There are four MATLAB m files listed below. These will tell you how to compute the parameters you need (two sets of reflection coefficients), produce a file (.asm) containing these parameters, and to implement a SOAP (sum of all pass) filter structure. To test the algorithm, use these MATLAB command lines:

```
EDU>filterstring='butter(13,3.5*2/48)';
EDU>x=[1,zeros(1,100)]; %unit pulse
EDU>y=soap(filterstring,x); %pulse response of SOAP
EDU>[b,a]=eval(filterstring);
EDU>yy=filter(b,a,x); %pulse response the usual way
EDU>max(abs(y-yy)), % compute the maximum error

ans =

    3.568949372145802e-10
```

MATLAB m-files

```
function y=soap(filterstring,x)
%
% y=soap(filterstring,x)
%
% x is a row vector input
% y is a row vector output
% computed using a Sum Of AllPass filter
%
% It is Assumed that the filter has ODD ORDER.
%
% 'filterstring' is consistent with MATLAB filter
%    design tools. Examples are:
%     'butter(11,2*3/48)'
%     'cheby1(13,.5,2*5/48)'
%
[z,p,b0]=eval(filterstring);
a1=1;
a2=1;
[y,k]=sort(imag(p));
n=length(k);
while n > 0
          a1=conv(a1,[1, -p(k(n))]);
          n = n-1;
          if n > 0
          a2=conv(a2,[1, -p(k(n))]);
          end
          n = n-1;
end
c1=reflec(real(a1));d1=sqrt(1-c1.*c1);
c2=reflec(real(a2));d2=sqrt(1-c2.*c2);
y=.5*(allpass(c1,d1,x)+allpass(c2,d2,x));
```

```
function soaptable(filterstring,filename)
%
%  soaptable(filterstring,filename)
%
%  Produce a table of coefficients for a sum of allpass filter
% (using a normalized Markel-Gray lattice structure).
% The filter must be an ODD ORDER
%  Butterworth, Chebychev, or elliptic filter
%
% 'filterstring' is consistent with MATLAB filter
%    design tools. Examples are:
%     'butter(11,2*3/48)'
%     'cheby1(11,.5,2*3/48)'
%
% 'filename' is the name of the file to be created
%
fn=[filename,'.asm'];
fid=fopen(fn,'wt')
[z,p,b0]=eval(filterstring);
a1=1;
a2=1;
[y,k]=sort(imag(p));
n=length(k);
while n > 0
          a1=conv(a1,[1, -p(k(n))]);
          n = n-1;
          if n > 0
          a2=conv(a2,[1, -p(k(n))]);
          end
          n = n-1;
end
c1=reflec(real(a1));d1=sqrt(1-c1.*c1);n1=length(c1);
c2=reflec(real(a2));d2=sqrt(1-c2.*c2);n2=length(c2);
fprintf(fid,'; The file %s contains\n',fn);
fprintf(fid,'; SOAP coefficients for the IIR filter\n');
fprintf(fid,'; H(z) = %s\n',filterstring);
fprintf(fid,'; \n',fn);
fprintf(fid,' dc %d ; size of the first allpass filter\n',n1);
fprintf(fid,'; reflection coefficients c(k),d(k)\n');
for k=[n1:-1:1]
   fprintf(fid,' dc %1.12g,%1.12g\n',c1(k),d1(k));
end
fprintf(fid,'; \n',fn);
fprintf(fid,' dc %d ; size of the second allpass filter\n',n2);
fprintf(fid,'; reflection coefficients c(k),d(k)\n');
for k=[n2:-1:1]
```

```
  fprintf(fid,' dc %1.12g,%1.12g\n',c2(k),d2(k));
end
fclose(fid)


function y=allpass(c,d,x)
%
% compute the output of an allpass filter
% with reflection coefficients c (row vector)
% and input x (row vector)
%
n=length(c);
v=zeros(1,n+1);
m=length(x);
y=zeros(1,m);
for k = [1:m]
        u(1)=x(k);
        for j=[n:-1:1]
                u=[d(j) -c(j);c(j) d(j)]*[u(1);v(j)];
                v(j+1)=u(2);
        end
        v(1)=u(1);
        y(k)=v(n+1);
end


function c=reflec(a)
%
% compute the reflection coefficients
% for a min phase polynomial A(z)
% via Jury's algorithm
%
n=length(a)-1;
c=zeros(1,n);
for m=[n:-1:1]
        c(m)=a(m+1);
        a=(a-c(m)*fliplr(a))/(1-c(m)^2);
        a(m+1)=[];
end
```