I have some comments here on your draft:

> \* Use the new HS Bremen Logo

>

> \* Have less paragraphs taken from other sources, Adding the reference number to the end of the paragraph is not good practice in a thesis, as a thesis shall be a report generated solely by the student and not a listing of paragraphs from others.

>

> \* Ty to create more figures yourself, such that you get a consistent look and get less figures with fuzzy artifacts.

>

> \* Have references to all figures and texts taken from other sources.

> \* Do not uses the dotted lines in the equations

>

> \* Figure 26: plot the figure, such that it becomes clear, that they match. Maybe on top of each other. You can also get the Theory curve from the AWR-model.

>

> \* Figure 34, 37: Can you overlay them with the measurements, then you will see how good or bad the models are.

>

> \* In your Chapter five you discuss your results. Its not clear to me, how your plots come together and how they are related. Please, make his clear.

>

> \* In the case study, I expect to see the circuit schematics of the designed filters. And it would be nice to see a before after plot of the effects of the filter. The legends of the plots are to small to see what they mean.

amplTable = ones(1, numberOfTones);

% Actual tone frequencies are calculated for each tone

freqTable = startFreq : toneSpacing : finalFreq;

% Phase initialized to tones

phaseTable = zeros(1, numberOfTones);

% Notch Insertion

if notchStarTone > 0 && notchNumOfTones > 0

amplTable(notchStarTone : notchStarTone + notchNumOfTones - 1) = 0.0;

end

% Phase Set Up

% Random Phase

if phaseDistType == 1

phaseTable = 2.0 \* pi .\* (rand(1, numberOfTones) - 0.5);

end

% Parabolic Phase (NEWMAN)

if phaseDistType == 2

phaseTable = 1:numberOfTones;

phaseTable = 1.0 - phaseTable .\* phaseTable;

phaseTable = -(pi / numberOfTones) .\* phaseTable;

phaseTable = wrapToPi(phaseTable);

end

% Rudin Sequence (near optimal for 2^N tones)

if phaseDistType == 3

numOfSteps = int16(round(log(numberOfTones) / log(2)));

if 2^numOfSteps < numberOfTones

numOfSteps = numOfSteps + 1;

end

numOfSteps = numOfSteps - 1;

temPhase(1:2) = 1;

for n = 1:numOfStep

m = int16(length(temPhase) / 2);

temPhase = [temPhase, temPhase(1 : m), -temPhase(m + 1 : 2 \* m)];

end

phaseTable = -0.5 \* pi .\* (temPhase(1 : numberOfTones) - 1);

end

% Multi-tone Waveform Calculation (I/Q)

multiToneWfm = zeros(1, wfmLength);

freqTable = 2.0 \* pi \* freqTable;

for i = 1:numberOfTones

multiToneWfm = multiToneWfm + ...

amplTable(i) .\* cos(freqTable(i) .\* xValues + phaseTable(i));

end

% Normalization to the -1.0/+1.0 Range

maxValue = max(abs(multiToneWfm));

multiToneWfm = multiToneWfm ./ maxValue;

% Calculation of PAPR (Crest Factor)

papr = 1.0 / std(multiToneWfm);

papr = 20.0 \* log10 (papr) - 3.0;

% File Creation

if saveFile

csvwrite(fName, multiToneWfm');

end

% Plot Spectrum

pSpec = pspectrum(flattopwin(1, wfmLength) .\* multiToneWfm);

pSpec = 10.0 \* log10(pSpec);

pSpec = pSpec-max(pSpec);

xFreq = 0:(length(pSpec)-1);

xFreq = xFreq / (2 \* length(pSpec));

xFreq = xFreq \*samplingRate;

plot(xFreq, pSpec);

axis([-inf +inf -80 20]);

title(strcat('Power Spectrum (dBc), PAPR = ', num2str(papr), 'dB'));

plot(xFreq, pSpec);

axis([-inf +inf -80 20]);

title(strcat('Power Spectrum (dBc), PAPR = ', num2str(papr), 'dB'));

%-- 10/13/2021 9:28 AM --%

Untitled

sparameters

%-- 10/24/2021 6:17 AM --%

obj1=sparameter('Common filter.s2p')

obj1=sparameters('Common filter.s2p')

s11=rfparam(S,1,1)

s11=rfparam(obj1,1,1)

s21=rfparam(obj1,2,1)

s12=rfparam(obj1,1,2)

s22=rfparam(obj1,2,2)

Z=50

A=((((1+s11)\*(1-s22))+(s12\*s21))/(2\*s21))

a1=1+s11

a2=1-s22

a3=s12\*s21

a3=s12.s21

a3=s12.\*s21

A=((((1+s11).\*(1-s22))+(s12.\*s21))/(2.\*s21))

B=50.\*((((1+s11).\*(1+s22))-(s12.\*s21))/(2.\*s21))

C=(1/50).\*((((1-s11).\*(1-s22))-(s12.\*s21))/(2.\*s21))

A=((((1-s11).\*(1+s22))+(s12.\*s21))/(2.\*s21))

A=((((1+s11)\*(1-s22))+(s12\*s21))/(2\*s21))

D=((((1-s11).\*(1+s22))+(s12.\*s21))/(2.\*s21))

A=((((1+s11).\*(1-s22))+(s12.\*s21))/(2.\*s21))

TF=50/((50.\*A)+B+(50.\*50.\*C)+(50.\*D))

TF=50./((50.\*A)+B+(50.\*50.\*C)+(50.\*D))

rfplot(TF)

TF=50./((50.\*A)+B+(2500.\*C)+(50.\*D))

TF=50./((50.\*A)+B+(2500.\*C)+(50.\*D));

rfplot(TF)

rfplot('TF')

rfplot(TF)

TF=(50)./((50.\*A)+B+(2500.\*C)+(50.\*D))

func=((50.\*A)+B+(2500.\*C)+(50.\*D))

func=((50.\*A).+B.+(2500.\*C).+(50.\*D))

func1=((50.\*A)

A=((((1-s11).\*(1+s22))+(s12.\*s21))/(2.\*s21))

cls

clr

clear

s11=rfparam(S,1,1)

s11=rfparam(obj1,1,1)

obj1=sparameters('Common filter.s2p')

s11=rfparam(obj1,1,1)

s12=rfparam(obj1,1,2)

s21=rfparam(obj1,2,1)

s22=rfparam(obj1,2,2)

A=((((1-s11).\*(1+s22))+(s12.\*s21))/(2.\*s21))

s11=rfparam(obj1,1,1)

s12=rfparam(obj1,1,2)

s21=rfparam(obj1,2,1)

s22=rfparam(obj1,2,2)

A1=((((1-s11).\*(1+s22))+(s12.\*s21)))

A2=(2.\*s21)

A=A1/A2

A1=((((1-s11).\*(1+s22))+(s12.\*s21)))

A2=(2.\*s21)

obj1=sparameters('Common filter.s2p')

s11=rfparam(obj1,1,1)

s12=rfparam(obj1,1,2)

s21=rfparam(obj1,2,1)

s22=rfparam(obj1,2,2)

A=((((1-s11).\*(1+s22))+(s12.\*s21))./(2\*s21))

B=50\*((((1+s11).\*(1+s22))-(s12.\*s21))./(2\*s21))

C=(1/50)\*((((1-s11).\*(1-s22))-(s12.\*s21))./(2\*s21))

D=((((1-s11).\*(1+s22))+(s12.\*s21))./(2\*s21))

TF=(50)/((50\*A)+B+(2500\*C)+(50\*D))

rfplot(TF)

help rfplot

size(TF)

real(TF)

x=real(TF)

plot(x)

size(TF)

TF=(50)./((50\*A)+B+(2500\*C)+(50\*D))

size(TF)

real(TF)

plot(TF)

obj2=sparameters('diff\_filter\_with\_transformers.s2p')

plot(TF)

S11=rfparam(obj2,1,1)

S12=rfparam(obj2,1,2)

S21=rfparam(obj2,2,1)

S22=rfparam(obj2,2,2)

A1=((((1-s11).\*(1+s22))+(s12.\*s21))./(2\*s21))

A1=((((1-S11).\*(1+S22))+(S12.\*S21))./(2\*S21))

B1=50\*((((1+S11).\*(1+S22))-(S12.\*S21))./(2\*S21))

C1=(1/50)\*((((1-S11).\*(1-S22))-(S12.\*S21))./(2\*S21))

D1=((((1-S11).\*(1+S22))+(S12.\*S21))./(2\*S21))

TF1=(50)./((50\*A1)+B1+(2500\*C1)+(50\*D1))

size(TF1)

plot(TF)

obj2=sparameters('diff\_filter\_with\_transformers.s2p')

S11=rfparam(obj2,1,1)

S12=rfparam(obj2,1,2)

S21=rfparam(obj2,2,1)

S22=rfparam(obj2,2,2)

A1=((((1-S11).\*(1+S22))+(S12.\*S21))./(2\*S21))

B1=50\*((((1+S11).\*(1+S22))-(S12.\*S21))./(2\*S21))

C1=(1/50)\*((((1-S11).\*(1-S22))-(S12.\*S21))./(2\*S21))

D1=((((1-S11).\*(1+S22))+(S12.\*S21))./(2\*S21))

TF1=(50)./((50\*A1)+B1+(2500\*C1)+(50\*D1))

plot(TF)

plot(TF1)

obj3=sparameters('transformer\_and\_transformer.s2p')

St11=rfparam(obj3,1,1)

St12=rfparam(obj3,1,2)

St21=rfparam(obj3,2,1)

St22=rfparam(obj3,2,2)

At1=((((1-St11).\*(1+St22))+(St12.\*St21))./(2\*St21))

Bt1=50\*((((1+St11).\*(1+St22))-(St12.\*St21))./(2\*St21))

Ct1=(1/50)\*((((1-St11).\*(1-St22))-(St12.\*St21))./(2\*St21))

Dt1=((((1-St11).\*(1+St22))+(St12.\*St21))./(2\*St21))

TFt1=(50)./((50\*At1)+Bt1+(2500\*Ct1)+(50\*Dt1))

plot(TFt1)

obj4=sparameters('common1.s2p')

Sc11=rfparam(obj4,1,1)

Sc12=rfparam(obj4,1,2)

Sc21=rfparam(obj4,2,1)

Sc22=rfparam(obj4,2,2)

Ac1=((((1-Sc11).\*(1+Sc22))+(Sc12.\*Sc21))./(2\*Sc21))

Bc1=50\*((((1+Sc11).\*(1+Sc22))-(Sc12.\*Sc21))./(2\*Sc21))

Cc1=(1/50)\*((((1-Sc11).\*(1-Sc22))-(Sc12.\*Sc21))./(2\*Sc21))

Dc1=((((1-Sc11).\*(1+Sc22))+(Sc12.\*Sc21))./(2\*Sc21))

TFc1=(50)./((50\*Ac1)+Bc1+(2500\*Cc1)+(50\*Dc1))

plot(TFc1)

%-- 11/19/2021 8:11 PM --%

load('matlab.mat')

%-- 11/19/2021 11:48 PM --%

Untitled

load Common filter.s2p

load('matlab.mat')

obj1=sparameter('Common filter.s2p')

obj1=sparameters('Common filter.s2p')