



6.111 Introductory Digital Systems Laboratory



■ Lecturers

- Prof. Anantha P. Chandrakasan – anantha@mit.edu 38-107 8-7619
- Prof. Donald E. Troxel – troxel@mit.edu 36-287 3-2570

■ Teaching Assistants (TAs)

- Judy L. Chen – jlchen@mit.edu 3-7350
- James B. Oey – jamesoey@mit.edu 3-7350
- Reid Williams – reidw@mit.edu 3-7350

■ Lab Aides (LAs)

- Jiafu Cen – jiafu@mit.edu
- Colin Weltin-Wu – cwu@mit.edu

■ Technician

- Marty O. Hughes – marty@mit.edu 38-500 3-4628

■ Sixth Floor Stock Clerk

- Pat Orr – patorr@mit.edu 38-600 3-4674



6.111 Goal



■ Design and Implement Complex Digital Systems

- Use a Hardware Design Language (VHDL).
- Implement with Multiple Integrated Circuits.
- Prior Digital Design Experience is NOT Required.
- Prerequisite is Something to do with Circuit Theory.
- 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.



Objectives



On completion of 6.111 students will have confidence in their abilities to conceive and carry out a complex digital systems design project in a team of two or three people.

More broadly, they will be ready to handle substantial, challenging design problems.

In particular, students will be able to:

1. explain the elements of digital system abstractions such as digital logic, Boolean algebra, flip-flops, and finite-state machines (FSMs).
2. design simple digital systems based on these digital abstractions, and the “digital paradigm” including discrete, sampled information.
3. use basic digital tools and devices such as logic analyzers, digital oscilloscopes, PALs (PLDs), PROMs, FPGAs, and VHDL.
4. work in a design team that can propose, design, successfully implement, and report on a digital systems design project.
5. communicate the purpose and results of a design project in written and oral presentations.



Approach



■ Knowledge

- Theory
- Examples
- Design Rules
- Guidelines

■ Environment

- Lab Space
- Oscilloscopes and Logic Analyzers
- Programming equipment, computers, and design software

■ Challenges

- Quizzes
- Problem Sets
- Lab Exercises
- Project



Labs



- **Lab 1**
 - Find the Digital Lab (38-600) and wire something.
 - Learn about equipment: oscilloscopes and logic analyzers
 - Program and test a PAL (PLD).

- **Lab 2**
 - Design and implement a complicated FSM.
 - Use VHDL to program an FPGA.
 - Learn how to use an SRAM.

- **Lab 3**
 - Design a complicated system with multiple FSMs.
 - Implement RAMs and ROMs in an FPGA.
 - Good prototype for your part of a final project.



Final Project



- **Unstructured Assignment**

- **Unstructured Solution**

- **You and the staff negotiate a proposal**
 - Proposal Conference
 - Design Review(s)
 - Early
 - Detailed

- **Design Presentation**

- **Staff will provide**
 - Help with design, debugging, and testing
 - Encouragement
 - Praise (as success evolves)



Pep Talk – Be on Time



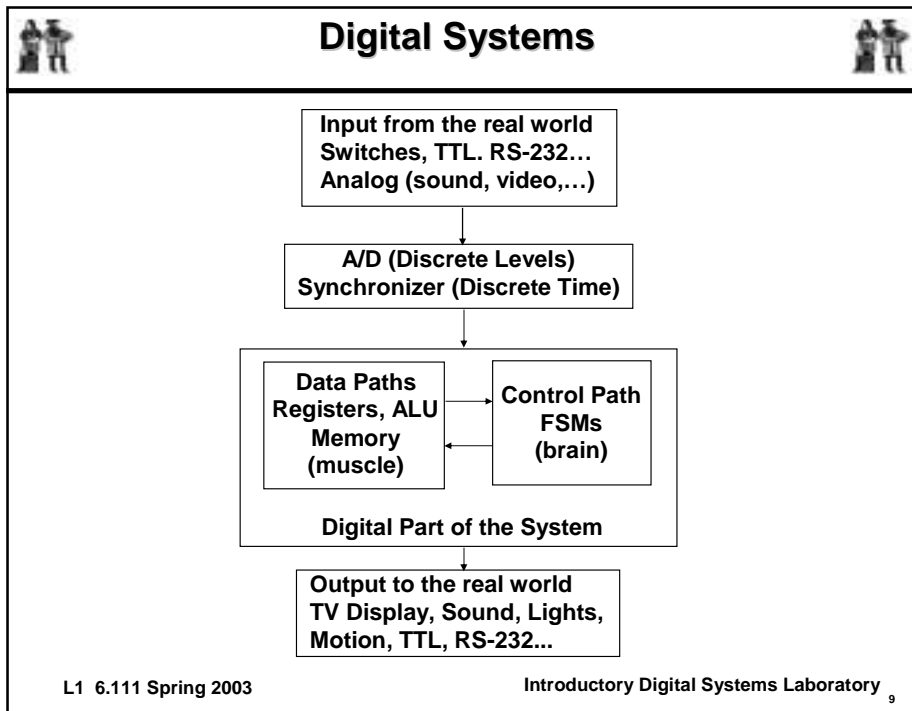
- **Start Early – Don't wait until near the deadline.**
 - You might have some questions.
 - Things often take longer than anticipated.
 - Some resource might not be immediately available.
- **Keep with it: finish early.**
- **Resources are finite.**
 - Equipment in the lab is limited.
 - TAs, LAs can be helpful, but they are pressed for time too.
 - Do not expect unlimited help near the lab deadline.
- **We impose late penalties.**
 - Problem sets must be turned in on time at lecture when they are due.
 - Problem set solutions will be passed out at the end of lecture.
 - Labs are penalized 20% per day (M-F except holidays).
 - Final Project **MUST** be done on time.



Grading and Collaboration



- **We start with a number.**
 - Then discuss everyone, especially performance in labs and project.
 - We consider lateness with labs.
 - Participation 5%
 - 2 Quizzes 20%
 - 5 Problem Sets 10%
 - 3 Lab Exercises 30%
 - 1 Project 35%
- **Cooperation**
 - Please be civil and don't hog resources such as computers.
- **Collaboration**
 - Do not collaborate with anyone when taking quizzes. Don't copy!
 - Discuss problem sets and labs with anyone, staff, former students, other students, etc.
 - Then do them individually.
 - Do not copy anything, including computer files, from anyone else.
 - Collaboration on the project is desirable (especially with your partners).
 - Copy anything you want (with attribution) for your project report.
 - Project reports may be joint with individual authors specified for each section.



- ## Implementation
- To implement digital circuits, we
 - Start with gates: AND, OR, NAND, NOR, NOT, etc.
 - These blocks are implemented with SSI (requires the most wiring).
 - Progress to building blocks: Registers, Counters, Shift Registers, Multiplexors, Selectors, etc.
 - These blocks are implemented with MSI (requires less wiring).
 - Progress to PALs, PLDs, CPLDs, FPGAs, ASICs, Custom VLSI chips.
 - These blocks require the least wiring.
 - We will not use ASICs or Custom VLSI chips.
 - These blocks (chips) are very complicated.
 - Designing them with gates is not very productive and error prone.
 - We need a higher level language such as VHDL or Verilog to specify the programming of PALs, CPLDs, and FPGAs.
 - We will use VHDL this term.
- L1 6.111 Spring 2003 Introductory Digital Systems Laboratory 10

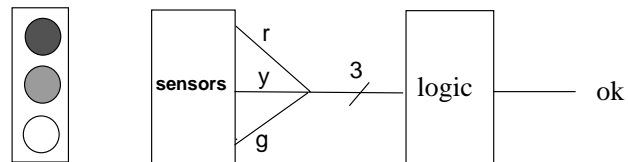


Stoplight Example



- **Logic to determine whether a stoplight is ok.**

- Most traffic lights have separate walk lights, but older (obsolete) Massachusetts stoplights use red and yellow on together to indicate that a pedestrian is supposed to have a chance to cross the street.



VHDL



- **VHSIC Hardware Description Language**

- A double acronym

- **Language to express digital systems**

- Structural
- Behavioral
- Timing

- **Rich and powerful language**

- **Basic standard environment**

- **Supports both**

- Hardware concepts and
- Software concepts



VHDL Example - Warp



```
-- Massachusetts (Obsolete) Stoplight Example
-- This file works for galaxy.
-- For maxplus2 the file name MUST be check.vhd,
-- i.e., the name of an entity contained in the file.
```

```
library ieee;
use ieee.std_logic_1164.all;
entity check is port(
  r, y, g:   in std_logic;
  ok:       out std_logic);
end check;
```

```
architecture logical of check is
  signal t1, t2, t3: std_logic;
begin
  t1 <= r and (not g);
  -- doesn't matter whether y is on or off
  t2 <= y and (not g);
  -- doesn't matter whether r is on or off
  t3 <= (not r) and (not y) and g;
  -- g on alone
  ok <= t1 or t2 or t3;
end logical;
```



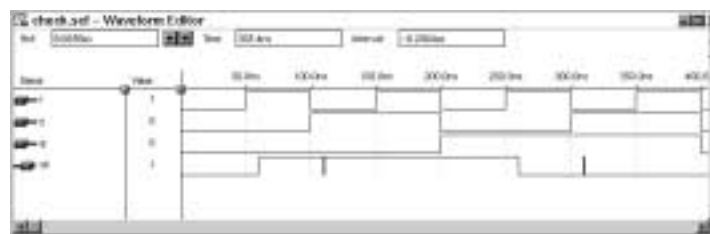
VHDL Example – MAX+Plus II



```
-- Massachusetts (Obsolete) Stoplight Example
-- This file works for galaxy.
-- For maxplus2 the file name MUST be check.vhd,
-- i.e., the name of an entity contained in the file.
```

```
library ieee;
use ieee.std_logic_1164.all;
entity check is port(
  r, y, g:   in std_logic;
  ok:       out std_logic);
end check;
```

```
architecture logical of check is
  signal t1, t2, t3: std_logic;
begin
  t1 <= r and (not g);
  -- doesn't matter whether y is on or off
  t2 <= y and (not g);
  -- doesn't matter whether r is on or off
  t3 <= (not r) and (not y) and g;
  -- g on alone
  ok <= t1 or t2 or t3;
end logical;
```





VHDL Characteristics



■ Advantages

- Shorter design cycle
- Improved design quality
- Vendor and technology independence
- Lower design cost
- Design management

■ Disadvantages

- A change of culture
 - Away from Schematic-based Design
 - Towards Language-based Design
- Cost of getting started
 - Selecting and paying for tools
- Debugging design problems



Misconceptions



- **Just code it in VHDL.**
 - The synthesis tool will design the logic.
- **Two different VHDL encodings that simulate the same way will synthesize to the same set of gates.**
- **The best style is OBJECT ORIENTED.**
 - It depends on the objects. (It always does.)
- **Synthesis just can't be as good as a design done by humans.**
 - Shades of assembly language versus a higher level language
- **Synthesis programs (fitters) are written by wizards who know what I had in mind for the design.**



What Can be Synthesized?



- **Combinational Functions**
 - Multiplexors, Selectors, Encoders, Decoders,
 - Comparators, Parity Generators,
 - Adders, Subtractors, ALUs,
 - ?Multipliers?
 - Miscellaneous logic
- **Counter based functions**
 - Counters, MAR, FIFO
- **Control Logic**
 - FSMs
 - Synchronizers



Observations



- **VHDL is a programming language.**
 - Many good and bad programs have been (will be) written.
- **Functionality is important**
 - BUT not enough!
 - Style is important.
 - Clarity is important.
- **Synthesis is hard.**
 - Fitter programs take clues from your VHDL code.
- **Decomposition of a large design into smaller, understandable sub-parts is essential.**



Design for Testing



- Design can be fun.
- Testing is hard work.
- Testing is more important than design.
- Untested designs are rarely good designs.
- Verification by simulation is often the **ONLY** way that parts of a design can be checked.
 - Physical realizations often do not allow access to internal signals.
- Almost all of us make mistakes.
 - Testing helps catch mistakes.



How To Use VHDL



- Design before typing.
 - At least a little!
- Start with a working VHDL file.
 - Modify it to do what you want to do.
- Compile files as they are typed in.
 - Don't wait until a design is completely entered before compiling and/or fitting the VHDL code.
- Use hierarchical design.
- Take one step at a time.



Lab Hours - Equipment



- Please be out by the indicated time.
 - Instrument room personnel get paid by the hour.
 - Please do not take advantage of them.
- The usual times are:
 - Monday through Friday – 9:00 AM to 11:45 PM
 - Saturday – 12:00 noon to 5:45 PM
 - Sunday – 12:00 noon to 11:45 PM
 - Hours for Holidays, Spring Break, etc. will be posted.
- Please report any equipment malfunctions (Logic Analyzers, Computers, etc.) by tagging such equipment with your name. Also send email to 6.111staff@mit.edu
- Send email to 6.111staff@mit.edu if you have questions, etc. All of us read that list and you will get an earlier response than by sending to an individual member of the teaching staff.



6.111 Software



- Use 'setup 6.111' not 'add 6.111'
 - 'setup 6.111' sources /mit/6.111/attachrc which attaches 6.111-nfs and sources /mit/6.111-nfs/attachrc which sets up your path and environment variables, etc.
- Cypress' WARP only runs on Sun workstations in the Digital Lab (38-600).
- Altera's MAX+PlusII software only runs on Sun workstations in Athena clusters. MAX+PlusII works on Athena Linux PCs in the Digital Lab (as well as on Ultra 5s).
- You may slogin to the Ultra 5s in the Digital Lab.
 - Their names are athpal0[12345] and eecs-ath-33.
 - 'setup 6.111' followed by 'ultra5' slogins to the least lightly loaded Ultra5 in the Digital Lab.
 - Don't forget to do 'setup 6.111' after slogin to an Ultra5.



Turn in your schedule sheet.



- **Fill in the form on the last page of the handout.**
 - You need fill in **ONLY** your Tuesday schedule.

- **Extra handouts are stored in the filing cabinet in the back left corner of the digital lab.**

- **Recitation assignments will be posted by Friday on the bulletin board in the Digital Lab and on the 6.111 web page.**

- **//<http://web.mit.edu/6.111/www/s2003>**
 - Also available from the EECS department web page
 - Shorthand which works (from MIT) is 'web/6.111'

- **Pick up lab kits starting Thursday at 1 pm.**
 - PS 1 due in one week and Lab 1 is due in two weeks.