

# CSE 369: Introduction to Digital Design

---

- ❖ Professor Georg Seelig, CSE 228  
([gseelig@uw.edu](mailto:gseelig@uw.edu))
  - ❖ Office Hours: email w/schedule for a slot
- ❖ Book: Brown & Vranesic *Fundamentals of Digital Logic with Verilog Design* (3<sup>rd</sup> Edition)
- ❖ TAs:
  - ❖ Ta-Tung Yen ([ttyen@uw.edu](mailto:ttyen@uw.edu))
  - ❖ Sixto Josue Rios ([jrios777@uw.edu](mailto:jrios777@uw.edu))
- ❖ Lab Hours: TBA (check website)

# Grading

---

- ❖ 70% - Labs
- ❖ 10% - Quizzes
- ❖ 20% - Final Exam
- ❖ Late penalties for uploading lab materials:
  - ❖ <24 hours: -10%
  - ❖ <48 hours: -30%
  - ❖ <72 hours: -60%
  - ❖ >72 hours: not accepted

# Joint Work Policy

---

- ❖ Labs will be done alone
- ❖ Students may not collaborate on labs/projects, nor between groups on the specifics of homeworks.
- ❖ OK:
  - ❖ Studying together for exams
  - ❖ Discussing lectures or readings
  - ❖ Talking about general approaches
  - ❖ Help in debugging, tools peculiarities, etc.
- ❖ Not OK:
  - ❖ Developing a lab together
- ❖ Violation of these rules is grounds for failing the class

# Class & Lab Meetings

---

- ❖ Labs:
  - ❖ Each student assigned a lab kit, can work where-ever.
  - ❖ In addition to the official sections, TAs will have some blocks of office hours to help with labs, etc.
  - ❖ Signups for lab demos will be posted shortly.
- ❖ Quiz: Tue, Oct 27 and Tue, Nov 24 in class
- ❖ Final: Mon, June 8, 8:30-10:20

# Motivation

---

- ❖ Readings: 1-1.4, 2-2.4
- ❖ Electronics an increasing part of our lives
  - ❖ Computers & the Internet
  - ❖ Car electronics
  - ❖ Robots
  - ❖ Electrical Appliances
  - ❖ Cellphones
  - ❖ Portable Electronics
- ❖ Class covers digital logic design & implementation

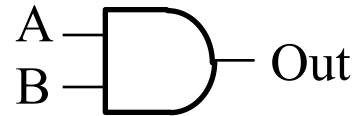




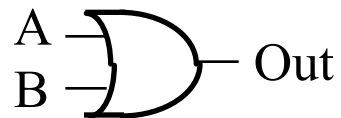
# Basic Logic Gates

---

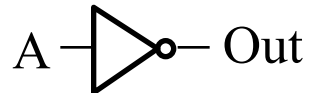
- ❖ AND: If A and B are True, then Out is True



- ❖ OR: If A or B is True, or both, then Out is True



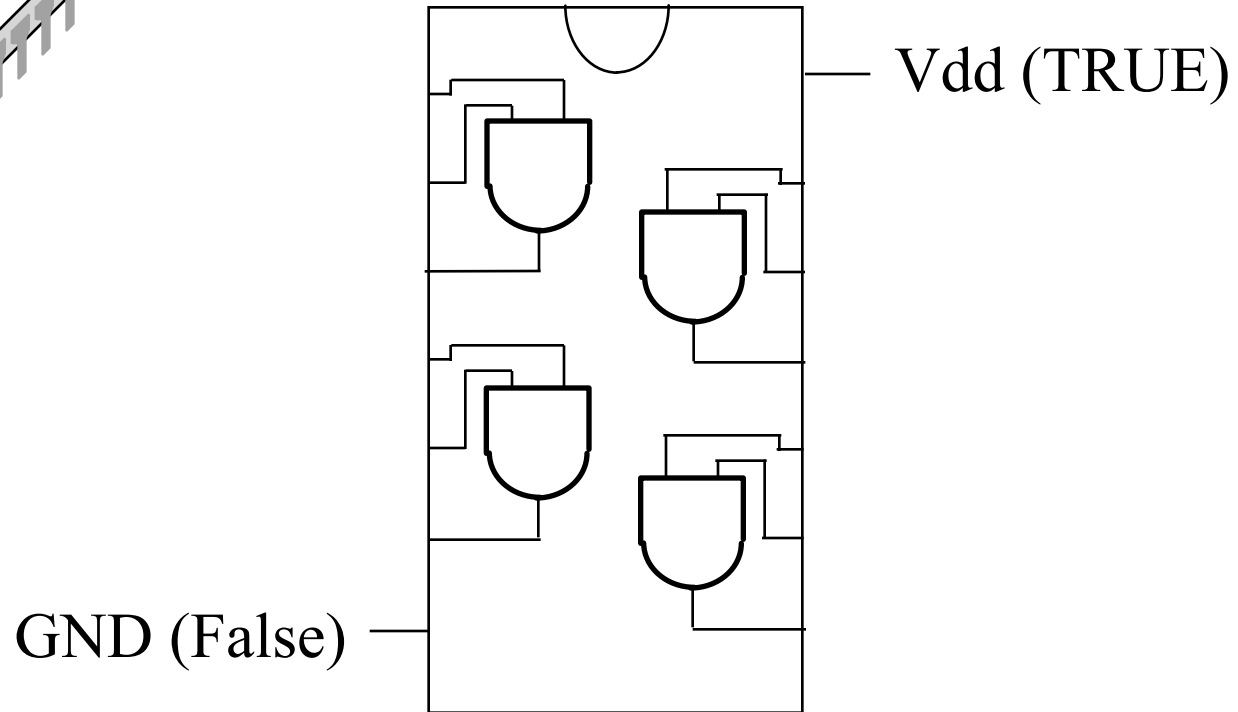
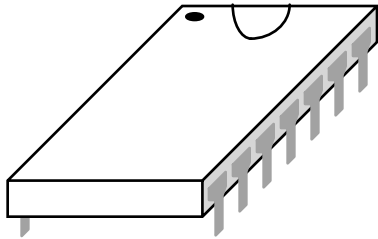
- ❖ Inverter (NOT): If A is False, then Out is True





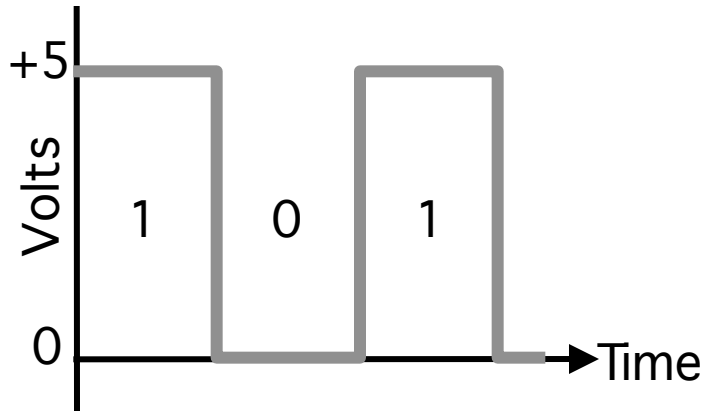
# TTL Logic

---



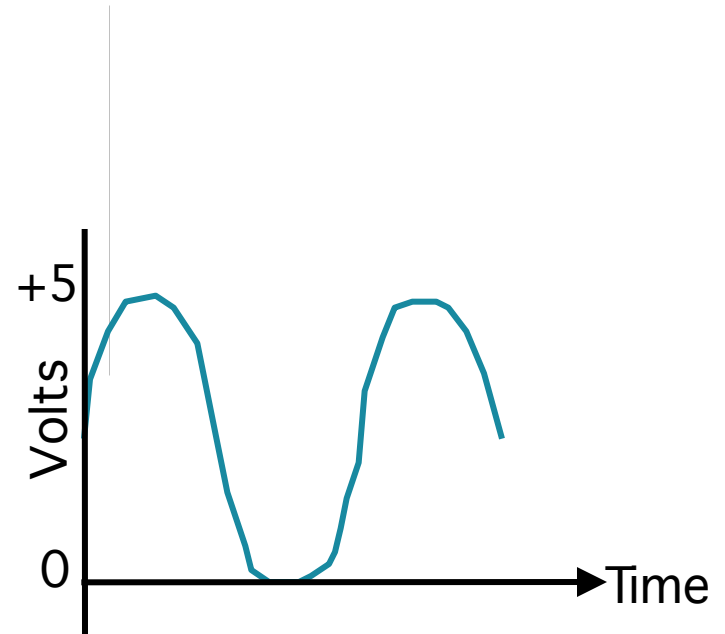
# Digital vs. Analog

---



**Digital:**  
only assumes discrete values

**Binary/Boolean (2 values)**  
yes, on, 5 volts, high, TRUE, "1"  
no, off, 0 volts, low, FALSE, "0"



**Analog:**  
values vary over a broad range  
continuously

# Advantages of Digital Circuits

---

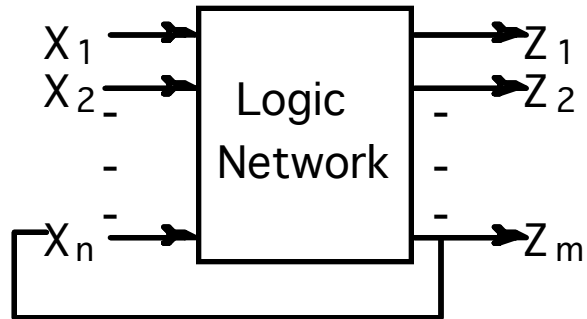
- ❖ Analog systems: slight error in input yields large error in output
- ❖ Digital systems more accurate and reliable
  - ❖ Readily available as self-contained, easy to cascade building blocks
- ❖ Computers use digital circuits internally
- ❖ Interface circuits (i.e., sensors & actuators) often analog

***This course is about logic design, not system design (processor architecture), not circuit design (transistor level)***

# Combinational vs. Sequential Logic

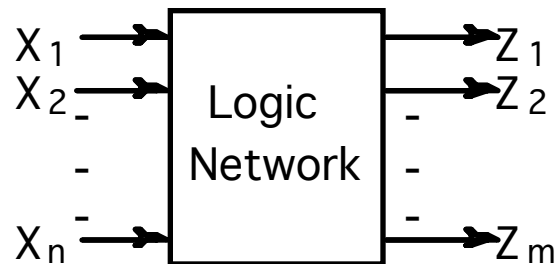
---

## *Sequential logic*



**Network implemented from logic gates. The presence of feedback distinguishes between *sequential* and *combinational* networks.**

## *Combinational logic*



**No feedback among inputs and outputs. Outputs are a function of the inputs only.**

# Boolean Elements and truth tables

---

**Algebra:** variables, values, operations

In Boolean algebra, the values are the symbols 0 and 1

If a logic statement is false, it has value 0

If a logic statement is true, it has value 1

**Operations:** AND, OR, NOT

X	Y	X AND Y
0	0	
0	1	
1	0	
1	1	

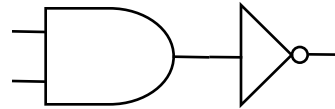
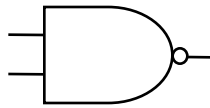
X	NOT X
0	
1	

X	Y	X OR Y
0	0	
0	1	
1	0	
1	1	

# NAND and NOR Gates

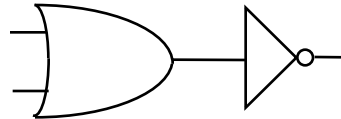
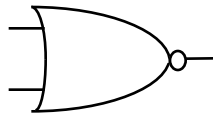
---

## ■ NAND Gate: NOT(AND(A, B))



X	Y	X NAND Y
0	0	1
0	1	1
1	0	1
1	1	0

## ■ NOR Gate: NOT(OR(A, B))

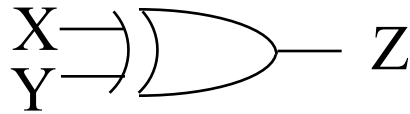


X	Y	X NOR Y
0	0	1
0	1	0
1	0	0
1	1	0

# XOR and XNOR Gates

---

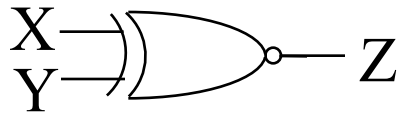
- XOR Gate:  $Z=1$  if  $X$  is different from  $Y$



$$X \oplus Y$$

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

- XNOR Gate:  $Z=1$  if  $X$  is the same as  $Y$



$$\overline{X \oplus Y}$$

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

# Boolean Equations

---

## *Boolean Algebra*

values: 0, 1

variables: A, B, C, . . . , X, Y, Z

operations: NOT, AND, OR, . . .

NOT X is written as  $\bar{X}$

X AND Y is written as  $X * Y$ , or sometimes  $X Y$  or  $X \& Y$

X OR Y is written as  $X + Y$

*Deriving Boolean equations from truth tables:*

A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Carry =

OR'd together *product* terms  
for each truth table  
row where the function is 1

if input variable is 0, it appears in  
complemented form;  
if 1, it appears uncomplemented

Sum =



# Boolean Algebra/Logic Minimization

---

$$\bar{A}BC_{in} + A\bar{B}C_{in} + AB\bar{C}_{in} + ABC_{in} \text{ vs. } AB + AC_{in} + BC_{in}$$

**Logic Minimization: reduce complexity of the gate level implementation**

- **reduce number of literals (gate inputs)**
- **reduce number of gates**
- **reduce number of levels of gates**

**fewer inputs implies faster gates in some technologies**

**fan-ins (number of gate inputs) are limited in some technologies**

**fewer levels of gates implies reduced signal propagation delays**

**number of gates (or gate packages) influences manufacturing costs**

# Basic Boolean Identities:

---

$$X + 0 = X$$

$$X * 1 = X$$

$$X + 1 = 1$$

$$X * 0 = 0$$

$$X + X = X$$

$$X * X = X$$

$$X + \bar{X} = 1$$

$$X * \bar{X} = 0$$

$$\overline{\bar{X}} = X$$

# Basic Laws

---

Commutative Law:

$$X + Y = Y + X$$

$$XY = YX$$

Associative Law:

$$X+(Y+Z) = (X+Y)+Z$$

$$X(YZ)=(XY)Z$$

Distributive Law:

$$X(Y+Z) = XY + XZ$$

$$X+YZ = (X+Y)(X+Z)$$

# Advanced Laws (Absorbtion)

---

■  $X + XY =$

■  $XY + X\bar{Y} =$

■  $X + \bar{X}Y =$

■  $X(X + Y) =$

■  $(X + Y)(X + \bar{Y}) =$

■  $X(\bar{X} + Y) =$

# Boolean Manipulations (cont.)

---

■ Boolean Function:  $F = \overline{X}YZ + XZ$

Truth Table:

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

Reduce Function:

# DeMorgan's Law

$$\overline{(X + Y)} = \bar{X} * \bar{Y}$$

X	Y	$\bar{X}$	$\bar{Y}$	$\overline{X+Y}$	$\bar{X} \cdot \bar{Y}$
0	0	1	1		
0	1	1	0		
1	0	0	1		
1	1	0	0		

$$\overline{(X * Y)} = \bar{X} + \bar{Y}$$

X	Y	$\bar{X}$	$\bar{Y}$	$\overline{X \cdot Y}$	$\bar{X} + \bar{Y}$
0	0	1	1		
0	1	1	0		
1	0	0	1		
1	1	0	0		

**DeMorgan's Law can be used to convert AND/OR expressions to OR/AND expressions**

**Example:**

$$Z = \bar{A} \bar{B} C + \bar{A} B C + A \bar{B} C + A B \bar{C}$$

$$\bar{Z} = (A + B + \bar{C}) * (A + \bar{B} + \bar{C}) * (\bar{A} + B + \bar{C}) * (\bar{A} + \bar{B} + C)$$

# DeMorgan's Law example

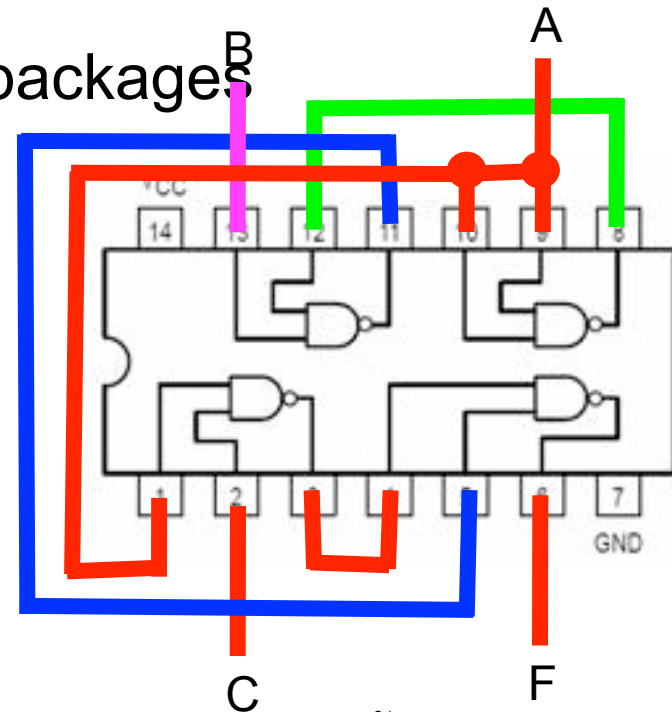
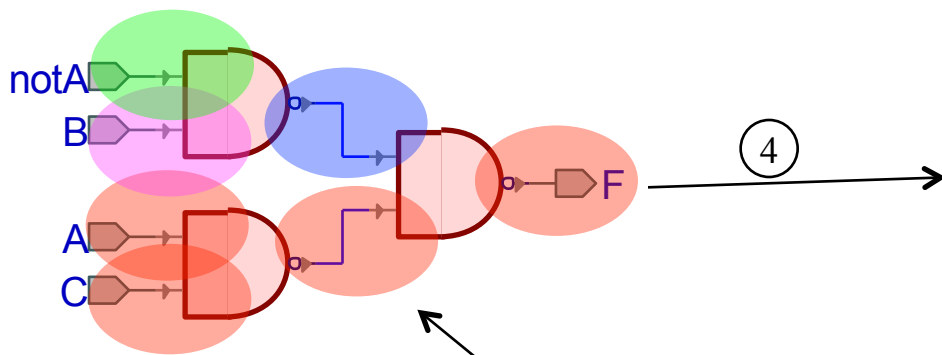
---

■ If  $F = (XY+Z)(\bar{Y}+\bar{X}Z)(X\bar{Y}+\bar{Z})$ ,

$$\bar{F} =$$

# Mapping truth tables to logic gates

- Given a truth table:
  - Write the Boolean expression
  - Minimize the Boolean expression
  - Draw as gates
  - Map to available gates
  - Determine number of packages and their connections



7 nets (wires)  
in this design



# Breadboarding circuits

