Ontwerp van Cosinus-gemoduleerde subband-filterbanken

Michiels, F.H.M.

Award date:
1994

TECHNISCHE UNIVERSITEIT EINDHOVEN

FACULTEIT ELEKTROTECHNIEK

VAKGROEP Signaalverwerking

SIGNAL PROCESSING

Ontwerp van cosinus-gemoduleerde subband-filterbanken

door

F.H.M. Michiels

ESP-13-94

Verslag van een afstudeeronderzoek, verricht in de vakgroep ESP, onder leiding van ir. J.H.F. Ritzerfeld, in de periode mei 1994 - december 1994.

Eindhoven, 20 december 1994.

Samenvatting

Subband coding wordt vaak toegepast in coderings- en compressie systemen. De te coderen/compresseren signalen (audio, spraak, enz.) worden gesplitst in verschillende frequentie banden (subband signalen), waarop de subband codering toegepast wordt. Voor de splitsing in verschillende frequentie banden worden digitale filters (een analyse filter bank) gebruikt.

Na de subband codering worden de ontstane subband signalen samengevoegd m.b.v. een synthese filter bank. Een maximaal gedecimeerde analyse- en synthese filterbank dienen zo ontworpen te worden dat het een perfect reconstructie systeem is. Paraunitairiteit is een belangrijk begrip bij het ontwerp van zo'n systeem. In het ontwerp van multirate filterbanken vormt de polyfase representatie een belangrijke rol omdat de theorie achter multirate systemen vereenvoudigd wordt maar het leidt ook tot rekenkundige efficiënte implementaties.

Cosinus gemoduleerde filter banken vormen een belangrijke klasse in de verzameling van M-kanaals uniforme filterbanken. Deze cosinus gemoduleerde analyse filter bank bestaat uit analyse filters die afgeleid worden van één enkel prototype filter. Dit systeem is bijzonder efficiënt in ontwerp en implementatie. De implementatie bestaat uit de implementatie van één filter dat uitgevoerd kan in een lattice- of transversale structuur en de implementatie van een Discrete Cosinus Transformatie van het type IV (DCT-IV).

In elke digitale filter bank implementatie dienen vermenigvuldigers coëfficiënten en interne signalen gerepresenteerd worden in een gequantiseerde vorm. Deze implementatie kan uitgevoerd worden in floating- of fixed-point architectuur.

Lattice structuren staan er bekend om dat zij bestand zijn tegen quantisatie effecten terwijl transversale structuren dit niet zijn. Daarentegen zijn transversale structuren makkelijk te implementeren op een DSP en lattice structuren niet. M.b.v. van simulaties wordt bekeken hoe zwaar deze voor- en nadelen tegen elkaar opwegen.

Inhoudsopgave

Hoofdstuk 1

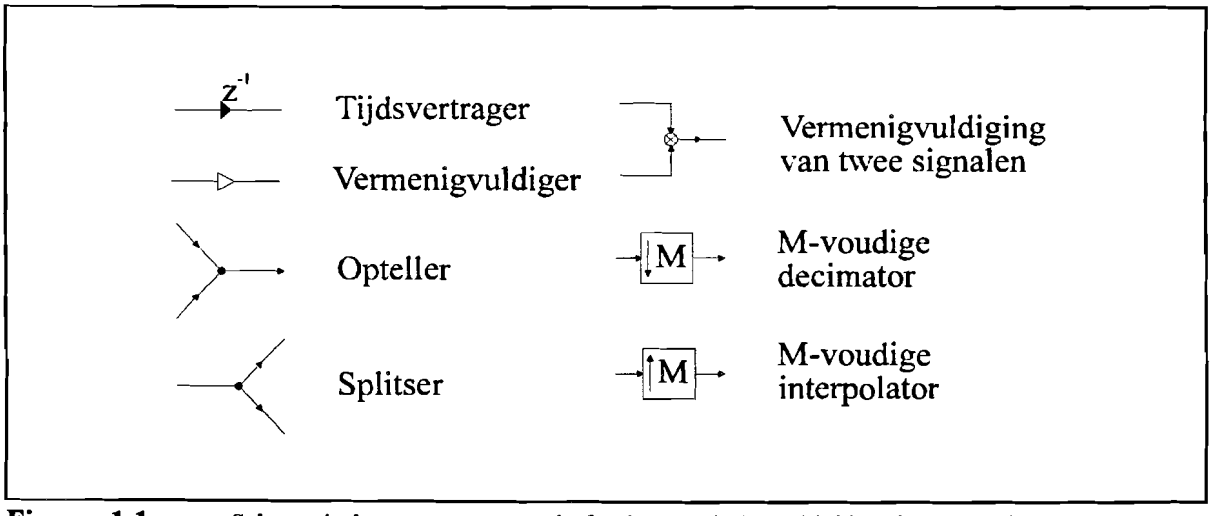
Inleiding in filter banken

1.1 Inleiding

In dit hoofdstuk geven we ten eerste aandacht aan gebruikte notaties en schematische presentatie van fundamentele bouwstenen die in diverse figuren in dit verslag gebruikt worden. In onderzoek naar en bij het ontwerpen van filterbanken is de polyfase representatie niet meer weg te denken. Samen met deze polyfase representatie wordt de definitie van paraunitairiteit van systemen gebruikt om filter banken met een perfecte reconstructie te ontwerpen. Voor een uitgebreide behandeling van deze zaken verwijzen we naar [1].

1.2 Notaties, Definities en Rekenregels

Signaal processing systemen bestaan uit een aantal fundamentele bouwstenen, die onderdeel zijn van filters, multirate systemen en filter banken. De bouwstenen die we in dit verslag gebruiken zijn in Figuur 1.1 schematisch weergegeven.



Figuur 1.1 Schematische weergave van de fundamentele bouwblokken in een multirate systeem.

De in dit verslag gebruikte notaties zijn hieronder opgesomd. Tevens zijn er in deze opsomming een aantal belangrijke definities opgenomen, die veel in de tekst gebruikt worden.

- $H_*(z)$: coëfficiënten conjugatie van $H(z)$.
- $\tilde{H}(z)=H_*(z^{-1})$: paraconjugatie van $H(z)$.
- $\tilde{\mathbf{H}}(z)=\mathbf{H}_*^T(z^{-1})$: paraconjugatie van $\mathbf{H}(z)$
- $\mathbf{H}_*(z)$: conjugatie van coëfficiënten $\mathbf{H}(z)$

- Met $[\mathbf{A}]_{ij}$ wordt het element uit matrix **A** met rij-index i en kolom-index j voorgesteld.
- $\text{diag}(a_0 , a_1 , \cdots , a_{M-1})$: de M×M diagonaal matrix **A**
- I_M : de M×M eenheid matrix
- J_M : de M×M reversed matrix, bijvoorbeeld:

$$J_3 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

- **A** is een M×M diagonaal matrix **JA** wordt dan reversed diagonaal genoemd.
- $\mathbf{A}^T$: de getransponeerde van matrix **A**
- De DFT en IDFT zijn respectievelijk gedefinieerd als:

$$X[k] = \sum_{n=0}^{M-1} x[n] W_M^{kn} , \qquad x[n] = \frac{1}{M} \sum_{k=0}^{M-1} X[k] W_M^{-kn} \qquad W_M^{kn} = e^{-j2\pi/M}$$

De DFT kan ook geschreven worden in matrix vorm waarbij gebruik gemaakt wordt van de DFT matrix $\mathbf{W}_M$ die bestaat uit de elementen W_M^{kn} . Vaak wordt het subscript M weggelaten omdat uit de context het vaak duidelijk is. Als we vector **X** laten bestaan uit de elementen X[k] $0 \le k \le M-1$ en vector **x** uit x[n] $0 \le n \le M-1$ dan zijn de DFT en IDFT respectievelijk gelijk aan $\mathbf{X}=\mathbf{W}\mathbf{x}$ en $\mathbf{x}=\mathbf{W}^*\mathbf{X}$.

Uit bovenstaande definities zijn een aantal belangrijke rekenregels af te leiden. Dit zullen we hieronder opsommen omdat ze herhaaldelijk gebruikt worden in de rest van dit verslag:
- $\tilde{H}(z)H(z)$ geëvalueerd op de eenheidscirkel is de magnitude squared response $|H(e^{j\omega})|^2$.
- Als H(z) een reel coëfficiënten FIR filter voorstelt dan is $z^{-N}H(z^{-1})$ de tijd-reversed van H(z).
- Als $H(z)=H_1(z)H_2(z)$ dan geldt ook $\tilde{H}(z)=\tilde{H}_1(z)\tilde{H}_2(z)$.
- Als $H(z)=H_1(z)+H_2(z)$ dan geldt ook $\tilde{H}(z)=\tilde{H}_1(z)+\tilde{H}_2(z)$.
- $\mathbf{J}\,\text{diag}(a_0 , a_1 , \cdots , a_{M-1})\mathbf{J}=\text{diag}(a_{M-1} , \cdots , a_1 , a_0)$
- **A** en **B** zijn M×M diagonaal matrices dan is **AJB** reversed diagonaal .
- **C** is reversed diagonaal dan geldt $\mathbf{JCJ}=\mathbf{C}^T$
- $(\mathbf{ABC})^T=\mathbf{A}^T\mathbf{B}^T\mathbf{C}^T$
- $(\mathbf{A}+\mathbf{B}+\mathbf{C})^T=\mathbf{A}^T+\mathbf{B}^T+\mathbf{C}^T$
- $\mathbf{J}^T=\mathbf{J}$ en $\mathbf{JJ}=\mathbf{I}$

Bij het ontwerpen en simuleren van filter banken maken we gebrruik van het software pakket 386-MATLAB versie 3.5. We merken op dat de indexering van matrices in MAT-LAB begint bij 1 i.p.v. 0 zoals bij de rekenregels hierboven.

1.3 Paraunitairiteit

Een systeem met r ingangen en p uitgangen met overdrachtsfuncties van elke ingang naar elke uitgang kan beschreven worden door de p×r transfer matrix $\mathbf{H}(z)=[H_{km}(z)]$, met $H_{km}(z)$ de overdrachtsfunctie van input m naar uitgang k. Deze transfer matrices zijn bijzonder geschikt om filterbanken te karakteriseren.

Een belangrijk begrip bij het ontwerpen van filterbanken is paraunitairiteit, die voor een transfer matrix als volgt gedefinieerd is:

$$\tilde{\boldsymbol{H}}(z)\boldsymbol{H}(z)=c\boldsymbol{I}_r, \qquad \forall_z, \qquad c>0 \tag{1.1}$$

Enkele eigenschappen van paraunitaire systemen zijn:

1 Als de M×1 transfer functie $\mathbf{H}(z)=[H_0(z)\cdots H_{M-1}(z)]^T$ paraunitair ($\tilde{\mathbf{H}}(z)\mathbf{H}(z)=c$) is dan geldt de power complementaire eigenschap:

$$\sum_{k=0}^{M-1} \tilde{H}_k(z)H_k(z)=c^2, \qquad \forall_z \tag{1.2}$$

2 Als $\mathbf{H}(z)$ paraunitair is dan zijn dat ook: $\mathbf{H}(z^M)$, $\mathbf{H}^T(z)$ en $\tilde{\mathbf{H}}(z)$.

3 Als $\mathbf{H}(z)$ een cascade (produkt) is van paraunitaire systemen dan is $\mathbf{H}(z)$ ook paraunitair.

4 Als $\mathbf{H}(z)$ paraunitair is en een vierkant matrix is dan geldt

$$\boldsymbol{H}^{-1}(z)=c\tilde{\boldsymbol{H}}(z), \qquad \forall_z$$

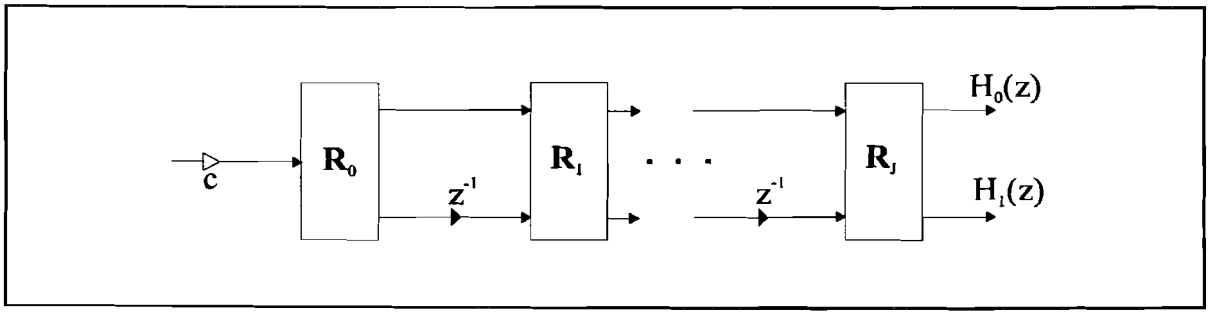
Bovenstaande eigenschap 1 willen we toelichten met een speciaal geval waarbij M=2, en waarbij we uitgaan van power-complementairiteit. Want indien twee functies power-complementair met elkaar zijn dan geldt:

$$\tilde{H}_0(z)H_0(z)+\tilde{H}_1(z)H_1(z)=c^2, \qquad c>0 \tag{1.4}$$

waardoor de vector

$$\boldsymbol{H}(z)=\begin{bmatrix} H_0(z) \\ H_1(z) \end{bmatrix} \tag{1.5}$$

paraunitair is. Dit systeem is te implementeren volgens een zogenaamde lattice structuur afgebeeld in Figuur 1.2.



Figuur 1.2 Een lattice structuur die twee functies implementeert met de power complementaire eigenschap.

De transfer matrix $\mathbf{H}(z)$ is te schrijven als:

$$\boldsymbol{H}(z)=\boldsymbol{R}_J\Delta(z)\boldsymbol{R}_{J-1}\Delta(z)\cdots\cdots\boldsymbol{R}_1\Delta(z)\boldsymbol{R}_0 \tag{1.6}$$

$\mathbf{R}_0$ is een 2×1 matrix waarvoor geldt dat $\mathbf{R}_0^T\mathbf{R}_0=1$ (genormaliseerd en paraunitair). De 2×2 matrix $\mathbf{R}_i$ $1\le i\le J$ staat bekend als een rotatie matrix waarvoor geldt dat $\mathbf{R}_i^T\mathbf{R}_i=1$ (genormaliseerd en paraunitair). Zowel $\mathbf{R}_0$ als $\mathbf{R}_i$ hebben één vrijheidsparameter θ die we lattice coëfficiënt noemen en beide matrices bestaan uit elementen die de cosinus of sinus zijn van deze lattice coëfficiënt. Er zijn twee keuzes voor de $\mathbf{R}_0$ matrix en acht voor die van $\mathbf{R}_i$. Formule (1.7) geeft de keuze die we in het gehele verslag zullen gebruiken. Tevens geeft deze formule ook de transfer functie $\Delta(z)$ die ook paraunitair is ($\tilde{\Delta}(z)\Delta(z)=\mathbf{I}$).

$$\Delta(z)=\begin{bmatrix}1 & 0\\ 0 & z^{-1}\end{bmatrix},\qquad \boldsymbol{R}_0=\begin{bmatrix}\cos\theta_0\\ \sin\theta_0\end{bmatrix},\qquad \boldsymbol{R}_i=\begin{bmatrix}\cos\theta_i & \sin\theta_i\\ \sin\theta_0 & -\cos\theta_i\end{bmatrix}_{i=1\cdots J} \tag{1.7}$$

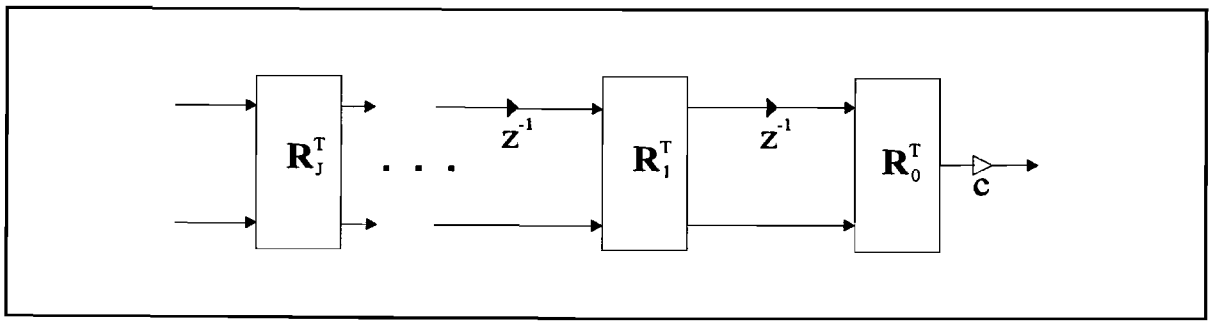
Het is duidelijk dat $\mathbf{H}(z)$ paraunitair is omdat het een cascade is van paraunitaire systemen. De paraconjugatie van de 2×1 transfer functie $\mathbf{H}(z)$, $\tilde{\mathbf{H}}(z)$ is een non-causaal systeem en dus niet te implementeren. $z^{-(J-1)}\tilde{\mathbf{H}}(z)$ is causaal en te schrijven als:

$$z^{-(J-1)}\tilde{\boldsymbol{H}}(z)=\boldsymbol{R}_0^T\Gamma(z)\boldsymbol{R}_1^T\Gamma(z)\cdots\cdots\boldsymbol{R}_{J-1}^T\Gamma(z)\boldsymbol{R}_J^T \tag{1.8}$$

met

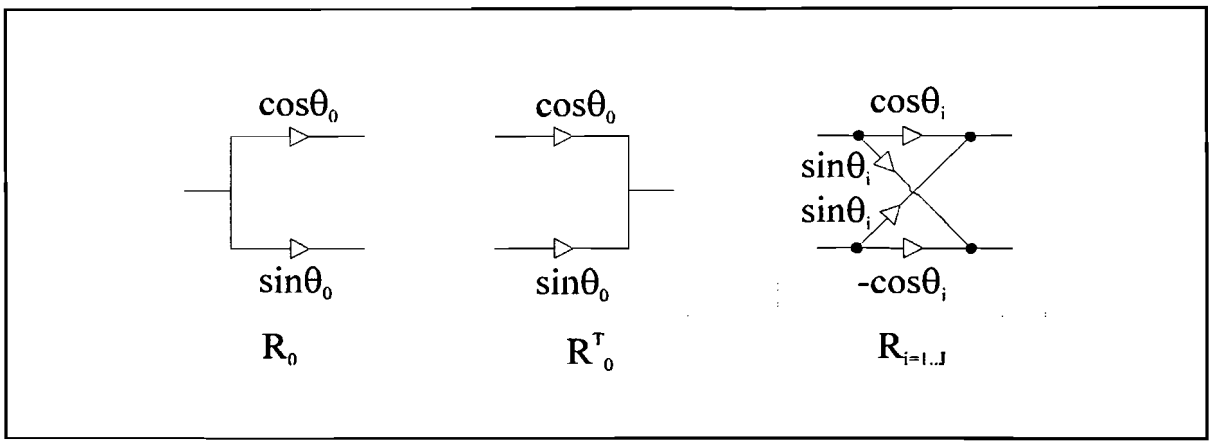
$$\Gamma(z)=z^{-1}\tilde{\Delta}(z)=\begin{bmatrix}z^{-1} & 0\\ 0 & 1\end{bmatrix},\qquad \boldsymbol{R}_0^T=\begin{bmatrix}\cos\theta_0 & \sin\theta_0\end{bmatrix},\qquad \boldsymbol{R}_i^T=\boldsymbol{R}_i=\begin{bmatrix}\cos\theta_i & \sin\theta_i\\ \sin\theta_0 & -\cos\theta_i\end{bmatrix}_{i=1\cdots J} \tag{1.9}$$

De 1×2 transfer functie $z^{-(J-1)}\tilde{\mathbf{H}}(z)$ is ook te implementeren volgens een lattice structuur, zie Figuur 1.3.

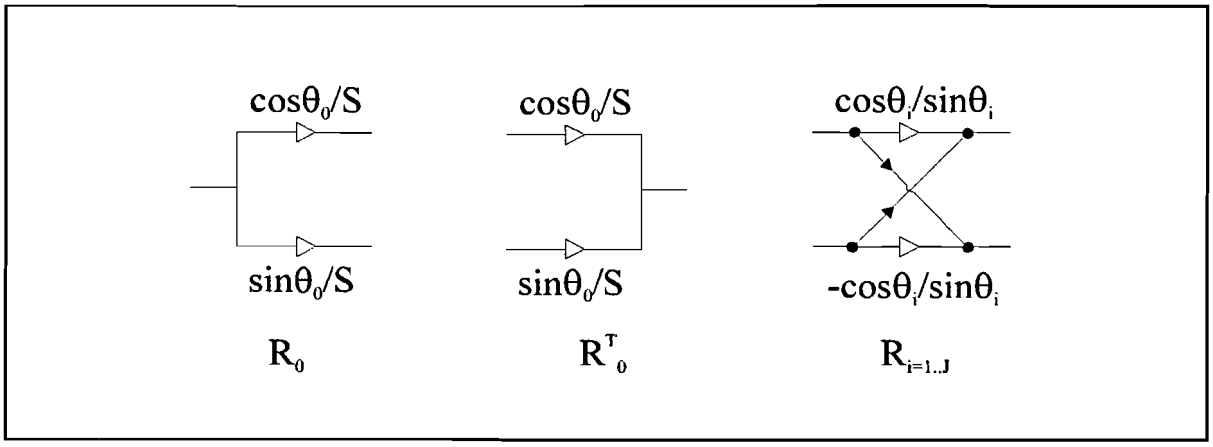


Figuur 1.3 De implementatie van de geparaconjugeerde functies uit Figuur 1.2.

De implementaties van de matrices $\mathbf{R}_0$, $\mathbf{R}_0^T$ en $\mathbf{R}_i$ zijn in Figuur 1.4 afgebeeld. De matrices $\mathbf{R}_i$ worden vier vermenigvuldigers lattices genoemd, omdat in de praktijk een meer efficiënte structuur wordt gebruikt bestaande uit twee vermenigvuldigers, zie Figuur 1.5.



Figuur 1.4 Bouwblokken van een vier-vermenigvuldigers lattice.



Figuur 1.5 Bouwblokken van een twee-vermenigvuldigers lattice.

De constante S is gelijk aan:

$$S=\prod_{i=1}^{J}\sin\theta_i \tag{1.10}$$

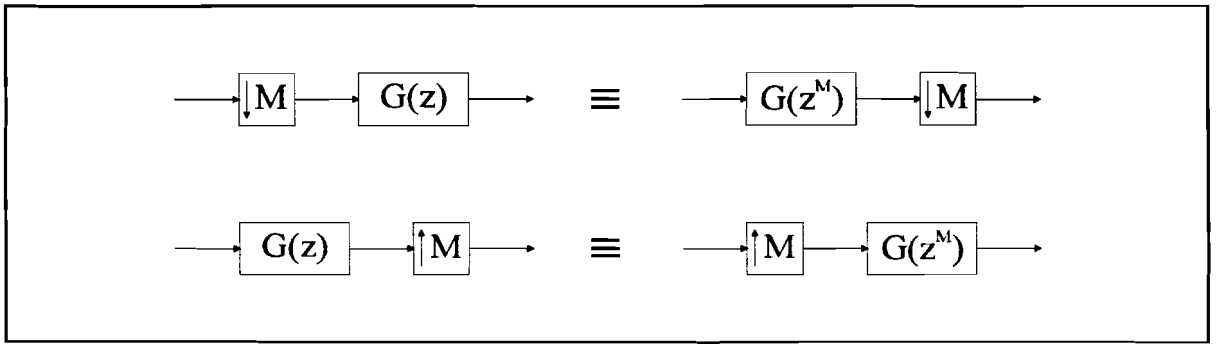
en de matrices $\mathbf{R}_0$, $\mathbf{R}_0^T$, $\mathbf{R}_i$ en $\mathbf{R}_i^T$ zijn nu te schrijven als:

$$\boldsymbol{R}_0=\begin{bmatrix}\cos\theta_0\cdot S\\ \sin\theta_0\cdot S\end{bmatrix},\qquad \boldsymbol{R}_0^T=\begin{bmatrix}\cos\theta_0\cdot S & \sin\theta_0\cdot S\end{bmatrix},\qquad \boldsymbol{R}_i^T=\boldsymbol{R}_i=\begin{bmatrix}\dfrac{\cos\theta_i}{\sin\theta_i} & 1\\ 1 & \dfrac{-\cos\theta_i}{\sin\theta_0}\end{bmatrix}_{i=1\cdots J} \tag{1.11}$$

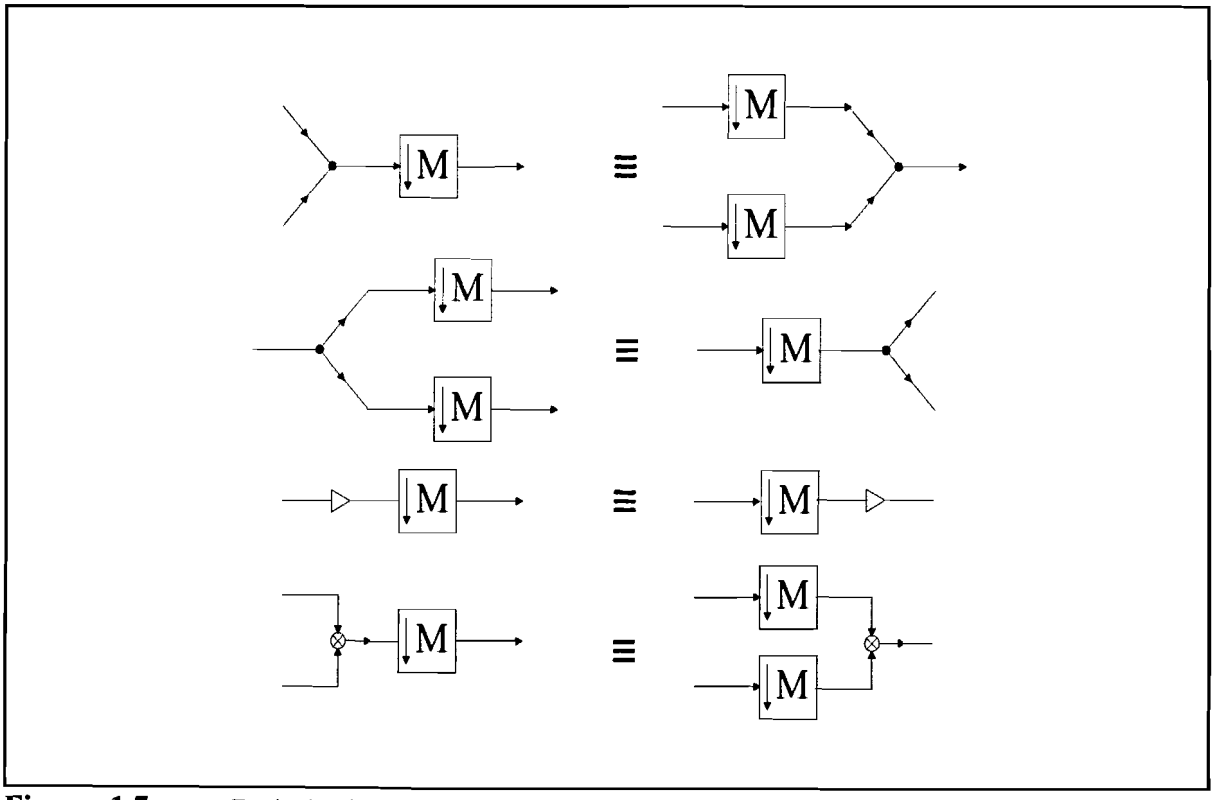
De matrices uit Formule (1.11) zijn ook paraunitair maar niet genormaliseerd ($\mathbf{R}_i^T\mathbf{R}_i=\alpha\mathbf{I}$ en $\mathbf{R}_0^T\mathbf{R}_0=\alpha$) .

1.4 Polyfase representatie

Een belangrijke definitie in de multirate signaal processing is de polyfase representatie. Deze definitie leidt tot rekenkundig efficiënte implementaties van decimatie- en interpolatie filters die bijvoorbeeld deel uit maken van filterbanken. In combinatie met de polyfase representatie worden de zogenaamde noble identities en enkele equivalenties dikwijls gebruikt. De noble identities zijn gegeven in Figuur 1.6 en de in dit verslag gebruikte equivalenties in Figuur 1.7.



Figuur 1.6 De noble identities.



Figuur 1.7 Equivalenties van samengestelde bouwblokken.

Een FIR filter (non-causaal/causaal) kan beschreven worden als:

$$H(z)=\sum_{n=-\infty}^{\infty} h[n]z^{-n} \tag{1.12}$$

We kunnen H(z), gegeven een integer M, uiteen rafelen en schrijven als:

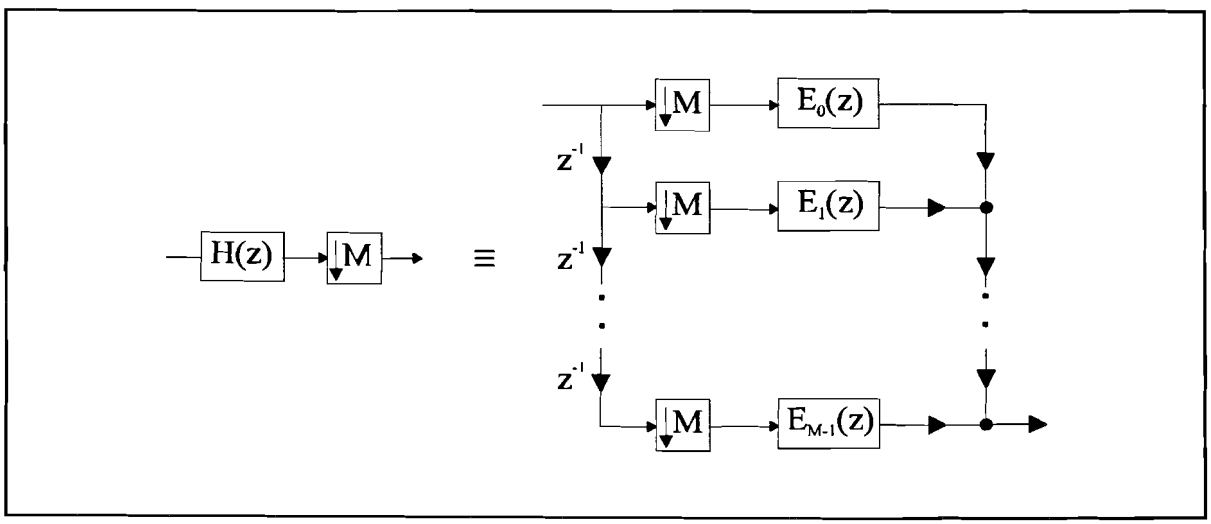
$$\begin{aligned} H(z)=&\sum_{n=-\infty}^{\infty} h[nM]z^{-nM} \\ &+z^{-1}\sum_{n=-\infty}^{\infty} h[nM+1]z^{-nM} \\ &\vdots \\ &+z^{-(M-1)}\sum_{n=-\infty}^{\infty} h[nM+M-1]z^{-nM} \end{aligned} \tag{1.13}$$

En in een compacte vorm als:

$$H(z)=\sum_{l=0}^{M-1} z^{-l}E_l(z^M), \qquad E_l(z)=\sum_{n=-\infty}^{\infty} e_l[n]z^{-n}, \tag{1.14}$$

$$e_l[n]=h[Mn+l], \qquad 0\le l\le M-1$$

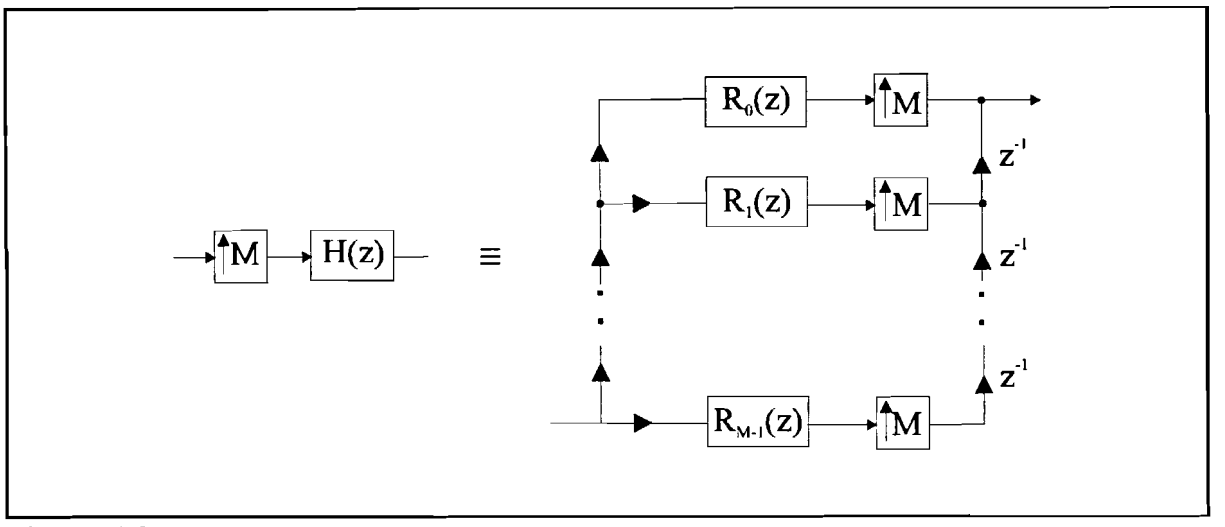
Formule (1.14) wordt een Type 1 polyfase representatie (met betrekking tot M) genoemd en $E_l(z)$ de polyfase componenten van H(z). Een decimatie filter kan gebruik makend van de polyfase representatie efficiënter geïmplementeerd worden dan directe implementatie, zie Figuur 1.8.



Figuur 1.8 Implementatie van een M-voudig decimatie filter.

Voor de efficiënte implementatie van een interpolatie filter (Figuur 1.9) wordt gebruik gemaakt van de Type 2 polyfase representatie, die als volgt gedefinieerd is:

$$H(z)=\sum_{l=0}^{M-1} z^{-(M-1-l)}R_l(z^M), \qquad R_l(z)=E_{M-1-l}(z) \tag{1.15}$$



Figuur 1.9 Implementatie van een M-voudig interpolatie filter.

Stel dat H(z) een causaal FIR filter met een impulsresponsie ter lengte L is. En indien L een veelvoud van M (L=kM) is dan kan $E_l(z)$ uit Formule (1.14) geschreven worden als:

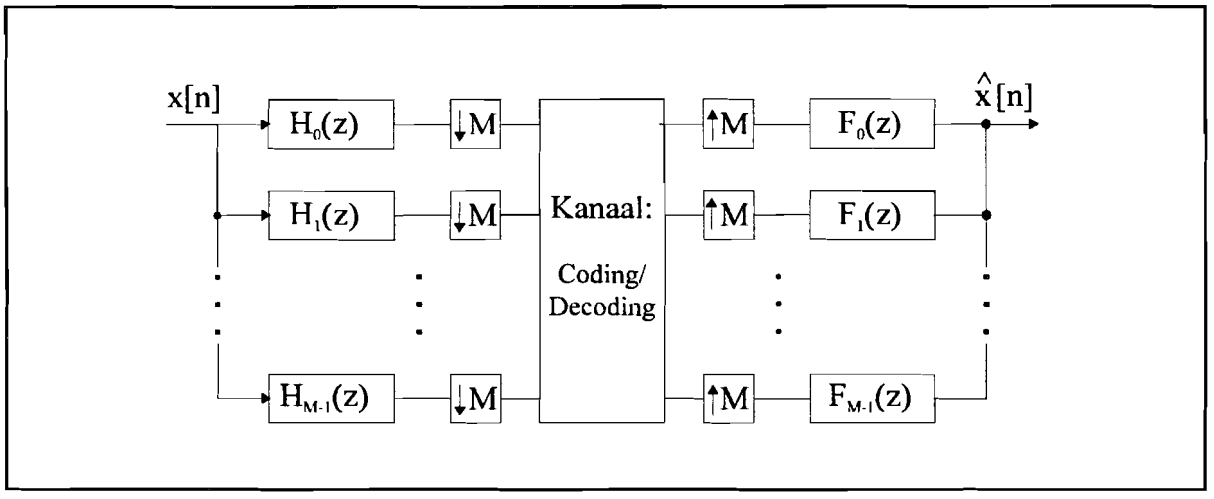
$$E_l(z)=\sum_{n=-0}^{k-1} e_l[n]z^{-n} \tag{1.16}$$

wat inhoudt dat de impulsresponsies van $E_l(z)$ allemaal dezelfde lengte k hebben.

1.5 M-kanaals filterbanken

Een M-kanaals maximaal gedecimeerde filter bank is afgebeeld in Figuur 1.10. Het ingangssignaal x[n] wordt in M subsignalen gesplitst d.m.v. M analyse filters $H_k(z)$ en daarna gedecimeerd. Het kanaal bestaat achtereenvolgens uit een codering gedeelte dat het aantal bits reduceert, een opslag- of transmissie kanaal en een decoderings gedeelte dat de aangeboden bits expandeert. In dit verslag veronderstellen we dat de uitgangssignalen van het kanaal gelijk zijn aan de ingangssignalen van het kanaal. De uitgangssignalen van het kanaal worden geïnterpoleerd m.b.v. M-voudige interpolators en samengevoegd (synthese) m.b.v. de synthese filters $F_k(z)$ tot $\hat{x}[n]$.

In [1] is een beknopte ontwerp procedure en enkele ontwerp voorbeelden gegeven van M-kanaals filter banken. [2] en [3] geven uitgebreidere procedures en voorbeelden.



Figuur 1.10 Een M-kanaals maximaal gedecimeerde filter bank.

Om tot een uitdrukking van $\hat{X}(z)$ te komen definiëren we eerst de volgende vectoren en matrix:

$$\boldsymbol{h}(z)=\begin{bmatrix} H_0(z) \\ H_1(z) \\ \vdots \\ H_{M-1}(z) \end{bmatrix} \quad \boldsymbol{f}(z)=\begin{bmatrix} F_0(z) \\ F_1(z) \\ \vdots \\ F_{M-1}(z) \end{bmatrix} \quad \boldsymbol{e}(z)=\begin{bmatrix} 1 \\ z^{-1} \\ \vdots \\ z^{-(M-1)} \end{bmatrix} \quad \boldsymbol{x}(z)=\begin{bmatrix} X(z) \\ X(zW) \\ \vdots \\ X(zW^{M-1}) \end{bmatrix} \tag{1.17}$$

$$\boldsymbol{H}(z)=\begin{bmatrix} H_0(z) & H_1(z) & \cdots & H_{M-1}(z) \\ H_0(zW) & H_1(zW) & \cdots & H_{M-1}(zW) \\ \vdots & \vdots & \ddots & \vdots \\ H_0(zW^{M-1}) & H_1(zW^{M-1}) & \cdots & H_{M-1}(zW^{M-1}) \end{bmatrix}$$

waardoor geldt dat:

$$\hat{X}(z)=\frac{1}{M}\boldsymbol{f}^T(z)\boldsymbol{H}^T(z)\boldsymbol{x}(z)=\boldsymbol{A}^T(z)\boldsymbol{x}(z) \quad met \quad \boldsymbol{A}(z)=\frac{1}{M}\boldsymbol{H}(z)\boldsymbol{f}(z)=\begin{bmatrix} A_0(z) \\ A_1(z) \\ \vdots \\ A_{M-1}(z) \end{bmatrix} \tag{1.18}$$

1.5.1 Perfecte Reconstructie (PR) filter banken

In een filterbank ontstaan verschillende soorten fouten zoals
- **aliasing**: het reconstructie signaal $\hat{X}(z)$ bestaat uit verschoven versies van $X(z)$, $X(zW^l)$,

l>0 ten gevolge van het decimatie en interpolatie proces. $X(zW^l)$ wordt de l-de aliasing term genoemd en $A_l(z)$ de versterking van deze term. De filterbank is aliasing vrij als:

$$A_l(z)=0, \qquad 1\le l\le M-1 \tag{1.19}$$

- **Amplitude en Fase Distorsie**: is de filterbank aliasing vrij dan kunnen we het reconstructie signaal $\hat{X}(z)$ schrijven als $\hat{X}(z)=A_0(z)X(z)$. $A_0(z)$ is gelijk aan de distorsie functie T(z):

$$T(z)=A_0(z)=\frac{1}{M}\cdot\sum_{k=0}^{M-1}H_k(z)F_k(z) \tag{1.20}$$

als |T(z)| geen constante is dan is er sprake van amplitude distorsie en als T(z) een niet-lineaire fase heeft dan spreken we van fase distorsie.

- **Coding/decoding errors**: Dit zijn fouten die optreden bij het compressie en decompressie proces in opslag of transmissie kanaal. In dit verslag gaan we er van uit dat deze fouten niet bestaan.

- **Quantisatie errors**: Omdat we van FIR filters uitgaan ontstaan er fouten t.g.v. coëfficiënten quantisatie en signaal quantisatie. Deze fouten bestaan zijn onder andere uit, overflow, extra aliasing , amplitude en fase distorsie.

De eerste twee fouten kunnen we samen nemen tot de volgende definitie:
Als we de filters $H_k(z)$ en $F_k(z)$ zo nemen dat er geen aliasing optreedt en als T(z) een delay is $T(z)=c\cdot z^{-k}$, $c\neq 0$ waardoor het systeem vrij is van amplitude- en fase distorsie. Er geldt dan dat het gereconstrueerde signaal $\hat{x}[n]$ gelijk is aan een vertraagde versie van het ingangssignaal x[n], eventueel met een andere amplitude: $\hat{x}[n]=c\cdot x[n-k]$ oftewel $\hat{X}(z)=c\cdot z^{-k}X(z)$. We spreken van perfecte reconstructie.

1.4.2 Polyfase Representatie

De reden om een maximaal gedecimeerde M-kanaals filterbank uit te drukken in polyfase componenten is twee ledig. Ten eerste, wat we al eerder opmerkten (paragraaf 1.3), de rekenkundige complexiteit neemt af en ten tweede dat er een belangrijke relatie bestaat tussen de analyse- en synthese filter die gebruikt maakt van de polyfase representatie en de paraunitaire eigenschap.

De analyse- en synthese filters kunnen in een polyfase representatie uitgedrukt worden:

$$H_k(z)=\sum_{l=0}^{M-1}z^{-l}E_{kl}(z^M) \qquad en \qquad F_k(z)=\sum_{l=0}^{M-1}z^{-(M-1-l)}R_{lk}(z^M) \tag{1.21}$$

Dus:

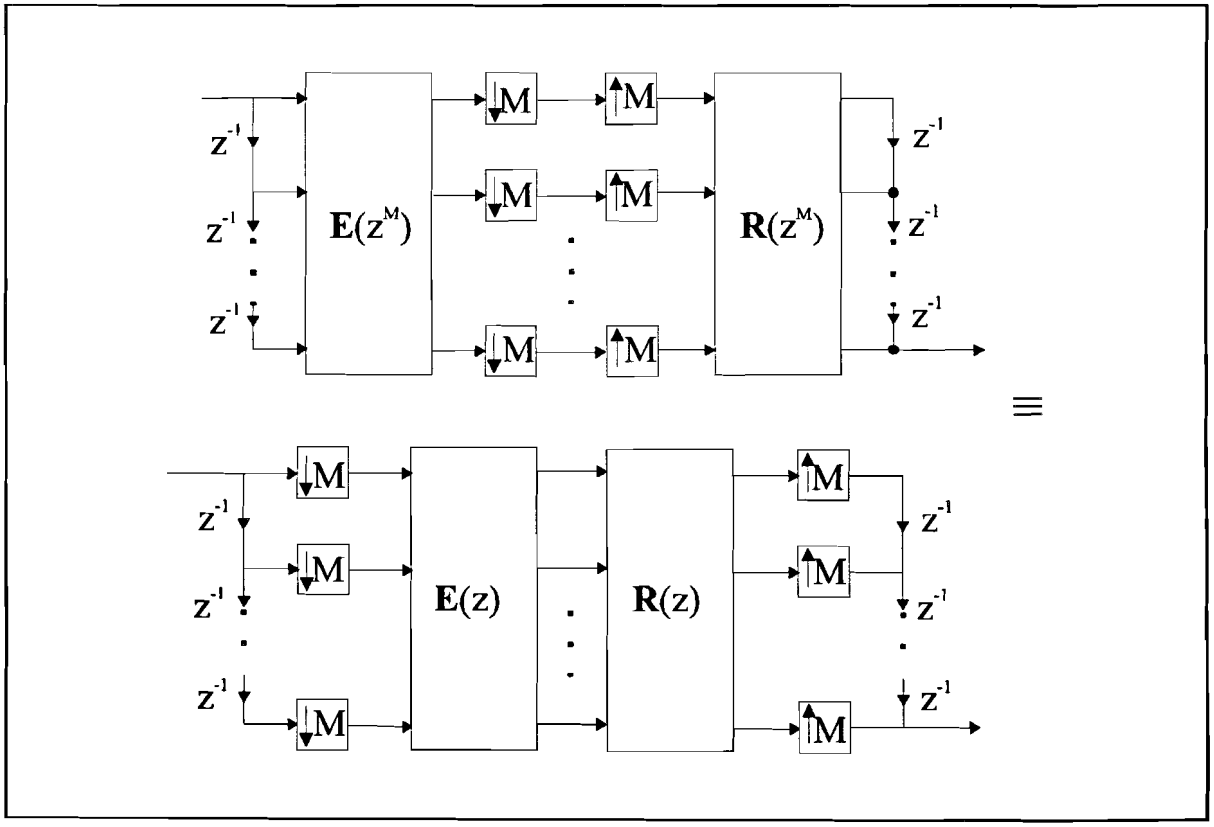
$$\boldsymbol{h}(z)=\boldsymbol{E}(z^M)\boldsymbol{e}(z), \qquad \boldsymbol{f}^T(z)=z^{-(M-1)}\tilde{\boldsymbol{e}}(z)\boldsymbol{R}(z^M) \tag{1.22}$$

met

$$\boldsymbol{E}(z)=\begin{bmatrix} E_{0,0}(z) & E_{0,1}(z) & \cdots & E_{0,M-1}(z) \\ E_{1,0}(z) & E_{1,1}(z) & \cdots & E_{1,M-1}(z) \\ \vdots & \vdots & \ddots & \vdots \\ E_{M-1,0}(z) & E_{M-1,1}(z) & \cdots & E_{M-1,M-1}(z) \end{bmatrix}$$

$$\boldsymbol{R}(z)=\begin{bmatrix} R_{0,0}(z) & R_{0,1}(z) & \cdots & R_{0,M-1}(z) \\ R_{1,0}(z) & R_{1,1}(z) & \cdots & R_{1,M-1}(z) \\ \vdots & \vdots & \ddots & \vdots \\ R_{M-1,0}(z) & R_{M-1,1}(z) & \cdots & R_{M-1,M-1}(z) \end{bmatrix} \tag{1.23}$$

Het schema van de filterbank uit Figuur 1.10 kan herschreven worden naar een equivalente vorm afgebeeld in Figuur 1.11.



Figuur 1.11 De polyfase representatie van een M-kanaals maximaal gedecimeerde filter bank.

Perfecte reconstructie ontstaat als geldt dat:

$$\boldsymbol{R}(z)\boldsymbol{E}(z)=c\cdot z^{-K}\cdot\boldsymbol{I} \tag{1.24}$$

De eenvoudigste manier om dit te verwezenlijken is uit te gaan van de paraunitaire eigenschap. Als we er voor zorgen dat de polyfase matrix **E**(z) paraunitair is dan geldt dat:

$$\tilde{\boldsymbol{E}}(z)\boldsymbol{E}(z)=\alpha\boldsymbol{I} \quad \Rightarrow \quad \boldsymbol{E}^{-1}(z)=\tilde{\boldsymbol{E}}(z)/\alpha, \quad \textit{met} \quad \alpha>0 \tag{1.25}$$

Kiezen we voor **R**(z) nu:

$$\boldsymbol{R}(z)=\frac{c}{\alpha}\cdot z^{-K}\cdot\tilde{\boldsymbol{E}}(z) \tag{1.26}$$

dan wordt de perfecte reconstructie conditie uit Formule (1.24) voldaan. De constante K dient een positief integer te zijn die er voor zorgt dat **R**(z) (en dus $F_k(z)$) causaal is.

Hoofdstuk 2

Cosine Modulated Filter banks

2.1 Inleiding

Het middelpunt in dit verslag zijn filter banken gebaseerd op cosinus modulatie (CM). In deze filter banken worden alle M analyse filters afgeleid van één prototype FIR LDF filter. Een aantal voordelen van CM filter banken zijn:

- De rekenkundige complexiteit is gelijk aan die van één filter plus modulatie kosten.
- De filterbank kan ontworpen worden met perfecte reconstructie.
- Het aantal te optimaliseren parameters is klein omdat uitgegaan wordt van één prototype dat lineaire fase bezit.
- 2M polyfase componenten kunnen in M power complementaire paren gegroepeerd worden waardoor ze te implementeren zijn m.b.v. lattice structuren.
- Wordt gebruik gemaakt van lattice structuren dan blijft de PR condities bestaan met de aanwezigheid van coëfficiënt quantisatie.

In [1] zijn cosinus gemoduleerde pseudo-QMF en perfecte reconstructie filter banken beschreven. In dit verslag zullen we alleen over de perfecte reconstructie systemen spreken. Sinds 1990 zijn deze cosinus gemoduleerde filter banken met perfecte reconstructie ontworpen. Daarover zijn enkele publikaties verschenen waarvan ik die van Koilpillai en Vaidyanathan [4] en [5] heb ik gebruikt bij de bestudering, ontwerp en simulaties van deze filter banken. Andere werken die overigens niets toevoegen aan eerder genoemde publikaties zijn: [6], [7], [8], [9], [10] en [11].

2.2 Van prototype naar reële coëfficiënt analyse filters

Een M-kanaals filter bank kan verkregen worden door een verzameling van M analyse filters $H_k(z)$ te relateren aan één prototype filter $H_0(z)$, d.m.v. $H_k(z)=H_0(zW^k)$ met $W=e^{-j2\pi/M}$. Dit betekent dat de frequentie responsies $H_k(e^{j\omega})$ uniforme verschoven versies van het prototype zijn. Als het prototype uitgedrukt wordt in polyfase componenten dan geldt:

$$H_0(z)=\sum_{l=0}^{M-1} z^{-l}E_l(z^M) \quad \rightarrow \quad H_k(z)=H_0(zW^k)=\sum_{l=0}^{M-1}(z^{-1}W^{-k})^l E_l(z^M) \tag{2.1}$$

De coëfficiënten $h_k[n]$ zijn in het algemeen complex zelfs als $h_0[n]$ reëel zijn, dit komt door de exponentiële modulatie.

Door cosinus modulatie toe te passen resulteert dit i.p.v. complexe coëfficiënt- in reëel coëfficiënt analyse filters. Dit wordt gedaan door 2M complexe filters af te leiden m.b.v.

exponentiële modulatie en deze te combineren tot paren. Het uitgangspunt is het prototype filter $P_0(z)$, dat een laag doorlaat karakter heeft met bandbreedte $\pi/2M$. De polyfase representatie van $P_0(z)$ is gelijk aan:

$$P_0(z)=\sum_{l=0}^{2M-1} z^{-l}G_l(z^{2M}) \tag{2.2}$$

$P_0(z)$ is FIR en heeft een symmetrische impuls response:

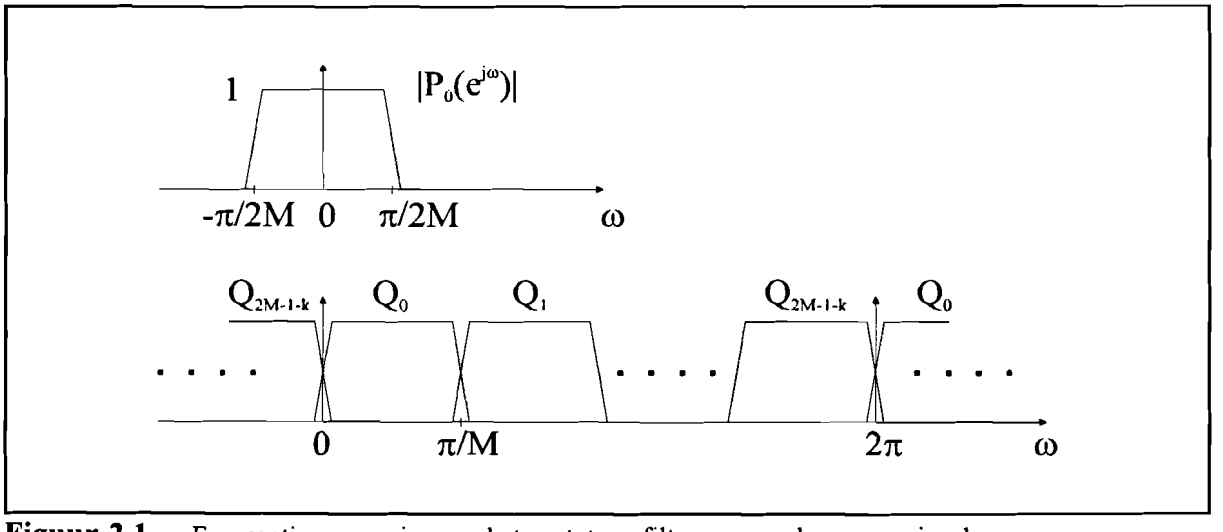
$$P_0(z)=z^{-(N-1)}\tilde{P}_0(z) \tag{2.3}$$

De complexe filter $Q_k(z)$ worden gegeven in termen van het prototype $P_0(z)$ door:

$$Q_k(z)=P_0(zW^{k+1/2}), \qquad W=e^{\frac{-j\pi}{M}}, \qquad 0\le k\le 2M-1 \tag{2.4}$$

Figuur 2.1 geeft de frequentie responsies van het prototype $P_0(z)$ en van de complexe filters $Q_k(z)$. De magnitude responsies van $Q_k(z)$ en $Q_{2M-1-k}(z)$ zijn gespiegelde versies van elkaar t.o.v. $\omega=0$, waardoor de impuls responsies van $Q_k(z)$ en $Q_{2M-1-k}(z)$ geconjugeerde van elkaar zijn, dus:

$$Q_{2M-1-k}(z)=Q_k^*(z), \qquad 0\le k\le M-1 \tag{2.5}$$



Figuur 2.1 Frequentie responsies van het prototype filter en verschoven versies daarvan.

In [1] wordt een afleiding gegeven hoe m.b.v. de complexe filter $Q_k(z)$ de analyse filter $H_k(z)$ samen gesteld kunnen worden. Deze afleiding slaan we in dit verslag over. Het resultaat is samen te vatten in onderstaande formule:

$$H_k(z)=d_kQ_k(z)+d_k^*Q_{2M-1-k}(z), \qquad 0\le k\le M-1 \tag{2.6}$$

met

$$d_k = e^{j\theta_k} W^{(k+\frac{1}{2})(N/2-\frac{1}{2})}, \qquad \theta_k = (-1)^k \frac{\pi}{4}, \qquad 0 \le k \le M-1 \tag{2.7}$$

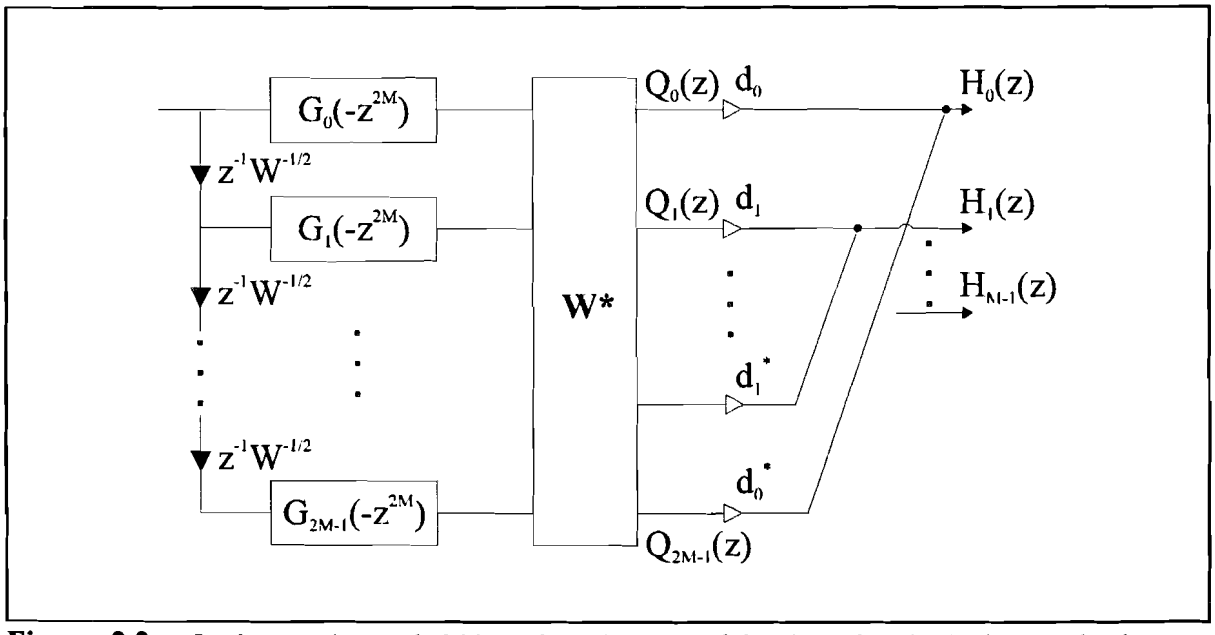
Om verder gebruik te maken van de polyfase representatie combineren we formule (2.2) en (2.4):

$$Q_k(z) = \sum_{l=0}^{2M-1} W^{-kl} (z^{-1} W^{-\frac{1}{2}})^l G_l(-z^{2M}), \qquad 0 \le k \le 2M-1 \tag{2.8}$$

Formule (2.8) samen met (2.6) geeft het volgende:

$$H_k(z) = \sum_{l=0}^{2M-1} \left(d_k W^{-kl} W^{-\frac{1}{2}l} + d_k^* W^{-(2M-1-k)l} W^{-\frac{1}{2}l} \right) z^{-l} G_l(-z^{2M}), \qquad 0 \le k \le M-1 \tag{2.9}$$

M.b.v Formule (2.9) kunnen de analyse filters geïmplementeerd worden volgens Figuur 2.2.



Figuur 2.2 Implementatie van de M-kanaals cosinus gemoduleerde analyse bank uitgevoerd m.b.v polyfase componenten en een IDFT.

Formule (2.9) is equivalent met :

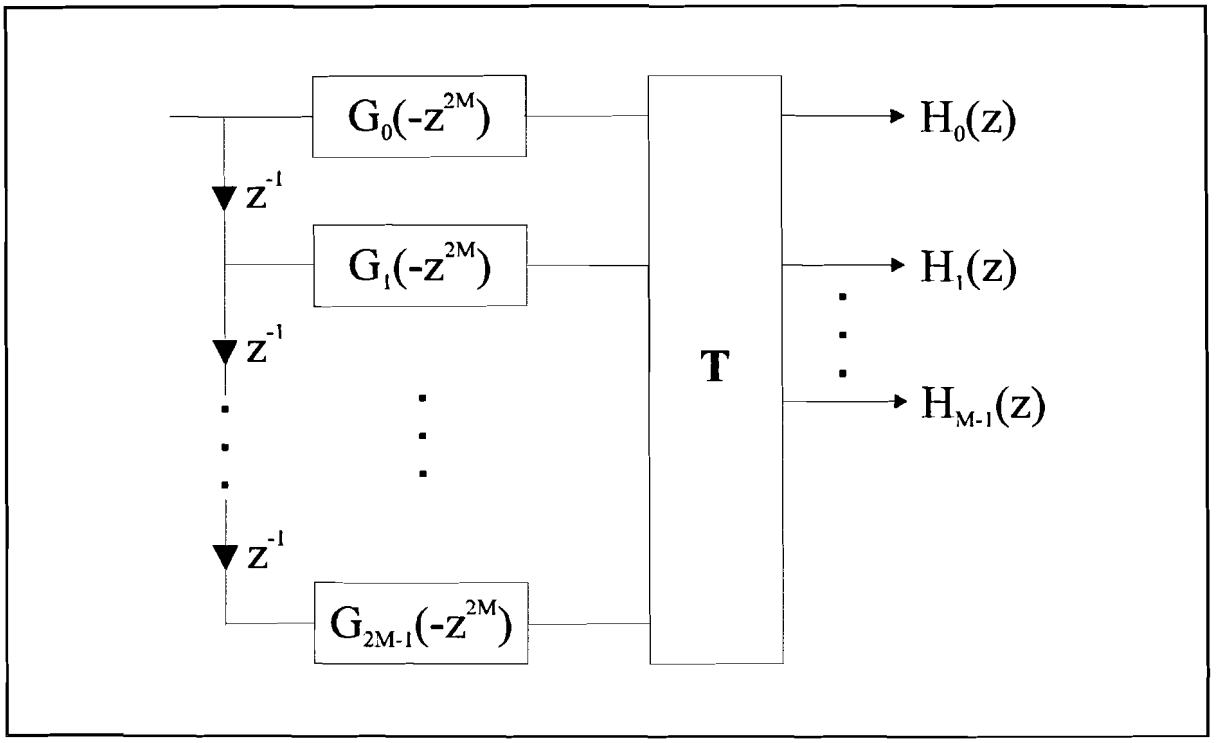
$$H_k(z) = \sum_{l=0}^{2M-1} \left(W^{-(k+\frac{1}{2})(l-N/2+\frac{1}{2})} e^{j\theta_k} + W^{(k+\frac{1}{2})(l-N/2+\frac{1}{2})} e^{-j\theta_k} \right) z^{-l} G_l(-z^{2M}), \quad 0 \le k \le M-1 \tag{2.10}$$

Door gebruik te gaan maken van de zogenaamde M×2M cosinus modulatie matrix **T**, kunnen de analyse filters uitgedrukt worden in de polyfase componenten en cosinus modulatie matrix **T**.

$$H_k(z) = \sum_{l=0}^{2M-1} t_{kl} z^{-l} G_l(-z^{2M})$$

$$t_{kl} = 2\cos\left(\frac{\pi}{M}(k+\tfrac{1}{2})(l-N/2+\tfrac{1}{2}) + \theta_k\right), \quad 0 \le k \le M-1 \tag{2.11}$$

Figuur 2.3 geeft een implementatie van de analyse bank volgens Formule (2.11).



Figuur 2.3 Implementatie van de M-kanaals cosinus gemoduleerde analyse bank uitgevoerd m.b.v polyfase componenten en een cosinus modulatie matrix **T**.

De methode uit hoofdstuk 1 om een PR systeem te ontwerpen was uit te gaan van een paraunitaire polyfase matrix **E**(z) en daar de polyfase matrix **R**(z) van de synthese bank op aan te passen. We kennen tot nu toe wel de polyfase componenten van het prototype $P_0(z)$ wel maar nog niet de polyfase matrix **E**(z). In paragraaf 2.3 wordt eerst een uitdrukking voor de polyfase matrix **E**(z) gevonden die in paragraaf 2.4 gebruikt wordt om condities te vinden waardoor hij paraunitair is. Daarbij wordt uitgegaan van de prototype filter lengte N=2mM, waarbij M het aantal kanalen en m een willekeurig positief integer voorstelt. Deze restrictie is nodig om de lengten van de impuls responsies van de polyfase componenten van $P_0(z)$ gelijk te laten zijn.

2.3 CM filter bank uitgedrukt in een polyfase structuur

Om tot een uitdrukking voor de polyfase matrix **R**(z) te komen dient er eerst een uitdrukking voor de polyfase matrix **E**(z) gevonden te worden. Een eerste aanzet wordt gemaakt

door Formule (2.11) in matrix notatie te schrijven:

$$\boldsymbol{h}(z)=\boldsymbol{T}\boldsymbol{g}(z) \tag{2.12}$$

met

$$\boldsymbol{g}(z)=\begin{bmatrix} G_0(-z^{2M}) \\ z^{-1}G_1(-z^{2M}) \\ \vdots \\ z^{-(2M-1)}G_{2M-1}(-z^{2M}) \end{bmatrix}=\begin{bmatrix} \boldsymbol{g}_0(z^{2M}) \\ z^{-M}\boldsymbol{g}_1(z^{2M}) \end{bmatrix}\cdot\boldsymbol{e}(z) \tag{2.13}$$

waar **e**(z) een delay vector (zie Formule (1.17)) is en $\mathbf{g}_i$(z) diagonaal matrices zijn:

$$\begin{aligned} \boldsymbol{g}_0(z) &= diag\left(G_0(-z), G_1(-z), \cdots, G_{M-1}(-z)\right) \\ \boldsymbol{g}_1(z) &= diag\left(G_M(-z), G_{M+1}(-z), \cdots, G_{2M-1}(-z)\right) \end{aligned} \tag{2.14}$$

De polyfase matrix **E**(z) van de analyse bank is gelijk aan:

$$\boldsymbol{h}(z)=\boldsymbol{E}(z^M)\boldsymbol{e}(z)\ , \qquad \boldsymbol{E}(z)=\boldsymbol{T}\begin{bmatrix} \boldsymbol{g}_0(z^2) \\ z^{-1}\boldsymbol{g}_1(z^2) \end{bmatrix} \tag{2.15}$$

Een polyfase representatie van de synthese bank (synthese filters) is gelijk aan:

$$\boldsymbol{f}^T(z)=z^{-(M-1)}\tilde{\boldsymbol{e}}(z)\boldsymbol{R}(z^M)=\boldsymbol{e}(z)\boldsymbol{R}(z^M) \tag{2.16}$$

Voor de keuze van polyfase matrix **R**(z) nemen we:

$$\boldsymbol{R}(z)=z^{-(2m-1)}\tilde{\boldsymbol{E}}(z) \tag{2.17}$$

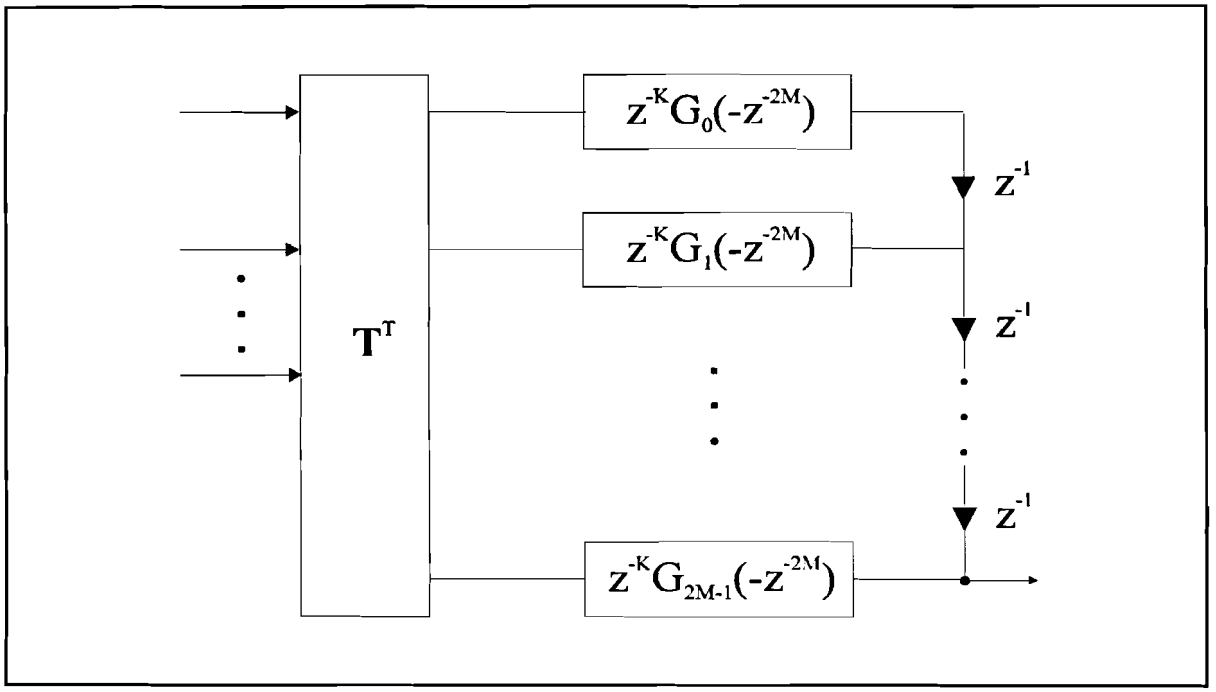
net als in hoofdstuk 1. De 2M polyfase componenten van P_0(z) hebben elk een impuls response ter lengte m, omdat de filter lengte van P_0(z) gelijk is aan N=2mM. Het extra delay om de polyfase matrix **R**(z) causaal te maken moet dus minimaal gelijk zijn aan 2m-1 tijdseenheden.
De synthese filter kunnen nu ook uitgedrukt worden in de polyfase matrix **E**(z):

$$\boldsymbol{f}^T(z)=z^{-(2mM-1)}\tilde{\boldsymbol{e}}(z)\tilde{\boldsymbol{E}}(z^M)=z^{-(N-1)}\boldsymbol{h}^T(z^{-1})=z^{-(N-1)}\boldsymbol{g}^T(z^{-1})\boldsymbol{T}^T \tag{2.18}$$

en in de analyse filters.

$$F_k(z)=z^{-(N-1)}H_k(z^{-1}), \qquad 0\le k\le M-1 \tag{2.19}$$

Figuur 2.4 geeft de implementatie van de synthese bank volgens Formule (2.18). Met k=2M(m-1).



Figuur 2.4 Implementatie van de M-kanaals cosinus gemoduleerde synthese bank uitgevoerd m.b.v polyfase componenten en een cosinus modulatie matrix **T**.

2.4 Paraunitairiteit van polyfase matrix E(z)

Om ervoor te zorgen dat de cosinus gemoduleerde filterbank de perfecte reconstructie eigenschap bezit is het nodig dat de polyfase matrix **E**(z) paraunitair is. De vragen wat zijn de restricties op **E**(z) zodat **E**(z) paraunitair is en wat zijn de gevolgen voor de polyfase componenten van $P_0(z)$ worden in deze paragraaf beantwoord. De eerste stap die gezet wordt is het partitioneren van de cosinus modulatie matrix **T** d.m.v.:

$$\boldsymbol{T}=[\boldsymbol{A}_0 \ \boldsymbol{A}_1] \tag{2.20}$$

Hieronder enkele eigenschappen van de partities die nodig zijn voor de beantwoording van de eerder gestelde vraag. Voor een bewijs van deze expressies wordt verwezen naar hoofdstuk 4.

$$\begin{aligned} A_0^T A_0 &= 2M(I-(-1)^m J) \\ A_1^T A_1 &= 2M(I+(-1)^m J) \\ A_1^T A_0 &= A_0^T A_1 = 0 \end{aligned} \tag{2.21}$$

De paraconjugatie van **E**(z) is gelijk aan:

$$\tilde{E}(z) = E_*^T(z^{-1}) = \begin{bmatrix} \tilde{g}_0(z^2) & z\tilde{g}_1(z^2) \end{bmatrix} \begin{bmatrix} A_0^T \\ A_1^T \end{bmatrix} \tag{2.22}$$

waardoor

$$\begin{aligned} \tilde{E}(z)E(z) = {} & 2M\left(\tilde{g}_0(z^2)g_0(z^2) + \tilde{g}_1(z^2)g_1(z^2)\right) \\ & -2M(-1)^m\left(\tilde{g}_0(z^2)Jg_0(z^2) - \tilde{g}_1(z^2)Jg_1(z^2)\right) \end{aligned} \tag{2.23}$$

De relatie $p_0[n]=p_0[N-1-n]$ (equivalent met formule (2.3)) geeft ook een restrictie op de polyfase componenten:

$$G_k(z) = z^{-(m-1)}\tilde{G}_{2M-1-k}(z), \qquad 0 \le k \le M-1 \tag{2.24}$$

Voor de diagonaal matrices $\mathbf{g}_i(z)$ houdt dit het volgende in:

$$\begin{aligned} g_1(z) &= z^{-(m-1)}(-1)^{m-1}J\tilde{g}_0(z)J \\ &\textit{en} \\ g_0(z) &= z^{-(m-1)}(-1)^{m-1}J\tilde{g}_1(z)J \end{aligned} \tag{2.25}$$

M.b.v. de twee formules uit (2.25) kan de volgende relatie opgesteld worden:

$$\tilde{g}_0(z)Jg_0(z) = z^{-(m-1)}(-1)^{m-1}\tilde{g}_0(z)\tilde{g}_1(z)J = \tilde{g}_1(z)Jg_1(z) \tag{2.26}$$

Substitueren we (2.26) in (2.23) dan ontstaat:

$$\tilde{E}(z)E(z) = 2M\left(\tilde{g}_0(z^2)g_0(z^2) + \tilde{g}_1(z^2)g_1(z^2)\right) \tag{2.27}$$

Het is duidelijk dat **E**(z) paraunitair is als het rechterlid van Formule (2.27) gelijk aan $\alpha\mathbf{I}$ is, wat equivalent is met: (α=1)

$$\tilde{G}_k(z)G_k(z)+\tilde{G}_{M+k}(z)G_{M+k}(z)=\frac{1}{2M} \qquad 0\le k\le M-1 \qquad \textbf{(2.28)}$$

De conditie uit Formule (2.28) komt overeen met de power complementaire eigenschap van de polyfase componenten $G_k(z)$ en $G_{M+k}(z)$.

Hoofdstuk 3

Implementatie Polyfase componenten $G_k(z)$

3.1 Inleiding

In dit hoofdstuk komt de implementatie van de polyfase componenten ter sprake. Bij deze implementatie kan gebruik gemaakt worden van twee verschillende structuren, de lattice- en transversale structuur. Het voordeel van de lattice structuur is dat deze structuur automatisch tot power complementaire functies leidt ongeacht de keuze van de vrijheidsparameters in zo'n structuur. De voor- en nadelen van beide structuren komen in volgende hoofdstukken aan de orde.

Om te beginnen vatten we het resultaat van vorig hoofdstuk samen:
Het prototype $P_0(z)$ is een reëel coëfficiënt symmetrisch FIR filter ter lengte N=2mM. Het prototype bestaat uit de polyfase componenten $G_k(z)$, $0 \le k \le 2M-1$. De analyse bank met polyfase matrix $\mathbf{E}(z)$, volgens Formule (2.15) is paraunitair als de volgende power complementaire geldt:

$$\tilde{G}_k(z)G_k(z)+\tilde{G}_{M+k}(z)G_{M+k}(z)=\frac{1}{2M} \qquad 0 \le k \le M-1 \tag{3.1}$$

In paragraaf 3.2 wordt afgeleid hoe de M vergelijkingen uit Formule (3.1) gereduceerd kunnen worden, zie [1], [4] en [5]. Paragraaf 3.3 geeft de implementatie via een lattice structuur, zie [1], [4] en [5]. En paragraaf 3.4 de implementatie via een transversale structuur, zie [12].

3.2 Power complementaire paren

Formule (3.1) bestaat in feite uit M relaties die gereduceerd kunnen worden afhankelijk van M (even/oneven), omdat het prototype filter symmetrisch is (Formule 2.24).

- M even:

$$\tilde{G}_k(z)G_k(z)+\tilde{G}_{M+k}(z)G_{M+k}(z)=\frac{1}{2M} \qquad 0 \le k \le \frac{M}{2}-1 \tag{3.2}$$

- M oneven:

$$\tilde{G}_k(z)G_k(z)+\tilde{G}_{M+k}(z)G_{M+k}(z)=\frac{1}{2M} \qquad 0 \le k \le \frac{M-1}{2}-1 \tag{3.3}$$

en

$$\tilde{G}_{\frac{M-1}{2}}(z)G_{\frac{M-1}{2}}(z)=\frac{1}{4M} \tag{3.4}$$

Vanwege de symmetrie geldt:

$$G_{\frac{M-1}{2}}(z)=z^{-(m-1)}\tilde{G}_{\frac{3M-1}{2}}(z) \tag{3.5}$$

Uit Formule (3.4) is duidelijk op te maken dat de polyfase component $G_{(M-1)/2}(z)$ en de daarmee samen hangende $G_{(3M-1)/2}(z)$ uit een delay moet bestaan. Dus:

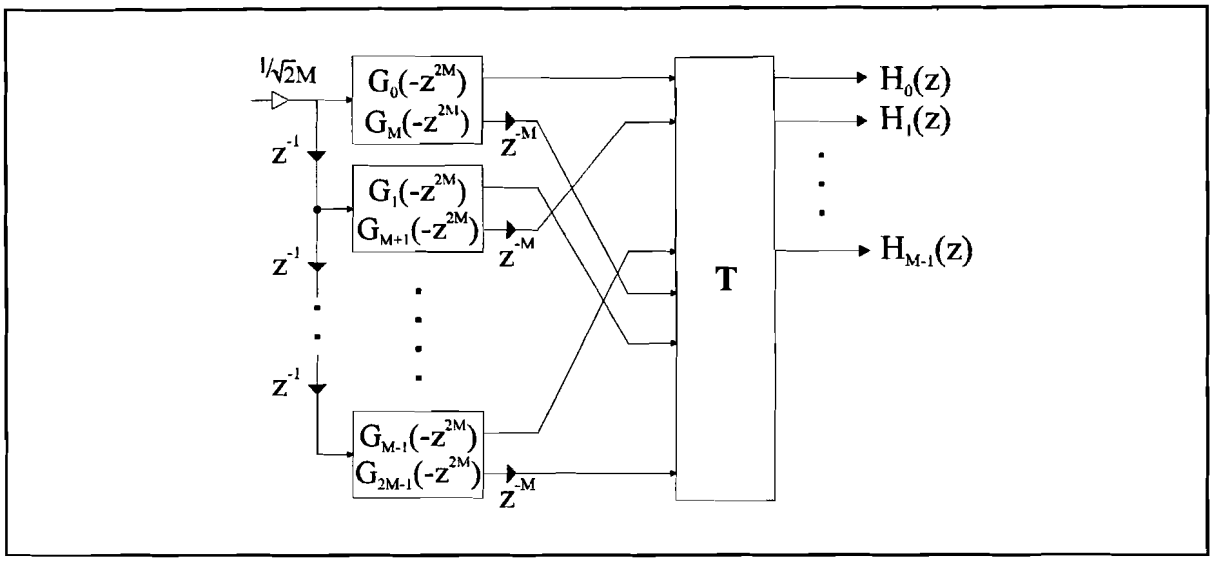
$$G_{\frac{M-1}{2}}(z)=\frac{1}{2\sqrt{M}}z^{-K}, \qquad G_{\frac{3M-1}{2}}(z)=\frac{1}{2\sqrt{M}}z^{-(m-1-K)} \tag{3.6}$$

met

$$K=\begin{cases}\dfrac{m-1}{2} & m \text{ is oneven}\\[2ex] \dfrac{m}{2} & m \text{ is even}\end{cases} \tag{3.7}$$

3.3 Implementatie via lattice structuren

Formule (3.1) geeft een conditie die gelijk is aan te zeggen dat de polyfase componenten $G_k(z)$ en $G_{M+k}(z)$ power complementair zijn. In paragraaf 1.3 hadden we opgemerkt dat als twee functies power complementair met elkaar zijn, deze te implementeren zijn m.b.v. een twee kanaals lattice. Dus de 2M polyfase componenten uit de analyse bank zijn te implementeren m.b.v. M twee-kanaals lattices. Alle vrijheidsgraden in zo'n structuur kunnen gebruikt worden om te zorgen dat dit leidt tot *goede* analyse filters, zonder dat hiermee de perfecte reconstructie in gevaar komt. Figuur 3.1 geeft de implementatie van de analyse bank en Figuur 3.2.a van de twee kanaals lattice structuur. In paragraaf 1.3 is al opgemerkt dat er verschillende mogelijkheden zijn voor keuze van de coëfficiënten in een lattice structuur. Deze coëfficiënten moeten we niet verwarren met de lattice coëfficiënten $\theta_{k,i}$, want een coëfficiënt is een sinus of cosinus van een lattice coëfficiënt of een deling daarvan.

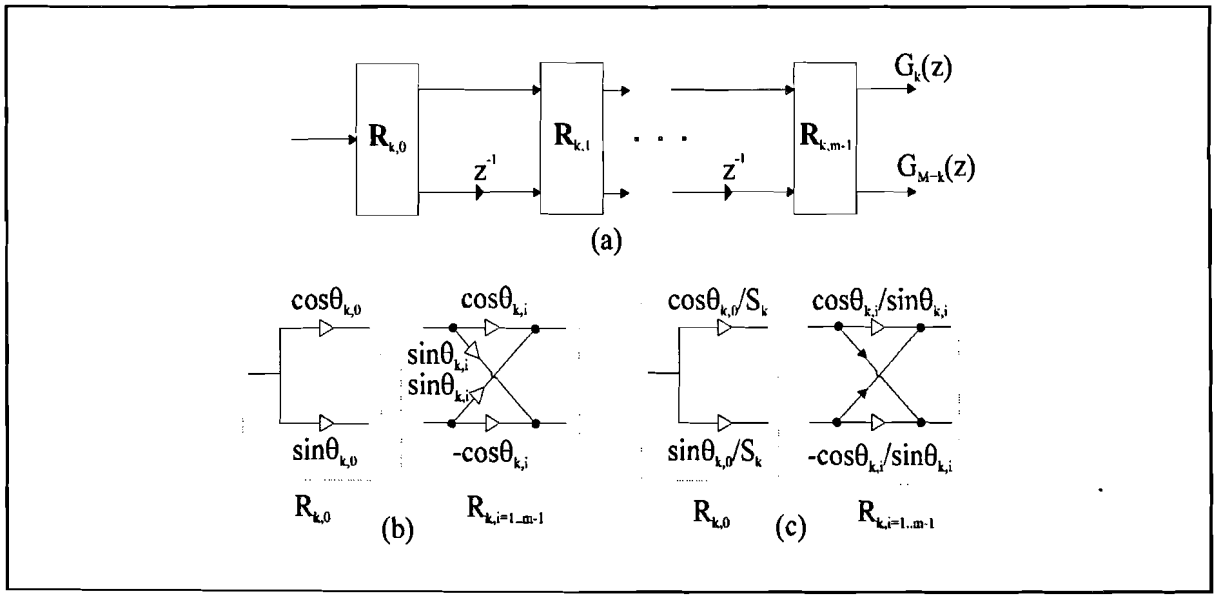


Figuur 3.1 Implementatie van de M-kanaals cosinus gemoduleerde analyse bank uitgevoerd m.b.v. twee kanaals lattice en de cosinus modulatie matrix **T**.

De twee kanaals lattice is uit te voeren met vier of twee vermenigvuldigers (respectievelijk afgebeeld in Figuur 3.2.b en 3.2.c) per elementaire matrix $\mathbf{R}_{k,i}$. De waarde S_k is gelijk aan:

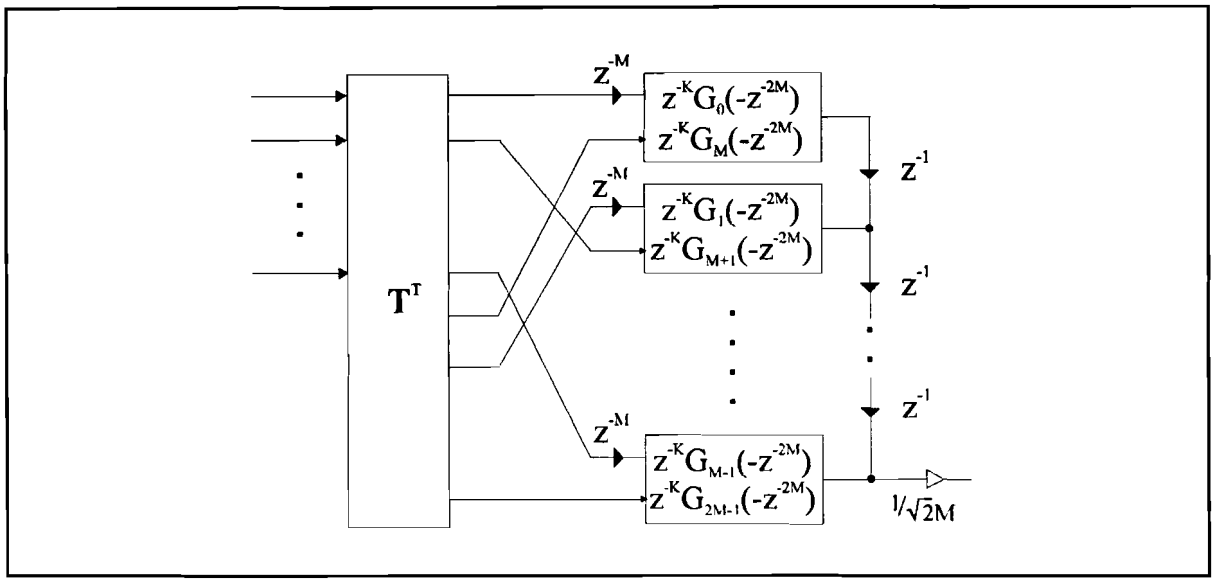
$$S_k = \prod_{i=1}^{m-1} \sin\theta_{k,i} \tag{3.8}$$

De lattice noemen we daarom ook een vier- of twee vermenigvuldigers lattice.

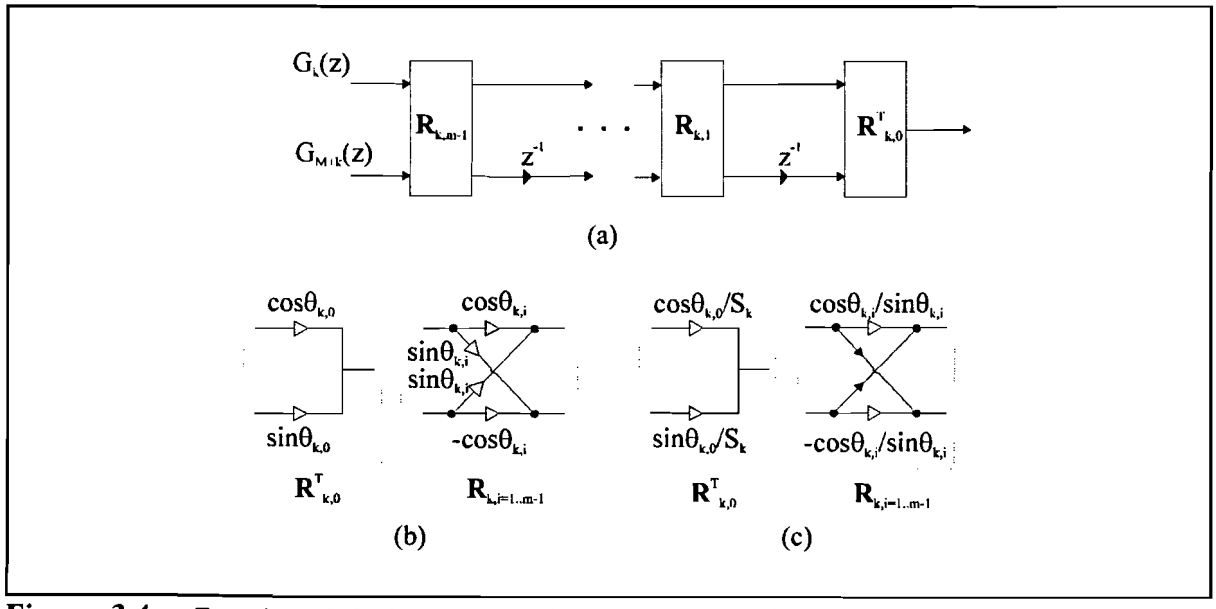


Figuur 3.2 Twee kanaals lattice structuur uitgevoerd met een vier- en twee vermenigvuldigers lattice.

In de synthese bank komen de paraconjugaties van de polyfase componenten voor; zie Formule (2.18). In paragraaf 1.3 is een implementatie gegeven van de paraconjugatie van de lattice structuur, die een power complementair paar implementeerde. Dus de tegenhanger van de analyse bank uit Figuur 3.1 is de synthese bank uit Figuur 3.3, met K=2M(m-1). Deze vertraging maakt de structuur causaal. De implementatie van de lattice structuur is afgebeeld in Figuur 3.4 en kan gebruikt worden in de synthese bank.



Figuur 3.3 Implementatie van de M-kanaals cosinus gemoduleerde synthese bank uitgevoerd m.b.v. twee kanaals lattice en de cosinus modulatie matrix **T**.

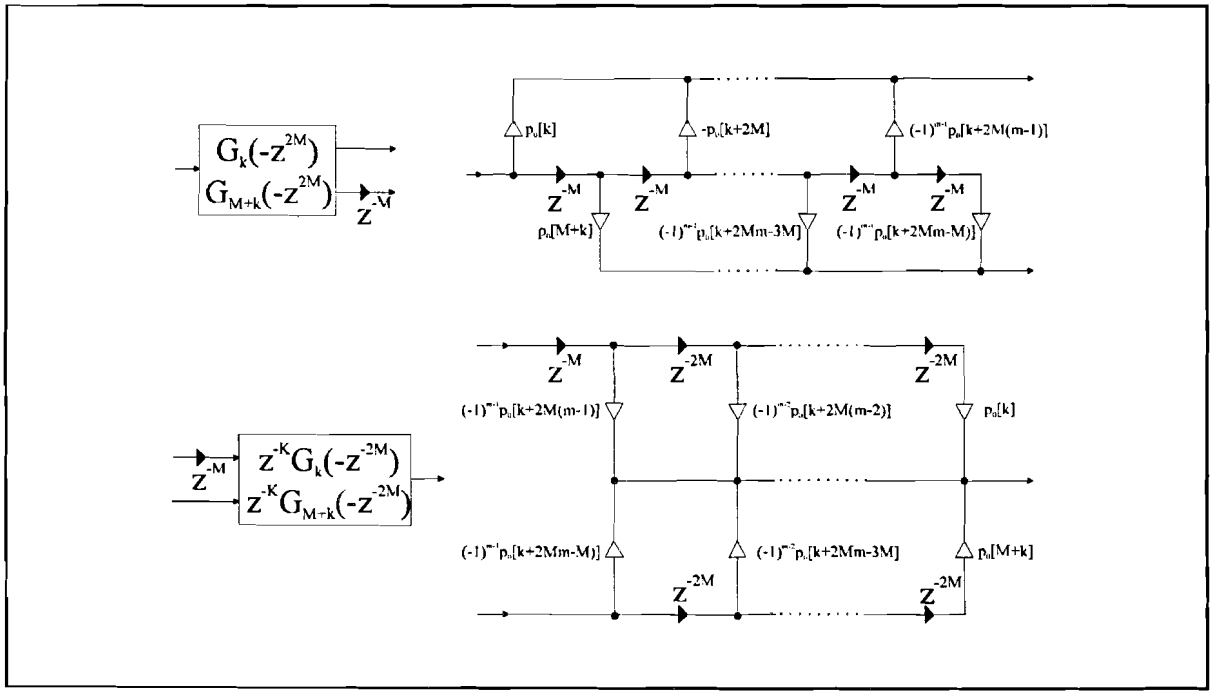


Figuur 3.4 Twee kanaals lattice structuur uitgevoerd met een vier- en twee vermenigvuldigers lattice.

De vier- en twee vermenigvuldigers lattices hebben een gelijk aantal vrijheidsparameters waardoor een ontwerp procedure voor beide structuren gelijk is. De vier vermenigvuldigers lattice heeft in een implementatie ongeveer twee keer zoveel vermenigvuldigingen en optellingen nodig dan de twee-vermenigvuldigers lattice.

3.4 Implementatie via transversale structuren

In plaats van de twee kanaals lattice structuren kan ook een transversale structuur gebruikt worden. Een transversale structuur biedt niet automatisch een power complementair paar van filters, waardoor de structuur niet vanzelf tot perfecte reconstructie leidt. Het aantal vrijheidsgraden in een lattice structuur is gelijk aan m en in de transversale structuur 2m. Er moeten dus restricties op de coëfficiënten gelegd worden, zodanig dat dit leidt tot een perfecte reconstructie filter bank. In Figuur 3.5 is een transversale structuur afgebeeld die *equivalent* is met de lattice structuur uit Figuur 3.2 en 3.4 indien de coëfficiënten m.b.v de lattice coëfficiënten gekozen zijn. Wat de voor- en nadelen van de lattice- en transversale structuren zijn, komt in volgende hoofdstukken ter sprake.



Figuur 3.5 Implementatie van de polyfase componenten m.b.v. een transversale structuur.

Hoofdstuk 4

Implementatie Cosinus Modulatie Matrix

4.1 Inleiding

In paragraaf 4.2 geven we een afleiding dat de implementatie van de cosinus modulatie matrix in feite bestaat uit de implementatie van een DCT van het type IV. In [1] en [5] is deze afleiding ook gegeven. In paragraaf 4.3 worden een aantal mogelijke implementaties gegeven, die afkomstig zijn uit [13], [14], [15] en [16]. In paragraaf 4.4 zijn de MATLAB functies gegeven die deze verschillende implementatie methoden simuleren.

4.2 Cosinus modulatie matrix uitgedrukt m.b.v een DCT

In paragraaf 2.4 hadden we opgemerkt dat de cosinus modulatie matrix gepartitioneerd kan worden in twee M×M matrices $\mathbf{A}_0$ en $\mathbf{A}_1$:

$$T=[A_0 \;\; A_1] \tag{4.1}$$

Deze twee matrices $\mathbf{A}_0$ en $\mathbf{A}_1$ kunnen geschreven als :

$$\begin{aligned} A_0 &= \sqrt{M}(-1)^{m_1} C(I-J) && m \text{ even} \\ A_1 &= \sqrt{M}(-1)^{m_1+1} C(I+J) && m=2m_1 \\ A_0 &= \sqrt{M}(-1)^{m_1} C(I+J) && m \text{ oneven} \\ A_1 &= \sqrt{M}(-1)^{m_1} C(I-J) && m=2m_1+1 \end{aligned} \tag{4.2}$$

waarbij C de Type IV Discrete Cosine Transform (DCT-IV) matrix voorstelt, die gedefinieerd is als:

$$[C]_{ij}=\cos\left(\frac{\pi}{M}(i+\tfrac{1}{2})(j+\tfrac{1}{2})\right) \tag{4.3}$$

De twee belangrijkste eigenschappen van **C** zijn:

$$C^T=C, \quad C^TC=I, \tag{4.4}$$

Gebruik makend van Formule (4.4) en de eigenschappen en rekenregels uit paragraaf 1.2 geeft dat de getransponeerde van de matrices $\mathbf{A}_0$ en $\mathbf{A}_1$ gelijk zijn aan:

$$\begin{aligned} A_0^T &= (I-J)C\sqrt{M}(-1)^{m_1} && m \text{ even} \\ A_1^T &= (I+J)C\sqrt{M}(-1)^{m_1+1} && m=2m_1 \\ A_0^T &= (I+J)C\sqrt{M}(-1)^{m_1} && m \text{ oneven} \\ A_1^T &= (I-J)C\sqrt{M}(-1)^{m_1} && m=2m_1+1 \end{aligned} \tag{4.5}$$

M.b.v. bovenstaande formule kunnen de volgende expressies zonder bewijs afgeleid worden:

$$\begin{aligned} A_0^T A_0 &= 2M(I-(-1)^m J) \\ A_1^T A_1 &= 2M(I+(-1)^m J) \\ A_1^T A_0 &= A_0^T A_1 = 0 \end{aligned} \tag{4.6}$$

De expressies uit Formule (4.6) zijn gebruikt in paragraaf 2.4.

4.3 Implementatie van de DCT-IV

Er bestaan vier Types, waarvan we alleen Type II en IV definiëren. De reële elementen van deze M×M matrices zijn gelijk aan:

$$\begin{bmatrix} \textit{Type II}: \sqrt{\frac{2}{M}} c_n C_M^{n(k+1/2)} \\ \textit{Type IV}: \sqrt{\frac{2}{M}} C_M^{(n+1/2)(k+1/2)} \end{bmatrix} \quad \textit{met} \quad C_j^i = \cos\left(\frac{\pi i}{j}\right), \quad c_i = \begin{cases} 1 & i \neq 0 \\ 1/\sqrt{2} & i = 0 \end{cases} \quad 0 \le n,m \le M-1 \tag{4.7}$$

We zullen verder niet meer spreken over de Type IV DCT maar over de DCT waarbij het Type IV weglaten. We hebben het over de Type II DCT als we het er uitdrukkelijk bij zeggen.

De implementatie van de Cosinus Modulatie matrix **T** bestaat in feite uit het implementeren van de DCT matrix **C**. Dit kan op vele manieren gebeuren. We zullen 4 methoden geven waarvan de eerste 3 gebruik maken een DFT (of een FFT als $M=2^i$, i positief integer), die in de praktijk in allerlei programmatuur aanwezig is, wat van de DCT niet gezegd kan worden. De DCT kan natuurlijk ook rechtstreeks geïmplementeerd worden, wat rekenkundig niet erg efficiënt is.

Voordat we overgaan tot de uiteenzetting van de diverse methoden sommen we eerst een aantal rekenregels op die daarbij gebruikt worden.

De DCT-IV van een vector $\underline{x}$, **X** is gedefinieerd als:

$$X[k] = \sqrt{\frac{2}{M}} \sum_{n=0}^{M-1} x[n]\, C_M^{(k+½)(n+½)}, \qquad 0 \le k \le M-1 \tag{4.8}$$

met

$$C_M^{(k+½)(n+½)} = \cos\left(\frac{\pi}{M}(k+½)(n+½)\right) = Re\left\{W_{2M}^{(k+½)(n+½)}\right\} \tag{4.9}$$

Elke cosinus kan in complexe termen uitgeschreven worden:

$$\cos\left(\frac{\pi}{M}(k+½)(n+½)\right) = W_{2M}^{(k+½)(n+½)} + W_{2M}^{-(k+½)(n+½)} \tag{4.10}$$

En via de definitie uit paragraaf 1.2 geldt:

$$W_M^{½k} = W_{2M}^{k} = e^{\frac{-j\pi k}{M}} \tag{4.11}$$

Met **W**(k,m,c) duiden we een vector ter lengte m aan met de volgende elementen:

$$W(k,m,c) = W_k^{l+c}, \qquad 0 \le l \le m-1 \tag{4.12}$$

Methode 1:

De reële ingangsvector **x** (ter lengte M) verlengen we met M nullen. En dan voeren we achtereen volgens de volgende bewerkingen uit:

- Vermenigvuldig de verlengde vector met **W**(4M,2M,½).
- Berekenen van de 2M-punts DFT.
- Vermenigvuldig eerste M uitgangen van de DFT met **W**(4M,M,0)
- Neem reële delen daarvan en vermenigvuldig met √(2/M)

Oftewel:

$$X[k] = \sqrt{\frac{2}{M}}\, Re\left\{W_{4M}^{k} \sum_{n=0}^{2M-1} W_{2M}^{kn} x[n]\, W_{4M}^{n+½}\right\}, \qquad 0 \le k \le M-1 \tag{4.13}$$

Nadeel: Reële deel operator, 2M-punts DFT, imaginaire operaties
Voordeel: Simpele implementatie

Methode 2:

De bewerkingen zijn achtereenvolgens:

- De reële ingangsvector **x** (ter lengte M) vormen we om naar een nieuwe vector $\underline{x}$ ter lengte 2M volgens:

$$\underline{x}[n]=\begin{cases} x[n]\cdot e^{-j\pi(n-\frac{1}{2})/2M} & n=0\cdots M-1 \\ x[2M-n+1]\cdot e^{j\pi(2M-n-\frac{1}{2})/2M} & n=M\cdots 2M-1 \end{cases} \tag{4.14}$$

- Berekenen van $\underline{x}$ een 2M-DFT.
- Vermenigvuldig de eerste M uitgangen van de DFT met $\sqrt{(2/M)}\cdot\frac{1}{2}\mathbf{W}(4M,M,0)$

Oftewel:

$$X[k]=\sqrt{\frac{2}{M}}\frac{1}{2}W_{4M}^{k}\left(\sum_{n=0}^{M-1}W_{2M}^{kn}x[n]\,W_{4M}^{n+\frac{1}{2}}+\sum_{n=M}^{2M-1}W_{2M}^{kn}x[2M-n-1]\,W_{4M}^{-(2M-n-\frac{1}{2})}\right) \tag{4.15}$$

$$0\le k\le M-1$$

Nadeel: de 2M-punts DFT, imaginaire operaties
Voordeel: Geen Reële deel operator, Simpele implementatie

Methode 3:

De bewerkingen zijn achtereenvolgens:

- De reële ingangsvector **x** (ter lengte M) vormen we om naar een nieuwe vector $\underline{x}$ ter lengte M volgens:

$$\underline{x}[n]\begin{cases} x[2n]\cdot e^{-j\pi n/M}, & 0\le n\le\left\lceil\frac{M}{2}\right\rceil \\ -x[2(M-n)-1]\cdot e^{-j\pi n/M}, & \left\lfloor\frac{M}{2}\right\rfloor\le n\le M-1 \end{cases} \tag{4.16}$$

- Berekenen van $\underline{x}$ een M-DFT.
- Vermenigvuldig uitgangen van de DFT met $\sqrt{(2/M)}\mathbf{W}(4M,M,\frac{1}{2})$

Oftewel:

$$X[k]=\sqrt{\frac{2}{M}}Re\left\{W_{4M}^{k+\frac{1}{2}}\sum_{n=0}^{2M-1}W_{M}^{kn}\underline{x}[n]\,W_{M}^{\frac{1}{2}n}\right\},\qquad 0\le k\le M-1 \tag{4.17}$$

Nadeel: Reële deel operator, imaginaire operaties.
Voordeel: M-punts DFT i.p.v. 2M-punts, Simpele implementatie

Methode 4:

Deze methode gaat uit van een recursieve betrekking met $M=2^i$ (i positief integer).De DCT-IV is te schrijven als:

$$\left.\begin{aligned} X[k] &= \frac{1}{2C_{2M}^{k+\frac{1}{2}}}[X1[k]+X2[k]] \\ X[M-k-1] &= \frac{1}{2S_{2M}^{k+\frac{1}{2}}}[X1[k]-X2[k]] \end{aligned}\right\} \quad S_j^i=\sin\left(\frac{\pi i}{j}\right), \quad 0\le k\le\frac{M}{2}-1 \tag{4.18}$$

met

$$\begin{aligned} X1[k] &= \sum_{n=0}^{M/2-1}(x[2n]+x[2n+1])C_{M/2}^{(k+\frac{1}{2})(n+\frac{1}{2})} \\ X2[k] &= \sum_{n=0}^{M/2-1}(x[2n]+x[2n-1])C_{M/2}^{n(k+\frac{1}{2})} \end{aligned} \qquad met \quad x[-1]=0 \tag{4.19}$$

Dus de DCT-IV wordt opgesplitst in een DCT-IV en een DCT-II.Ook deze DCT-II kan recursief geschreven worden:

$$\left.\begin{aligned} X[k] &= X3[k]+X4[k] \\ X[M-k-1] &= X3[k]-X4[k] \end{aligned}\right\} \quad 0\le k\le\frac{M}{2}-1 \tag{4.20}$$

met

$$\begin{aligned} X3[k] &= \sum_{n=0}^{M/2-1}x[2n]\,C_{M/2}^{n(k+\frac{1}{2})} \\ X4[k] &= \frac{1}{2C_M^{k+\frac{1}{2}}}\sum_{n=0}^{M/2-1}(x[2n+1]+x[2n-1])C_{M/2}^{n(k+\frac{1}{2})} \end{aligned} \tag{4.21}$$

Nadeel: Complexe implementatie
Voordeel: Geen imaginaire operaties

4.4 MATLAB's DCT functies

De MATLAB functie *dct_iv* simuleert de implementatie van de DCT-IV en kan gebruik maken van de functie *dct_ii* die de implementatie van de DCT-II simuleert.

De functies *dct_iv* en *dct_ii* hebben dezelfde in- en uitgangsparameters:
in: - x: ingangsvector of matrix.
- M: aantal kanalen.
- s: nummer van de te kiezen methode:
  s=0: de DCT wordt bepaald via de rechtstreekse methode m.b.v. Formule (4.3).
  s=1: via methode 1.
  s=2: via methode 2.
  s=3: via methode 3.
  s=4: via methode 4.

uit: Y: uitgangsvector of matrix.

Voor de programma tekst zie appendix A.1.

Hoofdstuk 5

Implementatie CM Filter Bank

5.1 Inleiding

De implementatie van de filterbank bestaat uit het implementeren van de analyse- en synthese bank (paragraaf 5.2). [1] en [5] geven hiervan enkele figuren die we in paragraaf 5.2 verder uitwerken. In paragraaf 5.3 komt de implementatie complexiteit aan de orde.

5.2 Implementatie van de analyse- en synthese bank

Bij de totale implementatie van de analyse bank maken we gebruik van de resultaten uit paragraaf 2.3 middels Formule (2.15):

$$h(z)=E(z^{M})e(z)\ , \qquad E(z)=T\begin{bmatrix} g_0(z^2) \\ z^{-1}g_1(z^2) \end{bmatrix} \tag{5.1}$$

en hoofdstuk 4 middels Formule (4.1)

$$T=[A_0 \ \ A_1] \tag{5.2}$$

en Formule (4.2)

$$\begin{aligned} A_0 &= \sqrt{M}(-1)^{m_1}C(I-J) && m \text{ even} \\ A_1 &= \sqrt{M}(-1)^{m_1+1}C(I+J) && m=2m_1 \\ A_0 &= \sqrt{M}(-1)^{m_1}C(I+J) && m \text{ oneven} \\ A_1 &= \sqrt{M}(-1)^{m_1}C(I-J) && m=2m_1+1 \end{aligned} \tag{5.3}$$

Combineren we nu Formule (5.2) en (5.3) met Formule (5.1) dan ontstaan de volgende twee uitdrukkingen voor de analyse bank:

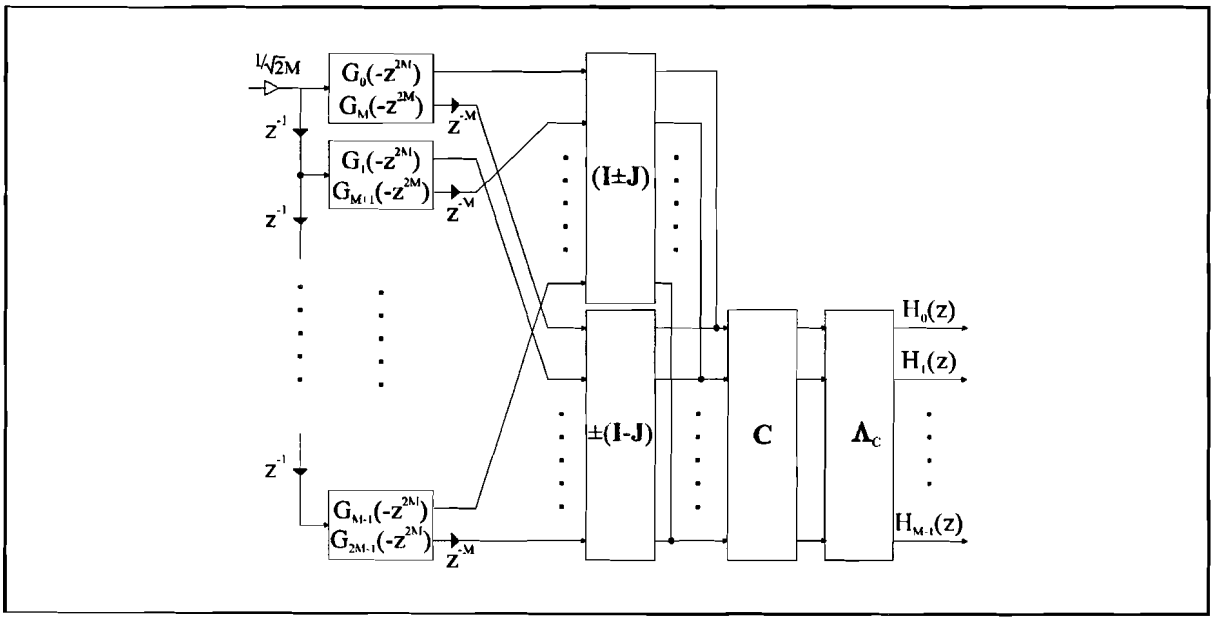
$$h(z)=\sqrt{M}(-1)^{m_1}C[I-J \quad -(I+J)]\cdot\begin{bmatrix} g_0(z^{2M}) \\ z^{-M}g_1(z^{2M}) \end{bmatrix}e(z) \qquad m \text{ even} \quad m=2m_1 \tag{5.4}$$

en

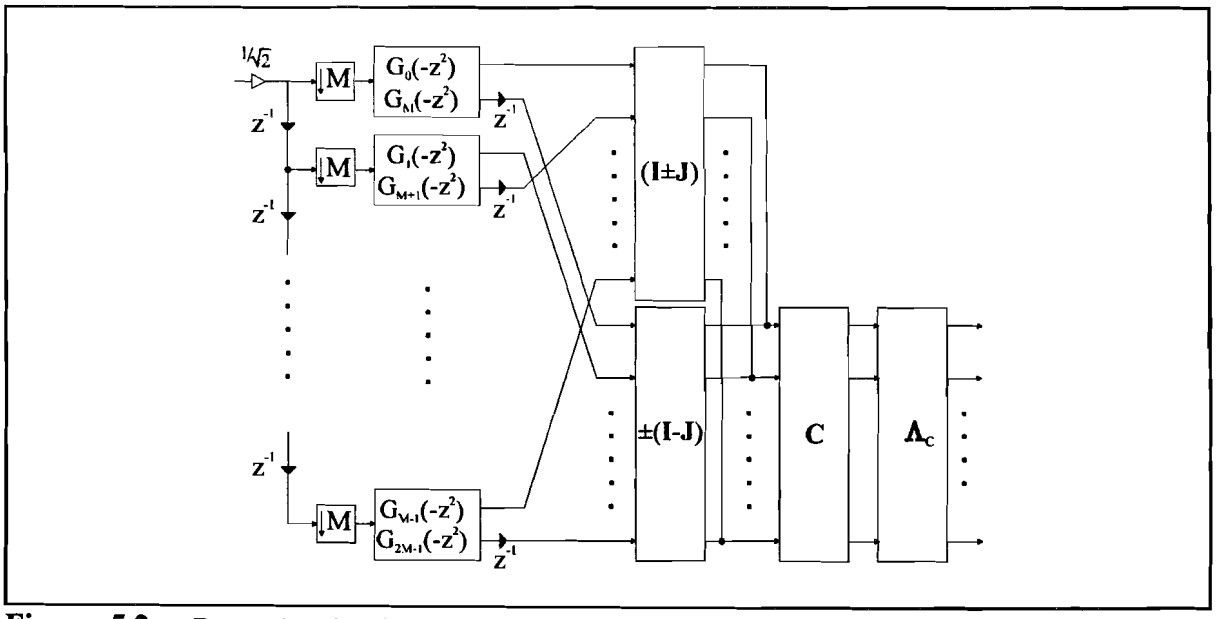
$$h(z)=\sqrt{M}(-1)^{m_1}C[I+J \quad I-J]\cdot\begin{bmatrix} g_0(z^{2M}) \\ z^{-M}g_1(z^{2M}) \end{bmatrix}e(z) \qquad m \text{ oneven} \quad m=2m_1+1 \tag{5.5}$$

Voor de implementatie van **C**, de DCT kan één van de methoden uit hoofdstuk 4 gekozen worden en voor de implementatie van de polyfase componenten kan een structuur (lattice of transversaal) uit hoofdstuk 3 gekozen worden.

Deze analyse bank is afgebeeld in Figuur 5.1. De decimators kunnen ook naar voren gebracht worden waarvan de implementatie in Figuur 5.2 afgebeeld is. Λ_c stelt in deze figuren een diagonaal matrix voor met de elementen $(-1)^{m_l}$.



Figuur 5.1 De ongedecimeerde cosinus gemoduleerde analyse bank.



Figuur 5.2 De maximaal gedecimeerde cosinus gemoduleerde analyse bank.

Een uitdrukking voor de synthese bank vinden we door Formule (2.18):

$$f^T(z)=z^{-(2mM-1)}\tilde{e}(z)\tilde{E}(z^M)=e(z)z^{-(2(m-1)M+M)}\tilde{E}(z^M) \tag{5.6}$$

te combineren met Formule (2.22):

$$\tilde{E}(z)=E_*^T(z^{-1})=\begin{bmatrix}\tilde{g}_0(z^2) & z\tilde{g}_1(z^2)\end{bmatrix}\begin{bmatrix}A_0^T\\ A_1^T\end{bmatrix} \tag{5.7}$$

Formule (2.25):

$$\begin{aligned} g_1(z)&=z^{-(m-1)}(-1)^{m-1}J\tilde{g}_0(z)J\\ &\textit{en}\\ g_0(z)&=z^{-(m-1)}(-1)^{m-1}J\tilde{g}_1(z)J\end{aligned} \tag{5.8}$$

en Formule (4.5):

$$\begin{aligned} A_0^T&=(I-J)C\sqrt{M}(-1)^{m_1} && m\ even\\ A_1^T&=(I+J)C\sqrt{M}(-1)^{m_1+1} && m=2m_1\\ A_0^T&=(I+J)C\sqrt{M}(-1)^{m_1} && m\ oneven\\ A_1^T&=(I-J)C\sqrt{M}(-1)^{m_1} && m=2m_1+1\end{aligned} \tag{5.9}$$

Dit levert de volgende twee uitdrukkingen voor de synthese bank op:

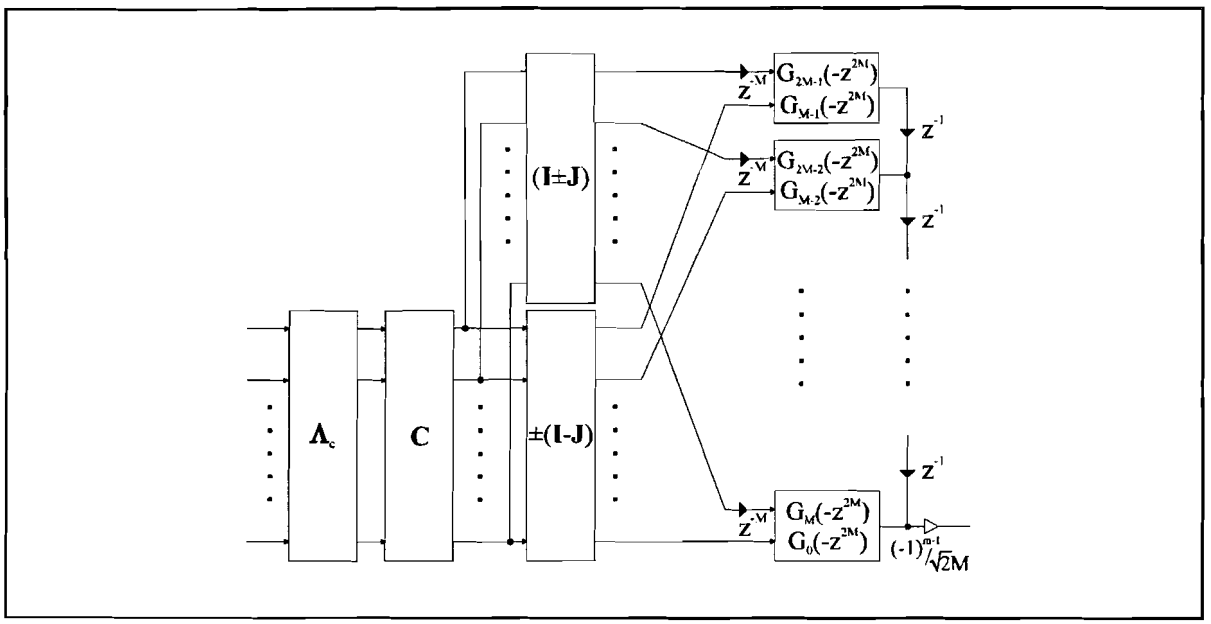
$$f^T(z)=\sqrt{M}e(z)\begin{bmatrix}z^{-M}Jg_1(z^{2M})J & Jg_0(z^{2M}J)\end{bmatrix}\cdot\begin{bmatrix}I-J\\ -(I+J)\end{bmatrix}(-1)^{m_1}C \quad m\ even \quad m=2m_1 \tag{5.10}$$

en

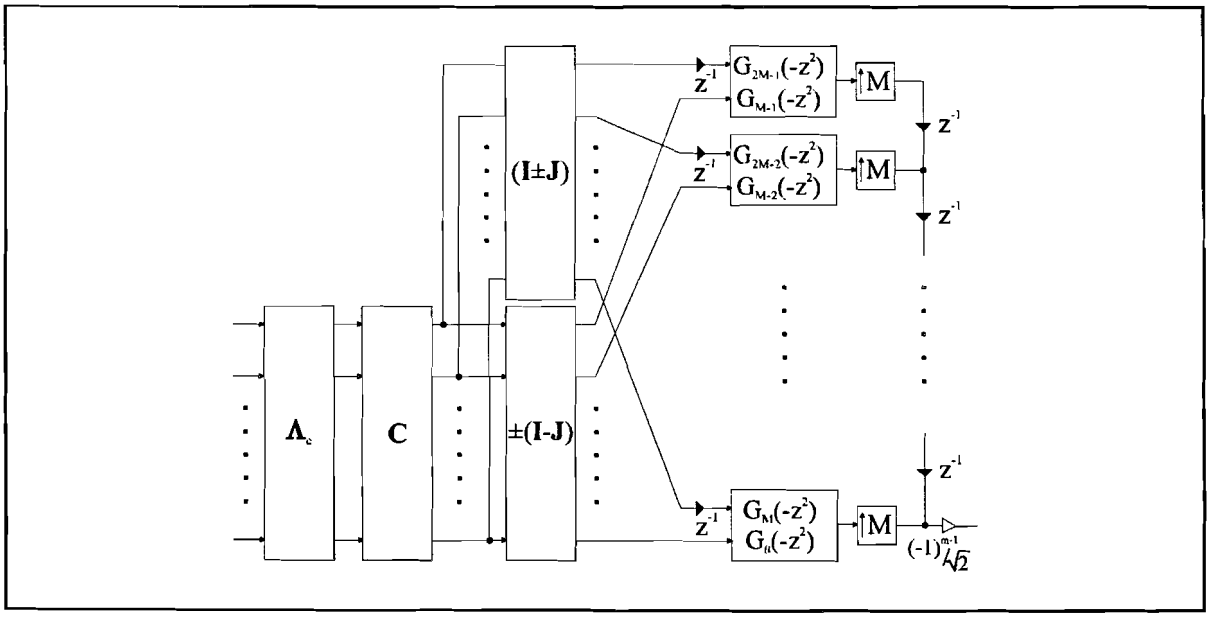
$$f^T(z)=\sqrt{M}e(z)\begin{bmatrix}z^{-M}Jg_1(z^{2M})J & Jg_0(z^{2M}J)\end{bmatrix}\cdot\begin{bmatrix}I+J\\ (I-J)\end{bmatrix}(-1)^{m_1}C \quad m\ oneven \quad m=2m_1+1 \tag{5.11}$$

Wat betreft de implementatie van de synthese bank geldt hetzelfde als wat we voor de analyse bank opgemerkt hadden.

Figuur 5.3 geeft de implementatie van een niet-geïnterpoleerde filterbank en Figuur 5.4 waar de interpolators naar buiten zijn geplaatst.



Figuur 5.3 De ongedecimeerde cosinus gemoduleerde synthese bank.



Figuur 5.4 De maximaal gedecimeerde cosinus gemoduleerde synthese bank.

5.3 Implementatie Complexiteit CM systeem

De implementatie complexiteit van een CM filter bank kan opgesplitst worden in de implementatie complexiteit van de polyfase componenten en de modulatie matrix. In deze paragraaf spreken we, zonder dat elke keer te vermelden over maximaal gedecimeerde filter banken.

De polyfase componenten kunnen geïmplementeerd worden m.b.v. een lattice of transversale structuur. We gaan ervan uit dat de implementatie m.b.v. een lattice structuur uitgevoerd wordt met een twee vermenigvuldigers lattice. De lattice structuur bestaat uit M 2-kanaals lattices die elk bestaan uit 2m vermenigvuldigers en 2(m-1) optellingen. Dus het aantal vermenigvuldigingen en optellingen van de totale polyfase structuur per tijdseenheid is respectievelijk gelijk aan 2m en 2(m-1). Deze aantallen gelden voor zowel de analyse- als synthese bank.

Het aantal vermenigvuldigingen en optellingen per tijdseenheid voor een transversale structuur in de analyse bank is gelijk aan het aantal dat nodig voor de lattice structuur. In de synthese bank zijn het aantal vermenigvuldigingen en optellingen per tijdseenheid respectievelijk gelijk aan 2m en 2m-1.

Het aantal delay elementen in de polyfase structuur van de analyse- en synthese bank met een lattice structuur is gelijk aan (2m-1)M. Dit voor de analyse bank met een transversale structuur is ook (2m-1)M, maar voor de synthese bank (4m-3)M.

De complexiteit van de modulatie matrix bestaat uit de complexiteit van de zogenaamde **IJ** matrix en DCT die afhankelijk is van de gekozen implementatie (zie de vier methoden uit hoofdstuk 4). Deze **IJ** matrix bestaat uit slechts optellingen en is gedefinieerd voor de analyse- en synthese bank respectievelijk als:

$$IJ_a = [I \pm J \quad \pm I - J] \tag{5.12}$$

en

$$IJ_s = \begin{bmatrix} I \pm J \\ \pm I - J \end{bmatrix} \tag{5.13}$$

De $\mathbf{IJ}_a$ matrix kan geïmplementeerd worden d.m.v. ongeveer 3 optellingen per tijdseenheid en de $\mathbf{IJ}_s$ matrix d.m.v. ongeveer 2 optellingen per tijdseenheid.

De implementatie complexiteit van de DCT is afhankelijk van de gekozen methode uit hoofdstuk 4. Wanneer we de DCT direct implementeren (via Formule (4.3)) dan zijn er M vermenigvuldigingen en M-1 optellingen per tijdseenheid nodig om de DCT te implementeren. Passen we bijvoorbeeld methode 4 met M=16 dan is het aantal vermenigvuldigingen en optellingen per tijdseenheid respectievelijk gelijk aan ongeveer 3 en 6.

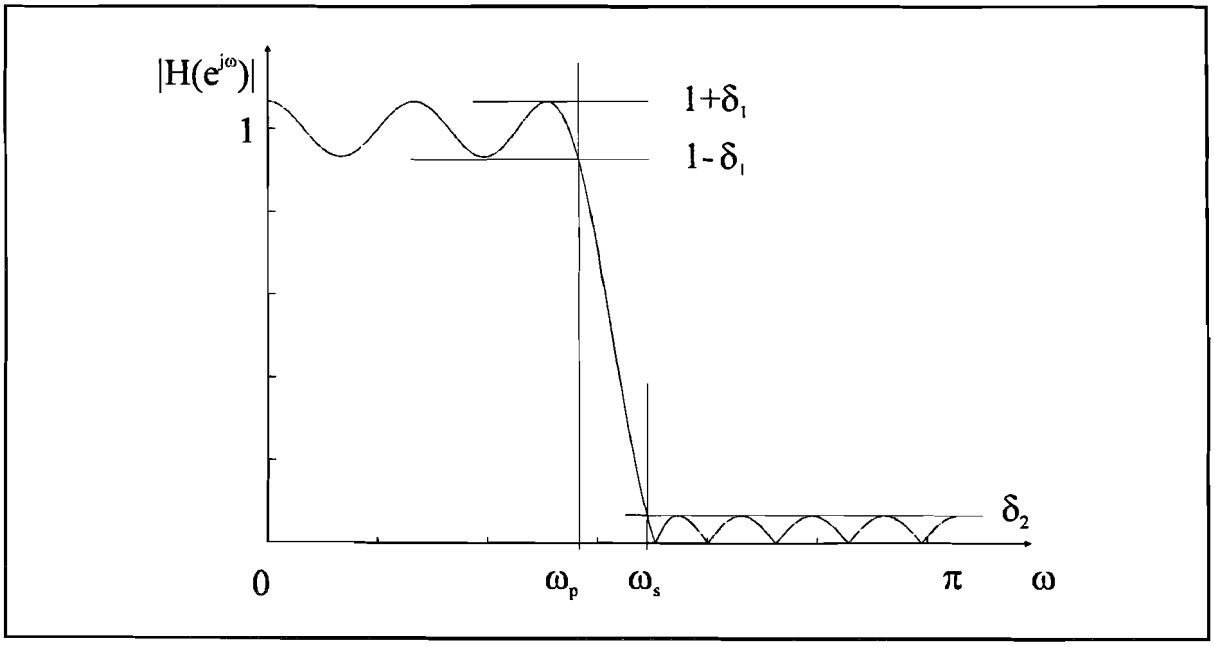
Hoofdstuk 6

Ontwerpen prototype

6.1 Inleiding

In dit hoofdstuk gaan we procedures afleiden om een prototype te ontwerpen t.b.v. een lattice en transversale structuur. Zie [1],[5] [17] en [18] voor meer literatuur over het ontwerpen van prototypen. In het geval van een lattice structuur hoeft geen extra ontwerp eis toegevoegd te worden om de filterbank de perfecte reconstructie te geven. In het geval van een transversale structuur is dit wel nodig. Bij de afleiding van deze procedures worden filter specificaties gebruikt. Figuur 6.1 geeft een frequentie response van een filter waarbij deze filter specificaties illustratief ondersteund worden. Dit is een *Equiripple* filter, maar de specificaties gelden ook voor een ander soort filter. De daarin gebruikte termen zijn hieronder gedefinieerd:

- δ_1 : maximum doorlaatband rimpel
- δ_2 : maximum stopband rimpel
- $A_s=-20\log_{10}\delta_2$: minimum stopband onderdrukking
- ω_p : doorlaatband grens met $|H(e^{j\omega p})|=1-\delta_1$
- ω_s :stopband grens met $|H(e^{j\omega s})|=\delta_2$
- $\Delta_f=(\omega_s-\omega_p)/2\pi$: genormaliseerde transitie bandbreedte
- ω_B : Bandbreedte $|H(e^{j\omega B})|=½$.



Figuur 6.1 Filter specificaties.

6.2 Ontwerp procedure t.b.v lattice structuur

In deze paragraaf gaan we een procedure afleiden om een prototype filter te ontwerpen m.b.v. lattice coëfficiënten. Deze procedure verschilt enorm met procedures waarbij gebruik gemaakt wordt van windows en het zogenaamde equiripple design.

Het ontwerpen van een CM filter bank is in feite het ontwerpen van één FIR laagdoorlaat (LDF) symmetrisch (lineaire fase) prototype filter. Als we uit gaan van de twee-kanaals lattice structuur dan is daarmee al aan de PR eis voldaan. Er hoeven dus geen beperkingen op de ontwerp parameters gelegd te worden. Deze ontwerp parameters zijn in feite de lattice coëfficiënten. Het ontwerpen bestaat dus uit het optimaliseren van de lattice coëfficiënten zodanig dat het LDF prototype met bandbreedte $\pi/2M$ een goede stopband onderdrukking heeft. Vanwege de PR eigenschap ontstaat ook een goede doorlaatband.

6.2.1 Aantal ontwerp parameters

Het aantal te optimaliseren lattice coëfficiënten is afhankelijk van M. Voor M even ontwerpen we M/2 power complementaire paren met elk m lattice coëfficiënten, dus mM/2 lattice coëfficiënten. Voor M oneven ontwerpen we (M-1)/2 power complementaire paren, dus het aantal te ontwerpen lattice coëfficiënten is m(M-1)/2. We vatten dit samen tot $m\lfloor M/2 \rfloor$ lattice coëfficiënten. Al deze lattice coëfficiënten verzamelen we in de volgende vector E:

$$E = [\theta_{0,0}, \theta_{0,1}, \cdots, \theta_{0,m-1}, \theta_{1,0}, \cdots, \theta_{1,m-1}, \theta_{2,0}, \cdots\cdots, \theta_{\lfloor M/2 \rfloor, m-1}] \qquad (6.1)$$

6.2.2 Ontwerp procedure

Het ontwerpen van het prototype $P_0(z)$ bestaat uit het minimaliseren van een objectieve functie, die iets zegt over de stopband. In Formule (6.2) worden twee objectieve functies gedefinieerd.

$$\Phi_1 = \int_{\frac{\pi}{2M}(1+\delta)}^{\pi} |P_0(e^{j\omega})|^2 d\omega \qquad \Phi_2 = \max_{\omega \in [\frac{\pi}{2M}(1+\delta), \pi]} |P_0(e^{j\omega})| \qquad (6.2)$$

Als gebruik gemaakt wordt van Φ_1 dan wordt de stopband energie geminimaliseerd en met Φ_2 het maximum van de filter response in de stopband (Minimax). We zien in beide objectieve functies dat gebruikt gemaakt wordt van de transfer functie, of impuls response (vanwege FIR filter). Om een impuls response van het prototype te vinden dienen m.b.v. de lattice coëfficiënten de polyfase componenten uitgeschreven te worden m.b.v. de volgende recursieve betrekking:

$$\begin{bmatrix} G_k^p(z) \\ G_{M+k}^p(z) \end{bmatrix} = \begin{bmatrix} \cos\theta_{k,p} & \sin\theta_{k,p} \\ \sin\theta_{k,p} & -\cos\theta_{k,p} \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix} \cdot \begin{bmatrix} G_k^{p-1}(z) \\ G_{M+k}^{p-1}(z) \end{bmatrix}, \qquad 1 \le p \le m-1 \tag{6.3}$$

met

$$\begin{bmatrix} G_k^0(z) \\ G_{M+k}^0(z) \end{bmatrix} = \begin{bmatrix} \cos\theta_{k,0} \\ \sin\theta_{k,0} \end{bmatrix} \tag{6.4}$$

voor $0 \le k \le M/2-1$ als M even is en voor M oneven $0 \le k \le (M-1)/2-1$. Voor M oneven geldt ook nog:

$$\begin{bmatrix} G_{\frac{M-1}{2}}^p(z) \\ G_{\frac{3M-1}{2}}^p(z) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix} \cdot \begin{bmatrix} G_{\frac{M-1}{2}}^{p-1}(z) \\ G_{\frac{3M-1}{2}}^{p-1}(z) \end{bmatrix}, \qquad 1 \le p \le m-1, \qquad \begin{bmatrix} G_{\frac{M-1}{2}}^0(z) \\ G_{\frac{3M-1}{2}}^0(z) \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix} \tag{6.5}$$

En vanwege de lineaire fase van $P_0(z)$ geldt:

$$\begin{aligned} G_{2M-1-k}(z) &= z^{-(m-1)}\tilde{G}_k(z) \\ G_{M-1-k}(z) &= z^{-(m-1)}\tilde{G}_{M+k}(z) \end{aligned} \tag{6.6}$$

waardoor de impulsrepsonse van $P_0(z)$ samen gesteld kan worden m.b.v. de impulsresponsies van de polyfase componenten, m.b.v. Formule (2.2):

$$P_0(z) = \sum_{l=0}^{2M-1} z^{-l} G_l(z^{2M}) \tag{6.7}$$

Het optimaliseren van de $m\lfloor M/2 \rfloor$ lattice coëfficiënten wordt uitgevoerd in een aantal stappen (Een sub-optimalisatie, een initialisatie en een minimalisatie). De sub-optimalisatie is een optionele stap, die zelf ook weer uit deze drie stappen kan bestaan. De reden om een sub-optimalisatie te gebruiken is het feit dat de snelheid van een minimalisatie methode voornamelijk bepaald wordt door de beginwaarden van de lattice coëfficiënten. Hoe dichter deze beginwaarden bij de optimale eindwaarden liggen des te sneller worden deze eindwaarden bereikt. Het vinden van goede beginwaarden geeft vooral problemen bij m>2. Een methode om goede beginwaarden te vinden is eerst een sub-optimalisatie uit te voeren voor een prototype met een lengte $N_1=2m_1M$, $(1<m_1<m)$ en de daarbij verkregen eindwaarden te gebruiken als beginwaarden voor de optimalisatie van het prototype met lengte N=2mM. De beginwaarden voor de sub-optimalisatie voor een prototype ter lengte N_1 kunnen de eindwaarden van de optimalisatie van een prototype ter lengte $N_2=2m_2M$, $(1<m_2<m_1)$ zijn. Uiteraard kan zo'n sub-optimalisatie meerdere keren toegepast worden, $(1< m_1 < m_2 \cdots < m_i < \cdots < m)$. Elke nieuwe optimalisatie voegt in feite m_i-m_{i-1} secties toe aan elk twee-kanaals lattice. Het vergroten van de prototype lengte kan ook gedaan wor-

den door het verhogen van het aantal kanalen. Indien we dit verhogen beperken met 1 en dit alleen doen bij M even dan neemt het aantal lattice coëfficiënten niet toe. De lengte van het prototype wordt nu met 2m verhoogd d.m.v. de toevoeging van de triviale lattice uit Formule (6.5).

6.2.3 Ontwerp stappen

De ontwerp procedure bestaat uit de volgende stappen:

- Initialisatie lattice coëfficiënten
  Het initialiseren kunnen we op splitsen in twee delen. De initialisatie van de allereerste optimalisatie en de initialisaties van de eventueel daarop volgende optimalisaties waarbij gebruik gemaakt wordt van verkregen eind waarden. De eerste geeft de lattice coëfficiënten de volgende waarden:

$$\theta_{k,p}=\begin{cases}\dfrac{\pi}{4}, & p=0,\\ \dfrac{\pi}{2}, & 0\le p\le m_1-1,\end{cases} \qquad 0\le k\le\left\lfloor\frac{M}{2}\right\rfloor \tag{6.8}$$

  Voor m_1 geldt $1<m_1<m$. Formule (6.8) geeft een prototype filter P_0 met een stopband onderdrukking $A_s\cong 13$dB en een stopband grens $\omega_s < \pi/M$. Elke polyfase component van $P_0(z)$ bestaat uit één delay. Een initialisatie die gebruikt maakt van de eindwaarden van een vorige sub-optimalisatie maakt de lattice coëfficiënten van de toegevoegde secties m_i-m_{i-1} gelijk aan $\pi/2$.
- Minimaliseren van stopband energie m.b.v. objectieve functie Φ_1 , waarbij de transfer functie van P_0 (of een prototype P'_0 met m_i secties) bepaald wordt m.b.v. de lattice coëfficiënten en de Formules (6.3)-(6.7) en een minimalisatie procedure (die gebruikt maakt van een quasi-Newton methode) uit de OPTIMIZATION toolbox van MATLAB de functie *fminu*. In MATLAB worden alle andere bewerkingen ook uitgevoerd worden.
- Tenslotte minimaliseren we het maximum in de stopband m.b.v. objectieve functie Φ_2 m.b.v. de functie *minimax* uit de OPTIMIZATION toolbox van MATLAB.

6.2.4 MATLAB ontwerp functies

De volgende functies zijn in MATLAB geschreven m.b.t. het ontwerpen van de optimale lattice coëfficiënten.

- *proto*: Bepaald de impulsresponsie van het prototype P_0 m.b.v. de lattice coëfficiënten.
  in: - E: lattice coëfficiënten vector.
  - M: aantal kanalen.
  - m: aantal secties.
  uit: P0: vector met de impulsresponsie.

- *obj1*: Objectieve functie Φ_1 die de stopband energie bepaald van het prototype m.b.v. de maat voor de stopbandgrens ω_s , δ.
  in: - E: lattice coëfficiënten vector .
  - M: aantal kanalen.
  - m: aantal secties.
  - d: maat voor de stopbandgrens (=δ).
  uit: O: de stopband energie.

- *obj2*: Objectieve functie Φ_1 die een aantal functie evaluaties geeft van de absolute waarden van het prototype m.b.v. de maat voor de stopbandgrens ω_s , δ.
  in: - E: lattice coëfficiënten vector.
  - M: aantal kanalen.
  - m: aantal secties.
  - d: maat voor de stopbandgrens (=δ).
  - L: aantal functie evaluaties in de stopband t.b.v. *minimax*.
  uit: PR0: vector ter lengte L met de functie evaluaties in de stopband van de absolute waarde van het prototype filter.

- *est_p0*: De functie die gebruik maakt van bovenstaande functies om de optimale lattice coëfficiënten te vinden
  in: - E: lattice coëfficiënten vector.
  - M: aantal kanalen.
  - m: aantal secties.
  - d: maat voor de stopband grens (=δ).
  - options: zie OPTIMIZATION toolbox van MATLAB.
  - m1: bij een eventuele sub-optimalisatie 0<m1<m .
  - L: aantal functie evaluaties in de stopband t.b.v. *minimax*.
  uit: - E1: lattice coëfficiënten vector van de sub-optimalisatie m.b.v. stopband energie.
  - E2: lattice coëfficiënten vector bepaald m.b.v. stopband energie.
  - E3: lattice coëfficiënten vector bepaald m.b.v. *minimax*.

- *extendp0*: M.b.v. van deze functie kunnen eerder berekende lattice coëfficiënten gebruikt worden om uit te breiden met één kanaal en/of meerdere secties.
  in: - E: lattice coëfficiënten vector bepaald m.b.v. stopband energie.
  - EE: lattice coëfficiënten vector bepaald m.b.v. *minimax*.
  - M: aantal kanalen.
  - m: aantal secties.
  - d: maat voor de stopband grens (=δ).
  - options: zie OPTIMIZATION toolbox van MATLAB.
  - L: aantal functie evaluaties in de stopband t.b.v. *minimax*.
  - m1: toevoeging van secties m≤m1.
  - M1: toevoegen van één kanaal als M1='inc'.
  uit: - E2: lattice coëfficiënten vector bepaald m.b.v. stopband energie.
  - E3: lattice coëfficiënten vector bepaald m.b.v. *minimax*.
  - M: nieuw aantal kanalen.
  - m: nieuw aantal secties.

- *plotp0*: Maakt een plot van de frequentie responsies van de prototypen uitgaande van drie lattice coëfficiënten vectoren.
  in: - E1: lattice coëfficiënten vector van de sub-optimalisatie m.b.v. stopbandenergie.
  - E2: lattice coëfficiënten vector bepaald m.b.v. stopband energie.
  - E3: lattice coëfficiënten vector bepaald m.b.v. *minimax*.
  - M: aantal kanalen.
  - m: aantal secties.
  - d: maat voor de stopband grens (=δ).
  - m1: aantal secties van de sub-optimalisatie
  uit: - As1: stopband onderdrukking van het prototype via E1.
  - As2: stopband onderdrukking van het prototype via E2.
  - As3: stopband onderdrukking van het prototype via E3.
  - ws1: stopband grens van het prototype via E1.
  - ws2: stopband grens van het prototype via E2.
  - ws3: stopband grens van het prototype via E3.

Zie ook Appendix A.2.1.

6.3 Ontwerp procedure t.b.v. transversale structuur

Er zijn twee manieren om het prototype te ontwerpen dat te implementeren is in een transversale structuur. Ten eerste de resultaten uit vorige paragraaf gebruiken om de polyfase componenten uit te schrijven in een impulsresponsie. Deze impuls responsies kunnen direct gebruikt worden in de transversale structuur. Een tweede methode is uit te gaan van een conventionele ontwerp methode, zoals een equiripple lineaire fase ontwerp methode. We zullen gebruik gaan maken van het *Remez Exchange Algoritme*, dat in MATLAB *remez* heet. Om de ontwerp parameters in te stellen maken we gebruik van de *bellanger's* formule, die een relatie geeft tussen filter lengte en de stopband rimpel, de doorlaatrimpel en de transitiebandbreedte:

$$N = \frac{2\log_{10}\left(\frac{1}{10\delta_1\delta_2}\right)}{3\Delta_f} \tag{6.9}$$

Naarmate het aantal kanalen toeneemt wordt de bandbreedte kleiner, π/2M. Indien we de stopband en doorlaat rimpel voor elke waarde van M gelijk laten zijn, dan moeten we de genormaliseerde transitie bandbreedte aan het aantal kanalen M relateren.:

$$\Delta_f = \frac{\Delta_\omega}{2M} \tag{6.10}$$

M.b.v. N=2mM en Formule (6.10) kunnen we Formule (6.9) om schrijven als:

$$m\Delta_\omega = 2/3\log_{10}\left(\frac{1}{10\delta_1\delta_2}\right) = c, \qquad c=constant \tag{6.11}$$

De transitie breedte is nu bepaald bij gegeven M en m, maar wat worden de stopband en doorlaatband grens. Deze definiëren we m.b.v. Δ_ω en een nieuwe variabele Δ_s die de verschuiving van de stopband grens t.o.v. $\pi/2M$ weergeeft en dus ook de verschuiving van de doorlaatband-grens:

$$\omega_s = \frac{1+\Delta_\omega\Delta_s}{2M} \qquad \omega_p = \frac{1-\Delta_\omega(1-\Delta_s)}{2M} \tag{6.12}$$

De vraag is hoe bepalen we Δ_s en uitgaande van welke conditie? Deze conditie maken we gelijk aan de stelling dat als we de magnitude response van alle analyse filters bij elkaar optellen er een constante uit moet komen:

$$\sum_{k=0}^{M-1} |H_k(e^{j\omega})|^2 = 1 \tag{6.13}$$

Dit is de distorsie functie. Dus de analyse filters zijn power complementair.

Uit een aantal experimenten bij verschillende Δ_s bleek indien $|H_0(e^{j\pi/M})|=1/\sqrt{2}$ het linker lid van Formule (6.13) ongeveer gelijk is aan een constante. Bij deze experimenten kwam een waarde voor Δ_s =0.62 dat een gemiddelde is voor alle situaties met verschillende M en m. Tevens gaan we ervan uit dat er tussen $|H_0(e^{j\pi/M})|$ en Δ_s een lineair verband bestaat.

De ontwerp methode bestaat uit de volgende stappen

1) Bepaal voor gegeven M en m: Δ_w (m.b.v. $m\Delta_w$=c=6).
2) Initieel maken we Δ_s =0.62
3) Bereken prototype P_0
4) Bepaal $H_0(z)$.
5) Bereken $|H_0(e^{j\pi/M})|$
6) Valt $\sqrt{2}\cdot|H_0(e^{j\pi/M})|-1$ binnen de gestelde nauwkeurigheid dan stoppen anders bepalen we Δ_s opnieuw m.b.v. een zogenaamde intervalmethode (Regula Falsi) en ga terug naar stap 3.

Er blijken slechts 3 of 4 iteraties nodig te zijn, waardoor deze methode zeer snel is t.o.v. de methode uit paragraaf 6.1.2. Deze manier van ontwerpen geeft geen filterbank met perfecte reconstructie. Deze filter bank is in feite een pseudo cosinus gemoduleerde filter bank waarbij de amplitude distorsie geminimaliseerd wordt en waarbij de fase distorsie en aliasing error geëlimineerd zijn. Zie [1] en [19] voor meer over pseudo cosinus gemoduleerde filter banken.

6.3.1 MATLAB ontwerp functies

De gebruikte MATLAB functies zijn:

- *remezp*: Functie gebruikt om een FIR te ontwerpen volgens de methode beschreven in paragraaf 6.2.
  in: - M: aantal kanalen.
  - m: aantal secties.
  uit: - P: impulsresponsie van het prototype.
- *plotp*: Plot het frequentie spectrum van het prototype.
  in: - P: impulsresponsie van het prototype.
  - M: aantal kanalen.
  uit: - Asp: stopband onderdrukking van het prototype P.
  - wsp: stopband grens van het prototype P.

Zie ook Appendix A.2.2.

6.4 Ontwerp Gegevens

In deze paragraaf tonen we alleen gegevens die direct iets te maken hebben met het prototype. Gegevens die betrekking hebben op de filterbank in het algemeen worden in hoofdstuk 8 besproken.

De enige variabele in de ontwerp procedure is de stopbandgrens factor δ. De invloed van δ op de stopbandgrens en stopband onderdrukking is in Tabel 6.1 uiteengezet. Tevens is in deze tabel de stopband energie E_s opgenomen. E_s is gelijk aan de uitkomst van de objectieve functie Φ_1 uit Formule (6.2) uitgaande van de lattice coëfficiënten.

Tabel 6.1
M=4 m=4 Invloed van de stopband-grens en stopband onderdrukking als functie van δ.

δ	via lattice coëfficiënten E_2			via lattice coëfficiënten E_3		
	A_s [dB]	ω_s [rad]	E_s	A_s [dB]	ω_s [rad]	E_s
0.5	25.31	0.6504	1.4923×10^{-3}	20.26	0.589	9.2168×10^{-3}
0.75	34.19	0.7302	1.7925×10^{-4}	31.32	0.687	6.9363×10^{-4}
0.99	43.42	0.8161	2.9183×10^{-5}	41.825	0.785	5.9337×10^{-5}
1.00	43.97	0.8222	2.8146×10^{-5}	42.16	0.785	5.2211×10^{-5}
1.20	45.77	0.902	1.8701×10^{-5}	47.8	0.865	1.4236×10^{-5}
1.25	46.48	0.9143	1.2596×10^{-5}	49.18	0.8836	1.0506×10^{-5}
1.50	40.39	0.9756	2.8748×10^{-5}	53.65	0.9756	4.257×10^{-6}

Zie Appendix B.1. voor de frequentie spectra van de prototypes via E_2 en E_3. E_2 en E_3 zijn de lattice coëfficiënten die respectievelijk d.m.v. het minimaliseren van de energie en het maximum in de stopband verkregen worden. A_{si} stellen de minimum stopband onderdrukking voor het prototype uitgaande van de lattice coëfficiënten E_i . Het aantal lattice coefficiënten blijft in alle gevallen gelijk waardoor de energie in de stopband in alle gevallen ongeveer gelijk blijft. Dus naarmate de bandbreedte van het filter breder wordt neemt de stopbandonderdrukking toe. Appendix B.2 geeft het frequentie spectra van het prototype filter ontworpen m.b.v. *remezp.*

Als het aantal vrijheidsgraden (lattice coëfficiënten) gelijk is dan wil dat niet zeggen dat in de lattice structuur de filter lengten gelijk zijn. In de transversale structuur geldt dat wel want daar geldt dat het aantal vrijheidsgraden gelijk is aan N/2=mM (N is altijd even). In Tabel 6.2 worden bij een gelijk aantal lattice coëfficiënten een aantal verschillende prototype filters via de lattice structuur berekend. Tevens zijn daarvoor ook de prototype filters berekend uitgaande van de transversale structuur.

Zie Appendix B.3 voor de frequentie spectra van de diverse prototype filters.

Tabel 6.2

Aantal lattice coëfficiënten is 24 en δ=0.99 via E3						P: via *remezp*		
m	M	N	A_s [dB]	ω_s [rad]	E_s	A_s [dB]	ω_s [rad]	$E_s \times 10^{-6}$
2	24	96	26.08	0.135	2.517×10^{-3}	60.29	0.2025	1,3526
3	16	96	37.51	0.1963	1.837×10^{-4}	57.93	0.2209	2.2732
4	12	96	42.86	0.2638	4.564×10^{-5}	53.25	0.2577	6.81
6	8	96	47.37	0.3927	1.821×10^{-5}	53.27	0.3191	6.6004
8	6	96	57.46	0.5216	2.135×10^{-6}	53.74	0.3866	5.8095
12	4	96	65.59	0.7854	2.317×10^{-7}	54.3	0.5154	4.8636
2	25	100	26.28	0.1289	2.755×10^{-3}	60.33	0.1963	1.35
3	17	102	36.59	0.1841	2.098×10^{-4}	58.1	0.2086	2.2598
4	13	104	38.22	0.2454	1.363×10^{-4}	53.27	0.2332	6.8117
6	9	108	38.54	0.3497	1.523×10^{-4}	53.33	0.2823	6.6034
8	7	112	40.79	0.4479	8.164×10^{-5}	53.81	0.3313	5.8131
12	5	120	42.48	0.6259	4.799×10^{-5}	54.43	0.4127	4.8968

Hoofdstuk 7

Quantisatie Effecten

7.1 Inleiding

De quantisatie effecten in een filter bank bestaan uit coëfficiënt quantisatie, signaal quantisatie en subband quantisatie, waarvan we de laatste hier niet bespreken. Deze effecten hebben verschillende uitwerkingen bij verschillende implementaties van de filterbank. De implementatie van de filterbank kan uitgevoerd worden m.b.v. floating- of fixed-point getallen representatie.

Door de quantisatie van de coëfficiënten kan de stopband onderdrukking van de analyse filters negatief beïnvloed worden. Een belangrijker effect is dat door coëfficiënt quantisatie de Perfecte Reconstructie van de filter bank verloren gaat. Volgens de literatuur blijft het effect van coëfficiënt quantisatie in CM filter banken beperkt als een lattice structuur toegepast wordt.

Het tweede quantisatie effect is (interne) signaal quantisatie. Bij het onderzoek naar de invloed van signaal quantisatie in filterbanken kan gebruik worden gemaakt van het feit dat interne-, ingangs- en uitgangssignalen beschreven kunnen worden door stochastische processen. We zullen in dit verslag daar niet op ingaan. Bij de fixed-point implementaties zijn begrippen als dynamisch bereik en schaling van signalen belangrijk. Zie [20] en [21] voor een uitgebreide behandeling van deze begrippen.

In dit hoofdstuk komen enkele notaties voor waarvan we hieronder de definitie geven:

- $Q[c]_R$: De gequantiseerde versie van c waarbij afronden (Rounding) wordt gebruikt. Dit is de default quantisatie vorm.
- $Q[c]_{MT}$: De gequantiseerde versie van c waarbij modulus afbreken (Magnitude Truncation) wordt gebruikt.
- $Q[c]_{VT}$: De gequantiseerde versie van c waarbij waarde afbreken (Value Truncation) wordt gebruikt.
- $Q[\mathbf{C}]_{R,MT,VT}$: De gequantiseerde versie van matrix **C**. Een gequantiseerde matrix is een matrix waarvan de elementen $c_{i,j}$ gequantiseerd zijn: $Q[c_{i,j}]_{R,MT,VT}$ met de quantisatie vorm: Rounding, Magnitude- of Value Truncation.

Waarbij c kan staan voor een coëfficiënt of signaal.

7.2 Paraunitairiteit behouden onder coëfficiënt quantisatie

In deze paragraaf gaan we ervan uit dat de elementen van de cosinus modulatie matrix niet gequantiseerd worden of we stellen dat: $Q[\mathbf{T}^T]Q[\mathbf{T}]=\mathbf{I}$. De lattice coëfficiënten uit de polyfase componenten quantiseren we wel. Met $Q[\sin\theta_i]$ duiden we de gequantiseerde waarde van $\sin\theta_i$ aan met de quantisatie vorm Rounding, en met $Q[\mathbf{R}_i]$ een matrix met de gequantiseerde elementen uit $\mathbf{R}_i$ (Formule (1.6)) met Rounding:

$$Q[\boldsymbol{R}_0]=\begin{bmatrix}Q[\cos\theta_0]\\ Q[\sin\theta_0]\end{bmatrix} \qquad Q[\boldsymbol{R}_i]=\begin{bmatrix}Q[\cos\theta_i] & Q[\sin\theta_i]\\ Q[\sin\theta_i] & -Q[\cos\theta_i]\end{bmatrix}_{i=1\cdots J} \tag{7.1}$$

De matrix $\mathbf{R}_i$ blijft paraunitair onder quantisatie: $Q[\mathbf{R}_i]Q[\mathbf{R}^T_i]=\alpha\mathbf{I}$. En verder geldt dat $Q[\mathbf{R}_0]Q[\mathbf{R}^T_0]=\alpha$. Wel is de factor α erbij gekomen maar dit heeft op de paraunitairiteit van de polyfase structuur geen invloed.

De twee vermenigvuldigers lattice uit Formule 1.10 wordt als volgt gequantiseerd:

$$Q[\boldsymbol{R}_0]=\begin{bmatrix}Q[\cos\theta_0]\cdot S\\ Q[\sin\theta_0]\cdot S\end{bmatrix} \qquad Q[\boldsymbol{R}_i]=\begin{bmatrix}Q\left[\dfrac{\cos\theta_i}{\sin\theta_i}\right] & 1\\ 1 & -Q\left[\dfrac{\cos\theta_i}{\sin\theta_i}\right]\end{bmatrix}_{i=1\cdots J} \qquad S=\prod_{i=1}^{J}\sin\theta_i \tag{7.2}$$

ook deze matrices blijven paraunitair onder quantisatie. Dus de filter bank blijft in beide situaties een PR systeem als er tenminste geldt dat $Q[\mathbf{T}^T]Q[\mathbf{T}]=\mathbf{I}$.

Worden de polyfase componenten uitgevoerd met een transversale structuur dan zal de PR niet gehandhaafd kunnen blijven.

7.3 Getallen Representaties

Bij de implementatie van de filterbank op bijvoorbeeld een Digital Signal Processor (DSP) wordt een zekere getallen representatie gebruikt, een floating-point of fixed-point getallen representatie. Bij de simulaties van de filterbank (die in hoofdstuk 8 besproken worden) worden beide getallen representaties gebruikt.

Vaak wordt een N-bits fixed-point 2's complement getallen representatie gebruikt die een getal x begrenst door -1 en $1-2^{-(N-1)}$. Bij de simulaties (in hoofdstuk 8) in MATLAB is het

makkelijker om de representatie volgens Formule (7.3) te kiezen, waarbij de fixed-point getallen gerepresenteerd worden door een geheel getal x d.m.v.:

$$x = \pm \sum_{i=0}^{N-2} x_i 2^i \,, \qquad met \quad x_i \in \{0,1\} \tag{7.3}$$

waarbij uitgegaan wordt van een binaire getallen voorstelling (grondtal 2). Dus x wordt begrensd door $-2^{N-1}+1 \le x \le 2^{N-1}-1$ en het kleinste positieve en negatieve getal is respectievelijk gelijk aan 1 en -1. Het aantal bits gebruikt bij de representatie van een fixed-point getal is N waarvan één bit gebruikt wordt als sign-bit.

Floating-point getallen waarbij uitgegaan wordt van een binaire voorstelling worden gerepresenteerd d.m.v.:

$$x = \left[\pm \sum_{i=0}^{N-2} m_i 2^i \right] \cdot 2^{\left[\pm \sum_{j=0}^{Ex-2} e_j 2^j \right]} \,, \qquad met \quad m_i, e_j \in \{0,1\} \tag{7.4}$$

De positieve representeerbare getallen x liggen in het bereik:

$$2^{-(2^{Ex-1}-1)} \le x \le (2^{N-1}-1) 2^{(2^{Ex-1}-1)} \tag{7.5}$$

Om MATLAB getallen te converteren naar floating-point getallen volgens vorige representatie vereist veel rekenwerk en complexiteit, waardoor de simulaties niet snel zullen verlopen. Wanneer we een decimale voorstelling van de floating-point getallen nemen dan is dit wel eenvoudig in MATLAB te verwezenlijken:

$$x = \left[\pm \sum_{i=0}^{N-1} m_i 10^i \right] \cdot 10^{\left[\pm \sum_{j=0}^{Ex-1} e_j 10^j \right]} \,, \qquad met \quad m_i, e_j \in \{0,1,\cdots,9\} \tag{7.6}$$

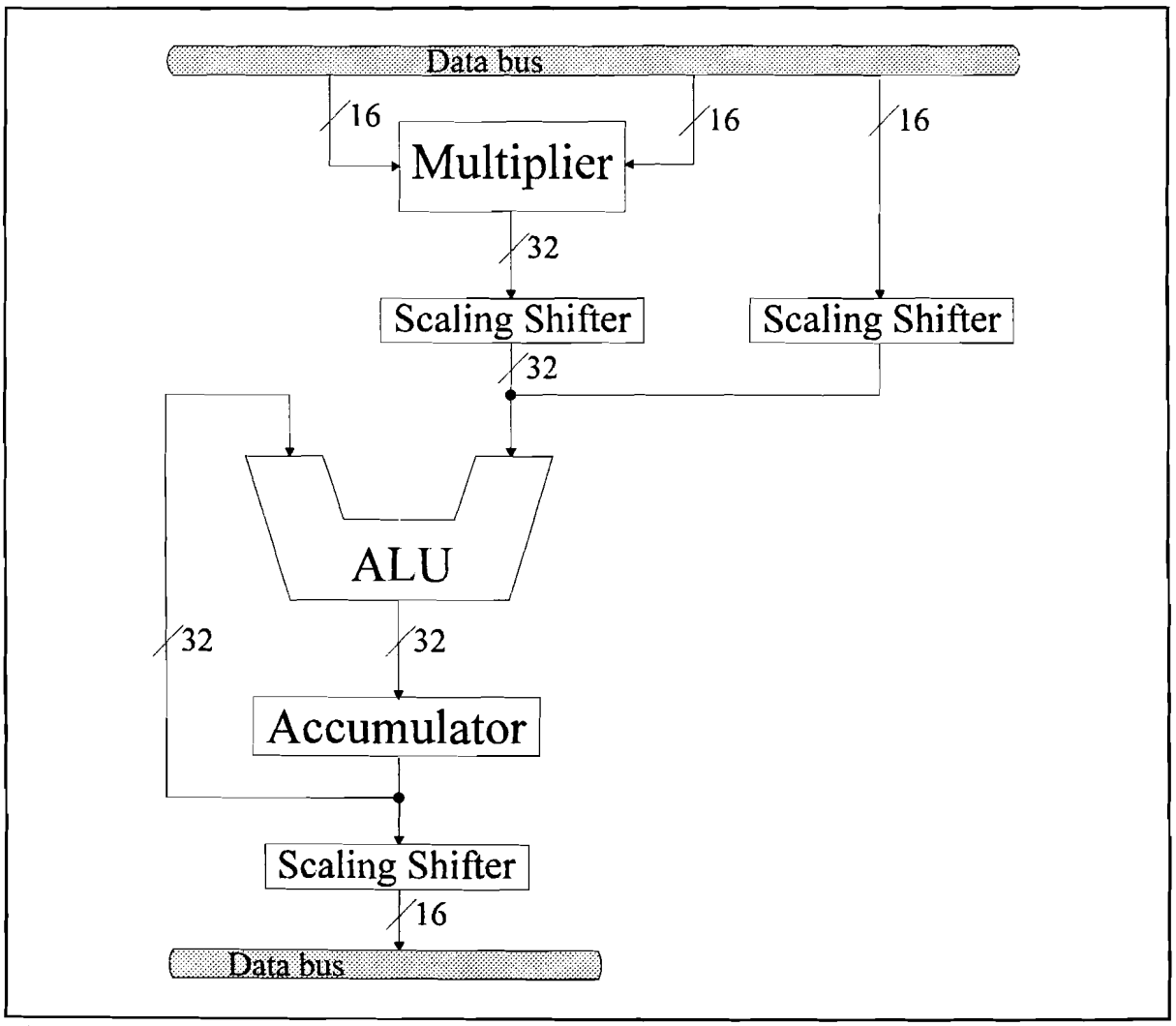
Om een praktische situatie met een binaire voorstelling te simuleren m.b.v. een decimale voorstelling geeft te grote verschillen. Daarom passen we de betekenis van Ex iets aan. Ex is nu de maximale exponent in plaats van de exponent lengte. Dit geeft voor een floating-point representatie het volgende:

$$x = \left[\pm \sum_{i=0}^{N-1} m_i 10^i \right] \cdot 10^E \,, \qquad met \quad m_i \in \{0,1,\cdots,9\} \quad en \quad -Ex \le E \le Ex \tag{7.7}$$

Bij de simulaties stellen we dus de floating-point getallen representatie in d.m.v. een mantisse lengte N en een maximale exponent grootte Ex.

7.4 Fixed-point architectuur

Om simulaties uit te voeren van een filterbank geïmplementeerd op een DSP moeten we in grote lijnen begrijpen hoe de DSP intern opgebouwd is en hoe hij berekeningen uitvoert. We gaan uit van een voorbeeld met de fixed-point DSP TMS320C25. De DSP TMS320C25 bezit een 16-bits databus, 16×16-bits vermenigvuldiger een 32 bits ALU, een 32-bits accumulator en schalings units (shifters) na de vermenigvuldiger en accumulator. Deze shifters kunnen gebruikt worden t.b.v. de schaling van signalen. Dit schalen gebeurt d.m.v. het verschuiven van bits. De DSP gebruik een 2's complement getallen representatie. Zie Figuur 7.1 voor zeer vereenvoudigd blokschema van het rekengedeelte van de TMS320C25. Zie ook [22].



Figuur 7.1 Blokschema van het rekengedeelte van de TMS320C25

In paragraaf 7.3 hadden we al opgemerkt dat vanwege de mogelijkheid om een fixed-point getallen representatie te simuleren in MATLAB kleine aanpassingen nodig waren. Een fixed-point geheel getal x is zodoende begrensd door: $-2^{N-1}+1 \leq x \leq 2^{N-1}-1$, zie Formule 7.3.

Als N voldoende groot is dan zullen deze twee uiterste waarden ongeveer gelijk zijn aan: -2^{N-1} en 2^{N-1}. Deze veronderstelling gebruiken we ook in de functies t.b.v. de simulaties m.b.v. MATLAB (Paragraaf 7.8).

De implementatie van een transversaal filter op een DSP kan d.m.v. enkele operaties uitgevoerd worden. Twee operaties waarin we hier geïnteresseerd zijn is de vermenigvuldiging van het ingangssignaal met de coëfficiënten en de optellingen van de afzonderlijke produkten. Als we een N-bits ingangssignaal met een N-bits coëfficiënt vermenigvuldigen dan ontstaat een 2N-bits resultaat dat begrensd wordt door de waarden: -2^{2N-2} en 2^{2N-2}. Dit 2N-bits produkt tellen we op bij een eventueel al bestaand 2N-bits tussenresultaat in de accumulator. Dit vermenigvuldigen en optellen kan een aantal keren herhaald worden afhankelijk van de orde van het geïmplementeerde filter. Om dit 2N-bits sommatieresultaat terug te brengen naar een N bits getal verschuiven we het 2N-bits resultaat N-1 bits, wat in feite een vermenigvuldiging van $2^{-(N-1)}$ met het 2N-bits eindresultaat voorstelt. De vermenigvuldigingsfactor wordt ook wel schalingsfactor genoemd.

Dit proces bekijken we wat nader. Als we twee 2N-bits produkten (beide verkregen door een vermenigvuldiging van een N-bits signaal en een N-bits coëfficiënt) bij elkaar optellen dan ontstaat een resultaat met een maximale absolute waarde van $2^{2N-2}+2^{2N-2}=2^{2N-1}$, wat nog net niet tot overflow leidt. Tellen we hierbij nog een 2N-bits produkt dan bestaat er al kans op overflow.

In de implementatie op een DSP moeten we zorgen dat signalen in een systeem het dynamisch bereik van de architectuur niet overschrijden. Wanneer signalen het dynamisch bereik overschrijden ontstaat *overflow*. Het is dus belangrijk om het totale systeem, in dit geval een transversaal filter, goed te quantiseren en schalen.

De strategie om een systeem goed te quantiseren/schalen bestaat uit het opdelen in zo'n klein mogelijke deelsystemen. Tevens gaan we er steeds vanuit dat de N-bits ingangssignalen van elk deelsysteem een maximaal dynamisch bereik hebben. Daarna quantiseren we de vermenigvuldigers coëfficiënten zo dat de kans op overflow nihil is. De 2N bits uitgangssignalen dienen daarna op een verantwoorde manier geschaald te worden om een N-bits resultaat over te houden dat een maximaal dynamisch bereik heeft. Hoe kleiner de kans op overflow hoe moeilijker het is om het dynamisch bereik van deze uitgangssignalen maximaal te maken.

Dus een goede strategie om een systeem te quantiseren/schalen bestaat uit het opdelen in zo'n klein mogelijke deelsystemen. De filter bank kan in de volgende delen worden opgesplitst:

- De polyfase componenten van de analyse- en synthese bank geïmplementeerd via een lattice- of transversale structuur. (Paragraaf 7.6)
- De **IJ** matrix uit de analyse bank (zie Paragraaf 7.7):

$$\boldsymbol{IJ}_a = [\boldsymbol{I} \pm \boldsymbol{J} \quad \pm \boldsymbol{I} - \boldsymbol{J}] \tag{7.8}$$

en uit de synthese bank:

$$IJ_s = \begin{bmatrix} I \pm J \\ \pm I - J \end{bmatrix} \tag{7.9}$$

- En de DCT matrix **C** (Paragraaf 7.8)

In de volgende paragrafen worden suggesties gedaan omtrent de schaling van signalen en quantisatie van coëfficiënten, met in het oog het dynamisch bereik en overflow. In hoofdstuk 8 bij de simulatie van de filterbank worden deze suggesties gebruikt en eventueel aangepast om een optimaal resultaat te krijgen. Dus de resultaten in de paragrafen 7.6 t/m 7.8 dienen uitsluitend als uitgangspunt.

7.5 Dynamisch Bereik, Overflow en Schaling m.b.t fixed-point architectuur

Met de fixed-point getallen representaties zijn twee begrippen onomstotelijk met elkaar verbonden, namelijk het dynamisch bereik en overflow. Overflow kan vermeden worden door de signalen (zowel interne- ingangs- als uitgangssignalen) te schalen d.m.v. een vermenigvuldiging met een schalingsfactor. Passen we te veel schaling toe, dan kan het dynamisch bereik van het uitgangssignaal dramatisch slecht worden. Het is daarom van groot belang op verschillende plaatsen in het systeem schaling toe te passen waardoor het dynamisch bereik van zowel interne- als uitgangssignalen zo groot mogelijk is tegen een zo klein mogelijke kans op overflow. Tevens is het van belang dat een goede quantisatie van diverse vermenigvuldigers coëfficiënten toegepast wordt.

Er kunnen verschillende soorten schaling toegepast worden. In dit verslag passen we een aantal soorten toe. Deze verklaren we aan de hand van een FIR filter met de impulsresponsie h[n] en uitgevoerd met een transversale structuur. De schaling methoden zijn:

ℓ^1-schaling:

$$\sum_n |h[n]| \leq 2^{N-1} \tag{7.10}$$

wat de meest strenge schaling is.

ℓ^2-schaling:

$$\sum_n |h[n]|^2 \leq 2^{N-1} \tag{7.11}$$

ℓ^∞-schaling:

$$\max_n \{|h[n]|\} \leq 2^{N-1} \tag{7.12}$$

wat de minst strenge schaling is.

7.6 Quantisatie/Schaling polyfase componenten

De polyfase componenten kunnen geïmplementeerd worden via een lattice- of transversale structuur. In paragraaf 7.4 hadden we als voorbeeld een transversaal filter geïmplementeerd op een DSP. Hierbij hadden de tussen resultaten een dynamisch bereik dat twee keer zo groot is als de ingangs- en uitgangssignalen. Echter bij de implementatie van een filter m.b.v. een lattice structuur dienen de tussenresultaten ook terug gebracht te worden naar N-bits om in het geheugen te kunnen bewaren. Waardoor de tussen resultaten een dynamisch bereik hebben dat gelijk is aan dat van ingangs- en uitgangssignalen, wat een nadeel is.

Een voordeel is dat we de polyfase componenten geïmplementeerd via een lattice structuur niet als geheel hoeven te zien maar we kunnen de lattice structuur splitsen in secties die we afzonderlijk quantiseren/schalen. Hierbij moet ook nog onderscheid gemaakt worden in de lattice structuur met vier en twee vermenigvuldigers.

7.6.1 Vier Vermenigvuldigers Lattice structuur

Om geen verwarring te laten ontstaan maken we onderscheid tussen de lattice coëfficiënten $\theta_{i,j}$ en de coëfficiënten uit een lattice structuur, $\sin\theta_{i,j}$ en $\cos\theta_{i,j}$. De verzameling coëfficiënten van alle secties bestaat uit de elementen: $R=\{\sin\theta_{0,0}$, $\cos\theta_{0,0}$, ··· , $\cos\theta_{\lfloor M/2 \rfloor m-1}\}$. De absolute waarden van al deze elementen zijn kleiner dan 1. De quantisatie van alle elementen kan geschieden door:

$$Q[R] = [R \cdot 2^{N-1}]_R \tag{7.13}$$

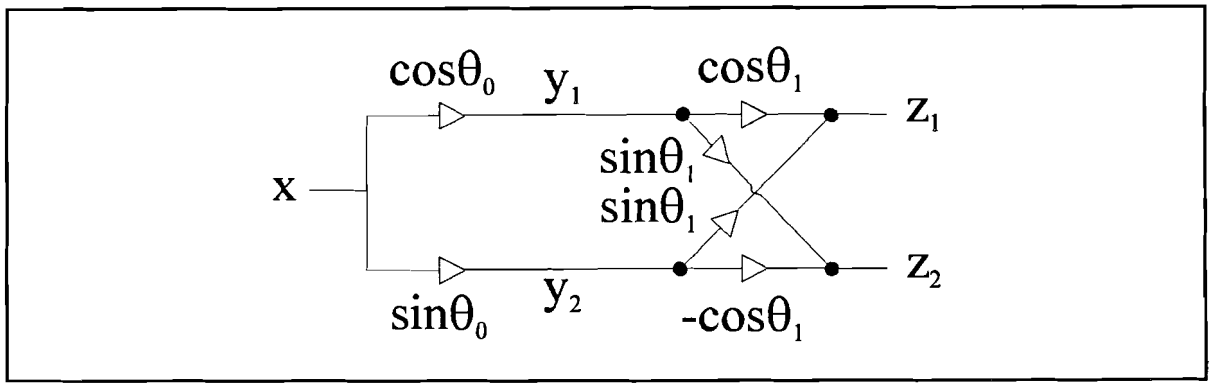
met $[\cdot]_R$ het afronden naar het dichtstbijzijnde integer. De uitgangssignalen van sectie 0 kunnen verschoven worden met N-1 bits met geen enkele kans op overflow.

Als we veronderstellen dat beide ingangssignalen van een sectie i ($1 \leq i \leq m-1$) gelijk zijn aan de maximaal te representeren waarde dan is de maximale absolute waarde van het uitgangssignaal gelijk aan:

$$2^{N-1}\left(2^{N-1}\max_{\theta}\{\sin\theta+\cos\theta\}\right)=2^{2N-3/2}\ , \qquad met\ \ \max_{\theta}\{\sin\theta+\cos\theta\}=\sqrt{2} \tag{7.14}$$

Het maximum treedt op bij $\theta=\pi/4$. Om dit 2N-bits uitgangssignaal terug te brengen naar N-bits dan kunnen we een verschuiving (schaling) toe passen van N bits met geen enkele kans op overflow. Er kan echter nooit de situatie optreden dat beide ingangen een maximale waarde hebben. De lattice structuur bestaat uit verliesvrije elementaire blokken, waardoor het vermogen aan de ingang gelijk is aan het vermogen aan de uitgang. Zie Figuur 7.2. Voor de signalen uit deze Figuur geldt:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}=\begin{bmatrix} \cos\theta_0 \\ \sin\theta_0 \end{bmatrix}\cdot x\ , \qquad \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}=\begin{bmatrix} \cos\theta_1 & \sin\theta_1 \\ \sin\theta_1 & -\cos\theta_1 \end{bmatrix}\cdot\begin{bmatrix} \cos\theta_0 \\ \sin\theta_0 \end{bmatrix}\cdot x=\begin{bmatrix} \cos(\theta_0-\theta_1) \\ \sin(\theta_0-\theta_1) \end{bmatrix}\cdot x \tag{7.15}$$



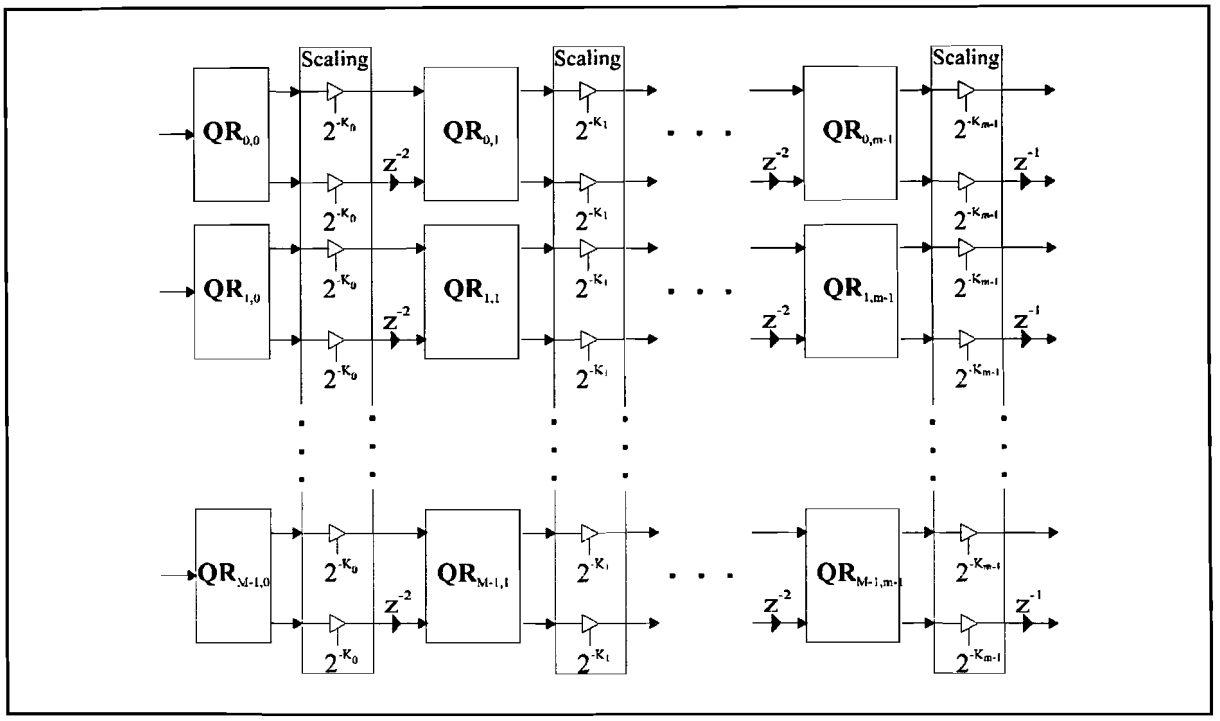
Figuur 7.2 Signalen in een lattice structuur.

Een maximum aan de uitgang z_1 ontstaat als x gelijk is aan de maximale waarde en $\theta_0=\theta_1=\pi/4$ waardoor $z_2=0$. Als we veronderstellen dat de coëfficiënten gequantiseerd zijn volgens Formule (7.13) en dat de uitgangssignalen van sectie 0 N-1 bits worden verschoven dan geldt dat $y_1=y_2=\frac{1}{2}\sqrt{2}\cdot 2^{N-1}$ en de ongeschaalde $z_1=\frac{1}{2}\sqrt{2}\cdot 2^{N-1}\cdot\sqrt{2}\cdot 2^{N-1}=2^{2N-2}$. Dus de uitgang z kan geschaald worden met een verschuiving van N-1 bits.

In de lattice structuur bevindt zich tussen alle secties nog een vertragingselement. Dus de twee ingangen van sectie 2 kunnen tegelijker tijd de maximale waarde hebben, waardoor een verschuiving van N bits van de uitgangssignalen nodig is om geen overflow op te laten treden. De maximale waarde van deze uitgangen is dus $2^{N-3/2}$. Maar de uitgangen van sectie 3 kunnen weer wel N-1 bits verschoven worden omdat de ingangen een maximale waarde van $2^{N-3/2}$ hebben. De uitgangen hebben nu weer een maximale waarde van 2^{N-1}. Voor de secties 4 en 5 geldt hetzelfde als voor respectievelijk 2 en 3. In het algemene geval geldt dus dat de uitgangen van de even secties geschaald kunnen worden d.m.v. het verschuiven van N-1 bits en de oneven met N bits.

Om het dynamisch bereik te vergroten, kunnen de secties apart gequantiseerd worden. Let wel dat bijvoorbeeld de secties i van de M lattices samen een deelsysteem vormen. We quantiseren niet elke sectie apart want dan zouden we elke lattice apart moeten schalen, waardoor dit gecompenseerd moet worden met extra vermenigvuldigingen of extra ver-

schuivingen. Zie Figuur 7.3. Hierin wordt na elke sectie i voor alle lattices dezelfde schaling toegepast. $\mathbf{QR}_{i,j}$ staat voor een gequantiseerde elementaire lattice sectie uit Figuur 1.4 en Figuur 1.5, dat wil zeggen dat de coëfficiënten uit zo'n elementaire sectie gequantiseerd zijn.



Figuur 7.3 De gequantiseerde lattice structuur van de polyfase componenten.

De verzameling coëfficiënten van de secties i bestaan uit de elementen: $R_i=\{\sin\theta_{0,i}, \cos\theta_{0,i}, \cdots, \cos\theta_{\lfloor M/2\rfloor,i}\}$. Het dynamisch bereik vergroten we door elk element uit deze verzameling te relateren aan het absolute maximum van deze verzameling, ℓ^∞-schaling. Het j^{de} element $R_i(j)$ uit de verzameling R_i quantiseren we dus als volgt:

$$Q[R_i(j)] = \left[\frac{R_i(j)}{\max\{|R_i|\}} \cdot 2^{N-1} \right]_R \tag{7.16}$$

met $[\cdot]_R$ het afronden naar het dichtstbijzijnde integer.

Voor de synthese bank geldt ongeveer hetzelfde. Het quantisatie proces is uiteraard gelijk aan dat van de analyse bank. De uitgangen van sectie m-1 schalen we d.m.v. een verschuiving van N bits. De uitgangen van sectie m-2 schalen we d.m.v. een verschuiving van N-1 bits. Het beurtelings verschuiven van N-1 en N bits gaat door t/m sectie 0.

7.6.2 Twee Vermenigvuldigers Lattice structuur

Voor de twee-vermenigvuldigers lattice geldt een ander verhaal, omdat de vermenigvuldiging met 1 geen vermenigvuldiging is maar hooguit een verschuiving (schaling). Dus de quantisatie van de coëfficiënten kan niet gedaan worden m.b.v. Formule (7.16). Een extra probleem vormen de coëfficiënten die verkregen zijn door een deling c/s=cosθ/sinθ, waardoor de coëfficiënten in absolute zin niet begrensd zijn door 1. Er kunnen dus grote verschillen ontstaan in de absolute waarde van de coëfficiënten c/s.

Net als in vorige paragraaf splitsen we de lattice structuur op in deelsystemen die elk bestaan uit de secties i van alle lattices. De quantisatie van de coëfficiënten van de secties 1···m-1 bestaat uit het vermenigvuldigingen met een macht van 2, 2^K. Deze K voor sectie i wordt gevonden door:

$$2^{N-2} \le \max_k \left\{ 1, \frac{\cos\theta_{k,i}}{\sin\theta_{k,i}} \right\} \cdot 2^K \le 2^{N-1} \qquad 0 \le k \le \lfloor M/2 \rfloor \tag{7.17}$$

en gequantiseerde coëfficiënten c/s worden gelijk aan:

$$Q\left[\frac{\cos\theta_{k,i}}{\sin\theta_{k,i}}\right] = \left[\frac{\cos\theta_{k,i}}{\sin\theta_{k,i}} \cdot 2^K\right]_R \qquad 0 \le k \le \lfloor M/2 \rfloor \tag{7.18}$$

en de vermenigvuldiging met 1 wordt een verschuiving van K bits.

De maximum uitgangssignaal van de lattices voor de secties 1···m-1 treedt op als de lattice coëfficiënt $\theta_{k,i}=\pi/4$ (c/s=1), en is gelijk aan 2^{2N-1}. Het 2N-bits uitgangssignalen dienen N bits verschoven te worden om N bits resultaten te krijgen zonder enige kans op overflow.

Voor sectie 0 kan wel Formule (7.16) toegepast worden. De 2N-bits uitgangssignalen van sectie 0 kunnen N-1 bits verschoven te worden om N bits resultaten te krijgen zonder enige kans op overflow.

Voor de synthese bank geldt precies hetzelfde.

In hoofdstuk 8 worden de simulaties besproken, waarbij verschillende resultaten bij verschillende situaties met elkaar vergeleken worden waarbij extra bits verschoven worden waardoor het dynamisch bereik en de kans op overflow vergroot worden.

7.6.3 Transversale structuur

Zijn de polyfase componenten geïmplementeerd volgens een transversale structuur dan zijn er twee situaties. De eerste waarbij de impuls responsies van de polyfase componenten direct uit de lattice coëfficiënten bepaald zijn, en de tweede waarbij de polyfase compo-

nenten uit het prototype, ontworpen m.b.v. de MATLAB's remez functie, bepaald zijn. Het quantiseren van de coëfficiënten kan niet opgesplitst worden in deelsystemen zoals bij de lattice structuur. De quantisatie van de 2M polyfase componenten kan gedaan worden d.m.v. een ℓ^1-schaling:

$$Q[g_k[n]] = \left\lfloor \frac{g_k[n]}{\max_k \left\{ \sum_{n=0}^{m-1} |g_k[n]| \right\}} \cdot 2^{N-1} \right\rfloor_R \tag{7.19}$$

die geen enkele kans op overflow geeft. Of een ℓ^2-schaling

$$Q[g_k[n]] = \left\lfloor \frac{g_k[n]}{\max_k \left\{ \sum_{n=0}^{m-1} g_k^2[n] \right\}} \cdot 2^{N-1} \right\rfloor_R \tag{7.20}$$

of ℓ^∞-schaling:

$$Q[g_k[n]] = \left\lfloor \frac{g_k[n]}{\max_{k,n} \{ g_k[n] \}} \cdot 2^{N-1} \right\rfloor_R \tag{7.21}$$

wat het grootste dynamisch bereik geeft. De simulatie in hoofdstuk 8 geven hiervan resultaten.

7.7 Quantiseren/Schalen IJ matrix

De implementatie van **IJ** matrix bestaat in feite uit optellingen en verschuivingen omdat alle vermenigvuldigingen bestaan uit een produkt van een signaal met een coëfficiënt dat een macht van twee is. Door de **IJ** matrix te quantiseren zodanig dat de elementen machten van twee blijven dan blijft de implementatie bestaan uit optellingen en verschuivingen.

Ten eerste veronderstellen we net als in vorige paragraaf dat de ingangssignalen een maximaal bereik hebben. Om te onderzoeken hoe **IJ** gequantiseerd moet worden maken we onderscheid in de **IJ** matrix van de analyse- en synthese bank, zie Formule (7.8) en (7.9). Van deze twee matrices sommen we aantal eigenschappen op:

- $\{-2,-1,0,1,2\} \in \mathbf{IJ}_{a,s}$
- $\max_{i,j}(|\mathbf{IJ}_{a,s}(i,j)|)=2$
- $\max_i(\sum_j |\mathbf{IJ}_a(i,j)|)=4$
- $\max_i(\sum_j \mathbf{IJ}_a(i,j)^2)=4$
- $\max_i(\sum_j |\mathbf{IJ}_s(i,j)|)=2$
- $\max_i(\sum_j \mathbf{IJ}_s(i,j)^2)=2$

We quantiseren $\mathbf{IJ}_a$ en $\mathbf{IJ}_s$ als volgt:

$$IJ_a \cdot 2^{N-2} \qquad IJ_s \cdot 2^{N-1} \tag{7.22}$$

zonder dat overflow kan optreden.

Volgens Formule (7.22) geldt $\max_i(\sum_j |\mathbf{IJ}_{(i,j)}|)=2^N$. Dus de 2N-bits uitgangssignalen kunnen door N bits verschuivingen geconverteerd worden naar N-bits signalen, zonder dat overflow optreedt.

7.8 Quantiseren/Schalen DCT matrix C

De DCT matrix **C** bestaat uit cosinus termen plus een vermenigvuldigingsfactor $\sqrt{(2/M)}$, zie Formule (4.3). Enkele eigenschappen van de DCT matrix **C** zijn:

- $\sum_j \mathbf{C}^2_{(i,j)}=\sqrt{(2/M)}\cdot M/2=\sqrt{(M/2)}, \quad 0\le i\le M-1$
- $\max_{i,j}(\mathbf{C}_{(i,j)})<\sqrt{(2/M)}$
- $\max_i(\sum_j |\mathbf{C}_{(i,j)}|)<\sqrt{(2/M)}\cdot M/\sqrt{2}=\sqrt{M}$

We quantiseren **C** m.b.v. de ℓ^2 -schaling als volgt:

$$\left[\sqrt{\frac{2}{M}}C\cdot 2^{N-1}\right]_R \tag{7.23}$$

er bestaat wel een kans op overflow, maar die is heel klein. Of we quantiseren **C** m.b.v. de ℓ^1 -schaling als volgt:

$$\left[\sqrt{\frac{1}{M}}C\cdot 2^{N-1}\right]_R \tag{7.24}$$

De 2N-bits uitgangssignalen kunnen door N bits verschuivingen geconverteerd worden naar N-bits signalen, zonder dat extra overflow optreedt.

7.9 MATLAB functies

- *flquant*: Quantisatie van een waarde, vector of matrix als input m.b.v. de floating-point getallen representatie

in: - x: input vector of matrix
- N: mantisse lengte
- Ex: maximale exponent grootte
- s: quantisatie vorm: s=0: Rounding, s=1: Magnitude Truncation, s=2: Value Truncation

uit: - y: gequantiseerde versie van x.

- *fxquant*: Quantisatie van een waarde, vector of matrix als input m.b.v. de fixed-point getallen representatie

in: - x: input vector of matrix.
- N: aantal bits.
- s: quantisatie vorm, zie *flquant*.
- in: als in='max' dan worden de inputs gerelateerd aan het maximum.

uit: - y: gequantiseerde versie van x.

- *qmat_mul*: Functie die de vermenigvuldiging van een vector of matrix met een andere vector of matrix waarbij gebruik gemaakt wordt van een floating- of fixed-point getallen representatie.

in: - A: input vector of matrix
- B: input vector of matrix
- status: status(1)=1: floating-point, status(1)=2: fixed-point
  status(2): quantisatie vorm, zie *flquant*
- N: mantisse lengte bij status(1)=1 en aantal bits bij status(1)=2.
- Ex:: maximale exponent grootte

uit: - C: uitgangsvector of matrix. (C=A*B)

- *qsum*: Het optellen van de elementen van een vector of de optelling van de elementen van de kolommen van een matrix waarbij gebruik gemaakt wordt van een floating- of fixed-point getallen representatie.

in: - X: input vector of matrix
- status: zie *qmat_mul*
- N: zie *qmat_mul*
- Ex: zie *qmat_mul*

uit: - S: uitgangswaarde of uitgangsvector.

Zie ook Appendix A.3 voor listing van deze functies.

Hoofdstuk 8

Simulatie van de CM Filter bank

8.1 Inleiding

In dit hoofdstuk wordt de cosinus gemoduleerde filterbank gesimuleerd m.b.v. MATLAB - functies. Van de functies die in dit hoofdstuk genoemd worden zijn de programma teksten in Appendix A.4. opgenomen. In volgende paragrafen komen een aantal resultaten van onderzoeken aan de orde. De onderzoeken bestaan uit het vinden van:

- Verschillen tussen twee- en vier vermenigvuldigers lattices bij een floating- en fixed-point architectuur.
- Verschillen tussen lattice- en transversale structuren uitgaande van lattice coëfficienten bij een floating- en fixed-point architectuur.
- Verschillen tussen een transversale structuur waarbij uitgegaan wordt van lattice coëfficiënten en prototypen ontworpen met *remez* bij een floating- en fixed-point architectuur.
- Strategieën en problemen bij de schaling en quantisatie van een filterbank met een lattice- en transversale structuur bij een fixed-point architectuur.

De resultaten van deze onderzoeken zullen in de verschillen paragrafen in dit hoofdstuk ter sprake komen. In hoofdstuk 9 zal een algemene conclusie getrokken worden van het totale onderzoek. De conclusie zal bestaan uit conclusies m.b.t. het ontwerpen van het prototype en de resultaten van hierboven genoemde onderzoeken.

8.2 Analyse- en synthese bank m.b.v. een lattice structuur

De twee MATLAB functies die de analyse- en synthese bank simuleren waarbij de polyfase componenten m.b.v. een lattice structuur geïmplementeerd zijn, zijn *abank_l* en *sbank_l*.

De inputs voor *abank_l* zijn:

- E: De lattice coëfficiënten
- M: Het aantal kanalen
- m: Aantal secties waaruit de polyfase componenten bestaan
- x: Ingangssignaal (vector)
- status: Een vector waarmee de status van de filterbank vastgelegd kan worden. Deze status bepaald de volgende vormen:
    - De soort van getallen-representatie (floating-point, fixed-point of de MATLAB

representatie). status(1) is dan respectievelijk 1, 2 en 0.
  - De quantisatie vorm (Rounding, Magnitude Truncation of Value Truncation).status(2) is dan respectievelijk 0,1 en 2.
  - Of de filter bank maximaal of helemaal niet gedecimeerd is. status(3) is dan respectievelijk 0 en 1.
  - Of de lattice structuur uitgevoerd wordt m.b.v. vier- of twee vermenigvuldigers lattices. status(4) is dan respectievelijk 0 en 1.
- N: In fixed-point getallen representatie aantal bits van de data. In floating point de mantisse-lengte. In MATLAB's getallen representatie speelt deze input geen rol.
- Ex: In floating point de maximale exponent. In fixed-point en MATLAB's getallen representatie speelt deze input geen rol.

En de uitgang is een matrix H. Als de input x een puls is (x=1) en de filterbank ingesteld is op een *niet-gedecimeerde filterbank* dan stellen de rijen van deze matrix H de impulsresponsies van de analyse filters $h_k[n]$ voor. M.b.v. de matrix H kan een frequentie spectrum bepaald worden d.m.v. de MATLAB functie *ploth*.

De inputs van *ploth* zijn:
- H: matrix met impulsresponsies van de analyse filters
- M: aantal kanalen
- m: aantal secties
- d: stopband grens factor

De uitgang As van *ploth* is de minimum stopband onderdrukking van de analyse filters.

In Appendix B.4 zijn voor enkele verschillende waarden van M en m een aantal frequentie-spectra afgebeeld. Het aantal lattice coëfficiënten voor filterbanken met deze M en m is 24. Als de MATLAB's getallen representatie gebruikt wordt dan is het verschil tussen filterbanken die gebruik maken van vier- of twee vermenigvuldigers lattices niet op te merken. Zie ook Tabel 8.1

De analyse filter responsies snijden elkaar op het 3 dB punt. Dit is een gevolg van de power complementaire eigenschap van de analyse filters:

$$\sum_{k=0}^{M-1} |H_k(e^{j\omega})|^2 = 1 \qquad \textbf{(8.1)}$$

De stopband onderdrukking, is vergeleken met het prototype een aantal dB lager. Dit is te wijten aan het feit dat de analyse filters een optelling is van twee verschoven prototypen waardoor de stopband onderdrukking maximaal 3 dB lager is. Dit maximum treedt op als de pieken in de stopband van twee verschoven prototypes samenvallen.

In combinatie met *abank_l* wordt *sbank_l* gebruikt. De inputs van *sbank_l* zijn:
- E: Dezelfde lattice coëfficiënten als bij *abank_l*
- M: Zie *abank_l*
- m: Zie *abank_l*
- H: De ingangssignalen wat de uitgangssignalen van de analyse bank (matrix H) zijn.

- status: Zie *abank_1*
- N: Zie *abank_1*
- Ex: Zie *abank_1*

De uitgang van *sbank_1* is uitgangssignaal y (vector) dat het geconstrueerde signaal is.

Om de performance van de filterbank te meten zijn een aantal maten daarvoor nodig. De gebruikte maten zijn de distorsie fouten: de maximum Aliasing Error (E_a) en de Peak-to Peak Reconstruction Error (Epp) en Mean Squared Error (MSE) van de input x en output y, waarbij de output berekend wordt m.b.v. een maximaal gedecimeerde filterbank.

De overdracht functie van de totale filterbank zonder decimators en interpolators, zie Figuur 5.1 en 5.3 wordt de distorsie functie T(z) genoemd en is gedefinieerd als

$$T(z) = \frac{1}{M} \cdot \sum_{k=0}^{M-1} H_k(z) F_k(z) \tag{8.2}$$

De response van de distorsie functie is constant (of 1) als geen amplitude distorsie optreedt. We definiëren de peak-to-peak van $M|T(e^{j\omega})|$ als de Peak-to-Peak Reconstruction Error E_{pp} :

$$E_{pp} = \delta_1 + \delta_2 \quad met \quad (1-\delta_1) \le M|T(e^{j\omega})| \le (1+\delta_2) \tag{8.3}$$

De uitgang van het gereconstrueerde signaal kan uitgedrukt worden in termen van de ingang.

$$y(z) = \frac{1}{M} f^T(z) H^T(z) x(z) = A^T(z) x(z) \tag{8.4}$$

met

$$x(z) = \begin{bmatrix} X(z) \\ X(zW) \\ \vdots \\ X(zW^{M-1}) \end{bmatrix} \quad H(z) = \begin{bmatrix} H_0(z) & H_1(z) & \cdots & H_{M-1}(z) \\ H_0(zW) & H_1(zW) & \cdots & H_{M-1}(zW) \\ \vdots & \vdots & \ddots & \vdots \\ H_0(zW^{M-1}) & H_1(zW^{M-1}) & \cdots & H_{M-1}(zW^{M-1}) \end{bmatrix} \tag{8.5}$$

$$f(z) = \begin{bmatrix} F_0(z) \\ F_1(z) \\ \vdots \\ F_{M-1}(z) \end{bmatrix} \qquad A(z) = \frac{1}{M} H(z) f(z) = \begin{bmatrix} A_0(z) \\ A_1(z) \\ \vdots \\ A_{M-1}(z) \end{bmatrix}$$

De maximum Aliasing Error wordt gevonden d.m.v. de Aliasing Error $E(\omega)$:

$$E(\omega) = \frac{1}{M}\sqrt{\sum_{l=1}^{M-1} |A_l(e^{j\omega})|^2}$$

zodat de maximale Aliasing Error E_a gelijk is aan:

$$E_a = \max_{\omega \in [0,\pi]} E(\omega)$$

De MATLAB functie die de distorsie fouten berekent en daarvan ook een frequentie spectra van laat zien, is de functie *distorsi*. De inputs zijn:
- E_P: De lattice coëfficiënten of de impuls response van het prototype afhankelijk van fbstatus(2) en fbstatus(3).
- M: Aantal Kanalen.
- m: Aantal secties.
- qstatus: Een vector waarmee de status van het quantisatie proces vastgelegd kan worden:
  - De soort van getallen representatie (floating-point, fixed-point of de MATLAB representatie). qstatus(1) is dan respectievelijk 1,2 en 0.
  - De quantisatie vorm (Rounding, Magnitude Truncation of Value Truncation). qstatus(2) is dan respectievelijk 0,1 en 2.
- fbstatus: Een vector waarmee de status van de filterbank vastgelegd kan worden:
  - Of de filter bank maximaal of helemaal niet gedecimeerd is. fbstatus(1) is dan respectievelijk 0 en 1.
  - Of de filterbank uitgevoerd met een lattice of transversale structuur wordt uitgevoerd. fbstatus(2) is dan respectievelijk gelijk aan 0 en 1.
  - Indien fbstatus(2)=0 dan bepaald fbstatus(3) of de lattice structuur uitgevoerd wordt m.b.v. vier- of twee vermenigvuldigers lattices. fbstatus(3) is dan respectievelijk 0 en 1. En indien fbstatus(2)=1 dan bepaald fbstatus(3) of de transversale structuur uit gaat van lattice coëfficiënten of een impuls response. fbstatus(3) is dan respectievelijk gelijk aan 0 en 1.

De uitgang van *distorsi* zijn de fouten:
- Ea: Maximale aliasing error,
- Epp: De Peak-to-Peak reconstruction error.

Tabel 8.1 geeft bij verschillende M en m voor zowel de vier als de twee vermenigvuldigers lattice de distorsie fouten. In deze tabel zijn ook de minimum stopband onderdrukking A_s en de stopband grens ω_s van het prototype gegeven. Ook de minimum stopband onderdrukking A_{sh} van de analyse filters is in deze tabel opgenomen.

Tabel 8.1 Distorsie fouten van filterbanken met lattices via E3									
M	m	proto		Filterbank met 4-verm. lattices			Filterbank met 2-verm. lattices		
		A_s [dB]	ω_s [rad]	A_{sb} [dB]	E_{pp} $\times 10^{-15}$	E_a $\times 10^{-15}$	A_{sb} [dB]	E_{pp} $\times 10^{-15}$	E_a $\times 10^{-15}$
6	4	42.66	0.5216	39.77	5.4401	1.3717	39.77	5.3291	1.3574
6	6	48.8	0.5216	46.70	5.6621	0.99846	46.70	5.7732	1.0147
6	8	57.46	0.5216	54.86	5.4401	2.2798	54.86	5.3291	2.2856
4	4	41.83	0.7854	39.07	5.5511	0.57925	39.07	5.7732	0.56856
4	8	55.08	0.7854	52.13	6.1062	2.2354	52.13	6.1062	2.3424
4	12	65.59	0.7854	63.58	5.2180	2.7238	63.58	4.9960	2.8150

Om de MSE te berekenen moet de vertraging van de totale filterbank bekend zijn. Deze vertraging is gelijk aan de lengte van de impulsresponsie van het prototype filter en die is gelijk aan 2mM. De MSE is gedefinieerd als:

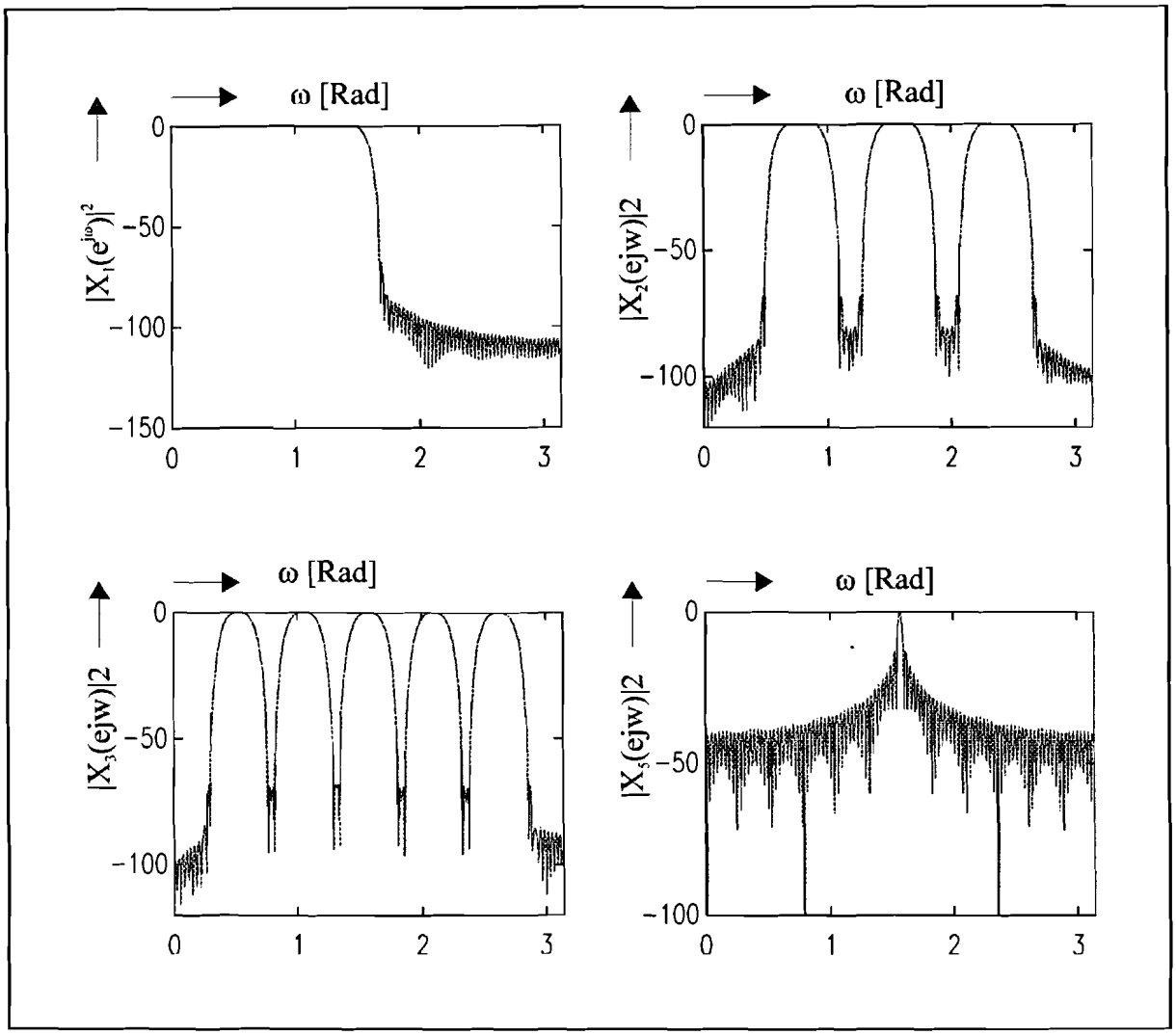
$$MSE = \frac{\sum_{i=0}^{L_y-1} (y[i]-x[i-2mM])^2}{\sum_{i=0}^{L_y-1} x[i-2mM]^2} \qquad x[i]=0 \quad als \quad \begin{cases} i<0 \\ i>L_x \end{cases} \tag{8.8}$$

met L_y en L_x respectievelijk de lengten van het uitgangssignaal y en ingangssignaal x.

De MATLAB functie die de MSE berekend is *mse*. De inputs van deze functie zijn:
- x: Input vector.
- y: Output vector.
- N: Vertraging filterbank.

En de uitgang is S die de MSE waarde geeft.

De MSE kan voor verschillende ingangssignalen berekend worden. De frequentie spectra van de te gebruiken ingangssignalen zijn in Figuur 8.1 opgenomen. Input x_4 is een delta-puls waarvan het frequentie spectra gelijk is aan 1. Tabel 8.2 en Tabel 8.3 geven voor respectievelijk de vier en twee vermenigvuldigers lattice de MSE waarden.



Figuur 8.1 Frequentie spectra ingangssignalen.

Tabel 8.2 MSE's van een Filterbank met 4-verm. lattices via E3 $\times 10^{-31}$

M	m	x_1	x_2	x_3	x_4	x_5
6	4	4.1489	3.4123	2.3981	5.1778	2.3520
6	6	2.4850	2.8084	2.5115	2.7827	1.5547
6	8	5.3556	4.1014	2.5688	2.9773	2.0754
4	4	0.46679	0.88484	1.1761	0.94447	0.35360
4	8	0.80664	0.81629	1.2189	1.0952	1.1078
4	12	0.84430	2.7915	1.0401	1.9492	1.9723

Tabel 8.3 MSE's van een Filterbank met 2-verm. lattices via E3 $\times 10^{-31}$						
M	m	x_1	x_2	x_3	x_4	x_5
6	4	2.7147	3.4231	2.4185	2.6508	1.1657
6	6	3.7443	3.1201	1.5147	2.1403	1.9139
6	8	2.2222	2.9946	4.8662	2.8324	1.2028
4	4	2.4815	3.1934	4.9475	0.98765	0.80152
4	8	1.2778	2.2591	1.5484	1.4626	1.8097
4	12	2.2646	2.4632	1.9415	1.4455	4.0317

De Mean Squared Error's zijn voor zowel de vier- als twee vermenigvuldigers lattice uit Tabel 8.2 en 8.3 verwaarloosbaar klein. De distorsie fouten uit Tabel 8.1 zijn ook heel klein vanwege de Perfecte Reconstructie.

8.3 Analyse- en synthese bank m.b.v. een transversale structuur

De twee MATLAB functies die de analyse- en synthese bank simuleren waarbij de polyfase componenten m.b.v. een transversale structuur geïmplementeerd zijn, zijn *abank_t* en *sbank_t*.

De inputs voor *abank_t* zijn:
- E: De lattice coëfficiënten of een impuls response van een FIR prototype filter ontworpen m.b.v. MATLAB functie *remezp*.
- M: Het aantal kanalen
- m: Aantal secties waaruit de polyfase componenten bestaan
- x: Ingangssignaal (vector)
- status: Een vector waarmee de status van de filterbank vastgelegd kan worden. Deze status bepaald de volgende vormen:
  - De soort van getallen-representatie (floating-point, fixed-point of de MATLAB representatie). status(1) is dan respectievelijk 0,1 en 2.
  - De quantisatie vorm (Rounding, Magnitude Truncation of Value Truncation).status(2) is dan respectievelijk 0,1 en 2.
  - Of de filter bank maximaal of helemaal niet gedecimeerd is. status(3) is dan respectievelijk 0 en 1.
  - Of de coëfficiënten in de transversale structuur bepaald worden door lattice coëfficiënten of door de impulsresponsie van een prototype filter (ontworpen m.b.v. *remezp*. status(4) is dan respectievelijk 0 en 1.
- N: In fixed-point getallen representatie aantal bits van de data. In floating point de

mantisse-lengte. In MATLAB's getallen representatie speelt deze input geen rol.
- Ex: In floating point de maximale exponent. In fixed-point en MATLAB's getallen representatie speelt deze input geen rol.

Ook hier geldt dat wanneer de input x een deltapuls is en de filterbank ingesteld is op een *niet-gedecimeerde filterbank* (status(3)=0) dan stellen de rijen van deze matrix H de impulsresponsies van de analyse filters $h_k[n]$ voor. In Appendix B.5 zijn voor enkele verschillende waarden van M en m een aantal frequentie-spectra afgebeeld, waarbij uitgegaan wordt uitgegaan van de impulsresponsie van een prototype (status(4)=1) ontworpen m.b.v. *remezp*. De frequentie spectra waarbij uitgegaan wordt van lattice coëfficiënten is identiek aan die van de frequentie spectra uit Appendix B.4. De minimum stopband onderdrukkingen van de analyse filters uit Appendix B.5 zijn ook lager dan die van het prototype filter. De reden hiervoor is dezelfde als in de vorige paragraaf.

In combinatie met *abank_t* wordt *sbank_t* gebruikt. De inputs van *sbank_t* zijn:
- E_P: Dezelfde lattice coëfficiënten of impuls responsie als bij *abank_t*
- M: Zie *abank_t*
- m: Zie *abank_t*
- H: De ingangssignalen wat de uitgangssignalen van de analyse bank (matrix H) zijn.
- status: Zie *abank_t*
- N: Zie *abank_t*
- Ex: Zie *abank_t*

De uitgang van *sbank_t* is uitgangssignaal y (vector) dat het gereconstrueerde signaal voorstelt.

Voor de dezelfde M en m als in Tabel 8.1 geeft Tabel 8.4 de distorsie fouten voor een filterbank met een transversale structuur waarbij uitgegaan wordt van lattice coëfficiënten E3 en van equiripple FIR prototype filter P (ontworpen m.b.v. *remezp*). In Tabel 8.4 zijn ook de minimum stopbandonderdrukking A_s en ω_s van het prototype P gegeven en ook de minimum stopbandonderdrukking van de analyse filters.

Tabel 8.4 Distorsie fouten van filterbanken met een transversale structuur.

M	m	Prototype via E3			Prototype P				
		A_{sh} [dB]	E_{pp} $\times 10^{-15}$	E_a $\times 10^{-15}$	A_s [dB]	ω_s [rad]	A_{sh} [dB]	E_{pp} $\times 10^{-2}$	E_a $\times 10^{-3}$
6	4	39.77	5.5511	1.3652	52.85	0.5093	49.87	1.7177	0.54506
6	6	46.70	5.7732	1.0063	53.12	0.4234	50.12	1.5282	0.92456
6	8	54.86	5.7732	2.2749	53.74	0.3866	50.76	1.5489	1.0341
4	4	39.07	5.4401	0.60055	52.51	0.7609	49.54	1.8067	0.26707
4	8	52.13	5.8842	2.2891	53.45	0.5768	50.48	1.5910	0.98087
4	12	63.58	5.4401	2.7124	54.3	0.5154	51.33	1.6044	1.1116

De distorsie fouten E_a en E_{pp} zijn bij de filterbanken waarbij uitgegaan is van prototype P beduidend groter dan de distorsie fouten van de filterbanken waarbij uitgegaan wordt van de lattice coëfficiënten. De oorzaak ligt in het feit dat een filterbank uitgaande van lattice coëfficiënten ontworpen is met de eis dat beide distorsie fouten nul zijn. Terwijl dat bij het ontwerpen van een filterbank uitgaande van het prototype P deze eis niet gesteld is. Het enigste dat gedaan is om deze fouten te beperken is de distorsie functie $|T(e^{j\omega})|$ te benaderen door een constante. Er is geen noemenswaardig verschil tussen de distorsie fouten voor de filterbank met een lattice structuur en een transversale structuur uitgaande van de lattice coëfficiënten. Appendix B.6 geeft van $1- |T(e^{j\omega})|$ en $E(\omega)$ het frequentie spectra voor M=4, m=12 en M=6, m=8. Uit deze figuren blijkt dat de grootste fouten gemaakt worden in de transitie gebieden van de analyse filters.

Tabel 8.5 en Tabel 8.6 geven de MSE's voor verschillende ingangssignalen van de filterbanken uitgaande van respectievelijk lattice coëfficiënten en prototype P.

Tabel 8.5 MSE's van filterbanken met een transversale uitgaande van E3. $\times 10^{-31}$

M	m	x_1	x_2	x_3	x_4	x_5
6	4	4.3159	4.2987	2.4220	2.8301	2.9608
6	6	5.9791	3.4681	2.2070	2.9239	3.2378
6	8	5.0679	5.9304	2.3101	15.795	5.6448
4	4	0.29907	0.81748	1.1415	0.93246	0.33004
4	8	0.48157	1.0629	0.82631	1.4135	0.36554
4	12	3.8745	5.6336	4.331	2.5647	5.3604

Tabel 8.6 MSE's van filterbanken met een transversale uitgaande van P. $\times 10^{-5}$

M	m	x_1	x_2	x_3	x_4	x_5
6	4	6.529	7.2425	4.0365	3.5022	1.3482
6	6	5.0472	4.5064	6.1500	1.6411	2.9191
6	8	4.8687	3.7754	7.3607	1.4826	3.5097
4	4	6.0519	4.5509	6.4754	5.6943	0.32048
4	8	2.5287	4.2866	1.4383	1.6566	1.79098
4	12	1.9860	3.3697	1.3718	1.2019	2.2157

De MSE's uit Tabel 8.5 zijn vergelijkbaar met die uit Tabel 8.2 en 8.3. Ook hier weer aanmerkelijke verschillen tussen de filterbanken waar uitgegaan is van lattice coëfficiënten en prototypes P. In volgende paragrafen gaan we onderzoeken hoe deze verschillen zullen zijn bij extra quantisatie (een floating- en fixed-point architectuur).

8.4 Floating-point architectuur

In deze paragraaf simuleren we de filterbank geïmplementeerd volgens een floating-point architectuur. Deze floating-point architectuur voeren we uit met een mantisse lengte N=4 en maximale exponent grootte Ex=4. Dus de te representeren waarde liggen tussen $-0.9999 \cdot 10^4$ en $0.9999 \cdot 10^4$ met als kleinst te representeren positieve waarde $0.0001 \cdot 10^{-4}$. In een binaire architectuur zou dit met ongeveer 16,5 bits te verwezenlijken zijn. Meeste floating point DSP's werken met meer bits (24 of 32). Om toch voor N=4 en Ex=4 te kiezen is tweeledig. Ten eerste zijn de meeste fixed-point DSP's 16 bits, waardoor verschillen tussen een fixed- en floating-point architectuur beter tot zijn recht komt. En ten tweede worden de verschillen bij verschillende implementatie (vier- en twee-vermenigvuldigers lattice en transversaal polyfase structuren) groter.

In deze paragraaf voeren we dezelfde simulaties uit als in vorige paragraaf. We zullen de diverse tabellen en figuren niet zo uitvoerig introduceren als in paragraaf 8.2 en 8.3.

In Tabel 8.1 waren er geen noemenswaardige verschillen tussen een filterbank met vier- of twee vermenigvuldigers lattices. Tabel 8.7 geeft wel verschillen bij extra floating-point quantisatie. Ondanks de extra quantisatie blijft de filterbank met vier vermenigvuldigers lattices redelijke stopband onderdrukkingen van de analyse filters hebben, terwijl dit niet bij de twee vermenigvuldigers lattices het geval is. Met name de filterbanken met 6 kanalen geven veel verschillen, vooral in de stopband onderdrukking van de analyse filters met M=6 en m=8. De reden hiervoor is dat de coëfficiënten in een twee vermenigvuldigers lattice structuur (niet te verwarren met de lattice coëfficiënten) in absolute zin niet begrensd zijn door 1, zoals bij de vier-vermenigvuldigers lattice. De coëfficiënten in een twee-vermenigvuldigers lattice kunnen door de deling $\cos\theta/\sin\theta$ heel groot of heel klein zijn, waardoor grote verschillen in coëfficiënten kunnen ontstaan. In paragraaf 8.3 zullen we hierop uitvoerig terug komen.

Tabel 8.7 Distorsie fouten van filterbanken met lattices via E3 met een Floating-point architectuur met mantisse lengte N=4 en maximale exponent Ex=4

M	m	4-mul			2-mul		
		A_{sh} [dB]	E_{pp} $\times 10^{-4}$	E_a $\times 10^{-4}$	A_{sh} [dB]	E_{pp} $\times 10^{-4}$	E_a $\times 10^{-4}$
6	4	39.67	4.1303	1.2832	39.72	4.2940	1.0228
6	6	46.24	11.703	1.9457	44.07	36.008	6.4644
6	8	52.83	13.249	2.1526	44.00	157.83	24.560
4	4	39.03	4.3751	1.5707	39.09	5.1509	1.0505
4	8	51.74	11.671	1.7167	51.70	12.713	2.4851
4	12	59.64	10.676	3.0692	59.12	11.520	2.9643

Tabel 8.8 is vergelijkbaar met tabel 8.4 uit paragraaf 8.3. In tabel 8.8 vallen een aantal dingen op. Ten eerste is voor M=6 en m=8 de stopband onderdrukking A_{sh} van de analyse filters uitgaande van E3 beter dan die van de filterbank zonder quantisatie. De oorzaak ligt waarschijnlijk in feit dat bij quantisatie polen verschuiven. Dit heeft tot gevolg dat pieken in de stopband ook verschuiven. In paragraaf 8.1 hadden we opgemerkt dat de analyse filters een optelling waren van twee verschoven prototype filters en dat daardoor A_{sh} kleiner is dan de stopband onderdrukking van het prototype. Door de verschuiving van de polen door het quantisatie proces is het goed mogelijk dat de pieken van de verschoven prototypes 'minder' samenvallen.

Het tweede punt is dat A_{sh}'s uit Tabel 8.7 allemaal kleiner zijn dan de overeenkomstige uit Tabel 8.8.

Ten derde, de distorsie fouten uit Tabel 8.7 verschillen vergeleken met die uit Tabel 8.8 uit de kolom 'via E3' niet veel. Dit hadden we eigenlijk niet verwacht omdat de filterbanken die gebruik maken van lattices betere PR-eigenschappen zouden hebben dan filterbanken met een transversale structuur, omdat een lattice structuur power complementair blijft ondanks quantisatie van de coëfficiënten. Een verklaring hiervoor is dat de tussen resultaten bij de implementatie van de transversale structuur in de accumulator (bijvoorbeeld 32 bits) blijft terwijl de tussen resultaten bij de implementatie van de lattices terug geschreven worden in het geheugen (16 bits). Dus het dynamisch bereik van de tussen resultaten in een transversale structuur is twee keer zo groot als bij een lattice structuur.

Het vierde punt dat opvalt is dat de distorsie fouten van de filterbank uitgaande van het prototype P niet veel verschillen vergeleken met die uit Tabel 8.4.

Het laatste wat we opmerken is het verschil in de distorsie fouten van de filter banken met een transversale via E3 en P. Dit verschil is in Tabel 8.8 minder groot dan in Tabel 8.4.

Tabel 8.8 Distorsie fouten van filterbanken met een transversale structuur met een Floating-point architectuur met mantisse lengte N=4 en maximale exponent Ex=4

M	m	via E3			via P		
		A_{sh} [dB]	E_{pp} $\times 10^{-4}$	E_a $\times 10^{-4}$	A_{sh} [dB]	E_{pp} $\times 10^{-2}$	E_a $\times 10^{-3}$
6	4	39.78	5.0667	0.81557	49.36	1.7385	0.54478
6	6	46.43	10.847	1.7121	49.72	1.5438	0.93472
6	8	54.33	15.562	1.5281	50.28	1.6001	1.0686
4	4	39.00	5.0826	1.1537	49.31	1.8350	0.27126
4	8	51.76	8.2554	0.72458	50.40	1.6308	1.0174
4	12	61.69	16.803	0.81983	50.88	1.6122	1.0632

Appendix B.7 geeft de frequentie spectra van de analyse filters van M=6 en m=8 van filterbanken met lattice structuren waarbij uitgegaan wordt van vier- of twee vermenigvuldigers lattices en van filterbanken met transversale structuren waarbij uitgegaan wordt van de lattice coëfficiënten E3 of van het prototype filter P. En Appendix B.8 geeft dat voor M=4 en m=12. Appendix B.9 en Appendix B.10 geven de spectra van de distorsie functies en aliasing error functies voor dezelfde situatie als die respectievelijk voor Appendix B.7 en B.8 gelden.

Tabellen 8.9 t/m 8.12 zijn respectievelijk onder de zelfde omstandigheden als de tabellen 8.2, 8.3, 8.5 en 8.6 afgezien van de extra floating-point quantisatie.

Tabel 8.9 MSE's van filterbanken met 4 vermenigvuldigers lattices E3 met een Floating-point architectuur met mantisse lengte N=4 en maximale exponent Ex=4

M	m	$x_1 \times 10^{-6}$	$x_2 \times 10^{-6}$	$x_3 \times 10^{-6}$	$x_4 \times 10^{-8}$	$x_5 \times 10^{-5}$
6	4	3.2797	4.2406	3.7831	7.4745	3.7085
6	6	4.6921	7.4536	4.9581	105.98	4.2115
6	8	6.2138	8.1143	6.3677	86.424	0.88794
4	4	3.1510	5.8887	4.4341	4.0000	2.3682
4	8	5.2136	6.3984	6.9012	106.29	0.37336
4	12	5.3848	8.1540	6.5503	10.748	0.76112

Tabel 8.10 MSE's van filterbanken met 2 vermenigvuldigers lattices E3 met een Floating-point architectuur met mantisse lengte N=4 en maximale exponent Ex=4

M	m	x_1	x_2	x_3	x_4	x_5
6	4	$3.8031 \cdot 10^{-6}$	$4.6560 \cdot 10^{-6}$	$4.0273 \cdot 10^{-6}$	$5.2477 \cdot 10^{-8}$	$3.6011 \cdot 10^{-5}$
6	6	$3.6735 \cdot 10^{-4}$	$3.9437 \cdot 10^{-4}$	$6.3922 \cdot 10^{-4}$	$5.5178 \cdot 10^{-8}$	$2.1822 \cdot 10^{-2}$
6	8	$5.1795 \cdot 10^{-3}$	$3.3524 \cdot 10^{-3}$	$5.6578 \cdot 10^{-3}$	$1.3287 \cdot 10^{-4}$	$3.4002 \cdot 10^{-1}$
4	4	$3.6214 \cdot 10^{-6}$	$6.0152 \cdot 10^{-6}$	$4.4405 \cdot 10^{-6}$	$4.0000 \cdot 10^{-8}$	$4.9460 \cdot 10^{-5}$
4	8	$1.0059 \cdot 10^{-5}$	$1.7193 \cdot 10^{-5}$	$1.4002 \cdot 10^{-5}$	$4.0000 \cdot 10^{-8}$	$1.3471 \cdot 10^{-5}$
4	12	$9.9153 \cdot 10^{-6}$	$1.9734 \cdot 10^{-5}$	$1.8431 \cdot 10^{-5}$	$9.9942 \cdot 10^{-7}$	$2.9078 \cdot 10^{-5}$

Tabel 8.11 MSE's van filterbanken met een transversale uitgaande van E3 met een Floating-point architectuur met mantisse lengte N=4 en maximale exponent Ex=4

M	m	$x_1 \times 10^{-6}$	$x_2 \times 10^{-6}$	$x_3 \times 10^{-6}$	$x_4 \times 10^{-8}$	$x_5 \times 10^{-5}$
6	4	1.4656	2.2411	2.0843	4.0000	3.6434
6	6	1.4851	2.2971	2.0870	4.0000	3.9020
6	8	1.7181	2.4500	2.1341	5.3042	4.4636
4	4	1.4222	2.6113	2.9278	5.1130	4.6885
4	8	1.3957	2.3076	2.3550	5.0363	5.3950
4	12	1.3621	2.4479	3.3890	5.1664	0.36350

Tabel 8.12 MSE's van filterbanken met een transversale uitgaande van P met een Floating-point architectuur met mantisse lengte N=4 en maximale exponent Ex=4

M	m	$x_1 \times 10^{-5}$	$x_2 \times 10^{-5}$	$x_3 \times 10^{-5}$	$x_4 \times 10^{-5}$	$x_5 \times 10^{-5}$
6	4	6.5163	7.4024	4.1627	3.2592	1.3982
6	6	5.084	4.9088	6.3987	1.6130	2.0573
6	8	5.0794	4.2858	7.7734	1.7910	4.2513
4	4	6.0743	4.7666	6.1261	5.1644	0.86630
4	8	2.5387	4.4959	1.6557	1.4535	2.0101
4	12	2.1742	3.7061	1.6237	1.2997	2.8984

Als we Tabel 8.9 met 8.10 met elkaar vergelijken valt het op dat de MSE's in Tabel 8.10 vooral in de rijen M=6 (en vooral m=6 en m=8) veel hoger zijn dan de verwachten waarde uit Tabel 8.9. De reden voor dit verschil is gelijk aan de verklaring voor het verschil in stopbandonderdrukking van een filterbank met vier- en twee vermenigvuldigers lattices in Tabel 8.7, namelijk dat door de deling $\cos\theta/\sin\theta$ de coëfficiënten in lattice structuur niet meer begrensd zijn door 1. Voor M=4 zijn de MSE's uit Tabel 8.9 en 8.10 ongeveer gelijk.

De MSE's uit Tabel 8.11 zijn uiteraard groter dan die uit Tabel 8.5. Ondanks de extra quantisatie blijven de MSE's uit Tabel 8.12 ongeveer gelijk aan die uit Tabel 8.6. Dit heeft ook tot gevolg dat het verschil uitgedrukt in de MSE's tussen de filterbanken met een lattice structuur en een transversale structuur uitgaande van P kleiner is.

Vergelijken we Tabel 8.9 met 8.11 dan zouden we verwachten dat resultaten uit Tabel 8.11 slechter zouden zijn dan die uit Tabel 8.9, omdat een lattice structuur beter bestand is tegen quantisatie effecten. Dit is niet het geval. De reden hiervoor is de dat de tussen resultaten in een transversale structuur niet naar het geheugen geschreven hoeven te worden, waardoor het dynamisch bereik van de tussen resultaten groter is.

8.5 Fixed-point architectuur

In hoofdstuk 7 zijn een aantal suggesties gedaan hoe de filter banken gequantiseerd en geschaald moeten worden. Met deze suggesties hebben we de filterbank met vier vermenigvuldigers lattice structuren gesimuleerd. We kwamen al gauw tot de conclusie dat als overflow optreedt dit tot rampzalige gevolgen leidt. Niet alleen de MSE's van diverse ingangssignalen met hun uitgangssignalen zijn dramatisch maar ook de stopband onderdrukking van de analyse filters, zie Appendix B.11. Deze Figuur geeft de analyse filter voor M=4 en m=4 met 1 en 2 overflows.

De totale filter bank kan gezien worden als een systeem dat op gesplitst kan worden in deel systemen De analyse- en synthese filter bank kunnen in modules opgedeeld worden. De eerste stap die we zetten is het gedeelte onderzoeken dat alle filter banken gemeen hebben. Dat is de cosinus modulatie matrix **T** en de getransponeerde daarvan $\mathbf{T}^T$. Deze cosinus modulatie matrix kan uitgeschreven worden in de DCT-IV matrix **C** en een zogenaamde **IJ** matrix voor de analyse- en synthese bank. De aan een schakeling van deze matrices vormt $\mathbf{IJ_sCCIJ_a}$ zie paragraaf 8.5.1. In paragraaf 8.5.2 en 8.5.3 worden de quantisatie/schaling van de polyfase componenten voor respectievelijk de lattice en transversale structuur onderzocht. In deze paragrafen worden ook de resultaten van de in die paragraaf besproken filter bank getoond.

8.5.1 Quantisatie/Schaling $\mathbf{IJ_sCCIJ_a}$

We schakelen de matrices $\mathbf{IJ_s}$, **C**, **C** en $\mathbf{IJ_a}$ aan elkaar en voeren aan de ingangen van $\mathbf{IJ_a}$ random signalen toe waarvan de waarden uniform verdeeld zijn tussen -2^{N-1} en 2^{N-1}. Het uitgangspunt is om nu de uitgangssignalen van elke matrix het dynamisch bereik zo groot mogelijk te maken zonder dat er overflow optreedt.

Volgens onderstaande quantisatie en schaling geeft $\mathbf{IJ_sCCIJ_a}$ de beste resultaten.

- Quantiseer $\mathbf{IJ_s}$ en $\mathbf{IJ_a}$ volgens Formule (7.22)
- Quantiseer **C** volgens:

$$[C \cdot 2^{N-1}]_R \tag{8.9}$$

- Schaal de uitgang van $\mathbf{IJ_a}$ d.m.v. het verschuiven van N bits.
- Schaal de uitgang van $\mathbf{IJ_s}$ d.m.v. het verschuiven van N-1 bits.
- Schaal de uitgang van **C** in zowel de analyse- als synthese bank d.m.v. het verschuiven van N-1 bits.

8.5.2 Quantisatie/Schaling polyfase componenten (lattice)

De coëfficiënten uit een vier- en twee vermenigvuldigers lattices structuur quantiseren we als volgt:

$$Q[R_i(j)] = \left[\frac{R_i(j)}{\max\{|R_i|\}} \cdot 2^{N-1} \right]_R \quad 0 \le i \le m-1 \quad 0 \le j \le 2\left\lfloor \frac{M}{2} \right\rfloor \tag{8.10}$$

met $R_i = \{\sin\theta_{0,i}, \cos\theta_{0,i}, \cdots, \cos\theta_{\lfloor M/2 \rfloor,i}\}$, zoals we al in paragraaf 7.6.1 en 7.6.2 voorstelden. Na diverse experimenten uitgevoerd te hebben bleek de volgende schaling van de uitgangssignalen van de lattices in de meeste gevallen de beste resultaten te hebben.

Schaling van de analyse filter bank:
- Schaal de uitgang van alle secties 0 d.m.v. het verschuiven van N-1 bits.
- Schaal de uitgang van alle secties 1 d.m.v. het verschuiven van N bits.
- Schaal de uitgang van alle secties 2 t/m m-1 d.m.v. het verschuiven van N-1 bits.

Schaling van de synthese filter bank:
- Schaal de uitgang van alle secties d.m.v. het verschuiven van N-1 bits.
- De uitgangen van de polyfase structuur schalen we d.m.v. het verschuiven van N bits. In de synthese bank worden deze uitgangen ten opzichte van elkaar vertraagd en bij elkaar opgeteld. Dit geldt eigenlijk alleen maar voor een filterbank die niet gedecimeerd is. Want in een gedecimeerde filterbank is deze optelling overbodig. In de niet gedecimeerde filterbank worden de N-bits uitgangen van de polyfase structuur geconverteerd naar 2N-bits en opgeteld waarna het resultaat van na M-1 optellingen geschaald wordt d.m.v. N bits te verschuiven.

De stopband onderdrukking van de analyse filters en distorsie fouten van een filterbank gequantiseerd en geschaald volgens bovenstaande zijn in Tabel 8.13 gegeven. In tabellen 8.14 en 8.15 zijn de MSE's voor diverse ingangssignalen gegeven.

Tabel 8.13 Distorsie fouten van filterbanken met lattices via E3 met een Fixed-point architectuur met N=16 bits.							
M	m	4-mul			2-mul		
		A_{sh} [dB]	E_{pp} $\times 10^{-4}$	E_a $\times 10^{-4}$	A_{sh} [dB]	E_{pp}	E_a
6	4	39.73	8.2705	0.85899	39.76	$9.4143 \cdot 10^{-4}$	$7.6653 \cdot 10^{-5}$
6	6	46.34	15.406	1.3765	31.72	$6.8604 \cdot 10^{-2}$	$7.0792 \cdot 10^{-3}$
6	8	53.64	11.863	1.3604	26.2	$1.2747 \cdot 10^{-1}$	$1.5699 \cdot 10^{-2}$
4	4	38.83	6.0911	1.2831	39.13	$9.8858 \cdot 10^{-3}$	$1.0234 \cdot 10^{-4}$
4	8	50.75	7.3640	1.4147	50.67	$2.6126 \cdot 10^{-3}$	$2.9341 \cdot 10^{-4}$
4	12	60.71	11.998	1.3469	61.22	$1.3439 \cdot 10^{-3}$	$1.6370 \cdot 10^{-4}$

Tabel 8.14 MSE's van filterbanken met 4 vermenigvuldigers lattices E3 met een Fixed-point architectuur met N=16 bits.						
M	m	$x_1 \times 10^{-7}$	$x_2 \times 10^{-7}$	$x_3 \times 10^{-7}$	$x_4 \times 10^{-7}$	$x_5 \times 10^{-7}$
6	4	8.5054	9.7143	11.260	1.7629	0.94225
6	6	12.680	11.177	14.878	2.7514	1.0651
6	8	12.223	11.082	13.326	4.4736	1.6507
4	4	8.4340	5.7339	10.691	0.69071	0.24964
4	8	9.9942	9.7135	12.451	4.0316	0.24332
4	12	12.579	9.6216	9.9924	2.7082	0.51361

De resultaten van een filter bank met vier vermenigvuldigers per lattice bij een fixed-point architectuur uit Tabel 8.13 en 8.14 zijn vergelijkbaar met de floating-point Tabellen 8.7 en 8.9.

Tabel 8.15 MSE's van filterbanken met 2 vermenigvuldigers lattices E3 met een Fixed-point architectuur met N=16 bits.

M	m	x_1	x_2	x_3	x_4	x_5
6	4	$0.98402 \cdot 10^{-6}$	$0.79322 \cdot 10^{-6}$	$0.93123 \cdot 10^{-6}$	$1.2249 \cdot 10^{-7}$	$0.45515 \cdot 10^{-6}$
6	6	$1.4040 \cdot 10^{-1}$	$1.6792 \cdot 10^{-1}$	$2.0017 \cdot 10^{-1}$	$2.500 \cdot 10^{-1}$	$5.0000 \cdot 10^{-3}$
6	8	1	1	1	1	1
4	4	$0.77924 \cdot 10^{-6}$	$0.78249 \cdot 10^{-6}$	$0.88177 \cdot 10^{-6}$	$0.91503 \cdot 10^{-7}$	$6.000 \cdot 10^{-8}$
4	8	$9.0263 \cdot 10^{-6}$	$9.4386 \cdot 10^{-6}$	$9.5841 \cdot 10^{-6}$	$38.916 \cdot 10^{-7}$	$0.44813 \cdot 10^{-6}$
4	12	$3.2199 \cdot 10^{-6}$	$2.9755 \cdot 10^{-6}$	$3.4615 \cdot 10^{-6}$	$7.8363 \cdot 10^{-7}$	$0.31896 \cdot 10^{-6}$

In de tabellen 8.13 en 8.15 vallen de dramatische resultaten van de filterbanken met 6 kanalen en twee-vermenigvuldigers lattices op. Voor M=6 en m=8 en m=6 zijn in Appendix B.12 de frequentie spectra van de analyse filters te zien. In paragraaf 7.6.2 hadden we al opgemerkt dat het goed fout kan gaan bij de twee-vermenigvuldigers lattices. Om dit probleem wat duidelijker te maken onderzoeken we lattice coëfficiënten en coëfficiënten uit de lattice structuur van de filterbank met M=6 en m=8 en vergelijken die met die van de filterbank met M=4 en m=12. De coëfficiënten horend bij M=4 en m=12 verkregen d.m.v. de cosinus gedeeld door de sinus van de lattice coëfficiënten zijn:

Tabel 8.16 De coëfficiënten in een twee vermenigvuldigers lattice voor M=4 en m=12.

$\cos(\theta_{i,j})/\sin(\theta_{i,j})$		j										
	S_i	1	2	3	4	5	6	7	8	9	10	11
i 0	$4.1723 \cdot 10^{-1}$	$9.5218 \cdot 10^{-2}$	$9.6278 \cdot 10^{-2}$	$5.2992 \cdot 10^{-2}$	$-3.482 \cdot 10^{-1}$	$-3.150 \cdot 10^{-1}$	$6.6791 \cdot 10^{-1}$	$-4.847 \cdot 10^{-2}$	$6.3577 \cdot 10^{-1}$	$6.4353 \cdot 10^{-2}$	-1.0867	$-1.543 \cdot 10^{-1}$
i 1	$7.2322 \cdot 10^{-1}$	$1.4950 \cdot 10^{-1}$	$-3.278 \cdot 10^{-1}$	$-2.355 \cdot 10^{-1}$	$9.9503 \cdot 10^{-2}$	$-3.732 \cdot 10^{-2}$	$-2.987 \cdot 10^{-2}$	$-4.171 \cdot 10^{-2}$	$5.9922 \cdot 10^{-1}$	$1.2759 \cdot 10^{-1}$	$-3.714 \cdot 10^{-1}$	$-6.236 \cdot 10^{-2}$

$\cos(\theta_{0,0}) \cdot S_0 = 0.1631$, $\sin(\theta_{0,0}) \cdot S_0 = 0.3840$, $\cos(\theta_{1,0}) \cdot S_1 = 0.529$, $\sin(\theta_{1,0}) \cdot S_1 = 0.4932$

Deze coëfficiënten in absolute zin zijn begrensd door 1.0867. In een vier-vermenigvuldigers lattice zijn de coëfficiënten begrensd door 1.

Voor M=6 en m=8 bekijken we ook de lattice coëfficiënten en andere coëfficiënten. De lattice coëfficiënten horend bij M=6 en m=8 zijn:

Tabel 8.17 Lattice coëfficiënten voor M=6 en m=8.									
$\theta_{i,j}$		j							
		0	1	2	3	4	5	6	7
i	0	1.9887	1.1710	1.5440	2.1484	2.2824	1.4857	$1.3177 \cdot 10^{-2}$	1.8053
	1	$7.3542 \cdot 10^{-1}$	1.4591	2.0070	2.5471	1.5719	1.0024	1.3772	1.4694
	2	$6.3089 \cdot 10^{-1}$	1.3606	1.8727	1.9344	1.8100	1.6827	1.2481	1.3697

De coëfficiënten horend bij M=6 en m=8 verkregen d.m.v. de cosinus van de lattice coëfficiënten zijn:

Tabel 8.18 De cosinus van de lattice coëfficiënten voor M=6 en m=8.									
$\cos(\theta_{i,j})$		j							
		0	1	2	3	4	5	6	7
i	0	$-4.0583 \cdot 10^{-1}$	$3.8922 \cdot 10^{-1}$	$2.6788 \cdot 10^{-2}$	$-5.4605 \cdot 10^{-1}$	$-6.5307 \cdot 10^{-1}$	$8.5030 \cdot 10^{-2}$	$9.9991 \cdot 10^{-1}$	$-2.323 \cdot 10^{-1}$
	1	$7.4155 \cdot 10^{-1}$	$1.1150 \cdot 10^{-1}$	$-4.2247 \cdot 10^{-1}$	$-8.2843 \cdot 10^{-1}$	$-1.1166 \cdot 10^{-3}$	$5.3828 \cdot 10^{-1}$	$1.9244 \cdot 10^{-1}$	$1.0118 \cdot 10^{-1}$
	2	$8.0751 \cdot 10^{-1}$	$2.0861 \cdot 10^{-1}$	$-2.9734 \cdot 10^{-1}$	$-3.5565 \cdot 10^{-1}$	$-2.3697 \cdot 10^{-1}$	$-1.1167 \cdot 10^{-1}$	$3.1709 \cdot 10^{-1}$	$1.9978 \cdot 10^{-1}$

De coëfficiënten horend bij M=6 en m=8 verkregen d.m.v. de sinus van de lattice coëfficiënten zijn:

Tabel 8.19 De sinus van de lattice coëfficiënten voor M=6 en m=8.									
$\sin(\theta_{i,j})$		j							
		0	1	2	3	4	5	6	7
i	0	$9.1395 \cdot 10^{-1}$	$9.2114 \cdot 10^{-1}$	$9.9964 \cdot 10^{-1}$	$8.3775 \cdot 10^{-1}$	$7.5730 \cdot 10^{-1}$	$9.9638 \cdot 10^{-1}$	$1.3177 \cdot 10^{-2}$	$9.7264 \cdot 10^{-1}$
	1	$6.7090 \cdot 10^{-1}$	$9.9376 \cdot 10^{-1}$	$9.0638 \cdot 10^{-1}$	$5.6010 \cdot 10^{-1}$	1.0000	$8.4276 \cdot 10^{-1}$	$9.8131 \cdot 10^{-1}$	$9.9487 \cdot 10^{-1}$
	2	$5.8986 \cdot 10^{-1}$	$9.7800 \cdot 10^{-1}$	$9.5477 \cdot 10^{-1}$	$9.3462 \cdot 10^{-1}$	$9.7152 \cdot 10^{-1}$	$9.9375 \cdot 10^{-1}$	$9.4840 \cdot 10^{-1}$	$9.7984 \cdot 10^{-1}$

De coëfficiënten horend bij M=6 en m=8 verkregen d.m.v. de cosinus gedeeld door de sinus van de lattice coëfficiënten zijn:

Tabel 8.20 De coëfficiënten in een twee vermenigvuldigers lattice voor M=6 en m=8.

$\cos(\theta_{i,j})/\sin(\theta_{i,j})$			j						
		S_i	1	2	3	4	5	6	7
i	0	$7.4600 \cdot 10^{-3}$	$4.2254 \cdot 10^{-1}$	$2.6797 \cdot 10^{-2}$	$-6.5181 \cdot 10^{-1}$	$-8.6236 \cdot 10^{-1}$	$8.5339 \cdot 10^{-2}$	$7.5885 \cdot 10^{1}$	$-2.3886 \cdot 10^{-1}$
	1	$4.1508 \cdot 10^{-1}$	$1.1220 \cdot 10^{-1}$	$-4.6611 \cdot 10^{-1}$	-1.4791	$-1.1166 \cdot 10^{-3}$	$6.3871 \cdot 10^{-1}$	$1.9610 \cdot 10^{-1}$	$1.0171 \cdot 10^{-1}$
	2	$7.8297 \cdot 10^{-1}$	$2.1330 \cdot 10^{-1}$	$-3.1143 \cdot 10^{-1}$	$-3.8053 \cdot 10^{-1}$	$-2.4392 \cdot 10^{-1}$	$-1.1237 \cdot 10^{-1}$	$3.3434 \cdot 10^{-1}$	$2.0389 \cdot 10^{-1}$

$\cos(\theta_{0,0}) \cdot S_0 = 3.0275 \cdot 10^{-1}$, $\sin(\theta_{0,0}) \cdot S_0 = 6.8181 \cdot 10^{-1}$, $\cos(\theta_{1,0}) \cdot S_1 = 3.078 \cdot 10^{-1}$, $\sin(\theta_{1,0}) \cdot S_1 = 2.7848 \cdot 10^{-1}$, $\cos(\theta_{2,0}) \cdot S_2 = 6.3226 \cdot 10^{-1}$, $\sin(\theta_{2,0}) \cdot S_2 = 4.6184 \cdot 10^{-1}$

De coëfficiënten zijn niet meer begrensd door ongeveer 1, zoals dat wel bij M=4 en m=12 het geval is. De waarde $\cos(\theta_{0,6})/\sin(\theta_{0,6}) = 75.885$ is relatief groot t.o.v. andere waarden met j=6. Tevens wordt $S_0 = 7.4600 \cdot 10^{-3}$ die heel klein t.o.v. andere S_i's. De signalen na de eerste sectie van lattice 0 (en dus ook lattice 5, vanwege de symmetrie) zijn daardoor ook klein en blijven na elke sectie in die lattice klein als geen extra schaling toegepast wordt voor alleen lattice 0 en 5. Dit geldt ook voor de lattice 1 en 2 (en 4 en 3) na de secties 6.

Als we de uitgangssignalen van sectie 0 van lattice 0 (en 5) en de uitgangssignalen van de secties 6 van lattices 1 en 2 (en 4 en 3) met dezelfde schalingsfactor (die een macht van 2 is) extra schalen dan ontstaan de resultaten uit Tabel 8.21. De resultaten in deze Tabel zijn stukken beter dan die uit Tabel 8.13 en 8.15, maar nog niet helemaal vergelijkbaar met resultaten uit de floating-point filter bank. Equivalent aan het schalen van de uitgangssignalen is het vergroten van de coëfficiënten. In dit geval doen we dit d.m.v. een vermenigvuldiging van 2^6, oftewel een verschuiving van 6 bits. In Appendix B.13 zijn de verbeterde frequentie spectra van de analyse filters voor M=6 en m=8 te zien.

Tabel 8.21 Resultaten van de verbeterde filterbank met M=6 en m=8 en een lattice structuur en een fixed-point architectuur.

A_{sb} [dB]	E_{pp} $\times 10^{-3}$	E_a $\times 10^{-4}$	MSE's $\times 10^{-5}$				
			x_1	x_2	x_3	x_4	x_5
48.54	3.1619	5.4605	2.5694	2.4804	2.4859	0.67349	0.45629

8.5.3 Quantisatie/Schaling polyfase componenten (transversaal)

We quantiseren de coëfficiënten uit de polyfase structuur volgens een ℓ^∞-schaling, zie Formule (7.21). De uitgangen van de polyfase structuur schalen we d.m.v. het verschuiven van N bits. In de synthese bank worden deze uitgangen ten opzichte van elkaar vertraagd en bij elkaar opgeteld. Dit geldt eigenlijk alleen maar voor een filterbank die niet gedecimeerd is want in een gedecimeerde filterbank is deze optelling overbodig. In de niet gedecimeerde filterbank worden de N-bits uitgangen van de polyfase structuur geconverteerd naar 2N-bits en opgeteld waarna het resultaat van na M-1 optellingen geschaald wordt d.m.v. N bits te verschuiven.

De resultaten van filterbanken met een transversale structuur worden in Tabellen 8.22 t/m 8.24 gegeven.

Tabel 8.22 Distorsie fouten van filterbanken met een transversale structuur met een Fixed-point architectuur met N=16 bits

M	m	via E3			via P		
		A_{sh} [dB]	E_{pp} $\times 10^{-4}$	E_a $\times 10^{-4}$	A_{sh} [dB]	E_{pp} $\times 10^{-2}$	E_a $\times 10^{-4}$
6	4	39.63	7.3936	0.89951	49.17	1.7012	6.5384
6	6	46.59	12.641	0.84923	49.60	1.5818	9.3777
6	8	53.66	11.159	1.2689	49.83	1.6120	11.145
4	4	38.96	6.9799	1.5663	49.20	1.8925	2.8465
4	8	51.28	9.4440	1.5537	49.68	1.6825	10.429
4	12	59.89	8.2877	2.0039	50.74	1.6476	11.903

Tabel 8.23 MSE's van filterbanken met een transversale structuur uitgaande van E3 met een Fixed-point architectuur met N=16 bits.

M	m	x_1 $\times 10^{-6}$	x_2 $\times 10^{-6}$	x_3 $\times 10^{-6}$	x_4 $\times 10^{-7}$	x_5 $\times 10^{-4}$
6	4	3.0301	4.9608	3.5724	1.9105	6.4327
6	6	3.2118	3.7833	3.6286	2.1621	2.4970
6	8	2.7915	3.0637	2.8277	2.9081	5.4167
4	4	2.4104	3.5230	3.9496	1.2455	7.3127
4	8	2.0939	3.8691	3.3043	0.85775	4.2500
4	12	2.3893	3.6628	3.2198	2.9806	2.7315

Tabel 8.24 MSE's van filterbanken met een transversale structuur uitgaande van P met een Fixed-point architectuur met N=16 bits.						
M	m	$x_1 \times 10^{-5}$	$x_2 \times 10^{-5}$	$x_3 \times 10^{-5}$	$x_4 \times 10^{-6}$	$x_5 \times 10^{-4}$
6	4	4.3254	4.1940	2.0613	7.2688	3.3333
6	6	4.2890	4.3172	3.7372	7.7780	0.75758
6	8	4.6311	3.7950	5.1004	9.0292	6.9431
4	4	2.9861	2.2260	3.1900	26.101	1.6111
4	8	2.0631	1.6110	1.5848	11.440	4.3444
4	12	1.9004	1.8705	1.5884	9.4719	3.4667

De resultaten uit Tabellen 8.22 t/m 8.24 zijn vergelijkbaar met de resultaten uit de Tabellen 8.8, 8.11 en 8.12 van de floating-point filter bank.

Hoofdstuk 9

Conclusies

De afstudeeropdracht bestond uit het ontwerpen en simuleren van een cosinus gemoduleerde filter bank met een willekeurig aantal kanalen M. De cosinus gemoduleerde filter bank kan geïmplementeerd worden m.b.v. een lattice structuur die automatisch tot perfecte reconstructie leidt. Er kan ook een transversale structuur gebruikt worden maar deze structuur leidt niet automatisch tot perfecte reconstructie.

Het ontwerpen van de filter bank bestaat in feite uit het ontwerpen van één FIR lineaire fase prototype filter met een lengte 2mM. Maken we gebruik van deze transversale structuur dan kunnen we conventionele ontwerp methoden zoals een equiripple design toepassen of gebruik maken van windows. De toepassing van een lattice structuur eist dat gebruik gemaakt wordt van optimalisatie methoden die veel tijd vergen. Zo'n optimalisatie methode verlangt ongeveer 50 keer zoveel tijd als een equiripple design.

De prototype filters ontwerpen m.b.v. een equiripple design resulteren in *betere* filters (betere stopband onderdrukking met een smallere transitiebandbreedte) dan de prototype filter ontworpen t.b.v. de lattice structuur. Maar zulke equiripple filters leiden niet tot perfecte reconstructie.

Bij de simulaties van de filter bank is uitgegaan van een lattice en transversale structuur bij een floating- en fixed-point architectuur. De lattice structuren kunnen uitgevoerd worden met een twee- en vier vermenigvuldigers lattice. In praktische uitvoeringen van de filter bank dient echter gebruik gemaakt te worden van een twee vermenigvuldigers lattice omdat de complexiteit van de filterbank dan gelijk is aan een filter bank met een transversale structuur.

In sommige situaties geeft de filter bank met een twee vermenigvuldigers lattice tegenvallende resultaten voor zowel een floating- als een fixed point architectuur. Maar met name de filter bank met een fixed-point architectuur geeft grote problemen bij de schaling van signalen en het quantiseren van coëfficiënten.

Theoretisch gezien zou de filter bank met een lattice structuur goede eigenschappen hebben m.b.t. de perfecte reconstructie bij de quantisatie van coëfficiënten. Dit blijkt teniet gedaan te worden indien de implementatie op een DSP uitgevoerd wordt. De DSP beschikt maar over één accumulator waardoor tussenresultaten in het geheugen weggeschreven dienen te worden. Het aantal bits van het tussen resultaat in het geheugen is de helft van het aantal bits waaruit het resultaat in de accumulator kan bestaan. De implementatie die gebruik maakt van een transversale structuur geeft dus ongeveer dezelfde resultaten als de implementatie die gebruikt maakt van een lattice structuur.

Tevens is een implementatie op een DSP van een filterbank die gebruik maakt van een lattice structuur complexer dan een die gebruik maak van de transversale structuur.

Indien het onderzoek voortgezet wordt dan zou het misschien verstandig zijn om het ontwerp van een prototype dat uitgaat van een transversale structuur te verbeteren omdat het blijkt dat de filterbank met een transversale structuur t.o.v. een lattice structuur enkele belangrijke voordelen heeft, zoals al was opgemerkt. Tevens zou daarbij ook onderzocht kunnen worden of het niet mogelijk is om niet-lineaire fase FIR filters als prototype filters te gebruiken.

Appendix A MATLAB functies

A.1 DCT-IV functies

```
function Y=dct_iv(M,x,s)
%
% Y=dct_iv(x,M,s), berekening van de 'Discrete Cosine Transform-IV'
%

% Deze functie berekend de DCT-IV op drie verschillende manieren:
%   - volgens een conventionele manier de input vector x
%     (ter lengte M) vermenigvuldigen met de C-matrix
%                                  ┌                    ┐
%        c(k,n)=√(2/M)· | cos((k-½)(n-½)/M) |   k,n=1      M
%                                  └                    ┘
%     het resultaat is de uitgangsvector Y. (s=0)
%
%   - de ingangsvector (ter lengte M) om vormen naar een nieuwe
%     ter lengte 2M volgens:
%                ┌            -jπ(n-½)/2M
%                | x(n) · e                                  n=1··M
%      x'(n)= |
%                |                      jπ(2M-n-½)/2M
%                | x(2M-n+1) · e                             n=M+1··2M
%                └
%     op x' een 2M-FFT uitvoeren en de het resultaat Y(k)
%     vermenigvuldigen volgens:
%                                -jπ(k-1)/2M
%       Y2(k)= Y(k) · ½ · e
%       (s=1)
%
%   - de ingangsvector (ter lengte M) om vormen naar een nieuwe
%                                -jπ(n-1)/M                  -jπ(n-1)/M
%     x'(n-1)=x(2(n-1)) · e          , x'(M+1-n)=x(2(n-1)+1) · e
%
%     voor n=1··M. Op x' een M-FFT uitvoeren en de het resultaat Y(k)
%     vermenigvuldigen volgens:
%                           -jπ(k-½)/2M
%       Y3(k)=Y(k) · e
%       (s=2)
%
%   - de recursieve methode (s=3)

f nargin==2
 s=0;
nd

f s==0
 C=zeros(M,M);
 for k=0:M-1,
   for l=0:M-1,
     C(k+1,l+1)=sqrt(2/M)*cos((k+1/2)*(l+1/2)*pi/M);
   end
 end
 Y=C*x;
lseif s==1
 Din1=zeros(2*M,M);
 for n=1:M
   Din1(n,n)=exp(-j*pi*(n-1/2)/(2*M));
   Din1(2*M-n+1,n)=exp(j*pi*(n-1/2)/(2*M));
 end
 Duit1=zeros(M,2*M);
 for k=1:M
   Duit1(k,k)=exp(-j*pi*(k-1)/(2*M))/2;
 end
 Y=sqrt(2/M)*real(Duit1*fft(Din1*x,2*M));
elseif s==2
 Din2=zeros(M,M);
 for n=1:fix(M/2)
   Din2(n,2*n-1)=1;
   Din2(M-n+1,2*n)=-1;
 end
 if rem(M,2)==1
   Din2((M+1)/2,M)=1;
 end
 Din3=zeros(M,M);
 for n=0:M-1
   Din3(n+1,n+1)=exp(-j*pi*n/M);
 end
 Duit2=zeros(M,M);
 for k=0:M-1
   Duit2(k+1,k+1)=exp(-j*pi*(k+1/2)/(2*M));
 end
 Y=sqrt(2/M)*real(Duit2*fft(Din3*Din2*x,M));
elseif s==3
 dctiv(M,x);
end


function Y=dctiv(M,x,s);
% Y=dctiv(M,x,s)
% Berekening van de DCT-IV volgens de recursieve methode
% M>=2
% Als s=1 dan wordt de input genormaliseerd, dit dient te gebeuren
% bij de eerste aanroep.

if nargin==3
 if s==1
  x=x*sqrt(2/M);
 end;
end;

if M==2
  Y=[cos(pi/8) cos((3*pi)/8) ; cos((3*pi)/8) cos((9*pi)/8)]*x;
else
  in1=zeros(M/2,M);
  in2=zeros(M/2,M);
  uit1=zeros(M,M);
  uit2=zeros(M,M);

  for i=1:M/2-1
    in1(i+1,2*i)=1;
    in1(i+1,2*i+1)=1;
  end
  in1(1,1)=1;

  for i=1:M/2
    in2(i,2*i-1)=1;
    in2(i,2*i)=1;
  end
```

```
uit1(M/2+1:M,1:M/2)=-fliplr(eye(M/2));
uit1(1:M/2,M/2+1:M)=eye(M/2);
uit1(M/2+1:M,M/2+1:M)=fliplr(eye(M/2));
uit1(1:M/2,1:M/2)=eye(M/2);

for i=0:M/2-1;
  uit2(i+1,i+1)=1/(2*cos((pi/(2*M))*(i+1/2)));
  uit2(M-i,M-i)=-1/(2*sin((pi/(2*M))*(i+1/2)));
end

Y(1:M/2,:)=dctii(M/2,in1*x);
Y(M/2+1:M,:)=dctiv(M/2,in2*x);
Y=uit2*uit1*Y;
nd



nction Y=dctii(M,x,s);
 Y=dctii(M,x)
 Berekening van de DCT-II volgens de recursieve methode
 M>=2

 M==2
Y=[1 1/sqrt(2);1 -1/sqrt(2)]*x;
se
in1=zeros(M/2,M);
in2=zeros(M/2,M);
uit2=zeros(M,M);

for i=1:M/2
  in1(i,2*i-1)=1;
end

if nargin==3
 if s==1
  in1(:,1)=in1(:,1)/sqrt(2);
 end
end

for i=1:M/2-1
  in2(i,i*2)=1;
  in2(i+1,i*2)=1;
end
in2(M/2,M)=1;

uit1=zeros(M/2,M/2);
for i=0:M/2-1;
  uit1(i+1,i+1)=1/(2*cos((pi/(2*M))*(2*i+1)));
end

uit2(1:M/2,1:M/2)=eye(M/2);
uit2(M/2+1:M,1:M/2)=fliplr(eye(M/2));
uit2(M/2+1:M,M/2+1:M)=-fliplr(eye(M/2));
uit2(1:M/2,M/2+1:M)=eye(M/2);

Y(1:M/2,:)=dctii(M/2,in1*x);
Y(M/2+1:M,:)=uit1*dctii(M/2,in2*x);
Y=uit2*Y;
nd
```

.2 Ontwerp functies

.2.1 Ontwerp functies t.b.v. lattice structuur

```
ction P0=Proto(E,M,m)
?0=Proto(E,M,m)

Proto geeft de impuls responsie af van het LDF FIR prototype
ilter dat gebruikt wordt bij 'cosine modulation filter banks'.
Dit filter is opgebouwd uit 2M polyfase componenten die
geimplementeerd zijn m.b.v. 2-kanaals lattices. De vector E
bevat de lattice coefficienten. M stelt het aantal kanalen
van de filterbank voor. En m het aantal secties per lattice.
E=[Θ(1,1),Θ(1,2),..,Θ(1,m),Θ(2,1),....,Θ(L,m)]
Θ(k,p) is de lattice coefficient van lattice k sectie p

=zeros(2*M,m);
em(M,2)==1,  % M oneven polyfase component G      en G
=fix(m/2);   % bestaan uit elk een delay.  (M-1)/2   (3M-1)/2
0((M+1)/2,K+1)=1/sqrt(2);
0((3*M+1)/2,m-K)=1/sqrt(2);

fix(M/2);
k=1:L,
=E((k-1)*m+1:k*m);
0(k,1)=cos(e(1));        % sectie 1
0(k+M,1)=sin(e(1));
r p=2:m              % sectie 2 t/m m
PP2=conv([0 1],P0(k+M,1:p-1));
P0(k+M,1:p)=sin(e(p))*P0(k,1:p)-cos(e(p))*PP2;
P0(k,1:p)=cos(e(p))*P0(k,1:p)+sin(e(p))*PP2;
nd
0(2*M+1-k,:)=fliplr(P0(k,:));
0(M+1-k,:)=fliplr(P0(M+k,:));

=reshape(P0,1,2*M*m)/(sqrt(2)*M);


ction O=obj1(E,M,m,d)
O=obj1(E,M,m,d)

O=obj1 berekend de de stopband energie voor een gegeven
lattice coefficienten vector E. M is het aantal kanalen en m
net aantal secties per lattice. d is een maat voor de stopband
grens: ws=π*(1+d)/(2*M)

=proto(E,M,m);  % Impuls responsie gegeven de lattice vector

2*m*M;          % Lengte prototype
=pi*(1+d)/(2*M); % Stopband grens

      -1                                    |    jw |2
PP0=P0(z  )*P0(z)  geëvalueerd op de eenheidscirkel is  |P0(e  )|


)=conv(P0,fliplr(P0));

  |π |    jw |2
O=|   |P0(e  )|  dw
  |
   π(1+d)/2M

PP0 kan geschreven worden als een som van cosinus termen
Integratie van PP0 geeft sinus termen

PP0(N)*(pi-ws);
i=1:N-1,
=O-PP0(N-i)*2*sin(ws*i)/i;
end


function [PR0,G]=obj2(E,M,m,d,L)
% [PR0,G]=obj2(E,M,m,d,L)
%
% obj2 geeft een vector PR0 ter lengte L af, die de waarde
%                  |    jw |
% van het prototype |P0(e  )| af geeft voor w=ws,ws+(π-ws)/L,....,π
% met ws=π(1+d)/2M. E is de vector met de lattice coefficienten
% waaruit het prototype P0 is uit opgebouwd. M is het aantal kanalen
% en m het aantal secties per lattice.

P0=proto(E,M,m);  % De symmetrische impulsresponsie van het prototype

N=2*m*M;          % Lengte van het prototype (is even)
ws=pi*(1+d)/(2*M); % Stopband grens

% De frequenties in stopband
w=ws:(pi-ws)/(L-1):pi;
%                                                    -jw(N-1)/2
% P0 kan uitgeschreven worden in cosinus termen vermenigvuligt met e
%

% PR0=abs(freqz(P0,1,w)) is langzamer
PR0=0;
for i=0:N/2-1,
  PR0=PR0+2*P0(N/2-i)*cos((1/2+i)*w);
end
PR0=abs(PR0);

G=[];  % geen constraints, zie minimax in de optimalisatie toolbox


function [E1,E2,E3]=est_p0(M,m,d,options,m1,K)
% [E1,E2,E3]=est_p0(M,m,d,options,m1,K)
%
% Deze functie berekend de lattice coefficienten van
% het FIR LDF prototype P0 dat gebruikt wordt door de
% cosinus modulated filterbank met M kanalen en m secties
% per lattice. De optimalisatie van de lattice coefficienten
% kan in één t/m drie stappen uitgevoerd worden.
% E2=est_p0(M,m,d) voert een optimalisatie uit m.b.v. een quasi-newton
% methode. Deze methode minimaliseerd de energie in de stopband.
% Als het aantal secties m groter is dan twee of drie dan kan een
% pre-optimalisatie uitgevoerd worden die uit gaat van m1 secties,
% [E1,E2]=est_p0(M,m,d,[],m1). E1 wordt als uitgangspunt voor de
% optimalisatie van E2. De pre-optimalisatie gebruikt ook de
% quasi-newton methode.
% Een laatste stap om de lattice coefficienten van de lattice met m
% secties te optimaliseren is een minimax methode met als uitgangspunt
% E2. Deze methode minimaliseerd het maximum in de stopband. K is het
% aantal functie evaluatie in stopband. Er zijn twee mogelijke functie
% aanroepen [E1,E2,E3]=est_p0(M,m,d,[],m1,K) waarbij alle drie stappen
% doorlopen worden en [E2,E3]=est_p0(M,m,d,[],[],K).
% d is een maat voor de stopband grens ws:  ws=π*(1+d)/2*M.
% gebruikt wordt voor quasi-newton optimalisatie.
% Zie voor foptions in de optimalisatie toolbox voor options.

options(1)=1;      % Informatie wordt getoond bij de minimalisatie
options(14)=400*m*fix(M/2); % bovengrens aantal iteraties verhogen
L=fix(M/2);

clc;
```

```
sp(' ');
isp('Uitvoeren van de minimalisatie');
isp('');

 nargin==4,
E1=[];
ni=initmin(M,m); %initialisatie van de lattice coefficienten
fprintf('Minimaliseren van de stopband energie met %2.0f secties\n\n',m);
E2=fminu('obj1',ini,options,[],M,m,d);
E3=[];
seif nargin==5,
ini=initmin(M,m1);
fprintf('Minimaliseren van de stopband energie met %2.0f secties\n\n',m1);
E1=fminu('obj1',ini,options,[],M,m1,d);
for i=1:L
 ini(m*(i-1)+1:m*(i-1)+m1)=E1(m1*(i-1)+1:m1*i);
 ini(m*(i-1)+m1+1:m*i)=ones(1,m-m1)*pi/2;
end
disp('');
fprintf('Minimaliseren van de stopband energie met %2.0f secties\n\n',m);
E2=fminu('obj1',ini,options,[],M,m,d);
E3=[];
se
f isempty(m1)==1,
 E1=[];
 ini=initmin(M,m);
 fprintf('Minimaliseren van de stopband energie met %2.0f secties\n\n',m);
 E2=fminu('obj1',ini,options,[],M,m,d);
lse
 ini=initmin(M,m1);
 fprintf('Minimaliseren van de stopband energie met %2.0f secties\n\n',m1);
 E1=fminu('obj1',ini,options,[],M,m1,d);
 for i=1:L
  ini(m*(i-1)+1:m*(i-1)+m1)=E1(m1*(i-1)+1:m1*i);
  ini(m*(i-1)+m1+1:m*i)=ones(1,m-m1)*pi/2;
 end
 disp('');
 fprintf('Minimaliseren van de stopband energie met %2.0f secties\n\n',m);
 E2=fminu('obj1',ini,options,[],M,m,d);
 disp('');
end
lisp('');
lisp('Laatste minimalisatie m.b.v. de minimax methode met als uitgangspunt');
lisp('de benadering van de lattice coefficienten gevonden d.m.v. het');
lisp('minimaliseren van de energie in de stopband');
lisp('');
E3=minimax('obj2',E2,options,[],[],[],M,m,d,K);
nd


nction [E2,E3,M,m]=extendp0(E,EE,M,m,d,options,K,m2,M1)
% [E2,E3,M,m]=extendp0(E,EE,M,m,d,options,K,m2,M1)
%

ptions(1)=1;       % Informatie wordt getoond bij de minimalisatie
ptions(14)=400*m*fix(M/2); % bovengrens aantal iteraties verhogen
=fix(M/2);


lc;
isp(' ');
isp('Uitvoeren van de minimalisatie');
isp('');

 nargin==9,
if rem(M,2)==0 % M even
  if strcmp(M1,'inc')
    M=M+1;
    if m2>m
      fprintf('Verlengen prototype tot %2.0f secties en 1 kanaal\n',m2);disp('');
      m1=m;m=m2;
      for i=1:L
        ini(m*(i-1)+1:m*(i-1)+m1)=E(m1*(i-1)+1:m1*i);
        ini(m*(i-1)+m1+1:m*i)=ones(1,m-m1)*pi/2;
      end
      if ~isempty(EE)
        for i=1:L
          ini1(m*(i-1)+1:m*(i-1)+m1)=EE(m1*(i-1)+1:m1*i);
          ini1(m*(i-1)+m1+1:m*i)=ones(1,m-m1)*pi/2;
        end
        disp('Minimaliseren stopband energie');
        E2=fminu('obj1',ini,options,[],M,m,d);
        disp('Minimax met als uitgangspunt vorige minimax resultaat');
        E3=minimax('obj2',ini1,options,[],[],[],M,m,d,K);
      else
        disp('Minimaliseren stopband energie');
        E2=fminu('obj1',ini,options,[],M,m,d);
        disp('Minimax met als uitgangspunt vorige berekening');
        E3=minimax('obj2',E2,options,[],[],[],M,m,d,K);
      end
    elseif m2==m
      disp('Aantal kanalen uitbreiden met 1 en aanpassen stopband grens');di
      disp('Minimaliseren stopband energie');
      E2=fminu('obj1',E,options,[],M,m,d);
      if isempty(EE)
        disp('Minimax met als uitgangspunt vorige berekening');
        E3=minimax('obj2',E2,options,[],[],[],M,m,d,K);
      else
        disp('Minimax met als uitgangspunt vorige minimax resultaat');
        E3=minimax('obj2',EE,options,[],[],[],M,m,d,K);
      end
    else
      disp('Aantal kanalen uitbreiden met 1');disp('');
      disp('Minimaliseren stopband energie');
      E2=fminu('obj1',E,options,[],M,m,d);
      if isempty(EE)
        disp('Minimax met als uitgangspunt vorige berekening');
        E3=minimax('obj2',E2,options,[],[],[],M,m,d,K);
      else
        disp('Minimax met als uitgangspunt vorige minimax resultaat');
        E3=minimax('obj2',EE,options,[],[],[],M,m,d,K);
      end
    end
  else
   clc;disp('input onjuist');pause(4);
  end
 else
  clc;disp('M is oneven en dus niet te verhogen');pause(4);
 end
elseif nargin==8
 if m2>m
  fprintf('Verlengen prototype tot %2.0f secties\n',m2);disp('');
  m1=m;m=m2;
  for i=1:L
   ini(m*(i-1)+1:m*(i-1)+m1)=E(m1*(i-1)+1:m1*i);
   ini(m*(i-1)+m1+1:m*i)=ones(1,m-m1)*pi/2;
  end
  if ~isempty(EE)
    for i=1:L
      ini1(m*(i-1)+1:m*(i-1)+m1)=EE(m1*(i-1)+1:m1*i);
      ini1(m*(i-1)+m1+1:m*i)=ones(1,m-m1)*pi/2;
    end
    disp('Minimaliseren stopband energie');
    E2=fminu('obj1',ini,options,[],M,m,d);
    disp('Minimax met als uitgangspunt vorige minimax resultaat');
    E3=minimax('obj2',ini1,options,[],[],[],M,m,d,K);
  else
    disp('Minimaliseren stopband energie');
    E2=fminu('obj1',ini,options,[],M,m,d);
    disp('Minimax met als uitgangspunt vorige berekening');
    E3=minimax('obj2',E2,options,[],[],[],M,m,d,K);
  end
 elseif m2==m
```

```
disp('Minimaliseren stopband energie');
E2=fminu('obj1',E,options,[],M,m,d);
if isempty(EE)
  disp('Minimax met als uitgangspunt vorige berekening');
  E3=minimax('obj2',E2,options,[],[],[],M,m,d,K);
else
  disp('Minimax met als uitgangspunt vorige minimax resultaat');
  E3=minimax('obj2',EE,options,[],[],[],M,m,d,K);
end
se
lc;disp('Onjuiste waarde voor m2');pause(4);
ıd
|


ction [As1,As2,As3,ws1,ws2,ws3]=plotp0(E1,E2,E3,M,m,d,m1)
 As1,As2,As3,ws1,ws2,ws3]=plotp(E1,E2,E3,M,m,d,m1)

Geeft de magnitude responsie in db als grafiek van het prototype.
De input is/zijn de lattice coefficienten E1, E2 en E3 van het
prototype. Als E1 E2 en/of E3=[] dan worden de daarbij horende
magnitude reponses niet afgebeeld. Het aantal secties van het proto-
type met lattice-coefficienten E1 is gelijk aan m1, en van E2 en E3 m.
d is een maat voor de stopband grens ws=π(1+d)/(2*M), met M het aantal
kanalen van de filterbank. Deze stopband grens wordt gebruikt voor de
berekening van de minimum stopband onderdrukking As1, As2 en As3 in db.


500;
s([0 pi -60 0]);

Berekenen magnitude responsie van de prototype.
-isempty(E1)
=proto(E1,M,m1);
H,w]=freqz(P,1,K);
1=20*log10(abs(H));
|

-isempty(E2)
=proto(E2,M,m);
H,w]=freqz(P,1,K);
2=20*log10(abs(H));
|

-isempty(E3)
=proto(E3,M,m);
H,w]=freqz(P,1,K);
3=20*log10(abs(H));
|

ge=round(K*((1+d))/(2*M));

plotten en berekenen stopband onderdrukking.
-isempty(E1) & ~isempty(E2) & ~isempty(E3),
-edge;while H1(i+1) < H1(i),i=i+1;end;
s1=-max(H1(i:K));
-edge;while H1(i)> -As1, i=i+1; end;
s1=w(i);
-edge;while H2(i+1) < H2(i),i=i+1;end;
s2=-max(H2(i:K));
-edge;while H2(i)> -As2, i=i+1; end;
s2=w(i);
-edge;while H3(i+1) < H3(i),i=i+1;end;
s3=-max(H3(i:K));
-edge;while H3(i)> -As3, i=i+1; end;
s3=w(i);
ot(w,H1,'-w',w,H2,'-g',w,H3,'-r');
le(['M=',num2str(M),' m=',num2str(m),' m1=',num2str(m1),..
                                        ' ws=',num2str(pi*(1+d)/(2*M))]);
xt(0.52,0.89,['E1: wit,  ws1=',num2str(ws1),' As1=',num2str(As1)],'sc');
xt(0.52,0.86,['E2: groen, ws2=',num2str(ws2),' As2=',num2str(As2)],'sc');
xt(0.52,0.83,['E3: rood,  ws3=',num2str(ws3),' As3=',num2str(As3)],'sc');
elseif ~isempty(E1) & ~isempty(E3),
  i=edge;while H1(i+1) < H1(i),i=i+1;end;
  As1=-max(H1(i:K));
  i=edge;while H1(i)> -As1, i=i+1; end;
  ws1=w(i);
  As2=[];
  i=edge;while H3(i+1) < H3(i),i=i+1;end;
  As3=-max(H3(i:K));
  i=edge;while H3(i)> -As3, i=i+1; end;
  ws3=w(i);
  plot(w,H1,'-w',w,H3,'-g');
  title(['M=',num2str(M),' m=',num2str(m),' m1=',num2str(m1),..
                                        ' ws=',num2str(pi*(1+d)/(2*M))]);
  text(0.52,0.89,['E1: wit,  ws1=',num2str(ws1),' As1=',num2str(As1)],'sc');
  text(0.52,0.83,['E3: rood,  ws3=',num2str(ws3),' As3=',num2str(As3)],'sc');
elseif ~isempty(E1) & ~isempty(E2),
  i=edge;while H1(i+1) < H1(i),i=i+1;end;
  As1=-max(H1(i:K));
  i=edge;while H1(i)> -As1, i=i+1; end;
  ws1=w(i);
  i=edge;while H2(i+1) < H2(i),i=i+1;end;
  As2=-max(H2(i:K));
  i=edge;while H2(i)> -As2, i=i+1; end;
  ws2=w(i);
  As3=[];
  plot(w,H1,'-w',w,H2,'-g');
  title(['M=',num2str(M),' m=',num2str(m),' m1=',num2str(m1),..
                                        ' ws=',num2str(pi*(1+d)/(2*M))]
  text(0.52,0.89,['E1: wit,  ws1=',num2str(ws1),' As1=',num2str(As1)],'sc');
  text(0.52,0.86,['E2: groen, ws2=',num2str(ws2),' As2=',num2str(As2)],'sc');
elseif ~isempty(E2) & ~isempty(E3),
  As1=[];
  i=edge;while H2(i+1) < H2(i),i=i+1;end;
  As2=-max(H2(i:K));
  i=edge;while H2(i)> -As2, i=i+1; end;
  ws2=w(i);
  i=edge;while H3(i+1) < H3(i),i=i+1;end;
  As3=-max(H3(i:K));
  i=edge;while H3(i)> -As3, i=i+1; end;
  ws3=w(i);
  plot(w,H2,'-w',w,H3,'-g');
  title(['M=',num2str(M),' m=',num2str(m),' m1=',num2str(m1), ..
                                        ' ws=',num2str(pi*(1+d)/(2*M))]
  text(0.52,0.86,['E2: groen, ws2=',num2str(ws2),' As2=',num2str(As2)],'sc');
  text(0.52,0.83,['E3: rood,  ws3=',num2str(ws3),' As3=',num2str(As3)],'sc');
elseif ~isempty(E1),
  i=edge;while H1(i+1) < H1(i),i=i+1;end;
  As1=-max(H1(i:K));
  i=edge;while H1(i)> -As1, i=i+1; end;
  ws1=w(i);
  As2=[];
  As3=[];
  plot(w,H1,'-w');
  title(['M=',num2str(M),' m=',num2str(m),' m1=',num2str(m1),..
                                        ' ws=',num2str(pi*(1+d)/(2*M))]);
  text(0.52,0.89,['E1: wit,  ws1=',num2str(ws1),' As1=',num2str(As1)],'sc');
elseif ~isempty(E3),
  As1=[];
  As2=[];
  i=edge;while H3(i+1) < H3(i),i=i+1;end;
  As3=-max(H3(i:K));
  i=edge;while H3(i)> -As3, i=i+1; end;
  ws3=w(i);
  plot(w,H3,'-w');
  title(['M=',num2str(M),' m=',num2str(m),' m1=',num2str(m1),..
                                        ' ws=',num2str(pi*(1+d)/(2*M))]
  text(0.52,0.83,['E3: rood,  ws3=',num2str(ws3),' As3=',num2str(As3)],'sc');
elseif ~isempty(E2),
  As1=[];
  i=edge;while H2(i+1) < H2(i),i=i+1;end;
  As2=-max(H2(i:K));
```

```
=edge;while H2(i)> -As2, i=i+1; end;
ws2=w(i);
As3=[];
plot(w,H2,'-w');
itle(['M=',num2str(M),' m=',num2str(m),' m1=',num2str(m1),...
                                        ' ws=',num2str(pi*(1+d)/(2*M))]);
ext(0.52,0.86,['E2: groen, ws2=',num2str(ws2),' As2=',num2str(As2)],'sc');
d
use
is;
```

.2.2. Ontwerp functies t.b.v. Transversale structuur

```
nction P=remezp(M,m)
 P=remezp(M,m)

 2/3*log10(1/(10*δ1*δ2))=m*dw volgens de formule van Bellanger
 we nemen m*dw=6

v=6/(m-1/(2*M));
v=6/m;
=2*M*m;
1=[];
2=[];
=[];
=[];
=[];

nargin==2
f m==2
 ds=0.69;
lse
 ds=0.62;
nd
printf('ds % 2.4e\n',ds);
vhile 1
 f=[0 (1-dw*(1-ds))/(2*M) (1+dw*ds)/(2*M) 1];
 P=remezs(2*M*m-1,f,[1 1 0 0]);
 for n=0:N-1
   h(n+1)=2*P(n+1)*cos((n-(N-1)/2)*pi/(2*M)+pi/4);
 end
 hh=freqz(h,1,[0 pi/M]);
 hs=sqrt(2)*abs(hh(2));
 fprintf('H0(0)= %2.4e          H0(π/%2.0f)= %2.4e\n',hh(1),M,abs(hh(2)));
 if abs(hs-1)<1e-5, % nauwkeurigheid
   break
 end
 if hs<1 % ds groter maken
   if isempty(ds2);
     ds1=ds;
     h1=hs;
     ds=ds+0.01;
   else
     ds1=ds;
     h1=hs;
     ds=(ds2-ds)*(1-h2+ds2*(h2-hs)/(ds2-ds))/(h2-hs);
   end
 else % ds kleiner maken
   if isempty(ds1);
     ds2=ds;
     h2=hs;
     ds=ds-0.01;
   else
     ds2=ds;
     h2=hs;
     ds=(ds-ds1)*(1-hs+ds*(hs-h1)/(ds-ds1))/(hs-h1);
   end
 end
 fprintf('Berekende ds % 2.4e\n',ds);
nd
```

```
end

function [Asp,wsp]=plotp(P,M))
% [Asp,wsp]=plotp(P,M)
%

K=512;

% Berekenen magnitude responsie van de prototype.

[H,w]=freqz(P,1,K);
H=20*log10(abs(H));
Lo=-inf;
Lo=max(Lo,max(H));
i=round(K/(2*M));while H(i+1) < H(i),i=i+1;end;
Asp=-max(H(i:K));
Hi=-inf;
Hi=max(Hi,Asp);
i=round(K/(2*M));while H(i)> -Asp, i=i+1; end;
wsp=w(i);
axis([0 pi -Hi-20 Lo]);
plot(w,H,'-w');
title(['M=',num2str(M),' N=',num2str(length(P))]);
text(0.52,0.89,['wsp=',num2str(wsp),' Asp=',num2str(Asp)],'sc');
pause
axis
```

.3 Quantisatoren en operators met quantisatie

```
nction y=flquant1(x,N,Ex,s)
y=flquant(x,N,Ex,s)

Quant maakt van een MATLAB floating-point getal x een
floating-point getal y met een mantisse lengte van N,
en een exponent van -Ex t/m Ex. N en Ex worden in een decimale
notatie uitgedrukt in tegenstelling tot de fixed-point
quantisatie functie fxquant.

Met s kan de soort discretisatie opgegeven worden:
  s=0 :  afronden naar het dichst bijzijnde integer (rounding)
  s=1 :  afronden naar nul (magnitude truncation)
  s=2 :  afronden naar beneden (value truncation)


0;
k]=size(x);
zeros(l,k);

Elke underflow geeft nul

s==0, % Rounding
or i=1:l
 for j=1:k
  if x(i,j)~=0
   E=log10(abs(x(i,j)));
   if E>=Ex % overflow
     y(i,j)=sign(x(i,j))*(1-10^(-N))*10^(Ex);
     o=o+1;
   elseif abs(E)<=Ex
     E=ceil(E);
     y(i,j)=round(x(i,j)*10^(N-E));
     y(i,j)=y(i,j)*10^(E-N);
   end
  end
 end
nd
eif s==1, % Magnitude Truncation
or i=1:l
 for j=1:k
  if x(i,j)~=0
   E=log10(abs(x(i,j)));
   if E>=Ex % overflow
     y(i,j)=sign(x(i,j))*(1-10^(-N))*10^(Ex);
     o=o+1;
   elseif abs(E)<=Ex
     E=ceil(E);
     y(i,j)=fix(x(i,j)*10^(N-E));
     y(i,j)=y(i,j)*10^(E-N);
   end
  end
 end
nd
eif s==2
or i=1:l
 for j=1:k
  if x(i,j)~=0
   E=log10(abs(x(i,j)));
   if E>=Ex % overflow
     y(i,j)=sign(x(i,j))*(1-10^(-N))*10^(Ex);
     o=o+1;
   elseif abs(E)<=Ex
     E=ceil(E);
     y(i,j)=floor(x(i,j)*10^(N-E));
     y(i,j)=y(i,j)*10^(E-N);
   end
  end
   end
  end
else
  error('Onbekende quantisatie vorm');  % s= 0, 1 of 2
end

if o, fprintf('%4.0fx Overflow\n',o);end;


function y=fxquant(x,N,s,in)
% y=fxquant(x,N,s,in)
%
% Fxquant maakt van een MATLAB floating-point getal x een
% fixed-point getal y met een mantisse lengte van N-1 bits en
% één sign-bit.
% Als in='max' dan wordt het maximum van absolute waarden van x
% gerepresenteerd met het grootste/kleinste fixed-point getal. Factor
% is dan een conversie factor. En de andere waarden worden daaraan
% gerelateerd. in='max' is optioneel.
%
% Met s kan de soort discretisatie opgegeven worden:
%   s=0 :  afronden naar het dichst bijzijnde integer (rounding)
%   s=1 :  afronden naar nul (magnitude truncation)
%   s=2 :  afronden naar beneden (value truncation)
%
o=0;
maxbin=2^(N-1);

if nargin==4
  if strcmp(in,'max')
    x=x*2^(N-1)/max(max(abs(x)));
  else
    error('Vierde input argument heeft onbekende waarde');
  end
end

if s==0
  y=round(x);
elseif s==1
  y=fix(x);
elseif s==2
  y=floor(x);
else
  error('Verkeerde status');
end

i=find(abs(y)>2^(N-1));
if ~isempty(i)
  [k,l]=size(y);
  y=reshape(y,1,k*l);
  for j=1:length(i)
    y(i(j))=sign(y(i(j)))*2^(N-1);
  end
  y=reshape(y,k,l);
  fprintf('%4.0fx Overflow\n',length(i));end;
end


function C=qmat_mul(A,B,status,N,Ex)
% C=qmat_mul(A,B,status,N,Ex)
%
% Het gequantiseerd vermenigvuldigen van twee matrixes A en B. C=A*B.
% Met status kan de soort quantisator opgegeven worden:
%   status(1)=1 : Een floating point quantisator met een mantisse lengte
%                 N, en een exponent van -Ex t/m Ex.
%   status(1)=2 : Fixed-point quantisator met een mantisse lengte van N-1
%                 bits en één sign-bit.
```

```
%   status(2)=0 : afronden naar het dichst bijzijnde integer (rounding)
%   status(2)=1 : afronden naar nul (magnitude truncation)
%   status(2)=2 : afronden naar beneden (value truncation)
%
% Bij matrix vermenigvuldiging worden elementaire optellingen en
% vermenigvuldigingen uitgevoerd. Wanneer twee getallen met een mantisse
% lengte N en exponent Ex met elkaar vermenigvuldigd worden ontstaat
% een getal met een mantisse en exponent van 2*N en 2*Ex. De tussenresulaten
% van matrix vermenigvuldiging worden gequantiseerd met een mantisse en
% exponent van 2*N en 2*Ex. De elementen van C hebben een mantisse en
% exponent die twee keer zo groot is als die van A en B.

f nargin==4
 Ex=[];
nd

rB,kB]=size(B);
rA,kA]=size(A);

f rB==kA
 C=zeros(rA,kB);
 AA=A';
for i=1:kB
  C(:,i)=qsum(diag(B(:,i))*AA,status,2*N,2*Ex)';
end
lse
error('matrix dimensies komen niet overeen');
nd



unction S=qsum(X,status,N,Ex)
% S=qsum(X,status,N,Ex)
%
% X is een matrix/vector waarvan de elementen gequantiseerd zijn volgens de
% overige inputs status, N en Ex. Voor vector X, S is de gequantiseerde som
% van de elementen. Voor matrix X, S is een rij-vector met de gequantiseerde
% som van elke kolom. Het eind resultaat S is een getal met een representatie
% volgens status, N en Ex. Tussenresultaten worden ook gerepresenteerd
% met N en Ex.
% Met status kan de soort quantisator opgegeven worden:
%   status(1)=1 : Een floating point quantisator met een mantisse lengte
%                 N, en een exponent van -Ex t/m Ex.
%   status(1)=2 : Fixed-point quantisator met een mantisse lengte van N-1
%                 bits en één sign-bit.
%   status(2)=0 : afronden naar het dichst bijzijnde integer (rounding)
%   status(2)=1 : afronden naar nul (magnitude truncation)
%   status(2)=2 : afronden naar beneden (value truncation)
%
=0;
,l]=size(X);
f status(1)==1 % floating point
if k==1 | l==1
  S=X(1);
  for i=2:length(X)
    S=S+X(i);
    if S~=0
      E=log10(abs(S));
      if E>=Ex % overflow
        S=sign(S)*(1-10^(-N))*10^(Ex);
        o=o+1;
      elseif abs(E)<=Ex
        E=ceil(E);
        if status(2)==0
          S=round(S*10^(N-E))*10^(E-N);
        elseif status(2)==1
          S=fix(S*10^(N-E))*10^(E-N);
        else
          S=floor(S*10^(N-E))*10^(E-N);
        end
      end
    end
  end
else
  S=zeros(1,l);
  for j=1:l
    S(j)=X(1,j);
    for i=2:k
      S(j)=S(j)+X(i,j);
      if S(j)~=0
        E=log10(abs(S(j)));
        if E>=Ex % underflow
          S(j)=sign(S(j))*(1-10^(-N))*10^(Ex);
          o=o+1;
        elseif abs(E)<=Ex
          E=ceil(E);
          if status(2)==0
            S(j)=round(S(j)*10^(N-E))*10^(E-N);
          elseif status(2)==1
            S(j)=fix(S(j)*10^(N-E))*10^(E-N);
          else
            S(j)=floor(S(j)*10^(N-E))*10^(E-N);
          end
        end
      end
    end
  end
elseif status(1)==2, % fixed point
 m=2^(N-1);
 i=find(abs(cumsum(X))>m);
 if ~isempty(i)
  o=length(i);
  if k==1 | l==1
    S=X(1);
    for i=2:length(X)
      S=X(i)+S;
      if abs(S) > m, % overflow
        S=sign(S)*m;
      end;
    end
  else
    S=zeros(1,l);
    for j=1:l
      S(j)=X(1,j);
      for i=2:k
        S(j)=X(i,j)+S(j);
        if abs(S(j))>m, % overflow
          S(j)=sign(S(j))*m;
        end;
      end
    end
  end
 else
  S=sum(X);
 end
else
  error('Fixed of floating-point quantisatie');
end

if o, fprintf('%4.0fx Overflow\n',o);end;
```

..4 Functies t.b.v. simuleren filterbank

.4.1 Functies t.b.v. het meten van de preformance en plotten frequentie spectra

```
ıction As=ploth(H,M,m,d)
As=ploth(H,M,m,d)

Geeft de magnitude respons in dB als grafiek van de
M analyse filters Hk. Rij k bevat filter Hk.
d is een maat voor de stopband grens ws=π(1+d)/(2*M), met M het aantal
kanalen van de filterbank. Deze stopband grens wordt gebruikt voor de
stopband minimale onderdrukking As.


·256;
ıg=zeros(K,M);

 i=1:M
HH,w]=freqz(H(i,:),1,K);
lag(:,i)=abs(HH);
l

ge=round(K/M);
dge;while Mag(i+1,1) < Mag(i,1),i=i+1;end;
=max(20*log10(Mag(i:K,1)));

-edge;while Mag(i-1,M) < Mag(i,M),i=i-1;end;
=max(As,max(20*log10(Mag(1:i,M))));

 j=1:M-2
=round(K*j/M);
 i>1,
 while Mag(i-1,j+1) < Mag(i,j+1),
  if i==2,
   i=1;
   break;
  else
   i=i-1;
  end;
 end;
nd;
 i~=1
 As1=max(20*log10(Mag(1:i,j+1)));
 As=max(As,As1);
nd
=round(K*(j+1)/M);
 i<K,
 while Mag(i+1,j+1) < Mag(i,j+1),
  if i==K-1,
   i=K;
   break;
  else
   i=i+1;
  end;
 end;
nd;
 i~=K
 As1=max(20*log10(Mag(i:K,j+1)));
 As=max(As,As1);
nd
l;
=-As;

s([0 pi -As-20 ceil(max(max(20*log10(Mag))))]);
t(w,20*log10(Mag))

:(['M=',num2str(M),'  m=',num2str(m),'  ws=',num2str(pi*(1+d)/(2*M)),..
                                                                    ' As=',num2str(As
pause;
axis;
function S=mse(x,y,N)
% S=mse(x,y,N)
%
% S is de mean square error van y t.o.v. x. y is de output van
% de filterbank en x de ingang. y is N tijdseenheden vertraagt
% t.o.v. x.
%

l=length(x);
X=zeros(1,length(y));
X(N:N+l-1)=x;

s1=0;s2=0;

for i=1:length(y)
  s1=s1+(y(i)-X(i))^2;
end

for i=1:l
  s2=s2+(x(i))^2;
end

S=s1/s2;


function [Ea,Epp]=distor(E_P,M,m,qstatus,fbstatus,N,Ex)
% [Ea,Epp]=distor(E_P,M,m,qstatus,fbstatus,N,Ex)
%
% Berekend de Peak-to-Peak Reconstructie fout Epp en de maximale
% aliasing fout Ea.
%
%            M                          jw
%  T(z)=1/M · Σ  H (z)·F (z) , (1-δ1)≤M | T(e  ) | ≤(1+δ1), Epp=δ1+δ2
%           k=1  k    k
%
%         -N     -1
%   F (z)=z  · H (z )    , N=2Mm   en
%    k          k
%
%              ┌ M-1 M      l      2 ┐½
%  Ea=max =1/M·| Σ  Σ    | H(zW )F (z) |   |
%        w     └ l=1 k=1  k     k     ┘
%

if fbstatus(2) % transversale structuur met een ongedecimeerde en gedecimeer
filterbank
  H=abank_t(E_P,M,m,1,[qstatus,0,fbstatus(3)],N,Ex);
else
  H=abank_l(E_P,M,m,1,[qstatus,0,fbstatus(3)],N,Ex);
end

if qstatus(1)==2, %fixed-point
  H=H/sum(H(1,:));
end

K=256;

F=fliplr(H); % synthese filters.

for k=1:M
```

```
HH(k,:)=conv(H(k,:),F(k,:));
nd

sum(HH)/M;
T,w]=freqz(t,1,K);
pp=max(1-abs(T*M))-min(1-abs(T*M));

=zeros(K,1);
/=exp(-j*2*pi/M);
=0:2*M*m-1;

or l=1:M-1
WH=H*diag(W.^(l*i));
for k=1:M
  HH(k,:)=conv(WH(k,:),F(k,:));
end
al=sum(HH);
E=E+abs(freqz(al,1,K)).^2;
nd
=sqrt(E)/M;
a=max(E);

xis([0 pi min(1-abs(T*M)) max(1-abs(T*M))]);
ubplot(211)
lot(w,1-abs(T*M),'w');
tle([' 1-|T(w)|, Peak-Peak reconstruction error Epp: ',num2str(Epp)]);

xis([0 pi min(E) Ea]);
ubplot(212)
lot(w,E,'w')
tle([' Aliasing error E(w) met maximum Ea: ',num2str(Ea)]);
ubplot(111)
ause
xis([1 2 3 4]);
```

A.4.2 Functies t.b.v. het simuleren van de filterbank met een lattice structuur

```
nction Y=abank_l(E,M,m,x,status,N,Ex)
 Y=abank_l(E,M,m,x,status,N,Ex)
%
% De cosinus-gemoduleerde M-kanaals FIR filter analyse bank.De analyse bank is
% opgebouwd uit lattice filters en een cosinus-modulatie matrix. De lattice
% filters bestaan uit m secties. De lattice-coefficienten bevinden zich in de
% vector E=[Θ(1,1),..,Θ(1,m),Θ(2,1),....,Θ(L,m)] met L=fix(M/2), Θ(k,p) is de
% lattice coefficient van lattice k sectie p. x is de input-vector en Y de
% output-matrix. status bepaald de vorm van de analyse filterbank:
%  status(1)=0  Een niet gequantiseerde filterbank
%  status(1)=1  Een gequantiseerde filterbank met floating-point representatie
%               met een mantisse lengte N, en een exponent van -Ex t/m Ex.
%  status(1)=2  Een gequantiseerde filterbank met fixed-point representatie
%               met een mantisse lengte van N-1 bits en één sign-bit. ( |x|≤1)
%  status(2)=0  Afronden naar het dichst bijzijnde integer (rounding)
%  status(2)=1  Afronden naar nul (magnitude truncation)
%  status(2)=2  Afronden naar beneden (value truncation)
%  status(3)=0  Een niet-gedecimeerde filterbank.
%  status(3)=1  Een maximaal gedecimeerde filterbank.
%  status(4)=0  De secties uit de lattice-filters met vier vermenigvuldigers.
%  status(4)=1  De secties uit de lattice-filters met twee vermenigvuldigers.
%
% Als x=1 dan bevat Y(i,:) de impuls responsie van analyse filter H (z)
%                                                    i

lc;disp('');disp('Opzetten van de lattice-structuur van de analyse filterbank')

% permutatie matrixen B1 B2 en B3
B1=zeros(2*M,2*M); B2=zeros(2*M,M); B3=zeros(M,2*M);

for i=1:M
  B1(2*i-1,2*i-1)=1;
end

for i=1:M
  B2(2*i,i)=1;
end

B3=B2';

R0=zeros(2*M,M);
R=zeros(2*M,2*M*(m-1));

if rem(M,2)==1,  % oneven aantal kanalen
  R0(M,(M+1)/2)=1/sqrt(2);     % lattice (M+1)/2 is slechts een delay
  R0(M+1,(M+1)/2)=1/sqrt(2);
  for i=1:m-1,
    R(M+1,(i-1)*2*M+M)=1;
    R(M,(i-1)*2*M+M+1)=1;
  end
end

L=fix(M/2);
if status(4), % lattice met 2 vermenigvuldigers
```

```
r i=1:L
a=1;
for j=1:m-1
  a=a*sin(E(m*(i-1)+j+1));
  R(2*i-1,(j-1)*2*M+2*i-1)=cos(E(m*(i-1)+j+1))/sin(E(m*(i-1)+j+1));
  R(2*i-1,(j-1)*2*M+2*i)=1;
  R(2*i,(j-1)*2*M+2*i-1)=1;
  R(2*i,(j-1)*2*M+2*i)=-cos(E(m*(i-1)+j+1))/sin(E(m*(i-1)+j+1));
  R(2*(M-i+1),(j-1)*2*M+2*(M-i+1))=cos(E(m*(i-1)+j+1))/sin(E(m*(i-1)+j+1));
  R(2*(M-i+1),(j-1)*2*M+2*(M-i+1)-1)=1;
  R(2*(M-i)+1,(j-1)*2*M+2*(M-i+1))=1;
  R(2*(M-i)+1,(j-1)*2*M+2*(M-i+1)-1)=-cos(E(m*(i-1)+j+1))/sin(E(m*(i-1)+j+1));
end
R0(2*i-1,i)=cos(E(m*(i-1)+1))*a;
R0(2*i,i)=sin(E(m*(i-1)+1))*a;
R0(2*(M-i+1),M-i+1)=cos(E(m*(i-1)+1))*a;
R0(2*(M-i)+1,M-i+1)=sin(E(m*(i-1)+1))*a;
ıd
ear a;
, % lattice met 4 vermenigvuldigers
r i=1:L
R0(2*i-1,i)=cos(E(m*(i-1)+1));
R0(2*i,i)=sin(E(m*(i-1)+1));
R0(2*(M-i+1),M-i+1)=cos(E(m*(i-1)+1));
R0(2*(M-i)+1,M-i+1)=sin(E(m*(i-1)+1));
for j=1:nı-1
  R(2*i-1,(j-1)*2*M+2*i-1)=cos(E(m*(i-1)+j+1));
  R(2*i-1,(j-1)*2*M+2*i)=sin(E(m*(i-1)+j+1));
  R(2*i,(j-1)*2*M+2*i-1)=sin(E(m*(i-1)+j+1));
  R(2*i,(j-1)*2*M+2*i)=-cos(E(m*(i-1)+j+1));
  R(2*(M-i+1),(j-1)*2*M+2*(M-i+1))=cos(E(m*(i-1)+j+1));
  R(2*(M-i+1),(j-1)*2*M+2*(M-i+1)-1)=sin(E(m*(i-1)+j+1));
  R(2*(M-i)+1,(j-1)*2*M+2*(M-i+1))=sin(E(m*(i-1)+j+1));
  R(2*(M-i)+1,(j-1)*2*M+2*(M-i+1)-1)=-cos(E(m*(i-1)+j+1));
end
ıd
l
ar L

coefficient quantisatie
tatus(1)==1, % floating-point
isp('Coefficient Quantisatie');
0=flquant(R0,N,Ex,status(2));
=flquant(R,N,Ex,status(2));
eif status(1)==2, % fixed-point
isp('Coefficient Quantisatie');
R0=fxquant(R0*2^(N-1),N,status(2));
R=fxquant(R,N,status(2),'max');      % R(i,j)≤1
R=fxquant(R*2^(N-1),N,status(2));
0=fxquant(R0,N,status(2),'max');
 status(4) % twee vermenigvuldigers lattice
for i=1:m-1
  K=N-1-ceil(log(max(max(abs(R(:,(i-1)*2*M+1:i*2*M)))))/log(2));
  R(:,(i-1)*2*M+1:i*2*M)=round(R(:,(i-1)*2*M+1:i*2*M)*2^K);
end
se  % vier vermenigvuldigers lattice
for i=1:m-1
  R(:,(i-1)*2*M+1:i*2*M)=fxquant(R(:,(i-1)*2*M+1:i*2*M),N,status(2),'max');
end
ıd
x=[];
l

nput quantisatie en pre-multiplying
tatus(1)==1        % floating-point
isp('Input Quantisatie');
 status(3) % maximaal gedecimeerde filterbank
x=flquant(x*flquant(1/sqrt(2),N,Ex,status(2)),N,Ex,status(2));
se
x=flquant(x*flquant(1/sqrt(2*M),N,Ex,status(2)),N,Ex,status(2));
ıd
elseif status(1)==2    % fixed-point
  disp('Input Quantisatie');
  if max(abs(x))>1
    disp('Schalen van input x');
    x=fxquant(x,N,status(2),'max');
  else  % |x|≤1
    disp('Schalen van input x');
    x=fxquant(x,N,status(2),'max');
%    x=fxquant(x*2^(N-1),N,status(2));
  end
else
  if status(3) % maximaal gedecimeerde filterbank
    x=x/sqrt(2);
  else
    x=x/sqrt(2*M);
  end
end

% verdeling input over kanalen
if status(3), % maximaal gedecimeerde filterbank
  k=1;
  D=zeros(M,1);
  S=zeros(M,(m-1)*2);
  K=length(x);
  X=zeros(1,ceil((K+M-1)/M)*M);
  X(M:M+K-1)=x;
  X=flipud(reshape(X,M,ceil((K+M-1)/M)));
  X(1,ceil((K+M-1)/M)+2*m-1)=0;
  K=ceil((K+M-1)/M)+2*m-1;
  Y=zeros(2*M,K);
else % niet gedecimeerde filterbank
  k=M;
  D=zeros(M,M);
  S=zeros(M,(m-1)*2*M);
  K=length(x);
  X=zeros(M,K+M*2*m-1);
  K=K+M*2*m-1;
  Y=zeros(2*M,K);
  for i=1:M,
    X(i,i:i+length(x)-1)=x;
  end
end

disp(' ');disp('Verwerken van de samples door de lattice-structuur');

disp('Sectie 1');
X=R0*X;
if status(1)==1, % floating-point
  X=flquant(X,N,Ex,status(2));
elseif status(1)==2 % fixed point, schaling
  X=fxquant(X*2^(-N+1),N,status(2));
end
clear R0;

disp('Sectie 2..m');disp('');
for i=1:K
  fprintf('  %3.0f',i);
  if rem(i,15)==0, fprintf(' \n'); end;
  U1=X(:,i);
  for j=1:m-1
    if status(1) % inputs alle met N en Ex
      U2=qmat_mul(R(:,(j-1)*2*M+1:j*2*M),B1*U1+B2*S(:,j*2*k),status(1:2),N
      if status(1)==1 % floating point
        U2=flquant(U2,N,Ex,status(2));
      elseif status(1)==2 % fixed, schaling
        if j==1
          U2=fxquant(U2*2^(-N),N,status(2));
        else
          U2=fxquant(U2*2^(-N+1),N,status(2));
        end
      end
```

```
  else
    U2=R(:,(j-1)*2*M+1:j*2*M)*(B1*U1+B2*S(:,j*2*k));
  end
  S(:,(j-1)*2*k+2:j*2*k)=S(:,(j-1)*2*k+1:j*2*k-1);
  S(:,(j-1)*2*k+1)=-B3*U1;
  UI=U2;
 end
 Y(:,i)=B1*U2+B2*D(:,k);
 if ~status(3), D(:,2:M)=D(:,1:M-1); end;
 D(:,1)=B3*U2;
nd
lear U1 U2 B1 B2 D S X K k R

lisp(' ');disp(' ');disp('Opzetten van de cosinus modulatie matrix T');

f rem(m,2)==1  % m oneven
 IJ(:,1:M)=eye(M)+fliplr(eye(M));
 IJ(:,M+1:2*M)=eye(M)-fliplr(eye(M));
 if rem((m-1)/2,2)==1
   Dc=-1;
 else
   Dc=1;
 end
lse
 IJ(:,1:M)=eye(M)-fliplr(eye(M));
 IJ(:,M+1:2*M)=-eye(M)-fliplr(eye(M));
 if rem((m/2),2)==1
   Dc=-1;
 else
   Dc=1;
 end
nd

C=zeros(M,M);
or i=0:M-1,
 for j=0:M-1,
   C(i+1,j+1)=sqrt(2/M)*cos((i+1/2)*(j+1/2)*pi/M);
 end
nd

% permutatie matrix B4
34=zeros(2*M,2*M);
34(1:M,1:2*M)=flipud(fliplr(B3));
34(M+1:2*M,1:2*M)=B3;
lear B3

lisp(' ');

f status(1)==1, % floating point,
 disp('Quantiseren C matrix');
 C=flquant(C,N,Ex,status(2));
 disp('Verwerken van de samples door de cosinus modulatie matrix T');
 disp('Verwerken samples door de IJ matrix');
 Y=flquant(qmat_mul(IJ,B4*Y,status(1:2),N,Ex),N,Ex,status(2));
 disp('Verwerken samples door de C matrix');
 Y=Dc*flquant(qmat_mul(C,Y,status(1:2),N,Ex),N,Ex,status(2));;
lseif status(1)==2, %fixed-point, quantisatie en schaling
 disp('Quantiseren C matrix');
% C=round(2^(N)*C/max(sum(abs(C)))); % l-1 schaling
 C=round(C*2^(N-1));                % l-2 schaling
 disp('Quantiseren IJ matrix');
 IJ=IJ*2^(N-2); % max(Σ|IJ|)=2^N,
 disp('Verwerken van de samples door de cosinus modulatie matrix T');
 disp('Verwerken samples door de IJ matrix');
 Y=qmat_mul(IJ,B4*Y,status(1:2),N);
 disp('Schalen uitgang IJ matrix');
 Y=fxquant(Y*2^(-N),N,status(2));
 disp('Verwerken samples door de C matrix');
 Y=qmat_mul(C,Y,status(1:2),N);
 disp('Schalen uitgang C matrix');
 Y=Dc*fxquant(Y*2^(-N+1),N,status(2));
else
  Y=Dc*C*IJ*B4*Y;
end;


function y=sbank_l(E,M,m,X,status,N,Ex)
% y=sbank_l(E,M,m,X,status,N,Ex)
%
% De cosinus-gemoduleerde M-kanaals FIR filter synthese bank.De synthese
% is opgebouwd uit lattice filters en een cosinus-modulatie matrix. De lattice
% filters bestaan uit m secties. De lattice-coefficienten bevinden zich in de
% vector E=[Θ(1,1),..,Θ(1,m),Θ(2,1),....,Θ(L,m)] met L=fix(M/2), Θ(k,p) is d
% lattice coefficient van lattice k sectie p. X is de input-matrix, de output
% van de functie abank_l. De input argumenten van abank_l en sbank_l mo
% gelijk zijn (afgezien van x en X) om zinvolle resultaten te verkrijgen. y is
% de output-vector. De getallen-representatie van X en y is afhankelijk van
% status, die de vorm van de synthese filterbank bepaald:
%  status(1)=0  Een niet gequantiseerde filterbank
%  status(1)=1  Een gequantiseerde filterbank met floating-point representati
%               met een mantisse lengte N, en een exponent van -Ex t/m Ex.
%  status(1)=2  Een gequantiseerde filterbank met fixed-point representatie
%               met een mantisse lengte van N-1 bits en één sign-bit.
%  status(2)=0  Afronden naar het dichst bijzijnde integer (rounding)
%  status(2)=1  Afronden naar nul (magnitude truncation)
%  status(2)=2  Afronden naar beneden (value truncation)
%  status(3)=0  Een niet-gedecimeerde filterbank.
%  status(3)=1  Een maximaal gedecimeerde filterbank.
%  status(4)=0  De secties uit de lattice-filters met vier vermenigvuldigers.
%  status(4)=1  De secties uit de lattice-filters met twee vermenigvuldigers.

disp(' ');
disp('Opzetten van de cosinus modulatie matrix T');

if rem(m,2)==1  % m oneven
  IJ(1:M,:)=eye(M)+fliplr(eye(M));
  IJ(M+1:2*M,:)=eye(M)-fliplr(eye(M));
  if rem((m-1)/2,2)==1
    Dc=-1;
  else
    Dc=1;
  end
else % m even m=2*m1
  IJ(1:M,:)=eye(M)-fliplr(eye(M));
  IJ(M+1:2*M,:)=-eye(M)-fliplr(eye(M));
  if rem((m/2),2)==1  % m1 oneven
    Dc=-1;
  else % m1 even
    Dc=1;
  end
end

C=zeros(M,M);
for i=0:M-1,
  for j=0:M-1,
    C(i+1,j+1)=sqrt(2/M)*cos((i+1/2)*(j+1/2)*pi/M);
  end
end

disp(' ');

if status(1)==1, % floating point
  disp('Quantiseren C matrix');
  C=flquant(C,N,Ex,status(2));
  disp('Verwerken van de input-samples door de cosinus modulatie matrix T
  disp('Input verwerken door C matrix');
  X=flquant(qmat_mul(C,Dc*X,status(1:2),N,Ex),N,Ex,status(2));
  disp('en door IJ matrix');
  X=flquant(qmat_mul(IJ,X,status(1:2),N,Ex),N,Ex,status(2));
elseif status(1)==2, %fixed-point, quantisatie en schaling
  disp('Quantiseren C matrix');
% C=round(2^(N)*C/max(sum(abs(C))));  % l-1 schaling
```

```
=round(C*2^(N-1));                % 1-2 schaling
sp('Quantiseren IJ matrix');
=IJ*2^(N-1); % max(Σ|IJ|)=2^N,
sp('Verwerken van de input-samples door de cosinus modulatie matrix T');
sp('Input-samples verwerken door C matrix');
=qmat_mul(C,Dc*X,status(1:2),N);
sp('Schalen uitgang C matrix');
=fxquant(X*2^(-N+1),N,status(2));
sp('Samples verwerken door IJ matrix');
=qmat_mul(IJ,X,status(1:2),N);
sp('Schalen uitgang IJ matrix');
=fxquant(X*2^(-N+1),N,status(2));

sp('Verwerken van de input-samples door de cosinus modulatie matrix T');
=C*Dc*X;
=IJ*X;
;
ır IJ C Dc

)(' ');
)('Opzetten van de lattice-structuur van de synthese filterbank')

ix(M/2); % aantal niet triviale lattices

)ermutatie matrixen B1 B2 B3 B4 en B5
=zeros(2*M,2*M);
=zeros(2*M,M);
=zeros(M,2*M);
=zeros(2*M,2*M);
=zeros(M,2*M);

i=1:M
1(2*i-1,2*M-i+1)=1;
|

i=1:M
2(2*i,i)=1;
|

i=1:M
3(i,M-i+1)=1;
|

i=1:M
4(2*i-1,2*i-1)=1;
|

=-B2';

=zeros(2,2*M); % sectie 0
zeros(2*M,2*M*(m-1)); % secties m..1

em(M,2)==1,                % aantal kanalen M oneven
0((M+1)/2,M)=1/sqrt(2);      % lattice (M+1)/2 is slechts een delay
0((M+1)/2,M+1)=1/sqrt(2);
)r i=1:m-1,
R(M+I,(i-1)*2*M+M)=1;
R(M,(i-1)*2*M+M+1)=1;
ıd
|

tatus(4)  % lattice met 2 vermenigvuldigers
)r i=1:L
a=1;
for j=1:m-1
  a=a*sin(E(m*(i-1)+j+1));
  R(2*i-1,(m-1-j)*2*M+2*i-1)=cos(E(m*(i-1)+j+1))/sin(E(m*(i-1)+j+1));
  R(2*i-1,(m-1-j)*2*M+2*i)=1;
  R(2*i,(m-1-j)*2*M+2*i-1)=1;
  R(2*i,(m-1-j)*2*M+2*i)=-cos(E(m*(i-1)+j+1))/sin(E(m*(i-1)+j+1));
  R(2*(M-i+1),(m-1-j)*2*M+2*(M-i+1))=cos(E(m*(i-1)+j+1))/sin(E(m*(i-1)+j+1));
  R(2*(M-i+1),(m-1-j)*2*M+2*(M-i+1)-1)=1;
  R(2*(M-i)+1,(m-1-j)*2*M+2*(M-i+1))=1;
  R(2*(M-i)+1,(m-1-j)*2*M+2*(M-i+1)-1)=..

-cos(E(m*(i-1)+j+1))/sin(E(m*(i-1)+j+1));
  end
  R0(i,2*i-1)=cos(E(m*(i-1)+1))*a;
  R0(i,2*i)=sin(E(m*(i-1)+1))*a;
  R0(M-i+1,2*(M-i+1))=cos(E(m*(i-1)+1))*a;
  R0(M-i+1,2*(M-i)+1)=sin(E(m*(i-1)+1))*a;
 end
 clear a;
else       % lattice met 4 vermenigvuldigers
 for i=1:L
  R0(i,2*i-1)=cos(E(m*(i-1)+1));
  R0(i,2*i)=sin(E(m*(i-1)+1));
  R0(M-i+1,2*(M-i+1))=cos(E(m*(i-1)+1));
  R0(M-i+1,2*(M-i)+1)=sin(E(m*(i-1)+1));
  for j=1:m-1
   R(2*i-1,(m-1-j)*2*M+2*i-1)=cos(E(m*(i-1)+j+1));
   R(2*i-1,(m-1-j)*2*M+2*i)=sin(E(m*(i-1)+j+1));
   R(2*i,(m-1-j)*2*M+2*i-1)=sin(E(m*(i-1)+j+1));
   R(2*i,(m-1-j)*2*M+2*i)=-cos(E(m*(i-1)+j+1));
   R(2*(M-i+1),(m-1-j)*2*M+2*(M-i+1))=cos(E(m*(i-1)+j+1));
   R(2*(M-i+1),(m-1-j)*2*M+2*(M-i+1)-1)=sin(E(m*(i-1)+j+1));
   R(2*(M-i)+1,(m-1-j)*2*M+2*(M-i+1))=sin(E(m*(i-1)+j+1));
   R(2*(M-i)+1,(m-1-j)*2*M+2*(M-i+1)-1)=-cos(E(m*(i-1)+j+1));
  end
 end
end
clear L

% coefficient quantisatie
if status(1)==1, % floating-point
 disp('Coefficient Quantisatie');
 R0=flquant(R0,N,Ex,status(2));
 R=flquant(R,N,Ex,status(2));
elseif status(1)==2, % fixed-point
 disp('Coefficient Quantisatie');
% R0=fxquant(R0*2^(N-1),N,status(2));
% R=fxquant(R,N,status(2),'max');      % R(i,j)≤1
% R=fxquant(R*2^(N-1),N,status(2));
 R0=fxquant(R0,N,status(2),'max');
 if status(4) % twee vermenigvuldigers lattice
  for i=1:m-1
   K=N-1-ceil(log(max(max(abs(R(:,(i-1)*2*M+1:i*2*M)))))/log(2));
   R(:,(i-1)*2*M+1:i*2*M)=round(R(:,(i-1)*2*M+1:i*2*M)*2^K);
  end
 else  % vier vermenigvuldigers lattice
  for i=1:m-1
   R(:,(i-1)*2*M+1:i*2*M)=fxquant(R(:,(i-1)*2*M+1:i*2*M),N,status(2),'ma
  end
 end
 Ex=[];
end

if status(3), % maximaaal gedecimeerde filterbank
 k=1;
 S=zeros(M,(m-1)*2);
 D=zeros(M,1);
 [l,K]=size(X);
 K=K+2*m-1;
 X(1,K)=0;
 y=zeros(M,K);
else % niet gedecimeerde filterbank
 k=M;
 S=zeros(M,(m-1)*2*M);
 D=zeros(M,M);
 [l,K]=size(X);
 K=K+M*(2*m-1);
 X(1,K)=0;
```

```
y=zeros(M,K);
nd

isp(' ');
isp('Verwerken van de samples door de lattice-structuur');

isp('Sectie m..1');disp('');
or i=1:K
 fprintf('  %3.0f',i);
 if rem(i,15)==0, fprintf(' \n'); end;
 U1=B1*X(:,i)+B2*D(:,k);
 if ~status(3), % niet gedecimeerde filterbank
   D(:,2:M)=D(:,1:M-1);
 end;
 D(:,1)=B3*X(:,i);
 for j=1:m-1
   if status(1)
     U2=qmat_mul(R(:,(j-1)*2*M+1:j*2*M),U1,status(1:2),N,Ex);
     if status(1)==1 % floating point
       U2=flquant(U2,N,Ex,status(2));
     elseif status(1)==2 % fixed point, schaling
       U2=fxquant(U2*2^(-N+1),N,status(2));
     end
   else
     U2=R(:,(j-1)*2*M+1:j*2*M)*U1;
   end
   U3=B4*U2+B2*S(:,j*2*k);
   S(:,(j-1)*2*k+2:j*2*k)=S(:,(j-1)*2*k+1:j*2*k-1);
   S(:,(j-1)*2*k+1)=B5*U2;
   U1=U3;
 end
 y(:,i)=R0*U3;
nd
isp('');disp('');

f status(1)==1
 disp('Quantisatie output lattices');
 y=flquant(y,N,Ex,status(2));
lseif status(1)==2
 disp('Quantisatie output lattices');
 y=fxquant(y*2^(-N+1),N,status(2)); % max(sinΘ+cosΘ)=√2,
nd

f status(3)  % maximaal gedecimeerde filterbank
 if status(1)==1  % floating-point
   disp('Output Quantisatie');
   y=(-1)^(m-1)*flquant(reshape(y,1,M*K)*...
     flquant(1/sqrt(2),N,Ex,status(2)),N,Ex,status(2));
 elseif status(1)==2 % fixed-point
   y=(-1)^(m-1)*reshape(y,1,M*K);
 else
   y=(-1)^(m-1)*reshape(y,1,M*K)/sqrt(2);
 end
lse    % niet gedecimeerde filterbank
 for i=2:M
   y(i,i:K+i-1)=y(i,1:K);
 end
 if status(1)==1, % floating-point
   disp('Output Quantisatie');
   y=qsum(y,status(1:2),2*N,2*Ex);
   y=flquant(y,N,Ex,status(2));
   y=(-1)^(m-1)*flquant(y*...
     flquant(1/sqrt(2*M),N,Ex,status(2)),N,Ex,status(2));
 elseif status(1)==2, % fixed-point
   disp('Output Quantisatie');
   y=qsum(y*2^N,status(1:2),2*N,2*Ex);
   y=fxquant(y*2^(-N),N,status(2));
   y=(-1)^(m-1)*y;
 else
   y=(-1)^(m-1)*sum(y)/sqrt(2*M);
 end
```

```
end;
```

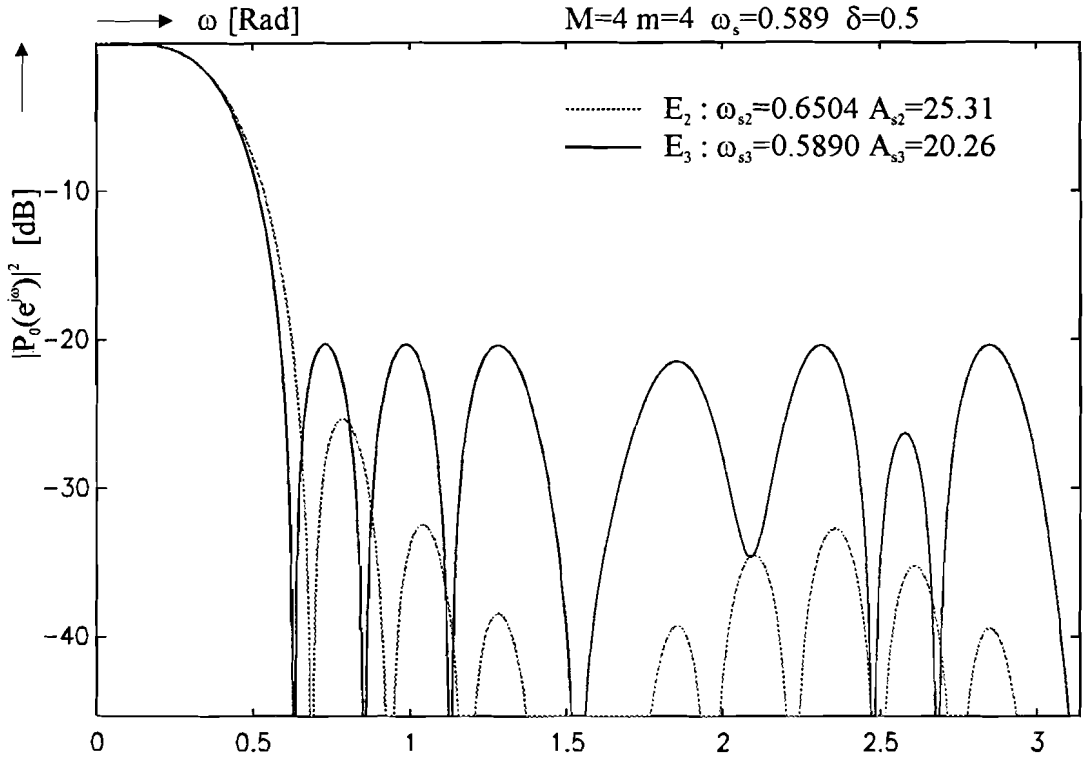
..4.3 Functies t.b.v. het simuleren van de filterbank met een transversale structuur

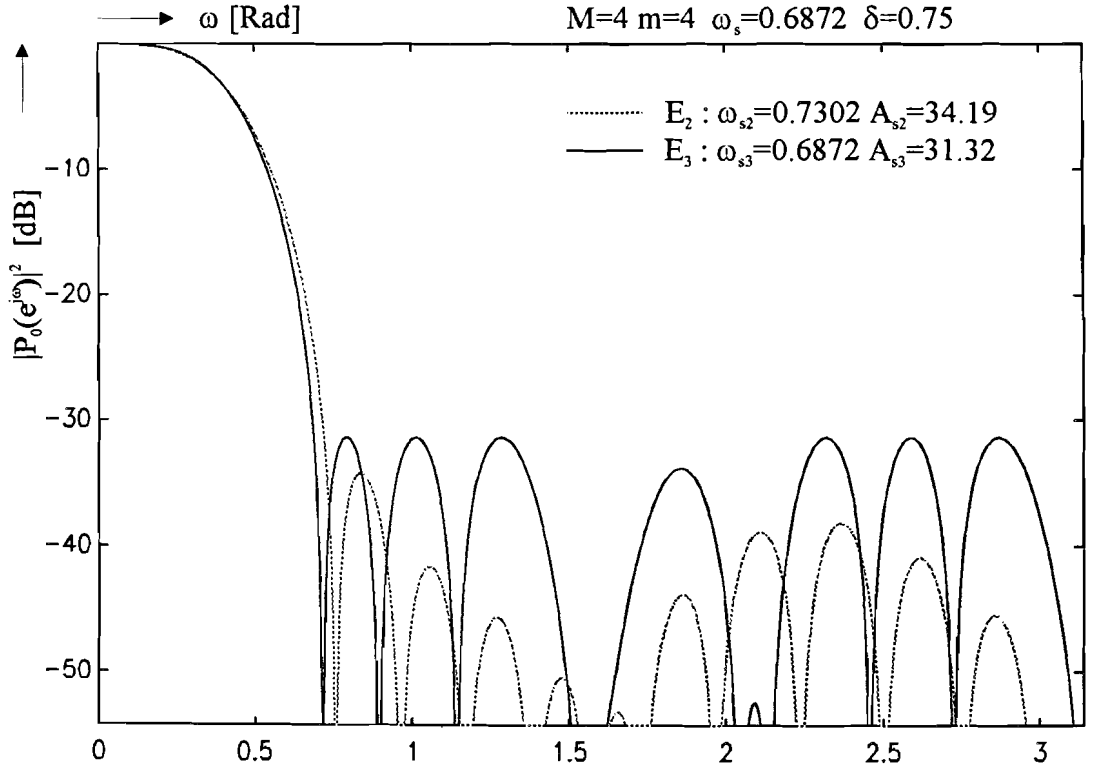
```
ıction Y=abank_t(E_P,M,m,x,status,N,Ex)
 Y=abank_t(E_P,M,m,x,status,N,Ex)

 De cosinus-gemoduleerde M-kanaals FIR filter analyse bank.De analyse bank is
 opgebouwd uit 2M transversale filters en een cosinus-modulatie matrix. x is
 de input-vector en Y de output-matrix. status bepaald de vorm van
 de analyse filterbank:
  status(1)=0  Een niet gequantiseerde filterbank
  status(1)=1  Een gequantiseerde filterbank met floating-point representatie
               met een mantisse lengte N, en een exponent van -Ex t/m Ex.
  status(1)=2  Een gequantiseerde filterbank met fixed-point representatie
               met een mantisse lengte van N-1 bits en één sign-bit. |x|≤1
  status(2)=0  Afronden naar het dichst bijzijnde integer (rounding)
  status(2)=1  Afronden naar nul (magnitude truncation)
  status(2)=2  Afronden naar beneden (value truncation)
  status(3)=0  Een niet-gedecimeerde filterbank.
  status(4)=0  impuls responsie transversale filters bepaald uit lattice
               coefficienten E_P=[Θ(1,1),..,Θ(1,m),Θ(2,1),....,Θ(L,m)]
  status(4)=1  impuls responsie transversale filters bepaald uit impuls
               responsie prototype E_P=P0.
 Als x=1 dan bevat Y(i,:) de impuls responsie van het analyse filter H (z)
                                                                    i

:;disp(' ');disp('Opzetten van de transversale-structuur van de analyse filterbank')

status(4), % E_P=P0
J=reshape(E_P,2*M,m)*M*sqrt(2);
ıe  % E_P=[Θ(1,1),..,Θ(1,m),Θ(2,1),....,Θ(L,m)]
J=reshape(proto(E_P,M,m),2*M,m)*M*sqrt(2);
d
2:2:m;
:,i)=-G(:,i);

status(1)==1, % floating-point
lisp('Coefficient Quantisatie');
J=flquant(G,N,Ex,status(2));
ıeif status(1)==2, % fixed-point
lisp('Coefficient Quantisatie');
 G=round(G*sqrt(1/(max(sum(G'.^2))))*2^(N-1));   % l-2 schaling
J=fxquant(G,N,0,'max');
Ex=[];
d

status(3) % gedecimeerde filterbank
K=length(x);
f status(1)==1        % floating-point
 disp('Input Quantisatie');
 x=flquant(x*flquant(1/sqrt(2),N,Ex,status(2)),N,Ex,status(2));
lseif status(1)==2    % fixed-point
 disp('Input Quantisatie');
 if max(abs(x)) > 1
   disp('Schalen van input x');
   x=fxquant(x,N,status(2),'max');
 else
   x=fxquant(x*2^(N-1),N,status(2));
 end
lse
 x=x/sqrt(2);
nd
lisp(' ');disp('Verwerken van de samples door de transversale-structuur');
K=zeros(1,ceil((K+M-1)/M)*M);
K(M:M+K-1)=x;
K=flipud(reshape(X,M,ceil((K+M-1)/M)));
K=ceil((K+M-1)/M);
Y=zeros(2*M,(m-1)*2+K+1);
 if status(1)
  for i=1:M
   for j=1:m
    X1(j,(j-1)*2+1:(j-1)*2+K)=X(i,:);
   end
   Y(i,1:(m-1)*2+K)=qmat_mul(G(i,:),X1,status(1:2),N,Ex);
   Y(i+M,2:(m-1)*2+K+1)=qmat_mul(G(i+M,:),X1,status(1:2),N,Ex);
  end
 else
  for i=1:M
   for j=1:m
    X1(j,(j-1)*2+1:(j-1)*2+K)=X(i,:);
   end
   Y(i,1:(m-1)*2+K)=G(i,:)*X1;
   Y(i+M,2:(m-1)*2+K+1)=G(i+M,:)*X1;
  end
 end
else % niet gedecimeerde filterbank
 K=length(x);
 if status(1)==1        % floating-point
  disp('Input Quantisatie');
  x=flquant(x*flquant(1/sqrt(2*M),N,Ex,status(2)),N,Ex,status(2));
 elseif status(1)==2    % fixed-point
  disp('Input Quantisatie');
  if max(abs(x)) > 1
   disp('Schalen van input x');
   x=fxquant(x,N,status(2),'max');
  else
   disp('Schalen van input x');
   x=fxquant(x,N,status(2),'max');
%    x=fxquant(x*2^(N-1),N,status(2));
  end
 else
  x=x/sqrt(2*M);
 end
 disp(' ');disp('Verwerken van de samples door de transversale-structuur');
 X=zeros(m,K+(m-1)*2*M);
 for i=1:m,
  X(i,(i-1)*2*M+1:(i-1)*2*M+K)=x;
 end
 K=K+(m-1)*2*M;
 if status(1)
  for i=1:M
   Y(i,i:K+i-1)=qmat_mul(G(i,:),X,status(1:2),N,Ex);
   Y(i+M,i+M:K+i-1+M)=qmat_mul(G(i+M,:),X,status(1:2),N,Ex);
  end
 else
  for i=1:M
   Y(i,i:K+i-1)=G(i,:)*X;
   Y(i+M,i+M:K+i-1+M)=G(i+M,:)*X;
  end
 end
end
if status(1)==1
 disp('Quantisatie output polyfase componenten');
 Y=flquant(Y,N,Ex,status(2));
elseif status(1)==2
 disp('Schaling en Quantisatie output polyfase componenten');
 Y=fxquant(Y*2^(-N),N,status(2));
end

disp(' ');disp('Opzetten van de cosinus modulatie matrix T');

if rem(m,2)==1  % m oneven
 IJ(:,1:M)=eye(M)+fliplr(eye(M));
 IJ(:,M+1:2*M)=eye(M)-fliplr(eye(M));
```

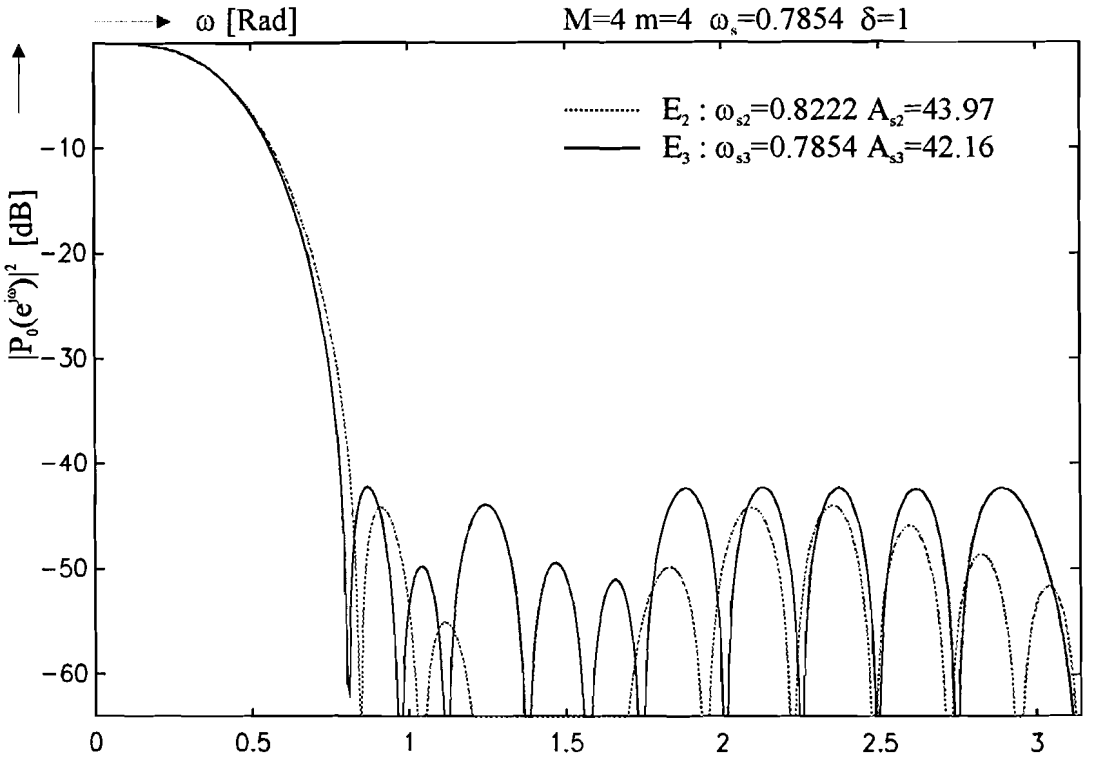
```
if rem((m-1)/2,2)==1
  Dc=-1;
else
  Dc=1;
end
lse
IJ(:,1:M)=eye(M)-fliplr(eye(M));
IJ(:,M+1:2*M)=-eye(M)-fliplr(eye(M));
if rem((m/2),2)==1
  Dc=-1;
else
  Dc=1;
end
nd

=zeros(M,M);
or i=0:M-1,
 for j=0:M-1,
   C(i+1,j+1)=sqrt(2/M)*cos((i+1/2)*(j+1/2)*pi/M);
 end
nd

isp(' ');

 status(1)==1, % floating point
 disp('Quantiseren C matrix');
C=flquant(C,N,Ex,status(2));
disp('Verwerken van de samples door de cosinus modulatie matrix T');
disp('Verwerken samples door de IJ matrix');
Y=flquant(qmat_mul(IJ,Y,status(1:2),N,Ex),N,Ex,status(2));
disp('Verwerken samples door de C matrix');
Y=qmat_mul(C,Y,status(1:2),N,Ex);
Y=Dc*flquant(Y,N,Ex,status(2));;
lseif status(1)==2, % fixed-point, quantisatie en schaling
disp('Quantiseren C matrix');
 C=fxquant(C,N,0,'max');
 C=round(2^(N)*C/max(sum(abs(C))));  % l-1 schaling
C=round(C*2^(N-1));                 % l-2 schaling
disp('Quantiseren IJ matrix');
IJ=IJ*2^(N-2); % max(Σ | IJ |)=2^N,
disp('Verwerken van de samples door de cosinus modulatie matrix T');
disp('Verwerken samples door de IJ matrix');
Y=qmat_mul(IJ,Y,status(1:2),N);
disp('Schalen uitgang IJ matrix');
Y=fxquant(Y*2^(-N),N,status(2));
disp('Verwerken samples door de C matrix');
Y=qmat_mul(C,Y,status(1:2),N);
disp('Schalen uitgang C matrix');
Y=Dc*fxquant(Y*2^(-N+1),N,status(2));
lse
disp('Verwerken van de samples door de cosinus modulatie matrix T');
Y=Dc*C*IJ*Y;
nd;


nction y=sbank_t(E_P,M,m,X,status,N,Ex)
 y=sbank_t(E_P,M,m,X,status,N,Ex)
 De cosinus-gemoduleerde M-kanaals FIR filter synthese bank. De synthese bank
 is opgebouwd uit 2M transversale filters en een cosinus-modulatie matrix. x
 is de input-vector ( | x | ≤1) en Y de output-matrix. status bepaald de vorm van
 de analyse filterbank:
  status(1)=0  Een niet gequantiseerde filterbank
  status(1)=1  Een gequantiseerde filterbank met floating-point representatie
               met een mantisse lengte N, en een exponent van -Ex t/m Ex.
  status(1)=2  Een gequantiseerde filterbank met fixed-point representatie
               met een mantisse lengte van N-1 bits en één sign-bit.
  status(2)=0  Afronden naar het dichst bijzijnde integer (rounding)
  status(2)=1  Afronden naar nul (magnitude truncation)
  status(2)=2  Afronden naar beneden (value truncation)
  status(3)=0  Een niet-gedecimeerde filterbank.
  status(3)=1  Een maximaal gedecimeerde filterbank.
  status(4)=0  impuls responsie transversale filters bepaald uit lattice
               coefficienten E_P=[Θ(1,1),..,Θ(1,m),Θ(2,1),....,Θ(L,m)]
  status(4)=1  impuls responsie transversale filters bepaald uit impuls
               responsie prototype E_P=P0.

disp(' ');disp('Opzetten van de cosinus modulatie matrix T');

if rem(m,2)==1  % m oneven
  IJ(1:M,:)=eye(M)+fliplr(eye(M));
  IJ(M+1:2*M,:)=eye(M)-fliplr(eye(M));
  if rem((m-1)/2,2)==1
    Dc=-eye(M);
  else
    Dc=eye(M);
  end
else
  IJ(1:M,:)=eye(M)-fliplr(eye(M));
  IJ(M+1:2*M,:)=-eye(M)-fliplr(eye(M));
  if rem((m/2),2)==1
    Dc=-eye(M);
  else
    Dc=eye(M);
  end
end

C=zeros(M,M);
for i=0:M-1,
  for j=0:M-1,
    C(i+1,j+1)=sqrt(2/M)*cos((i+1/2)*(j+1/2)*pi/M);
  end
end

disp(' ');

if status(1)==1, % floating point,
  disp('Quantiseren C matrix');
  C=flquant(C,N,Ex,status(2));
  disp('Verwerken van de input-samples door de cosinus modulatie matrix T')
  disp('Input verwerken door C matrix');
  X=flquant(qmat_mul(C,Dc*X,status(1:2),N,Ex),N,Ex,status(2));
  disp('en door IJ matrix');
  X=flquant(qmat_mul(IJ,X,status(1:2),N,Ex),N,Ex,status(2));
elseif status(1)==2, % fixed-point, quantisatie en schaling
  disp('Quantiseren C matrix');
% C=round(2^(N)*C/max(sum(abs(C))));  % l-1 schaling
  C=round(C*2^(N-1));                 % l-2 schaling
  disp('Quantiseren IJ matrix');
  IJ=IJ*2^(N-1); % max(Σ | IJ |)=2^N,
  disp('Verwerken van de input-samples door de cosinus modulatie matrix T')
  disp('Input-samples verwerken door C matrix');
  X=qmat_mul(C,Dc*X,status(1:2),N);
  disp('Schalen uitgang C matrix');
  X=fxquant(X*2^(-N+1),N,status(2));
  disp('Samples verwerken door IJ matrix');
  X=qmat_mul(IJ,X,status(1:2),N);
  disp('Schalen uitgang IJ matrix');
  X=fxquant(X*2^(-N+1),N,status(2));
else
  disp('Verwerken van de input-samples door de cosinus modulatie matrix T')
  X=C*Dc*X;
  X=IJ*X;
end;
clear IJ C Dc

disp(' ');disp('Opzetten van de transversale-structuur van de synthese filterban

if status(4), % E_P=P0
  G=reshape(E_P,2*M,m)*M*sqrt(2);
else  % E_P=[Θ(1,1),..,Θ(1,m),Θ(2,1),....,Θ(L,m)]
  G=reshape(proto(E_P,M,m),2*M,m)*M*sqrt(2);
```

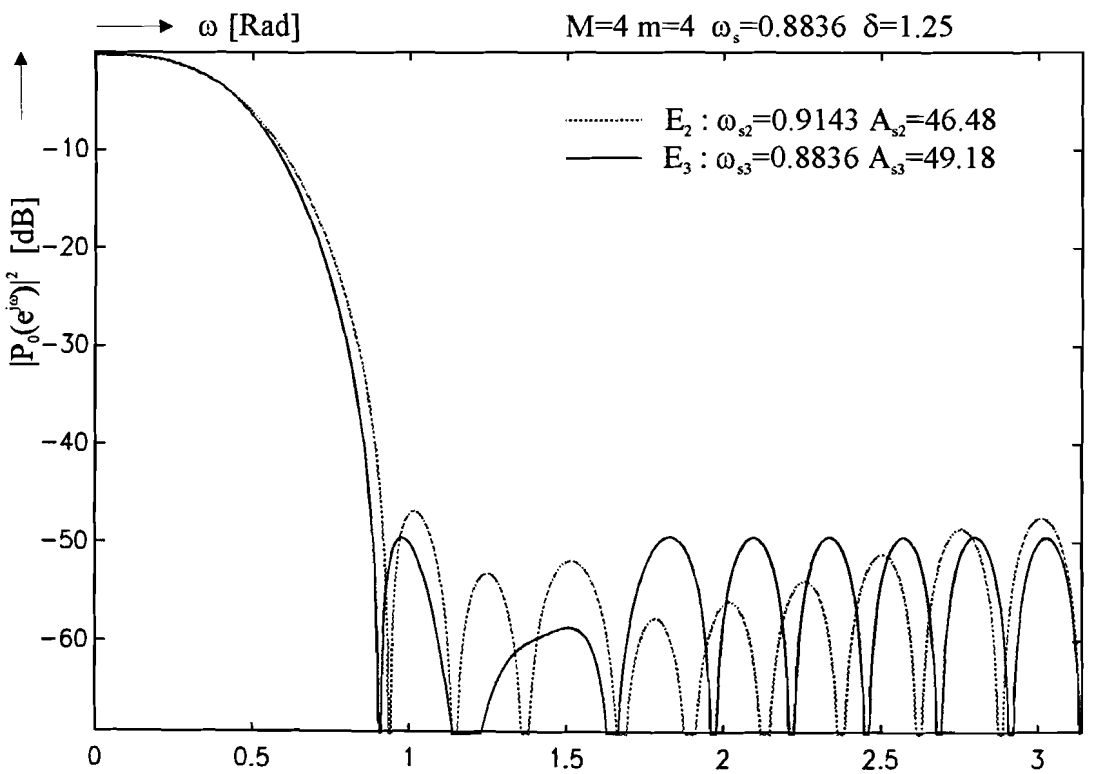
```
flipud(G);
:2:m;
,i)=-G(:,i);

tatus(1)==1, % floating-point
sp('Coefficient Quantisatie');
=flquant(G,N,Ex,status(2));
:if status(1)==2, % fixed-point
sp('Coefficient Quantisatie');
=fxquant(G,N,status(2),'max');


)(' ');disp('Verwerken van de samples door de transversale-structuur');disp('');

K]=size(X);

tatus(3), % maximaaal gedecimeerde filterbank
=zeros(M,(m-1)*2+K+1);
 status(1)==1
for i=1:M
  X1=zeros(m,2*m+K-1); X2=zeros(m,2*m+K-1);
  for j=1:m
    X1(j,j*2:j*2+K-1)=X(i,:);
    X2(j,j*2-1:(j-1)*2+K)=X(M+i,:);
  end
  Y(i,:)=..
  qmat_mul(G(i,:),X1,status(1:2),N,Ex)+qmat_mul(G(i+M,:),X2,status(1:2),N,Ex);
end
Y=flquant(Y,N,Ex,status(2));
y=reshape(flipud(Y),1,M*(m*2+K-1));
y=flquant(y*flquant(1/sqrt(2),N,Ex,status(2)),N,Ex,status(2));
lseif status(1)==2
for i=1:M
  X1=zeros(m,2*m+K-1); X2=zeros(m,2*m+K-1);
  for j=1:m
    X1(j,j*2:j*2+K-1)=X(i,:);
    X2(j,j*2-1:(j-1)*2+K)=X(M+i,:);
  end
  Y(i,:)=qmat_mul(G(i,:),X1,status(1:2),N)+qmat_mul(G(i+M,:),X2,status(1:2),N);
end
Y=fxquant(Y*2^(-N),N,status(2));
y=reshape(flipud(Y),1,M*(m*2+K-1));
lse
for i=1:M
  X1=zeros(m,2*m+K-1); X2=zeros(m,2*m+K-1);
  for j=1:m
    X1(j,j*2:j*2+K-1)=X(i,:);
    X2(j,j*2-1:(j-1)*2+K)=X(M+i,:);
  end
  Y(i,:)=G(i,:)*X1+G(M+i,:)*X2;
end
y=reshape(flipud(Y),1,M*(m*2+K-1))/sqrt(2);
nd
e    % niet gedecimeerde filterbank
 status(1)==1
for i=1:M
  X1=zeros(m,M*(m*2-1)+K); X2=zeros(m,M*(m*2-1)+K);
  for j=1:m
    X1(j,(j-1)*2*M+1+M:(j-1)*2*M+K+M)=X(i,:);
    X2(j,(j-1)*2*M+1:(j-1)*2*M+K)=X(M+i,:);
  end
  Y(i,M-i+1:m*2*M+K-i)=..
at_mul(G(i,:),X1,status(1:2),N,Ex)+qmat_mul(G(i+M,:),X2,status(1:2),N,Ex);
end
Y=flquant(Y,N,Ex,status(2));
y=flquant(qsum(Y,status(1:2),2*N,2*Ex),N,Ex,status(2));
y=flquant(y*flquant(1/sqrt(2*M),N,Ex,status(2)),N,Ex,status(2));
lseif status(1)==2, % fixed-point
for i=1:M
  X1=zeros(m,M*(m*2-1)+K); X2=zeros(m,M*(m*2-1)+K);
  for j=1:m
    X1(j,(j-1)*2*M+1+M:(j-1)*2*M+K+M)=X(i,:);
    X2(j,(j-1)*2*M+1:(j-1)*2*M+K)=X(M+i,:);
  end
  Y(i,M-i+1:m*2*M+K-i)=..
qmat_mul(G(i,:),X1,status(1:2),N)+qmat_mul(G(i+M,:),X2,status(1:2),N);
  end
  Y=fxquant(Y*2^(-N),N,status(2));
  y=fxquant(qsum(Y*2^N,status(1:2),2*N)*2^(-N),N,status(2));
 else
  for i=1:M
    X1=zeros(m,M*(m*2-1)+K); X2=zeros(m,M*(m*2-1)+K);
    for j=1:m
      X1(j,(j-1)*2*M+1+M:(j-1)*2*M+K+M)=X(i,:);
      X2(j,(j-1)*2*M+1:(j-1)*2*M+K)=X(M+i,:);
    end
    Y(i,M-i+1:m*2*M+K-i)=G(i,:)*X1+G(M+i,:)*X2;
  end
  y=sum(Y)/sqrt(2*M);
 end
end
y=y*(-1)^(m-1);
```

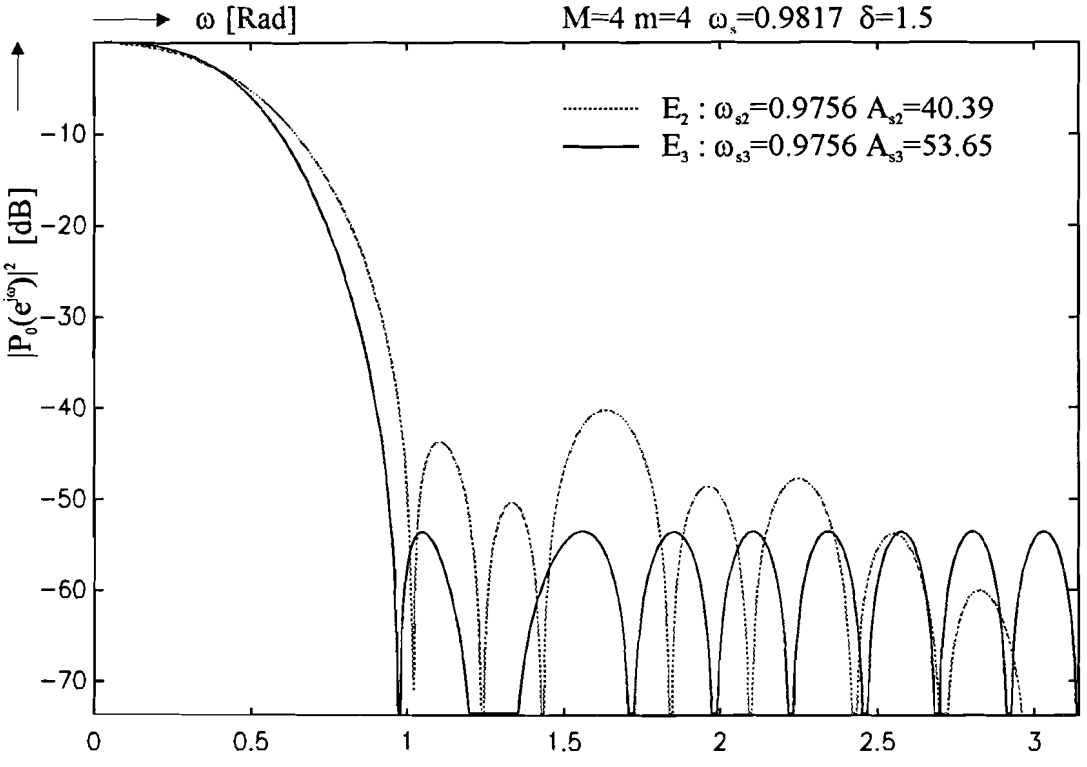
B.1 Frequentie spectra van de prototypen met M=4 en m=4 voor verschillende stopbandgrenzen.



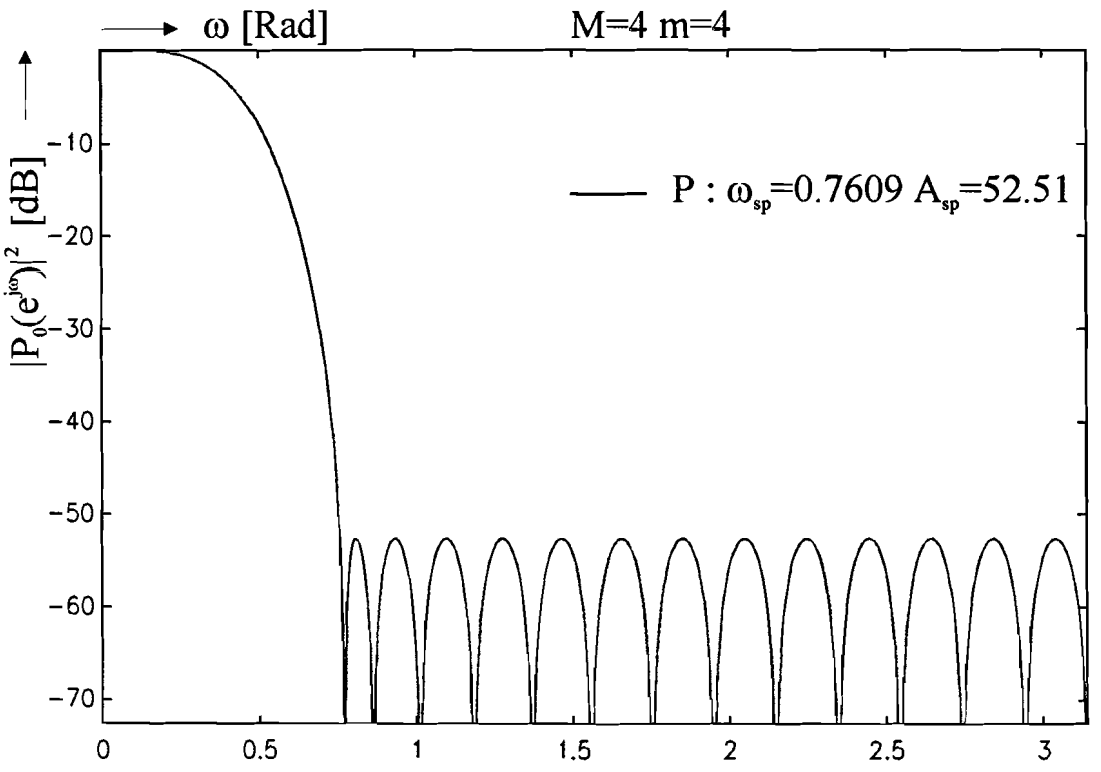



ω [Rad]
M=4 m=4 ω_s=0.7854 δ=1
E_2 : ω_s2=0.8222 A_s2=43.97
E_3 : ω_s3=0.7854 A_s3=42.16
|P_0(e^jω)|^2 [dB]
-10
-20
-30
-40
-50
-60
0
0.5
1
1.5
2
2.5
3

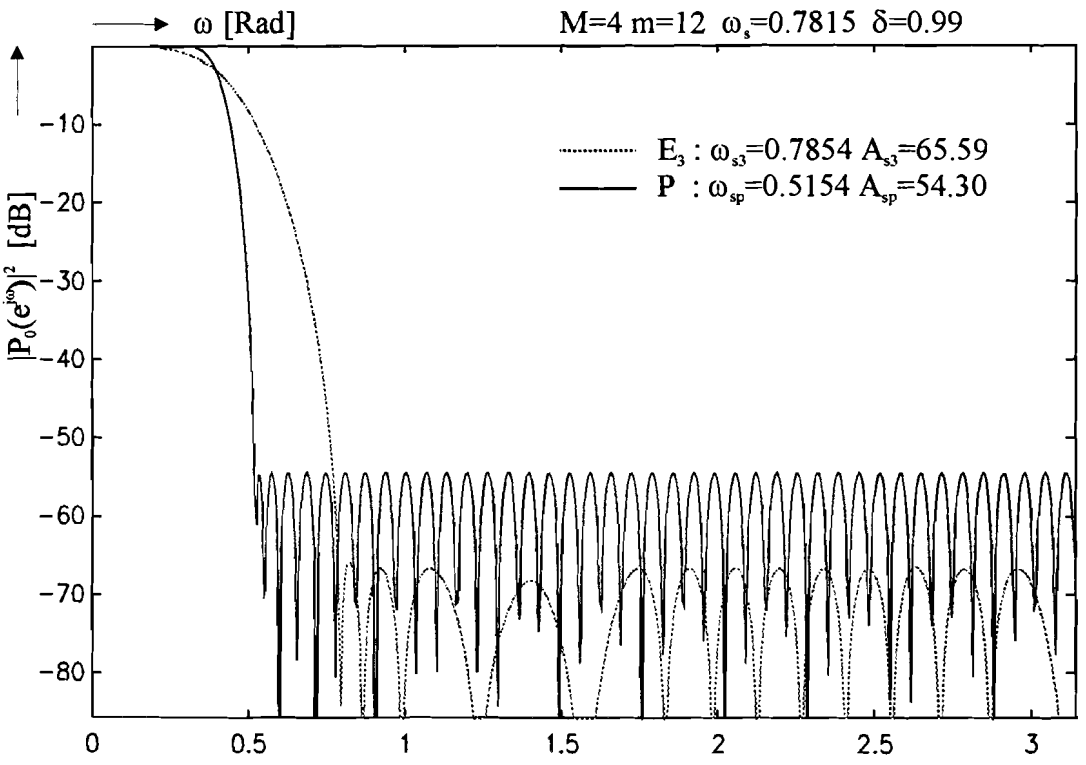

ω [Rad]
M=4 m=4 ω_s=0.8836 δ=1.25
E_2 : ω_s2=0.9143 A_s2=46.48
E_3 : ω_s3=0.8836 A_s3=49.18
|P_0(e^jω)|^2 [dB]
-10
-20
-30
-40
-50
-60
0
0.5
1
1.5
2
2.5
3

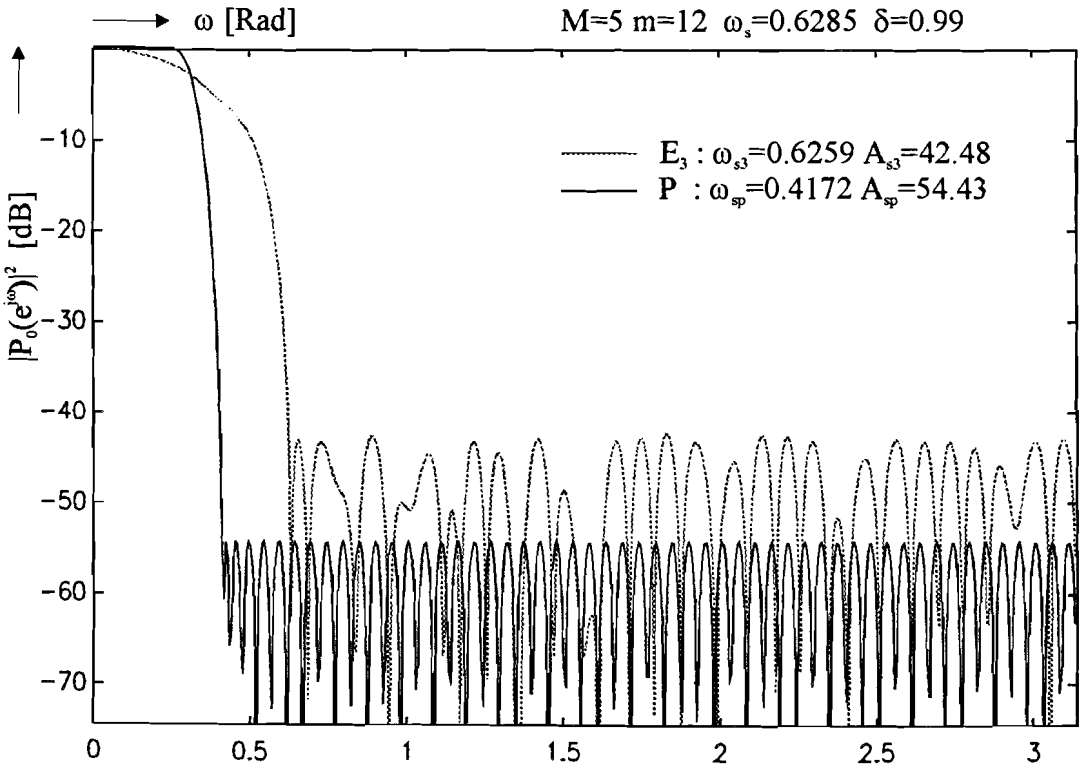

ω [Rad]
M=4 m=4 ω_s=0.9817 δ=1.5
E_2 : ω_s2=0.9756 A_s2=40.39
E_3 : ω_s3=0.9756 A_s3=53.65
|P_0(e^jω)|^2 [dB]
-10
-20
-30
-40
-50
-60
-70
0
0.5
1
1.5
2
2.5
3

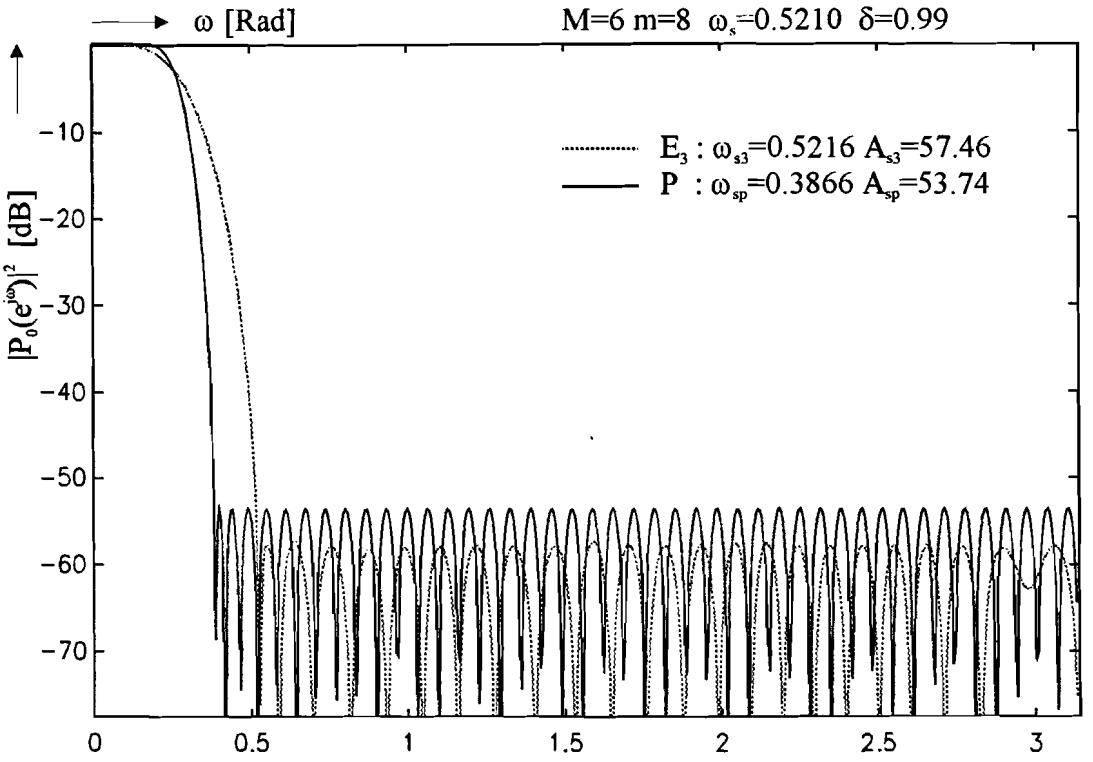
B.2 Frequentie spectra van een prototype met M=4 en m=4 ontworpen m.b.v. REMEZ.

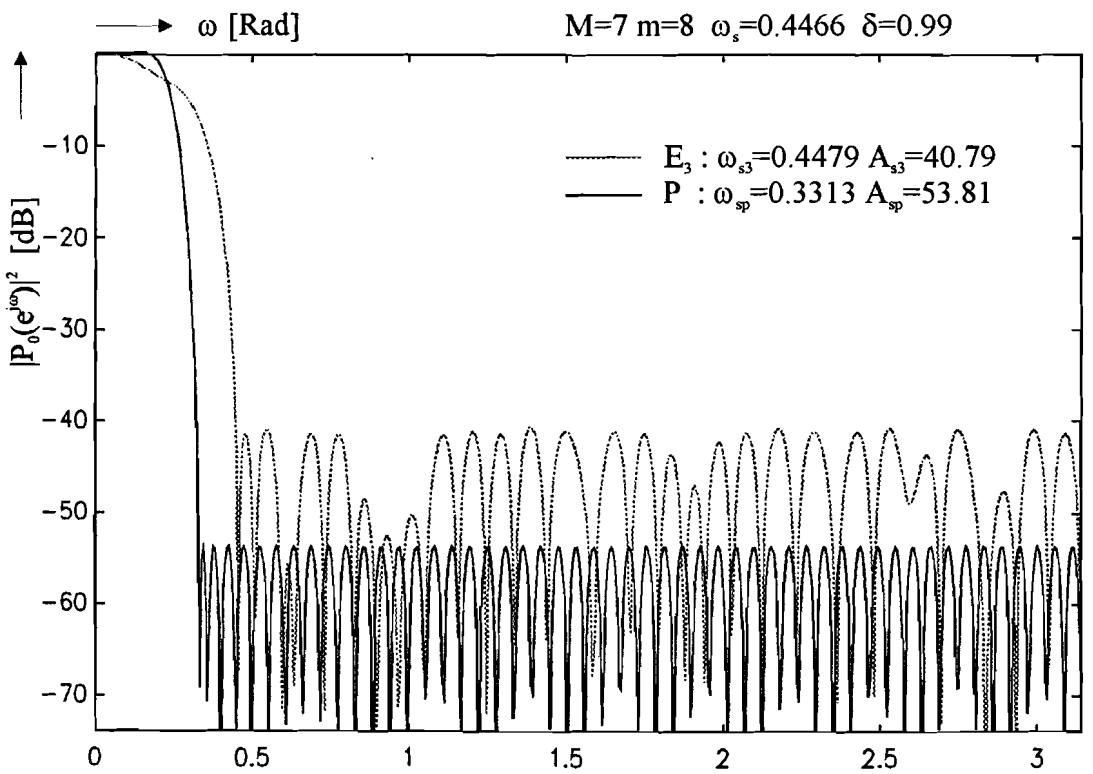


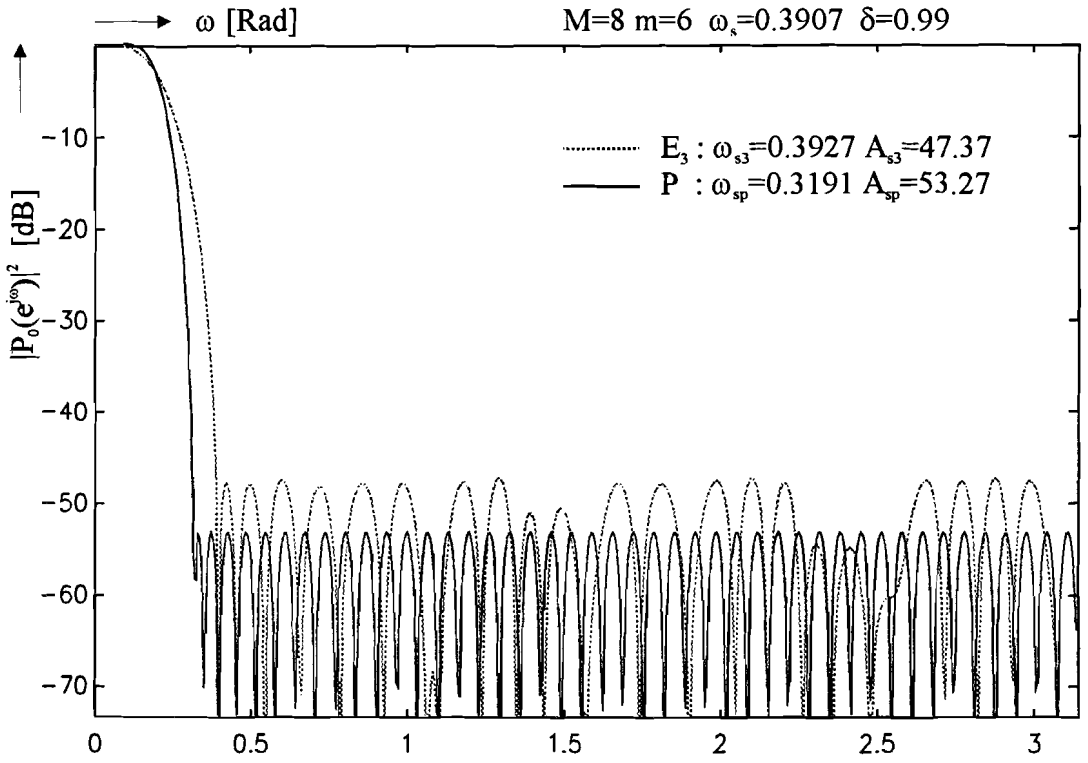
B.3 Frequentie spectra van prototypen met 24 lattice coefficienten.

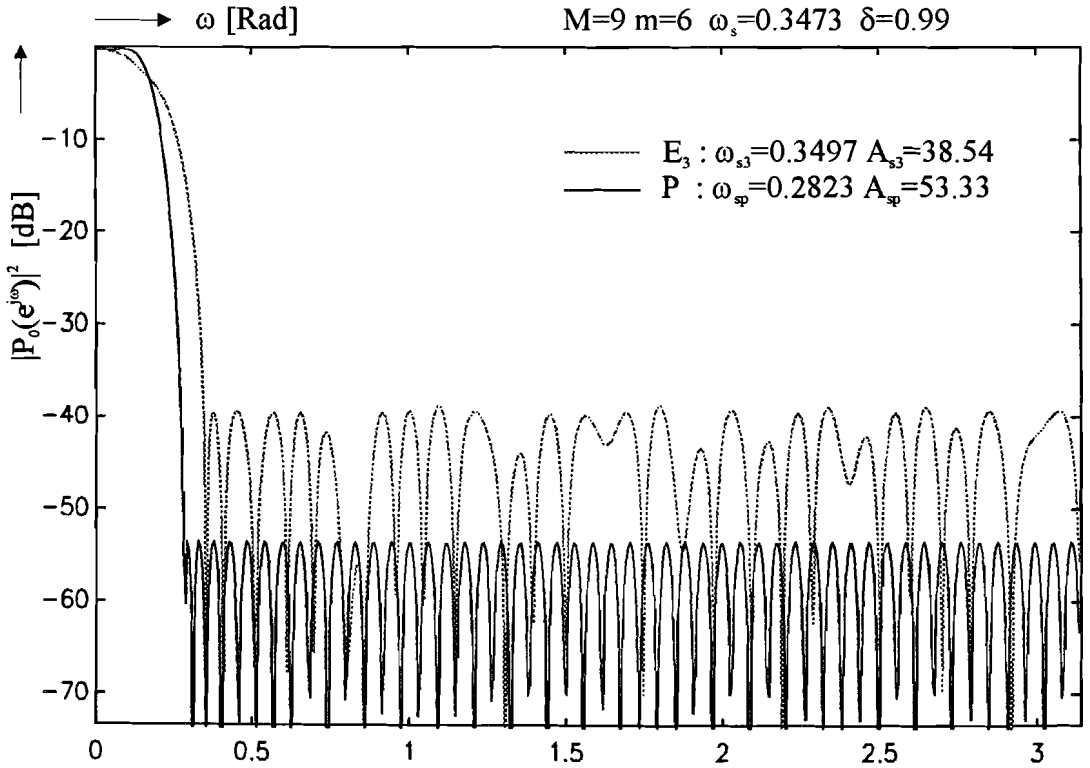


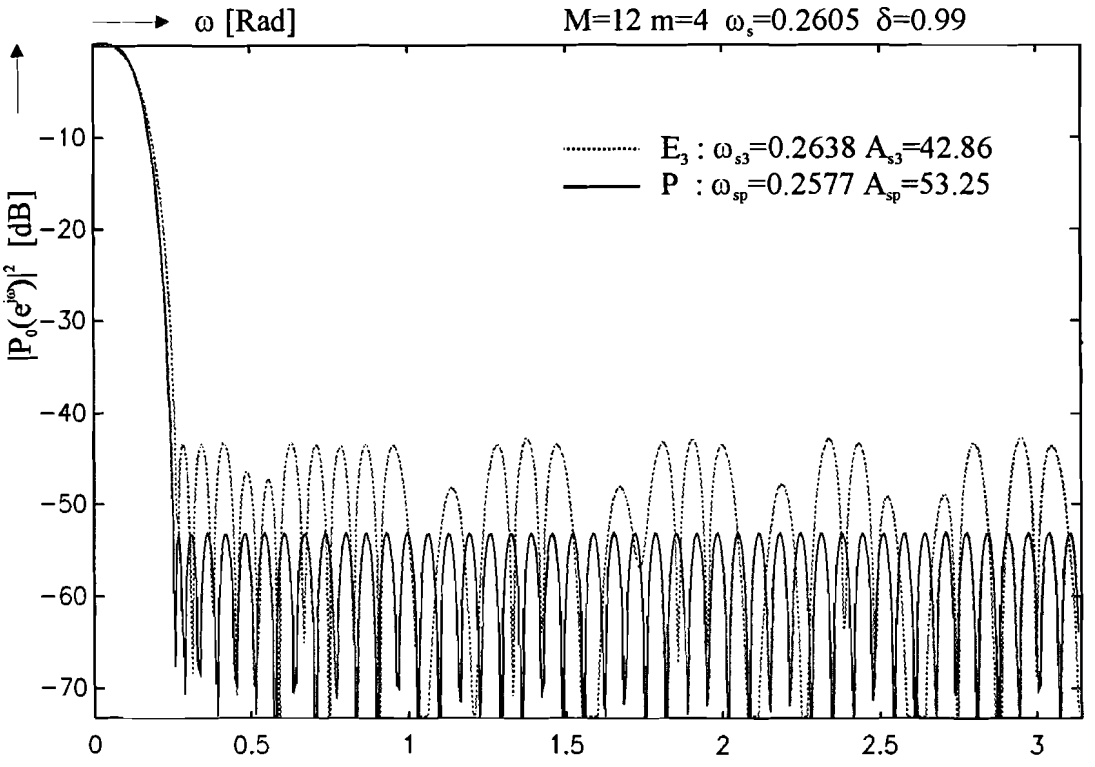


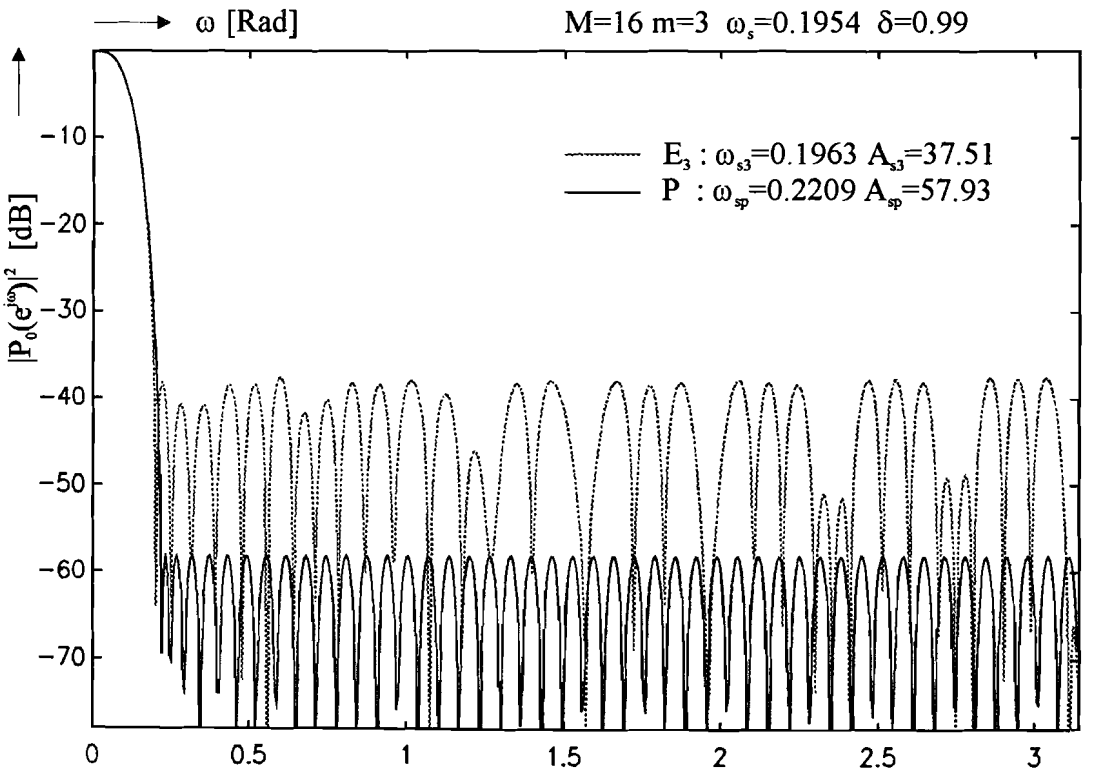

ω [Rad]
M=6 m=8 ω_s=0.5210 δ=0.99
E_3 : ω_s3=0.5216 A_s3=57.46
P : ω_sp=0.3866 A_sp=53.74
|P_0(e^jω)|^2 [dB]
-10
-20
-30
-40
-50
-60
-70
0
0.5
1
1.5
2
2.5
3


ω [Rad]
M=7 m=8 ω_s=0.4466 δ=0.99
E_3 : ω_s3=0.4479 A_s3=40.79
P : ω_sp=0.3313 A_sp=53.81
|P_0(e^jω)|^2 [dB]
-10
-20
-30
-40
-50
-60
-70
0
0.5
1
1.5
2
2.5
3

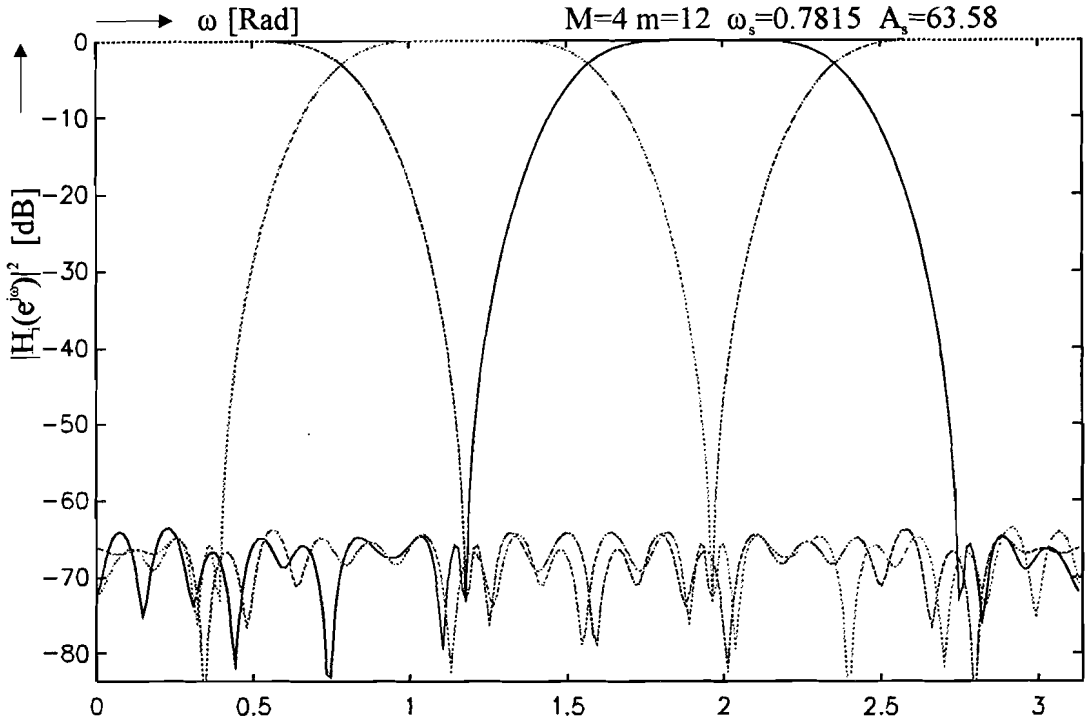

ω [Rad]
M=8 m=6 ω_s=0.3907 δ=0.99
E3 : ω_s3=0.3927 A_s3=47.37
P : ω_sp=0.3191 A_sp=53.27
|P0(e^jω)|^2 [dB]
-10
-20
-30
-40
-50
-60
-70
0
0.5
1
1.5
2
2.5
3

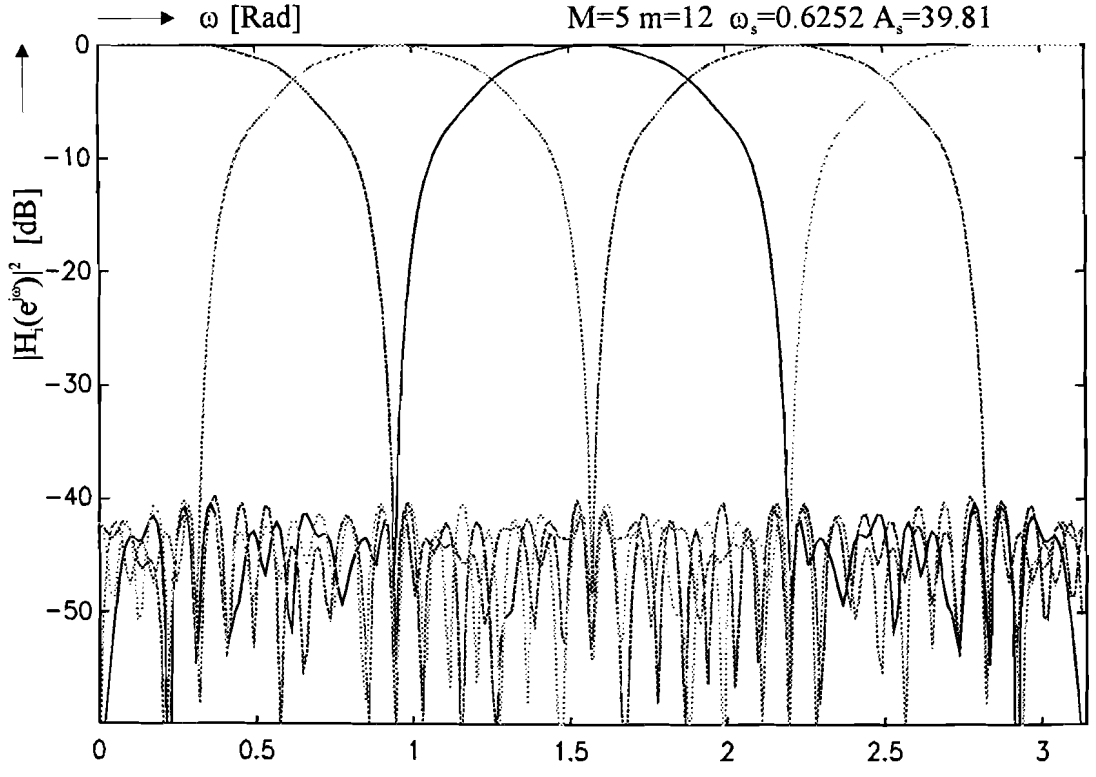

ω [Rad]
M=9 m=6 ω_s=0.3473 δ=0.99
E3 : ω_s3=0.3497 A_s3=38.54
P : ω_sp=0.2823 A_sp=53.33
|P0(e^jω)|^2 [dB]
-10
-20
-30
-40
-50
-60
-70
0
0.5
1
1.5
2
2.5
3

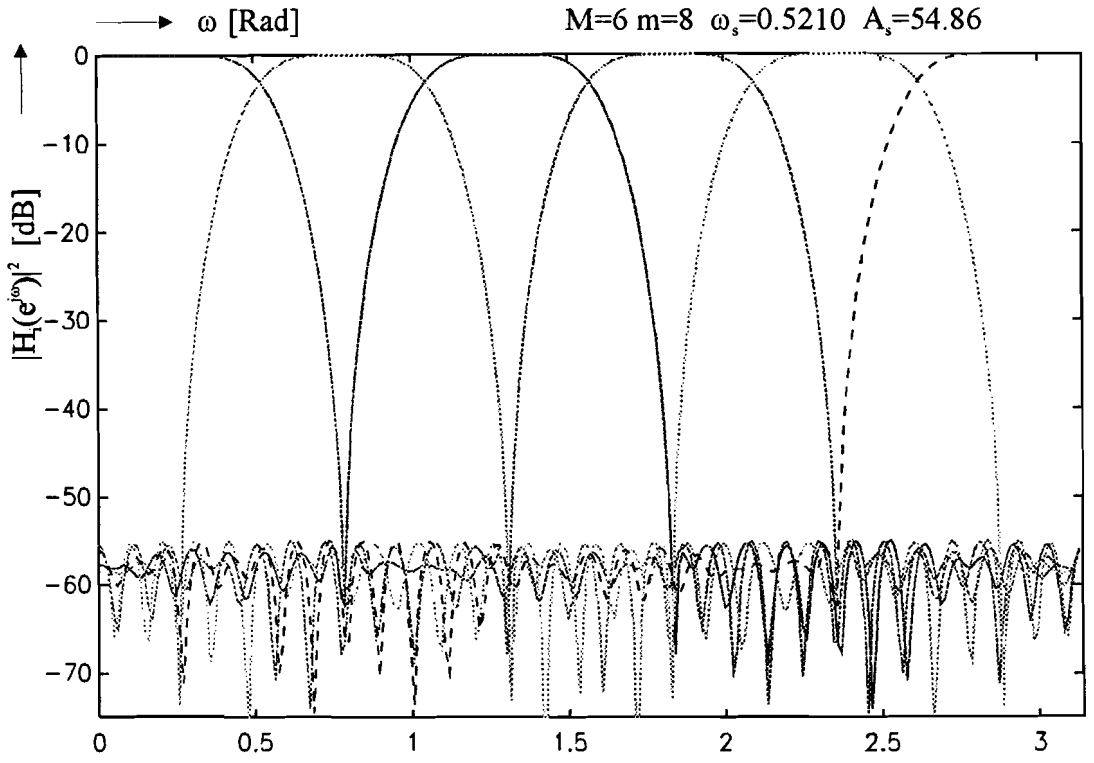

ω [Rad]
M=12 m=4 ω_s=0.2605 δ=0.99
E_3 : ω_s3=0.2638 A_s3=42.86
P : ω_sp=0.2577 A_sp=53.25
|P_0(e^jω)|^2 [dB]
-10
-20
-30
-40
-50
-60
-70
0
0.5
1
1.5
2
2.5
3

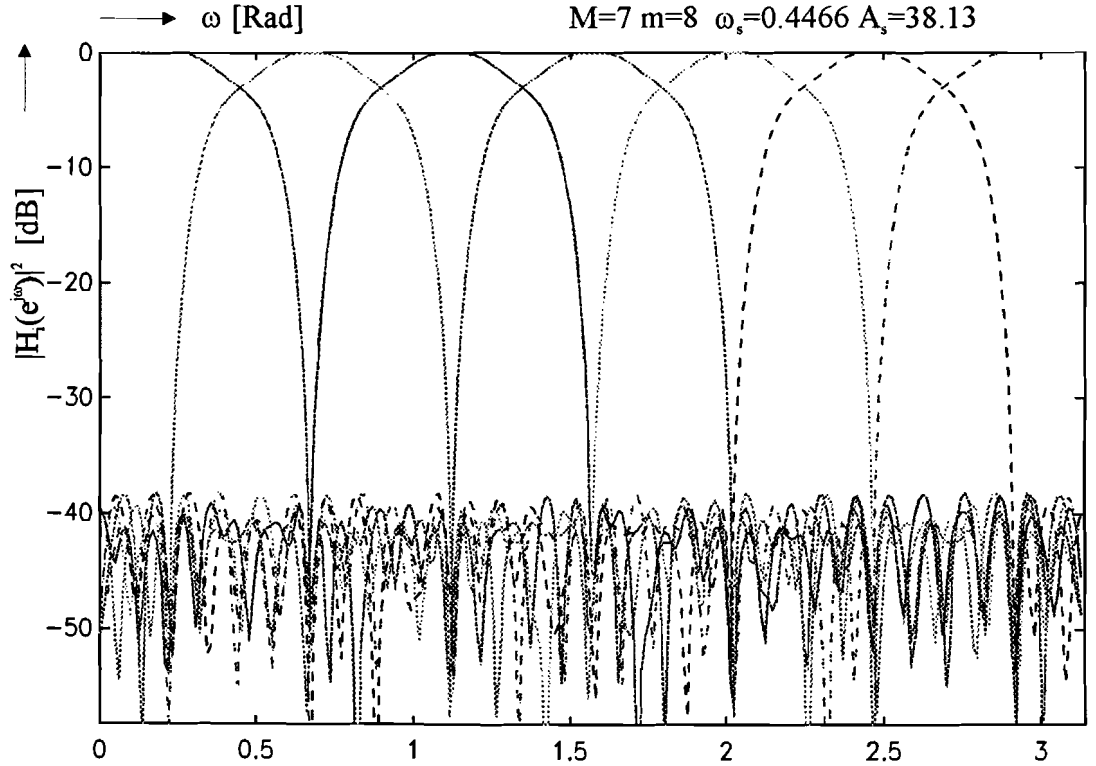

ω [Rad]
M=16 m=3 ω_s=0.1954 δ=0.99
E_3 : ω_s3=0.1963 A_s3=37.51
P : ω_sp=0.2209 A_sp=57.93
|P_0(e^jω)|^2 [dB]
-10
-20
-30
-40
-50
-60
-70
0
0.5
1
1.5
2
2.5
3

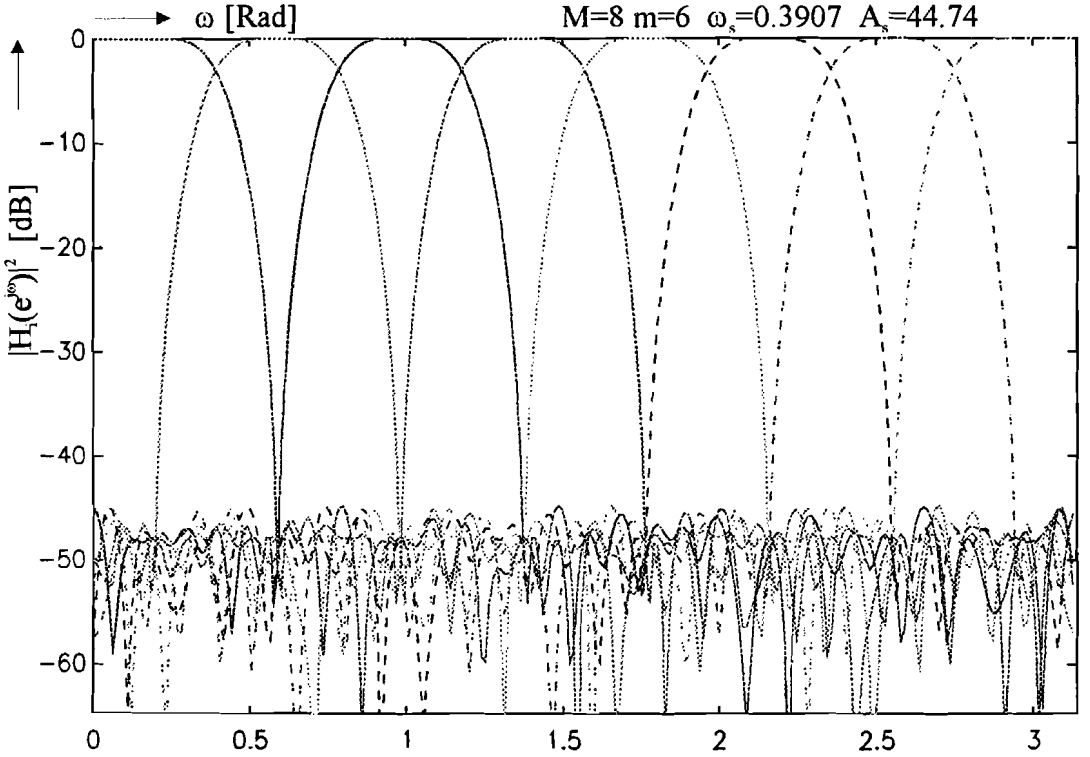
B.4 Frequentie spectra van analyse filters uitgaande van prototypen met 24 lattice coefficienten.

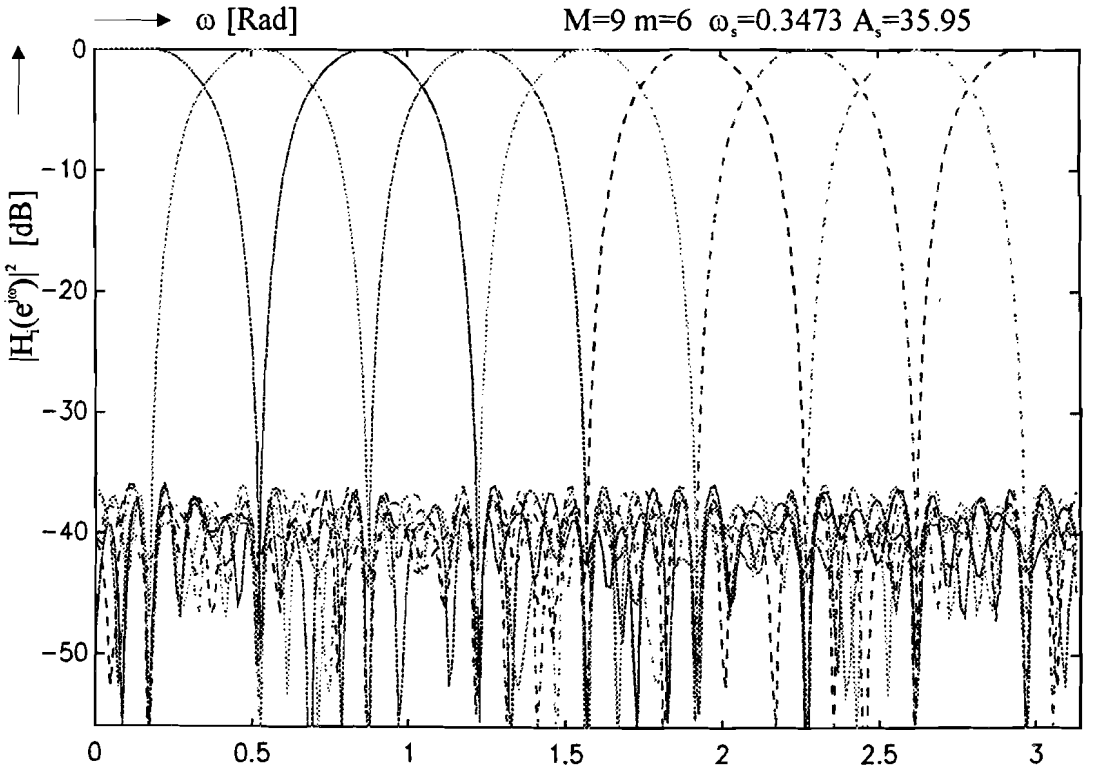


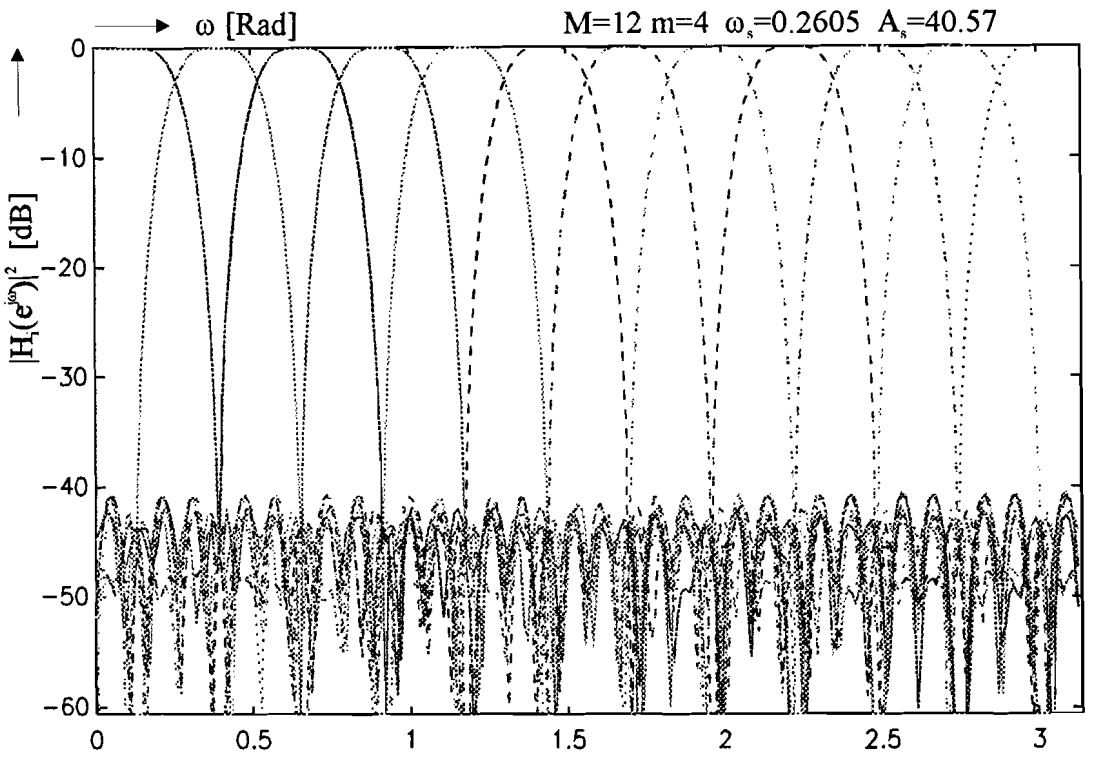


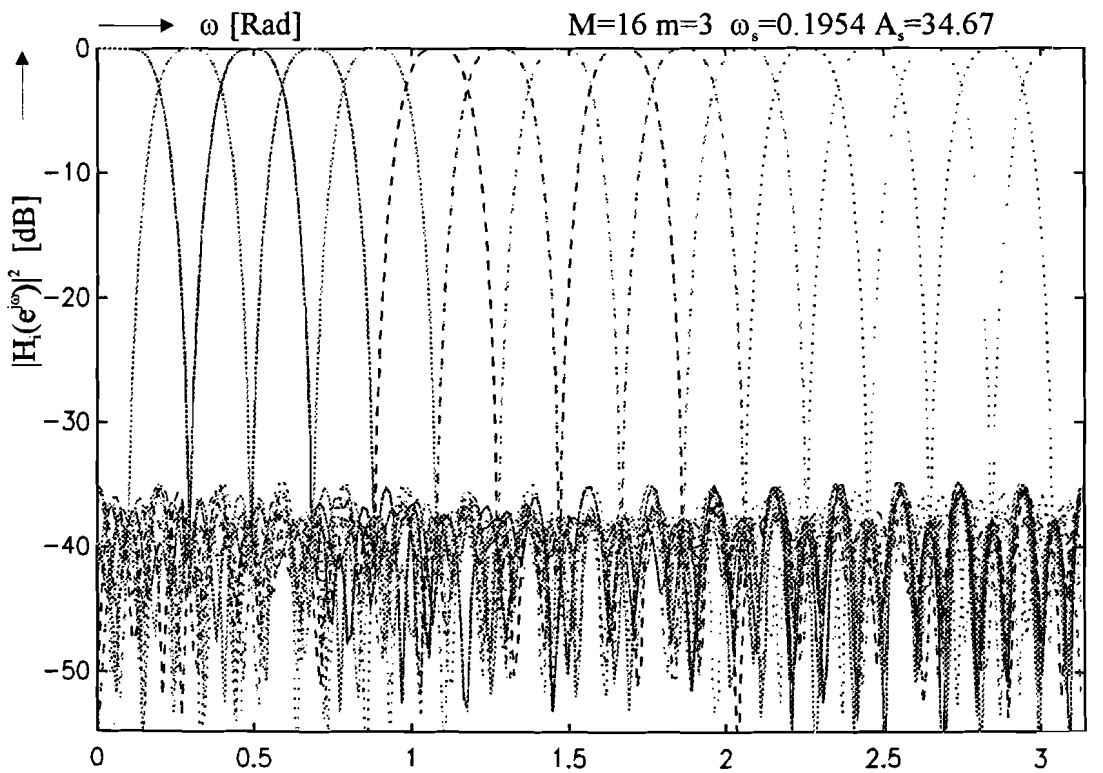

ω [Rad]
M=6 m=8 ω_s=0.5210 A_s=54.86
|H_1(e^jω)|^2 [dB]
0
-10
-20
-30
-40
-50
-60
-70
0
0.5
1
1.5
2
2.5
3


ω [Rad]
M=7 m=8 ω_s=0.4466 A_s=38.13
|H_1(e^jω)|^2 [dB]
0
-10
-20
-30
-40
-50
0
0.5
1
1.5
2
2.5
3

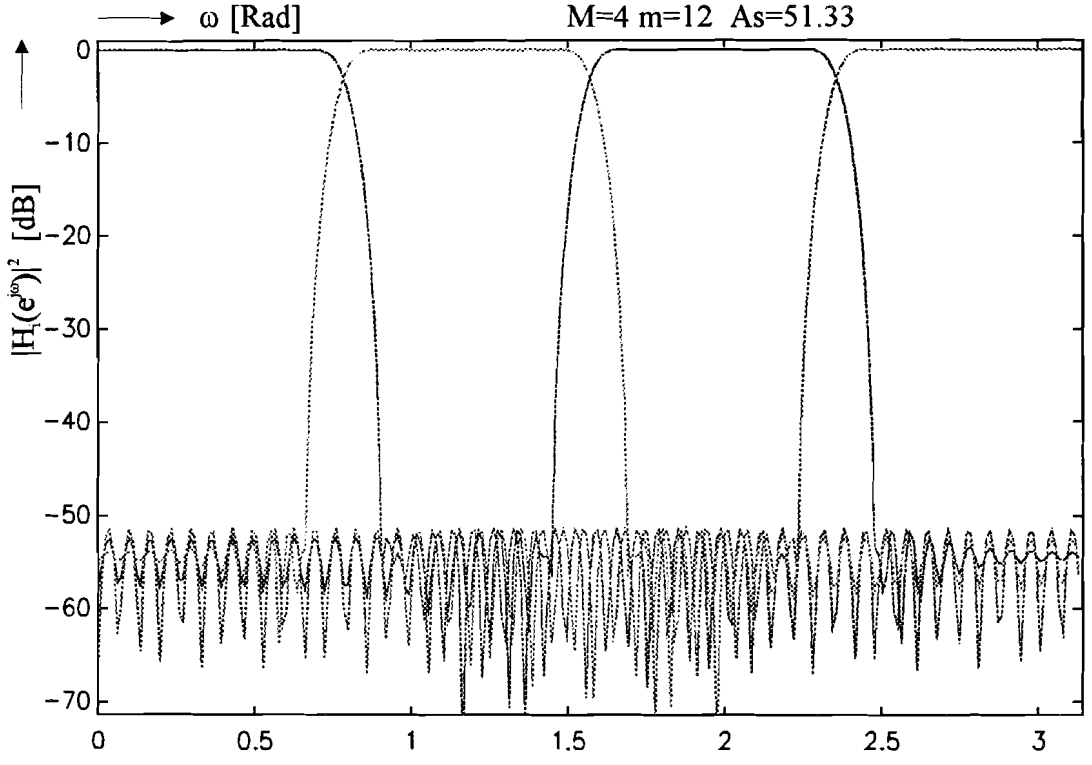

ω [Rad]
M=8 m=6 ω_s=0.3907 A_s=44.74
|H_1(e^jω)|^2 [dB]
0
-10
-20
-30
-40
-50
-60
0
0.5
1
1.5
2
2.5
3

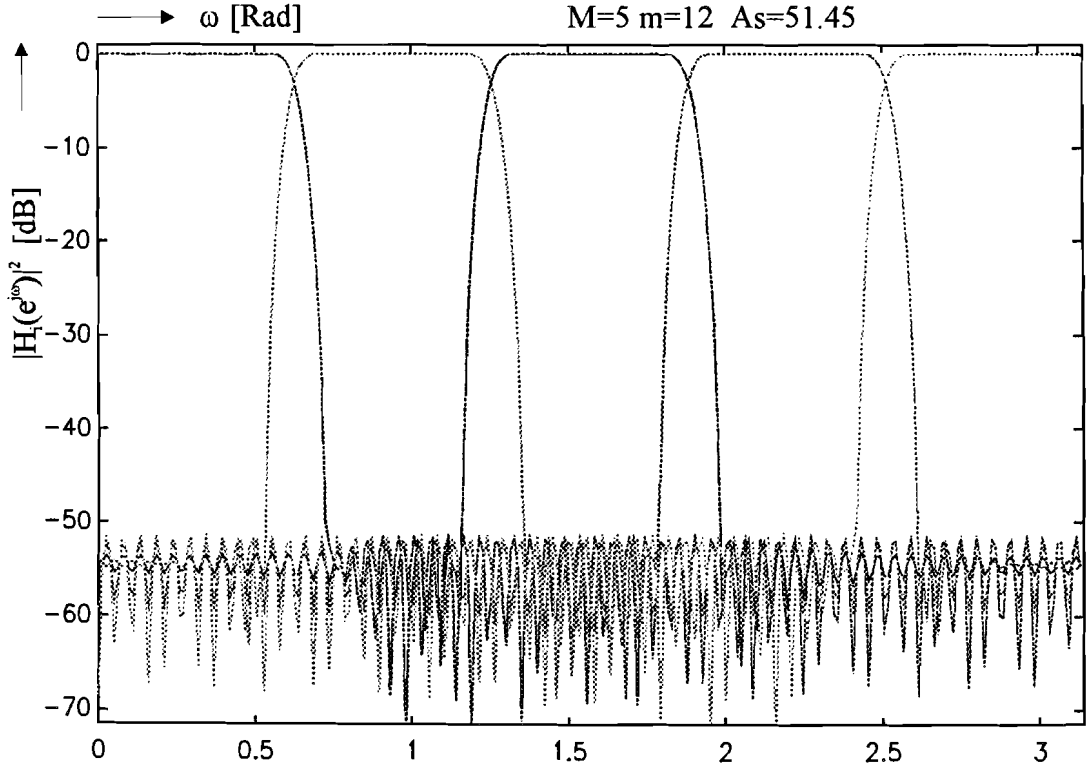

ω [Rad]
M=9 m=6 ω_s=0.3473 A_s=35.95
|H_1(e^jω)|^2 [dB]
0
-10
-20
-30
-40
-50
0
0.5
1
1.5
2
2.5
3

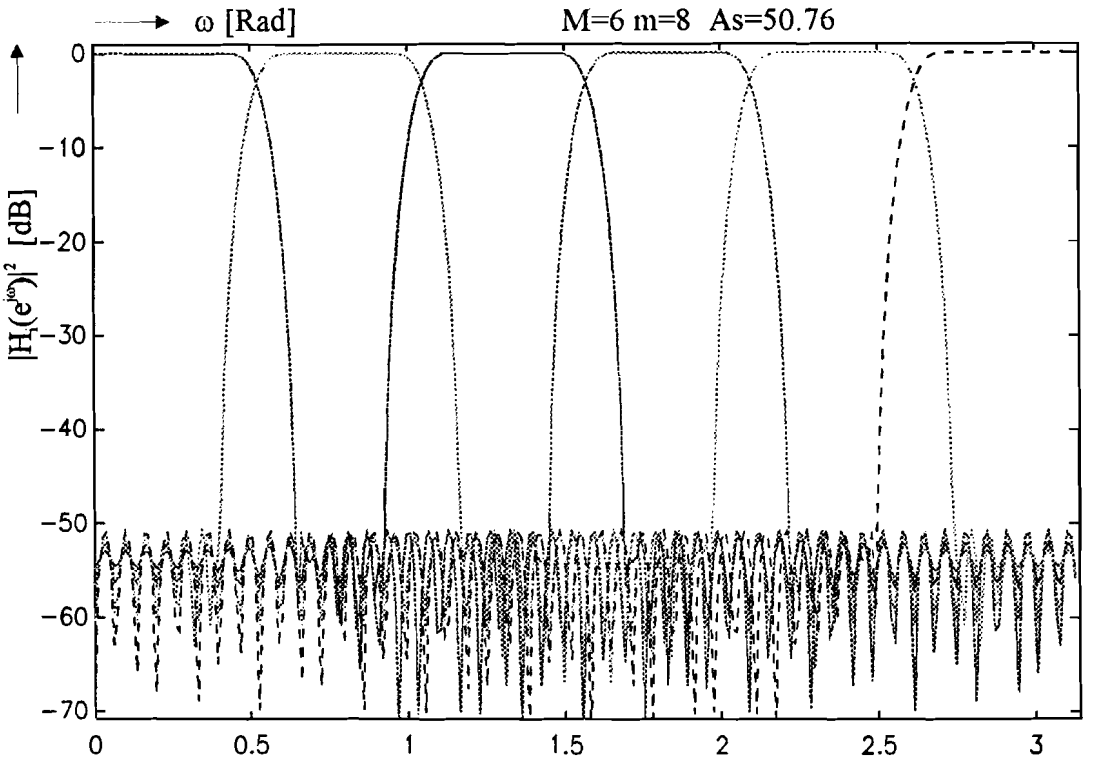

ω [Rad]
M=12 m=4 ω_s=0.2605 A_s=40.57
|H_1(e^jω)|^2 [dB]
0
-10
-20
-30
-40
-50
-60
0
0.5
1
1.5
2
2.5
3

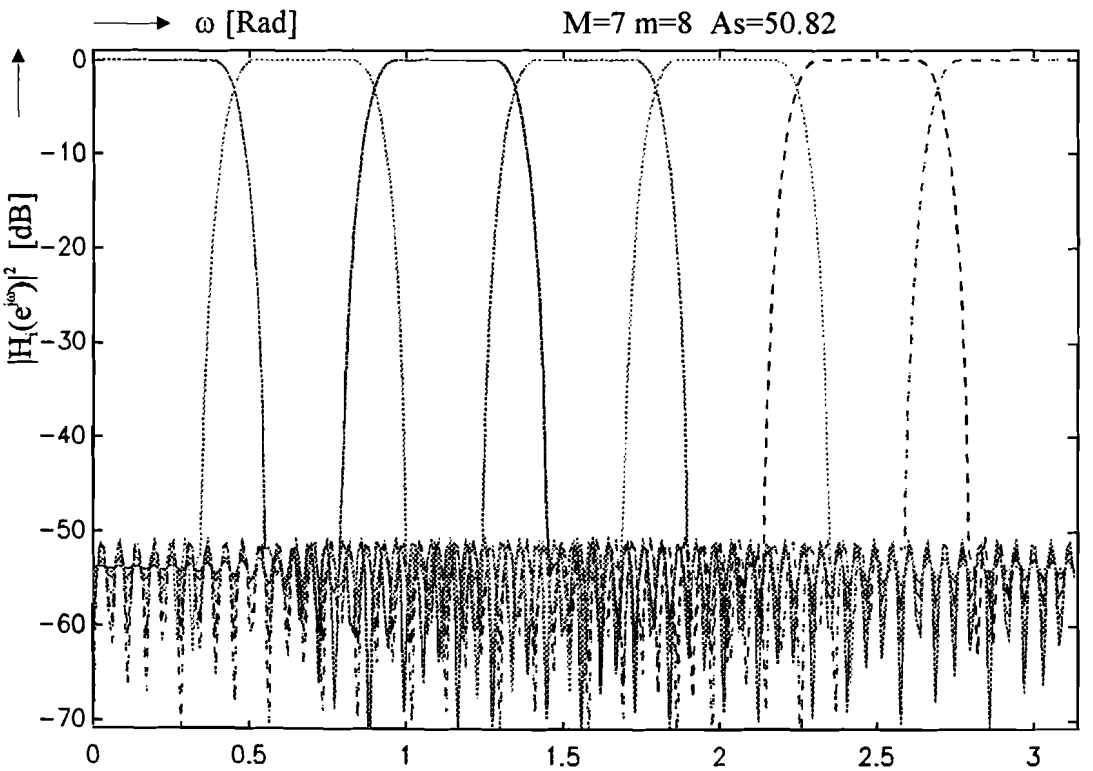

ω [Rad]
M=16 m=3 ω_s=0.1954 A_s=34.67
|H_1(e^jω)|^2 [dB]
0
-10
-20
-30
-40
-50
0
0.5
1
1.5
2
2.5
3

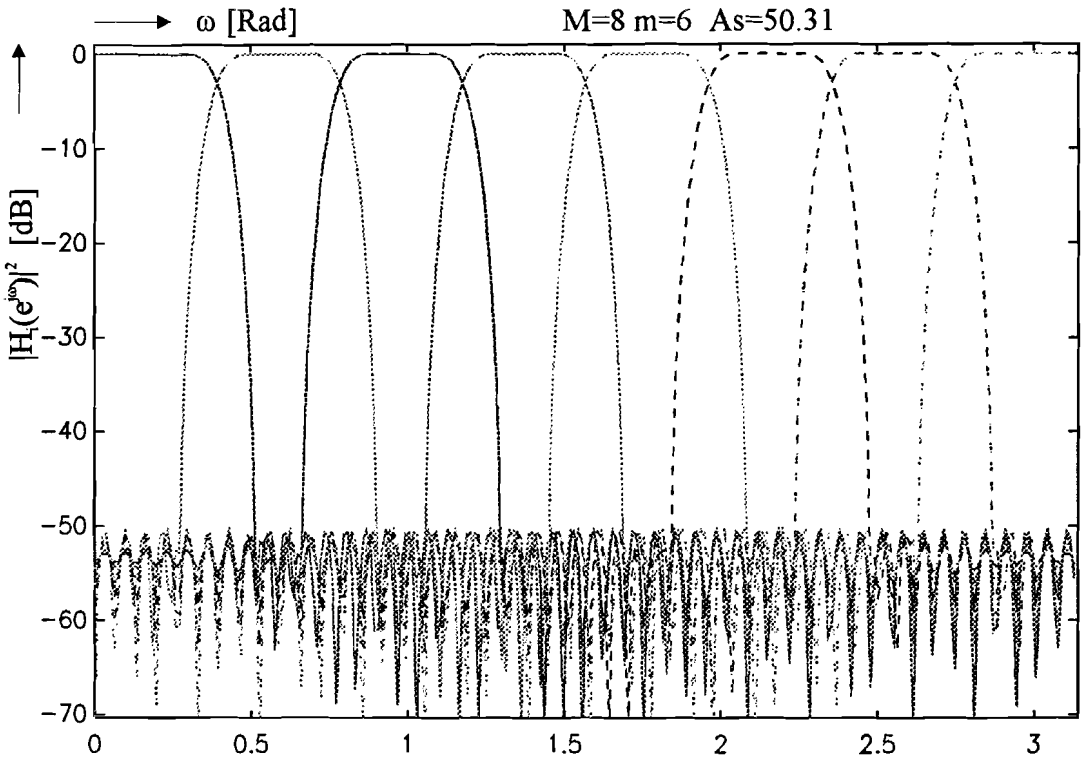
B.5 Frequentie spectra van analyse filters uitgaande van prototypen ontworpen m.b.v. REMEZ.

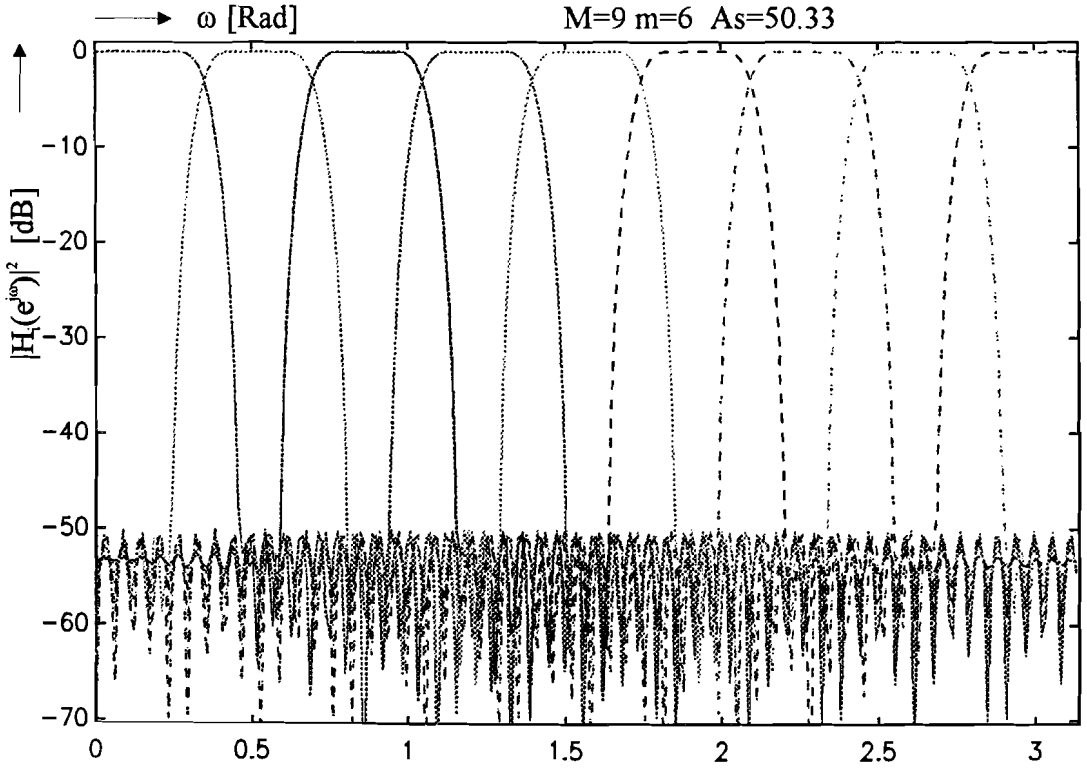


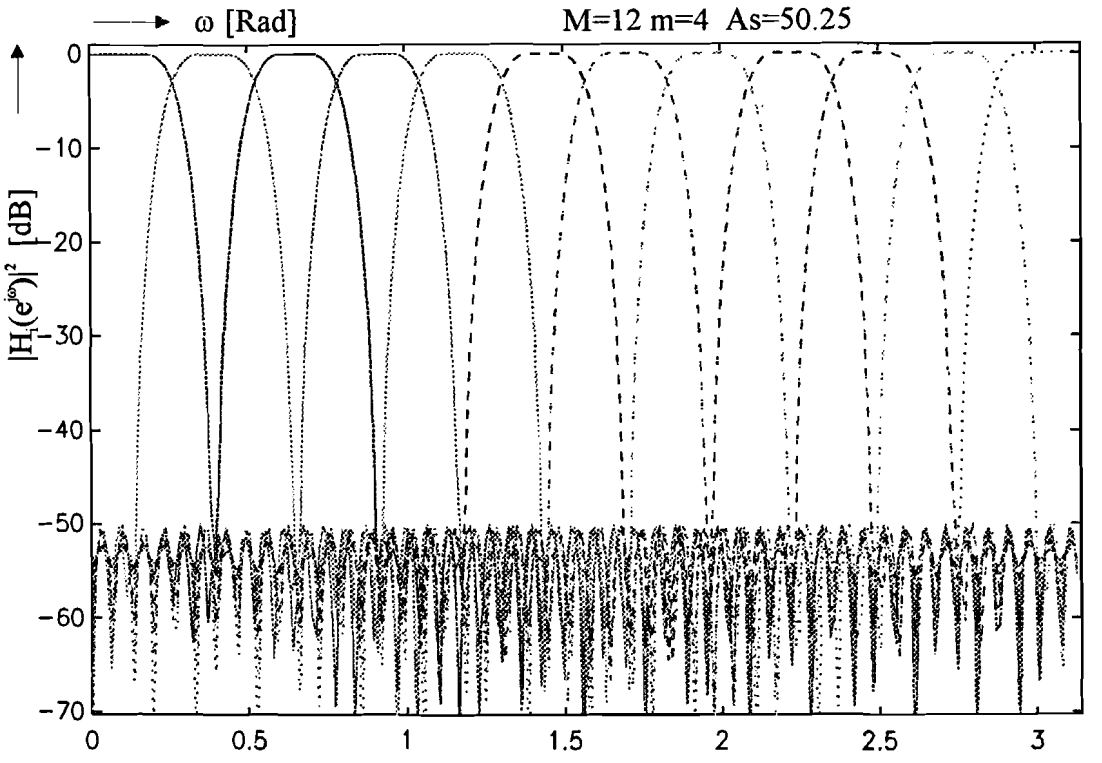


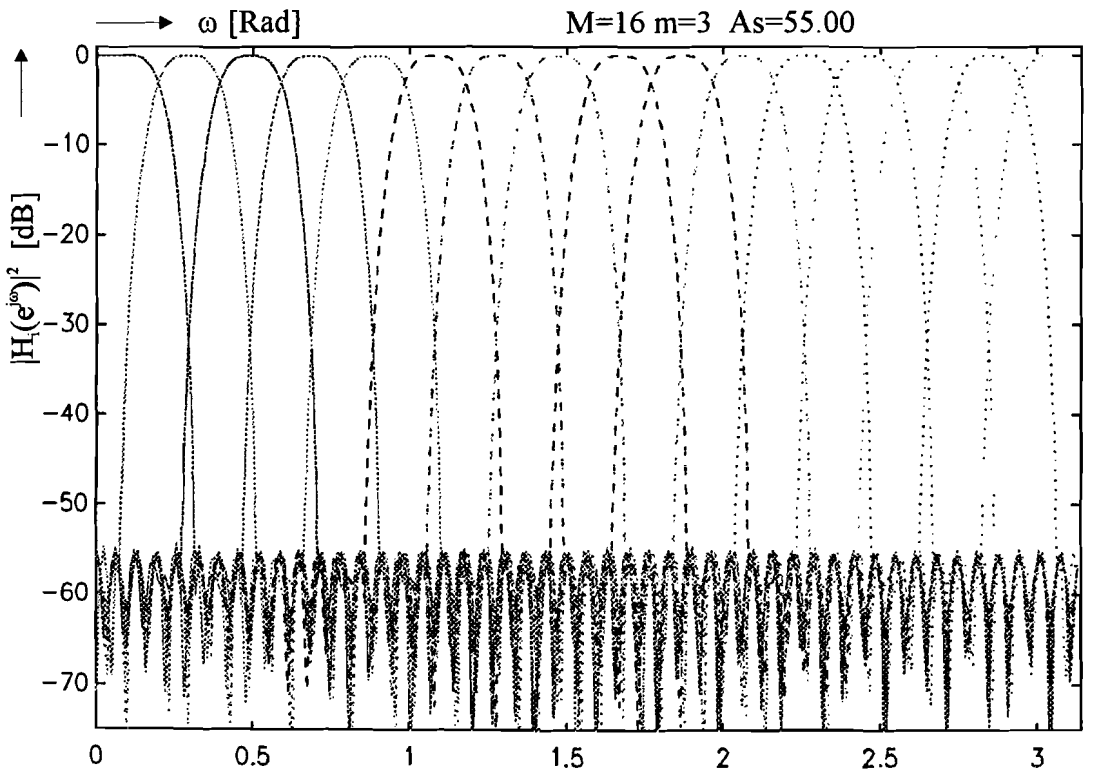

ω [Rad]
M=6 m=8 As=50.76
$|H_1(e^{j\omega})|^2$ [dB]
0
-10
-20
-30
-40
-50
-60
-70
0
0.5
1
1.5
2
2.5
3


ω [Rad]
M=7 m=8 As=50.82
$|H_1(e^{j\omega})|^2$ [dB]
0
-10
-20
-30
-40
-50
-60
-70
0
0.5
1
1.5
2
2.5
3

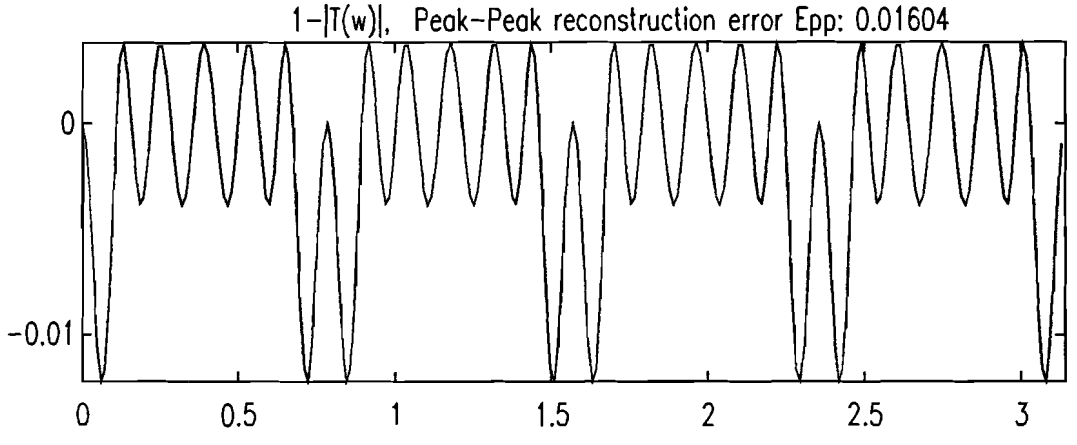

ω [Rad]
M=8 m=6 As=50.31
$|H_1(e^{j\omega})|^2$ [dB]
0
-10
-20
-30
-40
-50
-60
-70
0
0.5
1
1.5
2
2.5
3

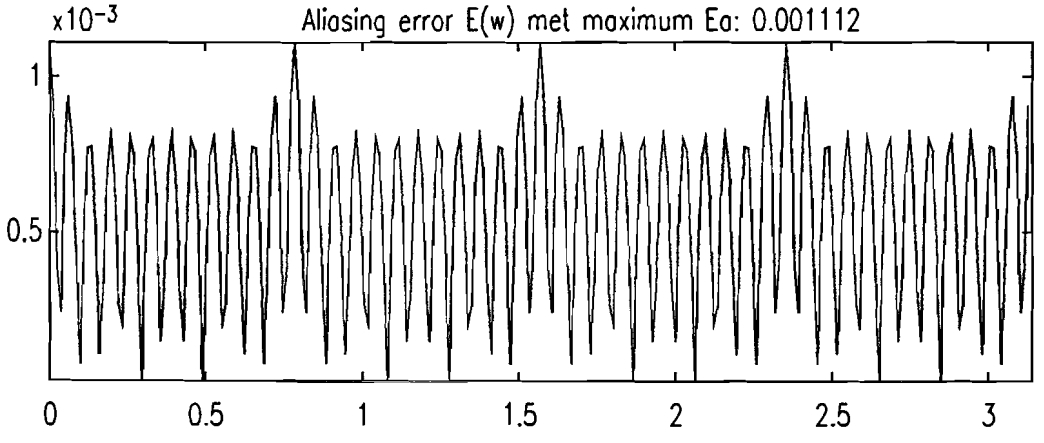

ω [Rad]
M=9 m=6 As=50.33
$|H_1(e^{j\omega})|^2$ [dB]
0
-10
-20
-30
-40
-50
-60
-70
0
0.5
1
1.5
2
2.5
3


ω [Rad]
M=12 m=4 As=50.25
$|H_1(e^{j\omega})|^2$ [dB]
0
-10
-20
-30
-40
-50
-60
-70
0
0.5
1
1.5
2
2.5
3

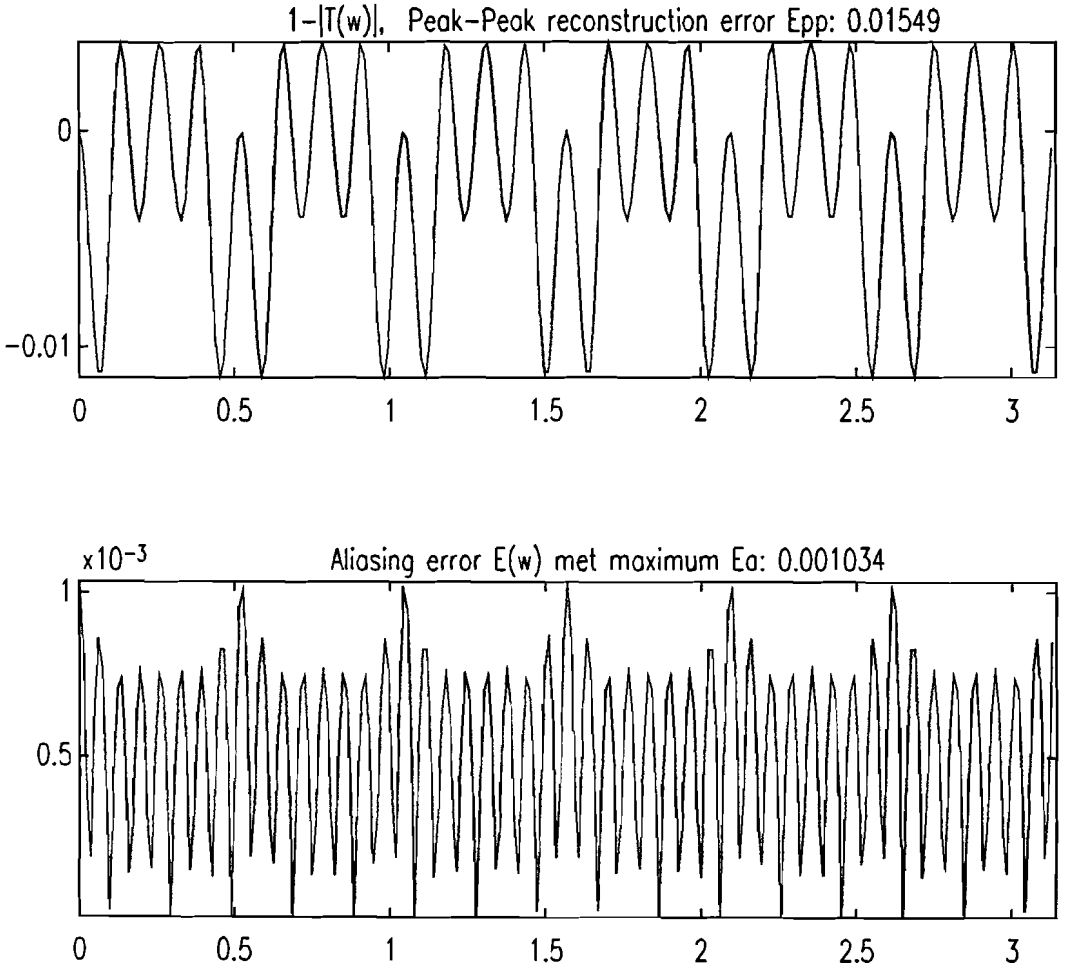

ω [Rad]
M=16 m=3 As=55.00
$|H_1(e^{j\omega})|^2$ [dB]
0
-10
-20
-30
-40
-50
-60
-70
0
0.5
1
1.5
2
2.5
3

B.6 Frequentie spectra van de distorsie functie en aliasing error van een filterbank met transversale stuctuur en een REMEZ prototype voor M=4 en m=12.

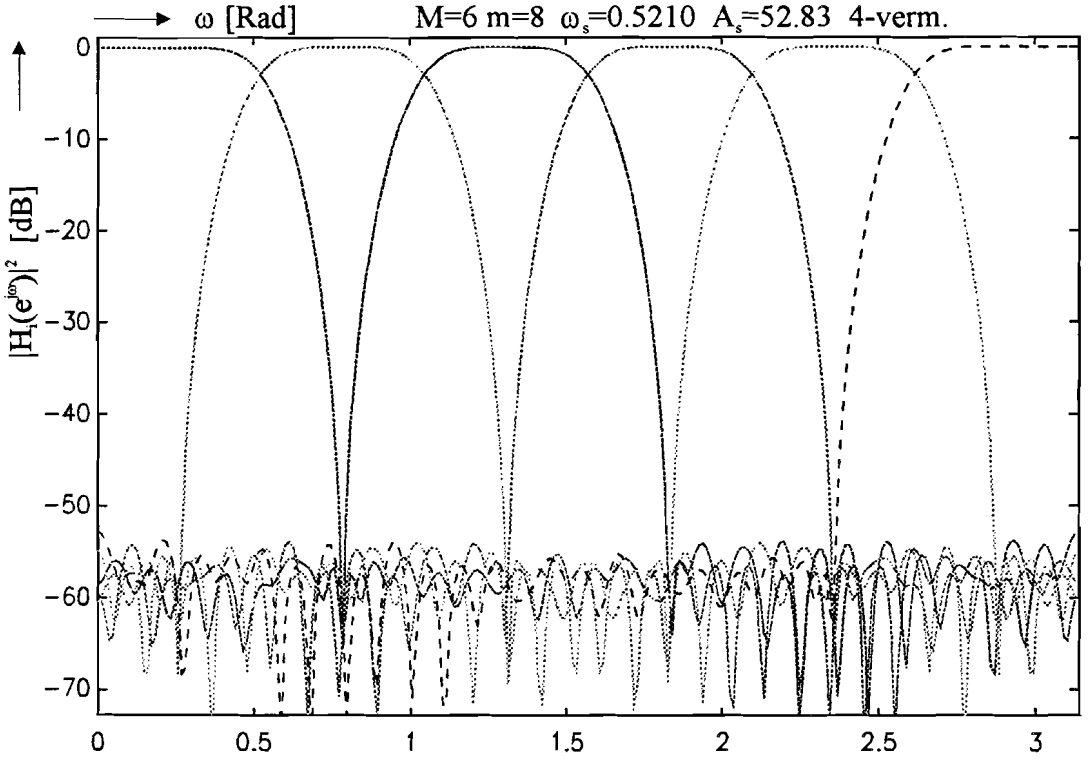


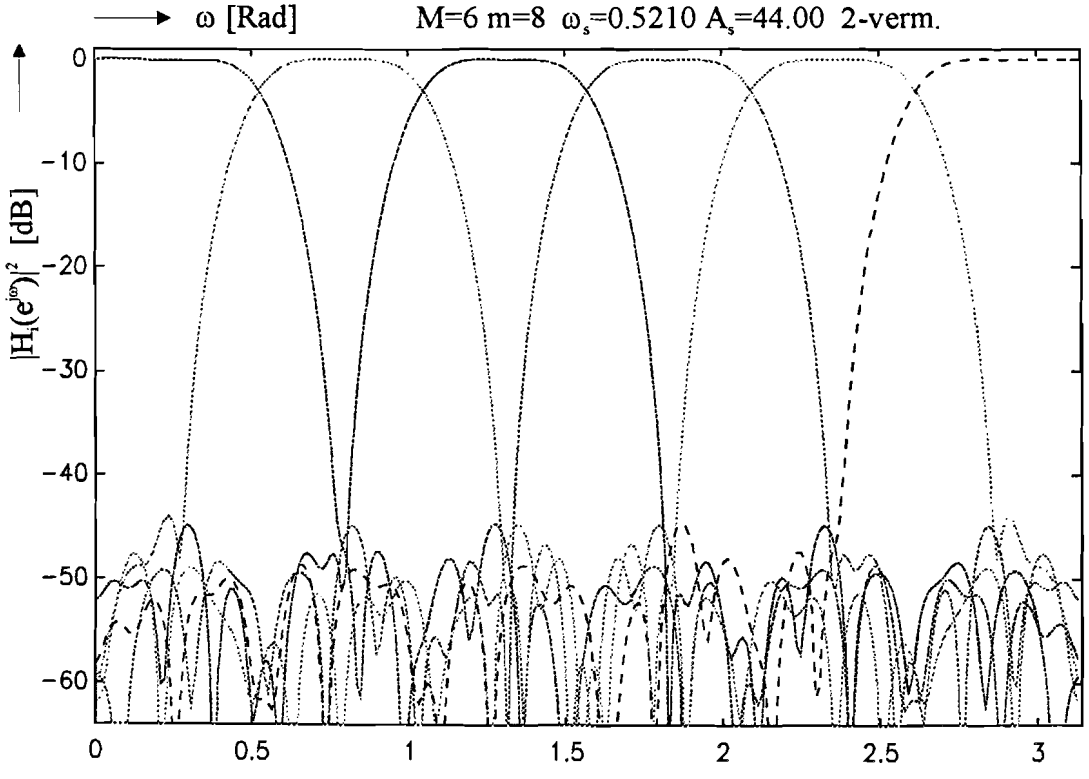


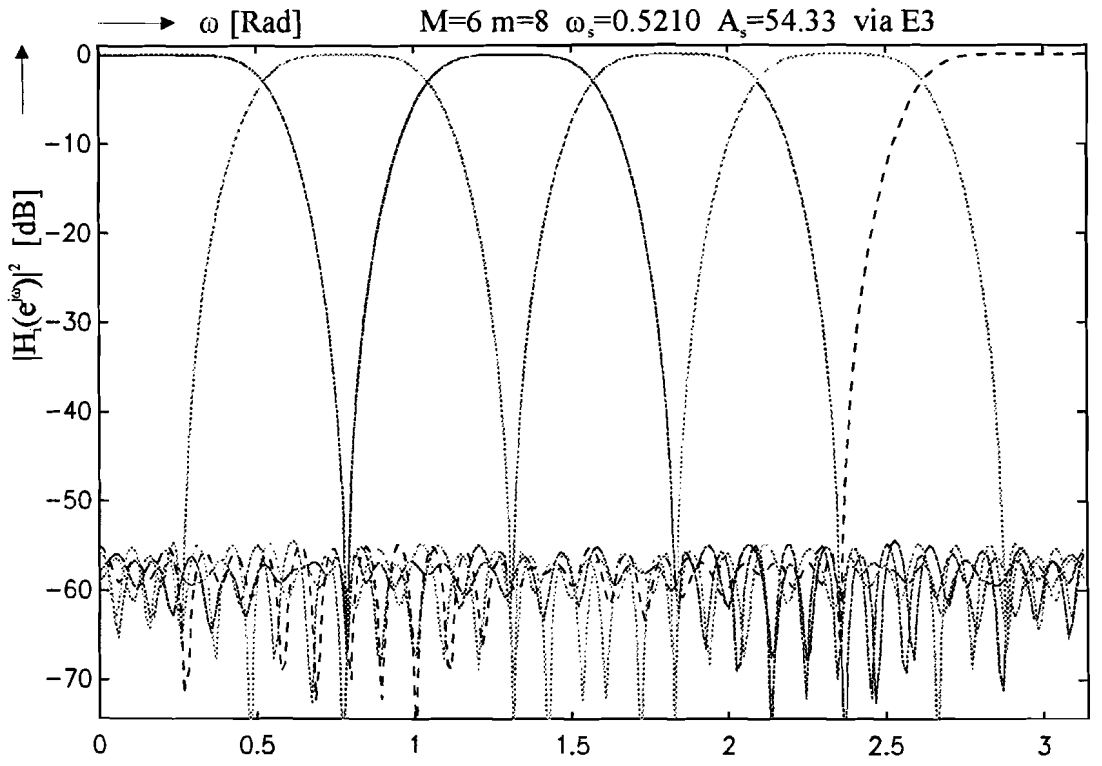
Frequentie spectra van de distorsie functie en aliasing error van een filterbank met transversale stuctuur en een REMEZ prototype voor M=6 en m=8.

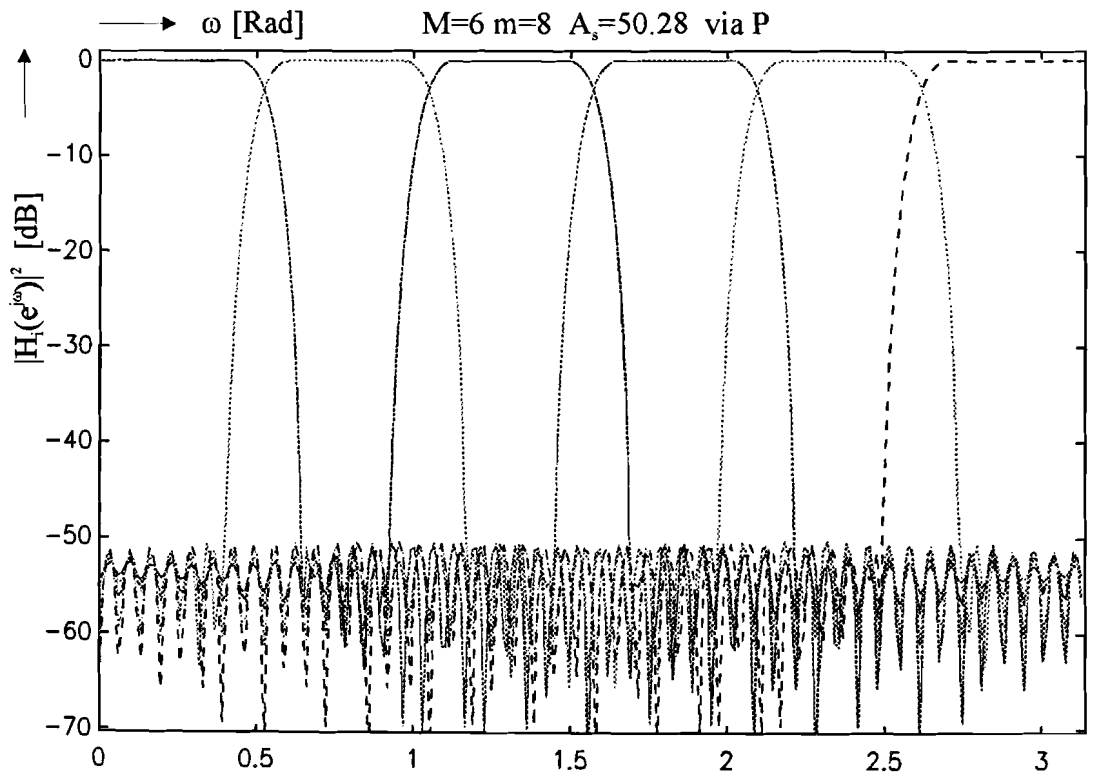


B.7 Frequentie spectra van analyse filters met M=6 en m=8 bij floating-point architectuur met N=4 en Ex=4.

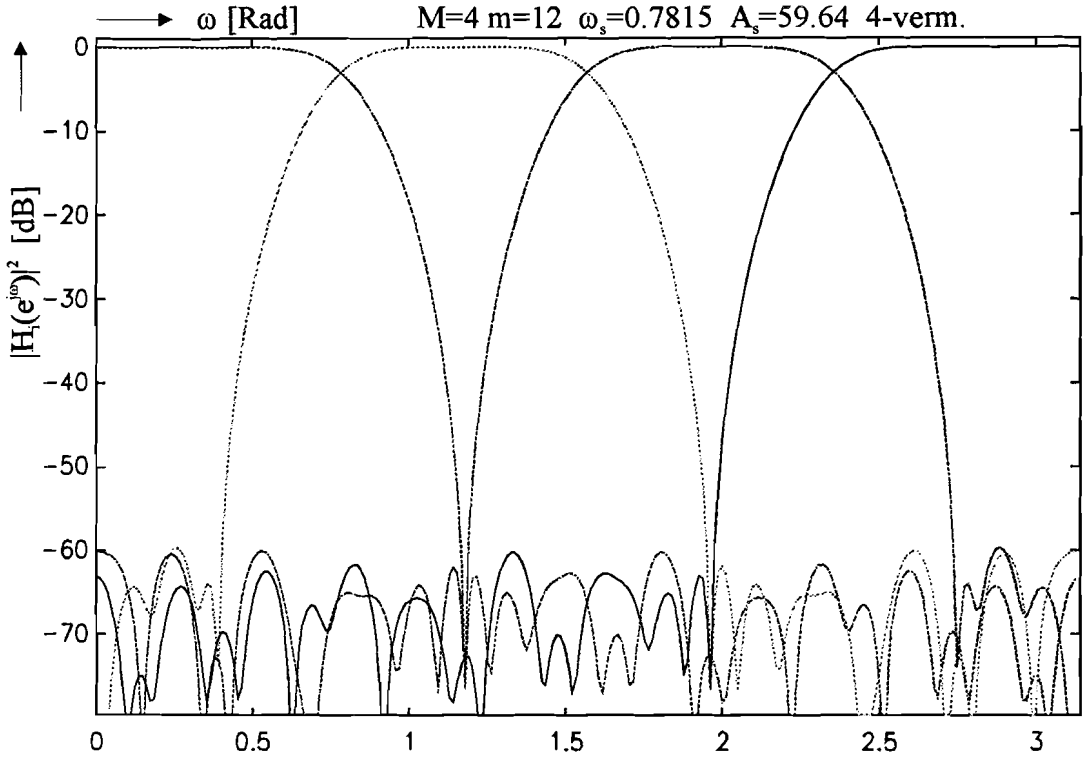


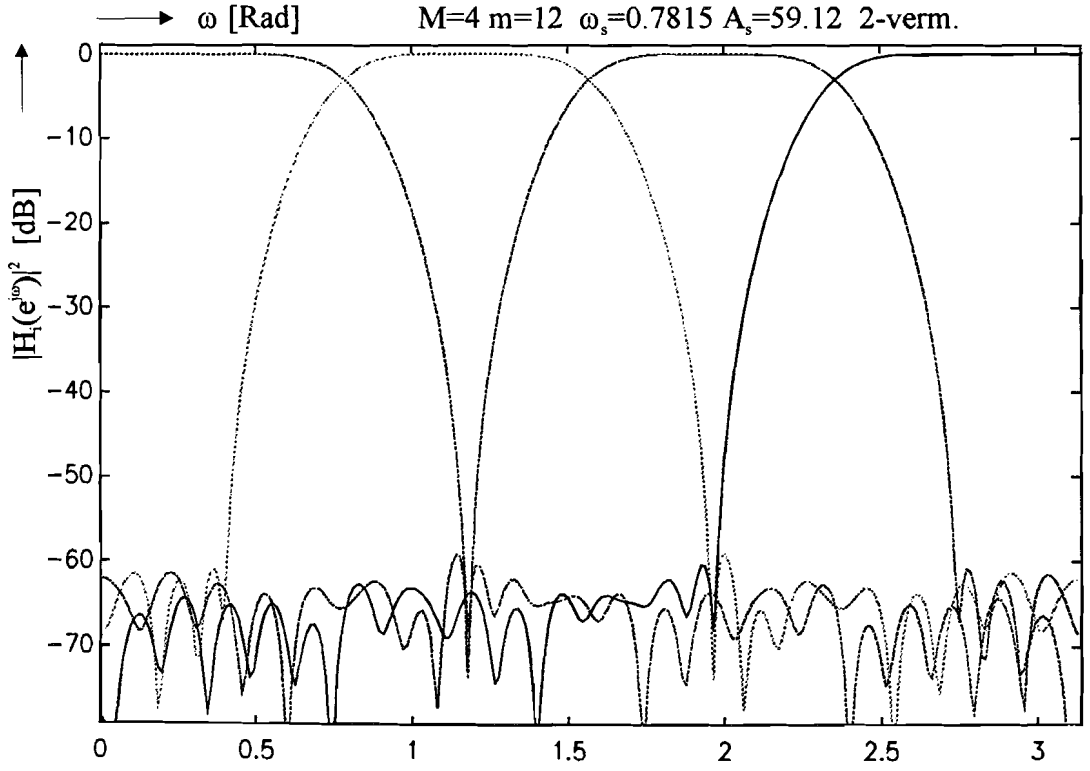


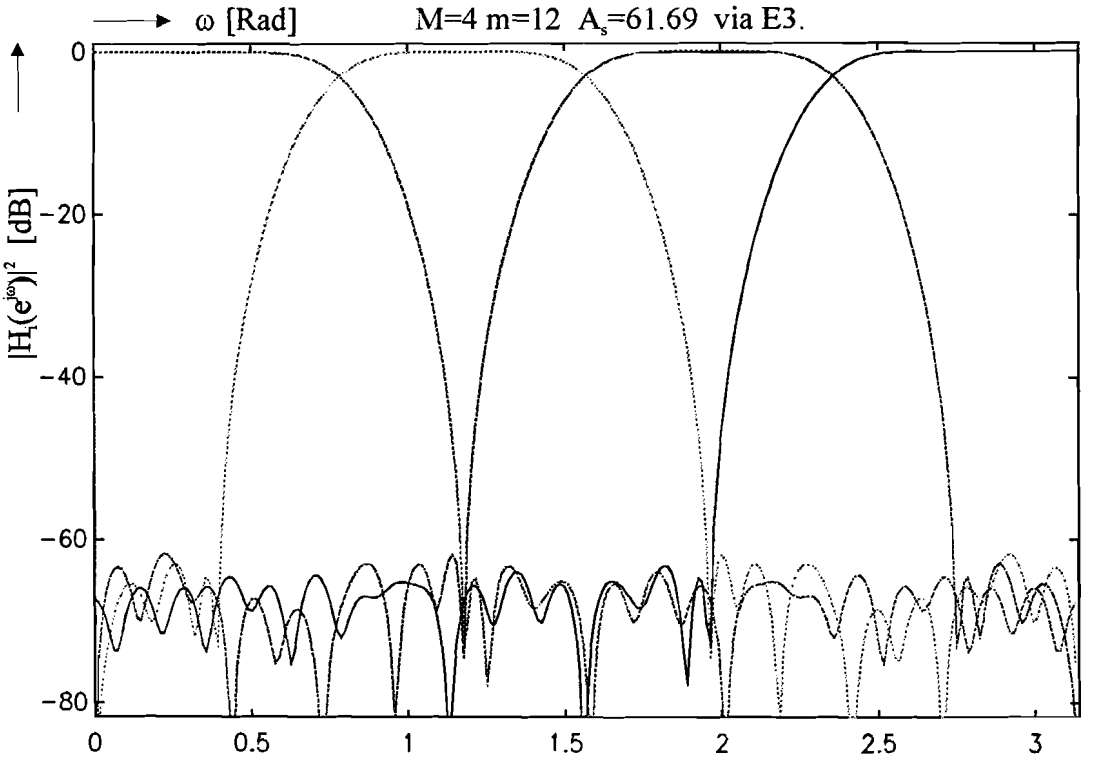

ω [Rad]
M=6 m=8 ω_s=0.5210 A_s=54.33 via E3
|H_t(e^jω)|² [dB]
0
-10
-20
-30
-40
-50
-60
-70
0
0.5
1
1.5
2
2.5
3

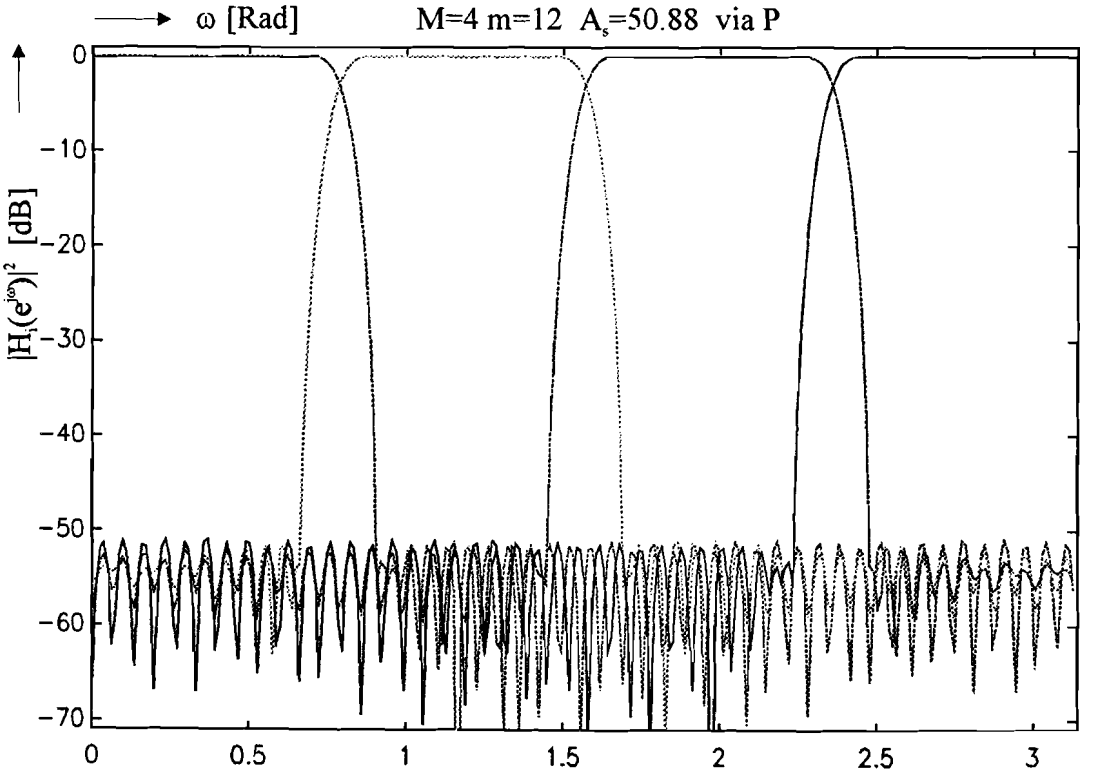

ω [Rad]
M=6 m=8 A_s=50.28 via P
|H_t(e^jω)|² [dB]
0
-10
-20
-30
-40
-50
-60
-70
0
0.5
1
1.5
2
2.5
3

B.8 Frequentie spectra van analyse filters met M=4 en m=12 bij floating-point architectuur met N=4 en Ex=4.

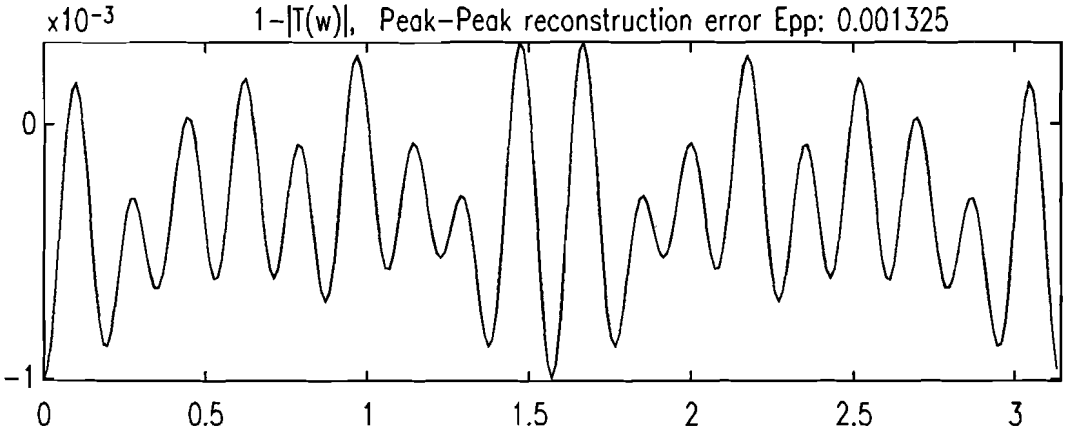


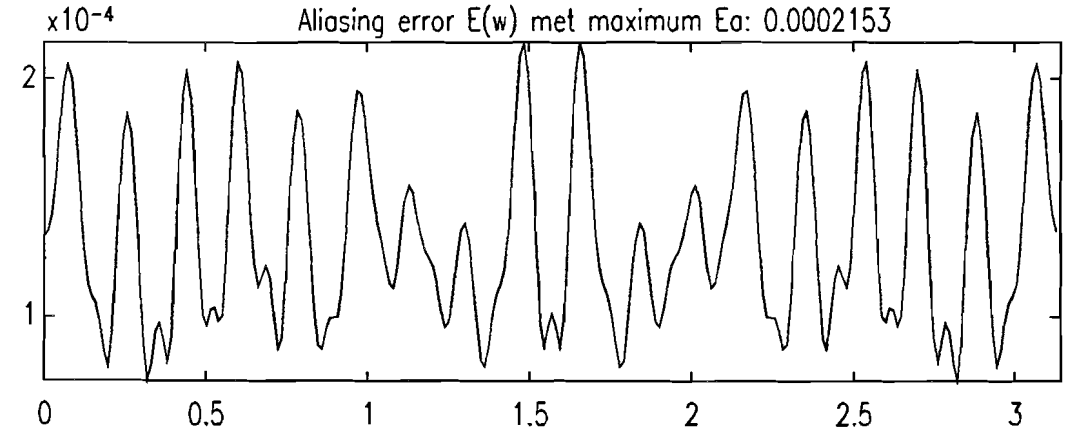



ω [Rad]
M=4 m=12 A_s=61.69 via E3.
$|H_1(e^{j\omega})|^2$ [dB]
0
-20
-40
-60
-80
0
0.5
1
1.5
2
2.5
3

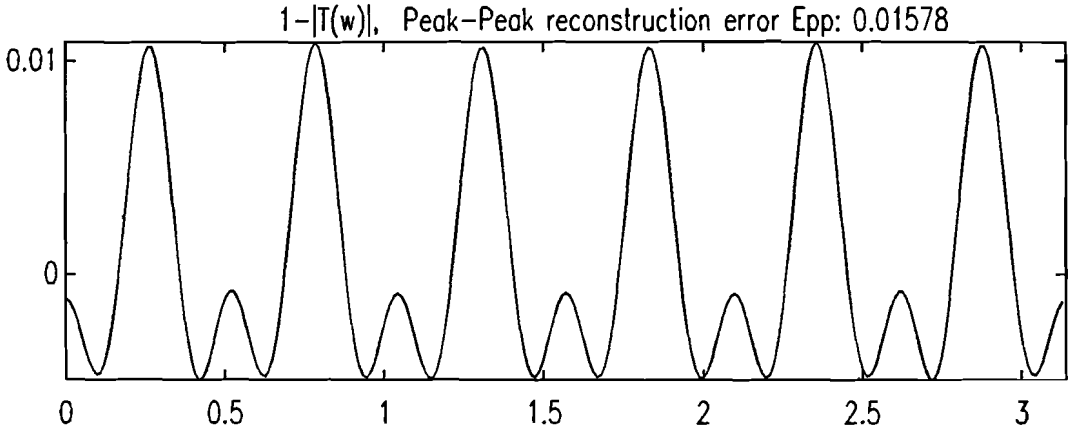

ω [Rad]
M=4 m=12 A_s=50.88 via P
$|H_1(e^{j\omega})|^2$ [dB]
0
-10
-20
-30
-40
-50
-60
-70
0
0.5
1
1.5
2
2.5
3

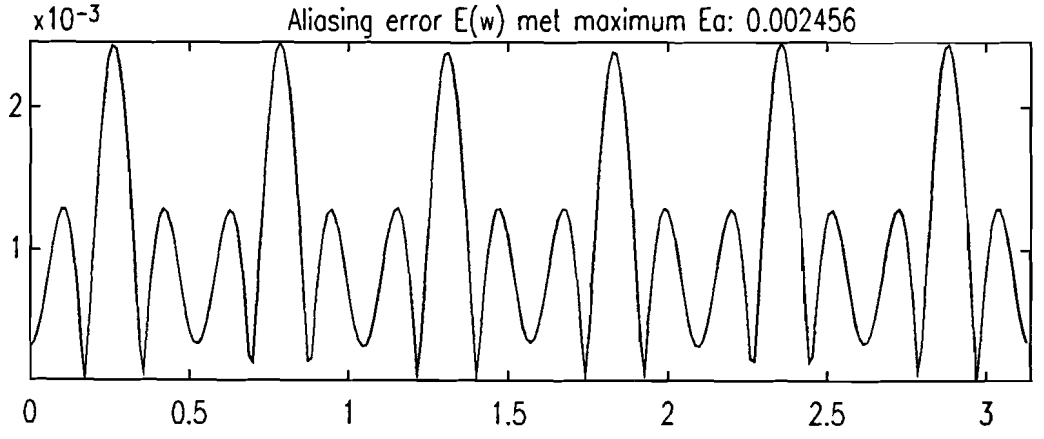
B.9 Frequentie spectra van de distorsie functie en aliasing error van een filterbank met lattice stuctuur en vier vermenigvuldigers lattices voor M=6 en m=8.



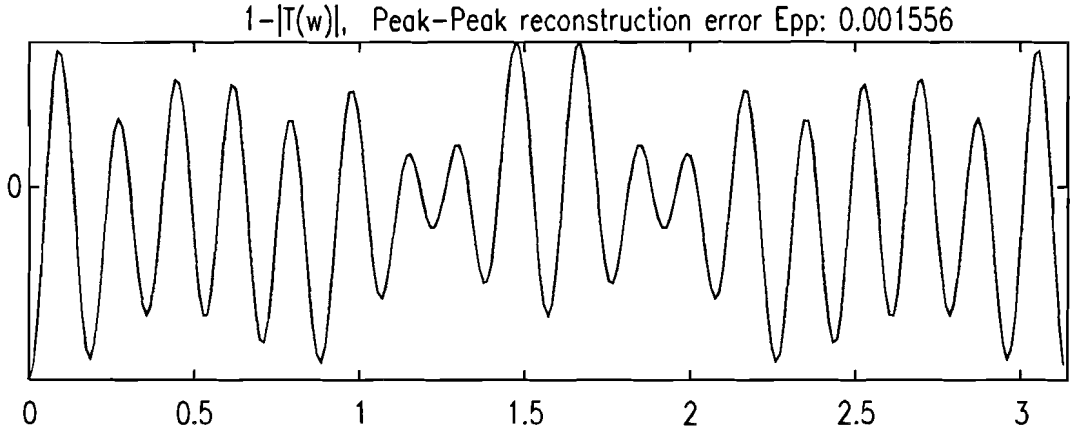


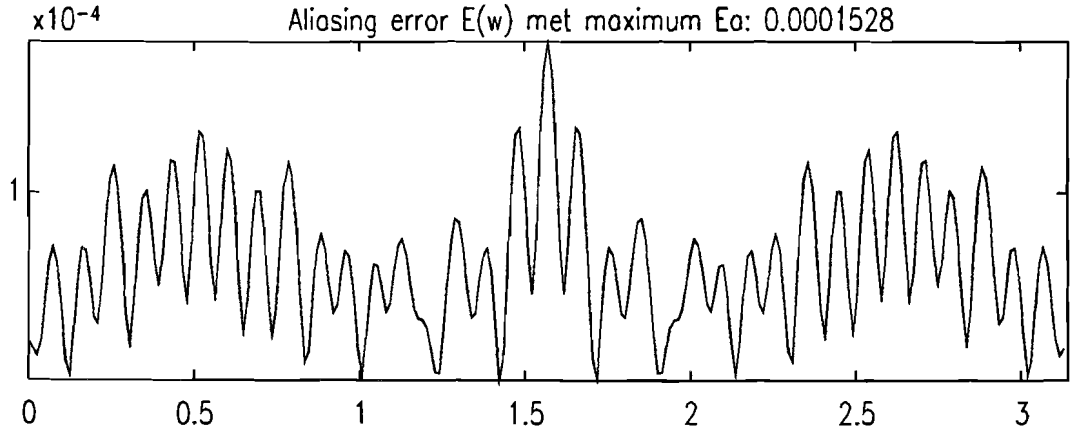
Frequentie spectra van de distorsie functie en aliasing error van een filterbank met lattice stuctuur en twee vermenigvuldigers lattices voor M=6 en m=8.



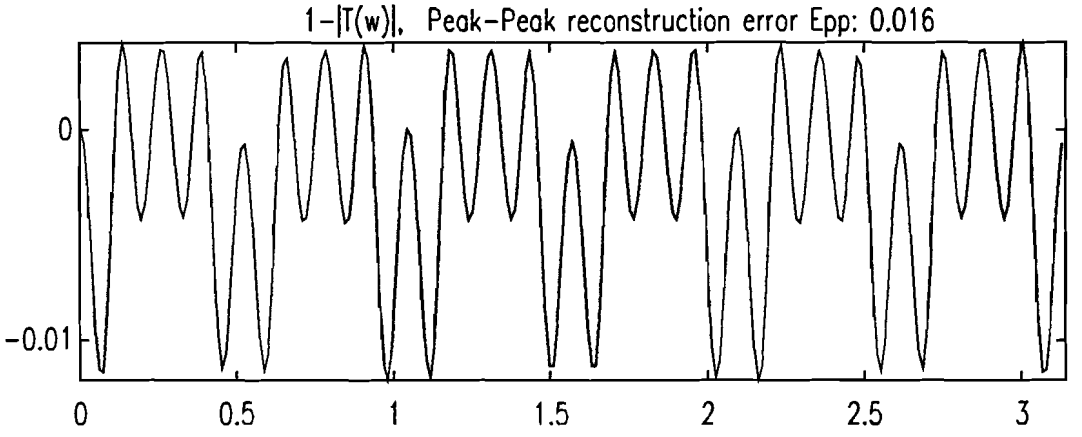


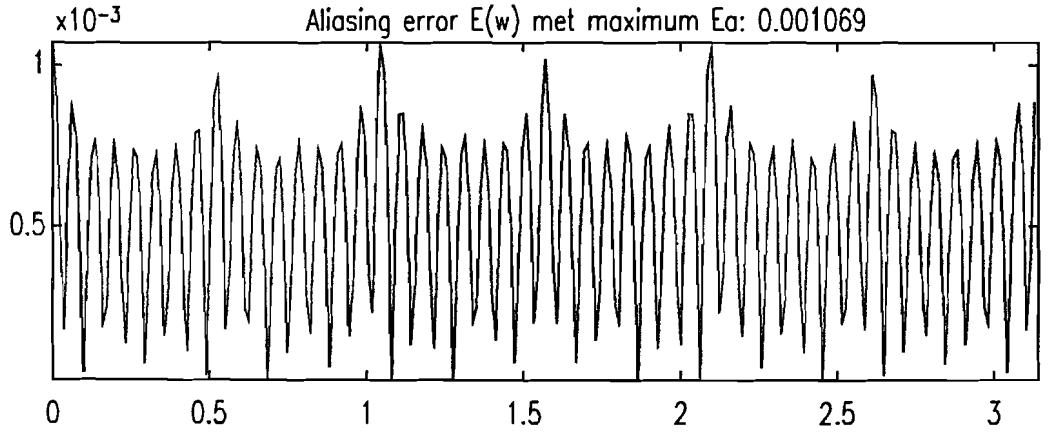
Frequentie spectra van de distorsie functie en aliasing error van een filterbank met transversale structuur en via lattice coefficienten voor M=6 en m=8.



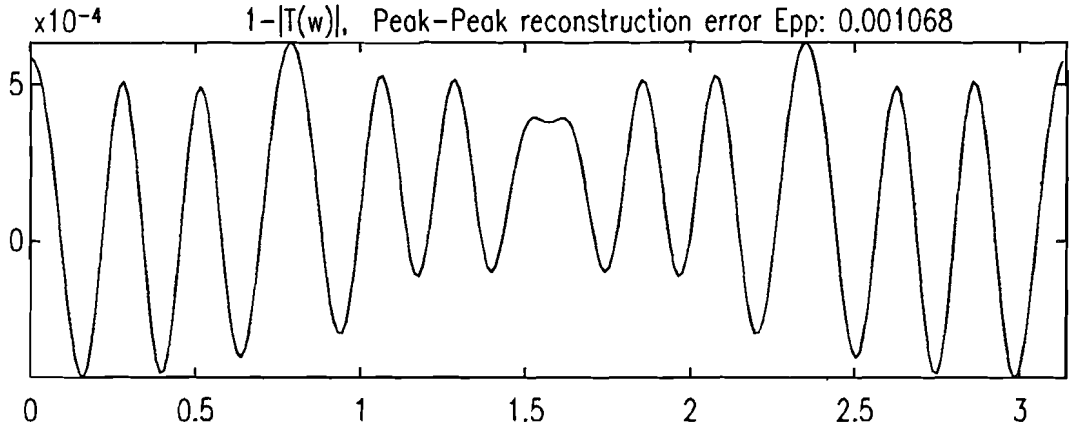


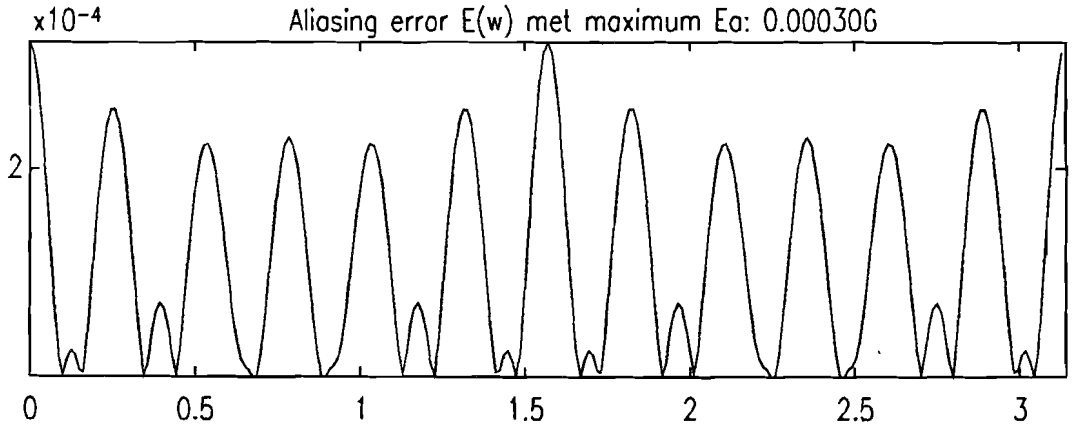
Frequentie spectra van de distorsie functie en aliasing error van een filterbank met transversale stuctuur en een REMEZ prototype voor M=6 en m=8.



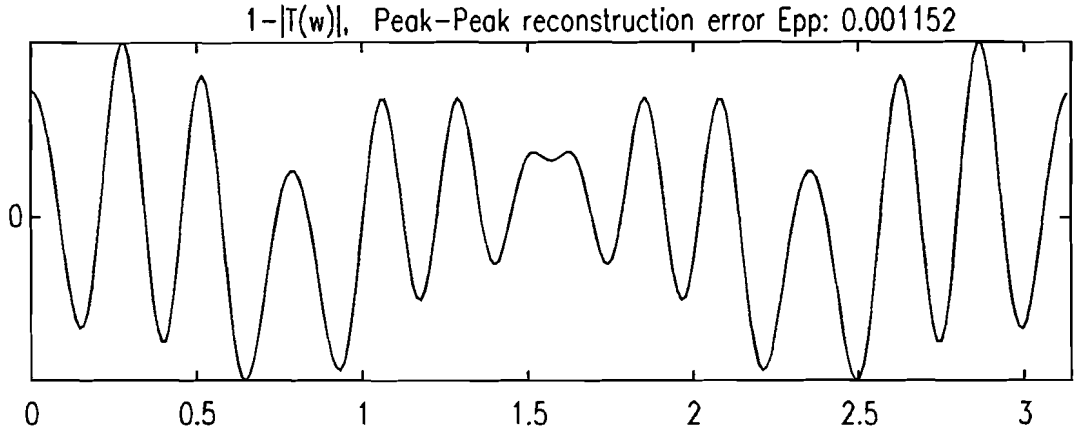


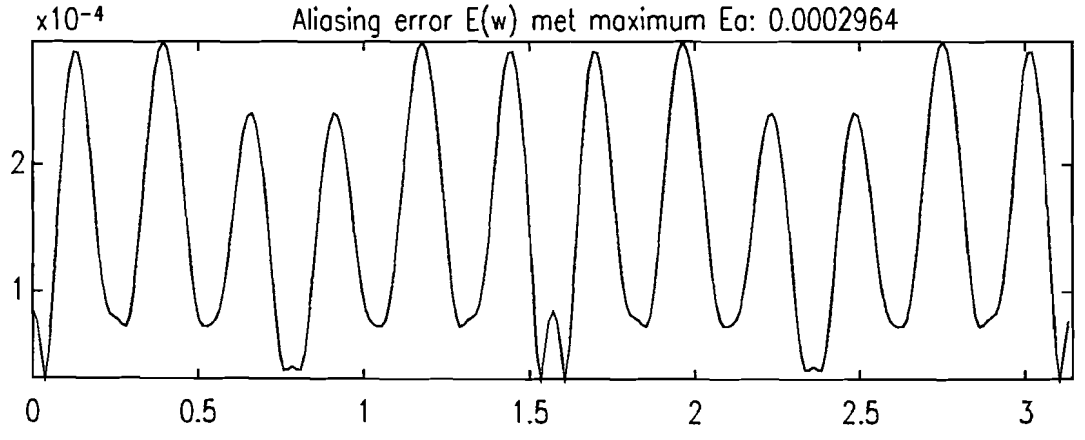
B.10 Frequentie spectra van de distorsie functie en aliasing error van een filterbank met lattice stuctuur en vier vermenigvuldigers lattices voor M=4 en m=12.



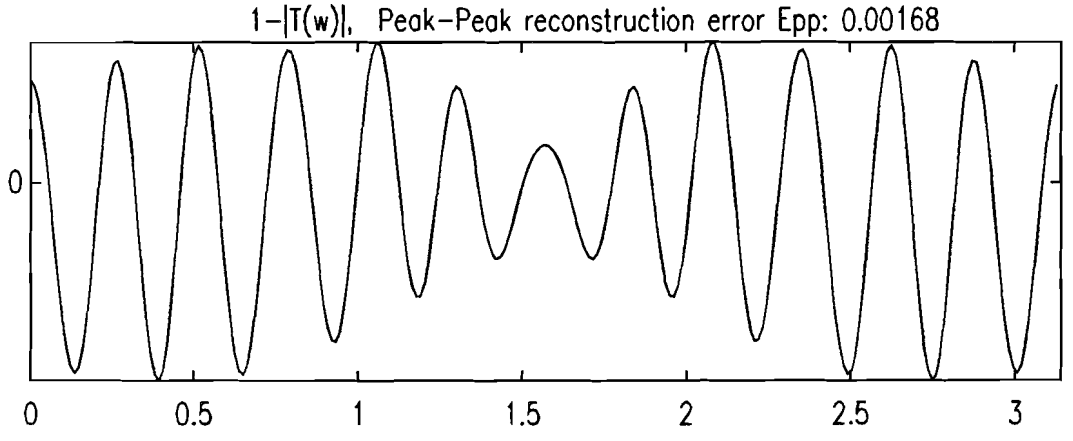


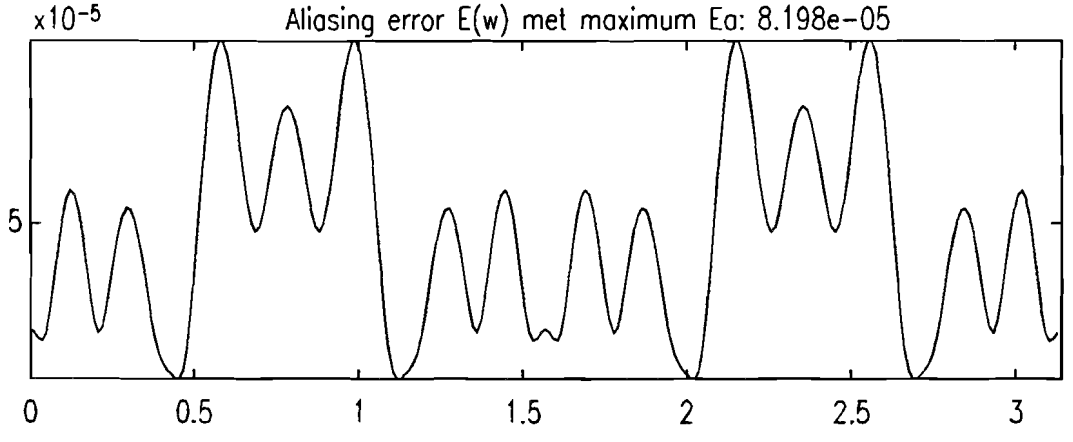
Frequentie spectra van de distorsie functie en aliasing error van een filterbank met lattice stuctuur en twee vermenigvuldigers lattices voor M=4 en m=12.



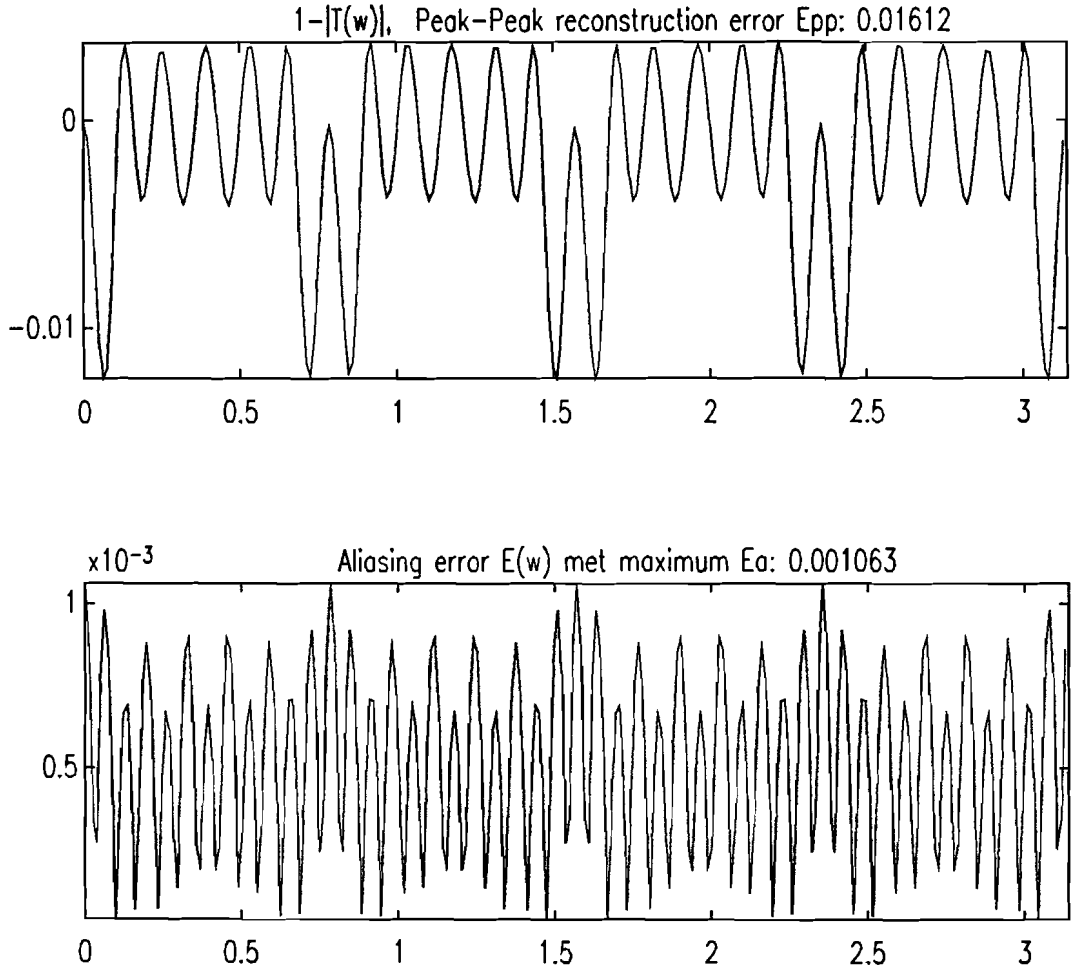


Frequentie spectra van de distorsie functie en aliasing error van een filterbank met transversale stuctuur en via lattice coefficienten voor M=4 en m=12.

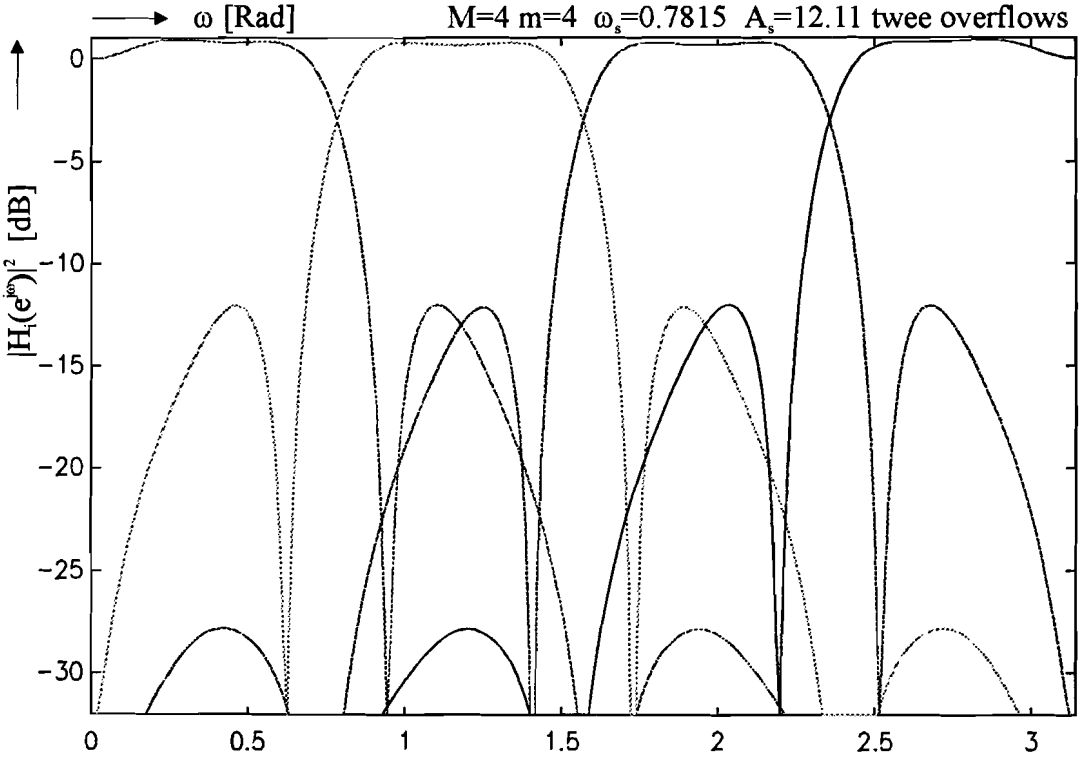


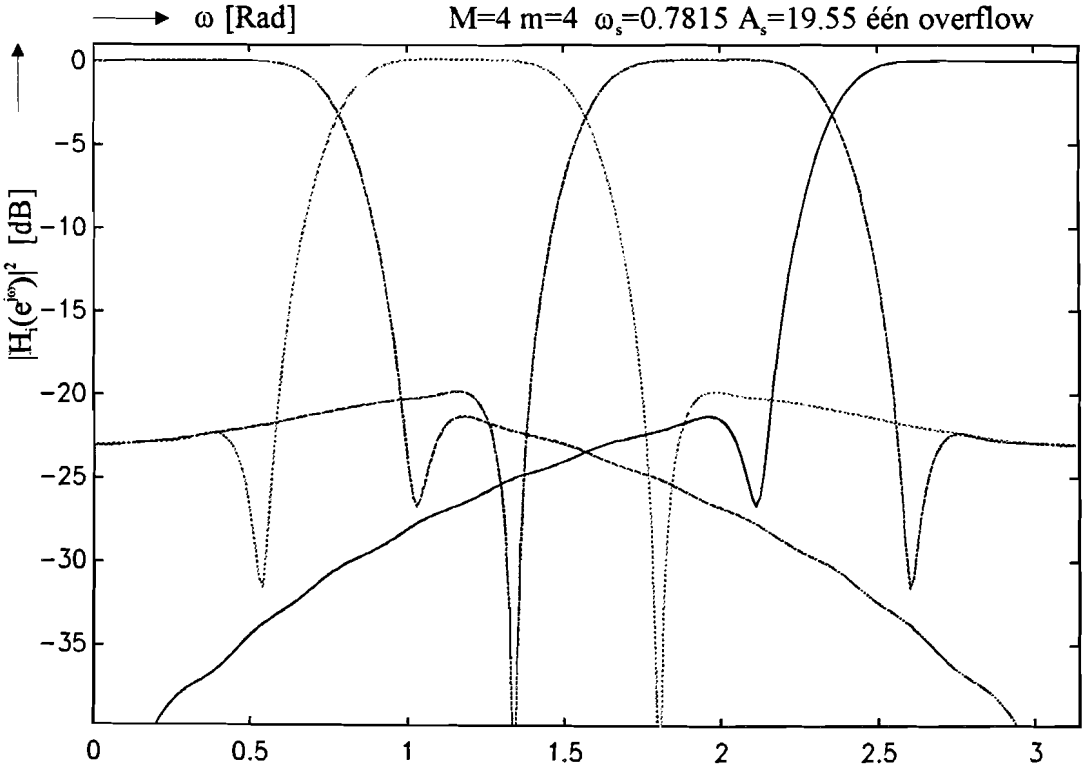


Frequentie spectra van de distorsie functie en aliasing error van een filterbank met transversale stuctuur en een REMEZ prototype voor M=4 en m=12.

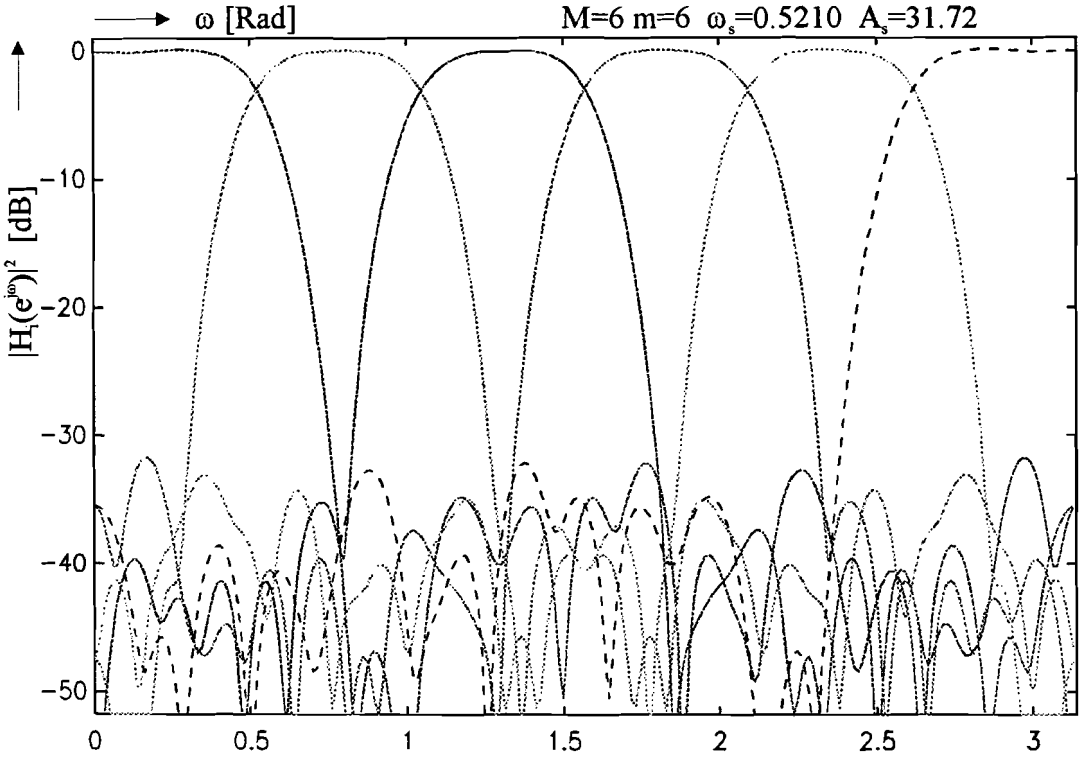


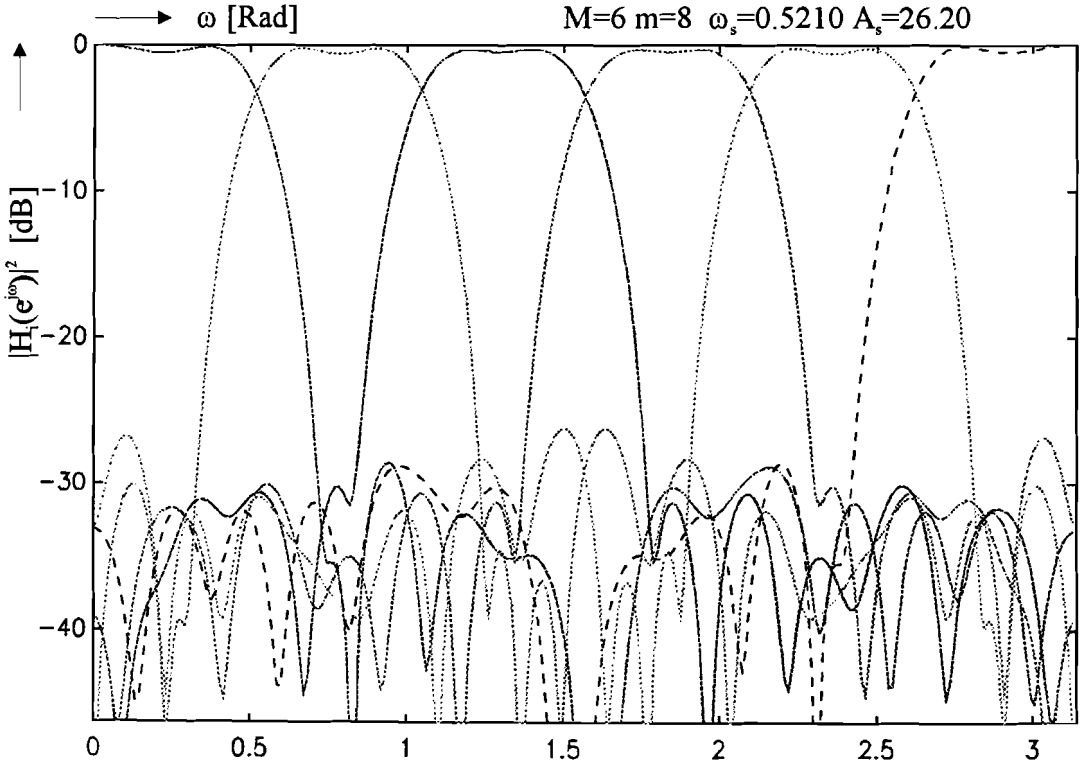
B.11 Frequentie spectra van analyse filters met M=4 en m=4 bij overflow met een fixed-point architectuur met N=16.



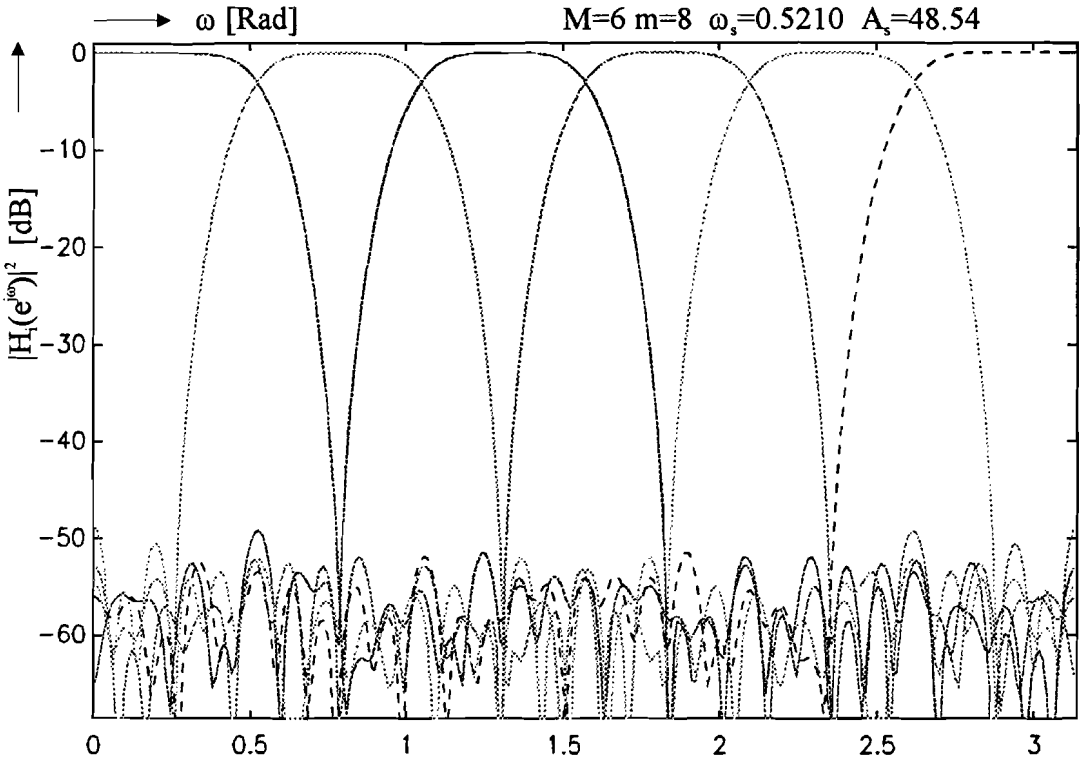


B.12 Frequentie spectra van analyse filters met M=6, m=6 en m=8 bij twee vermenigvuldigers lattices en fixed-point met N=16.





B.13 Frequentie spectra van analyse filters met M=6, m=8 bij verbeterde twee vermenigvuldigers lattices en fixed-point.



Bibliografie

[1] Vaidyanathan, P.P., 'Multirate systems and filter banks', Prentice hall signal processing series Oppenheim A.V., series editor. 1993 ISBN 0-13-605718-7

[2] Nguyen,T.Q. and Vaidyanathan, P.P. 'Maximally decimated perfect-reconstruction FIR filter banks with pairwise mirror-image analysis(and synthsis) frequency response' IEEE Trans. Acoustics Speech and Signal Processing. vol 36, No 5, pp 693-706 May 1988.

[3] Vaidyanathan, P.P., Nguyen,T.Q., Doganata, Z. and Saramaki, T., 'Improved techinque for design of perfect reconstruction FIR QMF Banks with lossless polyfase matrices.' IEEE Trans. ASSP. Vol 37 No 7 pp 1042-1056 July 1989.

[4] Koilpillai R.D. and Vaidyanathan P.P. 'New results on Cosine-Modulated FIR filter banks satisfying Perfect Reconstruction' Proc. IEEE Int. Conf. on ASSP, vol 3 pp 1793-1796, Toronto Canada May 1991.

[5] Koilpillai, R.D. and Vaidyanathan, P.P. 'Cosine-modulated FIR filter banks satisfying perfect reconstruction' IEEE Transactions on Signal Processing, vol. SP-40, pp. 770-783, April 1992.

[6] Malvar H.S. 'Modulated QMF filter banks with Perfect Reconstruction' Electronic letters vol 26 pp 906-907 June 1990.

[7] Ramstad, T.A. and Tanem, J.P. 'Cosine Modulated analysis-synthesis filter bank with critical sampling and Perfect Reconstruction' Proc. IEEE Int. Conf. on ASSP vol 3 pp 1789-1792, Toronto Canada May 1991.

[8] Nguyen, T.Q. 'A class of generalized cosine-modulated filter banks' Proc. IEEE Int. Symp. Circuits and Systems vol 2 pp 943-946, San Diego, CA May 1992.

[9] Mau J. 'Perfect Reconstruction Modulated filter Banks' Proc. IEEE Int. Conf. on ASSP vol 4 pp 273-270 March 1992.

[10] Nayebi, K. , Barnwell, T.P. III, and Smith, M.J.T. 'On the design of FIR analysis-synthesis filter bank with high computationally efficiency' IEEE Transactions on Signal Processing Vol 42 Iss 4 pp 825-834 April 1994.

[11] Nayebi, K. , Barnwell, T.P. III, and Smith, M.J.T. 'Design and Implementation of computationally efficiënt filter banks' IEEE International Symposium on Circuits and Systems pp 605-653 vol 1 1991.

[12] Enden, A.W.M van den 'Signaal Behandeling bij de DCC' uit de PATO cursus "Digitale Signaal Bewerking' deel II-10, Nov. 1993.

[13] Murthy, N.R. and Swany M.N.S. 'On the computation of running discrete cosine- and sine transforms' IEEE Transactions of Signal Processing Vol 40 Iss 6 pp 1430-1437 June 1992.

[14] Rao, K.R. and Yip, P. 'Discrete cosine transforms: Algorithmes, Advantages Applications' Academic Press San Diego CA, USA 1990 ISBN 0 12 580203 X

[15] Rao, K.R. and Yip, P.'Fast discrete transforms', in 'Handbook of digital signal processing', edited by D.F. Elliott, Academic Press, San Diego, CA, 1987

[16] Wang, Z. 'On computing the discrete Fourier and cosine transforms' IEEE Trans ASSP vol 33 pp 1341-1344 oct 1985.

[17] Saramaki, T. 'Designing prototype filters for PR cosine-modulated filter banks' IEEE International Symposium on Circuits and Systems pp 1605-1608 vol 3 1992.

[18] Nayebi, K., Barnwell, T.P. III, and Smith, M.J.T. "Time domain Filter bank analysis: A new design Theory' IEEE Trans. on Signal Processing Vol 40 no 6 pp 1412-1429 June 1992

[19] Nguyen, T.Q. 'Near perfect Reconstruction Pseudo-QMF Banks' IEEE Trans. on Signal Processing Vol 42 no 1 pp 65-76 Jan. 1994

[20] Rabiner, L.R., Gold, B., 'Theory and application of digital signal processing.' Prentice hall, Inc.,Englewood Cliffs, 1975.

[21] Vaidyanathan, P.P. 'Low-Noise and Low Sensitivity Digital Filters", in 'Handbook of digital signal processing', edited by D.F. Elliott, Academic Press, San Diego, CA, 1987

[22] Meer van A.C.P. 'Programmeerbare Digitale Signaalprocessoren' in 'Realisatie van Digitale Signaalbewerkende systemen' Ditaatnr 5750 Technische Universiteit Eindboven Jan. 22 1993.

[23] Jui Chi Yao and Chau-Yun Hsu, 'Fixed -point round-off error analysis for the discrete cosine transform' International Journal of Electronics Vol 72 Iss 1 pp 45-55 Jan. 1992.