6.004 Computation Structures
Spring 2009

Learning Objectives

On completion of 6.004, students will be able to

1. understand the role of abstraction in the design of large digital systems, and explain the major software and hardware abstractions in contemporary computer systems.
2. analyze the performance of digital systems using measures such as latency and throughput.
3. design simple hardware systems based on a variety of digital abstractions such as ROMs and logic arrays, logic trees, state machines, pipelining, and buses.
4. synthesize digital systems from a library of representative components and test the designs under simulation.
5. understand the operation of a moderately complex digital system -- a simple RISC-based computer -- down to the gate level, and be able to synthesize, implement, and debug its components.
6. appreciate the technical skills necessary to be a capable digital systems engineer.

Measurable Outcomes

Upon completion of 6.004, students will be able to

1. Identify flaws and limitations in simple systems implemented using the *static discipline* (noise assumptions, etc)
2. Identify flaws and limitations in simple systems implemented using *clocked registers with asynchronous inputs* (metastability issues).
3. Identify flaws and limitations in simple systems implemented using *pipelined processors* (pipeline hazards).
4. Identify flaws and limitations in simple systems implemented using *semaphores for process synchronization* (deadlocks).
5. Identify flaws and limitations in simple systems implemented using *shared-memory multiprocessors* (sequential inconsistency).
6. Characterize the logic function of combinational devices using CMOS, ROM, or PLA technologies.
7. Explain synthesis issues for combinational devices using CMOS, ROM, or PLA technologies from their functional specification.
8. Explain synthesis of acyclic circuits from combinational components.
9. Calculate performance characteristics of acyclic circuits with combinational components.
10. Explain and calculate performance characteristics of single-clock sequential circuits.
11. Design, debug, and test combinational circuits of the complexity of an arithmetic logic unit.
12. Design, debug, and test a controller for a finite-state machine.
13. Pipeline a combinational circuit for improved throughput.
14. Understand issues affecting microprocessor instruction set design.
15. Complete and debug the design of a simple CPU with a given RISC-based intruction set.
16. Measure the memory access performance of a processor, and tune cache design parameters to improve performance.
17. Analyze the operation of page-based virtual memory systems.
18. Translate simple programs from C to machine language.
19. Deduce processor state from a memory snapshot during execution.