L1: 6.111 Course Overview

Course Website: http://web.mit.edu/6.111/www/s2009/

Acknowledgements:
- Rex Min


- Based on Lecture Notes by Professor Anantha Chandrakasan
Lecturer
- Prof. Akintunde (Tayo) Akinwande – akinwand@mtl.mit.edu (39-553, x8-7974)

Course Assistant:
- Carolyn Collins – collins@mtl.mit.edu (39-5537, x3-0573)

Teaching Assistants (TAs) - x3-7350, lab hours in 38-600
- Adam Lerer (alerer@MIT.EDU)
- Chris Celio (celio@MIT.EDU)

Lab Aides (LAs)
- Lance Collins (ljcol25@MIT.EDU), Sam Gross(sgross@MIT.EDU)

Technical Instructor
- Gim P. Hom (gim@mit.edu, Room 38-644, x4-3373)

Stock Clerk
- Arlin Mason (lab kits) - arlin@mit.edu (38-600, x3-4674)
- John Sweeney (5th floor) - jsweeney@mit.edu (38-501, x3-0601)
Recommended Books

- **Logic Design:**

- **Verilog:** there are plenty of good Verilog books and on-line resources. We recommend the book below for a basic introduction to Verilog:
6.111 Goals and Prerequisite

- Design and Implement Complex Digital Systems
  - Fundamentals of logic design: combinational and sequential blocks
  - System integration with multiple components (memories, discrete components, FPGAs, etc.)
  - Use a Hardware Design Language (Verilog) for digital design
  - Interfacing issues with analog components (ADC, DAC, sensors, etc.)
  - Understand different design metrics: component/gate count and implementation area, switching speed, energy dissipation and power
  - Understand different design methodologies and mapping strategies (discrete logic, FPGAs vs. custom integrated circuits)
  - Design for test
  - Demonstrate a large scale digital or mixed-signal system

- Prerequisite
  - Prior digital design experience is NOT Required
  - 6.004 is not a prerequisite!
    - Take 6.004 before 6.111 or
    - Take 6.004 after 6.111 or
    - Take both in the same term
  - Must have basic background in circuit theory
  - Some basic material might be a review for those who have taken 6.004
Overview of Labs

- **Lab 1: Basics of Digital Logic (Discrete Devices)**
  - Learn about lab equipment in the Digital Lab (38-600): oscilloscopes and logic analyzers
  - Experiment with logic gates, flip-flops, device characterization
  - Introduction to Verilog

- **Lab 2: Simple FSM (Car Alarm Controller)**
  - Design and implement simple Finite State Machines (FSM)
  - Use Verilog to program an FPGA
  - Report and its revision will be evaluated for CI-M

- **Lab 3: Simple FSM (Memory Tester)**
  - Learn how to use an SRAM and testing techniques

- **Lab 4: Complex FSM (Pong Game)**
  - Design a system with multiple FSMs (Major/Minor FSM)
  - Video interface
Final Project

- Done in groups of two or three
- Open ended
- You and the staff negotiate a project proposal
  - Must emphasize digital concepts, but inclusion of analog interfaces (e.g., data converters, sensors or motors) common and often desirable
  - Proposal Conference
  - Design Review(s)
- Design presentation in class (% of the final grade for the in-class presentation)
- Top projects will be considered for design awards
- Staff will provide help with project definition and scope, design, debugging, and testing
- It is extremely difficult for a student to receive an A without completing the final project.
Grading and Collaboration

- **Grading Policy**
  - Approximate breakdown:
    - Quiz: 10%
    - 3 Problem Sets: 3%
    - 4 Lab exercises:
      - Lab 1: 9%
      - Lab 2: 10%
      - Lab 3: 8%
      - Lab 4: 11%
    - Writing (Lab 2 revision- part of CIM requirement): 10%
    - Participation (lecture, recitation, project presentations): 3%
    - Final Project: 36%

- We impose late penalties
  - Labs are penalized 20% per day (All labs must be completed to pass 6.111)
  - Final Project MUST be done on time

- **Collaboration**
  - Discuss labs with anyone (staff, former students, other students, etc.):
    - Then do them individually
    - Do not copy anything, including computer files, from anyone else
  - Collaboration (with your partners) on the project is desirable:
    - Project reports should be joint with individual authors specified for each section
    - Copy anything you want (with attribution) for your project report
The first digital systems were mechanical and used base-10 representation.

Most popular applications: arithmetic and scientific computation.
Meanwhile, in the World of Theory…

1854: George Boole shows that logic is math, not just philosophy!

Boolean algebra: the mathematics of binary values
Despite existence of relays and introduction of vacuum tube in 1906, digital electronics did not emerge for thirty years!

Claude Shannon notices similarities between Boolean algebra and electronic telephone switches

Shannon’s 1937 MIT Master’s Thesis introduces the world to binary digital electronics
Evolution of Digital Electronics

Vacuum Tubes
- ENIAC, 1946
- UNIVAC, 1951
  - 1900 adds/sec

Transistors
- First Transistor
  - Bell Labs, 1948
- IBM System/360, 1964
  - 500,000 adds/sec

VLSI Circuits
- Intel Itanium, 2003
  - 2,000,000,000 adds/sec
Goal of 6.111: Building binary digital solutions to computational problems

- **Problem Statement**
  - Labs & Design project
  - Product specs
- **Behavioral Description**
  - Algorithm selection, flowcharts, etc.
  - Algorithms, RTL, etc.
  - Flowcharts
  - State transition diagrams
- **Boolean Logic and State**
  - Conversion to binary, Boolean algebra
  - Logic equations
  - Circuit schematics
- **Hardware Implementation**
  - Device selection and wiring
  - TTL Gates (AND, OR, XOR, ...)
  - Modules (counter, shifter, ...)
  - Programmable Logic
Logic synthesis using a Hardware Description Language (HDL) automates the most tedious and error-prone aspects of design.

1. Problem Statement
   - Labs & Design project
   - Product specs
   - Algorithm selection, flowcharts, etc.

2. Behavioral Description
   - Algorithms, RTL, etc.
   - Flowcharts
   - State transition diagrams
   - Software-like programming

3. HDL Description
   - Verilog code
   - VHDL code
   - Automated synthesis

4. Hardware Implementation
   - Programmable Logic
   - Custom ASICs
Hardware structures can be modeled effectively in either VHDL and Verilog. Verilog is similar to C and a bit easier to learn.
Levels of Modeling in Verilog

- Behavioral or Algorithmic Level
  - Highest level in the Verilog HDL
  - Design specified in terms of algorithm (functionality) without hardware details. Similar to “c” type specification
  - Most common level of description

- Dataflow Level
  - The flow of data through components is specified based on the idea of how data is processed

- Gate Level
  - Specified as wiring between logic gates
  - Not practical for large examples

- Switch Level
  - Description in terms of switching (modeling a transistor)
  - No useful in general logic design – we won’t use it

A design mix and match all levels in one design is possible. In general Register Transfer Level (RTL) is used for a combination of Behavioral and Dataflow descriptions.
Verilog HDL

- **Misconceptions**
  - The coding style or clarity does not matter as long as it works
  - Two different Verilog encodings that simulate the same way will synthesize to the same set of gates
  - Synthesis just can’t be as good as a design done by humans
    - Shades of assembly language versus a higher level language

- **What can be Synthesized**
  - Combinational Functions
    - Multiplexors, Encoders, Decoders, Comparators, Parity Generators, Adders, Subtractors, ALUs, Multipliers
    - Random logic
  - Control Logic
    - FSMs

- **What can’t be Synthesized**
  - Precise timing blocks (e.g., delay a signal by 2ns)
  - Large memory blocks (can be done, but very inefficient)

Understand what constructs are used in simulation vs. hardware mapping
An FPGA is like an electronic breadboard that is wired together by an automated synthesis tool.

Built-in components are called macros.
Infer macros: choose the FPGA macros that efficiently implement various parts of the HDL code

```
always @ (posedge clk)
begin
  count <= count + 1;
end
```

“This section of code looks like a counter. My FPGA has some of those...”

Place-and-route: with area and/or speed in mind, choose the needed macros by location and route the interconnect

“This design only uses 10% of the FPGA. Let’s use the macros in one corner to minimize the distance between blocks.”
**Digital processing systems** consist of a datapath, memory, and control. Early machines for arithmetic had insufficient memory, and often depended on users for control.

Today’s digital systems are increasingly embedded into everyday places and things.

Richer interaction with the user and environment.
Cell Phone Processor (OMAP 2420) from TI
Real-World Performance Metrics

- **Cost**: commodity products
- **Speed**: scientific computing, simulation
- **Energy**: portable applications

- Commercial digital designs seek the most appropriate trade-offs for the target application...
- ...keeping time-to-market in mind
Design can be fun. Verification/testing is hard work.

Verification by simulation (and formally through test benches) is a critical part of the design process.

The physical hardware must be tested to debug the mapping process and manufacturing defects.

Physical realizations often do not allow access to internal signals. We will introduce formal methods to observe and control internal state.

Verification and Design for Test (DFT) are important components of digital design.
The Inverter: Voltage Transfer Characteristic

Digital circuits perform operations on logical (or Boolean) variables

A logical variable is a mathematical abstraction. In a physical implementation, such a variable is represented by an electrical quantity.

V(y) = V(x)

Switching Threshold

Truth Table

<table>
<thead>
<tr>
<th>IN</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Nominal Voltage Levels

V_{OH} = f(V_{OL})
V_{OL} = f(V_{OH})
V_M = f(V_M)
Example Noise Sources in Digital Circuits

- Noise sources: coupling, cross talk, supply noise, etc.
- Digital circuits must be robust against such noise sources
The Inverter: Noise Margin

\[ \text{Truth Table} \]

<table>
<thead>
<tr>
<th>IN</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- Large noise margins protect against various noise sources

\[ \text{NML} = V_{IL} - V_{OL} \]
\[ \text{NMH} = V_{OH} - V_{IH} \]
Regenerative Property

A chain of inverters

\[ v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5 \rightarrow v_6 \rightarrow \ldots \]

Simulated response

| Voltage gain | should be > 1 between logic states |
Lab Hours, Equipment, Computers

- The normal lab hours are (please be out by the indicated time):
  - Monday through Thursday – 9:00 AM to 11:45 PM
  - Friday – 9:00 AM to 5:15 PM
  - Saturday – CLOSED
  - Sunday – 1:00PM to 11:45 PM
  - Hours for Holidays, Spring Break, etc. is posted on the course website

- Please do not move or reconfigure computers and other lab equipment (logic analyzers, scopes, power supplies, etc.). Please turn off the power switch for the labkit when you are done for the day.

- Please report any equipment malfunctions (Logic Analyzers, Computers, labkit, etc.) by tagging such equipment. Also email 6.111-st09-staff@mit.edu

- We will use the following tools installed on the lab PCs (courtesy of Intel):
  - ModelSim (powerful front-end simulator for Verilog), Xilinx ISE (software for Xilinx FPGAs), Office (Microsoft word, power point, etc.)

- You can use WinSCP to transfer files between the lab PCs and athena

- Use a USB flash drive (provided with your kit) to save your work periodically

The 6.111 Lab

- Labkit based on a state-of-the-art Xilinx FPGA (6 Million gates)
  - Built-in audio/video interfaces, flash memory, high-speed SRAM
  - Advanced projects in audio/video, wireless, graphics, etc.

- State-of-the-art testing equipment (logic analyzers, scopes, computers)