L1: 6.111 Course Overview

Acknowledgements: Rex Min

6.111 Staff Contact Information

- In-charge and Lecturer
  - Prof. Anantha P. Chandrakasan – anantha@mtl.mit.edu (38-107, 8-7619)
- Course Assistant:
  - Margaret Flaherty – meg@mtl.mit.edu (38-107, 3-0016)
- Teaching Assistants (TAs)
  - Chris Forker (forker@mit.edu, ph: 3-7350, office hours in 38-600)
  - Charlie Kehoe (ckehoe@mit.edu, ph: 3-7350, office hours in 38-600)
  - Hyunjoo Jenny Lee (jennylee@mit.edu, ph: 3-7350, office hours in 38-600)
- Lab Aides (LAs)
  - TBD
- Technical Instructor
  - Keith Kowal (kkowal@MIT.EDU)
- Stock Clerk
  - Arlin Mason - arlin@mit.edu (38-600, 3-4674)

Recommended Books

- Logic Design:
  - Randy Katz, Gaetano Borriello, Contemporary Logic Design, Pearson Education, 2005
- Verilog: there are plenty of good Verilog books and on-line resources. We recommend the book below:
  - Samir Palnitkar, Verilog HDL, 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 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

Objectives

- On completion of 6.111 students will have confidence to conceive and carry out a complex digital systems design project
- 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, PROMs, FPGAs, and Verilog.
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.

Overview of Labs

- **Lab 1**
  - Learn about the lab kit and wire something
  - Learn about lab equipment in the Digital Lab (38-600): oscilloscopes and logic analyzers
  - Program and test a PAL (Programmable Array Logic Device)
  - Introduction to Verilog
- **Lab 2**
  - Design and implement simple Finite State Machines (FSM)
  - Use Verilog to program an FPGA
  - Learn how to use an SRAM
  - Report and its revision will be evaluated for CI-M
- **Lab 3**
  - Design a complicated system with multiple FSMs (Major/Minor FSM)
  - Memory interface and control
  - System-level I/O

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)
    - Early
    - Detailed
- 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.

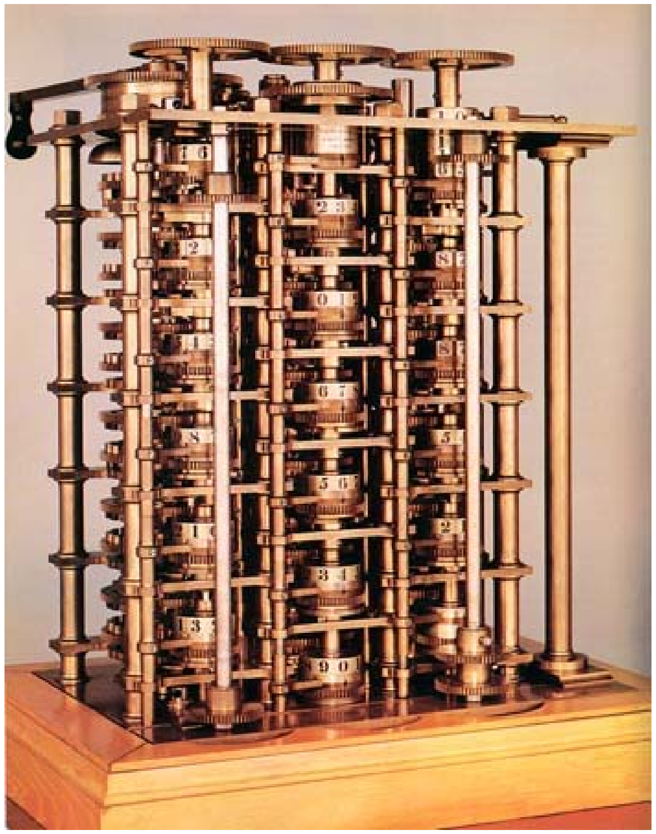
Grading and Collaboration

- Grading Policy
  - We start with a number. Then discuss everyone, especially performance in labs and project. Approximate breakdown:

• Quiz	10%
• 3 Lab Exercises	
❍ Lab 1	10%
❍ Lab 2	10%
❍ Lab 3	15%
• Writing (Lab 2 revision- part of CIM requirement)	10%
• 3 Problem Sets (emphasis on lab concepts)	6%
• Participation (lecture, recitation, labs)	4%
• Final Project	35%

- We impose late penalties
  - No late problem sets will be accepted
  - Labs are penalized 20% per day
  - Final Project MUST be done on time
- Cooperation
  - Please be courteous with shared resources as lab computers and equipment
- Collaboration
  - 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)
    - Project reports should be joint with individual authors specified for each section
    - Copy anything you want (with attribution) for your project report

The First Computer



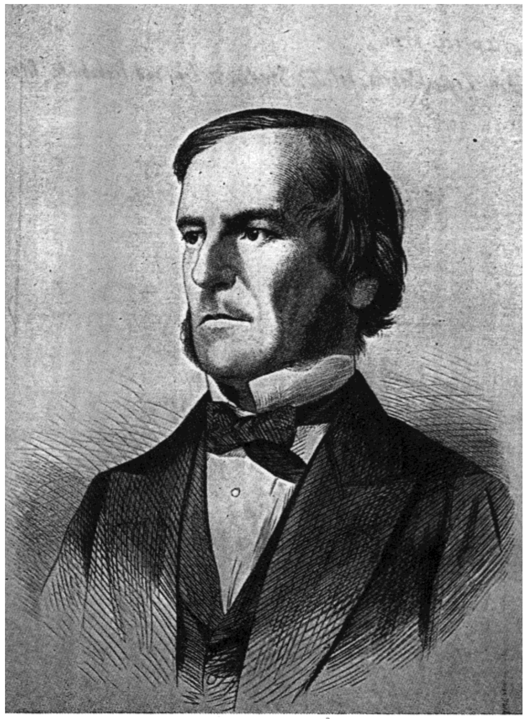
The Babbage Difference Engine (1832)

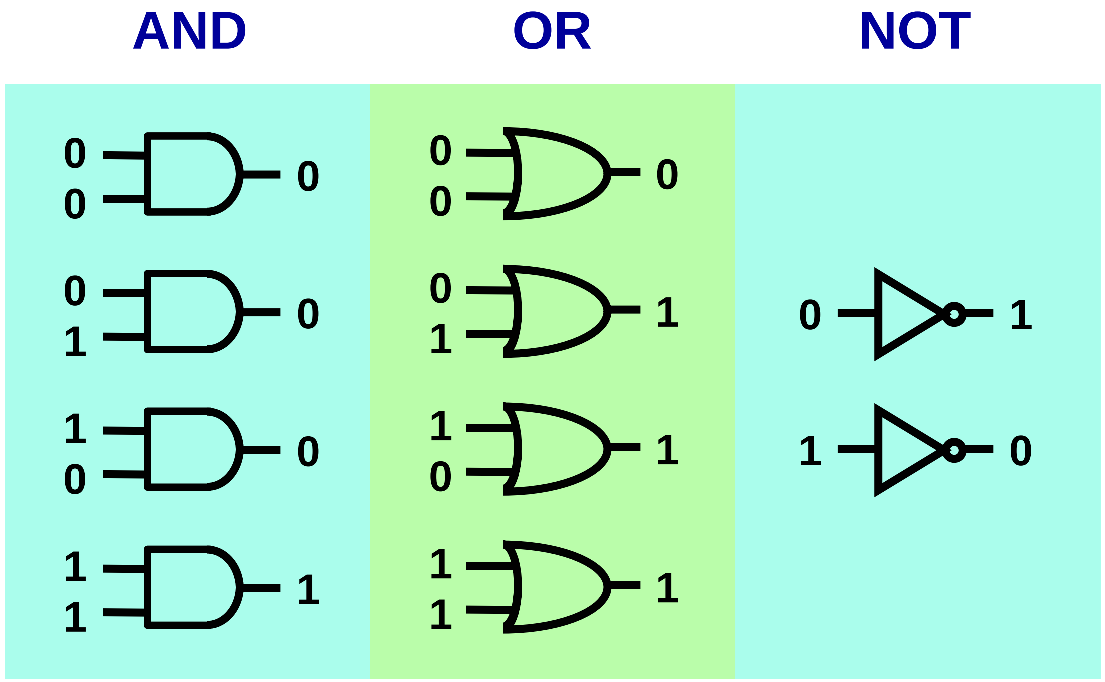
25,000 parts

cost: £17,470

- **The first digital systems were mechanical and used base-10 representation.**
- **Most popular applications: arithmetic and scientific computation**

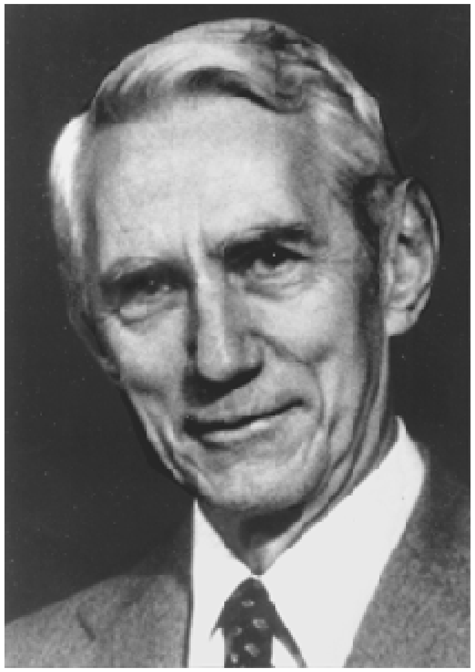
Meanwhile, in the World of Theory…

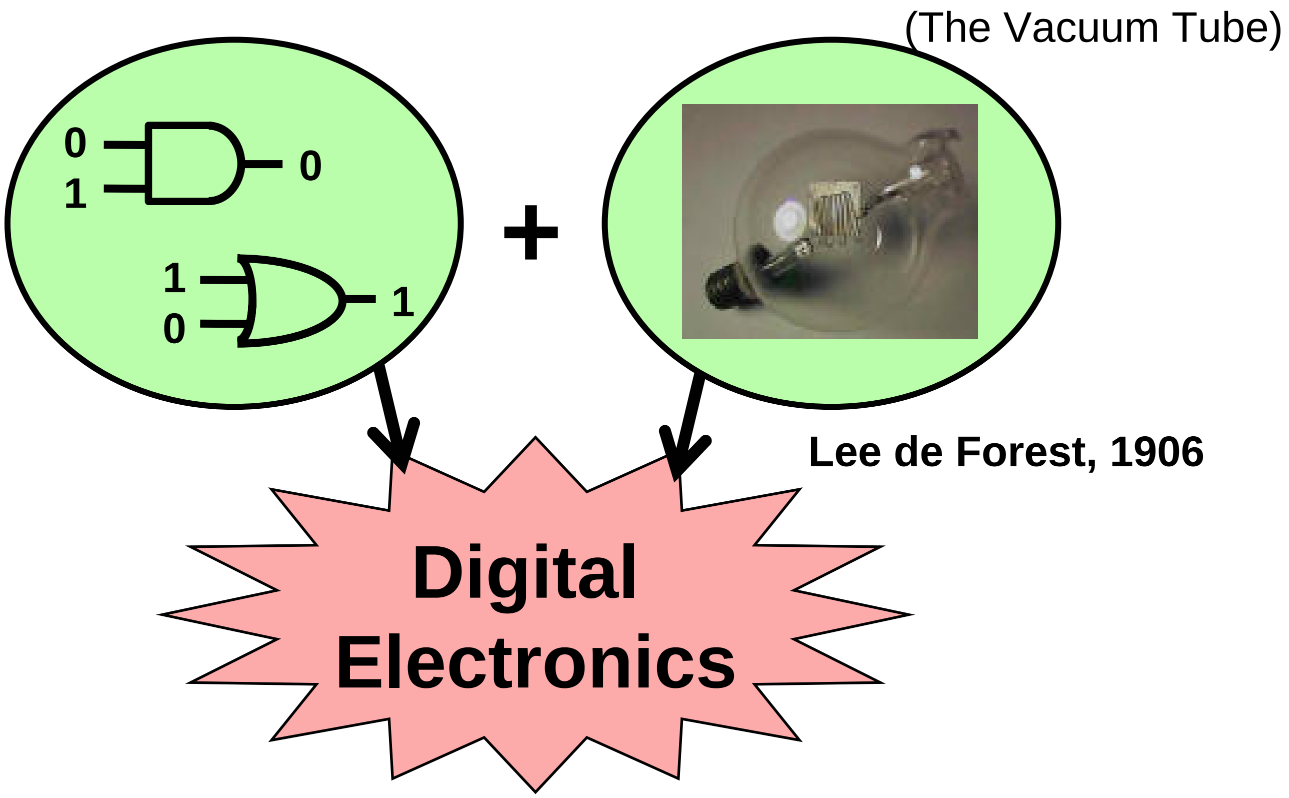

GEORGE BOOLE



- 1854: George Boole shows that logic is math, not just philosophy!
- Boolean algebra: the mathematics of binary values

The Key Link Between Logic and Circuits

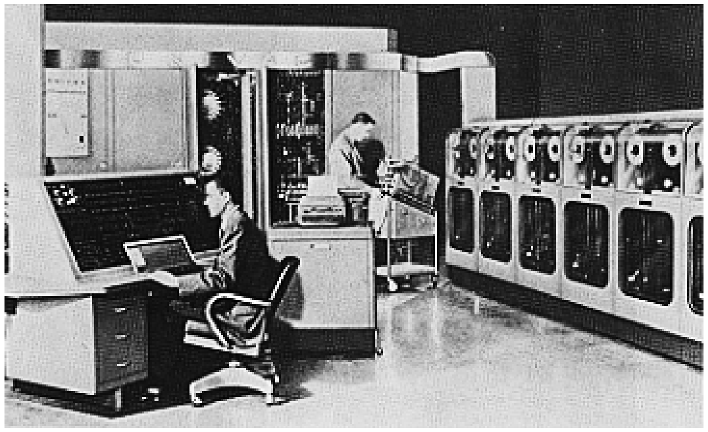




- Despite existence of relays and introduction of vacuum tube in 1906, ***digital*** electronics did not emerge for thirty years!
- Claude Shannon notices similarities between Boolean algebra and electronic telephone switches
- Shannon's 1937 MIT Master's Thesis introduces the world to binary digital electronics

65 Years of Digital Electronics

Vacuum Tubes



UNIVAC, 1951

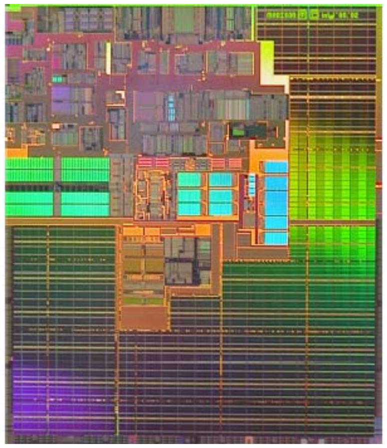
1900 adds/sec

Transistors

IBM System/360, 1964

500,000 adds/sec

VLSI Circuits

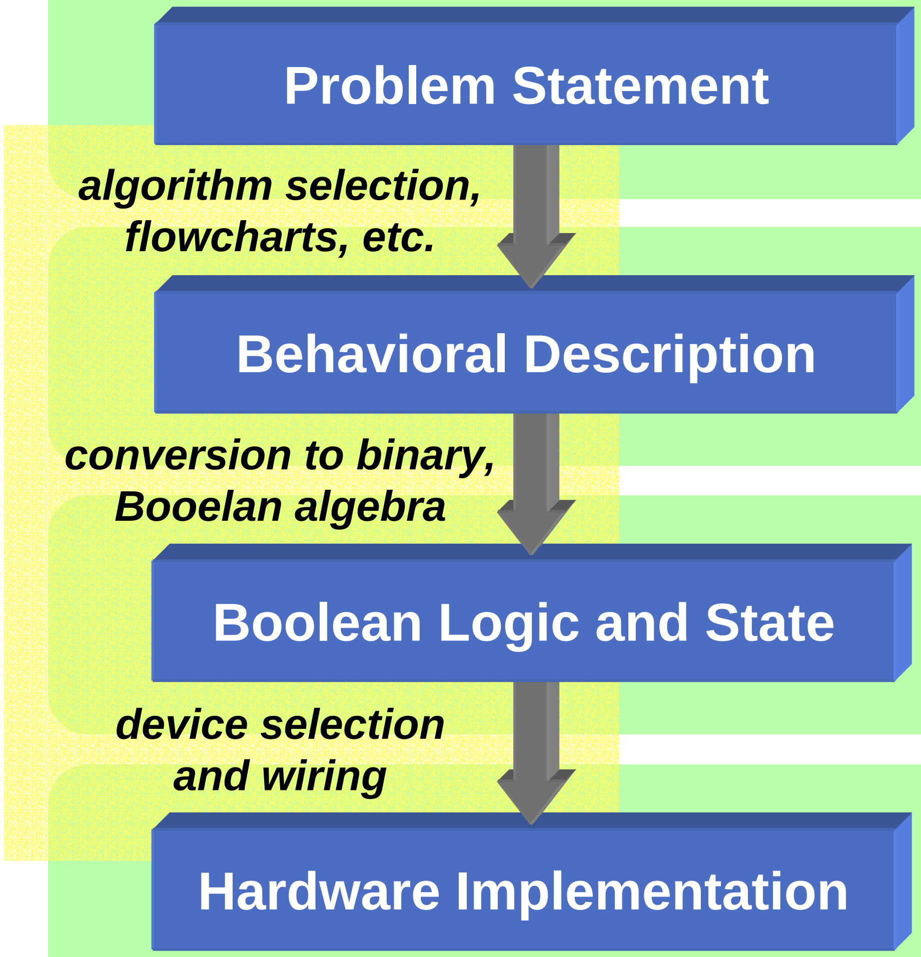


Intel Itanium, 2003

2,000,000,000 adds/sec

Building Digital Systems

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


Problem Statement
algorithm selection, flowcharts, etc.
Behavioral Description
conversion to binary, Booelan algebra
Boolean Logic and State
device selection and wiring
Hardware Implementation

- Problem sets
- Labs & Design project
- Product specs

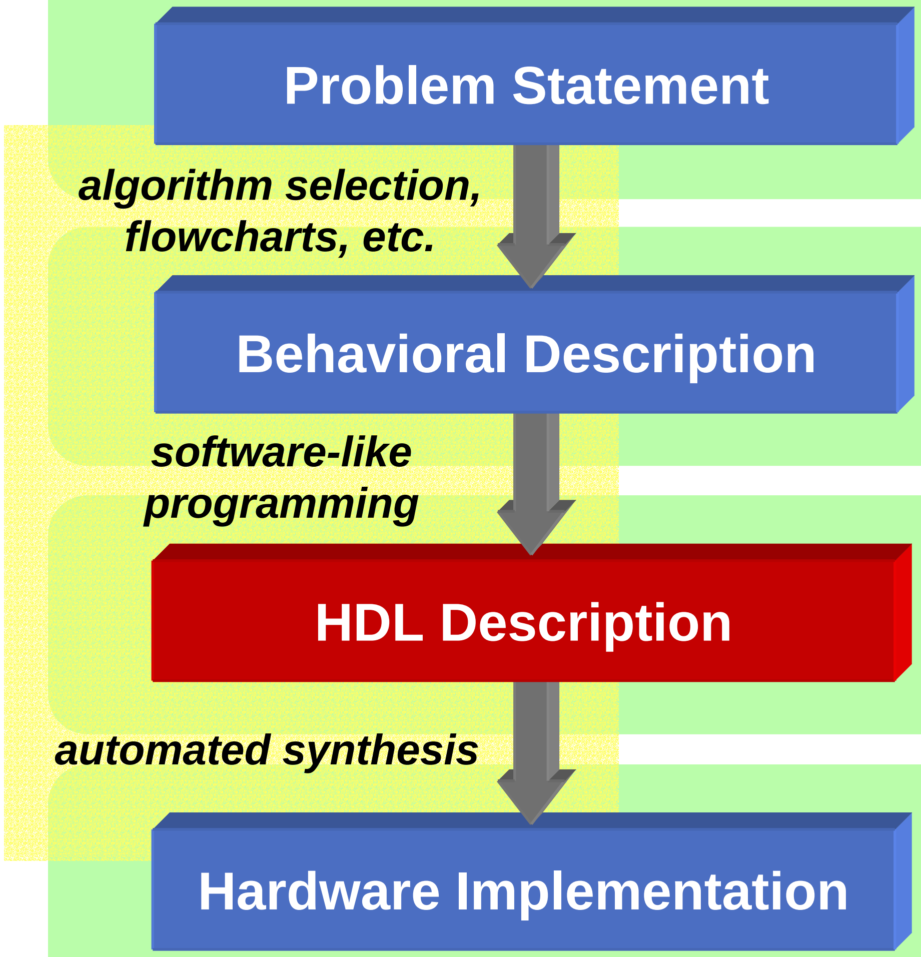
- Algorithms, RTL, etc.
- Flowcharts
- State transition diagrams

- Logic equations
- Circuit schematics

- TTL Gates (AND,OR,XOR…)
- Modules (counter, shifter,…)
- Programmable Logic

Building Digital Systems with HDLs

- Logic synthesis using a Hardware Description Language (HDL) automates the most tedious and error-prone aspects of design



- Problem sets
- Labs & Design project
- Product specs

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

- Verilog code
- VHDL code

- 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

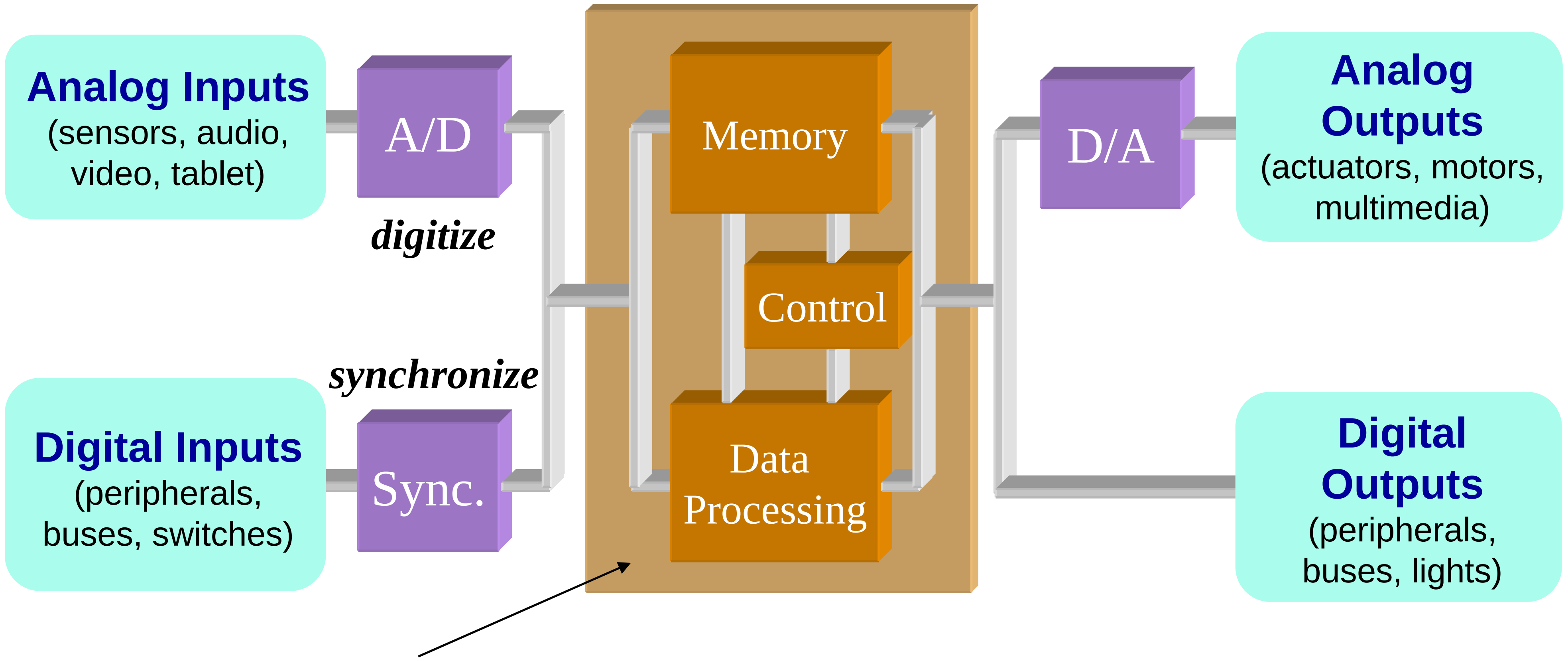
Hardware structures can be modeled effectively in either VHDL and Verilog. Verilog is similar to c and a bit easier to learn.

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

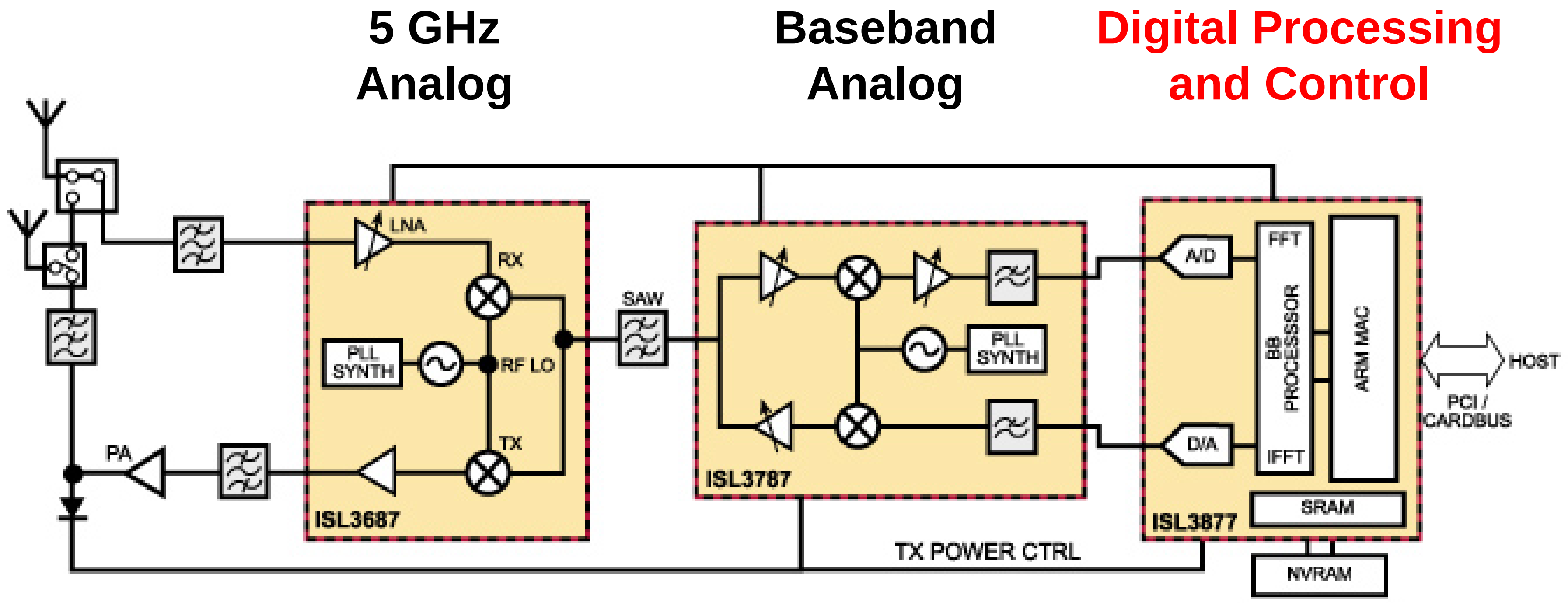
A design mix and match all levels in one design is possible. In general Register Transfer Level (RTL) is used for a combination of Behavioral and Dataflow descriptions

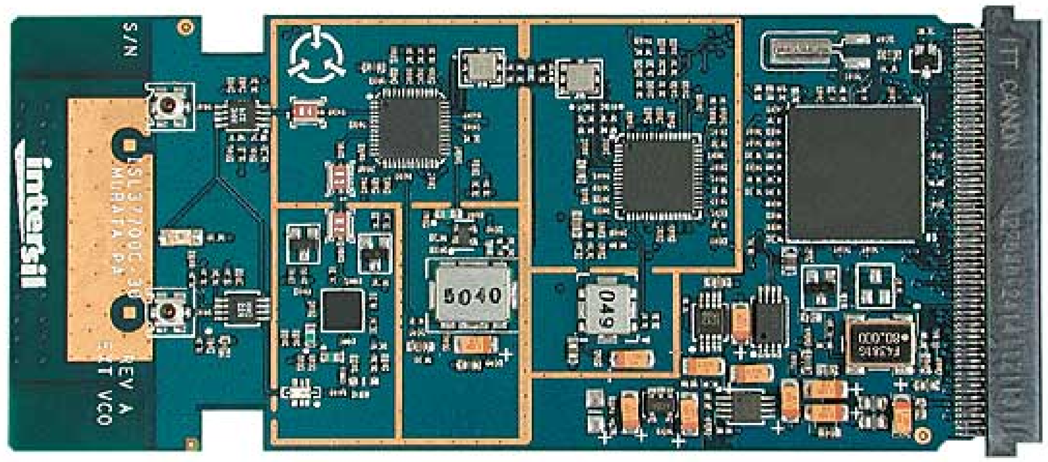
Embedded Digital System


Analog Inputs
(sensors, audio, video, tablet)
A/D
digitize
Digital Inputs
(peripherals, buses, switches)
synchronize
Sync.
Memory
Control
Data Processing
D/A
Analog Outputs
(actuators, motors, multimedia)
Digital Outputs
(peripherals, buses, lights)

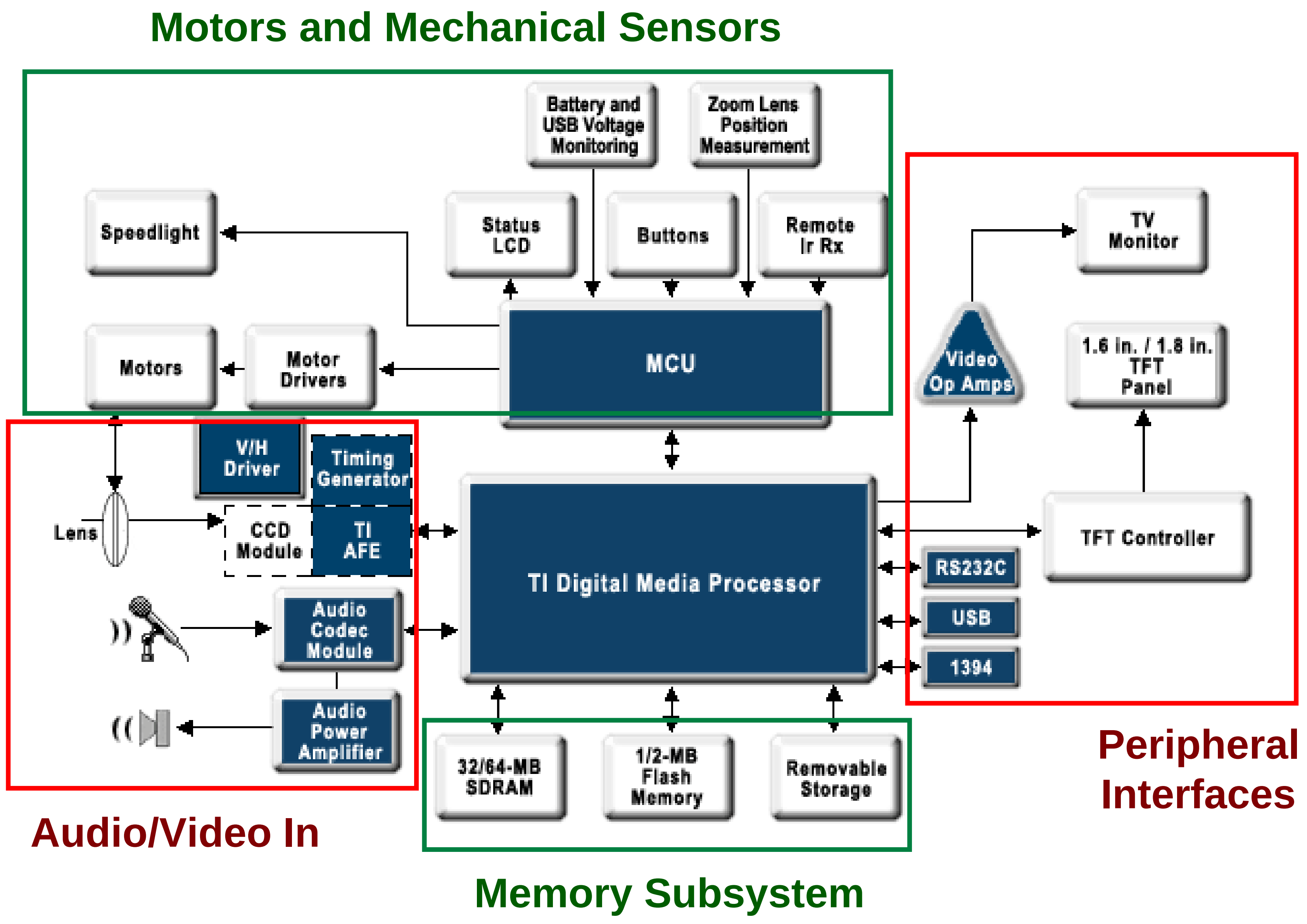
- **Digital processing systems consist of a datapath, memory, and control. Early machines for arithmetic had insufficient memory, and often depended on users for control**
- **Today's digital systems are increasingly embedded into everyday places and things**
- **Richer interaction with the user and environment**

Intersil 54 Mbps 802.11a System

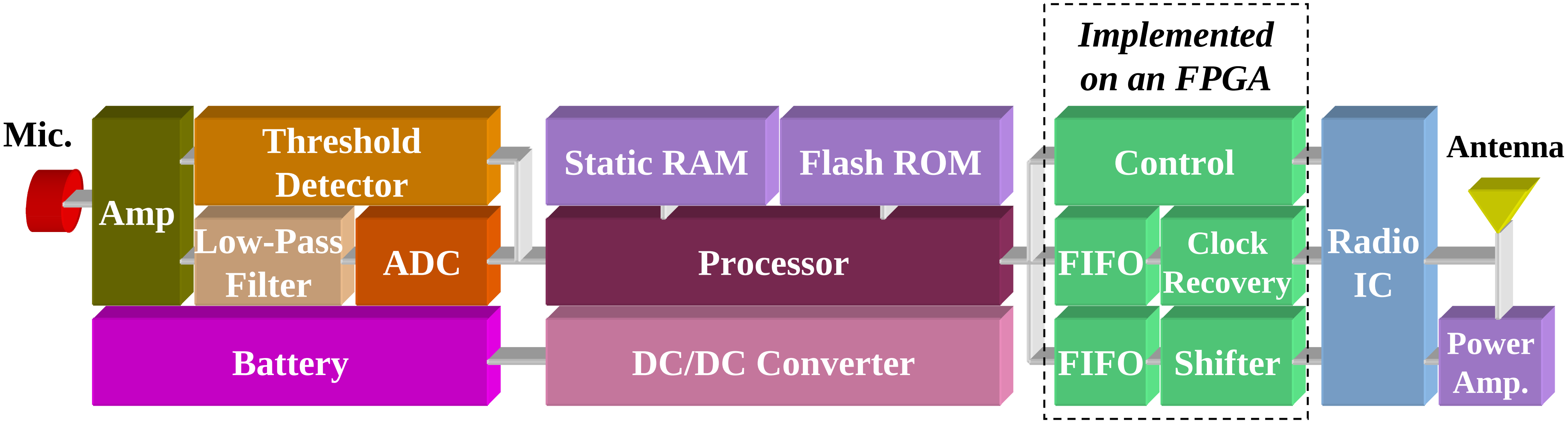




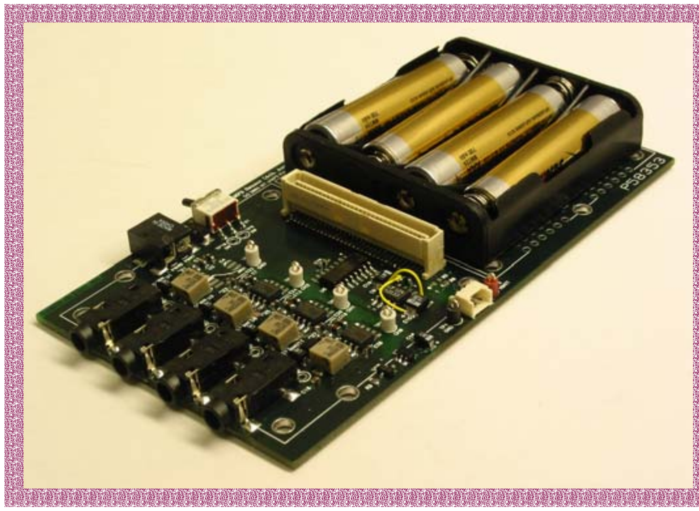
TI Digital Camera Controller

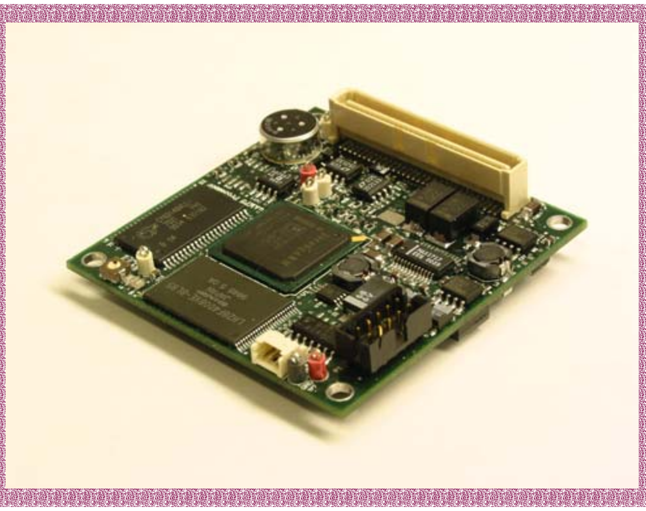


A Wireless Microsensor System (MIT MEng Thesis)



<u>Sensor</u> ***<u>Processor</u>*** ***<u>Radio</u>***


4-channel acoustic


206MHz StrongARM

2.4GHz ISM band

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**

Implementation Strategies

- **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, FPGAs, ASICs, Custom VLSI chips**
  - **These blocks require the least wiring**
  - **We will not use ASICs or Custom VLSI chips**
- **FPGA chips are very complicated**
  - **Designing them with gates is not very productive and error prone**
  - **We need a higher level language such as Verilog to specify the programming of FPGAs**

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
- Synthesis tool can’t be wrong, there is no need for extensive simulation

What Can be Synthesized?

- 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

Verification and Testing

- Design can be fun. Verification/testing is hard work.
- Untested designs are rarely good designs.
- Verification by simulation (and formally through testbenches) 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

Lab Hours - Equipment

- The normal lab hours are:
  - Monday through Thursday – 9:00 AM to 10:45 PM
  - Friday – 9:00 AM to 5:15 PM
  - Saturday – CLOSED
  - Sunday – noon to 10:45 PM
  - Hours for Holidays, Spring Break, etc. is posted on the course website
- Please be out by the indicated time
- New Logic Analyzers and scopes!
- Please do not move or reconfigure computers and other lab equipment (logic analyzers, scopes, power supplies, etc.)
- 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’
  - ‘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.
- We will use the following tools installed on the new PCs (courtesy of Intel):
  - Warp (to program PALs)
  - PAL programming software on two dedicated windows ’98 PCs in the lab
  - ModelSim (powerful front-end simulator for Verilog)
  - MAX+PlusII (simulator and programming software for Altera FPGAs)
  - Office (Microsoft word, power point, etc.)
- In addition, some of the tools are supported on Athena Sun platforms (Solaris and Linux). Please check the course web site for instructions on using them.
- You can use WinSCP to transfer files between the lab PCs and athena

Turn in Your Information Sheet

- Fill in the information and permission form on the last page of the handout (you can change your election at any time during the semester). We need this information to create computer accounts. Please turn it in at the end of lecture.
- Recitation assignments will be posted by Saturday on the 6.111 web page
- Extra handouts will be stored for some time in 38-107. Lecture notes will be available on the web site
- web.mit.edu/6.111/www/s2005
  - 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 (front desk, 38-600)