DIE ONTWIKKELING EN EVALUERING VAN 'N 8DPSK TRELLISKODE DATASENDER EN ONTVANGER

J. H. RAATH

DIE ONTWIKKELING EN EVALUERING VAN 'n 8DPSK TRELLISKODE DATASENDER EN ONTVANGER.

deur

Johannes Hendrikus Raath

Verhandeling ingelewer ter voldoening
aan die vereistes vir die

Magister Technologiae

Ingenieurswese : Elektries

in die
Fakulteit Ingenieurswese
aan die
Technikon Vrystaat

Julie 1996

Studieleier : Mnr. G.D. Jordaan

Central University of Technology, Free State

BEDANKINGS

Ek wil graag die volgende persone en instansies bedank vir u hulp en ondersteuning ter voltooiing van die projek :

My eggenote, Tania, vir haar hulp en ondersteuning gedurende die tydperk.

Die Stigting vir Navorsing en Ontwikkeling (SNO) vir finansiële bystand.

Technikon Vrystaat vir die geleentheid aan my gebied om die projek te voltooi.

My kollegas vir u belangstelling en ondersteuning, spesifiek Pierre, wat altyd gewillig was om te help met enige probleem.

Laastens maar die meeste, wil ek my studieleier, mnr. G.D. Jordaan, bedank vir sy onbaatsugtige hulp en leiding waarsonder dit onmoontlik sou gewees het om hierdie verhandeling te voltooi.

OPSOMMING

'n Groot toename in die aanvraag na datatransmissie in verskeie vorms het aanleiding daartoe gegee dat die tweede helfte van die twintigste eeu as die evolusie era van digitale kommunikasie beskou word.

Hierdie verhandeling ondersoek die gebruik van trelliskodemodulasie met differensiële fase-skuifsleuteling as modulasietegniek in 'n halfdupleks modem konfigurasie. Die sender is oorspronklik ontwerp om teen 'n datatempo van 3200 bisse per sekonde te funksioneer, maar a.g.v. probleme t.o.v. prosesseringstyd in die ontvanger is die datatempo van die hele stelsel verlaag na 1066 bisse per sekonde. Die bandwydte van die stelsel is ook ooreenkomstig afgeskaal.

Trelliskodemodulasie is gekies as modulasietegniek omdat dit t.o.v. gewone fase-skuifsleuteling (PSK), foutregstellende eienskappe plus al die gepaardgaande PSK eienskappe besit. Hierdie voordeel is moontlik sonder enige verlaging in datatempo of verhoging in bandwydte aangesien trelliskodemodulasie data-enkodering en -modulasie as 'n enkele tegniek implementeer. Dekodering van trelliskode by die ontvanger is m.b.v. 'n Viterbi dekodeerder gedoen om sodoende maksimum voordeel uit die trelliskodemodulasieproses te put.

Die ontwikkeling van die stelsel is voorafgegaan deur wiskundige simulasies van die sender en ontvanger, wat grotendeels bygedra het tot die identifisering van moontlike probleme en die voorkoming daarvan. Die wiskundige simulasies het 'n onmisbare deel bygedra tot die suksesvolle implementering van die stelsel in digitale argitektuur.

Die sender en ontvanger is gebaseer op die gebruik van die Peralex SIG56 kaart en die eksperimentele werk was hoofsaaklik sagteware georiënteerd. Die sender en ontvanger het elk van 'n SIG56 kaart gebruik gemaak waar DSP56001 prosesseerders al die seinprosessering uitgevoer het en TLC32044 omskakelaars gebruik is vir al die digitalisering- en filteringsprosesse.

Die sender is geëvalueer aan die hand van die gestuurde sein se frekwensiespektrum en konstellasie. Dit is gevind dat die sender redelik stabiel is t.o.v. die generering van die

verlangde faseveranderinge en dat die frekwensiespektrum van die uitsetsein binne die bepaalde frekwensiestrek val.

Metings van die foutbistempo teenoor seinruisverhoudings is geneem ter evaluering van die prestasievermoë van die ontvanger. Die ontvanger is eerstens geëvalueer deur slegs die prestasievermoë van die DPSK modulasieproses te meet. Daarna is die hele ontvanger, met Viterbi dekodering ingesluit, geëvalueer. In beide gevalle is gevind dat dit baie na-aan aan die teoretiese waardes presteer (tipies 2dB tot 3dB swakker). Teen 'n seinruis-energieverhouding van +/-15dB het die stelsel opgehou om bevredigend te funksioneer a.g.v. die sinchronisasiedetektor wat nie meer sinchronisasie kon bewerkstellig nie.

Kundigheid t.o.v. veral die volgende aspekte is tydens die uitvoering van die projek bekom :

- Trelliskodemodulasie en die gebruik van konvolusiekodes in die algemeen.
- Gebruik van DSP in 'n persoonlike rekenaar omgewing. Die integrering van die sagteware wat op die persoonlike rekenaar en die SIG56 kaart uitgevoer word was insiggewend.
- Die implementering van die Viterbi dekoderingsalgoritme in DSP.
- Simboolsinchronisasie en die gebruik van 'n korrelasiedetektor.

SUMMARY

A large increase in the demand for data transmission in its various forms has led to the viewpoint that the second half of the twentieth century is regarded as the era of evolution for digital communication.

This thesis explores the use of the trellis code modulation with differential phase-shift keying as modulation technique in a half-duplex modem configuration. The transmitter was initially designed to operate at a data rate of 3200 bits per second, but as a result of problems with processing time in the receiver the data rate of the whole system was down-graded to 1066 bits per second. The bandwidth of the system was also scaled down accordingly.

Trellis code modulation was chosen for the modulation process because of its error correcting and other features with respect to phase-shift keying (PSK). This advantage is obtained without any loss of data rate, or increase in bandwidth requirements because with trellis code modulation data-encoding and modulation are a single technique. Decoding of the trellis code at the receiver is accomplished with a Viterbi decoder to take maximum advantage of the process of trellis code modulation.

Development of the system was preceded by mathematical simulations of the transmitter and receiver, and this largely contributed to the determination and avoidance of problems. Mathematical simulations played a vital role in the ultimate implementation of the system in digital architecture.

The transmitter and receiver were based on the use of the Peralex SIG56 card with the Motorola DSP 56001 processor while the TLC32044 converter was used for all the digitising and filtering processes.

The transmitter was evaluated in terms of the frequency spectrum and constellation of the transmitted signal. It was found that the transmitter was reasonably stable with respect to generation of the required phase shifts and the frequency spectrum of the output signal fell within the predicted frequency range.

Measurements were made of the bit error rate with regard to the signal-to-noise ratio to determine the performance of the receiver. The receiver was first evaluated by determining its performance with the DSPK modulation process. After this the entire receiver was evaluated, including the Viterbi decoding process. In both cases it was found that performance of the process/techniques were very close to the theoretical values (typically 2dB to 3dB poorer).

Satisfactory operation ceased at a signal to noise energy ratio of ± 15dB as a result of the synchronous detector not being able to maintain synchronisation.

During execution of the project insight was gained with respect to the following:

- Trellis code modulation and the use of convolution codes in general.
- Use of DSP in a personal computer environment. The integration of software used in the personal computer and the DSP 56001 was illuminating.
- The implementation of the Viterbi decoding algorithm in DSP.
- Symbol synchronisation and the use of a correlation detector.

Central University of Technology, Free State

INHOUD

LYS VAN FIGURE

Hoofstuk 2

Hoofstuk 3

Hoofstuk 4

Hoofstuk 5

Hoofstuk 6

LYS VAN TABELLE

Hoofstuk 2

Hoofstuk 4

Hoofstuk 5

Hoofstuk 6

1. INLEIDING

1.1 PROBLEEMSTELLING

'n Groot toename in die aanvraag na datatransmissie in enige vorm - van rekenaardatabanke tot afgeleë dataterminale, vir 'n verskeidenheid van toepassings met 'n toenemende behoefte aan akkuraatheid - het aanleiding daartoe gegee dat die tweede helfte van die twintigste eeu as die evolusie era van digitale kommunikasie beskou word *[27, p. 3].*

Onderontwikkelde gebiede beskik gewoonlik nie oor hoëspoed digitale kommunikasielyne nie en daarom moet datakommunikasie oor bestaande analoog spraakkanaal telefoonlyne met beperkte bandwydte plaasvind. Om kommunikasie in hierdie gebiede te bevorder is 'n kommunikasiestelsel/modem nodig wat aanpas by die betrokke kanaalkarakteristieke en voldoen aan onderstaande tweeledige probleem waaraan digitale kommunikasie onderwerp word :

1. Sover moontlik 'n vermindering in die aantal bisse wat oor 'n kommunikasiekanaal gestuur moet word - sonder 'n afname in die voorafbepaalde standaard van getrouheid - bewerkstellig.
2. Die gestuurde data moet, ten spyte van hoë kanaalruis, sover moontlik korrek ontvang word.

Om genoemde probleme in 'n kommunikasiestelsel aan te spreek handel hierdie projek oor die ontwikkeling van 'n datasender wat van Trelliskodemodulasie met 8DPSK gebruik maak. 'n Ontvanger wat gebruik maak van 'n korrelasiedetektor en Viterbi dekodering is vir genoemde sender ontwikkel.

Beide die sender en ontvanger maak van digitale seinprosessering (DSP) gebruik. Die belangrikste rede vir die gebruik hiervan is die potensiële aanpasbaarheid by veranderde behoeftes en tegnologiese ontwikkeling.

Die kommunikasiestelsel soos ontwikkel is in terme van die bisfouttempo onder gedefinieerde seinruisverhoudings getoets ten einde die prestasievermoë van verskillende tegnieke en prosesse te evalueer.

1.2 DOELWIT VAN DIE ONDERSOEK

Die doel met hierdie projek was :

1. Die bestudering en evaluering van trelliskodemodulasie en Viterbi dekodering in 'n datakommunikasiestelsel wat geïmplementeer is in digitale argitektuur, vir werking oor 'n spraakkanaal telefoonlyn teen 3200 bisse per sekonde.
2. Deur uitvoering van die projek gespesialiseerde kennis aangaande die toepassing van DSP ontwerptegnieke in te win, en die haalbaarheid van gespesifiseerde spesifikasies (sien paragraaf 1.3) te bepaal deur gebruikmaking van hierdie tegniek.

1.3 HIPOTESE

Die ondersoek was gerig op die toetsing van die volgende hipotese :

'n Bidireksionele halfdupleks modem wat van DSP tegnieke gebruik maak kan ontwikkel word vir die stuur en ontvangs van 'n 3200 bis per sekonde 8DPSK Trelliskode sein met Viterbi foutregstelling.

1.4 AFBAKENING VAN DIE STUDIETERREIN

Ter voortbouing op bestaande projekte aan die Technikon is besluit op 'n 2/3 konvolusie enkodeerder vir gebruik in die sender.

Differensiële fase-skuifsleuteling is as modulasieproses gebruik omdat dit soortgelyke eienskappe as PSK besit t.o.v. bandwydte en omhulling - wat ruisimmuniteit bevorder.

In die ontvanger is besluit op 'n korrelasiedetektor - wat in die literatuur as die optimale demodulator beskou word *[18, p. 236]* - om die veranderings in fase waar te neem soos deur die sender gelewer.

Sagte besluitneming Viterbi dekodering is gebruik vir foutregstelling in die ontvanger ten einde maksimum voordeel uit die trelliskodemodulasieproses te put.

1.5 NAVORSINGSMETODE

Die eksperimentele werk is in die volgende fases uitgevoer,

- Wiskundige simulasie van die enkodering- en modulasieprosesse in die sender asook die invloed van ruis op die uitsetsein.
- Implementering van die sender in digitale argitektuur (SIG56 kaart met DSP56001 prosesseerder).
- Wiskundige simulasie van 'n moontlike sinchronisasiestelsel, asook die demodulasie en dekoderingsprosesse in die ontvanger.
- Implementering van die ontvanger in digitale argitektuur (SIG56 kaart met DSP56001 prosesseerder).
- Evaluering van die sender in sy geheel.
- Evaluering van die ontvanger t.o.v. die modulasieproses met en sonder Viterbi dekodering.

1.6 PROBLEME ONDERVIND

Die volgende probleme is tydens die uitvoering van die projek ondervind.

- Sinchronisasie by die ontvanger. Na verskeie pogings is 'n suksesvolle sinchronisasiedetektor ontwikkel.
- 'n Tekort aan monsters per simbool by die ontvanger vir gebruik tydens die demodulasieproses. Die probleem is opgelos deur addisionele monsters in die ontvanger te genereer.

- Die bepaling van die Euklidiese afstande van ontvangde fases is oorbodig gemaak deur van 'n vorm van korrelasie gebruik te maak.
- 'n Tekort aan prosesseringstyd in die ontvanger om die beplande datatempo te realiseer - a.g.v. die Viterbi dekoderingsproses wat te veel tyd in beslag neem - kon nie oorbrug word nie. Derhalwe is die datatempo na 'n laer waarde aangepas.

2. DIGITALE KOMMUNIKASIE

2.1 INLEIDING

Die doel van 'n elektroniese kommunikasiestelsel is die oordra van 'n informasiedraende sein vanaf 'n bron tot by 'n bestemming oor 'n kommunikasiekanaal. Hierdie stelsel kan analoog of digitaal wees. In 'n analoog stelsel varieer die informasiedraende sein se amplitude in tyd en word dit direk gebruik om een of ander karakteristiek van 'n sinusgolf te verander. Daarinteen word die informasiedraende sein in 'n digitale kommunikasiestelsel geprosesseer sodat dit deur 'n reeks diskrete waardes van 'n sinusgolf voorgestel kan word.

Fig.2.1 toon 'n blokdiagram van 'n digitale kommunikasiestelsel wat uit drie basiese seinprosesseringsprosesse bestaan nl: bronkodering, kanaalkodering en modulasie. Die bron is digitaal van aard.

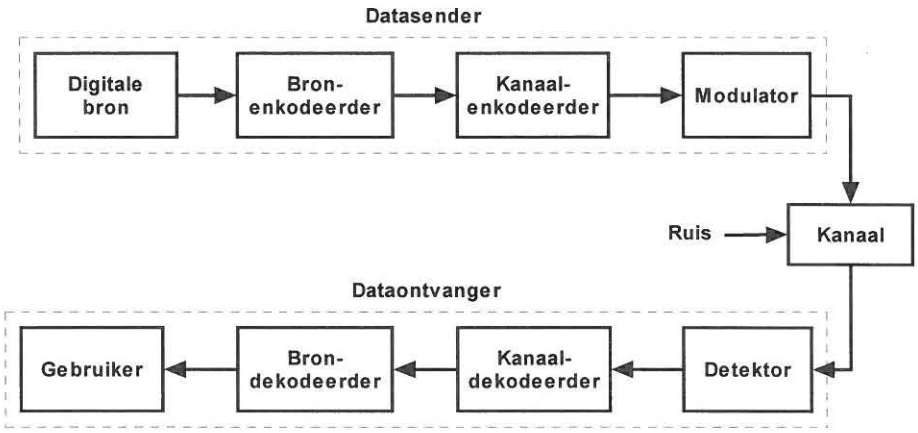


Fig. 2.1 : Basiese digitale kommunikasiestelsel.

In die bron-enkodeerder, as deel van die datasender, word die digitale sein wat in die bron gegenereer is na 'n ander digitale formaat omgeskakel met die doel om versending van oortollige bisse te voorkom, om sodoende 'n meer effektiewe weergawe van die bron uitset te lewer. Die brondekodeerder, as deel van die ontvanger, doen die omgekeerde en lewer dus 'n weergawe van die oorspronklike digitale bron inset. Die primêre voordeel verkry uit bronkodering is 'n verlaging in die vereiste bandwydte.

In die sender is die kanaal-enkodeerder se doel om die bron ge-enkodeerde digitale sein te verander in 'n kanaal insetsein deur die sein op 'n voorgeskrewe manier te prosesseer. Die kanaaldekodeerder in die ontvanger hervorm die sein om die oorspronklike ge-enkodeerde sein so akkuraat moontlik te herkonstrueer. Die gekombineerde rol van die kanaal-enkodeerder en -dekodeerder is om betroubare kommunikasie oor 'n kanaal met ruis te verseker deur oortolligheid tot die sein te voeg wat foutregstelling moontlik maak by die ontvanger.

Modulasie word gedoen om die frekwensiespektrum van die basisbandsein te transformeer na die deurlaatfrekwensieband van die kanaal. Die proses vind plaas deurdat die modulator sekere karakteristieke van 'n sinus draaggolf verander in ooreenstemming met die kanaal-enkodeerder uitset. Demodulasie word in die detektor gedoen en produseer 'n sein wat die variasies van die kanaal-enkodeerder uitset navolg, behalwe vir die afwykings a.g.v. ruis wat tydens transmissie oor die kanaal geïnduseer word.

Gewoonlik word enkodering en modulasie afsonderlik gedoen. Die toevoeging van oortollige simbole vir foutregstelling deur die kanaal-enkodeerder veroorsaak dan 'n verhoging in die benodigde transmissie bandwydte. In sommige toepassings word enkodering en modulasie saam gedoen en op so 'n manier dat geen verhoging in transmissie bandwydte nodig is nie. Trelliskodemodulasie, wat in hierdie projek gebruik word, is 'n voorbeeld van so 'n stelsel *[10, p. 5 & 421]*.

2.2 KOMMUNIKASIE IN 'n BANDDEURLAATKANAAL

Die transmissiekanaal is die belangrikste deel van 'n kommunikasiestelsel, want dit is die medium wat die inligting moet oordra en daarsonder is kommunikasie onmoontlik. Die kanaal beperk die oordra van inligting deur die gestuurde sein te verwring t.o.v. amplitude en fase, a.g.v. onder andere beperkte bandwydte. Die betrokke kommunikasiestelsel moet die geskikste transmissiemetode of modulasieproses gebruik om by die tipe kanaal se beperkinge aan te pas.

Wanneer digitale data of 'n basisbandsein oor 'n kanaal soos die telefoonkanaal, wat oorspronklik vir analoog spraak ontwerp is, gestuur word, moet die digitale pulse

aangepas word om ander karakteristieke aan te neem. Hierdie verandering is nodig omdat 'n vierkantsgolf se harmonieke en gelykstroomkomponente verswak of verwyder word deur die kanaal *[9, p. 34]*. Dit is dan onmoontlik om hierdie verwringde vierkantsgolf by die ontvanger te herwin. Die gebruik van transmissiebruê en lynaanpassingstransformators in telefoonstelsels is die oorsaak van hierdie verwringing. Die pulsspektrum van die gestuurde sein moet dus gekonsentreer wees binne die spesifieke banddeurlaatfrekwensiestrek van die kanaal, en om hierdie rede word sinusgolwe as digitale simbole gebruik. 'n Frekwensiestrek van 300Hz tot 3400Hz is tipies vir 'n analoog spraakkanaal *[4, p. 14]*.

Die hoeveelheid inligting wat in 'n sekere tydsinterval oor 'n kanaal gestuur kan word staan bekend as die kanaalkapasiteit. Die Hartley-Shannon teorie druk die kanaalkapasiteit (C) as volg uit:

$$C = BW \log_2(1 + \frac{S}{N}) \qquad (2.1)$$

waar BW die kanaalbandwydte en S/N die minimum aanvaarbare seinruisverhouding is *[25, p. 3]*. Hierdie is slegs 'n teoretiese limiet waarna gestreef word, maar wat nooit in die praktyk bereik word nie.

2.2.1 Modulasieprosesse

Soos alreeds genoem word daar van sinusgolwe gebruik gemaak om inligting oor 'n kanaal met beperkte bandwydte te stuur. Die sinusgolwe se karakteristieke, nl: amplitude, frekwensie en fase, word gevarieer in ooreenstemming met die veranderings in die basisbandsein om sodoende hierdie inligting effektief oor die banddeurlaatkanaal te dra. Die variasie van genoemde karakteristieke staan onderskeidelik bekend as: *Amplitudemodulasie (AM), Frekwensiemodulasie (FM)* en *Fasemodulasie (PM).*

Bogenoemde AM, FM en PM seine kan op twee afsonderlike maniere verkry word. In die een tegniek, wat algemeen in analoog kommunikasiestelsels voorkom, word die gemoduleerde sein gevorm deur die spesifieke karakteristiek (amplitude, frekwensie of fase) van die sinus draaggolf direk met die amplitude van die basisbandsein te moduleer.

In die tweede tegniek nl. digitale modulasie, word die draaggolfsein direk t.o.v. die binêre data wat oorgedra moet word gemoduleer en benodig nie 'n analoog basisbandsein vir die modulasieproses nie. Die AM-variasie staan bekend as *Amplitude-skuifsleuteling (ASK),* FM as *Frekwensie-skuifsleuteling (FSK)* en PM as *Fase-skuifsleuteling (PSK).*

Indien hierdie prosesse 'n binêre datastring van 1010 moet oordra, kan figure 2.2 tot 2.4 as elk van bogenoemde tipes modulasie se uitsetseine voorgehou word. Langs elke voorbeeld is 'n vektorvoorstelling van die betrokke uitsetsein se moontlike variasies wat bekend staan as 'n konstellasiediagram. Posisie a dui telkens op 'n logika 1 inset en posisie b op 'n logika 0 inset.

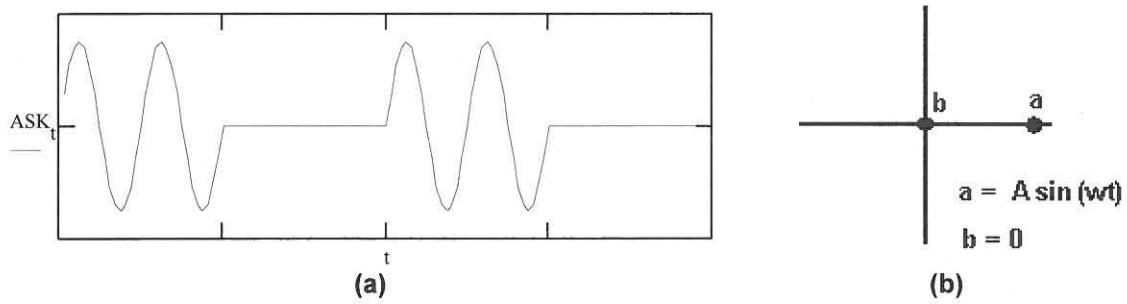


Fig. 2.2 : (a) ASK golfvorm en (b) konstellasie

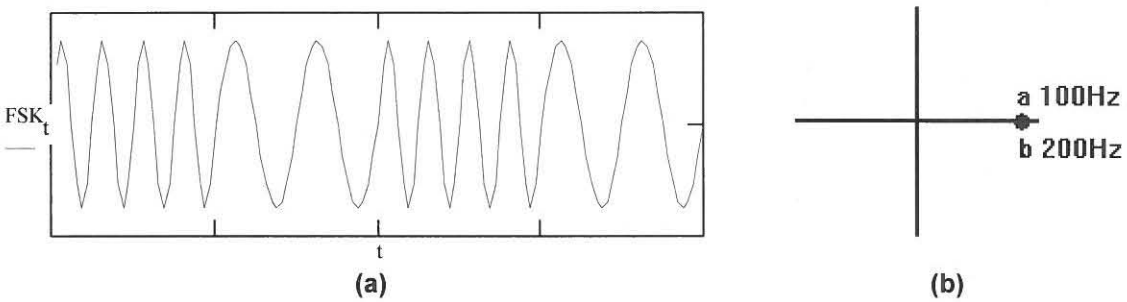


Fig. 2.3 : (a) FSK golfvorm en (b) konstellasie

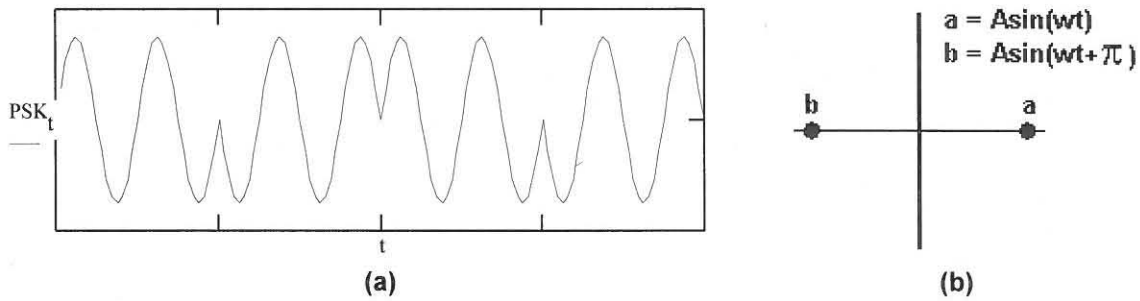


Fig. 2.4 : (a) PSK golfvorm en (b) konstellasie

2.2.2 Bandwydte effektiwiteit

Binêre sleuteling stelsels kan slegs een bis informasie per gemoduleerde sinuspuls of simbool oordra. Om hierdie bistempo te verhoog kan die sinuspulswydte vernou word, maar dit gaan 'n verhoging in bandwydte tot gevolg hê - wat geen verbetering in die bandwydte effektiwiteit lewer nie. 'n Ander opsie om die bandwydte effektiwiteit te verbeter is om elke gemoduleerde sinuspuls of simbool meer as een bis informasie te laat verteenwoordig. Dit het *Veelvlak-ASK (MASK), -FSK (MFSK) en -PSK (MPSK)* tot gevolg waar die databisse saamgevoeg word in groepe van $\log_2 M$. As M = 4 word die bisse saamgevoeg in groepe van twee en as M = 8 word die bisse in groepe van drie saamgevoeg. Elke bisgroep verteenwoordig dan 'n unieke waarde of vlak.

'n Eenvoudige voorbeeld van veelvlak transmissie is 'n MASK sein met M = 4 (4ASK). Die sein kan oënskynlik 'n tempo van twee maal die van 'n binêre ASK (BASK) sein met dieselfde bandwydte lewer a.g.v. die twee bisse wat per simbool oorgedra word. Fig.2.5 is 'n voorbeeld van die konstellasiediagram van 'n 4ASK sein.

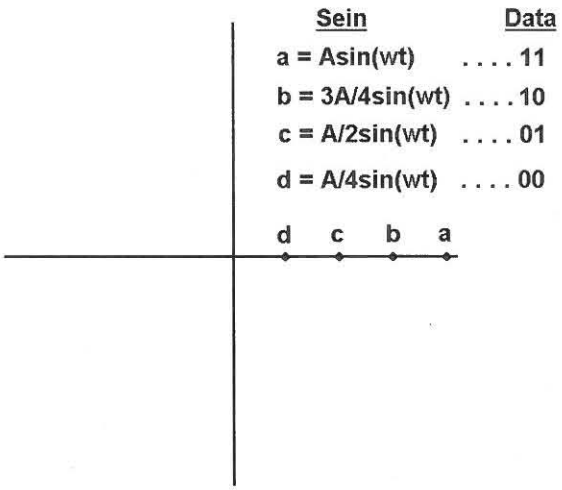


Fig. 2.5 : 4ASK konstellasie

Uit fig.2.5 word gesien dat die vier verskillende amplitudes eweredig gespasieer is tot by die maksimum amplitude van die draaggolf, punt a. Die spasiëring tussen die aangrensende amplitudes is nou nader aan mekaar as in die BASK geval en dit veroorsaak dat MASK 'n verhoging in fouttempo lewer a.g.v. swakker ruisimmuniteit. In fig.2.6 *[19, p. 355]* word getoon hoe die moontlikheid van simboolfoute t.o.v. die

seinruisverhouding toeneem vir verskillende waardes van M in 'n ASK stelsel. Dieselfde verskynsel kom ook voor in 'n PSK stelsel. In 'n FSK stelsel neem die moontlikheid van simboolfoute af met die verhoging van M, maar dit het 'n verhoging in bandwydte tot gevolg.

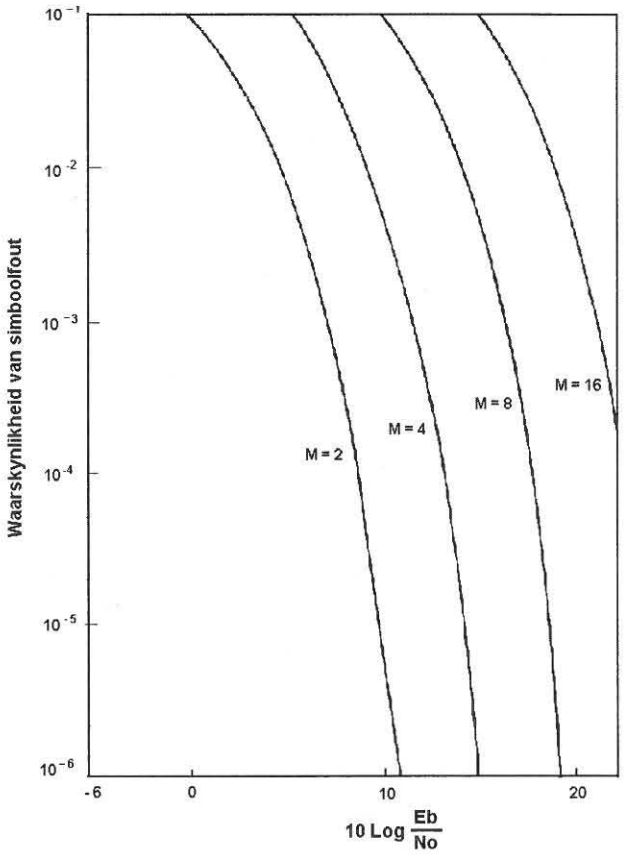


Fig. 2.6 : Waarskynlikheid van simboolfoute vir MASK

Aangesien PSK in die stelsel soos ontwikkel, gebruik is, word PSK vervolgens in meer besonderhede bespreek.

2.3 FASE-SKUIFSLEUTELING (PSK)

In die geval van analoog kommunikasie bestaan daar 'n groot ooreenkoms tussen frekwensiemodulasie en fasemodulasie. Dit is omdat die frekwensie van 'n golfvorm gedefinieer kan word as die tyd afgeleide van die oombliklike fase. In digitale kommunikasie is die verskil tussen frekwensiemodulasie en fasemodulasie meer beduidend omdat daar van diskrete golfvorms gebruik gemaak word.

Soos genoem, word PSK verkry deur die fase van 'n draaggolf te varieer t.o.v. die digitale inligting in die basisbandsein. In die binêre geval van PSK (BPSK) word daar tussen twee konstante fases geskakel terwyl die draagfrekwensie konstant bly.

Roden *[19, p. 401]* analiseer die BPSK modulasieproses as volg:

Die twee seine wat gebruik word om 'n binêre "0" en "1" te stuur kan as volg uitgedruk word:

$$s_0(t) = A\cos(2\pi f_c t + \theta_0) \qquad 0 < t \le T \qquad (2.2)$$

$$s_1(t) = A\cos(2\pi f_c t + \theta_1) \qquad 0 < t \le T \qquad (2.3)$$

waar θ_0 en θ_1 konstante fases afhangend van die versende bis, en T die omgekeerde van die bistempo is. 'n Ander manier om hierdie BPSK sein uit te druk is as volg:

$$s_i(t) = A\cos[2\pi f_c t + \beta d_i(t)] \qquad 0 < t \le T \qquad (2.4)$$

waar $d_i(t)$ 'n waarde van -1 of 1 aanneem afhangend of 'n 0 of 'n 1 gestuur word. β is die verandering in fase of modulasie-indeks. D.m.v. trigonometriese reëls kan (2.4) uitgebrei word na

$$s_i(t) = A\cos(2\pi f_c t)\cos[\beta d_i(t)] - A\sin(2\pi f_c t)\sin[\beta d_i(t)]. \qquad (2.5)$$

Deur van die gelyke en ongelyke eienskappe van cos en sin gebruik te maak kan hierdie vergelyking vereenvoudig word na

$$s_i(t) = A\cos(\beta)\cos(2\pi f_c t) - A d_i(t)\sin(\beta)\sin(2\pi f_c t). \qquad (2.6)$$

Die eerste term in vergelyking (2.6) stel die draaggolf voor en die drywing daarvan kan uitgedruk word as:

$$P_c = \frac{A^2\cos^2\beta}{2} \qquad (2.7)$$

Die tweede term stel die gemoduleerde inligting, of sybande voor en die drywing daarvan is

$$P_d = \frac{A^2 \sin^2 \beta}{2}. \qquad (2.8)$$

As hierdie twee terme gesommeer word, word die totale drywing van die sein verkry wat gelyk is aan $A^2/2$. Dit stem ooreen met die drywing van 'n gewone sinusgolf met amplitude A.

Deur die modulasie-indeks β te vervang met $\pi/2$ vereenvoudig vergelyking (2.6) na

$$s_i(t) = -Ad_i(t)\cos(2\pi f_c t) \qquad (2.9)$$

Hierdie vergelyking stel dus 'n sein met 'n onderdrukte draaggolf voor. As $d_i(t) = +1$ of -1 is :

$$\begin{aligned} s_0(t) &= +A\cos(2\pi f_c t) \\ s_1(t) &= -A\cos(2\pi f_c t) \\ s_1(t) &= -s_0(t) \end{aligned} \qquad (2.10)$$

en die twee seine is omgekeerd relatief tot mekaar.

In die voorbeeld van 'n BPSK sein (fig.2.4(a)) is $\beta = \pi/2$ en die 180° spasiëring tussen die twee seine kan in die konstellasiediagram, fig.2.4(b), gesien word.

2.3.1 Frekwensiespektrum van PSK sein

As die onderdrukte draaggolf tipe PSK sein oorweeg word, en daar word aanvaar dat 'n pseudo-ewekansige datastroom gestuur word, word die energiespektrum soos in fig.2.7 *[5, p. 136]* gelewer. Die PSK spektrum korrespondeer met die van 'n keer-nie-terug-na-zero (NRZ) sein behalwe dat dit in die frekwensie-as geskuif is. Die benodigde bandwydte vir 'n PSK sein is gelyk aan twee maal die hoogste basisbandfrekwensie *[5, p. 135]*.

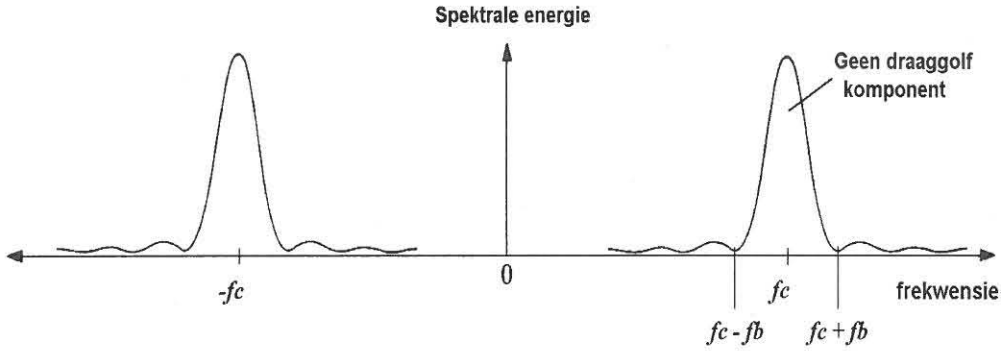


Fig. 2.7 : Energiespektrum van 'n PSK sein

2.3.2 Kwadratuur fase-skuifsleuteling (QPSK)

Wanneer twee binêre basisbandseine op twee ortogonale seine, soos $A\sin 2\pi ft$ en $A\cos 2\pi ft$, onderskeidelik fase gemoduleer, en saam oor dieselfde kanaal gestuur word, lewer die gesamentlike sein 'n konstellasie met vier fases wat eweredig op die omtrek van 'n sirkel gespasieer is (fig.2.8.).

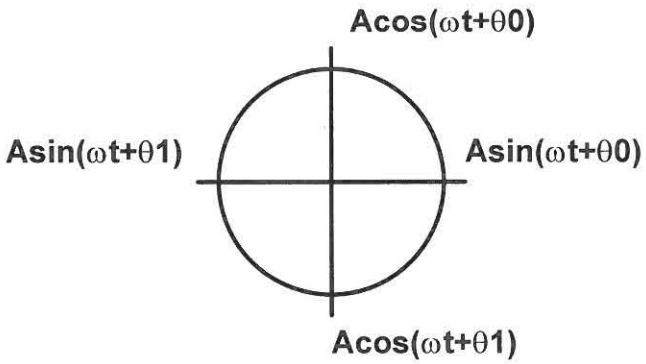


Fig. 2.8 : QPSK Konstellasiediagram

Hierdie kombinering van twee BPSK seine lewer 'n bistempo van twee maal die van 'n enkele BPSK sein met dieselfde bandwydte *[17, p. 255]*. Dit staan bekend as kwadratuur fase-skuifsleuteling (QPSK) en is dieselfde as viervlak PSK (4PSK). QPSK het 'n verhoging in fouttempo t.o.v. BPSK tot gevolg a.g.v. die sein se veelvlak eienskappe.

Bogenoemde QPSK sein is verkry deur van twee afsonderlike basisbandseine gebruik te maak. 'n Ander manier om QPSK te verkry is deur die binêre data in groepe van twee

bisse op te deel wat dan 'n viervlaksein lewer waar elke vlak gekoppel word aan een van die vier moontlike fases.

Om die moontlikheid van foute te minimaliseer word die fases verkieslik aan die hand van die Graykode aan die vlakke of binêre bisgroepe gekoppel, sodat indien 'n fout voorkom en 'n aangrensende fase ontvang word, dit 'n verandering van slegs 1 bis veroorsaak *[4, p. 65]*.

Tabel 2.1 is 'n voorbeeld van die kwantifisering van die binêre data en die koppeling daarvan aan sekere fases vir 'n 8PSK stelsel.

Tabel 2.1 : 8PSK simbool fases

Binêre groepe	Basis-8 vlak	Fase
000	0	0°
001	1	45°
011	2	90°
010	3	135°
110	4	180°
100	5	225°
101	6	270°
111	7	315°

2.3.3 Differensiële fase-skuifsleuteling (DPSK).

Gemoduleerde seine kan by 'n ontvanger op twee maniere gedemoduleer word nl: koherente en nie-koherente demodulasie.

Koherente demodulasie kan gedoen word wanneer daar presiese replikas van die moontlike ontvangde seine by die ontvanger is. Dus moet daar presiese kennis van die draaggolffase by die ontvanger wees - wat beteken dat die ontvanger presies in fase met die sender moet wees. Hierdie replikas van die moontlike ontvangde seine word dan met die ontvangde sein ge-kruiskorreleer en 'n besluit oor die mees waarskynlike ontvangde

simbool word geneem t.o.v. vooraf bepaalde korrelasievlakke. In die nie-koherente geval hoef die absolute fase van die draaggolf nie bekend te wees nie, wat 'n laer kompleksiteit demodulator tot gevolg het, met 'n effense verswakking in prestasievermoë.

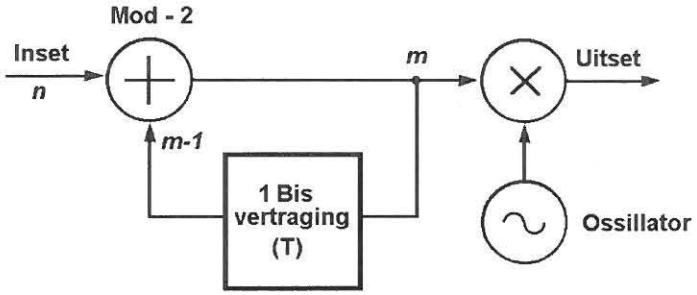
Die deteksie van PSK vertrou op die vermoë van die ontvanger om die draaggolf in die korrekte fase uit die ontvangde sein te verkry (d.w.s. koherente demodulasie). Alhoewel 'n draaggolf bepaal kan word vir PSK d.m.v. nie-liniêre prosessering, bestaan daar nog steeds onsekerheid of dit in fase is, of in fase plus 'n veelvoud van $2\pi/M$ waar M die hoeveelheid vlakke in die modulasieproses voorstel *[23, p. 508]*.

Die probleem kan opgelos word as die binêre data differensieël ge-enkodeer en dan fase gemoduleer word. Die proses staan bekend as differensieël ge-enkodeerde PSK, of as differensiële fase-skuifsleuteling (DPSK) indien die modulasie en kodering gesamentlik as een vorm van sleuteling beskou word. Soos die naam aandui lewer DPSK 'n sein waar die informasie gedra word in die verskil in fase tussen twee opeenvolgende simbole en nie in die absolute fase soos in gewone PSK nie.

Omdat dit met DPSK deteksie nie nodig is om die absolute polariteit van die simboolfase te bepaal nie, is draaggolfherwinning nie nodig nie en nie-koherente demodulasie is moontlik. Die moontlikheid van foute met DPSK is dubbeld die van PSK a.g.v. die aard van differensiële kodering, maar as hierdie verdubbeling in terme van seinruisverhouding gesien word, is die verswakking relatief klein (3dB vir $M > 4$) *[18, p. 266]*.

2.3.3.1 Modulasie van DPSK

Fig.2.9 is 'n voorbeeld van 'n binêre DPSK modulator waar die binêre data deur 'n eenvoudige logikakring differensieël ge-enkodeer word. Die enkodeerder se uitset word vlak aangepas (0 word 1 en 1 word -1) om die draaggolf mee te moduleer. Die uitset van die modulator is 'n sinusgolf wat van polariteit verander ooreenkomstig met die ge-enkodeerde binêre data.



n	m-1	m	θ
0	0	0	π
1	0	1	0°
1	1	0	π
0	0	0	π

Fig. 2.9 : DPSK Modulator

Dieselfde metode kan vir veelvlak DPSK gebruik word deur die binêre data op te breek in groepe van $(\log_2 M)$ (bv. 3-bis groepe as M = 8). Hierdie waarde word differensieël geenkodeer en vlak aangepas t.o.v. 'n vooraf bepaalde orde waarmee die draaggolf daarna gemoduleer word (sien tabel 2.1).

2.3.3.2 Demodulasie van DPSK

By die ontvanger van bogenoemde binêre DPSK sein kan die demodulasie gedoen word deur twee opeenvolgende pulse of simbole met mekaar te vergelyk om die faseverandering wat plaasgevind het te bepaal.

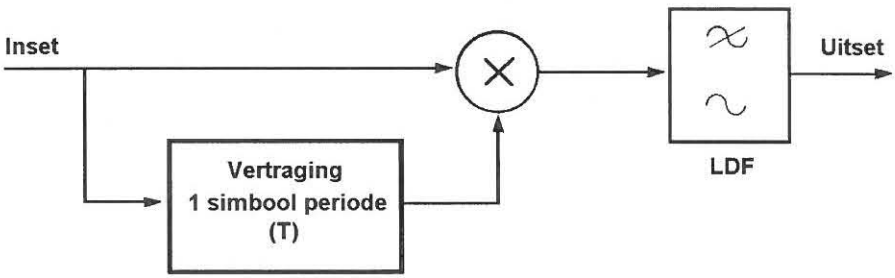


Fig. 2.10 : DPSK demodulator

Die gebruik van 'n relatief eenvoudige demodulator (sien fig.2.10) is moontlik waar twee opeenvolgende simbole met mekaar vermenigvuldig word. Wanneer daar 'n faseverandering van 180° tussen twee opeenvolgende simbole plaasgevind het lewer die vermenigvuldiging 'n golfvorm met 'n negatiewe gemiddelde waarde (fig.2.11 - simbool 1 & 2). 'n Golfvorm met 'n positiewe gemiddelde waarde word verkry as daar geen

verandering tussen twee opeenvolgende simbole plaasgevind het nie (fig.2.11 - simbool 3) *[5, p. 197]*.

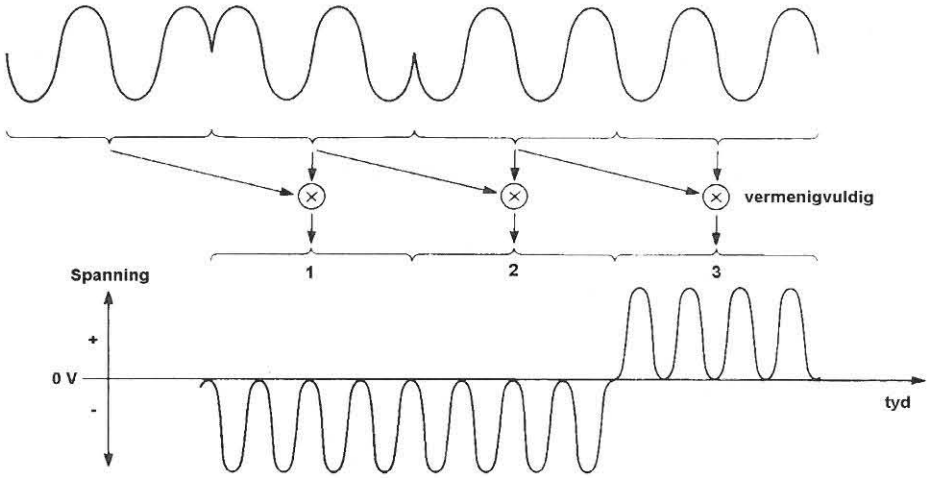


Fig. 2.11 : Differensiële deteksie van DPSK

Alhoewel dit maklik is om hierdie tipe deteksie te implementeer, is 'n nadeel daarvan dat wanneer die simbole met mekaar vermenigvuldig word, albei met ruis besoedel is wat die prestasievermoë van die tegniek negatief beïnvloed.

Meerfasige of veelvlak demodulasie kan bewerkstellig word deur van dieselfde beginsel as hierbo gebruik te maak nl. korrelasie of filter deteksie. Die ontvangde sein word nou net gekorreleer met die M moontlike seine waarna 'n besluit geneem word t.o.v. die grootste ooreenstemming of korrelasie.

Fig.2.12 is 'n 4PSK korrelasiedetektor wat in hierdie projek in 'n gewysigde vorm gebruik is vir die deteksie van 8DPSK. (Die wysiging word in paragraaf 4.3.1.4 bespreek.)

In die 4PSK korrelasiedetektor word die ontvangde puls S_i vermenigvuldig met die 4 moontlike variasies van die 4PSK sein, nl.

$$\sin(\omega t + 0^\circ),\quad \sin(\omega t + 90^\circ),\quad \sin(\omega t + 180^\circ)\quad \text{en}\quad \sin(\omega t + 270^\circ)$$

Die onderskeie produkte word dan geïntegreer oor 'n tydsinterval van een simbool (T). Daarna word die vier korreleerders se uitsette (y1, y2, y3 en y4) vergelyk en daar word op

die fase variasie met die grootste korrelasie besluit as die waarskynlikste ontvangde simbool.

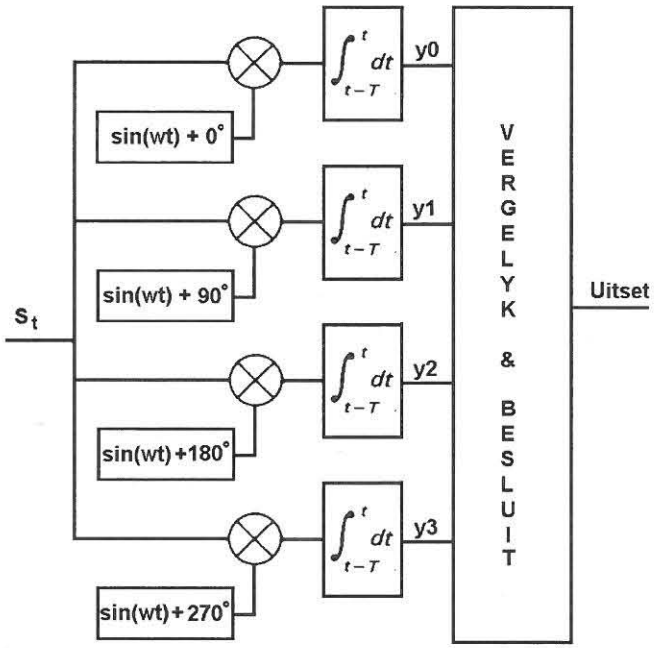


Fig. 2.12 : 4PSK Korrelasiedetektor

Die 4PSK korrelasiedetektor in fig.2.12 kan vereenvoudig word deur slegs van die twee korreleerders,

$$\sin(\omega t + 0^\circ) \quad \text{en} \quad \sin(\omega t + 90^\circ)$$

gebruik te maak en 'n addisionele besluit te neem. Die twee korreleerders se uitsette word weereens met mekaar vergelyk, en die grootste absolute waarde van korrelasie word bepaal. Deur dan na die polariteit van hierdie waarde te kyk kan 'n besluit geneem word t.o.v. die korrekte fase aangesien,

$$\sin(\omega t + 0^\circ) = -\sin(\omega t + 180^\circ) \quad \text{en} \quad \sin(\omega t + 90^\circ) = -\sin(\omega t + 270^\circ).$$

2.3.3.3 *Prestasievermoë van DPSK*

As daar eerstens na die prestasievermoë van PSK t.o.v. ASK en FSK gekyk word kan die volgende gevolgtrekkings gemaak word:

- Die benodigde bandwydte vir FSK is gewoonlik baie groter as vir PSK en ASK. PSK en ASK se bandwydte is twee maal die basisbandfrekwensie *[20, p. 211]*.
- PSK en FSK besit 'n konstante omhulling wat bestand is teen amplitude nie-liniariteit en is dus meer ruisbestand as ASK *[10, p. 273]*.
- PSK benodig 'n S/N verhouding van 3dB laer as ASK en FSK ten einde dieselfde fouttempo te lewer *[19, p. 440]*.
- MFSK presteer beter as MPSK by hoë seinruisverhoudings, maar daar is 'n prys wat betaal moet word a.g.v. MFSK se groot benodigde bandwydte *[19, p. 440]*.

DPSK besit dieselfde eienskappe as PSK t.o.v. bandwydte en omhulling. Daar is wel 'n verskil in die moontlikheid van foute a.g.v. die differensiële aard van DPSK. Fig.2.13 *[10, p. 312]* is 'n aanduiding van hoe DPSK presteer t.o.v. PSK en ander stelsels by verskillende S/N verhoudings.

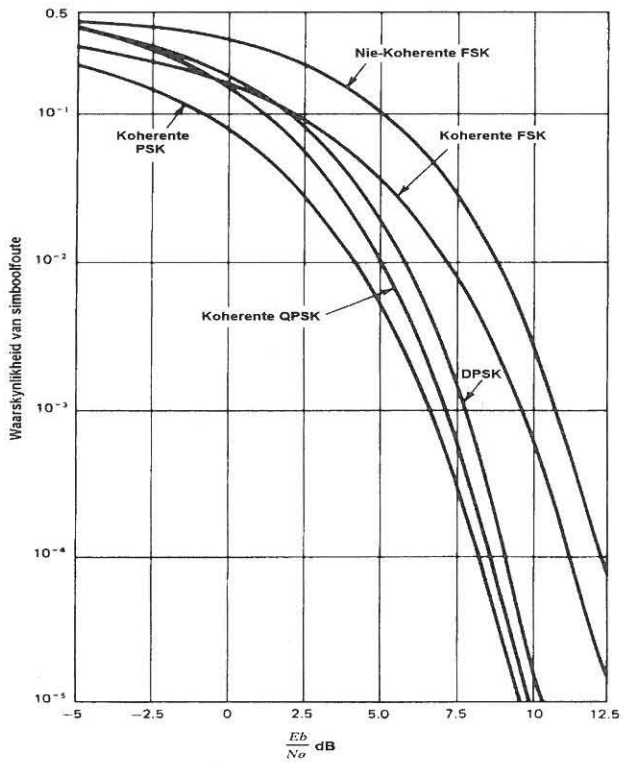


Fig. 2.13 : Waarskynlikheid van simboolfoute vir BPSK, QPSK, BDPSK en BFSK

Uit die grafiek blyk dat DPSK redelik goed presteer as in ag geneem word dat dit 'n nie-koherente stelsel is. Nog 'n voordeel van DPSK is die eenvoudige implementering daarvan soos hierbo uiteengesit.

2.4 KANAAL KODERING

Data kodering word hoofsaaklik om drie redes gedoen:

1. Dit verseker dat die data gelykstroom gebalanseerd is wat beteken dat daar ewe veel nulle en ene oor 'n verlengde tydperk voorkom. Hierdie balansering is van belang vir transformator gekoppelde stelsels wat ontwerp is om grondlusse te voorkom.
2. Kodering help met kloksinchronisasie deur lang stringe opeenvolgende nulle of ene te voorkom.
3. Kodering maak dit moontlik om foute te beheer en te korrigeer sodat 'n betroubare reproduksie van die gestuurde data by die ontvanger herwin kan word.

[21, p. 27]

Daar is twee tipes kodes in algemene gebruik, nl. Blokkodes en Konvolusiekodes. 'n Blokkode deel die datastring op in datawoorde van k bisse elk wat dan 'n kode lewer met 2^k verskillende datawoorde. Hierdie datawoorde word deur die koderingsproses omgeskakel in kodewoorde wat n aantal bisse lank is waar $k < n$. Die kode staan bekend as 'n (n,k) blokkode en die tempo word deur $R = k/n$ voorgestel. Aangesien elke n-bis kodewoord 'n k-bis datawoord verteenwoordig is die enkodeerder geheueloos en kan dit deur kombinasielogika geïmplementeer word. Vir 'n effektiewe binêre blokkode moet $R < 1$. Meer oortollige bisse kan tot die kodewoord toegevoeg word om beter foutregstellingsvermoë te verkry *[19, p. 183]*. Praktiese waardes vir k strek vanaf 3 tot 'n paar honderd en vir R vanaf 1/4 tot 7/8. Waardes buite hierdie limiete is moontlik, maar lewer gewoonlik praktiese probleme *[6, p. 9]*. Die bekende (7,4) Hammingkode is 'n voorbeeld van 'n blokkode.

'n Konvolusiekode neem ook k-bis datawoorde van die datastring en kodeer dit om n-bis kodewoorde te lewer. Elke kodewoord word nie alleen deur die k-bis datawoord in dieselfde tydsinterval bepaal nie, maar ook deur die vorige m datawoorde. Die kode besit

dus 'n geheue of konstante lengte van m en staan bekend as 'n (n,k,m) konvolusiekode met die tempo wat deur $R = k/n$ voorgestel word. Tipies is k en n klein heelgetalle (1 tot 8) sodat $k < n$ en R besit waardes van 1/4 tot 7/8. Soos by blokkodes kan die foutregstellingsvermoë verbeter word deur meer oortollige bisse tot die kodewoord te voeg. Hierdie vermeerdering van oortollige bisse veroorsaak 'n verlaging in bistempo. Die foutregstellingsvermoë by konvolusiekodes word verbeter deur die geheue waarde, m, van die kode te vergroot *[13, p. 287]*. Tipiese waardes vir m strek van 2 tot 60 *[6, p. 9]*.

Hierdie projek handel primêr oor konvolusiekodering en daarom sal slegs hierdie tipe kodering meer spesifiek bespreek word.

2.4.1 Konvolusiekodes

Konvolusiekodes kan onderverdeel word in sistematiese en nie-sistematiese kodes. 'n Sistematiese kode is 'n kode waarvan die eerste bisse in die n-bis kodewoord presiese weergawes van die k-bis datawoord is.

Fig.2.14 is 'n voorbeeld van 'n (2,1,3) nie-sistematiese konvolusie enkodeerder. Die enkodeerder bestaan uit 'n 3-bis skuifregister (m = 3) met twee modulus-2 sommeerders (n = 2) en 'n multiplekseerder om die enkodeerder uitsette seriaal te maak.

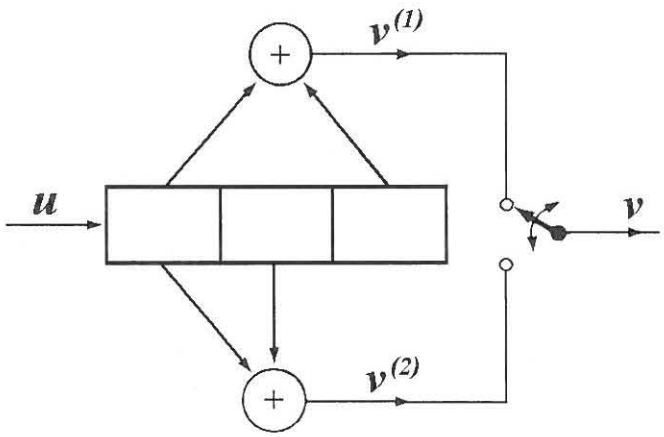


Fig. 2.14 : (2,1,3) Binêre konvolusiekode enkodeerder

Vanaf *[13, p. 288]* is die volgende afleiding vir die konvolusiekode enkodeerder in fig.2.14 ontwikkel.

Die binêre informasie sein

$$u = (u_0 + u_1 + u_2 + \ldots) \tag{2.11}$$

word een bis op 'n slag (k = 1) in die enkodeerder ingeskuif. Die twee enkodeerder uitsette

$$v^{(1)} = (v_0^{(1)}, v_1^{(1)}, v_2^{(1)}, \ldots) \text{ en } v^{(2)} = (v_0^{(2)}, v_1^{(2)}, v_2^{(2)}, \ldots)$$

kan verkry word deur die konvolusie van die inset u met die enkodeerder se twee "impuls weergawes". Twee tipiese impuls weergawes kan verkry word deur $u = (100\ldots)$ te maak en dan die twee ooreenkomstige uitset stringe te bepaal. Die impuls weergawes kan geskryf word as,

$$g^{(1)} = (g_0^{(1)}, g_1^{(1)}, \ldots, g_m^{(1)})$$
$$g^{(2)} = (g_0^{(2)}, g_1^{(2)}, \ldots, g_m^{(2)}).$$

en dus vir die enkodeerder in fig.2.14 as

$$g^{(1)} = (101) \text{ en } g^{(2)} = (110).$$

Die impuls weergawes $g^{(1)}$ en $g^{(2)}$ staan bekend as die volgorde generators van die kode. Die enkoderingsvergelykings kan nou as volg geskryf word:

$$v^1 = u * g^{(1)} \text{ en } v^2 = u * g^{(2)},$$

waar $*$ dui op die konvolusieproses met al die bewerkinge as modulus-2 optelling. Met $l \geq 0$ kan die uitsette dan deur die volgende voorgestel word:

$$v_l^{(j)} = \sum_{i=0}^{m} u_{l-i} g_i^{(j)} = u_l g_0^{(j)} + u_{l-1} g_1^{(j)} + \ldots + u_{l-m} g_m^{(j)} \qquad j = 1,2 \ldots n \qquad (2.12)$$

met $u_{l-i} = 0$ waar $l < i$. Dus vir die enkodeerder in fig.2.14 is,

$$v_l^{(1)} = u_l + 0 + u_{l-2} \qquad (2.13)$$

$$v_l^{(2)} = u_l + u_{l-1} + 0 \qquad (2.14)$$

Na die enkodering word die twee uitsetstringe gemultiplekseer na 'n enkele datastroom sodat:

$$v = (v_0^{(1)} v_0^{(2)}, v_1^{(1)} v_1^{(2)}, v_2^{(1)} v_2^{(2)}, \ldots) \qquad (2.15)$$

wat bekend staan as die kodewoord, wat uiteindelik oor die kanaal gestuur gaan word. Hieruit is dit dus duidelik dat 'n konvolusiekode uit 'n aaneenlopende reeks binêre getalle bestaan.

Indien 'n datastring van 1101001 (met die mees onlangse bis regs) in die enkodeerder in fig.2.14 ingeskuif word sal die uitset 111011010110110 wees. Om hierdie uitsette te verkry is daar aanvaar dat die datastroom deur 'n aantal zero's voorafgegaan is.

2.4.1.1 Kodeboom

'n Meer grafiese manier vir die voorstel van die in- en uitsette van 'n konvolusiekode enkodeerder kan verkry word aan die hand van 'n kodeboom wat die uitset aandui vir elke moontlike insetkombinasie. Fig.2.15 toon die kodeboom vir die enkodeerder in fig.2.14. As by punt A begin word, bepaal die eerste insetbis of die vertakking na punt B of C plaasvind. Indien dit 'n 1 is beweeg ons afwaarts tot by punt C waar die dienooreenkomstige 11 uitset op die tak aangedui is. As die eerste insetbis 0 was sou ons opwaarts beweeg het tot by punt B wat 'n 00 uitset sou lewer. So vertak die boom verder in elke tak na 'n nuwe vlak vir elke nuwe insetbis en kan die boom tot in oneindigheid groei om die gekodeerde uitset vir 'n sekere inset datastroom te verteenwoordig.

As die boom se takke oorweeg word, word gesien dat die takke geometries na die volgende vlak uitsprei met elke databis wat in die enkodeerder ingeskuif word, in so 'n mate dat die moontlike insetkombinasies verdubbel met elke stap. Die feit dat die uiset data bepaal word deur 'n beperkte aantal insetbisse, wat gelyk is aan die getal posisies in die skuifregister, veroorsaak dat die boom herhalend begin raak na 'n aantal vlakke van vertakking. Daar kan dus in fig.2.15 gesien word dat die boonste helfte van die boom 'n identiese weergawe van die onderste helfte word na 3 vlakke of stappe (sien punte H tot K in die boonste en onderste helfte van die boom). As hierdie stadium in die boom bereik word is daar slegs vier uitsetkombinasies, wat herhaal in elke vlak, onafhanklik van die insetdata. Die stippellyn wat deur die boom getrek is, is 'n aanduiding van die vertakking deur die boom, en dus die kodewoorde, vir tipiese insetdata. Koppeling van 1,1,0,1 op die inset lewer dus 11, 10, 11, 01 op die enkodeerder uitset.

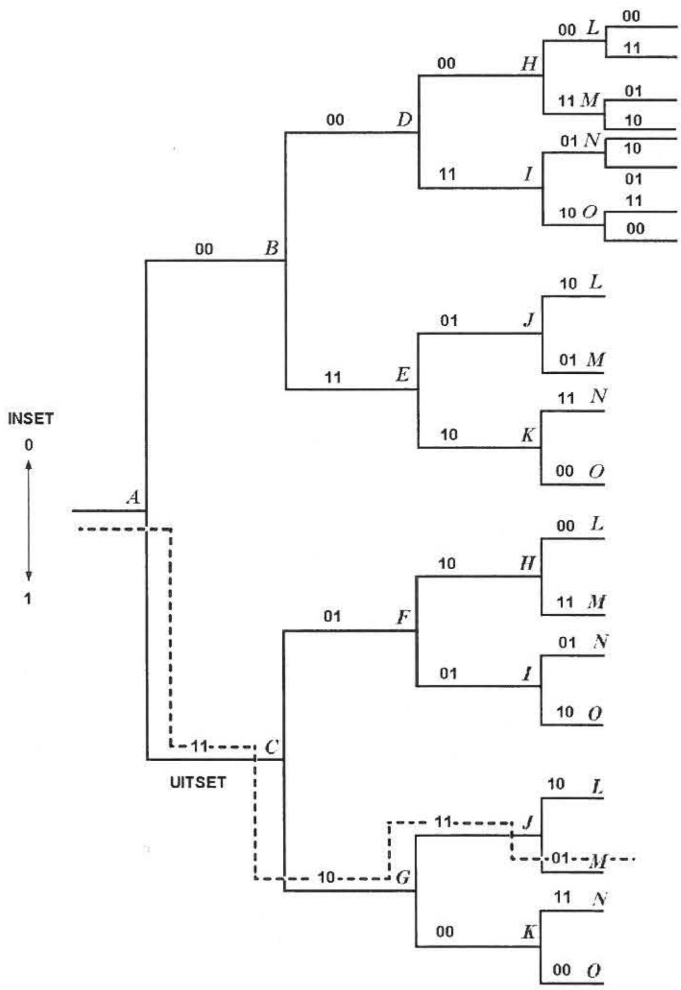


Fig. 2.15 : Kodeboom vir konvolusiekode enkodeerder.

2.4.1.2 Trellisdiagram

'n Vereenvoudigde grafiese weergawe van die boom staan bekend as 'n trellis. Die ooreenstemmende trellis vir fig.2.15 word in fig.2.16 getoon met die punte dienooreenkomstig gemerk en die uitsette op die takke aangedui.

Die trellis verdeel by elke punt in twee paaie waar die boonste pad 'n 0 inset verteenwoordig en die onderste 'n 1 inset. As daar na die punte in die horisontale groepe (ABDHL, CEIM, FJN en GKO) gekyk word, word opgemerk dat die individuele punte binne 'n groep dieselfde eienskappe besit. D.w.s. indien 'n datastring by enige punt binne 'n horisontale groep begin, word dieselfde uitset verkry as wat die geval sou wees as die datastring by enige ander punt binne daardie groep sou begin het. Hierdie trellis of ekwivalente diagram van die boom wys duidelik daarop dat die konvolusiekode vier definitiewe state besit wat oor en oor sal herhaal tot in oneindigheid. Soos in die boom in fig.2.15 dui die stippellyn die pad wat gevolg sal word deur die trellis en die kodewoord uitset vir 'n datainset van 1101.

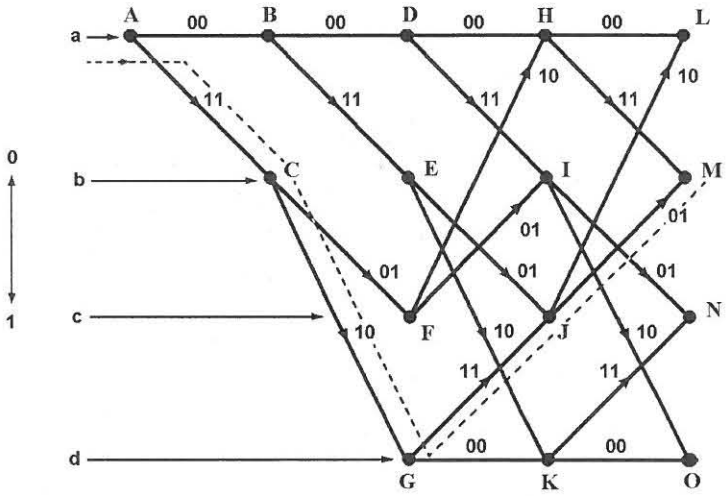


Fig. 2.16 : Trellisdiagram van konvolusie enkodeerder

2.4.1.3 Standdiagram

Dieselfde inligting verkry uit die trellis kan uit 'n standdiagram verkry word, waarvan fig.2.17 'n voorbeeld is. Die vier state a,b,c,d, in fig.2.17 verteenwoordig die horisontale groepe in die trellis soos in fig.2.16 aangedui. Daar is twee moontlike insetbisse na elke

staat toe, met twee ooreenkomstige uisetbisse in hakies aangedui. Pyltjies toon die moontlike standveranderings aan. Die stippellyn dui die volgorde van standverandering aan, en die ooreenkomstige uitsetdata, vir dieselfde insetdata soos bespreek t.o.v. die boom en trellis.

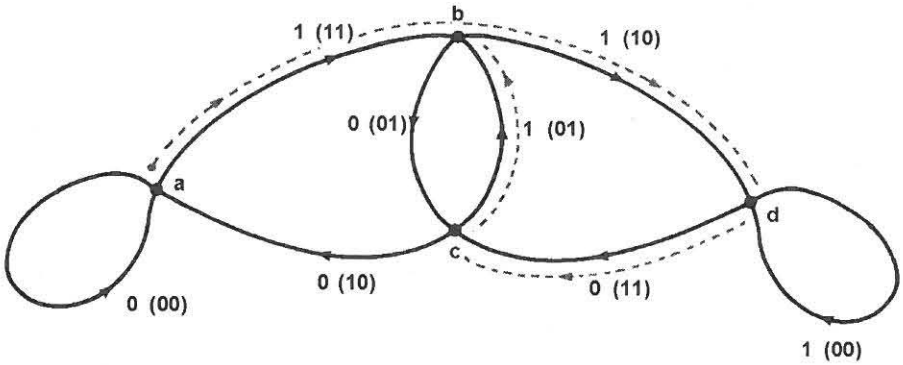


Fig. 2.17 : Standdiagram van konvolusie enkodeerder

2.4.1.4 Katastrofiese kodes

'n Probleem wat opduik met die ontwerp van 'n konvolusiekode is die moontlikheid dat die kode katastrofies van aard mag wees. Dit beteken dat indien daar 'n eindige aantal foute in die datastroom oor die kanaal insluip, dit 'n oneindige aantal foute in die dekodering van die data tot gevolg het *[27, p. 250]*. 'n Katastrofiese kode kan uit die kode se standdiagram geïdentifiseer word deur op te let vir lusse in die diagram wat 'n 00 uitset lewer, behalwe die lus rondom die zero staat - a-staat in fig.2.17 (0 inset lewer 00 uitset) *[13, p. 308]*. Volgens fig.2.17 lewer bostaande enkodeerder dus 'n katastrofiese kode a.g.v. die lus rondom die d-staat in die standdiagram waar 'n 1 inset 'n 00 uitset lewer.

2.4.2 Dekodering van konvolusiekodes.

'n Belangrike maatstaf van 'n konvolusiekode is die minimum vry afstand, wat 'n aanduiding is van die minimum Hammingafstand tussen enige twee moontlike opeenvolgende kodewoorde, soos tussen stappe AC(11) en CF(01), of tussen AC(11) en CG(10) op die trellis (fig.2.16). Die afstand in albei gevalle is 1. Vir die beste prestasie van 'n kode moet die minimum vry afstand so groot as moontlik wees. Die belangrikheid hiervan sal later na vore kom.

Daar bestaan verskeie metodes vir die dekodering van konvolusiekodes nl: Volgorde dekodering, Rant dekodering en Viterbi dekodering.

Viterbi dekodering is die optimum en die algemeenste tegniek vir die dekodering van konvolusiekodes in moderne stelsels. Dit is vernoem na A.J. Viterbi wat in 1967 vir die eerste keer van hierdie dekoderingstegniek gebruik gemaak het *[5, p. 218]*. Die Viterbi dekoderingsproses besit 'n vaste lengte wat eksponensieel groei soos die konstante lengte van 'n konvolusiekode verander. Daarom is dit slegs prakties in kodes met 'n kort konstante lengte.

Die Volgorde dekoderingstegniek was die eerste algoritme wat ontwikkel is om konvolusiekodes mee te dekodeer. Dit is oorspronklik deur Wozencraft in 1957 en 1961 voorgestel en is later in 1963 deur Fano aangepas *[18, p. 475]*. Volgorde dekodering is nie ideaal in die dekodering van konvolusiekodes nie, maar die dekoderingsproses is onafhanklik van die konstante lengte van 'n kode. Die Fano volgorde dekoderingsalgoritme soek die kortste pad deur die trellis deur een pad op 'n slag te toets. Die aantal berekeninge benodig om 'n datastring te dekodeer is redelik ewekansig. Alhoewel die meeste stringe baie vinnig gedekodeer word is daar somtyds lang soektogte wat veroorsaak dat sommige datastringe onvoltooid of glad nie gedekodeer word nie.

Meerderheidslogika of Rant dekodering is oorspronklik vir die dekodering van blokkodes ontwikkel, maar in 1963 het Massey die toepaslikheid daarvan vir konvolusiekodes aangedui. Rant dekodering verskil van Viterbi dekodering en Volgorde dekodering aangesien die finale besluit t.o.v. 'n gegewe datastring op slegs een konstante lengte van die ontvangde blokke i.p.v. die hele ontvangde datastring gebaseer is. Dit lei tot swakker prestasievermoë as Viterbi of Volgorde dekodering, maar die implementering daarvan is baie eenvoudiger *[13, p. 388]*.

Hierdie projek maak gebruik van die Viterbi proses en daarom sal slegs hierdie tipe dekodering verduidelik word.

2.4.2.1 *Die Viterbi algoritme*

Die Viterbi algoritme gebruik die trellisstruktuur en die ontvangde data om die mees waarskynlike pad deur die trellis te bepaal. In kort vergelyk die algoritme die ontvangde datastring met al die moontlike gestuurde kombinasies en kies dan die datastring of kombinasie met die minimum-afstand *[19, p. 191]*. Die minimum-afstand kan die Hammingafstand tussen opeenvolgende kodewoorde of die Euklidiese afstand tussen opeenvolgende kodevektore (kodevektore word later saam met trelliskodemodulasie bespreek) wees. As die Hammingafstand gebruik word, word daar van hardebesluitneming Viterbi dekodering gepraat, en indien die Euklidiese afstand gebruik word van sagtebesluitneming Viterbi dekodering. Ter verduideliking van die Viterbi proses word van die enkodeerder in fig.2.14 gebruik gemaak (wat die Hammingafstand as die minimum-afstand gebruik).

Veronderstel die datastring (11 10 11 01), soos deur die enkodeerder in fig.2.14 gelewer, word ontvang. Aangesien al die state in die horisontale groepe in die trellis identiese eienskappe besit kan ons op enige plek in die trellis begin. Die eerste dekoderingstap sal wees om die Hammingafstande van die eerste ontvangde datapaar "11" t.o.v. die ooreenstemmende bisse in die trellis te bepaal. Hierdie oombliklike Hammingafstande is in hakies op die trellis in fig.2.18(a) aangedui.

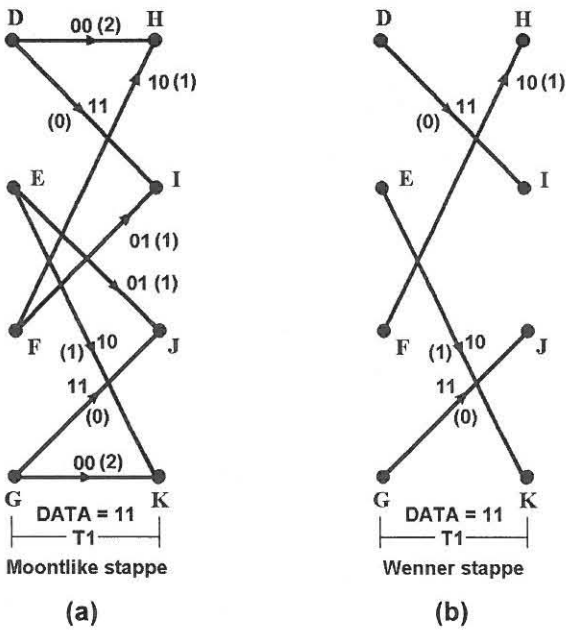


Fig. 2.18 : (a) Moontlike - en (b) wennerstappe op trellis

Uit die diagram kan gesien word dat daar agt moontlike stappe met verskillende afstande na die vier individuele nodusse of state is vir tydinterval T1. Die twee stappe na elke nodus se afstande word nou met mekaar vergelyk en die stap met die kleinste afstand word behou en die groter afstand stap word geëlimineer. Hierdie stap met die kleinste afstand staan bekend as die wennerstap of pad. Daar bly nou slegs vier wenners oor, een na elke nodus, soos in fig.2.18(b). Teen tydinterval T2 in fig.2.19(a) word die Hammingafstande vir die trellis bispare t.o.v. die nuwe ontvangde datapaar "10" bepaal om die oombliklike Hammingafstand te verkry (aangedui as die linker waarde in die hakies op die trellis).

Maar die agt moontlike stappe tydens T2 is afkomstig vanaf nodusse wat onderskeidelik voorafgegaan is deur vier wennerstappe tydens T1, wat elk klaar 'n sekere afstandswaarde besit. Om die vier wenners vir T2 te verkry moet die agt stappe se afstande onderskeidelik met die voorafgaande afstand tot by die betrokke nodus, waarvandaan elk van die agt stappe hul oorsprong het, gesommeer word om 'n totale afstand te verkry (aangedui as die tweede waarde in die hakies).

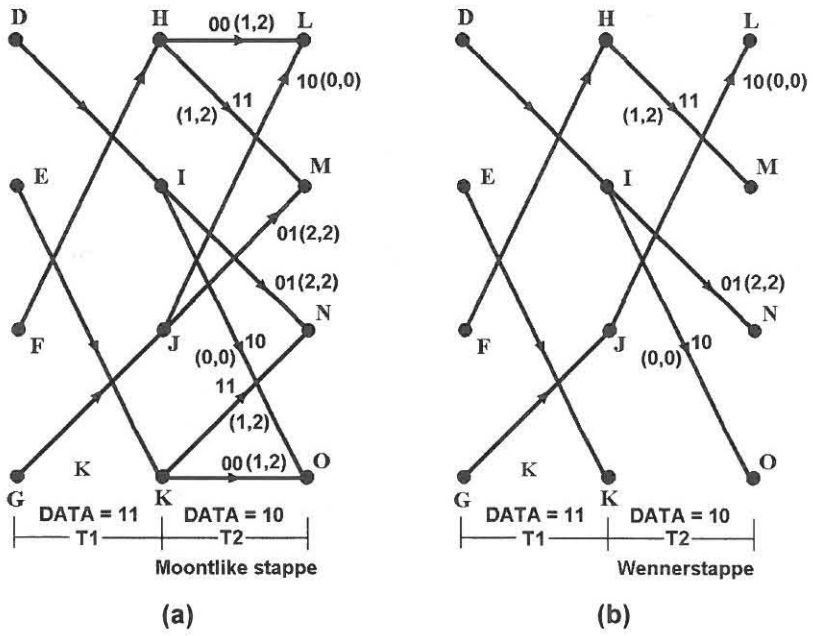


Fig. 2.19 : (a) Kumulatiewe afstande en (b) wennerstappe

Hierdie afstande moet dan met mekaar vergelyk word om 'n wenner vir elke nodus (L tot O) te bepaal. Net soos tydens T1 word die vier stappe met die kortste totale afstande behou en die ander geëlimineer. Hierdie proses word herhaal vir elke ontvangde data

bispaar of simbool sodat daar uiteindelik net die vier paaie of wenners deur die trellis loop. Indien paaie tot by dieselfde nodus 'n gelyke afstand het, kan enige van die twee paaie gekies word soos wat by nodus N en nodus M plaasgevind het tydens T2.

Na 'n aantal stappe deur die trellis, tipies 4 of 5 keer die konstante lengte *[15, p. 3-4]*, word die vier paaie met mekaar vergelyk en die kortste een of wennerpad word gekies. Daar word dan deur die pad teruggestap en die ooreenkomstige bis "1" of "0", afhangend daarvan of die trellis op of af beweeg het, word uitgelees vir elke stap wat dan die gedekodeerde data is.

Fig.2.20 toon die vier wennerpaaie na vier tydsintervalle met die totale afstande tot by elke nodus in hakies aangedui. Daar kan gesien word dat aan die einde van T4 daar twee wennerpaaie is, albei met 'n afstand van nul. As daar nou deur die twee paaie afsonderlik terug beweeg word sal 'n gedekodeerde waarde van "1101" en "0010" onderskeidelik verkry word. Die gedekodeerde "1101" stem ooreen met die oorspronklike onge-enkodeerde data, maar volgens die trellis is die gedekodeerde "0010" net so 'n geldige antwoord, alhoewel dit glad nie ooreenstem met die "1101" insetdata nie. Hierdie fout ontstaan a.g.v. die addisionele lus rondom die d staat in die standdiagram (sien fig.2.17). Uit bostaande voorbeeld kan nou gesien word hoe 'n swak ontwerpte konvolusiekode geaffekteer word deur hierdie ekstra zero lus wat katastrofiese foute tot gevolg kan hê.

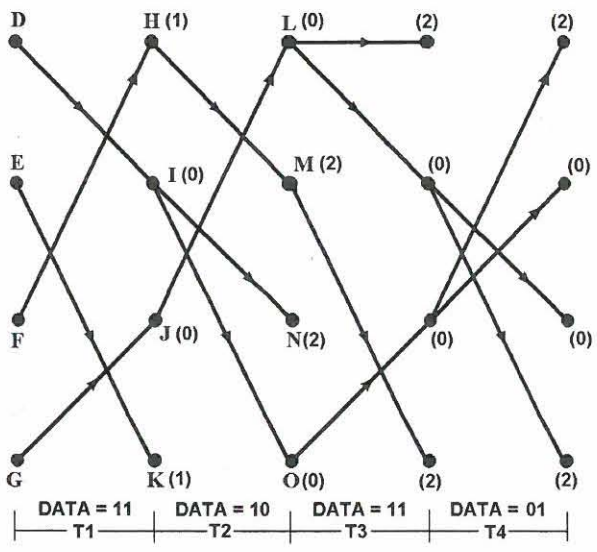


Fig. 2.20 : Wennerpaaie vir ontvangde 11101101

Alhoewel bogenoemde kode katastrofies van aard is kan die foutregstellende eienskappe van die Viterbi dekodering nog steeds aan die hand daarvan verduidelik word. Veronderstel dieselfde datastring soos hierbo gebruik (11101101) word gestuur maar (00101101) word ontvang, dus met 'n fout in die eerste kodewoord. Hierdie ontvangde kodewoorde sal die wennerpaaie deur die trellis soos in fig.2.21 lewer, met die pad afstande tot by elke nodus in hakies aangedui.

Die ontvangde string kodewoorde (00101101) kan nie 'n pad deur die trellis met 'n zero afstand realiseer nie a.g.v. die foutiewe kodewoord, wat 'n onmoontlike verandering in die ontvangde kodewoorde (vanaf die "00" kodewoord na die "10" kodewoord) suggereer. As gevolg hiervan sal die waarskynlikste pad, d.w.s. die pad met die kortste afstand deur die trellis, gevolg word. Uit fig.2.21 word gesien dat daar twee paaie met 'n kortste totale afstand van "1" is na vier tydsintervalle.

As hierdie twee paaie terug gevolg word sal 'n gedekodeerde uitset van "1101" en "0010" verkry word. Die "1101" uitset stem ooreen met die oorspronklike onge-enkodeerde data en die "0010" uitset nie. Dit wil dus voorkom of die Viterbi dekodering die fout reggestel het indien die 1101 uitset aanvaar word. 'n Probleem ontstaan egter omdat albei uitsette 'n afstand van 1 het, wat dit onmoontlik vir die Viterbi proses maak om te onderskei tussen die korrekte en foutiewe uitset.

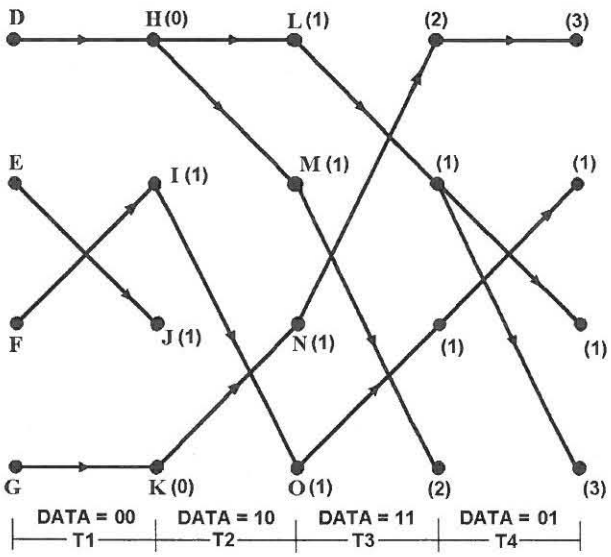


Fig. 2.21 : Wennerpaaie vir ontvangde 00101101

Alhoewel bogenoemde kode katastrofies van aard is kan die foutregstellende eienskappe van die Viterbi dekodering nog steeds aan die hand daarvan verduidelik word. Veronderstel dieselfde datastring soos hierbo gebruik (11101101) word gestuur maar (00101101) word ontvang, dus met 'n fout in die eerste kodewoord. Hierdie ontvangde kodewoorde sal die wennerpaaie deur die trellis soos in fig.2.21 lewer, met die pad afstande tot by elke nodus in hakies aangedui.

Die ontvangde string kodewoorde (00101101) kan nie 'n pad deur die trellis met 'n zero afstand realiseer nie a.g.v. die foutiewe kodewoord, wat 'n onmoontlike verandering in die ontvangde kodewoorde (vanaf die "00" kodewoord na die "10" kodewoord) suggereer. As gevolg hiervan sal die waarskynlikste pad, d.w.s. die pad met die kortste afstand deur die trellis, gevolg word. Uit fig.2.21 word gesien dat daar twee paaie met 'n kortste totale afstand van "1" is na vier tydsintervalle.

As hierdie twee paaie terug gevolg word sal 'n gedekodeerde uitset van "1101" en "0010" verkry word. Die "1101" uitset stem ooreen met die oorspronklike onge-enkodeerde data en die "0010" uitset nie. Dit wil dus voorkom of die Viterbi dekodering die fout reggestel het indien die 1101 uitset aanvaar word. 'n Probleem ontstaan egter omdat albei uitsette 'n afstand van 1 het, wat dit onmoontlik vir die Viterbi proses maak om te onderskei tussen die korrekte en foutiewe uitset.

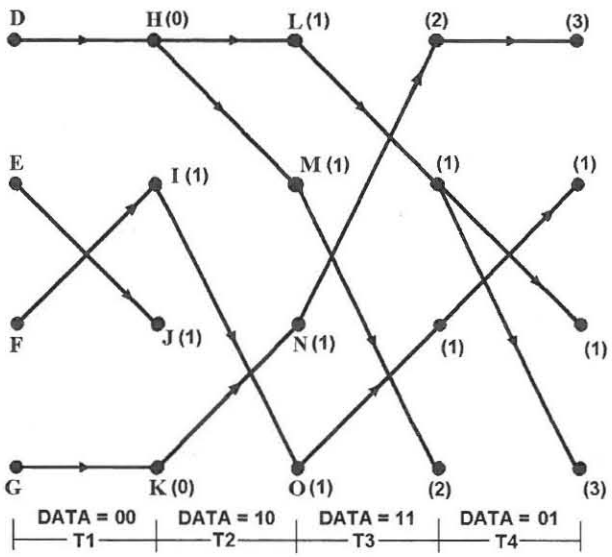


Fig. 2.21 : Wennerpaaie vir ontvangde 00101101

Ten spyte van hierdie tekortkoming van die betrokke kode kan die Viterbi dekoderingsproses se foutregstellende eienskappe waargeneem word. Dit kan dus gesê word dat die Viterbi dekoderingsproses foutregstelling bewerkstellig deur die reëls van die spesifieke konvolusiekode toe te pas oor 'n aantal ontvangde kodewoorde.

Soos hierbo genoem is kan Viterbi dekodering van sagtebesluitneming of hardebesluitneming gebruik maak. Sagtebesluitneming is 'n oppervlakkige modifikasie van die proses soos uiteengesit (hardebesluitneming Viterbi dekodering). Die Hammingafstand word eenvoudig vervang deur 'n sagtebesluitnemingsafstand/Euklidiese afstand (sien paragraaf 2.4.3). Verder bly die res van die dekoderingstappe dieselfde *[6, p. 235]*.

Die voordeel van sagtebesluitneming Viterbi dekodering lê in die feit dat die afstand waardes gebruik in die dekoderingsproses nie net dui op die ontvangde data nie, maar ook op die mate van verwringing veroorsaak deur ruis op die ontvangde sein. Deur gebruik te maak van hierdie waardes (Euklidiese afstande) kan 'n definitiewe wins - tipies 2dB vir n=3 en 2.2dB vir n=4 - t.o.v. hardebesluitneming dekodering verkry word *[22, p. v]*.

2.4.2.2 *Implementering van die Viterbi algoritme*

Twee belangrike aspekte wat in ag geneem moet word met die implementering van die Viterbi algoritme is as volg :

- Dekodeerder geheue

  Vir 'n enkodeerder met x state moet die ooreenkomstige Viterbi dekodeerder x woorde kan stoor, een vir elke wennerstap. Die woord wat gestoor word, moet in staat wees om beide die wennerpad en die padafstand tot op daardie oomblik te kan stoor. Aangesien die geheue benodigdhede eksponensieel vermeerder met die aantal state word 256 state as die praktiese limiet vir die Viterbi algoritme beskou *[13, p. 337]*.

- Pad geheue

Indien 'n datastring van L simbole lank ontvang word vir dekodering m.b.v. die optimum Viterbi algoritme, word x (sien hierbo) dekodeerder geheuewoorde gestoor vir elke simbool in die ontvangde L kodewoord string. Eers as die totale L kodewoorde ontvang is, word die wennerpad met die kortste afstand gekies om sodoende die ooreenkomstige datawoord uitset te lewer. A.g.v. beperking op die beskikbare rekenaar geheuekapasiteit is dit onmoontlik om die onderskeie L simbool paaie en hul afstande deur die trellis vir lang kodewoord stringe te stoor voordat die gedekodeerde datawoord string uitgelees kan word (soos in paragraaf 2.4.2.1 gebruik is tydens verduidelikking van die Viterbi dekoderingsproses).

'n Algemene oplossing in gebruik is om die y mees onlangs ontvangde kodewoorde vir elke pad te stoor en 'n besluit t.o.v. die wennerpad op een van die volgende drie maniere te neem :

1. 'n **Arbitrêre wennerpad** word gekies en die eerste k-bis (soos gedefinieer in paragraaf 2.4) datawoord in hierdie pad word uitgelees as die gedekodeerde bisse.
2. Van die x moontlike paaie se eerste k-bis datawoorde word die datawoord wat die **meeste in al die paaie voorkom** gekies.
3. Die wennerpad met die **kortste afstand** word gekies en die eerste k-bis datawoord in hierdie pad word uitgelees as die gedekodeerde bisse.

Nadat die datawoord uitgelees is, word die y-simbool venster opgedateer of aangeskuif met die volgende ontvangde simbool. Die besluit t.o.v. die wennerpad word weer herhaal om die ooreenkomstige uitset datawoord, y simbole terug in die ontvangde kodewoord string, uit te lees.

Die tipiese lengte y of vertraging waaroor praktiese Viterbi algoritmes werk, is 4 tot 5 keer die konstante lengte van die kode $(y = 5 \times m)$. Dit is d.m.v. rekenaar simulasies bewys dat hierdie lengte 'n weglaatbare degradasie in prestasievermoë het in vergelyking met die optimum Viterbi algoritme *[18, p. 459]*.

2.4.3 Trelliskodemodulasie

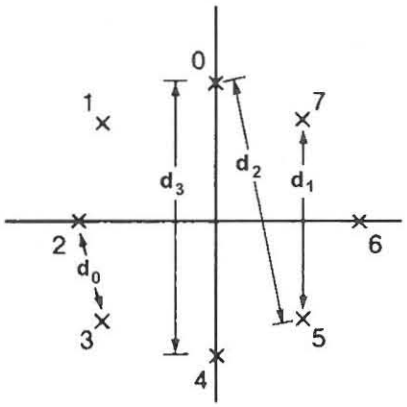
Normaalweg vind die kodering van data apart van die modulasieproses in 'n sender plaas. Foutregstelling word bewerkstellig deur van kodes gebruik te maak wat by die sender addisionele bisse tot die datastroom byvoeg en 'n negatiewe invloed op die bistempo per kanaalbandwydte tot gevolg het. Om hierdie verhouding van foutregstelling en bandwydte effektiwiteit te optimaliseer moet die enkodering en modulasieproses as 'n eenheid behandel word *[3, p. 67]*.

Die kombinasie van konvolusie kodering en modulasie staan bekend as trelliskodemodulasie (TCM) en is deur Dr. Gottfried Ungerboeck ontwikkel. In die ontwerp van Trellis kodes word die klem gelê op die Euklidiese afstand tussen kodevektore, en nie op die optimalisering van die Hammingafstand tussen kodewoorde soos in binêre foutregstellingskodes nie.

TCM voeg oortolligheid tot die datasein, deur die aantal vlakke in 'n veelvlak modulasie-proses te verdubbel. Waar 'n M-veelvlak sein $M = 2^k$ moontlike seine lewer om 'n sekere bistempo oor te dra, sal Ungerboeck se tegniek $M = 2^{k+1}$ moontlike simbole lewer om dieselfde bistempo oor te dra *[20, p. 697]*. Dus vir QPSK is die ooreenstemmende trelliskodemodulasie stelsel se fasevariasies soortgelyk aan 8PSK.

Sonder kodering is die prestasievermoë van 'n 8PSK stelsel afhanklik van die afstand d_0 (sien fig.2.22), wat 'n verswakking is t.o.v. QPSK se d_1 afstand tussen seinpunte. Die doel met trellis enkodering is om die minimum vry Euklidiese afstand tussen moontlike seine te vergroot om sodoende die stelsel se prestasievermoë te verbeter.

Die basis van trelliskodemodulasie is die opdeel van 'n sekere konstellasie in subgroepe van 2,4,8,..ens. met 'n groeiende minimum Euklidiese afstand tussen die individuele punte in elke groep, sodat daar 'n maksimum Euklidiese afstand tussen die moontlike punte in elke groep is (sien fig.2.23) *[12, p. 339]*. Hierdie proses staan bekend as plasing d.m.v. stelverdeling.



$d_3 = 2$ $d_2 = 1.852$ $d_1 = \sqrt{2}$ $d_0 = 0.765$

Fig. 2.22 : Euklidiese afstande in 8PSK konstellasie

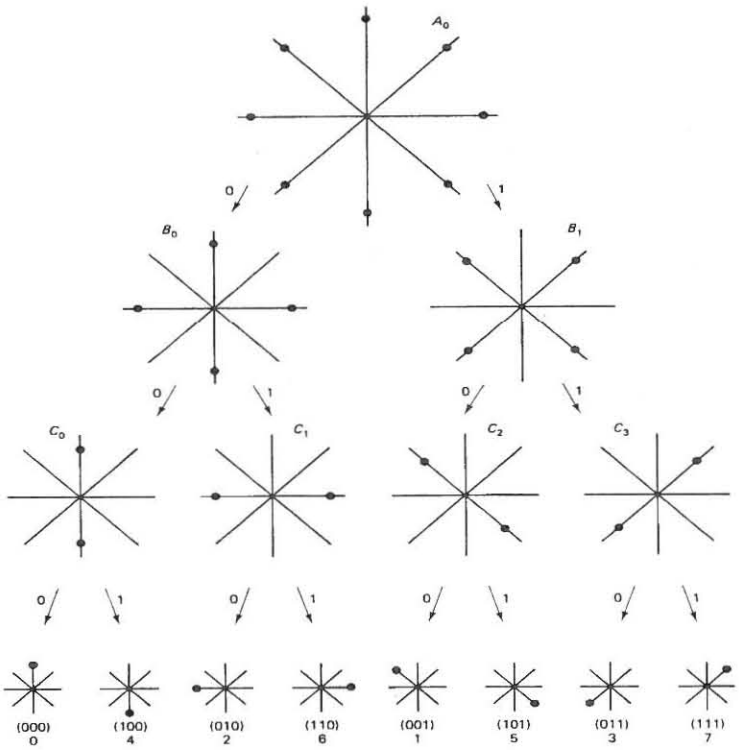


Fig. 2.23 : Stelverdeling van 'n 8-punt konstellasie

Die stelverdeling word so gedoen dat die groepe onderskeidelik ooreenstem met die moontlike stappe (vanaf 'n sekere staat) in 'n konvolusiekode se trellis. Fig.2.24 is 'n voorbeeld van 'n 2-staat enkodeerder se trellis met ooreenstemming t.o.v. die 8PSK stelverdeling in fig.2.23. Die twee groepe 0,2,4,6 (B_0) en 1,3,5,7 (B_1) in fig.2.23 stem ooreen met die moontlike stappe vanaf die onderskeie state in die trellis.

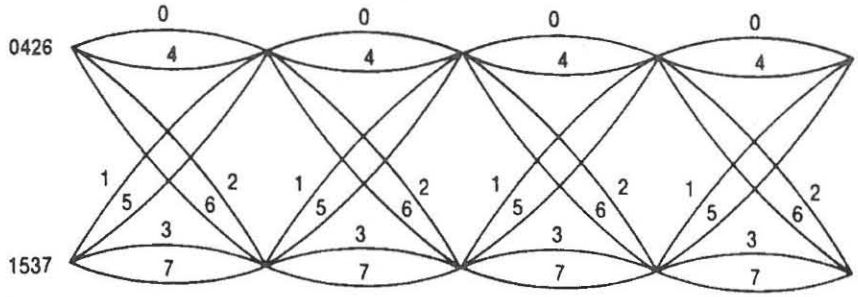


Fig. 2.24 : 2-Staat trellis

Die trellis toon twee state. Die enkodeerder kan in dieselfde staat bly deur 'n (0 of 4) of 'n (3 of 7) te stuur afhangend van die staat waarin die enkodeerder verkeer. Soortgelyk moet 'n (2 of 6) of 'n (1 of 5) gestuur word om die enkodeerder van staat te laat verander.

Indien 'n fout sou voorkom beteken dit dat die dekodeerder 'n ander pad sal volg. As net nulle gestuur word, is die moontlike afwykings in die ontvangde volgorde vir 'n enkele fout as volg: (0,4,0,0), (0,2,1,0), (0,2,5,0), (0,6,1,0) of (0,6,5,0) (sien fig.2.25).

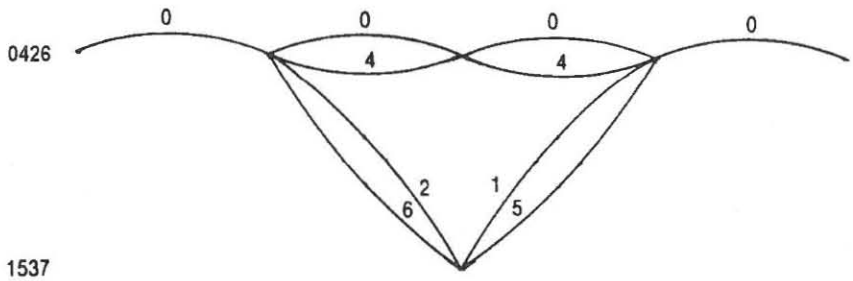


Fig. 2.25 : Moontlike foute in 2-staat trellis

Die Euklidiese afstand vir die fout (0,4,0,0) t.o.v. die korrekte pad is:

(0,4,0,0) : $$(s_0 - s_4) + (s_4 - s_0) = \sqrt{(d_3)^2 + (d_3)^2} = 2.83$$

waar d_3 die afstand in fig.2.22 is vir 'n faseverandering van seinpunt-0 na seinpunt-4. Net so is die ooreenkomstige afstande vir die ander afwykings :

(0,2,1,0) : $$(s_0 - s_2) + (s_1 - s_0) = 1.608$$

(0,2,5,0) : $$(s_0 - s_2) + (s_5 - s_0) = 2.33$$

(0,6,1,0) : $(s_0 - s_6) + (s_1 - s_0) = 1.608$

(0,6,5,0) : $(s_0 - s_6) + (s_5 - s_0) = 2.33$

Die (0,2,1,0) en (0,6,1,0) afwykings is dus die waarskynlikste foute wat sal plaasvind omdat hulle die naaste aan die gestuurde 0000 is in terme van Euklidiese afstand. Hierdie kleinste waarde is die minimum Euklidiese afstand vir die trellis in fig.2.24. As hierdie afstand met die van die ongekodeerde QPSK se minimum Euklidiese afstand tussen aangrensende seinpunte $(\sqrt{2})$ vergelyk word, is daar 'n verbetering van 1.608/1.414 of 'n wins van,

$$G = 10Log\frac{(1.608)^2}{(1.414)^2} = 1.1dB\,.$$

'n Verdere verbetering is moontlik deur die state van die trellis enkodeerder te vermeerder. In vergelyking met ongekodeerde QPSK (4PSK) is die verbetering in prestasievermoë onderskeidelik 3dB vir 'n 4-staat, 3.6dB vir 'n 8-staat en 4.1dB vir 'n 16-staat enkodeerder *[3, p. 200]*.

2.5 OPSOMMING

In die tradisionele benadering tot voorwaartse foutregstelling word die koderingsproses apart van die modulasieproses uitgevoer en die bandwydte effektiwiteit word verminder deur die toevoeging van oortolligheid.

Met trelliskodemodulasie word kodering en modulasie gekombineer om beduidende koderingswinste t.o.v. gewone ongekodeerde veelvlak modulasie te lewer - sonder 'n verhoging in bandwydte. Trelliskodes soos deur Ungerboeck beskryf kombineer veelvlak fasemodulasie met konvolusiekodes. By die ontvanger word trelliskode seine gedekodeer m.b.v. sagtebesluitneming Viterbi dekodering.

Die kombinasie van hierdie twee prosesse (trelliskodering en Viterbi dekodering) lewer koderingswinste wat die prestasie van kommunikasiestelsels in kanale met beperkte bandwydte verbeter.

3. DIGITALE SEINPROSESSERING

3.1 INLEIDING

Digitale seinprosessering (DSP) is die digitale verteenwoordiging van seine en die gebruik van digitale prosesseerders om die sein mee te analiseer, te verander en om inligting daaruit te verkry. DSP word in verskeie gebiede waar analoog stelsels in die verlede gebruik was, gebruik, asook in nuwe toepassings wat voorheen onmoontlik was om met analoog stelsels te realiseer. Die volgende voordele maak DSP baie aantreklik vir gebruik.

- Gewaarborgde akkuraatheid afhangend van die aantal bisse/woordlengte van die prosesseerder - wat die akkuraatheid van die stelsel bepaal.
- Perfekte herhaalbaarheid is moontlik omdat daar geen variasies is a.g.v. komponenttoleransies nie.
- Geen afwyking in prestasievermoë a.g.v. temperatuur en veroudering van komponente.
- Die voordeel van voortdurende ontwikkeling in halfgeleiertegnologie t.o.v. betroubaarheid, kleiner fisiese grootte, laer koste, laer kragverbruik en hoër spoed.
- DSP stelsels kan geprogrammeer en hergeprogrammeer word om 'n verskeidenheid van funksies uit te voer sonder om die hardeware te verander.
- DSP kan gebruik word om funksies te verrig wat nie moontlik is met analoog stelsels nie.

DSP is onderhewig aan nadele, maar die voortdurende ontwikkeling in tegnologie is vinnig besig om dit uit te wis. Die huidige nadele is as volg:

- DSP ontwerpe is duur - veral in hoë spoed of groot bandwydte toepassings.
- Die ontwerptyd is baie lank, veral as die regte sagteware en toerusting nie beskikbaar is nie.
- In stelsels met 'n relatiewe kort woordlengte vind 'n ernstige degradasie in prestasievermoë plaas.

[11, p. 2]

3.2 DIGITALISERING

In ‘n DSP stelsel is daar dikwels ‘n omskakeling vanaf ‘n analoog sein na ‘n digitale sein of omgekeerd. Die volgende onderwerpe is van groot belang t.o.v. analoog-na-digitaal en digitaal-na-analoog omskakeling.

3.2.1 Monstering

Monstering is die eerste proses in die omskakeling van ‘n analoog sein na ‘n digitale sein. Na aanleiding van die Nyquist en/of Shannon monsterteorie kan gesê word dat indien die frekwensie elemente in 'n aaneenlopende sein strek vanaf 0 tot W Hz, kan die sein ten volle voorgestel of herbou word vanaf 'n reeks eenvormig gespasieërde monsters, indien die monsterfrekwensie $2W$ monsters per sekonde oorskry *[5, p. 44]*.

Die monsters wat 'n sein voorstel kan gesien word as die uitset van 'n impulsmodulator (met eenvormig gespasieerde impulse teen f_s) as die draaggolf, wat gemoduleer word met die analoog sein met 'n maksimum frekwensie f_B (sien fig.3.1) *[1, p. 249]*.

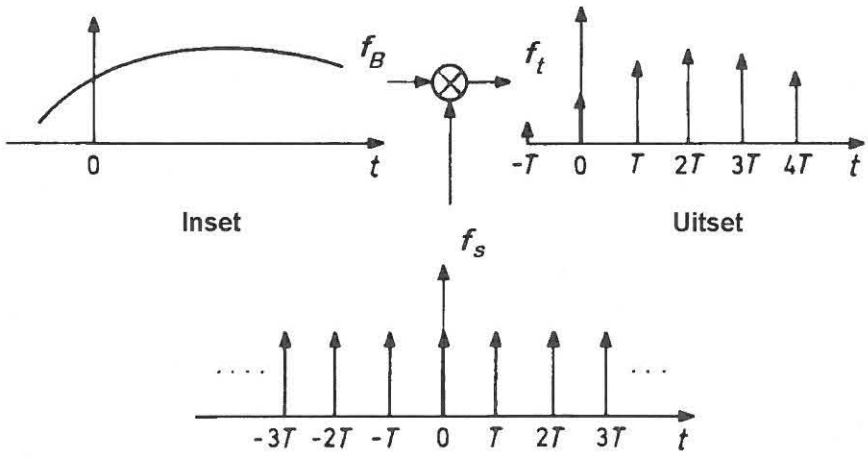


Fig. 3.1 : Monstering as impulsmodulasie.

Hierdie vermenigvuldiging van die twee seine met mekaar in die tyd-as lewer ‘n periodiese spektrum in die frekwensie-as met 'n periode gelyk aan die monsterfrekwensie *[10, p. 136]*. Fig.3.2 *[1, p. 251]* is ‘n voorbeeld van die impulsmodulator se uitset in die frekwensie-as vir verskillende verhoudings van f_s tot f_B.

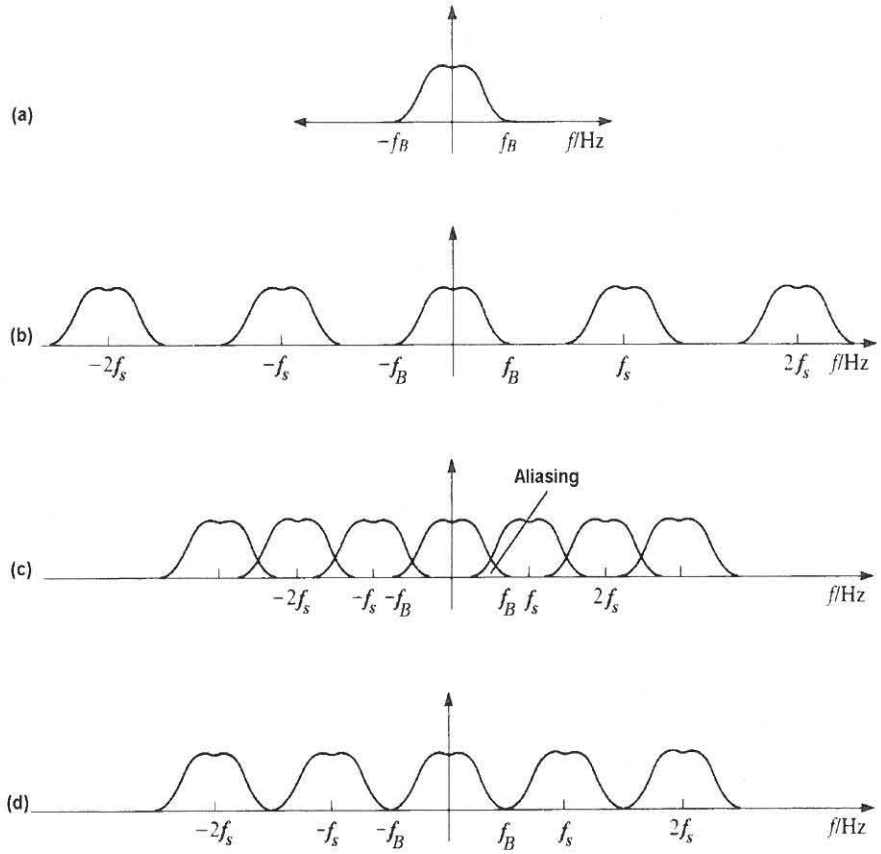


Fig. 3.2 : Die frekwensiespektrum vir (a) die basisbandsein, (b) oor-bemonstering $(f_s > 2f_B)$**, (c) onder-bemonstering** $(f_s < 2f_B)$**, (d) kritiese bemonstering** $(f_s = 2f_B)$**.**

Uit fig.3.2 kan die draaggolfharmonieke met 'n bandwydte van $2f_B$ waargeneem word. As die monsterfrekwensie verlaag word tot onder $2f_B$, soos in fig.3.2(c), sal die frekwensiespektrums oorvleuel wat die verskynsel van "aliasing" laat voorkom. "Aliasing" veroorsaak distorsie wat dit onmoontlik maak om die ware sein by die ontvanger te herwin. Dit kan dus voorkom word deur die basisbandsein se bandwydte te beperk tot $\leq f_s/2$ of deur 'n monsterfrekwensie van $> 2f_B$ te gebruik.

3.2.2 Kwantifisering

Na monstering is die volgende stap in die analoog na digitale omskakeling, die kwantifisering van die monsters. Elke monster van 'n vaste waarde word in een van 'n eindige reeks vlakke omgeskakel.

As gevolg van die eindige reeks vlakke kan 'n analoog sein in die algemeen nie presies gekodeer word nie en moet dus afgerond word tot die naaste moontlike vlak (sien fig.3.3(a)) *[5, p. 238]*.

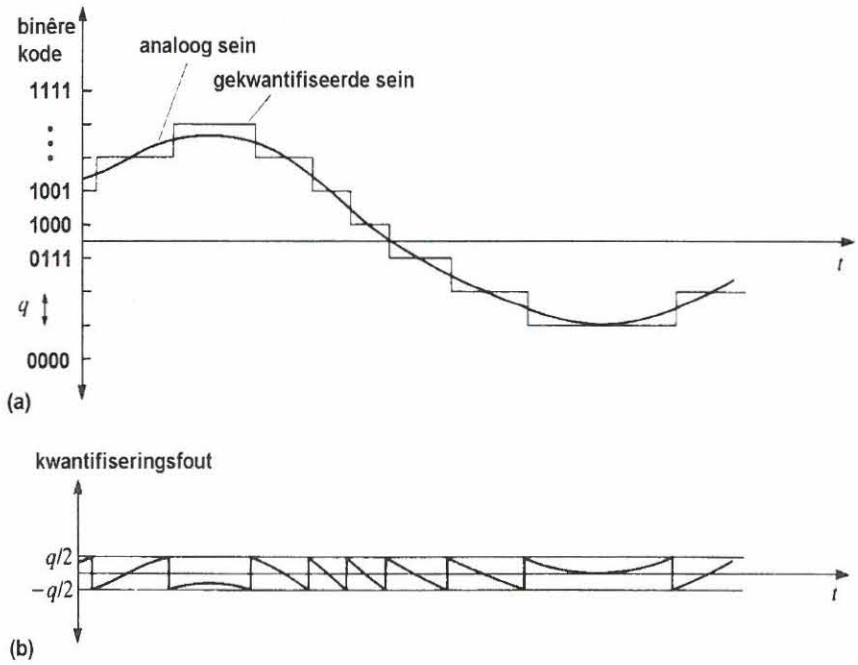


Fig. 3.3 : Kwantifiseringsproses van 'n 4-bis A-na-D omskakelaar.

Hierdie fout wat ontstaan a.g.v. afronding staan bekend as kwantifiseringsruis (sien fig.3.3(b)). Die seinkwantifiseringsruisverhouding word deur die volgende vergelyking voorgestel.

$$\frac{S}{N} = \frac{sein}{kwantifiseringsruis}$$

Hierdie verhouding verhoog met +/- 6dB vir elke bis vermeerdering in die woordlengte van die kwantifiseringsvlakkode *[8, p. 10]*. Dus moet daar so veel moontlik kwantifiseringsvlakke wees om die kwantifiseringsruis se invloed tot 'n minimum te beperk.

3.3 DSP56001 SEINPROSESSEERDER.

Die DSP56001 prosesseerder is gekies vir die implementering van die enkodering en modulasie in die sender asook die sinchronisering, demodulasie en dekodering in die ontvanger. Hierdie prosesseerder se geskiktheid vir kommunikasie toepassings kan aan die hand van die volgende voordele beskou word.

- Verwerkingspoed van 10.25 miljoen instruksies per sekonde (MIPS).
- Die datawoorde is 24-bisse lank.
- Parallelle argitektuur - wat beteken dat die rekenkundige en logiese eenheid, adresgenereringseenheid en programbeheerder almal gelyktydig in parallel werk.
- Hoë vlak van integrasie - buiten die drie verwerkingseenhede is daar geheue, drie koppelvlakke, 'n klokgenerator en sewe busse (drie adres en vier data) aanboord van die DSP56001.
- Program ontwikkeling is moontlik in beide saamstel - of hoëvlak taal.
- 62 Woord instruksiestel met 'n "doen vir" (DO loop) en "herhaal" (REPEAT) instruksie wat die skryf van reguitlyn kode onnodig maak.
- 'n Ingeboude sinusgolf opsoektabel vir die implementering van draaggolfmodulasie.

[14, p. 1-8]

3.3.1 Kortlikse oorsig van die DSP56001 argitektuur.

Fig.3.4 *[14, p. 2-2]* is 'n blokdiagram van die DSP56001 met 'n kort uiteensetting van die onderskeie dele daarvan in die volgende paragrawe.

Databusse

Die beweeg van data in die prosesseerder vind plaas m.b.v. vier bidireksionele 24-bis databusse nl: X-databus (XDB) en Y-databus (YDB) (oordra van data tussen X en Y geheuelokasies), Program databus (PDB) (oordra van program instruksies) en die Globale databus (GDB) (alle ander data oordrag, asook in- en uitset kommunikasie met die koppelvlakke). Die oordrag van data tussen databusse geskied in die interne busskakelaar. Al hierdie databusse word na die eksterne 24-bis databus gemultiplekseer.

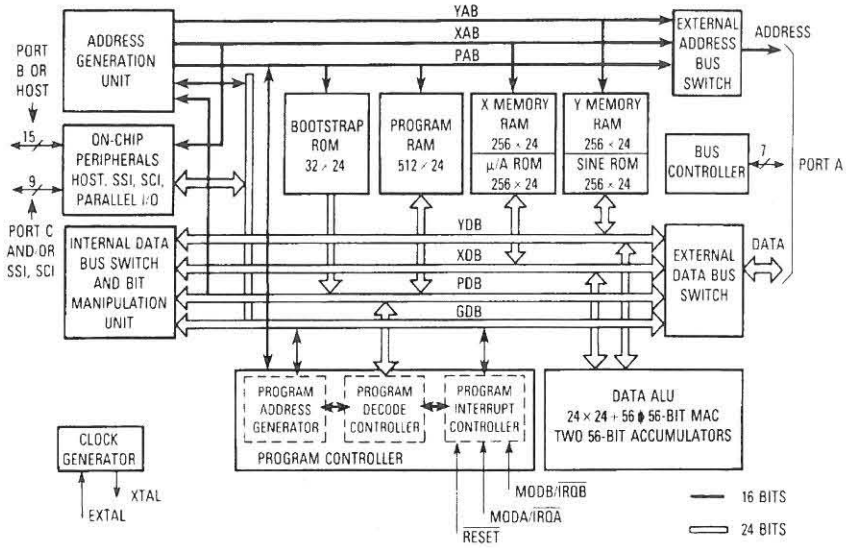


Fig. 3.4 : Blokdiagram van die DSP56001 prosesseerder.

Adresbusse

Vir die interne X- en Y- data geheuelokasies word die adresse daarvan gespesifiseer deur twee eenrigting 16-bis busse (XAB en YAB), en die programgeheue adresse deur 'n 16-bis adresbus (PAB). Al drie adresbusse word ook gemultiplekseer na die eksterne adresbus.

Rekenkundige en logiese eenheid (RLE)

Die data RLE doen al die wiskundige en logiese bewerkinge d.m.v. twee-se-komplement berekeninge. Die data ALU het vier 24-bis inset registers (X0, X1, Y0, Y1) en twee 56-bis akkumulator registers (A, B) wat onderskeidelik kan opdeel in twee 24-bis registers (A0, A1, B0, B1) en 'n 8-bis verlengingsregister (A2,B2).

Adresgenereringseenheid

Die adresgenereringseenheid bestaan uit drie groepe van 8 registers elk nl: adresregisters (R0 - R7), afstel registers (N0 - N7) en aanpassingsregisters (M0 - M7). Elkeen van bogenoemde registers is 16 bisse lank.

X en Y Datageheue

Die aanboordgeheue van die prosesseerder neem die eerste 512 geheue lokasies van die X data geheue in beslag waarvan 256 lokasies deur 'n 24-bis wye ROM gevul word wat vooraf met μ-wet en A-wet opsoektabelle geprogrammeer is. Die

X data geheue kan vergroot word deur eksterne geheue toe te voeg tot by 'n maksimum van 65536 lokasies.

Die Y data geheue is soortgelyk aan die X data geheue behalwe dat die ROM spasie 'n 256-punt sinusgolf opsoektabel bevat.

Programgeheue

Die aanboord programgeheue bestaan uit 512, 24-bis wye RAM geheuelokasies wat verleng kan word tot 65536 lokasies d.m.v. eksterne uitbreiding.

Poorte

Poort A (die uitbreidingspoort) bestaan uit 'n gewone 24-bis databus vir koppeling met eksterne RAM of ander koppelvlaktoestelle.

Poort B of die gasheer koppelvlak bestaan uit 'n 8-bis voldupleks poort met sewe kontrolelyne vir die beheer van data-oordrag tussen die DSP56001 en 'n gasheer rekenaar.

Poort C bestaan uit 'n seriale kommunikasiekoppelvlak (SCI) en 'n sinchrone seriale koppelvlak (SSI). Die SCI is 'n voldupleks poort vir 8-bis seriale datakommunikasie met ander DSP's of modems. Die SSI is 'n meer aanpasbare seriale poort waarvan die karakteristieke deur die gebruiker geprogrammeer kan word - vir kommunikasie met 'n verskeidenheid van seriale toestelle.

3.3.2 Peralex SIG56 DSP kaart

Twee Peralex SIG56 digitale seinprosesseringskaarte is gebruik in hierdie projek, een vir die sender en een vir die ontvanger.

Die SIG56 kaarte is ontwerp om in die uitbreidingspoort van 'n IBM aanpasbare rekenaar te prop. Die kaart maak gebruik van die bogenoemde DSP56001 prosesseerder en 'n TLC32044 analoog koppelvlakkring vir A-na-D insette en D-na-A uitsette vir die stelsel. Eksterne geheue is tot die DSP56001 op die kaart bygevoeg om 'n totaal van 32K woorde programgeheue, 16K X-geheue en 16K Y-geheue te lewer *[16, p. 10]*.

Pascal drywer sagteware is saam met die SIG56 kaart ontvang om enige datavloei tussen die kaart en gasheer rekenaar te hanteer.

In die gebruik van die SIG56 kaart is dit nodig om die werking van die analoog koppelvlak (TLC32044) te verstaan, asook die opstel daarvan om verskillende in- en uitset filters teen verskillende monsterfrekwensies te realiseer.

3.3.3 TLC32044 Analoog koppelvlakkring

Die TLC32044 koppelvlak is 'n volledige A-na-D en D-na-A inset/uitset stelsel in een geïntegreerde stroombaan. Fig.3.5 *[24, p.3]* is 'n blokdiagram van die TLC32044.

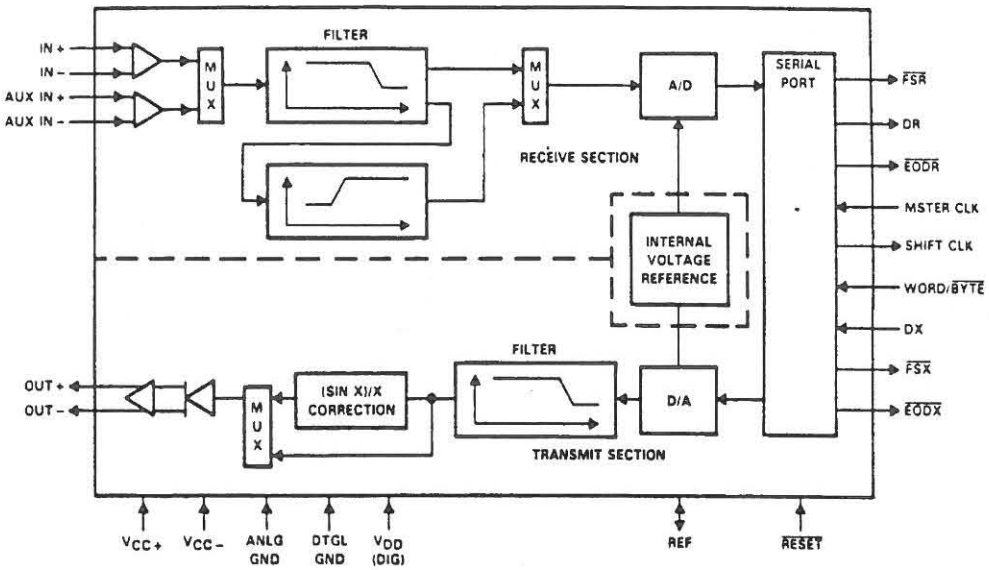


Fig. 3.5 : TLC32044 blokdiagram

Seriale poort

Die seriale poort dra die seriale data oor na, of lees dit van die digitale seinprosesseerder, en kan beheer word d.m.v. sagteware.

Omskakelaars

Die A-na-D en D-na-A omskakelaars het elk 'n resolusie van 14 bisse met 'n interne spanningsverwysing vir stabiliteit.

Insetfilter

Die inset banddeurlaatfilter bestaan uit 'n agtste orde onderdeurlaat - en vierde orde bodeurlaat Chebychev filters. Die banddeurlaatfilter is d.m.v. skakelkapasitor-tegnologie geïmplementeer en kan deur sagteware aan- of afgeskakel word.

Fig.3.6 *[24, p. 29]* toon die basiese frekwensieweergawe van bogenoemde banddeurlaatfilter. Hierdie karakteristiek kan m.b.v. sagteware aangepas word.

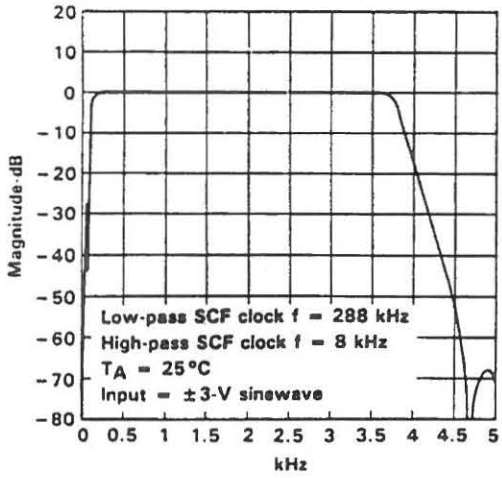


Fig. 3.6 : Frekwensieweergawe van TLC32044 banddeurlaatfilter

Hierdie weergawe word gelewer met die skakelkapasitorfilter (SCF) klokfrekwensie (f_{SCF}) gelyk aan 288kHz met 'n 8kHz monsterfrekwensie. As f_{SCF} nie 288kHz is nie word die frekwensieweergawe se boonste afsnyfrekwensie geskaal t.o.v. die verhouding van f_{SCF} tot 288kHz en kan uitgedruk word as:

$$\text{Boonste afsnyfrekwensie} = f_N \times \frac{f_{SCF}}{288kHz} \qquad (3.1)$$

waar (f_N) die genormaliseerde afsnyfrekwensie is. Die onderste afsnyfrekwensie van die deurlaatband is 100Hz.

Die interne tydkonfigurasie van die analoog koppelvlakkring word in fig.3.7 *[24, p. 9]* getoon met verskeie opsies om 'n 288kHz skakelkapasitorfilter klokfrekwensie, banddeurlaatfilter te realiseer deur die ontvangteller-A (RA) waarde

ooreenkomstig te verander vir twee meesterklokfrekwensies (f_M). f_M is 5.184MHz op die SIG56 kaart en f_{SCF} kan as volg bereken word:

$$f_{SCF} = \frac{f_M}{2 \times (tellerA)} \qquad (3.2)$$

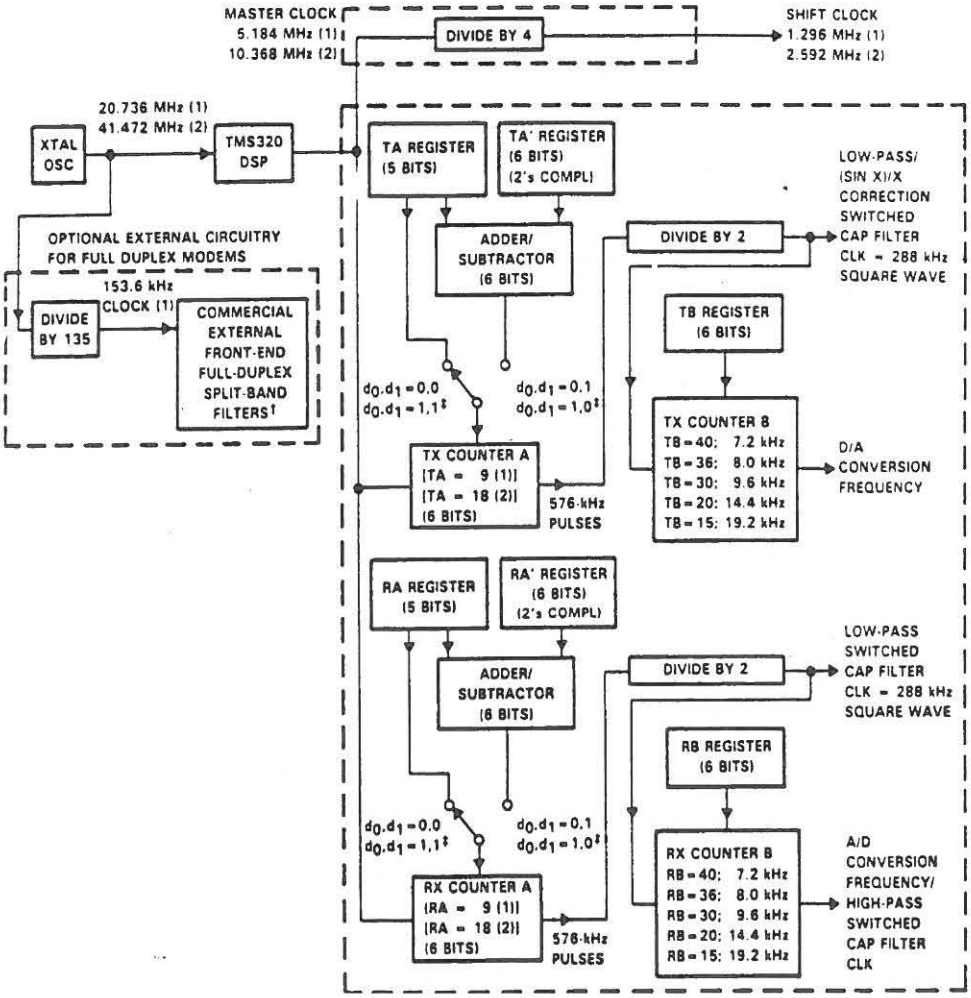


Fig. 3.7 : Konfigurasie van die TLC32044 koppelvlakkring

Die A-na-D omskakelingstempo, of monsterfrekwensie, f_s, word verkry deur die skakelkapasitorfilter klokfrekwensie te deel deur die inhoud van die ontvangteller-B (RB).

$$f_s = \frac{f_{SCF}}{(tellerB)} \qquad (3.3)$$

Uitset filter

Die uitset herkonstruksiefilter is 'n agtste orde onderdeurlaat Chebychev filter, ook in skakelkapasitortegnologie geïmplementeer.

Die frekwensieweergawe van die onderdeurlaatfilter word in fig.3.8 *[24, p. 29]* getoon.

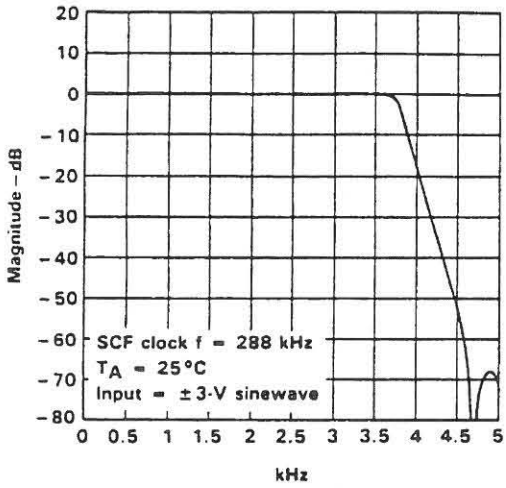


Fig. 3.8 : Frekwensieweergawe van TLC32044 onderdeurlaatfilter

Die boonste afsnyfrekwensie vir die onderdeurlaatfilter word op dieselfde wyse as by die insetfilter bepaal deur vergelyking (3.1) behalwe dat die sendteller-A (TA) gebruik word. Die sendteller-B (TB) word in verg.(3.3) gebruik om die D-na-A omskakelingstempo te bepaal.

(Sin x)/x korreksiefilter

Die (sin x)/x korreksie van die D-na-A se uitset word deur hierdie filter gedoen en volg die uitset van die skakelkapasitorfilter na. D.m.v. sagteware kan die (sin x)/x filter aan- of afgeskakel word.

3.3.4 Tipiese gebruik van die DSP56001 prosesseerder en SIG56 kaart

Hier volg 'n kortlikse uiteensetting van die tipiese stappe wat gevolg word om die SIG56 kaart te gebruik.

1. Skryf 'n program in saamsteltaal (bv. prog.asm) vir die Motorola DSP56001 om die verlangde proses uit te voer.
2. Die program word dan saamgestel waartydens dit omgeskakel word na DSP56001 uitvoerbare kode (bv. prog.lod).
3. Skryf 'n Pascal program wat op die gasheerrekenaar uitgevoer word en die vloei tussen die gasheer en die SIG56 gaan beheer.
4. Lees die uitvoerbare kode (prog.lod) in die DSP56001 in deur van die Peralex drywersagteware gebruik te maak.
5. Begin die Pascal program op die gasheer wat die DSP program inisieër en die nodige beheer uitoefen.

3.4 PROJEK OPSTELLING

Die projek is hoofsaaklik sagteware georiënteerd wat van die gasheer rekenaar en twee SIG56 kaarte gebruik maak. Fig.3.9 is 'n blokdiagram van die projek opstelling met die twee SIG56 kaarte in die gasheer rekenaar.

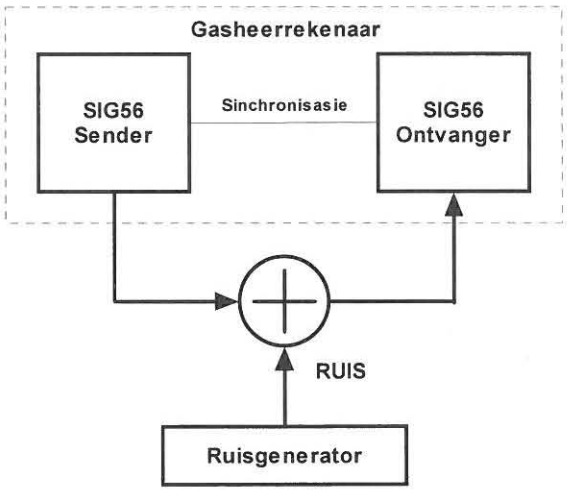


Fig. 3.9 : Blokdiagram van projek opstelling

Die gasheer rekenaar maak gebruik van 'n Pascal program vir die beheer van die DSP kaarte, asook die data vloei tussen die DSP kaarte en rekenaar. Die twee SIG56 kaarte word elk onderskeidelik met die sender en ontvanger saamteltaalprogramme gelaai. Die eksterne koppeling tussen die sender en ontvanger is met ruis besoedel tydens die evaluasie van die stelsel.

4. STELSEL ONTWERP & SIMULASIE

4.1 INLEIDING

Aan die begin van hierdie projek was die plan om 'n modem te lewer met die volgende spesifikasies:

1. Werking oor 'n spraakband telefoonlyn.
2. Datatempo van 3200 databisse per sekonde.
3. Gebruik van trelliskodemodulasie en Viterbi dekodering.
4. Algehele implementering van die stelsel d.m.v. DSP.

Dit het egter geblyk dat dit onmoontlik is om met die DSP kaart soos gebruik (en bespreek in hoofstuk 3) die Viterbi dekodering teen die verlangde tempo te verrig. Na deeglike oorweging van alle moontlikhede is besluit om die datatempo van die stelsel te verminder na ongeveer 1067 bisse per sekonde - met al die relevante waardes as volg ooreenkomstig afgeskaal.

Monsterfrekwensie	:	14400Hz	na	4800Hz
Draaggolffrekwensie	:	1800Hz	na	600Hz
Simbooltempo	:	1600 baud	na	533.33 baud

Alhoewel die datatempo verminder is, kon die primêre doel van hierdie projek nog steeds ondersoek word, nl. die gebruik van trelliskodemodulasie met 8DPSK en Viterbi dekodering in 'n DSP stelsel. Onderstaande is 'n bespreking van die stelsel soos ontwikkel.

4.2 SENDER

4.2.1 Blokdiagram van sender

Die werking van die sender kan blokdiagrammaties soos in fig.4.1 voorgestel word.

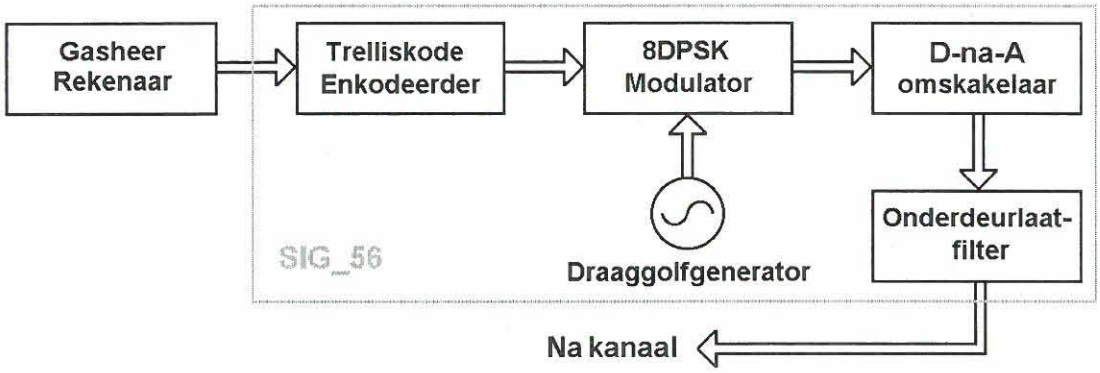


Fig. 4.1 : Blokdiagram van sender

4.2.1.1 Gasheerrekenaar

Die gasheerrekenaar ('n 486 DX4 is hiervoor gebruik) voorsien die DSP56001 van data in die vorm van 16-bis pseudo-ewekansige heelgetalle. Die heelgetalle is gegenereer d.m.v. 'n Pascal program en gestoor in 'n datalêer, vanwaar dit as 'n blok heelgetalle uitgelees word na die DSP56001 vir verwerking en uiteindelike transmissie.

4.2.1.2 Trelliskode enkodeerder

'n 2/3 Ungerboeck trelliskode enkodeerder *[28, p. 60]* is gebruik vir kanaalkodering en word in fig.4.2 getoon. Die enkodeerder se trellisdiagram word in bylae C getoon.

Die 16-bis heelgetalle data vanaf die gasheerrekenaar word in die DSP56001 opgebreek in bispare en ingelees in die enkodeerder. Modulus-2 sommering met voorafgaande bisse soos aangetoon in fig.4.2 vind plaas om 'n ge-enkodeerde 3-bis

uitset te lewer. Met 'n data bistempo van 1067 bisse per sekonde word 'n ge-enkodeerde simbooltempo van ongeveer 533 baud verkry - (vir elke twee data insetbisse lewer die enkodeerder, een 3-bis uitset simbool).

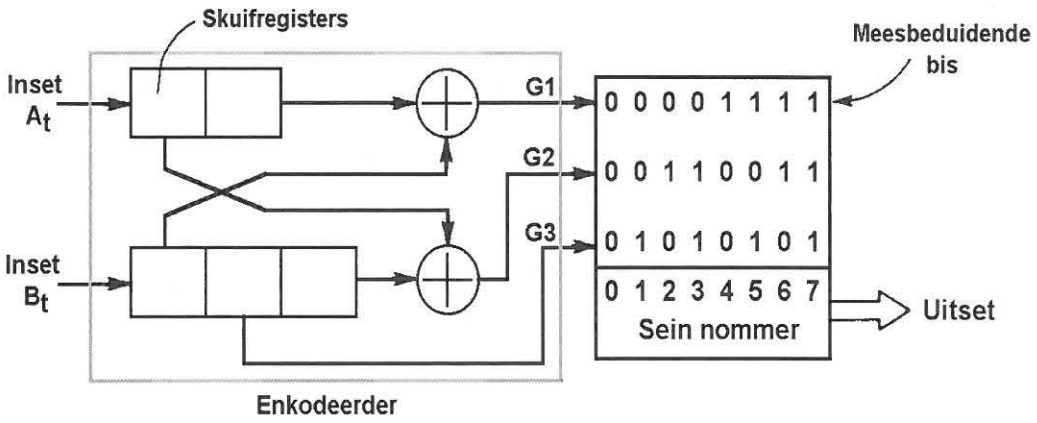


Fig. 4.2 : Ungerboeck 2/3 trelliskode enkodeerder

4.2.1.3 Draaggolfgenerator

A.g.v. die verhouding van die monsterfrekwensie tot die draaggolffrekwensie (8 : 1) kan een herhaling van die draaggolf deur agt monsters voorgestel word, d.w.s. een monster elke 45° van die sinusgolf soos tabel 4.1 toon.

Tabel 4.1 Sinusgolf monsters

Monster no.	Sinusgolffase	Amplitude
0	0°	0
1	45°	0.707
2	90°	1
3	135°	0.707
4	180°	0
5	225°	- 0.707
6	270°	- 1
7	315°	- 0.707

Deur hierdie agt monster waardes - sinusgolf amplitudes - opeenvolgend uit te lees word 'n aaneenlopende sinus draaggolf verkry (sien monsters 0 tot 15 in fig.4.3).

4.2.1.4 8DPSK modulator

Voor modulasie van die draaggolf is die drie-bis uitsetwaarde vanaf die enkodeerder omgeskakel na die ooreenkomstige desimale waarde om een van agt moontlike vlakke voor te stel. Hierdie omskakeling kan in die tabel op die uitset van die enkodeerder in fig.4.2 waargeneem word. Elk van hierdie vlakke word aan 'n differensiële fase gekoppel volgens die TCM bepalings. Tabel 4.2 toon die relatiewe fases aan.

Tabel 4.2 : 8DPSK simbool fases vir TCM

Binêre groepe	Basis-8 vlak	Fase
000	0	0°
001	1	45°
010	2	90°
011	3	135°
100	4	180°
101	5	225°
110	6	270°
111	7	315°

Die simbooltempo van 533 baud, teen 'n draaggolffrekwensie van 600Hz, noodsaak 9 monsters per baud. Elke negende monster word 'n nuwe datavlak - en relatiewe fase - versend.

$$\frac{600}{533.33} \times \frac{8}{1} = 9$$

In fig. 4.3 vind 'n faseverandering van 270° of sprong van ses monsters (t.o.v. tabel 4.1) plaas vanaf monster 17 na monster 18. Volgens tabel 4.2 verteenwoordig dit 'n enkodeerder uitset van $110_{(2)}$. Die nuwe fase word vir nege monsters (vanaf 18 tot

26) volgehou waarna die enkodeerder uitset opnuut 'n verandering in fase veroorsaak.

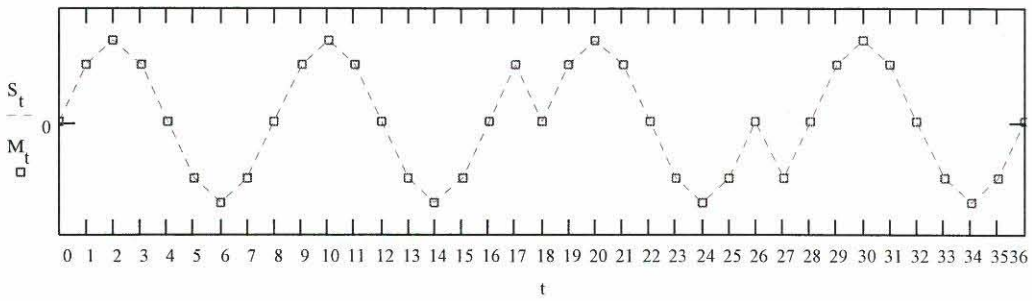


Fig. 4.3 Generering van 8DPSK sein.

4.2.1.5 D-na-A omskakelaar

Die TLC32044 D-na-A omskakelaar (in hoofstuk 3 bespreek) skakel die gemoduleerde sein om na 'n analoog sein vir transmissie op die kanaal en bepaal ook die tempo waarteen die monsters uitgelees word - 4800 monsters per sekonde in hierdie projek. Die waarde van register TB vir die D-na-A omskakelingsfrekwensie is bepaal deur eerstens die skakelkapasitorfrekwensie vir 'n onderdeurlaatfilter afsnyfrekwensie van 2040Hz (minder as die helfte van die monsterfrekwensie) uit vergelyking (3.1) te verkry.

$$f_{SCF} = \frac{2040Hz \times 288kHz}{3400Hz} = 172.8kHz$$

Met f_{SCF} gelyk aan 172.8kHz is TB uit vergelyking (3.3) bepaal

$$tellerB = \frac{f_{SCF}}{f_S} = \frac{172.8kHz}{4800Hz} = 36$$

4.2.1.6 *Onderdeurlaatfilter*

Die onderdeurlaatfilter, wat deel uitmaak van die TLC32044 D-na-A omskakelaar (soos bespreek in hoofstuk 3), verwyder die hoë frekwensie komponente van die gemoduleerde draaggolf voor transmissie. Vir 'n onderdeurlaat afsnyfrekwensie van 2040Hz, wat in die projek gebruik is, is die inhoud van register TA as volg vanaf vergelyking (3.2) bereken.

$$tellerA = \frac{f_M}{2 \times f_{SCF}} = \frac{5.184\,MHz}{2 \times 172.8kHz} = 15$$

4.2.1.7 *Kanaal*

'n Spraakband telefoonkanaal is as transmissiemedium gesimuleer met 'n deurlaatfrekwensie van 300Hz tot 3400Hz waarvan die boonste deurlaatfrekwensie aangepas is na 2040Hz a.g.v. die aanpassing van die stelsel genoem in punt 4.1.

4.2.2 Simulasie van sender

Voor die ontwerp en skryf van die sender sagteware is daar eers 'n wiskundige simulasie van die sender en sy onderskeie onderafdelings se werking gedoen. Die simulasie is om die volgende redes gedoen.

1. Ondersoek die uitvoerbaarheid van die stelsel.
2. Bepaling van die verwagte uitsetsein se golfvorm en frekwensiespektrum.

Die simulasie is d.m.v. 'n Mathcad 5.0 sagteware pakket gedoen en sal vervolgens aan die hand van die sender blokdiagram in fig.4.1 bespreek word. Alhoewel die simulasie gedoen is voor die bistempo aanpassing, is die verhouding van die monstertempo tot die draaggolffrekwensie en simbooltempo dieselfde as in die oorspronklike beplanning. Die relatiewe waardes in die tyd- en frekwensie- as is dus onveranderd en steeds korrek. In die bespreking word daar slegs na die resultate van die simulasie verwys. Die volledige sender simulasie is aangeheg as bylae A.1.

4.2.2.1 Gasheerrekenaar

Eerstens is pseudo-ewekansige binêre data gegenereer, wat die data vanaf die gasheerrekenaar voorstel (sien fig.4.4).

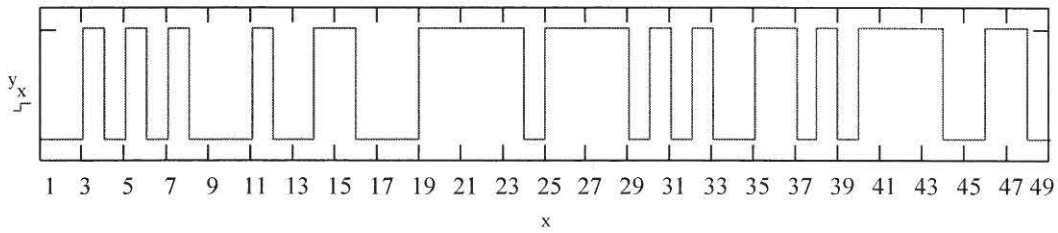


Fig. 4.4 : Pseudo-ewekansige data

4.2.2.2 Opdeel van datastroom

Die binêre datastroom is opgedeel in twee strome om die verlangde twee bisse vir die 2/3 trelliskode enkodeerder inset te lewer (sien fig.4.5).

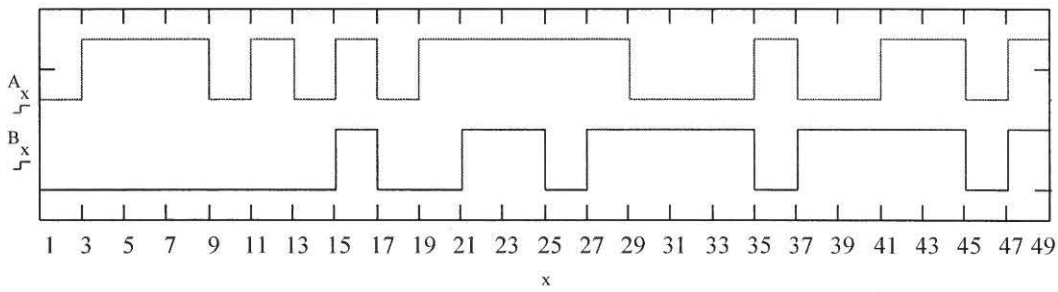


Fig. 4.5 : Twee datastrome as enkodeerder inset

4.2.2.3 Enkodeerder

Kodering is gedoen deur twee bisse per slag, een van elke datastroom, modulus-2 te sommeer met voorafgaande databisse soos deur die drie kodegenerators (G1, G2

en G3) in fig.4.2 getoon. Die drie generators lewer 'n ge-enkodeerde 3-bis uitset soos fig.4.6 aandui.

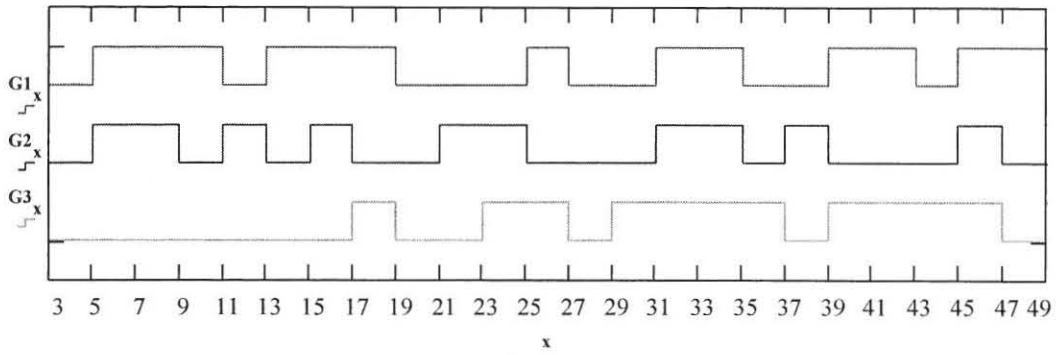


Fig. 4.6 : Enkodeerder 3-bis uitset

4.2.2.4 8DPSK Modulasie

Die 3-bis waardes word omgeskakel na hul ooreenkomstige desimale getal om die agt seinvlakke te verkry (sien fig.4.7). G1 is as die meesbeduidende bis en G3 as die minsbeduidende geneem.

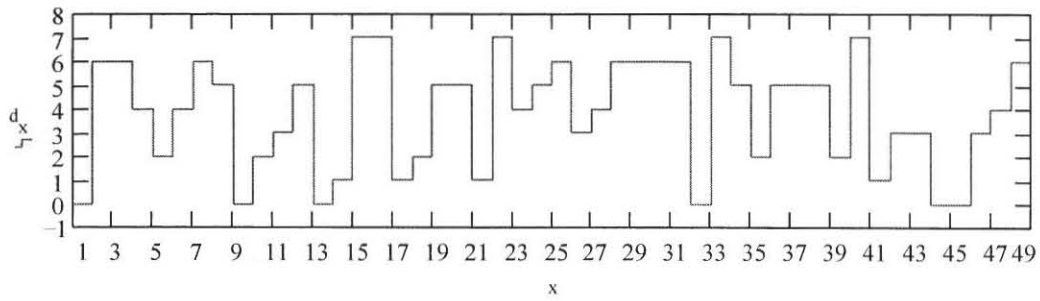


Fig. 4.7 : 8-Vlak sein as modulator inset

Aan die hand van tabel 4.2 is die agt vlakke elk aan een van die agt moontlike fases gekoppel. Fig. 4.8 toon die ooreenkomstige fase waardes van die digitale simbole in radiale aan.

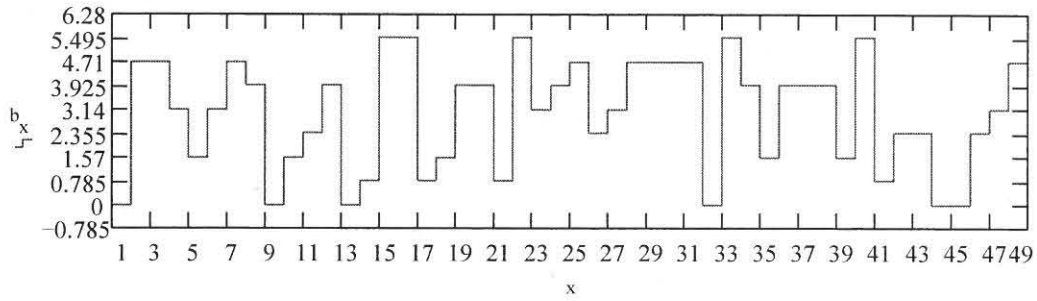


Fig. 4.8 : Verandering in draaggolffase a.g.v. modulasie

Die modulator uitset word verkry deur die faseveranderings in fig.4.8 diferensieël op 'n sinusgolf toe te pas. Fig.4.9 toon die gemoduleerde draaggolf met 'n faseverandering elke negende monster - of 405° - van die draaggolf.

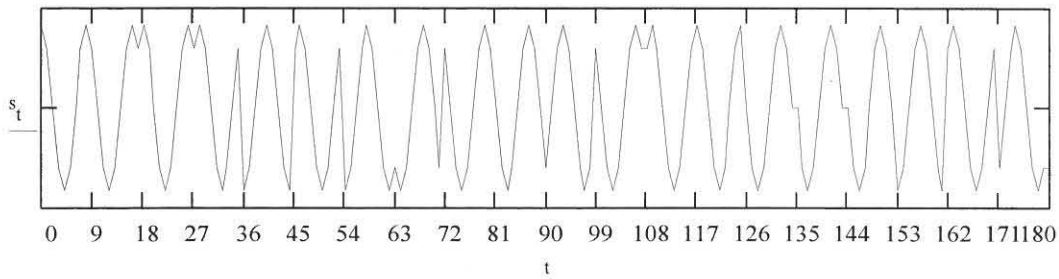


Fig. 4.9 : Gemoduleerde draaggolf

4.2.2.5 Onderdeurlaatfilter

Die effek van die onderdeurlaatfilter op die gemoduleerde sinusgolf is verkry deur eerstens 'n Fourier analise op die sein te doen wat 'n aanduiding gee van die sein se frekwensie samestelling. Die modulator uitset in die frekwensie-as kan in fig.4.10 waargeneem word waar die merkers 512 en 967 verhoudingsgewys die draaggolf frekwensie en die verlangde filter afsnyfrekwensie onderskeidelik aandui.

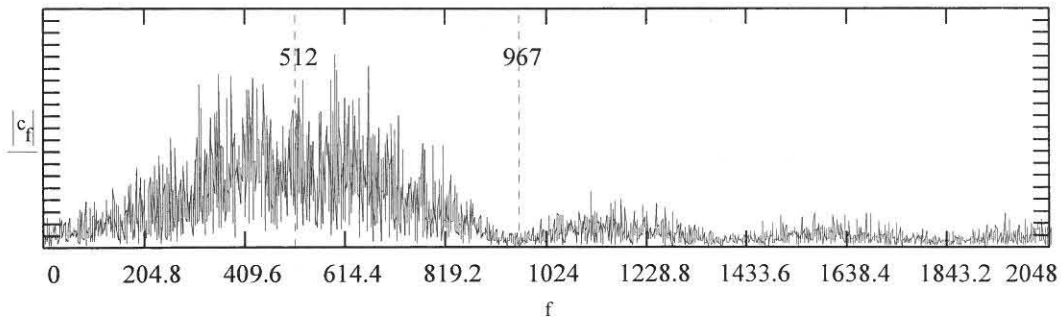


Fig. 4.10 : Frekwensiespektrum van gemoduleerde draaggolf

Die spektrale waardes van die sein is deur 'n onderdeurlaat Butterworth filter vergelyking verwerk. Fig.4.11(a) toon die filter karakteristiek aan en fig.4.11(b) die oorblywende frekwensie komponente van die 8DPSK sein nadat dit deur die filter verwerk is. Alhoewel die filter wat hier gebruik is nie dieselfde is as die Chebychev filter in die TLC32044 omskakelaar wat in die projek gebruik word nie, is beide 'n agtste-orde onderdeurlaatfilter, met 'n redelike ooreenstemming in hul karakteristieke.

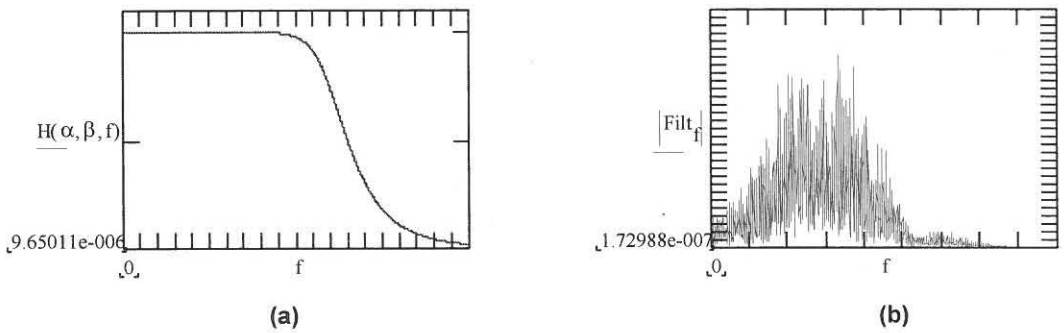


Fig. 4.11 : (a) Onderdeurlaatfilter karakteristiek, (b) frekwensiespektrum van gefilterde sein

Om die gemoduleerde sinusgolf golfvorm te herwin is 'n inverse Fourier transformasie op die oorblywende frekwensiespektrum (fig.4.11(b)) gedoen. Fig.4.12 toon die sender uitsetsein na die onderdeurlaatfilter.

'n Vergelyking van figure 4.9 en 4.12 toon duidelik die vervorming van die oorspronklike gemoduleerde sein a.g.v. die verwydering van die sein se hoë frekwensie komponente.

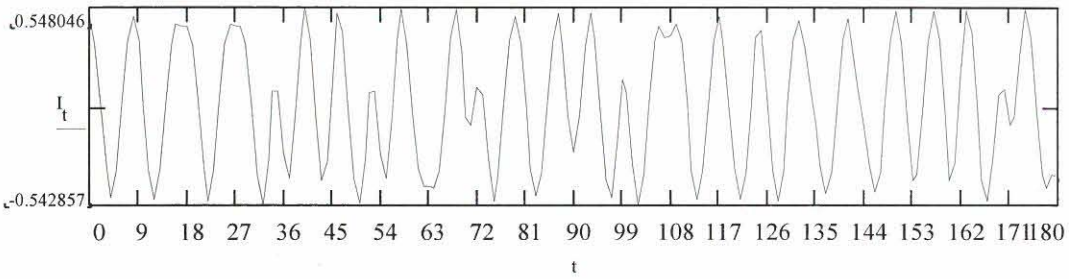


Fig. 4.12 : Sender uitsetsein

4.2.2.6 Kanaal

Die invloed van 'n transmissiekanaal is gesimuleer deur ruis te genereer (sien fig.4.13).

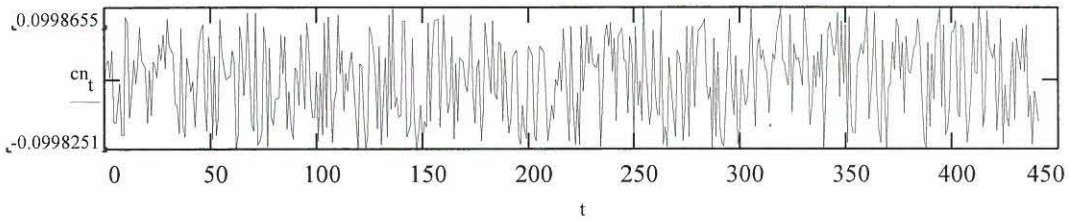


Fig. 4.13 : Ruis

Deur die ruis met die sender uitsetsein (sien fig.4.12) te sommeer word die resultante sein golfvorm in fig.4.14 verkry.

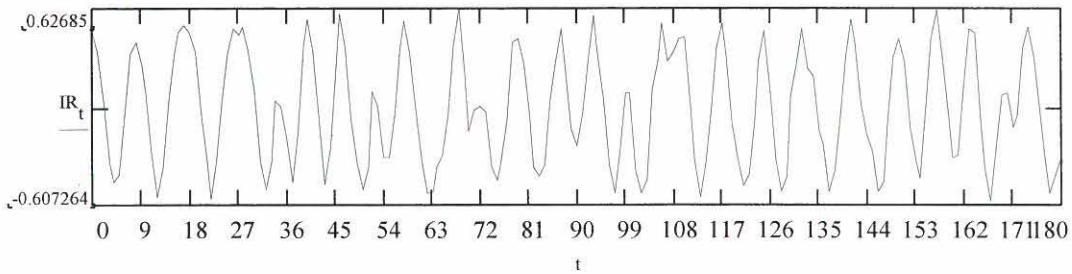


Fig. 4.14 : Ontvangde sein

Fig.4.14 is dus die gesimuleerde golfvorm wat by die ontvanger sal eindig vir deteksie. Die konstellasiediagram van 'n kort datareeks van hierdie ruisbesoedelde sein word in fig.4.15 getoon.

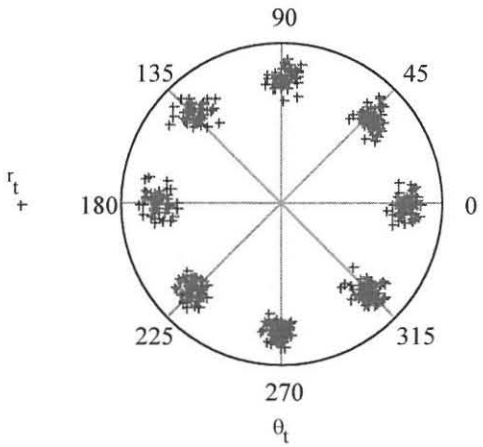


Fig. 4.15 : Konstellasie van ontvangde sein met ruis

Die konstellasie van die modulator uitset in fig.4.9 word in fig.4.16(a) getoon en in fig.4.16(b) die konstellasie van die sender uitsetsein soos in fig.4.12. As hierdie konstellasies vergelyk word met die konstellasie in fig.4.15, word 'n duidelike degradasie waargeneem. Eerstens vanaf fig.4.16(a) na fig.4.16(b) a.g.v. die invloed van die onderdeurlaatfilter. 'n Verdere degradasie het plaasgevind vanaf fig.4.16(b) na fig.4.15 a.g.v. die toevoeging van ruis op die transmissiekanaal.

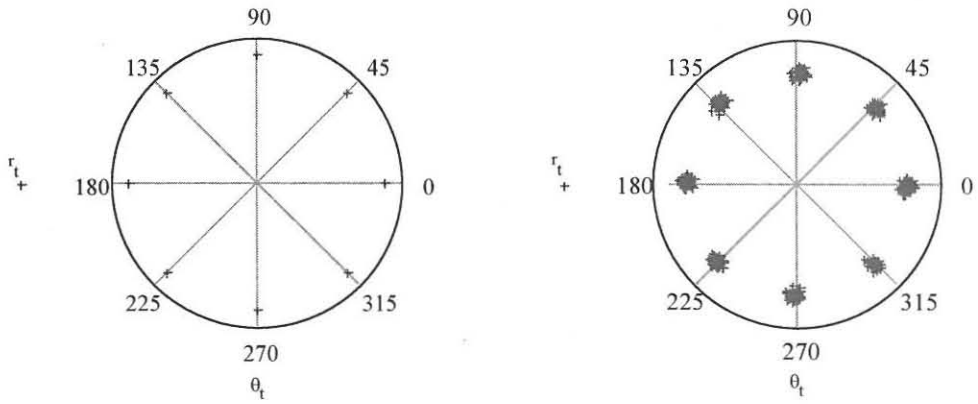


Fig. 4.16 : Konstellasie van (a) modulator uitset en (b) sender uitset

4.3 ONTVANGER

4.3.1 Blokdiagram van die ontvanger

Die ontvanger kan net soos die sender in blokke opgedeel word om die demodulasie en dekodering van die betrokke prosesse (soos in punt 4.1 gespesifiseer) te verduidelik. Fig.4.17 is die blokdiagram van die ontvanger.

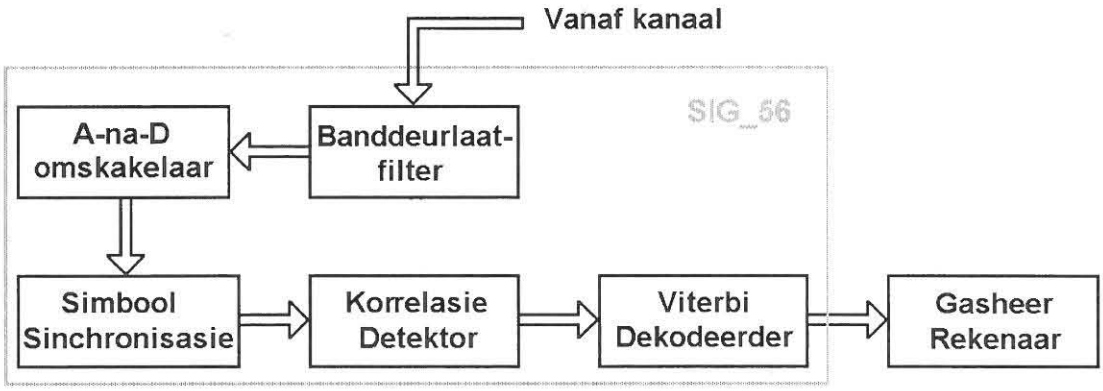


Fig. 4.17 : Blokdiagram van die ontvanger

4.3.1.1 Banddeurlaatfilter

Die ontvangde sein word beperk deur die banddeurlaatfilter tot 'n deurlaatband van 100 - 2040Hz. Net soos by die onderdeurlaatfilter in die sender is die banddeurlaatfilter deel van die TLC32044 omskakelaar wat in hoofstuk 3 bespreek is. Register RA is as volg uit vergelyking (3.2) bereken en opgestel om die afsnyfrekwensie van 2040Hz te kry.

$$tellerA = \frac{f_M}{2 \times f_{SCF}} = \frac{5.184\,MHz}{2 \times 172.8kHz} = 15$$

4.3.1.2 A-na-D omskakelaar

Die gefilterde sein word bemonster teen 'n frekwensie van 4800Hz en gekwantifiseer na 'n 14-bis binêre waarde wat deur die DSP56001 verwerk kan word. Die waarde van RB is as volg met vergelyking (3.3) bereken om 'n monsterfrekwensie van 4800Hz te verkry.

$$tellerB = \frac{f_{SCF}}{f_S} = \frac{172.8kHz}{4800Hz} = 36$$

4.3.1.3 Simboolsinchronisasie

Voor demodulasie en dekodering by die ontvanger kan plaasvind moet die ontvanger eers gesinchroniseerd met die sender wees t.o.v. die simbooltempo. Daarom word daar 'n leerfasesein voor die informasiedraende sein gestuur. Die leerfasesein bestaan uit 'n draaggolf wat teen die simbooltempo, met 180° van fase verander. Na die leerfase word vir 10 baud, 0° faseverandering op die draaggolf toegepas om as teken te dien vir die ontvanger dat die informasiedraende sein direk daarna gaan volg.

Sinchronisering van die ontvangde sein t.o.v. die simbooltempo word verkry deur die ontvangde leerfasesein te manipuleer d.m.v. integrasie en 'n aantal vertragings (fig.4.18). Sodoende word 'n aanduiding verkry van die oomblik van faseverandering tussen twee aangrensende simbole in die leerfasesein.

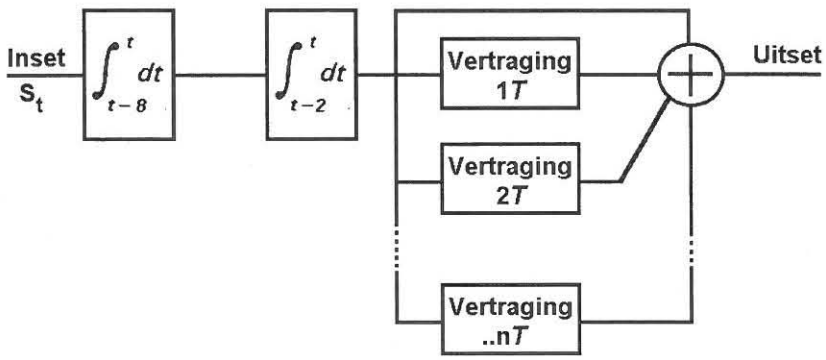


Fig. 4.18 : Simbool-sinchronisasiedetektor

Die sinchronisasie deteksie prosedure word eerste uitgevoer ten einde die ontvanger te sinchroniseer met die 180° veranderende simboolfases, en daarna om die datasein gesinchroniseerd te hou. Die sinchronisasiedetektor word in meer besonderhede onder punt 4.3.2.1 behandel.

4.3.1.4 Korrelasiedetektor

'n Vereenvoudigde 8PSK filterdetektor, soortgelyk aan die 4PSK filterdetektor in fig.2.12, is aangepas vir die deteksie van die differensiële faseveranderings in die ontvangde 8DSPK sein (S_t), (t = 1 monster tydinterval).

Die aanpassing is gedoen t.o.v. die verwysingssein waarvan 4 verskillende fase weergawes afgelei word vir korrelasie. In die DSPK filterdetektor is die ontvangde simbool (S_T) met vier verskillende weergawes van die vorige simbool (S_{T-1}) gekorreleer. Die vier weergawes $(S_{T-1}+0°)$, $(S_{T-1}+45°)$, $(S_{T-1}+90°)$ en $(S_{T-1}+135°)$, maak deel uit van vier moontlike fasegroepe nl. (0°, 180°), (45°, 225°), (90°, 270°) en (135°, 315°), almal t.o.v. S_{T-1} (sien fig.4.19).

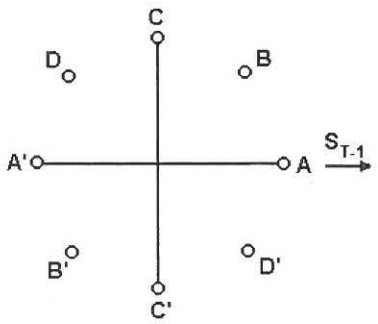


Fig. 4.19 : Fasegroepe vir korrelasie

Die weergawe $(S_{T-1}+0°)$, $(S_{T-1}+45°)$, $(S_{T-1}+90°)$ of $(S_{T-1}+135°)$, met die grootste absolute korrelasie t.o.v. S_T, bepaal watter fasegroep (A,A'), (B,B'), (C,C') of (D,D') die mees waarskynlike ontvangde fase bevat. Die polariteit van die korrelasie onderskei tussen die twee fases in elke groep om die mees waarskynlike

ontvangde fase aan te dui. As die korrelasie positief is, is die mees waarskynlike ontvangde fase A, B, C of D en A', B', C' of D' as die korrelasie waarde negatief is.

Die detektor is verder gewysig a.g.v. die verhouding van die simbooltempo tot die draaggolffrekwensie - wat dit nie toelaat om die ontvangde simbool te korreleer met 'n vertraagde weergawe van die vorige simbool, 1 simbool tydsinterval tevore nie. Die 9 monsters (405° van die draaggolf) in 1 simbool periode veroorsaak dat, indien 'n simbool (fig.4.20(a). monster 18 tot 26) gekorreleer word met die vertraagde weergawe van die vorige ontvangde simbool (fig.4.20(a), monster 9 tot 17) daar 'n verskil van 45° tussen die twee simbole is, alhoewel daar geen faseverskuiwing tussen die twee simbole plaasgevind het nie (sien fig. 4.20(b)).

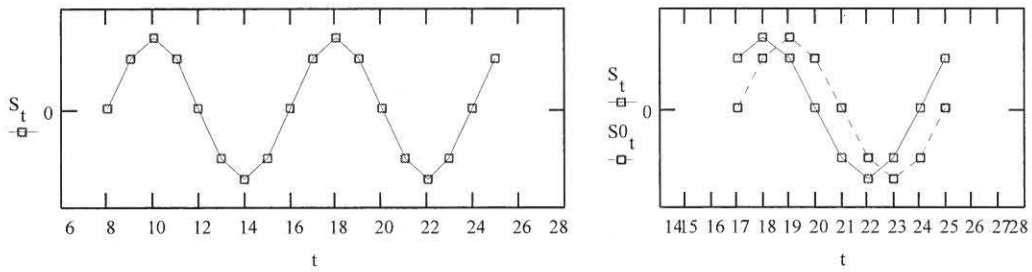


Fig. 4.20 : (a) 2-Simbool draaggolf en (b) vergelyking van twee simbole met mekaar

Daarom word die ontvangde simbool (S_T) gekorreleer met die verskillende fase verskuiwings (0°, 45°, 90°, 135°) van die sein S_t wat vertraag is met 360°, dus 8 monsters (S_{t-8}), en nie met 9 monsters of een simbool tydsinterval nie.

Fig.4.21 is die blokdiagram van die aangepaste filterdetektor soos dit in hierdie projek gebruik is. Die faseverskuiwings (0°, 45°, 90°, 135°) van die 360° vertraagde sein (S_{t-8}) kan as volg t.o.v. S_t geskryf word;

$(S_{t-8} + 0^\circ) = S_{t-8}$ of 'n vertraging van 360°

$(S_{t-8} + 45^\circ) = S_{t-7}$ of 'n vertraging van 315°

$(S_{t-8} + 90^\circ) = S_{t-6}$ of 'n vertraging van 270°

$(S_{t-8} + 135^\circ) = S_{t-5}$ of 'n vertraging van 225°

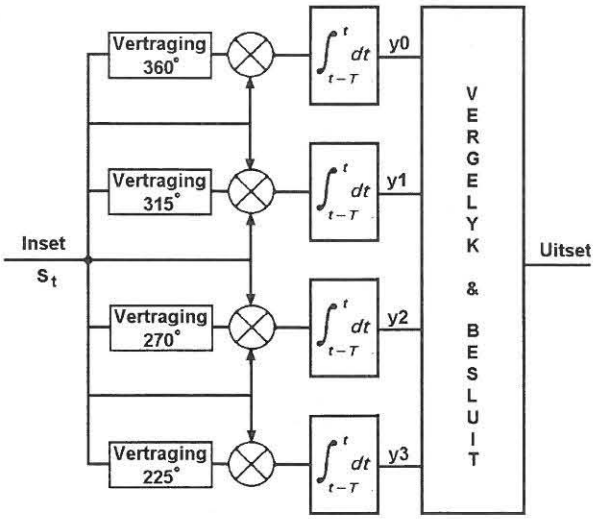


Fig. 4.21 : 8DSPK Korrelasiedetektor (filter detektor)

'n Verdere probleem ontstaan wanneer die vertraagde simbool geskuif moet word om die 4 fases vir korrelasie te lewer. Omdat 'n simbool uit slegs 9 monsters bestaan, en daar net 8 monsters vir korrelasie oor is na die 360° vertraging, is daar in die 135° verskuiwing geval, (die grootste verskuiwing) slegs 5 monsters van die 135° verskuifde sein bruikbaar vir korrelasie met die ontvangde simbool (sien fig.4.22). Deur slegs die 5 monsters te gebruik vir korrelasie i.p.v. 8 monsters word die ruisimmuniteit van die detektor aansienlik verswak.

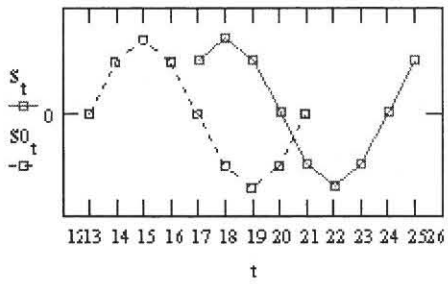


Fig. 4.22 : Korrelasie van 135° verskuiwing met die ontvangde simbool.

Om bogenoemde verswakking teen te werk is 4 addisionele monsters vir die 135° weergawe, 3 vir die 90° weergawe, 2 vir die 45° weergawe en 1 vir die 0° weergawe in die ontvanger gegenereer sodat die ontvangde simbool met 9 monsters van die 4 vertraagde weergawes vergelyk kan word. In die simulasie van die detektor is die addisionele monsters direk afgelei vanaf die ander monsterwaardes in die vertraagde simbole aangesien $S_t = -S_{(t-4)}$ vir 'n sinusgolf. Vir $S0_t$ in fig.4.22 kan 'n

addisionele monster, bv. monster 14, as volg verkry word $S0_{14} = -S0_{14-4} = -S0_{10}$ (sien fig.4.23).

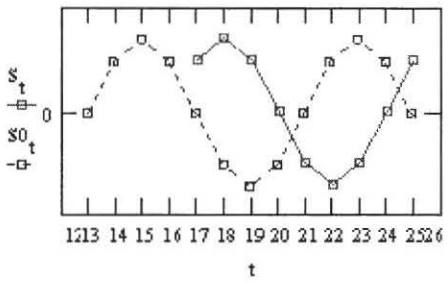


Fig. 4.23 : Generering van addisionele monsters

Hierdie oplossing was slegs suksesvol in die simulasie met 'n teoretiese draaggolf maar kon nie in die werklike detektor geïmplementeer word nie a.g.v. die verwringing van die sein deur die filters en eksterne ruis.

In die DSP detektor is die addisionele monsters verkry deur die ontvangde simbool telkens op die interne sinustabel van die DSP56001 te pas en sodoende 'n suiwer sinusgolf weergawe van die ontvangde simbool te verkry - waarvandaan addisionele monsters afgelees kon word.

4.3.1.5 Viterbi dekodeerder

Viterbi dekodering word gedoen t.o.v. 'n 8-staat trelliskode (sien 2/3 enkodeerder in fig.4.2).

Die dekodering vind plaas oor 'n simbool venster van 15 baud (5 x m) wat beteken dat elke ontvangde simbool uitset eers 15 baud na ontvangs gelewer word. Die Viterbi dekoderingsproses, met sagtebesluitneming, gebruik die Euklidiese afstand in die trellis om die kortste pad en mees waarskynlike ontvangde simbool, te bepaal. In die projek is daar egter t.o.v. elke ontvangde simbool van die verskil in korrelasie van die onderskeie fases tot die maksimum korrelasie waarde gebruik gemaak vir die bepaling van die kortste pad. Sodoende is ekstra prosessering uitgeskakel.

Ten einde hierdie aanpassing te evalueer is die relatiewe prestasie van die twee tegnieke (Euklidiese afstand en verskil-in-korrelasie waarde) teen verskillende fasehoeke gesimuleer. Fig.4.24 toon 'n fase afwyking van 0 - 180° vanaf 'n bepaalde fasehoek, teenoor die ooreenkomstige Euklidiese afstand en korrelasie. Die waardes is genormaliseer ten einde die vergelyking van die kurwes te vergemaklik.

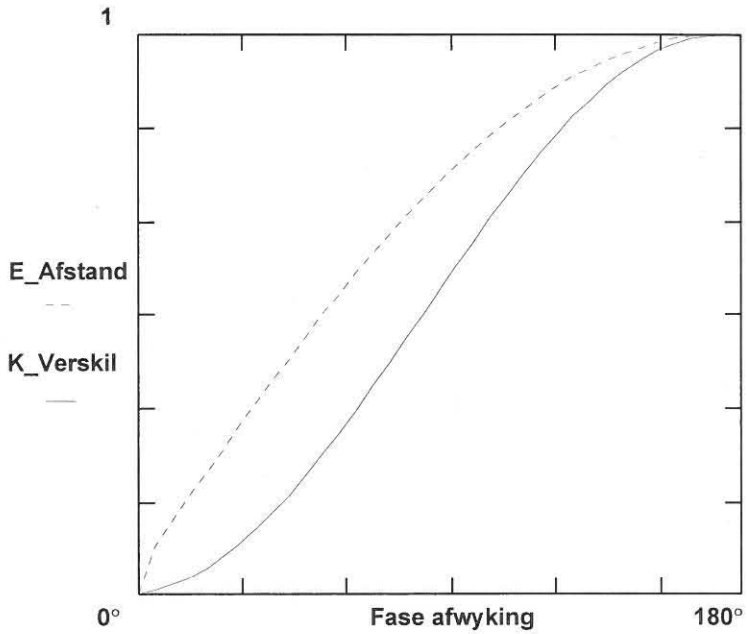


Fig. 4.24 : Verandering in genormaliseerde Euklidiese afstand en korrelasie t.o.v. fase afwyking

Uit die grafiek kan waargeneem word dat die gebruik van die korrelasie waardes i.p.v. die Euklidiese afstand 'n effense degradering in prestasievermoë sal lewer teen veral klein hoeke. Dit is egter ook duidelik dat die berekende korrelasie waardes wel met welslae gebruik kan word om 'n unieke fasehoek te identifiseer.

4.3.1.6 Gasheerrekenaar

Soos aangedui in fig.3.9 is een gasheerrekenaar vir beide die sender en ontvanger gebruik.

4.3.2 Simulasie van die ontvanger

Dele van die ontvanger is ook m.b.v. Mathcad 5.0 gesimuleer. Die simulasie is t.o.v. die volgende onderafdelings van die ontvanger in fig.4.17 gedoen.

1. Sinchronisasiedetektor - om 'n metode te verkry waardeur die ontvanger die oomblik van verandering in fase tussen opeenvolgende simbole kan bepaal.
2. 8DPSK Korrelasiedetektor om die uitvoerbaarheid en werking van die gewysigde filterdetektor te ondersoek.

As gevolg van die kompleksiteit van die Viterbi dekoderingsproses was dit nie moontlik om die dekodering met hierdie pakket te simuleer nie.

Fig. 4.25 toon 'n blokdiagram van die koppeling tussen die sinchronisasiedetektor (fig.4.18) en filterdetektor (fig.4.21) in die ontvanger.

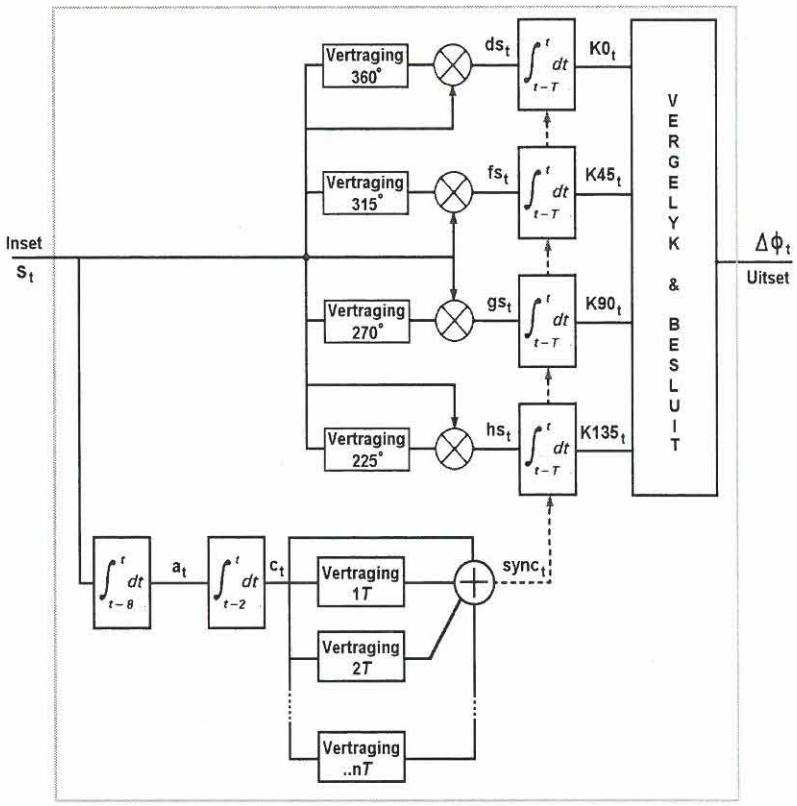


Fig. 4.25 : Blokdiagram van sinchronisasie- en filterdetektor.

4.3.2.1 Sinchronisasiedetektor

Die simulasie van die sinchronisasiedetektor sal vervolgens t.o.v. die blokdiagram in fig.4.25 bespreek word met die volledige simulasie aangeheg as bylae A.2.

4.3.2.1.1 Sein vir deteksie

Ter verduideliking van die sinchronisasiedetektor se basiese werking sal dit eers t.o.v. 'n leerfasesein, soos in fig.4.26 aangedui, bespreek word. Die volledige simulasie in bylae A.2 toon die werking t.o.v. 'n pseudo-ewekansige datasein, soos deur die gesimuleerde sender gelewer.

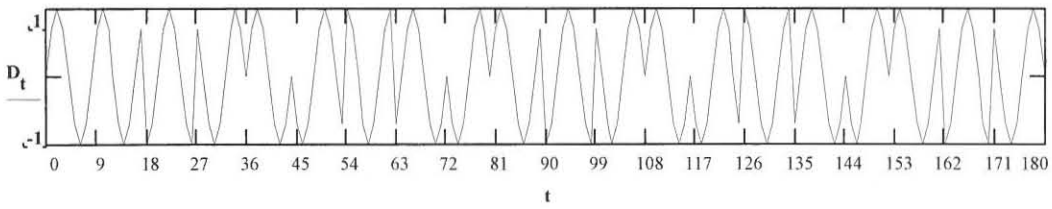


Fig. 4.26 : Leerfasesein

4.3.2.1.2 Gemiddeld oor 8 monsters

Die ontvangde leerfasesein (s_t) word eerstens geïntegreer oor 'n periode van 360° (8 monsters) van die draaggolf. Aangesien 'n simbool uit 9 monsters bestaan, lewer die integrasie 'n sein a_t waarvan die waardes tydens die 8ste en 9de monsters van elke simbool gelyk is aan zero (sien fig. 4.27). 'n Aantal toevallige zero punte kom ook in a_t, voor.

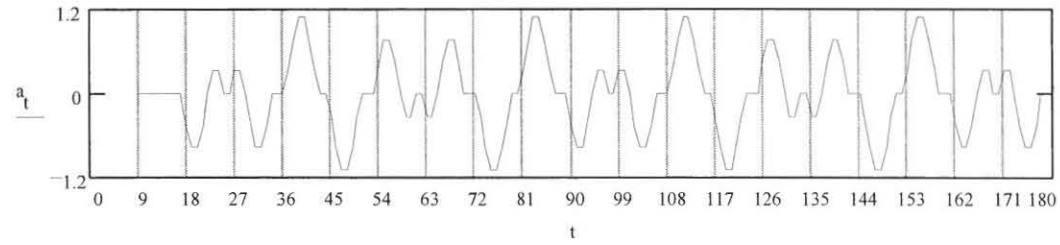


Fig. 4.27 : Uitset van 8 monster integreerder

Deur die absolute waarde van sein a_t te bepaal word 'n sein (b_t) verkry (sien fig.4.28) waarvan die fundamentele frekwensie gelyk is aan die leerfasesein simbooltempo. Fig.4.29 toon die frekwensie elemente van sein b_t waar die merker op punt 455 die ooreenkomstige frekwensie aandui, van

$$\frac{455}{4096\_monsters} \times 14400 monsters / sekonde = 1600 baud$$

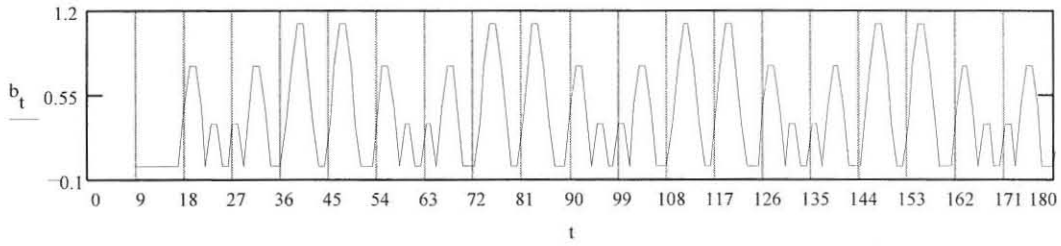


Fig. 4.28 : Absolute waarde van 8-monster integreerder uitset

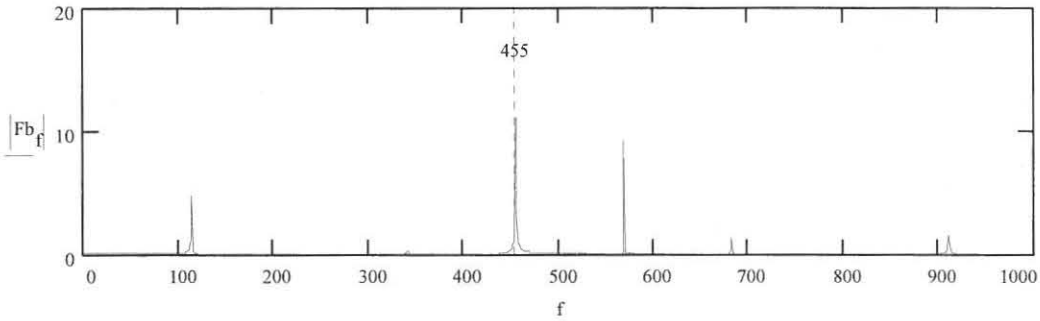


Fig. 4.29 : Frekwensie elemente van sein b_t

4.3.2.1.3 Gemiddeld oor 2 monsters

Alhoewel sein b_t 'n duidelike aanduiding van die simbooltempo lewer is dit nie moontlik om direk daaruit 'n aanduiding van die presiese oomblik van fase verandering in die leerfasesein waar te neem nie. Om onderskeid te tref tussen die twee monster zero-tydperk wat in sein b_t voorkom tydens die einde van elke simbool

(monster 8 & 9), en die ander toevallige zero punte in sein b_t, word sein b_t geïntegreer oor 2 monsters. Dit lewer 'n sein c_t wat in fig. 4.30 getoon word.

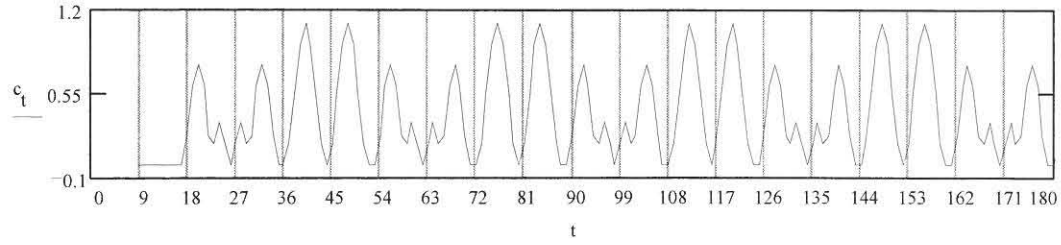


Fig. 4.30 : Uitset van 2-monster integreerder

Uit fig. 4.30 kan waargeneem word dat sein c_t gelyk is aan zero tydens monster 9 van elke tweede simbool, en t.o.v. die ander simbole tydens monster 9 en monster (9 +/- 1).

4.3.2.1.4 Gemiddeld oor n-baud

Ten einde die geskiktheid van sein c_t verder te verbeter word dit gesommeer met n-simbool vertragings van homself. Met n=2 word 'n sein gelewer soos fig. 4.31 toon[1].

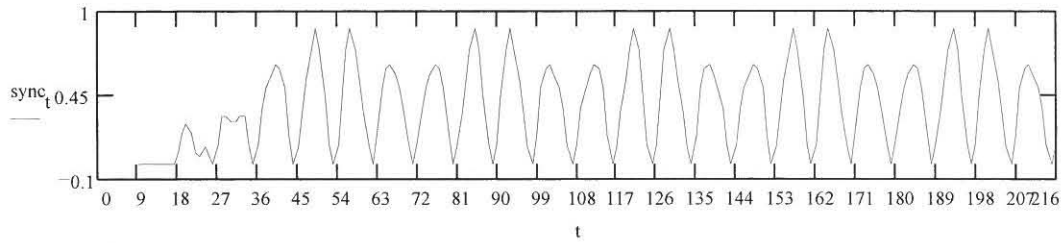


Fig. 4.31 : Sinchronisasiesein

Die sein in fig. 4.31 word as sinchronisasiesein gebruik waar die zero of minimum waardes van die sein dui op die laaste monster, van 'n simbool. Fig.4.32 toon die frekwensie elemente van die sinchronisasiesein en die onderdrukking van

[1] Soos later sal blyk is die waarde van n uiteindelik na 7 verander.

frekwensies naby die simbooltempo kan duidelik waargeneem word as dit met fig.4.29 vergelyk word.

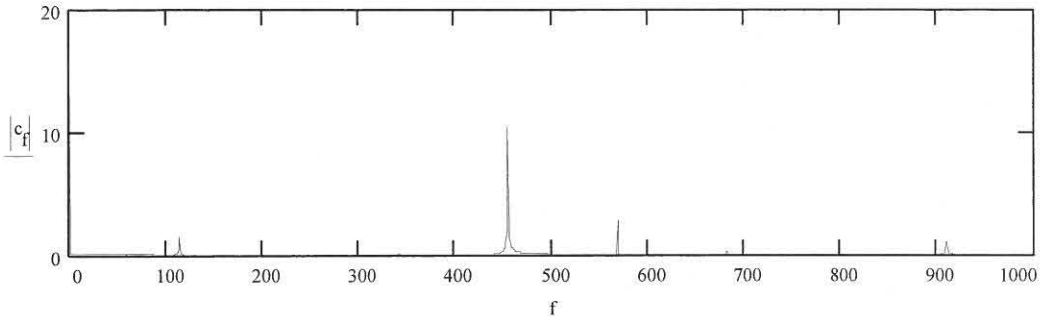


Fig. 4.32 : Frekwensiespektrum van die sinchronisasiesein

Bogenoemde sinchronisasiesein (soos in fig.4.31) is verkry vanaf 'n gesimuleerde leerfasesein. Figure 4.33 en 4.34 toon die frekwensiespektrum van sinchronisasieseine gelewer deur die detektor vir :

1. Gesimuleerde, leerfase- en pseudo-ewekansige dataseine (sender uitset).
2. Gemonsterde DSP sender leerfase- en pseudo-ewekansige dataseine soos dit by die DSP ontvanger ontvang word vir deteksie.

Vanaf fig.4.33 tot fig.4.34 word 'n duidelike degradasie in die amplitude van die simbooltempo frekwensie element waargeneem.

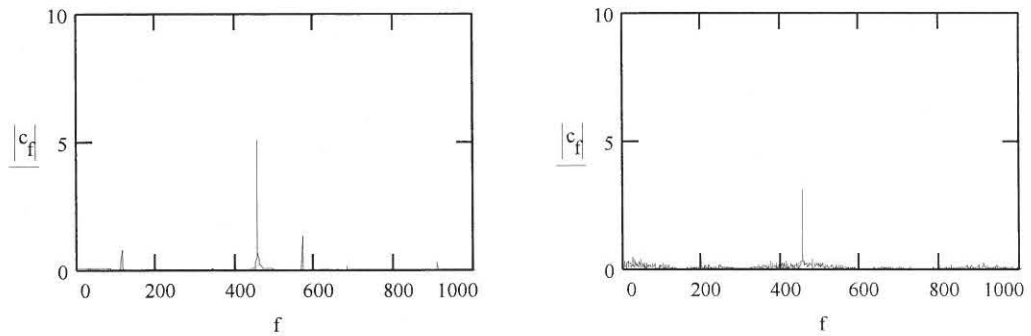


Fig. 4.33 : Frekwensiespektrum verkry vanaf 'n gesimuleerde gefilterde (a) leerfasesein en (b) pseudo-ewekansige datasein.

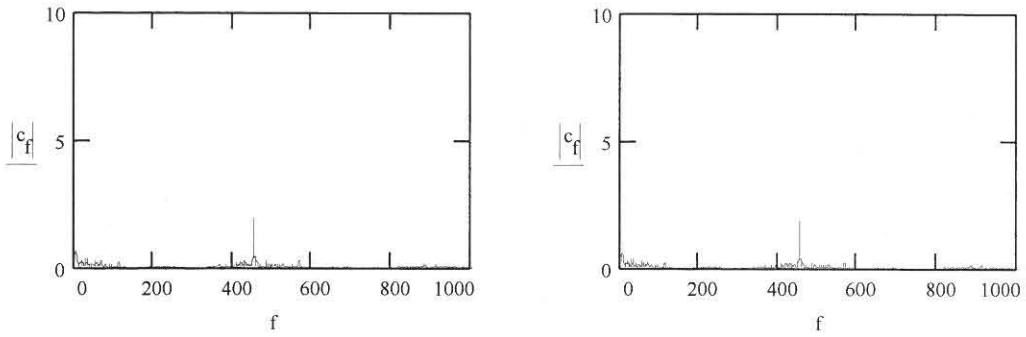


Fig. 4.34 : Frekwensiespektrum verkry vanaf DSP sender (a) leerfasesein en (b) pseudo-ewekansige datasein.

Die sinchronisasiesein verkry deur verwerking van die ontvangde DSP sender seine lewer 'n definitiewe simbooltempo frekwensie komponent (sien fig.4.34(a) en (b)), alhoewel dit aansienlik kleiner in amplitude is as die ooreenstemmende weergawes van die gesimuleerde seine. Fig.4.35 toon die sinchronisasiesein soos bepaal vir die DSP sender pseudo-ewekansige datasein. Fig.4.34(b) toon die frekwensiespektrum van hierdie sein.

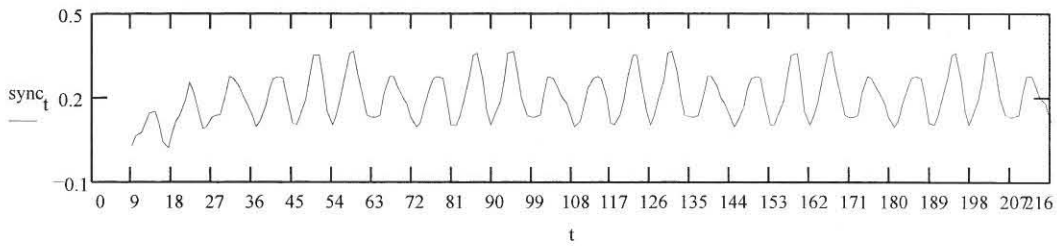


Fig. 4.35 : Sinchronisasiesein vir ontvangde DSP pseudo-ewekansige datasein

Uit fig.4.35 word waargeneem dat die sinchronisasiesein se golfvorm baie onstabiel voorkom. Deur die gemiddelde sinchronisasiesein oor meer simbole (n) te bepaal soos hierbo bespreek, kan 'n meer stabiele sinchronisasiesein verkry word. Figure 4.36 en 4.37 toon die sinchronisasieseine met n = 5 en 7 onderskeidelik vir die ontvangde DSP sender pseudo-ewekansige datasein.

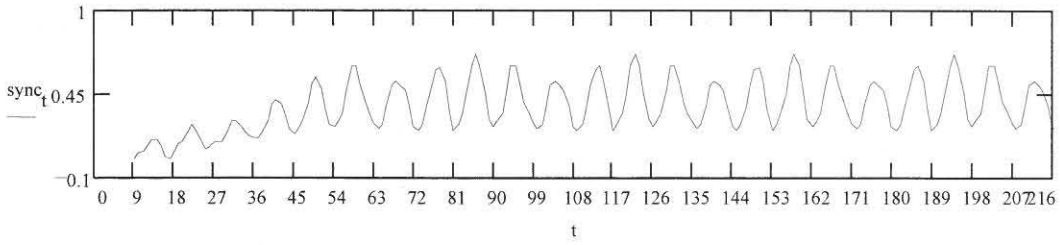


Fig. 4.36 : Sinchronisasiesein vir DSP pseudo-ewekansige data met n = 5

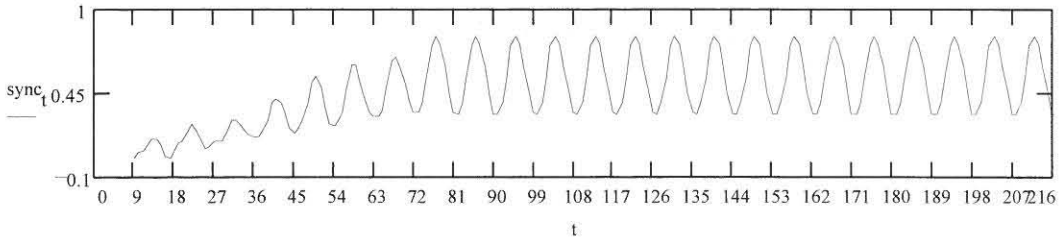


Fig. 4.37 : Sinchronisasiesein vir DSP pseudo-ewekansige data met n = 7

Dit kan dus waargeneem word uit bostaande figure dat die vermeerdering van "n" 'n aansienlike verbetering in die stabiliteit van die sinchronisasiesein tot gevolg het.

4.3.2.2 8DPSK Korrelasiedetektor

Soos in die blokdiagram (sien fig. 4.25) getoon, ontvang die demodulator 'n sinchronisasiesein wat die einde van 'n simbool aandui. In hieropvolgende bespreking van die demodulator simulasie word die ontvangde sein (s_t) as gesinchroniseerd beskou met die punte 9,18.....t wat dui op die begin, of eerste monster, van elke ontvangde simbool. Fig.4.38 toon 'n gesimuleerde pseudo-ewekansige datasein vir demodulasie. Die volledige simulasie van die demodulator word as bylaag A.3 aangeheg met die gesimuleerde sender uitsetsein as inset.

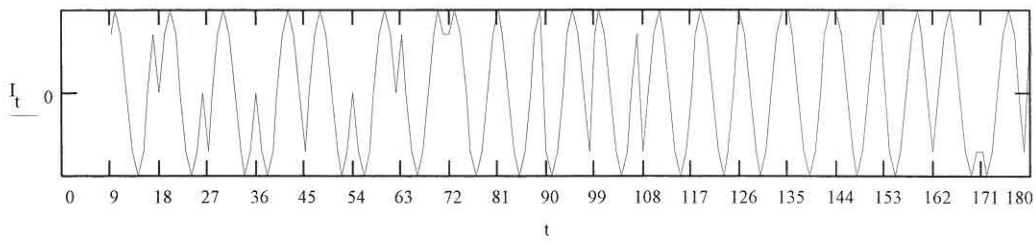


Fig. 4.38 : Pseudo-ewekansige datasein vir demodulasie

4.3.2.2.1 Vertraging en fase verskuiwing.

Soos in punt 4.3.1.4 verduidelik, word die ontvangde sein vertraag met 225°, 270°, 315° en 360° onderskeidelik om die 4 weergawes van die vorige ontvangde simbool te lewer. Fig.4.39 toon die ontvangde sein s_t tesame met 4 vertraagde seine waar $hs_t = s_t - 225^\circ$, $gs_t = s_t - 270^\circ$, $fs_t = s_t - 315^\circ$ en $ds_t = s_t - 360^\circ$. Die mees onlangse ontvangde simbool van sein s_t, is heel regs in fig.4.39.

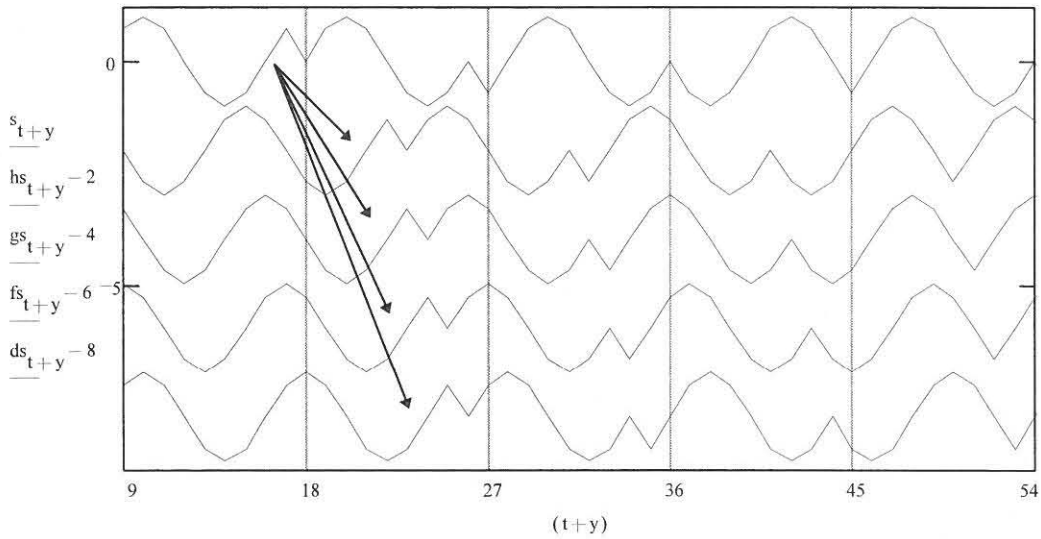


Fig. 4.39 : Vier vertragings van ontvangde sein

4.3.2.2.2 Genereer addisionele monsters.

Addisionele monsters word vir die vertraagde seine (hs, gs, fs en ds) gegenereer vir korrelasie soos in punt 4.3.1.4 bespreek. Fig. 4.40 toon die ontvangde sein s_t, asook die vertraagde weergawes met hul addisionele monsters.

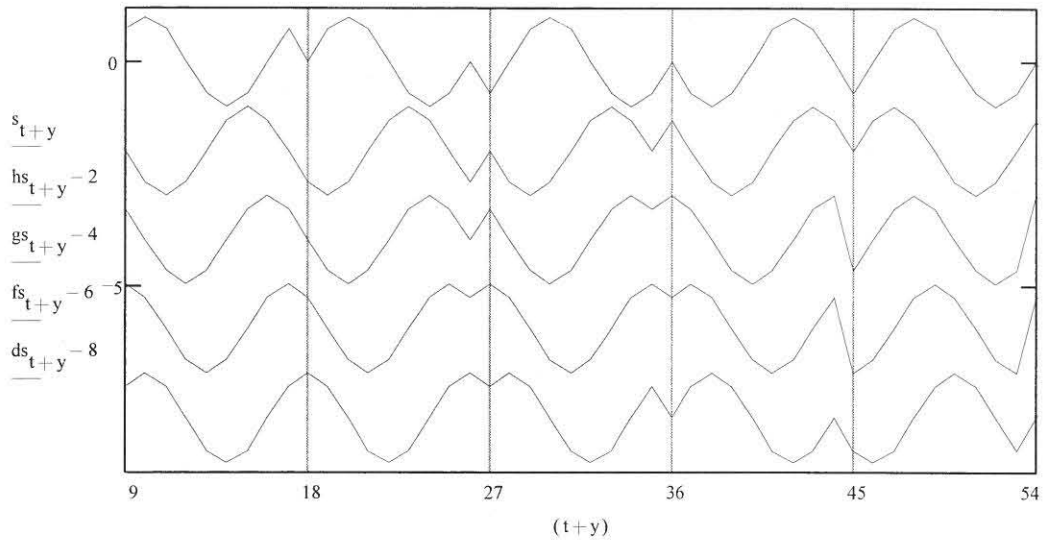


Fig. 4.40 : Vier vertragings met addisionele monsters

4.3.2.2.3 Korrelasie oor 9 monsters

Die ontvangde simbool s_T, word met die 4 fase weergawes gekorreleer om 4 uitset waardes vir elke simbool te lewer. Fig. 4.41 toon elke simbool se 4 korrelasie waardes. Die absolute grootste korrelasie waarde dui op die waarskynlike ontvangde fasegroep, en die polariteit van korrelasie op die fase in die groep soos reeds in punt 4.3.1.4 genoem. Dus as die simbool $s_{18tot26}$ in fig.4.40 gekorreleer word met die 4 vertragings van die vorige simbool s_{9tot17} word die korrelasie uitset waardes verkry soos tydens punt 18 in fig. 4.41.

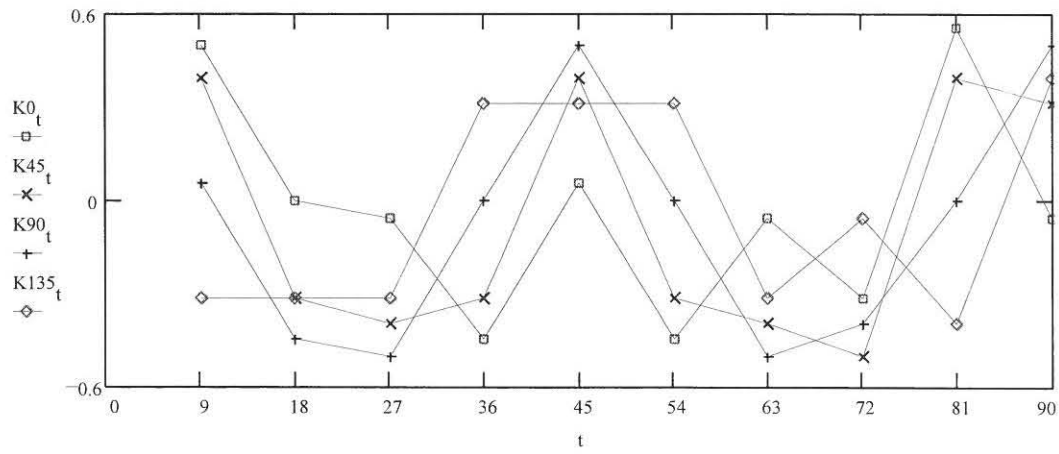


Fig. 4.41 : Korrelasie waardes t.o.v. vorige simbool

Tydens punt 18 is die korrelasie van die 90° verskuiwing (K90) die absolute grootste, maar is negatief. Dit dui daarop dat die ontvangde simbool $s_{18tot26}$ 'n fase verandering van 270° (90° + 180°) ondergaan het t.o.v. simbool s_{9tot17}. Dit verteenwoordig 'n datasein van 6 (sien tabel 4.2). Hierdie proses word ten opsigte van elke simbooltyd herhaal - soos deur die sinchronisasiedetektor - en die faseverandering t.o.v. elke simbool bepaal. Fig. 4.42 toon die differensiële faseveranderings vir 'n kort reeks in die ontvangde sein soos deur die detektor bepaal.

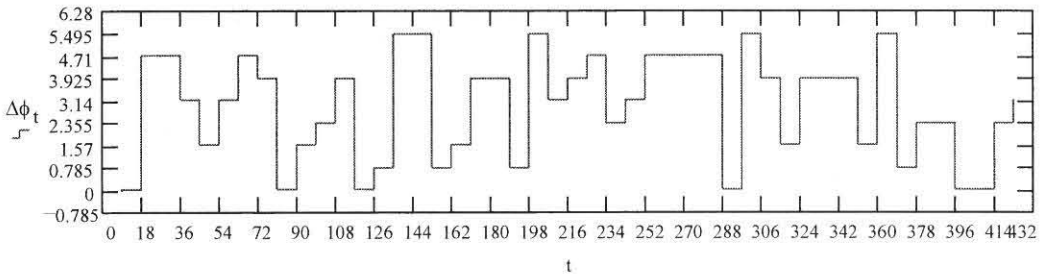


Fig. 4.42 : Waarskynlike ontvangde fases

As fig.4.42 met die oorspronklike differensiële fases vir modulasie in die sender simulasie (fig.4.8) vergelyk word, is dit duidelik dat die ontvangde sein korrek gedemoduleer is.

4.3.2.2.4 Verskil in korrelasie waardes.

Fig.4.43 toon die verskil in korrelasie waardes van al 8 moontlike ontvangde fases relatief tot die geïdentifiseerde mees waarskynlike fase. Hierdie waardes word tydens die Viterbi dekoderingsproses gebruik (sien punt 4.3.1.5).

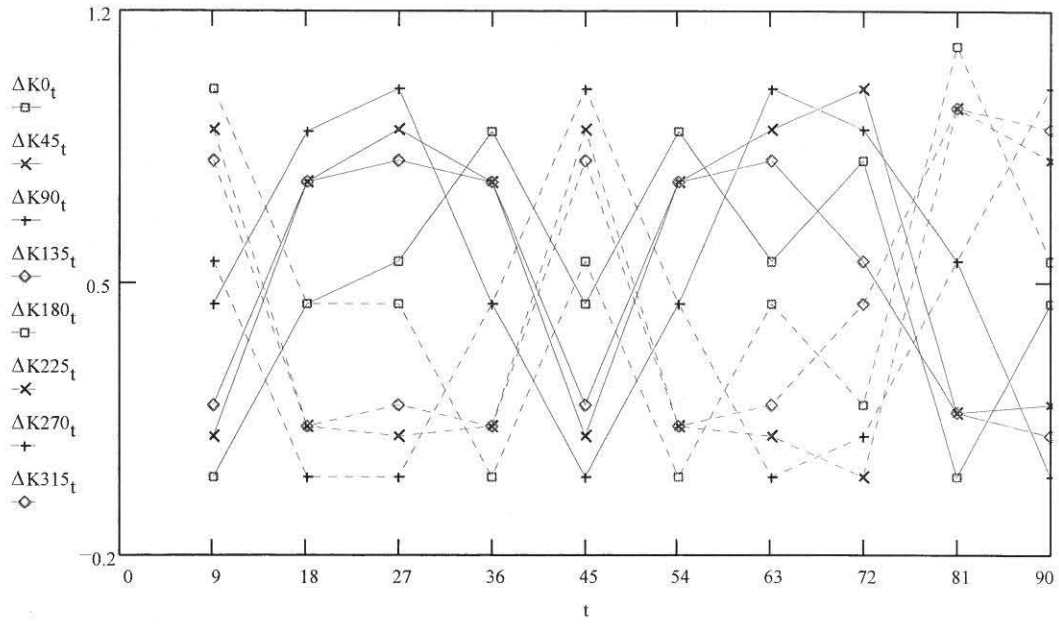


Fig. 4.43 : Verskil in korrelasie waardes

4.4 OPSOMMING

Die simulasie van die sender het baie tot die algemene insig van die werking van die stelsel bygedra, sowel as om 'n aanduiding te verskaf van die haalbaarheid daarvan om so 'n sein oor 'n gewone telefoonlyn te versend.

Die ontvanger simulasie resultate het gedui op die moontlikheid van die fisiese implementering van die sinchronisasie en demodulasieprosesse. Daar is besluit op n=7 vir gebruik in die DSP sinchronisasiedetektor aangesien dit die minimum waarde - minimum prosessering benodig - is waarmee 'n relatiewe stabiele sinchronisasiesein verkry word vanaf die ontvangde DSP sender datasein.

Deur toetsing is verder gevind dat die sinchronisasiesein 'n faseverskuiwing van 90° of twee monsters ondergaan wanneer daar gesinchroniseer word t.o.v. 'n gemonsterde DSP datasein. Hierdie faseverskuiwing is toe te skryf aan die monstering en filtering van die gestuurde analoog datasein.

5. STELSEL SAGTEWARE

5.1 INLEIDING

Deeglike oorweging van die simulasies soos in hoofstuk 4 uiteengesit het die beplanning van die struktuur van die benodigde sagteware moontlik gemaak. Die stelsel bestaan hoofsaaklik uit drie stelle sagteware wat die onderskeie blokke (buiten die analoog ruisgenerator) in fig.3.9 verteenwoordig.

1. Gasheer sagteware wat primêr uit 'n Pascal program (tx_rx.pas), uitgevoer op die gemeenskaplike gasheerrekenaar vir die beheer van datavloei na en vanaf die DSP56001 sender en ontvanger, bestaan. Die Pascal program maak verder gebruik van prosedures, gelewer deur Peralex vir die SIG56 kaart.
2. Sender DSP sagteware wat gebruik maak van die herstel.asm, data_in.asm en send.asm saamsteltaalprogramme.
3. Ontvanger DSP sagteware wat gebruik maak van die herstel.asm en ontv.asm saamsteltaalprogramme.

5.2 GASHEERREKENAAR SAGTEWARE

5.2.1 Peralex prosedures

Die volgende Pascal prosedures deur Peralex gelewer, word in die gasheer Pascal program gebruik.

- DSP_Reset(baseaddress) - Herstel die DSP56001 van enige vorige gelaaide program.
- DSP_Load(baseaddress) - Laai die programkode (*.lod) in die DSP56001.
- DSP_Go(baseaddress) - Inisieer 'n program wat in die DSP56001 gelaai is.
- DSP_WriteFlag(baseaddress,timeout) - Toets die DSP56001 se statusregister totdat die SIG56 kaart gereed is vir die ontvangs van data.

- DSP_ReadFlag(baseaddress,timeout) - Toets die DSP56001 se status register totdat data gelees kan word vanaf die SIG56 kaart.
- DSP_WriteInteger(baseaddress,timeout) - Skryf 'n 16-bis heelgetal waarde na die SIG56 kaart.
- DSP_ReadLongInt(baseaddress,timeout) - Lees 'n 24-bis heelgetal vanaf die SIG56 kaart.

5.2.2 Gasheerrekenaar Pascal program (tx_rx.pas)

Die Pascal program word eerste bespreek omdat dit 'n algehele beeld skep van die volgorde van stappe in die transmissie en ontvangs van data tussen die sender en ontvanger. Die program (tx_rx.pas) word in bylae B.1 getoon en word aan die hand van die vloeikaart in fig.5.1 bespreek.

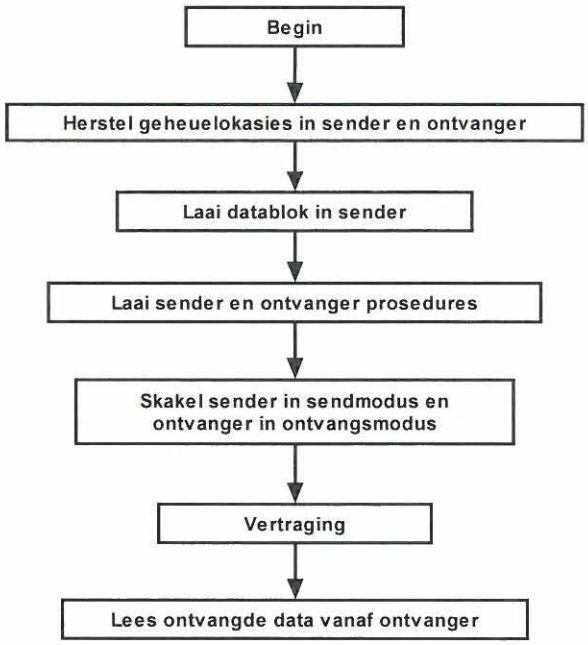


Fig. 5.1 : Vloeikaart van Pascal program

5.2.2.1 Herstel sender en ontvanger geheue

Die herstel.lod program word in beide die sender en ontvanger gelaai met die Peralex "DSP_Load" prosedure. Dit dra die herstel.lod program vanaf die gasheer na die

DSP56001 sender en ontvanger oor. Die herstel.lod saamsteltaalprogram word in die sender en ontvanger begin deur die Peralex "DSP_Go" prosedure .

5.2.2.2 Laai data in sender

Pseudo-ewekansige data word in die DSP sender gelaai vanaf 'n lêer (rdm.dat), in die vorm van 16-bis datawoorde. 'n Blok van 512 datawoorde (8192 bisse) word na die DSP56001 oorgedra deur van die Peralex "DSP_WriteFlag" en "DSP_WriteInteger" prosedures gebruik te maak. Die 512 datawoorde word tydens transmissie herhaaldelik gestuur om 'n pseudo-ewekansige datastroom te lewer wat elke,

$$\frac{8192(bisse)}{1067(bisse/sekonde)} \approx 7.7(sekondes)$$

herhaal word.

5.2.2.3 Laai sender en ontvanger programme.

Die DSP send.lod[2] program word in die sender SIG56 kaart geheue ingelees, gereed vir uitvoering. Op dieselfde wyse word ontv.lod in die ontvanger SIG56 kaart geheue ingelees.

5.2.2.4 Begin sender en ontvanger.

Die send.lod program word eerste in die sender DSP geïnisieer en direk daarna, die ontv.lod program in die ontvanger.

---

[2] Die *.lod programme na verwys is telkens die saamgestelde weergawe van die *.asm programme met dieselfde name.

5.2.2.5 Vertraging.

Die gasheer program word vertraag vir 'n bepaalde transmissietyd, afhangend van die hoeveelheid bisse wat oorgedra moet word. A.g.v. beperkte geheue in die ontvanger word die datablok (512 × 16-bis woorde) vyf keer versend en ontvang waarna dit uitgelees word na die gasheer vir evaluasie. Die totale vertraging benodig vir die oordrag van vyf datablokke is gelyk aan,

$$7.7 \times 5 \approx 38.5 sekondes \ .$$

'n Totale vertraging van 40 sekondes word gebruik.

5.2.2.6 Lees data uit vanaf ontvanger.

Ontvangde datawoorde in die DSP56001 ontvanger word uitgelees na die gasheer deur van die "DSP_ReadFlag" en "DSP_ReadLongInt" prosedures gebruik te maak. Die datawoord word in die 24-bis formaat uitgelees om die heelgetal datawoord met 'n teken uit te lees. Die datawoorde word in 'n lêer (ontv.dat) gestoor vir latere evaluering.

5.3 SENDER SAGTEWARE

Die sender bestaan hoofsaaklik uit die send.asm program vir datatransmissie maar maak ook gebruik van die herstel.asm en data_in.asm programme soos benodig om die sender mee op te stel voordat datatransmissie plaasvind.

SENDER	
Gasheer	DSP56001
Pascal program (TX_RX.pas)	herstel.asm
	data_in.asm
	send.asm

5.3.1 Herstel sender program (herstel.asm)

Die herstel.lod program word voor datatransmissie in die sender gelaai, en uitgevoer. Dit word gedoen ten einde al die X- en Y eksterne geheuelokasies wat in die sender DSP program gebruik word na zero te herstel sodat die programme vanaf 'n totale herstelde staat begin funksioneer en nie gebruik maak van waardes wat moontlik deur 'n vorige program gelaat is nie. Die program se saamsteltaal weergawe word in bylae B.2 getoon.

Eerstens word die DSP56001 gekonfigureer om in modus 2 (normale uitgebreide modus) te funksioneer. Dit implementeer 'n geheue uitleg soos in punt 3.3.1 uiteengesit. Daarna word zero's in die eksterne X- en Y geheuelokasies $200 - $7FF gelaai.

5.3.2 Inlees van data vir versending (data_in.asm)

Hierdie program lees 'n blok van 512, 16-bis datawoorde in vanaf die gasheerrekenaar. Die saamsteltaal weergawe word in bylae B.3 getoon.

Soos by die herstel.asm program word die DSP weer gekonfigureer om in modus-2 te funksioneer. Daarna word die data soos gelewer deur die gasheer, deur die DSP56001/gasheerrekenaarkoppelvlak (DSP56001 poort B), ingelees. Die datawoorde word in geheuelokasies X : $400 - $5FF gestoor.

5.3.3 Versending van data (send.asm).

Die send.asm saamsteltaalprogram verrig die prosesse soos uiteengesit in die sender blokdiagram (sien fig.4.1) nl. kodeer en moduleer die gestoorde datawoorde en lees dit uit na die D-na-A omskakelaar - waar die omskakeling na die analoogsein plaasvind. Bylae B.4 bevat die sender saamsteltaalprogram en dit kan aan die hand van die vloeidiagram in fig.5.2 bespreek word.

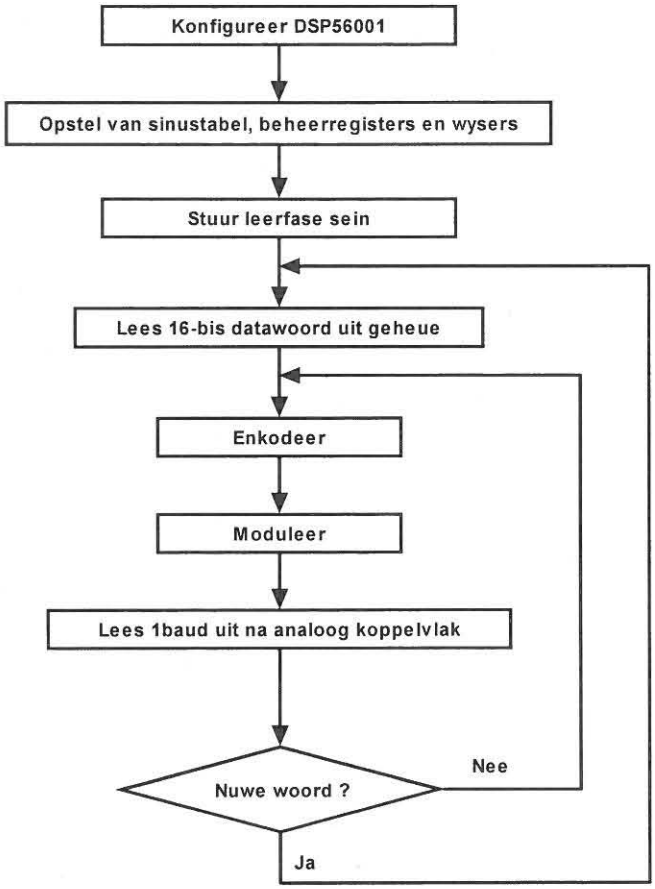


Fig. 5.2 : Vloeidiagram van die sender saamsteltaalprogram

5.3.3.1 DSP56001 konfigurasie

Net soos by die vorige twee saamsteltaalprogramme word die DSP56001 gekonfigureer om in modus-2 te funksioneer.

5.3.3.2 Opstel van sinustabel, beheerregisters en wysers (prosedure - OPSTEL)

'n 8-Punt sinustabel word gelaai met die amplitude waardes van 'n sinusgolf op 0°, 45°, 90°, 135°, 180°, 225°, 270° en 315°. Poort C word opgestel vir algemene inset/uitset doeleindes met die onderbrekings ontmagtig.

Die analoog koppelvlakkring se tydkonfigurasiewaardes van toepassing op die D-na-A omskakeling en uitset onderdeurlaatfilter word opgestel met TB = 36 en TA = 15. Dit

verteenwoordig die aangepaste waardes soos in punte 4.2.1.5 en 4.2.1.6 bereken. Die koppelvlak beheerregister word opgestel met die verstek waardes, met die (sinx)/x korreksiefilter aangeskakel.

Die volgende adresregisters word as sirkuleringsbuffers gebruik :

r1 - Wyser vir die blok datawoorde.
m1 - Spesifiseer 'n 512 sirkuleringsbuffer vir die blok datawoorde.
n1 - Inkrementeringswaarde van r1 in die buffer.

r2 - Wyser vir die sinus opkyktabel.
m2 - spesifiseer 'n 8 stap sirkuleringsbuffer vir die sinustabel.
n2 - Inkrementeringswaarde van sinustabel.

5.3.3.3 Stuur leerfasesein (prosedure - IDENT)

Die verskillende fases van die leerfasesein soos in punt 4.3.1.3 uiteengesit word op die draaggolf gemoduleer. Die modulasieproses word onder punt 5.3.3.6 bespreek.

5.3.3.4 Lees 'n 16-bis datawoord (prosedure - LEES_INT)

'n 16-Bis datawoord word elke agtste baud uit die datawoord blok gelees omdat twee bisse per baud uit die 16-bis datawoord geskuif word vir enkodering. Die datawoordwyser is 'n sirkuleringsbuffer van 512 stappe, wat die datablok herhaaldelik laat uitlees vir versending soos in punt 5.2.2.2 verduidelik.

5.3.3.5 Trelliskode enkodering en vlak omskakeling (prosedure - ENKODERING)

Twee insetbisse vir die trellis enkodeerder word elke baud vanaf die 16-bis datawoord verkry deur die woord twee posisies na links te skuif en die twee uitgeskuifde bisse te gebruik. Die trellis enkodeerder skuifregister word een posisie aangeskuif en die twee insetbisse word ooreenkomstig ingeskuif.

Die modulus-2 sommering van die kodegenerators word gedoen d.m.v. die eksklusiewe-OF funksie. Elk van die kodegenerators se uitsette word getoets en indien dit gestel is, word 'n ooreenkomstige waarde (G1=4, G2=2 en G1=1) gelewer wat gekombineer word om een van agt waardes in die 8-vlak sein te lewer (sien fig. 4.2).

5.3.3.6 8DPSK modulasie en uitset (prosedure - UITSET)

Soos verduidelik in punt 4.2.1.3, word een sinusgolfsiklus verteenwoordig deur 8 monsters. Soortgelyk bestaan die sinus opkyktabel uit 8 punte vir een sinusgolf. Deur die adresregister waarna r2 (sinustabelwyser) wys, opeenvolgend uit te lees en r2 te inkrementeer met 1 word 'n sinusgolf verkry in stappe van 45°. Elke baud (negende monster) word die aanpassingsregister n2 ge-inkrementeer met die ooreenkomstige 8-vlak waarde om 'n faseverandering van (8-vlak waarde) × 45° te veroorsaak. Sodoende word 'n differensieël gemoduleerde sinusgolf gerealiseer.

Die gemoduleerde sinusgolf word na die koppelvlak inset/uitset register $FFEF gelees waarvandaan die D-na-A omskakelaar dit omskakel na 'n analoog sein.

5.4 ONTVANGER SAGTEWARE

Die ontvanger bestaan hoofsaaklik uit die ontv.asm saamsteltaalprogram vir ontvangs van die gestuurde datasein, maar maak ook gebruik van die herstel.asm program. Die herstel.asm en ontv.asm program word deur die tx_rx.pas program in die DSP56001 ontvanger gelaai en geïnisieer, soos onder punte 5.2.2.1, -3 en -4 bespreek.

ONTVANGER	
Gasheer	DSP56001
Pascal program	herstel.asm
(tx_rx.pas)	ontv.asm

Hierdie Pascal program lees ook die ontvangde data vanaf die DSP56001 ontvanger na die gasheerrekenaar (sien punt 5.2.2.6).

5.4.1 Herstel ontvanger program (herstel.asm)

Die herstel.lod program word voor die ontvangs van data in die ontvanger gelaai en uitgevoer om al die eksterne X- en Y geheuelokasies wat deur die ontvanger DSP program gebruik gaan word, na zero te herstel.

5.4.2 Ontvangs van data (ontv.asm)

Die ontvanger saamsteltaalprogram (ontv.lod) word in die DSP56001 gelaai nadat die herstel.lod program uitgevoer is. Die ontv.asm verrig die volgende prosesse soos uiteengesit in die ontvanger blokdiagram (sien fig. 4.17) :

- Monstering van die ontvangde sein en digitalisering deur die A-na-D omskakelaar
- Simboolsinchronisasie
- Korrelasie deteksie
- Viterbi dekodering
- Uitlees van data na gasheerrekenaar

'n Beskrywing van die algehele proses van ontvangs in die DSP56001 kan aan die hand van die struktuurkaart in fig.5.3 gedoen word.

1. Eerstens word die DSP56001 gekonfigureer en die nodige registers en wysers word opgestel.

2. Vervolgens word die ontvangde sein bemonster. Die formaat van die sein soos deur die sender gestuur, word in fig.5.4 getoon. Namate die leerfase ontvang word, word die sinchronisasiedetektor 100 baud gegun om te stabiliseer en simboolsinchronisasie te bewerkstellig (sien fig.5.4).

3. A.g.v. die tekort aan prosesseringstyd word die sinchronisasiedetektor na afloop van die 100 baud sinchronisasie uitgeskakel. Sinchronisasie word dan oorgelaat aan die eksterne koppeling van die sender D-na-A klok aan die ontvanger A-na-D klok soos in fig.3.9 aangedui. Dit beteken dat dieselfde hoeveelheid monsters in 'n tydsinterval

deur die ontvanger geneem gaan word as wat die sender lewer, dit sal dus in sinchronisasie bly.

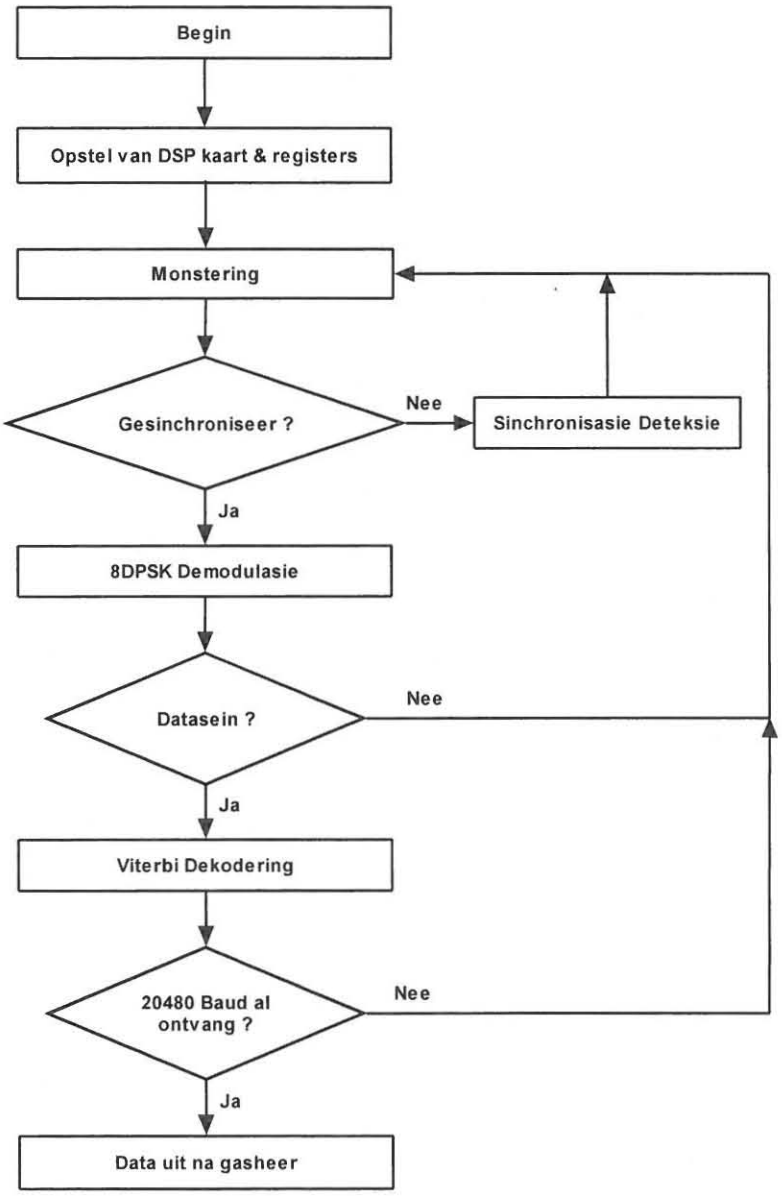


Fig. 5.3 : Struktuurkaart van die DSP56001 ontvanger sagteware

4. Sodra die sinchronisasiedetektor uitgeskakel word, begin demodulasie van die ontvangde sein. Tydens hierdie stadium word daar nog steeds van die leerfase 180° faseveranderings ontvang (sien fig.5.4). Die uitset van die demodulator word dan vir 'n reeks bekende faseveranderings, nl. 10 baud 180°, gevolg deur 10 baud met 0° faseveranderings, getoets. Dit word gedoen om die einde van die leerfase en die begin van die datafase, aan te dui.

5. Sodra die einde van die leerfase (10 baud 0° faseveranderings) geïdentifiseer word, word daar voortgegaan met die demodulasie en dekodering van die datasein.

6. Soos in punt 5.2.2.5 bespreek, ontvang die ontvanger 5 keer 'n datablok van 512 heelgetalle (integers) en lees dit aan die einde uit na die gasheerrekenaar wat dit met die tx_rx.pas program ontvang.

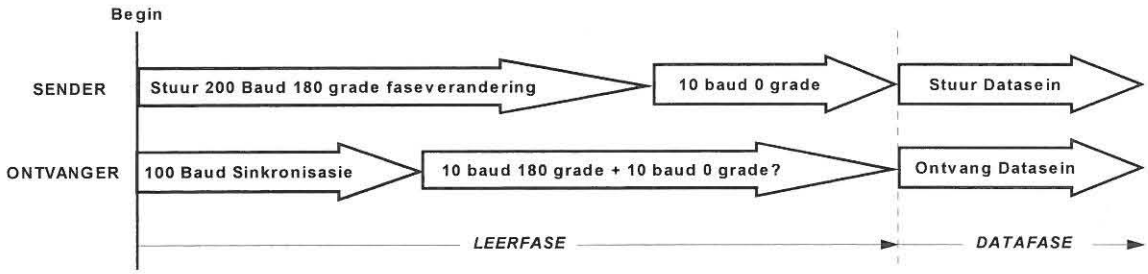


Fig. 5.4 : Tydkonfigurasie van sender en ontvanger prosesse

Bylae B.5 bevat die ontvanger saamsteltaalprogram (ontv.asm) en tabel 5.1 toon die samestelling van die prosedures en sub-prosedures daarin.

Tabel 5.1 : Uiteensetting van ontvanger saamsteltaalprogram.

ONTVANGER SAAMSTELTAALPROGRAM (ontv.asm) Prosedures :			
HOOF Sub-prosedures :	**SYNC_DETEKTOR Sub-prosedures :**	**DEMODULATOR Sub-prosedures :**	**VITERBI Sub-prosedures :**
OPSTEL	DETEKSIE	AFSTEL	NODUS_A
LEES_MONSTER	ZERO_DET_9	FILTER_DET_DEMOD	NODUS_B
LEES_UIT	TEL&HERSTEL	HERSTEL	NODUS_C
		PAS_BAUD	NODUS_D
		FASE_TOETS	NODUS_E
			NODUS_F
			NODUS_G
			NODUS_H
			SKRYF_NUWE_IN_OU
			KORTSTE_VAN_PAAIE
			TERUG_SPOOR
			DATA_NA_INT

Die ontvanger saamsteltaalprogram sal vervolgens aan die hand van die struktuurkaart in fig.5.3 en tabel 5.1 bespreek word.

5.4.2.1 DSP Konfigurasie en opstel van ontvanger beheerregisters en wysers (prosedure - HOOF, sub-prosedure - OPSTEL).

Die DSP56001 word net soos by die sender gekonfigureer om in modus-2 te funksioneer.

Die interne sinustabelwaardes van die DSP56001 prosesseerder word afgeskaal - vir latere berekeninge - deur dit 14 posisies na regs te skuif - wat beteken dat die waardes deur 2^{14} gedeel word. Die geskaalde sinustabel word in geheuelokasies X:$500 tot X:$5ff gestoor.

Poort C word opgestel vir algemene inset/uitset doeleindes met die onderbrekings ontmagtig. Die analoog koppelvlakkring se tydkonfigurasiewaardes van toepassing op die A-na-D omskakeling en inset banddeurlaatfilter word opgestel met RA = 15 en RB = 36. Dit verteenwoordig die aangepaste waardes soos in punte 4.3.1.1 en 4.3.1.2 bereken. Die koppelvlak beheerregister word opgestel met die verstek waardes.

Registers en wysers soos later benodig in die sinchronisasie, demodulasie en dekoderingsprosesse word ooreenkomstig gestel.

5.4.2.2 Lees van 'n monster vanaf A-na-D (prosedure - LEES_MONSTER).

Die A-na-D omskakelaar monster die ontvangde sein teen die vooraf bepaalde frekwensie (4800 monsters per sekonde). Hierdie monster waarde word gelees vanaf die DSP56001 koppelvlak inset/uitset register (x:$ffef) en word geskaal deur die monsterwaarde 12 posisies na regs te skuif $(\div 2^{12})$.

Vir die ontvanger om intyds te funksioneer moet die demodulasie en dekodering van 'n simbool voltooi wees voordat daar met die volgende simbool se demodulasie en

dekodering begin word. Dus, sodra al 9 monsters van 'n simbool ontvang is, moet die simbool gedemoduleer en gedekodeer word voordat die volgende simbool ontvang word.

Die tydinterval vanaf die laaste monster van een simbool tot die eerste monster van 'n volgende is egter onvoldoende om al die demodulasie- en dekoderingsprosesse van die volledige ontvangde simbool uit te voer. Daarom word van die prosedures tussen die neem van monsters uitgevoer sodat die beskikbare tyd optimaal benut word om die simbooluitset te lewer voordat al 9 monsters van die volgende simbool ontvang is (sien fig.5.5).

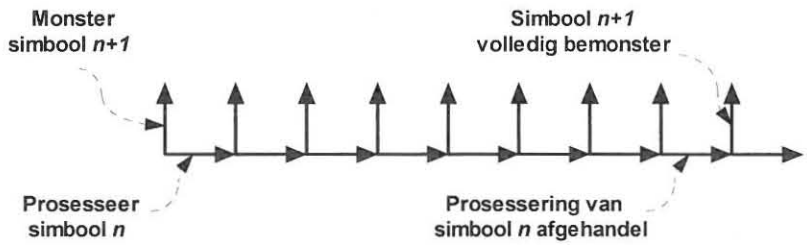


Fig. 5.5 : Prosessering van simbool in tyd

5.4.2.3 Sinchronisasie (prosedure - SYNC_DETEKTOR)

Soos reeds genoem word die sinchronisasiedetektor 100 baud gegun om sinchronisasie te bewerkstellig voordat demodulasie begin. Die bepaling van die sinchronisasiesein, en die ooreenkomstige sinchronisering van die ontvanger t.o.v. die ontvangde sein, kan aan die hand van die vloeidiagram in fig.5.6 beskryf word.

5.4.2.3.1 Bepaling van sinchronisasiesein waarde (prosedure - DETEKSIE).

Soos in die simulasie word die ontvangde sein geïntegreer oor 8 monsters (sien punt 4.3.2.1.2). Dit word bewerkstellig deur die ontvangde monster in 'n 8-stap sirkuleringsregister in te lees en die som van die 8 waardes in die sirkuleringsregister te bepaal.

Na die 8-monster integrasie word die absolute waarde van die integrale waarde bepaal. Soos met die 8-monster integrasie hierbo word die 2-monster integrasie (sien punt 4.3.2.1.3) op soortgelyke wyse bewerkstellig met 'n 2-stap sirkuleringsregister.

Vir die bepaling van die gemiddelde sinchronisasiesein waarde oor 8 simbole (ontvangde simbool + n-baud soos na verwys in punt 4.3.2.1.4) word die uitset van die 2-monster integreerder in 'n 64-stap sirkuleringsregister gestoor. Dit word dan gesommeer met die vorige 2-monster integreerder uitset waardes wat 1,2 tot 7 -simbole terug bereken is - elk in veelvoude van 9 stappe terug in die 64-stap sirkuleringsregister.

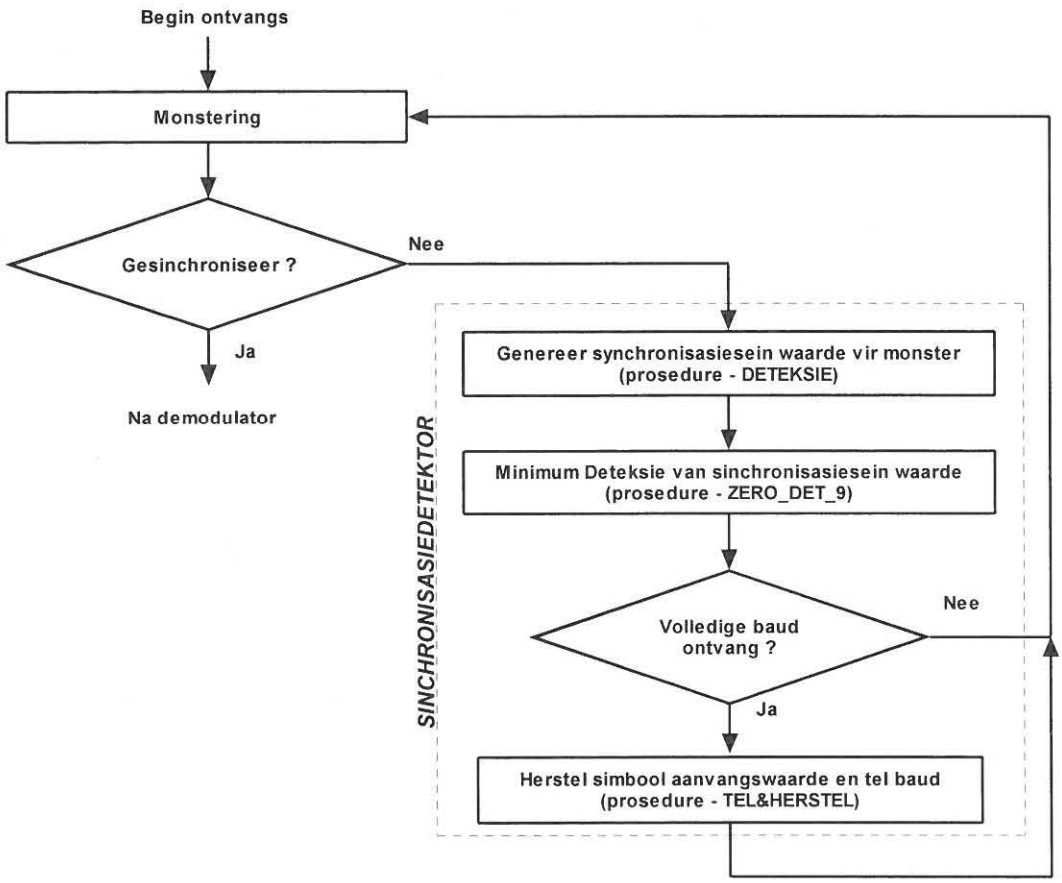


Fig. 5.6 : Vloeidiagram van prosedure - SYNC_DETEKTOR

5.4.2.3.2 Deteksie van minimum sinchronisasiesein waarde (prosedure - ZERO_DET_9)

Hierdie prosedure doen zero of minimum deteksie van die sinchronisasiesein oor 'n 9 monster venster om die ontvangde monsters op te deel in gesinchroniseerde 9 monster

simbole. Die ZERO_DET_9 prosedure lewer 'n "gesinchroniseerde simbool", waarvan die eerste monster, die monster met die kleinste sinchronisasiesein waarde in die 9 monster venster is. Die "gesinchroniseerde simbool" se 9 monsters word in geheueposisies y:$250 - y:$258 gestoor.

Soos in die simulasie van die sinchronisasiedetektor waargeneem (sien punt 4.3.2.1.4) dui die minimum waarde van die sinchronisasiesein op die laaste monster in 'n simbool. D.w.s. in geheuelokasie y:$250 is die laaste monster van 'n simbool s_{T-1} en in geheuelokasies y:$251 - 258 die eerste 8 monsters van die volgende simbool, s_T.

5.4.2.3.3 Herstel van wysers en tel baud (prosedure - TEL&HERSTEL)

Sodra 'n "gesinchroniseerde simbool" geïdentifiseer is, word die nodige wyser registers herstel en die waarde van die aantal baud sover ontvang word ge-inkrementeer.

5.4.2.4 8DPSK Demodulasie

Die implementering van die gewysigde filterdetektor word vervolgens aan die hand van die vloeidiagram in fig.5.7 bespreek.

A.g.v. sinchronisasie wat klaar bewerkstellig is (wanneer daar oorgeskakel word vanaf die sinchronisasiedetektor na die demodulator) word die "gesinchroniseerde simbole", verkry deur die geneemde monsters opeenvolgend in 9 monster groepe te stoor.

5.4.2.4.1 Afstel van simbool (prosedure - AFSTEL)

Soos reeds genoem in punt 5.4.2.3.2 bestaan die "gesinchroniseerde simbool" (y:$250 - y:$258) nie uit die korrekte 9 monsters van 'n betrokke simbool nie. Om hierdie korreksie te doen moet die "gesinchroniseerde simbool", vertraag word met een monster. Verder is in punt 4.4 genoem dat die sinchronisasiesein 90° of 2 monsters geskuif het t.o.v. die korrekte oomblik van faseverandering tussen simbole a.g.v. die monstering en filtrering van die sein. Daarom word die "gesinchroniseerde simbool" in totaal met 3 monsters

vertraag om die korrekte 9 monster, gesinchroniseerde simbool (x:$290 - x:$298), te lewer vir demodulasie.

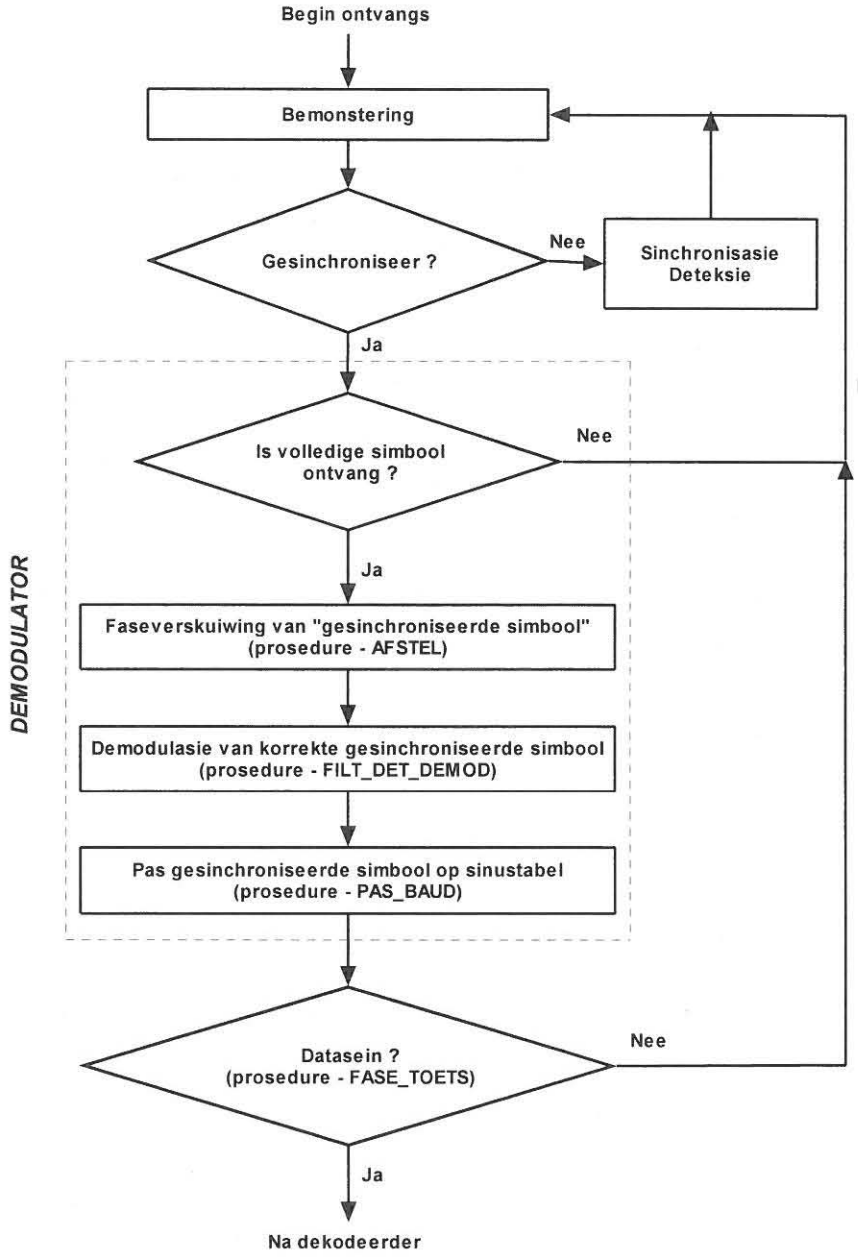


Fig. 5.7 : Vloeidiagram van prosedure - DEMODULATOR

5.4.2.4.2 Filterdetektor (prosedure - FILT_DET_DEMOD)

Net soos in die simulasie (sien punt 4.3.2.2.1) word die ontvangde simbool (x:$290 - x:$298) gekorreleer met vier faseverskuiwings (0°, 45°, 90° en 135°) relatief tot die vorige ontvangde simbool. Die ontvangde simbool word vermenigvuldig met elke

faseverskuiwing. Na die vermenigvuldigingsproses word die produkte gesommeer oor 'n 9 monster venster om die 4 korrelasie waardes te lewer. Verder word elkeen van die 4 faseverskuiwings se groep-maat (sien fig.4.19) se korrelasie waarde bepaal, deur die negatiewe korrelasie waarde van die ooreenkomstige faseverskuiwing (0°, 45°, 90° of 135°) te neem. Op hierdie wyse word die waardes t.o.v. 225° tot 315° bepaal. Die 8 korrelasie waardes van faseverskuiwing 0° tot 315° word gestoor in geheuelokasies x:$260 tot x:$267. Die grootste korrelasie waarde word geïdentifiseer en in geheuelokasie x:$26a gestoor - met die ooreenkomstige fase in geheuelokasie x:$26b.

Laastens word die verskil-in-korrelasie waardes t.o.v. die grootste korrelasie waarde bepaal en gestoor in geheuelokasies y:$300 - y:$307 vir gebruik in die Viterbi dekoderingsproses (soos in punt 4.3.1.5 bespreek).

5.4.2.4.3 Generering van addisionele monsters (prosedure - PAS_BAUD)

Die proses - PAS_BAUD neem die ontvangde simbool (x:$290 - x:$298) en bepaal 'n ooreenkomstige beginpunt (eerste monster (y:$208)) daarvan op die geskaalde sinustabel.

A.g.v. die probleem van te min monsters beskikbaar vir bevredigende korrelasie (soos in punt 4.3.1.4 bespreek) word die ontvangde simbool (x:$290 - x:$298) gepas op die geskaalde weergawe van die interne sinustabel (y:$500 - y:$5ff). Dit word gedoen om die fase van die ontvangde simbool vanaf die sinustabel te verkry. Deur van 'n sirkuleringsbuffer oor die sinustabel strek gebruik te maak, kan die ontvangde simbool effektief verleng word vir korrelasie doeleindes.

Hierdie sinustabel weergawe van die ontvangde simbool word gebruik as die vertraagde simbool d.w.s. simbool S_{T-1} vir korrelasie met die volgende ontvangde simbool S_T tydens die demodulasieproses. 'n Voordeel hieraan verbonde is die feit dat die ontvangde simbool S_T gekorreleer word met 'n sinusgolf sonder enige verwringing of ruis.

5.4.2.4.4 Toets vir datasein identifikasie (prosedure - FASE_TOETS)

Nadat die sinchronisasiedetektor afgeskakel is en die demodulator aangeskakel is, word die fase uitset van die demodulator getoets vir 'n reeks faseveranderings van 10 baud 180° en 10 baud 0° opeenvolgend. Hierdie toetsing word deur prosedure FASE_TOETS gedoen, wat 'n datasein vlag (y:$20b) stel indien die uitset van die demodulator aan bogenoemde vereiste voldoen.

5.4.2.5 Viterbi dekodering

Die werking van die Viterbi dekoderingsproses kan t.o.v. die volgende stappe gelys word.

1. Neem die 8 afstande (verskil-in-korrelasie waardes) en bepaal die wennerstap na elkeen van die 8 nodusse in die trellisboom.
2. Stoor die 8 wennerstappe as die nuutste stappe in die betrokke 8 paaie deur die trellis. Stoor ook die totale afstande (som van verskil-in-korrelasie waardes) van die onderskeie paaie.
3. Vergelyk die 8 paaie se totale afstande om die pad met die kortste afstand (wennerpad) te verkry.
4. Lees die datawoord uit wat 15 baud ($5 \times m$) terug in die wennerpad voorkom.

Die Viterbi dekoderingsproses word vervolgens aan die hand van die vloeidiagram in fig.5.8 bespreek.

5.4.2.5.1 Bepaling van die wennerstap tot by elke nodus in die trellis (prosedure - NODUS_A, B, C, D, E, F, G, H)

Punte 1 en 2 van die dekoderingsproses soos hierbo genoem word eerstens uitgevoer t.o.v. die 8 nodusse A - H. A.g.v. die herhalende aard van die prosedure t.o.v. die verskillende nodusse word slegs een van die prosedures (NODUS_A) bespreek. Tabel 5.2 toon die ooreenkomste en verskille tussen die verskillende nodusse soos afgelei van die trellis in bylae C.

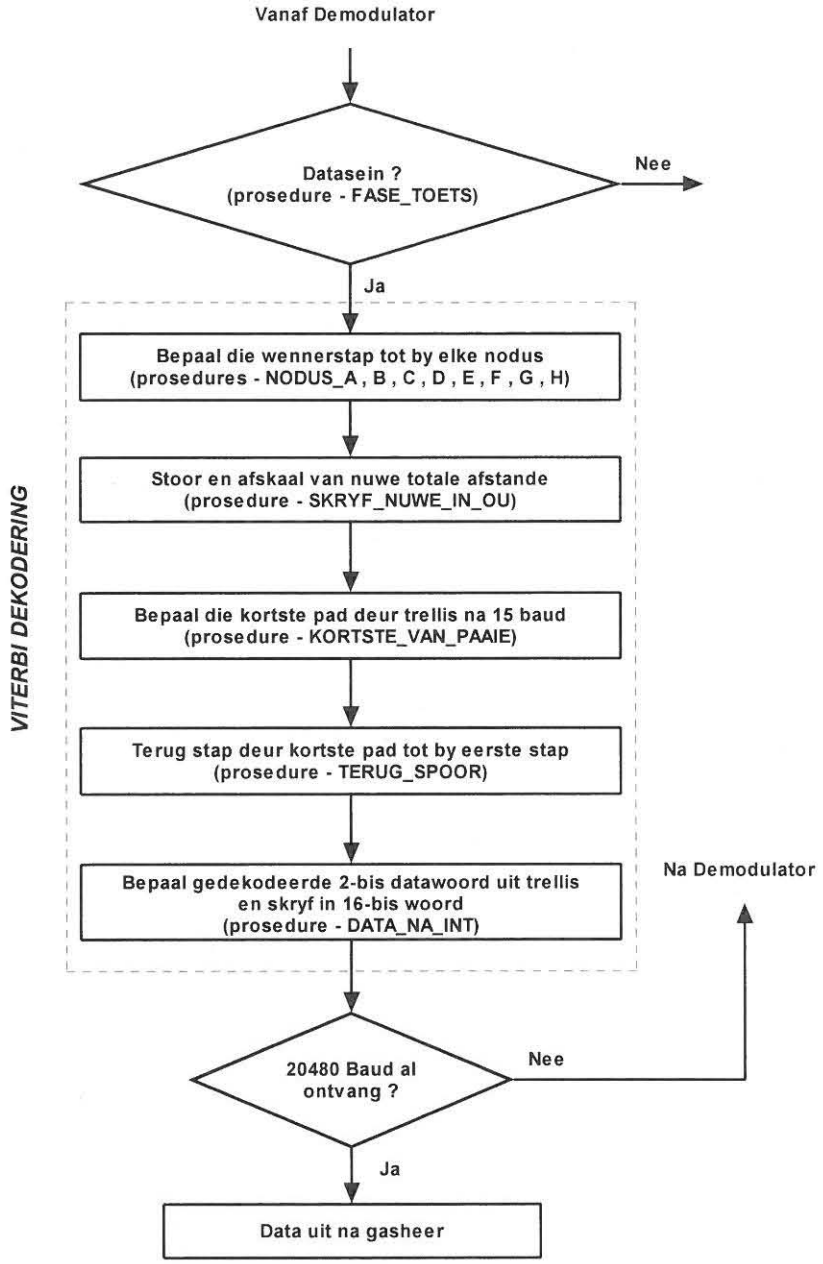


Fig. 5.8 : Vloeidiagram van prosedure - VITERBI

Die stappe vanaf 'n nodus is die moontlike stappe in die trellis wat gevolg kan word vanaf die betrokke nodus. Die moontlike stappe na 'n nodus is die moontlike stappe wat in die trellis gevolg kan word om by die betrokke nodus uit te kom vanaf die ooreenkomstige oorsprong nodusse (as nodusnommers aangedui). Die moontlike stappe stel die moontlike ontvangde gekodeerde simbole - fases - voor waar elke moontlike stap 'n sekere afstand (verskil-in-korrelasie waarde) besit soos deur die demodulasieproses bepaal.

Tabel 5.2 : Ooreenkomste en verskille van nodusse in die trellis.

Nodus (staat)	Nodus (nr.)	Stappe vanaf nodus (simbool)	Stappe na nodus (simbool)	Oorsprong nodus (nr.)
A	0	0 , 4 , 2 , 6	0 , 4 , 2 , 6	0 , 2 , 4 , 6
B	1	1 , 5 , 3 , 7	4 , 0 , 6 , 2	0 , 2 , 4 , 6
C	2	4 , 0 , 6 , 2	2 , 6 , 0 , 4	0 , 2 , 4 , 6
D	3	5 , 1 , 7 , 3	6 , 2 , 4 , 0	0 , 2 , 4 , 6
E	4	2 , 6 , 0 , 4	1 , 5 , 3 , 7	1 , 3 , 5 , 7
F	5	3 , 7 , 1 , 5	5 , 1 , 7 , 3	1 , 3 , 5 , 7
G	6	6 , 2 , 4 , 0	3 , 7 , 1 , 5	1 , 3 , 5 , 7
H	7	7 , 3 , 5 , 1	7 , 3 , 5 , 1	1 , 3 , 5 , 7
Stap no :		0 , 1 , 2 , 3	0 , 1 , 2 , 3	

- **Sub-prosedure VIER_AFSTANDE :** Die eerste stap in prosedure - NODUS_A is om die 4 stap-afstande (verskil-in-korrelasie waardes) van die 4 moontlike stappe (0, 4, 2, 6) na nodus A te sommeer met die totale afstand van die pad wat eindig by die nodus vanwaar die 4 onderskeie stappe hul oorsprong het.

- **Sub-prosedure KORTSTE_STAP :** Nadat die 4 totale afstande verkry is, word die totale afstande vergelyk en die stap met die kortste totale afstand word geïdentifiseer en geneem as die wennerstap of mees waarskynlike stap na nodus A.

- **Sub-prosedure SKRYF_IN_MATRIKS :** Die kortste totale afstand na nodus A en die oorsprong nodus van die kortste stap, soos deur prosedure KORTSTE_STAP bepaal, word nou in 'n matriks gestoor. Tabel 5.3 toon die matriks vir al die nodusse (A - H) waar die X-geheuelokasies die oorsprong nodusse na elke nodus stoor en die Y-geheuelokasies die totale afstand tot by die betrokke nodus.

Die matriks stoor al die oorsprong nodusse en totale afstande na elke nodus (A - H) oor 'n 16 baud venster. Al die nodusse (A - H) se totale afstande en oorsprong nodusse word instap met mekaar t.o.v. die ontvangde simbool in die matriks gestoor deur die matriks as 'n 16 stap sirkuleringsregister te laat funksioneer.

Tabel 5.3 : Matriks geheuelokasies

MATRIKS		
NODUS	**OORSPRONG NODUS**	**AFSTAND**
A (0)	X:$400 - X:$40F	Y:$400 - Y:$40F
B (1)	X:$410 - X:$41F	Y:$410 - Y:$41F
C (2)	X:$420 - X:$42F	Y:$420 - Y:$42F
D (3)	X:$430 - X:$43F	Y:$430 - Y:$43F
E (4)	X:$440 - X:$44F	Y:$440 - Y:$44F
F (5)	X:$450 - X:$45F	Y:$450 - Y:$45F
G (6)	X:$460 - X:$46F	Y:$460 - Y:$46F
H (7)	X:$470 - X:$47F	Y:$470 - Y:$47F

Prosedures VIER_AFSTANDE, KORTSTE_STAP en SKRYF_IN_MATRIKS, soos hierbo na verwys, word vir nodusse A - H t.o.v. elke ontvangde simbool uitgevoer.

5.4.2.5.2 Stoor van totale afstande as oorsprong afstande (prosedure - SKRYF_NUWE_IN_OU)

Nadat die totale afstande tot by elke nodus bepaal en in die matriks geskryf is, word dit gestoor as die nuwe oorsprong afstande (afstande tot by nodusse A - H) vir gebruik tydens sub-prosedure VIER_AFSTANDE van die volgende ontvangde simbool.

5.4.2.5.3 Bepaal die kortste pad (prosedure - KORTSTE_VAN_PAAIE)

Prosedure KORTSTE_VAN_PAAIE vergelyk die totale afstande tot en met die 8 mees onlangse nodusse in die matriks en die nodus met die kortste afstand word gestoor in register (y:$370). Hierdie nodus is dan die laaste nodus in die kortste pad van die 8 moontlike wennerpaaie deur die trellis.

5.4.2.5.4 Bepaal van nodus 15 baud terug (prosedure - TERUG_SPOOR)

Prosedure TERUG_SPOOR neem die laaste nodus in die kortste pad soos deur prosedure KORTSTE_VAN_PAAIE bepaal is en stap terug deur die matriks om die nodus 15 stappe terug in die kortste pad te lewer.

5.4.2.5.5 Verkry datawoord uit trellis en stoor in 16-bis heelgetal (prosedure - DATA_NA_INT)

Die nodus waarde (soos deur prosedure TERUG_SPOOR gelewer) word geneem en die stap-nommer (sien laaste ry in die stappe_na_nodus kolom van tabel 5.2) van die stap wat lei na die betrokke nodus word bepaal. Hierdie stap-nommer word dan geneem as die 2-bis gedekodeerde datawoord.

Die 2-bis datawoord word dan in 'n 16-bis woord ingeskuif wat gestoor word sodra dit gevul is met agt 2-bis datawoorde. Die 16-bis woorde word uitgelees na die gasheerrekenaar sodra 20480 baud datawoorde (5 × 512 16-bis woorde) ontvang is.

5.5 OPSOMMING

Die sender bestaan uit hardeware (SIG56 kaart) en sagteware. Die sagteware bestaan uit twee hoofkomponente nl, die gasheerrekenaar Pascal program (tx-rx.pas) en die DSP56001 saamsteltaalprogramme (herstel.asm, data_in.asm en send.asm). Die Pascal program beheer die laai van die saamsteltaalprogramme na die DSP56001 asook die inisiasie daarvan. Die saamsteltaalprogramme herstel die DSP56001, laai data in samewerking met die Pascal program en doen die nodige seinverwerking om die 8DPSK trelliskodemodulasie sein te lewer.

Die ontvanger, net soos die sender, bestaan ook uit hardeware (SIG56 kaart) en sagteware. Die sagteware bestaan uit twee dele nl. die gasheerrekenaar Pascal program (tx_rx.pas) en DSP56001 saamsteltaalprogramme (herstel.asm en ontv.asm). Die Pascal program laai die saamsteltaalprogramme, inisieer dit, en lees die ontvangde data uit vanaf die DSP56001 in samewerking met die ontv.asm saamsteltaalprogram in die

DSP56001. Al die nodige seinverwerking vir die demodulasie en dekodering van die ontvangde sein, word deur die ontv.asm saamsteltaalprogram in die ontvanger DSP56001 verrig.

Tydens die ontwikkeling van die sagteware is die werking van beide die sender en ontvanger saamsteltaal sagteware prosedures deurentyd getoets. Dit is gedoen deur die prosedure/afdeling se uitset na 'n lêer uit te lees en dit dan in MCAD te korreleer met die verwagte gesimuleerde waardes. 'n Voorbeeld van so 'n toets is fig.5.9 wat die uitset korrelasie waardes van die filterdetektor in die DSP56001 ontvanger toon vir 'n tipiese reeks ontvangde data vanaf die DSP56001 sender. Die leerfasesein met die aanvanklike 180° faseveranderings en 10 baud 0° faseveranderings gevolg deur pseudo-ewekansige data kan duidelik waargeneem word.

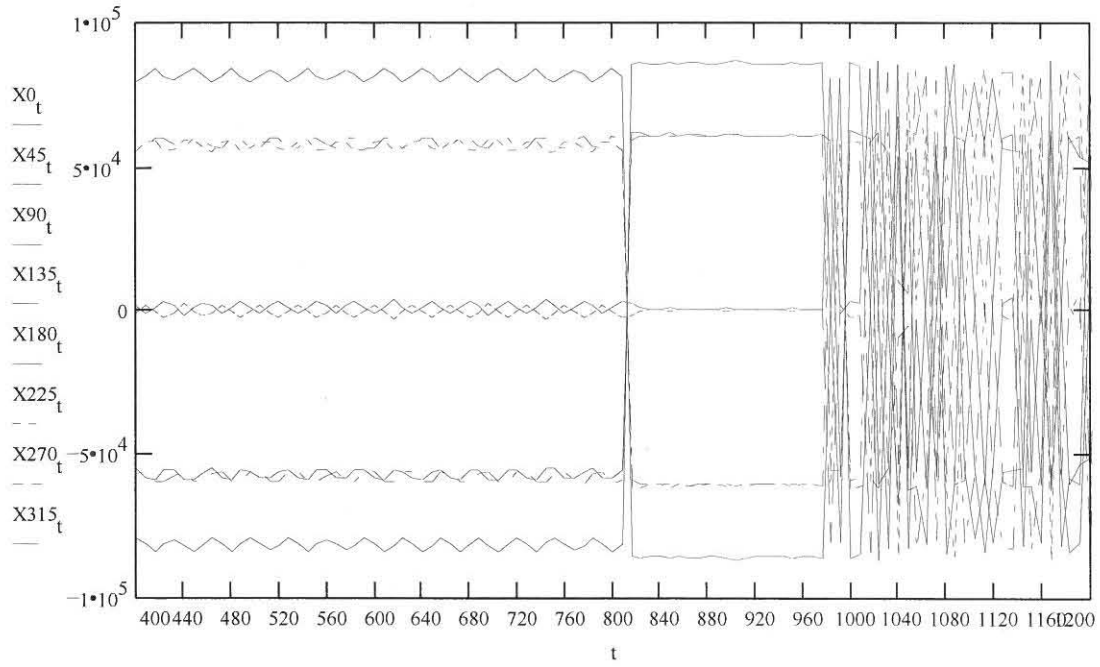


Fig. 5.9 : 8 Korrelasie uitset waardes van die filterdetektor in die DSP56001 ontvanger

Deur bogenoemde toetsprosedure kon die werking van die meeste saamsteltaal prosedures in die DSP56001 bevestig word voordat daar voortgegaan is met die ontwikkeling van 'n volgende prosedure.

6. EVALUERING

6.1 INLEIDING

Om 'n algehele beeld van die ontwikkelde stelsel se prestasievermoë te verkry, is dit t.o.v. die volgende aspekte geëvalueer :

1. Konstellasie verspreiding van die gestuurde en ontvangde seine.
2. Die spektrale eienskappe van die sender uitsetsein.
3. Die simboolfouttempo (SER) van die 8DPSK demodulator.
4. Die bisfouttempo (BER) van die stelsel met en sonder Viterbi dekodering.

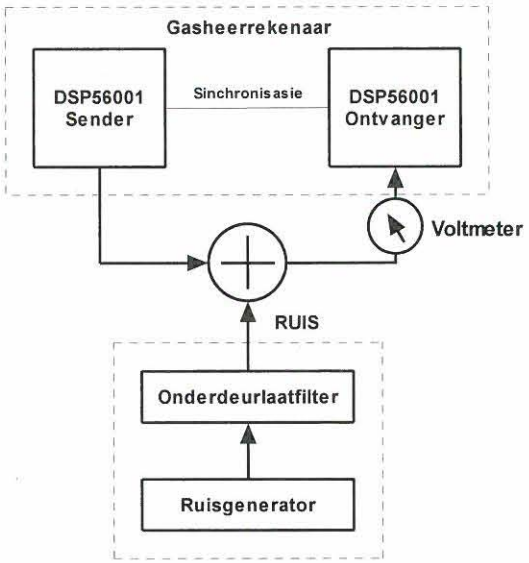


Fig. 6.1 : Opstelling van die stelsel tydens evaluering

Tydens die evaluering en die neem van metings was die stelsel opgestel soos in fig.6.1 getoon, onderworpe aan die volgende spesifikasies :

1. Sender monstertempo = 4800Hz.
2. Ontvanger monstertempo = 4800Hz.
3. Simboolsinchronisasie is deur die ontvanger bewerkstellig tydens die leerfase waarna die eksterne koppeling tussen die D-na-A en A-na-D klokke die monstering tussen die sender en ontvanger instap gehou het.

4. Die "kanaal" tussen die sender en ontvanger is baie kort en het basies geen invloed op die stelsel, behalwe vir ruis wat ekstern gegenereer en tot die "kanaal" toegevoeg is nie.
5. Die voltmeter gebruik vir die neem van w.g.k. spanningswaardes van die gemoduleerde- en ruissein was 'n Rhode & Schwartz videoruismeter (model: BN120312).

6.2 RUISGENERATOR

Vir die generering van ruis tydens die toetsing van die ontvanger, is 'n wyeband ruisgenerator gekonstrueer. Fig.6.2 toon die kringdiagram van die ruisgenerator *[7]*.

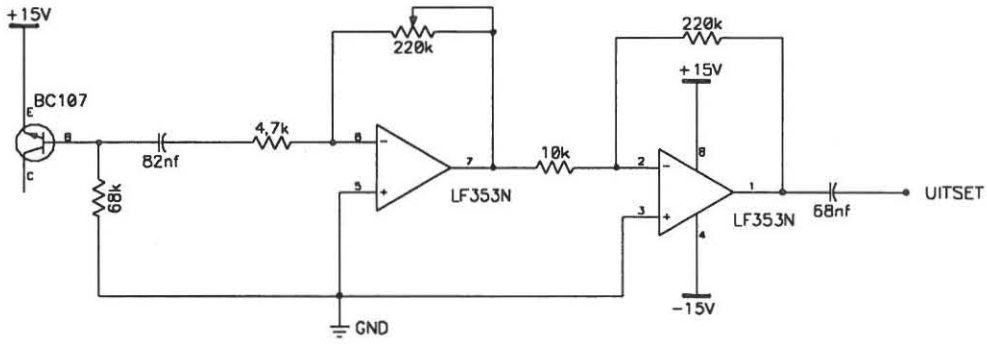


Fig. 6.2 : Wyeband ruisgenerator

Die ruis word gegenereer deur die basis-emitter voegvlak van 'n transistor (BC107) teen voor te span tot in die deurbreekgebied. Die stroom deur die PN voegvlak lewer 'n ruisspanning. Die ruis bestaan uit termiese ruis asook haelruis.

Die ruisspanning van die PN voegvlak word dan deur twee wyeband LF353 operasionele versterkers versterk. Die ruiswins word verstel deur die verstelbare terugvoerweerstand van die eerste operasionele versterker. Fig.6.3 toon die frekwensiespektrum van die ruissein oor 'n strek van 0 - 10 kHz. Alhoewel die ruisgenerator nie ideale witruis met oneindige bandwydte en amplitude genereer nie, lewer dit 'n konstante spektrum in die ontvanger bandwydte (0Hz - 2040 Hz).

Om die korrekte w.g.k. waarde van die ruis in die toepaslike frekwensiestrek (2040Hz) van die stelsel te bepaal, word die ruis vanaf die ruisgenerator gefilter met 'n onderdeurlaatfilter van 0Hz - 2300Hz. Bylae D toon die kringdiagram van die genoemde filter (aangepas vanaf *[26, p. 89]*). Die karakteristiek van die aangepaste filter is eksperimenteel bepaal en word in fig.6.4 getoon.

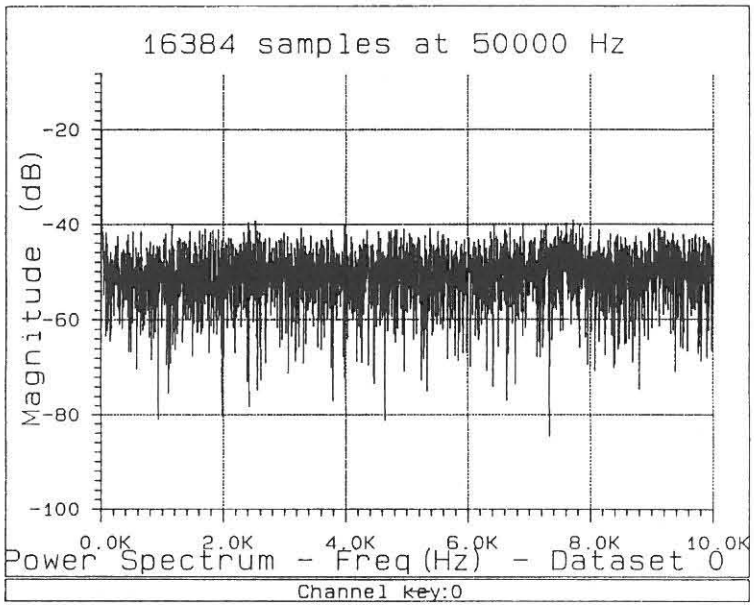


Fig. 6.3 : Ruisgenerator frekwensiespektrum 0 - 10 kHz

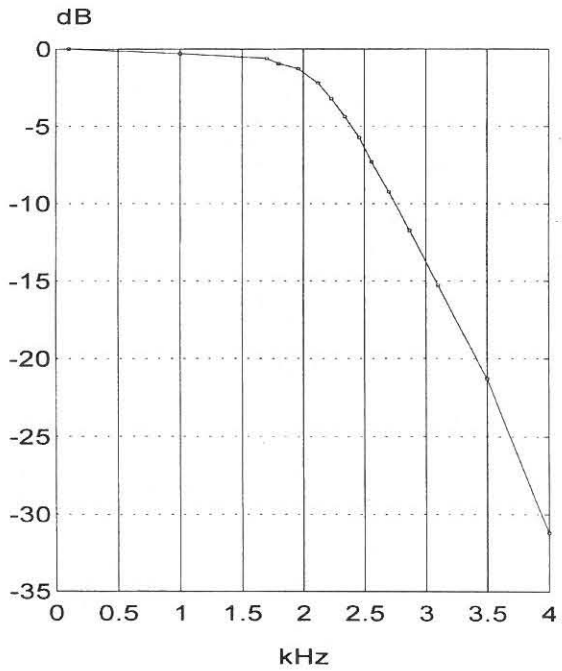


Fig. 6.4 : Karakteristiek van die onderdeurlaatfilter

Die gefilterde ruis word met die DSP sender uitset gesommeer d.m.v. 'n operasionele versterkerkring (sien bylae D). Die uitset van die sommeerder word herlei na die ontvanger waar die ruisbesoedelde sein ontvang word.

6.3 SENDER RESULTATE

Die volgende metings is geneem t.o.v. die sender uitset :

1. Uitsetgolfvorm.
2. Konstellasie verspreiding.
3. Frekwensiespektrum.

6.3.1 Sender uitsetgolfvorm

Figuur 6.5 toon die uitsetgolfvorm van die sender vir 'n reeks pseudo-ewekansige data. Die konstante amplitude van die sein ('n eienskap van PSK) asook die faseveranderings tydens elke simbool kan duidelik waargeneem word.

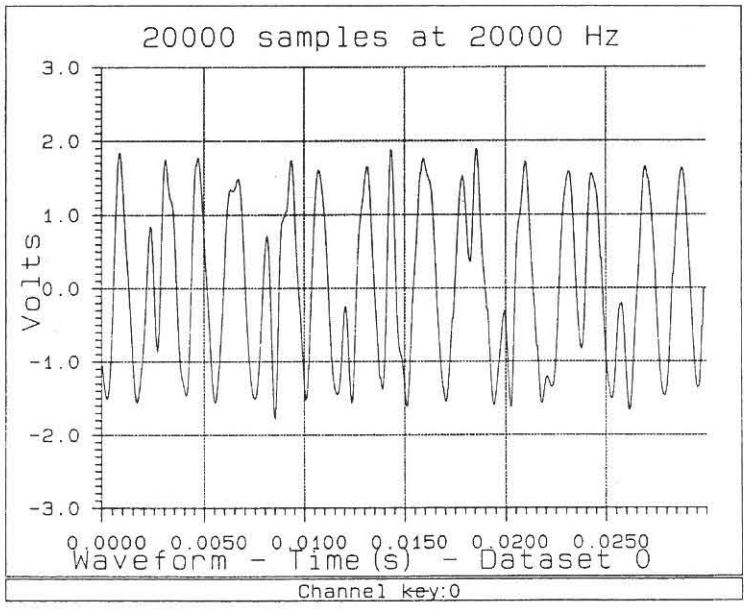
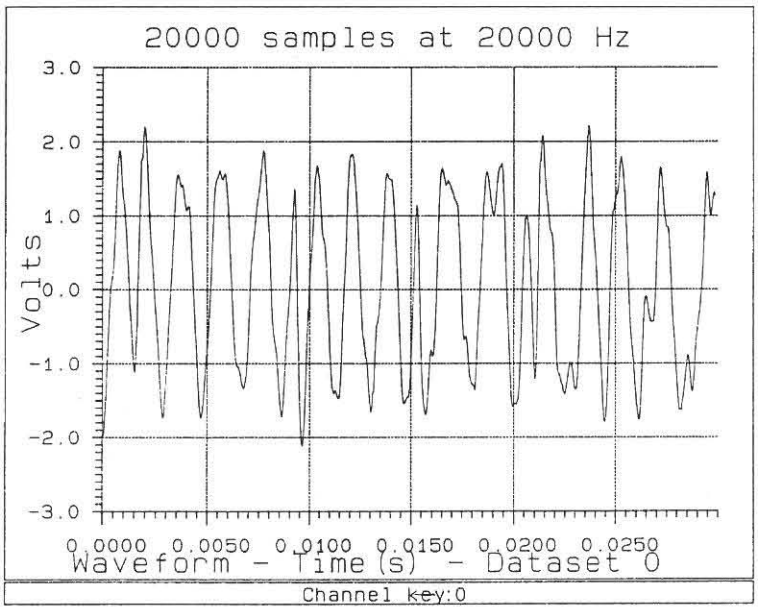
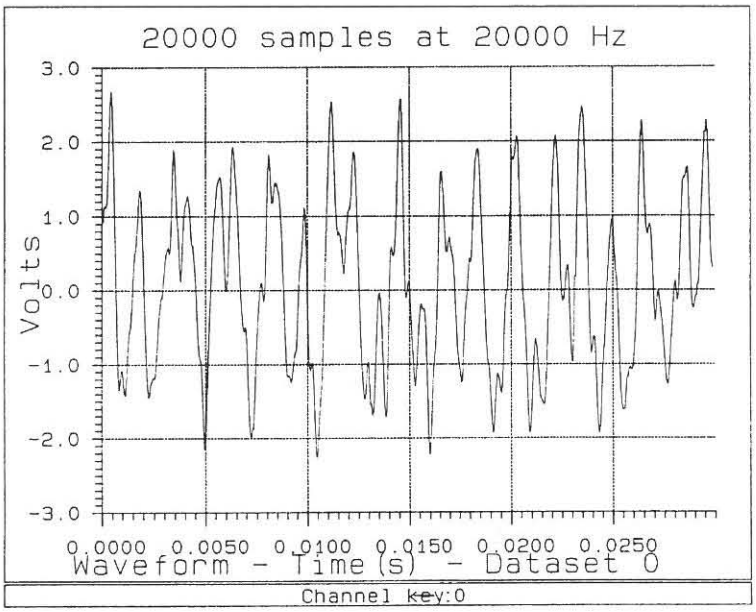


Fig. 6.5 : Pseudo-ewekansige data uitsetsein van die sender

Die sein in fig.6.5 is soos gelewer deur die sender, onaangeraak deur enige ruis. Figure 6.6(a) en 6.6(b) toon hoe ruis die golfvorms van twee kort reekse pseudo-ewekansige dataseine verwring. Fig.6.6(a) illustreer 'n sein met 'n seinruisverhouding (*E/No*) van 21dB en fig.6.6(b) 'n verhouding van 15dB.



(a)



(b)

Fig. 6.6 : Datasein teen (a) 21dB *E/No* en teen (b) 15dB *E/No*

6.3.2 Konstellasie verspreiding.

Die konstellasie van 'n reeks data soos deur die sender gestuur, is verkry deur die sender uitset te bemonster teen 4800 bisse per sekonde en die monsters te stoor in 'n lêer. Die monsterwaardes is toe uit die lêer gelees en simbool-gesinchroniseer d.m.v. 'n gesimuleerde MCAD sinchronisasiedetektor. Die faseveranderings van die opeenvolgende simbole is relatief tot mekaar op 'n konstellasie geplot (sien bylae A.1 vir simulasie).

Fig.6.7 toon die konstellasie van 'n kort reeks pseudo-ewekansige data soos deur die sender gelewer. Alhoewel hierdie meting nie kwantantief van aard is is nie, dui die konstellasie duidelik die stabiliteit of vermoë van die sender om konstante faseveranderings te genereer aan.

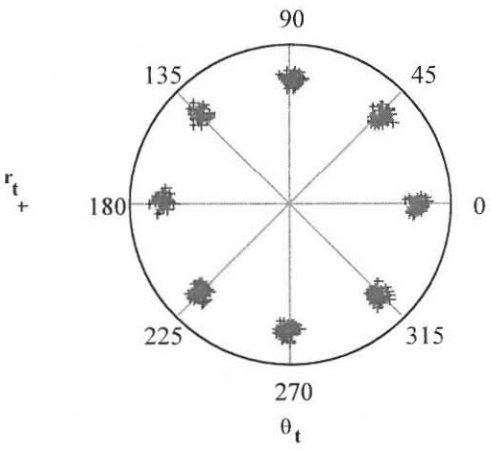


Fig. 6.7 : Konstellasie van die sender uitset

6.3.3 Spektrale eienskappe van die sender uitset

Die frekwensiespektrum van die sender uitsetsein in fig.6.5 word in fig.6.8 getoon.

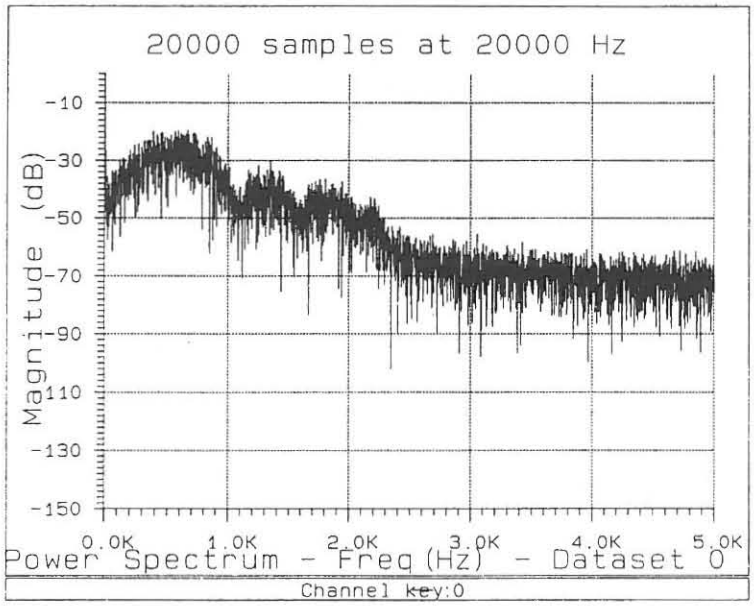


Fig. 6.8 : Frekwensiespektrum van die sender uitset

6.4 ONTVANGER RESULTATE

Die ontwikkelde 8DPSK ontvanger met Viterbi dekodering is getoets teenoor 'n identiese 8DPSK ontvanger sonder Viterbi dekodering. Die volgende metings is geneem by die ontvanger :

1. Konstellasie verspreiding van die ontvangde sein.
2. Die simboolfouttempo van die 8DPSK demodulator (kontrole ontvanger).
3. Die bisfouttempo van die 8DPSK demodulator met Viterbi dekodering (toets ontvanger).

6.4.1 Konstellasie verspreiding

Die konstellasie van die ontvangde sein is net soos by die sender bepaal (sien punt 6.3.2), behalwe dat die waardes wat gebruik word, ná die banddeurlaatfilter in die ontvanger uitgelees en gestoor word vir die MCAD verwerking. Die konstellasies wat hier volg is dus die konstellasies soos die 8DPSK demodulator die ontvangde sein waarneem.

Ter illustrasie van die effek van ruis op die konstellasie van die gestuurde sein toon figure 6.9(a) tot (c) die konstellasie van kort reekse pseudo-ewekansige data t.o.v. verskeie seinruisverhoudings. Fig.6.9(a) toon 'n konstellasie met geen ruis, fig.6.9(b) 'n konstellasie met 'n 21dB seinruisverhouding (*E/No*) en fig.6.9(c) 'n konstellasie met 'n 15dB seinruisverhouding (*E/No*).

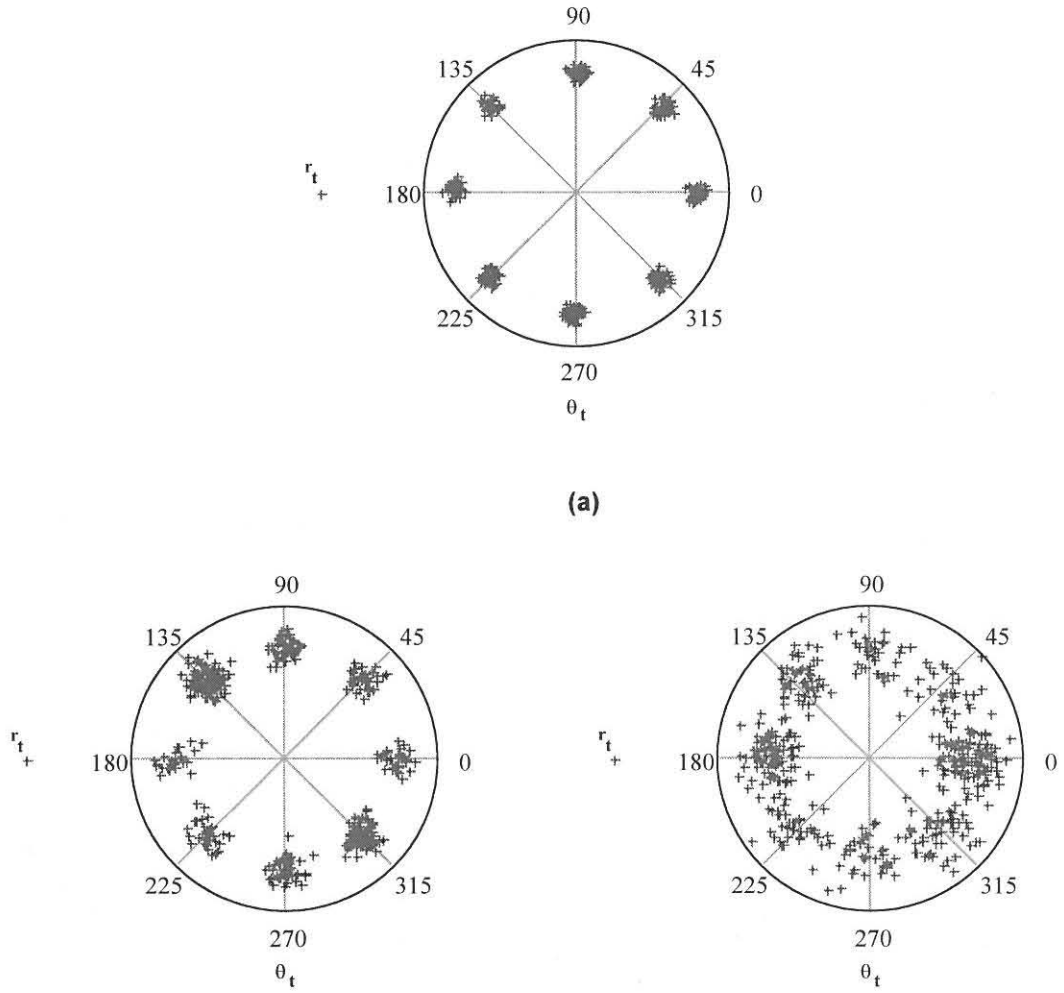


Fig. 6.9 : Ontvanger konstellasie met (a) geen ruis, (b) 21dB *E/No* en (c) 15dB *E/No*

Uit fig.6.9(c) kan waargeneem word dat die ruis 'n beduidende afwyking in die ontvangde simboolfases veroorsaak, sodat oorvleueling met aangrensende fasehoeke plaasvind en uitkenning van unieke fases bemoeilik word.

6.4.2 Bepaling van fouttempo

Om die prestasievermoë van die ontwikkelde kommunikasiestelsel in die teenwoordigheid van ruis te bepaal is die fouttempo van die stelsel gemeet. Die fouttempo is die aantal foutiewe simbole of bisse wat by die ontvanger ontvang word t.o.v. die totale aantal simbole of bisse wat oorspronklik vanaf die sender gestuur is.

Met die stelsel soos in fig.6.1 getoon, is die volgende prosedure gevolg in die neem van lesings ter bepaling van die simbool- en bisfouttempo.

1. Die ruisvlak word afgestel na nul en die sender geïnisieer. Die w.g.k. spanningswaarde van die gemoduleerde sein word dan gemeet by die inset van die ontvanger.
2. Die sender word afgeskakel en die ruis word verstel tot 'n sekere w.g.k. spanningswaarde.
3. Met die ruis op die betrokke waarde word die totale stelsel geïnisieer en die gestuurde sein, besoedel met ruis, word by die ontvanger ontvang.
4. Nadat die ontvanger 40960 bisse ontvang het word dit uitgelees na die gasheerrekenaar waar dit vergelyk word met die oorspronklike datalêer wat in die sender gelaai is voor die aanvang van transmissie.
5. Die vergelyking en die bepaling van die aantal bisfoute tot die aantal bisse gestuur, word deur 'n Pascal program verrig (KONTROLE.pas). Hierdie program bepaal die bisfouttempo van die stelsel met trellis enkodering en Viterbi dekodering. Die aantal foute, asook waar dit in die transmissieproses voorgekom het, en die bisfouttempo (BER) word in 'n kontrole lêer gestoor. Die program is aangeheg as bylae B.6.
6. Ten einde statisties meer betroubare resultate te kry is hierdie proses (punte 1 tot 5) 10 keer vir elke seinruisverhouding herhaal.

Stap nr. 4 in bogenoemde proses is effens gewysig in die geval van die 8DPSK ontvanger sonder Viterbi dekodering (kontrole ontvanger). Aangesien die twee ontvangers (toets en kontrole) van dieselfde sender gebruik maak, ontvang die kontrole ontvanger trellis gekodeerde simbole. Om die databisse uit die 3-bis kodewoorde - simbole - te verkry sou 'n addisionele dekodeerder benodig word. Daarom is daar besluit om die ontvangde simbole te vergelyk met die simbole soos deur die trellis enkodeerder gelewer vir modulasie by die sender. Dus word die simboolfouttempo (SER) t.o.v. verskillende

seinruisverhoudings by die kontrole ontvanger gemeet. Hierdie bepaling van simboolfoute word, soortgelyk aan die bepaling van die bisfouttempo, deur 'n Pascal program gedoen.

6.4.2.1 Resultate : 8DPSK en Trelliskode ontvanger

Tabel 6.1 lys die gemete resultate van die toets ontvanger - 8DPSK met trellis enkodering en Viterbi dekodering - en die kontrole ontvanger - ongekodeerde 8DPSK. Die w.g.k. spanningswaarde van die gemoduleerde sein (Vs) is deurentyd gelyk aan 1,2V.

Tabel 6.1 : Gemete resultate

RUIS (Vn) (Vwgk)	KONTROLE (SER)	TOETS (BER)
0.15	0.000007	
0.17	0.000153	
0.19	0.000516	
0.21	0.00106	
0.24	0.00328	
0.27	0.00558	0.000005
0.30	0.0120	0.000090
0.34	0.0258	0.000481
0.38	0.0420	0.00112
0.43	0.0732	0.00312

Vervolgens word die twee seine hoofsaaklik t.o.v. hul energie bespreek. Die gemoduleerdesein-energie (Ps) en die ruis-energie (Pn) word as die kwadraat van hul w.g.k. spannings (Vs & Vn) geneem. Die weerstand van die ontvanger (R) word as genormaliseer beskou en word daarom weggelaat in die energie berekening.

Die ruis w.g.k. spanning (Vn) moet verkieslik na die banddeurlaatfilter van die ontvanger, net voor die demodulator, gemeet word *[2, p. 451]*. A.g.v. die praktiese probleme daaraan verbonde in hierdie projek is die ruis w.g.k. spanning voor die banddeurlaatfilter van die DSP56001 ontvanger gemeet. 'n Fout word hierdeur veroorsaak deurdat minder ruisenergie as wat gemeet word, die demodulator bereik. Dit is a.g.v. die ruisgenerator-onderdeurlaatfilter (fn) wat nie dieselfde bandwydte as die banddeurlaatfilter ($fdsp$) van die DSP ontvanger besit nie. Die fout is voor gekorrigeer deur die gemete ruis-energie waardes (Pn) aan te pas met 'n verhouding Eq (sien uitdrukking 6.1). Eq is as volg bepaal,

$$Eq = \frac{eq_{fdsp}}{eq_{fn}} \tag{6.1}$$

waar die ekwivalente waardes $(eq_{fdsp} \& eq_{fn})$ van die twee filters verkry is deur hul onderskeie oordragkarakteristieke $(H(f))$ te integreer oor 'n gelyke bandwydte - vergelyking (6.2). Die $H(f)$ waardes vir die onderdeurlaatfilter (fn) is vanaf eksperimentele waardes verkry en die banddeurlaatfilter $(fdsp)$ se waardes is afgelei vanaf die spesifikasies soos deur Texas Instruments verskaf *[24, p. 29]*.

$$eq_f = \int_0^{\infty} [H(f)]^2 \, df \tag{6.2}$$

Die volgende waarde is vir Eq verkry,

$$Eq = \frac{eq_{fdsp}}{eq_{fN}} = \frac{0.479}{0.528} = 0.907$$

Dit lewer 'n gekorrigeerde seinruis-energiewaarde van :

$$E/N = 10Log\left[\frac{(V_s)^2}{0.907 \times (V_n)^2}\right] \tag{6.3}$$

Om die aanpassing/korreksie van die ruisenergie te kontroleer is 'n gemoduleerde sein en 'n tipiese ruissein afsonderlik deur die ontvanger bemonster. Die monsterwaardes (nà die ontvanger insetfilter) is m.b.v. MCAD simulasies vergelyk met die aangepaste berekende waardes en goeie korrelasies is verkry.

Met die ruisenergie (Pn) aangepas soos in (6.3), is die seinruis-energieverhouding (E/No) as volg bereken,

$$E/No = 10Log\left(\frac{Ps}{Pn} \cdot BW \cdot T\right) \tag{6.4}$$

waar BW (2040Hz) die bandwydte van die stelsel is met T die simboolperiode.

6.4.2.2 Simboolfouttempo van die 8DPSK ontvanger

Ter evaluering van die 8DPSK ontvanger word die simboolfouttempo vergelyk met 'n teoretiese model van sodanige ontvanger wat deur die volgende formule voorgestel word,

$$SER \approx erfc\left(\sqrt{\frac{2E}{No}}\sin\left(\frac{\pi}{2M}\right)\right) \qquad M \geq 4 \qquad (6.5)$$

waar $erfc$ die komplementêre foutfunksie is en M die aantal vlakke in die DPSK modulasieproses voorstel *[10, p. 317]*. Fig.6.10 toon die grafiek van die gemete waardes (met aangepaste ruiswaarde) van die 8DPSK ontvanger teenoor die teoretiese waardes soos uit formule 6.5 bepaal.

Uit fig.6.10 kan waargeneem word dat die 8DPSK demodulator effens swakker presteer (3.5dB by 1×10^{-5} SER tot 2dB by 7×10^{-2} SER) as die teoretiese model. Die degradasie kan hoofsaaklik toegeskryf word aan die onperfekte sinchronisasie van die ontvanger t.o.v. die sender a.g.v. die nie-koherente aard van die stelsel.

6.4.2.3 Bisfouttempo van 8DPSK met en sonder Trellis enkodering en Viterbi dekodering

Om die kontrole ontvanger met die toets ontvanger te vergelyk moet die gemete prestasies van albei op dieselfde assestelsel voorgestel word. Vir hierdie doel is die twee ontvangers se resultate omgeskakel na bisfouttempo's teen verskillende bisruis-energieverhoudings (Eb/No). Op hierdie wyse kan verskillende stelsels op 'n gelyke basis vergelyk word. Die bisruis-energie waardes is as volg bereken,

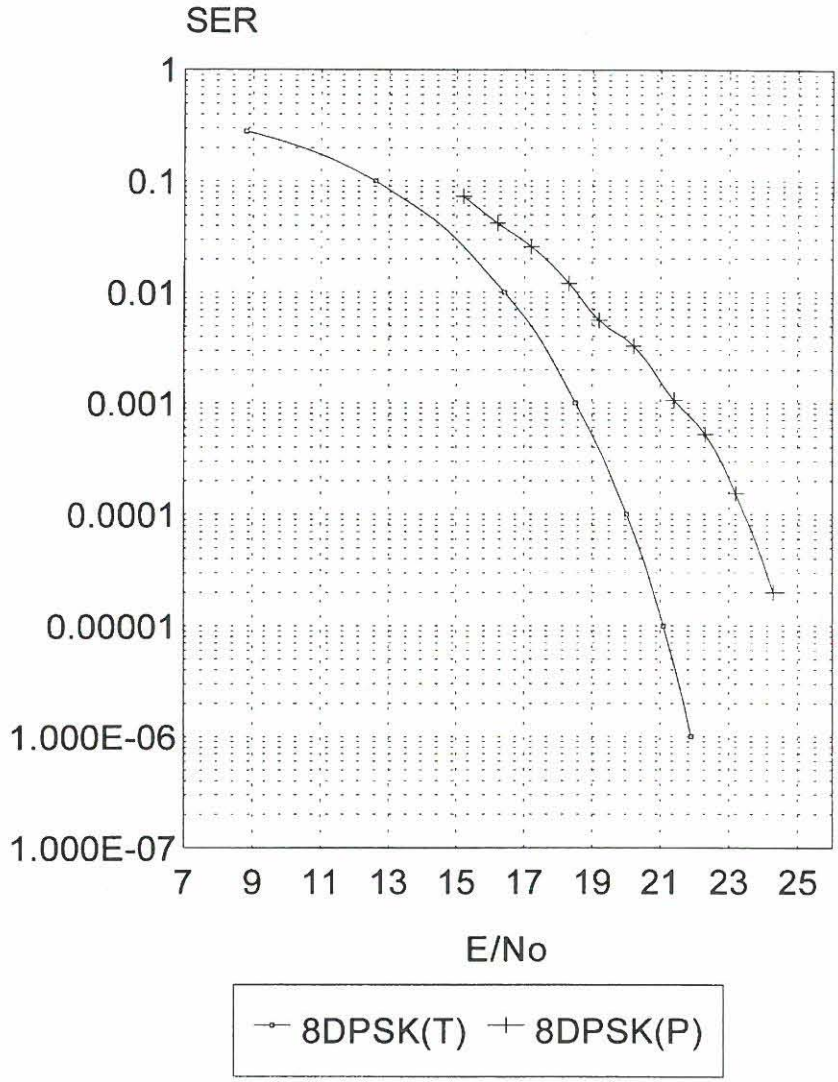


Fig. 6.10 : Simboolfouttempo van 8DPSK demodulator- prakties en teoreties (SER tot E/No)

$$Eb/No(toets) = \frac{E}{No} \times \frac{1}{2} \qquad (6.6)$$

$$Eb/No(kontrole) = \frac{E}{No} \times \frac{1}{3} \qquad (6.7)$$

Die toets ontvanger se E/No waarde word gehalveer omdat die koderingsproses veroorsaak dat slegs twee databisse per simbool oorgedra word. Die ongekodeerde 8DPSK lewer drie databisse met elke ontvangde simbool.

Om die simboolfouttempo (SER) van die kontrole stelsel om te skakel na bisfouttempo (BER), is van die volgende vergelyking gebruik gemaak,

$$BER = \frac{n_G}{\log_2 M} \times SER \qquad (6.8)$$

[2, p. 175].

M dui die aantal punte in die betrokke konstellasie aan. n_G is die gemiddelde aantal foutiewe bisse per foutiewe simbool as slegs die aangrensende gebiede in die konstellasie in ag geneem word. Vir die 8DPSK ontvanger in hierdie projek - wat gebruik maak van die trellis konstellasie in tabel 4.2 - is $n_G = 1.75$. Tabel 6.2 toon aan hoe n_G bepaal is.

Tabel 6.2 : Bepaling van die Gray-getal, n_G

Fase	Simbool	Moontlike bisfoute/simboolfout
0°	000	
		1
45°	001	
		2
90°	010	
		1
135°	011	
		3
180°	100	
		1
225°	101	
		2
270°	110	
		1
315°	111	
		3
0°	000	
Totaal :		**14**
Gemiddeld :		**1.75**

Nadat al bogenoemde aanpassings op die gemete waardes in tabel 6.1 gedoen is, is die resultate soos in tabel 6.3 verkry. Die ooreenkomstige grafiek word in fig.6.11 getoon. Bylae E toon 'n tabulering wat bostaande omskakelingsproses toelig.

Tabel 6.3 : Bisfouttempo's van die kontrole en toets ontvanger

Eb/No (dB)	Kontrole (8DPSK) (BER)	Toets (Trellis+Viterbi) (BER)
9.97	0.0427	
11.05	0.0245	
12.01	0.0150	
12.16		0.00312
13.10	0.007	
13.28		0.00112
14.01	0.00325	
14.19		0.000481
15.04	0.00191	
15.29		0.00009
16.19	0.000618	
16.20		0.000005
17.07	0.000301	
18.03	0.00009	
12.2	0.000004	

Uit fig.6.11 kan waargeneem word dat die ontwikkelde 8DPSK stelsel met trellis enkodering en Viterbi dekodering 'n wins van 2.9dB tot 1.9dB teen bisfouttempo's van tussen 5×10^{-6} en 3×10^{-3} onderskeidelik lewer m.b.t. die kontrole ontvanger. Hierdie waardes korrespondeer redelik met die teoretiese limiet van 3.6dB wins vir 'n 8PSK trelliskode stelsel. Die swakker as teoreties verwagte prestasie van die 8DPSK trelliskode stelsel kan vermoedelik toegeskryf word aan die differensiële aard van DPSK wat 'n invloed het op die prestasievermoë van die trellis foutregstellingskode *[2, p. 222]*.

6.5 GEVOLGTREKKING

Uitgebreide wiskundige simulasies van die kommunikasiestelsel is vooraf gedoen. Dit het baie bygedra tot die verkryging van insig deur die navorser en dus tot die suksesvolle implementering van die stelsel.

Die 8DPSK sender met trellis enkodering is ontwikkel en geïmplementeer m.b.v. die SIG56 DSP kaart. Die beoogde datatempo (3200 bisse/sekonde) is in die sender sonder enige probleme bereik. So ook is die 8DPSK gewysigde filterdetektor in die DSP56001 ontvanger geïmplementeer en het dit geen beperkinge op die oorspronklike datatempo geplaas nie. T.o.v. prestasievermoë het die gewysigde filterdetektor goed gepresteer (binne 2 tot 3dB vanaf die teoretiese waardes).

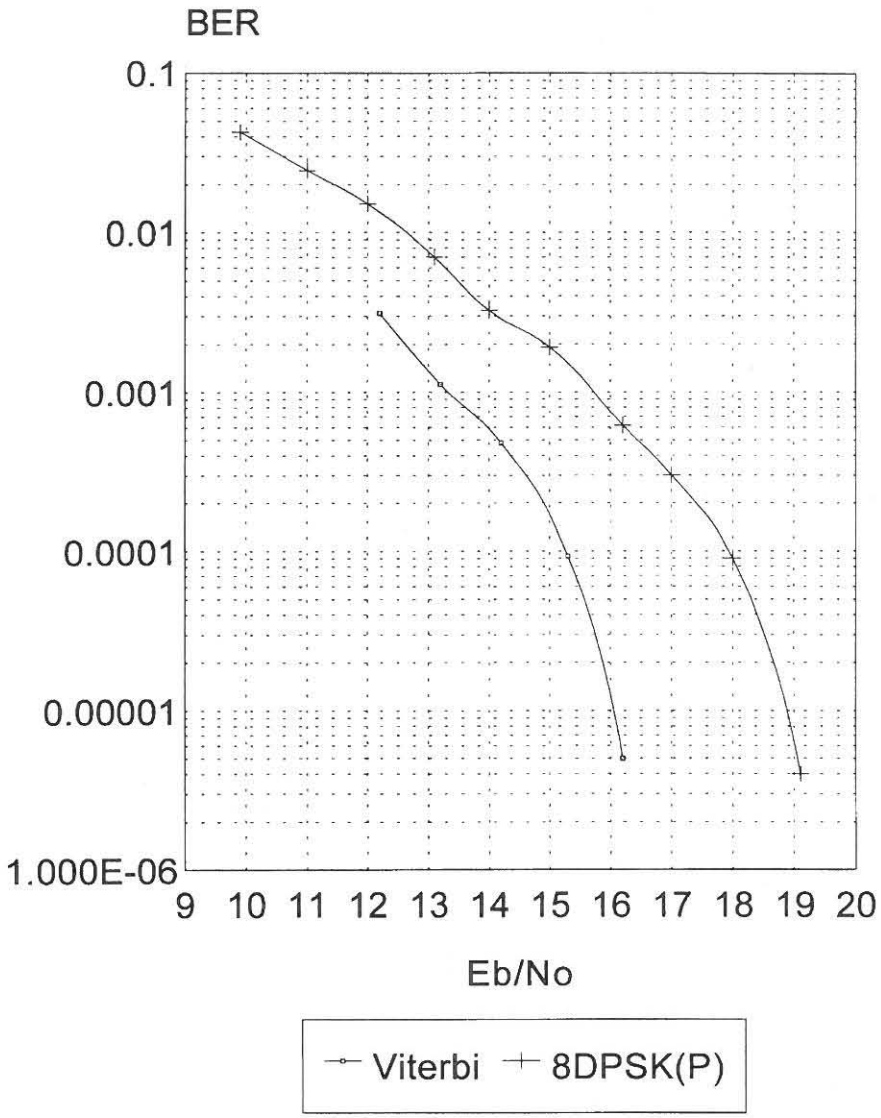


Fig. 6.11 : Bisfouttempo van die toets en kontrole ontvangers (BER tot Eb/No)

Met die implementering van Viterbi dekodering in die ontvanger het daar 'n probleem ontstaan t.o.v. prosesseringstyd teen die verlangde datatempo. Alhoewel die sinchronisasiedetektor uitgeskakel is (sien punt 5.4.2) en die sagteware (saamsteltaalprogram - ontv.asm) sover as moontlik ge-optimaliseer is (sien punt 5.4.2.2) kon die DSP56001 ontvanger nie die program vinnig genoeg prosesseer om 'n datatempo van 3200 bisse per sekonde te realiseer nie. Die datatempo van die hele stelsel is verlaag tot 1066 bisse/sekonde wat 'n bereikbare prosesseringstempo vir die DSP56001 ontvanger met Viterbi dekodering was. Die tekort aan prosesseringstyd vir die Viterbi dekoderingsproses kan moontlik aan die hand van die volgende punte aangespreek word,

- Vinniger prosesseerder.
- Parallel dekodering vir vinniger verwerking van die Viterbi algoritme *[13 p. 345]*.
- Vinniger Viterbi algoritme (korter dekoderingslengte (< $5M$) of minder state).
- Hardeware implementering.
- Die relatiewe meriete van Rant dekodering kan opnuut oorweeg word.

Die gebruik van verskil-in-korrelasie waardes i.p.v. Euklidiese afstande in die Viterbi dekoderingsproses (sien punt 4.3.1.5) het bevredigend gefunksioneer, aangesien dit steeds 'n wins van +/- 2dB t.o.v. 8DPSK sonder Viterbi dekodering gelewer het. Dit is 'n aanvaarbare degradasie in vergelyking met die teoretiese maksimum waarde van 3.6dB. Na aanleiding van fig.4.24 word aanvaar dat die gebruik van Euklidiese afstande nie 'n beduidende verbetering in resultate sou lewer nie.

T.o.v. die ontwikkelde sinchronisasiedetektor in die ontvanger is waargeneem dat sinchronisasie-sluit probleme begin intree het vanaf 'n seinruis-energieverhouding van 18.3dB (E/No - soos in fig.6.10). Dit het geleidelik toegeneem tot by 'n seinruis-energieverhouding van 15.2dB. Teen 'n seinruis-energieverhouding van minder as 15dB het die sinchronisasiedetektor nie meer op die leerfasesein gesluit nie - sodat geen betroubare resultate verkry kon word nie. Gebruik van 'n geskikte banddeurlaatfilter in die sinchronisasiedetektor sal waarskynlik hierdie situasie kan verbeter ten koste van addisionele prosesseringstyd.

Die gebruik van 'n digitale struktuur in die ontwikkeling van 'n datakommunikasiestelsel soos beskryf word aanbeveel a.g.v. die stabiliteit en aanpasbaarheid daarvan (sien punt 3.1).

Die beoogde stelsel is dus met welslae ontwikkel en geëvalueer - afgesien daarvan dat die verlangde datatempo nie gehandhaaf kon word nie.

Die relatiewe meriete van die gebruik van PSK teenoor DPSK in 'n toepassing soos nagevors behoort verder ondersoek te word.

7. SAMEVATTING

Die implementering van 'n trelliskode datakommunikasiestelsel wat gebruik maak van 8DPSK en DSP tegnologie is met sukses ondersoek. Onderstaande is 'n kortlikse samevatting van die voorafgaande verslag.

Hoofstuk twee behandel die teoretiese agtergrond vir digitale kommunikasie in 'n banddeurlaat kanaal. Daar is in die algemeen na die verskillende modulasie tegnieke en hul prestasievermoëns verwys, maar meer spesifiek na differensiële fase-skuifsleuteling. Verskillende foutregstellingskodes en moontlike dekoderingstegnieke is bespreek met die klem op trellis enkodering en Viterbi dekodering.

Digitale seinprosessering word in hoofstuk 3 bespreek t.o.v. die voordele en nadele daaraan verbonde met spesifieke verwysing na verskeie aspekte wat gepaard gaan met die digitalisering van 'n sein. Die digitale argitektuur gebruik in hierdie projek nl. die SIG56 kaart met die DSP56001 mikroprosesseerder en TLC32044 analoog koppelvlakkring aanboord, se uitleg en basiese opstelling en werking is gegee.

Hoofstuk 4 behandel die spesifieke koderings- en modulasietegnieke asook die ooreenkomstige demodulasie- en dekoderingstegnieke soos gebruik in hierdie projek. Verder is daar na die wiskundige simulasies wat gebruik is in die ontwerp en ontfouting van die projek verwys.

Die stapsgewyse implementering van die sender en ontvanger in sagteware is in hoofstuk 5 uiteengesit en verduidelik.

Hoofstuk 6 lê die eksperimentele werk se resultate voor. Die toerusting gebruik, asook die prosedure gevolg in die evaluering van die stelsel is verduidelik. Seinkonstellasiegrafieke van die gestuurde en ontvangde dataseine, asook die frekwensiespektrum van die gestuurde sein, is getoon. Die gemete simboolfouttempo en bisfouttempo van die 8DPSK demodulator alleen, en 8DPSK demodulator met trelliskode en Viterbi dekodering is onderskeidelik geplot. 'n Kortlikse beskrywing van die resultate is gegee.

Kundigheid t.o.v. veral die volgende aspekte is bekom tydens die uitvoering van die projek,

1. Trelliskodemodulasie en die gebruik van konvolusiekodes in die algemeen.
2. Gebruik van DSP in 'n persoonlike rekenaar omgewing. Die integrering van die sagteware wat op die persoonlike rekenaar en DSP56001 uitgevoer word was insiggewend.
3. Die implementering van die Viterbi dekoderings-algoritme in DSP.
4. Simboolsinchronisasie en die gebruik van 'n korrelasiedetektor.

BYLAES

A : SIMULASIES

B : SAGTEWARE

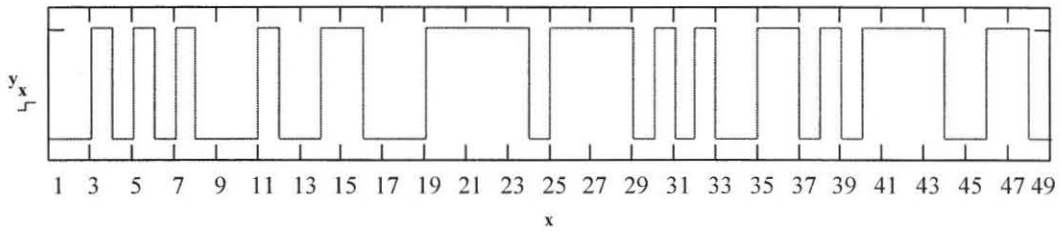
C : 8-STAAT TRELLIS

D : 2300Hz ONDERDEURLAATFILTER

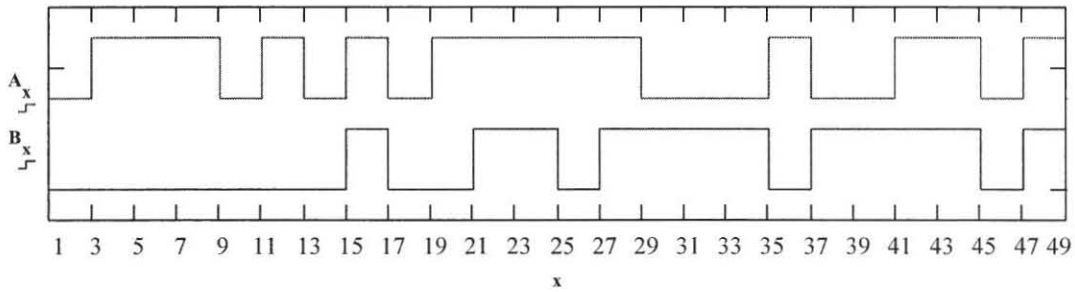
E : TABEL VAN RESULTATE

BYLAE A.1 : SIMULASIE VAN SENDER

Pseudo-ewekansige datagenerator : $x := 1..1200 \quad y_x := \Phi \ \text{rnd}\ 1 - 0.5$



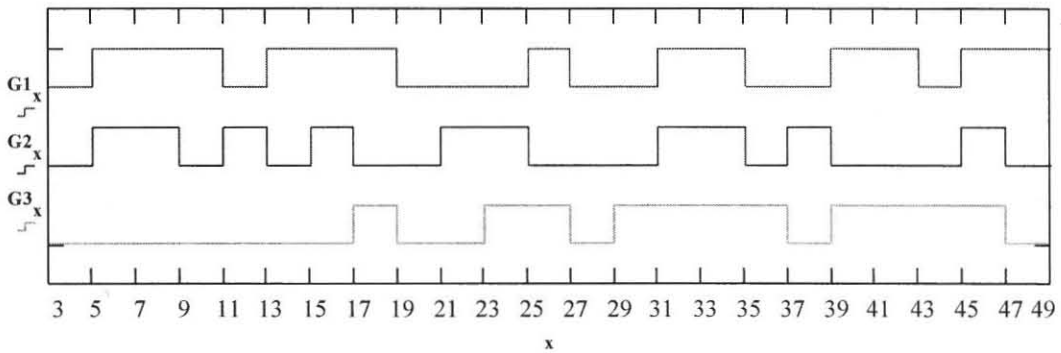
Opdeel in twee datastrome : $x := 1,3..1200 \quad A_x := y_x \quad B_x := y_{x-1}$



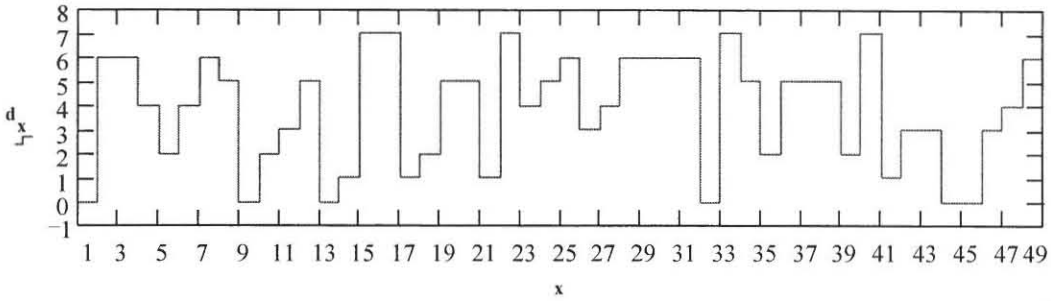
Trelliskode enkodeerder $x := 5,7..1200$

Skuifregisters : $R0_x := A_x \quad R2_x := A_{x-2} \quad R1_x := B_x \quad R3_x := B_{x-2} \quad R5_x := B_{x-4}$

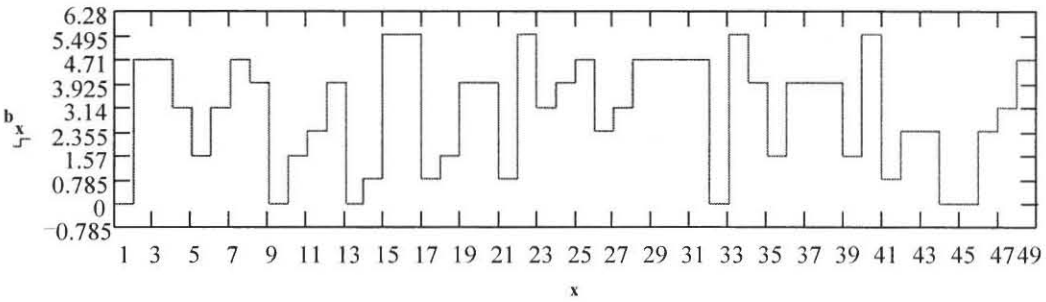
Kodegenerators : $G1_x := \text{xor}\ R2_x, R1_x \quad G2_x := \text{xor}\ R0_x, R5_x \quad G3_x := R3_x \quad x := 1,3..120$



Omskakeling : $d_x := G1_x \cdot 4 + G2_x \cdot 2 + G3_x$



Vlakke na radiale : $b_x := d_x \cdot \frac{\pi}{4}$



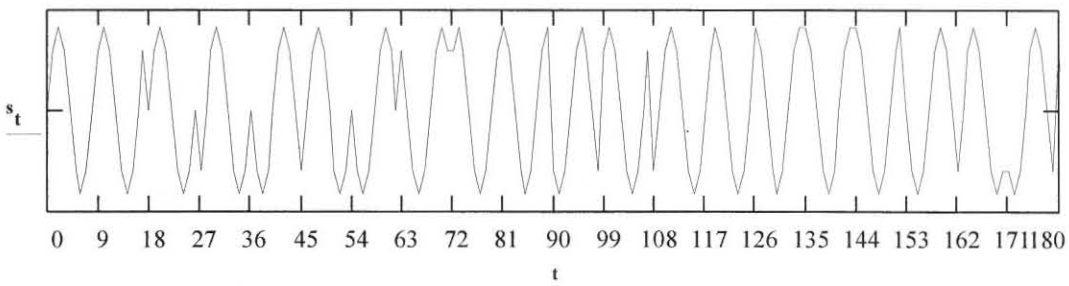
8DPSK Modulasie

Faseverandering: $c_x := b_x \quad c_x := c_x + c_{x-1}$

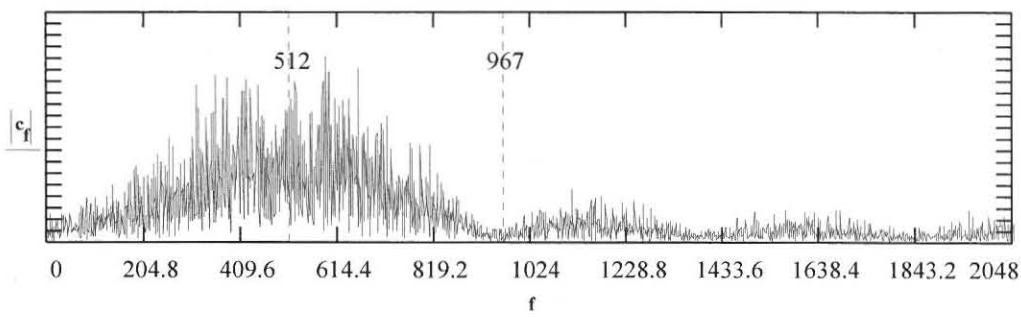
Genereer van 9 monsters per simbool : $m := 0,1..8 \quad n := 9,18..5391 \quad \phi_{n+m} := \frac{c_n}{9}$

$t := 0,1..4096$

8DPSK sein : $fc := 0.125 \quad w := 2 \cdot \pi \cdot fc \quad s_t := 1 \cdot \sin(w \cdot t + \phi_t) \quad \text{WRITEPRN}(RF\_sin_{dat}) := s_t$



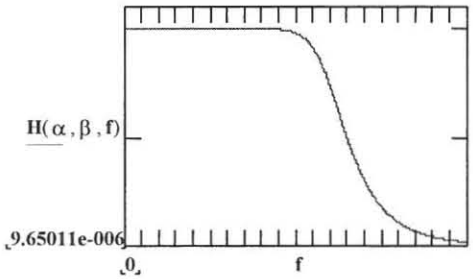
Frekwensiespektrum : $t := 0,1..4096 \quad c := \text{cfft}(s) \quad n := \text{last}(c) \quad n := 4096 \quad f := 0..n$



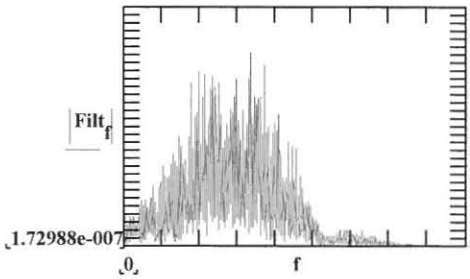
$n := 967 \quad d := \frac{1}{4096} \quad fo := n \cdot d \quad fb := fo \cdot 14400$ Afsny frekwensie : $fb = 3.4 \cdot 10^3$

Onderdeurlaatfilter : $\alpha := \frac{fo}{d} \quad \beta := 8 \quad H(\alpha, \beta, t) := \frac{1}{\sqrt{1 + \left(\frac{t}{\alpha}\right)^{2\beta}}} \quad Filt_f := H(\alpha, \beta, f) \cdot c_f$

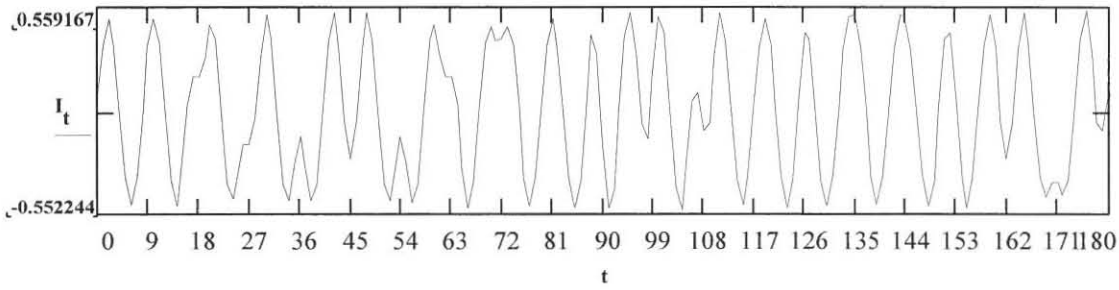
Filter karakteristiek



Sein deurgelaat

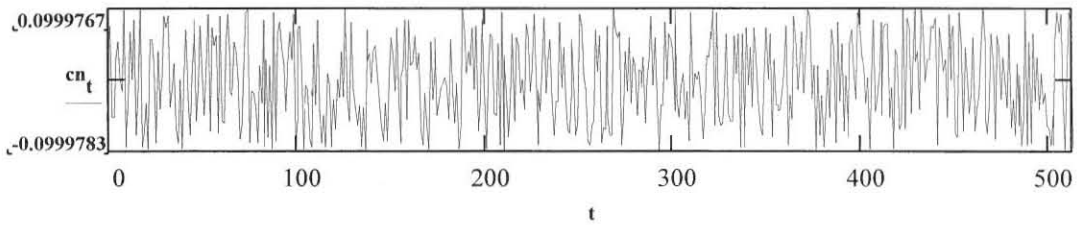


Rf uitsetsein : $t := 0, 1 .. 4096 \quad TX := icfft(Filt) \quad I_t := Re\ TX_t \quad WRITEPRN\ RF\_uit_{dat} := I_t$

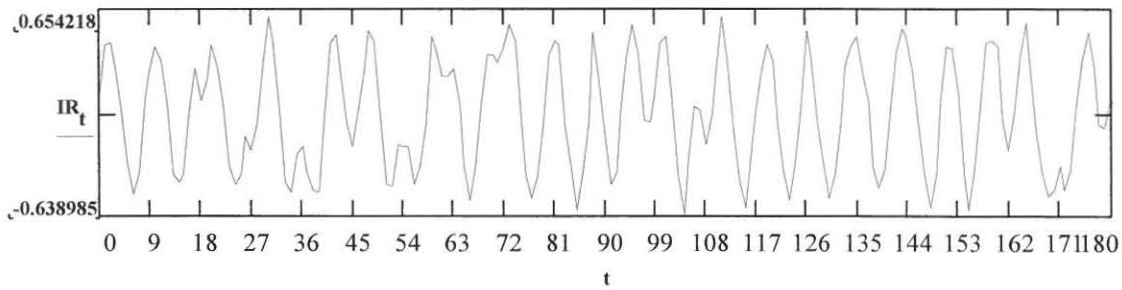


Invloed van kanaal

Ruis : $n := 0.2 \quad cn_t := rnd(1) - 0.5 \cdot n$



Ontvangde sein : $IR_t := I_t + cn_t \quad WRITEPRN\ RF\_ontv_{dat} := IR_t$



Sein konstellasie :

Sin en Cos verwysing : $fc := 0.125 \quad w := 2 \cdot \pi \cdot fc \quad \phi := 0$

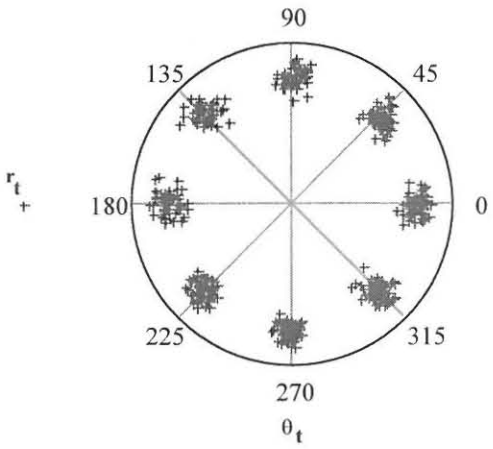
$s_t := 200 \cdot \sin(w \cdot t + \phi) \quad c_t := 200 \cdot \cos(w \cdot t + \phi)$

Vermenigvildig : $I\_SIN_t := IR_t \cdot s_t \quad I\_COS_t := IR_t \cdot c_t$

Gemiddeld oor 8 monsters : $t := 9, 18 .. 4085 \quad n := 0, 1 .. 7$

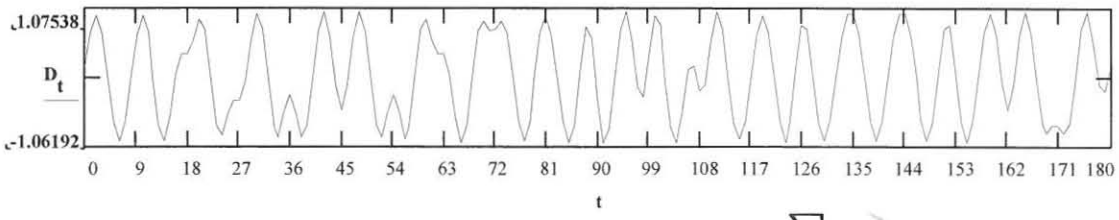
$y_t := \sum_n I\_SIN_{t+n} \quad x_t := \sum_n I\_COS_{t+n}$

Polêre vorm : $z_t := x_t + y_t \cdot j \quad r_t := |z_t| \quad \theta_t := \arg z_t$

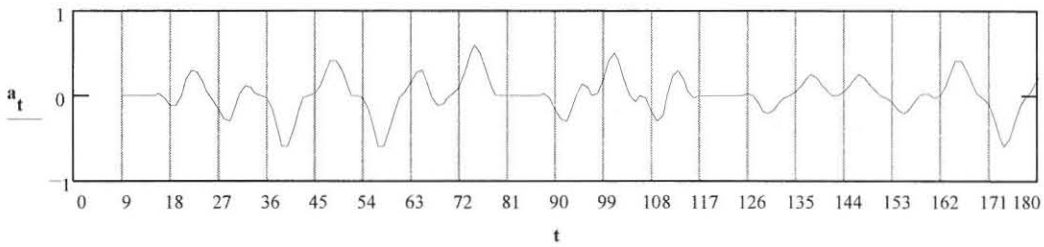


BYLAE A.2 : SIMULASIE VAN SINCHRONISASIEDETEKTOR

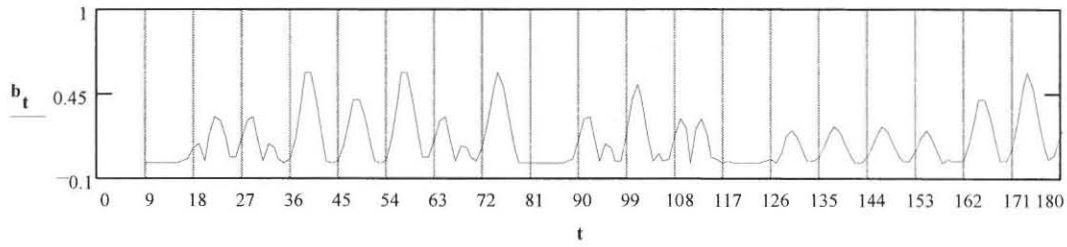
Sein vir sinchronisasie : $t := 0, 1 .. 4096$ $D := READPRN(RF\_uit_{dat})$ $k := 0.52$ $D := D \cdot \frac{1}{k}$



Gemmiddeld oor een siklus : $t := 9, 10 .. 4096$ $r := 0, 1 .. 7$ $a_t := \frac{\sum D_{t-r}}{8}$

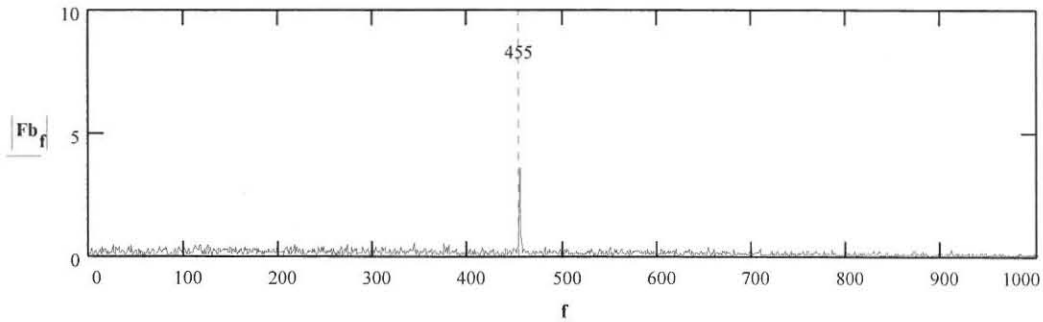


Absolute waarde : $b_t := |a_t|$

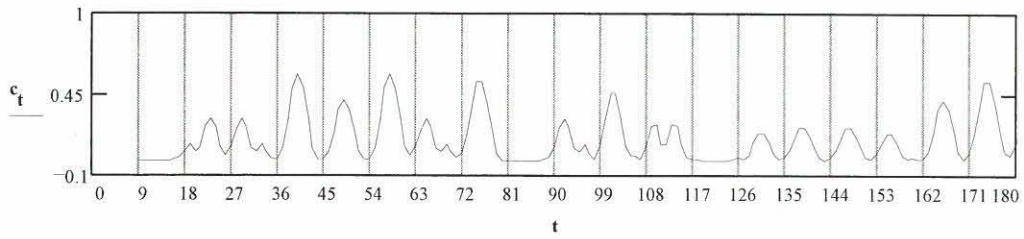


Frekwensiespektrum : $t := 0, 1 .. 4096$ $Fb := cfft(b)$ $n := 4096$ $f := 0 .. n$

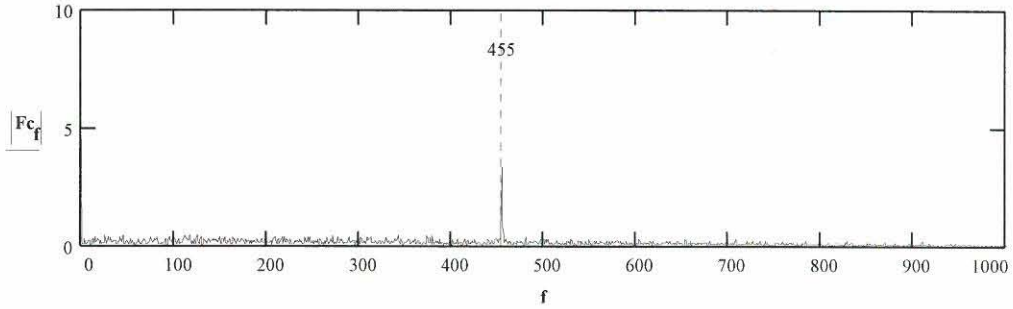
$n := 455$ $d := \frac{14400}{4096}$ $fo := n \cdot d$ $fo = 1.6 \cdot 10^3$



Gemmiddeld oor twee monsters : $t := 9, 10 .. 4096 \quad c_t := \frac{b_t + b_{t-1}}{2}$

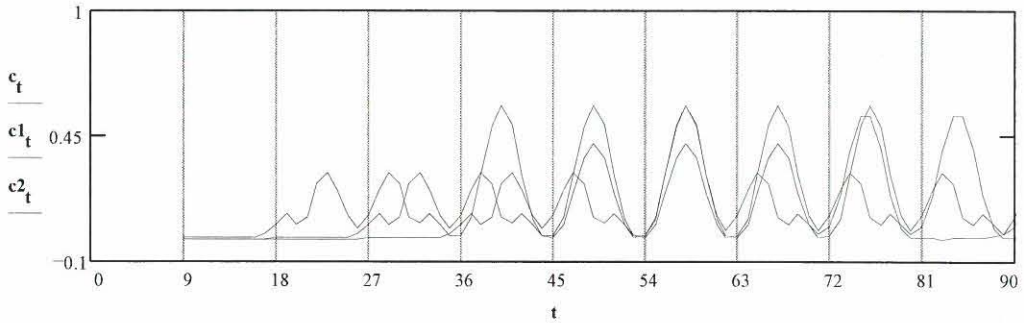


Frekwensiespektrum : $Fc := cfft(c)$

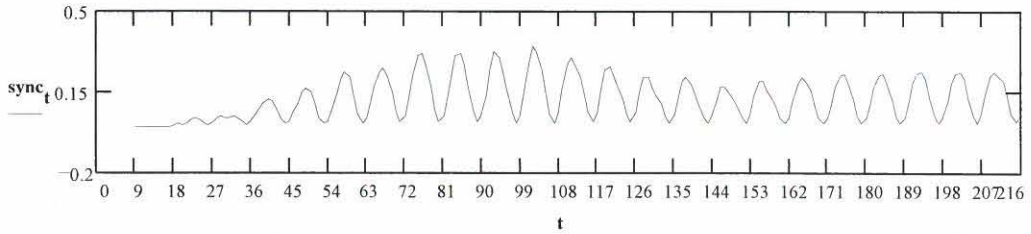


Vertraging van een simbool : $t := 9, 10 .. 4096 \quad c1_t := c_{t-9} \quad c2_t := c1_{t-9} \quad c3_t := c2_{t-9}$

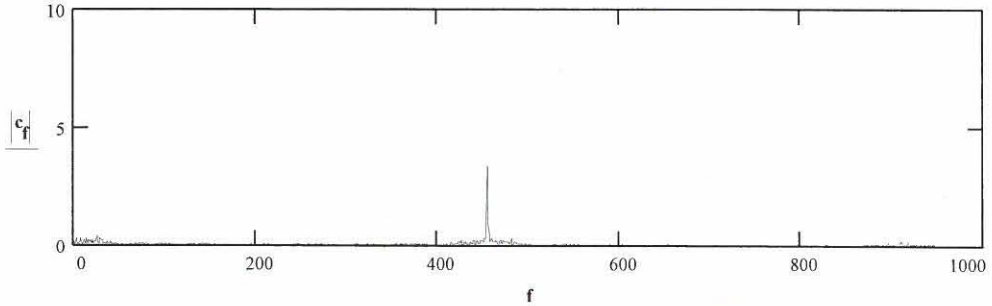
$c4_t := c3_{t-9} \quad c5_t := c4_{t-9} \quad c6_t := c5_{t-9} \quad c7_t := c6_{t-9}$



Gemiddeld oor n baud : $sync_t := \frac{|c_t| + |c1_t| + |c2_t| + |c3_t| + |c4_t| + |c5_t| + |c6_t| + |c7_t|}{8}$

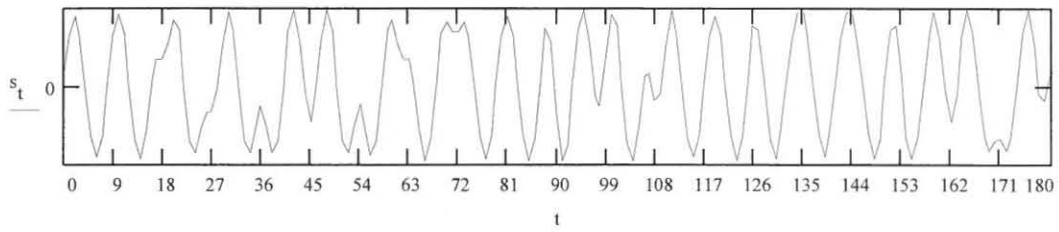


Frekwensiespektrum : $t := 0, 1 .. 4096 \quad c := cfft(sync)$



BYLAE A.3 : SIMULASIE VAN ONTVANGER

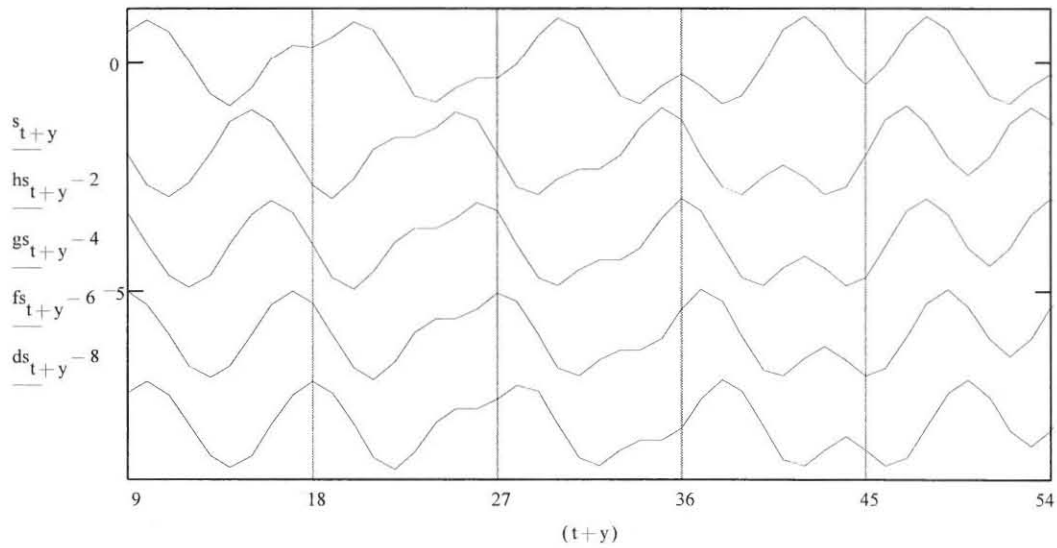
Ontvangde rf sein: $t := 0, 1 .. 512$ $s := READPRN(RF\_uit_{dat})$ $k := 0.52$ $s := s \cdot \frac{1}{k}$



8DPSK Demodulasie

Vertraging + Faseverskuiwing (0,45,90,135) $t := 9, 18 .. 512$ $y := 0, 1 .. 8$

$ds_{t+y} := s_{t+y-8}$ $fs_{t+y} := s_{t+y-7}$ $gs_{t+y} := s_{t+y-6}$ $hs_{t+y} := s_{t+y-5}$



Genereer van addisionele monsters vir vertraagde seine

$ds_{t+8} := ds_{t+4} \cdot -1$

$fs_{t+7} := fs_{t+3} \cdot -1$

$fs_{t+8} := fs_{t+4} \cdot -1$

$gs_{t+6} := gs_{t+2} \cdot -1$

$gs_{t+7} := gs_{t+3} \cdot -1$

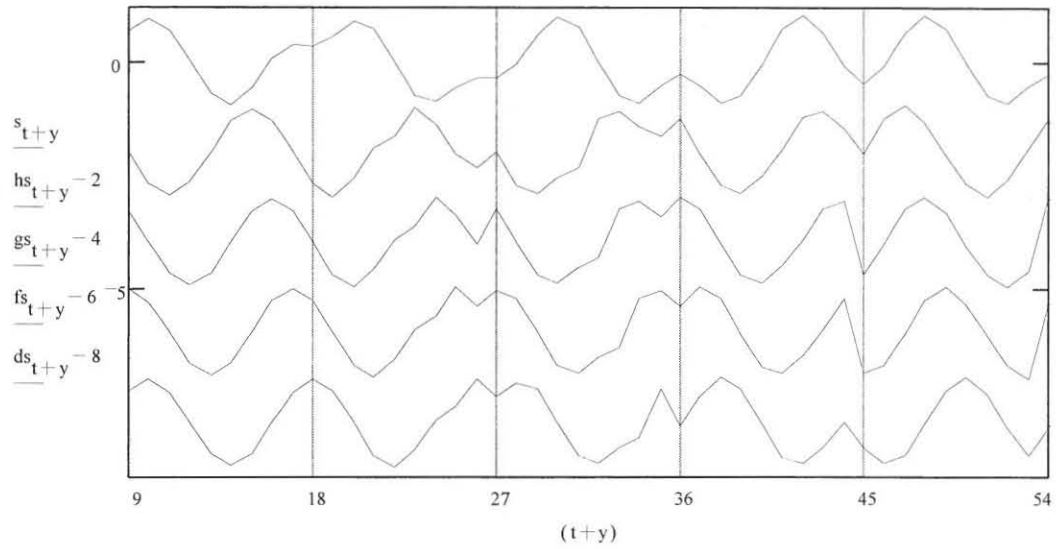
$gs_{t+8} := gs_{t+4} \cdot -1$

$hs_{t+5} := hs_{t+1} \cdot -1$

$hs_{t+6} := hs_{t+2} \cdot -1$

$hs_{t+7} := hs_{t+3} \cdot -1$

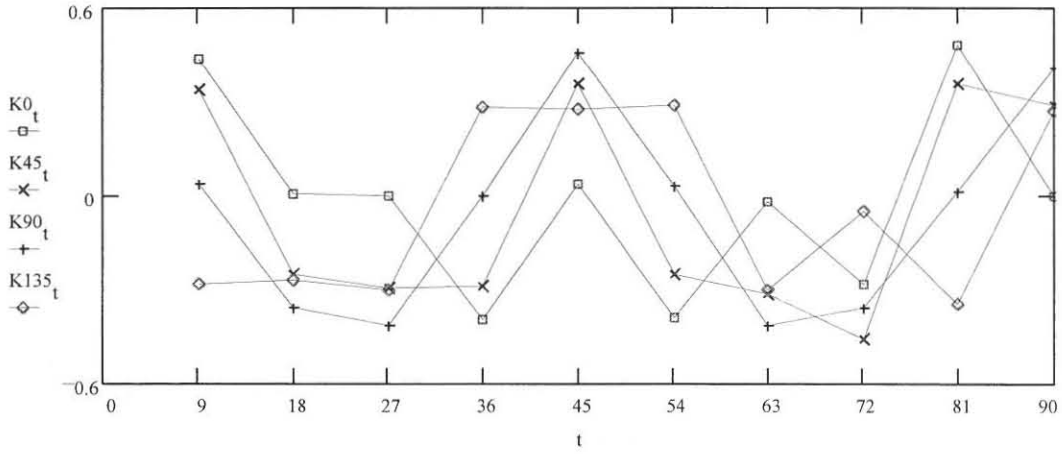
$hs_{t+8} := hs_{t+4} \cdot -1$



Korrelasie van sein oor 9 monsters :

$e1_{t+y} := s_{t+y} \cdot ds_{t+y}$ $e2_{t+y} := s_{t+y} \cdot fs_{t+y}$ $e3_{t+y} := s_{t+y} \cdot gs_{t+y}$ $e4_{t+y} := s_{t+y} \cdot hs_{t+y}$

$$K0_t := \frac{\sum e1_{t+y}}{9} \quad K45_t := \frac{\sum e2_{t+y}}{9} \quad K90_t := \frac{\sum e3_{t+y}}{9} \quad K135_t := \frac{\sum e4_{t+y}}{9}$$



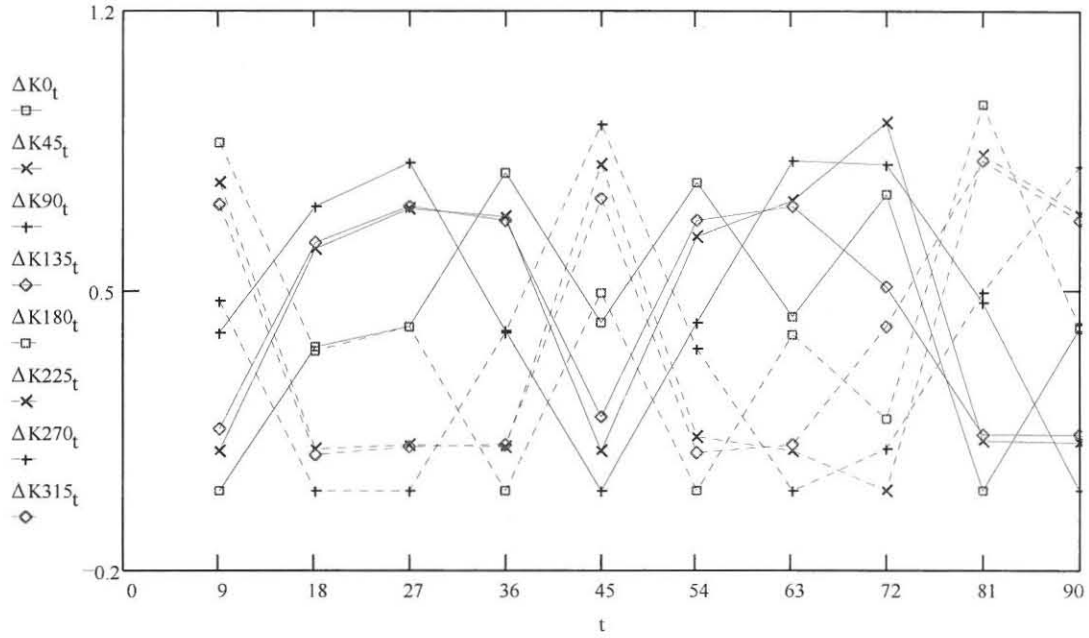
Kies die grootste : $\phi_t := 0$ $\Delta\phi_t := 0$

$\phi_t := \text{if } |K0_t| > |K45_t|, K0_t, K45_t$ $\phi_t := \text{if } |\phi_t| > |K90_t|, \phi_t, K90_t$ $\phi_t := \text{if } |\phi_t| > |K135_t|, \phi_t, K135_t$

Verskil in korrelasie waardes t.o.v. grootste

$\Delta K0_t := |\phi_t - |K0_t||$ $\Delta K180_t := |\phi_t - |K0_t| \cdot -1|$ $\Delta K45_t := |\phi_t - |K45_t||$ $\Delta K225_t := |\phi_t - |K45_t| \cdot -1|$

$\Delta K90_t := |\phi_t - |K90_t||$ $\Delta K270_t := |\phi_t - |K90_t| \cdot -1|$ $\Delta K135_t := |\phi_t - |K135_t||$ $\Delta K315_t := |\phi_t - |K135_t| \cdot -1|$



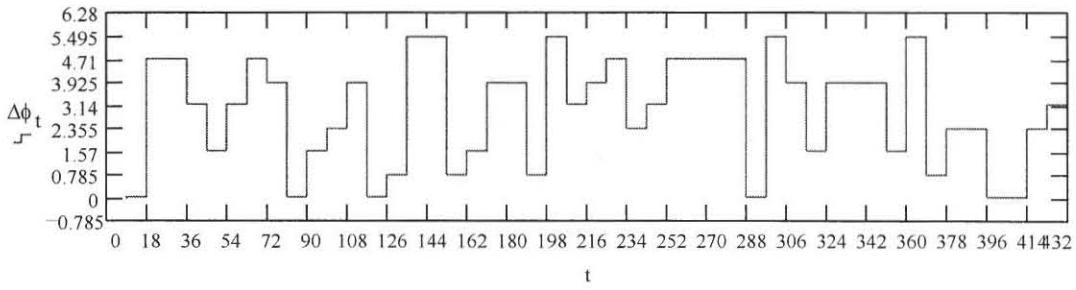
Besluit t.o.v. polariteit :

$$\Delta\phi_t := \text{if}\left(|\phi_t| = K0_t, 0, \Delta\phi_t\right) \qquad \Delta\phi_t := \text{if}\left(|\phi_t| = -K0_t, \pi, \Delta\phi_t\right) \qquad \Delta\phi_t := \text{if}\left(|\phi_t| = K45_t, \frac{\pi}{4}, \Delta\phi_t\right)$$

$$\Delta\phi_t := \text{if}\left(|\phi_t| = -K45_t, 5\cdot\frac{\pi}{4}, \Delta\phi_t\right) \qquad \Delta\phi_t := \text{if}\left(|\phi_t| = K90_t, 2\cdot\frac{\pi}{4}, \Delta\phi_t\right) \qquad \Delta\phi_t := \text{if}\left(|\phi_t| = -K90_t, 6\cdot\frac{\pi}{4}, \Delta\phi_t\right)$$

$$\Delta\phi_t := \text{if}\left(|\phi_t| = K135_t, 3\cdot\frac{\pi}{4}, \Delta\phi_t\right) \qquad \Delta\phi_t := \text{if}\left(|\phi_t| = -K135_t, 7\cdot\frac{\pi}{4}, \Delta\phi_t\right)$$

Differensiële verandering in fase :



BYLAE B.1 : HOOFPROGRAM TX_RX.PAS

```
program TX_RX;

uses
 crt,dsplib4;

var
   a,b                                :longint;
   k,l                                :integer;
   din,dout                           :text;
   baseaddress1,baseaddress2          :word;

begin

 assign(din,'rdm.dat');
 reset(din);
 assign(dout,'ontv.dat');
 rewrite(dout);

 writeln;
 writeln(' OPSTEL ');

 BaseAddress1:=$2b0;
 BaseAddress2:=$2c0;
 DSP_Reset(baseaddress1);
 DSP_Reset(baseaddress2);
 DSP_load(baseaddress1,'herstel');
 DSP_load(baseaddress2,'herstel');
 DSP_Go(BaseAddress1);
 DSP_Go(BaseAddress2);
 writeln;
 writeln('SENDER & ONTVANGER (X:$200-800 + Y:$200-800 Skoon)');
 Delay(200);

 DSP_Reset(baseaddress2);
 DSP_load(baseaddress2,'Data_in');
 DSP_Go(BaseAddress2);

 for k:=1 to 512 do
          begin
          readln(din,a);
          DSP_WriteFlag(baseaddress2,200);
          DSP_WriteInteger(baseaddress2,a);
          end;

 writeln;
 writeln('RDM.dat in SENDER gelaai');


 writeln;
```

```
  writeln('                TX --------- RX  ');

  DSP_Reset(baseaddress1);
  DSP_Reset(baseaddress2);
  DSP_load(baseaddress1,'ontv');
  Delay(200);
  DSP_load(baseaddress2,'send');
  Delay(200);

  DSP_Go(BaseAddress2);
  DSP_Go(BaseAddress1);

  Sound(800);
  Delay(25);
  Nosound;
  writeln('begin ');
  writeln(' ');
      for l:= 1 to 10 do
      begin
      delay(2000);
      write(' * ');
      delay(2000);
      a:=10 - l;
      write(a);
      end;

  for l:=1 to 2560 do
            begin
            DSP_ReadFlag(baseaddress1,200);
            DSP_ReadLongInt(baseaddress1,b);
            writeln(dout,b);
            end;

  close(din);
  close(dout);

  Sound(800);
  Delay(25);
  Nosound;
  writeln;
  writeln;
  writeln('2560 Integers ontvang ');

  DSP_Reset(baseaddress1);
  DSP_Reset(baseaddress2);

end.
```

Central University of Technology, Free State

BYLAE B.2 : PROGRAM HERSTEL.ASM

```
;*********************************************************************
;
;                    *** HERSTEL ***
;*********************************************************************
;
        opt     fc,mu,s,w,mex
begin   equ     $0080
        org     p:$0000
        jmp     Hoof
        clr     a
        move    a,x:$fffe
        org     p:begin

Hoof

;Maak registers skoon            X:200-7ff + Y:200-7ff
;--------------------------

        clr     a
        move    #>$200,r7
        .LOOP   #$5ff
        move    a,x:(r7)
        move    a,y:(r7)
        move    (r7)+
        .ENDL
end

;---------------------------------------------------------------------
```

BYLAE B.3 : PROGRAM DATA_IN.ASM

```
;*************************************************************************
;                          *** DATA_IN ***
;*************************************************************************

        opt     fc,mu,s,w,mex
begin   equ     $0080
        org     p:$0000
        jmp     Hoof
        clr     a
        move    a,x:$fffe
        org     p:begin

Hoof

Lees_Int

        move    #>$400,r1

        .LOOP   #512
in      btst    #0,x:$ffe9
        jcc     in
        move    x:$ffeb,a1
        move    a1,x:(r1)+

        .ENDL

        rts

end

;-------------------------------------------------------------------------
```

Central University of Technology, Free State

BYLAE B.4 : PROGRAM SEND.ASM

```
;*****************************************************************************
;                    ******* SENDER ******
;*****************************************************************************

;DSP 56001 konfigurasie

        opt     fc,mu,s,w,mex
begin   equ     $0080
        org     p:$0000
        jmp     Hoof
        org     begin
        ori     #%00000100,omr

Hoof
        jsr     OPSTEL
        jsr     IDENT

begin1
        jsr     LEES_INT

        .LOOP   #8
        jsr     ENKODEERDER
        jsr     UITSET
        .ENDL

        jmp     begin1

end

;*****************************************************************************
;*****************************************************************************
;-----------------------------------------------------------------------------
OPSTEL
;----------
;Sinus tabel
;---------------

        move    #$0,x0                  ;0
        move    x0,x:$200
        move    #$2c3000,x0             ;45
        move    x0,x:$201
        move    #$3e8000,x0             ;90
        move    x0,x:$202
        move    #$2c3000,x0             ;135
        move    x0,x:$203
        move    #$0,x0                  ;180
        move    x0,x:$204
        move    #-$2c3000,x0            ;225
```

```
        move    x0,x:$205
        move    #-$3e8000,x0            ;270
        move    x0,x:$206
        move    #-$2c3000,x0            ;315
        move    x0,x:$207


;Poort_C Opstel
;--------------------
        clr     b
        move    b1,x:$FFEF

; stel beheer registers
        move    #$4000,a
        move    a,x:$FFEC
        move    #$3A00,a
        move    a,x:$FFED
        movep   #$FFFF1E,y:$FFE0        ;AIC meester klokfrekwensie opstel

; stel poort C registers
        move    #$01FF,a
        move    a,x:$FFE1               ;beheer register
        move    #$010E,a
        move    a,x:$FFE3               ;data rigting register
        move    #$0000,a
        move    a,x:$FFFF               ;ontmagtig onderbrekings

;Opstel van Analoog Koppelvlak
;-----------------------------------------
        clr     a
        move    a,x:$ffef
        clr     b
        move    #$ffffff,b

; Stel van TA & RA registers

        move    #$1e3c00,a

pri1    btst    #6,x:$ffee
        jcc     pri1
        move    b,x:$ffef
sec1    btst    #6,x:$ffee
        jcc     sec1
        move    a,x:$ffef
        clr     a

; Stel van TB & RB registers

        move    #$489200,a              ;4800 monsters per sekonde

pri2    btst    #6,x:$ffee
        jcc     pri2
        move    b,x:$ffef
sec2    btst    #6,x:$ffee
        jcc     sec2
        move    a,x:$ffef
        clr     a
```

```
; Stel van Beheer register

        move    #$00E700,a

pri3    btst    #6,x:$ffee
        jcc     pri3
        move    b,x:$ffef
sec3    btst    #6,x:$ffee
        jcc     sec3
        move    a,x:$ffef
        clr     b

;16-Bis datawoord leeswyser

        move    #>$400,r1
        move    #>511,m1
        move    #>1,n1

;Sinus tabel wyser

        move    #>1,n2
        move    #>$7,m2
        move    #>$200,r2

; Herstel enkodeerder skuifregisters

        move    #$0,x1
        move    x1,y:$211
        move    x1,y:$212
        move    x1,y:$213
        move    x1,y:$214
        move    x1,y:$215

        rts

;-------------------------------------------------------------------------------
IDENT
;-------
        move    #>$4,y0                 ;180's uit
        move    y0,y:$219
        .LOOP   #200
        jsr     UITSET
        .ENDL

        move    #>$0,y0                 ;0's uit
        move    y0,y:$219
        .LOOP   #10
        jsr     UITSET
        .ENDL

        rts

;-------------------------------------------------------------------------------
LEES_INT
;-------------

        clr     a
        nop
```

```
        move    x:(r1)+,a1
        nop
        move    a1,y:$200               ;datawoord register
        rts

;-------------------------------------------------------------------------------
ENKODERING
;------------------

;Skuif bispaar uit datawoord

        clr     a
        move    y:$200,a1               ;lees datawoord

        lsl     a                       ;bis1 na carry
        move    a1,b1                   ;stoor geskuifde woord
        clr     a
        rol     a                       ;carry na a1
        move    a1,y:$201               ;bis1 uitset

        move    b1,a1                   ;laai geskuifde woord
        lsl     a                       ;bis2 na carry
        move    a1,b1                   ;stoor geskuifde woord
        clr     a
        rol     a                       ;carry na a1
        move    a1,y:$202               ;bis2 uitset

        move    b1,y:$200               ;stoor geskuifde datawoord

;Skuif twee bisse in skuifregisters

        move    y:$212,y1               ;S1 na
        move    y1,y:$214               ;S3
        move    y:$201,y1               ;bis1 na
        move    y1,y:$212               ;S1

        move    y:$213,y1               ;S4 na
        move    y1,y:$215               ;S6
        move    y:$211,y1               ;S2 na
        move    y1,y:$213               ;S4
        move    y:$202,y1               ;bis2 na
        move    y1,y:$211               ;S2

;Enkodering en omskakeling na 8-vlak waarde

        clr     b
        move    b1,y:$219               ;herstel 8-vlak register

        move    y:$214,a
        move    y:$211,y1
        eor     y1,a                    ;Kodegenerator G1
        jclr    #$0,a1,Som2             ;toets vir meesbeduidende bis
        move    #>$4,b1
        move    b1,y:$219               ;vermeerder met 4

Som2
        move    y:$215,a
        move    y:$212,y1
```

```
        eor     y1,a                    ;Kodegenerator G2
        jclr    #$0,a1,Som1             ;toets vir 2^1 bis
        move    #>$2,x0
        move    y:$219,b1
        add     x0,b                    ;vermeerder met 4
        move    b1,y:$219

Som1
        move    y:$213,x0               ;Kodegenerator G3
        move    y:$219,b1
        add     x0,b                    ;vermeerder met minsbeduidende bis
        move    b1,y:$219               ;Uitset 8-vlak waarde

        rts

;-------------------------------------------------------------------------------
UITSET
;---------

;Differensiële faseverandering

        move    y:$219,n2               ;faseverandering
        nop
        lua     (r2+n2)r2               ;Differensiële verandering
        move    #>1,n2                  ;herstel n5 na 45°)

;Genereer sein vanaf sinus tabel en lees uit

        .LOOP   #9
idle1   btst    #6,x:$ffee
        jcc     idle1
        move    x:(r5)+n5,b             ;Gemoduleerde simbool uit
        move    #$fffc00,x0             ;Masker
        and     x0,b
        movep   b,x:$ffef

        .ENDL

        rts
;-------------------------------------------------------------------------------
```

BYLAE B.5 : PROGRAM ONTV.ASM

```
;        ******************************************************
;        ***********  ONTVANGER 4800 Hz  *************
;        ******************************************************


;DSP Konfigurasie
        opt     fc,mu,s,w,mex
begin   equ     $0080
        org     p:$0000
        jmp     HOOF
        org     p:begin
        ori     #%00000100,omr

;*****************************************************************************
;                       HOOF PROGRAM
;*****************************************************************************
HOOF
        jsr     OPSTEL

TOETS                                           ;TOETS vir 100 baud sync
        move    x:$26d,a
        move    #>100,x0
        .IF     a <EQ> x0

;-----------------------------------------------------------------------------
DEMODULATOR
;----------------------
        jsr     LEES_MONSTER
        move    y:$200,a
        move    a,y:(r4)+

        .IF     r4 <EQ> #>$259                  ;TOETS of Baud vol is ?
        jsr     AFSTEL
        jsr     FILTER_DET_DEMOD
        move    #>$250,r4                       ;herstel baud beginpunt


        jsr     LEES_MONSTER                    ;MONSTER 1 van volgende baud
        move    y:$200,a
        move    a,y:(r4)+
        jsr     PAS_BAUD


        jsr     FASE_TOETS                      ;TOETS vir 10 baud 180 en 0 grade
        move    y:$20b,a                        ;opeenvolgend.
        .IF     a <EQ> #>1
```

```
;-------------------------------------------------------------------------------
VITERBI
;------------
        jsr     LEES_MONSTER            ;MONSTER 2 van volgende baud
        move    y:$200,a
        move    a,y:(r4)+
        jsr     NODUS_A
        jsr     NODUS_B

        jsr     LEES_MONSTER            ;MONSTER 3 van volgende baud
        move    y:$200,a
        move    a,y:(r4)+
        jsr     NODUS_C
        jsr     NODUS_D

        jsr     LEES_MONSTER            ;MONSTER 4 van volgende baud
        move    y:$200,a
        move    a,y:(r4)+
        jsr     NODUS_E
        jsr     NODUS_F

        jsr     LEES_MONSTER            ;MONSTER 5 van volgende baud
        move    y:$200,a
        move    a,y:(r4)+
        jsr     NODUS_G
        jsr     NODUS_H

        jsr     LEES_MONSTER            ;MONSTER 6 van volgende baud
        move    y:$200,a
        move    a,y:(r4)+
        jsr     SKRYF_NUWE_IN_OU
        jsr     KORTSTE_VAN_PAAIE

        jsr     LEES_MONSTER            ;MONSTER 7 van volgende baud
        move    y:$200,a
        move    a,y:(r4)+
        jsr     TERUG_SPOOR
        jsr     DATA_NA_INT
        move    (r7)+

;-------------------------------------------------------------------------------
        .ENDI

        .ENDI

        jmp     DEMODULATOR

        .ENDI
;-------------------------------------------------------------------------------
SYNC-DETEKTOR
;------------------------
        jsr     LEES_MONSTER
        jsr     DETEKSIE
        jsr     ZERO_DET_9
        .IF     r4 <EQ> #>$258          ;TOETS of Baud vol is ?
        jsr     TEL&HERSTEL

        .ENDI
```

```
;-------------------------------------------------------------------------------
        jmp     TOETS
end

;*******************************************************************************

;-------------------------------------------------------------------------------
OPSTEL
;----------
;Skaal Sin tabel
;-------------------
        move    #>$100,r1
        move    #>$400,n1
        .LOOP   #256
        move    y:(r1),a
        do      #14,dvd14
        asr     a
dvd14
        move    a,y:(r1+n1)
        move    (r1)+
        .ENDL

;POORT_C Opstel
;----------------------
        clr     b
        move    b1,x:$FFEF

; stel beheer registers
        move    #$4000,a
        move    a,x:$FFEC
        move    #$3A00,a
        move    a,x:$FFED
        movep   #$FFFF1E,y:$FFE0

; stel poort C registers
        move    #$01FF,a
        move    a,x:$FFE1               ;  beheer register
        move    #$010E,a
        move    a,x:$FFE3               ;  data rigting register
        move    #$0000,a
        move    a,x:$FFFF               ;  ontmagtig onderbrekings

;Opstel van AIC
;-------------------
        clr     a
        move    a,x:$ffef
        clr     b
        move    #$ffffff,b

; Stel van TA & RA registers

        move    #$1E3C00,a

pri1    btst    #6,x:$ffee
        jcc     pri1
        move    b,x:$ffef
sec1    btst    #6,x:$ffee
```

```
        jcc     sec1
        move    a,x:$ffef
        clr     a

; Stel van TB & RB registers

        move    #$489200,a              ;4800 monsters per sekonde

pri2    btst    #6,x:$ffee
        jcc     pri2
        move    b,x:$ffef
sec2    btst    #6,x:$ffee
        jcc     sec2
        move    a,x:$ffef
        clr     a

; Stel van Beheer register

        move    #$00E700,a

pri3    btst    #6,x:$ffee
        jcc     pri3
        move    b,x:$ffef
sec3    btst    #6,x:$ffee
        jcc     sec3
        move    a,x:$ffef
        clr     b

;Viterbi opstel
;----------------
;Begin nodusse vir paaie 1 tot 8

        move    #>$0,r1                 ;0 - 7 in X:$(400,410...470)
        move    #>$400,r2
        move    #>$10,n2
        .LOOP   #8
        move    r1,a1
        move    (r1)+
        move    a1,x:(r2)+n2
        .ENDL

;moontlike afkomstige nodusse ACEG vir nodusse ABCD.

        move    #>$0,x0
        move    x0,x:$340               ;A
        move    #>$2,x0
        move    x0,x:$341               ;C
        move    #>$4,x0
        move    x0,x:$342               ;E
        move    #>$6,x0
        move    x0,x:$343               ;G

;moontlike afkomstige nodusse BDFH vir nodusse EFGH.

        move    #>$1,x0
        move    x0,x:$344               ;B
        move    #>$3,x0
        move    x0,x:$345               ;D
```

```
        move    #>$5,x0
        move    x0,x:$346               ;F
        move    #>$7,x0
        move    x0,x:$347               ;H

;Maak registers skoon                   X:200-2ff + Y:200-2ff
;---------------------------
        clr     a
        move    a,y:$384                ;"Long_Int" uitset register

;Stel Registers
;-----------------
        move    #>$240,y0               ;Sync deteksie wysers
        move    y0,y:$201
        move    #>$240,y0
        move    y0,y:$202
        move    #>$200,y0
        move    y0,y:$203

        move    #>$250,r4               ;Baud aanvangswaarde
        move    #>$ffffff,x0
        move    x0,x:$250

        move    #>$270,y1               ;Baud beginpunt in AFSTEL
        move    y1,y:$20c

        move    #>$500,y1               ;Beginpunt wyser van VORIGE BAUD
        move    y1,y:$208

        move    #>0,r7                  ;Viterbi matriks S-buffer wysers
        move    #>15,m7
        move    #>1,n7

        move    #>$380,y1               ;Dekodeerder S-Buffer wyser
        move    y1,y:$205
        clr     a
        move    a,y:$206

        move    #>$600,a1               ;uitlees register wyser
        move    a1,y:$204

        rts

;-------------------------------------------------------------------------------
LEES_MONSTER
;---------------------
din     btst    #6,x:$ffee              ; data van " AIC" na ram
        jcc     din
        movep   x:$ffef,b
        move    #$FFFC00,x0
        and     x0,b
        move    b,y:$200
        move    y:$200,a                ;Skaal waardes af
        do      #12,dvde
        asr     a
dvde
        move    a,y:$200
        move    #$0,b
```

```
        move    b,x:$ffef
        rts

;-------------------------------------------------------------------------------
DETEKSIE
;-------------
        move    y:$201,r1               ;laai wysers
        move    #>1,n1
        move    #>7,m1
        move    y:$202,r2
        move    #>1,n2
        move    #>1,m2
        move    y:$203,r3
        move    #>9,n3
        move    #>63,m3

;Som_8
        move    x:(r1),x1               ;ou waarde (ow)
        move    x:$248,a                ;totaal (t)
        sub     x1,a    y:$200,x0       ;t - ow
        move    x0,x:(r1)+n1            ;nuwe in s_buffer
        add     x0,a                    ;nuwe waarde (nw)
        move    a,x:$248                ;t + nw = nt ;stoor nt van 8 samples
        abs     a

;Som_2
        move    a,y:(r2)+n2
        move    y:(r2),x0
        add     x0,a

;Buffer_64
        move    a,x:(r3)-n3             ;Mt
        move    x:(r3)-n3,x0            ;Mt-9
        add     x0,a    x:(r3)-n3,x0    ;Mt-18
        add     x0,a    x:(r3)-n3,x0    ;Mt-27
        add     x0,a    x:(r3)-n3,x0    ;Mt-36
        add     x0,a    x:(r3)-n3,x0    ;Mt-45
        add     x0,a    x:(r3)-n3,x0    ;Mt-54
        add     x0,a    x:(r3),x0       ;Mt-63
        add     x0,a                    ;Som van 8 vertragings

        move    r1,y:$201               ;stoor wysers
        move    r2,y:$202
        move    r3,y:$203
        move    #>$ffff,m1              ;maak S-buffers tot niet.
        move    #>$ffff,m2
        move    #>$ffff,m3

        rts

;-------------------------------------------------------------------------------
ZERO_DET_9
;----------------

        move    a,y1
        move    x:$250,x0
        move    (r4)+
        .IF     y1 <LO> x0              ;vergl. met vorige waarde
```

```
        move    #>$250,r4
        nop
        .ENDI
        move    y1,x:(r4)
        move    y:$200,y0
        move    y0,y:(r4)                       ;sorteer monsters

        rts

;---------------------------------------------------------------------------
AFSTEL
;----------

        move    #>$250,r1                       ; Y:250 - 258 na X:279 - 281
        move    #>$29,n1
        .LOOP   #9
        move    y:(r1),a
        move    a,x:(r1+n1)
        move    (r1)+
        nop
        .ENDL

        move    #>$273,r1                       ; X:273 - 27b na X:290 - 298
        move    #>$1d,n1
        .LOOP   #9
        move    x:(r1),a
        move    a,x:(r1+n1)
        move    (r1)+
        nop
        .ENDL

        move    #>$270,r1                       ; X:279 - 281 na X:270 - 278
        move    #>$9,n1

        .LOOP   #9
        move    x:(r1+n1),a
        move    a,x:(r1)
        move    (r1)+
        nop
        .ENDL

        rts

;---------------------------------------------------------------------------
FILTER_DET_DEMOD
;---------------------------

        move    #>$500,r1                       ;SIN_tabel S-buffer vir VORIGE BAUD
        move    #>255,m1
        move    #>32,n1
        move    y:$208,r1                       ;Beginpunt wyser van VORIGE BAUD

        move    #>$260,r3                       ;Uitset wyser
        move    #>4,n3

        clr     a
        move    a,x:$260                        ;Herstel
        move    a,x:$261
```

```
        move    a,x:$262
        move    a,x:$263

        lua     (r1)+n1,r1              ;eerste vertraging : +45deg = 0deg.

;Deteksie                               ;Produk & Som

        .LOOP   #4                      ;4 Vertragings
        move    r1,y:$208               ;stoor vertragings wyser
        move    #>$290,r2               ;Beginpunt wyser van BAUD

        .LOOP   #9                      ;Gemiddeld oor 9 monsters
        move    y:(r1)+n1,y1            ;VORIGE BAUD
        move    x:(r2)+,x1              ;BAUD
        mpy     x1,y1,a
        asr     a
        move    a0,a
        move    x:(r3),x1
        add     x1,a
        move    a,x:(r3)
        .ENDL

        move    x:(r3),a                ;Waardes van ander vertragings
        neg     a                       ;180=-0, 225=-45, 270=-90, 315=-135
        move    a,x:(r3+n3)
        move    (r3)+

        move    y:$208,r1               ;Herstel vertraging beginpunt
        nop
        lua     (r1)+n1,r1              ;Volgende vertraging = +45
        .ENDL

        move    #>$ffff,m1              ;Maak S-buffer tot niet.

;Kies die grootste

        move    x:$260,a                ;Opstel
        move    a,x:$26a
        clr     b
        move    b,x:$26b
        move    #>$1,r1                 ;Fase wyser
        move    #>$260,n1

        .LOOP   #7
        move    x:(r1+n1),b
        .IF     a <LT> b                ;as a >= b dan a=a
        move    b,a                     ;as b > a dan a=b
        move    x:(r1+n1),x1
        move    x1,x:$26a               ;Grootste filt-produk waarde "signed".
        move    r1,x:$26b               ;Fase waarde van grootste waarde.
        .ENDI
        move    (r1)+
        .ENDL

;Verskil in korrelasie
                                        ; = ( +/- grootste ) - (moontlikhede)
        move    #>$260,r1
        move    #>$a0,n1
```

```
        .LOOP  #8
        move   x:$26a,a
        move   x:(r1),x1
        sub    x1,a
        abs    a
        move   a,y:(r1+n1)              ;na Y:(260 + a0) = 300
        move   (r1)+
        .ENDL

        rts

;-------------------------------------------------------------------------------
PAS_BAUD
;-------------
Opstel
        move   #>$500,r1                ;Opstel SIN - tabel S-buffer
        move   #>255,m1
        move   #>32,n1

        move   #>$500,r2                ;tabel wyser
        move   #>255,m2
        move   #>32,n2                  ;Eerste venstergrootte

        clr    a
        move   a,y:$390                 ;Huidige waarde
        move   a,y:$391                 ;Grootste waarde

        .LOOP  #2                       ;Pas oor 2 siklusse

        .LOOP  #8                       ;Pas op sintabel in hoeveel stappe
        move   r2,r1                    ;laai SIN_wyser
        move   #>$290,r3                ;Monster wyser begin

        .LOOP  #9                       ;Pas oor hoeveel monsters      !!
        move   y:(r1)+n1,y1
        move   x:(r3)+,x1
        mpy    x1,y1,a
        asr    a
        move   a0,a
        move   y:$390,x0
        add    x0,a
        move   a,y:$390
        .ENDL

        move   y:$391,b                 ;Bepaal die grootste ooreenkoms
        .IF    a <GT> b
        move   a,y:$391                 ;stoor grootste
        move   r2,y:$208                ;SIN beginpunt " Y:$208 "
        .ENDI

        clr    a
        move   a,y:$390                 ;herstel huidige totaal
        lua    (r2)+n2,r2               ;volgende fase

        .ENDL

        clr    a
        move   a,y:$391                 ;herstel grootste totaal van vorige venster
```

```
        move    y:$208,r2               ;beginpunt van beste "pas"
        move    #>16,n2
        nop
        lua     (r2)-n2,r2              ;nuwe begin = (beginpunt - 16)
        move    #>$4,n2                 ;stap grootte = 4  (4/256 = +/- 3 grade)

        .ENDL

        move    #>$ffff,m1
        move    #>$ffff,m2

        rts

;-------------------------------------------------------------------------------
TEL&HERSTEL
;------------------
        move    #>1,x1
        move    x:$26d,a
        add     x1,a
        move    a,x:$26d                ;tel baud

        move    #>$250,r4               ;herstel baud beginpunt
        move    #>$ffffff,x0
        move    x0,x:$250               ;herstel beginpunt register
        rts

;-------------------------------------------------------------------
FASE_TOETS
;-----------------
        move    y:$20b,a
        .IF     a <EQ> #>0              ;As 0 grade vlag nie gestel is
;-----------------------------------
        move    y:$20a,a
        .IF     a <EQ> #>0              ;As 180 grade vlag nie gestel is
        move    x:$26b,a                ;Fase in
        move    #>$4,x0
        move    #>$1,x1

        .IF     a <NE> x0               ;TOETS vir 10 baud 180 grade
        move    #>$0,x1
        move    x1,x:$26e
        .ENDI
        move    x:$26e,a
        add     x1,a
        move    a,x:$26e

        move    #>10,x0
        .IF     a <EQ> x0
        move    #>$1,a1
        move    a1,y:$20a               ;Stel vlag
        .ENDI
        .ENDI
;---------------------------------
        move    y:$20a,a
        .IF     a <EQ> #>1              ;As 180 grade vlag gestel is
        move    x:$26b,a
        move    #>$0,x0
        move    #>$1,x1
```

```
        .IF     a <NE> x0               ;TOETS vir 10 baud 0 grade
        move    #>$0,x1
        move    x1,x:$26f
        .ENDI
        move    x:$26f,a
        add     x1,a
        move    a,x:$26f

        move    #>10,x0
        .IF     a <EQ> x0
        move    #>$1,a1
        move    a1,y:$20b               ;Stel vlag
        .ENDI
        .ENDI
;--------------------------------
        .ENDI

        rts

;-------------------------------------------------------------------------------
;*******************************************************************************
;                       ** VITERBI **
;*******************************************************************************
;-------------------------------------------------------------------------------
NODUS_A
;------------
        move    y:$300,x0               ;Huidige 4 stap afstande na nodus.
        move    x0,x:$321
        move    y:$304,x0
        move    x0,x:$322
        move    y:$302,x0
        move    x0,x:$323
        move    y:$306,x0
        move    x0,x:$324

        move    #$310,r1                ;Afkomstige nodus pad afstande
        jsr     VIER_AFSTANDE

        move    #$340,r2                ;4 Totale afstande tot by huidige nodus
        jsr     KORTSTE_STAP

        move    #$310,r1                ;Nuwe totale afstand uitset wyser
        move    #$400,r6                ;Matriks wyser
        move    #$10,n6
        jsr     SKRYF_IN_MATRIKS

        rts

;-------------------------------------------------------------------------------
NODUS_B
;------------
        move    y:$304,x0
        move    x0,x:$321
        move    y:$300,x0
        move    x0,x:$322
        move    y:$306,x0
        move    x0,x:$323
```

```
        move    y:$302,x0
        move    x0,x:$324

        move    #$310,r1
        jsr     VIER_AFSTANDE

        move    #$340,r2
        jsr     KORTSTE_STAP

        move    #$314,r1
        lua     (r6)+n6,r6
        jsr     SKRYF_IN_MATRIKS

        rts

;-------------------------------------------------------------------------------
NODUS_C
;------------
        move    y:$302,x0
        move    x0,x:$321
        move    y:$306,x0
        move    x0,x:$322
        move    y:$300,x0
        move    x0,x:$323
        move    y:$304,x0
        move    x0,x:$324

        move    #$310,r1
        jsr     VIER_AFSTANDE

        move    #$340,r2
        jsr     KORTSTE_STAP

        move    #$311,r1
        lua     (r6)+n6,r6
        jsr     SKRYF_IN_MATRIKS

        rts

;-------------------------------------------------------------------------------
NODUS_D
;------------
        move    y:$306,x0
        move    x0,x:$321
        move    y:$302,x0
        move    x0,x:$322
        move    y:$304,x0
        move    x0,x:$323
        move    y:$300,x0
        move    x0,x:$324

        move    #$310,r1
        jsr     VIER_AFSTANDE

        move    #$340,r2
        jsr     KORTSTE_STAP

        move    #$315,r1
```

```
        lua     (r6)+n6,r6
        jsr     SKRYF_IN_MATRIKS

    rts

;-------------------------------------------------------------------------------
NODUS_E
;------------
        move    y:$301,x0
        move    x0,x:$321
        move    y:$305,x0
        move    x0,x:$322
        move    y:$303,x0
        move    x0,x:$323
        move    y:$307,x0
        move    x0,x:$324

        move    #$314,r1
        jsr     VIER_AFSTANDE

        move    #$344,r2
        jsr     KORTSTE_STAP

        move    #$312,r1
        lua     (r6)+n6,r6
        jsr     SKRYF_IN_MATRIKS

        rts

;-------------------------------------------------------------------------------
NODUS_F
;------------
        move    y:$305,x0
        move    x0,x:$321
        move    y:$301,x0
        move    x0,x:$322
        move    y:$307,x0
        move    x0,x:$323
        move    y:$303,x0
        move    x0,x:$324

        move    #$314,r1
        jsr     VIER_AFSTANDE

        move    #$344,r2
        jsr     KORTSTE_STAP

        move    #$316,r1
        lua     (r6)+n6,r6
        jsr     SKRYF_IN_MATRIKS

        rts

;-------------------------------------------------------------------------------
NODUS_G
;------------
        move    y:$303,x0
        move    x0,x:$321
```

```
        move    y:$307,x0
        move    x0,x:$322
        move    y:$301,x0
        move    x0,x:$323
        move    y:$305,x0
        move    x0,x:$324

        move    #$314,r1
        jsr     VIER_AFSTANDE

        move    #$344,r2
        jsr     KORTSTE_STAP

        move    #$313,r1
        lua     (r6)+n6,r6
        jsr     SKRYF_IN_MATRIKS

        rts

;-------------------------------------------------------------------------------
NODUS_H
;-------------
        move    y:$307,x0
        move    x0,x:$321
        move    y:$303,x0
        move    x0,x:$322
        move    y:$305,x0
        move    x0,x:$323
        move    y:$301,x0
        move    x0,x:$324

        move    #$314,r1
        jsr     VIER_AFSTANDE

        move    #$344,r2
        jsr     KORTSTE_STAP

        move    #$317,r1
        lua     (r6)+n6,r6
        jsr     SKRYF_IN_MATRIKS

        rts

;-------------------------------------------------------------------------------
VIER_AFSTANDE
;----------------------
        move    #>$321,r2               ;inset/uitset wyser

        .LOOP   #$4
        move    x:(r1),a                ;afstand tot by afkomstige nodus
        move    x:(r2),x0               ;afstand van huidige stap
        add     x0,a
        move    a,y:(r2)                ;4 totale afstande tot by huidige nodus
        move    (r1)+
        move    (r2)+

        .ENDL
```

```
        rts

;-------------------------------------------------------------------------------
KORTSTE_STAP
;--------------------
        move    #>$321,r1
        move    #>$1,n1
        move    #>$1,n2
        move    y:(r1),a                ;4 totale afstande
        move    x:(r2),y0               ;moontlike afkomstige nodus

        .LOOP   #3
        move    y:(r1+n1),x0            ;totale afstande
        .IF     a <GT> x0
        move    x0,a
        move    x:(r2+n2),y0
        .ENDI
        move    (r1)+
        move    (r2)+
        .ENDL
        move    a,y:$330                ;kortste afstand
        move    y0,y:$331               ;afkomstige nodus

        rts

;-------------------------------------------------------------------------------
SKRYF_IN_MATRIKS
;---------------------------

        move    r6,r0
        move    r7,n0
        move    y:$331,x0
        move    x0,x:(r0+n0)            ;afkomstige nodus
        move    y:$330,y0
        move    y0,y:(r0+n0)            ;kortste afstand

        move    y0,y:(r1)               ;nuwe totale afstand tot by nodus.

        rts

;-------------------------------------------------------------------------
SKRYF_NUWE_IN_OU
;----------------------------
        move    #>$310,r1               ;Stoor nuwe as ou nodus afstande
        .LOOP   #8
        move    y:(r1),y0

        .IF     y0 <GT> #>$FFFFF
        move    #>$310,r2
        .LOOP   #8
        move    y:(r2),a
        asr     a
        asr     a
        move    a,y:(r2)+
        .ENDL
        .ENDI

        move    y:(r1),y0
```

```
        move    y0,x:(r1)+
        .ENDL

        rts

;-------------------------------------------------------------------------------
KORTSTE_VAN_PAAIE
;------------------------------
        move    #>$400,r0               ;8 nodus afstande
        move    r7,n0
        move    #>$0,r1                 ;8 nodus wyser
        lua     (r0)+n0,r0              ;Waar in matriks
        move    #>$10,n0

        move    r1,y1                   ;Nodus 0 in
        move    y:(r0),a                ;Pad afstand tot by nodus 0 (A)

        .LOOP   #7
        lua     (r0)+n0,r0
        move    (r1)+                   ;Volgende nodus
        move    y:(r0),x0               ;Pad afstand tot by volgende nodus
        .IF     a <GT> x0
        move    x0,a                    ;kortste pad
        move    r1,y1                   ;eind nodus
        .ENDI
        nop
        .ENDL

        move    y1,y:$370               ;eind nodus met kortste pad

        rts

;-------------------------------------------------------------------------------
TERUG_SPOOR
;--------------------
        move    #$400,r0
        move    r7,n0
        move    #>15,m0
        move    y:$370,y1               ;Laai eind nodus
        lua     (r0)+n0,r0              ;Waar in matriks
        move    #>$10,x1
        move    #>$361,r3               ;Terugspoor pad wyser

        .LOOP   #14
        mpy     x1,y1,a
        asr     a
        move    a0,a
        move    r0,x0
        add     x0,a
        move    a,r1
        nop
        move    x:(r1),y1               ;vorige nodus
        move    y1,y:(r3)+              ;terugspoor nodusse in Y361..Y36E
        move    (r0)-
        .ENDL

        move    y1,y:$370
        move    #>$ffff,m0
```

```
        rts

;-------------------------------------------------------------------------------
DATA_NA_INT
;------------------
        move    y:$370,a                ;Laai eerste nodus
        move    y:$206,r2               ;Laai "Int" wyser

;Skakel nodus om na data

        .IF     a <GT> #>$3             ; {Nodus 0 & 4 = (0)Data 00}
        move    #>$4,x1                 ; {Nodus 1 & 5 = (1)Data 01}
        sub     x1,a                    ; {Nodus 2 & 6 = (2)Data 10}
        move    a,y:$370                ; {Nodus 3 & 7 = (3)Data 11}
        .ENDI

;Skuif 2-bis data blok in "Integer"

        clr     a
        move    y:$370,a0               ;Laai data 2-bit waarde
        asr     a                       ;mins beduidende bis na carry
        rol     a1
        move    a1,b1                   ;stoor mins beduidende bis

        asr     a                       ;mees beduidende bis na carry
        move    y:$384,a1
        rol     a1                      ;mees beduidende bis in "INT"
        move    a1,y:$384
        move    (r2)+

        move    b1,a0
        asr     a                       ;mins beduidende bis na carry
        move    y:$384,a1
        rol     a1                      ;mins beduidende bis in "INT"
        move    a1,y:$384
        move    (r2)+

        .IF     r2 <EQ> #>16            ;toets of Int vol is
        move    y:$384,a1
        move    a1,y:$209
        jsr     LEES_UIT
        clr     a
        move    a,y:$384
        move    a,r2                    ;herstel Int wyser
        .ENDI

        move    r2,y:$206               ;stoor Int wyser

        rts

;-------------------------------------------------------------------------------
LEES_UIT
;-------------
        move    y:$204,r0               ;uitlees wyser
        move    y:$209,a1
        move    a1,x:(r0)+
```

```
        move    r0,y:$204                       ;stoor wyser
        .IF     r0 <GT> #>$1003                 ;lees uit na file as geheue vol is.
        move    #>$600,r0
        .LOOP   #2560                           ;2560 Integers
out1    btst    #1,x:$FFE9
        jcc     out1
        move    x:(r0),a1
        movep   a1,x:$ffeb
        move    (r0)+
        .ENDL
        .ENDI

        rts


;-------------------------------------------------------------------------------
;*******************************************************************************
```

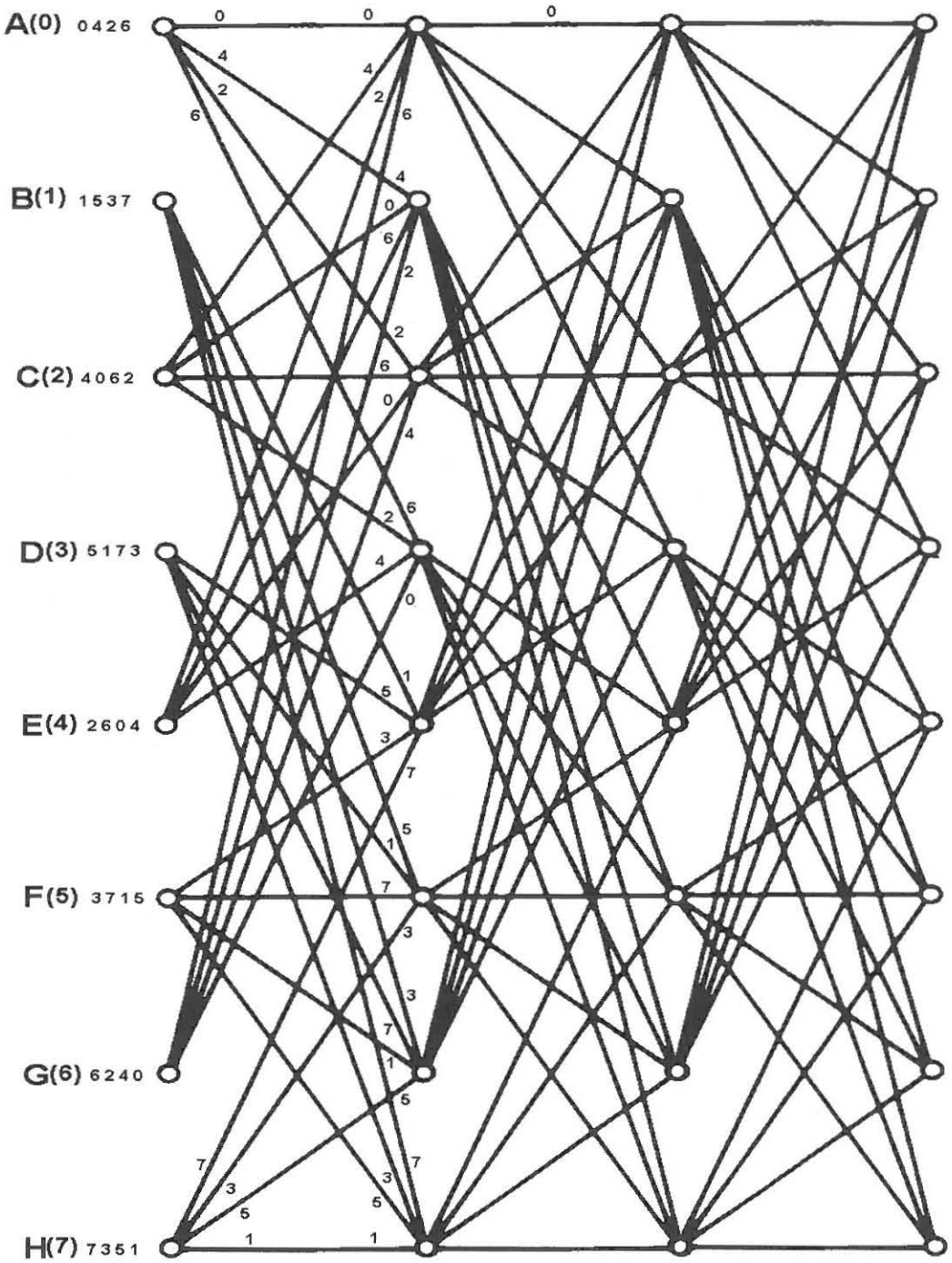
BYLAE B.6 : PROGRAM KONTROLE.PAS

```
program KONTROLE;
uses
 crt,dsplib4;

var
   bs,a,b,timeout:longint;
   tx,rx,dx,bet,bae,r,ie,bt,be,k,l,m,n:integer;
   ber:real;
   din,dout,err:text;
   baseaddress:word;

begin
  writeln;
  writeln('KONTROLE ');
  l:=0;
  n:=0;
  ie:=0;
  assign(din,'rdm.dat');
  reset(din);
  assign(dout,'ontv.dat');
  reset(dout);
  assign(err,'errlog.dat');
  rewrite(err);

  readln(dout,rx);
  readln(dout,rx);
  bet:=0;
  be:=0;

  for m:=1 to 5 do
     begin
     reset(din);

     for k:=1 to 512 do
        begin
        readln(din,tx);
        readln(dout,rx);
        dx:=(tx XOR rx);
        n:=n+1;
        If (dx <> 0) then
          begin
          writeln(err,n,' TX=',tx,' RX=',rx);
          ie:=ie+1;
          for l:= 0 to 15 do
             begin
             bt:=$0001;
             bt:=bt SHL l;
             be:=(bt AND dx);
```

```
            If (be > 0) then
              begin
              bet:=bet+1;
              end;
            end;
          end;
        end;
    end;

  bae:=bet div 2;
  ber:=bet/40960;

  writeln;
  writeln(' “Integer” fout = ',ie);
  writeln;
  writeln(' Simboolfout = ',bae);
  writeln;
  writeln(' Bisfout = ',bet);
  writeln;
  writeln(' Bisfouttempo = ',ber);

  writeln(err);
  writeln(err,' “Integer” fout = ',ie);
  writeln(err);
  writeln(err,' Simboolfout = ',bae);
  writeln(err);
  writeln(err,' Bisfout = ',bet);
  writeln(err);
  writeln(err,' Bisfouttempo = ',ber);

  close(din);
  close(dout);
  close(err);
end.
```

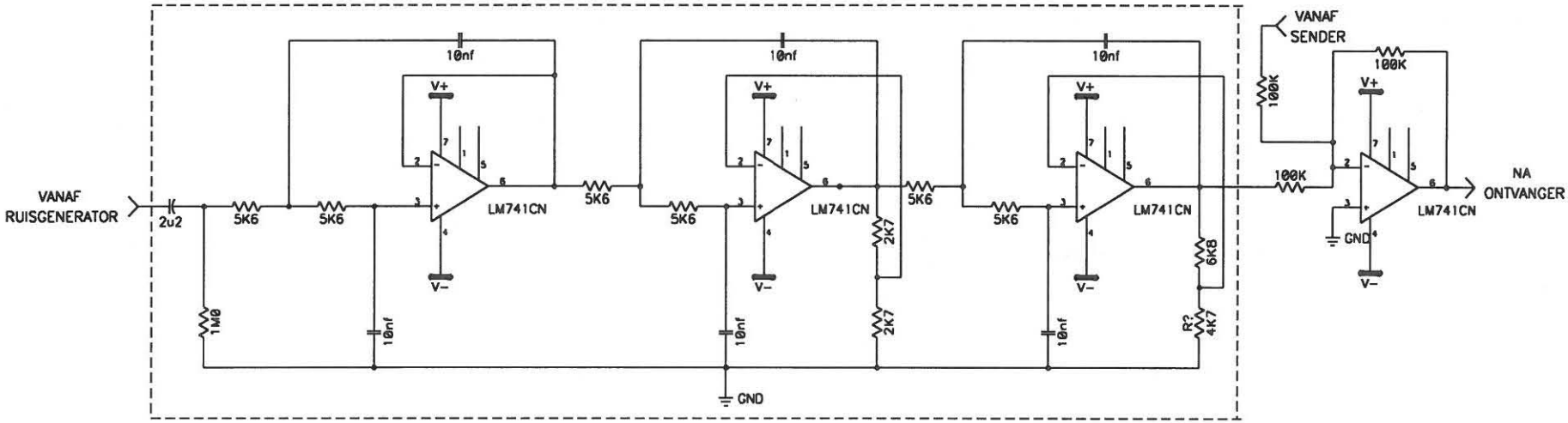
Central University of Technology, Free State

BYLAE C : 8-STAAT TRELLIS



Central University of Technology, Free State

BYLAE D : ONDERDEURLAATFILTER



Central University of Technology, Free State

BYLAE E : TABEL VAN RESULTATE

RUIS Vwgk (Vn)	Ruis energie (aangepas)	SEIN/RUIS (E/No)	8DPSK (Eb/No)	Trellis & Viterbi (Eb/No)	8DPSK (SER)	8DPSK (BER)	8DPSK + trellis & Viterbi (BER)
0.15	0.020408	24.315	19.544	21.304	0.000007	0.000004	
0.17	0.026212	23.228	18.456	20.217	0.000153	0.00009	
0.19	0.032743	22.262	17.49	19.251	0.000516	0.000301	
0.21	0.039999	21.392	16.621	18.382	0.00106	0.000618	
0.24	0.052243	20.232	15.461	17.222	0.003278	0.001912	
0.27	0.6612	19.209	14.438	16.199	0.005587	0.003259	0.000005
0.3	0.08163	18.294	13.523	15.284	0.012005	0.007003	0.00009
0.34	0.104849	17.207	12.436	14.197	0.025823	0.015063	0.000481
0.38	0.130971	16.241	11.47	13.231	0.042006	0.024504	0.001121
0.43	0.167704	15.167	10.396	12.157	0.073207	0.042704	0.003115

BRONNELYS

1. **Baher, H.** *Analog and Digital Signal Processing*, New York, John Wiley & Sons, 1991.

2. **Bic, J.C., Duponteil, D. and Imbeaux, J.C.** *Elements of Digital Communications*, Chichester, John Wiley & Sons, 1991.

3. **Biglieri, E., Divsalar, D., McLane, P.J. and Simon, M.K.** *Introduction to Trellis-Coded Modulation with Applications*, New York, Macmillan Publishing Company, 1991.

4. **Bingham, J.A.C.** *The Theory and Practice of Modem Design*, New York, John Wiley & Sons, 1988.

5. **Bissell, C.C. and Chapman, D.A.** *Digital Signal Transmission*, New York, Cambridge University Press, 1992.

6. **Clark, G.C. and Cain, J.B.** *Error-Correction Coding for Digital Communications*, New York, Plenum Press, 1988.

7. **Coetzer, H.F.** *Development of a sequency division multiplex system for the transmission of digital data*, Bloemfontein, Technikon Free State, Unpublished technical report, 1996.

8. **Courtenay, T.D.** *The Development of a novel modem structure for connection of rural to diginet*, Cape Town, Unpublished thesis University of Cape Town, 1990.

9. **Green, D.C.** *Data Communication*, New York, John Wiley & Sons, 1991.

10. **Haykin, S.** *Digital Communications*, New York, John Wiley & Sons, 1988.

11. **Ifeachor, E.C. and Jervis, B.W.** *Digital Signal Processing : A Practical Approach*, Workingham, Addison Wesley Publishing Company, 1993.

12. **Lafrance, P.** *Fundamental Concepts in Communication*, Englewood Cliffs, Prentice-Hall International Editions, 1990.

13. **Lin, S. and Costello, D.J.** *Error Control Coding: Fundamentals and Applications*, Englewood Cliffs, Prentice-Hall International Editions, 1983.

14. **Motorola.** *DSP56000/DSP56001 Digital Signal Processor User's Manual*, Phoenix, Motorola Inc., 1990.

15. **Motorola.** *Motorola Digital Signal Processors*, Phoenix, Motorola Inc., 1989.

16. **Peralex.** *Peralex SIG-56 Reference manual*, Kalk Bay, Peralex, 1992.

17. **Peyton, Z. and Peebles, J.R.** *Digital Communication Systems*, Englewood Cliffs, Prentice-Hall International Editions, 1987.

18. **Proakis, J.G.** *Digital Communications*, New York, McGraw-Hill International Editions, 1989.

19. **Roden, M.S.** *Digital Communication Systems Design*, Englewood Cliffs, Prentice-Hall International Editions, 1988.

20. **Schwartz, M.** *Information Transmission, Modulation and Noise*, New York, Mc-Graw-Hill International Editions, 1990.

21. **Crouch, S.E.C. and Jedwab, J.** Coding in 100VG-AnyLAN. *Hewlett Packard Journal*, vol.46, no.4, 1995, pp. 27-32.

22. **Stanford Telecommunications.** *The Forward Error Correction Handbook*, Santa Clara, Stanford Telecommunications Inc., 1992.

23. **Stark, H., Tuteur, F.B. and Anderson, J.B.** *Modern Electrical Communications*, London, Prentice-Hall International Editions, 1988.

24. **Texas Instruments.** *Application notes - TLC320441, TLC32044C Voice-band analog interface circuits*, Dallas, Texas Instruments, 1988.

25. **Tomasi, W.** *Advanced Electronic Communications Systems*, Englewood Cliffs, Prentice-Hall International Editions, 1987.

26. **v.d. Walt, K.N.** *Development of a programmable time-domain speech-synthesis system*, Bloemfontein, Unpublished thesis Technikon Free State, 1995.

27. **Viterbi, A.J. and Omura, J.K.** *Principles of Digital Communication and Coding*, Tokyo, McGraw-Hill International Editions, 1979.

28. **Ungerboeck, G.** Channel coding with multilevel/phase signals, *IEEE transactions on information theory*, vol.28, no.1, 1982, pp. 55-66.

Central University of Technology, Free State

ADDISIONELE BRONNE

(Geraadpleeg maar nie na verwys nie)

1. **Bateman, A. and Yates, W.** *Digital Signal Processing Design*, London, Pitman Publishing, 1988.

2. **Black, U.** *The V Series Recommendations*, New York, McGraw-Hill International Editions, 1991.

3. **CCITT.** *Data Communication over the Telephone Network*, Geneva, XIth Plenary Assembly, 1988.

4. **Frerking, M.E.** *Digital Signal Processing in Communication Systems*, New York, Van Nostrand Reinhold, 1994.

5. **Gibson, J.D.** *Digital and Analog Communications*, New York, Macmillan Publishing Company, 1993.

6. **Gitlin, R.D., Hayes, J.F. and Weinstein, S.B.** *Data Communications Principles*, New York, Plenum Press, 1994.

7. **Hennefeld, J.** *Using Turbo Pascal 4.0-6.0*, Boston, PWS-Kent Publishing Company, 1992.

8. **Jeruchim, M.C., Balaban, P. and Shanmugan, K.S.** *Simulation of Communication Systems*, New York, Plenum Press, 1992.

9. **Jones, N. B. and McWatson**, J. D. *Digital Signal Processing : Principles, Devices and Applications*, UK, Peter Peregromus Ltd, 1990.

10. **Kingsbury, N.G.** A 1200 Bit/s QPSK Full Duplex Modem, *IEEE Journal of solid-state circuits*, vol. sc-19, no.6, 1984, pp. 878-887.

11. **Korn, I.** Error probability of offset differential phase shift keying with intersymbol and adjacent channel interference, *IEEE Proceedings*, vol.135, no.2, 1988, pp. 175-182.

12. **Lathi, B.P.** *Modern Digital and Analog Communication Systems*, Philadelphia, Holt, Rinehart and Winston Inc., 1989.

13. **Mathsoft**, *Mathcad 5.0 User's Guide*, Cambridge, Mathsoft Inc., 1994.

14. **Olivier, G.C.J.J.** *'n Ondersoek na die gebruik van Trelliskode as enkoderingstegniek vir datakommunikasie*, Bloemfontein, Ongepubliseerde verhandeling Technikon Vrystaat, 1995.

15. **Orfanidis, S.J.** *Optimum Signal Processing*, New York, McGraw-Hill International Editions, 1990.

16. **Roberts, R.A. and Mullis, C.T.** *Digital Signal Processing*, Massachusetts, Addison Wesley Publishing Company, 1987.

17. **Sinnema, W. and McGovern, T.** *Digital, Analog, and Data Communication*, London, Prentice-Hall International Editions, 1986.

18. **Schweber, W.L.** *Data Communications*, New York, McGraw-Hill International Editions, 1988.

19. **Ungerboeck, G.** Trellis-coded modulation with redundant signal sets, *IEEE Communications magazine*, vol.25, no.2, 1987, pp. 5-21.