Tools and Laboratory Exercises That Reinforce DSP Concepts and Motivate Technology Students

Richard E. Pfile, William C. Conrad
Indiana University-Purdue University at Indianapolis

Abstract

The goal of the DSP course in the EET department at IUPUI is to teach students how to program real-time DSP processors and to understand theoretical DSP concepts to the extent that they can comprehend literature typically seen in DSP data books and application notes.

Students are taught to program a Motorola 56002 DSP processor in assembly language during the first half of the course. This helps them to understand the unique properties of DSP processors and how to use the instruction set to efficiently implement DSP algorithms. The hardware is inexpensive and provides a hands-on learning experience that particularly benefits tactile learners. Because many college age students have an interest in audio systems, the audio applications used in the laboratory also motivate students to learn.

We have two hardware laboratories that are a bit different from the usual filtering and FFT exercises. In one laboratory students measure the frequency spectrum of a toy organ using a spectrum analyzer and implement the complex tone in a wave table on the hardware. This exercise helps students grasp the concept of time and frequency domains and gives students practice using circular buffers. In another laboratory, students write a simple real-time reverb program using comb filters and then use Matlab to generate an impulse time response of the system. The simple comb filter has an obvious time response that helps students understand an impulse response and provides a great starting point for introducing the z-transform.

To help demonstrate theory, Matlab laboratories are used to demonstrate the Fourier series, Fourier transform, convolution of a simple moving average filter, and impulse responses and pole-zero plots of different z-transforms. Simulation provides an avenue to reinforce basic signal processing concepts.

I. Introduction

The EET department at Indiana University-Purdue University at Indianapolis has developed an applied Digital Signal Processing course that focuses on the

implementation of DSP algorithms using a real time DSP processor. Technology programs stress the application of technology for problem solving, and using DSP processors in the laboratory allows students to implement DSP algorithms in real-time systems that have many practical applications. Theory is introduced as needed to solve application-based problems. The applications provide reinforcement for the theory and a motivation for learning the theory. Because DSP processors have an instruction set optimized for implementation of DSP algorithms, the programs are quite short and programming can be taught in very little time. The ability to program in assembly language is a prerequisite for the course. With the assembly language prerequisite, I have found that it is possible to teach both DSP theory and programming a DSP processor in one four credit hour course.

DSP theory is also taught in the course. The goal is for students to be capable of reading and understanding data books and application notes written by DSP chip manufacturers. The math required to attain this goal is met with a trigonometry course and two applied calculus courses.

II. DSP Topics Taught

To achieve the goal of teaching students to comprehend manufacturers applications notes the course covers Fourier Series, Fourier Transform, Convolution, FIR and IIR filter design, Z-transform, DFT, and FFT. The course lectures are built around the laboratories and theory is introduced in a just-in-time fashion whenever it is needed to solve a problem. Matlab exercises are used to reinforce basic concepts.

In prerequisite courses, students have typically worked out trigonometric Fourier series exercises, but in the DSP course they are introduced to solving Fourier series problems using exponential notion. This helps to get students used to working with complex exponential numbers, a concept that is useful throughout the course.

Discrete convolution is taught using both a moving average filter and the FIR filter as practical examples. Moving average filters with variable numbers of coefficients are simulated using Matlab. For this exercise high frequency sine wave noise is removed from a lower frequency signal.

We use the windowing method to calculate FIR filter coefficients. The IDTFT is introduced and used to calculate the impulse response for an ideal lowpass filter. The only calculus required is the integration of a complex exponential and students are familiar with this since they have calculated Fourier series problems. Code to implement a FIR filter on the 56002 processor is also presented.

IIR filter design is taught using several techniques. This is the point in the course where the z-plane and z-transform are introduced. We start IIR filter design by judiciously placing poles and zeros in positions in the z-plane to design IIR filters by inspection. The z-transform is used to examine the stability of the filters designed. The impulse response of the z-transform is determined three ways: (1) by long division, (2) using partial fraction expansion and table lookup, and (3) by applying an impulse to the z-transform's difference equation. Matlab simulations

are used to examine the frequency and impulse responses of z-transforms. A plot of a Matlab simulation of a system with a complex pole and zero is shown in figure 1. Code to implement a FIR filter on the 56002 processor is also presented.

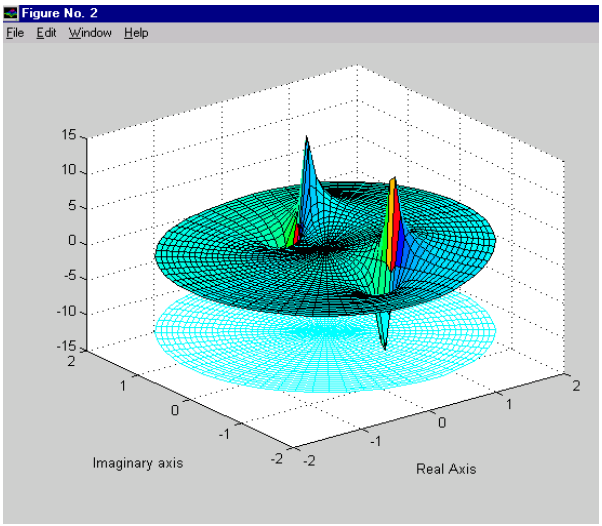

Figure 1. Matlab frequency response

IIR filters are also designed using the Impulse Invariant Method and the bilinear z-transform method (BZT).

Students calculate DFT coefficients and an overview of the FFT is presented along with the reverse-carry addressing used by the microprocessor to implement the FFT.

III. Laboratory Hardware

A fixed-point processor was selected for the course because they are used in a growing body cost sensitive applications. A 24 bit fixed point processor has a dynamic range of 144 dB ($20 \log_{10}(2^{24})$) which is adequate for all but the most demanding applications.

We chose a Motorola DSP56002, an upgrade of the DSP56000, for the course. This 24 bit fixed point processor is available in versions with clock speeds up to 100 MHz, can perform parallel instructions, and has three parallel memory spaces that can be accessed simultaneously for high throughput. The processor also has built-in peripheral functions such as a synchronous serial interface, serial communications interface, multiple interrupts, and parallel I/O ports which make it flexible and easy to use for student projects. We want student to know the characteristics of DSP processors.

A low cost evaluation system for the 56002 processor called the 56002 EVM is available. It retails for $149.00. but is available for $120. with educational discounts (contact Motorola University Support). The EVM comes complete with an assembler and full screen debugger. It has 32K of memory and can be

powered by a 9-volt plug-in wall transformer or dc supply. The EVM has stereo analog inputs and outputs with 1/8 inch stereo plugs making the EVM compatible with a variety of common audio devices such as cassette and CD players, keyboards, earphones and powered speakers. Students enjoy trying out DSP algorithms using CDs or cassettes they bring in from home. Several students purchase units on their own so they can use them at home and for personal projects.

The EVM uses a 16-bit multimedia audio codec for the A/D and D/A functions. The codec is programmable and can be setup for sample rates from 4 kHz to 48 kHz. It also has programmable input and output amplifiers that allow the unit to be compatible with a variety of equipment.

IV. Development System Software

The 56000 cross assembler supplied with the EVM is high quality and provides all of the functionality of the commercial assembler except the capability of linking object modules. Because the 56000 processor is optimized for DSP algorithms, the programs tend to be rather short and the inability to link modules in the classroom environment is not a hindrance. Otherwise the assembler is complete with macros and a full slate of assembler directives including special directives to generate sine, cosine and random number tables.

The debugging software for the evaluation module has pull-down menus and is easy to use. The debugging screen displays processor registers, disassembled program code, and memory. Programs can be stepped through or run at full speed. Breakpoints can be inserted and memory locations displayed and modified using symbolic names or actual addresses. Data can be displayed in hexadecimal, decimal, binary, or fractional notation.

V. Laboratories

The laboratories are such a key component of the course that the lectures are designed to support the laboratory experiences.

- Introductory Laboratories

Students spend the first three laboratories becoming familiar with the Motorola 56002 architecture and EVM software. In the first laboratory, students load executable files, examine and modify memory and registers, step through code, set program breakpoints, and exercise other capabilities of the debugging software. Using the full-screen debugger, students see the 56002 registers and memory on the screen at one time and quickly become familiar with the architecture of the chip.

The second introductory laboratory focuses on the fractional number representation used in DSP processors. Fractional numbers are left justified to the right of the decimal point and students examine the numbers after move, multiplication and addition instructions have been executed. They also examine the effects of rounding and the treatment of numbers as they are moved to

registers of various lengths.[1]

In the final introductory laboratory, students write the code to sample a stereo signal using the audio codec on the EVM board and then send it right back out the codec D/A converter. Interfacing to the codec chip is somewhat complex, but Motorola supplies the code to access it. Using an include statement the codec software can be assembled with the student's program and they can be isolated from the many details involved in accessing the codec. The codec is read and written to by reading and writing to memory locations. The codec interrupts the processor at the prescribed sampling rate and a program only needs to monitor some control bits to determine when data is ready. The codec is accessed using the Motorola Synchronous Serial Interface (SSI). Depending on the goals and time available in a course, details of the SSI interface can be covered in class or skipped entirely.

- Sine Wave Generators

For the forth laboratory students generate a sine wave. This simple exercise allows students to become familiar with circular queues, a central feature of DSP processors. Loading the size of the queue minus one in a register and setting a pointer register to the beginning of the queue can specify a circular queue of any length, on the 56002 processor. When the pointer is incremented to the end of the queue it will automatically wrap around back to the beginning of the queue.[2]

In the next laboratory an interrupt is set up to step through the circular queue at a set rate to make different frequency tones corresponding to the notes of guitar strings. A timer/counter with an interrupt is used to step through the sine table at the appropriate rate. Using the timer with interrupts is easy to teach and is an important concept for DSP processors. We usually have several students in the class who play guitar and are particularly motivated by this laboratory.

For the next laboratory we measure the harmonics of a complex tone on a small electronic keyboard and duplicate the sound using the DSP. The harmonics are measured using an oscilloscope that has FFT calculation capability. Going through the process of measuring the organ harmonics, helps students to better understand the frequency domain and gives them practice making decibel calculations. Sine waves at five different frequencies and amplitudes make up the tone. There are two methods of implementing this algorithm. One method we have used is to create a single sine table and use multiple pointers to access the sine table at different rates to produce the harmonics. This exercise allows students to use interrupts and get practice with multiple pointers. The 56002 can implement this in just a few instructions using the circular registers with the associated index registers. Another method of solving the problem is to create a wave table that contains the harmonics and have the software step through the table.

- Special Effects

We have two experiments that are fun and have significant educational value. Both of these laboratories use comb filters that are simple and provide a good entry point into the theory of feedback systems. The first is a reverberation generator. Reverberation describes an electrical or mechanical process in which

sound bounces off walls and gets combined with its own echoes. Reverb is often used to liven up music.

The reverb is basically a comb filter which works by adding at each sample time a delayed and attenuated version of each past output. Figure 1 shows a block diagram of a comb filter.

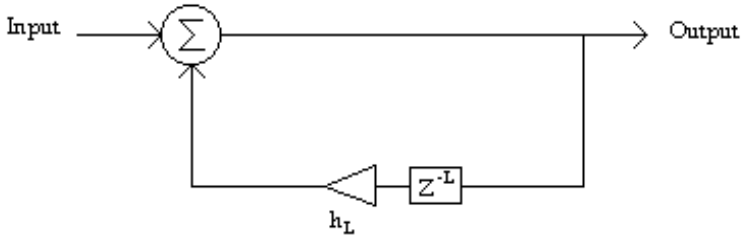

Figure 2. Comb Filter with Delay L.

The difference equation for the filter is:

$$y_n = x_n - b^k y_{n-k}$$

The z-transform for the equation is:

$$H(\omega) = \frac{1}{1 + b^k z^{-k}}$$

The frequency response of the comb filter emphasizes frequencies at the poles and attenuates other frequencies. If k is eight in the above equation the system has eight poles evenly spread out in the z-plane.

Several comb filters with different delays and feedback gains can be combined to produce a striking sound. I use this exercise to introduce students to the z-transform and difference equations. Because the z-transform and difference equations for a comb filter are simple, they provide a good example for introducing feedback systems.

In a second laboratory a comb filter with a random number series input is used with a comb filter to simulate a plucked-string instrument such as a guitar. Frequencies in the random number series that match the loop delay period are enhanced and others are attenuated. To better simulate an actual string, high frequencies components of the signal need to be attenuated at a faster rate than low frequency signals.[3] A simple two state moving average filter performs this function. The block diagram for the system is shown in figure 3.

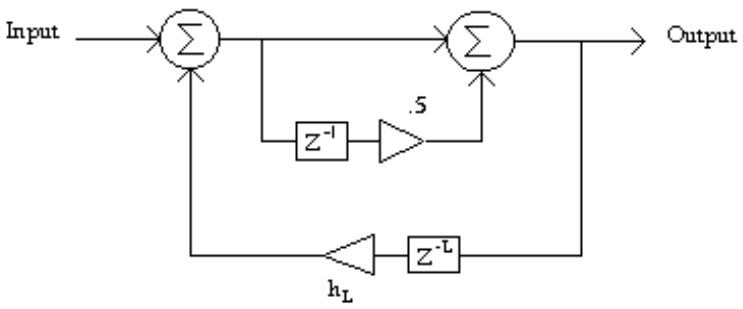

Figure 3. Plucked-string filter.

- Filters

For the next laboratory we implement a finite-duration impulse response (FIR) filter. We use the software package QEDesign to calculate the coefficients, although these are quite easy to calculate manually. Matlab with the Signal Processing Toolbox can also be used to calculate coefficients for the FIR digital filter. Matlab has a module (fir1) which will generate the coefficients for a lowpass filter and put them in a row vector.[4]

After the coefficients are calculated students implement a lowpass filter on the 56002 EVM. Again because the 56002 has specialized DSP instructions the program is very short. The code for the FIR filter loop (code to initialize registers is omitted) is shown below:

```
                ; FIR filter
                ; ****Place this code at insertion point B
                ;  Get A/D value
                move        a,x0
                ; Move A/D data into buffer
                ; Move 1st coefficient into y0
                clr         a           x0,x:(r0)+      y:(r4)+,y0
                ; Repeat multiply and accumulate, ntaps -1 times
                rep         #ntaps-1
                mac         x0,y0,ax:(r0)+,x0       y:(r4)+,y0
                ; Last multiply and accumulate with rounding
                macr        x0,y0,a(r0)-
                ; Put data in right channel too
                move        a,b
;*****************************************************
```

In the above program the filter coefficients are located in y memory and the analog signal data is stored in x memory. By locating the data and coefficients in two different memory spaces, parallel access may be executed, increasing execution speed. A function generator can provide a suitable input and an oscilloscope or headphones can be used to monitor the output.

The last two laboratories of the class are an Infinite Impulse Response (IIR) filter and a Fast Fourier Transform (FFT). The code for the IIR filter is quite compact, but the FFT algorithm is somewhat complex involving nested loops to solve the butterflies for the Decimation-in-Time butterflies.[5] Fortunately Motorola provides a good reference with 56002 implementations of a FFT implementation.[6]

Summary

It is easy to implement real-time processing in a DSP course. Adding a real time laboratory to a DSP course is inexpensive and if students are already familiar with microprocessors it does not take much classroom time to teach DSP processors.

1. Mohamed El-Sharkawy, *Real Time Digital Signal Processing Applications with Motorola's DSP 56000 Family*, Prentice Hall, Englewood Cliffs, New Jersey, 1990.

2. *DSP56000 Digital Signal Processor Family Manual*, Motorola Inc, Phoenix, Arizona, 1992.

3. Ken Steiglitz, *A Digital Signal Processing Primer with Applications in to Digital Audio and Computer Music*, Addision-Wesley Publishing Company, Inc., Menlo Park, California, 1996.

4. Thomas P. Krauss, Loren Shure, and John N. Little, *Signal Processing Toolbox for Use with MATLAB*, The Mathworks Inc., Natich, Massachusetts, 1993.

5. Emmanuel C. Ifeachor and Barrie W. Jervis, *Digital Signal Processing-A Practical Approach*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1993.

6. Guy R. L. Sohie and Wei Chen, *Implementation of Fast Fourier Transforms on Motorola's Digital Signal Processors*, Motorola Inc., Phoenix, Arizona, 1993.

Richard E. Pfile is a professor of Electrical Engineering Technology at IUPUI. He received his B.S. from the University of Loiuisville and his M.Eng. from the University of Michigan. He has won the Outstanding Teaching Award and has received Teaching Excellcellence awards from the School of Engineering and Technology at IUPUI. He teaches courses in microprocessor systems, computer networks and digital signal processing. He has fifteen years of teaching experience and eight years of industrial experience, including three years as a systems engineer.

William Conrad received his MEng degree in general engineering from The Pennsylvania State University in 1968. He is a member of the Indiana University-Purdue University at Indianapolis Electrical Engineering Technology faculty since 1991. Before joining IUPUI, he was associated with the John Deere Foundry East Moline, serving as a senior engineer in the Plant Engineering Department. He has also taught in the Electrical Engineering Technology Department at Michigan Technological University, Houghton MI. He is a registered professional engineer.