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

DANIEL LLAMOCCA

Electrical and Computer Engineering Department,

Oakland University

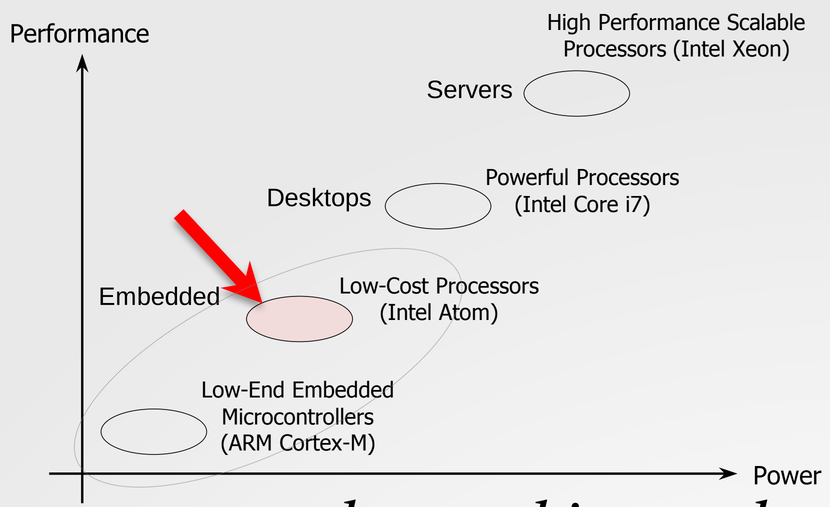
March 19th, 2022.

Outline

- ***Motivation***
- ***Hardware/Software Equipment***
- ***Course Structure***
- ***Assessment***
- ***Outcomes***
- ***Conclusions***

Motivation

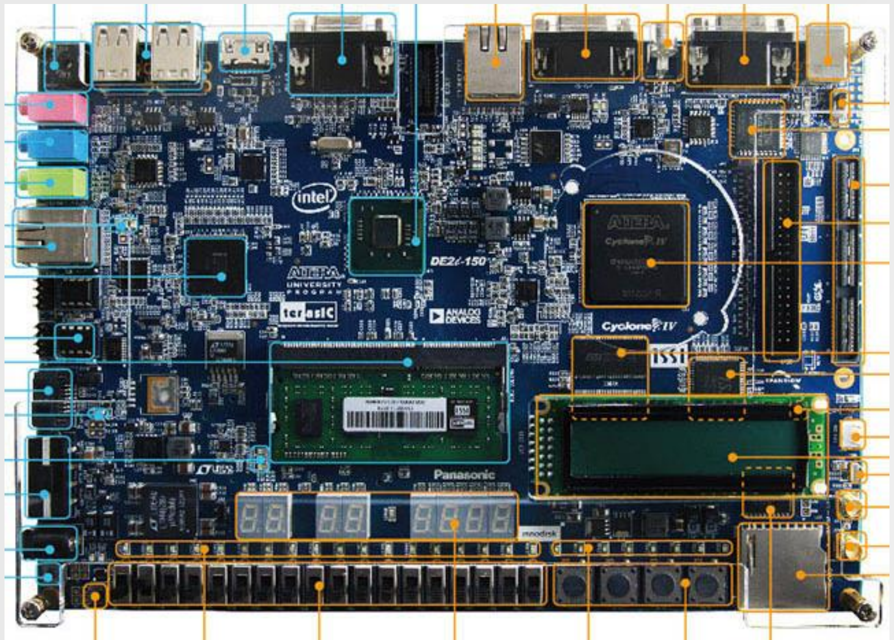
- ***Embedded Design** is covered in a typical undergraduate curriculum in Computer Engineering using standard low-power microcontrollers.*

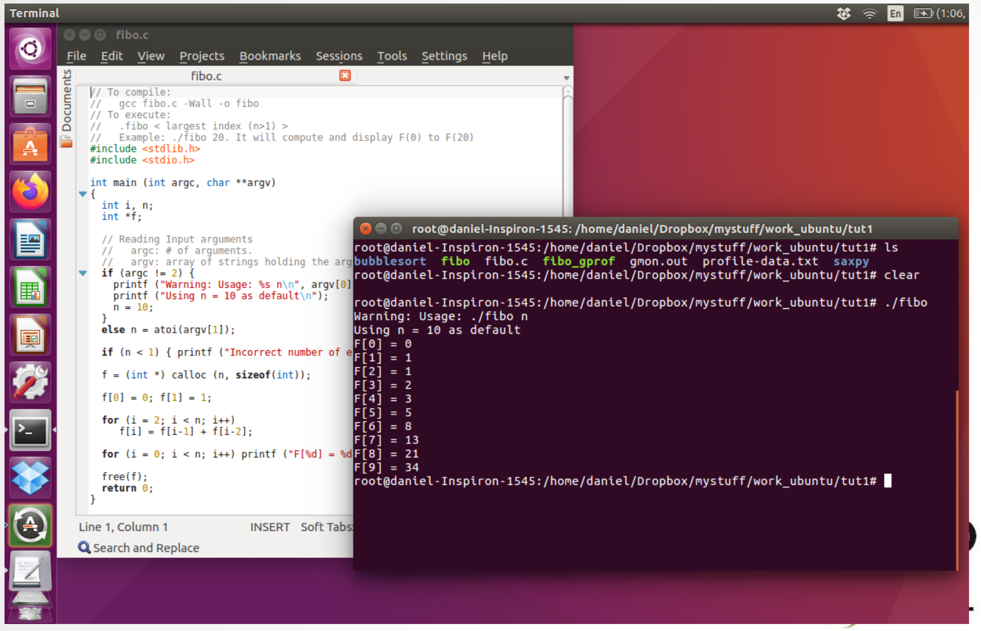

Performance
High Performance Scalable Processors (Intel Xeon)
Servers
Desktops
Powerful Processors (Intel Core i7)
Embedded
Low-Cost Processors (Intel Atom)
Low-End Embedded Microcontrollers (ARM Cortex-M)
Power

- ***Powerful embedded microprocessors** can be used instead (e.g.: Intel Atom, ARM Cortex-A9). They allow for:*
  - *More opportunities for learning skills (e.g.: parallel programming models) in high demand in industry today*
  - *Focus on high-end emerging industry-relevant applications (e.g: beamforming, multi-sensor fusion) and workloads.*
- *We present a research and educational infrastructure for **high-performance embedded programming** tailored to the needs of both our graduating students & industry.*

Hardware/Software Equipment

- ***Hardware: DE2i-150 FPGA Development Kit (provided)***
  - *Full featured computer: it includes an Intel Atom N2600 + 2 GB RAM, and an Altera FPGA (not used).*
- ***Software:** Ubuntu 12.04.4 version installed on the boards.*
  - *Applications are written in C/C++.*
  - *Application development: simple text editor or IDE (e.g.: Clion).*





```
// To compile:
//   gcc fibo.c -Wall -o fibo
// To execute:
//   .fibo < largest index (n>1) >
//   Example: ./fibo 20. It will compute and display F(0) to F(20)
#include <stdlib.h>
#include <stdio.h>

int main (int argc, char **argv)
{
  int i, n;
  int *f;

  // Reading Input arguments
  //   argc: # of arguments.
  //   argv: array of strings holding the arg
  if (argc != 2) {
    printf ("Warning: Usage: %s n\n", argv[0]
    printf ("Using n = 10 as default\n");
    n = 10;
  }
  else n = atoi(argv[1]);

  if (n < 1) { printf ("Incorrect number of e

  f = (int *) calloc (n, sizeof(int));

  f[0] = 0; f[1] = 1;

  for (i = 2; i < n; i++)
     f[i] = f[i-1] + f[i-2];

  for (i = 0; i < n; i++) printf ("F[%d] = %d

  free(f);
  return 0;
}
```

```
root@daniel-Inspiron-1545: /home/daniel/Dropbox/mystuff/work_ubuntu/tut1
root@daniel-Inspiron-1545:/home/daniel/Dropbox/mystuff/work_ubuntu/tut1# ls
bubblesort  fibo  fibo.c  fibo_gprof  gmon.out  profile-data.txt  saxpy
root@daniel-Inspiron-1545:/home/daniel/Dropbox/mystuff/work_ubuntu/tut1# clear

root@daniel-Inspiron-1545:/home/daniel/Dropbox/mystuff/work_ubuntu/tut1# ./fibo
Warning: Usage: ./fibo n
Using n = 10 as default
F[0] = 0
F[1] = 1
F[2] = 1
F[3] = 2
F[4] = 3
F[5] = 5
F[6] = 8
F[7] = 13
F[8] = 21
F[9] = 34
root@daniel-Inspiron-1545:/home/daniel/Dropbox/mystuff/work_ubuntu/tut1#
```

Course Structure

- ***Topics***: These were carefully selected and grouped into seven units. Lecture notes were developed for each unit:
  - *Embedded Multi-Core Systems: Overview of microprocessors for embedded applications; details of the Intel® Atom™ processor.*
  - *C/C++ Language Programming Fundamentals: C/C++ constructs required to work with the parallel programming frameworks.*
  - *Multi-threaded applications: Standard programming interface for multi-threads: direct thread specification (pthreads)*
  - *Multi-core applications: We cover the Intel TBB, a template for parallel programming on multi-core processors. Computations are broken down into tasks rather than threads.*
  - *Real-Time Programming: Here, we cover some introductory topics on real-time programing.*
  - *Design & Optimization of Embedded Real-Time Systems: system profiling and application profiling for single-core and multi-core.*
  - *Applications: CNNs, Beamforming, Cylinder Pressure Est.*

Course Structure

- ***Assignments***: The evaluations were organized as follows:
  - *Four (4) homeworks*
  - *Eight (8) laboratories: They were focused on the development of embedded applications using the parallel frameworks.*
  - *Take-home midterm exam (application implementation)*
  - *Final Project (groups of 2): students implemented a scalable application and perform extensive performance comparisons based on app. parameters and problem sizes. This includes a final report.*

Lab	Description
1	Board Setup, Basic Utilities. Basic C applications (num., 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): varying # 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

Course Structure

- *The table list the topics along with the associated assignments*

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

- ***Laboratory Equipment**: Intel® donated 26 DE2i-150 Development Kit. The plan is to fit a laboratory room with the equipment, but students prefer to borrow the boards.*
- ***Textbook**: Not required. We have enough material:*
  - *Lecture Notes, Tutorials, Source Code.*

Course Structure

- ***Accompanying Tutorial:*** *A total of 8 step-by-step tutorials were developed. Each ones includes pdf notes and source code:*
  1. *Getting Started with the Hardware and Software Platform: DE2i-250 Dev. Kit and Ubuntu Linux: installation, setup, examples.*
  2. *C/C++: 2D Convolution in C, neuron implementation in C+.*
  3. *Pthreads: Basic examples, 2D convolution, dot product (mutex)*
  4. *Pthreads: Matrix multiplication*
  5. *TBB – parallel_for. Basic examples: element-wise vector ops, 3-element moving average, grayscale morphological ops.*
  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. *TBB – parallel_pipeline. Modulus of two vectors, sum of squared values in a vector, processing incoming vectors,*
  8. *Real-Time Programming. Handling signals (setup, catch), configure and test the Real-Time Clock.*

Course Structure

- ***Online material:*** *The material is freely available at the Reconfigurable Computing Research Laboratory website at Oakland University.*
- *Class: (click to show some samples) *Update: ECE4772*
  - *Lecture notes for the seven units*
  - *Four homeworks with solutions*
  - *Midterm Exam*
  - *Final project guidelines along with students' final reports and presentations.*
  - *Eight laboratory experiments*
- *Tutorial: (click to show some samples)*
  - *Eight step-by-step tutorials: pdf notes, source code.*
- We are exploring other venues to deploy the material: seminar series and summer workshops.

Assessment

- *Learning outcomes include competence in topics, proficiency in multi-threading/multi-core programming, and oral/written presentation.*

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 were identified that improve student engagement, learning outcomes, and student success:*
  - *Multi-threaded/multi-core: The class builds upon students' prior knowledge on uP and lets them explore parallelization strategies.*
  - *Opportunities for research in state-of-the-art topics: task-based specification via TBB.*

Outcomes

- ***Student Projects:*** *Students 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 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.*

Outcomes

- ***Student Survey:*** *At the end of the semester, students completed an anonymous survey. This 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 avg. rating for each question was identical:*
  - *Fall 2020: 5 students, 3 responses. Average Rating (Q1 –Q5): 4.7/5.0*
  - *Fall 2021. 5 students, 4 responses. Average Rating (Q1-Q5): 5.0/5.0.*
- *Students rated their experience very highly, though the number of responses is <u>small</u>. Students also provided comments:*
  - *They were excited about the contents and material, and very engaged with the lab. experiments. An undergrad said they wanted to pursue this area of research in their masters'.*

Conclusions

- We presented results and outcomes from the implementation of a course in high performance embedded programming.
- The covered material (freely available) was successful in terms of student engagement and learning outcomes. Students rated their overall experience highly.
- There is room for improvement: we plan to add more topics (virtualization, SIMD, onetbb).
- The proposed course complements the OU CE program by offering training and research opportunities to undergraduate students in the latest embedded technology.
- 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.