RECRLab
Reconfigurable Computing Research Laboratory

Towards an Embedded System Curricula for the next-generation workforce

DANIYAH ALASWAD DANIEL LLAMOCCA, BYRON GILLESPIE

Electrical and Computer Engineering Department,

Oakland University

March 23rd, 2019

OAKLAND UNIVERSITY

Outline

- *Introduction*
- *Development*
- *Proposed Embedded Curriculum*
- *Implementation Stage*

Introduction

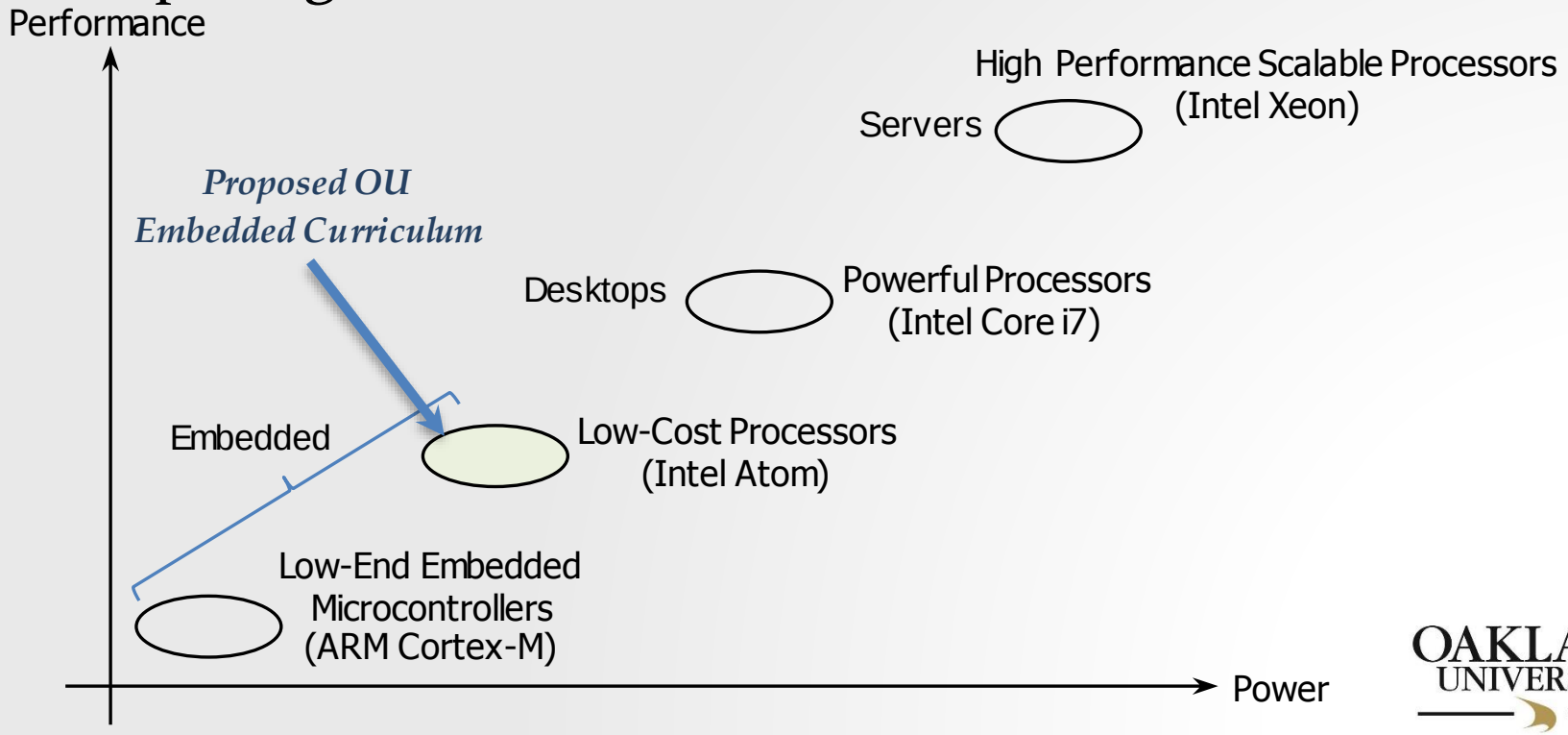
- ***Embedded Design*** *is covered in a typical undergraduate curriculum in Computer Engineering using standard low-power microcontrollers.*

- ***Powerful embedded microprocessors*** *can be used instead (e.g.: Intel Atom, ARM Cortex-A9). They allow for i) a wider range of opportunities for learning skills in high demand in industry today, and ii) focus on high-end emerging industry-relevant applications and workloads.*

- *We propose a* ***high-end embedded curriculum*** *tailored to the needs of both our graduating students and industry. The goal is to train the next-generation workforce with the latest embedded technology using current industry-relevant applications.*

Development

- Plan: develop and implement an embedded curriculum that meets the needs of the next-generation of graduating students entering the workforce.
- Platform: Intel Atom®. The focus in on emerging industry-relevant applications and workloads that push the envelope of computing.



Development

- The Intel® Hardware and Software Curriculum is organized into five main topics, of which we focus on two:
  - ***Hardware/Software***: *Real-Time Programming, Endian Neutral Programming, Multi-Core/Multi-Threading, Firmware, Virtualization*
  - ***Applications Environments***: *Security and Secure Applications, Power Aware Applications, Networking Applications, Embedded Software Development and Debug Tools, Reliability and Serviceability, Safety and Certification.*
- Gap Analysis: The Comp. Engineering (CE)undergraduate curriculum at Oakland University has a strong emphasis in Embedded Systems and Digital Design. Still, some topics listed in the Intel® HW/SW Curriculum were not covered:
  - *Not covered: Firmware, Virtualization, Power Aware Applications, Networking Applications*
  - *Partially covered: Real-Time Programming, Endian Neutral Programming, Multi-Core/Multi-Threading, Security and Secure Applications, Reliability and Serviceability.*

Proposed Embedded Curriculum

- Topics already covered by the OU CE curriculum were not included.
- Though we would like to cover all these topics, when deploying the curriculum, we will start by covering the most important ones.

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 and 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, Memory, Network, I/O, virtualization	Device emulation, Interrupt Delivery
		***Application*: Dual operation: navigation system and back seat displays**	
	Endian Neutral Programming	Code Portability, Compile Time Controls	Data Transfer, Data Types
		Byte Swap Macros, Network Byte Ordering	Data Storage and Shared Memory
	Firmware	System Initialization – Boot loaders/BIOS	Microcode
		Firmware and Driver Development	Application Programming Interface (API)
Applications Environments	Networking Applications	Network Stack/OSI Model, TCP/IP protocol: IPv4, IPv6	Bluetooth
		Wireless 3G/4G technologies	Ethernet/IEEE 802.x specifications
	Embedded SW Dev. and Debug Tools	Open source vs proprietary tools	Assemblers/compilers/linkers. Single Stepping
		Debuggers, JTAG debug. 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). Security Algorithms (DES, AES, etc)	Open Source implementations (Open SSL)
	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

Implementation

The activities leading to the implementation and deployment of the Embedded curriculum are listed as follows:

- Basic Command of the Hardware and Software Tools
  - *Yocto Linux and the Development Board (Terasic DE2i-150 Dev. Kit). A tutorial is has been developed.*
  - *Basic software applications. A tutorial has been completed.*
- Development of Sample Applications and Documentation
  - *Multi-threading: matrix multiplication, 2-D convolution, Convolutional Neural Networks (in progress). A set of tutorials have been completed.*
  - *Virtualization, Real-Time Programming and Multi-Core (in progress)*
  - *Others (not yet started).*
- Deployment of the curriculum: **not all the topics in the proposed Embedded Curriculum will be covered.*
  - *Proposed Venues: Seminar Series, Summer Workshop, Elective Course.*
  - *It was decided to start with an Elective course once all the sample applications have been tested and documented.*

Implementation

Elective Course: Here, we list some tentative features of this senior-level elective course.

- Structure: 4 homeworks, 6 laboratory experiments, 1 Midterm Exam, and 1 Final Project.
- List of topics to cover:
  - *Classic Topics: real-time programming, multi-core/multi-thread computing, virtualization, and endian neutral programming.*
  - *Scale concept of systems (from embedded microcontrollers to supercomputers), pushing the envelope of computing.*
  - *Special emphasis: emergency applications and workloads that are industry-relevant.*
- Set of Tutorials on the Development Tools (next slide).
- Plan to: i) fit a room with equipment (15-20 boards), ii) ensure that the teaching materials can be re-purposed for a different Intel microprocessor and/or Dev. Board.

Implementation

Tutorial Structure: In the table, items shaded in green are completed sections (software routines tested, tutorials developed). Shaded in purple: currently in progress.

Getting Started	▪ Getting Started with the DE2i-250 Development Kit and Yocto Linux: accessing features of the system, editing text files.		
	▪ First Example in C: code editing and compilation		
	▪ Conditional statements, loops		
	▪ Arrays, pointers, functions. Use of makefiles		
	▪ Dynamic memory allocation		
Multi-threads/ multi-core	Multi-threads	▪ pthreads: Basic Example.	
		▪ Matrix multiplication. Performance comparison with non-threaded case.	
		▪ 2D Convolution. Performance comparison with non-threaded case.	
		Industry-relevant applications	▪ Convolutional Neural Networks
			▪ Sensor Fusion
			▪ Adaptive Beamforming
			▪ Image Filtering: Spatial and Radon-based 2D Convolution
	▪ Threading Building Blocks (TBB)		
	▪ OpenMP		
	▪ NUMA Programming		
Real-Time Programming	▪ Real-Time Operating System (RTOS)		
	▪ Interrupt: Sources, ISRs, Latency		
	▪ Direct Memory Access		
Virtualization	▪ Hypervisors		
	▪ CPU virtualization		
	▪ Memory virtualization		
	▪ I/O Virtualization		

Implementation

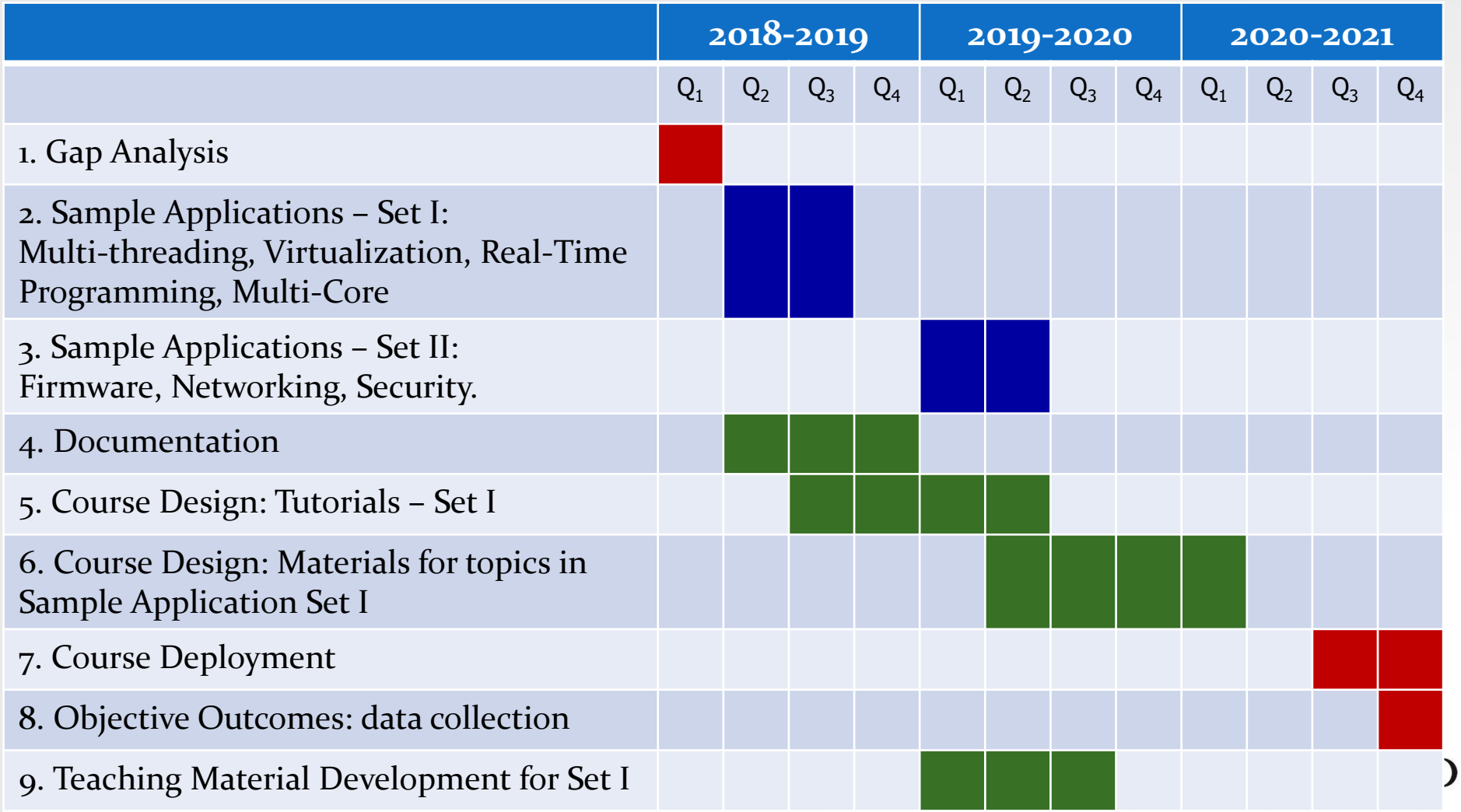
Tutorials: Most of the material developed still needs to go through more rigorous formatting and finishing touches.

- At this point, we have three complete step-by-step tutorials. These tutorials include all the software files:
  - *Introduction to Hardware and Software Tools and C Programming*: Basics of the Development Kit, Yocto Linux, C code basics (arrays, loops, pointers, functions), and compilation.
  - *Multi-threading for Matrix Multiplication*: Customizable multi-threaded application. Performance comparisons with a non-threaded version is included.
  - *Multi-threading for 2D Convolution*: Customizable multi-threaded application for image convolution. Performance comparisons with a non-threaded version is included.

Implementation

RECRLab
Reconfigurable Computing Research Laboratory

Timeline: This is organized into 9 tasks, indicated which ones will be completed for each quarter. The project started Fall 2018



Conclusions

- A new embedded curriculum has been presented tailored to high-performance embedded microprocessor
- 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 (tutorials, course design) and work is currently underway.
- The proposed curriculum is meant to complement the OU CE program by offering training and research opportunities to undergraduate students in the latest embedded technology.