Towards an Embedded Systems Curricula for the next-generation workforce

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

Byron Gillespie
Intel Corporation, Chandler, AZ, USA
byron.r.gillespie@intel.com

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: real-time programming, virtualization, operating systems. However, this is not covered in a standard Computer Engineering curriculum.

Oakland University and Intel corporation have partnered in order to develop and implement an embedded curriculum that meets the needs of the next-generation of graduating students entering the workforce. We first evaluate the undergraduate Computer Engineering curriculum at Oakland University with respect to the Hardware and Software curriculum proposed by Intel.

This work presents an embedded curriculum tailored to the needs of both our graduating students and industry. The plan is to deliver the contents via various venues: content-updating of some courses, seminar series, a summer workshop, and summer research experiences for undergraduates, and an elective course. Our goal is to train the next-generation workforce with the latest embedded technology on current industry-relevant applications.

1. INTRODUCTION

Embedded real-time programming is a well-known approach for the implementation of applications that are ubiquitous in many industries. It usually relies on low-power embedded microcontrollers with limited features. Another interesting approach is to use powerful microprocessors for embedded applications. This is the latest embedded technology utilized by industry today. However, most embedded curriculums in Computer Engineering are geared towards embedded microcontrollers. There exists then a need to properly train the next generation workforce with the latest embedded technology, e.g., the Intel Atom® platform[1].

We plan to accomplish this via the design and implementation of a research and educational infrastructure for embedded real time processing using the DE2i-150 Development Board that includes an Intel Atom® N2600 processor running a Linux-based embedded OS. An earlier work has successfully used this board for an introductory embedded programming tutorial[2]. The embedded group at Intel has developed a curriculum called "Hardware and Software Curriculum Focus" which itemizes the skills that an Ideal Graduate should have. To address this, we propose a new embedded curriculum that focuses on industry-relevant applications.

This requires a careful assessment of the Computer Engineering undergraduate curriculum at Oakland University to avoid unnecessary overlapping of material. We envision that a complete curriculum will be ready to be deployed in two years. The content will 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 developing an embedded curriculum geared towards effectively preparing the next-generation workforce on the features and challenges associated with the Intel Atom® platform, as well as to provide students with a hands-on learning experience. Here, we describe the exploration phase of the project, where we lay out the groundwork for implementation of the new embedded curricula. We describe both the Oakland University curriculum and the Intel curriculum, a gap analysis between the two, and a resulting draft high-performance embedded curriculum. The rest of the manuscript is divided as follows. Section 2 describes the Oakland University Computer Engineering undergraduate curriculum. Section 3 describes the Intel curriculum. Section 4 presents the gap analysis. Section 5 shows the proposed embedded curriculum. This includes some early work on the tutorials, guides, and examples. Section 6 presents the conclusions.

TABLE I. OAKLAND UNIVERSITY COMPUTER ENGINEERING CURRICULUM. MOST COURSES ARE 4 CREDITS

Eng. Core (21 credit)		EGR1200	Engineering Graphics and CAD	
		EGR1400	Computer Problem Solving in Engineering and Computer Science	Intro. C# and MATLAB. Applications in mechanical, electrical, and computer eng.
		EGR2400	Intro. to Electrical and Computer Eng.	
		EGR2500	Introduction to Thermal Engineering	
		EGR2600	Intro. to Industrial and Systems Eng.	
		EGR2800	Design and Analysis of Electromechanical Systems	Intro.to electromechanical systems (statics, linear, rotational dynamics) and μprocessors.
Professional Subjects (36 credits)		CSI2290	Introduction to Data Structures in C	C programming including arrays, structures, pointers. Data structures: stacks, queues, lists. Sorting and searching algorithms.
		ECE2005	Electric Circuits	
		ECE2700	Digital Logic Design	Boolean Algebra, comp. arithmetic, combinational and synchronous sequential circuits. Intro. to VHDL.
		ECE3100	Electronic Circuits and Devices	
		ECE3204	Signals and Systems	
		ECE3710	Computer Hardware Design	Design of digital circuits and systems for computers, I/O, and specific applications.
		ECE3720	Microprocessors	Microcomputer systems: sensor interfacing, I/O, memory systems.
		ECE4721	Embedded System Design	Embedded system design and analysis using ARM Cortex CPU. Architecture, instruction set, memory management units, exceptions.
		ECE4999	Senior Design	
Professional Electives (12 credit)	Select one	CSI3640	Computer Organization	RISC/CISC architectures, assembly language, assemblers, loaders, linkers, ALU, I/O, memory organization, cache, virtual memory.
		CSI3610	Design and Analysis of Algorithms	Divide-and-conquer, dynamic programming, and greedy algorithms. Computational complexity. Algorithms for parallel and distributed architectures
	Select two	CSI3370	Software Engineering and Practice	Select from junior and senior classes in Computer Science as well as Electrical and Computer Engineering. Here, we list the relevant Computer Science classes that are listed suggestions in the catalog program description.
		CSI4240	Cloud Computing	
		CSI4360	Concurrent and Multi-Core Programming	
		CSI4480	Information Security Practice	
		CSI4500	Fundamentals of Operating Systems	
		CSI3430	Theory of Computation	

2. COMPUTER ENGINEERING CURRICULUM DESCRIPTION

Oakland University's Computer Engineering program is divided as follows: i) General Education (28 credits), Mathematics and Science (32 credits), ii) Engineering Core (21 credits), Professional Subjects (36 credits), Professional Electives (12 credits). These are listed in Table I. Courses relevant to the proposed Intel curriculum (see Section 3) are highlighted in green, whereas relevant elective courses are highlighted in gray.

The curriculum has a strong emphasis in Computer Hardware Design (8 credits) and embedded systems (8 credits). In addition, students must take at least 2 courses (8 credits) in Computer Science and can select up to 2 more courses in Computer Science.

TABLE II. INTEL HARDWARE AND SOFTWARE CURRICULUM: CS/CE OBJECTIVES[3]

Hardware/Software	Real Time Programming	Hard Real-Time, Soft-Real Time	Interrupt Service Routines, Priorities, Masking	
		Run to completion/Pre-Emption	Interrupt Latency	
		Direct Memory Access Controllers	Interrupt Sources (Inter-processor, hardware arbitration, advanced programmable int. controller)	
		Real Time Clock		
	Endian Neutral Programming	Code Portability	Network Byte Ordering	
		Compile Time Controls	Byte Swap Macros	
		Data Storage and Shared Memory	Data Transfer, Data Types	
	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 OpenMP	Software development tools for multi-threading	
		Debugging and testing multi-threaded applications, common parallel prog. problems		
	Firmware	System Initialization – Boot loaders/BIOS	Microcode	
		Firmware and Driver Development	Application Programming Interface (API)	
		ROM image		
	Virtualization	Hypervisors	PCI-SIG I/O Virtualization (IOV)	
		CPU virtualization, Memory virtualization	Single Root IOV (SR-IOV)	
		Network Virtualization	Device emulation	
		I/O Virtualization	Interrupt delivery	
Applications Environments	Security and Secure Applications	Internet Protocol Security (IPsec)	Private and Public-key encryption	
		Secure Sockets Layer (SSL)	Applications	*Statefull Firewall*
		Security Algorithms (DES, 3DES, AES)		*Intrusion Detection/Prevention*
		Cryptographic hash function	Open source implementations (OpenSSL)	
	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	
	Networking Applications	Network Stack/OSI Model	Bluetooth	
		Wireless 3G/4G technologies	Ethernet/IEEE 802.x specifications	
		TCP/IP protocol, IPv4, IPv6	Routing protocols, Proxy	
	Embedded Software Dev. and Debug Tools	Cross Development Environments	Open source vs proprietary tools	
		Assemblers/compilers/linkers	JTAG debug	
		Debuggers	Single Stepping	
		Software profiling + code coverage tools	Virtual memory mapping	
	Reliability & Serviceability, Safety and Certification	Parity or ECC protection	Partitioning/domaining of comp. components	
		CRC checksums for data	Computer clustering capability	
		Lock-step to perform master-checker	Virtual machines	
		Avoid single point of failures	Temperature Sensors	
		Hot swapping of components	Failover capability	

3. INTEL HARDWARE AND SOFTWARE CURRICULUM

The Intel Hardware and Software Curriculum focus[3] is organized into five main topics: i) Hardware/Software, ii) Applications Environments, iii) Hardware Interfaces, iv) Hardware Development, and v) Hardware/Software Partitioning. Since the focus is on embedded applications, we selected two main topics to focus on: Hardware/Software and Applications Environments. Table II lists the topics. Topics that are fully or partially covered by the OU CE curriculum are highlighted (green: mandatory, gray: optional).

4. GAP ANALYSIS

The CE curriculum has a strong emphasis in Embedded Systems and Digital Hardware Design (see Table I). However, many topics listed in the Intel HW/SW Curriculum (Table II) were not covered. The specific gaps in the CE curriculum are listed in Table III. The goal is not to cover all these gaps, but rather to identify areas where there is room for improvement.

TABLE III. GAPS AT OAKLAND UNIVERSITY COMPUTER ENGINEERING (CE) CURRICULUM

	Topics	**Notes**
Not covered at all	Firmware Virtualization Power aware applications Networking applications	Networking is missing. CE students can take *CSI2470 – Introduction to Computer Networks* (Computer Science class), but CSI2470 is missing from the list of Computer Science electives in the CE program.
Partially covered	Real-Time Programming Endian Neutral Programming Multi-core/multi-threading Security and Secure Applications Reliability and Serviceability	CE students can only cover Multi-core/multi-threading and Security and Secure Applications if they take these two Computer Science elective courses: *CSI4360 – Concurrent and Multi-Core Programming*, *CSI4480 – Information Security Practice*.
Covered to a significant extent	Embedded Software Development and Debug Tools	Though CE students cover topics in this area, there is room for upgrading the topics and including new topics (e.g.: virtual memory mapping, single stepping)

Some gaps can be addressed by minor changes to the CE curricula. We can add topics on real-time programming, endian neutral programming, multi-core/multi-threading on the senior course *Embedded System Design*. We can also list Introduction to Computer Networks as a Computer Science Elective in the CE program. However, most of the gaps can be addressed via the proposed embedded curriculum. To this effect, a draft of the Embedded Curriculum based on the Intel® Atom platform is detailed in Section 5.

4.2 Reported Employment Outcomes of OU graduating students in Computer Engineering

Identifying where most of graduating students work allows us to tailor the applications to the needs of those students. The First-Destination Survey, administered annually by Oakland University, captures the post-graduation outcomes of Oakland University SECS graduating students. The survey content, timeline and procedures align with guidelines published by the National Association of Colleges and Employers (NACE). The most recent surveys are from Summer 2016, Winter 2016, and Spring 2017 semesters. It reported that graduating students work in the following industries: automotive (67%), engineering (17%), other (16%).

The main takeaway of this First-Destination Survey is that CE graduating students work overwhelmingly in the automotive industry (auto manufacturers and suppliers). Thus, the proposed embedded curriculum will include applications relevant to the automotive industry today (e.g.: navigation, multi-sensor fusion, digital displays, video processing, deep learning, connected vehicles, autonomous vehicles).

5. PROPOSED EMBEDDED CURRICULUM (FIRST DRAFT)

Based on the gap analysis and the reported employment outcomes of OU graduating CE students, a draft curriculum is presented in Table V. It is heavily based on the Intel Hardware and Software Curriculum of Table II. Topics already covered by our Computer Engineering curriculum are not included. We also established priorities of some topics over others. Topics highlighted in green are of utmost importance. Topics highlighted in red could be eventually discarded from the curriculum. We also list applications.

TABLE V. PROPOSED CURRICULUM (FIRST DRAFT) 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 OpenMP	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 Virtualization	Device emulation
		I/O Virtualization	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 – Boot loaders/BIOS	Microcode
		Firmware and Driver Development	Application Programming Interface (API)
Applications Environments	Networking Applications	Network Stack/OSI Model	Bluetooth
		Wireless 3G/4G technologies	Ethernet/IEEE 802.x specifications
		TCP/IP protocol, IPv4, IPv6	Routing protocols, Proxy
	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 Applications	Internet Protocol Security (IPSec)	Private and Public-key encryption
		Secure Sockets Layer (SSL)	Open Source implementations (Open SSL)
		Security Algorithms (DES, 3 DES, AES)	Cryptographic hash function (SHA-x, MD5)
	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

The plan is to deliver the contents via:

- Set of Tutorials: These will cover the most important topics in Table V.
- Elective Class in Computer Engineering: This will be a senior undergraduate course. The design and implement will follow the framework of an early work[4].
- Seminar Series, Summer Workshop, Summer Research Experiences for Undergraduates.

5.1 Implementation of the Embedded Curriculum

This curriculum implementation will incorporate practices that support student learning and teaching effectiveness[5], such as knowledge organization, motivation, practice, and group learning. It will be implemented on the Terasic DE2i-150 Development Kit[6]. Among the planned activities (some have already started), we list:

- Getting started with the Hardware and Software Tools: A tutorial is available on Yocto Linux[7] installation on the Development Kit as well as working with the file system.
- Implementation of a basic examples: a tutorial was drafted on how to deal with the basics of C programming: arrays, loops, functions, dynamic memory allocation, makefiles, etc.
- Multi-core/Multi-threading: The following common parallel programming problems have been implemented as multi-threading applications: matrix multiplication and 2-D convolution. Two tutorials are available. Industry-relevant applications need to be covered, e.g.: sensor fusion, beamforming, convolutional neural networks. Other topics to cover are: Treading Building Blocks, OpenMP, Programming with threading APIs, NUMA Programming
- Real-Time Programing examples: This is yet to be started. Here, we will include examples dealing with this topic, such as interrupts, real-time clock, direct memory access.
- Virtualization: This is yet to be started: This is a very important topics that includes hypervisors, network virtualization, CPU virtualization, and I/O virtualization.

6. CONCLUSIONS

A new embedded curriculum has been presented tailored to high-performance microprocessors that can be used in an embedded system. This was accomplished by assessing the gaps in the undergraduate curriculum and by targeting industry applications in the local area. We have laid out a development plan (topics, tutorials, venues) and work is currently underway.

The proposed curriculum is not meant to replace a course in the computer engineering program, but rather complement the program by offering training and research opportunities to undergraduate students in the latest embedded technology. An elective is an option pending the results of the summer workshop and research experience for undergraduates.

Bibliography

1. Intel® Atom™ Processor D2000 and N2000 Series, *Intel Corporation*, July 2012.
2. G. Drayer and A. Howard, "Evaluation of an Introductory Embedded Systems Programming Tutorial using Hands-On Learning Methods", *Proceedings of the 121st ASEE Annual Conference & Exposition*, June 2014.
3. Embedded University Program: Hardware and Software Curriculum Focus, *Intel Corporation*, March 2018.
4. D. Llamocca, "Design and Implementation of a Reconfigurable Computing Course for efficient Hardware/Software Co-Design in Reconfigurable Systems", *Proceedings of the 2016 ASEE Northeast Section Conference*, 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
6. Terasic DE2i-150 Getting Started Guide, Terasic Corporation.
7. Rudolf J. Streif, *Embedded Linux Systems with the Yocto Project*, 1st ed., Prentice Hall, 2016.