DSP IN HETEROGENEOUS MULTICORE EMBEDDED SYSTEMS – A LABORATORY EXPERIMENT

Pavel Lifshits, Alon Eilam, Yair Moshe, and Nimrod Peleg

Signal and Image Processing Laboratory (SIPL)
Department of Electrical Engineering, Technion – Israel Institute of Technology
Technion City, Haifa 32000, Israel
Phone: +(972)-4-8294746, fax: +(972)-4-8292796, e-mail: yair@ee.technion.ac.il
http://sipl.technion.ac.il/

ABSTRACT

Undergraduate engineering students who are learning Digital Signal Processing (DSP) are expected to have the ability to implement their theoretical knowledge in various applications soon after graduation. In this paper, we present a laboratory experiment developed for undergraduate students that addresses the challenge of getting them familiar with implementing DSP algorithms in heterogeneous multicore systems. In a top-down approach, the students first gain control of the development environment, and then implement DSP algorithms on a general purpose and on a digital signal processor core. Through the experiment, they get to appreciate the advantages of DSP core architecture in performing signal processing algorithms, and learn methods for timing and data transfer between cores while meeting real-time constraints. In a limited time frame, this hands-on laboratory experiment exposes the students to state-of-the-art multicore development practices and increases their knowledge and interest in DSP and in embedded programming.

Index Terms— *Electrical engineering education, heterogeneous multicore processing, digital signal processing, fixed-point arithmetic, BeagleBoard*

1. INTRODUCTION

While DSP theoretical fundamentals are solid and long lasting, target systems and laboratory tools are subject to rapid changes. Heterogeneous multicore processors are nowadays at the heart of numerous applications such as cellular phones, robots, digital video, imaging, automotive, computer vision and many more. The term "heterogeneous multicore processor" refers to the integration of different types of processors such as a general purpose processor, a DSP and a graphics accelerator into the same chip with a structure designed to allow these internal cores to interconnect. This architecture provides high performance, low power consumption and smaller program size, since each task is performed by a core that is optimized for it. This system architecture brings about changes in the methods used for implementing signal processing techniques in embedded systems.

The traditional goals of laboratory experiments for teaching DSP to undergraduate students are to achieve a better understanding of the theoretical concepts and let the students experience DSP in real-time. These goals are met by assigning students with the task of implementing algorithms in MATLAB, and porting them from MATLAB to C [1]. In many cases, DSP Starter Kits, such as Texas Instruments DSK6416 or DSK6713, are used [2]. Some experiments are performed with no operating system, and many times processing tasks are run at the interrupt level [3]. Nowadays DSPs are typically used as powerful coprocessors in elaborated systems. Therefore, practical experience gained through lab experiments with standalone DSP platforms is not sufficient any more [4]. Recently, few educators suggested to incorporate the use of heterogeneous multicore embedded systems into undergraduate students curriculum. However, these works, in spite of their use of heterogeneous multicore chips, usually suggest to expose students to one core only [5] or assume a full semester for getting to know the embedded platform and using it [6]. Allowing a full semester for implementation may not leave enough time to gain deep theoretical knowledge. Therefore, giving students a taste of implementation in a shorter time period may be desired.

At our university (Technion – Israel Institute of Technology), emphasis is given in the undergraduate curriculum to theoretical studies in order to provide a sound engineering background. As practical tools and practices change rapidly and may be useful only within a limited context, students are exposed to them mostly in laboratories and projects in the later stages of their studies. One of the main means for gaining practical knowledge in electrical engineering studies is a "laboratory in electrical engineering", given in the 3rd year (out of 4) of the undergraduate

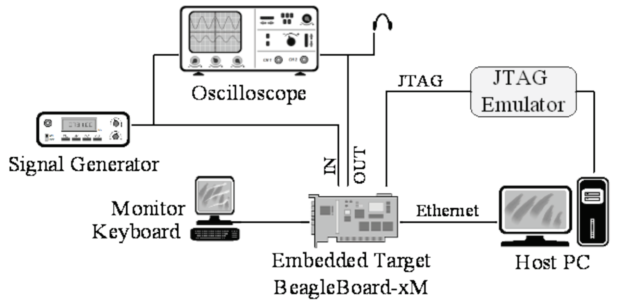

Fig. 1. Hardware layout of the experiment in the first lab session.

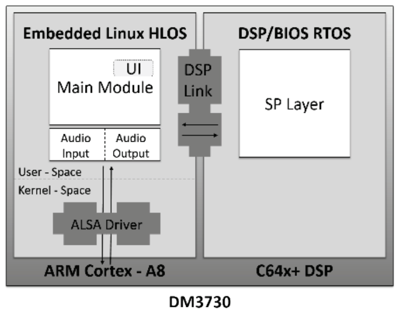

Fig. 2. Layout of a skeleton of a multimedia application given to the students during the experiment.

curriculum. In this one year course, each student must select and carry out 7 out of 34 experiments. The experiments are offered in different areas of electrical engineering and require only basic relevant theoretical knowledge (with different prerequisites for each experiment). Each experiment is divided into two sessions, 4-hours each, and is performed by students in pairs. Before each session, the students are required to read some background material and answer a set of questions in a preparatory report. After each session, the students are required to document their results and submit a final report. The students are graded based of their hands-on work and on their preparatory and final report.

In this paper, we present a laboratory experiment by which the students gain practical experience with a heterogeneous multicore environment in addition to traditional training in DSP practice. In the context of the aforementioned laboratory course, the experiment is performed in two sessions, 4-hours each. The experiment is designed for undergraduate students who took at least a course in programming (where they gain some experience with C), signals and systems (where they also gain some experience with MATLAB), a basic course in signal processing and a basic course in logic and digital systems design. Neither prior knowledge of the implemented algorithms is assumed, nor any preliminary experience in development on an embedded platform. Due to the limited time frame of the experiment, we have designed this experiment in a top-down approach. The students begin with a simple application running on the embedded platform and gradually add signal processing functionality into this application. The experiment is a major revision of a previous experiment with a similar concept, developed at the Signal and Image Processing Laboratory (SIPL) [7].

The infrastructure used in this experiment is the BeagleBoard-xM development board [8], a desktop computer, and some standard lab equipment – a signal generator and an oscilloscope. BeagleBoard-xM is a low-power open-source embedded development platform that is a mini computer. It was designed with open source software development in mind, and as a means of demonstrating Texas Instruments DM3730 system-on-a-chip [9]. It serves as an educational tool to teach open source hardware and open source software capabilities. The DM3730 is compatible with Texas Instruments OMAP 3 architecture. It contains an ARM Cortex-A8 as the general purpose processor, a C64x+ DSP and a SGX510 graphics accelerator. The operating system used in this experiment is the Angstrom distribution of Linux which is tailored for embedded devices.

The remainder of this paper is organized as follows. In Section 2, we give an overview of the experiment setup. In Section 3, we describe the tasks students have to perform and the insights they gain during the experiment. Finally, in Section 4 we summarize lessons learned during the experiment and draw some conclusions.

2. SYSTEM OVERVIEW

Fig. 1 depicts the hardware setup of this experiment. We chose the BeagleBoard-xM as the laboratory target platform because of its low cost, availability, strong DM3730 processor, sufficient peripheral access, and the existence of a large and active development community. The rising popularity of embedded Linux and its free licensing makes it a good choice for a target operating system. The BeagleBoard-xM is connected to a host computer using a dedicated Ethernet connection and a JTAG emulator. The Ethernet connection is the main connection channel between the host and target, while the JTAG connection is used for debugging the DSP core. The BeagleBoard-xM is equipped with a keyboard and is connected to a monitor. This is done to help the students understand the ability of the embedded platform to constitute a standalone system.

As shown in Fig. 1, at the first lab session a signal generator is connected to the audio-in jack of the BeagleBoard-xM and to channel 1 of an oscilloscope, while the audio-out jack of the BeagleBoard-xM is split between channel 2 of the oscilloscope and the headphones. This allows to demonstrate visually the idea of real-time constraints. It also allows to demonstrate some basic digital signal processing concepts such as sampling and latency. During the experiment, the students are required to push the processing burden to the limit and beyond. They observe how too slow performance of a task affects the output signal displayed on the oscilloscope, and can also hear the disrupted audio. At the second lab session, the audio-in jack is connected to the development host PC line-out and both the right and left channels of the audio-out stereo jack are

Task	Educational Target
Build a skeleton application on the ARM core and perform an audio loopback	Be familiar with the lab setup and development environment
Run the ARM core in debug mode	Learn how to debug in an embedded environment
Add an FIR filter to the ARM code	Perform a basic signal processing operation
Overload the ARM processing power	Be familiar with the effect of not meeting real-time processing requirements
Port the FIR function to the DSP core and check performance enhancement	Demonstrate the advantage of using DSP architecture for signal processing tasks
Run the DSP core in debug mode	Learn how to debug a co-processor application in a multicore environment
Analyze DSP assembly code before and after optimization	Learn how the optimizer is utilizing the DSP architecture to improve performance
Compare implementation of the FIR filter in floating-point vs. fixed-point	Demonstrate the limitations and advantages of fixed-point implementation of DSP algorithms

Table 1. Tasks performed in the first lab session.

connected to channel 1 and channel 2 of the oscilloscope, respectively. We use the host PC to generate more complicated signals. We use the stereo property of the line to provide an input signal to one channel and a reference signal to the other channel of the BeagleBoard to demonstrate an echo cancelation algorithm. The oscilloscope is used to display graphically the convergence of an adaptive LMS filter coefficients.

Due to the heterogeneous nature of system-on-chip platforms, a variety of development and debugging tools are being used. In order to allow the students to concentrate on the concepts and algorithms, we have built a unified development framework. This framework produces all types of executable code, whether the target platform is the embedded ARM core, or the DSP core. Debugging tools are also preconfigured for both the ARM and the DSP cores. As a result, the students work only with Code Composer Studio development environment [10] throughout this experiment. In addition to the unified development framework, we provide a skeleton of a multimedia application. The skeleton includes means for input and output, memory management, thread scheduling, inter-processor communication, etc. Fig. 2 depicts the design of the application at an advanced stage of the lab experiment. The figure shows how the application uses the different cores and the APIs used by the different modules. The ARM core runs the embedded Linux high-level operating system, while the DSP core runs the DSP/BIOS real-time operating system. The ARM core executes the master part of the application which includes audio input, audio output and a simple user interface. Audio is handled through the ALSA (Advanced Linux Sound Architecture) driver. The signal processing algorithm is executed on the DSP core and a DSPLink API is used to interconnect the ARM and DSP. The application also provides basic means of profiling by measuring the execution time of the signal processing module. This allows the students to compare different implementations, for example, fixed-point vs. floating-point, and verify that real-time constraints are met as long as the processing rate is greater than the data rate. The students have access to all the source code. During the lab experiment, they compile and build the entire application. Yet they are writing their own code only in dedicated areas in the skeleton program. Most system parameters, such as buffer sizes and sampling rate, are defined at the same location by pre-compiler macro constants. Using this approach, a complex system is created and modified by the students within a limited time frame. Consequently, students are provided with sufficient time to experiment with system parameters, algorithm development and to exploit the DSP core architecture.

Task	Educational Target
Implement an adaptive filter in floating point	Add an advanced signal processing functionality to the system
Perform noise reduction of a noisy tone	Demonstrate adaptive filter convergence in a simple setup
Adaptive echo cancelation of speech signals	Learn the effect of filter length and other system parameters on echo cancelation quality
Port the adaptive filter to fixed-point	Be familiar with the challenge of limited dynamic range due to fixed-point implementation

Table 2. Tasks performed in the second lab session.

3. EXPRIMENT PROCEDURE

When performing the experiment, the students are facing the challenge of using a complex development environment that is mostly new. They are not that experienced in C programming, the development tools are new, the application runs under the Embedded Linux operating system, and they are not necessarily familiar with the specific lab equipment. In this challenging environment, they are required to implement, debug, and test a DSP algorithm in a relatively short time. By the traditional approach, it is common to begin by developing the algorithm in MATLAB, port it to the C language, and then get familiar with the development environment. Using this bottom-up approach, the full picture, along with the biggest difficulties, are revealed to the student only in advanced stages of the experiment. We have found that this bottom-up approach is causing difficulties to complete the experiment in the given limited time frame. To overcome this difficulty, we have designed this experiment with a top-down approach as summarized in Table 1. The students begin with a simple application running on the

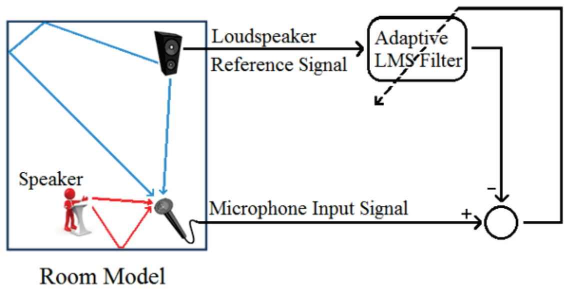

Fig. 3. Acoustic echo cancellation experiment setup.

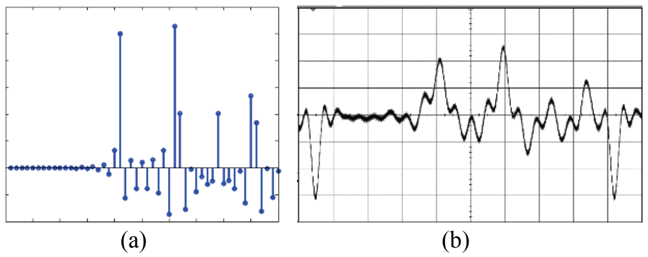

Fig. 4. Adaptive filter coefficients displayed to demonstrate LMS algorithm convergence in (a) MATLAB and (b) an oscilloscope.

embedded platform and gradually add signal processing functionality to this application.

The first task of the students is to build a simple application running on the ARM core. This provides the students with an overview of the system right from the beginning. The aforementioned skeleton application is provided as a baseline to which functionality is added. In this application, it is required to loop a 1 KHz input signal from the analog input to the analog output, and monitor both signals on the oscilloscope. Gradually increasing the input signal's frequency to over half the sampling rate, demonstrates the limited input bandwidth. Due to the use of input and output buffers, a delay exists from input to output, referred to as latency. This is a key parameter in real-time signal processing systems. The students are asked to determine the latency of the system using the fact that delay in time domain causes a phase shift. They inject a tone and measure the phase difference between input and output signals. The effect of buffers size and sampling rate on latency are demonstrated as well. Then, a "do nothing" loop is added to the application. When the loop size parameter is large enough, the output signal displayed on the oscilloscope becomes discontinuous since the output buffer is not ready in due time. This way, the students learn to identify the effect of lack of processing resources on system output. It is demonstrated that the time difference between output signal irregularities is proportional to the input/output buffers sizes.

Next, debugging the ARM code is practiced. The students are asked to insert breakpoints, run the program step-by-step, display variables and display memory contents both in numeric and graphical form. At this stage of the experiment, the students have gained the basic skills and a sufficient system understanding to develop signal processing applications. As most students are not experienced with fixed-point arithmetic, a homework assignment of designing an FIR high-pass filter using MATLAB and transforming it to fixed-point is given before the lab. At the lab, the FIR function is first implemented in the C language on the ARM core. The filter order is gradually increased up to the point where the output signal is distorted as a result of not meeting real-time processing requirements due to computational overload. Comparison is done between floating-point and fixed-point implementations.

After practicing signal processing on the ARM core, the students port this task to the DSP core. It is demonstrated that the dedicated architecture of the DSP core, provides for much higher FIR order in real-time processing compared to the ARM implementation. Further performance enhancement is achieved by running the DSP code optimizer. The students are asked to study and analyze the optimized FIR function assembly code, and point out the way DSP architecture is utilized in the optimized code. This exposes the students to the DSP architecture as well as to optimization concepts such as software pipelining.

In the second lab session, the students benefit from the practical experience gained in the first session, and develop an adaptive echo canceling application. The tasks performed by the students in the second lab sessions are summarized in Table 2. Theoretical background is provided in the experiment booklet, based on [11], and the algorithm is simulated using MATLAB in a preparatory homework assignment. In the lab, an LMS adaptive filter is implemented in floating-point and then in fixed-point and used to perform an echo canceling function on the DSP core. Fig. 3 illustrates the experiment setup. The reference signal is the loudspeaker audio. The microphone input signal is the sum of in-room speaker and loudspeaker audio that have passed through a room model. The adaptive LMS filter estimates the room transfer function. The estimated loudspeaker audio that have passed through the room model is subtracted from the microphone signal. In the resulting audio, the loudspeaker echo is attenuated. The adaptive filter coefficients are displayed on the oscilloscope to demonstrate algorithm convergence in real-time. As illustrated in Fig. 4, the students observe and compare the filter coefficients of the simulated room model with the estimated transfer function which gradually converges and is displayed on the oscilloscope.

4. LESSONS LEARNED AND CONCLUSION

In this paper, we describe a novel laboratory experiment for undergraduate students. Students performing the experiment use the ARM and DSP cores on a BeagleBoard-xM development board. The experiment is the result of the authors' experience in teaching embedded DSP to undergraduate electrical engineering students for many years. It is based loosely on previous experiments such as [7] and was designed to teach students the cutting edge of modern heterogeneous embedded systems. In this section, we share with the readers some of our insights and conclusions while designing and running this experiment.

An educator should first set goals that describe the desired outcome of an experiment and only then design the experiment based on these goals. An experiment in embedded DSP systems can have many desired outcomes, namely, insights that should be gained by the students performing the experiment. Examples of such goals are better understanding of theoretical concepts, acquaintance with engineering tools and gaining experience that is relevant to future work in the industry. These goals are important and should be taken into account. Nevertheless, we believe that the most important goal of an experiment in embedded DSP systems is to have the students understand the limitations of technology in real-world applications. These limitations are common hurdles in DSP implementation. For example, meeting run-time constraints or memory constraints, or dealing with limited numeric precision (usually fixed-point).

One of the merits of a good engineer is his or her ability to apply a systematic approach towards practical problems. To train students to acquire this ability, and based on our experience, porting an algorithm from simulations to an embedded platform that operates in real-time, we have put the systematic approach at the center of the experiment. In all the stages of this experiment, we emphasize how the parts of the system make a whole. We let students use lab tools such as a signal generator and an oscilloscope, and software tools, such as ARM and DSP development environments. The students are exposed to different trade-offs in each component and among different components. These trade-offs are demonstrated, for example, by porting part of the implementation from the ARM core to the DSP core, by transforming the code from floating-point to fixed-point numerical precision, and by considering latency vs. processor load. As we put the system in the center, the experiment is structured in a top-down approach. The whole system is presented first, and the details are presented later. This is not a common approach in similar experiments and we have found out that it helps students to develop a good engineering practice while dealing with trade-offs.

Due to the complexity of embedded systems, many undergraduate experiments dealing with such systems are based on basic examples such as a simple filter or an FFT. We have found that dealing with more complex real-life challenges in such an experiment contributed a lot to students' motivation. The challenge in selecting a nontrivial problem for an undergraduate experiment, and especially with strict time constraints, is that the students may not be able to grasp all the theory or deal with all the technical details of its implementation. The most difficult task for the educator is to design such an abstraction for the experiment that, on one hand, gives the student enough details to feel comfortable with the problem at hand and with the development environment, and, on the other hand, hides some of the details of complex theory and complex development environment. This delicate task requires a lot of experience from the educator and some trial-and-error.

Until writing this paper, several dozens of students have performed the lab experiment. In order to assess the impact of the experiment and to validate the attainment of its educational goals, the final reports submitted by the students were analyzed and a short interview with each student who performed the experiment was conducted. From most of the reports, it is evident that, after participating in the lab, the students gained basic understanding of real-time system concepts, fixed-point vs. floating-point representation and embedded multi-core architecture. During the interviews, all students stated that the experiment raised their interest in real-time embedded systems. Some have also noted that this experiment encouraged them to proceed their studies in the field of signal processing and/or embedded systems.

ACKNOWLEDGMENT

We would like to thank Prof. David Malah, head of SIPL, for his support and advice. Creating the experiment described in this paper was supported, in part, by Texas Instruments European University Program.

REFERENCES

[1] T. B. Welch, C. H. Wright, and M. G. Morrow, *Real-Time Digital Signal Processing from MATLAB® to C with the TMS320C6x DSPs, 2nd ed*: CRC Press, 2012.

[2] R. Chassaing and D. Reay, *Digital Signal Processing and applications with the TMS320C6713 and TMS320C6416 DSK, 2nd ed.*: Wiley, 2008.

[3] S. M. Kuo, B. H. Lee, and W. Tian, *Real-time Digital Signal Pocessing: Implementations and Applications*: Wiley, 2006.

[4] C. Wicks, "Lessons Learned: Teaching Real-time Signal Processing [DSP Education]," *IEEE Signal Processing Magazine,* vol. 26, pp. 181-185, 2009.

[5] M. G. Morrow, C. H. Wright, and T. B. Welch, "Real-time DSP for Adaptive Filters: A Teacing Opportunity," presented at the 38th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Vancouver, Canada, 2013.

[6] K. Mankodiya, M. Klostermann, S. Vogt, O. Christ, and U. G. Hofmann, "Teaching Signal Processing with a Dual-core Embedded Platform," presented at the 4th European Education and Research Conference (EDERC), Nice, France, 2010.

[7] O. Bryt, A. Elron, P. Lifshits, T. Zidenberg, Y. Moshe, and N. Peleg, "A Laboratory Experiment for Real-time Echo Cancellation using a Beagleboard," presented at the 5th European DSP Education and Research Conference (EDERC), Amsterdam, Netherlands, 2012.

[8] BeagleBoard-xM: http://beagleboard.org/Products/BeagleBoard-xM

[9] *DM3730, DM3725 Digital Media Processors (SPRS685D).* http://www.ti.com/product/dm3730: Texas Instruments, 2011.

[10] Code Composer Studio IDE v5.5: http://www.ti.com/tool/ccstudio

[11] S. O. Haykin, *Adaptive Filter Theory* 4th ed.: Prentice Hall, 2001.