ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION

Paper ID #36065

Design and Implementation of a High-Performance Embedded Course for the next-generation workforce

Dr. Daniel Llamocca, Oakland University

Daniel Llamocca received the B.Sc. degree in electrical engineering from Pontificia Universidad Católica del Perú, in 2002, and the M.Sc. degree in electrical engineering and the Ph.D. degree in computer engineering from the University of New Mexico at Albuquerque, in 2008 and 2012, respectively. He is currently an Associate Professor with Oakland University. His research deals with run-time automatic adaptation of hardware resources to time varying constraints with the purpose of delivering the best hardware solution at any time. His current research interests include reconfigurable computer architectures for signal, image, and video processing, high-performance architectures for computer arithmetic, communication, and embedded interfaces, embedded system design, and run-time partial reconfiguration techniques on field-programmable gate arrays.

Design and Implementation of a High-Performance Embedded Course for the next-generation workforce

Daniel Llamocca
Electrical and Computer Engineering Department
Oakland University, Rochester, MI, USA
llamocca@oakland.edu

Abstract
Embedded system design is covered on a typical undergraduate curriculum in Computer Engineering using standard embedded microcontrollers with limited features. Powerful embedded microprocessors (e.g.: Intel Atom®) offer a wider range of opportunities for learning skills in high demand in industry today, but this is not covered in a standard Computer Engineering curriculum. Oakland University partnered with Intel Corporation to develop and implement an embedded curriculum that meets the needs of the next generation of graduating students entering the workforce.

We present the results of the design and implementation of a new embedded curriculum targeted to powerful embedded processors. This includes the design of a brand-new senior undergraduate course along with a comprehensive tutorial on high-performance embedded programming. We provided students with carefully designed activities that emphasize the optimal usage of powerful microprocessors for embedded applications. The students became proficient in techniques to maximize the performance of an embedded application by optimizing the use of computer resources via techniques such as parallelism and pipelining.

The embedded curriculum was deployed in a classroom and a laboratory setting. The learning materials (course notes, assignments, laboratory experiments, step-by-step tutorials) are made freely available online. Results are encouraging and the course is now offered on a regular basis.

1. INTRODUCTION

Many industry applications rely on embedded real-time programming that uses low-power embedded microcontrollers with limited features. The latest embedded technology, however, uses powerful microprocessors (e.g.: Intel Atom ®) and parallel programming models. Thus, there is a need to train the next-generation workforce with the latest embedded technology.

This was accomplished this via the design and implementation of a research and educational infrastructure for embedded real time processing using an embedded kit (DE2i-150 Development Board that includes an Intel Atom® N2600 processor running Linux-based embedded OS). An earlier work has successfully used this Kit for an introductory embedded programming tutorial[1]. Our new embedded curriculum focuses on industry-relevant applications.

This required a careful assessment of the Computer Engineering undergraduate curriculum at Oakland University to avoid unnecessary overlapping of material. After two years designing the curriculum, we deployed the curriculum in a classroom setting over two semesters. The content can also be delivered in various venues: regular class, seminars, and training sessions that will also offer research opportunities for undergraduate students at Oakland University.

Our goal is to strengthen the Computer Engineering (CE) curriculum at Oakland University by implementing an embedded curriculum geared towards preparing the next-generation

workforce on the features and challenges associated with embedded power microprocessors (e.g.: Intel Atom®), as well as to provide students with a hands-on learning experience.

In this work, we describe the design and implementation phase of the project. The work is divided as follows. Section 2 describes the finalized high performance embedded curriculum. Section 3 covers the implementation stage: the design and structure of the senior undergraduate course as well as the accompanying online tutorial. Section 4 presents the analysis of the work. Section 5 presents the conclusions.

2. FINALIZED EMBEDDED CURRICULUM

In our earlier work[2], we proposed an embedded curriculum based on i) the Intel Hardware and Software Curriculum Focus[3], and ii) a gap analysis and reported employment outcomes of our graduating Computer Engineering (CE) students.

The embedded curriculum was slightly altered based on the results of the implementation stage. It is organized into two main topics listed in Table I: Hardware/Software and Applications Environments. Topics highlighted in green are of utmost importance.

TABLE I. FINALIZED EMBEDDED BASED ON THE INTEL ATOM® PLATFORM

Hardware/Software	Real Time Programming	Hard Real-Time, Soft-Real Time	Interrupts: Sources, ISRs, Latency
		Run to completion/Pre-Emption	Direct Memory Access Controllers
		Real-Time Operating System	Application: Digital Display for Auto Navigation
	Multi-Core/Multi-Threading	Elements of Parallel programming and multi-threading	*Data/Task Parallelism*
			Inter-process communication, thread synchronization
			Re-Entrant and Thread Safe Programming
		NUMA Programming	Programming with threading APIs
		Threading building blocks	Software development tools for multi-threading
		Debugging and testing multi-threaded applications, common parallel prog. problems	
		Applications: Image filtering, beamforming for smart antennas, multi-sensor fusion	
	Virtualization	Hypervisors	PCI-SIG I/O Virtualization (IOV)
		CPU virtualization, Memory virtualization	Single Root IOV (SR-IOV)
		Network, I/O Virtualization	Device emulation, Interrupt Delivery
		Application: Dual operation: navigation system and back seat displays	
	Endian Neutral Programming	Code Portability	Network Byte Ordering
		Compile Time Controls	Data Storage and Shared Memory
		Byte Swap Macros	Data Transfer, Data Types
	Firmware	System Initialization – Bootloaders/BIOS	Microcode
		Firmware and Driver Development	Application Programming Interface (API)
Applications Environments	Networking Applications	Network Stack/OSI Model	TCP/IP protocol, IPv4, IPv6
		Wireless 3G/4G technologies, Bluetooh	Ethernet/IEEE 802.x specifications
	Embedded SW Dev. and Debug Tools	Open source vs proprietary tools	Assemblers/compilers/linkers
		Debuggers, JTAG debug	Single Stepping
		Software profiling + code coverage tools	Virtual memory mapping
	Security and Secure apps	Internet Protocol Security (IPSec)	Private and Public-key encryption
		Secure Sockets Layer (SSL), Open SSL	Security Algorithms (DES, 3 DES, AES)
	Reliability & Serviceability, Safety and Certification	ECC protection, CRC checksums	Partitioning/domaining of computer components
		Lock-step to perform master-checker	Computer clustering capability
		Avoid single point of failures	Virtual machines
		Hot swapping of components	Failover capability
	Power Aware Applications	Dynamic Power Management (DPM)	Dynamic Voltage/Frequency Scaling (DVS)
		Profile of application over memory banks	Shutting down unused peripherals
		Power-aware scheduling	System Sleep Modes

3. IMPLEMENTATION OF THE EMBEDDED CURRICULUM

There are several avenues for the delivery of the contents: Seminar Series, Summer Workshop, Summer Research Experience for Undergraduates, and an Elective Course in Computer Engineering. Based on prior experience developing courses[4], we decided to design a course.

The partnership with Intel® included support during the development of the curriculum adjusted to the needs of Oakland University. In addition, we received feedback as to which topics were important to include in the course. Finally, Oakland University also received support in the laboratory materials (embedded processing kits).

The course ended up including a subset of topics from Table I. In addition, it was decided that the course be target towards senior undergraduates (and graduates) who already have some exposure to programming on microprocessors. The focus is on "high-performance embedded programming" using parallel programming frameworks and real-time programming.

3.1 Hardware and Software Tools

We utilize the DE2i-150 FPGA Development Kit, a full-featured computer that combines high-performance processing (Intel® Atom™ N2600 processor) and high configurability (Altera Cyclone IV GX FPGA). These two devices are connected via PCI Express interface. Each device includes a range of peripherals connected to them. Table II gives a brief overview.

TABLE II. BRIEF OVERVIEW OF THE FEATURES OF THE TERASIC DE2I-150 FPGA DEVELOPMENT KIT.

Microprocessor	**FPGA System**
Intel® Atom™ Dual Core Processor N2600, 1.6 GHz	Cyclone IV EP4CGX150DF31 FPGA
1 MB L2 cache (512 KB per core)	Three 50 MHz oscillator clock inputs
Intel® NM10 Express Chipset	128 MB SDRAM
2 GB DDR3 SDRAM	64 MB Flash Memory
64 GB SSD	4 MB SSRAM
HDMI, VGA output	SD Card: SPI and 4-bit SD mode
4 USB ports	Display: VGA, LCD
Gigabit Ethernet, 802.11 b/g/n	Ethernet port

Fig. 1 (right) depicts this board (front side) The embedded Intel® Atom™ processor inside this board allows for high-performance and parallel programming. The course contents do not depend on the board, which can be updated when required.

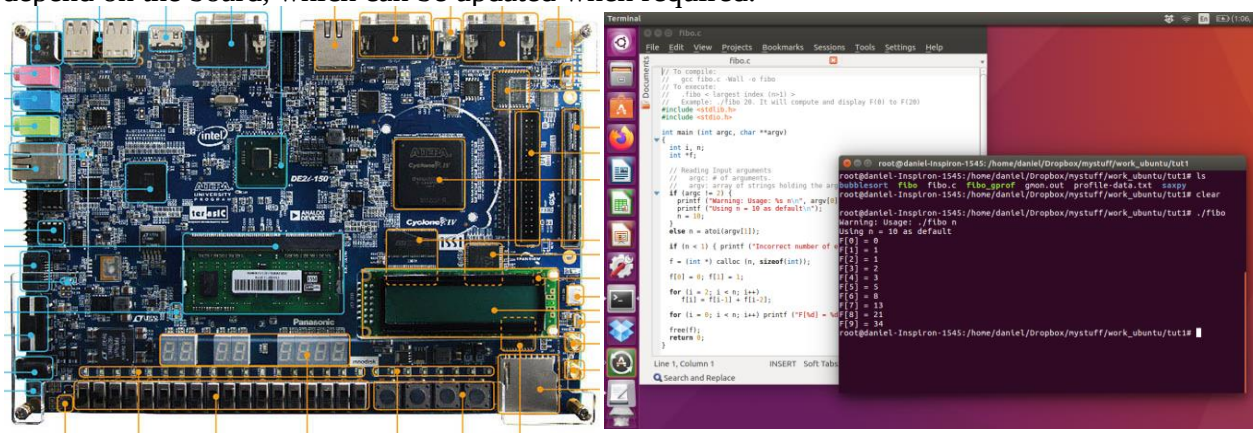

Figure 1. Hardware and Software Platform. (right) Terasic DE2i-150 FPGA Development Kit.(left) Ubuntu Linux

As for the software tools, we installed Ubuntu 12.04.4 version on the boards. Software applications are written in C and C++ (requiring the *gcc* and *g++* compilers). For application development, students can use a simple text editor or an IDE (e.g.: CLion).

3.2 Course Structure

Table III outlines the organization of the course. There are seven units (with available lecture notes), assignments, and step-by-step tutorials (Refer to Section 3.3.).

TABLE III. OUTLINE OF COURSE TOPICS, ASSOCIATED ASSIGNMENTS AND EXTRA MATERIAL

Unit	Topic	Assignments	Associated Material
1	Embedded Multi-Core Systems	Homework 1 Laboratory 1	Lecture Notes – Unit 1 Tutorial 1: Getting Started with the Hardware & Software Platform
2	C/C++ Language Programming Fundamentals	Homework 2 Laboratory 2	Lecture Notes – Unit 2 Tutorial 2: C/C++ Programming
3	Multi-threaded applications	Laboratory 3	Lecture Notes – Unit 3 Tutorial 3, Tutorial 4: pthreads
4	Multi-core applications	Homework 3 Laboratory 4 Laboratory 5 Laboratory 6 Laboratory 7	Lecture Notes – Unit 4 Tutorial 5: Threading Building Blocks (TBB)
5	Real-Time Programming	Homework 4 Laboratory 8	Lecture Notes – Unit 5 Tutorial 6: Real-Time Programming
6	Design and Optimization of Embedded Real-Time Systems		Lecture Notes – Unit 6
7	Applications		Lecture Notes – Unit 7

3.2.1. Topics

For each of the seven units, there are extensive lecture notes (.pdf).

- *Embedded Multi-Core Systems*: An overview of microprocessors for embedded applications is presented, including more details of the Intel® Atom™ processor.
- *C/C++ Language Programming Fundamentals*: We cover specific C/C++ constructs required in order to utilize the parallel programming frameworks (e.g.: *pthreads*, *tbb*).
- *Multi-threaded applications*: Here, we cover a standardized programming interface for multi-threads. This is direct thread specification.
- *Multi-core applications*: Here, we cover the Intel Threading Building Blocks, which is a template for parallel programming on multi-core processors. Computations are broken down into tasks rather than into specific threads.
- *Real-Time Programming*: Here, we cover some introductory topics on real-time programing.
- *Design & Optimization of Embedded Real-Time Systems:* Some techniques on single-core and multi-core performance optimization are explored: system profiling, application profiling.
- *Applications:* Real-world applications are explored: Convolutional Neural Networks, Adaptive Beamforming, Cylinder Pressure Estimation.

3.3.2 Assignments

The evaluations were organized into four homeworks, eight laboratories, a take-home midterm exam, and a final project. Students can form groups of two in the final project.

Laboratory assignments were focused on the development of embedded applications using the parallel frameworks (*pthreads*, TBB). Table IV describes each laboratory experiment.

TABLE IV. LABORATORY EXPERIMENTS

Lab	Description
1	Board Setup, Basic Utilities Implementation of basic applications on C: numerical sequence, computation of π.
2	C/C++ Programming. Image Convolution in C. Neural Network Layer Implementation in C++ using functors.
3	*pthreads*: Centered moving average (window size = 7): implementation and analysis with varying number of threads
4	TBB: *parallel_for*. Gamma Correction applied to a grayscale image.
5	TBB: *parallel_for + parallel_reduce*: Neural Network Layer Implementation.
6	TBB: *parallel_for + parallel_reduce*: Image Histogram computation
7	TBB: *parallel_pipeline*. Streaming vectors. Computation time analysis with different number of vectors and vector sizes.
8	Real-Time Programming: Handling signals and RTC configuration

In their midterm exam, students were given a take home test where they needed to implement a popular parallelizable application (e.g.: SAXPY) using both *pthreads* and TBB programming frameworks. They were also asked to analyze the computation time given different number of threads and different problem sizes.

In the final project, students were evaluated in their ability to i) implement a scalable application using either *pthreads* or TBB, and ii) perform extensive computation time comparisons based on different application parameters and problem sizes. In addition, students submitted a final report that includes their methodology and results.

3.3.3 Laboratory Equipment

Through the partnership with Intel®, we were donated 26 DEI2i-150 Development Kits. We planned to fit a laboratory room with the equipment (along with keyboards, mice, and screens). Due to the online format of delivery, we let students borrow the boards for the entire semester.

3.3.4 Textbook

Given the extensive material generated (lecture notes, tutorials, source code), students are not required to use a textbook. The syllabus did list some references (most of them freely available).

3.3 Tutorial Structure

An accompanying tutorial was developed. A total of 8 step-by-step tutorials are available. For each tutorial, there are pdf notes, and source code. The topics are listed as follows:

1. *Tutorial 1: Getting Started with the Hardware and Software Platform*. Getting Started with the DE2i-250 Development Kit and Ubuntu Linux: installation, setup, examples.
2. *Tutorial 2: C/C++ Programming.* 2D Convolution in C, neuron implementation in C+.
3. *Tutorial 3: Multi-threading.* Example: basic ones, 2D convolution. Mutexes (dot product)
4. *Tutorial 4: Multi-threading.* Example: Matrix multiplication
5. *Tutorial 5: TBB – parallel_for.* Multiple basic examples: element-wise vector operations, 3-element moving average, grayscale morphological operations.

6. *Tutorial 6: TBB – parallel_reduce.* Examples: array accumulation, computation of π, dot product, maximum out of each row in a matrix, add a group of vectors element-wise.
7. *Tutorial 7: TBB – parallel_pipeline.* Modulus of two vectors, sum of squared values in a vector, processing incoming vectors,
8. *Tutorial 8: Real-Time Programming.* Handling signals (setup, catch), configure and test the Real-Time Clock.

3.4 Online dissemination of material

For the wider community, we are making the course material and the tutorials available and hosted at the Reconfigurable Computing Research Laboratory website at Oakland University. The developed material was organized as follows: i) seven units with lecture notes, ii) four homeworks with solutions and a midterm exam, iii) eight laboratory experiments, and iv) final project guideline along with students' final reports and presentations. The entire material is available online at the class website (this is for Fall 2021).

We also developed step-by-step tutorials on embedded programming with the Intel® Atom. The material, that includes eight units (each with pdf notes and source code files), is available at the High-Performance Embedded Programming with the Intel Atom Platform website.

4. ANALYSIS AND RESULTS

4.1. Course Assessment

We grouped the course objectives into seven learning outcomes that include competence in topics, proficiency in multi-threading/multi-core programming, and oral/written presentation. Table V lists the learning outcomes and the associated class activities.

TABLE V. LEARNING OUTCOMES AND ASSOCIATED ACTIVITIES

Student Learning Outcomes	**Activities**
Describe the generalized architecture of the Intel Atom® microprocessor.	Laboratory 1 Homework 1
Implement software applications with C/C++ on Ubuntu Linux.	Laboratory 2 Homework 2
Implement real-time embedded applications on Ubuntu Linux.	Laboratory 8 Homework 4
Design and implement multi-threaded software applications.	Laboratory 3 Midterm Exam
Design and implement multi-core applications to enable parallelism and pipelining.	Laboratory 4,5,6,7 Homework 3 Midterm Exam
Design applications that utilize the computer resources in a scalable fashion	Laboratory 7 Final Project
Work in a team environment to design a real-time multi-threaded embedded application and communicate the results in a written report and an oral presentation.	Final Project

Two main reasons have been identified that improve student engagement, learning outcomes, and student success[5]:

- Multi-threaded/multi-core embedded software design: The class builds upon students' prior knowledge on microprocessors and allows them to explore parallelization strategies.

- Opportunities for research in state-of-the-art topics: We covered research on cutting edge topics on embedded programming: task-based specification (Intel® Threading Building Blocks).

5. OUTCOMES

The course, called "High-Performance Embedded Programming" has been taught during the Fall 2020 semester (online format), and in Fall 2021. It has become a regularly schedule course.

5.1. Student Projects

This is the list of projects that students successfully completed, where they developed the applications using TBB/pthreads, and performed time comparisons based on application parameters and sizes.

- *Grayscale Image Morphology*: Implementation of Dilation, Erosion, Opening, Closing, Boundary Extraction.
- *Matrix Multiplication*: Implementation and of the Strassen Algorithm.
- Convolutional Neural Network: 2 convolutional layers (6 @ 24x24, 24 @ 8x8, 3 fully connected layers (384, 128, 10).
- A* Search Algorithm: A GUI was included that showed in real-time how the algorithm calculates the optimal path.
- *Thermal Analysis*: Finding and evaluation of heat sources in thermal imaging. Proper use of parallel pipelines.
- *Maldelbrot*: This implementation also included the use SIMD instructions, resulting in high speedups (~15X). Given the successful results of including SIMD, we plan to include SIMD as part of the course topics in future implementations.

5.2 Student Survey

At the end of the semester, students completed an anonymous survey. The following is a selection of the questions asked.

- Q1: The instructor did a good job of making the objectives of the course clear to me.
- Q2: The instructor stimulated and deepened my interest in the subject.
- Q3: The instructor motivated me to do my best work.
- Q4: Value of the laboratory component of the course.
- Q5: Overall rating of this course as a learning experience.

For each of the questions (Q1-Q5), all students provided the same ratings, and thus the average rating for each question was identical. The results per semester were as follows:

- Fall 2020: 5 registered students, 3 responses. Average Rating (Q1,Q2,Q3,Q4,Q5): 4.7/5.0
- Fall 2021. 5 registered students, 4 responses. Average Rating (Q1,Q2,Q3,Q4,Q5): 5.0/5.0.

The results show that students rated their experience very highly. However, we note that the number of responses is small. In the same survey, students also provided comments regarding the course. They were excited about the contents and the developed material (lecture notes and assignments). They were very engaged with the laboratory experiments. In particular, a senior undergrad mentioned that they wanted to pursue this area of research in their masters'. Due to the small number of students, most students worked on their own, which did not facilitate team building. We expect this issue to be resolved with increased enrollment in the coming semesters.

6. CONCLUSIONS

We presented results and analysis resulting from the implementation of a course in high performance embedded programming. The covered material, laboratory experiments, and final projects were successful in terms of student engagement and learning outcomes.
Students rated their overall experience highly. Based on student feedback and instructor experience, there is room for improvement that will be addressed in future implementation of this course. We plan to progressively add more topics (virtualization, SIMD, onetbb) to cover the Embedded Curriculum in Table I. The freely available class material and tutorials will be updated on a regular basis.

Bibliography

1. G. Drayer and A. Howard, "Evaluation of an Introductory Embedded Systems Programming Tutorial using Hands-On Learning Methods", *ASEE Annual Conference & Exposition*, 2014.
2. Alaswad, D., Llamocca, D., Gillespie, B., "Towards an Embedded Systems Curricula for the next-generation workforce", *2019 ASEE North Central Section Conference*, Grand Rapids, Michigan, March 2019.
3. Embedded University Program: Hardware and Software Curriculum Focus, *Intel Corporation*, March 2018.
4. Llamocca, D., "Design and Implementation of a Reconfigurable Computing Course for efficient Hardware/Software Co-Design in Reconfigurable Systems", *2016 ASEE Northeast Section Conference*, Kingston, Rhode Island, April 2016.
5. Wieman, C., Gilbert, S., "The Teaching Practices Inventory: A New Tool for Characterizing College and Univ. Teaching in Mathematics and Science", *CBE-Life Sciences Education*, vol.13, no.3, pp. 552-569, 2014.