TECHNISCHE HOGESCHOOL DELFT
Afdeling der Elektrotechniek

Aard : Digitale modulatie methoden. Deel I.
Omvang : Deel I : 59 blz. - Deel II : 75 blz.
Datum : Juli 1982

Vakgroep : Transmissie van Informatie

Auteur : Bart Wijne

Titel : Ontwerp en simulatie van een correlatie ontvanger met Viterbi processor voor "Correlative Phase Shift Keying" CORPSK(4-5)

Korte inhoud: Zie voorwoord

Mentor: Ir. J.S. van Sinttruijen

INHOUD DEEL II

Voorwoord

Een ontwerp voor een digitale ontvanger voor Correlative Phase Shift Keying CORPSK(4-5) voor Mobiele Radio met bitrate = 72 Kbit/s wordt gepresenteerd.
CORPSK(4-5) heeft een zeer smal frekwentiespectrum rond de draaggolf (-60 dB bij f_{bit}). Dit wordt bereikt zonder filtering van het eindmodulaat.
De ontvanger maakt gebruik van een Viterbi processor om op efficiënte wijze een correlatie-integraal te bepalen.
Voor de ontvanger met en zonder Viterbi processor is een simulatieprogramma ontworpen. Resultaten van deze simulatie worden vergeleken met resultaten uit de literatuur van DQPSK.
Dit werk is gedaan bij Philips' Telecommunicatie Industrie te Huizen op de afdeling Voor Ontwikkeling Radio, in het kader van een afstudeeropdracht voor het Ingenieursdiploma van de Technische Hogeschool te Delft bij de vakgroep Transmissie van de Informatie onder voorzitterschap van Prof.Dr.Ir. J.L. Bordewijk.

De mentoren zijn Ir. D. Muilwijk van PTI en Ir. J.S. van Sinttruyen van THD.
De auteur dankt allen die hebben bijgedragen in de totstandkoming van dit afstudeerverslag.

Bart Wijne , 1 juli 1982

In het laatste hoofdstuk wordt een bespiegeling van de toepassingsmogelijkheden en toekomstverwachtingen voor CORPSK gegeven en wordt een samenvatting van de belangrijkste conclusies uit de voorgaande hoofdstukken gegeven.
De beschrijving van de voor de simulatie ontworpen software wordt gegeven in een aantal appendices.
Deze appendices zijn opgenomen in het tweede deel van het verslag.

1. Correlative Phase Shift Keying

1.1 Introductie

De toenemende vraag naar digitale transmissie in de radiofrekwentieband, stelt zware eisen aan de signaaleigenschappen en systeemorganisatie van digitale modulatietechnieken. Ten aanzien van de problematiek rond spectrumschaarste kunnen oplossingen gezocht worden in de volgende technieken.

1) Nieuwe locaties op hogere frekwenties.
2) Betere verdeling van de huidige locaties.
3) Frekwentie "re-use" technieken zoals : geografisch "re-use", "narrow-beam" antennes en dubbele polarisatie systemen.
4) Efficiënte broncoderingstechnieken.
5) Spectrum efficiënte modulatietechnieken.

De in dit verslag behandelde techniek heeft betrekking op het laatste punt van bovenstaande opsomming.
Voor een digitale modulatietechniek zijn de volgende systeemeigenschappen van belang.

a) Spectrum-efficiëntie, vermogensdichtheid van het frekwentiespectrum buiten de kanaalbandbreedte.
b) Vermogens-efficiëntie.
c) Constant omhullende. (geen amplitudemodulatie)
d) Robuust en eenvoudig ontwerp van de ontvanger en/of zender.

1.2 Amplitudemodulatie; fasemodulatie

De eindversterker van een radiosysteem zal om redenen van vermogensefficiëntie zo volledig mogelijk worden uitgestuurd, de versterkingskarakteristiek van de versterker is dan tgv verzadiging, niet lineair.
Als een in bandbreedte begrensd signaal een niet lineaire versterker passeert, dan zal bij amplitudemodulatie,

signaalvervorming optreden, hetgeen een verbreding van het frekwentiespectrum van dit signaal tot gevolg heeft. Bij fasemodulatie treedt geringe signaalvervorming op tgv niet-lineaire versterkers, er treedt dan alleen harmonische vervorming op. Voor spectrumefficiënte radiosystemen ligt het daarom voor de hand, fasemodulatie toe te passen.

1.3 Digitale modulatiemethoden

Vier-fase modulatiemethoden zoals QPSK en MSK vormen een goed compromis tussen noodzakelijke bandbreedte en vermogensefficiëntie, het is echter nodig dmv filters het spectrum buiten het kanaal tot een aanvaardbaar niveau te onderdrukken.
Het filter introduceert amplitudevariaties en als dit gefilterde modulaat een niet-lineaire versterker passeert zal het spectrum wederom verbreden.
Filteren na de eindversterker is meestal onpraktisch.
Verscheidene andere fasemodulatie methoden welke een oplossing bieden voor dit probleem zijn in de afgelopen jaren geïntroduceerd.
Phase-shaped QPSK [1] , [2] .
Continuous phase frequency shift keying (CPFSK) [3] .
Partial response FM [4] - [6] en multi-h CPFSK [7] , [8] .
Een overzicht van deze methoden wordt gegeven in [9] .
Uit deze artikelen zijn belangrijke conclusies te trekken.
De vermogensefficiëntie neemt toe tov de vermogensefficiëntie van QPSK als meer variatie in de mogelijke faseveranderingen per symbool wordt geïntroduceerd.
Methoden met lineaire faseveranderingen hebben een te hoge vermogensdichtheid buiten de hoofdbandbreedte.
Partial response FM met gefilterde pulsen heeft zowel een smalle hoofdbandbreedte als een lage vermogensdichtheid buiten de hoofdbandbreedte.

signaalvervorming optreden, hetgeen een verbreding van het frekwentiespectrum van dit signaal tot gevolg heeft. Bij fasemodulatie treedt geringe signaalvervorming op tgv niet-lineaire versterkers, er treedt dan alleen harmonische vervorming op. Voor spectrumefficiënte radiosystemen ligt het daarom voor de hand, fasemodulatie toe te passen.

1.3 Digitale modulatiemethoden

Vier-fase modulatiemethoden zoals QPSK en MSK vormen een goed compromis tussen noodzakelijke bandbreedte en vermogensefficiëntie, het is echter nodig dmv filters het spectrum buiten het kanaal tot een aanvaardbaar niveau te onderdrukken.
Het filter introduceert amplitudevariaties en als dit gefilterde modulaat een niet-lineaire versterker passeert zal het spectrum wederom verbreden.
Filteren na de eindversterker is meestal onpraktisch.
Verscheidene andere fasemodulatie methoden welke een oplossing bieden voor dit probleem zijn in de afgelopen jaren geïntroduceerd.
Phase-shaped QPSK [1] , [2] .
Continuous phase frequency shift keying (CPFSK) [3] .
Partial response FM [4] - [6] en multi-h CPFSK [7] , [8] .
Een overzicht van deze methoden wordt gegeven in [9] .
Uit deze artikelen zijn belangrijke conclusies te trekken.
De vermogensefficiëntie neemt toe tov de vermogensefficiëntie van QPSK als meer variatie in de mogelijke faseveranderingen per symbool wordt geïntroduceerd.
Methoden met lineaire faseveranderingen hebben een te hoge vermogensdichtheid buiten de hoofdbandbreedte.
Partial response FM met gefilterde pulsen heeft zowel een smalle hoofdbandbreedte als een lage vermogensdichtheid buiten de hoofdbandbreedte.

1.4 CORPSK

Correlative Phase Shift Keying (CORPSK) is een klasse van fase-modulätiemethoden welke nauw verwant zijn aan de genoemde methoden. De faseveranderingen zijn gecorreleerd en verlopen vloeiend waardoor een goede spectrumefficiëntie ontstaat. De correlatie veroorzaakt ook de gewenste variatie in de mogelijke faseveranderingen per symbool hetgeen resulteert in een goede vermogensefficiëntie.

1.5 De definitie van CORPSK

We beschouwen een datareeks $\{a_m\}$ $m \in N$
met als symbolenalfabet $\{a_1 \ldots\ldots a_q\}$

Het modulaat van een fase gemoduleerd signaal geven we als volgt aan :

$$m(t) = \cos(\omega_c t+\phi(t))$$

Het principe van CORPSK is als volgt samen te vatten :

1) De informatie uit $\{a_m\}$ wordt vertaald in faseveranderingen ϕ_m zodanig dat geen abrupte maar geleidelijk verlopende faseveranderingen ontstaan.
Het faseverschil over een symboolperiode bedraagt

$$\Delta\Phi_m = \Phi((m+1)T_s)-\Phi(mT_s) = c_m \frac{2\pi}{n}$$

n is een positief geheel getal en komt overeen met het aantal mogelijke fasepunten, ten tijde $t=mT_s$.
c_m bevat de informatie.

2) Opeenvolgende faseveranderingen zijn gecorreleerd. De faseverandering over een symboolperiode is een functie van het ingangsniveau, gedurende die periode en tenminste één voorafgaande periode. Dit wordt gerealiseerd door de reeks $\{c_m\}$ uit de reeks $\{a_m\}$ af te

leiden via een correlatiecoderingsformule. Het aantal mogelijke faseveranderingen (c_m) moet groter zijn dan het aantal mogelijke amplitudeniveau's (a_m).

3) Het fasetraject verloopt vloeiend, de afgeleide van de fase is een continue functie van de tijd.

Het op deze manier gedefinieerde signaal doorloopt vaste fasepunten (veelvouden van $\frac{2\pi}{n}$) op het eind van ieder symboolinterval.

De belangrijkste parameters van CORPSK zijn :
a) Het aantal mogelijke amplitudeniveau's A.
b) Het aantal mogelijke faseveranderingen M.
c) De correlatiecoderingsregel bv 1+D (duobinary) ("D" is de "delay operator").

Algemene notatie CORPSK(A-M,1+D).

1.6 CORPSK-modulator

In principe bevat een CORPSK-modulator een correlatie-encoder, een premodulatie-filter en een fasemodulator, zie fig. 1.1.
Omdat er tgv de correlatie-encoder meestal geen eenduidigheid meer is tussen het symbool (a_m) en de faseverandering (c_m) is een precoder nodig om foutpropagatie tegen te gaan.
De modulator kan een fasemodulator zijn, als de fase begrensd is of een frekwentiemodulator als de fase onbegrensd is.

Om de juiste fasepositie aan het eind van elk symboolinterval te bereiken moet de faseverandering of de frekwentieverandering van de modulator geregeld worden door een terugkoppellus.

Het laagdoorlaat premodulatie-filter $H(\omega)$ zorgt voor een glad faseverloop. Een belangrijke eis aan dit filter is dat de fase aan de uitgang door de vaste fasepunten moet gaan aan het eind van ieder symboolinterval. Als het filter wordt gevolgd door een fasemodulator, moet de overdrachtsfunctie aan het eerste Nyquist kriterium voldoen :

$$n_I(kT_s) = \begin{cases} 1 & \text{voor } k=0 \\ 0 & \text{voor } k\neq 0 \text{ , } k \text{ geheel} \end{cases} \qquad \text{(impuls responsie)}$$

(⊛ = convolutieteken)

$$c(t) = \sum_m c_m \, \delta(t-mT_s)$$

$$e(t) = c(t) \circledast n_I(t)$$

$$\phi_p(t) = \frac{2\pi}{n} \, c(t) \circledast n_I(t) \qquad \text{(fasemodulator)}$$

$\frac{2\pi}{n}$ is de overdrachtsconstante van de fasemodulator.

Als het premodulatie-filter gevolgd wordt door een frekwentiemodulator moet het voldoen aan het derde kriterium van Nyquist om de integratie-operatie in de modulator te compenseren.

$$\int_{(2k-1)(T_s/2)}^{(2k+1)(T_s/2)} n_{III}(t)dt = \begin{cases} 1 & \text{voor } k=0 \\ 0 & \text{voor } k\neq 0 \text{ , } k \text{ geheel} \end{cases}$$

De fase Φ_f van de frekwentiemodulator is nu :

$$\Phi_f(t) = \frac{2\pi}{nT_s} \int_{-\infty}^{t} c(t) \circledast n_{III}(t)dt$$

met $\frac{2\pi}{nT_s}$ als overdrachtsconstante van de modulator.

In [10] wordt aangetoond dat de volgende relatie geldt :

$$N_{III}(\omega) = N_I(\omega) \frac{(\omega T_s/2)}{\sin(\omega T_s/2)}$$

Onder aanname dat $N_I(\omega)=0$, $\omega \geq \frac{2\pi}{T_s}$

In het vervolg nemen we aan dat de modulator een frekwentie-modulator is en dat we voor de datareeks, blokvormen i.p.v. impulsen gebruiken.
Hieruit volgt :

$$H(\omega) = N_I(\omega) \left(\frac{(\omega T_s/2)}{\sin(\omega T_s/2)} \right)^2 \qquad (1)$$

Voor $N_I(\omega)$ kiezen we een "raised cosine" karakteristiek.

$$N_I(\omega) = \begin{cases} 1, & 0 \leq |\omega| \leq \pi(1-\alpha)/T_s \\ \frac{1}{2}\left(1-\sin\left((\omega T_s-\pi)/2\alpha\right)\right), & \\ \pi(1-\alpha)/T_s \leq \omega \leq \pi(1+\alpha)/T_s & \\ 0, \text{ elders} & \end{cases} \qquad (2)$$

$$C_m(\omega) = c_m \frac{\sin(\omega T_s/2)}{(\omega T_s/2)} e^{-j\omega m T_s} \qquad (3)$$

$$\phi_m(\omega) = C_m(\omega) H(\omega) \frac{K}{j\omega}$$

K is een schaalfactor opdat $\Delta\phi_b$ de vereiste waarde bereikt.

Het totale spectrum van de fase is :

$$\phi(\omega) = K \sum_m \frac{C_m(\omega) H(\omega)}{j\omega}$$

Substitueren we (1) (2) en (3) hierin dan vinden we een singulariteit op $\omega=0$ welke we kunnen elimineren door $\sum_m c_m = 0$ te maken [12] (m is eindig).

Tgv de karakteristiek van $N_I(\omega)$, is $\phi(\omega)$ begrensd in bandbreedte. Voor de verdere analyse kunnen we een DFT (Discrete Fourier Transformation) gebruiken met verwaarloosbare gevolgen voor de nauwkeurigheid.

$$F\{\cos\phi(t)\} = q_c(\omega)+jr_c(\omega)$$

$$F\{\sin\phi(t)\} = q_s(\omega)+jr_s(\omega)$$

$$|m(\omega)|^2 = \{q_c(\omega)-r_s(\omega)\}^2+\{q_s(\omega)+r_c(\omega)\}^2$$

$m(\omega)$ is het naar nul verschoven frekwentiespectrum van het fasegemoduleerde signaal.

$$m(t) = \cos(\omega_c t+\phi(t))$$

Figuur 1.2 geeft nu de op deze manier met de computer berekende spectra voor een reeks van 2^{10} bits.
In de figuur 1.2 zijn weergegeven de spectra voor QPSK, CORPSK(4-5) (zie volgende paragraaf), CORPSK(4-7,1+D) en TFM (Tamed Frequency Modulation). TFM valt ook onder de klasse van CORPSK [12] .
Voor CORPSK(4-5) heeft het premodulatie-filter een "roll off" factor [$\alpha = 0,5$] , voor de overige geldt [$\alpha=0$] .
Conclusie : De spectrumefficiëntie van CORPSK is beduidend beter dan die van QPSK.

1.7 Digitale Modulator

Een andere manier om een fasegemoduleerd signaal te verkrijgen is gebaseerd op de ontbinding van het signaal m(t) in zijn componenten.

$$m(t) = \cos(\omega_c t).\cos\phi(t)-\sin(\omega_c t).\sin\phi(t)$$

Alle mogelijke $\cos\phi(t)$ en $\sin\phi(t)$ of hun "truncated versions" worden opgeslagen in een geheugen. Het ingangssignaal wordt als adres voor dit geheugen gebruikt.
Een blokdiagram is weergegeven in fig. 1.3.

1.8 CORPSK(4-5)

Quaternary input en niet-lineaire codering.

In [1] wordt "phase-shaped 4-PSK" gepresenteerd.
De modulator bestaat uit een lineaire fasemodulator en een premodulatie-filter.
Een voorbeeld van een fasetraject van deze modulator wordt gegeven in fig. 1.4, de ononderbroken lijn.
De grote veranderingen tgv richtingsverandering in de fase kunnen worden vermeden als we een correlatie in de opeenvolgende faseveranderingen aanbrengen en deze veranderingen beperken tot plus en min π radialen of minder.
Dit is weergegeven door de onderbroken lijn in fig. 1.4 welke door dezelfde fasepunten gaat (modulo 2π) maar via een veel geleidelijker traject. De correlatie-encodering voor CORPSK(4-5) moet een quaternair ingangssignaal {0,1,2,3} vertalen in een symmetrisch uitgangssignaal met 5 niveau's {-2,-1,0,1,2}. De uitgangsniveau's moeten overeenkomen met faseveranderingen $\{-\pi, -\pi/2, 0, \pi/2, \pi\}$ en de codering moet ervoor zorgen dat het verschil tussen opvolgende faseveranderingen minimaal is. Hieraan is voldaan als geen opeenvolgende faseveranderingen van $\pm\pi$, $\mp\pi$ voorkomen.
De correlatiecodering is in tabel 1.5 weergegeven. De keuze +2 of -2 hangt af van de voorgeschiedenis, de nu volgende coderingsregel geeft het beste resultaat. Als $a_m=0$ dan is

sign c_m = sign$(c_{m-1}-c_{m-2})$ als $c_{m-1}-c_{m-2}=0$ dan is
sign c_m = sign$(c_{m-2}-c_{m-3})$.

Voor CORPSK(4-5) is geen precoding noodzakelijk, omdat er een eenduidigheid is tussen de symbolen (a_m) en de faseveranderingen (c_m).
Voor de "raised cosine" karakteristiek van het premodulatie-filter kiezen we een "roll of" faktor, $\alpha = \frac{1}{2}$ als goed compromis tussen de frekwentiespectrumbegrenzing en de grootte van de "intersymbol interference".

1.9 CORPSK-Demodulator

Om een CORPSK modulaat te verkrijgen zijn de volgende stappen doorlopen.
We gaan uit van een symbolenreeks met alfabet $\{a_1 \ldots a_q\}$.
Deze reeks wordt via een correlatiecodering omgezet in een symbolenreeks met alfabet $\{c_1 \ldots c_m\}$ $\quad m>q$

Vervolgens voeren we deze symbolenreeks toe aan een premodulatie-filter. We hebben nu een hoeveelheid "intersymbol interference" aangebracht, deze "intersymbol interference" is echter volledig bekend. Het nu verkregen signaal stuurt een lineaire fasemodulator aan waardoor een modulaat met constant omhullende ontstaat. Door deze stappen zijn opeenvolgende faseveranderingen gecorreleerd en is de informatie per symbool uitgesmeerd over meerdere symboolperioden.
Om deze over meer symboolperioden, uitgesmeerde informatie weer uit het modulaat te halen is een ontvanger nodig welke niet per symbool detekteert maar welke een symboolbeslissing maakt op grond van de over meerdere symboolperioden verkregen informatie.
Een optimale ontvanger voor CORPSK is een coherente detektor waarin het ingangssignaal van de ontvanger wordt gecorreleerd met replica's van het uitgezonden signaal [11] , [12] .
We kunnen twee implementaties onderscheiden :
1) Een "matched filter" ontvanger.
2) Een correlatie ontvanger.
Beide gevolgd door, of een beslissingscircuit volgens [11], er wordt dan over meerdere symboolperioden per symbool gecorreleerd, of een Viterbi processor volgens [13] , er wordt dan over één symboolperiode per symbool gecorreleerd, de Viterbi processor stelt deze correlatieresultaten efficiënt samen zodat correlatieresultaten over meerdere symboolperioden worden verkregen terwijl slechts een deel van de correlatieresultaten wordt verwerkt. De "matched filter" oplossing zal verder niet beschouwd worden om de volgende redenen :

- Het aantal "matched filters" is groot terwijl de impulsresponsies van alle filters verschillend zijn.

- Als voor banddoorlaat filters wordt gekozen, dan kunnen "Surface Acoustic Wave" (SAW-)filters worden toegepast, maar ieder I.C. filter vereist de ontwikkeling van een eigen masker, hetgeen alleen bij grote aantallen economisch aantrekkelijk is. Bovendien is dan een groot aantal "mixers" nodig en een even groot aantal laagdoorlaat filters.

- Na filteren en bemonsteren is nog een rekenkundige verwerking nodig, zodat elk filter door een AD-convertor moet worden gevolgd.

Voor de verdere beschrijving van de CORPSK-ontvanger zijn de volgende parameters van belang :

- p = aantal fasepunten (fasepunten zijn alle mogelijke waarden die de fase ten tijde $t=mT_s$ kan aannemen, zie definitie CORPSK).
- m = aantal monsters per symbool in de quadratuurcomponenten en de replica's.
- L = aantal symboolresponsies van het premodulatiefilter waaruit een replica is samengesteld.
- M = Aantal elementen uit het symbolenalfabet $\{c_m\}$.

In de ontvanger wordt in ieder symboolinterval het ontvangen signaal s(t) gecorreleerd met alle (pM^L) replica's $r_i(t)$ $(i=1,2 \ldots pM^L)$.
In de ontvanger zonder Viterbi processor zijn de replica's drie symboolintervallen lang (L=3).
In de ontvanger met Viterbi processor zijn de replica's één symboolinterval lang terwijl L de waarden 1,2 of 3 kan aannemen.
In de Viterbi processor wordt een beperkt aantal correlatieresultaten geselekteerd. Dit geselekteerde aantal wordt in een geheugen opgeslagen en wordt met de correlatieresultaten

van het volgende symboolinterval samengesteld. Zo wordt op efficiënte wijze een aantal correlatieresultaten over meerdere symboolintervallen verkregen. In hoofdstuk 2 wordt de werking van het Viterbi Algoritme nader toegelicht.

We zullen nu eerst de wijze van correleren nader uitwerken :

$$Z_{n,i} = \int_{nT}^{(n+1)T} s(t).r_i(t)dt$$

n geeft aan het n^e interval.
s(t) is het gezonden signaal.
$r_i(t)$ zijn de replica's.

$$s(t) = \sqrt{(2E/T)}\ \cos[\omega_o t+\Phi(t)] + n(t)$$

$$r_i(t) = \sqrt{(2E/T)}\ \cos[\omega_o t+\Phi_i(t)]$$

met n(t) als de toegevoegde ruis.

$$Z_{n,i} = \int_T s(t).r_i(t)dt$$

$$Z_{n,i} = (E/T) \int_T \cos[2\omega_o t+\Phi(t)+\Phi_i(t)]\,dt +$$

$$+ (E/T) \int_T \cos[\Phi(t)-\Phi_i(t)]\,dt + n'(t)$$

$\omega_o >> (2\pi/T)$ waardoor de eerste term te verwaarlozen is.

$$Z_{n,i} = (E/T) \int_T \cos[\Phi(t)-\Phi_i(t)]\,dt + n'(t)$$

$$Z_{n,i} = (E/T) \left[\int_T \cos\Phi(t)\cos\Phi_i(t)dt + \int_T \sin\Phi(t)\sin\Phi_i(t)dt\right] + n'(t)$$

Hieruit volgt dat de correlatie niet met replica's van het RF signaal hoeft te worden uitgevoerd, maar dat de gedemoduleerde quadratuurcomponenten $\sin\Phi$ en $\cos\Phi$ kunnen worden gecorreleerd met de replica's $\sin\Phi_i$ en $\cos\Phi_i$ waarna het resultaat moet worden gesommeerd. Dit betekent een verdubbeling van het aantal replica's tot $2.pM^L$. Dit aantal kan echter met een factor p worden gereduceerd door te stellen :

$$\Phi_i(t) = \Theta_k(t)+\Theta_j$$

fasetraject ≡ faseverandering + fasepunt

De faseverandering van 0^0 naar 90^0 is hetzelfde als de faseverandering van 180^0 naar 270^0, alleen het fasepunt van vertrek (of aankomst) verschilt.

Met verwaarlozen van de ruisterm krijgen we :

$$Z_{n,i} = \frac{E}{T}\cos\Theta_j \int_T [\cos\Phi(t)\cos\Theta_k(t)+\sin\Phi(t)\sin\Theta_k(t)]\,dt +$$

$$+ \frac{E}{T}\sin\Theta_j \int_T [\sin\Phi(t)\cos\Theta_k(t)-\cos\Phi(t)\sin\Theta_k(t)]\,dt$$

Alleen de replica's van de sinus en cosinus van de M^L faseveranderingen Θ_k dienen te worden opgeslagen en te worden vermenigvuldigd met de gedemoduleerde signalen $\sin\Phi(t)$ en $\cos\Phi(t)$.
De verzameling faseveranderingen kan in twee deelverzamelingen worden verdeeld welke symmetrisch rond nul liggen.
De berekeningen dienen voor slechts één deelverzameling te worden uitgevoerd.
De produkten met $\cos\Theta_k(t)$ zijn voor beide deelverzamelingen gelijk. De produkten met $\sin\Theta_k(t)$ moeten worden geïnverteerd om de resultaten voor de tweede deelverzameling te verkrijgen.
Dit reduceert het aantal replica's nogmaals met een factor 2 tot M^L.

De integraties worden bepaald door de monsters van, het ontvangen signaal en de replica's, te vermenigvuldigen en op te tellen. Het inverteren voor een bepaalde waarde van k kan na de optelling van m produkten geschieden (m is het aantal monsters per symboolperiode).
Het totaal aantal bewerkingen per symboolinterval bedraagt nu :

$(2m+p)M^L$ vermenigvuldigingen
$(2m+p)M^L$ optellingen
M^L inverteringen

Dit aantal is exclusief het Viterbi proces. Het aantal operaties hierin neemt toe met het aantal fasetrajecten dat we willen opslaan (L).

1.9.1 Conclusie

Vergelijken we de "matched filter" ontvanger met de correlatie-ontvanger, dan kunnen we concluderen, dat de eerste nogal veel filters en "custom-IC" ontwikkeling vraagt terwijl de correlatie-ontvanger in principe met standaard digitale IC's kan worden gebouwd. De "matched filter" ontvanger kan hogere "bitrates" verwerken omdat er geen correlatieresultaten behoeven te worden berekend.
Bij gebruik van een Viterbi processor wordt slechts gecorreleerd over één symboolperiode. Binnen de processor worden deze resultaten zo verwerkt dat samengestelde correlatieresultaten over meerdere symboolperioden ontstaan. Doordat we per symboolinterval maar een deel van de correlatieresultaten selekteren levert dit een reduktie van het aantal operaties op terwijl we toch correlatieresultaten over meerdere symboolintervallen verkrijgen. Hoe dit wordt bereikt wordt in het volgende hoofdstuk uitgelegd.

2. Het Viterbi Algoritme

2.1 Introductie

Bij CORPSK is de faseverandering gedurende een symboolinterval samengesteld uit de bijdrage van verscheidene opeenvolgende inputsymbolen, maw de informatie in één inputsymbool wordt uitgesmeerd over meer symboolintervallen in het fasetraject.
Als gevolg hiervan zijn achtereenvolgende faseveranderingen gecorreleerd. Voor CORPSK kunnen we, bij lineaire correlatie-encodering, voor de fasefunctie $\phi(t)$ schrijven :

$$\Phi(t) = 2\pi h \int_{-\infty}^{t} \sum_{m=-\infty}^{n} a_m g(\tau - mT_s)\, d\tau \qquad [14]$$

$$-\infty < t < (n+1)T_s$$

Met g(t) een impuls (met oppervlakte ½) die de momentane frekwentie voor een eenheidsinputsymbool is

$$g(t) = \frac{d\Phi_u(t)}{dt} \frac{1}{2\pi h}$$

en $\Phi_u(t)$ is het fasetraject tgv een eenheidsinputsymbool.
De pulsresponsie g(t) is een zogenaamde "partial-response" pulse, welke breder is dan één symboolinterval. Samenvattend, transformeert CORPSK inputsymbolen in symbolen met langere periodeduur en/of een gladder verloop, waaruit de fase op lineaire wijze wordt afgeleid.
h is de modulatie index gedefinieerd als het faseverschil in radialen, behorende bij de eenheidsstapresponsie van het premodulatiefilter, gedeeld door 2π radialen.

$h \hat{=} \Delta\phi_1/2\pi$

a_m is een symbool uit de inputsymbolenreeks.

Het Viterbi Algoritme is oorspronkelijk uitgevonden om convolutionele codes te decoderen [15] - [17] .
Later bleek het VA een kortste-route-algoritme te zijn welke reeds lang bekend was in de operationele research [16] - [23] , welke op zijn beurt weer als een variant van het dynamisch-programmeren kan worden gezien [22] - [24] .
De toepasbaarheid van het VA in "partial-response systems" is ontdekt door Omura en Kobayashi te U.C.L.A. [25] - [29] .
In zijn artikel stelt Forney [13] dat het VA een recursieve optimale oplossing is, voor het bepalen van een toestandsvolgorde van een tijddiskreet Markov proces met een eindig aantal toestanden.

De structuur van de ontvanger bestaat nu uit een coherente demodulator, een digitale correlator en een recursieve niet-lineaire processor, het Viterbi Algoritme, zie fig. 2.1.
Deze structuur vormt een "maximum likelihood estimator" voor de totale uitgezonden datareeks [32] .

Het Viterbi Algoritme heeft een recursieve structuur.
De complexiteit van de structuur is evenredig met (M^L).
M is het aantal elementen in het symbolenalfabet $\{c_m\}$ terwijl L het aantal symboolresponsies is waaruit een replica is samengesteld. In het Viterbi Algoritme vinden geen vermenigvuldigingen plaats. Wel vinden er circa pM^L optellingen en vergelijkingen plaats per ontvangen symbool. Van deze optellingen wordt een deel opgeslagen in een geheugen. Er zijn circa $10.p.L.M^{L-1}$ geheugenwoorden van $^2\log(pM^{L-1})$ bits nodig, voor data-opslag.

2.2 Het Viterbi Algoritme praktisch

Essentieel binnen het Viterbi Algoritme is de definitie van toestand. Deze definitie wordt in de volgende paragraaf gegeven. In het ontvangen signaal zijn in één symboolinterval oneindig veel fasetrajecten mogelijk.
Binnen de ontvanger maken we een benadering van de werkelijkheid door slechts een eindig aantal fasetrajecten binnen een symboolinterval toe te laten.
Deze fasetrajecten zijn de replica's welke we in een geheugen opgeslagen hebben.
Een replica is een fasetraject welke van de ene toestand naar de andere toestand (of dezelfde toestand) loopt.
Tussen twee toestanden is maximaal één fasetraject mogelijk. En naar één toestand zijn maximaal M fasetrajecten mogelijk.
Het aantal toestanden en het aantal replica's hangt samen.
Bij elke toestand horen M replica's. Het maximum correlatieresultaat van deze M replica's met het ontvangen signaal, vermeerderd met een historische correlatiewaarde wordt in een geheugen bewaard, de andere correlatieresultaten worden niet verwerkt. Zo vinden we bij elke toestand een maximum samengesteld correlatieresultaat.
Bij deze geselecteerde replica's per toestand hoort een symbool. Dit symbool wordt ook in een geheugen bewaard.
Ook de symbolen van vorige selecties zijn bewaard gebleven, zo vormen ze een aantal fasetrajecten over meerdere symboolperioden. Dit aantal is gelijk aan het aantal toestanden.
Van de nu geselecteerde, samengestelde correlatieresultaten wordt het maximum bepaald. Het bij dit maximum, in het geheugen opgeslagen fasetraject, (symbolensequentie), bevat nu het ontvangen symbool. Welk symbool uit de geselecteerde symbolensequentie dit is wordt bepaald door de lengte van het observatie-interval.

Samenvattend : Bij elke toestand T_i hoort een viertal registers (geheugens). De L_i en L'_i registers bevatten een samengesteld correlatieresultaat

De S_i en S_i' registers bevatten een fasetraject (symbolen-sequentie of toestandssequentie) van vrij te bepalen lengte, deze lengte noemen we het observatie-interval, N_{obs}. De accenten van L_i' en S_i' duiden op nieuw ontstane waarden tgv een toestandsovergang uit de waarden van L_i en S_i. Voor een hypothetisch stelsel zullen we nu de werking van het Viterbi Algoritme nogmaals in kortschrift toelichten. Stel een nieuwe toestand T_k' kan uit slechts twee oude toestanden T_i en T_j worden bereikt. Bij de overgangen van deze toestanden naar de nieuwe toestand horen de correlatieresultaten c_{ik} en c_{jk} en de symbolen s_{ik} en s_{jk}. Zie fig. 2.2.

In het volgende wordt in kortschrift het Viterbi Algoritme toegelicht. De achterliggende gedachte is de volgende : Als een maximum samengesteld correlatieresultaat bij een bijbehorend fasetraject is gevonden dan is met bepaalde waarschijnlijkheid te zeggen, door welke toestand het fasetraject gegaan moet zijn een N_{obs} aantal perioden geleden :

$\forall k$: betekent : voor alle toestanden.

In kortschrift werkt het Viterbi Algoritme als volgt :

1. $\forall k$: do

$d_i = c_{ik} + L_i$

$d_j = c_{jk} + L_j$

$d_x = \max(d_i, d_j)$ $(x = iV_j)$ selectie

$L_k' = d_x$ (correlatiesom)

$S_k' = S_x$ (data pad)

S_k' shift one (verlies oudste symbool)

S_k' set first = $s_{x,k}$ (nieuwste symbool)

2. $\forall k$: do $\quad L'_q = \max\{L'_k\}$ $\quad (q \in \{k\})$ selectie

Data Out : S'_q last $\quad (S_{q,k-p})$

3. $\forall k$: do $\quad L'_k \rightarrow L_k$ $\quad$ (updating)

$S'_k \rightarrow S_k$

Als $L_q > L_{max}$ dan $\forall k$, $\quad Lk = Lk - L_{max}$ (overflow begrenzing)

L'_q geeft het meest waarschijnlijke fasetraject, S'_q aan.

Het oudste symbool bevat in S'_q is het gezonden symbool een N_{obs} aantal perioden geleden.

2.3 Toestandsdefinitie

Het aantal toestanden (pM^{L-1}) is evenredig met het aantal replica's (circa $2pM^L$). Tussen twee toestanden is maximaal één replica-fasetraject mogelijk en naar één toestand zijn maximaal M replica-fasetrajecten mogelijk. Vanuit deze gegevens is de definitie van toestand te bepalen.
De eenvoudigst denkbare replica is een symboolresponsie {L=1} van het premodulatiefilter. Tussen twee fasepunten is nu slechts één replica-fasetraject mogelijk, derhalve komt de definitie van toestand overeen met een fasepunt.
De volgende stap is een replica opgebouwd uit de uit twee symboolresponsies samengestelde responsie van het premodulatiefilter {L=2}. Er zijn nu pM toestanden. De definitie van toestand wordt nu bepaald door de combinatie van het fasepunt Θ_j en het symbool s_{ik}. Zie fig. 2.3^b (nb $s_{ik} \in \{c \ldots c_m\}$).
De derde stap is een replica welke opgebouwd is uit de uit drie symboolresponsies samengestelde responsie van het premodulatiefilter {L=3}. Er zijn nu pM^2 toestanden. De definitie van toestand wordt nu bepaald door de combinatie van het fasepunt Θ_j en de symbolen s_{ik} en s_{ky}. Zie fig. 2.3^c. Voor alle drie de replica's geldt dat slechts één symboolperiode van de responsie in het geheugen wordt opgeslagen. (Ononderbroken lijn)

3. Ontwerp van een CORPSK(4-5) ontvanger voor Mobiele Radio

3.1 Introductie

Om een ontwerp te kunnen voorstellen is het van belang een afschatting te maken van de parameters welke invloed hebben op de complexiteit van de ontvanger.
De volgende parameters zullen worden onderzocht :

h = modulatie index ($h = \frac{\Delta\Phi_1}{\pi}$, $\Delta\Phi_1$ is de fasestap in de stapresponsie van het premodulatiefilter)

p = aantal fasepunten

m = aantal monsters per symbool in de quadratuur-componenten en de replica's

q = aantal quantisatie bits per monster

N_{obs} = lengte van het observatie-interval uitgedrukt in symboolperioden

L = correlatielengte uitgedrukt in het aantal symbolen waaruit de replica's zijn samengesteld

T = aantal toestanden binnen het Viterbi Algoritme

Invloed van h en p

h Bepaalt het aantal fasepunten p en de waarden van $\cos\Theta_j$ en $\sin\Theta_j$ (Θ_j zijn de fasepunten). Er is gekozen voor h=0,5, p=4 omdat dit de complexiteit van de ontvanger beperkt, $\cos\Theta_j$ en $\sin\Theta_j$ zijn dan immers 0 of 1 en bovendien is dan een vergelijking met QPSK mogelijk.

Invloed van m

Het aantal monsters per symboolinterval voor de replica's en het gedemoduleerde signaal is afhankelijk van de bandbreedte van de signalen $\sin\phi(t)$ en $\cos\phi(t)$. Bij de ontvanger is het analoge signaal met de toegevoegde ruis door de ontvangfilters in bandbreedte beperkt.

Voor wat betreft de zender is het informatiedragende signaal tgv de correlatiecodering en de zendfilters in bandbreedte beperkt.

Als f_{max} de hoogst significante frekwentie van het uitgezonden signaal is (en van de replica's in de ontvanger) dan is een bemonsterfrekwentie van $2f_{max}$ voldoende vermits de ruis ook door de ontvangfilters tot f_{max} is beperkt. Een belangrijke vraag is wat als hoogste significante frekwentie beschouwd moet worden, kiezen we bv de -60 dB bandbreedte dan is weinig degradatie te verwachten voor een bemonsterfrekwentie van $2f_{max}$.

Nb : "matched" filters hebben een frekwentiekarakteristiek welke afgeleid is van het ontvangen signaal hetgeen betekent dat spectrale componenten op -30 dB in het ontvangfilter nogmaals met 30 dB worden verzwakt.

Voor CORPSK(4-5) geldt dat voor f_{bit} de vermogensdichtheid circa -70 dB bedraagt en voor $\frac{1}{2}f_{bit}$ circa -20 dB. Zie fig. 2.3. Voor de ontvanger moet daarom met een m van 2,4 of 8 rekening worden gehouden.

(Nb : m is het aantal monsters per symbool

$f_s = \frac{1}{2}f_b \rightarrow 2f_b = 4f_s$)

m heeft alleen invloed op de complexiteit van de correlator en op de eisen die aan de A/D converter worden gesteld. Voor mobiele radio gelden bitrates van maximaal 72 Kbit/s. Voor een quaternair signaal betekent dit een symboolperiodeduur van 27,8 µs. Bij m=4 betekent dit een maximale conversietijd van 6 µs voor de AD-converter.

(TRW TDC 1002J 8 bit conversietijd 1 µs)

TRW levert ook de snelste multiplier-accumulators die op dit moment verkrijgbaar zijn.

TRW TDS 1008J 8 bit "double precision", "speed" 70 ns.

Per multiplier-accumulator kunnen dan voor m=4, maximaal 96 produkt-sommen worden verwerkt.

Invloed van q

Het aantal kwantisatiebits voor de monsters van het signaal en de replica's willen we zo laag mogelijk kiezen opdat de operanden zo klein mogelijk zijn.
Dit heeft een hogere verwerkingssnelheid en een kleiner aantal IC's voor de rekenkundige verwerking tot gevolg.
In [14] wordt aangetoond dat kwantisatie van 8 bits voor $\sin\Phi$ en $\cos\Phi$ en een bemonsterfrekwentie van 2 f_{bit} een kwantisatieruisvermogen per bitrate-bandbreedte van -56 dB tov het totale signaalvermogen bedraagt.
q waarden van 4,6 en 8 bits zijn interessant om te onderzoeken. De invloed van q op de complexiteit is niet makkelijk aan te geven. Voor vermenigvuldigingen is de verwerkingssnelheid ongeveer omgekeerd evenredig met het aantal bits terwijl het aantal poorten toeneemt met de wortel uit q [14] .

Invloed van N_{obs}

De lengte van het observatie-interval heeft geen invloed op het aantal rekenkundige en logische bewerkingen maar wel op de capaciteit van het geheugen voor data-opslag in de Viterbi processor. De grootte van het geheugen is evenredig met de lengte van het observatie-interval. N_{obs} heeft echter geen grote invloed op de complexiteit van de ontvanger.

Invloed van L en T

De correlatielengte L, het aantal toestanden T (pM^{L-1}) en het aantal replica's (circa $2pM^{L}$) hangen samen.
L heeft een exponentiële invloed op de complexiteit van de ontvanger.
De waarden van L hoeven aan de kant van de zender en aan de kant van de ontvanger niet overeen te komen. Aan de zenderzijde zijn waarden voor L van 3,5 en 7 van belang [30] .
Aan de ontvangzijde zijn waarden van L > 3 niet aanvaardbaar omdat ze de complexiteit van de ontvanger dan te groot maken. Voor CORPSK(4-5) geldt p=4 M=5. Voor L=3 komen we dan op 1000 replica's waarvan er (M^{L}=)125 in een geheugen

dienen te worden opgeslagen. Tgv de correlatiecodering vallen nog een aantal replica's af zodat per symboolinterval het totaal aantal te verwerken replica's 96 bedraagt, 48 voor de $\cos\Phi_i(t)$ en 48 voor de $\sin\Phi_i(t)$.

Er kan dan met één TRW multiplier-accumulator per 48 produkt-sommen worden volstaan (zie ook, Invloed van m). In de Viterbi processor dienen p.(96) vergelijkingen per symboolinterval te worden uitgevoerd (voor L=3 p=4 M=5). Dit zijn 384 vergelijkingen. Dit is met één comparator te doen, N74S85N (4 bit expandable comparingtime 17 ns).

3.2 Ontwerpoverwegingen

Met bovenstaande in gedachten dient het ontwerp een "pipelining" structuur te bezitten zodat voor elk onderdeel een verwerkingscyclus van één symboolperiode beschikbaar is. Volgens dit "pipelining" principe is de ontvanger dan in drie delen op te splitsen :

1) Coherente demodulator met signaalbemonstering.
2) Digitale Correlator.
3) Viterbi processor.

Tussen elk deel zal een buffergeheugen nodig zijn voor data-opslag.

Nb: de carrier- en klokregeneratie blijven buiten beschouwing

3.3 Beschrijving van het ontwerp

3.3.1 Coherente demodulatie en bemonstering

In fig. 3.1 zijn de eerste twee delen in detail weergegeven. Het eerste deel, de coherente demodulator en de analoog-digitaal omzetter, behoeft geen nadere toelichting. Per symboolperiode worden de monsters in een dubbel uitgevoerd buffergeheugen opgeslagen. Het eerste deel van het buffergeheugen dient voor data-opslag, het tweede deel voor uitlezen van de data welke in de voorgaande periode is opgeslagen.

Na iedere symboolperiode vindt omschakeling van de twee buffergeheugens plaats, hierdoor ontstaat het gewenste "pipelining" effekt.

3.3.2 De digitale correlator

De data-toevoer voor de digitale correlator wordt verzorgd door het buffergeheugen en een ROM waarin de monsters van de replica's zijn opgeslagen. Voor p=4 M=5 en L=3 zijn er 48 replica's voor $\cos\Phi_i(t)$ en 48 replica's voor $\sin\Phi_i(t)$.

$$Z_{n,i} = \cos\theta_j \left[\Sigma\cos\phi(t)\cos\theta_k(t) \pm \Sigma\sin\phi(t)\sin\theta_k(t) \right] + \\ + \sin\theta_j \left[\Sigma\sin\phi(t)\cos\theta_k(t) \mp \Sigma\cos\phi(t)\sin\theta_K(t) \right]$$

$$Z_{n,i} = \{\pm 1, 0\}\left[\Sigma ac \pm \Sigma bd \right] + \{0, \pm 1\}\left[\Sigma bc \mp \Sigma ad \right]$$

Er dienen twee produktsommen te worden bepaald per replica ipv vier, omdat $|\cos\theta_j| = 0$ als $|\sin\theta_j| = 1$ en andersom. Het totaal aantal te verwerken produktsommen is 4x48 hetgeen op 4 multiplier-accumulators duidt. (1 multiplier-accumulator kan maximaal 96 produktsommen verwerken als het aantal monsters m=4 per symbool bedraagt). Voor de volledigheid zijn de vermenigvuldigers voor vermenigvuldiging met $\cos\theta_j$ en $\sin\theta_j$ ook getekend maar omdat $\cos\theta_j$ en $\sin\theta_j$ of ± 1 of 0 zijn, zijn deze vermenigvuldigers voor CORPSK(4-5) niet nodig.
De resultaten van de correlator worden wederom in een dubbel uitgevoerd buffergeheugen opgeslagen.
Het aantal op te bergen gegevens bedraagt 4x48 = 192. De overige 192 resultaten vinden we door de eerste 192 te inverteren omdat $\cos\theta_j$ en $\sin\theta_j$ in het derde en vierde kwadrant 0 of -1 zijn.

3.3.3 Viterbi processor

3.3.3.1 Introductie

De Viterbi processor is in twee delen te splitsen.

In deel 1 worden de circa pM^L optellingen en vergelijkingen verricht terwijl in het tweede gedeelte de dataverwerking en opslag plaats vindt.

Samenvattend, deel 1 berekent en selecteert de informatie welke door deel 2 verwerkt wordt met als uiteindelijk resultaat een beslissing over het gezonden symbool. Eventueel kan tussen deze twee delen weer een dubbel buffergeheugen worden geplaatst opdat elk deel een verwerkingscyclus ter lengte van een symboolperiode heeft, dit is in principe echter niet nodig omdat de verwerkingscyclus van de dataverwerking een additionele tijd verbruikt van N_{obs}x90ns (Accesstime van, RAM plus een Register) hetgeen klein is tov de verwerkingstijd van deel 1 (circa pM^Lx35ns).

We noemen deel 1 de replicabeslissingsschakeling en deel 2 de symboolbeslissingsschakeling.

3.3.3.2 De replicabeslissingsschakeling

In fig. 3.2 is fig. 3.1 nogmaals getekend met nu ook het eerste deel van de Viterbi processor in detail getekend. Buffer 1 bevat de correlatieresultaten c_{ij} van de digitale correlator. Buffer 2 bevat de samengestelde correlatieresultaten de zgn registers L_i uit het Viterbi Algoritme. De buffers 1 en 2 worden aangestuurd door de ROM's 1 en 2 zodat in de gewenste volgorde de resultaten L_i+c_{ij} ontstaan welke worden toegevoerd aan register A. Register B selecteert uit de waarden die in register A verschijnen het maximum. Van deze maxima in register B, selecteert register C het maximum. Register B bevat dus telkens de nieuwe samengestelde correlatieresultaten L_i', welke worden gecopieerd in buffer 3.

Na elke toestand T_i, hetgeen aangegeven wordt door ROM1 of ROM2, wordt register B op een maximaal negatieve waarde ingesteld. Na elk symboolinterval wordt register C ook maximaal negatief ingesteld.
Buffer 3 bevat nu de nieuwe samengestelde correlatie-resultaten, de zgn L_i' registers uit het VA en register C bevat het maximum uit L_i', is dit maximum groter dan L_{max} dan worden alle registers L_i' met de waarde L_{max} verminderd en in Buffer 2 gecopieerd, is de inhoud van register C kleiner dan L_{max} dan wordt Buffer 3 zonder meer in Buffer 2 gecopieerd.
De registers A, B en C bestaan uit drie delen.
De Reg A^o, B^o en C^o bevatten correlatieresultaten.
De Reg A^1, B^1 en C^1 bevatten een bijbehorend symbool.
De Reg A^2, B^2 en C^2 bevatten het toestandsnummer van de toestand van herkomst van bijbehorende replica.
Voor p=4 M=5 en L=3, duurt het copiëren van Buffer 3 naar Buffer 2 circa pM^{L-1}x45ns, pM^{L-1} is het aantal registers L' en 45ns is de "accesstime" van een RAM pM^{L-1}x45 ≡ 4,5 µs. Voor pM^L replica's betekent dit een verwerkingsperiode van (27,6 µs - 4,5 µs)/pM^L 40 ns.
De ROM's hebben een accesstime van 70 ns.
Om toch de vereiste snelheid te halen moet een aantal componenten in de replicabeslissingsschakeling dubbel worden uitgevoerd. In fig. 3.2 is dit door de schaduwwerking aangegeven (markeringsrand).

3.3.3.3 De symboolbeslissingsschakeling

De S-registers binnen het VA bevatten een symbolenreeks. Tijdens elke symboolperiode wordt aan elk S-register een nieuw symbool toegevoegd en verwisselen de registers van volgorde. Zie fig. 3.5. De registers bevatten een datasequentie welke overeenkomt met een bepaald fasetraject. Als we nu de datasequentie op dezelfde manier in het geheugen opslaan, als het fasetraject door de

toestanden verloopt dan is een efficiënte geheugenorganisatie mogelijk, zie fig. 3.3. Behalve het symbool moet dan ook de toestand (geheugenplaats) van herkomst opgeslagen worden. Zie fig. 3.6. Deze methode van geheugenorganisatie is mogelijk omdat als, teruggaande in de tijd, twee fasetrajecten samenvallen, ze teruggaande in de tijd zullen blijven samenvallen, omdat een toestand maar één toestand van herkomst kan bezitten (zie def. van toestand).
De plaats van toevoeging van een nieuw symbool wordt aangegeven door een kolompointer. De toestand van herkomst vormt nu een adres van een geheugenlocatie. Zo'n adres is opgebouwd uit een rijpointer en een kolompointer. De rijpointer geeft de toestand aan. (Er zijn evenveel S-registers als toestanden) en de kolompointer geeft aan met welk symbool we tussen nieuwste en oudste symbool te maken hebben. Het opslaan van de toestand van herkomst T_i is nu de richtlijn waarlangs het oudste symbool stap voor stap wordt teruggevonden.

Fig. 3.7 geeft de hardware-realisatie van de symboolbeslissingsschakeling.
Per symbool komen binnen, voor elke toestand; de toestand van herkomst plus het bijbehorende symbool (dit is de inhoud van Reg B van de replicabeslissingsschakeling). Waar deze gegevens worden opgeborgen in het geheugen wordt aangegeven door een toestandteller via D_3, de toestanden worden sequentieel afgewerkt door de replicabeslissingsschakeling (1 .... T).
D_3 vormt de rijpointer voor de geheugenlocatie. De kolompointer is eveneens een teller welke via D_1 wordt toegevoerd. D_1 en D_3 vormen een adres van een geheugenelement waarin de toestand van herkomst en het bijbehorende symbool (Inh Reg B) worden opgeslagen. Nb : de toestand van herkomst komt overeen met de rijïnformatie, de kolominformatie wordt afgeleid van D_1, via D_2. Dit is de vorige kolom.

Zijn alle Registers S bijgewerkt dan bevat Register C van de replicabeslissingsschakeling de startinformatie (Toestand van herkomst) voor de terugzoekcyclus. Dit Register C bevat de rijïnformatie, de kolominformatie betrekken we van D_2. D_4 bevat nu het startgeheugenadres welke de RAM aanstuurt, de output van de RAM is nu het volgende geheugenadres. Zo wordt de RAM N_{obs} maal uitgelezen (lengte observatie-interval = N_{obs}).

Daar waar een asterix is aangegeven zijn de outputs "tristate" zodat de juiste informatie verschijnt op de RAM output- en adreslijnen.
De laatst uitgelezen geheugenlocatie bevat nu het gedetecteerde symbool welke via D_5 wordt uitgelezen. De symboolbeslissingsschakeling werkt parallel aan de replicabeslissingsschakeling. Additionele tijd is nodig voor de terugzoekcyclus. Deze tijdsduur bedraagt N_{obs}x90ns $\sim$ 3 µs, 90 ns is de RAM accesstime vermeerderd met de Delaytime van een D-fl.fl.

3.4 Conclusie

Voor een bitrate van maximaal 72 Kbit/s is de ontvanger met standaard TTL bouwstenen te bouwen. (Als we de multiplier-accumulators als standaard beschouwen). Voor p=4, M=5, m=4, L=3 is de replicabeslissingsschakeling de tijdkritische schakeling, parallel processing is dan reeds gedeeltelijk noodzakelijk zoals aangegeven. Voor p=4, M=5, m=8, L=2 is de digitale correlator het meest tijdkritisch. Er kunnen dan 48 produktsommen per multiplier-accumulator worden verwerkt terwijl het aantal te verwerken replica's 24 bedraagt.
Er zijn dan 2 multiplier-accumulators nodig. In de replicabeslissingsschakeling is dan geen parallel processing nodig. Voor p=4, M=5, m=4, L=2 zou met 1 multiplier-accumulator kunnen worden volstaan.

Het hier gepresenteerde ontwerp is een principe-ontwerp zonder verdere detaillering.
De timingsschakeling ontbreekt evenals bv de tellers voor de aansturing van de ROM's.
Beschrijving van deze additionele logica zou alleen maar verwarrend werken en wordt als onnodig gezien omdat het een recht toe recht aan ontwerp is. Het ontwerp van deze additionele logica moet echter niet onderschat worden, omdat vooral de sturing van de dubbele buffergeheugens en de stuurpulsen voor de replicabeslissingsschakeling en de symboolbeslissingsschakeling niet eenvoudig zal zijn.
Het valt in dit verband aan te bevelen met rationele deelfrekwenties van één moederklok te werken zonder analoog vertraagde klokken.
Testen kan dan plaats vinden voor lagere bitrates dan 72 Kbit/s zodat de tijdkritische onderdelen van de schakeling niet meer tijdkritisch zijn.

4. De simulatieresultaten

4.1 Introductie

De simulatie is geïmplementeerd op een VAX-minicomputer. De programmeertaal is Pascal. In de appendix "Pascal" is beknopt de informatie over Pascal en het systeem gegeven welke voor een begrip van de ontwikkelde programmatuur nodig is. In de appendix "De simulatie" wordt een algemene beschrijving van de programmatuur gegeven terwijl in de appendix "Procedures" een nadere toelichting op de gebruikte formules wordt gegeven.

De simulatie bestaat uit een drietal modulen :

- Het eerste moduul bevat de zendersimulatie. De belangrijkste in te voeren parameters zijn : Het aantal symbolen, het aantal monsters per symbool, het aantal bits per symbool en de "roll of factor" van de "raised cosine" karakteristiek van het Nyquist-III premodulatiefilter.
  De zender genereert een "random" bitreeks waaruit een symbolenreeks wordt gevormd. Vervolgens wordt hieruit volgens het CORPSK(4-5) principe de quadratuurcomponenten $\cos\phi_i(nT_s)$ en $\sin\phi_i(nT_s)$ gevormd.

- Het tweede moduul bevat een ontvangersimulatie volgens [11] . Deze simulatie geeft resultaten van een optimale ontvanger zonder Viterbi processor.
  Er vindt correlatie met alle replica's over drie symboolperioden plaats. De correlatieresultaten van replica's met gelijk middelste symbool worden via een e-macht gesommeerd. $(\sum_{ij} \exp(2c_{ij}/N_o))$.
  Het maximum van deze sommaties bepaalt het gedetekteerde symbool (zie [11]). De in te lezen parameters zijn dezelfde als voor het derde moduul. In het tweede moduul worden de kwantisatieparameters niet gebruikt.

- Het derde moduul bevat een simulatie van een CORPSK(4-5) ontvanger met een Viterbi processor.
  De belangrijkste in te voeren parameters zijn :
  De parameters voor het ontvangfilter, de kwantisatie parameters, de ruisparameters, het aantal monsters per symboolinterval voor de replica's en de quadratuurcomponenten $\cos\phi(mT_s)$ en $\sin\phi(mT_s)$, het aantal fasepunten (p) en het aantal symbolen waaruit een replicaresponsie is samengesteld (L).

4.2 Simulatieresultaten algemeen

Voor de grafiek zijn een aantal parameters zo gekozen dat zij geen degradatie van de ontvangerprestaties opleveren. Dit zijn :

1) Het aantal kwantisatiebits voor de quadratuurcomponenten C-KBSC = 6 (C-XXXX zijn variabelen, gebruikt in de software).
2) Het aantal kwantisatiebits, na de digitale correlator en in de Viterbi processor C-KBSL = 12.
3) Het aantal monsters per symboolinterval voor de replica's en de quadratuurcomponenten $\cos\phi(mT_s)$ en $\sin\phi(mT_s)$ C-SPSO = 8.
4) De resultaten voor een observatie-interval van 1 tot 52 zijn steeds bepaald. De lengte van het observatie-interval heeft geen grote invloed op de rekentijd. Dat observatie-interval met de beste resultaten wordt steeds als resultaat voor de grafiek gekozen. N_{obs} is dan circa 10.L

In de grafiek 1 zijn de volgende krommen weergegeven.

1) De foutenkromme voor ideale DQPSK (aan de literatuur ontleend).
2) De foutenkromme voor een CORPSK(4-5) ontvanger zonder Viterbi processor volgens [11] met L=3.

3) De foutenkrommen voor een CORPSK(4-5) ontvanger met Viterbi processor voor L = 1,2 en 3.

De replica's kunnen uit 1, 2 of 3 symboolresponsies zijn samengesteld. Voor de ontvanger zonder Viterbi processor (moduul twee grafiekkromme 2) zijn de replica's uit drie symboolresponsies samengesteld (L=3) en de lengte van de replica's bedraagt ook drie symboolperioden. Er vindt dus correlatie over drie symboolperioden plaats per symboolinterval. Voor de ontvanger met Viterbi processor (moduul drie grafiekkrommen 3) zijn de replica's uit één, twee of drie symboolresponsies samengesteld (L=1,2 of 3). De lengte van de replica's bedraagt hier echter, ongeacht de waarde van L, altijd één symboolperiode.
Voor L=3 is dit de middelste periode van de uit drie symboolresponsies samengestelde responsie.
Voor L=5 is geen noemenswaardige verbetering van de prestaties van de ontvanger met Viterbi processor te verwachten, omdat de buitenste twee symboolresponsies een minimale invloed zullen hebben op het middelste deel van de samengestelde responsie.

4.3 Simulatieresultaten voor het "breadboard" model

Het aantal IC's voor een "breadboard" model wordt o.a. bepaald door de te kiezen kwantisering. Voor een minimale configuratie is het dan noodzakelijk acht kwantiseringsbits na de digitale correlator en in de Viterbi processor te kiezen. Het aantal kwantiseringsbits in de quadratuurcomponenten als het aantal monsters per symbool vier bedraagt is dan maximaal vier. SPSO=4, KBSC=4, KBSL=4, KBSL $\geq$ KBSC + $^{2}\log(4.\text{SPSO})$, zie appendix "Procedures" (procedure startwaarden).
De degradatie in prestatie van de ontvanger tgv deze kwantisatie is voor elk van de drie krommen 3 op één punt bepaald en bedraagt circa 0,5 dB (GSNR = 5 dB).
Er treedt geen degradatie op tgv het terugbrengen van het aantal monsters per symbool van 8 naar 4.

De ontvanger met Viterbi processor en L=3 geeft de beste prestaties. Zoals in het hoofdstuk 3 is beschreven heeft dit een eis tav de ROM "accesstime" in de "replicabeslissingsschakeling" tot gevolg (40 ns). De ROM's 82S115 hebben een accesstime van 70 ns. Voor een minimale configuratie (geen parallelprocessing) moeten we dan gebruik maken van snellere ROM's (ECL) of een ontvanger met Viterbi processor met L=2 kiezen. Kiezen we voor dit laatste dan is de degradatie in prestatie tov de ontvanger met Viterbi processor en L=3 circa 1 dB.

4.4 Conclusies

- De CORPSK(4-5) ontvanger met Viterbi processor (L=3, p=4) voldoet aan de verwachting dat de prestaties die van DQPSK voor wat betreft de foutenkromme moet benaderen, de kromme voor CORPSK(4-5) ligt circa 0,3 dB lager. Bovendien is het uitgezonden spectrum van CORPSK(4-5) veel smaller, dan dat van ongefilterde DQPSK.

- Voor een "breadboard" model kunnen we met een minimum configuratie volstaan voor "bitrates" lager of gelijk aan 72 KBit/s (mobiele radio) opgebouwd uit standaard TTL digitale IC's. Voor een ontvanger met Viterbi processor (L=3, p=4) zijn PROM's nodig met een "accesstime" < 40 ns (ECL) voor een minimum configuratie.

- Tov de ontvanger zonder Viterbi processor (L=3) zijn de prestaties van de ontvanger met Viterbi processor (L=3) circa 0,75 dB beter. Dit is ook te verwachten omdat de eerste ontvanger over drie perioden correleert terwijl de Viterbi processor correlatieresultaten over één periode zo samenstelt dat correlatieresultaten over een in principe oneindig aantal perioden (in het verleden) worden verkregen.

- De lengte van het observatie-interval blijkt een niet al te grote invloed op de resultaten te hebben, praktische waarden voor N_{obs} blijken in de orde van 10.L te liggen.

- Een kwantisatie van 8 bits in de Viterbi processor en 4 bits in de quadratuurcomponenten en de replica's heeft een degradatie van 0,5 dB tov de krommen in grafiek 1 tot gevolg.

- De gegevens voor de replica's vinden we in matrix BB. In de output file Pout.dat worden deze gegevens afgedrukt.

- De gegevens voor de ROM2 vinden we in matrix TT, de toestanden van herkomst.Deze matrix TT wordt in de output file Pout.dat afgedrukt als de variabele C-FOLT in de input file Pinp.dat "Yes" is.

- De gegevens voor de ROM1 vinden we in de matrix XX, de toestanden van aankomst. Deze matrix XX wordt in de output file Pout.dat afgedrukt als de variabele C-FOLT in de input file Pinp.dat "Yes" is.

- Voor het ontvangfilter is een 10^e orde Butterworth filter gebruikt met een $2f_{bit}$ bandbreedte.
  Terugbrengen van de bandbreedte tot f_{bit} is niet diepgaand onderzocht. De variabele MRGE geeft de ruismarge ten aanzien van maximaal signaalniveau en maximale kwantisatiewaarde aan. Deze waarde is 1,5 gekozen. Uit de literatuur [31] vinden we dat voor grove kwantisatie, deze ruismarge kleiner dan 1 moet worden. Dit is niet nader onderzocht.

- Het aantal monsters per symbool in de kwadratuurkomponenten en de replica's kiezen we 4, dit geeft geen noemenswaardige degradatie tov de simulatieresultaten uit grafiek 1.

5. Samenvatting

5.1 CORPSK-principe

CORPSK is een digitale modulatietechniek.
Deze modulatietechniek wordt gebruikt om informatie via een transmissiekanaal van een zender naar een ontvanger te transporteren op een te kiezen plaats in het frekwentie-spectrum (FDM).
De informatie wordt overgedragen in de vorm van een symbolen-reeks, vandaar de benaming "digitale modulatietechniek".
Deze symbolenreeks wordt dmv een correlatiecodering en een premodulatiefilter omgezet in een analoog signaal. (Een geleidelijk en vloeiend verlopend signaal zonder abrupte overgangen). Deze "partial response" techniek smeert de informatie van één symbool uit over meerdere symboolperioden.
Het analoge signaal stuurt een fasemodulator aan. Het resultaat is dat de informatie zich bevindt in een smal frekwentiespectrum, rond de draaggolf, met een sterke "roll of" van dit frekwentiespectrum en zonder spectrale componenten buiten dit frekwentiespectrum.
Tgv de fasemodulator bevat het modulaat geen amplitudemodulatie. Het modulaat heeft een zgn "constant omhullende".
Om deze over meer symboolperioden, uitgesmeerde informatie weer uit het modulaat te halen is een complexe ontvanger nodig welke niet per symbool detecteert maar welke een symboolbeslissing maakt op grond van de over meerdere symboolperioden verkregen informatie.

5.2 CORPSK-toepassingsmogelijkheden

De toepasbaarheid van CORPSK ligt besloten in z'n eigenschappen die als volgt zijn samen te vatten :

- Gebruik van verzadigde eindversterkers tgv een "constant omhullende",
- Geen spectrumverbreding hierdoor van een reeds smal spectrum,

- Beperkt gebruik van filters en een modulaat zonder amplitudevariatie,
- Lagere distorsie als gevolg hiervan,
- Complexe ontvanger.

Tgv de complexiteit van de ontvanger moet gedacht worden aan professionele toepassingen zoals bv Satelliet-communicatie, straalverbindingen en digitale communicatie over coaxiale kabels waarin boven de, voor coaxiale kabels normale band, op FDM wijze informatie wordt getransporteerd. CORPSK(4-5) is te vergelijken met DQPSK. De voornaamste conclusies zijn dan : CORPSK(4-5) heeft een beduidend betere spectrumefficiëntie, de vermogensefficiëntie en de prestaties van de ontvanger (B.E.R.) liggen in dezelfde grootteorde terwijl de ontvanger voor CORPSK(4-5) complexer is dan die voor DQPSK.

Dat de complexiteit voor consumententoepassing een belemmering is komt door het kostenaspect. Dit aspect is te elimineren omdat de ontvanger in principe een digitale schakeling is welke als geïntegreerd circuit uit te voeren is. Vooral voor Mobiele Radio is dit een zeer aantrekkelijk aspect van CORPSK (bitrate $\leq$ 72 Kbit/s) omdat geen parallel processing noodzakelijk zal zijn voor deze lagere bitsnelheden.

Voor Satelliet-communicatie moet aan hogere bitsnelheden worden gedacht. Huidige studies [14] tonen aan dat bitsnelheden van 2-8 Mbit/s met de huidige technologie mogelijk zijn en dat voor de toekomst verwacht wordt dat 25 Mbit/s haalbaar zal zijn.
Andere toepassingsmogelijkheden vinden we op het gebied van digitale audio voor radio en/of televisie. Door minder hoge eisen aan de prestaties van de ontvanger te stellen behoeft deze minder complex te zijn (niet coherente ontvanger en Viterbi processor met L=1) terwijl de andere eigenschappen zoals spectrumefficiëntie gehandhaafd blijven.

5.3 CORPSK toekomstverwachtingen

In dit verslag zijn de simulatieresultaten voor een CORPSK(4-5) zender-ontvanger systeem gepresenteerd. Om goede prestaties te behalen in de ontvanger bij een correlatie-interval van één symboolperiode is gebruik gemaakt van een Viterbi processor. Het totale systeem is digitaal uit te voeren en dus integreerbaar. Als eenmaal zo'n "IC" is gerealiseerd dan is de verwachting dat er een enorme markt voor Digitale Mobiele Radio Netten zal zijn (circa 1990).

Vanwege de spectrumefficiëntie en vermogensefficiëntie eigenschappen van CORPSK is de ESA (European Space Agency) geïnteresseerd in CORPSK en in het algemeen de klasse van CPM (Correlatieve Phase Modulatie methoden). Dit uit zich in het verlenen van studie-opdrachten aan Philips voor onderzoek naar de mogelijkheden en prestaties (laboratorium-opstellingen) van CPM principes zodat ook de verwachtingen voor CORPSK op het gebied van Satelliet-communicatie hoog gespannen zijn.

Zijn eenmaal professionele realisaties op genoemde twee gebieden gerealiseerd dan zal het toepassingsgebied van CORPSK zich ook over de andere genoemde gebieden uitspreiden. Al met al mag van CORPSK nog een hoop in de toekomst verwacht worden, dit zal echter niet alleen van de eigenschappen van CORPSK alleen afhangen doch ook van het tijdstip waarop voornoemde IC-realisatie en studie-opdrachten gerealiseerd zullen zijn, omdat ook in andere laboratoria de research naar "partial response" technieken niet stilstaat. En in het verleden zijn voorbeelden te vinden waarop de betere technieken te laat gepresenteerd werden waardoor minder goede technieken al gemeengoed geworden waren.

5.4 Conclusie tav de CORPSK(4-5)-ontvanger

Bij een keuze tussen "matched" filters en een digitale correlator heeft de laatste het voordeel dat één correlator resultaten voor alle replica's berekent, door de replica's sequentieel aan de correlator toe te voeren. Bij gebruik van "matched" filters zal voor elke replica een apart "matched" filter moeten worden ontworpen (parallel processing).
Mbv een Viterbi processor kunnen we een correlatieresultaat over een bepaald interval bepalen door efficiënt gebruik te maken van correlatieresultaten per deelinterval.
De ontvanger zal daarom een digitale correlator en een Viterbi processor bevatten.
In zijn artikel bewijst Forney [13] dat het Viterbi Algoritme met een observatie-interval (N_{obs}) dezelfde resultaten oplevert als een direkte correlatie over N_{obs} symboolintervallen.

5.5 Conclusie tav het ontwerp voor de ontvanger

Het ontwerp bevat een "pipelining" structuur, dwz dat voor elk onderdeel een verwerkingstijd van één symboolinterval beschikbaar is. De overdracht van gegevens geschiedt dmv omschakelende buffergeheugens. Het ontwerp bestaat uit drie delen : 1) De coherente demodulator met de signaalbemonstering, 2) De digitale correlator en 3) de Viterbi processor.

De ontvanger is voor een bitrate van 72 kBit/s met standaard TTL bouwstenen te bouwen. Gedeeltelijke parallel processing is nodig voor een Viterbi processor met L=3 tenzij er ROM's gebruikt worden met een accesstime < 40 ns.
Voor de correlator wordt een multiplier-accumulator van TRW (de TDC 1008J 8-bit double precision, speed 70 ns) aanbevolen. Voor L=3 bevat de demodulator vier van deze multiplier-accumulators, welke per stuk 96x4 vermenigvuldiging in één symboolinterval kunnen verwerken.

5.6 Conclusies tav de simulatie van de ontvanger

Voor de algemene simulatieresultaten zijn een aantal parameters zo gekozen dat zij geen invloed hebben op de prestaties van de ontvanger. Dit zijn :

a) De kwantisatieparameters, 6 bits voor de quadratuurcomponenten en 12 bits na de digitale correlator en binnen de Viterbi processor.
b) Het aantal monsters per symboolinterval in de quadratuurcomponenten en de replica's. Dit aantal is 8.
c) De lengte van het observatie-interval, steeds zijn de resultaten bepaald voor een observatie-interval van 1 t/m 52 symboolintervallen, de beste resultaten zijn steeds gekozen. N_{obs} is dan circa 10.L.

Grafiek 1 geeft de resultaten voor DQPSK en voor CORPSK(4-5) zonder Viterbi processor en met Viterbi processor (L=1,2 en 3).
Voor een kwantisatie van 4 bits in de quadratuurcomponenten en 8 bits na de digitale correlator en in de Viterbi processor moet met een degradatie van circa 0,5 dB rekening gehouden worden.
Terugbrengen van het aantal monsters in de replica's en de quadratuurcomponenten tot 4 heeft nagenoeg geen invloed.
De beste resultaten worden verkregen met een Viterbi processor met L=3, de ontvangercomplexiteit tov L=2 of L=1 is dan niet noemenswaardig groter zodat een ontvanger met Viterbi processor en L=3 wordt aanbevolen.
Bij realisatie van een discrete schakeling en gebruik van PROM's N82S115N "accesstime" 70 ns is voor een ontvanger met Viterbi processor en L=3 parallel processing noodzakelijk.
Voor een discrete schakeling wordt daarom een ontvanger met Viterbi processor en L=2 aanbevolen, er kan dan met één multiplier-accumulator worden volstaan, omdat het aantal bewerkingen (replica's) dan met een faktor 4 wordt gereduceerd. Als voor een discrete schakeling de bitrate tot 16 Kbit/s wordt teruggebracht kan voor L=3 ook met één multiplier-accumulator worden volstaan.

5.7 Afsluiting

In dit verslag is een minimum configuratie ontwerp voor een CORPSK(4-5) ontvanger met Viterbi processor voorgesteld. Mbv een simulatie is bepaald wat de te verwachten prestaties van zo'n ontvanger zullen zijn. De simulatieresultaten zijn algemeen en gelden voor elke bitrate. Het minimum configuratie ontwerp is opgezet voor bitrates $\leq$ 72 kbit/s voor Mobiele Radio. Het ontwerp is een principeschema zonder nadere detaillering. Voor het daadwerkelijke bouwen van de schakeling is een nadere uitwerking van het timingcircuit voor de sturing van de buffers die nodig zijn voor het "pipelining" principe en voor de sturing van de Viterbi processor. Verder is een "lay-out" nodig voor de inrichting van de "breadboards". Het ontwerp zal ongeveer 5 Eurocards beslaan. Het realiseren van de laboratoriumopstelling in dit verslag voorgesteld inclusief het testen en meten zal circa drie maanden in beslag nemen.

In deel twee van dit verslag zijn de appendices neergelegd welke de software van de simulatie beschrijven.

Lijst van gebruikte symbolen en hun betekenis

CORPSK - Correlative Phase Shift Keying

DQPSK - Differential Quaternary Phase Shift Keying

QPSK - Quaternary Phase Shift Keying

MSK - Minimum Shift Keying

FM - Frequency Modulation

$H(\omega)$ - Overdrachtsfunctie van het premodulatiefilter

$n_I(t)$ - Impulsresponsie van een filter dat voldoet aan 1^e Nyquistkriterium

$n_{III}(t)$ - Impulsresponsie van een filter dat voldoet aan 3^e Nyquistkriterium

$r_i(t)$ - Replica

$s_i(t)$ - Ontvangen signaal

$Z_{ni}(t)$ - Correlatieresultaat (analoog)

$\phi(t)$ - fasetraject

$\theta_k(t)$ - faseverandering

θ_j - fasepunt

$g(t)$ - "partial response" pulse

$m(t)$ - modulaat

VA - Viterbi Algoritme

T_i, T_i' - Toestandsregister, bevat een toestandsnummer

L_i, L_i' - Historieregister, bevat een samengesteld correlatieresultaat

S_i, S_i' - Fasetrajectregister, bevat een symbolensequentie of een toestandssequentie

N_{obs} - Lengte van het observatie-interval binnen het VA

c_{ij} - Correlatieresultaat (digitaal)

s_{ij} - symbool

A - Aantal symbolen in alfabet $\{a_m\}$ inputreeks

M - Aantal symbolen in alfabet $\{c_m\}$, is aantal faseveranderingen

p - Aantal fasepunten, of nummer van een fasepunt

m - Aantal bemonsteringen per symbool

L - Aantal symboolresponsies waaruit een Replica is samengesteld

h - Modulatie-index

q - Aantal kwantisatiebits

T - Aantal toestanden of toestandsnummer

N_o - Ruisvermogen per Hertz

n(t) - Ruissignaal

$\Delta\phi$ - Faseverschil

Literatuur

Referenties

1 L.J. Greenstein. "Spectra of PSK signals with overlapping baseband pulses." IEEE Trans. Commun., vol. COM-25. Mei 1977.

2 V.K. Prabhu. "PSK-type modulation with overlapping baseband pulses." IEEE Trans. Commun., vol. COM-25. Sept. 1977.

3 W.P. Osborne en M.B. Luntz. "Coherent and noncoherent detection of CPFSK." IEEE Trans. Commun., vol. COM-22. Aug. 1974.

4 G.J. Garrison. "A power spectral density analysis for digital FM." IEEE Trans. Commun., vol. COM-23. Nov. 1975.

5 T. Aulin, N. Rydbeck en C-E. Sundberg. "Bandwidth efficient digital FM with coherent phase tree demodulation." in Proc. ICC 1979, Boston, MA. Juni 1979.

6 ------. "Performance of constant envelope M-ary digital FM systems and their implementation." in Proc. NTC 1979. Washington, DC. Nov. 1979.

7 J.B. Anderson en D.P. Taylor. "A bandwidth-efficient class of signal-space codes." IEEE Trans. Inform. Theory, vol. IT-24. Nov. 1978.

8 Y. Tanaka, H. Harashima en H. Miyakawa. "Multi-mode binary CPFSK." Electron. Commun. Japan, vol. 58-A, no. 11, 1975.

9 N. Rydbeck en C-E. Sundberg. "Recent results on spectrally efficient constant envelope digital modulation methods." in Proc. ICC 1979, Boston. Juni 1979.

10 S. Pasupathy. "Nyquist's third criterion." Proc. IEEE, vol. 62. Juni 1974.

11 W.P. Osborne, M.B. Luntz. "Coherent and noncoherent detection of CPFSK." Trans. on Comm. Aug. 1974. IEEE.

12 D. Muilwijk. "Correlative Phase shift Keying - A class of constant envelope Modulation techniques." Trans. on Comm. Maart 1981. IEEE.

13 G. Forney. "The Viterbi Algorithm." Proc. IEEE. Maart 1973.

14 D. Muilwijk, J.H. Schadé. "CPM." Zie na 31.

15 A.J. Viterbi. "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm." IEEE Trans. Inform. Theory, vol. IT-13, pp. 260-269, Apr. 1967.

16 G.D. Forney, Jr. "Review of random tree codes." NASA Ames Res. Cen., Moffett Field, Calif., Contr. NAS2-3637, NASA CR 73176, Final Rep., Appendix A, Dec. 1967.

17 A.J. Viterbi. "Convolutional codes and their performance in communication systems." IEEE Trans. Commun. Technol., vol. COM-19, pp. 751-772, Oct. 1971.

18 G.J. Minty. "A comment on the shortest-route problem." Oper. Res., vol. 5, p. 724, Oct. 1957.

19 M. Pollack en W. Wiebenson. "Solutions of the shortest-route problem - A review." Oper. Res., vol. 8, pp. 224-230. Maart 1960.

20 O. Wing. "Algorithms to find the most reliable path in a network." IRE Trans. Circuit Theory (Corresp.), vol. CT-8, pp. 78-79. Maart 1961.

21 S.C. Parikh en I.T. Frisch. "Finding the most reliable routes in communication systems." IEEE Trans. Commun. Syst., vol. CS-11, pp. 402-406, Dec. 1963.

22 R. Busacker en T. Saaty, Finite Graphs and Networks : An Introduction with Applications. New York : McGraw-Hill, 1965.

23 Y.S. Fu. "Dynamic programming and optimum routes in probabilistic communication networks." in IEEE Int. Conv. Rec., pt. 1, pp. 103-105, Maart 1965.

24 J.K. Omura. "On the Viterbi decoding algorithm." IEEE Trans. Inform. Theory, vol. IT-15, pp. 177-179. Jan. 1969.

25 J.K. Omura. "On optimum receivers for channels with inter-symbol interference." (Abstract), presented at the IEEE Int. Symp. Information Theory, Noordwijk, Holland. Juni 1970.

26 J.K. Omura. "Optimal receiver design for convolutional codes and channels with memory via control theoretic concepts.", unpublished.

27 H. Kobayashi en D.T. Tang. "On decoding and error control for a correlative level coding system." (Abstract), presented at the IEEE Int. Symp. Information Theory, Noordwijk, Holland. Juni 1970.

28 H. Kobayashi. "Application of probabilistic decoding to digital magnetic recording systems." IBM J. Res. Develop., vol. 15, pp. 64-74. Jan. 1971.

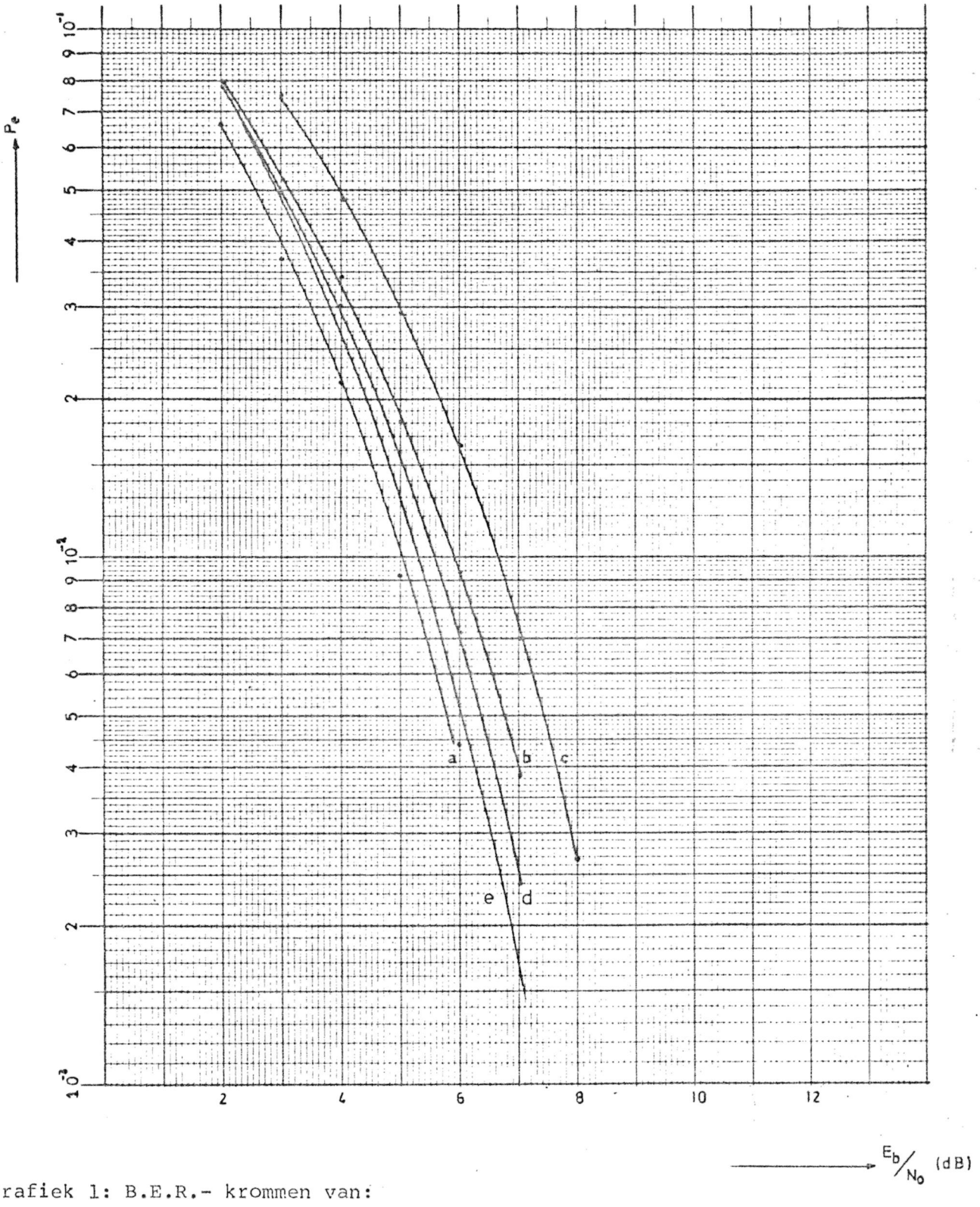
29 H. Kobayashi. "Correlative level coding and maximum-likelihood decoding." IEEE Trans. Inform. Theory, vol. IT-17, pp. 586-594. Sept. 1971.

30 J.M. Ockers. "Een digitale CORPSK(4-5) Modulator. Philips-HTS-Hilversum ex no. 8215.

31 S.H. Lebowitz en S.A. Rhodes. "Performance of coded 8PSK signaling for satellite communications."

14 D. Muilwijk en J.H. Schadé. "Correlative Phase Modulation for fixed satellite services (Phase I)." Draft Report European Space Agency, Philips Telecommunication Industry.

32 G. Forney. "Maximum-Likelihood sequence estimation of digital sequences in the presence of intersymbol interference." IEEE Trans. on Inf. Theory. May 1972.



Grafiek 1: B.E.R.- krommen van:

a),b),c) Ontvanger met Viterbi processor, resp. L=3,2 en 1.

d) Ontvanger zonder Viterbi processor, correlatie over drie symboolperioden.

e) DQPSK (uit de literatuur)

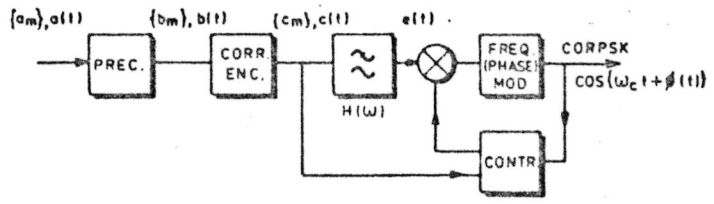


Fig. 1:1 Basic CORPSK modulator.

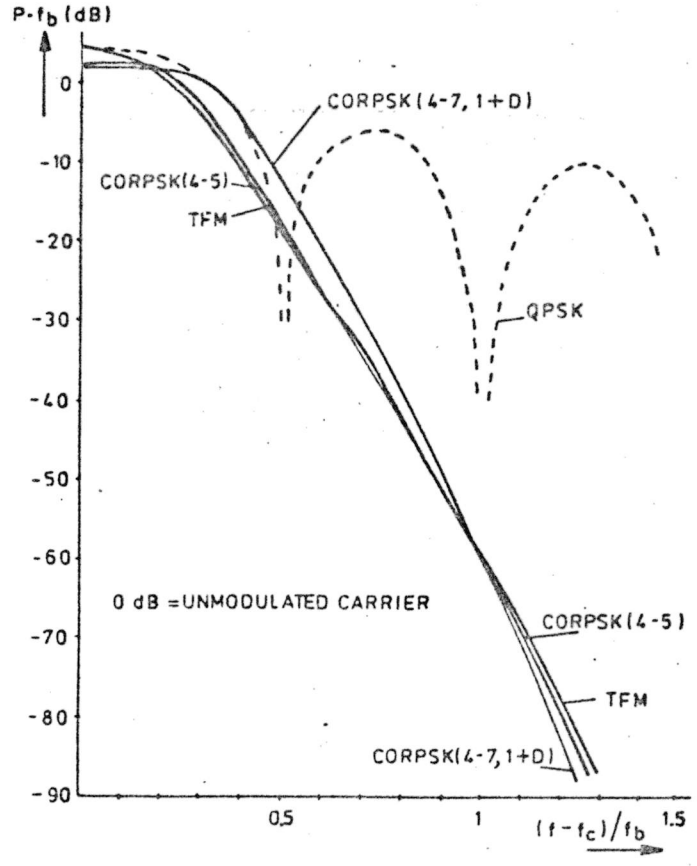


Fig. 1:2 Spectra of TFM, CORPSK(4-5), CORPSK(4-7, 1 + D), and QPSK.

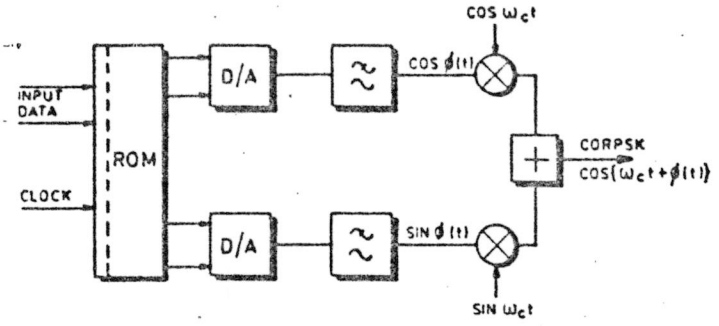


Fig 1:3 Quadrature CORPSK modulator.

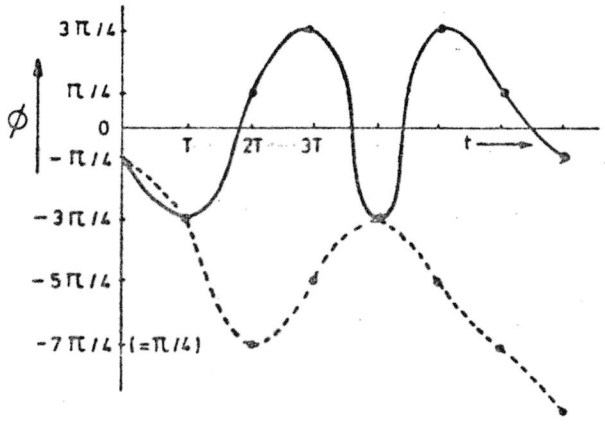


Fig. 1:4 Phase paths of phase shaped 4PSK and CORPSK(4-5).

Input a_m	Output c_m	$\Delta\phi_m$
1	−1	$-\pi/2$
2	0	0
3	+1	$+\pi/2$
0	+2 or −2	$+\pi$ or $-\pi$

Tabel 1:5 Correlatiecodering voor CORPSK(4-5)

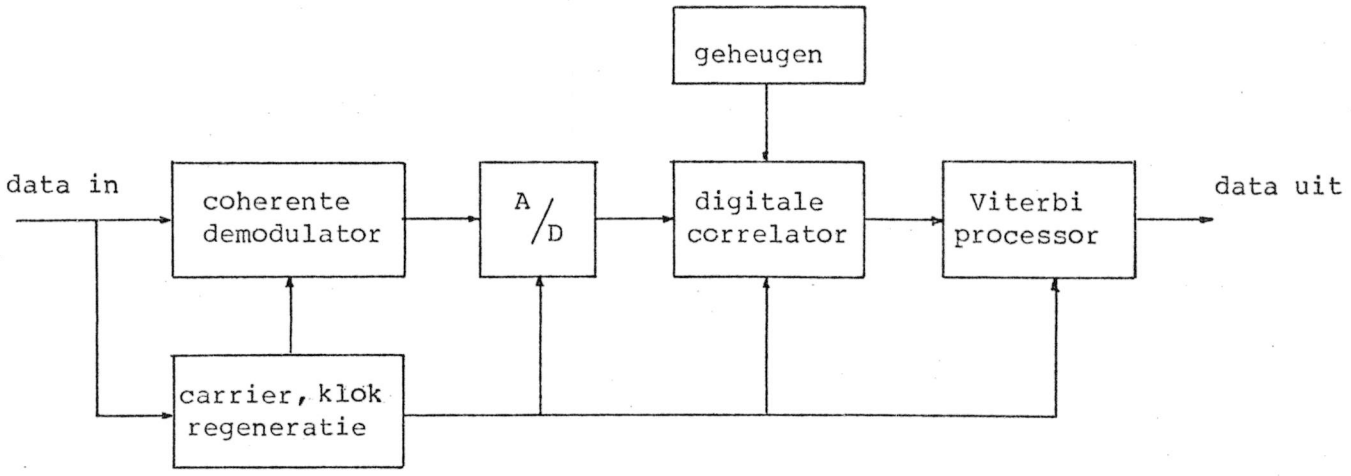


Fig. 2:1 Optimale ontvanger voor CORPSK

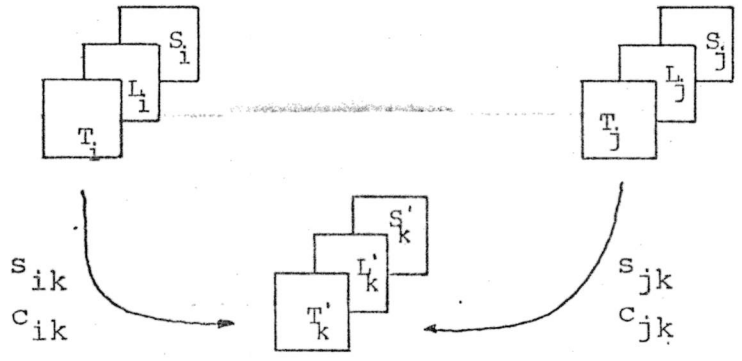


Fig 2:2 Toestandsverandering.

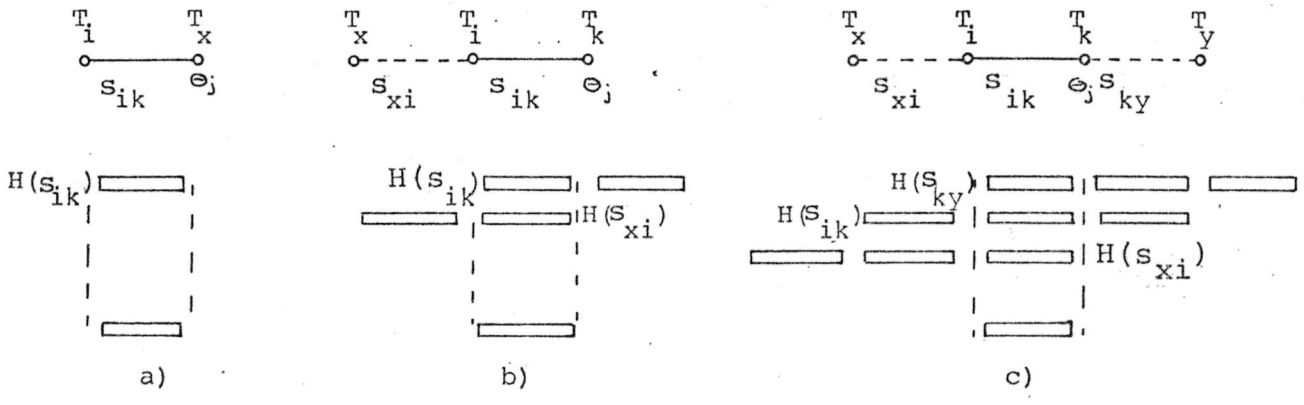


Fig. 2:3 Replica's samengesteld uit: a) 1, b) 2, of c) 3 symboolresponsies.

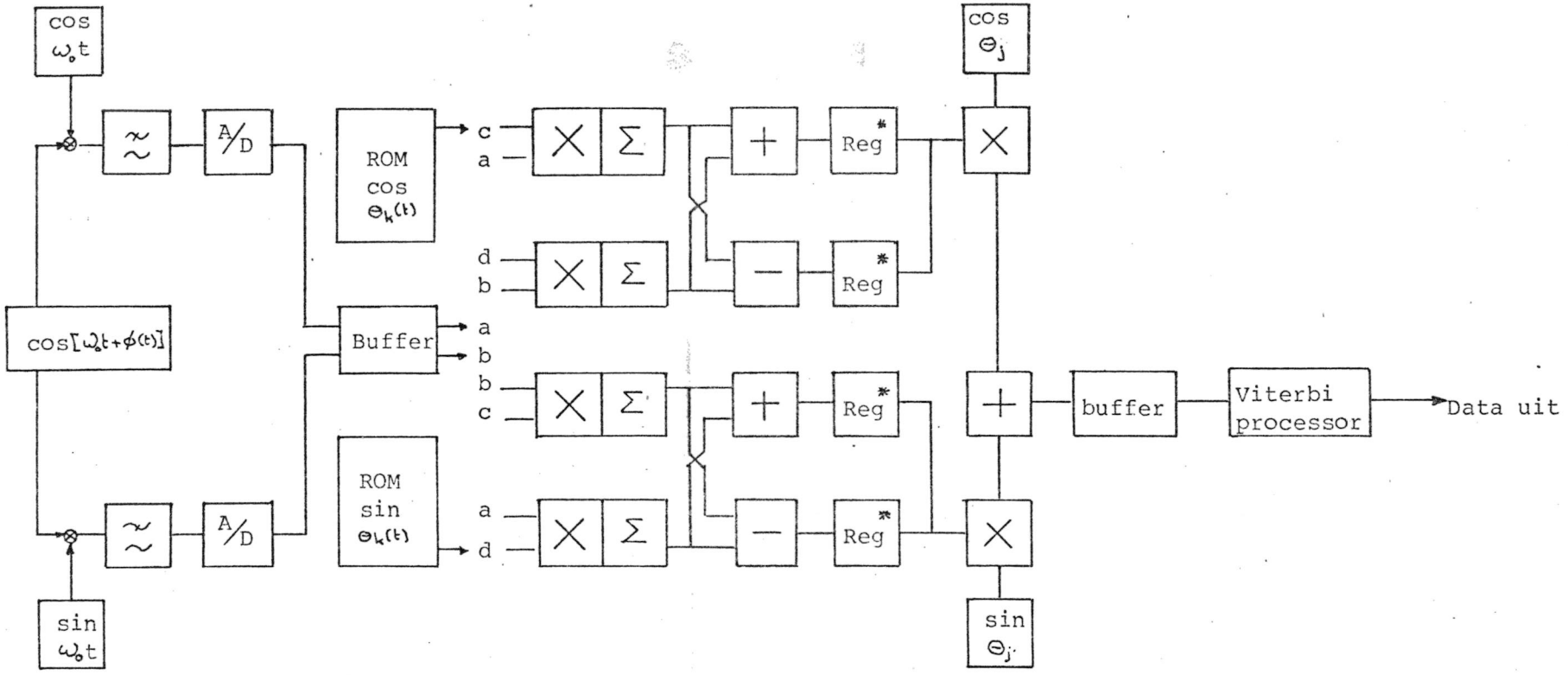


Fig. 3:1 Principe schema 1

$$Z_{ni} = \cos\theta_j\left[\Sigma\cos\phi(t)\cos\theta_k(t) \pm \Sigma\sin\phi(t)\sin\theta_k(t)\right] + \sin\theta_j\left[\Sigma\sin\phi(t)\cos\theta_k(t) \mp \Sigma\cos\phi(t)\sin\theta_k(t)\right]$$

$$Z_{ni} = \{\pm1,0\}\left[\Sigma ac \pm \Sigma bd\right] + \{0\pm1\}\left[\Sigma bc \mp \Sigma ad\right]$$

* 3-state

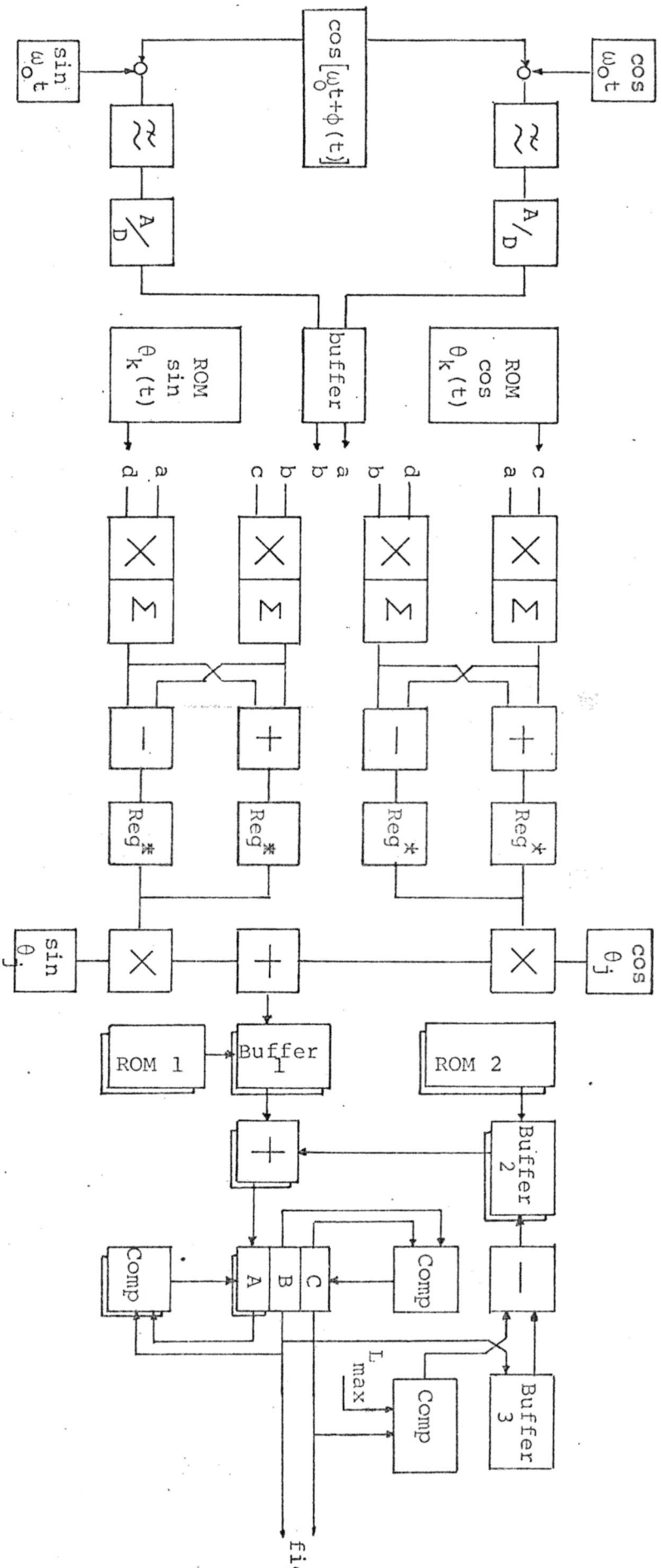


fig. 3:7

Fig. 2:3 Principe schema 2

$$Z_{ni} = \cos\theta_j \left[\Sigma\cos\phi(t)\cos\theta_k(t) \pm \Sigma\sin\phi(t)\sin\theta_k(t)\right] + \sin\theta_j \left[\Sigma\sin\phi(t)\cos\theta_k(t) \mp \Sigma\cos\phi(t)\sin\theta_k(t)\right]$$

$$Z_{ni} = \{\pm 1, 0\}\left[\Sigma\ ac \pm \Sigma\ bd\right] + \{0, \pm 1\}\left[\Sigma\ bc \mp \Sigma\ ad\right]$$

* 3-state

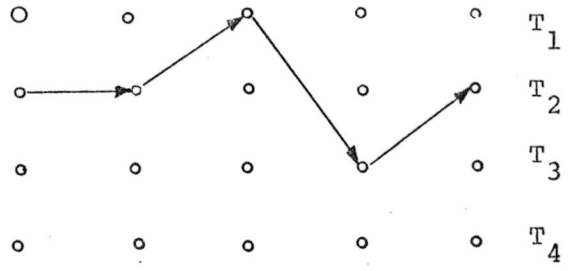


Fig. 3:3 Data opslag (o) zijn geheugenelementen

S_5	S_4	S_3	S_2	S_1
4	2	2	1	3

T_i toestand van herkomst

oudste symbool — nieuwste symbool

Fig.3:4 Gegevens, symbolen en toestandssequentie

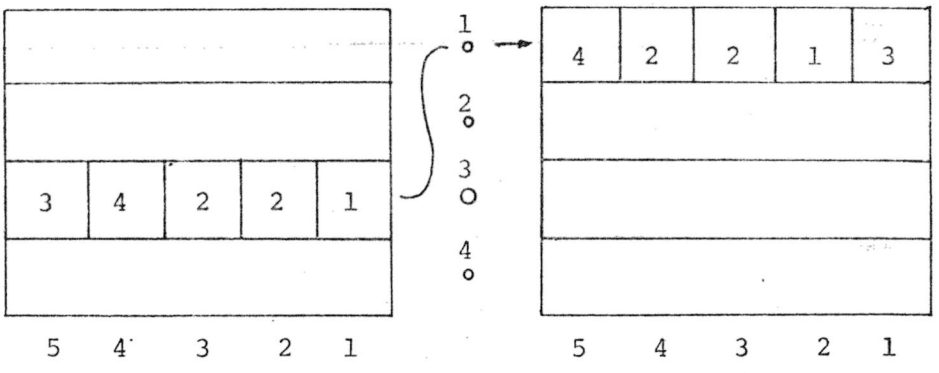
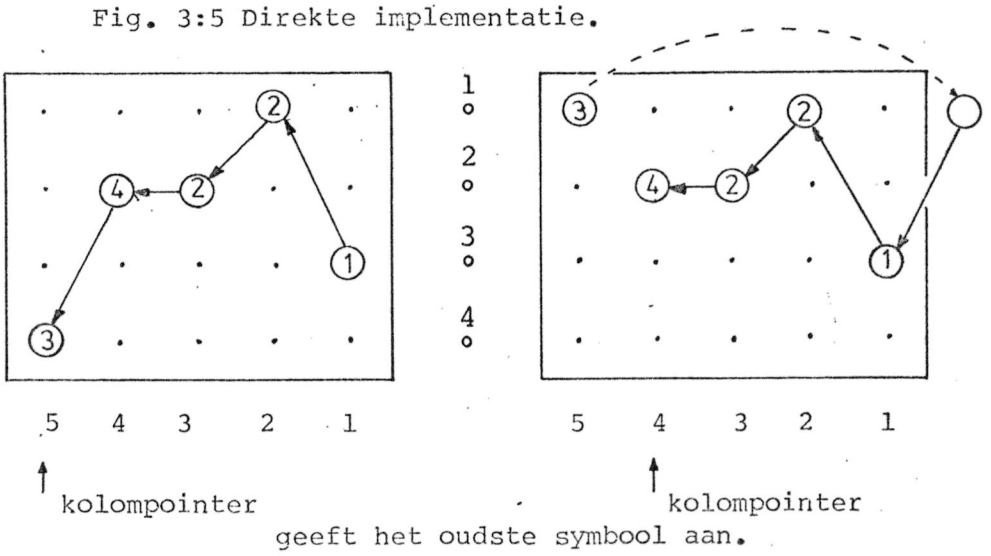


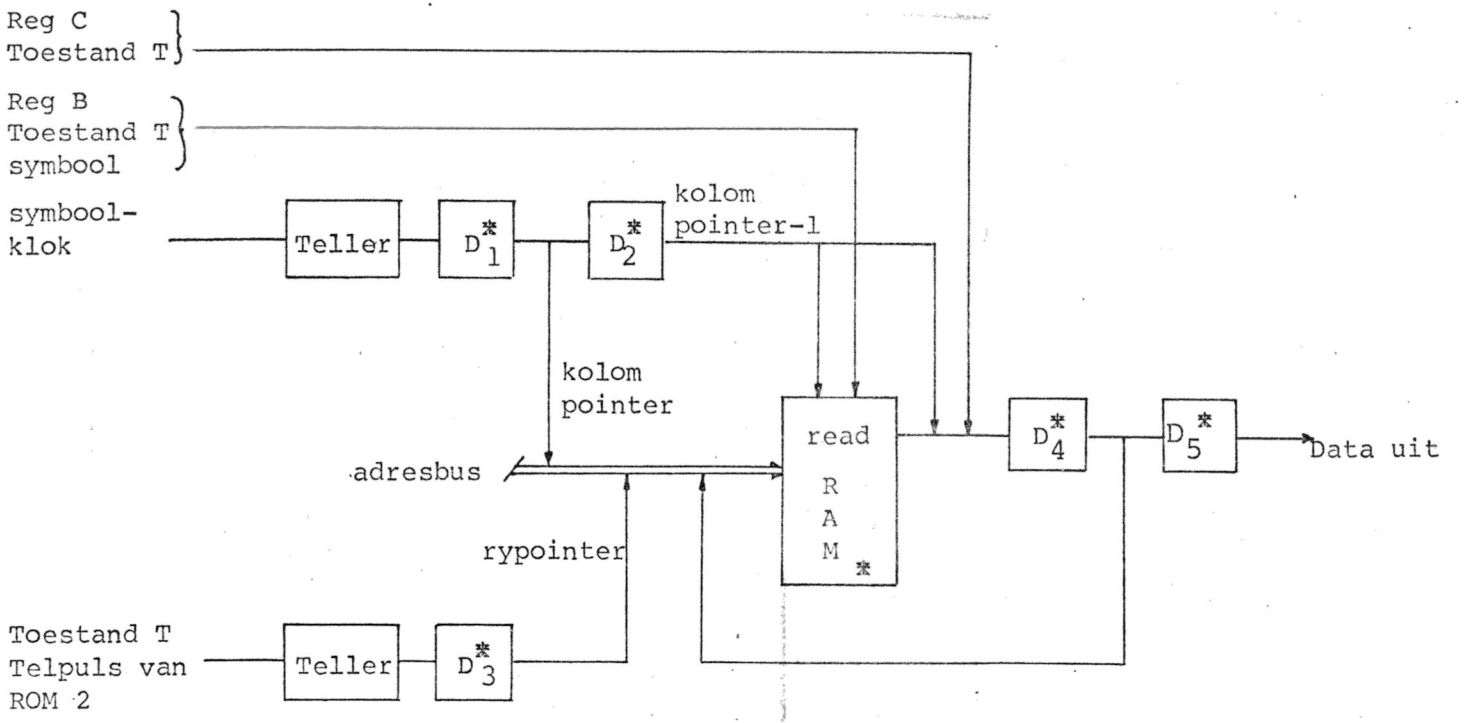
Fig. 3:5 Direkte implementatie.

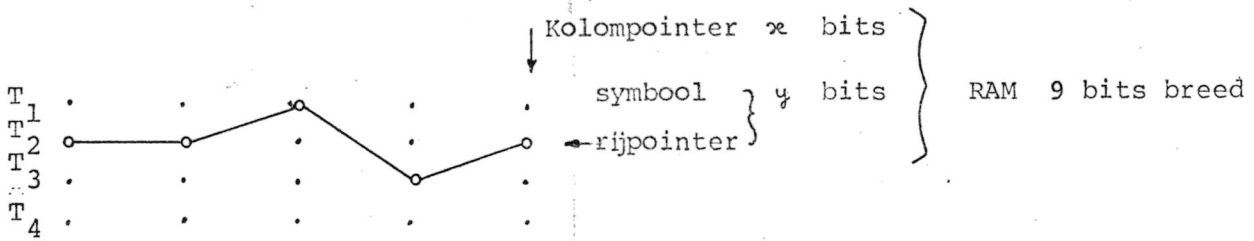


geeft het oudste symbool aan.

Fig. 3:6 Implementatie met efficiente geheugenorganisatie.

(a) a = geheugeninhoud. S_x (symbool) staat ook in het geheugen (hier niet aangegeven).





Fasetrajekt in RAM (o) is een geheugenelement.

Fig. 3:7 Symboolbeslissingsschakeling. * 3-state

TECHNISCHE HOGESCHOOL DELFT
Afdeling der Elektrotechniek

Aard : Digitale modulatie methoden. Deel II.
Omvang : Deel I : 59 blz. - Deel II : 75 blz.
Datum : Juli 1982
UDC :
INSPEC :

Vakgroep : Transmissie van Informatie

Codenummer: 05-1-565-28-186

Auteur : Bart Wijne

Titel : Ontwerp en simulatie van een correlatie ontvanger met Viterbi processor voor "Correlative Phase Shift Keying" CORPSK(4-5)

Korte inhoud: Zie voorwoord

Mentor : Ir. J.S. van Sinttruijen

TRANSMISSIE VAN INFORMATIE

DEEL II APPENDICES BY HET AFSTUDEERVERSLAG:

ONTWERP EN SIMULATIE, VAN EEN CORRELATIEONTVANGER MET VITERBI PROCESSOR, VOOR "CORRELATIVE PHASE SHIFT KEYING"

CORPSK(4-5)

AFSTUDEERFASE OKT '81 - JUNI '82

PHILIPS HUIZEN - TH-DELFT

5 JULI 1982

B WIJNE

LARENSESTRAAT 24

070 - 465326

2574 VJ DEN HAAG

035 - 894492 (PHILIPS)

Inhoud:

Inleiding appendices.

Deze appendices behoren by het verslag :

Ontwerp en simulatie, van een correlatieontvanger met Viterbi processor, voor "Correlative Phase Shift Keying"
---------- CORPSK(4-5) ----------

Appendix 1 bevat een algemene beschryving van het simulatie-programma.

Appendix 2 bevat een verklaring van formules welke gebruikt zyn in het simulatieprogramma.

Appendix 3 bevat een verklaring van de algemeen gedeclareerde grootheden welke door het gehele programma heen gebruikt zyn.

Appendix 4 geeft een korte verklaring van de Pascalinstrukties en systeeminstrukties welke gebruikt zyn om het programma te maken en te laten werken op de computer.

Appendix 5 bevat de programmatuur in de volgende onderverdeling:

a) Voorbeelden van de iputfile en outputfile "Pinp.dat" en "Pout.dat",
b) de file met de algemene declaraties, "BWHEAD5.PAS",
c) het Acquisitie programma: Program Acquisitie(Pinp,Pout), welke de startprocedures van de gewenste Modules aanroept,
d) de drie Modules:
   1- Module VTI_Ontvangersimulatie(Pinp;Pout)
   2- Module MLE_Ontvangersimulatie(Pinp,Pout)
   3- Module Zendersimulatie(Pinp,Pout)

De keuze tussen Module 1 of Module 2 wordt bepaald door de variabele C_ORDV in de inputfile Pinp.dat.
Als C_ORDV=0 wordt Module 1 gekozen.
Als C_ORDV=1,2 of 3 wordt Module 2 gekozen met respectievelyk L= 1,2 of 3.

In Module 1 worden de variabelen C_KBSL, C_KBSC, C_LBIN, C_OBIN niet gebruikt, ze worden echter wel ingelezen, omdat de inleesroutine "initialisatie" voor alle Modules hetzelfde is.

1. APPENDIX; DE SIMULATIE

BY HET AFSTUDEERVERSLAG:

ONTWERP EN SIMULATIE,
VAN EEN CORRELATIEONTVANGER
MET VITERBI PROCESSOR, VOOR
"CORRELATIVE PHASE SHIFT KEYING"
---------- CORPSK(4-5) ----------

AFSTUDEERFASE OKT '81 - JUNI '82

PHILIPS HUIZEN - TH-DELFT

5 JULI 1982

B WIJNE

LARENSESTRAAT 24

070 - 465326

2574 VJ DEN HAAG

035 - 894492 (PHILIPS)

De simulatie is uitgevoerd op een VAX minicomputer. Het simulatieprogramma ia geschreven in de programmeertaal PASCAL. In de appendix "PASCAL" worden beknopt die PASCAL-istrukties en systeeminstrukties verklaart welke in het ontwikkelde programma zyn gebruikt.

Het programma bestaat uit vyf onderdelen:

1: 'BWHEAD5.PAS' Dit is een filename. Deze file bevat de algemene declaraties. Deze declaraties moeten voor elk programmadeel hetzelfde zyn. Om deze reden zyn deze declaraties in een aparte file geplaatst.

2: Module Zendersimulatie(Pinp,Pout); "Pinp.dat" en "Pout.dat" zyn resp. de inputfile en de outputfile. Dit Module bevat het programma dat de CORPSK(4-5)-zender simuleert. De start Function is: "Zender(Parameter list)".

3: Module VTI_ontvangersimulatie(Pinp,Pout); Dit Module bevat het programma dat de CORPSK(4-5)-ontvanger met Viterbi processor simuleert. De start Procedure is: "VTI_ontv".

4: Module MLE_ontvangersimulatie(Pinp,Pout); Dit Module bevat het programma dat de CORPSK(4-5)-ontvanger zonder Viterbi processor simuleert. De start Procedure is: "MLE_ontv".

5: Program Aquisitie(Pinp,Pout); Dit programma roept de startprocedures uit de gewenste Modules aan.

Delen 2,3,4 en 5 hebben alle als eerste instruktie :
%INCLUDE 'BWHEAD5.PAS' .

Het programma leest de te verwerken gegevens uit de inputfile "Pinp.dat" en schryft de resultaten in de outputfile "Pout.dat".

1) BWHEAD5.PAS

De algemene declaraties van een programma zyn zichtbaar en geldig binnen elke procedure van het programma en de gebruikte Modules.
Een beperkt aantal declaraties is gebruikt voor parameteroverdracht van de Module Zendersimulatie(Pinp,Pout) naar de Modules Ontvangersimulatie.
Deze parameters zyn de matrices AB en CS en de variabelen BPSY, SPSY en SYMB. AB is een matrix met als elementen de monsters van alle symbolen van de quadratuurcomponenten cosϕ en sinϕ
CS is een hulpmatrix die by de matrix AB hoort.
BPSY is de variabele voor het aantal bits per symbool, voor CORPSK(4-5) is de waarde van deze variabele gelyk aan twee.
In de Module Zendersimulatie(Pinp.Pout) is SPSY de variabele voor het aantal monsters per symbool in de quadratuurcomponenten.
Deze variabele bepaalt mede de nauwkeurigheid waarmee de "Fast Fourier Transformation" wordt bepaald. Daarom kiezen we minimaal SPSY:=16. SYMB is de variabele voor het aantal symbolen wat door de zender wordt gegenereerd. SYMB is een veelvoud van twee. Ingevoerd moet worden M:=dilog(SYMB*SPSY).
De Type declaraties die met deze parameteroverdracht samenhangen zyn:
TYPE COMPLEX, DIMAB, DIMCS, DIMY, MTRAB, MTRCS.
Voor een nadere verklaring van de overige declaraties, zie de appendix "Declaraties".

2) Module Zendersimulatie(Pinp,Pout);

Dit Module bevat een reeds eerder op de afdeling VOR ontwikkeld programma. Het programma verwerkt sequentieel de uit de file "Pinp.dat" gelezen gegevens

Zie Function Zender(pm list); (appendix "programmatuur").

A) ANEW_WEER: De initialisatie gegevens M, BPSY en MSPSY worden ingelezen. By een gewenst aantal symbolen "SYMB" en een gewenst aantal monsters per symbool "SPSY" is M:=dilog(SYMB*SPSY) MSPSY:=dilog(SPSY).

B) REEKSGEN: Er wordt een binaire reeks gegenereerd en opgeslagen in de matrix R. Hieruit wordt een symbolenreeks samengesteld. Tevens wordt de som van het aantal symbolen bepaald. De reeks kan gemodificeerd worden door het aantal te wyzigen symbolen, en deze gewyzigde symbolen zelf, op te geven. Dit omdat in bepaalde gevallen de som van alle symbolen nul moet zyn en het programma hierin niet voorziet.

C) LEVELSHIFT: Volgens het CORPSK(4-5) voorschrift wordt een symbool aan het symbolenalfabet toegevoegd en wordt een correlatie in de symbolenreeks aangebracht. Is de som van alle symbolen ongelyk aan nul dan wordt de Function Zender(pm list) "FALSE" . We kunnen de som van alle symbolen nul maken door de reeks in B) te modificeren.

D) IMP_INT: Is de som van alle symbolen van de aangeboden reeks ongelyk aan nul dan wordt het programma beeindigd.
Is de som gelyk aan nul dan wordt een "Fast Fourier Transformation" op de symbolenreeks toegepast welke het signaal transformeert van het tyddomein naar het frequentiedomein.

E) NYQ_III, RAISEDCOSINE, SQUAREPULSE: Er kan nu een frequentie "shaping" worden toegepast door te kiezen uit bovenstaande programmaonderdelen. Voor CORPSK kiezen we NYQ_III. de "roll off factor" moet dan worden ingelezen.

F) XPT_TIJD: Het signaal wordt mbv een "Fast Fourier Transformation" van het frequentiedomein naar het tyddomein getransformeerd.

G) PHASEDEV: Is een experimenteel te bepalen schaalfactor (2, 1, 0.5, 0.25) zodanig dat de fase door veelvouden van π gaat op de bemonstertydstippen. Voor CORPSK(4-5) is deze factor gelyk aan één.

3) Module VTI_ontvangersimulatie(Pinp,Pout);

De kern van het programma wordt gevormd door een vyftal procedures, dit zyn:

1) RUIS_FILTER, 2) CORRELATOR, 3) HISTORIE, 4) SURVIVORS en 5) SYMBOL(Z).

De procedures bestaan voornamelyk uit matrix manipulaties. Om niet steeds via de pm-list de matrices te moeten overdragen zyn alle matrices by de algemene declaraties gedeclareerd. De matrices zyn te herkennen aan een dubbele notatie: AA, BB, CC enz. met uitzondering van de matrices AB en CS welke door de Module Zendersimulatie(Pinp,Pout) worden gegenereerd.

Er volgt nu een algemene beschryving van de procedures in volgorde van essentie.
De appendix "Procedures" geeft een veklaring van de gebruikte formules in de procedures van het Module VTI_Ontvangersimulatie.

Procedure RUIS_FILTER;

Hulpprocedures : G05DDF, G05CBF, FFTC_1, FFTC, FILTRMXFL.

Matrices : AB, CS, UU, RR.

Zie fig. 1 De matrices AB en CS worden door het Module Zendersimulatie(Pinp,Pout) gegenereerd. De elementen van de matrix AB bevatten de monsters van de quadratuur-componenten cosϕ en sinϕ in het tyddomein.
De elementen van de matrix UU worden gevormd door by de elementen van de matrix AB onafhankelyke ruis op te tellen.
Hierna vindt een Butterworth bewerking op de elementen van de matrix UU plaats.
UU[I].R bevat een monster van cosϕ vermeerderd met ruis.
UU[I].I bevat een monster van sinϕ vermeerderd met ruis.
Outputmatrix is de matrix UU.

Procedure CORRELATOR.

Hulpprocedures : ADC

Matrices : AA, DD, UU, CC

Zie fig 2 De elementen van de matrix UU bevatten de monsters van alle symbolen van de quadratuurcomponenten cosϕ en sinϕ .
De procedures CORRELATOR, HISTORIE en SURVIVORS zyn procedures die voor elk symbool moeten worden uitgevoerd. Hiertoe worden telkens uit de matrix UU, de monsters van een symbool van de quadratuurcomponenten in de matrices AA en DD gecopieerd. Hierna vindt kwantisatie en correlatie met alle replica's uit de matrix BB plaats. De correlatieresultaten worden opgeslagen in de matrix CC.
Kwantisatie vindt plaats met behulp van de hulpprocedure ADC(RX) en met de operatie X:=(X DIV SF2). De variabelen SF zyn schaalfactoren.
Outputmatrix is de matrix CC.

Procedure HISTORIE.

Hulpprocedures : State

Matrices : CC, TT, LL.

Zie fig 3 Na correlatie met alle replica's zyn de correlatie-resultaten voor de fasepunten 0° en 90° in de matrix CC opgeslagen. De resultaten voor de fasepunten 180° en 270° worden bepaald door inverteren van de resultaten van de fasepunten 0° en 90°. De derde dimensie van de matrix CC hangt samen met het fasepunt. Hierna moeten de correlatieresultaten vermeerderd worden met de bybehorende historieregisters uit de matrix LL. Welk element uit de matrix LL by welk element uit de matrix CC moet worden opgeteld is opgeslagen in de elementen van de matrix TT, welke dezelfde dimensies heeft als de matrix CC. Outputmatrix is de matrix CC.

Procedure SURVIVORS.

Hulpprocedures : geen

Matrices : CC, SS, TT, LL, VV, XX.

Zie fig 4 In de matrix CC moet nu naar een maximaal samengesteld correlatieresultaat per toestand worden gezocht By ieder element uit de matrix CC hoort een toestand van herkomst en een toestand van aankomst. De indices van deze toestanden zyn in de elementen van respectievelyk de matrix TT en de matrix XX opgeslagen welke dezelfde dimensies hebben als de matrix CC. Uit de elementen van de matrix CC wordt nu per toestand met behulp van de elementen van de matrix XX het maximum resultaat per toestand gezocht. Dit zyn de nieuwe samengestelde correlatieresultaten Li welke worden opgeslagen in de matrix LL. By elk van het maximum per toestand wordt de toestand van herkomst uit de matrix TT in de matrix SS opgeslagen. De plaats waar dit, in de matrix SS wordt opgeslagen, wordt bepaalt door de toestand van aankomst (rynummer) en het symboolnummer (kolomnummer). Nadat alle elementen uit de matrix CC zyn afgewerkt wordt het maximum uit de elemenenten van de matrix LL bepaald. Het elementnummer van dit maximum komt overeen met een toestandsnummer, dit nummer wordt opgeslagen in de matrix VV. Dit element geeft aan, het startelement in de overeenkomstige kolom van de matrix SS, waar de zoekcyclus voor het bepalen van het ontvangen symbool moet beginnen.
De inhoud van de elementen van de matrix LL groeien aan, wordt een maximum LMAX overschreden dan worden alle elementen van de matrix LL met dit maximum LMAX verminderd.
Outputmatrices zyn de matrix SS en de matrix VV.

Procedure SYMBOL(Z).

Hulpprocedures : State

Matrices : VV, SS, YY, ZZ, FF.

De parameter "Z" geeft de lengte van het observatieinterval aan.
De procedure SYMBOL(Z) zoekt voor een bepaalde "Z" in de
matrix SS naar de ontvangen symbolen. De dimensies (2) van de
matrix SS zyn : Aantal ryen is gelyk aan het aantal toestanden.
Aantal kolommen is gelyk aan het aantal symbolen.
De dimensie (1) van de matrix VV is :
Aantal elementen is gelyk aan het aantal symbolen.
De matrix SS bevat voor alle symbolen (kolommen), de toestand van
herkomst, van de maximum samengestelde correlatieresultaten per toestand.
Een kolomnummer komt overeen met een symboolnummer en een regelnummer
komt overeen met een indexnummer van de toestand van aankomst.
Van een symboolnummer X ,dus kolomnummer X in de matrix SS,
bevat elementnummer X van de matrix VV, de index van de toestand van
aankomst welke hoort by het maximum samengestelde correlatieresultaat.
Dit is de startregel in kolom nummer X van de matrix SS waar de
zoekcyclus begint. De inhoud van het aldus gevonden element uit de
matrix SS geeft aan, het regelnummer voor kolom X-1 waar we het
volgende element vinden. Zo zoeken we "Z" kolommen terug in de matrix SS.
Het nu gevonden element is een toestand van herkomst waaruit met behulp
van de definitie van toestand het ontvangen symbool kan worden
bepaald. Voor verschillende observatieintervallen kan zo, uit
dezelfde matrix SS met behulp van de matrix VV steeds weer de
ontvangen symbolenreeks worden bepaald. De gevonden symbolen-
reeks wordt vergeleken met de door de Module Zendersimulatie(Pinp,Pout)
overgedragen symbolenreeks in de matrix YY, eventuele bitfouten
worden, per symbool in de matrix ZZ en per observatie interval in
de matrix FF opgeslagen . Outputmatrices zyn de matrix ZZ en de matrix FF.

Procedure Replica's

Hulpprocedures : Check (intern).

Matrices : SS (intern), AA, BB .

Deze procedure genereert de replica's waarmee het ingangssignaal per symboolperiode wordt gecorreleerd. In de matrix AA is de stapresponsie van het premodulatie filter ingelezen via de procedure "initialisatie". Uit deze stapresponsie worden nu eerst de symboolresponsies afgeleid en in de matrix SS opgeborgen. Hierna worden afhankelyk van de variabele ORDV (L-parameter van het Viterbi Algoritme) de replica's uit de symboolresponsies samengesteld en opgeborgen in de matrix BB. De symbolen waaruit een replica is samengesteld worden volgens TOT:=(100*X)+(10*Y)+(Z) in element BB[I,0] opgeborgen X,Y en Z zyn de drie symbolen waaruit voor L=3 een replica is samengesteld, voor L=2 is X=0 en voor L=1 is X=0 en Z=0. De stapresponsie wordt ingelezen voor 32 monsters per symboolperiode, moet dit aantal monsters kleiner zyn dan wordt de matrix BB gemodificeerd. Outputmatrix is de matrix BB.

Procedure Toestand_locaties

Hulpprocedures : Fase (intern), Mir (intern),
Tstd (intern)

Matrices : TT, XX, BB

Deze procedure bepaalt by elke replica de toestand van herkomst en de toestand van aankomst en bergt deze op in de matrix TT en de matrix XX. Deze matrices hebben dezelfde dimensies als de matrix CC.
De replica's uit de matrix BB zyn faseveranderingen naar het fasepunt 0° De replica's voor de overige drie fasepunten worden hieruit afgeleid. Deze fasepunten worden door de derde dimensie in de matrices XX, TT en CC aangegeven (TT[-,-,z]). Van de replica's uit de matrix BB moeten we ook hun gespiegelde waarde nemen, of we met deze gespiegelde of met de niet gespiegelde replica te maken hebben wordt bepaald door de tweede dimensie van de matrices XX, TT en CC (XX[-,y,-]). De eerste dimensie komt overeen met het replica nummer uit de matrix BB (CC[x,-,-]).
Outputmatrices zyn de matrix XX en de matrix TT.

Procedure VTI_Ontv

Hulpprocedures : Alle.
Matrices : Alle.

Dit is de startprocedure van het Module VTI_Ontvangersimulatie(Pinp,Pout).
Deze procedure roept de overige procedures in de juiste volgorde aan.

Procedure START_WAARDEN.

Hulpprocedures : Geen.
Matrices : LL, FF, ZZ, RR.

Deze initialisatieprocedure moet voor elk stel kwantisatieparameters worden uitgevoerd. De schaalfactoren SF en de factor LMAX worden berekend. De matrices LL, FF en ZZ worden geinitialiseerd.

Procedure GEGEVENS.

Hulpprocedures : Geen.
Matrices : BB, RR, TT, XX.

Dit is een uitvoer procedure. De uit de inputfile "Pinp.dat" ingelezen variabelen worden in de outputfile "Pout.dat" afgedrukt. Dit dient als controle op het ingevoerde gegevenspakket.

Procedure RESULTS.

Hulpprocedures : Geen
Matrices : RR, FF, ZZ.

Dit is een uitvoerprocedure. Resultaten uit de matrix FF en ZZ worden afgedrukt.

Procedure INITIALISATIE.

Hulpprocedures : Geen.
Matrices : QQ, RR, XX, TT, BB.

Dit is een inlees procedure. Alle in te lezen variabelen worden via deze procedure ingelezen.

Procedure ADC : Hulpprocedure by Correlator.
Procedure STATE : Conversie procedure.
Procedure Fase : Hulpprocedure by Replica's
Procedure Mir : Hulpprocedure by Replica's
Procedure TSTD : Hulpprocedure by Replica's

Procedure G05DDF en G05CBF (Extern)

Hulpprocedures : Zie appendix "procedures"
Matrices : idem

Dit zyn procedures voor de ruisgeneratie. De procedures zyn afkomstig uit de algememe bibliotheek [naglib]nag/lib.

Procedures FFTC_1 en FFTC (Extern)

Hulpprocedures : Zie appendix "programmatuur"
Matrices : AB, CS

Dit zyn procedures uit de Module Zendersimulatie(Pinp,Pout).
Het zyn "Fast Fourier Transformations" resp naar het frekwentie-domein en naar het tyddomein.

Procedure FILTRMXFL. (Extern)

Hulpprocedures : Zie appendix "programmatuur"
Matrices : UU, CS.

Dit is een procedure uit de Module Zendersimulatie(Pinp,Pout).
De bewerking is een Butterworth filtering in het frekwentie-domein.

4) Module MLE_Ontvangersimulatie(Pinp,Pout)

Dit Module is nagenoeg hetzelfde als het vorige Module alleen de Viterbi processor ontbreekt. We zullen in het kort de belangrykste verschillen aangeven.

De kern van het Module wordt gevormd door een viertal procedures dit zyn:

1) RUIS_FILTER 2) CORRELATOR_MLE 3) SOMMATOR
4) SYMBOL_MLE

Procedure Correlator_MLE

De procedure is identiek aan die in de Module VTI_Ontvangersimulatie, er vindt echter geen kwantisatie plaats en de correlatie geschiedt over drie symboolperioden in plaats van over een (L=3).

Procedure Sommator

Dit is een hulpprocedure by de procedure "Symbol_MLE". Alle correlatieresultaten van de replica's met hetzelfde middelste symbool worden via een e-macht gesommeerd. De outputvariabele is deze gesommeerde waarde.

Procedure Symbol_MLE

Met behulp van de procedure "Sommator" worden de samengestelde correlatieresultaten voor alle symbolen met elkaar vergeleken. Dat symbool met het grootste samengestelde correlatieresultaat wordt als het ontvangen symbool gekozen. Dit symbool wordt vergeleken met het gezonden symbool in de matrix YY, bitfouten en/of symboolfouten worden opgeborgen in de matrix FF en de matrix ZZ

Procedure Repl_MLE

Deze procedure genereert op dezelfde wyze als in het Module "VTI_Ontvangersimulatie" 48 replica's (L=3). De replica's zyn hier echter drie symboolperioden lang terwyl ze in de Module "VTI_Ontvangersimulatie" slechts een symboolperiode lang zyn

De overige procedures zyn met wat kleine, voor zichzelf sprekende veranderingen, gelyk aan de procedures in de Module "VTI_Ontvangersimulatie" .

2. APPENDIX; PROCEDURES

BY HET AFSTUDEERVERSLAG:

ONTWERP EN SIMULATIE,
VAN EEN CORRELATIEONTVANGER
MET VITERBI PROCESSOR, VOOR
"CORRELATIVE PHASE SHIFT KEYING"
---------- CORPSK(4-5) ----------

AFSTUDEERFASE OKT '81 - JUNI '82

PHILIPS HUIZEN - TH-DELFT

5 JULI 1982

B WIJNE

LARENSESTRAAT 24

070 - 465326

2574 VJ DEN HAAG

035 - 894492 (PHILIPS)

In deze appendix worden details uit de procedures van de Module VTI_Ontvangersimulatie nader toegelicht.

Procedure Initialisatie

Hulpprocedure Geen

VRZ1 heeft 15 elementen. Deze elementen komen alle voor in de input file "Pinp.dat", de volgorde is niet van belang. Achter deze elementen staat de waarde van de overeenkomstige variabele vermeld. Voor een verklaring van de variabelen: Zie appendix Declaraties.

Procedure Gegevens

Hulpprocedure Geen

De variabelen ingelezen door de procedure initialisatie worden ter controle en ter verduidelyking afgedrukt in de output file "Pout.dat". Een beperkt aantal variabelen wat by een specifieke output hoort wordt apart by die output afgedrukt mbv de procedure Results.

Procedure Results

Hulpprocedure Geen

De variabelen die by de specifieke output horen zyn de variabelen KBSC, KBSL en GSNR waarmee samenhangen de variabelen SF1, SF2 en LMAX. Deze variabelen worden apart afgedrukt by de byhorende output.
Voor een gegeven Observatieinterval x kunnen de eerste x symbolen niet gedetekteerd worden. Het aantal symbolen is dan SYMB - OBIN per Ruisrealisatie. Voor alle ruisrealisaties is het aantal symbolen: Aantal:=(RSRE-LBRE+1)*(SYMB-OBIN) waarin OBIN varieert van LBIN t/m OBIN. In FF[I] staan de gemaakte bitfouten per observatie interval. In ZZ[I,J] staan de gemaakte bitfouten en symboolfouten per symbool voor alle observatie intervallen. Als de variabele "FOLT" "YES" is, worden de matrix TT, de matrix XX en de matrix ZZ afgedrukt.

Procedure : Replica's

Hulpprocedures : Check (intern)

Een replica is samengesteld uit 1,2 of 3 symboolresponsies (L=1,2 of 3). De Function Check selekteert uit alle replica's alleen die replica's die in het geheugen dienen te worden opgeslagen (Spiegeling). Bovendien wordt met het correlatievoorschrift gecontroleerd of geen verboden symbolencombinaties in het geheugen worden opgenomen.
De symboolvariabelen X, Y en Z doorlopen alle mogelyke symbolen (0,1,2,3,4). ORDV komt overeen met L.
als L=1 is X=0 en Z=0, als L=2 is X=0. Het aantal toestanden TSTN is pM^(L-1) , p=4, M=5 en L=1,2 of 3.
By het samenstellen van een replica moet ervoor gezorgd worden dat de fase voor het middelste symbool voor t=Ts door 0° gaat. De stapresponsie in matrix AA gaat voor het middelste symbool op t=Ts door 0°. Alleen symbool Z kan daarom voor een fasewaarde <> 0° op t=Ts voor het middelste symboolinterval van de replica zorgen. Daarom wordt deze waarde van de fase voor symbool Z op t=Ts van het samengestelde resultaat afgetrokken. DCL=SS[Z,1,2]. De eerste dimensie SS[Z,-,-] geeft het symbool aan, de tweede dimensie SS[-,Y,-] geeft het monsternummer aan en de derde dimensie SS[-,-,X] geeft het Xde symboolinterval van de symboolresponsie aan, er zyn vyf symboolintervallen.
De replica's worden in de matrix BB opgeslagen. BB[I,0] bevat de gebruikte symbolencombinatie.

Procedure : Toestand_Locaties

Hulpprocedures : Fase (intern), Mir (intern), TSTD (intern)

Function Fase(Y) : Als by een gegeven symbool Y de fase van fasepunt (A*π/2) naar fasepunt 0° loopt, dan geeft Function Fase(Y) de waarde A. A,Y ∈ (0,1,2,3,4).

Function Mir(Y) : Geeft van een symbool zyn gespiegelde terug. 0 ↔ 0, 1 ↔ 2, 3 ↔ 4. Y ∈ (0,1,2,3,4).

Function TSTD(X,Y,Z) : Met behulp van TSTD wordt de toestandsnummering voor de toestanden van herkomst en aankomst berekend en opgeborgen in de matrix TT en de matrix XX. De toestandsnummering rond elk fasepunt is op een constante na gelyk opgebouwd. We nummeren de fasepunten P=1,2,3 en 4 , zie tabel 2. Een toestandsnummer wordt nu bepaald door het fasepunt P en de twee opvolgende symbolen A en B, als A het symbool is welke de fase tot fasepunt P heeft geleid, A,B ∈ (0,1,2,3,4). Een toestandsnummer "T" wordt nu bepaald door:

$$T = (P-1)*TPFP + A*int(TPFP/M) + B + 1$$

M is het aantal symbolen in het symbolenalfabet, M=5.
TPFP is het aantal toestanden per fasepunt.

Voor L=3 volgt hieruit T = 25(P-1) + 5A + B + 1
Voor L=2 volgt hieruit T = 5(P-1) + A + 1 B=0
Voor L=1 volgt hieruit T = P A=B=0

De toestandsnummering wordt in de matrix TT en de matrix XX opgeslagen. De derde dimensie van deze matrices komt overeen met een fasepunt, TT[-,-,Z] XX[-,-,Z]. Voor het fasepunt 0° geldt voor deze derde dimensie de waarde z=1. De tweede dimensie duidt op al dan niet gespiegelde replica's, TT[-,Y,-] XX[-,Y,-], voor gespiegelde replica's geldt Y=2 en voor niet gespiegelde replica's geldt Y=1.

In de matrix TT bergen we de nummering van de toestanden van herkomst op. Voor TT[-,1,1] wordt het fasepunt bepaalt door het symbool Y, TT[-,1,1]=TSTD(-,-,Y). De symbolen X en Y bepalen de toestandsnummering rond dit fasepunt.
TT[I,1,1]=TSTD(X,Y,Y) TT[I,2,1]=TSTD(MIR(X),MIR(Y),MIR(Y))

In de matrix XX bergen we de nummering van de toestanden van aankomst op. Voor XX[-,1,1] zyn we op fasepunt 0° (Z=1), XX[-,-,1]=TSTD(-,-,0). De toestandsnummering by dit fasepunt wordt bepaald door de symbolen Y en Z.
XX[I,1,1]=TSTD(Y,Z,0) XX[I,2,1]=TSTD(MIR(Y),MIR(Z),MIR(0))

De overige toestandsnummeringen leiden we uit bovenstaande af.

TT[I,Y,Z]=(TT[I,Y,1] + (Z-1)*TPFP)
XX[I,Y,Z]=(XX[I,Y,1] + (Z-1)*TPFP)

Omdat de fase, modulo(4*TT/2) telt zal de toestandsnummering, modulo(4*TPFP) tellen. De telling loopt nu van 0 t/m 99 voor L=3, door er 1 by te tellen loopt de telling van 1 t/m 100.

Voor L=1 moeten we, voor de matrix TT, een modificatie in de toestandsnummering aanbrengen. Tabel 4 geeft de nummering voor de toestand van herkomst volgens bovenstaand algoritme. Tabel 5 geeft de gewenste modificatie. Via tabel 6 is nu uit een nummer van een toestand van herkomst het ontvangen symbool af te leiden. Voor L=2 en L=3 is zonder modificatie het ontvangen symbool uit de toestandsnummering af te leiden, omdat dit symbool mede de toestandsnummering definieert.

Function ADC

Hulpprocedure Geen

ADC is een kwantisatie bewerking. Het signaal RX is een signaal (een quadratuurcomponent cosϕ of sinϕ dus < 1) vermeerdert met ruis. Het signaal wordt versterkt met de factor SFF en daarna afgerond op de dichtstbyzynde gehele waarde. SF1 is de maximum gehele waarde van het versterkte signaal RX. Tussen SFF en SF1 zit de ruismarge MRGE, zie de procedure Startwaarden.

Procedure Correlator

Hulpprocedure ADC

De matrix UU bevat de monsters van de quadratuurcomponenten van alle symbolen welke door de zender gegenereerd zyn, vermeerderd met ruis. Het aantal monsters in de zender en de ontvanger verschilt minimaal een factor 2, omdat de zender start op t:=0 moet voor het eerste monster in de ontvanger gelden t:=0.5*Ts
Deze factor is de variabele INCR (increment).
De indices van de elementen van de matrix UU lopen van 0 tot 32768 dwz dat de index voor het eerste element van de ontvanger is Z:=1. (INCR DIV 2)
initialisatie waarde: Z:=(SMB-1)*SPSY-(INCR DIV 2)
SMB is een teller variabele voor het aantal symbolen SMB is minimaal gelyk aan 1.
SPSY is het aantal monsters in de quadratuur componenten van de zender. Stel INCR=2 dan is de initiele waarde van Z = -1 in de "loop" is de eerste instruktie Z:=Z+INCR dus Z wordt Z = 1.
UU.R[] wordt per symbool in AA[] gecopieerd (SPSO monsters) en UU.I[] wordt per symbool in DD[] gecopieerd (SPSO monsters).
Voor alle replica's worden nu de factoren S(AC), S(BD), S(BC) en S(AD) bepaald, als A=cosϕ , B=sinϕ , C=cos(θk), D=sin(θk) met ϕ is een signaalrealisatie en θk is een replica, S staat voor het sommatie teken.
RA, RB, RC, RD zyn de niet gekwantiseerde waarden van A, B, C, D.
Kwantisatie vindt plaats mbv de function ADC(--)
Nadat de factoren S(AC), S(BD), S(BC) en S(AD) zyn bepaald vindt "truncatie" plaats. (verlies van de minst significante bits)
Dit geschiedt mbv de operatie S(--) DIV SF2.
SF2 is een schaalfactor, zie procedure Startwaarden.
De matrix CC heeft drie dimensies:
1) Het aantal regels is gelyk aan het aantal replica's:
2) Het aantal kolomen is 2, kolom1 is voor het correlatie-resultaat van de replica en kolom2 voor het correlatie-resultaat van de gespiegelde replica (θk = -θk)
3) Het aantal pagina's is gelyk aan het aantal fasepunten.
De procedure berekent alleen de correlatieresultaten voor de fasepunten 0° en 90° daar de andere resultaten door inverteren kunnen worden bepaald.

$$Zi=COS(\theta_j)\{S(COS\,\phi\,COS(\theta k))+S(SIN\,\phi\,SIN(\theta k))\}+ SIN(\theta_j)\{S(SIN\,\phi\,COS(\theta k))-S(COS\,\phi\,SIN(\theta k))\}.$$

$$Zi = \{1,0,-1,0\}\{S(AC)+S(BD)\} + \{0,1,0,-1\}\{S(BC)-S(AD)\}$$

Zi is het correlatie resultaat.

Function State

Hulpprocedure Geen

State is een conversieprocedure die de coversie uitvoert volgens tabel6 Het zet voor replica's met L=1 de element inhoud uit de matrix TT om in de bybehorende toestand.

Procedure Historie

Hulpprocedure Geen

De correlatieresultaten voor de fasepunten 180° en 270° worden bepaald door de resultaten van 0° en 90° te inverteren.
Met behulp van de elementen uit de matrix TT worden by de elementen uit de matrix CC elementen uit de matrix LL opgeteld.
Als we over replica's met L=1 beschikken (TSTN=4) dan dient de coversie "function State" op het betreffende element uit de matrix TT te worden toegepast, omdat de inhoud van deze elementen dan zowel door de toestand als door het symbool worden bepaald.

Procedure Symbol

Hulpprocedure State

In de matrix SS wordt naar het ontvangen symbool gezocht mbv een zoekroutine. Het symboolnummer is het startnummer van de kolom in de matrix SS. De startregel van de matrix SS vinden we in de matrix VV op elementnummer = symboolnummer. De inhoud van het nu gevonden SS-element wyst naar een regelnummer van de kolom met kolomnummer = symboolnummer-1. Zo vinden we een nieuw SS-element en de routine herhaalt zich net zolang als het observatieinterval diep is. Het zo uiteindelyk gevonden SS-element is ,evenals alle SS-elementen een toestandsnummer. Het by dit toestandsnummer horende symbool is het gedetekteerde symbool. Voor replica's met L=1 moet telkens op een gevonden SS-element de Function State toegepast worden om de toestand uit de inhoud van dit element te herleiden.
Het case-statement binnen de procedure bepaalt uit een SS-element het bybehorende symbool. Het correcte symbool vinden we in de matrix YY (door de zender gegenereerd) en wel op element YY[SMB-OBIN-1] (=YY[LB-1]). Een symboolfout is gemaakt als het verschil ABS(YY[LB-1]-Y) = 4 en een bitfout is gemaakt als dit verschil 2 of 6 bedraagt. Deze fouten worden opgeslagen in de matrix FF en de matrix ZZ.

Procedure Ruisfilter

Hulpprocedures G05DDF, G05CBF, FFTC_1, FILTRMXFL, FFTC.

Mbv de Ruisgeneratoren G05DDF en G05CBF (normale verdeling) wordt ruis aan de quadratuurcomponenten uit de matrix AB toegevoegd. Hierna vindt Butterworth filtering plaats mbv de externe procedures FFTC_1 hetgeen een transformatie naar het frekwentiedomein is , FILTRMXFL is en Butterworth filtering, en FFTC hetgeen een transformatie terug naar het tyddomein is.
G05DDF en G05CBF zyn externe procedures uit de algemene bibliotheek [NAGLIB]NAG/LIB. (FORTRAN)
FFTC_1, FFTC en FILTRMXFL zyn externe procedures uit de Module Zendersimulatie(Pinp,Pout).

Procedure Startwaarden

Hulpprocedure Geen

Startwaarden bepaalt een aantal schaalfactoren die samenhangen met de kwantisatie gegevens.
Het correlatieresultaat hangt af van het aantal monsters in de quadratuurcomponenten van de ontvanger. (SPSO)
KBSC is het aantal kwantisatiebits voor de quadratuur-componenten, hier gaat een tekenbit vanaf.
SF1 = 2**(KBSC-1)-1. By bv KBSC is 4 bits zyn de gekwantiseerde quadratuurcomponenten sinϕ en cosϕ maximaal gelyk aan zeven.
Stel nu A=2**(KBSC-1) en B=2**(KBSL-1)
Dan geldt max{Zi} = SPSO*A**2. Zi is een correlatieresultaat.
Alvorens "overflow" begrenzing van de L-registers toe te passen moeten er minimaal 4 optellingen van correlatieresultaten hebben plaats gevonden. Zodat max{4*Zi} = 4*SPSO*A**2
Dit maximum willen we nu met (KBSL-1) bits weergeven.
Het aantal bits in max{4*Zi} bedraagt: 2*KBSC+dilog(SPSO).
Is dit aantal groter dan (KBSL-1) dan verwaarlozen we de laagst significante bits. KBSL =< 2*KBSC+dilog(SPSO)+1
De truncatiefactor bedraagt: SF2=(4*SPSO*A**2)DIV(B**2).
SF2 is altyd een macht van 2.
Dilog(SF2) = 2*KBSC-(KBSL-1)+dilog(SPSO) .
De inhoud van de L-registers moet ten alle tyden kleiner zyn dan (4*SPSO*A**2)DIV(SF2) We kiezen daarom
LMAX=ROUND(3*(SPSO*(SF1**2))/SF2)
We testen Li>LMAX, Li kan dan max 4*Zi aannemen hetgeen het absolute max. is.

Procedure Ontvanger

Hulpprocedures Alle

De factor RSDV zal worden toegelicht.

RSDV Ruisdeviatie
GSNR Genormaliseerde Signaal Ruis Verhouding
BPSY Bits per symbool
SPSY monsters per symbool in de quadratuur componenten van de zender
SPBT monsters per bit in de draaggolf van de ontvanger.

Draaggolf K*COS(ω_0t+ϕ) Psign. = 1 {genormaliseerd}
Ruis Uc.COS(ω_0t)+Us.SIN(Wot) Pruis = RSDV**2

De ruisbandbreedte = SPBT/(Tb*2) (Onafhankelyke ruis.)

SPBT = SPSY/BPSY

GSNR = 10*log (Signaalvermogen)/(Ruisvermogen per Fbit-bandbreedte)

GSNR = 10*log (Signaalenergie per bit)/(Ruisvermogen per Hz) [dB]

GSNR = 10*log (Ps*Tb)/(Pr*Tb*2*BPSY/SPSY)

RSDV = SQRT (SPSY/(BPSY*2*10**(GSNR/10)))

3. APPENDIX DECLARATIES

BY HET AFSTUDEERVERSLAG:

ONTWERP EN SIMULATIE,
VAN EEN CORRELATIEONTVANGER
MET VITERBI PROCESSOR, VOOR
"CORRELATIVE PHASE SHIFT KEYING"
---------- CORPSK(4-5) ----------

AFSTUDEERFASE OKT '81 - JUNI '82

PHILIPS HUIZEN - TH-DELFT

5 JULI 1982

B WIJNE

LARENSESTRAAT 24

070 - 465326

2574 VJ DEN HAAG

035 - 894492 (PHILIPS)

Verklaring van de declaraties binnen BWHEADS.PAS .

VRZ1 Is een verzameling van in te lezen variabelen voor de initialisatie van de ontvanger.

C_ORDV Keuzevariabele voor de simulatie met of zonder Viterbi processor, ORDV=0 betekent zonder, ORDV= 1,2 of 3 betekent een Viterbi processor met L= 1,2 of 3.

C_FPTN Variabele voor het aantal fasepunten.

C_MRGE Variabele voor de ruismarge ten opzichte van de grootste gehele waarde by gegeven kwantisatie.

C_SPSO Variabele voor het aantal monsters in de quadratuur-componenten van de ontvanger.

C-LBIN Grenzen waarbinnen het observatieinterval van het
C_OBIN Viterbi algoritme ligt. Alle waarden tussen LBIN en OBIN worden een keer als observatieinterval gebruikt.

C_LBRE Grenzen waarbinnen de ruisrealisaties liggen, als het
C_RSRE aantal symbolen = 1024 en LBRE = 1 en RSRE = 3 dan worden 3072 symbolen door de ontvanger ontvangen, nl 3 x 1024 symbolen elk van de 1024 symbolen met een andere ruisrealisatie.

C_FOLT Folt is "yes" of "no" al of niet worden uitgevoerd de matrix TT, de matrix XX en hoeveel bitfouten en symboolfouten er per symbool zyn gemaakt door de ontvanger. Uitvoer is { - symb.no - X1 X2 } X1 is het aantal bitfouten en X2 is het aantal symboolfouten.

C_GSNR Invoer van de genormaliseerde signaal-ruis verhoudingen. Ingevoerd moet worden a) het aantal, b) de waarden in dB.

C_GSNR := 3 0 6 10 betekent dat het programma een uitvoer geeft voor 3 GSNR waarden nl voor: GSNR := 0 dB, GSNR := 6 dB en voor GSNR := 10 dB.

C_RSGM Variabele voor het ruisgemiddelde.

C_KBSL Variabelen voor de kwantisatie gegevens. KBSC is het
C_KBSL aantal bits voor de quadratuurcomponenten. KBSL is het aantal bits na correlatie. Het verschil tussen KBSL en KBSC moet minimaal 3 bits bedragen. Vb van invoer:

C_KBSC := 3 3 4 8
C_KBSL := 3 6 8 16

Het programma zal 3 outputs geven resp voor de volgende combinaties [KBSC,KBSL], [3,6],[4,8] en [8,16].

OGNN Het programma zal nu 5*2*32 getallen inlezen. Deze getallen vormen de stapresponsie van het premodulatie filter over 5 symboolperioden en 32 monsters per symboolperiode. De 2 is noodzakelyk omdat voor elk monster een nummer aanwezig is.

FLTR Na deze variabele verwacht het programma 4 variabelen resp. van een Butterworthfilter, 1) de orde, 2) de bandbreedte tov Fbit 3) De centrale frekwentie van het filter tov Fbit en 4) De "frequency shift" van het filter.
Na deze variabelen geplaatste tekst, op dezelfde regel wordt genegeerd door het programma.

VRZ2 Verzameling waarin de elementen die de variabele "FOLT" kan aannemen staan.

Type declaraties. (Zie ook appendix Pascal)

Type Complex = Record R,I:Real end;
Aan elke variabele van het Type complex wordt een tweetal dimensies toegevoegd. bv. Var A:Complex betekent dat er een variabele A.R en een variabele A.I bestaat, beide van het type Real. (Reele en Imaginaire deel)

Type DIM(X) en DM(X) zyn subranges. Bv DIMAB = -1..32768
32x1024 = 32768 dit staat voor maximaal 1024 symbolen van 32 monsters in de zender te genereren. DIMCS = 0.5 X DIMAB.
DIMY = 4096. Dwz maximaal 1024.symbolen van 4 bits.
Alle overige DM(X) hebben te maken met de ontvanger. Hun grote is afhankelyk van de variabele die erachter tussen tekstakkolades is vermeld.

Algemene matrix declaraties.

In het kort zal de funktie van de matrices worden aangegeven.

AB- Bevat alle monsters van alle symbolen van de quadratuur componenten I:sinϕ R:cosϕ Het totaal aantal monsters is afhankelyk van Symb x SPSY. Symb= het aantal symbolen en SPSY= het aantal monsters per symbool in de zender.
CS- Hulpmatrix by de matrix AB of de matrix UU
UU- UU[] = AB[] + Ruis
AA- Copy van steeds SPSO monsters uit UU.R
DD- Copy van steeds SPSO monsters uit UU.I
BB- Opslag van de replica's.
CC- Opslag van correlatieresultaten, correlatie tussen AA,DD en BB.
FF- Opslag van het aantal bitfouten per observatie interval.
LL- Opslag van de toestandsregisters Li.
QQ- Leest drie ASCII-karakters in, bedoeld om ":=" te lezen.
RR- Bevat gegevens van: GSNR, KBSC en KBSL.
SS- Het aantal regels is gelyk aan het aantal toestanden en het aantal kolommen is gelyk aan het aantal symbolen. Bevat per symbool (dus per kolom) de survivor per toestand (elke regel een survivor). Elke survivor per toestand is een toestand van herkomst en wyst dus naar een regelnummer in de kolom voor de huidige kolom. Zo vormt de SS-matrix een aantal Fasetrajecten gelyk aan het aantal toestanden. (Een survivor is een toestand behorende by een maximaal samengesteld correlatieresultaat)
TT- Toestandsmatrix de inhoud per element is een indexnummer van een toestandsregister Li (matrix LL) Zo geeft de matrix TT aan welk L-register (matrix LL) er by welk element uit de matrix CC moet worden opgeteld. CC[i,j,z]:=CC[i,j,z]+LL[TT[i,j,z]].
VV- Is de matrix waarin de survivor per symbool staat, dit is het start-element voor de betreffende kolom van de matrix SS.
XX- Toestandsmatrix. De inhoud per element is een indexnummer van een toestandsregister Li (matrix LL). Zo geeft de matrix XX aan welke elementen uit de matrix CC by een nieuwe toestand horen.
YY- Bevat de symbolenreeks welke door de zender is gegenereerd.
ZZ- Bevat de bitfouten en symboolfouten per symbool ZZ[i,j] index i staat voor het symboolnummer en index j is 1 of 2 en geeft aan resp. het aantal bitfouten of het aantal symboolfouten.

Variabelen declaraties.

GEG1 Variabele welke een waarde uit VRZ1 kan aannemen.
FOLT Variabele welke een waarde uit VRZ2 kan aannemen.
INCR De zender en ontvanger werken niet noodzakelykerwys met hetzelfde aantal monsters. INCR:=(SPSY DIV SPSO).
BPSY Variabele voor het aantal bits per symbool.
SPSY Variabele voor het aantal monsters in de quadratuur componenten van de zender.
SPSO Variabele voor het aantal monsters in de quadratuur componenten van de ontvanger.
OBIN Zie C_OBIN.
LBIN Zie C_LBIN.
TSTN Variabele voor het aantal toestanden binnen het Viterbi Algoritme.
FPTN Zie C_FPTN.
SYMB Variabele voor het aantal symbolen.
SMB Teller variabele voor het aantal symbolen.
GSNR Teller variabele voor het aantal genorm. SNR-waarden.
KBS Teller variabele voor het aantal kwantisatie gegevens.
ORDE Variabele voor het Butterworthfilter, zie FLTR
ORDV Zie C_ORDV.
BAND Idem.
CARR Idem.
SHFT Idem.
SF1 Schaalfactor voor de kwantisatie.
SF2 Idem.
SFF Idem.
BTSM =N Parameters uit de zendersimulatie N is het totaal
BTSN =M aantal monsters : N := SYMB X SPSY en M := dilog (N)
LMAX Variabele voor overflow begrenzing van de L-registers
RSRE Zie C_RSRE.
LBRE Zie C_LBRE.
MRGE Zie C_MRGE.
RSGM Zie C_RSGM.
RUIS Teller variabele loopt van LBRE t/m RSRE
RSDV Variabele voor de ruisdeviatie.
ORGN Variabele voor het aantal replica's wat gegenereerd wordt.
TPFP Variabele voor het aantal toestanden per fasepunt.

4. APPENDIX PASCAL

BY HET AFSTUDEERVERSLAG:

ONTWERP EN SIMULATIE,
VAN EEN CORRELATIEONTVANGER
MET VITERBI PROCESSOR, VOOR
"CORRELATIVE PHASE SHIFT KEYING"
---------- CORPSK(4-5) ----------

AFSTUDEERFASE OKT '81 - JUNI '82

PHILIPS HUIZEN - TH-DELFT

5 JULI 1982

B WIJNE

LARENSESTRAAT 24

070 - 465326

2574 VJ DEN HAAG

035 - 894492 (PHILIPS)

Een in pascal programmeertaal geschreven programma bestaat uit de volgende onderdelen:

a) Program heading.
Dit is de eerste regel. Hierin staat de naam van het programma en als parameter list de te gebruiken externe files.
Bv. Program Acquisitie(Pinp,Pout);

b) Declaratie blok.
Direkt na de Program heading vinden we het declaratie blok. Hierin worden alle in het programma te gebruiken grootheden gedeclareerd.

c) Program body.
Het eigenlyke programma bestaat uit een aantal "statements" vooraf gegaan door: "Begin{0}" en afgesloten door "End{0}." .

Een programma is op te delen in een aantal subprogramma's, procedures genaamd.

Een procedure bestaat net als een programma uit:
a) Procedure heading.
b) Declaratie blok.
c) Procedure body.

Grootheden binnen een procedure gedeclareerd zyn buiten die procedure niet zichtbaar, maar wel binnen procedures, aangeroepen binnen die procedure (side effect).

De programmeertaal Pascal kent statements (instructies) en compound statements (meervoudige instrukties). Een compound statement is een samenvoeging van een aantal statements tussen de instructies "Begin" en "End" . Statements worden gescheiden door punt komma's (;). Een voorbeeld van een statement is: x:=cos(y)+sin(z); Een voorbeeld van een compound statement bestaande uit twee statements is:

```
Begin{1}  x:=cos(y)+sin(z);
          y:=cos(p)+sin(q)   End{1};
```

De toevoeging {1} aan Begin en End is niet noodzakelyk, het vergroot echter de leesbaarheid van een programma, daar zichtbaar is welke Begin en End by elkaar horen. Begin en eind van "verklarende tekst" wordt aangegeven met accolades {}, alles wat tussen {} staat wordt door de compiler genegeerd.
Een compound statement wordt door de compiler als een statement gezien.

Overdracht van variabelen tussen procedures.

Willen we variabelen van een procedure naar buiten brengen dan kan dat op twee manieren.

1) De variabele wordt niet in de procedure gedeclareerd maar in het algemene declaratie blok van het programma. De variabele is dan binnen elke procedure zichtbaar en veranderbaar.

2) De variabele wordt via de parameterlist (pm list) van de procedure heading overgedragen.
Nadeel van 1) is nl dat de naam toekenning aan de variabele eenmalig is en dus permanent.

In een parameter list kennen we lokale- en globale variabelen. Lokale variabelen worden alleen voor het gebruik binnen de procedure overgedragen, de variabele wordt buiten de procedure niet door de procedure gewyzigd. Globale variabelen, zyn variabelen die, door de procedure gewyzigd, ook buiten die procedure gewyzigd zyn, ze zyn als het ware overschreven door de procedure.
Voorbeeld van een procedure met parameter list :

```
Procedure Test(x,y:Integer; Var p,q:Real);
```

x en y zyn lokale variabelen van het Type integer.
p en q zyn globale variabelen van het Type real.
Onderscheid wordt gemaakt met de instruktie "Var" , Var blyft geldig tot de eerst volgende puntkomma.
Aanroep van de procedure:

```
Test(a,b,c,d);

executie (a → x , b → y , c → p , d → q), Begin procedure,
                         (p → c , q → d)  End;
```

Functions.

naast procedures kennen we functions. Het verschil tussen een procedure en een function is dat een function op zich een variabele is. Byvoorbeeld:

```
Function Test(pm list):Real;
Declaratie blok;
Begin{0}  - Statements -  Test:=X  End{0};
```

De aanroep is nu byvoorbeeld: Y:=Test(pm list);

Declaraties

Een declaratie blok bestaat voornamelyk uit de volgende delen:

a) Constanten declaraties
b) Type declaraties
c) Var(iabelen) declaraties

Type Text = File of Char; is een standaard declaratie, dit Type behoeft dus niet apart gedeclareerd te worden.

a) Constanten declaraties.

- Const naam = const waarde -

```
PI=3.1415926535897 93
```

b) Type declaraties.

- Het Type Scalar -

```
Type VRZ1=(Rood, Wit, Blauw, Paars);
Var  VERF1,VERF2 : VRZ1;
```

VERF1 en VERF2 zyn nu variabelen waaraan een element uit VRZ1 toegekend kan worden.

- Het Type Array -

```
Type MTRA=Array[I] of T;
Var  AA : MTRA;
```

I is een subrange bv. I=1..10, T is een Type. Variabele AA is nu van het Type Array. De elementen van AA zyn van het Type T.

- Het Type Record -

```
Type Complex=Record I,R : Real  End{record};
Var  X1 : Complex;
```

Variabele X1 valt nu in twee delen uiteen.
X1.R is een variabele van het Type Real en
X1.I is een variabele van het Type Real.

- Het Type Subrange -

```
Type DIMAB=  1..10 ;     (subrange van Integer).
Z='x'..'z';     (subrange van Char    ).
```

c) Variabelen declaraties

- Variabele naam : Type -

"Variabele" declaraties kunnen varieren van simpel:

```
A : Integer;  tot   RUIS : Real;
```

Waarby we door de namen goed te kiezen min of meer de betekenis van de variabele aangeven.

Files

By gebruik van externe files moeten deze in de Program heading vermeld worden. Bovendien moeten andere files m.u.v. Output en Input, gedeclareerd worden. Interne Files worden niet vermeld in de Program heading, maar moeten wel gedeclareerd worden.

```
Voorbeeld: Program X(Input,Output,Pons);
           Var Pons : Text;
```

Input, Output en Pons zyn externe files van het Type Text.

Alvorens in een file geschreven kan worden moet de initialisatie instructie "Rewrite(filename)" gegeven worden en voor lezen in een file, "Reset(filename)"; dit zyn "pointer set" instrukties.

Algemeen gebruikte statements:

```
WHILE B DO S;
REPEAT S1,....Sn UNTIL B;
FOR C:=E1 TO      E2 DO S;
FOR C:=E1 DOWNTO E2 DO S;

IF B THEN S1 ELSE S2;

CASE I OF T1:S1 ; T2:S2 ; .... Tn:Sn
OTHERWISE Sm   END{CASE};

WRITE(FILENAME,'textstring');   WRITELN(  );
WRITE(FILENAME,V:M:FL);         WRITELN(  );
READ(FILENAME,VARIABELEN);      READLN(   );
```

B is een boolean variabele.
Si is een procedure, een statement of een compound statement.
Ei,C zyn variabelen van het Type Integer.
V is een variabele van het Type Real.
M is het totaal aantal velden inclusief punt en teken.
FL is het aantal fracties (aantal plaatsen na de punt).
I $\in$ (T1...Tn) V $\notin$ (T1...Tn)
als I $\in$ (T1...Tn) wordt de bybehorende Si uitgevoerd.
als I $\notin$ (T1...Tn) wordt Sm uitgevoerd.

Operands.

```
ABS(E); ROUND(E); TRUNC(E) ;
SIN(E); COS(E)  ; ARCTAN(E);
SQR(E); SQRT(E) ; LN(E)    ; EXP(E);
(X**Y); (X*Y)   ; (X+Y)    ; (X-Y) ; (X/Y) ;
```

Integer operands.

A DIV B ; Deling zonder restwaarde.
A MOD B ; Restwaarde van een deling.

Boolean operands.

```
AND ;  OR ;   NOT ;
```

Programma vormen.

In het algemeen bestaat een programma nu uit:

1) - Program heading -
2) - Declaratie blok -
3) - Procedures en/of Functions -
4) - Program body -

De eenvoudigste vorm van een Program body is:

Begin{O} Startprocedure End{O}.

De Program body roept dan de startprocedure aan waarmee begonnen wordt.

Een Module is een programma zonder Program body.

In het algemeen bestaat een module uit:

1) - Module heading -
2) - Declaratie blok -
3) - Procedures en/of Functions -

Modules worden apart gecompileerd.

Een programma dat gebruik maakt van een procedure uit een andere module declareert deze procedure als extern. Alleen de procedure heading met de instruktie extern eraan toegevoegd wordt in de declaratie opgenomen.

Voorbeeld:

- Program heading -
- Declaratie blok -
- Procedure Test(pm list); extern; -
- Program body -

Een Program en een Module moeten dezelfde "file list" en dezelfde algemene declaraties bezitten.

Het creeren van een "executable program" kent nu de volgende systeem stappen:

1) Het invoeren van een programma op een file met behulp van de teksteditor. Het Type van de file is ".PAS" dus Filename.PAS wil zeggen dat het een te compileren file is.

2) Na compilatie is het Type van de file veranderd, in ".OBJ" dus Filename.OBJ wil zeggen dat het een gecompileerde file is.

3) De Program.OBJ files en de Module.OBJ files moeten nu tot een "executable program" worden gevormd via een "link" opdracht. De file krygt de naam van de eerst genoemde file in de "Link" opdracht en het Type wordt nu ".EXE" Filename.EXE wil zeggen dat er van een "executable program" sprake is.

Input en Output files zyn van het Type ".DAT" dus Filename.DAT

Een opdracht "RUN Filename.EXE" stelt het programma nu in werking.

5. APPENDIX PROGRAMMATUUR

BY HET AFSTUDEERVERSLAG:

ONTWERP EN SIMULATIE,
VAN EEN CORRELATIEONTVANGER
MET VITERBI PROCESSOR, VOOR
"CORRELATIVE PHASE SHIFT KEYING"
---------- CORPSK(4-5) ----------

AFSTUDEERFASE OKT '81 - JUNI '82

PHILIPS HUIZEN - TH-DELFT

5 JULI 1982

B WIJNE

LARENSESTRAAT 24

070 - 465326 .

2574 VJ DEN HAAG

035 - 894492 (PHILIPS)

5:1 Inputfile; PINP.DAT

```
ANEW_WEER
                corps (4_5) BW gegevens
                14  2  4      M,BPSY,MSPSY
REEKS_GEN
                gemodificeerde reeks
                13
                -1 -3  3  3  3  3  3  3  3  3  3 -3 -3
LEVELSHIFT
                nivo vermeerdering. 4_5
                 4     0     topniivo, aantal, symbols.
IMP_INT
                omzetting in freq spectrum
NYQ_III
                nyquist filtering
                0.5  alpha
XPT_TIJD
                omzetting in tijdsignaal
PHASE_DEV
                instelling juiste fase verandering.
                1
IQ_STR
                omzetting naar MF spectrum met storing
                -1.0  0.7    amplitude en frekw
FLTR
                10  2  9  0  orde bb carrier frekshift


C_KBSC := 1     6
C_KBSL := 1     12
C_GSNR := 2     4  5

C_LBIN := 1    C_OBIN := 52

C_ORDV := 2    C_SPSO := 8
C_LBRE := 1    C_RSRE := 6
C_FPTN := 4    C_MRGE := 1.5
C_RSGM := 0    C_FOLT := NO
```

320	1.00000	321	0.99943	322	0.99872	323	0.99788
324	0.99691	325	0.99580	326	0.99458	327	0.99325
328	0.99182	329	0.99030	330	0.98873	331	0.98712
332	0.98549	333	0.98388	334	0.98232	335	0.98083
336	0.97946	337	0.97823	338	0.97719	339	0.97637
340	0.97581	341	0.97554	342	0.97561	343	0.97603
344	0.97685	345	0.97810	346	0.97979	347	0.98194
348	0.98458	349	0.98770	350	0.99131	351	0.99542
352	1.00000	353	1.00504	354	1.01052	355	1.01639
356	1.02263	357	1.02917	358	1.03597	359	1.04295
360	1.05005	361	1.05718	362	1.06425	363	1.07118
364	1.07787	365	1.08421	366	1.09009	367	1.09541
368	1.10005	369	1.10390	370	1.10684	371	1.10878
372	1.10958	373	1.10916	374	1.10740	375	1.10422
376	1.09952	377	1.09323	378	1.08528	379	1.07560
380	1.06414	381	1.05087	382	1.03577	383	1.01881
384	1.00000	385	0.97936	386	0.95691	387	0.93270
388	0.90678	389	0.87922	390	0.85012	391	0.81955
392	0.78764	393	0.75449	394	0.72025	395	0.68505
396	0.64904	397	0.61236	398	0.57519	399	0.53768
400	0.50000	401	0.46232	402	0.42481	403	0.38764
404	0.35096	405	0.31495	406	0.27975	407	0.24551
408	0.21236	409	0.18045	410	0.14988	411	0.12078
412	0.09322	413	0.06730	414	0.04309	415	0.02064
416	0.00000	417	-0.01881	418	-0.03577	419	-0.05087
420	-0.06414	421	-0.07560	422	-0.08528	423	-0.09323
424	-0.09952	425	-0.10422	426	-0.10740	427	-0.10916
428	-0.10958	429	-0.10878	430	-0.10684	431	-0.10390
432	-0.10005	433	-0.09541	434	-0.09009	435	-0.08421
436	-0.07787	437	-0.07118	438	-0.06425	439	-0.05718
440	-0.05005	441	-0.04295	442	-0.03597	443	-0.02917
444	-0.02263	445	-0.01639	446	-0.01052	447	-0.00504
448	0.00000	449	0.00458	450	0.00869	451	0.01230
452	0.01542	453	0.01806	454	0.02021	455	0.02190
456	0.02315	457	0.02397	458	0.02439	459	0.02446
460	0.02419	461	0.02363	462	0.02281	463	0.02177
464	0.02054	465	0.01917	466	0.01768	467	0.01612
468	0.01451	469	0.01288	470	0.01127	471	0.00970
472	0.00818	473	0.00675	474	0.00542	475	0.00420
476	0.00309	477	0.00212	478	0.00128	479	0.00057

++++++++++++ ANEW_WEER ++++++++++++++++
corps (4_5) BW gegevens

1024 SYMBOLS, 2 BITS/SYMBOL, 16 SAMPLES/SYMBOL,

++++++++++++ REEKS_GEN ++++++++++++++++
gemodificeerde reeks

UITKOMST CONTROL <>0: 24

++++++++++++ LEVELSHIFT ++++++++++++++++
nivo vermeerdering. 4_5

TOPLEVEL 4

0-2-4-4-4-4-4-4-4-4-4-2-2 0 4-2 2 4 4-2 2 4 2 0-4 2 4-2-2-4-2 0
4 4 2-2-2-4-2 4 4 4 2-2 0 2 0 2 2 2 0 0 2 4 2 2-2-4 2 0-2-2 0 2
2-2-2-2-4 2 2 2 2 4-2 0-2 0-2 2 2-2-2 2 4 2 2 2-4 2 0-2 0-2-4-4
0 0 4 2 0-4 0 2 0 0 2-2-4-4 0 2 2 4-2 2 0-2 0 4 2 2-2-2-4-2 0 0
0-2 2 2-2 2-2 0-2-2 0 0 0 4 4 4 0-4-2 4-2 2 0-4 2-2 0 0 2 0 0 0
-4-2-2 2-2-4 0 0-2-2 0 4 0-4-4-4-2 2 0 2 0-4 2 2 0 0 2 0-2-2 2 0
0-4-4 2 2-2-4-2 4 2 2-4 0-2-4 2 4 0-4 2 0-2 2 0 2 2-2 0 0 2 4 4
4 0 2 4-2-4 0 2 4 0-2 2 0-2 2-2 0 0 4 2 2 2-2-4-2 0-2-2 0 4 4 0
0 0 2 0-4 0 4-2-2-4 0 2-2-2 0 4-2-2-2-2-4 0 0 0 0-2 2-2 2-2 2 4
4 2 2-4 2 0 2 4 2 0 0 2 4-2-2 2 0-2 2 2 4 4 0-2 0-2-4 0 4 4 0-2
2 0 2-2 2 4 4-2-2 0-2-4 0 0 0 4 2-2 2-2-4-2-2 0 0-2 2 2 2-2 2 4
2-4-4 2 4-2 0 2 0-2 2 2 0 0-4 2 4 4 0 2 0-2-4 0 2 2-2 0 4-2 0 0
0 2-2-2 2-2-4-4-4 2 2 4-2-4-2 0-2 2 4 0 0-4 2 4 0-4-2 4 2-2-4 0
-2-4-4 2-2 2 4-2-4-4 2 4 0-2-4-2 0 4 0-2 2 2 0-4 0-2 0 0 2-2 2 2
-2-4-4 2 4 4 2 0 2 4-2-4-4-2 0 4 2 0-4-4-2 4 4-2-4 0-2-4 0-2 2-2
0 4 0-2-2-2-4 0 4 4 4 2-2-4-2 4-2-2 0 4 2-2-2-2-4-2-2-2-2 0 4 4
4 4 4-2-4-2 4-2 2 4 0-2 2 4 2-2 2 4 2 0 0-2 0 2 2-2 2 2 0-2-2 0
-2-2 2 2 2-2-2 0-2 0 0 0 0 4 4 0-4 0 2 0-4-2 2 0 0 2-2 0 0 4-2-2
-2 2 0-4-4-4 2 2 0 2 0 2 4 4-2-2-4 2 0-4-4 2 0 0 2 0 2 2-2-4-4-2
0 0-2-4 0 0 4 0-2 2-2-4 0 4 2 2-4 0 2 0-2 0 4-2-2 2 2 0-4-4 2 4
4-2-4-2 0 2-2 0 4 4-2-2-2-4-2 2 2 2 4 0-2 0-2 0 4 0 0-4-4-2 2-2
-4-2 2 4 4 2-2 0-2-4-2-2-2 2 4 0 0 0-2 0 4 0-4 0 0 2-2-4 0 4-2-2
0 4-2 2 2 2 0-4 2 4 2 2 0 2 4-2 0 0 2 4 0 0-4-2 0 4 0 2-2-2-4 0
2 2 2 4 0 2 4 2-4 0 2 4-2 0 4-2 0 4 4-2 2 2 0 2-2 0-2-2-4-4-4 0
0-2-4-2 2-2 2 4 0-2-2 0-2 2-2-2-2 2 4 4 4 4 2-4-2 4-2-4 2-2 0 4
2 0 0 0 2 0 0-4 0 2 2-2-4 0 2 4 4 2-2-4 2 0 2 2 4-2-2-4 2 4 0 0
-2-4 2-2 2-2 0 2 2 4 4 4-2 0-2-4-2 2 2-2 0 4 2-4-4-4-2 4 2 0 2-2
0 2 2 0 0 0 2 4 4 0-4-2 0 2-2-4 0 0 2 2 4 0-4-2 0-2 2 0-4-4 0-2
-2-4-2 2-2-2 0 4 2 2 2 2 2 0 2 4 2-4-4-2 0 2 4-2 2 2 0-2 2 4 2 2-2
0-2-4 2 2 2-2 0 2 4 2 2.2 0-2-4 2 4 4 0-2-4 2 0-4 0-2-4-4-2 2-2
0 2-2 0 0 0 2 2 2-2 2 0-2 0 0-2-2 2 2-2 2 2 4 2 2-4 2 4-2 0-2-4
2-2-2 2 4-2-2-2 0-2 2 2 2 2-2 0-2 0-2-2-2 2 2-2-2-2 0-2-2-2-2-2

```
++++++++++++        IMP_INT  +++++++++++++++
              omzetting in freq spectrum


++++++++++++        NYQ_III  +++++++++++++++
              nyquist filtering

ALFA VOOR NYQIII:       0.50000

++++++++++++      XPT_TIJD  +++++++++++++++
              omzetting in tijdsignaal


++++++++++++     PHASE_DEV  +++++++++++++++
              instelling juiste fase verandering.

 FACTOR:      1.00000

++++++++++++         IQ_STR  +++++++++++++++
              omzetting naar MF spectrum met storing

STOORSTERKTE:      -1.00000    STOORFREQ:       0.70000



INITIALISATIE VAN DE ONTVANGER:

C_LBRE :=     1       C_RSRE :=     1
C_RSGM :=    0.00     C_FPTN :=     4
C_MRGE :=    1.50     C_SPSO :=     8
C_LBIN :=     1       C_OBIN :=    52

  TSTN :=   100          ORGN :=    48

C_KBSC :=    6
C_KBSL :=   12
C_GSNR :=    2


FILTER:  ORDE= 10 CARR= 9.0 SHFT= 0.0 BAND= 2.0  Verh tov Fbit
```

ORIGINELEN:

0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100	-0.04	-0.09	-0.11	-0.11	-0.09	-0.06	-0.04	-0.01
300	-0.07	-0.17	-0.21	-0.21	-0.18	-0.13	-0.07	-0.02
1	0.01	0.04	0.06	0.09	0.11	0.11	0.09	0.04
101	-0.03	-0.05	-0.04	-0.02	0.02	0.04	0.05	0.03
301	-0.06	-0.13	-0.15	-0.12	-0.07	-0.02	0.01	0.01
102	-0.05	-0.12	-0.17	-0.20	-0.20	-0.17	-0.12	-0.05
302	-0.08	-0.21	-0.28	-0.30	-0.29	-0.24	-0.16	-0.06
3	0.02	0.07	0.13	0.18	0.21	0.21	0.17	0.07
104	-0.06	-0.16	-0.24	-0.29	-0.30	-0.28	-0.21	-0.08
304	-0.09	-0.24	-0.34	-0.39	-0.39	-0.34	-0.24	-0.09
10	0.96	0.85	0.72	0.58	0.42	0.28	0.15	0.04
110	0.92	0.76	0.61	0.47	0.33	0.22	0.11	0.03
210	0.99	0.94	0.83	0.68	0.51	0.34	0.19	0.05
310	0.89	0.68	0.51	0.36	0.24	0.15	0.08	0.02
410	1.03	1.02	0.94	0.79	0.60	0.41	0.22	0.06
11	0.97	0.89	0.78	0.67	0.53	0.39	0.24	0.08
111	0.93	0.80	0.68	0.56	0.44	0.32	0.20	0.07
211	1.00	0.97	0.89	0.77	0.62	0.45	0.27	0.09
311	0.90	0.72	0.57	0.45	0.35	0.26	0.16	0.06
411	1.04	1.06	1.00	0.88	0.71	0.52	0.31	0.10
12	0.95	0.81	0.66	0.49	0.32	0.17	0.06	0.01
112	0.91	0.73	0.55	0.38	0.23	0.11	0.03	0.00
212	0.98	0.90	0.76	0.59	0.41	0.24	0.10	0.02
312	0.87	0.64	0.44	0.27	0.14	0.04	-0.01	-0.01
412	1.02	0.98	0.87	0.70	0.50	0.30	0.14	0.03
13	0.98	0.92	0.85	0.76	0.64	0.49	0.32	0.11
113	0.94	0.84	0.74	0.65	0.55	0.43	0.28	0.10
213	1.01	1.01	0.96	0.86	0.73	0.56	0.36	0.13
413	1.05	1.09	1.06	0.97	0.82	0.62	0.39	0.14
114	0.90	0.69	0.48	0.29	0.12	0.00	-0.06	-0.04
314	0.86	0.61	0.38	0.18	0.03	-0.06	-0.09	-0.05
30	1.91	1.70	1.44	1.15	0.85	0.56	0.30	0.09
130	1.88	1.61	1.33	1.04	0.76	0.50	0.26	0.08
230	1.95	1.79	1.55	1.26	0.94	0.62	0.34	0.10
330	1.84	1.53	1.23	0.94	0.67	0.43	0.23	0.07
31	1.92	1.74	1.50	1.24	0.96	0.67	0.39	0.12
131	1.89	1.65	1.40	1.13	0.87	0.60	0.35	0.11
231	1.96	1.82	1.61	1.35	1.05	0.73	0.42	0.13
331	1.85	1.57	1.29	1.03	0.78	0.54	0.31	0.10
32	1.90	1.66	1.38	1.06	0.74	0.45	0.21	0.05
132	1.87	1.58	1.27	0.95	0.65	0.39	0.18	0.04
232	1.94	1.75	1.48	1.17	0.83	0.52	0.25	0.06
332	1.83	1.49	1.16	0.85	0.56	0.32	0.14	0.03
33	1.93	1.77	1.57	1.33	1.06	0.77	0.47	0.16
133	1.90	1.69	1.46	1.22	0.97	0.71	0.43	0.15
233	1.97	1.86	1.68	1.44	1.15	0.84	0.51	0.17
333	1.86	1.60	1.35	1.12	0.88	0.65	0.40	0.14

TMTR

1	1	26	26	51	51	76	76
6	11	31	36	56	61	81	86
16	21	41	46	66	71	91	96
1	1	26	26	51	51	76	76
6	11	31	36	56	61	81	86
16	21	41	46	66	71	91	96
6	11	31	36	56	61	81	86
16	21	41	46	66	71	91	96
1	1	26	26	51	51	76	76
6	11	31	36	56	61	81	86
16	21	41	46	66	71	91	96
27	78	52	3	77	28	2	53
32	88	57	13	82	38	7	63
37	83	62	8	87	33	12	58
42	98	67	23	92	48	17	73
47	93	72	18	97	43	22	68
27	78	52	3	77	28	2	53
32	88	57	13	82	38	7	63
37	83	62	8	87	33	12	58
42	98	67	23	92	48	17	73
47	93	72	18	97	43	22	68
27	78	52	3	77	28	2	53
32	88	57	13	82	38	7	63
37	83	62	8	87	33	12	58
42	98	67	23	92	48	17	73
47	93	72	18	97	43	22	68
27	78	52	3	77	28	2	53
32	88	57	13	82	38	7	63
37	83	62	8	87	33	12	58
47	93	72	18	97	43	22	68
32	88	57	13	82	38	7	63
42	98	67	23	92	48	17	73
54	55	79	80	4	5	29	30
59	65	84	90	9	15	34	40
64	60	89	85	14	10	39	35
69	75	94	100	19	25	44	50
54	55	79	80	4	5	29	30
59	65	84	90	9	15	34	40
64	60	89	85	14	10	39	35
69	75	94	100	19	25	44	50
54	55	79	80	4	5	29	30
59	65	84	90	9	15	34	40
64	60	89	85	14	10	39	35
69	75	94	100	19	25	44	50
54	55	79	80	4	5	29	30
59	65	84	90	9	15	34	40
64	60	89	85	14	10	39	35
69	75	94	100	19	25	44	50

XMTR

1	1	26	26	51	51	76	76
1	1	26	26	51	51	76	76
1	1	26	26	51	51	76	76
2	3	27	28	52	53	77	78
2	3	27	28	52	53	77	78
2	3	27	28	52	53	77	78
3	2	28	27	53	52	78	77
3	2	28	27	53	52	78	77
4	5	29	30	54	55	79	80
5	4	30	29	55	54	80	79
5	4	30	29	55	54	80	79
6	11	31	36	56	61	81	86
6	11	31	36	56	61	81	86
6	11	31	36	56	61	81	86
6	11	31	36	56	61	81	86
6	11	31	36	56	61	81	86
7	13	32	38	57	63	82	88
7	13	32	38	57	63	82	88
7	13	32	38	57	63	82	88
7	13	32	38	57	63	82	88
7	13	32	38	57	63	82	88
8	12	33	37	58	62	83	87
8	12	33	37	58	62	83	87
8	12	33	37	58	62	83	87
8	12	33	37	58	62	83	87
8	12	33	37	58	62	83	87
9	15	34	40	59	65	84	90
9	15	34	40	59	65	84	90
9	15	34	40	59	65	84	90
9	15	34	40	59	65	84	90
10	14	35	39	60	64	85	89
10	14	35	39	60	64	85	89
16	21	41	46	66	71	91	96
16	21	41	46	66	71	91	96
16	21	41	46	66	71	91	96
16	21	41	46	66	71	91	96
17	23	42	48	67	73	92	98
17	23	42	48	67	73	92	98
17	23	42	48	67	73	92	98
17	23	42	48	67	73	92	98
18	22	43	47	68	72	93	97
18	22	43	47	68	72	93	97
18	22	43	47	68	72	93	97
18	22	43	47	68	72	93	97
19	25	44	50	69	75	94	100
19	25	44	50	69	75	94	100
19	25	44	50	69	75	94	100
19	25	44	50	69	75	94	100

****

KBSC = 6 KBSL = 12 GSNR = 2 RSDV = 1.59
SF1 = 31 SF2 = 16 SPSO = 8 LMAX = 1442
TSTN =100 ORGN = 48 SPSY = 16 BAND = 2.0

****

OBIN	SYMBOLS	BIT ERRORS	- B.E.R. -
1	1023	164	8.016E-02
2	1022	151	7.387E-02
3	1021	146	7.150E-02
4	1020	142.	6.961E-02
5	1019	139	6.820E-02
6	1018	138	6.778E-02
7	1017	138	6.785E-02
8	1016	137	6.742E-02
9	1015	136	6.700E-02
10	1014	136	6.706E-02
11	1013	136	6.713E-02
12	1012	136	6.719E-02
13	1011	136	6.726E-02
14	1010	136	6.733E-02
15	1009	136	6.739E-02
16	1008	135	6.696E-02
17	1007	134	6.653E-02
18	1006	134	6.660E-02
19	1005	134	6.667E-02
20	1004	134	6.673E-02
21	1003	134	6.680E-02
22	1002	134	6.687E-02
23	1001	134	6.693E-02
24	1000	134	6.700E-02
25	999	134	6.707E-02
26	998	134	6.713E-02
27	997	134	6.720E-02
28	996	134	6.727E-02
29	995	134	6.734E-02
30	994	134	6.740E-02
31	993	133	6.697E-02
32	992	132	6.653E-02
33	991	132	6.660E-02
34	990	132	6.667E-02
35	989	132	6.673E-02
36	988	132	6.680E-02
37	987	132	6.687E-02
38	986	132	6.694E-02
39	985	132	6.701E-02
40	984	132	6.707E-02
41	983	132	6.714E-02
42	982	132	6.721E-02
43	981	131	6.677E-02
44	980	130	6.633E-02
45	979	130	6.639E-02
46	978	130	6.646E-02
47	977	130	6.653E-02
48	976	130	6.660E-02
49	975	130	6.667E-02
50	974	130	6.674E-02
51	973	130	6.680E-02
52	972	130	6.687E-02

Matrix ZZ

- 2- :	1	0	- 23- :	52	0	- 24- :	52	0	- 37- :	1	0
- 66- :	3	0	- 67- :	2	0	- 75- :	52	0	- 76- :	0	52
- 77- :	3	0	- 78- :	52	0	- 79- :	1	0	- 93- :	50	0
- 94- :	1	51	- 95- :	52	0	- 116- :	3	0	- 117- :	0	2
- 118- :	2	0	- 119- :	1	0	- 122- :	52	0	- 123- :	52	0
- 129- :	52	0	- 130- :	52	0	- 176- :	50	0	- 177- :	51	0
- 179- :	52	0	- 180- :	52	0	- 197- :	52	0	- 198- :	52	0
- 209- :	1	0	- 213- :	52	0	- 214- :	52	0	- 216- :	52	0
- 217- :	52	0	- 224- :	1	0	- 238- :	52	0	- 239- :	52	0
- 255- :	52	0	- 256- :	52	0	- 266- :	50	0	- 267- :	1	51
- 268- :	52	0	- 275- :	0	50	- 276- :	0	51	- 280- :	1	0
- 283- :	52	0	- 284- :	52	0	- 288- :	52	0	- 290- :	52	0
- 311- :	1	0	- 319- :	2	0	- 320- :	1	0	- 332- :	51	0
- 333- :	51	0	- 338- :	2	0	- 339- :	1	0	- 349- :	1	0
- 350- :	1	0	- 357- :	1	0	- 362- :	1	0	- 377- :	52	0
- 378- :	52	0	- 393- :	2	0	- 394- :	1	0	- 414- :	52	0
- 415- :	0	52	- 416- :	52	0	- 421- :	52	0	- 422- :	52	0
- 426- :	1	0	- 429- :	52	0	- 430- :	52	0	- 445- :	51	0
- 446- :	51	0	- 447- :	52	0	- 448- :	52	0	- 449- :	4	0
- 450- :	3	0	- 454- :	1	0	- 482- :	52	0	- 483- :	51	1
- 510- :	1	0	- 511- :	1	0	- 521- :	1	0	- 534- :	52	0
- 535- :	52	0	- 547- :	52	0	- 548- :	52	0	- 557- :	52	0
- 558- :	52	0	- 564- :	52	0	- 565- :	1	0	- 566- :	52	0
- 574- :	51	0	- 575- :	52	0	- 617- :	2	0	- 618- :	52	0
- 619- :	52	0	- 628- :	52	0	- 629- :	52	0	- 632- :	52	0
- 633- :	52	0	- 636- :	52	0	- 637- :	52	0	- 640- :	52	0
- 641- :	52	0	- 642- :	1	0	- 660- :	52	0	- 661- :	0	52
- 662- :	52	0	- 673- :	52	0	- 674- :	52	0	- 679- :	49	0
- 680- :	2	0	- 681- :	52	0	- 683- :	52	0	- 684- :	52	0
- 692- :	1	0	- 716- :	51	0	- 717- :	52	0	- 718- :	2	0
- 719- :	1	0	- 721- :	1	0	- 722- :	0	1	- 723- :	50	0
- 724- :	51	0	- 733- :	2	0	- 734- :	52	0	- 736- :	52	0
- 743- :	0	52	- 744- :	51	1	- 745- :	52	0	- 746- :	51	1
- 747- :	52	0	- 753- :	52	0	- 754- :	52	0	- 767- :	52	0
- 768- :	52	0	- 778- :	0	1	- 779- :	1	0	- 819- :	1	0
- 825- :	52	0	- 826- :	52	0	- 837- :	52	0	- 838- :	52	0
- 864- :	52	0	- 865- :	52	0	- 871- :	0	2	- 873- :	52	0
- 874- :	52	0	- 877- :	52	0	- 878- :	0	52	- 879- :	0	52
- 880- :	52	0	- 883- :	51	0	- 884- :	52	0	- 909- :	1	0
- 910- :	1	0	- 911- :	52	0	- 912- :	52	0	- 919- :	52	0
- 920- :	52	0	- 931- :	1	0	- 938- :	52	0	- 939- :	52	0
- 942- :	52	0	- 943- :	52	0	- 971- :	50	0	- 972- :	51	0
- 981- :	42	0	- 982- :	41	0	- 993- :	31	0	- 994- :	30	0
-1008- :	16	0	-1009- :	15	0	-1016- :	8	0	-1017- :	7	0
-1019- :	3	0	-1020- :	3	0						

Matrix XX, matrix TT en replica's voor L=2

XMTR

1	1	6	6	11	11	16	16
1	1	6	6	11	11	16	16
1	1	6	6	11	11	16	16
2	3	7	8	12	13	17	18
2	3	7	8	12	13	17	18
2	3	7	8	12	13	17	18
2	3	7	8	12	13	17	18
2	3	7	8	12	13	17	18
4	5	9	10	14	15	19	20
4	5	9	10	14	15	19	20
4	5	9	10	14	15	19	20
4	5	9	10	14	15	19	20

TMTR

1	1	6	6	11	11	16	16
2	3	7	8	12	13	17	18
4	5	9	10	14	15	19	20
6	16	11	1	16	6	1	11
7	18	12	3	17	8	2	13
8	17	13	2	18	7	3	12
9	20	14	5	19	10	4	15
10	19	15	4	20	9	5	14
11	11	16	16	1	1	6	6
12	13	17	18	2	3	7	8
13	12	18	17	3	2	8	7
14	15	19	20	4	5	9	10

ORIGINELEN:

0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100	-0.04	-0.09	-0.11	-0.11	-0.09	-0.06	-0.04	-0.01
300	-0.07	-0.17	-0.21	-0.21	-0.18	-0.13	-0.07	-0.02
10	0.96	0.85	0.72	0.58	0.42	0.28	0.15	0.04
110	0.92	0.76	0.61	0.47	0.33	0.22	0.11	0.03
210	0.99	0.94	0.83	0.68	0.51	0.34	0.19	0.05
310	0.89	0.68	0.51	0.36	0.24	0.15	0.08	0.02
410	1.03	1.02	0.94	0.79	0.60	0.41	0.22	0.06
30	1.91	1.70	1.44	1.15	0.85	0.56	0.30	0.09
130	1.88	1.61	1.33	1.04	0.76	0.50	0.26	0.08
230	1.95	1.79	1.55	1.26	0.94	0.62	0.34	0.10
330	1.84	1.53	1.23	0.94	0.67	0.43	0.23	0.07

Matrix XX, matrix TT en replica's voor L=1

XMTR

1	1	2	2	3	3	4	4
1	1	2	2	3	3	4	4
1	1	2	2	3	3	4	4

TMTR

1	1	6	6	11	11	16	16
2	3	7	8	12	13	17	18
4	5	9	10	14	15	19	20

ORIGINELEN:

0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
10	0.96	0.85	0.72	0.58	0.42	0.28	0.15	0.04
30	1.91	1.70	1.44	1.15	0.85	0.56	0.30	0.09

5:3 Decaratiefile; BWHEAD5.DAT

```
CONST PI=3.141592653589793;

TYPE  VRZ1=(C_ORDV,C_FPTN,C_MRGE,C_SPSO,C_LBIN,C_OBIN,C_LBRE,C_FOLT,
            C_RSRE,C_GSNR,C_RSGM,C_KBSC,C_KBSL,OGNN,FLTR);
      VRZ2=(YES,NO);

      COMPLEX= RECORD R,I:REAL END;

      DIMAB= -1..32768; DIMCS= -1..16384; DIMY= 0..4096;
      DM1=0..97{3*SPSO}; DM2=1..70{ORGN}; DM3=1..2;
      DM4=1..4 {FPTN}; DM5=1..100{TSTN}; DM6=1..1024{SYMB};
      DM8=1..4; DM10=1..60{OBIN}; DM11=1..161{5*SPSO};

      MTRAB=ARRAY[DIMAB] OF COMPLEX; MTRCS=ARRAY[DIMCS] OF COMPLEX;

      MTRA=ARRAY[DM11] OF REAL; MTRB=ARRAY[DM2,DM1] OF REAL;
      MTRC=ARRAY[DM2,DM3,DM4] OF INTEGER; MTRD=ARRAY[DM1] OF REAL;
      MTRF=ARRAY[DM10] OF INTEGER; MTRH=ARRAY[DM2,DM3,DM4] OF REAL;
      MTRL=ARRAY[DM5] OF INTEGER; MTRQ=PACKED ARRAY[1..3] OF CHAR;
      MTRR=ARRAY[0..10,1..3] OF INTEGER; MTRS=ARRAY[DM5,DM6] OF INTEGER;
      MTRY=ARRAY[DIMY] OF INTEGER;
      MTRZ=ARRAY[DM6] OF INTEGER; MTRV=ARRAY[DM6,1..2] OF INTEGER;

VAR   AB :MTRAB; UU :MTRAB; CS :MTRCS;

      AA : MTRA; BB : MTRB; CC : MTRC; DD : MTRD;
      FF : MTRF; LL : MTRL; QQ : MTRQ; RR : MTRR;
      SS : MTRS; TT : MTRC; VV : MTRZ; XX : MTRC;
      YY : MTRY; ZZ : MTRV; HH : MTRH; GEG1:VRZ1; FOLT:VRZ2;

      INCR,BPSY,SPSY,SYMB,LBIN,SMB,TSTN,FPTN,ORGN,SPSO,OBIN,GSNR,KBS,
      ORDV,TPFP,ORDE,SF1,SF2,BTSN,BTSM,LMAX,RSRE,RUIS,LBRE : INTEGER;

      BAND,CARR,SHFT,MRGE,SFF,RSGM,RSDV : REAL;  PINP,POUT:TEXT;
```

5:4 PROGRAM ACQUISITIE(PINP,POUT);

```
%INCLUDE 'BWHEAD5.PAS'

PROCEDURE MLE_ONTV;EXTERN;

PROCEDURE VTI_ONTV;EXTERN;

PROCEDURE INITIALISATIE;EXTERN;

FUNCTION  ZENDER(VAR R:ARRAY[DIMY] OF INTEGER;
                 VAR M,N:INTEGER):BOOLEAN;EXTERN;

BEGIN{0} RESET(PINP); REWRITE(POUT);

IF ZENDER(YY,BTSM,BTSN) THEN

BEGIN{1} INITIALISATIE;

IF ORDV=0 THEN MLE_ONTV ELSE VTI_ONTV

END{1} END{0}.
```

```
5:5   MODULE VTI_ONTVANGERSIMULATIE(PINP,POUT);

      %INCLUDE 'BWHEAD5.PAS'

      FUNCTION  GO5DDF(A,B:REAL):REAL;EXTERN;
      PROCEDURE GO5CBF(X:INTEGER);EXTERN;
      PROCEDURE FFTC_1(M:INTEGER; VAR A,CS:ARRAY[DIMAB] OF COMPLEX);EXTERN;
      PROCEDURE FILTMXFL(N,SPSY,BPSY,GR:INTEGER; VAR A:ARRAY[DIMAB] OF COMPLEX;
                         F3,FC,FO:REAL);EXTERN;
      PROCEDURE FFTC(M:INTEGER; VAR A,CS:ARRAY[DIMAB] OF COMPLEX);EXTERN;


      PROCEDURE INITIALISATIE;
                VAR I,J,Z,TELLER : INTEGER;
                BEGIN{0}
                FOR TELLER:=1 TO 15 DO BEGIN{1} {AANTAL ELEMENTEN IN VRZ1}
                READ(PINP,GEG1);CASE GEG1 OF

                C_MRGE : READ(PINP,QQ,MRGE); C_FPTN : READ(PINP,QQ,FPTN);
                C_ORDV : READ(PINP,QQ,ORDV); C_FOLT : READ(PINP,QQ,FOLT);
                C_LBIN : READ(PINP,QQ,LBIN); C_OBIN : READ(PINP,QQ,OBIN);
                C_LBRE : READ(PINP,QQ,LBRE); C_RSRE : READ(PINP,QQ,RSRE);
                C_RSGM : READ(PINP,QQ,RSGM); C_SPSO : READ(PINP,QQ,SPSO);
                C_KBSC : BEGIN{1} READ(PINP,QQ,RR[0,1]);
                         FOR I:=1 TO RR[0,1] DO READ(PINP,RR[I,1]) END{1};
                C_KBSL : BEGIN{2} READ(PINP,QQ,RR[0,2]);
                         FOR I:=1 TO RR[0,2] DO READ(PINP,RR[I,2]) END{2};
                C_GSNR : BEGIN{3} READ(PINP,QQ,RR[0,3]);
                         FOR I:=1 TO RR[0,3] DO READ(PINP,RR[I,3]) END{3};
                 OGNN  : BEGIN{6} FOR I:=1 TO 160 DO
                         READ(PINP,AA[I],AA[I]) END{6};

                 FLTR  : READLN(PINP,ORDE,BAND,CARR,SHFT);

                END{CASE} END{1}; INCR:=(SPSY DIV SPSO);

                END{0};
```

```
PROCEDURE REPLICAS;

        VAR  SS:ARRAY[0..4,1..32,1..5] OF REAL;
             INCR,TOT,B,C,X,Y,A,I,J,Z:  INTEGER;
             DCL:REAL;

        FUNCTION CHECK :  BOOLEAN;
             BEGIN{0} IF (X+Y=7) OR (Y+Z=7) OR (Y=2) OR (Y=4) OR
             (TOT-Z=400) OR (TOT-Z=200) THEN CHECK:=FALSE
             ELSE CASE TOT OF 103,204,303,404,14,214,313,414,2,4:
             CHECK:=FALSE OTHERWISE CHECK:=TRUE END{CASE} END{0};

        BEGIN{0}

        FOR J:=1 TO 5 DO FOR I:=1 TO 32 DO

        BEGIN{1}    DCL:=AA[(J-1)*32+I];

        SS[1,I,J]:=  DCL; SS[2,I,J]:=  -DCL; SS[0,I,J]:=0;
        SS[3,I,J]:=2*DCL; SS[4,I,J]:=-2*DCL; END{1}; I:=0;


        FOR Y:=0 TO 4 DO BEGIN{A}
        FOR C:=0 TO 4 DO
        FOR B:=0 TO 4 DO BEGIN{B}

        CASE ORDV OF

          1:BEGIN X:=0; Z:=0; C:=4; B:=4; TSTN:=4 END;
          2:BEGIN X:=B; Z:=0; C:=4; TSTN:=20 END;
          3:BEGIN X:=B; Z:=C; TSTN:=100 END END{CASE};

        TOT:=(100*X)+(10*Y)+Z ; IF CHECK THEN BEGIN{2}

        I:=I+1; BB[I,0]:=TOT; DCL:=SS[Z,1,2];

        FOR J:=1 TO 32 DO BEGIN{3}

        BB[I,J]:=SS[X,J,4]+SS[Y,J,3]+SS[Z,J,2]-DCL;

        END{3}  END{2}  END{B}  END{A};  ORGN := I;


        INCR:=32 DIV SPSO; IF INCR>1 THEN BEGIN{3}
        FOR I:=1 TO ORGN DO BEGIN{4} Y:=1+(INCR DIV 2);
        FOR X:=1 TO SPSO DO BEGIN{5}
        BB[I,X]:=BB[I,Y]; Y:=Y+INCR  END{5} END{4} END{3};

        END{0};
```

```
PROCEDURE TOEST_LOCATIES;

        VAR I,J,X,Y,Z:INTEGER;

        FUNCTION FASE(Y:INTEGER):INTEGER; BEGIN{0}
             CASE Y OF 0:Y:=0; 1:Y:=1; 2:Y:=3; 3,4:Y:=2;
             END{CASE}; FASE:=Y  END{0};

        FUNCTION MIR(Y:INTEGER):INTEGER; BEGIN{0}
             CASE Y OF 0:Y:=0; 1:Y:=2; 2:Y:=1; 3:Y:=4; 4:Y:=3;
             END{CASE}; MIR:=Y  END{0};

        FUNCTION TSTD(X,Y,Z:INTEGER):INTEGER; BEGIN{0}
             CASE ORDV OF 1:TSTD:=FASE(Z); 2:TSTD:=5*FASE(Z)+X;
             3:TSTD:=25*FASE(Z)+5*X+Y END{CASE} END{0};

        BEGIN{0} TPFP:=TSTN DIV FPTN; {aantal toestanden per fase}

        FOR I:=1 TO ORGN DO BEGIN{6} X:=ROUND(BB[I,0]);

        Z:=X MOD 10; X:=X DIV 10; Y:=X MOD 10; X:=X DIV 10;

        TT[I,1,1]:=TSTD(X,Y,Y); TT[I,2,1]:=TSTD(MIR(X),MIR(Y),MIR(Y));
        XX[I,1,1]:=TSTD(Y,Z,0); XX[I,2,1]:=TSTD(MIR(Y),MIR(Z),0);

        FOR Z:=4 DOWNTO 1 DO BEGIN{7}

        TT[I,1,Z]:=((TT[I,1,1]+((Z-1)*TPFP)) MOD (4*TPFP))+1 ;
        XX[I,1,Z]:=((XX[I,1,1]+((Z-1)*TPFP)) MOD (4*TPFP))+1 ;
        TT[I,2,Z]:=((TT[I,2,1]+((Z-1)*TPFP)) MOD (4*TPFP))+1 ;
        XX[I,2,Z]:=((XX[I,2,1]+((Z-1)*TPFP)) MOD (4*TPFP))+1

        END{7} END{6};

        IF ORDV=1 THEN BEGIN{1} X:=0; FOR I:=1 TO 3 DO FOR J:=1 TO 2 DO
        BEGIN{2} TT[I,J,1]:=X; X:=X+1; TT[1,1,1]:=1; FOR Z:=4 DOWNTO 2 DO
        TT[I,J,Z]:=(Z-1)*5+TT[I,J,1]; TPFP:=5 END{2} END{1};

        END{0};
```

```
PROCEDURE GEGEVENS;
          VAR I,J,Z:INTEGER;
          BEGIN{0} PAGE(POUT);
          WRITELN(POUT);WRITELN(POUT);WRITELN(POUT);

          WRITE(POUT,'INITIALISATIE VAN DE ONTVANGER:');
          WRITELN(POUT);WRITELN(POUT);

          WRITELN(POUT,'C_LBRE',QQ,LBRE:5,'      C_RSRE',QQ,RSRE:5);
          WRITELN(POUT,'C_RSGM',QQ,RSGM:7:2,'    C_FPTN',QQ,FPTN:5);
          WRITELN(POUT,'C_MRGE',QQ,MRGE:7:2,'    C_SPSO',QQ,SPSO:5);
          WRITELN(POUT,'C_LBIN',QQ,LBIN:5,'      C_OBIN',QQ,OBIN:5);
          WRITELN(POUT,'  TSTN',QQ,TSTN:5,'        ORGN',QQ,ORGN:5);

          WRITE(POUT,'C_KBSC :=');FOR I:=1 TO RR[0,1] DO
          WRITE(POUT,RR[I,1]:4);       WRITELN(POUT);
          WRITE(POUT,'C_KBSL :=');FOR I:=1 TO RR[0,2] DO
          WRITE(POUT,RR[I,2]:4);       WRITELN(POUT);
          WRITE(POUT,'C_GSNR :=');FOR I:=1 TO RR[0,3] DO
          WRITE(POUT,RR[I,3]:4);       WRITELN(POUT);

          WRITELN(POUT,'FILTER:  ORDE=',ORDE:3,' CARR=',CARR:4:1,
          ' SHFT=',SHFT:4:1,' BAND=',BAND:4:1,'  Verh tov Fbit');
          WRITELN(POUT);WRITELN(POUT);

          IF FOLT=YES THEN BEGIN{1}
          WRITELN(POUT,'TMTR',
          '                                          XMTR');
          WRITELN(POUT);
          FOR I:=1 TO ORGN DO BEGIN{3} WRITELN(POUT);
          FOR Z:=1 TO FPTN DO
          WRITE(POUT,TT[I,1,Z]:5,TT[I,2,Z]:5);
          WRITE(POUT,'     ');
          FOR Z:=1 TO FPTN DO
          WRITE(POUT,XX[I,1,Z]:5,XX[I,2,Z]:5)
          END{3}; WRITELN(POUT);WRITELN(POUT);

          WRITELN(POUT,'ORIGINELEN:'); WRITELN(POUT);
          FOR I:=1 TO ORGN DO BEGIN{4} WRITELN(POUT);
          WRITE(POUT,ROUND(BB[I,0]):3,'    ');
          FOR Z:=1 TO SPSO DO BEGIN{5}
          WRITE(POUT,BB[I,Z]:7:2); IF SPSO>8 THEN
          Z:=Z-1+(SPSO DIV 8) END{5} END{4} END{1}; PAGE(POUT);

          END{0};
```

```
PROCEDURE RESULTS;

        VAR  I,X,Z,AANTAL,ERROR: INTEGER;  BER:REAL;

        BEGIN{0}
        WRITELN(POUT);
        WRITELN(POUT,'    ************************************************************');
        WRITELN(POUT,'     KBSC =',RR[KBS,1]:3,'    KBSL =',
        RR[KBS,2]:3,'  GSNR =',RR[GSNR,3]:3,'   RSDV =',RSDV:6:2);
        WRITELN(POUT,'      SF1 =',SF1:3       ,'     SF2 =',
        SF2:3       ,'  SPSO =', SPSO:3      ,'    LMAX =   ',LMAX:3 );
        WRITELN(POUT,'     TSTN =',TSTN:3      ,'    ORGN =',
        ORGN:3      ,'  SPSY =', SPSY:3      ,'    BAND =   ',  BAND:4:1 );
        WRITELN(POUT,'    ************************************************************');

        WRITELN(POUT);WRITELN(POUT,'     OBIN         SYMBOLS     ',
                                   '   BIT ERRORS     - B.E.R.-   ');
        WRITELN(POUT);

        FOR I:=LBIN TO OBIN DO BEGIN{1}
        AANTAL:=(RSRE-LBRE+1)*(SYMB-1); ERROR :=FF[I];
        IF ERROR=0 THEN BER:=0 ELSE BER:=ERROR/(BPSY*AANTAL);

        WRITELN(POUT,'     ',I:3,'          ',AANTAL:5,
                '           ',ERROR:4,'         ',BER:10) END{1};


        IF (FOLT = YES) THEN BEGIN{1}

        WRITELN(POUT);WRITELN(POUT); X:=0;
        FOR Z:=1 TO SYMB DO
        IF ((ZZ[Z,1]<>0) OR (ZZ[Z,2]<>0)) THEN BEGIN{2}
        IF (X MOD 4 = 0) THEN WRITELN(POUT);
        WRITE(POUT,'-',Z:4,'- :  ',ZZ[Z,1]:2,' ',ZZ[Z,2]:2,'    ');
        X:=X+1  END{2}; WRITELN(POUT);WRITELN(POUT) END{1};

        END{0};
```

```
PROCEDURE  RUIS_FILTER;

           VAR Z:INTEGER; X,Y:REAL; BEGIN{0} G05CBF(RUIS);

           FOR Z:=0 TO (BTSN-1) DO BEGIN{1}
           IF (RR[GSNR,3] = 100) THEN BEGIN{2} X:=0; Y:=0 END{2}
           ELSE BEGIN{3} X:=G05DDF(RSGM,RSDV);
                         Y:=G05DDF(RSGM,RSDV) END{3};

           UU[Z].R:=AB[Z].R+X ; UU[Z].I:=AB[Z].I+Y END{1};

           FFTC_1(BTSM,UU,CS); {NAAR FREKW DOMEIN}
           FILTMXFL(BTSN,SPSY,BPSY,ORDE,UU,BAND,CARR,SHFT);
           FFTC(BTSM,UU,CS); {NAAR TIJD DOMEIN}

           END{0};

FUNCTION   ADC(RX:REAL):INTEGER;
           VAR Y:INTEGER; BEGIN{0}
           Y:=ROUND(RX*SFF);
           IF Y> SF1 THEN Y:= SF1;
           IF Y<-SF1 THEN Y:=-SF1; ADC:=Y;
           END{0};

PROCEDURE CORRELATOR;
           VAR RA,RB,RC,RD : REAL;
               Z,I,J,A,B,C,D,SAC,SBD,SBC,SAD : INTEGER;
           BEGIN{0} Z:=(SMB-1)*SPSY - (INCR DIV 2);
           FOR J:=1 TO SPSO DO BEGIN{1} Z:=Z+INCR;
           DD[J]:=(UU[Z].I); AA[J];=(UU[Z].R) END{1};

           FOR I:=1 TO ORGN DO BEGIN{2}
           SAC:=0; SBD:=0; SBC:=0; SAD:=0;

           FOR J:=1 TO SPSO DO BEGIN{3}
           RA:=AA[J]; RB:=DD[J];
           RC:=COS(0.5*PI*BB[I,J]);
           RD:=SIN(0.5*PI*BB[I,J]);

           A:=ADC(RA); B:=ADC(RB);
           C:=ADC(RC); D:=ADC(RD);

           SAC:=SAC+A*C; SBD:=SBD+B*D;
           SBC:=SBC+B*C; SAD:=SAD+A*D  END{3};

           SAC:=(SAC DIV SF2); SBD:=(SBD DIV SF2);
           SBC:=(SBC DIV SF2); SAD:=(SAD DIV SF2);

           CC[I,1,1]:=SAC+SBD;   CC[I,2,1]:=SAC-SBD;
           CC[I,1,2]:=SBC-SAD;   CC[I,2,2]:=SBC+SAD;

           END{2} END{0};
```

```
FUNCTION  STATE(T: INTEGER): INTEGER;

          VAR I: INTEGER; BEGIN{0} CASE T OF

          1,8,17,14,15 :  I:=1;
          2,6,19,20,13 :  I:=2;
          3,9,10,16,12 :  I:=4;
          4,5,7 ,18,11 :  I:=3    END{CASE};

          STATE:=I    END{0};


PROCEDURE HISTORIE;

          VAR I,J,Z,T: INTEGER; BEGIN{0}
          FOR Z:=1 TO 4     DO BEGIN{1}
          FOR J:=1 TO 2     DO BEGIN{2}
          FOR I:=1 TO ORGN DO BEGIN{3}

          IF Z<3 THEN CC[I,J,Z+2]:=-CC[I,J,Z];

          T:=TT[I,J,Z];

          IF TSTN=4 THEN T:=STATE(T);

          CC[I,J,Z]:=CC[I,J,Z]+LL[T]

          END{3} END{2} END{1} END{0};


PROCEDURE SURVIVORS;

          VAR I,P,Q,R,X,TST: INTEGER;  BEGIN{0}

          FOR I:=1 TO TSTN DO LL[I]:=-32;

          FOR P:=1 TO ORGN DO      FOR Q:=1 TO 2 DO
          FOR R:=1 TO 4 DO BEGIN{1} TST:=XX[P,Q,R];

          IF CC[P,Q,R]>LL[TST] THEN BEGIN{2}

          LL[TST]:=CC[P,Q,R];  SS[TST,SMB]:=TT[P,Q,R]

          END{2} END{1};

          X:=1;  FOR I:=2 TO TSTN DO
          IF LL[I]>LL[X] THEN X:=I;   VV[SMB]:=X;
          IF LL[X]>LMAX THEN
          FOR I:=1 TO TSTN DO LL[I]:=LL[I]-LMAX;

          END{0};
```

```
PROCEDURE SYMBOL(OBIN: INTEGER);

          VAR SBL,LB,X,Y,J,T,ERROR: INTEGER;

          BEGIN{0} ERROR:=0;
          FOR SBL:=1 TO SYMB DO BEGIN{1}
          X:=VV[SBL]; Y:=X;
          LB:=(SBL-OBIN);

          IF LB<1 THEN X:=0 ELSE BEGIN{2}
          FOR J:=SBL DOWNTO (LB+1) DO

          IF TSTN>4 THEN X:=SS[X,J] ELSE BEGIN{3}
          X:=SS[Y,J]; Y:=STATE(X); X:=SS[Y,J-1]
          END{3} END{2};


          X:=(X-1) MOD TPFP; IF ORDV=3 THEN X:=X DIV 5;

          CASE X OF
                       0 : Y:=  0;
                       1 : Y:= -2;
                       2 : Y:=  2;
                       3 : Y:= -4;
                       4 : Y:=  4    END{CASE};


          IF LB>0 THEN BEGIN{4}    X:=ABS(YY[LB-1]-Y);

          CASE X OF

          2,6 : BEGIN ERROR:=ERROR+1; ZZ[LB,1]:=ZZ[LB,1]+1 END;
            4 : BEGIN ERROR:=ERROR+2; ZZ[LB,2]:=ZZ[LB,2]+1 END;

          OTHERWISE   ERROR:=ERROR  END{CASE}

          END{4} END{1};

          FF[OBIN]:=FF[OBIN]+ERROR

          END{0};
```

```
PROCEDURE START_WAARDEN;

          VAR Z: INTEGER;

          BEGIN{0}

          SF1:=((2**(RR[KBS,1]-1))-1); SFF:=SF1/MRGE;
          SF2:=(4*SPS0*((SF1+1)**2))DIV(2**(RR[KBS,2]-1));
          LMAX:=ROUND(3*(SPS0*(SF1**2))/SF2);
          FOR Z:=LBIN TO OBIN DO FF[Z]:=0;
          FOR Z:=1    TO TSTN DO LL[Z]:=0;
          FOR Z:=1    TO SYMB DO BEGIN
          ZZ[Z,1]:=0; ZZ[Z,2]:=0 END

          END{0};


PROCEDURE VTI_ONTV;  {viterbi ontvanger}

          VAR Z: INTEGER;

          BEGIN{0}

          REPLICAS; TOEST_LOCATIES; GEGEVENS;

          FOR GSNR:=1 TO RR[0,3] DO BEGIN{1}

          RSDV:=SQRT(SPSY/(2*BPSY*10**(RR[GSNR,3]/10)));

          FOR KBS:=1 TO RR[0,1] DO BEGIN{2}

          START_WAARDEN;

          FOR RUIS:=LBRE TO RSRE DO BEGIN{3}

          RUIS_FILTER;

          FOR SMB:=1 TO SYMB DO BEGIN{4}

          CORRELATOR; HISTORIE; SURVIVORS  END{4};

          FOR Z:=LBIN TO OBIN DO SYMBOL(Z) END{3};

          RESULTS END{2} END{1}; END{0};

          END.
```

5:6 MODULE MLE_ONTVANGERSIMULATIE (PINP,POUT);

```
%INCLUDE 'BWHEAD5.PAS'  {algemeen declaratieblok}

PROCEDURE RUIS_FILTER;  EXTERN;
PROCEDURE START_WAARDEN;EXTERN;


PROCEDURE REPL_MLE; {genereren van 48 originelen}

          VAR  SS:ARRAY[0..4,1..32,1..5] OF REAL;
               TOT,B,C,X,Y,A,I,J,Z,INCR : INTEGER;
               DCL:REAL;

          FUNCTION CHECK : BOOLEAN; BEGIN{0}
          TOT:=(100*X)+(10*Y)+Z ;
          IF (X+Y=7) OR (Y+Z=7) OR (Y=2) OR (Y=4)
          OR (TOT-Z=400) OR (TOT-Z=200)
          THEN CHECK:=FALSE
          ELSE CASE TOT OF 103,204,303,404,14,214,313,414,2,4:
          CHECK:=FALSE
          OTHERWISE CHECK:=TRUE END{CASE} END{0};

          BEGIN {0}

          FOR J:=1 TO 5  DO
          FOR I:=1 TO 32 DO

          BEGIN{1}      DCL:= AA[(J-1)*32+I];

          SS[0,I,J]:=0;
          SS[1,I,J]:=  DCL; SS[2,I,J]:=  -DCL;
          SS[3,I,J]:=2*DCL; SS[4,I,J]:=-2*DCL;

          END{1}; I:=0;

          FOR Y:=0 TO 4 DO
          FOR X:=0 TO 4 DO
          FOR Z:=0 TO 4 DO

          IF CHECK THEN BEGIN{2}

          I:=I+1; DCL:=SS[Z,1,3]; BB[I,0]:=TOT;

          FOR J:=1 TO 32 DO BEGIN{3}
          BB[I,J]   :=SS[X,J,3]+SS[Y,J,2]+SS[Z,J,1]-DCL;
          BB[I,32+J]:=SS[X,J,4]+SS[Y,J,3]+SS[Z,J,2]-DCL;
          BB[I,64+J]:=SS[X,J,5]+SS[Y,J,4]+SS[Z,J,3]-DCL;
          END{3} END{2};

          INCR:= 32 DIV SPSO; IF (INCR>1)  THEN   BEGIN{4}
          FOR A:=1 TO 48 DO BEGIN{5} I:=1;
          FOR Z:=1 TO (3*SPSO) DO BEGIN{6}
          BB[A,Z]:=BB[A,I]; I:=I+INCR END{6} END{5} END{4}

          END{0};
```

```
PROCEDURE GEGEVENS_MLE;  {output van initiele variabelen}

        VAR I,J,Z:INTEGER;
        BEGIN{0}
        WRITELN(POUT);WRITELN(POUT);WRITELN(POUT);

        WRITE(POUT,'INITIALISATIE VAN DE ONTVANGER:');
        WRITELN(POUT);WRITELN(POUT);

        WRITELN(POUT,'C_LBRE',QQ,LBRE:5,'      C_RSRE',QQ,RSRE:5);
        WRITELN(POUT,'C_RSGM',QQ,RSGM:7:2,'    C_FPTN',QQ,FPTN:5);
        WRITELN(POUT,'C_MRGE',QQ,MRGE:7:2,'    C_SPSO',QQ,SPSO:5);

        WRITE(POUT,'C_GSNR :=');  FOR I:=1 TO RR[0,3] DO
        WRITE(POUT,RR[I,3]:4);  WRITELN(POUT);

        WRITELN(POUT,'FILTER:  ORDE=',ORDE:3,' CARR=',CARR:4:1,
        ' SHFT=',SHFT:4:1,' BAND=',BAND:4:1,'  Verh tov Fbit');
        WRITELN(POUT);WRITELN(POUT);

        PAGE(POUT);WRITELN(POUT,'ORIGINELEN:');  WRITELN(POUT);
        FOR I:=1 TO 48 DO BEGIN{4} WRITELN(POUT);
        WRITE(POUT,ROUND(BB[I,0]):3,'   ');
        FOR Z:=1 TO 3*SPSO DO BEGIN{5}
        WRITE(POUT,BB[I,Z]:7:2);
        IF SPSO>4 THEN Z:=Z-1+(SPSO DIV 4) END{5} END{4};
        PAGE(POUT);
        END{0};


PROCEDURE RESULTS_MLE;  {output van berekende variabelen}

        VAR I,X,Z,AANTAL,ERROR:INTEGER;  BER:REAL;
        BEGIN{0} AANTAL:=(RSRE-LBRE+1)*(SYMB-2);  ERROR :=FF[1];
        IF ERROR=0 THEN BER:=0 ELSE BER:=ERROR/(BPSY*AANTAL);
        WRITELN(POUT);
        WRITELN(POUT,' ******************************************************');
        WRITELN(POUT,'  SPSY =',SPSY:3,'     GSNR =',RR[GSNR,3]:3,
        '          SYMBOLS =',AANTAL:6 );
        WRITELN(POUT,'  SPSO =',SPSO:3,'     RSDV =   ',RSDV:4:2,
        '    BIT ERRORS =',ERROR:6  );
        WRITELN(POUT,'               '    ,'     BAND =   ',BAND:4:2,
        '            BER =',BER:8    );
        WRITELN(POUT,' ******************************************************');


        IF (FOLT = YES) THEN BEGIN{1}
        WRITELN(POUT);WRITELN(POUT);  X:=0;
        FOR Z:=1 TO SYMB DO
        IF ((ZZ[Z,1]<>0) OR (ZZ[Z,2]<>0)) THEN BEGIN{2}
        IF (X MOD 4 = 0) THEN WRITELN(POUT);
        WRITE(POUT,'-',Z:4,'- :  ',ZZ[Z,1]:2,' ',ZZ[Z,2]:2,'    ');
        X:=X+1   END{2};  WRITELN(POUT);WRITELN(POUT)  END{1};

        END{0};
```

```
PROCEDURE CORRELATOR_MLE;

     {correlatie van 48 originelen met een signaalrealisatie}

          VAR A,B,C,D,SAC,SBD,SBC,SAD :REAL;
              Z,I,J :  INTEGER;

          BEGIN{0} Z:=(SMB-2)*SPSY ;
          FOR J:=1 TO (3*SPSO) DO BEGIN{1}
          DD[J]:=(UU[Z].I); AA[J]:=(UU[Z].R);
          Z:=Z+INCR;   END{1};

          FOR I:=1 TO 48 DO BEGIN{2}
          SAC:=0; SBD:=0; SBC:=0; SAD:=0;

          FOR J:=1 TO (3*SPSO) DO BEGIN{3}
          A:=AA[J]; B:=DD[J];
          C:=COS(0.5*PI*BB[I,J]);
          D:=SIN(0.5*PI*BB[I,J]);

          SAC:=SAC+A*C; SBD:=SBD+B*D;
          SBC:=SBC+B*C; SAD:=SAD+A*D  END{3};

          HH[I,1,1]:=SAC+SBD;   HH[I,2,1]:=SAC-SBD;
          HH[I,1,2]:=SBC-SAD;   HH[I,2,2]:=SBC+SAD;

          END{2} END{0};



FUNCTION  SOMMATOR(X,Y,UX,UY: INTEGER'):REAL;

   {sommeren van alle correlaties van overeenkomende symbolen}

          VAR Z,I,J,P,Q :  INTEGER; A,B,C,NO : REAL;

          BEGIN{0} C:=0;

          FOR Z:=1 TO  2 DO
          FOR I:=X TO UX DO
          FOR J:=Y TO UY DO

          BEGIN{1}  A:=(HH[I,J,Z])/(RSDV**2);
          B:=EXP(A)+EXP(-1*A);    C:=C+B      END{1};

          SOMMATOR:=C

          END{0};
```

```
PROCEDURE SYMBOL_MLE;

    {symbolbeslissing en error-registratie}

        VAR I,X,Y :  INTEGER; EE:ARRAY[0..4] OF REAL;


        BEGIN{0}

        EE[0]:=SOMMATOR(1,1,11,2)-SOMMATOR(1,1,1,1);
        EE[1]:=SOMMATOR(12,1,32,1);
        EE[2]:=SOMMATOR(12,2,32,2);
        EE[3]:=SOMMATOR(33,1,48,1);
        EE[4]:=SOMMATOR(33,2,48,2);

        X:=0; FOR I:=1 TO 4 DO IF EE[I]>EE[X] THEN X:=I;

        CASE X OF
                    0 : Y:=  0;
                    1 : Y:= -2;
                    2 : Y:=  2;
                    3 : Y:= -4;
                    4 : Y:=  4  END{CASE};

        X:=ABS(YY[SMB-1]-Y);

        CASE X OF

        2,6 : BEGIN FF[1]:=FF[1]+1; ZZ[SMB,1]:=ZZ[SMB,1]+1 END;
          4 : BEGIN FF[1]:=FF[1]+2; ZZ[SMB,2]:=ZZ[SMB,2]+1 END;

        OTHERWISE   FF[1]:=FF[1]  END{CASE}

        END{0};
```

```
PROCEDURE MLE_ONTV;
    {hoofdprocedure, aanroepen van procedures in gewenste volgorde}
        VAR Z: INTEGER;
        BEGIN{0}
        REPL_MLE; GEGEVENS_MLE;
        FOR GSNR:=1 TO RR[0,3] DO BEGIN{1}
        RSDV:=SQRT(SPSY/(2*BPSY*10**(RR[GSNR,3]/10)));
        START_WAARDEN;
        FOR RUIS:=LBRE TO RSRE DO BEGIN{2}
        RUIS_FILTER;
        FOR SMB:=2 TO (SYMB-1) DO BEGIN{4}
        CORRELATOR_MLE; SYMBOL_MLE END{4} END{2};
        RESULTS_MLE END{1}; END{0};
        END.
```

```
5:7 MODULE ZENDERSIMULATIE(PINP,POUT);

  %INCLUDE 'BWHEAD5.PAS'

   PROCEDURE TABU;
             BEGIN WRITE(POUT,'                              ') END;
   PROCEDURE PLUS(VAR A,B,C: COMPLEX);
             BEGIN A.R:=B.R+C.R; A.I:=B.I+C.I END (* PROCEDURE PLUS*);
   PROCEDURE MIN( VAR A,B,C: COMPLEX);
             BEGIN A.R:=B.R-C.R; A.I:=B.I-C.I END (*PROCEDURE MIN*);
   PROCEDURE PLUSCO(VAR A,B,C: COMPLEX);
             BEGIN A.R:=B.R+C.R; A.I:=B.I-C.I END (* PROCEDURE PLUSCO*);
   PROCEDURE MINCO( VAR A,B,C: COMPLEX);
             BEGIN A.R:=B.R-C.R; A.I:=B.I+C.I END (*PROCEDURE MINCO*);
   PROCEDURE MAAL( VAR A,B,C: COMPLEX);
             VAR HO: REAL;
             BEGIN HO:=B.R*C.R-B.I*C.I; A.I:=B.I*C.R+B.R*C.I;A.R:=HO
             END (*PROCEDURE MAAL*);
   PROCEDURE MAALCO( VAR A,B,C: COMPLEX);
             VAR HO: REAL;
             BEGIN HO:=B.R*C.R+B.I*C.I; A.I:=B.I*C.R-B.R*C.I;A.R:=HO
             END (*PROCEDURE MAAL*);
   PROCEDURE DEEL (VAR A,B,C: COMPLEX);
             VAR HR,HI,HO: REAL;
             BEGIN HR:=SQR(C.R)+SQR(C.I); HI:=C.I/HR; HR:=C.R/HR;
             HO:=B.R*HR+B.I*HI; A.I:=B.I*HR-B.R*HI;A.R:=HO
             END (*PROCEDURE DEEL*);
   PROCEDURE DEELCO (VAR A,B,C: COMPLEX);
             VAR HR,HI,HO: REAL;
             BEGIN HR:=SQR(C.R)+SQR(C.I); HI:=C.I/HR; HR:=C.R/HR;
             HO:=B.R*HR-B.I*HI; A.I:=B.I*HR+B.R*HI;A.R:=HO
             END (*PROCEDURE DEELCO*);
   PROCEDURE MLC(VAR A:COMPLEX;C: REAL);
             BEGIN A.R:=A.R*C; A.I:=A.I*C
             END (*PROCEDURE MLC*);
   PROCEDURE MLPLR(VAR A,B:COMPLEX);
             VAR H: REAL;
             BEGIN H:=A.R*B.R-A.I*B.I; A.I:=A.R*B.I+A.I*B.R; A.R:=H
             END (*PROCEDURE MLPLR*);
   PROCEDURE EQU(R,I: REAL; VAR A : COMPLEX);
             BEGIN A.R:=R; A.I:=I
             END (*PROCEDURE EQU*);
   PROCEDURE DLC(VAR A: COMPLEX; C:REAL);
             BEGIN A.R:=A.R/C; A.I:=A.I/C
             END (*PROCEDURE DLC*);
   PROCEDURE PHASESH(VAR A: COMPLEX; PHI: REAL);
             VAR X,Y,H: REAL;
             BEGIN X:=COS(PHI); Y:=SIN(PHI); H:=A.R*X-A.I*Y;
             A.I:=A.R*Y+A.I*X; A.R:=H
             END (*PROCEDURE PHASESH*);
   FUNCTION  ABSCM(VAR A: COMPLEX): REAL;
             BEGIN ABSCM:=SQRT(SQR(A.R)+SQR(A.I))
             END (*FUNCTION ABSCM*);
   PROCEDURE CONJ(I,J: INTEGER; VAR A:ARRAY[DIMAB] OF COMPLEX);
             VAR K: INTEGER;
             BEGIN FOR K:= I TO J DO A[K].I:=-A[K].I
             END (*PROCEDURE CONJ*);
```

```
PROCEDURE VERME(N: INTEGER; VAR A,B: ARRAY[DIMAB] OF COMPLEX);
          VAR I: INTEGER; HO: REAL;
          BEGIN FOR I:=0 TO N DO
          BEGIN HO:=A[I].R*B[I].R-A[I].I*B[I].I;
          A[I].I:=A[I].R*B[I].I+A[I].I*B[I].R; A[I].R:=HO
          END END (*PROCEDURE VERME*);
PROCEDURE VARA1(N: INTEGER; VAR A: ARRAY[DIMAB] OF COMPLEX; X: REAL);
          VAR CI: INTEGER;
          BEGIN FOR CI:=0 TO N DO WITH A[CI] DO
          BEGIN R:=R*X; I:=I*X
          END END (*PROCEDURE VARA1*);
FUNCTION  DB(X: REAL): REAL;
          BEGIN DB:=4.342944819*LN(X) END (*FUNCTION DB*);
PROCEDURE NYQIII(M,MSYMB: INTEGER; ALF: REAL;
          VAR A: ARRAY[DIMAB] OF COMPLEX);
          VAR X,Y,Z,H: REAL; I,F,Q,N: INTEGER; NUL: COMPLEX;
          BEGIN N:=2**(M-1); Q:=2**MSYMB; H:=PI/ALF;
          F:=ROUND(Q*(1-ALF)/2+0.5); Y:=PI/Q; Z:=Y/2;
          FOR I:=F-1 DOWNTO 1 DO
          BEGIN X:=I*Y; MLC(A[I],X/SIN(X))
          END; I:=F; F:=ROUND(Q*(1+ALF)/2-0.5);
          FOR I:=I TO F DO MLC(A[I],Z*(1-SIN(H*(I/Q-0.5)))*I/SIN(Y*I));
          EQU(0,0,NUL); FOR I:=F+1 TO N DO A[I]:=NUL
          END (*PROCEDURE NYQIII*);
PROCEDURE VULCS(M: INTEGER; VAR CS: ARRAY[DIMAB] OF COMPLEX);
          LABEL 100,200;
          CONST PI=3.141592653589793;
          VAR I,J,N,ND8,Q,ND4,ND2 : INTEGER; T,HO: REAL; TWEE: BOOLEAN;
          BEGIN TWEE:=FALSE; IF ROUND(CS[-1].R)=M THEN GOTO 100;
          IF ROUND(CS[-1].R)=-M THEN
          BEGIN IF M>0 THEN GOTO 100 ELSE CS[-1].R:=M
          END ELSE BEGIN CS[-1].R:=M; TWEE:=M<0; M:=ABS(M)
          END;
          200: N:=2**ABS(M); ND2:=N DIV 2; ND4:=N DIV 4;
          ND8:=N DIV 8; T:=2*PI/N;
          IF M>0 THEN BEGIN EQU(1,0,CS[0]); J:=1; Q:=ND8 END ELSE
          BEGIN J:= ND2; Q:=ND2+ ND8-1 END;
          FOR I:= J TO Q DO
          BEGIN IF M>0 THEN HO:=I*T ELSE HO:=(I-J+0.5)*T;
          EQU(COS(HO),SIN(HO),CS[I])
          END; IF M>0 THEN BEGIN I:=1; Q:= ND8-1; J:=ND4 END ELSE
          BEGIN I:=ND2; Q:=ND2+ND8-1; J:=ND2+ND4 END;
          FOR I:= I TO Q DO
          BEGIN J:=J-1; EQU(CS[I].I,CS[I].R,CS[J])
          END; IF M>0 THEN BEGIN I:=0; Q:= ND4-1; J:=Q END ELSE BEGIN
          I:=ND2; Q:=ND2+ND4-1; J:=Q END; FOR I:=I TO Q DO
          BEGIN J:=J+1; EQU(-CS[I].I,CS[I].R,CS[J])
          END; IF TWEE THEN BEGIN TWEE:=FALSE; M:=-M; GOTO 200
          END; 100:
          END (*PROCEDURE VULCS*);
```

```
PROCEDURE XUITP(M: INTEGER; VAR A,CS: ARRAY[DIMAB] OF COMPLEX);
          (* NB LEVERT 2*X*N*)
          VAR I,K,N,Q,ND2,L,R: INTEGER;
              G,S,SC,H: COMPLEX;
              WISSEL: BOOLEAN;
          BEGIN N:=2**(M-1);ND2:=N DIV 2; K:=ND2; L:=1; VULCS(-M+1,CS);
          WISSEL:=TRUE; Q:=ND2-1; FOR I:=1 TO Q DO
             BEGIN R:=N-I; PLUSCO(G,A[R],A[I]); MINCO(S,A[R],A[I]);
             IF WISSEL THEN BEGIN SC:=CS[K]; K:=K+1 END ELSE BEGIN
             SC:=CS[L]; L:=L+1 END; WISSEL:=NOT WISSEL;
             MAAL(H,SC,S); EQU(G.R+H.I,H.R-G.I,A[I]);
             EQU(G.R-H.I,H.R+G.I,A[R])
             END; PLUSCO(G,A[0],A[N]); MINCO(S,A[0],A[N]);
          EQU(G.R+S.I,G.I-S.R,A[0]);
          EQU(G.R-S.I,-G.I-S.R,A[N]);
          A[ND2].I:=-A[ND2].I*2; A[ND2].R:=A[ND2].R*2
          END (*PROCEDURE XUITP*);
PROCEDURE FFTC(M: INTEGER; VAR A,CS: ARRAY[DIMAB] OF COMPLEX);
          VAR N,NQ,NQ2,J,K,N2,JMX : INTEGER;
              H0,T : REAL;
              H,Z : COMPLEX;
              B0 : BOOLEAN;
              BS : ARRAY[1..20] OF INTEGER;
              B0A: ARRAY [1..20] OF BOOLEAN;
          BEGIN VULCS(M,CS);
          JMX:=2**M; N:=JMX DIV 2; NQ2:=N; N2:=1;
          WHILE NQ2>0 DO
                BEGIN NQ:=1; K:=N2; J:=0;
                WHILE J<JMX DO
                   BEGIN H:=A[J]; PLUS(A[J],H,A[J+NQ2]); J:=J+NQ2;
                   MIN(A[J],H,A[J]); J:=J+NQ2
                   END; WHILE K<N DO
                   BEGIN Z:=CS[K]; K:=K+N2; J:=NQ; NQ:=NQ+1;
                   WHILE J<JMX DO
                      BEGIN H:=A[J]; PLUS(A[J],H,A[J+NQ2]); J:=J+NQ2;
                      MIN(H,H,A[J]); MAAL(A[J],H,Z); J:=J+NQ2
                      END
                   END;NQ2:=NQ2 DIV 2; N2:=N2*2
                END; FOR J:=1 TO M DO B0A[J]:=FALSE;
                BS[M]:=1; FOR J:=M-1 DOWNTO 1 DO BS[J]:=2*BS[J+1];
                N:=JMX-1; FOR K:=1 TO N DO
                BEGIN B0:=TRUE; J:=0;
                WHILE B0 DO
                   BEGIN J:=J+1; B0:=B0 AND B0A[J]; B0A[J]:= NOT B0A[J];
                   END; NQ:=0; FOR J:=1 TO M DO
                   IF B0A[J] THEN NQ:=NQ+BS[J];
                   IF NQ> K THEN
                   BEGIN H:=A[K]; A[K]:=A[NQ]; A[NQ]:=H
          END END END (*PROCEDURE FFTC*);
FUNCTION  CONTROL(MSYMB: INTEGER; VAR K: INTEGER;
          VAR R: ARRAY[DIMY] OF INTEGER): BOOLEAN;
          VAR I,N: INTEGER;
          BEGIN N:=2**MSYMB-1;K:=R[0]; FOR I:=1 TO N DO K:=K+R[I];
          CONTROL:=K=0
          END (*FUNCTION CONTROL*);
```

```
PROCEDURE XPT(M: INTEGER; VAR A,CS: ARRAY[DIMAB] OF COMPLEX);
          VAR N,P: INTEGER;
          BEGIN P:=M-1;
          N:=2**P; CONJ(0,N,A); XUITP(M,A,CS); CONJ(0,N,A); FFTC(P,A,CS)
          END (*PROCEDURE XPT*);
PROCEDURE IQR(M: INTEGER;VAR A,CS: ARRAY[DIMAB] OF COMPLEX);
          VAR I,N,K: INTEGER;
              Y,X: REAL;
          BEGIN N:=2**ABS(M); K:=N-1;X:=PI/2;
          FOR I:=N DIV 2 -1 DOWNTO 0 DO .
             BEGIN Y:=X*A[I].I; A[K].I:=-SIN(Y); A[K].R:=COS(Y); K:=K-1;
             Y:=X*A[I].R; A[K].I:=-SIN(Y); A[K].R:=COS(Y); K:=K-1
             END; FFTC(M,A,CS); CONJ(0,N-1,A); VARA1(N-1,A,1/N)
          END (*PROCEDURE IQR*);
PROCEDURE FILTMXFL(N,SPSY,BPSY,GR: INTEGER;
          VAR A: ARRAY[DIMAB] OF COMPLEX;F3,FC,FO: REAL);
          VAR I,NN: INTEGER;
              X,Y,ABD: REAL;
          BEGIN Y:=SPSY/BPSY/N; ABD:=FO; FO:=FC+FO;
          NN:=N DIV 2-1; GR:=2*GR;
          DLC(A[0],SQRT(1+((ABD)/F3*(1+FO/FC))**GR));
          FOR I:=1 TO NN DO
              BEGIN X:=I*Y;
              DLC(A[I],SQRT(1+((X-ABD)/F3*(1+FO/(X+FC)))**GR));
              DLC(A[N-I],SQRT(1+((X+ABD)/F3*(1+FO/(FC-X)))**GR));
              END; I:=NN+1; DLC(A[I],SQRT(1+(2*I*Y/F3)**GR))
          END (*PROCEDURE FILTMXFL*);
PROCEDURE GRDELAY(T1,T3,F1,F2,F3: REAL; M,SPSY,BPSY: INTEGER;
                          VAR A: ARRAY[DIMAB] OF COMPLEX);
          VAR I,N :INTEGER;
              G,H,FF,TA,TB: REAL;
          BEGIN FF:=T1/(F1-F2); H:=T3/(F3-F2);TA:=(FF-H)/(F1-F3);
          TB:=FF+H-TA*(F1+2*F2+F3); TA:=-TA/3*PI*2; TB:=-TB*PI;
          N:=2**M; H:=SPSY/(BPSY*N);I:=N DIV 2; FF:=I*H;
          PHASESH(A[I],SQR(FF)*(TA*FF+TB));
          FOR I:=I-1 DOWNTO 1 DO
              BEGIN FF:=I*H; G:=FF*FF; PHASESH(A[I],G*(TB+FF*TA));
              PHASESH(A[N-I],G*(TB-FF*TA))
              END
          END (*PROCEDURE GRDELAY*);
PROCEDURE SQP(K,MSYMB: INTEGER; VAR A:ARRAY[DIMAB] OF COMPLEX);
          VAR PIM: REAL;
              I: INTEGER;
          BEGIN MSYMB:=2**MSYMB; PIM:=PI/MSYMB; FOR I:=1 TO K DO
          MLC(A[I],SIN(PIM*I)/(PI*I));MLC(A[0],1/MSYMB)
          END (*PROCEDURE SQP*);
```

```
PROCEDURE EENMR(MSYMB,NIV: INTEGER; VAR R: ARRAY[DIMY] OF INTEGER;
          VAR HELLING: BOOLEAN);
          VAR I,J,K,N,COUNTER:INTEGER;
             BEGIN HELLING := FALSE; COUNTER := 0; N:=2**MSYMB-1;
             FOR I:=0 TO N DO R[I]:= R[I]+1;
             J:=R[N] ; K:=R[N-1] ;
             WHILE NOT (COUNTER = 3) DO
             BEGIN FOR I:=0 TO N DO
             BEGIN IF J > K THEN HELLING := FALSE ;
             IF J < K THEN HELLING := TRUE  ;
             IF ABS(R[I])=NIV THEN
             IF HELLING THEN R[I]:=-NIV ELSE R[I]:=NIV ;
             K:=J ; J:=R[I]
             END;
             IF              ABS ( R[N] - R[0] ) = 2*NIV
             THEN BEGIN IF COUNTER=2 THEN HELLING:=FALSE ELSE R[0]:=R[N];
                        COUNTER := COUNTER + 1
                  END
             ELSE BEGIN     COUNTER := 3;
                            HELLING := TRUE
                  END
   END
                    (*NA  AANROEPEN EENMR, CONTROLEER HELLING, ALS
                    HELLING = FALSE IS EEN FOUT  IN HET EERSTE EN
                    LAATSTE SYMBOOL AANWEZIG, HET PROGRAMMA DIENT DAN
                    GESTOPT TE WORDEN, MET DE HAND DIENT DE SERIE VAN
                    LEVELSHIFT GEWIJZIGD TE WORDEN.*)

END(*PROCEDURE EENMR*);

PROCEDURE IMPINT(K,MSYMB: INTEGER;
          VAR A,CS:ARRAY[DIMAB] OF COMPLEX; VAR R:ARRAY[DIMY] OF INTEGER);
          (*BEREKENT K+1 TERMEN VAN TRANSFORM IMPULSF R*C EN INTEGREERT*)
          VAR I,J,P,S,O1: INTEGER;
              Z: COMPLEX;
              C,SOM,PIO: REAL;
              WISSEL: BOOLEAN;
          BEGIN O1:=MSYMB-1; MSYMB:=2**MSYMB; (*2**MSYMB =AANTAL PULSEN*)
          C:=MSYMB/4/PI;
          J:=MSYMB DIV 2-1; WISSEL:=TRUE; P:=0;
          S:=MSYMB-1; FOR I:=0 TO S DO
             BEGIN IF WISSEL THEN A[P].R:=R[I]*C ELSE BEGIN
             A[P].I:=-R[I]*C; P:=P+1 END; WISSEL:=NOT WISSEL
             END; FFTC(O1,A,CS); CONJ(0,J,A); J:=J+1; A[J]:=A[0];
          XUITP(O1+1,A,CS); PIO:=PI/MSYMB;
          FOR I:=1 TO J DO PHASESH(A[I],-I*PIO);
          P:=MSYMB; IF P>K THEN P:=K ; I:=MSYMB-P;
          WHILE P>J DO
             BEGIN EQU(-A[I].R,A[I].I,A[P]); P:=P-1; I:=I+1
             END; I:=0; FOR P:=MSYMB TO K DO BEGIN EQU(-A[I].R,-A[I].I,
          A[P]); I:=I+1 END;
          SOM:=0; FOR I:=MSYMB-1 DOWNTO 1 DO SOM:=SOM+R[I]*I;
          EQU(-SOM,0,A[0]); FOR I:=1 TO K DO
          EQU(A[I].I/I,-A[I].R/I,A[I])
          END (*PROCEDURE IMPINT*);
```

```
PROCEDURE FN(K,ER,MSYMB: INTEGER; VAR A: ARRAY[DIMAB] OF COMPLEX);
          VAR RPIO,TPI,H: REAL;
              I: INTEGER;
              Z: COMPLEX;
          BEGIN MSYMB:=2**MSYMB; RPIO:=ER/MSYMB; TPI:=2*PI;
          FOR I:=1 TO K DO
             BEGIN H:=RPIO*I;MLC(A[I],SIN(H*PI)/(1-H*H)/I/TPI)
              END; EQU(A[0].R*ER/MSYMB/2,0,A[0])
          END (*PROCEDURE FN*);
          (*REEKSGEN generates a random impuls row R with 2**BPSY levels.*)
PROCEDURE REEKSGEN(MSYMB,BPSY: INTEGER;
          VAR R: ARRAY[INTEGER] OF INTEGER);
          VAR I,IMAX,P,M,TEL,SBB,Q,W,X,JJ,K : INTEGER;
              A: ARRAY [1..16] OF BOOLEAN;
              B: BOOLEAN;
          FUNCTION EXOR(A,B:BOOLEAN):BOOLEAN;
          BEGIN EXOR:=(A AND (NOT B)) OR (B AND (NOT A))
          END;
          BEGIN FOR I:= 1 TO 16 DO A[I]:= FALSE;SBB:=BPSY; TEL:=1;
          IMAX:=2**MSYMB-1; JJ:=2**BPSY-1; B:=A[1];
          FOR I:=0 TO IMAX DO
          BEGIN X:=0; FOR P:= 1 TO BPSY DO
            BEGIN TEL:=TEL-1; IF TEL <1 THEN
              BEGIN Q:=0; W:=1; WHILE SBB>=W DO
                BEGIN W:=W*2; Q:=Q+1
                END; M:=MSYMB+Q-1; SBB:=SBB-W DIV 2; TEL:=2**M; A[M]:=TRUE
              END; IF B THEN X:=X+1 ; X:=X*2;
             CASE M OF
             4,10,5,7: B:=EXOR(A[M],A[3]);
             6,9: B:=EXOR(A[M],A[5]);
             8: B:=EXOR(A[4],EXOR(A[5],EXOR(A[6],A[M])));
             11: B:=EXOR(A[M],A[2]);
             12,13,17,19: B:=EXOR(A[6],EXOR(A[8],EXOR(A[11],A[M])));
             14: B:=EXOR(A[9],EXOR(A[11],EXOR(A[13],A[M])));
             15,16: B:=EXOR(A[11],EXOR(A[13],EXOR(A[14],A[M])));
             18: B:=EXOR(A[M],A[3]);
             20: B:=EXOR(A[M],A[7])
             END; FOR K:=M DOWNTO 2 DO A[K]:=A[K-1]; A[1]:=B
             END; R[I]:=X-JJ
          END END (*PROCEDURE REEKSGEN*);
PROCEDURE IQSTR(M,FBIT:INTEGER;DF,ST: REAL; VAR A,CS: ARRAY[DIMAB] OF COMPLE
          VAR IC,N,K,L: INTEGER;
              X:REAL;
          BEGIN N:=2**M; K:=N-1;X:=PI/2;
          DF:=ROUND(DF*FBIT);DF:=2*PI*DF/N;
          FOR IC:=N DIV 2 -1 DOWNTO 0 DO
              BEGIN A[2*IC+1].R:=A[IC].I*X; A[2*IC].R:=A[IC].R*X
              END;IF ST>0 THEN
              BEGIN L:=N DIV 8; FOR IC:=0 TO K DO
                   BEGIN A[L].I:=A[IC].R+DF*L; L:=L+1;
                   IF L>K THEN L:=0
                   END; FOR IC:=0 TO K DO WITH A[IC] DO
                   BEGIN X:=SIN(R)+ST*SIN(I); R:=COS(R)+ST*COS(I);
                   I:=X
                   END
              END ELSE
              BEGIN FOR IC:=0 TO K DO WITH A[IC] DO
                   BEGIN I:=SIN(R); R:=COS(R)
                   END
              END END (*PROCEDURE IQSTR*);
```

```
FUNCTION  ATAN(VAR A:COMPLEX; VAR OUD:REAL):REAL;
          VAR X: REAL;
              I:INTEGER;
          BEGIN IF ABS(A.R)<1.0E-30 THEN
                BEGIN X:=PI/2; IF A.R*A.I<0 THEN X:=-X
                END ELSE  X:=ARCTAN(A.I/A.R); IF A.R<0 THEN
                BEGIN IF A.I>0 THEN X:=PI+X ELSE X:=X-PI
                END; IF ABS(OUD-X)>PI THEN
                BEGIN I:=ROUND((OUD-X)/2/PI); X:=X+I*2*PI
                END;ATAN:=X; OUD:=X
          END (*FUNCTION ATAN*);
PROCEDURE PHIPRINT(VAR POUT: TEXT; N,INI,INCR,KLMN2,PL:INTEGER;
          VAR A: ARRAY[DIMAB] OF COMPLEX);
          VAR I,J,K,P,R:INTEGER; X,Y:REAL;
          BEGIN I:=INI; R:=2*PL; K:=R; X:=0;Y:=PI/2;
          P:=K*KLMN2;  (*KAR PER REGEL*)
          WHILE I<N DO
          BEGIN WRITE(POUT,I:PL,ATAN(A[I],X)/Y:(PL+5):5);I:=I+INCR;K:=K+R;
          IF (K>132) OR (K>P) THEN BEGIN
          K:=R; WRITELN(POUT) END
          END END (*PROCEDURE PHIPRINT*);
PROCEDURE SKEWAMPL(N,FBIT:INTEGER; PRPFB:REAL;
          VAR A: ARRAY[DIMAB] OF COMPLEX);
          VAR I,N2:INTEGER;
              X:REAL;
          BEGIN N2:=N DIV 2-1; PRPFB:=PRPFB/FBIT/100;
          FOR I:=1 TO N2 DO
              BEGIN X:=PRPFB*I; MLC(A[I],1+X); MLC(A[N-I],1-X)
          END END (*PROCEDURE SKEWAMPL*);
PROCEDURE NLIN1(N:INTEGER; DBDB,GRDB:REAL;
          VAR A: ARRAY[DIMAB] OF COMPLEX);
          CONST C=0.1515973726;
          VAR D,SOM,Y,X:REAL;
              I:INTEGER;
              Z:COMPLEX;
          BEGIN D:=GRDB*C; SOM:=0; FOR I:=N-1 DOWNTO 0 DO
          SOM:=SOM+SQR(A[I].R)+SQR(A[I].I); SOM:=SQRT(SOM/N);
          FOR I:=N-1 DOWNTO 0 DO
              BEGIN X:=ABSCM(A[I])/SOM;Y:=(1-DBDB)/X+DBDB;
              X:=D*(X-1); Z.R:=Y*COS(X); Z.I:=Y*SIN(X);MLPLR(A[I],Z)
          END END (*PROCEDURE NLIN1*);
PROCEDURE LIMITER(N:INTEGER;VAR A: ARRAY[DIMAB] OF COMPLEX);
          VAR I:INTEGER;
          BEGIN FOR I:=N-1 DOWNTO 0 DO DLC(A[I],SQRT(SQR(A[I].R)+SQR(A[I].I)
          END (*PROCEDURE LIMITER*);
PROCEDURE PHIINP(VAR POUT:TEXT; N,INI,INCR,KLMN2,PL:INTEGER;
          VAR A: ARRAY[DIMAB] OF COMPLEX);
          VAR I,K,J,P,R: INTEGER; H:REAL;
          BEGIN I:=INI; R:=2*PL; K:=R;
          P:=K*KLMN2; (*KAR PER REGEL*)
          WHILE I<N DO BEGIN J:=I DIV 2;
          IF I MOD 2=0 THEN H:=A[J].R ELSE H:=A[J].I;
          WRITE(POUT,I:PL,H:(PL+5):5); I:=I+INCR; K:=K+R;
          IF (K>132) OR (K>P) THEN BEGIN
          K:=R; WRITELN(POUT) END
          END END (*PROCEDURE PHIINP*);
PROCEDURE FFTC_1(M:INTEGER;VAR A,CS:ARRAY[DIMAB] OF COMPLEX);
          VAR N:INTEGER;
          BEGIN N:=2**M-1; CONJ(0,N,A); FFTC(M,A,CS);
          CONJ(0,N,A); VARA1(N,A,1/(N+1))
          END (*PROCEDURE FFTC_1 , VAN TIJD DOM NAAR FREQ DOMEIN*);
```

```
BEGIN{0} WHILE NOT ((PAR10=IQ_STR) OR EOF(PINP)) DO BEGIN{1}

        READLN(PINP,PAR10); CASE PAR10 OF

        ANEW_WEER:
            BEGIN  PRENTHOOFD(PAR10);READLN(PINP,M,BPSY,MSPSY);
            N:=2**M; N2:=N DIV 2; SPSY:=2**MSPSY; MSYMB:=M-MSPSY;
            SYMB:=2**MSYMB;FBIT:=(N*BPSY) DIV SPSY;
            TABU; WRITELN(POUT,SYMB:4,'  SYMBOLS,     ',BPSY:3,
            '  BITS/SYMBOL,     ',SPSY:3,'  SAMPLES/SYMBOL,')
            END;
        REEKS_GEN:
            BEGIN PRENTHOOFD(PAR10);REEKSGEN(MSYMB,BPSY,R);
            DCLEVELCHECK END;
        LEVELSHIFT:
            BEGIN PRENTHOOFD(PAR10); READ(PINP,II);
            TABU; WRITELN(POUT,' TOPLEVEL',II:4); EENMR(MSYMB,II,R,HELLING);
            DCLEVELCHECK; PRIR;
            IF NOT HELLING THEN BEGIN WHILE NOT EOF(PINP) DO READLN(PINP);
            TABU; WRITELN(POUT,'CHECK FIRST AND LAST SYMBOL');
            ZENDER:=FALSE END; PAGE(POUT)
            END;
        IMP_INT:
            BEGIN PRENTHOOFD(PAR10);     IF NOT CONTROL(MSYMB,II,R) THEN
            BEGIN TABU; WRITELN(POUT,'DCLEVELCHECK IS FALSE, MODIFY SERIES');
            WHILE NOT EOF(PINP) DO READLN(PINP); ZENDER:=FALSE END
            ELSE IMPINT(N2,MSYMB,AB,CS,R) END;
        SQUAREPULS:
            BEGIN PRENTHOOFD(PAR10);SQP(N,MSYMB,AB) END;
        RAISEDCOSINE:
            BEGIN PRENTHOOFD(PAR10);READLN(PINP,II);
            TABU;WRITELN(POUT,' RC-PULS OVER',II:3,'  SYMBOLEN');
            FN(N,II,MSYMB,AB) END;
        NYQ_III:
            BEGIN PRENTHOOFD(PAR10); READLN(PINP,RALG);
            TABU;WRITELN(POUT,'ALFA VOOR NYQIII:   ',RALG:10:5);
            NYQIII(M,MSYMB,RALG,AB) END;
        XPT_TIJD:
            BEGIN PRENTHOOFD(PAR10);XPT(M,AB,CS) END;
        PHASE_DEV:
            BEGIN PRENTHOOFD(PAR10); READLN(PINP,R1);
            TABU;WRITELN(POUT,' FACTOR:  ',R1:10:5); VARA1(N2,AB,R1/FBIT) END;
        IQ_STR:
            BEGIN PRENTHOOFD(PAR10); READLN(PINP,STST,STFREQ);
            TABU;WRITELN(POUT,'STOORSTERKTE:   ',STST:10:5,'   STOORFREQ:   ',
            STFREQ:10:5); IQSTR(M,FBIT,STFREQ,STST,AB,CS);
            ZENDER:=TRUE END;

        END{CASE} END{1} END{0};

        END.
```

```
FUNCTION  ZENDER(VAR R:ARRAY[DIMY] OF INTEGER;VAR M,N:INTEGER):BOOLEAN;

          TYPE PCK10=(REEKS_GEN,IMP_INT,NYQ_III,XPT_TIJD,SQUAREPULS,
                      IQ_STR,PHASE_DEV,ANEW_WEER,LEVELSHIFT,RAISEDCOSINE)
          VAR II,N2,MSYMB,MSPSY,FBIT:INTEGER; HELLING:BOOLEAN;
              R1,RALG,STST,STFREQ:REAL; PAR10:PCK10;
              PAR80:PACKED ARRAY[1..72] OF CHAR;


PROCEDURE PRENTHOOFD(VAR PAR10: PCK10);
          BEGIN WRITELN(POUT);TABU;
          WRITELN(POUT,'++++++++++++',PAR10,'  ++++++++++++++++');
          READLN(PINP,PAR80); WRITELN(POUT,PAR80); WRITELN(POUT)
          END (*PROCEDURE PRENTHOOFD*);

PROCEDURE PRIR;
          VAR I,II:INTEGER;
          BEGIN II:=SYMB-1; TABU; FOR I:=0 TO II DO
          BEGIN IF I MOD 32 =0 THEN BEGIN WRITELN(POUT); TABU END;
          WRITE(POUT,R[I]:2)
          END; WRITELN(POUT)
          END (*PROCEDURE PRIR*);

PROCEDURE DCLEVELCHECK; VAR I:INTEGER;
          BEGIN READ(PINP,II);II:=II-1;IF II>-1 THEN
          FOR I:=0 TO II DO READ(PINP,R[I]);
          READLN(PINP); IF NOT CONTROL(MSYMB,II,R) THEN BEGIN
          TABU; WRITELN(POUT,' UITKOMST CONTROL <>0: ',II) END
          END (*PROCEDURE DCLEVELCHECK*);
```

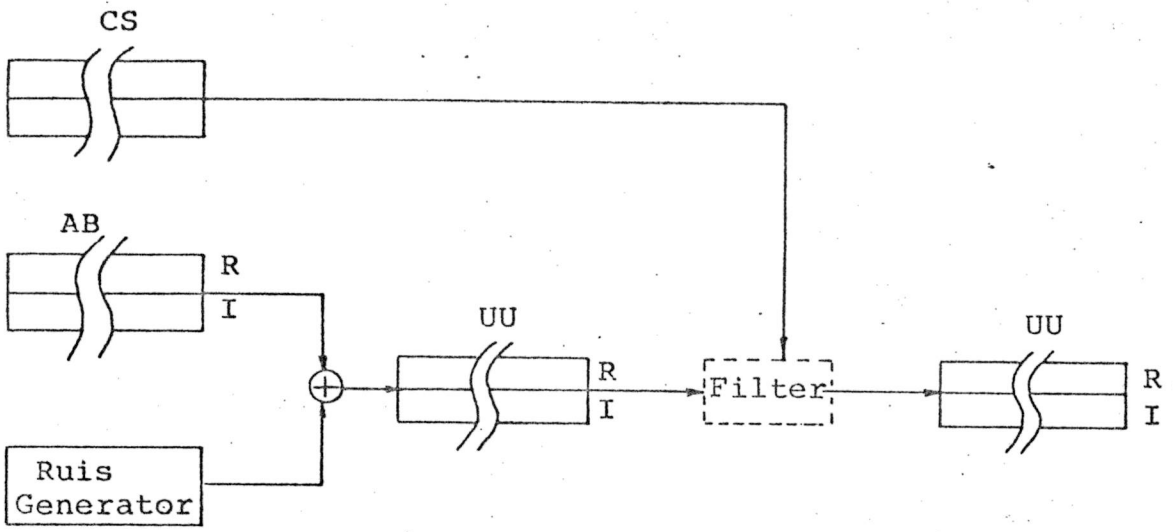


fig. 1 procedure Ruis-Filter

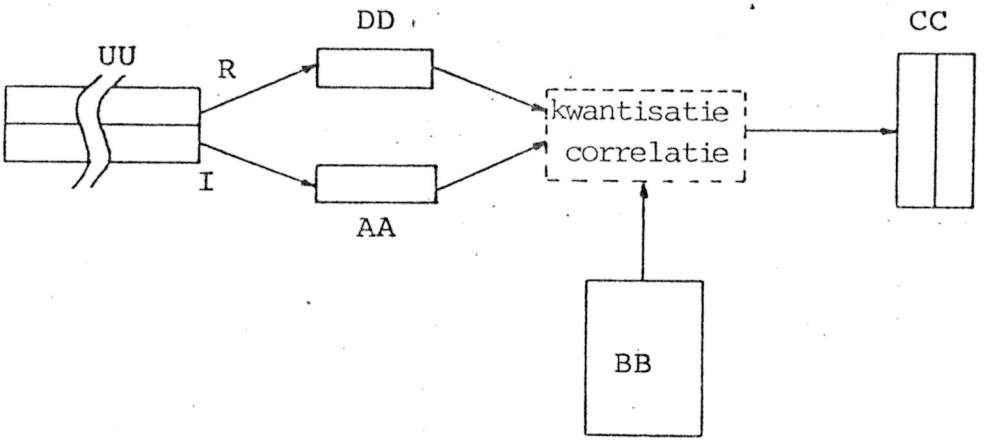


fig. 2 procedure Correlator

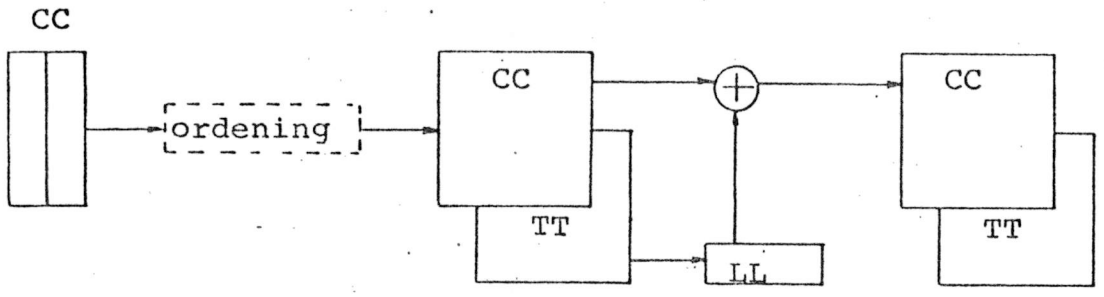


fig.3 procedure Historie

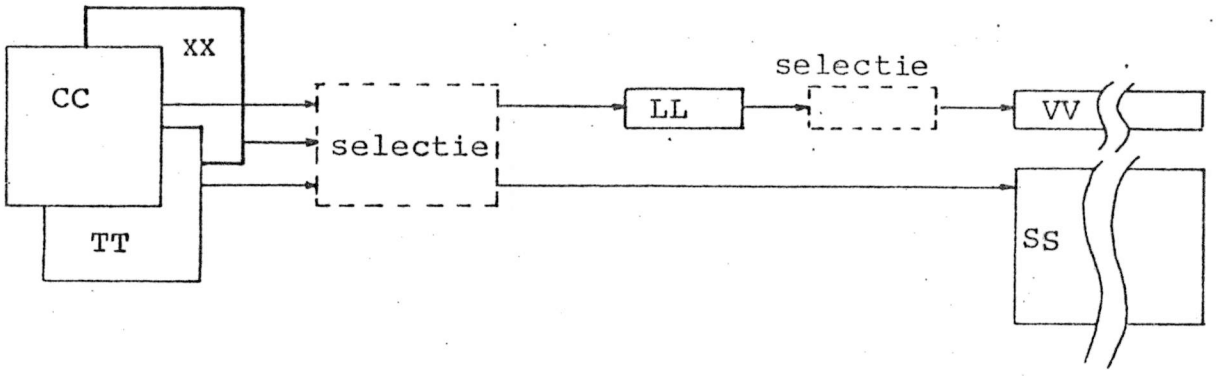


fig. 4 procedure Survivors

Bits	symbool	symbool	fase ver-andering
00	-1	0	0
01	-3	-2	$-\pi/2$
10	1	2	$\pi/2$
11	3	-4	$-\pi$
11	3	4	π

Tabel 1 Symbolen en fase verandering

Toestands nummering	fasepunt
1	0
2	$\pi/2$
3	π
4	$-\pi/2$

Tabel 2 Toestandsnummering voor replica's van de eerste orde

symbool / fase-punt	0, -2, -4	0, 2, 4
0	S(AC)+S(BD)	S(AC)-S(BD)
$\pi/2$	S(BC)+S(AD)	S(BC)-S(AD)
π	-S(AC)-S(BD)	-S(AC)+S(BD)
$-\pi/2$	-S(BC)-S(AD)	-S(BD)+S(AD)

Tabel 3 Correlatie tabel

$A = \cos(\phi)$ $B = \sin(\phi)$ $C = \cos(\theta_k)$ $D = \sin(\theta_k)$

ϕ = signaal θ_k = origineel $S(AC) = \sum_k (A.C)$