



# Course Syllabus

ENEE 646: Digital Computer Design, Fall 2002

Prof. Bruce Jacob

## 1. Basic Information

### Time & Place

Lecture: TuTh 2:00–3:15 am, CHE-2108

### Professor

Bruce L. Jacob: AVW-1325, blj@eng.umd.edu  
Office hours: *Open-door policy (for the moment)*

### Teaching Assistant

Kakaraparthi Himabindu, hima@eng.umd.edu

### Class Home Page

<http://www.ece.umd.edu/courses/enee646/>

### Class Email List

enee646-0101-fall02@coursemail.umd.edu

### Class Schedule

This is a weekly schedule of my hours, including class time and office hours (if scheduled), but also including other things that make me unavailable. The schedule is subject to change; changes will be announced in class.

	Mon	Tue	Wed	Thu	Fri
9-9:30					
9:30-10					
10-10:30					
10:30-11					
11-11:30					
11:30-12	Usually not in office: out to lunch				
12-12:30					
12:30-1					
1-1:30					Meetings with graduate research assistants
1:30-2					
2-2:30		Lecture (CHE-2108)		Lecture (CHE-2108)	
2:30-3					
3-3:30					
3:30-4					
4-4:30					

## 2. Course Overview

This course covers the architecture and design of microprocessors, memory hierarchies, and system-level software. It is intended to give you a solid understanding of how digital computers are implemented today. We will cover concepts such as pipelines, caching, superscalar execution, out-of-order execution, very-long-instruction-word architectures (VLIW), precise interrupts, and low-level operating system mechanisms.

You will learn these concepts by not only reading about them but by building quite a few of them; you will partially design and “build” simple computers at various levels of sophistication. You will build a simple machine, then a pipelined machine, and then you will have the opportunity to improve on a design reflecting current thinking in high-performance microprocessors. The term “building” in this course will mean implementing your design using the Verilog *hardware description language (HDL)*.

Building in Verilog (or in any HDL, for that matter) is interesting for several reasons. First, you must be infinitely more precise in your design than if you built the model in a high-level language such as C. This is good because it forces you to understand all the finer points of your design and the ramifications of your choices—if you are not thorough, it will not work. Second, the performance/power/die-area results that you obtain are infinitely more believable than had you built the model in a high-level language such as C. It is good to learn this because, historically, lots of money has been spent on ideas that looked good on paper but whose implementations fell far short of projected measurements. Lastly, an HDL implementation is just steps away from actual silicon. If your design works, you could get it fabricated.

## 3. Prerequisites

Students must have knowledge of digital logic design and computer organization. You should understand digital design concepts such as multiplexers, gates, boolean algebra, finite-state machines, and flip-flops. You should understand fundamental computer organization: what the program counter is, what a register file is for, how busses are used, what happens in the hardware to effect instruction execution, etc. It would help if you also understand and are reasonably fluent in programming C or Perl, because Verilog is very C-like (as is Perl).

## 4. Course Material

The required text for the course:

*Computer Architecture: A Quantitative Approach, 3rd Ed.*, by Hennessy and Patterson.

This is a widely used textbook that describes in reasonable detail most of today’s thinking in high-performance architecture. It is a good, solid book and is valuable as a reference for later on.

Because it is often worthwhile to go to the horse’s mouth for information, when discussing some high-performance designs, we will go to the original descriptions of those designs, written by the designers. For example, I will hand out copies of papers describing Tomasulo’s hardware algorithm for out-of-order execution, Smith and Pleszkun’s reorder buffer for precise interrupts, and Sohi & Vajapeyam’s register update unit for providing both out-of-order execution and precise interrupts.

For learning Verilog, there is a brief handout on the course website, and the bookstore has ordered the following textbook (which is optional, not required):

*A Verilog HDL Primer, (either edition)*, by J. Bhasker.

## 5. Class Projects

Four projects will be assigned during the term, each of which will require a substantial time commitment on your part. You will find the work load in this course to be extremely heavy.

- Project 1: C-language implementation of a simple assembler, and Verilog implementation of a simple CPU
- Project 2: Verilog implementation of a pipelined CPU
- Project 3: Addition to Project 2 of precise-interrupt support, and Assembly-code implementation of an operating system's TLB-miss handler
- Project 4: Synthesize pipelined CPU

The most common reason for not doing well on projects is not starting them early enough. You will be given plenty of time to complete each project. However, if you wait until the last minute to start, you **will not** be able to finish. Plan to do some work on a project every day. Also plan to have it finished at least 1 week ahead of the due date—many unexpected problems arise during programming, especially in the debugging phase. The computing sites can become quite crowded as deadlines approach, making it difficult to get a computer. **Plan** for these things to happen. Your lack of starting early is **not** an excuse for turning in your project late, even if some unfortunate situations arise such as having your computer crash.

There are many sources of help on which you can draw. Simple questions can be submitted to the professor and fellow classmates via email (**use the email list given on page 1**). These will typically be answered within the day, often more quickly during working hours. Keep in mind, however, that many types of questions cannot be answered without seeing your project. If you have detailed questions, your best option is to speak to the professor in person during office hours. Bring along a listing of your project and the output from a run if available. Students are also encouraged to help one another. *One of the best ways for you to make sure that you understand a concept is to explain it to someone else.* Keep in mind, however, that you should not expect anyone else to do any part of your project for you. The project that you turn in must be your own.

Projects are due at 5:00 pm on the due date. We will allow a grace period and accept projects until 11:59 pm. Sometimes unexpected events make it difficult to get a project in on time. For this reason, each person will have a total of 3 free late days to be used for projects throughout the semester. **These late days should only be used to deal with unexpected problems such as computer crashes, illness, or submission problems.** They should not be used simply to start later on a project or because you are having difficulty completing the project. Projects received after the due date (assuming that you have no late days left) will receive a zero, even if it is just one second late. I advise you to save at least one or two late days for the last project. Weekend days are counted in exactly the same way as weekdays (e.g. if the project deadline is Friday and you turn it in Sunday, that's two days late). You will be submitting your projects electronically via email attachment.

The projects will be graded primarily for *correctness*: doing all the required tasks, simulating at the correct hardware level, and giving correct results.

## 6. Exams

You are expected to take both the midterm and final exams at the scheduled times. Unless a (documented) medical or personal emergency is involved in your missing an exam, you will receive a zero for that exam. If you anticipate conflicts with the exam time, you must come talk to the instructor about it at least **1 month** before the exam date. The exam dates are given at the

beginning of the term so that you can avoid scheduling job interviews or other commitments on exam days. Outside commitments are not considered a valid reason for missing an exam. **Exams will be closed book, closed notes.**

## 7. Grading Policy

Final grades will be based on the total of points earned on the projects, homeworks, and exams. The tentative point breakdown is as follows:

- Projects: 50%
- Midterm Exam: 25%
- Final Exam: 25%

Incompletes will generally **not** be given. According to university policy, doing poorly in a course is not a valid reason for an incomplete. If you are having problems in the course, your best bet is to come talk to the instructor as soon as you are aware of it.

## 8. Lecture Schedule

Week of	Subject	Projects
Sep. 2	Intro continued: overview of topics and the Verilog language	P1 out
Sep. 9	What is important: performance/power/cost	
Sep. 16	Processor implementation: pipelines	
Sep. 23	Processor implementation: pipelines	P1 due, P2 out
Sep. 30	Processor implementation: pipelines	
Oct. 7	Memory systems: caches	
Oct. 14	<b>Review &amp; Midterm</b> (Oct. 17, in class)	
Oct. 21	Memory systems: caches	P2 due, P3 out
Oct. 28	Memory systems: DRAM architectures	
Nov. 4	Operating systems: the use of interrupts	
Nov. 11	Operating systems: memory management & multitasking	
Nov. 18	Instruction-level parallelism: origins	P3 due, P4 out
Nov. 25	Instruction-level parallelism: the middle ages	
Dec. 2	Instruction-level parallelism: today	
Dec. 9	<b>Final Review</b>	P4 due
Exams	<b>Final Exam</b> (Thursday, December 19, 10:30am, in classroom)	

## 9. Special Needs

If you have a documented disability that requires special needs, please see me as soon as possible, and certainly no later than the third week of classes.