Experiences with the Blackfin Architecture in an Embedded Systems Lab

Michael Benjamin, David Kaeli, Richard Platcow
Department of Electrical and Computer Engineering
Northeastern University
Boston, MA 02115

Abstract

At Northeastern University we are building a number of courses upon a common embedded systems platform. The goal is to reduce the learning curve associated with new architectures and programming environments. The platform selected is based on the Analog Devices Blackfin digital signal processor.

In this paper we discuss our recent experience developing anew undergraduate embedded systems lab. Students learn to utilize the embedded DSP platform to address a number of different applications, including controller design, RS-232 communication, encryption, and image processing. This platform provides a rich design exploration sandbox replete with programming and simulation tools. We describe our use of this platform in our Microprocessor-based Design Laboratory and discuss how this platform can be used in a range of classes.

I. Introduction

At Northeastern University, our electrical and computer engineering students pursue 5-year B.S. degrees and gain workplace experience with up to 1.5 years of cooperative education. To better prepare our students for the hands-on work of the real world, we are building a common educational platform allowing development in real-world systems, while reducing the learning curve associated with working on a new platform.

Such a platform must support a range of classical topics including signal processing, controls, computer architecture, and embedded systems. But the platform's tools must also be easy to use, and the associated documentation and pedagogic material must be sufficiently rich. Additionally, the platform must be practical enough that implementations using it at least hold the attentions of students and at best ignite their imaginations.

We have selected the Blackfin architecture from Analog Devices (ADI) to provide such a platform. The Blackfin combines the functionality of a digital signal processor with that of a micro-controller unit. The architecture supports a number of tool-chains, operating systems, and development environments.

In 2003, our Digital Signal Processing class began using the ADI Blackfin. Based on very positive student feedback with this platform, we have begun to deploy this platform in a number of classes. In the Summer/Fall of 2005, our Microprocessor-based Design lab was redesigned to use the Blackfin. The following Spring 2006 semester, the revised lab was offered for the first time. The lab presented the concepts of performing basic I/O, micro-control program design, RS-232 communication, encryption, and image processing.

It is our goal for this lab to provide a platform upon which our students quickly build. Our experience to date is that students taking the DSP course that utilizes this platform in the lab, utilize this same set of tools to implement their senior capstone projects.

The objective of this paper is to describe our very positive experiences with this platform, and to discuss our future plans to build off this experience by replicating it in other labs. The rest of the paper is organized as follows. Section II discusses related work. Section III gives a brief introduction to the Blackfin processor architecture. Section IV describes the hardware platform and the supporting materials for the lab. Section VI concludes the paper and Section V describes how this model can be extended to future labs.

II. Related Work

ADI supports academic programs through their University Program Initiative [1]. This program makes available a series of online training modules, including videos, PDF transcripts, presentation slides, and some code examples; workshops and seminars in North America, Europe, and China; and training publications [2]–[4].

A number of universities around the world have begun to utilize the ADI Blackfin EZ-KIT platform to support their classes. EZ-KIT evaluation boards integrate a Blackfin processor with a range of peripherals. The *ECE-ADI Project* at University of Calgary has a rather extensive collection of audio, video, microcontroller labs, and presentation materials [5]. The University of Massachusetts Lowell new *Handy Board* [9] utilizes the Blackfin. A one-day course at the University of Parma, Italy provides hands-on experience using the Blackfin [6]. California Polytechnic State University has utilized the Blackfin in a course and has also described how it could be used to revamp an entire curriculum [7]. This most recent adoption has motivated the curricular changes at Northeastern to adopt the Blackfin EZ-KIT.

III. Blackfin Processor

The Blackfin was designed to provide micro-controller (MCU) and digital signal processing (DSP) functionality in a single processor, while allowing flexibility between the needs of control and DSP. With this duality in mind, the Blackfin

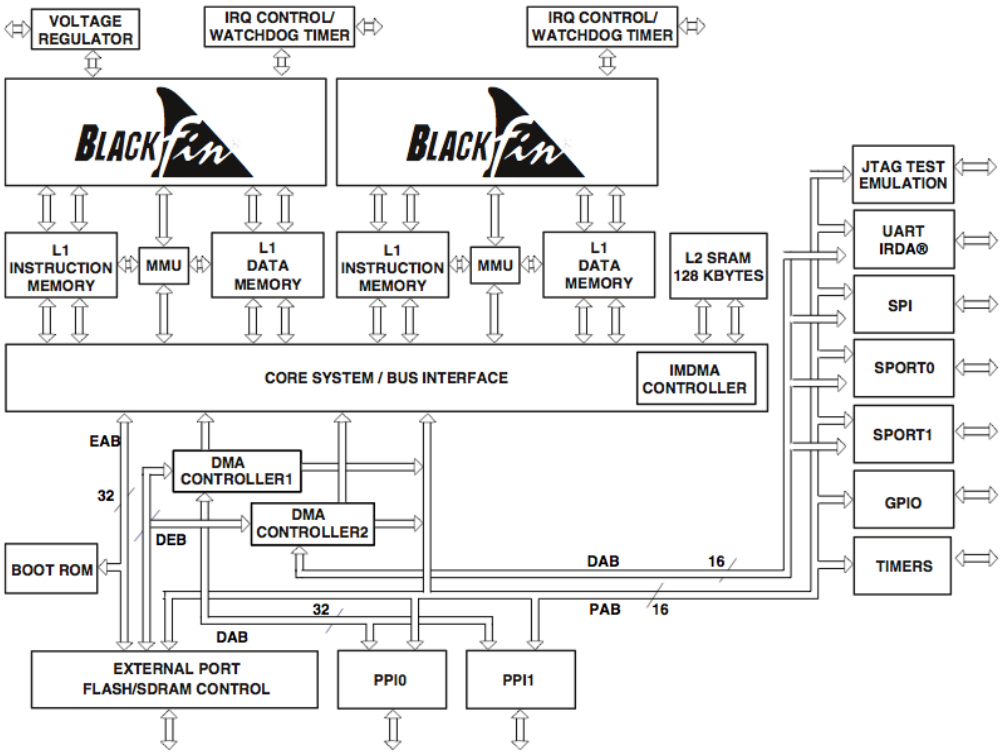

Fig. 1. Blackfin BF561 processor overview [17].

incorporates a Single Instruction, Multiple Data (SIMD) processor with features such as a variable-length RISC instructions, software-programmable on-chip PLL, watchdog timer, real-time clock, memory management unit, 100 Mbps serial ports (SPORTs), UART controllers (with IRDA support), and SPI ports. The MMU supports multiple direct memory access (DMA) channels for data transfers between peripherals and SDRAM, FLASH, SRAM memory subsystems, and also supports configurable on-chip instruction and data caches. The Blackfin hardware supports 8-bit, 16-bit, and 32-bit arithmetic operations - but is optimized for 16-bit operations.

A. Architecture

The Blackfin architecture is based on the Micro Signal Architecture [15] developed jointly by Analog Devices (ADI) and Intel, which includes 32-bit RISC instruction set and 8-bit video instruction set with dual 16-bit multiply-accumulate (MAC) units. ADI has been able to achieve a balance between the needs of DSP and MCU with the Blackfin instruction set architecture. Using C/C++, a developer can rely on the compiler to generate highly dense yet computationally efficient code; or the developer can write targeted assembly. For real-time needs, operating system support becomes critical - the
Blackfin supports memory protection and supports a number of operating systems.

Development on the Blackfin is supported by a number of tool chains, including ADI's VisualDSP++ Integrated Development and Debugging Environment (VDSP++ IDDE) and GCC. Supported operating systems include INTEGRITY [8] and velOSity [10] from Green Hills Software, ThreadX [11] from Express Logic, Nucleus from Mentor Graphics [12], Fusion [13] from Unicoi Systems, and the μClinux embedded Linux/micro-controller project [14].

Blackfin comes in both single-core (e.g., BF533, BF535 and BF537) and dual-core (e.g., BF561) models. The organization of the BF561 used in our Microprocessor-based Lab is shown in Figure 1. This chip provides for symmetric multiprocessing, while also providing low power consumption.

IV. Lab Environment

The Microprocessor Lab course was co-requisite to the Microprocessor-based Design course, which met twice weekly for 100-minute lectures. The lab section consisted of 15 undergraduates, most in their junior year. The students worked in groups of 3 or 4 in the laboratory.

The course was split into 5 individual experiments over a 14-week semester - each lab consisting of two 2-hour sessions. A sixth student-designed lab was also assigned, in which students proposed a project that could easily lead to an engineering capstone design project (all seniors are required to complete a capstone project in their senior year). The labs were conducted every other week to provide students ample time to write up their reports. In addition to the textbook [16] used in the course, a number of hardware and software manuals were used from ADI [17]–[20].

A. Goals of the Lab

In addition to our fundamental goal to strengthen students' comprehension and retention of lecture material, additional goals for the lab course were:

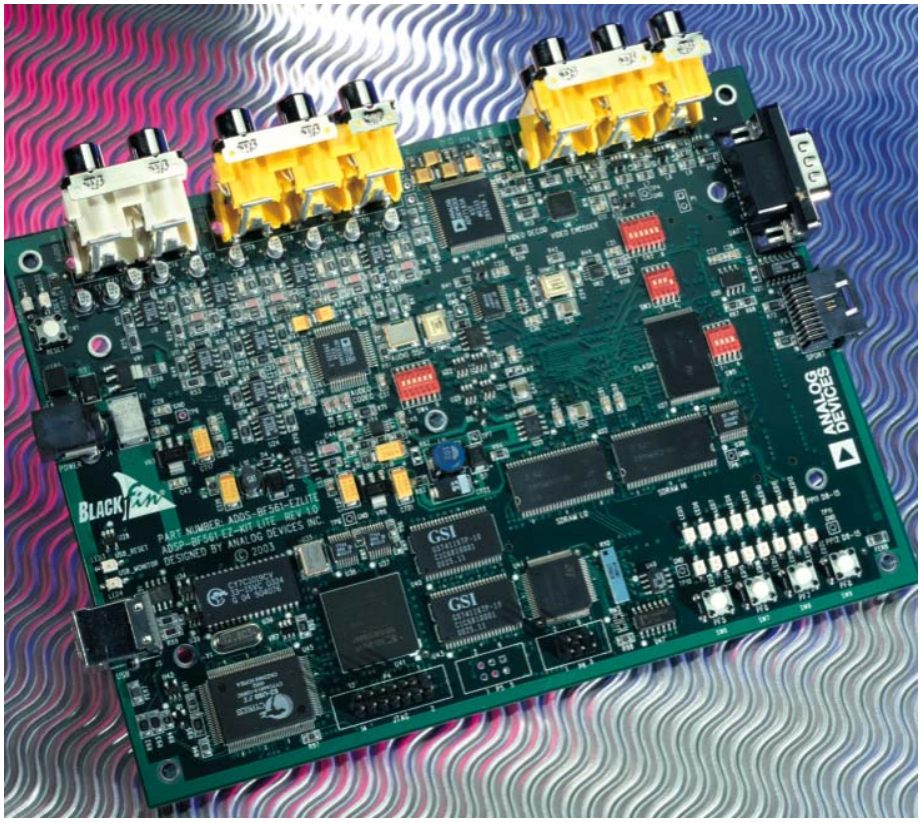

Fig. 2. Analog Device's Blackfin BF561 EZ-KIT Lite evaluation board.

- To balance breadth and depth of embedded systems education.
- To encourage students to develop skills for problem solving, group/time management, and self-learning.
- To prepare students for technical writing in industry and academia.

To balance breadth and depth, we chose to use the C language and the VDSP++ IDDE. Assembly is discussed in the course lectures, and we show in lab how students can use inline assembly in C subroutines. To encourage independence in problem solving and learning, students were assigned a student-design lab and used Wikipedia articles as reference material. To strengthen technical writing, each report had an "executive summary" generally describing the lab and a "technical abstract" to address such detail.

B. Equipment

In this lab we utilize the Blackfin BF561 EZ-KIT Lite evaluation board (shown in Figure 2). The BF561 board integrates an ADSP-BF561 symmetric dual-core processor designed for consumer multimedia into an evaluation board which includes 64MB of SDRAM, 8MB FLASH, stereo audio channels (2 input/3 output), video jacks (3 input/3 output), UART/RS232 line driver/receiver, 20 LEDs, 5 push buttons (1 reset, 4 programmable flags), JTAG ICE 14-pin header and and expansion interface.

Our lab setup includes an Ontrak ADR101 serial data acquisition interface board. This allows us to interface the BF561 with a relay using a RS-232 protocol and a basic string-as-command interface (see Figure 3).

Fig. 3. Ontrak ADR101 serial data acquisition interface (using RS232).

C. Programming Environment

We utilize the ADI VDSP++ programming framework in the lab, shown in Figure 4. This framework is quite extensive and we are only able to touch on the basics in this lab (in the future we are looking to utilize this platform in a freshman programming class). Section V provides a discussion of how

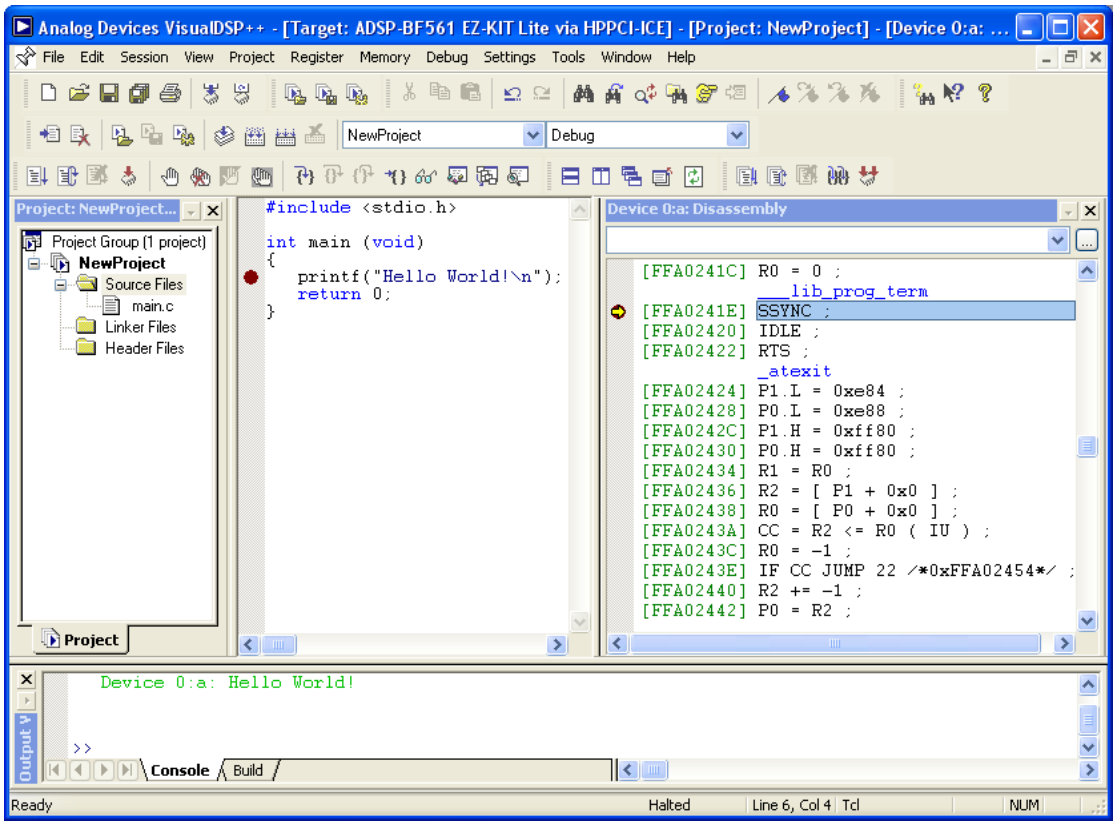

Fig. 4. Analog Device's Visual DSP++ 4.0 IDDE.

advanced labs might further exploit the capabilities provided by VDSP++.

D. Experiments

Next we describe the series of experiments assigned to the students. They are presented in the order in which they are assigned.

1) Basic Input/Output: This lab introduces how to use the programming tools provided in order to produce input and output. In the first part of lab, the students develop a simple C program for standard I/O. In the second part, they evaluate a definite integral using the Monte Carlo method. In the third part, students use push-buttons and LEDs on the BF561 EZKIT Lite board to implement simple board-level I/O. The students use Monte Carlo simulation code [21] to evaluate $\int_0^1 e^{-\frac{1}{2}x^2} dx$ - corresponding to the probability density function for a Gaussian distribution (without the scaling factor $\frac{1}{\sigma\sqrt{2\pi}}$, where σ^2 denotes the variance). This exercise is intended as a demonstration of the available C functionality on the Blackfin and also as a conceptual introduction to the theoretical topics of stochastic and numerical methods.

To demonstrate simple I/O, one group's solution was to develop a simple game similar to *Simon*. The game controls two LEDs in the bottom row, and are located close to a pushbutton.The game awards a point to the user if the user hits the correct button in time, and keeps score in binary on the top row of the LEDs.

A solution was given to the students that featured a *tennis game*. LEDs light one row at a time in succession, moving from one side to another. The direction changes as the appropriate *racket* button is pressed.

2) Washing Machine Controller: This lab covers how to implement an abstract finite state machine (FSM) model of a washing machine in an embedded controller design. In the first step, the students produce the FSM and implement it in C, using `stdin` and `stdout` to prototype sensors and control devices. An example FSM is given in Figure 5.

In the second step, students revise their I/O function to use a UART controller library which sends strings to a Windows XP Hyper Terminal session via an RS-232 interface. Finally, the students build a washing machine prototype using a breadboard, LEDs for outputs to the controller (such as valves to supply and drain water, and a motor to agitate the load) and switches for inputs to the controller (e.g., water-level sensors, door open/close sensors, etc.).

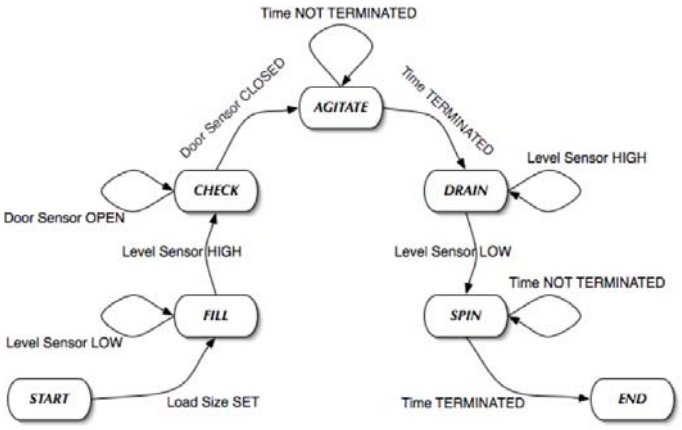

Fig. 5. Example of FSM for washing machine control.

3) RS-232 Communication: In this lab students use the UART controller on the BF561 to interface with the ADR101 and the breadboard washing machine prototype they built in the previous lab. To verify their designs, the program writes the correct ADR commands to a Hyper Terminal session. In their reports, the students were asked to explain the RS232/UART communication step-by-step.

4) Encryption: In this lab, the students implement an RSA cryptography scheme that encrypts a text file and sends the encrypted characters from one BF561 to another. The data is then decrypted and displayed on the VSDP++ console. To encourage self-learning, the students were instructed to a read`Wikipedia articles that explained the overall RSA algorithm and numerical methods for exponentiation.

The basic RSA cryptography scheme discussed is summarized as follows. To transform a character with the ASCII value of x into an encrypted number c we can use r (the public exponent), and n (the public modulus, which is the product of two random, private prime numbers $n = p \times q$) using the following equation:

$$\texttt{encrypt}(x) = x^r \bmod n = c \qquad (1)$$

Similarly to transform a decrypted number c into a character with the ASCII value of x, we use s (the private exponent) and n (again, the public modulus).

$$\texttt{decrypt}(c) = c^s \bmod n = x \qquad (2)$$

For very large values of r and s, the exponentiation requires the use of a numerical method (such as the *Square-and Multiply* algorithm, binary or modular exponentiation). The student reports discuss which numerical method was best, in terms of Big-O notation, execution time and number of operations, and also how to solve communication problems (e.g., detecting when the value of an encrypted character was greater than the range of character values, how to send integers as strings, etc.). Students also address the RSA Factoring Challenge [22] - in particular what the meaning of recent factorizations of challenge numbers such as (RSA-640 in November 2005) implies about the security of encryption keys.

5) Image Processing: Given the complexity of introducing image processing in this lab, limitations were placed on the scope of this experiment. We assumed the lab would use a single frame of video. The frame would be represented by a file in RGB24 format (24-bits for red, green, and blue – each one byte) with a pixel represented by a line, each RGB value comma-separated on the line.

The first part of the lab is to read the image, convert it to RGB565 using a given C macro, and display it in VDSP++'s Image Viewer. The file took some time to load because of the USB interface - but using a high-performance PCI (HPPCI) emulator the loading of the file was very fast. The second part of the lab is to develop code for filtering out pixels below a fixed threshold. The third part of the lab is to perform a *fire detection* given a video frame. The students applied the same simple filter, but this time the threshold corresponded to the colorspace occupied by fire. Figure 6 shows the before and after results for filtering based on the colorspace corresponding to fire.

6) Student-Design: This final lab was intended to be act in part as a segue to capstone design. Students were allowed to design their own final lab but were not required to fully implement the design. Each group wrote a proposal that included:

- the goals and motivations of the work
- related work
- problem(s) addressed
- solution(s) proposed
- justification of the approach (listing advantages and disadvantages)
- a time schedule for implementation (with Gantt charts)
- division of work between group members

(a) Before filtering

(b) After filtering (with a darken transform)

Fig. 6. Video frame processing.

- any deliverables (such as data collection or prototyped product)
- market considerations for product(s), or contribution to field(s) for research

E. Student Feedback

The Spring 2006 lab consisted of only 15 students, but students were asked to fill out an anonymous web survey to rate the lab. On average, most students found the lab to both difficult but very useful. Students answered that they had learned as much as expected, improved their time-management skills, found the group-based evaluations somewhat useful, found the Wikipedia articles very useful, and overall felt that the balance of breadth and depth was good.

V. Extensions

The new Microprocessor-based Design Lab is only in its first year and will continue to be refined. We plan to improve both the hardware and software aspects of the platform. We are also providing materials online for other labs to use and to encourage interested persons to help develop the work.

Some features of the VDSP++ IDDE were mentioned but not dealt with thoroughly. Advanced labs could incorporate profiling using the VDSP++ Profile Guided Optimization (PGO) Tool. Once students discovered *hot code* in their programs, assembly code could be hand tuned (a typical exercise of commercial implementations). Also, because the Blackfin can use a variety of development platforms, the lab would be well served to incorporate GCC/μClinux as an example of an alternative tool chain.

In addition to improving software-related topics, there remains room for improving hardware-related topics. Taking advantage of both BF561 cores for performance and power should be explored (for instance, the first lab might implement a *doubles game*, with the rows and buttons corresponding to a one-ball, four player game). In the future, we hope to make use of the extender boards available for the BF561 such as the USB-LAN board for disk I/O or network communication; and the FPGA board for acceleration. Indeed, we would like the students to utilize the FPGA more heavily in different steps in the labs. Also because the instruction and data caches are configurable, an advanced discussion of caching should be included.

Finally, as an outgrowth of the lab, we have developed a website for the Northeastern University Blackfin Labs *NEUfin Project*. The goal of the project is provide for an open discussion and development of relevant labs and tutorials for others. The material produced by the project is intended to be shared with other schools. The website is located at:

`http://www.ece.neu.edu/groups/nucar/BFLab`

VI. Summary

Northeastern University is adopting a common embedded platform to be used in a number of undergraduate courses.

The Blackfin architecture has been selected upon which to base this platform because of its DSP and MCU functionality, its support of operating systems and programming languages, its documentation, its use in industry, and also for ADI's commitment to support universities. Already this platform has been used in the Digital Signal Processing Lab and Microprocessor-based Design Lab. Both labs were received very well by the students and provided a wide range of educational material.

Future courses which will use this platform include the digital logic design lab and the hardware description language course (using the FPGA extender board). The goal will be to see the hardware/software co-design tradeoff of utilizing an FPGA versus high-level language. There are also plans to incorporate elements of these tools into the undergraduate Computer Architecture course.

ACKNOWLEDGMENT

The authors would like to thank Mimi Pichey, Giuseppe Olivadoti, Richard Gentile, and Ken Butler from Analog Devices for their generous donation of Blackfin EZ-KIT Lite evaluation boards, software, and extender boards, and for their support of this project. The authors would also like to acknowledge the efforts of Kaushal Sanghai who provided technical support on the BF561.

REFERENCES

[1] http://analog.com/processors/resources/teachingResources.html, *University Program Teaching Resources*, Analog Devices web page.
[2] http://www.demosondemand.com/clients/analogdevices/001/page/, *Blackfin Online Learning and Development*, Analog Devices web page.
[3] http://www.analog.com/processors/epWebCasts/, *Webcasts*, Analog Devices web page.
[4] http://www.analog.com/processors/training/index.html, *Learning & Development*, Analog Devices web page.
[5] http://www.enel.ucalgary.ca/People/Smith/ECE-ADI-Project/Index/originalindex.htm, *ECE-ADI Project Home Page*, University of Calgary.
[6] http://ee.unipr.it/SHARC2116x/, *Hands-on ADI DSP 1-day course*, University of Parma Italy.
[7] Diana Franklin and John Seng, *Experiences with the Blackfin architecture of Embedded Systems Education*, ICSA-2005, Workshop on Computer Architecture Education, 2005.
7
[8] http://www.ghs.com/products/rtos/integrity.html, *INTEGRITY Real-Time Operating System*, Green Hills Software web page.
[9] http://robots.net/article/1866.html, *A New and Improved Handy Board*.
[10] http://www.ghs.com/products/velosity.html, *velOSity Real-Time Operating System*, Green Hills Software web page.
[11] http://www.rtos.com/txtech.asp, *ThreadX Technical Features*, Express Logic website.
[12] http://www.mentor.com/products/embedded_software/nucleus_rtos/index.cfm, *Nucleus Real-Time Operating System*, Mentor Graphics web page.
[13] http://www.unicoi.com/fusion_rtos/rtos_blackfin.htm, *Fusion embedded RTOS for Blackfin*, Unicoi Systems web page.
[14] http://blackfin.uclinux.org, *Blackfin μClinux Project*, web page
[15] http://www.intel.com/design/msa/highlights.pdf, *Micro Signal Architecture from ADI and Intel*.
[16] Tammy Noergaard, *Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers*, First edition, Elsevier Inc, 2006.
[17] Analog Devices, Inc., One Technology Way, Norwood, MA 02062. *ADSP-BF561 Blackfin Processor Hardware Reference*, revision 1.0 edition, July 2005, Part Number 82-000561-01.
[18] Analog Devices, Inc., One Technology Way, Norwood, MA 02062. *ADSP-BF53x/BF56x Blackfin Processor Programming Reference*, revision 1.0 edition, June 2005, Part Number 82-000556-01.
[19] Analog Devices, Inc., One Technology Way, Norwood, MA 02062. *VisualDSP++ 4.0 Getting Started Guide* revision 1.0 edition, January 2005, Part Number 82-000420-01.
[20] Analog Devices, Inc., One Technology Way, Norwood, MA 02062. *VisualDSP++ 4.0 C/C++ Compiler and Library Manual for Blackfin Processor* revision 3.0 edition, January 2005, Part Number 82-000410-03.
[21] Richard Johnsonbaugh and Martin Kaelin, *C for Scientists and Engineers*, Chapter 5, pages 192-195, First edition, Prentice-Hall, 1997.
[22] http://www.rsasecurity.com/rsalabs/challenges/factoring/, *The RSA Factoring Challenge FAQ*, RSA Security web page.