121st ASEE Annual Conference & Exposition
Indianapolis, IN
June 15-18, 2014

Paper ID #10325

Teaching Digital Signal Processing Course with a Real-Time Digital Crossover System for Electrical and Computer Engineering Technology Students

Prof. Jean Jiang, Purdue University, North Central

DR. JEAN JIANG is currently with the College of Engineering and Technology at Purdue University North Central in Westville, Indiana. She received her Ph.D. degree in Electrical Engineering from the University of New Mexico in 1992. Before she joined Purdue University in 2009 as a faculty member, Dr. Jiang had taught analog signal processing, digital signal processing, and control systems for a number of years as a professor at DeVry University in Atlanta, Georgia. Dr. Jiang is a member of the Institute of Electronic and Electronic Engineers (IEEE). Her principal technical areas are in digital signal processing, adaptive signal processing, and control systems. She has published a number of papers in these areas. She has co-authored three textbooks: Digital Signal Processing: Fundamentals and Applications, Elsevier/Academic Press, 2nd edition, 2013, Fundamentals of Analog and Digital Signal Processing, Second Edition, AuthorHouse, 2008 and Analog Signal Processing and Filter Design, Linus Publications, 2009.

Teaching Digital Signal Processing Course with a Real Time Digital Crossover System for Electrical and Computer Engineering Technology Students

Abstract

This paper presents pedagogies and experiences for teaching digital signal processing (DSP) with real-time processing experiences in the electrical and computer engineering technology (ECET) program using a digital crossover system. In our ECET curriculum, the engineering technology students often face their challenging in studying the signal processing related courses due to the use of heavy mathematics. Although adopting MATLAB as an analysis and simulation tool is a great help, the engineering technology students are especially motivated by their real-time processing learning experiences. To accommodate such a learning environment, our DSP course with laboratories has been improved, in which each lab concurrently requires MATLAB simulations as well as real-time processing coding with a digital signal processor such as Texas Instruments' TMS320C6713 DSK.

This DSP course is the second signal processing course in electrical and computer engineering technology (ECET) program. The pre-requisite includes student working knowledge and skills of Laplace transform, Fourier series, Fourier transform, and different types of analog active filter design. After completing the course, students not only become familiar with MATLAB software development tools, but also gain the real-time analog signal processing experience. In this class, they learn to design digital low-pass, high-pass and band-pass filters and then program them using MATLAB and C, and apply the software and hardware interface for real speech applications such as in a crossover system. Comparing with the traditional DSP course which mainly focus on heavy mathematical development in sampling and recovering, spectrum analysis, FIR or IIR filter design with the limited computer simulations, our real-time crossover project allows students easily understand how a speech signal or music being processed and separated by a bass (woofer) and a tweeter after they design the proper low-pass and high-pass filters and apply two DSP boards (TI TMS320C6713 DSK). Therefore, in this paper, we first present a complete digital crossover platform, MATLAB design and simulations, developed C code inserted in both of DSP boards, crossover outputs and test results. Secondly, we address the outcomes of students learning achievement including continuous applications of their acquired DSP skills in other related courses and their motivations for continuing to pursue the upper-level signal processing related courses such as the advanced digital signal processing with multimedia applications. More importantly, we examine the course assessment according to analysis of the collected data from students' learning outcome survey, project evaluation, and further address the possible improvements of the course content based on our assessment.

I. Introduction

The application of real-time digital filtering technology has been found widely in modern electronic products, communication and control systems, computer peripherals, and multimedia devices[1,4-7]. In the area of digital signal processing (DSP) education for the engineering technology curriculum, the real-time DSP implementations are considered to be advanced topics to be covered. The trend of using real-time DSP filters in the industry has generated an increasing demand for engineering technology graduates with this particular working knowledge and skills. Many engineering technology programs have already offered a standard initial signal processing course in their undergraduate curricula, particularly in the electrical and computer engineering technology (ECET) curriculum. The first signal processing course generally covers basic techniques such as common analog functions, Laplace transform, Fourier series, Fourier transform, and different types of analog active filter design. Students in the DSP course apply these established skills for designing, implementing, and verifying various applications such as the digital crossover audio systems, speech signal processing, and so on. Specifically, the skills of the digital FIR filter design and signal spectral analysis are necessary for low-pass, band-pass and high-pass filter designs and verifications in the area of audio processing. Furthermore, a grasp of concepts and principles of the first signal processing course indicates the gained knowledge of the analog signal processing (ASP) course, in which three major topics, the Laplace transform, Fourier analysis, and analog filters, are covered. For example, the Laplace transform and Laplace transfer function serve for the analog filter design and digital IIR filter design, while the Fourier analysis supports for spectral analysis and digital FIR filter design with window functions. Hence, the prerequisite of the first signal processing course implies the ASP course.

Teaching digital filters to ECET students in the second DSP course appears challenging because it requires applying relatively advanced mathematics with the optimization theory and s-domain to z-domain conversions. However, with our teaching pedagogy, this barrier can be overcome through design of different types of filters by MATLAB and applying them to real-time implementations. MATLAB is a necessary tool used to verify filter performance and perform simulations of various digital filter applications. To motivate our technology students oriented about hands-on experience, we required them to perform real-time DSP using a floating-point digital signal processor[2-3], TMS320C6713 DSK (development starter kit), to develop a real-time DSP project such as a digital crossover system.

The paper is organized as follows. We first explain the course prerequisites and describe our class content, and then we introduce real-time signal processing hands-on project using DSP boards (TMS320C6713 DSK) and simulation tools such as MATLAB and MultiSIM. We will also present the course assessment and outcome, which include how the students apply their gained DSP knowledge to their capstone senior projects. Finally, we will address possible improvement of the course content and associated laboratories.

II. Course Prerequisite Requirements

In this section, we explain the course pre-requisites, which can be divided into three categories, as described below.

A. Digital Signal Processing Course Requirement

The first signal processing course covering the key topics of analog signal processing, such as common analog functions, Laplace transform, Fourier series, Fourier transform, and different types of analog active filter design. Students in the DSP course apply these established skills for designing, implementing, and verifying various applications such as the digital crossover audio systems, speech signal processing, and so on. Specifically, the skills of the digital FIR filter design and signal spectral analysis are necessary for low-pass and band-pass filter designs and verifications in the area of speech processing. Furthermore, a grasp of concepts and principles of the first signal processing course indicates the gained knowledge of the analog signal processing (ASP) course, in which three major topics, the Laplace transform, Fourier analysis, and analog filters, are covered. For example, the Laplace transform and Laplace transfer function serve for the analog filter design and digital IIR filter design, while the Fourier analysis supports for spectral analysis and digital FIR filter design with window functions. Hence, the prerequisite of the first signal processing course implies the ASP course.

B. Math Requirement

While satisfying the prerequisite of the signal processing course, students are gaining maturity in the comprehension and application of math including basic calculus, and proficiency in using algebra. A firm grasp of calculus concepts is also beneficial in understanding the advanced course materials such as the employment of the derivative operation or difference equations. Since the calculus course is a prerequisite for the first DSP course or the combined ASP and DSP course, it is not necessary that we list it as an additional prerequisite.

C. Software Requirement

To design, analyze, and simulate the DSP algorithms, MATLAB programming is required; this requirement was enforced in the previous signal processing course. In addition, MultiSIM can be used to verify different filter design.

As a summary, the DSP course needs the prerequisites as listed below:

1. Analog signal processing
2. MATLAB programming and MultiSIM simulation.

III. Course Content and the Associated Real-Time Project

We have divided the course content into two portions. First, the DSP fundamentals were covered, such as the sampling theorem, the z-transform and z-transfer functions, the discrete Fourier Transform(DFT), FFT algorithm, signal spectrum analysis, filter frequency responses, and filter implementations using the direct-form I and direct-form II, and so on. Seven (7) weeks were spent on teaching DFT, signal spectrum, transfer functions and frequency responses, which are considered as tough frequency domain analysis, since technology students usually have hard time to picture the signal spectra and frequency responses for different types of digital filters. That was also the main reason a real-time DSP project was introduced parallelly, which let students experience the DSP math could be implemented and displayed in the real world. The second portion introduced FIR design and IIR design with an emphasis on real-time digital filter implementation and applications. We focus on the real-time DSP audio project in this paper.

The course was taught in 16 weeks with 3 lecture hours and 3 laboratory hours per week. The textbook selected was "Digital Signal Processing: Fundamentals and Applications." [1] published by Elsevier, 2008. The textbook presents course materials at an appropriate math level, uses an ample amount of simplified and clearly worked examples, adopts MATLAB programs to demonstrate simulations, and provides application examples to motivate students. Simplification of real-time DSP implementations to the engineering technology level is a plus.
To minimize the time for learning different simulation tools, we simply selected MATLAB, which was familiarized by students when they took the first signal processing course as a major simulation and design tool. However, other simulation tools were also welcomed when time was permitted.

The DSP boards (TMS320C6713 DSK) and MATLAB were chosen as a platform for teaching real-time signal processing and students were asked to construct a digital crossover system in this course. A typical two-band digital crossover can be designed as shown in Figure 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 lowpass filter and a highpass filter in parallel. Then the separated audio signals can be amplified and sent to their respective corresponding speaker drivers.

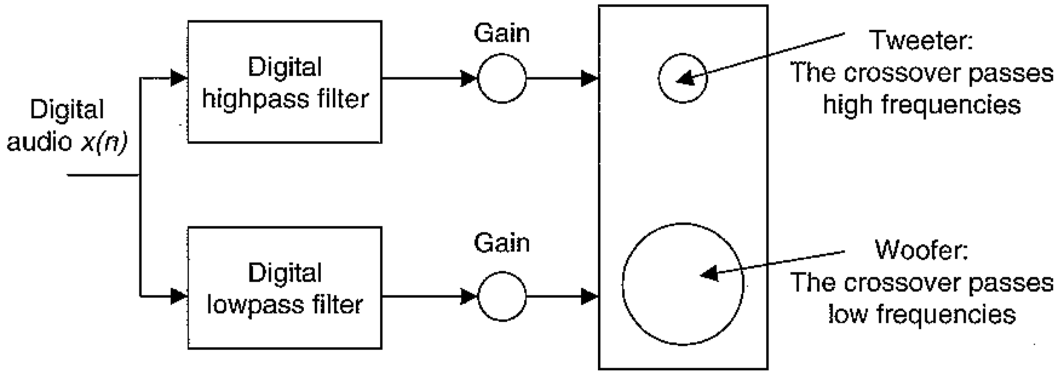
Digital audio x(n)
Digital highpass filter
Digital lowpass filter
Gain
Gain
Tweeter: The crossover passes high frequencies
Woofer: The crossover passes low frequencies

Figure 1 Two-band Digital Crossover.

Real-Time Digital System Implementations

The project consists of two portions: MATLAB simulation and real-time DSP implementation with DSP boards.

A. MATLAB Simulation

The MATLAB simulation must be completed prior to the real-time implementation. Its objective is to design a lowpass filter and a highpass filter so that their combined frequency response is flat, while keeping transitions as sharp as possible to prevent audio signal distortion in the transition frequency range. Both lowpass and highpass filter include a cutoff frequency of 1,500

Hz with 57 taps. The different frequency responses for lowpass filtering or highpass filtering are shown in Figure 2.

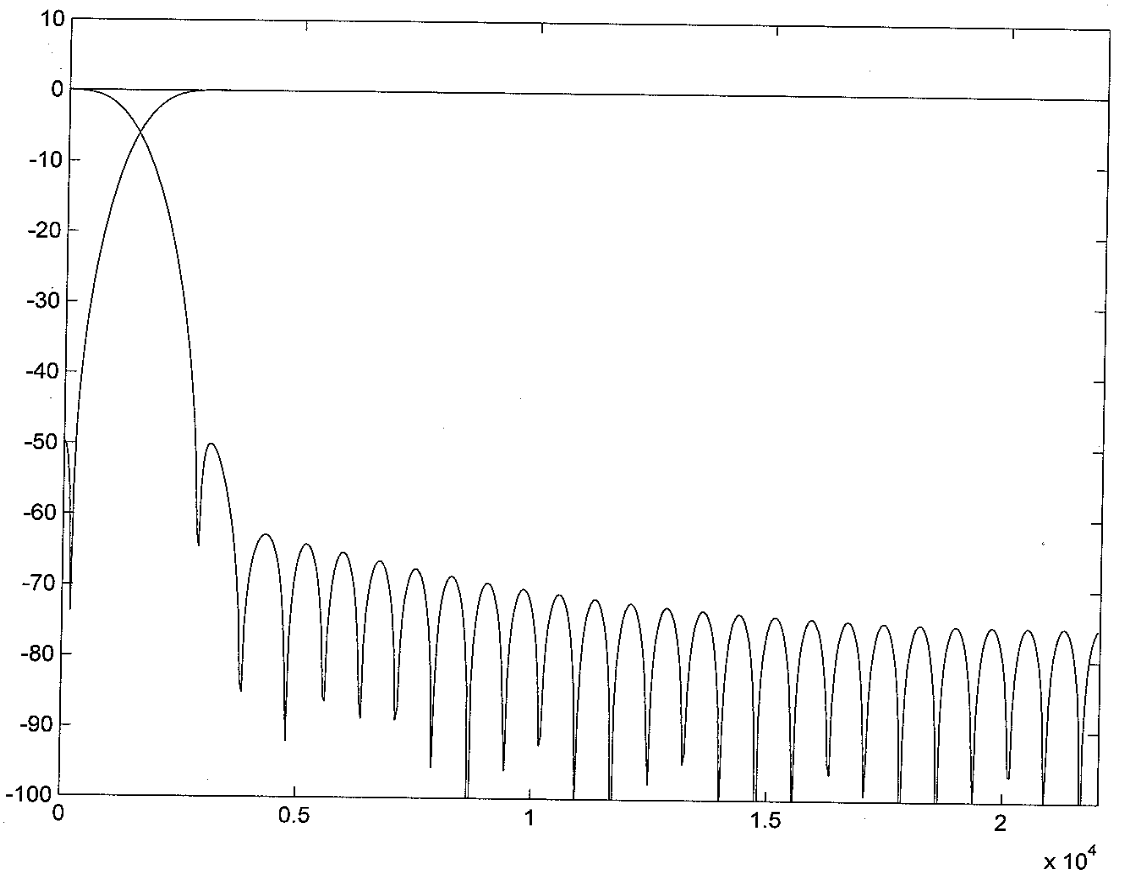

Figure 2a The combined magnitude response for the digital crossover.

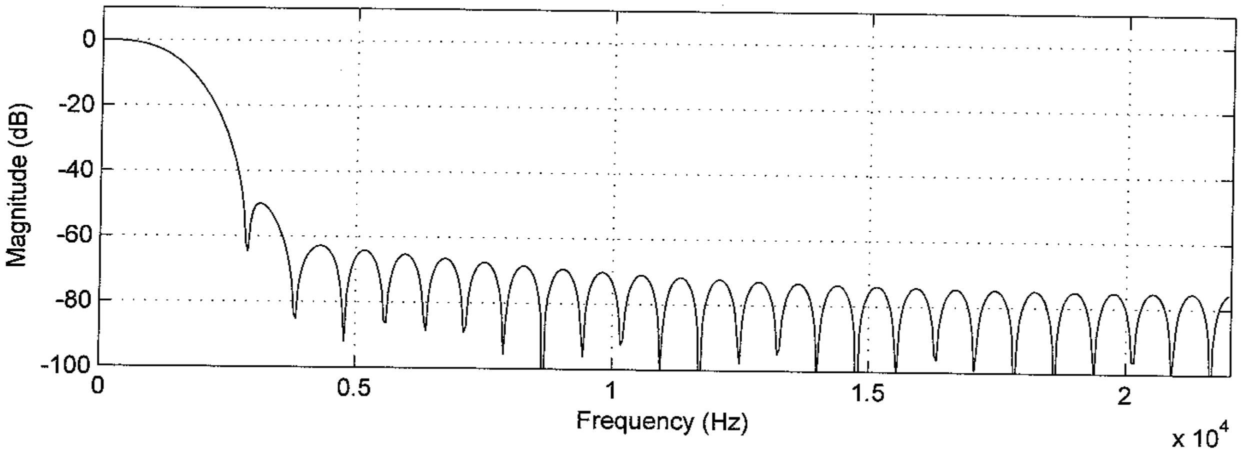

Figure 2b Magnitude response for lowpass filter.

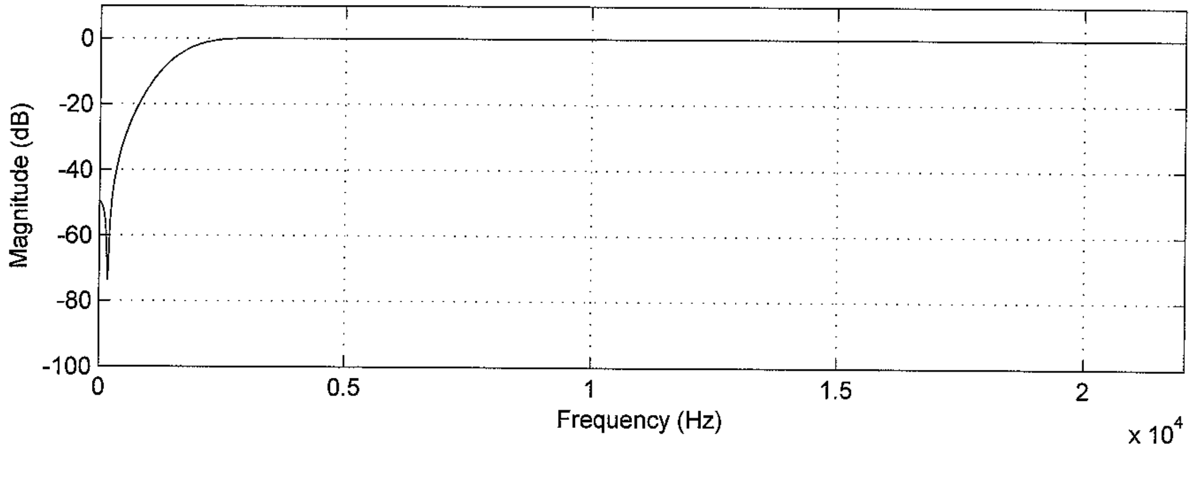

Figure 2c Magnitude response for highpass filter.

B. Real-time Laboratory Implementation

We first introduce a tutorial to verify the input and output signals in the system to establish the students' working knowledge. The block diagram of a real-time DSP system, its single DSP board setup and program segment for verifying input and output signals are shown in Figure 3a, Figure 3b and Figure 3c, respectively, where the sampling rate is 8,000 samples per second.

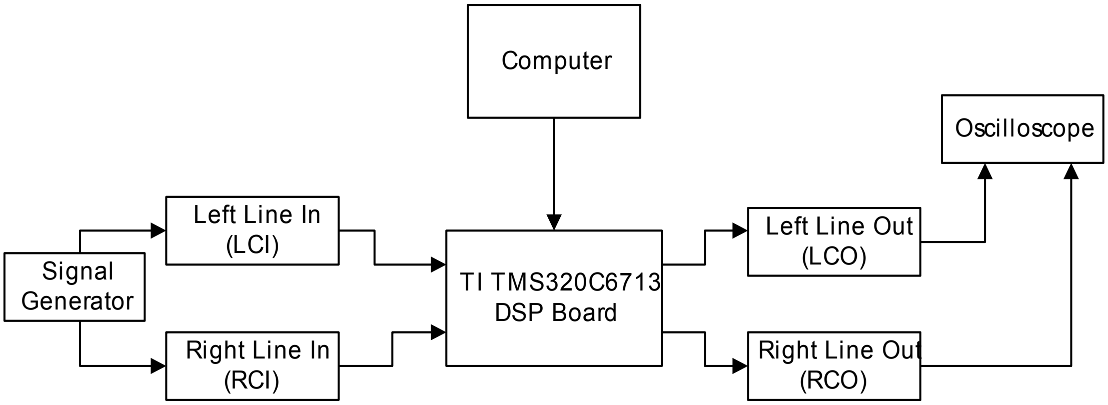

Figure 3a Block diagram of the DSP system.

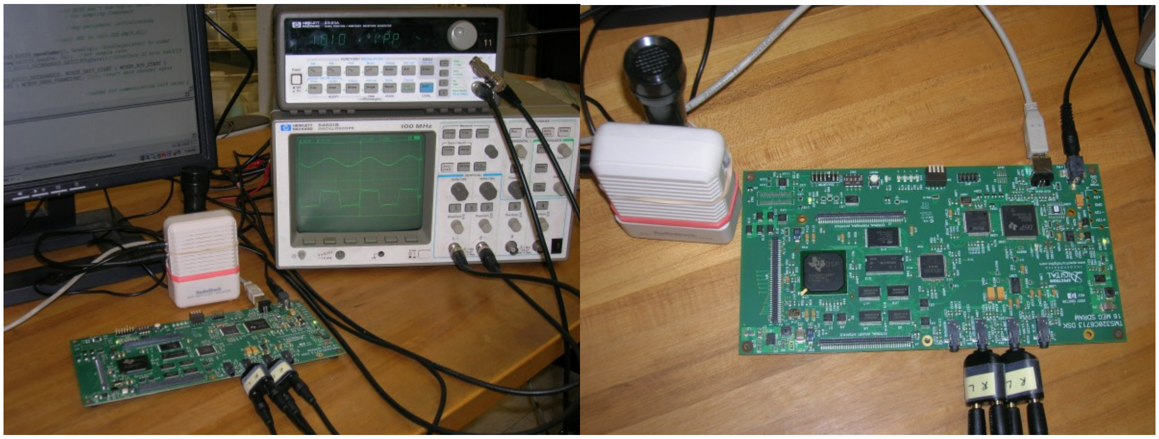

Figure 3b Real-time DSP laboratory setup.

```
float xL[1]={0.0};
float xR[1]={0.0};
float yL[1]={0.0};
float yR[1]={0.0};
interrupt void c_int11()
{
     float lc; /*left channel input */
     float rc; /*right channel input */
     float lcnew; /*left channel output */
     float rcnew; /*right channel output */
     int i;
//Left channel and right channel inputs
   AIC23_data.combo=input_sample();
    lc=(float) (AIC23_data.channel[LEFT]);
    rc= (float) (AIC23_data.channel[RIGHT]);
// Insert DSP algorithm below
    xL[0]= lc;  /*Input from the left channel */
    xR[0]= rc;  /*Input from the right channel */
    yL[0]=xL[0]; /* simplest DSP equation for the left channel */
    yR[0]=xR[0]; /* simplest DSP equation for the right channel */
// End of the DSP algorithm
     lcnew=yL[0];
     rcnew=yR[0];
     AIC23_data.channel[LEFT]=(short) lcnew;
     AIC23_data.channel[RIGHT]=(short) rcnew;
     output_sample(AIC23_data.combo);
}
```

Figure 3c. Program segment for verifying input and output.

Then a crossover DSP project was assigned. Students were divided into groups to develop the real-time projects and to generate their design reports. During the project's developing phase, students obtained the advice from their instructor and worked under the supervision of their instructor.

Figure 4a shows a digital crossover system that uses two TI DSP boards. One DSP board is used for the lowpass filter and the other for the highpass filter. Using the designed digital lowpass filter, one TI DSK produces low frequency components to drive a woofer. Meanwhile, implementing the designed digital highpass filter on the other TI DSK generates high frequency components to drive a tweeter. A set of speakers are also connected to each board and the music is heard through both set of speakers. The output of the speakers is dependent on the connected

DSP boards. Finally, the sample program segments for the lowpass filtering and the highpass filtering are given in Figure 4b and 4c, respectively.

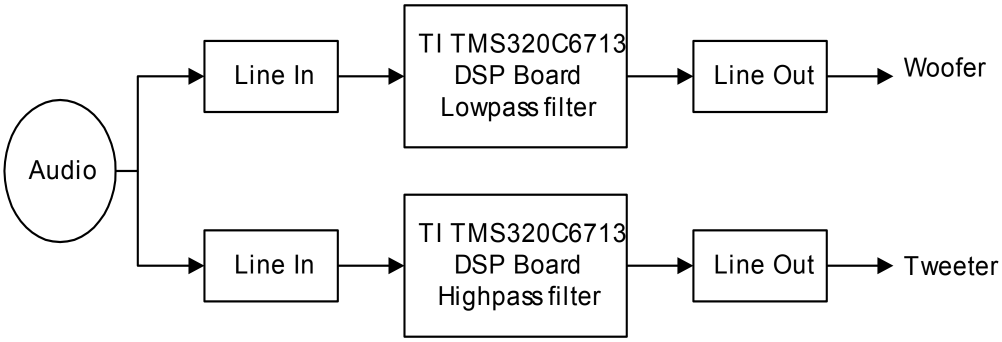
Audio
Line In
TI TMS320C6713 DSP Board Lowpass filter
Line Out
Woofer
Line In
TI TMS320C6713 DSP Board Highpass filter
Line Out
Tweeter

Figure 4a Laboratory setup for a digital crossover system.

```
Uint32 input_sample()                              //for 32-bit input
{
    short CHANNEL_data;

    if (poll) while(!MCBSP_rrdy(DSK6713_AIC23_DATAHANDLE));//if ready to receive
         AIC_data.uint=MCBSP_read(DSK6713_AIC23_DATAHANDLE);//read data

//Swapping left and right channels (see comments in output_sample())
    CHANNEL_data=AIC_data.channel[RIGHT];          //swap left and right channel
    AIC_data.channel[RIGHT]=AIC_data.channel[LEFT];
    AIC_data.channel[LEFT]=CHANNEL_data;

    return(AIC_data.uint);
}

short input_left_sample()                              //input to left channel
{
    if (poll) while(!MCBSP_rrdy(DSK6713_AIC23_DATAHANDLE));//if ready to receive
     AIC_data.uint=MCBSP_read(DSK6713_AIC23_DATAHANDLE);//read into left channel
    return(AIC_data.channel[LEFT]);                    //return left channel data
}

short input_right_sample()                             //input to right channel
{
    if (poll) while(!MCBSP_rrdy(DSK6713_AIC23_DATAHANDLE));//if ready to receive
     AIC_data.uint=MCBSP_read(DSK6713_AIC23_DATAHANDLE);//read into right channel
    return(AIC_data.channel[RIGHT]);                   //return right channel data
}

// DSP coefficients, variables, and constants
float yr[1]={0};
float xr[57]={0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0};
float b[57]={-0.000268067102911,  -0.000479548550466,  -0.000744481079920,
-0.001088262533223,  -0.001522694437199,  -0.002040166137739,
-0.002609377429647,  -0.003173202897853,  -0.003649126603227,
-0.003932450753003,  -0.003902226040124,  -0.003429587413324,
-0.002387931620007,  -0.000664165631433,   0.001829891318777,
0.005146939224362,   0.009295433763665,   0.014233496388969,
0.019866183241039,   0.026046547745487,   0.032580634719737,
0.039236224904193,   0.045754835299434,   0.051866199863290,
0.057304231913689,   0.061823324243860,   0.065213789202115,
0.067315284712047,   0.068027210884354,   0.067315284712047,
0.065213789202115,   0.061823324243860,   0.057304231913689,
0.051866199863290,   0.045754835299434,   0.039236224904193,
0.032580634719737,   0.026046547745487,   0.019866183241039,
 0.014233496388969,   0.009295433763665,   0.005146939224362,
 0.001829891318777,  -0.000664165631433,  -0.002387931620007,
 -0.003429587413324,  -0.003902226040124,  -0.003932450753003,
 -0.003649126603227,  -0.003173202897853,  -0.002609377429647,
 -0.002040166137739,  -0.001522694437199,  -0.001088262533223,
 -0.000744481079920,  -0.000479548550466,  -0.000268067102911};

 float yl[1]={0};
 float xl[57]={0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0};


 int i;

 interrupt void c_int11()
 {
    float lc;
    float rc;
    float lcnew;
    float rcnew;
 //Left channel and right channel are switched
     AIC23_data.combo=input_sample();
     lc=(float) (AIC23_data.channel[LEFT]);
     rc= (float) (AIC23_data.channel[RIGHT]);
     // Inser DSP algorithm bellow
 for(i=56; i>0; i--)
 {
 xr[i]=xr[i-1];
 xl[i]=xl[i-1];
 }
 xr[0]=rc;
 xl[0]=lc;
 yr[0]=0;
 yl[0]=0;
 for(i=0; i<57; i++)
 {
 yr[0]=yr[0]+b[i]*xr[i];
 yl[0]=yl[0]+b[i]*xl[i];
 }

 rcnew=yr[0];
 lcnew=yl[0];

     //
     AIC23_data.channel[LEFT]=(short) lcnew;
     AIC23_data.channel[RIGHT]=(short) rcnew;
     output_sample(AIC23_data.combo);
 }



 void main()
 {
     comm_intr();
     while(1);
 }
```

Figure 4b Program segment for lowpass filtering.

```
Uint32 input_sample()                                  //for 32-bit input
{
    short CHANNEL_data;

    if (poll) while(!MCBSP_rrdy(DSK6713_AIC23_DATAHANDLE));//if ready to receive
         AIC_data.uint=MCBSP_read(DSK6713_AIC23_DATAHANDLE);//read data

//Swapping left and right channels (see comments in output_sample())
    CHANNEL_data=AIC_data.channel[RIGHT];          //swap left and right channel
    AIC_data.channel[RIGHT]=AIC_data.channel[LEFT];
    AIC_data.channel[LEFT]=CHANNEL_data;

    return(AIC_data.uint);
}

short input_left_sample()                                  //input to left channel
{
    if (poll) while(!MCBSP_rrdy(DSK6713_AIC23_DATAHANDLE));//if ready to receive
     AIC_data.uint=MCBSP_read(DSK6713_AIC23_DATAHANDLE);//read into left channel
    return(AIC_data.channel[LEFT]);                    //return left channel data
}

short input_right_sample()                                 //input to right channel
{
    if (poll) while(!MCBSP_rrdy(DSK6713_AIC23_DATAHANDLE));//if ready to receive
     AIC_data.uint=MCBSP_read(DSK6713_AIC23_DATAHANDLE);//read into right channel
    return(AIC_data.channel[RIGHT]);                   //return right channel data
}

// DSP coefficients, variables, and constants
float yr[1]={0};
float xr[57]={0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0};
float b[57]={0.000268067102911,   0.000479548550466,   0.000744481079920,
0.001088262533223,   0.001522694437199,   0.002040166137739,
0.002609377429647,   0.003173202897853,   0.003649126603227,
0.003932450753003,   0.003902226040124,   0.003429587413324,
0.002387931620007,   0.000664165631433,  -0.001829891318777,
-0.005146939224362,  -0.009295433763665,  -0.014233496388969,
-0.019866183241039,  -0.026046547745487,  -0.032580634719737,
-0.039236224904193,  -0.045754835299434,  -0.051866199863290,
-0.057304231913689,  -0.061823324243860,  -0.065213789202115,
-0.067315284712047,   0.931972789115646,  -0.067315284712047,
-0.065213789202115,  -0.061823324243860,  -0.057304231913689,
-0.051866199863290,  -0.045754835299434,  -0.039236224904193,
-0.032580634719737,  -0.026046547745487,  -0.019866183241039,
-0.014233496388969,  -0.009295433763665,  -0.005146939224362,
-0.001829891318777,   0.000664165631433,   0.002387931620007,
0.003429587413324,   0.003902226040124,   0.003932450753003,
0.003649126603227,   0.003173202897853,   0.002609377429647,
0.002040166137739,   0.001522694437199,   0.001088262533223,
0.000744481079920, 0.000479548550466,   0.000268067102911};

float yl[1]={0};
float xl[57]={0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0};


int i;

interrupt void c_int11()
{
   float lc;
   float rc;
   float lcnew;
   float rcnew;
//Left channel and right channel are switched
    AIC23_data.combo=input_sample();
    lc=(float) (AIC23_data.channel[LEFT]);
    rc= (float) (AIC23_data.channel[RIGHT]);
    // Inser DSP algorithm bellow
for(i=56; i>0; i--)
{
xr[i]=xr[i-1];
xl[i]=xl[i-1];
}
xr[0]=rc;
xl[0]=lc;
yr[0]=0;
yl[0]=0;
for(i=0; i<57; i++)
{
yr[0]=yr[0]+b[i]*xr[i];
yl[0]=yl[0]+b[i]*xl[i];
}

rcnew=yr[0];
lcnew=yl[0];

    //
    AIC23_data.channel[LEFT]=(short) lcnew;
    AIC23_data.channel[RIGHT]=(short) rcnew;
    output_sample(AIC23_data.combo);
}



void main()
{
    comm_intr();
    while(1);
}
```

Figure 4c Program segment for highpass filtering.

IV. Course Outcome and Assessment

Upon the completion of the course, a survey was conducted to ask each student to evaluate his or her achievement. Since the course title is “Real-time Digital Signal Processing”, so the class project is required to build up a Real-time DSP project. No comparison is available between the class with the project and without the project. However, in our future classes, other DSP real-time applications could be given to students, such as system modeling using an adaptive filter, a noise canceller, an echo canceller, etc.
Table 1 indicates survey results collected from the previous three semesters for a total number of 35 students. Note that the rating scale in Table 1 was based on the percentage of the overall students.

Table I. Student Survey for Achievement.

Rating Scale	Level of understanding	Excitements	Textbook
4 - excellent	80%	85%	90%
3 - good	20%	10%	10%
2 - fair	0%	5%	0%
1 -unsatisfactory	0%	0%	0%

Most of the students remained excited about the course, since the hands-on real-time laboratories had motivated them. The textbook also helped a great deal to develop concepts using the worked numerical examples and MATLAB simulation examples. Table 2 summarized the DSP project evaluation.

Table II. Evaluations of DSP Project.

Rating Scale	Project Performance
4 - excellent	80%
3 - good	20%
2 - fair	0%
1 - unsatisfactory	0%

As shown in Table 2, 80% of students gave an “excellent” rating during the evaluation while the remaining percentage obtained a “good” rating. The evaluation data shows promising results in which students continue to apply their gained DSP knowledge to their career development. It is very encouraging to teach the real-time DSP in the engineering technology program. Real-time implementations play a great part in our ECE and ECET programs, therefore, our real-time audio project showed the significant benefit to students, since they could experience the theoretical math being applied and played by watching the results from the scope and spectrum analyzer or hearing from the speaker. In fact, more and more instructors are encouraged to offer students real-world applications in our department.

After learning DSP courses, the technology students were able to apply their newly gained knowledge and skills to their senior projects and our real-time DSP labs served as good preparation and practice for senior projects. In our campus, senior students are required to present and demonstrate their senior projects in the senior project fair, in which those projects were evaluated by the engineering technology faculty members and other senior students.

V. Future Improvement

Based on our experiences from teaching DSP courses, we felt that in Portion 1, all the lectures containing well-established topics including the digital spectrum, the FIR and IIR filter implementations and developed laboratories are suitable. Even though the topics of DFT, FFT, bilinear transform method and optimum design seemed challenging to our technology students due to the demand of their math proficiency to understand certain subjects, we still have successfully delivered the course materials with an emphasis on principles and hands-on applications instead of theoretical development. On the other side, based on the DSP industrial trend, we could improve the course by introducing additional topics such as adaptive filtering, subband coding and wavelet coding (as well as their applications). To improve our lab, we should make use of the lab equipment fund to adopt more advanced DSP platforms with multi-channel ADCs and DACs, so that more practical real-time DSP laboratory projects can be developed.

VI. Conclusions

It has been a continuous demand in the industry for engineering technology students to possess a working knowledge of the advanced and real-time DSP techniques. The traditional treatment of teaching those subjects using the profound mathematics is not appropriate. However, with the mathematical simplification equipped with numerical examples, MATLAB simulations, and real-time laboratories, the technology students are able to grasp concepts effectively and apply their gained DSP knowledge to their careers and future technical practice.

VII. Bibliography

1. L. Tan and J. Jiang, Digital Signal Processing: Fundamentals and Applications, Elsevier/ Academic Press, 2nd edition, 2013.

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

3. Texas Instruments, Code Composer Studio: Getting Started Guide, Texas Instruments, Dallas, Texas, 2001.

4. L. Tan, J. Jiang, “Teaching Advanced Digital Signal Processing with Multimedia Applications in Engineering Technology Programs,” ASEE Annual Conference, June 2009.

5. de Vegte, Joyce Van. Fundamentals of Digital Signal Processing, Prentice-Hall Publishing, 2002.

6. Ifeachor, Emmanuel and Jervis, Barrie. Digital Signal Processing, A Practical Approach, Prentice-Hall Publishing, 2002.

7. C. Wicks, “Lessons Learned: Teaching Real-Time Signal Processing,” IEEE Signal Processing Magazine, pp. 181-185, November 2009.