Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.111 Introductory Digital Systems Laboratory

Lecture # 1

The Goal of 6.111

Transform students into engineers,
capable of designing and implementing
complex digital systems.

Use a hardware description Language (VHDL)

Implement with multiple existing integrated circuits

Prior digital design experience is not necessary

Some experience with circuits is a prerequisite
(6.002, 6.071 or equivalent)

6.004 is NOT a prerequisite

Take 6.004 first
Take 6.111 first
Take 6.004 and 6.111 in the same term

Objectives and Outcomes

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 digital oscilloscopes, PALs, PROMs, and VHDL.
4. work in a design team that can propose, design, successfully implement, and report on a digital circuit design project.
5. communicate the purpose and results of a design project in written and oral presentations.

Approach:

Knowledge:
- Theory of Digital Electronic Systems
- Examples
- Design Rules
- Guidelines (from experience)

Environment:
- Lab Space
- Equipment: logic analyzers, oscilloscopes, computers, design software

Challenges
- Quizzes
- Problem Sets
- Lab Exercises
- Project

Lab 1:

Find the lab and wire something

Learn about equipment: 'scopes, logic analysers

Program and test a PAL (A PLD)

Lab 2:

Design and implement a complicated FSM

Use VHDL to program a CPLD

Lab 3:

Design exercise using multiple FSM’s

Use VHDL to program a complex FPGA

Final Project:

- Unstructured Assignment
- Unstructured Solution
- You and the staff negotiate a proposal
  - Proposal Conference
  - Design Review(s)
    - Early
    - Detailed
- Staff will provide
  - Help with design, debugging and testing
  - Encouragement
  - Praise (as success evolves)

Necessary Details: Grading and Collaboration

Quizzes (2):	20%
Problem Sets (5)	10%
Lab Exercises (3)	35%
Final Project	35%

Collaboration Policy:

- Please be civil and don't hog resources such as computers
- Do not collaborate with anyone on quizzes
- Do not copy anything from anyone else
- You may discuss homework and labs, then do them individually:
  - turn in only your own work

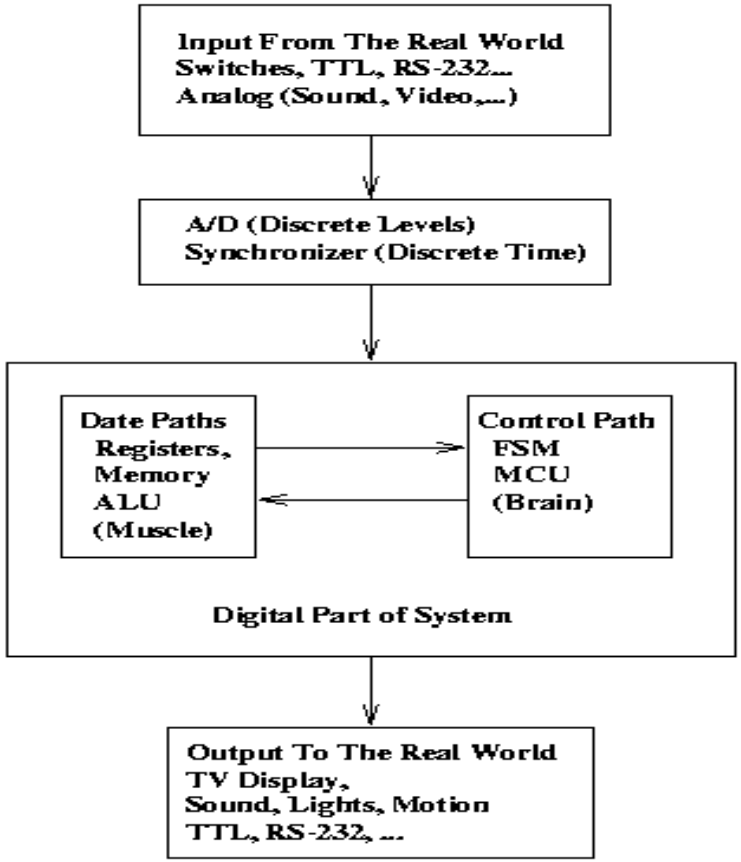
Project phase

- Collaboration is welcomed
- Get help from anyone who will help you
- Copying material is OK (with proper attribution)

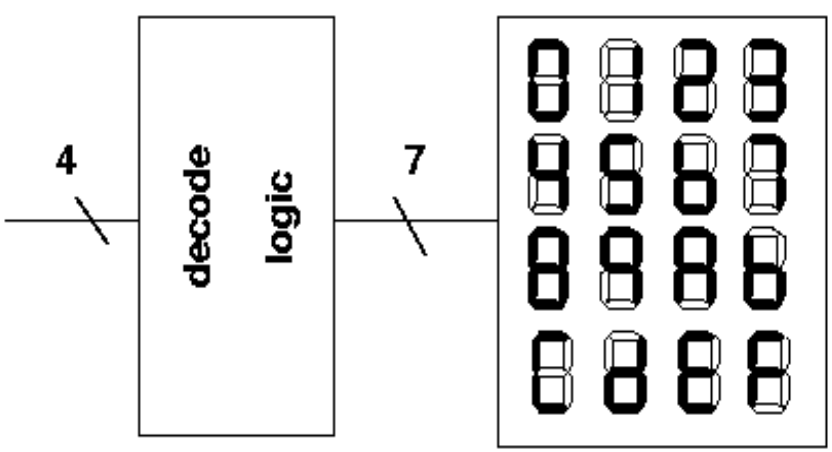
Pep Talk: Be on Time

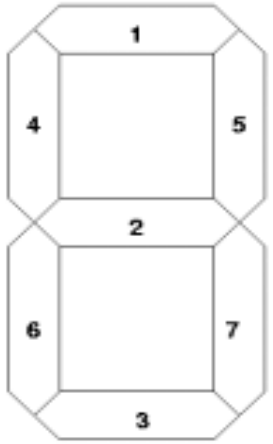
- Start Early: Don’t wait until near the deadline
- Keep with it: finish early
- Resources are finite
  - Equipment in the lab
  - TA’s can be of help, but are pressed late
  - Do not expect unlimited help late in a lab
- We impose late penalties:
  - Homework MUST be on time
  - Lab assignments: 20% per day
  - Final Project: Must be done on time

Digital Systems:



An Example: Drive a 7 Segment Display

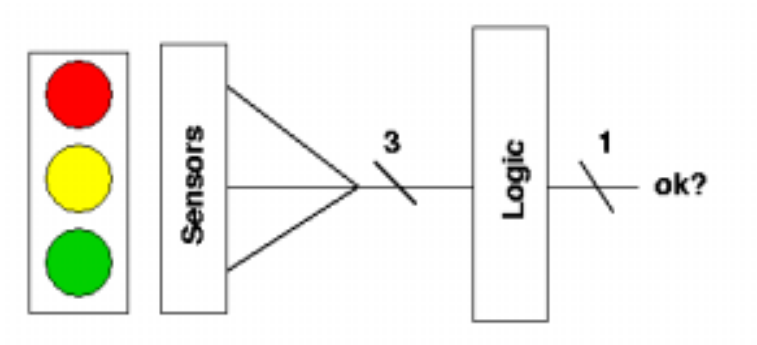




The seven-segment display can be made to display any of, say, 16 characters (0-F).

Input	Character	Segment 1
0 0 0 0	0	ON
0 0 0 1	1	OFF
0 0 1 0	2	ON
0 0 1 1	3	ON
0 1 0 0	4	OFF
0 1 0 1	5	ON
0 1 1 0	6	ON
0 1 1 1	7	ON
1 0 0 0	8	ON
1 0 0 1	9	ON
1 0 1 0	A	ON
1 0 1 1	b	OFF
1 1 0 0	c	OFF
1 1 0 1	d	OFF
1 1 1 0	E	ON
1 1 1 1	F	ON

Another Example: Logic to determine if a stoplight is working:



Now obsolete, but in Massachusetts Red+Yellow used to mean “walk” So the check must allow for that combination plus any ONE light ON

To implement 'logic', we:

Start with Gates: AND, OR, NAND, NOR,

Progress to Building Blocks which will become paradigms: Registers, Counters, Shift Registers, Multiplexors, Selectors, etc.

These things you can wire together and they are all in your kit

Then we progress to more complex programmable logic devices: PALs, CPLD's and FPGA's and we need a language to use to program them. This brings us to VHDL

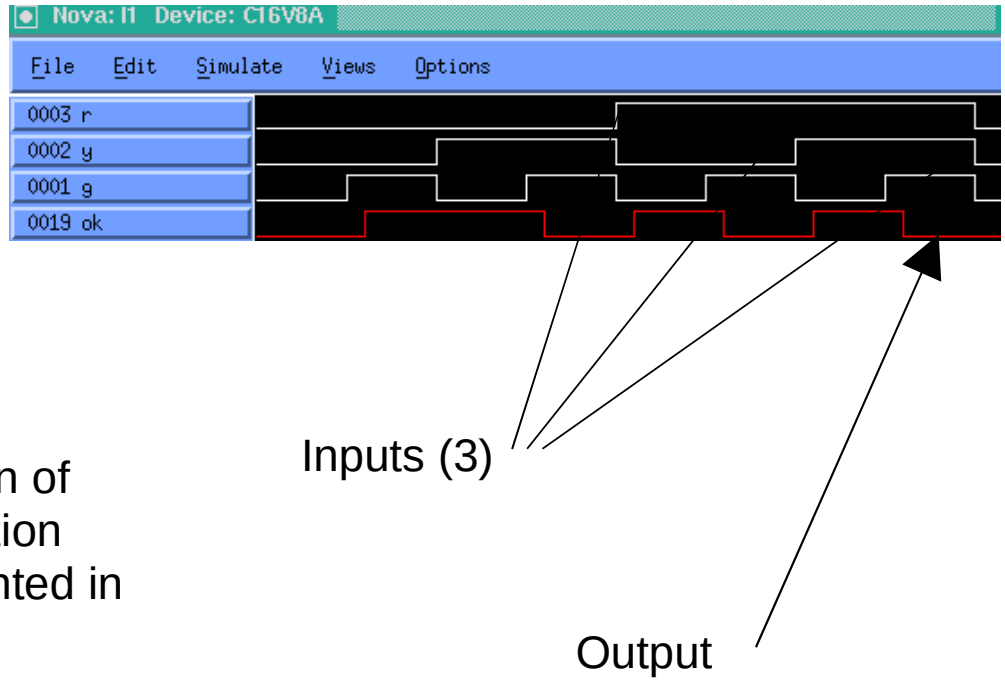
VHDL

VHSIC Hardware Description Language

- Language to express digital systems
  - Structural
  - Behavioral
  - Timing

- Rich and powerful language
- Basic standard environment
- Supports both
  - Hardware concepts
  - Software concepts

```
-- Massachusetts (Obsolete) Stoplight Example
library ieee;
use ieee.std_logic_1164.all;
entity check is port(
        r, y, g:        in std_logic;
        ok: out std_logic;
        signal t1, t2, t3: inout std_logic);
end check;
architecture logical of check is
begin
        my_label: process(r, y, g, t1, t2, t3) begin
                t1 <= r and (not g);
           t2 <= y and (not g);
           t3 <= (not r) and (not y) and g;
           ok <= t1 or t2 or t3;
           end process;
    end logical;
```



Here is a simulation of that function implemented in a PLD

To do to get started in 6.111:

Fill in the form on the last page of the handout. Only your Tuesday schedule is important Turn it in **NOW**.

Go get lab kit -- they should be ready by Friday

Get a computer account -- log onto Sunpal1 or Sunpal2 with your Athena login.

Recitation assignments will be posted in the lab, target date is Friday