Table of Contents

```
function nchfilter3(designFile,neff,lam11,dlam1,m,n,BW)
```

Straight forward theory and design specs

params

1. neff: material/structure dependent.
2. m: wavelenght number, m=Lam/FSR.

Spec

1. n: Number of channels.
2. lam11: Wavelength window = [lam11, lam11+FSR1]
3. BW; This will set the minimun cross-couplingvalue, which is related to the gap between the waveguide and the ring.
4. The limits on losses will put limits on the ring sizes. I haven't considred this effect here for now.
5. FSR: is defined based on wavelength diff between channles and the fist wavelength. (FSR > n*lam11, where n is the number of channels in one path.)

Design constraints:

1. Avoid resonant spliting; which gives us the maximun value for cross-coupling coeff.
2. Lower crosstalk which calls for high-Q filter, i.e. higher ordef filters.

```
Note:
Spec is to have the 8 channles in the first channel's first FSR interval
Meaning the channels window is [lam1 lam1+fsr1]
n=8 for an 8-ch filter
We chose to have equally spced channeles with wavelength difference as dla
```

Function's Inputs

1. designFile: which is the full filename (including the path) of output file. This output file contained required netlist params, design spec and some output info about the channel wavelength, FSRs, FWHMs.

2. neff: Index of reflection which is choosen.
3. lam11: Wavelength for the first channel (comes from desin spec)
4. dlam1: Space between channels (comes from desin spec). We have chosen to work on equally spaced channels. It is simple to accomodate not-equally-spaced channels if needed.
5. m: Wavelength number, It is chosen based on wavelength window size. (The calculation is done in parallerFilter.m.
6. n: Number of channels which is given as design spec.

Function's Outputs:

1. Ring sizes,
2. cross-coupling coef.
3. Will also print the calculated, channel wavelenght, FSRs and ... at location "./designSpecs/designSpecPara_<TIME STAMP>.txt"

```
[r1,fsr1,lam12]= getCH1(neff,lam11,dlam1,m,n);

lam(1)=lam11;
r(1)=r1;
fsr(1)=fsr1;
```

```
Not enough input arguments.

Error in nchfilter3 (line 42)
[r1,fsr1,lam12]= getCH1(neff,lam11,dlam1,m,n);
```

just to initialize fwhm

```
fwhm(1)=0;
```

speed of light

```
c = 299792458;
```

Calculating wavelengths, ring sizes and FSRs for each channel.

```
for i=2:1:n
    lam(i)=lam(1)+(i-1)*dlam1;
    r(i)= r(1)*(lam(i)/lam12);
    fsr(i)=(lam(i)^2)/(neff*2*pi*r(i));
end
```

Mlam is the assumed main wavelenght to calculate the total bandwidth. Get the max value for the cross coupling from the constriant on individual channle bandwidth. Channels should not overlap

```
Mlam = lam(1);

% Assumption a=1.0, 'a' is the attenuation factor.
a = 1;
kmax = maxK(neff,r(1),lam(1),r(8),lam(8),a);
```

```
kmin=minK(c,neff,Mlam,r,lam,a,n,BW);

csmax = sqrt(kmax); % cs is the cross-coupling factor used is ospcie
csmin = sqrt(kmin); % cs is the cross-coupling factor used is ospcie

% b is just a random number to pick k between kmin and kmax
b= 0.8;
k = b*kmax; % That is just a picked value less than kmax
% loop to make sure k is greater than kmin
while k < kmin
    b=b+0.005;
    k=b*kmax;
end

% c is the cross-coupling factor used is ospice
cs= k^2;

% Assumption: All rings have same cross-coupling coeff.
% 'rc' is the self-coupling factor (trasmission factor)
rc = sqrt(1-k^2);

% Calulationg widthline (FWHM) of each channel.
for i=1:n
    fwhm(i)=((1-rc^2.*a)*lam(i)^2)/(pi*2*pi*r(i)*neff*rc*sqrt(a));
end

totDlam = sum(fwhm);
% totBW is the bandwidth of the siganl which should be >= bandwidth
 given
% as design spec.
totBW= c*(totDlam/Mlam^2);

% Write design spec and the calculated output in the output file.
outID=fopen(designFile,'a+');
fprintf(outID,'%s\n','Design Spec:');
SpecFormat ='lam11=%1.5e\nm=%d\nn=%d\nBW=%1.5e\n';
fprintf(outID,SpecFormat, lam11, m, n, BW);
fprintf(outID,'Design param and outputs\n');
fprintf(outID,'neff=%1.5e\n', neff);
fprintf(outID,'k=%1.5e ==> cs=%1.5e\n', eval(k), eval(cs));
fprintf(outID,'a=%1.5e\n', a);
fprintf(outID,'%10s , %10s , %10s, %10s\n','lam' , 'R' , 'FSR'
 , 'FWHM');
for i=1:1:n
    fprintf(outID,'%1.4e , %1.4e , %1.4e, %1.4e
\n',lam(i),r(i),fsr(i),fwhm(i));
end

fprintf(outID,'\n\n%s\n\n','Table of wavelengths for the channels in
 this path');
for i=1:1:n
    fprintf(outID,'ch%d: %1.4e , %1.4e , %1.4e, %1.4e, %s
\n',i ,lam(i),lam(i)+fsr(i), lam(i)+2*fsr(i),lam(i)+3*fsr(i), '...');
end
```

```
end
```

Function getCH1

To get design params for channel one (based on explanation above and striaght forward theory.

```
function [r,fsr,lamf,dlam]= getCH1(neff,lam11,dlam,m,n)
r=(m*lam11)/(neff*2*pi);
fsr=(lam11^2)/(neff*2*pi*r);
lamf=lam11+fsr;
% Spec is to have the 8 channles in the ch1's first FSR interval
% Meaning the channels window is [lam1 lam2] where lam2 is slightly
 begger than lam1+fsr1
% The calculation to choose FSR and m are done in parallelFilter.m
% n=8 for an 8-ch filter
% Note: We chose to have equally spced channeles with wavelength
 difference as dlam
% i.e. dlam = 3*(10^(-9)) which is given as an input to this function;
end
```

Function maxK

Resosnat spliting happens because of channel interference with each other.This can be avoided by proper design of filter widthline. The funcntion maxK gives us the maximum value of cross-coupling to avoid resonant spiliting.

```
function k1 = maxK(neff,r1,lam1,r2,lam2,a)
syms k
rk=sqrt(1-k^2);
fwhm1=((1-a*rk^2)*lam1^2)/(pi*2*pi*r1*neff*rk*sqrt(a));
fwhm2=((1-a*rk^2)*lam2^2)/(pi*2*pi*r2*neff*rk*sqrt(a));
x1=lam1+0.5*fwhm1;
x2=lam2-0.5*fwhm2;
[solk]=solve(x1-x2==0);
k1=vpa(solk);
k1=k1(k1>0);
end
```

Function minK

To meet the bandwidth spec, cross-coupling can not be too too low. The minK function gives us the minimun value for cross-coupling to meet the bandwidth spec.

```
function k1=minK(c,neff,Mlam,r,lam,a,n,BWv)
syms k
rk=sqrt(1-k^2);

for i=1:n
    fwhm(i)=((1-rk^2.*a)*lam(i)^2)/(pi*2*pi*r(i)*neff*rk*sqrt(a));
end

bw = (c/Mlam^2)*sum(fwhm);
```

```
[solk]=solve(bw==25*BWv);
k1=vpa(solk);
k1=k1(k1>0);
vpa(fwhm(1));
vpa(bw);
end
```

Published with MATLAB® R2015b