11th INTERNATIONAL SYMPOSIUM on
POWER ELECTRONICS - Ee 2001
XI Međunarodni simpozijum Energetska elektronika – Ee 2001
NOVI SAD, YUGOSLAVIA, OCT. 31 – NOV. 2, 2001

Practical aspects of teaching DSP in laboratory

Vladan Jevremović, Slobodan Vukosavić*
Faculty of Engineering, University of Novi Sad, Novi Sad, Yugoslavia
*Faculty of Electrical Engineering, University of Belgrade, Belgrade, Yugoslavia

Abstract: ***One of the challenges for instructors teaching DSP classes is to achieve a balance in teaching important theoretical concepts and providing practical exposure to DSP hardware. This paper deals with the design of FIR filters by using Matlab/DSK GUIs that allow students to quickly explore DSP concepts in Matlab and then download their Matlab based designs to run on a certain DSK platform. This approach provides students with exposure to DSP hardware without requiring DSP programming. For instructors who wish to include DSP programming in their courses, having students develop their own Matlab/DSK GUI application can serve as an interesting motivational tool.***
Key Words: ***DSP, filter, laboratory, teaching***

1. INTRODUCTION

Teaching the DSP basics always, apart from strong theoretical background that should be presented to the students, involves a practical aspect – like implementing some classic digital signal processing and control blocks (filters, convolution, decimators, regulators, etc). The key to any successful hardware implementation is a quality software programming. Unfortunately, is often practice that not all students are familiar with assembler or C language suited for particular DSP platform. The question of many lecturers [1] remains the same: How to combine practical know-how with knowledge from textbooks? In order to answer this question, this paper presents MATLAB program for FIR filters implementation and testing on Texas Instruments DSK (development software kit) TMS320C50. The main reason why FIR filters have been chosen for this program, is that these filers are most common blocks in almost any discrete-time signal processing system.

2. PARAMETERS OF FIR FILTERS

The most essential feature of FIR (Finite Impulse Response) filters is, by definition in [2] - the finite length of the impulse response. Among many ways to design these types of filters, the design method based on the FTD and windowing seems the most appropriate for Matlab implementation. The starting point in this design method is the frequency response $\boldsymbol{H_d}(\boldsymbol{e}^{j\theta})$ that should be realized as closely as possible with a FIR filter. Frequency response can be classified. However, the ideal impulse response $\boldsymbol{h_d}[\boldsymbol{n}]$ obtained with an aid of IFTD can not be used for filter design, since:

1) $\boldsymbol{h_d}[\boldsymbol{n}]$ has very large or even infinite duration ('length'), and
2) $\boldsymbol{h_d}[\boldsymbol{n}]$ is non-causal (i.e. $\boldsymbol{h_d}[\boldsymbol{n}] \neq 0$ for $\boldsymbol{n} < 0$).

These facts bring us to the conclusion that it is necessary to:

1) limit the length of $\boldsymbol{h_d}[\boldsymbol{n}]$ to an acceptable (finite) number of $\boldsymbol{K}$ samples,
2) introduce sufficient shift ('delay') to obtain a causal impulse response.

Truncation of infinitely long impulse response of discrete filter and the introduction of the shift gives extra linear phase shift. This truncation is performed with window function $\boldsymbol{w}[\boldsymbol{n}]$ as multiplication with impulse response in time domain:

$$\boldsymbol{h}[\boldsymbol{n}] = \boldsymbol{h_d}[\boldsymbol{n}] \cdot \boldsymbol{w}[\boldsymbol{n}] \qquad (1)$$

or convolution of transfer functions in frequency domain:

$$\boldsymbol{H}(\boldsymbol{e}^{j\theta}) = \boldsymbol{H_d}(\boldsymbol{e}^{j\theta}) * \boldsymbol{W}(\boldsymbol{e}^{j\theta}) \qquad (2)$$

This process is known as 'windowing' and the result of windowing is determined by the choice of particular window function.

Finally, the following set of FIR filter parameters is formed:

1) sampling frequency (sampling rate) $\boldsymbol{f_s}$,
2) filter order $\boldsymbol{N}$,
3) type of filter (low-pass, high-pass, band-pass or band-stop),
4) cut-off frequency $\boldsymbol{f_c}$ (for low-pass and high-pass filters), or band-start and band-stop frequency ($\boldsymbol{f_1}$, $\boldsymbol{f_2}$),
5) type of window function (rectangular, Bartlett, Hanning, Hamming or Blackman window).

3. WHY MATLAB?

Among many questions, one imposes the most - why using Matlab? The problem of writing program for FIR filter design can be solved with the help of high-level languages like Visual Basic, Visual C++ or Delphi. But programming in these languages can give a few megabytes of software code and it is very time-consuming. Besides, design method with FTD and windowing requires numerical computation routines that are not a usual ready-made part of these software packages. Matlab is comprehensive tool for solving many mathematical and engineering problems including signal processing.

The concept of this program is based on the set of diverse toolboxes designed for wide spectra of mathematical and engineering problems. The Signal Processing Toolbox provides functions that support a range of filter design methodologies and it is very much suited for this purpose.

4. GRAPHICAL USER INTERFACE

A good graphical user interface should allow interactive viewing and analysis of signals and their spectra, but since this paper deals with the filter design capabilities, it is more like that following items are particularly interesting:

1) magnitude plot,
2) gain plot,
3) impulse response plot.

A "window based" FIR design GUI (for different plots) is presented in Fig. 1, 2 and 3. Once again, Matlab 5.3 which offers the easy-to-use toolbox for GUI design, helped building this interface. For improved clarity, the GUI developed here is relatively simple with only a few filter design options and no "user input" error checking features. The task is to develop a Matlab/DSK GUI that will allow a user to choose FIR filter design parameters described in previous paragraphs. After choosing the desired parameters, the user can "click on a pushbutton" that will load and run the filter design on the DSK.

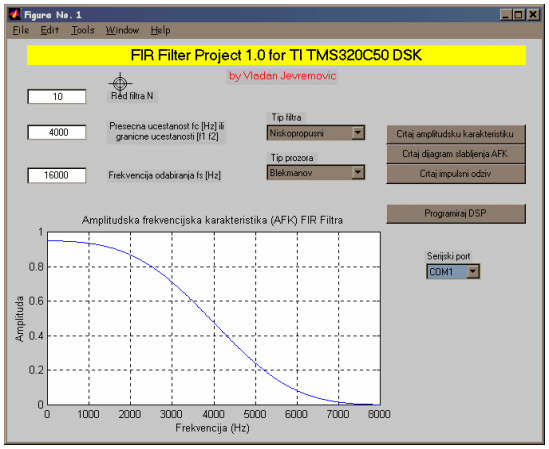

Fig. 1. ***Screen-capture of filter magnitude plot***

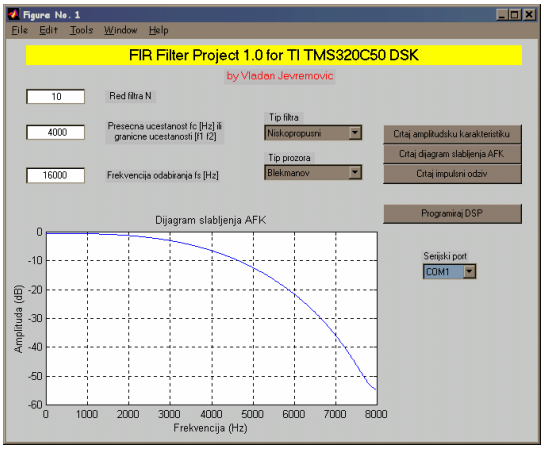

Fig. 2. ***Screen-capture of filter gain plot***

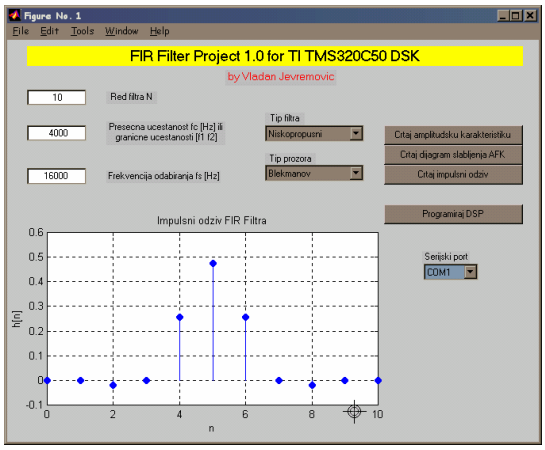

Fig. 3. ***Screen-capture of filter impulse response plot***

5. INTER-PROGRAM COMMUNICATION

After creating the graphical user interface the only remaining thing to be solved is communication between Matlab i.e. computer and DSK board. One potential solution is the use of CPORT shareware toolbox for Matlab, which contains commands for serial port communication. The only disadvantage of this solution is that it requires knowledge of data transfer protocol that is applied in data interchange between computer and DSK board. Our solution implies the use of ready-made assembler program for FIR filtering and header file with filter taps (coefficients) produced by Matlab filter design program.

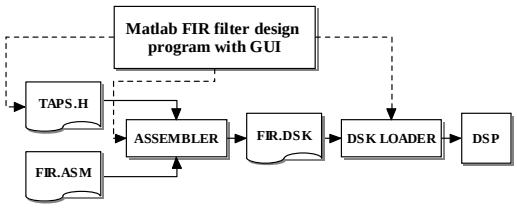

Fig. 4. ***Block-diagram of inter-program communication***

The sequence of actions that precedes the implementation of FIR filter is a specific example of 'inter-program' communication:

1) The user sets filter parameters with the help of program's GUI,
2) Matlab program then calculates filter coefficients and writes them into the header file named as 'TAPS.H'.
3) After that program invokes assembler to assemble file 'FIR.ASM' using the data from header file and creates file 'FIR.DSK'.
4) Finally, Matlab program calls a DSK-loader program which transfers file 'FIR.DSK' via serial cable and writes it into the flash memory of DSK board. The filtering can start from now on.

Inter-program communication is inexpensive way to form an advanced software development toolkit with average low-priced ($99) existing one, that can be compared with a high-priced evaluation boards ($1000-$3000). The difference and advantages are obvious in this case.

6. TEST ENVIRONMENT

A laboratory test environment should put student in position to quickly test large number of filters and see the results of his experiment at the very same time. Fig. 5 and 6 show two proposed solutions for the test environment.

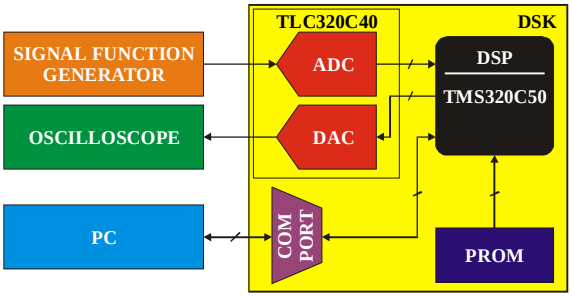

Fig. 5. ***Block-diagram of 'visual' test environment***

First test environment consists of personal computer with the corresponding software, DSK board, signal function generator and oscilloscope. Signal function generator generates complex periodical signals (square, triangular or sine waveform). DSK board containing CODEC (A/D and D/A converter) TLC320C40 [3] samples this signal and processes it through the FIR filter implemented in DSP. Filtering gives output signal which is visualized by the student with oscilloscope.

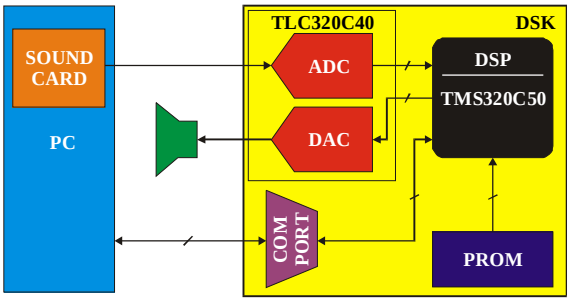

Fig. 6. ***Block-diagram of 'audio' test environment***

Second test environment is more audio-like and includes personal computer, DSK board, sound card (or any other adjustable sound source) and a loudspeaker. Instead of viewing output signal waveforms, student can now 'hear' the output signal and distinguish the difference in quality of input and output audio signals.

7. CONCLUSION

Note that this program does not require programming by the student. Our experiences have shown that once most students 'play' with this program a bit, become comfortable with the DSK, and start to see what the device can do, they want to learn how to program the DSK. This is how learning DSP can be both fun and creative for the student at the same time.

8. LITERATURE

[1] C. H. G. Wright, T. B. Welch, M. G. Morrow, "***Making DSP Fun for Students Using Matlab and the C31 DSK***", Texas Instruments DSPS Fest, Houston, TX, August 1999.

[2] W. M. van den Enden, N. A. M. Verhoeckx, "***Discrete-time signal processing – An introduction***", Prentice Hall, New York, 1989.

[3] Texas Instruments, "***TMS320C5x Application Guide***", Literature number: BPRA063, October 1997.