Politecnico di Torino

Guiding Electromagnetic Systems

Microstrip Lowpass Filter Design

Project 6, Report

Referents: Paola Pirinoli

Authors: Burco Adriele, Burrello Alessio, Cesarano Giuseppe, Fogliato Martina

Contents

Aim of the project

The main requirements of this group project were:

- Design of both T and π prototype network for a lowpass filter fulfilling the given data (reported in Table 1)
- Design the same filter using the stepped impedance technique
- Simulation of the frequency response of the filter
- Design the filter using AWR, performing extraction, printing the corresponding GERBER file and simulating again the frequency response
- Measure the actual frequency response of the filter comparing it with the expected one

Response type	Equal ripple 0.5dB
f_1[GHz]	2.4
$R_0[\Omega]$	50
Insertion loss [dB] @ $f = 4.8$[GHz]	> 30

Table 1: provided data for lowpass filter design

Theoretical Design

Prototype filter

The first step was to identify the order of the filter respecting the Insertion Loss at the given frequency and the equal ripple 0.5dB response type. By a quick look at the graph provided on the slides (using $\Omega = \frac{\omega}{\omega_c} = 2$), we could see that the smallest possible order was N=5. This corresponds to: $g_1 = 1.7058$, $g_2 = 1.2296$, $g_3 = 2.5408$, $g4 = g2$, $g5 = g1$, $g6 = 1$.
We decided to perform all computations using RFTool provided by MatLab, which allows us to model, analyze and visualize RF components:

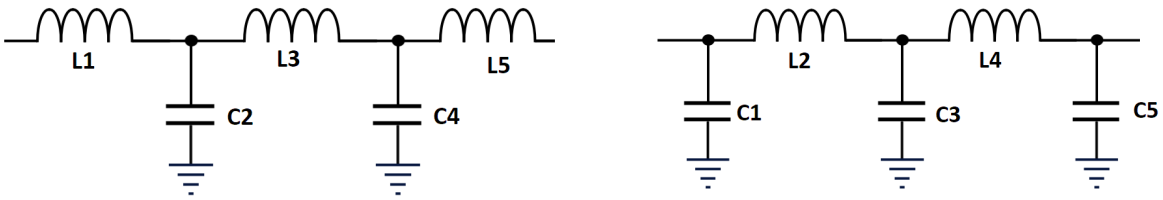
```
%% T  network
Lt(1)= Z0*g1/wc, Ct(1)=g2/(Z0*wc), Lt(2)=Z0*g3/wc, Ct(2)=g4/(Z0*wc),
Lt(3)=Z0*g5/wc

LP_T_filter = rfckt.lclowpasstee('L',Lt,'C',Ct);
analisys_T  = analyze(LP_T_filter,f);

%% pi Network
Cp(1)=g1/(Z0*wc), Lp(1)=Z0*g2/wc, Cp(2)=g3/(Z0*wc), Lp(2)=Z0*g4/wc,
Cp(3)=g5/(Z0*wc)

LP_pi_filter = rfckt.lclowpasspi('L',Lp,'C',Cp);
analisys_pi  = analyze(LP_pi_filter,f);
```

The code makes MatLab compute the conversion and denormalization of g parameters into L and C and analyze both networks (T and π).
The structures are represented in the following Figure:



Stepped impedance technique

Next step was to design one of the two networks (we chose the T one) using Stepped Impedance technique. The overall filter will be composed of 5 microstrip line portions, representing L and C components, of different length, width and characteristic impedance.
Again, we exploited RFTool:

```
%% microstrip lowpass filter

w1=0.0005818; w2=0.0146734;
er=4; h=0.002;
e_eff1=(er+1)/2+(er-1)/2*1/(sqrt(1+12*h/w1));
e_eff2=(er+1)/2+(er-1)/2*1/(sqrt(1+12*h/w2));
Z_inf_l=20;
```

```
Z_inf_h=120;
k1=2*pi/c*sqrt(e_eff1)*fc;
k2=2*pi/c*sqrt(e_eff2)*fc;
l1= (g1*R0)/(Z_inf_h*k1)
l2= (g2*Z_inf_l)/(R0*k2)
l3= (g3*R0)/(Z_inf_h*k1)
l5=l1;
l4=l2;

microstrip1=rfckt.microstrip('EpsilonR',er,'Height',h,'Linelength',l1,'Width',w1);
microstrip2=rfckt.microstrip('EpsilonR',er,'Height',h,'Linelength',l2,'Width',w2);
microstrip3=rfckt.microstrip('EpsilonR',er,'Height',h,'Linelength',l3,'Width',w1);
microstrip4=rfckt.microstrip('EpsilonR',er,'Height',h,'Linelength',l4,'Width',w2);
microstrip5=rfckt.microstrip('EpsilonR',er,'Height',h,'Linelength',l5,'Width',w1);

net=rfckt.cascade;
net.Ckts={microstrip1,microstrip2,microstrip3,microstrip4,microstrip5};
an_net=analyze(net,f_int);
```

From the code above it can be seen that the width of the microstrip corresponds to 581.8µm for the inductive part and 14.6734mm for the capacitive part, which we computed using another, not reported, MatLab script to implement the formula.
Characteristic impedances were given: 120Ω corresponds to the inductor and 20Ω to the capacitor.
As it can be seen, k and ϵ_{eff} has been computed with the proper formula.
We obtained the following lengths: $l_1 = 8.6$mm, $l_2 = 5.3$mm, $l_3 = 12.7$mm.
Then, the five microstrip lines have been connected together using the *cascade* RFTool command.

Simulation of frequency response with MatLab

The following plots show the simulation of the frequency response of the filter with the first script (prototype T) and then with the second one (stepped impedance), on which we highlighted the attenuations at 2.4GHz and 4.8GHz:

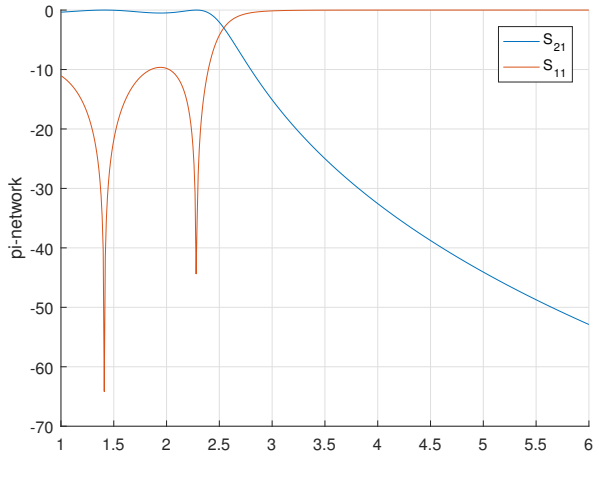


Figure 1: prototype filter simulation

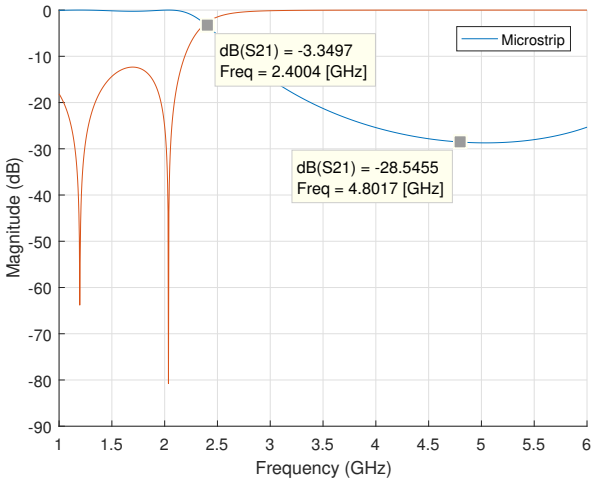


Figure 2: stepped impedance simulation

The blue curve represents S21 while the red one S11. It can be seen that the behavior is the desired one although it presents some differences between the first and the second implementation.

AWR Design

We designed the filter in AWR, adding two 50Ω ports and using length and width values just found through the MatLab implementation and we draw the corresponding layout.
Then, we associated the network elements with an Extract Block in the schematic.
However, while performing extraction, which should simulate something closer to the real behavior of the circuit, we noticed that the original requirements were not respected anymore (3dB frequency and attenuation). Therefore, we performed a tuning operation of the width and length parameters in order to have a final outcome which is within requirements.
We got the following new parameters:

- $l_1 = 6.678\text{mm}, \quad w_1 = 0.451\text{mm}$
- $l_1 = 4.617\text{mm}, \quad w_1 = 13.760\text{mm}$
- $l_1 = 9.221\text{mm}, \quad w_1 = 0.525\text{mm}$

Figure 3 shows the layout after this phase.

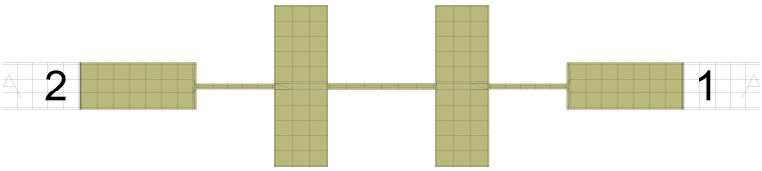


Figure 3: Layout of the filter

The layout of the filter, for which we chose Copper, lays above two layers of dielectric material ($\epsilon_r = 4$, $h = 1\text{mm}$).
After that, we added ground as last step, paying attention the two edges coincide with the ones of the ground layer, in order to attach connectors properly during the measurement phase.
Eventually we exported the GERBER file, checking with a proper viewer software that the ground was flush with the conductor edges.

Simulation of frequency response with AWR

Figure 4 shows the bode plot of S21 parameter before and after the extraction phase, with the new values of w and l.
It can be noticed that the pink curve does not respect the requirements anymore, but the blue one does. The curve after extraction should be closer to the result of measurements on the produced filter, taking into account the influence of parasitic components too.
We highlighted the attenuation at 2.4GHz and 4.8GHz, together with the minimum and maximum value of the ripple in the passband.

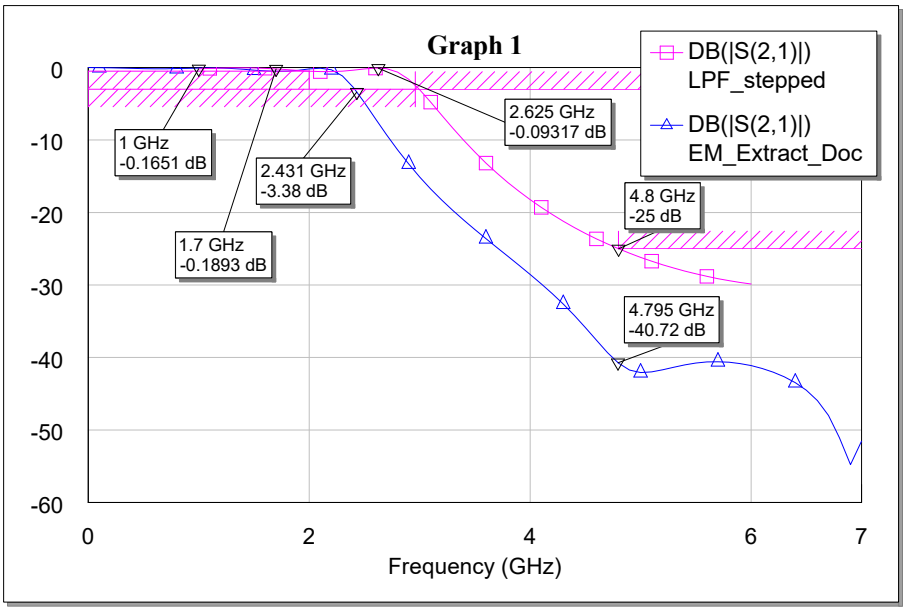


Figure 4: Frequency response with AWR before (pink) and after (blue) extraction

Measurements

The filter implemented with the stepped impedance technique appears as in Figure 3. We can notice the presence of three very thin sections, representing the inductors, alternating with two sections which are much thicker, that represent the inductors. Once the filter was produced, we measured the useful parameters of the scattering matrix allowing us to see if the circuit behaves as expected. The measurements were done with the network analyzer.

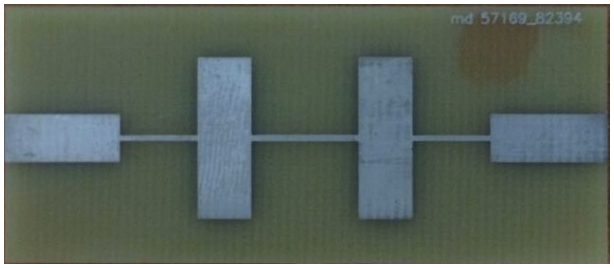


Figure 5: Microstrip filter after production

From Figures 6, 7 and 8 we can notice that:

- The magnitude of the parameter S_{21} at the cutoff frequency, that is 2.4 GHz, is -4.57 dB.
- The magnitude of the parameter S_{21} at the frequency 4.8 GHz, is -34.83 dB. We wanted an Insertion Loss at 4.8 GHz bigger than 30dB, so this requirement has been satisfied
- The value of the parameter S_{11} is 0, as expected.

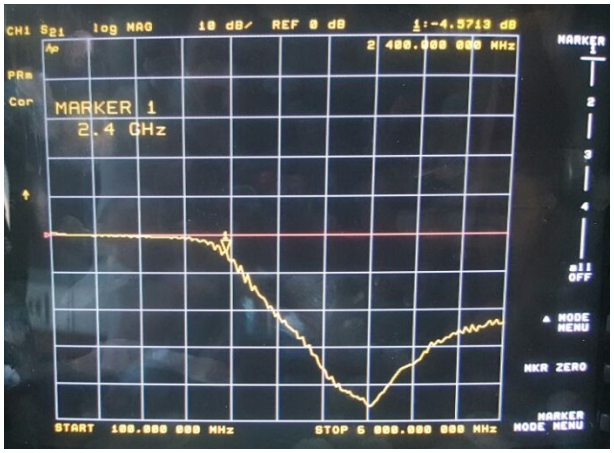


Figure 6: S_{21} @ 2.4 GHz

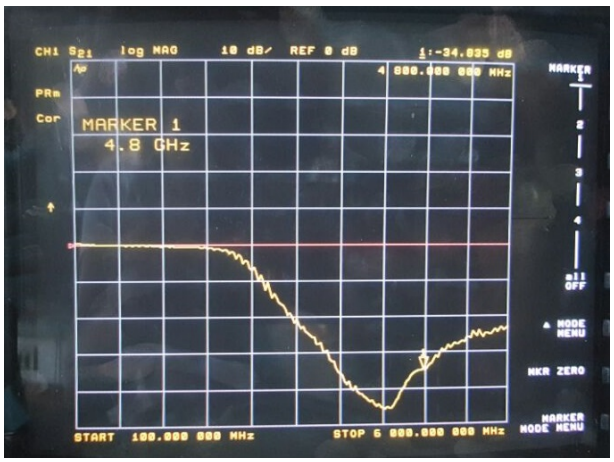


Figure 7: S_{21} @ 4.8 GHz

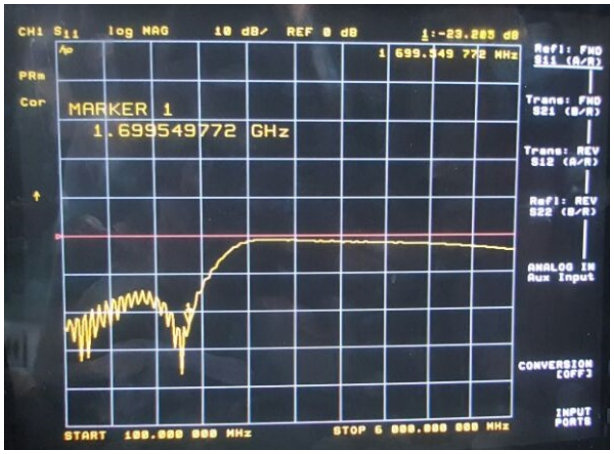


Figure 8: S11

Comments and conclusions

Overall, we could be satisfied with the recorded results, since the original requirements of the assignments (ripple, attenuation at 4.8GHz and -3dB frequency at 2.4GHz) are all met. The cutoff frequency is actually slightly smaller than expected, at 2.4GHz we have an attenuation of 4.57 dB, but this uncertainties could be due to the connectors of the network analyzer.
In fact, we repeated the measurements a few times, since the connectors affected the behavior.
Also, comparing the picture of the screen of the network analyzer with the frequency simulation performed by AWR (after extraction, Figure 4) we could notice quite matching behavior.