**ФНЧ 3 порядка (500кГц)**

С помощью существующих усилительных элементов был собран фильтр ФНЧ 3-го порядка с частотой среза 500 кГц. Номиналы элементов были рассчитаны с помощью программы расчета (Приложение А). На рисунке 1 представлен фильтр.

В таблице 1 и на рисунке 2 представлены характеристики фильтра

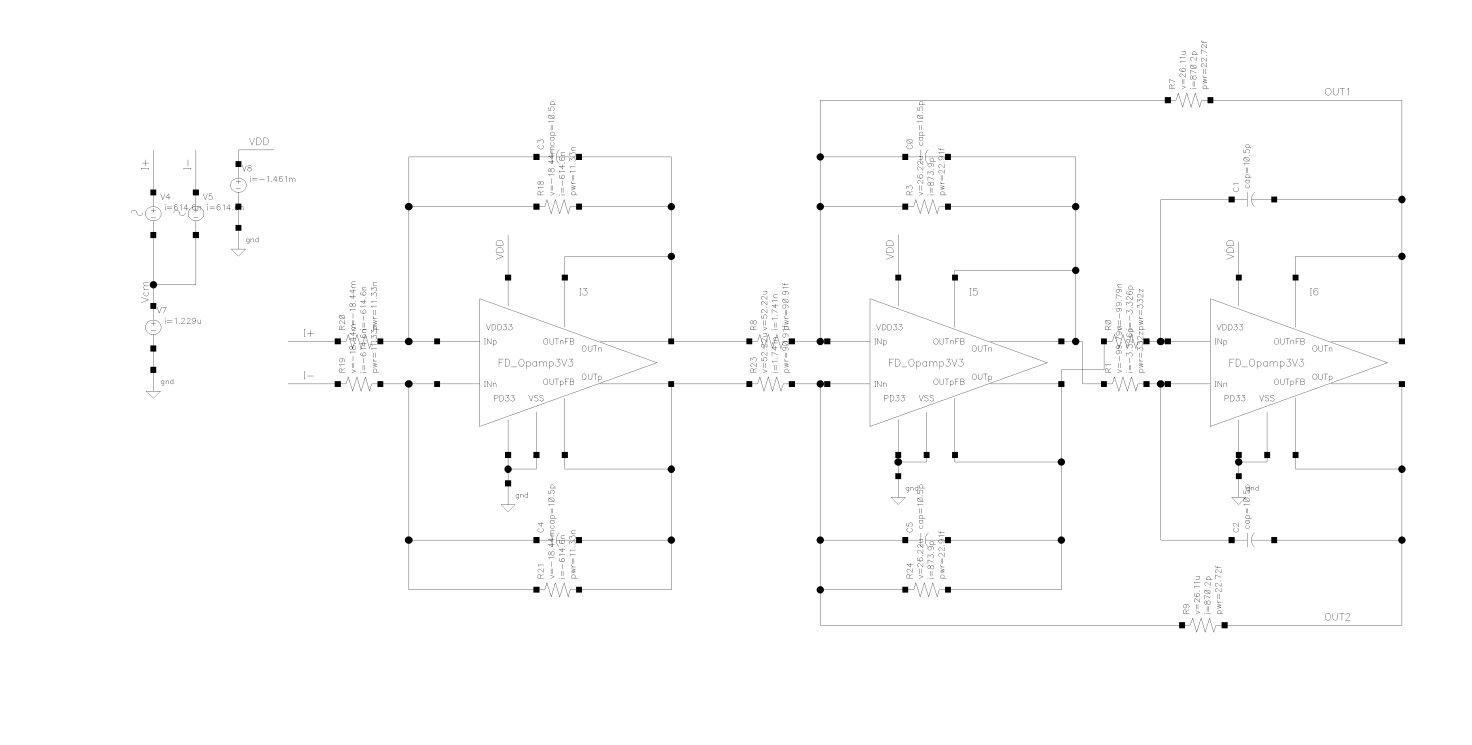


Рисунок 1 – Установка для моделирования

Таблица 1 – Параметры фильтра

|  |  |  |
| --- | --- | --- |
| Параметр | Значение | Размерность |
| Тип фильтра | Баттерворт |  |
| Порядок фильтра | 3 |  |
| Частота среза | 500 | кГц |
| Усиление | 0 | дБ |
| Ток потребления | 1.5 | мА |
| Напряжение питания | 3.3 | В |

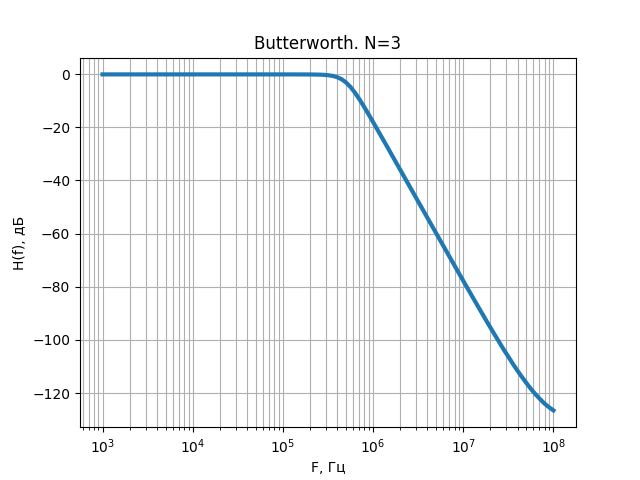


Рисунок 2 – Коэффициент передачи

Используемые элементы для построения ФНЧ были взяты из блока полифазных фильтров, где важную роль играло выходное сопротивление ОУ. В данном случае требования к выходному сопротивлению не такие жесткие и ток потребления возможно уменьшить до значений менее 1 мА.

Также стоит отметить, что первый каскад фильтра может быть включен в выходной каскад пассивного смесителя, что также уменьшит общий ток потребления.

Приложение А – Листинг программы расчета

from scipy import signal  
import matplotlib.pyplot as plt  
import numpy as np  
  
plt.rcParams["font.family"] = "Century Gothic"  
plt.rcParams["font.size"] = "14"  
  
#parameters of filter  
Order = 3  
Freq = 500e3  
start\_f = 1e5  
stop\_f = 5e6  
N\_points = 100  
  
#initial parameters for Cap/Res  
R\_2\_4 = 30e3  
pi = 3.1415  
H = 1  
  
#calculate numerator(b) and denumerator(a)  
b, a = signal.butter(Order, Freq, btype='lowpass', analog=True, output='ba')  
  
#create H(f) function  
f, h = signal.freqs(b, a, worN=np.logspace(np.log10(start\_f), np.log10(stop\_f), N\_points))  
  
#calculate poles/zeros  
z, p , k = signal.butter(Order, Freq, btype='lowpass', analog=True, output='zpk')  
  
print('\*---Poles---\*')  
for i in range(len(p)):  
 print(np.round(p[i], 3))  
print('\*-----------\*')  
  
  
#calculate R/C for blocks  
print('\*---Filter---\*')  
for i in range(len(p)):  
 round\_n = 3  
 U\_Freq = abs(p[i])/Freq  
 Q = -abs(p[i]) / (2 \* p[i].real)  
  
 #calculate components  
 C\_1\_2 = np.sqrt(1/(((2\*pi\*Freq)\*\*2)\*(R\_2\_4\*\*2)))  
 R\_1 = Q\*R\_2\_4  
 R\_3 = R\_2\_4/H  
  
 #round parameters  
  
 R\_1 = round(R\_1, round\_n)  
 R\_2\_4 = round(R\_2\_4, round\_n)  
 R\_3 = round(R\_3, round\_n)  
  
 if Order%2 != 0:  
 if i < int(Order/2):  
 print('\*------\*')  
 print(f'Second order block {i+1}. Q={round(Q,round\_n)}')  
 print(f'R1={R\_1} Ом\nR2={R\_2\_4} Ом\nR3={R\_3} Ом\nR4={R\_2\_4} Ом\nR5={R\_2\_4} Ом')  
 print(f'C1={C\_1\_2} Ф\nC2={C\_1\_2} Ф')  
 print('\*------\*')  
  
 elif i == int(Order/2):  
 print('\*------\*')  
 print(f'First order block {i+1}. Q={round(Q,round\_n)}')  
 print(f'R1={R\_2\_4} Ом\nC1={C\_1\_2} Ф')  
 print('\*------\*')  
 else:  
 if i < int(Order/2):  
 print('\*------\*')  
 print(f'Second order block {i + 1}. Q={round(Q, round\_n)}')  
 print(f'R1={R\_1} Ом\nR2={R\_2\_4} Ом\nR3={R\_3} Ом\nR4={R\_2\_4} Ом\nR5={R\_2\_4} Ом')  
 print(f'C1={C\_1\_2} Ф\nC2={C\_1\_2} Ф')  
 print('\*------\*')  
  
#Plot conditionals  
color\_1='Tab:orange'  
color\_2='Tab:blue'  
  
#Plot H(f)  
plt.figure()  
plt.title(f'Butterworth. N={Order}')  
plt.semilogx(f, 20\*np.log10(abs(h)), linewidth=3, color=color\_1, label='Ideal')  
plt.xlabel('F, Гц')  
plt.ylabel('H(f), дБ')  
plt.legend()  
plt.grid(which='both', axis='both')  
plt.show()

**Results**

\*---Poles---\*

(-250000+433012.702j)

(-500000-0j)

(-250000-433012.702j)

\*-----------\*

\*---Filter---\*

\*------\*

Second order block 1. Q=1.0

R1=30000.0 Ом

R2=30000.0 Ом

R3=30000.0 Ом

R4=30000.0 Ом

R5=30000.0 Ом

C1=1.0610642474401825e-11 Ф

C2=1.0610642474401825e-11 Ф

\*------\*

\*------\*

First order block 2. Q=0.5

R1=30000.0 Ом

C1=1.0610642474401825e-11 Ф

\*------\*