ELPL Embedded Low-Power Laboratory

Digital Logic Design

4190.201.001

2010 Spring Semester

1. Introduction

Naehyuck Chang
Dept. of EECS/CSE
Seoul National University
naehyuck@snu.ac.kr

About the lecturer

- Instructor: Professor Naehyuck Chang
  - 20 year hardware design experience
  - Founding and leading Embedded Low-Power Laboratory
  - IEEE Transactions CAD AE and two more international journal AE
  - Including DAC, ICCAD, ISLPED, etc., TPC member of top-tier conferences
  - ISLPED 2009 TPC Cochair, 2010 General Vice Chair and 2011 General Chair
  - ACM SIGDA Executive Committee
    - Technical Activity Chair
    - Low-Power Technical Committee Chair
  - Published more than 70 technical papers in low-power and embedded systems area
  - IEEE and ACM Senior Member
  - Contact:
    - Office: 301-506
    - Phone: (SNU) 1834
    - Email: naehyuck @ elpl.snu.ac.kr

About the teaching assistants

- Byeongho Gim (김병호)
  - Email: bhgim @ elpl.snu.ac.kr
  - Office: 301-551
  - Phone: (SNU) 1836
- Hyun-Jin Kim (김현진)
  - Email: hjkim @ elpl.snu.ac.kr
  - Office: 301-551
  - Phone: (SNU) 1836
- Young Shin Koh (고영신)
  - Email: yskoh @ elpl.snu.ac.kr
  - Office: 301-551
  - Phone: (SNU) 1836

Logic design laboratory course

- **MUST enroll both Digital Logic Design and Logic Design Laboratory at the same time**

No exception

No excuse

Course information

- Two 75 min classes a week
  - Tuesday and Thursday
  - 11:00 AM to 12:15 PM
- Laboratory class
  - Tuesday and Thursday
  - 7:00 PM to 10:00 PM
  - 302-310-2
- Language
  - English

What is design?

- **A plan or drawing produced to show the look and function**
  - Given a specification of a problem, come up with a way of solving it choosing appropriately from a collection of available components
  - While meeting some criteria for size, performance, cost, power, beauty, elegance, etc.
- Design a complicated system
  - Divide and conquer
- How to divide a big problem?
  - Functional dividing
  - Spatial dividing
- Other ways toward efficient design
  - Design reuse
  - Design methodology
    - To manage a process effectively and efficiently
- Design tools
  - Formal description and modeling
  - Systematic optimization
  - Mathematics

Logic design

- Digital circuits
  - Interconnected collections of switches to perform a desired function
- Major resources
  - Switches
  - Interconnects
    - Interconnections are more important in recent designs
- Digital abstraction
  - 0 or 1
  - Noise immunity
  - A group of signals represent analog quantity after quantization
- Choose a right components to solve the problem
  - Constraints: speed, power, area, etc.
- Types of circuits
  - Combinational circuits
  - Sequential circuits
- Logic design is a set of abstractions and methodologies
  - Devise, understand, and manipulate large collections of digital circuits

Logic design

- What is logic design?
  - Determining the collection of digital logic components to perform a specified control and/or data manipulation and/or communication function and the interconnections between them
  - Which logic components to choose?
    - There are many implementation technologies (e.g., off-the-shelf fixed-function components, programmable devices, transistors on a chip, etc.)
  - The design may need to be optimized and/or transformed to meet design constraints

Why study logic design?

- Obvious reasons
  - Fundamentals of computer system hardware and software design
  - We live in the Electric Age (neither Stone Age, nor Bio Age)
    - Implementation basis for all modern computing devices (digital semiconductor devices)
  - Theory: target independent
    - Provide a model of how a computer works
  - Practice: target dependent
    - Building large things from small components
- More important 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
  - **Neither hardware, nor software cannot be a system alone**
    - **To have a leading position, you must know both**
    - **We have plenty of software course and interest though**

What will we learn in this class?

- Languages of logic design
  - Boolean algebra, logic minimization, state, timing, CAD tools, and so on
  - Discrepancy between programming languages and hardware description
- The concept of state in digital systems
  - Analogous to variables and program counters in software systems
- How to specify/simulate/realize our designs
  - Schematic design
    - No hardware description languages are used except for Boolean equations
  - Tools to simulate the workings of our designs
  - Contemporary logic design and implementation methods
  - Prototype implementation and debugging concepts/techniques
- Contrast with software design
  - Sequential and parallel implementations
  - Specify algorithm as well as computing/storage resources it will use
  - Understanding the differences between theory and practice
    - Target devices are not ideal
    - More sensitive to the optimal design, i.e., cost

Coverage of this course

- Mandatory knowledge and experience to understand logic design and implementation
  - Intermediate goal: optimal design of combinational circuits
  - Final goal: synchronous state machines
- CAD based front-end design
  - The same CAD tool as used in Computer System Design: Xilinx ISE
  - Schematic and Boolean equation entry
  - Logic simulation
- No logic synthesis
  - Until a concrete concept of logic design is established
  - Computer System Design will cover logic synthesis using Verilog HDL
- **Legacy component based design**
  - TTL 74 family or equivalent
  - Boolean equations with PAL
- **Implementation and debugging techniques**
  - Lab course

Applications of logic design

- Conventional computer design
  - CPUs, busses, peripherals, storages and so forth
- Networking and communications
  - Phones, modems, routers, sensors and so forth
- **Embedded products**
  - In cars, toys, appliances, factories, aircrafts, spaceships, entertainment devices, robots, and so forth
- Scientific equipment
  - Testing, sensing, reporting, and so forth
- The world of computing is much much bigger than just PCs!
  - Computer engineering is not only programming with a PC or a server environment
  - **All the modern devices, systems, equipments fall in computer engineering**
  - Logic design course is the first step toward computers and their applications

A quick history lesson

- 1850: George Boole invents Boolean algebra
  - Maps logical propositions to symbols
  - Permits manipulation of logic statements using mathematics
- 1938: Claude Shannon links Boolean algebra to switches
  - His Masters' thesis
- 1945: John von Neumann develops the first stored program computer
  - Its switching elements are vacuum tubes (a big advance from relays)
- 1946: ENIAC... The world's first completely electronic computer
  - 18,000 vacuum tubes
  - Several hundred multiplications per minute
- 1947: Shockley, Brittain, and Bardeen invent the transistor
  - Replaces vacuum tubes
  - Enable integration of multiple devices into one package
  - Gateway to modern electronics

Contemporary logic design

- Our systems are becoming more complex
  - Integrate more functions

- Very narrow market windows and short design times
  - Cell phone market window
- Digital hardware is becoming so cheap

Insignia DVD Player with MP3 Playback/JPEG Viewer
Enjoy your favorite DVDs in widescreen with this **DVD player** that lets you view Kodak Picture CDs to create slideshows on your TV.
Add to Shopping List
$26.99
Best Buy
4,961 seller ratings

Philips DVD/DivX Player
Key Features: Progressive scan,DivX playback,Tuner not included,Slim design,Remote control with batteries,A/V cable ... Keywords: **dvd player**.
Add to Shopping List
$36.99
Circuit City
6,136 seller ratings

Digital and analog signals

- 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

Digital and analog signal

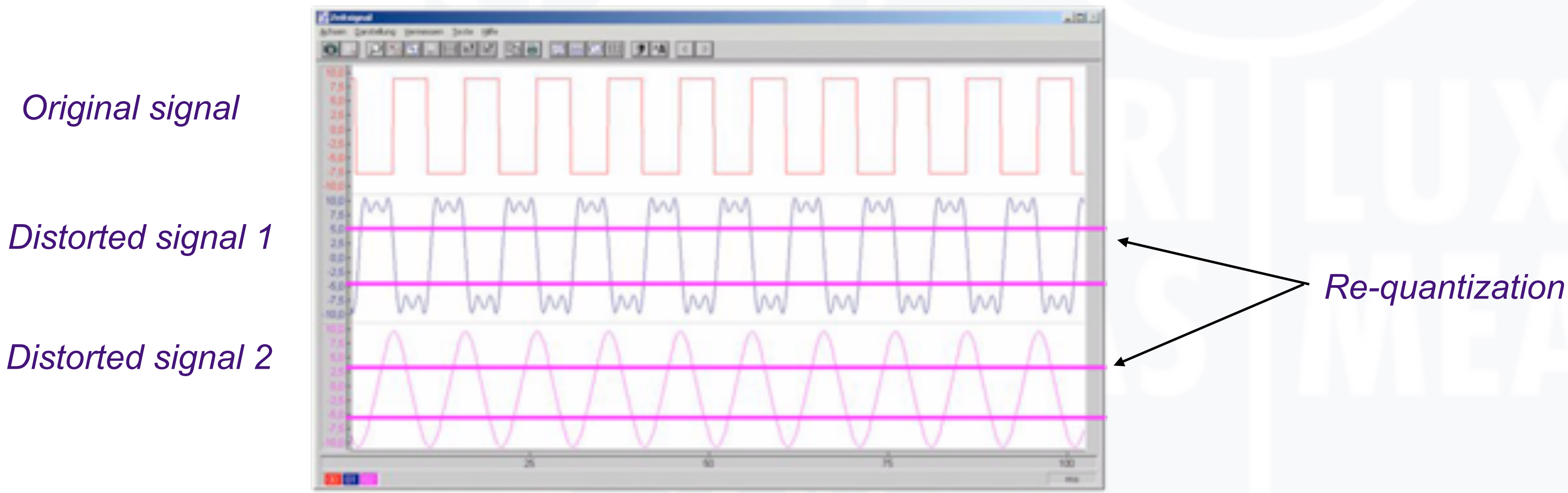
- Continuous time and discrete time signals
  - $Y=f(t)$: t is a continuous variable
  - $Y=F(t_i)$: t_i is an index
  - Sampling converts continuous time signal to discrete time signal
- Digital signal
  - Logic 0 and logic 1
  - A group of digital signals can represent analog quantity by quantization
- Analog signals are converted to digital signals via sampling and quantization

Digital and analog signal

- Analog signals
  - Continuous time signals
  - Continuous values
  - Natural signals
  - Noise prone
    - Additive noise permanently changes the original signal

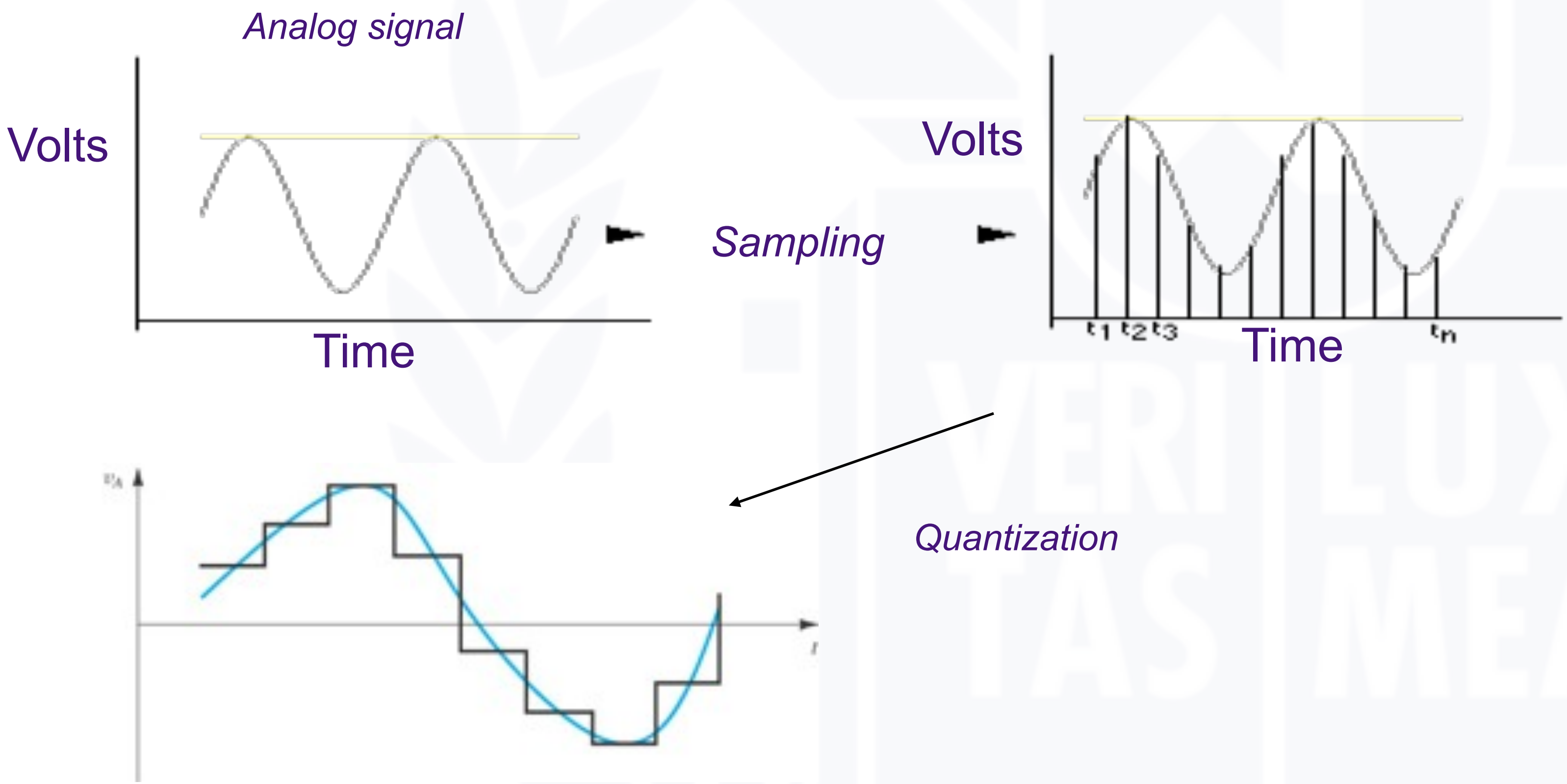
Digital and analog signal

- Digital signals
  - Discrete time signals
  - Discrete values
  - Not an original signal
    - Sampling and quantization error Noise immunity within noise margin
    - Additive noise can be canceled



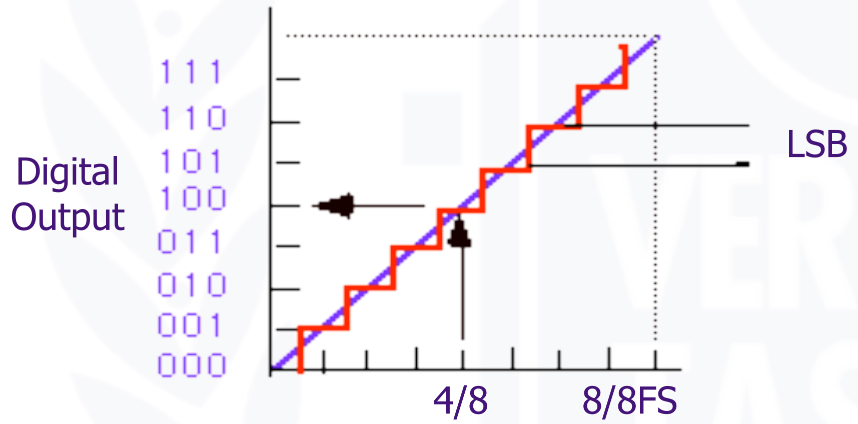
Digital and analog signal

- Signal translations



Digital and analog signal

- Resolution: The number of bits the ADC uses to represent the digital data determines the resolution
  - Example: A 3-bit ADC. For a full-scale voltage of 10 volts, the resolution will be 10/(23)=1.25 V.
  - However, if we increase the number of bits to 12, then resolution becomes 10/(212)=2.44 mV.



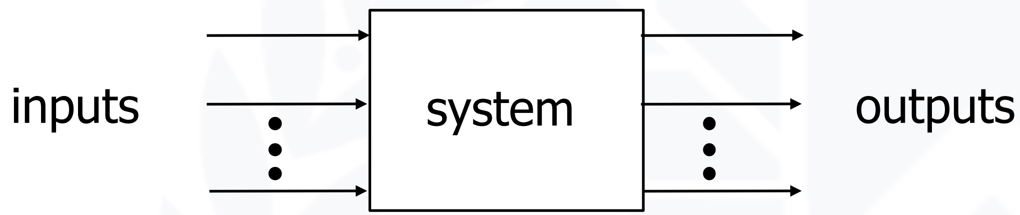
Embedded Low-Power Laboratory

Mapping from physical world to binary 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

Combinational circuit

- 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

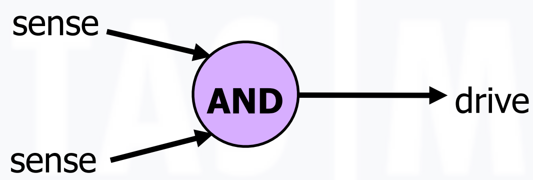
  - AND, NAND

  - OR, NOR

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

What is digital hardware?

- Collection of devices that sense and/or control wires that carry a digital value (i.e., a physical quantity that can be interpreted as a "0" or "1")
  - Example: digital logic where voltage < 0.8 V is a "0" and > 2.0 V is a "1"
  - Example: pair of transmission wires where a "0" or "1" is distinguished by which wire has a higher voltage (differential)
  - Example: orientation of magnetization signifies a "0" or a "1"
- Primitive digital hardware devices
  - Logic computation devices (sense and drive)
    - Are two wires both "1" - make another be "1" (AND)
    - Is at least one of two wires "1" - make another be "1" (OR)
    - Is a wire "1" - then make another be "0" (NOT)
  - Memory devices (store)
    - Store a value
    - Recall a previously stored value



What is happening now in digital design?

- Important trends in how industry does hardware design
  - Larger and larger designs
  - Shorter and shorter time to market and market window
  - Cheaper and cheaper products
- To optimize
  - Scale
    - Pervasive use of computer-aided design tools over hand methods
    - Multiple levels of design representation
  - Time
    - Emphasis on abstract design representations
    - Programmable rather than fixed function components
    - Automatic synthesis techniques
    - Importance of sound design methodologies
  - Cost
    - Higher levels of integration
    - Use of simulation to debug designs
    - Simulate and verify before you build

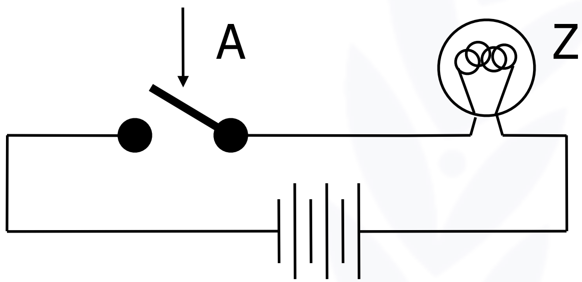
Computation: abstract vs. implementation

- Up to now, computation has been a mental exercise (paper, programs)
- This class is about physically implementing computation using physical devices that use voltages to represent logical values
- Basic units of computation are:
  - representation: "0", "1" on a wire
set of wires (e.g., for binary ints)
  - assignment: x = y
  - data operations: x + y – 5
  - control:

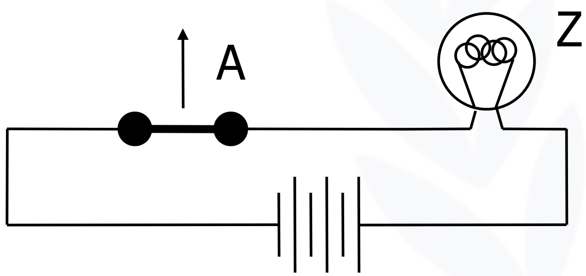
    sequential statements: A; B; C
    conditionals: if x == 1 then y
    loops: for (i = 1 ; i == 10, i++)
    procedures: A; proc(...); B;
- We will study how each of these are implemented in hardware and composed into computational structures

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)



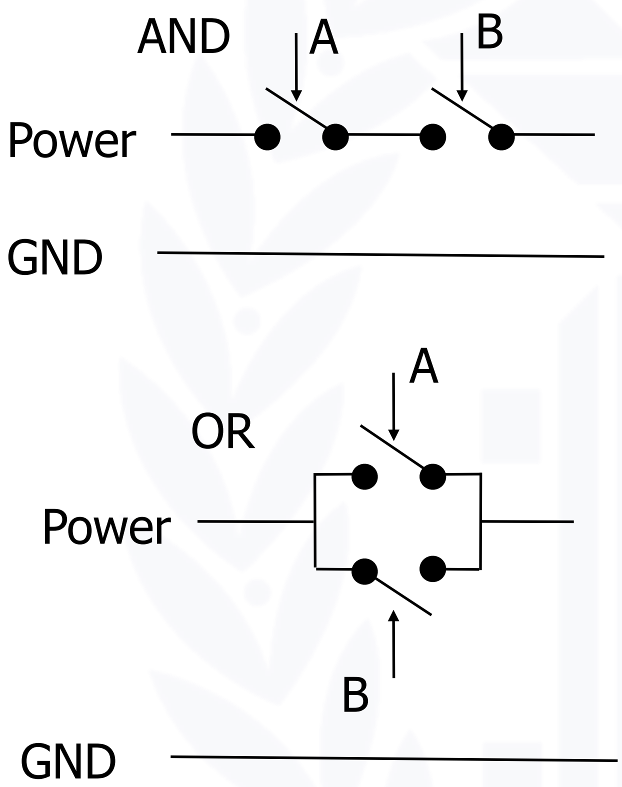
open switch (if A is "0" or unasserted)
and turn off light bulb (Z)

$Z \equiv A$

Embedded Low-Power Laboratory

Switches (cont'd)

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



Z ≡ A and B

Z ≡ A or B

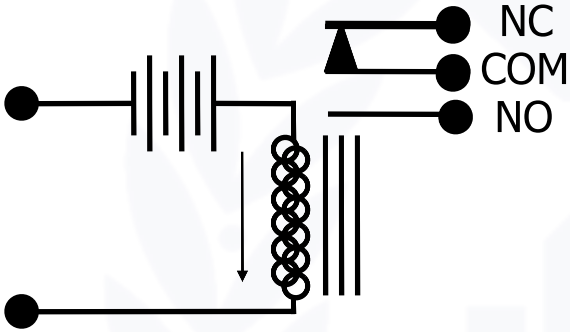
Switching networks

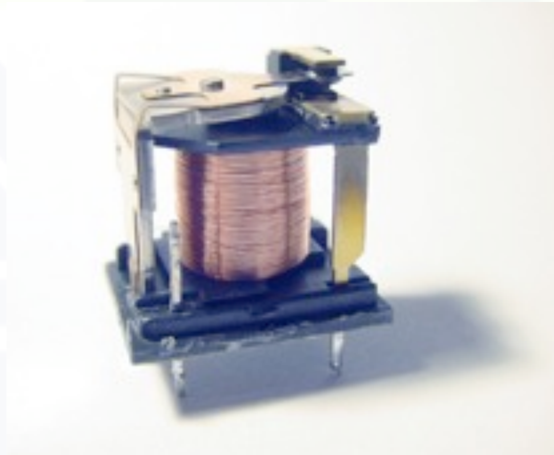
- Switch settings
  - determine whether or not a conducting path exists to light the light bulb
- To build larger computations
  - use a light bulb (output of the network) to set other switches (inputs to another network).
- Connect together switching networks
  - to construct larger switching networks,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

Relay networks

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

conducting
path composed
of switches
closes circuit





current flowing through coil
magnetizes core and causes normally
closed (nc) contact to be pulled open

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

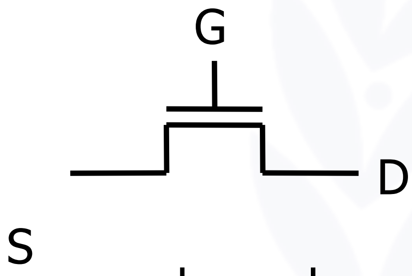
- What determines the switching speed of a relay network?

Transistor networks

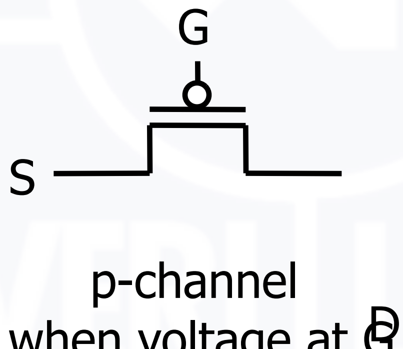
- Relays are seldom used much anymore
  - Some traffic light controllers are still electro-mechanical
  - Programmable logic controller (PLC)
    - Ladder network
- Modern digital systems are designed in CMOS technology
  - MOS stands for Metal-Oxide on Semiconductor
  - C is for complementary because there are both normally-open and normally-closed switches
- MOS transistors act as voltage-controlled switches
  - Similar, though easier to work with than relays
  - Small, low-power and fast

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:
voltage(G) > voltage (S) + ε



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

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
- Threshold voltage determines the switching speed and power consumption
  - High threshold is low power and low speed
  - Low threshold is high power and high speed

Representation of digital designs

- Physical devices (transistors, relays)
- Switches
- Truth tables
- Boolean algebra
- Gates
- Waveforms
- Finite state behavior
- Register-transfer behavior
- Concurrent abstract specifications
- Prototype implementation
- Design practice

scope of 4190.201 (Physical devices through Concurrent abstract specifications)

scope of the lab (Prototype implementation, Design practice)