2006-797: REAL TIME SYSTEMS LABORATORY DEVELOPMENT: EXPERIMENTS FOCUSING ON A DUAL CORE PROCESSOR

Mukul Shirvaikar, University of Texas-Tyler

MUKUL SHIRVAIKAR received the Ph.D. degree in Electrical and Computer Engineering from the University of Tennessee in 1993. He is currently an Associate Professor of Electrical Engineering at the University of Texas at Tyler. He has also held positions at Texas Instruments and the University of West Florida. His research interests include real-time imaging, embedded systems and pattern recognition.

Mark Humphries, University of Texas-Tyler

MARK HUMPHRIES received his Master's in Electrical Engineering in 2005 from the University of Texas at Tyler, and is a practicing engineer at General Dynamics Inc. in Longview, Texas. He developed real-time systems labs for the OMAP platform. His interests include real-time imaging, open source software and spatial geometry optimization algorithms for multi-faceted cubes.

Leonardo Estevez, Texas Instruments Inc.

LEONARDO ESTEVEZ received the Ph.D. degree in Electrical and Computer Engineering from Texas A & M University in 1997. He is currently the OMAP Software Architecture and Requirements Manager at Texas Instruments. He has many national engineering awards including SHPE in 2001 and the HENAAC "Most Promising Engineer" award in 2002. He is an Associate Editor for the Journal of Real Time Imaging.

Real Time Systems Laboratory Development: Experiments Focusing on a Dual Core Processor

Abstract

This paper presents the laboratory curriculum developed for a senior-level elective course in Real Time Systems. The labs developed for this semester long course are aimed at providing a challenging experience to electrical and computer engineering students and exposing them to state-of-the-art tools from industry. The projects were developed on the OMAP 5912 starter kit module supplied by Texas Instruments (TI). The open multimedia architecture platform (OMAP) technology from TI consists mainly of dual-core processor chips. The OMAP 5912 chip has an ARM processor and a C55 digital signal processor (DSP) in the same package. The Linux kernel runs on the ARM processor and the DSP-BIOS kernel runs on the TI C55 DSP in tandem. The real time software development tools for this system are the Code Composer Studio integrated development environment (IDE) and the Monta Vista Linux environment. The platform is thus ideally suited to expose students to real time systems. The projects developed cover the following topics sequentially: introduction to the environment, real time operating systems, software development and application debugging. Some of the applications covered are: implementing a finite impulse response (FIR) filter and testing with audio, modifying the filter for different band pass characteristics, testing an audio codec and implementing an embedded web server. TI expects to disseminate the instructional resources developed and tested in this course to other universities and industry partners.

Introduction

Dual-core processors have recently entered mainstream computing in PC systems, and it is critical for students of computer engineering to be exposed to them early in their career. This paper extends past work [1], which presented the development of some introductory labs using TI's OMAP 5912 Starter Kit (OSK). The Real Time Systems senior elective course at the University of Texas at Tyler combines lectures along with an integrated lab. The students are required to have at least one course in structured programming, and a course or prior experience with the operation of microprocessors, but Linux experience is not required. The lecture portion of the course introduces students to real-time system concepts including, hard and soft deadlines, scheduling algorithms, inter-task communication and synchronization. The lab portion of the course reinforces these theoretical concepts and provides hands-on familiarity with software, hardware, and development tools essential for real time systems development professionals. For example, hard deadlines have to be met in order to complete dual-core data processing.

The initial version of the lab procedures utilized a digital signal processor (DSP) based system. Texas Instruments' (TI) introduction of a cost effective dual core processor (OMAP 5912) development system has enabled a new lab curricula. The OMAP 5912 gives us the flexibility of development for a general-purpose processor based (GPP) system combined with the processing efficiency of a DSP based system. The sheer number of new concepts introduced to students in this course reflects the reality they have to face in the new job market. In order to fit in as professionals the students have to be conversant with real-time, computer architecture, DSP,

networking and other concepts. It is not possible to address all these topics without using a relatively advanced and mature platform. Further, the complexity involved in successfully debugging real-time systems makes it very difficult to design lab sequences that do not "frustrate" the contemporary student who has a "point-and-click" mentality. The lab curriculum and flow was deliberately chosen to kindle student interest with applications involving media.

We have developed lab procedures that illustrate how development is accomplished for dual core systems specifically and for real time and embedded systems in general. A description of development tools that are used for each processor core and for inter-processor communication is provided. The students work with the Linux operating system, boot loader operation, cross compiler tools, debug tools, kernel uploading, flashing, and driver tools for dual processor tasks.

The lab "Introduction to Dual Core Development" is designed to give the students an introduction to using the LinuxDSP tools and DSP gateway for developing co-processing applications. They are also introduced to the concept of patching a kernel release for compatibility with the target platform.

Following the introductory lab, "Dual Core Implementation of a FIR Filter" gives the students an in depth look at the development process for an application that uses both processing cores to perform real-time filtering of an audio file. Exposed are the concepts of a FIR filter, inter-processor communication, threaded applications, and the representation of devices as files within the Linux operating system.

Laboratory Procedure to Introduce Dual Core Development

The objective of this lab is to introduce the students to tools that can be used for developing applications that can utilize both processing cores. The students are introduced to the TI supplied *LinuxDSP* tools [2] and the open source DSP Gateway [3] driver that allows inter-processor communication. The students are also introduced to the concept of patching a kernel for particular uses, the OMAP platform and DSP Gateway in this case. Figure 1 is a flowchart for the procedures that are followed in the lab.

This lab requires that the Linux kernel to be used for the board to be the 2.6.12 version. To be suitable for use by the board and this project, the 2.6.12 kernel hosted at kernel.org must be patched with the OMAP patch and the DSP Gateway 3.3 patch. The students are guided through this process and the kernel is subsequently built for the OSK board. Students are instructed to configure kernel features that support the DSP Gateway driver using the *menuconfig* utility. DSP Gateway provides the necessary support for the Linux kernel running on the ARM core to load and execute programs on the DSP core. Also provided with DSP Gateway is a modified version of the TI DSP/BIOS kernel to provide services for DSP tasks. The built kernel is uploaded and flashed into the board's memory to provide the necessary support when the board is booted up.

DSP Gateway allows the DSP core to appear as a set of device files to the Linux kernel running on the ARM core. This facilitates application development by presenting a familiar interface to Linux developers for the DSP. However, to load programs into the DSP memory and begin execution, the *dspctl* utility needs to be built for the ARM side Linux [4]. Once built, the students

have the tools necessary to load and execute DSP tasks and present the interface for Linux programs.

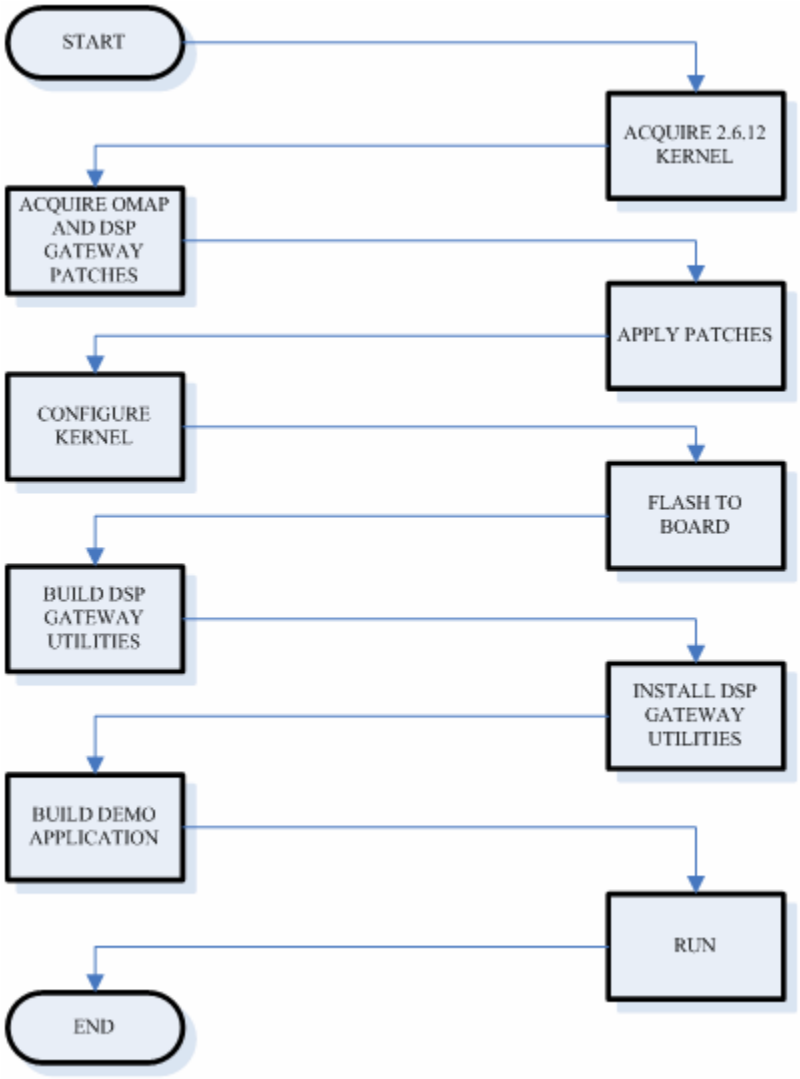

Figure 1. Flowchart of procedures for introducing dual core development.

At this point, students learn the communications methods used between the ARM and DSP cores. DSP Gateway uses mailbox communication to transfer inter-processor commands and small messages. Mailboxes are useful for transferring commands and low volumes of data. For high volume data transfer, buffers are used and pointers to these buffers are transferred between the processors. The pointers for the buffered data can be transferred through the typical mailbox. Frame buffers set aside large blocks of memory in a memory space that can be accessed by both the ARM and DSP for large volume transfers.

The first task students will execute is a demo application that comes with the DSP Gateway tools. The students must build the DSP side application using the TI LinuxDSP tools and copy

the resultant DSP executable file to the OSK board's file system. There is a complementary application provided for the ARM side which must be built using the ARM GNU cross complier (GCC) utilities. This application is also copied to the board's file system. For the Linux kernel to provide access to DSP tasks as files, files representing the tasks have to be installed as device modules in the "/dev" directory of the board's file system. This provides a consistent view of the DSP tasks as though each task is a typical Linux device. The Linux model of device presentation represents almost all peripheral devices as though they are files contained in the file system. Data is sent to devices by simply writing to the device files. The students are instructed how to accomplish this and then the students are ready to run the demo application.

The students use the *dspctl* utility to load the DSP executable and start the DSP. Once this is completed, the students only need to run the ARM side demo application to see the results. A sample of the demo applications output can be seen in Figure 2.

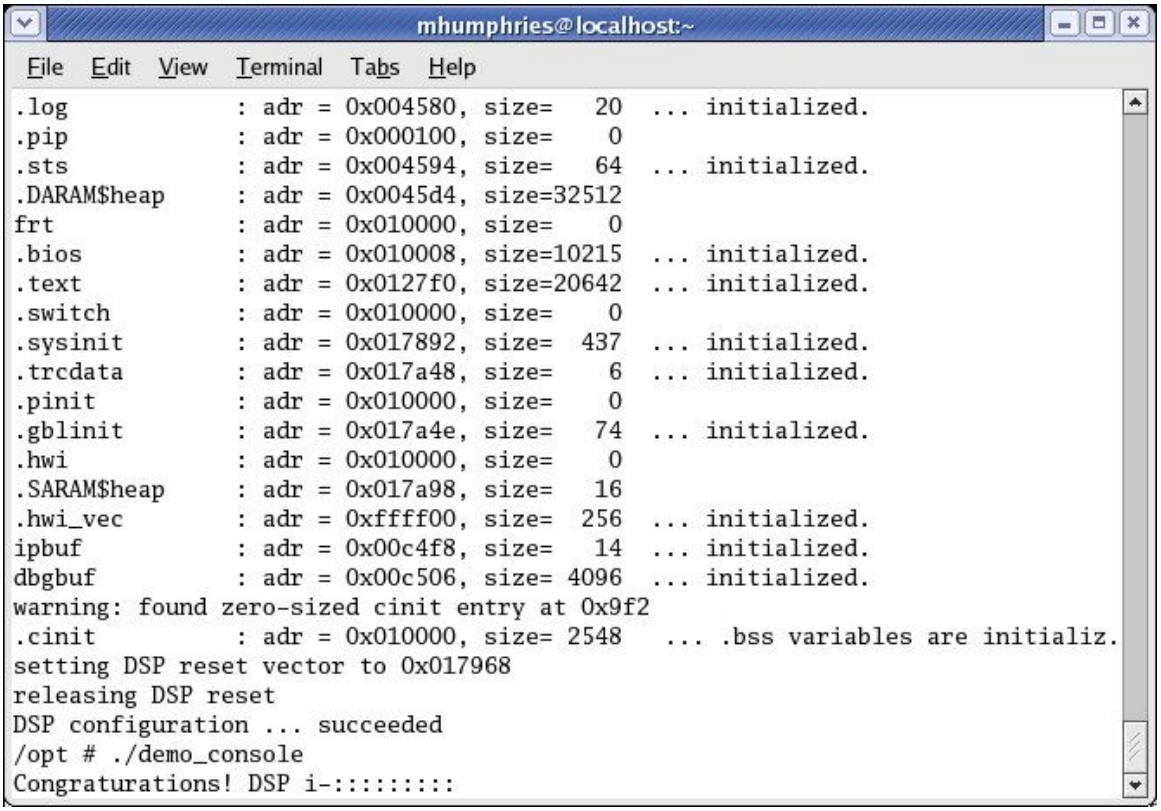

Figure 2. Output of the demo application for DSPgateway.

Dual Core Implementation of a FIR Filter

This lab provides insight to a practical application of the ARM/DSP core combination. The students see the inner workings of a finite impulse response (FIR) filter implemented on the OMAP platform utilizing the DSP Gateway driver. Presented in this lab are digital signal filtering, using predefined C callable DSP routines and buffering to transfer large amounts of

audio data. The lab also shows how to implement threaded applications in Linux. The flow of operations for this laboratory procedure is shown graphically in Figure 3.

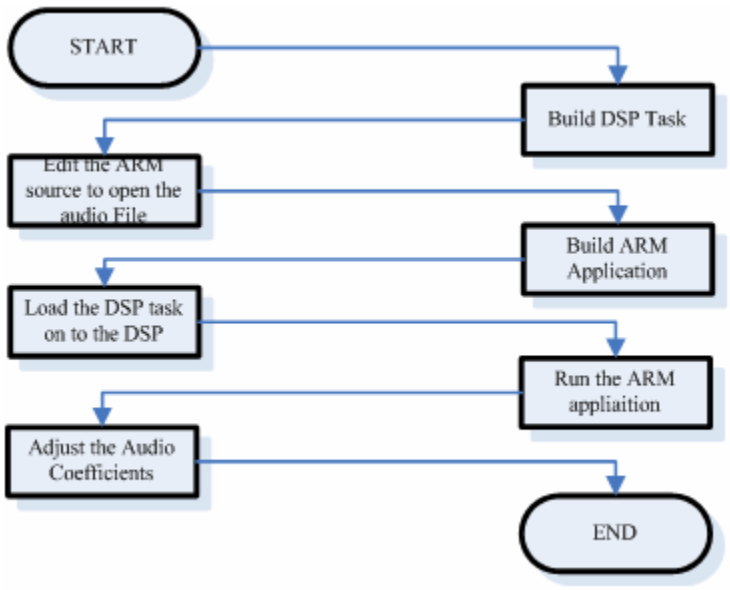

Figure 3. Flow of operations for a dual core FIR filter.

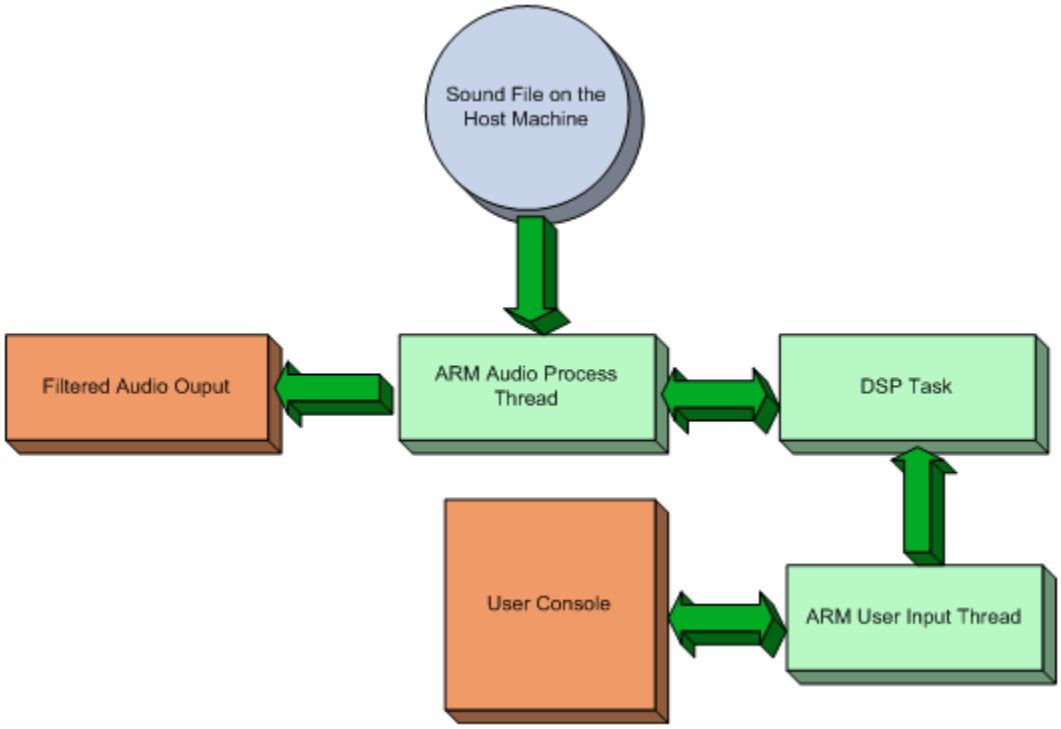

Figure 4. Schematic of operation for the FIR filter application.

The students are provided with the source code to build a DSP executable for FIR filtering of audio data and source code to build the Linux application that is responsible for sending audio

data to the DSP for processing. The DSP application makes use of a hand optimized assembly routine for FIR filtering supplied as apart of the TI DSP algorithm set. The programs communicate via the DSP Gateway driver that was installed in lab six.

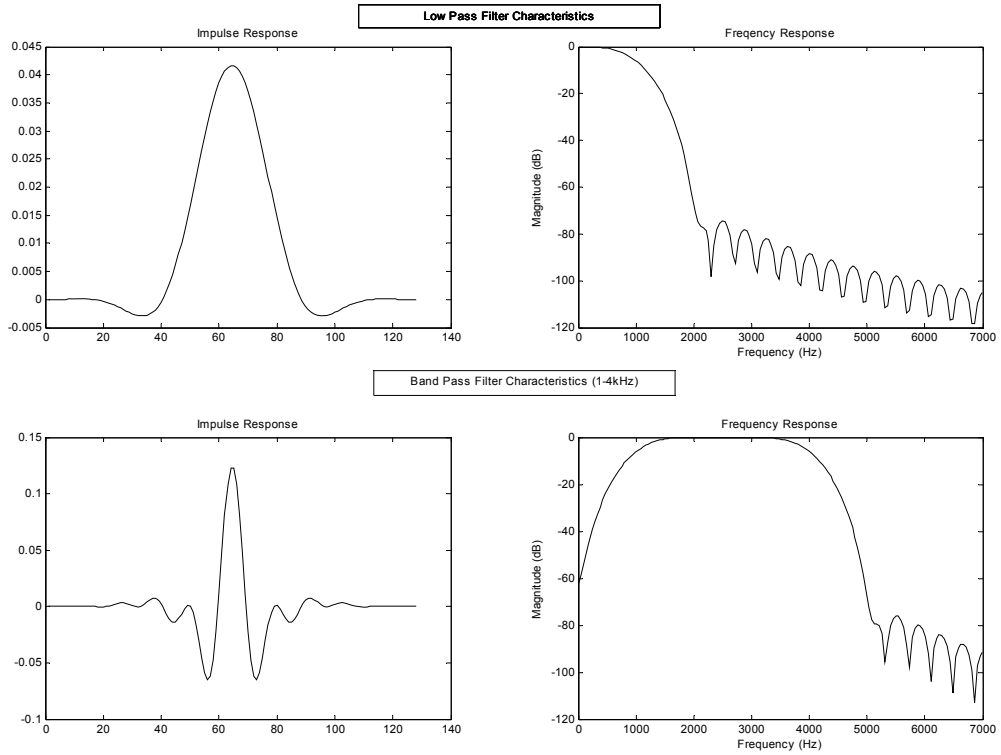

Figure 5. Characteristics of the filters used in the audio filtering lab. (Top row) 1 kHz lowpass filter response. (Bottom Row) Filter response for the 1 to 4 kHz bandpass filter.

The DSP program consists of three main functional components: the data receive function, the data send function, and the control functions. The data receive function is used to accept the buffered audio data from the Linux side application and filter it before storing the data. The data send function sends processed data to the Linux application so that the filtered audio can be sent to the audio codec for playing. The control function accepts commands from the Linux application to allow the filter coefficients to be updated for tailored behavior by the user. Figure4 shows the basic operations that are accomplished by the FIR filter application.

The filter coefficients used by the filtering function are actually the summation of five filters, which can be seen in Figure 5 and Figure 6, covering five bands of the audio spectrum. The filters used in this lab were designed by using Matlab's Filter Design and Analysis Tool [5]. To allow audio equalizer type of functionality, the user supplies information as to how each audio band should be amplified or attenuated in the Linux application. The Linux application, as can be inferred, provides the user interface to the filtering application. The Linux application is a threaded application that is responsible for sending unfiltered audio data to the DSP and receiving the filtered audio data from the DSP and subsequently sending this data to the audio

codec to produce sound in one thread. The other thread of the Linux application is responsible for user input to indicate which band is to be altered and how.

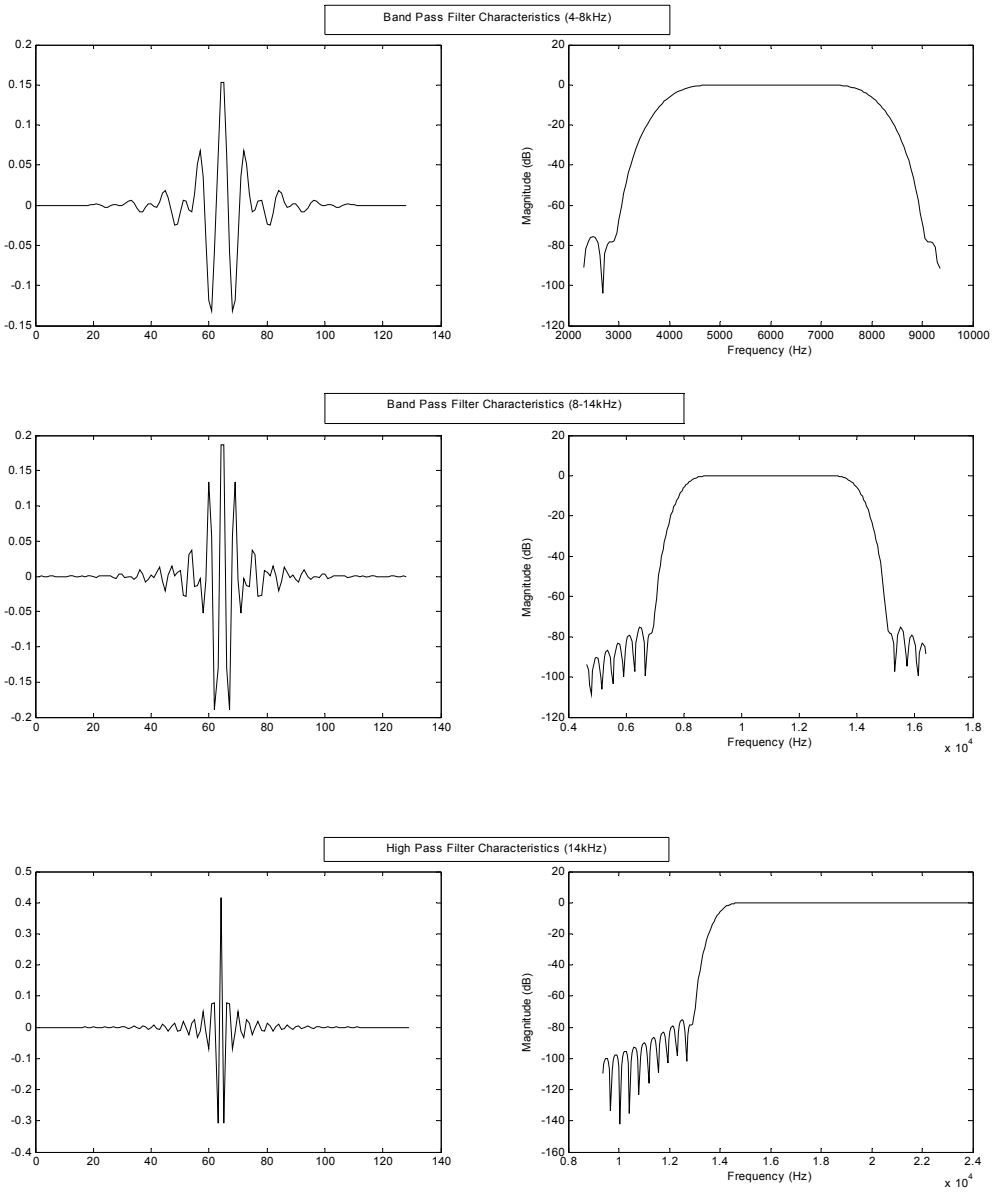

Figure 6. Characteristics of the filters used in the audio filtering lab. (Top Row) Filter responses for the 4 to 8 kHz bandpass filter. (Middle Row) Filter responses for the 8 to 14 kHz bandpass filter. (Bottom Row) Filter response for the 14 kHz highpass filter.

Conclusion

These labs provide a strong foundation for developing applications that utilize dual core processors. Students are presented with tools that are common in the development community

while applying techniques with interesting applications. The labs have been designed to be accessible to anyone with a typical junior level electrical engineering or computer science background.

Bibliography

1. M. Humphries, M. Shirvaikar, L. Estevez, "Real Time Systems Laboratory Development Using the TI OMAP Platform," in Proceedings of the 2005 American Society for Engineering Education Annual Conference and Exposition, June 2005.

2. Linux DSP Tools Downloads. Texas Instruments Incorporated Web Site. https://www-a.ti.com/downloads/sds_support/targetcontent/LinuxDspTools/index.html (December 4, 2005).

3. DSP Gateway. DSP Gateway Project Web Site. http://dspgateway.sourceforge.net/pub/index.php (December 4, 2005).

4. Kobayashi, Toshiro and Takahashi, Kiyotaka. 2005. Linux DSP Gateway Specification Rev 3.3. Nokia.

5. Signal Processing Toolbox 6.2. 2004. The MathWorks, Inc.