L1: 6.111 Course Overview

Acknowledgements:

Materials in this lecture are courtesy of the following sources and are used with permission.

Rex Min

J. Rabaey, A. Chandrakasan, B. Nikolic. *Digital Integrated Circuits: A Design Perspective.* Prentice Hall/Pearson, 2003.

- **Logic Design:**
  - ☐ Randy Katz, Gaetano Borriello, <u>Contemporary Logic Design</u>, Pearson Education, 2005

- **Verilog: there are plenty of good Verilog books and on-line resources. We recommend the book below for a basic introduction to Verilog:**
  - ☐ Samir Palnitkar, <u>Verilog HDL</u>, Pearson Education (2nd edition)

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**

- **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 (Traffic Light 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

- **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   **7%**
      - ○ Lab 4   **11%**
    - Writing (Lab 2 revision- part of CIM requirement)   **10%**
    - Participation (lecture, recitation, project presentations)   **3%**
    - Final Project   **37%**

- **We impose late penalties**
  - □ **Labs are penalized 20% per day**
  - □ **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

Photograph of the Babbage Difference Engine.

Image removed due to copyright restrictions.

The Babbage Difference Engine (1834)

25,000 parts
cost: £17,470

- **The first digital systems were mechanical and used base-10 representation.**

- **Most popular applications: arithmetic and scientific computation**

Photograph of George Boole.

Image removed due to copyright restrictions.

AND	OR	NOT
0 0 → 0	0 0 → 0	
0 1 → 0	0 1 → 1	0 → 1
1 0 → 0	1 0 → 1	1 → 0
1 1 → 1	1 1 → 1	
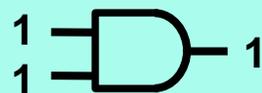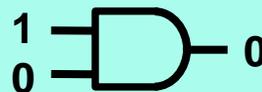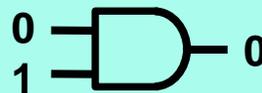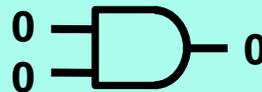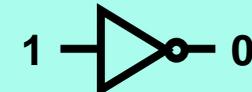
- **1854: George Boole shows that logic is math, not just philosophy!**

- **Boolean algebra: the mathematics of binary values**
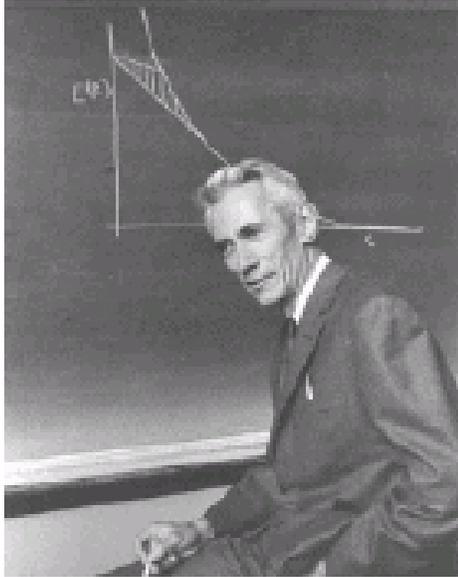
Claude Shannon

Courtesy of Jonah Sacks. Used with permission.

Lee de Forest, 1906
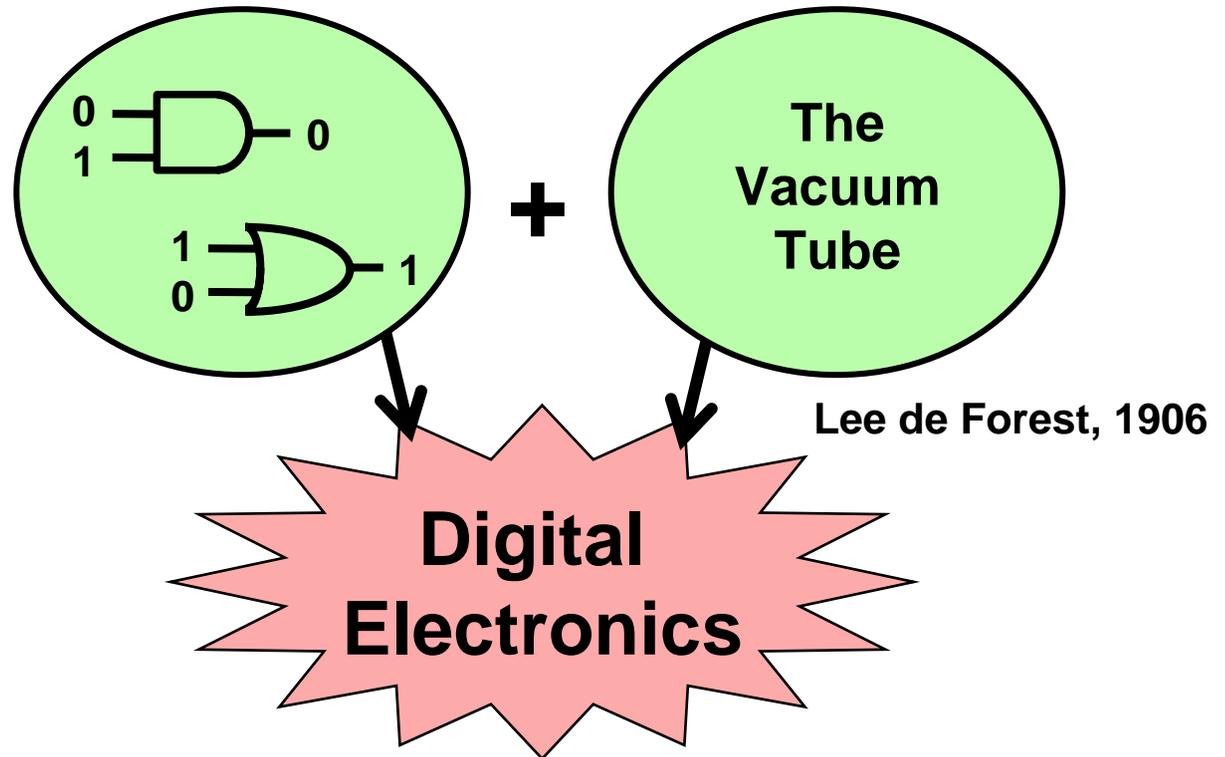
- Despite existence of **relays** and introduction of **vacuum tube** in 1906, *digital* electronics did not emerge for thirty years!

- 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

4004, 1971

Intel Itanium, 2003

2,000,000,000
adds/sec

Building Digital Systems

- **Goal of 6.111: Building binary digital solutions to computational problems**

Problem Statement
- Labs & Design project
- Product specs

algorithm selection, flowcharts, etc.

Behavioral Description
- Algorithms, RTL, etc.
- Flowcharts
- State transition diagrams

conversion to binary, Booelan algebra

Boolean Logic and State
- Logic equations
- Circuit schematics

device selection and wiring

Hardware Implementation
- 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**

Problem Statement
- Labs & Design project
- Product specs

algorithm selection, flowcharts, etc.

Behavioral Description
- Algorithms, RTL, etc.
- Flowcharts
- State transition diagrams

software-like programming

HDL Description
- Verilog code
- VHDL code

automated synthesis

Hardware Implementation
- Programmable Logic
- Custom ASICs

Verilog and VHDL
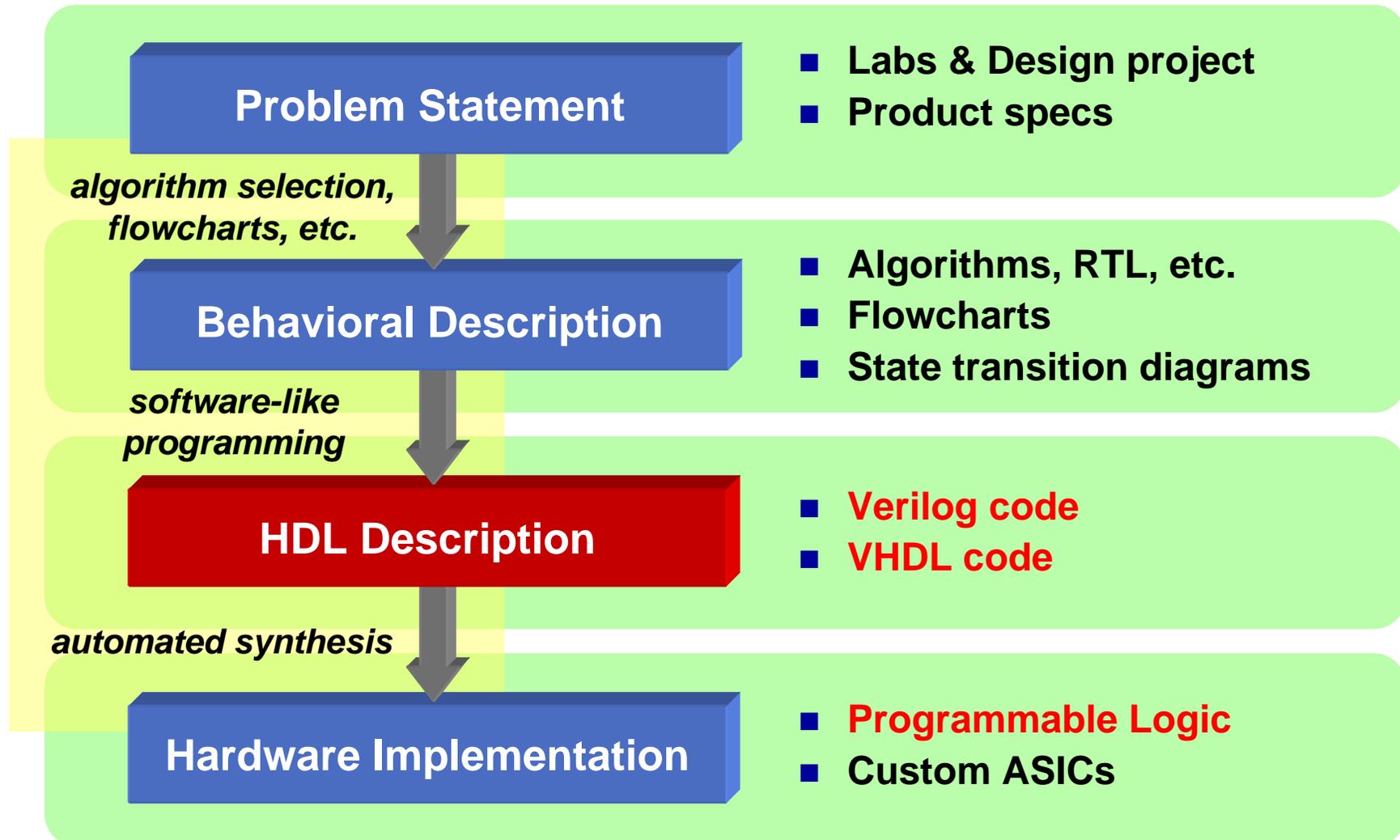
VHDL	Verilog
■ **Commissioned in 1981 by Department of Defense; now an IEEE standard**	■ **Created by Gateway Design Automation in 1985; now an IEEE standard**
■ **Initially created for ASIC synthesis**	■ **Initially an interpreted language for gate-level simulation**
■ **Strongly typed; potential for verbose code**	■ **Less explicit typing (e.g., compiler will pad arguments of different widths)**
■ **Strong support for package management and large designs**	■ **No special extensions for large designs**

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

Embedded Digital System
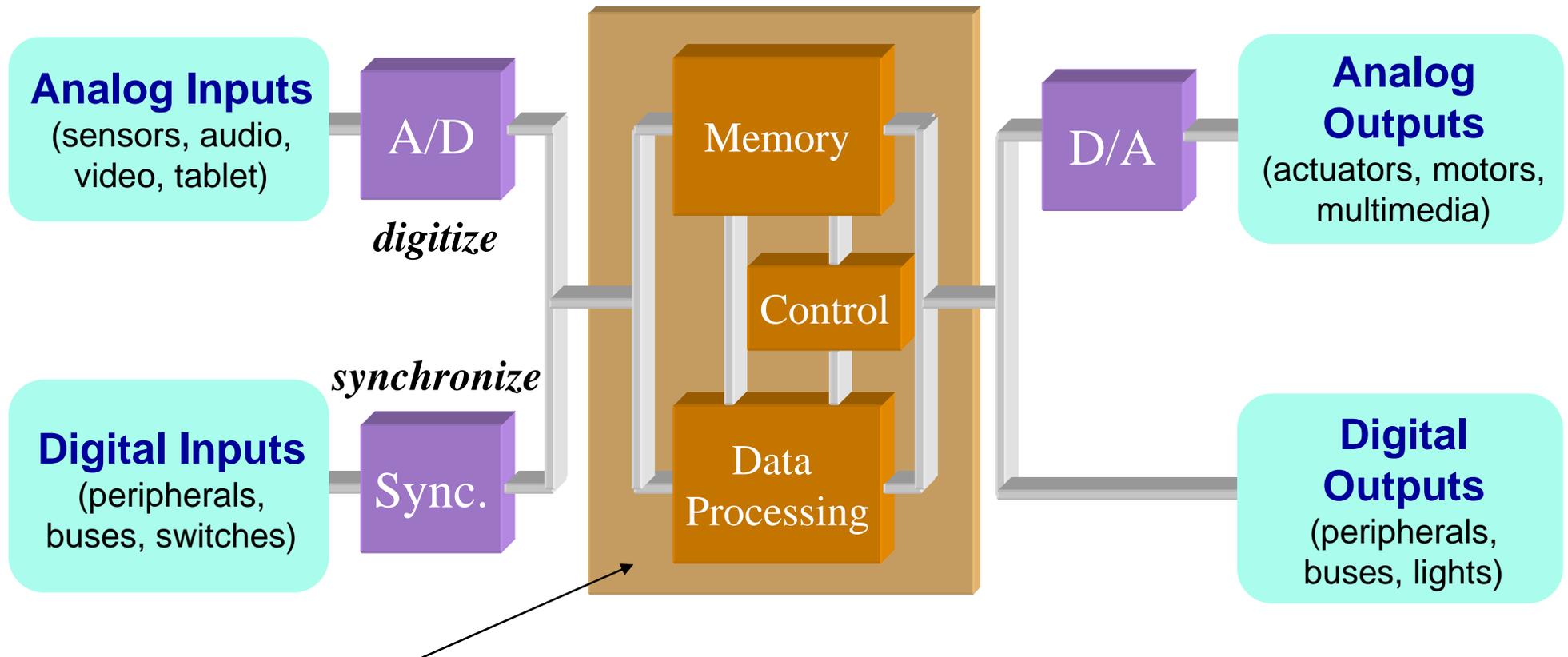
- **Digital processing systems** consist of a datapath, memory, and control. Early machines for arithmetic had insufficient memory, and often depended on users for control

- Today's digital systems are increasingly embedded into everyday places and things

- Richer interaction with the user and environment

Motors and Mechanical Sensors
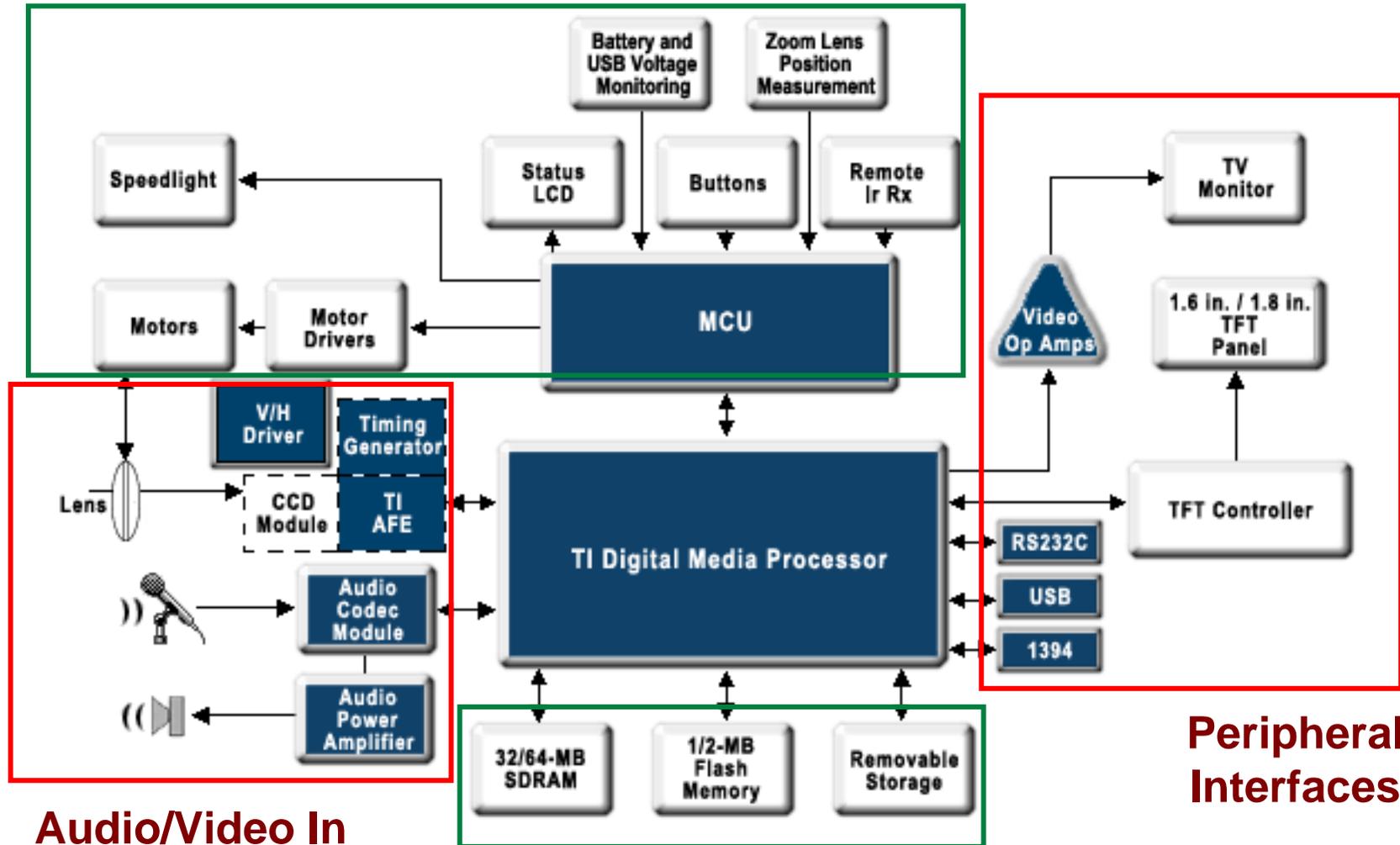
Audio/Video In

Memory Subsystem

Peripheral Interfaces

Courtesy of Texas Instruments. Used with permission.

Real-World Performance Metrics

| Cost	Speed	Energy
commodity products	scientific computing, simulation	portable applications

- **Commercial digital designs seek the most appropriate trade-offs for the target application…**

- **…keeping time-to-market in mind**

Verification and Testing

- **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
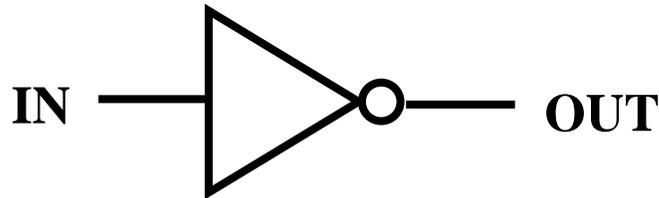
IN — ▷○— **OUT**

Truth Table

| IN	OUT
0	1
1	0

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_{OH}

f

V(y)=V(x)

V_M **(Switching Threshold)**

V_{OL}

V_{OL} ← → V_{OH} V(x)

Nominal Voltage Levels
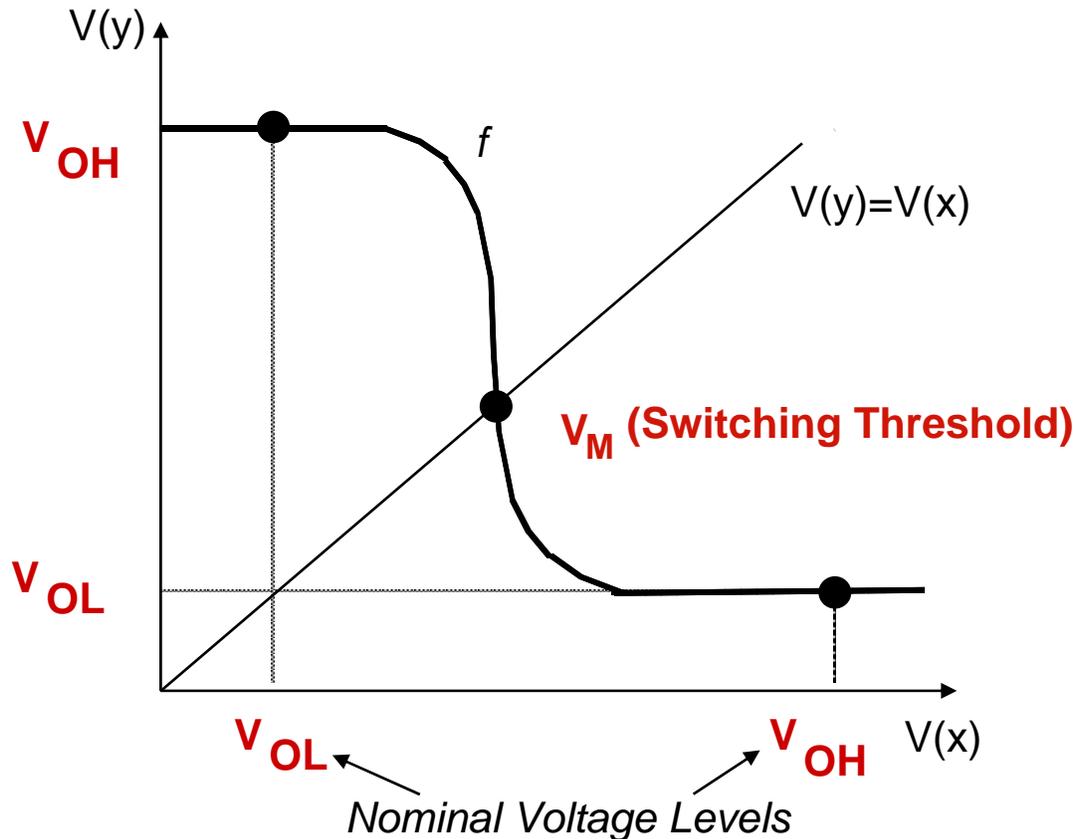
$$V_{OH} = f(V_{OL})$$
$$V_{OL} = f(V_{OH})$$
$$V_M = f(V_M)$$
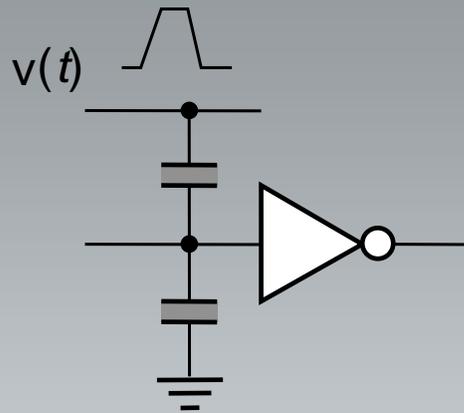
Capacitive coupling
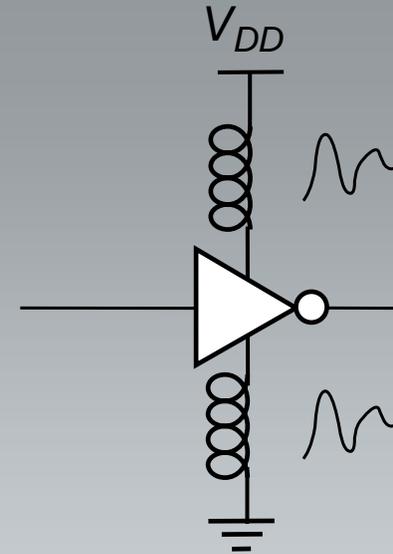
Power and ground noise

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

Truth Table

IN	OUT
0	1
1	0

IN ──▷○── OUT

"1"

V_{OH}

V_{IH}

Undefined Region

V_{IL}

"0"

V_{OL}

$V(y)$

V_{OH}

Slope = -1

Slope = -1

V_{OL}

$V(x)$

V_{OL} V_{IL} V_{IH} V_{OH}
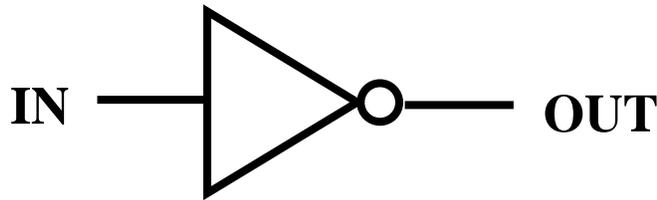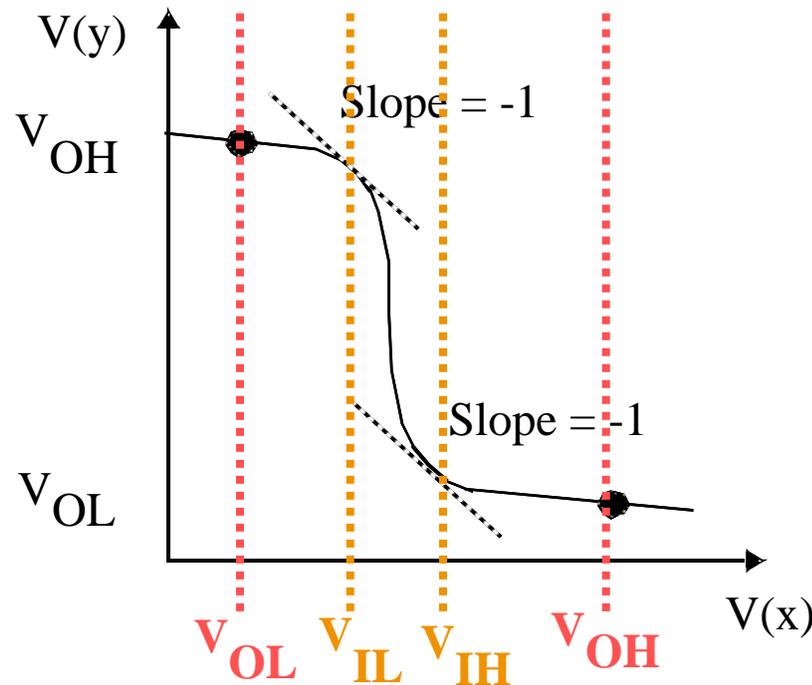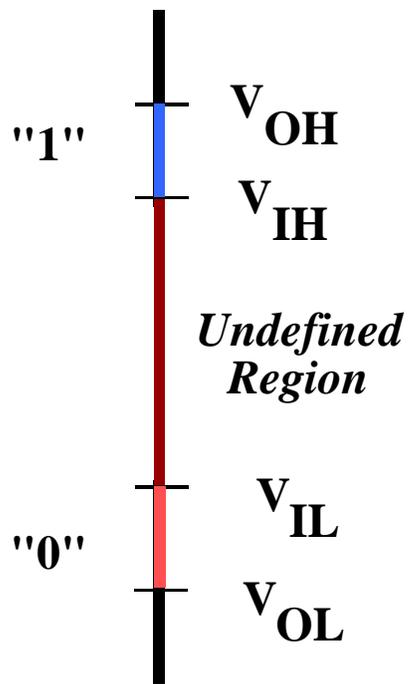
$$NM_L = V_{IL} - V_{OL}$$
$$NM_H = V_{OH} - V_{IH}$$

- **Large noise margins** protect against various noise sources

Regenerative Property

A chain of inverters

v_0 v_1 v_2 v_3 v_4 v_5 v_6

out

v_3

$f(v)$

v_1

$finv(v)$

v_2 v_0

in

Simulated response

V (Volt)

5

3

v_0

1

v_1

v_2

2 1

0 2 4 6 8 10

t (nsec)
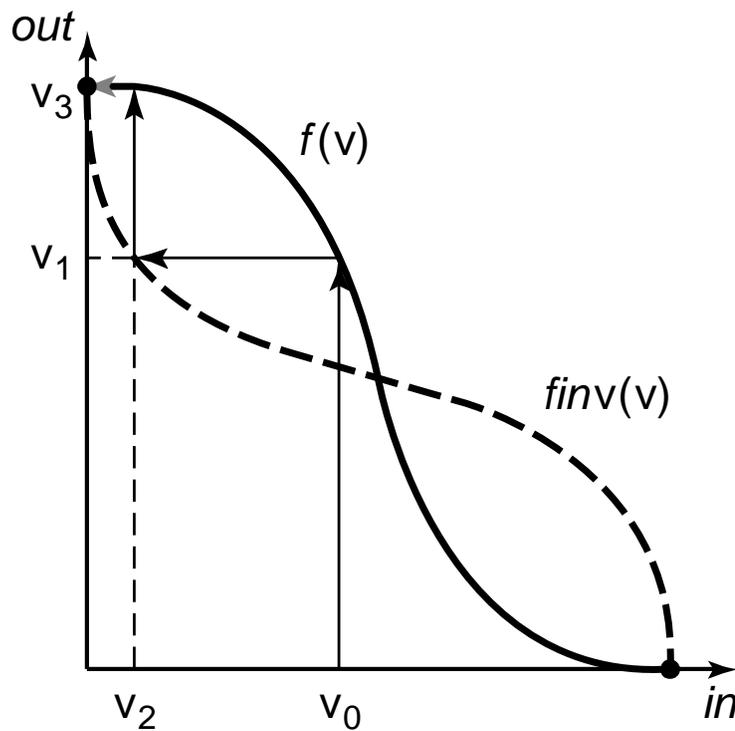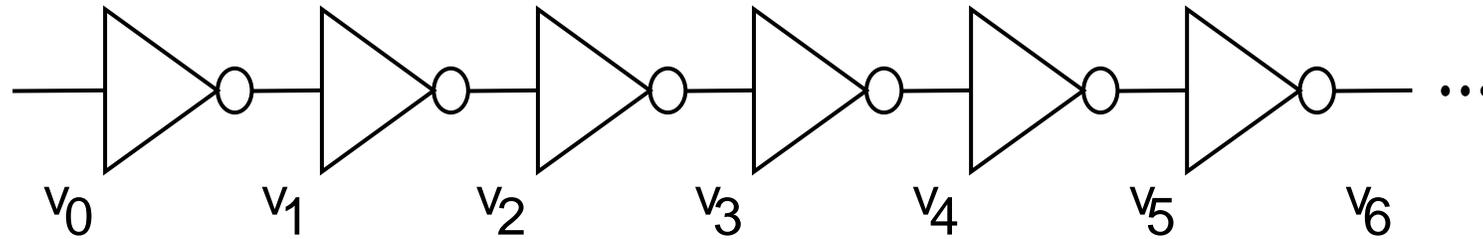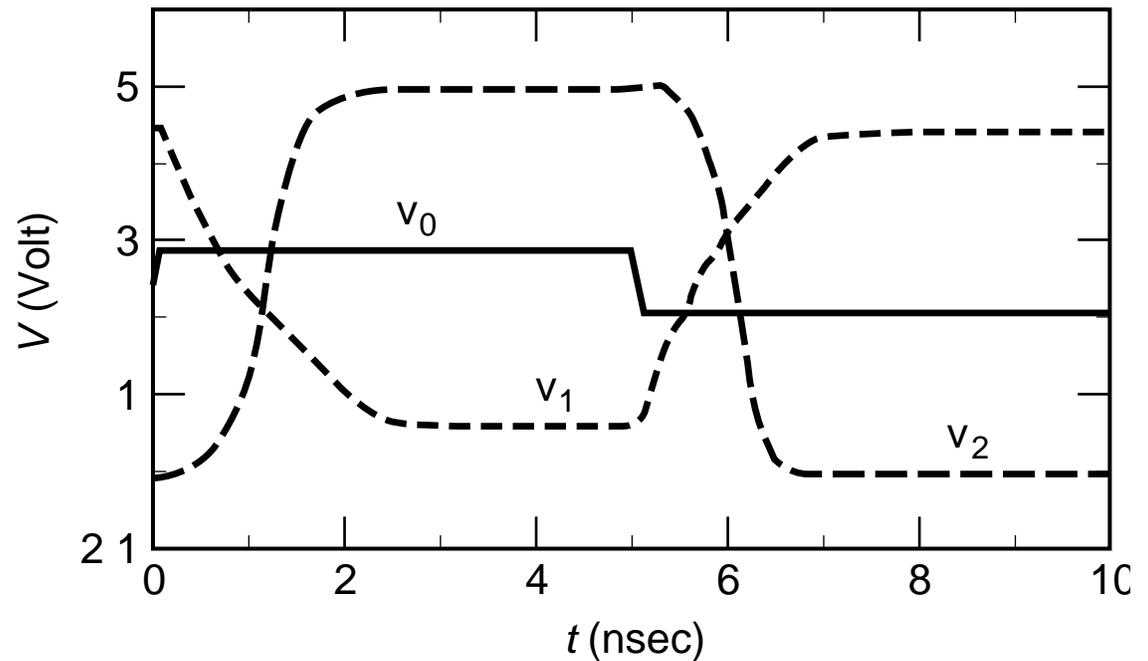
| Voltage gain | should be > 1 between logic states

- **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 – noon to 11:45 PM**

- **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.**

- **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 the MIT server.**

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

- **On MIT server use 'setup 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.**
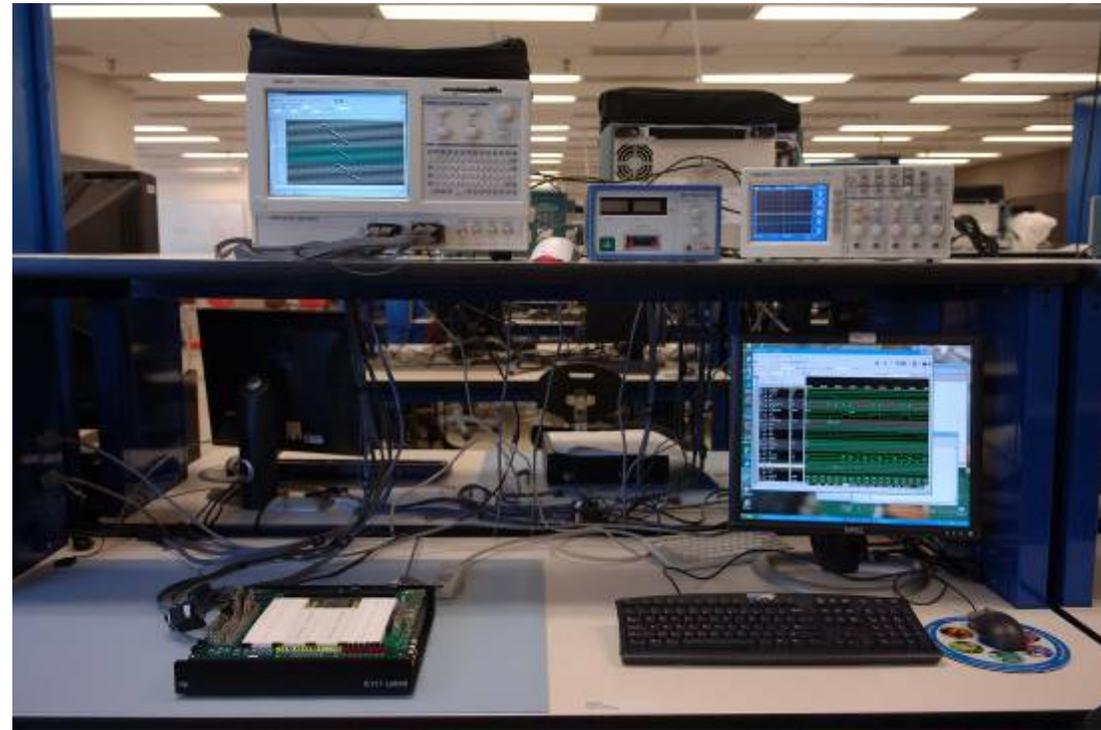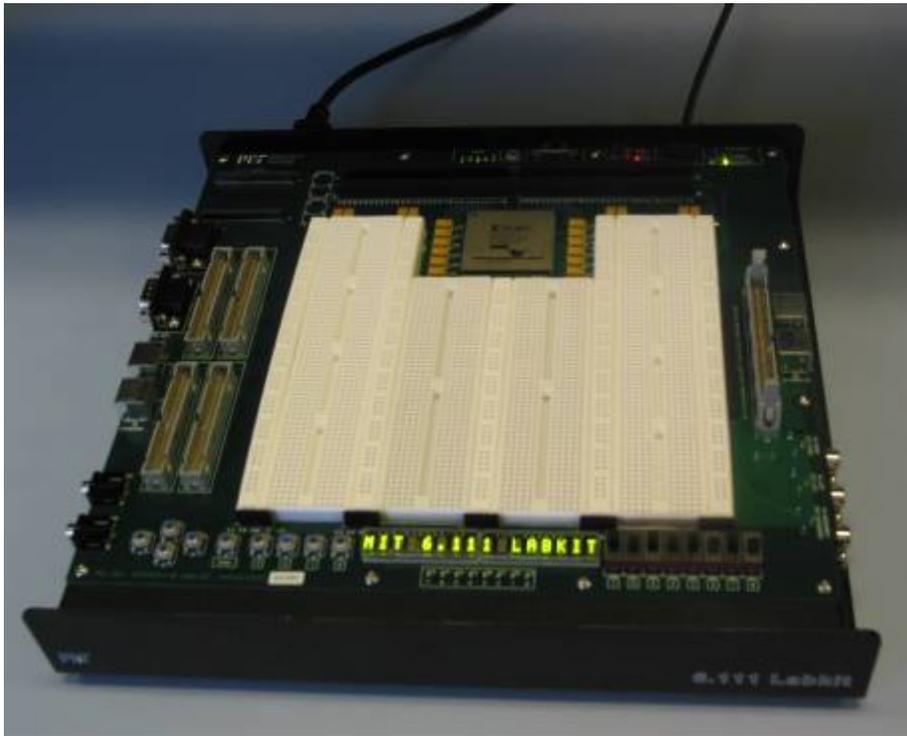
Courtesy of Tony Rinaldo. Used with permission.

- **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)**