14th European Signal Processing Conference (EUSIPCO 2006), Florence, Italy, September 4-8, 2006, copyright by EURASIP

A SIMULINK© AND TEXAS INSTRUMENTS C6713® BASED DIGITAL SIGNAL PROCESSING LABORATORY

Sharon Gannot and Vadim Avrin

School of Engineering, Bar-Ilan University
Ramat-Gan 52900, Israel
phone: + (972) 3 531 7618, fax: + (972) 3 534 0697, email: gannot@eng.biu.ac.il
web: `www.eng.biu.ac.il/ gannot`

ABSTRACT

In this contribution[1] a *digital signal processing* educational lab, established at the School of Electrical and Computers Engineering at Bar-Ilan University, Israel is presented.

A unique educational approach is adopted. In this approach sophisticated algorithms can be implemented in an intuitive top-level design using Simulink©. Simultaneously, our approach gives the students the opportunity to conduct hands-on experiments with real signals and hardware, using Texas instruments (TI) C6713 evaluation boards. By taking this combined approach, we tried to focus the efforts of the students on the DSP problems themselves rather than on the actual programming.

A comprehensive ensemble of experiments, which expose the students to a wide spectrum of DSP concepts, is introduced in this paper. The experiments were designed to enable the illustration and demonstration of theoretical aspects, already acquired by several DSP courses in the curriculum.

1. INTRODUCTION AND MOTIVATION

The last four decades have witnessed *digital signal processing* (DSP) becoming a very well established discipline. DSP finds applications in numerous fields, e.g. speech processing, communications and bio-medical engineering. The ease of manipulating discrete-time data and the abundant theoretical background, together with the low cost and availability of sampling and processing hardware, makes digital solutions a first choice, when handling continuous-time signals. It is therefore not surprising that the Hi-Tech industry is craving for young engineers with profound theoretical knowledge as well as application skills.

Most modern Electrical Engineering curricula consist therefore of several courses in DSP. Unfortunately, The DSP courses are usually perceived by the students as too theoretical and mathematically intensive. Although we highly regard the profound mathematical basis, it is our aim to make DSP more tangible. Our main goal in establishing the DSP laboratory is to demonstrate and illustrate the concepts taught in the theoretical courses and enable the students to experience the ideas in practice. Although, nowadays, all DSP courses use Matlab© exercises for illustrating the theory, we think that hands-on experience using real-time hardware is crucial for the basic understanding of the material.

For many years, DSP educators are trying to bridge the gap between theory and practice. As early as in the year 1992 a combined Lecture-Lab course was presented [1]. This course was divided into a lecture part and a lab part which interacted with each other. The Analog Devices ADSP-2101 processor with its own software tools was used in this lab.

The concept of establishing a library of building blocks was introduced in [2], making the teaching task faster and easier. The same concept was adopted in [3], where a real-time signal waveform library and a processing system, based on the TMS320C5x DSP starter kit, was introduced as a building block for an undergraduate lab.

The TMS320C5402 DSP starter Kit (DSK5402) is used in [4] and [5] as the core of a teaching lab. In the former a combination of C and Assembly programming languages is presented. In the latter the main programming language is Assembly. Since Assembly programming might become very tedious, the students are given nearly ready-to-use Assembly code segments which they only need to complete and modify as they progress in the experiments.

A entirely different approach for creating DSP labs is presented by Gan *et al.* [6, 7]. Instead of concentrating on C (or Assembly) programming, the students are working with Matlab and Simulink. The powerful *Real-time workshop* and *Embedded Target for TI C6000 DSP* toolboxes [8] are used to translate a DSP system to real-time hardware. However, in the above references the idea is only shortly exemplified and no attempt is made to create a lab which is entirely based on the Simulink-DSP hardware connection.

In the current contribution we describe an educational lab, based on Simulink© and Texas Instruments C6713®, established in our University. This lab adopts the above concept of using Simulink in conjunction with DSP hardware and further expands and enhances it. A comprehensive ensemble of experiments, which expose the students to a wide spectrum of DSP concepts, is introduced.

We have a twofold reason for selecting our approach for designing the educational DSP lab. First, the use of Simulink enables the creation of sophisticated algorithms in an intuitive top-level design. Simultaneously, this approach gives the students the opportunity to conduct hands-on experiments with real signals and hardware. We tried to focus the efforts of the students on the DSP problems themselves rather than on the actual programming. As we do not wish to neglect important programming skills, part of the experiments are dedicated to C programming of algorithms using the *code composer studio* (CCS) tool provided by TI. By fulfilling the lab requirements the students gain some important DSP tools, e.g. Simulink and CCS, while enhancing their already established skills in Matlab.

We do not pretend that our educational lab is training the students for becoming DSP programmers. Students who wish to enhance their skills in real-time programming and to specialize in DSP algorithms are given the opportunity to elect a graduation project in the DSP lab. In this short paper we do not elaborate on this DSP *project lab*.

The structure of this work is as follows. In Sec. 2 the experiments are listed and the equipment used is described. In Sec. 3 we elaborate on two representative experiments, namely the *sampling theorem* and *adaptive filtering* experiments. We summarize the concept behind the educational lab and draw some conclusions in Sec. 4.

2. DESCRIPTION OF THE LABORATORY

The educational lab consists of seven experiments. Two meetings are dedicated to each experiment, each of which is three hours long. The experiments are conducted in a dedicated work station. This section describes the content of the lab.

[1]The authors are grateful to the TI Europe University Program, and especially Mr. Robert Owen, for supporting the establishment of the DSP laboratory in Bar-Ilan University.

2.1 Work Station Setup

Each pair of students carry out the experiments in a dedicated work station. Each lab station is equipped with the following items: Personal computer with Matlab, Simulink and CCS, C6713 DSP interface box (see Fig. 2), microphone, loudspeakers, headphones, oscilloscope, and an arbitrary signal generator. A typical station is depicted in Fig. 1. The core of the station is the DSK C6713 evaluation

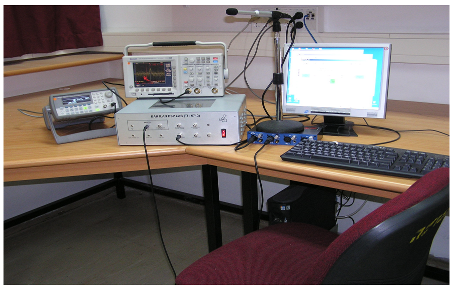

Figure 1: A typical station setup

board installed within a specially designed interface box[2] depicted in Fig. 2. This box provides all the necessary interface connectors to the peripherals and protects the board itself from any damage.

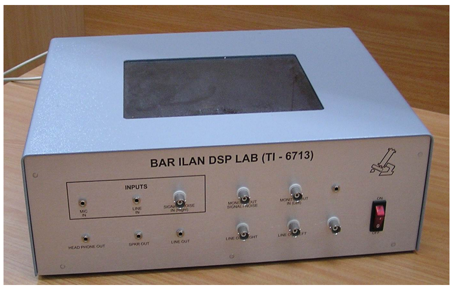

Figure 2: DSK C6713 interface box

2.2 The Experiment Procedure

The students starts to prepare themselves to the experiment at home. A lab manual, available online, includes a survey of the relevant theoretical background. At the beginning of the lab meeting, the students have to hand out a short report, comprised of answers to several relevant questions, which they have prepared in advance. This procedure guarantees the readiness of the students to the conduct the experiment. In the laboratory meeting itself the students are given a questionnaire with self-explanatory instructions. Incomplete Simulink models are provided. Students can upload these files to the Matlab workspace, complete and correct them and then use in their simulations. The Simulink model is then downloaded to the DSP board, using the *real-time workshop* toolbox.

2.3 The Experiments

The lab is designed to give the students an overview the of most important DSP concepts in their studies. The experiments represent a variety of topics studied in several course, starting from the basic *signals and system* course, proceeding with the *introduction to digital signal processing* course, and concluding at the advanced *statistical signal processing* course.

In Sec. 3 we elaborate on two of the experiments. Here only a brief overview of all the experiments is given.

[2]We thank Dr. Arie Yeredor and Mr. Shlomo Shimony from the DSP lab at Tel-Aviv University for letting us use their prototype designed for the DSK C5402.

Tools: The first meeting is dedicated to an introduction to the basic tools. The *code composer studio*® is presented and a simple example is discussed. This experiment is based on TI documentation [9] and does not involve Simulink. Basic hardware components are explored, including the coder-decoder (CODEC) circuit, which serves as the analog-to-digital and digital-to-analog interface. To demonstrate the basic abilities of the DSP board a sine wave generator is implemented. Finally, the *general extension language* (GEL) scripts, enabling the programmer to automate processes, are introduced .

Sampling theorem: The sampling theorem is reviewed in the second experiment. We explore many important aspects of sampling, e.g Nyquist theorem, perfect and non-perfect reconstruction filters, aliasing effects, the use of anti-aliasing filters, decimation and interpolation, and quantization. We conclude this experiment by a design task, in which the students are asked to simulate and implement a system which demonstrates the speech reverberation phenomenon.

Filter design: Filter design tools are introduced in the third experiment. The Matlab *filter design and analysis* tool (FDATool [8]) is explored. *Finite impulse response* (FIR) with *generalized linear phase* (GLP) as well as *infinite impulse response* (IIR) filters are designed. The filters characteristics, e.g. attenuation ability versus the filter order, the delay, the frequency and phase responses, and the zero-pole constellation, are demonstrated. Two special filters, i.e. differentiator, and notch are designed and tested using real signals.

Filter implementation: The fourth experiment is dedicated again to the CCS, this time dealing with various filter implementations, e.g. IIR direct forms I and II. The designed filter coefficients are exported to the DSP hardware and the filters are implemented using C programming language.

Communications: Basic aspects of communications, e.g. *amplitude modulation* (AM), *double side band* (DSB), synchronous and asynchronous detection, and *frequency division multiplexing* (FDM), are introduced in the fifth experiment, as an example of the Fourier transform characteristics.

Adaptive filtering: The sixth experiment is dedicated to adaptive filtering. The *least mean squares* (LMS), the *recursive least squares* (RLS), and Kalman algorithms are introduced. Noise cancellation using Widrows method [10] is demonstrated.

Spectral Analysis: The seventh experiment concentrates on fast Fourier transform (FFT) procedures for spectral analysis. In particular, the detection of DTMF tones is introduced and a detection method using the Goertzel algorithm is implemented.

3. SAMPLE EXPERIMENTS

Two of the experiments listed in Sec. 2 are detailed now, namely the *sampling theorem* and *adaptive filtering* experiments. Each experiment comprises several problems represented by a Simulink block diagram (a file with extension .mdl). The relation to the hardware is represented in all diagrams by the two blocks C6713 DSK ADC and DAC, symbolizing the CODEC input and output streams, respectively. The sampling rate, number of bits and the source (either line-in or microphone-in) can be set by the user. Determining the sampling rate automatically selects the anti-aliasing filter cutoff frequency.

3.1 Sampling Theorem

Several topics are covered by this experiment: sampling and reconstruction, Nyquist theorem, aliasing and anti-aliasing filters, and quantization effects. Finally a design example of a discrete-time system, which implements continuous-time domain requirements, is given.

3.1.1 Sampling and reconstruction

In the first, simple experiment, the external signal generator is used to produce a sine wave with variable frequency. The signal is sampled and reconstructed by the DSP board. The CODEC output is examined on the oscilloscope. The students are expected to verify that any frequency above half the sampling frequency is attenuated. No aliasing is encountered in this experiment.

3.1.2 *Quantization effects*

The issue of finite-word representation of samples is of crucial importance in real-time applications. It is easy to determine the amount of distortion as a function of the number of bits per sample, using the system depicted in Fig. 3 The quantization effects are examined by using a pure sine wave as well as a speech signal. The stairs-like nature of the quantized sine wave is observed on the oscilloscope. The quantized speech quality is subjectively evaluated using the headset.

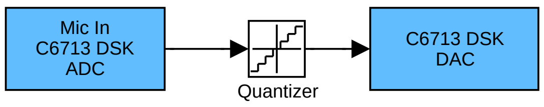

Figure 3: Quantization effect.

3.1.3 *Aliasing phenomenon and anti-aliasing filters*

Aliasing effects are examined using the up-sample and down-sample system depicted in Fig. 4 (Top). Let $s[n] = \cos[\omega_0 n]$ be a sampled input signal. After a consecutive application of the down-sampler and up-sampler the output signal will be given by $s'[n] = \frac{1}{2}\cos[\omega_0 n] + \frac{1}{2}\cos[(\pi - \omega_0)n]$. By using the oscilloscope in the spectral analysis mode, the mirror signal can be easily encountered and its frequency can be determined.

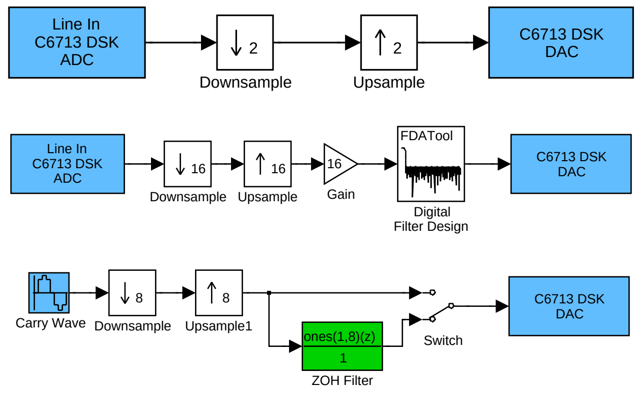

Figure 4: Decimation and interpolation. Top: aliasing phenomenon. Middle: perfect anti-aliasing filter. Bottom: ZOH anti-aliasing filter.

The system depicted in Fig. 4 (Middle) is then used to examine the concept of anti-aliasing filters. Matlab FDATool toolbox [8] is used for designing (an almost perfect) reconstruction filter. The students should verify, using an input sine wave from the signal generator, that the maximal output frequency is constrained to half of the sampling frequency divided by 16. Any higher frequency is aliased into the designated frequency band.

Finally, the reconstruction filter is replaced by the simpler zero-order-hold (ZOH) filter, as depicted in Fig. 4 (Bottom). The stairs-like nature of the output signal is evident.

3.1.4 *Discrete time system*

In this experiment the students are asked to assemble the knowledge they already acquired to synthesize a discrete time system that can simulate a known physical phenomenon. We chose to demonstrate the reverberation phenomenon, which is often encountered in acoustic environments where the speech source is reflected several times by objects in the room before being picked-up by the microphone. The time difference between reflections and the level of each of the reflections can be determined from the room geometry. The signal is modelled as $y(t) = s(t) + \alpha_1 s(t-\tau_1) + \alpha_2 s(t-\tau_2) + \alpha_3 s(t-\tau_3)$. The students should implement a Simulink block (as depicted in Fig. 5), simulating the above continuous-time model, and determine the involved parameters. The reverberation phenomenon could be demonstrated by hearing the DSP box output signal.

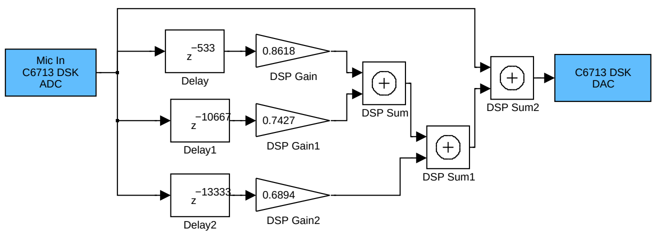

Figure 5: Simulation of reverberation pheomenon.

3.2 Adaptive Filtering

In this experiment a frequently encountered complicated real-life scenario, the interference cancellation problem, is treated. Students are expected to integrate concepts from the advanced statistical signal processing discipline with a profound understanding of the physical phenomenon to design a working solution.

3.2.1 *Echo Cancellation problem*

A very basic problem in adaptive filtering is the problem of acoustic echo cancellation. Two speakers, denoted as the near-end talker and the far-end talker, conduct an open speaker-phone conversation. The remote talker's voice, emerging from the loudspeaker of the near-end party, is received by the respective microphone and transmitted back to its originating party. Thus, the near-end party is transmitting the desired near-end talker contaminated by the far-end echo. The goal of the echo cancellation algorithm is to mitigate any echo component received by the microphone, thus avoiding its retransmission to the other party.

Let $s[n]$ be the near-end talker and $e[n]$ the far-end echo signal transmitted by the near-end loudspeaker. The signal received by the microphone at the near-end side is thus $y[n] = s[n] + e[n] * a_n[n]$, where a_n[n] is the time-varying acoustic path and $*$ denotes the convolution operation. Define also $v[n] = e[n] * a_n[n]$. To avoid transmission of the echo signal, an echo cancellation system, depicted in Fig. 6 is applied at the near-end side. The adaptive filter is applied to the reference (loudspeaker signal, $e[n]$) and its output is subtracted from the received signal. Constraining the adaptive filter to a FIR structure of order L, the estimated echo-free signal is given by:

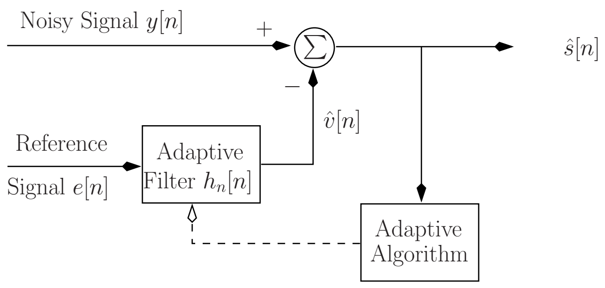

Figure 6: The basic echo cancellation system.

$$\hat{s}[n] = y[n] - \hat{v}[n] = y[n] - \sum_{l=0}^{L} h_n[l]e[n-l]. \tag{1}$$

Define the echo canceller coefficients and the echo state-vector:

$$\mathbf{h}_n^T = [\, h_n[0] \;\; h_n[1] \;\; \cdots \;\; h_n[L] \,] \quad (2)$$
$$\mathbf{e}_n^T = [\, e[n] \;\; e[n-1] \;\; \cdots \;\; e[n-L] \,]. \quad (3)$$

Widrow *et al.* [10] formulated the LMS algorithm for obtaining the minimum output power. It can be shown that the minimum output power is obtained when $\hat{s}[n] = s[n]$. It is well known that the normalized version of the LMS algorithm for updating the filter coefficients

$$\mathbf{h}_n = \mathbf{h}_{n-1} + \frac{\mu}{\mathbf{e}_n^T \mathbf{e}_n + \delta} \mathbf{e}_n (y[n] - \mathbf{e}_n^T \mathbf{h}_n) \quad (4)$$

tracks the desired solution, provided that the step-size μ fulfills $0 < \mu < 2$. δ is a small number inserted in the denominator to avoid division by zero.

The goal of the experiment is to gain insight on the behavior of the adaptive algorithm, namely to test the influence of the step-size, the noise characteristics and the acoustic path transfer function on the convergence rate and on the obtained performance. For this part of the experiment the Matlab Demo [8] is used.

3.2.2 *Single Microphone Speech Enhancement Algorithm*

The LMS algorithm described above can be used, with certain modifications, as a single microphone speech enhancement algorithm that can suppress periodic or almost periodic interference signals. In the single microphone case no reference signal is available and the algorithm should use the different statistical properties of the speech and noise signals instead. The correlation-time (the time lag before the correlation function vanishes) of the speech signal is quite short. On the other hand, a periodic noise has a very long correlation-time (almost a “predictable” signal). Hence, we define the primary input signal to be a delayed version of the measurement $y[n]$ and the reference signal to be the measurement itself. We then apply the following modified normalized LMS algorithm

$$\mathbf{h}_n = \mathbf{h}_{n-1} + \frac{\mu}{\mathbf{y}_n^T \mathbf{y}_n + \delta} \mathbf{y}_n (y[n-\Delta] - \mathbf{y}_n^T \mathbf{h}_n) \quad (5)$$

where,

$$\mathbf{y}_n^T = [\, y[n] \;\; y[n-1] \;\; \cdots \;\; y[n-L] \,]$$

and Δ is a controllable delay. The system is fully described in Fig. 7. Note, that the algorithm aims at minimizing the output power of the

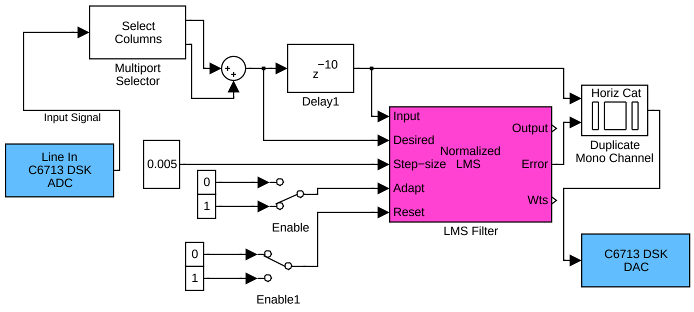

Figure 7: Single microphone speech enhancement algorithm.

system. This is done by eliminating any component at the measurement signal that is correlated with the reference signal. Since only the periodic interference signal can be predicted by the reference signal, the algorithm tends to mitigate the periodic noise while imposing only minor distortion on the speech signal.

In the experiment, the students are given the system in Fig. 7 without the delay branch. Based on the theoretical background the students are expected to reveal the missing parts themselves. Then, the effect of modifying the step-size, the filter length, and the delay value on the convergence rate and obtainable performance is tested. The noise signal is switched between a sine wave and a square wave. It should be verified that a shorter filter length is required for obtaining the desired cancellation while using the sine wave. Unofficial hearing tests should prove that the system is working properly: the periodic signal is almost cancelled whereas the speech maintains its natural quality.

4. CONCLUSIONS

We presented a unique approach of designing a DSP laboratory which uses Simulink for top-level design of DSP concepts, and, simultaneously, gives the students the opportunity to experience real-time hardware implementation. This combined approach focuses on DSP problems and concepts rather than programming, and enables the illustration of sophisticated algorithms. One of the major attributes of the proposed lab is its flexibility. Indeed, our concept was adopted and the experiments were easily converted to establish the *advanced communications* lab in our school.

The DSP lab is a mandatory course for the students who elect DSP as one of their major disciplines. It comprises a comprehensive set of seven experiments, covering most of the undergraduate DSP curriculum. Students who wish to enhance their skills in real-time programming and to specialize in DSP algorithms are given the opportunity to undertake, in conjunction with the educational lab, a graduation project in the DSP lab.

The DSP educational lab is already active for two consecutive academic years and is very well appreciated by the students, as a major contributor to their DSP education.

REFERENCES

[1] V. K. Ingle and J. G. Proakis, “A DSP Course based on Lecture/Lab Integration,” *IEEE Signal Processing Magazine*, vol. 9, no. 4, pp. 25–29, Oct. 1992.

[2] R. Chassaing, W. Anakwa, and A. Richardson, “Real-Time Digital Signal Processing in Education,” in *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Minneapolis, USA, Apr. 1993, IEEE, vol. 1, pp. 28–31.

[3] L. Shuo, L. Teng, L. Fanghui, and T. Liyu, “Real-time Signal Waveform Library and Processing System Based onTMS320C5x DSP Starter Kit,” in *International Conference on Signal Processing (ICSP)*, Beijing, China, Oct. 1998, vol. 2, pp. 1662–1665.

[4] W.-S. Gan, “Teaching and Learning the Hows and Whys of Real-Time Digital Signal Processing,” *IEEE Trans. on Education*, vol. 45, no. 4, pp. 336–343, Nov. 2002.

[5] Jacob Fainguelernt and Arie Yeredor, “Bridging the gap between dsp theory and real-time implementation,” in *European DSP Education and Research Symposium (EDERS2004)*, Birmingham, United Kingdom, 2004, IEEE and TI.

[6] W.-S. Gan, Y.-K. Chong, W. Gong, and W.-T. Tan, “Rapid Prototyping System for Teaching Real-Time Digital Signal Processing,” *IEEE Trans. on Education*, vol. 43, no. 1, pp. 19–24, Feb. 2000.

[7] W. S. Gan and S.M. Kuo, “Transition from Simulink to MATLAB in Real-Time Digital Signal Processing Education,” *The International Journal of Engineering Education*, vol. 21, no. 4, 2005, Special issue on MATLAB and Simulink in Engineering Education.

[8] The Mathworks Inc., *Matlab and Simulink User’s Guide*, 2005.

[9] Texas Instruments Inc., *TMS320C6416/C6713 DSK One-Day Workshop - Student Guide*, 3.1 edition, Aug. 2003.

[10] B. Widrow *et al.*, “Adaptive Noise Cancelling: Principals and Applications,” *Proceeding of the IEEE*, vol. 63, no. 12, pp. 1692–1716, Dec. 1975.