EECS150

Components and Design
Techniques for Digital Systems

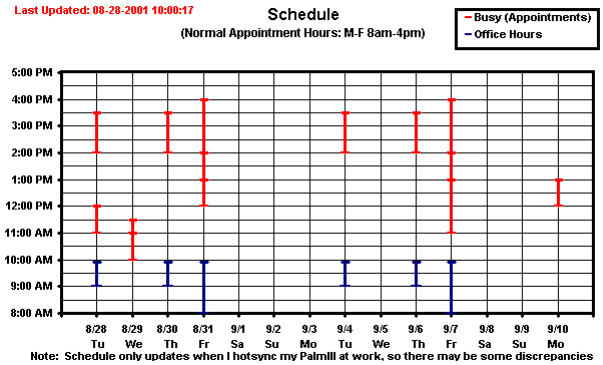
Fall 2001

Personnel

- Professor:
  - Vivek Subramanian
  - viveks@eecs.berkeley.edu
  - 571 Cory Hall
  - (510) 643-4535
- Readers:
  - Ming Wai Choy
  - Cuong (Duke) Hoang
  - Guan Hui
  - Lijue Zhong
- TAs:
  - Mike Lowey (*Head TA)*
  - Mark Feng
  - Neha Kumar
  - Randy Huang
  - Jason Hu
  - Richard Lu
  - Richard Pon
  - Yuh Meei Seah
  - Gabriel Eirea
  - Kerry Kimes

Office Hours

- Tuesday/Thursday, 9-10am
- Friday, 8-10am
- Or by appointment



Student Conduct

- Anyone caught cheating will be failed and/or expelled from the University
- Activities classified as cheating include any activities that give you an unfair advantage over other students, including, but not limited to:
  - Copying solutions from other students
  - Sharing your solutions with other students
  - Altering laboratory equipment to prevent other students from using it

Timing

- **Lectures:**
  - 10 Evans, Tuesdays & Thursdays, 2:00pm-3:30pm
- **Discussions:**
  - Section 101 – Mondays, 11:00am-12:00pm, 405 Davis
  - Section 102 – Mondays, 2:00pm-3:00pm, 75 Evans
  - Section 103 – Mondays, 3:00pm-4:00pm, 71 Evans
  - Section 104 – Wednesdays, 10:00am-11:00am, 9 Evans
  - Section 105 – Wednesdays, 4:00pm-5:00pm, 71 Evans
  - Section 106 – Thursdays, 9:00am-10:00am, 71 Evans
  - Section 107 – Thursdays, 11:00am-12:00pm, 285 Cory

Timing - Labs

- **Lab Lecture:**
  - Fridays, 2:00pm-3:00pm, 10 Evans
- **Laboratories:**
  - Section 011 – Mondays, 9:00am-12:00pm, 204B Cory
  - Section 012 – Mondays, 5:00pm-8:00pm, 204B Cory
  - Section 013 – Tuesdays, 9:00am-12:00pm, 204B Cory
  - Section 014 – Tuesdays, 5:00pm-8:00pm, 204B Cory
  - Section 015 – Wednesdays,9:00am-12:00pm,204B Cory
  - Section 016 – Wednesdays, 5:00pm-8:00pm, 204B Cory
  - Section 017 – Thursdays, 9:00am-12:00pm, 204B Cory
  - The first lab will be held in 349 Davis. The rest of the labs will be held in 204b Cory.
  - There are also two complete lab setups in the Student IEEE office, 204A Cory.

Laboratory Policies

- Be aware of your environment, and take responsibility for your safety.
- Take good care of the lab. Report any equipment problems to your TA, and make a log entry in the lab logbook.
- Do not modify any hardware or software.
- No eating or drinking in lab at any time. Lab must be left clean and orderly after each session.
- After the first week you will be given your individual account for which you will be responsible. Do not share your account/password, do not leave without logging out, and do not abuse the privilege.

Policies

- Homeworks are posted on Thursdays, collected the following Friday at noon. Submit assignments to CS150 box, on door of Cory 218.
- It is excellent practice to complete as much of your laboratory as you can before you actually arrive for your lab. Computers in 349 Davis are available with cardkey access for your use 24/7.
- TA's will help with problems and check off labs. If you don't get checked off during your assigned lab, you can get checked off during the TA's office hours or during the Make-Up lab ONLY.
- Labs must be checked off by Friday at noon during the week it is assigned for you to receive full credit. Half credit will be assigned for getting checked off up to one week late.

Policies

- Students will work in groups of two. Your partner can be anyone from your assigned section, and can change from week to week.
- For the final project, you MUST select a partner from your lab section. Choose your lab section carefully.
- If you want to change lab sections, you need to find someone who can swap with you. The head TA Mike Lowey is responsible for all lab section changes.
- The 204B lab is only open during lab sessions and TA office hours. It is reserved at other times for TA/staff maintenance and development. For your convenience, 349 Davis is open 24 hours a day, 7 days a week.

Grading

- Homework (graded on effort, not correctness): 10%
- Laboratories: 18%
- Tests (3): 21%
- Project: 31%
- Final: 20%

What will we learn in EECS 150?

- Language of logic design
  - Boolean algebra, logic, state, timing, CAD tools
- Concept of state in digital systems
  - Analogous to variables, program counters in software
- How to specify/simulate/compile our designs
  - Hardware description languages
  - Tools to simulate the workings of our designs
  - Logic compilers to synthesize the hardware blocks
  - Mapping onto programmable hardware

A quick history lesson

- 1850: George Boole invents Boolean algebra
- 1938: Claude Shannon links Boolean algebra to switches
- 1945: John von Neumann develops first stored program computer
  - Its switching elements are vacuum tubes (a big advance from relays)
- 1946: ENIAC--world's first all electronic computer
  - 18,000 vacuum tubes
  - Several hundred multiplications per minute
- 1947: Shockley, Brittain, and Bardeen invent the transistor

What is digital hardware?

- Collection of devices that sense and/or control wires carrying a digital value (i.e., a physical quantity interpreted as a "0" or "1")
  - logic where voltage < 0.8V is "0" and > 2.0V is "1"
  - e.g., orientation of magnetization signifies "0" or "1"
- Primitive digital hardware devices
  - Logic computation devices (sense and drive)
    - two wires both "1" - make another be "1" (AND)
    - at least one of two wires "1" - make another be "1" (OR)
    - a wire "1" - then make another be "0" (NOT)
  - Memory devices (store)
    - store a value
    - recall a value previously stored

drive →

Trends in digital design

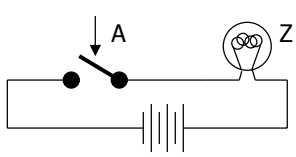
- Big change in how industry does design
  - Larger designs
  - Shorter time to market
- Scale
  - Pervasive use of computer-aided design tools
  - Multiple levels of design representation
- Time
  - Emphasis on abstract design representations
  - Programmable rather than fixed function components
  - Automatic synthesis techniques
- Cost
  - Use of simulation to debug designs

Computation

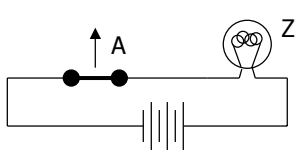
- Basic units of computation:
  - representation: "0", "1" on a wire
    set of wires (e.g., for binary integers)
  - assignment: x = y
  - data operations: x + y – 5
  - control:
    - sequential statements: A; B; C
    - conditionals: if x == 1 then y
    - loops: for (i = 1 ; i == 10, i++)
    - procedures: A; proc(...); B;

Physical Implementation - Switch

- Implementing a simple circuit (arrow shows action if wire changes to "1"):



close switch (if A is "1" or asserted) and turn on light bulb (Z)



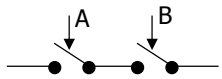
open switch (if A is "0" or unasserted) and turn off light bulb (Z)

$Z \equiv A$

Switches (cont'd)

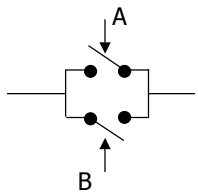
- Compose switches into more complex ones (Boolean functions):

AND



Z ≡ A <u>and</u> B

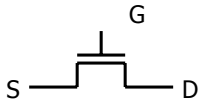
OR



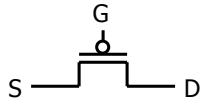
Z ≡ A <u>or</u> B

MOS transistors

- MOS transistors act as switches as follows:
  - if voltage on gate terminal is (some amount) higher/lower than source terminal then a conducting path is established between drain and source terminals

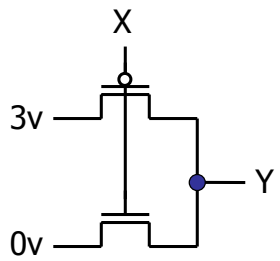


n-channel
open when voltage at G is low
closes when:
voltage(G) > voltage (S) + ε



p-channel
closed when voltage at G is low
opens when:
voltage(G) < voltage (S) – ε

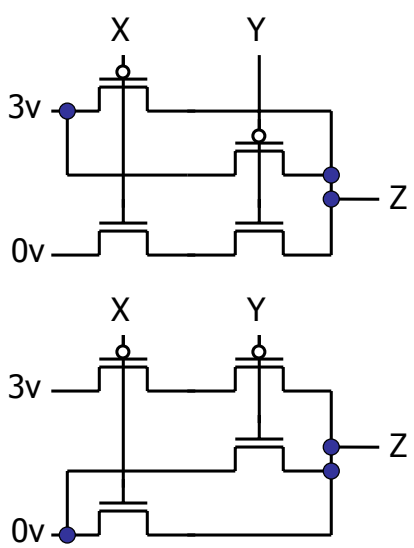
MOS networks



what is the relationship between x and y?

x	y
0 volts	
3 volts	

Two input networks



what is the relationship between x, y and z?

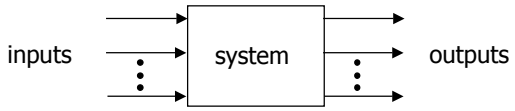
x	y	z
0 volts	0 volts	
0 volts	3 volts	
3 volts	0 volts	
3 volts	3 volts	

Logic levels in the real world

Technology	State 0	State 1
Relay logic	Circuit Open	Circuit Closed
CMOS logic	0.0-1.0 volts	2.0-3.0 volts
Transistor transistor logic (TTL)	0.0-0.8 volts	2.0-5.0 volts
Fiber Optics	Light off	Light on
Dynamic RAM	Discharged capacitor	Charged capacitor
Nonvolatile memory (erasable)	Trapped electrons	No trapped electrons
Programmable ROM	Fuse blown	Fuse intact
Bubble memory	No magnetic bubble	Bubble present
Magnetic disk	No flux reversal	Flux reversal
Compact disc	No pit	Pit

CombinationalLogic

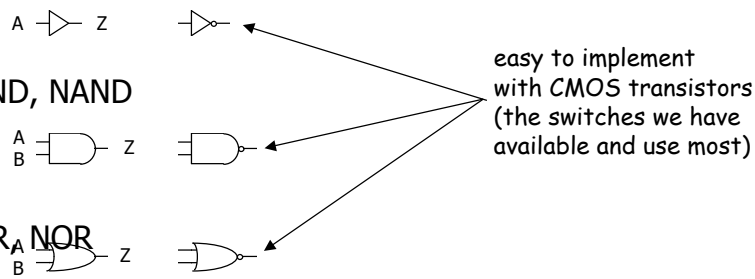
- A simple model of a digital system is a unit with inputs and outputs:



- Combinational means "memory-less"
  - a digital circuit is combinational if its output values only depend on its input values

Combinational logic symbols

- Common combinational logic systems have standard symbols called logic gates
  - Buffer, NOT



Sequential logic

- Sequential systems
  - Exhibit behaviors (output values) that depend not only on the current input values, but also on previous input values
- In reality, all real circuits are sequential
  - The outputs do not change instantaneously after an input change
- A fundamental abstraction of digital design is to reason (mostly) about steady-state behaviors
  - Look at outputs only after sufficient time has elapsed for the system to settle down

An example

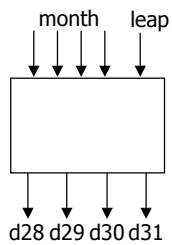
- Calendar subsystem: number of days in a month (to control watch display)
  - used in controlling the display of a wrist-watch LCD screen
  - inputs: month, leap year flag
  - outputs: number of days

Implementation in software

```
integer number_of_days ( month,
  leap_year_flag) {
   switch (month) {
      case 1: return (31);
      case 2: if (leap_year_flag == 1) then return
        (29) else return (28);
      case 3: return (31);
      ...
      case 12: return (31);
      default: return (0);
   }
}
```

Combinational Implementation

- Encoding:
  - binary number for month
  - four wires for 28, 29, 30, and 31



month	leap	d28	d29	d30	d31
0000	–	–	–	–	–
0001	–	0	0	0	1
0010	0	1	0	0	0
0010	1	0	1	0	0
0011	–	0	0	0	1
0100	–	0	0	1	0
0101	–	0	0	0	1
0110	–	0	0	1	0
0111	–	0	0	0	1
1000	–	0	0	0	1
1001	–	0	0	1	0
1010	–	0	0	0	1
1011	–	0	0	1	0
1100	–	0	0	0	1
1101	–	–	–	–	–
111–	–	–	–	–	–

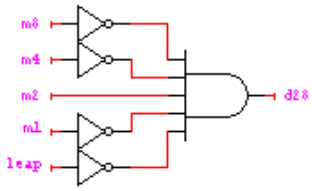
Combinational example (cont'd)

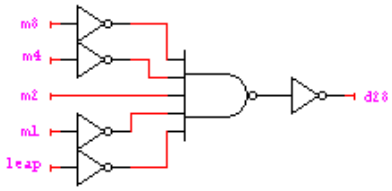
- Truth-table to logic to switches to gates
  - d28 = 1 when month=0010 and leap=0
  - d28 = m8'•m4'•m2•m1'•leap'
  - d31 = 1 when month=0001 or month=0011 or ... month=1100
  - d31 = (m8'•m4'•m2'•m1) + (m8'•m4'•m2•m1) + ... (m8•m4•m2'•m1')
  - d31 = can we simplify more?

month	leap	d28	d29	d30	d31
0001	–	0	0	0	1
0010	0	1	0	0	0
0010	1	0	1	0	0
0011	–	0	0	0	1
0100	–	0	0	1	0
...					
1100	–	0	0	0	1
1101	–	–	–	–	–
111–	–	–	–	–	–
0000	–	–	–	–	–

Combinational example (cont'd)

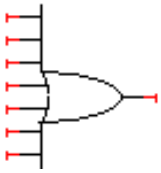
- d28 = m8'•m4'•m2•m1'•leap'
- d29 = m8'•m4'•m2•m1'•leap
- d30 = (m8'•m4•m2'•m1') + (m8'•m4•m2•m1') + (m8•m4'•m2'•m1) + (m8•m4'•m2•m1)
- d31 = (m8'•m4'•m2'•m1) + (m8'•m4'•m2•m1) + (m8'•m4•m2'•m1) + (m8'•m4•m2•m1) + (m8•m4'•m2'•m4') + (m8•m4'•m2•m1') + (m8•m4•m2'•m1')

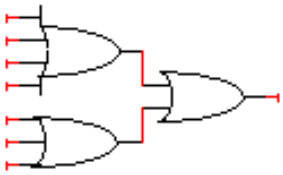




Combinational example (cont'd)

- d28 = m8'•m4'•m2•m1'•leap'
- d29 = m8'•m4'•m2•m1'•leap
- d30 = (m8'•m4•m2'•m1') + (m8'•m4•m2•m1') + (m8•m4'•m2'•m1) + (m8•m4'•m2•m1)
- d31 = (m8'•m4'•m2'•m1) + (m8'•m4'•m2•m1) + (m8'•m4•m2'•m1) + (m8'•m4•m2•m1) + (m8•m4'•m2'•m4') + (m8•m4'•m2•m1') + (m8•m4•m2'•m1')





Another example

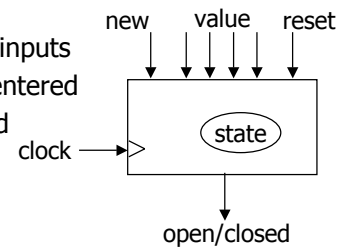
- Door combination lock:
  - punch in 3 values in sequence and the door opens; if there is an error the lock must be reset; once the door opens the lock must be reset
  - inputs: sequence of input values, reset
  - outputs: door open/close
  - memory: must remember combination
    or always have it available as an input

Implementation in software

```
integer combination_lock ( ) {
    integer v1, v2, v3;
    integer error = 0;
    static integer c[3] = 3, 4, 2;

    while (!new_value( ));
    v1 = read_value( );
    if (v1 != c[1]) then error = 1;

    while (!new_value( ));
    v2 = read_value( );
    if (v2 != c[2]) then error = 1;

    while (!new_value( ));
    v3 = read_value( );
    if (v2 != c[3]) then error = 1;

    if (error == 1) then return(0); else return (1);
}
```

Sequential Implementation

- Encoding:
  - how many values in sequence?
  - how do we know a new input value is entered?
  - how do we represent the states of the system?
- Behavior:
  - clock wire tells us when it's ok to look at inputs
  - sequential: sequence of values must be entered
  - sequential: remember if an error occurred
  - finite-state specification



Abstraction - FSM

- Finite-state diagram
  - States: 5 states
    - represent point in execution of machine
    - each state has outputs
  - Transitions: 6 from state to state, 5 self transitions, 1 global
    - changes of state occur when clock says it's ok
    - based on value of inputs
  - Inputs: reset, new, results of comparisons
  - Output: open/closed