

**COVER SHEET - NEW and REVISED COURSES**

Commission on Undergraduate Studies and Policies/ Commission on Graduate Studies and Policies/ University Core Curriculum Committee  
Effective August 1993

\*SEE I - VIII for Basic Course Proposal Guidelines\*  
\*SEE APPENDIX FOR NOTES, EXPLANATIONS AND ADDITIONAL GUIDELINES\*  
\*PRINT CLEARLY, TYPE or COMPLETE ELECTRONICALLY\*

APPROVED  
CGC 4-8-10  
CGSP 4-21-10

PROPOSAL DATE: 11/11/09 DEPARTMENT: ECE

COURSE DESIGNATOR AND NUMBER: ECE 5510 (CS 5510)

TITLE OF COURSE: Multiprocessor Programming

TRANSCRIPT (ADP) TITLE (MAX-30 Characters): Multiprocessor Programming

INSTRUCTOR and/or DEPARTMENTAL CONTACT: B. Ravindran CONTACT MAILCODE: 0111

CONTACT PHONE: 1-3777 CONTACT E-MAIL: binoy@vt.edu

CHECK IF GRADUATE CREDIT IS REQUESTED (15 copies required for CGSP)

CHECK ONLY ONE OF THE FOLLOWING BOXES

NEW COURSE  REVISED COURSE [Revision > 20% \_\_\_\_\_ Revision < 20% \_\_\_\_\_ ]

NEW COURSE & INCLUSION IN THE CORE [Area \_\_\_\_\_ ]  OTHER: \_\_\_\_\_  
Include Attachment, if Needed

REVISED COURSE FOR INCLUSION IN THE CORE OR CORE AREA CHANGE

\*Courses routed directly to the University Core Committee MUST be endorsed by the appropriate Department Head or Dean.  
\*The Chair of the University Core Committee shall inform the appropriate college curriculum committee of all courses under review by the core committee.

\*A Attach Statement from Dean or Departmental Representative as to whether Teaching this Course will Require or Generate the Need for Additional Departmental Resources.

\*B Attach Appropriate Letters of Support from Affected Departments and/or Colleges.

\*C Effective Semester: Fall 2010

\*D Change in Title From: \_\_\_\_\_

To: \_\_\_\_\_

\*E Change in Lecture and/or Lab Hours From: \_\_\_\_\_ To: \_\_\_\_\_

\*F Change in Credit Hours From: \_\_\_\_\_ To: \_\_\_\_\_

\*G Percentage of Revision from Current Syllabus: \_\_\_\_\_ Revision Summary: \_\_\_\_\_

\*H Course Number(s) and Title(s) to be Deleted from the Catalogue with APPROVAL of course: \_\_\_\_\_

APPROVAL SIGNATURES

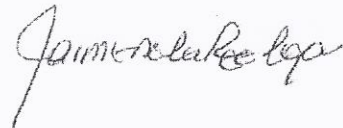
Department Representative [Signature] Date: 03/22/10

College Curriculum Committee Representative [Signature] Date: 2/23/10

College Dean [Signature] Date: 2/23/10

Feb. 8, 2010

TO: CoE Curriculum Committee  
FROM: Dr. J. De La Ree, ECE Assistant Department Head  
SUBJECT: New Course Proposal for ECE 5510



Attached is the new course proposal for ECE 5510 Multiprocessor Programming. This is a new course proposal. The department has permission from the Registrar to use this number ending with 0 (email attached).

With the approval of this new course, no additional resources will be required for the ECE Department.

If you have any questions regarding this course proposal, please contact me.

Attachment:  
Course proposal, ECE 5510 Multiprocessor Programming



VIRGINIA POLYTECHNIC INSTITUTE  
AND STATE UNIVERSITY

Department of Computer Science  
College of Engineering

Mail Code 0106, Blacksburg, Virginia 24061  
(540) 231-6262 Fax: (540) 231-4240  
ribbens@vt.edu

February 16, 2010

TO: Course Approval Committees  
FROM: Calvin Ribbens *Cal Ribbens*  
Associate Department Head  
RE: ECE/CS 5510

The Computer Science Department supports the Electrical and Computer Engineering Department's proposal for ECE 5510, which will be cross-listed as CS 5510. We understand that ECE should be considered the home department for this course, and that if ECE ceases to teach the course, a new course proposal will need to be submitted before the course can be offered again.

## **ECE 5510 (CS 5510)** **Multiprocessor Programming**

### **Catalog Description**

Principles and practice of multiprocessor programming. Illustration of multiprocessor programming principles through the classical mutual exclusion problem, correctness properties of concurrency (e.g., linearizability), shared memory properties (e.g., register constructions), and synchronization primitives for implementing concurrent data structures (e.g., consensus protocols). Illustration of multiprocessor programming practice through programming patterns such as spin locks, monitor locks, the work-stealing paradigm, and barriers. Discussion of concurrent data structures (e.g., concurrent linked lists, queues, stacks, hash maps, skiplists) through synchronization patterns ranging from coarse-grained locking to fine-grained locking to lock-free structures, atomic synchronization primitives, elimination, and transactional memory. Pre: Graduate standing and 4534 or 4550 (3H, 3C).

Course Number: 5510 (CS 5510)

ADPTitle: Multiprocessor Programming

### **Learning Objectives**

Having successfully completed this course, the student will be able to:

- Explain fundamental principles of multiprocessor programming including:
  - concurrency correctness properties such as consistency, linearizability, progress, fairness, and deadlock-freedom;
  - properties of shared memory such as register constructions and atomic snapshots; and
  - synchronization primitives for implementing concurrent data structures, ranging from simple ones (e.g., atomic registers, consensus protocols, FIFO queues) to powerful universal constructions (e.g., universality of consensus).
- Write multiprocessor programs using:
  - programming patterns including spin locks and contention, monitor locks and waiting, work-stealing and parallelism, and barriers;
  - concurrent data structures including concurrent linked lists, concurrent queues, concurrent stacks, concurrent hash maps, and concurrent skiplists;
  - synchronization patterns including coarse-grained locking, fine-grained locking, optimistic locking, lazy synchronization, non-blocking synchronization, atomic synchronization primitives, elimination, parallel search; and
  - transactional memory abstractions including software transactional memory.

- Demonstrate properties of multiprocessor programs including their correctness, fairness, consistency, linearizability, progress, and deadlock-freedom.

## **Justification**

The computer industry is undergoing a paradigm shift, as chip manufacturers are increasingly investing in, and manufacturing a new generation of multi-processor chips called multicores, as it is becoming increasingly difficult to enhance clock speeds by packing more transistors in the same chip without increasing power consumption and overheating. Consequently, application software performance can no longer be improved by simply relying on increased clock speeds of single processors; software must explicitly be written to exploit the hardware parallelism of multiprocessors. Programming multi-processors is fundamentally different from programming single-processors, due to the need to understand how concurrent computations on separate processors coordinate with one another, in contrast with understanding how concurrent computations on the same processor coordinate with one another. This is a complex and intricate problem, and requires new abstractions, algorithms, and programming tools, which are precisely the focus of the course.

Currently, the Computer Engineering curriculum does not cover this topic. Thus, students are not exposed to the fundamentally and radically changing (multiprocessor) computer programming landscape, and lack the knowledge and skills needed to effectively program emerging (multiprocessor) computers. The course directly addresses this deficiency by focusing on abstractions, algorithms, and tools needed to program multiprocessor computers, and thereby exposing Computer Engineering students to the foundations and practice of multiprocessor programming and strengthening the Computer Engineering curriculum.

The course focuses on reasoning about concurrent multiprocessor programs, writing multiprocessor programs, establishing their correctness, and analyzing their performance. These activities are different and more complex than that for concurrent single-processor programs, because of the vast number of ways that the steps of concurrent threads can be interleaved on multiprocessors, increasing the difficulty of reasoning about their correctness (e.g., safety, liveness). Furthermore, the complex memory hierarchy and memory management operations that are invisible to sequential programming (as they hide behind simple programming abstractions) break down for multiprocessor programming from a performance standpoint – features of the underlying memory system must often be exploited to achieve adequate performance.

This requires applying extensive and in-depth knowledge that builds on undergraduate learning of foundational computer architecture principles and sequential programming principles. In addition, graduate students in this course will have the ability to demonstrate knowledge of advanced abstractions, algorithms, and programming tools necessary for programming multiprocessor computers. Furthermore, students, also will be required to evaluate and critique graduate level, advanced research articles on the principles and practice of multiprocessor programming.

## Prerequisites and Co-requisites

Graduate standing and 4534 or 4550.

## Texts and Special Teaching Aids

### Required Texts:

Herlihy, M., & Shavit, N. (2008). *The Art of Multiprocessor Programming*, paperback, 528 pages, Morgan Kaufman.

## Syllabus

<i>Topic</i>	<i>Percentage</i>
Introduction to shared objects and synchronization, and the mutual exclusion problem	5
Correctness properties of concurrent programs (consistency, linearizability, progress, fairness, deadlock-freedom)	10
Foundations of shared memory (register constructions, atomic snapshots)	10
Synchronization operations for concurrent data structures (atomic registers, consensus protocols, FIFO queues, universality of consensus)	15
Programming patterns: Spin locks and contention, monitor locks and waiting, work-stealing and parallelism, barriers	10
Concurrent data structures (concurrent linked lists, concurrent queues, concurrent stacks, concurrent hash maps, concurrent skiplists)	20
Synchronization patterns: coarse-grained locking, fine-grained locking, optimistic locking, lazy synchronization, non-blocking synchronization	15
Atomic synchronization primitives, elimination, parallel search	10
Transactional memory	5
	100%