EXPERIENCES IN A REAL-TIME DSP DESIGN COURSE

Mehmet Zeytinoğlu and N.W. Ma

Department of Electrical and Computer Engineering
Ryerson Polytechnic University,
Toronto, Ontario, M5B 2K3, Canada
e-mail: {mzeytin,bma}@ee.ryerson.ca

ABSTRACT

This paper presents our experience at Ryerson Polytechnic University in designing and offering a fourth year DSP course which emphasizes the teaching of analytic tools to study and design DSP systems as well as the real-time implementation of such systems. One of the inherent advantages of an integrated DSP course is the ease with which abstract mathematical algorithms can be brought to life through real-time implementation.

We first discuss our philosophy which determined the approach we have taken in structuring the DSP course and in particular its laboratory component. We present the course outline, the laboratory set-up, the experiments and the projects. In introducing the laboratory experiments and projects we highlight the advantages of real-time implementation over simulations.

1. INTRODUCTION

Digital Signal Processing (DSP) is one of the most rapidly developing disciplines that has emerged from within electrical and computer engineering in the last 40 years. The unabated technological innovations in digital computers and microprocessors paved the way for the migration of traditional analog signal processing systems to a digital world. While many analog signal processing techniques can be translated directly to a digital domain, a number of novel digital signal processing algorithms opened up the possibilities which were hitherto unheard of. One can cite audio, image, and video processing and compression as examples of such exciting new algorithms.

In parallel with the ever increasing speed and processing power resulting from advances in semiconductor manufacturing, researchers from around the world continue to design and implement powerful signal processing algorithms. Major developments in DSP have proceeded along three distinctly identifiable paths. In particular, these are:

- development of new DSP algorithms (MPEG);
- development of a theoretical framework for the understanding of discrete time signals and systems (wavelet theory vs. filterbanks);
- development of integrated circuit architectures suitable for efficient DSP (reconfigurable DSP architectures, FPGAs with DSP core).

Of course, the expanding interest in DSP is also the result of the diversity of DSP applications: communications (cellular and mobile communications), automotive engineering (engine management systems for safety and fuel efficiency), consumer electronics (compact disk players, HDTV), computing (head positioning systems in computer hard disks, ATM switches) are just few of the application areas where DSP has made significant inroads. Consequently, interest in DSP related courses at an undergraduate, graduate and professional development levels has also increased.

Typical university level DSP courses range from a system level course where topics are covered using a block diagram approach in a simulated environment to a real-time oriented course where the emphasis is on real-time implementation and hardware architectures.

While the ideal solution may be to cover the material in two separate course, one introducing the theory and the other implementation issues, the already crowded undergraduate curricula usually does not allow such luxuries. If there is only a single DSP course in the curriculum adhering to one specific approach, this may have serious drawbacks. DSP theory is developing very rapidly; hence there is an increasingly larger volume of theoretical fundamentals to be conveyed. Yet, the increasing significance of real-time implementation of DSP algorithms also requires a solid understanding of hardware architectures suitable for DSP.

In this paper, we present our experience in structuring an undergraduate DSP course. We will first discuss our philosophy which determined the approach we have taken in structuring the DSP course and in particular its laboratory component. We present the course outline, the laboratory set-up, the experiments and the projects. In introducing the laboratory experiments and projects we highlight the advantages of real-time implementation over simulations.

2. COURSE STRUCTURE

Our approach to this course has been very much a reflection of our educational philosophy as faculty members within a polytechnic university. The course is offered in a 3 hours per week lectures and 3 hours per week laboratory format over 13 weeks for a total of 78 contact hours. This gave us flexibility in formatting the course in a way which emphasizes implementation issues in the projects/laboratory work while simultaneously being able to offer a full and rigorous coverage of the discrete system fundamentals. While working on the projects students spend on the average 2–3 hours per week on this course outside of the regularly scheduled contact hours.

We offer this DSP course as a 4th year technical elective course. It is a very popular course typically selected by over 85 students, which roughly corresponds to ninety per cent of the 4th year enrollment. All students selecting the DSP course would have completed two (relevant) third year courses: *Signals and Systems* and *Microprocessor Systems*. We build on the material covered in these courses. Students would also completed courses on computer architecture, communication and control systems, such that examples from these topics can be introduced.

2.1. Our Philosophy

From the very beginning our main goal has been to design a course which provides the students with a solid understanding of DSP fundamentals. This approach is essential for life-long learning where students after graduation will be able to upgrade their skills and knowledge in this ever evolving field. The following themes, however, spawned lengthy debates through which the course structure started to emerge.

• **Simulation vs Real-Time Implementation.** We wanted the student to be familiar with real-time implementation issues. Testing various DSP algorithms in a simulated environment is tremendously useful to understand the concepts and to observe the interrelation among parameters that typically govern DSP algorithms; for example, long-term and short-term prediction filters, effects of pole-zero locations on the system behaviour, filtering. However, implementation issues (such as buffering, block processing, interrupt driven processing) can only be understood when the student works on a real-time implementation. Therefore, in designing the course we relied on simulations (mostly in a Matlab environment using Mathworks supplied or in-house developed toolboxes) in support of the lectures. In the laboratory component of the course we operated exclusively with real time implementation (see the laboratory projects section 2.3 for further details).

• **Programming in C vs Assembler.** Programming in a high-level language such as C has the inherent advantage of easy transportability. Yet the appreciation of the DSP architecture is only possible by programming in assembler where the student has the opportunity to explore the processor architecture highly tuned for efficient digital signal processing. A good example is the FIR filter implementation where the programmer has to make extensive use parallel memory access features.

• **Floating vs Fixed Point Architecture.** Floating point DSPs are increasingly becoming more popular. This is simply a result of their decreasing price. Furthermore, some of the more recent DSP algorithms require the higher precision and greater dynamic range achievable only with a floating point architecture. Yet, we also recognize that there always will be a demand for fixed point DSPs, primarily at lower end of the DSP spectrum where fixed point DSPs continue to dominate the market. Therefore, we believe that students who complete this DSP course must have a solid understanding of issues relevant to fixed point implementation of DSP operations.

• **Hardware Architecture.** There are a number of issues that come up and/or learned from the real-time implementation of DSP algorithms, such as pipelining, parallel memory access, barrel shift operations, bit reversed addressing. An intimate knowledge of such architectural features of current crop of DSPs provides the student with an understanding of architectural features important for efficient, real-time implementation of DSP algorithms. As the design of dedicated DSP integrated circuits is increasingly becoming more widespread a solid understanding of the hardware architectures is essential.

2.2. Course Outline

The course introduces the basic principles and techniques of DSP, which are reinforced through design and implementation of real-time algorithms.

Major Topics:

• **Discrete time description of signals and systems:** discrete-time sequences, basic DSP operations, A/D, D/A conversion.

• **Frequency domain analysis:** sinusoidal steady-state response of LTI systems, frequency response, stability, analog and digital frequencies.

• **$\mathcal{Z}$-transform and its applications in signal processing:** properties, transfer function, evaluation of the inverse $\mathcal{Z}$-transform, connection between the time and the $\mathcal{Z}$-domain, graphical concepts, effects of poles and zeros on the frequency response, graphical design of filters.

• **Discrete transforms:** discrete Fourier series, finite duration sequences and the discrete Fourier transform (DFT), the DFT approximation to the Fourier transforms, properties, periodic and circular convolution, FFT, decomposition in time and in frequency, variations on the basic algorithm, fast convolution.

• **Digital filter structures and design techniques:** Introduction to digital filters, types of digital filters: FIR and IIR filters, choosing between FIR and IIR filters, FIR and IIR filter design techniques, system implementation, finite length register effects in fixed point digital filters, quantization, filter structures free of overflow oscillations, scaling, roundoff noise, coefficient sensitivity.

• **Selected topics:** Multirate signal processing, sampling rate conversion, adaptive filtering.

2.3. Laboratory

During the first half of the term students learn DSP programming. They implement DSP algorithms which are the basic building blocks frequently used in the design of more advanced DSP applications. In the second half of the term students work on two projects which emphasize the real-time signal processing.

Laboratory Set-up

In the laboratory students implement DSP algorithms on the Motorola DSP Application Development System (ADS). The ADS contains a 40-MHz Motorola 56002 fixed-point DSP processor, and a single channel A/D-D/A subsystem that is capable of sampling signals up to 50 kHz. Students access the ADS through control software which interacts with the user and controls the application development module. Each ADS is connected to a host computer, a Sun SPARCstation, that allows the student to run the assembler, linker, simulator and the ADS control software. Through the host computer students also access other related software for simulation or digital filter design (Matlab/DFP). Each "DSP-station" is equipped with an oscilloscope, a function generator and a spectrum analyzer. For audio based experiments such as the speech scrambler-descrambler project we also provide an audio source (typically a CD player fed to the DSP-stations) and a speaker to listen to the processed audio output.

Since 1995 we have also been using the Motorola DSP evaluation modules (DSP56002-EVM). This EVM is a low cost alternative to the Motorola ADS with nearly identical functionality. Students find the modest investment (about Cdn. $120) to acquire such a system worthwhile, as the EVM allows them to work on projects outside of the laboratory. Furthermore, students who continue working on DSP based 4th year design projects frequently use the EVM as the basis of their projects. This frees them up from the time-consuming and error-prone process of designing and building their own DSP boards. All major semiconductor manufacturers market similar full-function, low-cost evaluation modules based on their DSPs. These kits typically include 2 channel A/D-D/A subsystems and DSP code development software. Therefore, by using these inexpensive evaluation modules, it is possible to set up a DSP laboratory based on any modern DSP platform very economically.

In the first two experiments students design and implement FIR and IIR filters. A good understanding of this elementary DSP operation is essential as digital filters constitute the backstay of all the subsequent experiments and projects. While Matlab with the Signal Processing Toolbox provides all the functions required to design digital filters, the subtleties of how to use these functions escape many students as they do not use them on a regular basis. To overcome this difficulty we developed the Digital Filter Package (DFP) which provides a user friendly GUI front-end to digital filter design with MATLAB. DFP extends the basic digital filter design functionality of MATLAB in two important ways. Filter coefficients can be quantized. This feature is of particular importance if the filter will eventually be implemented on a fixed point digital signal processor. DFP also generates assembler code for the designed digital filter. In the current DFP release (Version 1.1) this option is available only for the Motorola DSP56k family of fixed point processors.

Experiments and Projects

Experiments

- Basic Programming with the Motorola ADS.
- FIR Filtering.
- IIR Filtering and Speech Scrambler.

Projects

- Spectrum Analyzer.
- An Interactive Four Band Audio Equalizer.
- Digital VCO and PLL.
- DSB-AM modulation and demodulation.
- Multirate implementation of digital filters.

In a typical offering of the course students complete two projects. The first two projects (spectrum analyzer and equalizer) are both based on wideband signal decomposition. Therefore, if the first project the students work on is the spectrum analyzer, then the equalizer becomes the second project. This approach allows the students to capitalize on the experience and understanding gained from the signal decomposition used in the spectrum analyzer experiment. Figure 1 depicts the block diagram of the spectrum analyzer. The "filterbank" in Figure 1 is implemented using both an FFT based signal decomposition and also a 5-stage, 32-band regular tree decomposition. In the 32-band filterbank case students use multirate signal processing techniques to achieve efficiency in implementation.

In the interactive equalizer project (see Figure 2 for a block diagram) we also introduce the student to interactive control of the DSP algorithm. Through a GUI front end (Figure 3) running on the host computer students can change the subband gains. This information is then communicated from the serial port on the host computer to the serial port of the DSP by generating an interrupt. The interface routine is generic and allow students to incorporate other interactive control elements into their design projects.

The digital VCO/PLL project together with the DSB-AM project constitutes a "natural" pair as the PLL is used in the coherent detection of the DSB-AM signal. The main elements of the VCO-PLL projects include a closed loop feedback system, sine-wave synthesis using a table look-up method, phase shifting with circular buffer. These basic DSP subprojects familiarize the student with the implementation techniques used in the real-time realization of many DSP algorithms.

In recent years multirate signal processing have gained importance. Therefore, it is essential that the students completing a DSP course are well-versed with the analysis and real-time implementation of multirate systems. While the implementation of decimation and interpolation are most straight forward in a simulated environment such as Matlab, they present special challenges if implemented in a real-time environment. The filterbank project, and the multirate implementation of digital filters are substantial projects that expose the students to this exciting area of DSP.

3. CONCLUSION

In this paper we presented our experience in designing and offering a fourth year DSP course which emphasizes the teaching of analytic tools to analyze and design DSP systems as well as the real-time implementation of such algorithms. One of the inherent advantages of teaching such a DSP course is the ease with which the rather abstract mathematical algorithms can be brought to life through real-time implementation.

We recognize the development of ever more powerful DSP design tools. These tools include block diagram simulation (Simulink, SPW, Ptolemy), fixed-point arithmetic analysis and automatic code generation for various target DSP architectures. An engineer working in the DSP algorithm development will utilize these advanced tools to achieve high productivity in his/her work. We strongly believe that real-time implementation of DSP algorithms by low level programming a DSP provides the student with an appreciation and intimate understanding of the subtleties of real-time programming. Implementing DSP algorithms in a simulation environment is also a rewarding experience mostly because of the ease with which rather complicated concepts can be visualized.

4. REFERENCES

The most recent version of the DFP is available for free download from the Mathworks ftp site at

`ftp://ftp.mathworks.com/pub/contrib/`

and also from the department's WEB site

`http://www.ee.ryerson.ca/ mzeytin/dfp/.`

All course related material is available form the first author's WEB site

`http://www.ee.ryerson.ca/ mzeytin/ele792/.`

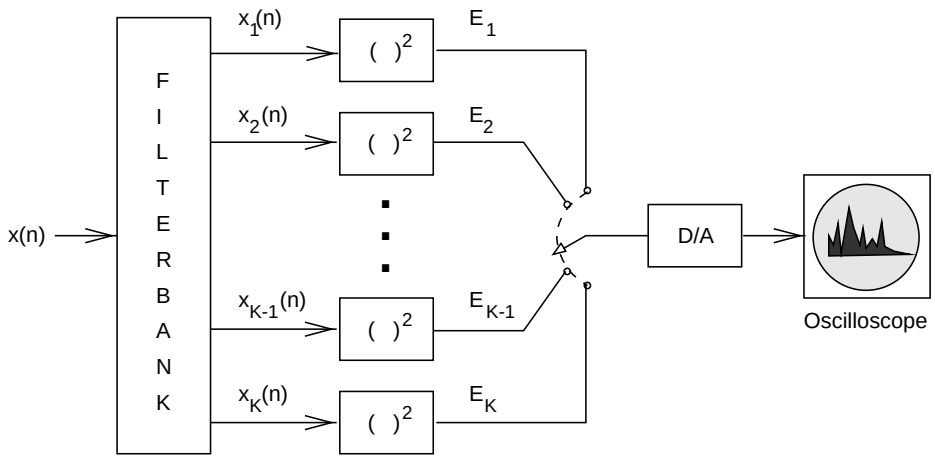

Figure 1: K-band spectrum analyzer

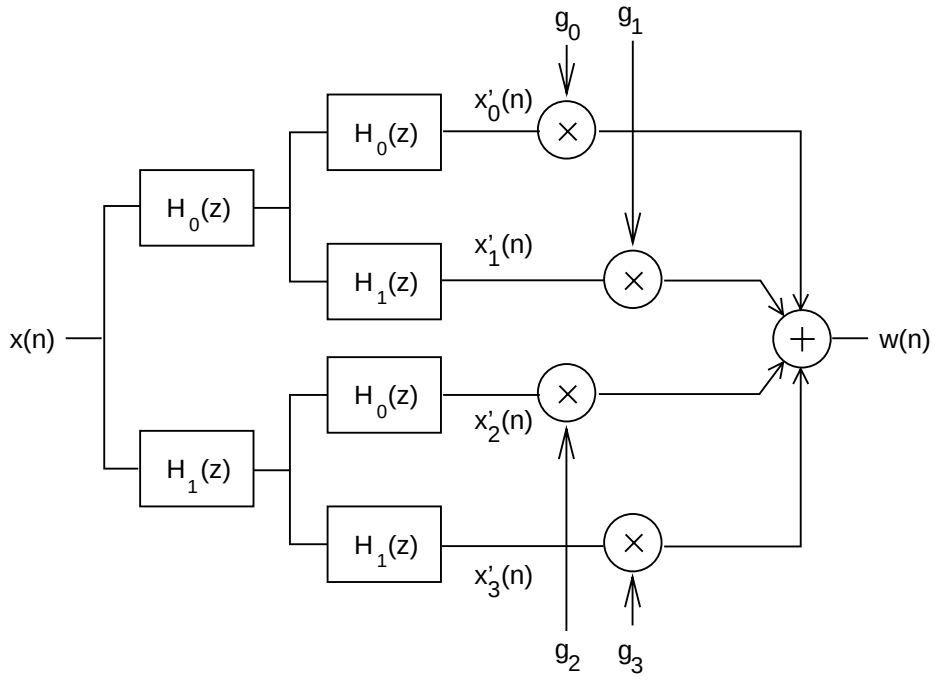

Figure 2: Equalizer block diagram

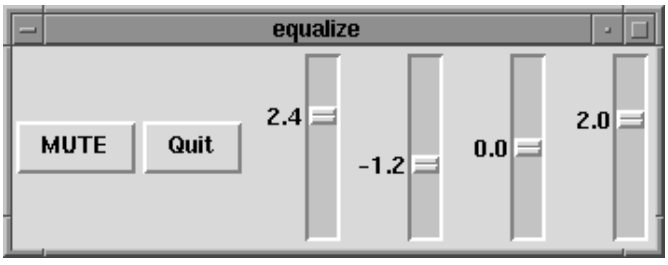

Figure 3: Equalizer GUI for adjusting subband gains.