

---

# Digital Logic Design

## ECEN 3233

### Module 1: Introduction

---

Dr. Louis Johnson  
School of Electrical and Computer Engineering  
Oklahoma State University

Fall 2007

---

# Why study logic design?

- Obvious reasons

- this course is part of the ECEN requirements
- it represents a fundamental set of design skills, but to truly understand something you must understand the fundamental principles
- it is the implementation basis for all modern computing devices
  - building large things from small components
  - provides a model of how a computer works
- you will gain valuable team and laboratory experience you will use later

- More reasons

- the inherent parallelism in hardware is often our first exposure to parallel computation
- it offers an interesting counterpoint to software design and is therefore useful in furthering our understanding of computation in general

# The Digital Revolution

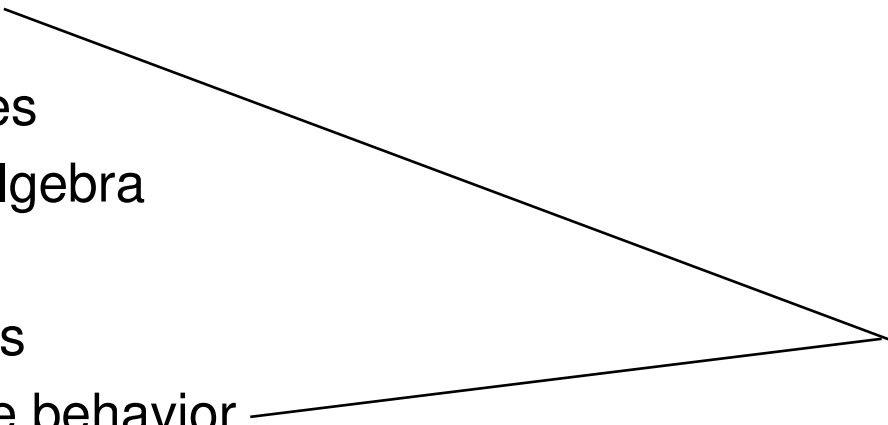
- Processing power
  - doubling every 18 months (Moore's Law)
  - a factor of about 100 every decade
- Storage capacity
  - doubling every 12 months
  - a factor of about 1000 every decade
- Optical communications capacity (fiber)
  - doubling every 9 months
  - a factor of about 10000 every decade
- These are some of the reasons that this area and Electrical Engineering and Computer Engineering, in general, are so exciting!

Moore's Law – The number of transistors on an integrated circuit doubles approximately every 18 months

# What will we talk about in this class?

- Basic physical electronic devices – but this isn't an electronics class
  - transistors, resistors, wires, LEDs, etc.
- The language of logic design
  - Boolean algebra, logic minimization, state, timing, CAD tools
- Representations of logic design
  - truth tables, timing diagrams, schematic diagrams
- Combinational logic
- Sequential logic
- The concept of state in digital systems
- How to specify/simulate/compile/realize (build) our designs (laboratory)
  - realization with discrete logic devices
  - realization with programmable logic devices
    - hardware description languages (very brief introduction)
    - tools to simulate the workings of our designs
    - logic compilers to synthesize the hardware blocks of our designs
    - mapping onto programmable hardware
- Contrast with software design
  - sequential and parallel implementations

# Representation of digital designs

- Physical devices (transistors, relays)
  - **Switches**
  - Truth tables
  - Boolean algebra
  - Gates
  - Waveforms
  - **Finite state behavior**
  - Register-transfer behavior
  - Concurrent abstract specifications
- scope of ECEN 3233
- 

# Digital Design is ...

- using digital logic –
  - which is described by Boolean Algebra, and
  - whose physical basis is transistor switches
  
- developing an efficient and effective solution to a problem –
  - within size, cost, power, and other constraints, and
  - within these constraints
    - represent as logic statements, and
    - implement with physical hardware
  
- using logical values encoded as physical quantities –
  - if ( $0V < voltage < 0.8V$ ) then *symbol* is a logical “0” (*false*)
  - if ( $2.0V < voltage < 5.0V$ ) then *symbol* is a logical “1” (*true*)
  - for example

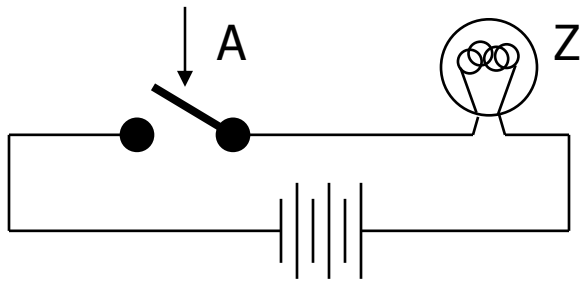
# ECEN 3233: concepts/skills/abilities

- Understanding the basics of logic design (concepts)
- Understanding sound design methodologies (concepts)
- Modern specification methods (concepts)
- Familiarity with a full set of CAD tools (skills)
- Realize digital designs in an implementation technology (skills)
- Appreciation for the differences and similarities (abilities) in hardware and software design

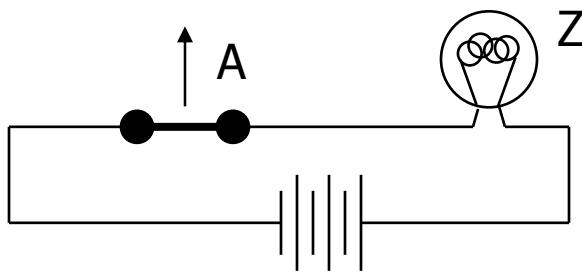
New abilities: (1) to design, build and debug logic circuits using discrete logic components, (2) to accomplish the logic design task with the aid of computer-aided design tools and map a problem description into an implementation with programmable logic devices after validation via simulation and (3) to develop a basic understanding of the advantages/disadvantages as compared to a software implementation

# Switches: basic element of physical implementations

- 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)

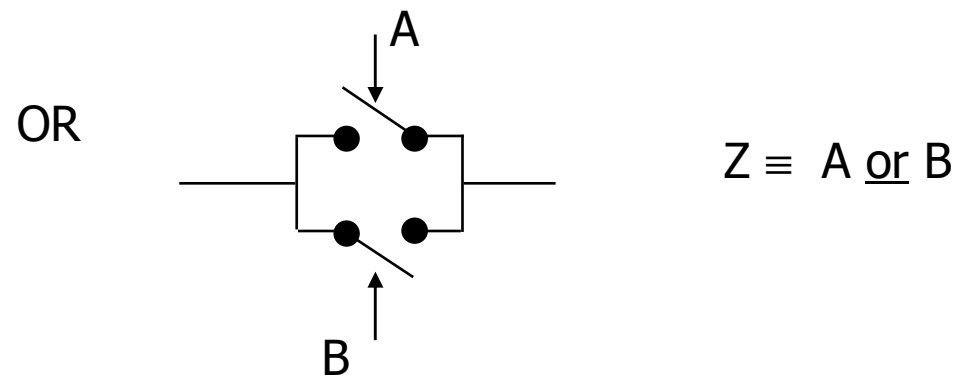
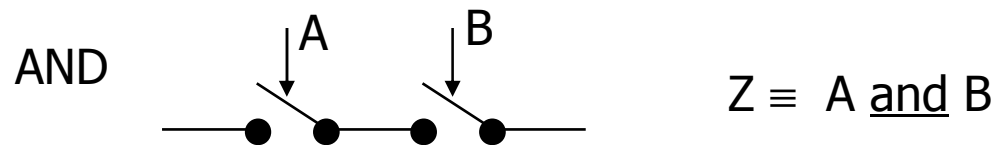


open switch (if A is "0" or unasserted)  
and turn off light bulb (Z)

$$Z \equiv A$$

## Switches (cont'd)

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



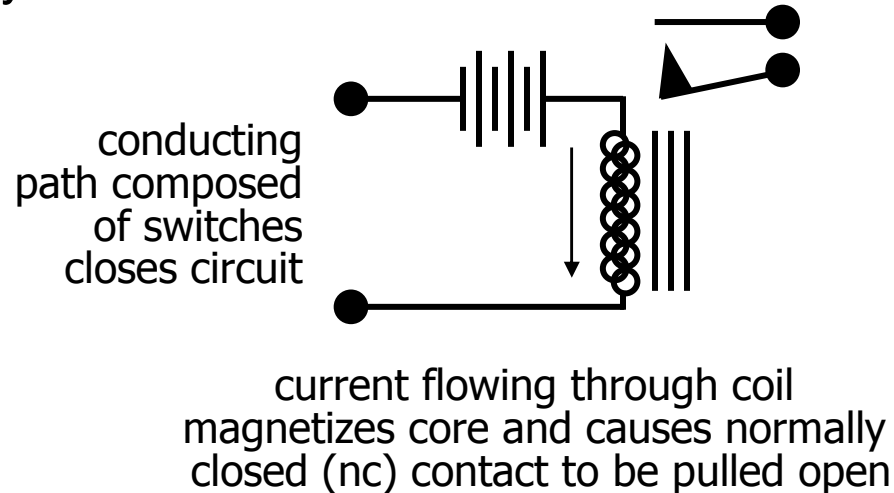
---

# Switching networks

- Switch settings
  - determine whether or not a conducting path exists to light the light bulb
- To build larger computations
  - use a light bulb (output of the network) to set other switches (the outputs of one circuit as the inputs to another network).
- Connect together switching networks
  - to construct larger switching networks, i.e., there is a way to connect outputs of one network to the inputs of the next.

# Relay networks

- A simple (and old) way to convert between conducting paths and switch settings is to use (electro-mechanical) relays.
- What is a relay?



when no current flows, the spring of the contact returns it to its normal position

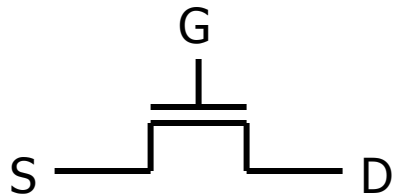
What determines the switching speed of a relay network?

# Transistor networks

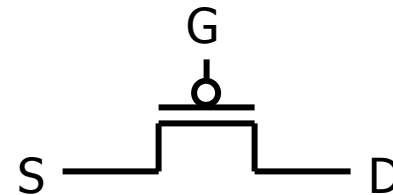
- Relays aren't used much anymore
  - slow and big
  - some traffic light controllers are still electro-mechanical (high power circuits)
- Modern digital systems are designed in CMOS technology
  - MOS stands for Metal-Oxide on Semiconductor
  - C is for complementary because there are both normally-open and normally-closed switches
- MOS transistors act as voltage-controlled switches
  - similar, though easier to work with than relays.

# MOS transistors

- MOS transistors have three terminals: drain, gate, and source
  - they act as switches in the following way:  
if the voltage on the gate terminal is (some amount) higher/lower than the source terminal then a conducting path will be established between the drain and source terminals



n-channel  
open when voltage at G is low  
closes when:  
 $\text{voltage}(G) > \text{voltage}(S) + \epsilon$



p-channel  
closed when voltage at G is low  
opens when:  
 $\text{voltage}(G) < \text{voltage}(S) - \epsilon$

---

# Speed of MOS networks

- What influences the speed of CMOS networks?
  - charging and discharging of voltages on wires and gates of transistors
- Capacitors hold charge
  - capacitance is at gates of transistors and wire material
- Resistors slow movement of electrons
  - resistance mostly due to transistors

---

# Digital vs. analog

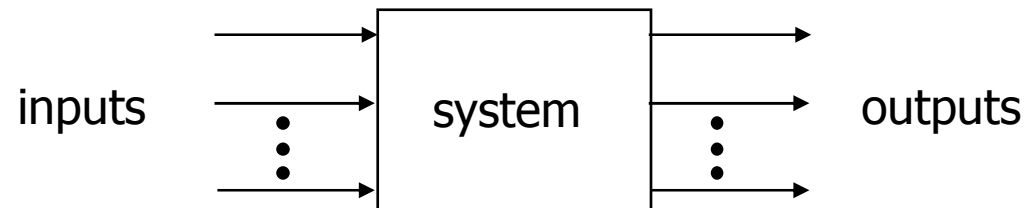
- Convenient to think of digital systems as having only discrete, digital, input/output values
- In reality, real electronic components exhibit continuous, analog, behavior
- Why do we make the digital abstraction anyway?
  - switches operate this way
  - easier to think about a small number of discrete values
- Why does it work?
  - does not propagate small errors in values
  - always resets to 0 or 1

# Mapping from physical world to binary world

<b>Technology</b>	<b>Logic 0</b>	<b>Logic 1</b>
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
Compact disc	No pit	Pit

# Combinational Logic

- 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



- AND, NAND



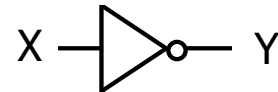
easy to implement  
with CMOS transistors  
(the switches we have  
available and use most)

- OR, NOR



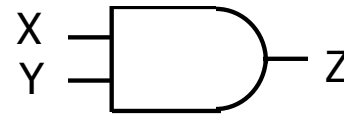
# Logic functions have many representations

■ NOT:  $X'$       $\bar{X}$       $\sim X$



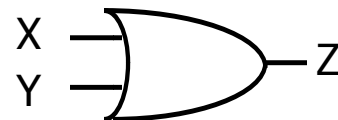
X	Y
0	1
1	0

■ AND:  $X \cdot Y$       $XY$       $X \wedge Y$



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

■ OR:  $X + Y$       $X \vee Y$



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

Boolean  
Algebra

Logic Symbol

Truth Table

# Sequential Logic

- Sequential systems
  - exhibit behaviors (output values) that depend not only on the current input values, but also on previous input values
  - Think of a “digital clock” – the next value depends on the previous one and the value of any inputs (“clock set” button, etc.)
- In reality, all real circuits are sequential
  - because 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 the outputs only after sufficient time has elapsed for the system to make its required changes and settle down

# Summary

- That was a brief introduction to a few topics in digital logic
- We will concentrate on
  - converting solutions to problems into combinational and sequential networks effectively organizing the design hierarchically
  - doing so with a modern set of design tools that lets us handle large designs effectively
  - taking advantage of optimization opportunities
- What do you need to do now?
  - Read Chapter 1: pp. 1-27
  - Be sure you can login to Desire2Learn and get into the ECEN 3233 site.
  - Review the web site: [ecen3233.okstate.edu](http://ecen3233.okstate.edu)
    - The Week 2 reading assignment should be completed before Tuesday's class
    - You will be notified on the announcements page when quizzes are due and exams are scheduled
  - Attend lab next week
- Now lets really get started with Combinational Logic