Introducing Parallel Computing in Undergraduate Curriculum
Cordelia M. Brown, Yung-Hsiang Lu, Samuel Midkiff
Electrical and Computer Engineering, Purdue University

Curriculum Change, not a New Course

- Students learn different aspects of parallel computing in many courses across four years.
- The changes are integrated into the existing curriculum.
- Steps:
  - Identify which courses to change
  - Determine the orders of the changes
  - Eliminate duplicates and unnecessary contents
  - Change the course requirements (ABET)
  - Implement and integrate changes

Orders of Changes
Object-Oriented Programming (Java Thread, QThread, Synchronization)
⇒ Introduction + Digital Logic (Carry Look-Ahead)
⇒ Microcontroller (Interrupts, Hardware Description Language)
  + Computer Architecture (Pipeline, Multiple Cores, Cache Coherence)
⇒ C Programming (Pthread, Pipeline, Mutual Exclusion, Amdahl’s Law)

Observations and Discussion

- Students are eager to learn parallel computing. Most students already know processors have multiple cores.
- Students can understand important concepts through examples in everyday life
  - Washer-dryer as an example of pipeline
  - Simultaneous withdrawal from ATM motivates the needs for mutual exclusion and synchronization
  - Traffic lights regulate the access of exclusive resources (the intersection of streets).

Evaluation (Advanced C Programming Spring 2013)

- Understanding of Amdahl’s Law
- Understanding of Different Parallel Computing Paradigm: client-server, pipeline, SIMD/MIMD ...
- Understanding of the Need for Mutual Exclusion
- Performance Scaling

* The numbers mean the years when students take the courses.

The time for computing only, excluding the time for reading and writing files

- Observations: Most students can apply the concepts and write simple parallel programs using pthread.
- Common performance problems: redundant statements in the inner loops.
  Unnecessary protection of private data.