SELF_EQUALIZED, EQUIRIPPLE MICROWAVE FILTER

Louis W. Hendrick and Paul J. Tatomir

Hughes Aircraft Co.

Summary

A computer program for designing self-equalized equiripple, even order filters is presented. The program permits its operator to attempt to meet a given set of filter response specifications by positioning either the imaginary axis zeros (i.e., the insertion loss poles) or the complex plane zeros (i.e., the equalization poles) of $S_{21}(s)$, or both. Once the zeros are optimally located, the program will first synthesize a symmetric canonical, cross-coupled array filter from the two-port, short-circuit Y-parameters. Next, the program will convert this structure, by a series of plane rotations of the coupling matrix, into an assymmetric canonical configuration. Both calculated and measured response data are presented.

Introduction

Although externally group delay equalized filters are still used in many applications (1), self-equalized filters offer certain advantages. First, in communication satellite applications, for example, each filter must perform to some minimum standard over a range of temperatures. Typically, a filter that is used to demultiplex one channel from the entire communication band must be delay equalized to meet its specifications. If it is externally equalized, it will be attached to its equalizer by way of a circulator. Special care must be taken to ensure that the filter and equalizer are at or near the same temperature, or the center frequencies of the two will differ enough to ruin the performance of the combination. This may mean that metal straps must be used to thermally couple the filter and equalizer. On the other hand, a self-equalized filter can be made from a single metal housing and will require less effort to ensure that it will work properly over a temperature range. Due to the fact that the self-equalized design does not need an extra circulator and thermal coupling straps, it will be lighter than the equivalent externally equalized filter, associated devices, and hardware. The reduction in weight, an important aspect for satellite applications, due to the large number of filters used in a satellite, can be considerable.

Other advantages to using self-equalized filters are increased reliability due to fewer component parts, reduced volume, and ease of tuning a single filter versus tuning two separate devices.

A computer program that can be used to design self-equalized, equiripple, even order filters is shown in Fig. 1. This program is written in Pascal for the Hewlett-Packard 9836 desk top computer. To conserve space, many of the operations are simply described as algorithms and are not fully written out; these are enclosed in rows of asterics. A description of this program and both calculated and measured response data follow.

Program Description

Certain points in the program are marked with numbers enclosed in braces, e.g. 1 . These location marks will be used to aid identifying the various parts of the program.

Between 1 and 5 , the data type "complex" and the complex constants, czero = (0.0,0.0), cone = (1.0,0.0), and cj = (0.0,1.0), are defined, the variables used in the program are declared and dimensioned, the standard complex arithmetic operations are defined, center frequency, FO, and unloaded Q of the filter are entered. These items are assumed to be fixed with respect to the optimization process. Next whatever filter specifications that are to be used are entered.

Between 5 and 18 , there is a sequence of operations that will determine the S-parameters of the filter in the complex domain. The bandwidth of the filter (BW) the return loss (RL) level the number of electrical resonators (N), that is, the order of the filter. The number of transmission zeros or finite frequency loss poles N_{te}, and finally the number of equalization poles non-j omega axis zeros of S_{21} (N_{ze}) are entered. N, N_{zt}, and N_{ze} are all even integers.

The imaginary axis zeros of S_{21} are entered between 7 and 8. These are the same as the finite frequency loss poles and are assumed to be normalized to the bandedge of the filter, i.e., an entry of 2 is two half-bandwidths from the center of the filter. Since the filter is assumed to be symmetric about its center frequency, only $N_{zt}/2$ of these zeros are entered. A strategy for initial placement of the zeros can be found in (2).

Between 8 and 9, the zeros used to shape the delay response are entered. First the real and then the imaginary part are entered. The imaginary part is normally less than 1, i.e., the zero is within the passband of the filter, while the passband of the filter, while the real part is between 0 and 1. These zeros must either occur as pairs on the sigma axis, or as quads symmetric about both the sigma and omega axes.

Between 10 and 11, the transformed variable technique is used to determine the S-parameters of the equiripple filter with the given zeros of S_{21} and the assumed bandwidth and return loss. The method used is the same as in (2) except that it has been extended to the case with zeros not on the imaginary axis.

Between 17 and 18, the filter is analyzed and compared to its requirements. At 18, the designer enters whether he is satisfied with the response. By iteratively moving the zeros, a near optimum filter can be designed.

Between 18 and 24, the filter is synthesized in the form of a symmetric canonical, cross-couplings are converted by plane rotations into the assymmetric canonical network. This structure was introduced in (3), but only the cases N = 6 and N = 8 cases were detailed. Between 24 and 25, the algorithm for the general even order case, where N is greater than 6, is given.

0149-645X/85/0000-0469 $01.00 © 1985 IEEE

Results

Many filters have been successfully designed using this program. An experienced operator can design a 10th order self-equalized filter in less than one hour on a desk top HP9836 computer. The slowest aspect of the program is the filter analysis section. The optimization technique is straight forward; the zeros of S_{21} affect the response in direct fashion that can be learned by experimenting with the program. On the other hand, methods that optimized the couplings directly require careful selection and re-selection of weighing parameters to ensure that the program is converging to a reasonable solution.

Fig. 2 shows the response of one such design. This is the calculate response of a $N = 10$, $N_{ze} = 4$, $N_{zt} = 4$ filter. This design was built to operate at 12 GHz using circular waveguide resonators operating in the TE_{113} mode. It was constructed using the assymmetric canonical couplings. The measured response is shown in Fig. 3. Both the calculated and measured delay responses clearly show the effect of the equalization zeros of S21.

Conclusion

A computer program for designing self-equalized filters was presented. The program incorporated an extension of the transformed variable technique to non-imaginary axis zeros, a synthesis technique for cross-coupled arrays from the polynomial description of the network admittance parameters, and plane-rotation conversion of any even order, greater than or equal to 6, symmetric to assymmetric canonical couplings. The program can be run on a desk top computer like an HP9836.

References

(1) S. J. Fiedfiusfko, R. C. Chapman, "Miniature Filters and Equalizers Utilizing Dual Mode Dielectric Resonator Loaded Cavities", 1982 IEEE MTT-S International Microwave Symposium Digest.
(2) R. W. Daniels, Approximation Methods in Electronic Filter Design, 1974, McGraw-Hill Book Company.
(3) S. J. Fiedfiusfko, "Assymmetric Canonical Dielectric Resonator Filters", 1983 IEEE MTT-S International Symposium Digest.

```
program filsyn(input,output);
{1};
{**************************
DEFINE
 type:  complex
 complex constants:
    czero,cone,cj
**************************};
{2};{**************************
DECLARE VARIABLES AND ARRAYS
**************************};
{3};{**************************
 DEFINE COMPLEX FUNCTIONS:
  cadd,cdvd,cmag,carg,cpower,
  csqrt
 **************************};
{4};
begin
 writeln
   (' enter center freq F0');
 readln(F0);
 writeln
     (' enter Q of filter');
 readln(Q);
 {****enter filter specs***};
 {repeat}
  {5};
  writeln
    (' enter bandwidth BW');
  readln(BW);
  writeln
   (' enter return loss RL');
  readln(RL);
  EPSILON:=
      1.0/sqrt(exp((RL/10.0)*
              ln(10.0))-1.0);
  {6};
  writeln
         (' enter number N',
          ' of resonators');
  readln(N);
  writeln
  (' enter number Nzt',
   ' of transmission zeros');
  readln(Nzt);
  writeln
   (' enter number Nze',
    'of equalization zeros');
  readln(Nze);
  Nzf:=Nzt+Nze;
  Nzi:=N-Nzf;
  {7};
  i:= 1;
  while (i<=Nzt div 2) do
  begin
   SIGMAz[i]:= 0.0;
   writeln


    ('enter imaginary axis
          'zero OMEGAz[i]')
   readln(OMEGAz[i]);
   i:= i+1;
  end;
{8};
while (i<=Nzf div 2) do
begin
 writeln
  (' enter real axiz zero'
   ' SIGMAz[i]');
 writeln
 (' enter imag - axiz zero',
  ' OMEGAz[i]');
 readln(SIGMAz[i]);
 readln(OMEGAz[i]);
 if (SIGMA[i]<>0.0)
  then
  begin
   SIGMAz[i+1]:=
                  -SIGMAz[i];
   OMEGAz[i+1]:=
                   OMEGAz[i];
   i:= i+2;
  end;
  else
  i:=i+1;
end;
{9};
for i:= 1 to (Nzf div 2) do
begin
 Sz[i].re:= SIGMAz[i];
 Sz[i].im:= OMEGAz[i];
 Zz[i]:=
  csqrt
  (cadd(cone,cdvd
  (cone,cpower(2,Sz[i]))));
end;
{10};{**********************
 FORM POLYNOMIALS IN z:
 A11h(z):=EPSILON*
      product of terms
      (Ev([(z+1)^Ni] *
       [(z+Zz[i])^2]))
   for i:=1,2,...,Nzf div 2
 A21h(z):=product of terms
  ([(z^2-1.0)^(Nzi div 2)]*
  [z^2-Zz[i]^2])
  for i:=1,2,...,Nzf div 2
 **********************};
{11};{**********************
 FACTOR POLYNOMIAL A11h(z)

    INTO TERMS
  (z^2 - Z11[i]^2)
  for i:=1,2,...,N div 2
 **********************};
{12};{**********************
 FACTOR POLYNOMIAL A21h(z)
    INTO TERMS
  (z^2 - Z21[i]^2)
  for i:=1,2,...,N div 2
 **********************};
{13};{**********************
 FACTOR POLYNOMIAL
    A11h(z)^2+A21h(z)^2
     INTO TERMS
  (z^4 + Pz[i]*z^2 +Qz[i])
  for i:=1,2,...N/2
 **********************};
{14};
for i:= 1 to N div 2 do
begin
 A[i]:=
 1.0/sqrt(1.0+Pz[i]+Qz[i]);
 B[i]:= (1.0+0.5*Pz[i])/
        (1.0+Pz[i]+Qz[i]);
 SIGMAp[i]:=
     sqrt((A[i]-B[i])/2.0);
 OMEGAp[i]:=
      sqrt((A[i]+B[i])/2.0)
end;
{15};{**********************
FORM POLYNOMIALS IN s:
 A11(s):= product of terms
   [s^2-1.0/(Z11[i]^2-1.0)]
         for i=1,2,...,N/2
 A21(s):= [product of terms
       (1.0/(1-Z21[i]^2))]*
          [product of terms
             s^2-
   (SIGMAz[i]+j*OMEGA[i])^2]
 B(s):= product of terms
       [s^2+2.0*SIGMAp[i]*s
  +SIGMAp[i]^2+OMEGAp[i]^2]
 **********************};
 {16};{**********************
 FORM RATIOS OF POLYNOMIALS
        IN s
 S11(s):= A11(s)/B(s);
 S21(S):= j*A21(s)/B(s)
 **********************};
 {17};{**********************

  ANALYZE FILTER RESPONSE
  AT FREQENCY F, S11(s)
  AND S21(s) ARE CACULATED
  AT s:=
   (F0/BW*Q)+j(F-F0)/(BW/2.)
  **********************};
{until
  (satisfied with response)};
{18};{**********************
FORM POLYNOMIALS IN s
 P11(s):=
       .5* Odd(B(s)+A11(s));
 P11p(s):= P11(s));
 P21(s):= .5*A21(s);
 P21p(s):= P21(s));
 Q(s):= .5 Ev(B(s)+A11(s));
 Qp(s):= Q(s)
**********************};
{19};{**********************
for k:=
      (N div 2) downto 1 do
begin
 Kcoup[k]:=
    sqrt
    (leading coefficient of
                P11p(s));
 Kcoupb[k]:=
        leading coef. of
          P21p(s)/
       (j*Kcoup[k]*Kcoup[k])
 FORM POLYNOMIALS IN s
  Q(s):=
    (P11(s)^2-P21(s)^2)/
     ((Kcoup[k]**4)*Q(s));
  Qp(s):= Q(s);
  P11(s):= P11(s)/
       Kcoupb[k]**2 -Q(s)*s;
  P11p(s):= P11(s));
  P21(s):=
     -P21(s)/Kcoup[k]**2 +
      cj*Kcoupb[k]*Q(s);
  P21p(s):= P21(s));
end;
**********************};
{20};
for i:= 1 to N+2 do
begin
 for j:= 1 to N+2 do
 begin
  Jcoup[i,j]:= 0.0
 end;
end;
{21};
for i:= 1 to (N div 2) do
begin
 j:= i+1;
 Jcoup[i,j]:=
      Kcoup[(N div 2)-(i-1)];
 Jcoup[j,i]:= Jcoup[i,j];
 Jcoup[N+3-i,N+3-j]:=
                 Jcoup[i,j];
 Jcoup[N+3-j,N+3-i]:=
                 Jcoup[i,j];
 write
 (i,j,N+3-i+N+3-j,
  Jcoup[i,j]);
end;
{22};
for i:= 2 to (N div 2)+1 do
begin
 j:= N+1-i;
 Jcoup[i,j]:=
     Kcoupb[N div 2 - (i-2)];
 Jcoup[j,i]:= Jcoup[i,j]
end;
{23};
for i:= 1 to N+2 do
begin
 for j:= 1 to N+2 do
 begin
  Jcoupp[i,j]:= Jcoup[i,j];
 end;
end;
```

Fig. 1 Filter Synthesis Program

```
{24};
for i:= 4 to (N div 2)+1 do
begin
 j:= N+4-i;
 THETA:= arctan(Jcoup[i-1,j]/
  Jcoup[i-1,i]);
 C:= cos(THETA);
 S:= sin(THETA);

 for k:= 1 to N+2 do
 begin
  Jcoupp[i,k]:=

   C*Jcoup[i,k]+S*Jcoup[j,k];
  Jcoupp[k,i]:= Jcoupp[i,k];
  Jcoupp[j,k]:=
      -S*Jcoup[i,k]+
      C*Jcoup[j,k];
  Jcoupp[k,j]:= Jcoupp[j,k];
 end;
end;

{25};
for i:= 1 to N+1 do
begin
 j:= i+1;
 write(i,j,Jcoupp[i,j]);
end;

{26};
for i:= 2 to (N div 2)-1 do
begin
 j:= N-i;
 write(i,j,Jcoupp[i,j]);
end;
{27};
i:= 2;
j:= N+1;
write(i,j,Jcoupp[i,j]);
{28};
end.
```

Fig. 1 continued

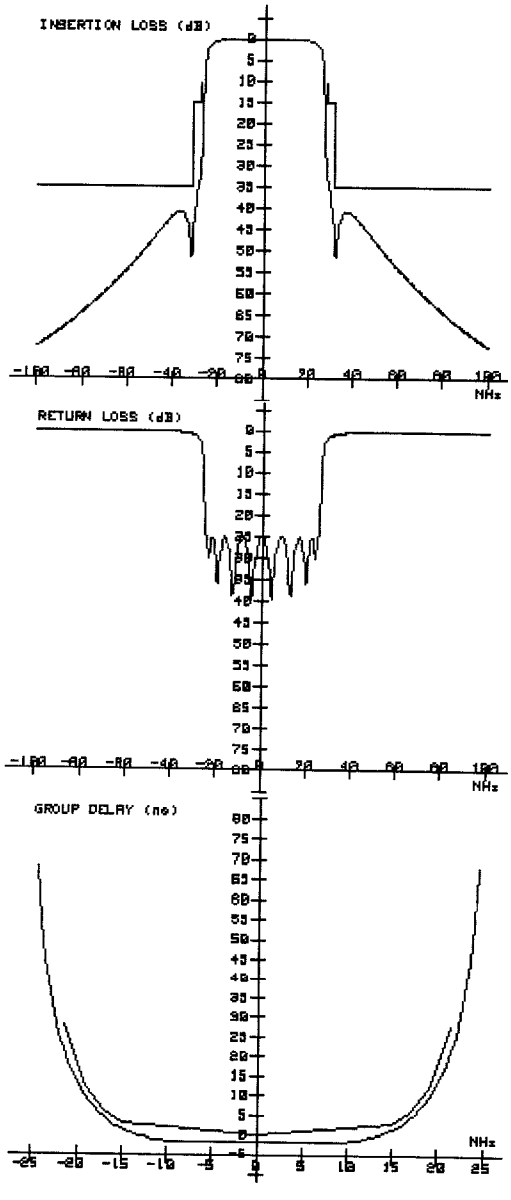

Fig. 2 Calculated Filter Response

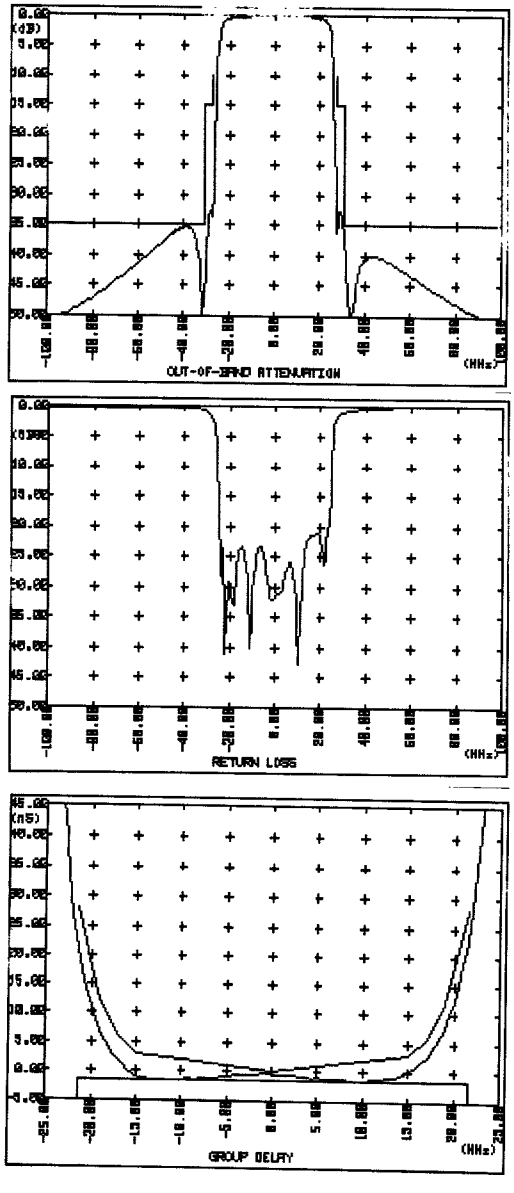

Fig. 3 Measured Filter Response