h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Hochschule Darmstadt

Fachbereich Elektrotechnik und Informationstechnik

International Master of Science of Electrical Engineering and Information Technology

System Driven Hardware Design

Bandpass Test Assignment

12.07.2020

Lecturer: Prof. Dr.-Ing. BANNWARTH, Stephan

Laboratory Evaluator: MSc. HUBRICH, Michael

AGUILAR AGUILA ISAIAS, Oscar   Matr. 766468

Table of Contents

Table of Figures

Table of Tables

1. Summary

During this semester's System Driven Hardware Design (SDHD) course, a band pass amplifier was designed, in order to amplify the signal of a IPM 165 24GHz radar and then feed it to a FreeSoc2's ADC module, in order to subsequently pass that signal's data through a UART port to Matlab, and to be visualized and analyze through a Fast Fourier Transform (FFT).

This project was divided into three parts:

1) Design of the circuit's PCB.
2) Implementation of the FreeSoc's software.
3) Testing the designed circuit using and software together.

The main deliverable for this part of the project is the Bandpass testing (3). Therefore, the following report focuses on the testing of both the software and the hardware together, by using a breadboard to contain the designed circuit, as a PCB was not manufacturable during these Corona times, as well as using the already-implemented FreeSoc2 software along with a professor-provided-now-modified Matlab software, to capture different voltage outputs at different frequencies to test for both amplification and filtering done by the circuit. These values will then be compared to the professor's simulated data from a simulation on LTSpice of the same circuit, by using his raw data and then using the LTspice2Matlab function.

2. Measurement set-up description

a) Hardware

The amplifier hardware was implemented following the schematic from Figure 1.

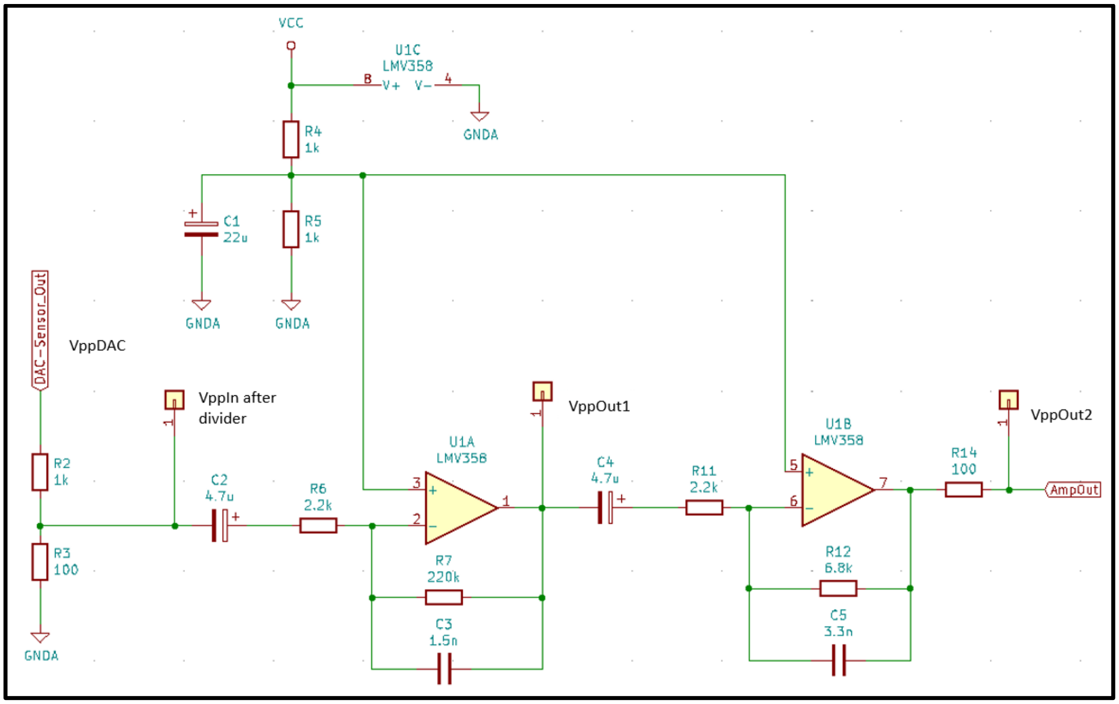


Figure 1 Amplifier / Filtering Circuit Diagram

It is worth mentioning that instead of using the actual sensor for this mock-up testbench, a voltage divider paired with the FreeSoc's DAC sine wave generator was used in order to emulate the input of the circuit. The designed total gain of the amplifier being 300 and the desired output voltage being in the range of 2.5V, the input voltage divider was designed with the following output voltage:

$$V_{out,divider} = V_{in,bandpass} = \frac{V_{out,bandpass}}{A_{total}} = \frac{2.5V}{300} = 8.3mV$$

where V_{in} corresponds to the amplifier's input. Furthermore the voltage divider then uses the input Vpp voltage delivered by the FreeSoc's DAC (40mV) as follows:

$$V_{out,divider} = \frac{R_3}{R_3 + R_2} V_{in,divider}$$

$$\frac{40mV}{8.3mV} = \frac{R_3 + R_2}{R_3}$$

$$4.82R_3 = R_3 + R_2$$

$$R_3(4.82 - 1) = R_2$$

$$R_3(3.82) = R_2$$

However, upon further discussion in class, following the professor's recommendation on what was functional in his circuit, a relation of $R_3(10) = R_2$ was chosen instead, upon the professor's recommendation of using R3=100Ω, R2=1KΩ.

The resulting circuit can be appreciated in both Figure 2 and Figure 3.

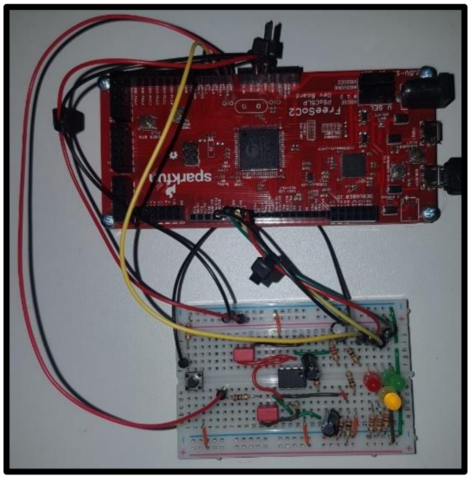


Figure 2 Complete picture of implemented hardware

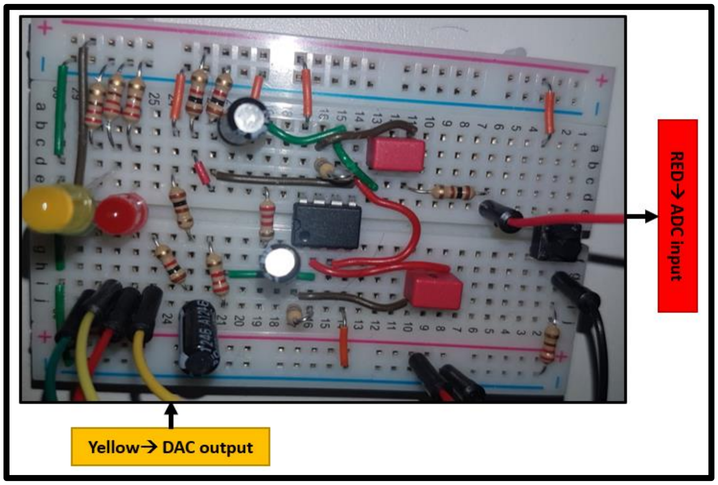


Figure 3 Detailed hardware view of the amplifier/bandpass circuit

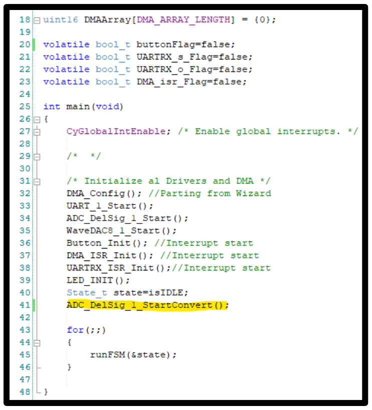
b) Software

Two parts of the software had to be setup for the measurements:

1. PSOC programming in C

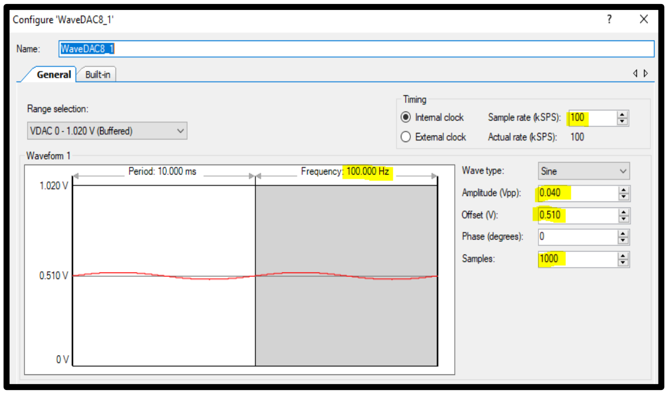
This part, although relevant for the project altogether, is not the focus of this deliverable. It however, is worth mentioning and explaining some tweaks along with some changes necessary for the testbench's testability:

- In order to correctly and more precisely measure the values from the testbench, the following recommendation was taken from the laboratory second review: the ADC was left on since the modules' initialization. The measurement just turned out clearer this way, and the code's modularity made it easier to implement said change (Figure 4).
- The software was modified to expect only 5 transfers per measurement.
- In order to measure different frequencies, the DAC's output had to be changed to match the desired frequencies (Figure 5). Additionally, the ADC had to be modified accordingly in order to comply to the Shannon-Nyquist Criterion in order to correctly sample higher frequencies.



```
uint16 DMAArray[DMA_ARRAY_LENGTH] = {0};

volatile bool_t buttonFlag=false;
volatile bool_t UARTRX_s_Flag=false;
volatile bool_t UARTRX_o_Flag=false;
volatile bool_t DMA_isr_Flag=false;

int main(void)
{
    CyGlobalIntEnable; /* Enable global interrupts. */

    /*  */

    /* Initialize al Drivers and DMA */
    DMA_Config(); //Parting from Wizard
    UART_1_Start();
    ADC_DelSig_1_Start();
    WaveDAC8_1_Start();
    Button_Init(); //Interrupt start
    DMA_ISR_Init(); //Interrupt start
    UARTRX_ISR_Init();//Interrupt start
    LED_INIT();
    State_t state=isIDLE;
    ADC_DelSig_1_StartConvert();

    for(;;)
    {
        runFSM(&state);
    }

}
```

Figure 4 Code Snippet from PSOC Creator's main file



Configure 'WaveDAC8_1'

Name: WaveDAC8_1

General | Built-in

Range selection: VDAC 0 - 1.020 V (Buffered)

Timing

● Internal clock   Sample rate (kSPS): 100

○ External clock   Actual rate (kSPS): 100

Waveform 1

Period: 10.000 ms   Frequency: 100.000 Hz

Wave type: Sine

Amplitude (Vpp): 0.040

Offset (V): 0.510

Phase (degrees): 0

Samples: 1000

Figure 5 Wave DAC configuration for 100Hz, 40mVpp

2. Matlab

The testbench software provided by the professor was modified into "Testbench_COM2Matlab_OAguilar.m", in order to print out the maximum voltage, the minimum voltage and the peak to peak voltage that were measured. Additionally, this part of the software was modified to only expect 5 transfers per measurement as well as adding the correct indexes for the Fast Fourier Transform and Samples plots, as shown in Figure 6.

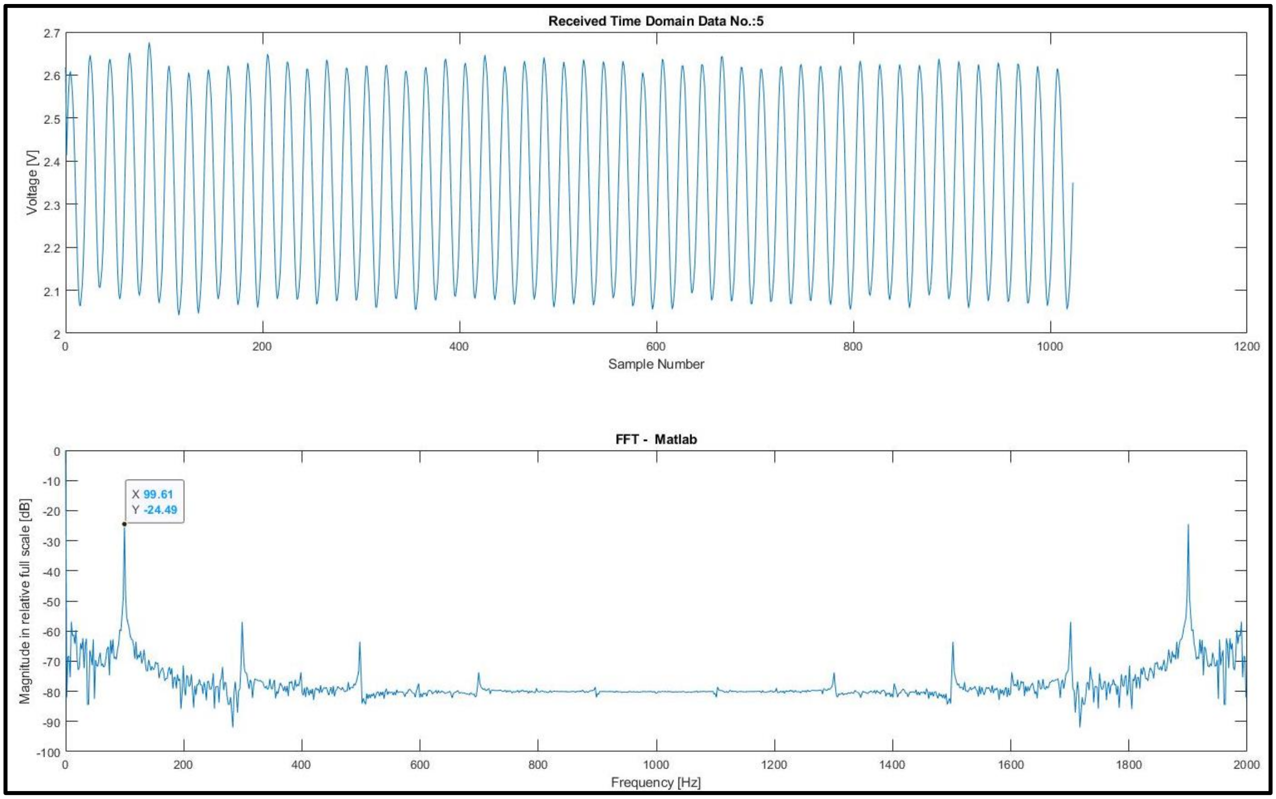


Figure 6 Received Time Domain No. 5 for 100Hz and FFT

Furthermore, the aforementioned program was used to measure (with the FreeSoc2's ADC input) the peak to peak voltages in 3 different stages of the circuit, labeled in Figure 1 as follows: VppDAC, VppOut1 and VppOut2. Additionally, VppIn after divider was calculated by using $V_{ppin\ after\ divider} = \frac{R_3}{R_3+R_2} V_{ppDAC}$, as the voltage was considered too small to actually be measured by the FreeSoc2's ADC.

Subsequently, by changing the DAC's frequency output (along with the ADC's sampling frequency), different frequencies were measured to be able to analyze the system's reaction to different inputs and its correct functionality. The desired frequencies were to be 10-equally-spaced values in a logarithmic scale. These values were calculated by using Matlab's *logspace* command as *logspace(0,3,10).* However, the output included the values 1, 2.2 and 4.6Hz. These values were changed to other values due to these values not being interesting in our frequency range. Therefore, after substituting these values and rounding the rest of the obtained values to the nearest 10's or 100's values, the frequencies 10, 20, 40, 50, 80, 100, 200, 500, 800, and 1000Hz were selected for testing.

After measuring the established voltages for the established frequencies, the gain in decibels was calculated relative to each stage of the amplification:

$$A_{1_{dB}} = 20 \times log_{10}(A_1);\ where\ A_1 = \frac{V_{ppOut1}}{V_{ppIn\ after\ Divider}}$$

$$A_{2_{dB}} = 20 \times log_{10}(A_2);\ where\ A_2 = \frac{V_{ppOut2}}{V_{ppOut1}}$$

$$A_{total_{dB}} = 20 \times log_{10}(A_{total});\ where\ A_{total} = A_2 \times A_1$$

This in turn produced the following table of results:

Table 1 Results for measurements.

frequency[Hz]	VppDAC[V]	VppIn(after Divider)[V]*	VppOut1[V]	A1*	A1[dB]	A1 LTSpice[db]	VppOut2[V]	A2*	A2[dB]	A2 LTSpice [dB]	Atotal*	Atotal[dB]	Atotal LTSpice [dB]
10	0.0236	0.002145455	0.1293	60.2669492	35.60158414	34.6	0.224	1.73240526	4.77298987	4.401	104.40678	40.374574	39.11
20	0.0242	0.0022	0.1878	85.3636364	38.62545814	37.7	0.3859	2.05484558	6.25558381	7.513	175.409091	44.881042	45.46
40	0.0245	0.002227273	0.2112	94.8244898	39.53841029	39.01	0.5367	2.54119318	8.10075363	8.839	240.967347	47.6391639	48.17
50	0.0235	0.002136364	0.2027	94.8808511	39.54357143	39.19	0.5771	2.84706463	9.08794651	9.035	270.131915	48.6315179	48.57
80	0.0241	0.002190909	0.2094	95.5767635	39.6070464	39.32	0.622	2.9703916	9.45627415	9.246	283.900415	49.0633205	48.93
100	0.0238	0.002163636	0.2098	96.9663866	39.73242424	39.3	0.6181	2.94613918	9.38506519	9.299	285.676471	49.1174894	48.97
200	0.023	0.002090909	0.1824	87.2347826	38.81379366	38.83	0.5433	2.97861842	9.48029742	9.367	259.83913	48.2940911	48.57
500	0.0224	0.002036364	0.1384	67.9642857	36.64561514	36.22	0.426	3.07803468	9.76547018	9.371	209.196429	46.4110853	45.97
800	0.0224	0.002036364	0.1044	51.2678571	34.19690331	33.64	0.3013	2.88601533	9.20597266	9.341	147.959821	43.402876	43.35
1000	0.0297	0.0027	0.0968	35.8518519	31.09023186	32.06	0.2428	2.50826446	7.9874665	9.31	89.9259259	39.0776984	41.74
						Divider:		*=calculated					
						R2[Ohm]	1000						
						R1[Ohm]	100						

The purple, green and cyan colored results on Table 1 (please refer to the attached "SDHD_3.xlsx", in case better visualization is needed), where then plotted on top of the professor's simulation plot, in order to graphically contrast the measured data against simulated data, as seen in Figure 7. The professor's simulated data can also be contrasted in Table 1 as they are shown in blue, yellow and orange. All the table's coloring matches Figure 7's coloring.

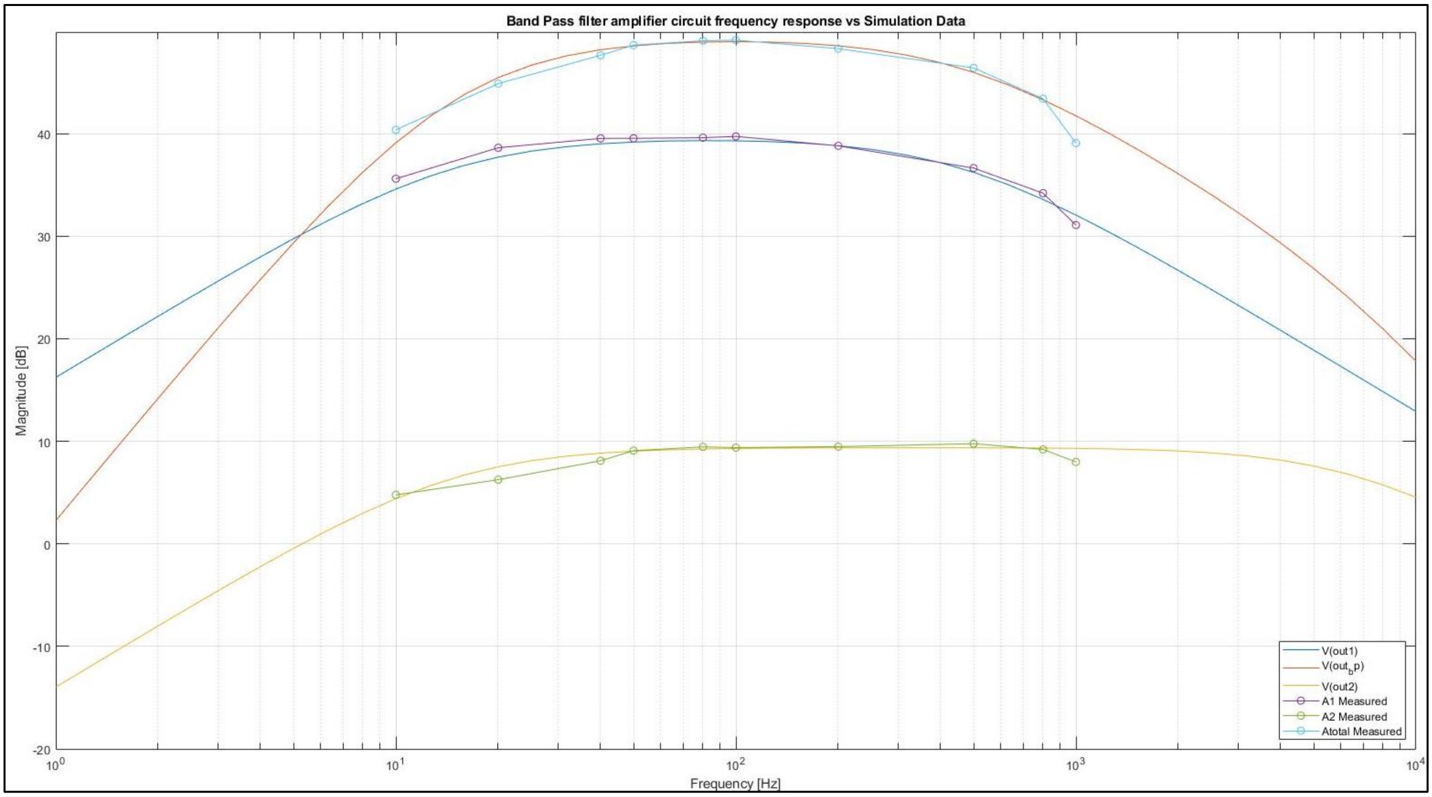


Figure 7 Band Pass filter amplifier circuit frequency response vs Simulation Data

3. Conclusion

The amplifier band pass, as seen in Figure 7 adjusts to the simulated data closely. This means that the testbench, and as such the developed circuit, behaves like the simulated data.

Additionally, since the amplifier was designed to have a 300 gain, and the measured values for 100Hz ranged (in several iterations that are not depicted in the table) within values between 280 and 305, it can be said that the amplifier did really amplify to the desired value.

This work, however, would be greatly benefitted from having an actual oscilloscope and a better multimeter to test the circuit on. Due to the corona situation, this project suffered from minor setbacks, such as not having desirable measurement equipment, but at the end, the result was properly achieved.

4. Outlook

Further work could be done by manufacturing the designed PCB board. Now that the hardware was tested, the actual PCB could be built, knowing that both the hardware implementation and the software part of this project work together as expected, all that is left is an actual product (PCB implementation).

Before soldering the components, some continuity tests could be ran on the PCB, to be able to know if the PCB was correctly produced. Afterwards, the components could be soldered y sections, testing different sections of the board little by little, making sure the components work as expected. This could be done by implementing a test plan of different parameter, just like the tests ran on this report.

Once the PCB was properly tested, still using the DAC plus the divider as an (outer) input, the radar could be plugged in (due to it being connected through a soldered socket, this input change would be relatively easy). Once it is tested correctly, now the radar product could be considered as done.