Audio Processing with Channel Filtering using DSP Techniques

Jean Jiang*
*Department of Engineering Technology
Purdue University Northwest
Westville, IN, USA
jjiang@pnw.edu

Abstract— **In this paper, a novel surround sound system operated by DSP boards is developed. Besides woofer(s) and tweeter(s), the audio system has been improved by adding additional speakers for handling mid-frequencies by designing a particular band-pass filter to process its mid-range. Meanwhile, three audio amplifiers are built to drive woofer (low frequency), mid-frequency speaker, and tweeter (high frequency), respectively. The practical audio amplifiers are also created with moderate gains in order to power speakers, in which the amplification circuits are modified using the data sheet of audio chip LM386 IC. Three different types of filters, which are low-pass, band-pass and high-pass, are also designed using the Remez exchange algorithm by MATLAB. Then, the obtained filter coefficients are loaded into two DSP boards (*TMS320C6713)* to process the sound signals in the full audio frequency range. Finally, the processed audio data are enhanced by audio amplifiers strong enough to power three sets of speakers: bass, mid-range, and tweeters. In addition to developing and testing the system, the research methodology and design is summarized and future improvements are discussed.**

Keywords — Audio amplifiers, optimal design method, digital signal processing (DSP) techniques, Remez exchange function, Parks-McClellan algorithm, TMS320C6713 DSP board, bass, mid-range speaker, tweeter.

I. INTRODUCTION

In audio systems, there is often a situation where the application requires the entire audible range of frequencies, but this is beyond the capability of any single speaker driver. Therefore, people often combine several drivers, such as the speaker cone and horns, each covering different frequency ranges, to reproduce the full audio frequency range [1-2, 4].

A typical two-band digital crossover can be designed as shown in Figure 1[1]. There are two speaker drivers. The woofer responds to low frequencies, and the tweeter responds to high frequencies. The incoming digital audio signal is split into two bands by using a low-pass filter and a high-pass filter in parallel. Then the separated audio signals are amplified and

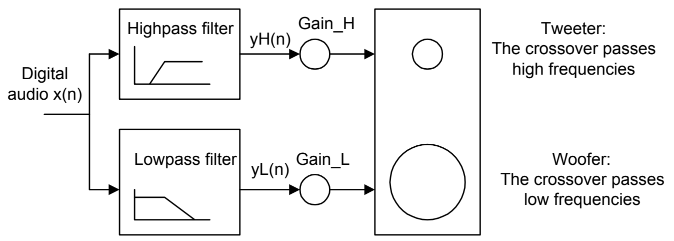

Digital audio x(n)
Highpass filter
Lowpass filter
yH(n)
yL(n)
Gain_H
Gain_L
Tweeter: The crossover passes high frequencies
Woofer: The crossover passes low frequencies

Fig. 1 General Block Diagram of a Sound System

send them to their respective corresponding speaker drivers. Hence, the objective is to design the low-pass filter and the high-pass filter so that their combined frequency response is flat, while keeping the transition as sharp as possible to prevent audio signal distortion in the transition frequency range. Although traditional crossover systems are designed using active circuits (analog filters) [7] or passive circuits/filters, the digital crossover system provides a cost-effective, high quality solution with programmable ability and flexibility.

In this paper, the audio system has been improved by adding an additional speaker for handling mid-frequencies by designing a particular band-pass filter to process its mid-range. Meanwhile, three audio amplifiers are needed to drive a woofer (low frequency), mid-frequency speaker, and a tweeter (high frequency), respectively.

We first create practical audio amplifiers with moderate gains in order to power speakers, in which the amplification circuits are modified using the data sheet of audio chip LM386 IC. Then, three different types of filters, which are low-pass, band-pass and high-pass, are designed using the Remez algorithm by MATLAB. We then load the filter coefficients into two DSP boards (*TMS320C6713)* [3, 5-6] to process the sound signal in the full audio frequency range. Finally, the processed audio data are enhanced by audio amplifiers strong enough to power three sets of speakers: bass, mid-range, and tweeters. Note that the low and band pass filters are designed around the sampling frequency of 24 kHz, and the DSP boards are set as the same sampling rate. However, the sampling

978-1-5386-4649-6/18/$31.00 ©2018 IEEE

frequency of 48 kHz is chosen for tweeters in order to offer frequency responses up to 24 kHz, since the human ear can detect up to 20 kHz.

II. AUDIO AMPLIFIERS

In this audio system, there are several sets of speakers which are all powered by audio amplifiers designed in this paper, and each group is divided into different channels. But firstly, an amplifier using a LM386 IC is designed to obtain a moderate gain for a weak input signal to power a speaker. A few parts are needed to construct such an audio amplifier, including a speaker, a LM386 amplifier chip, capacitors, and resistors.

The circuit originally built was an amplifier with a gain of 20, according to the data sheet. The circuit is shown in Figure 2.

The circuit works as expected; however, low-frequency signals below 1 kHz are of subpar quality. Therefore, we can encounter extremely distorted sounds which make the speaker sound as if it has been blown. This undesirable effect suggests that the circuit must be modified significantly. Therefore, we have redesigned the audio amplifier circuit as shown in Figure 3. The major improvement in the new audio amplifier is the addition of a first-order low-pass filter at the voltage source V_{cc}.

Finally, a 47 µF capacitor and a 15 V DC voltage source are selected after testing the new circuit. The modified and finely adjusted amplifier, which is shown in Figure 4, produces a distinctly cleaner sound than what was obtained from the original circuit. With this modification, the voltage source V_{cc} or V_1 provides enough power to the speaker to amplify low-frequency sounds, since bass frequencies draw more power to produce sound than higher frequencies. The gain produced by this circuit is about 9.3 while the original circuit based on the data sheet has a gain of about 20. Decreasing the gain is important because an input from an iPhone (or any other audio device) has a max voltage output of about 2 $V_{p\text{-}p}$, while a LM386 chip can handle a maximum of 15 V; therefore, the amplified signal should not to be greater than 15 $V_{p\text{-}p}$. Otherwise, clipping will occur, resulting in a distorted output signal. In addition, a higher resistance has been added between the input and the positive input-terminal on the LM386 chip. This modification produces a large enough voltage drop across resistor R_1 and guarantees a small enough voltage input such that signals at the amplifier output terminal will not be distorted. In summary, after making these adjustments, a single channel audio system can be built successfully by hooking up an audio input from a computer or an iPod, amplifying the signal, and listening to the output through speakers with minimal distortion.

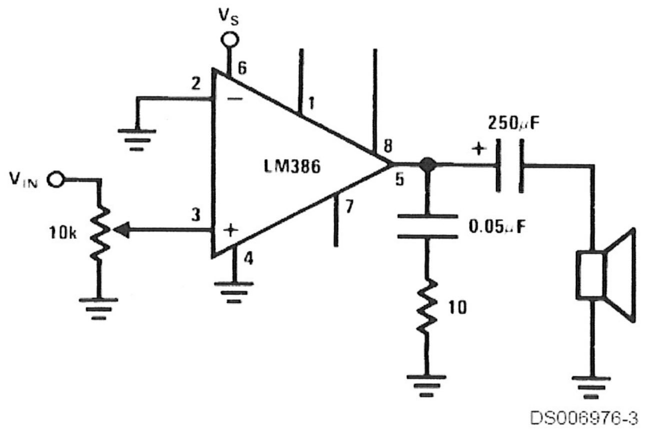


Fig. 2. Original Audio Amplifier with Minimum Parts

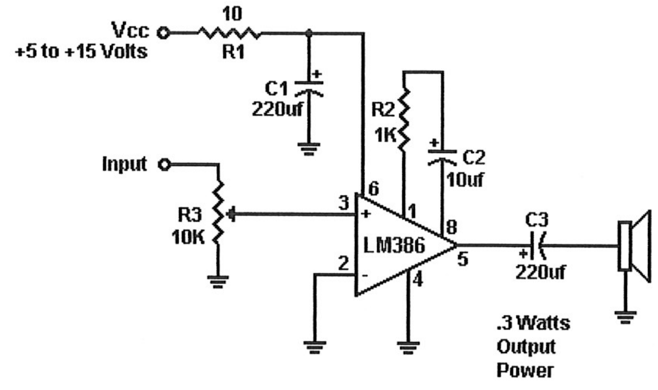


Fig.3. Improved Audio Amplifier Schematic

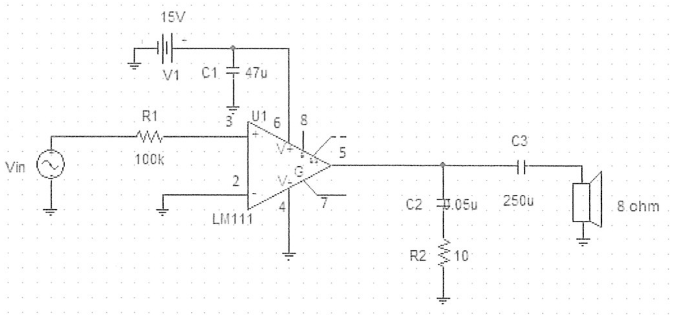


Fig. 4. Final Audio Amplifier Setup

III. FILTER DESIGN AND SOFTWARE CONSIDERATIONS

One of the project objectives is to filter audio signals out to separate sets of speakers. There are three sets of speakers: bass, mid-range, and tweeters; therefore, three filters are needed. One filter is for bass (low frequencies), one for mid-range frequencies, and one for tweeter (high frequencies). MATLAB is used to design such filters and to figure out sets of filter coefficients.

In this audio system, all three digital filters are chosen as FIR types with orders of 25, thereby obtaining an FIR low-pass with 25 taps, an FIR band-pass with 25 coefficients, and an FIR high-pass with 25 coefficients. A typical FIR filter has the standard format as:

$$y(n) = \sum_{i=0}^{K} b_i x(n-i) \tag{1}$$
$$= b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) + \cdots + b_K x(n-K)$$

where b_i are FIR filter coefficients and $K+1$ denotes the FIR filter length. Applying the Z-transform on both sides on Equation (1) yields:

$$Y(z) = b_0 X(z) + b_1 z^{-1} X(z) + \cdots + b_K z^{-K} X(z) \tag{2}$$

Factoring out $X(z)$ on the right hand side of Equation (2) and then dividing $X(z)$ on both sides, we have the transfer function, which depicts the FIR filter, as:

$$H(z) = \frac{Y(z)}{X(z)} = b_0 + b_1 z^{-1} + \cdots + b_K z^{-K}\,. \tag{3}$$

There are many algorithms available to design an FIR filter; for this system, the Parks-McClellan algorithm, is chosen for its efficiency and flexibility. The FIR filter design using the Parks-McClellan algorithm was developed based on the idea of minimizing the maximum approximation error in a Chebyshev polynomial approximation to the desired filter magnitude response. In particular, the Remez exchange algorithm is adopted in MATLAB in order to obtain the filter coefficients as follows:

A). A low-pass filter is designed with a cutoff frequency of 1000 Hz, and its coefficient array lpb[ ] created by MATLAB is listed below:

lpb[25] = -0.0232, 0.0271, 0.0304, 0.0398, 0.0524, 0.0669, 0.0820, 0.0969, 0.1107, 0.1223, 0.1312, 0.1368, 0.1387, 0.1368, 0.1312, 0.1223, 0.1107, 0.0969, 0.0820, 0.0669, 0.0524, 0.0398, 0.0304, 0.0271, -0.0232

The filter frequency responses are shown in Figure 5.

B). A band-pass filter is designed with a lower cutoff frequency of 1500 Hz and higher cutoff frequency of 8500 Hz, and its coefficient array bpb[ ] created by MATLAB is listed below:

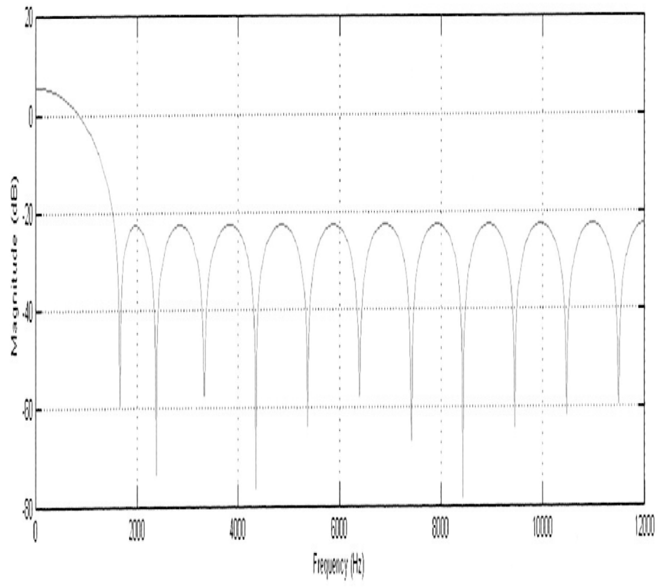
bpb[25] = 0.0112, 0.0113, -0.0514, -0.0513, 0.0077, -0.1813, 0.0718, -0.2446, -0.0269, -0.0938, -0.2914, 0.1251, 0.5579, 0.1251, -0.2914, -0.0938, -0.0269, -0.2446, 0.0718, -0.1813, 0.0077, -0.0513, -0.0514, 0.0113, 0.0012

The filter frequency responses are shown in Figure 6.

C). A high-pass filter is designed with a cutoff frequency of 9000 Hz, and its coefficient array hpb[ ] created by MATLAB is listed below:

hpb[25] = 0.0808, 0.0664, 0.0220, -0.0674, -0.1521, -0.1722, -0.1074, -0.0029, 0.0546, 0.0006, -0.1528, -0.3176, 0.6120, -0.3176, -0.1528, 0.0006, 0.0546, -0.0029, -0.1074, -0.1722, -0.1521, -0.0674, 0.0220, 0.0664, 0.0808

The filter frequency responses are shown in Figure 7.



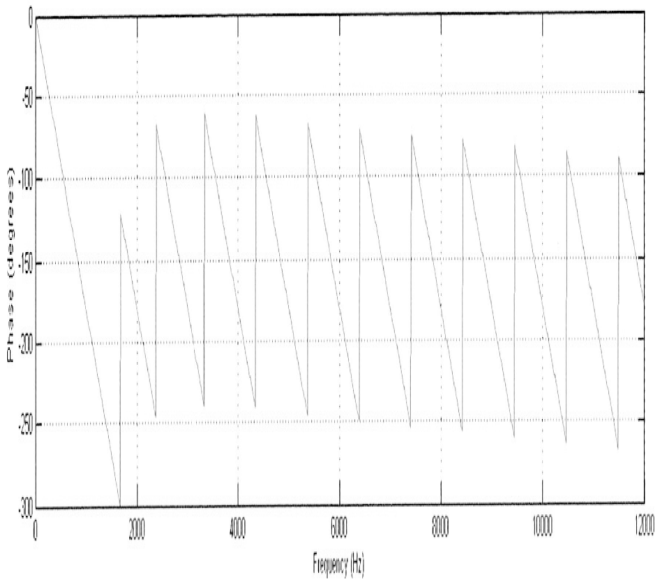


Fig. 5 Frequency Responses of Low-pass Filter

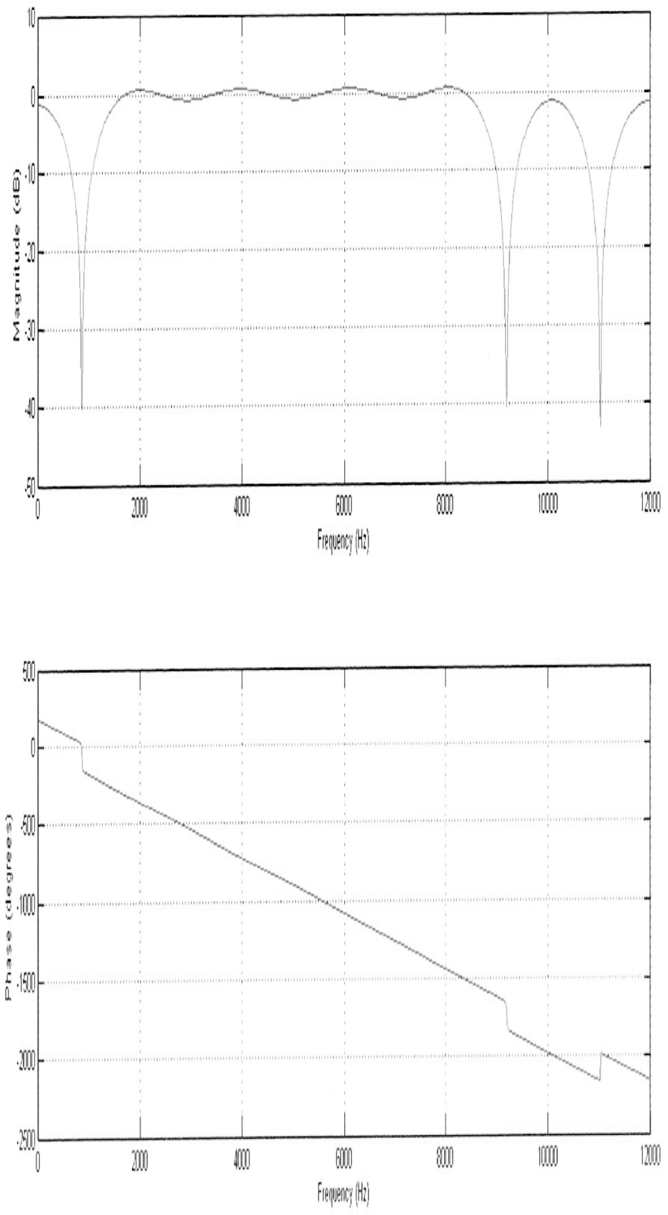


Fig. 6 Frequency Responses of Band-pass Filter

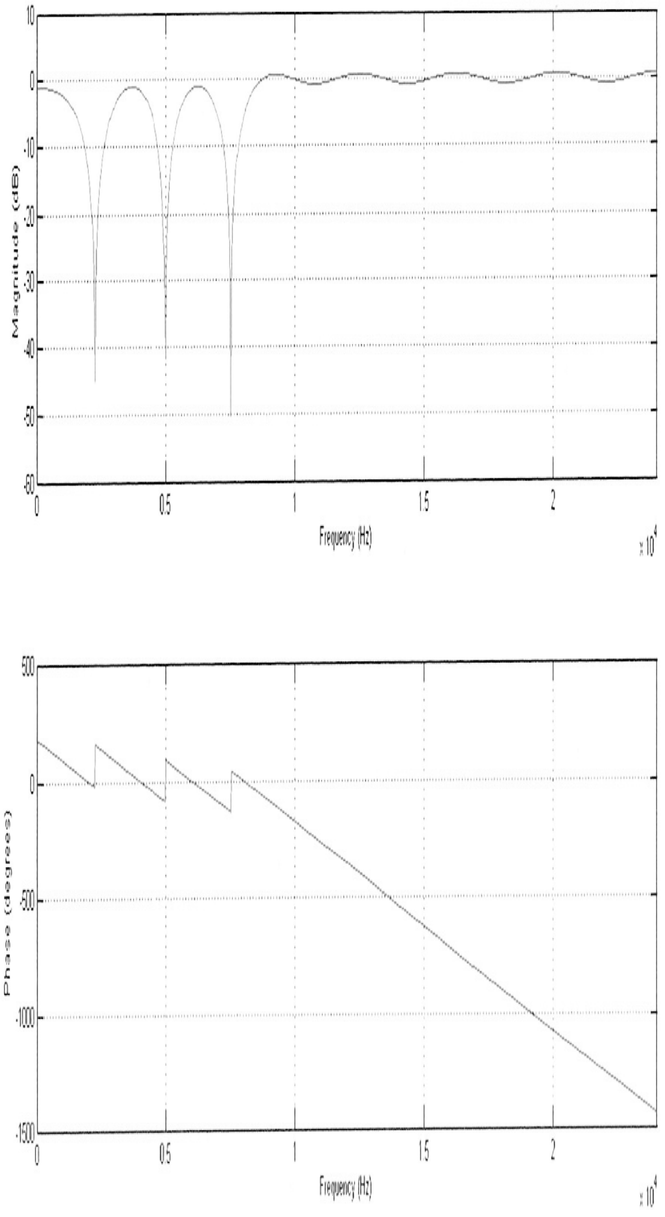


Fig. 7 Frequency Responses of High-pass Filter

D). DSP board applications:

A TI TMS320C6713 DSP board offers only one line out, in which there are one ground pin, left channel and right channel. This would only offer two channels to implement the system; however, three different types of filters are needed for the audio processing. Therefore, at least two DSP boards must be used to push three filters through to different speakers.

The software to program DSP boards is written in C and the code segment is shown in Fig. 8.

```
//C6713dskinit.c Includes functions from TI in the C6713 CSL and
C6713DSK BSL

#include "C6713dskinit.h"
#define using_bios;

Uint32 fs=DSK6713_AIC23_FREQ_48KHZ;                          //Set Sampling
Frequency (predefined in dskc6713_aic23.h file)

union {Uint32 combo; short channel[2];} AIC23_data;         //??


//#define using_bios                                         //if BIOS don't use top
of vector table
extern Uint32 fs;                                            //for sampling frequency

void c6713_dsk_init()                                        //dsp-peripheral
initialization
{
DSK6713_init();                                              //call BSL to init DSK-
EMIF,PLL)

hAIC23_handle=DSK6713_AIC23_openCodec(0, &config);          //handle(pointer) to
codec
DSK6713_AIC23_setFreq(hAIC23_handle, fs);                   //set sample rate
MCBSP_config(DSK6713_AIC23_DATAHANDLE,&AIC23CfgData);       //interface 32 bits
toAIC23

MCBSP_start(DSK6713_AIC23_DATAHANDLE, MCBSP_XMIT_START | MCBSP_RCV_START |
        MCBSP_SRGR_START | MCBSP_SRGR_FRAMESYNC, 220);      //start data channel
again
}

void comm_poll()                                             //added for
communication/init using polling
{
        poll=1;                                              //1 if using polling
   c6713_dsk_init();                                         //init DSP and codec
}

void comm_intr()                                             //for communication/init
using interrupt
{
        poll=0;                                              //0 since not polling
   IRQ_globalDisable();                                      //disable interrupts
        c6713_dsk_init();                                    //init DSP and codec
        CODECEventId=MCBSP_getXmtEventId(DSK6713_AIC23_codecdatahandle);//McBSP1
Xmit

#ifndef using_bios                                           //do not need to point
to vector table
        IRQ_setVecs(vectors);                                //point to the IRQ
vector table


  #endif                                                     //since interrupt vector
  handles this

        IRQ_map(CODECEventId, 11);                           //map McBSP1 Xmit to
  INT11
        IRQ_reset(CODECEventId);                             //reset codec INT 11
     IRQ_globalEnable();                                     //globally enable
  interrupts
        IRQ_nmiEnable();                                     //enable NMI interrupt
     IRQ_enable(CODECEventId);                               //enable CODEC eventXmit
  INT11

        output_sample(0);                                    //start McBSP interrupt
  outputting a sample
  }

  void output_sample(int out_data)                           //for out to Left and
  Right channels
  {
        short CHANNEL_data;

        AIC_data.uint=0;                                     //clear data structure
        AIC_data.uint=out_data;                              //32-bit data -->data
  structure

  //The existing interface defaults to right channel. To default instead to the
  //left channel and use output_sample(short), left and right channels are swapped
  //In main source program use LEFT 0 and RIGHT 1 (opposite of what is used here)
        CHANNEL_data=AIC_data.channel[RIGHT];                //swap left and right
  channels
        AIC_data.channel[RIGHT]=AIC_data.channel[LEFT];
        AIC_data.channel[LEFT]=CHANNEL_data;
     if (poll) while(!MCBSP_xrdy(DSK6713_AIC23_DATAHANDLE)); //if ready to transmit
                MCBSP_write(DSK6713_AIC23_DATAHANDLE,AIC_data.uint);//write/output
  data
  }

  void output_left_sample(short out_data)                    //for output from left
  channel
  {
        AIC_data.uint=0;                                     //clear data structure
        AIC_data.channel[LEFT]=out_data;                     //data from Left channel
  -->data structure

        if (poll) while(!MCBSP_xrdy(DSK6713_AIC23_DATAHANDLE));//if ready to
  transmit
                MCBSP_write(DSK6713_AIC23_DATAHANDLE,AIC_data.uint);//output left
  channel
  }

  void output_right_sample(short out_data)                   //for output from right
  channel
  {
        AIC_data.uint=0;                                     //clear data structure
        AIC_data.channel[RIGHT]=out_data; //data from Right channel -->data
  structure

 :
```

Fig. 8 DSP Board Code Segment

When setting up two DSP boards, we could set up each line into the signal source (a computer, an iPod, etc.) and run the boards. From their line-out, these boards then transfer the manipulated or filtered audio signals to their respective speakers, and the final result sounds satisfactory.

The overall system block diagram is shown in Figure 9.

IV. Results and Discussion

For this audio system development, our prototype offers good quality of sound by using three different filters for bass, mid-range frequency, and high frequency in two DSP boards. The system set up is shown in Figure 10.

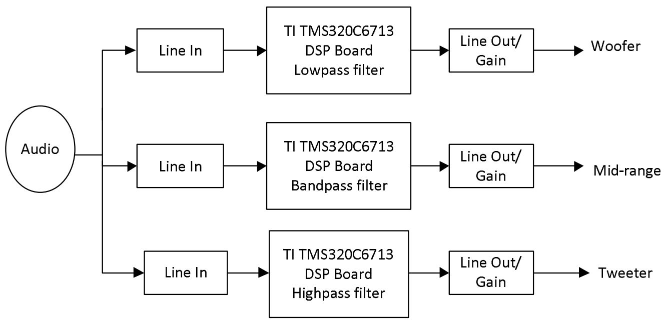

Audio
Line In
TI TMS320C6713 DSP Board Lowpass filter
Line Out/ Gain
Woofer
Line In
TI TMS320C6713 DSP Board Bandpass filter
Line Out/ Gain
Mid-range
Line In
TI TMS320C6713 DSP Board Highpass filter
Line Out/ Gain
Tweeter

Fig. 9 Sound System Block Diagram

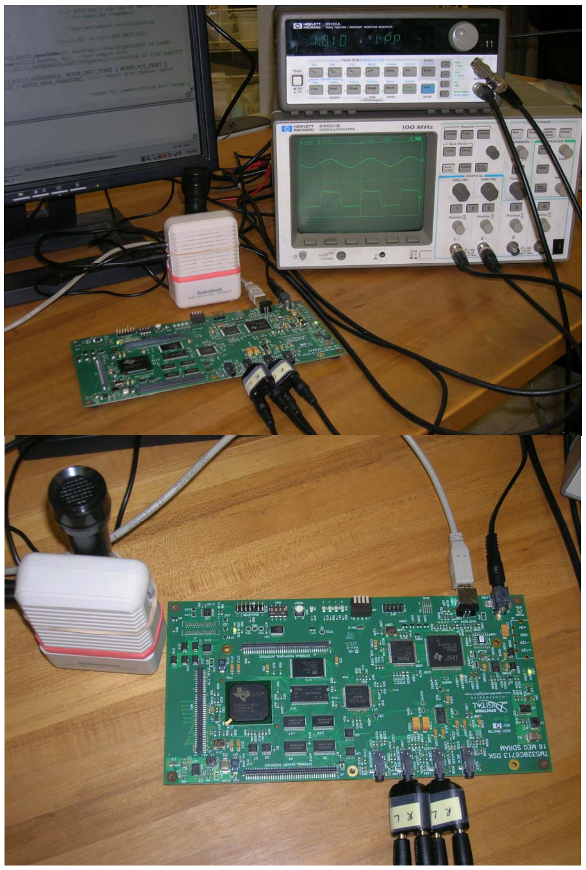


Fig. 10 Audio System with DSP Board Setup

We intend to eventually create a larger amplifier to drive higher quality speakers and therefore design something that, for example, could be placed in living room for a home theater experience. The LM3886 chip could be a good candidate for a dual voltage input of up to 60 volts. More development, testing, and sophisticated parts are needed to obtain a surround effect. In addition, we could mount the system in a more convenient way and arrange the boards to make the system's appearance more appealing for commercial use.

V. Conclusions

In this paper, an advanced sound system operated by DSP boards with 3 different types of filters – a low-pass, a band-pass and a high pass filter – is developed. The system consists of three improved, high quality audio amplifiers, two DSP boards, and three set of speakers. The system can produce good quality of sound in different frequency ranges. The DSP boards are therefore successfully used in audio filtering and are shown to offer high quality of sound. Future improvements include creating more powerful amplifiers, more sound channels, and giving a surround effect.

References

[1] L. Tan, J. Jiang, Digital Signal Processing: Fundamentals and Applications, 2nd Edition, Elsevier, 2013.

[2] J. Jiang, L. Tan, "Teaching Adaptive Filters and Applications in Electrical and Computer Engineering Technology Program," *2012 Proceedings of the American Society for Engineering Education*, San Antonio, Texas, June 2012..

[3] N. Kehtaranavaz, and B., Simsek, C6x-Based Digital Signal Processing, Prentice Hall, Upper Saddle River, New Jersey 07458, 2000.

[4] L. Tan, J. Jiang, "Teaching Advanced Digital Signal Processing with Multimedia Applications in Engineering Technology Programs," 2009 Proceedings of the American Society for Engineering Education, Austin, Texas, June 2009.

[5] Texas Instruments, TMS320C6x CPU and Instruction Set Reference Guide, Literature ID# SPRU 189C, Texas Instruments, Dallas, Texas, 1998.

[6] Texas Instruments, Code Composer Studio: Getting Started Guide, Texas Instruments, Dallas, Texas, 2001.

[7] L. Tan, J. Jiang, Analog Signal Processing and Filter Design, 2nd Edition, Linus Publications, 2016.