert fysikk. Her med* $N_{DSR} = 5$ *og* $t_{DSR} = 75ns$.



Figur 5.63: *Simulert sannsynlighet for å miste en hendelse i DSR bufferet som funksjon av støyen med program som behandler DM som alternativ 1 og MC generert fysikk. Her med* $N_{DSR} = 6$ *og* $t_{DSR} = 50ns$.

Tabell 5.11 viser tapet i utbufferet ved forskjellige parametere. En kan ikke se noen vesentlig forskjell fra simuleringer gjort med program som generer fysikken selv.

Noise % ——— Out speed	Buffersize N_{out}	0.0%	0.15%	0.4%	0.8%	1.3%	2.0%
$t_{DSR} = 100ns$ 20 Mbyte/s	300byte	0.0	0.0	0.0	0.0	0.0	0.0
$t_{DSR} = 75ns$ 20 Mbyte/s	300byte	0.0	0.0	0.01	0.4	1.3	1.0
$t_{DSR} = 50ns$ 20 Mbyte/s	300byte	-	-	1.2	4.6	14.2	26.2

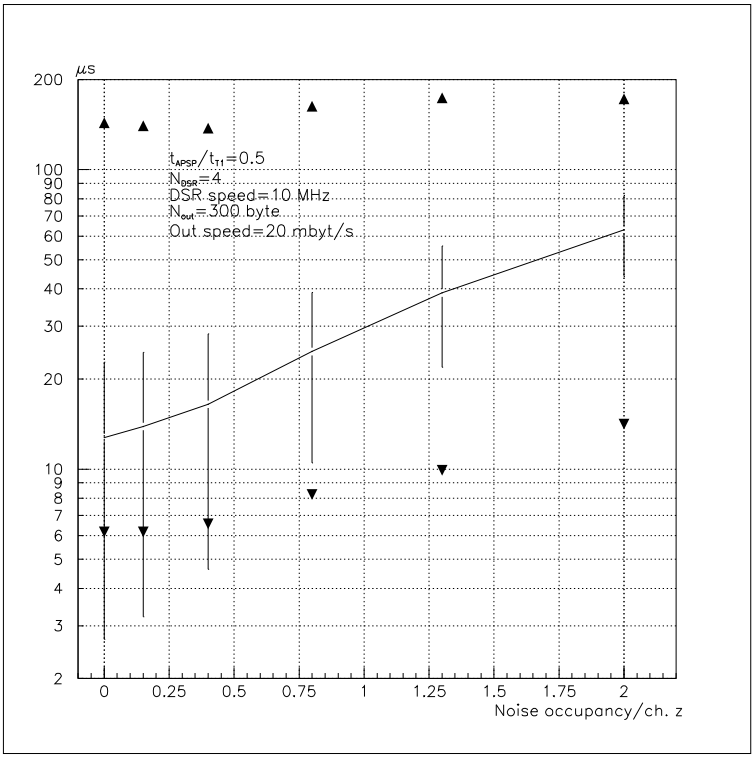
Tabell 5.11: *Simulert tap i % av data i utbufferet ved forskjellige parametere med program som behandler DM som alternativ 1 og MC generert fysikk.*

Figur 5.64 og 5.65 viser gjennomsnittlige forsinkelse en hendelse bruker på å bli lest ut av DM fra en trigger blir gitt til hele hendelsen er ute av DM. En kan også se maksimal og minimal tidene som er funnet fra simuleringene . En kan se en økning i den gjennomsnittlige prosesseringstiden ved lav støy i forhold til simulering gjort med program som genererer fysikken selv. En ser også en økning i den maksimale prosesseringstiden.

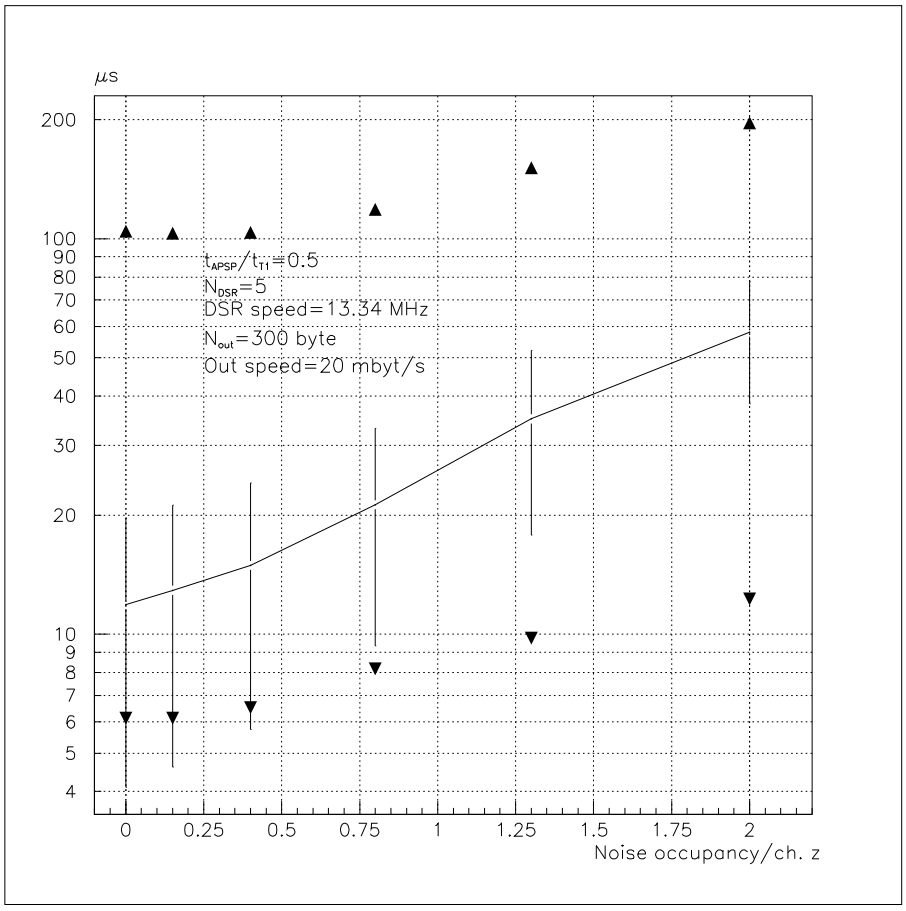
Figur 5.66, 5.67, 5.68, 5.69, 5.70 og 5.71 viser fordelingen av hvor lang tid en hendelse bruker på å bli lest ut av DM fra en trigger blir gitt til hele hendelsen er ute av DM. En kan se at en har fått en hale ut til høyre i forhold til simuleringer gjort med program som generer fysikken selv.

Figur 5.72, 5.73 og 5.74 viser den totale ineffektiviteten for DM inkludert ineffektivitet pga. triggere som kommer for tett og ineffektivitet i ADB[14].

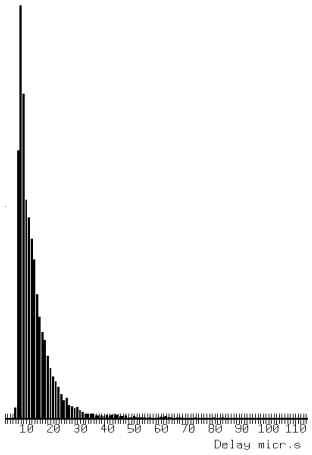
[14] Evt. ineffektiviteten i utbuffereter ikke inkludert.



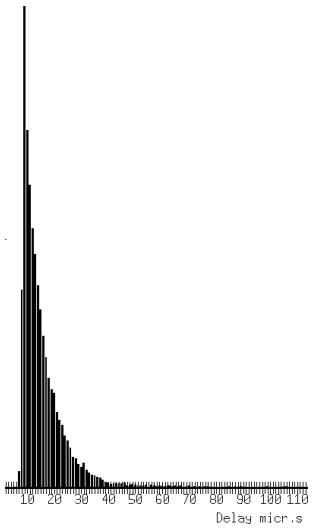
Figur 5.64: *Simulert forsinkelse fra en trigger gis til hele hendelsener lest ut av DM som funksjon av støyen med program som behandler DM som alternativ 1 og MC generert fysikk. Her med* $N_{DSR} = 4$ *og* $t_{DSR} = 100ns$*. Strekene tilsvarer* $\pm 1\sigma$ *og merkene viser den maksimale og minimale verdien som simuleringen har gitt.*



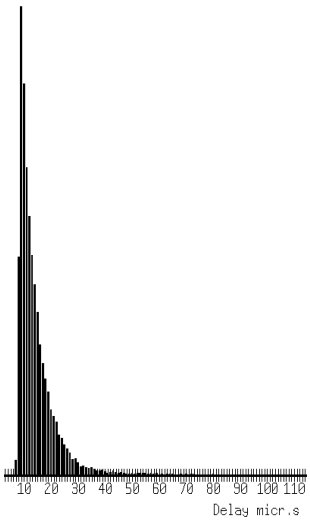
Figur 5.65: *Simulert forsinkelse fra en trigger gis til hele hendelsener lest ut av DM som funksjon av støyen med program som behandler DM som alternativ 1 og MC generert fysikk. Her med* $N_{DSR} = 5$ *og* $t_{DSR} = 75ns$. *Strekene tilsvarer* $\pm 1\sigma$ *og merkene viser den maksimale og minimale verdien som simuleringen har gitt.*



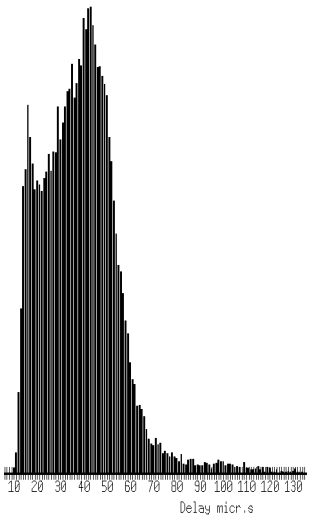
Figur 5.66: *Simulert fordeling av forsinkelsen en hendelse bruker før den er lest ut av DM med program som behandler DM som alternativ 1 og MC generert fysikk. Her med støy*= 0.15%, $N_{DSR} = 4$, $t_{DSR} = 100ns$. *Midlet er her* $13.9\mu s$.



Figur 5.67: *Simulert fordelingav forsinkelsen en hendelse bruker før den er lest ut av DM med program som behandler DM som alternativ 1 og MC generert fysikk. Her med støy*= 0.4%, $N_{DSR} = 4$, $t_{DSR} = 100ns$. *Midlet er her* $16.5\mu s$.

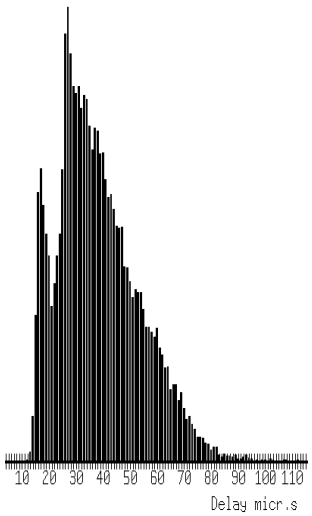


Figur 5.68: *Simulert fordeling av forsinkelsen en hendelse bruker før den er lest ut av DM med program som behandler DM som alternativ 1 og MC generert fysikk. Her med støy*= 0.4%, $N_{DSR} = 5$, $t_{DSR} = 75ns$. *Midlet er her* $14.9\mu s$.

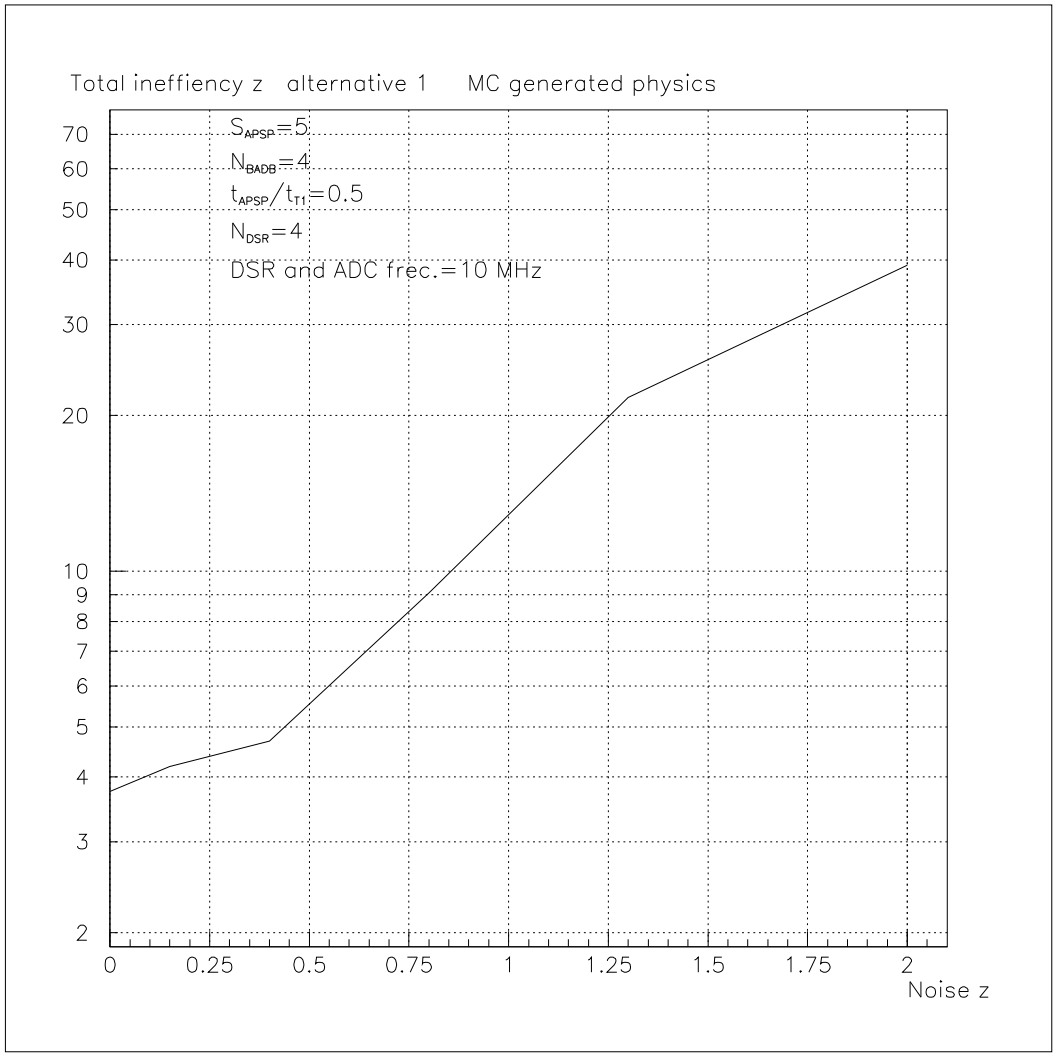


Figur 5.69: *Simulert fordeling av forsinkelsen en hendelse bruker før den er lest ut av DM med program som behandler DM som alternativ 1 og MC generert fysikk. Her med støy*= 1.3%, $N_{DSR} = 4$, $t_{DSR} = 100ns$. *Midlet er her* $38.8\mu s$.

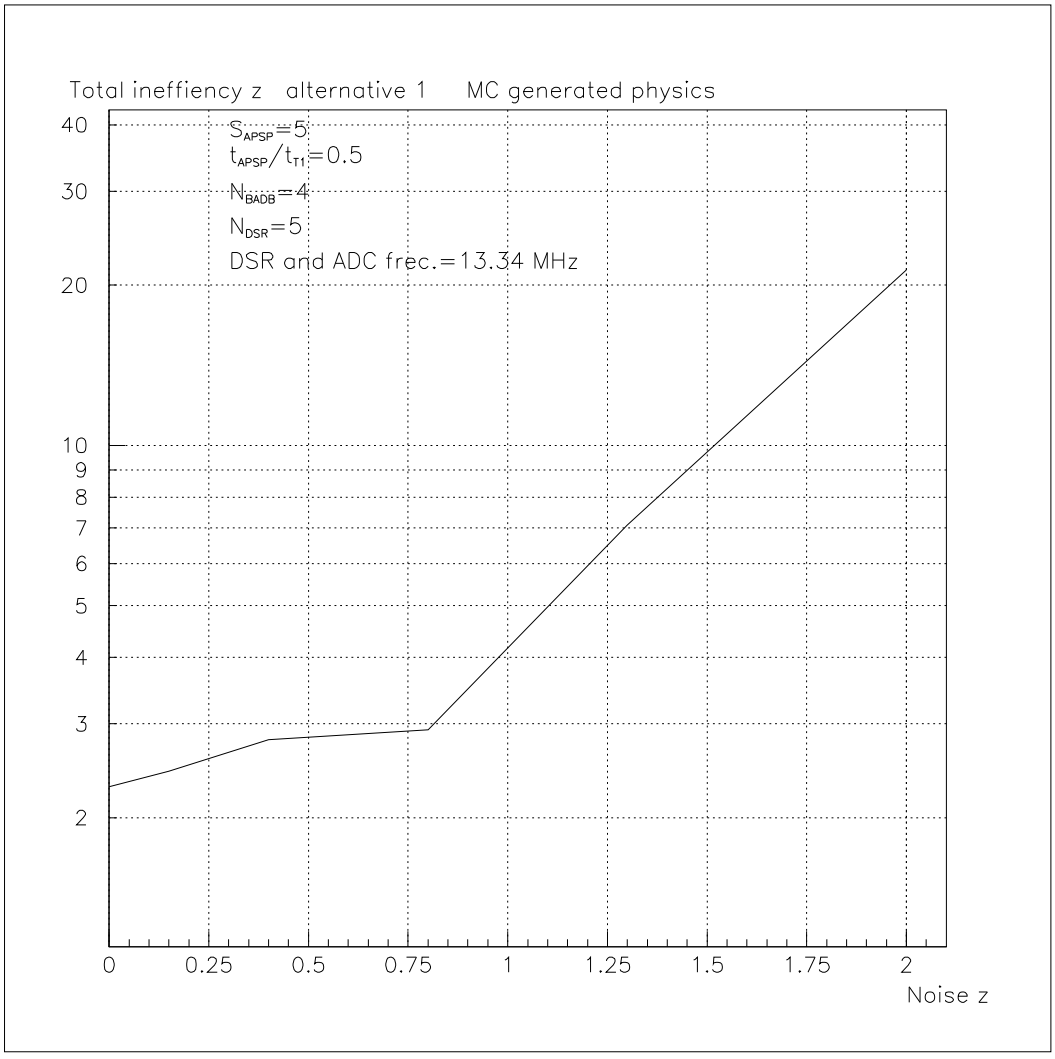
Figur 5.70: *Simulertfordeling av forsinkelsen en hendelse bruker før den er lest ut av DM med program som behandler DM som alternativ 1 og MC generert fysikk. Her med støy*= 1.3%, $N_{DSR} = 5$, $t_{DSR} = 75ns$. *Midlet er her* $35.0\mu s$.



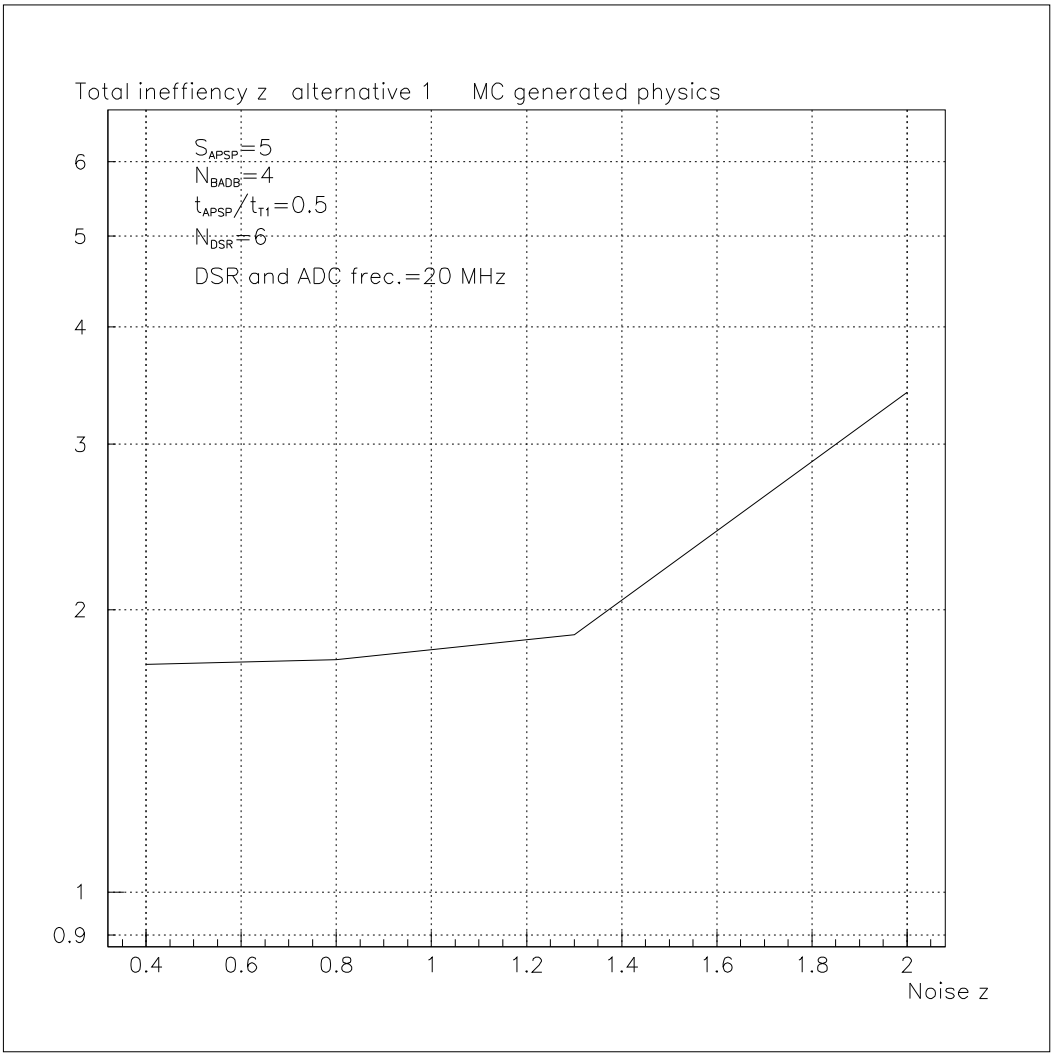
Figur 5.71: *Simulert fordeling av forsinkelsen en hendelse bruker før den er lest ut av DM med program som behandler DM som alternativ 1 og MC generert fysikk. Her med støy*= 2%, $N_{DSR} = 6$, $t_{DSR} = 50ns$. *Midlet er her* $38.4\mu s$.



Figur 5.72: *Simulert total ineffektivitet for DM som funksjon av støyen med program som behandler DM som alternativ 1 og MC generert fysikk. Her med* $S_{APSP} = 5$, $N_{ADB} = 4$, $\frac{t_{APSP}}{t_{T1}} = 0.5$, $N_{DSR} = 4$ *og* $t_{DSR} = 100ns$.



Figur 5.73: *Simulert total ineffektivitet for DM som funksjon av støyen med program som behandler DM som alternativ 1 og MC generert fysikk. Her med* $S_{APSP} = 5$, $N_{ADB} = 4$, $\frac{t_{APSP}}{t_{T1}} = 0.5$, $N_{DSR} = 5$ *og* $t_{DSR} = 75ns$.



Figur 5.74: *Simulert total ineffektivitet for DM som funksjon av støyen med program som behandler DM som alternativ 1 og MC generert fysikk. Her med* $S_{APSP} = 5$, $N_{ADB} = 4$, $\frac{t_{APSP}}{t_{T1}} = 0.5$, $N_{DSR} = 6$ *og* $t_{DSR} = 50ns$.

5.3 Korrelerte tap i forskjellige detektormoduler.

Så lenge detektor systemet er synkront vil tapet av data være 100% korrelert. I alle detektorer i hele eksperimentet vil en ha en synkronitet frem til nivå 1 triggeren er gitt til submodulene.

Imellom detektormodulene vil synkroniteten være avhengig av det alternativet for utlegg en velger. Ved alternativ 1 vil det være synkronitet frem til diskriminator for de forskjellige DM i innerdetektoren. Ved alternativ 2 og 3 vil en kun ha synkronitet frem til bufferet i ADB.

Ved simulering av DM som alternativ 1 og MC generert fysikk har en ved 0.40% støy, $N_{DSR} = 5$ og $t_{DSR} = 100ns$ fått en sannsynlighet for tap pga. DSR bufferoverflyt tilsvarende 2.2%. Ved å se når det er tap i en DM og sammenligne dette med når det er tap i en annen DM kan en se hvor stor korrelasjonen av tap mellom to DM er. En finner at i $\approx 14\%$ av tilfellene har begge mistet hendelsen samtidig. Dette viser at det er en viss korrelasjon av tap mellom de forskjellige DM.

Tidligere i kapitelet om ADB har vi funnet det korrelerte tapet. Vi skal her diskutere de lokale ukorrelerte ineffektivitetene. Pga. at korrelasjonen av tap i DSR er så liten (14% i eksemplet over) og en ser bort fra tap i utbufferet kan vi tilnærmet si at:

$$\varepsilon_{async} \approx \varepsilon_{DSR} \tag{5.8}$$

for innerdetektoren ved alternativ 1.

For laget ved $r = 20cm$ med ≈ 400 dobbeltsidige DM, 0.4% støy, $N_{DSR} = 4$ og $t_{DSR} = 100ns$ vil fra formel 2.6 $P_{\varepsilon=1}$ bli ≈ 0.000002 ved bruk av fysikk generert fra MC simuleringer og ≈ 0.7 ved bruk av program som genererer fysikken selv.

Som nevnt ovenfor er det pessimistisk å kun bruke fysikk fra RoI. For å finne $P_{\varepsilon=1}$ kan en isteden omskrive formel 2.6 og multiplisere de to sannsynlighetene funnet ovenfor på følgende måte:

$$P_{\varepsilon=1} \approx {\varepsilon_{DSR-MC}}^{n_{RoI}} \cdot {\varepsilon_{DSR-RND}}^{(n-n_{RoI})} \tag{5.9}$$

hvor ε_{DSR-MC} og $\varepsilon_{DSR-RND}$ er effektiviteten funnet ved program som genererer fysikken fra MC simuleringer og program som genererer fysikken selv respektivt. n_{RoI} og $(n - n_{RoI})$ vil være antallet DM i og utenfor RoI. En kan anta $n_{RoI} = 15$ og $(n - n_{RoI}) = 385$ for laget ved $r = 20cm$. Figur 5.75 og 5.76 viser sannsynligheten for å ikke miste noen DM pga. DSR overflyt ved bruk av formelen ovenfor.

5.4 Kommentarer til resultatene.

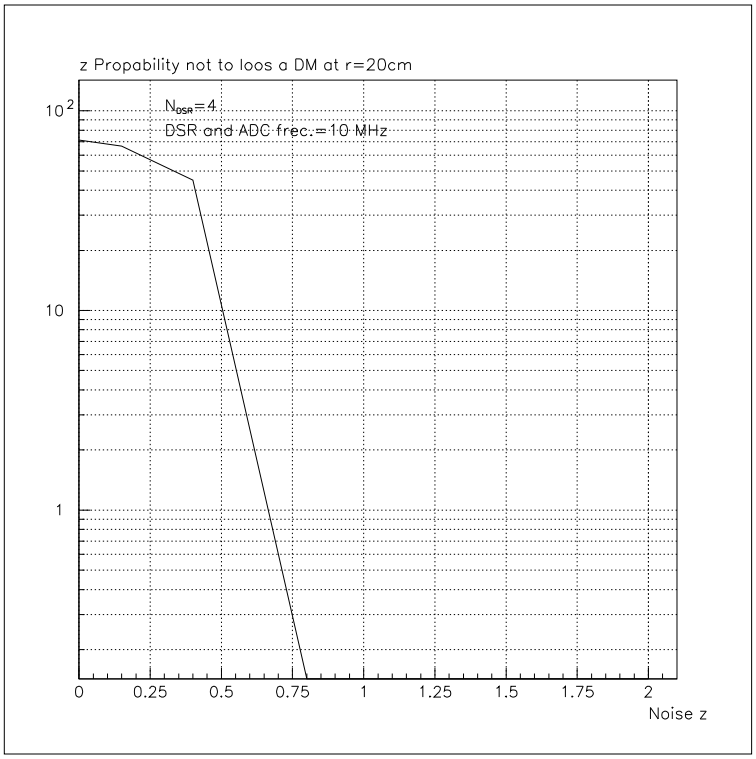
5.4.1 Dataraten i DM.

Antall signaler generert i DM ved simuleringer stemmer bra overens med beregnede.

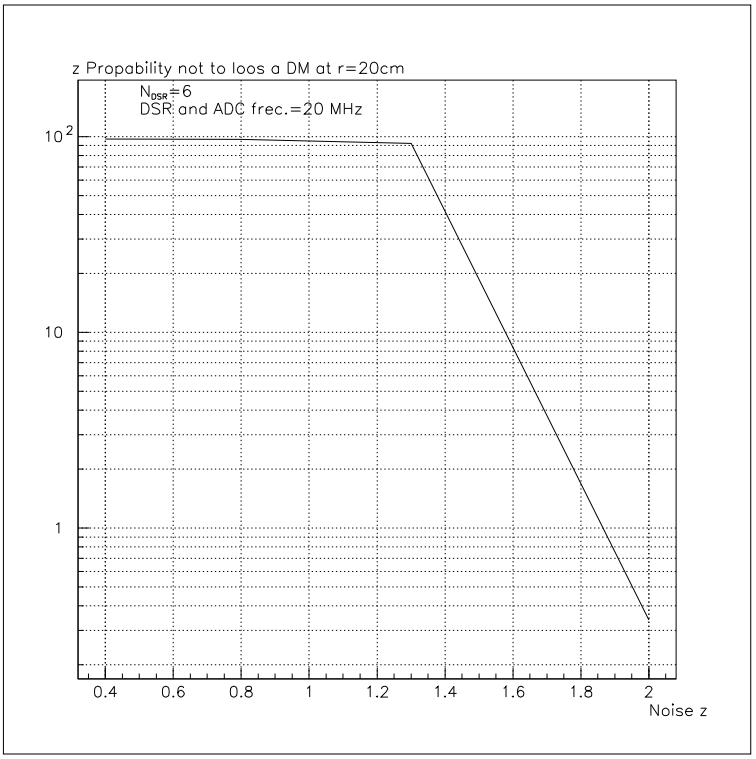
Den gjennomsnittlige størrelsen på en pakke ut av DM funnet ved simuleringer er fra $\approx 1\%$ til $\approx 3\%$mindre enn det som er beregnet[15]. Grunnen til dette er hovedsakelig at ved signal i to kanaler som maksimum har 2 kanaler imellom vil en kun lese ut en kanaladresse [16], og ved signal i to kanaler som har mindre enn 2 kanaler imellom vil de ha felles nabokanaler. En kan av dette se at denne pakkingen vil gi $\approx 3\%$ reduksjon i dataraten ved 2% støy og avtagende ved

[15] 0.0% støy og 2.0% støy

[16] Det vil være to nabokanaler imellom



Figur 5.75: *Simulert sannsynlighet for at en ikke mister noen DM i detektorlaget ved* $r = 20cm$ *og 15 DM i RoI av totalt 400 DM.*



Figur 5.76: *Simulert sannsynlighet for at en ikke mister noen DM i detektorlaget ved* $r = 20cm$ *og 15 DM i RoI av totalt 400 DM.*

lavere støy og derfor ikke føre til noen vesentlig reduksjon av dataraten [17]. Det vil derfor være nærliggende å tenke seg at en ikke foretar denne typen pakking. En fordel med dette vil være at en kan ha kontroll over hvilke kanaler som blir lest ut pga. at de var over grenseverdien til diskriminatoren og hvilke kanaler som blir lest ut pga. at de er nabokanaler. Dette pga. at en vil tilegne kanaladresse spesifikt til kanalen med treff. En kan da f.eks. tenke seg at det gis en form for signal fra DSR til MC når kanalen med treff leses ut.

Den beregnede raten Mbyte/s ut av DM stemmer bra overens med simuleringene bortsett fra den samme reduksjonen som nevnt ovenfor.

5.4.2 DSR bufferet.

I kapittel 5.1.2 er det benyttet to metoder for å finne fordelingen av hvor mange hendelser det er i utbufferet. Begge metodene baserer seg på at hendelsene kommer helt tilfeldig fra APSP med en rate α. I simuleringene vil ikke hendelsene komme tilfeldig fordelt, fordi:

- Triggerene som kommer for tett er fjernet.
- Ved klumping[18] av flere triggere som kommer tett etter hverandre har disse gitt overflyt i ADB og noen av disse er blitt fjernet.
- Det vil alltid være et minste intervall t_{APSP} mellom hver hendelse ut fra APSP.

Alle disse effektene vil gjøre at opphopningen og tapet i DSR-bufferet blir mindre ved simuleringer enn ved beregningene. Dette forklarer også forskyvningen til høyre av den beregnede fordelingen av hvor mange hendelser som er i DSR bufferet i forhold til simuleringene. En kan se at den metoden som er funnet i denne oppgaven her stemmer bedre overens med simuleringene enn metoden funnet fra [20]. Dette gjelder spesielt ved beregninger av ineffektivitet for DSR. Tilnærmelsene nevnt ovenfor, antagelsen om at prosesseringstiden er konstant og forenklingen gjort i ligning 4.11 gjør at resultatene må behandles med varsomhet selv om de stemmer bra med simuleringene.

5.4.3 Ineffektivitet ved variasjon av areal.

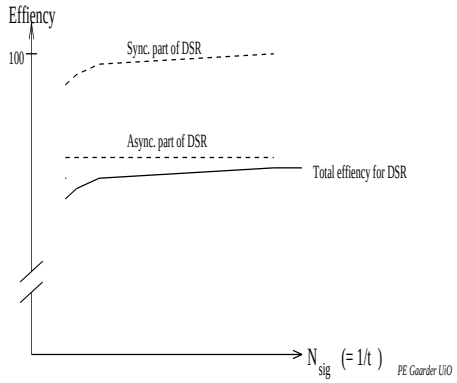
En kan av tabell 5.8 se at det er et lite avvik i ineffektiviteten, ε_{async} for DSR, selv om prosesstiden varieres omvendt proporsjonalt med N_{sig}. N_{sig} er da gitt av formel 5.1 og vil være det gjennomsnittlige antall signal påtrykt detektormodulen. Det kan da virke som at det er en feil i antagelsen om at ε_{async} vil være konstant om prosesstiden til de asynkrone prosesser varieres omvendt proporsjonalt med N_{sig}. Noen av grunnene til avviket kan være at:

- ved en økt sannsynlighet for signal i en kanal vil en ha en økt grad av pakking av data ut av DSR pga. at kanalene med signal vil ha flere felles nabokanaler. Denne sammenhengen kan finnes analytisk.
- tapet i DSR er ikke 100% asynkront pga. at trigger-raten vil være felles for de forskjellige DSR. Av dette følger det at tap pga. DSR bufferoverflyt er noe korrelert mellom de forskjellige DM.

[17] En må være klar over at den pakkingen nevnt ovenfor ikke har noen sammenheng med pakkingen som gjøres i forbindelse med adressen til nabokanalene til stripen med signal. Dette vil være en reduksjon av størrelse $\approx 2 \cdot N_{byte}$ byte for hver datapakke.

[18] Typisk flere enn 4-5 triggere innen $\approx t_{APSP}$.

Fra det siste punktet vil det da være nærliggende å anta at en kan uttrykke graden av korrelasjon mellom forskjellige detektorer analytisk ut fra hvor mye ε avviker fra å være konstant om tiden til prosessene settes proporsjonalt med $\frac{1}{N_{sig}}$. F.eks. om $\frac{d\varepsilon}{dN_{sig}} = 0$ viser dette at detektorene er 100% asynkrone. Figur 5.77 viser kvalitativt hvordan en kan tenke seg effektiviteten til DSR som funksjon av N_{sig} eller $\frac{1}{t_{DSR}}$. Den nederste kurven viser den totale effektiviteten til DSR. Den midterste kurven viser den effektiviteten en trolig kunne forvente om DSR var totalt asynkron. Differansen mellom disse kurvene vil da vise hvor synkron DSR er, eller hvor stor korrelasjon det er mellom de forskjellige detektor moduler.



Figur 5.77: *Kvalitativt eksempel på oppsplitting av effektiviteten for DSR i synkrone og asynkrone deler.*

5.4.4 Prosesseringstiden til DM.

Beregningen av forsinkelsen en hendelse bruker fra en trigger gis til hele hendelsen er ute av bufferet stemmer bra overens med simuleringene når det ikke er tap av data i DSR og utlesningshastigheten fra utbufferet er tilstrekkelig til å holde antallet byte her lavt.

Ved tap av triggere i DSR bufferet vil den gjennomsnittlige utlesningstiden bli redusert i forhold til beregnet tid. Dette pga. at en mister hendelser som ellers ville gitt et forholdsvis stort bidrag til den gjennomsnittlige utlesningstiden [19].

5.4.5 Tap i utbufferet.

Om utbufferet fylles opp kan en se at forsinkelsen øker betydelig i forhold til beregningene. Dette pga. at det ikke er tatt hensyn til utbufferet i beregningene. Når en ikke har tap i DSR bufferet kan en derfor indirekte se belastningen på utbufferet ved å sammenligne differansen mellom beregningene og simuleringene.

En kan se at en har en økende ineffektivitet i utbufferet ved økt hastighet til prosessene foran. Dette fordi en da får en gjennomsnittlig økt rate med data inn til utbufferet og fordi en i perioder får en økt intensitet.

En kan finne at en har et maksimalpunkt for ineffektiviteten til utbufferet som funksjon av støyen. Dette vil oppstå når utlesningshastigheten ut av utbufferet er omtrent den samme

[19] Når en mister triggere i DSR er dette pga. at bufferet er fullt og dermed har en også stor forsinkelse.

eller raskere en utlesningshastigheten fra ADC. Grunnen til dette er at en kun vil ha overflyt i utbufferet når det legges til 20 byte[20] for hver hendelse. Ved stor støy vil hendelsene være større og dette tilleget vil ha relativt mindre effekt. Den ineffektivitet som er vist i denne oppgaven er derfor hovedsakelig pga. disse 20 byte som legges til utbufferet i løpet av kun en BC. Det er derfor mere korrekt å betrakte de ineffektiviteter i utbufferet som er funnet i denne oppgaven kvalitativt og ikke kvantitativt.

Ved simulering av DM som alternativ 4 er det vist % av tiden DSR bufferet er fullt. En kan se at denne har en kraftig økning i forhold til alternativ 1 og 2 uten at ineffektivitet for DM øker i samme grad. Dette viser at en ikke ukritisk kan benytte tiden et buffer er fullt som ineffektivitet om det er en korrelasjon til prosessen foran og hendelsene ikke kommer tilfeldig inn til bufferet.

Ved simuleringer av DM med fysikk fra MC simuleringer er det kun lagt til 0.1 prosentpoeng støy for å kompensere for lavere luminositet. For å vurdere resultatene for DM vil dette være noe lavt. Dette pga. en pessimistisk bør tenke seg at DM alltid ligger i RoI som et verste tilfelle. En kan derfor, om en finner den virkelige støyen til $\approx 0.15\%$, lese av figurene ved en støy$\approx 0.25\%$.

[20] Hode og hale ol.

Kapittel 6

Konklusjon.

Det vil være av stor betydning å ha en god forståelse for graden av synkronitet eller korrelasjon mellom de forskjellige detektorer. Dette pga. at ved et eksperiment som ATLAS vil dette være nødvendig for å kunne si noe om sannsynligheten for at ingen detektorer skal være ineffektive ved en gitt hendelse.

En kan se at så lenge en har forholdsvis mange submoduler i eksperimentet bør disse submodulene, så langt det er mulig, være synkrone i sitt tap av data. Ideelt bør den delen av utlesningssystem som ligger inne i selve detektortønnen være synkront. En bør da velge alternativ 1 som beskrevet i denne oppgaven. En vil da ha god forståelse av systemet frem til og med diskriminatoren. Etter diskriminatoren bør en fortsatt diskutere graden av synkronitet og sannsynligheten for å kunne lese ut en hel hendelse uten tap av data i eksperimentet.

Dette tilsier at en bør revurdere den løsningen av DM etter diskriminatoren som er beskrevet i denne oppgaven.

Mulige løsninger kan være:

- Ha en intelligent trigger prosess som undertrykker triggere som kommer for tett og prioritering for Higgs partikkel etc. Simuleringer gjort med program som generer fysikken fra MC viser at dette har liten effekt. En vil få en noe forbedret effektivitet for DSR, men en totalt redusert effektivitet.

- Bruke alternativ 3 og ha et system som gir melding til GLV1 om at ADB er på grensen til å ha bufferoverflyt. GLVL1 vil da undertrykke triggere helt til det gis beskjed om at ADB har kapasitet igjen. En vil da ha et system uten tap av data, men det vil være uklart hvor stor del av triggerene GLV1 må undertrykke.

- Redusere oppløsningen til detektoren ved hard belastning. Eks. gå fra 8 bit til binær oppløsning ved hard belastning.

- Analog direkte utlesning av alle kanaler for hver brikke. Noe som fører til full synkron operasjon av utlesningssystemet.

- Ekstremt hurtige prosesser, eks. en ADC for hver kanal eller bare noen få kanaler. Dette vil innebære at disse prossesene ofte er uvirksomme men vil kunne håndtere større lokale fluktuasjoner av partikkler forbundet med QCD-jet strukturer.

Flere av disse løsningene studeres i dag innen ATLAS eller bør studeres nærmere.

Tillegg A

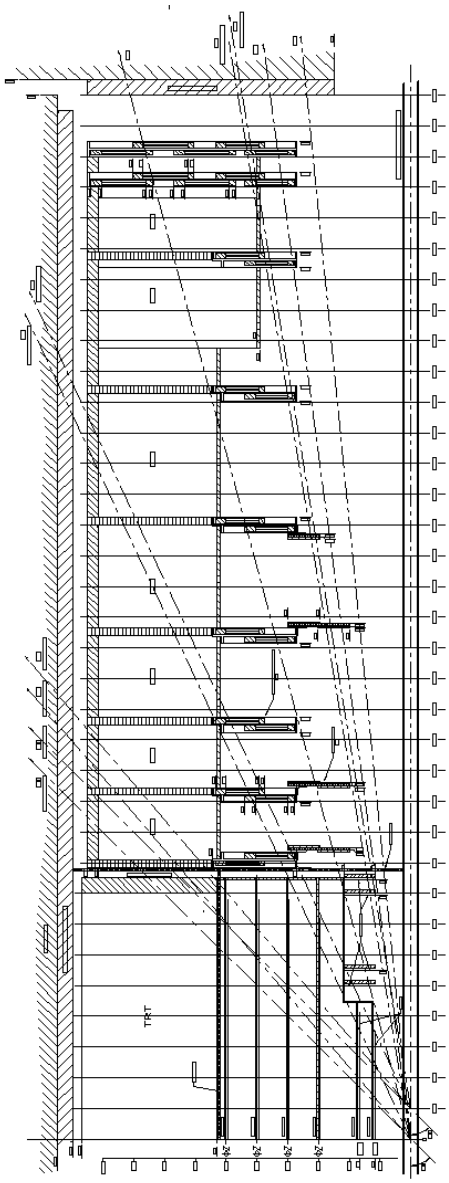
Ordliste.

ADB	*Analog Delay and Buffer.*
APSP	*Analog Pulse Shape Processor.*
ATLAS	***A** **T**oroidal **L**HC **A**pparatu**s**.*
Avgjøerlse	*Trigger.*
Avlesninger	*Samples.*
Brikke	*Chip*, mikroprosessor, integrert krets.
BC	*Bunch Crossing*, tidspunktet for hendelsene.
Brudd	*Violation.*
BS	*Bus Snooker.*
CM modul	*Control and Monitoring module.*
DSR	*Derandomizing and Sparsifying Readout.*
DM	*Detector Module.*
FELIX	FE brikke utviklet av RD-20.
FE	*Front End*, fremste del av detektor-elektronikken.
Forspenning	*Bias.*
Grenseverdi	*Threshold.*
Henfall	*Decay.*
Hendelse	*Event.*
Hendelse rate	*Event rate.*
Hermetisk lukket	*Hermetic.*
Jet	Sprut av partikler utfra en fragmentert kvark eller gluon.
LEP	*Large Electron Positron Collider.*
LHC	*Large Hadron Collider.*
Minimum bias hendelse	*Minimum bias event.*
MC brikke	*Module Control chip.*
Massefart	*Momentum, p=mv.*
Overgang, grenseflate	*Interface.*
P_T	*Transverse momentum.*
Pileup	Ekstra signal ovenpå det en er interessert i.
PE	*Protocol Engine.*
Senteravstand	*Pitch.*
Spredning	*Scattering.*
Spor-gjenkjennelse	*Pattern recognition.*
Støt parameter	*Impact* parameter.

T1	Trigger 1 avgjørelse
a_{LVL}	a_L, reduksjonsfaktor, ($a_{LVL1} \approx 400$).
a_{APSP}	a_A, antallet BC APSP bruker på å prosessere.
CAL_{INP}	Testsignal til forsterker i FE.
N_{APSP}	Antall lagringsceller APSP bruker, (normalt 4).
N_{sig}	Antallet kanaler med signal.
N_{celler}	Antallet seller i ADB.
$N_{channels}$	Antallet striper pr. brikke, (normalt 128).
N_{xchip}	Antallet brikker i x retning, (normalt 9).
N_{ychip}	Antallet brikker i y retning, (normalt 9).
N_{chip}	Antallet brikker totalt på DM, (normalt 18).
N_{byte}	Gjennomsnittlig størrelse for en datapakke
N_{pip}	Lengden på pipelinen.
N_{ADB}	Bufferdybden til ADB.
N_{DSR}	Bufferdybden til DSR bufferet.
N_{out}	Antall byte i utbufferet
$P_r(t)$	Sans. for å få r hendelser i løpet av tiden t.
P_{sample}	Sannsynligheten for å miste en T1 pga. at de kommer for tett.
S_{APSP}	Intervallet lagringsceller APSP går over fra og med første til og med siste, (normalt 5).
S_{ADB}	Antall ADB buffere i bruk.
S_{DSR}	Antall DSR buffere i bruk.
t_T	Gjennomsnittlig tid mellom to triggere.
t_{Tdelay}	Tiden fra BC, til T1 kommer til FE, ($t_{T1delay} < 2\mu s$).
$t_{DMdelay}$	Tiden fra en T1 mottas av DM til hele hendelsen er lest ut av DM.
t_{APSP}	Tiden APSP bruker på å prosessere.
t_{DSR}	Tiden DSR bruker på å lese ut en kanal
t_{ADC}	Tiden ADC bruker på å konvertere en verdi
t_{out}	Utlesningshastigheten fra DM, tid/byte.
t_p	Peaking time, stige-tid.
t_{total}	Totale tiden beregninger eller simuleringer er gjort over.
α	Gjennomsnittlig triggerrate, (normalt 100 KHz).
Δt	Tiden mellom to kollisjoner, (normalt 25 ns).
ε_{DAQ}	Den totale effektiviteten for DAQ systemet.
ε_{sync}	Effektivitet i den synkrone delen.
ε_{async}	Effektivitet i den asynkrone delen.
σ	Gjennomsnittlig utlesningsrate for en hendelse fra DSR bufferet.
σ_{nsd}	*Non single* diffraktive tverrsnit.
σ_{tot}	Totalt tverrsnitt.
σ_{el}	Elastisk tverrsnitt.
σ_{sd}	*Single* diffraktivt tverrsnit.
τ	Levetid.

Tillegg B

Siste versjon av innerdetektoren.



Tillegg C

MODSIM II

MODSIM II is a modular, block-structured high-level programming language which provides direct support for object-oriented programming and discrete-event simulation. MODSIM II is supported on a variety of machine architectures, sequential and parallel. The language is object-oriented. An *object* is an encapsulation of data record which describes the state of the object and procedures called methods which describes its behaviors. The language is divided into *modules*, each module is stored in a separate fil, mainmodule M—.mod, definitionmodule D—.mod and implementation module I—.mod. MODSIM II have a great variety of different ready-made procedures and modules stored in *library files*. Users familiar with Modula-2, Ada and Simula will recognize this approach. A module which requires re-compilation can be identified and scheduled for compilation, and analyzing the effects on other modules. The Compiler/linker will also automatically manage the compilation and linking of those portions of programs which are developed in the *C* language.

MODSIM II have a option SIMGRAPHICS II. Using SIMGRAPHICS II you can easily incorporate animation, presentation graphics and graphical user interface into your program. On a SUN computer SUNVIEW dialog boxes and menus are used. On DEC computer system, DEC WINDOWS dialog boxes and pull down menus are produced.

An additional program, SIMDRAW, can be used to build a graphical representation of a objects, dialog boxes, level meters, pie charts, 2-D plots, histograms, graphs and moving objects like a car. There is an option SIMVIDEO enables you to save the graphics commands during the execution of a program with graphic representation. You type -video as a option after the executable program, after the execution there will be saved a file *programname*.vid. If you run SIMVIDEO you will get a graphical view of an *ordinary home VCR*. With this you can edit or take a closer look at your simulation.

MODSIM II is a fast way programming tool for making a simulation program and a survey of a system, with a large library and a graphical representation.

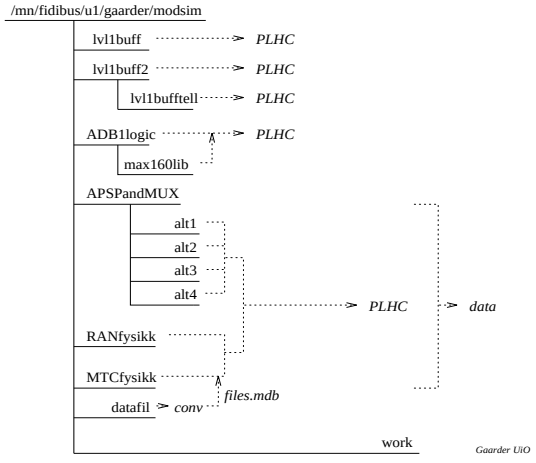
Tillegg D

Forklaring og bruk av MODSIM II programmene.

Generelt

Kildekoden og programmene finnes på nettverket tilhørende *fidibus.uio.no*.

Programmene finnes under /mn/fidibus/u1/gaarder/modsim/ og har flere versjoner. De forskjellige løsningene er bygd inn i spesifikke moduler. De moduler som er spesifikke for et utlegg av DM må da kopieres over til f.eks /mn/fidibus/u1/gaarder/modsim/work der en så må kompilere og linke programmet. Figur D.1 viser hvor en finner de forskjellige programmer. Modulene har generelt samme navn selv om de er beregnet for forskjellig utlegg av DM, dette for at det skal være enkelt å kombinere dem. Det er også i størst mulig utstrekning forsøkt å skille ut egenskapene til hver enkelt modul slik at utskiftning av en type modul ikke påvirker de andre moduler programmet blir bygd opp av.



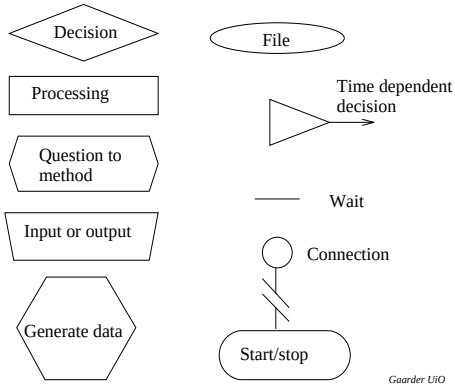
Figur D.1: *Strukturen for linking av forskjellige programmer.*

Ved oppstart av programmene kan en velge om en vil kjøre med grafikk eller ikke. Selv om en har grafisk skjerm er det å anbefale å kjøre lange simuleringer uten grafikk pga. at det øker hastigheten på programmene samtidig som en slipper problemer med evt. brudd på nettverket

fra terminalen til arbeidsstasjonen.

Generelt er programmene laget med en BC eller Δt som minste tidsenhet. Dette gjør at alle tidsparametere må settes i antall BC.

Figur D.2 viser betydningen av de forskjelligesymboler som er brukt i flytskjemaene. Modsim



Figur D.2: *Symboler brukt i flytskjemaene.*

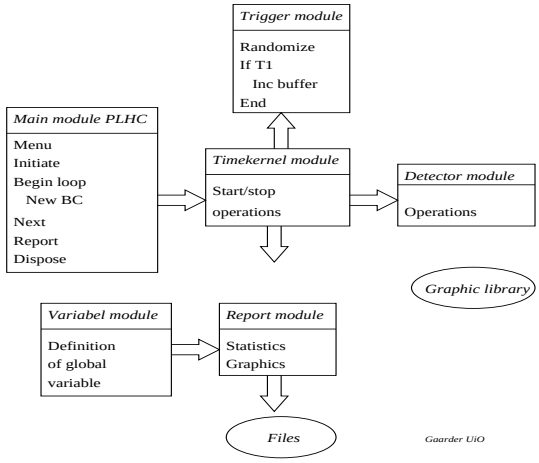
har et eget statistikkbibliotek som er benyttet i programmene. Dette har da beregnet det midlet og σ som følger målingene. Errorbaren på de grafene som har dette tilsvarer $\pm 1\sigma$.

Der programmets egen statistikk ikke har beregnet σ er denne normalt estimert til:

$$\sigma = \frac{\sqrt{\#tap}}{\#tap}. \tag{D.1}$$

D.1 Program som simulerer ADB som et standard køproblem.

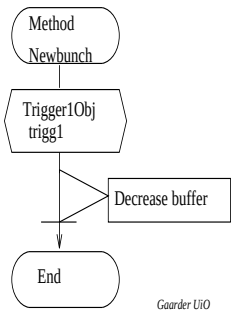
Programmet finnes under /lvl1buff2. Figur D.3 viser en oversikt over modulene som er laget for å kunne bygge opp programmet.



Figur D.3: *Modul oversikt over programmet som er brukt for å simulere ADB som et standard køproblem.*

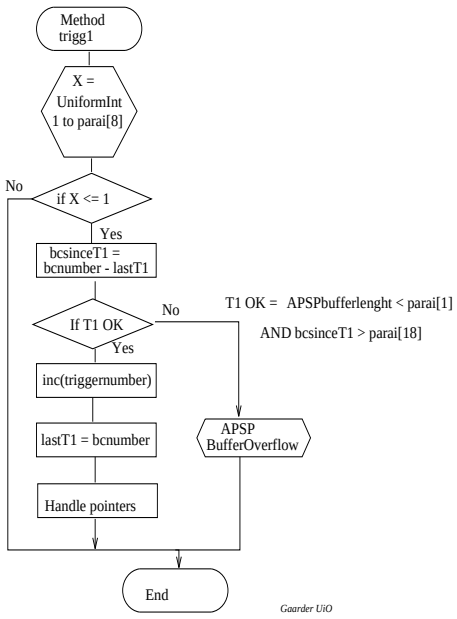
Hovedprogrammet starter en rutine som genererer en T1 ja eller nei avgjørelse for hver BC. Er intervallet til forrige trigger stort nok samtidig som det er plass i bufferet vil en øke bufferet med en. APSP vil starte så fort det er noe i bufferet. APSP vil så etter en gitt tid redusere bufferet med en, dvs. hendelsen opptar plass i bufferet helt til APSP er ferdig. Alle simuleringene er kjørt med samme *random seed* slik at rekken av triggere har vært den samme i fordeling og antall.

Figur D.4 viser grovt hvordan metoden som håndterer tidsflyten virker. Denne metoden blir kjørt for hver BC.



Figur D.4: *Grov oversikt over hvordan metoden som håndterer tidsflyten.*

Figur D.5 viser grovt hvordan metoden *trigg1* virker. PARAI[8] vil være trigger reduksjonen a_{LVL1}, PARAI[1] er det samme som N_{BADB} og PARAI[18] er $S_{APSP} - 1$.



Figur D.5: *Metode som genererer en T1 avgjørelse.*

Ved simulering av systemet som et standard køproblem er følgende parametere vært mulig å forandre:

- *Antall BC i simuleringen*, dette tilsvarer lengden av simuleringen. Antallet har vært 10^8 BC om annet ikke er nevnt. Dette skulle gi tilstrekkelig god statistikk for simuleringene som er gjort og tilsvarer $2.5s$ drift av LHC ved $40MHz$.
- *Reduksjon pga. trigger 1*, dette tilsvarer reduksjonen som blir gjort i LVL1. Denne er normalt satt til $a_{LVL1} = 400$ om annet ikke er nevnt. Det vil tilsvare en trigger 1 rate på 100 KHz.
- *Antall triggere før det kan gis en ny trigger*, dette er dødtiden pga. den destruktive utlesningen av ADB. Antallet tilsvarer hvor mange BC som må gå etter at det ble gitt en trigger før det kan komme en ny. Om denne f.eks er satt til 1 vil en trigger som kommer rett etter en annen bli forkastet. Denne er normalt satt til $N_{APSP} = 4$ ($S_{APSP} = 5$)[1] eller $N_{APSP} = 3$ ($S_{APSP} = 4$)[1], eller 0 når det er gjort simuleringer uten denne effekten.
- *Dybde på ADB*, dette er bufferdybden til ADB. Denne er normalt satt til $N_{ADB} = 4$.
- *Prosesstiden til APSP*, dette er tiden APSP bruker ($\frac{t_{APSP}}{\Delta t}$) på å prosessere en hendelse. Denne er normalt satt til 200 som tilsvarer $t_{APSP} = 5\mu s$.

Resultater en kan lese av etter en simulering vil være:

- *% av tid det har vært et gitt antall hendelser i bufferet*, dette regnet i antall BC som har gått med en bestemt størrelse på bufferet i forhold til totalt antall BC simuleringen har gått over.

[1] APSP hopper over en avlesning

- % *av triggere det har vært et gitt antall hendelser i bufferet*, dette regnet i antall triggere som har gått med en bestemt størrelse på bufferet i forhold til totalt antall triggere som er gitt.

- *Antall triggere som ikke er forkastet*, dette vil være antall triggere som er ferdig behandlet av APSP.

- *Antall triggere som er mistet pga. overflyt i bufferet.* Ved gjentatte simuleringer med $\frac{t_{APSP}}{t_{T1}} = 0.5$, $N_{ADB} = 4$ og $N_{APSP} = 4$ men ved forskjellig *random seed* har ineffektiviteten variert $\approx \pm 0.4\%$ av den gjennomsnittlige målingen [2].

- *Antall triggere som er mistet pga. at de kom for tett etter forrige.* Ved gjentatte simuleringer med $\frac{t_{APSP}}{t_{T1}} = 0.5$, $N_{ADB} = 4$ og $N_{APSP} = 4$ men ved forskjellig *random seed* har ineffektiviteten variert $\approx \pm 0.3\%$ av den gjennomsnittlige målingen.

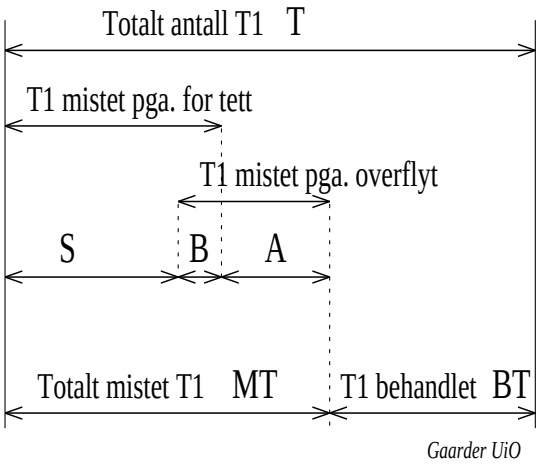
- *Antall triggere som er mistet pga. at de kom for tett etter forrige samtidig som bufferet var fullt.* Dette vil være triggere som er mistet av begge ovenfornevnte grunner. Disse er ikke regnet med ovenfor. Ved gjentatte simuleringer med $\frac{t_{APSP}}{t_{T1}} = 0.5$, $N_{ADB} = 4$ og $N_{APSP} = 4$ men ved forskjellig *random seed* har ineffektiviteten variert $\approx \pm 0.6\%$ av den gjennomsnittlige målingen.

En typisk simulering uten bruk av grafikk vil være:

```
hope% PLHC
**********************************
Graphical Display  Y/N ?? n
Numbers of bunch crossings in simulation ???  100000000
Trigger1 rejection  ??? 400
Numbers of bc before next trigger1 ??? 4
Buffersize APSP buffers ??? 4
APSP deconvolutiontime (in BCs) ??? 200
Report interval ??? 1000000
---> Create the detector-module and datastructure
Simulation Starting at Fri Jun  25 19:53:14 1994
---> Simulation Finished at Fri Jun 25 20:40:33 1993 It Took:2837 Second
50.856662 % of time with buffersize 0
32.936838 % of time with buffersize 1
12.028000 % of time with buffersize 2
3.507519 % of time with buffersize 3
0.670444 % of time with buffersize 4
Number of trigger who is not rejected 245714
Number of BCO's 100000000
Number of trigger_1s who are rejected  4045 1.646223% of counted
Number of trigger_1s who are rejected from APSP overflow  15910.647501% of coud
Number of trigger_1s who are rejected because both 80
************************************************************
```

[2] Tilsvarer f.eks. ineffektivitet $= 0.64\% \pm 0.003\%$

Tallene ovenfor tilsvarer BT=245714, MT=4045, A=1591, S=2374 og B=80 i figur D.6. Der B er de triggerene som ville bli mistet både pga. at de kommer for tett og fordi bufferet var fullt da de kom.



Figur D.6: *Innkommende trigger rate og oppsplitting av denne.*

Totalt antall triggere vil da være T=BT+MT som tilsvarer 249759 i det simulerte eksempelet ovenfor. Dette vil også gjelde for alle de andre simuleringene. Forventningsverdien av antallet triggere er $10^8/400 = 250000$. Dette tilsier at det statistiske avviket av antallet triggere generert i denne (og de andre) simuleringen avviker med $\approx 0.1\%$ under forventningsverdien.

Ved $\frac{t_{APSP}}{t_{T1}} = 1$ blir fordelingen som tabell D.1 viser. Den øverste rekken viser en bufferdybde på 4 og den andre en bufferdybde på 10. Tilstanden i hver ende kuttes til $\approx \frac{1}{2}$ pga. at en kun

#	0	1	2	3	4	5	6	7	8	9	10
%	13.42	22.92	25.57	25.46	12.63						
%	5.01	8.70	9.93	10.15	10.15	10.20	10.22	10.24	10.22	10.20	4.99

Tabell D.1: *Simulert fordeling av hvor mange hendelser som er i bufferet ved $t_{APSP}/t_{T1} = 1$ og en bufferdybde på henholdsvis 4 og 10.*

har mulighet for å gå til denne tilstanden fra en side, mens en i de andre tilstandene kan gå fra 2 sider. Forskyvningen mot høyre viser at systemet er svakt ustabilt. Ved å redusere t_{APSP}/t_{T1} til 0.975 vil fordelingen bli som tabell D.1 viser. Tabellen viser at systemet er stabilt. En kan da anta at systemet er på grensen til å være stabilt ved $t_{APSP}/t_{T1} \approx 1$.

Simulert tap av triggere pga. bufferoverflyt blir benyttet på 3 måter i oppgaven:

D.1.1 Metode ved sammenligning med analytisk beregning.

Dette vil si at en ikke mister triggere pga. at de kommer for tett. En kunne da regnet at ineffektiviteten var (A+B)/T. Dette vil gi et for lite resultat. Grunnen til dette er at det er en

#	0	1	2	3	4	5	6	7	8	9	10
%	6.27	10.56	11.54	11.23	10.66	10.12	9.61	9.14	8.70	8.27	3.90

Tabell D.2: *Simulert fordeling av hvor mange hendelser som er i bufferet ved* $t_{APSP}/t_{T1} = 0.975$.

del triggere som kommer for tett, S, som er fjernet og en har ikke lenger en Poisson fordeling av innkommende triggere. For å kunne sammenligne med analytiske beregninger må en derfor sette det minste intervallet til neste trigger til 0, dvs. ikke noe tap pga. at triggere kommer for tett.

D.1.2 Metode ved sammenligning med tiden bufferet er fullt.

En har da funnet de triggere som kommer når bufferet er fullt og dividert dette med totalt antall triggere over hele tiden. En får da at ineffektiviteten er (A+B)/T. Dette vil gi en ineffektivitet på 0.67% i eksempelet ovenfor.

D.1.3 Metode for å finne ineffektivitet for detektormodulens ADB.

Ineffektivitet for detektormodulens ADB vil være antall triggere forkastet av ADB dividert på antall triggere gitt til ADB. Forutsetter her at triggere som kommer for tett er forkastet på forhånd. En får da at ineffektiviteten er A/(BT+A). Dette tilsvarer 0.64% i eksempelet ovenfor.

D.2 Program som simulerer ADB komplett med registre.

Programmet finnes i to versjoner, med grafikk og uten grafikk. I versjonen uten grafikk er programmet trimmet ned for å øke hastigheten. Programmet med grafikk finnes under /mn/fidibus/u1/gaarder/modsim/ADB1logic. Programmet uten grafikk finnes under /mn/fidibus/u1/gaarder/modsim/ADB1logic/nographic. Det finnes også to grafiske filer, med henholdsvis en max ADBlengde på 160 celler og max 50 celler. Ved bruk av en spesifikk grafisk fil må denne hentes inn og kompileres/linkes til programmet.

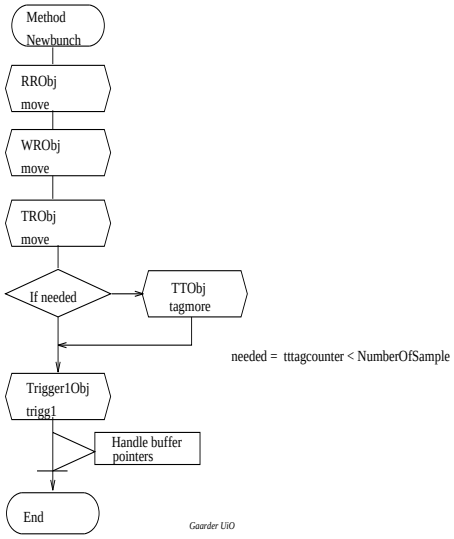
Figur D.3 viser en oversikt over modulene som er laget for å kunne bygge opp programmet. Det er hovedsakelig detektormodulen som er forandret fra simuleringen av ADB som et standard køproblem, i tillegg til at en har et ekstra grafisk vindu som viser tilstanden til ADB registrene. Figur D.7 viser grovt hvordan metoden som styrer tiden virker.

Figur D.8 viser grovt hvordan metoden *trigg1* virker.

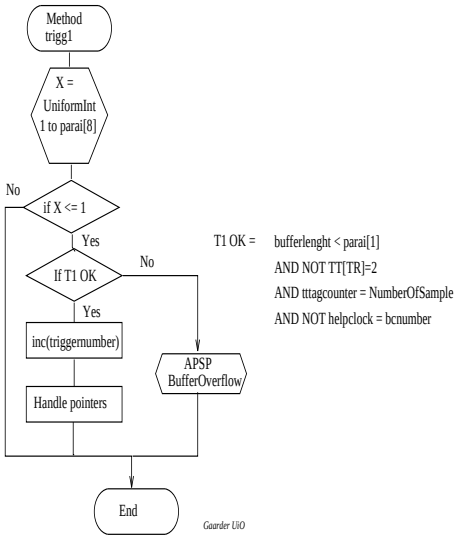
Ved simulering er følgende parametere i tillegg til de nevnt ovenfor vært mulig å forandre:

- *Dybde på ADB*, dette er buffer dybden til ADB. Denne er normalt satt til $N_{ADB} = 4$.
- *Lengde på pipeline*, dette er lengden av pipelinen i antall celler. N_{pip} er normalt satt til 30 ved grafikkfil som kun tar 50 celler og 80 [3] ved grafikkfil som tar 160 celler.
- *Antall avlesninger APSP gjør og bruker*, dette vil være N_{APSP}.

[3] $N_{pip} = 80$ tilsvarer en forsinkelse på $t_{T1delay} = 2\mu s$



Figur D.7: *Grov oversikt over hvordan metoden som håndterer tidsflyten ved fullstendig simulering av ADB.*



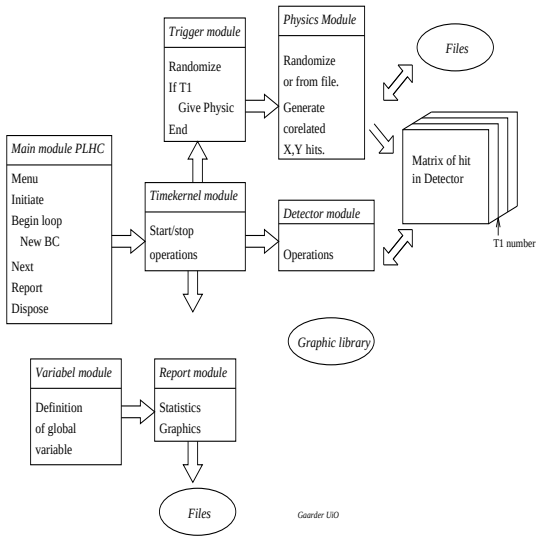
Figur D.8: *Metode som genererer en T1 avgjørelse. Her brukes registrene direkte for å avgjøre om det skal gis en T1.*

- *WR og TR hopp.* Dette er hvor langt WR pekeren hopper for hver BC. Denne er normalt satt til 1.
- *Den cellen APSP evt. hopper over.* APSP vil ikke bruke denne cellen. Om denne parameteren er 0 eller $> N_{APSP}$ vil $S_{APSP} = N_{APSP}$. Om denne parameteren er > 0 og $\leq N_{APSP}$ vil $S_{APSP} = N_{APSP} + 1$. En har muligheten til å hoppe over inntil to individuelle celler som vil gi $S_{APSP} = N_{APSP} + 2$.
- *Trigger seed.* Dette er frøet som random-generatoren starter med.
- *Antall BC i simuleringen*, dette tilsvarer lengden av simuleringen. Antallet har været 10^8 BC om annet ikke er nevnt.

Verdier en kan lese av etter en simulering vil være de samme som ved simulering med programmet som håndterer ADB som et standard køproblem.

D.3 Programmer som simulerer hele DM.

Programmet er en utvidelse av programmet beskrevet i D.1. Figur D.9 viser en oversikt over modulene som er laget for å kunne bygge opp programmet.



Figur D.9: *Moduloversikt over programmet som er brukt for å gjøre en fullstendig simulering av DM.*

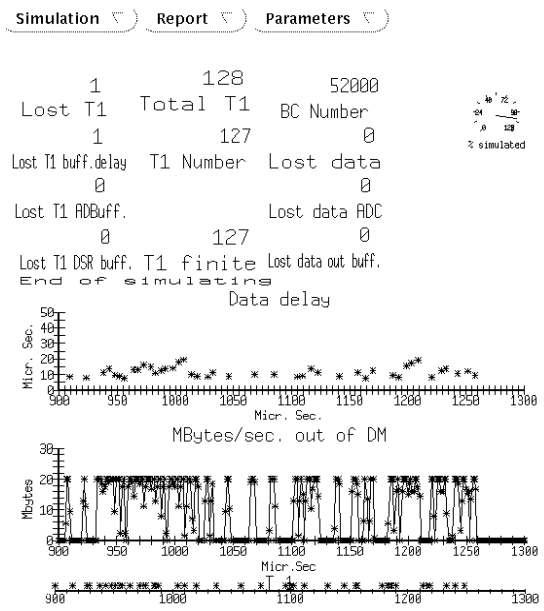
Figur D.5 viser grovt hvordan metoden *trigg1* virker. Ved oppstart av programmet vil en få opp flere grafiske vinduer som representerer tilstandene i systemet og statistikk over simuleringen. Oppløsningen i noen av grafene vil være avhengig av hvor ofte de oppdateres. Dette kan velges i *setup* menyen. Figur D.10 viser hovedvinduet. Øverst er det *buttons* til de forskjellige menyer og valg. Nedenfor kan en se hvor mange triggere og hvor mye data som er mistet i DM. Disse telles kontinuerlig. Nedenfor der er en tekstlinje som gir meldinger om tilstanden og hva som skjer i systemet. Helt nederst er det en tidsakse som viser når det er blitt gitt triggere. Triggerene inn til DM blir kontinuerlig merket av her. Det er også en graf som kontinuerlig viser dataraten ut av DM og en graf som viser hvor lang tid en hendelse har brukt før den er lest ut av DM. En kan her se at dataraten ut av DM er kuttet ved 20 Mbyte/s, dette pga. at utlesningshastigheten fra DM er satt til 2 BC/byte, eller 20 Mbyte/s ved $\Delta t = 25ns$. Dataraten blir regnet ut ved å summere opp antallet byte som er lest ut av DM fra forrige gang grafen ble oppdatert og dividere dette med tiden som har gått[4]. Den gjennomsnittlige dataraten som er brukt i oppgaven er funnet ved å dividere totalt antall byte som er lest ut av DM med den totale simuleringstiden.

Forsinkelsen en hendelse bruker er differansen i simuleringstid fra triggeren ble gitt til all data som tilhører hendelsen til er lest ut av utbufferet.

Figur D.11 viser hvor mange treff det var i FE brikkene til DM ved en spesifikk trigger. En kan se at det antallet treff som var i brikkene i x-retning er likt fordelt på brikkene i y-retning[5]. I virkeligheten skulle treffene fra x-retning fordeles tilfeldig til brikkene i y-retning. I

[4]Reporttiden

[5]Innbyrdes i hver brikke er treffene tilfeldig fordelt på stripene og ikke likt i x og y retning



Figur D.10: *Hovedvinduet til programmet.*

de simuleringene som er foretatt her vil ikke dette ha noen innvirkning. En bør allikevel huske dette ved bruk av datafilen som genereres av dataene ut fra DM.

Figur D.12 viser fordelingen av hvor mange hendelser det har vært i ADB. Til høyre vises hvor mange hendelser det er i ADB ved en spesifikk tid. Ved fullkjørt simulering listes ut den gjennomsnittlige fordelingen av hvor mange hendelser det har vært i bufferet.

Figur D.13 viser fordelingen av hvor mange hendelser det har vært i DSR bufferet. Til høyre vises hvor mange hendelser det er i DSR bufferet ved en spesifikk tid. Ved avsluttet simulering listes den gjennomsnittlige fordelingen av hvor mange hendelser det har vært i bufferet ut.

Figur D.14 viser hvor mange byte det er i utbufferet.

Figur D.15 viser fordelingen av hvor mange byte det er i datapakkene som leses ut av DM. Det gjøres en statistikk over antallet byte i hver pakke parallelt med oppdateringen av grafen. Grafene som viser gjennomsnittlig antall byte for hver pakke i oppgaven er basert på denne statistikken og ikke vindu D.15[6].

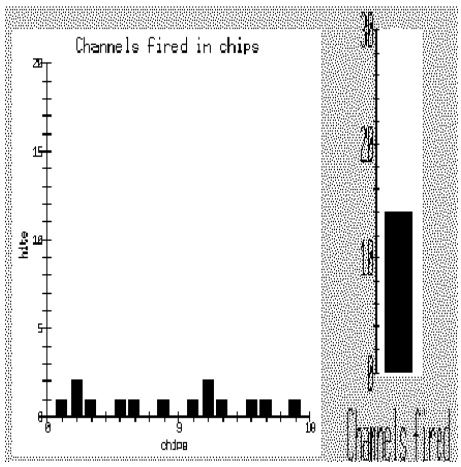
Figur D.16 viser fordelingen av hvor lang tid i μs det har tatt fra DM har mottatt en T1 til hele hendelsen er lest ut av DM. Grafen har en øvre grense på $200\mu s$. Statistikken som gjøres parallelt med at grafen oppdateres har ingen øvre grense.

Ved simulering av systemet kan en forandre de parametere som figur D.17 viser.

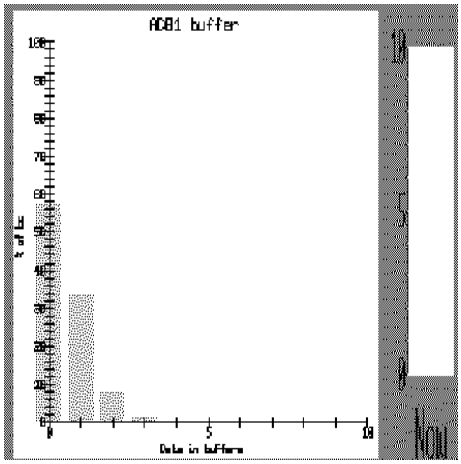
Dette er et grafisk vindu som kommer opp om en velger *setup* i menyen til hovedvinduet. Normalt er parameterene satt som i figur D.17 om annet ikke er nevnt. Parameterene en kan variere vil være de samme som i tillegg D.1, men en vil utenom disse også ha:

- *Utlesningstiden for DSR*, dette er tiden DSR bruker på å lese ut en kanal. Denne er normalt satt til 4 eller $t_{DSR} = 100ns$.

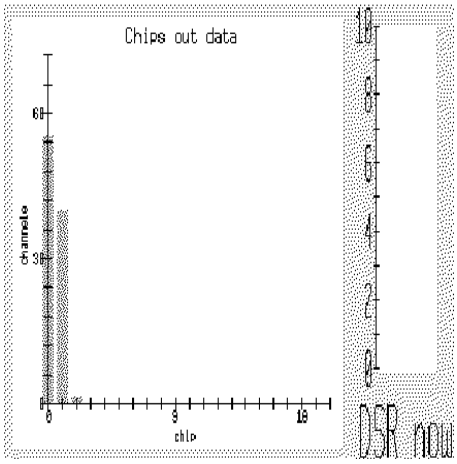
[6]Som ventet stemmer de bra overens



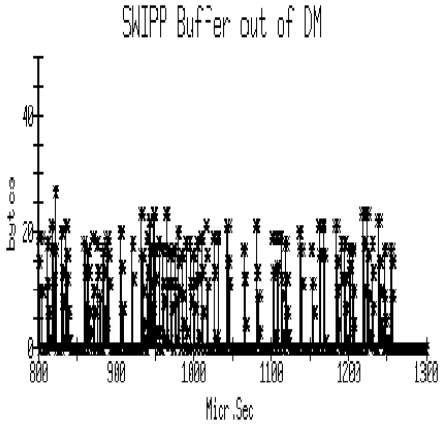
Figur D.11: *Grafisk vindu som viser hvor mange striper som er truffet av en partikkel for hver brikke og hvor mange treff det totalt var for hele DM.*



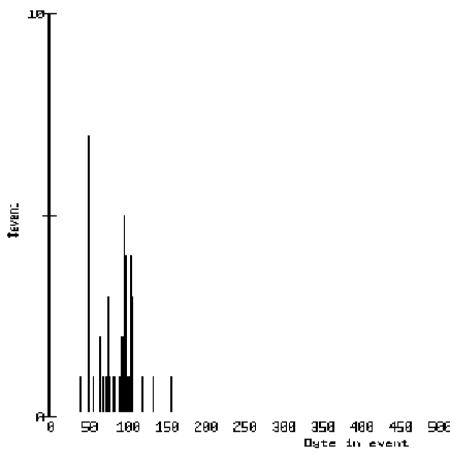
Figur D.12: *Grafisk vindu som viser fordelingen av hvor mange hendelser det har vært i ADB over foreløpig simulert tid.*



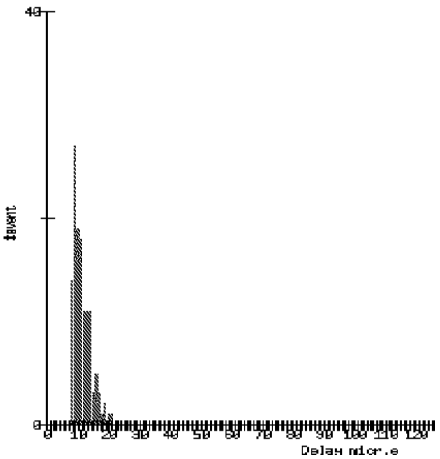
Figur D.13: *Grafisk vindu som viser fordelingen av hvor mange hendelser det har vært i DSR bufferet over foreløpig simulert tidsperiode.*



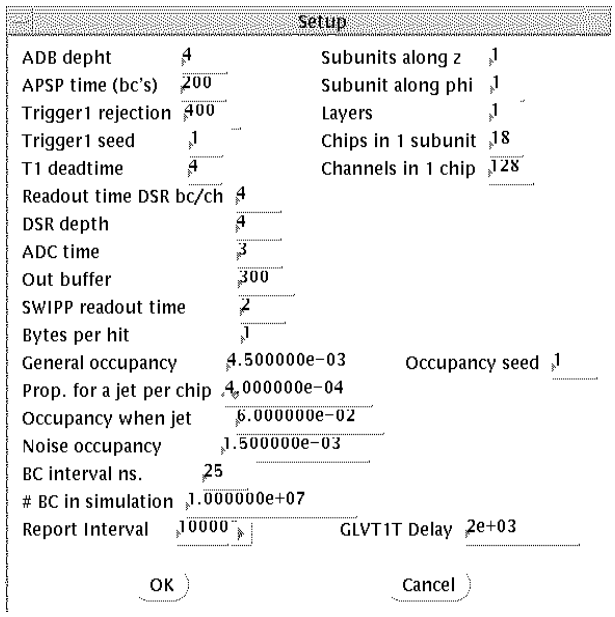
Figur D.14: *Vindu som viser hvor mange byte det er i utbufferet med simulert tid som abscisse.*



Figur D.15: *Vindu som viser fordelingen av hvor mange byte det har vært i hver pakke som er lest ut av DM.*



Figur D.16: *Vindu som viser fordelingen av hvor lang tid det har tatt å lese ut en hendelse av DM.*



Figur D.17: *Parametere som kan varieres ved simulering av hele DM.*

- *Prosesseringstiden til ADC*, dette er tiden ADC bruker for å konvertere en kanal med data. Om en vil unngå overflyt i ADC må må denne settes til en BC mindre en tiden DSR bruker. Dette pga. at ADC blir prioritert etter DSR i programmet. Denne er normalt satt til 3 som i praksis gir $t_{ADC} = 100ns$ eller en frekvens på 10 MHz.

- *Utbufferet*, dette er størrelsen i byte til bufferet ut av DM. Dette er normalt satt til 300 byte.

- *Utlesningstiden*, dette er tiden den optiske kabelen ut av DSR bruker på å lese ut en byte. Denne er normalt satt til 2 som gir $t_{out} = 50ns$ eller en utlesningshastighet på 20 Mbyte/s.

- *Byte pr. kanal*, dette er antall byte data som blir generert ut av ADC for hver verdi. Denne er normalt satt til 1 som tilsvarer en 8 bit ADC.

- *Generell occupancy*, dette er sannsynligheten for at det er signal i en kanal pga. treff av en partikkel. Denne er normalt satt til 0.0045 som tilsvarer en sannsynlighet på 0.45% for signal i en stripe pga. en partikkel.

- *Sannsynligheten for en jet i en brikke*. Denne er normalt satt til 0.0004 som tilsvarer en sannsynlighet på 0.04% for at det skal være en jet i en brikke.

- *Occupancy når det er en jet*, dette er den nye sannsynligheten for treff av en partikkel i en kanal når det er en jet i brikken. Denne vil da erstatte den generelle sannsynligheten for denne brikken. Denne er normalt satt til 0.06 som gir en ny sannsynlighet på 6% for signal i en stripe pga. en partikkel.

- *Støy.* Dette er sannsynligheten for at en får et signal i en kanal pga. en støypuls høyere en grensen til diskriminatoren. Denne er normalt satt til 0.0015 som tilsvarer en sannsynlighet på 0.15% som tilsvarer en grense pa 3σ som gitt i kapittel 3.3.

- *BC i simuleringen*, denne er normalt satt til 10 000 000.

10 000 000 BC tilsvarer $0.25s$ drift av LHC ved $\Delta t = 25ns$. Ved $a_{LVL1} = 400$ kan en forvente 25000 triggere. Ved *random seed* = 1 har simuleringene gitt 24825 triggere, som er 0.7% under forventningsverdien.

I tillegg til resultater som nevnt i appendix D.1 og vinduene og grafene som nevnt ovenfor vil en også få listet ut resultater som:

- *Det totale antallet triggere gitt.*

- *Totalt antall triggere som er mistet i DM.* Ved gjentatte simuleringer med støy=1.3%, $N_{DSR} = 5$ og $t_{DSR} = 75ns$, men ved forskjellig *random seed* har den totale inneffektiviteten for DM variert $\approx \pm 4\%$ av den gjennomsnittlige målingen.

- *Antallet triggere som har passert igjennom DM.*

- *Antallet triggere som er mistet pga. DSR bufferoverflyt.* Inneffektiviteten som er benyttet i oppgaven er funnet ved å dividere antallet triggere mistet med antallet triggere ut av ADB. Ved gjentatte simuleringer med støy=1.3%, $N_{DSR} = 5$ og $t_{DSR} = 75ns$, men ved forskjellig *random seed* har inneffektiviteten for DSR variert $\approx \pm 4\%$ av den gjennomsnittlige målingen.

- *Totalt antall byte ut av DM.*

- *Totalt antall byte mistet i DM.* Dette er antall byte mistet i DM ikke inkludert tap av hele hendelser, dvs. det er tap av deler av hendelser. Ved gjentatte simuleringer med støy=1.3%, $N_{DSR} = 5$ og $t_{DSR} = 75ns$, men ved forskjellig *random seed* har ineffektiviteten variert $\approx \pm 5\%$ av den gjennomsnittlige målingen.

- *Byte tapt i ADC.* Denne vil normalt være 0 pga. at ADC bruker samme klokke som DSR[7]. Dvs. parameteren til ADC er justert til å være minst like rask som DSR.

- *Byte tapt i utbufferet.* Dette er antallet byte tapt pga. overflyt i utbufferet. Programmet er laget slik at det alltid reserverer 20 byte til hodet og halen til pakkene som ligger i PE. Dvs. en får overflyt om det kommer en eller flere byte fra ADC når $S_{out} > N_{out} - 20byte$. Ved gjentatte simuleringer med støy=1.3%, $N_{DSR} = 5$ og t_{DSR}, men ved forskjellig *random seed* har ineffektiviteten variert $\approx \pm 5\%$ av den gjennomsnittlige målingen.

- *Den gjennomsnittlige tiden i μs en hendelse bruker igjennom DM.* Dette vil være tiden fra triggeren gis til all data som hører til den samme hendelse er lest ut av utbufferet. Variansen, max. og min. verdier oppgis også. Ved gjentatte simuleringer med støy=1.3%, $N_{DSR} = 5$ og t_{DSR}, men ved forskjellig *random seed* har den gjennomsnittlige forsinkelsen variert $\approx \pm 0.5\%$ av den gjennomsnittlige målingen.

- *Gjennomsnittlig antall signal for hver hendelse.* Dette vil være antallet signaler inkludert støy som har passert diskriminatorens grenseverdi. Variansen oppgis også.

[7] Om ikke dette var gjort ville en i verste tilfelle miste all data i ADC pga. at denne ligger i serie med DSR.

- *Gjennomsnittlig antall byte i en pakke.* Dette vil være den gjennomsnittlige størelsen i byte for en hendelse etter at den er pakket i utbufferet. Variansen, max. og min. oppgis også.
- *Gjennomsnittlig rate, ut av DM i Mbyte/s.* Denne er funnet ved å telle antall byte og dividere med tiden.
- *Fordelingen av hvor mange hendelser som har vært i DSR bufferet i %.*

D.3.1 Generering av datafil ut av DM.

Programmet generer en datafil til bruk for simuleringer av DAQ systemet etter DM. I [4] kan en se et eksempel på dette. Filen blir generert ut fra data føre pakking inn til PE. Dvs. evt. tap av data etter ADC er ikke fjernet. Filen kalles *data* og genereres i *METHOD PutDataIntoSWIPPbuffer* i *OBJECT ADCObj*. Filen skrives ut i *PROCEDURE FileWrite* i *OBJECT ReportObj* i report module[8].
Filens format er:
$T1nr$ *bcgen*:[*chipadr*]*bcnow*(*chann*)*bcnow*(*chann*)*bcnow*(*chann*)[*chipadr*]*bcnow*(*chann*) osv. :
der $T1nr$ er triggernummeret til hendelsen, *bcgen* er BC nummeret til hendelsen[9], *chipadr* er adressen til brikken, *bcnow* er BC nummeret[10] når ADC sender verdien til MC og *chann* er stripenummeret til verdien som blir lest ut. En linje vil da tilsvare en hendelse.

D.3.2 Program som genererer fysikken selv.

Modulen som er spesifikk for dette programmet er *PhysicsModule* som beskrevet i figur D.9. Først genererer programmet en evt. jet i FE brikken. Om det er en jet i brikken blir parameteren for sannsynligheten for treff i en stripe som er satt i *setup* menyen skiftet med sannsynligheten for treff i en stripe ved en jet. Sannsynligheten for signal pga. støy blir addert direkte til sannsynligheten for signal pga. partikkel treff. Programmet genererer så treff i silisium stripene ut fra en sannsynlighet som er gitt som beskrevet ovenfor. Etter at det er blitt generert treff for en retning blir det samme antallet treff fordelt i den andre retningen. Innad i hver brikke i den andre retningen blir treffene tilfeldig fordelt på stripene. Det gjøres også korreksjoner for evt. treff av to partikler i en stripe. Ved utlesning av FE brikken blir sidekanalene til de som har blitt truffet også lest ut.

D.3.3 Program som genererer fysikken fra MC simuleringer.

Fra [41] har en fysikksimuleringer av ATLAS innerdetektoren. Dette er to filer som ligger under:
/gaarder/modsim/datafil/jetlow1.hits
/gaarder/modsim/datafil/jethi1.hits
Jetlow1.hits er simuleringer med 2 minimum bias hendelser og jethi1.hits er simuleringer med 16 minimum bias hendelser. Begge er med $\Delta t = 25ns$ og $a_{LVL1} = 400$. Filene består av 256 jet hendelser som har blitt akseptert av nivå 1 triggeren og har formatet:

[8] Normalt er genereringen og utskrivingen av filen kommentert ut for å øke hastigheten på simuleringene.
[9] Dvs. når hendelsen skjedde.
[10] Dvs. hva tiden er når ADC er ferdig med å prosessere verdien fra denne kanalen.

EVENT ***n***		FORMAT('EVENT',i5)
ROIEP ***etamin etamax phimin phimax thresh***		FORMAT('ROIEP',4f6.2,i3)
LAYER *n phimin_bd zmin_bd n_phi_bd n_z_bd*		FORMAT('LAYER',i2,4i4)
n n n n n n	Number of hits per board	FORMAT(25i3)
n n n n n n	*n_phi_bd* lines of *n_z_bd* numbers	
n n n n n n		

med fortran formatet bakerst.

Filen består av en liste av hendelser, som hver består av 1 eller flere RoI. Hver RoI oppgir sin min. og max. η og ϕ koordinat og energi grensen for triggeren som genererte den, der 1=100GeV, 2=35GeV, 3=15GeV og 4=7GeV.

Ut fra energigrensen og koordinatene til RoI blir detektormodulene som er i det interessante området for hvert lag definert. Med hjørnet *phimin_bdzmin_b* og *rullet* ut til en *n_phi_bd*$*$*n_z_bd* matrise av moduler med n kanaler med signal i hver.

Simuleringen er gjort med en eldre versjon av innerdetektoren en den som er skissert i denne oppgaven. Det innerste laget er ved $r = 30cm$ og inkluderer totalt antall treff på begge sider av en dobbeltsidigdetektor.

Under /gaarder/modsim/datafil/conv er et program som konverterer formatet over til et format som kan brukes av *physics Module* beregnet for MC generert fysikk. Før kjøring av programmet *conv* må en kopierer en MC fil *jethi1.hits* over til en fil *input*. På enden av *input* filen må det være en EOF. Ved kjøring av programmet kan en velge hvilke lag en vil generer fysikk fra, antallet brikker[11], støyen som skal legges på og antallet hendelser. Lagene en genererer fysikk fra vil være lag 3 til 8 der lag 3 er ved *r=30 cm*. Det totale antallet hendelser vil være det en velger multiplisert med antallet hendelser i den filen som det er generert fra. Programmet velger tilfeldig en av detektormodulene i matrisen til *input* filen og fordeler antallet treff n til brikkene i den nye filen *fysikkdata.mdb*. Samtidig blir den valgte støy lagt jevnt på.

Formatet på den nye filen blir:

k k k k k k k k k k k k k k k k k k
k k k k k k k k k k k k k k k k k k
k k k k k k k k k k k k k k k k k k
k k k k k k k k k k k k k k k k k k

osv.

Hvor ***k*** er antallet treff i en brikke og en linje er en hendelse for en detektor modul. Her med 18 brikker.

I denne oppgaven er det brukt 12654[12] hendelser for å generere filen *fysikkdata.mdb* som tilsvarer 5061600 BC om annet ikke er nevnt. Om en kjører en simulering av DM med mere enn 12654 triggere vil programmet starte å lese fra begynnelsen av *fysikkdata.mdb* igjen. I denne oppgaven er det brukt 10000000 BC ved simulering av DM om annet ikke er nevnt.

Physics Module fordeler tilfeldig det antallet treff k som er gitt i *fysikkdata.mdb* til hver enkelt brikke. Det vil ikke ha noen innvirkning på simuleringen å variere parameterene som har med fysikkgenereringen i *setup* vinduet ved kjøring av program som genererer fysikk fra MC simuleringer.

[11] Partall
[12] $57 \cdot 222$

D.3.4 Alternativ 1.

Prosessene er uavhengige og tap skjer på samme sted som ineffektivitet oppstår. Her behandler programmet DM som beskrevet i alternativ 1 i kapittel 3.4. Modulen som er forandret for dette alternativet er *timekernel module*. Modulen finnes under /gaarder/modsim/APSPandMUX/alt1/ .

D.3.5 Alternativ 2.

APSP starter ikke med å prosessere en ny hendelse før det er en helt ledig bufferplass i DSR bufferet. Her behandler programmet DM som beskrevet i alternativ 2 i kapittel 3.4. Modulen som er forandret for dette alternativet er *timekernel module*. Modulen finnes under /gaarder/modsim/APSPandMUX/alt2/ .

D.3.6 Alternativ 3.

Prosessene starter ikke å prosessere om det kan oppstå ineffektivitet i prosesser som ligger etter. Her behandler programmet DM som beskrevet i alternativ 3 i kapittel 3.4. Det er ikke laget egne simuleringsprogrammer for dette alternativet. Grunnen til dette er at en i praksis kan kreve at en ikke skal miste data etter ADC[13] og dermed kan en simulere dette systemet som i appendix D.3.5 og dermed bruke samme moduler.

D.3.7 Alternativ 4.

Her behandler programmet DM som beskrevet i alternativ 2 i kapittel 3.4 bortsett fra at isteden for å vente med å starte å prosessere data så starter APSP, men holder på data til DSR gir signal om at den er klar. Modulen som er forandret for dette alternativet er *timekernel module*. Modulen finnes under /gaarder/modsim/APSPandMUX/alt4/ .

[13] tapet vil evt. være så lite at det ikke har noen innvirkning på dataflyten i DM

Litteratur

[1] *LHC detector timing distribution, Muon Tracking Workshop*, 4-5 July 1991.

[2] R.J.Hawkings A.R.Weidberg. *TRD Occupancies.* CERN, 8 April 1993. ATLAS Internal Note, INDET-NO-026.

[3] Frode B.Nilsen. *Towards the Large Hadron Collider at CERN.* IfI UiO Norway, 30 August 1992.

[4] Frode B.Nilsen. *Application of a switched interconnection concept for instrumentation at a high-energy physics experiment.* IfI UiO Norway, 9 May 1993. Thesis for the degree Candidatus Scientarium.

[5] CACI Products Company. *MODSIM II^R, The Language for Object-Oriented Programming*, January 1992. User's Manual.

[6] CACI Products Company. *MODSIM II^R, The Language for Object-Oriented Programming*, January 1992. Tutorial.

[7] CACI Products Company. *MODSIM II^R, The Language for Object-Oriented Programming*, January 1992. Reference Manual.

[8] CACI Products Company. *SIMGRAPHICS II^{TM}*, January 1992. Reference Manual.

[9] Paolo Cattaneo. *Capasitance calculation in a microstrip detector and its applications to signal processing*, 22 January 1990. Nuclear Instruments and Methods in Physics Research A295 (1990) 207-218.

[10] J.Soffer C.Bourrely and T.T Wu. *Preprint*, 1990. preprint CERN-TH. 5862.

[11] CERN. *Summary of RD20 Electronic Frontend Meeting*, 9 December 1992.

[12] CERN. *Transparency copies of the ATLAS Plenary Meeting*, 8 April 1993.

[13] CERN, Geneva Switzerland. *RD 20 , Development of Hihh Resolution Silicon Strip Detectors for Experiments High Luminosity at LHC*, 13 May 1992. CERN/DRDC 92-28.

[14] LHCC/I 2 CERN/LHCC/92-4. *ATLAS, Letter of intent for a General-Purpose pp Experiment at the LHC at CERN.* CERN, Geneva Switzerland, 1 October 1992.

[15] RD3 Collaboration. *R&D for a liquid argon preshower*, 1992. CERN/DRDC/92-40.

[16] The ATLAS Collaboration. *Progress Report on ATLAS Milestones.* CERN, Geneva Switzerland, 15 October 1993. CERN/LHCC/93-51.

[17] The ATLAS Colloboration. *The ATLAS Inner Detector.* CERN, Geneva Switzerland, 31 March 1993. CERN/LHCC/93-24.

[18] D.Denergi. *Proceedings of the Aachen Workshop.* Vol 1,p57.

[19] R.Brenner H.von der Lippe J.Michel E.Nygård T.Ødegaard N.A.Smith P.Weilhammer K.Yoshioka. *Design and Performance of an Analog Delay and Buffer chip for use with silicon strip detectors at LHC.* European Organization for Nuclear Research, 13 May 1993. CERN-PPE/93-.

[20] D.R.Cox and W.L.Smith. *Queues.* Methuen's Statistical Monographs, 1967.

[21] ECFA/CERN. *Towards the LHC Experimental*, March 1992. Evian-les-Bains, Franc.

[22] G.Jarlskog D.Rein (editors). *Large Hadron Collider Workshop, Aachen.* European Committee for Future Accelerators, 4-9 October 1990. CERN 90-10/ECFA 90-133, Vol.1.

[23] T.Høgh et al. *A Low Noise Low Power CMOS Charge Sensitive Amplifer for Silicon Detectors at LHC.* Presented at the Sixth European Symposium on Semiconductor Detectors, Milian, 1992.

[24] B.Aubert et al.(RD3 Collaboration). *Liquid Argon Calorimetry with LHC-Performance Specifications.* CERN/DRDC/90-31, 1990.

[25] F.Abe et.al. *Evidence for Top Quark Production in $\bar{p}p$ Collisions at $\sqrt{s} = 1.8TeV$.* CDF Collaboration, 1994. Preprint, FERMILAB-PUB-94/097-E/CDF/PUB/TOP/PUBLIC/2561.

[26] M.Caccia et.al. *A Si strip detector with integrated coupling capasitors*, 27 April 1987. Nuclear Instruments and Methods in Physics Research A260 (1987) 124-131.

[27] Lorenzo Foa. *Collider physics part II, summer schol.* CERN, July 1993.

[28] G.Hall. *RD20 pipeline buffer depth and ineffiency*, January 1993. RD20/TN/12.

[29] Donald H.Perkins. *Introduction to High Energy Physics.* Third Edition.

[30] J.F.Genat. *Summary of RD20 Electronic Frontend Meeting on 9 December 1992.* CERN, 9th-10th December 1992. J.F.Genat presented his idea of interstrip inteligence.

[31] J.Gareyte. *The CERN Large Hadron Collider, summer school.* CERN, 6th August 1992.

[32] O.Søråsen F.B.Nilsen E.Nygård J.M.Østby. *A switched optical interconnection network used for large scale instrumentation* . Department of Informatics UoO, 7th Oct 1992. ATLAS Inner Detector Electronics Meeting CERN.

[33] Helge Langehaug and Torbjørn Gjemdal. *High layer communication protocol for an interconnected large scale instrumentation application.* University of Oslo, 14 August 1994. Cand. Scient Thesis.

[34] Einar Nygård. *ATLAS Inner Detector Panel meeting-CERN.* RD20, 10 January. Front-End Electronics (FEE).

[35] P.Jenni. *LHC physics and it's experimental aspects.* CERN, Geneva Switzerland, 22 may 1990.

[36] P.Weilhammer. *Feasibility of Si strip detectors with capacitive charge division and deconvolution front-end electronics in the ATLAS Si tracker.* CERN, Geneva Switzerland. RD20 draft.

[37] P.Weilhammer. *Si strip in the ATLAS id with capacitive charge division at high luminosity.* CERN, Geneva Switzerland, Jan 1994. IDSG Meeting.

[38] P.weilhammer. *Spatial resolution with Si strip detectors using charge division.* CERN, Geneva Switzerland, 1994. RD20 Meeting.

[39] S.Gadomski G.Hall T.Høgh P.Jalocha E.Nygård P.Weilhammer. *The deconvolution method of fast pulse shaping at hadron colliders.* European Organization for Nuclear Research, 31 Januar 1992. CERN-PPE/92-24.

[40] RD2 and RD11 Collaborations. *A study of a second level track trigger for ATLAS.* Preprint University of Oxford. OUNP-93-07.

[41] R.Hawkings. *RoI data file.* University of Oxford, 15 Nov 1993. hawkings@v1.ph.ox.ac.uk.

[42] P.E.Gaarder T.Ødegaard F.B.Nilsen S.Stapnes E.Nygaard R.Skinderviken. *A description of a detector module for the SITV detector in ATLAS.* A colloboration between the Institute of physics UoO, the Institute of Informatics UoO and SI Oslo, Juli 1993. Intern Draft.

[43] R.Turchetta. *Spatial resolution of silicon microstrip detectors.* LEPSI, ULP/CNRS, Strasbourg, France, 19 March 1993. Nuclear Instruments and Methods in Physics Research A335 (1993) 44-58.

[44] Trond S.Høgh. *Event Time and Pulse Height Estimation in Silicon Detectors Using Deconvolution Techniques.* Norwegian Institute of Technology-NTH, April 1992. Diploma Thesis.

[45] Roar Skinderviken. *Title unknown.* University of Oslo, 1994. Master's Thesis.

[46] The American Physical Society. *Physical Review D*, 1 August 1994.

[47] C.Bourrely J.Soffer T.T.Wu. *Z. Phys.*, 1988. C 37 369.

[48] G.Brianti W.Scandale. *Cern's Large Hadron Collider: a new tool for investigating the microcosm*, 1992. Particle World. Vol3, No.2, p. 101-107.