CS150: Components and Design Techniques for Digital Systems

Professor A. Richard Newton
566 Cory Hall, 2-2967, rnewton@ic.eecs

Spring 1997

- **Understand Components of Digital Systems**
  - Emphasis on **integrated circuit** implementations (CMOS)
  - Especially **programmable devices** (FPGA, PLD, ROM, RAM, etc.)
- **Translation of Design Problems into Hardware Representation**
- **Design of Hardware Logic Systems**
  - Combinational Logic
  - Sequential Logic: Synchronous & Asynchronous
- **Architecture of Digital Systems**
- **Arithmetic Circuits and Number Systems**
- **Introduction to Microprogramming**
- **Testing of Digital Systems**
- **Introduction to Analog Interfaces**

Organizational Matters

- All handouts will be available **on the Web only**
  - http://www-inst.EECS.Berkeley.EDU/~cs150/
  - Please limit your use of the printers! Try to use them for review online only.
- Most lectures will be videotaped and tapes are available for review in McLaughlin Hall.
- All students are required to check the cs150 news group: **cory.eecs.berkeley.edu**

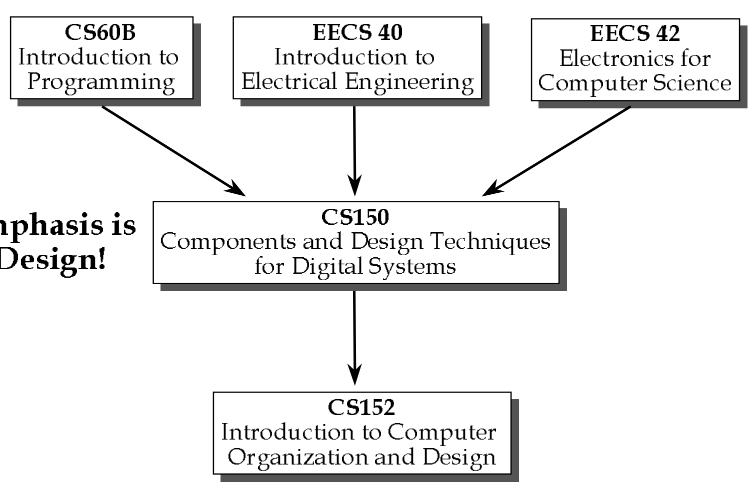
  within 24 hours of any announcement.

Overview

❍ This Lecture:
- ➜ **What is the course all about & why is it important?**
- ➜ **What is a digital system? What is a binary digital system?**
- ➜ **Boolean Algebra, Truth tables**
- ➜ **Operators: inversion, and, or, xor, xnor (eq)**
- ➜ **Design Example: Translating a word problem to a combinational logic function**
- ➜ **Design Example: Translating a word problem into a sequential design language**
- ➜ **State Transition Graph**
- ➜ **State Transition Table**
- ➜ **Mealy and Moore Forms**

Where CS150 Fits

Emphasis is Design!



What is a Digital Circuit?

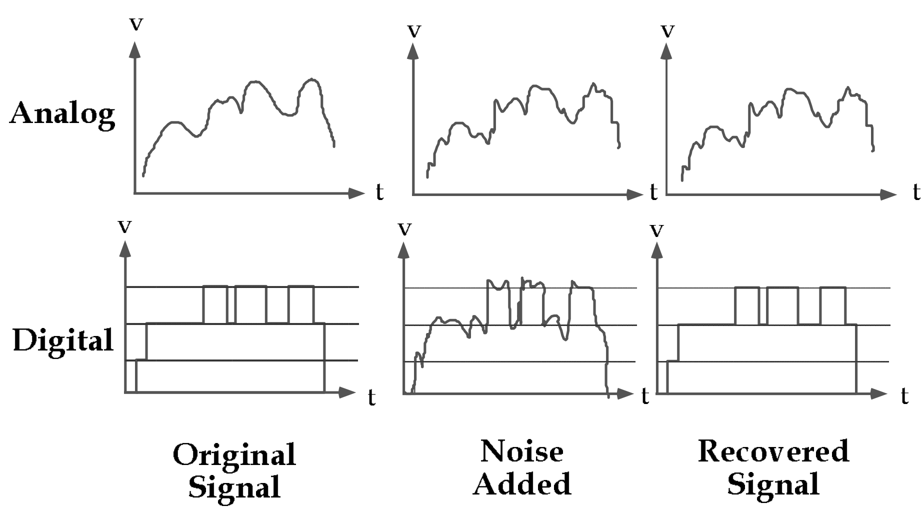
- ❍ Defined by the *interpretation* of the signals (waveforms, if in time) in the circuit.

  Analog
  - ➜ **Continuous Values**
  - ➜ **Fast, economical, low accuracy**
  - ➜ **Susceptible to noise & distortion**

  DIGITAL
  - ➜ **Discrete Levels**
  - ➜ **Accuracy related to cost (no. of "bits")**
  - ➜ **Less susceptible to noise**
- ❍ *Binary* - 2 Levels or States
- ❍ *Multi-valued* - More than two Levels

CS150 Newton 1.1.5

Signal Recovery from Noise



Remember: All digital circuits are implemented using analog signals. Their analog characteristics are often important!

CS150 Newton 1.1.6

Simplified Model of Design Representation

changing role of time

Clocked, Synchronous Sequential Logic Circuits & Systems
As below, but the output depends only on the previous input values at times determined by the value(s) of a special "clock" signal.
e.g. Microcomputers, DSPs, controllers, some I/O devices.

Asynchronous Sequential Logic Circuits & Systems
As below, but output depends on previous inputs and the order in which they changed - from the moment the power was turned on!
e.g.. Latches, many I/O systems, some high-speed circuits.

Combinational Logic Circuits & Systems
Digital: Binary, output is independent of previous inputs and is always the same if you wait long enough (no storage).
e.g. Decoders/Encoders, format converters, full adder

Physical Design
Analog (Voltages & Currents): Noise, power, heat, wires, size, etc.
e.g. ICs, Boards, Modules, Boxes, Systems

Classification of Designs

❍ COMBINATIONAL

Output *only* depends on the instantaneous input values.

IDEAL: Output responds instantly.

REALITY: Propagation delay, spurious outputs ("glitches") that *must* eventually settle.

=> No feedback path exists from the outputs to the inputs.*

❍ SEQUENTIAL

Circuit contains "memory" (store) so the output *may* depend on the sequence of previous inputs (since power-up!)

REALITY: Output depends on the *duration* of the inputs.

=> There must be at least one feedback path from the outputs to the inputs.

* Is this *really* true? Why?

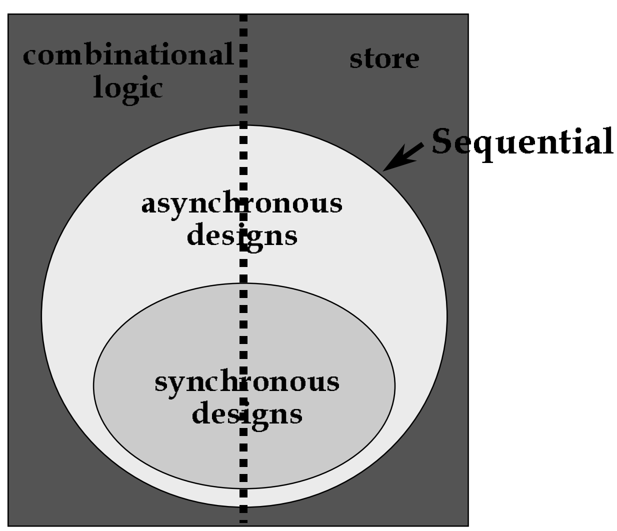
Classification of Designs

- ❍ **SYNCHRONOUS**
  - ➜ All feedback occurs at the same time, as defined by a special signal called a clock.
- ❍ **ASYNCHRONOUS**
  - ➜ Feedback occurs whenever values change. No special clock signals.

Remember: Avoid asynchronous design if you can! Try to identify the unavoidable asynchronous interfaces and convert them to synchronous form as reliably and with as few gates as possible.

CS150 Newton 1.1.9

Representing the Digital Design Problem: Basic Classification of Designs



CS150 Newton 1.1.10

Boolean Algebra

George Boole **1854** **Invented multi-valued discrete algebra**
E. V. Huntington **1904** **Postulates & Theorems**
Claude E. Shannon **1938** **Applied to relay circuits**

	TRUE (logic1)	**FALSE (logic 0)**
Relays:	Contacts closed	Contacts open
MOS logic:	High voltage (>4.8V)	Low voltage (<0.2V)
Magnetic core:	Magnetized up	Magnetized down
Fluids:	Fluid flowing	Not flowing

Fundamental Operations

POSITIVE LOGIC ("Active High"):

True = Logic1 (high V), False = Logic0 (low V)

NOTATION:

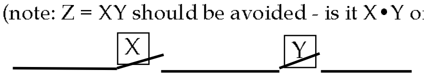
NOT (Negation, complementation, inversion)

$Z = \overline{X}$, $Z = X'$, $Z = \sim X$

AND: Switches in series, "product"

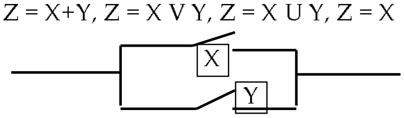
$Z = X \bullet Y$, $Z = X \wedge Y$, $Z = X \cap Y$, $X = X \,\&\, Y$

(note: Z = XY should be avoided - is it X•Y or a signal named "XY"?)



OR: Switches in parallel, "sum"

$Z = X+Y$, $Z = X \vee Y$, $Z = X \cup Y$, $Z = X \mid Y$



XOR: Exclusive OR - either X or Y but not both

$Z = X \oplus Y$; (XNOR or EQ: $Z = X \odot Y = \overline{X \oplus Y}$)

Design Example: Word Problem

A car will not start when the ignition is on iff:

(a) Doors are closed & the seat belts are unbuckled
(b) The seat belts are buckled & the parking break is on
(c) The parking break is off and the doors are not closed

When can the car start if the switch is on?

CS150 Newton 1.1.13

Truth Tables

- A **Truth Table** is a tabular notation for representing the logic value of a function for *all possible combinations* of the values of its arguments.

X	Z
0	**1**
1	**0**

NOT

X	Y	Z
0	**0**	**0**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

AND

X	Y	Z
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**1**

OR

X	Y	Z
0	**0**	
0	**1**	
1	**0**	
1	**1**	

XOR

CS150 Newton 1.1.14

Design Example: Translation to Truth Table

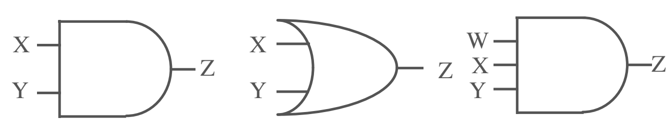
Define variables:
S = car starts, B = belts are buckled, D = Door is closed, P = brake is on

B	D	P	S'	S
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

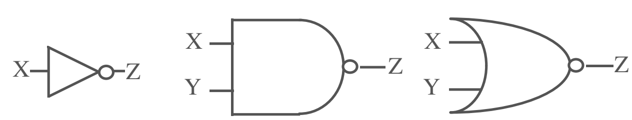
S =

CS150 Newton 1.1.15

Schematic Symbols



AND: Z = X•Y OR: Z = X + Y 3-input AND: Z = W•X•Y

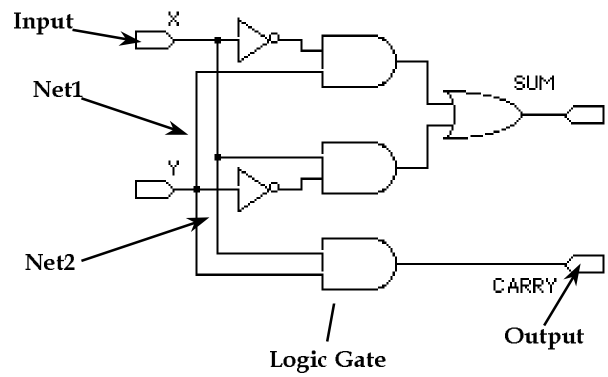


inverter: Z = X' NAND: Z = (X•Y)' NOR: Z = (X+Y)'

XOR: Z = X ⊕ Y XNOR: Z = (X ⊕ Y)' = X⊙Y

CS150 Newton 1.1.16

Schematic Diagram Representation

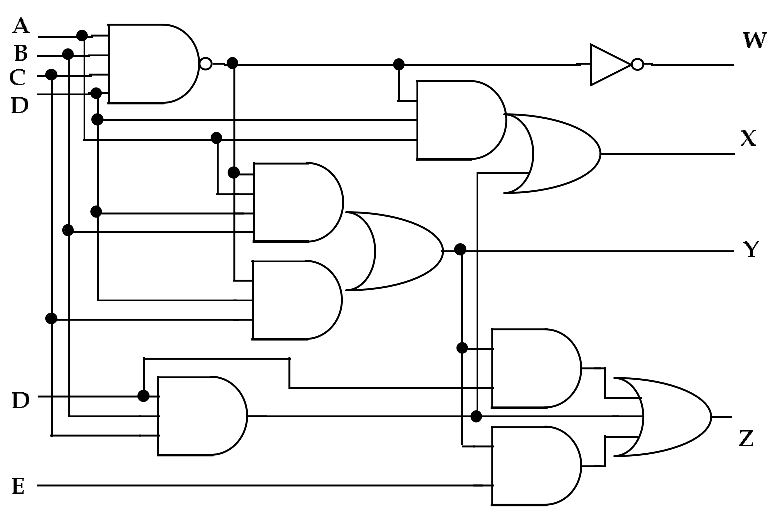


Net: an electrically-connected collection of wires.

Netlist: a tabulation of gate inputs & outputs and the nets they are connected to.

CS150 Newton 1.1.17

`Complex´ Schematic Diagram



CS150 Newton 1.1.18

Design Example: Schematic Diagram

B

S

D

P

B

S

D

P

CS150 Newton 1.1.19

Design Example: Automobile Lock

The automobile theft rate in Muldavia is so high that CyclesPerSecond rental car agency has decided to add a new security device to their cars. The initial system design has been completed and our consulting firm has been retained to implement the device. The agency designers hand us the following description:

"Please build us a small black box (2"x3"x0.5") we can attach to the dash that consists of a keypad (keys 0-9) and two LEDs, one green and one red. It should perform as follows:

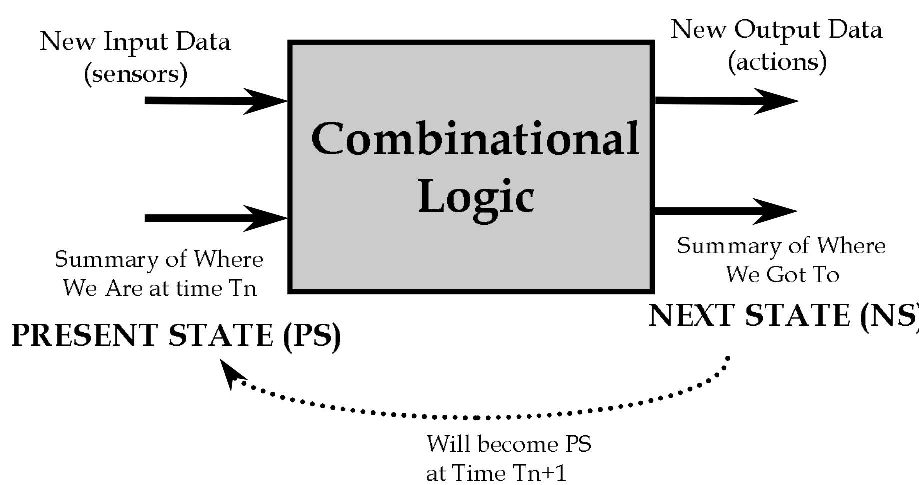
- ➜ When the **ignition is turned on**, the **red LED should light up** and the **keypad is activated**. However, **the car will not start**.
- ➜ If the **driver enters a correct four-digit code**, the **green LED goes on** as well and **the car can now be started** by **turning the ignition switch further clockwise** to the start position.
- ➜ If the **code is not correct**, the **green light will not go on and the car will not start**."

CS150 Newton 1.1.20

Where Do We Start?

- ❍ Write down the inputs and outputs and list the (symbolic) values they can take.
- ❍ Choose a "language" in which to express the behavior of the machine. Is a truth table sufficient here? How would you use it?
- ❍ For sequential systems, we will start by using *state transition tables* or *state transition graphs*.
- ❍ We will be assuming *discrete-valued time* - "instant" to "instant." At any particular instant, the finite number of storage elements in the machine will have particular, well-defined values stored in them - we will be describing problems which can be implemented directly with a *Finite-State Machine*.

General Structure of Our Problem at Time Tn



Choosing a Language to Represent the Problem

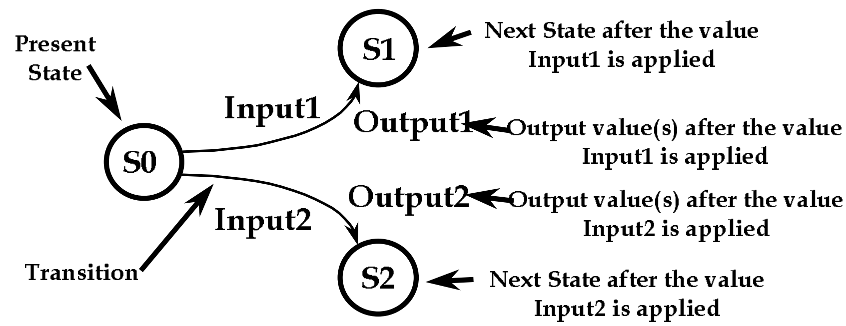
We need a "language" to represent:

(1) The Present State of the machine

(2) For each possible input value:

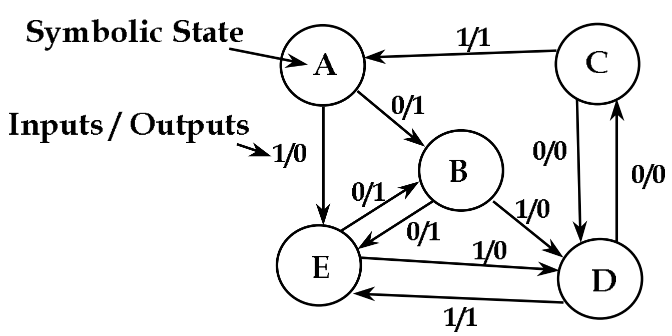
(2a) The corresponding output value(s)

(2b) The corresponding Next State



CS150 Newton 1.1.23

Example Finite-State Machine State Transition Diagram (Mealy)



CS150 Newton 1.1.24

Example Finite-State Machine State Transition Table (Mealy)

Primary Input	0		1	
Present State	**Next State**	**Primary Output**	**Next State**	**Primary Output**
A	B	1	E	0
B	E	1	D	0
C	D	0	A	1
D	C	0	E	1
E	D	1	D	0

CS150 Newton 1.1.25

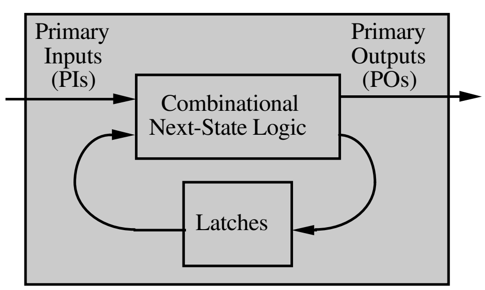
Design Example: State Transition Graph

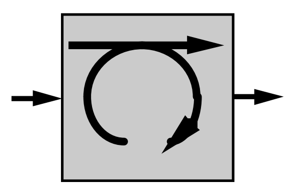
CS150 Newton 1.1.26

Design Example: State Transition Table

CS150 Newton 1.1.27

Finite-State Machines

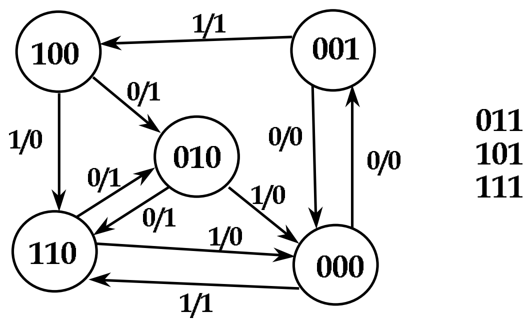




Mealy Machine

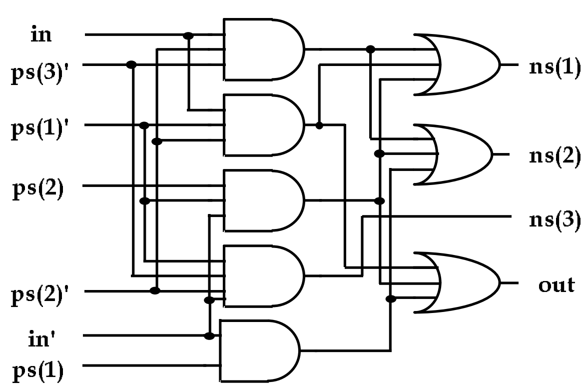
CS150 Newton 1.1.28

Example Finite-State Machine Encoded States (Mealy)



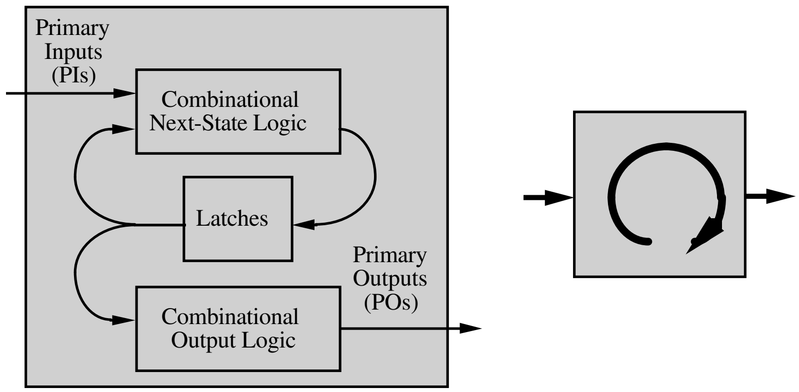
CS150 Newton 1.1.29

Example Finite-State Machine Next-State Logic (Mealy)



CS150 Newton 1.1.30

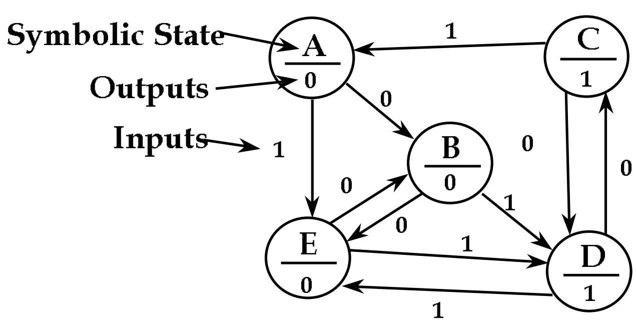
Finite-State Machines



Moore Machine

CS150 Newton 1.1.31

Example Finite-State Machine State Transition Diagram (Moore)



CS150 Newton 1.1.32

Example Finite-State Machine State Transition Table (Moore)

Present State	Next State		Primary Output
Primary Input	**0**	**1**	
A	B	E	0
B	E	D	0
C	D	A	1
D	C	E	1
E	D	D	0

Design Example: Moore Graph and Table

This Space Left Intentionally Blank

Exercise For Thursday

- ❍ List the **five things that bug you the most** right now - that you would like to see a better solution to and where technology might be able to help.
- ❍ Examples:
  - ➜ Food that goes bad in the refrigerator
  - ➜ Trying to find an empty space in a crowded parking lot
  - ➜ The increase in volume associated with advertising on TV
  - ➜ Remote controls for consumer products
- ❍ To be **submitted in class, with your name, on Thursday**.

Top Complaints, Spring 1996

Tally	Concern
28	**traffic, transportation problems, tailgaters, slow drivers in the fast lane**
19	**alarm clocks: setting them, waking up, loud noises, failure**
19	**financial problems**
17	**forgetting: lights, fan, locks**
17	**waiting, in line or otherwise**
15	**commercials: long, annoying, false advertising, loud, phone solicitations**
15	**telebears, voicemail, phone menus**
12	**cooking problems, washing dishes**
12	**keys: not working, looking for them in the dark, damaging clothes, locking them in car**
11	**computer: slow, crashing, screen freezing**
11	**school problems: scheduling, registration, overcrowded, requirements, bureaucracy**
11	**television: bad reception, broken, poor design**
10	**forgetting names, passwords, phone numbers, meetings, dates**
10	**homework**
10	**losing books, keys, or personal items**
10	**poor design of consumer electronics**

Top Complaints, Spring 1996

Tally	Concern
9	not getting enough sleep, having to wake up early, early classes, being woken up
8	falling asleep: in class, while driving, with lights and heater on, oversleeping
8	grades: bad grades, being judged on grades
8	parking problems
8	pollution, litter
8	public transportation-slow, off schedule, missing bus, waiting
8	rain
8	tobacco smoke
7	aggressive, insecure, arrogant, intolerant, obnoxious, ignorant, lazy people
7	answering machine, "empty messages"
7	bathroom: putting the seat down, dirty, no paper, etc.
7	busy signal on phone, when using modem, call waiting
7	love, ex girlfriends, ex-boyfriends, friendship or lack of

CS150 Newton 1.1.37