122nd ASEE Annual Conference & Exposition
June 14 - 17, 2015
Seattle, WA
Paper ID #12241

Real-Time Digital Signal Processing Demonstration Platform

Dr. Joseph P Hoffbeck, University of Portland

Joseph P. Hoffbeck (hoffbeck@up.edu) is an Associate Professor of Electrical Engineering at the University of Portland in Portland, Oregon. He has a Ph.D. from Purdue University, West Lafayette, Indiana. He previously worked with digital cell phone systems at Lucent Technologies (formerly AT&T Bell Labs) in Whippany, New Jersey. His technical interests include communication systems, digital signal processing, and remote sensing.

©American Society for Engineering Education, 2015

Real-Time Digital Signal Processing Demonstration Platform

Abstract

In order to demonstrate various digital signal processing (DSP) algorithms to students or potential students, a program was developed that runs in real-time on low cost, commercially available hardware. The program includes several common DSP algorithms such as lowpass filter, highpass filter, echo, reverb, quantization, aliasing, simple speech recognition, and fast Fourier transform (FFT). The program allows the user to easily switch between algorithms, to adjust the parameters of the algorithms, and to immediately hear the results. The demonstration hardware consists of the TMS320C5515 eZdsp USB Stick, a powered microphone, an audio source such as an MP3 player or cellphone, and speakers. Undergraduate electrical engineering students were shown the demonstration and were surveyed to determine which algorithms they found most interesting. The C language source code for the software is available from the author for free, so this program can be modified by instructors who wish to make their own demonstrations or used as a convenient starting point for student projects.

Introduction

The material in a DSP course is often highly theoretical and mathematical, and so it is useful to connect the theory to real-world applications with laboratory experiments, simulations, or demonstrations. Fortunately, there are many interesting applications of DSP that can help captivate students and motivate them to learn the theoretical material. Perhaps the best way to expose students to the applications of DSP is with a laboratory course, but at some institutions a full laboratory course in DSP is not feasible due to time, space, and funding constraints. Alternatives to laboratories include projects, simulations, and demonstrations. Although many DSP algorithms can be demonstrated using offline (not real-time) processing, some demonstrations are just more compelling if they operate in real-time. A real-time demonstration with audio signals, for example, can be more interesting because the user can use his/her own voice as the input and hear the results immediately.

Many authors have explored various ways to demonstrate DSP concepts and algorithms. Some have used software such as MATLAB[1, 2, 3, 4, 5], LabView[4, 5, 6, 7], J-DSP[8], or DirectX[9] that runs on a personal computer (PC). Some have incorporated commercially available DSP boards including the TMS320C6713 DSK[10, 11] (about $400), OMAPL138 Low Cost Development Kit[12, 13] (about $200), Logic PD Zoom OMAP-L138 Experimenters Kit (about $500), and DSK5510[14] (about $400). Some have combined DSP boards with software such as MATLAB/Simulink[15, 16], winDSK[17, 18, 19, 20], J-DSP[21], and LabView[22]. Courses in DSP have also been taught using FPGA boards[23, 24] and microcontrollers[25, 26].

This paper describes a real-time demonstration platform based on the TMS320C5515 eZdsp USB Stick Development Tool. This DSP board, which is available for about $80, is less expensive than any of the other DSP boards listed above, yet it has all the features necessary to conveniently demonstrate DSP with audio signals, including a high quality audio interface, a small graphical display, two buttons for user input, and five LEDs. Compared to the more expensive boards, this board does have some disadvantages, including a slower processor with no floating-point hardware and less on-board memory, but it is more than adequate for demonstrating audio applications in real-time. Furthermore, the source code for the software is available from the author for free, which may be helpful for others who wish to develop their own demonstrations or for student projects.

Hardware

Manipulating audio signals is an excellent way to illustrate DSP concepts. This approach allows students to directly experience the effect of various algorithms and to hear the effect of using different parameters values. As shown in Figure 1, the hardware required for the audio demonstrations is fairly simple. The audio source can be any device that has line level or headphone output, such as MP3 player, cellphone, or PC. The output of the DSP board can be monitored with headphones or inexpensive computer speakers. The PC is used to download the program and to provide power for the DSP board. However, if the program is written to the flash memory on the DSP board, then the PC could be replaced with a USB power supply.

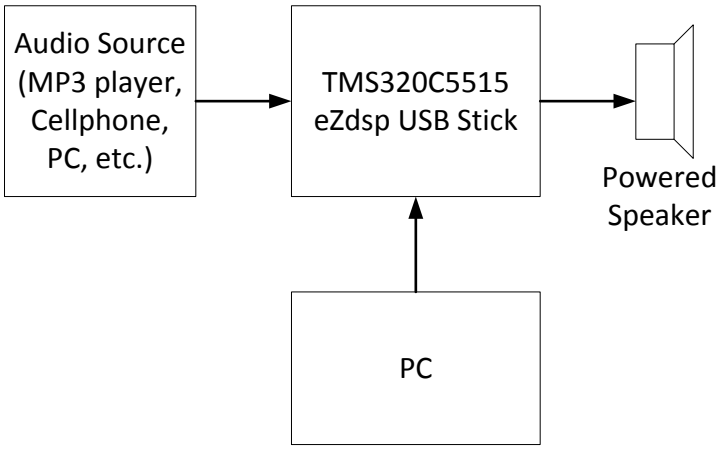

Figure 1: Hardware Block Diagram

The TMS320C5515 eZdsp USB Stick Development Tool, which is shown in Figure 2, was chosen mainly for its low cost and high quality audio interface. It is based on the Texas Instruments TMS320C5515 fixed-point digital signal processor that provides 240 MIPS, 320 KB of RAM, and a hardware FFT accelerator. The line level audio input and headphone/line level audio output are provided by the TLV320AIC3204 stereo audio codec. The codec's analog-to-

digital converters (ADC) have a signal-to-noise ratio (SNR) of 93 dB, the digital-to-analog converters (DAC) have an SNR of 100 dB, and the maximum sampling rate is 192 kHz.

Other features of the development tool that make the demonstration much more convenient are the two push button switches that can be used to change between various algorithms, the small 96x16 pixel display which can display two lines with up to 19 characters each as well as simple graphics, and five LEDs that can be used to provide feedback to the user. The development tool includes everything that is needed to write and debug C language (and assembly language) programs including an embedded XDS100 emulator and Code Composer Studio compiler/debugger. The development tool is physically small (2.65 by 3.35 inches), and it connects to a PC through a USB port, which also supplies power to the device.

One useful feature that is missing from the development tool is a microphone input. However, a battery-powered microphone can be used directly with the line level input, or a regular microphone can be used in conjunction with a preamp.

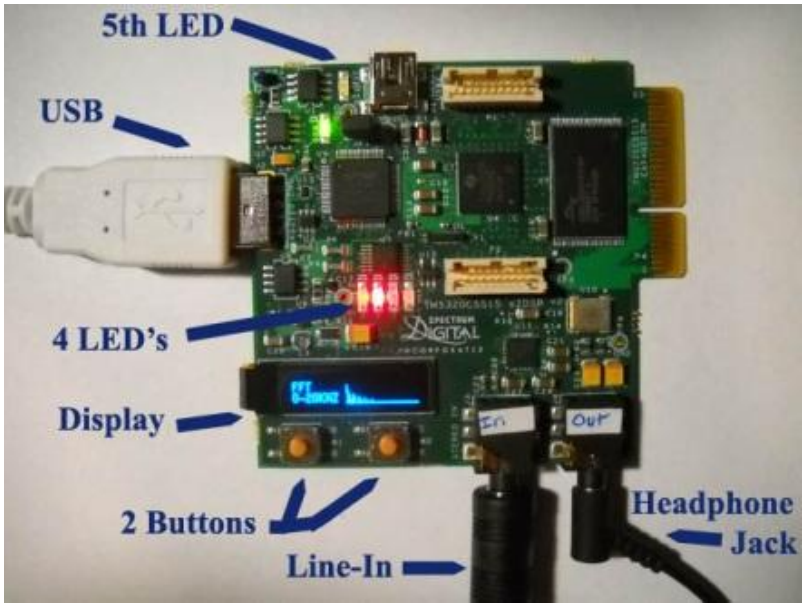

Figure 2: TMS320C5515 eZdsp USB Stick Development Tool

Software

A program was written to demonstrate various common DSP algorithms in real-time by processing audio signals. When it starts up, the program initializes the board, sets the sampling rate on the codec to 48 kHz, and begins the first algorithm which is pass-through. Pass-through is used to verify that the audio setup is working properly and to listen to the audio input signal. While in pass-through mode, four of the LEDs on the board are used to display a simple volume unit (VU) meter, where a higher input level (amplitude) leads to more of the LEDs being lit. It is important to have a convenient way to set the input level because if the input level is too high the signal will be clipped which causes unwanted distortion, and if the signal is too low, the quantization noise from the ADC can be noticeable. The VU meter is displayed for all of the algorithms except for speech recognition where the LEDs are used for other purposes.

The program monitors the push button switches, and if the right button is pressed the program switches to the next algorithm and updates the display to show the name of the current algorithm. If the left button is pressed, the program returns to the previous algorithm (and updates the display). Some of the algorithms have user-controlled parameters which can be changed by pressing both switches simultaneously, which rotates between each of the parameters for that algorithm and the mode that allows the user to change the algorithm. The pass-through algorithm, for example, allows the user to change the gain on the ADC's to accommodate level differences between a microphone and other devices.

After pass-through, the next algorithm is a 6^{th} order infinite impulse response (IIR) Butterworth lowpass filter (LPF) with a cutoff frequency of about 1 kHz. An excellent way to demonstrate this filter is to use music with some low frequency instruments and some high frequency instruments, an example of which is "Take Five" by Dave Brubeck. Other signals that are useful for the demonstration are white noise, chirp signal (frequency-swept sinusoid), and a series of short tones with increasing frequency. A MATLAB program was used to generate these signals and save them in sound files, which were then played by an MP3 player for the demonstration.

Following the LPF is a 100^{th} order finite impulse response (FIR) highpass filter (HPF) with a cutoff frequency of about 1 kHz. The HPF can best be demonstrated using the same input signals as the LPF (see above).

The next algorithm generates echo, which demonstrates one of the simplest FIR filters as shown in Figure 3. The program allows the user to change the delay. The best way to demonstrate echo is to use the user's voice with a powered microphone or a regular microphone with a preamp.

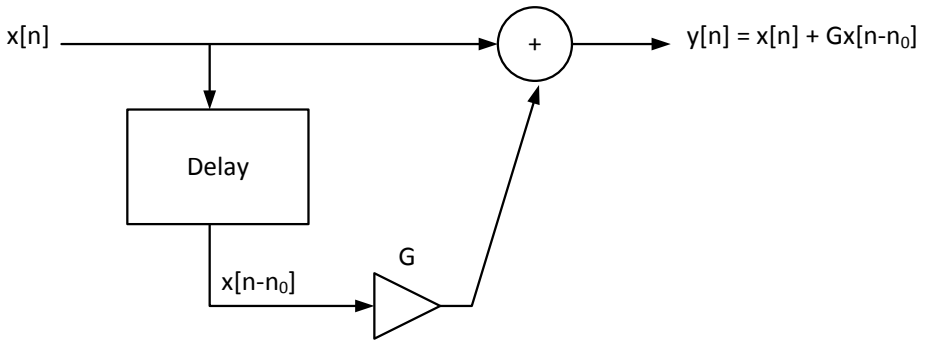

Figure 3: Block Diagram of Echo

The simple reverb algorithm shown in Figure 4 was also implemented. It too is best demonstrated using a microphone. The program allows the user to hear the effect of changing both the delay and the gain (G) to get various audio effects.

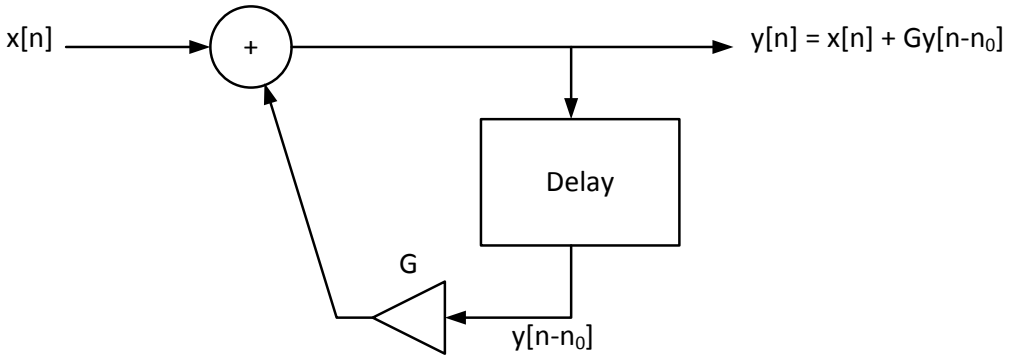

Figure 4: Block Diagram of Reverb

When signals are converted from analog to digital, one of the operations is quantization which causes noise to be added to the signal. In order to demonstrate the effect of quantization, the next algorithm quantizes the input signal to a user-defined number of bits by setting the least significant bits to zeros. The program allows the user to adjust the number of bits from 1 to 16. One way to demonstrate quantization noise is to gradually reduce the number of bits from 16 down to one while listening to the output. It is surprising to most people that speech and music are still understandable even when quantized to just 1 bit.

In addition to quantization, analog-to-digital conversion also requires sampling. After sampling, any frequency components that are above one half of the sampling frequency (fs/2) are folded down to a frequency below one half of the sampling frequency. This effect causes distortion called aliasing. To demonstrate aliasing, the program downsamples the input signal to a sampling rate of just 2 kHz, which causes frequency components above 1 kHz to be folded down to a frequency below 1 kHz. The program then upsamples and filters the signal before sending

to the codec (see Figure 5). Using the chirp signal is an excellent way to demonstrate aliasing. As the pitch of the input chirp signal increases, aliasing causes the output pitch to alternately increase and decrease.

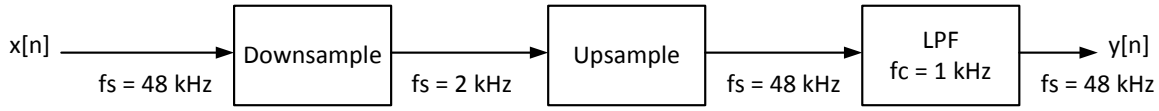

Figure 5: Block Diagram for Aliasing

A simple speaker-independent voice recognition algorithm that attempts to recognize the words "Yes" and "No" was also implemented. The algorithm uses energy-based thresholds to determine the start and end of a word and lights up the green LED to indicate that the input has a high enough level to begin processing. When the end of the word occurs, the algorithm turns off the green LED, and decides if the word was "Yes" or "No" based on the ratio of low frequency power to high frequency power determined from the fast Fourier transform (FFT). If the algorithm determines the word was "Yes", it turns on the blue LED and prints "Yes" to the display. If the algorithm determines the word was "No", then it turns on the red LED and prints "No" to the display. This algorithm is not as reliable as more sophisticated ones, but is it simple to understand and implement and works reasonably well. The best way to demonstrate this algorithm is to use the microphone to allow the user to say "Yes" and "No" and see if the algorithm correctly identifies the word.

The last algorithm demonstrates the FFT by displaying a graph of the magnitude of the FFT on the vertical axis versus frequency on the horizontal axis as shown in Figure 6. The graph updates in real-time, so it shows the current short-time spectrum of the input signal. If the input signal is the chirp signal, for example, the display will show a spike that slowly moves to the right as the frequency of the chirp signal increases. If the input signal is music, the graph continuously changes to reflect the current sounds, which is interesting to watch.

Figure 6: Close-up of Display for FFT

The demonstration program was based on a publically available example program that initializes the board and the codec, and sets up an efficient double buffering scheme using the direct memory access (DMA) controller[27]. The functions that write to the display were based on publically available examples[28], but were re-written so that the program can write to the display without violating the real-time schedule for processing the audio signal. As a precaution, the fifth LED on the board is used to warn the user if the program ever misses the real-time schedule, which would cause distortion in the output signal. The C language program that implements the demonstration is available for free by sending email to hoffbeck@up.edu.

Use of the Demonstration Platform

The demonstration platform was designed to be used in the author's MATLAB-based DSP lecture course where there is no laboratory component. Although the students in this course write MATLAB programs to process recorded and generated audio signals, they do not see the algorithms run in real-time. So the demonstration allows them to experience the real-time version of the algorithms and see them run on hardware that is typical for commercial products.

The demonstration program could also be used to try to increase interest in the field of DSP. To this end, the demonstration was shown to a small number of undergraduate students at the author's university who had not yet taken the DSP course. The students were then surveyed to determine which algorithms they found most interesting. The survey had the students rate each algorithm on a scale of 1 to 5, where 1 was "Not at all Interesting" and 5 was "Very Interesting". The average score for each algorithm is shown in Figure 7 for the nine students who returned the survey.

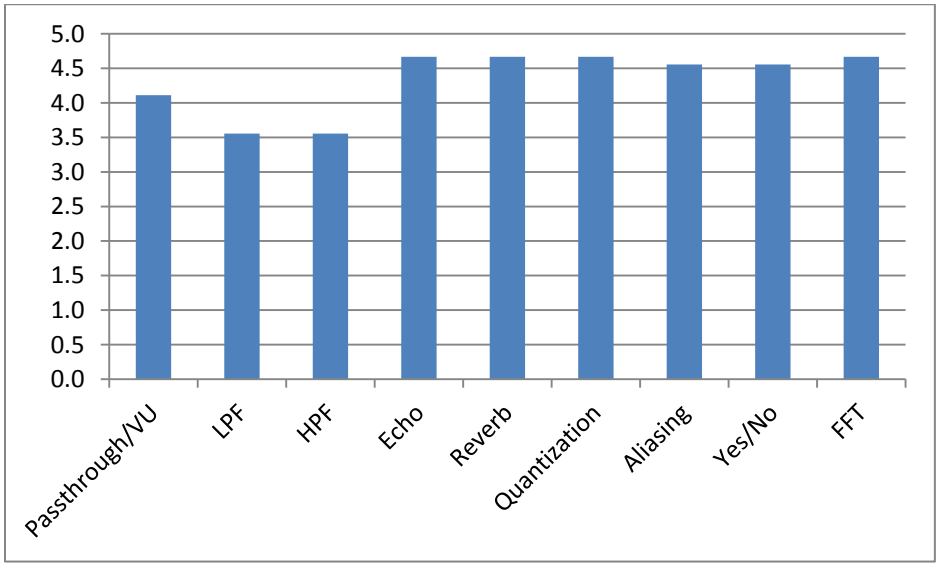

Figure 7: Results of Survey

There was a four-way tie for the top average student rating with echo, reverb, quantization, and FFT all receiving an average score of 4.7 (out of 5). Aliasing and Yes/No were next with 4.6, followed by passthrough/VU with 4.1, and LPF and HPF trailing behind at 3.6.

The survey also had a space for written comments, and three of the five students who wrote comments said that the demonstrations were interesting or cool. One student commented that he/she had seen LPF and HPF in classes before and so rated those lower, and one wrote that he/she thought that DSP would be a great class. The survey suggests that the students found the demonstrations to be interesting, but since the sample size was small, any conclusions would be preliminary.

Conclusion

A method for demonstrating DSP algorithms and concepts was described. The program runs in real-time on inexpensive, commercially available hardware. Since the software is freely available from the author, the demonstration can be modified or additional algorithms can be added. The demonstration was shown to some undergraduate electrical engineering students, and they found the demonstrations to be interesting.

References

1. Pamela Bhatti, Jessica Falcone, and James McClellan, "The Coding of Sound by a Cochlear Prosthesis: An Introductory Signal Processing Lab", 2010 ASEE Annual Conference & Exposition.

2. Joseph P. Hoffbeck, "Enhance your DSP Course with these Interesting Projects", 2012 ASEE Annual Conference & Exposition.

3. Joseph P. Hoffbeck, "Using Real RF Signals Such as FM Radio to Teach Concepts in Communication Systems", 2008 ASEE Annual Conference & Exposition.

4. Wayne Padgett, "Fixed-Point DSP Implementation: Advanced Signal Processing Topics and Conceptual Learning", 2007 ASEE Annual Conference & Exposition.

5. Mark Yoder and Bruce Black, "A Study of Graphical vs. Textual Programming for Teaching DSP", 2006 ASEE Annual Conference & Exposition.

6. Murat Tanyel, "Putting Bells & Whistles on DSP Toolkit of LabView", 2011 ASEE Annual Conference & Exposition.

7. Jean Jiang and Li Tan, "Teaching Speech and Audio Processing Implementations using LabView Program and DAQ Boards", 2013 ASEE Annual Conference & Exposition.

8. Venkatraman Atti and Andreas Spanias, "The Java-DSP (J-DSP) Project-From the Prototype to the Full Implementation and Dissemination", 2005 ASEE Annual Conference & Exposition.

9. Peter Goodmann, "Using Microsoft DirectX in a DSP Laboratory", 2005 ASEE Annual Conference & Exposition.

10. Bruce Dunne, "DSP-Based Low Cost Digital Communications Laboratory", 2006 ASEE Annual Conference & Exposition.

11. Zhibin Tan, William H. Blanton, and Qianru Zhang, "Real-time EEG Signal Processing Based on TI's TMS320C6713 DSK", 2013 ASEE Annual Conference & Exposition.

12. Felipe L. Carvalho and Ravi T. Shankar, "Biomedical Signal Processing: Designing an Engineering Laboratory Course Using Low-Cost Hardware and Software", 2014 ASEE Annual Conference & Exposition.

13. Cameron Wright, Thad Welch, and Michael Morrow, "RT-DSP Using 'See Through'", 2014 ASEE Annual Conference & Exposition.

14. Buket Barkana, "A Graduate Level Course: Audio Processing Laboratory", 2010 ASEE Annual Conference & Exposition.

15. Tim Lin, Saeed Monemi, and Zekeriya Aliyazicioglu, "Interactive Learning Discrete Time Signals and Systems with MATLAB and TI DSK6713 DSP Kit", 2007 ASEE Annual Conference & Exposition.

16. Lisa Huettel, "Integration of a DSP Hardware Based Laboratory into an Introductory Signals and Systems Course", 2006 ASEE Annual Conference & Exposition.

17. Michael G. Morrow, Cameron H. G. Wright, and Thad B. Welch, "Old Tricks for a New Dog: An Innovative Software Tool for Teaching Real-Time DSP on a New Hardware Platform", 2011 ASEE Annual Conference & Exposition.

18. Michael G. Morrow, Cameron H. G. Wright, and Thad B. Welch, "An Inexpensive Approach for Teaching Adaptive Filters using Real-Time DSP on a New Hardware Platform", 2013 ASEE Annual Conference & Exposition.

19. Gerald Vineyard, Thad Welch, Cameron Wright, and Michael Morrow, "A Hardware Approach to Teaching FSK", 2007 ASEE Annual Conference & Exposition.

20. Cameron Wright, Thad Welch, Mark Allie, and Michael Morrow, "Using Real-Time DSP to Enhance Student Retention and Engineering Outreach Efforts", 2008 ASEE Annual Conference & Exposition.

21. Ashwinn Natarajan, Andreas Spanias, Chih-Wei Huang, and Rony Ferzli, "Interfacing J-DSP with a TI DSK for use in a Signal Processing Class", 2006 ASEE Annual Conference & Exposition.

22. Douglas Williams and Arif Uluagac, "Building Hardware-Based Low-Cost Experimental DSP Learning Modules", 2008 ASEE Annual Conference & Exposition.

23. Tyson Hall and David Anderson, "Teaching Hardware Design of Fixed Point Digital Signal Processing Systems", 2007 ASEE Annual Conference & Exposition.

24. James Kang and Alan Felzer, "A Digital Signal Processing Laboratory Course Using Field Programmable Gate Array Boards", 2005 ASEE Annual Conference & Exposition.

25. Dick Blandford, "DSP on Generic Machines", 2006 ASEE Annual Conference & Exposition.

26. Li Tan and Jean Jiang, "Teaching Digital Filter Implementations Using the 68HC12 Microcontroller", 2011 ASEE Annual Conference & Exposition.

27. https://code.google.com/p/c5505-ezdsp/

28. http://support.spectrumdigital.com/boards/usbstk5515/reva/